AWS CHEAT SHEETS

AWS Compute Services

Amazon Elastic Compute Cloud (EC2)

- A Linux-based/Windows-based/Mac-based virtual server that you can provision.

Features

- Server environments called **instances**.
- Package OS and additional installations in a reusable template called **Amazon Machine Images**
- 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.**
- 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.

Instance states

- To prevent accidental termination, enable termination protection.

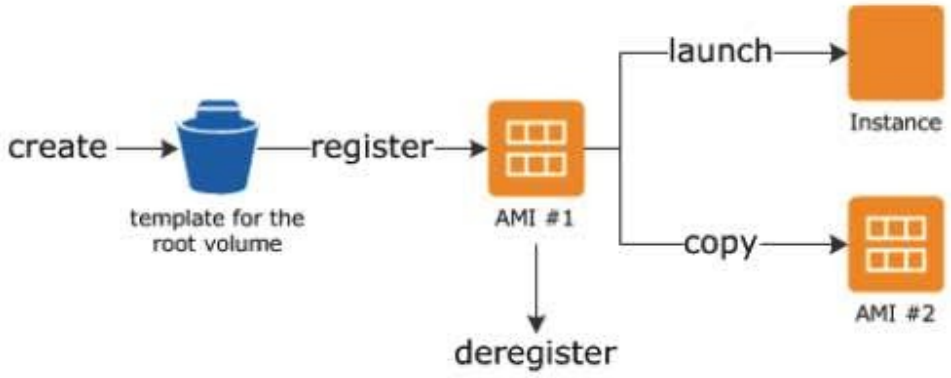
Root Device Volumes

- 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).

- Amazon EBS-backed Instances
    - An Amazon EBS-backed instance can be stopped and later restarted without affecting data stored in the attached volumes.
    - 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



- 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.
- 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.

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.
    - 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**
- 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 **network interface** is a logical networking component in a VPC that represents a virtual network card, which directs traffic to your instance
- Scale with **EC2 Scaling Groups** and distribute traffic among instances using **Elastic Load Balancer**.

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.
- View all categories of instance metadata from within a running instance at **http://169.254.169.254/latest/meta-data/**
- Retrieve user data from within a running instance at **http://169.254.169.254/latest/user-data**

Amazon Elastic Container Registry (ECR)

- A managed AWS Docker registry service. Stores your Docker images that you can use to deploy on your EC2, ECS, or Fargate deployments.

Features

- ECR supports Docker Registry HTTP API V2 allowing you to use Docker CLI commands or your preferred Docker tools in maintaining your existing development workflow.
- You can transfer your container images to and from Amazon ECR via HTTPS.

Components

- **Registry**
  - A registry is provided to each AWS account; you can create image repositories in your registry and store images in them.
  - The URL for your default registry is https://*aws_account_id*.dkr.ecr.*region*.amazonaws.com.
- **Authorization token**
  - Your Docker client needs to authenticate to ECR registries as an AWS user before it can push and pull images. The AWS CLI *get-login* command provides you with authentication credentials to pass to Docker.
- **Repository**
  - An image repository contains your Docker images.
  - **ECR lifecycle policies** enable you to specify the lifecycle management of images in a repository.
- **Repository policy**
  - You can control access to your repositories and the images within them with repository policies.
- **Image**
  - You can push and pull Docker images to your repositories. You can use these images locally on your development system, or you can use them in ECS task definitions.
  - You can replicate images in your private repositories across AWS regions.

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.

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.

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**.
- 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.
- 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.
  - 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.
- Clusters
  - When you run tasks using ECS, you place them in a **cluster**, which is a logical grouping of resources.
  - 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.
  - Enabling managed Amazon ECS cluster auto scaling allows ECS to manage the scale-in and scale-out actions of the Auto Scaling group.
- 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.

    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 (AWS ECS 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.

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

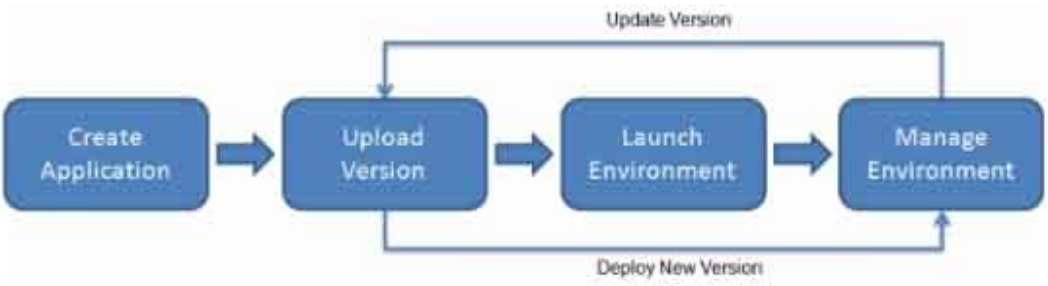
- Fargate task definitions require that the network mode is set to *awsvpc*. The *awsvpc* network mode provides each task with its own elastic network interface.
- Fargate task definitions only support the *awslogs* log driver for the log configuration. This configures your Fargate tasks to send log information to Amazon CloudWatch Logs.
- Task storage is **ephemeral**. After a Fargate task stops, the storage is deleted.

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.

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.
- Elastic Beanstalk supports Docker containers.
- Elastic Beanstalk Workflow



- Your application's domain name is in the format: *subdomain.region*.elasticbeanstalk.com

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.

AWS Lambda

- A serverless compute service. Function-as-a-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.

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.
- **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.

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.
- Commonly used SAM CLI commands
    - The **sam init** command generates pre-configured AWS SAM templates.
    - The **sam local c**ommand 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 c**ommands 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.

AWS Storage Services

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).

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.
- 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.

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.
- Good for storing static web content or media. Can be used to host static websites.

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.
- You can host static websites by configuring your bucket for website hosting.

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.
    - 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, you need to EXPLICITLY enable it.
- 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
- 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:
- 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.
    - Only certain S3 events are recorded on CloudTrail Event History. For Object level events you **enable server access logging for an S3 bucket**
- 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

Amazon S3 Bucket Policies for VPC Endpoints

What are VPC endpoints?

