JON BONSO AND ADRIAN FORMARAN

AWS CERTIFIED DEVELOPER ASSOCIATE EXAM

Tutorials Dojo
Study Guide and Cheat Sheets

TABLE OF CONTENTS

INTRODUCTION 5

AWS CERTIFIED DEVELOPER ASSOCIATE EXAM OVERVIEW 6

AWS CERTIFIED DEVELOPER ASSOCIATE EXAM - STUDY GUIDE AND TIPS 9

- Study Materials 9
- AWS Services to Focus On 11
- Common Exam Scenarios 13
- Validate Your Knowledge 16
  - Sample Practice Test Questions: 17
  - Question 1 17
  - Question 2 19

AWS CHEAT SHEETS 22

- AWS OVERVIEW 22
  - AWS Global infrastructure 22
  - AWS Pricing 25
  - AWS Well-Architected Framework - Five Pillars 27
  - AWS Well-Architected Framework - Design Principles 30
  - AWS Well-Architected Framework - Disaster Recovery 35
- COMPUTE 37
  - Amazon EC2 39
  - AWS Elastic Beanstalk 51
  - AWS Lambda 57
  - Amazon Elastic Container Service (ECS) 64
    - ECS Task Placement Strategies 69
  - AWS Serverless Application Model (SAM) 72
- STORAGE 75
  - Amazon S3 75
  - Amazon S3 Glacier 91
  - Amazon EBS 95
  - Amazon EFS 103
- DATABASE 108
  - Amazon Aurora 109
  - Amazon Relational Database Service (RDS) 117
  - Amazon DynamoDB 129

DynamoDB Scan vs Query 146
AWS Lambda Integration with Amazon DynamoDB Streams 148
Calculating the Required Read and Write Capacity Unit for your DynamoDB Table 151
Amazon Elasticache 154
Amazon Redshift 161
NETWORKING AND CONTENT DELIVERY 165
Amazon API Gateway 165
How to Invalidate API Gateway Cache 170
Amazon CloudFront 172
AWS Elastic Load Balancing 180
Amazon Route 53 189
Amazon VPC 201
AWS Transit Gateway 216
SECURITY AND IDENTITY 216
AWS Identity and Access Management (IAM) 217
AWS Key Management Service (AWS KMS) 224
Working with Customer Master Keys (CMKs) using the AWS KMS API 227
Amazon Cognito 231
Amazon Cognito User Pools vs Identity Pools 235
AWS WAF 242
AWS Secrets Manager 246
Amazon Inspector 249
MANAGEMENT 253
AWS Auto Scaling 254
AWS CloudFormation 261
AWS CloudTrail 264
Amazon CloudWatch 266
AWS Systems Manager 271
AWS BILLING AND COST MANAGEMENT 277
DEVELOPER 279
AWS CodeDeploy 281
AWS CodePipeline 289
AWS CodeBuild 292
AWS CodeCommit 296
AWS CodeCommit Repository 298
AWS X-Ray 305
Instrumenting your Application with AWS X-Ray 308

ANALYTICS 312
Amazon Kinesis 312
Kinesis Scaling, Resharding and Parallel Processing 323
APPLICATION 326
Amazon SQS 327
Amazon SNS 332
Amazon SWF 336
AWS Step Functions 340

COMPARISON OF AWS SERVICES 344
S3 vs EBS vs EFS 344
Amazon S3 vs Glacier 347
S3 Standard vs S3 Standard-IA vs S3 One Zone-IA vs S3 Intelligent Tiering 348
AWS DataSync vs Storage Gateway 348
RDS vs DynamoDB 350
RDS vs Aurora 352
CloudTrail vs CloudWatch 357
Security Group vs NACL 358
EBS-SSD vs HDD 360
Elastic Beanstalk vs CloudFormation vs OpsWorks vs CodeDeploy 364
Global Secondary Index vs Local Secondary Index 365
S3 Pre-Signed URLS vs CloudFront Signed URLS vs Origin Access Identity 368
CloudWatch Agent vs SSM Agent vs Custom Daemon Scripts 370
Amazon SWF vs AWS Step Function vs Amazon SQS 370
Application Load Balancer vs Network Load Balancer vs Classic Load Balancer vs Gateway Load Balancer 372
S3 Transfer Acceleration vs Direct Connect vs VPN vs Snowball vs Snowmobile 376
EC2 Container Services ECS vs Lambda 379
SNI Custom SSL vs Dedicated IP Custom SSL 379
EC2 Instance Health Check vs ELB Health Check vs Auto Scaling and Custom Health Check 381
Multi-AZ deployments vs. Multi-Region deployments vs. Read Replicas 382

FINAL REMARKS AND TIPS 382

ABOUT THE AUTHORS 383

INTRODUCTION

The trend of emerging technologies that have a large impact on industries is a common theme in today's headlines. More and more companies are adopting newer technologies that will allow them to grow and increase returns. The Amazon Web Services (AWS) Cloud is one example of this. As of now, there are more than a million active users of Amazon Web Services. These users are a mix of small businesses, independent contractors, large enterprises, developers, students, and many more. There is no doubt that AWS already has its own community of users, and the number of newcomers is increasing rapidly each day.

With AWS, you don't have to purchase and manage your own infrastructure. This is a huge cost savings for your company, and it is a sigh of relief as well for your sysadmin team. Although AWS offers a huge collection of services and products that require little configuration, they also offer traditional ones like VMs and storage devices that work similar to their physical counterparts. This means that transitioning from an on-premises type of environment to an AWS virtualized environment should be simple and straightforward.

We think that developers benefit the most from the cloud. You have access to your applications from anywhere as long as you have a browser and an Internet connection (and maybe an SSH/RDP client too). You can spin up machines in a matter of minutes from a catalog of OS and versions that suit your needs. You also have control over your storage devices, to which you can also provision and unprovision easily. Aside from traditional VMs, AWS also has other options for you such as container instances, batch instances, and serverless models that can easily be integrated with APIs. You also do not need to worry about databases. You can host them on your own in VMs with your own licenses or use AWS-provided licenses, or you can even use a managed database so you do not need to worry about infrastructure. If these are too much for a simple app that you just want to test, AWS has a platform-as-a-service solution wherein you can simply deploy your code and the service handles the infrastructure for you. In essence, AWS has already provided the tools that their users need to quickly take advantage of the cloud.

With all of the benefits presented in using AWS, the true value of being an AWS Certified Developer is having your knowledge and skills in developing in AWS recognized by the industry. With this recognition, you'll gain more opportunities for career growth, financial growth, and personal growth as well. Certified individuals are able to negotiate for higher salaries, aim for promotions, or land in higher job positions. Becoming an AWS Certified Developer Associate is also a great way to start on your journey in DevOps, which is one of the most in-demand and well-compensated roles in the IT industry today. This certificate is a must have in your portfolio if you wish to progress your career in AWS.

Note: We took extra care to come up with these study guides and cheat sheets, however, this is meant to be just a supplementary resource when preparing for the exam. We highly recommend working on **hands-on sessions** and **practice exams** to further expand your knowledge and improve your test taking skills

AWS CERTIFIED DEVELOPER ASSOCIATE EXAM OVERVIEW

In 2013, Amazon Web Services (AWS) began the Global Certification Program with the primary purpose of validating the technical skills and knowledge for building secure and reliable cloud-based applications using the AWS platform. By successfully passing the AWS exam, individuals can prove their AWS expertise to their current and future employers. The AWS Certified Solutions Architect - Associate exam was the first AWS certification that was launched, followed by other two role-based certifications: Systems Operations (SysOps) Administrator and Developer Associate later that year.

Exam Details

The AWS Certified Developer - Associate examination is intended for individuals who perform a development role and have one or more years of hands-on experience developing and maintaining an AWS-based application.

Exam Code:	DVA-C01
No. of Questions:	65
Score Range:	100/1000
Cost:	150 USD (Practice exam: 20 USD)
Passing Score:	720/1000
Time Limit:	2 hours 10 minutes (130 minutes)
Format:	Scenario based. Multiple choice/multiple answers
Delivery Method:	Testing center or online proctored exam

Exam Domains

The AWS Certified Developer - Associate exam has five (5) different domains, each with corresponding weight and topic coverage. The domains are: **Deployment (22%), Security (26%), Development with AWS Services (30%), Refactoring (10%)** and **Monitoring and Troubleshooting (12%).**

Domain 1: Deployment

- 1.1 Deploy written code in AWS using existing CI/CD pipelines, processes, and patterns
- 1.2 Deploy applications using Elastic Beanstalk
- 1.3 Prepare the application deployment package to be deployed to AWS
- 1.4 Deploy serverless applications

Domain 2: Security

- 2.1 Make authenticated calls to AWS services
- 2.2 Implement encryption using AWS services
- 2.3 Implement application authentication, and authorization

Domain 3: Development with AWS Services

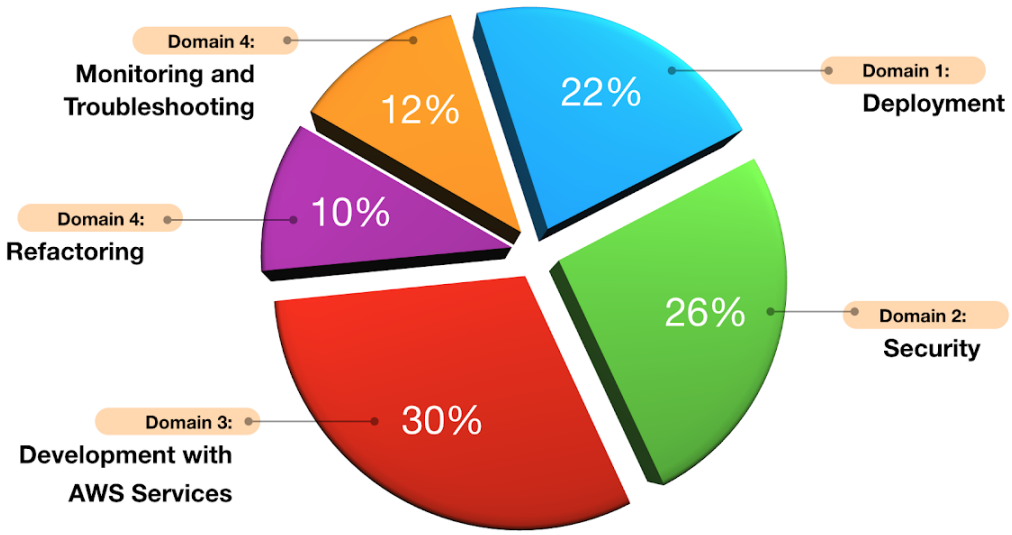
3.1 Write code for serverless applications
3.2 Translate functional requirements into application design
3.3 Implement application design into application code
3.4 Write code that interacts with AWS services by using APIs, SDKs, and AWS CLI

Domain 4: Refactoring

4.1 Optimize application to best use AWS services and features
4.2 Migrate existing application code to run on AWS

Domain 5: Monitoring and Troubleshooting

5.1 Write code that can be monitored
5.2 Perform root cause analysis on faults found in testing or production



Exam Scoring

You can get a score from 100 to 1,000 with a minimum passing score of **720** when you take the AWS Certified Developer Associate exam. AWS uses a scaled scoring model to associate scores across multiple exam types that may have different levels of difficulty. Your complete score report will be sent to you by email 1 - 5 business days after your exam. However, as soon as you finish your exam, you'll immediately see a pass or fail notification on the testing screen.

For individuals who unfortunately do not pass their exams, you must wait 14 days before you are allowed to retake the exam. There is no hard limit on the number of attempts you can retake an exam. Once you pass, you'll receive various benefits such as a discount coupon which you can use for your next AWS exam.

Once you receive your score report via email, the result should also be saved in your AWS Certification account already. The score report contains a table of your performance on each domain and it will indicate whether you have met the level of competency required for these domains. Take note that you do not need to achieve competency in all domains for you to pass the exam. At the end of the report, there will be a score performance table that highlights your strengths and weaknesses which will help you determine the areas you need to improve on.

Exam Benefits

If you successfully passed any AWS exam, you will be eligible for the following benefits:

- **Exam Discount** - You'll get a 50% discount voucher that you can apply for your recertification or any other exam you plan to pursue. To access your discount voucher code, go to the "Benefits" section of your AWS Certification Account, and apply the voucher when you register for your next exam.

- **Free Practice Exam** - To help you prepare for your next exam, AWS provides another voucher that you can use to take any official AWS practice exam for free. You can access your voucher code from the "Benefits" section of your AWS Certification Account.

- **AWS Certified Store** - All AWS certified professionals will be given access to exclusive AWS Certified merchandise. You can get your store access from the "Benefits" section of your AWS Certification Account.

- **Certification Digital Badges** - You can showcase your achievements to your colleagues and employers with digital badges on your email signatures, Linkedin profile, or on your social media accounts. You can also show your Digital Badge to gain exclusive access to Certification Lounges at AWS re:Invent, regional Appreciation Receptions, and select AWS Summit events. To view your badges, simply go to the "Digital Badges" section of your AWS Certification Account.

- **Eligibility to join AWS IQ -** With the AWS IQ program, you can monetize your AWS skills online by providing hands-on assistance to customers around the globe. AWS IQ will help you stay sharp and be well-versed on various AWS technologies. You can work at the comforts of your home and decide when or where you want to work. Interested individuals must be based in the US, have an Associate, Professional, or Specialty AWS Certification and be over 18 of age.

You can visit the official AWS Certification FAQ page to view the frequently asked questions about getting AWS Certified and other information about the AWS Certification: https://aws.amazon.com/certification/faqs/**.**

AWS CERTIFIED DEVELOPER ASSOCIATE EXAM - STUDY GUIDE AND TIPS

The AWS Certified Developer Associate certification is for those who are interested in handling cloud-based applications and services. Typically, applications developed in AWS are sold as products in the AWS Marketplace. This allows other customers to use the customized, cloud-compatible application for their own business needs. Because of this, AWS developers should be proficient in using the AWS CLI, APIs and SDKs for application development.

The AWS Certified Developer Associate exam (or AWS CDA for short) will test your ability to:

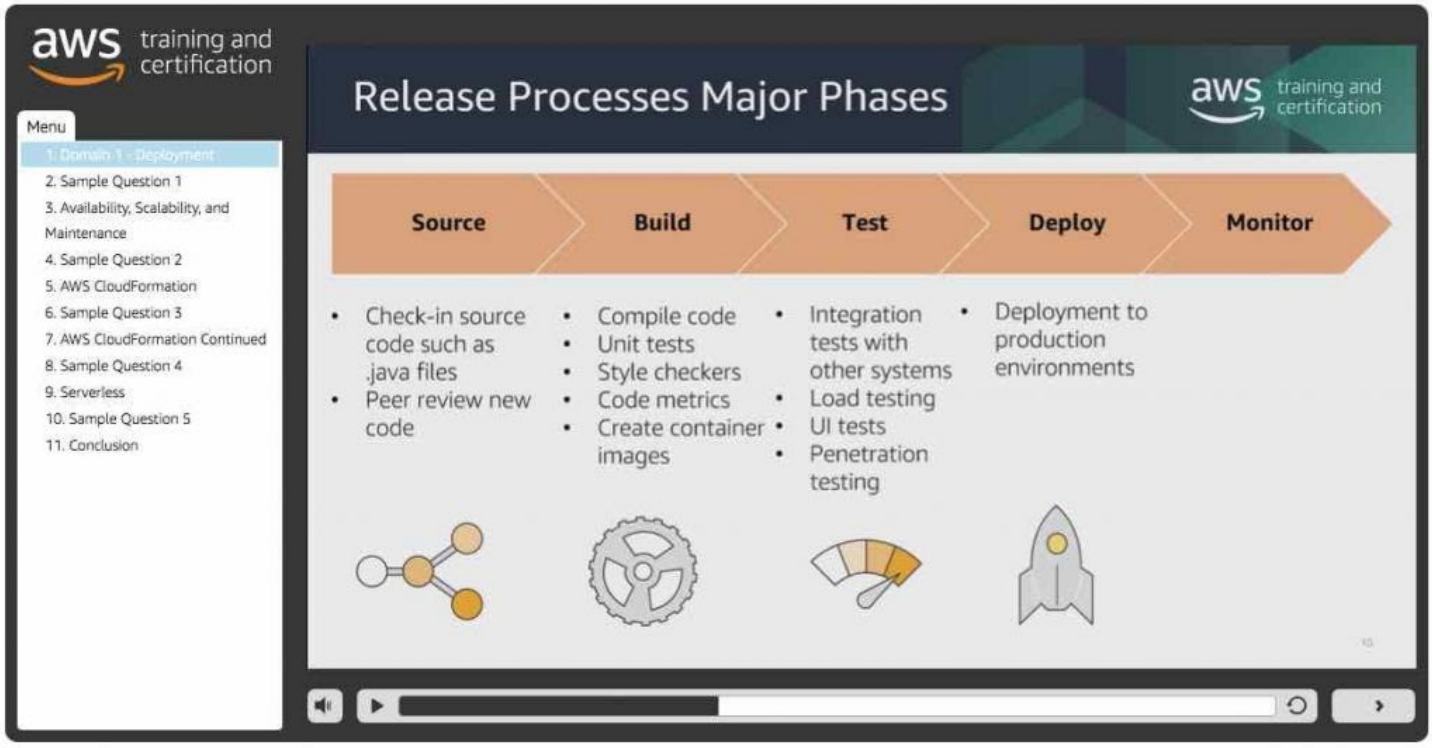
- Demonstrate an understanding of core AWS services, uses, and basic AWS architecture best practices.
- Demonstrate proficiency in developing, deploying, and debugging cloud-based applications using AWS.

Having prior experience in programming and scripting for both standard, containerized and/or serverless applications will greatly make your review easier. Additionally, we recommend having an AWS account available for you to play around with to better visualize parts in your review that involves code. For more details regarding your exam, you can check out this AWS exam blueprint.

Study Materials

If you are not well-versed in the fundamentals of AWS, we suggest that you visit our AWS Certified Cloud Practitioner review guide to get started. AWS also offers a free virtual course called AWS Cloud Practitioner Essentials that you can take in their training portal. Knowing the basic concepts and services of AWS will make your review more coherent and understandable for you.

The primary study materials you'll be using for your review are the: FREE AWS Exam Readiness video course, official AWS sample questions, AWS whitepapers, FAQs, AWS cheat sheets, and AWS practice exams.



For whitepapers, they include the following:

1. Microservices on AWS - This paper introduces the ways you can implement a microservice system on different AWS Compute platforms. You should study how these systems are built and the reasoning behind the chosen services for that system.
2. Running Containerized Microservices on AWS - This paper talks about the best practices in deploying a containerized microservice system in AWS. Focus on the example scenarios where the best practices are applied, how they are applied, and using which services to do so.
3. Optimizing Enterprise Economics with Serverless Architectures - Read upon the use cases of serverless in different platforms. Understand when is it best to use serverless vs maintaining your own servers. Also familiarize yourself with the AWS services that are under the serverless toolkit.
4. Serverless Architectures with AWS Lambda - Learn about Serverless and Lambda as much as you can. Concepts, configurations, code and architectures are all important and are most likely to come up in the exam. Creating a Lambda function of your own will help you remember features faster.
5. Practicing Continuous Integration and Continuous Delivery on AWS Accelerating Software Delivery with DevOps - If you are a developer aiming for the DevOps track, then this whitepaper is packed with practices for you to learn. CI/CD involves many stages that allows you to deploy your applications faster. Therefore, you should study the different deployment methods and understand how each of them works. Also, familiarize yourself with the implementation of CI/CD in AWS. We recommend performing a lab of this in your AWS account.

6. Blue/Green Deployments on AWS - Blue/Green Deployments is a popular deployment method that you should learn as an AWS Developer. Study how blue/green deployments are implemented and using what set of AWS services. It is also crucial that you understand the scenarios where blue/green deployments are beneficial, and where they are not. Do NOT mix up your blue environment from your green environment.
7. Architecting for the Cloud: AWS Best Practices - Be sure to understand the best practices in AWS since exam questions will focus their scenarios around these best practices. The whitepaper contains a number of design principles with examples for each. These will help you realize which services are most suitable for which kinds of situations.
8. AWS Security Best Practices - Understand the security best practices and their purpose in your environment. Some services offer more than one form of security feature, such as multiple key management schemes for encryption. It is important that you can determine which form is most suitable to the given scenarios in your exam.
9. AWS Well-Architected Framework - This whitepaper is one of the most important papers that you should study for the exam. It discusses the different pillars that make up a well-architected cloud environment. Expect the scenarios in your exam to be heavily based upon these pillars. Each pillar will have a corresponding whitepaper of its own, that discusses the respective pillar in more detail.

Also check out this article: **Top 5 FREE AWS Review Materials**.

AWS Services to Focus On

AWS offers extensive documentation and well-written FAQs for all of their services. These two will be your primary source of information when studying AWS. You need to be well-versed in a number of AWS products and services since you will almost always be using them in your work. I recommend checking out Tutorials Dojo's AWS Cheat Sheets which provides a summarized but highly informative set of notes and tips for your review on these services.

Services to study for:

1. Amazon EC2 / ELB / Auto Scaling - Be comfortable with integrating EC2 to ELBs and Auto Scaling. Study the commonly used AWS CLI commands, APIs and SDK code under these services. Focus as well on security, maintaining high availability, and enabling network connectivity from your ELB to your EC2 instances.
2. AWS Elastic Beanstalk - Know when Elastic Beanstalk is more appropriate to use than other compute solutions or infrastructure as a code solutions like CloudFormation or OpsWorks. Experiment with the service yourself in your AWS account, and understand how you can deploy and maintain your own application in Beanstalk.
3. Amazon ECS - Study how you can manage your own cluster using ECS. Also, figure out how ECS can be integrated to a CI/CD pipeline. Be sure to read the FAQs thoroughly since the exam includes multiple questions about containers.

4. AWS Lambda - The best way to learn Lambda is to create a function yourself. Also remember that Lambda allows custom runtimes that a customer can provide himself. Figure out what services can be integrated with Lambda, and how Lambda functions can capture and manipulate incoming events. Lastly, study the Serverless Application Model (SAM).
5. Amazon RDS / Amazon Aurora - Understand how RDS integrates with your application through EC2, ECS, Elastic Beanstalk and more. Compare RDS to DynamoDB and Elasticache and determine when RDS is best used. Also know when it is better to use Amazon Aurora than Amazon RDS, and when RDS is more useful than hosting your own database inside an EC2 instance.
6. Amazon DynamoDB - You should have a complete understanding of the DynamoDB service as this is very crucial in your exam. Read the DynamoDB documentation since it is more detailed and informative than the FAQ. As a developer, you should also know how to provision your own DynamoDB table, and you should be capable of tweaking its settings to meet application requirements.
7. Amazon Elasticache - Elasticache is a caching service that you'll be encountering often in the exam. Compare and contrast Redis from Memcached. Determine when Elasticache is more suitable than DynamoDB or RDS.
8. Amazon S3 - S3 is usually your go-to storage for objects. Study how you can secure your objects through KMS encryption, ACLs, and bucket policies. Know how S3 stores your objects to keep them highly durable and available. Also learn about lifecycle policies. Compare S3 to EBS and EFS to know when is S3 more preferred than the other two.
9. Amazon EFS - EFS is used to set up file systems for multiple EC2 instances. Compare and contrast S3 to EFS and EBS. Also study on file encryption and optimizing EFS performance.
10. Amazon Kinesis- There are usually tricky questions on Kinesis so you should read its documentation too. Focus on Kinesis Data Streams. Also have an idea of the other Kinesis services. Familiarize yourself with Kinesis APIs, Kinesis Sharding, and integration with storage services such as S3 or compute services such as Lambda.
11. Amazon API Gateway - API gateway is usually used together with AWS Lambda as part of the serverless application model. Understand API Gateway's structure such as resources, stages and methods. Learn how you can combine API Gateway with other AWS services such as Lambda or CloudFront. Determine how you can secure your APIs so that only a select number of people can execute it.
12. Amazon Cognito - Cognito is used for mobile and web authentication. You usually encounter Cognito questions in the exam along with Lambda, API Gateway, and DynamoDB. This usually involves some mobile application requiring an easy sign up/sign in feature from AWS. It is highly suggested that you try using Cognito to better understand its features.
13. Amazon SQS - Study the purpose of different SQS queues, timeouts and how your messages are handled inside queues. Messages in an SQS queue are not deleted when polled, so be sure to read on that as well. There are different polling mechanisms in SQS, so you should compare and contrast each one.
14. Amazon CloudWatch - CloudWatch is your primary monitoring tool for all your AWS services. Be sure to know what metrics can be found under CloudWatch monitoring, and what metrics require a

CloudWatch agent installed. Also study CloudWatch Logs, CloudWatch Alarms and Billing monitoring. Differentiate the kinds of logs stored in CloudWatch vs logs stored in CloudTrail.

15. AWS IAM - IAM is the security center of your cloud. Therefore, you should familiarize yourself with the different IAM features. Study how IAM policies are written, and what each section in the policy means. Understand the usage of IAM user roles and service roles. You should have read upon the best practices whitepaper in securing your AWS account through IAM.
16. AWS KMS - KMS contains keys that you use to encrypt EBS, S3, and other services. Know what these services are. Learn the different types of KMS keys and on which situations is each type of key used.
17. AWS CodeBuild / AWS CodeCommit / AWS CodeDeploy / AWS CodePipeline - These are your tools in implementing CI/CD in AWS. Study how you can build applications in CodeBuild (buildspec), and how you'll prepare configuration files (appspec) for CodeDeploy. CodeCommit is a git repository so having knowledge in Git will be beneficial. I suggest to build a simple pipeline of your own in CodePipeline to see how you should manage your code deployments. It is also important to learn how you can rollback to your previous application version after a failed deployment. The whitepapers above should have explained in-place deployments and blue/green deployments, and how to perform automation.
18. AWS CloudFormation - Study the structure of CloudFormation scripts and how you can use them to build your infrastructure. Be comfortable with both json and yaml formats. Read a bit about stacksets. List down the services that use CloudFormation in the backend for provisioning AWS resources, such as AWS SAM, and processes such as in CI/CD.

Aside from the concepts and services, you should study about the AWS CLI, the different commonly used APIs (for services such as EC2, EBS or Lambda), and the AWS SDKs. Read up on the AWS Serverless Application Model (AWS SAM) and AWS Server Migration Services as well as these may come up in the exam. It will also be very helpful to have experience interacting with AWS APIs and SDKs, and troubleshooting any errors that you encounter while using them.

Common Exam Scenarios

Scenario	Solution
AWS Lambda	
An application running in a local server is converted to a Lambda function. When the function was tested, an Unable to import module error shows.	Install the missing modules in your application's folder and package it into ZIP file before uploading to AWS Lambda.
A Developer is writing a Lambda function that will be used to send a request to an API in different environments (Prod, Dev, Test). The function needs to automatically invoke the correct API call based on the environment.	Use Environment Variables

A Lambda function needs a temporary storage to store files while executing.	Store the files in the /tmp directory
A Lambda function is writing data into an RDS database. The function needs to reuse database connection to reduce execution time.	Use execution context by placing the database connection logic outside of the event handler.
A Developer needs to increase the CPU available to a Lambda function to process data more efficiently.	Increase the allocated memory of the function.
Amazon API Gateway	
A Developer has an application that uses a RESTful API hosted in API Gateway. The API requests are failing with a "No 'Access-Control-Allow-Origin' header is present on the requested resource" error message.	Enable CORS in the API Gateway Console.
A website integrated with API Gateway requires user requests to reach the backend server **without intervention** from the API Gateway. Which integration type should be used?	HTTP_PROXY
A serverless application is composed of AWS Lambda, DynamoDB, and API Gateway. Users are complaining about getting HTTP 504 errors.	The API requests are reaching the maximum integration timeout for API Gateway (29 seconds).
How to invalidate API Gateway cache?	1. Send a request with a Cache-Control: max-age header. 2. Enable the Require Authorization option on your API cache settings.
A developer needs to deploy different API versions in API Gateway	Use stage variables
Amazon DynamoDB	
A Developer needs a cost-effective solution to delete session data in a DynamoDB table.	Expire session data with DynamoDB TTL
New changes to a DynamoDB table should be recorded in another DynamoDB table.	Use DynamoDB Streams
Reduce the DynamoDB database response time.	Use DynamoDB Accelerator (DAX)

Choosing the best partition key for the DynamDB table.	Use the partition key with the highest cardinality (e.g. student ID, employee ID)
An application is using a DynamoDB database with Global Secondary Index. DynamoDB requests are returning a ProvisionedThroughputExceededException error. Why is this happening?	The write capacity of the GSI is less than the base table.
CloudFormation and AWS SAM	
What section must be added to a CloudFormation template to include resources defined by AWS SAM?	Transform
A developer needs a reliable framework for building serverless applications in AWS	AWS SAM
A CloudFormation stack creation process failed unexpectedly	CloudFormation will rollback by deleting resources that it has already created.
A CloudFormation template will be used across multiple AWS accounts	Use CloudFormation StackSets
Deployment and Security	
It is required that incoming traffic is shifted in two increments. 10% of the traffic must be shifted in the first increment, and the remaining 90% should be deployed after some minutes.	Canary
You need to authenticate users of a website using social media identity profiles.	Amazon Cognito Identity Pools
A company has two accounts. The developers from Account A needs to access resources on Account B.	Use cross-account access role
Multiple developers need to make incremental code updates to a single project and then deploy the new changes.	Use AWS CodeCommit as the code repository and directly deploy the new package using AWS CodeDeploy.
A Developer is planning a task placement strategy for an ECS cluster based on the least available amount of CPU.	Use binpack
Relevant API/CLI commands	

A Developer needs to decode an encoded authorization failure message.	Use the aws sts decode-authorization-message command.
How can a Developer verify permission to call a CLI command without actually making a request?	Use the --dry-run parameter along with the CLI command.
A Developer needs to deploy a CloudFormation template from a local computer.	Use the aws cloudformation package and aws cloudformation deploy command
A Developer has to ensure that no applications can fetch a message from an SQS queue that's being processed or has already been processed.	Increase the VisibilityTimeout value using the ChangeMessageVisibility API and delete the message using the DeleteMessage API.
A Developer has created an IAM Role for an application that uploads files to an S3 bucket. Which API call should the Developer use to allow the application to make upload requests?	Use the AssumeRole API

Validate Your Knowledge

The AWS CDA exam will be packed with tricky questions. It would be great if you could get a feel of how the questions are structured through practice tests. Luckily, **Tutorials Dojo** offers a great set of **practice questions** for you to try out **here**. These practice tests will help validate your knowledge of what you've learned so far and fill in any missing details that you might have skipped in your review. You can pair our practice exams with this study guide eBook to further help in your exam preparations.

Sample Practice Test Questions:

Question 1

A web application hosted in Elastic Beanstalk has a configuration file named .ebextensions/debugging.config which has the following content:

option_settings:

aws:elasticbeanstalk:xray:

XRayEnabled: true

For its database tier, it uses RDS with Multi-AZ deployments configuration and Read Replicas. There is a new requirement to record calls that your application makes to RDS and other internal or external HTTP web APIs. The tracing information should also include the actual SQL database queries sent by the application, which can be searched using the filter expressions in the X-Ray Console.

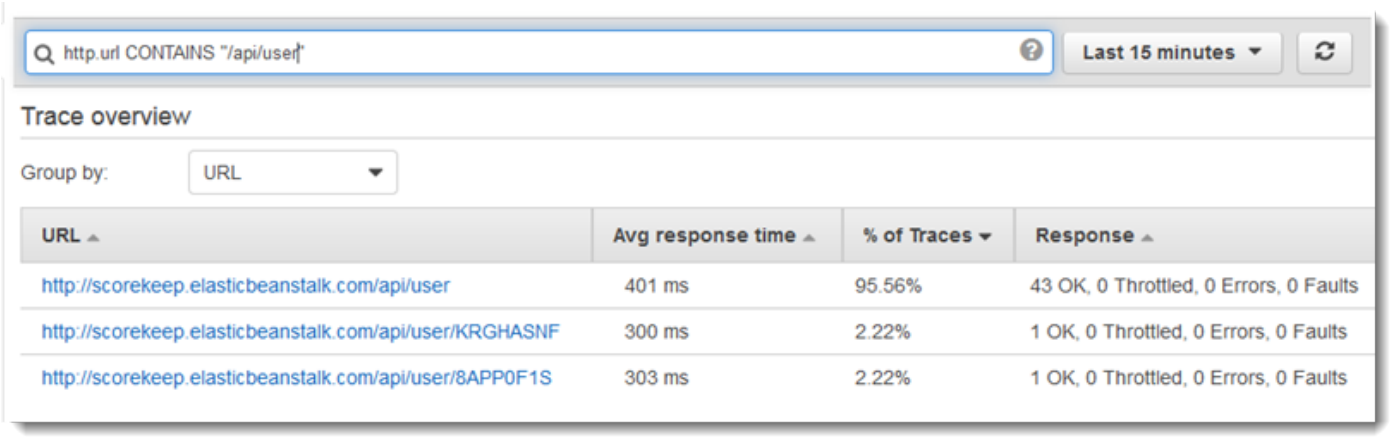
Which of the following should you do to satisfy the above task?

1. Add metadata in the segment document.

2. Add annotations in the segment document.
3. Add metadata in the subsegment section of the segment document.
4. Add annotations in the subsegment section of the segment document.

Correct Answer: 4

Even with sampling, a complex application generates a lot of data. The AWS X-Ray console provides an easy-to-navigate view of the service graph. It shows health and performance information that helps you identify issues and opportunities for optimization in your application. For advanced tracing, you can drill down to traces for individual requests, or use **filter expressions** to find traces related to specific paths or users.



When you instrument your application, the X-Ray SDK records information about incoming and outgoing requests, the AWS resources used, and the application itself. You can add other information to the segment document as annotations and metadata.

Annotations are simple key-value pairs that are indexed for use with filter expressions. Use annotations to record data that you want to use to group traces in the console, or when calling the GetTraceSummaries API. X-Ray indexes up to 50 annotations per trace.

Metadata are key-value pairs with values of any type, including objects and lists, but that are not indexed. Use metadata to record data you want to store in the trace but don't need to use for searching traces. You can view annotations and metadata in the segment or subsegment details in the X-Ray console.

A trace segment is a JSON representation of a request that your application serves. A trace segment records information about the original request, information about the work that your application does locally, and subsegments with information about downstream calls that your application makes to AWS resources, HTTP APIs, and SQL databases.

Hence, **adding annotations in the subsegment section of the segment document** is the correct answer.

Adding annotations in the segment document is incorrect because although the use of annotations is correct, you have to add this in the ***subsegment*** section of the *segment* document since you want to trace the downstream call to RDS and not the actual request to your application.

Adding metadata in the segment document is incorrect because metadata is primarily used to record custom data that you want to store in the trace but not for searching traces since this can't be picked up by filter expressions in the X-Ray Console. You have to use annotations instead. In addition, you have to add this in the *subsegment* section of the *segment* document since you want to trace the downstream call to RDS and not the actual request to your application.

Adding metadata in the subsegment section of the segment document is incorrect because, just as mentioned above, metadata is just used to record custom data that you want to store in the trace but not for searching traces.

References:
https://docs.aws.amazon.com/xray/latest/devguide/xray-concepts.html#xray-concepts-annotations
https://docs.aws.amazon.com/xray/latest/devguide/xray-console-filters.html

Check out this AWS X-Ray Cheat Sheet:
https://tutorialsdojo.com/aws-x-ray/

Question 2

A developer wants to expose a legacy web service that uses an XML-based Simple Object Access Protocol (SOAP) interface through API Gateway. However, there is a compatibility issue since most modern applications communicate data in JSON format.

Which is the most cost-effective method that will overcome this issue?

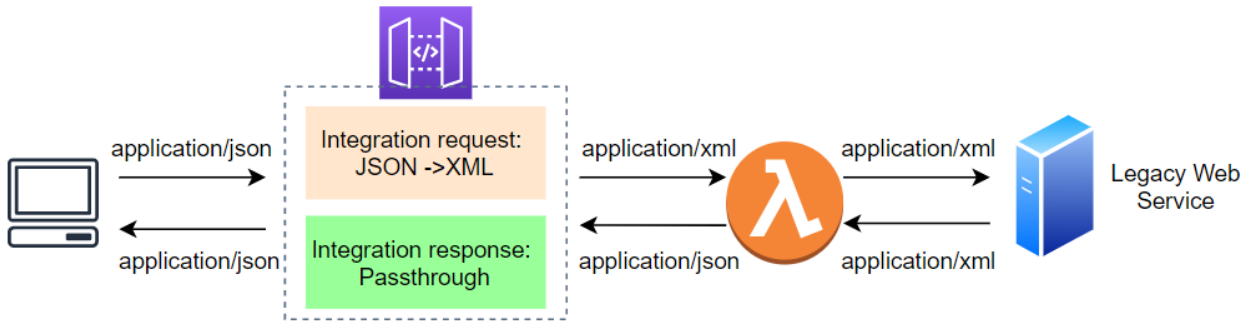
1. Use API Gateway to create a WebSocket API. Transform the incoming JSON into XML using mapping templates. Forward the request into the SOAP interface by using a Lambda function and parse the response (XML) into JSON before sending back to API Gateway.
2. Use API Gateway to create a RESTful API. Transform the incoming JSON into XML for the SOAP interface through an Application Load Balancer and vice versa. Put the legacy web service behind the ALB.
3. Use API Gateway to create a RESTful API. Send the incoming JSON to an HTTP server hosted on an EC2 instance and have it transform the data into XML and vice versa before sending it to the legacy application.

4. Use API Gateway to create a RESTful API. Transform the incoming JSON into XML using mapping templates. Forward the request into the SOAP interface by using a Lambda function and parse the response (XML) into JSON before sending back to API Gateway.

Correct Answer: 4

You (or your organization) probably has some existing web services that respond to older protocols such as XML-RPC or SOAP. You can use the API Gateway to modernize these services.

In API Gateway, an API's method request can take a payload in a different format from the corresponding integration request payload, as required in the backend. Similarly, the backend may return an integration response payload different from the method response payload, as expected by the frontend. API Gateway lets you use mapping templates to map the payload from a method request to the corresponding integration request and from an integration response to the corresponding method response.



Hence, the correct answer is: **Use API Gateway to create a RESTful API. Transform the incoming JSON into XML using mapping templates. Forward the request into the SOAP interface by using a Lambda function and parse the response (XML) into JSON before sending back to API Gateway.**

The option that says: **Use API Gateway to create a WebSocket API. Transform the incoming JSON into XML using mapping templates. Forward the request into the SOAP interface by using a Lambda function and parse the response (XML) into JSON before sending back to API Gateway** is incorrect. The WebSocket protocol is mainly used for applications that require bidirectional persistent connection such as push notifications and chat messaging applications. For this scenario, using a REST-based approach is the more appropriate solution.

The option that says: **Use API Gateway to create a RESTful API. Transform the incoming JSON into XML for the SOAP interface through an Application Load Balancer and vice versa. Put the legacy web service behind the ALB** is incorrect because ALB is not capable of transforming data.

The option that says: **Use API Gateway to create a RESTful API. Send the incoming JSON to an HTTP server hosted on an EC2 instance and have it transform the data into XML and vice versa before sending it to the**

legacy application is incorrect. Although this could work, this means that you'll have to provision and run an EC2 instance 24/7, which is more expensive than just using a Lambda Function.

References:
https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-lambda-proxy-integrations.html
https://docs.aws.amazon.com/apigateway/latest/developerguide/rest-api-data-transformations.html
https://github.com/mwittenbols/How-to-use-Lambda-and-API-Gateway-to-consume-XML-instead-of-JSON

Check out this Amazon API Gateway Cheat Sheet:
https://tutorialsdojo.com/amazon-api-gateway/

Click **here** for more **AWS Certified Developer Associate practice exam questions**.

Check out our other AWS practice test courses **here**:

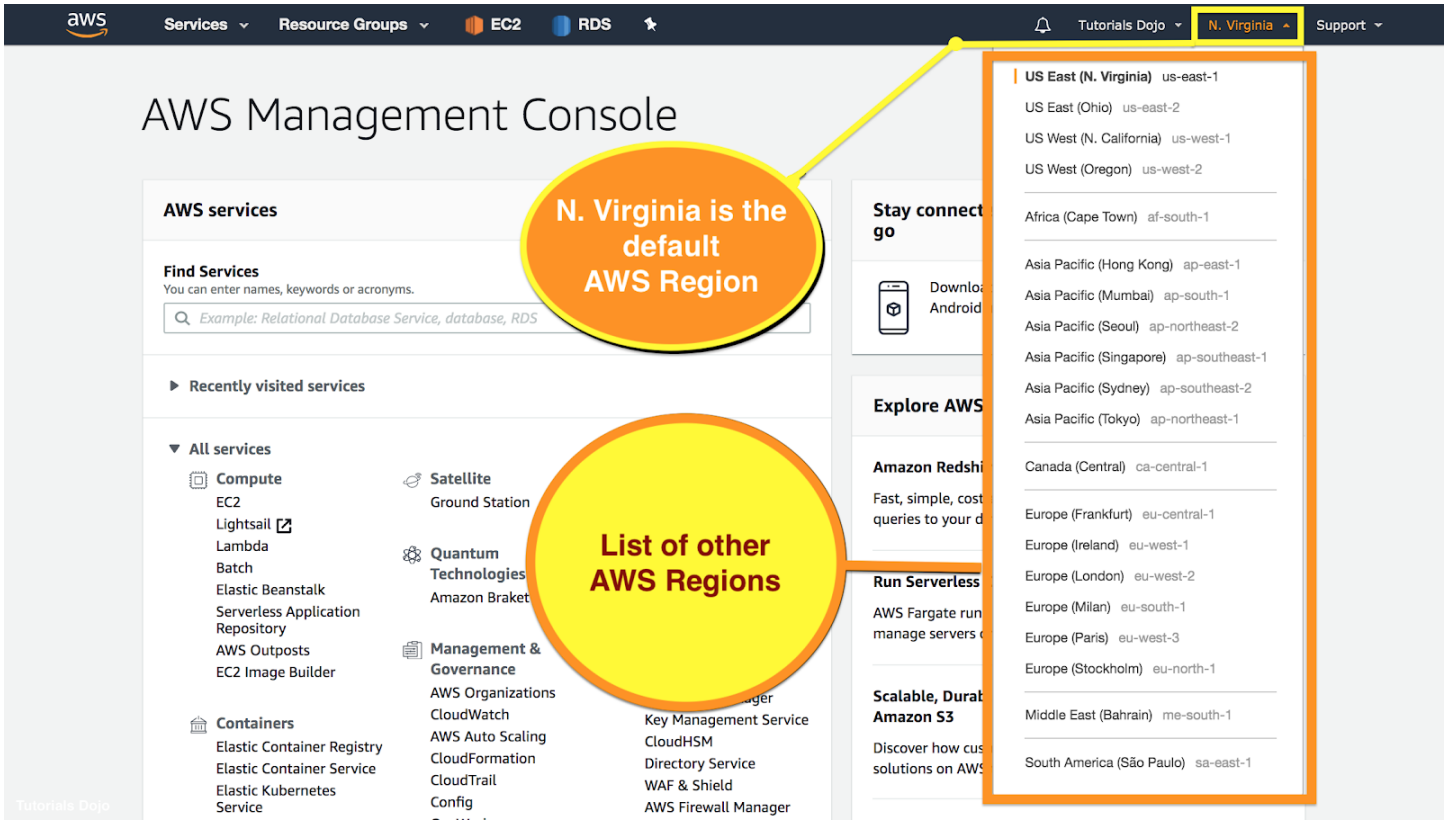
The AWS CDA certification is one of the most sought after certifications in the DevOps industry. It validates your knowledge of the AWS Cloud and foundational DevOps practices. It is an achievement of its own if you become AWS certified. Hence, it will be best if you could get proper sleep the day before your exam. Review any notes that you have written down, and go over the incorrect items in your **practice tests** if you took it. You should also check again the venue, the time, and the things needed for your exam. As so, we wish you the best of luck and the best of outcome

AWS CHEAT SHEETS

AWS OVERVIEW

AWS Global infrastructure

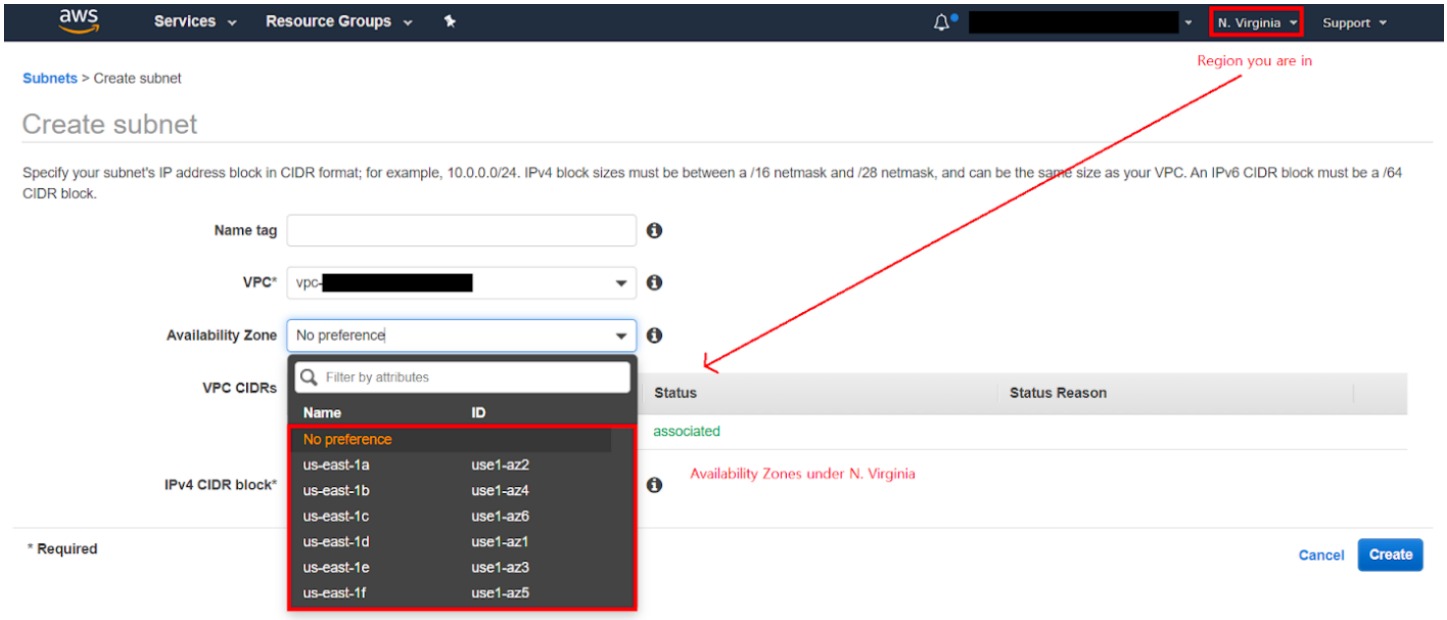
- The AWS Global infrastructure is built around **Regions** and **Availability Zones** (AZs).
- **Regions** provide multiple, physically separated and isolated **Availability Zones** which are connected with low latency, high throughput, and highly redundant networking

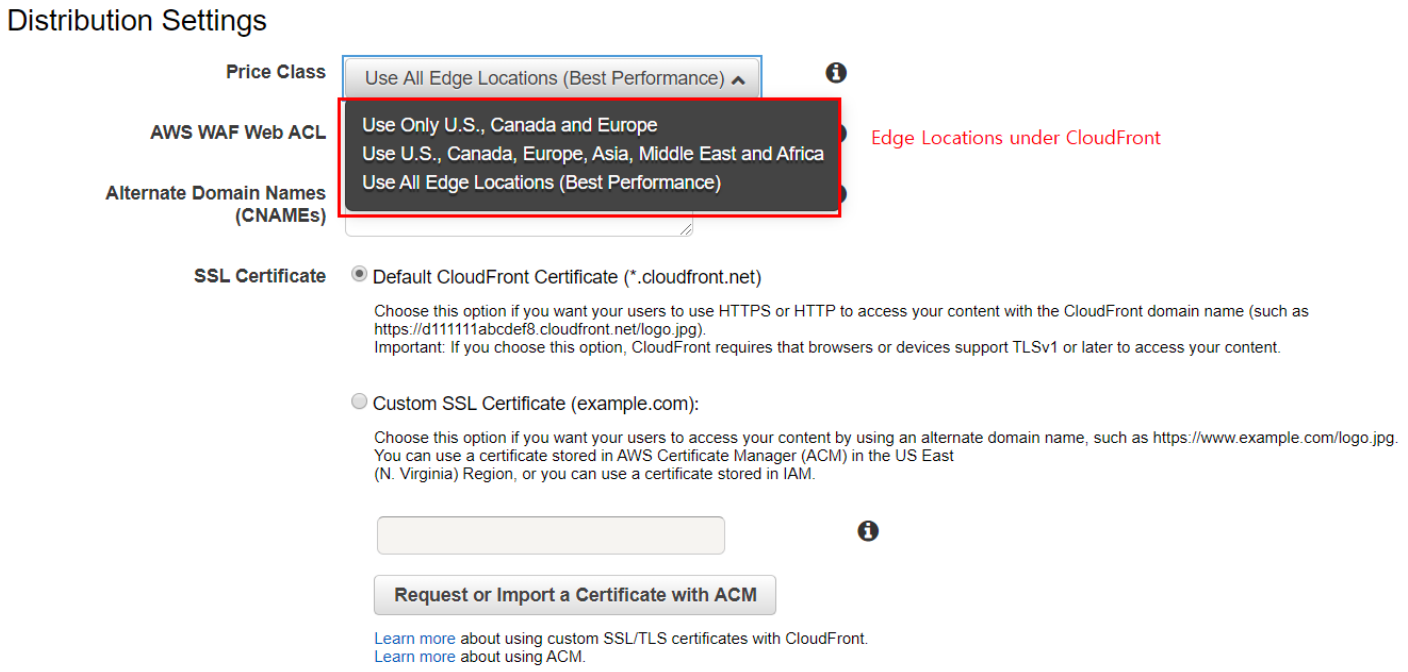


- **Availability Zones** offer highly availability, fault tolerance, and scalability.
    - Consist of one or more discrete data centers, each with redundant power, networking, and connectivity, housed in separate facilities.
    - An Availability Zone is represented by a **region code** followed by a **letter identifier**; for example, us-east-1a.

- An **AWS Local Region** is a single datacenter designed to complement an existing AWS Region. An **AWS Local Zone** places AWS compute, storage, database, and other select services closer to large population, industry, and IT centers where no AWS Region exists today.
- To deliver low-latency content to users around the globe, AWS has placed **Points of Presence**, which are either edge locations or edge caches. These points are used by Cloudfront and Lambda@edge services.
- **Edge locations** are locations where end users access services located at AWS.

View the Interactive AWS Global Infrastructure Map here.



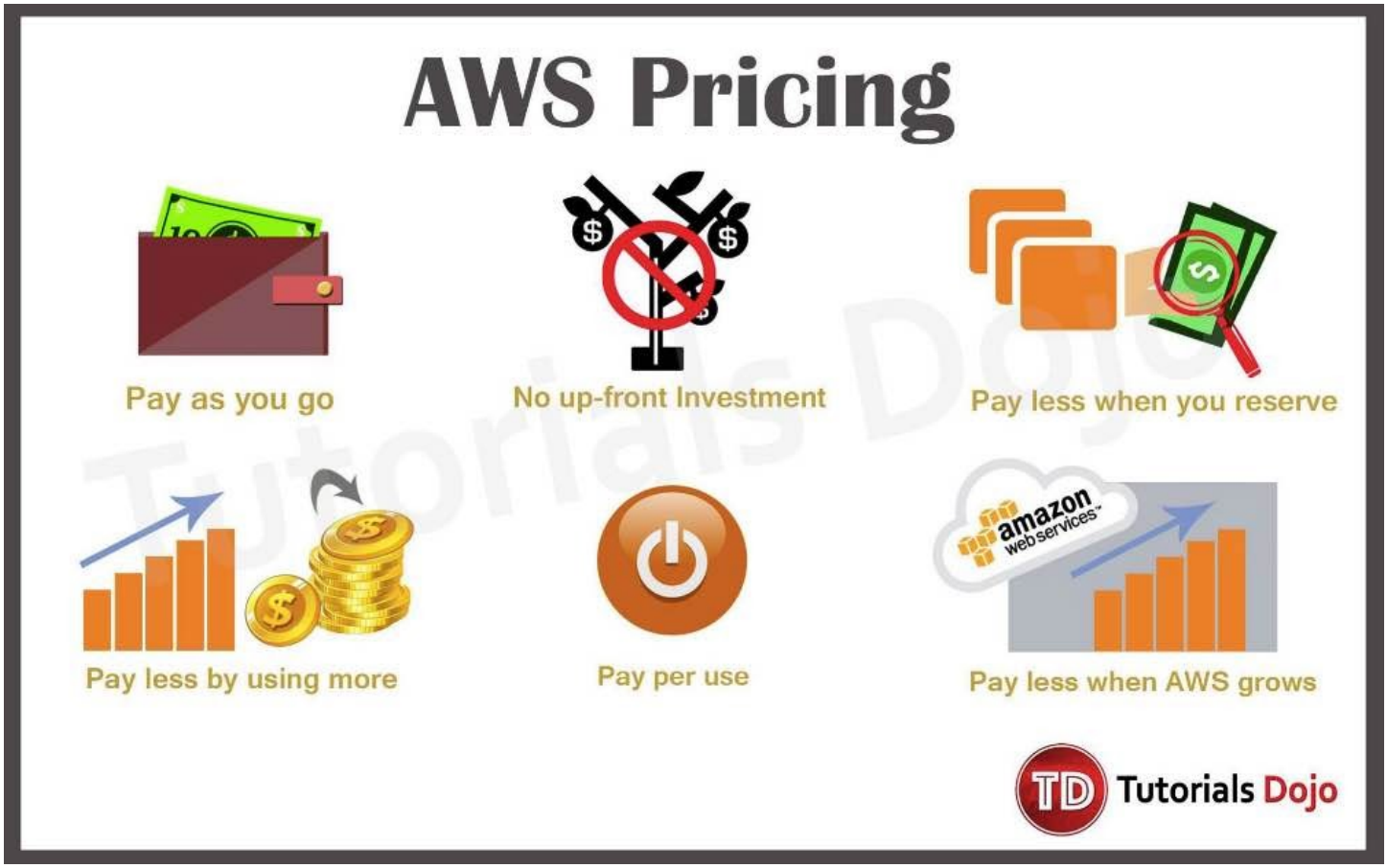


Sources:

https://aws.amazon.com/about-aws/global-infrastructure/
https://docs.aws.amazon.com/aws-technical-content/latest/aws-overview/global-infrastructure.html
https://www.infrastructure.aws/

AWS Pricing

- There are three fundamental drivers of cost with AWS:
  - Compute
  - Storage
  - Outbound data transfer.
- AWS offers pay-as-you-go for pricing.



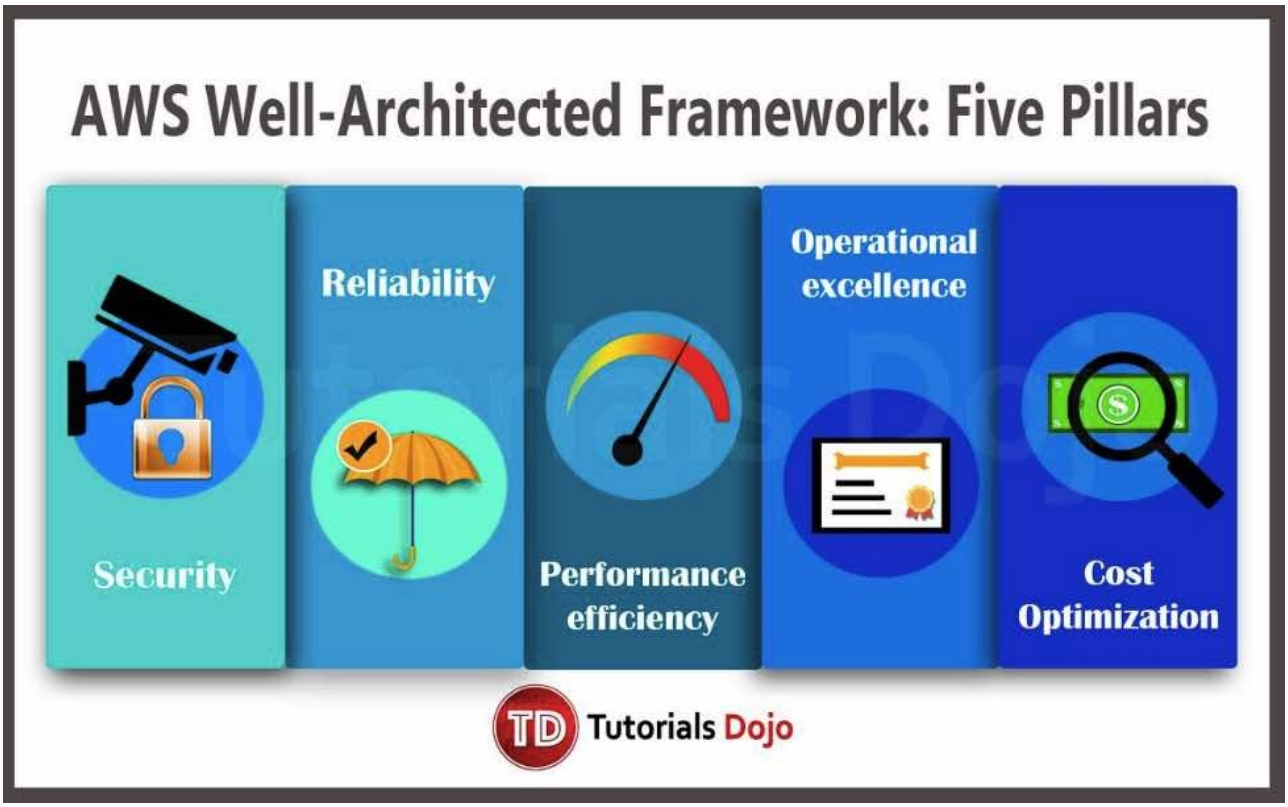
- For certain services like **Amazon EC2, Amazon EMR,** and **Amazon RDS**, you can invest in reserved capacity. With Reserved Instances, you can save up to 75% over equivalent on-demand capacity. When you buy Reserved Instances, the larger the upfront payment, the greater the discount.
  - With the **All Upfront** option, you pay for the entire Reserved Instance term with one upfront payment. This option provides you with the largest discount compared to On-Demand instance pricing.
  - With the **Partial Upfront** option, you make a low upfront payment and are then charged a discounted hourly rate for the instance for the duration of the Reserved Instance term.

    - The **No Upfront** option does not require any upfront payment and provides a discounted hourly rate for the duration of the term.
- There are also volume-based discounts for services such as **Amazon S3.**
- For new accounts, AWS Free Tier is available.
    - Free Tier offers limited usage of AWS products at no charge for 12 months since the account was created. More details at https://aws.amazon.com/free/.
- You can estimate your monthly AWS bill using **AWS Pricing Calculator**.
    - Estimate the cost of migrating your architecture to the cloud.
    - Generate the lowest cost estimate for your workload.

Sources:
https://d1.awsstatic.com/whitepapers/aws_pricing_overview.pdf
https://aws.amazon.com/pricing/
https://aws.amazon.com/ec2/pricing/reserved-instances/pricing/

AWS Well-Architected Framework - Five Pillars

Having well-architected systems greatly increases the plausibility of business success which is why AWS created the AWS Well-Architected Framework. This framework is composed of five pillars that help you understand the pros and cons of decisions you make while building cloud architectures and systems on the AWS platform. You will learn the architectural best practices for designing and operating reliable, efficient, cost-effective and secure systems in the cloud by using the framework. It also provides a way to consistently measure your architectures against best practices and identify areas for improvement.



1. Operational Excellence

- The ability to run and monitor systems to deliver business value and to continually improve supporting processes and procedures.
- There are three best practice areas and tools for operational excellence in the cloud:
    - Prepare - AWS Config
    - Operate - Amazon CloudWatch
    - Evolve - Amazon Elasticsearch Service
- Key AWS service:
    - AWS CloudFormation for creating templates. (See AWS Management Tools Cheat Sheet)

2. Security

- The ability to protect information, systems, and assets while delivering business value through risk assessments and mitigation strategies.
- There are five best practice areas and tools for security in the cloud:
    - Identity and Access Management - IAM, Multi-Factor Authentication, AWS Organizations
    - Detective Controls - AWS CloudTrail, AWS Config, Amazon GuardDuty
    - Infrastructure Protection - Amazon VPC, Amazon CloudFront with AWS Shield, AWS WAF
    - Data Protection - ELB, Amazon Elastic Block Store (Amazon EBS), Amazon S3, and Amazon Relational Database Service (Amazon RDS) encryption, Amazon Macie, AWS Key Management Service (AWS KMS)
    - Incident Response - IAM, Amazon CloudWatch Events
- Key AWS service:
    - AWS Identity and Access Management (IAM)

3. Reliability

- The ability of a system to recover from infrastructure or service disruptions, dynamically acquire computing resources to meet demand, and mitigate disruptions such as misconfigurations or transient network issues.
- There are three best practice areas and tools for reliability in the cloud:
    - Foundations - IAM, Amazon VPC, AWS Trusted Advisor, AWS Shield
    - Change Management - AWS CloudTrail, AWS Config, Auto Scaling, Amazon CloudWatch
    - Failure Management - AWS CloudFormation, Amazon S3, AWS KMS, Amazon Glacier
- Key AWS service:
    - Amazon CloudWatch

4. Performance Efficiency

- The ability to use computing resources efficiently to meet system requirements, and to maintain that efficiency as demand changes and technologies evolve.
- There are four best practice areas for performance efficiency in the cloud:
    - Selection - Auto Scaling for Compute, Amazon EBS and S3 for Storage, Amazon RDS and DynamoDB for Database, Route53, VPC, and AWS Direct Connect for Network
    - Review - AWS Blog and What's New section of the website
    - Monitoring - Amazon CloudWatch
    - Tradeoffs - Amazon Elasticache, Amazon CloudFront, AWS Snowball, Amazon RDS read replicas.
- Key AWS service:
    - Amazon CloudWatch

5. Cost Optimization

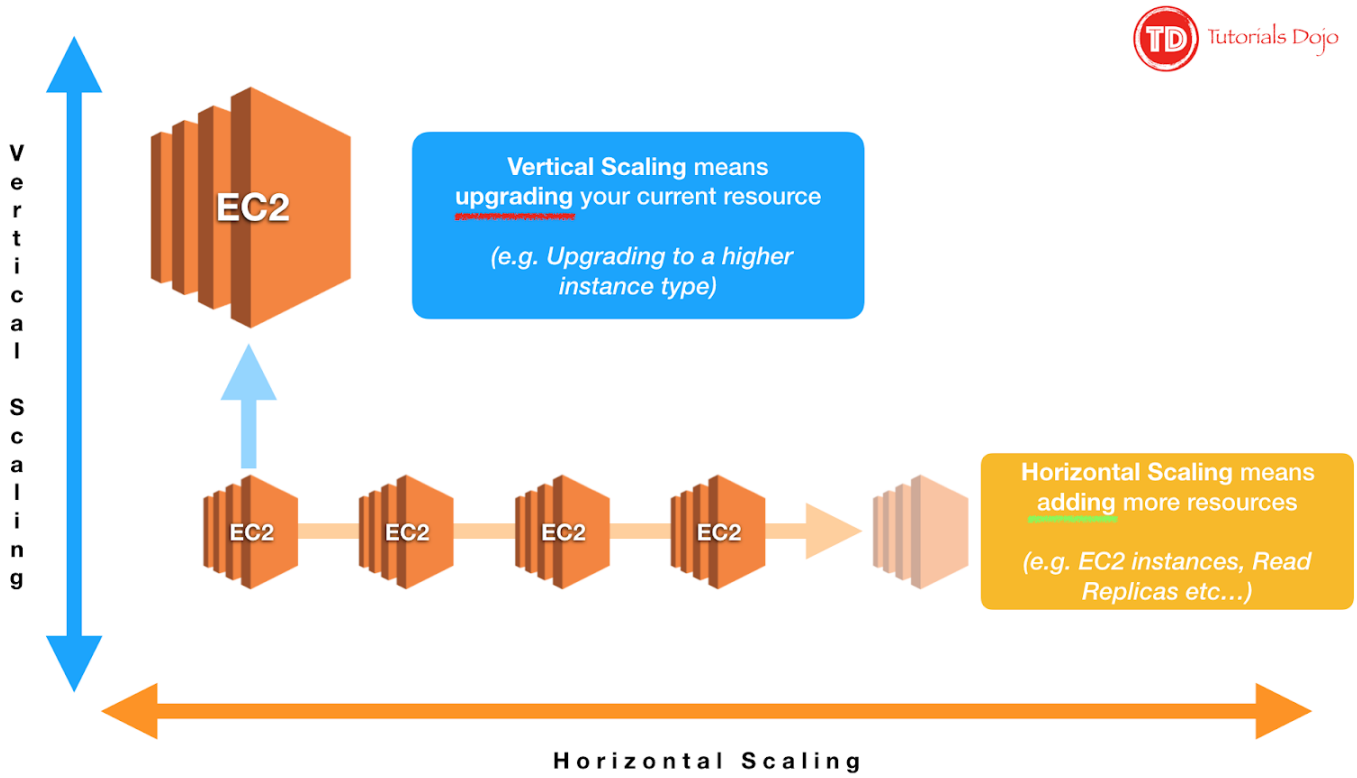
- The ability to avoid or eliminate unneeded cost or suboptimal resources.
- There are four best practice areas and tools for cost optimization in the cloud:
    - Cost-Effective Resources - Cost Explorer, Amazon CloudWatch and Trusted Advisor, Amazon Aurora for RDS, AWS Direct Connect with Amazon CloudFront
    - Matching supply and demand - Auto Scaling
    - Expenditure Awareness - AWS Cost Explorer, AWS Budgets
    - Optimizing Over Time - AWS News Blog and the What's New section on the AWS website, AWS Trusted Advisor
- Key AWS service:
    - Cost Explorer

Source:
https://d1.awsstatic.com/whitepapers/architecture/AWS_Well-Architected_Framework.pdf

AWS Well-Architected Framework - Design Principles

1. Scalability

- **Scaling Horizontally** - an increase in the number of resources
- **Scaling Vertically** - an increase in the specifications of an individual resource



2. Disposable Resources Instead of Fixed Servers

- **Instantiating Compute Resources** - automate setting up of new resources along with their configuration and code
- **Infrastructure as Code** - AWS assets are programmable. You can apply techniques, practices, and tools from software development to make your whole infrastructure reusable, maintainable, extensible, and testable.

3. Automation

- **Serverless Management and Deployment** - being serverless shifts your focus to automation of your code deployment. AWS handles the management tasks for you.

- **Infrastructure Management and Deployment** - AWS automatically handles details, such as resource provisioning, load balancing, auto scaling, and monitoring, so you can focus on resource deployment.
- **Alarms and Events** - AWS services will continuously monitor your resources and initiate events when certain metrics or conditions are met.

4. Loose Coupling

- **Well-Defined Interfaces** - reduce interdependencies in a system by allowing various components to interact with each other only through specific, technology agnostic interfaces, such as RESTful APIs.
- **Service Discovery** - applications that are deployed as a set of smaller services should be able to be consumed without prior knowledge of their network topology details. Apart from hiding complexity, this also allows infrastructure details to change at any time.
- **Asynchronous Integration** - interacting components that do not need an immediate response and where an acknowledgement that a request has been registered will suffice, should integrate through an intermediate durable storage layer.
- **Distributed Systems Best Practices** - build applications that handle component failure in a graceful manner.

5. Services, Not Servers

- **Managed Services** - provide building blocks that developers can consume to power their applications, such as databases, machine learning, analytics, queuing, search, email, notifications, and more.
- **Serverless Architectures** - allow you to build both event-driven and synchronous services without managing server infrastructure, which can reduce the operational complexity of running applications.

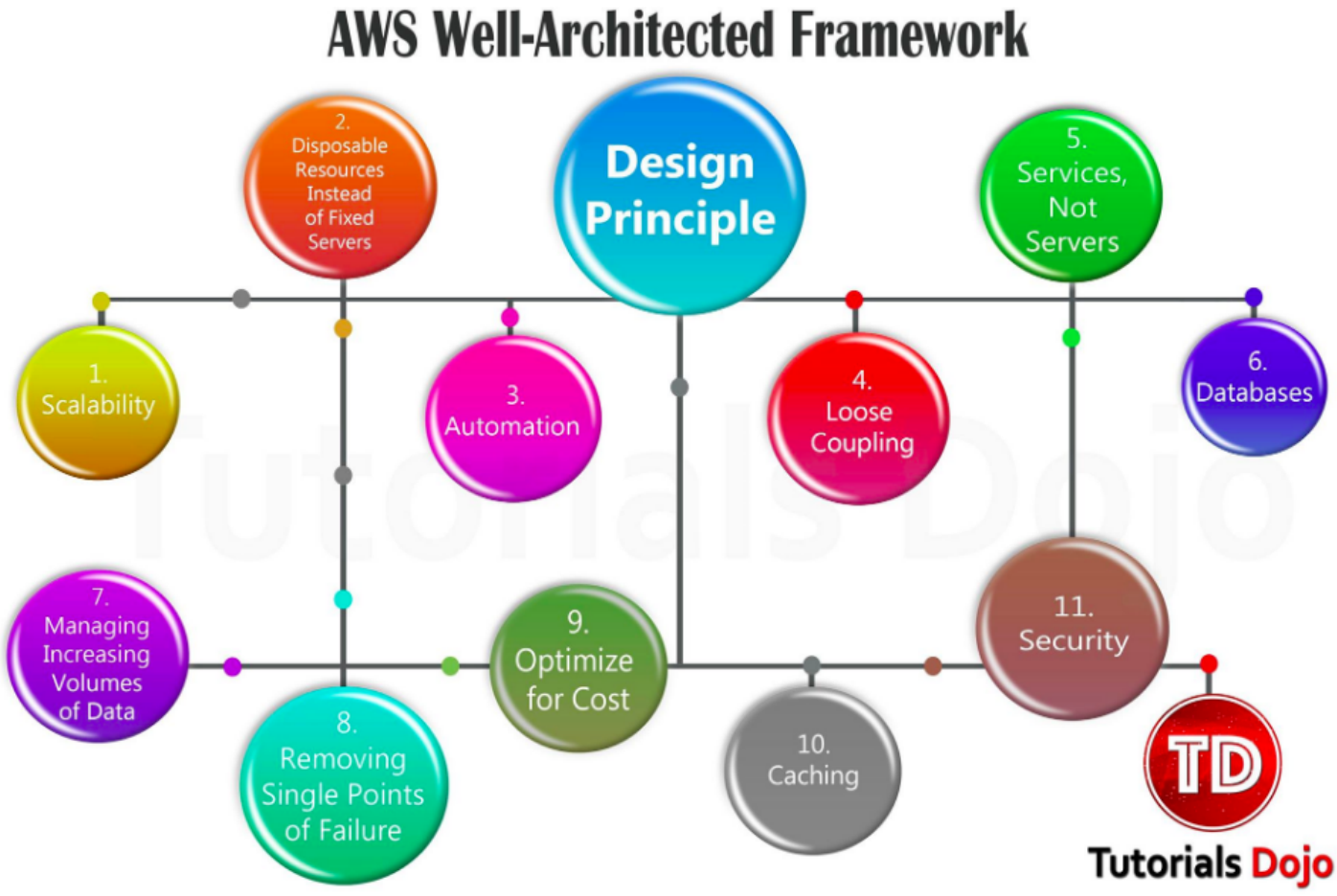
6. Databases

- Choose the Right Database Technology for Each Workload
- **Relational Databases** provide a powerful query language, flexible indexing capabilities, strong integrity controls, and the ability to combine data from multiple tables in a fast and efficient manner.
- **NoSQL Databases** trade some of the query and transaction capabilities of relational databases for a more flexible data model that seamlessly scales horizontally. It uses a variety of data models, including graphs, key-value pairs, and JSON documents, and are widely recognized for ease of development, scalable performance, high availability, and resilience.
- **Data Warehouses** are a specialized type of relational database, which is optimized for analysis and reporting of large amounts of data.
- **Graph Databases** uses graph structures for queries.
    - Search Functionalities
        - Search is often confused with query. A query is a formal database query, which is addressed in formal terms to a specific data set. Search enables datasets to be queried that are not precisely structured.

- A search service can be used to index and search both structured and free text format and can support functionality that is not available in other databases, such as customizable result ranking, faceting for filtering, synonyms, and stemming.

7. Managing Increasing Volumes of Data

- **Data Lake** - an architectural approach that allows you to store massive amounts of data in a central location so that it's readily available to be categorized, processed, analyzed, and consumed by diverse groups within your organization.



8. Removing Single Points of Failure

- **Introducing Redundancy**
  - **Standby redundancy** - when a resource fails, functionality is recovered on a secondary resource with the failover process. The failover typically requires some time before it completes, and

during this period the resource remains unavailable. This is often used for stateful components such as relational databases.
    - Active redundancy - requests are distributed to multiple redundant compute resources. When one of them fails, the rest can simply absorb a larger share of the workload.
- **Detect Failure** - use health checks and collect logs
- **Durable Data Storage**
    - **Synchronous replication** - only acknowledges a transaction after it has been durably stored in both the primary storage and its replicas. It is ideal for protecting the integrity of data from the event of a failure of the primary node.
    - **Asynchronous replication** - decouples the primary node from its replicas at the expense of introducing replication lag. This means that changes on the primary node are not immediately reflected on its replicas.
    - **Quorum-based replication** - combines synchronous and asynchronous replication by defining a minimum number of nodes that must participate in a successful write operation.
- **Automated Multi-Data Center Resilience** - utilize AWS Regions and Availability Zones (Multi-AZ Principle). (See Disaster Recovery section)
- **Fault Isolation and Traditional Horizontal Scaling** - Shuffle Sharding

9. Optimize for Cost

- **Right Sizing** - AWS offers a broad range of resource types and configurations for many use cases.
- **Elasticity** - save money with AWS by taking advantage of the platform's elasticity.
- **Take Advantage of the Variety of Purchasing Options** - Reserved Instances vs Spot Instances (See AWS Pricing)

10. Caching

- **Application Data Caching** - store and retrieve information from fast, managed, in-memory caches.
- **Edge Caching** - serve content by infrastructure that is closer to viewers, which lowers latency and gives high, sustained data transfer rates necessary to deliver large popular objects to end users at scale.

11. Security

- **Use AWS Features for Defense in Depth** - secure multiple levels of your infrastructure from network down to application and database.
- **Share Security Responsibility with AWS** - AWS handles security OF the Cloud while customers handle security IN the Cloud.
- **Reduce Privileged Access** - implement Principle of Least Privilege controls.
- **Security as Code** - firewall rules, network access controls, internal/external subnets, and operating system hardening can all be captured in a template that defines a *Golden Environment*.

- **Real-Time Auditing** - implement continuous monitoring and automation of controls on AWS to minimize exposure to security risks.

12. Cloud Architecture Best Practices

There are various best practices that you can follow which can help you build an application in the AWS cloud. The notable ones are:

1. **Decouple your components** - the key concept is to build components that do not have tight dependencies on each other so that if one component were to fail for some reason, the other components in the system will continue to work. This is also known as loose coupling. This reinforces the Service-Oriented Architecture (SOA) design principle that the more loosely coupled the components of the system are, the better and more stable it scales.
2. **Think parallel** - This internalizes the concept of parallelization when designing architectures in the cloud. It encourages you to implement parallelization whenever possible and to also automate the processes of your cloud architecture.
3. **Implement elasticity** - This principle is implemented by automating your deployment process and streamlining the configuration and build process of your architecture. This ensures that the system can scale in and scale out to meet the demand without any human intervention.
4. **Design for failure** - This concept encourages you to be a pessimist when designing architectures in the cloud and assume that the components of your architecture will fail. This reinforces you to always design your cloud architecture to be highly available and fault-tolerant.

Sources:
https://d1.awsstatic.com/whitepapers/AWS_Cloud_Best_Practices.pdf
https://www.slideshare.net/AmazonWebServices/best-practices-for-architecting-in-the-cloud-jeff-barr

AWS Well-Architected Framework - Disaster Recovery

- **RTO** is the time it takes after a disruption to restore a business process to its service level.
- **RPO** is the acceptable amount of data loss measured in time before the disaster occurs.
- Disaster Recovery With AWS
    - **Backup and Restore** - storing backup data on S3 and recover data quickly and reliably.
    - **Pilot Light** for Quick Recovery into AWS - quicker recovery time than backup and restore because core pieces of the system are already running and are continually kept up to date.
    - **Warm Standby** Solution - a scaled-down version of a fully functional environment is always running in the cloud
    - **Multi-Site** Solution - run your infrastructure on another site, in an active-active configuration.
    - AWS Production to an AWS DR Solution **Using Multiple AWS Regions** - take advantage of AWS' multiple availability zones
- Services
    - **S3** as a destination for backup data that might be needed quickly to perform a restore.
    - **Import/Export** for transferring very large data sets by shipping storage devices directly to AWS.
    - **Server Migration Service** for performing agentless server migrations from on-premises to AWS.
    - **Database Migration Service and Schema Conversion Tool** for moving databases from on-premises to AWS and automatically converting SQL schema from one engine to another.
    - **Glacier** for longer-term data storage where retrieval times of several hours are adequate.
    - **Storage Gateway** copies snapshots of your on-premises data volumes to S3 for backup. You can create local volumes or EBS volumes from these snapshots.
    - Preconfigured servers bundled as **Amazon Machine Images (AMIs)**.
    - **Elastic Load Balancing (ELB)** for distributing traffic to multiple instances.
    - **Route 53** for routing production traffic to different sites that deliver the same application or service.
    - **Elastic IP address** for static IP addresses.
    - **Virtual Private Cloud (VPC)** for provisioning a private, isolated section of the AWS cloud.
    - **Direct Connect** for a dedicated network connection from your premises to AWS.
    - **Relational Database Service (RDS)** for scale of a relational database in the cloud.
    - **DynamoDB** for a fully managed NoSQL database service to store and retrieve any amount of data and serve any level of request traffic.
    - **Redshift** for a petabyte-scale data warehouse service that analyzes all your data using existing business intelligence tools.
    - **CloudFormation** for creating a collection of related AWS resources and provision them in an orderly and predictable fashion.
    - **Elastic Beanstalk** is a service for deploying and scaling web applications and services developed.
    - **OpsWorks** is an application management service for deploying and operating applications of all types and sizes.

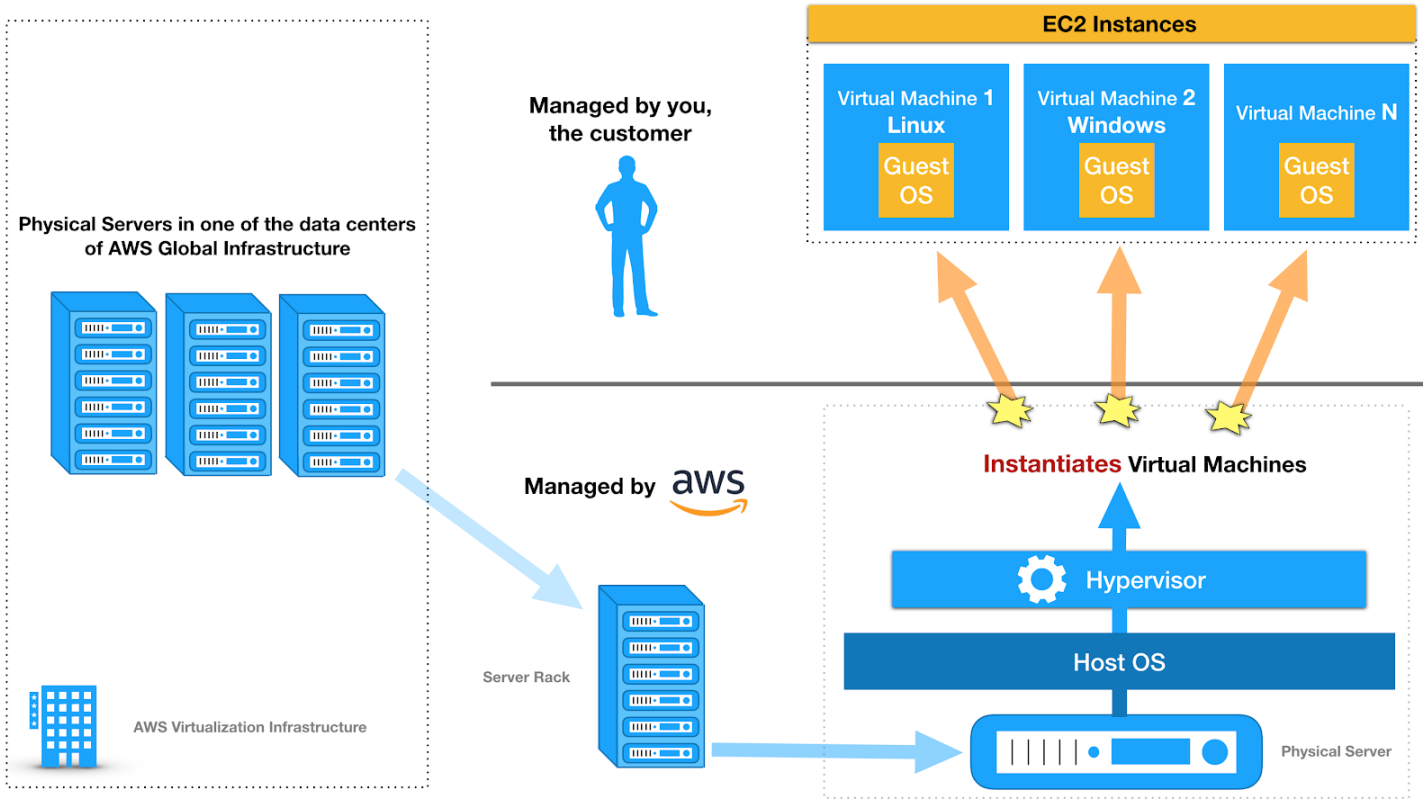
Source:
https://www.slideshare.net/AmazonWebServices/disaster-recovery-options-with-aws

COMPUTE

AWS provides a variety of cost-effective and flexible computing services to meet the needs of your organization such as Amazon Elastic Compute Cloud (EC2), Amazon Elastic Container Service (ECS), Amazon Elastic Container Service for Kubernetes (EKS), Amazon Lightsail, AWS Batch, and AWS Lambda to name a few. For some services like Amazon EC2, you have extensive control of the underlying resources while for others, AWS has full control.

With these computing services in AWS, you can dynamically provision a number of resources and pay only the computing resources you actually consume. This significantly reduces the upfront capital investment required and replaces it with lower variable costs. Instead of the traditional long-term contracts or up-front commitments, you can opt to pay your compute resources in AWS using an On-Demand or Spot pricing option to easily discontinue your cloud resources if you don't need them, effectively reducing your operating expenses. Amazon EC2 is a commonly used AWS service which you can integrate with various features and services like Amazon Machine Image, Instance Store, Elastic Block Store, Elastic Network Interface, Elastic IP, Auto Scaling, Elastic Load Balancer, Placements Groups, Enhanced Networking, Security Groups and so much more.

Have you ever heard people say "Amazon Linux EC2 **Instance**" instead of "Amazon Linux EC2 **Server**" when they launch a compute resource in AWS? It is because AWS is programmatically creating a new virtual machine (VM) **instance**, rather than providing you with an actual physical **server**, when you launch an EC2 Instance. AWS has a powerful virtualization infrastructure that is composed of physical servers that they manage. Each physical server has a host operating system that runs a virtual machine monitor (VMM), also known as a hypervisor, which instantiates multiple VM "instances" that you can use. These instances use guest operating systems that you can manage.



AWS manages, operates, and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the service operates. Conversely, the customer is responsible for the management of the guest operating system such as installing patches and doing the necessary security configuration.

You can also use these compute services in AWS to run your High Performance Computing (HPC) applications. Basically, HPC requires a higher storage I/O and large amounts of memory to perform a complex task. Moving your HPC workloads to AWS eliminates the unnecessary wait times and long job queues that are associated with limited on-premises HPC resources. Since there are no upfront capital expenditures or lengthy procurement cycles, you can get significant cost savings whenever you process time-flexible, stateless workloads.

Amazon EC2

- A Linux-based/Windows-based/Mac-based virtual server that you can provision.
- You are limited to running On-Demand Instances per your vCPU-based On-Demand Instance limit, purchasing 20 Reserved Instances, and requesting Spot Instances per your dynamic Spot limit per region.

Features

- The **AWS Nitro System** is the underlying platform of the next generation of EC2 instances. Traditionally, hypervisors protect the physical hardware and bios, virtualize the CPU, storage, networking, and provide a rich set of management capabilities. With the Nitro System, these functions are offloaded to dedicated hardware and software, thereby reducing costs of your instances in the process. Hence, the Nitro Hypervisor delivers performance that is indistinguishable from bare metal and performs better than its predecessor: the Xen Hypervisor.
- Server environments called **instances**.
- Package OS and additional installations in a reusable template called **Amazon Machine Images.**
- Various configurations of CPU, memory, storage, and networking capacity for your instances, known as **instance types**
    - t-type and m-type for general purpose
    - c-type for compute optimized
    - r-type, x-type and z-type for memory optimized
    - d-type, h-type and i-type for storage optimized
    - f-type, g-type and p-type for accelerated computing
- Secure login information for your instances using **key pairs**
- Storage volumes for temporary data that are deleted when you STOP or TERMINATE your instance, known as **instance store volumes.** Take note that you can stop an EBS-backed instance but not an Instance Store-backed instance. You can only either start or terminate an Instance Store-backed instance.
- Persistent storage volumes for your data using **Elastic Block Store volumes** (see aws storage services).
- Multiple physical locations for deploying your resources, such as instances and EBS volumes, known as **regions** and **Availability Zones** (see AWS overview)**.**
- A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using **security groups** (see aws networking and content delivery).
- Static IPv4 addresses for dynamic cloud computing, known as **Elastic IP addresses** (see aws networking and content delivery).
- Metadata, known as **tags**, that you can create and assign to your EC2 resources
- Virtual networks you can create that are logically isolated from the rest of the AWS cloud, and that you can optionally connect to your own network, known as **virtual private clouds** or **VPC**s (see aws networking and content delivery).
- Add a script that will be run on instance boot called **user-data**.

- **Host Recovery for Amazon EC2** automatically restarts your instances on a new host in the event of an unexpected hardware failure on a Dedicated Host.
- **EC2 Hibernation** is available for On-Demand and Reserved Instances running on freshly launched M3, M4, M5, C3, C4, C5, R3, R4, and R5 instances running Amazon Linux and Ubuntu 18.04 LTS. You can enable hibernation for your EBS-backed instances at launch. You can then hibernate and resume your instances through the AWS Management Console, or through the AWS SDK and CLI using the existing stop-instances and start-instances commands. Hibernation requires an EC2 instance to be an encrypted EBS-backed instance.

Instance States

- **Start** - run your instance normally. You are continuously billed while your instance is running.
- **Stop** - is just a normal instance shutdown. You may restart it again anytime. All EBS volumes remain attached, but data in instance store volumes are deleted. You won't be charged for usage while instance is stopped. You can attach or detach EBS volumes. You can also create an AMI from the instance, and change the kernel, RAM disk, and instance type while in this state.
- **Terminate** - instance performs a normal shutdown and gets deleted. You won't be able to restart an instance once you terminate it. The root device volume is deleted by default, but any attached EBS volumes are preserved by default. Data in instance store volumes are deleted.
- To prevent accidental termination, enable termination protection.

Root Device Volumes

- The root device volume contains the image used to boot the instance.
- Instance Store-backed Instances
    - Any data on the instance store volumes is deleted when the instance is terminated (instance store-backed instances do not support the Stop action) or if it fails (such as if an underlying drive has issues).
    - You should also back up critical data from your instance store volumes to persistent storage on a regular basis.
- Amazon EBS-backed Instances
    - An Amazon EBS-backed instance can be stopped and later restarted without affecting data stored in the attached volumes.
    - When in a stopped state, you can modify the properties of the instance, change its size, or update the kernel it is using, or you can attach your root volume to a different running instance for debugging or any other purpose.
    - By default, the root device volume for an AMI backed by Amazon EBS is deleted when the instance terminates.

AMI

- Includes the following:

    - A template for the root volume for the instance (OS, application server, and applications)
    - Launch permissions that control which AWS accounts can use the AMI to launch instances
    - A block device mapping that specifies the volumes to attach to the instance when it's launched
- Backed by Amazon EBS - root device for an instance launched from the AMI is an Amazon EBS volume. AMIs backed by Amazon EBS snapshots can use EBS encryption.
- Backed by Instance Store - root device for an instance launched from the AMI is an instance store volume created from a template stored in S3.

Characteristic	Amazon EBS-Backed AMI	Amazon Instance Store-Backed AMI
Boot time for an instance	Usually less than 1 minute	Usually less than 5 minutes
Size limit for a root device	16 TiB	10 GiB
Root device volume	Amazon EBS volume	Instance store volume
Data persistence	By default, the root volume is deleted when the instance terminates.* Data on any other Amazon EBS volumes persists after instance termination by default.	Data on any instance store volumes persists only during the life of the instance.
Modifications	The instance type, kernel, RAM disk, and user data can be changed while the instance is stopped.	Instance attributes are fixed for the life of an instance.
Charges	You're charged for instance usage, Amazon EBS volume usage, and storing your AMI as an Amazon EBS snapshot.	You're charged for instance usage and storing your AMI in Amazon S3.
AMI creation/bundling	Uses a single command/call	Requires installation and use of AMI tools
Stopped state	Can be placed in stopped state where instance is not running, but the root volume is persisted in Amazon EBS	Cannot be in stopped state; instances are running or terminated

- You can copy AMIs to different regions.

Pricing

- On-Demand - pay for the instances that you use by the second, with no long-term commitments or upfront payments.
- Reserved - make a low, one-time, up-front payment for an instance, reserve it for a *one*- or *three*-year term, and pay a significantly lower hourly rate for these instances. It has two offering classes: Standard and Convertible.

- The Standard class provides the most significant discount but you can only modify some of its attributes during the term. It can also be sold in the Reserved Instance Marketplace.
- The Convertible class provides a lower discount than Standard Reserved Instances, but can be exchanged for another Convertible Reserved Instance with different instance attributes. However, this one cannot be sold in the Reserved Instance Marketplace.

	Standard RI	Convertible RI
Terms *(average discount off On-Demand)*	1 year (40%) 3 years (60%)	1 year (31%) 3 years (54%)
Change Availability Zone, Instance size (for Linux OS), Networking type	Yes	Yes
Change instance families, operating system, tenancy, and payment option		Yes
Benefit from Price Reductions		Yes

Tutorials Dojo

- Spot - request unused EC2 instances, which can lower your costs significantly. Spot Instances are available at up to a 90% discount compared to On-Demand prices.
  - Spot Instances with a defined duration (also known as **Spot blocks**) are designed not to be interrupted and will run continuously for the duration you select. This makes them ideal for jobs that take a finite time to complete, such as batch processing, encoding and rendering, modeling and analysis, and continuous integration.
  - A **Spot Fleet** is a collection of Spot Instances and optionally On-Demand Instances. The service attempts to launch the number of Spot Instances and On-Demand Instances to meet your specified target capacity. The request for Spot Instances is fulfilled if there is available capacity and the maximum price you specified in the request exceeds the current Spot price. The Spot Fleet also attempts to maintain its target capacity fleet if your Spot Instances are interrupted.
  - A **Spot Instance pool** is a set of unused EC2 instances with the same instance type, operating system, Availability Zone, and network platform.
  - You can start and stop your Spot Instances backed by Amazon EBS at will.
  - Allocation strategy for Spot Instances
    - **LowestPrice** - The Spot Instances come from the pool with the lowest price. This is the default strategy.
    - **Diversified** - The Spot Instances are distributed across all pools.
    - **CapacityOptimized** - The Spot Instances come from the pool with optimal capacity for the number of instances that are launching.

- **InstancePoolsToUseCount** - The Spot Instances are distributed across the number of Spot pools that you specify. This parameter is valid only when used in combination with lowestPrice.

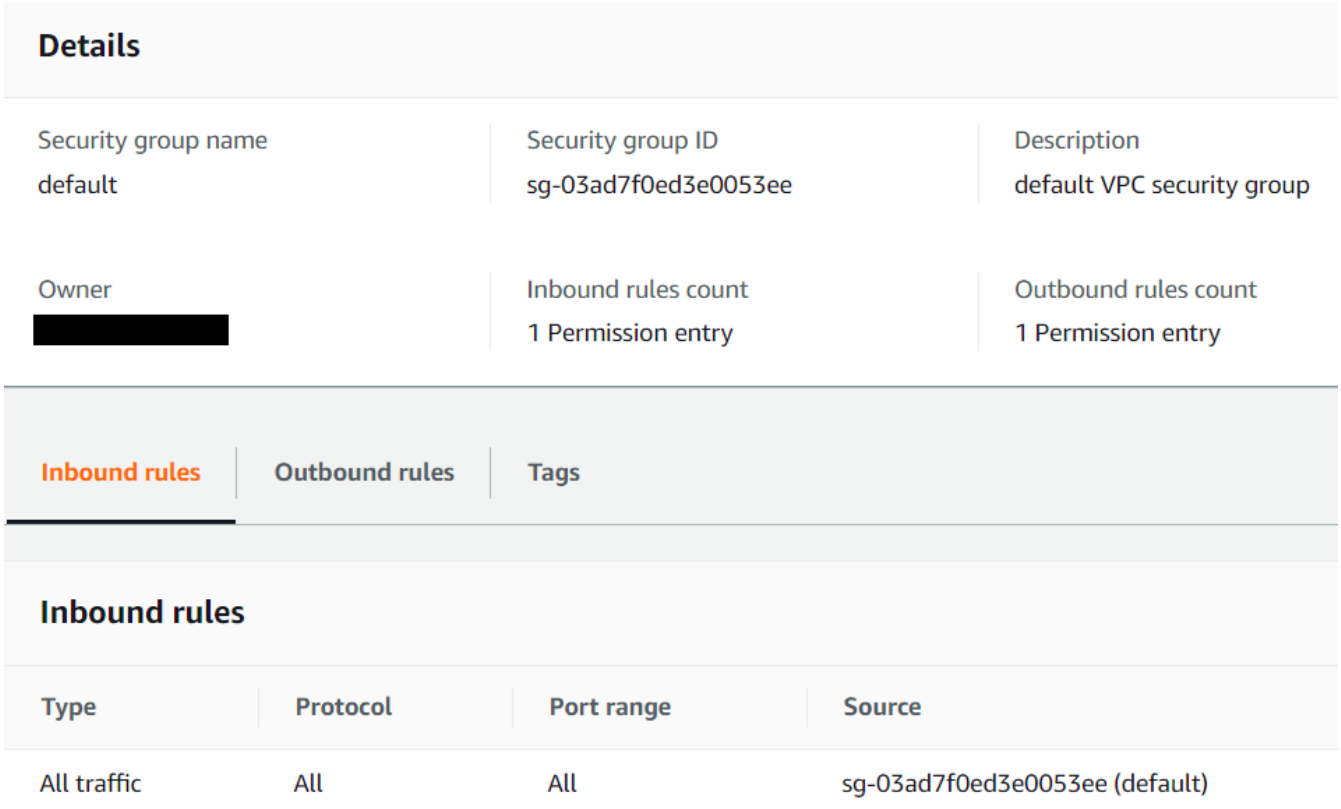
	Spot Instances	On-Demand Instances
Launch time	Can only be launched immediately if the Spot Request is active and capacity is available.	Can only be launched immediately if you make a manual launch request and capacity is available.
Available capacity	If capacity is not available, the Spot Request continues to automatically make the launch request until capacity becomes available.	If capacity is not available when you make a launch request, you get an insufficient capacity error (ICE).
Hourly price	The hourly price for Spot Instances varies based on demand.	The hourly price for On-Demand Instances is static.
Instance interruption	You can't stop and start an Amazon EBS-backed Spot Instance; only the Amazon EC2 Spot service can do this. The Amazon EC2 Spot service can interrupt an individual Spot Instance if capacity is no longer available, the Spot price exceeds your maximum price, or demand for Spot Instances increases.	You determine when an On-Demand Instance is interrupted (stopped or terminated).

- Dedicated Hosts – pay for a physical host that is fully dedicated to running your instances, and bring your existing per-socket, per-core, or per-VM software licenses to reduce costs.
- Dedicated Instances – pay, by the hour, for instances that run on single-tenant hardware.
- There is a data transfer charge when copying AMI from one region to another
- EBS pricing is different from instance pricing. (see AWS storage services)
- AWS imposes a small hourly charge if an Elastic IP address is not associated with a running instance, or if it is associated with a stopped instance or an unattached network interface.
- You are charged for any additional Elastic IP addresses associated with an instance.
- If data is transferred between these two instances, it is charged at "Data Transfer Out from EC2 to Another AWS Region" for the first instance and at "Data Transfer In from Another AWS Region" for the second instance.

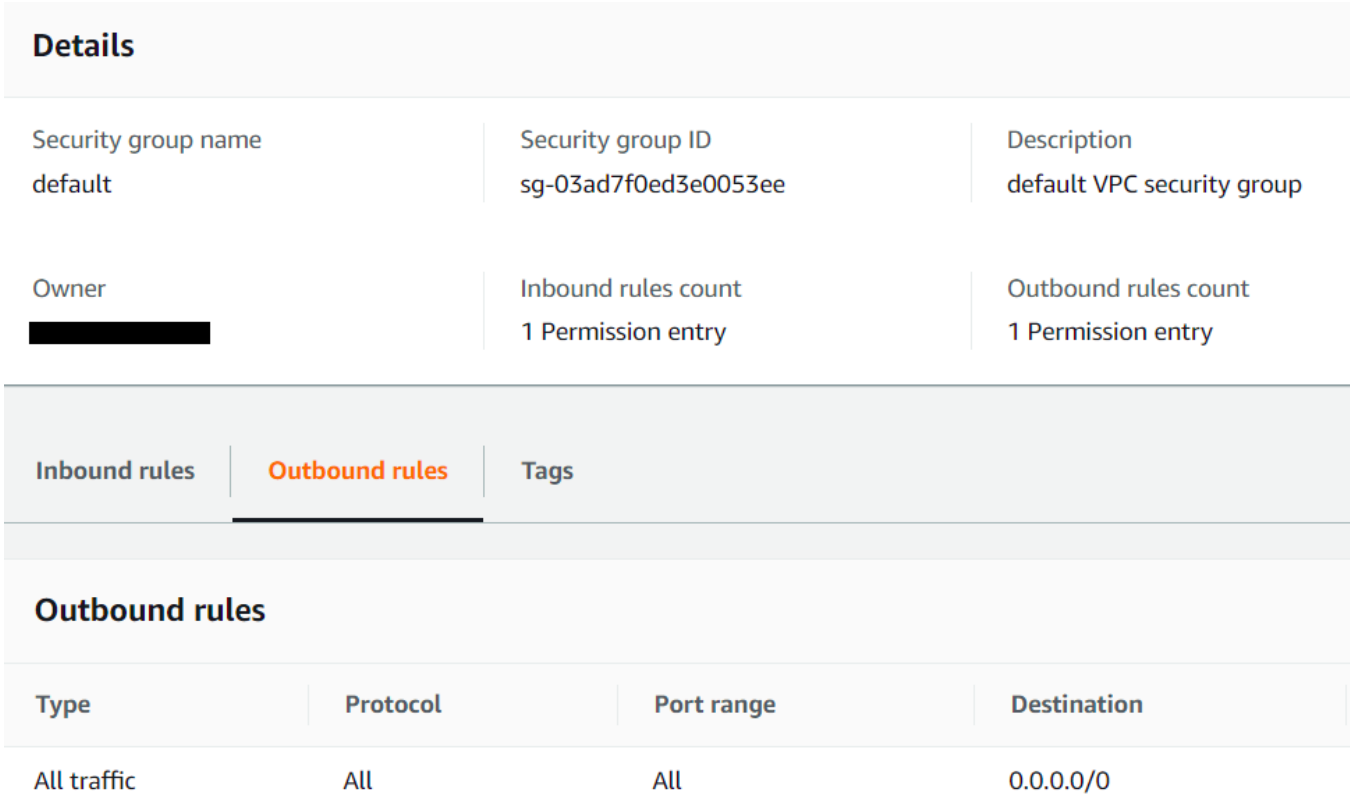
Security

- Use IAM to control access to your instances (see AWS Security and Identity Service).
    - IAM policies

    - IAM roles
- Restrict access by only allowing trusted hosts or networks to access ports on your instance.
- A **security group** acts as a virtual firewall that controls the traffic for one or more instances.
    - Create different security groups to deal with instances that have different security requirements.
    - You can add rules to each security group that allow traffic to or from its associated instances.
    - You can modify the rules for a security group at any time.
    - New rules are automatically applied to all instances that are associated with the security group.
    - Evaluates all the rules from all the security groups that are associated with an instance to decide whether to allow traffic or not.
    - By default, security groups allow **all outbound traffic**.
    - Security group rules are **always permissive**; you can't create rules that deny access.
    - Security groups are **stateful**
- If you don't specify a security group when you launch an instance, the instance is automatically associated with the **default security group** for the VPC, which has the following rules:
    - Allows all inbound traffic from other instances associated with the default security group



    - Allows all outbound traffic from the instance.



