# AWS (Amazon Web Series)

# IAM (Identity and Access Management)

* IAM provides access to accounts services where we can manage User, Roles, Groups & Policy
* It applies globally to all AWS regions.

**Users**: We create users and assign necessary permissions to them in the form of policies.

**Groups:** We can create groups for ex. Dev, QA etc. and attach policies at the group level.

**Policy:** A policy is a set of permission attached to user, group, or roles

**{**

**"Version": "2012-10-17",**

**"Statement": [**

**{**

**"Effect": "Allow",**

**"Action": \*,**

**"Resource": "\*"**

**}**

### Policy types:

1. **Identity Based Policy:** Applicable on users, groups of users, and roles.

* **AWS Managed Policy**: Policies which are managed by AWS.
* **Custom Managed Policy**: Policies which are created by user as per requirement.
* **Inline Policy**: Policies that you add directly to a single user, group, or role. Inline policies maintain a strict one-to-one relationship between a policy and an identity. They are deleted when you delete the identity.

1. **Resource Based Policy:** To add specific users to access specific resources, also to enable cross-account access
2. **Session based Policy:** Create a temporary session for a role.

Resource Based Policy is the priority out of three policies

More than one policy can be attached to a user or a group at the same time.

Policies can’t be attached directly to resources like EC2 instance, S3 bucket etc.,

**Basic Policy structure:**

**Sid**: It is a unique identifier for each statement within an AWS policy document.

**Effect**: Can restrict the permissions by allow or deny

**Principal:** who is assuming the policy

**Resource:** Here we mention the resources on which the policy is created

**Action**: Here we can mention what are the actions to be performed on the resources ex: read, write.

Ex:

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "FirstStatement",

"Effect": "Allow",

"Action": ["iam:ChangePassword"],

"Resource": "\*"

},

Q: Can we attach multiple policies to user group or a role.

Yes, we attach, I think we can attach 20 policies to a role.

### Roles

A role is a set of permissions that grant access to actions and resources in AWS.

* Roles comes between services, like ec2 wants to access S3 or non-AWS user (hybrid account) should access AWS Resources.
* Policies can’t be attached to AWS resources hence roles come into picture.
* EC2 can be attached one role at a time.
* Can we assign multiple roles to a EC2 instance?

No, we can’t. we can assign only single role to EC2 instance.

### Assume Role:

Returns a set of temporary security credentials that you can use to access AWS resources that you might not normally have access to. These temporary credentials consist of an access key ID, a secret access key, and a security token.

Example:

1. create a IAM user

2. Add him ec2 full access policy

3. Try to list the bucket (aws s3api list-buckets) - you can't list the bucket because the user is not having the permission to list bucket (access denied)

4. Grant the user to assume a role

Create a role

Attach a policy s3 full access policy

get inside the role ---- trust Relationship change it to user instead of ec2

AWS: "ARN OF IAM USER"

5. aws sts assume-role --role-arn <enter the role arn> --role-session-name s3-access-example --duration-seconds 3600

6. copy the accesskey secretkey and seeison token

7 set AWS\_ACCESS\_KEY\_ID=<enter the copied access key>

set AWS\_SECRET\_ACCESS\_KEY=<enter the copied session key>

set AWS\_SESSION\_TOKEN=<sessiontoken>

8. aws s3api list-buckets

9. Remove the env variable

set AWS\_ACCESS\_KEY\_ID=

set AWS\_SECRET\_ACCESS\_KEY=

set AWS\_SESSION\_TOKEN=

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

# CloudWatch

Amazon Cloud Watch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real time. It can automatically set alarms, track, and collect metrics, and respond to changes in your AWS resources.

Text, timeline

Description automatically generated with medium confidence

**Metrics:**

Metrics are data about the performance of your systems.

Basic monitoring: which polls for every 5 minutes

Detailed monitoring: which polls for every 1 minute.

**Alarm:**

CloudWatch Alarms feature allows you to watch CloudWatch metrics and to receive notifications when the metrics fall outside of the levels (high or low thresholds) that you configure.

Ex:

If CPU utilization goes beyond the static threshold alarm goes to alarm state

**Three states in CW Alarm:** - Alarm state, Insufficient data, OK state

**Logs:**

CloudWatch Logs enables you to centralize the logs from all your systems, applications, and AWS.

# Cloud Trial

It is mainly used to track user and API usage activity.

Any action taken by users, roles and AWS services are recorded to cloud trial.

**Two types of events**

Management Events: - Management operations performed on AWS ex: - EC2 Instance Launch, S3 Object Creation, IAM Role Creation

Data Events: - Collect the log of resource operations performed on or within a resource. Also collects Creation, modification, or deletion of data stored in AWS services, such as Amazon S3 or Amazon DynamoDB

Insights Events: - Identifies the unusual activity, error or user behaviour in AWS account.

Cloud Trials files are encrypted using S3 server-side encryption & placed into S3 bucket.

Even if resources are deleted by use of cloud trail can be tracked.

CloudTrail logs are stored in an S3 bucket and can be used for security analysis, compliance auditing, and troubleshooting.