A VPC endpoint is what you use to privately connect your VPC to supported AWS services, such as Amazon S3. It adds a gateway entry in your VPC's route table so that communication between your AWS resources, such as Amazon EC2 instances, and your S3 bucket pass through the gateway instead of the public internet. As a result, VPC endpoint is a regional service. You should create the endpoint in the same region as the VPC you want to link it to.

VPC endpoints are best used when you have compliance requirements or sensitive information stored in S3 that should not leave the Amazon network. A VPC endpoint is also a better option for private network connections in AWS, as compared to using a VPN solution or a NAT solution since it is easier to setup and offers you more network bandwidth at your disposal.

AWS Database Services

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.
- **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 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.
- **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

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.

Core Components

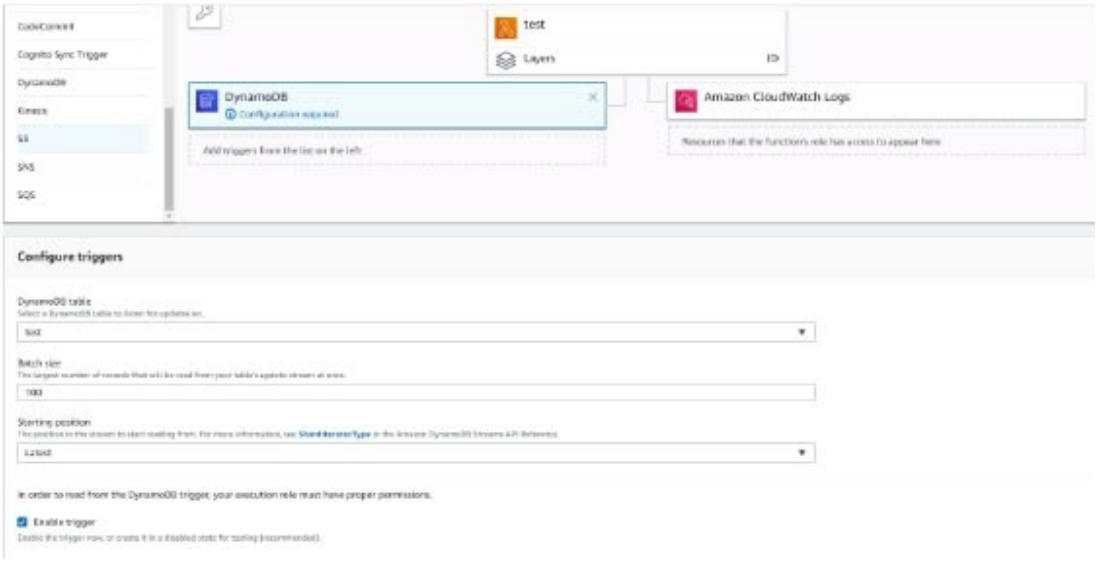
- **Tables** - a collection of items
  - DynamoDB stores data in a table, which is a collection of data.
  - Are schemaless.
  - There is an initial limit of 256 tables per region.
- **Items** - a collection of attributes
  - DynamoDB uses **primary keys** to uniquely identify each item in a table and **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.
- **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 Accelerator (DAX)

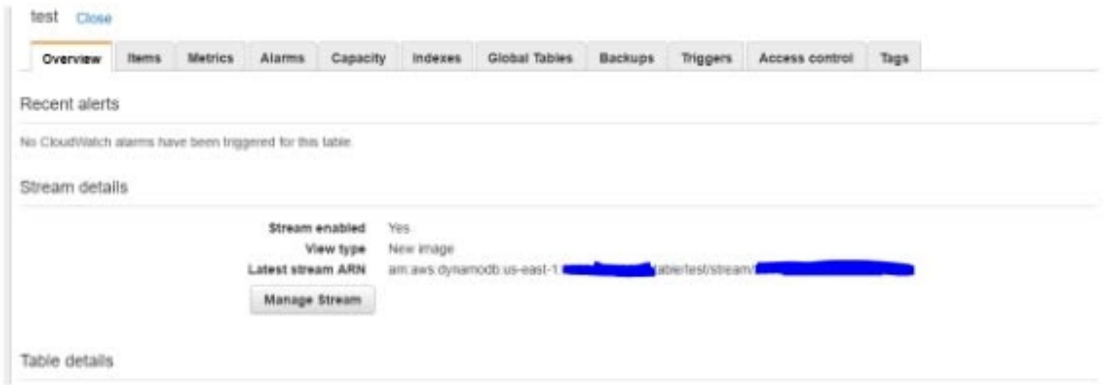
- DAX is a fully managed, highly available, in-memory cache for DynamoDB.
- **DynamoDB Accelerator (DAX)** delivers microsecond response times for accessing eventually consistent data.
- It requires only minimal functional changes to use DAX with an existing application since it is API-compatible with 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.

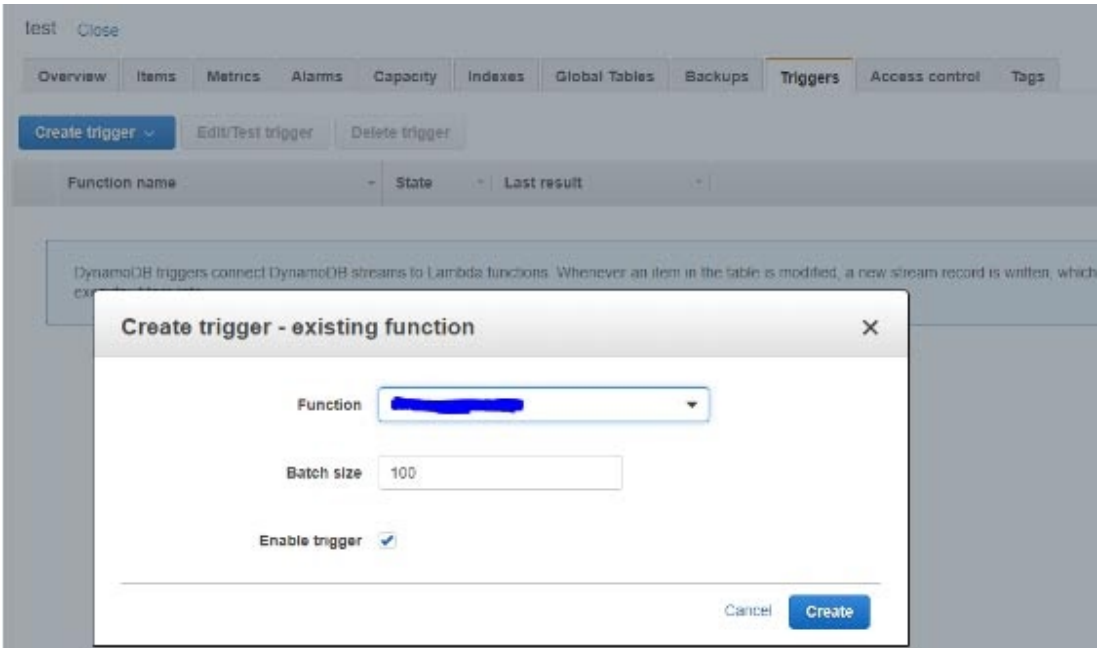
Lambda Integration With Amazon DynamoDB Streams