Details

Security group name	Security group ID	Description
default	sg-03ad7f0ed3e0053ee	default VPC security group
Owner	**Inbound rules count**	**Outbound rules count**
[redacted]	1 Permission entry	1 Permission entry

Inbound rules | Outbound rules | Tags

Outbound rules

Type	Protocol	Port range	Destination
All traffic	All	All	0.0.0.0/0

- Disable password-based logins for instances launched from your AMI, since passwords can be cracked or found.
- You can replicate the network traffic from an EC2 instance within your Amazon VPC and forward that traffic to security and monitoring appliances for content inspection, threat monitoring, and troubleshooting.

Networking

- An **Elastic IP address** is a static IPv4 address designed for dynamic cloud computing. With it, you can mask the failure of an instance or software by rapidly remapping the address to another instance in your account.
- If you have not enabled auto-assign public IP address for your instance, you need to associate an Elastic IP address with your instance to enable communication with the internet.
- An Elastic IP address is for use in a specific region only.
- By default, all AWS accounts are limited to five (5) Elastic IP addresses per region, because public (IPv4) internet addresses are a scarce public resource.
- By default EC2 instances come with a private IP.
- An elastic **network interface** is a logical networking component in a VPC that represents a virtual network card, which directs traffic to your instance

- Every instance in a VPC has a default network interface, called the **primary network interface** (eth0). You cannot detach a primary network interface from an instance.
- You can create and attach additional network interfaces. The maximum number of network interfaces that you can use varies by instance type.
- You can attach a network interface to an instance in a different subnet as long as its within the same AZ
- Default interfaces are terminated with instance termination.
- Scale with **EC2 Scaling Groups** and distribute traffic among instances using **Elastic Load Balancer**.
- You can configure EC2 instances as **bastion hosts** (aka jump boxes) in order to access your VPC instances for management, using SSH or RDP protocols

Monitoring

- EC2 items to monitor
    - CPU utilization, Network utilization, Disk performance, Disk Reads/Writes using EC2 metrics
    - Memory utilization, disk swap utilization, disk space utilization, page file utilization, log collection using a monitoring agent/CloudWatch Logs
- Automated monitoring tools include:
    - System Status Checks - monitor the AWS systems required to use your instance to ensure they are working properly. These checks detect problems with your instance that require AWS involvement to repair.
    - Instance Status Checks - monitor the software and network configuration of your individual instance. These checks detect problems that require your involvement to repair.
    - Amazon CloudWatch Alarms - watch a single metric over a time period you specify, and perform one or more actions based on the value of the metric relative to a given threshold over a number of time periods.
    - Amazon CloudWatch Events - automate your AWS services and respond automatically to system events.
    - Amazon CloudWatch Logs - monitor, store, and access your log files from Amazon EC2 instances, AWS CloudTrail, or other sources.
- Monitor your EC2 instances with CloudWatch. By default, EC2 sends metric data to CloudWatch in 5-minute periods.
- You can also enable detailed monitoring to collect data in 1-minute periods.

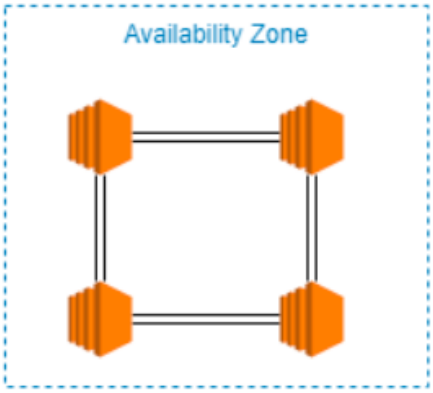
Instance Metadata and User Data

- **Instance metadata** is data about your instance that you can use to configure or manage the running instance.
- Instance metadata and user data are not protected by cryptographic methods.
- View all categories of instance metadata from within a running instance at **http://169.254.169.254/latest/meta-data/**

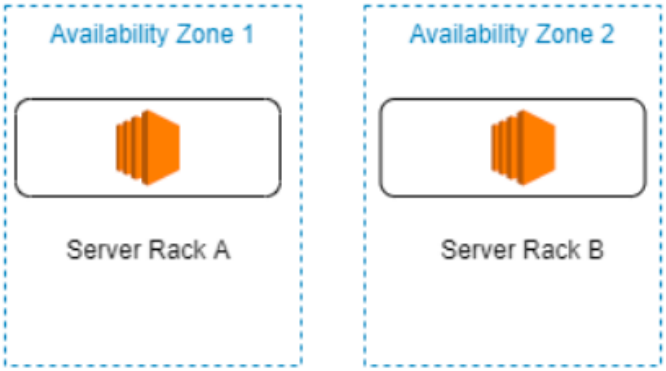
- You can pass two types of user data to EC2: shell scripts and cloud-init directives.
- User data is limited to 16 KB.
- If you stop an instance, modify its user data, and start the instance, the updated user data is not executed when you start the instance.
- Retrieve user data from within a running instance at **http://169.254.169.254/latest/user-data**

Placement Groups

- You can launch or start instances in a **placement group**, which determines how instances are placed on underlying hardware.
    - Cluster - clusters instances into a low-latency group in a single Availability Zone. Recommended for applications that benefit from low network latency, high network throughput, or both, and if the majority of the network traffic is between the instances in the group.
    - Spread - spreads instances across underlying hardware. Recommended for applications that have a small number of critical instances that should be kept separate from each other. Note: A spread placement group can span multiple Availability Zones, and you can have a maximum of seven running instances per Availability Zone per group.
- Partition placement groups is an Amazon EC2 placement strategy that helps reduce the likelihood of correlated failures for large distributed and replicated workloads such as HDFS, HBase and Cassandra running on EC2.



Cluster Placement



Spread Placement

- Partition placement groups spread EC2 instances across logical partitions and ensure that instances in different partitions do not share the same underlying hardware. In addition, partition placement groups offer visibility into the partitions and allow topology aware applications to use this information to make intelligent data replication decisions, increasing data availability and durability.

Rules

- The name you specify for a placement group must be unique within your AWS account for the region.
- You can't merge placement groups.
- An instance can be launched in one placement group at a time; it cannot span multiple placement groups.
- Instances with a tenancy of host cannot be launched in placement groups.

Storage

- **EBS** (see AWS Storage Services)
  - Provides durable, block-level storage volumes that you can attach to a running instance.
  - Use as a primary storage device for data that requires frequent and granular updates.
  - To keep a backup copy of your data, create a snapshot of an EBS volume, which is stored in S3. You can create an EBS volume from a snapshot, and attach it to another instance.
- **Instance Store**
  - Provides temporary block-level storage for instances.
  - The data on an instance store volume persists only during the life of the associated instance; if you stop or terminate an instance, any data on instance store volumes is lost.
- **Elastic File System (EFS)** (see AWS Storage Services)
  - Provides scalable file storage for use with Amazon EC2. You can create an EFS file system and configure your instances to mount the file system.
  - You can use an EFS file system as a common data source for workloads and applications running on multiple instances.
- **S3** (see AWS Storage Services)
  - Provides access to reliable and inexpensive data storage infrastructure.
  - Storage for EBS snapshots and instance store-backed AMIs.
- **Resources and Tagging**
  - EC2 resources include images, instances, volumes, and snapshots. When you create a resource, AWS assigns the resource a *unique resource ID*.
  - Some resources can be used in all regions (global), and some resources are specific to the region or Availability Zone in which they reside.

Resource	Type	Description
AWS account	Global	You can use the same AWS account in all regions.
Key pairs	Global or Regional	The key pairs that you create using EC2 are tied to the region where you created them. You can create your own RSA key pair and upload it to the region in which you want to use it; therefore, you can make your key pair globally available by uploading it to each region.

Amazon EC2 resource identifiers	Regional	Each resource identifier, such as an AMI ID, instance ID, EBS volume ID, or EBS snapshot ID, is tied to its region and can be used only in the region where you created the resource.
User-supplied resource names	Regional	Each resource name, such as a security group name or key pair name, is tied to its region and can be used only in the region where you created the resource. Although you can create resources with the same name in multiple regions, they aren't related to each other.
AMIs	Regional	An AMI is tied to the region where its files are located within S3. You can copy an AMI from one region to another.
Elastic IP addresses	Regional	An Elastic IP address is tied to a region and can be associated only with an instance in the same region.
Security groups	Regional	A security group is tied to a region and can be assigned only to instances in the same region. You can't enable an instance to communicate with an instance outside its region using security group rules.
EBS snapshots	Regional	An EBS snapshot is tied to its region and can only be used to create volumes in the same region. You can copy a snapshot from one region to another.
EBS volumes	Availability Zone	An EBS volume is tied to its Availability Zone and can be attached only to instances in the same Availability Zone.
Instances	Availability Zone	An instance is tied to the Availability Zones in which you launched it. However, its instance ID is tied to the region.

    - You can optionally assign your own metadata to each resource with **tags**, which consists of a key and an optional value that you both define.

Networking

- **Enhanced Networking** - It provides higher bandwidth, higher packet per second (PPS) performance, and consistent lower inter-instance latencies, which is being used in Placement Groups. It uses single root I/O virtualization (SR-IOV) to provide high-performance networking capabilities. SR-IOV is a method of

device virtualization that provides higher I/O performance and lower CPU utilization when compared to traditional virtualized network interfaces.

- **Elastic Fabric Adapter (EFA) -** This is a network device that you can attach to your EC2 instance to significantly accelerate machine learning applications and High Performance Computing (HPC). It empowers your computing resources to achieve the application performance of an on-premises HPC cluster, with the elasticity and scalability provided by AWS. Compared with a TCP transport that is traditionally used in cloud-based HPC systems, EFA provides lower and more consistent latency and higher throughput as it enhances the performance of inter-instance communication.

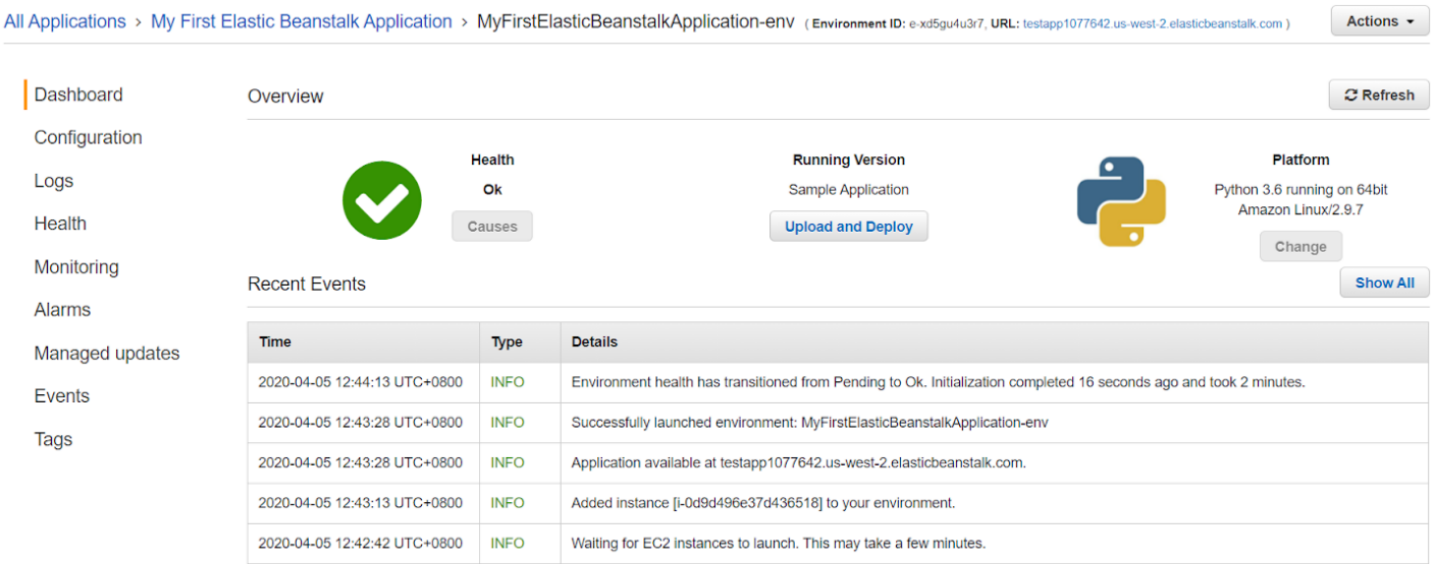
Sources:
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/
https://aws.amazon.com/ec2/features/
https://aws.amazon.com/ec2/pricing/
https://aws.amazon.com/ec2/faqs/

AWS Elastic Beanstalk

- Allows you to quickly deploy and manage applications in the AWS Cloud without worrying about the infrastructure that runs those applications.
- Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring for your applications.
- It is a Platform-as-a-Service
- Elastic Beanstalk supports the following languages:
    - Go
    - Java
    - .NET
    - Node.js
    - PHP
    - Python
    - Ruby
- Elastic Beanstalk supports the following web containers:
    - Tomcat
    - Passenger
    - Puma
- Elastic Beanstalk supports Docker containers.
- Your application's domain name is in the format: *subdomain.region*.elasticbeanstalk.com

Environment Pages

- The **Configuration** page shows the resources provisioned for this environment. This page also lets you configure some of the provisioned resources.
- The **Health** page shows the status and detailed health information about the EC2 instances running your application.
- The **Monitoring** page shows the statistics for the environment, such as average latency and CPU utilization. You also use this page to create alarms for the metrics that you are monitoring.
- The **Events** page shows any informational or error messages from services that this environment is using.
- The **Tags** page shows tags — key-value pairs that are applied to resources in the environment. You use this page to manage your environment's tags.



Elastic Beanstalk Concepts

- **Application** - a logical collection of Elastic Beanstalk components, including environments, versions, and environment configurations. It is conceptually similar to a folder.
- **Application Version** - refers to a specific, labeled iteration of deployable code for a web application. An application version points to an Amazon S3 object that contains the deployable code. Applications can have many versions and each application version is unique.
- **Environment** - a version that is deployed on to AWS resources. Each environment runs only a single application version at a time, however you can run the same version or different versions in many environments at the same time.
- **Environment Tier** - determines whether Elastic Beanstalk provisions resources to support an application that handles HTTP requests or an application that pulls tasks from a queue. An application that serves HTTP requests runs in a **web server environment**. An environment that pulls tasks from an Amazon SQS queue runs in a **worker environment**.
- **Environment Configuration** - identifies a collection of parameters and settings that define how an environment and its associated resources behave.
- Configuration Template - a starting point for creating unique environment configurations.
- There is a limit to the number of application versions you can have. You can avoid hitting the limit by applying an *application version lifecycle policy* to your applications to tell Elastic Beanstalk to delete application versions that are old, or to delete application versions when the total number of versions for an application exceeds a specified number.

Environment Types

- Load-balancing, Autoscaling Environment - automatically starts additional instances to accommodate increasing load on your application.

- Single-Instance Environment - contains one Amazon EC2 instance with an Elastic IP address.

Environment Configurations

- Your environment contains:
  - Your **EC2 virtual machines** configured to run web apps on the platform that you choose.
  - An **Auto Scaling group** that ensures that there is always one instance running in a single-instance environment, and allows configuration of the group with a range of instances to run in a load-balanced environment.
  - When you enable load balancing, Elastic Beanstalk creates an **Elastic Load Balancing load balancer** to distributes traffic among your environment's instances.
  - Elastic Beanstalk provides integration with **Amazon RDS** to help you add a database instance to your Elastic Beanstalk environment : **MySQL, PostgreSQL, Oracle, or SQL Server**. When you add a database instance to your environment, Elastic Beanstalk provides connection information to your application by setting environment properties for the database hostname, port, user name, password, and database name.
  - You can use **environment properties** to pass secrets, endpoints, debug settings, and other information to your application. Environment properties help you run your application in multiple environments for different purposes, such as development, testing, staging, and production.
  - You can configure your environment to use **Amazon SNS** to notify you of important events that affect your application.
  - Your environment is available to users at a **subdomain of elasticbeanstalk.com**. When you create an environment, you can choose a unique subdomain that represents your application.
- Configuration changes that modify the launch configuration or VPC settings require terminating all instances in your environment and replacing them.
  - **Rolling updates** - to prevent downtime, Elastic Beanstalk applies your configuration changes in batches, keeping a minimum number of instances running and serving traffic at all times.
  - **Immutable updates** - a temporary Auto Scaling group is launched outside of your environment with a separate set of instances running on the new configuration, which are placed behind your environment's load balancer. Old and new instances both serve traffic until the new instances pass health checks, at which time the new instances are moved into your environment's Auto Scaling group and the temporary group and old instances are terminated.
- You can cancel in-progress updates that are triggered by environment configuration changes. You can also cancel the deployment of a new application version in progress. Elastic Beanstalk will perform a rollback. You cannot stop Elastic Beanstalk from rolling back to the previous environment configuration once it begins to cancel the update.

Worker Environments

- If your AWS Elastic Beanstalk application performs operations or workflows that take a long time to complete, you can offload those tasks to a dedicated worker environment.

- To avoid running long-running tasks locally, you can use the AWS SDK for your programming language to send them to an Amazon Simple Queue Service (Amazon SQS) queue, and run the process that performs them on a separate set of instances. Worker instances can take items from the queue when they have the capacity to run them.
- Worker environments run a **daemon process** provided by Elastic Beanstalk. The daemon is responsible for managing your SQS messages.
- You can define periodic tasks in a file named **cron.yaml** in your source bundle to add jobs to your worker environment's queue automatically at a regular interval. If you configure your worker environment with an Amazon SQS **FIFO queue**, periodic tasks aren't supported.
- Elastic Beanstalk worker environments support Amazon SQS dead-letter queues. A **dead-letter queue** is a queue where other (source) queues can send messages that for some reason could not be successfully processed.

Deployments

- Deployment Policies
    - **All at once** - this is the default deployment used by your environment.
    - **Rolling** - Elastic Beanstalk splits the environment's Amazon EC2 instances into batches and deploys the new version of the application to one batch at a time. So some of your instances will serve the old version of your application while the deployment is occurring.
    - **Rolling with additional batch** - configure your environment to launch a new batch of instances before taking any instances out of service to maintain full capacity during deployments.
    - **Immutable** - launch a full set of new instances running the new version of the application in a separate Auto Scaling group, alongside the instances running the old version. Immutable deployments can prevent issues caused by partially completed rolling deployments.
    - **Traffic splitting** - lets you perform canary testing as part of your application deployment. Elastic Beanstalk launches a full set of new instances, then forwards a specified percentage of incoming client traffic to the new application version for a specified evaluation period. If the new instances stay healthy, Elastic Beanstalk forwards all traffic to them and terminates the old ones. If the new instances don't pass health checks, or if you choose to abort the deployment, Elastic Beanstalk moves traffic back to the old instances and terminates the new ones.
- You can add AWS Elastic Beanstalk configuration files (.ebextensions) to your web application's source code to configure your environment and customize the AWS resources that it contains. Configuration files are YAML- or JSON-formatted documents with a .config file extension that you place in a folder named .ebextensions and deploy in your application source bundle.

AWS Elastic Beanstalk Deployment Methods						
Method	**Impact of failed deployment**	**Deploy time**	**Zero downtime**	**No DNS change**	**Rollback process**	**Code deployed to**
All at once	Downtime	FASTEST	⊗ No	✓ Yes	Manual redeploy	Existing instances
Rolling	Single batch out of service; any successful batches before failure running new application version	SLOWER than All-at-Once Method *	✓ Yes	✓ Yes	Manual redeploy	Existing instances
Rolling with an additional batch	Minimal if first batch fails; otherwise, similar to **Rolling**	SLOWER than Rolling Method *	✓ Yes	✓ Yes	Manual redeploy	New and existing instances
Immutable	Minimal	SLOW	✓ Yes	✓ Yes	Terminate new instances	New instances
Traffic splitting	Percentage of client traffic routed to new version temporarily impacted	SLOW **	✓ Yes	✓ Yes	Reroute traffic and terminate new instances	New instances
Blue/green	Minimal	SLOW	✓ Yes	⊗ No	Swap URL	New instances

** Varies depending on batch size.*
*** Varies depending on evaluation time option setting.*

Tutorials Dojo Tutorials Dojo

Blue/Green deployments with Elastic Beanstalk

- You can avoid downtime by performing a blue/green deployment, where you deploy the new version to a separate environment, and then swap the CNAMEs of the two environments to redirect traffic to the new version instantly.
- After Elastic Beanstalk completes the swap operation, do not immediately terminate your old environment until the DNS changes are propagated and your old DNS records expire.

CDA Exam Notes:

So when to use Elastic Beanstalk vs other compute options?

The benefit of using Elastic Beanstalk is that it is PaaS. This means that you just have to define your environment settings and AWS handles the provisioning to the integration for you. After that, you can simply upload your application and you would already have a fully working environment. Management is easy and you do not need to understand how different services work together before you can launch your application. That being said, it also limits your environment depending on whether Elastic Beanstalk supports the configuration you want or not.

Monitoring

- Elastic Beanstalk Monitoring console displays your environment's status and application health at a glance.
- Elastic Beanstalk reports the health of a web server environment depending on how the application running in it responds to the health check.
- **Enhanced health reporting** is a feature that you can enable on your environment to allow AWS Elastic Beanstalk to gather additional information about resources in your environment. Elastic Beanstalk analyzes the information gathered to provide a better picture of overall environment health and aid in the identification of issues that can cause your application to become unavailable.
- You can create alarms for metrics to help you monitor changes to your environment so that you can easily identify and mitigate problems before they occur.
- EC2 instances in your Elastic Beanstalk environment generate logs that you can view to troubleshoot issues with your application or configuration files.

Security

- When you create an environment, Elastic Beanstalk prompts you to provide two AWS IAM roles: a **service role** and an **instance profile**.
    - Service Roles - assumed by Elastic Beanstalk to use other AWS services on your behalf.
    - Instance Profiles - applied to the instances in your environment and allows them to retrieve application versions from S3, upload logs to S3, and perform other tasks that vary depending on the environment type and platform.
- User Policies - allow users to create and manage Elastic Beanstalk applications and environments.

Pricing

- There is no additional charge for Elastic Beanstalk. You pay only for the underlying AWS resources that your application consumes.

Sources:
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg
https://aws.amazon.com/elasticbeanstalk/details/
https://aws.amazon.com/elasticbeanstalk/pricing/
https://aws.amazon.com/elasticbeanstalk/faqs/

AWS Lambda

- A serverless compute service.
- Lambda executes your code only when needed and scales automatically.
- Lambda functions are stateless - no affinity to the underlying infrastructure.
- You choose the amount of memory you want to allocate to your functions and AWS Lambda allocates proportional CPU power, network bandwidth, and disk I/O.
- AWS Lambda is SOC, HIPAA, PCI, ISO compliant.
- Natively supports the following languages:
    - Node.js
    - Java
    - C#
    - Go
    - Python
    - Ruby
    - PowerShell
- You can also provide your own custom runtime.

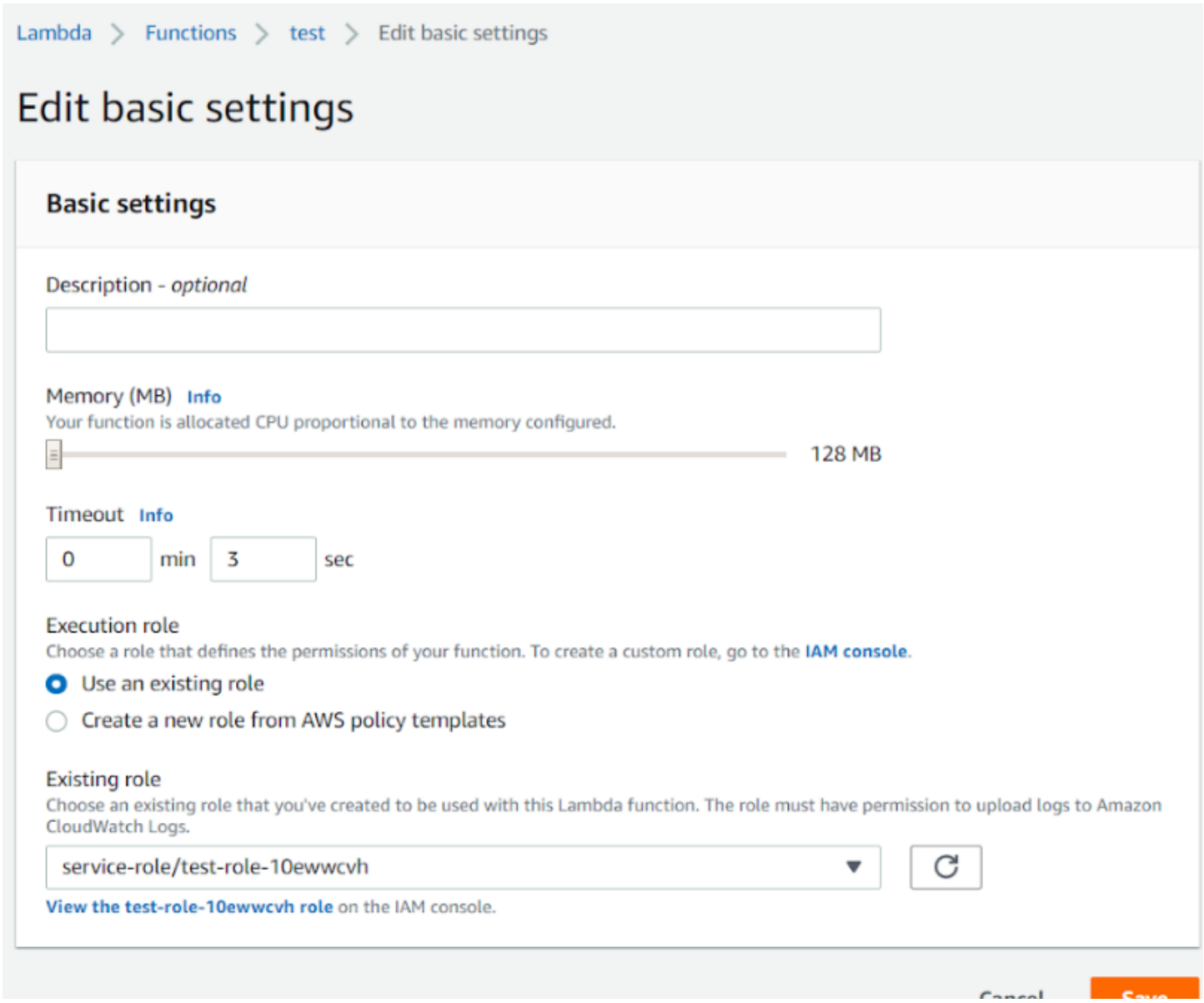
Components of a Lambda Application

- **Function** – a script or program that runs in Lambda. Lambda passes invocation events to your function. The function processes an event and returns a response.
- **Runtimes** – Lambda runtimes allow functions in different languages to run in the same base execution environment. The runtime sits in-between the Lambda service and your function code, relaying invocation events, context information, and responses between the two.
- **Layers** – Lambda layers are a distribution mechanism for libraries, custom runtimes, and other function dependencies. Layers let you manage your in-development function code independently from the unchanging code and resources that it uses.
- **Event source** – an AWS service or a custom service that triggers your function and executes its logic.
- **Downstream resources** – an AWS service that your Lambda function calls once it is triggered.
- **Log streams** – While Lambda automatically monitors your function invocations and reports metrics to CloudWatch, you can annotate your function code with custom logging statements that allow you to analyze the execution flow and performance of your Lambda function.
- AWS Serverless Application Model

Lambda Functions

- You upload your application code in the form of one or more *Lambda functions*. Lambda stores code in Amazon S3 and encrypts it at rest.
- To create a Lambda function, you first package your code and dependencies in a deployment package. Then, you upload the deployment package to create your Lambda function.

- After your Lambda function is in production, Lambda automatically monitors functions on your behalf, reporting metrics through Amazon CloudWatch.
- Configure **basic function settings** including the description, memory usage, execution timeout, and role that the function will use to execute your code.



- **Environment variables** are always encrypted at rest, and can be encrypted in transit as well.

CDA Exam Notes:

Should you use Lambda's environment variables? Or should you use Parameter Store? Or Secrets Manager?

The answer, as always, depends on your use case. Each of them has their own set of benefits, but generally you should be using the built-in environment variables. It is free to use and it works as if you are creating environment variables or aliases in a terminal. You also have the option to encrypt your variables using AWS KMS, but that entails additional charges.

You can likewise use Parameter Store if you have variables that need to be referenced not just from a single Lambda function. For example, you want a centralized service where you can update environment variables in one go. It's also free and, similarly, you can use SecureString datatype to have KMS encrypt your values.

The main benefit of using Secrets Manager is that you can rotate your keys or variables automatically. This is typically used in setups that involve a database or access key credential. Secrets Manager is a paid service, however, so unless you have secrets to rotate, better go with the free services.

- **Versions and aliases** are secondary resources that you can create to manage function deployment and invocation.
- A **layer** is a ZIP archive that contains libraries, a custom runtime, or other dependencies. Use layers to manage your function's dependencies independently and keep your deployment package small.
- You can configure a function to mount an Amazon EFS file system to a local directory. With Amazon EFS, your function code can access and modify shared resources securely and at high concurrency.

Ways to Upload your Code

- There are three ways to add your code into your Lambda function:
    - Add code inline through the function's IDE - enter your function code as is in the built-in editor. Choose this option if your code is short or it does not have any external package dependencies.
    - Upload a zip file of your code which will be stored in S3 - upload a zip file which contains your code. Use this option if your code has package dependencies or if your code is long. You need to ensure a few things when using this option:
        - The filename of your code should match your lambda handler's filename part (*filename.handler_function)*
        - Your *main* function should have the same function name as your lambda handler's handler function part (*filename.handler_function)*

        - Your file should be executable by your function's programming language. Verify the filename extension.
    - Reference a path to an S3 bucket where your code is stored - if you already store your function code in S3, you can just reference it in your lambda function. This way, you can easily build a CI/CD pipeline and your lambda functions will run the latest code that you upload in your S3 bucket. This option has the same requirements as uploading a zip file.

Invoking Functions

- Lambda supports **synchronous** and **asynchronous invocation** of a Lambda function. You can control the invocation type only when you invoke a Lambda function (referred to as *on-demand invocation*).
- An **event source** is the entity that publishes events, and a Lambda function is the custom code that processes the events.
- *Event source mapping* maps an event source to a Lambda function. It enables automatic invocation of your Lambda function when events occur. Lambda provides event source mappings for the following services.
    - Amazon Kinesis
    - Amazon DynamoDB
    - Amazon Simple Queue Service
- Your functions' *concurrency* is the number of instances that serve requests at a given time. When your function is invoked, Lambda allocates an instance of it to process the event. When the function code finishes running, it can handle another request. If the function is invoked again while a request is still being processed, another instance is allocated, which increases the function's concurrency.
- To ensure that a function can always reach a certain level of concurrency, you can configure the function with **reserved concurrency**. When a function has reserved concurrency, no other function can use that concurrency. Reserved concurrency also limits the maximum concurrency for the function.
- To enable your function to scale without fluctuations in latency, use **provisioned concurrency**. By allocating provisioned concurrency before an increase in invocations, you can ensure that all requests are served by initialized instances with very low latency.

CDA Exam Notes:

What is an execution context?

When AWS Lambda executes your Lambda function, it provisions and manages the resources needed to run your Lambda function. The execution context is a temporary runtime environment that initializes any external dependencies of your Lambda function, such as database connections or HTTP endpoints. This provides better performance for your subsequent invocations because there is no need to re-initialize those external dependencies (cold start).

After a Lambda function is executed, AWS Lambda maintains the execution context for some time in anticipation of another Lambda function invocation. This way, you can design your functions such that they can verify if there are existing database connections or HTTP endpoints that are reusable.

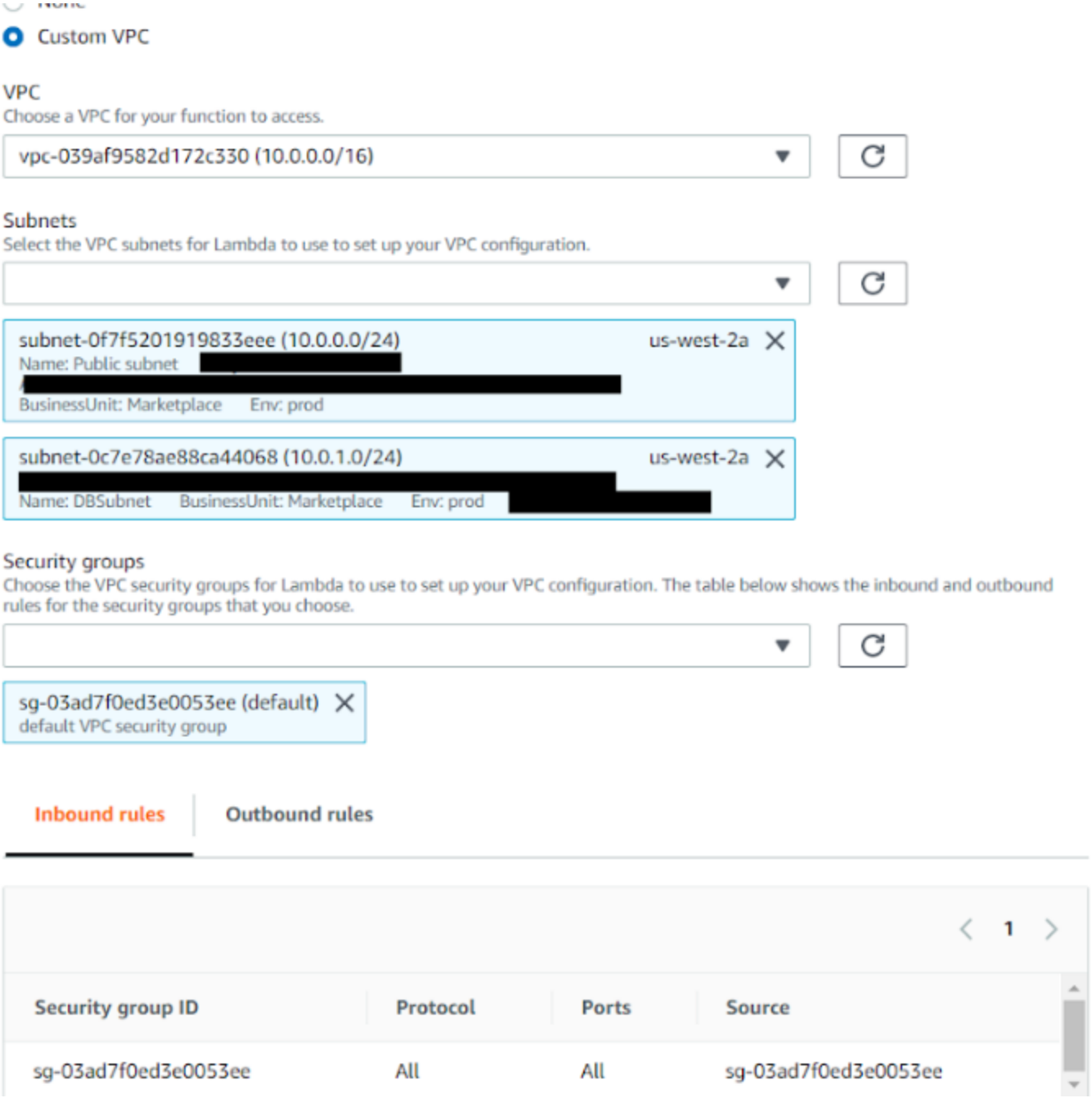
Each execution context also provides 512 MB of additional disk space in the /tmp directory. The directory content remains when the execution context is frozen, similar to a cache.

You should ensure that any background processes or callbacks in your code are complete before the code exits so that subsequent invocations won't resume any unfinished processes.

Configuring a Lambda Function to Access Resources in a VPC

In AWS Lambda, you can set up your function to establish a connection to your virtual private cloud (VPC). With this connection, your function can access the private resources of your VPC during execution like EC2, RDS and many others.

By default, AWS executes your Lambda function code securely within a VPC. Alternatively, you can enable your Lambda function to access resources inside your private VPC by providing additional VPC-specific configuration information such as VPC subnet IDs and security group IDs. It uses this information to set up elastic network interfaces which enable your Lambda function to connect securely to other resources within your VPC.



Lambda@Edge

- Lets you run Lambda functions to customize content that CloudFront delivers, executing the functions in AWS locations closer to the viewer. The functions run in response to CloudFront events, without provisioning or managing servers.
- You can use Lambda functions to change CloudFront requests and responses at the following points:

    - After CloudFront receives a request from a viewer (viewer request)
    - Before CloudFront forwards the request to the origin (origin request)
    - After CloudFront receives the response from the origin (origin response)
    - Before CloudFront forwards the response to the viewer (viewer response)
- You can automate your serverless application's release process using AWS CodePipeline and AWS CodeDeploy.
- Lambda will automatically track the behavior of your Lambda function invocations and provide feedback that you can monitor. In addition, it provides metrics that allows you to analyze the full function invocation spectrum, including event source integration and whether downstream resources perform as expected.

Pricing

- You are charged based on the total number of requests for your functions and the duration, the time it takes for your code to execute.

Sources:
https://docs.aws.amazon.com/lambda/latest/dg
https://aws.amazon.com/lambda/features/
https://aws.amazon.com/lambda/pricing/
https://aws.amazon.com/lambda/faqs/

Amazon Elastic Container Service (ECS)

- A container management service to run, stop, and manage Docker containers on a cluster.
- ECS can be used to create a consistent deployment and build experience, manage, and scale batch and **Extract-Transform-Load** (ETL) workloads, and build sophisticated application architectures on a microservices model.
- Amazon ECS is a regional service.

Features

- You can create ECS clusters within a new or existing VPC.
- After a cluster is up and running, you can define task definitions and services that specify which Docker container images to run across your clusters.
- AWS Compute SLA guarantees a Monthly Uptime Percentage of at least 99.99% for Amazon ECS.

Components

- Containers and Images
    - Your application components must be architected to run in **containers** — containing everything that your software application needs to run: code, runtime, system tools, system libraries, etc.
    - Containers are created from a read-only template called an **image**.
    - Images are typically built from a **Dockerfile**, a plain text file that specifies all of the components that are included in the container. These images are then stored in a **registry** from which they can be downloaded and run on your cluster.
    - When you launch a container instance, you have the option of passing *user data* to the instance. The data can be used to perform common automated configuration tasks and even run scripts when the instance boots.
    - Docker Volumes can be a local instance store volume, EBS volume or EFS volume. Connect your Docker containers to these volumes using Docker drivers and plugins.
- Task Components
    - **Task definitions** specify various parameters for your application. It is a text file, in JSON format, that describes one or more containers, up to a maximum of ten, that form your application.
    - Task definitions are split into separate parts:
        - Task family - the name of the task, and each family can have multiple revisions.
        - IAM task role - specifies the permissions that containers in the task should have.
        - Network mode - determines how the networking is configured for your containers.
        - Container definitions - specify which image to use, how much CPU and memory the container are allocated, and many more options.
        - Volumes - allow you to share data between containers and even persist the data on the container instance when the containers are no longer running.
        - Task placement constraints - lets you customize how your tasks are placed within the infrastructure.

        - Launch types - determines which infrastructure your tasks use.
- Tasks and Scheduling
    - A **task** is the instantiation of a task definition within a cluster. After you have created a task definition for your application, you can specify the number of tasks that will run on your cluster.
        - Each task that uses the Fargate launch type has its own isolation boundary and does not share the underlying kernel, CPU resources, memory resources, or elastic network interface with another task.
    - The **task scheduler** is responsible for placing tasks within your cluster. There are several different scheduling options available.
        - REPLICA — places and maintains the desired number of tasks across your cluster. By default, the service scheduler spreads tasks across Availability Zones. You can use task placement strategies and constraints to customize task placement decisions.
        - DAEMON — deploys exactly one task on each active container instance that meets all of the task placement constraints that you specify in your cluster. When using this strategy, there is no need to specify a desired number of tasks, a task placement strategy, or use Service Auto Scaling policies.
    - You can upload a new version of your application task definition, and the ECS scheduler automatically starts new containers using the updated image and stop containers running the previous version.
    - Amazon ECS tasks running on both Amazon EC2 and AWS Fargate can mount Amazon Elastic File System (EFS) file systems.
- Clusters
    - When you run tasks using ECS, you place them in a **cluster**, which is a logical grouping of resources.
    - Clusters are Region-specific.
    - Clusters can contain tasks using both the Fargate and EC2 launch types.
    - When using the Fargate launch type with tasks within your cluster, ECS manages your cluster resources.
    - When using the EC2 launch type, then your clusters are a group of container instances you manage. These clusters can contain multiple different container instance types, but each container instance may only be part of one cluster at a time.
    - Before you can delete a cluster, you must delete the services and deregister the container instances inside that cluster.
    - Enabling managed Amazon ECS cluster auto scaling allows ECS to manage the scale-in and scale-out actions of the Auto Scaling group. On your behalf, Amazon ECS creates an AWS Auto Scaling scaling plan with a target tracking scaling policy based on the target capacity value that you specify.
- Services
    - ECS allows you to run and maintain a specified number of instances of a task definition simultaneously in a cluster.

    - In addition to maintaining the desired count of tasks in your service, you can optionally run your service behind a load balancer.
    - There are two deployment strategies in ECS:
        - **Rolling Update**
            - This involves the service scheduler replacing the current running version of the container with the latest version.
            - The number of tasks ECS adds or removes from the service during a rolling update is controlled by the deployment configuration, which consists of the minimum and maximum number of tasks allowed during a service deployment.
        - **Blue/Green Deployment with AWS CodeDeploy**
            - This deployment type allows you to verify a new deployment of a service before sending production traffic to it.
            - The service must be configured to use either an Application Load Balancer or Network Load Balancer.
- Container Agent
    - The **container agent** runs on each infrastructure resource within an ECS cluster.
    - It sends information about the resource's current running tasks and resource utilization to ECS, and starts and stops tasks whenever it receives a request from ECS.
    - Container agent is only supported on Amazon EC2 instances.
- You can attach multiple target groups to your Amazon ECS services that are running on either Amazon EC2 or AWS Fargate. This allows you to maintain a single ECS service that can serve traffic from both internal and external load balancers and support multiple path based routing rules and applications that need to expose more than one port.
- The Classic Load Balancer doesn't allow you to run multiple copies of a task on the same instance. You must statically map port numbers on a container instance. However, an Application Load Balancer uses **dynamic port mapping**, so you can run multiple tasks from a single service on the same container instance.
- If a service's task fails the load balancer health check criteria, the task is stopped and restarted. This process continues until your service reaches the number of desired running tasks.
- Services with tasks that use the *awsvpc* network mode, such as those with the Fargate launch type, do not support Classic Load Balancers. You must use NLB instead for TCP.

AWS Fargate

- You can use Fargate with ECS to run containers without having to manage servers or clusters of EC2 instances.
- You no longer have to provision, configure, or scale clusters of virtual machines to run containers.
- Fargate only supports container images hosted on Elastic Container Registry (ECR) or Docker Hub.

Task Definitions for Fargate Launch Type

- Fargate task definitions require that the network mode is set to *awsvpc*. The *awsvpc* network mode provides each task with its own elastic network interface.
- Fargate task definitions require that you specify CPU and memory at the task level.
- Fargate task definitions only support the *awslogs* log driver for the log configuration. This configures your Fargate tasks to send log information to Amazon CloudWatch Logs.
- Task storage is **ephemeral**. After a Fargate task stops, the storage is deleted.
- Amazon ECS tasks running on both Amazon EC2 and AWS Fargate can mount Amazon Elastic File System (EFS) file systems.
- Put multiple containers in the same task definition if:
    - Containers share a common lifecycle.
    - Containers are required to be run on the same underlying host.
    - You want your containers to share resources.
    - Your containers share data volumes.
- Otherwise, define your containers in separate tasks definitions so that you can scale, provision, and deprovision them separately.

Task Definitions for EC2 Launch Type

- Create task definitions that group the containers that are used for a common purpose, and separate the different components into multiple task definitions.
- After you have your task definitions, you can create services from them to maintain the availability of your desired tasks.
- For EC2 tasks, the following are the types of data volumes that can be used:
    - **Docker volumes**
    - **Bind mounts**
- Private repositories are only supported by the EC2 Launch Type.

Monitoring

- You can configure your container instances to send log information to CloudWatch Logs. This enables you to view different logs from your container instances in one convenient location.
- With CloudWatch Alarms, watch a single metric over a time period that you specify, and perform one or more actions based on the value of the metric relative to a given threshold over a number of time periods.
- Share log files between accounts, monitor CloudTrail log files in real time by sending them to CloudWatch Logs.

Tagging

- ECS resources, including task definitions, clusters, tasks, services, and container instances, are assigned an Amazon Resource Name (ARN) and a unique resource identifier (ID). These resources can be tagged with values that you define, to help you organize and identify them.

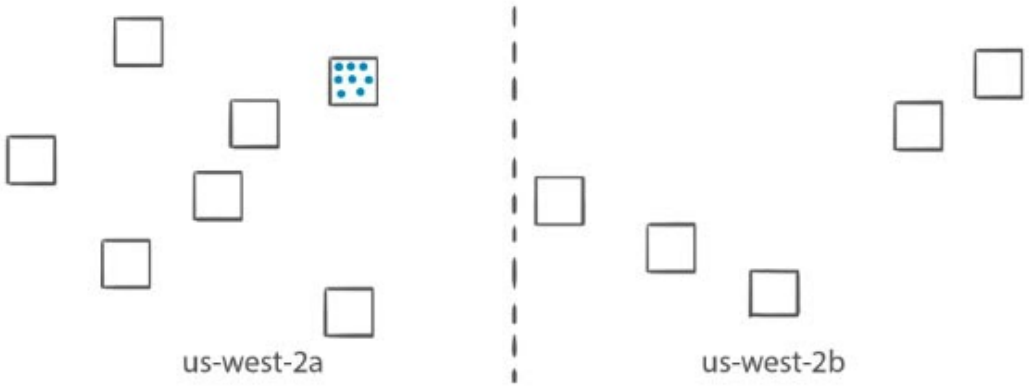
Pricing

- With Fargate, you pay for the amount of vCPU and memory resources that your containerized application requests. vCPU and memory resources are calculated from the time your container images are pulled until the Amazon ECS Task terminates.
- There is no additional charge for EC2 launch type. You pay for AWS resources (e.g. EC2 instances or EBS volumes) you create to store and run your application.

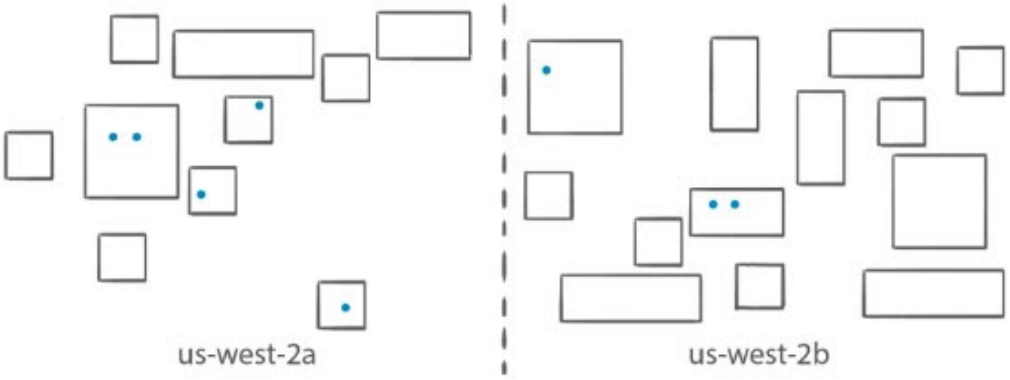
Sources:
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html
https://aws.amazon.com/ecs/features/
https://aws.amazon.com/ecs/pricing/
https://aws.amazon.com/ecs/faqs/

ECS Task Placement Strategies

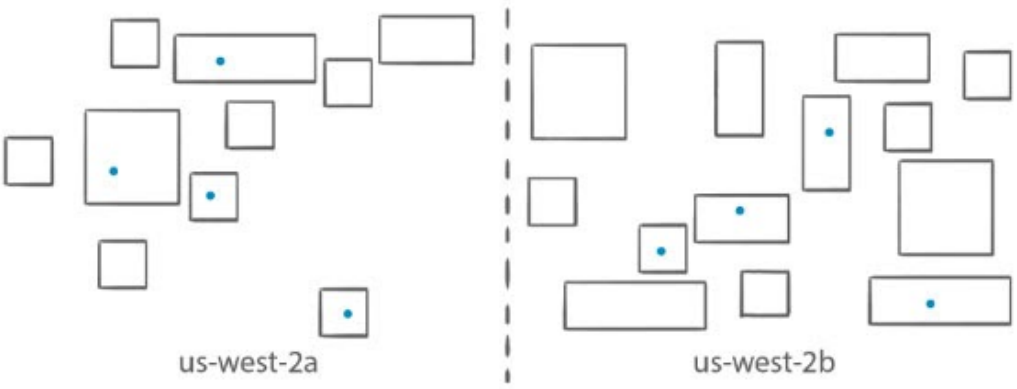
- A *task placement strategy* is an algorithm for selecting instances for task placement or tasks for termination. When a task that uses the EC2 launch type is launched, Amazon ECS must determine where to place the task based on the requirements specified in the task definition, such as CPU and memory. Similarly, when you scale down the task count, Amazon ECS must determine which tasks to terminate.
- A *task placement constraint* is a rule that is considered during task placement.
    - You can use constraints to place tasks based on Availability Zone or instance type.
    - You can also associate attributes, which are name/value pairs, with your container instances and then use a constraint to place tasks based on attribute.
- Task placement strategy types:
    - **Binpack** - Place tasks based on the least available amount of CPU or memory. This minimizes the number of instances in use and allow you to be cost-efficient. For example, you have running tasks in c5.2xlarge instances that are known to be CPU intensive but are not memory consuming. You can maximize your instances' memory allocation by launching tasks in them instead of spawning a new instance.



    - **Random** - Place tasks randomly. You use this strategy when task placement or termination does not matter.



- **Spread** - Place tasks evenly based on the specified value. Accepted values are attribute key-value pairs, instanceId, or host. Spread is typically used to achieve high availability by making sure that multiple copies of a task are scheduled across multiple instances. **Spread across Availability Zones** is the default placement strategy used for services.



- You can combine different strategy types to suit your application needs.
- Task placement strategies are a best effort.
- By default, Fargate tasks are spread across Availability Zones.
- By default, ECS uses the following placement strategies:
  - When you run tasks with the RunTask API action, tasks are placed randomly in a cluster.
  - When you launch and terminate tasks with the CreateService API action, the service scheduler spreads the tasks across the Availability Zones (and the instances within the zones) in a cluster.
- You can update your placement strategies and constraints without having to recreate a service with the desired changes.

Sources:
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task-placement.html
https://aws.amazon.com/blogs/compute/amazon-ecs-task-placement/

AWS Serverless Application Model (SAM)

- An open-source framework for building serverless applications.
- It provides shorthand syntax to express functions, APIs, databases, and event source mappings.
- You create a **JSON** or **YAML** configuration template to model your applications.
- During deployment, SAM transforms and expands the SAM syntax into **AWS CloudFormation syntax**. Any resource that you can declare in an AWS CloudFormation template you can also declare in an AWS SAM template.
- The **SAM CLI** provides a Lambda-like execution environment that lets you locally build, test, and debug applications defined by SAM templates. You can also use the SAM CLI to deploy your applications to AWS.
- You can use AWS SAM to build serverless applications that use **any runtime supported by AWS Lambda**. You can also use SAM CLI to locally debug Lambda functions written in Node.js, Java, Python, and Go.
- Template Anatomy
    - If you are writing an AWS Serverless Application Model template alone and not via CloudFormation, the **Transform** section is required.
    - The **Globals** section is unique to AWS SAM templates. It defines properties that are common to all your serverless functions and APIs. All the **AWS::Serverless::Function**, **AWS::Serverless::Api**, and **AWS::Serverless::SimpleTable** resources inherit the properties that are defined in the **Globals** section.
    - The **Resources** section can contain a combination of AWS CloudFormation resources and AWS SAM resources.
- Overview of Syntax
    - AWS::Serverless::Api
        - This resource type describes an API Gateway resource. It's useful for advanced use cases where you want full control and flexibility when you configure your APIs.
    - AWS::Serverless::Application
        - This resource type embeds a serverless application from the AWS Serverless Application Repository or from an Amazon S3 bucket as a nested application. Nested applications are deployed as nested stacks, which can contain multiple other resources.
    - AWS::Serverless::Function
        - This resource type describes configuration information for creating a Lambda function. You can describe any event source that you want to attach to the Lambda function—such as Amazon S3, Amazon DynamoDB Streams, and Amazon Kinesis Data Streams.
    - AWS::Serverless::LayerVersion
        - This resource type creates a Lambda layer version that contains library or runtime code needed by a Lambda function. When a serverless layer version is transformed, AWS SAM also transforms the logical ID of the resource so that old layer versions are not automatically deleted by AWS CloudFormation when the resource is updated.

    - AWS::Serverless::SimpleTable
        - This resource type provides simple syntax for describing how to create DynamoDB tables.
- Commonly used SAM CLI commands
    - The sam init command generates pre-configured AWS SAM templates.
    - The sam local command supports local invocation and testing of your Lambda functions and SAM-based serverless applications by executing your function code locally in a Lambda-like execution environment.
    - The sam package and sam deploy commands let you bundle your application code and dependencies into a “deployment package" and then deploy your serverless application to the AWS Cloud.
    - The sam logs command enables you to fetch, tail, and filter logs for Lambda functions.
    - The output of the sam publish command includes a link to the AWS Serverless Application Repository directly to your application.
    - Use sam validate to validate your SAM template.
- Controlling access to APIs
    - You can use AWS SAM to control who can access your API Gateway APIs by enabling authorization within your AWS SAM template.
        - A **Lambda authorizer** (formerly known as a custom authorizer) is a Lambda function that you provide to control access to your API. When your API is called, this Lambda function is invoked with a request context or an authorization token that is provided by the client application. The Lambda function returns a *policy document* that specifies the operations that the caller is authorized to perform, if any. There are two types of Lambda authorizers:
            - Token based type receives the caller's identity in a bearer token, such as a JSON Web Token (JWT) or an OAuth token.
            - Request parameter based type receives the caller's identity in a combination of headers, query string parameters, stageVariables, and $context variables.
        - **Amazon Cognito user pools** are user directories in Amazon Cognito. A client of your API must first sign a user in to the user pool and obtain an identity or access token for the user. Then your API is called with one of the returned tokens. The API call succeeds only if the required token is valid.
- The optional **Transform** section of a CloudFormation template specifies one or more macros that AWS CloudFormation uses to process your template. Aside from macros you create, AWS CloudFormation also supports the **AWS::Serverless** transform which is a macro hosted on AWS CloudFormation.
    - The AWS::Serverless transform specifies the version of the AWS Serverless Application Model (AWS SAM) to use. This model defines the AWS SAM syntax that you can use and how AWS CloudFormation processes it.

Sources:
https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/what-is-sam.html

https://aws.amazon.com/serverless/sam/
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/transform-section-structure.html

STORAGE

Amazon S3

- S3 stores data as objects within **buckets**.
- An **object** consists of a file and optionally any metadata that describes that file.
- A **key** is the unique identifier for an object within a bucket.
- Storage capacity is virtually unlimited.

Buckets

- For each bucket, you can:
  - Control access to it (create, delete, and list objects in the bucket)
  - View access logs for it and its objects
  - Choose the geographical region where to store the bucket and its contents.
- **Bucket name** must be a unique DNS-compliant name.
  - The name must be unique across all existing bucket names in Amazon S3.
  - After you create the bucket you cannot change the name.
  - The bucket name is visible in the URL that points to the objects that you're going to put in your bucket.
- By default, you can create up to 100 buckets in each of your AWS accounts.
- You can't change its Region after creation.
- You can host static websites by configuring your bucket for website hosting.
- You can't delete an S3 bucket using the Amazon S3 console if the bucket contains 100,000 or more objects. You can't delete an S3 bucket using the AWS CLI if versioning is enabled.

Data Consistency Model

- read-after-write consistency for PUTS of new objects in your S3 bucket in all regions
- eventual consistency for read-after-write HEAD or GET requests
- eventual consistency for overwrite PUTS and DELETES in all regions
- strong read-after-write consistency for any storage request

Storage Classes

- Storage Classes for Frequently Accessed Objects
  - S3 **STANDARD** for **general-purpose** storage of frequently accessed data.
- Storage Classes for Infrequently Accessed Objects
  - S3 **STANDARD_IA** for long-lived, but **less frequently accessed** data. It stores the object data redundantly across multiple geographically separated AZs.

    - S3 **ONEZONE_IA** stores the object data in only one AZ. Less expensive than STANDARD_IA, but data is not resilient to the physical loss of the AZ.
    - These two storage classes are suitable for objects larger than 128 KB that you plan to store for **at least 30 days**. If an object is less than 128 KB, Amazon S3 charges you for 128 KB. If you delete an object before the 30-day minimum, you are charged for 30 days.
- Amazon S3 Intelligent Tiering
    - S3 Intelligent-Tiering is a storage class designed for customers who want to optimize storage costs automatically when data access patterns change, without performance impact or operational overhead.
    - S3 Intelligent-Tiering is the first cloud object storage class that delivers automatic cost savings by moving data between two access tiers — frequent access and infrequent access — when access patterns change, and is ideal for data with unknown or changing access patterns.
    - S3 Intelligent-Tiering monitors access patterns and moves objects that have not been accessed for 30 consecutive days to the infrequent access tier. If an object in the infrequent access tier is accessed later, it is automatically moved back to the frequent access tier.
    - S3 Intelligent-Tiering supports the archive access tier. If the objects haven't been accessed for 90 consecutive days, it will be moved to the archive access tier. After 180 consecutive days of no access, it is automatically moved to the deep archive access tier.
    - There are no retrieval fees in S3 Intelligent-Tiering.
- GLACIER
    - For long-term **archive**
    - Archived objects are not available for real-time access. You must first restore the objects before you can access them.
    - You cannot specify GLACIER as the storage class at the time that you create an object.
    - Glacier objects are visible through S3 only.
    - Retrieval Options
        - **Expedited** - allows you to quickly access your data when occasional urgent requests for a subset of archives are required. For all but the largest archived objects, data accessed are typically made available within 1–5 minutes. There are two types of Expedited retrievals: On-Demand requests are similar to EC2 On-Demand instances and are available most of the time. Provisioned requests are guaranteed to be available when you need them.
        - **Standard** - allows you to access any of your archived objects within several hours. Standard retrievals typically complete within 3–5 hours. This is the default option for retrieval requests that do not specify the retrieval option.
        - **Bulk** - Glacier's lowest-cost retrieval option, enabling you to retrieve large amounts, even petabytes, of data inexpensively in a day. Bulk retrievals typically complete within 5–12 hours.
    - For S3 Standard, S3 Standard-IA, and Glacier storage classes, your objects are automatically stored across multiple devices spanning a **minimum of three** Availability Zones.

	S3 Standard	S3 Intelligent-Tiering *	S3 Standard-IA	S3 One Zone-IA **	S3 Glacier	S3 Glacier Deep Archive
Designed for durability	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)
Designed for availability	99.99%	99.9%	99.9%	99.5%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99.9%	99.9%
Availability Zones	≥3	≥3	≥3	1	≥3	≥3
Minimum capacity charge per object	N/A	N/A	128KB	128KB	40KB	40KB
Minimum storage duration charge	N/A	30 days	30 days	30 days	90 days	180 days
Retrieval fee	N/A	N/A	per GB retrieved	per GB retrieved	per GB retrieved	per GB retrieved
First byte latency	milliseconds	milliseconds	milliseconds	milliseconds	select minutes or hours	select hours
Storage type	Object	Object	Object	Object	Object	Object
Lifecycle transitions	Yes	Yes	Yes	Yes	Yes	Yes

* S3 Intelligent-Tiering charges a small tiering fee and has a minimum eligible object size of 128KB for auto-tiering. Smaller objects may be stored but will always be charged at the Frequent Access tier rates

** Because S3 One Zone-IA stores data in a single AWS Availability Zone, data stored in this storage class will be lost in the event of Availability Zone destruction.

Tutorials Dojo

- Amazon S3 Glacier Deep Archive
    - A new Amazon S3 storage class providing secure and durable object storage for long-term retention of data that is accessed rarely in a year.
    - S3 Glacier Deep Archive offers the lowest cost storage in the cloud, at prices lower than storing and maintaining data in on-premises magnetic tape libraries or archiving data offsite.
    - All objects stored in the S3 Glacier Deep Archive storage class are replicated and stored across at least three geographically-dispersed Availability Zones, protected by 99.999999999% durability, and can be restored within 12 hours or less.
    - S3 Glacier Deep Archive also offers a bulk retrieval option, where you can retrieve petabytes of data within 48 hours.

S3 API

- REST - use standard HTTP requests to create, fetch, and delete buckets and objects. You can use S3 virtual hosting to address a bucket in a REST API call by using the HTTP *Host* header.

- SOAP - support for SOAP over HTTP is deprecated, but it is still available over HTTPS. However, new Amazon S3 features will not be supported for SOAP. AWS recommends using either the REST API or the AWS SDKs.

Bucket Configurations

Subresource	Description
location	Specify the AWS Region where you want S3 to create the bucket.
policy and *ACL* (access control list)	All your resources are private by default. Use bucket policy and ACL options to grant and manage bucket-level permissions.
cors (cross-origin resource sharing)	You can configure your bucket to allow cross-origin requests. CORS defines a way for client web applications that are loaded in one domain to interact with resources in a different domain.
website	You can configure your bucket for static website hosting.
logging	Logging enables you to track requests for access to your bucket. Each access log record provides details about a single access request, such as the requester, bucket name, request time, request action, response status, and error code, if any.
event notification	You can enable your bucket to send you notifications of specified bucket events.
versioning	AWS recommends VERSIONING AS A BEST PRACTICE to recover objects from being deleted or overwritten by mistake.
lifecycle	You can define lifecycle rules for objects in your bucket that have a well-defined lifecycle.
cross-region replication	Cross-region replication is the automatic, asynchronous copying of objects across buckets in different AWS Regions.
tagging	S3 provides the *tagging* subresource to store and manage tags on a bucket. AWS generates a cost allocation report with usage and costs aggregated by your tags.

requestPayment	By default, the AWS account that creates the bucket (the bucket owner) pays for downloads from the bucket. The bucket owner can specify that the person requesting the download will be charged for the download.
transfer acceleration	Transfer Acceleration enables fast, easy, and secure transfers of files over long distances between your client and an S3 bucket. It takes advantage of Amazon CloudFront's globally distributed **edge locations**.

Objects

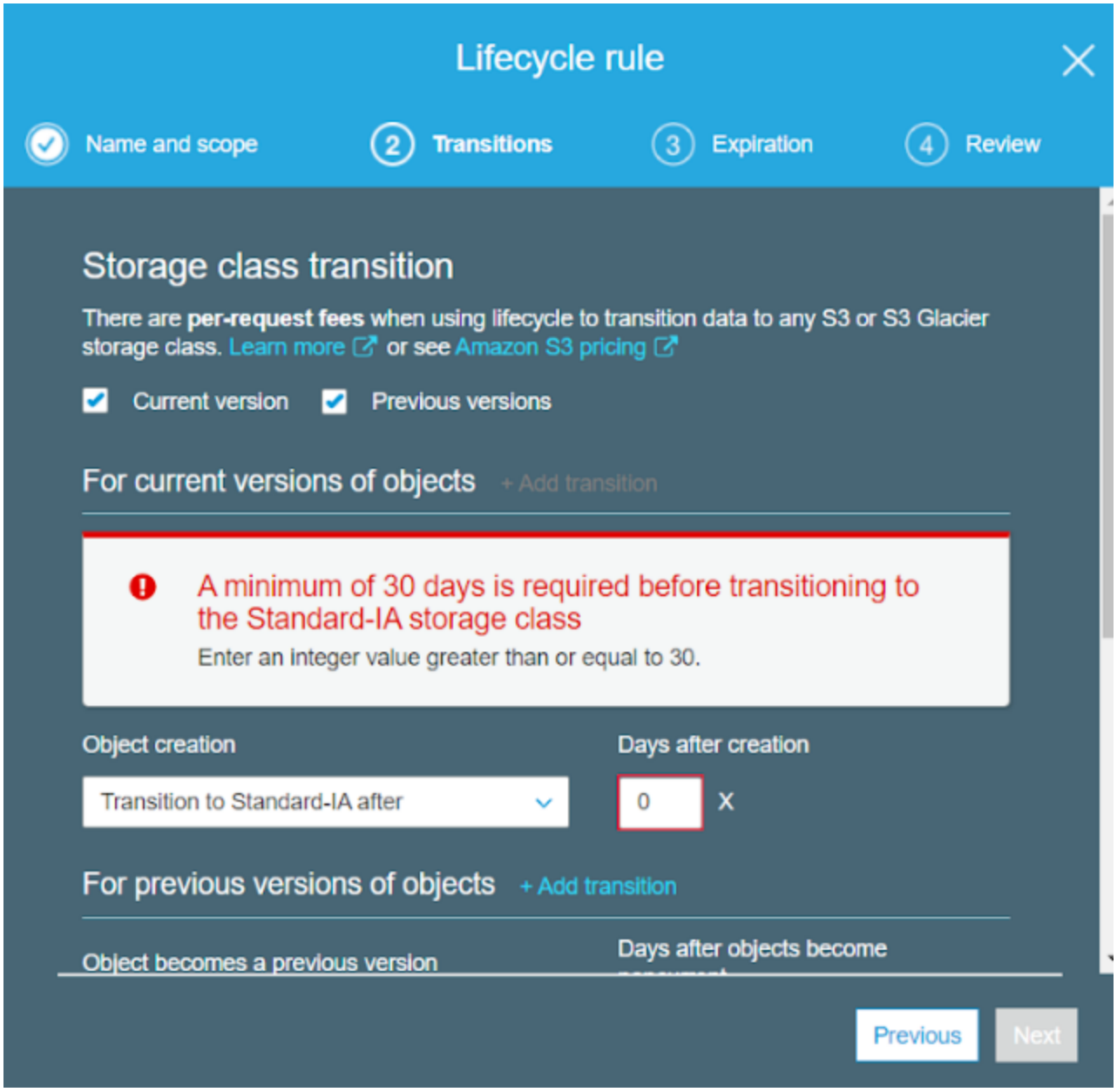
- Are private by default. Grant permissions to other users.
- Each S3 object has **data**, a **key**, and **metadata**.
- You cannot modify object metadata after object is uploaded.
- Two kinds of metadata
    - System metadata

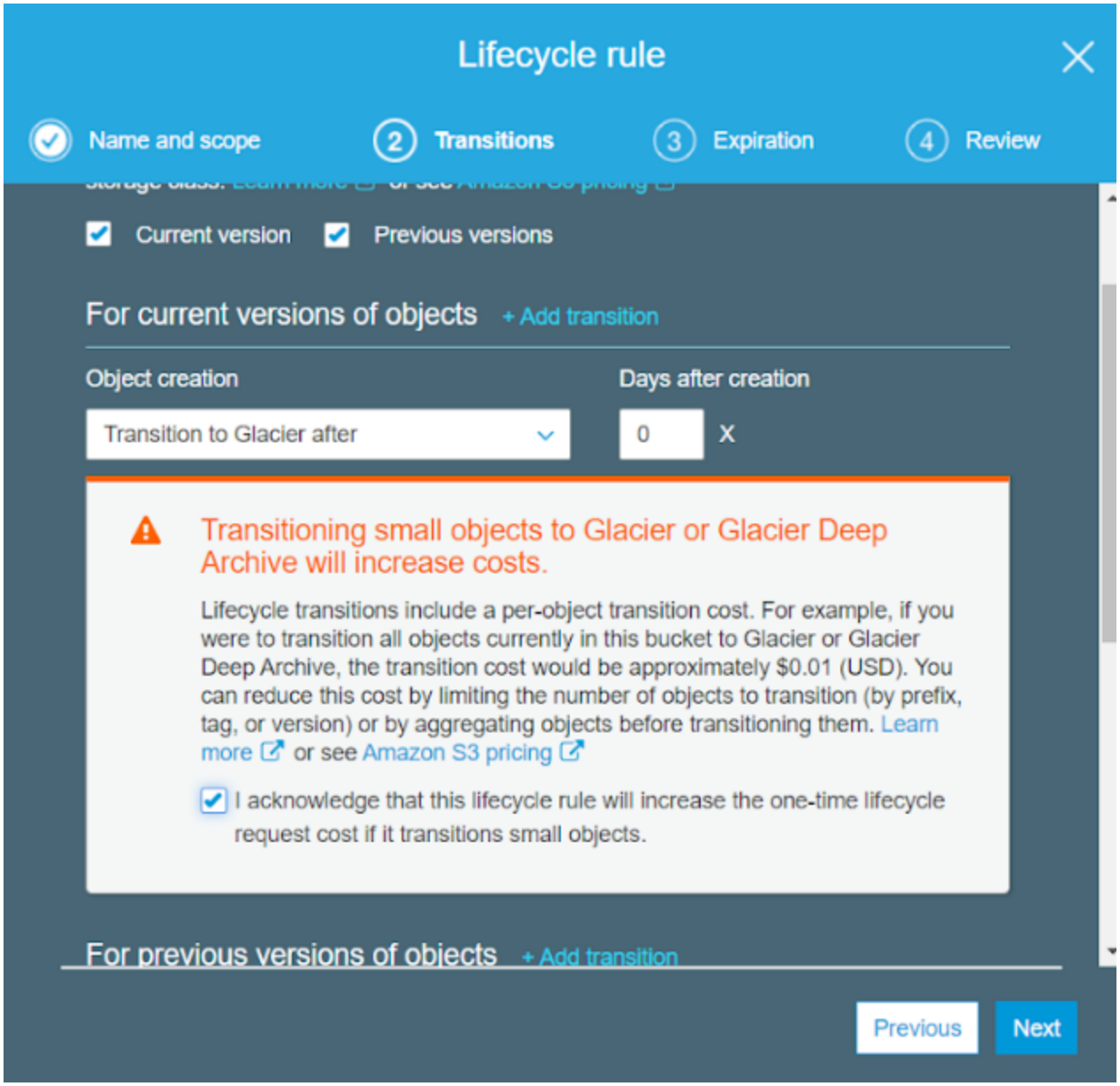
Name	Description	Can User Modify the Value?
Date	Current date and time.	No
Content-Length	Object size in bytes.	No
Last-Modified	Object creation date or the last modified date, whichever is the latest.	No
Content-MD5	The base64-encoded 128-bit MD5 digest of the object.	No
x-amz-server-side-encryption	Indicates whether server-side encryption is enabled for the object, and whether that encryption is from the AWS Key Management Service (SSE-KMS) or from AWS managed encryption (SSE-S3).	Yes
x-amz-version-id	Object version. When you enable versioning on a bucket, Amazon S3 assigns a version number to objects added to the bucket.	No

x-amz-delete-marker	In a bucket that has versioning enabled, this Boolean marker indicates whether the object is a delete marker.	No
x-amz-storage-class	Storage class used for storing the object.	Yes
x-amz-website-redirect-location	Redirects requests for the associated object to another object in the same bucket or an external URL.	Yes
x-amz-server-side-encryption-aws-kms-key-id	If x-amz-server-side-encryption is present and has the value of aws:kms, this indicates the ID of the AWS Key Management Service (AWS KMS) master encryption key that was used for the object.	Yes
x-amz-server-side-encryption-customer-algorithm	Indicates whether server-side encryption with customer-provided encryption keys (SSE-C) is enabled.	Yes

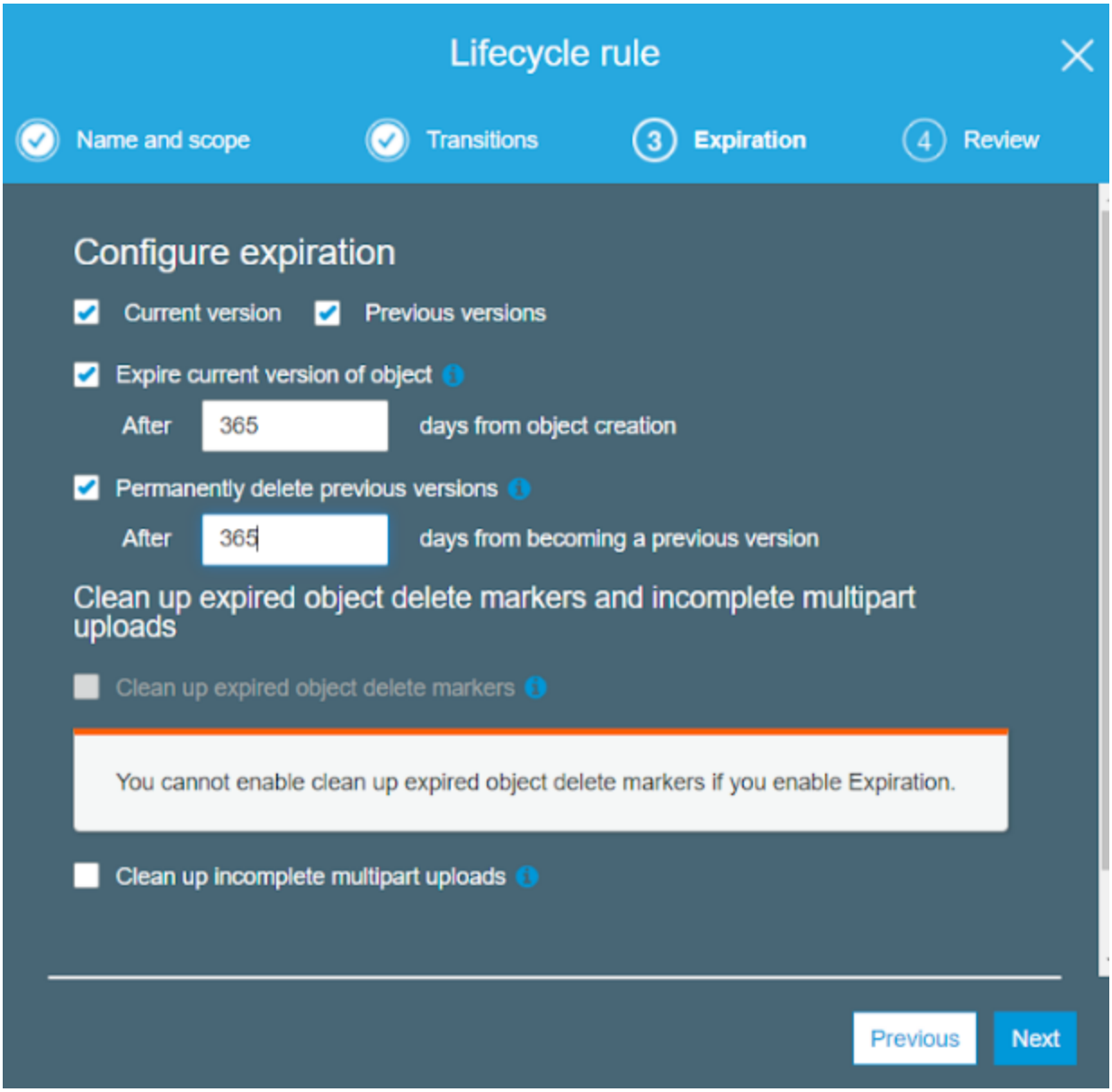
    - User-defined metadata - key-value pair that you provide.
- You can upload and copy objects of up to **5 GB** in size in a single operation. For objects greater than 5 GB up to **5 TB**, you must use the **multipart upload API**.
- Tagging
    - You can associate up to 10 tags with an object. Tags associated with an object must have unique tag keys.
    - A tag key can be up to 128 Unicode characters in length and tag values can be up to 256 Unicode characters in length.
    - Key and values are case sensitive.
- Object Delete
    - Deleting Objects from a Version-Enabled Bucket
        - Specify a non-versioned delete request - specify only the object's key, and not the version ID.
        - Specify a versioned delete request - specify both the key and also a version ID.
    - Deleting Objects from an MFA-Enabled Bucket
        - If you provide an invalid MFA token, the request always fails.
        - If you are not deleting a versioned object, and you don't provide an MFA token, the delete succeeds.
- Object Lock
    - Prevents objects from being deleted or overwritten for a fixed amount of time or indefinitely.
    - Objection retention options:
        - Retention period - object remains locked until the retention period expires.

        - Legal hold - object remains locked until you explicitly remove it.
    - Object Lock works only in versioned buckets and only applies to individual versions of objects.
- Object Ownership
    - With the *bucket-owner-full-control* ACL, you can automatically assume ownership of objects that are uploaded to your buckets.
- S3 Select
    - S3 Select is an Amazon S3 capability designed to pull out only the data you need from an object, which can dramatically improve the performance and reduce the cost of applications that need to access data in S3.
    - Amazon S3 Select works on objects stored in CSV and JSON format, Apache Parquet format, JSON Arrays, and BZIP2 compression for CSV and JSON objects.
    - CloudWatch Metrics for S3 Select lets you monitor S3 Select usage for your applications. These metrics are available at 1-minute intervals and lets you quickly identify and act on operational issues.
- Lifecycle Management
    - A *lifecycle configuration* is a set of rules that define actions that is applied to a group of objects.
        - Transition actions—Define when objects transition to another storage class. For S3-IA and S3-One-Zone, the objects must be stored at least 30 days in the current storage class before you can transition them to another class.


Lifecycle rule
Name and scope
2 Transitions
3 Expiration
4 Review
Storage class transition
There are per-request fees when using lifecycle to transition data to any S3 or S3 Glacier storage class. Learn more or see Amazon S3 pricing
Current version
Previous versions
For current versions of objects
+ Add transition
A minimum of 30 days is required before transitioning to the Standard-IA storage class
Enter an integer value greater than or equal to 30.
Object creation
Days after creation
Transition to Standard-IA after
0
X
For previous versions of objects
+ Add transition
Object becomes a previous version
Days after objects become
Previous
Next



- Expiration actions—Define when objects expire. S3 deletes expired objects on your behalf.



Pricing

- S3 charges you only for what you actually use, with no hidden fees and no overage charges
- No charge for creating a bucket, but only for storing objects in the bucket and for transferring objects in and out of the bucket.

Charge	Comments
Storage	You pay for storing objects in your S3 buckets. The rate you're charged depends on your objects' size, how long you stored the objects during the month, and the storage class.
Requests	You pay for requests, for example, GET requests, made against your S3 buckets and objects. This includes lifecycle requests. The rates for requests depend on what kind of request you're making.
Retrievals	You pay for retrieving objects that are stored in STANDARD_IA, ONEZONE_IA, and GLACIER storage.
Early Deletes	If you delete an object stored in STANDARD_IA, ONEZONE_IA, or GLACIER storage before the minimum storage commitment has passed, you pay an early deletion fee for that object.
Storage Management	You pay for the storage management features that are enabled on your account's buckets.
Bandwidth	You pay for all bandwidth into and out of S3, except for the following: • Data transferred in from the internet • Data transferred out to an Amazon EC2 instance, when the instance is in the same AWS Region as the S3 bucket • Data transferred out to Amazon CloudFront You also pay a fee for any data transferred using Amazon S3 Transfer Acceleration.

Networking

- Hosted-style access
  - Amazon S3 routes any virtual hosted–style requests to the US East (N. Virginia) region by default if you use the endpoint s3.amazonaws.com, instead of the region-specific endpoint.
  - Format:
    - http://bucket.s3.amazonaws.com
    - http://bucket.s3-aws-region.amazonaws.com
- Path-style access
  - In a path-style URL, the endpoint you use must match the Region in which the bucket resides.
  - Format:
    - US East (N. Virginia) Region endpoint, http://s3.amazonaws.com/bucket
    - Region-specific endpoint, http://s3-aws-region.amazonaws.com/bucket
- Customize S3 URLs with CNAMEs - the bucket name must be the same as the CNAME.

- **Amazon S3 Transfer Acceleration** enables fast, easy, and secure transfers of files over long distances between your client and an S3 bucket. It takes advantage of Amazon CloudFront's globally distributed **edge locations**.
- Transfer Acceleration cannot be disabled, and can only be suspended.
- Transfer Acceleration URL is: *bucket*.s3-accelerate.amazonaws.com

Security

- Policies contain the following:
    - **Resources** – buckets and objects
    - **Actions** – set of operations
    - **Effect** – can be either allow or deny. Need to explicitly grant allow to a resource.
    - **Principal** – the account, service or user who is allowed access to the actions and resources in the statement.
- Resource Based Policies
    - Bucket Policies
        - Provides **centralized access control** to buckets and objects based on a variety of conditions, including S3 operations, requesters, resources, and aspects of the request (e.g., IP address).
        - Can either **add or deny permissions** across all (or a subset) of objects within a bucket.
        - IAM users need additional permissions from root account to perform bucket operations.
        - Bucket policies are limited to 20 KB in size.
    - Access Control Lists
        - A list of grants identifying grantee and permission granted.
        - ACLs use an S3–specific XML schema.
        - You can grant permissions only to other AWS accounts, not to users in your account.
        - You cannot grant conditional permissions, nor explicitly deny permissions.
        - Object ACLs are limited to 100 granted permissions per ACL.
        - The only recommended use case for the bucket ACL is to grant **write** permissions to the **S3 Log Delivery group**.
- User Policies
    - AWS IAM (see AWS Security and Identity Services)
        - IAM User Access Keys
        - Temporary Security Credentials
- Versioning
    - Use versioning to keep multiple versions of an object in one bucket.
    - Versioning protects you from the consequences of unintended overwrites and deletions.
    - You can also use versioning to archive objects so you have access to previous versions.
    - Since versioning is disabled by default, need to EXPLICITLY enable.
    - When you PUT an object in a versioning-enabled bucket, the non-current version is not overwritten.

    - When you DELETE an object, all versions remain in the bucket and Amazon S3 inserts a delete marker.
    - Performing a simple GET Object request when the current version is a delete marker returns a 404 Not Found error. You can, however, GET a non-current version of an object by specifying its version ID.
    - You can permanently delete an object by specifying the version you want to delete. Only the owner of an Amazon S3 bucket can permanently delete a version.
- Encryption
    - Server-side Encryption using
        - **Amazon S3-Managed Keys (SSE-S3)**
        - **AWS KMS-Managed Keys (SSE-KMS)**
        - **Customer-Provided Keys (SSE-C)**
    - Client-side Encryption using
        - AWS KMS-managed customer master key
        - client-side master key
- MFA Delete
    - MFA delete grants additional authentication for either of the following operations:
        - Change the versioning state of your bucket
        - Permanently delete an object version
    - MFA Delete requires two forms of authentication together:
        - Your security credentials
        - The concatenation of a valid serial number, a space, and the six-digit code displayed on an approved authentication device
- Cross-Account Access
    - You can provide another AWS account access to an object that is stored in an Amazon Simple Storage Service (Amazon S3) bucket. These are the methods on how to grant cross-account access to objects that are stored in your own Amazon S3 bucket:
        - Resource-based policies and AWS Identity and Access Management (IAM) policies for programmatic-only access to S3 bucket objects
        - Resource-based Access Control List (ACL) and IAM policies for programmatic-only access to S3 bucket objects
        - Cross-account IAM roles for programmatic and console access to S3 bucket objects
- Requester Pays Buckets
    - Bucket owners pay for all of the Amazon S3 storage and data transfer costs associated with their bucket. To save on costs, you can enable the Requester Pays feature so the requester will pay the cost of the request and the data download from the bucket instead of the bucket owner. Take note that the bucket owner always pays the cost of storing data.
- Monitoring
    - Automated monitoring tools to watch S3:

        - Amazon CloudWatch Alarms – Watch a single metric over a time period that you specify, and perform one or more actions based on the value of the metric relative to a given threshold over a number of time periods.
        - AWS CloudTrail Log Monitoring – Share log files between accounts, monitor CloudTrail log files in real time by sending them to CloudWatch Logs, write log processing applications in Java, and validate that your log files have not changed after delivery by CloudTrail.
    - Monitoring with CloudWatch
        - Daily Storage Metrics for Buckets - You can monitor bucket storage using CloudWatch, which collects and processes storage data from S3 into readable, daily metrics.
        - Request metrics - You can choose to monitor S3 requests to quickly identify and act on operational issues. The metrics are available at 1 minute intervals after some latency to process.
    - You can have a maximum of 1000 metrics configurations per bucket.
    - Supported event activities that occur in S3 are recorded in a CloudTrail event along with other AWS service events in **Event history**.
- Website Hosting
    - Enable website hosting in your bucket **Properties**.
    - Your static website is available via the region-specific website endpoint.
    - You must make the objects that you want to serve publicly readable by writing a bucket policy that grants everyone *s3:GetObject* permission.

Key Difference	REST API Endpoint	Website Endpoint
Access control	Supports both public and private content.	Supports only publicly readable content.
Error message handling	Returns an XML-formatted error response.	Returns an HTML document.
Redirection support	Not applicable	Supports both object-level and bucket-level redirects.
Requests supported	Supports all bucket and object operations	Supports only GET and HEAD requests on objects.
Responses to GET and HEAD requests at the root of a bucket	Returns a list of the object keys in the bucket.	Returns the index document that is specified in the website configuration.
Secure Sockets Layer (SSL) support	Supports SSL connections.	Does not support SSL connections.

- S3 Events Notification
  - To enable notifications, add a *notification configuration* identifying the events to be published, and the destinations where to send the event notifications.
  - Can publish following events:
    - A new object created event
    - An object removal event
    - A Reduced Redundancy Storage (RRS) object lost event
  - Supports the following destinations for your events:
    - Amazon Simple Notification Service (Amazon SNS) topic
    - Amazon Simple Queue Service (Amazon SQS) queue
    - AWS Lambda
- Cross Region Replication
  - Enables automatic, asynchronous copying of objects across buckets in different AWS Regions.
  - When to use:
    - Comply with compliance requirements
    - Minimize latency
    - Increase operational efficiency
    - Maintain object copies under different ownership
  - Requirements of CRR:
    - Both source and destination buckets must have versioning enabled.
    - The source and destination buckets must be in different AWS Regions.
    - S3 must have permissions to replicate objects from the source bucket to the destination bucket on your behalf.
    - If the owner of the source bucket doesn't own the object in the bucket, the object owner must grant the bucket owner READ and READ_ACP permissions with the object ACL.
  - Only the following are replicated:
    - Objects created after you add a replication configuration.
    - Both unencrypted objects and objects encrypted using Amazon S3 managed keys (SSE-S3) or AWS KMS managed keys (SSE-KMS), although you must explicitly enable the option to replicate objects encrypted using KMS keys. The replicated copy of the object is encrypted using the same type of server-side encryption that was used for the source object.
    - Object metadata.
    - Only objects in the source bucket for which the bucket owner has permissions to read objects and access control lists.
    - Object ACL updates, unless you direct S3 to change the replica ownership when source and destination buckets aren't owned by the same accounts.
    - Object tags.
  - What isn’t replicated

        - Objects that existed before you added the replication configuration to the bucket.
        - Objects created with server-side encryption using customer-provided (SSE-C) encryption keys.
        - Objects created with server-side encryption using AWS KMS–managed encryption (SSE-KMS) keys.
        - Objects in the source bucket that the bucket owner doesn't have permissions for.
        - Updates to bucket-level subresources.
        - Actions performed by lifecycle configuration.
        - Objects in the source bucket that are replicas created by another cross-region replication. You can replicate objects from a source bucket to **only one destination bucket**.
    - CRR delete operations
        - If you make a DELETE request without specifying an object version ID, S3 adds a delete marker.
        - If you specify an object version ID to delete in a DELETE request, S3 deletes that object version in the source bucket, but it doesn't replicate the deletion in the destination bucket. This protects data from malicious deletions.

S3 Batch Operations is a new feature that makes it simple to manage billions of objects stored in Amazon S3. Customers can make changes to object properties and metadata, and perform other storage management tasks – such as copying objects between buckets, replacing tag sets, modifying access controls, and restoring archived objects from Amazon S3 Glacier – for any number of S3 objects in minutes.

Sources:
https://docs.aws.amazon.com/AmazonS3/latest/dev/Welcome.html
https://aws.amazon.com/s3/faqs/

Amazon S3 Glacier

- **Long-term archival** solution optimized for infrequently used data, or "cold data."
- Glacier is a REST-based web service.
- You can store an unlimited number of archives and an unlimited amount of data.
- You cannot specify Glacier as the storage class at the time you create an object.
- It is designed to provide an average annual durability of 99.999999999% for an archive. Glacier synchronously stores your data across multiple AZs before confirming a successful upload.
- To prevent corruption of data packets over the wire, Glacier uploads the checksum of the data during data upload. It compares the received checksum with the checksum of the received data and validates data authenticity with checksums during data retrieval.
- Glacier works together with **Amazon S3 lifecycle rules** to help you automate archiving of S3 data and reduce your overall storage costs. Requested archival data is copied to S3 One Zone-IA

Data Model

- **Vault**
  - A container for storing archives.
  - Each vault resource has a unique address with form: https://*region-specific endpoint*/*account-id*/vaults/*vaultname*
  - You can store an unlimited number of archives in a vault.
  - Vault operations are Region specific.
- **Archive**
  - Can be any data such as a photo, video, or document and is a base unit of storage in Glacier.
  - Each archive has a unique address with form: https://*region-specific-endpoint*/*account-id*/vaults/*vault-name*/archives/*archive-id*
- **Job**
  - You can perform a select query on an archive, retrieve an archive, or get an inventory of a vault. Glacier Select runs the query in place and writes the output results to Amazon S3.
  - Select, archive retrieval, and vault inventory jobs are associated with a vault. A vault can have multiple jobs in progress at any point in time.
- **Notification Configuration**
  - Because jobs take time to complete, Glacier supports a notification mechanism to notify you when a job is complete.

Glacier Operations

- Retrieving an archive (asynchronous operation)
- Retrieving a vault inventory (list of archives) (asynchronous operation)
- Create and delete vaults
- Get the vault description for a specific vault or for all vaults in a region
- Set, retrieve, and delete a notification configuration on the vault

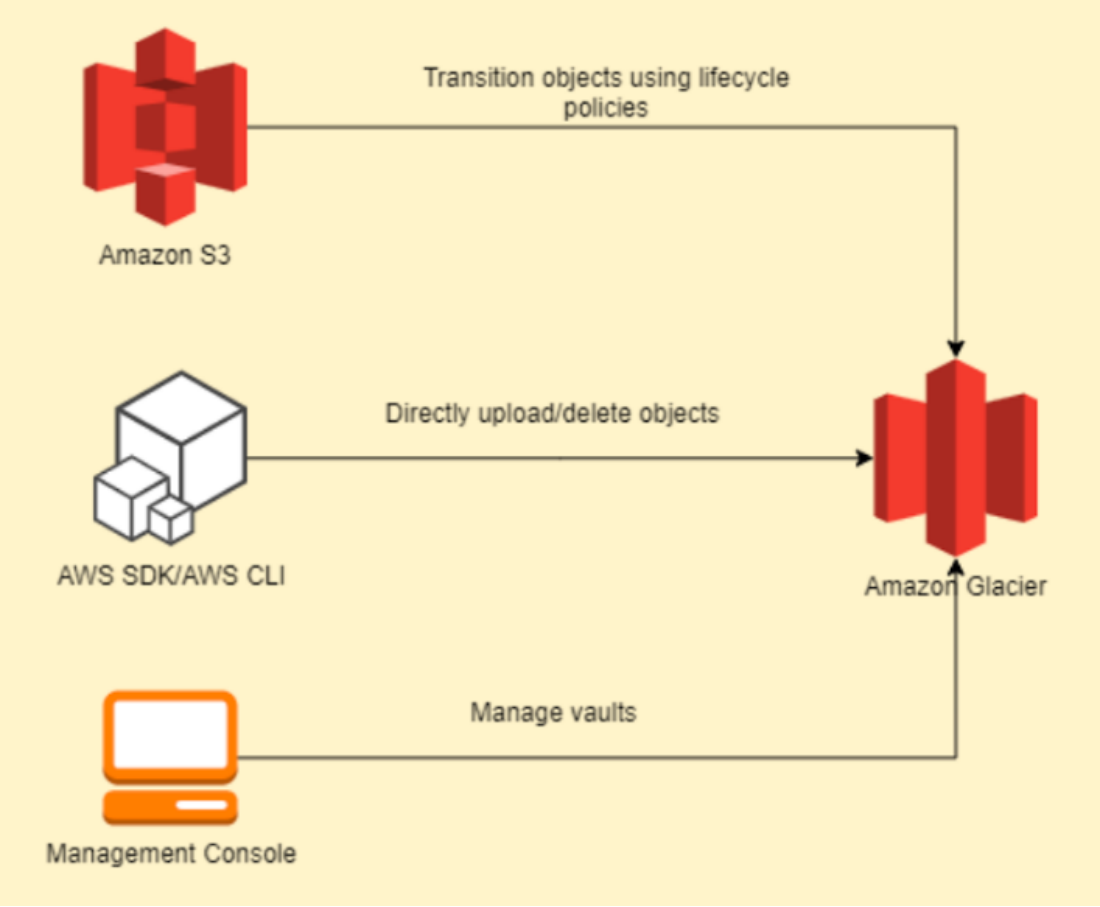
- Upload and delete archives. You cannot update an existing archive.
- Glacier jobs — select, archive-retrieval, inventory-retrieval

Vaults

- Vault operations are region specific.
- Vault names must be unique within an account and the region in which the vault is being created.
- You can delete a vault only if there are no archives in the vault as of the last inventory that Glacier computed and there have been no writes to the vault since the last inventory.
- You can retrieve vault information such as the vault creation date, number of archives in the vault, and the total size of all the archives in the vault.
- Glacier maintains an inventory of all archives in each of your vaults for disaster recovery or occasional reconciliation. A **vault inventory** refers to the list of archives in a vault. Glacier updates the vault inventory approximately once a day. Downloading a vault inventory is an asynchronous operation.
- You can assign your own metadata to Glacier vaults in the form of **tags**. A tag is a key-value pair that you define for a vault.
- **Glacier Vault Lock** allows you to easily deploy and enforce compliance controls for individual Glacier vaults with a vault lock policy. You can specify controls such as "**write once read many**" (WORM) in a vault lock policy and lock the policy from future edits. Once locked, the policy can no longer be changed.

Archives

- Glacier supports the following basic archive operations: upload, download, and delete. Downloading an archive is an asynchronous operation.
- You can upload an archive in a single operation or upload it in parts.
- Using the multipart upload API, you can upload large archives, up to about 10,000 x 4 GB.
- You cannot upload archives to Glacier by using the management console. Use the AWS CLI or write code to make requests, by using either the REST API directly or by using the AWS SDKs.
- You cannot delete an archive using the Amazon S3 Glacier (Glacier) management console. Glacier provides an API call that you can use to delete one archive at a time.



- After you upload an archive, you cannot update its content or its description. The only way you can update the archive content or its description is by deleting the archive and uploading another archive.
- Glacier does not support any additional metadata for the archives.

Glacier Select

- You can perform filtering operations using simple SQL statements directly on your data in Glacier.
- You can run queries and custom analytics on your data that is stored in Glacier, without having to restore your data to a hotter tier like S3.
- When you perform select queries, Glacier provides three data access tiers:
  - **Expedited** - data accessed is typically made available within 1–5 minutes.
  - **Standard** - data accessed is typically made available within 3–5 hours.
  - **Bulk** - data accessed is typically made available within 5–12 hours.

Glacier Data Retrieval Policies

- Set data retrieval limits and manage the data retrieval activities across your AWS account in each region.

- Three types of policies:
    - Free Tier Only - you can keep your retrievals within your daily free tier allowance and not incur any data retrieval cost.
    - Max Retrieval Rate - ensures that the peak retrieval rate from all retrieval jobs across your account in a region does not exceed the bytes-per-hour limit you set.
    - No Retrieval Limit

Security

- Glacier encrypts your data at rest by default and supports secure data transit with SSL.
- Data stored in Amazon Glacier is immutable, meaning that after an archive is created it cannot be updated.
- Access to Glacier requires credentials that AWS can use to authenticate your requests. Those credentials must have permissions to access Glacier vaults or S3 buckets.
- Glacier requires all requests to be signed for authentication protection. To sign a request, you calculate a digital signature using a cryptographic hash function that returns a hash value that you include in the request as your signature.
- Glacier supports policies only at the vault level.
- You can attach identity-based policies to IAM identities.
- A Glacier vault is the primary resource and resource-based policies are referred to as *vault policies*.
- When activity occurs in Glacier, that activity is recorded in a CloudTrail event along with other AWS service events in *Event History*.

Pricing

- You are charged per GB per month of storage
- You are charged for retrieval operations such as retrieve requests and amount of data retrieved depending on the data access tier - Expedited, Standard, or Bulk
- Upload requests are charged.
- You are charged for data transferred out of Glacier.
- Pricing for Glacier Select is based upon the total amount of data scanned, the amount of data returned, and the number of requests initiated.
- There is a charge if you delete data within 90 days.

Sources:
https://docs.aws.amazon.com/amazonglacier/latest/dev/
https://aws.amazon.com/glacier/features/?nc=sn&loc=2
https://aws.amazon.com/glacier/pricing/?nc=sn&loc=3
https://aws.amazon.com/glacier/faqs/?nc=sn&loc=6

Amazon EBS

- **Block level storage** volumes for use with EC2 instances.
- Well-suited for use as the primary storage for file systems, databases, or for any applications that require fine granular updates and access to raw, unformatted, block-level storage.
- Well-suited to both database-style applications (random reads and writes), and to throughput-intensive applications (long, continuous reads and writes).
- New EBS volumes receive their maximum performance the moment that they are available and do not require initialization (formerly known as pre-warming). However, storage blocks on volumes that were restored from snapshots must be initialized (pulled down from Amazon S3 and written to the volume) before you can access the block.
- Termination protection is turned off by default and must be manually enabled (keeps the volume/data when the instance is terminated)
- You can have up to 5,000 EBS volumes by default
- You can have up to 10,000 snapshots by default

Features

- Different types of storage options: **General Purpose SSD (gp2,gp3)**, **Provisioned IOPS SSD (io1,io2)**, **Throughput Optimized HDD (st1)**, and **Cold HDD (sc1)** volumes up to **16 TiB** in size.
- You can mount multiple volumes on the same instance, and you can mount a Provisioned IOPS volume to multiple instances at a time using Amazon EBS Multi-Attach.
- Enable Multi-Attach on EBS Provisioned IOPS io1 volumes to allow a single volume to be concurrently attached to up to sixteen AWS Nitro System-based Amazon EC2 instances within the same AZ.
- You can create a file system on top of these volumes, or use them in any other way you would use a block device (like a hard drive).
- You can use encrypted EBS volumes to meet data-at-rest encryption requirements for regulated/audited data and applications.
- You can create point-in-time **snapshots** of EBS volumes, which are persisted to Amazon S3. Similar to AMIs. Snapshots can be copied across AWS regions.
- Volumes are created in a specific AZ, and can then be attached to any instances in that same AZ. To make a volume available outside of the AZ, you can create a snapshot and restore that snapshot to a new volume anywhere in that region.
- You can copy snapshots to other regions and then restore them to new volumes there, making it easier to leverage multiple AWS regions for geographical expansion, data center migration, and disaster recovery.
- Performance metrics, such as bandwidth, throughput, latency, and average queue length, provided by Amazon CloudWatch, allow you to monitor the performance of your volumes to make sure that you are providing enough performance for your applications without paying for resources you don't need.

- You can detach an EBS volume from an instance explicitly or by terminating the instance. However, if the instance is running, you must first unmount the volume from the instance.
- If an EBS volume is the root device of an instance, you must stop the instance before you can detach the volume.
- You can use AWS Backup, an automated and centralized backup service, to protect EBS volumes and your other AWS resources. AWS Backup is integrated with Amazon DynamoDB, Amazon EBS, Amazon RDS, Amazon EFS, and AWS Storage Gateway to give you a fully managed AWS backup solution.
- With AWS Backup, you can configure backups for EBS volumes, automate backup scheduling, set retention policies, and monitor backup and restore activity.
- EBS fast snapshot restore allows you to create a volume from a snapshot that is fully initialized. This removes the latency of I/O operations on the block when accessed for the first time.

