

Contents

[AWS Well Architected Framework 5](#_Toc148613146)

[Pillars 5](#_Toc148613147)

[Shared Responsibility Model 6](#_Toc148613148)

[Compute Services 6](#_Toc148613149)

[EKS – Elastic Kubernetes Service 6](#_Toc148613150)

[ECR – Elastic Container Registry 9](#_Toc148613151)

[ECS – Elastic Container Service 10](#_Toc148613152)

[Types of Deployments of ECS 13](#_Toc148613153)

[EC2 – Elastic Compute Cloud 14](#_Toc148613154)

[Elastic Beanstalk 15](#_Toc148613155)

[Storage Services 16](#_Toc148613156)

[AWS Backup 16](#_Toc148613157)

[EBS – Elastic Block Storage 16](#_Toc148613158)

[EFS – Elastic file storage 19](#_Toc148613159)

[S3 – Simple Storage Service 21](#_Toc148613160)

[Storage Gateway 29](#_Toc148613161)

[Amazon FSx for Lustre Simplified: 32](#_Toc148613162)

[Amazon FSx for Windows File Server 32](#_Toc148613163)

[AWS Snowball 33](#_Toc148613164)

[Database Services 34](#_Toc148613165)

[Aurora 34](#_Toc148613166)

[**DynamoDB –** NoSQL support, 36](#_Toc148613167)

[Elasticache 37](#_Toc148613168)

[RDS 40](#_Toc148613169)

[RDS Security: 41](#_Toc148613170)

[RDS Enhanced Monitoring: 41](#_Toc148613171)

[Redshift 42](#_Toc148613172)

[Redshift Spectrum 43](#_Toc148613173)

[How does Redshift Spectrum work? 43](#_Toc148613174)

[Getting started with Redshift Spectrum step by step 44](#_Toc148613175)

[AWS DocumentDB 44](#_Toc148613176)

[AWS Neptune 44](#_Toc148613177)

[Network 45](#_Toc148613178)

[Subnet 45](#_Toc148613179)

[Subnetting 45](#_Toc148613180)

[API Gateway 46](#_Toc148613181)

[Cloudfront 46](#_Toc148613182)

[Route 53 48](#_Toc148613183)

[VPC – Virtual Private Cloud 49](#_Toc148613184)

[AWS Direct Connect 50](#_Toc148613185)

[AWS Global accelerator: 51](#_Toc148613186)

[Transit Gateway: 52](#_Toc148613187)

[VPC Flow logs 52](#_Toc148613188)

[NAT – Network access Translation 53](#_Toc148613189)

[NAT Instance and NAT Gateways 53](#_Toc148613190)

[Nat Gateways 53](#_Toc148613191)

[AWS Site to Site VPN 54](#_Toc148613192)

[AWS Private Link 55](#_Toc148613193)

[Security 55](#_Toc148613194)

[Security Group: 55](#_Toc148613195)

[NACL, CloudFront, ALB, WAF 56](#_Toc148613196)

[Web Application Firewall (WAF) 56](#_Toc148613197)

[Network access control List (NACL) 57](#_Toc148613198)

[Load Balancers 57](#_Toc148613199)

[AWS Cognito and Web Identity Federation 61](#_Toc148613200)

[Key Management System (KMS) 61](#_Toc148613201)

[AWS EC2 System Manager 64](#_Toc148613202)

[Lambda 64](#_Toc148613203)

[Elastic Container Service (ECS) 64](#_Toc148613204)

[AWS Shield - Distributed Denial of Service (DDOS) 65](#_Toc148613205)

[AWS Firewall Manager 66](#_Toc148613206)

[AWS Identity and Access Management (IAM) 67](#_Toc148613207)

[Amazon GuardDuty 70](#_Toc148613208)

[Amazon Inspector 71](#_Toc148613209)

[Amazon Macie 72](#_Toc148613210)

[AWS System Manager 72](#_Toc148613211)

[AWS Secrets Manager 73](#_Toc148613212)

[AWS Trusted Advisor 73](#_Toc148613213)

[Management Tools 73](#_Toc148613214)

[CloudWatch (PERFORMANCE) 73](#_Toc148613215)

[Cloud Trail: (Auditing) 75](#_Toc148613216)

[Cloud Front 77](#_Toc148613217)

[Cloud Formation: 77](#_Toc148613218)

[AWS Organizations 78](#_Toc148613219)

[Service Control Policies (SCP) 78](#_Toc148613220)

[Directory Services 80](#_Toc148613221)

[AWS Applications Services 81](#_Toc148613222)

[VPN Cloud Hub 81](#_Toc148613223)

[SQS – Simple Query Service 81](#_Toc148613224)

[Simple Work Force Service (SWF) 82](#_Toc148613225)

[Vertical scaling and Horizontal scaling 83](#_Toc148613226)

[AWS Kinesis 84](#_Toc148613227)

[Amazon Cognito: 85](#_Toc148613228)

[Orchestration and Deployments 85](#_Toc148613229)

[Cost Optimization 85](#_Toc148613230)

[EC2 Auto scaling 85](#_Toc148613231)

[Parameters 86](#_Toc148613232)

[AWS Application auto-scaling 86](#_Toc148613233)

[Auto-Scaling 87](#_Toc148613234)

[Auto Scaling Default Termination Policy: 87](#_Toc148613235)

[Auto Scaling Cooldown Period: 88](#_Toc148613236)

[Backup and Recovery 88](#_Toc148613237)

[Serverless Platform 90](#_Toc148613238)

[AWS Machine Learning Components 90](#_Toc148613239)

[Service Catalogue 90](#_Toc148613240)

[Service Catalogue Templates 91](#_Toc148613241)

# AWS Well Architected Framework

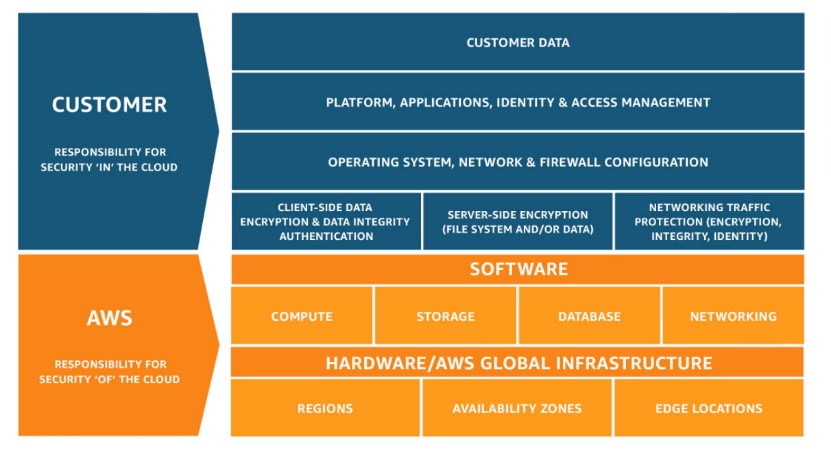
## Pillars

6 pillars, Help to guide to design the AWS/application/infrastructure architecture as per best practices.

* Operational Excellence Pillar
* Security Pillar
* Reliability Pillar
* Performance Efficiency Pillar
* Cost Optimization Pillar
* Sustainability

|  |  |
| --- | --- |
| **Name** | **Description** |
| **Operational Excellence** | The ability to support development and run workloads effectively, gain insight into their operations, and to continuously improve supporting processes and procedures to deliver business value. |
| **Security** | The security pillar describes how to take advantage of cloud technologies to protect data, systems, and assets in a way that can improve your security posture. |
| **Reliability** | The reliability pillar encompasses the ability of a workload to perform its intended function correctly and consistently when it’s expected to. This includes the ability to operate and test the workload through its total lifecycle. This paper provides in-depth, best practice guidance for implementing reliable workloads on AWS. |
| **Performance Efficiency** | The ability to use computing resources efficiently to meet system requirements, and to maintain that efficiency as demand changes and technologies evolve. |
| **Cost Optimization** | The ability to run systems to deliver business value at the lowest price point. |
| **Sustainability** | The ability to continually improve sustainability impacts by reducing energy consumption and increasing efficiency across all components of a workload by maximizing the benefits from the provisioned resources and minimizing the total resources required. |
| **Resiliency** | The ability for a system to recover from a failure induced by load, attacks, and failures.  A resilient workload has the capability to recover when stressed by load (more requests for service), attacks (either accidental through a bug, or deliberate through intention), and failure of any component in the workload's components |

## Shared Responsibility Model



# Compute Services

## EKS – Elastic Kubernetes Service

A managed service that allows you to run Kubernetes on AWS without installing, operating, or maintaining your own Kubernetes control plane or nodes. Integration with various AWS services to provide scalability and security for your applications:

* Amazon ECR for container images
* Elastic Load Balancing for load distribution
* IAM for authentication
* Amazon VPC for isolation

**Amazon EKS Components**

1. Clusters: An EKS cluster is made up of two main components:

* EKS control plane
  + It is made up of nodes that run the Kubernetes software (API server & *etcd*).
    - etcd (pronounced et-see-dee) is an open source, distributed, consistent key-value store for shared configuration, service discovery, and scheduler coordination of distributed systems or clusters of machines
  + Each cluster is single-tenant and unique, and runs on its own set of EC2 instances.
  + Cluster control plane is provisioned across multiple AZs and fronted by an ELB Network Load Balancer.
  + Use AWS KMS to encrypt data stored by *etcd*nodes and associated EBS volumes.
* EKS nodes
  + A cluster consists of one or more EC2 nodes on which pods are scheduled.
  + A Kubernetes pod is the way that Kubernetes runs containers on a compute instance and includes containers and specifications for how they should run, networking, and storage. A pod can be a single container or multiple containers that always run together.
  + Connects to the cluster’s control plane via the API server endpoint.
* The API server endpoint is public to the internet by default, but you can enable private access to keep communication between nodes and the API server within the VPC.
* EKS supports two autoscaling products:
  + Cluster Auto-scaler – uses AWS Auto Scaling groups.
  + Karpenter – works directly with the Amazon EC2 Fleet.
* By default, cluster control plane logs aren’t sent to CloudWatch Logs. In order to send logs for your cluster, you have to enable each log type individually.
* EKS cluster uses IAM / OIDC for authentication and Kubernetes RBAC for authorization.

1. Nodes

* Nodes must be in the same VPC as the subnets you chose when creating a cluster.
* From the perspective of the Kubernetes API, nodes represent the compute resources provisioned for your cluster.
* Taints and tolerations prevent pods from being scheduled on the wrong nodes.
  + Self-managed nodes
    - A cluster can have several node groups.
    - A node group is a collection of one or more EC2 instances deployed in an Amazon EC2 Auto Scaling group.
    - In a node group, instances must have the following characteristics:
      * Same instance type
      * Running the same AMI
      * Uses the same EKS node IAM role
  + Node groups with different instance types and host operating systems can exist in a cluster.
  + There are two methods for updating self-managed node groups in a cluster to use a new AMI:
    - Migrating to a new node group
    - Updating an existing self-managed node group
  + Managed node groups
    - Automates the provisioning and lifecycle management of nodes in EKS clusters.
    - Every managed node is provisioned as part of Amazon EC2 Auto Scaling group.
    - When nodes are launched as part of a managed node group, they are automatically tagged for auto-discovery by Kubernetes Cluster Autoscaler.
    - Use node group to apply Kubernetes labels to nodes.
    - Multiple managed node groups can exist in a single cluster.
    - When you create a managed node group, you have the option of selecting On-Demand or Spot instances.
    - To ensure that your applications remain available, node updates and terminations drain nodes automatically.

1. AWS Fargate

* You must first define a Fargate profile before scheduling pods on Fargate in your cluster.
  + A pod is the smallest execution unit in Kubernetes. A pod encapsulates one or more applications. Pods are ephemeral by nature, if a pod (or the node it executes on) fails, Kubernetes can automatically create a new replica of that pod to continue operations
* If a pod matches more than one Fargate profile, Amazon EKS picks one at random.
* Fargate profiles are immutable and contains the following components:
  + Pod execution role
  + Subnets
  + Selectors
  + Namespace
  + Labels
* Fargate runs only one pod per node.
* Pod storage is ephemeral, and data is encrypted with AWS Fargate managed keys.
* To encrypt ephemeral pod storage, you can use AWS Fargate managed keys.

1. Workloads

* Workloads are deployed in containers and define the applications that run on a Kubernetes cluster
* A pod can contain one or more containers.
* Vertical Pod Autoscaler adjusts your pods’ CPU and memory reservations.
* Horizontal Pod Autoscaler adjusts the number of pods in a deployment, replication controller, or replica set based on CPU utilization.

1. EKS Connector

* Enables you to register and connect any Kubernetes cluster to AWS.
* You can view the status, configuration, and workloads of the cluster in the Amazon EKS console after it has been connected.

**Amazon EKS Networking**

There are three ways to create a VPC for an EKS cluster:

* Private subnets
  + Three private subnets are distributed across different AZs.
  + Nodes have the option of sending and receiving internet traffic via a NAT instance or NAT gateway.
  + The cluster endpoint can only be accessed via your VPC. Traffic from worker nodes to the endpoint will remain within your VPC.
* Public subnets
  + Three public subnets are distributed across different AZs.
  + Nodes are assigned public IPv4 addresses by default and can send and receive internet traffic via an internet gateway.
  + The cluster endpoint can be accessed from outside your VPC. Traffic from worker nodes will leave your VPC to connect to the endpoint.
* Public and private subnets
  + Each AZ has one private and public subnet.
    - Nodes are deployed to private subnets.
    - Load balancers are assigned to public subnets to load balance traffic to pods running on nodes.
  + Public IPv4 addresses are automatically assigned to nodes deployed in public subnets.
  + IPv6 addresses can be assigned to nodes in both public and private subnets.
  + A NAT gateway (IPv4) or an egress-only Internet gateway (IPv6) can be used to allow pods to communicate outbound to the internet.
  + The cluster endpoint can be accessed from outside your VPC. Traffic from worker nodes to the endpoint will remain within your VPC.

The cluster security group manages communication between the control plane and the cluster’s compute resources (worker nodes and Fargate pods).

* Pod networking
  + Container Network Interface (CNI) is a plugin that assigns a private IPv4/IPv6 address from VPC to each pod.
  + VPC CNI plugin is deployed to each EC2 node in a Daemonset under the name *aws-node* and consists of two components:
  + L-IPAM daemon
    - Creates and attaches network interfaces to EC2 instances.
    - Assigns secondary IP addresses to network interfaces.
    - Maintains a warm pool of IP addresses that will be assigned to pods on each node.
  + CNI plugin
    - Configures the host network and adds the correct network interface to the pod namespace.
  + You can’t assign both IPv4 and IPv6 addresses (dual-stacked) to pods and services.
  + With security groups for pods, you can control the inbound and outbound network traffic to and from your pods.
  + Attach multiple network interfaces to a pod using the Multus CNI plugin.
  + CNI metrics helper is a tool that allows you to:
    - Scrape network interface and IP address information.
    - Aggregate metrics at the cluster level.
    - Publish the cluster’s CNI metrics to CloudWatch.
* AWS Load Balancer Controller
  + In charge of managing AWS Elastic Load Balancers in a Kubernetes cluster and provisions the following load balancers:
    - ALB when you create a Kubernetes *Ingress*.
    - NLB when you create a Kubernetes service of type *LoadBalancer*.
* CoreDNS
  + A DNS service within EKS clusters that allows individual containers to easily discover and connect to other containers in the cluster.
  + By default, two replicas of the CoreDNS image are deployed to an EKS cluster.
* Kube-proxy
  + Maintains network rules on each Amazon EC2 node.
  + Enables network communication to pods from network sessions inside/outside of the cluster.
* Calico
  + A network policy engine to implement network segmentation and tenant isolation.
  + Pod selectors and labels can be used to assign network policies to pods.

## ECR – Elastic Container Registry

A managed AWS Docker registry service. Amazon ECR is a regional service.

1. 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.
2. ECR stores both the containers you create and any container software you buy through AWS Marketplace.
3. 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.
4. ECR stores both the containers you create and any container software you buy through AWS Marketplace.

**Components**

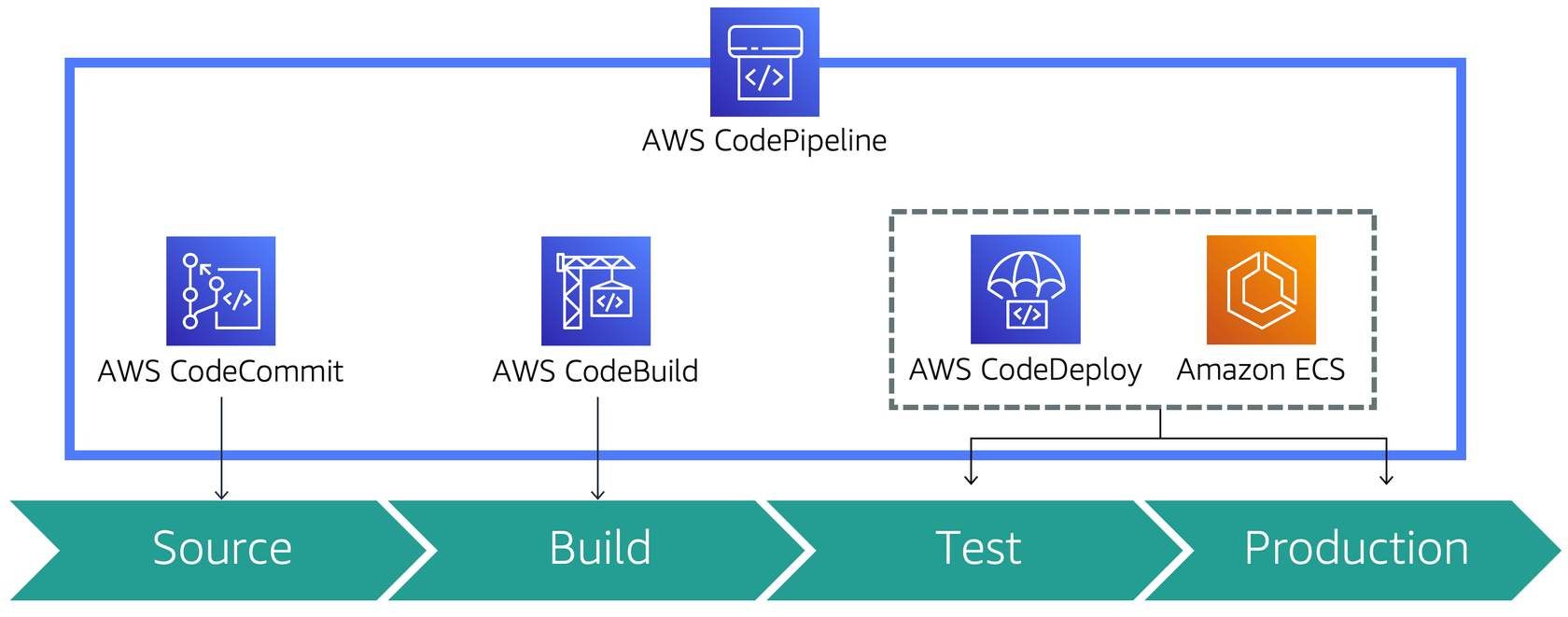
1. Registry
   * A registry is provided to each AWS account; you can create image repositories in your registry and store images in them.
   * The URL for default registry is https://aws\_account\_id.dkr.ecr.region.amazonaws.com.
   * You must be authenticated before you can use your registry.
2. Authorization 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.
3. Repository
   * An image repository contains your Docker images.
   * ECR uses resource-based permissions to let you specify who has access to a repository and what actions they can perform on it.
   * ECR lifecycle policies enable you to specify the lifecycle management of images in a repository.
4. Repository policy
   * You can control access to your repositories and the images within them with repository policies.
5. 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.

## ECS – Elastic Container Service

* A container management service to run, stop and manage Docker containers on a cluster.
* ECS can be used to create a consistent deployment and build experience, manage, and scale batch and **Extract-Transform-Load** (ETL) workloads, and build sophisticated application architectures on a microservices model.
* Amazon ECS is a regional service.
* AWS Compute SLA guarantees a Monthly Uptime Percentage of at least 99.99% for Amazon ECS.
* Amazon ECS Exec is a way for customers to execute commands in a container running on Amazon EC2 instances or AWS Fargate. ECS Exec gives you interactive shell or single command access to a running container.

**Components**

* Containers and Images
  + Your application components must be architected to run in containers ー containing everything that your software application needs to run: code, runtime, system tools, system libraries, etc.
  + Containers are created from a read-only template called an image.
  + Images are typically built from a Dockerfile, a plain text file that specifies all of the components that are included in the container. These images are then stored in a registry from which they can be downloaded and run on your cluster.
  + When you launch a container instance, you have the option of passing user data to the instance. The data can be used to perform common automated configuration tasks and even run scripts when the instance boots.
  + Docker Volumes can be a local instance store volume, EBS volume, or EFS volume. Connect your Docker containers to these volumes using Docker drivers and plugins.
* Task Components
  + Task definitions specify various parameters for your application. It is a text file, in JSON format, that describes one or more containers, up to a maximum of ten, that form your application.
  + Task definitions are split into separate parts:
    - Task family – the name of the task, and each family can have multiple revisions.
    - IAM task role – specifies the permissions that containers in the task should have.
    - Network mode – determines how the networking is configured for your containers.
    - Container definitions – specify which image to use, how much CPU and memory the container is allocated, and many more options.
    - Volumes – allow you to share data between containers and even persist the data on the container instance when the containers are no longer running.
    - Task placement constraints – lets you customize how your tasks are placed within the infrastructure.
    - Launch types – determines which infrastructure your tasks use.
* Tasks and Scheduling
  + A task is the instantiation of a task definition within a cluster. After you have created a task definition for your application, you can specify the number of tasks that will run on your cluster.
    - Each task that uses the Fargate launch type has its own isolation boundary and does not share the underlying kernel, CPU resources, memory resources, or elastic network interface with another task.
  + The task scheduler is responsible for placing tasks within your cluster. There are several different scheduling options available.
    - REPLICA — places and maintains the desired number of tasks across your cluster. By default, the service scheduler spreads tasks across Availability Zones. You can use task placement strategies and constraints to customize task placement decisions.
    - DAEMON — deploys exactly one task on each active container instance that meets all of the task placement const raints that you specify in your cluster. When using this strategy, there is no need to specify a desired number of tasks, a task placement strategy, or use Service Auto Scaling policies.
  + You can upload a new version of your application task definition, and the ECS scheduler automatically starts new containers using the updated image and stop containers running the previous version.
  + Amazon ECS tasks running on both Amazon EC2 and AWS Fargate can mount Amazon Elastic File System (EFS) file systems.
* Clusters
  + When you run tasks using ECS, you place them in a cluster, which is a logical grouping of resources.
  + Clusters are Region-specific.
  + Clusters can contain tasks using both the Fargate and EC2 launch types.
  + When using the Fargate launch type with tasks within your cluster, ECS manages your cluster resources.
  + When using the EC2 launch type, then your clusters are a group of container instances you manage. These clusters can contain multiple different container instance types, but each container instance may only be part of one cluster at a time.
  + Before you can delete a cluster, you must delete the services and deregister the container instances inside that cluster.
  + Enabling managed Amazon ECS cluster auto-scaling allows ECS to manage the scale-in and scale-out actions of the Auto Scaling group. On your behalf, Amazon ECS creates an AWS Auto Scaling scaling plan with a target tracking scaling policy based on the target capacity value that you specify.
* Services
  + ECS allows you to run and maintain a specified number of instances of a task definition simultaneously in a cluster.
  + In addition to maintaining the desired count of tasks in your service, you can optionally run your service behind a load balancer.
  + There are two deployment strategies in ECS:
    - **Rolling Update**
      * This involves the service scheduler replacing the currently running version of the container with the latest version.
      * The number of tasks ECS adds or removes from the service during a rolling update is controlled by the deployment configuration, which consists of the minimum and maximum number of tasks allowed during service deployment.
    - **Blue/Green Deployment with AWS CodeDeploy**
      * This deployment type allows you to verify a new deployment of a service before sending production traffic to it.
      * The service must be configured to use either an Application Load Balancer or Network Load Balancer.
* Container Agent
  + The container agent runs on each infrastructure resource within an ECS cluster.
  + It sends information about the resource’s current running tasks and resource utilization to ECS, and starts and stops tasks whenever it receives a request from ECS.
  + The container agent is only supported on Amazon EC2 instances.
* Service Load Balancing
  + Amazon ECS services support the Application Load Balancer, Network Load Balancer, and Classic Load Balancer ELBs. Application Load Balancers are used to route HTTP/HTTPS (or layer 7) traffic. Network Load Balancers are used to route TCP or UDP (or layer 4) traffic. Classic Load Balancers are used to route TCP traffic.
  + You can attach multiple target groups to your Amazon ECS services that are running on either Amazon EC2 or AWS Fargate. This allows you to maintain a single ECS service that can serve traffic from both internal and external load balancers and support multiple path-based routing rules and applications that need to expose more than one port.
  + The Classic Load Balancer doesn’t allow you to run multiple copies of a task on the same instance. You must statically map port numbers on a container instance. However, an Application Load Balancer uses dynamic port mapping, so you can run multiple tasks from a single service on the same container instance.
  + If a service’s task fails the load balancer health check criteria, the task is stopped and restarted. This process continues until your service reaches the number of desired running tasks.
  + Services with tasks that use the awsvpc network mode, such as those with the Fargate launch type, do not support Classic Load Balancers. You must use NLB instead of TCP.