- 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.

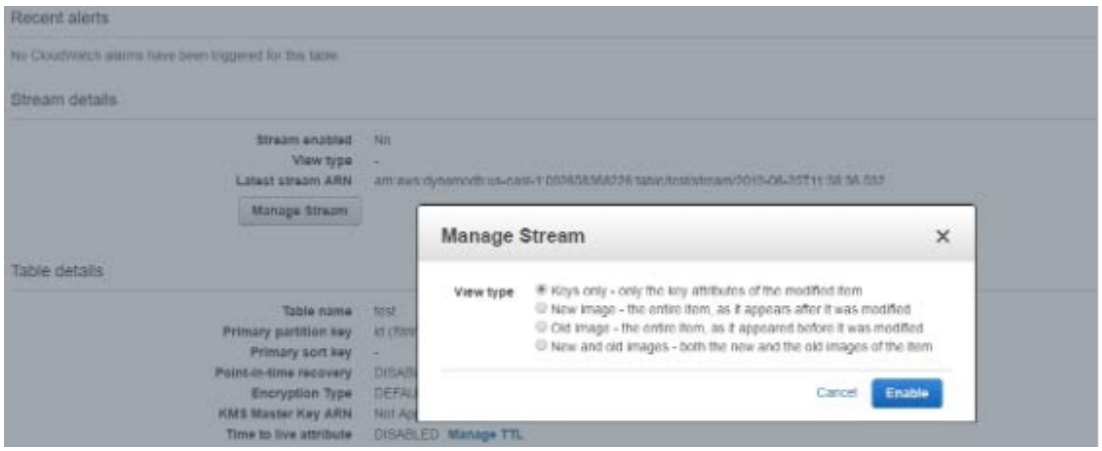


- After you enable DynamoDB Streams on a table, associate the DynamoDB table with a Lambda function. AWS Lambda polls the stream and invokes your Lambda function synchronously when it detects new stream records.





- 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.



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

Amazon RDS

- 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.
- 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.

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.
- 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**.

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**.
- 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**

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.

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.

AWS Networking & Content Delivery

Amazon API Gateway

- Enables developers to create, publish, maintain, monitor, and secure APIs at any scale.
- 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.
    - 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.
- **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.

Amazon Route 53

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

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.

  - **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 -** policy on how to redirect users based on configured routing policy
  - **Subdomain** - name below the zone apex. Example: portal.tutorialsdojo.com
  - Time to live (TTL) - time that the DNS record is cached by querying servers.
- Health Checking Concepts
  - **DNS failover** - a method for routing traffic away from unhealthy resources and to healthy resources.
  - Endpoint - the URL or endpoint on which the health check will be performed.
  - Health check - the metric on which to determine if an endpoint is healthy or not.

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 (zone apex) of a DNS namespace using a CNAME record.

AWS Elastic Load Balancing (ELB)

- 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.
- **Cross Zone Load Balancing** - when enabled, each load balancer node distributes traffic across the registered targets in all enabled AZs.

Three Types of Load Balancers

- **Application Load Balancer**
- **Network Load Balancer**
- **Classic load Balancer**
- **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:

AWS Security & Identity Services

Amazon GuardDuty

- An intelligent threat detection service. It analyzes billions of events across your AWS accounts from AWS CloudTrail (AWS user and API activity in your accounts), Amazon VPC Flow Logs (network traffic data), and DNS Logs (name query patterns).
- GuardDuty is a regional service.
- Threat detection categories
    - **Reconnaissance** -- Activity suggesting reconnaissance by an attacker, such as unusual API activity, intra-VPC port scanning, unusual patterns of failed login requests, or unblocked port probing from a known bad IP.
    - **Instance compromise** -- Activity indicating an instance compromise, such as cryptocurrency mining, backdoor command and control activity, malware using domain generation algorithms, outbound denial of service activity, unusually high volume of network traffic, unusual network protocols, outbound instance communication with a known malicious IP, temporary Amazon EC2 credentials used by an external IP address, and data exfiltration using DNS.
    - **Account compromise** -- Common patterns indicative of account compromise include API calls from an unusual geolocation or anonymizing proxy, attempts to disable AWS CloudTrail logging, changes that weaken the account password policy, unusual instance or infrastructure launches, infrastructure deployments in an unusual region, and API calls from known malicious IP addresses.
- Amazon GuardDuty provides three severity levels (Low, Medium, and High) to allow you to prioritize response to potential threats.
- CloudTrail Event Source
    - Currently, GuardDuty only analyzes CloudTrail management events. (Read about types of CloudTrail trails for more information)
    - GuardDuty processes all CloudTrail events that come into a region, including global events that CloudTrail sends to all regions, such as AWS IAM, AWS STS, Amazon CloudFront, and Route 53.
- VPC Flow Logs Event Source
    - VPC Flow Logs capture information about the IP traffic going to and from Amazon EC2 network interfaces in your VPC.
- DNS Logs Event Source
    - If you use AWS DNS resolvers for your EC2 instances (the default setting), then GuardDuty can access and process your request and response DNS logs through the internal AWS DNS resolvers. Using other DNS resolvers will not provide GuardDuty access to its DNS logs.
- GuardDuty vs Macie
    - Amazon GuardDuty provides broad protection of your AWS accounts, workloads, and data by helping to identify threats such as attacker reconnaissance, instance compromise, and account compromise. Amazon Macie helps you protect your data in Amazon S3 by helping you classify

what data you have, the value that data has to the business, and the behavior associated with access to that data.