Types of EBS Volumes

- General Purpose SSD (gp2)
  - Base performance of 3 IOPS/GiB, with the ability to burst to 3,000 IOPS for extended periods of time.
  - Support up to 16,000 IOPS and 250 MB/s of throughput.
  - The **burst duration** of a volume is dependent on the size of the volume, the burst IOPS required, and the credit balance when the burst begins. Burst IO duration is computed using the following formula:

    Burst duration = (Credit balance) [(Burst IOPS) - 3 (Volume size in GiB)]

  - If your *gp2* volume uses all of its I/O credit balance, the maximum IOPS performance of the volume remains at the baseline IOPS performance level and the volume's maximum throughput is reduced to the baseline IOPS multiplied by the maximum I/O size.
  - Throughput for a gp2 volume can be calculated using the following formula, up to the throughput limit of 160 MiB/s:

    Throughput in MiB/s = (Volume size in GiB) (IOPS per GiB) × (I/O size in KiB)

- Provisioned IOPS SSD (io1)
  - Designed for I/O-intensive workloads, particularly database workloads, which are sensitive to storage performance and consistency.
  - Allows you to specify a consistent IOPS rate when you create the volume
- Throughput Optimized HDD (st1)
  - Low-cost magnetic storage that focuses on throughput rather than IOPS.
  - Throughput of up to 500 MiB/s.
  - Subject to throughput and throughput-credit caps, the available throughput of an *st1* volume is expressed by the following formula:

(Volume size)(Credit accumulation rate per TiB) = Throughput

- Cold HDD (sc1)
  - Low-cost magnetic storage that focuses on throughput rather than IOPS.
  - Throughput of up to 250 MiB/s.

Volume Name	General Purpose SSD		Provisioned IOPS SSD	
Volume type	gp3	gp2	io2	io1
Description	General Purpose SSD volume that balances price performance for a wide variety of transactional workloads	General Purpose SSD volume that balances price performance for a wide variety of transactional workloads	High performance SSD volume designed for **business-critical** latency-sensitive applications	High performance SSD volume designed for latency-sensitive transactional workloads
Use Cases	virtual desktops, medium sized single instance databases such as MSFT SQL Server and Oracle DB, low-latency interactive apps, dev & test, boot volumes	Boot volumes, low-latency interactive apps, dev & test	Workloads that require sub-millisecond latency, and sustained IOPS performance or more than 64,000 IOPS or 1,000 MiB/s of throughput	Workloads that require sustained IOPS performance or more than 16,000 IOPS and I/O-intensive database workloads
Volume Size	1 GB – 16 TB	1 GB – 16 TB	4 GB – 16 TB	4 GB – 16 TB
Durability	99.8% - 99.9% durability	99.8% - 99.9% durability	99.999%	99.8% - 99.9%
Max IOPS / Volume	16,000	16,000	64,000	64,000
Max Throughput / Volume	1000 MB/s	250 MB/s	1,000 MB/s	1,000 MB/s
Max IOPS / Instance	260,000	260,000	160,000	260,000
Max IOPS / GB	N/A	N/A	500 IOPS/GB	50 IOPS/GB
Max Throughput / Instance	7,500 MB/s	7,500 MB/s	4,750 MB/s	7,500 MB/s
Latency	single digit millisecond	single digit millisecond	single digit millisecond	single digit millisecond
Multi-Attach	No	No	Yes	Yes

Volume Name	Throughput Optimized HDD	Cold HDD
Volume type	st1	sc1
Description	Low cost HDD volume designed for frequently accessed, throughput-intensive workloads	Throughput-oriented storage for data that is infrequently accessed Scenarios where the lowest storage cost is important
Use Cases	Big data, data warehouses, log processing	Colder data requiring fewer scans per day
Volume Size	125 GB – 16 TB	125 GB – 16 TB
Durability	99.8% - 99.9% durability	99.8% - 99.9% durability
Max IOPS / Volume	500	250
Max Throughput / Volume	500 MB/s	250 MB/s
Max IOPS / Instance	260,000	260,000
Max IOPS / GB	N/A	N/A
Max Throughput / Instance	7,500 MB/s	7,500 MB/s
Multi-Attach	No	No

FEATURES	SSD Solid State Drive	HDD Hard Disk Drive
Best for workloads with:	***small, random*** I/O operations	***large, sequential*** I/O operations
Can be used as a bootable volume?	Yes	No
Suitable Use Cases	- Best for **transactional workloads** - Critical business applications that require sustained IOPS performance - Large database workloads such as MongoDB, Oracle, Microsoft SQL Server and many others…	- Best for ***large streaming workloads*** requiring consistent, fast throughput at a low price - Big data, Data warehouses, Log processing - Throughput-oriented storage for large volumes of data that is ***infrequently*** accessed
Cost	moderate / high	low
Dominant Performance Attribute	IOPS	Throughput (`MiB/s`)

Encryption

- Data stored at rest on an encrypted volume, disk I/O, and snapshots created from it are all encrypted.
- Also provides encryption for data in-transit from EC2 to EBS since encryption occurs on the servers that hosts EC2 instances.
- The following types of data are encrypted:
  - Data at rest inside the volume
  - All data moving between the volume and the instance
  - All snapshots created from the volume
  - All volumes created from those snapshots

- Uses AWS Key Management Service (AWS KMS) master keys when creating encrypted volumes and any snapshots created from your encrypted volumes.
- Volumes restored from encrypted snapshots are automatically encrypted.
- EBS encryption is only available on certain instance types.
- There is no direct way to encrypt an existing unencrypted volume, or to remove encryption from an encrypted volume. However, you can migrate data between encrypted and unencrypted volumes.
- You can now enable Amazon Elastic Block Store (EBS) Encryption by Default, ensuring that all new EBS volumes created in your account are encrypted.

Monitoring

- Cloudwatch Monitoring two types: Basic and Detailed monitoring
- Volume status checks provide you the information that you need to determine whether your EBS volumes are impaired, and help you control how a potentially inconsistent volume is handled. List of statuses include:
    - Ok - normal volume
    - Warning - degraded volume
    - Impaired - stalled volume
    - Insufficient-data - insufficient data
- Volume events include a start time that indicates the time at which an event occurred, and a duration that indicates how long I/O for the volume was disabled. The end time is added to the event when I/O for the volume is enabled.
- Volume events are:
    - Awaiting Action: Enable IO
    - IO Enabled
    - IO Auto-Enabled
    - Normal
    - Degraded
    - Severely Degraded
    - Stalled

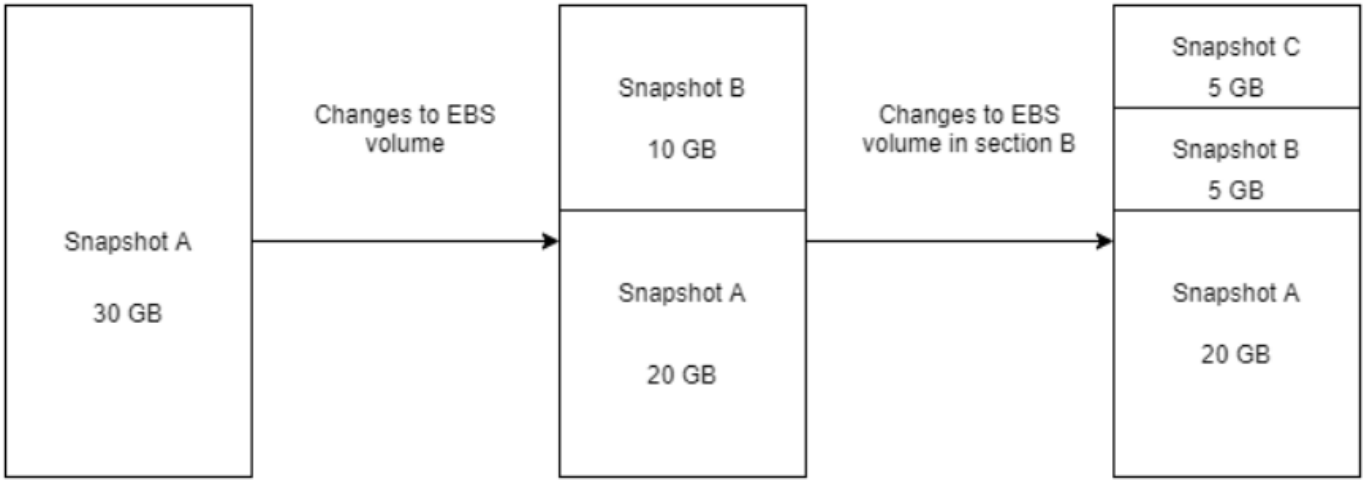
Modifying the Size, IOPS, or Type of an EBS Volume on Linux

- If your current-generation EBS volume is attached to a current-generation EC2 instance type, you can increase its size, change its volume type, or (for an io1 volume) adjust its IOPS performance, all without detaching it.
- EBS currently supports a maximum volume size of 16 TiB.
- Two partitioning schemes commonly used on Linux and Windows systems: master boot record (MBR) and GUID partition table (GPT).
- An EBS volume being modified goes through a sequence of states. The volume enters first the **Modifying** state, then the **Optimizing** state, and finally the **Complete** state.

- You can expand a partition to a new size. Expand by using *parted* or *gdisk*.
- Use a file system–specific command to resize the file system to the larger size of the new volume. These commands work even if the volume to extend is the root volume. For ext2, ext3, and ext4 file systems, this command is *resize2fs*. For XFS file systems, this command is *xfs_growfs*.
- Decreasing the size of an EBS volume is not supported.

EBS Snapshots

- Back up the data on your EBS volumes to S3 by taking point-in-time snapshots.
- Snapshots are **incremental** backups, which means that only the blocks on the device that have changed after your most recent snapshot are saved. This minimizes the time required to create the snapshot and saves on storage costs by not duplicating data.



- When you delete a snapshot, only the data unique to that snapshot is removed.
- You can share a snapshot across AWS accounts by modifying its access permissions.
- You can make copies of your own snapshots as well as snapshots that have been shared with you.
- A snapshot is constrained to the Region where it was created.
- EBS snapshots broadly support EBS encryption.
- You can't delete a snapshot of the root device of an EBS volume used by a registered AMI. You must first deregister the AMI before you can delete the snapshot.
- Each account can have up to **5 concurrent snapshot copy requests** to a single destination Region.
- User-defined tags are not copied from the source snapshot to the new snapshot.
- Snapshots are constrained to the Region in which they were created. To share a snapshot with another Region, copy the snapshot to that Region.
- Snapshots that you intend to share must instead be encrypted with a custom CMK.

Amazon EBS–Optimized Instances

- Provides the best performance for your EBS volumes by minimizing contention between EBS I/O and other traffic from your instance.
- EBS–optimized instances deliver dedicated bandwidth between 500 Mbps and 60,000 Mbps to EBS.
- For instance types that are EBS–optimized by default, there is no need to enable EBS optimization and no effect if you disable EBS optimization.

Pricing

- You are charged by the amount you provision in GB per month until you release the storage.
- Provisioned storage for *gp2* volumes, provisioned storage and provisioned IOPS for io1 volumes, provisioned storage for st1 and sc1 volumes will be billed in per-second increments, with a 60 second minimum.
- With Provisioned IOPS SSD (io1) volumes, you are also charged by the amount you provision in IOPS per month.
- After you detach a volume, you are still charged for volume storage as long as the storage amount exceeds the limit of the AWS Free Tier. You must delete a volume to avoid incurring further charges.
- Snapshot storage is based on the amount of space your data consumes in Amazon S3.
- Copying a snapshot to a new Region does incur new storage costs.
- When you enable EBS optimization for an instance that is not EBS-optimized by default, you pay an additional low, hourly fee for the dedicated capacity.

Improving Performance

- Use EBS-Optimized Instances
- Understand How Performance is Calculated
- Understand Your Workload
- Be Aware of the Performance Penalty When Initializing Volumes from Snapshots
- Factors That Can Degrade HDD Performance
- Increase Read-Ahead for High-Throughput, Read-Heavy Workloads on st1 and sc1
- Use a Modern Linux Kernel
- Use RAID 0 (Redundant Array of Independent Disks) to Maximize Utilization of Instance Resources
- Track Performance Using Amazon CloudWatch

Sources:
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AmazonEBS.html
https://aws.amazon.com/ebs/faqs/

Amazon EFS

A fully-managed **file storage service** that makes it easy to set up and scale file storage in the Amazon Cloud.

Features

- The service manages all the file storage infrastructure for you, avoiding the complexity of deploying, patching, and maintaining complex file system configurations.
- EFS supports the Network File System version 4 protocol.
- Multiple Amazon EC2 instances can access an EFS file system at the same time, providing a common data source for workloads and applications running on more than one instance or server.
- EFS file systems store data and metadata across multiple Availability Zones in an AWS Region.
- EFS file systems can grow to petabyte scale, drive high levels of throughput, and allow massively parallel access from EC2 instances to your data.
- EFS provides file system access semantics, such as strong data consistency and file locking.
- EFS enables you to control access to your file systems through Portable Operating System Interface (POSIX) permissions.
- Moving your EFS file data can be managed simply with AWS DataSync - a managed data transfer service that makes it faster and simpler to move data between on-premises storage and Amazon EFS.
- You can schedule automatic incremental backups of your EFS file system using the EFS-to-EFS Backup solution.
- Amazon EFS Infrequent Access (EFS IA) is a new storage class for Amazon EFS that is cost-optimized for files that are accessed less frequently. Customers can use EFS IA by creating a new file system and enabling Lifecycle Management. With Lifecycle Management enabled, EFS automatically will move files that have not been accessed for 30 days from the Standard storage class to the Infrequent Access storage class.

Performance Modes

- General purpose performance mode (default)
    - Ideal for latency-sensitive use cases.
- Max I/O mode
    - Can scale to higher levels of aggregate throughput and operations per second with a tradeoff of slightly higher latencies for file operations.

Throughput Modes

- Bursting Throughput mode (default)
    - Throughput scales as your file system grows.
- Provisioned Throughput mode
    - You specify the throughput of your file system independent of the amount of data stored.

Mount Targets

- To access your EFS file system in a VPC, you create one or more **mount targets** in the VPC. A mount target provides an IP address for an NFSv4 endpoint.
- You can create one mount target in each Availability Zone in a region.
- You mount your file system using its DNS name, which will resolve to the IP address of the EFS mount target. Format of DNS is *File-system-id*.efs.*aws-region*.amazonaws.com
- When using Amazon EFS with an on-premises server, your on-premises server must have a Linux based operating system.

Access Points

- EFS Access Points simplify how applications are provided access to shared data sets in an EFS file system.
- EFS Access Points work together with AWS IAM and enforce an operating system user and group, and a directory for every file system request made through the access point.

Components of a File System

- ID
- creation token
- creation time
- file system size in bytes
- number of mount targets created for the file system
- file system state
- mount target

Data Consistency in EFS

- EFS provides the **open-after-close consistency semantics** that applications expect from NFS.
- Write operations will be durably stored across Availability Zones.
- Applications that perform synchronous data access and perform non-appending writes will have **read-after-write consistency** for data access.

Managing File Systems

- You can create encrypted file systems. EFS supports encryption in transit and encryption at rest.
- Managing file system network accessibility refers to managing the mount targets:
    - Creating and deleting mount targets in a VPC
    - Updating the mount target configuration
- You can create new tags, update values of existing tags, or delete tags associated with a file system.
- The following list explains the metered data size for different types of file system objects.

    - **Regular files** – the metered data size of a regular file is the logical size of the file rounded to the next 4-KiB increment, except that it may be less for sparse files.
        - A sparse file is a file to which data is not written to all positions of the file before its logical size is reached. For a sparse file, if the actual storage used is less than the logical size rounded to the next 4-KiB increment, Amazon EFS reports actual storage used as the metered data size.
    - **Directories** – the metered data size of a directory is the actual storage used for the directory entries and the data structure that holds them, rounded to the next 4 KiB increment. The metered data size doesn't include the actual storage used by the file data.
    - **Symbolic links and special files** – the metered data size for these objects is always 4 KiB.
- **File system deletion is a destructive action that you can't undo.** You lose the file system and any data you have in it, and you can't restore the data. You should always unmount a file system before you delete it.
- You can use AWS DataSync to automatically, efficiently, and securely copy files between two Amazon EFS resources, including file systems in different AWS Regions and ones owned by different AWS accounts. Using DataSync to copy data between EFS file systems, you can perform one-time migrations, periodic ingest for distributed workloads, or automate replication for data protection and recovery.

Mounting File Systems

- To mount your EFS file system on your EC2 instance, use the mount helper in the *amazon-efs-utils package*.
- You can mount your EFS file systems on your on-premises data center servers when connected to your Amazon VPC with AWS Direct Connect or VPN.
- You can use **fstab** to automatically mount your file system using the mount helper whenever the EC2 instance is mounted on reboots.

Lifecycle Management

- You can choose from five EFS Lifecycle Management policies (7, 14, 30, 60, or 90 days) to automatically move files into the EFS Infrequent Access (EFS IA) storage class and save up to 85% in cost.

Monitoring File Systems

- Amazon CloudWatch Alarms
- Amazon CloudWatch Logs
- Amazon CloudWatch Events
- AWS CloudTrail Log Monitoring
- Log files on your file system

Security

- You must have valid credentials to make EFS API requests, such as create a file system.
- You must also have permissions to create or access resources.
- When you first create the file system, there is only one root directory at /. By default, only the root user (UID 0) has read-write-execute permissions.
- Specify EC2 security groups for your EC2 instances and security groups for the EFS mount targets associated with the file system.
- You can use AWS IAM to manage Network File System (NFS) access for Amazon EFS. You can use IAM roles to identify NFS clients with cryptographic security and use IAM policies to manage client-specific permissions.

Pricing

- You pay only for the storage used by your file system.
- Costs related to Provisioned Throughput are determined by the throughput values you specify.

EFS vs EBS vs S3

- Performance Comparison

	Amazon EFS	Amazon EBS Provisioned IOPS
Per-operation latency	Low, consistent latency.	Lowest, consistent latency.
Throughput scale	Multiple GBs per second	Single GB per second

	Amazon EFS	Amazon S3
Per-operation latency	Low, consistent latency.	Low, for mixed request types, and integration with CloudFront.
Throughput scale	Multiple GBs per second	Multiple GBs per second

- Storage Comparison

	Amazon EFS	Amazon EBS Provisioned IOPS

Availability and durability	Data are stored redundantly across multiple AZs.	Data are stored redundantly in a single AZ.
Access	Up to thousands of EC2 instances, from multiple AZs, can connect concurrently to a file system.	A single EC2 instance in a single AZ can connect to a file system.
Use cases	Big data and analytics, media processing workflows, content management, web serving, and home directories.	Boot volumes, transactional and NoSQL databases, data warehousing, and ETL.

	Amazon EFS	**Amazon S3**
Availability and durability	Data are stored redundantly across multiple AZs.	Stored redundantly across multiple AZs.
Access	Up to thousands of EC2 instances from multiple AZs can connect concurrently to a file system.	One to millions of connections over the web.
Use cases	Big data and analytics, media processing workflows, content management, web serving, and home directories.	Web serving and content management, media and entertainment, backups, big data analytics, data lake.

- We have more comparisons for EFS, S3, and EBS in our **Comparison of AWS Services** section.

Sources:
https://docs.aws.amazon.com/efs/latest/ug/
https://aws.amazon.com/efs/pricing/
https://aws.amazon.com/efs/faq/
https://aws.amazon.com/efs/features/
https://aws.amazon.com/efs/when-to-choose-efs/

DATABASE

AWS offers purpose-built databases for all your application needs. Whether you need a Relational, Key-Value, In-memory, or any other type of data store, AWS would most likely have a database service that you can use.

Relational databases store data with predefined schemas and "relationships" between the tables, hence the "Relational" name. It is designed to support ACID (Atomicity, Consistency, Isolation, Durability) transactions with strong data consistency to maintain referential integrity. Key-value databases are suitable for storing and retrieving large volumes of data. It delivers quick response times even in large volumes of concurrent requests.

In-memory databases are primarily used for applications that require real-time access to data. It is capable of delivering data to applications in microseconds and not just in milliseconds since the data are directly stored in memory and not on disk. Aside from this, AWS also offers Document, Time Series, Ledger, and many other database types.

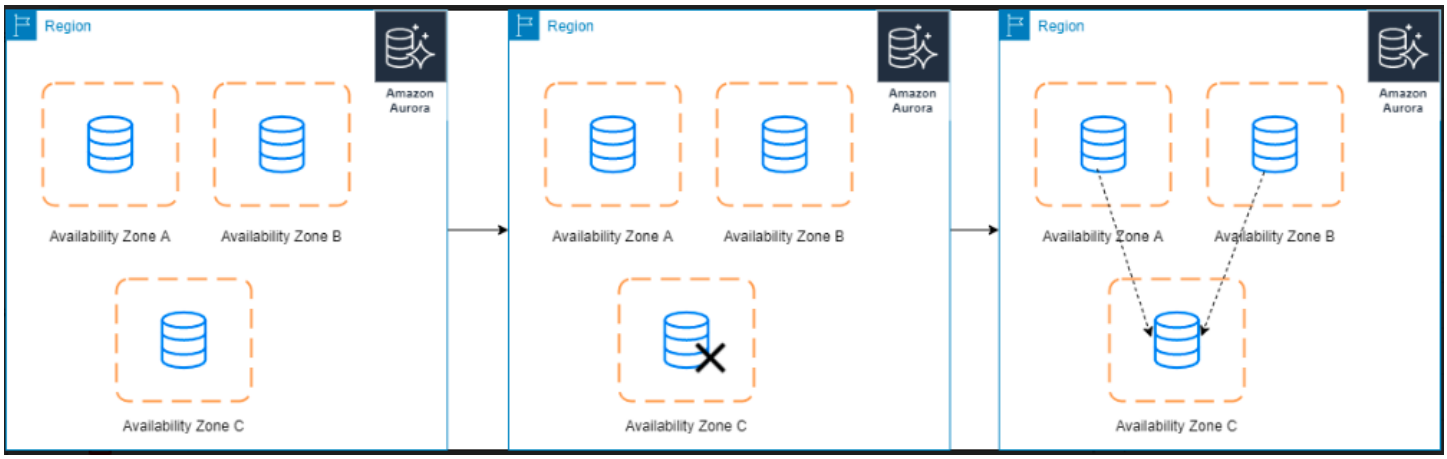
Database Type	Use Cases	AWS Service/s
Relational	Traditional applications, ERP, CRM, e-commerce	Amazon Aurora, Amazon RDS, Amazon Redshift
Key-value	High-traffic web apps, e-commerce systems, gaming applications	Amazon DynamoDB
Document	Content management, catalogs, user profiles	Amazon DocumentDB (with MongoDB compatibility)
In-memory	Caching, session management, gaming leaderboards, geospatial applications	Amazon ElastiCache for Memcached Amazon ElastiCache for Redis
Wide column	High scale industrial apps for equipment maintenance, fleet management, and route optimization	Amazon Keyspaces (for Apache Cassandra)
Graph	Fraud detection, social networking, recommendation engines	Amazon Neptune
Time series	IoT applications, DevOps, industrial telemetry	Amazon Timestream
Ledger	Systems of record, supply chain, registrations, banking transactions	Amazon QLDB

Tutorials Dojo

Amazon Aurora

- A fully managed relational database engine that's compatible with **MySQL** and **PostgreSQL**.
- With some workloads, Aurora can deliver up to five times the throughput of MySQL and up to three times the throughput of PostgreSQL.
- Aurora includes a high-performance storage subsystem. The underlying storage grows automatically as needed, up to 64 terabytes. **The minimum storage is 10GB.**
- Aurora will keep your database up-to-date with the latest patches.
- Aurora supports quick, efficient cloning operations.
    - You can share your Amazon Aurora DB clusters with other AWS accounts for quick and efficient database cloning.
- Aurora is fault-tolerant and self-healing.
- DB Clusters
    - An Aurora **DB cluster** consists of one or more DB instances and a cluster volume that manages the data for those DB instances.
    - An Aurora **cluster volume** is a virtual database storage volume that spans multiple AZs, with each AZ having a copy of the DB cluster data.
    - Cluster Types:
        - **Primary DB instance** – Supports read and write operations, and performs all of the data modifications to the cluster volume. Each Aurora DB cluster has one primary DB instance.
        - **Aurora Replica** – Connects to the same storage volume as the primary DB instance and supports only read operations. Each Aurora DB cluster can have up to 15 Aurora Replicas in addition to the primary DB instance. Aurora automatically fails over to an Aurora Replica in case the primary DB instance becomes unavailable. You can specify the failover priority for Aurora Replicas. Aurora Replicas can also offload read workloads from the primary DB instance.
- Aurora Endpoints
    - **Cluster endpoint** - connects to the current primary DB instance for a DB cluster. This endpoint is the only one that can perform write operations. Each Aurora DB cluster has one cluster endpoint and one primary DB instance.
    - **Reader endpoint** - connects to one of the available Aurora Replicas for that DB cluster. Each Aurora DB cluster has one reader endpoint. The reader endpoint provides load-balancing support for read-only connections to the DB cluster. Use the reader endpoint for read operations, such as queries. You can't use the reader endpoint for write operations.
    - **Custom endpoint** - represents a set of DB instances that you choose. When you connect to the endpoint, Aurora performs load balancing and chooses one of the instances in the group to handle the connection. You define which instances this endpoint refers to, and you decide what purpose the endpoint serves.
    - **Instance endpoint** - connects to a specific DB instance within an Aurora cluster. The instance endpoint provides direct control over connections to the DB cluster. The main way that you use

instance endpoints is to diagnose capacity or performance issues that affect one specific instance in an Aurora cluster.
    - When you connect to an Aurora cluster, the host name and port that you specify point to an intermediate handler called an *endpoint*.
    - Types of Endpoints
- Storage and Reliability
    - Aurora data is stored in the cluster volume, which is designed for reliability. A cluster volume consists of copies of the data across multiple Availability Zones in a single AWS Region.
    - Aurora automatically detects failures in the disk volumes that make up the cluster volume. When a segment of a disk volume fails, Aurora immediately repairs the segment. When Aurora repairs the disk segment, it uses the data in the other volumes that make up the cluster volume to ensure that the data in the repaired segment is current.



    - Aurora preloads the buffer pool with the pages for known common queries that are stored in an in-memory page cache when a database starts up after it has been shut down or restarted after a failure.
    - Aurora is designed to recover from a crash almost instantaneously and continue to serve your application data without the binary log. Aurora performs crash recovery asynchronously on parallel threads, so that your database is open and available immediately after a crash.
    - Amazon Aurora Auto Scaling works with Amazon CloudWatch to automatically add and remove Aurora Replicas in response to changes in performance metrics that you specify. This feature is available in the PostgreSQL-compatible edition of Aurora. There is no additional cost to use Aurora Auto Scaling beyond what you already pay for Aurora and CloudWatch alarms.
    - Dynamic resizing automatically decreases the allocated storage space from your Aurora database cluster when you delete data.
- High Availability and Fault Tolerance
    - When you create Aurora Replicas across Availability Zones, RDS automatically provisions and maintains them synchronously. The primary DB instance is synchronously replicated across

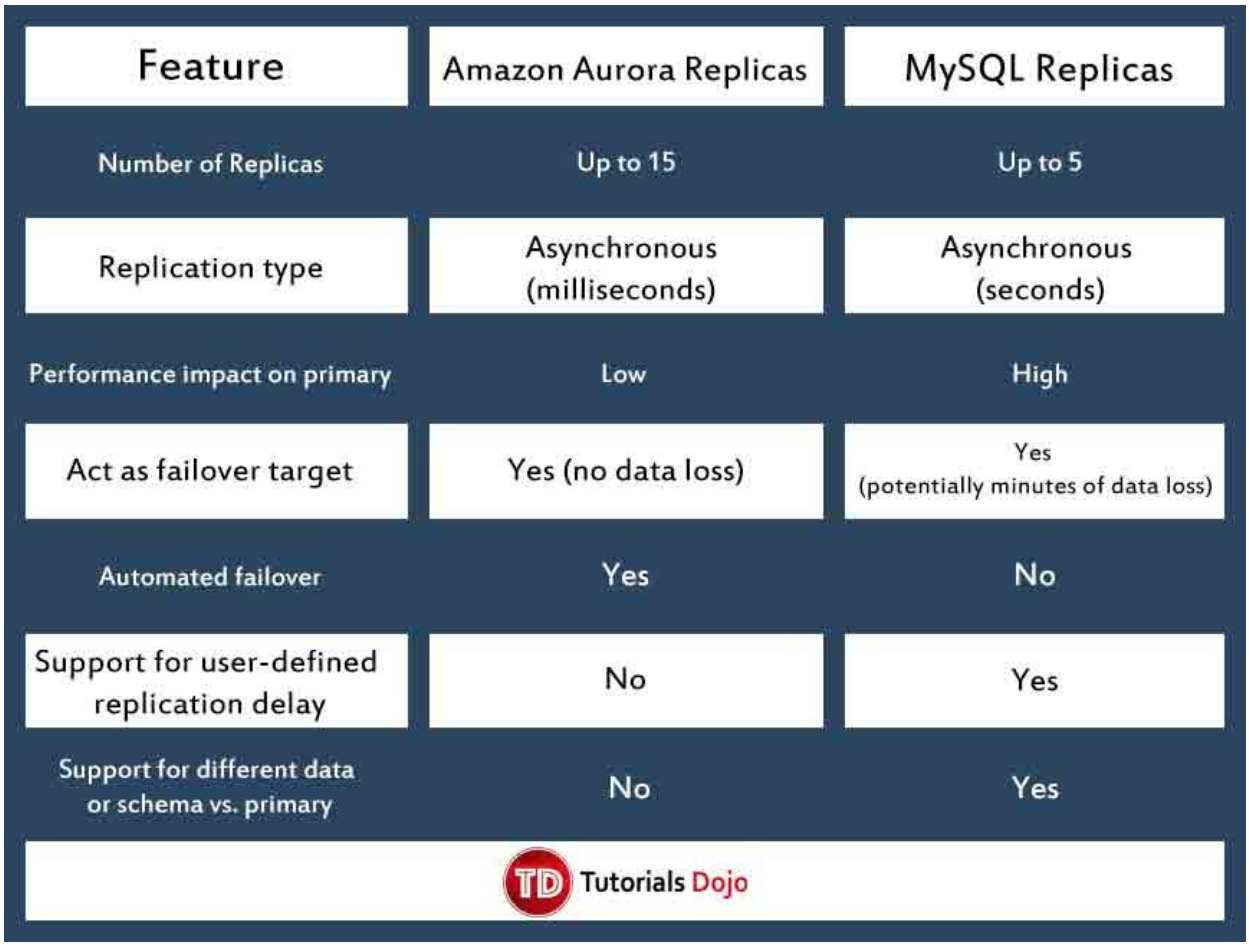
Availability Zones to Aurora Replicas to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.
    - An Aurora DB cluster is fault tolerant by design. If the primary instance in a DB cluster fails, Aurora automatically fails over to a new primary instance in one of two ways:
        - By promoting an existing Aurora Replica to the new primary instance
        - By creating a new primary instance
    - Aurora storage is also self-healing. Data blocks and disks are continuously scanned for errors and repaired automatically.
    - Aurora backs up your cluster volume automatically and retains restore data for the length of the backup retention period, from 1 to 35 days.
    - Aurora automatically maintains **6 copies of your data across 3 Availability Zones** and will automatically attempt to recover your database in a healthy AZ with no data loss.
    - Aurora has a Backtrack feature that rewinds or restores the DB cluster to the time you specify. However, take note that the Amazon Aurora Backtrack feature is not a total replacement for fully backing up your DB cluster since the limit for a backtrack window is only 72 hours.
    - With Aurora MySQL, you can set up cross-region Aurora Replicas using either logical or physical replication. Aurora PostgreSQL does not currently support cross-region replicas.
- Aurora Global Database
    - An Aurora global database spans multiple AWS Regions, enabling low latency global reads and disaster recovery from region-wide outages.
    - Consists of one primary AWS Region where your data is mastered, and one read-only, secondary AWS Region.
    - Aurora global databases use dedicated infrastructure to replicate your data.
    - Aurora global databases introduce a higher level of failover capability than a default Aurora cluster.
    - An Aurora cluster can recover in less than 1 minute even in the event of a complete regional outage. This provides your application with an effective Recovery Point Objective (RPO) of 5 seconds and a Recovery Time Objective (RTO) of less than 1 minute.
- DB Cluster Configurations
    - Aurora supports two types of instance classes
        - **Memory Optimized**
        - **Burstable Performance**
    - **Aurora Serverless** is an on-demand, autoscaling configuration for Amazon Aurora (supports both MySQL and PostgreSQL). An *Aurora Serverless DB cluster* automatically starts up, shuts down, and scales up or down capacity based on your application's needs.
        - A non-Serverless DB cluster for Aurora is called a *provisioned DB cluster*.
        - Instead of provisioning and managing database servers, you specify Aurora Capacity Units (ACUs). Each ACU is a combination of processing and memory capacity.
        - You can choose to pause your Aurora Serverless DB cluster after a given amount of time with no activity. The DB cluster automatically resumes and services the connection requests after receiving requests.

- Aurora Serverless does not support fast failover, but it supports **automatic multi-AZ failover.**
- The cluster volume for an Aurora Serverless cluster is always encrypted. You can choose the encryption key, but not turn off encryption.
- You can set the following specific values:
  - **Minimum Aurora capacity unit** – Aurora Serverless can reduce capacity down to this capacity unit.
  - **Maximum Aurora capacity unit** – Aurora Serverless can increase capacity up to this capacity unit.
  - **Pause after inactivity** – The amount of time with no database traffic to scale to zero processing capacity.
- You pay by the second and only when the database is in use.
- You can share snapshots of Aurora Serverless DB clusters with other AWS accounts or publicly. You also have the ability to copy Aurora Serverless DB cluster snapshots across AWS regions.
  -

- Limitations of Aurora Serverless
  - Aurora Serverless is only available for the following:
    - Aurora with MySQL version 5.6 compatibility
    - Aurora with PostgreSQL version 10.7 compatibility
      -
  - The port number for connections must be:
    - 3306 for Aurora MySQL
    - 5432 for Aurora PostgreSQL
  - You can't give an Aurora Serverless DB cluster a public IP address. You can access an Aurora Serverless DB cluster only from within a virtual private cloud (VPC) based on the Amazon VPC service.
  - Each Aurora Serverless DB cluster requires two AWS PrivateLink endpoints. If you reach the limit for PrivateLink endpoints within your VPC, you can't create any more Aurora Serverless clusters in that VPC.
  - A DB subnet group used by Aurora Serverless can't have more than one subnet in the same Availability Zone.
  - Changes to a subnet group used by an Aurora Serverless DB cluster are not applied to the cluster.
  - Aurora Serverless doesn't support the following features:
    - Loading data from an Amazon S3 bucket
    - Saving data to an Amazon S3 bucket
    - Invoking an AWS Lambda function with an Aurora MySQL native function
    - Aurora Replicas
    - Backtrack
    - Multi-master clusters

        - Database cloning
        - IAM database authentication
        - Restoring a snapshot from a MySQL DB instance
        - Amazon RDS Performance Insights
    - When you reboot the primary instance of an Aurora DB cluster, RDS also automatically restarts all of the Aurora Replicas in that DB cluster. When you reboot the primary instance of an Aurora DB cluster, no failover occurs. When you reboot an Aurora Replica, no failover occurs.
    - **Deletion protection** is enabled by default when you create a production DB cluster using the AWS Management Console. However, deletion protection is disabled by default if you create a cluster using the AWS CLI or API.
        - For Aurora MySQL, you can't delete a DB instance in a DB cluster if both of the following conditions are true:
            - The DB cluster is a Read Replica of another Aurora DB cluster.
            - The DB instance is the only instance in the DB cluster.
- Aurora Multi Master
    - The feature is available on Aurora MySQL 5.6
    - Allows you to create multiple read-write instances of your Aurora database across multiple Availability Zones, which enables uptime-sensitive applications to achieve continuous write availability through instance failure.
    - In the event of instance or Availability Zone failures, Aurora Multi-Master enables the Aurora database to maintain read and write availability with zero application downtime. There is no need for database failovers to resume write operations.
- Tags
    - You can use Amazon RDS tags to add metadata to your RDS resources.
    - Tags can be used with IAM policies to manage access and to control what actions can be applied to the RDS resources.
    - Tags can be used to track costs by grouping expenses for similarly tagged resources.
- Monitoring
    - Subscribe to **Amazon RDS events** to be notified when changes occur with a DB instance, DB cluster, DB cluster snapshot, DB parameter group, or DB security group.
    - Database log files
    - **RDS Enhanced Monitoring** — Look at metrics in real time for the operating system.
    - **RDS Performance Insights** monitors your Amazon RDS DB instance load so that you can analyze and troubleshoot your database performance.
    - Use CloudWatch Metrics, Alarms and Logs
- Security
    - Use IAM to control access.
    - To control which devices and EC2 instances can open connections to the endpoint and port of the DB instance for Aurora DB clusters in a VPC, you use a VPC security group.

    - You can make endpoint and port connections using Transport Layer Security (TLS) / Secure Sockets Layer (SSL). In addition, firewall rules can control whether devices running at your company can open connections to a DB instance.
    - Use RDS encryption to secure your RDS instances and snapshots at rest.
    - You can authenticate to your DB cluster using AWS IAM database authentication. IAM database authentication works with Aurora MySQL and Aurora PostgreSQL. With this authentication method, you don't need to use a password when you connect to a DB cluster. Instead, you use an *authentication token*, which is a unique string of characters that Amazon Aurora generates on request.
- Aurora for MySQL
    - Performance Enhancements
        - Push-Button Compute Scaling
        - Storage Auto-Scaling
        - Low-Latency Read Replicas
        - Serverless Configuration
        - Custom Database Endpoints
        - *Fast insert* accelerates parallel inserts sorted by primary key.
        - ***Aurora MySQL parallel query*** is an optimization that parallelizes some of the I/O and computation involved in processing data-intensive queries.
        - You can use the *high-performance Advanced Auditing* feature in Aurora MySQL to audit database activity. To do so, you enable the collection of audit logs by setting several DB cluster parameters.
    - Scaling
        - Instance scaling - scale your Aurora DB cluster by modifying the DB instance class for each DB instance in the DB cluster.
        - Read scaling - as your read traffic increases, you can create additional Aurora Replicas and connect to them directly to distribute the read load for your DB cluster.



- Aurora for PostgreSQL
  - Performance Enhancements
    - Push-button Compute Scaling
    - Storage Auto-Scaling
    - Low-Latency Read Replicas
    - Custom Database Endpoints
  - Scaling
    - Instance scaling
    - Read scaling
  - Amazon Aurora PostgreSQL now supports logical replication. With logical replication, you can replicate data changes from your Aurora PostgreSQL database to other databases using native PostgreSQL replication slots, or data replication tools such as the AWS Database Migration Service.
  - Rebooting the primary instance of an Amazon Aurora DB cluster also automatically reboots the Aurora Replicas for that DB cluster, in order to re-establish an entry point that guarantees read/write consistency across the DB cluster.

    - You can import data (supported by the PostgreSQL COPY command) stored in an Amazon S3 bucket into a PostgreSQL table.
    - Aurora PostgreSQL support for Kerberos and Microsoft Active Directory provides the benefits of single sign-on and centralized authentication of Aurora PostgreSQL database users. In addition to password-based and IAM-based authentication methods, you can also authenticate using AWS Managed Microsoft AD Service.
- Pricing
    - You are charged for DB instance hours, I/O requests, Backup storage and Data transfer.
    - You can purchase **On-Demand Instances** and pay by the hour for the DB instance hours that you use, or **Reserved Instances** to reserve a DB instance for a one-year or three-year term and receive a significant discount compared to the on-demand DB instance pricing.

Sources:
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
https://aws.amazon.com/rds/aurora/details/mysql-details/
https://aws.amazon.com/rds/aurora/details/postgresql-details/
https://aws.amazon.com/rds/aurora/global-database/
https://aws.amazon.com/rds/aurora/parallel-query/
https://aws.amazon.com/rds/aurora/serverless/
https://aws.amazon.com/rds/aurora/pricing/
https://aws.amazon.com/rds/aurora/faqs/

Amazon Relational Database Service (RDS)

- Industry-standard relational database
- RDS manages backups, software patching, automatic failure detection, and recovery.
- You can have automated backups performed when you need them, or manually create your own backup snapshot. You can use these backups to restore a database.
- Supports **Aurora**, **MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server**.
- You can get high availability with a primary instance and a synchronous secondary instance that you can fail over to when problems occur. You can also use MySQL, MariaDB, or PostgreSQL Read Replicas to increase read scaling.
- Basic building block of RDS is the **DB instance**, which is an isolated database environment in the cloud.
- You can have up to 40 Amazon RDS DB instances.
- Each DB instance runs a **DB engine**.
- You can select the computation and memory capacity of a DB instance, determined by its **DB instance class**. If your needs change over time, you can change DB instances.
- Each DB instance has minimum and maximum storage requirements depending on the storage type and the database engine it supports.
- You can run your DB instance in several AZs, an option called a **Multi-AZ deployment**. Amazon automatically provisions and maintains a secondary standby DB instance in a different AZ. Your primary DB instance is synchronously replicated across AZs to the secondary instance to provide data redundancy, failover support, eliminate I/O freezes, and minimize latency spikes during system backups.

DB Engines

- MySQL DB instance:
    - You're limited to 10,000 tables if you are either using Provisioned IOPS or General Purpose storage and the DB instance is 200 GiB or larger in size.
    - You're limited to 1000 tables if you are either using Standard or General Purpose storage and the DB instance is less than 200 GiB in size.
    - MySQL database instances can be launched with up to 64 TiB of storage and provisioned I/O performance of up to 80,000 IOPS.
    - The Point-In-Time Restore and snapshot restore features of Amazon RDS for MySQL require a crash-recoverable storage engine and are supported for the InnoDB storage engine only.
    - InnoDB is the recommended and supported storage engine for MySQL DB instances on Amazon RDS.
    - The **database name** is the name of a database hosted in your DB instance. Databases hosted by the same DB instance must have a unique name within that instance.
    - You can now enforce password policies in your Amazon RDS for MySQL databases using the MySQL *validate_password* plugin. This improves the security of your databases by defining minimum password length, required characters, and other rules.

- MariaDB instance:
  - The point-in-time restore and snapshot restore features of Amazon RDS for MariaDB require a crash-recoverable storage engine.
  - MariaDB database instances can be launched with up to 64 TiB of storage and provisioned I/O performance of up to 80,000 IOPS.
  - The **database name** is the name of a database hosted in your DB instance. Databases hosted by the same DB instance must have a unique name within that instance.
- PostgreSQL instance:
  - You can improve performance with PostgreSQL on Amazon RDS when loading data into a DB instance and when using the PostgreSQL autovacuum feature.
  - PostgreSQL database instances can be launched with up to 64 TiB of storage and provisioned I/O performance of up to 80,000 IOPS.
  - The **database name** is the unique name of a database hosted in your DB instance, and is not required when creating a DB instance.
- Microsoft SQL Server instance:
  - Up to 10 can be SQL Server DB instances under the "License Included" model. You can have 40 DB instances for SQL Server under the "BYOL" licensing model.
  - The maximum number of databases supported on a DB instance depends on the instance class type and the availability mode—Single-AZ, Multi-AZ Database Mirroring (DBM), or Multi-AZ Availability Groups (AGs). The Microsoft SQL Server system databases don't count toward this limit.

Instance Class Type	Single-AZ	Multi-AZ with DBM	Multi-AZ with Always On AGs
db.*.micro to db.*.medium	30	N/A	N/A
db.*.large	30	30	30
db.*.xlarge to db.*.16xlarge	100	50	75
db.*.24xlarge	100	50	100

  - For Multi-AZ enabled instances:
    - Use Amazon RDS DB events to monitor failovers.
    - If your application caches DNS values, set time to live (TTL) to less than 30 seconds.
    - AWS recommends to NOT enable Simple Recover, Offline, or Read-only modes because they turn off transaction logging, which is required for Multi-AZ.
    - Test to determine how long it takes for your DB instance to failover.
    - Deploy your applications in all Availability Zones.
  - **Database name** is not a supported parameter.

- Oracle instance:
    - Up to 10 can be Oracle instances under the "License Included" model. You can have 40 DB instances for Oracle under the "BYOL" licensing model.
    - **Database name** is used to set the value of ORACLE_SID, which must be supplied when connecting to the Oracle RDS instance.
    - You can now create Amazon RDS for Oracle database instances with up to 64 TiB of storage (32 TiB originally) and provisioned I/O performance of up to 80,000 IOPS.
- DB Instance:
    - Class Types
        - Standard
        - Memory Optimized
        - Burstable Performance
    - You can change the CPU and memory available to a DB instance by changing its DB instance class. Specify the following processor features to optimize your DB instance for specific workloads or business needs:
        - Number of CPU cores
        - Threads per core
    - Endpoint: rds.*<region>*.amazonaws.com
    - Storage
        - Amazon RDS for MySQL, MariaDB, PostgreSQL, Oracle, and Microsoft SQL Server use Amazon EBS volumes for database and log storage.
        - Storage types :

            General Purpose SSD (gp2)
            - MySQL, MariaDB, Oracle, and PostgreSQL DB instances: 20 GiB–64 TiB storage size
            - SQL Server for Enterprise, Standard, Web, and Express editions: 20 GiB–16 TiB storage size

            Provisioned IOPS SSD (io1)

Database Engine	Range of Provisioned IOPS	Range of Storage
MariaDB	1,000–80,000	100 GiB–64 TiB
SQL Server, Enterprise and Standard editions	1000–32,000 or 64,000 for Nitro-based m5 instance types	20 GiB–16 TiB
SQL Server, Web and Express editions	1000–32,000 or 64,000 for Nitro-based m5 instance types	100 GiB–16 TiB
MySQL	1,000–80,000	100 GiB–64 TiB

Oracle	1,000–80,000	100 GiB–64 TiB
PostgreSQL	1,000–80,000	100 GiB–64 TiB

        - For production OLTP use cases, use **Multi-AZ deployments** for enhanced fault tolerance with Provisioned IOPS storage for fast and predictable performance.
            - You can use PIOPS storage with Read Replicas for MySQL, MariaDB or PostgreSQL.
        - Magnetic
            - Doesn't allow you to scale storage when using the SQL Server database engine.
            - Doesn't support elastic volumes.
            - Limited to a maximum size of 3 TiB.
            - Limited to a maximum of 1,000 IOPS.
- Instance Lifecycle - includes creating, modifying, maintaining and upgrading, performing backups and restores, rebooting, and deleting the instance.
    - You can't stop an **Amazon RDS for SQL Server DB** instance in a Multi-AZ configuration.
    - You can stop a DB instance for up to seven days. If you do not manually start your DB instance after seven days, your DB instance is automatically started.
    - You can't stop a DB instance that has a Read Replica, or that is a Read Replica.
    - You can't modify a stopped DB instance.
    - To delete a DB instance, you must specify the name of the instance and whether to take a final DB snapshot of the instance.
    - You can enable **deletion protection** so that users can't delete a database. Deletion protection is disabled by default.

RDS Storage Auto Scaling

- RDS Storage Auto Scaling automatically scales storage capacity in response to growing database workloads, with zero downtime.
- Amazon RDS for MariaDB, Amazon RDS for MySQL, Amazon RDS for PostgreSQL, Amazon RDS for SQL Server and Amazon RDS for Oracle support RDS Storage Auto Scaling.
- RDS Storage Auto Scaling continuously monitors actual storage consumption, and scales capacity up automatically when actual utilization approaches provisioned storage capacity.
- Auto Scaling works with new and existing database instances.

Security

- Security Groups

    - **DB Security Groups** - controls access to a DB instance that is not in a VPC. By default, network access is turned off to a DB instance. This SG is for the EC2-Classic platform.
    - **VPC Security Groups** - controls access to a DB instance inside a VPC. This SG is for the EC2-VPC platform.
    - **EC2 Security Groups** - controls access to an EC2 instance and can be used with a DB instance.
- Practices
    - Assign an individual **IAM** account to each person who manages RDS resources. Do not use AWS root credentials to manage RDS resources.
    - Grant each user the minimum set of permissions required to perform his or her duties.
    - Use IAM groups to effectively manage permissions for multiple users.
    - Rotate your IAM credentials regularly.
    - Use **security groups** to control what IP addresses or Amazon EC2 instances can connect to your databases on a DB instance.
    - Run your DB instance in an Amazon Virtual Private Cloud (**VPC**) for the greatest possible network access control.
    - Use **Secure Socket Layer (SSL) connections** with DB instances running the MySQL, MariaDB, PostgreSQL, Oracle, or Microsoft SQL Server database engines.
    - Use RDS encryption to secure your RDS instances and snapshots at rest.
    - Use the security features of your DB engine to control who can log in to the databases on a DB instance.
- A *resource owner* is the AWS account that created a resource. That is, the resource owner is the AWS account of the *principal entity* (the root account, an IAM user, or an IAM role) that authenticates the request that creates the resource.
- A *permissions policy* describes who has access to what. Policies attached to an IAM identity are *identity-based policies* (IAM policies) and policies attached to a resource are *resource-based policies*. Amazon RDS supports only identity-based policies (IAM policies).
- MySQL and PostgreSQL both support **IAM database authentication**.
- Encryption
    - At rest and in-transit.
    - Manage keys used for encrypted DB instances using the AWS KMS. KMS encryption keys are specific to the region that they are created in.
    - RDS encryption is currently available for all database engines and storage types. RDS encryption is available for most DB instance classes.
    - You can't have an encrypted Read Replica of an unencrypted DB instance or an unencrypted Read Replica of an encrypted DB instance.
    - You can't restore an unencrypted backup or snapshot to an encrypted DB instance.
    - You can use **SSL** from your application to encrypt a connection to a DB instance running MySQL, MariaDB, SQL Server, Oracle, or PostgreSQL.
- Amazon RDS supports the following scenarios for accessing a DB instance in a VPC:

DB Instance	Accessed By
In a VPC	An EC2 Instance in the Same VPC
	An EC2 Instance in a Different VPC
	An EC2 Instance Not in a VPC
	A Client Application Through the Internet
Not in a VPC	An EC2 Instance in a VPC
	An EC2 Instance Not in a VPC
	A Client Application Through the Internet

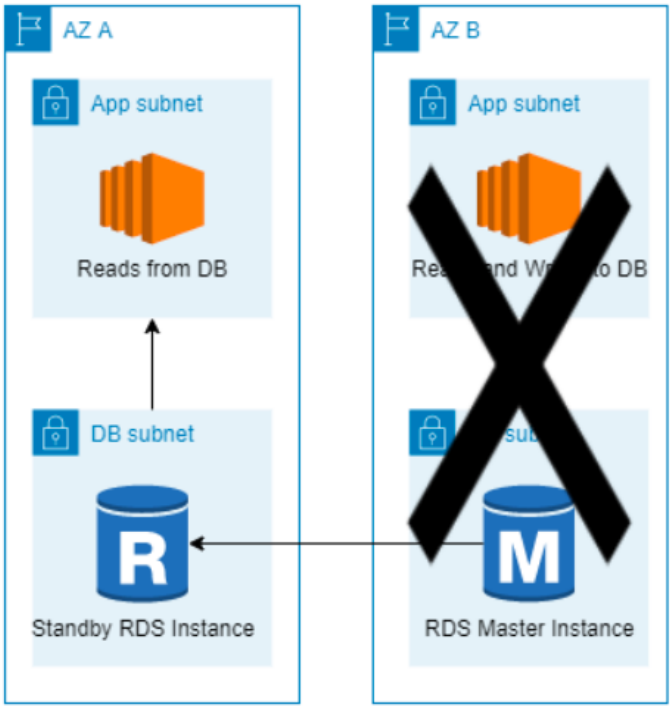
- Working with a DB Instance in a VPC
  - Your VPC must have **at least two subnets**. These subnets must be in two **different Availability Zones** in the region where you want to deploy your DB instance.
  - If you want your DB instance in the VPC to be publicly accessible, you must enable the VPC attributes *DNS hostnames* and *DNS resolution*.
  - Your VPC must have a DB subnet group that you create.
  - Your VPC must have a VPC security group that allows access to the DB instance.
  - The CIDR blocks in each of your subnets must be large enough to accommodate spare IP addresses for Amazon RDS to use during maintenance activities, including failover and compute scaling.
  - When an option group is assigned to a DB instance, it is linked to the supported platform the DB instance is on, either VPC or EC2-Classic.
  - If you restore a DB instance into a different VPC or onto a different platform, you must either assign the default option group to the DB instance, assign an option group that is linked to that VPC or platform, or create a new option group and assign it to the DB instance.

Tagging

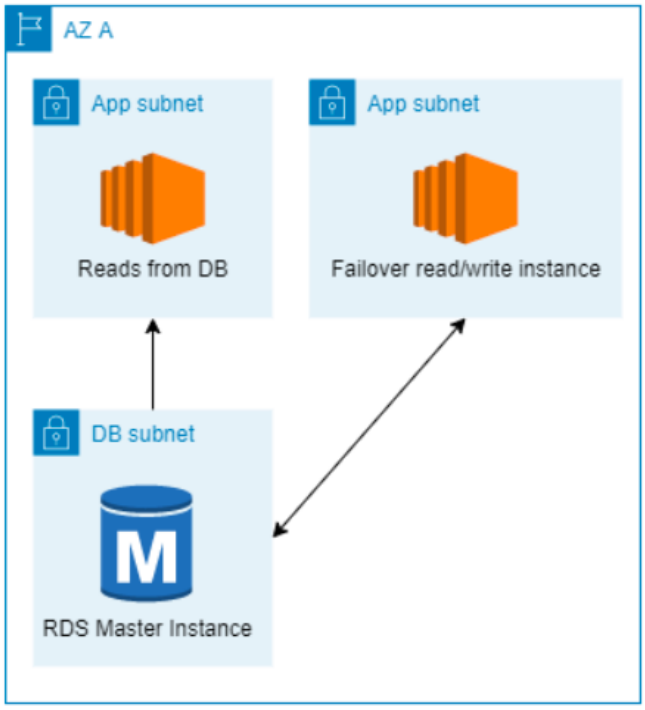
- An RDS tag is a **name-value pair** that you define and associate with an RDS resource. The name is referred to as the key. Supplying a value for the key is optional.
- All Amazon RDS resources can be tagged.
- Use tags to organize your AWS bill to reflect your own cost structure.
- A *tag set* can contain as many as 50 tags, or it can be empty.

High Availability using Multi-AZ

- Multi-AZ deployments for **Oracle, PostgreSQL, MySQL, and MariaDB** DB instances use **Amazon's failover technology**. **SQL Server DB** instances use **SQL Server Mirroring**.
- **Amazon RDS for SQL Server** offers **Always On Availability Groups** for the Multi-AZ configuration in all AWS Regions
- You can modify a DB instance in a Single-AZ deployment to a Multi-AZ deployment.
- The primary DB instance switches over automatically to the standby replica if any of the following conditions occur:
    - An Availability Zone outage
    - The primary DB instance fails
    - The DB instance's server type is changed
    - The operating system of the DB instance is undergoing software patching
    - A manual failover of the DB instance was initiated using **Reboot with failover**



Standby DB instance in another subnet. AZ B experiences outage.

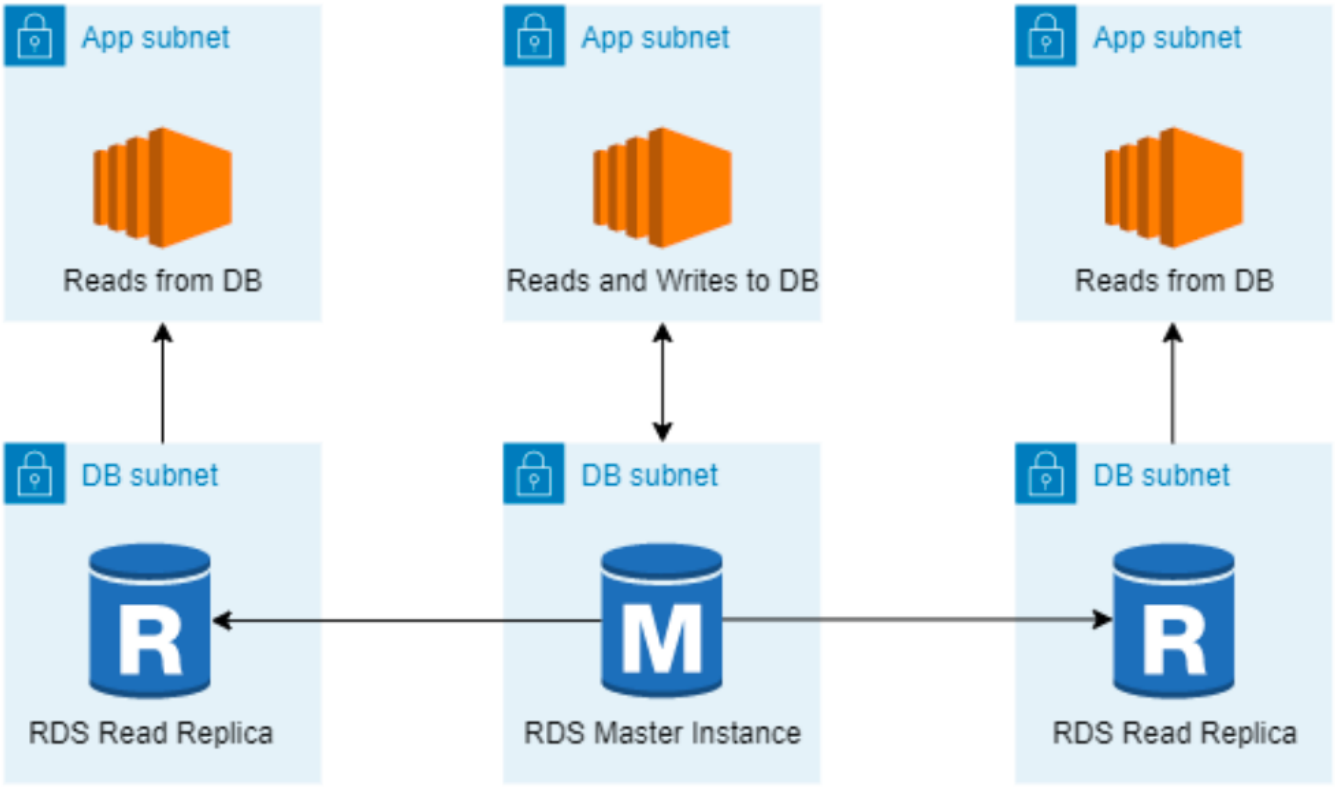


Read replica promoted to Master instance. Spawn a new read/write app in AZ A. Reroute traffic using failover policies.

Read Replicas

- Updates made to the source DB instance are asynchronously copied to the Read Replica.

- You can reduce the load on your source DB instance by routing read queries from your applications to the Read Replica.



- You can elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads.
- You can create a Read Replica that has a different storage type from the source DB instance.
- A source DB instance can have cross-region Read Replicas in multiple regions. Due to the limit on the number of access control list (ACL) entries for a VPC, cannot have more than five cross-region Read Replica instances.
- PostgreSQL does **physical replication**. MySQL and MariaDB do **logical replication**.
- You can create a manual snapshot of a PostgreSQL Read Replica, but you can't enable automatic backups. You can enable automatic backups on a MySQL or MariaDB Read Replica.
- When creating a Read Replica, there are a few things to consider.
    - Enable automatic backups on the source DB instance by setting the backup retention period to a value other than 0.
    - Automatic backups are supported only for Read Replicas running MySQL 5.6 and later.
- Oracle supports Read Replicas with Active Data Guard for customers who are using the **Bring Your Own License model with Oracle Database Enterprise Edition and have licensed the Active Data Guard Option**. This feature allows RDS Oracle customers to offload their read workload from the primary DB Instance, and also supports the scalability of read workloads over a farm of up to **five read replicas**.

- You can promote a Read Replica to become a standalone DB instance.

Multi-AZ Deployments	Read Replicas
Synchronous replication - highly durable	Asynchronous replication - highly scalable
Only database engine on primarily instance is active	All read replicas are accesible and can be used for read scaling
Automated backups are taken from standby	No backups configured by default
Always span two Availability Zones within a single Region	Can be within an Availability Zone, Cross-AZ, or Cross-Region
Database engine version upgrades happen on primary	Database engine version upgrade is independent from source instance
Automatic failover to standby when a problem is detected	Can be manually promoted to a standalone database instance

Backups and Restores

- Your DB instance must be in the **ACTIVE state** for automated backups to occur. Automated backups and automated snapshots don't occur while a copy is executing in the same region for the same DB instance.
- The first snapshot of a DB instance contains the data for the full DB instance. Subsequent snapshots of the same DB instance are incremental.
- The default backup retention period is one day if you create the DB instance using the RDS API or the AWS CLI, or seven days if you used the AWS Console.
- Manual snapshot limits are limited to 100 per region.
- You can copy a snapshot within the same AWS Region, you can copy a snapshot across AWS Regions, and you can copy a snapshot across AWS accounts.
- When you restore a DB instance to a point in time, the default DB parameter and default DB security group is applied to the new DB instance.
- When you restore an Oracle DB instance to a point in time, you can specify a different Oracle DB engine, license model, and DBName to be used.
- When you restore a SQL Server DB instance to a point in time, each database within that instance is restored to a point in time within 1 second of each other database within the instance.
- You can retain Amazon RDS automated backups (system snapshots and transaction logs) when you delete a database instance.
- You can export Amazon RDS or Amazon Aurora snapshots to Amazon S3 as Apache Parquet.

Monitoring

- Amazon CloudWatch
- RDS Events
  - An Amazon RDS event is created when the reboot is completed.
  - Be notified when changes occur with a DB instance, DB snapshot, DB parameter group, or DB security group.
  - Uses the Amazon Simple Notification Service (SNS) to provide notification when an Amazon RDS event occurs.
- Database log files
- Using enhanced monitoring to identify operating system issues for the following:
  - MariaDB
  - Microsoft SQL Server
  - MySQL version 5.5 or later
  - Oracle
  - PostgreSQL
- Enhanced monitoring is available for all DB instance classes except for db.m1.small. Enhanced Monitoring is available in all regions except for AWS GovCloud (US).
- Enhance monitoring metrics include:
  - IOPS - the number of I/O operations completed each second.
  - Latency - the elapsed time between the submission of an I/O request and its completion.
  - Throughput - the number of bytes each second that are transferred to or from disk.
  - Queue Depth - the number of I/O requests in the queue waiting to be serviced.
- CloudWatch gathers metrics about CPU utilization **from the hypervisor** for a DB instance, and Enhanced Monitoring gathers its metrics **from an agent** on the instance.
- Instance Status - indicates the health of the instance. Here are some statuses:

DB Instance Status	Billed	Description
available	Billed	The instance is healthy and available.
backing-up	Billed	The instance is currently being backed up.
creating	Not billed	The instance is being created. The instance is inaccessible while it is being created.
deleting	Not billed	The instance is being deleted.
failed	Not billed	The instance has failed and Amazon RDS can't recover it. Perform a point-in-time restore to the latest restorable time of the instance to recover the data.

maintenance	Billed	Amazon RDS is applying a maintenance update to the DB instance. This status is used for instance-level maintenance that RDS schedules well in advance.
rebooting	Billed	The instance is being rebooted because of a customer request or an Amazon RDS process that requires the rebooting of the instance.
starting	Billed for storage	The DB instance is starting.
stopped	Billed for storage	The DB instance is stopped.
stopping	Billed for storage	The DB instance is being stopped.
storage-full	Billed	The instance has reached its storage capacity allocation. This is a critical status, and we recommend that you fix this issue immediately. To do so, scale up your storage by modifying the DB instance. To avoid this situation, set Amazon CloudWatch alarms to warn you when storage space is getting low.

- RDS **Performance Insights** monitors your DB instance load so that you can analyze and troubleshoot your database performance. You can visualize the database load and filter the load by waits, SQL statements, hosts, or users.
- CloudTrail captures all API calls for RDS as events.

Pricing

- With Amazon RDS, you pay only for the RDS instances that are active.
- The data transferred for cross-region replication incurs RDS data transfer charges.
- Instances are billed for DB instance hours (per second), Storage (per GiB per month), I/O requests (per 1 million requests per month), Provisioned IOPS (per IOPS per month), Backup storage (per GiB per month), and Data transfer (per GB).
    - Amazon RDS is billed in one-second increments for database instances and attached storage. Pricing is still listed on a per-hour basis, but bills are now calculated down to the second and show usage in decimal form. There is a 10 minute minimum charge when an instance is created, restored or started.
- RDS purchasing options:
    - **On-Demand Instances** – Pay by the hour for the DB instance hours that you use.

    - **Reserved Instances** – Reserve a DB instance for a one-year or three-year term and receive a significant discount compared to the on-demand DB instance pricing.
- You are charged for using Enhanced Monitoring.
- Amazon RDS is now billed in one-second increments for database instances and attached storage. Pricing is still listed on a per-hour basis, but bills are now calculated down to the second and show usage in decimal form. There is a 10 minute minimum charge when an instance is created, restored or started.

Best Practices

- Monitor your memory, CPU, and storage usage.
- Scale up your DB instance when you are approaching storage capacity limits.
- Enable automatic backups and set the backup window to occur during the daily low in write IOPS.
- If your database workload requires more I/O than you have provisioned, recovery after a failover or database failure will be slow. Increase the I/O capacity of a DB instance by:
    - Migrate to a DB instance class with High I/O capacity.
    - Convert from standard storage to either General Purpose or Provisioned IOPS storage, depending on how much of an increase you need.
    - If you convert to Provisioned IOPS storage, make sure you also use a DB instance class that is optimized for Provisioned IOPS..
    - If you are already using Provisioned IOPS storage, provision additional throughput capacity.
- If your client application is caching the Domain Name Service (DNS) data of your DB instances, set a time-to-live (TTL) value of less than 30 seconds.
- Test failover for your DB instance.

Sources:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
https://aws.amazon.com/rds/features/
https://aws.amazon.com/rds/pricing/
https://aws.amazon.com/rds/faqs/

Amazon DynamoDB

- NoSQL database service that provides fast and predictable performance with seamless scalability.
- Offers encryption at rest.
- You can create database tables that can store and retrieve any amount of data, and serve any level of request traffic.
- You can scale up or scale down your tables' throughput capacity without downtime or performance degradation, and use the AWS Management Console to monitor resource utilization and performance metrics.
- Provides on-demand backup capability as well as enable point-in-time recovery for your DynamoDB tables. With point-in-time recovery, you can restore that table to any point in time during the **last 35 days**.
- All of your data is stored in partitions, backed by solid state disks (SSDs) and automatically replicated across multiple AZs in an AWS region, providing built-in high availability and data durability.
- You can create tables that are automatically replicated across two or more AWS Regions, with full support for multi-master writes.
- AWS now specifies the IP address ranges for Amazon DynamoDB endpoints. You can use these IP address ranges in your routing and firewall policies to control outbound application traffic. You can also use these ranges to control outbound traffic for applications in your Amazon Virtual Private Cloud, behind AWS Virtual Private Network or AWS Direct Connect.

Core Components

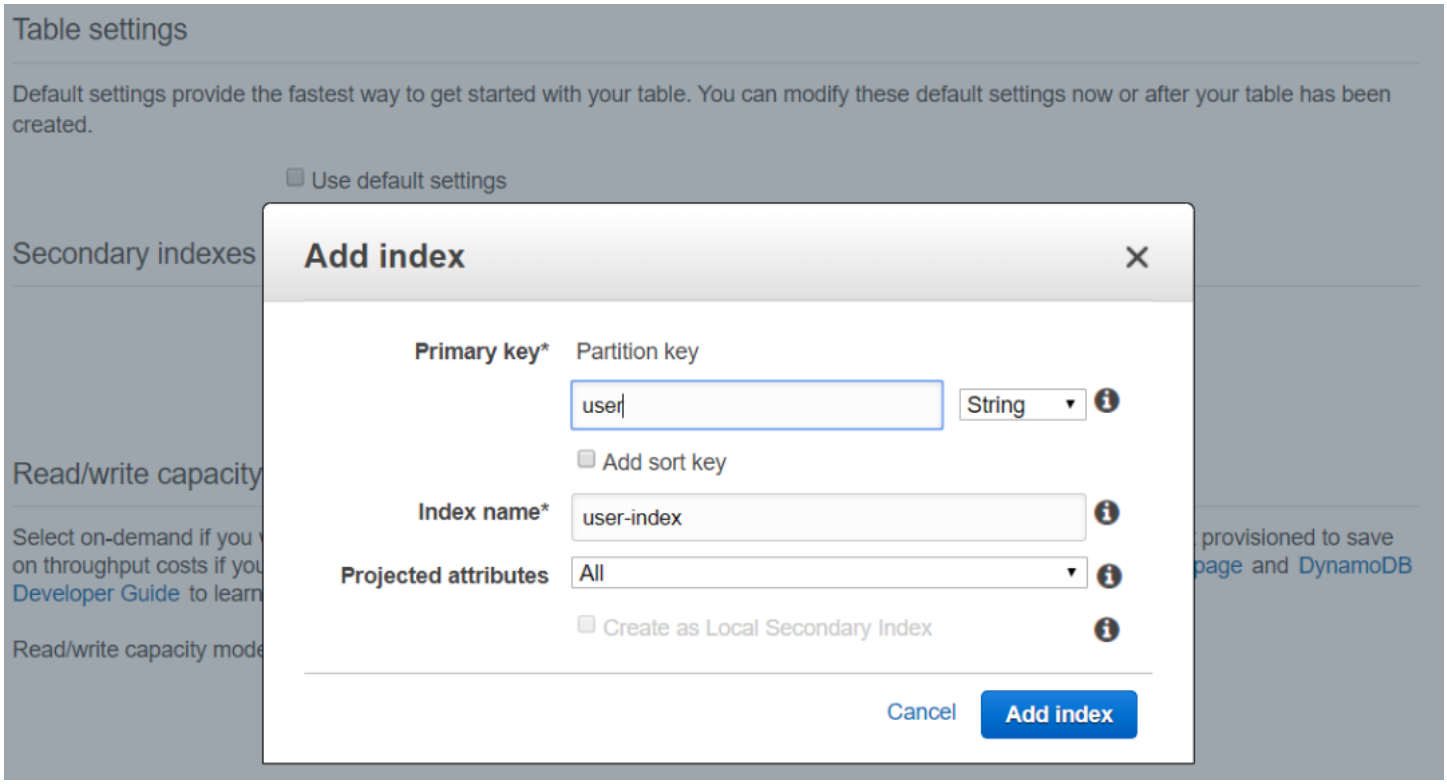
- **Tables** - a collection of items
    - DynamoDB stores data in a table, which is a collection of data.
    - Are schemaless.
    - There is an initial limit of 256 tables per region.
- **Items** - a collection of attributes
    - DynamoDB uses **primary keys** to uniquely identify each item in a table and **secondary indexes** to provide more querying flexibility.
    - Each table contains zero or more items.
- **Attributes** - a fundamental data element
    - DynamoDB supports nested attributes up to 32 levels deep.
- **Primary Key** - uniquely identifies each item in the table, so that no two items can have the same key. Must be scalar.
    - **Partition key** - a simple primary key, composed of one attribute.
    - **Partition key and sort key** (*composite primary key*) - composed of two attributes.
    - DynamoDB uses the partition key value as input to an internal hash function. The output from the hash function determines the partition in which the item will be stored. All items with the

same partition key are stored together, in sorted order by sort key value. If no sort key is used, no two items can have the same partition key value.

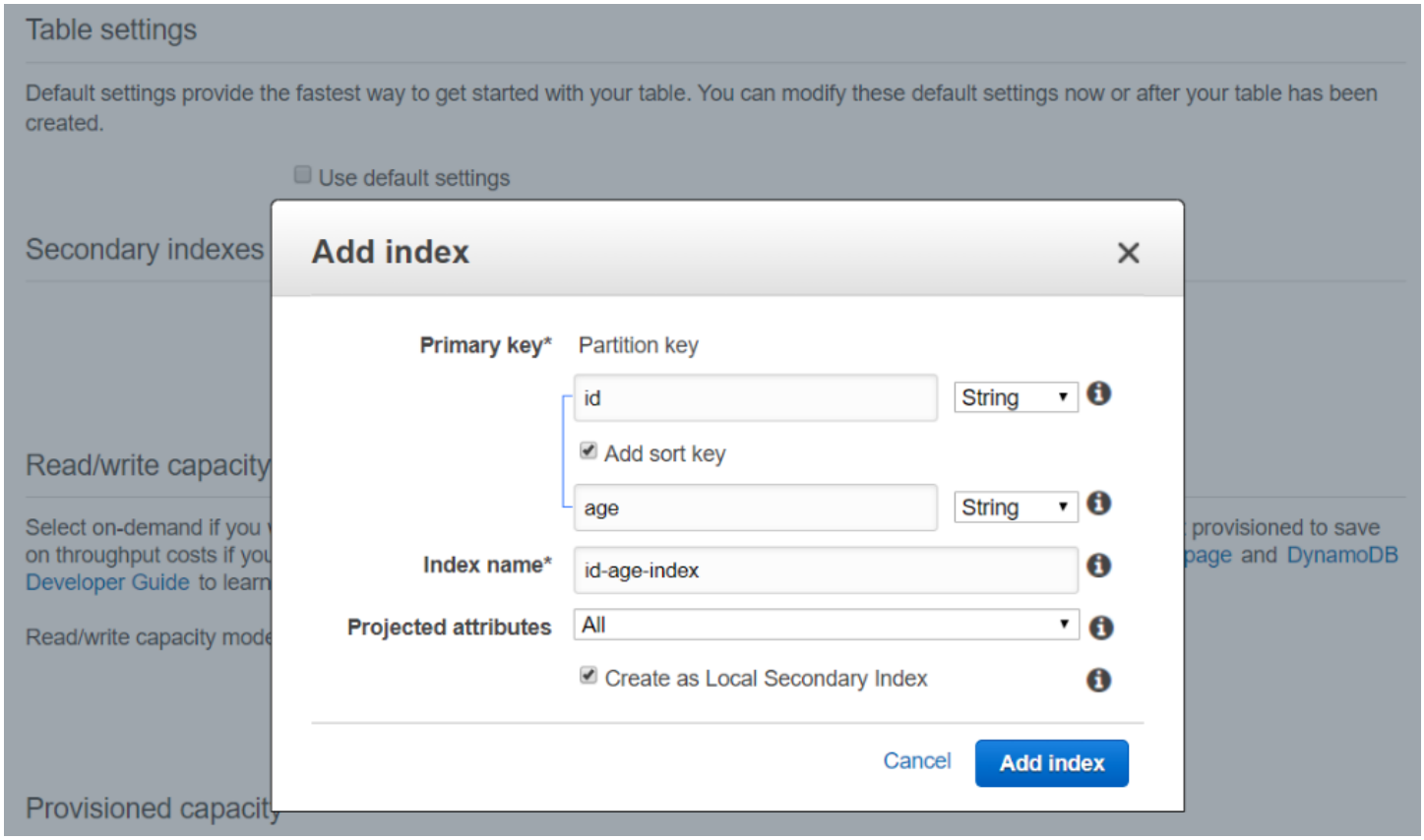
CDA Exam Notes:

To maximize performance, try to use an attribute that has mostly unique values as the partition key. This is to avoid an uneven data distribution if you have a sort key, or a partition key clash if you don't have a sort key. Best types of relationships are one-to-one and one-to-few (like one student to one or few classes), while the worst type is one-to-many (like one class to many students).

- **Secondary Indexes** - lets you query the data in the table using an alternate key, in addition to queries against the primary key.
    - You can create one or more secondary indexes on a table.
    - Two kinds of indexes:
        - **Global secondary index** – An index with a partition key and sort key that can be different from those on the table.



- **Local secondary index** – An index that has the same partition key as the table, but a different sort key.



  - You can define up to 20 global secondary indexes and 5 local secondary indexes per table.
- **DynamoDB Streams** - an optional feature that captures data modification events in DynamoDB tables.
  - The naming convention for DynamoDB Streams endpoints is *streams.dynamodb..amazonaws.com*
  - Each event is represented by a *stream record*, and captures the following events:
    - A new item is added to the table: captures an image of the entire item, including all of its attributes.
    - An item is updated: captures the "before" and "after" image of any attributes that were modified in the item.
    - An item is deleted from the table: captures an image of the entire item before it was deleted.
  - Each stream record also contains the name of the table, the event timestamp, and other metadata.
  - Stream records are organized into groups, or ***shards***. Each shard acts as a container for multiple stream records, and contains information required for accessing and iterating through these records.

  - Stream records have a lifetime of 24 hours; after that, they are automatically removed from the stream.
  - You can use DynamoDB Streams together with AWS Lambda to create a *trigger*, which is a code that executes automatically whenever an event of interest appears in a stream.
  - DynamoDB Streams enables powerful solutions such as data replication within and across Regions, materialized views of data in DynamoDB tables, data analysis using Kinesis materialized views, and much more.

Data Types for Attributes

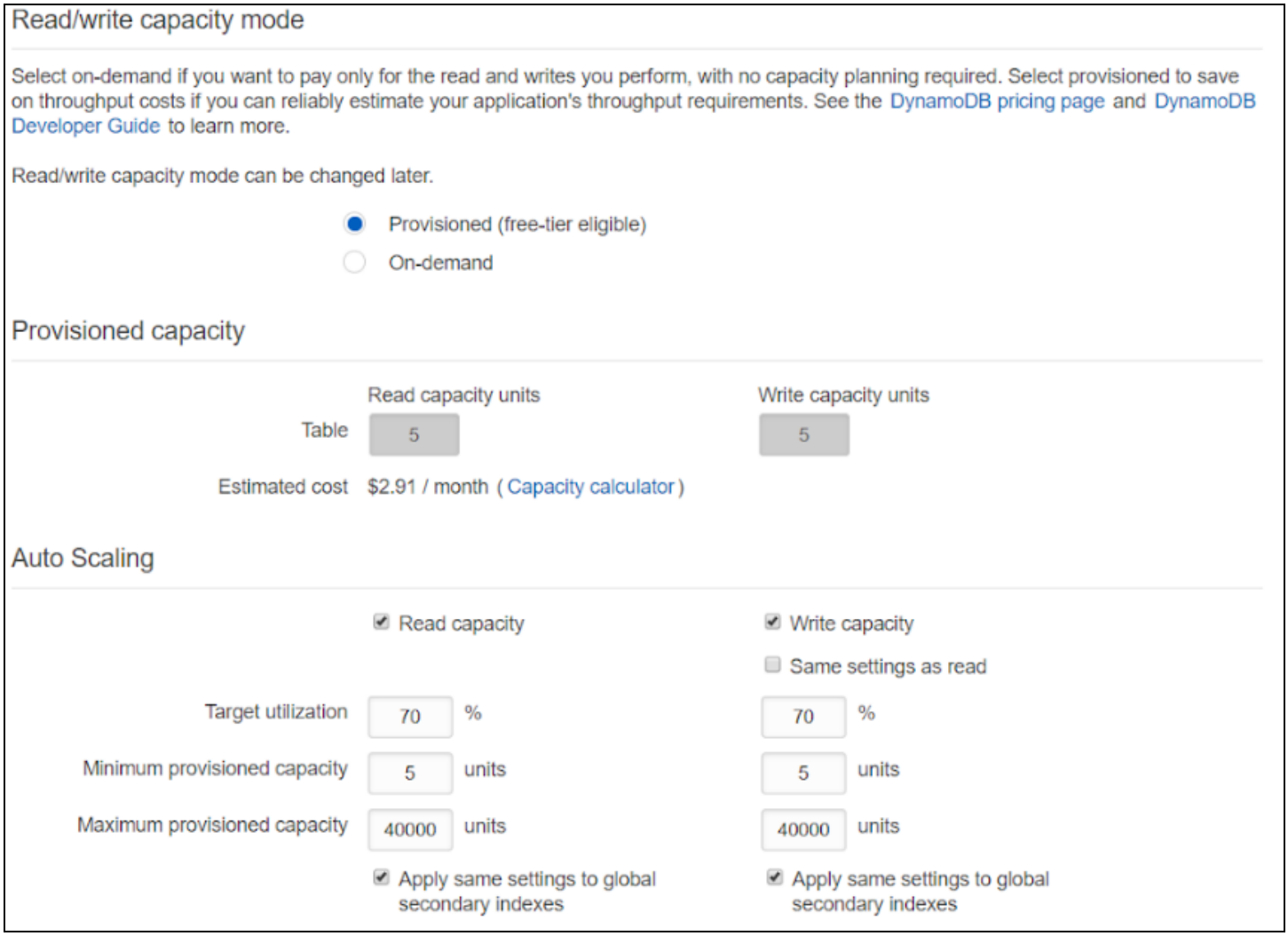
- **Scalar Types** – A scalar type can represent exactly one value. The scalar types are number, string, binary, Boolean, and null. Primary keys should be scalar types.
- **Document Types** – A document type can represent a complex structure with nested attributes—such as you would find in a JSON document. The document types are list and map.
- **Set Types** – A set type can represent multiple scalar values. The set types are string set, number set, and binary set.

Other Notes:

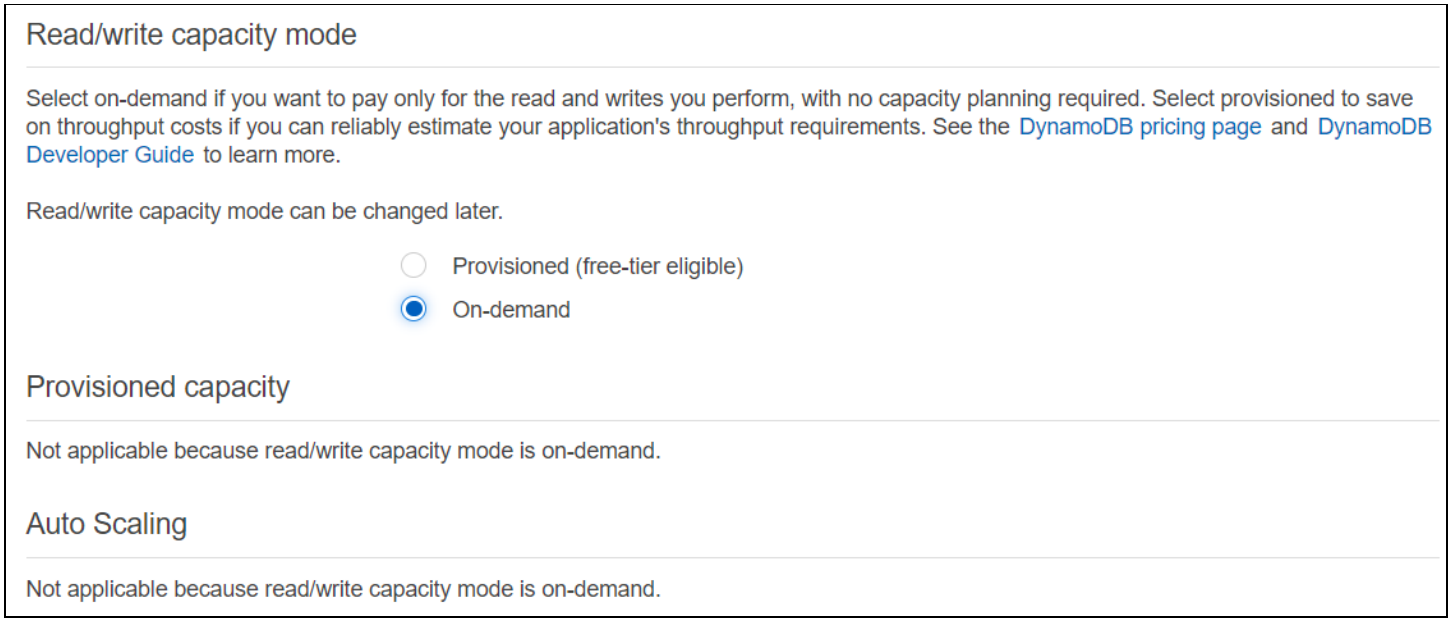
- When you read data from a DynamoDB table, the response might not reflect the results of a recently completed write operation. The response might include some stale data, but you should **eventually have consistent reads**.
- When you request a **strongly consistent read**, DynamoDB returns a response with the most up-to-date data, reflecting the updates from all prior write operations that were successful. A strongly consistent read might not be available if there is a network delay or outage.
- DynamoDB does not support strongly consistent reads across AWS regions
- When you create a table or index in DynamoDB, you must specify your throughput capacity requirements for read and write activity in terms of:
  - One **read capacity unit** represents one strongly consistent read per second, or two eventually consistent reads per second, for an item up to 4 KB in size. If you need to read an item that is larger than 4 KB, DynamoDB will need to consume additional read capacity units.
  - One **write capacity unit** represents one write per second for an item up to 1 KB in size. If you need to write an item that is larger than 1 KB, DynamoDB will need to consume additional write capacity units.
- *Throttling* prevents your application from consuming too many capacity units. DynamoDB can throttle read or write requests that exceed the throughput settings for a table, and can also throttle read request exceeds for an index.
- When a request is throttled, it fails with an **HTTP 400** code (Bad Request) and a *ProvisionedThroughputExceededException*.

Throughput Management

- Provisioned throughput - manually defined maximum amount of capacity that an application can consume from a table or index. If your application exceeds your provisioned throughput settings, it is subject to request throttling. Free tier eligible.
    - DynamoDB auto scaling
        - Define a range (upper and lower limits) for **read and write capacity units**, and define a target utilization percentage within that range.
        - A table or a global secondary index can increase its **provisioned read and write capacity** to handle sudden increases in traffic, without request throttling.
        - DynamoDB auto scaling can decrease the throughput when the workload decreases so that you don't pay for unused provisioned capacity.
    - Reserved capacity - with reserved capacity, you pay a one-time upfront fee and commit to a minimum usage level over a period of time, for cost-saving solutions.



- Amazon DynamoDB on-demand is a flexible capacity mode for DynamoDB capable of serving thousands of requests per second without capacity planning. When you choose on-demand capacity mode, DynamoDB instantly accommodates your workloads as they ramp up or down to any previously reached traffic level. If a workload's traffic level hits a new peak, DynamoDB adapts rapidly to accommodate the workload. DynamoDB on-demand offers simple pay-per-request pricing for read and write requests so that you only pay for what you use, making it easy to balance costs and performance.



Capacity Unit Consumption

- CUC for Reads - strongly consistent read request consumes one read capacity unit, while an eventually consistent read request consumes 0.5 of a read capacity unit.
    - GetItem - reads a single item from a table.
    - BatchGetItem - reads up to 100 items, from one or more tables.
    - Query - reads multiple items that have the same partition key value.
    - Scan - reads all of the items in a table
- CUC for Writes
    - PutItem - writes a single item to a table.
    - UpdateItem - modifies a single item in the table.
    - DeleteItem - removes a single item from a table.
    - BatchWriteItem - writes up to 25 items to one or more tables.

DynamoDB Auto Scaling

- When you use the AWS Management Console to create a new table, DynamoDB auto scaling is enabled for that table by default.
- Uses the AWS Application Auto Scaling service to dynamically adjust provisioned throughput capacity on your behalf, in response to actual traffic patterns.
- You create a *scaling policy* for a table or a global secondary index. The scaling policy specifies whether you want to scale read capacity or write capacity (or both), and the minimum and maximum provisioned capacity unit settings for the table or index. The scaling policy also contains a *target utilization*, which is the percentage of consumed provisioned throughput at a point in time.
- DynamoDB auto scaling doesn't prevent you from manually modifying provisioned throughput settings.

- If you enable DynamoDB auto scaling for a table that has one or more global secondary indexes, AWS highly recommends that you also apply auto scaling uniformly to those indexes.

Tagging

- Tags can help you:
  - Quickly identify a resource based on the tags you've assigned to it.
  - See AWS bills broken down by tags.
- Each DynamoDB table can have only one tag with the same key. If you try to add an existing tag (same key), the existing tag value will be updated to the new value.
- Maximum number of tags per resource: 50

DynamoDB Items

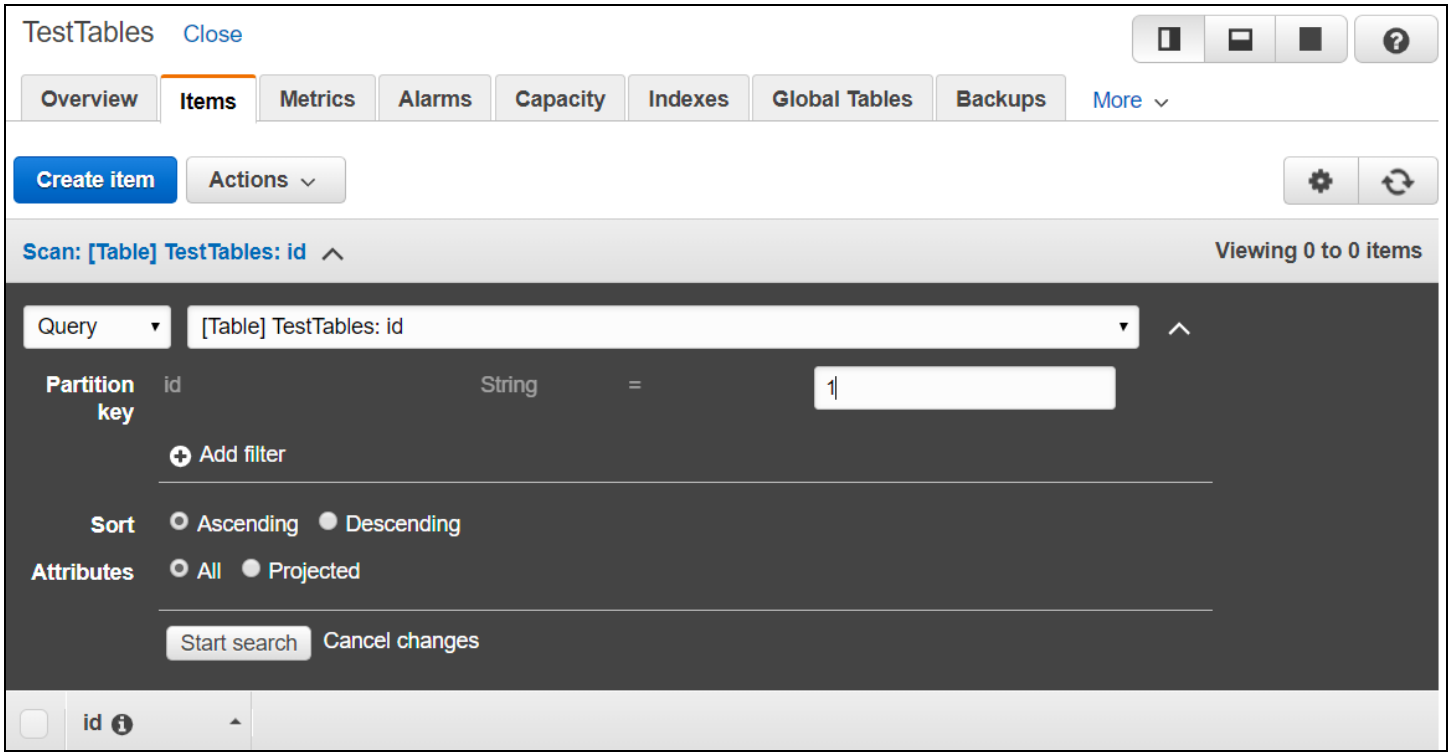
- You can use the *UpdateItem* operation to implement an **atomic counter** - a numeric attribute that is incremented, unconditionally, without interfering with other write requests.
- DynamoDB optionally supports conditional writes for these operations: *PutItem, UpdateItem, DeleteItem*. A conditional write will succeed only if the item attributes meet one or more expected conditions.
- Conditional writes can be *idempotent* if the conditional check is on the same attribute that is being updated. DynamoDB performs a given write request only if certain attribute values in the item match what you expect them to be at the time of the request.
- Expressions
  - To get only a few attributes of an item, use a **projection expression**.
  - An **expression attribute name** is a placeholder that you use in an expression, as an alternative to an actual attribute name. An expression attribute name must begin with a #, and be followed by one or more alphanumeric characters.
  - **Expression attribute values** are substitutes for the actual values that you want to compare — values that you might not know until runtime. An expression attribute value must begin with a :, and be followed by one or more alphanumeric characters.
  - For *PutItem, UpdateItem* and *DeleteItem* operations, you can specify a **condition expression** to determine which items should be modified. If the condition expression evaluates to true, the operation succeeds; otherwise, the operation fails.
  - An **update expression** specifies how *UpdateItem* will modify the attributes of an item—for example, setting a scalar value, or removing elements from a list or a map.

Time To Live (TTL)

- Allows you to define when items in a table expire so that they can be automatically deleted from the database.

DynamoDB Queries

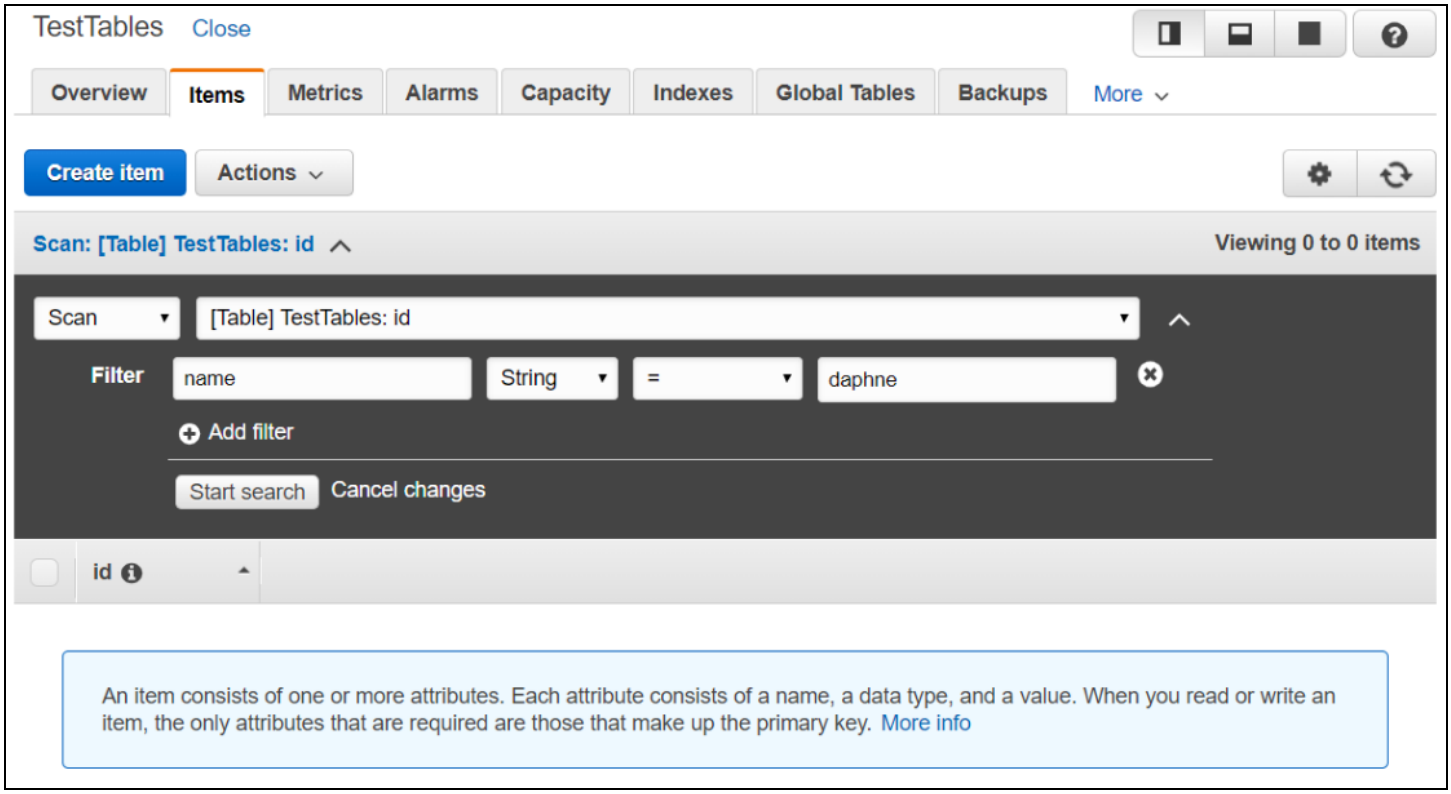
- The *Query* operation finds items based on primary key values. You can query any table or secondary index that has a composite primary key (a partition key and a sort key).



- A key condition expression is a search criteria that determines the items to be read from the table or index.
- You must specify the partition key name and value as an equality condition.
- You can optionally provide a second condition for the sort key. The sort key condition must use one of the following comparison operators: =, <, <=, >, >=, BETWEEN, AND
- A single *Query* operation can retrieve a maximum of 1 MB of data.
- For further refining of Query results, you can optionally provide a **filter expression**, to determine which items within the Query results should be returned to you. All of the other results are discarded.
- The Query operation allows you to limit the number of items that it returns in the result by setting the **Limit** parameter to the maximum number of items that you want.
- DynamoDB paginates the results from Query operations, where Query results are divided into "pages" of data that are 1 MB in size (or less).
- **ScannedCount** is the number of items that matched the key condition expression, before a filter expression (if present) was applied.
- **Count** is the number of items that remain, after a filter expression (if present) was applied.

DynamoDB Scans

- A *Scan* operation reads every item in a table or a secondary index. By default, a Scan operation returns all of the data attributes for every item in the table or index.



- Scan always returns a result set. If no matching items are found, the result set will be empty.
- A single Scan request can retrieve a maximum of 1 MB of data.
- You can optionally provide a filter expression.
- You can limit the number of items that is returned in the result.
- DynamoDB paginates the results from Scan operations.
- ScannedCount is the number of items evaluated, before any ScanFilter is applied.
- Count is the number of items that remain, after a filter expression (if present) was applied.
- A Scan operation performs eventually consistent reads, by default.
- By default, the Scan operation processes data sequentially.

On-Demand Backup and Restore

- You can use IAM to restrict DynamoDB backup and restore actions for some resources.
- All backup and restore actions are captured and recorded in AWS CloudTrail.
- Backups
    - Each time you create an on-demand backup, the entire table data is backed up.

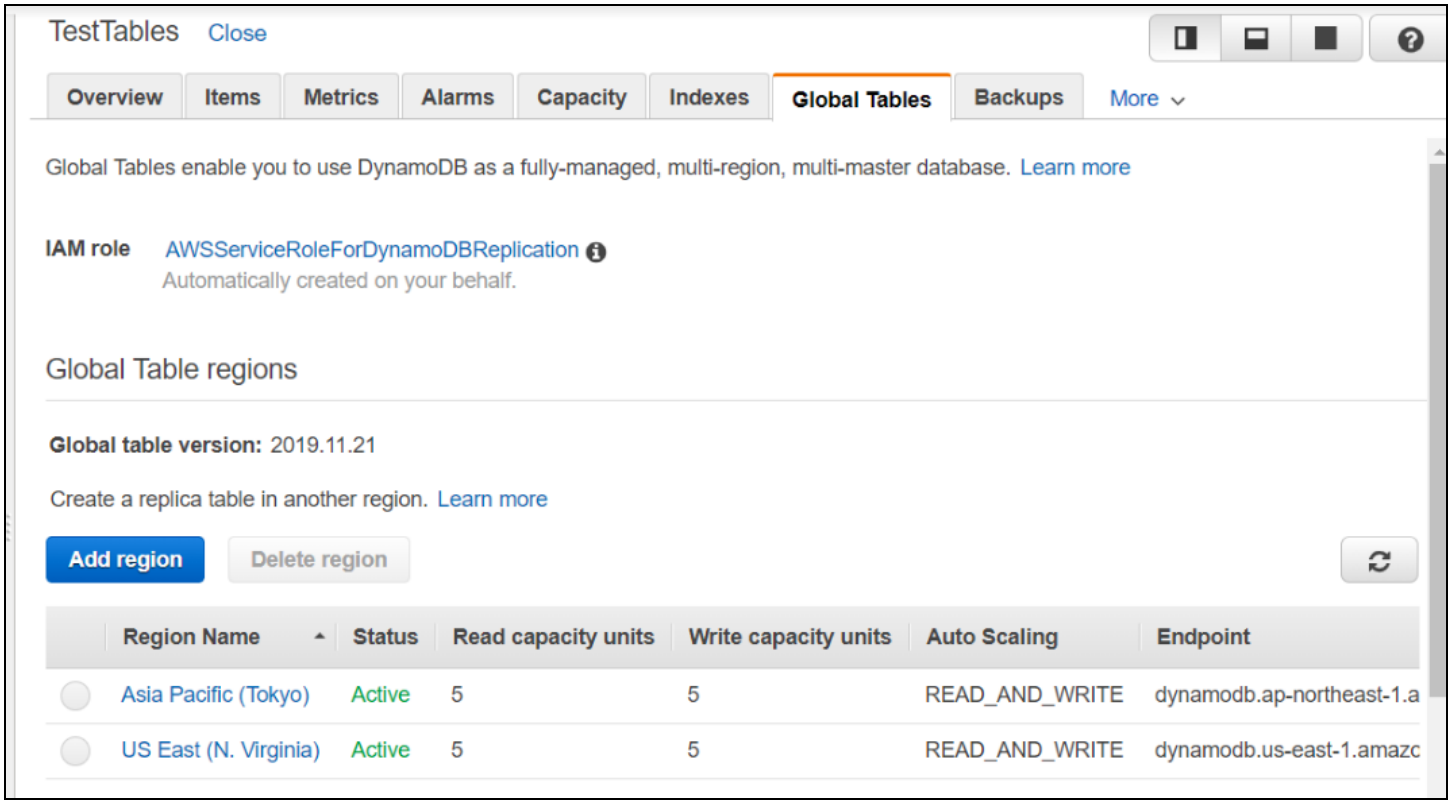
    - All backups and restores in DynamoDB work without consuming any provisioned throughput on the table.
    - DynamoDB backups do not guarantee causal consistency across items; however, the skew between updates in a backup is usually much less than a second.
    - You can restore backups as new DynamoDB tables in other regions.
    - Included in the backup are:
        - Database data
        - Global secondary indexes
        - Local secondary indexes
        - Streams
        - Provisioned read and write capacity
    - While a backup is in progress, you can't do the following:
        - Pause or cancel the backup operation.
        - Delete the source table of the backup.
        - Disable backups on a table if a backup for that table is in progress.
- Restore
    - You cannot overwrite an existing table during a restore operation.
    - You restore backups to a new table.
    - For tables with even data distribution across your primary keys, the restore time is proportional to the largest single partition by item count and not the overall table size.
    - If your source table contains data with significant skew, the time to restore may increase.