### Types of Deployments of ECS

1. Rolling updates with Amazon ECS

The rolling update deployment type uses Amazon ECS as the deployment controller. When you start a new service deployment, the Amazon ECS service scheduler replaces the currently running tasks with new tasks. The service’s deployment configuration determines the number of tasks that Amazon ECS adds or removes from the service during a rolling update. The minimumHealthyPercent and maximumPercent control the behavior of the deployment.

1. Blue/green deployments with AWS CodeDeploy

Blue/green deployments in Amazon ECS use CodeDeploy as the deployment controller. This deployment starts with two target groups for your load balancer:

* A blue group with tasks launched from the current version of the task definition
* A green group where CodeDeploy launches tasks using the new version of the task definition

When the new version has been verified, CodeDeploy begins cutting over production traffic to the green target group.

Blue/green deployment types - There are three ways that Code Deploy can cut traﬃc over during a blue/green deployment:

* Canary deployment
* Linear deployment
* All-at-once deployment

## EC2 – Elastic Compute Cloud

EC2 spins up resizable server instances that can scale up and down quickly. An instance is a virtual server in the cloud. With Amazon EC2, you can set up and configure the operating system and applications that run on your instance. Its configuration at launch is a live copy of the Amazon Machine Image (AMI) that you specify when you launch the instance. EC2 has an extremely reduced time frame for provisioning and booting new instances and EC2 ensures that you pay as you go, pay for what you use, pay less as you use more, and pay even less when you reserve capacity. When your EC2 instance is running, you are charged on CPU, memory, storage, and networking. When it is stopped, you are only charged for EBS storage.

You can launch different types of instances from a single AMI. An instance type essentially determines the hardware of the host computer used for your instance. Each instance type offers different compute and memory capabilities. You should select an instance type based on the amount of memory and computing power that you need for the application or software that you plan to run on top of the instance.

* When you launch a new EC2 instance, EC2 attempts to place the instance in such a way that all of your VMs are spread out across different hardware to limit failure to a single location. You can use placement groups to influence the placement of a group of interdependent instances that meet the needs of your workload.
* When you launch an instance in Amazon EC2, you have the option of passing user data to the instance when the instance starts. This user data can be used to run common automated configuration tasks or scripts. For example, you can pass a bash script that ensures top is installed on the new EC2 host and is always active.

EC2 Instance Pricing:

* **On-Demand instances** are based on a fixed rate by the hour or second. As the name implies, you can start an On-Demand instance whenever you need one and can stop it when you no longer need it. There is no requirement for a long-term commitment.
* **Reserved instances** ensure that you keep exclusive use of an instance on 1 or 3 year contract terms. The long-term commitment provides significantly reduced discounts at the hourly rate.
* **Spot instances** take advantage of Amazon’s excess capacity and work in an interesting manner. In order to use them, you must financially bid for access. Because Spot instances are only available when Amazon has excess capacity, this option makes sense only if your app has flexible start and end times. You won’t be charged if your instance stops due to a price change (e.g., someone else just bid a higher price for the access) and so consequently your workload doesn’t complete. However, if you terminate the instance yourself you will be charged for any hour the instance ran. Spot instances are normally used in batch processing jobs.

Standard Reserved vs. Convertible Reserved vs. Scheduled Reserved:

* Standard Reserved Instances have inflexible reservations that are discounted at 75% off of On-Demand instances. Standard Reserved Instances cannot be moved between regions. You can choose if a Reserved Instance applies to either a specific Availability Zone, or an Entire Region, but you cannot change the region.
* Convertible Reserved Instances are instances that are discounted at 54% off of On-Demand instances, but you can also modify the instance type at any point. For example, you suspect that after a few months your VM might need to change from general purpose to memory optimized, but you aren't sure just yet. So if you think that in the future you might need to change your VM type or upgrade your VMs capacity, choose Convertible Reserved Instances. There is no downgrading instance type with this option though.
* Scheduled Reserved Instances are reserved according to a specified timeline that you set. For example, you might use Scheduled Reserved Instances if you run education software that only needs to be available during school hours. This option allows you to better match your needed capacity with a recurring schedule so that you can save money.

**EC2 Placement groups**

1. Clustered Placement Groups: grouping of instances within single AZ, recommended for application with low network latency or n/w throughput. (Group Instances)
2. Spread Placement groups: group of instances which are placed in distinct underlying h/w, recommended for applications with small number of critical instances which need to be kept separated. (Individual instances)
3. Partitioned Placement group: divides each EC2 group into logical segment called partitions, each partition has its own set of racks, own n/w power source.

* Up to 7 partitions per AZ
* Can span across multiple AZs in the same region
* Up to 100s of EC2 instances
* The instances in a partition do not share racks with the instances in the other partitions
* A partition failure can affect many EC2 but won’t affect other partitions
* EC2 instances get access to the partition information as metadata

## Elastic Beanstalk

It is used to assist easy deploy and manage applications in AWS w/o worrying about the infra that runs those apps, simply upload the app and this will create capacity, LB, scaling and health monitoring.

Elastic Beanstalk is another way to script out your provisioning process by deploying existing applications to the cloud. Elastic Beanstalk is aimed toward developers who know very little about the cloud and want the simplest way of deploying their code.

* Just upload your application and Elastic Beanstalk will take care of the underlying infrastructure.
* Elastic Beanstalk has capacity provisioning, meaning you can use it with autoscaling from the get-go. Elastic Beanstalk applies updates to your application by having a duplicate ready with the already updated version. This duplicate is then swapped with the original. This is done as a preventative measure in case your updated application fails. If the app does fail, Elastic Beanstalk will switch back to the original copy with the older version and there will be no downtime experienced by the users who are using your application.

You can use Elastic Beanstalk to even host Docker as Elastic Beanstalk supports the deployment of web applications from containers. With Docker containers, you can define your own runtime environment, your own platform, programming language, and any application dependencies (such as package managers or tools) that aren't supported by other platforms. Elastic Beanstalk makes it easy to deploy Docker as Docker containers are already self-contained and include all the configuration information and software required to run.

# Storage Services

## AWS Backup

* Fully managed centralized backup service:
  + Policy based – backup plan, access policies, retention policy
  + Tag Based
  + Backup schedule
  + Incremental Backup
  + Cross region Backup
* Container to organize backups created from plans
* KMS encryption key can be defined and applied to backups
* Access policies can be assigned to role, user, group to restrict access to vault

Recovery Points

* Represent resource contents at a point in time
* Saved in Backup vaults, found in console or CLI

## EBS – Elastic Block Storage

An Amazon EBS volume is a durable, block-level storage device that you can attach to a single EC2 instance. You can think of EBS as a cloud-based virtual hard disk. You can use EBS volumes as primary storage for data that requires frequent updates, such as the system drive for an instance or storage for a database application. You can also use them for throughput-intensive applications that perform continuous disk scans.

* Amazon EBS provides the ability to create snapshots (backups) of any EBS volume and write a copy of the data in the volume to S3, where it is stored redundantly in multiple Availability Zones
* An EBS snapshot reflects the contents of the volume during a concrete instant in time.
* An image (AMI) is the same thing, but includes an operating system and a boot loader so it can be used to boot an instance.
* AMIs can also be thought of as pre-baked, launchable servers. AMIs are always used when launching an instance.

| Volume Types | Description | Use Cases | Volume Size | Max IOPS | Max Throughput |
| --- | --- | --- | --- | --- | --- |
| **EBS Provisioned IOPS SSD** | Highest performance SSD designed for latency-sensitive transactional workloads | I/O-intensive NoSQL and relational databases | 4 GB– 16 TB | 64,000 | 1,000 MB/s |
| **EBS General Purpose SSD** | General purpose SSD that balances price and performance for a wide variety of transactional workloads | Boot volumes, low-latency interactive apps, development, and test | 1 GB–16 TB | 16,000 | 250 MB/s |
| **Throughput Optimized HDD** | Low-cost HDD designed for frequently accessed, throughput-intensive workloads | Big data, data warehouses, log processing | 500 GB– 16 TB | 500 | 500 MB/s |
| **Cold HDD** | Lowest cost HDD designed for less frequently accessed workloads | Colder data requiring fewer scans per day | 500 GB– 16 TB | 250 | 250 MB/s |

**EBS Snapshot**

* EBS Snapshots are point in time copies of volumes. A snapshot is constrained to the region where it was created.
* Snapshots only capture the state of change from when the last snapshot was taken. This is what is recorded in each new snapshot, not the entire state of the server.
* Because of this, it may take some time for your first snapshot to be created. This is because the very first snapshot's change of state is the entire new volume. Only afterwards will the delta be captured because there will then be something previous to compare against.
* EBS snapshots occur asynchronously which means that a volume can be used as normal while a snapshot is taking place.
* When creating a snapshot for a future root device, it is considered best practices to stop the running instance where the original device is before taking the snapshot.
* The easiest way to move an EC2 instance and a volume to another availability zone is to take a snapshot.
* When creating an image from a snapshot, if you want to deploy a different volume type for the new image (e.g. General Purpose SSD -> Throughput Optimized HDD) then you must make sure that the virtualization for the new image is hardware-assisted.

**EBS Root Device Storage:**

* All AMI root volumes (where the EC2's OS is installed) are of two types: EBS-backed or Instance Store-backed
* When you delete an EC2 instance that was using an Instance Store-backed root volume, your root volume will also be deleted. Any additional or secondary volumes will persist however.
* If you use an EBS-backed root volume, the root volume will not be terminated with its EC2 instance when the instance is brought offline. EBS-backed volumes are not temporary storage devices like Instance Store-backed volumes.
* EBS-backed Volumes are launched from an AWS EBS snapshot, as the name implies
* Instance Store-backed Volumes are launched from an AWS S3 stored template. They are ephemeral, so be careful when shutting down an instance!
* Secondary instance stores for an instance-store backed root device must be installed during the original provisioning of the server. You cannot add more after the fact. However, you can add EBS volumes to the same instance after the server's creation.
* With these drawbacks of Instance Store volumes, why pick one? Because they have a very high IOPS rate. So while an Instance Store can't provide data persistence, it can provide much higher IOPS compared to network attached storage like EBS.

**EBS Encryption**

* EBS encryption offers a straight-forward encryption solution for EBS resources that doesn't require you to build, maintain, and secure your own key management infrastructure.
* It uses AWS Key Management Service (AWS KMS) customer master keys (CMK) when creating encrypted volumes and snapshots.
* You can encrypt both the root device and secondary volumes of an EC2 instance. When you create an encrypted EBS volume and attach it to a supported instance type, the following types of data are encrypted:
  + Data at rest inside the volume
  + All data moving between the volume and the instance
  + All snapshots created from the volume
  + All volumes created from those snapshots
* EBS encrypts your volume with a data key using the AES-256 algorithm.
* Snapshots of encrypted volumes are naturally encrypted as well. Volumes restored from encrypted snapshots are also encrypted. You can only share unencrypted snapshots.

RAID 5 and RAID 6 are not recommended for Amazon EBS because the parity write operations of these RAID modes consume some of the IOPS available to your volumes. Depending on the configuration of your RAID array, these RAID modes provide 20-30% fewer usable IOPS than a RAID 0 configuration.

**AMI in EBS (Instance Store)**

* Instance store volumes are sometimes called Ephemeral storage.
* Instance store volumes cannot be stopped. If it stops all data is looseds
* EBS backed instances can be stopped, No data will be lost in this
* By default, both root volumes will be deleted on termination. EBS volume can tell AWS to keep EBS Volume.
* EBS can be stored in 16 linux instances

## EFS – Elastic file storage

EFS provides a simple and fully managed elastic NFS file system for use within AWS. EFS automatically and instantly scales your file system storage capacity up or down as you add or remove files without disrupting your application.

* Simple scalable storage for file servers, for use with EC2 Instance
* Network attached Storage (NAS)
* Does not support public interface by default
* Can be accessed with multiple EC2 instance at the same time
* Fully managed service, Multi AZ replication, throughput scales automatically,
* Not available in all region, no cross-region capability
* Accessibility – Requires NFS, Mount File system to EBS Store, mount DNS Name to EC2
* EFS Security – IAM User permissions, EC2 security systems can be set to inbound rules, NACL can be used to control traffic, file Root only access by default
* Aside from EC2 instances, you can also mount EFS filesystems on ECS tasks, EKS pods, and Lambda functions.
* Multiple Amazon EC2 instances can access an EFS file system at the same time, providing a common data source for workloads and applications running on more than one instance or server.
* EFS file systems store data and metadata across multiple Availability Zones in an AWS Region.
* EFS file systems can grow to petabyte scale, drive high levels of throughput, and allow massively parallel access from EC2 instances to your data.
* EFS provides file system access semantics, such as strong data consistency and file locking.
* EFS enables you to control access to your file systems through Portable Operating System Interface (POSIX) permissions.
* Moving your EFS file data can be managed simply with AWS DataSync – a managed data transfer service that makes it faster and simpler to move data between on-premises storage and Amazon EFS.
* You can schedule automatic incremental backups of your EFS file system using the EFS-to-EFS Backup solution.
* Amazon EFS Infrequent Access (EFS IA) is a new storage class for Amazon EFS that is cost-optimized for files that are accessed less frequently. Customers can use EFS IA by creating a new file system and enabling Lifecycle Management. With Lifecycle Management enabled, EFS automatically will move files that have not been accessed for 30 days from the Standard storage class to the Infrequent Access storage class. To further lower your costs in exchange for durability, you can use the EFS IA-One Zone storage class.

**ENI Vs ENA Vs EFA**

ENI- Elastic Network Interface: essentially a virtual network card

An elastic network interface is a networking component that represents a virtual network card. When you provision a new instance, there will be an ENI attached automatically and you can create and configure additional network interfaces if desired. When you move a network interface from one instance to another, network traffic is redirected to the new instance.

* Allow primary private IPV4 address, IPv6, security groups
* Used for mgmt network, n/w and security appliances
* Low budget, high availability sol.
* ENI is used mainly for low-budget, high-availability network solutions, However, if you suspect you need high network throughput then you can use Enhanced Networking ENI.
* Enhanced Networking ENI uses single root I/O virtualization to provide high-performance networking capabilities on supported instance types. SR-IOV provides higher I/O and lower throughput and it ensures higher bandwidth, higher packet per second (PPS) performance, and consistently lower inter-instance latencies. SR-IOV does this by dedicating the interface to a single instance and effectively bypassing parts of the Hypervisor which allows for better performance.

ENA – Enhanced Networking, uses single IO virtualization to provide high performance

* Lower CPU utilization, high bandwidth, lower packer per second performance, no additional charge for enhanced n/w, good n/s performance
* 10Gbps to 100 GBPS for supported instance types
* ENA over VF to be selected for network related questions

EFA - Elastic fabric adapter: N/w device adapter to attach to EC2 for High performance computing for ML applications

* Provides lower and more consistent latency and higher throughput.
* Used for OS by passing
* EFAs accelerate the work required from the above use cases. EFA provides lower and more consistent latency and higher throughput than the TCP transport traditionally used in cloud-based High-Performance Computing systems.
* EFA can also use OS-bypass (on linux only) that will enable ML and HPC applications to interface with the Elastic Fabric Adaptor directly, rather than be normally routed to it through the OS. This gives it a huge performance increase.

Encrypted Root device volume: EBS on EC2

1. **Spot Fleet:**

* Collection of spot Instances, optionally, on -demand instances
* Attempts to launch the number of spot instances and on-demand instances.
* **Spot instances** can save upto 90% on cost of On-Demand
* You can block the spot instances from terminating using spot block

1. **EC2 Hibernate:**

* Hibernate the EC2 instance, OS is told to suspend to desk, saves the contents from instance RAM
* This helps to boot much faster, terminate long running process
* Root device needs to be encrypted
* Instance Ran must be less than 150 GB, hibernate cannot be there for more than 60 days

## S3 – Simple Storage Service

Amazon Simple Storage Service (Amazon S3) is an object storage service offering industry-leading scalability, data availability, security, and performance.

Cost-effective storage classes and easy-to-use management features, you can optimize costs, organize data, and configure fine-tuned access controls to meet specific business, organizational, and compliance requirements.

**Buckets**

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

**S3 Object Properties**

* Key: Name of the object
* Value: Sequence of bytes/contents of the object
* Versioning: Version of the object
* Metadata: Objects can further have metadata (data about data) to classify the contents

**S3 Storage Tiers**

S3 offers various storage tiers that help control cost, availability and durability of the data

* **S3 Standard**: This is the most costly alternative with 99.99% availability (probability that the object will be available when needed) and 99.999999999% durability (probability that data will not be lost)
* **S3 IA (Infrequently accessed)**: This still replicates data to different zones in a region but less costlier than standard. Retrieval fees are charged
* **S3 IA — 1 Zone**: Multiple zones replication is not done and costs even lesser than S3 IA
* **S3 Intelligent Tiering**: Storage tiering is managed by AWS instead which uses Machine learning to decide on which tier to use for the bucket based on historical patterns of usage
* **S3 Glacier**: Used when cost has to be less but time of retrieval can be configurable ranging from minutes to hours
* **S3 Glacier deep archive**: This is cheapest option where retrieval time of more than 12 hours is acceptable. This is used for data which might be rarely needed and time of retrieval being around ~12 hours is still acceptable

**Encryption**

* Server-Side Encryption using
  + Amazon S3-Managed Keys (SSE-S3)
  + AWS KMS-Managed Keys (SSE-KMS)
  + Customer-Provided Keys (SSE-C)
* Client-side Encryption using
  + AWS KMS-managed customer master key
  + client-side master key
* MFA Delete
  + MFA delete grants additional authentication for either of the following operations:
    - Change the versioning state of your bucket
    - Permanently delete an object version
  + MFA Delete requires two forms of authentication together:
    - Your security credentials
    - The concatenation of a valid serial number, a space, and the six-digit code displayed on an approved authentication device
* Cross-Account Access
  + You can provide another AWS account access to an object that is stored in an Amazon Simple Storage Service (Amazon S3) bucket. These are the methods on how to grant cross-account access to objects that are stored in your own Amazon S3 bucket:
    - Resource-based policies and AWS Identity and Access Management (IAM) policies for programmatic-only access to S3 bucket objects
    - Resource-based Access Control List (ACL) and IAM policies for programmatic-only access to S3 bucket objects
    - Cross-account IAM roles for programmatic and console access to S3 bucket objects
  + You can create cross-account access points using the S3 console or the AWS CLI.
  + Supports failover controls for S3 Multi-Region access points.
* Requester Pays Buckets
  + Bucket owners pay for all of the Amazon S3 storage and data transfer costs associated with their bucket. To save on costs, you can enable the Requester Pays feature so the requester will pay the cost of the request and the data download from the bucket instead of the bucket owner. Take note that the bucket owner always pays the cost of storing data.

**S3 Security Policies**

* Access to S3 object/bucket can be controlled with ACL control lists or Bucket Policies
* Bucket policies work at bucket level BUT Access Control Lists can go all the way down to individual objects
* Access logging can be configured for S3 buckets which logs all access requests for S3, made by different users
* Encryption in transit is achieved by HTTPS (SSL / TLS)
* Encryption at rest is achieved in two ways
  + Service Side encryption (Can be further managed by AWS in three ways)
* Client Side encryption — Client himself manages the encryption/decryption and uploads the encrypted data only

**S3 Encryption:**

S3 data can be encrypted both in transit and at rest.

1. **Encryption In Transit**: When the traffic passing between one endpoint to another is indecipherable. Anyone eavesdropping between server A and server B won’t be able to make sense of the information passing by. Encryption in transit for S3 is always achieved by SSL/TLS.
2. **Encryption At Rest:** When the immobile data sitting inside S3 is encrypted. If someone breaks into a server, they still won’t be able to access encrypted info within that server. Encryption at rest can be done either on the server-side or the client-side. The server-side is when S3 encrypts your data as it is being written to disk and decrypts it when you access it. The client-side is when you personally encrypt the object on your own and then upload it into S3 afterwards.

You can encrypt on the AWS supported server-side in the following ways:

* S3 Managed Keys / SSE - S3 (server side encryption S3 ) - when Amazon manages the encryption and decryption keys for you automatically. In this scenario, you concede a little control to Amazon in exchange for ease of use.
* AWS Key Management Service / SSE - KMS - when Amazon and you both manage the encryption and decryption keys together.
* Server Side Encryption w/ customer provided keys / SSE - C - when I give Amazon my own keys that I manage. In this scenario, you concede ease of use in exchange for more control.

**S3 Lifecycle Management**

* Automates the moving of objects between the different storage tiers.
* Can be used in conjunction with versioning.
* Lifecycle rules can be applied to both current and previous versions of an object.

**S3 Versioning**

* S3 versioning can be used to maintain multiple copies of objects
* Once enabled, it cannot be disabled on the same bucket
* When versioning is enabled, S3 stores all versions of an object including all writes and even deletes.
* It is a great feature for implicitly backing up content and for easy rollbacks in case of human error.
* It can be thought of as analogous to Git.
* Once versioning is enabled on a bucket, it cannot be disabled - only suspended.
* Versioning integrates w/ lifecycle rules so you can set rules to expire or migrate data based on their version.
* Versioning also has MFA delete capability to provide an additional layer of security.

**S3 Cross Region Replication:**

* Cross region replication only work if versioning is enabled.
* When cross region replication is enabled, no pre-existing data is transferred. Only new uploads into the original bucket are replicated. All subsequent updates are replicated.
* When you replicate the contents of one bucket to another, you can actually change the ownership of the content if you want. You can also change the storage tier of the new bucket with the replicated content.
* When files are deleted in the original bucket (via a delete marker as versioning prevents true deletions), those deletes are not replicated.
* Cross Region Replication Overview
* What is and isn’t replicated such as encrypted objects, deletes, items in glacier, etc.

**S3 Transfer Acceleration:**

* Transfer acceleration makes use of the CloudFront network by sending or receiving data at CDN points of presence (called edge locations) rather than slower uploads or downloads at the origin.
* This is accomplished by uploading to a distinct URL for the edge location instead of the bucket itself. This is then transferred over the AWS network backbone at a much faster speed.
* You can test transfer acceleration speed directly in comparison to regular uploads.

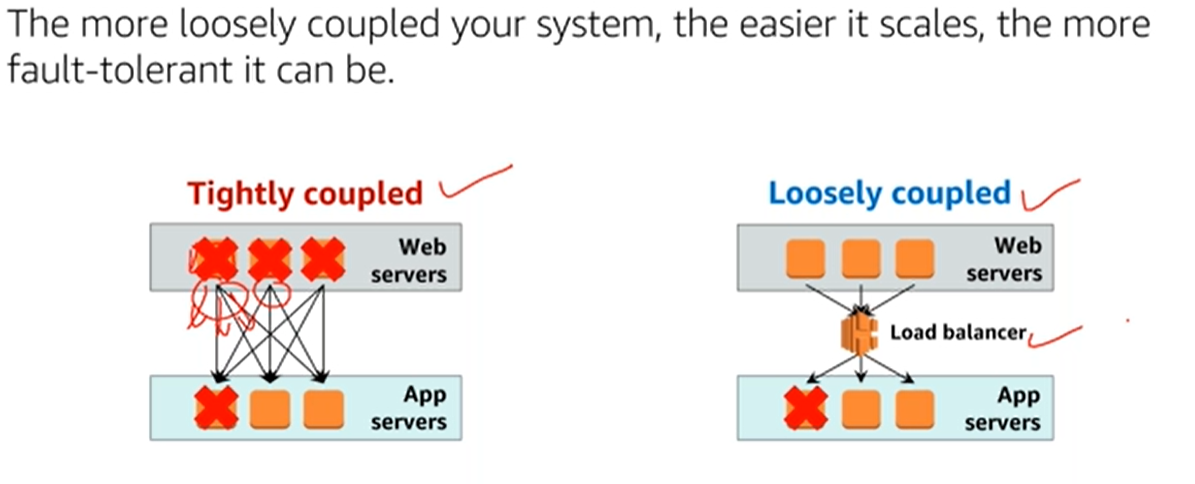
**S3 Event Notifications:**

The Amazon S3 notification feature enables you to receive and send notifications when certain events happen in your bucket. To enable notifications, you must first configure the events you want Amazon S3 to publish (new object added, old object deleted, etc.) and the destinations where you want Amazon S3 to send the event notifications. Amazon S3 supports the following destinations where it can publish events:

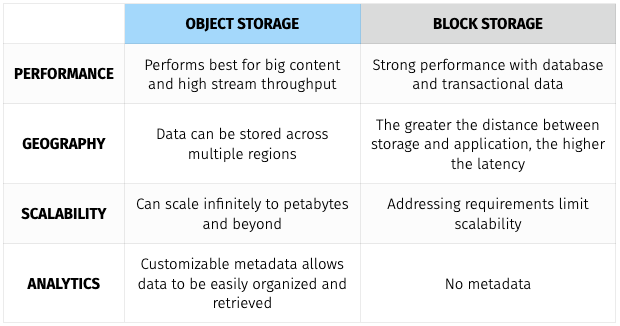
* Amazon Simple Notification Service (Amazon SNS) - A web service that coordinates and manages the delivery or sending of messages to subscribing endpoints or clients.
* Amazon Simple Queue Service (Amazon SQS) - SQS offers reliable and scalable hosted queues for storing messages as they travel between computers.
* AWS Lambda - AWS Lambda is a compute service where you can upload your code and the service can run the code on your behalf using the AWS infrastructure. You package up and upload your custom code to AWS Lambda when you create a Lambda function. The S3 event triggering the Lambda function also can serve as the code's input.