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.

Assessment Reports

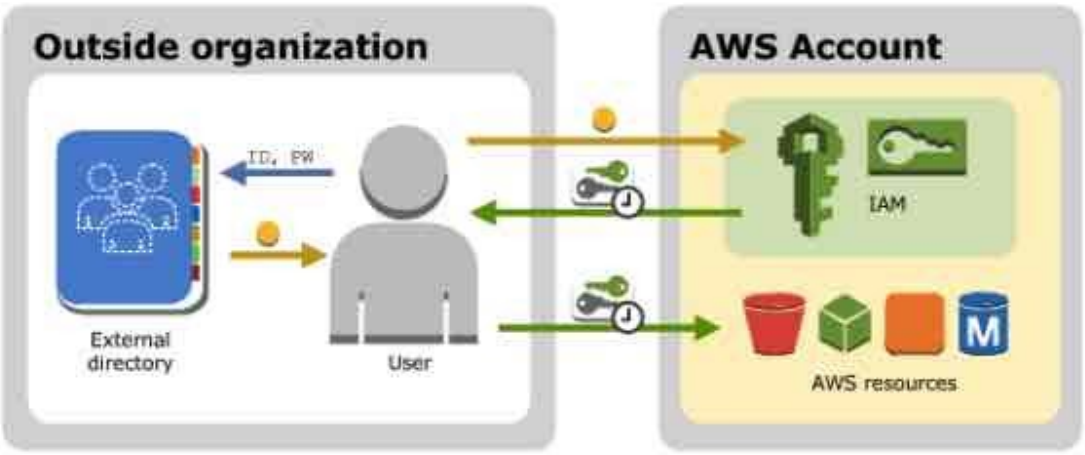
- A document that details what is tested in the assessment run, and the results of the assessment.
- You can view the following types of assessment reports:
    - **Findings report** - this report contains the following information:
        - Executive summary of the assessment
        - EC2 instances 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.

Amazon Macie

- A security service that uses machine learning to automatically discover, classify, and protect sensitive data in AWS. Macie recognizes sensitive data such as personally identifiable information (PII) or intellectual property.
- Amazon Macie allows you to achieve the following:
    - Identify and protect various data types, including PII, PHI, regulatory documents, API keys, and secret keys
    - Verify compliance with automated logs that allow for instant auditing
    - Identify changes to policies and access control lists
    - Observe changes in user behavior and receive actionable alerts
    - Receive notifications when data and account credentials leave protected zones
    - Detect when large quantities of business-critical documents are shared internally and externally
- **Concepts**
    - An **Alert** is a notification about a potential security issue that Macie discovers. Alerts appear on the Macie console and provide a comprehensive narrative about all activity that occurred over the last 24 hours.
        - Basic alerts – Alerts that are generated by the security checks that Macie performs. There are two types of basic alerts in Macie:
            - Managed (curated by Macie) basic alerts that you can't modify. You can only enable or disable the existing managed basic alerts.
            - Custom basic alerts that you can create and modify to your exact specifications.
        - Predictive alerts – Automatic alerts based on activity in your AWS infrastructure that deviates from the established normal activity baseline. More specifically, Macie continuously monitors IAM user and role activity in your AWS infrastructure and builds a model of the normal behavior. It then looks for deviations from that normal baseline, and when it detects such activity, it generates automatic predictive alerts.
    - **Data source** is the origin or location of a set of data.
        - AWS CloudTrail event logs and errors, including Amazon S3 object-level API activity. You can't modify existing or add new CloudTrail events to the list that Macie manages. You can enable or disable the supported CloudTrail events, thus instructing Macie to either include or exclude them in its data security process.
        - Amazon S3 objects. You can integrate Macie with your S3 buckets and/or specify S3 prefixes
    - **User**, in the context of Macie, a user is the AWS Identity and Access Management (IAM) identity that makes the request.

AWS Identity & 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.
    - 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.
- **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.

When to Create IAM User	When to Create an IAM Role
You created an AWS account and you're the only person who works in your account.	You're creating an application that runs on an Amazon EC2 instance and that application makes requests to AWS.
Other people in your group need to work in your AWS account, and your group is using no other identity mechanism.	You're creating an app that runs on a mobile phone and that makes requests to AWS.
You want to use the command-line interface to work with AWS.	Users in your comapany are authenticated in your corporate network and want to be able to use AWS without having no sign in again (federate into AWS)

Tutorials Dojo

AWS Key Management Service

- 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.

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.

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 (username and password), and the connection details used to access a secured service.
- 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.

AWS Management Tools

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.
        - 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*.

- **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**.

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 Agent

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

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
- **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.
    - 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.
    - 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.

    - Scaling Options
        - Scale to maintain current instance levels at all times
        - Manual Scaling
        - Scale based on a schedule
        - Scale based on a demand
    - 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.
- **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.

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 enables you to build custom extensions to your stack template using AWS Lambda.

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.

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.
- **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 change 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, and 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.
- Drift detection identifies unmanaged changes or changes made to stacks outside of CloudFormation. When CloudFormation performs drift detection on a stack set, it performs drift detection on the stack associated with each stack instance in the stack set. If the current state of a resource varies from its expected state, that resource is considered to have drifted.
- If one or more resources in a stack has drifted then the stack itself is considered to have drifted, and the stack instances that the stack is associated with is considered to have drifted as well.
- If one or more stack instances in a stack set has drifted, the stack set itself is considered to have drifted.

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.
- **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.
- **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.

AWS Config

- A fully managed service that provides you with an AWS resource inventory, configuration history, and configuration change notifications to enable security and governance.