## Difference between Cloud watch and Cloud Trial

**CloudWatch** is primarily focused on monitoring the performance of AWS resources and applications, while **CloudTrail** is focused on providing a history of activity in your AWS account for security, compliance, and auditing purposes.

# Simple Notification Service

Amazon Simple Notification Service is a notification service provided as part of Amazon Web Service.

It provides a low-cost infrastructure for the mass delivery of messages, predominantly to mobile users.

**Topic:**

Diagram

Description automatically generated Diagram

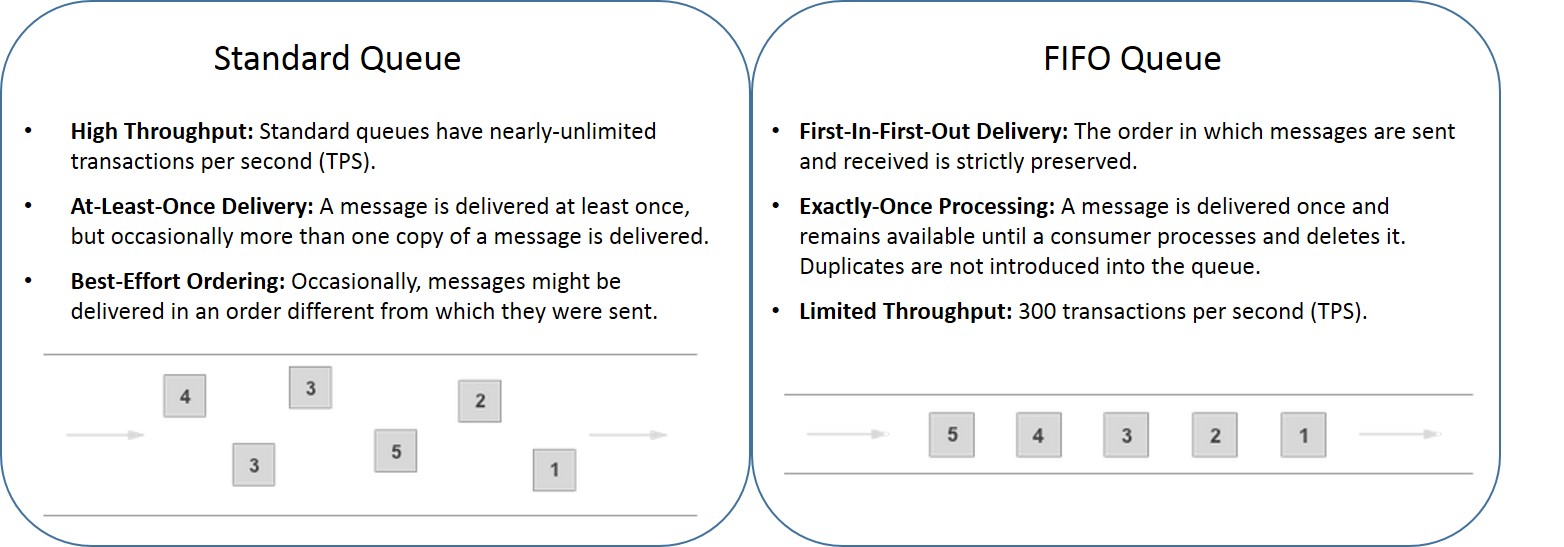
Description automatically generated

## SNS Use case

There was a requirement in my current organization where I need to send notification for critical server. So we set up cloud watch alarm for critical server so if any action state change occurs with that instance there was a notification pinging or SMS will be recived. SNS uses a push mechanism to immediately deliver messages to subscribed endpoints

# Simple Queue Service (SQS)

It is hosted queue for storing messages as they travel between application or microservices.



## SQS Use Case

We had a microservice application where there were continuously generating data and this needs to be consumed by multiple other microservice applications, so I created a queue (SQS) & helped developers to integrating this. SNS uses a push mechanism to immediately deliver messages to subscribed endpoints

# Virtual Private Cloud

Amazon Virtual Private Cloud (Amazon VPC) lets you provision a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define like EC2 instance Databases.

### CIDR:(Classes Inter domain routing)

Classless Inter-Domain Routing is a method for allocating [IP addresses](https://en.wikipedia.org/wiki/IP_address) and for [IP routing](https://en.wikipedia.org/wiki/IP_routing).

Ex: The IPv4 block 192.168.100.0/22 represents the 1024 IPv4 addresses from 192.168.100.0 to 192.168.103.255.

I.e., 2^ (32-22) = 2^10 = 1024 IPv4 addresses.

### SUBNET

* A **subnet** is a range of IP addresses in your VPC. You can launch AWS resources into a specified subnet.
* Public Subnet: -A public subnet is a subnet that's associated with a route table that has a route to an internet gateway
* Private Subnet: -The subnet does not have a direct route to an internet gateway. Resources in a private subnet require a [NAT device](https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat.html) to access the public internet.

### ROUTING TABLE

* A route table is used to control the flow of network traffic between subnets and other network resources

### INTERNET GATEWAY

* Internet gateway allows communication between instances in a VPC and the internet.