**S3 Cross Region Replication**

* This is used when you want to replicate the contents of a bucket automatically to another bucket (The destination bucket could also enforce different storage tiers on the objects)
* Versioning has to be mandatorily enabled on both source and destination buckets
* Delete markers OR object deletions are not replicated on the destination bucket. This is done intentionally by Amazon to inadvertently replicating deletion of objects
* Amazon S3 Object Lock prevents the deletion of objects and allows you to meet regulatory requirements, It can be enabled only on new buckets. It automatically enables versioning.



The following table highlights key differences between object and block storage:

[](https://user-images.githubusercontent.com/13093517/83915925-352c5780-a742-11ea-975b-53d4e5d07e7c.png)

* Data uploaded into S3 is spread across multiple files and facilities. The files uploaded into S3 have an upper-bound of 5TB per file and the number of files that can be uploaded is virtually limitless. S3 buckets, which contain all files, are named in a universal namespace so uniqueness is required. All successful uploads will return an HTTP 200 response.
* S3 Key Details:
* Objects (regular files or directories) are stored in S3 with a key, value, version ID, and metadata. They can also contain torrents and sub resources for access control lists which are basically permissions for the object itself.
* The data consistency model for S3 ensures immediate read access for new objects after the initial PUT requests. These new objects are introduced into AWS for the first time and thus do not need to be updated anywhere so they are available immediately.
* The data consistency model for S3 also ensures immediate read access for PUTS and DELETES of already existing objects, [since December 2020](https://aws.amazon.com/fr/about-aws/whats-new/2020/12/amazon-s3-now-delivers-strong-read-after-write-consistency-automatically-for-all-applications/).
* Amazon guarantees 99.999999999% (or 11 9s) durability for all S3 storage classes except its Reduced Redundancy Storage class.
* S3 comes with the following main features:
  + tiered storage and pricing variability
  + lifecycle management to expire older content
  + versioning for version control
  + encryption for privacy
  + MFA deletes to prevent accidental or malicious removal of content
  + access control lists & bucket policies to secure the data
* S3 charges by:
  + storage size
  + number of requests
  + storage management pricing (known as tiers)
  + data transfer pricing (objects leaving/entering AWS via the internet)
  + transfer acceleration (an optional speed increase for moving objects via Cloudfront)
  + cross region replication (more HA than offered by default)
  + Bucket policies secure data at the bucket level while access control lists secure data at the more granular object level.
  + By default, all newly created buckets are private.
  + S3 can be configured to create access logs which can be shipped into another bucket in the current account or even a separate account all together. This makes it easy to monitor who accesses what inside S3.
* There are 3 different ways to share S3 buckets across AWS accounts:
  + For programmatic access only, use IAM & Bucket Policies to share entire buckets
  + For programmatic access only, use ACLs & Bucket Policies to share objects
  + For access via the console & the terminal, use cross-account IAM roles
  + S3 is a great candidate for static website hosting. When you enable static website hosting for S3 you need both an index.html file and an error.html file. Static website hosting creates a website endpoint that can be accessed via the internet.
  + When you upload new files and have versioning enabled, they will not inherit the properties of the previous version.

[](https://user-images.githubusercontent.com/13093517/83919060-e1247180-a747-11ea-9336-e92ee163ac7a.png)

S3 and ElasticSearch:

* If you are using S3 to store log files, ElasticSearch provides full search capabilities for logs and can be used to search through data stored in an S3 bucket.
* You can integrate your ElasticSearch domain with S3 and Lambda. In this setup, any new logs received by S3 will trigger an event notification to Lambda, which in turn will then run your application code on the new log data. After your code finishes processing, the data will be streamed into your ElasticSearch domain and be available for observation.

Maximizing S3 Read/Write Performance:

* If the request rate for reading and writing objects to S3 is extremely high, you can use sequential date-based naming for your prefixes to improve performance. Earlier versions of the AWS Docs also suggested to use hash keys or random strings to prefix the object's name. In such cases, the partitions used to store the objects will be better distributed and therefore will allow better read/write performance on your objects.
* If your S3 data is receiving a high number of GET requests from users, you should consider using Amazon CloudFront for performance optimization. By integrating CloudFront with S3, you can distribute content via CloudFront's cache to your users for lower latency and a higher data transfer rate. This also has the added bonus of sending fewer direct requests to S3 which will reduce costs. For example, suppose that you have a few objects that are very popular. CloudFront fetches those objects from S3 and caches them. CloudFront can then serve future requests for the objects from its cache, reducing the total number of GET requests it sends to Amazon S3.

S3 Server Access Logging:

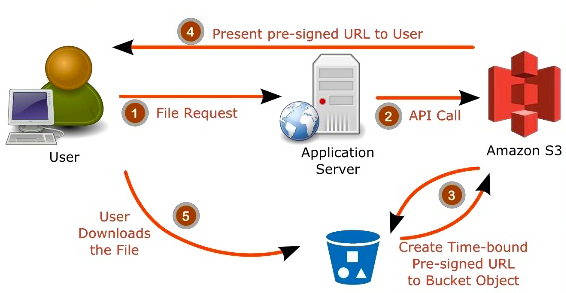
* Server access logging provides detailed records for the requests that are made to a bucket. Server access logs are useful for many applications. For example, access log information can be useful in security and access audits. It can also help you learn about your customer base and better understand your Amazon S3 bill.
* By default, logging is disabled. When logging is enabled, logs are saved to a bucket in the same AWS Region as the source bucket.
* Each access log record provides details about a single access request, such as the requester, bucket name, request time, request action, response status, and an error code, if relevant.
* It works in the following way:
  + S3 periodically collects access log records of the bucket you want to monitor
  + S3 then consolidates those records into log files
  + S3 finally uploads the log files to your secondary monitoring bucket as log objects

S3 Multipart Upload:

* Multipart upload allows you to upload a single object as a set of parts. Each part is a contiguous portion of the object's data. You can upload these object parts independently and in any order.
* Multipart uploads are recommended for files over 100 MB and is the only way to upload files over 5 GB. It achieves functionality by uploading your data in parallel to boost efficiency.
* If transmission of any part fails, you can retransmit that part without affecting other parts. After all parts of your object are uploaded, Amazon S3 assembles these parts and creates the object.
* Possible reasons for why you would want to use Multipart upload:
* Multipart upload delivers the ability to begin an upload before you know the final object size.
  + Multipart upload delivers improved throughput.
  + Multipart upload delivers the ability to pause and resume object uploads.
  + Multipart upload delivers quick recovery from network issues.
  + You can use an AWS SDK to upload an object in parts. Alternatively, you can perform the same action via the AWS CLI.
* You can also parallelize downloads from S3 using **byte-range fetches**. If there's a failure during the download, the failure is localized just to the specific byte range and not the whole object.

S3 Pre-signed URLs:

* All S3 objects are private by default, however the object owner of a private bucket with private objects can optionally share those objects without having to change the permissions of the bucket to be public.
* This is done by creating a pre-signed URL. Using your own security credentials, you can grant time-limited permission to download or view your private S3 objects.
* When you create a pre-signed URL for your S3 object, you must do the following:
  + Provide your security credentials.
  + Specify a bucket.
  + Specify an object key.
  + Specify the HTTP method (GET to download the object).
  + Specify the expiration date and time.
* The pre-signed URLs are valid only for the specified duration and anyone who receives the pre-signed URL within that duration can then access the object.
* The following diagram highlights how Pre-signed URLs work:

[](https://user-images.githubusercontent.com/13093517/84213482-c6773300-aa8e-11ea-84a1-3c17e14197bc.png)

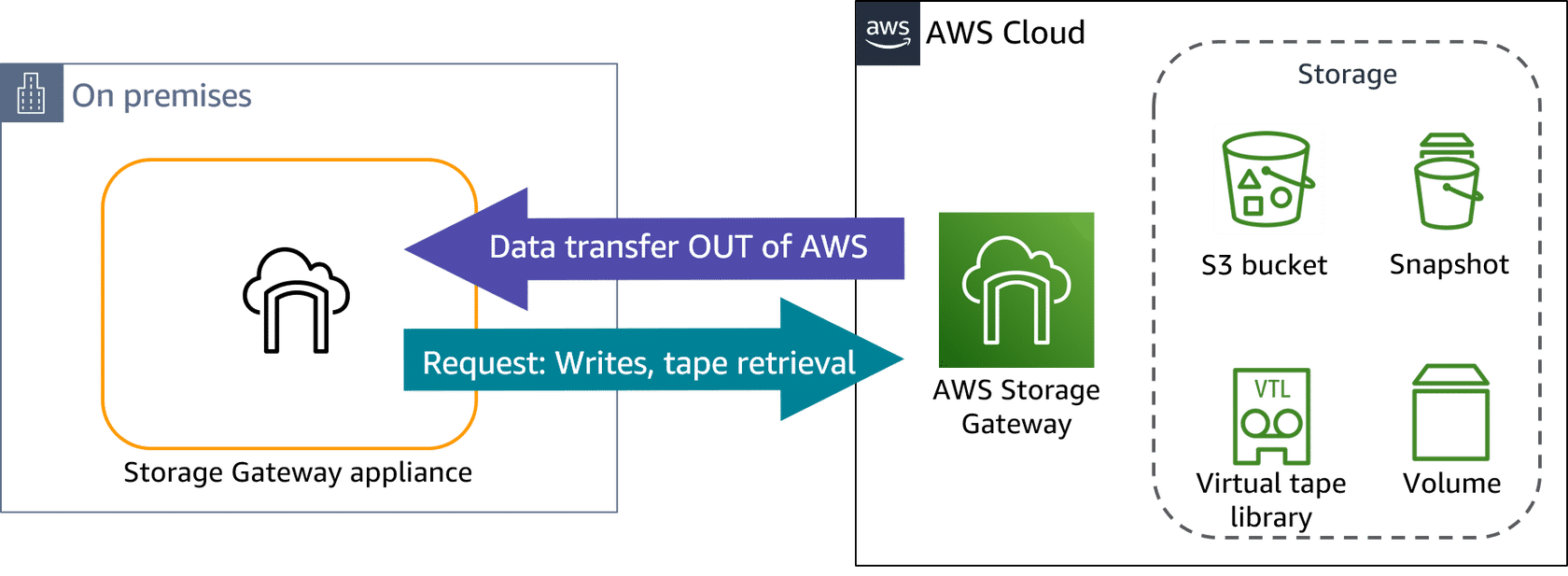
S3 Select:

* S3 Select is an Amazon S3 feature that is designed to pull out only the data you need from an object, which can dramatically improve the performance and reduce the cost of applications that need to access data in S3.
* Most applications have to retrieve the entire object and then filter out only the required data for further analysis. S3 Select enables applications to offload the heavy lifting of filtering and accessing data inside objects to the Amazon S3 service.
* As an example, let’s imagine you’re a developer at a large retailer and you need to analyze the weekly sales data from a single store, but the data for all 200 stores is saved in a new GZIP-ed CSV every day.
* Without S3 Select, you would need to download, decompress and process the entire CSV to get the data you needed.
* With S3 Select, you can use a simple SQL expression to return only the data from the store you’re interested in, instead of retrieving the entire object.
* By reducing the volume of data that has to be loaded and processed by your applications, S3 Select can improve the performance of most applications that frequently access data from S3 by up to 400% because you’re dealing with significantly less data.
* You can also use S3 Select for Glacier.

## Storage Gateway

Storage Gateway is a service that connects on-premise environments with cloud-based storage in order to seamlessly and securely integrate an on-prem application with a cloud storage backend. Storage Gateway comes in three flavors: File Gateway, Volume Gateway and Tape Gateway.

* Enable hybrid cloud storage between on-prem and AWS
* Virtual appliance connects directly to the infra as file server, as a disk vol
* All Storage Gateway types use standard storage protocols, local caching, and secure and optimized data transfers.
* S3 File Gateway deploys on premises and connects with a local cache so it can provide low-latency access to frequently accessed data.
* The Storage Gateway appliance provides your applications with low-latency access to data by maintaining a cache of recently written or read data. The cache is Least Recently Used (LRU) managed.
* As your applications write data to the AWS storage, the gateway first stores the data in the on-premises disks that are used for cache storage. Then the gateway uploads the data to AWS. The cache store acts as the on-premises durable store for data. If your application requests data, the gateway first checks the cache storage for the data, before checking AWS.
* Storage Gateway supports security features, access control, and security compliance certifications. Data is encrypted at transit and at rest. Your data at rest is encrypted by default using Amazon S3 server-side encryption (S3-SSE). Alternatively, Storage Gateway integrates with AWS KMS, so you can choose to encrypt your data using your own encryption keys. By integrating with IAM, you manage and secure access to your data
* S3 File Gateway provides access to objects in Amazon S3 as files using NFS or SMB protocols.
* **File gateway -** Operates via NFS or SMB and is used to store files in S3 over a network filesystem mount point in the supplied virtual machine. Simply put, you can think of a File Gateway as a file system mount on S3.
  + Interface to S3 with file protocols, Objects can be accessed direct to S3
* **Volume Gateway -** Operates via iSCSI and is used to store copies of hard disk drives or virtual hard disk drives in S3. These can be achieved via Stored Volumes or Cached Volumes. Simply put, you can think of Volume Gateway as a way of storing virtual hard disk drives in the cloud
  + iSCSI gateway access to S3, Access to S3 through EBS Snapshots to create new EBS vol
* **Tape gateway - Operates as a Virtual Tape Library**
  + iSCSI virtual tape library access to S3
  + Data can be archived to glacier
* With the Storage Gateway, you pay only for what you use. You're charged based on the amount of data transferred, the type and amount of storage that you use, and the requests you make. In addition, if you have deployed Storage Gateway using a hardware appliance, you also have the cost of the appliance.
* There are three elements to how you will be billed for Storage Gateway:
  + Storage: Fees are based on the type of storage that you use with your gateway (for example, Amazon S3 and Amazon EBS) and how it is configured.
  + Requests: Fees are based on data operations performed through the gateway including data ingest into AWS.
  + Data transfer: Fees are based on data transferred out of the Storage Gateway service and into your Storage Gateway appliance.
* These elements vary based on the Region, storage type, and gateway host specifics.



Storage Gateway provides a strategic, four-step approach to gateway creation, helping customers quickly deploy and implement the hybrid storage service

Deploying S3 Gateway appliance

* Choose from virtual or physical appliance options to deploy your on-premises gateway appliance:
* Download a **VM appliance** from the console and deploy it on one of the following:
  + VMware ESXi Hypervisor
    - Integrates with VMware HA
  + Microsoft Hyper-V
  + Linux Kernel-based Virtual Machine (KVM)
* You can choose to deploy your S3 File Gateway appliance in the cloud instead of on premises.
  + Deploy it as an **Amazon Machine Image (AMI)** in Amazon Elastic Compute Cloud (Amazon EC2). Storage Gateway provides an AMI that contains the gateway VM image.
  + Deploying a gateway on Amazon EC2 can be useful if you don't have an existing on-premises footprint and for learning how to set up and operate a Storage Gateway solution.
* All data transferred between the gateway and Amazon Web Services (AWS) storage is encrypted using Secure Sockets Layer (SSL). Data transfers are done through HTTPS. Objects are encrypted with Amazon S3 server-side encryption keys (SSE-S3) or optionally with AWS Key Management Service (AWS KMS) managed keys using SSE-KMS.
* To reduce data transfer overhead, the gateway uses multipart uploads and copy put, so only changed data in your files is uploaded to Amazon S3. Then, data that is already in the cloud is used to create a new version of the object.
* When a file is written to the file gateway by an NFS or SMB client, the file gateway uploads the file's data to Amazon S3 followed by its metadata (ownerships, timestamps, and so on).
* Uploading the file data creates an S3 object, and uploading the metadata for the file updates the metadata for the S3 object. This process creates another version of the object, resulting in two versions. If versioning is enabled for the S3 bucket, both versions of the object will be stored.
* The file upload notification is different than the Amazon S3 event notification. The file upload notification provides a notification for each individual file that is uploaded to Amazon S3 through S3 File Gateway. Amazon S3 event notifications provide notifications that include partial file uploads. Therefore, there is no way to tell from the Amazon S3 event notification that the file upload has been completed.

Security

* IAM is used to control who can sign in as an authorized user of an account (authenticated) and what they have permission to do (authorized). You manage access in AWS by creating policies with specific permissions and attaching them to IAM identities or resources. AWS deﬁnes several authentication identities. These include AWS account root user, IAM user, IAM group, and IAM role.
* So, if you want to control access to administrative actions on the S3 File Gateway, you create an IAM policy and attach it to an IAM role, or, you create an IAM policy and attach it to an IAM user or IAM group.
* Use multi-factor authentication (MFA) with each account.
* Use Secure Sockets Layer/Transport Layer Security (SSL/TLS) to communicate with AWS resources. TLS 1.2 or later is recommended.
* Set up API and user activity logging with AWS CloudTrail.
* Use AWS encryption solutions, along with all default security controls within AWS services.
* If you require Federal Information Processing Standards (FIPS) 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint.

A volume recovery point is a point in time in which all data in the volume is consistent and from which you can create a snapshot or clone a volume.

Data that is written to the Storage Gateway cache, but has not been uploaded to AWS, is referred to as dirty

## Amazon FSx for Lustre Simplified:

Amazon FSx for Lustre makes it easy and cost effective to launch and run the open source Lustre file system for high-performance computing applications. With FSx for Lustre, you can launch and run a file system that can process massive data sets at up to hundreds of gigabytes per second of throughput, millions of IOPS, and sub-millisecond latencies.

* FSx for Lustre is compatible with the most popular Linux-based AMIs, including Amazon Linux, Amazon Linux 2, Red Hat Enterprise Linux (RHEL), CentOS, SUSE Linux and Ubuntu.
* Since the Lustre file system is designed for high-performance computing workloads that typically run on compute clusters, choose EFS for normal Linux file system if your requirements don't match this use case.
* FSx Lustre has the ability to store and retrieve data directly on S3 on its own.

## Amazon FSx for Windows File Server

Amazon FSx for Windows File Server provides fully managed Microsoft Windows file servers, backed by a fully native Windows file system. FSx for Windows File Server has the features, performance, and compatibility to easily lift and shift enterprise applications to the AWS Cloud.

Amazon FSx supports a broad set of enterprise Windows workloads with fully managed file storage built on Microsoft Windows Server. Amazon FSx has native support for Windows file system features and for the industry-standard Server Message Block (SMB) protocol to access file storage over a network. Amazon FSx is optimized for enterprise applications in the AWS Cloud, with native Windows compatibility, enterprise performance and features, and consistent sub-millisecond latencies.

**Security and data protection**

Amazon FSx provides multiple levels of security and compliance to help ensure that your data is protected. It automatically encrypts data at rest (for both file systems and backups) using keys that you manage in AWS Key Management Service (AWS KMS). Data in transit is also automatically encrypted using SMB Kerberos session keys. It has been assessed to comply with ISO, PCI-DSS, and SOC certifications, and is HIPAA eligible.

Amazon FSx provides access control at the file and folder level with Windows access control lists (ACLs). It provides access control at the file system level using Amazon Virtual Private Cloud (Amazon VPC) security groups. In addition, it provides access control at the API level using AWS Identity and Access Management (IAM) access policies. Users accessing file systems are authenticated with Microsoft Active Directory. Amazon FSx integrates with AWS CloudTrail to monitor and log your API calls letting you see actions taken by users on your Amazon FSx resources.

## AWS Snowball

Snowball is a giant physical disk that is used for migrating high quantities of data into AWS. It is a peta-byte scale data transport solution. Using a large disk like Snowball helps to circumvent common large scale data transfer problems such as high network costs, long transfer times, and security concerns. Snowballs are extremely secure by design and once the data transfer is complete, the snowballs are wiped clean of your data.

AWS Snowball Edge Storage Optimized devices provide 24 vCPUs of compute capacity, coupled with 80 terabytes (TB) of usable block or Amazon S3-compatible object storage. It is well-suited for local storage and large-scale data transfers. In addition, you can order AWS Snowball Edge Storage Optimized devices for data transfer-only jobs, without compute capacity enabled, at a reduced cost.

* AWS Snowball Edge Compute Optimized devices provide 52 vCPUs, 42 TB of usable block or object storage. They include an optional GPU for use cases, such as advanced machine learning and full motion video analysis, in disconnected environments.
* Petabyte scale data transfer to S3, enforces encryption
* Control jobs with console, CLI or SDK
* Standalone snowball local client
* 50//80 TB models available
* Petabyte scale jobs possible with multiple devices

**Snowball Edge and Snowmobile:**

* Snowball Edge is a specific type of Snowball that comes with both compute *and* storage capabilities via AWS Lambda and specific EC2 instance types. This means you can run code within your snowball while your data is en route to an Amazon data center. This enables support of local workloads in remote or offline locations and as a result, Snowball Edge does not need to be limited to a data transfer service. An interesting use case is with airliners. Planes sometimes fly with snowball edges onboard so they can store large amounts of flight data and compute necessary functions for the plane’s own systems. Snowball Edges can also be clustered locally for even better performance.
* A type of Snowball device with on-board storage and compute power for select AWS capabilities. It can undertake local processing and edge-computing workloads in addition to transferring data between your local environment and the AWS Cloud.
* Has on-board S3-compatible storage and compute to support running Lambda functions and EC2 instances.
* Options for device configurations
  + Storage optimized – this option has the most storage capacity at up to 80 TB of useable storage space, 24 vCPUs, and 32 GiB of memory for compute functionality. You can transfer up to 100 TB with a single Snowball Edge Storage Optimized device.
  + Compute optimized – this option has the most compute functionality with 52 vCPUs, 208 GiB of memory, and 7.68 TB of dedicated NVMe SSD storage for instances. This option also comes with 42 TB of additional storage space.
  + Compute Optimized with GPU – identical to the compute optimized option, save for an installed GPU, equivalent to the one available in the P3 Amazon EC2 instance type.
* Snowmobile is an exabyte-scale data transfer solution. It is a data transport solution for 100 petabytes of data and is contained within a 45-foot shipping container hauled by a semi-truck. This massive transfer makes sense if you want to move your entire data center with years of data into the cloud.

# Database Services

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

## Aurora

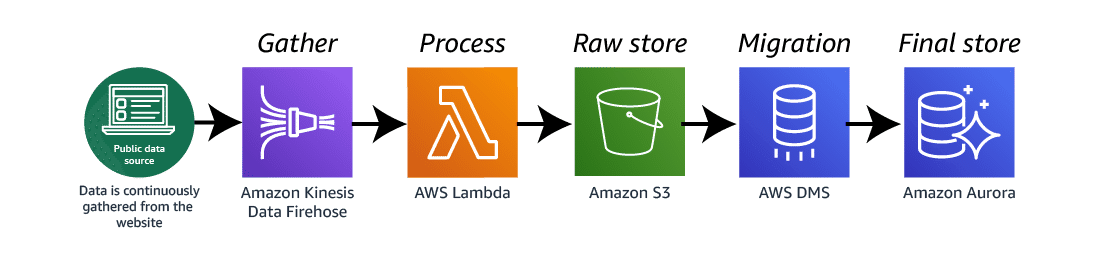
* Mysql and PostgreSQL compatible relational DB, cost effective, open source.
* On-demand and auto-scaling
* Simple storage option for infrequent, intermittent and unpredictable workloads
* Scale with 10GB to 64 TB with increments in 10 GB, 2 copies of data contained in each AZ, 3 AZ with 6 copies of data
* 3 types of Aurora replicas: Aurora Replica, MySQL Read Replica, PostgresQL
* Aurora Serverless provides a relative effective simple, cost effective for infrequent, intermittent, or unpredicted workloads.
* Compute resources can scale upto 32 vCPU cores and 244GB of RAM
* Aurora can automatically handle loss of 2 copies of data without affecting write capability and 3 copies of data without affecting read availability
* Storage for Aurora is self-healing -> Data blocks are continuously checked for errors and fixed
* Aurora automated backups or snapshots does not affect performance of running clusters

**Aurora Endpoints**

* Cluster endpoint – connects to the current primary DB instance for a DB cluster. This endpoint is the only one that can perform write operations. Each Aurora DB cluster has one cluster endpoint and one primary DB instance.
* Reader endpoint – connects to one of the available Aurora Replicas for that DB cluster. Each Aurora DB cluster has one reader endpoint. The reader endpoint provides load-balancing support for read-only connections to the DB cluster. Use the reader endpoint for read operations, such as queries. You can’t use the reader endpoint for write operations.
* Custom endpoint – represents a set of DB instances that you choose. When you connect to the endpoint, Aurora performs load balancing and chooses one of the instances in the group to handle the connection. You define which instances this endpoint refers to, and you decide what purpose the endpoint serves.
* Instance endpoint – connects to a specific DB instance within an Aurora cluster. The instance endpoint provides direct control over connections to the DB cluster. The main way that you use instance endpoints is to diagnose capacity or performance issues that affect one specific instance in an Aurora cluster.
* When you connect to an Aurora cluster, the host name and port that you specify point to an intermediate handler called an endpoint.