Features

- Multi-account, multi-region data aggregation gives you an enterprise-wide view of your ***Config rule*** compliance status, and you can associate your AWS organization to quickly add your accounts.
- Provides you pre-built rules to evaluate your AWS resource configurations and configuration changes, or create your own custom rules in AWS Lambda that define your internal best practices and guidelines for resource configurations.
- **Config records** details of changes to your AWS resources to provide you with a configuration history, and automatically deliver it to an S3 bucket you specify.
- Receive a notification whenever a resource is created, modified, or deleted.
- Config enables you to record software configuration changes within your EC2 instances and servers running on-premises, as well as servers and Virtual Machines in environments provided by other cloud providers. You gain visibility into:
    - operating system configurations
    - system-level updates
    - installed applications
    - network configuration
- Config can provide you with a **configuration snapshot** - a point-in-time capture of all your resources and their configurations.

Concepts

Configuration History

- A collection of the configuration items for a given resource over any time period, containing information such as when the resource was first created, how the resource has been configured over the last month, etc.
- Config automatically delivers a configuration history file for each resource type that is being recorded to an S3 bucket that you specify.
- A configuration history file is sent every six hours for each resource type that Config records.

Configuration item

- A record of the configuration of a resource in your AWS account. Config creates a configuration item whenever it detects a change to a resource type that it is recording.
- The components of a configuration item include metadata, attributes, relationships, current configuration, and related events.

Configuration Recorder

- Stores the configurations of the supported resources in your account as configuration items.

- By default, the configuration recorder records all supported resources in the region where Config is running. You can create a customized configuration recorder that records only the resource types that you specify.
- You can also have Config record supported types of *global resources* which are IAM users, groups, roles, and customer managed policies.

Configuration Item

- The configuration of a resource at a given point-in-time. A CI consists of 5 sections:
  - Basic information about the resource that is common across different resource types.
  - Configuration data specific to the resource.
  - Map of relationships with other resources.
  - CloudTrail event IDs that are related to this state.
  - Metadata that helps you identify information about the CI, such as the version of this CI, and when this CI was captured.

Resource Relationship

- Config discovers AWS resources in your account and then creates a map of relationships between AWS resources.

Config rule

- Represents your desired configuration settings for specific AWS resources or for an entire AWS account.
- Provides customizable, predefined rules. If a resource violates a rule, Config flags the resource and the rule as noncompliant, and notifies you through Amazon SNS.
- Evaluates your resources either **in response to configuration changes** or **periodically**.

- Multi-Account Multi-Region Data Aggregation
  - An aggregator collects configuration and compliance data from the following:
    - Multiple accounts and multiple regions.
    - Single account and multiple regions.
    - An organization in AWS Organizations and all the accounts in that organization.

Monitoring

- Use Amazon SNS to send you notifications every time a supported AWS resource is created, updated, or otherwise modified as a result of user API activity.
- Use Amazon CloudWatch Events to detect and react to changes in the status of AWS Config events.
- Use AWS CloudTrail to capture API calls to Config.

AWS Health

- Provides ongoing visibility into the state of your AWS resources, services, and accounts.
- The service delivers alerts and notifications triggered by changes in the health of AWS resources.
- The **Personal Health Dashboard**, powered by the AWS Health API, is available to all customers. The dashboard requires no setup, and it is ready to use for authenticated AWS users. The Personal Health Dashboard organizes issues in three groups:
    - Open issues - restricted to issues whose start time is within the last seven days.
    - Scheduled changes - contains items that are ongoing or upcoming.
    - Other notifications - restricted to issues whose start time is within the last seven days.
- You can centrally aggregate your AWS Health events from all accounts in your AWS Organization. The AWS Health Organizational View provides centralized and real-time access to all AWS Health events posted to individual accounts in your organization, including operational issues, scheduled maintenance, and account notifications.

AWS OpsWorks

- A configuration management service that helps you configure and operate applications in a cloud enterprise by using **Puppet** or **Chef**.
- AWS OpsWorks Stacks and AWS OpsWorks for Chef Automate (1 and 2) let you use Chef cookbooks and solutions for configuration management, while OpsWorks for Puppet Enterprise lets you configure a Puppet Enterprise master server in AWS.

OpsWorks for Puppet Enterprise

- Provides a fully-managed Puppet master, a suite of automation tools that enable you to inspect, deliver, operate, and future-proof your applications, and access to a user interface that lets you view information about your nodes and Puppet activities.
- Uses puppet-agent software.
- **Features**
    - AWS manages the Puppet master server running on an EC2 instance. You retain control over the underlying resources running your Puppet master.
    - You can choose the weekly maintenance window during which OpsWorks for Puppet Enterprise will automatically install updates.
    - Monitors the health of your Puppet master during update windows and automatically rolls back changes if issues are detected.
    - You can configure automatic backups for your Puppet master and store them in an S3 bucket in your account.
    - You can register new nodes to your Puppet master by inserting a user-data script, provided in the *OpsWorks for Puppet Enterprise StarterKit*, into your Auto Scaling groups.
    - Puppet uses SSL and a certification approval process when communicating to ensure that the Puppet master responds only to requests made by trusted users.
- Deleting a server also deletes its events, logs, and any modules that were stored on the server. Supporting resources are also deleted, along with all automated backups.

OpsWorks for Chef Automate

- Lets you create AWS-managed Chef servers that include Chef Automate premium features, and use the Chef DK and other Chef tooling to manage them.
- Uses chef-client.
- **Features**
    - You can use Chef to manage both Amazon EC2 instances and on-premises servers running Linux or Windows.
    - You receive the full Chef Automate platform which includes premium features that you can use with Chef server, like Chef Workflow, Chef Visibility, and Chef Compliance.

  - You provision a managed Chef server running on an EC2 instance in your account. You retain control over the underlying resources running your Chef server and you can use Knife to SSH into your Chef server instance at any time.
  - You can set a weekly maintenance window during which OpsWorks for Chef Automate will automatically install updates.
  - You can configure automatic backups for your Chef server and is stored in an S3 bucket.
  - You can register new nodes to your Chef server by inserting user-data code snippets provided by OpsWorks for Chef Automate into your Auto Scaling groups.
  - Chef uses SSL to ensure that the Chef server responds only to requests made by trusted users. The Chef server and Chef client use bidirectional validation of identity when communicating with each other.
- Deleting a server also deletes its events, logs, and any cookbooks that were stored on the server. Supporting resources are deleted also, along with all automated backups.