### NACL

1. A network access control list (ACL) is an optional layer of security for your VPC that acts as a firewall for controlling traffic in and out of one or more subnets. (Firewall at subnet level)
2. Inbound means – incoming (Ingress)
3. Outbound means – outgoing (egress)
4. Table

   Description automatically generatedAlways explicit deny take precedence over allow

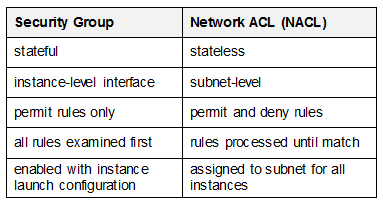
### Security Group:

A **security group** acts as a virtual firewall for your instance to control inbound and outbound traffic.

Graphical user interface, application

Description automatically generated

### Difference between NACL and Security Group



## Stateful set and Stateless set

A stateful security group rule tracks the state of a network connection and allows return traffic automatically. This means that if an outbound connection is initiated by an instance in the security group, the security group automatically allows the return traffic. The rule will also allow new incoming traffic for an established connection. Stateful rules are typically used for protocols that require a connection to be established, such as TCP.

A stateless security group rule does not track the state of a network connection and treats each incoming and outgoing packet independently. In other words, if an outbound connection is initiated by an instance in the security group, the security group will not automatically allow the return traffic. A separate rule is required to allow the return traffic. Stateless rules are typically used for protocols that do not require a connection to be established, such as UDP.

### Nat gateway

A NAT gateway is a Network Address Translation (NAT) service. You can use a NAT gateway so that instances in a private subnet can connect to services outside your VPC, but external services cannot initiate a connection with those instances.

### Bastion host

A **bastion host** is a **server** whose purpose is to provide access to a private network from an external network

### Difference between NACL and Security Group

## VPC CREATION

* Create VPC, allocate CIDR range while creating VPC.
* Create subnets within the allocated CIDR range.
* Create Routing Table within VPC
* Create Internet Gateway
* It we attach Routing table to Internet Gateway and Subnet then it is known as public subnet.
* Subnet which is not attached to Internet gateway is known as private subnet.
* We have NACL and Security Groups which is used to manage the traffic in VPC.
* NACL is firewall at subnet level.
* Security group is firewall at instance level.
* **Some IP addresses are reserved they are**
* 10.180.0.0 Network address
* 10.180.0.1 VPC Router
* 10.180.0.2 DNS server (**DNS**. (Domain Name System) The Internet's system for converting alphabetic names into numeric IP addresses)
* 10.180.0.3 Future use
* 10.180.0.255 N/W Broadcast address
* VPC spans multiple Availability zones.

## VPC Quota or VPC limitations

* 5 VPC per region
* 5 IGW per region
* Subnet per VPC 200
* IPv4 CIDR blocks per VPC 4
* Elastic IP addresses per region 5
* Internet gateways per region 5
* NAT gateways per Availability Zone 5
* Network ACLs per VPC 200
* Rules per network ACL 200

# VPC types

\*Amazon VPC with a single public subnet only.

\*Amazon VPC with public and private subnets.

\*Amazon VPC with public and private subnets and AWS Site-to-Site VPN access.

\*Amazon VPC with a private subnet only and AWS Site-to-Site VPN access.

## VPC Peering:

* A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IPv4 addresses or IPv6 addresses.
* Instances in either VPC can communicate with each other as if they are within the same network.
* You can create a VPC peering connection between your own VPCs, or with a VPC in another AWS account.
* The VPCs can be in different regions (also known as an inter-region VPC peering connection).

**Conditions:**

* CIDR block shouldn’t overlap
* Transitive peering relationships are not supported. i.e here VPC B cannot connect with VPC C.
* If the VPCs are in different regions, inter-region data transfer costs apply.
* You cannot have more than one VPC peering connection between the same two VPCs at the same time.

# VPC ENDPOINTS:

VPC Endpoint allows you to connect the VPC to your AWS services without the help of an Internet Gateway, NAT device, VPN or an AWS Direct Connect connection.

**Types of vpc endpoint**

\***Gateway endpoint**: A [gateway endpoint](https://docs.aws.amazon.com/vpc/latest/userguide/vpce-gateway.html) is a target for a route in a route table to connect VPC resources to S3 or DynamoDB. Traffic is then routed from instances in a subnet to one of these two services.

\***Interface endpoint**: An [interface endpoint](https://docs.aws.amazon.com/vpc/latest/userguide/vpce-interface.html) is an elastic network interface that allows a private IP address in a subnet to connect VPC resources to a number of AWS services, such as CloudFormation, Elastic Load Balancers (ELBs), SNS.

## Elastic IP Addresses:

An Elastic IP address is a static IP address that can be assigned to an EC2 instance or another resource in a VPC. It can be remapped to another instance or resource, without disrupting the underlying network or service.

# VPN:

Is mainly used to establish a secure and private tunnel from you network or device to aws network.

* Aws site-to-site vpn: enables you to securely connect your on-premises network to your vpc.
* AWS client vpn: enables you to securely connect users to AWS or on premises network.