**Aurora Global Database**

* An Aurora global database spans multiple AWS Regions, enabling low latency global reads and disaster recovery from region-wide outages.
* Consists of one primary AWS Region where your data is mastered, and one read-only, secondary AWS Region.
* Aurora global databases use dedicated infrastructure to replicate your data.
* Aurora global databases introduce a higher level of failover capability than a default Aurora cluster.
* An Aurora cluster can recover in less than 1 minute even in the event of a complete regional outage. This provides your application with an effective Recovery Point Objective (RPO) of 5 seconds and a Recovery Time Objective (RTO) of less than 1 minute.
* Has managed planned failover capability, which lets you change which AWS Region hosts the primary cluster while preserving the physical topology of your global database and avoiding unnecessary application changes.

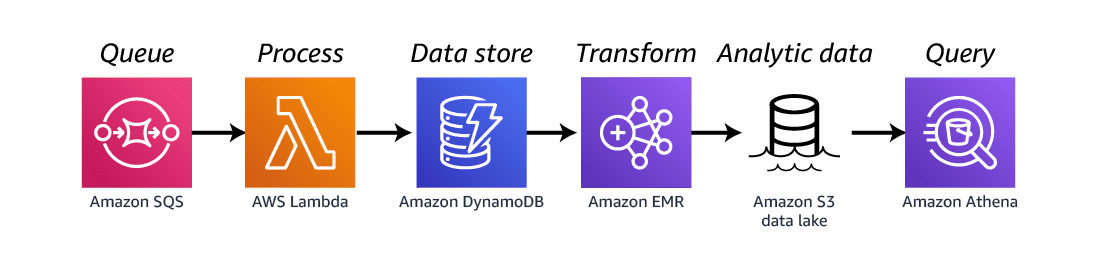


## **DynamoDB –** NoSQL support,

* Eventual Consistent Reads: consistent across all copies of data reached within one second, Best read performance
* Strongly consistent read: returns read result that returns all writes that received successful response prior to read, less than 1 sec response,
* Scales seamlessly through API and CLI calls
* Amazon DynamoDB allows for a flexible schema, so each item can have variation in the attributes outside of the primary and secondary key.
* DynamoDB accelerator (DAX): fully manageable, high available, in-memory cache, 10x performance, compatible with API calls, Its useful when there are lot of transactions happening between application and DynamoDB, to get read and write performance improvements.
* Point in time Recovery – accidental deletion, restore up to 35 days data, incremental backup in DynamoDB
* Managed Multi-Master, Multi region Replication, schema-less
* Fine-Grained Access Control for DynamoDB allows you to restrict access to some rows and columns in the table using IAM policies. It can be useful for: A multiplayer game table which stores High-scores in the table. You want to display users scores and their nicknames but probably not want to expose their emails.
* Globally distributed database, multi-region redundancy for DR, app latency under 1 sec, Eventual read consistency with maximum delay of 1s
* Relational databases are comprised of tables, records, and fields. DynamoDB is a non-relational database comprised of tables, items, and attributes.
* Incoming data transfer **IS NOT** charged if in a single region. If you cross regions, you will be charged at both ends of the transfer
* DynamoDB supports two types of secondary indexes:
  + Global secondary index — An index with a partition key and a sort key that can be different from those on the base table.
  + Local secondary index — An index that has the same partition key as the base table, but a different sort key.
* DynamoDB supports concepts of streams where any modification to existing record in the table is written out on a data stream which can be processed by compute capabilities like AWS Lambda. Lambda can then take decisions based on that event stream or send a SNS notification instead
* DynamoDB streams can also be configured to send out two copies of state (previous / current) with the primary key attribute to reflect on the actual change that has happened on the table data
* DynamoDB supports DAX, to cache responses and improve time from milliseconds to microseconds
* You can scale up or scale down your tables’ throughput capacity without downtime or performance degradation, and use the AWS Management Console to monitor resource utilization and performance metrics.
* Provides on-demand backup capability as well as enable point-in-time recovery for your DynamoDB tables. With point-in-time recovery, you can restore that table to any point in time during the last 35 days.
* Spot and Reserved represent types of Amazon EC2 instances that can be purchased but are not directly related to DynamoDB.

**DynamoDB components:**

* **Tables**– Similar to other database systems, DynamoDB stores data in tables. A table is a collection of data. For instance, you could have a table called People that you could use to store personal contact information about friends, family, or anyone else of interest. You could also have a Cars table to store information about vehicles that people drive.
* **Items**– Each table contains zero or more items. An item is a group of attributes that is uniquely identifiable among all the other items. In a People table, each item represents a person. In a Cars table, each item represents one vehicle. Items in DynamoDB are similar in many ways to rows, records, or tuples in other database systems. In DynamoDB, there is no limit to the number of items you can store in a table.
* **Attributes**– Each item is composed of one or more attributes. An attribute is a fundamental data element, something that does not need to be broken down any further. For example, an item in a People table might contain attributes called PersonID, LastName, FirstName, and so on. In a Department table, an item might have attributes such as DepartmentID, Name, Manager, and so on. Attributes in DynamoDB are similar in many ways to fields or columns in other database systems.



**SQL v/s NoSQL**

* SQL optimized for storage, NoSQL for Speed
* SQL is normalized, NoSQL is Denormalised
* SQL Data has joins and rules
* SQL is rich, NoSQL has simple queries and scans
* SQL scaled vertically, NoSQL horizontally
* SQL is suitable to OLAP (Online analytical Processing) while NoSQL is suitable for OLTP (Transactional processing)

## Elasticache

* Managed in-memory cache service
* Key value stores
* Ultra-fast access to cached data
* Reduces load on data, Multi-AZ capability
* Low Latency
* Web service that makes easy to deploy, operate and scale in-memory cache, Helps improves performance of web applications.
* ElastiCache is a web service that makes it easy to deploy, operate, and scale an in-memory cache in the cloud. The service improves the performance of web applications by allowing you to retrieve information from fast, managed, in-memory caches, instead of relying entirely on slower disk-based databases
* Amazon ElastiCache can be used to significantly improve latency and throughput for many read-heavy applications workloads (such as social networking, gaming media sharing and Q&A portals) or computing intensive workloads (such as a recommendation engine)
* Caching improves application performance by storing critical pieces of data in memory for low-latency access. Cached information may include the results of I/O-intensive database queries or the results of computationally-intensive calculations.
* ElastiCache is a remote caching service. A remote cache is also called a side cache. It is a separate instance dedicated to storing cache data in memory.
* 2 in-memory Cache Engines: Memcached and Reddis
* Memcache: Multi-thread performance,
  + Free and Open source project, High Performance, Distributed memory object caching system
  + Max data vol 4.7 TiB
  + Lost data cannot be recovered
  + Simple scaling by adding/removing nodes
  + Use for Simple data, No
* Redis: Advance data type, Ranking, sub capabilities, Multi AZ
  + In-memory data structure store, free and open source
  + Supports data structure as strings, Hashes, list, sets,
  + Key value size upto 512MB, data vol 3.5 TiB
  + Persistence; lost data can be recovered
  + Read replica
  + Very large command set available
  + Use for advanced data, auto sorting of data, pub/sub capabilities

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

**EMR:**

Big data processing, consists of master node, core node and task node, log configuration to s3 on 5 min internal from master node.

## RDS

Two types

1. Multi Az for DR
2. Read Replica for Performance

* Redshift is data warehousing solution
* You cannot login to VM on which RDS is hosted, Patching of RDS and DB is Amazon’s responsibility
* RDS is not serverless
* BackUps
  + Automated backup – Recover database from any points within a retention point, (1 to 35 days), AWS will choose recent data. Stored in S3 equivalent to RDS size,
  + DB Snapshot – User initiated, stored after you delete the RDS instance, restored version will be new RDS server with new DNS endpoint.
* Amazon RDS Proxy efficiently manages database connections. It sits between client applications (including lambdas) and RDS.
* When it comes to RDS, there are two kinds of backups:
  + automated backups
  + database snapshots
* **Automated backups** allow you to recover your database to any point in time within a retention period (between one and 35 days). Automated backups will take a full daily snapshot and will also store transaction logs throughout the day. When you perform a DB recovery, RDS will first choose the most recent daily backup and apply the relevant transaction logs from that day. Within the set retention period, this gives you the ability to do a point in time recovery down to the precise second. Automated backups are enabled by default. The backup data is stored freely up to the size of your actual database (so for every GB saved in RDS, that same amount will freely be stored in S3 up until the GB limit of the DB). Backups are taken within a defined window so latency might go up as storage I/O is suspended in order for the data to be backed up.
* **DB snapshots** are done manually by the administrator. A key different from automated backups is that they are retained even after the original RDS instance is terminated. With automated backups, the backed up data in S3 is wiped clean along with the RDS engine. This is why you are asked if you want to take a final snapshot of your DB when you go to delete it.
* When you go to restore a DB via automated backups or DB snapshots, the result is the provisioning of an entirely new RDS instance with its own DB endpoint in order to be reached.

### RDS Security:

* You can authenticate to your DB instance using IAM database authentication. IAM database authentication works with MySQL and PostgreSQL. With this authentication method, you don't need to use a password when you connect to a DB instance. Instead, you use an authentication token.
* An authentication token is a unique string that Amazon RDS generates on request. Authentication tokens have a lifetime of 15 minutes. You don't need to store user credentials in the database because authentication is managed externally using IAM.
* IAM database authentication provides the following benefits:
  + Network traffic to and from the database is encrypted using Secure Sockets Layer (SSL).
  + You can use IAM to centrally manage access to your database resources, instead of managing access individually on each DB instance.
  + For applications running on Amazon EC2, you can use profile credentials specific to your EC2 instance to access your database instead of a password, for greater security
* Encryption at rest is supported for all six flavors of DB for RDS. Encryption is done using the AWS KMS service. Once the RDS instance has encryption enabled, the data in the DB becomes encrypted as well as all backups (automated or snapshots) and read replicas.
* After your data is encrypted, Amazon RDS handles authentication of access and decryption of your data transparently with a minimal impact on performance. You don't need to modify your database client applications to use encryption.
* Amazon RDS encryption is currently available for all database engines and storage types. However, you need to ensure that the underlying instance type supports DB encryption.
* You can only enable encryption for an Amazon RDS DB instance when you create it, not after the DB instance is created and DB instances that are encrypted can't be modified to disable encryption.

### RDS Enhanced Monitoring:

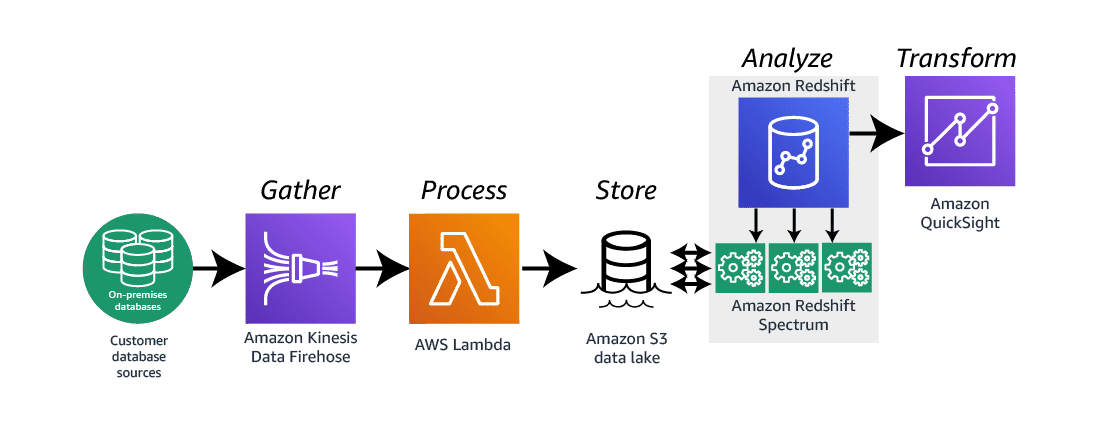
* RDS comes with an Enhanced Monitoring feature. Amazon RDS provides metrics in real time for the operating system (OS) that your DB instance runs on. You can view the metrics for your DB instance using the console, or consume the Enhanced Monitoring JSON output from CloudWatch Logs in a monitoring system of your choice.
* By default, Enhanced Monitoring metrics are stored in the CloudWatch Logs for 30 days. To modify the amount of time the metrics are stored in the CloudWatch Logs, change the retention for the RDS OS Metrics log group in the CloudWatch console.
* Take note that there are key differences between CloudWatch and Enhanced Monitoring Metrics. CloudWatch gathers metrics about CPU utilization from the hypervisor for a DB instance, and Enhanced Monitoring gathers its metrics from an agent on the instance. As a result, you might find differences between the measurements, because the hypervisor layer performs a small amount of work that can be picked up and interpreted as part of the metric.

**Connecting RDS to PostgreSQL database**

1. Create a custom VPC -> Security group,
2. After that we can create inbound rules for PostgreSQL with IPV4 and IPV6. Select outbound rule s and create
3. Create RDS Instance, standard create, select PostgreSQL instance and version 12.7, select the free tier
4. Provide name and password for SQL, select public access to yes, password authentication and create database with name and instance type.
5. We can get the end-point details from the database create (RDS)

## Redshift

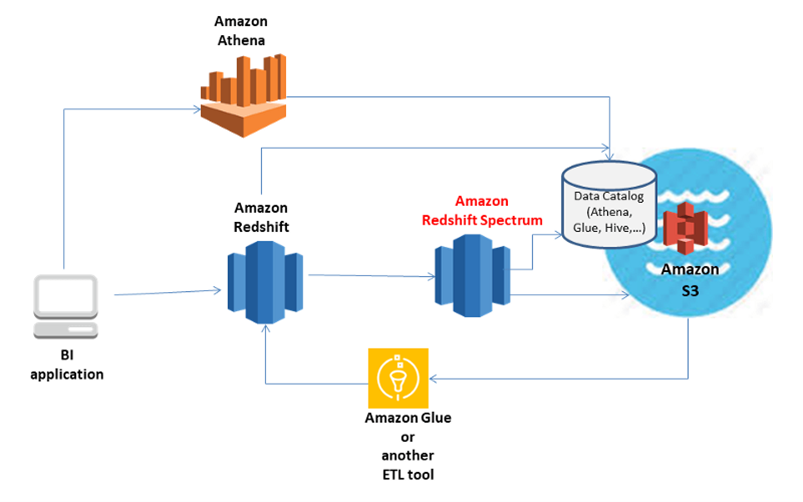
* Fast, powerful, petabyte scalable data warehouse service.
* OLAP vs OLTP (online analytics Processing, online transactional processing)
* Redshift can be configured as: Single Node, Multi node (Lead Node: manages the client connection and receives signal, Compute Node: Store data and perform queries and computation)
* Advanced compression, achieve significant compression relative to traditional data stores.
* Massive parallel processing (MPP): automatically distribute data and query data load to all nodes.
* Backups: 1-day retention period, maximum of 35 days, maintain at least 3 copies of your data, asynchronously replicate your snapshots to S3 in another region
* Available in only 1 AZ, used for BI
* 1/10th of the cost of other data warehousing solutions
* Helps with OLAP requirement — to derive analytics out of data
* Automated backups are by default done every day
* Redhisft can currently run only in OneAZ -> For same reason, Redshift offers asynchronous backup replication in S3 to another region for Disaster Recovery (DR)
* Cross region replication can be set up
* Amazon Redshift clusters are comprised of nodes. Compute nodes divide work among slices. Each slice is assigned a portion of the node’s memory and drive space. When you connect to an Amazon Redshift cluster, you use the SQL endpoint.
* There are only two types of nodes in Amazon Redshift: a single leader node and one or more compute nodes. Amazon Redshift cannot use OLE DB drivers.



### Redshift Spectrum

* Redshift Spectrum is a feature of the Amazon Redshift data warehouse. The spectrum allows for fast, complex, and efficient analysis of objects stored in the AWS cloud. The spectrum allows for a seamless analysis since it is directly embedded into the Amazons framework.
* Redshift Spectrum reduces the time and effort required to perform analysis on data as it eliminates the requirement to move the stored data from the storage service to a database as it can directly query data inside the s3 bucket.

### How does Redshift Spectrum work?

[](https://miro.medium.com/max/803/1*ciQyw5qQmc4RbLXMlNzcVA.png)

Redshift Spectrum divides user queries into filtered subsets that run concurrently. These requests are distributed across thousands of AWS-managed nodes to maintain query speed and consistent performance. Redshift Spectrum can be scaled to query over exabytes of data, and when S3 data is aggregated, it’s sent back to your on-premises Redshift cluster for final processing.

Redshift Spectrum requires a SQL client connected to your Redshift cluster. Multiple clusters can access the same S3 dataset at the same time, but you can only query data stored in the same AWS Region

### Getting started with Redshift Spectrum step by step

To get started using Amazon Redshift Spectrum, follow these steps:

* Step 1. Create an IAM role for Amazon Redshift
* Step 2: Associate the IAM role with your cluster
* Step 3: Create an external schema and an external table
* Step 4: Query your data in Amazon S3

## AWS DocumentDB

* Fully managed document database service designed to be fast, scalable, and highly available.
* Data is stored in JSON-like documents.
* Compatible with MongoDb.
* Flexible schema and indexing.
* Amazon DocumentDB provides flexible indexing, powerful ad-hoc queries, and analytics over collections of documents.
* Authentication is enabled by default and is done using the MongoDB tools and drivers. Data at rest is encrypted using keys you manage through AWS KMS, and data in transit is encrypted using TLS.
* Commonly used for content management, user profiles, and real-time big data

**Amazon DocumentDB Endpoints**

1. Cluster endpoint
   * Connects to cluster’s current primary instance.
   * Can be used for both read and write operations.
2. Reader endpoint
   * Connects to one of the available replicas of the cluster.
   * Use for read operations only.
   * If the cluster has more than one replica, the reader endpoint will direct each request to DocumentDB replicas.
3. Instance endpoint
   * Connects to a specific instance in the cluster.
   * Use for specialized workloads that will only affect specific replica instances.

## AWS Neptune

Amazon Neptune is a dynamic tool that should be implemented in situations where the data is not only highly connected but also requires queries that exploit this connected structure. There are many design, development, and performance benefits to using a graph database optimized for graph workloads.

* Neptune is made up of three main components: database instances, database replicas, and a database cluster volume.
* Neptune supports two popular graph query languages: Apache TinkerPop and SPARQL.
  + **Apache TinkerPop** is an open source computing framework for graph databases and graph analytic systems
  + **SPARQL** can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware.
* PL/SQL and HTSQL are both query languages, but they are not implemented by Neptune.
* Graph databases are not relational databases. Key-value and document databases are forms of non-relational databases; however, they differ from graph database technology.
* You can secure user access with IAM, secure connections with HTTPS, and encrypt data using 256-bit AES.

# Network

## Subnet

A subnet, or subnetwork, is a smaller network inside of a larger network.

* A subnet consists of a smaller portion of your IP address range. Subnets are how you add structure and functionality to your VPCs. Subnets make networks more efficient because network traffic can travel shorter distances without passing through unnecessary routers to reach its destination.
* Each subnet must reside entirely within one Availability Zone and cannot span zones. By launching instances in separate Availability Zones, you can protect your applications from the failure of a single zone. You use a public subnet for resources that must be connected to the internet and a private subnet for resources that won't be connected to the internet.
* A subnet in AWS is a smaller range of IP addresses taken from the larger VPC CIDR block.
* By default, Amazon Virtual Private Cloud (Amazon VPC) uses the IPv4 addressing protocol; you can't turn off this behavior. When you create a VPC, you must specify an IPv4 CIDR block (a range of private IPv4 addresses). You can optionally assign an IPv6 CIDR block to your VPC and assign IPv6 addresses from that block to instances in your subnets.

### Subnetting

Subnetting is the process of dividing a network into smaller network sections. Subnetting is done by borrowing bits from the host portion of the IP address.

* This is done with the use of a subnet mask. A subnet mask is a 32-bit address that helps separate the network address from host address. When you see /24 or /16 in an IP address, you know that the first 24 bits or the first 16 bits are the network.
* For example, in an IP address written like this: 10.0.1.4/24, the network is 10.0.1.x and the .4 is the host. The subnet mask /24 (slash 24) tells us that the first 24 bits of the address are for the network. Written out, the subnet mask looks like 255.255.255.0 with 8 bits for each of the first three sections being represented as 1s (8 bits times 3 sections: 8 x 3 = 24. 24 bits). Written out, it would look like 11111111.11111111.11111111.0.

An IP address of 10.0.1.4/16 means the following:

* 10.0 is the network portion.
* .1.4 is the host portion.
* The subnet mask is /16 and looks like 255.255.0.0.
* Written out it is 11111111.1111111.0.0.

An IP address of 10.0.1.4/26 means the following:

* 10.0.1 is the network portion.
* .4 (or any number between .1 and .64) is the host portion.
* The subnet mask is /26 and looks like 255.255.255.192.
* Written out it is 11111111.1111111.11111111.11000000.

CIDR

A CIDR block is a range of IP addresses. The IP addresses in the block are sequential (for example 1.1.1.1, 1.1.1.2, 1.1.1.3).

* The allowed block size is between a /28 netmask and /16 netmask.
* If you create more than one subnet in a VPC, the CIDR blocks of the subnets cannot overlap.
* For example, if you create a VPC with CIDR block 10.0.0.0/24, it supports 256 IP addresses. You can break this CIDR block into two subnets, each supporting 128 IP addresses.
* One subnet uses CIDR block 10.0.0.0/25 (for addresses 10.0.0.0–10.0.0.127).
* The other uses CIDR block 10.0.0.128/25 (for addresses 10.0.0.128–10.0.0.255)

## API Gateway

API Gateway is an entry-point for various types of resources acting as a front door entry mechanism with support for:

* HTTPS/TLS
* Invocation for resources like EC2, Lambda etc.
* Caching response at it’s own layer, thereby reducing requests to things like Lambda on every API invocation

API Gateway uses the following things to realize an API that can be exposed to the end-user

* A container which defines the API to be exposed
* Request types that are supported for the API container (GET, POST, OPTIONS etc.)
* URL Paths that have to be supported as part of the API (/main, /help, /register etc.)
* Destinations like Lambda, EC2 instances etc. which receive the request
* CORS (Cross origin resource sharing is needed to be enabled if requests can originate from various different sources) — FYI, CORS is always enforced by the client (like browsers)

API Gateway supports throttling API requests on global or API level and also supports caching by defining a fixed data size for storage to be provisioned. With caching enabled you can then avoid passing on redundant calls to the backend systems

## Cloudfront

* Cloudfront is a content delivery Network that is used to speed up distribution of static and dynamic contents to users.
* Cloudfront is a service that is used to store cached content at edge locations so that global users get it from their nearest location
* Cloudfront can monitor a S3 bucket, EBS load balancer, EC2 machine etc. and then it can just get new data and store it locally so that users can then download it directly from the Cloudfront distribution URL
* We can also invalidate cached objects in Cloudfront, but these invalidations are normally charged by AWS
* Objects are stored/cached by the edge locations until the TTL expires; after which a new version is then fetched again from the central server
* Edge locations can also be used to write data and not just READ data. Write scenario is used mostly in case of S3 Transfer acceleration where a user writes data to a local edge location and then AWS takes care of actually transferring the data to real S3 bucket
* Origins are the source from which cloudfront gets the data (S3, ELB, EC2 instance etc.)

CloudFront Key Details:

* When content is cached, it is done for a certain time limit called the Time To Live, or TTL, which is always in seconds
* If needed, CloudFront can serve up entire websites including dynamic, static, streaming and interactive content.
* Requests are always routed and cached in the nearest edge location for the user, thus propagating the CDN nodes and guaranteeing best performance for future requests.
* There are two different types of distributions:
  + **Web Distribution**: web sites, normal cached items, etc
  + **RTMP**: streaming content, adobe, etc
* Edge locations are not just read only. They can be written to which will then return the write value back to the origin.
* Cached content can be manually invalidated or cleared beyond the TTL, but this does incur a cost.
* You can invalidate the distribution of certain objects or entire directories so that content is loaded directly from the origin every time. Invalidating content is also helpful when debugging if content pulled from the origin seems correct, but pulling that same content from an edge location seems incorrect.
* You can set up a failover for the origin by creating an origin group with two origins inside. One origin will act as the primary and the other as the secondary. CloudFront will automatically switch between the two when the primary origin fails.
* Amazon CloudFront delivers your content from each edge location and offers a Dedicated IP Custom SSL feature. SNI Custom SSL works with most modern browsers.
* If you run PCI or HIPAA-compliant workloads and need to log usage data, you can do the following:
  + Enable CloudFront access logs.
  + Capture requests that are sent to the CloudFront API.
* An Origin Access Identity (OAI) is used for sharing private content via CloudFront. The OAI is a virtual user that will be used to give your CloudFront distribution permission to fetch a private object from your origin (e.g. S3 bucket).