OpsWorks Stacks

- Provides a simple and flexible way to create and manage stacks and applications.
- You can create stacks that help you manage cloud resources in specialized groups called **layers**. A layer represents a set of EC2 instances that serve a particular purpose, such as serving applications or hosting a database server. Layers depend on Chef recipes to handle tasks such as installing packages on instances, deploying apps, and running scripts.
- **Features**
  - You can deploy EC2 instances from template configurations, including EBS volume creation.
  - You can configure the software on your instances on-demand or automatically based on lifecycle events, from bootstrapping the base OS image into a working server to modifying running services to reflect changes.
  - OpsWorks Stacks can auto heal your stack. If an instance fails in your stack, OpsWorks Stacks can replace it with a new one.
  - You can adapt the number of running instances to match your load, with time-based or load-based auto scaling.
  - You can use OpsWorks Stacks to configure and manage both Linux and Windows EC2 instances.
  - You can use AWS OpsWorks Stacks to deploy, manage, and scale your application on any Linux server such as EC2 instances or servers running in your own data center.
- **Instance Types**
  - **24/7 instances** are started manually and run until you stop them.
  - **Time-based instances** are run by OpsWorks Stacks on a specified daily and weekly schedule. They allow your stack to automatically adjust the number of instances to accommodate predictable usage patterns.

- **Load-based instances** are automatically started and stopped by OpsWorks Stacks, based on specified load metrics, such as CPU utilization. They allow your stack to automatically adjust the number of instances to accommodate variations in incoming traffic.
    - Load-based instances are available only for Linux-based stacks.

- **Lifecycle Events**
    - You can run recipes manually, but OpsWorks Stacks also lets you automate the process by supporting a set of five lifecycle events:
        - **Setup** occurs on a new instance after it successfully boots.
        - **Configure** occurs on all of the stack's instances when an instance enters or leaves the online state.
        - **Deploy** occurs when you deploy an app.
        - **Undeploy** occurs when you delete an app.
        - **Shutdown** occurs when you stop an instance.

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.

AWS Trusted Advisor

- Trusted Advisor analyzes your AWS environment and provides best practice recommendations in five categories:
    - Cost Optimization
    - Performance
    - Security
    - Fault Tolerance
    - Service Limits
- Access to the seven core Trusted Advisor checks are available to all AWS users.
- Access to the full set of Trusted Advisor checks are available to Business and Enterprise Support plans.

AWS Analytics Services

Amazon Elasticsearch (ES)

- Amazon ES lets you search, analyze, and visualize your data **in real-time**. This service manages the capacity, scaling, patching, and administration of your Elasticsearch clusters for you, while still giving you direct access to the Elasticsearch APIs.
- The service offers open-source Elasticsearch APIs, managed Kibana, and integrations with Logstash and other AWS Services. This combination is often coined as the **ELK Stack**.
- **Concepts**
    - An Amazon ES **domain** is synonymous with an Elasticsearch cluster. Domains are clusters with the settings, instance types, instance counts, and storage resources that you specify.
    - You can create multiple Elasticsearch indices within the same domain. Elasticsearch automatically distributes the indices and any associated replicas between the instances allocated to the domain.
    - Amazon ES uses a **blue/green deployment process** when updating domains. Blue/green typically refers to the practice of running two production environments, one live and one idle, and switching the two as you make software changes.
- **Data Ingestion**
    - Easily ingest structured and unstructured data into your Amazon Elasticsearch domain with **Logstash**, an open-source data pipeline that helps you process logs and other event data.
    - You can also ingest data into your Amazon Elasticsearch domain using Amazon Kinesis Firehose, AWS IoT, or Amazon CloudWatch Logs.
    - You can get faster and better insights into your data using **Kibana**, an open-source analytics and visualization platform. Kibana is automatically deployed with your Amazon Elasticsearch Service domain.
    - You can load streaming data from the following sources using AWS Lambda event handlers:
        - Amazon S3
        - Amazon Kinesis Data Streams and Data Firehose
        - Amazon DynamoDB
        - Amazon CloudWatch
        - AWS IoT
    - Amazon ES exposes three Elasticsearch logs through CloudWatch Logs:
        - error logs
        - search slow logs - These logs help fine tune the performance of any kind of search operation on Elasticsearch.
        - index slow logs - These logs provide insights into the indexing process and can be used to fine-tune the index setup.
    - Kibana and Logstash
        - Kibana is a popular open source visualization tool designed to work with Elasticsearch.

- The URL is *elasticsearch-domain-endpoint*/_plugin/kibana/.
- You can configure your own Kibana instance aside from using the default provided Kibana.
- Amazon ES uses **Amazon Cognito** to offer username and password protection for Kibana. (Optional feature)
- Logstash provides a convenient way to use the bulk API to upload data into your Amazon ES domain with the S3 plugin. The service also supports all other standard Logstash input plugins that are provided by Elasticsearch.
- Amazon ES also supports two Logstash output plugins:
  - standard Elasticsearch plugin
  - *logstash-output-amazon-es* plugin, which signs and exports Logstash events to Amazon ES.

Amazon Kinesis

- 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 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 **not** 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.
  - **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.

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.
    - 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.

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.

- 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.

AWS Developer Tools

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 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.

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.

- **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.
- **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.