DynamoDB Transactions

- Amazon DynamoDB transactions simplify the developer experience of making coordinated, all-or-nothing changes to multiple items both within and across tables.
- Transactions provide atomicity, consistency, isolation, and durability (ACID) in DynamoDB, helping you to maintain data correctness in your applications.
- You can group multiple Put, Update, Delete, and ConditionCheck actions. You can then submit the actions as a single TransactWriteItems operation that either succeeds or fails as a unit.
- You can group and submit multiple Get actions as a single TransactGetItems operation.
- Amazon DynamoDB supports up to 25 unique items and 4 MB of data per transactional request.

Global Tables

- Global tables provide a solution for deploying a multi-region, multi-master database, without having to build and maintain your own replication solution.
- You specify the AWS regions where you want the table to be available. DynamoDB performs all tasks to create identical tables in these regions, and propagate ongoing data changes to all of them.



- Replica Table (Replica, for short)
  - A single DynamoDB table that functions as a part of a global table.
  - Each replica stores the same set of data items.
  - Any given global table can only have one replica table per region.
  - You can add new or delete replicas from global tables.
- To ensure eventual consistency, DynamoDB global tables use a "*last writer wins*" reconciliation between concurrent updates, where DynamoDB makes a best effort to determine the last writer.
- If a single AWS region becomes isolated or degraded, your application can redirect to a different region and perform reads and writes against a different replica table. DynamoDB also keeps track of any writes that have been performed, but have not yet been propagated to all of the replica tables.
- Requirements for adding a new replica table
  - The table must have the same partition key as all of the other replicas.
  - The table must have the same write capacity management settings specified.
  - The table must have the same name as all of the other replicas.
  - The table must have DynamoDB Streams enabled, with the stream containing both the new and the old images of the item.
  - None of the replica tables in the global table can contain any data.
- If global secondary indexes are specified, then the following conditions must also be met:
  - The global secondary indexes must have the same name.

- The global secondary indexes must have the same partition key and sort key (if present).

Security

- Encryption
  - Encrypts your data at rest using an AWS Key Management Service (AWS KMS) managed encryption key for DynamoDB.
  - Encryption at rest can be enabled only when you are creating a new DynamoDB table.
  - After encryption at rest is enabled, it can't be disabled.
  - Uses AES-256 encryption.
  - The following are encrypted:
    - DynamoDB base tables
    - Local secondary indexes
    - Global secondary indexes
  - Authentication and Access Control
    - Access to DynamoDB requires credentials.
    - Aside from valid credentials, you also need to have permissions to create or access DynamoDB resources.
    - Types of Identities
      - **AWS account root user**
      - **IAM user**
      - **IAM role**
  - You can create indexes and streams only in the context of an existing DynamoDB table, referred to as *subresources*.
  - Resources and subresources have unique Amazon Resource Names (**ARNs**) associated with them.
  - A *permissions policy* describes who has access to what.
    - Identity-based Policies
      - Attach a permissions policy to a user or a group in your account
      - Attach a permissions policy to a role (grant cross-account permissions)
    - Policy Elements
      - Resource - use an ARN to identify the resource that the policy applies to.
      - Action - use action keywords to identify resource operations that you want to allow or deny.
      - Effect - specify the effect, either allow or deny, when the user requests the specific action.
      - Principal - the user that the policy is attached to is the implicit principal.
  - Web Identity Federation - Customers can sign in to an identity provider and then obtain temporary security credentials from AWS Security Token Service (AWS STS).

Monitoring

- Automated tools:
  - **Amazon CloudWatch Alarms** – Watch a single metric over a time period that you specify, and perform one or more actions based on the value of the metric relative to a given threshold over a number of time periods.
  - **Amazon CloudWatch Logs** – Monitor, store, and access your log files from AWS CloudTrail or other sources.
  - **Amazon CloudWatch Events** – Match events and route them to one or more target functions or streams to make changes, capture state information, and take corrective action.
  - **AWS CloudTrail Log Monitoring** – Share log files between accounts, monitor CloudTrail log files in real time by sending them to CloudWatch Logs, write log processing applications in Java, and validate that your log files have not changed after delivery by CloudTrail.
- Using the information collected by CloudTrail, you can determine the request that was made to DynamoDB, the IP address from which the request was made, who made the request, when it was made, and additional details.

DynamoDB Accelerator (DAX)

- DAX is a fully managed, highly available, in-memory cache for DynamoDB.
- **DynamoDB Accelerator (DAX)** delivers microsecond response times for accessing eventually consistent data.
- It requires only minimal functional changes to use DAX with an existing application since it is API-compatible with DynamoDB.
- For read-heavy or bursty workloads, DAX provides increased throughput and potential cost savings by reducing the need to overprovision read capacity units.
- DAX lets you scale on-demand.
- DAX is fully managed. You no longer need to do hardware or software provisioning, setup and configuration, software patching, operating a reliable, distributed cache cluster, or replicating data over multiple instances as you scale.
- DAX is not recommended if you need strongly consistent reads
- DAX is useful for read-intensive workloads, but not write-intensive ones.
- DAX supports server-side encryption but not TLS.
- Use Cases
  - Applications that require the fastest possible response time for reads.
  - Applications that read a small number of items more frequently than others. For example, limited-time on-sale items in an ecommerce store.
  - Applications that are read-intensive, but are also cost-sensitive. Offload read activity to a DAX cluster and reduce the number of read capacity units that you need to purchase for your DynamoDB tables.

    - Applications that require repeated reads against a large set of data. This will avoid eating up all your DynamoDB resources which are needed by other applications.
- To achieve high availability for your application, provision your DAX cluster with at least three nodes, then place the nodes in multiple Availability Zones within a Region.
- There are two options available for scaling a DAX cluster:
    - **Horizontal scaling**, where you add read replicas to the cluster. A single DAX cluster supports up to 10 read replicas, and you can add or remove replicas while the cluster is running.
    - **Vertical scaling**, where you select different node types. Larger nodes enable the cluster to store more data in memory, reducing cache misses and improving overall application performance. You can't modify the node types on a running DAX cluster. Instead, you must create a new cluster with the desired node type.

Best Practices

- Know the Differences Between Relational Data Design and NoSQL

Relational database systems (RDBMS)	NoSQL database
In RDBMS, data can be queried flexibly, but queries are relatively expensive and don't scale well in high-traffic situations.	In a NoSQL database such as DynamoDB, data can be queried efficiently in a limited number of ways, outside of which queries can be expensive and slow.
In RDBMS, you design for flexibility without worrying about implementation details or performance. Query optimization generally doesn't affect schema design, but normalization is very important.	In DynamoDB, you design your schema specifically to make the most common and important queries as fast and as inexpensive as possible. Your data structures are tailored to the specific requirements of your business use cases.
For an RDBMS, you can go ahead and create a normalized data model without thinking about access patterns. You can then extend it later when new questions and query requirements arise. You can organize each type of data into its own table.	For DynamoDB, by contrast, you shouldn't start designing your schema until you know the questions it will need to answer. Understanding the business problems and the application use cases up front is essential. You should maintain as few tables as possible in a DynamoDB application. Most well designed applications require **only one** table.

	It is important to understand three fundamental properties of your application's access patterns: 1. Data size: Knowing how much data will be stored and requested at one time will help determine the most effective way to partition the data. 2. Data shape: Instead of reshaping data when a query is processed, a NoSQL database organizes data so that its shape in the database corresponds with what will be queried. 3. Data velocity: DynamoDB scales by increasing the number of physical partitions that are available to process queries, and by efficiently distributing data across those partitions. Knowing in advance what the peak query loads might be helps determine how to partition data to best use I/O capacity.

- Design and Use Partition Keys Effectively
    - DynamoDB provides some flexibility in your per-partition throughput provisioning by providing **burst capacity**.
    - To better accommodate uneven access patterns, **DynamoDB adaptive capacity** enables your application to continue reading and writing to 'hot' partitions without being throttled, by automatically increasing throughput capacity for partitions that receive more traffic.
    - Amazon DynamoDB now applies adaptive capacity in real time in response to changing application traffic patterns, which helps you maintain uninterrupted performance indefinitely, even for imbalanced workloads. In addition, instant adaptive capacity helps you provision read and write throughput more efficiently instead of overprovisioning to accommodate uneven data access patterns. Instant adaptive capacity is on by default at no additional cost for all DynamoDB tables and global secondary indexes.
    - The optimal usage of a table's provisioned throughput depends not only on the workload patterns of individual items, but also on the partition-key design. In general, you will use your provisioned throughput more efficiently as the ratio of partition key values accessed to the total number of partition key values increases.
    - Structure the primary key elements to avoid one heavily requested partition key value that slows overall performance.
    - Distribute loads more evenly across a partition key space by adding a random number to the end of the partition key values. Then you randomize the writes across the larger space.
    - A randomizing strategy can greatly improve write throughput, but it's difficult to read a specific item because you don't know which suffix value was used when writing the item. Instead of

using a random number to distribute the items among partitions, use a number that you can calculate based upon something that you want to query on.
    - Distribute write activity efficiently during data upload by using the sort key to load items from each partition key value, keeping more DynamoDB servers busy simultaneously and improving your throughput performance.
- Use Sort Keys to Organize Data
    - Well-designed sort keys gather related information together in one place where it can be queried efficiently.
    - Composite sort keys let you define hierarchical (one-to-many) relationships in your data that you can query at any level of the hierarchy.
- Use indexes efficiently by keeping the number of indexes to a minimum and avoid indexing tables that experience heavy write activity.
- Choose Projections Carefully.
- Optimize Frequent Queries to Avoid Fetches.
- Be Aware of Item-Collection Size Limits When Creating Local Secondary Indexes.
- For Querying and Scanning Data
    - Performance considerations for scans
    - Avoiding sudden spikes in read activity
    - Taking advantage of parallel scans

Pricing

- DynamoDB charges per GB of disk space that your table consumes. The first 25 GB consumed per month is free.
- DynamoDB charges for Provisioned Throughput ---- WCU and RCU, Reserved Capacity and Data Transfer Out.
- You should round up to the nearest KB when estimating how many capacity units to provision.
- There are additional charges for DAX, Global Tables, On-demand Backups (per GB), Continuous backups and point-in-time recovery (per GB), Table Restorations (per GB), and Streams (read request units).

Sources:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html?shortFooter=true
https://aws.amazon.com/dynamodb/faqs/

DynamoDB Scan vs Query

Scan

The *Scan* operation returns one or more items and item attributes by accessing every item in a table or a secondary index. The total number of scanned items has a maximum size limit of 1 MB.

Scan operations proceed sequentially; however, for faster performance on a large table or secondary index, applications can request a parallel Scan operation.

Scan uses eventually consistent reads when accessing the data in a table; therefore, the result set might not include the changes to data in the table immediately before the operation began. If you need a consistent copy of the data, as of the time that the *Scan* begins, you can set the ConsistentRead parameter to true when you submit a scan request.

In general, Scan operations are less efficient than other operations in DynamoDB. If possible, avoid using a Scan operation on a large table or index with a filter that removes many results. Using Scan over large data sets may use up the provisioned throughput for a large table or index in a single operation.

Instead of using a large Scan operation, you can apply the following techniques to minimize the impact of a scan on a table's provisioned throughput:

1. **Reduce page size** - because a Scan operation reads an entire page (by default, 1 MB), you can reduce the impact of the scan operation by setting a smaller page size.
2. **Isolate scan operations** - perform scans on a table that is not taking "mission-critical" traffic. You can configure applications to handle this load by rotating traffic periodically between two tables, whose data is replicated with one another.

Query

The *Query* operation finds items based on primary key values. You can query any table or secondary index that has a composite primary key (a partition key and a sort key). A *Query* operation will return all of the items from the table or index with the partition key value you provided.

A *Query* operation always returns a result set. If no matching items are found, the result set will be empty. Query results are always sorted by the sort key value. If the data type of the sort key is Number, the results are returned in numeric order; otherwise, the results are returned in order of UTF-8 bytes.

A single *Query* operation can retrieve items up to a maximum data size of 1MB. You can query a table, a local secondary index, or a global secondary index. For a query on a table or on a local secondary index, you can set the ConsistentRead parameter to true and obtain a strongly consistent result. Global secondary indexes

support eventually consistent reads only, so do not specify ConsistentRead when querying a global secondary index.

For faster response times, design your tables and indexes so that your applications can use Query instead of Scan.

Parallel Scans

With a parallel scan, your application has multiple workers that are all running Scan operations concurrently. Using parallel scan can sometimes provide more benefits to your applications compared to sequential scan. Since DynamoDB stores your data across multiple physical storage partitions for rapid access, you are not constrained by the maximum throughput of a single partition. Although, this can quickly consume all of your table's provisioned read capacity. In that case, other applications that need to access the table might be throttled.

A parallel scan can be the right choice if the following conditions are met:

- The table size is 20 GB or larger.
- The table's provisioned read throughput is not being fully used.
- Sequential Scan operations are too slow.

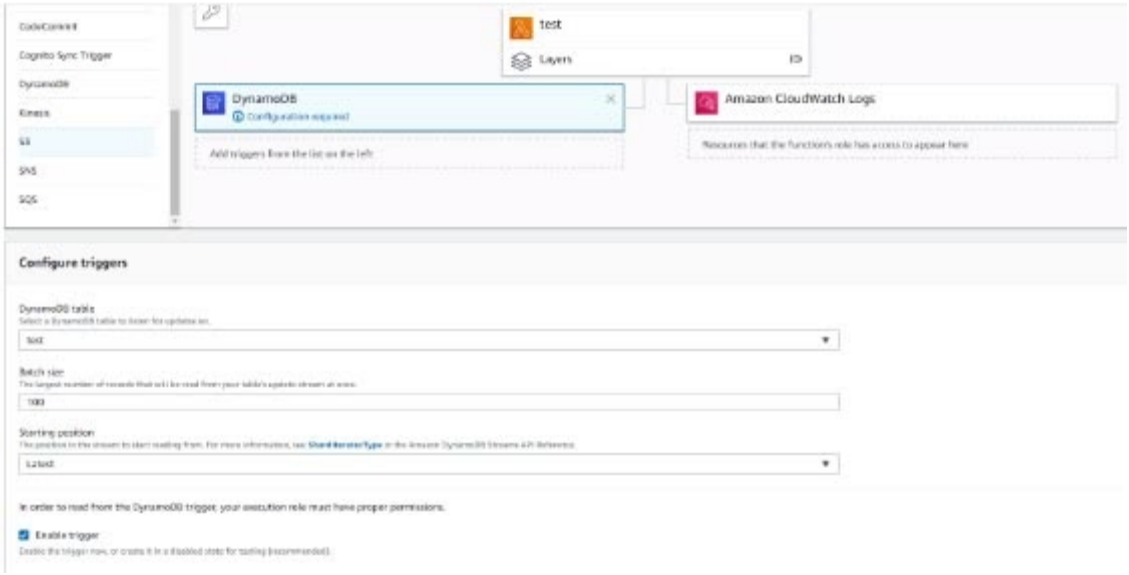
Monitor your parallel scans to optimize your provisioned throughput use, while also making sure that your other applications aren't starved of resources. In which case, DynamoDB's Scan function accepts the following additional parameters:

- **TotalSegments** denotes the number of workers that will access the table concurrently.
- **Segment** denotes the segment of table to be accessed by the calling worker.

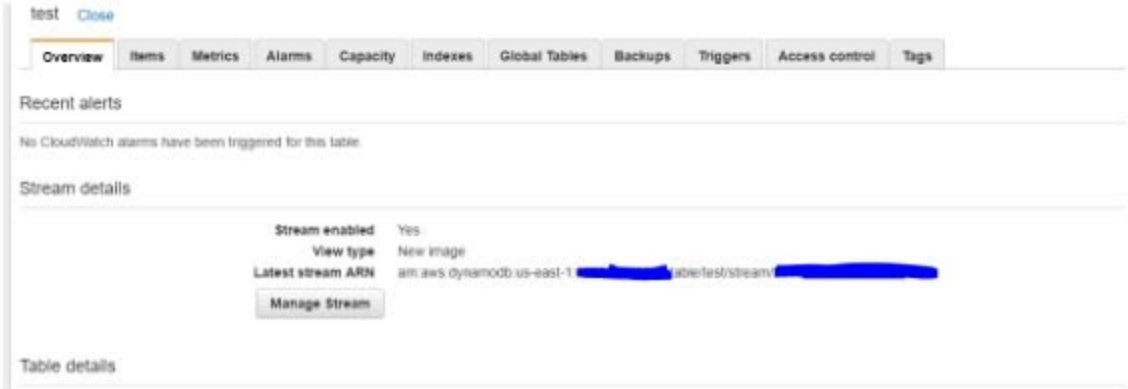
Sources:
https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_Scan.html
https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_Query.html
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-query-scan.html

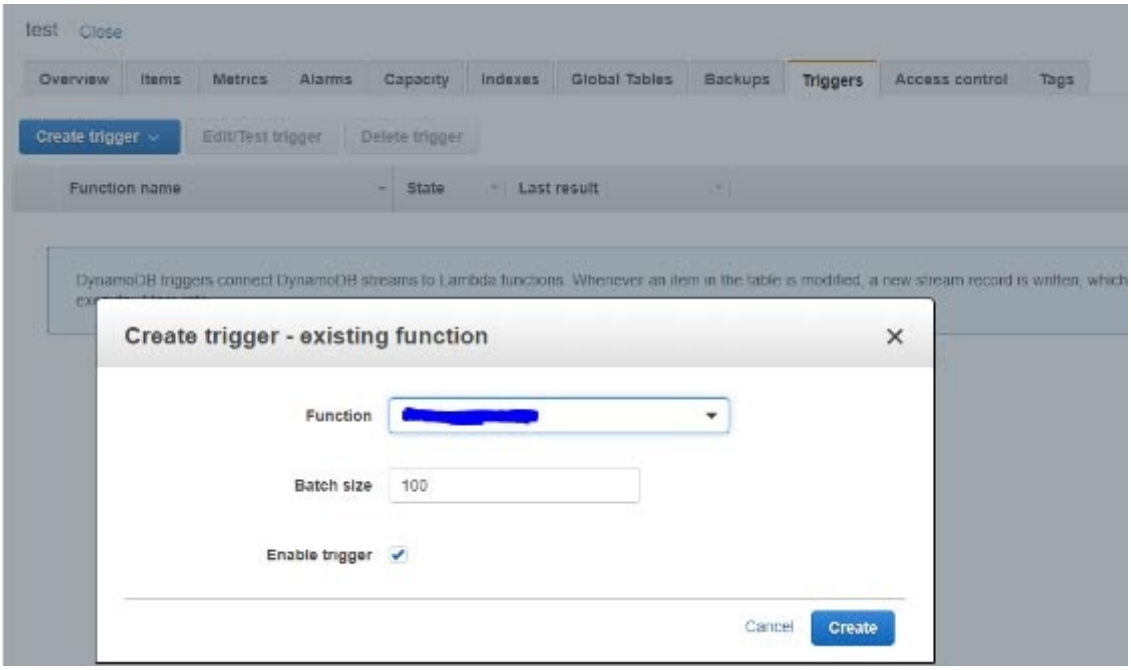
AWS Lambda Integration with Amazon DynamoDB Streams

Amazon DynamoDB is integrated with AWS Lambda so that you can create *triggers*, which are pieces of code that automatically respond to events in DynamoDB Streams. With triggers, you can build applications that react to data modifications in DynamoDB tables.



- After you enable DynamoDB Streams on a table, associate the DynamoDB table with a Lambda function. AWS Lambda polls the stream and invokes your Lambda function synchronously when it detects new stream records.





- Configure the StreamSpecification you want for your DynamoDB Streams:
  - StreamEnabled (Boolean) - indicates whether DynamoDB Streams is enabled (true) or disabled (false) on the table.
  - StreamViewType (string) - when an item in the table is modified, StreamViewType determines what information is written to the stream for this table. Valid values for StreamViewType are:
    - KEYS_ONLY - Only the key attributes of the modified items are written to the stream.
    - NEW_IMAGE - The entire item, as it appears after it was modified, is written to the stream.
    - OLD_IMAGE - The entire item, as it appeared before it was modified, is written to the stream.
    - NEW_AND_OLD_IMAGES - Both the new and the old item images of the items are written to the stream.

Stream View Types	
KEYS_ONLY	Only the key attributes of the modified items are written to the stream
NEW_IMAGE	The entire item, as it appears after it was modified, is written to the stream.
OLD_IMAGE	The entire item, as it appeared before it was modified, is written to the stream.
NEW_AND_OLD_IMAGES	Both the new and the old item images of the items are written to the stream.

Tutorials Dojo
Tutorials Dojo

Sources:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.Lambda.html
https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_StreamSpecification.html

Calculating the Required Read and Write Capacity Unit for your DynamoDB Table

Read Capacity Unit

On-Demand Mode

When you choose on-demand mode, DynamoDB instantly accommodates your workloads as they ramp up or down to any previously reached traffic level. If a workload's traffic level hits a new peak, DynamoDB adapts rapidly to accommodate the workload. The request rate is only limited by the DynamoDB throughput default table limits, but it can be raised upon request.

For on-demand mode tables, you don't need to specify how much read throughput you expect your application to perform. DynamoDB charges you for the reads that your application performs on your tables in terms of read request units.

- 1 read request unit (RRU) = 1 strongly consistent read of up to 4 KB/s = 2 eventually consistent reads of up to 4 KB/s per read.
- 2 RRUs = 1 transactional read request (one read per second) for items up to 4 KB.
- For reads on items greater than 4 KB, total number of reads required = (total item size / 4 KB) rounded up.

Provisioned Mode

If you choose provisioned mode, you specify the number of reads and writes per second that you require for your application. You can use auto scaling to adjust your table's provisioned capacity automatically in response to traffic changes.

For provisioned mode tables, you specify throughput capacity in terms of read capacity units (RCUs).

- 1 read capacity unit (RCU) = 1 strongly consistent read of up to 4 KB/s = 2 eventually consistent reads of up to 4 KB/s per read.
- 2 eventually consistent reads of up to 4 KB/s per read is also equivalent to 1 eventually consistent read of up to 8 KB/s per read.
- To get the *RCU per Item*, you have to divide the average item size by 4 KB (for strong consistency) or 8 KB (for eventual consistency).
- 2 RCUs = 1 transactional read request (one read per second) for items up to 4 KB.
- For reads on items greater than 4 KB, total number of reads required = (total item size / 4 KB) rounded up.

CDA Exam Notes:

Example: You have items stored in your DynamoDB table with an average item size of **2** KB. Your application will be sending 20 eventually consistent read requests per second. To get the RCU, just do these 3 steps:

Step #1 - Get the Average Item Size by rounding up to 4 KB

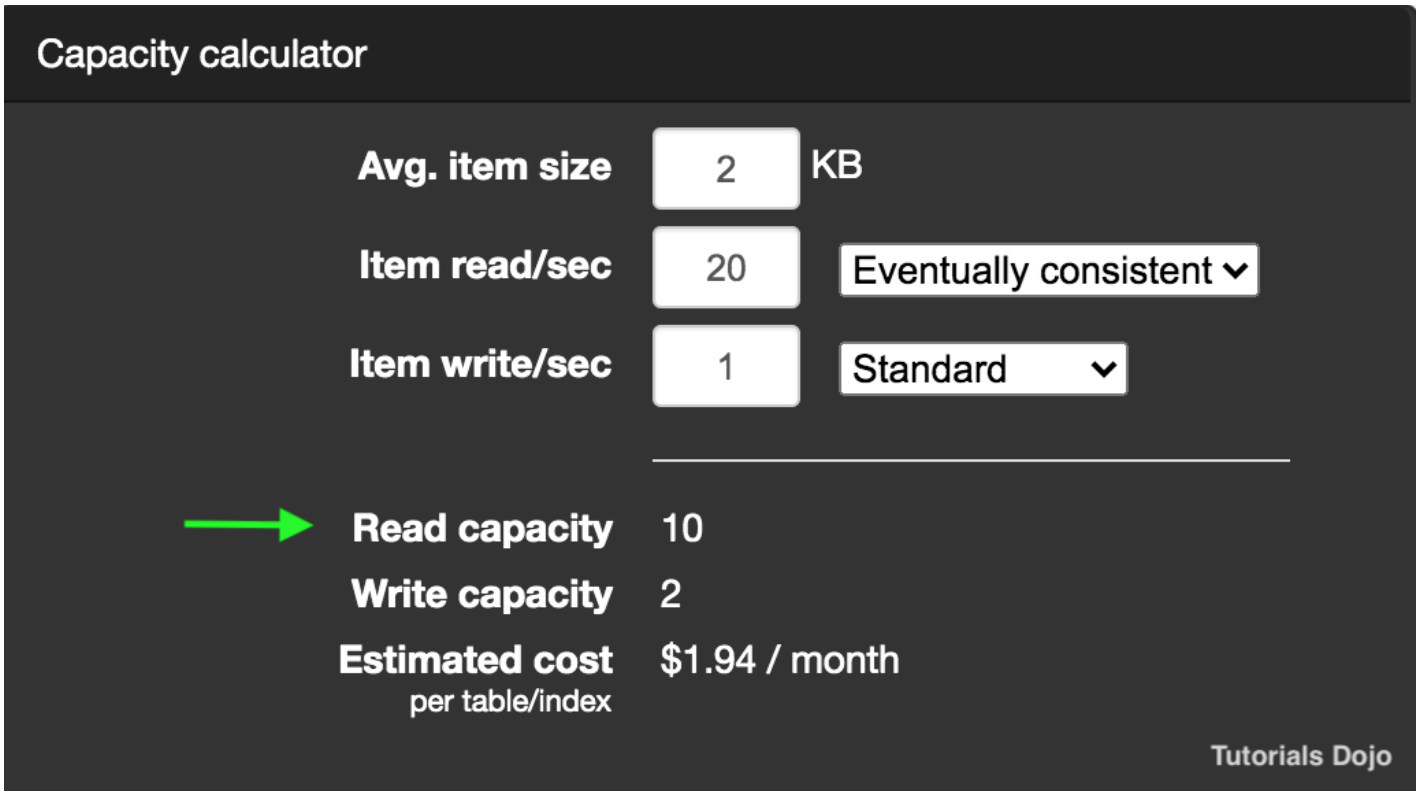
Average Item Size = 2 KB = **4 KB** (rounded up)

Step #2 - Get the *RCU per Item* by dividing the Average Item Size by 8 KB *(for eventual consistency)*

= 4 KB / 8 KB
= **0.5** RCU per Item

Step #3 - **Multiply the RCU per item to the number of items to be written per second**

= **0.5** RCU per item × **20** reads per second
= **10** RCU



Write Capacity Unit

On-Demand Mode

For on-demand mode tables, you don't need to specify how much write throughput you expect your application to perform. DynamoDB charges you for the writes that your application performs on your tables in terms of write request units.

- 1 write request unit (WRU) = 1 write of up to 1 KB/s.
- 2 WRUs = 1 transactional write request (one write per second) for items up to 1 KB.
- For writes greater than 1 KB, total number of writes required = (total item size / 1 KB) rounded up

Provisioned Mode

For provisioned mode tables, you specify throughput capacity in terms of write capacity units (WCUs).

- 1 write capacity unit (WCU) = 1 write of up to 1 KB/s.
- 2 WCUs = 1 transactional write request (one write per second) for items up to 1 KB.
- For writes greater than 1 KB, total number of writes required = (total item size / 1 KB) rounded up

Examples

1. For 5 KB average item size:
    1. 1 read capacity unit needed to perform 1 eventually consistent read per second
    2. 2 read capacity units needed to perform 1 strongly consistent read per second
    3. 4 read capacity units needed to perform 1 transactional read per second
    4. 5 write capacity units needed to perform 1 standard write per second
    5. 10 write capacity units needed to perform 1 transactional write per second
2. For 9 KB average item size:
    1. 2 read capacity unit needed to perform 1 eventually consistent read per second
    2. 3 read capacity units needed to perform 1 strongly consistent read per second
    3. 6 read capacity units needed to perform 1 transactional read per second
    4. 9 write capacity units needed to perform 1 standard write per second
    5. 18 write capacity units needed to perform 1 transactional write per second
3. For 13 KB average item size:
    1. 2 read capacity unit needed to perform 1 eventually consistent read per second
    2. 4 read capacity units needed to perform 1 strongly consistent read per second
    3. 8 read capacity units needed to perform 1 transactional read per second
    4. 13 write capacity units needed to perform 1 standard write per second
    5. 26 write capacity units needed to perform 1 transactional write per second

CDA Exam Notes:

Example: An application is writing **75** items to a DynamoDB table every second, where each item is **11.5** KB in size. To get the WCU, you just have to follow these 3 simple steps below:

Step #1 Get the Average Item Size

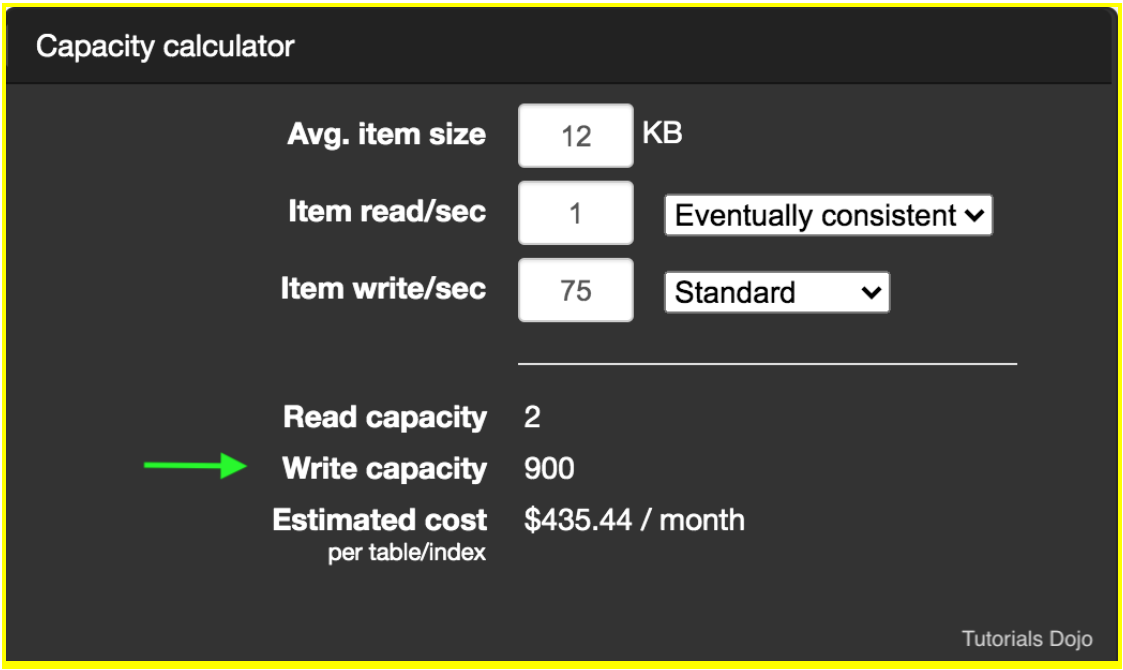
= 11.5 KB = **12 KB** (Rounded up)

Step #2 Get the *WCU per Item* by dividing the Average Item Size by 1 KB

Divide the Average Item Size by 1KB and round up the result:
= 12 KB / 1 KB
= **12 WCU per Item**

Step #3 Multiply the WCU per item to the number of items to be written per second

= **12** WCU per item × **75** writes per second
= **900** WCU



Source:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ReadWriteCapacityMode.html

Amazon Elasticache

- ElastiCache is a distributed **in-memory cache** environment in the AWS Cloud.
- ElastiCache works with both the **Redis** and **Memcached** engines.
- Components
    - ElastiCache Nodes
        - A **node** is a fixed-size chunk of secure, network-attached RAM. A node can exist in isolation from or in some relationship to other nodes.
        - Every node within a cluster is the same instance type and runs the same cache engine. Each cache node has its own Domain Name Service (DNS) name and port.
- If a maintenance event is scheduled for a given week, it will be initiated and completed at some point during the 60 minute maintenance window you specify.
- Elasticache can be used for storing session state.
- ElastiCache Redis
    - Existing applications that use Redis can use ElastiCache with almost no modification.
    - Features
        - Automatic detection and recovery from cache node failures.
        - Multi-AZ with automatic failover of a failed primary cluster to a read replica in Redis clusters that support replication.
        - Redis (cluster mode enabled) supports partitioning your data across up to 250 shards.
        - Redis supports in-transit and at-rest encryption with authentication so you can build HIPAA-compliant applications.
        - Flexible Availability Zone placement of nodes and clusters for increased fault tolerance.
        - Data is persistent.
        - Can be used as a datastore.
        - Not multi-threaded.
        - Amazon ElastiCache for Redis supports self-service updates, which allows you to apply service updates at the time of your choosing and track the progress in real-time.
    - Cache data if:
        - It is slow or expensive to acquire when compared to cache retrieval.
        - It is accessed with sufficient frequency.
        - It is relatively static, or if rapidly changing, staleness is not a significant issue.
    - **Redis sorted sets** guarantee both uniqueness and element ordering. Each time a new element is added to the sorted set it's reranked in real time. It's then added to the set in its appropriate numeric position.
    - In the **Redis publish/subscribe** paradigm, you send a message to a specific channel not knowing who, if anyone, receives it. Recipients of the message are those who are subscribed to the channel.
    - **Redis hashes** are hashes that map string names to string values.
    - Components

        - **Redis Shard** - a grouping of one to six related nodes. A Redis (cluster mode disabled) cluster always has one shard. A Redis (cluster mode enabled) cluster can have 1–90 shards.
            - A *multiple node shard* implements replication by have one read/write primary node and 1–5 replica nodes.
            - If there is more than one node in a shard, the shard supports replication with one node being the read/write primary node and the others read-only replica nodes.
        - **Redis Cluster** - a logical grouping of one or more ElastiCache for Redis Shards. Data is partitioned across the shards in a Redis (cluster mode enabled) cluster.
    - For improved fault tolerance, have at least two nodes in a Redis cluster and enabling **Multi-AZ with automatic failover**.
    - Replica nodes use asynchronous replication mechanisms to keep synchronized with the primary node.
    - If any primary has no replicas and the primary fails, you lose all that primary's data.
    - You can use backup and restore to **migrate** to Redis (cluster mode enabled) and resize your Redis (cluster mode enabled).
    - Redis (cluster mode disabled) vs Redis (cluster mode enabled)

	Redis (cluster mode disabled)	Redis (cluster mode enabled)
Shards (node groups)	1	1-90
Replicas for each shard (node group)	0-5	0-5
Data partitioning	No	Yes
Add/Delete replicas	Yes	Yes
Add/Delete node groups	No	No
Supports scale up	Yes	No
Supports engine upgrades	Yes	Yes
Promote replica to primary	Yes	No
Multi-AZ with automatic failover	Yes, with at least 1 replica. Optional. On by default.	Required
Backup/Restore	Yes	Yes

    - You can vertically scale up or scale down your sharded Redis Cluster on demand. Amazon ElastiCache resizes your cluster by changing the node type, while the cluster continues to stay online and serve incoming requests.
    - You can set up automatic snapshots or initiate manual backups, and then seed new ElastiCache for Redis clusters. You can also export your snapshots to an S3 bucket of your choice for disaster recovery, analysis or cross-region backup and restore.
    - Endpoints
        - **Single Node Redis** (**cluster mode disabled**) Endpoints - used to connect to the cluster for both reads and writes.
        - **Multi-Node Redis** (**cluster mode disabled**) Endpoints - use the primary endpoint for all writes to the cluster. The read endpoint points to your read replicas.
        - **Redis** (**cluster mode enabled**) Endpoints - has a single configuration endpoint. By connecting to the configuration endpoint, your application is able to discover the primary and read endpoints for each shard in the cluster.
    - Parameter Groups
        - **Cache parameter group** is a named collection of engine-specific parameters that you can apply to a cluster.
        - Parameters are used to control memory usage, eviction policies, item sizes, and more.
    - Redis Security
        - ElastiCache for Redis node access is restricted to applications running on whitelisted EC2 instances. You can control access of your cluster by using subnet groups or security groups. By default, network access to your clusters is turned off.
        - By default, all new ElastiCache for Redis clusters are launched in a VPC environment. Use subnet groups to grant cluster access from Amazon EC2 instances running on specific subnets.
        - ElastiCache for Redis supports TLS and in-place encryption for nodes running specified versions of the ElastiCache for Redis engine.
        - You can use your own customer managed customer master keys (CMKs) in AWS Key Management Service to encrypt data at rest in ElastiCache for Redis.
    - Redis Backups
        - A point-in-time copy of a Redis cluster.
        - Backups consist of all the data in a cluster plus some metadata.
    - Global Datastore
        - A new feature that provides fully managed, secure cross-region replication. You can now write to your ElastiCache for Redis cluster in one region and have the data available for reading in two other cross-region replica clusters.
        - In the unlikely event of regional degradation, one of the healthy cross-region replica clusters can be promoted to become the primary cluster with full read/write capabilities.
- ElastiCache Memcached
    - Features
        - Automatic detection and recovery from cache node failures.

        - Automatic discovery of nodes within a cluster enabled for automatic discovery, so that no changes need to be made to your application when you add or remove nodes.
        - Flexible Availability Zone placement of nodes and clusters.
        - **ElastiCache Auto Discovery** feature for Memcached lets your applications identify all of the nodes in a cache cluster and connect to them.
        - ElastiCache node access is restricted to applications running on whitelisted EC2 instances. You can control the instances that can access your cluster by using subnet groups or security groups.
        - It is not persistent.
        - Supports large nodes with multiple cores or threads.
        - Does not support multi-AZ failover or replication
        - Does not support snapshots
    - Components
        - **Memcached cluster** - a logical grouping of one or more ElastiCache Nodes. Data is partitioned across the nodes in a Memcached cluster.
            - Memcached supports up to 100 nodes per customer for each Region with each cluster having 1–20 nodes.
            - When you partition your data, use *consistent hashing*.
        - **Endpoint** - the unique address your application uses to connect to an ElastiCache node or cluster.
            - Each node in a Memcached cluster has its own endpoint.
            - The cluster also has an endpoint called the *configuration endpoint*.
        - **ElastiCache parameter group** - a named collection of engine-specific parameters that you can apply to a cluster. Parameters are used to control memory usage, eviction policies, item sizes, and more.
        - ElastiCache allows you to control access to your clusters using **security groups**. By default, network access to your clusters is turned off.
        - A **subnet group** is a collection of subnets that you can designate for your clusters running in a VPC environment. If you create a cluster in a VPC, then you must specify a *cache subnet group*. ElastiCache uses that cache subnet group to choose a subnet and IP addresses within that subnet to associate with your cache nodes.
    - Mitigating Failures

        - Node Failures
            - Spread your cached data over more nodes. Because Memcached does not support replication, a node failure will always result in some data loss from your cluster.
        - Availability Zone Failure
            - Locate your nodes in as many Availability Zones as possible. In the unlikely event of an AZ failure, you will lose the data cached in that AZ, not the data cached in the other AZs.

    - ElastiCache uses DNS entries to allow client applications to locate servers (nodes). The DNS name for a node remains constant, but the IP address of a node can change over time.
-
- Caching Strategies

    - **Lazy Loading** - a caching strategy that loads data into the cache only when necessary.
        - Only requested data is cached.
        - Node failures are not fatal.
        - There is a cache miss penalty.
        - Stale data.
    - **Write Through** - adds data or updates data in the cache whenever data is written to the database.
        - Data in the cache is never stale.
        - Write penalty vs. Read penalty. Every write involves two trips: A write to the cache and a write to the database.
        - Missing data.
        - Cache churn.
    - By adding a time to live (TTL) value to each write, we are able to enjoy the advantages of each strategy and largely avoid cluttering up the cache with superfluous data.
- Scaling ElastiCache for Memcached Clusters
    - Scaling Memcached Horizontally
        - The Memcached engine supports partitioning your data across multiple nodes. Because of this, Memcached clusters scale horizontally easily. A Memcached cluster can have 1 to 20 nodes. To horizontally scale your Memcached cluster, just add or remove nodes.
    - Scaling Memcached Vertically
        - When you scale your Memcached cluster up or down, you must create a new cluster. Memcached clusters always start out empty unless your application populates it.
- Monitoring
    - The service continuously monitors the health of your instances. In case a node experiences failure or a prolonged degradation in performance, ElastiCache will automatically restart the node and associated processes.
    - ElastiCache provides both host-level metrics and metrics that are specific to the cache engine software. These metrics are measured and published for each Cache node in 60-second intervals.
    - Monitor events with **ElastiCache Events**. When significant events happen on a cache cluster, including failure to add a node, success in adding a node, the modification of a security group, and others, ElastiCache sends a notification to a specific SNS topic.
    - Monitor costs with tags.
-
- Redis VS Memcached

- Memcached is designed for **simplicity** while Redis offers a **rich set of features** that make it effective for a wide range of use cases.

	Redis (cluster mode enabled)	Redis (cluster mode disabled)	Memcached
Data Types	string, sets, sorted sets, lists, hashes, bitmaps, hyperloglog, geospatial indexes	string, sets, sorted sets, lists, hashes, bitmaps, hyperloglog, geospatial indexes	string, objects (like databases)
Data Partitioning (distribute your data among multiple nodes)	Supported	Unsupported	Supported
Modifiable cluster	Only versions 3.2.10 and later	Yes	Yes
Online resharding	Only versions 3.2.10 and later	No	No
Encryption	3.2.6, 4.0.10 and later	3.2.6, 4.0.10 and later	Unsupported
Sub-millisecond latency	Yes	Yes	Yes
FedRAMP, PCI DSS and HIPAA compliant	3.2.6, 4.0.10 and later	3.2.6, 4.0.10 and later	No
Multi-threaded (make use of multiple processing cores	No	No	Yes
Node type upgrading	No	Yes	No
Engine upgrading	Yes		
Cluster replication (create multiple copies of a primary cluster)	Supported	Supported	Unsupported
Multi-AZ for automatic failover	Required	Optional	Unsupported
Transactions (execute a group of commands as an isolated and atomic operation)	Supported	Supported	Unsupported
Pub/Sub capability	Yes	Yes	No
Backup and restore (keep your data on disk with a point in time snapshot)	Supported	Supported	Unsupported
Lua Scripting (execute transactional Lua scripts)	Supported	Supported	Unsupported
Use Case	• You need to partition your data across two to 90 node groups (clustered mode only). • You need geospatial indexing (clustered mode or non-clustered mode). • You don't need to support multiple databases. • Plus features of non-clustered mode	• You need complex data types, such as strings, hashes, lists, sets, sorted sets, and bitmaps. • You need to sort or rank in-memory datasets. • You need persistence of your key store. • You need to replicate your data from the primary to one or more read replicas for read intensive applications. • You need automatic failover if your primary node fails. • You need pub/sub capabilities. • You need backup and restore capabilities. • You need to support multiple databases.	• You need the simplest model possible. • You need to run large nodes with multiple cores or threads. • You need the ability to scale out and in, adding and removing nodes as demand on your system increases and decreases. • You need to cache objects, such as a database. • Needs Auto Discovery to simplify the way an application connects to a cluster.

Tutorials Dojo

- Pricing
    - With on-demand nodes you pay only for the resources you consume by the hour without any long-term commitments.
    - With Reserved Nodes, you can make a low, one-time, up-front payment for each node you wish to reserve for a 1 or 3 year term. In return, you receive a significant discount off the ongoing hourly usage rate for the Node(s) you reserve.
    - ElastiCache provides storage space for one snapshot free of charge for each active ElastiCache for Redis cluster. Additional backup storage is charged.
    - EC2 Regional Data Transfer charges apply when transferring data between an EC2 instance and an ElastiCache Node in different Availability Zones of the same Region.

Sources:
https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/
https://aws.amazon.com/elasticache/redis-details/
https://docs.aws.amazon.com/AmazonElastiCache/latest/mem-ug/
https://aws.amazon.com/elasticache/redis-vs-memcached/
https://aws.amazon.com/elasticache/features/
https://aws.amazon.com/elasticache/pricing/

Amazon Redshift

- A fully managed, **petabyte-scale data warehouse** service.
- Redshift extends data warehouse queries to your data lake. You can run analytic queries against petabytes of data stored locally in Redshift, and directly against exabytes of data stored in S3.
- RedShift is an OLAP type of DB.
- Currently, Redshift only supports Single-AZ deployments.
- Features
    - Redshift uses **columnar storage**, data compression, and zone maps to reduce the amount of I/O needed to perform queries.
    - It uses a **massively parallel processing** data warehouse architecture to parallelize and distribute SQL operations.
    - Redshift uses machine learning to deliver high throughput based on your workloads.
    - Redshift uses **result caching** to deliver sub-second response times for repeat queries.
    - Redshift automatically and continuously backs up your data to S3. It can asynchronously replicate your snapshots to S3 in another region for disaster recovery.
- Components
    - **Cluster** - a set of **nodes**, which consists of a leader node and one or more compute nodes.
        - Redshift creates one database when you provision a cluster. This is the database you use to load data and run queries on your data.
        - You can scale the cluster in or out by adding or removing nodes. Additionally, you can scale the cluster up or down by specifying a different node type.
        - Redshift assigns a 30-minute maintenance window at random from an 8-hour block of time per region, occurring on a random day of the week. During these maintenance windows, your cluster is not available for normal operations.
        - Redshift supports both the EC2-VPC and EC2-Classic platforms to launch a cluster. You create a **cluster subnet group** if you are provisioning your cluster in your VPC, which allows you to specify a set of subnets in your VPC.
    - **Redshift Nodes**
        - The *leader node* receives queries from client applications, parses the queries, and develops query execution plans. It then coordinates the parallel execution of these plans with the compute nodes and aggregates the intermediate results from these nodes. Finally, it returns the results back to the client applications.
        - *Compute nodes* execute the query execution plans and transmit data among themselves to serve these queries. The intermediate results are sent to the leader node for aggregation before being sent back to the client applications.
        - Node Type
            - **Dense storage (DS)** node type - for large data workloads and use hard disk drive (HDD) storage.
            - **Dense compute (DC)** node types - optimized for performance-intensive workloads. Uses SSD storage.

    - **Parameter Groups** - a group of parameters that apply to all of the databases that you create in the cluster. The default parameter group has preset values for each of its parameters, and it cannot be modified.
- Database Querying Options
    - Connect to your cluster and run queries on the AWS Management Console with the Query Editor.
    - Connect to your cluster through a SQL client tool using standard ODBC and JDBC connections.
- Enhanced VPC Routing
    - By using Enhanced VPC Routing, you can use VPC features to manage the flow of data between your cluster and other resources.
    - You can also use VPC flow logs to monitor *COPY* and *UNLOAD* traffic.
- RedShift Spectrum
    - Enables you to run queries against exabytes of data in S3 without having to load or transform any data.
    - Redshift Spectrum supports Enhanced VPC Routing.
    - If you store data in a columnar format, Redshift Spectrum scans only the columns needed by your query, rather than processing entire rows.
    - If you compress your data using one of Redshift Spectrum's supported compression algorithms, less data is scanned.
- Cluster Snapshots
    - Point-in-time backups of a cluster. There are two types of snapshots: automated and manual. Snapshots are stored in S3 using SSL.
    - Redshift periodically takes incremental snapshots of your data every 8 hours or 5 GB per node of data change.
    - Redshift provides free storage for snapshots that is equal to the storage capacity of your cluster until you delete the cluster. After you reach the free snapshot storage limit, you are charged for any additional storage at the normal rate.
    - Automated snapshots are enabled by default when you create a cluster. These snapshots are deleted at the end of a retention period, which is one day, but you can modify it. You cannot delete an automated snapshot manually.
    - By default, manual snapshots are retained indefinitely, even after you delete your cluster.
    - You can share an existing manual snapshot with other AWS accounts by authorizing access to the snapshot.
    - You can configure Amazon Redshift to automatically copy snapshots (automated or manual) for a cluster to another AWS Region. For automated snapshots, you can also specify the retention period to keep them in the destination AWS Region. The default retention period for copied snapshots is seven days.
    - If you store a copy of your snapshots in another AWS Region, you can restore your cluster from recent data if anything affects the primary AWS Region. You can configure your cluster to copy snapshots to only one destination AWS Region at a time.
- Monitoring

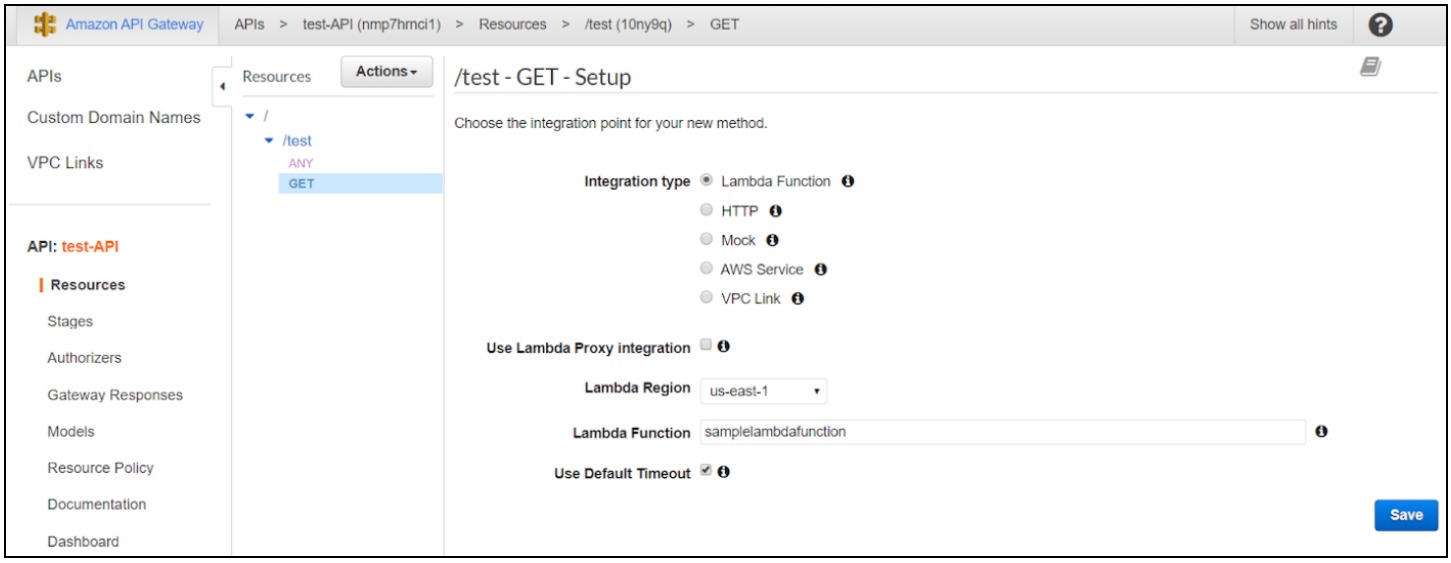
    - Use the *database audit logging* feature to track information about authentication attempts, connections, disconnections, changes to database user definitions, and queries run in the database. The logs are stored in S3 buckets.
    - Redshift tracks events and retains information about them for a period of several weeks in your AWS account.
    - Redshift provides performance metrics and data so that you can track the health and performance of your clusters and databases. It uses CloudWatch metrics to monitor the physical aspects of the cluster, such as CPU utilization, latency, and throughput.
    - *Query/Load performance data* helps you monitor database activity and performance.
    - When you create a cluster, you can optionally configure a CloudWatch alarm to monitor the average percentage of disk space that is used across all of the nodes in your cluster, referred to as the *default disk space* alarm.
- Security
    - By default, an Amazon Redshift cluster is only accessible to the AWS account that creates the cluster.
    - Use IAM to create user accounts and manage permissions for those accounts to control cluster operations.
    - If you are using the EC2-Classic platform for your Redshift cluster, you must use Redshift security groups.
    - If you are using the EC2-VPC platform for your Redshift cluster, you must use VPC security groups.
    - When you provision the cluster, you can optionally choose to encrypt the cluster for additional security. Encryption is an immutable property of the cluster.
    - Snapshots created from the encrypted cluster are also encrypted.
- Pricing
    - You pay a per-second billing rate based on the type and number of nodes in your cluster.
    - You pay for the number of bytes scanned by RedShift Spectrum
    - You can reserve instances by committing to using Redshift for a 1 or 3 year term and save costs.

Sources:
https://docs.aws.amazon.com/redshift/latest/mgmt/
https://aws.amazon.com/redshift/features/
https://aws.amazon.com/redshift/pricing/
https://aws.amazon.com/redshift/faqs/

NETWORKING AND CONTENT DELIVERY

Amazon API Gateway

- Enables developers to create, publish, maintain, monitor, and secure APIs at any scale.
- This is a HIPAA eligible service.
- Allows creating, deploying, and managing a RESTful API to expose backend HTTP endpoints, Lambda functions, or other AWS services.



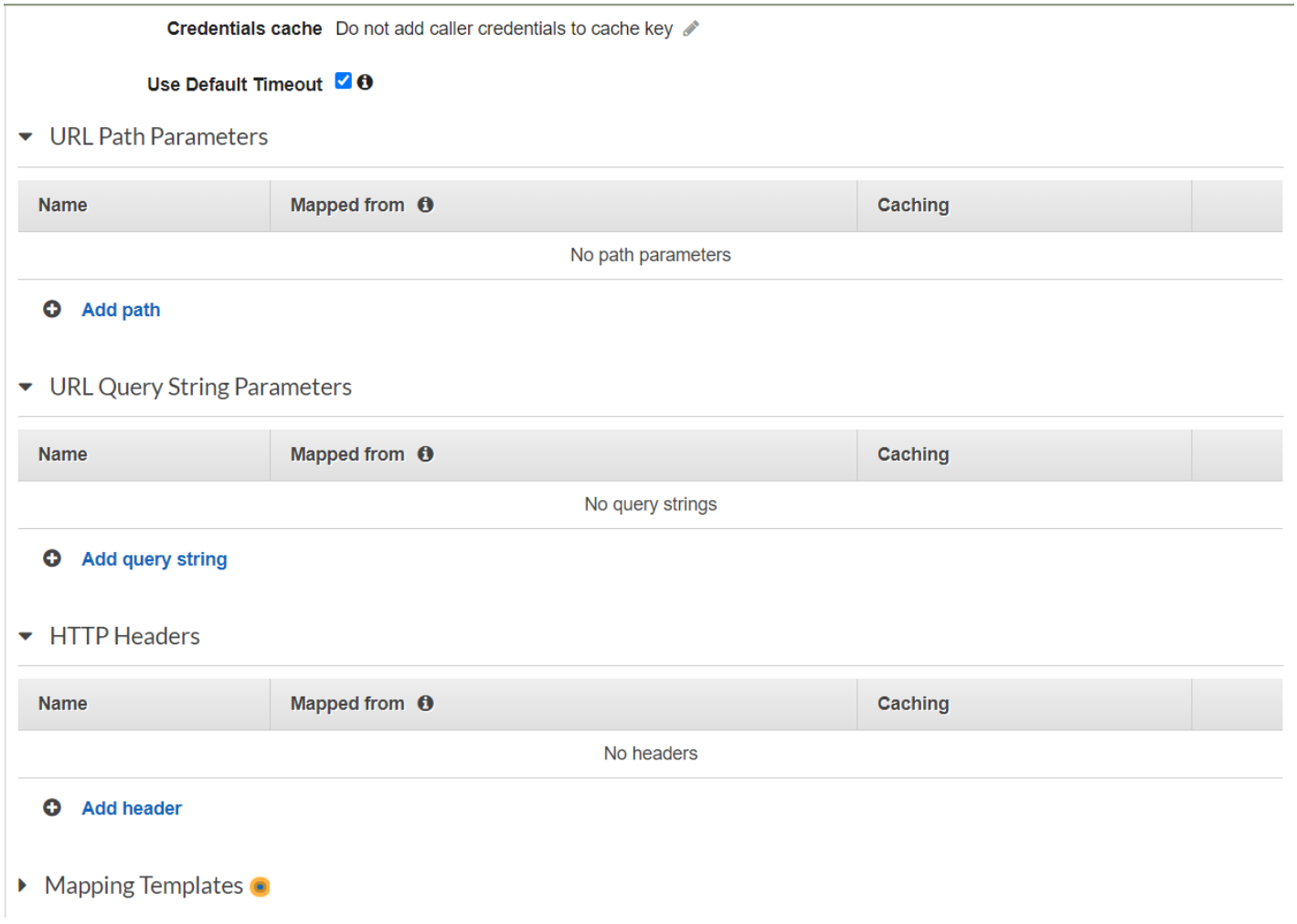
- Together with Lambda, API Gateway forms the app-facing part of the AWS serverless infrastructure.
- Concepts
    - **API deployment** - a point-in-time snapshot of your API Gateway API resources and methods. To be available for clients to use, the deployment must be associated with one or more API stages.
    - **API endpoints** - host names APIs in API Gateway, which are deployed to a specific region and of the format: *rest-api-id*.execute-api.*region*.amazonaws.com
    - **API key** - An alphanumeric string that API Gateway uses to identify an app developer who uses your API.
    - **API stage** - A logical reference to a lifecycle state of your API. API stages are identified by API ID and stage name.
    - **Model** - Data schema specifying the data structure of a request or response payload.
    - **Private API** - An API that is exposed through interface VPC endpoints and isolated from the public internet
    - **Private integration** - An API Gateway integration type for a client to access resources inside a customer's VPC through a private API endpoint without exposing the resources to the public internet.

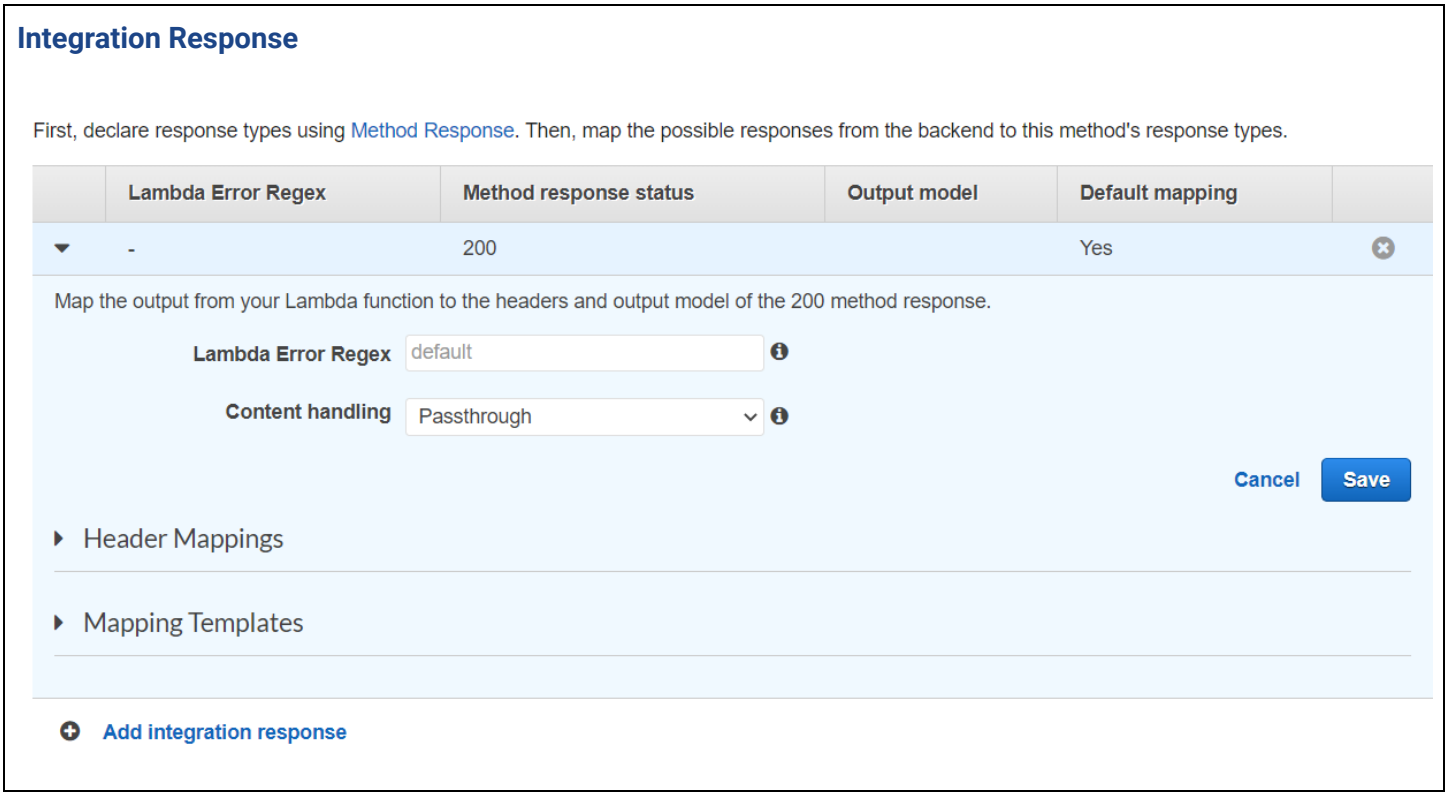
- **Proxy integration** - You can set up a proxy integration as an HTTP proxy integration type or a Lambda proxy integration type.
  - For the HTTP proxy integration, API Gateway passes the entire request and response between the frontend and an HTTP backend.
  - For the Lambda proxy integration, API Gateway sends the entire request as an input to a backend Lambda function.

CDA Exam Notes:

Most of the time, you'll want to use Lambda proxy integration when using API Gateway with Lambda, so you can handle the request as an AWS event. If you want to customize the request and response data that will be handled by your Lambda function, you can disable proxy integration and manually map the data to API Gateway's integration request and response.

Integration Request





    - **Usage plan** - Provides selected API clients with access to one or more deployed APIs. You can use a usage plan to configure throttling and quota limits, which are enforced on individual client API keys.
- API Endpoint Types
    - **Edge-optimized API endpoint**: The default host name of an API Gateway API that is deployed to the specified region while using a CloudFront distribution to facilitate client access typically from across AWS regions. API requests are routed to the nearest CloudFront Point of Presence.
    - **Regional API endpoint**: The host name of an API that is deployed to the specified region and intended to serve clients, such as EC2 instances, in the same AWS region. API requests are targeted directly to the region-specific API Gateway without going through any CloudFront distribution.
        - You can apply latency-based routing on regional endpoints to deploy an API to multiple regions using the same regional API endpoint configuration, set the same custom domain name for each deployed API, and configure latency-based DNS records in Route 53 to route client requests to the region that has the lowest latency.
    - **Private API endpoint**: Allows a client to securely access private API resources inside a VPC. Private APIs are isolated from the public Internet, and they can only be accessed using VPC endpoints for API Gateway that have been granted access.
- Features

    - API Gateway can execute Lambda code in your account, start Step Functions state machines, or make calls to Elastic Beanstalk, EC2, or web services outside of AWS with publicly accessible HTTP endpoints.
    - API Gateway helps you define plans that meter and restrict third-party developer access to your APIs.
    - API Gateway helps you manage traffic to your backend systems by allowing you to set throttling rules based on the number of requests per second for each HTTP method in your APIs.
    - You can set up a cache with customizable keys and time-to-live in seconds for your API data to avoid hitting your backend services for each request.
    - API Gateway lets you run multiple versions of the same API simultaneously with **API Lifecycle**.
    - After you build, test, and deploy your APIs, you can package them in an API Gateway usage plan and sell the plan as a Software as a Service (SaaS) product through AWS Marketplace.
    - API Gateway offers the ability to create, update, and delete documentation associated with each portion of your API, such as methods and resources.
    - Amazon API Gateway offers general availability of HTTP APIs, which gives you the ability to route requests to private ELBs and IP-based services registered in AWS CloudMap such as ECS tasks. Previously, HTTP APIs enabled customers to only build APIs for their serverless applications or to proxy requests to HTTP endpoints.
- All of the APIs created expose **HTTPS endpoints only**. API Gateway does not support unencrypted (HTTP) endpoints.
- Controlling access to APIs
    - You can control who can access your API Gateway APIs by creating an authorizer.
        - A **Lambda authorizer** (formerly known as a custom authorizer) is a Lambda function that you provide to control access to your API. When your API is called, this Lambda function is invoked with a request context or an authorization token that is provided by the client application. The Lambda function returns a *policy document* that specifies the operations that the caller is authorized to perform, if any. There are two types of Lambda authorizers:
            - Token based type receives the caller's identity in a bearer token, such as a JSON Web Token (JWT) or an OAuth token.
            - Request parameter based type receives the caller's identity in a combination of headers, query string parameters, stageVariables, and $context variables.
        - **Amazon Cognito user pools** are user directories in Amazon Cognito. A client of your API must first sign a user in to the user pool and obtain an identity or access token for the user. Then your API is called with one of the returned tokens. The API call succeeds only if the required token is valid.
- Monitoring
    - API Gateway console is integrated with CloudWatch, so you get backend performance metrics such as API calls, latency, and error rates.
    - You can set up custom alarms on API Gateway APIs.
    - API Gateway can also log API execution errors to CloudWatch Logs.

- Security
    - To authorize and verify API requests to AWS services, API Gateway can help you leverage signature version 4. Using signature version 4 authentication, you can use IAM and access policies to authorize access to your APIs and all your other AWS resources.
    - You can enable AWS WAF for your APIs in Amazon API Gateway, making it easier to protect your APIs against common web exploits such as SQL injection and Cross-Site Scripting (XSS).
- Pricing
    - You pay only for the API calls you receive and the amount of data transferred out.
    - API Gateway also provides optional data caching charged at an hourly rate that varies based on the cache size you select.

Sources:
https://docs.aws.amazon.com/apigateway/latest/developerguide/
https://aws.amazon.com/api-gateway/features/
https://aws.amazon.com/api-gateway/pricing/
https://aws.amazon.com/api-gateway/faqs/

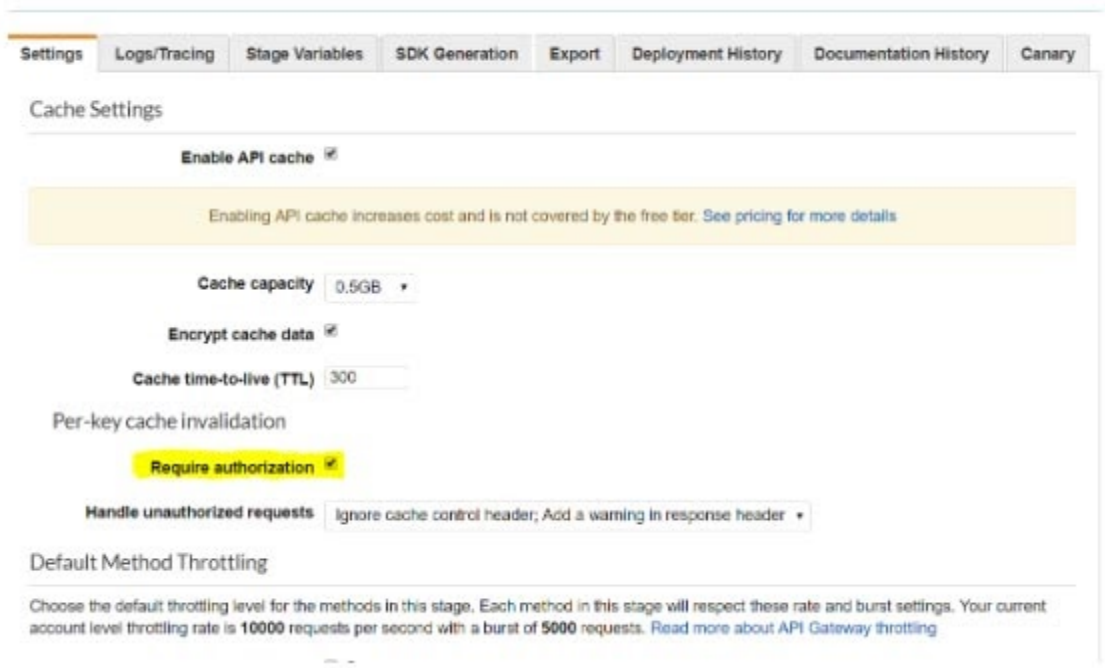
How to Invalidate API Gateway Cache

To invalidate an existing cache entry of a request and retrieve the latest data from the integration endpoint, one must send the request together with the **Cache-Control: max-age=0** header. If the recipient is authorized to communicate directly to the integration endpoint, then the integration endpoint will respond with the latest data for the request. This also replaces the existing cache entry with the new response.