CloudFront Signed URLs and Signed Cookies:

Signed URLs are recommended for Individual application or file downloads and Situations where cookies are not supported

Signed Cookies are recommended for Securing multiple files (You have a subscriber website) and situations where you do not want to change application URLs

* CloudFront signed URLs and signed cookies provide the same basic functionality: they allow you to control who can access your content. These features exist because many companies that distribute content via the internet want to restrict access to documents, business data, media streams, or content that is intended for selected users. As an example, users who have paid a fee should be able to access private content that users on the free tier shouldn't.
* If you want to serve private content through CloudFront and you're trying to decide whether to use signed URLs or signed cookies, consider the following:
  + Use signed URLs for the following cases:
    - You want to use an RTMP distribution. Signed cookies aren't supported for RTMP distributions.
    - You want to restrict access to individual files, for example, an installation download for your application.
    - Your users are using a client (for example, a custom HTTP client) that doesn't support cookies.
  + Use signed cookies for the following cases:
    - You want to provide access to multiple restricted files. For example, all of the files for a video in HLS format or all of the files in the paid users' area of a website.
    - You don't want to change your current URLs.

## Route 53

* Can be used to create HTTP website
* Its name comes from DNS which is on 53, hence the name route 53,
* DNS: Domain Name Server, convert to IP4 or Ip6 address.
* ELBs do not have pre-defined Ipv4 addresses, need to define them using DNS, Alias record over CName.
* 3 days to register the DNS buy from AWS.
* Simple Routing Policy: only 1 record with multiple IP address. Route 53 returns random value to user.
* Weighted routing policy: allow split your traffic based upon different weights.
* Latency based Routing: route based upon the latency of the traffic, latency resource record set for EC2 or ELB in each region, select the lowest latency in that region and then return resp. EC2.
* **Failover Routing Policy:** Used when active/passive set up is required, Route53 will monitor health check-up using Health checkup
* **Geolocation Routing Policy**: choose where your traffic will be sent based upon the geographic location of users.
* **Geo-proximity Routing:** lets route 53 route traffic based upon the geo location of the users, route more or less traffic to a given resource based upon value called bias.
* **Multi-value answer Routing:** configure Route53 to return multiple values, such as IP address to web servers, in response to DNS queries.

There are different types of records used in DNS system:

* SOA records
* A records
* CNAME records
* MX records
* PTR records
* Alias records
* NS records

A request from browser first goes to Top level domain (.com, .au, .gov etc), from there request is forwarded to Name Servers (NS), which fetch the details of A records and answer the request with respective IP address which can then be used by the browser to initiate a TCP connection. When given a choice between Alias record and a CNAME record, Alias record usually offers more benefits.

**Route 53 – Alias Records**

* Maps a hostname to an AWS resource
* An extension to DNS functionality
* Automatically recognizes changes in the resource’s IP addresses
* Unlike CNAME, it can be used for the top node of a DNS namespace (Zone Apex), e.g.: example.com
* Alias Record is always of type A/AAAA for AWS resources (IPv4 / IPv6)
* You can’t set the TTL

You cannot set an ALIAS record for an EC2 DNS name

## VPC – Virtual Private Cloud

Amazon VPC is the networking layer for Amazon EC2. The following are the key concepts for VPCs:

* **Virtual private cloud (VPC)** — A virtual network dedicated to your AWS account. Amazon VPC supports VPC between range 28 and 16 in size, subnets cannot be larger than the VPC in which they are.
* **Subnet** — A range of IP addresses in your VPC.
* **Route table** — A set of rules, called routes, that are used to determine where network traffic is directed.
* **Internet gateway** — A gateway that you attach to your VPC to enable communication between resources in your VPC and the internet.
* **VPC endpoint** — Enables you to privately connect your VPC to supported AWS services and VPC endpoint services powered by PrivateLink without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection.
* **CIDR block** —Classless Inter-Domain Routing. An internet protocol address allocation and route aggregation methodology.
* **Egress-only internet gateways**: A type of internet gateway that allows an EC2 instance in a subnet to access the internet but prevents resources on the internet from initiating communication with the instance.
* **Carrier gateways**: For subnets in Wavelength Zones, this type of gateway allows inbound traffic from a telecommunication carrier network in a specific location and outbound traffic to a telecommunication carrier network and the internet.
* **NAT gateway**: A managed AWS service that allows EC2 instances in private subnets to connect to the internet, other VPCs, or on-premises networks.
* **NAT instance**: An EC2 instance in a public subnet that allows instances in private subnets to connect to the internet, other VPCs, or on-premises networks.
* **DHCP options sets**: Configuration information (such as domain name and domain name server) passed to EC2 instances when they are launched into VPC subnets.
* **Prefix lists**: A collection of CIDR blocks that can be used to configure VPC security groups, VPC route tables, and AWS Transit Gateway route tables and can be shared with other AWS accounts using Resource Access Manager (RAM).
* **Network ACLs**: An optional layer of security for your VPC that acts as a firewall for controlling traffic in and out of your subnets.
* **Flow Logs:**  will store all the logs from different network components

Note:

* When you create VPC default Route table, NACL and Default Security group will be created
* No subnets will be created, no Internet gateway will be created
* AWS reserves 5 IP address within the subnets
* Only 1 internet gateway per VPC

1. Virtual private Network is the VPN Concentrator on the AWS side of connection, Each VPN has 2 sides of tunnel, each tunnel using a Virutal private gateway IP address
2. Route table need to be created in a VPC which will be pointing toward the Internet Gateway and will be associated to the subnet of instances that need public access, If these route table with subnets are not defined, a main route table will be created default with this which will not allow traffic, i.e., a private subnet will be created as default but we need to crate a public route table with subnet.

**VPC Endpoint:**

* Enables connectivity between VPC and supported AWS services, Instances in VPC does not requires Public PD address to communicate with each other.
* 2 types:
  + Interface endpoints: elastic n/w interface with private IP address that serves as entry point for traffic
  + Gateway endpoints: look like NAT Gateway

## AWS Direct Connect

* Connection from On-premise to AWS Cloud, Using direct connect make private connection from AWS to datacenter, office or location to reduce the n/w costs.
* Used for high throughput workloads i.e. (lots of n/w traffic)
* Stable or reliable n/w connection
* Provides a dedicated private connection from a remote network to your VPC
* Dedicated connection must be setup between your DC and AWS Direct Connect locations
* You need to setup a Virtual Private Gateway on your VPC
* Access public resources (S3) and private (EC2) on same connection
* Use Cases:
  + Increase bandwidth throughput - working with large data sets – lower cost
  + More consistent network experience - applications using real-time data feeds
  + Hybrid Environments (on prem + cloud)
* Supports both IPv4 and IPv6
* Steps to create Direct Console:
  + Create virtual interface in Direct connect
  + Go to VPC Console and then to VPC connection, create customer gateway and Virtual private gateway
  + Select VPN connection and create new VPN
  + Select VPN and customer gateway



* Dedicated Connections:
  + 1Gbps,10 Gbps and 100 Gbps capacity
  + Physical ethernet port dedicated to a customer
  + Request made to AWS first, then completed by AWS Direct Connect Partners
* Hosted Connections:
  + 50Mbps, 500 Mbps, to 10 Gbps capacity
  + Connection requests are made via AWS Direct Connect Partners
  + Capacity can be added or removed on demand
  + 1, 2, 5, 10 Gbps available at select AWS Direct Connect Partners

## AWS Global accelerator:

* Create accelerator to improve availability and performance of apps for local and global users. Directs traffic to optimal end points over AWS global n/w.
* Default 2 static IP address given by AWS Global accelerator.
* It assign each accelerator a DNS that points to global accelerator assigns to you.
* Network zone services the static IP address for your acceleration from Unique IP address to IP subnet.
* Listener Endpoint group is associated with every AWS region, include one or more end point

## Transit Gateway:

AWS Transit Gateway is a highly available and scalable service that provides interconnectivity between VPCs and your on-premises network. Within a Region, AWS Transit Gateway provides a method for consolidating and centrally managing routing between VPCs with a hub-and-spoke network architecture.

AWS Transit Gateway Network Manager provides a single global view of your private network. You can define the resources you want to monitor on the AWS Transit Gateway Network Manager dashboard. And you can visualize your network on a topology diagram or a geographical map. You can quickly access usage metrics and establish alerts for changes in the status of the resources you have registered. AWS Transit Gateway Network Manager also integrates with many software-defined networking in a wide area network (SD-WAN) AWS Partner offerings. So you can monitor your on-premises network resources from one centralized console.

* Allows to have transitive peering between VPCs and data center
* Works on hub and spoke model, regional basis.
* Use across multiple AWS account using RAM
* Use route tables to talk to one another, works with Direct connect and VPN connection.

## VPC Flow logs

* Feature that enables you to capture info about the IP traffic going to and from n/w interfaces in the VPC. Stored using cloudwatch logs.
* Can be created at 3 different levels:
  + VPC
  + Subnet
  + N/w interface level
* You need to create IAM roles while creating the VPC Flowlogs
* Cannot enable flow logs for VPC that are peered with VPC unless it is in your account, cant be done across accounts
* You can’t change the configuration of a flow log after creation
* Not all IP traffic is monitored;
  + DNS traffic
  + Windows instance
  + Traffic to and from 169.254.169.254 to instance metadata
  + DHCP traffic
  + Traffic to reserved default VPC traffic

**Bastion Hosts**

* Special purpose computer on a n/w specifically configured and designed to withstand attacks, single application hosts are deployed to reduce the chance of threat to computer.
* It is either in DMZ or outside the Firewall
* Securely administer EC2 instance, also called as Jump servers
* NAT Gateways cannot be used as Bastion Hosts

## NAT – Network access Translation

### NAT Instance and NAT Gateways

* While creating NAT, disable source/destination check on instance
* NAT instance must be in public subnet
* There must be a route out of private subnet to NAT Instance for it to work
* The amount of traffic flow from NAT instance depends upon the instance size
* High AZ can be created using autoscaling groups, multiple subnets in different AZs

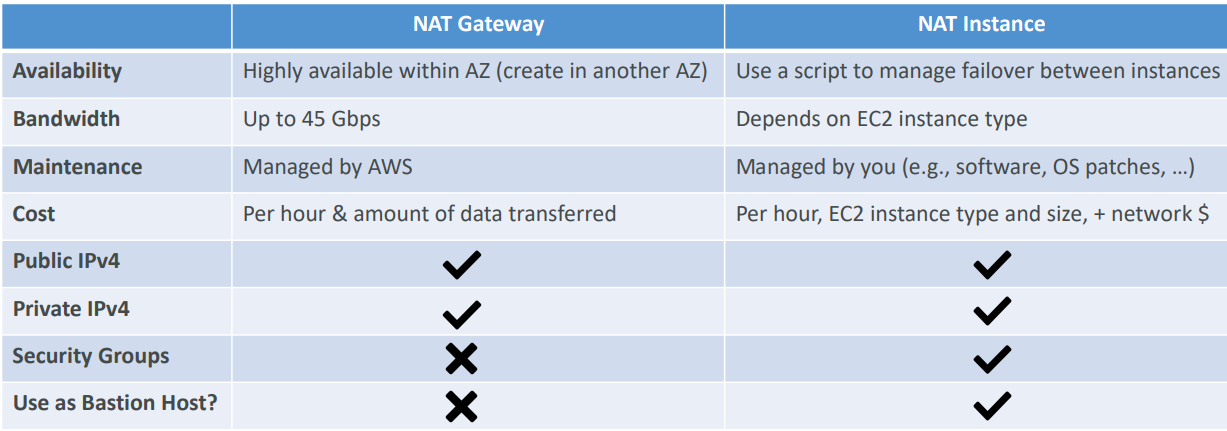
### Nat Gateways

* Redundant inside the AZ
* Need to add security Groups
* Automatically assigned to public IP Address
* Route tables need to be updated
* If you have resources in Multi-AZ and share 1 NAT Gateway, in event of NAT gateway’s AZ is down, resources in another AZ will lose the internet access.

Traffic pattern for Multi-tier architecture

Internet gateway -> Application Load Balancer -> Applications -> Data -> Applications -> network address translation (NAT) solution -> internet gateway

|  |  |  |
| --- | --- | --- |
| **Attribute** | **NAT gateway** | **NAT instance** |
| Availability | Highly available. NAT gateways in each Availability Zone are implemented with redundancy. Create a NAT gateway in each Availability Zone to ensure zone-independent architecture. | Use a script to manage failover between instances. |
| Bandwidth | Scale up to 100 Gbps. | Depends on the bandwidth of the instance type. |
| Maintenance | Managed by AWS. You do not need to perform any maintenance. | Managed by you, for example, by installing software updates or operating system patches on the instance. |
| Performance | Software is optimized for handling NAT traffic. | A generic AMI that's configured to perform NAT. |
| Cost | Charged depending on the number of NAT gateways you use, duration of usage, and amount of data that you send through the NAT gateways. | Charged depending on the number of NAT instances that you use, duration of usage, and instance type and size. |
| Type and size | Uniform offering; you don’t need to decide on the type or size. | Choose a suitable instance type and size, according to your predicted workload. |

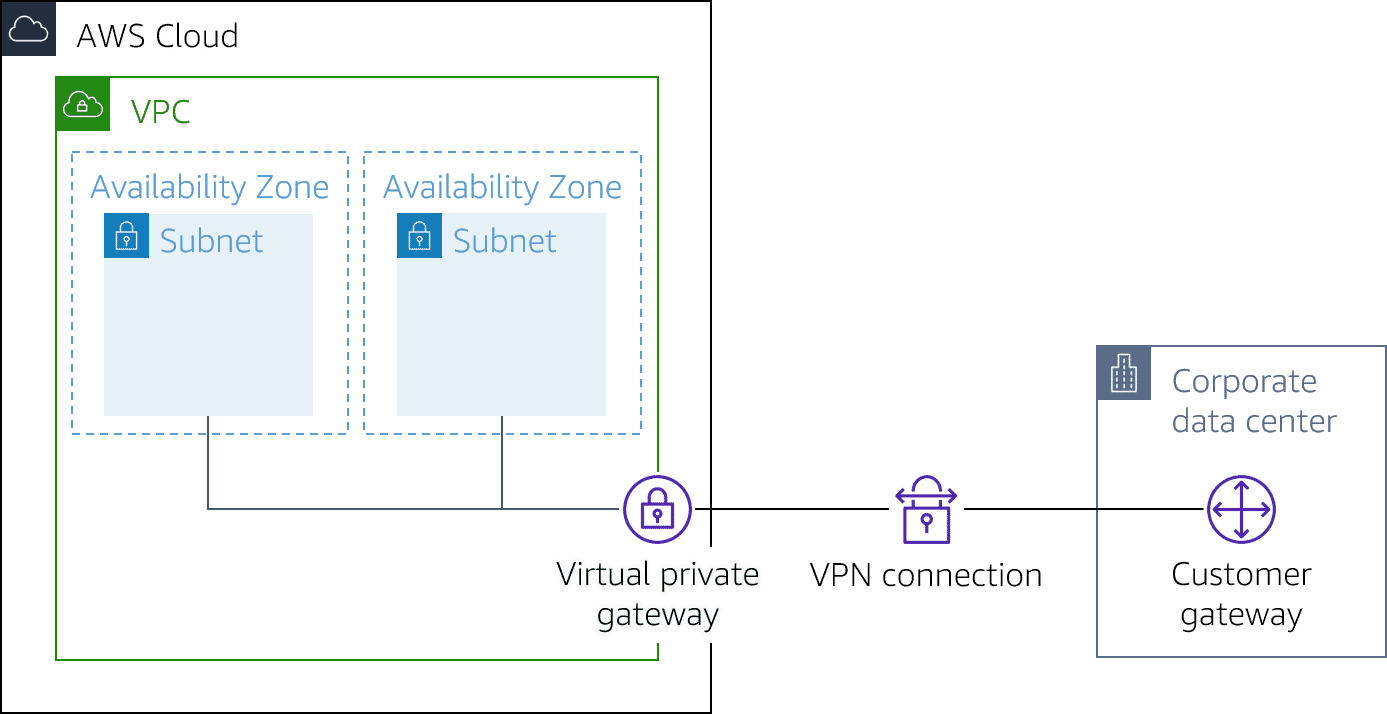


## AWS Site to Site VPN

AWS VPN is comprised of two services:

* AWS Site-to-Site VPN enables you to securely connect your on-premises network to Amazon VPC, for example your branch office site.
* AWS Client VPN enables you to securely connect users to AWS or on-premises networks, for example remote employees.

One AWS Site-to-Site VPN connection consists of two tunnels. Each tunnel terminates in a different Availability Zone on the AWS side, but it must terminate on the same customer gateway on the customer side



AWS Site-to-Site VPN limitations

* IPv6 traffic is partially supported. AWS Site-to-Site VPN supports IPv4/IPv6-Dualstack through separate tunnels for inner traffic. IPv6 for outer tunnel connection not supported.
* AWS Site-to-Site VPN does not support Path MTU Discovery. The greatest Maximum Transmission Unit (MTU) available on the inside tunnel interface is 1,399 bytes.
* Throughput of AWS Site-to-Site VPN connections is limited. When terminating on a virtual private gateway, only one tunnel out of the pair can be active and carry a maximum of 1.25 Gbps. However, real-life throughput will be about 1 Gbps. When terminating on AWS Transit Gateway, both tunnels in the pair can be active and carry an aggregate maximum of 2.5 Gbps. However, real-life throughput will be 2 Gbps. Each flow (for example, TCP stream) will still be limited to a maximum of 1.25 Gbps, with a real-life value of about 1 Gbps.
* Maximum packets per second (PPS) per VPN tunnel is 140,000.
* AWS Site-to-Site VPN terminating on AWS Transit Gateway supports equal-cost multi-path routing (ECMP) and multi-exit discriminator (MED) across tunnels in the same and different connection. ECMP is only supported for Site-to-Site VPN connections activated on an AWS Transit Gateway. MED is used to identify the primary tunnel for Site-to-Site VPN conncetions that use BGP. Note, BFD is not yet supported on AWS Site-to-Site VPN, though it is supported on Direct Connect.
* AWS Site-to-Site VPN endpoints use public IPv4 addresses and therefore require a public virtual interface to transport traffic over Direct Connect. Support for AWS Site-to-Site VPN over private Direct Connect is not yet available.
* For globally distributed applications, the accelerated Site-to-Site VPN option provides a connection to the global AWS backbone through AWS Global Accelerator. Because the Global Accelerator IP space is not announced over a Direct Connect public virtual interface, you cannot use accelerated Site-to-Site VPN with a Direct Connect public virtual interface.

## AWS Private Link

The service provider hosting the logging analytics SaaS service can create an endpoint service by deploying a Network Load Balancer in front of their service. You can connect to the Network Load Balancer using AWS PrivateLink. You create a VPC endpoint in each subnet. The subnets host instances that will send logs to the analytics service and use AWS PrivateLink to connect to the service provider's Network Load Balancer.

# Security

Security Components

## Security Group:

A security group is an AWS firewall solution that performs one primary function: to filter incoming and outgoing traffic from an EC2 instance. It accomplishes this filtering function at the TCP and IP layers, via their respective ports, and source/destination IP addresses.

* Every Security Group works in a similar fashion to a firewall as it carries a set of rules that filter traffic entering and leaving the EC2 instances
* AWS Security Groups have a set of rules that filter traffic in two ways: inbound and outbound. Since AWS security groups are assigned differently, you won’t be needing the same rules for both inbound and outbound traffic
* Security Group rules are based on ALLOWs and there is no concept of DENY when it comes to Security Groups. This means you cannot explicitly deny or blacklist specific ports via Security Groups, you can only implicitly deny them by excluding them in your ALLOWs list. Because of the above detail, everything is blocked by default. You must go in and intentionally allow access for certain ports.
* Security groups are specific to a single VPC, so you can't share a Security Group between multiple VPCs. However, you can copy a Security Group to create a new Security Group with the same rules in another VPC for the same AWS Account.
* Security Groups are regional and can span AZs, but can't be cross-regional.
* Outbound rules exist if you need to connect your server to a different service such as an API endpoint or a DB backend. You need to enable the ALLOW rule for the correct port though so that traffic can leave EC2 and enter the other AWS service.

## NACL, CloudFront, ALB, WAF

* NACL will run directly on the EC2 instance and will display the host IP address to Ec2 Instance.
* ALB will only allow the ALB SG Ips and will block the range of user IP if causing security threat, connection will terminate at ALB level
* Introducing WAF will add an additional layer of protection, IP blocking and filtering
* Through Cloudfront we can filter the client IP as only CloudFront IP gets directed through the NACL to ALB to EC2 instance, Attaching WAF provides an additional layer of protection to the network design

### Web Application Firewall (WAF)

AWS WAF is a web application that lets you allow or block the HTTP(s) requests that are bound for CloudFront, API Gateway, Application Load Balancers, EC2, and other Layer 7 entry points into your AWS environment. AWS WAF gives you control over how traffic reaches your applications by enabling you to create security rules that block common attack patterns, such as SQL injection or cross-site scripting, and rules that filter out specific traffic patterns that you can define. WAF's default rule-set addresses issues like the OWASP Top 10 security risks and is regularly updated whenever new vulnerabilities are discovered.

At the simplest level, AWS WAF lets you choose one of the following behaviors:

* Allow all requests except the ones that you specify: This is useful when you want CloudFront or an Application Load Balancer to serve content for a public website, but you also want to block requests from attackers.
* Block all requests except the ones that you specify: This is useful when you want to serve content for a restricted website whose users are readily identifiable by properties in web requests, such as the IP addresses that they use to browse to the website.
* Count the requests that match the properties that you specify: When you want to allow or block requests based on new properties in web requests, you first can configure AWS WAF to count the requests that match those properties without allowing or blocking those requests. This lets you confirm that you didn't accidentally configure AWS WAF to block all the traffic to your website. When you're confident that you specified the correct properties, you can change the behavior to allow or block requests.

web application firewall that helps protect your web applications or APIs against common web exploits and bots that may affect availability, compromise security, or consume excessive resources.

* Agile protection against web attacks
* Save time with managed rules
* Easily monitor, block, or rate-limit bots
* Ease of deployment & maintenance

Integrate WAF with Lambda to increase the security of the network

### Network access control List (NACL)

* Inbound rules – rule 100 for IPv4 and Rule 101 for IPv6 addresses.
* Increment of 100 is recommended by AWS
* Ephemeral Port: Short lived transport port for internal communication.
* VPC default comes with NACL, allow all inbound and outbound traffic default
* Creating ACLs, denies all inbound and outbound traffic by default
* Block IP address using NACLs
* Always rule with less number will be evaluated in order of execution,
* NACL are stateless, responses to inbound are subject to rules in outbound traffic

**Security Group VS NACL**

|  |  |
| --- | --- |
| Security Group | NACL |
| Operates at instances Level | Defines as Subnet Level |
| Supports allow rules only | Supports allow and Deny Rules, , Need to allow traffic with the allow rule, if not no traffic allowed |
| Stateful; return traffic is automatically allowed | Stateless; return traffic must be explicitly allowed by rules |
| Evaluates all rules before deciding to allow traffic | Process all rules in order when deciding which rule to allow traffic |
| Applies to instance only if someone specifies the SG | Applies to all instances in the subnets; no need for someone to apply the SG in Subnet |

### Load Balancers

**Custom VPC and ELBs**

* 3 types of LB – Application LB, N/w LB and Classic LB
* You need atleast 2 type of public subnets to create a Load Balancer
* Application Load balancer: best suited for load balancing of HTTP and HTTPs applications, operate at layer 7 and application aware (knowledge of applications and hence make intelligence operations)
* Network load balancers: load balancing of TCP traffic where extreme performance is required, operating in layer 4, capable of handling million of request per sec maintaining ultra low latency
* Classic load balancer: legacy elastic load balancer, load balance at HTTP/HTTPs and layer 7, layer 4, cheaper option. It can react to 504 (Application issue) error
* To correct the error in LB, you need to look out for X-Forwarded-For header to search IPv4 address
* Instances monitored by ELB are reported as inservice or Outservice
* LB have their own DNS Name, no IP address
* Elastic Load Balancer: automatically distributes incoming application traffic across multiple targets and virtual appliances in one or more Availability Zones (AZs).
  + SSL certificate must be deployed on ELB:
  + 1. Create with AWS Certificate Manager
  + 2. Upload certificate to ELB
  + Backend EC2 administration
    - Health checks – inservice, out of servers
    - ELB resumes routing to unhealthy instance once it has been restored to healthy
    - It can be used with an autoscaling group as well as default autoscaling check

#### Elastic Load Balancer - ELB

ELB Advanced Features

* To enable IPv6 DNS resolution, you need to create a second DNS resource record so that the ALIAS AAAA record resolves to the load balancer along with the IPv4 record.
* The X-Forwarded-For header, via the Proxy Protocol, is simply the idea for load balancers to forward the requester's IP address along with the actual request for information from the servers behind the LBs. Normally, the servers behind the LBs only see that the IP sending it traffic belongs to the Load Balancer. They usually have no idea about the true origin of the request as they only know about the computer (the LB) that asks them to do something. But sometimes we may want to route the original IP to the backend servers for specific use cases and have the LB’s IP address ignored. The X-Forwarded-For header makes this possible.
* Sticky Sessions bind a given user to a specific instance throughout the duration of their stay on the application or website. This means all of their interactions with the application will be directed to the same host each time. If you need local disk for your application to work, sticky sessions are great as users are guaranteed consistent access to the same ephemeral storage on a particular instance. The downside of sticky sessions is that, if done improperly, it can defeat the purpose of load balancing. All traffic could hypothetically be bound to the same instance instead of being evenly distributed.
* Path Patterns create a listener with rules to forward requests based on the URL path set within those user requests. This method, known as path-based routing, ensures that traffic can be specifically directed to multiple back-end services. For example, with Path Patterns you can route general requests to one target group and requests to render images to another target group. So the URL, “www.example.com/” will be forwarded to a server that is used for general content while “www.example.com/photos” will be forwarded to another server that renders images.