**Do we have another way we can connect to the resources in a private subnet?**

* We can setup a vpn server in the public subnet and configure it to connect to resources residing the private subnet.

## Difference b/w NAT gateway and NAT instance

Graphical user interface, text

Description automatically generated with medium confidence

# Transit gateway:

A transit gateway is a network transit hub that you can use to interconnect our virtual private clouds (VPCs) and on-premises networks.

**Difference between transit gateway and vpc peering**

\*Vpc peering can only be used for two vpc, but transit gateway can we used to connect multiple vpc.

\*We cannot attach VPN and aws direct gateway to vpc peering, but we can attach to transit gateway.

# Elastic Cloud Compute (EC2)

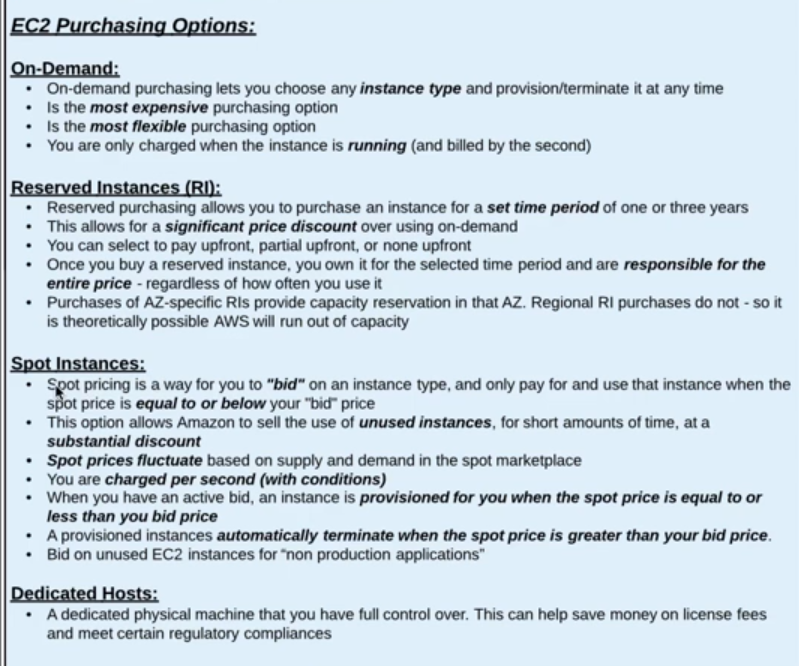
An **EC2 instance** is a virtual server in Amazon's Elastic Compute Cloud (**EC2**)

## EC2 instance types:

**Text

Description automatically generated**

### Purchasing options:

****

## EBS (Elastic block storage)

Provides block level storage volumes for use with EC2 instances

**Text

Description automatically generated**

### EBS Types:

**Text

Description automatically generated**

## EFS:

Amazon Elastic file systemis a regional service storing data within and across multiple Availability Zones (AZs) for high availability and durability

### Difference between EBS v/s EFS v/s S3

Table

Description automatically generated

* EBS: - When you need a high-performance storage service for a single instance
* EFS may be used whenever you need a shared file storage option for multiple EC2 instances with automatic, high-performance scaling.
* S3 is good at storing long-term data due to its archiving system. Things like reports and records, which may go unused for years, can be stored on S3 at a lower cost than the other two storage services

### Snapshot EBS

* You can back up the data on your Amazon EBS volumes to Amazon S3 by taking point-in-time snapshots. Snapshots are incremental backups

### Snapshots are stored in S3

* Launch two ec2 instance in different az’s (instance1 & instance2)
* Create EBS volume and attach it to instance1
* The volumes are attached to instance1 you can verify it by logging into instance1 and executing “lsblk” command, but it’s not mounted you can verify it through by running command “df -TH”
* Mount the volume to instance1
* Format the disk with ext4: “mkfs -t ext4 /dev/xvdf”
* Create a directory in root: 1. “cd /” 2. “mkdir /mnt/mydisk”
* Mount the disk: “mount /dev/xvdf /mnt/mydisk”
* you can verify that disk is mounted by running “df -TH” command.
* Create some files
* Take a snapshot
* Unmount the disk
* umount /mnt/mydisk
* Detach the volume from ec2 instance.
* delete the volume
* Create a new volume from snapshot
* Attach the volume to newly created instance2.
* Mount the volume to instance2
* Create a directory in root: 1. cd / 2. mkdir /mnt/mydisk
* mount /dev/xvdf /mnt/mydisk

### Data life cycle Manager:

You can use Amazon Data Lifecycle Manager to automate the creation, retention, and deletion of snapshots taken to back up your Amazon EBS volumes

## Amazon machine image (AMI):

* An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance.
* You can launch multiple instances from a single AMI when you need multiple instances with the same configuration.

### Difference between Snapshot and AMI

An EBS snapshot is a backup of a single EBS volume. The EBS snapshot contains all the data stored on the EBS volume at the time the EBS snapshot was created.