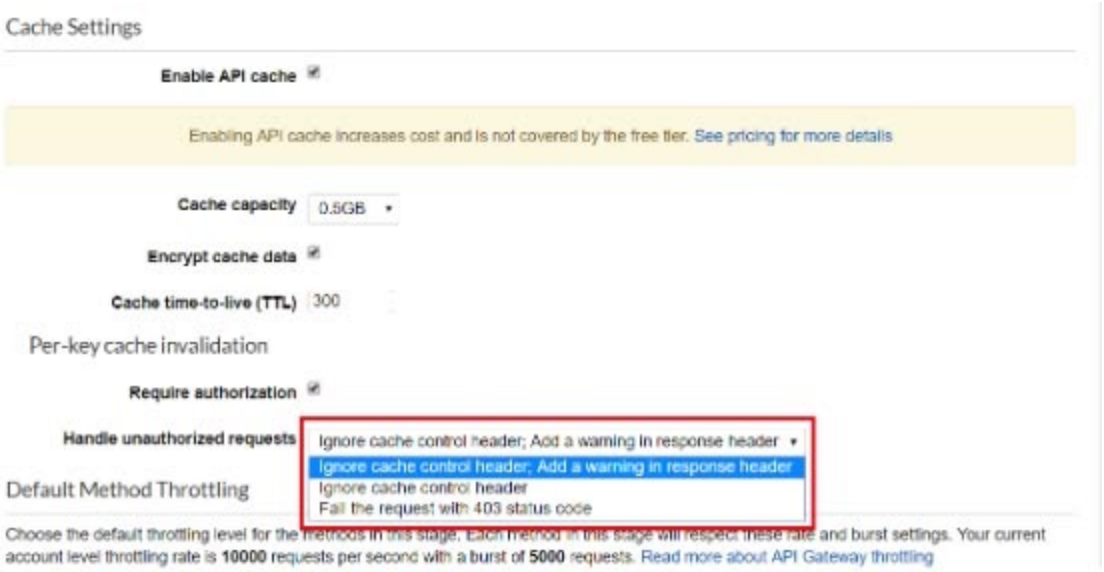
The IAM Policy that grants a client to invalidate the cache follows:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:InvalidateCache"
      ],
      "Resource": [
        "arn:aws:execute-api:region:account-id:api-id/stage-name/GET/resource-path-specifier"
      ]
    }
  ]
}
```

An alternative option for requiring authorization, aside from using the policy above, is to place a checkmark on **Require Authorization** checkbox. This checkbox can be seen in the Settings tab of your Deployment stage, after you enable API caching.



If you have enabled caching and authorization, you can also configure how unauthorized requests are handled:



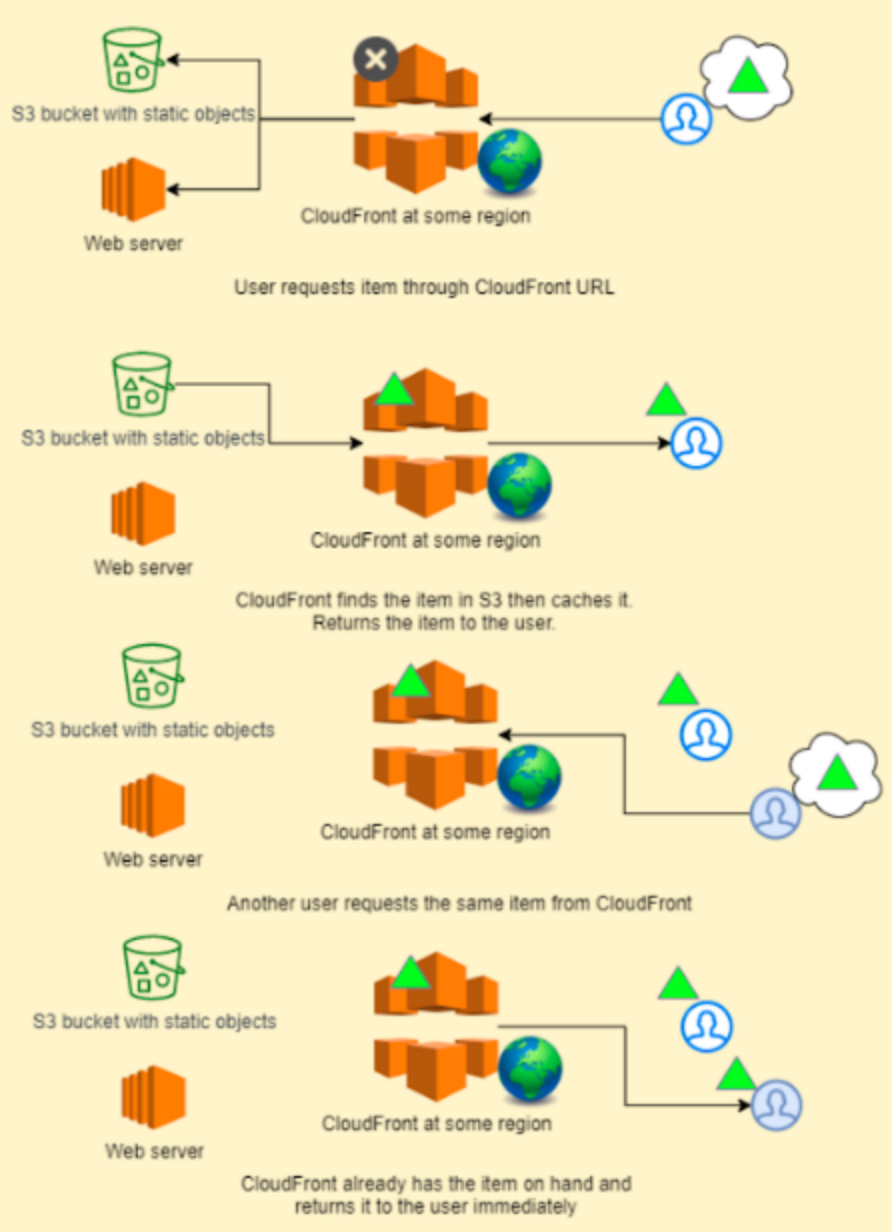
- **Fail the request with 403 status code**: returns a 403 Unauthorized response.
- **Ignore cache control header; Add a warning in response header**: process the request and add a warning header in the response.

- **Ignore cache control header**: process the request and do not add a warning header in the response.

Source:
https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-caching.html#invalidate-method-caching

Amazon CloudFront

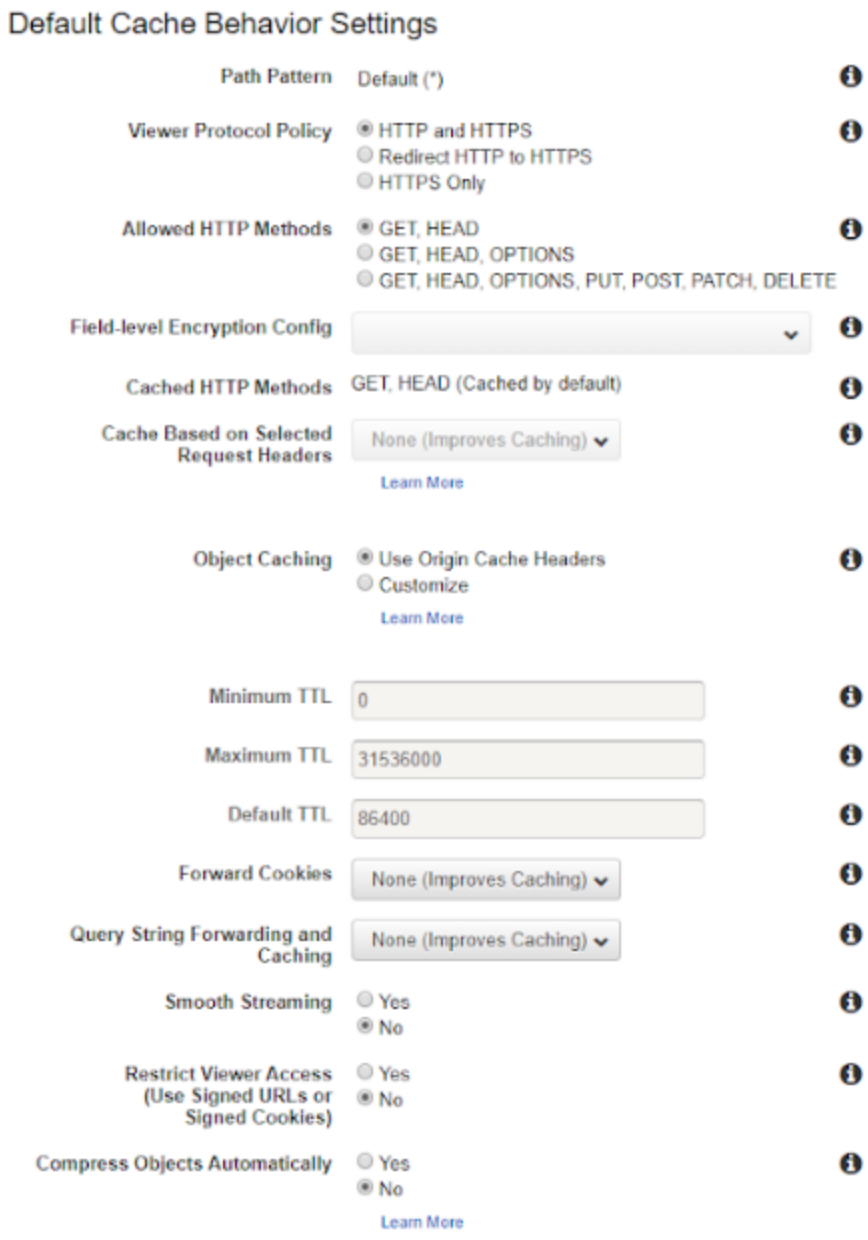
- A web service that speeds up distribution of your static and dynamic web content to your users. A Content Delivery Network (CDN) service.
- It delivers your content through a worldwide network of data centers called **edge locations**. When a user requests content that you're serving with CloudFront, the user is routed to the edge location that provides the lowest latency, so that content is delivered with the best possible performance.
    - If the content is already in the edge location with the lowest latency, CloudFront delivers it immediately.
    - If the content is not in that edge location, CloudFront retrieves it from an origin that you've defined
- How CloudFront Delivers Content
    - You specify **origin servers**, like an S3 bucket or your own HTTP server, from which CloudFront gets your files which will then be distributed from CloudFront edge locations all over the world.
    - Upload your files to your origin servers. Your files, also known as **objects**.
    - Create a **CloudFront distribution**, which tells CloudFront which origin servers to get your files from when users request the files through your web site or application. At the same time, you specify details such as whether you want CloudFront to log all requests and whether you want the distribution to be enabled as soon as it's created.
    - CloudFront assigns a domain name to your new distribution that you can see in the CloudFront console.
    - CloudFront sends your distribution's configuration (but not your content) to all of its **edge locations**—collections of servers in geographically dispersed data centers where CloudFront caches copies of your objects.



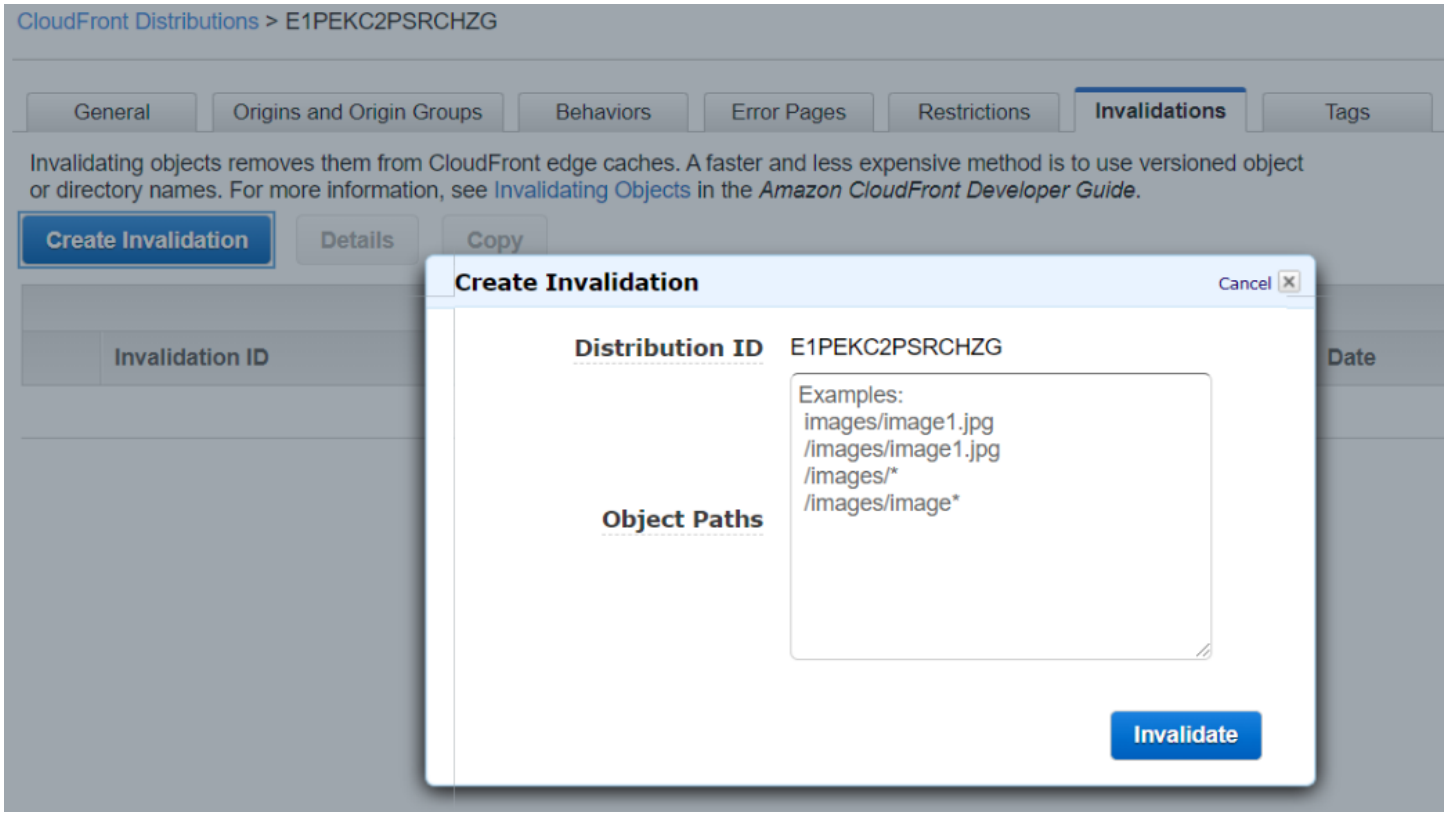
- CloudFront supports the **WebSocket protocol** as well as the **HTTP protocol** with the following HTTP methods:
  - GET
  - HEAD
  - POST
  - PUT
  - DELETE
  - OPTIONS
  - PATCH

- Using **Lambda@Edge** with CloudFront enables a variety of ways to customize the content that CloudFront delivers. It can help you configure your CloudFront distribution to serve private content from your own custom origin, as an option to using signed URLs or signed cookies.(See AWS Compute Services Lambda Lambda@Edge)
- CloudFront also has **regional edge caches** that bring more of your content closer to your viewers, even when the content is not popular enough to stay at a CloudFront edge location, to help improve performance for that content.
- You can use a zone apex name on CloudFront
- CloudFront supports wildcard CNAME
- Different CloudFront Origins
    - **Using S3 buckets for your origin** - you place any objects that you want CloudFront to deliver in an S3 bucket.
    - **Using S3 buckets configured as website endpoints for your origin**
    - **Using a mediastore container or a media package channel for your origin** - you can set up an S3 bucket that is configured as a MediaStore container, or create a channel and endpoints with MediaPackage. Then you create and configure a distribution in CloudFront to stream the video.
    - **Using EC2 or other custom origins** - A custom origin is an HTTP server, for example, a web server.
    - **Using CloudFront Origin Groups for origin failover** - use origin failover to designate a primary origin for CloudFront plus a second origin that CloudFront automatically switches to when the primary origin returns specific HTTP status code failure responses.
- Objects are cached for 24 hours by default. You can invalidate files in CloudFront edge caches even before they expire.
- You can configure CloudFront to automatically compress files of certain types and serve the compressed files when viewer requests include *Accept-Encoding: gzip* in the request header.
- CloudFront can cache different versions of your content based on the values of query string parameters.
- CloudFront Distributions
    - You create a **CloudFront distribution** to tell CloudFront where you want content to be delivered from, and the details about how to track and manage content delivery.
    - You create a distribution and choose the configuration settings you want:
        - Your content origin—that is, the Amazon S3 bucket, MediaPackage channel, or HTTP server from which CloudFront gets the files to distribute. You can specify any combination of up to 25 S3 buckets, channels, and/or HTTP servers as your origins.
        - Access—whether you want the files to be available to everyone or restrict access to some users.
        - Security—whether you want CloudFront to require users to use HTTPS to access your content.
        - Cookie or query-string forwarding—whether you want CloudFront to forward cookies or query strings to your origin.

        - Geo-restrictions—whether you want CloudFront to prevent users in selected countries from accessing your content.
        - Access logs—whether you want CloudFront to create access logs that show viewer activity.
    - You can use distributions to serve the following content over HTTP or HTTPS:
        - Static and dynamic download content.
        - Video on demand in different formats, such as Apple HTTP Live Streaming (HLS) and Microsoft Smooth Streaming.
        - A live event, such as a meeting, conference, or concert, in real time.
    - Values that you specify when you create or update a distribution
        - Delivery Method - Web or RTMP.
        - Origin Settings - information about one or more locations where you store the original versions of your web content.
        - Cache Behavior Settings - lets you configure a variety of CloudFront functionality for a given URL path pattern for files on your website.
        - Custom Error Pages and Error Caching
        - Restrictions - if you need to prevent users in selected countries from accessing your content, you can configure your CloudFront distribution either to allow users in a **whitelist** of specified countries to access your content or to not allow users in a **blacklist** of specified countries to access your content.
- Cache Behavior Settings
    - The functionality that you can configure for each cache behavior includes:
        - The path pattern.
        - If you have configured multiple origins for your CloudFront distribution, which origin you want CloudFront to forward your requests to.
        - Whether to forward query strings to your origin.
        - Whether accessing the specified files requires signed URLs.
        - Whether to require users to use HTTPS to access those files.
        - The minimum amount of time that those files stay in the CloudFront cache regardless of the value of any Cache-Control headers that your origin adds to the files.



- After creating your CloudFront distribution, you can invalidate its cached items by creating an invalidation request.



- Price Class
  - Choose the price class that corresponds with the maximum price that you want to pay for CloudFront service. By default, CloudFront serves your objects from edge locations in all CloudFront regions.
- Performance and Availability
  - CloudFront also allows you to set up multiple origins to enable redundancy with **Origin Failover**. To set up origin failover, you must have a distribution with at least two origins. Next, you create an origin group for your distribution that includes the two origins, setting one as the primary. Finally, you define a cache behavior in which you specify the origin group as your origin.
    - The two origins in the origin group can be any combination of the following: AWS origins, like Amazon S3 buckets or Amazon EC2 instances, or custom origins, like your own HTTP web server.
    - When you create the origin group, you configure CloudFront to failover to the second origin for GET, HEAD, and OPTIONS HTTP methods when the primary origin returns specific status codes that you configure.
  - CloudFront is optimized for both dynamic and static content, providing extensive flexibility for optimizing cache behavior, coupled with network-layer optimizations for latency and throughput.
- Using HTTPS with CloudFront

    - You can choose HTTPS settings both for communication between viewers and CloudFront, and between CloudFront and your origin.
    - If you want your viewers to use HTTPS and to use alternate domain names for your files, you need to choose one of the following options for how CloudFront serves HTTPS requests:
        - Use a dedicated IP address in each edge location
        - Use Server Name Indication (SNI)
- Monitoring
    - The billing report is a high-level view of all of the activity for the AWS services that you're using, including CloudFront.
    - The usage report is a summary of activity for a service such as CloudFront, aggregated by hour, day, or month. It also includes usage charts that provide a graphical representation of your CloudFront usage.
    - CloudFront console includes a variety of reports based on the data in CloudFront access logs:
        - CloudFront Cache Statistics Reports
        - CloudFront Popular Objects Report
        - CloudFront Top Referrers Report
        - CloudFront Usage Reports
        - CloudFront Viewers Reports
    - You can use AWS Config to record configuration changes for CloudFront distribution settings changes.
    - CloudFront integrates with Amazon CloudWatch metrics so that you can monitor your website or application.
    - Capture API requests with AWS CloudTrail. CloudFront is a global service. To view CloudFront requests in CloudTrail logs, you must update an existing trail to include global services.
- Security
    - CloudFront, AWS Shield, AWS WAF, and Route 53 work seamlessly together to create a flexible, layered security perimeter against multiple types of attacks including network and application layer DDoS attacks.
    - You can deliver your content, APIs or applications via SSL/TLS, and advanced SSL features are enabled automatically.
    - Through geo-restriction capability, you can prevent users in specific geographic locations from accessing content that you're distributing through CloudFront.
    - With **Origin Access Identity** feature, you can restrict access to an S3 bucket to only be accessible from CloudFront.
    - **Field-Level Encryption** is a feature of CloudFront that allows you to securely upload user-submitted data such as credit card numbers to your origin servers.
- Pricing
    - Charge for storage in an S3 bucket.
    - Charge for serving objects from edge locations.
    - Charge for submitting data to your origin.
        - Data Transfer Out

        - HTTP/HTTPS Requests
        - Invalidation Requests,
        - Dedicated IP Custom SSL certificates associated with a CloudFront distribution.
    - You also incur a surcharge for HTTPS requests, and an additional surcharge for requests that also have field-level encryption enabled.
- Compliance
    - CloudFront has been validated as being compliant with Payment Card Industry (PCI) Data Security Standard (DSS).
    - CloudFront is a HIPAA eligible service.
    - CloudFront is compliant with SOC measures.

Sources:
https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide
https://aws.amazon.com/cloudfront/features/
https://aws.amazon.com/cloudfront/pricing/
https://aws.amazon.com/cloudfront/faqs/

AWS Elastic Load Balancing

- Distributes incoming application or network traffic across multiple targets, such as **EC2 instances**, **containers (ECS)**, **Lambda functions,** and **IP addresses**, in multiple Availability Zones.
- When you create a load balancer, you must specify one public subnet from at least two Availability Zones. You can specify only one public subnet per Availability Zone.

General features

- Accepts incoming traffic from clients and routes requests to its registered targets.
- Monitors the health of its registered targets and routes traffic only to healthy targets.
- Enable deletion protection to prevent your load balancer from being deleted accidentally. Disabled by default.
- Deleting ELB won't delete the instances registered to it.
- **Cross Zone Load Balancing** - when enabled, each load balancer node distributes traffic across the registered targets in all enabled AZs.
- Supports SSL Offloading which is a feature that allows the ELB to bypass the SSL termination by removing the SSL-based encryption from the incoming traffic.

Three Types of Load Balancers

- **Application Load Balancer**
    - Functions at the application layer, the **seventh layer** of the Open Systems Interconnection (OSI) model.
    - Allows HTTP and HTTPS.
    - At least 2 subnets must be specified when creating this type of load balancer.
    - Components:
        - A *load balancer* serves as the single point of contact for clients.
        - A *listener* checks for connection requests from clients. You must define a default rule for each listener that specifies a target group, condition, and priority.
        - *Target group* routes requests to one or more registered targets. You can register a target with multiple target groups, and configure health checks on a per target group basis.
    - Benefits
        - Support for path-based and host-based routing.
        - Support for routing requests to multiple applications on a single EC2 instance.
        - Support for registering targets by IP address, including targets outside the VPC for the load balancer.
        - Support for containerized applications.
        - Support for monitoring the health of each service independently.
        - Support for redirecting requests from one URL to another.
        - Support for returning a custom HTTP response.

        - Support for the load balancer to authenticate users of your applications before routing requests using OIDC-compliant identity providers and/or Amazon Cognito user pools.
        - Support for registering Lambda functions as targets.
    - Cross-zone load balancing is always enabled.
    - If you specify targets using an **instance ID**, traffic is routed to instances using the primary private IP address specified in the primary network interface for the instance. If you specify targets using **IP addresses**, you can route traffic to an instance using any private IP address from one or more network interfaces.
    - You can also specify Lambda functions are targets to serve HTTP(S) requests.
    - Supports load balancer-generated cookies only for sticky sessions.
    - HTTP/2 Support
    - WebSockets Support
    - Monitoring:
        - CloudWatch metrics - retrieve statistics about data points for your load balancers and targets as an ordered set of time-series data, known as *metrics*.
        - Access logs - capture detailed information about the requests made to your load balancer and store them as log files in S3.
        - Request tracing - track HTTP requests.
        - CloudTrail logs - capture detailed information about the calls made to the Elastic Load Balancing API and store them as log files in S3.
- **Network Load Balancer**
    - Functions at the **fourth layer** of the Open Systems Interconnection (OSI) model. Uses TCP, TLS and UDP connections.
    - At least 1 subnet must be specified when creating this type of load balancer, but the recommended number is 2.
    - Components:
        - A load balancer serves as the single point of contact for clients.
        - A listener checks for connection requests from clients.
        - A target group routes requests to one or more registered targets. You can register a target with multiple target groups. You can configure health checks on a per target group basis.
    - Benefits
        - Ability to handle volatile workloads and scale to millions of requests per second.
        - Support for static IP addresses for the load balancer, or assign one Elastic IP address per subnet enabled for the load balancer.
        - Support for registering targets by IP address.
        - Support for routing requests to multiple applications on a single EC2 instance (register each instance or IP address with the same target group using multiple ports).
        - Support for containerized applications.
        - Support for monitoring the health of each service independently.
    - Cross-zone load balancing is disabled by default.

- If you specify targets using an **instance ID**, the source IP addresses of the clients are preserved and provided to your applications. If you specify targets by **IP address**, the source IP addresses are the private IP addresses of the load balancer nodes.
- Network Load Balancers support connections from clients over inter-region VPC peering, AWS managed VPN, and third-party VPN solutions.
- You can deploy services that rely on the UDP protocol, such as Authentication and Authorization, Logging, DNS, and IoT, behind a Network Load Balancer
- Offers multi-protocol listeners, allowing you to run applications such as DNS that rely on both TCP and UDP protocols on the same port behind a Network Load Balancer.
- You CANNOT enable or disable Availability Zones for a Network Load Balancer after you create it.
- Network Load Balancers use Proxy Protocol version 2 to send additional connection information such as the source and destination.
- Preserves the client side source IP allowing the back-end to see the IP address of the client. This can then be used by applications for further processing.
- Automatically provides a static IP per Availability Zone (subnet) that can be used by applications as the front-end IP of the load balancer.
- Zonal Isolation
- In the event that your Network load balancer is unresponsive, integration with Route 53 will remove the unavailable load balancer IP address from service and direct traffic to an alternate Network Load Balancer in another region.
- Supports TLS termination on Network Load Balancers. Additionally, Network Load Balancers preserve the source IP of the clients to the back-end applications, while terminating TLS on the load balancer.
- Monitoring:
  - CloudWatch metrics - retrieve statistics about data points for your load balancers and targets as an ordered set of time-series data, known as *metrics*.
  - VPC Flow Logs - capture detailed information about the traffic going to and from your Network Load Balancer.
  - CloudTrail logs - capture detailed information about the calls made to the Elastic Load Balancing API and store them as log files in Amazon S3.

- **Classic Load Balancer**
  - Distributes incoming application traffic across multiple EC2 instances in multiple Availability Zones.
  - For use with EC2 classic only. Register instances with the load balancer. AWS recommends using Application or Network load balancers instead.
  - To ensure that your registered instances are able to handle the request load in each AZ, keep approximately the same number of instances in each AZ registered with the load balancer.
  - Benefits
    - Support for EC2-Classic
    - Support for TCP and SSL listeners

        - Support for sticky sessions using application-generated cookies
    - An **Internet-facing load balancer** has a publicly resolvable DNS name, so it can route requests from clients over the Internet to the EC2 instances that are registered with the load balancer. Classic load balancers are always Internet-facing.
    - Monitoring:
        - CloudWatch metrics - retrieve statistics about ELB-published data points as an ordered set of time-series data, known as *metrics*.
        - Access logs - capture detailed information for requests made to your load balancer and stores them as log files in the S3 bucket that you specify.
        - CloudTrail logs - keep track of the calls made to the Elastic Load Balancing API by or on behalf of your AWS account
- **HTTP Headers**
    - Application Load Balancers and Classic Load Balancers support **X-Forwarded-For, X-Forwarded-Proto**, and **X-Forwarded-Port** headers.
- Choose whether to make an **internal load balancer** or an **Internet-facing load balancer**. Classic Load Balancer in EC2-Classic must be an Internet-facing load balancer.
    - The nodes of an Internet-facing load balancer have public IP addresses.
    - The nodes of an internal load balancer have only private IP addresses.
- Public DNS name format for your load balancers
    - EC2-VPC : *name-1234567890.region*.elb.amazonaws.com (supports IPv4 addresses only)
    - EC2-Classic: (support both IPv4 and IPv6 addresses)
        - *name-123456789.region*.elb.amazonaws.com
        - ipv6.*name-123456789.region*.elb.amazonaws.com
        - dualstack.*name-123456789.region*.elb.amazonaws.com
- **Load Balancer States**
    - **Provisioning** - The load balancer is being set up.
    - **Active** - The load balancer is fully set up and ready to route traffic.
    - **Failed** - The load balancer could not be set up.
- By default, ELB idle timeout value to **60 seconds**. If a target doesn't send data at least every 60 seconds while the request is in flight, the load balancer can close the front-end connection. For back-end connections, enable the *HTTP keep-alive* option for your EC2 instances.
- You can register each EC2 instance or IP address with the same target group multiple times using different ports, which enables the load balancer to route requests to microservices.
- **Listeners** define the port and protocol to listen on.
- **Listener rules** determine how the load balancer routes requests to the targets in one or more target groups. You can add rules that specify different target groups based on the content of the request. If no rules are found, the default rule will be followed. Parts are:
    - Rule priority
    - Rule action
    - Rule conditions

- **Slow Start Mode** gives targets time to warm up before the load balancer sends them a full share of requests.
- **Sticky sessions** route requests to the same target in a target group. You enable sticky sessions at the target group level. You can also set the duration for the stickiness of the load balancer-generated cookie, in seconds. Useful if you have stateful applications.
- **Health checks** verify the status of your targets. The statuses for a registered target are:

Value	Description
initial	The load balancer is in the process of registering the target or performing the initial health checks on the target.
healthy	The target is healthy.
unhealthy	The target did not respond to a health check or failed the health check.
unused	The target is not registered with a target group, the target group is not used in a listener rule for the load balancer, or the target is in an Availability Zone that is not enabled for the load balancer.
draining	The target is deregistering and connection draining is in process.

Security, Authentication and Access Control

- Use IAM Policies to grant permissions
- Resource-level permissions
- Security groups that control the traffic allowed to and from your load balancer. Recommended rules for internet-facing load balancer:

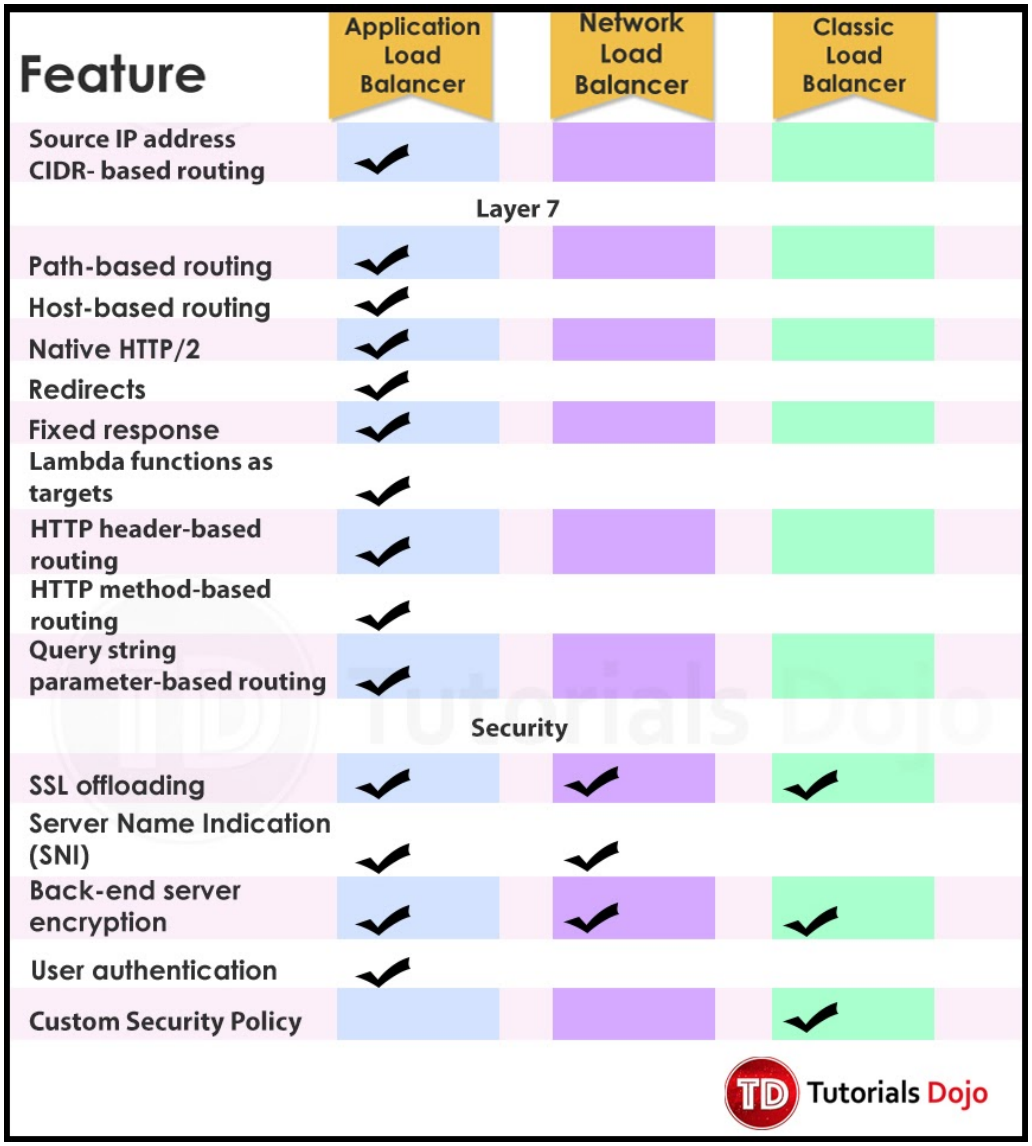
Inbound	
Source	**Port Range**
0.0.0.0/0	*listener*
Outbound	
Destination	***Port Range***
instance security group	*instance listener*
instance security group	*health check*

For internal load balancer:

Inbound	
Source	**Port Range**
VPC CIDR	*listener*
Outbound	
Destination	**Port Range**
instance security group	*instance listener*
instance security group	*health check*

Summary of Features

Feature	Application Load Balancer	Network Load Balancer	Classic Load Balancer
Protocols	HTTP HTTPS	TCP, UDP, TLS	TCP, SSL/TLS, HTTP, HTTPS
Platforms	VPC	VPC	EC2-Classic, VPC
Healthchecks	✓	✓	✓
Cloudwatch Metrics	✓	✓	✓
Logging	✓	✓	✓
Zonal Failover	✓	✓	✓
Connection Draining (deregistration delay)	✓		✓
Load Balancing to multiple ports on the same instance	✓	✓	
IP addresses as targets	✓	✓ (TCP, TLS)	
Load balancer deletion protection	✓	✓	
Configurable idle connection timeout	✓		✓
Cross-zone load balancing	✓	✓	✓
Sticky sessions	✓	✓	✓
Static IP		✓	
Elastic IP address		✓	
Preserve Source IP address		✓	
Resource-based IAM permissions	✓	✓	✓
Tag-based IAM permissions	✓	✓	
Slow start	✓		
Web sockets	✓	✓	
PrivateLink Support		✓ (TCP, TLS)	



Pricing

- You are charged for each hour or partial hour that an Application Load Balancer is running and the number of Load Balancer Capacity Units (LCU) used per hour.
- You are charged for each hour or partial hour that a Network Load Balancer is running and the number of Load Balancer Capacity Units (LCU) used by Network Load Balancer per hour.
- You are charged for each hour or partial hour that a Classic Load Balancer is running and for each GB of data transferred through your load balancer.

Sources:
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html
https://docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html
https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/introduction.html
https://aws.amazon.com/elasticloadbalancing/features/
https://aws.amazon.com/elasticloadbalancing/pricing/?nc=sn&loc=3

Amazon Route 53

- A highly available and scalable Domain Name System (DNS) web service used for domain registration, DNS routing, and health checking.

Key Features

- Resolver
- Traffic flow
- Latency based routing
- Geo DNS
- Private DNS for Amazon VPC
- DNS Failover
- Health Checks and Monitoring
- Domain Registration
- CloudFront and S3 Zone Apex Support
- Amazon ELB Integration

Domain Registration

- Choose a domain name and confirm that it's available, then register the domain name with Route 53. The service automatically makes itself the DNS service for the domain by doing the following:
    - Creates a hosted zone that has the same name as your domain.
    - Assigns a set of four name servers to the hosted zone. When someone uses a browser to access your website, such as www.example.com, these name servers tell the browser where to find your resources, such as a web server or an S3 bucket.
    - Gets the name servers from the hosted zone and adds them to the domain.
- If you already registered a domain name with another registrar, you can choose to transfer the domain registration to Route 53.
- Enable DNSSEC signing on new or existing public hosted zones. DNSSEC provides data integrity verification and
  data origin authentication for DNS.
- You can also configure DNSSEC on domain registration.

Routing Internet Traffic to your Website or Web Application

- Use the Route 53 console to register a domain name and configure Route 53 to route internet traffic to your website or web application.
- After you register your domain name, Route 53 automatically creates a **public hosted zone** that has the same name as the domain.
- To route traffic to your resources, you create **records**, also known as *resource record sets*, in your hosted zone.

- You can create special Route 53 records, called **alias records**, that route traffic to S3 buckets, CloudFront distributions, and other AWS resources.
- Each record includes information about how you want to route traffic for your domain, such as:
    - Name - name of the record corresponds with the domain name or subdomain name that you want Route 53 to route traffic for.
    - Type - determines the type of resource that you want traffic to be routed to.
    - Value

Route 53 Health Checks

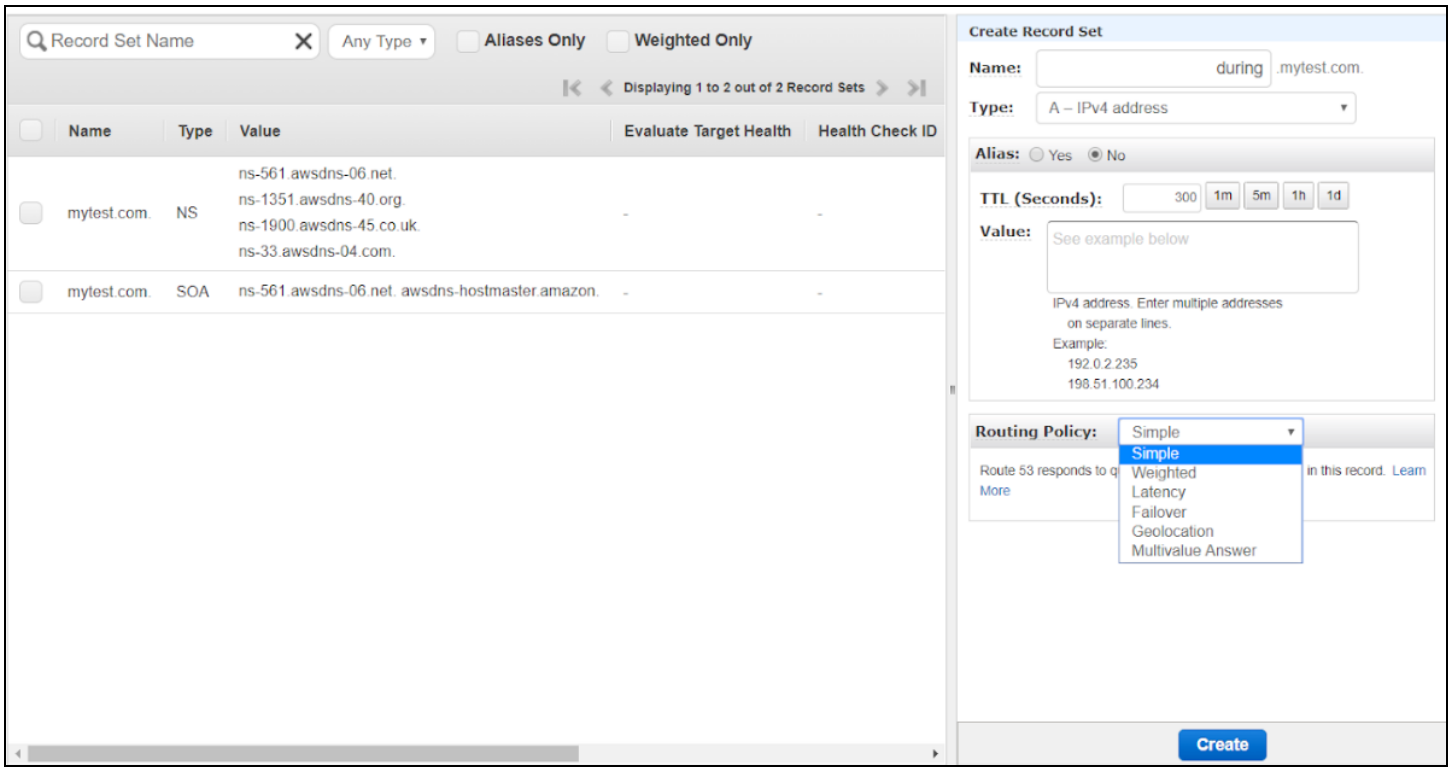
- Create a health check and specify values that define how you want the health check to work, such as:
    - The IP address or domain name of the endpoint that you want Route 53 to monitor.
    - The protocol that you want Route 53 to use to perform the check: HTTP, HTTPS, or TCP.
    - The **request interval** you want Route 53 to send a request to the endpoint.
    - How many consecutive times the endpoint must fail to respond to requests before Route 53 considers it unhealthy. This is the **failure threshold**.
- You can configure a health check to check the health of one or more other health checks.
- You can configure a health check to check the status of a CloudWatch alarm so that you can be notified on the basis of a broad range of criteria.

Know the following Concepts

- Domain Registration Concepts - domain name, domain registrar, domain registry, domain reseller, top-level domain
- DNS Concepts
    - **Alias record** - a type of record that you can create to route traffic to AWS resources.
    - DNS query
    - DNS resolver
    - Domain Name System (DNS)
    - Private DNS
    - **Hosted zone** - a container for records, which includes information about how to route traffic for a domain and all of its subdomains.
    - **Name servers** - servers in the DNS that help to translate domain names into the IP addresses that computers use to communicate with one another.
    - **Record** (DNS record) - an object in a hosted zone that you use to define how you want to route traffic for the domain or a subdomain.
    - **Routing policy**
    - **Subdomain**
    - Time to live (TTL)
- Health Checking Concepts
    - **DNS failover** - a method for routing traffic away from unhealthy resources and to healthy resources.

- endpoint
- health check

Routing Policies



- **Simple routing policy** – route internet traffic to a single resource that performs a given function for your domain. You can't create multiple records that have the same name and type, but you can specify multiple values in the same record, such as multiple IP addresses.
- **Failover routing policy** – use when you want to configure active-passive failover.
- **Geolocation routing policy** – use when you want to route internet traffic to your resources based on the location of your users.
- **Geoproximity routing policy** – use when you want to route traffic based on the location of your resources and, optionally, shift traffic from resources in one location to resources in another.
    - You can also optionally choose to route more traffic or less to a given resource by specifying a value, known as a ***bias***. A bias expands or shrinks the size of the geographic region from which traffic is routed to a resource.
    - The effect of changing the bias for your resources depends on a number of factors, including the following:
        - The number of resources that you have.
        - How close the resources are to one another.
        - The number of users that you have near the border area between geographic regions.

- **Latency routing policy** – use when you have resources in multiple locations and you want to route traffic to the resource that provides the best latency.
- **Multivalue answer routing policy** – use when you want Route 53 to respond to DNS queries with up to eight healthy records selected at random.
- **Weighted routing policy** – use to route traffic to multiple resources in proportions that you specify.
- When you register a domain or transfer domain registration to Route 53, it configures the domain to renew automatically. The automatic renewal period is typically one year, although the registries for some top-level domains (TLDs) have longer renewal periods.
- When you register a domain with Route 53, it creates a hosted zone that has the same name as the domain, assigns four name servers to the hosted zone, and updates the domain to use those name servers.

Hosted Zones

- Route 53 automatically creates the Name Server (NS) and Start of Authority (SOA) records for the hosted zones.
- Route 53 creates a set of 4 unique name servers (a delegation set) within each hosted zone.
- Public hosted zone - route internet traffic to your resources
- Private hosted zone - route traffic within an Amazon VPC. You create a private hosted zone, and specify the VPCs that you want to associate with the hosted zone.
    - To use private hosted zones, you must set the following VPC settings to true:
        - *enableDnsHostnames*
        - enableDnsSupport
    - In a private hosted zone, you can associate Route 53 health checks only with weighted and failover records.
    - You can use the following routing policies when you create records in a private hosted zone:
        - Simple
        - Failover
        - Multivalue answer
        - Weighted

Records

- Create records in a hosted zone. Records define where you want to route traffic for each domain name or subdomain name. The name of each record in a hosted zone must end with the name of the hosted zone.
- Alias Records
    - Route 53 **alias records** provide a Route 53–specific extension to DNS functionality. Alias records let you route traffic to selected AWS resources. They also let you route traffic from one record in a hosted zone to another record.

    - You can create an alias record at the top node of a DNS namespace, also known as the zone apex.
- CNAME Record
    - You cannot create an alias record at the top node of a DNS namespace using a CNAME record.
- Alias records vs CNAME records

CNAME Records	Alias Records
You can't create a CNAME record at the zone apex.	You can create an alias record at the zone apex. Alias records must have the same type as the record you're routing traffic to.
Route 53 charges for CNAME queries.	Route 53 doesn't charge for alias queries to AWS resources.
A CNAME record redirects queries for a domain name regardless of record type.	Route 53 responds to a DNS query only when the name and type of the alias record matches the name and type in the query.
A CNAME record can point to any DNS record that is hosted anywhere.	An alias record can only point to selected AWS resources or to another record in the hosted zone that you're creating the alias record in.
A CNAME record appears as a CNAME record in response to dig or Name Server (NS) lookup queries.	An alias record appears as the record type that you specified when you created the record, such as A or AAAA.

Supported DNS Record Types

- **A Record Type** - the value for an A record is an IPv4 address in dotted decimal notation.
- **AAAA Record Type** - the value for a AAAA record is an IPv6 address in colon-separated hexadecimal format.
- CAA Record Type - lets you specify which certificate authorities (CAs) are allowed to issue certificates for a domain or subdomain.
- **CNAME Record Type** - a CNAME Value element is the same format as a domain name.
- **MX Record Type** - each value for an MX record actually contains two values, *priority* and *domain name*.
- NAPTR Record Type
- **NS Record Type** - identifies the name servers for the hosted zone. The value for an NS record is the domain name of a name server.
- PTR Record Type - is the same format as a domain name.
- **SOA Record Type** - provides information about a domain and the corresponding Amazon Route 53 hosted zone.
- SPF Record Type

- SRV Record Type
- TXT Record Type

DNS Domain Name Format

- Names of domains, hosted zones, and records consist of a series of labels separated by dots, which can be up to 63 bytes long. The total length of a domain name cannot exceed 255 bytes, including the dots.
- You can create hosted zones and records that include * in the name.

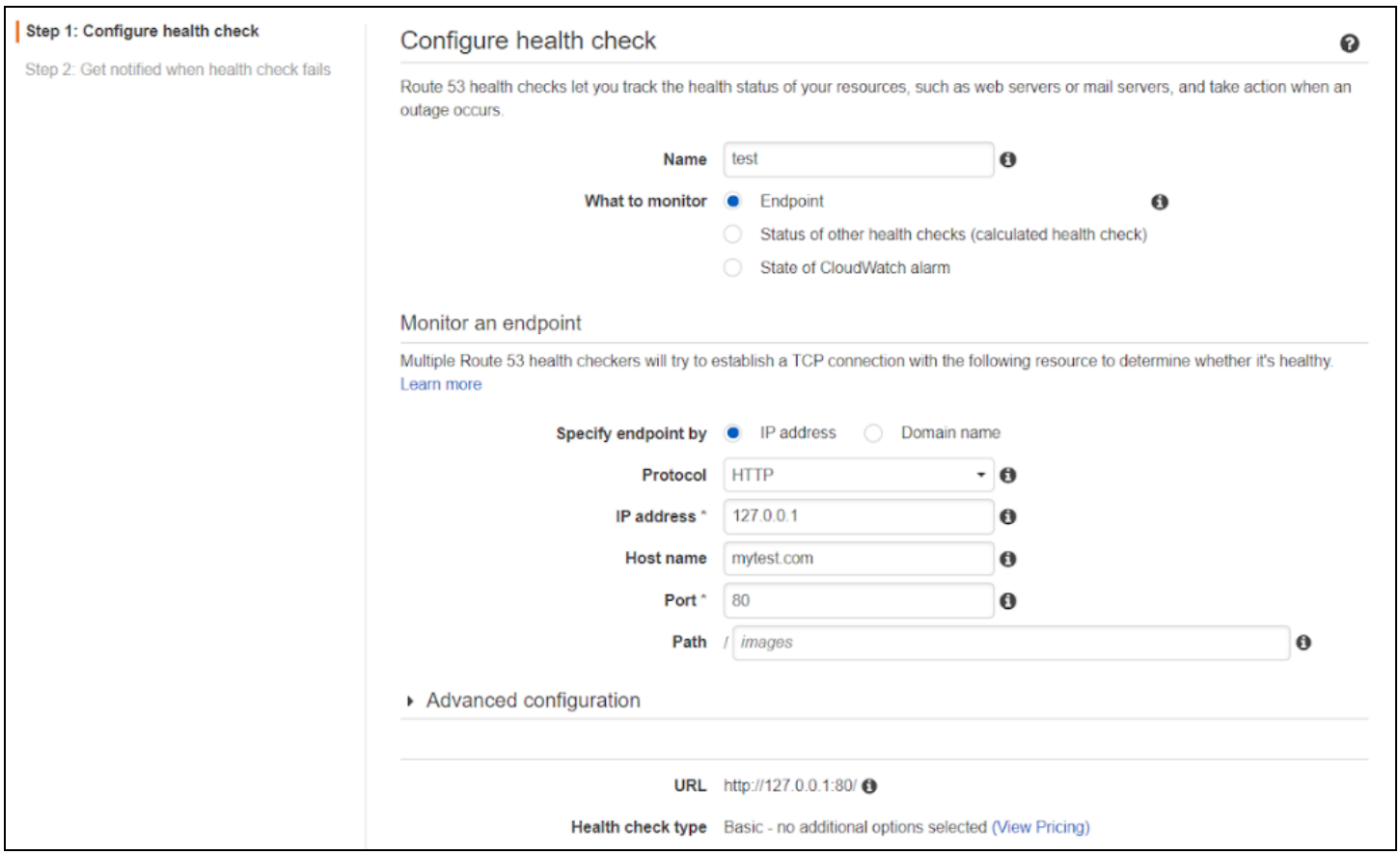
Using Traffic Flow to Route DNS Traffic

- You use the visual editor to create a **traffic policy**. A **traffic policy** includes information about the routing configuration that you want to create:
    - the routing policies that you want to use
    - resources that you want to route DNS traffic to, such as the IP address of each EC2 instance and the domain name of each ELB load balancer.
- Create a policy record where you specify the hosted zone in which you want to create the configuration that you defined in your traffic policy. It's also where you specify the DNS name that you want to associate the configuration with.

Route 53 Resolvers

- Resolver answers DNS queries for VPC domain names such as domain names for EC2 instances or ELB load balancers, and performs recursive lookups against public name servers for all other domain names.
- DNS resolvers on your network can forward DNS queries to Resolver in a specified VPC. You can also configure Resolver to forward queries that it receives from EC2 instances in your VPCs to DNS resolvers on your network.
- Resolver is **regional**.
- An **inbound endpoint** specifies the VPC that queries pass through on the way from your network to Resolver.
- To forward DNS queries that originate on EC2 instances in one or more VPCs to your network, you create an **outbound endpoint** and one or more rules.

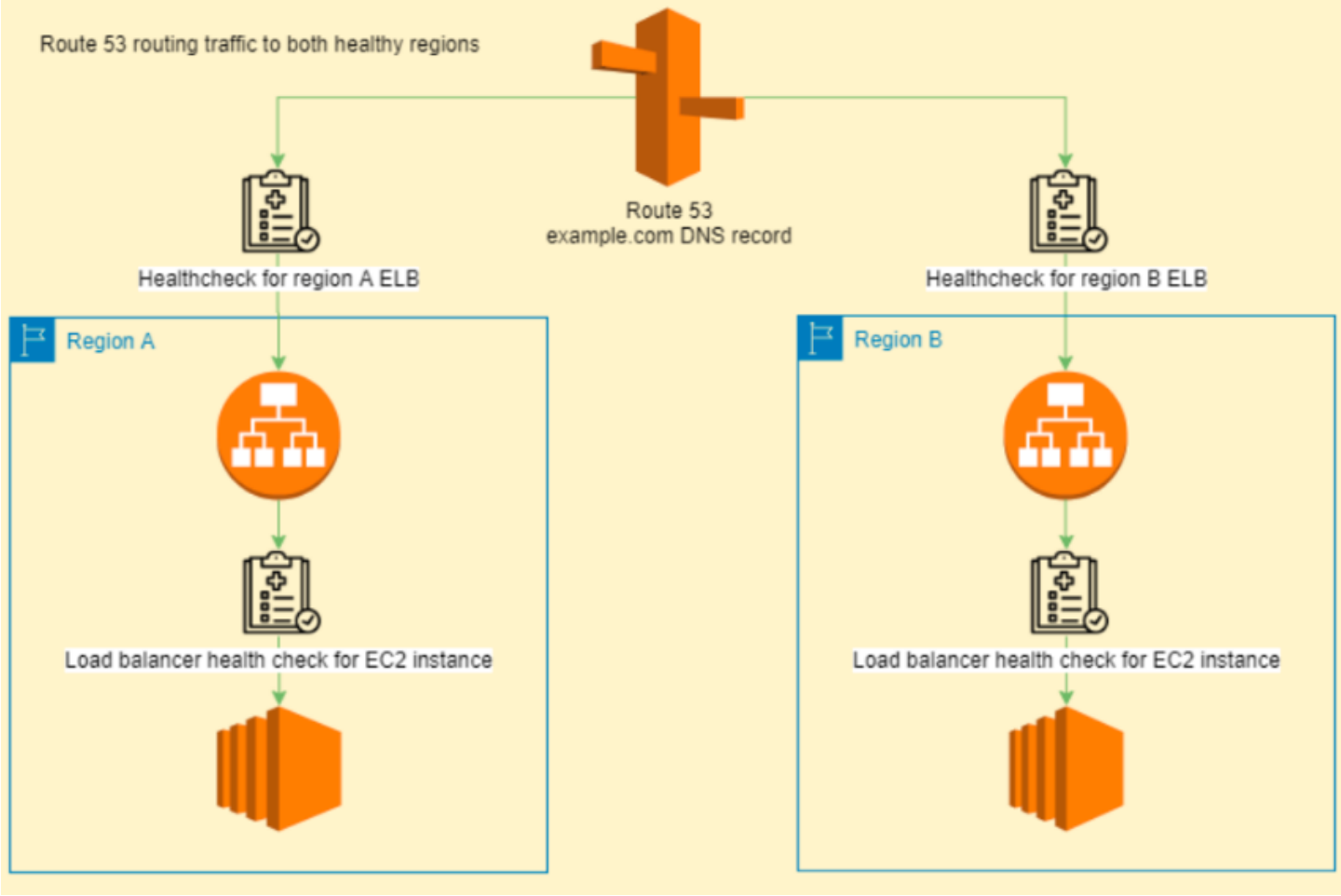
Route 53 Health Checks and DNS Failover

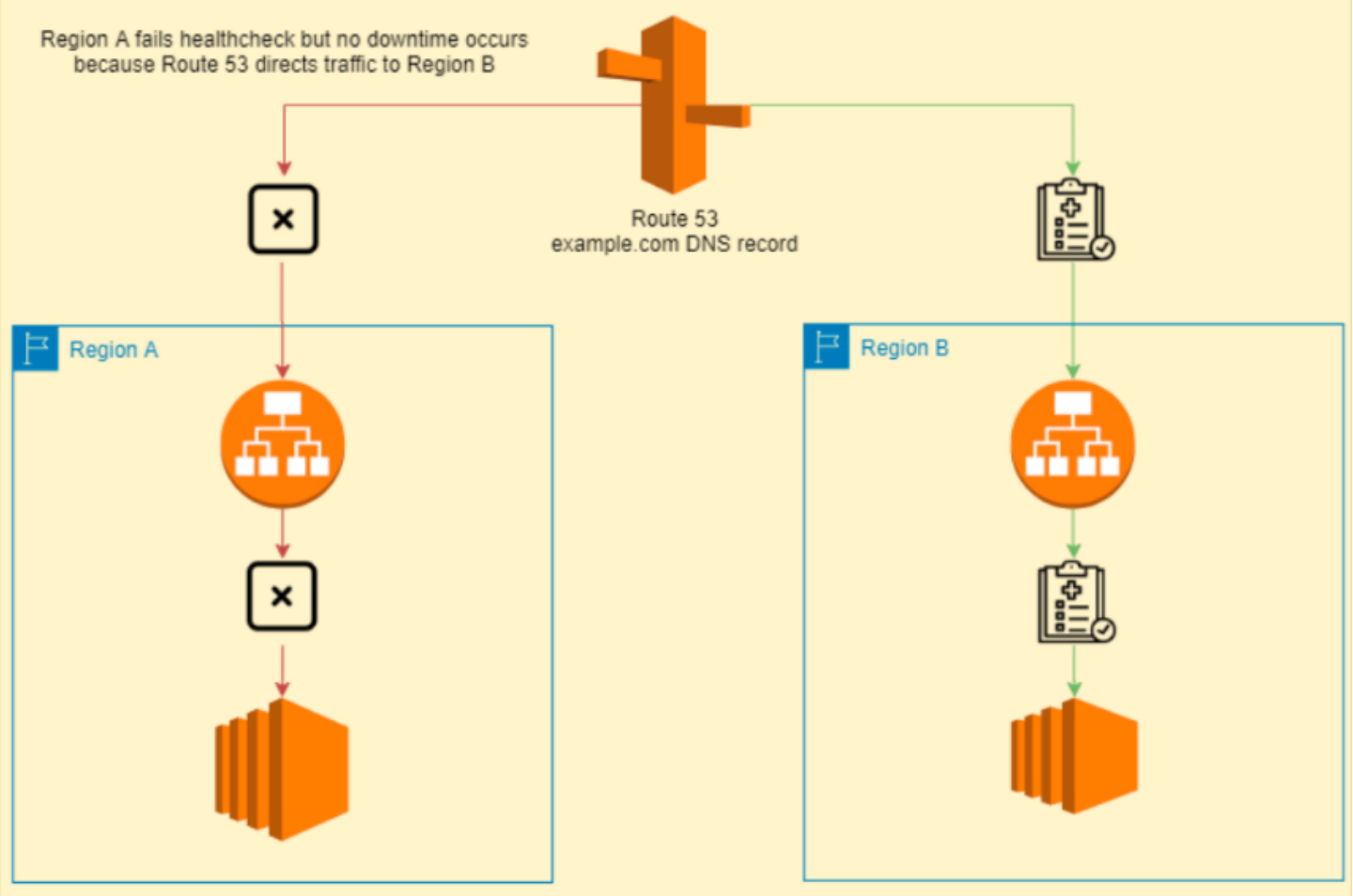


- Each health check that you create can monitor one of the following:
  - The health of a specified resource, such as a web server
  - The status of other health checks
  - The status of an Amazon CloudWatch alarm
- Each health checker evaluates the health of the endpoint based on two values:
  - Response time
  - Whether the endpoint responds to a number of consecutive health checks that you specify (the failure threshold)
- Types of health checks
  - **HTTP and HTTPS health checks** – Route 53 must be able to establish a TCP connection with the endpoint within four seconds. In addition, the endpoint must respond with an HTTP status code of 2xx or 3xx within two seconds after connecting.
  - **TCP health checks** – Route 53 must be able to establish a TCP connection with the endpoint within ten seconds.
  - **HTTP and HTTPS health checks with string matching** – Route 53 must be able to establish a TCP connection with the endpoint within four seconds, and the endpoint must respond with an HTTP status code of 2xx or 3xx within two seconds after connecting. After a Route 53 health

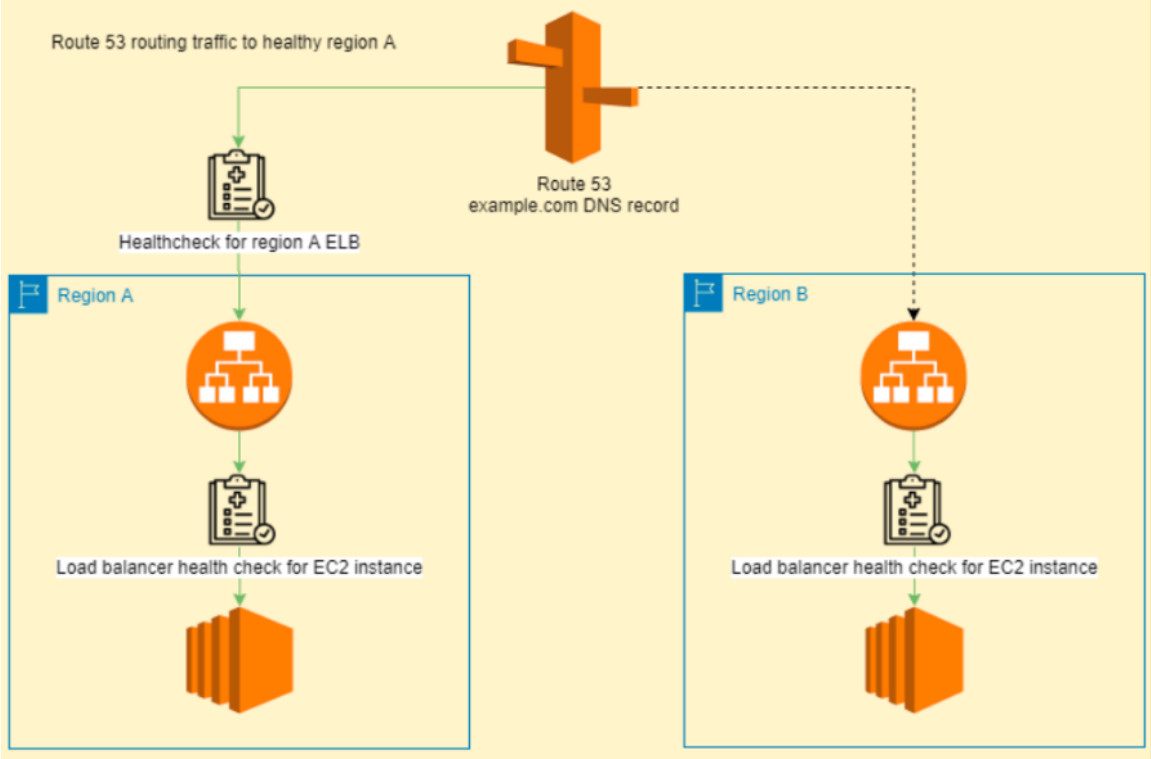
checker receives the HTTP status code, it must receive the response body from the endpoint within the next two seconds.

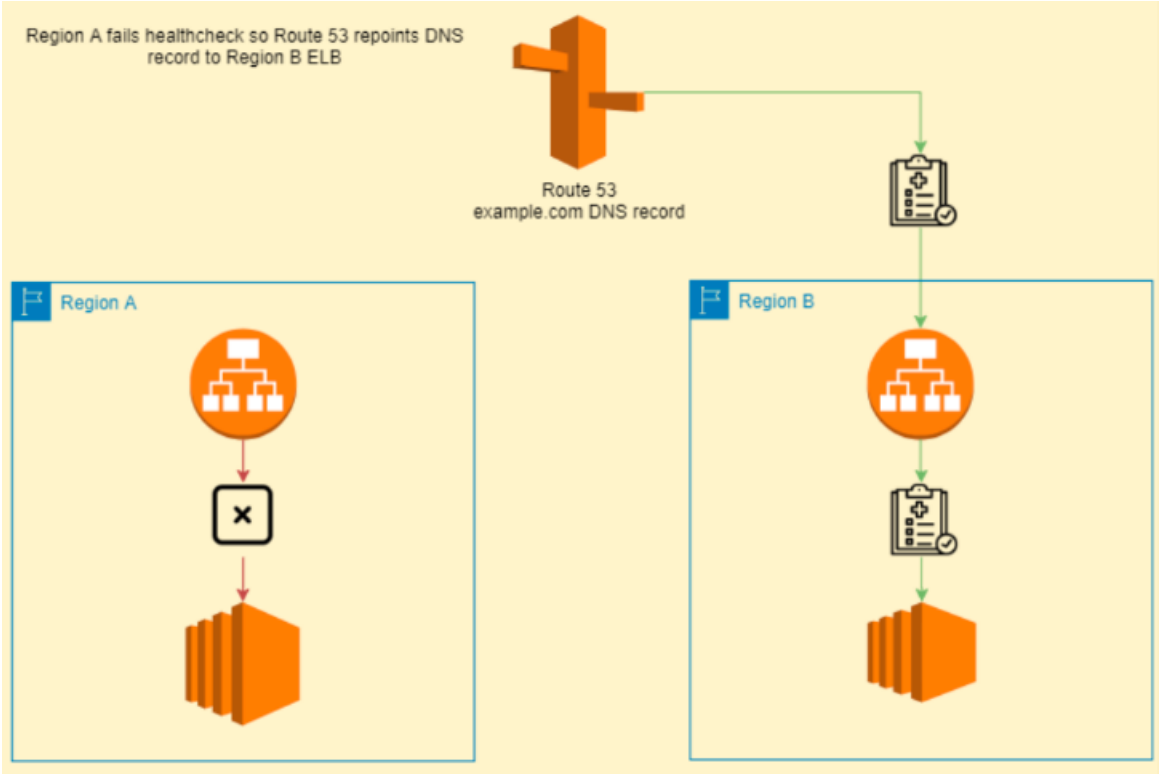
- Health-checking features to route traffic only to the healthy resources:
  - Check the health of EC2 instances and other resources (non-alias records)
  - Evaluate the health of an AWS resource (alias records)
- Two types of failover configurations
  - **Active-Active Failover** - all the records that have the same name, the same type, and the same routing policy are active unless Route 53 considers them unhealthy. Use this failover configuration when you want all of your resources to be available the majority of the time.





- **Active-Passive Failover** - use this failover configuration when you want a primary resource or group of resources to be available the majority of the time and you want a secondary resource or group of resources to be on standby in case all the primary resources become unavailable.


Route 53 routing traffic to healthy region A
Route 53
example.com DNS record
Healthcheck for region A ELB
Region A
Load balancer health check for EC2 instance
Region B
Load balancer health check for EC2 instance


Region A fails healthcheck so Route 53 repoints DNS record to Region B ELB
Route 53
example.com DNS record
Region A
Region B

    - When responding to queries, Route 53 includes only the healthy primary resources.
- To create an **active-passive failover configuration** with one primary record and one secondary record, you just create the records and specify **Failover for the routing policy**.
- To configure **active-passive failover** with multiple resources for the primary or secondary record, create records with the same name, type, and routing policy for your primary resources. If you're using AWS resources that you can create alias records for, specify **Yes** for **Evaluate Target Health**.
- You can also use weighted records for active-passive failover, with caveats.
- You can configure Amazon Route 53 to log information about the queries that Route 53 receives. Query logging is available only for public hosted zones.

Authentication and Access Control

- Authenticate with IAM before allowing to perform any operation on Route 53 resources.
- Every AWS resource is owned by an AWS account, and permissions to create or access a resource are governed by permissions policies. A *permissions policy* specifies who has access to what.

Monitoring

- The Route 53 dashboard provides detailed information about the status of your domain registrations, including:
    - Status of new domain registrations
    - Status of domain transfers to Route 53
    - List of domains that are approaching the expiration date
- You can use Amazon CloudWatch metrics to see the number of DNS queries served for each of your Route 53 public hosted zones. With these metrics, you can see at a glance the activity level of each hosted zone to monitor changes in traffic.
- You can monitor your resources by creating Route 53 health checks, which use CloudWatch to collect and process raw data into readable, near real-time metrics.
- Log API calls with CloudTrail

Pricing

- A hosted zone is charged at the time it's created and on the first day of each subsequent month. To allow testing, a hosted zone that is deleted within 12 hours of creation is not charged, however, any queries on that hosted zone will still incur charges.
- Billion queries / month
- Queries to Alias records are provided at no additional cost to current Route 53 customers when the records are mapped to the following AWS resource types:
    - Elastic Load Balancers
    - Amazon CloudFront distributions
    - AWS Elastic Beanstalk environments
    - Amazon S3 buckets that are configured as website endpoints

- Traffic flow policy record / month
- Pricing for domain names varies by Top Level Domain (TLD)

Sources:
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/Welcome.html
https://aws.amazon.com/route53/features/
https://aws.amazon.com/route53/pricing/

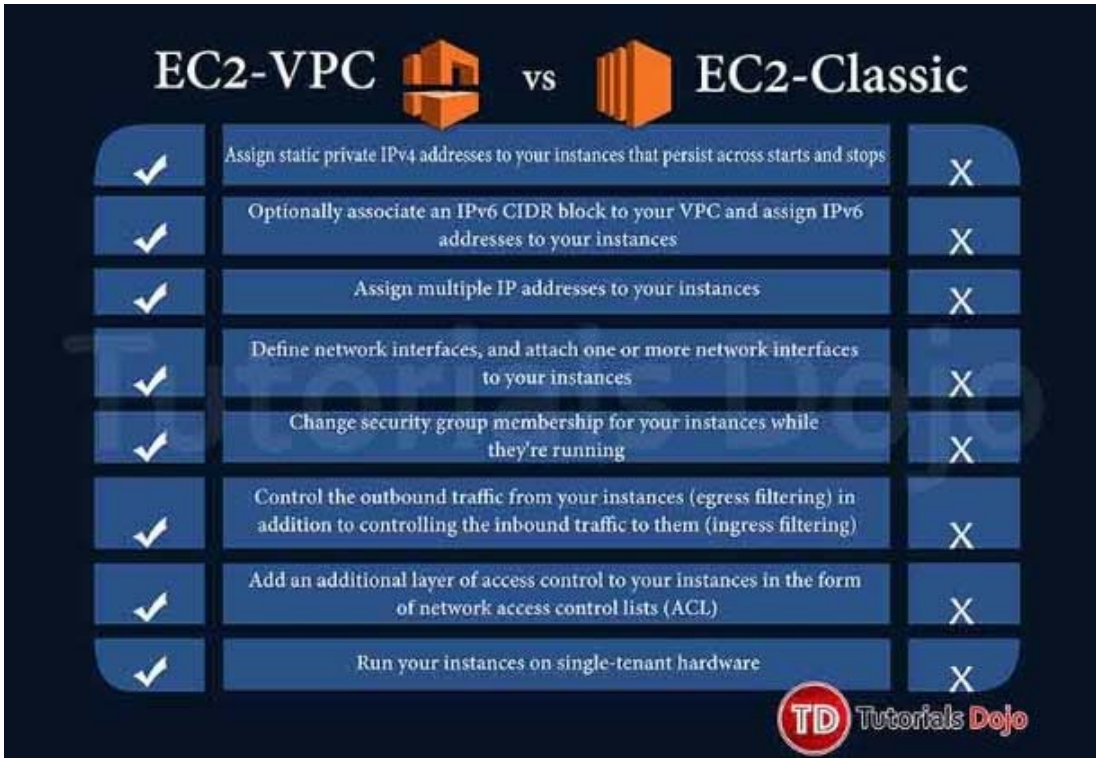
Amazon VPC

- Create a virtual network in the cloud dedicated to your AWS account where you can launch AWS resources
- Amazon VPC is the networking layer of Amazon EC2
- A VPC spans all the Availability Zones in the region. After creating a VPC, you can add one or more subnets in each Availability Zone.

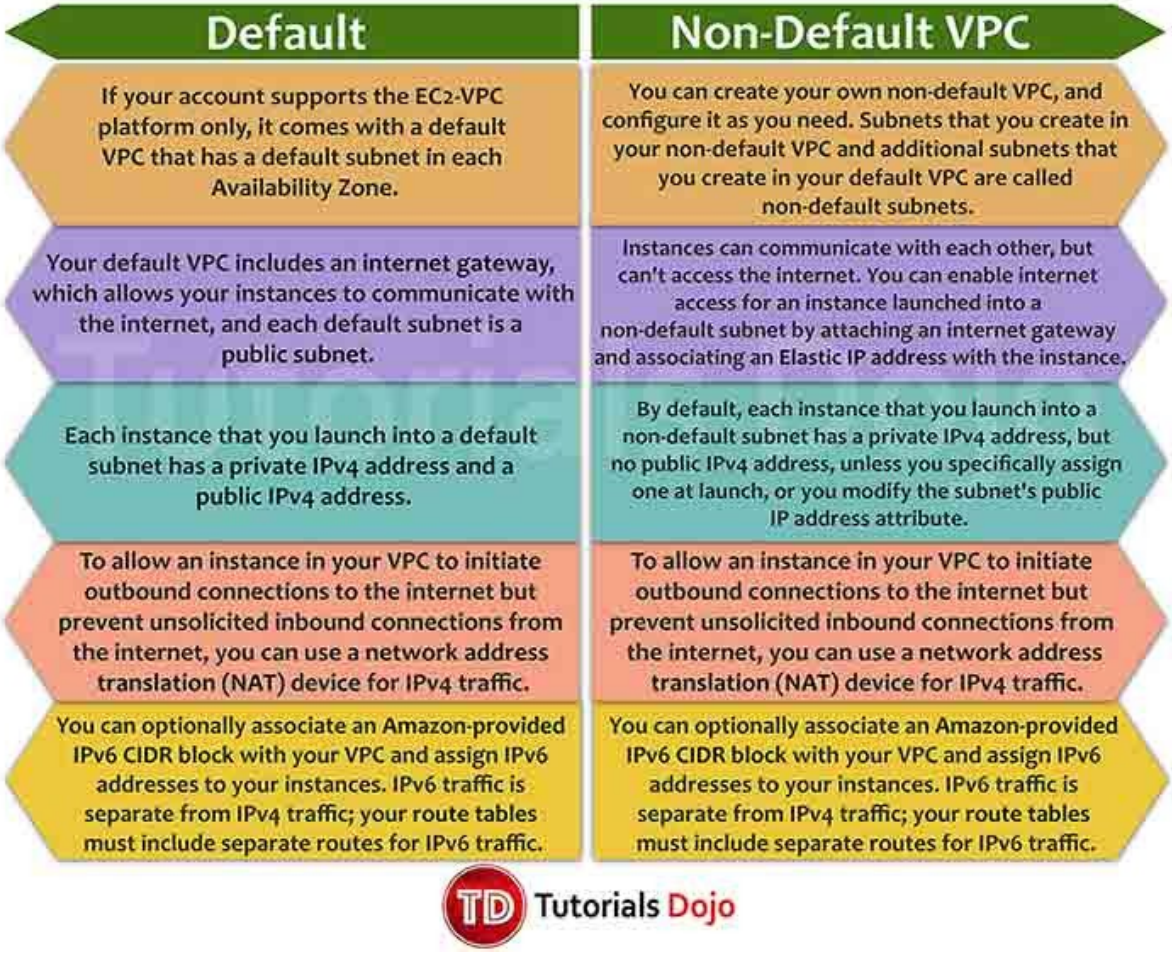
Key Concepts

- A **virtual private cloud** (VPC) allows you to specify an IP address range for the VPC, add subnets, associate security groups, and configure route tables.
- A **subnet** is a range of IP addresses in your VPC. You can launch AWS resources into a specified subnet. Use a **public subnet** for resources that must be connected to the internet, and a **private subnet** for resources that won't be connected to the internet.
- To protect the AWS resources in each subnet, use **security groups** and **network access control lists (ACL)**.
- Expand your VPC by adding secondary IP ranges.

EC2-VPC vs EC2-Classic



Default vs Non-Default VPC



Accessing a Corporate or Home Network

- You can optionally connect your VPC to your own corporate data center using an **IPsec AWS managed VPN connection**, making the AWS Cloud an extension of your data center.
- A **VPN connection** consists of:
    - a **virtual private gateway** (which is the VPN concentrator on the Amazon side of the VPN connection) attached to your VPC.
    - a **customer gateway** (which is a physical device or software appliance on your side of the VPN connection) located in your data center.
    - A diagram of the connection
- **AWS Site-to-Site Virtual Private Network** (VPN) connections can be moved from a virtual private gateway to an **AWS Transit Gateway** without having to make any changes on your customer gateway.

Transit Gateways enable you to easily scale connectivity across thousands of Amazon VPCs, AWS accounts, and on-premises networks.

- **AWS PrivateLink** enables you to privately connect your VPC to supported AWS services, services hosted by other AWS accounts (VPC endpoint services), and supported AWS Marketplace partner services. You do not require an internet gateway, NAT device, public IP address, AWS Direct Connect connection, or VPN connection to communicate with the service. Traffic between your VPC and the service does not leave the Amazon network.
- AWS PrivateLink-Supported Services:

- Amazon API Gateway
- Amazon AppStream 2.0
- AWS App Mesh
- Application Auto Scaling
- Amazon Athena
- AWS Auto Scaling
- Amazon Cloud Directory
- AWS CloudFormation
- AWS CloudTrail
- Amazon CloudWatch
- Amazon CloudWatch Events
- Amazon CloudWatch Logs
- AWS CodeBuild
- AWS CodeCommit
- AWS CodePipeline
- AWS Config
- AWS DataSync
- Amazon EC2 API
- Amazon EC2 Auto Scaling
- Amazon Elastic File System
- Elastic Load Balancing
- Amazon Elastic Container Registry
- Amazon Elastic Container Service
- AWS Glue
- AWS Key Management Service
- Amazon Kinesis Data Firehose
- Amazon Kinesis Data Streams
- Amazon Rekognition
- Amazon SageMaker and Amazon SageMaker Runtime
- Amazon SageMaker Notebook
- AWS Secrets Manager
- AWS Security Token Service
- AWS Server Migration Service
- AWS Service Catalog
- Amazon SNS
- Amazon SQS
- AWS Systems Manager
- AWS Storage Gateway
- AWS Transfer for SFTP
- Amazon WorkSpaces
- Endpoint services hosted by other AWS accounts
- Supported AWS Marketplace partner services

- You can create a **VPC peering connection** between your VPCs, or with a VPC in another AWS account, and enable routing of traffic between the VPCs using private IP addresses. You cannot create a VPC peering connection between VPCs that have overlapping CIDR blocks.
- Applications in an Amazon VPC can securely access AWS PrivateLink endpoints across VPC peering connections. The support of VPC peering by AWS PrivateLink makes it possible for customers to

privately connect to a service even if that service's endpoint resides in a different Amazon VPC that is connected using VPC peering.
- AWS PrivateLink endpoints can now be accessed across both intra- and inter-region VPC peering connections.

VPC Use Case Scenarios

- VPC with a Single Public Subnet
- VPC with Public and Private Subnets (NAT)
- VPC with Public and Private Subnets and AWS Managed VPN Access
- VPC with a Private Subnet Only and AWS Managed VPN Access

Subnets

- When you create a VPC, you must specify a range of IPv4 addresses for the VPC in the form of a Classless Inter-Domain Routing (CIDR) block (example: 10.0.0.0/16). This is the **primary CIDR block** for your VPC.
- You can add one or more subnets in each Availability Zone of your VPC's region.
- You specify the CIDR block for a subnet, which is a subset of the VPC CIDR block.
- A CIDR block must not overlap with any existing CIDR block that's associated with the VPC.
- Types of Subnets
    - Public Subnet - has an internet gateway
    - Private Subnet - doesn't have an internet gateway
    - VPN-only Subnet - has a virtual private gateway instead
- IPv4 CIDR block size should be between a /16 netmask (65,536 IP addresses) and /28 netmask (16 IP addresses).
- The **first four IP addresses and the last IP address in each subnet CIDR block** are **NOT available** for you to use, and cannot be assigned to an instance.
- You cannot increase or decrease the size of an existing CIDR block.
- When you associate a CIDR block with your VPC, a route is automatically added to your VPC route tables to enable routing within the VPC (the destination is the CIDR block and the target is *local*).
- You have a limit on the number of CIDR blocks you can associate with a VPC and the number of routes you can add to a route table.
- The following rules apply when you add IPv4 CIDR blocks to a VPC that's part of a **VPC peering connection**:
    - If the VPC peering connection is active, you can add CIDR blocks to a VPC provided they do not overlap with a CIDR block of the peer VPC.
    - If the VPC peering connection is pending-acceptance, the owner of the requester VPC cannot add any CIDR block to the VPC. Either the owner of the accepter VPC must accept the peering connection, or the owner of the requester VPC must delete the VPC peering connection request, add the CIDR block, and then request a new VPC peering connection.

    - If the VPC peering connection is pending-acceptance, the owner of the accepter VPC can add CIDR blocks to the VPC. If a secondary CIDR block overlaps with a CIDR block of the requester VPC, the VPC peering connection request fails and cannot be accepted.
- If you're using AWS Direct Connect to connect to multiple VPCs through a direct connect gateway, the VPCs that are associated with the direct connect gateway must not have overlapping CIDR blocks.
- The CIDR block is ready for you to use when it's in the *associated* state.
- You can disassociate a CIDR block that you've associated with your VPC; however, you cannot disassociate the primary CIDR block.

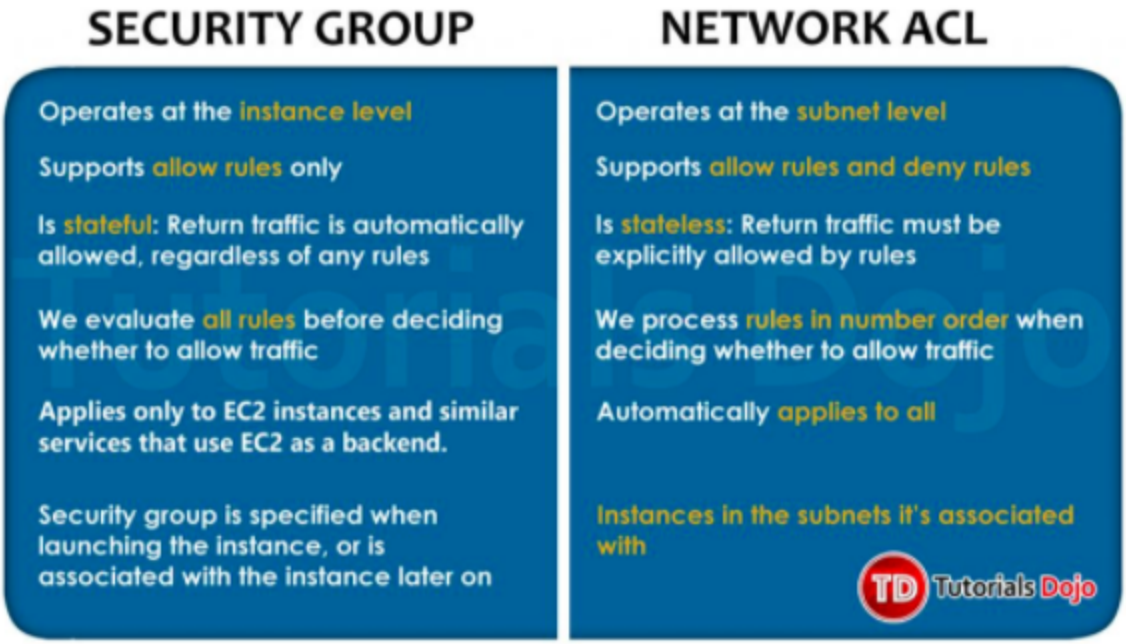
Subnet Routing

- Each subnet must be associated with a **route table**, which specifies the allowed routes for **outbound traffic** leaving the subnet.
- Every subnet that you create is automatically associated with the main route table for the VPC.
- You can change the association, and you can change the contents of the main route table.
- You can allow an instance in your VPC to initiate outbound connections to the internet over IPv4 but prevent unsolicited inbound connections from the internet using a **NAT gateway or NAT instance**.
- To initiate outbound-only communication to the internet over IPv6, you can use an egress-only internet gateway.

Subnet Security

- Security Groups — control inbound and outbound traffic for your instances
    - You can associate one or more (up to five) security groups to an instance in your VPC.
    - If you don't specify a security group, the instance automatically belongs to the default security group.
    - When you create a security group, it has no inbound rules. By default, it includes an outbound rule that allows all outbound traffic.
    - Security groups are associated with network interfaces.
- Network Access Control Lists — control inbound and outbound traffic for your subnets
    - Each subnet in your VPC must be associated with a network ACL. If none is associated, automatically associated with the default network ACL.
    - You can associate a network ACL with multiple subnets; however, a subnet can be associated with only one network ACL at a time.
    - A network ACL contains a numbered list of rules that is evaluated in order, starting with the lowest numbered rule, to determine whether traffic is allowed in or out of any subnet associated with the network ACL.
    - The default network ACL is configured to **allow all traffic to flow in and out** of the subnets to which it is associated.
- Flow logs — capture information about the IP traffic going to and from network interfaces in your VPC that is published to CloudWatch Logs.

- Flow logs can help you with a number of tasks, such as:
    - Diagnosing overly restrictive security group rules
    - Monitoring the traffic that is reaching your instance
    - Determining the direction of the traffic to and from the network interfaces
- Flow log data is collected outside of the path of your network traffic, and therefore does not affect network throughput or latency. You can create or delete flow logs without any risk of impact to network performance.
- After you've created a flow log, it can take several minutes to begin collecting and publishing data to the chosen destinations. Flow logs do not capture real-time log streams for your network interfaces.
- VPC Flow Logs can be sent directly to an Amazon S3 bucket which allows you to retrieve and analyze these logs yourself.
- Amazon security groups and network ACLs don't filter traffic to or from link-local addresses or AWS-reserved IPv4 addresses. Flow logs do not capture IP traffic to or from these addresses.



- Diagram of security groups and NACLs in a VPC

VPC Networking Components

- Network Interfaces
    - a virtual network interface that can include:
        - a primary private IPv4 address
        - one or more secondary private IPv4 addresses

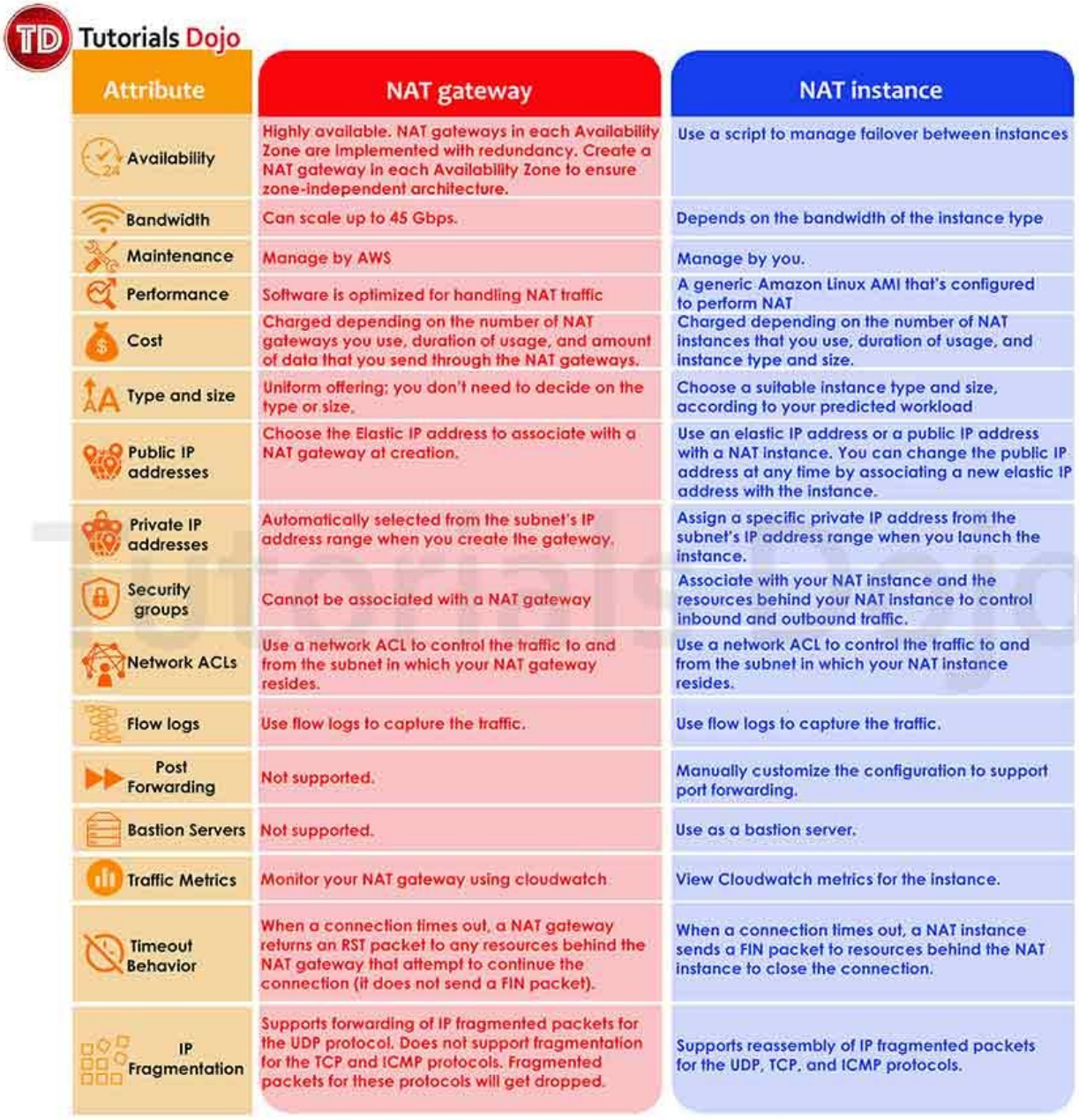
        - one Elastic IP address per private IPv4 address
        - one public IPv4 address, which can be auto-assigned to the network interface for eth0 when you launch an instance
        - one or more IPv6 addresses
        - one or more security groups
        - a MAC address
        - a source/destination check flag
        - a description
    - Network interfaces can be attached and detached from instances, however, you cannot detach a primary network interface.
- Route Tables
    - contains a set of rules, called *routes*, that are used to determine where network traffic is directed.
    - A subnet can only be associated with one route table at a time, but you can associate multiple subnets with the same route table.
    - You cannot delete the main route table, but you can replace the main route table with a custom table that you've created.
    - You must update the route table for any subnet that uses gateways or connections.
    - Uses the most specific route in your route table that matches the traffic to determine how to route the traffic (longest prefix match).
- Internet Gateways
    - Allows communication between instances in your VPC and the internet.
    - Imposes no availability risks or bandwidth constraints on your network traffic.
    - Provides a target in your VPC route tables for internet-routable traffic, and performs network address translation for instances that have been assigned public IPv4 addresses.
    - The following table provides an overview of whether your VPC automatically comes with the components required for internet access over IPv4 or IPv6.
    - To enable access to or from the Internet for instances in a VPC subnet, you must do the following:
        - Attach an Internet Gateway to your VPC
        - Ensure that your subnet's route table points to the Internet Gateway.
        - Ensure that instances in your subnet have a globally unique IP address (public IPv4 address, Elastic IP address, or IPv6 address).
        - Ensure that your network access control and security group rules allow the relevant traffic to flow to and from your instance

	Default VPC	Non-default VPC

Internet gateway	Yes	Yes, if you created the VPC using the first or second option in the VPC wizard. Otherwise, you must manually create and attach the internet gateway.
Route table with route to internet gateway for IPv4 traffic (0.0.0.0/0)	Yes	Yes, if you created the VPC using the first or second option in the VPC wizard. Otherwise, you must manually create the route table and add the route.
Route table with route to internet gateway for IPv6 traffic (::/0)	No	Yes, if you created the VPC using the first or second option in the VPC wizard, and if you specified the option to associate an IPv6 CIDR block with the VPC. Otherwise, you must manually create the route table and add the route.
Public IPv4 address automatically assigned to instance launched into subnet	Yes (default subnet)	No (non-default subnet)
IPv6 address automatically assigned to instance launched into subnet	No (default subnet)	No (non-default subnet)

- Egress-Only Internet Gateways
    - VPC component that allows outbound communication over IPv6 from instances in your VPC to the Internet, and prevents the Internet from initiating an IPv6 connection with your instances.
    - An egress-only Internet gateway is stateful.
    - You cannot associate a security group with an egress-only Internet gateway.
    - You can use a network ACL to control the traffic to and from the subnet for which the egress-only Internet gateway routes traffic.
- NAT
    - Enable instances in a private subnet to connect to the internet or other AWS services, but prevent the internet from initiating connections with the instances.
    - NAT Gateways
        - You must specify the **public subnet** in which the NAT gateway should reside.
        - You must specify an **Elastic IP address** to associate with the NAT gateway when you create it.
        - Each NAT gateway is created in a specific Availability Zone and implemented with redundancy in that zone.
        - Deleting a NAT gateway disassociates its Elastic IP address, but does not release the address from your account.
        - A NAT gateway supports the following protocols: TCP, UDP, and ICMP.

        - You cannot associate a security group with a NAT gateway.
        - A NAT gateway can support up to 55,000 simultaneous connections to each unique destination.
        - A NAT gateway cannot send traffic over VPC endpoints, VPN connections, AWS Direct Connect, or VPC peering connections.
        - A NAT gateway uses ports 1024-65535. Make sure to enable these in the inbound rules of your network ACL.
    - NAT Instance vs NAT Gateways



Attribute	NAT gateway	NAT instance
Availability	Highly available. NAT gateways in each Availability Zone are implemented with redundancy. Create a NAT gateway in each Availability Zone to ensure zone-independent architecture.	Use a script to manage failover between instances
Bandwidth	Can scale up to 45 Gbps.	Depends on the bandwidth of the instance type
Maintenance	Manage by AWS	Manage by you.
Performance	Software is optimized for handling NAT traffic	A generic Amazon Linux AMI that's configured to perform NAT
Cost	Charged depending on the number of NAT gateways you use, duration of usage, and amount of data that you send through the NAT gateways.	Charged depending on the number of NAT instances that you use, duration of usage, and instance type and size.
Type and size	Uniform offering; you don't need to decide on the type or size.	Choose a suitable instance type and size, according to your predicted workload
Public IP addresses	Choose the Elastic IP address to associate with a NAT gateway at creation.	Use an elastic IP address or a public IP address with a NAT instance. You can change the public IP address at any time by associating a new elastic IP address with the instance.
Private IP addresses	Automatically selected from the subnet's IP address range when you create the gateway.	Assign a specific private IP address from the subnet's IP address range when you launch the instance.
Security groups	Cannot be associated with a NAT gateway	Associate with your NAT instance and the resources behind your NAT instance to control inbound and outbound traffic.
Network ACLs	Use a network ACL to control the traffic to and from the subnet in which your NAT gateway resides.	Use a network ACL to control the traffic to and from the subnet in which your NAT instance resides.
Flow logs	Use flow logs to capture the traffic.	Use flow logs to capture the traffic.
Post Forwarding	Not supported.	Manually customize the configuration to support port forwarding.
Bastion Servers	Not supported.	Use as a bastion server.
Traffic Metrics	Monitor your NAT gateway using cloudwatch	View Cloudwatch metrics for the instance.
Timeout Behavior	When a connection times out, a NAT gateway returns an RST packet to any resources behind the NAT gateway that attempt to continue the connection (it does not send a FIN packet).	When a connection times out, a NAT instance sends a FIN packet to resources behind the NAT instance to close the connection.
IP Fragmentation	Supports forwarding of IP fragmented packets for the UDP protocol. Does not support fragmentation for the TCP and ICMP protocols. Fragmented packets for these protocols will get dropped.	Supports reassembly of IP fragmented packets for the UDP, TCP, and ICMP protocols.

- DHCP Options Sets
  - **Dynamic Host Configuration Protocol (DHCP)** provides a standard for passing configuration information to hosts on a TCP/IP network.
  - You can assign your own domain name to your instances, and use up to four of your own DNS servers by specifying a special set of DHCP options to use with the VPC.
  - Creating a VPC automatically creates a set of DHCP options, which are domain-name-servers=AmazonProvidedDNS, and domain-name=domain-name-for-your-region, and associates them with the VPC.

    - After you create a set of DHCP options, you can't modify them. Create a new set and associate a different set of DHCP options with your VPC, or use no DHCP options at all.
- DNS
    - AWS provides instances launched in a default VPC with public and private DNS hostnames that correspond to the public IPv4 and private IPv4 addresses for the instance.
    - AWS provides instances launched in a non-default VPC with private DNS hostname and possibly a public DNS hostname, depending on the DNS attributes you specify for the VPC and if your instance has a public IPv4 address.
    - Set VPC attributes *enableDnsHostnames* and *enableDnsSupport* to true so that your instances receive a public DNS hostname and Amazon-provided DNS server can resolve Amazon-provided private DNS hostnames.
        - If you use custom DNS domain names defined in a private hosted zone in Route 53, the *enableDnsHostnames* and *enableDnsSupport* attributes must be set to true.
- VPC Peering
    - A networking connection between two VPCs that enables you to route traffic between them privately. Instances in either VPC can communicate with each other as if they are within the same network.
- Elastic IP Addresses
    - A **static, public IPv4 address**.
    - You can associate an Elastic IP address with any instance or network interface for any VPC in your account.
    - You can mask the failure of an instance by rapidly remapping the address to another instance in your VPC.
    - Your Elastic IP addresses remain associated with your AWS account until you explicitly release them.
    - AWS imposes a small hourly charge when EIPs aren't associated with a running instance, or when they are associated with a stopped instance or an unattached network interface.
    - You're limited to five Elastic IP addresses.
- VPC Endpoints
    - Privately connect your VPC to supported AWS services and VPC endpoint services powered by PrivateLink without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection.
    - Endpoints are virtual devices.
    - Two Types
        - **Interface Endpoints**
            - An elastic network interface with a private IP address that serves as an entry point for traffic destined to a supported service.
            - Can be accessed through AWS VPN connections or AWS Direct Connect connections, through intra-region VPC peering connections from Nitro instances, and through inter-region VPC peering connections from any type of instance.

- For each interface endpoint, you can choose only one subnet per Availability Zone. Endpoints are supported within the same region only.
- You can add endpoint policies to interface endpoints. The Amazon VPC endpoint policy defines which principal can perform which actions on which resources. An endpoint policy does not override or replace IAM user policies or service-specific policies. It is a separate policy for controlling access from the endpoint to the specified service.
- An interface endpoint supports IPv4 TCP traffic only.

■ **Gateway Endpoints**

- A gateway that is a target for a specified route in your route table, used for traffic destined to a supported AWS service.
- You can create multiple endpoints in a single VPC, for example, to multiple services. You can also create multiple endpoints for a single service, and use different route tables to enforce different access policies from different subnets to the same service.
- You can modify the endpoint policy that's attached to your endpoint, and add or remove the route tables that are used by the endpoint.
- Endpoints are supported within the same region only. You cannot create an endpoint between a VPC and a service in a different region.
- Endpoints support IPv4 traffic only.
- You must enable DNS resolution in your VPC, or if you're using your own DNS server, ensure that DNS requests to the required service (such as S3) are resolved correctly to the IP addresses maintained by AWS.

○ You can create your own application in your VPC and configure it as an AWS PrivateLink-powered service (referred to as an *endpoint service*). You are the *service provider*, and the AWS principals that create connections to your service are *service consumers*.

- ClassicLink
  - Allows you to link an EC2-Classic instance to a VPC in your account, within the same region. This allows you to associate the VPC security groups with the EC2-Classic instance, enabling communication between your EC2-Classic instance and instances in your VPC using private IPv4 addresses.

VPN Connections

VPN connectivity option	Description
AWS managed VPN	You can create an IPsec VPN connection between your VPC and your remote network. On the AWS side of the VPN connection, a *virtual private gateway* provides two VPN endpoints (tunnels) for automatic failover. You configure your *customer gateway* on the remote side of the VPN connection.

AWS VPN CloudHub	If you have more than one remote network, you can create multiple AWS managed VPN connections via your virtual private gateway to enable communication between these networks.
Third party software VPN appliance	You can create a VPN connection to your remote network by using an Amazon EC2 instance in your VPC that's running a third party software VPN appliance. AWS does not provide or maintain third party software VPN appliances; however, you can choose from a range of products provided by partners and open source communities.
AWS Direct Connect	You can also use AWS Direct Connect to create a dedicated private connection from a remote network to your VPC. You can combine this connection with an AWS managed VPN connection to create an IPsec-encrypted connection.

- Specify a private Autonomous System Number (ASN) for the virtual private gateway. If you don't specify an ASN, the virtual private gateway is created with the default ASN (64512). You cannot change the ASN after you've created the virtual private gateway.
- When you create a VPN connection, you must:
    - Specify the type of routing that you plan to use (static or dynamic)
    - Update the route table for your subnet
- If your VPN device supports Border Gateway Protocol (BGP), specify **dynamic routing** when you configure your VPN connection. If your device does not support BGP, specify **static routing**.
- VPG uses path selection to determine how to route traffic to your remote network. Longest prefix match applies.
- Each VPN connection has two tunnels, with each tunnel using a unique virtual private gateway public IP address. It is important to configure both tunnels for redundancy.

Pricing

- Charged for VPN Connection-hour
- Charged for each “NAT Gateway-hour" that your NAT gateway is provisioned and available.
- Data processing charges apply for each Gigabyte processed through the NAT gateway regardless of the traffic’s source or destination.
- You also incur standard AWS data transfer charges for all data transferred via the NAT gateway.
- Charges for unused or inactive Elastic IPs.

Sources:
https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html
https://aws.amazon.com/vpc/details/
https://aws.amazon.com/vpc/pricing/

https://aws.amazon.com/vpc/faqs/

AWS Transit Gateway

- A networking service that uses a hub and spoke model to enable customers to connect their on-premises data centers and their Amazon Virtual Private Clouds (VPCs) to a single gateway.
- With this service, customers only have to create and manage a single connection from the central gateway into each on-premises data center, remote office, or VPC across your network.
- If a new VPC is created, it is automatically connected to the Transit Gateway and will also be available to every other network that is also connected to the Transit Gateway.

Features

- **Inter-region peering**
  - Transit Gateway leverages the AWS global network to allow customers to route traffic across AWS Regions.
  - Inter-region peering provides an easy and cost-effective way to replicate data for geographic redundancy or to share resources between AWS Regions.
- **Multicast**
  - Enables customers to have fine-grain control on who can consume and produce multicast traffic.
  - It allows you to easily create and manage multicast groups in the cloud instead of the time-consuming task of deploying and managing legacy hardware on-premises.
  - This multicast solution is also scalable so the customers can simultaneously distribute a stream of content to multiple subscribers.
- **Automated Provisioning**
  - Customers can automatically identify the Site-to-Site VPN connections and the on-premises resources with which they are associated using AWS Transit Gateway.
  - Using the Transit Gateway Network Manager, you can also manually define your on-premises network.

Source:
https://aws.amazon.com/transit-gateway/

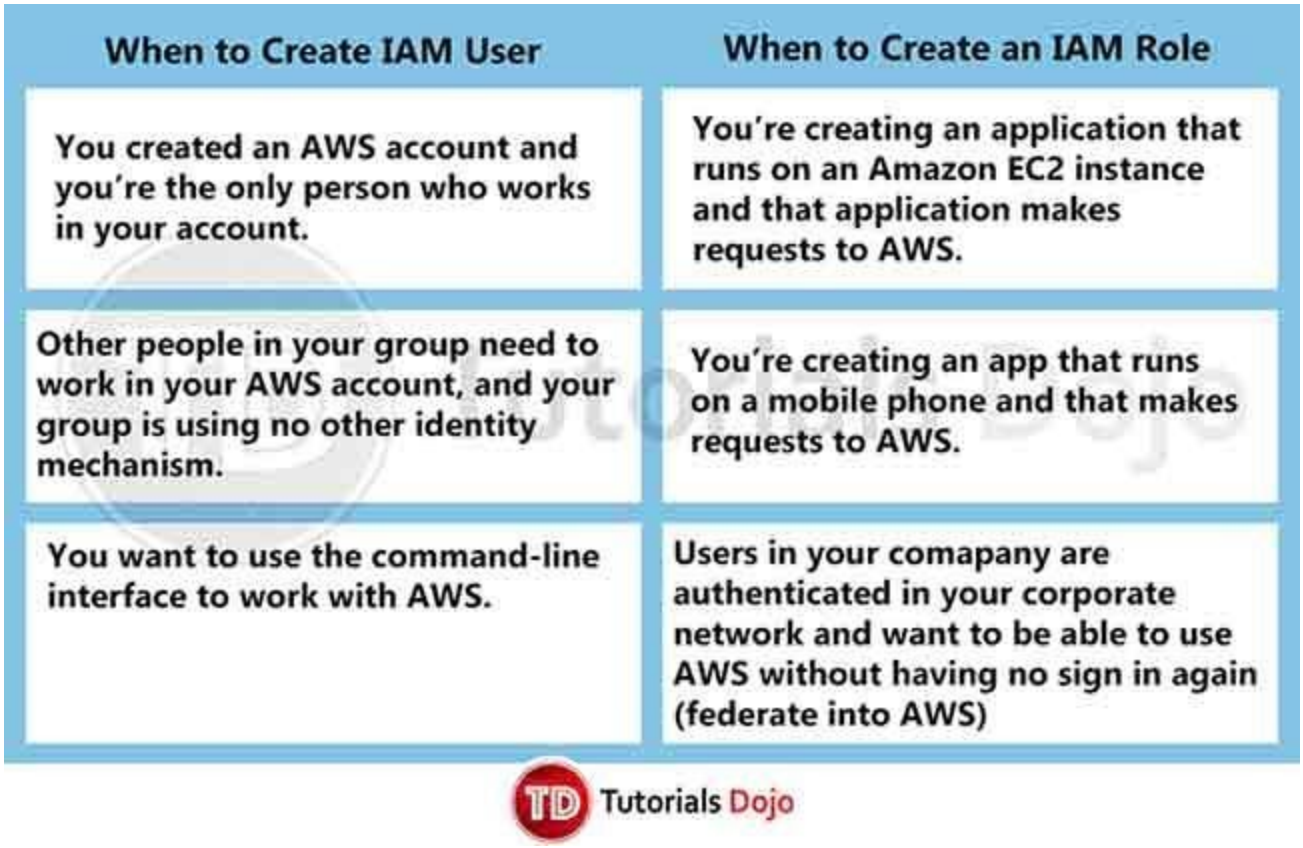
SECURITY AND IDENTITY

AWS Identity and Access Management (IAM)

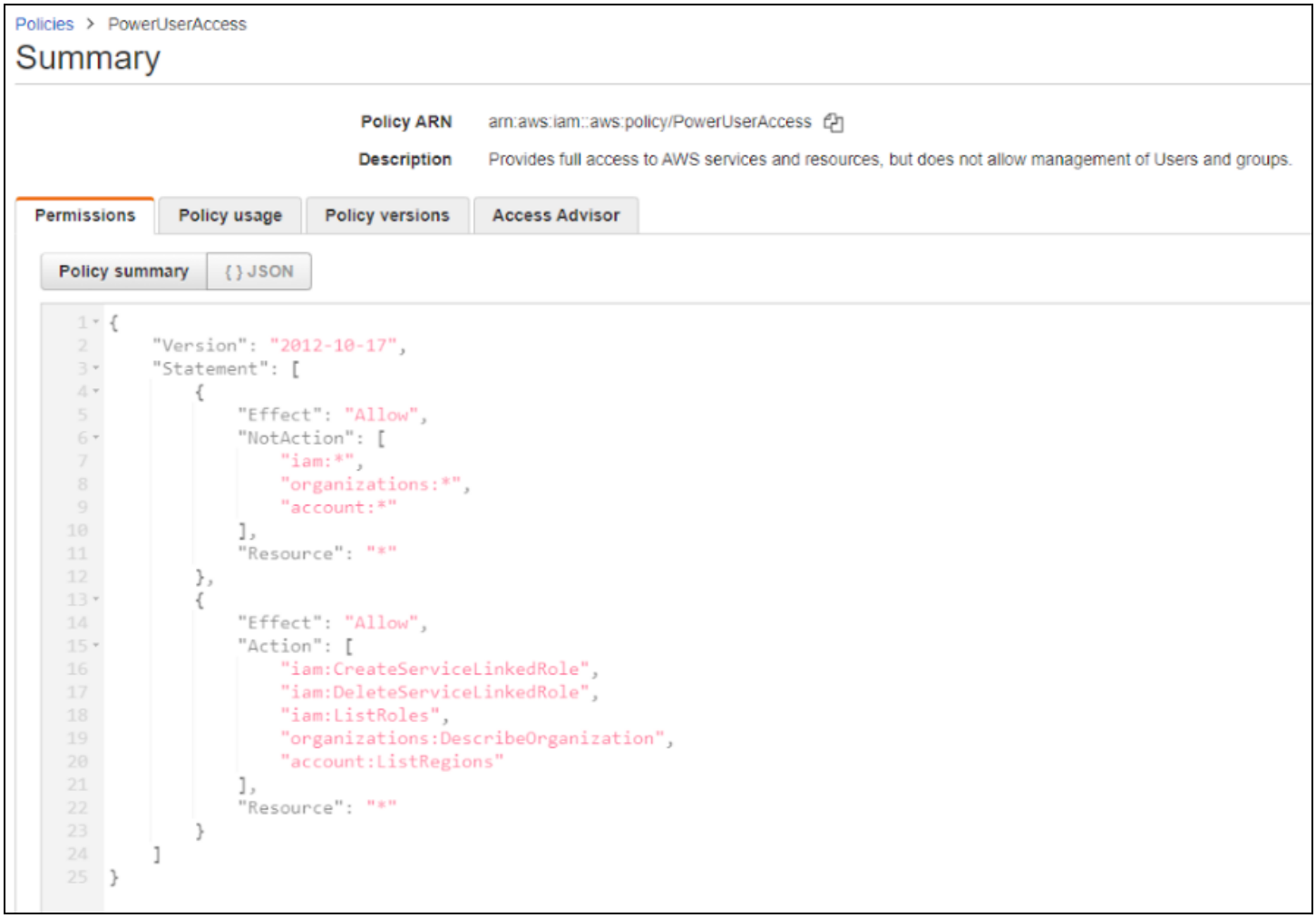
- Control who is authenticated (signed in) and authorized (has permissions) to use resources.
- AWS account **root user** is a single sign-in identity that has complete access to all AWS services and resources in the account.
- **Features**
    - You can grant other people permission to administer and use resources in your AWS account without having to share your password or access key.
    - You can grant different permissions to different people for different resources.
    - You can use IAM features to securely provide credentials for applications that run on EC2 instances which provide permissions for your applications to access other AWS resources.
    - You can add two-factor authentication to your account and to individual users for extra security.
    - You can allow users to use **identity federation** to get temporary access to your AWS account.
    - You receive AWS CloudTrail log records that include information about **IAM identities** who made requests for resources in your account.
    - You use an **access key** (an access key ID and secret access key) to make programmatic requests to AWS. An Access Key ID and Secret Access Key can only be uniquely generated once and must be regenerated if lost.
    - IAM has been validated as being compliant with Payment Card Industry (PCI) Data Security Standard (DSS).
    - IAM is *eventually consistent*. IAM achieves high availability by replicating data across multiple servers within Amazon's data centers around the world.
    - IAM and AWS Security Token Service (STS) are offered at no additional charge.
    - Your unique account sign-in page URL: https://*My_AWS_Account_ID*.signin.aws.amazon.com/console/
    - You can use IAM tags to add custom attributes to an IAM user or role using a tag key–value pair.
    - You can generate and download a credential report that lists all users on your AWS account. The report also shows the status of passwords, access keys, and MFA devices.
- **Infrastructure Elements**
    - **Principal**
        - An entity that can make a request for an action or operation on an AWS resource. Users, roles, federated users, and applications are all AWS principals.
        - Your AWS account root user is your *first principal*.
    - **Request**
        - When a principal tries to use the AWS Management Console, the AWS API, or the AWS CLI, that principal sends a *request* to AWS.
        - Requests includes the following information:

        - **Actions or operations** – the actions or operations that the principal wants to perform.
        - **Resources** – the AWS resource object upon which the actions or operations are performed.
        - **Principal** – the user, role, federated user, or application that sent the request. Information about the principal includes the policies that are associated with that principal.
        - **Environment data** – information about the IP address, user agent, SSL enabled status, or the time of day.
        - **Resource data** – data related to the resource that is being requested.
    - **Authentication**
        - To authenticate from the console as a user, you must sign in with your user name and password.
        - To authenticate from the API or AWS CLI, you must provide your access key and secret key.
    - **Authorization**
        - AWS uses values from the *request context* to check for policies that apply to the request. It then uses the policies to determine whether to allow or deny the request.
        - Policies types can be categorized as *permissions policies* or *permissions boundaries*.
            - *Permissions policies* define the permissions for the object to which they're attached. These include identity-based policies, resource-based policies, and ACLs.
            - *Permissions boundary* is an advanced feature that allows you to use policies to limit the maximum permissions that a principal can have.
        - To provide your users with permissions to access the AWS resources in their own account, you need **identity-based policies**.
        - **Resource-based policies** are for granting cross-account access.
        - Evaluation logic rules for policies:
            - By default, **all requests are denied**.
            - An *explicit allow* in a permissions policy overrides this default.
            - A *permissions boundary* overrides the allow. If there is a permissions boundary that applies, that boundary must allow the request. Otherwise, it is implicitly denied.
            - An explicit deny in any policy overrides any allows.
    - **Actions or Operations**
        - Operations are defined by a service, and include things that you can do to a resource, such as viewing, creating, editing, and deleting that resource.
    - **Resource**
        - An object that exists within a service. The service defines a set of actions that can be performed on each resource.
- **Users**

- **IAM Users**
  - Instead of sharing your root user credentials with others, you can create individual **IAM users** within your account that correspond to users in your organization. IAM users are not separate accounts; they are users within your account.
  - Each user can have its own password for access to the AWS Management Console. You can also create an individual access key for each user so that the user can make programmatic requests to work with resources in your account.
  - By default, a brand new IAM user has **NO permissions** to do anything.
  - Users are global entities.
- **Federated Users**
  - If the users in your organization already have a way to be authenticated, you can federate those user identities into AWS.
- **IAM Groups**
  - An IAM group is a collection of IAM users.
  - You can organize IAM users into IAM groups and attach access control policies to a group.
  - A user can belong to multiple groups.
  - Groups cannot belong to other groups.
  - Groups do not have security credentials, and cannot access web services directly.
- **IAM Role**
  - A role does not have any credentials associated with it.
  - An IAM user can assume a role to temporarily take on different permissions for a specific task. A role can be assigned to a federated user who signs in by using an external identity provider instead of IAM.
  - **AWS service role** is a role that a service assumes to perform actions in your account on your behalf. This service role must include all the permissions required for the service to access the AWS resources that it needs.
    - **AWS service role for an EC2 instance** is a special type of service role that a service assumes to launch an EC2 instance that runs your application. This role is assigned to the EC2 instance when it is launched.
    - **AWS service-linked role** is a unique type of service role that is linked directly to an AWS service. Service-linked roles are predefined by the service and include all the permissions that the service requires to call other AWS services on your behalf.
  - An instance profile is a container for an IAM role that you can use to pass role information to an EC2 instance when the instance starts.
- Users or groups can have multiple policies attached to them that grant different permissions.

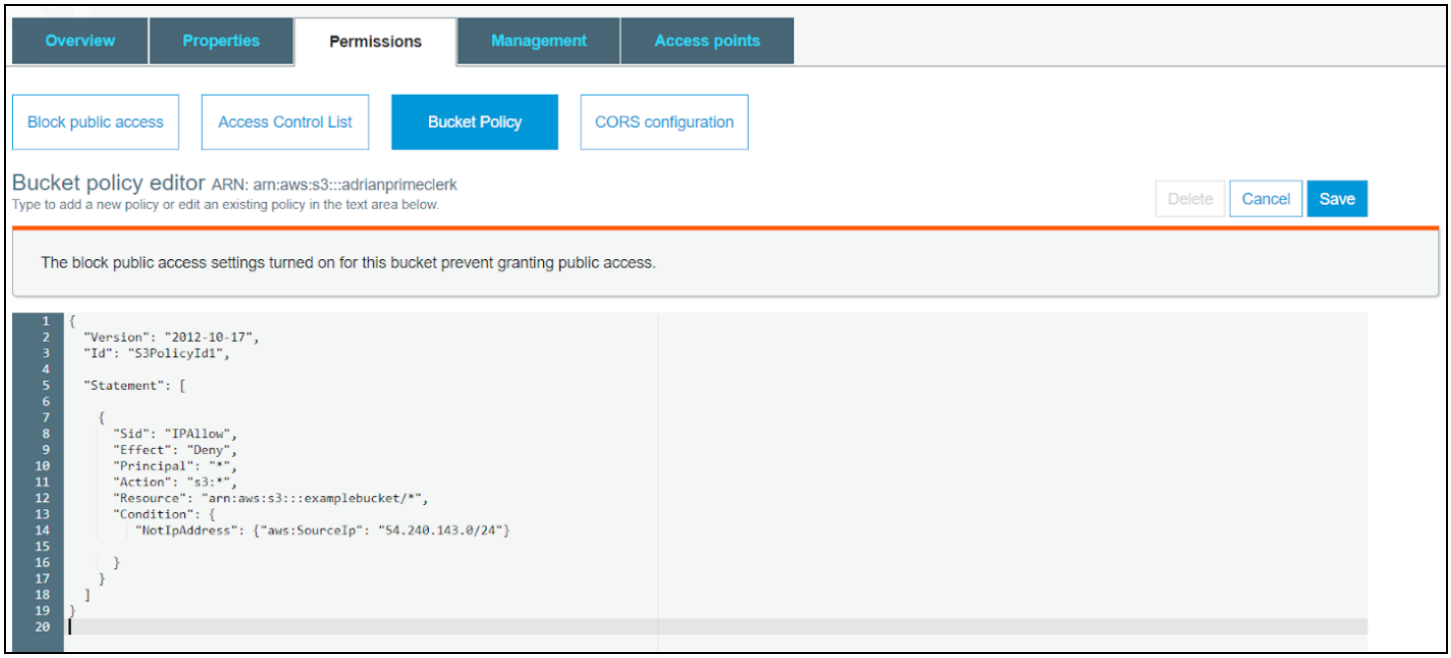


- **Policies**
  - Most permission policies are JSON policy documents.
  - The IAM console includes *policy summary tables* that describe the access level, resources, and conditions that are allowed or denied for each service in a policy.
  - The *policy summary table* includes a list of services. Choose a service there to see the *service summary*.
  - This *summary table* includes a list of the actions and associated permissions for the chosen service. You can choose an action from that table to view the action *summary*.
  - To assign permissions to federated users, you can create an entity referred to as a **role** and define permissions for the **role**.
  - **Identity-Based Policies**
    - Permissions policies that you attach to a principal or identity.
    - **Managed policies** are standalone policies that you can attach to multiple users, groups, and roles in your AWS account.
    - **Inline policies** are policies that you create and manage and that are embedded directly into a single user, group, or role.



Resource-based Policies

- Permissions policies that you attach to a resource such as an Amazon S3 bucket.
- Resource-based policies are only inline policies.
- **Trust policies** - resource-based policies that are attached to a role and define which principals can assume the role.



- **AWS Security Token Service (STS)**
  - Create and provide trusted users with temporary security credentials that can control access to your AWS resources.
  - Temporary security credentials are short-term and are not stored with the user but are generated dynamically and provided to the user when requested.
  - By default, AWS STS is a global service with a single endpoint at https://sts.amazonaws.com.
- Assume Role Options
  - AssumeRole - Returns a set of temporary security credentials that you can use to access AWS resources that you might not normally have access to. These temporary credentials consist of an access key ID, a secret access key, and a security token. Typically, you use *AssumeRole* within your account or for cross-account access.
    - You can include multi-factor authentication (MFA) information when you call *AssumeRole*. This is useful for cross-account scenarios to ensure that the user that assumes the role has been authenticated with an AWS MFA device.
  - AssumeRoleWithSAML - Returns a set of temporary security credentials for users who have been authenticated via a SAML authentication response. This allows you to link your enterprise identity store or directory to role-based AWS access without user-specific credentials or configuration.
  - AssumeRoleWithWebIdentity - Returns a set of temporary security credentials for users who have been authenticated in a mobile or web application with a web identity provider. Example providers include Amazon Cognito, Login with Amazon, Facebook, Google, or any OpenID Connect-compatible identity provider.
- STS Get Tokens

  - GetFederationToken - Returns a set of temporary security credentials (consisting of an access key ID, a secret access key, and a security token) for a federated user. You must call the GetFederationToken operation using the long-term security credentials of an IAM user. A typical use is in a proxy application that gets temporary security credentials on behalf of distributed applications inside a corporate network.
  - GetSessionToken - Returns a set of temporary credentials for an AWS account or IAM user. The credentials consist of an access key ID, a secret access key, and a security token. You must call the GetSessionToken operation using the long-term security credentials of an IAM user. Typically, you use GetSessionToken if you want to use MFA to protect programmatic calls to specific AWS API operations.
- **Best Practices**
  - Lock Away Your AWS Account Root User Access Keys
  - Create Individual IAM Users
  - Use Groups to Assign Permissions to IAM Users
  - Use AWS Defined Policies to Assign Permissions Whenever Possible
  - Grant Least Privilege
  - Use Access Levels to Review IAM Permissions
  - Configure a Strong Password Policy for Your Users
  - Enable MFA for Privileged Users
  - Use Roles for Applications That Run on Amazon EC2 Instances
  - Use Roles to Delegate Permissions
  - Do Not Share Access Keys
  - Rotate Credentials Regularly
  - Remove Unnecessary Credentials
  - Use Policy Conditions for Extra Security
  - Monitor Activity in Your AWS Account

Sources:
https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html
https://aws.amazon.com/iam/faqs/

AWS Key Management Service (AWS KMS)

- A managed service that enables you to easily encrypt your data. KMS provides a highly available key storage, management, and auditing solution for you to encrypt data within your own applications and control the encryption of stored data across AWS services.

Features

- KMS is integrated with CloudTrail, which provides you the ability to audit who used which keys, on which resources, and when.
- Customer master keys (CMKs) are used to control access to data encryption keys that encrypt and decrypt your data.
- You can choose to have KMS automatically rotate master keys created within KMS once per year without the need to re-encrypt data that has already been encrypted with your master key.
- To help ensure that your keys and your data is highly available, KMS stores multiple copies of encrypted versions of your keys in systems that are designed for 99.999999999% durability.

Concepts

- **Customer Master Keys (CMKs)** - You can use a CMK to encrypt and decrypt up to 4 KB of data. Typically, you use CMKs to generate, encrypt, and decrypt the data keys that you use outside of KMS to encrypt your data. Master keys are 256-bits in length.
- There are three types of CMKs:

Type of CMK	Can view	Can manage	Used only for my AWS account
Customer managed CMK	Yes	Yes	Yes
AWS managed CMK	Yes	No	Yes
AWS owned CMK	No	No	No

    - ***Customer managed CMKs*** are CMKs that you create, own, and manage. You have full control over these CMKs, including establishing and maintaining their key policies, IAM policies, and grants, enabling and disabling them, rotating their cryptographic material, adding tags, creating aliases that refer to the CMK, and scheduling the CMKs for deletion.

    - ***AWS managed CMKs*** are CMKs in your account that are created, managed, and used on your behalf by an AWS service that integrates with KMS. You can view the AWS managed CMKs in your account, view their key policies, and audit their use in CloudTrail logs. However, you cannot manage these CMKs or change their permissions. And, you cannot use AWS managed CMKs in cryptographic operations directly; the service that creates them uses them on your behalf.
    - ***AWS owned CMKs*** are not in your AWS account. They are part of a collection of CMKs that AWS owns and manages for use in multiple AWS accounts. AWS services can use AWS owned CMKs to protect your data. You cannot view, manage, or use AWS owned CMKs, or audit their use.
- **Data keys** - Encryption keys that you can use to encrypt data, including large amounts of data and other data encryption keys.
    - You can use CMKs to generate, encrypt, and decrypt data keys. However, KMS does not store, manage, or track your data keys, or perform cryptographic operations with data keys.
    - Data keys can be generated at 128-bit or 256-bit lengths and encrypted under a master key you define.
- **Envelope encryption** -The practice of encrypting plaintext data with a data key, and then encrypting the data key under another key. The top-level plaintext key encryption key is known as the *master key*.
- **Encryption Context** - All KMS cryptographic operations accept an encryption context, an optional set of key–value pairs that can contain additional contextual information about the data.
- **Key Policies** - When you create a CMK, permissions that determine who can use and manage that CMK are contained in a document called the key policy.
- **Grants** - A grant is an alternative to the key policy. You can use grants to give long-term access that allows AWS principals to use your customer managed CMKs.
- **Grant Tokens** - When you create a grant, the permissions specified in the grant might not take effect immediately due to eventual consistency. If you need to mitigate the potential delay, use a grant token instead.
- When you enable ***automatic key rotation*** for a customer managed CMK, KMS generates new cryptographic material for the CMK every year. KMS also saves the CMK's older cryptographic material so it can be used to decrypt data that it encrypted.
- An *alias* is an optional display name for a CMK. Each CMK can have multiple aliases, but each alias points to only one CMK. The alias name must be unique in the AWS account and region.

Importing Keys

- A CMK contains the **key material** used to encrypt and decrypt data. When you create a CMK, by default AWS KMS generates the key material for that CMK. But you can create a CMK without key material and then import your own key material into that CMK.
- When you import key material, you can specify an expiration date. When the key material expires, KMS deletes the key material and the CMK becomes unusable. You can also delete key material on demand.

Deleting Keys

- Deleting a CMK deletes the key material and all metadata associated with the CMK and is irreversible. You can no longer decrypt the data that was encrypted under that CMK, which means that data becomes unrecoverable.
- You can create a CloudWatch alarm that sends you a notification when a user attempts to use the CMK while it is pending deletion.
- You can temporarily disable keys so they cannot be used by anyone.
- KMS supports custom key stores backed by AWS CloudHSM clusters. A key store is a secure location for storing cryptographic keys.
- You can connect directly to AWS KMS through a private endpoint in your VPC instead of connecting over the internet. When you use a VPC endpoint, communication between your VPC and AWS KMS is conducted entirely within the AWS network.

Pricing

- Each customer master key that you create in KMS, regardless of whether you use it with KMS-generated key material or key material imported by you, costs you until you delete it.
- For a CMK with key material generated by KMS, if you opt-in to have the CMK automatically rotated each year, each newly rotated version will raise the cost of the CMK per month.

Sources:
https://docs.aws.amazon.com/kms/latest/developerguide
https://aws.amazon.com/kms/features/
https://aws.amazon.com/kms/pricing/
https://aws.amazon.com/kms/faqs/

Working with Customer Master Keys (CMKs) using the AWS KMS API

What is AWS Key Management Service?

AWS Key Management Service (or KMS for short) is the service you use to securely store your encryption keys in AWS. If you need data encryption on your AWS resources, such as EBS volumes or RDS databases, you can use AWS KMS to simplify the process for you. You start using the service by requesting the creation of a customer master key or CMK. By default, AWS KMS creates the key material for your CMK. You also have the option of importing your own keys to AWS if you wish to. Note that during key rotation, if you imported your own key, you will have to manage the rotation yourself.

Users and developers who manage security can interact with AWS KMS programmatically via the CLI or SDK. These utilize the AWS KMS API for all of the transactions. You also do not use your standard username and password when interacting with the KMS API, so be sure to enter your access keys and secret access keys instead. Once you have configured your AWS profile in your local machine, you can start executing API calls. If you encounter any errors during API calls, check if your IAM User has been granted the necessary permissions to perform that action.

List of Commonly Used AWS KMS APIs

Below are some of the KMS API commands that you should know of:

- Create, describe, list, enable, and disable CMK keys

To create a customer master key (CMK), run the CreateKey operation. By default, this command creates a symmetric CMK for you. Also, if the key is created via API, only the root user of the AWS account who owns this key has full access.

```
aws kms create-key
```

You can also create an asymmetric CMK if that is what you need. Here, you must specify the CustomerMasterKeySpec parameter, which determines the encryption algorithm that KMS will use. Also, you must specify a KeyUsage value of ENCRYPT_DECRYPT or SIGN_VERIFY. You cannot change these properties after the CMK is created.

```
awz kms create-key --customer-master-key-spec RSA_4096
--key-usage ENCRYPT_DECRYPT
```

```
aws kms create-key --customer-master-key-spec ECC_NIST_P521
--key-usage SIGN_VERIFY
```

You can also specify some additional parameters during key creation

- BypassPolicyLockoutSafetyCheck (bool) - Indicates whether to bypass the key policy lockout safety check.
- CustomKeyStoreId (string) - Creates the CMK in the specified custom key store and the key material in its associated AWS CloudHSM cluster.
- Description (string)
- Origin (string) - The source of the key material for the CMK. You cannot change the origin after you create the CMK. The default is AWS_KMS.
- Policy (string) - The key policy to attach to the CMK.
- Tags (dict)

To list all the CMKs under your account, run the ListKeys operation.

```
aws kms list-keys
```

To see the metadata and other details of a specific key, run the DescribeKey operation and enter the key id of interest.

```
aws kms describe-key --key-id
1234abcd-12ab-34cd-56ef-1234567890ab
```

If you want to set which keys should and should not be used, run the EnableKey and DisableKey operations and enter the key id of interest.

```
aws kms enable-key --key-id
1234abcd-12ab-34cd-56ef-1234567890ab
```

```
aws kms disable-key --key-id
1234abcd-12ab-34cd-56ef-1234567890ab
```

- Encrypt, decrypt, and re-encrypt content

To encrypt plaintext into ciphertext using your CMK, run Encrypt command. Enter the key id you'd like to use and the data you'd like to encrypt. After running it, the API will return your ciphertext in blob format, the key id used and the encryption algorithm used.

```
aws kms encrypt --key-id
1234abcd-12ab-34cd-56ef-1234567890ab --plaintext
fileb://ExamplePlaintextFile
```

You can specify additional parameters during encryption

- EncryptionAlgorithm (SYMMETRIC_DEFAULT | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256)
- EncryptionContext (dict)

To decrypt your ciphertext, run the Decrypt command and enter your ciphertext blob. If you used an asymmetric CMK to encrypt this text, then you need to specify a key id parameter. You should also specify the encryption algorithm and encryption context if the defaults were not used. After running it, the API will return your plaintext, key id used for decryption, and the encryption algorithm used.

```
aws kms decrypt --ciphertext-blob
fileb://ExampleCiphertextFile
```

You can specify the following additional parameters during decryption

- EncryptionAlgorithm (SYMMETRIC_DEFAULT | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256)
- EncryptionContext (dict)

If you will be rotating keys or changing encryption algorithms, run the ReEncrypt command to re-encrypt your data to a new CMK or algorithm. If you used an asymmetric key to encrypt your plaintext, you must specify the source key id used during encryption. If you used a non-default encryption algorithm and an encryption context, be sure to indicate them in the API call.

```
aws kms re-encrypt --ciphertext-blob
fileb://ExampleCiphertextFile --destination-key-id
1234abcd-12ab-34cd-56ef-1234567890ac
```

You can specify the following additional parameters during re-encryption

    - DestinationEncryptionAlgorithm (SYMMETRIC_DEFAULT | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256)
    - DestinationEncryptionContext (dict)
    - SourceEncryptionAlgorithm (SYMMETRIC_DEFAULT | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256)
    - SourceEncryptionContext (dict)
    - SourceKeyId (string)
- Set, list, and retrieve key policies

Key policies are the primary way to control access to CMKs in AWS KMS. To add a key policy to a CMK, run the PutKeyPolicy command. This API requires you to specify the key id to which the policy will be applied to, a policy name with "*default*" as the value and the KMS policy itself.

```
aws kms put-key-policy --key-id
1234abcd-12ab-34cd-56ef-1234567890ab --policy-name default
--policy file://key_policy.json
```

Example key_policy.json

```
{
    "Version" : "2012-10-17",
    "Id" : "key-default",
    "Statement" : [
        {
            "Sid" : "Enable IAM User Permissions",
            "Effect" : "Allow",
            "Principal" : {
                "AWS" : "arn:aws:iam::123456789000:root"
            },
            "Action" : "kms:",
            "Resource" : "*"
        }
    ]
}
```

To list down all the key policies attached to a CMK, run the ListKeyPolicies command. You must supply the key id of the CMK in the API call. The returned value will be a list of policy names.