ELB Security

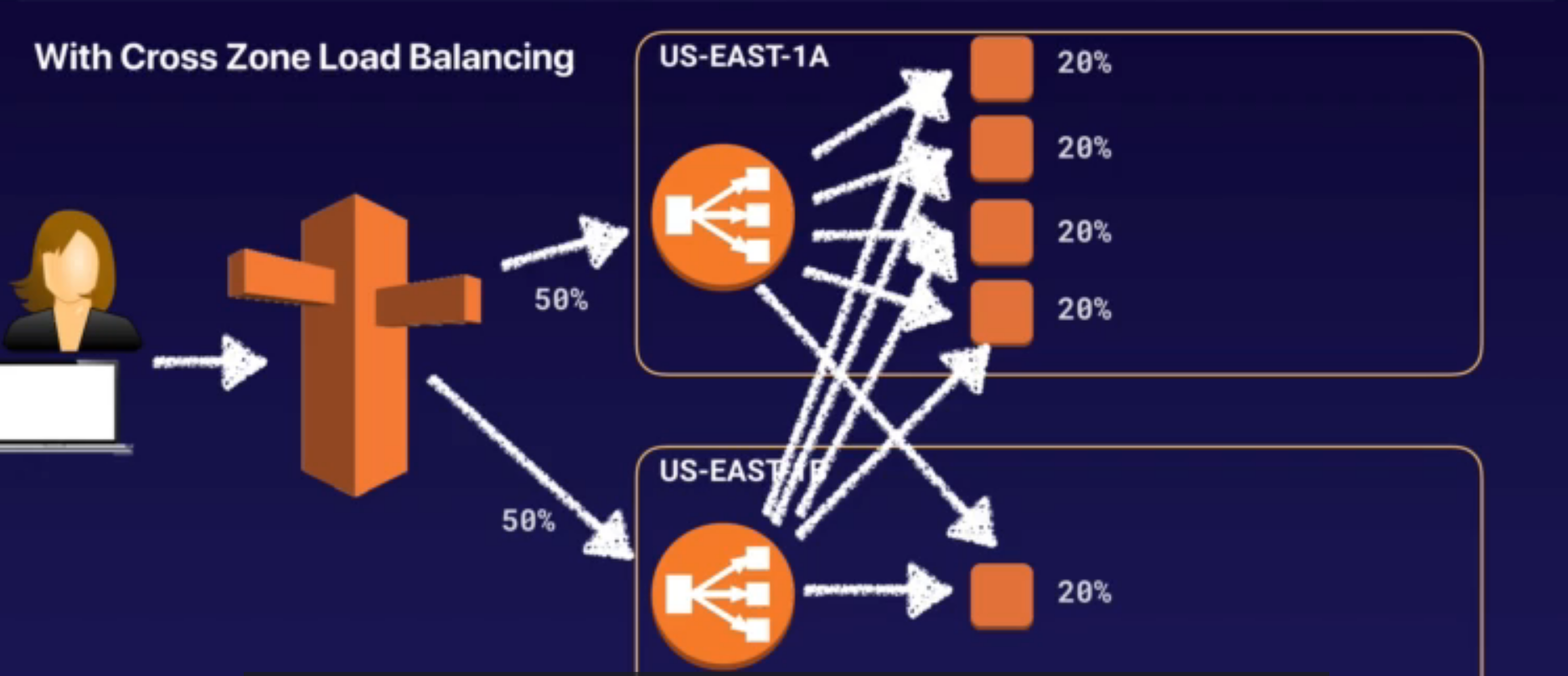
* ELB supports SSL/TLS & HTTPS termination. Termination at load balancer is desired because decryption is resource and CPU intensive. Putting the decryption burden on the load balancer enables the EC2 instances to spend their processing power on application tasks, which helps improve overall performance.
* Elastic Load Balancers (along with CloudFront) support Perfect Forward Secrecy. This is a feature that provides additional safeguards against the eavesdropping of encrypted data in transit through the use of a uniquely random session key. This is done by ensuring that the in-use part of an encryption system automatically and frequently changes the keys it uses to encrypt and decrypt information. So if this latest key is compromised at all, it will only expose a small portion of the user's recent data.
* Classic Load Balancers do not support Server Name Indication (SNI). SNI allows the server (the LB in this case) to safely host multiple TLS Certificates for multiple sites all under a single IP address (the Alias record or CName record in this case). To allow SNI, you have to use an Application Load Balancer instead or use it with a CloudFront web distribution.

Sticky Sessions:

* Classic load balancer routes each request independently to registered EC2 instance with smallest POD. Sticky session allows you to bind a user’s session to specific EC2 instance, ensuring all the request from the user are sent to specific same EC2 instance.
* This can be enabled for ALB, but traffic will be sent over to target group level rather than to EC2

#### Cross Zone Load Balancing

* Distribute the traffic from the load balancer across different AZ/region. This can help distribute the load to different EC2 spread across several AZ for users.
* Cross-zone load balancing reduces the need to maintain equivalent numbers of instances in each enabled Availability Zone, and improves your application's ability to handle the loss of one or more instances. However, we still recommend that you maintain approximately equivalent numbers of instances in each enabled Availability Zone for higher fault tolerance.



**Path Patterns:**

* A listener with rules to forward the request based upon URL path. This is path based routing.

#### Application Load Balancer - ALB

Here are some primary features of Application Load Balancer.

* ALB routes traffic based on request data. ALB makes routing decisions based on the HTTP protocol, like the URL path (/upload) and host, HTTP headers and method, and the source IP address of the client. This enables granular routing to target groups.
* ALB sends responses directly to the client. ALB has the ability to reply directly to the client with a fixed response, such as a custom HTML page. It can also send a redirect to the client, which is useful when you must redirect to a specific website or redirect a request from HTTP to HTTPS, removing that work from your backend servers.
* ALB uses TLS offloading. Speaking of HTTPS and saving work from backend servers, ALB understands HTTPS traffic. To pass HTTPS traffic through ALB, an SSL certificate is provided by either importing a certificate by way of IAM or AWS Certificate Manager (ACM) services, or by creating one for free using ACM. This ensures that the traffic between the client and ALB is encrypted.
* ALB authenticates users. On the topic of security, ALB can authenticate users before they are allowed to pass through the load balancer. ALB uses the OpenID Connect protocol and integrates with other AWS services to support protocols like SAML, LDAP, Microsoft Active Directory, and more.
* ALB secures traffic. To prevent traffic from reaching the load balancer, you configure a security group to specify the supported IP address ranges.
* ALB uses the round-robin routing algorithm. ALB ensures each server receives the same number of requests in general. This type of routing works for most applications.
* ALB uses the least outstanding request routing algorithm. If the requests to the backend vary in complexity where one request might need a lot more CPU time than another, then the least outstanding request algorithm is more appropriate. It’s also the right routing algorithm to use if the targets vary in processing capabilities. An outstanding request is when a request is sent to the backend server and a response hasn’t been received yet. For example, if the EC2 instances in a target group aren’t the same size, one server’s CPU utilization will be higher than the other if the same number of requests are sent to each server using the round-robin routing algorithm. That same server will have more outstanding requests as well. Using the least outstanding request routing algorithm would ensure an equal usage across targets.
* ALB uses sticky sessions. If requests must be sent to the same backend server because the application is stateful, use the sticky session feature. This feature uses an HTTP cookie to remember across connections which server to send the traffic to.
* Finally, ALB is specifically for HTTP and HTTPS traffic. If your application uses a different protocol, consider the Network Load Balancer.

#### Network Load Balancer - NLB

Here are some primary features of Network Load Balancer.

* Network Load Balancer supports TCP, UDP, and TLS protocols. HTTPS uses TCP and TLS as protocols. However, NLB operates at the connection layer, so it doesn’t understand what an HTTPS request is. That means all features that are required to understand the HTTP and HTTPS protocol, like routing rules based on that protocol, authentication, and least outstanding request routing algorithm, are not available with NLB.
* NLB uses a flow hash routing algorithm. The algorithm is based on:
  + Protocol
  + Source IP address and source port
  + Destination IP address and destination port
  + TCP sequence number
* If all of the parameters are the same, the packets are sent to the exact same target. If any of them are different in the next packets, the request might be sent to a different target.
* NLB has sticky sessions. Different from ALB, these sessions are based on the source IP address of the client, instead of a cookie.
* NLB supports TLS offloading. NLB understands the TLS protocol. It can also offload TLS from the backend servers, similar to how ALB works.
* NLB handles millions of requests per second. While ALB can also support this number of requests, it needs to scale to reach that number. This takes time. NLB can instantly handle millions of requests per second.
* NLB supports static and elastic IP addresses. In some situations, an application client needs to send requests directly to the load balancer IP address instead of using DNS. For example, this is useful if your application can’t use DNS or if the connecting clients require firewall rules based on IP addresses. In this case, NLB is the right type of load balancer to use.
* NLB preserves source IP address. NLB preserves the source IP address of the client when sending the traffic to the backend. With ALB, if you look at the source IP address of the requests, you will find the IP address of the load balancer. While with NLB, you would see the real IP address of the client, which is required by the backend application in some cases.

## AWS Cognito and Web Identity Federation

AWS Cognito builds upon two concepts:

* User pools — They handle user registration, authentication, password reset etc.
* Identity Pools — They are more aligned with authorization as they return the AWS credentials which can be temporarily used to assume a role, using which the user can then access various AWS resources

**Web Identity Federations**

When you build a mobile app for example, you cannot distribute AWS credentials along with the application code. When the application needs to access any AWS resource, it can instead generate a temporary AWS token which maps to a particular role and using that temporary token it accesses the specified resource. This avoids bundling any secure credential directly with the source code. For fetching an auth token, the app first authenticates the user against Google, Facebook, Amazon etc. or any other provider which support OIDC (Open ID Connect) connect capability.

So, Web Identity providers are to be used when user logs in against well-known services, whereas SAML based assertion is to be used, when user logs in against an IdP and the IdP further validates against AD. One benefit with IdP based SAML is that user can now access all applications using the same IdP based authentication which he had to do only once.

## Key Management System (KMS)

* Regional secure key management and encryption and decryption
* Manages customer master keys (CMKs), will not leave the region stored initially
* Ideal for s3 objects, database passwords and API keys stored in system manager Parameter store
* Encrypt and decrypt data upto 4 KB
* Pay per API Call, Audit capability using cloudtrail
* FIPS 140-2 level 2, FIPS 140-2 Level 3 – CloudHSM
* KMS is integrated with CloudTrail, which provides you the ability to audit who used which keys, on which resources, and when.
* Customer master keys (CMKs) are used to control access to data encryption keys that encrypt and decrypt your data.
* You can choose to have KMS automatically rotate master keys created within KMS once per year without the need to re-encrypt data that has already been encrypted with your master key.
* To help ensure that your keys and your data is highly available, KMS stores multiple copies of encrypted versions of your keys in systems that are designed for 99.999999999% durability.
* You can connect directly to AWS KMS through a private endpoint in your VPC instead of connecting over the Internet. When you use a VPC endpoint, communication between your VPC and AWS KMS is conducted entirely within the AWS network.
* You can define VPC Endpoint policies, enabling you to increase the granularity of your security controls by specifying which principals can access your endpoint, which API calls they can make, and which resources they can access.
* 3 types of CMK, Customer master Keys:
  + AWS Managed CMKs
  + Customer managed CMKs
  + AWS Owned CMKs
* Symmetric CMKs and Asymmetric CMKs difference-



There are three types of CMKs:

* 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.

**Importing Keys**

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

**Deleting Keys**

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

**CloudHSM**

* Dedicated hardware security model (HSM)
* FIPS 140-2 Level 3
* Manage you’re your keys, No access to AWS managed content
* Runs within the VPC
* Single tenant, Dedicated h/w, multi-AZ cluster
* Industry standard API, PKCS#11
* Java Cryptography Extensions (JCE), Microsoft CryptoNG (CNG)
* 1 HSM per subnet
* Offload the SSL/TLS processing for web servers
* Protect the private keys for an issuing certificate authority (CA)
* Enable transparent data encryption (TDE) for Oracle databases

**System Manager Parameter Store**

* Component of AWS System Manager (SSM)
* Secure Serverless storage for configuration and secrets: passwords, DB connections, License code, API keys
* Value can be encrypted using KMS
* Store parameters in Hierarchies and track versions
* Set TTL and enforce password expiration

### AWS EC2 System Manager

Amazon EC2 Simple Systems Manager (SSM) is an Amazon Web Services tool that allows an IT professional to automatically configure virtual servers in a cloud or in on-premises data center. An IT pro uses scripts, commands or the Elastic Compute Cloud (EC2) console to manage EC2 instances, virtual machines (VMs) or servers hosted on other clouds, or within local environments such as Windows.

Amazon EC2 Simple Systems Manager includes the EC2 Run Command feature, which an IT team uses as needed to execute commands for VMs, as well as an SSM Config feature that enables a team to set configuration preferences

### Lambda

* Compute service where you can upload code and create lambda functions, provisioning and managing servers that is used to run the code, It scales out automatically
* It is serverless, Independent service, 1 Lambda function can trigger other lambda functions
* Can be used to backup s3 buckets to other S3 buckets, RDS cannot trigger lambda
* Can be used for event driven compute service or can be used as a computer service to run code in response of HTTP service request.
* For serverless architecture we will be creating API Gateway at front end to serve the request, then Lambda functions to scale out automatically and then DB store like DynamoDB, aurora etc.
* 1 Million requests are free
* Lambda functions are short lived; the Lambda max timeout is 900 seconds (15 minutes). Lambda runs your code for a set amount of time before timing out. Timeout is the maximum amount of time in seconds that a Lambda function can run. The default value for this setting is 3 seconds, but you can adjust this in increments of 1 second up to a maximum value of 15 minutes

**Serverless Application Model (SAM)**

* Open source framework that allow to build serverless applications,
* CloudFormation extension optimized for serverless application
* Supports anything that cloud formation supports
* Run applications serverless

## Elastic Container Service (ECS)

* Container:
  + package that contains an app, libraries, run time and tools required to run, run on container engine like docker,
  + Provides isolation benefits of virtualization with less overhead and faster VMs
* Managed container orchestration service
* Create clusters to manage fleets of containers
* ECS manages EC2 or fargate instnaces
* Schedules containers for optimal placements
* Define rules for CPU and memory requirements
* Monitor resource utilization
* Free service
* ELB, Cloudwatch and cloudtrail
* 6 components:
  + Cluster: Logical collection of ECS resources
  + Task Definition: Defines your apps
  + Task: Single running copy of container defined by task definition
  + Container Definition: defines individual containers a task uses
  + Service: Allow task definitions to be scaled by adding tasks
  + Registry: Storage for container images
* Fargate:
  + Serverless container engine
  + Eliminates need to provision and manage servers
  + Specify and pay for resources per app
  + Works with both ECS and EKS
  + Isolation and security
* EKS
  + Elastic kubernetes service
  + K8 is open source s/w to deploy and manage containerized applications at scale
  + Containers are grouped in pods
  + It supports both ec2 and fargate
* ECR – Elastic Container Registry and Docker Hub ACR
  + Managed container docker registry
  + Store, manage and deploy images
  + Integrated with ECs and EKS
  + High available
  + Integrated with IAM
* ECA plus ELB
  + Distribute traffic across tasks
  + Supports ALB, NLB and CLB
  + Use ALB to support HTTP (Layer 7)
  + Use NLB and CLB to route TCP (layer 4)
  + ALB allows:
    - Dynamic port mapping
    - Path based routing
    - Priority rules
* Amazon ECS now supports Windows containers on container instances that are launched with the Amazon ECS-optimized Windows Server AMI and on AWS Fargate

## AWS Shield - Distributed Denial of Service (DDOS)

* DOS – attack to damage the availability of Site
* DDOS – DOS attacks from multiple sources, Manual intervention is difficult
* type: Infra/Application attack : simple, designed to attack network, block people to acces the network, attack the login screen
* Solve – Deploy WAF in the network

**AWS Shield**

* Provide first line of defence against DDOS, uses the Cloudfront and Route 53 availability protection
* Advanced – protect against the advanced DDoS attacks, Cost protection

Shield Tiers and Features

* Standard
  + All AWS customers benefit from the automatic protections of Shield Standard.
  + Shield Standard provides always-on network flow monitoring which inspects incoming traffic to AWS and detect malicious traffic in real-time.
  + Uses several techniques like deterministic packet filtering, and priority based traffic shaping to automatically mitigate attacks without impact to your applications.
  + When you use Shield Standard with CloudFront and Route 53, you receive comprehensive availability protection against all known infrastructure attacks.
  + You can also view all the events detected and mitigated by AWS Shield in your account.
* Advanced
  + Shield Advanced provides enhanced detection, inspecting network flows and also monitoring application layer traffic to your Elastic IP address, Elastic Load Balancing, CloudFront, or Route 53 resources.
  + It handles the majority of DDoS protection and mitigation responsibilities for layer 3, layer 4, and layer 7 attacks.
  + You have 24×7 access to the AWS DDoS Response Team. To contact the DDoS Response Team, customers will need the Enterprise or Business Support levels of AWS Premium Support.
  + It automatically provides additional mitigation capacity to protect against larger DDoS attacks. The DDoS Response Team also applies manual mitigations for more complex and sophisticated DDoS attacks.
  + It gives you complete visibility into DDoS attacks with near real-time notification via CloudWatch and detailed diagnostics on the “AWS WAF and AWS Shield” Management Console.
  + Shield Advanced comes with “DDoS cost protection”, a safeguard from scaling charges as a result of a DDoS attack that cause usage spikes on your AWS services. It does so by providing service credits for charges due to usage spikes.
  + It is available globally on all CloudFront and Route 53 edge locations.
  + With Shield Advanced you will be able to see the history of all incidents in the trailing 13 months.

## AWS Firewall Manager

* Central security management service
* Allow to manage WAF rules around different applications used in AWS,
* good for rapid response,
* Streamline applications compliance

**AWS OpsWork**

* Configurational Management platform
* Provides more control over infra design than Elastic beanstalk
* Infra as code using chef recipes for fine grained control
* Consist of CM model based upon stack

**AWS Fargate**

* Fully managed infrastructure and scaling
* Deploy container applications on serverless environment
* Individual tasks running on containers provide high secure isolation as compared to multiple containers on EC2
* Allocated right amount of compute resources
* Fargate own the infra, customer own and manages tasks

**AWS AppSync**

* Fully managed GraphQL server to connect with DynamoDB, Aurora, Elastic search, Lambda, HTTP Endpoint
  + GraphQL is an open-source data query and manipulation language for APIs, and a runtime for fulfilling queries with existing data
* Real time data subscription and synchronization
* Apollo client plugin to receive front ends and offline support

## AWS Identity and Access Management (IAM)

AWS Identity and Access Management (IAM) provides fine-grained access control across all of AWS. With IAM, you can specify who can access which services and resources, and under which conditions. With IAM policies, you manage permissions to your workforce and systems to ensure least-privilege permissions.

* IAM offers a centralized hub of control within AWS and integrates with all other AWS Services. IAM comes with the ability to share access at various levels of permission and it supports the ability to use identity federation (the process of delegating authentication to a trusted external party like Facebook or Google) for temporary or limited access.
* IAM comes with MFA support and allows you to set up custom password rotation policy across your entire organization.
* It is also PCI DSS compliant i.e. payment card industry data security standard. (passes government mandated credit card security regulations).
* unique account sign-in page URL: https://My\_AWS\_Account\_ID.signin.aws.amazon.com/console/
* You can use IAM tags to add custom attributes to an IAM user or role using a tag key–value pair.
* You can generate and download a credential report that lists all users on your AWS account. The report also shows the status of passwords, access keys, and MFA devices.

**Elements**

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

**IAM Entities:**

* **Users** - any individual end user such as an employee, system architect, CTO, etc.
* **Groups** - any collection of similar people with shared permissions such as system administrators, HR employees, finance teams, etc. Each user within their specified group will inherit the permissions set for the group.
* **Roles** - any software service that needs to be granted permissions to do its job, e.g- AWS Lambda needing write permissions to S3 or a fleet of EC2 instances needing read permissions from an RDS MySQL database.
* **Policies** - the documented rule sets that are applied to grant or limit access. In order for users, groups, or roles to properly set permissions, they use policies. Policies are written in JSON and you can either use custom policies for your specific needs or use the default policies set by AWS.
* IAM Policies are separated from the other entities above because they are not an IAM Identity. Instead, they are attached to IAM Identities so that the IAM Identity in question can perform its necessary function.

**IAM Key Details:**

* IAM is a global AWS services that is not limited by regions. Any user, group, role or policy is accessible globally. The root account with complete admin access is the account used to sign up for AWS. Therefore, the email address used to create the AWS account for use should probably be the official company email address.
* New users have no permissions when their accounts are first created. This is a secure way of delegating access as permissions must be intentionally granted. When joining the AWS ecosystem for the first time, new users are supplied an access key ID and a secret access key ID when you grant them programmatic access. These are created just once specifically for the new user to join, so if they are lost simply generate a new access key ID and a new secret access key ID. Access keys are only used for the AWS CLI and SDK so you cannot use them to access the console.
* When creating your AWS account, you may have an existing identity provider internal to your company that offers Single Sign On (SSO). If this is the case, it is useful, efficient, and entirely possible to reuse your existing identities on AWS. To do this, you let an IAM role be assumed by one of the Active Directories. This is because the IAM ID Federation feature allows an external service to have the ability to assume an IAM role.
* IAM Roles can be assigned to a service, such as an EC2 instance, prior to its first use/creation or after its been in used/created. You can change permissions as many times as you need. This can all be done by using both the AWS console and the AWS command line tools.
* You cannot nest IAM Groups. Individual IAM users can belong to multiple groups, but creating subgroups so that one IAM Group is embedded inside of another IAM Group is not possible.
* With IAM Policies, you can easily add tags that help define which resources are accessible by whom. These tags are then used to control access via a particular IAM policy. For example, production and development EC2 instances might be tagged as such. This would ensure that people who should only be able to access development instances cannot access production instances.

Priority Levels in IAM:

* **Explicit Deny:** Denies access to a particular resource and this ruling cannot be overruled.
* **Explicit Allow**: Allows access to a particular resource so long as there is not an associated Explicit Deny.
* **Default Deny** (or Implicit Deny): IAM identities start off with no resource access. Access instead must be granted.

Best Practices

* Lock Away Your AWS Account Root User Access Keys
* Create Individual IAM Users
* Use Groups to Assign Permissions to IAM Users
* Use AWS Defined Policies to Assign Permissions Whenever Possible
* Grant Least Privilege
* Use Access Levels to Review IAM Permissions
* Configure a Strong Password Policy for Your Users
* Enable MFA for Privileged Users
* Use Roles for Applications That Run on Amazon EC2 Instances
* Use Roles to Delegate Permissions
* Do Not Share Access Keys
* Rotate Credentials Regularly
* Remove Unnecessary Credentials
* Use Policy Conditions for Extra Security
* Monitor Activity in Your AWS Account

## Amazon GuardDuty

Amazon GuardDuty is a threat detection service that continuously monitors your AWS accounts and workloads for malicious activity and delivers detailed security findings for visibility and remediation

* Accurate, account-level threat detection
* Continuous monitoring across AWS accounts without added cost and complexity
* Threat detections developed and optimized for the cloud
* Threat severity levels for efficient prioritization
* Highly available threat detection
* Amazon GuardDuty provides three severity levels (Low, Medium, and High) to allow you to prioritize response to potential threats.
* CloudTrail Event Source
* GuardDuty analyzes CloudTrail management events and S3 data events
* 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.
* GuardDuty generates findings when it detects unexpected and potentially malicious activity in your AWS environment. These are viewable via Console, GuardDuty CLI or API operations.
* A Finding’s summary includes:
  + Finding type – a concise yet readable description of the potential security issue.
  + Severity – a finding’s assigned severity level of either High, Medium, or Low.
  + Region – the AWS region in which the finding was generated.
  + Count – the number of times GuardDuty generated the finding after you enabled GuardDuty in your AWS account.
  + Account ID – the ID of the AWS account in which the activity took place that prompted GuardDuty to generate this finding.
  + Resource ID – the ID of the AWS resource against which the activity took place that prompted GuardDuty to generate this finding.
  + Threat list name – the name of the threat list that includes the IP address or the domain name involved in the activity that prompted GuardDuty to generate the finding.
  + Last seen – the time (your local timezone if checked through console, and UTC if checked through CLI or API) at which the activity took place that prompted GuardDuty to generate this finding.
* A finding’s Resource affected section includes:
  + Resource role – a value that usually is set to Target because the affected resource can be a potential target of an attack.
  + Resource type – the type of the affected resource. This value is either AccessKey or Instance.
  + Instance ID – the ID of the EC2 instance involved in the activity that prompted GuardDuty to generate the finding.
  + Port – the port number for the connection used during the activity that prompted GuardDuty to generate the finding.
  + Access key ID – access key ID of the user engaged in the activity that prompted GuardDuty to generate the finding.
  + Principal ID – the principal ID of the user engaged in the activity that prompted GuardDuty to generate the finding.
  + User type – the type of user engaged in the activity that prompted GuardDuty to generate the finding.
  + User name – The name of the user engaged in the activity that prompted GuardDuty to generate the finding.