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.
    1. 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.
    2. In an AWS Lambda deployment, a deployment group defines a set of CodeDeploy configurations for future deployments of an AWS Lambda function.
    3. 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 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.
    1. ApplicationStop
    2. DownloadBundle
    3. BeforeInstall
    4. Install
    5. AfterInstall
    6. ApplicationStart
    7. 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

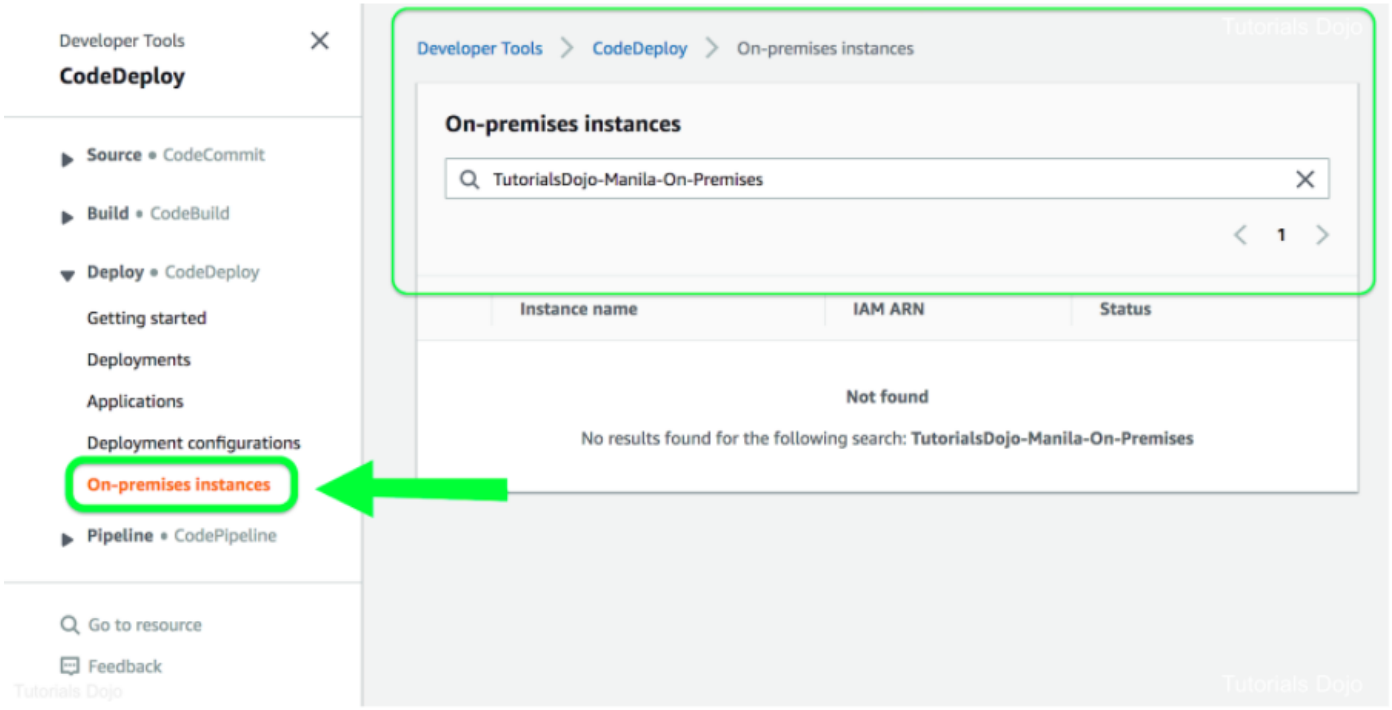
- You can use the CodeDeploy console or the create-deployment command to deploy the function revision specified in the AppSpec file to the deployment group.
- You can use the CodeDeploy console or the stop-deployment command to stop a deployment. When you attempt to stop the deployment, one of three 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.
- With AWS CodeDeploy, you can also deploy your applications to your on-premises data centers. Your on-premises instances will have a prefix of “mi-xxxxxxxxx”.


Developer Tools
CodeDeploy
Source • CodeCommit
Build • CodeBuild
Deploy • CodeDeploy
Getting started
Deployments
Applications
Deployment configurations
On-premises instances
Pipeline • CodePipeline
Go to resource
Feedback
Developer Tools > CodeDeploy > On-premises instances
On-premises instances
TutorialsDojo-Manila-On-Premises
1
Instance name
IAM ARN
Status
Not found
No results found for the following search: TutorialsDojo-Manila-On-Premises

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 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.
- **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*.
    - 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.

AWS Application Services

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 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

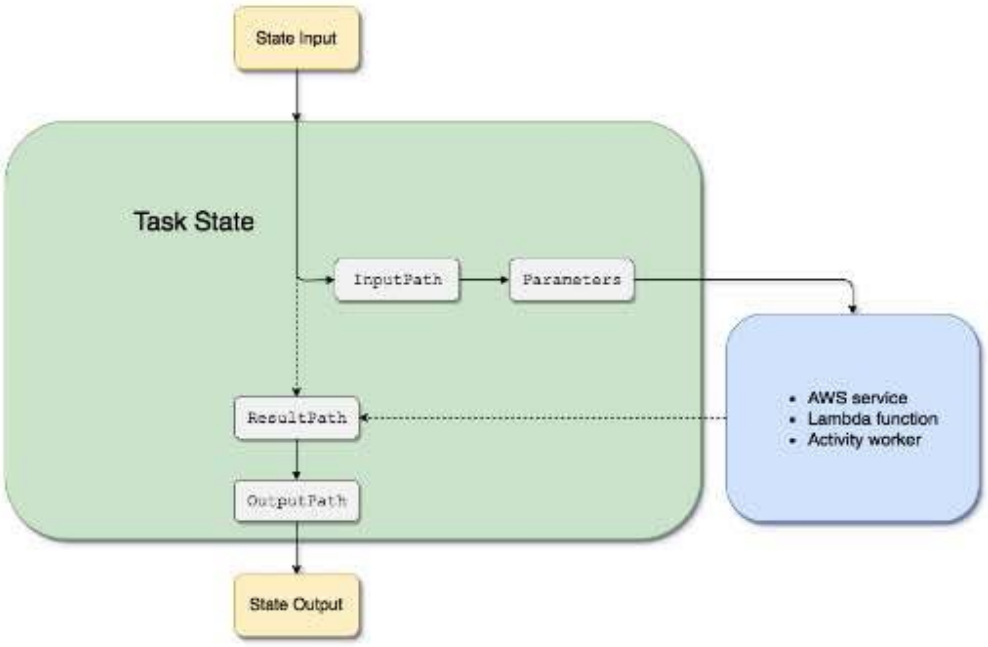
- Instead of including a specific destination address in each message, a publisher sends a message to a **topic**. SNS matches the topic to a list of subscribers who have subscribed to that topic, and delivers the message to each of those subscribers.
- Each topic has a unique name that identifies the SNS endpoint for publishers to post messages and subscribers to register for notifications.
- A topic can support subscriptions and notification deliveries over multiple transports.