```
aws kms list-key-policies --key-id
1234abcd-12ab-34cd-56ef-1234567890ab
```

To view the contents of a policy, run the GetKeyPolicy command. You must supply the key id of the CMK and the policy name in the API call.

```
aws kms get-key-policy --key-id
1234abcd-12ab-34cd-56ef-1234567890ab --policy-name default
```

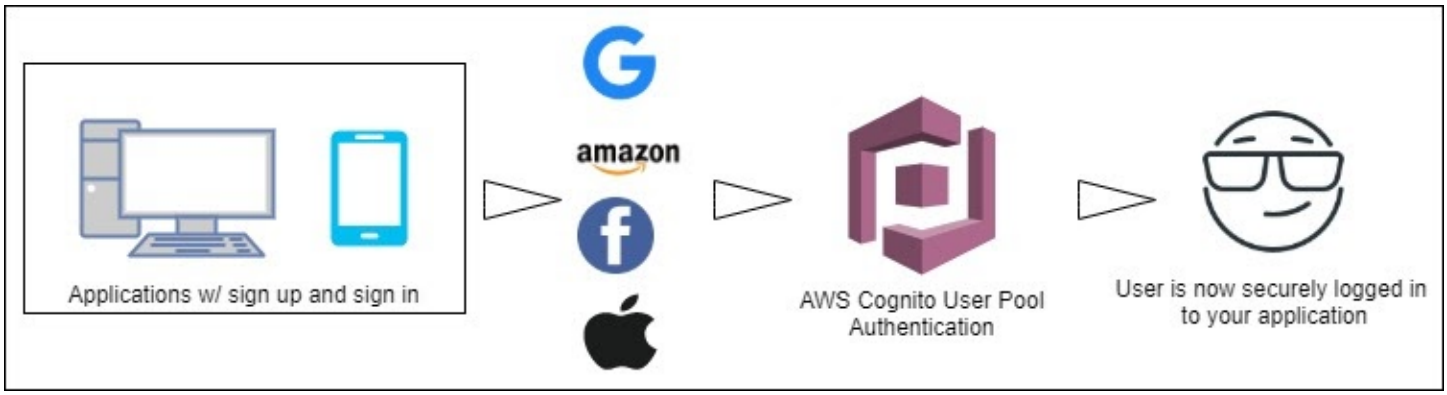
Final thoughts

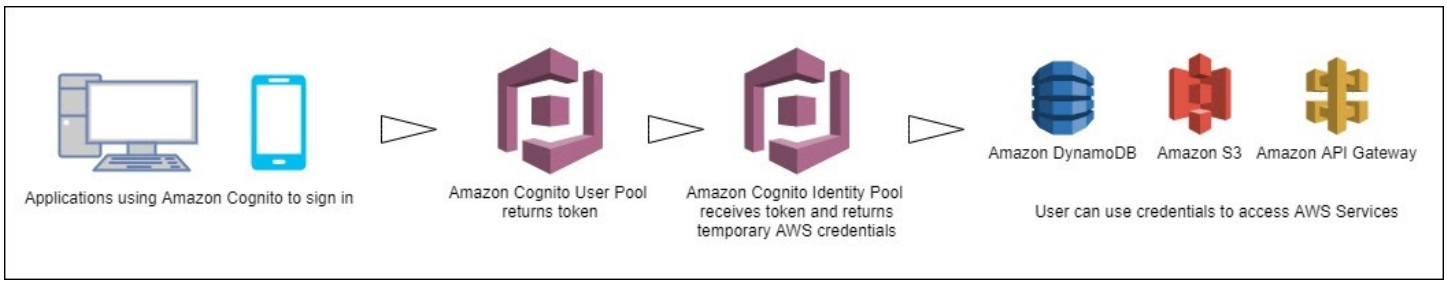
CMKs enable you to have control over your encryption needs in AWS. If you have applications that use AWS KMS, be sure to check the AWS SDK documentation for the appropriate syntax for your application's programming language. And thus, it is a good practice to always encrypt any valuable data you have at rest and in transit, and AWS KMS helps you in doing just that.

Sources:
https://docs.aws.amazon.com/kms/latest/developerguide/programming-keys.html
https://docs.aws.amazon.com/kms/latest/APIReference/API_Operations.html
https://docs.aws.amazon.com/cli/latest/reference/kms/

Amazon Cognito

- A user management and authentication service that can be integrated to your **web or mobile applications**. Amazon Cognito also enables you to authenticate users through an **external identity provider** and provides **temporary security credential**s to access your app's backend resources in AWS or any service behind Amazon API Gateway. Amazon Cognito works with external identity providers that support SAML or OpenID Connect, social identity providers (Facebook, Twitter, Amazon, Google, Apple) and you can also integrate your own identity provider.
- An Amazon Cognito ID token is represented as a **JSON Web Token** (JWT). Amazon Cognito uses JSON Web Tokens for token authentication.
- How It Works





User Pools

- User pools are user directories that provide sign-up and sign-in options for your app users.
- Users can sign in to your web or mobile app through Amazon Cognito, or federate through a third-party identity provider (IdP).
- You can use the aliasing feature to enable your users to sign up or sign in with an email address and a password or a phone number and a password.
- User pools are each created in one AWS Region, and they store the user profile data only in that region. You can also send user data to a different AWS Region.
- A User Pool is like a *directory* of users.
- Manage Users

        - After you create a user pool, you can create, confirm, and manage users accounts.
        - Amazon Cognito User Pools groups lets you manage your users and their access to resources by mapping IAM roles to groups.
        - User accounts are added to your user pool in one of the following ways:
        - The user signs up in your user pool's client app, which can be a mobile or web app.
        - You can import the user's account into your user pool.
        - You can create the user's account in your user pool and invite the user to sign in.
    - Getting started with User Pools
        1. Create a user pool
        2. Add an app to enable the Hosted Web UI
        3. (Optional) Add social sign-in feature to your user pool
        4. (Optional) Add SAML IdP feature to your user pool

CDA Exam Notes:

At this point, we recommend that you try creating your own User Pool in your AWS account. Elaborating the steps in the cheatsheets alone won't help you understand how to use Amazon Cognito User Pools. By simulating it yourself, you'll be able to remember the different settings under Cognito User Pools and you'll understand the flow of the process better.

- **Identity Pools**
    - Use this feature if you want to federate users to your AWS services.
    - Identity pools enable you to grant your users temporary AWS credentials to access AWS services, such as Amazon S3 and DynamoDB.
    - Identity pools support anonymous guest users, as well as the following identity providers:
        - Amazon Cognito user pools
        - Social sign-in with Facebook, Google, and Login with Amazon
        - OpenID Connect (OIDC) providers
        - SAML identity providers
        - Developer authenticated identities
    - To save user profile information, your identity pool needs to be integrated with a user pool.
    - Amazon Cognito Identity Pools can support unauthenticated identities by providing a unique identifier and AWS credentials for users who do not authenticate with an identity provider.
    - The permissions for each authenticated and non-authenticated user are controlled through IAM roles that you create.
    - Once you have an OpenID Connect token, you can then trade this for temporary AWS credentials via the *AssumeRoleWithWebIdentity* API call in AWS Security Token Service (STS). This call is no different than if you were using Facebook, Google+, or Login with Amazon directly, except that you are passing an Amazon Cognito token instead of a token from one of the other public providers.

- Common Use Cases
    - Enable your users to authenticate with a user pool.
    - After a successful user pool sign-in, your web or mobile app will receive user pool tokens from Amazon Cognito. You can use those tokens to control access to your server-side resources.
    - Access resources with API Gateway and Lambda with a User Pool. API Gateway validates the tokens from a successful user pool authentication, and uses them to grant your users access to resources including Lambda functions, or your own API.
    - After a successful user pool authentication, your app will receive user pool tokens from Amazon Cognito. You can exchange them for temporary access to other AWS services with an identity pool.
    - Enable your users access to AWS services through an identity pool. In exchange, the identity pool grants temporary AWS credentials that you can use to access other AWS services.
    - Grant your users access to AWS AppSync resources with tokens from a successful Amazon Cognito authentication (from a user pool or an identity pool).
    - Amazon Cognito is also commonly used together with AWS Amplify, a framework for developing web and mobile applications with AWS services.

CDA Exam Notes:

So what are some hints that will tell you whether you'll need to use identity pool or user pool?

User pools are convenient when you have an application that needs sign up and sign in functionality. This can be done through manual registration of an account or through some other identity provider. Also, user pools allow you to access and manage user data, in case you have attributes regarding a user other than a username and password.

Identity pools, ironically, are not designed to create individual identities for your users a.k.a. their own account. Instead, identity pools are primarily for authentication (federation). If users need temporary access to your AWS resources, and not by logging in using a personal account, then you should go with this pool. The key terms to always remember here are **temporary** and **AWS credentials**.

And yes, you can combine identity pools with user pools. More details are provided in the next section.

Amazon Cognito Sync

- Store and sync data across devices using Cognito Sync.
- You can programmatically trigger the sync of data sets between client devices and the Amazon Cognito sync store by using the synchronize() method in the AWS Mobile SDK. The synchronize() method reads the **latest version** of the data available in the Amazon Cognito sync store and compares it to the local, cached copy. After comparison, the synchronize() method

writes the latest updates as necessary to the local data store and the Amazon Cognito sync store.
    - The Amazon Cognito Sync store is a key/value pair store linked to an Amazon Cognito identity. There is no limit to the number of identities you can create in your identity pools and sync store.
    - Each user information store can have a maximum size of 20MB. Each data set within the user information store can contain up to 1MB of data. Within a data set you can have up to 1024 keys.
    - With Cognito Streams, you can push sync store data to a Kinesis stream in your AWS account.
- Advanced Security Features
    - When Amazon Cognito detects unusual sign-in activity, such as sign-in attempts from new locations and devices, it assigns a risk score to the activity and lets you choose to either prompt users for additional verification or block the sign-in request.
    - Users can verify their identities using SMS or a Time-based One-time Password (TOTP) generator.
    - When Amazon Cognito detects users have entered credentials that have been compromised elsewhere, it prompts a password change.
- Integration with AWS Lambda
    - You can create an AWS Lambda function and then trigger that function during user pool operations such as user sign-up, confirmation, and sign-in (authentication) with a Lambda trigger.
    - Amazon Cognito invokes Lambda functions synchronously. When called, your Lambda function must respond within 5 seconds. If it does not, Amazon Cognito retries the call. After 3 unsuccessful attempts, the function times out.
    - You can create a Lambda function as a backend to Cognito that serves auth challenges to users signing in.
- Pricing
    - If you are using Cognito Identity to create a User Pool, you pay based on your monthly active users (MAUs) only. A user is counted as a MAU if, within a calendar month, there is an identity operation related to that user, such as sign-up, sign-in, token refresh or password change.
        - The Cognito Your User Pool feature has a free tier of 50,000 MAUs for users who sign in directly to Cognito User Pools or through social identity providers, and 50 MAUs for users federated through SAML 2.0 based identity providers.
    - You pay an additional fee when you enable advanced security features for Amazon Cognito.
    - Amazon Cognito uses Amazon SNS for sending SMS messages for Multi-Factor Authentication (MFA) and phone number verification, so there are associated SNS costs as well

Sources:
https://aws.amazon.com/cognito/
https://aws.amazon.com/cognito/faqs/

https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html
Overview of Amazon Cognito User Pools and Federated Identities

Amazon Cognito User Pools vs Identity Pools

With the proliferation of smartphones in our connected world, more and more developers are quickly deploying their applications on the cloud. One of the first challenges in developing applications is allowing users to log in and authenticate on your applications. There are multiple stages involved in user verification and most of these are not visible from the end-user. AWS provides an easy solution for this situation.

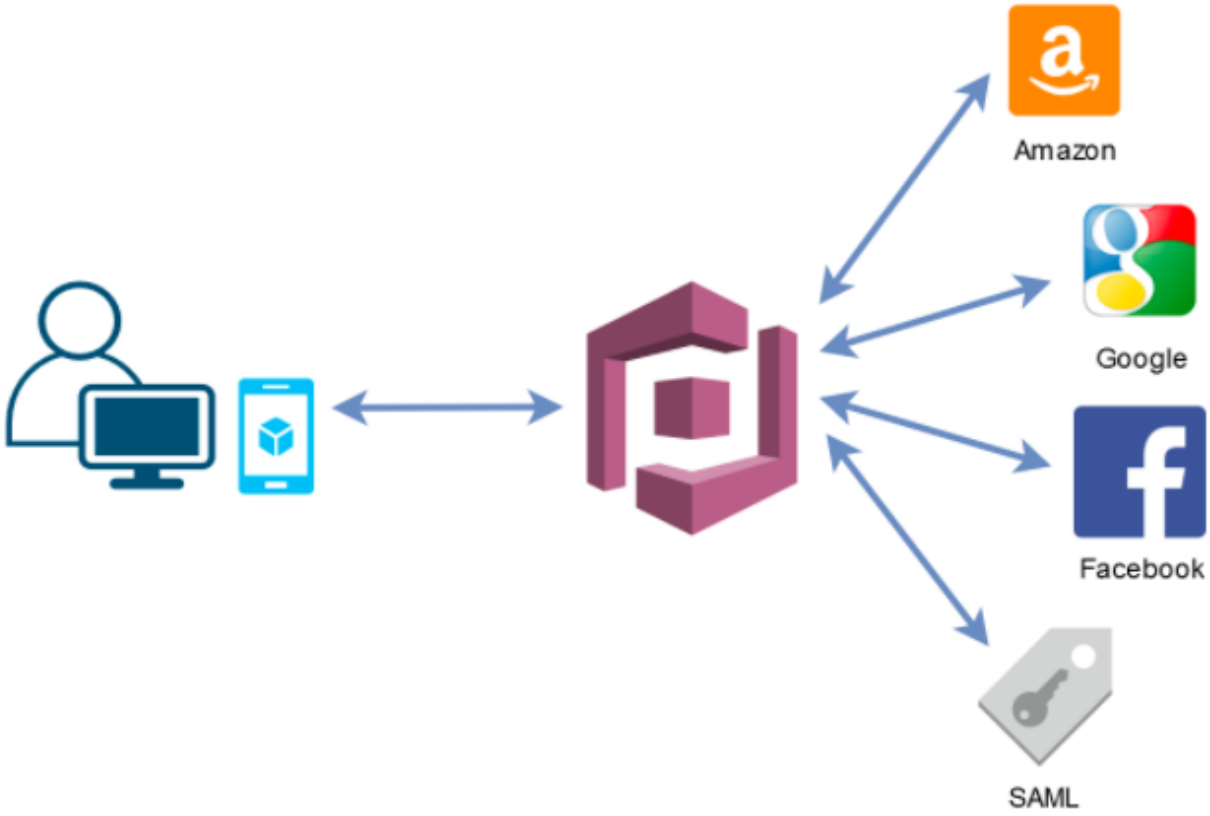
User Identity verification is at the core of Amazon Cognito. It provides solutions for three key areas of user identification:

1. **Authentication** – provides users sign-up and sign-in options. Enables support for federation with Enterprise Identities (Microsoft AD), or Social Identities (Amazon, Facebook, Google, etc.)
2. **Authorization** – sets of permission or operations allowed for a user. It provides fine-grained access control to resources.
3. **User Management** – allows management of user lifecycles, such as importing users, onboarding users, disabling users, and storing and managing user profiles.

In this post, we'll talk about Cognito User Pools and Identity Pools, including an overview of how they are used to provide authentication and authorization functionalities that can be integrated on your mobile app.

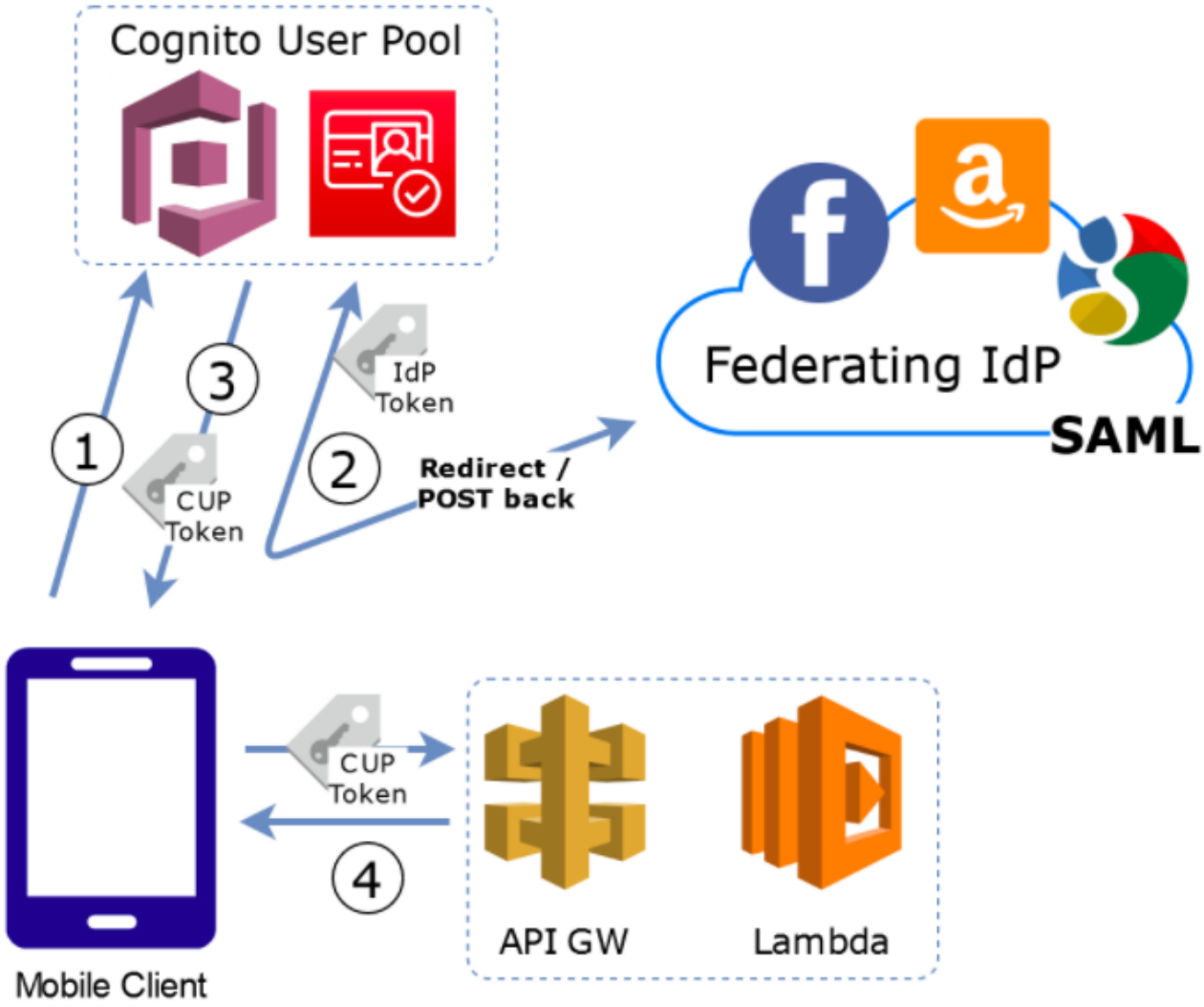
Amazon Cognito User Pools

Amazon Cognito User Pools are used for authentication. To verify your user's identity, you will want to have a way for them to login using username/passwords or federated login using Identity Providers such as Amazon, Facebook, Google, or a SAML supported authentication such as Microsoft Active Directory. You can configure these Identity Providers on Cognito, and it will handle the interactions with these providers so you only have to worry about handling the Authentication tokens on your app.



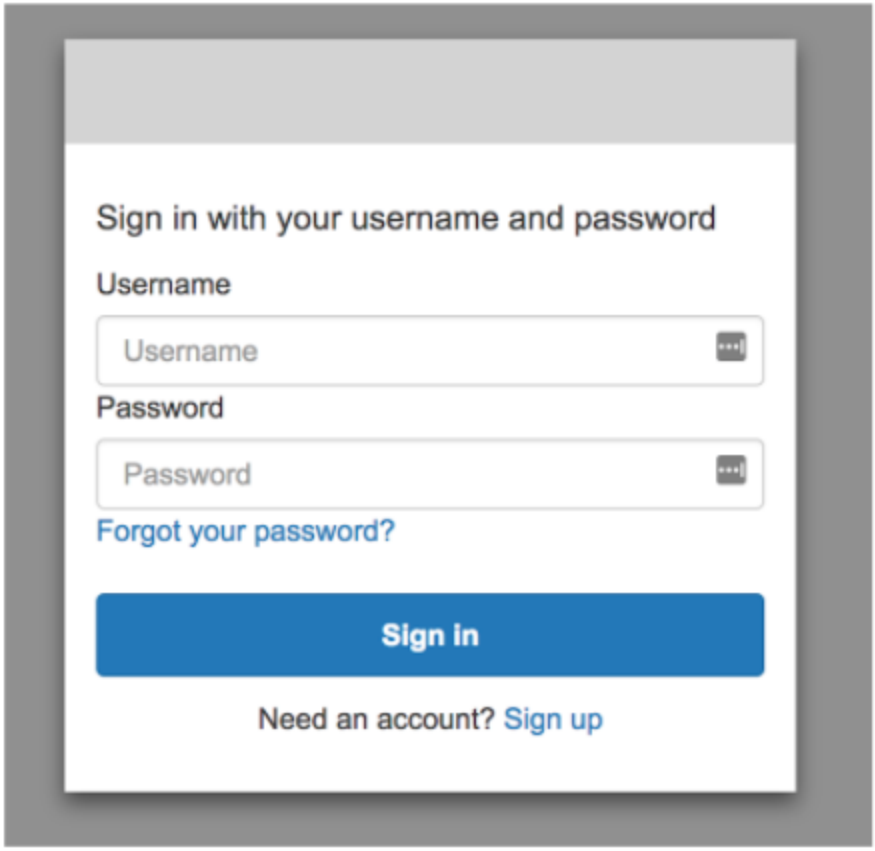
With Cognito User Pools, you can provide sign-up and sign-in functionality for your mobile or web app users. You don't have to build or maintain any server infrastructure on which users will authenticate.

This diagram shows how authentication is handled with Cognito User Pools:



1. Users send authentication requests to Cognito User Pools.
2. The Cognito user pool verifies the identity of the user or sends the request to Identity Providers such as Facebook, Google, Amazon, or SAML authentication (with Microsoft AD).
3. The Cognito User Pool Token is sent back to the user.
4. The person can then use this token to access your backend APIs hosted on your EC2 clusters or in API Gateway and Lambda.

If you want a quick login page, you can even use the pre-built login UI provided by Amazon Cognito which you just have to integrate on your application.



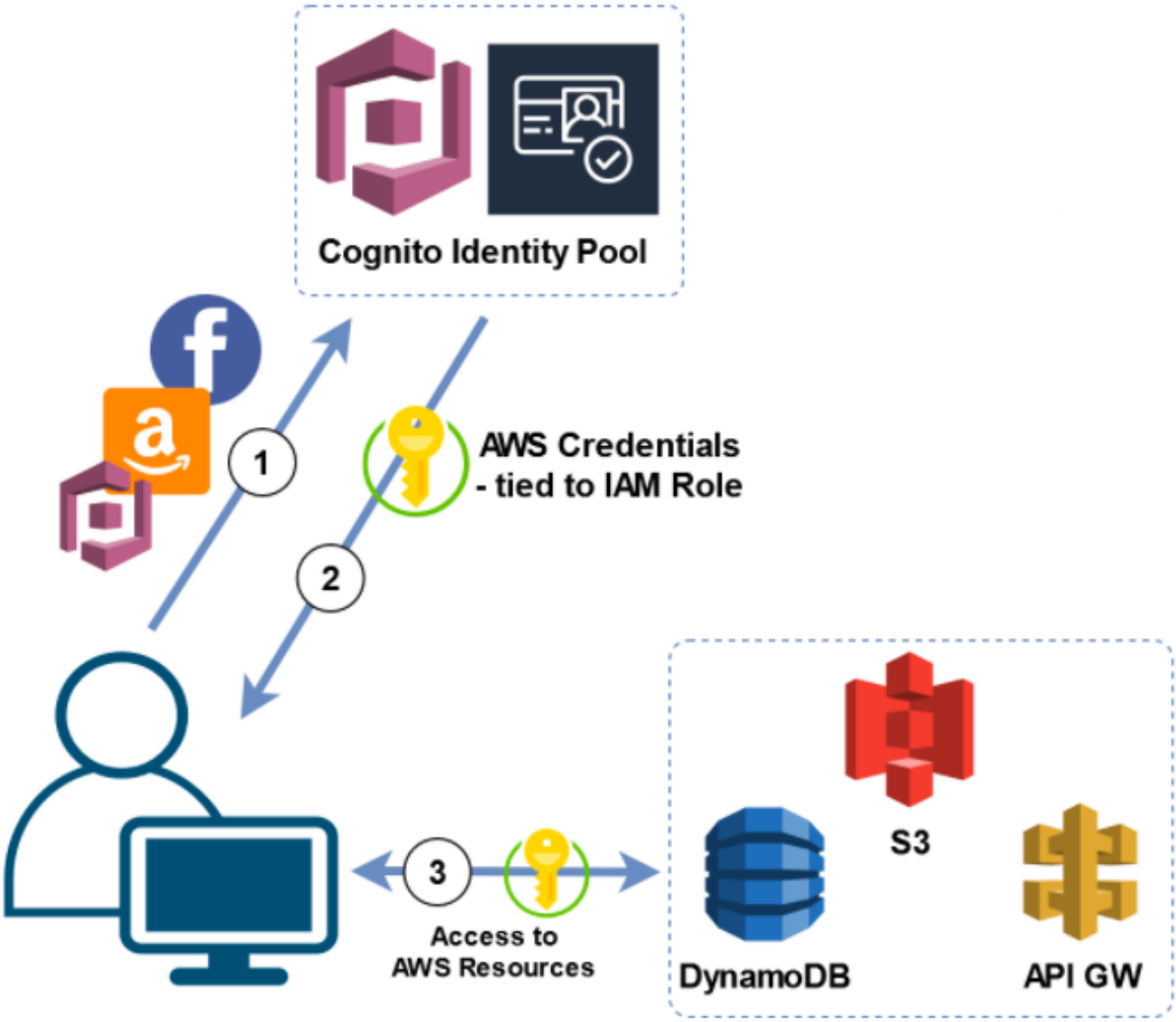
On the Amazon Cognito User Pool page, you can also manage users if you need to. You can reset the password, disable/enable users, and enroll/delete users or other actions needed for User Management.

Amazon Cognito Identity Pools

Cognito Identity Pools (Federated Identities) provides different functionality compared to User Pools. Identity Pools are used for User Authorization. You can create unique identities for your users and federate them with your identity providers. Using identity pools, users can obtain temporary AWS credentials to access other AWS services.

Identity Pools can be thought of as the actual mechanism authorizing access to AWS resources. When you create Identity Pools, think of it as defining who is allowed to get AWS credentials and use those credentials to access AWS resources.

This diagram shows how authorization is handled with Cognito Identity Pools:



1. The web app or mobile app sends its authentication token to Cognito Identity Pools. The token can come from a valid Identity Provider, like Cognito User Pools, Amazon, or Facebook.
2. Cognito Identity Pool exchanges the user authentication token for temporary AWS credentials to access resources such as S3 or DynamoDB. AWS credentials are sent back to the user.
3. The temporary AWS credentials will be used to access AWS resources.

You can define rules in Cognito Identity Pools for mapping users to different IAM roles to provide fine-grain permissions.

1. Here's a table summary describing Cognito User Pool and Identity Pool:

Cognito User Pools	Cognito Identity Pools
Handles the IdP interactions for you	Provides AWS credentials for accessing resources on behalf of users
Provides profiles to manage users	Supports rules to map users to different IAM roles
Provides OpenID Connect and OAuth standard tokens	Free
Priced per monthly active user	

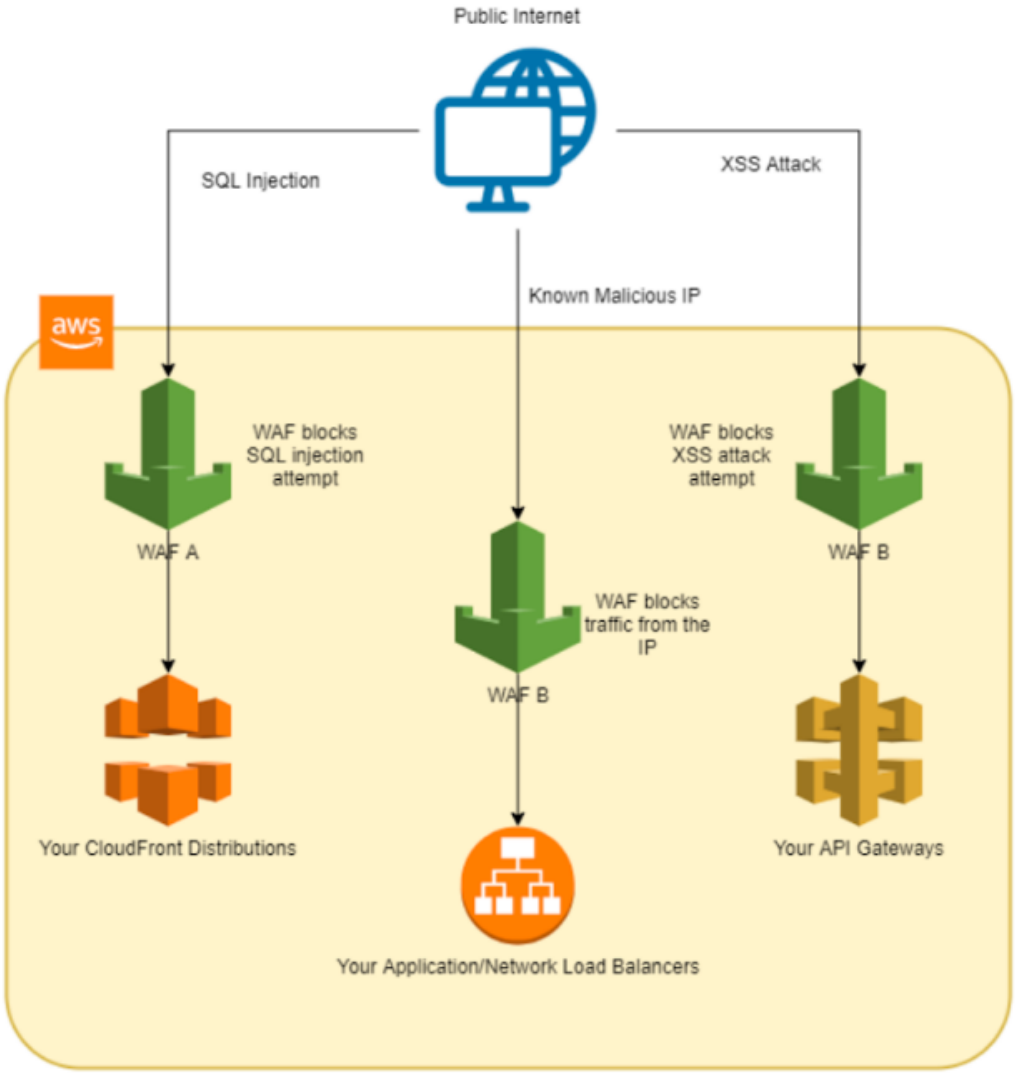
Sources:
https://aws.amazon.com/premiumsupport/knowledge-center/cognito-user-pools-identity-pools/
https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-identity-pools.html
https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-identity.html
https://docs.aws.amazon.com/cognito/latest/developerguide/authentication.html
https://docs.aws.amazon.com/cognito/latest/developerguide/switching-identities.html

AWS WAF

- A web application firewall that helps protect web applications from attacks by allowing you to configure rules that **allow, block, or monitor (count) web requests** based on conditions that you define.
- These conditions include:
    - IP addresses
    - HTTP headers
    - HTTP body
    - URI strings
    - SQL injection
    - cross-site scripting.

Features

- WAF lets you create rules to filter web traffic based on conditions that include IP addresses, HTTP headers and body, or custom URIs.
- You can also create rules that block common web exploits like SQL injection and cross site scripting.
- For application layer attacks, you can use WAF to respond to incidents. You can set up proactive rules like *Rate Based Blacklisting* to automatically block bad traffic, or respond immediately to incidents as they happen.
- WAF provides real-time metrics and captures raw requests that include details about IP addresses, geo locations, URIs, User-Agent and Referers.
- **AWS WAF Security Automations** is a solution that automatically deploys a single web access control list (web ACL) with a set of AWS WAF rules designed to filter common web-based attacks. The solution supports log analysis using Amazon Athena and AWS WAF full logs.



Conditions, Rules, and Web ACLs

- You define your conditions, combine your conditions into rules, and combine the rules into a web ACL.
- **Conditions** define the basic characteristics that you want WAF to watch for in web requests.
- You combine conditions into **rules** to precisely target the requests that you want to allow, block, or count. WAF provides two types of rules:
    - **Regular rules** - use only conditions to target specific requests.
    - **Rate-based rules** - are similar to regular rules, with a rate limit. Rate-based rules count the requests that arrive from a specified IP address every five minutes. The rule can trigger an action if the number of requests exceed the rate limit.
- **WAF Managed Rules** are an easy way to deploy pre-configured rules to protect your applications common threats like application vulnerabilities. All Managed Rules are automatically updated by AWS Marketplace security Sellers.

- After you combine your conditions into rules, you combine the rules into a **web ACL**. This is where you define an action for each rule—allow, block, or count—and a default action, which determines whether to allow or block a request that doesn't match all the conditions in any of the rules in the web ACL.

Pricing

- WAF charges based on the number of web access control lists (web ACLs) that you create, the number of rules that you add per web ACL, and the number of web requests that you receive.

Sources:
https://docs.aws.amazon.com/waf/latest/developerguide
https://aws.amazon.com/waf/features/
https://aws.amazon.com/waf/pricing/
https://aws.amazon.com/waf/faqs/

AWS Secrets Manager

- A secret management service that enables you to easily rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle.
- Features
    - AWS Secrets Manager encrypts secrets at rest using encryption keys that you own and store in AWS Key Management Service [customer managed keys]. When you retrieve a secret, Secrets Manager decrypts the secret and transmits it securely over TLS to your local environment.
    - You can rotate secrets on a schedule or on demand by using the Secrets Manager console, AWS SDK, or AWS CLI.
    - Secrets Manager natively supports rotating credentials for databases hosted on Amazon RDS and Amazon DocumentDB and clusters hosted on Amazon Redshift.
    - You can extend Secrets Manager to rotate other secrets, such as credentials for Oracle databases hosted on EC2 or OAuth refresh tokens, by using custom AWS Lambda functions.
- A secret consists of a set of credentials (user name and password), and the connection details used to access a secured service.
- A secret also contains **metadata** which include:
    - Basic information includes the name of the secret, a description, and the Amazon Resource Name (ARN) to serve as a unique identifier.
    - The ARN of the AWS KMS key Secrets Manager uses to encrypt and decrypt the protected text in the secret. If you don't provide this information, Secrets Manager uses the default AWS KMS key for the account.
    - Information about how frequently to rotate the key and what Lambda function to use to perform the rotation.
    - A user-provided set of tags. You can attach tags as key-value pairs to AWS resources for organizing, logical grouping, and cost allocation.
- A secret can contain **versions**:
    - Although you typically only have one version of the secret active at a time, multiple versions can exist while you rotate a secret on the database or service. Whenever you change the secret, Secrets Manager creates a new version.
    - Each version holds a copy of the encrypted secret value.
    - Each version can have one or more *staging labels* attached identifying the stage of the secret rotation cycle.
- Supported Secrets
    - Database credentials, on-premises resource credentials, SaaS application credentials, third-party API keys, and SSH keys.
    - You can also store JSON documents.
- To retrieve secrets, you simply replace secrets in plain text in your applications with code to pull in those secrets programmatically using the Secrets Manager APIs.
- Secrets can be cached on the client side, and updated only during a secret rotation.

- During the secret rotation process, Secrets Manager tracks the older credentials, as well as the new credentials you want to start using, until the rotation completes. It tracks these different versions by using *staging labels*.
- How Secret Rotation Works
    - The rotation function contacts the secured service authentication system and creates a new set of credentials to access the database. Secrets Manager stores these new credentials as the secret text in a new version of the secret with the AWSPENDING staging label attached.
    - The rotation function then tests the AWSPENDING version of the secret to ensure that the credentials work, and grants the required level of access to the secured service.
    - If the tests succeed, the rotation function then moves the label AWSCURRENT to the new version to mark it as the default version. Then, all of the clients start using this version of the secret instead of the old version. The function also assigns the label AWSPREVIOUS to the old version. The version that had AWSPREVIOUS staging label now has no label, and therefore deprecated.
- Network Setup for Secret Rotation
    - When rotating secrets on natively supported services, Secrets Manager uses CloudFormation to build the rotation function and configure the network connection between the two.
        - If your protected database service **runs in a VPC and is not publicly accessible**, then the CloudFormation template configures the **Lambda rotation function to run in the same VPC**. The rotation function can communicate with the protected service **directly within the VPC**.
        - If you run your protected service as a **publicly accessible resource**, in a VPC or not, then the CloudFormation template configures the **Lambda rotation function not to run in a VPC**. The Lambda rotation function communicates with the protected service **through the publicly accessible connection point**.
    - By default, the Secrets Manager endpoints run on the public Internet. If you run your Lambda rotation function and protected database or service in a VPC, then you must perform one of the following steps:
        - **Add a NAT gateway to your VPC**. This enables traffic that originates in your VPC to reach the public Secrets Manager endpoint.
        - **Configure Secrets Manager service endpoints directly within your VPC**. This configures your VPC to intercept any request addressed to the public regional endpoint, and redirect the request to the private service endpoint running within your VPC.
- You can create two secrets that have different permissions
    - User Secret - can be used to connect to linked services, but it cannot be rotated. The user will have to wait for the master secret to be rotated and propagated for it to change.
    - Master Secret - has sufficient permissions to rotate secrets of linked services. This scenario is typically used when you have users that are actively using the old secret, and you do not want to break operations after you rotate the secret. You can have your users update their clients first before using the newly rotated credentials.

CDA Exam Notes:

Secrets Manager vs Systems Manager Parameter Store vs Storing variables locally?

Secrets Manager is a great service to use if you want to decouple your sensitive data from your applications. You can also manage your secrets without modifying your applications directly, such as performing encryption and secret rotation. Your secrets are kept in a centralized location which can be securely referenced by different applications at the same time, wherever they are. The catch is, this is a paid service.

Parameter Store is similar to Secrets Manager in use case except you cannot rotate your secrets as easily as with Secrets Manager. You also cannot generate random secrets immediately. The benefit in using this service, however, is that it's free.

If by chance, these services do not fit your use case and you want to manage your variables yourself then choose to store them locally in your instances or applications instead. Possible reasons might be because of compliance requirements, or your variables undergo some processing and you need fine-grained control over that.

- Security
  - By default, Secrets Manager does not write or cache the secret to persistent storage.
  - By default, Secrets Manager only accepts requests from hosts that use the open standard Transport Layer Security (TLS) and Perfect Forward Secrecy.
  - You can control access to the secret using AWS Identity and Access Management (IAM) policies.
  - You can tag secrets individually and apply tag-based access controls.
  - You can configure VPC endpoints to keep traffic between your VPC and Secrets Manager within the AWS network.
  - Secrets Manager does not immediately delete secrets. Instead, Secrets Manager immediately makes the secrets inaccessible and scheduled for deletion after a recovery window of a **minimum of seven days**. Until the recovery window ends, you can recover a secret you previously deleted.
  - By using the CLI, you can delete a secret without a recovery window.
- Compliance
  - Secrets Manager is HIPAA, PCI DSS and ISO, SOC, FedRAMP, DoD SRG, IRAP, and OSPAR compliant.
- Pricing
  - You pay based on the number of secrets stored and API calls made per month.

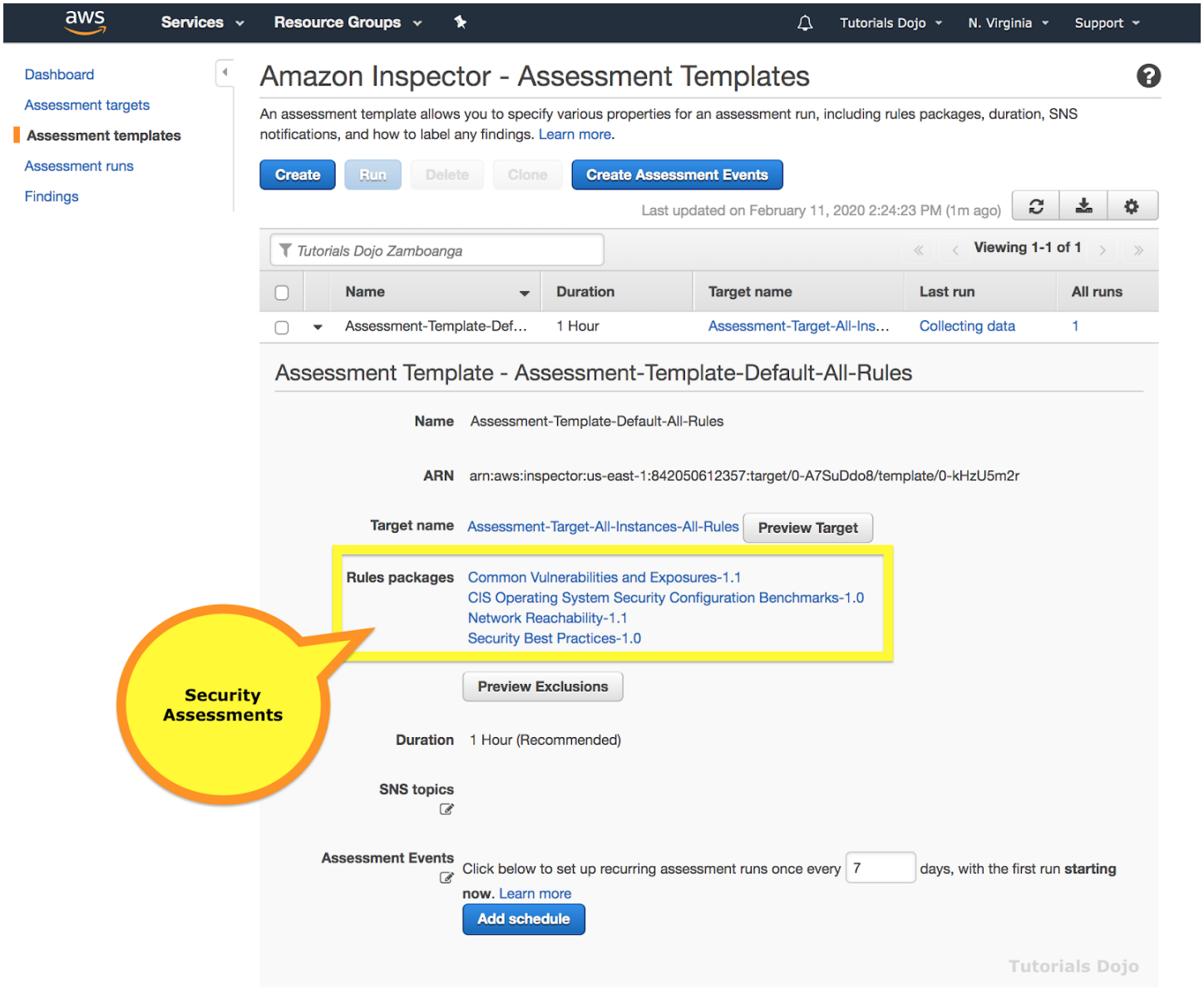
Sources:
https://aws.amazon.com/secrets-manager/
https://aws.amazon.com/secrets-manager/faqs/
https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html
Understanding AWS Secrets Manager

Amazon Inspector

- An automated security assessment service that helps you test the network accessibility of your EC2 instances and the security state of your applications running on the instances.
- Inspector uses IAM *service-linked roles*.

Features

- Inspector provides an engine that analyzes system and resource configuration and monitors activity to determine what an assessment target looks like, how it behaves, and its dependent components. The combination of this telemetry provides a complete picture of the assessment target and its potential security or compliance issues.
- Inspector incorporates a built-in library of rules and reports. These include checks against best practices, common compliance standards and vulnerabilities.
- Automate security vulnerability assessments throughout your development and deployment pipeline or against static production systems.
- Inspector is an API-driven service that uses an optional *agent*, making it easy to deploy, manage, and automate.



Concepts

- **Inspector Agent** - A software agent that you can install on all EC2 instances that are included in the assessment target, the security of which you want to evaluate with Inspector.
- **Assessment run** - The process of discovering potential security issues through the analysis of your assessment target's configuration and behavior against specified rules packages.
- **Assessment target** - A collection of AWS resources that work together as a unit to help you accomplish your business goals. Inspector assessment targets can consist only of EC2 instances.
- **Assessment template** - A configuration that is used during your assessment run, which includes
    - Rules packages against which you want Inspector to evaluate your assessment target,

    - The duration of the assessment run,
    - Amazon SNS topics to which you want Inspector to send notifications about assessment run states and findings,
    - Inspector-specific attributes (key-value pairs) that you can assign to findings generated by the assessment run that uses this assessment template.
    - After you create an assessment template, you can't modify it.
- **Finding** - A potential security issue discovered during the assessment run of the specified target.
- **Rule** - A security check performed during an assessment run. When a rule detects a potential security issue, Inspector generates a finding that describes the issue.
- **Rules package** - A collection of rules that corresponds to a security goal that you might have.
- **Telemetry** - EC2 instance data collected by Inspector during an assessment run and passed to the Inspector service for analysis.
- The telemetry data generated by the Inspector Agent during assessment runs is formatted in JSON files and delivered in near-real-time over TLS to Inspector, where it is encrypted with a per-assessment-run, ephemeral KMS-derived key and securely stored in an S3 bucket dedicated for the service.

Rules Packages and Rules

- Inspector compares the behavior and the security configuration of the assessment targets to selected security *rules packages*.
- *Rules* are grouped together into distinct rules packages either by category, severity, or pricing.
- Each rule has an assigned severity level
    - **High**, **Medium**, and **Low** levels all indicate a security issue that can result in compromised information confidentiality, integrity, and availability within your assessment target.
    - The **Informational** level simply highlights a security configuration detail of your assessment target.
- The findings generated by **rules in the Network Reachability package** show whether your ports are reachable from the internet through an internet gateway, a VPC peering connection, or a VPN through a virtual gateway. These findings also highlight network configurations that allow for potentially malicious access, such as mismanaged security groups, ACLs, IGWs, and so on.

Assessment Reports

- A document that details what is tested in the assessment run, and the results of the assessment.
- You can view the following types of assessment reports:
    - **Findings report** - this report contains the following information:
        - Executive summary of the assessment
        - EC2 instances evaluated during the assessment run
        - Rules packages included in the assessment run
        - Detailed information about each finding, including all EC2 instances that had the finding

- **Full report** - this report contains all the information that is included in a findings report, and additionally provides the list of rules that passed on all instances in the assessment target.

Pricing

- Pricing is based on two dimensions
  - The number of EC2 instances included in each assessment
  - The type(s) of rules package you select: host assessment rules packages and/or the network reachability rules package

Sources:
https://docs.aws.amazon.com/inspector/latest/userguide
https://aws.amazon.com/inspector/pricing/
https://aws.amazon.com/inspector/faqs/

MANAGEMENT

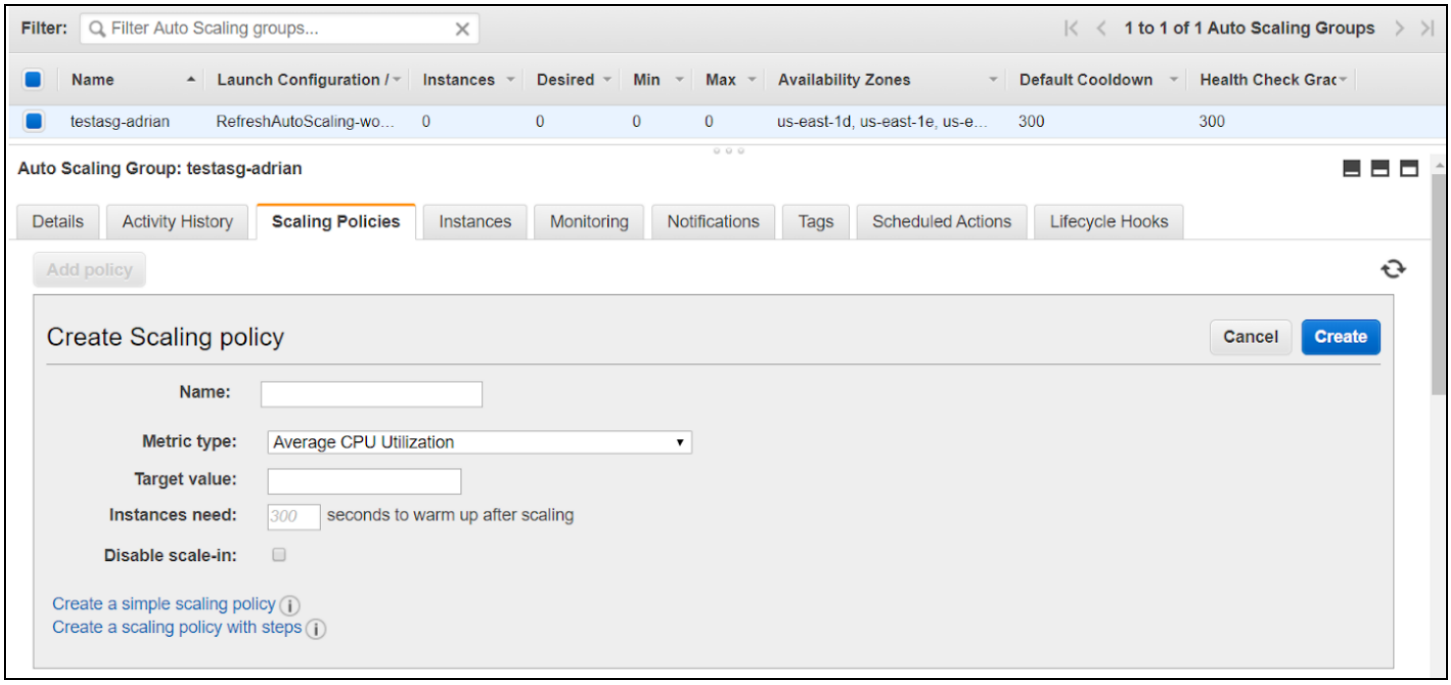
AWS Auto Scaling

- Configure automatic scaling for the AWS resources quickly through a scaling plan that uses **dynamic scaling** and **predictive scaling**.
- Optimize for availability, for cost, or a balance of both.
- Scaling in means decreasing the size of a group while scaling out means increasing the size of a group.
- Useful for
    - Cyclical traffic such as high use of resources during regular business hours and low use of resources overnight
    - On and off traffic patterns, such as batch processing, testing, or periodic analysis
    - Variable traffic patterns, such as software for marketing campaigns with periods of spiky growth
- It is a region specific service.
- Features
    - Launch or terminate EC2 instances in an Auto Scaling group.
    - Launch or terminate instances from an EC2 Spot Fleet request, or automatically replace instances that get interrupted for price or capacity reasons.
    - Adjust the ECS service desired count up or down in response to load variations.
    - Enable a DynamoDB table or a global secondary index to increase or decrease its provisioned read and write capacity to handle increases in traffic without throttling.
    - Dynamically adjust the number of Aurora read replicas provisioned for an Aurora DB cluster to handle changes in active connections or workload.
    - Use **Dynamic Scaling** to add and remove capacity for resources to maintain resource utilization at the specified target value.
    - Use **Predictive Scaling** to forecast your future load demands by analyzing your historical records for a metric. It also allows you to schedule scaling actions that proactively add and remove resource capacity to reflect the load forecast, and control maximum capacity behavior. Only available for EC2 Auto Scaling groups.
    - AWS Auto Scaling scans your environment and automatically discovers the scalable cloud resources underlying your application, so you don't have to manually identify these resources one by one through individual service interfaces.
    - You can suspend and resume any of your AWS Application Auto Scaling actions.
- Amazon EC2 Auto Scaling
    - Ensuring you have the correct number of EC2 instances available to handle your application load using **Auto Scaling Groups**.
    - An **Auto Scaling group** contains a collection of EC2 instances that share similar characteristics and are treated as a logical grouping for the purposes of instance scaling and management.
    - You specify the minimum, maximum and desired number of instances in each Auto Scaling group.

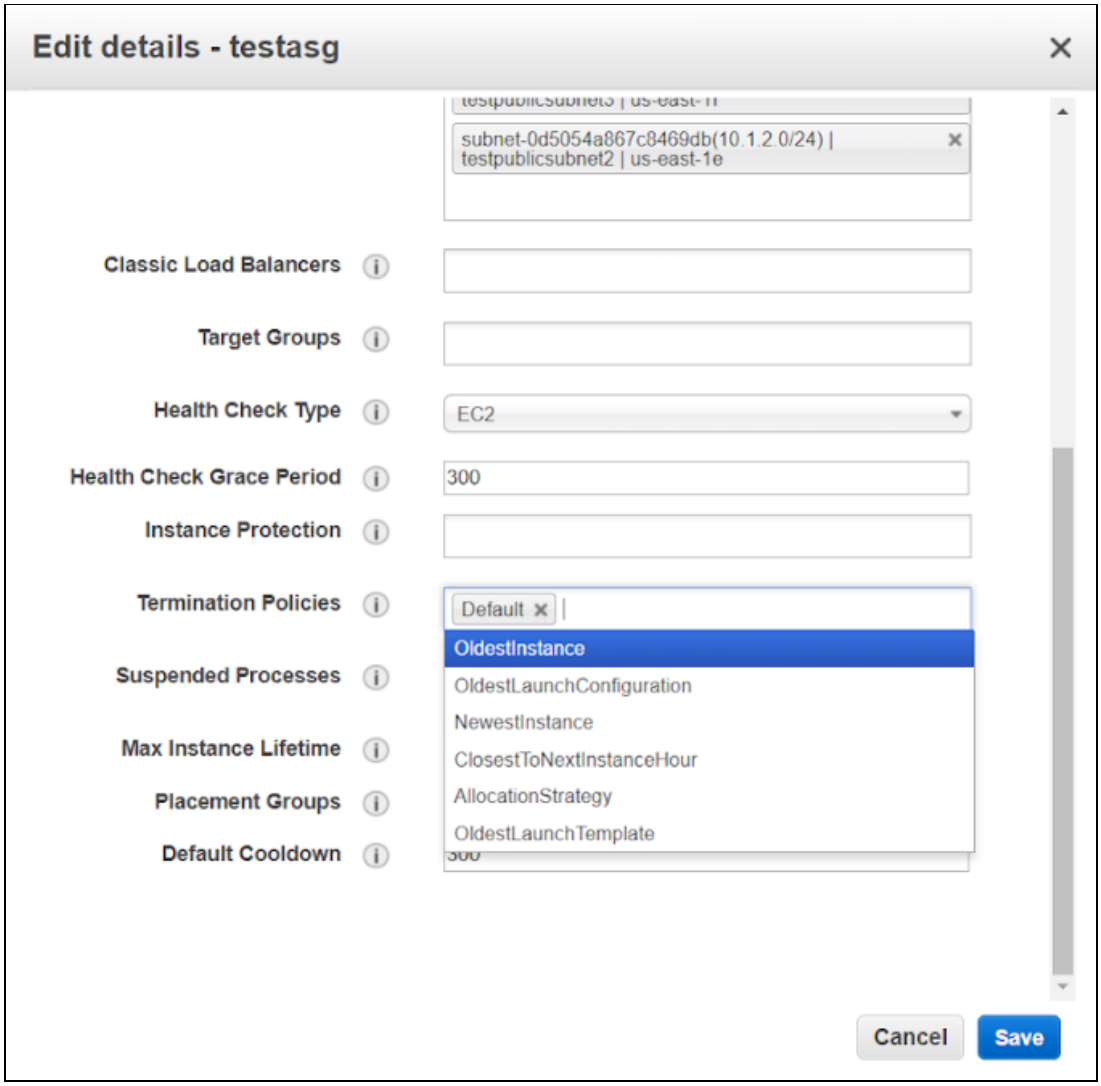
- Key Components

Groups	Your EC2 instances are organized into *groups* so that they are treated as a logical unit for scaling and management. When you create a group, you can specify its minimum, maximum, and desired number of EC2 instances.
Launch configurations	Your group uses a *launch configuration* as a template for its EC2 instances. When you create a launch configuration, you can specify information such as the AMI ID, instance type, key pair, security groups, and block device mapping for your instances.
Scaling options	How to scale your Auto Scaling groups.

- You can add a **lifecycle hook** to your Auto Scaling group to perform custom actions when instances launch or terminate.
- Scaling Options
  - Scale to maintain current instance levels at all times
  - Manual Scaling
  - Scale based on a schedule
  - Scale based on a demand
- Scaling Policy Types
  - **Target tracking scaling**—Increase or decrease the current capacity of the group based on a target value for a specific metric.
  - **Step scaling**—Increase or decrease the current capacity of the group based on a set of scaling adjustments, known as step adjustments, that vary based on the size of the alarm breach.
  - **Simple scaling**—Increase or decrease the current capacity of the group based on a single scaling adjustment.
- The **cooldown period** is a configurable setting that helps ensure to not launch or terminate additional instances before previous scaling activities take effect.
  - EC2 Auto Scaling supports cooldown periods when using simple scaling policies, but not when using target tracking policies, step scaling policies, or scheduled scaling.



- Amazon EC2 Auto Scaling marks an instance as unhealthy if the instance is in a state other than *running*, the system status is *impaired*, or Elastic Load Balancing reports that the instance failed the health checks.
- Termination of Instances
  - When you configure automatic scale in, you must decide which instances should terminate first and set up a **termination policy**. You can also use **instance protection** to prevent specific instances from being terminated during automatic scale in.
  - Default Termination Policy
  - Custom Termination Policies
    - *OldestInstance* - Terminate the oldest instance in the group.
    - *NewestInstance* - Terminate the newest instance in the group.
    - *OldestLaunchConfiguration* - Terminate instances that have the oldest launch configuration.
    - *ClosestToNextInstanceHour* - Terminate instances that are closest to the next billing hour.



You can create **launch templates** that specifies instance configuration information when you launch EC2 instances, and allows you to have multiple versions of a template.
A **launch configuration** is an instance configuration template that an Auto Scaling group uses to launch EC2 instances, and you specify information for the instances.

- You can specify your launch configuration with multiple Auto Scaling groups.
- You can only specify one launch configuration for an Auto Scaling group at a time, and you can't modify a launch configuration after you've created it.
- When you create a VPC, by default its tenancy attribute is set to *default*. You can launch instances with a tenancy value of *dedicated* so that they run as single-tenancy instances. Otherwise, they run as shared-tenancy instances by default.
- If you set the tenancy attribute of a VPC to *dedicated*, all instances launched in the VPC run as single-tenancy instances.

- When you create a launch configuration, the default value for the instance placement tenancy is *null* and the instance tenancy is controlled by the tenancy attribute of the VPC.

Launch Configuration Tenancy	VPC Tenancy = default	VPC Tenancy = dedicated
not specified	shared-tenancy instance	Dedicated Instance
default	shared-tenancy instance	Dedicated Instance
dedicated	Dedicated Instance	Dedicated Instance

- If you are launching the instances in your Auto Scaling group in EC2-Classic, you can link them to a VPC using *ClassicLink*.

- Application Auto Scaling
  - Allows you to configure automatic scaling for the following resources:
    - Amazon ECS services
    - Spot Fleet requests
    - Amazon EMR clusters
    - AppStream 2.0 fleets
    - DynamoDB tables and global secondary indexes
    - Aurora replicas
    - Amazon SageMaker endpoint variants
    - Custom resources provided by your own applications or services.
  - Features
    - **Target tracking scaling**—Scale a resource based on a target value for a specific CloudWatch metric.
    - **Step scaling**— Scale a resource based on a set of scaling adjustments that vary based on the size of the alarm breach.
    - **Scheduled scaling**—Scale a resource based on the date and time.
  - Target tracking scaling
    - You can have multiple target tracking scaling policies for a scalable target, provided that each of them uses a different metric.
    - You can also optionally disable the scale-in portion of a target tracking scaling policy.
  - Step scaling
    - Increase or decrease the current capacity of a scalable target based on a set of scaling adjustments, known as **step adjustments**, that vary based on the size of the alarm breach.
  - Scheduled scaling

        - Scale your application in response to predictable load changes by creating *scheduled actions*, which tell Application Auto Scaling to perform scaling activities at specific times.
    - The *scale out cooldown period* is the amount of time, in seconds, after a scale out activity completes before another scale out activity can start.
    - The *scale in cooldown period* is the amount of time, in seconds, after a scale in activity completes before another scale in activity can start.
- You can attach one or more classic ELBs to your existing Auto Scaling Groups. The ELBs must be in the same region.
- Auto Scaling rebalances by launching new EC2 instances in the AZs that have fewer instances first, only then will it start terminating instances in AZs that had more instances
- Monitoring
    - **Health checks** - identifies any instances that are unhealthy
        - Amazon EC2 status checks (default)
        - Elastic Load Balancing health checks
        - Custom health checks.
    - Auto scaling does not perform health checks on instances in the ***standby*** state. Standby state can be used for performing updates/changes/troubleshooting without health checks being performed or replacement instances being launched.
    - **CloudWatch metrics** - enables you to retrieve statistics about Auto Scaling-published data points as an ordered set of time-series data, known as *metrics*. You can use these metrics to verify that your system is performing as expected.
    - **CloudWatch Events** - Auto Scaling can submit events to CloudWatch Events when your Auto Scaling groups launch or terminate instances, or when a lifecycle action occurs.
    - **SNS notifications** - Auto Scaling can send Amazon SNS notifications when your Auto Scaling groups launch or terminate instances.
    - CloudTrail logs - enables you to keep track of the calls made to the Auto Scaling API by or on behalf of your AWS account, and stores the information in log files in an S3 bucket that you specify.
- Security
    - Use IAM to help secure your resources by controlling who can perform AWS Auto Scaling actions.
    - By default, a brand new IAM user has NO permissions to do anything. To grant permissions to call Auto Scaling actions, you attach an IAM policy to the IAM users or groups that require the permissions it grants.

Sources:
https://docs.aws.amazon.com/autoscaling/plans/userguide/what-is-aws-auto-scaling.html
https://aws.amazon.com/autoscaling/features/
https://docs.aws.amazon.com/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html

https://docs.aws.amazon.com/autoscaling/application/userguide/what-is-application-auto-scaling.html
https://aws.amazon.com/autoscaling/pricing/
https://aws.amazon.com/autoscaling/faqs/

AWS CloudFormation

- A service that gives developers and businesses an easy way to create a collection of related AWS resources and provision them in an orderly and predictable fashion.

Features

- CloudFormation allows you to model your entire infrastructure in a text file called a **template**. You can use JSON or YAML to describe what AWS resources you want to create and configure. If you want to design visually, you can use *AWS CloudFormation Designer*.
- CloudFormation automates the provisioning and updating of your infrastructure in a safe and controlled manner. You can use **Rollback Triggers** to specify the CloudWatch alarm that CloudFormation should monitor during the stack creation and update process. If any of the alarms are breached, CloudFormation rolls back the entire stack operation to a previously deployed state.
- **CloudFormation Change Sets** allow you to preview how proposed changes to a stack might impact your running resources.
- **AWS StackSets** lets you provision a common set of AWS resources across multiple accounts and regions with a single CloudFormation template. StackSets takes care of automatically and safely provisioning, updating, or deleting stacks in multiple accounts and across multiple regions.
- CloudFormation enables you to build custom extensions to your stack template using AWS Lambda.

CloudFormation vs Elastic Beanstalk

- Elastic Beanstalk provides an **environment** to easily **deploy and run** applications in the cloud.
- CloudFormation is a convenient **provisioning mechanism** for a broad range of AWS resources.

Concepts

- **Templates**
  - A JSON or YAML formatted text file.
  - CloudFormation uses these templates as blueprints for building your AWS resources.
- **Stacks**
  - Manage related resources as a single unit.
  - All the resources in a stack are defined by the stack's CloudFormation template.
- **Change Sets**
  - Before updating your stack and making changes to your resources, you can generate a change set, which is a summary of your proposed changes.
  - Change sets allow you to see how your changes might impact your running resources, especially for critical resources, before implementing them.
- With AWS CloudFormation and AWS CodePipeline, you can use continuous delivery to automatically build and test changes to your CloudFormation templates before promoting them to production stacks.

- *CloudFormation artifacts* can include a stack template file, a template configuration file, or both. AWS CodePipeline uses these artifacts to work with CloudFormation stacks and change sets.
    - **Stack Template File** - defines the resources that CloudFormation provisions and configures. You can use YAML or JSON-formatted templates.
    - **Template Configuration File** - a JSON-formatted text file that can specify template parameter values, a stack policy, and tags. Use these configuration files to specify parameter values or a stack policy for a stack.
- Through the AWS PrivateLink, you can use CloudFormation APIs inside of your Amazon VPC and route data between your VPC and CloudFormation entirely within the AWS network.

Stacks

- If a resource cannot be created, CloudFormation rolls the stack back and automatically deletes any resources that were created. If a resource cannot be deleted, any remaining resources are retained until the stack can be successfully deleted.
- Stack update methods
    - Direct update
    - Creating and executing change sets
- **Drift detection** enables you to detect whether a stack's actual configuration differs, or has drifted, from its expected configuration. Use CloudFormation to detect drift on an entire stack, or on individual resources within the stack.
    - A resource is considered to have drifted if any if its actual property values differ from the expected property values.
    - A stack is considered to have drifted if one or more of its resources have drifted.
- To share information between stacks, export a stack's output values. Other stacks that are in the same AWS account and region can import the exported values.
- You can nest stacks.

Templates

- Templates include several major sections. The Resources section is the only required section.

```
---
AWSTemplateFormatVersion: "version date"