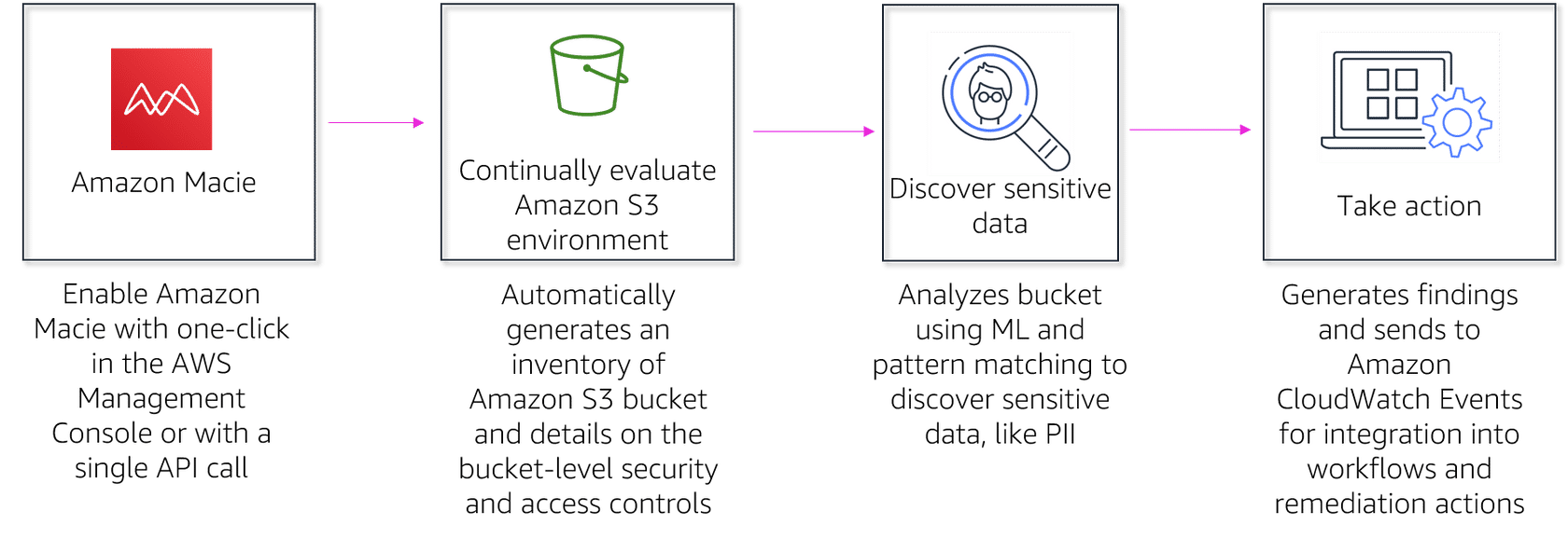
## Amazon Inspector

Amazon Inspector is an automated vulnerability management service that continually scans AWS workloads for software vulnerabilities and unintended network exposure. Amazon Inspector automatically discovers and scans Amazon EC2 instances and container images residing in Amazon Elastic Container Registry (Amazon ECR) for software vulnerabilities and unintended network exposure.

* Centrally manage multiple Amazon Inspector accounts
* Continuously scan your environment for vulnerabilities and network exposure
* Assess vulnerabilities accurately with the Amazon Inspector Risk score
* Identify high-impact findings with the Amazon Inspector dashboard
* 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.

## Amazon Macie

Amazon Macie is a fully managed data security and data privacy service that uses machine learning and pattern matching to discover and protect your sensitive data in AWS. As organizations manage growing volumes of data, identifying and protecting their sensitive data at scale can become increasingly complex, expensive, and time-consuming. Amazon Macie automates the discovery of sensitive data at scale and lowers the cost of protecting your data



* Identifying sensitive data in data migrations
* Maintaining regulatory compliance
* Assessing your data privacy and security
* 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

AWS STS – Security Token Service (STS)

AWS provides AWS Security Token Service (AWS STS) as a web service that enables you to request temporary, limited-privilege credentials for AWS Identity and Access Management (IAM) users or for users you authenticate (federated users).

## AWS System Manager

AWS Systems Manager provides a tool to help with automating operations with things like patch deployment, deploying automation scripts to your instances, and a lot more. Doing these sorts of tasks manually, one instance at a time, is like inviting the elusive creatures that cause you to get paged at 3 AM directly into your metaphorical server room. Through the use of Systems Manager, you can push patches and scripts to be run on an instance, or group of instances, in an automated fashion to avoid logging directly into a production box to make those changes.

## 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.

1. 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.
2. You can rotate secrets on a schedule or on demand by using the Secrets Manager console, AWS SDK, or AWS CLI.
3. Secrets Manager natively supports rotating credentials for databases hosted on Amazon RDS and Amazon DocumentDB and clusters hosted on Amazon Redshift.
4. 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.

**Security**

1. By default, Secrets Manager does not write or cache the secret to persistent storage.
2. By default, Secrets Manager only accepts requests from hosts that use the open standard Transport Layer Security (TLS) and Perfect Forward Secrecy.
3. You can control access to the secret using AWS Identity and Access Management (IAM) policies.
4. You can tag secrets individually and apply tag-based access controls.
5. You can configure VPC endpoints to keep traffic between your VPC and Secrets Manager within the AWS network.
6. Secrets Manager does not immediately delete secrets. Instead, Secrets Manager immediately makes the secrets inaccessible and scheduled for deletion after a recovery window of a minimum of seven days. Until the recovery window ends, you can recover a secret you previously deleted.
7. By using the CLI, you can delete a secret without a recovery window.

**Compliance**

* Secrets Manager is HIPAA, PCI DSS and ISO, SOC, FedRAMP, DoD SRG, IRAP, and OSPAR compliant.

## AWS Trusted Advisor

AWS Trusted Advisor provides recommendations that help you follow AWS best practices. Trusted Advisor evaluates your account by using checks. These checks identify ways to optimize your AWS infrastructure, improve security and performance, reduce costs, and monitor service quotas. You can then follow the recommendations to optimize your services and resources.

# Management Tools

## CloudWatch (PERFORMANCE)

* Monitoring services to monitor AWS resources, as well as applications on AWS
* Monitor performances of EC2, autoscaling, ELB, Route 53, Storage: EBS, Storage gateway, cloud formation
* CloudWatch collects monitoring and operational data in the form of logs, metrics, and events.
* You can use CloudWatch to detect anomalous behavior in your environments, set alarms, visualize logs and metrics side by side, take automated actions, troubleshoot issues, and discover insights to keep your applications running smoothly.
* Within the compute domain, CloudWatch can inform you about the health of EC2 instances, Autoscaling Groups, Elastic Load Balancers, and Route53 Health Checks. Within the storage and content delivery domains, CloudWatch can inform you about the health of EBS Volumes, Storage Gateways, and CloudFront.
* Host level metric : CPU, N/w memory
* Cloud watch alarms created on trigger.
* Cloud Watch Service Lens – visualize and analyze the health, Performance and availability of applications in single place, integrating AWS X-Ray with cloudwatch
* Container insights – metrics and logs specifically for containerized applications and microservices, available for ECS, EKS, K8
* Cloudwatch logs – Monitor, store and access log files from EC2, VPC, FLowlogs, Cloudtrail, API Gateway, ECS, EKS, Lambda; logs are indefinite and never expire; Retention setting can be used to changed these setting
* There are 3 states in the cloud watch alarms:
  + OK — The metric is below the threshold.
  + ALARM — When the metric crosses the threshold.
  + INSUFFICIENT\_DATA — Not enough data available for the metric to determine the alarm state
* Security: Logs can be encrypted using AWS KMS customer master key CMK; IAM policies can be used to manage resources; resource-based policies can be attached to log destination
* With regards to EC2, CloudWatch can only monitor host level metrics such as CPU, network, disk, and status checks for insights like the health of the underlying hypervisor.
* CloudWatch with EC2 will monitor events every 5 minutes by default, but you can have 1 minute intervals if you use Detailed Monitoring.

CloudWatch Logs:

* You can use Amazon CloudWatch Logs to monitor, store, and access your log files from Amazon EC2 instances, AWS CloudTrail, Amazon Route 53, and other sources. You can then retrieve the associated log data from CloudWatch Logs.
* It helps you centralize the logs from all of your systems, applications, and AWS services that you use, in a single, highly scalable service.
* You can create log groups so that you join logical units of CloudWatch Logs together.
* You can stream custom log files for further insights.

CloudWatch Events:

* Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.
* You can use events to trigger lambdas for example while using alarms to inform you that something went wrong.

CloudWatch Alarms:

* CloudWatch alarms send notifications or automatically make changes to the resources you are monitoring based on rules that you define.
* For example, you can create custom CloudWatch alarms which will trigger notifications such as surpassing a set billing threshold.
* CloudWatch alarms have two states of either ok or alarm

CloudWatch Metrics:

* CloudWatch Metrics represent a time-ordered set of data points.
* These basically are a variable you can monitor over time to help tell if everything is okay, e.g. Hourly CPU Utilization.
* CloudWatch Metrics allows you to track high resolution metrics at sub-minute intervals all the way down to per second.

CloudWatch Dashboards:

* CloudWatch dashboards are customizable home pages in the CloudWatch console that you can use to monitor your resources in a single view
* These dashboards integrate with CloudWatch Metrics and CloudWatch Alarms to create customized views of the metrics and alarms for your AWS resources.
* You can create cross-account cross-Region dashboards, which summarize your CloudWatch data from multiple AWS accounts and multiple Regions into one dashboard
* You can use Amazon **CloudWatch Synthetics** to create canaries, configurable scripts that run on a schedule, to monitor your endpoints and APIs. Canaries follow the **same routes and perform the same actions as a customer, which makes it possible for you to continually verify your customer experience even when you don't have any customer traffic on your applications**. By using canaries, you can discover issues before your customers do. Canaries are scripts written in Node.js or Python. They create Lambda functions in your account that use Node.js or Python as a framework. Canaries work over both HTTP and HTTPS protocols. Canaries offer programmatic access to a headless Google Chrome Browser via Puppeteer or Selenium Webdriver.

To test alarms and notifications, set the alarm state to Alarm using CLI

* aws cloudwatch set-alarm-state --alarm-name "myalarm" --state-value ALARM --state-reason "testing purposes"

## Cloud Trail: (Auditing)

AWS service that helps you enable governance, compliance, and operational and risk auditing of your AWS account.

* Increases the visibility into user and resource activity by recording AWS management console actions and API calls, Record: users actions, IP address, time for the calls made.
* Visibility into user and resource activity
* CloudTrail Events stores the last 90 days of events in its Event History. This is enabled by default and is no additional cost.This event history simplifies security analysis, resource change tracking, and troubleshooting.
* There are two types of events that can be logged in CloudTrail: management events and data events.
* Management events provide information about management operations that are performed on resources in your AWS account.
* By default, CloudTrail Events log files are encrypted using Amazon S3 server-side encryption (SSE). You can also choose to encrypt your log files with an AWS Key Management Service (AWS KMS) key. As these logs are stored in S3, you can define Amazon S3 lifecycle rules to archive or delete log files automatically

High Performances Computing (HPC):

How to achieve HPC on AWS

1. Data transfer – Snowmobile/snowball, AWS data sync (to store in EBS, S3), Direct Connect
2. Compute and N/w – EC2 instances or CPU optimizes, Spot Instances, Placement groups
3. Storage – Instance attached storage (S3, EFS, FSx)
4. Orchestration and Automation – AWS Batch, parallel cluster

**FSx**

* Amazon FSx is a file system offering from AWS. It is offered in two variants:
  + FSx for Windows
  + FSx for Lustre (High performance compute)
* FSx is basically a high performance file system that can be used for compute intensive workloads offering high data throughput. Users can additionally configure the throughput irrespective of the data storage size of the file system (unlike EFS).

**Amazon FSx for Windows Lustre**

* FSx is frequently used as file storage for Windows systems as it offers SMB protocol support. Additionally, it also offers integrations with other storage services like S3, where data can be temporarily copied from S3 to AWS FSx for high throughput needs from a filesystem perspective; and later the result can be copied back to S3 after the computations are completed.
* FSx for Windows also permits connectivity between on-premise servers and AWS so those same on-premise servers can make use of Amazon FSx too. You can use Microsoft Active Directory to authenticate into the file system.
* Amazon FSx for Windows provides multiple levels of security and compliance to help ensure your data is protected. Amazon FSx automatically encrypts your data at-rest and in-transit.

**Amazon FSx for Lustre Simplified**

Amazon FSx for Lustre makes it easy and cost effective to launch and run the open source Lustre file system for high-performance computing applications. With FSx for Lustre, you can launch and run a file system that can process massive data sets at up to hundreds of gigabytes per second of throughput, millions of IOPS, and sub-millisecond latencies.

* FSx for Lustre is compatible with the most popular Linux-based AMIs, including Amazon Linux, Amazon Linux 2, Red Hat Enterprise Linux (RHEL), CentOS, SUSE Linux and Ubuntu.
* Since the Lustre file system is designed for high-performance computing workloads that typically run on compute clusters, choose EFS for normal Linux file system if your requirements don't match this use case.

## Cloud Front

* Content – static and dynamic
* Origins – s3, ec2, ELB, on-premise
* Private content protection
* Security improvement
  + AWS Shield standard and advanced
  + AWS WAF

## Cloud Formation:

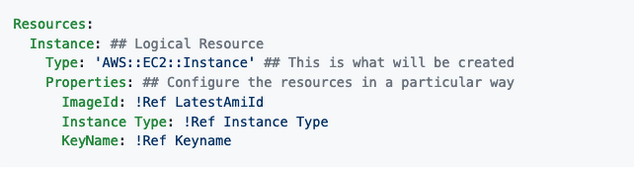
CloudFormation is an automated tool for provisioning entire cloud-based environments. It is similar to Terraform where you codify the instructions for what you want to have inside your application setup (X many web servers of Y type with a Z type DB on the backend, etc). It makes it a lot easier to just describe what you want in markup and have AWS do the actual provisioning work involved. Service for deploying and managing aws resources in a consistent and reproductible manner

Steps:

1. Create Cloud Formation using templates
2. Use a sample template – Pulling a s3 URL
3. Specify details including DBName, DBPass, DBRootpass, instancetype, SSHLocation
4. Click create stack
   * It is a way of completely scripting your cloud environment
   * Quick Start is a bunch of cloud formation templates already built to assist you create complex env easily

The main use case for CloudFormation is for advanced setups and production environments as it is complex and has many robust features.

* CloudFormation templates can be used to create, update, and delete infrastructure.
* The templates are written in YAML or JSON
* A full CloudFormation setup is called a stack.
* Once a template is created, AWS will make the corresponding stack. This is the living and active representation of said template. One template can create an infinite number of stacks.
* The Resources field is the only mandatory field when creating a CloudFormation template
* Rollback triggers allow you to monitor the creation of the stack as it's built. If an error occurs, you can trigger a rollback as the name implies.
* AWS Quick Starts is composed of many high-quality CloudFormation stacks designed by AWS engineers.
* Content at edge locations will be removed only after cache expiration timeout. You can use CloudFront for private, public and geo restricted content distribution. CloudFront hides your origin servers and reduces the attack surface. In addition, it is designed to automatically absorb certain infrastructure level Denial of Service attacks and shield your servers
* An example template that would spin up an EC2 instance:



Does AWS CloudFormation support Amazon EC2 tagging? Yes. Amazon EC2 resources that support the tagging feature can also be tagged in an AWS template

## AWS Organizations

AWS Organizations is an account management service that enables you to consolidate multiple AWS accounts into an organization that you create and centrally manage.

It's not possible to merge multiple AWS accounts together. However, you can use AWS Organizations to manage multiple AWS accounts, and then combine the billing in one consolidated bill. You can also transfer some AWS resources between AWS accounts.

AWS Organizations Key Details:

* Best practices are to use the root account to manage billing only with separate accounts used to deploy resources.
* The point of AWS Organizations is to deploy permissions to the separate accounts underneath the root account and have those policies trickle down. AWS Organizations helps you centrally govern your environment as you grow and scale your workloads on AWS.
* You can use organizational units (OUs) to group similar accounts together to administer as a single unit. This greatly simplifies the management of your accounts.
* You can attach a policy-based control to an OU, and all accounts within the OU automatically inherit the policy. So if your company's developers all have their own sandbox AWS account, they can be treated as a single unit and be restricted by the same policies.
* With AWS Organizations, we can enable or disable services using Service Control Policies (SCPs) broadly on organizational units or more specifically on individual accounts
* Use SCPs with AWS Organizations to establish access controls so that all IAM principals (users and roles) adhere to them. With SCPs, you can specify Conditions, Resources, and NotAction to deny access across accounts in your organization or organizational unit. For example, you can use SCPs to restrict access to specific AWS Regions, or prevent deleting common resources, such as an IAM role used for your central administrators

### Service Control Policies (SCP)

* Service control policies (SCPs) are a type of organization policy that you can use to manage permissions in your organization. SCPs offer central control over the maximum available permissions for all accounts in your organization.
* SCPs help you to ensure your accounts stay within your organization’s access control guidelines. SCPs are available only in an organization that has all features enabled. SCPs aren't available if your organization has enabled only the consolidated billing features.
* SCPs alone are not sufficient to granting permissions to the accounts in your organization. No permissions are granted by an SCP. An SCP defines a guardrail, or sets limits, on the actions that the account's administrator can delegate to the IAM users and roles in the affected accounts.
* The administrator must still attach identity-based or resource-based policies to IAM users or roles, or to the resources in your accounts to actually grant permissions. The effective permissions are the logical intersection between what is allowed by the SCP and what is allowed by the IAM and resource-based policies.

Effect of these policies:

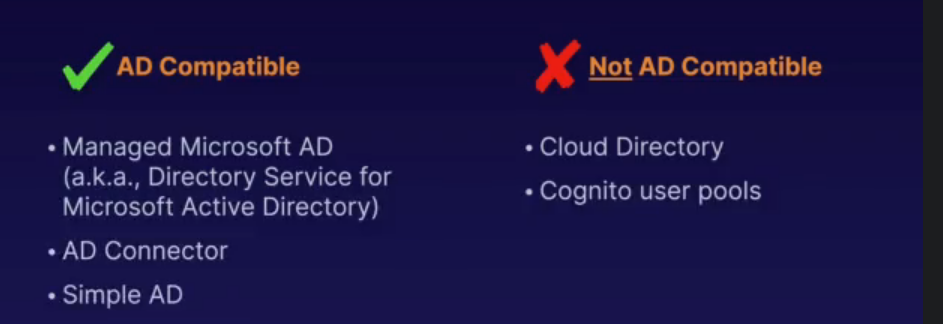
* SCPs affect only IAM users and roles that are managed by accounts that are part of the organization. SCPs don't affect resource-based policies directly. They also don't affect users or roles from accounts outside the organization. For example, consider an Amazon S3 bucket that's owned by account A in an organization. The bucket policy (a resource-based policy) grants access to users from account B outside the organization. Account A has an SCP attached. That SCP doesn't apply to those outside users in account B. The SCP applies only to users that are managed by account A in the organization.
* An SCP restricts permissions for IAM users and roles in member accounts, including the member account's root user. 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 the account administrator attaches the AdministratorAccess IAM policy with \*/\* permissions to the user.
* SCPs affect only member accounts in the organization. They have no effect on users or roles in the management account.
* Users and roles must still be granted permissions with appropriate IAM permission policies. A user without any IAM permission policies has no access, even if the applicable SCPs allow all services and all actions.
* If a user or role has an IAM permission policy that grants access to an action that is also allowed by the applicable SCPs, the user or role can perform that action.
* If a user or role has an IAM permission policy that grants access to an action that is either not allowed or explicitly denied by the applicable SCPs, the user or role can't perform that action.
* SCPs affect all users and roles in attached accounts, including the root user. The only exceptions are those described in Tasks and entities not restricted by SCPs.
* SCPs do not affect any service-linked role. Service-linked roles enable other AWS services to integrate with AWS Organizations and can't be restricted by SCPs.
* When you disable the SCP policy type in a root, all SCPs are automatically detached from all AWS Organizations entities in that root. AWS Organizations entities include organizational units, organizations, and accounts. If you re-enable SCPs in a root, that root reverts to only the default FullAWSAccess policy automatically attached to all entities in the root. Any attachments of SCPs to AWS Organizations entities from before SCPs were disabled are lost and aren't automatically recoverable, although you can manually reattach them.
* If both a permissions boundary (an advanced IAM feature) and an SCP are present, then the boundary, the SCP, and the identity-based policy must all allow the action.

## Directory Services

**AWS Directory Service:**

* Family of AWS MS, allow to connect with 10 on-prem AD, SSO to any joined EC2
* AWS Managed Microsoft AD – AD domain controller running Windows server ( 2 defaults,2 AZ), Extend current AD using AD-tenant.

**AWS V/s Customer MS:**

* This is the fifferent kind of tasks that
* AWS and customer will be responsible to perform in case of any migration or Managed services.
* **AD Connector:** to be used for Directory gateway for on-prem AD, avoid caching info in cloud, allow users to login to AWS using AD, join EC2 instance to AD domain, scaling.
* **Amazon cognito user pools**: Managed user directory for Saas applications, sign-up and sign-off on web or app, works with social media.
* **AD compatible and Not AD Compatible:**

**Amazon Resource Name (ARN):**

* Arn begins with string: arn:partition:service:region:account\_id: | ARN:AWS:S3|EC2|RDS:US-east-1|12fr2
* And ends with: resource , resource\_type: resource, resource\_type/resource,

**IAM policies:**

* JSON doc that defines permissions, identity policy, Resource policy, each action records a API request, effect/action/resource. Deny supersedes allows policies
* 2 types of policies: AWS(denoted by orange icon) or custom made policies

**AWS SSO:**

* Centrally manage AWS accounts and apps, AD and SAML integration (2.0),

# AWS Applications Services

## VPN Cloud Hub

* Single point of contact to connect your VPN infra into.
* Low cost, easy to manage, operates over the public internet, but all traffic will be encrypted

**AWS N/w costs:**

* Use private IP address over public to save costs
* To cut the n/w cost, we need to group all EC2 instance in same AZ, and use Private IP address

## SQS – Simple Query Service

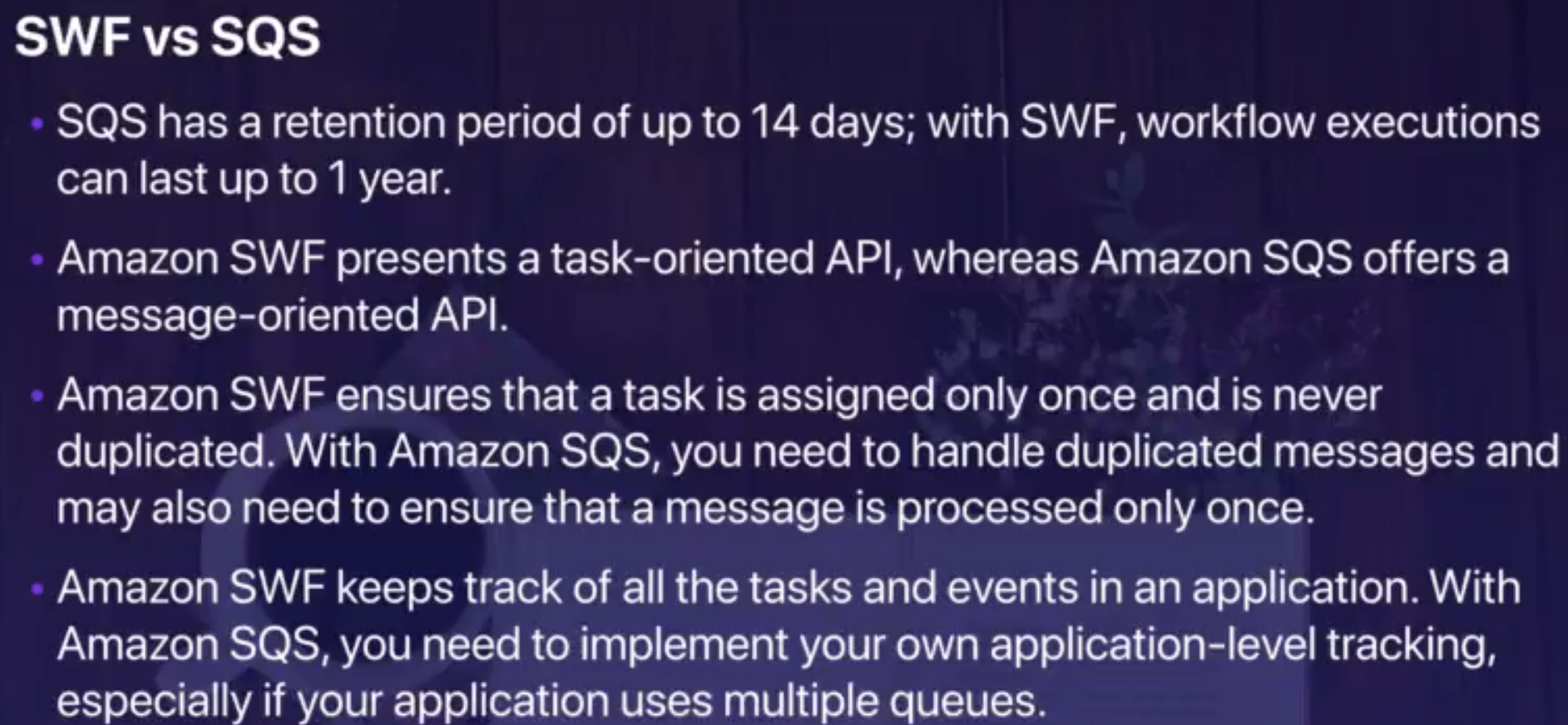
* It is a pull based, ec2 based pull service, msg are in 265 KB, retention period is 4 days
* It is web service that gives access to message queue that can be used to store msgs while waiting for their processing.
* Distributed queue service that enables web apps service to quickly and reliably queue msg that one component generates to be consumed by other
* Two types of SQS:
  + Standard Queue: unlimited number of transactions per sec, msg gets delivered at least once, provide best effort ordering
  + FIFO Queue: Delivery and exactly once processing, same order followed, allow multiple ordered msg groups within single queue, limited to 300 transaction/sec,
* Visibility timed out causes the message to be delivered more than once for a user, visible time out is around 12 hours
* Long polling is a way of retrieving msg from Amazon SQS queues, it does not return a request until messages arrive in msg queue
* Decoupling your infrastructure will be SQS, Need Ec2 to transmit msgs
* Visibility timeout - Other consumers will not receive a message being processed for the configured time period
  + default - 30 seconds, min - 0, max - 12 hours
* DelaySeconds - The time period before a new message is visible on the queue
* Message retention period - The maximum period a message can be on the queue:
  + Default - 4 days, Min - 60 seconds, Max - 14 day
* You can configure an existing SQS queue to trigger an AWS Lambda function when new messages arrive in a queue.
  + Your queue and Lambda function must be in the same AWS Region.
  + FIFO queues also support Lambda function triggers.
  + You can associate only one queue with one or more Lambda functions.
  + You can’t associate an encrypted queue that uses an AWS managed Customer Master Key for SQS with a Lambda function in a different AWS account.
* You can delete all the messages in your queue by purging them.
* **Long polling** helps reduce the cost by eliminating the number of empty responses and false empty responses. While the regular **short polling** returns immediately, even if the message queue being polled is empty, long polling doesn’t return a response until a message arrives in the message queue, or the long poll times out.
  + Short polling occurs when the WaitTimeSeconds parameter of a ReceiveMessage request is set to 0.
* To prevent other consumers from processing a message redundantly, SQS sets a visibility timeout, a period of time SQS prevents other consumers from receiving and processing the message. The default visibility timeout for a message is 30 seconds. The minimum is 0 seconds. The maximum is 12 hours.