An AMI image is a backup of an entire EC2 instance. Associated with an AMI image is EBS snapshots. Those EBS snapshots are the backups of the individual EBS volumes attached to the EC2 instance at the time the AMI image was created

## Elastic load Balancer (ELB):

Elastic Load Balancing automatically distributes your incoming traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in one or more Availability Zones. It monitors the health of its registered targets, and routes traffic only to the healthy targets.

## Types of Load balancer:

**Application load balancer:**

* Used mainly for web application running http and https protocols. Operates at request level.
* The Application Load Balancer operates at the request level only. If you're dealing with HTTP requests, which you are for your web application, It also supports advanced features like host and path-based routing
* We can create target groups in order to route to traffic to the respective paths

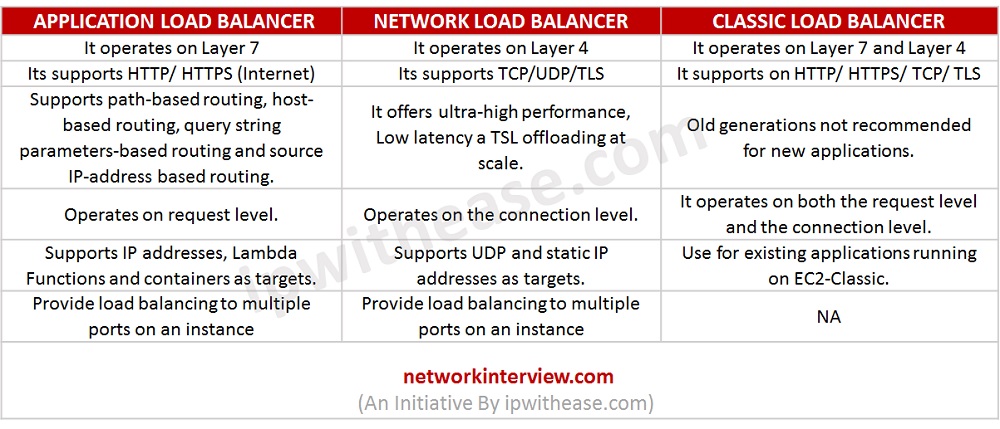
**Network Load balancer:**

* Ultra-high Performance at very low latency. Can handle millions of requests per second.
* Operates at connection level, routing traffic to targets with in VPC.

**Classic load Balancer:**

* Used for applications that were built in existing EC2 classic env.
* Operates both at connection & request level.

## Difference b/w ALB vs NLB vs CLB



### Listeners: -

A listener is a process that checks for connection requests using the protocol and port that you configured.

### Target Group: -

Target group tells a load balancer where to direct traffic to: EC2 instances, fixed IP addresses; or AWS Lambda functions and perform health check on the targets.

## Create Load Balancer

[Step 1: Select a load balancer type](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-getting-started.html#select-load-balancer-type)

[Step 2: Define your load balancer](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-getting-started.html#define-load-balancer)

[Step 3: Assign security groups to your load balancer in a VPC](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-getting-started.html#select-vpc-security-group)

[Step 4: Configure health checks for your EC2 instances](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-getting-started.html#configure-health-check)

[Step 5: Register EC2 instances with your load balancer](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-getting-started.html#register-ec2instances)

[Step 6: Tag your load balancer](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-getting-started.html#create-tags)

[Step 7: Create and verify your load balancer](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-getting-started.html#create-load-balancer)

## ELASTIC LOAD BALANCING HEALTH CHECKS

1. HealthCheckProtocol

2. HealthCheckPort

3. HealthCheckPath

4. HealthCheckTimeoutSeconds

5. HealthCheckIntervalSeconds

6. HealthyThresholdCount

7. UnhealthyThresholdCount

# Autoscaling

AWS Auto Scaling monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost. Amazon EC2 Auto Scaling helps ensure that your application always has the right amount of capacity to handle the current traffic demand.

## Types

### 1.Horizontal auto scaling

Horizontal autoscaling involves adding more instances to the application tier as the load increases. It helps to distribute the load across multiple instances and ensures that the application can handle more traffic.

### 2.Vertical auto scaling

Vertical autoscaling involves increasing the resources of a single instance in response to increased demand. This can be achieved by changing the instance type or by adding more CPU or memory to the instance. AWS provides options to scale up or scale down an instance, depending on the requirements

## Auto scaling setup

* Create an auto scaling and switch to launch configuration
* Create a launch configuration in that select the AMI and instance type, volumes and configure the security groups
* Come back to auto scaling and select the created launch configuration
* Select vpc and subnets and select the load balancer type
* Enter min and max scaling and select the scaling policies
* Then click on create auto scaling.

## Scaling Policies

1. **Target tracking scaling policies**: We make use of this policy to auto scale if the CPU percent is around 50% and it a used for simple applications.
2. **Step scaling and simple scaling**: It is based out with cloud watch alarms and it will trigger the alarm if CPU utilization is more than 75% and will scale up and if we have set 25% than it will automatically scale down.
3. **Schedule Action**: In this case you can scale up the instance based on the period of time

**Launch configuration**: A launch configuration is an instance configuration template that an Auto Scaling group uses to launch EC2 instances.