Description:
  String

Metadata:
  template metadata

Parameters:
  set of parameters

Mappings:
  set of mappings

Conditions:
  set of conditions

Transform:
  set of transforms

Resources:
  set of resources

Outputs:
  set of outputs
```

- **CloudFormation Designer** is a graphic tool for creating, viewing, and modifying CloudFormation templates. You can diagram your template resources using a drag-and-drop interface, and then edit their details using the integrated JSON and YAML editor.
- Custom resources enable you to write custom provisioning logic in templates that CloudFormation runs anytime you create, update (if you changed the custom resource), or delete stacks.
- Template macros enable you to perform custom processing on templates, from simple actions like find-and-replace operations to extensive transformations of entire templates.

StackSets

- CloudFormation StackSets allow you to roll out CloudFormation stacks over multiple AWS accounts and in multiple Regions with just a couple of clicks. StackSets is commonly used together with AWS Organizations to centrally deploy and manage services in different accounts.
- Administrator and target accounts - An *administrator account* is the AWS account in which you create stack sets. A stack set is managed by signing in to the AWS administrator account in which it was

created. A *target account* is the account into which you create, update, or delete one or more stacks in your stack set.
- Stack sets - A *stack set* lets you create stacks in AWS accounts across regions by using a single CloudFormation template. All the resources included in each stack are defined by the stack set's CloudFormation template. A stack set is a regional resource.
- Stack instances - A *stack instance* is a reference to a stack in a target account within a region. A stack instance can exist without a stack; for example, if the stack could not be created for some reason, the stack instance shows the reason for stack creation failure. A stack instance can be associated with only one stack set.
- Stack set operations - Create stack set, update stack set, delete stacks, and delete stack set.
- Tags - You can add tags during stack set creation and update operations by specifying key and value pairs.

Monitoring

- CloudFormation is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in CloudFormation. CloudTrail captures all API calls for CloudFormation as events, including calls from the CloudFormation console and from code calls to the CloudFormation APIs.

Security

- You can use IAM with CloudFormation to control what users can do with AWS CloudFormation, such as whether they can view stack templates, create stacks, or delete stacks.
- A *service role* is an IAM role that allows CloudFormation to make calls to resources in a stack on your behalf. You can specify an IAM role that allows CloudFormation to create, update, or delete your stack resources.
- You can improve the security posture of your VPC by configuring CloudFormation to use an interface VPC endpoint.

Pricing

- No additional charge for CloudFormation. You pay for AWS resources created using CloudFormation in the same manner as if you created them manually.

Sources:
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/
https://aws.amazon.com/cloudformation/features/
https://aws.amazon.com/cloudformation/pricing/
https://aws.amazon.com/cloudformation/faqs/

AWS CloudTrail

- Actions taken by a user, role, or an AWS service in the AWS Management Console, AWS Command Line Interface, and AWS SDKs and APIs are recorded as **events**.
- CloudTrail is enabled on your AWS account when you create it.
- CloudTrail focuses on auditing API activity.
- View events in **Event History**, where you can view, search, and download the past 90 days of activity in your AWS account.
- **Trails**
    - Create a **CloudTrail trail** to archive, analyze, and respond to changes in your AWS resources.
    - Types
        - A trail that applies to **all regions** - CloudTrail records events in each region and delivers the CloudTrail event log files to an S3 bucket that you specify. This is the default option when you create a trail in the CloudTrail console.
        - A trail that applies to **one region -** CloudTrail records the events in the region that you specify only. This is the default option when you create a trail using the AWS CLI or the CloudTrail API.
    - You can create an **organization trail** that will log all events for all AWS accounts in an organization created by AWS Organizations. Organization trails must be created in the management account.
    - By default, CloudTrail event log files are encrypted using Amazon S3 server-side encryption. You can also choose to encrypt your log files with an AWS Key Management Service key.
    - You can store your log files in your S3 bucket for as long as you want, and also define S3 lifecycle rules to archive or delete log files automatically.
    - If you want notifications about log file delivery and validation, you can set up Amazon SNS notifications.
    - CloudTrail publishes log files about every five minutes.
- **Events**
    - The record of an activity in an AWS account. This activity can be an action taken by a user, role, or service that is monitorable by CloudTrail.
    - Types
        - Management events
            - Logged by default
            - Management events provide insight into management operations performed on resources in your AWS account, also known as *control plane operations*.
        - Data events
            - Not logged by default
            - Data events provide insight into the resource operations performed on or in a resource, also known as *data plane operations*.
            - Data events are often high-volume activities.

- For global services such as IAM, STS, CloudFront, and Route 53, events are delivered to any trail that includes global services, and are logged as occurring in US East (N. Virginia) Region.
- You can filter logs by specifying Time range and one of the following attributes: Event name, User name, Resource name, Event source, Event ID, and Resource type.
- **Monitoring**
    - Use **CloudWatch Logs** to monitor log data. CloudTrail events that are sent to CloudWatch Logs can trigger alarms according to the metric filters you define.
    - To determine whether a log file was modified, deleted, or unchanged after CloudTrail delivered it, you can use CloudTrail log file integrity validation.
- **Price**
    - The first copy of management events within each region is delivered free of charge. Additional copies of management events are charged.
    - Data events are recorded and charged only for the Lambda functions and S3 buckets you specify.
    - Once a CloudTrail trail is set up, S3 charges apply based on your usage, since CloudTrail delivers logs to an S3 bucket.

Sources:
https://docs.aws.amazon.com/awscloudtrail/latest/userguide/
https://aws.amazon.com/cloudtrail/features/
https://aws.amazon.com/cloudtrail/pricing/
https://aws.amazon.com/cloudtrail/faqs/

Amazon CloudWatch

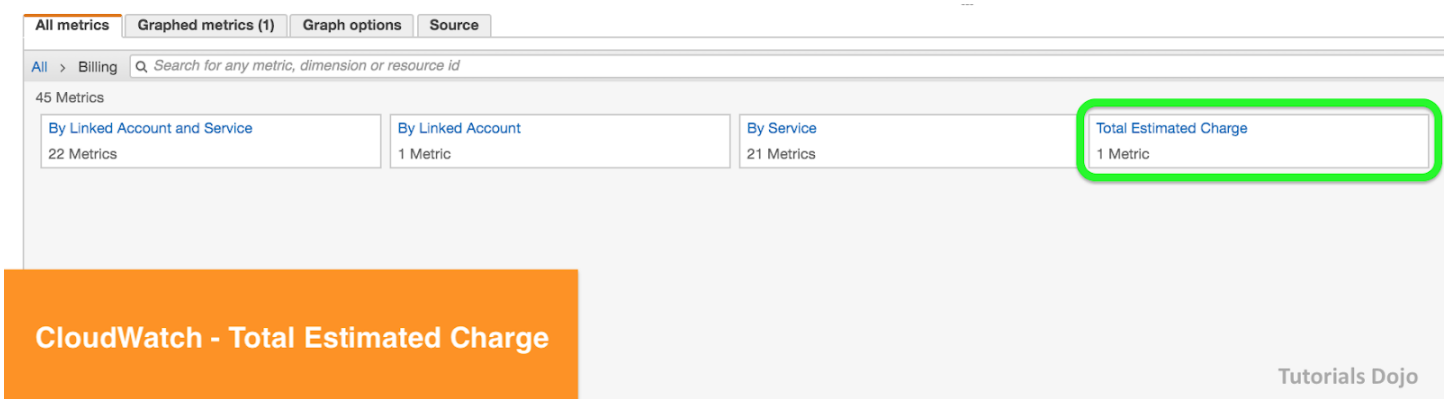
- Monitoring tool for your AWS resources and applications.
- Display metrics and create alarms that watch the metrics and send notifications or automatically make changes to the resources you are monitoring when a threshold is breached.
- CloudWatch is basically a metrics repository. An AWS service, such as Amazon EC2, puts metrics into the repository and you retrieve statistics based on those metrics. If you put your own custom metrics into the repository, you can retrieve statistics on these metrics as well.
- CloudWatch does not aggregate data across regions. Therefore, metrics are completely separate between regions.
- **CloudWatch Concepts**
    - **Namespaces** - a container for CloudWatch metrics.
        - There is no default namespace.
        - The AWS namespaces use the following naming convention: AWS/*service*.
    - **Metrics** - represents a time-ordered set of data points that are published to CloudWatch.
        - Exists only in the region in which they are created.
        - Cannot be deleted, but they automatically expire after 15 months if no new data is published to them.
        - As new data points come in, data older than 15 months is dropped.
        - Each metric data point must be marked with a *timestamp*. The timestamp can be up to two weeks in the past and up to two hours into the future. If you do not provide a timestamp, CloudWatch creates a timestamp for you based on the time the data point was received.
        - By default, several services provide free metrics for resources. You can also enable **detailed monitoring**, or publish your own application metrics.
        - **Metric math** enables you to query multiple CloudWatch metrics and use math expressions to create new time series based on these metrics.
        - **Important note for EC2 metrics:** CloudWatch does not collect memory utilization and disk space usage metrics right from the get go. You need to install CloudWatch Agent in your instances first to retrieve these metrics.
    - **Dimensions** - a name/value pair that uniquely identifies a metric.
        - You can assign up to 10 dimensions to a metric.
        - Whenever you add a unique dimension to one of your metrics, you are creating a new variation of that metric.
    - **Statistics** - metric data aggregations over specified periods of time.
        - Each statistic has a unit of measure. Metric data points that specify a unit of measure are aggregated separately.
        - You can specify a unit when you create a custom metric. If you do not specify a unit, CloudWatch uses *None* as the unit.
        - A *period* is the length of time associated with a specific CloudWatch statistic. The default value is 60 seconds.

- CloudWatch aggregates statistics according to the period length that you specify when retrieving statistics.
- For large datasets, you can insert a pre-aggregated dataset called a *statistic set*.

Statistic	Description
Minimum	The lowest value observed during the specified period. You can use this value to determine low volumes of activity for your application.
Maximum	The highest value observed during the specified period. You can use this value to determine high volumes of activity for your application.
Sum	All values submitted for the matching metric added together. Useful for determining the total volume of a metric.
Average	The value of Sum / SampleCount during the specified period. By comparing this statistic with the Minimum and Maximum, you can determine the full scope of a metric and how close the average use is to the Minimum and Maximum. This comparison helps you to know when to increase or decrease your resources as needed.
SampleCount	The count (number) of data points used for the statistical calculation.
pNN.NN	The value of the specified percentile. You can specify any percentile, using up to two decimal places (for example, p95.45). Percentile statistics are not available for metrics that include any negative values.

- **Percentiles** - indicates the relative standing of a value in a dataset. Percentiles help you get a better understanding of the distribution of your metric data.
- **Alarms** - watches a single metric over a specified time period, and performs one or more specified actions, based on the value of the metric relative to a threshold over time
  - You can create an alarm for monitoring CPU usage and load balancer latency, for managing instances, and for billing alarms.
  - When an alarm is on a dashboard, it turns red when it is in the *ALARM* state.
  - Alarms invoke actions for sustained state changes only.
  - Alarm States
    - **OK**—The metric or expression is within the defined threshold.
    - **ALARM**—The metric or expression is outside of the defined threshold.
    - **INSUFFICIENT_DATA**—The alarm has just started, the metric is not available, or not enough data is available for the metric to determine the alarm state.
  - You can also monitor your estimated AWS charges by using Amazon CloudWatch Alarms. However, take note that you can only track the estimated AWS charges in CloudWatch and

not the actual utilization of your resources. Remember that you can only set coverage targets for your reserved EC2 instances in AWS Budgets or Cost Explorer, but not in CloudWatch.



- When you create an alarm, you specify three settings:
  - **Period** is the length of time to evaluate the metric or expression to create each individual data point for an alarm. It is expressed in seconds.
  - **Evaluation Period** is the number of the most recent periods, or data points, to evaluate when determining alarm state.
  - **Datapoints to Alarm** is the number of data points within the evaluation period that must be breaching to cause the alarm to go to the ALARM state. The breaching data points do not have to be consecutive, they just must all be within the last number of data points equal to **Evaluation Period**.
  - For each alarm, you can specify CloudWatch to treat missing data points as any of the following:
    - *missing*—the alarm does not consider missing data points when evaluating whether to change state (default)
    - *notBreaching*—missing data points are treated as being within the threshold
    - *breaching*—missing data points are treated as breaching the threshold
    - *ignore*—the current alarm state is maintained
- You can now create tags in CloudWatch alarms that let you define policy controls for your AWS resources. This enables you to create resource level policies for your alarms.

CloudWatch Dashboard

- Customizable home pages in the CloudWatch console that you can use to monitor your resources in a single view, even those spread across different regions.
- There is no limit on the number of CloudWatch dashboards you can create.
- All dashboards are **global**, not region-specific.
- You can add, remove, resize, move, edit or rename a graph. You can metrics manually in a graph.

CloudWatch Events

- Deliver near real-time stream of system events that describe changes in AWS resources.

- Events respond to these operational changes and take corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information.
- Concepts
  - **Events** - indicates a change in your AWS environment.
  - **Targets** - processes events.
  - **Rules** - matches incoming events and routes them to targets for processing.

CloudWatch Logs

- Features
  - Monitor logs from EC2 instances in real-time
  - Monitor CloudTrail logged events
  - By default, logs are kept indefinitely and never expire
  - Archive log data
  - Log Route 53 DNS queries
- **CloudWatch Logs Insights** enables you to interactively search and analyze your log data in CloudWatch Logs using queries.
- **CloudWatch Vended logs** are logs that are natively published by AWS services on behalf of the customer. **VPC Flow logs** is the first Vended log type that will benefit from this tiered model.
- After the CloudWatch Logs agent begins publishing log data to Amazon CloudWatch, you can search and filter the log data by creating one or more metric filters. **Metric filters** define the terms and patterns to look for in log data as it is sent to CloudWatch Logs.
- Filters **do not** retroactively filter data. Filters only publish the metric data points for events that happen after the filter was created. Filtered results return the first 50 lines, which will not be displayed if the timestamp on the filtered results is earlier than the metric creation time.
- Metric Filter Concepts
  - filter pattern - you use the pattern to specify what to look for in the log file.
  - metric name - the name of the CloudWatch metric to which the monitored log information should be published.
  - metric namespace - the destination namespace of the new CloudWatch metric.
  - metric value - the numerical value to publish to the metric each time a matching log is found.
  - default value - the value reported to the metric filter during a period when no matching logs are found. By setting this to 0, you ensure that data is reported during every period.
- You can create two subscription filters with different filter patterns on a single log group.

CloudWatch Agent

- Collect more logs and system-level metrics from EC2 instances and your on-premises servers.
- Needs to be installed.

Authentication and Access Control

- Use IAM users or roles for authenticating who can access

- Use Dashboard Permissions, IAM identity-based policies, and service-linked roles for managing access control.
- A *permissions policy* describes who has access to what.
  - Identity-Based Policies
  - Resource-Based Policies
- There are no CloudWatch Amazon Resource Names (ARNs) for you to use in an IAM policy. Use an * (asterisk) instead as the resource when writing a policy to control access to CloudWatch actions.

Pricing

- You are charged for the number of metrics you have per month
- You are charged per 1000 metrics requested using CloudWatch API calls
- You are charged per dashboard per month
- You are charged per alarm metric (Standard Resolution and High Resolution)
- You are charged per GB of collected, archived and analyzed log data
- There is no Data Transfer IN charge, only Data Transfer Out.
- You are charged per million custom events and per million cross-account events
- Logs Insights is priced per query and charges based on the amount of ingested log data scanned by the query.

Sources:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring
https://aws.amazon.com/cloudwatch/features/
https://aws.amazon.com/cloudwatch/pricing/
https://aws.amazon.com/cloudwatch/faqs/

AWS Systems Manager

- Allows you to centralize operational data from multiple AWS services and automate tasks across your AWS resources.

Features

- Create logical groups of resources such as applications, different layers of an application stack, or production versus development environments.
- You can select a resource group and view its recent API activity, resource configuration changes, related notifications, operational alerts, software inventory, and patch compliance status.
- Collects information about your instances and the software installed on them.
- Allows you to safely automate common and repetitive IT operations and management tasks across AWS resources.
- Provides a browser-based interactive shell and CLI for managing Windows and Linux EC2 instances, without the need to open inbound ports, manage SSH keys, or use bastion hosts. Administrators can grant and revoke access to instances through a central location by using IAM policies.
- Helps ensure that your software is up-to-date and meets your compliance policies.
- Lets you schedule windows of time to run administrative and maintenance tasks across your instances.

SSM Agent is the tool that processes Systems Manager requests and configures your machine as specified in the request. SSM Agent must be installed on each instance you want to use with Systems Manager. On newer AMIs and instance types, SSM Agent is installed by default. On older versions, you must install it manually.

Capabilities

- **Automation**
  - Allows you to safely automate common and repetitive IT operations and management tasks across AWS resources
  - A **step** is defined as an initiated action performed in the Automation execution on a per-target basis. You can execute the entire Systems Manager automation document in one action or choose to execute one step at a time.
  - Concepts
    - **Automation document** - defines the Automation workflow.
    - **Automation action** - the Automation workflow includes one or more steps. Each step is associated with a particular action or plugin. The action determines the inputs, behavior, and outputs of the step.
    - **Automation queue** - if you attempt to run more than 25 Automations simultaneously, Systems Manager adds the additional executions to a queue and displays a status of *Pending*. When an Automation reaches a terminal state, the first execution in the queue starts.
  - You can schedule Systems Manager automation document execution.

- **Resource Groups**
  - A collection of AWS resources that are all in the same AWS region, and that match criteria provided in a query.
  - Use Systems Manager tools such as *Automation* to simplify management tasks on your groups of resources. You can also use groups as the basis for viewing monitoring and configuration *insights* in Systems Manager.
- **Built-in Insights**
  - Show detailed information about a single, selected resource group.
  - Includes recent API calls through CloudTrail, recent configuration changes through Config, Instance software inventory listings, instance patch compliance views, and instance configuration compliance views.
- **Systems Manager Activation**
  - Enable hybrid and cross-cloud management. You can register any server, whether physical or virtual to be managed by Systems Manager.
- **Inventory Manager**
  - Automates the process of collecting software inventory from managed instances.
  - You specify the type of metadata to collect, the instances from where the metadata should be collected, and a schedule for metadata collection.
- **Configuration Compliance**
  - Scans your fleet of managed instances for patch compliance and configuration inconsistencies.
  - View compliance history and change tracking for Patch Manager patching data and State Manager associations by using AWS Config.
  - Customize Systems Manager Compliance to create your own compliance types.
- **Run Command**
  - Remotely and securely manage the configuration of your managed instances at scale.
  - **Managed Instances** - any EC2 instance or on-premises server or virtual machine in your hybrid environment that is configured for Systems Manager.
- **Session Manager**
  - Manage your EC2 instances through an interactive one-click browser-based shell or through the AWS CLI.
  - Makes it easy to comply with corporate policies that require controlled access to instances, strict security practices, and fully auditable logs with instance access details, while still providing end users with simple one-click cross-platform access to your Amazon EC2 instances.
  - You can use AWS Systems Manager Session Manager to tunnel SSH (Secure Shell) and SCP (Secure Copy) traffic between a client and a server.
- **Distributor**
  - Lets you package your own software or find AWS-provided agent software packages to install on Systems Manager managed instances.
  - After you create a package in Distributor, which creates an Systems Manager document, you can install the package in one of the following ways.
    - One time by using Systems Manager Run Command.

        - On a schedule by using Systems Manager State Manager.
- **Patch Manager**
    - Automate the process of patching your managed instances.
    - Enables you to scan instances for missing patches and apply missing patches individually or to large groups of instances by using EC2 instance tags.
    - For security patches, Patch Manager uses *patch baselines* that include rules for auto-approving patches within days of their release, as well as a list of approved and rejected patches.
    - You can use AWS Systems Manager Patch Manager to select and apply Microsoft application patches automatically across your Amazon EC2 or on-premises instances.
    - AWS Systems Manager Patch Manager includes common vulnerability identifiers (CVE ID). CVE IDs can help you identify security vulnerabilities within your fleet and recommend patches.
- **Maintenance Window**
    - Set up recurring schedules for managed instances to execute administrative tasks like installing patches and updates without interrupting business-critical operations.
    - Supports running four types of tasks:
        - Systems Manager Run Command commands
        - Systems Manager Automation workflows
        - AWS Lambda functions
        - AWS Step Functions tasks
- **Systems Manager Document (SSM)**
    - Defines the actions that Systems Manager performs.
    - Types of SSM Documents

Type	Use with	Details
Command document	Run Command, State Manager	Run Command uses command documents to execute commands. State Manager uses command documents to apply a configuration. These actions can be run on one or more targets at any point during the lifecycle of an instance.
Policy document	State Manager	Policy documents enforce a policy on your targets. If the policy document is removed, the policy action no longer happens.
Automation document	Automation	Use automation documents when performing common maintenance and deployment tasks such as creating or updating an AMI.
Package document	Distributor	In Distributor, a package is represented by a Systems Manager document. A package document includes attached ZIP archive files that contain software or assets to install on

		managed instances. Creating a package in Distributor creates the package document.

- Can be in JSON or YAML.
- You can create and save different versions of documents. You can then specify a default version for each document.
- If you want to customize the steps and actions in a document, you can create your own.
- You can tag your documents to help you quickly identify one or more documents based on the tags you've assigned to them.

State Manager

- A service that automates the process of keeping your EC2 and hybrid infrastructure in a state that you define.
- A *State Manager association* is a configuration that is assigned to your managed instances. The configuration defines the state that you want to maintain on your instances. The association also specifies actions to take when applying the configuration.

Parameter Store

- Provides secure, hierarchical storage for configuration data and secrets management.
- You can store values as plain text or encrypted data with *SecureString*.
- Parameters work with Systems Manager capabilities such as Run Command, State Manager, and Automation.

OpsCenter

- OpsCenter helps you view, investigate, and resolve operational issues related to your environment from a central location.
- OpsCenter complements existing case management systems by enabling integrations via Amazon Simple Notification Service (SNS) and public AWS SDKs. By aggregating information from AWS Config, AWS CloudTrail logs, resource descriptions, and Amazon CloudWatch Events, OpsCenter helps you reduce the mean time to resolution (MTTR) of incidents, alarms, and operational tasks.

Monitoring

- SSM Agent writes information about executions, scheduled actions, errors, and health statuses to log files on each instance. For more efficient instance monitoring, you can configure either SSM Agent itself or the CloudWatch Agent to send this log data to CloudWatch Logs.
- Using CloudWatch Logs, you can monitor log data in real-time, search and filter log data by creating one or more metric filters, and archive and retrieve historical data when you need it.
- Log System Manager API calls with CloudTrail.

Security

- Systems Managers is linked directly to IAM for access controls.

Pricing

- For your own packages, you pay only for what you use. Upon transferring a package into Distributor, you will be charged based on the size and duration of storage for that package, the number of Get and Describe API calls made, and the amount of out-of-Region and on-premises data transfer out of Distributor for those packages.
- You are charged based on the number and type of Automation steps.

Sources:
https://docs.aws.amazon.com/systems-manager/latest/userguide
https://aws.amazon.com/systems-manager/features/
https://aws.amazon.com/systems-manager/pricing/
https://aws.amazon.com/systems-manager/faq/

AWS BILLING AND COST MANAGEMENT

- **Cost Explorer** tracks and analyzes your AWS usage. It is free for all accounts.
- Use **Budgets** to manage budgets for your account.
- Use **Bills** to see details about your current charges.
- Use **Payment History** to see your past payment transactions.
- AWS Billing and Cost Management closes the billing period at midnight on the last day of each month and then calculates your bill.
- At the end of a billing cycle or at the time you choose to incur a one-time fee, AWS charges the credit card you have on file and issues your invoice as a downloadable PDF file.
- With CloudWatch, you can create billing alerts that notify you when your usage of your services exceeds thresholds that you define.
- Use **cost allocation tags** to track your AWS costs on a detailed level. AWS provides two types of cost allocation tags, an *AWS generated tags* and *user-defined tags*.

AWS Free Tier

- When you create an AWS account, you're automatically signed up for the free tier for **12 months**.
- You can use a number of AWS services for free, as long as you haven't surpassed the allocated usage limit.
- To help you stay within the limits, you can track your free tier usage and set a **billing alarm with AWS Budgets** to notify you if you start incurring charges.

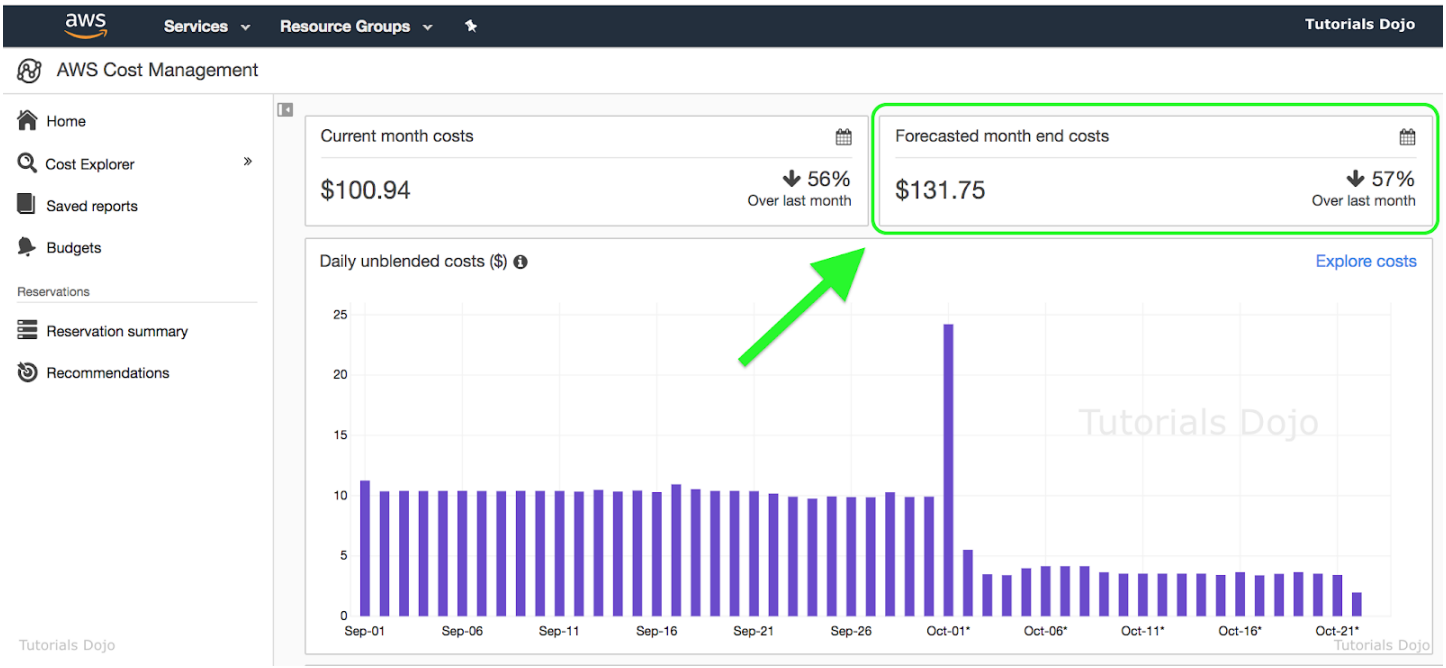
AWS Cost and Usage Reports

- The AWS Cost and Usage report provides information about your use of AWS resources and estimated costs for that usage.
- The AWS Cost and Usage report is a .csv file or a collection of .csv files that is stored in an S3 bucket. Anyone who has permissions to access the specified S3 bucket can see your billing report files.
- You can use the Cost and Usage report to track your Reserved Instance Utilization, charges, and allocations.
- Report can be automatically uploaded into AWS Redshift and/or AWS QuickSight for analysis.

AWS Cost Explorer

- Cost Explorer includes a default report that helps you visualize the costs and usage associated with your TOP FIVE cost-accruing AWS services, and gives you a detailed breakdown on all services in the table view.
- You can view data for up to the last 12 months, forecast how much you're likely to spend for the next three months, and get recommendations on what Reserved Instances to purchase.

- Cost Explorer must be enabled before it can be used. You can enable it only if you're the owner of the AWS account and you signed in to the account with your root credentials.

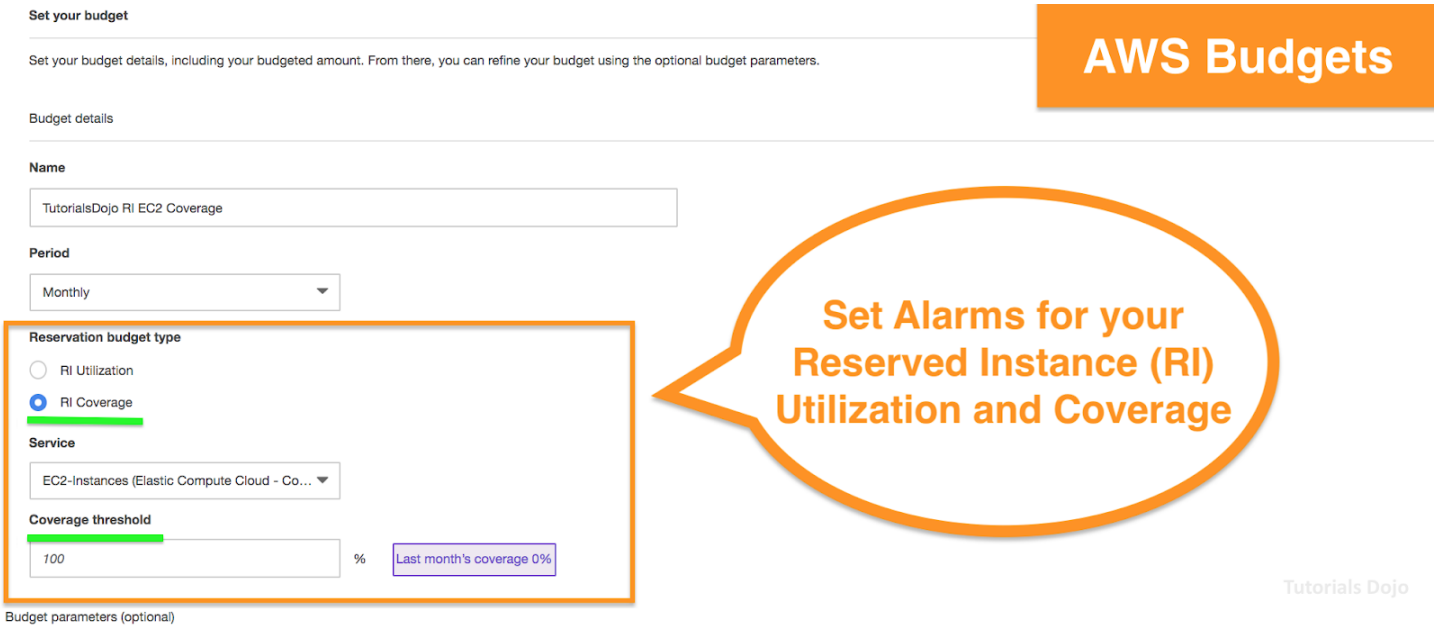


- If you're the owner of a management account in an organization, enabling Cost Explorer enables Cost Explorer for all of the organization accounts. You can't grant or deny access individually.
- You can create forecasts that predict your AWS usage and define a time range for the forecast.
- Other default reports available are:
    - The **EC2 Monthly Cost and Usage report** lets you view all of your AWS costs over the past two months, as well as your current month-to-date costs.
    - The **Monthly Costs by Linked Account report** lets you view the distribution of costs across your organization.
    - The **Monthly Running Costs report** gives you an overview of all of your running costs over the past three months, and provides forecasted numbers for the coming month with a corresponding confidence interval.

AWS Budgets

- Set custom budgets that alert you when your costs or usage exceed or are forecasted to exceed your budgeted amount.
- With Budgets, you can view the following information:
    - How close your plan is to your budgeted amount or to the free tier limits
    - Your usage to date, including how much you have used of your Reserved Instances
    - Your current estimated charges from AWS and how much your predicted usage will incur in charges by the end of the month

- How much of your budget has been used



- Budget information is updated up to three times a day.
- Types of Budgets:
  - **Cost budgets** – Plan how much you want to spend on a service.
  - **Usage budgets** – Plan how much you want to use one or more services.
  - **RI utilization budgets** – Define a utilization threshold and receive alerts when your RI usage falls below that threshold.
  - **RI coverage budgets** – Define a coverage threshold and receive alerts when the number of your instance hours that are covered by RIs fall below that threshold.
- Budgets can be tracked at the monthly, quarterly, or yearly level, and you can customize the start and end dates.
- Budget alerts can be sent via email and/or Amazon SNS topic.
- First two budgets created are free of charge.

Sources:
https://aws.amazon.com/aws-cost-management/aws-budgets/
https://aws.amazon.com/aws-cost-management/aws-cost-explorer/
https://aws.amazon.com/aws-cost-management/aws-cost-and-usage-reporting/
https://aws.amazon.com/aws-cost-management/faqs/
https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2

DEVELOPER

As an AWS Developer, you will be encountering many occasions wherein you'll create CI/CD pipelines for your workloads. AWS offers a full set of services to help you get started with building your very own pipeline and deployment scheme, with each service having its own unique features and benefits offered at a low price. These services are AWS CodeStar, AWS CodeDeploy, AWS CodeCommit, AWS CodeBuild, and AWS CodePipeline. They are built such that they can be easily integrated with one another, and they also natively support integration with other AWS services and third-party tools.

You should already know that the cloud is an important instrument in practicing DevOps, and the goal is to automate as much as possible to produce behaviors that are both predictable and reproducible. This allows organizations to innovate at a higher velocity to better serve their customers. In AWS, you will have a sense of familiarity while using their services. If you have used tools like Chef, Jenkins, Ansible, Bamboo and Git before, the same concepts apply with AWS DevOps toolkit.

AWS CodeStar is similar to a code editor that includes additional features like a project management dashboard. You can collaborate with your teammates through a unified user interface, which lets your team easily manage software development processes.

AWS CodeDeploy allows you to deploy your packages and software updates to AWS Compute services and on-premises servers. You can choose an Amazon S3 bucket or Github repository for your source of updates, customize how you want your updates to be deployed, and configure rollback procedures in case your deployment fails.

AWS CodeCommit is similar to Github in that it is a source control service where you can upload and store your code in a repository, and share it with other collaborators. It fully supports Git commands so there is no additional learning curve if you have used Git before.

AWS CodeBuild is a fully managed service that compiles your code, runs different user-defined tests for you, and produces a software package that is ready for deployment. You just need to select the OS environment where you want to test your code in, define a buildspec that will instruct CodeBuild on what to do with your code, and a location to store your finalized package.

AWS CodePipeline is a continuous delivery tool that allows you to automate your release process. You define your source of new releases, which can be AWS CodeCommit, Amazon ECR, Amazon S3 or Github, and AWS CodePipeline will automatically detect these new releases. You then define stages in your pipeline which can build and test the release, or test the release and deploy it to your machines in a controlled and predictable fashion.

These applications are fully managed and secured for you by AWS, so you can focus instead on operations and delivery to your customers.

AWS CodeDeploy

- A **fully managed deployment service** that automates software deployments to a variety of compute services such as Amazon EC2, AWS Fargate, AWS Lambda, and your on-premises servers.
- Concepts
    - An **Application** is a name that uniquely identifies the application you want to deploy. CodeDeploy uses this name, which functions as a container, to ensure the correct combination of revision, deployment configuration, and deployment group are referenced during a deployment.
    - **Compute platform** is the platform on which CodeDeploy deploys an application (EC2, ECS, Lambda, On-premises servers).
    - **Deployment configuration** is a set of deployment rules and deployment success and failure conditions used by CodeDeploy during a deployment.
    - **Deployment group** contains individually tagged instances, Amazon EC2 instances in Amazon EC2 Auto Scaling groups, or both.
        - In an Amazon ECS deployment, a deployment group specifies the Amazon ECS service, load balancer, *optional* test listener, and two target groups. It also specifies when to reroute traffic to the replacement task set and when to terminate the original task set and ECS application after a successful deployment.
        - In an AWS Lambda deployment, a deployment group defines a set of CodeDeploy configurations for future deployments of an AWS Lambda function.
        - In an EC2/On-Premises deployment, a deployment group is a set of individual instances targeted for a deployment.
            - In an in-place deployment, the instances in the deployment group are updated with the latest application revision.
            - In a blue/green deployment, traffic is rerouted from one set of instances to another by deregistering the original instances from a load balancer and registering a replacement set of instances that typically has the latest application revision already installed.
    - A **Revision**
        - for an AWS Lambda deployment is a YAML- or JSON-formatted application specification file (**AppSpec file**) that specifies information about the Lambda function to deploy. The revision can be stored in Amazon S3 buckets.
        - for an Amazon ECS deployment is a YAML- or JSON-formatted file that specifies the Amazon ECS task definition used for the deployment, a container name and port mapping used to route traffic, and optional Lambda functions run after deployment lifecycle events.
        - for an EC2/On-Premises deployment is an archive file that contains source content (source code, webpages, executable files, and deployment scripts) and an application

specification file. The revision can be stored in Amazon S3 buckets or GitHub repositories.

- **Target revision** is the most recent version of the application revision that you have uploaded to your repository and want to deploy to the instances in a deployment group.
- A deployment goes through a set of predefined phases called **deployment lifecycle events**. A deployment lifecycle event gives you an opportunity to run code as part of the deployment.
  - ApplicationStop
  - DownloadBundle
  - BeforeInstall
  - Install
  - AfterInstall
  - ApplicationStart
  - ValidateService
- Features
  - CodeDeploy protects your application from downtime during deployments through **rolling updates** and **deployment health tracking**.
  - AWS CodeDeploy tracks and stores the recent history of your deployments.
  - CodeDeploy is platform and language agnostic.
  - CodeDeploy uses a **file and command-based install** model, which enables it to deploy any application and reuse existing setup code. The same setup code can be used to consistently deploy and test updates across your environment release stages for your servers or containers.
  - CodeDeploy integrates with Amazon Auto Scaling, which allows you to scale EC2 capacity according to conditions you define such as traffic spikes. Notifications are then sent to AWS CodeDeploy to initiate an application deployment onto new instances before they are placed behind an Elastic Load Balancing load balancer.
  - When using AWS CodeDeploy with on-premises servers, make sure that they can connect to AWS public endpoints.
  - AWS CodeDeploy offers two types of deployments:
    - With **in-place deployments**, the application on each instance in the deployment group is stopped, the latest application revision is installed, and the new version of the application is started and validated. Only deployments that use the EC2/On-Premises compute platform can use in-place deployments.
    - With **blue/green deployments**, once the new version of your application is tested and declared ready, CodeDeploy can shift the traffic from your old version (blue) to your new version (green) according to your specifications.
  - **Deployment groups** are used to match configurations to specific environments, such as a staging or production environments. An application can be deployed to multiple deployment groups.
  - You can integrate AWS CodeDeploy with your continuous integration and deployment systems by calling the public APIs using the AWS CLI or AWS SDKs.

Application Specification Files

- The AppSpec file is a YAML-formatted or JSON-formatted file that is used to manage each deployment as a series of lifecycle event hooks.
- For ECS Compute platform, the file specifies
  - The name of the ECS service and the container name and port used to direct traffic to the new task set.
  - The functions to be used as validation tests.
- For Lambda compute platform, the file specifies
  - The AWS Lambda function version to deploy.
  - The functions to be used as validation tests.
- For EC2/On-Premises compute platform, the file is always written in YAML and is used to
  - Map the source files in your application revision to their destinations on the instance.
  - Specify custom permissions for deployed files.
  - Specify scripts to be run on each instance at various stages of the deployment process.

Deployments

- You can use the CodeDeploy console or the create-deployment command to deploy the function revision specified in the AppSpec file to the deployment group.
- You can use the CodeDeploy console or the stop-deployment command to stop a deployment. When you attempt to stop the deployment, one of three things happens:
  - The deployment stops, and the operation returns a status of succeeded.
  - The deployment does not immediately stop, and the operation returns a status of pending. After the pending operation is complete, subsequent calls to stop the deployment return a status of succeeded.
  - The deployment cannot stop, and the operation returns an error.
- With Lambda functions and EC2 instances, CodeDeploy implements **rollbacks by redeploying**, as a new deployment, a previously deployed revision.
- With ECS services, CodeDeploy implements **rollbacks by rerouting traffic** from the replacement task set to the original task set.
- The **CodeDeploy agent** is a software package that, when installed and configured on an EC2/on-premises instance, makes it possible for that instance to be used in CodeDeploy deployments. The agent is not required for deployments that use the Amazon ECS or AWS Lambda.
- CodeDeploy monitors the health status of the instances in a deployment group. For the overall deployment to succeed, CodeDeploy must be able to deploy to each instance in the deployment and deployment to at least one instance must succeed.
- You can specify a minimum number of healthy instances as a number of instances or as a percentage of the total number of instances required for the deployment to be successful.
- CodeDeploy assigns two health status values to each instance:

        - Revision health - based on the application revision currently installed on the instance. Values include *Current, Old* and *Unknown*.
        - Instance health - based on whether deployments to an instance have been successful. Values include *Healthy* and *Unhealthy.*
    - Blue/Green Deployments
        - EC2/On-Premises compute platform
            - You must have one or more Amazon EC2 instances with identifying Amazon EC2 tags or an Amazon EC2 Auto Scaling group.
            - Each Amazon EC2 instance must have the correct IAM instance profile attached.
            - The CodeDeploy agent must be installed and running on each instance.
            - During replacement, you can either
                - use the Amazon EC2 Auto Scaling group you specify as a template for the replacement environment; or
                - specify the instances to be counted as your replacement using EC2 instance tags, EC2 Auto Scaling group names, or both.
        - AWS Lambda platform
            - You must choose one of the following deployment configuration types to specify how traffic is shifted from the original Lambda function version to the new version:
                - Canary: Traffic is shifted in two increments. You can choose from predefined canary options that specify the percentage of traffic shifted to your updated Lambda function version in the first increment and the interval, in minutes, before the remaining traffic is shifted in the second increment.
                - Linear: Traffic is shifted in equal increments with an equal number of minutes between each increment. You can choose from predefined linear options that specify the percentage of traffic shifted in each increment and the number of minutes between each increment.
                - All-at-once: All traffic is shifted from the original Lambda function to the updated Lambda function version all at once.
        - With Amazon ECS, production traffic shifts from your ECS service's original task set to a replacement task set all at once.
    - Advantages of using Blue/Green Deployments vs In-Place Deployments
        - An application can be installed and tested in the new replacement environment and deployed to production simply by rerouting traffic.
        - If you're using the EC2/On-Premises compute platform, switching back to the most recent version of an application is faster and more reliable. Traffic can just be routed back to the original instances as long as they have not been terminated. With an in-place deployment, versions must be rolled back by redeploying the previous version of the application.

        - If you're using the EC2/On-Premises compute platform, new instances are provisioned and contain the most up-to-date server configurations.
        - If you're using the AWS Lambda compute platform, you control how traffic is shifted from your original AWS Lambda function version to your new AWS Lambda function version.
    - Deployment Configurations
        - This is a set of rules and success and failure conditions used by CodeDeploy during a deployment.
        - For EC2/On-Premises Compute Platform
            - The deployment configuration specifies the number or percentage of instances that must remain available at any time during a deployment.
            - You can use one of the three predefined deployment configurations provided by AWS or create a custom deployment configuration.

Deployment Configuration	Description
CodeDeployDefault.AllAtOnce	**In-place deployments**: Attempts to deploy an application revision to as many instances as possible at once. The status of the overall deployment is displayed as **Succeeded** if the application revision is deployed to one or more of the instances. The status of the overall deployment is displayed as **Failed** if the application revision is not deployed to any of the instances. **Blue/green deployments**: • Deployment to replacement environment: Follows the same deployment rules as CodeDeployDefault.AllAtOnce for in-place deployments. • Traffic rerouting: Routes traffic to all instances in the replacement environment at once. Succeeds if traffic is successfully rerouted to at least one instance. Fails after rerouting to all instances fails.

CodeDeployDefault.HalfAtATime	**In-place deployments**: Deploys to up to half of the instances at a time (fractions rounded down). The overall deployment succeeds if the application revision is deployed to at least half of the instances (fractions rounded up). Otherwise, the deployment fails. **Blue/green deployments**: • Deployment to replacement environment: Follows the same deployment rules as CodeDeployDefault.HalfAtATime for in-place deployments. • Traffic rerouting: Routes traffic to up to half the instances in the replacement environment at a time. Succeeds if rerouting to at least half of the instances succeeds. Otherwise, fails.
CodeDeployDefault.OneAtATime	**In-place deployments**: Deploys the application revision to only one instance at a time. For deployment groups that contain only one instance, the overall deployment is successful only if deployment to the single instance is successful. For deployment groups that contain more than one instance: • The overall deployment succeeds if the application revision is deployed to all of the instances. An exception is if deployment to the last instance fails, the overall deployment still succeeds. This is because CodeDeploy allows only one instance at a time to be taken offline with the CodeDeployDefault.OneAtATime configuration. • The overall deployment fails as soon as the application revision fails to be deployed to any but the last instance. **Blue/green deployments**: • Deployment to replacement environment: Follows same deployment rules as CodeDeployDefault.OneAtATime for in-place deployments.

	• Traffic rerouting: Routes traffic to one instance in the replacement environment at a time. Succeeds if traffic is successfully rerouted to all replacement instances. Fails after the very first rerouting failure. An exception is if the last instance fails to register, the overall deployment still succeeds.

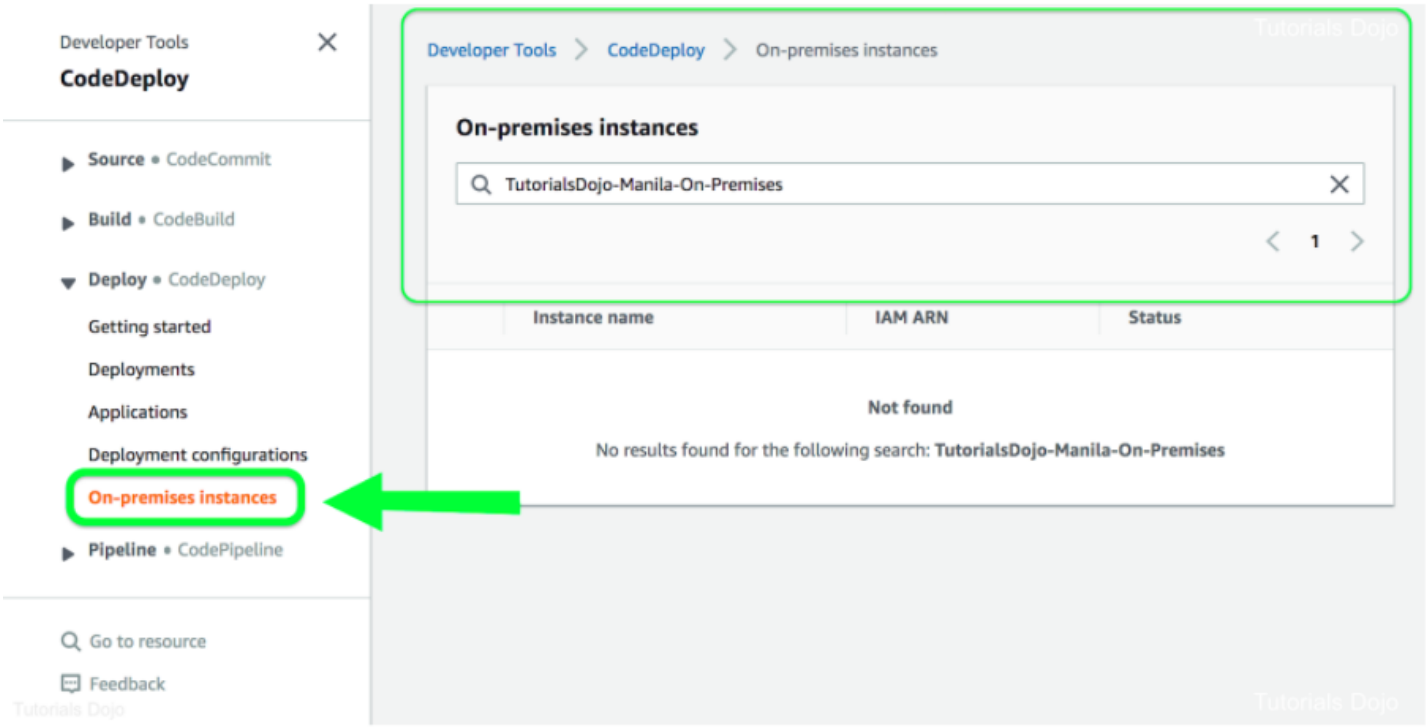
        - If you don't specify a deployment configuration, CodeDeploy uses the CodeDeployDefault.OneAtATime deployment configuration.
    - For ECS Compute Platform
        - There are three ways traffic can be shifted during a deployment:
            - **Canary**: Traffic is shifted in two increments. You can choose from predefined canary options that specify the percentage of traffic shifted to your updated Amazon ECS task set in the first increment and the interval, in minutes, before the remaining traffic is shifted in the second increment.
            - **Linear**: Traffic is shifted in equal increments with an equal number of minutes between each increment. You can choose from predefined linear options that specify the percentage of traffic shifted in each increment and the number of minutes between each increment.
            - **All-at-once**: All traffic is shifted from the original Amazon ECS task set to the updated Amazon ECS task set all at once.
        - You can also create your own custom canary or linear deployment configuration.

Deployment Configuration	Description
CodeDeployDefault.ECSLinear10PercentEveryXMinutes (Values of X :1, 3)	Shifts 10 percent of traffic every X minutes until all traffic is shifted.
CodeDeployDefault.ECSCanary10PercentXMinutes (Values of X: 5,15)	Shifts 10 percent of traffic in the first increment. The remaining 90 percent is deployed X minutes later.
CodeDeployDefault.ECSAllAtOnce	Shifts all traffic to the updated Amazon ECS container at once.

    - For Lambda Compute Platform

Deployment Configuration	Description
CodeDeployDefault.LambdaCanary10PercentXMinutes (Values of X: 5, 10, 15, 30)	Shifts 10 percent of traffic in the first increment. The remaining 90 percent is deployed X minutes later.
CodeDeployDefault.LambdaLinear10PercentEveryXMinutes (Values of X: 1, 2, 3, 10)	Shifts 10 percent of traffic every X minutes until all traffic is shifted.
CodeDeployDefault.LambdaAllAtOnce	Shifts all traffic to the updated Lambda functions at once.

- With AWS CodeDeploy, you can also deploy your applications to your on-premises data centers.



- Monitoring

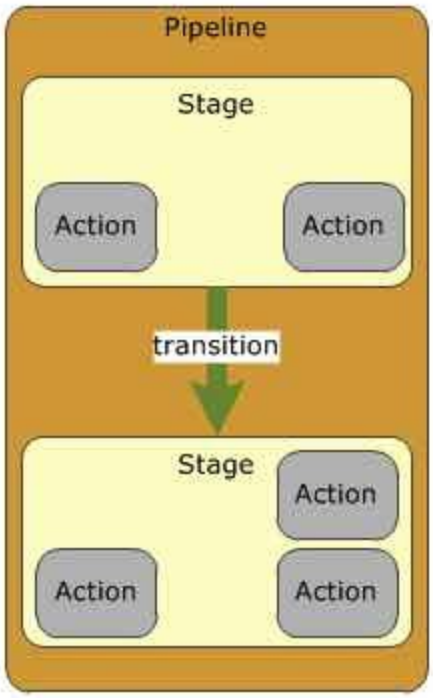
    - In CodeDeploy, you should at the minimum monitor the following items
        - Deployment events and status
        - Instance events and status
    - Tools and Services
        - Amazon CloudWatch Alarms, Events and Logs
        - AWS CloudTrail
        - Amazon SNS
        - AWS CodeDeploy console
- Pricing
    - There is no additional charge for code deployments to Amazon EC2 or AWS Lambda.
    - You are charged per on-premises instance update using AWS CodeDeploy.

Sources:
https://aws.amazon.com/codedeploy/features/?nc=sn&loc=2
https://docs.aws.amazon.com/codedeploy/latest/userguide/welcome.html
https://aws.amazon.com/codedeploy/faqs/?nc=sn&loc=6

AWS CodePipeline

- A fully managed **continuous delivery service** that helps you automate your release pipelines for application and infrastructure updates.
- You can easily integrate AWS CodePipeline with third-party services such as GitHub or with your own custom plugin.
- Concepts
    - A **pipeline** defines your release process workflow, and describes how a new code change progresses through your release process.
    - A pipeline comprises a series of **stages** (e.g., build, test, and deploy), which act as logical divisions in your workflow. Each stage is made up of a sequence of actions, which are tasks such as building code or deploying to test environments.
        - Pipelines must have **at least two stages**. The first stage of a pipeline is required to be a source stage, and the pipeline is required to additionally have at least one other stage that is a build or deployment stage.
    - Define your pipeline structure through a **declarative JSON** document that specifies your release workflow and its stages and actions. These documents enable you to update existing pipelines as well as provide starting templates for creating new pipelines.
    - A **revision** is a change made to the source location defined for your pipeline. It can include source code, build output, configuration, or data. A pipeline can have multiple revisions flowing through it at the same time.
    - A **stage** is a group of one or more actions. A pipeline can have two or more stages.
    - An **action** is a task performed on a revision. Pipeline actions occur in a specified order, in serial or in parallel, as determined in the configuration of the stage.
        - You can add actions to your pipeline that are in an AWS Region different from your pipeline.
        - There are six types of actions
            - Source
            - Build
            - Test
            - Deploy
            - Approval
            - Invoke
    - When an action runs, it acts upon a file or set of files called **artifacts**. These artifacts can be worked upon by later actions in the pipeline. You have an artifact store which is an S3 bucket in the same AWS Region as the pipeline to store items for all pipelines in that Region associated with your account.
    - The stages in a pipeline are connected by **transitions.** Transitions can be disabled or enabled between stages. If all transitions are enabled, the pipeline runs continuously.

- An **approval action** prevents a pipeline from transitioning to the next action until permission is granted. This is useful when you are performing code reviews before code is deployed to the next stage.



- Features
  - AWS CodePipeline provides you with a graphical user interface to create, configure, and manage your pipeline and its various stages and actions.
  - A pipeline starts automatically (default) when a change is made in the source location, or when you manually start the pipeline. You can also set up a rule in CloudWatch to automatically start a pipeline when events you specify occur.
  - You can model your build, test, and deployment actions to run **in parallel** in order to increase your workflow speeds.
  - AWS CodePipeline can pull source code for your pipeline directly from AWS CodeCommit, GitHub, Amazon ECR, or Amazon S3.
  - It can run builds and unit tests in AWS CodeBuild.
  - It can deploy your changes using AWS CodeDeploy, AWS Elastic Beanstalk, Amazon ECS, AWS Fargate, Amazon S3, AWS Service Catalog, AWS CloudFormation, and/or AWS OpsWorks Stacks.
  - You can use the CodePipeline Jenkins plugin to easily register your existing build servers as a custom action.

    - When you use the console to create or edit a pipeline that has a GitHub source, CodePipeline creates a **webhook**. A webhook is an HTTP notification that detects events in another tool, such as a GitHub repository, and connects those external events to a pipeline. CodePipeline deletes your webhook when you delete your pipeline.
- As a best practice, when you use a Jenkins build provider for your pipeline's build or test action, install Jenkins on an Amazon EC2 instance and configure a separate EC2 instance profile. Make sure the instance profile grants Jenkins only the AWS permissions required to perform tasks for your project, such as retrieving files from Amazon S3.
- AWS CodePipeline now supports **Amazon VPC endpoints powered by AWS PrivateLink**. This means you can connect directly to CodePipeline through a private endpoint in your VPC, keeping all traffic inside your VPC and the AWS network.
- You can view details for each of your pipelines, including when actions last ran in the pipeline, whether a transition between stages is enabled or disabled, whether any actions have failed, and other information. You can also view a history page that shows details for all pipeline executions for which history has been recorded. Execution history is retained for up to 12 months.
- Limits
    - Maximum number of total pipelines per Region in an AWS account is 300
    - Number of stages in a pipeline is minimum of 2, maximum of 10
- Pricing
    - You are charged per active pipeline each month. Newly created pipelines are free to use during the first 30 days after creation.

Sources:
https://aws.amazon.com/codepipeline/features/?nc=sn&loc=2
https://aws.amazon.com/codepipeline/pricing/?nc=sn&loc=3
https://docs.aws.amazon.com/codepipeline/latest/userguide/welcome.html
https://aws.amazon.com/codepipeline/faqs/?nc=sn&loc=5

AWS CodeBuild

- A fully managed **continuous integration service** that compiles source code, runs tests, and produces software packages that are ready to deploy.
- Concepts
  - A **build project** defines how CodeBuild will run a build. It includes information such as where to get the source code, which build environment to use, the build commands to run, and where to store the build output.
  - A **build environment** is the combination of operating system, programming language runtime, and tools used by CodeBuild to run a build.
  - The **build specification** is a YAML file that lets you choose the commands to run at each phase of the build and other settings. Without a build spec, CodeBuild cannot successfully convert your build input into build output or locate the build output artifact in the build environment to upload to your output bucket.
    - If you include a build spec as part of the source code, by default, the build spec file must be named buildspec.yml and placed in the root of your source directory.
  - A collection of input files is called **build input artifacts or build input** and a deployable version of a source code is called **build output artifact or build output**.
- Features
  - AWS CodeBuild runs your builds in preconfigured build environments that contain the operating system, programming language runtime, and build tools (such as Apache Maven, Gradle, npm) required to complete the task. You just specify your source code's location and select settings for your build, such as the build environment to use and the build commands to run during a build.
  - AWS CodeBuild builds your code and stores the artifacts into an Amazon S3 bucket, or you can use a build command to upload them to an artifact repository.
  - AWS CodeBuild provides build environments for
    - Java
    - Python
    - Node.js
    - Ruby
    - Go
    - Android
    - .NET Core for Linux
    - Docker
  - You can define the specific commands that you want AWS CodeBuild to perform, such as installing build tool packages, running unit tests, and packaging your code.
  - You can choose from **three levels of compute capacity** that vary by the amount of CPU and memory to best suit to your development needs.
    - Build.general1.small - 3GB memory, 2 vCPU

        - Build.general1.medium - 7GB memory, 4 vCPU
        - Build.general1.large - 15GB memory, 8 vCPU
    - You can integrate CodeBuild into existing CI/CD workflows using its source integrations, build commands, or Jenkins integration.
    - CodeBuild can connect to AWS CodeCommit, S3, GitHub, and GitHub Enterprise and Bitbucket to pull source code for builds.
    - CodeBuild allows you to use Docker images stored in another AWS account as your build environment, by granting resource level permissions.
    - It now allows you to access Docker images from any private registry as the build environment. Previously, you could only use Docker images from public DockerHub or Amazon ECR in CodeBuild.
    - You can access your past build results through the console, CloudWatch, or the API. The results include outcome (success or failure), build duration, output artifact location, and log location.
    - You can automate your release process by using **AWS CodePipeline** to test your code and run your builds with CodeBuild.
- Steps in a Build Process
    - CodeBuild will create a temporary compute container of the class defined in the build project
    - CodeBuild loads it with the specified runtime environment
    - CodeBuild downloads the source code
    - CodeBuild executes the commands configured in the project
    - CodeBuild uploads the generated artifact to an S3 bucket
    - Then it destroys the compute container
- Build Duration is calculated in minutes, from the time you submit your build until your build is terminated, rounded up to the nearest minute.
- You can save time when your project builds by using a cache. A build project can use one of two types of caching:
    - Amazon S3 - stores the cache in an Amazon S3 bucket that is available across multiple build hosts. This is a good option for small intermediate build artifacts that are more expensive to build than to download. Not the best option for large build artifacts because they can take a long time to transfer over your network, which can affect build performance.
    - Local - stores a cache locally on a build host that is available to that build host only. This is a good option for large intermediate build artifacts because the cache is immediately available on the build host. Build performance is not impacted by network transfer time.
    - If you use a local cache, you must choose one or more of three cache modes:
        - source cache
        - Docker layer cache
        - custom cache.
- Monitoring and Security
    - You can specify a key stored in the AWS Key Management Service to encrypt your artifacts.
    - CodeBuild provides security and separation at the infrastructure and execution levels.

    - You can use Amazon CloudWatch to watch your builds, report when something is wrong, and take automatic actions when appropriate.
    - You can monitor your builds at two levels:
        - At the project level: These metrics are for all builds in the specified project only.
        - At the AWS account level: These metrics are for all builds in one account
    - *ProjectName* is the only AWS CodeBuild metrics dimension. If it is specified, then the metrics are for that project. If it is not specified, then the metrics are for the current AWS account.
- Pricing
    - You are charged for compute resources based on the duration it takes for your build to execute. The per-minute rate depends on the compute type you use.

Sources:
https://aws.amazon.com/codebuild/features/?nc=sn&loc=2
https://aws.amazon.com/codebuild/pricing/?nc=sn&loc=3
https://aws.amazon.com/codebuild/faqs/?nc=sn&loc=5
https://docs.aws.amazon.com/codebuild/latest/userguide/getting-started.html

AWS CodeCommit

- A **fully-managed source control** service that hosts secure Git-based repositories, similar to Github.
- You can create your own code repository and use Git commands to interact with your own repository and other repositories.
- You can store and version any kind of file, including application assets such as images and libraries alongside your code.
- The AWS CodeCommit Console lets you visualize your code, pull requests, commits, branches, tags and other settings.
- Concepts
    - An *active user* is any unique AWS identity (IAM user/role, federated user, or root account) that accesses AWS CodeCommit repositories during the month. AWS identities that are created through your use of other AWS Services, such as AWS CodeBuild and AWS CodePipeline, as well as servers accessing CodeCommit using a unique AWS identity, count as active users.
    - A **repository** is the fundamental version control object in CodeCommit. It's where you securely store code and files for your project. It also stores your project history, from the first commit through the latest changes.
    - A **file** is a version-controlled, self-contained piece of information available to you and other users of the repository and branch where the file is stored.
    - A **pull request** allows you and other repository users to review, comment on, and merge code changes from one branch to another.
    - An **approval rule** is used to designate a number of users who will approve a pull request before it is merged into your branch.
    - A **commit** is a snapshot of the contents and changes to the contents of your repository. This includes information like who committed the change, the date and time of the commit, and the changes made as part of the commit.
    - In Git, **branches** are simply pointers or references to a commit. You can use branches to separate work on a new or different version of files without impacting work in other branches. You can use branches to develop new features, store a specific version of your project from a particular commit, etc.
- Repository Features
    - You can share your repository with other users.
    - If you add AWS tags to repositories, you can set up notifications so that repository users receive email about events, such as another user commenting on code.
    - You can create triggers for your repository so that code pushes or other events trigger actions, such as emails or code functions.
    - To copy a remote repository to your local computer, use the command ‘git clone’
    - To connect to the repository after the name is changed, users must use the ‘git remote set-url’ command and specify the new URL to use.

    - To push changes from the local repo to the CodeCommit repository, run 'git push *remote-name branch-name*'.
    - To pull changes to the local repo from the CodeCommit repository, run 'git pull *remote-name branch-name*'.
    - You can create up to 10 triggers for Amazon SNS or AWS Lambda for each CodeCommit repository.
    - You can push your files to two different repositories at the same time.
- File Features
    - You can organize your repository files with a directory structure.
    - CodeCommit automatically tracks every change to a file.
    - You can use the AWS Console or AWS CLI and the 'put-file' command to add a file or submit changes to a file in a CodeCommit repository.
- Pull Requests
    - Pull requests require two branches: a source branch that contains the code you want reviewed, and a destination branch, where you merge the reviewed code.
    - Create pull requests to let other users see and review your code changes before you merge them into another branch.
    - Create approval rules for your pull requests to ensure the quality of your code by requiring users to approve the pull request before the code can be merged into the destination branch. You can specify the number of users who must approve a pull request. You can also specify an approval pool of users for the rule.
    - To review the changes on files included in a pull request and resolve merge conflicts, you use the CodeCommit console, the 'git diff' command, or a diff tool.
    - After the changes have been reviewed and all approval rules on the pull request have been satisfied, you can merge a pull request using the AWS Console, AWS CLI or with the 'git merge' command.
    - You can close a pull request without merging it with your code.
- Commit and Branch Features
    - If using the AWS CLI, you can use the 'create-commit' command to create a new commit for your branch.
    - If using the AWS CLI, you can use 'create-branch' command to create a new branch for your repository..
    - You can also use Git commands to manage your commits and branches.
    - Create a new commit to a branch using 'git commit -m *message*'.
    - Create a branch in your local repo by running the git checkout -b *new-branch-name* command.
- Migration from Git repositories to CodeCommit
    - You can migrate a Git repository to a CodeCommit repository in a number of ways: by cloning it, mirroring it, or migrating all or just some of the branches.
    - You can also migrate your local repository in your machine to CodeCommit.
- High Availability
    - CodeCommit stores your repositories in Amazon S3 and Amazon DynamoDB.

- Security
    - You can transfer your files to and from AWS CodeCommit using HTTPS or SSH. For HTTPS authentication, you use a static username and password. For SSH authentication, you use public keys and private keys.
    - Your repositories are also automatically encrypted at rest through AWS KMS using customer-specific keys.
    - You can give users temporary access to your AWS CodeCommit repositories through a number of methods: SAML, Federation, Cross-Account Access, or through third party authorizes with Amazon Cognito.
- Monitoring
    - CodeCommit uses AWS IAM to control and monitor who can access your data as well as how, when, and where they can access it.
    - CodeCommit helps you monitor your repositories via AWS CloudTrail and AWS CloudWatch.
    - You can use Amazon SNS to receive notifications for events impacting your repositories. Each notification will include a status message as well as a link to the resources whose event generated that notification.
- Pricing
    - The first 5 active users per month are free of charge. You also get to have unlimited repositories, with 50 GB-month total worth of storage, and 10,000 Git requests/month at no cost.
    - You are billed for each active user beyond the first 5 per month. You also get an additional 10GB-month of storage per active user, and an additional 2,000 Git requests per active user.
        - A Git request includes any push or pull that transmits repository objects, including a direct edit in the console or through the CodeCommit APIs.

Sources:
https://aws.amazon.com/codecommit/
https://docs.aws.amazon.com/codecommit/latest/userguide/welcome.html
https://aws.amazon.com/codecommit/faqs/

AWS CodeCommit Repository

Amazon Web Services, with its ever-growing breadth of services, offers a fully-managed version control system where developers can privately store their application source code like Github or Bitbucket. CodeCommit can be used as a staging ground coupled with CodeDeploy and CodePipeline to seamlessly deploy code to Amazon EC2 instances.

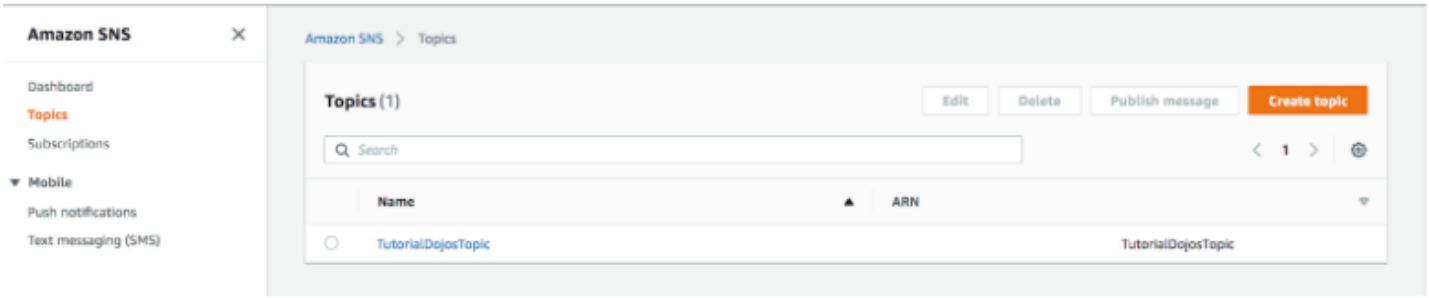
In this article, I will discuss how we can leverage Amazon SNS to send notifications whenever there are events in our repository. This is particularly useful when we need to keep our work colleagues up-to-date if there are events in our repository.

Triggers are used for two things. The first is for notifying users of an event that occurs in our repository by using a simple email notification. Second is by triggering a function to allow us to interact with third-party applications such as Jenkins or other continuous integration and deployment services.

To get started, we need to create an SNS topic (TutorialsDojoTopic) and subscription for the email notification. AWS will send a confirmation email to the subscribed email address.

Create a topic:

Amazon SNS > Topics > Create topic

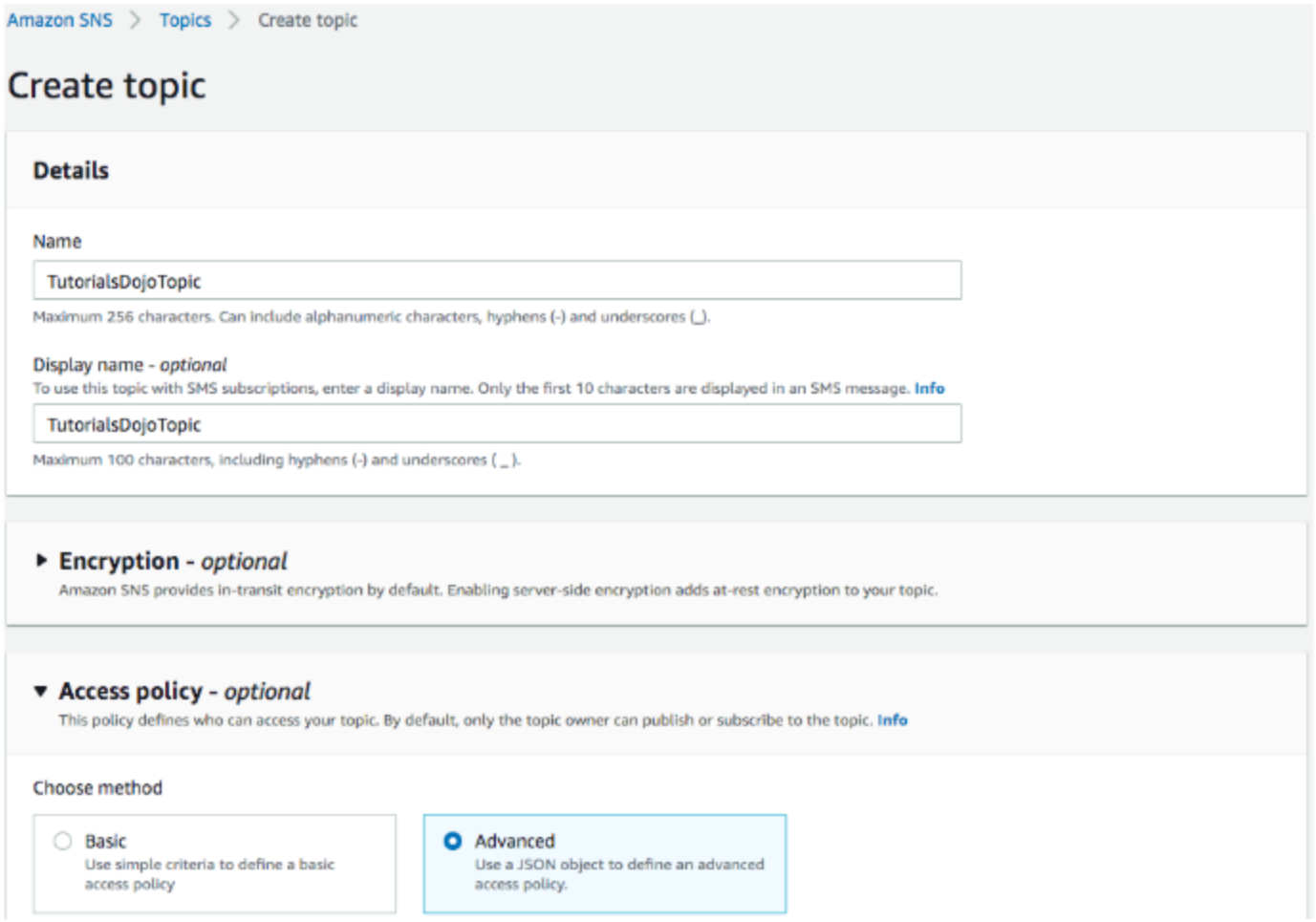


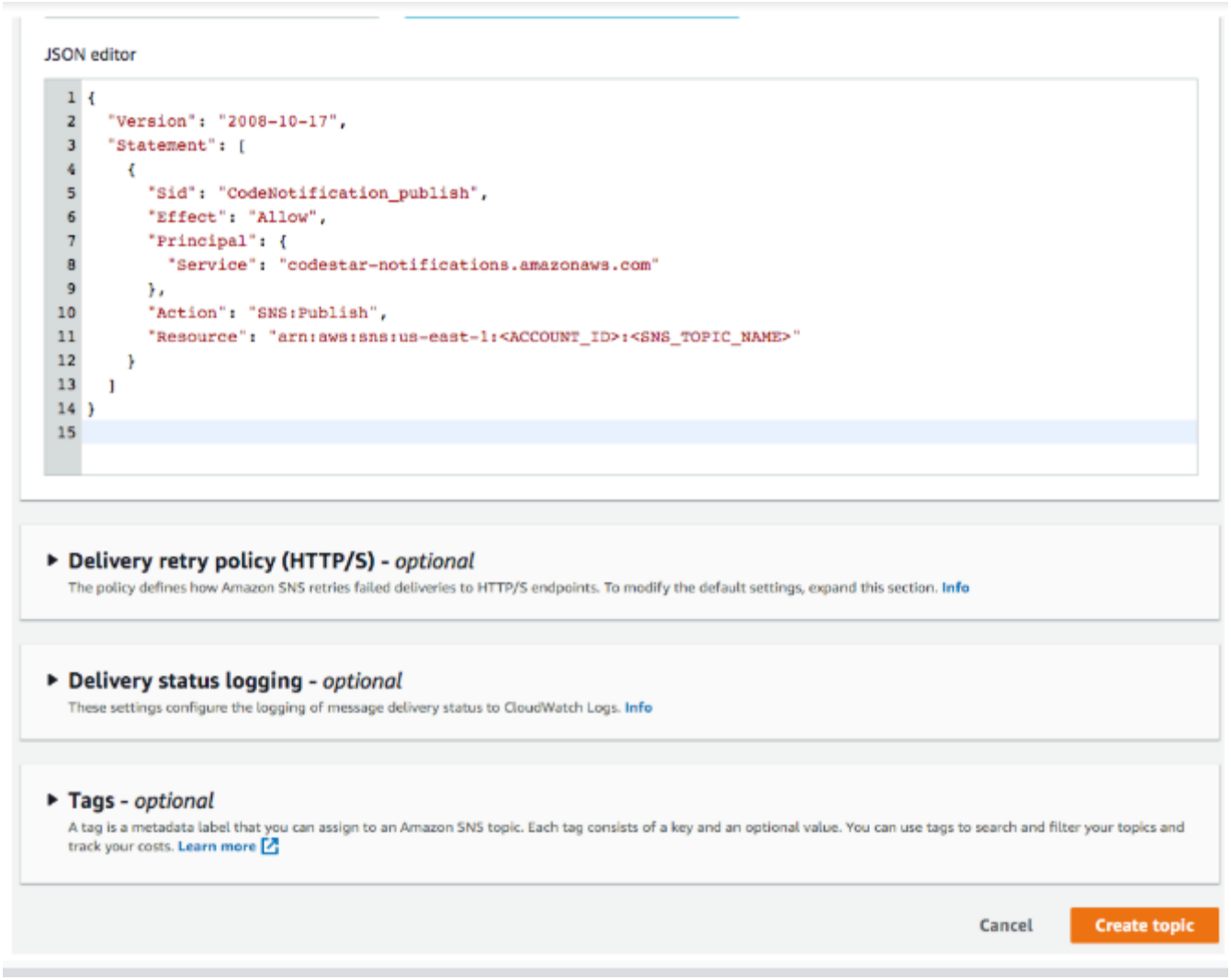
In order for Codecommit Notifications to have the necessary permissions to publish emails, you need to modify the Access Policy with the following:

```
{
  "Version": "2008-10-17",
  "Statement": [
{
   "Sid": "CodeNotification_publish",
   "Effect": "Allow",
   "Principal": {
          "Service": "codestar-notifications.amazonaws.com"
   },
   "Action": "SNS:Publish",
   "Resource": "arn:aws:sns:us-east-1:<ACCOUNT_ID>:<SNS_TOPIC_NAME>"
```

```
}
  ]
}
```

Do not forget to change the account ID and SNS topic name. Remove the < and > characters once done.

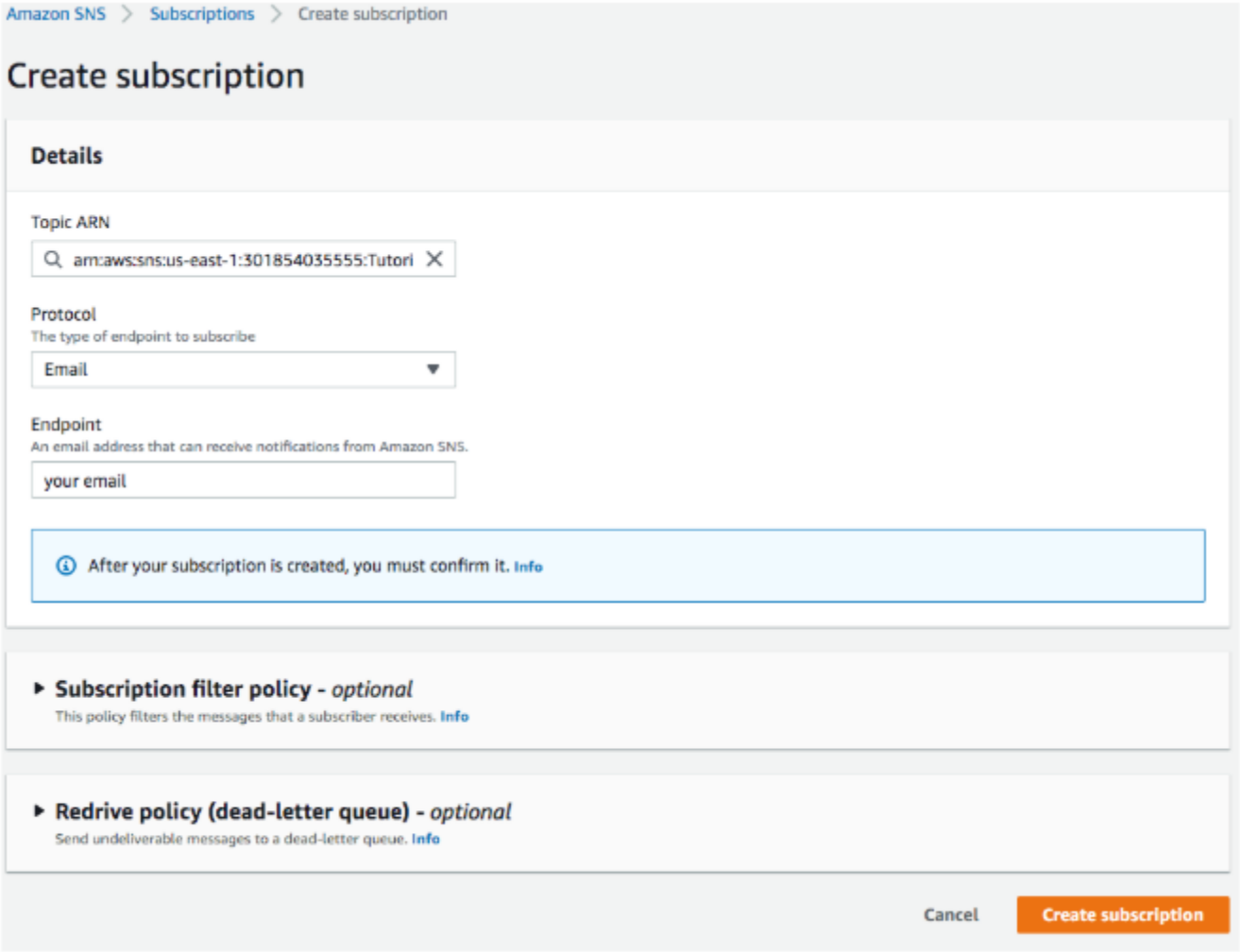




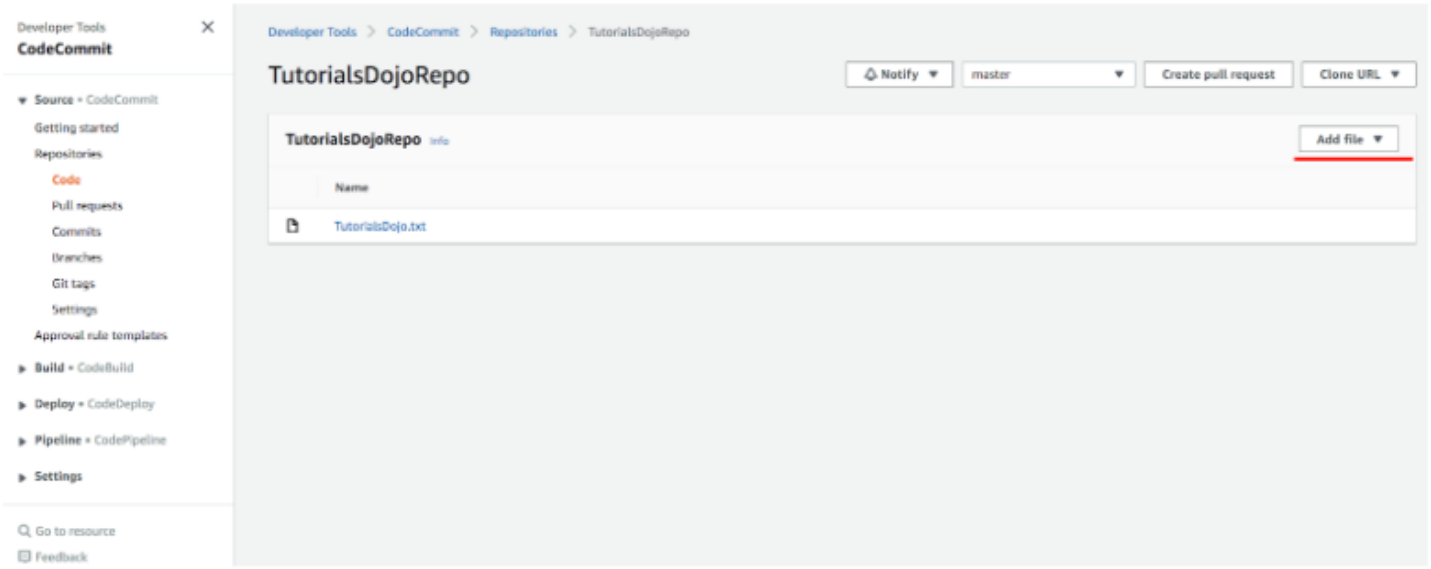
After creating our SNS topic, we need a subscription. The purpose of the subscription is it specifies what the SNS topic will do when it is triggered to a specified endpoint like Lambda, SQS, or an email address.

Create a subscription, navigate to:

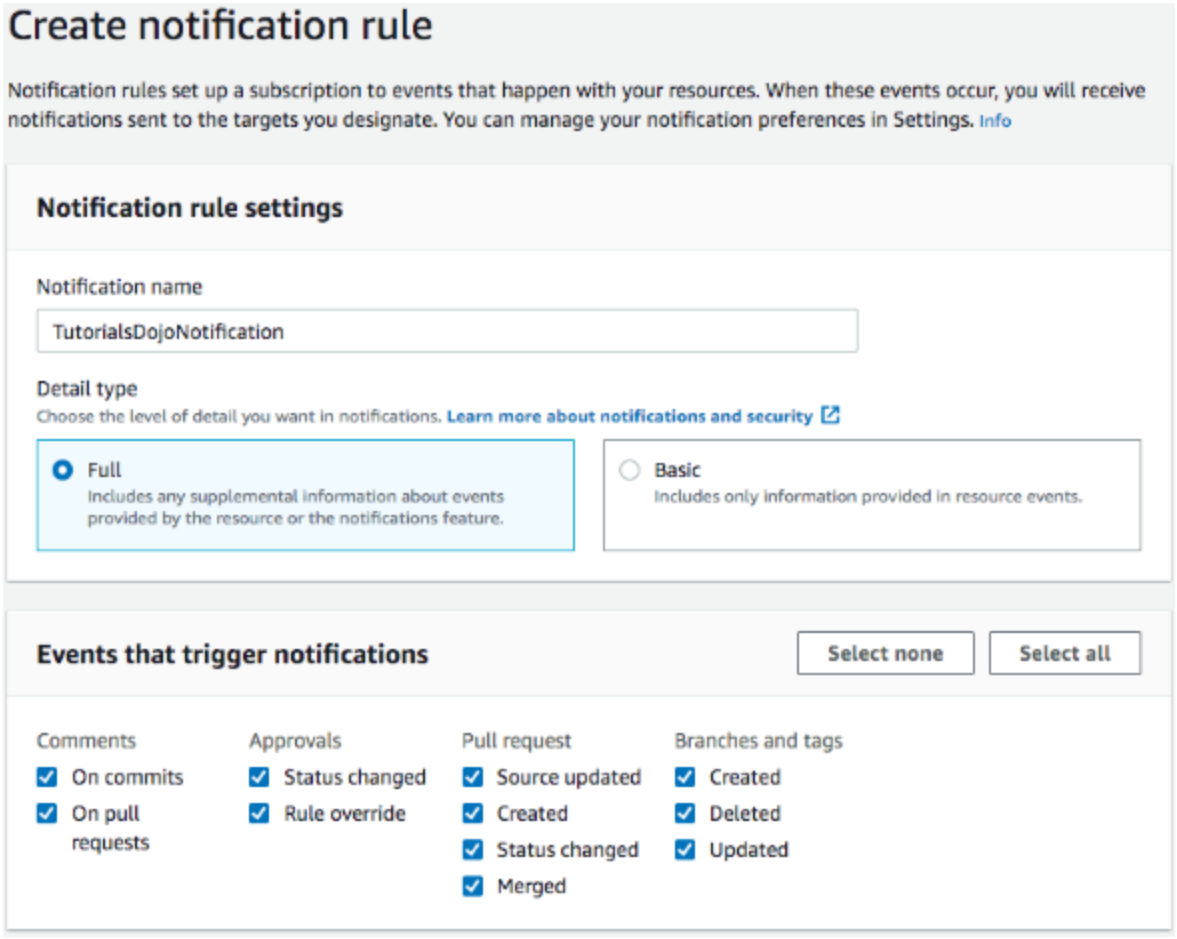
Amazon SNS > Subscriptions > Create Subscriptions

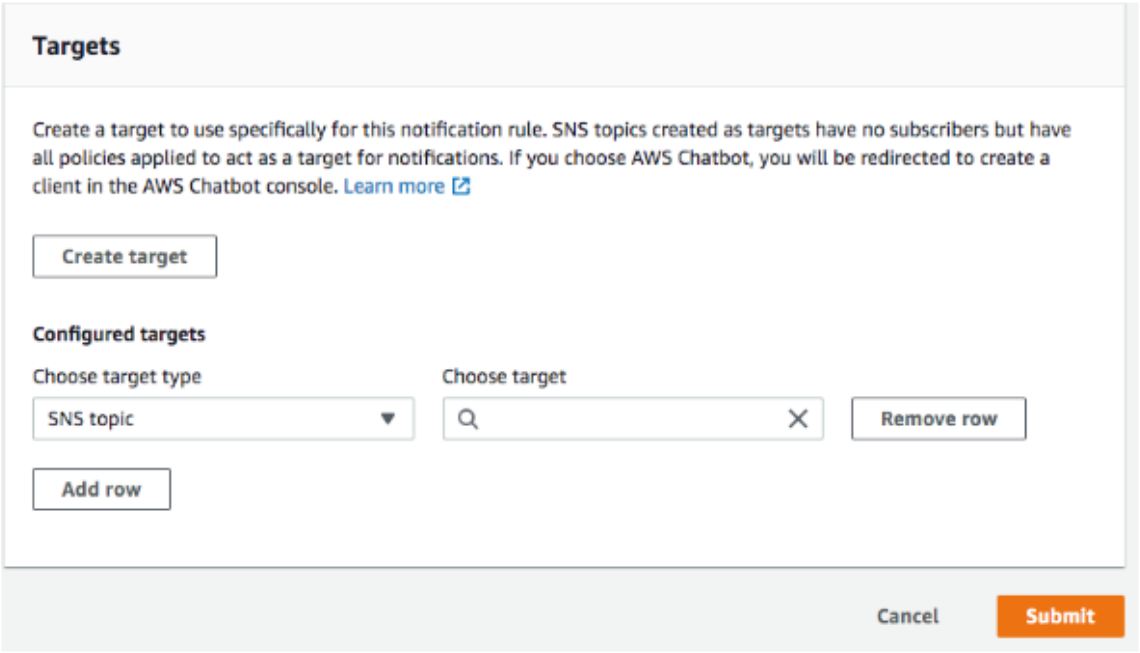


After you have created an SNS topic and subscription, head over to Codecommit. Here we have already created the TutorialsDojoRepo repository with a file named TutorialsDojo.txt. You can manually create or upload a file to our repository by heading over to the “Add File” section.



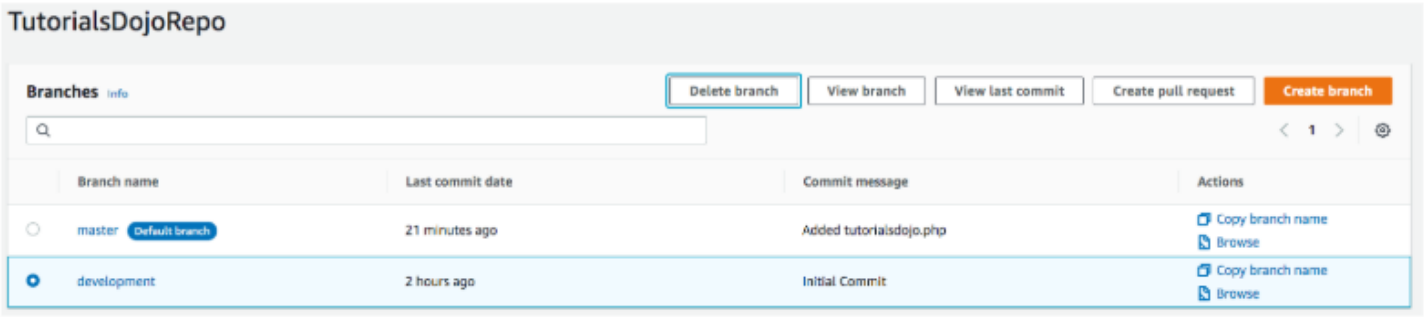
To create our Notifications, go to Settings > Notifications > Create Notifications


Create notification rule
Notification rules set up a subscription to events that happen with your resources. When these events occur, you will receive notifications sent to the targets you designate. You can manage your notification preferences in Settings. Info
Notification rule settings
Notification name
TutorialsDojoNotification
Detail type
Choose the level of detail you want in notifications. Learn more about notifications and security
Full
Includes any supplemental information about events provided by the resource or the notifications feature.
Basic
Includes only information provided in resource events.
Events that trigger notifications
Select none
Select all
Comments
On commits
On pull requests
Approvals
Status changed
Rule override
Pull request
Source updated
Created
Status changed
Merged
Branches and tags
Created
Deleted
Updated



To determine if the connection between CodeCommit and the SNS topic works, head over to the Notification Rule that you just created then navigate at the bottom. Under Notification Rule Targets, the status must be Active.

Now let us create a scenario wherein a colleague accidentally deletes a branch. To achieve our goal, we must receive an email from AWS stating that a branch has been deleted.



Sources:
https://docs.aws.amazon.com/codecommit/latest/userguide/how-to-notify.html
https://docs.aws.amazon.com/sns/latest/dg/sns-tutorial-create-topic.html
https://docs.aws.amazon.com/sns/latest/dg/sns-tutorial-create-subscribe-endpoint-to-topic.html

AWS X-Ray

- AWS X-Ray analyzes and debugs production, distributed applications, such as those built using a microservices architecture. With X-Ray, you can identify performance bottlenecks, edge case errors, and other hard to detect issues.
- Concepts
    - A **segment** provides the name of the compute resources running your application logic, details about the request sent by your application, and details about the work done.
    - A segment can break down the data about the work done into **subsegments**. A subsegment can contain additional details about a call to an AWS service, an external HTTP API, or an SQL database.
        - For services that don't send their own segments, like Amazon DynamoDB, X-Ray uses subsegments to generate *inferred segments* and *downstream nodes* on the service map. This lets you see all of your downstream dependencies, even if they don't support tracing, or are external.
        - Subsegments represent your application's view of a downstream call as a client. If the downstream service is also instrumented (like an AWS SDK client), the segment that it sends replaces the inferred segment generated from the upstream client's subsegment.
    - X-Ray uses the data that your application sends to generate a **service graph**. Each AWS resource that sends data to X-Ray appears as a service in the graph.
    - A **service graph** is a JSON document that contains information about the services and resources that make up your application. The X-Ray console uses the service graph to generate a visualization or *service map*. Service graph data is retained for 30 days.
    - **Edges** connect the services that work together to serve requests. Edges connect clients to your application, and your application to the downstream services and resources that it uses.
    - A *trace ID* tracks the path of a request through your application. A **trace** collects all the segments generated by a single request.
        - That request is typically an HTTP GET or POST request that travels through a load balancer, hits your application code, and generates downstream calls to other AWS services or external web APIs.
    - To ensure efficient tracing and provide a representative sample of the requests that your application serves, the X-Ray SDK applies a **sampling algorithm** to determine which requests get traced.
        - By default, the X-Ray SDK records the first request each second, and five percent of any additional requests.
    - For advanced tracing, you can drill down to traces for individual requests, or use **filter expressions** to find traces related to specific paths or users.
    - **Groups** are a collection of traces that are defined by a filter expression. Groups are identified by their name or an Amazon Resource Name, and contain a filter expression.
    - **Annotations** are simple key-value pairs that are indexed for use with filter expressions. Use annotations to record data that you want to use to group traces.

        - A segment can contain multiple annotations.
        - System-defined annotations include data added to the segment by AWS services, whereas user-defined annotations are metadata added to a segment by a developer.
    - **Metadata** are key-value pairs with values of any type, including objects and lists, but that are not indexed. Use metadata to record data you want to store in the trace but don't need to use for searching traces.
    - When an exception, error or fault occurs while your application is serving an instrumented request, the X-Ray SDK records details about the error, including the stack trace, if available.
- Features
    - AWS X-Ray can be used with applications running on Amazon EC2, Amazon ECS, AWS Lambda, AWS Elastic Beanstalk. You just integrate the X-Ray SDK with your application and install the X-Ray agent.
    - AWS X-Ray provides an end-to-end, cross-service, application-centric view of requests flowing through your application by aggregating the data gathered from individual services in your application into a single unit called a *trace*.
    - The X-Ray SDK captures metadata for requests made to MySQL and PostgreSQL databases (self-hosted, Amazon RDS, Amazon Aurora), and Amazon DynamoDB. It also captures metadata for requests made to Amazon SQS and Amazon SNS.
    - You can set the **trace sampling rate** that is best suited for your production applications or applications in development. X-Ray continually traces requests made to your application and stores a sampling of the requests for your analysis.
    - AWS X-Ray creates a map of services used by your application with trace data. This provides a view of connections between services in your application and aggregated data for each service, including average latency and failure rates. You can create dependency trees, perform cross-availability zone or region call detections, and more.
    - AWS X-Ray lets you add annotations to data emitted from specific components or services in your application.
- How X-Ray Works
    - AWS X-Ray receives data from services as *segments*. X-Ray then groups segments that have a common request into *traces*. X-Ray processes the traces to generate a *service graph* that provides a visual representation of your application.
- X-Ray SDK
    - The X-Ray SDK provides:
        - **Interceptors** to add to your code to trace incoming HTTP requests
        - **Client handlers** to instrument AWS SDK clients that your application uses to call other AWS services
        - An **HTTP client** to use to instrument calls to other internal and external HTTP web services
    - AWS X-Ray supports tracing for applications that are written in Node.js, Java, and .NET.
    - Instead of sending trace data directly to X-Ray, the SDK sends JSON segment documents to an X-Ray daemon process listening for UDP traffic.

    - The **X-Ray daemon** buffers segments in a queue and uploads them to X-Ray in batches.
- AWS Service Integration and Service Graph
    - You can easily integrate AWS services with AWS X-Ray. Service integration can include adding tracing headers to incoming requests, sending trace data to X-Ray, or running the X-Ray daemon.
    - X-Ray uses trace data from the AWS resources that power your cloud applications to generate a detailed service graph.
    - You can use the service graph to identify bottlenecks, latency spikes, and other issues to solve to improve the performance of your applications.
    - There are four types of X-Ray integration:
        - Active instrumentation – Samples and instruments incoming requests.
        - Passive instrumentation – Instrument requests that have been sampled by another service.
        - Request tracing – Adds a tracing header to all incoming requests and propagates it downstream.
        - Tooling – Runs the X-Ray daemon to receive segments from the X-Ray SDK.
    - The following services provide X-Ray integration:
        - AWS Lambda – Active and passive instrumentation of incoming requests on all runtimes. AWS Lambda adds two nodes to your service map, one for the AWS Lambda service, and one for the function.
        - Amazon API Gateway – Active and passive instrumentation. API Gateway uses sampling rules to determine which requests to record, and adds a node for the gateway stage to your service map.
        - Elastic Load Balancing – Request tracing on application load balancers. The application load balancer adds the trace ID to the request header before sending it to a target group.
        - AWS Elastic Beanstalk – Tooling.
- Pricing
    - You pay based on the number of traces recorded, retrieved, and scanned. A trace represents a request to your application and may include multiple data points, such as for calls to other services and database access.
    - The maximum size of a trace is 500 KB.
    - Trace data is retained for 30 days from the time it is recorded at no additional cost.

Sources:
https://aws.amazon.com/xray/features/
https://aws.amazon.com/xray/pricing/
https://docs.aws.amazon.com/xray/latest/devguide/aws-xray.html
https://aws.amazon.com/xray/faqs/

Instrumenting your Application with AWS X-Ray

Instrumenting your Node.js application

1. The AWS X-Ray SDK for Node.js provides middleware that you can use to instrument incoming HTTP requests. You need to add the SDK to your application's dependencies, usually via package.json.
2. Initialize the SDK client and add it to your application prior to declaring routes.

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.setDaemonAddress('host:port');
app.use(AWSXRay.express.openSegment('MyApp'));
```

3. Lastly, use the SDK exceptions after declaring routes.

```
app.get('/', function (req, res) {
  res.render('index');
});
app.use(AWSXRay.express.closeSegment());
```

Instrumenting your Java application

The AWS X-Ray SDK for Java provides a servlet filter that you can add to your application to start tracing incoming HTTP requests. First, if you are using Maven and Tomcat, you need to add the SDK as a dependency in your build configuration.

1. Starting in release 1.3, you can instrument your application using aspect-oriented programming (AOP) in Spring. What this means is that you can instrument your application, while it is running on AWS, without adding any code to your application's runtime.

```
*in a pom.xml file*
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>aws-xray-recorder-sdk-core</artifactId>
  <version>version</version>
</dependency>
```

2. Add a servlet filter to your deployment descriptor to trace incoming HTTP requests.

```
*in a web.xml file*
<filter>
  <filter-name>AWSXRayServletFilter</filter-name>
  <filter-class>com.amazonaws.xray.javax.servlet.AWSXRayServletFilter</filter-class>
</filter>
```

```
<filter-mapping>
  <filter-name>AWSXRayServletFilter</filter-name>
  <url-pattern>*</url-pattern>
</filter-mapping>
```

Instrumenting your C# .Net application

1. The AWS X-Ray SDK for .NET provides a message handler that you can add to your HTTP configuration to trace incoming HTTP requests. Add the AWSXRayRecorder package to your application with NuGet.
2. Add the message handler to your HTTP configuration.

```
using Amazon.XRay.Recorder.Handler.Http;
public static class AppConfig
{
    public static void Register(HttpConfiguration config)
    {
        config.MessageHandlers.Add(new TracingMessageHandler());
    }
}
```

Instrumenting your Python application

1. The X-Ray SDK for Python is a library for web applications that provides classes and methods for generating and sending trace data to the X-Ray daemon. You first download the SDK with pip.

```
pip install aws-xray-sdk
```

2. Add the Python (Django, Flask, etc) middleware to your code to instrument incoming HTTP requests.

Instrumenting your Go application

1. The X-Ray SDK for Go is a set of libraries for Go applications that provide classes and methods for generating and sending trace data to the X-Ray daemon. You first download the SDK with the go get command.

```
go get -u github.com/aws/aws-xray-sdk-go/...
```

2. Use xray.Handler to instrument incoming HTTP requests.

```
func main() {
  http.Handle("/", xray.Handler(xray.NewFixedSegmentNamer(appname),
http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
```

```
    w.Write([]byte("Hello World!"))
  }))
  http.ListenAndServe(":8000", nil)
}
```

To avoid calling the service every time your application serves a request, the SDK sends the trace data to an X-Ray daemon, which collects segments for multiple requests and uploads them in batches. Use the following scripts to automatically install a daemon to your AWS host:

1. Linux EC2

```
#!/bin/bash
curl
https://s3.dualstack.us-east-1.amazonaws.com/aws-xray-assets.us-east-
1/xray-daemon/aws-xray-daemon-2.x.rpm -o /home/ec2-user/xray.rpm
yum install -y /home/ec2-user/xray.rpm
```

2. Windows EC2

```
<powershell>
if ( Get-Service "AWSXRayDaemon" -ErrorAction SilentlyContinue ) {
   sc.exe stop AWSXRayDaemon
   sc.exe delete AWSXRayDaemon
}
$targetLocation = "C:\Program Files\Amazon\XRay"
if ((Test-Path $targetLocation) -eq 0) {
   mkdir $targetLocation
}
$zipFileName = "aws-xray-daemon-windows-service-2.x.zip"
$zipPath = "$targetLocation\$zipFileName"
$destPath = "$targetLocation\aws-xray-daemon"
if ((Test-Path $destPath) -eq 1) {
   Remove-Item -Recurse -Force $destPath
}
$daemonPath = "$destPath\xray.exe"
$daemonLogPath = "$targetLocation\xray-daemon.log"
$url =

"https://s3.dualstack.us-east-1.amazonaws.com/aws-xray-assets.us-east-
1/xray-daemon/aws-xray-daemon-windows-service-2.x.zip"
Invoke-WebRequest -Uri $url -OutFile $zipPath
Add-Type -Assembly "System.IO.Compression.Filesystem"
[io.compression.zipfile]::ExtractToDirectory($zipPath, $destPath)
New-Service -Name "AWSXRayDaemon" -StartupType Automatic
-BinaryPathName "`"$daemonPath`" -f `"$daemonLogPath`""
sc.exe start AWSXRayDaemon
```

```
</powershell>
```

3. ECS

   a. You can simply pull the Docker image that you can deploy alongside your application.

   ```
   docker pull amazon/aws-xray-daemon
   ```

   b. Or create a folder and download the daemon.

   ```
   mkdir xray-daemon && cd xray-daemon
   curl https://s3.dualstack.us-east-1.amazonaws.com/aws-xray-assets.us
   -east-1/xray-daemon/aws-xray-daemon-linux-2.x.zip -o
   ./aws-xray-daemon-linux-2.x.zip
   ~/xray-daemon$ unzip -o aws-xray-daemon-linux-2.x.zip -d .
   ```

   c. Then, create a Dockerfile with the following content.

   ```
   *~/xray-daemon/Dockerfile*
   FROM ubuntu:version
   COPY xray /usr/bin/xray-daemon
   CMD xray-daemon -f /var/log/xray-daemon.log &
   ```

   d. Finally, build the image with docker build

   ```
   docker build -t xray
   ```

4. Elastic Beanstalk

   - X-Ray daemon is already built-in in Elastic Beanstalk. Enable it through a configuration file placed under *ebextensions* or through the console settings.

   ```
   # .ebextensions/xray-daemon.config
   option_settings:
     aws:elasticbeanstalk:xray:
       XRayEnabled: true
   ```

5. Lambda

   - X-Ray automatically runs in the background when you enable tracing for your functions.

Sources:
https://us-east-1.console.aws.amazon.com/xray/home?region=us-east-1#/getting-started
https://docs.aws.amazon.com/xray/latest/devguide/xray-daemon.html

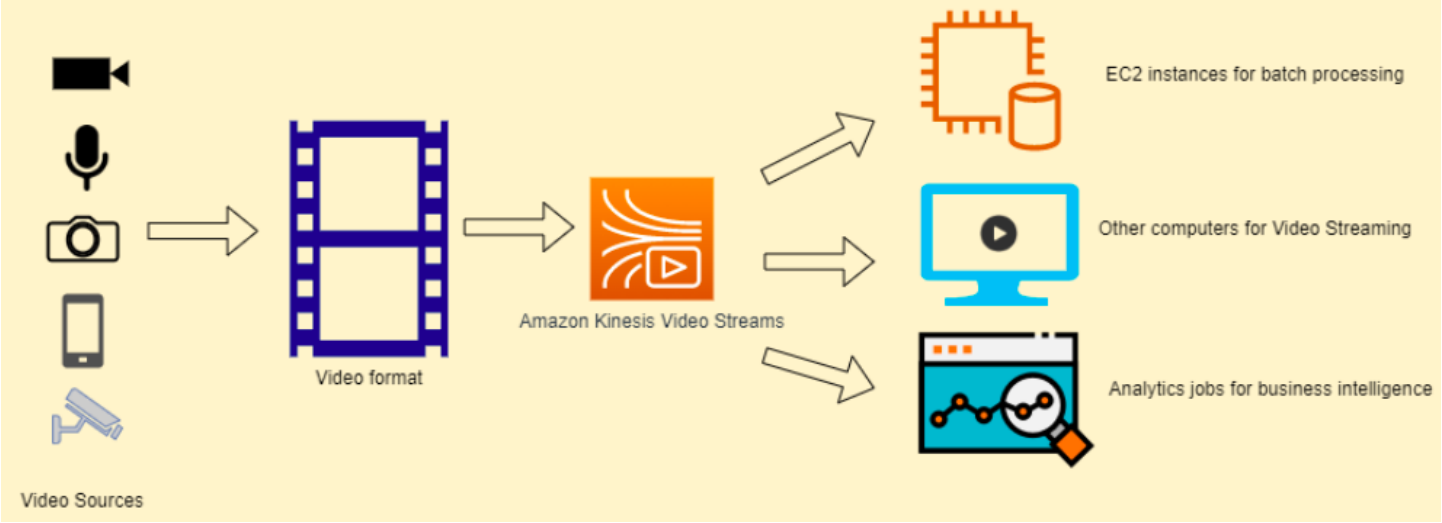
ANALYTICS

Amazon Kinesis

- For your own packages
- Makes it easy to collect, process, and analyze real-time, streaming data.
- Kinesis can ingest real-time data such as video, audio, application logs, website clickstreams, and IoT telemetry data for machine learning, analytics, and other applications.

Kinesis Video Streams

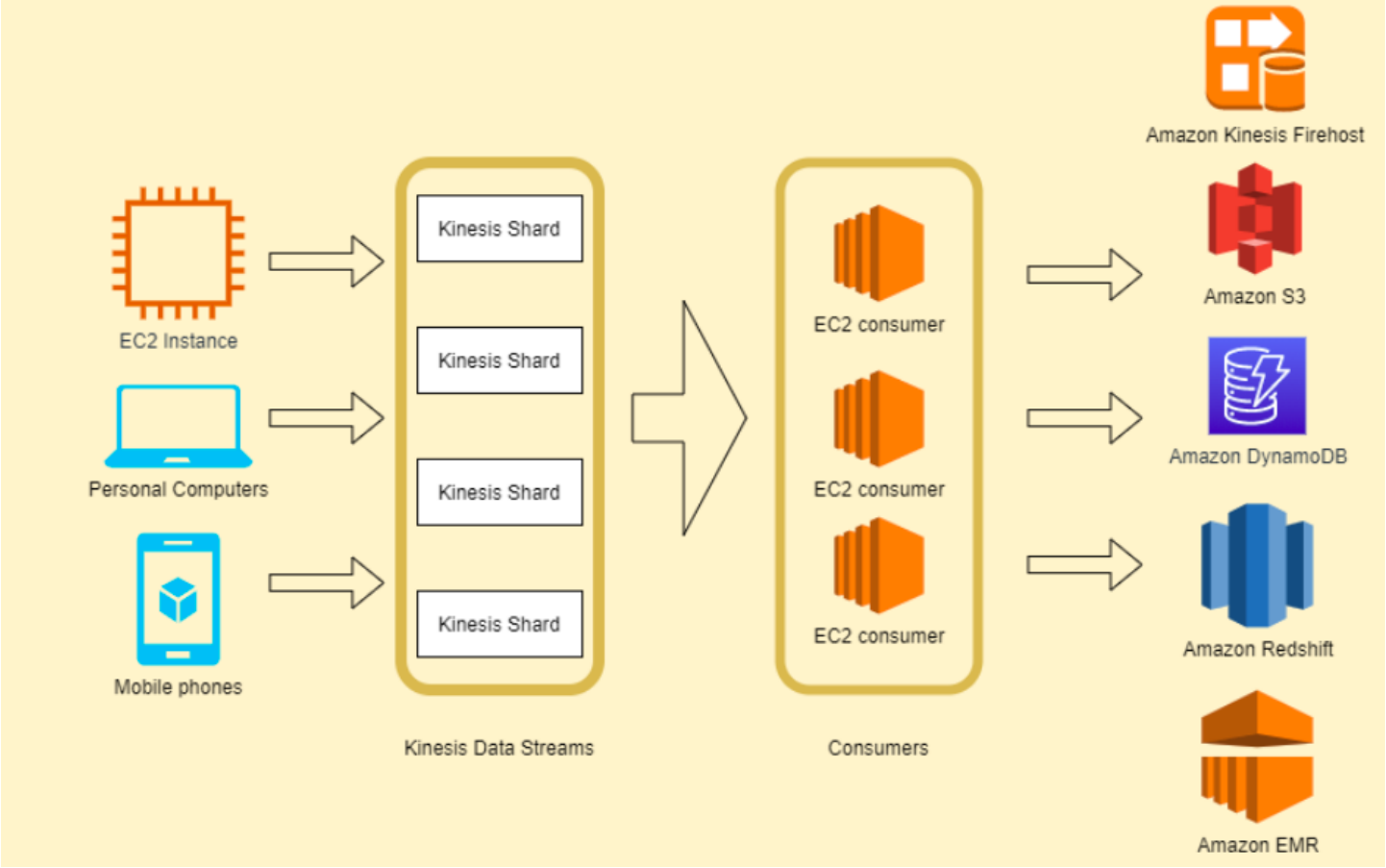
- A fully managed AWS service that you can use to stream live video from devices to the AWS Cloud, or build applications for real-time video processing or batch-oriented video analytics.
- **Benefit**
  - You can connect and stream from millions of devices.
  - You can configure your Kinesis video stream to durably store media data for custom retention periods. Kinesis Video Streams also generates an index over the stored data based on producer-generated or service-side timestamps.
  - Kinesis Video Streams is serverless, so there is no infrastructure to set up or manage.
  - You can build real-time and batch applications on data streams.
  - Kinesis Video Streams enforces Transport Layer Security (TLS)-based encryption on data streaming from devices, and encrypts all data at rest using AWS KMS.
- **Components**
  - **Producer** – Any source that puts data into a Kinesis video stream.
  - **Kinesis video stream** – A resource that enables you to transport live video data, optionally store it, and make the data available for consumption both in real time and on a batch or ad hoc basis.
    - **Time-encoded data** is any data in which the records are in a time series, and each record is related to its previous and next records.
    - A **fragment** is a self-contained sequence of frames. The frames belonging to a fragment should have no dependency on any frames from other fragments.
    - Upon receiving the data from a producer, Kinesis Video Streams stores incoming media data as **chunks**. Each chunk consists of the actual media fragment, a copy of media metadata sent by the producer, and the Kinesis Video Streams-specific metadata such as the fragment number, and server-side and producer-side timestamps.
  - **Consumer** – Gets data, such as fragments and frames, from a Kinesis video stream to view, process, or analyze it. Generally these consumers are called Kinesis Video Streams applications.



- **Kinesis Video Streams provides**
  - APIs for you to create and manage streams and read or write media data to and from a stream.
  - A console that supports live and video-on-demand playback.
  - A set of producer libraries that you can use in your application code to extract data from your media sources and upload to your Kinesis video stream.
- **Video Playbacks**
  - You can view a Kinesis video stream using either
    - **HTTP Live Streaming** (HLS) - You can use HLS for live playback.
    - ***GetMedia* API** - You use the *GetMedia* API to build your own applications to process Kinesis video streams. *GetMedia* is a real-time API with low latency.
- **Metadata**
  - Metadata is a mutable key-value pair. You can use it to describe the content of the fragment, embed associated sensor readings that need to be transferred along with the actual fragment, or meet other custom needs.
  - There are two modes in which the metadata can be embedded with fragments in a stream:
    - Nonpersistent: You can affix metadata on an ad hoc basis to fragments in a stream, based on business-specific criteria that have occurred.
    - Persistent: You can affix metadata to successive, consecutive fragments in a stream based on a continuing need.
- **Pricing**
  - You pay only for the volume of data you ingest, store, and consume through the service.

Kinesis Data Stream

- A massively scalable, highly durable data ingestion and processing service optimized for streaming data. You can configure hundreds of thousands of data producers to continuously put data into a Kinesis data stream.
- **Concepts**
    - **Data Producer** - An application that typically emits data records as they are generated to a Kinesis data stream. Data producers assign partition keys to records. Partition keys ultimately determine which shard ingests the data record for a data stream.
    - **Data Consumer** - A distributed Kinesis application or AWS service retrieving data from all shards in a stream as it is generated. Most data consumers are retrieving the most recent data in a shard, enabling real-time analytics or handling of data.
    - **Data Stream** - A logical grouping of shards. There are no bounds on the number of shards within a data stream. A data stream will retain data for **24 hours, or up to 7 days** when extended retention is enabled.
    - **Shard** - The base throughput unit of a Kinesis data stream.
        - A shard is an append-only log and a unit of streaming capability. A shard contains an ordered sequence of records ordered by arrival time.
        - Add or remove shards from your stream dynamically as your data throughput changes.
        - One shard can ingest up to 1000 data records per second, or 1MB/sec. Add more shards to increase your ingestion capability.
        - When consumers use **enhanced fan-out**, one shard provides 1MB/sec data input and 2MB/sec data output for each data consumer registered to use enhanced fan-out.
        - When consumers do **no**t use **enhanced fan-out**, a shard provides 1MB/sec of input and 2MB/sec of data output, and this output is shared with any consumer not using enhanced fan-out.
        - You will specify the number of shards needed when you create a stream and can change the quantity at any time.

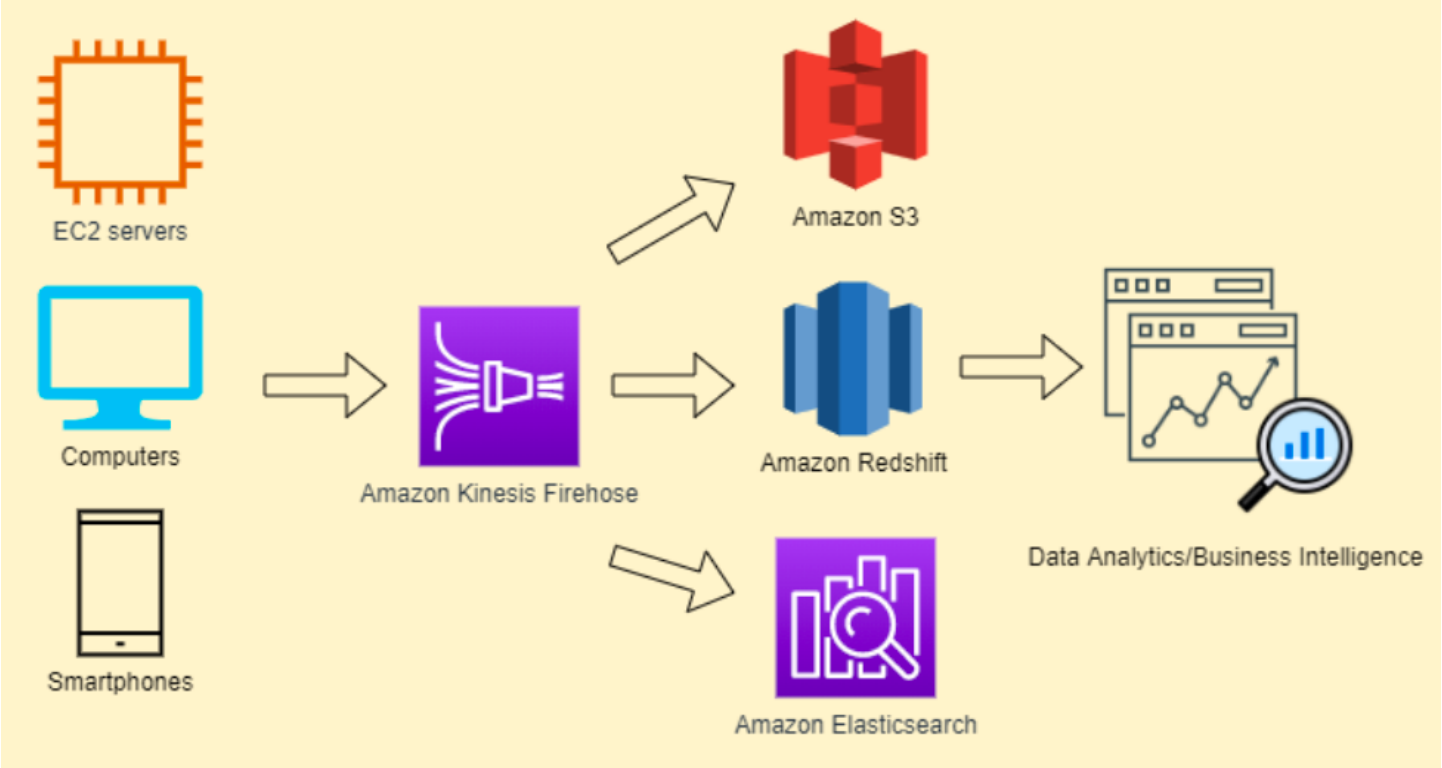


- **Data Record**
  - A record is the unit of data stored in a Kinesis stream. A record is composed of a sequence number, partition key, and data blob.
  - A data blob is the data of interest your data producer adds to a stream. The maximum size of a data blob is 1 MB.
- **Partition Key**
  - A partition key is typically a meaningful identifier, such as a user ID or timestamp. It is specified by your data producer while putting data into a Kinesis data stream, and useful for consumers as they can use the partition key to replay or build a history associated with the partition key.
  - The partition key is also used to segregate and route data records to different shards of a stream.
- **Sequence Number**
  - A sequence number is a unique identifier for each data record. Sequence number is assigned by Kinesis Data Streams when a data producer calls *PutRecord* or *PutRecords* API to add data to a Kinesis data stream.

- **Amazon Kinesis Agent** is a pre-built Java application that offers an easy way to collect and send data to your Amazon Kinesis data stream.
- **Monitoring**
  - You can monitor shard-level metrics in Kinesis Data Streams.
  - You can monitor your data streams in Amazon Kinesis Data Streams using CloudWatch, Kinesis Agent, Kinesis libraries.
  - Log API calls with CloudTrail.
- **Security**
  - Kinesis Data Streams can automatically encrypt sensitive data as a producer enters it into a stream. Kinesis Data Streams uses AWS KMS master keys for encryption.
  - Use IAM for managing access controls.
  - You can use an interface VPC endpoint to keep traffic between your Amazon VPC and Kinesis Data Streams from leaving the Amazon network.
- **Pricing**
  - You are charged for each shard at an hourly rate.
  - PUT Payload Unit is charged with a per million PUT Payload Units rate.
  - When consumers use enhanced fan-out, they incur hourly charges per consumer-shard hour and per GB of data retrieved.
  - You are charged for an additional rate on each shard hour incurred by your data stream once you enable extended data retention.

Kinesis Data Firehose

- The easiest way to load streaming data into data stores and analytics tools.
- It is a fully managed service that automatically scales to match the throughput of your data.
- It can also batch, compress, and encrypt the data before loading it.
- **Features**
    - It can capture, transform, and load streaming data into S3, Redshift, Elasticsearch Service, and Splunk, enabling near real-time analytics with existing business intelligence tools and dashboards being used today.
    - You can specify a batch size or batch interval to control how quickly data is uploaded to destinations. Additionally, you can specify if data should be compressed.
    - Once launched, your delivery streams automatically scale up and down to handle gigabytes per second or more of input data rate, and maintain data latency at levels you specify for the stream.
    - Kinesis Data Firehose can convert the format of incoming data from JSON to Parquet or ORC formats before storing the data in S3.
    - You can configure Kinesis Data Firehose to prepare your streaming data before it is loaded to data stores. Kinesis Data Firehose provides pre-built Lambda blueprints for converting common data sources such as Apache logs and system logs to JSON and CSV formats. You can use these pre-built blueprints without any change, or customize them further, or write your own custom functions.
- **Concepts**
    - **Kinesis Data Firehose Delivery Stream** - The underlying entity of Kinesis Data Firehose. You use Kinesis Data Firehose by creating a Kinesis Data Firehose delivery stream and then sending data to it.
    - **Record** - The data of interest that your data producer sends to a Kinesis Data Firehose delivery stream. A record can be as large as 1,000 KB.
    - **Data Producer** - Producers send records to Kinesis Data Firehose delivery streams.
    - **Buffer Size and Buffer Interval** - Kinesis Data Firehose buffers incoming streaming data to a certain size or for a certain period of time before delivering it to destinations. Buffer Size is in MBs and Buffer Interval is in seconds.



- **Stream Sources**
  - You can send data to your Kinesis Data Firehose Delivery stream using different types of sources:
    - a Kinesis data stream,
    - the Kinesis Agent,
    - or the Kinesis Data Firehose API using the AWS SDK.
  - You can also use CloudWatch Logs, CloudWatch Events, or AWS IoT as your data source.
  - Some AWS services can only send messages and events to a Kinesis Data Firehose delivery stream that is in the same Region.
- **Data Delivery and Transformation**
  - Kinesis Data Firehose can invoke your Lambda function to transform incoming source data and deliver the transformed data to destinations.
  - Kinesis Data Firehose buffers incoming data up to 3 MB by default.
  - If your Lambda function invocation fails because of a network timeout or because you've reached the Lambda invocation limit, Kinesis Data Firehose retries the invocation three times by default.
  - Kinesis Data Firehose can convert the format of your input data from JSON to Apache Parquet or Apache ORC before storing the data in S3. Parquet and ORC are columnar data formats that save space and enable faster queries compared to row-oriented formats like JSON.
  - Data delivery format:

        - **For data delivery to S3**, Kinesis Data Firehose concatenates multiple incoming records based on buffering configuration of your delivery stream. It then delivers the records to S3 as an S3 object.
        - **For data delivery to Redshift**, Kinesis Data Firehose first delivers incoming data to your S3 bucket in the format described earlier. Kinesis Data Firehose then issues an Redshift *COPY* command to load the data from your S3 bucket to your Redshift cluster.
        - **For data delivery to ElasticSearch**, Kinesis Data Firehose buffers incoming records based on buffering configuration of your delivery stream. It then generates an Elasticsearch bulk request to index multiple records to your Elasticsearch cluster.
        - **For data delivery to Splunk**, Kinesis Data Firehose concatenates the bytes that you send.
    - Data delivery frequency
        - The frequency of data delivery to S3 is determined by the **S3 Buffer size** and **Buffer interval** value that you configured for your delivery stream.
        - The frequency of data *COPY* operations from S3 to Redshift is determined by how fast your Redshift cluster can finish the *COPY* command.
        - The frequency of data delivery to ElasticSearch is determined by the **Elasticsearch Buffer size** and **Buffer interval** values that you configured for your delivery stream.
        - Kinesis Data Firehose buffers incoming data before delivering it to Splunk. The buffer size is 5 MB, and the buffer interval is 60 seconds.
- **Monitoring**
    - Kinesis Data Firehose exposes several metrics through the console, as well as CloudWatch for monitoring.
    - Kinesis Agent publishes custom CloudWatch metrics, and helps assess whether the agent is healthy, submitting data into Kinesis Data Firehose as specified, and consuming the appropriate amount of CPU and memory resources on the data producer.
    - Log API calls with CloudTrail.
- **Security**
    - Kinesis Data Firehose provides you the option to have your data automatically encrypted after it is uploaded to the destination.
    - Manage resource access with IAM.
- **Pricing**
    - You pay only for the volume of data you transmit through the service. You are billed for the volume of data ingested into Kinesis Data Firehose, and if applicable, for data format conversion to Apache Parquet or ORC.

Kinesis Data Analytics

- Analyze streaming data, gain actionable insights, and respond to your business and customer needs in real time. You can quickly build SQL queries and Java applications using built-in templates and operators for common processing functions to organize, transform, aggregate, and analyze data at any scale.
- **General Features**
  - Kinesis Data Analytics is **serverless** and takes care of everything required to continuously run your application.
  - Kinesis Data Analytics elastically scales applications to keep up with any volume of data in the incoming data stream.
  - Kinesis Data Analytics delivers sub-second processing latencies so you can generate real-time alerts, dashboards, and actionable insights.
- **SQL Features**
  - Kinesis Data Analytics supports standard ANSI SQL.
  - Kinesis Data Analytics integrates with Kinesis Data Streams and Kinesis Data Firehose so that you can readily ingest streaming data.
  - SQL applications in Kinesis Data Analytics support two types of inputs:
    - A **streaming data source** is continuously generated data that is read into your application for processing.
    - A **reference data source** is static data that your application uses to enrich data coming in from streaming sources.
  - Kinesis Data Analytics provides an easy-to-use schema editor to discover and edit the structure of the input data. The wizard automatically recognizes standard data formats such as JSON and CSV.
  - Kinesis Data Analytics offers functions optimized for stream processing so that you can easily perform advanced analytics such as anomaly detection and top-K analysis on your streaming data.
- **Java Features**
  - Kinesis Data Analytics includes open source libraries based on Apache Flink. Apache Flink is an open source framework and engine for building highly available and accurate streaming applications.
  - You can use the Kinesis Data Analytics Java libraries to integrate with multiple AWS services.
  - You can create and delete durable application backups through a simple API call. You can immediately restore your applications from the latest backup after a disruption, or you can restore your application to an earlier version.
  - Java applications in Kinesis Data Analytics enable you to build applications whose processed records affect the results exactly once, referred to as **exactly once processing**.
  - The service stores previous and in-progress computations, or state, in running application storage. State is always encrypted and incrementally saved in running application storage.

- An **application** is the primary resource in Kinesis Data Analytics. Kinesis data analytics applications continuously read and process streaming data in real time.
    - You write application code using SQL to process the incoming streaming data and produce output. Then, Kinesis Data Analytics writes the output to a configured destination.
    - You can also process and analyze streaming data using Java.
- **Components**
    - Input is the streaming source for your application. In the input configuration, you map the streaming source to an in-application data stream(s).
    - **Application code** is a series of SQL statements that process input and produce output.
    - You can create one or more in-application streams to store the **output**. You can then optionally configure an application output to persist data from specific in-application streams to an external destination.
- An **in-application data stream** is an entity that continuously stores data in your application for you to perform processing.
- Kinesis Data Analytics provisions capacity in the form of **Kinesis Processing Units (KPU)**. A single KPU provides you with the memory (4 GB) and corresponding computing and networking.
- For Java applications using Apache Flink, you build your application locally, and then make your application code available to the service by uploading it to an S3 bucket.
- Kinesis Data Analytics for Java applications provides your application 50 GB of running application storage per Kinesis Processing Unit. Kinesis Data Analytics scales storage with your application.
- Running application storage is used for **saving application state using checkpoints**. It is also accessible to your application code to use as **temporary disk for caching data** or any other purpose.
- **Pricing**
    - You are charged an hourly rate based on the average number of Kinesis Processing Units (or KPUs) used to run your stream processing application.
    - For Java applications, you are charged a single additional KPU per application for application orchestration. Java applications are also charged for running application storage and durable application backups.

Sources:
https://aws.amazon.com/kinesis/
https://aws.amazon.com/kinesis/video-streams/faqs/
https://aws.amazon.com/kinesis/data-streams/faqs/
https://aws.amazon.com/kinesis/data-firehose/faqs/
https://aws.amazon.com/kinesis/data-analytics/faqs/

Kinesis Scaling, Resharding and Parallel Processing

- **Kinesis Resharding** enables you to increase or decrease the number of shards in a stream in order to adapt to changes in the rate of data flowing through the stream.
- A shard is the base throughput unit of an Amazon Kinesis data stream. You pay for each shard at an hourly rate. If you want to improve performance of the stream, you can add more shards and if the data stream is underutilized, you can remove the excess shards. This process of adding and removing shards in the data stream is called "resharding".
- There are two types of shards in Amazon Kinesis: "hot" and "cold". A hot shard is the one that receives more data while the cold shard gets less data than expected.
- There are two types of operations in Kinesis Resharding: "split" and "merge". For hot shards, you have to split them to increase capacity for the hash keys that target those shards. Conversely, the cold shards must be merged in order to save costs and since it is underutilized.
- Resharding is always pairwise. You cannot split into more than two shards in a single operation, and you cannot merge more than two shards in a single operation.
- The Kinesis Client Library (KCL) tracks the shards in the stream using an Amazon DynamoDB table, and adapts to changes in the number of shards that result from resharding. When new shards are created as a result of resharding, the KCL discovers the new shards and populates new rows in the table.
- The workers automatically discover the new shards and create processors to handle the data from them. The KCL also distributes the shards in the stream across all the available workers and record processors.

	SPLIT SHARDS	MERGE SHARDS
Description	Split the hot shards to increase capacity	Merge cold shards to better utilize the unused capacity.
Positive Effect	Double the stream's capacity	Reduce cost
Negative Effect	Might create unnecessary cost	Reduce capacity of your stream.
Suitable Shard Type To Use	Hot Shards	Cold Shards

- When you use the KCL, you should ensure that **the number of instances does not exceed the number of shards** (except for failure standby purposes).

- **Each shard is processed by exactly one KCL worker and has exactly one corresponding record processor**.
- **One worker can process any number of shards**.

Examples:

If the number of shards in the Kinesis data stream is 6, the maximum number of EC2 instances that can be used to process data from all the shards is 6. The minimum number of instances is 1 but take note that this will result in poor performance.

If the number of shards in the Kinesis data stream is 10, you cannot use 11 EC2 instances to process the data from all the 10 shards. Remember that each shard is processed by exactly 1 KCL worker.

- You can scale your application to use more than one EC2 instance when processing a stream. By doing so, you allow the record processors in each instance to work in parallel. When the KCL worker starts up on the scaled instance, it load-balances with the existing instances, so now each instance handles the same amount of shards.
- To scale up processing in your application:
  - Increase the instance size (because all record processors run in parallel within a process)
  - Increase the number of instances up to the maximum number of open shards (because shards can be processed independently)
  - Increase the number of shards (which increases the level of parallelism)

Source:
https://docs.aws.amazon.com/streams/latest/dev/kinesis-record-processor-scaling.html

Amazon Athena

- An interactive query service that makes it easy to analyze data **directly** in S3 using standard SQL.

Features

- Athena is **serverless**.
- Has a built-in query editor.
- Uses *Presto*, an open source, distributed SQL query engine optimized for low latency, ad hoc analysis of data.
- Athena supports a wide variety of data formats such as CSV, JSON, ORC, Avro, or Parquet.
- Athena automatically executes queries in **parallel**, so that you get query results in seconds, even on large datasets.
- Athena uses Amazon S3 as its underlying data store, making your data highly available and durable.
- Athena integrates with Amazon QuickSight for easy data visualization.
- Athena integrates out-of-the-box with AWS Glue.

Athena uses a managed **Data Catalog** to store information and schemas about the databases and tables that you create for your data stored in S3.

Partitioning

- By partitioning your data, you can restrict the amount of data scanned by each query, thus improving performance and reducing cost.
- Athena leverages *Hive* for partitioning data.
- You can partition your data by any key.

Queries

- You can query geospatial data.
- You can query different kinds of logs as your datasets.
- Athena stores query results in S3.
- Athena retains query history for 45 days.
- Athena does not support user-defined functions, *INSERT INTO* statements, and stored procedures.
- Athena supports both simple data types such as INTEGER, DOUBLE, VARCHAR and complex data types such as MAPS, ARRAY and STRUCT.
- Athena supports querying data in Amazon S3 Requester Pays buckets.

Security

- Control access to your data by using IAM policies, access control lists, and S3 bucket policies.
- If the files in the target S3 bucket is encrypted, you can perform queries on the encrypted data itself.

Pricing

- You pay only for the queries that you run. You are charged based on the amount of data scanned by each query.
- You are not charged for failed queries.
- You can get significant cost savings and performance gains by compressing, partitioning, or converting your data to a columnar format, because each of those operations reduces the amount of data that Athena needs to scan to execute a query.

Sources:
https://docs.aws.amazon.com/athena/latest/ug/
https://aws.amazon.com/athena/features
https://aws.amazon.com/athena/pricing
https://aws.amazon.com/athena/faqs

APPLICATION

Amazon SQS

- A hosted queue that lets you integrate and decouple distributed software systems and components.
- SQS supports both **standard** and **FIFO queues**.
- SQS uses pull based (polling) not push based
- Users can access Amazon SQS from their VPC using VPC endpoints, without using public IPs, and without needing to traverse the public internet. VPC endpoints for Amazon SQS are powered by AWS PrivateLink.

Benefits

  - You control who can send messages to and receive messages from an SQS queue.
  - Supports server-side encryption.
  - SQS stores messages on multiple servers for durability.
  - SQS uses redundant infrastructure to provide highly-concurrent access to messages and high availability for producing and consuming messages.
  - SQS can scale to process each buffered request and handle any load increases or spikes independently.
  - SQS locks your messages during processing, so that multiple producers can send and multiple consumers can receive messages at the same time.

Types of Queues

Standard Queue	FIFO Queue
Available in all regions **Unlimited Throughput** - Standard queues support a nearly unlimited number of transactions per second (TPS) per action. **At-Least-Once Delivery** - A message is delivered at least once, but occasionally more than one copy of a message is delivered. **Best-Effort Ordering** - Occasionally, messages might be delivered in an order different from which they were sent.	Available in the US East (N. Virginia), US East (Ohio) US West (Oregon), EU (Ireland), Asia Pacific (Sydney), and Asia Pacific (Tokyo)regions. **High Throughput** - By default, FIFO queues support up to 3,000 messages per second with batching. (Can request a limit increase). FIFO queues support up to 300 messages per second (300 send, receive, or delete operations per second) without batching. **Exactly-Once Processing** - A message is delivered once and remains available until a consumer processes and deletes it. Duplicates aren't introduced into the queue. **First-in-First-Out Delivery** - The order in which messages are sent and received is strictly preserved.
Send data between applications when the throughput is important. Tutorials Dojo	Send data between applications when the order of events is important.

- You can include structured metadata (such as timestamps, geospatial data, signatures, and identifiers) with messages using **message attributes**.
- **Message timers** let you specify an initial invisibility period for a message added to a queue. The default (minimum) invisibility period for a message is 0 seconds. The maximum is 15 minutes.
- SQS doesn't automatically delete a message after receiving it for you, in case you don't successfully receive the message.
- You can subscribe one or more SQS queues to an Amazon SNS topic from a list of topics available for the selected queue.
- You can configure an existing SQS queue to trigger an AWS Lambda function when new messages arrive in a queue.
    - Your queue and Lambda function must be in the same AWS Region.
    - FIFO queues also support Lambda function triggers.