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 to 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 to the endpoint, you select whether SNS uses HTTP or HTTPS to send the POST request to the endpoint.

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.
    - 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.

Comparison of AWS Services

AWS CloudTrail vs Amazon 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.

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).

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.

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.

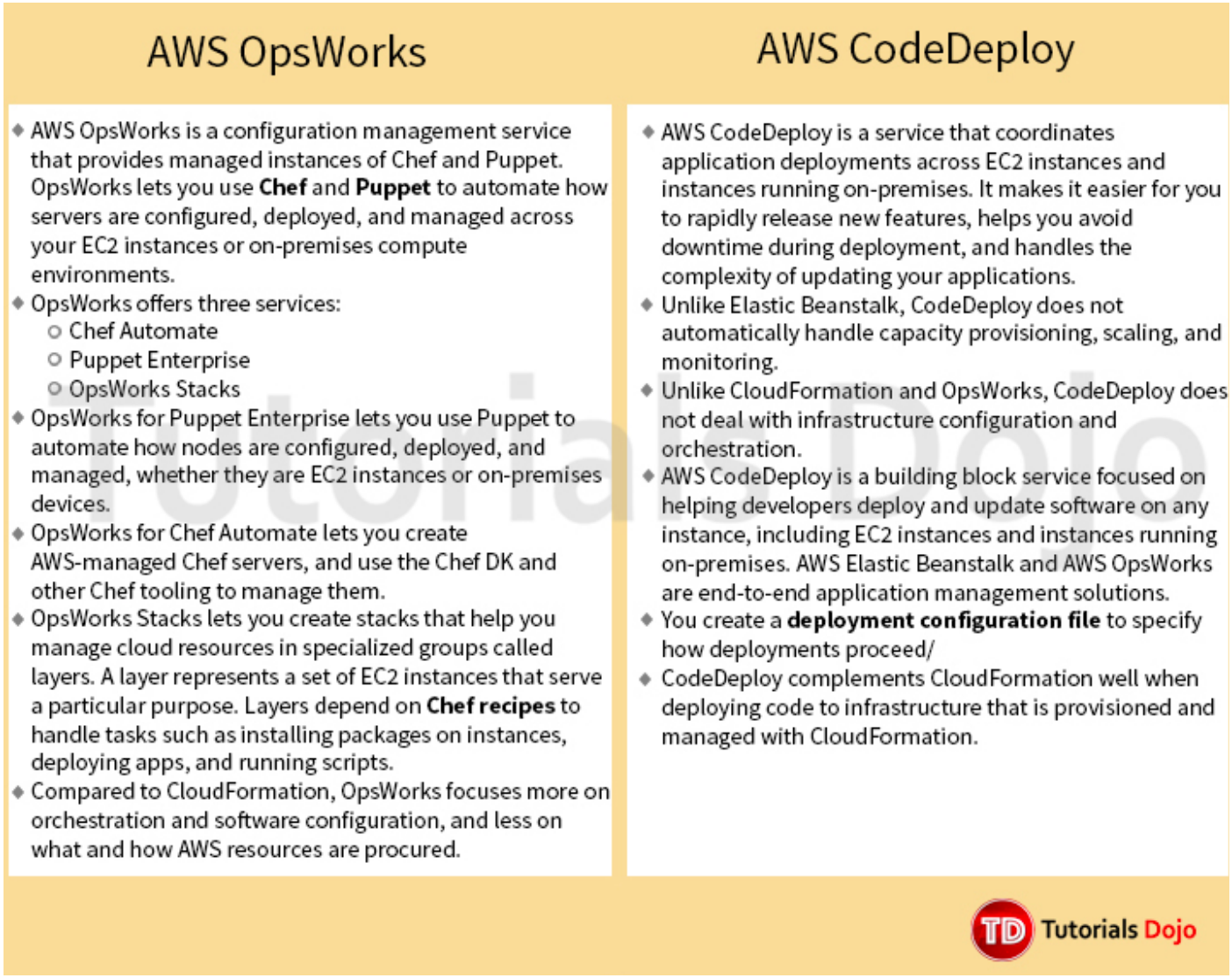
Elastic Beanstalk vs CloudFormation vs OpsWorks vs CodeDeploy

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.

Service Control Policies vs IAM Policies

Service Control Policies (SCP)

- SCPs are mainly used along with AWS Organizations organizational units (OUs).
- SCPs do not replace IAM Policies such that they do not provide actual permissions. To perform an action, you would still need to grant appropriate IAM Policy permissions.
- Even if a Principal is allowed to perform a certain action (granted through IAM Policies), an attached SCP will override that capability if it enforces a Deny on that action.
- SCP takes precedence over IAM Policies.
- SCPs can be applied to the root of an organization or to individual accounts in an OU.
- When you apply an SCP to an OU or an individual AWS account, you choose to either enable (whitelist), or disable (blacklist) the specified AWS service. Access to any service that isn't explicitly allowed by the SCPs associated with an account, its parent OUs, or the management account is denied to the AWS accounts or OUs associated with the SCP.
- Any account has only those permissions permitted by every parent above it. If a permission is blocked at any level above the account, either implicitly (by not being included in an Allow policy statement) or explicitly (by being included in a Deny policy statement), a user or role in the affected account can't use that permission, even if there is an attached IAM policy granting Administrator permissions to the user.
- SCPs affect only principals that are managed by accounts that are part of the organization.

IAM Policies

- IAM Policies operate at the Principal level. There are two types of IAM policies
  - Identity-based policies - attached to an IAM user, group, or role.
  - Resource-based policies - attached to an AWS resource such as an S3 bucket.
- IAM Policies can grant/deny a Principal permissions to perform certain actions to certain resources. This can be used together with SCP to ensure stricter controls in AWS Organizations.
  An IAM policy can be applied only to IAM users, groups, or roles, and it can never restrict the root identity of the AWS account.
- IAM Policies cannot be attached to OUs.
- An IAM Policy can allow or deny actions. An explicit allow overrides an implicit deny. An explicit deny overrides an explicit allow.

Tutorials Dojo