**Launch template:** It captures all the parameters required to launch an ec2 instance (AMI, type, key pair, security group, etc)

# Route53:

Amazon Route 53 is a highly available and scalable Domain Name Server (DNS) web service, where we can point IP address to domain name or point host name to another host name.

**Importance**: -

* If a website needs a name, Route 53 registers the name for the website (domain name)
* Route 53 helps to connect the browser with the website or web app when the user enter domain name
* Route 53 checks health of resources by sending automated requests over the internet to a resource

**Diagram

Description automatically generated**

### A Record:

Maps IP address to domain name ex: 10.180.0.0 to myapp.mydomain.com

### CNAME Record:

Maps hostname to another host name: us-east.2.elb.amazonaws.com to myapp.mydomain.com

### Alias Record:

Points a host name to AWS Resource ex: myapp.mydomain.com to us-east.2.elb.amazonaws.com

### Latency Routing Policy:

Use when you have resources in multiple AWS Regions, and we want to route traffic to the region that provides the best latency.

If a website has to be installed or hosted across multiple AWS regions, then Latency routing policy is used.

### Weighted Routing Policy:

Use to route traffic to multiple resources in proportions that you specify.

Weighted routing routes multiple resources with a single domain name and control the traffic is routed to each resource. It is useful load balancing and testing new versions of software

**Chart

Description automatically generated with medium confidence**

# What happens when you type a URL in the browser and press enter?

* The browser checks the cache for a DNS record to find the corresponding IP address of google.com
* If the requested URL is not in cache, DNS server initiates a DNS query to find IP address of server that hosts google.com
* The browser initiates a TCP connection with the server
* The browser sends an HTTP request to the webserver
* The server handles the request and sends back a response
* The server sends out an HTTP response: - response contains the web page that is requested as status code, cookies.
* The browser displays the HTML content

**https://medium.com/@maneesha.wijesinghe1/what-happens-when-you-type-an-url-in-the-browser-and-press-enter-bb0aa2449c1a**

# Simple storage service (S3)

Amazon S3 has a simple web services interface that you can use to store and retrieve any amount of data, at any time, from anywhere on the web.

**Single operation upload:**

* It’s a traditional upload where you will upload the object in one part
* A single operation upload can upload the file up to 5GB in size.

**Upload object in parts:**

* Using multipart upload, you can upload the large objects up to 5TB.
* You can use multipart upload for the objects from 5MB to 5TB in size.

**Rules for bucket naming:**

* Bucket names must be between 3 and 63 characters long.
* Bucket names can consist only of lowercase letters, numbers, dots . and hyphens -.
* Bucket names must begin and end with a letter or number
* Bucket names must not be formatted as an IP address (for example, 192.168.5.4).
* Bucket names can't begin with xn-- (for buckets created after February 2020).

**Limitation of S3 bucket:**

* Only 100 buckets can be created per account.
* Can hold unlimited objects

## S3 Storage classes:

* **Standard:**
* Designed for general- and all-purpose storage.
* Default storage option

This tier should be used for objects where durability and availability are important. Some good uses cases include:

* Public websites, including static content. This is great to use as a backend for CloudFront distributions.
* Log storage
* Configuration files
* **Reduced Redundancy storage**
* Designed for non-critical objects
* Less expensive than standard
* **Infrequent access**
* Designed for less frequently accessed objects.
* This class of storage is great for data that needs to be stored reliably, be able to be retrieved quickly
* **Glacier**
* Designed for long term archival storage
* May take several hours to retrieve the objects from this storage
* Cheapest s3 storage class

### S3 Life cycle policy:

An object lifecycle policy is a set of rules that automate the migration of the object storage class to different storage class

By default, lifecycle policies are disabled for a bucket

### S3 Encryption:

**Two ways of protecting information with S3**

1. **Server side/At rest:**

Request Amazon S3 to encrypt your object before saving it on disks in its data centers and then decrypt it when you download the objects.

Graphical user interface, text, application

Description automatically generated

1. **In-transit/Client-side encryption:**

Encrypt data client-side and upload the encrypted data to Amazon S3. In this case, you manage the encryption process, the encryption keys, and related tools.

**Diagram

Description automatically generated with medium confidence**

# How S3 Versioning works

Mainly used to keep the multiple Version of a object in a single bucket, and enable you to restore objects when it is accidentally deleted.

•for eg:- if you delete an object, amazon S3 inserts an delete mark which becomes as current object version and we can recover the previous version of object.

By default, when we create a S3 bucket the bucket is un versioned we need to enable it

when you enabled versioning, Amazon S3 which automatically generate the unique ID for the object is being stored.

for eg. there is a bucket you if you add an object (xyz.jpg) for the 1st time the unique ID will be created on the 2nd time. also, if you add the same object (xyz.jpeg) it will create a new unique id will be generated

If Versioning is not enabled the object will be get replaced

# Lambda

AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume.