    - You can associate only one queue with one or more Lambda functions.
    - You can't associate an encrypted queue that uses an AWS managed Customer Master Key for SQS with a Lambda function in a different AWS account.
- You can delete all the messages in your queue by **purging** them.
- **Long polling** helps reduce the cost by eliminating the number of empty responses and false empty responses. While the regular **short polling** returns immediately, even if the message queue being polled is empty, long polling doesn't return a response until a message arrives in the message queue, or the long poll times out.
    - Short polling occurs when the *WaitTimeSeconds* parameter of a *ReceiveMessage* request is set to 0.
- To prevent other consumers from processing a message redundantly, SQS sets a **visibility timeout**, a period of time SQS prevents other consumers from receiving and processing the message. The default visibility timeout for a message is 30 seconds. The minimum is 0 seconds. The maximum is 12 hours.
- SQS supports **dead-letter queues**, which other queues can target for messages that can't be processed successfully.
- **Delay queues** let you postpone the delivery of new messages to a queue for a number of seconds.

Basic SQS Architecture

    - Main Parts
        - The components of your distributed system
        - The queue
        - The messages
    - Standard Queues
        - Default queue type.
        - Makes a best effort to preserve the order of messages.
        - Stores copies of your messages on multiple servers for redundancy and high availability.
        - Consumes messages using **short polling** (default) - take a subset of SQS servers (based on a weighted random distribution) and returns messages from only those servers.
    - FIFO Queues
        - The order in which messages are sent and received is strictly preserved and a message is delivered once and remains available until a consumer processes and deletes it.
        - Duplicates aren't introduced into the queue.
        - FIFO queues support **message groups** that allow multiple ordered message groups within a single queue.
    - When you create a new queue, you must specify a *queue name* unique for your AWS account and region. This becomes your queue url. https://sqs.*region*.amazonaws.com/*accountnumber*/*queuename*
    - Each message receives a system-assigned *message ID* for identifying messages.
    - Every time you receive a message from a queue, you receive a *receipt handle* for that message.
- You can use **cost allocation tags** to organize your AWS bill to reflect your own cost structure.

- Send, receive, or delete messages in **batches** of up to 10 messages or 256KB.

Dead-Letter Queues

- A dead-letter queue lets you set aside and isolate messages that can't be processed correctly to determine why their processing didn't succeed.
- Setting up a dead-letter queue allows you to do the following:
  - Configure an alarm for any messages delivered to a dead-letter queue.
  - Examine logs for exceptions that might have caused messages to be delivered to a dead-letter queue.
  - Analyze the contents of messages delivered to a dead-letter queue to diagnose software or the producer's or consumer's hardware issues.
  - Determine whether you have given your consumer sufficient time to process messages.
- When to use a dead-letter queue
  - When you have a standard SQS queue, to avoid additional costs from SQS handling failed messages over and over again. Dead-letter queues can help you troubleshoot incorrect message transmission operations.
  - To decrease the number of messages and to reduce the possibility of exposing your system to poison-pill messages (messages that can be received but can't be processed).
- When not to use a dead-letter queue
  - When you want to be able to keep retrying the transmission of a message indefinitely in your SQS standard queue.
  - When you don't want to break the exact order of messages or operations in your SQS FIFO queue.

Best Practices

- Extend the message's visibility timeout to the maximum time it takes to process and delete the message. If you don't know how long it takes to process a message, as long as your consumer still works on the message, keep extending the visibility timeout .
- Using the appropriate polling mode.
- Configure a dead-letter queue to capture problematic messages.
- To avoid inconsistent message processing by standard queues, avoid setting the number of maximum receives to 1 when you configure a dead-letter queue.
- Don't create reply queues per message. Instead, create reply queues on startup, per producer, and use a correlation ID message attribute to map replies to requests. Don't let your producers share reply queues.
- Reduce cost by batching message actions.
- Use message deduplication IDs to monitor duplicate sent messages.

CDA Exam Notes:

Your producer (application) should provide message deduplication ID values for each message sent in the following scenarios:

- Messages sent with identical message bodies that Amazon SQS must treat as unique.
- Messages sent with identical content but different message attributes that Amazon SQS must treat as unique.
- Messages sent with different content that Amazon SQS must treat as duplicates, like having retry counts in the message body

For other message duplication troubleshooting scenarios, you may also increase your visibility timeout. Be aware of its impact to your system (might take a while before your message can get processed again in case it previously failed) and the maximum limit for your timeout setting.

- **Monitoring, Logging, and Automating**
  - Monitor SQS queues using CloudWatch
  - Log SQS API Calls Using AWS CloudTrail
  - Automate notifications from AWS Services to SQS using CloudWatch Events
- **Security**
  - Use IAM for user authentication.
  - SQS has its own resource-based permissions system that uses policies written in the same language used for IAM policies.
  - Protect data using Server-Side Encryption and AWS KMS.
  - SSE encrypts messages as soon as Amazon SQS receives them. The messages are stored in encrypted form and Amazon SQS decrypts messages only when they are sent to an authorized consumer.
- **Pricing**
  - You are charged per 1 million SQS requests. Price depends on the type of queue being used. Requests include:
    - API Actions
    - FIFO Requests
    - A single request of 1 to 10 messages, up to a maximum total payload of 256 KB
    - Each 64 KB chunk of a payload is billed as 1 request
    - Interaction with Amazon S3
    - Interaction with AWS KMS
  - Data transfer out of SQS per TB/month after consuming 1 GB for that month

Sources:
https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide
https://aws.amazon.com/sqs/features/

https://aws.amazon.com/sqs/pricing/
https://aws.amazon.com/sqs/faqs/

Amazon SNS

- A web service that makes it easy to set up, operate, and send notifications from the cloud. SNS follows the **"publish-subscribe" (pub-sub) messaging** paradigm, with notifications being delivered to clients using a **"push"** mechanism rather than to periodically check or "poll" for new information and updates.

Features

- SNS is an **event-driven** computing hub that has native integration with a wide variety of AWS event sources (including EC2, S3, and RDS) and AWS event destinations (including SQS, and Lambda).
    - **Event-driven computing** is a model in which subscriber services automatically perform work in response to events triggered by publisher services. It can automate workflows while decoupling the services that collectively and independently work to fulfil these workflows.
- **Message filtering** allows a subscriber to create a filter policy, so that it only gets the notifications it is interested in.
- **Message fanout** occurs when a message is sent to a topic and then replicated and pushed to multiple endpoints. Fanout provides asynchronous event notifications, which in turn allows for parallel processing.
- **SNS mobile notifications** allows you to fanout mobile push notifications to iOS, Android, Fire OS, Windows and Baidu-based devices. You can also use SNS to fanout text messages (SMS) to 200+ countries and fanout email messages (SMTP).
- **Application and system alerts** are notifications, triggered by predefined thresholds, sent to specified users by SMS and/or email.
- **Push email** and **text messaging** are two ways to transmit messages to individuals or groups via email and/or SMS.
- SNS provides durable storage of all messages that it receives. When SNS receives your *Publish* request, it stores multiple copies of your message to disk. Before SNS confirms to you that it received your request, it stores the message in multiple Availability Zones within your chosen AWS Region.
- SNS allows you to set a TTL (Time to Live) value for each message. When the TTL expires for a given message that was not delivered and read by an end user, the message is deleted.

SNS provides simple APIs and easy integration with applications.

Publishers and Subscribers

- Publishers communicate asynchronously with subscribers by producing and sending a message to a topic, which is a logical access point and communication channel.
- Subscribers consume or receive the message or notification over one of the supported protocols when they are subscribed to the topic.
- Publishers create topics to send messages, while subscribers subscribe to topics to receive messages.

- SNS FIFO topics support the forwarding of messages to SQS FIFO queues. You can also use SNS to forward messages to standard queues.

SNS Topics

- Instead of including a specific destination address in each message, a publisher sends a message to a **topic**. SNS matches the topic to a list of subscribers who have subscribed to that topic, and delivers the message to each of those subscribers.
- Each topic has a unique name that identifies the SNS endpoint for publishers to post messages and subscribers to register for notifications.
- A topic can support subscriptions and notification deliveries over multiple transports.

The SNS service will attempt to deliver messages from the publisher in the order they were published into the topic, so no guarantee.

SNS also logs the the delivery status of notification messages sent to topics with the following SNS endpoints:

- Application
- HTTP
- Lambda
- SQS
- Amazon Kinesis Data Firehose

Message Attributes

- Amazon SNS supports delivery of *message attributes*. Message attributes allow you to provide structured metadata items (such as time stamps, geospatial data, signatures, and identifiers) about the message. Message attributes are optional and separate from, but sent along with, the message body.
- You can use message attributes to help structure the push notification message for mobile endpoints. The attributes are not delivered to the mobile endpoint, as they are when sending messages with message attributes to SQS endpoints.
- You can also use message attributes to make your messages filterable with *subscription filter policies*. You apply filter policies to topic subscriptions.
- Message attributes contain a name, type, and value that must not be empty or null. The message body should not be empty or null also.

Message Filtering

- A filter policy is a simple JSON object.
- By default, a subscriber of an SNS topic receives every message published to the topic. The filter policy contains attributes that define which messages the subscriber receives.

Raw Message Delivery

- By default, messages are delivered encoded in JSON that provides metadata about the message and topic.
- You can send large payload messages using AWS SDK that supports AWS Signature Version 4 signing.
- You can also enable raw message delivery for messages delivered to either SQS endpoints or HTTP/S endpoints.

System to System Messaging

- When a message is published to an SNS topic that has a **Lambda function** subscribed to it, the Lambda function is invoked with the payload of the published message. The Lambda function receives the message payload as an input parameter and can manipulate the information in the message, publish the message to other SNS topics, or send the message to other AWS services.
- When you subscribe a **SQS queue** to a SNS topic, you can publish a message to the topic and SNS sends a SQS message to the subscribed queue. The SQS message contains the subject and message that were published to the topic along with metadata about the message in a JSON document.
- When you subscribe an **HTTP/s endpoint** to a topic, you can publish a notification to the topic and SNS sends an HTTP POST request delivering the contents of the notification to the subscribed endpoint. When you subscribe the endpoint, you select whether SNS uses HTTP or HTTPS to send the POST request to the endpoint.

User Notifications

- You have the ability to send **push notification messages** directly to apps on mobile devices. Push notification messages sent to a mobile endpoint can appear in the mobile app as message alerts, badge updates, or even sound alerts.
- *Direct addressing* allows you to deliver notifications directly to a single endpoint, rather than sending identical messages to all subscribers of a topic. This is useful if you want to deliver precisely targeted messages to each recipient.
- You can use SNS to send **text messages**, or SMS messages, to SMS-enabled devices. You can send a message directly to a phone number, or you can send a message to multiple phone numbers at once by subscribing those phone numbers to a topic and sending your message to the topic.
- You can use the **Delivery Status** feature to get information on the final disposition of your SMS message.
- SMS messages that are of high priority to your business should be marked as *Transactional*. This ensures that messages such as those that contain one-time passwords or PINs get delivered over routes with the highest delivery reliability.
- SMS messages that carry marketing messaging should be marked *Promotional*. Amazon SNS ensures that such messages are sent over routes that have a reasonable delivery reliability but are substantially cheaper than the most reliable routes.

SNS Delivery Retries

- All messages sent to SNS are processed and delivered immediately. If a message cannot be successfully delivered on the first attempt, SNS implements a 4-phase retry policy:
  1) retries with no delay in between attempts
  2) retries with some minimum delay between attempts
  3) retries with some back-off model (linear or exponential)
  4) retries with some maximum delay between attempts

Monitoring

- Monitoring SNS topics using CloudWatch
- Logging SNS API calls using CloudTrail

Security

- SNS provides encrypted topics to protect your messages from unauthorized and anonymous access. The encryption takes place on the server side.
- SNS supports VPC Endpoints via AWS PrivateLink. You can use VPC Endpoints to privately publish messages to SNS topics, from a VPC, without traversing the public internet.
- Using access control policies, you have detailed control over which endpoints a topic allows, who is able to publish to a topic, and under what conditions.
- You can enable AWS X-Ray for your messages passing through Amazon SNS, making it easier to trace and analyze messages as they travel through to the downstream services.

Pricing

- You pay based on the number of notifications you publish, the number of notifications you deliver, and any additional API calls for managing topics and subscriptions. Delivery pricing varies by endpoint type.

Sources:
https://docs.aws.amazon.com/sns/latest/dg
https://aws.amazon.com/sns/features/
https://aws.amazon.com/sns/pricing/
https://aws.amazon.com/sns/faqs/

Amazon SWF

- A fully-managed state tracker and task coordinator in the Cloud. You create desired workflows with their associated tasks and any conditional logic you wish to apply and store them with SWF.

Features

- SWF promotes a separation between the control flow of your background job's stepwise logic and the actual units of work that contain your unique business logic.
- SWF manages your workflow execution history and other details of your workflows across 3 availability zones.
- SWF lets you write your application components and coordination logic in any programming language and run them in the cloud or on-premises.
- SWF is highly scalable. It gives you full control over the number of workers that you run for each activity type and the number of instances that you run for a decider.
- SWF also provides the **AWS Flow Framework** to help developers use asynchronous programming in the development of their applications.

Concepts

Workflow

- A set of activities that carry out some objective, together with logic that coordinates the activities.
- Workflows coordinate and manage the execution of activities that can be run asynchronously across multiple computing devices and that can feature both sequential and parallel processing.
- Each workflow runs in an AWS resource called a *domain*, which controls the workflow's scope.
- An AWS account can have multiple domains, each of which can contain multiple workflows, but workflows in different domains can't interact.
- When you register an activity to a workflow, you provide information such as a name and version, and some timeout values based on how long you expect the activity to take.

Activity Task

- An **activity task** tells an activity worker to perform its function.
- SWF stores tasks and assigns them to workers when they are ready, tracks their progress, and maintains their state, including details on their completion.
- To coordinate tasks, you write a program that gets the latest state of each task from SWF and uses it to initiate subsequent tasks.
- Activity tasks can run synchronously or asynchronously. They can be distributed across multiple computers, potentially in different geographic regions, or they can all run on the same computer.
- Activity tasks for a running workflow execution appear on the *activity task list*, which is provided when you schedule an activity in the workflow.
- If you don't specify a task list when scheduling an activity task, the task is automatically placed on the default task list.

Lambda task
- Executes a Lambda function instead of a traditional SWF activity.

Decision task
- A Decision task tells a decider that the state of the workflow execution has changed so that the decider can determine the next activity that needs to be performed. The decision task contains the current workflow history.
- SWF assigns each decision task to exactly one decider and allows only one decision task at a time to be active in a workflow execution.

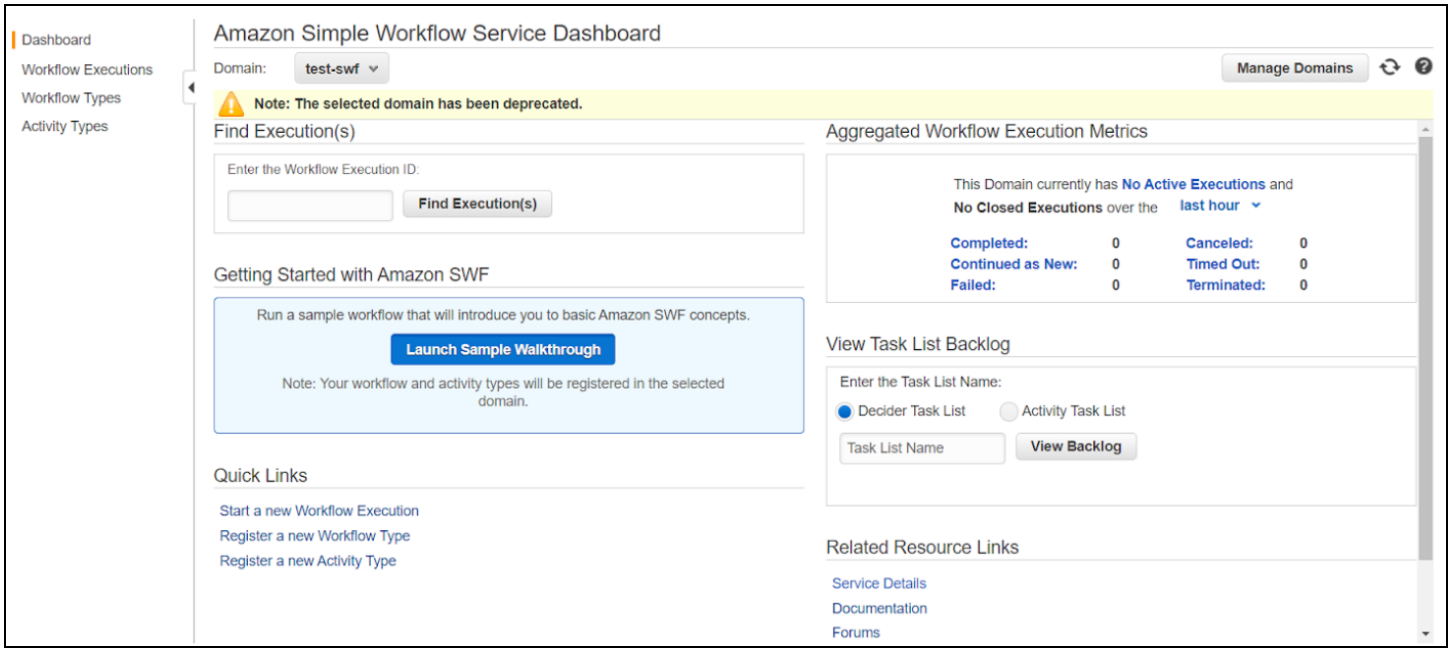
Workflow Starter
- Any application that can initiate workflow executions.

Activity Worker
- An **activity worker** is a program that receives activity tasks, performs them, and provides results back.
- Implement workers to perform tasks. These workers can run either on cloud infrastructure, or on your own premises.
- Different activity workers can be written in different programming languages and run on different operating systems.
- Assigning particular tasks to particular activity workers is called *task routing*. Task routing is optional.

Decider
- A software program that contains the coordination logic in a workflow.
- It schedules activity tasks, provides input data to the activity workers, processes events that arrive while the workflow is in progress, and ultimately ends the workflow when the objective has been completed.
- Both activity workers and the decider receive their tasks by polling the SWF service.

Workflow Execution History
- The workflow execution history is composed of events, where an event represents a significant change in the state of the workflow execution.
- SWF informs the decider of the state of the workflow by including, with each decision task, a copy of the current workflow execution history.

Polling
- Deciders and activity workers communicate with SWF using *long polling*.



Workflow Execution

1. Write activity workers that implement the processing steps in your workflow.
2. Write a decider to implement the coordination logic of your workflow.
3. Register your activities and workflow with Amazon SWF.
4. Start your activity workers and decider.
5. Start one or more executions of your workflow. Each execution runs independently and you can provide each with its own set of input data. When an execution is started, Amazon SWF schedules the initial decision task. In response, your decider begins generating decisions which initiate activity tasks. Execution continues until your decider makes a decision to close the execution.
6. Filter and view complete details of running as well as completed executions.

SWF provides service operations that are **accessible through HTTP requests.**

Endpoints

- To reduce latency and to store data in a location that meets your requirements, SWF provides endpoints in different regions.
- Each endpoint is completely independent. When you register an SWF domain, workflow or activity, it exists only within the region you registered it in.

AWS Flow Framework

- An enhanced SDK for writing distributed, asynchronous programs that can run as workflows on SWF.

- It is available for the Java and Ruby programming languages, and it provides classes that simplify writing complex distributed programs.

Pricing

- You pay for workflow executions when you start them and for each 24-hour period until they are completed. The first 24 hours of workflow execution are free.
- You pay for "markers" (custom workflow execution log entries), start timers, or receive signals of additional tasks generated from a workflow.
- Data transferred between SWF and other AWS services **within a single region is free** of charge. Data transferred between SWF and other AWS services **in different regions will be charged** at Internet Data Transfer rates on both sides of the transfer. First 1 GB of transferred data is free.

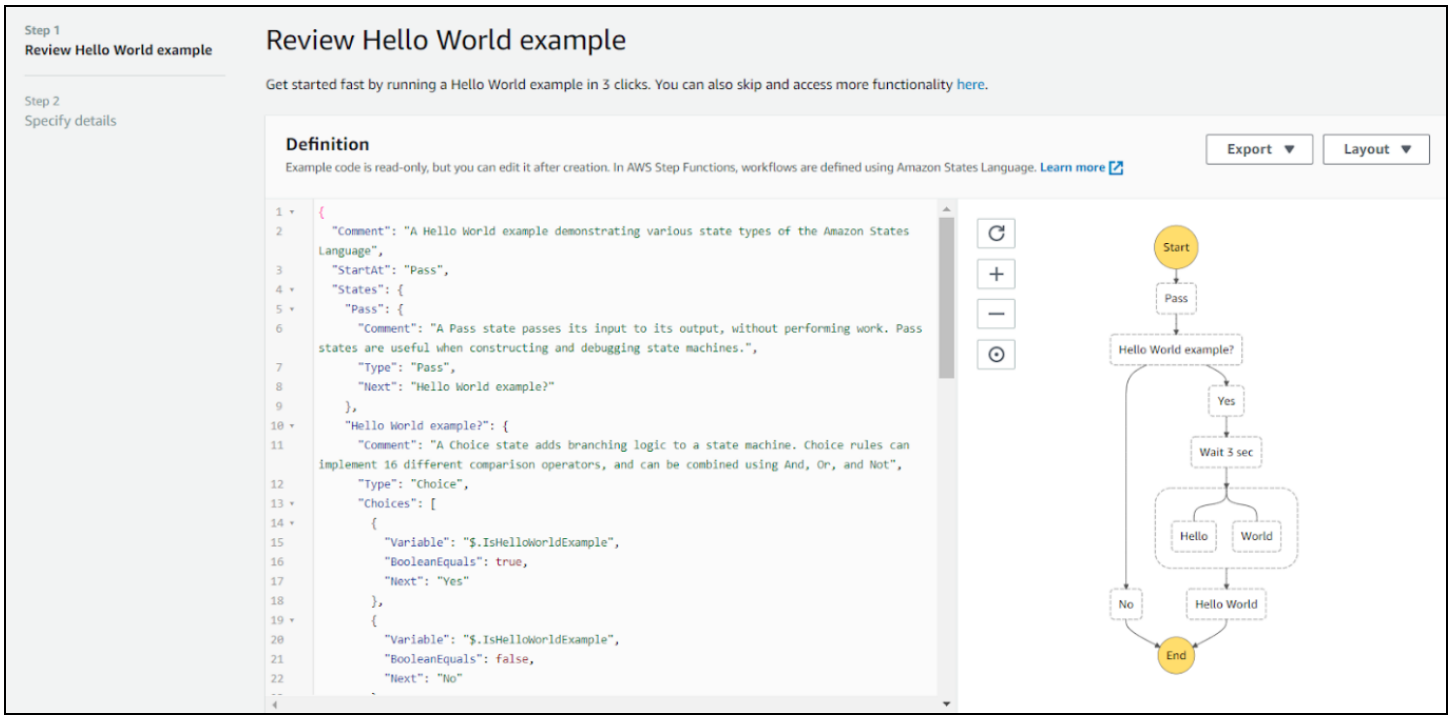
SQS vs SWF

- SQS offers reliable, highly-scalable hosted queues for storing messages while they travel between applications or microservices.
- SWF is a web service that makes it easy to coordinate work across distributed application components.
- SWF API actions are task-oriented. SQS API actions are message-oriented.
- SWF keeps track of all tasks and events in an application. SQS requires you to implement your own application-level tracking.
- SWF offers several features that facilitate application development, such as passing data between tasks, signaling, and flexibility in distributing tasks.

Sources:
https://docs.aws.amazon.com/amazonswf/latest/developerguide/
https://aws.amazon.com/swf/faqs/
https://aws.amazon.com/swf/pricing/

AWS Step Functions

- AWS Step Functions is a web service that provides **serverless orchestration** for modern applications. It enables you to coordinate the components of distributed applications and microservices using visual workflows.
- Concepts
    - Step Functions is based on the concepts of **tasks** and **state machines**.
        - A task performs work by using an activity or an AWS Lambda function, or by passing parameters to the API actions of other services.
        - A finite state machine can express an algorithm as a number of states, their relationships, and their input and output.
    - You define state machines using the **JSON-based Amazon States Language**.



    - A state is referred to by its *name*, which can be any string, but which must be unique within the scope of the entire state machine. An instance of a state exists until the end of its execution.
        - There are 6 types of states:
            - Task state - Do some work in your state machine. AWS Step Functions can invoke Lambda functions directly from a task state.
            - Choice state - Make a choice between branches of execution
            - Fail or Succeed state - Stop an execution with a failure or success
            - Pass state - Simply pass its input to its output or inject some fixed data

        - Wait state - Provide a delay for a certain amount of time or until a specified time/date
        - Parallel state - Begin parallel branches of execution
    - Common features between states
        - Each state must have a *Type* field indicating what type of state it is.
        - Each state can have an optional *Comment* field to hold a human-readable comment about, or description of, the state.
        - Each state (except a Succeed or Fail state) requires a *Next* field or, alternatively, can become a terminal state by specifying an ***End*** field.
- **Activities** enable you to place a task in your state machine where the work is performed by an ***activity worker*** that can be hosted on Amazon EC2, Amazon ECS, or mobile devices.
- Activity tasks let you assign a specific step in your workflow to code running in an activity worker. Service tasks let you connect a step in your workflow to a supported AWS service.
- With **Transitions**, after executing a state, AWS Step Functions uses the value of the *Next* field to determine the next state to advance to. States can have multiple incoming transitions from other states.
- State machine data is represented by JSON text. It takes the following forms:
    - The initial input into a state machine
    - Data passed between states
    - The output from a state machine
- Individual states receive JSON as input and usually pass JSON as output to the next state.
- A **state machine execution** occurs when a state machine runs and performs its tasks. Each Step Functions state machine can have multiple simultaneous executions.
- **State machine updates** in AWS Step Functions are **eventually consistent**.
- By default, when a state reports an error, AWS Step Functions causes the execution to **fail entirely**.
    - Task and Parallel states can have a field named *Retry* and *Catch* to retry an execution or to have a fallback state.
- The Step Functions console displays a graphical view of your state machine's structure, which provides a way to visually check a state machine's logic and monitor executions.

Features

- Using Step Functions, you define your **workflows as state machines**, which transform complex code into easy to understand statements and diagrams.
- Step Functions provides ready-made steps for your workflow called **states** that implement basic service primitives for you, which means you can remove that logic from your application. States are able to:
    - pass data to other states and microservices,
    - handle exceptions,
    - add timeouts,
    - make decisions,
    - execute multiple paths in parallel,

        - and more.
    - Using Step Functions **service tasks**, you can configure your Step Functions workflow to call other AWS services.
    - Step Functions can coordinate any application that can make an **HTTPS** connection, regardless of where it is hosted—Amazon EC2 instances, mobile devices, or on-premises servers.
    - AWS Step Functions coordinates your existing Lambda functions and microservices, and lets you modify them into new compositions. The tasks in your workflow can run anywhere, including on instances, containers, functions, and mobile devices.
    - Nesting your Step Functions workflows allows you to build larger, more complex workflows out of smaller, simpler workflows.
    - Step Functions keeps the logic of your application strictly separated from the implementation of your application. You can add, move, swap, and reorder steps without having to make changes to your business logic.
    - Step Functions maintains the state of your application during execution, including tracking what step of execution it is in, and storing data that is moving between the steps of your workflow. You won't have to manage state yourself with data stores or by building complex state management into all of your tasks.
    - Step Functions automatically handles errors and exceptions with **built-in try/catch and retry**, whether the task takes seconds or months to complete. You can automatically retry failed or timed-out tasks, respond differently to different types of errors, and recover gracefully by falling back to designated cleanup and recovery code.
    - Step Functions has **built-in fault tolerance and maintains service capacity across multiple Availability Zones in each region**, ensuring high availability for both the service itself and for the application workflow it operates.
    - Step Functions **automatically scales** the operations and underlying compute to run the steps of your application for you in response to changing workloads.
    - AWS Step Functions has a 99.9% SLA.
    - AWS Step Functions enables access to metadata about workflow executions so that you can easily identify related resources. For example, your workflow can retrieve its execution ID or a timestamp indicating when a task started. This makes it easier to correlate logs for faster debugging and to measure workflow performance data.
    - It also supports callback patterns. Callback patterns automate workflows for applications with human activities and custom integrations with third-party services.
    - AWS Step Functions supports workflow execution events, which make it faster and easier to build and monitor event-driven, serverless workflows. Execution event notifications can be automatically delivered when a workflow starts or completes through CloudWatch Events, reaching targets like AWS Lambda, Amazon SNS, Amazon Kinesis, or AWS Step Functions for automated response to the event.

How Step Functions Work

- Orchestration centrally manages a workflow by breaking it into multiple steps, adding flow logic, and tracking the inputs and outputs between the steps.

    - As your applications execute, Step Functions maintains application state, tracking exactly which workflow step your application is in, and stores an event log of data that is passed between application components. That means that if networks fail or components hang, your application can pick up right where it left off.
- Amazon States Language
    - A JSON-based, structured language used to define your state machine, a collection of states, that can do work (*Task states*), determine which states to transition to next (*Choice states*), stop an execution with an error (*Fail states*), and so on.
    - When creating a state machine, only the *States* field is required in the JSON document.
- Security and Monitoring
    - AWS Step Functions can execute code and access AWS resources (such as invoking an AWS Lambda function). To maintain security, you must grant Step Functions access to those resources by using an IAM role.
    - Step Functions is a HIPAA eligible service and is also compliant with SOC, PCI, and FedRAMP.
    - Step Functions delivers real-time diagnostics and dashboards, integrates with Amazon CloudWatch and AWS CloudTrail, and logs every execution, including overall state, failed steps, inputs, and outputs.
- Pricing
    - Step Functions counts a state transition each time a step of your workflow is executed. You are charged for the total number of state transitions across all your state machines, including retries.
- Common Use Cases
    - Step Functions can help ensure that long-running, multiple ETL jobs execute in order and complete successfully, instead of manually orchestrating those jobs or maintaining a separate application.
    - By using Step Functions to handle a few tasks in your codebase, you can approach the transformation of monolithic applications into microservices as a series of small steps.
    - You can use Step Functions to easily automate recurring tasks such as patch management, infrastructure selection, and data synchronization, and Step Functions will automatically scale, respond to timeouts, and retry failed tasks.
    - Use Step Functions to combine multiple AWS Lambda functions into responsive serverless applications and microservices, without having to write code for workflow logic, parallel processes, error handling, timeouts or retries.
    - You can also orchestrate data and services that run on Amazon EC2 instances, containers, or on-premises servers.

Sources:
https://aws.amazon.com/step-functions/features/
https://aws.amazon.com/step-functions/pricing/
https://docs.aws.amazon.com/step-functions/latest/dg/welcome.html
https://aws.amazon.com/step-functions/faqs/

COMPARISON OF AWS SERVICES

S3 vs EBS vs EFS

Tutorials Dojo	S3	EBS	EFS
Type of storage	Object storage. You can store virtually any kind of data in any format.	Persistent block level storage for EC2 instances.	POSIX-compliant file storage for EC2 instances
Features	Accessible to anyone or any service with the right permissions	Deliver performance for workloads that require the lowest-latency access to data from a single EC2 instance	Has a file system interface, file system access semantics (such as strong consistency and file locking), and concurrently-accessible storage for multiple EC2 instances
Max Storage Size	Virtually unlimited	16 TiB for one volume	Unlimited system size
Max File Size	Individual Amazon S3 objects can range in size to a maximum of 5 terabytes.	Equivalent to the maximum size of your volumes	47.9 TiB for a single file
Performance (Latency)	Low, for mixed request types, and integration with CloudFront	Lowest, consistent; SSD-backed storages include the highest performance Provisioned OPS SSD and General Purpose SSD that balance price and performance.	Low, consistent; use Max I/O mode for higher performance
Performance (Throughput)	Multiple GBs per second; supports multi-part upload	Up to 2 GB per second. HDD-backed volumes include Throughput Optimized HDD for frequently accessed, throughput intensive workloads and Cold HDD for less frequently accessed data.	10+ GB per second. Bursting Throughput mode scales with the size of the file system. Provisioned Throughput mode offers higher dedicated throughput than burstring throughput.
Durability	Stored redundantly across multiple AZs; has 99.999999999% durability	Stored redundantly in a single AZ	Stored redundantly across multiple AZs
Availability	S3 Standard - 99.99% availability S3 Standard-IA - 99.9% availability S3 One Zone-IA - 99.5% availability. S3 Intelligent Tiering - 99.9%	Has 99.999% availability	99.9% SLA. Runs in multi-AZ

Tutorials Dojo	S3	EBS	EFS
Scalability	Highly scalable	Manually increase/decrease your memory size. Attach and detach additional volumes to and from your EC2 instance to scale.	EFS file systems are elastic, and automatically grow and shrink as you add and remove files.
Data Accessing	One to millions of connections over the web; S3 provides a REST web services interface	Single EC2 instance in a single AZ Amazon EBS Multi-Attach enables you to attach a single Provisioned IOPS SSD (io1 or io2) volume to up to 16 Nitro-based instances that are in the same Availability Zone.	One to thousands of EC2 instances or on-premises servers, from multiple AZs, regions, VPCs, and accounts concurrently
Access Control	Uses bucket policies and IAM user policies. Has *Block Public Access* settings to help manage public access to resources.	IAM Policies, Roles, and Security Groups	Only resources that can access endpoints in your VPC, called a *mount target*, can access your file system; POSIX-compliant user and group-level permissions
Encryption Methods	Supports SSL endpoints using the HTTPS protocol, Client-Side and Server-Side Encryption (SSE-S3, SSE-C, SSE-KMS)	Encrypts both data-at-rest and data-in-transit through EBS encryption that uses AWS KMS CMKs.	Encrypt data at rest and in transit. Data at rest encryption uses AWS KMS. Data in-transit uses TLS.
Backup and Restoration	Use versioning or cross-region replication	All EBS volume types offer durable snapshot capabilities.	EFS to EFS replication through third party tools or AWS DataSync
Pricing	Billing prices are based on the location of your bucket. Lower costs equals lower prices. You get cheaper prices the more you use S3 storage.	You pay GB-month of provisioned storage, provisioned IOPS-month, GB-month of snapshot data stored in S3	You pay for the amount of file system storage used per month. When using the Provisioned Throughput mode you pay for the throughput you provision per month.
Use Cases	Web serving and content management, media and entertainment, backups, big data analytics, data lake	Boot volumes, transactional and NoSQL databases, data warehousing & ETL	Web serving and content management, enterprise applications, media and entertainment, home directories, database backups, developer tools, container storage, big data analytics
Service endpoint	Can be accessed within and outside a VPC (via S3 bucket URL)	Accessed within one's VPC	Accessed within one's VPC

Amazon S3 vs Glacier

- Amazon S3 is a durable, secure, simple, and fast storage service, while Amazon S3 Glacier is used for archiving solutions.
- Use S3 if you need low latency or frequent access to your data. Use S3 Glacier for low storage cost, and you do not require millisecond access to your data.
- You have three retrieval options when it comes to Glacier, each varying in the cost and speed it retrieves an object for you. You retrieve data in milliseconds from S3.
- Both S3 and Glacier are designed for durability of 99.999999999% of objects across multiple Availability Zones.
- S3 and Glacier are designed for availability of 99.99%.
- S3 can be used to host static web content, while Glacier cannot.
- In S3, users create buckets. In Glacier, users create archives and vaults.
- You can store a virtually unlimited amount of data in both S3 and Glacier.
- A single Glacier archive can contain 40TB of data.
- S3 supports Versioning.
- You can run analytics and querying on S3.
- You can configure a lifecycle policy for your S3 objects to automatically transfer them to Glacier. You can also upload objects directly to either S3 or Glacier.
- S3 Standard-IA and One Zone-IA have a minimum capacity charge per object of 128KB. Glacier's minimum is 40KB.
- Objects stored in S3 have a minimum storage duration of 30 days (except for S3 Standard). Objects that are archived to Glacier have a minimum 90 days of storage. Objects that are deleted, overwritten, or transitioned to a different storage class before the minimum duration will incur the normal usage charge plus a pro-rated request charge for the remainder of the minimum storage duration.
- Glacier has a per GB retrieval fee.
- You can transition objects from some S3 storage classes to another. Glacier objects can only be transitioned to the Glacier Deep Archive storage class.
- S3 (standard, intelligent-tiering, standard-IA, and one zone-IA) and Glacier are backed by an SLA.

S3 Standard vs S3 Standard-IA vs S3 One Zone-IA vs S3 Intelligent Tiering

	S3 Standard	S3 Standard-Infrequent Access (IA)	S3 One Zone-Infrequent Access (IA)	S3 Intelligent Tiering
Features	General-purpose storage of frequently accessed data	For long-lived, rapid but less frequently accessed data; data is stored redundantly in multiple AZs	For long-lived, rapid but less frequently accessed data; data is stored redundantly in only one AZ of your choice	For long-lived data that have unpredictable access patterns
Durability	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)
Availability	99.99%	99.9%	99.5%	99.9%
Availability SLA	99.9%	99%	99%	99%
Number of Availability Zones	At least 3	At least 3	Only 1	At least 3
Minimum capacity charge per object	N/A	128KB	128KB	N/A
Minimum storage duration charge	N/A	30 days	30 days	30 days
Inserting data	Directly PUT into S3 Standard	Directly PUT into S3 Standard-IA or set Lifecycle policies to transition objects from the S3 Standard to the S3 Standard-IA storage class.	Directly PUT into S3 One Zone-IA or set Lifecycle policies to transition objects from the S3 Standard to the S3 One Zone-IA storage class.	Directly PUT into S3 Intelligent-Tiering or set Lifecycle policies to transition objects from the S3 Standard to the S3 Intelligent-Tiering storage class.
Retrieval fee	N/A	per GB retrieved	per GB retrieved	N/A
First byte latency	milliseconds	milliseconds	milliseconds	milliseconds
Storage transition	S3 Standard to all other S3 storage types including Glacier	S3 Standard-IA to S3 One Zone-IA or S3 Glacier	S3 One Zone-IA to S3 Glacier	S3 Intelligent to S3 One Zone-IA or S3 Glacier
Use Cases	Cloud applications, dynamic websites, content distribution, mobile and gaming applications, and big data analytics.	Ideally suited for long-term file storage, older sync and share storage, and other aging data.	For infrequently-accessed storage, like backup copies, disaster recovery copies, or other easily recreatable data.	Data with unknown or changing access patterns, optimize storage costs automatically, and unpredictable workloads

Tutorials Dojo

Additional Notes:

- Data stored in the S3 One Zone-IA storage class will be lost in the event of AZ destruction.
- S3 Standard-IA costs less than S3 Standard in terms of storage price, while still providing the same high durability, throughput, and low latency of S3 Standard.
- S3 One Zone-IA has 20% less cost than Standard-IA.
- It is recommended to use multipart upload for objects larger than 100MB.

AWS DataSync vs Storage Gateway

	DataSync	Storage Gateway
Description	AWS DataSync is an online data transfer service that simplifies, automates, and accelerates the process of copying large amounts of data to and from AWS storage services over the Internet or over AWS Direct Connect.	AWS Storage Gateway is a hybrid cloud storage service that gives you on-premises access to virtually unlimited cloud storage by linking it to S3. Storage Gateway provides 3 types of storage interfaces for your on-premises applications: file, volume, and tape.
How it Works	Uses an **agent** which is a virtual machine (VM) that is owned by the user and is used to read or write data from your storage systems. You can activate the agent from the Management Console. The agent will then read from a source location, and sync your data to Amazon S3, Amazon EFS, or Amazon Fsx for Windows File Server.	Uses a **Storage Gateway Appliance** - a VM from Amazon - which is installed and hosted on your data center. After the setup, you can use the AWS console to provision your storage options: File Gateway, Cached Volumes, or Stored Volumes, in which data will be saved to Amazon S3. You can also purchase the hardware appliance to facilitate the transfer instead of installing the VM.
Protocol	DataSync connects to existing storage systems and data sources with standard storage protocols (NFS, SMB), or using the Amazon S3 API.	Storage Gateway provides a standard set of storage protocols such as iSCSI, SMB, and NFS.
Storage	AWS DataSync can copy data between Network File Systems (NFS), SMB file servers or self-managed object storages. It can also move data between your on-premises storage and AWS Snowcone, Amazon S3, Amazon EFS, or Amazon FSx.	File Gateway enables you to store and retrieve objects in Amazon S3 using file protocols such as NFS and SMB. Volume Gateway stores your data locally in the gateway and syncs them to Amazon S3. It also allows you to take point-in-time copies of your volumes with EBS snapshots which you can restore and mount to your appliances as iSCSI devices. Tape Gateway data is immediately stored in Amazon S3 and can be archived to Amazon S3 Glacier or Amazon S3 Glacier Deep Archive.
Pricing	You are charged standard request, storage, and data transfer rates to read from and write to AWS services, such as Amazon S3, Amazon EFS, Amazon FSx for Windows File Server, and AWS KMS.	You are charged based on the type and amount of storage you use, the requests you make, and the amount of data transferred out of AWS.
Combination	You can use a combination of DataSync and File Gateway to minimize your on-premises' operational costs while seamlessly connecting on-premises applications to your cloud storage. AWS DataSync enables you to automate and accelerate online data transfers to AWS storage services. File Gateway then provides your on-premises applications with low latency access to the migrated data. Tutorials Dojo	

RDS vs DynamoDB

Tutorials Dojo	RDS	DynamoDB
Type of database	Managed relational (SQL) database	Fully managed key-value and document (NoSQL) database
Features	Has several database instance types for different kinds of workloads and supports six database engines - Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database, and SQL Server.	Delivers single-digit millisecond performance at any scale.
Storage Size	- 128 TB for Aurora engine. - 64 TB for MySQL, MariaDB, Oracle, and PostgreSQL engines. - 16 TB for SQL Server engine.	Supports tables of virtually any size.
Number of tables per unit	Depends on the database engine	256
Performance	General Purpose Storage is an SSD-backed storage option that delivers a consistent baseline of 3 IOPS per provisioned GB with the ability to burst up to 3,000 IOPS. Provisioned IOPS Storage is an SSD-backed storage option designed to deliver a consistent IOPS rate that you specify when creating a database instance, up to 40,000 IOPS per database instance. Amazon RDS provisions that IOPS rate for the lifetime of the database instance. Optimized for OLTP database workloads. Magnetic – Amazon RDS also supports magnetic storage for backward compatibility.	Single-digit millisecond read and write performance. Can handle more than 10 trillion requests per day with peaks greater than 20 million requests per second, over petabytes of storage. DynamoDB Accelerator (DAX) is an in-memory cache that can improve the read performance of your DynamoDB tables by up to 10 times—taking the time required for reads from milliseconds to microseconds, even at millions of requests per second. You specify the read and write throughput for each of your tables.
Availability and durability	Amazon RDS Multi-AZ deployments synchronously replicates your data to a standby instance in a different Availability Zone Amazon RDS will automatically replace the compute instance powering your deployment in the event of a hardware failure..	DynamoDB global tables replicate your data automatically across 3 Availability Zones of your choice of AWS Regions and automatically scale capacity to accommodate your workloads.
Backups	The automated backup feature enables point-in-time recovery for your database instance. Database snapshots are user-initiated backups of your instance stored in Amazon S3 that are kept until you explicitly delete them.	Point-in-time recovery (PITR) provides continuous backups of your DynamoDB table data, and you can restore that table to any point in time up to the second during the preceding 35 days. On-demand backup and restore allows you to create full backups of your DynamoDB tables' data for data archiving.

	RDS	DynamoDB
Scalability	The Amazon Aurora engine will automatically grow the size of your database volume. The MySQL, MariaDB, SQL Server, Oracle, and PostgreSQL engines allow you to scale on-the-fly with zero downtime. RDS also supports storage auto scaling Read replicas are available in Amazon RDS for MySQL, MariaDB, and PostgreSQL as well as Amazon Aurora.	Support tables of virtually any size with horizontal scaling. For tables using on-demand capacity mode, DynamoDB instantly accommodates your workloads as they ramp up or down to any previously reached traffic level. For tables using provisioned capacity, DynamoDB delivers automatic scaling of throughput and storage based on your previously set capacity.
Security	Isolate your database in your own virtual network. Connect to your on-premises IT infrastructure using industry-standard encrypted IPsec VPNs. You can configure firewall settings and control network access to your database instances. Integrates with IAM.	Integrates with IAM.
Encryption	Encrypt your databases using keys you manage through AWS KMS. With encryption enabled, data stored at rest is encrypted, as are its automated backups, read replicas, and snapshots. Supports Transparent Data Encryption in SQL Server and Oracle. Supports the use of SSL to secure data in transit.	DynamoDB encrypts data at rest by default using encryption keys stored in AWS KMS.
Maintenance	Amazon RDS will update databases with the latest patches. You can exert optional control over when and if your database instance is patched.	No maintenance since DynamoDB is serverless.
Pricing	A monthly charge for each database instance that you launch. Option to reserve a DB instance for a one or three year term and receive discounts in pricing, compared to On-Demand instance pricing.	Charges for reading, writing, and storing data in your DynamoDB tables, along with any optional features you choose to enable. There are specific billing options for each of DynamoDB's capacity modes.
Use Cases	Traditional applications, ERP, CRM, and e-commerce.	Internet-scale applications, real-time bidding, shopping carts, and customer preferences, content management, personalization, and mobile applications. Tutorials Dojo

Additional notes:

- DynamoDB has built-in support for ACID transactions.
- DynamoDB uses filter expressions because it does not support complex queries.
- Multi-AZ deployments for the MySQL, MariaDB, Oracle, and PostgreSQL engines utilize synchronous physical replication. Multi-AZ deployments for the SQL Server engine use synchronous logical replication.

RDS vs Aurora

	Aurora	RDS
Type of database	Relational database	
Features	• MySQL and PostgreSQL compatible. • 5x faster than standard MySQL databases and 3x faster than standard PostgreSQL databases. • Use Parallel Query to run transactional and analytical workloads in the same Aurora database, while maintaining high performance. • You can distribute and load balance your unique workloads across different sets of Aurora DB instances using custom endpoints. • Aurora Serverless allows for on-demand, autoscaling of your Aurora DB instance capacity.	• Has several database instance types for different kinds of workloads and supports five database engines - MySQL, PostgreSQL, MariaDB, Oracle, and SQL Server. • Can use either General Purpose Storage and Provisioned IOPS storage to deliver a consistent IOPS performance
Maximum storage capacity	• 128 TB	• 64 TB for MySQL, MariaDB, Oracle, and PostgreSQL engines • 16 TB for SQL Server engine
DB instance classes	• Memory Optimized classes - for workloads that need to process large data sets in memory. • Burstable classes - provides the instance the ability to burst to a higher level of CPU performance when required by the workload.	• Standard classes - for a wide range of workloads, you can use general purpose instance. It offers a balance of compute, memory, and networking resources. • Memory Optimized classes - for workloads that need to process large data sets in memory. • Burstable classes - provides the instance the ability to burst to a

		higher level of CPU performance when required by the workload.
Availability and durability	• Amazon Aurora uses RDS Multi-AZ technology to automate failover to one of up to 15 Amazon Aurora Replicas across three Availability Zones • Amazon Aurora Global Database uses storage-based replication to replicate a database across multiple AWS Regions, with typical latency of less than 1 second. • Self-healing: data blocks and disks are continuously scanned for errors and replaced automatically.	• Amazon RDS Multi-AZ deployments synchronously replicates your data to a standby instance in a different Availability Zone. • Amazon RDS will automatically replace the compute instance powering your deployment in the event of a hardware failure.
Backups	• Point-in-time recovery to restore your database to any second during your retention period, up to the last five minutes. • Automatic backup retention period up to thirty-five days. • Backtrack to the original database state without needing to restore data from a backup.	• The automated backup feature enables point-in-time recovery for your database instance. • Database snapshots are user-initiated backups of your instance stored in Amazon S3 that are kept until you explicitly delete them.

Scalability	• Aurora automatically increases the size of your volumes as your database grows larger (increments of 10 GB). • Aurora also supports replica auto-scaling, where it automatically adds and removes DB replicas in response to changes in performance metrics. • Cross-region replicas provide fast local reads to your users, and each region can have an additional 15 Aurora replicas to further scale local reads.	• The MySQL, MariaDB, SQL Server, Oracle, and PostgreSQL engines scale your storage automatically as your database workload grows with zero downtime. • Read replicas are available for Amazon RDS for MySQL, MariaDB, PostgreSQL, Oracle, and SQL Server. Amazon RDS creates a second DB instance using a snapshot of the source DB instance and uses the engines' native asynchronous replication to update the read replica whenever there is a change to the source. • Can scale compute and memory resources (vertically) of up to a maximum of 32 vCPUs and 244 GiB of RAM.
Security	• Isolate the database in your own virtual network via VPC. • Connect to your on-premises IT infrastructure using encrypted IPsec VPNs or Direct Connect and VPC Endpoints. • Configure security group firewall and network access rules to your database instances. • Integrates with IAM.	
Encryption	• Encrypt your databases using keys you manage through AWS KMS. With Amazon Aurora encryption, data stored at rest is encrypted, as are its automated backups, snapshots, and replicas in the same cluster. • Supports the use of SSL (AES-256) to secure data in transit.	• Encrypt your databases using keys you manage through AWS KMS. With Amazon RDS encryption, data stored at rest is encrypted, as are its automated backups, read replicas, and snapshots. • Supports Transparent Data Encryption in SQL Server and Oracle. • Supports the use of SSL to secure data in transit

DB Authentication	• Password authentication • Password and IAM database authentication	• Password authentication • Password and IAM database authentication • Password and Kerberos authentication
Maintenance	• Amazon Aurora automatically updates the database with the latest patches. • Amazon Aurora Serverless enables you to run your database in the cloud without managing/maintaining any database infrastructure.	• Amazon RDS will update databases with the latest major and minor patches on scheduled maintenance windows. You can exert optional control over when and if your database instance is patched.
Monitoring	• Use Enhanced Monitoring to collect metrics from the operating system instance. • Use Performance Insights to detect database performance problems and take corrective action. • Uses Amazon SNS to receive a notification on database events.	
Pricing	• A monthly charge for each database instance that you launch if you use on-demand. This includes both the instance compute capacity and the amount of storage being used. • Option to reserve a DB instance for a one or three-year term (reserve instances) and receive discounts in pricing.	

Use Cases	• Enterprise applications - a great option for any enterprise application that uses relational database since it handles provisioning, patching, backup, recovery, failure detection, and repair. • SaaS applications - without worrying about the underlying database that powers the application, you can concentrate on building high-quality applications. • Web and mobile gaming - since games need a database with high throughput, storage scalability, and must be highly available. Aurora suits the variable use pattern of these apps perfectly.	• Web and mobile applications - since the application needs a database with high throughput, storage scalability, and must be highly available. RDS also fulfills the needs of such highly demanding apps. • E-commerce applications - a managed database service that offers PCI compliance. You can just focus on building high-quality customer experiences without thinking of the underlying database. • Mobile and online games - game developers don't need to worry about provisioning, scaling, and monitoring of database servers since RDS manages the database infrastructure.

CloudTrail vs CloudWatch

- CloudWatch is a monitoring service for AWS resources and applications. CloudTrail is a web service that records API activity in your AWS account. They are both useful monitoring tools in AWS.
- By default, CloudWatch offers free basic monitoring for your resources, such as EC2 instances, EBS volumes, and RDS DB instances. CloudTrail is also enabled by default when you create your AWS account.
- With CloudWatch, you can collect and track metrics, collect and monitor log files, and set alarms. CloudTrail, on the other hand, logs information on who made a request, the services used, the actions performed, parameters for the actions, and the response elements returned by the AWS service. CloudTrail Logs are then stored in an S3 bucket or a CloudWatch Logs log group that you specify.
- You can enable detailed monitoring from your AWS resources to send metric data to CloudWatch more frequently, with an additional cost.
- CloudTrail delivers one free copy of management event logs for each AWS region. Management events include management operations performed on resources in your AWS account, such as when a user logs in to your account. Logging data events are charged. Data events include resource operations performed on or within the resource itself, such as S3 object-level API activity or Lambda function execution activity.
- CloudTrail helps you ensure compliance and regulatory standards.
- CloudWatch Logs reports on application logs, while CloudTrail Logs provide you specific information on what occurred in your AWS account.
- CloudWatch Events is a near real time stream of system events describing changes to your AWS resources. CloudTrail focuses more on AWS API calls made in your AWS account.
- Typically, CloudTrail delivers an event within 15 minutes of the API call. CloudWatch delivers metric data in 5 minutes periods for basic monitoring and 1 minute periods for detailed monitoring. The CloudWatch Logs Agent will send log data every five seconds by default.

Security Group vs NACL

Security Group	Network Access Control List
Acts as a firewall for associated Amazon EC2 instances	Acts as a firewall for associated subnets
Controls both inbound and outbound traffic at the instance level	Controls both inbound and outbound traffic at the subnet level
You can secure your VPC instances using only security groups	Network ACLs are an additional layer of defense.
Supports allow rules only	Supports allow rules and deny rules
Stateful (Return traffic is automatically allowed, regardless of any rules)	Stateless (Return traffic must be explicitly allowed by rules)
Evaluates all rules before deciding whether to allow traffic	Evaluates rules in number order when deciding whether to allow traffic, starting with the lowest numbered rule.
Applies only to the instance that is associated to it	Applies to all instances in the subnet it is associated with
Has separate rules for inbound and outbound traffic	Has separate rules for inbound and outbound traffic
A newly created security group denies all inbound traffic by default	A newly created nACL denies all inbound traffic by default
A newly created security group has an outbound rule that allows all outbound traffic by default	A newly created nACL denies all outbound traffic default
Instances associated with a security group can't talk to each other unless you add rules allowing it	Each subnet in your VPC must be associated with a network ACL. If none is associated, the default nACL is selected.
Security groups are associated with network interfaces	You can associate a network ACL with multiple subnets; however, a subnet can be associated with only one network ACL at a time.

Tutorials Dojo

Your VPC has a default security group with the following rules:

1. Allow inbound traffic from instances assigned to the same security group.
2. Allow all outbound IPv4 traffic and IPv6 traffic if you have allocated an IPv6 CIDR block.

Your VPC has a default network ACL with the following rules:

1. Allows all inbound and outbound IPv4 traffic and, if applicable, IPv6 traffic.
2. Each network ACL also includes a non modifiable and non removable rule whose rule number is an asterisk. This rule ensures that if a packet doesn't match any of the other numbered rules, it's denied.

EBS-SSD vs HDD

On a given volume configuration, certain I/O characteristics drive the performance behavior for your EBS volumes. SSD-backed volumes, such as General Purpose SSD (gp2) and Provisioned IOPS SSD (io1, io2), deliver consistent performance whether an I/O operation is random or sequential. HDD-backed volumes like Throughput Optimized HDD (st1) and Cold HDD (sc1) deliver optimal performance only when I/O operations are large and sequential.

In the exam, always consider the difference between SSD and HDD as shown on the table below. This will allow you to easily eliminate specific EBS-types in the options which are not SSD or not HDD, depending on whether the question asks for a storage type which has ***small, random*** I/O operations or ***large, sequential*** I/O operations.

FEATURES	SSD Solid State Drive	HDD Hard Disk Drive
Best for workloads with:	***small, random*** I/O operations	***large, sequential*** I/O operations
Can be used as a bootable volume?	Yes	No
Suitable Use Cases	- Best for **transactional workloads** - Critical business applications that require sustained IOPS performance - Large database workloads such as MongoDB, Oracle, Microsoft SQL Server and many others…	- Best for ***large streaming workloads*** requiring consistent, fast throughput at a low price - Big data, Data warehouses, Log processing - Throughput-oriented storage for large volumes of data that is ***infrequently*** accessed
Cost	moderate / high	low
Dominant Performance Attribute	IOPS	Throughput (`MiB/s`)

Provisioned IOPS SSD (io1, io2) volumes are designed to meet the needs of I/O-intensive workloads, particularly database workloads, that are sensitive to storage performance and consistency. Unlike gp2, which uses a bucket and credit model to calculate performance, an io1 volume allows you to specify a consistent IOPS rate when you create the volume, and Amazon EBS delivers within 10 percent of the provisioned IOPS performance 99.9 percent of the time over a given year. Provisioned IOPS SSD io2 is an upgrade of Provisioned IOPS SSD io1. It offers higher 99.999% durability and higher IOPS per GiB ratio with 500 IOPS per GiB, all at the same cost as io1 volumes.

Volume Name	General Purpose SSD		Provisioned IOPS SSD	
Volume type	gp3	gp2	io2	io1
Description	General Purpose SSD volume that balances price performance for a wide variety of transactional workloads	General Purpose SSD volume that balances price performance for a wide variety of transactional workloads	High performance SSD volume designed for **business-critical** latency-sensitive applications	High performance SSD volume designed for latency-sensitive transactional workloads
Use Cases	virtual desktops, medium sized single instance databases such as MSFT SQL Server and Oracle DB, low-latency interactive apps, dev & test, boot volumes	Boot volumes, low-latency interactive apps, dev & test	Workloads that require sub-millisecond latency, and sustained IOPS performance or more than 64,000 IOPS or 1,000 MiB/s of throughput	Workloads that require sustained IOPS performance or more than 16,000 IOPS and I/O-intensive database workloads
Volume Size	1 GB – 16 TB	1 GB – 16 TB	4 GB – 16 TB	4 GB – 16 TB
Durability	99.8% - 99.9% durability	99.8% - 99.9% durability	99.999%	99.8% - 99.9%
Max IOPS / Volume	16,000	16,000	64,000	64,000
Max Throughput / Volume	1000 MB/s	250 MB/s	1,000 MB/s	1,000 MB/s
Max IOPS / Instance	260,000	260,000	160,000	260,000
Max IOPS / GB	N/A	N/A	500 IOPS/GB	50 IOPS/GB
Max Throughput / Instance	7,500 MB/s	7,500 MB/s	4,750 MB/s	7,500 MB/s
Latency	single digit	single digit	single digit	single digit

	millisecond	millisecond	millisecond	millisecond
Multi-Attach	No	No	Yes	Yes

Volume Name	Throughput Optimized HDD	Cold HDD
Volume type	st1	sc1
Description	Low cost HDD volume designed for frequently accessed, throughput-intensive workloads	Throughput-oriented storage for data that is infrequently accessed Scenarios where the lowest storage cost is important
Use Cases	Big data, data warehouses, log processing	Colder data requiring fewer scans per day
Volume Size	125 GB – 16 TB	125 GB – 16 TB
Durability	99.8% - 99.9% durability	99.8% - 99.9% durability
Max IOPS / Volume	500	250
Max Throughput / Volume	500 MB/s	250 MB/s
Max IOPS / Instance	260,000	260,000
Max IOPS / GB	N/A	N/A
Max Throughput / Instance	7,500 MB/s	7,500 MB/s
Multi-Attach	No	No

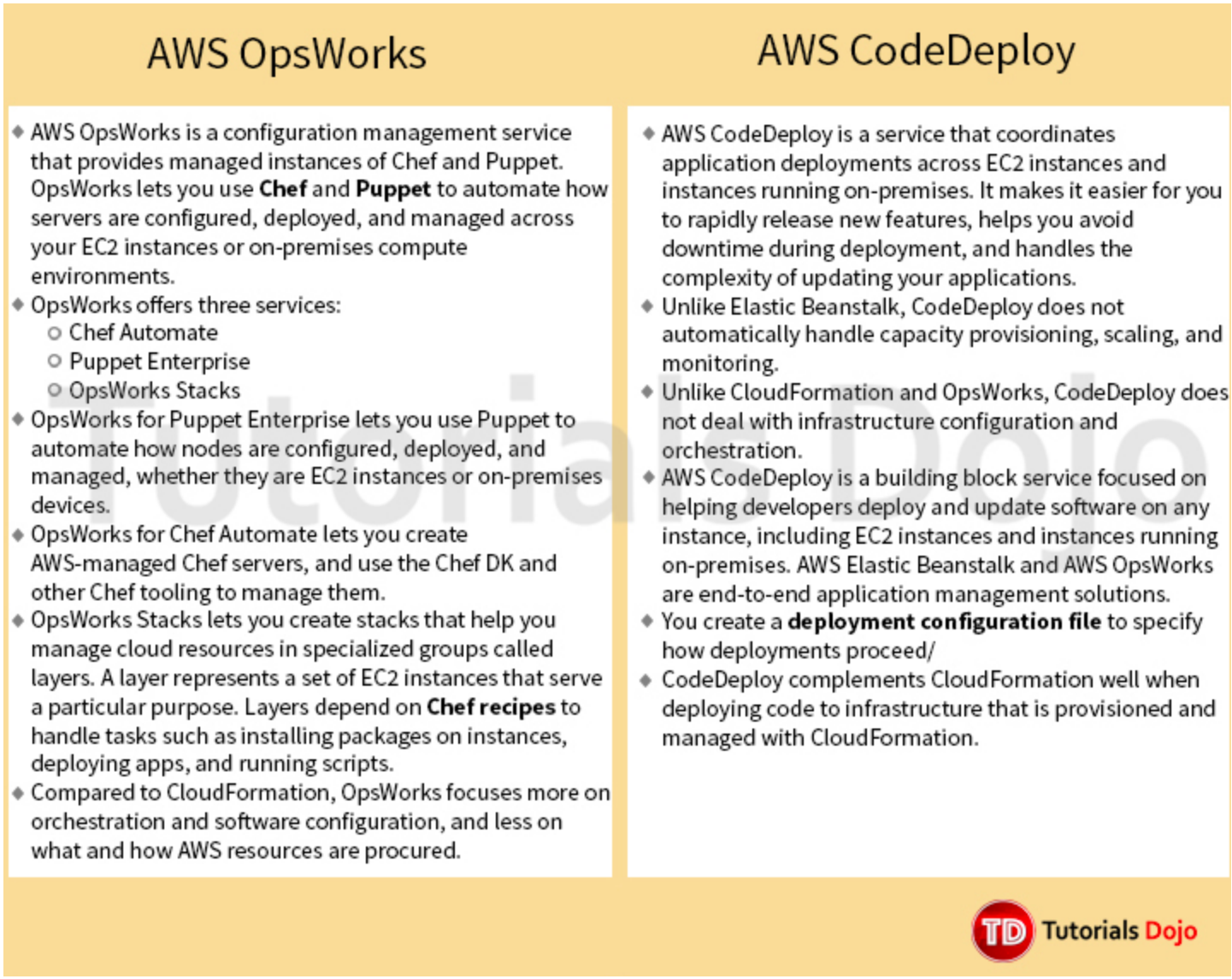
Elastic Beanstalk vs CloudFormation vs OpsWorks vs CodeDeploy

AWS Elastic Beanstalk

- AWS Elastic Beanstalk makes it even easier for developers to **quickly deploy and manage applications** in the AWS Cloud. Developers simply upload their application, and Elastic Beanstalk **automatically handles the deployment details** of capacity provisioning, load balancing, auto-scaling, and application health monitoring.
- This **platform-as-a-service solution** is typically for those who want to deploy and manage their applications within minutes in the AWS Cloud without worrying about the underlying infrastructure.
- AWS Elastic Beanstalk supports the following languages and development stacks:
  - Apache Tomcat for Java applications
  - Apache HTTP Server for PHP applications
  - Apache HTTP Server for Python applications
  - Nginx or Apache HTTP Server for Node.js applications
  - Passenger or Puma for Ruby applications
  - Microsoft IIS for .NET applications
  - Java SE
  - Docker
  - Go
- Elastic Beanstalk also supports deployment versioning. It maintains a copy of older deployments so that it is easy for the developer to rollback any changes made on the application.

AWS CloudFormation

- AWS CloudFormation is a service that gives developers and businesses an easy way to create a **collection of related AWS resources** and provision them in an orderly and predictable fashion. This is typically known as **"infrastructure as code".**
- The main difference between CloudFormation and Elastic Beanstalk is that CloudFormation deals more with the AWS infrastructure rather than applications. AWS CloudFormation introduces two concepts:
  - The **template**, a JSON or YAML-format, text-based file that describes all the AWS resources and configurations you need to deploy to run your application.
  - The **stack**, which is the set of AWS resources that are created and managed as a single unit when AWS CloudFormation instantiates a template.
- CloudFormation also supports a rollback feature through template version controls. When you try to update your stack but the deployment failed midway,
- CloudFormation will automatically revert the changes back to their previous working states.
- CloudFormation supports Elastic Beanstalk application environments. This allows you, for example, to create and manage an AWS Elastic Beanstalk–hosted application along with an RDS database to store the application data.
- AWS CloudFormation can be used to bootstrap both Chef (Server and Client) and Puppet (Master and Client) softwares on your EC2 instances.
- CloudFormation also supports OpsWorks. You can now model OpsWorks components (stacks, layers, instances, and applications) inside CloudFormation templates, and provision them as CloudFormation stacks. This enables you to document, version control, and share your OpsWorks configuration.
- AWS CodeDeploy is a recommended adjunct to CloudFormation for managing the application deployments and updates.



Additional Notes:

- Elastic Beanstalk, CloudFormation, or OpsWorks are particularly useful for **blue-green deployment method** as they provide a simple way to clone your running application stack.
- CloudFormation and OpsWorks are best suited for the **prebaking AMIs**.
- CodeDeploy and OpsWorks are best suited for performing **in-place application upgrades**. For **disposable upgrades**, you can set up a cloned environment with Elastic Beanstalk, CloudFormation, and OpsWorks.

Global Secondary Index vs Local Secondary Index

A *secondary index* is a data structure that contains a subset of attributes from a table, along with an alternate key to support Query operations. An Amazon DynamoDB table can have multiple secondary indexes.

	Global secondary index	Local secondary index
Definition	An index with a partition key and a sort key that can be different from those on the base table.	An index that has the same partition key as the base table, but a different sort key.
Span of query	Queries on the index can span all of the data in the base table, across all partitions.	Every partition of a local secondary index is scoped to a base table partition that has the same partition key value.
Primary Key Schema	The partition key and, optionally, the sort key	Must be both partition key and sort key
Partition Key Attributes	Any base table attribute of type string, number, or binary.	Must have the same attribute as the partition key of the base table.
Sort Key Attributes	Any base table attribute of type string, number, or binary.	Any base table attribute of type string, number, or binary.
Key Values	Do not need to be unique	The sort key value does not need to be unique for a given partition key value.
Size Restrictions Per Partition Key Value	No restriction.	For each partition key value, the total size of all indexed items must be 10 GB or less.
Index Operations	• Can be created during creation of a table. • Can be created on an existing table. • Can be deleted from an existing table.	• Can be created during creation of a table.
Read Consistency for Queries	Supports eventual consistency only from asynchronous updates and deletes	You can choose either eventual consistency or strong consistency.
Provisioned Throughput Consumption	Every global secondary index has its own provisioned throughput settings for read and write activity that you need to	Queries or scans on a local secondary index consume read capacity units from the base table.

	specify. Queries or scans on a global secondary index consume capacity units from the index, not from the base table. This is also the case for global secondary index updates due to table writes.	When you write to a table and its local secondary indexes are updated, these updates consume write capacity units from the base table.
Projected Attributes	With global secondary index queries or scans, you can only request the attributes that are projected into the index.	If you query or scan a local secondary index, you can request attributes that are not projected in to the index. DynamoDB will automatically fetch those attributes from the table.
Index limit (default)	20 indexes	5 indexes per table

Global Secondary Index Read/Write Capacity Calculation (Provisioned Throughput Mode)

- **Eventually consistent reads consume ½ read capacity unit.** Therefore, each query can retrieve up to 8KB of data per capacity unit (4KB x 2). The maximum size of the results returned by a Query operation is 1 MB.
- **The total provisioned throughput cost for a write consists of the sum of write capacity units consumed by writing to the base table and those consumed by updating the global secondary indexes.** If a write to a table does not require a global secondary index update, then no write capacity is consumed from the index.
- Cost of a write operation depends on the ff factors (assuming up to 1 KB item size):
    - If you write a new item to the table that defines an indexed attribute, or you update an existing item to define a previously undefined indexed attribute, one write operation is required to put the item into the index.
    - If an update to the table changes the value of an indexed key attribute, two writes are required, one to delete the previous item from the index and another write to put the new item into the index.
    - If an item was present in the index, but a write to the table caused the indexed attribute to be deleted, one write is required to delete the old item projection from the index.
    - If an item is not present in the index before or after the item is updated, there is no additional write cost for the index.
    - If an update to the table only changes the value of projected attributes in the index key schema, but does not change the value of any indexed key attribute, then one write is required to update the values of the projected attributes into the index.

Local Secondary Index Read/Write Capacity Calculation (Provisioned Throughput Mode)

- An index query can use either eventually consistent or strongly consistent reads. **One strongly consistent read consumes one read capacity unit; an eventually consistent read consumes only half**

of that. The number of read capacity units is the sum of all projected attribute sizes across all of the items returned (from index and base table), and the result is then rounded up to the next 4 KB multiple. The maximum size of the results returned by a Query operation is 1 MB.

- **The total provisioned throughput cost for a write is the sum of write capacity units consumed by writing to the table and those consumed by updating the local secondary indexes.**
- Cost of a write operation depends on the ff factors (assuming up to 1 KB item size):
    - If you write a new item to the table that defines an indexed attribute, or you update an existing item to define a previously undefined indexed attribute, one write operation is required to put the item into the index.
    - If an update to the table changes the value of an indexed key attribute, two writes are required, one to delete the previous item from the index and another write to put the new item into the index.
    - If an item was present in the index, but a write to the table caused the indexed attribute to be deleted, one write is required to delete the old item projection from the index.
    - If an item is not present in the index before or after the item is updated, there is no additional write cost for the index.

S3 Pre-Signed URLS vs CloudFront Signed URLS vs Origin Access Identity

S3 Pre-signed URLs	CloudFront Signed URLs	Origin Access Identity (OAI)
All S3 buckets and objects by default are **private**. Only the object owner has permission to access these objects. Pre-signed URLs use the owner's security credentials to grant others time-limited permission to download or upload objects.	You can control user access to your private content in two ways • Restrict access to files in CloudFront edge caches • Restrict access to files in your Amazon S3 bucket (unless you've configured it as a website endpoint)	You can configure an S3 bucket as the origin of a CloudFront distribution. OAI prevents users from viewing your S3 files by simply using the direct URL for the file. Instead, they would need to access it through a CloudFront URL.
When creating a pre-signed URL, you (as the owner) need to provide the following: • Your security credentials • An S3 bucket name • An object key • Specify the HTTP method (GET to download the object or PUT to upload an object) • Expiration date and time of the URL.	You can configure CloudFront to require that users access your files using either **signed URLs** or **signed cookies**. You then develop your application either to create and distribute signed URLs to authenticated users or to send Set-Cookie headers that set signed cookies on the viewers for authenticated users. When you create signed URLs or signed cookies to control access to your files, you can specify the following restrictions: • An expiration date and time for the URL • (Optional) The date and time the URL becomes valid • (Optional) The IP address or range of addresses of the computers that can be used to access your content You can use signed URLs or signed cookies for any CloudFront distribution, regardless of whether the origin is an Amazon S3 bucket or an HTTP server.	To require that users access your content through CloudFront URLs, you perform the following tasks: • Create a special CloudFront user called an **origin access identity**. • Give the origin access identity permission to read the files in your bucket. • Remove permission for anyone else to use Amazon S3 URLs to read the files (through bucket policies or ACLs). You cannot set OAI if your S3 bucket is configured as a website endpoint.

CloudWatch Agent vs SSM Agent vs Custom Daemon Scripts

CloudWatch Agent	SSM Agent (AWS Systems Manager)	Custom Daemon Scripts
CloudWatch agent allows you to collect more system-level metrics from your EC2 and on-premises servers than just the standard CloudWatch metrics. It also enables you to retrieve custom metrics from your applications or services using the *StatsD* and *collectd* protocols. *StatsD* is supported on both Linux servers and servers running Windows Server. *collectd* is supported only on Linux servers. You can use CloudWatch agent to collect logs from your servers and send them to CloudWatch Logs. Metrics collected by the CloudWatch agent are billed as custom metrics. You can install CloudWatch Agent using three ways: • via Command Line • via SSM Agent • via AWS CloudFormation	SSM Agent is Amazon software that runs on your EC2 instances and your hybrid instances that are configured for Systems Manager. SSM Agent processes requests from the Systems Manager service in the cloud and configures your machine as specified in the request. You can manage servers without having to log in to them using automation. SSM Agent sends status and execution information back to the Systems Manager service by using the *EC2 Messaging* service. SSM Agent runs on Amazon EC2 instances using root permissions (Linux) or SYSTEM permissions (Windows). CloudWatch agent replaces SSM agent in sending metric logs to CloudWatch Logs.	You use custom scripts (such as cron or bash scripts) if the two previously mentioned agents do not fit your needs. CloudWatch agent is useful for collecting system-level metrics and logs. You can create custom scripts that perform some modifications before the metrics are sent out. SSM Agent is also useful for automation purposes, though Systems Manager does not have a document for every case scenario. You may also have some compliance requirements that would require SSM Agent to be disabled (recall that SSM agent runs at root level permissions).

Amazon SWF vs AWS Step Function vs Amazon SQS

Amazon Simple Workflow (SWF)

- A web service that makes it easy to coordinate work across distributed application components.
- In Amazon SWF, **tasks** represent invocations of logical steps in applications. Tasks are processed by **workers** which are programs that interact with Amazon SWF to get tasks, process them, and return their results.
- The coordination of tasks involves managing execution dependencies, scheduling, and concurrency in accordance with the logical flow of the application.

AWS Step Functions

- A fully managed service that makes it easy to coordinate the components of distributed applications and microservices using **visual workflows**.
- You define **state machines** that describe your workflow as a series of steps, their relationships, and their inputs and outputs. State machines contain a number of **states**, each of which represents an individual step in a workflow diagram. States can perform work, make choices, pass parameters, initiate parallel execution, manage timeouts, or terminate your workflow with a success or failure.

Amazon SQS

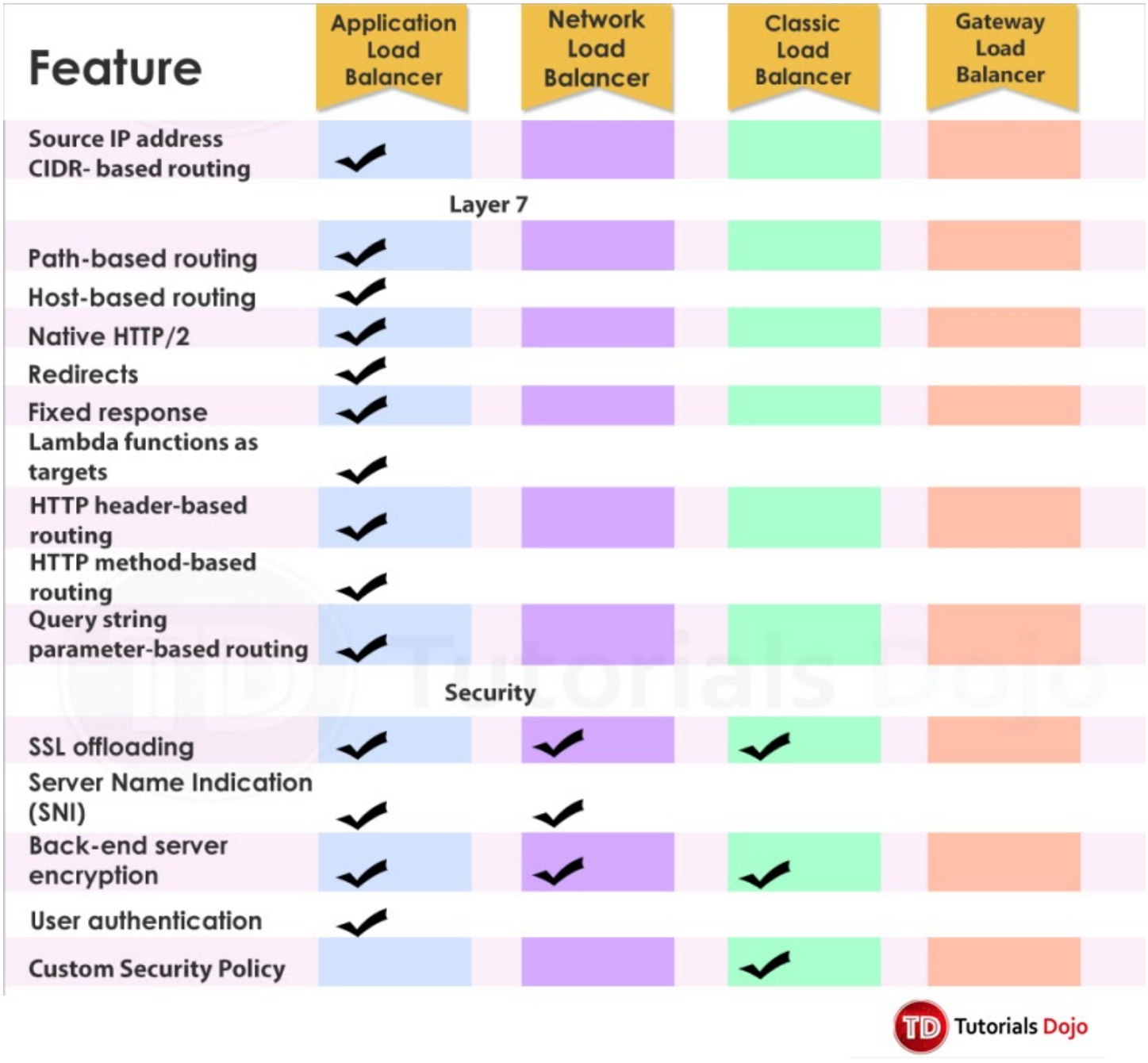
- A message queue service used by distributed applications to exchange messages through a polling model, and can be used to decouple sending and receiving components.
- FIFO (first-in-first-out) queues preserve the exact order in which messages are sent and received. Standard queues provide a loose-FIFO capability that attempts to preserve the order of messages.

Amazon SWF vs AWS Step Functions	AWS Step Functions vs Amazon SQS	Amazon SQS vs AWS SWF
Consider using AWS Step Functions for all your new applications, since it provides a more productive and agile approach to coordinating application components **using visual workflows**. If you require **external signals** (deciders) to intervene in your processes, or you would like to launch child processes that return a result to a parent, then you should consider Amazon SWF.	Use Step Functions when you need to coordinate service components in the development of highly scalable and auditable **applications**. Use SQS when you need a reliable, highly scalable, hosted **queue** for sending, storing, and receiving **messages** between services.	SWF API actions are **task-oriented**. SQS API actions are **message-oriented**. The SWF Console and visibility APIs provide an **application-centric view** that lets you search for executions, drill down into an execution's details, and administer executions. SQS requires **implementing** such additional functionality.

With Step Functions, you write state machines in **declarative JSON**. With Amazon SWF, you write a **decider program** to separate activity steps from decision steps. This provides you complete control over your orchestration logic, but increases the complexity of developing applications. You may write decider programs in the programming language of your choice, or you may use the Flow framework, which is a library for building SWF applications, to use programming constructs that structure asynchronous interactions for you.	Step Functions **keeps track** of all tasks and events in an application. Amazon SQS requires you to **implement your own** application-level tracking, especially if your application uses multiple queues. The Step Functions Console and visibility APIs provide an **application-centric view** that lets you search for executions, drill down into an execution's details, and administer executions. Amazon SQS requires **implementing** such additional functionality.	SWF **keeps track** of all tasks and events in an application. SQS requires you to **implement your own** application-level tracking, especially if your application uses multiple queues.
	Step Functions offers several **features that facilitate application development**, such as passing data between tasks and flexibility in distributing tasks. Amazon SQS requires you to **implement** some application-level functionality.	SWF offers several **features that facilitate application development**, such as passing data between tasks, signaling, and flexibility in distributing tasks. SQS requires you to **implement** some application-level functionality.
	You can use Amazon SQS to build basic workflows to coordinate your distributed application, but you get this facility out-of-the-box with Step Functions, alongside other application-level capabilities.	In addition to a core SDK that calls service APIs, SWF provides the **Flow Framework** with which you can write distributed applications using programming constructs that structure asynchronous interactions.

Application Load Balancer vs Network Load Balancer vs Classic Load Balancer vs Gateway Load Balancer

Feature	Application Load Balancer	Network Load Balancer	Classic Load Balancer	Gateway Load Balancer
Protocols	HTTP HTTPS	TCP, UDP, TLS	TCP, SSL/TLS, HTTP, HTTPS	IP
Platforms	VPC	VPC	EC2-Classic, VPC	VPC
Healthchecks	✓	✓	✓	✓
Cloudwatch Metrics	✓	✓	✓	✓
Logging	✓	✓	✓	✓
Zonal Failover	✓	✓	✓	✓
Connection Draining (deregistration delay)	✓		✓	
Load Balancing to multiple ports on the same instance	✓	✓		✓
IP addresses as targets	✓	✓ (TCP, TLS)		✓
Load balancer deletion protection	✓	✓		✓
Configurable idle connection timeout	✓		✓	
Cross-zone load balancing	✓	✓	✓	✓
Sticky sessions	✓	✓	✓	✓
Static IP		✓		
Elastic IP address		✓		
Preserve Source IP address		✓		✓
Resource-based IAM permissions	✓	✓	✓	✓
Tag-based IAM permissions	✓	✓		✓
Slow start	✓			
Web sockets	✓	✓		✓
PrivateLink Support		✓ (TCP, TLS)		✓ (GWLBE)



Feature	Application Load Balancer	Network Load Balancer	Classic Load Balancer	Gateway Load Balancer
Source IP address CIDR- based routing	✓			
Layer 7				
Path-based routing	✓			
Host-based routing	✓			
Native HTTP/2	✓			
Redirects	✓			
Fixed response	✓			
Lambda functions as targets	✓			
HTTP header-based routing	✓			
HTTP method-based routing	✓			
Query string parameter-based routing	✓			
Security				
SSL offloading	✓	✓	✓	
Server Name Indication (SNI)	✓	✓		
Back-end server encryption	✓	✓	✓	
User authentication	✓			
Custom Security Policy			✓	

Common features between the load balancers:

- Has instance health check features
- Has built-in CloudWatch monitoring
- Logging features
- Support zonal failover

- Support cross-zone load balancing (evenly distributes traffic across registered instances in enabled AZs)
- Resource-based IAM permission policies
- Tag-based IAM permissions
- Flow stickiness - all packets are sent to one target and return the traffic that comes from the same target.

S3 Transfer Acceleration vs Direct Connect vs VPN vs Snowball vs Snowmobile

S3 Transfer Acceleration (TA)

- Amazon S3 Transfer Acceleration makes public Internet transfers to S3 faster, as it leverages Amazon CloudFront's globally distributed AWS Edge Locations.
- There is no guarantee that you will experience increased transfer speeds. If S3 Transfer Acceleration is not likely to be faster than a regular S3 transfer of the same object to the same destination AWS Region, AWS will not charge for the use of S3 TA for that transfer.
- This is not the best transfer service to use if transfer disruption is not tolerable.
- S3 TA provides the same security benefits as regular transfers to Amazon S3. This service also supports multi-part upload.
- ***S3 TA vs AWS Snow****
    - The AWS Snow* Migration Services are ideal for moving large batches of data at once. In general, if it will take more than a week to transfer over the Internet, or there are recurring transfer jobs and there is more than 25Mbps of available bandwidth, S3 Transfer Acceleration is a good option.
    - Another option is to use AWS Snowball or Snowmobile to perform initial heavy lift moves and then transfer incremental ongoing changes with S3 Transfer Acceleration.
- ***S3 TA vs Direct Connect***
    - AWS Direct Connect is a good choice for customers who have a private networking requirement or who have access to AWS Direct Connect exchanges. S3 Transfer Acceleration is best for submitting data from distributed client locations over the public Internet, or where variable network conditions make throughput poor.
- ***S3 TA vs VPN***
    - You typically use (IPsec) VPN if you want your resources contained in a private network. VPN tools such as OpenVPN allow you to setup stricter access controls if you have a private S3 bucket. You can complement this further with the increased speeds from S3 TA.
- ***S3 TA vs Multipart Upload***
    - Use multipart upload if you are uploading large files and you want to handle failed uploads gracefully. With multipart upload, each part of your upload is a contiguous portion of the object's data. You can upload these object parts independently and in any order. If transmission of any part fails, you can retransmit that part without affecting other parts.
    - For S3 TA, as the name implies, accelerates your transfer speeds, not just for upload but also for download speed. There is no reason why you can't use S3 TA and multipart upload together, but if you are only handling small files, using multipart upload is not necessary.

AWS Direct Connect

- Using AWS Direct Connect, data that would have previously been transported over the Internet can now be delivered through a **private physical network connection** between AWS and your datacenter or corporate network. Customers' traffic will remain in AWS global network backbone, after it enters AWS global network backbone.
- Benefits of Direct Connect vs internet-based connections
    - reduced costs
    - increased bandwidth

    - a more consistent network experience
- Each AWS Direct Connect connection can be configured with one or more **virtual interfaces**. Virtual interfaces may be configured to access AWS services such as Amazon EC2 and Amazon S3 using public IP space, or resources in a VPC using private IP space.
- You can run IPv4 and IPv6 on the same virtual interface.
- Direct Connect does not support multicast.
- A Direct Connect connection is **not redundant**. Therefore, a second line needs to be established if redundancy is required. Enable *Bidirectional Forwarding Detection* (BFD) when configuring your connections to ensure fast detection and failover.
- AWS Direct Connect offers SLA.
- Direct Connect vs IPsec VPN
    - A VPC VPN Connection utilizes IPSec to establish **encrypted network connectivity** between your intranet and Amazon VPC **over the Internet.** VPN Connections can be configured in minutes and are a good solution if you have an immediate need, have low to modest bandwidth requirements, and can tolerate the inherent variability in Internet-based connectivity. AWS Direct Connect **does not involve the Internet**; instead, it uses **dedicated, private network connections** between your intranet and Amazon VPC.
- You can combine one or more Direct Connect dedicated network connections with the Amazon VPC VPN. This combination provides an IPsec-encrypted private connection that also includes the benefits of Direct Connect.

AWS VPN

- AWS VPN is comprised of two services:
    - AWS Site-to-Site VPN enables you to securely connect your on-premises network or branch office site to your Amazon VPC.
    - AWS Client VPN enables you to securely connect users to AWS or on-premises networks.
- Data transferred between your VPC and datacenter routes over an encrypted VPN connection to help maintain the confidentiality and integrity of data in transit.
- If data that passes through Direct Connect moves in a dedicated private network line, AWS VPN instead encrypts the data before passing it through the Internet.
- VPN connection throughput can depend on multiple factors, such as the capability of your customer gateway, the capacity of your connection, average packet size, the protocol being used, TCP vs. UDP, and the network latency between your customer gateway and the virtual private gateway.
- All the VPN sessions are **full-tunnel VPN**. (cannot split tunnel)
- AWS Site-to-Site VPN enable you to create **failover** and CloudHub solutions **with AWS Direct Connect**.
- AWS Client VPN is designed to connect devices to your applications. It allows you to choose from **OpenVPN-based client**.

Snowball

- Snowball is a **petabyte-scale data transport** solution that uses secure appliances to transfer large amounts of data into and out of AWS.
- Benefits of Snowball include:
    - lower network costs,
    - Shorter transfer times,

    - and security using 256-bit encryption keys you manage through AWS Key Management Service (KMS)..
- Similar to Direct Connect, AWS Snowball is **physical hardware**. It includes a 10GBaseT network connection. You can order a device with either **50TB** or an **80TB** storage capacity.
- Data transported via Snowball are stored in Amazon S3 once the device arrives at AWS centers.
- AWS Snowball is not only for shipping data into AWS, but also out of AWS.
- AWS Snowball can be used as a quick order for additional temporary petabyte storage.
- For security purposes, data transfers must be completed **within 90 days of a Snowball's preparation**.
- When the transfer is complete and the device is ready to be returned, the E Ink shipping label will automatically update to indicate the correct AWS facility to ship to, and you can track the job status by using Amazon Simple Notification Service (SNS), text messages, or directly in the console.
- Snowball is the best choice if you need to more securely and quickly transfer terabytes to many petabytes of data to AWS. Snowball can also be the right choice if you don't want to make expensive upgrades to your network infrastructure, if you frequently experience large backlogs of data, if you're located in a physically isolated environment, or if you're in an area where high-bandwidth Internet connections are not available or cost-prohibitive.
- If you will be transferring data to AWS on an ongoing basis, it is better to use AWS Direct Connect.
- If multiple users located in different locations are interacting with S3 continuously, it is better to use S3 TA.
- You **cannot** export data directly from S3 Glacier. It should be first restored to S3.

Snowmobile

- Snowmobile is Snowball with larger storage capacity. Snowmobile is literally a mobile truck.
- Snowmobile is an **Exabyte-scale data transfer** service.
- You can transfer up to **100PB** per Snowmobile.
- Snowmobile uses multiple layers of security to help protect your data including dedicated security personnel, GPS tracking, alarm monitoring, 24/7 video surveillance, and an optional escort security vehicle while in transit. All data is encrypted with 256-bit encryption keys you manage through the AWS Key Management Service (KMS).
- After the data transfer is complete, the Snowmobile will be returned to your designated AWS region where your data will be uploaded into the AWS storage services such as S3 or Glacier.
- Snowball vs Snowmobile
    - To migrate large datasets of 10PB or more in a single location, you should use Snowmobile. For datasets less than 10PB or distributed in multiple locations, you should use Snowball.
    - If you have a high speed backbone with hundreds of Gb/s of spare throughput, then you can use Snowmobile to migrate the large datasets all at once. If you have limited bandwidth on your backbone, you should consider using multiple Snowballs to migrate the data incrementally.
    - Snowmobile **does not** support data export. Use Snowball/Snowball Edge for this cause.
- When the data import has been processed and verified, AWS performs a software erasure based on NIST guidelines.

EC2 Container Services ECS vs Lambda

Amazon EC2 Container Service (ECS)

- Amazon ECS is a highly scalable, high performance container management service that supports Docker containers and allows you to easily run applications on a managed cluster of Amazon EC2 instances. ECS eliminates the need for you to install, operate, and scale your own cluster management infrastructure.
- With ECS, deploying containerized applications is easily accomplished. This service fits well in running batch jobs or in a microservice architecture.
  You have a central repository where you can upload your Docker Images from ECS container for safekeeping called Amazon ECR.
- Applications in ECS can be written in a stateful or stateless matter.
- The Amazon ECS CLI supports Docker Compose, which allows you to simplify your local development experience as well as easily set up and run your containers on Amazon ECS.
- Since your applications still run on EC2 instances, server management is your responsibility. This gives you more granular control over your system.
- It is up to you to manage scaling and load balancing of your EC2 instances as well, unlike in AWS Lambda where functions scale automatically.
- You are charged for the costs incurred by your EC2 instances in your clusters. Most of the time, Amazon ECS costs more than using AWS Lambda since your active EC2 instances will be charged by the hour.
- One version of Amazon ECS, know as AWS Fargate, will fully manage your infrastructure so you can just focus on deploying containers. AWS Fargate has a different pricing model from the standard EC2 cluster.
- ECS will automatically recover unhealth containers to ensure that you have the desired number of containers supporting your application.

Tutorials Dojo

AWS Lambda

- AWS Lambda is a function-as-a-service offering that runs your code in response to events and automatically manages the compute resources for you, since Lambda is a serverless compute service. With Lambda, you do not have to worry about managing servers, and directly focus on your application code.
- Lambda automatically scales your function to meet demands. It is noteworthy, however, that Lambda has a maximum execution duration per request of 900 seconds or 15 minutes.
- To allow your Lambda function to access other services such as Cloudwatch Logs, you would need to create an execution role that has the necessary permissions to do so.
- You can easily integrate your function with different services such as API Gateway, DynamoDB, CloudFront, etc. using the Lambda console.
- You can test your function code locally in the Lambda console before launching it into production. Currently, Lambda supports only a number of programming languages such as Java, Go, PowerShell, Node.js, C#, Python, and Ruby. ECS is not limited by programming languages since it mainly caters to Docker.
- Lambda functions must be stateless since you do not have volumes for data storage.
- You are charged based on the number of requests for your functions and the duration, the time it takes for your code to execute. To minimize costs, you can throttle the number of concurrent executions running at a time, and the execution time limit of the function.
- With Lambda@Edge, AWS Lambda can run your code across AWS locations globally in response to Amazon CloudFront events, such as requests for content to or from origin servers and viewers.
  This makes it easier to deliver content to end users with lower latency.

SNI Custom SSL vs Dedicated IP Custom SSL

Server Name Indication (SNI) Custom SSL	Dedicated IP Custom SSL
♦ Relies on the SNI extension of the TLS protocol, which allows multiple domains to serve SSL traffic over the same IP address. ♦ Offers the same level of security when using Dedicated IP Custom SSL. ♦ If you configure CloudFront to serve HTTPS requests using SNI, CloudFront associates your alternate domain name with an IP address for each edge location. The IP address to your domain name is determined during the SSL/TLS handshake negotiation, and isn't dedicated to your distribution. ♦ Some older browsers do not support SNI and will not be able to establish a connection with CloudFront to load the HTTPS version of your content. ♦ You can use SNI Custom SSL with no upfront or monthly fees for certificate management.	♦ Mainly useful for browsers that do not support SNI. ♦ For this feature, the Amazon content delivery network allocates dedicated IP addresses to serve your SSL content at each Edge location. ♦ You will need to upload a SSL certificate and associate it with your CloudFront distributions. ♦ You can associate more than two custom SSL certificate with your AWS Account by submitting a CloudFront Limit Increase Form. ♦ This method works for every HTTPS request, regardless of the browser or other viewer that the user is using. ♦ Because of the added cost associated with dedicating IP addresses per SSL certificate, AWS charges a fixed monthly fee of $600 for each custom SSL certificate you associate with your content delivery network distributions, pro-rated by the hour. ♦ You can switch to using a custom SSL/TLS certificate with SNI instead and eliminate the charge that is associated with dedicated IP addresses.

EC2 Instance Health Check vs ELB Health Check vs Auto Scaling and Custom Health Check

EC2 instance health check

- Amazon EC2 performs automated checks on every running EC2 instance to identify hardware and software issues.
- Status checks are performed every minute and each returns a pass or a fail status.
  - If all checks pass, the overall status of the instance is OK.
  - If one or more checks fail, the overall status is impaired.
- Status checks are built into EC2, so they cannot be disabled or deleted.
- You can create or delete alarms that are triggered based on the result of the status checks.
- There are two types of status checks
  - **System Status Checks**
    - These checks detect underlying problems with your instance that require AWS involvement to repair. When a system status check fails, you can choose to wait for AWS to fix the issue, or you can resolve it yourself.
  - **Instance Status Checks**
    - Monitor the software and network configuration of your individual instance. Amazon EC2 checks the health of an instance by sending an address resolution protocol (ARP) request to the ENI. These checks detect problems that require your involvement to repair.

Tutorials Dojo

Elastic Load Balancer (ELB) health check

- To discover the availability of your registered EC2 instances, a load balancer periodically sends pings, attempts connections, or sends requests to test the EC2 instances.
- The status of the instances that are healthy at the time of the health check is InService. The status of any instances that are unhealthy at the time of the health check is OutOfService.
- When configuring a health check, you would need to provide the following:
  - a specific port
  - protocol to use
    - HTTP/HTTPS health check succeeds if the instance returns a 200 response code within the health check interval.
    - A TCP health check succeeds if the TCP connection succeeds.
    - An SSL health check succeeds if the SSL handshake succeeds.
  - ping path
- ELB health checks do not support WebSockets.
- The load balancer routes requests only to the healthy instances. When an instance becomes impaired, the load balancer resumes routing requests to the instance only when it has been restored to a healthy state.
- The load balancer checks the health of the registered instances using either
  - the default health check configuration provided by Elastic Load Balancing or
  - a health check configuration that you configure (auto scaling or custom health checks for example).
- Network Load Balancers use active and passive health checks to determine whether a target is available to handle requests.
  - With **active health checks**, the load balancer periodically sends a request to each registered target to check its status. After each health check is completed, the load balancer node closes the connection that was established.
  - With **passive health checks**, the load balancer observes how targets respond to connections, which enables it to detect an unhealthy target before it is reported as unhealthy by active health checks. You cannot disable, configure, or monitor passive health checks.

Auto Scaling and Custom health checks

- All instances in your Auto Scaling group start in the healthy state. Instances are assumed to be healthy unless EC2 Auto Scaling receives notification that they are unhealthy. This notification can come from one or more of the following sources:
  - Amazon EC2 (default)
  - Elastic Load Balancing
  - A custom health check.
- After Amazon EC2 Auto Scaling marks an instance as unhealthy, it is scheduled for replacement. If you do not want instances to be replaced, you can suspend the health check process for any individual Auto Scaling group.
- If an instance is in any state other than running or if the system status is impaired, Amazon EC2 Auto Scaling considers the instance to be unhealthy and launches a replacement instance.
- If you attached a load balancer or target group to your Auto Scaling group, Amazon EC2 Auto Scaling determines the health status of the instances by checking both the EC2 status checks and the Elastic Load Balancing health checks.
- Amazon EC2 Auto Scaling waits until the health check grace period ends before checking the health status of the instance. Ensure that the health check grace period covers the expected startup time for your application.
- Health check grace period does not start until lifecycle hook actions are completed and the instance enters the InService state.
- With custom health checks, you can send an instance's health information directly from your system to Amazon EC2 Auto Scaling.

Multi-AZ deployments vs. Multi-Region deployments vs. Read Replicas

Multi-AZ deployments	Multi-Region deployments	Read Replicas
Main purpose is high availability	Main purpose is disaster recovery and local performance	Main purpose is scalability
Non-Aurora: synchronous replication; Aurora: asynchronous replication	Asynchronous replication	Asynchronous replication
Non-Aurora: only the primary instance is active; Aurora: all instances are active	All regions are accessible and can be used for reads	All read replicas are accessible and can be used for readscaling
Non-Aurora: automated backups are taken from standby; Aurora: automated backups are taken from shared storage layer	Automated backups can be taken in each region	No backups configured by default
Always span at least two Availability Zones within a single region	Each region can have a Multi-AZ deployment	Can be within an Availability Zone, Cross-AZ, or Cross-Region
Non-Aurora: database engine version upgrades happen on primary; Aurora: all instances are updated together	Non-Aurora: database engine version upgrade is independent in each region; Aurora: all instances are updated together	Non-Aurora: database engine version upgrade is independent from source instance; Aurora: all instances are updated together
Automatic failover to standby (non-Aurora) or read replica (Aurora) when a problem is detected	Aurora allows promotion of a secondary region to be the master	Can be manually promoted to a standalone database instance (non-Aurora) or to be the primary instance (Aurora)

FINAL REMARKS AND TIPS

Thanks to the innovations brought by the cloud, developers such as yourself can work with less inconvenience and more flexibility. It is an exciting time to be in the development field, not only for experienced developers, but also for those new to the space. We do believe that there are practices in software development that should continue to be applied even after moving from an on-premises environment to AWS. Although AWS shares what they believe are the best practices for developing applications in their environment, the main benefit actually of the cloud is having the freedom to implement what you think is best for your applications. Additionally, there is less worry with experimentations because resources in AWS are replaceable and should be treated as such. This gives you the opportunity to figure out what works best for your workloads and implement it in a cost-effective manner.

Hopefully our cheat sheets have helped you a lot in your review for the AWS Certified Developer Associate exam. This AWS certificate is sought after by developers all over the world, since it proves that they have the knowledge and skills to develop applications properly in AWS. Passing an AWS certification exam is no easy feat and the AWS community celebrates exam passers with pride and joy. We at Tutorials Dojo are dedicated to help you achieve these results too. We create our content specifically to help you prepare for your exams and equip you with the important information so you can be successful in your role. We have written blogs, guides, cheat sheets, and practice exams that are also constantly being updated based on our experiences and on the feedback of our students. Your feedback is very important to us since it helps us improve our content and serve the community better.

And with that, we at Tutorials Dojo thank you for supporting us through this eBook. If you wish to validate what you have learned so far, now is a great time to check out our AWS Certified Developer Associate Practice Exam as well. You can also try the free sampler version of our full practice test course here. It will fill in the gaps in your knowledge that you are not aware of, and will give you a sense of the actual exam environment. That way, you'll know what to expect in the actual exam and you can pace yourself through the questions better. If you have any issues, concerns, or constructive feedback on our eBook, feel free to contact us at support@tutorialsdojo.com.

Goodluck on your exam, and we'd love to hear back from you.

ABOUT THE AUTHORS

Jon Bonso (8x AWS Certified)

Born and raised in the Philippines, Jon is the Co-Founder of **Tutorials Dojo**. Now based in Sydney, Australia, he has over a decade of diversified experience in Banking, Financial Services, and Telecommunications. He's 8x AWS Certified and has worked with various cloud services such as Google Cloud, and Microsoft Azure. Jon is passionate about what he does and dedicates a lot of time creating educational courses. He has given IT seminars to different universities in the Philippines for free and has launched educational websites using his own money and without any external funding.

Adrian Formaran (3x AWS Certified)

As a Computer Scientist and a proud university scholar, Adrian has a passion for learning cutting edge technologies, such as blockchain, cloud services, and information security, and is passionate about teaching these to others as well. He currently has 3 AWS certifications under his belt, including the AWS Certified Solutions Architect Professional. He also has a deep love for mathematics, sciences, and philosophy. A gamer at heart.