## Simple Work Force Service (SWF)

* Web service that makes it easy to coordinate work across distributed application components, enables applications with wide range of use cases to be designed into tasks
* Coordinating the applications
* Carry out the activities

SQS vs SWF



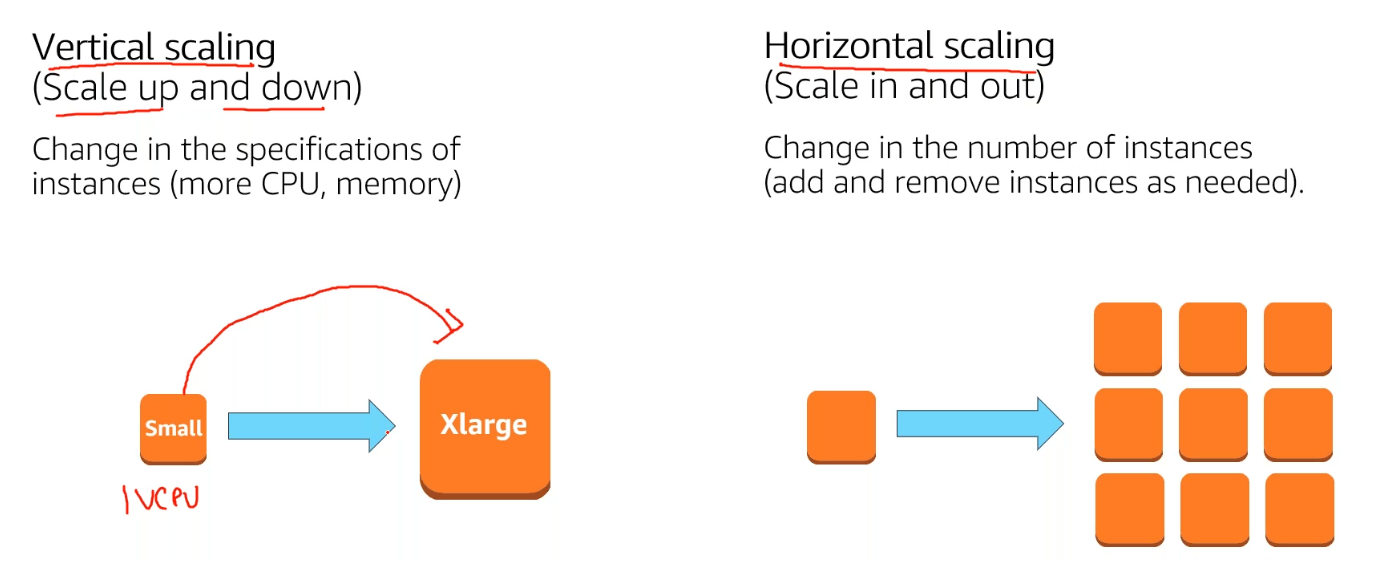
**SNS**

* Web service that makes it easy to setup, operate and send notification from cloud.
* All msgs are published on multi AZ
* pushed service, pay as you go

**Elastic Transcoder**

* Media transcoder in cloud, convert media file into different formats , Pay as per minute,
* Storing: s3 to lambda to elastic transcoder to S3

## Vertical scaling and Horizontal scaling



**API Gateway**

* Front door for applications to access data, business logic or functionality from the back-end services
* Server less technology
* Gateway cache for result caching
* Web sockets capable
* Cache capability to increase performance in secs,
* Expose Https endpoints to define Restful API
* Serverless-ly connect to services like lambda and DynamoDB
* Send each API endpoint to different target
* Scale effortless
* Throttle APU gateway to prevent attacks
* Track and control using API Key
* Connect to cloud watch to log monitoring
* Deploy API
  + Uses API Gateway domain
  + Use custom domain
  + Support AWS Certificate manager
* API Gateway Caching: cache endpoints response, reduce the no. of calls made to endpoint, improve latency of request to API,
* It will record the responses from the endpoint for specific period of time, in sec

## AWS Kinesis

* Platform on AWS to send your streaming data to, it makes easy to load and analyse data, providing ability to build your own custom apps
* Shards is a uniquely identified continuous sequence of data records
  + Each shard can provide support up to 1000 PUT records per sec
  + One shard provides a capacity of 1MB/sec data input
  + No. of shards required specified when stream created
  + Sequence number gets automatically assigned by kinesis when record gets created
* Data record have sequence number, partition key and data blob (~1 MB)
* Stream created using management console
* Development tools - AWS SDKs, Kinesis Client library, Kinesis connector library, kinesis agent
* 3 types:
  1. Kinesis Steams-
     + data storage for 24 hrs to 7 days, data stored on Shards (5 transaction per sec for read;
     + total data read for 2MB per record), which will be connected to EC2 which will be stored on DynamoDB, S3 after analysis
     + Collect and process large stream of data in real time
     + Applications process data sequentially and incrementally on a record to record basis
     + Not available in free tier, billed on shard basis
     + Managed service- Elasticity and durability (3 AZ)
     + Multiple applications can consume data from same stream
  2. Kinesis firehouse- Used to store on redshift, used with lambda
     + Deliver streaming data to destinations (S3, Redshift)
     + No need to write applications or manage resource
     + Data can be transformed using lambda code, which can be buffered
  3. Kinesis analytics – analyse data inside kinesis using standard SQL
     + Supports firehouse, lambda and kinesis data streams
  4. Kinesis Video stream – live stream from devices,
     + Data encryption at rest

**Amazon Kinesis Data Analytics**

Amazon Kinesis Data Analytics provides an efficient way to analyze streaming data, gain actionable insights, and respond to your business and customer needs in real time. It reduces the complexity of building, managing, and integrating streaming applications with other AWS services. Customers can quickly build SQL queries or sophisticated Java applications using built-in templates and operators for common processing functions to organize, transform, aggregate, and analyze data at any scale.

Kinesis Data Analytics takes care of everything required to run real-time applications nearly continuously. It scales automatically to match the volume and throughput of incoming data.

The Kinesis Data Analyticsapplicationis the primary resource in Kinesis Data Analytics that customers use to write the analytic logic to process the streaming data continuously in real time. The application can be written in ANSI SQL with an extension for streaming data or Java language using Apache Flink and the AWS SDK to process incoming data and product output. Kinesis Data Analytics then writes the output to a configured destination.

**How Kinesis Data Analytics works**

* Each Kinesis Data Analytics application consists of three primary components:
* **Input** – Streaming source for the application. In the input configuration, customers must map the input to an in-application data stream so that the data flows from the source into the in-application data stream. Then they can process data from the in-application data stream using their application code.
* **Application code** – Series of Java operators or SQL statements that process data from the in-application data stream and produces output. The application can split the in-application data stream into multiple streams and apply different logic to the separate streams.
* **Output** – Where the application code eventually writes to after completing processing. For example, output could be to a Kinesis Data Firehose that persists the data to Amazon S3 or Amazon Redshift.

## Amazon Cognito:

* Provides web identity federation with signin and sign up access to users, act as identity broker
* Uses Push synchronization to push updates and synchronize user data across multiple devices.
* User pools- user directories used to manage sign-up and sign-in functionality for mobile and web apps, successful sign-in generates a JSON token
* Identity pools- generate temp AWS credentials to access AWS services like S3

# Orchestration and Deployments

## Cost Optimization



AWS Budgets

AWS Cost and Usage Report

AWS Cost Explorer

Saving Plans

AWS Compute optimizer

## EC2 Auto scaling

* EC2 allow to create ec2 instance that can be scaled up and down depending on the condition set
* Enables elasticity by scaling horizontally through adding or terminating ec2 instances
* Enables fault tolerance through health checkup
* Can span across multi AZ
* Scaling policy
  1. Defines how much you want to scale
  2. Uses alarm and policies to determine scaling
* Types of dynamic scaling options
  1. Target tracking
     + scale based upon target value
     + autoscaling creates and manages cloudwatch alarms that trigger autoscaling policies
  2. Step
     + Different scaling adjustment value for different band of conditions
     + Allow small changes in capacity for small changes and large changes
  3. Simple
     + Increase or decrease based upon single scaling demand
     + Cooldown can be adjusted before next scaling event
     + May react too much to small changes

### Parameters

* Health Check Grace Period is the amount of time that Auto Scaling waits before checking the health status of the EC2 instance. EC2 must pass EC2 status checks and ELB health checks (if they are part of ELB) before Auto scaling considers the instance healthy. If EC2 fails the health check, it will be considered unhealthy, and auto-scaling will initiate steps to replace the instance. You can increase this grace period for new instances when an application or server requires a longer time to initialize and come online. The default value is 300 seconds
* Cooldown period (applies only for simple scaling). When a simple scaling policy adds or terminates instances, it waits for the cooldown period before another simple scaling activity is initiated
* The Auto Scaling warmup parameter applies to newly launched instances and ensures instances have sufficient time to initialize before they start reporting CloudWatch data
* ELB Deregistration Delay. When a target is removed from ELB, it goes through a deregistration phase. Deregistration can happen during an Auto Scaling scale-down event. ELB stops sending new requests to targets that are deregistering. However, there could be requests currently in flight on the server. The deregistration delay helps in-flight requests to complete

**ECS auto scaling**

* Use AWS application auto scaling service
* Inc or dec number of tasks based upon the scaling policy: target, step or scheduled

## AWS Application auto-scaling

* Enables auto scaling for ecs scaling, spot fleet, EMR, DynamoDB, sagemaker, aurora replica
* Uses service linked IAM role for permissions that it requires to call other service

**DynamoDB Auto scaling**

* Adjusts provisioned throughput capacity of tables and global secondary indexes
* Reduces throttling of requests

## Auto-Scaling

* 3 components:
  1. **Groups:** Logical component, Webserver group
  2. **Configuration templates:** groups uses a launch template or launch configuration as configuration template for its EC2 Instances
  3. **Scaling Options:** provides several ways to scale your auto scaling groups.
* Type of scaling options
  1. Maintain the current instance level at all time
  2. Scale manually
  3. Scale based on schedule
  4. Scale based on demand
  5. Use Predictive scaling
* When it comes to actually scale your instance groups, the Auto Scaling service is flexible and can be done in various ways:
  + Auto Scaling can scale based on the demand placed on your instances. This option automates the scaling process by specifying certain thresholds that, when reached, will trigger the scaling. This is the most popular implementation of Auto Scaling.
  + Auto Scaling can ensure the current number of instances at all times. This option will always maintain the number of servers you want running even when they fail.
  + Auto Scaling can scale only with manual intervention. If you want to control all of the scaling yourself, this option makes sense.
  + Auto Scaling can scale based on a schedule. If you can reliably predict spikes in traffic, this option makes sense.
  + Auto Scaling based off of predictive scaling. This option lets AWS AI/ML learn more about your environment in order to predict the best time to scale for both performance improvements and cost-savings.
* In maintaining the current running instance, Auto Scaling will perform occasional health checks on the running instances to ensure that they are all healthy. When the service detects that an instance is unhealthy, it will terminate that instance and then bring up a new one online.
* When designing HA for your Auto Scaling, use multiple AZs and multiple regions wherever you can.
* Auto Scaling allows you to suspend and then resume one or more of the Auto Scaling processes in your Auto Scaling Group. This can be very useful when you want to investigate a problem in your application without triggering the Auto Scaling process when making changes.
* You can specify your launch configuration with multiple Auto Scaling groups. However, you can only specify one launch configuration for an Auto Scaling group at a time.
* You cannot modify a launch configuration after you've created it. If you want to change the launch configuration for an Auto Scaling group, you must create a new launch configuration and update your Auto Scaling group to inherit this new launch configuration.

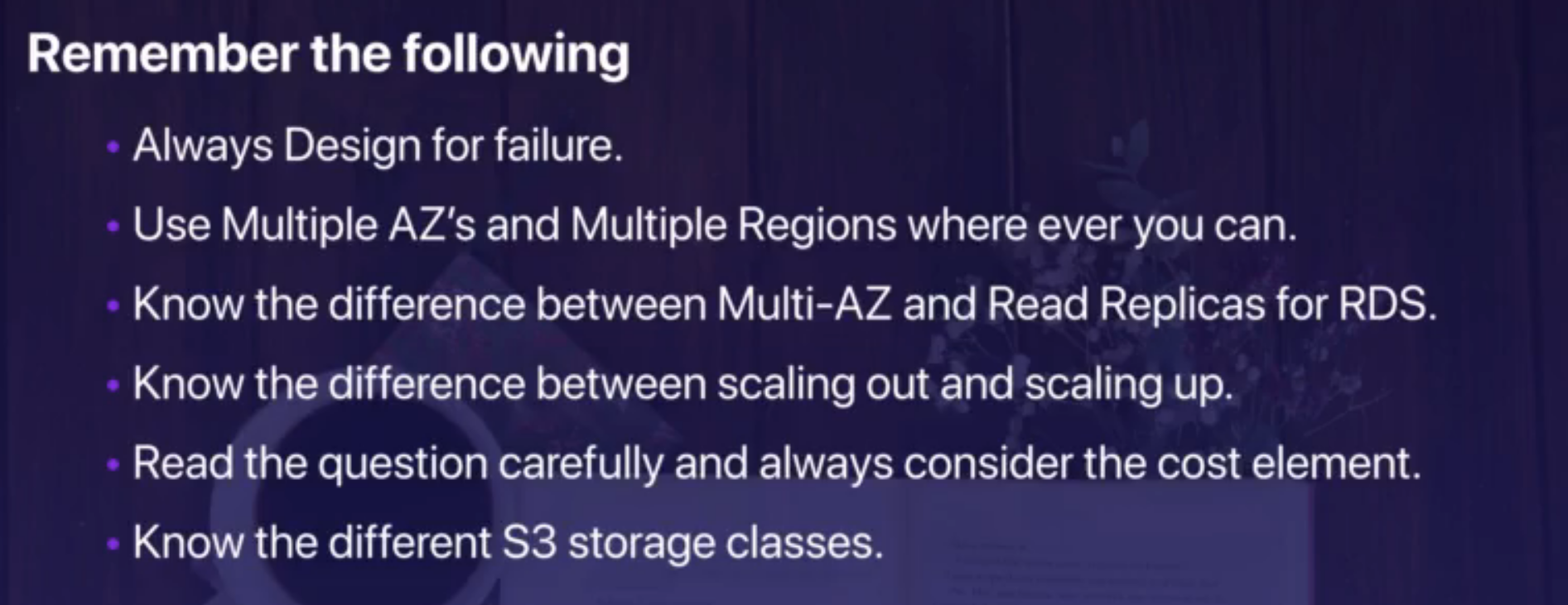
### Auto Scaling Default Termination Policy:

* The default termination policy for an Auto Scaling Group is to automatically terminate a stopped instance, so unless you've configured it to do otherwise, stopping an instance will result in termination regardless if you wanted that to happen or not. A new instance will be spun up in its place.
* The default termination policy will spare instances that you tell it in case some servers are running critical systems or applications. These critical servers are protected from "scale in", which is just the deletion process of instances deemed superfluous to requirements.
* The default termination policy is designed to help ensure that your network architecture spans Availability Zones evenly. With the default termination policy, the behavior of the Auto Scaling group is as follows:
  + If there are instances in multiple Availability Zones, it will terminate an instance from the Availability Zone with the most instances. If there is more than one Availability Zone with the same max number of instances, it will choose the Availability Zone where instances use the oldest launch configuration.
  + It will then determine which unprotected instances in the selected Availability Zone use the oldest launch configuration. If there is one such instance, it will terminate it.
  + If there are multiple instances to terminate, it will determine which unprotected instances are closest to the next billing hour. (This helps you maximize the use of your EC2 instances and manage your Amazon EC2 usage costs.) If there are some instances that match this criteria, they will be terminated.

### Auto Scaling Cooldown Period:

* The cooldown period is a configurable setting for your Auto Scaling Group that helps to ensure that it doesn't launch or terminate additional instances before the previous scaling activity takes effect.
* After the Auto Scaling Group scales using a policy, it waits for the cooldown period to complete before resuming further scaling activities if needed.
* The default waiting period is 300 seconds, but this can be modified.

**HA Architecture**



# Backup and Recovery

RPO – Recovery point Objective

* Max age of files that can be restored from Backup
* High backup frequency means low RPO

RTO – recovery time Objective

* Maximum length of time a system can be down
* Glaciers can take hours to take to restore backup, s3 could be quicker

Full vs incremental backup

* Full backup – captures entire system, requires large disk space, Fast RTO, poor RPO
* Incremental Backup- Captures only changes made since last backup, less disk space, slower RTOO, Fast RPO

Amazon Machine Images (AMI)

* Provides backup of system Images
* Registered and saved to S3
* Can copy to different region, Sell on AWS Marketplace to different users when not required
* Contains: EBS snapshots, or instance store root vol templates, launch permissions

Elastic Block Storage (EBS)

* Incremental snapshots saved to s3, Console or CLI ec2 create snapshot command
* Restore to new EBS vol then unmount existing EBS vol and mount new
* Can be copied to different regions

Amazon Elastic File system (EFS)

* Node JS Application on lambda
* Deployed on cloud formation
* Incremental backups
* Customer defined schedule

RDS Backup and recovery

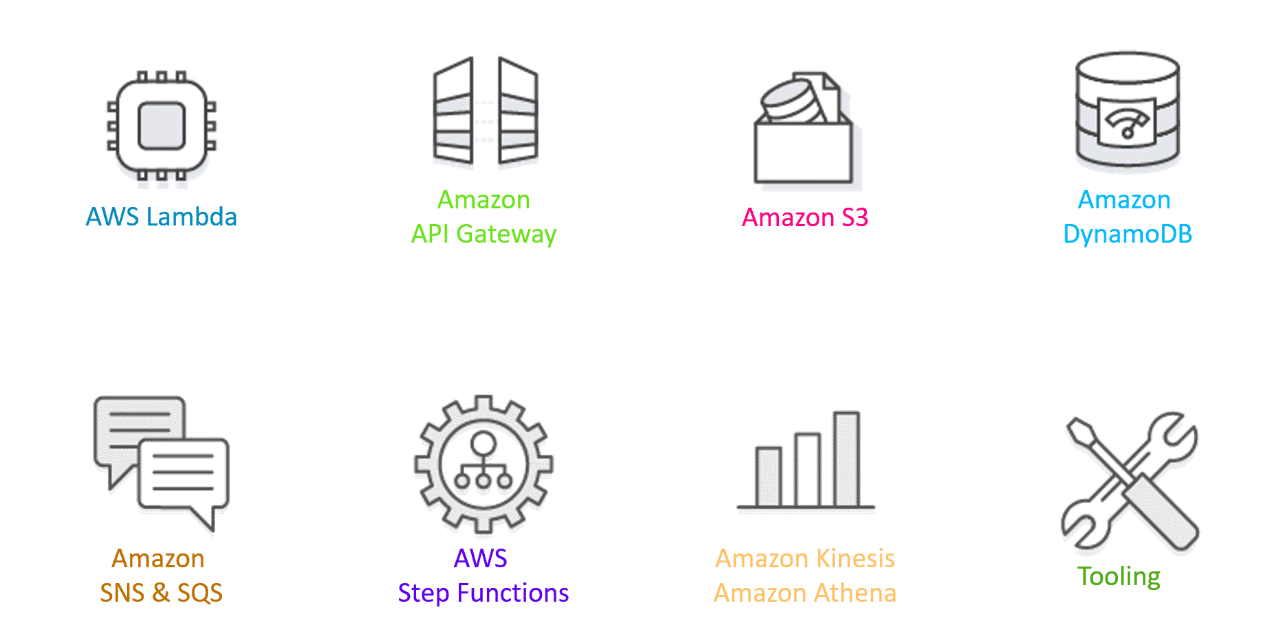
* **Automated Backups:**
  + Daily snapshot of data and transactions logs
  + Retention period of 1 day
  + Automated backup are deleted when DB instance are deleted, Backup window are period incremental backups
  + Periodic backup combined with transaction log provides recovery to last restored time
* **RDS DB Snapshot:**
  + User initiated- not deleted when DB instance is delete, need to specifically delete these
* **DynamoDB Table Import/export:**
  + Created through console or CLI SDK, export table and log files to S3
  + Can be setup as daily export
  + Simple import back to table with console or CLI SDK

**Stored Volume Vs Cached Volume:**

Volume Gateway's Stored Volumes let you store data locally on-prem and backs the data up to AWS as a secondary data source. Stored Volumes allow low-latency access to entire datasets, while providing high availability over a hybrid cloud solution. Further, you can mount Stored Volumes on application infrastructure as iSCSI drives so when data is written to these volumes, the data is both written onto the on-prem hardware and asynchronously backed up as snapshots in AWS EBS or S3.

Volume Gateway's Cached Volumes differ as they do not store the entire dataset locally like Stored Volumes. Instead, AWS is used as the primary data source and the local hardware is used as a caching layer. Only the most frequently used components are retained onto the on-prem infrastructure while the remaining data is served from AWS. This minimizes the need to scale on-prem infrastructure while still maintaining low-latency access to the most referenced data

# Serverless Platform



# AWS Machine Learning Components

* Rekognition: face detection, labeling, celebrity recognition
* Transcribe: audio to text (ex: subtitles)
* Polly: text to audio
* Translate: translations
* Lex: build conversational bots – chatbots
* Connect: cloud contact center
* Comprehend: natural language processing
* SageMaker: machine learning for every developer and data scientist
* Forecast: build highly accurate forecasts
* Kendra: ML-powered search engine
* Personalize: real-time personalized recommendations
* Textract: detect text and data in documents

# Service Catalogue

Why is my product not showing up in AWS Service Catalog?

To see the product(Not under Administration) you have to go to your portfolio and assign Group,Role or User to portfolio. Once you assign, the product under portfolio will be visible

## Service Catalogue Templates

Service Catalog Template Constraints

To limit the options that are available to end users when they launch a product, you apply template constraints. Apply template constraints to ensure that the end users can use products without breaching the compliance requirements of your organization. You apply template constraints to a product in a Service Catalog portfolio. A portfolio must contain one or more products before you can define template constraints.

A template constraint consists of one or more rules that narrow the allowable values for parameters that are defined in the product's underlying AWS CloudFormation template. The parameters in an AWS CloudFormation template define the set of values that users can specify when creating a stack. For example, a parameter might define the various instance types that users can choose from when launching a stack that includes EC2 instances.