 Just upload your code and Lambda takes care of everything required to run and scale your code with high availability

* Manage your virtual functions not really caring about the servers
* Run on demand
* Scaling is automated
* Manual repetitive work

**AWS Lambda Languages:**

NodeJS, Python, Python3, Gr00vy, java, csharp, Scala and GO

**AWS Lambda Integration**

Kinesis, API Gateway, DynamoDB, AWS S3, CloudWatch Events, CloudWatch logs, SNS

and Incognito

## Key Management Service (KMS)

**AWS** Key Management Service (**AWS KMS**) is a managed service that makes it easy for you to create and control customer master keys (CMKs), the encryption keys used to encrypt your data.

## AWS secrets

AWS Secrets Manager helps you protect secrets needed to access your applications, services, and IT resources. The service enables you to easily rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle.

## Relational Database Service (RDS)

Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the cloud.

Amazon RDS can automatically back up your database and keep your database software up to date with the latest version

**Which relational database engines does Amazon RDS support?**

**Amazon RDS database engines:**

* Amazon Aurora
* PostgreSQL
* MySQL
* MariaDB
* Oracle
* Microsoft SQL Server

**Encryption in RDS:**

Encryption at rest is supported for

* Amazon Aurora
* PostgreSQL
* MySQL
* MariaDB
* Oracle
* Microsoft SQL Server

**Q: Can we enable encryption on exciting DB**

Encrypting existing DBs is not supported. To do this, you’ll need to create a new

encrypted instance and migrate data to it. The encryption key can be stored in KMS.

Q: **Which is the non-relational database supported in AWS**

Amazon DynamoDB is the NoSQL database supported by AWS

## Cognito

* It Mainly provides authentication authorization and user management for your application
* It provides a managed user pool to mange identity for the application

Cognito provides user flows:

* Signup
* Signin
* Forgot or change password
* Multifactor authentication
* Email and phone verification

It also provides software development kit to your mobile or web application, and also provide lambda triggers in order to customize any of these user flows with you own business logic

It also provides a built-in hosted UI for these user flows

Social identity can be integrated

Facebook

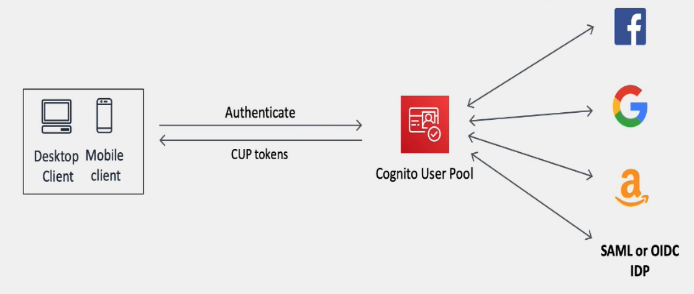
Google

Amazon

SAML

After authentication the user the Cognito provides the best practice way of accessing the AWS resources securely from the app by providing temporary credentials

### User Pool:



* User pools acts as mediator between your app and external

social identity providers

* you can add multiple identity providers as you need.
* The user pool manages the token exchange with each of the

providers and gives your app standard user pool tokens of

same format

### Identity poll:

Diagram

Description automatically generated

* Where you exchange the authentication token to get temporary aws credentials which you can use to access the resources directly from the app
* These can be used independently of each other or used together

### Difference between user pool and identity pool

**AWS Cognito User Pools** is there to authenticate users for your applications

Say you were creating a new web or mobile app and you were thinking about how to handle user registration, authentication, and account recovery, you don’t need to implement user authentication inside your application, rather you can integrate AWS Cognito User Pools, which will manage user sign-up, sign-in, password policies.

**AWS Cognito Identity pool**:

* This is a service which was designed to authorize your users to use the various AWS services. The source of these users could be a Cognito User Pool or even Facebook or Google.

In other words, Identity Pools are used to assign IAM roles to users (who had been authenticated through a separate Identity Provider which could be Cognito User Pools or Social logins (e.g; Gmail, Facebook & etc.)). Because these users are assigned an IAM role, they each have their own set of IAM permissions, allowing them to access AWS resources directly.

So, the difference is

* AWS Cognito User Pools: Granting access to a application
* AWS Cognito Identity Pools: Granting access to amazon service

## Difference between IAM and Cognito

**AWS IAM** gives securely and control access to AWS services and resources for your users

**AWS Cognito** It Mainly provides authentication authorization and user management for your application

# IAAS vs PAAS vs SAAS

Graphical user interface, text, application

Description automatically generated

# SNS (Simple Notification Service)

Create SNS

\* In AWS console go to SNS

\* Click on Create Topic and enter all the required details

\* Define who can publish messages to topic and who can subscribe to topic in access policy

\* Then create subscription

\* Click on create subscription enter the ARN of topic

\* Select the endpoint like http,email or sms and enter the details under endpoint block like email-id or phone number.

**Create mobile push notification using SNS**

\* Click on mobile push notification

\* Create a application platform

\* Enter name of application

\* Select platform of the application weather its a ios application or google application

\* Then create application endpoint by entering the device token and user id

\* If the notification has to be sent to many subscribers then copy the ARN of the platform application if incase it has to be sent to single subscriber then copy the ARN of application endpoint.

# VPC(Virtual Private Cloud)

**Steps to Create VPC**

\*Click on create VPC

\*Enter cidr block details

\*Click on create vpc

**Steps to create subnets**

\*Click on create subnets in vpc

\*select the created vpc

\*Enter name for the subnets and select the availability zone

\*Enter the CIDR block

\*Click on create subnets

**Steps to create route table**

\*Go to VPC, click on create route table

\*Enter name of the route table

\*Select VPC

\*Create route table

\*Then select route table and click on subnet associations

\*Click on edit subnets associations then select the subnets save associations.

**Steps to create Internet Gateway**

\*Go to VPC and click on create internet gateway

\*Enter name for IGW

\*Click on create IGW

\*Then go to route table select route table click on routes and edit routes

\*Add routes enter 0.0.0.0/0 and enter IGW name and save changes

**Steps to create NAT Gateway**

\*Go to vpc select create nat gateway

\*Enter name for nat gateway and select public select and allocate elastic ip and click on create nat gateway.

\*Go to route table and select private subnet and click on routes and edit routes

\*Enter 0.0.0.0/0 and enter nat gateway name and save changes.

# ELB (Elastic load balancer)

Types of load balancer

\***Application load balancer**: It works at request level and it is used for web application which uses http and https and it operates at the 7th layer .

**How to configure application load balancer**

\*Application load balancer is used for micro service application and can be routed to the application in two ways either by mentioning the path or the port number for example if the hhtp url consists of path it will be redirected to the specific micro service or if consists of port number it will be redirected to the specific microservice.

**Steps to create application load balancer**

\*Create a ec2 instance and under configuration block enter the required script of the application

\*Then create a target group and select instance as target types and http as protocol and 80 as port number and select the vpc

\*Then enter the path of the application

\*Once the target group is created u have to register the target go to target and select the instance it will be registered to the target

\*After this go to load balancer and click on create application load balancer and give name to application load balancer and scheme should be internet facing so that it can be accessed over the internet else select internal in scheme.

\*Listeners should be http and port number should be 80

\*Select the vpc and availability zone

\*Then select the existing target groups and click on create load balancer.

# NETWORK LOAD BALANCER:

It works at connection level and routes the traffic within the vpc and it can accept millions of request per second and it operates at the 4th layer.

**Steps to create network load balancer**

\*Create a ec2 instance and under configuration block enter the required script of the application

\*Then create a target group and select instance as target types and http as protocol and 80 as port number and select the vpc

\*Then enter the path of the application

\*Once the target group is created u must register the target go to target and select the instance it will be registered to the target

\*After this go to load balancer and click on create network load balancer and give name to application load balancer and scheme should be internet facing so that it can be accessed over the internet else select internal in scheme.

\*Listeners should be tcp and port number should be 80

\*Select the vpc and availability zone

\*Then select the existing target groups and click on create load balancer.

# Static Web Page

A static website consists of a series of HTML files, each one representing a physical page of a website

**Configuring Your S3 Bucket for Static Website Hosting**

* + Navigate to S3 in the AWS Console.
  + Click into your bucket.
  + Click the “Properties” section.
  + Click the “Static website hosting” option.
  + Select “Use this bucket to host a website”.
  + Enter “index.html” as the Index document

# 3 Tier Architecture

* + **Presentation Tier**: This layer handles the presentation of data to end-users and interacts with the user interface. It includes web servers, content delivery networks (CDNs), and load balancers.
  + **Application Tier**: This layer handles the business logic of the application and includes application servers, middleware, and APIs.
  + **Data Tier**: This layer stores and manages the application data, and includes database servers, file servers, and storage systems.

**Here are the steps to set up a three-tier architecture in AWS:**

* + **Create a VPC (Virtual Private Cloud) in AWS**: A VPC allows you to create an isolated network environment in AWS where you can launch your resources. You can create a VPC using the AWS Management Console or AWS CLI.
  + **Create subnets**: Create subnets within the VPC to isolate the different layers of the architecture. Typically, the presentation tier will be in a public subnet, and the application and data tiers will be in private subnets.
  + **Create security groups**: Create security groups to control traffic between the different layers of the architecture. For example, you can allow traffic from the internet to the presentation tier but restrict traffic from the presentation tier to the data tier.
  + **Launch EC2 instances**: Launch EC2 instances in each subnet to represent the different layers of the architecture. For example, you can launch a web server in the presentation tier subnet, an application server in the application tier subnet, and a database server in the data tier subnet.
  + **Configure load balancing**: Configure a load balancer to distribute traffic between multiple web servers in the presentation tier. AWS provides load balancing services such as Application Load Balancer (ALB) and Network Load Balancer (NLB).
  + **Set up databases**: Set up databases in the data tier subnet using services such as Amazon RDS or Amazon DynamoDB.
  + **Connect the layers**: Connect the layers of the architecture using APIs, web services, and other methods to enable communication between the layers.
  + **Monitor and optimize**: Monitor the performance of the architecture and optimize it for better scalability, availability, and cost efficiency.