**Concept : Load Balancer**

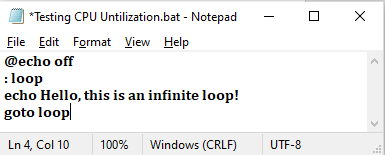
**Let Understand EC2 load balancing by taking CPU load capacity as example:**

Open **Task Manager** window-> click on **Performance**

Here you see the CPU utilization on running processes.

**Let see CPU utilization by increase the load on CPU:**

* Create the following windows batch file in Notepad

****

* Open the above **.bat** file multiple times, you can see the increased **CPU utilization** at **Task Manager** under “**performance**” in every time you open the **.bat** file.

**NOTE :**

A **.BAT (batch) file** is a script file in Windows that contains a series of commands to be executed by the **Command Prompt (cmd.exe)**. It is used to automate repetitive tasks.

To see the code – right click on the **.bat** file and select **Edit** option.

**@echo off** - This hide command execution details from appearing in the console.

If not used, the **echo** Hello, this is an infinite loop! command itself will

be displayed before execution/output.

**:loop** - This is a **label** in the batch script. Labels start with a colon (:) and act as a

reference point for the **goto** command.

**goto loop** -This sends execution back to the :loop label, creating an **infinite loop**.

The script keeps running indefinitely by printing **"Hello, this is an**

**infinite loop!"** until the user manually stops it.

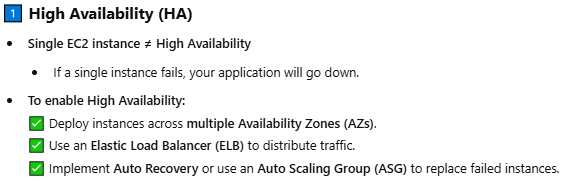
(e.g., Stop execution by pressing Ctrl + C,

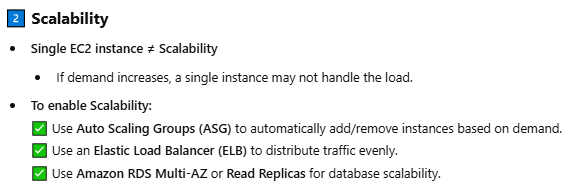
close the command prompt manually,

**End the process via Task Manager** (cmd.exe under "Processes").

**Ensuring Availability and Scalability on EC2 instance:**

Launching a single EC2 instance **does not** automatically provide **high availability** and **scalability**. You need to configure AWS services to achieve these features.





**Scaling / Auto Scaling :** Scaling up server hardware in order to make the application available to users when higher number of requests are made.

**Types of scaling:**

1. **Vertical scaling** –

Increasing **RAM and CPU** capacity to handle increasing application Requests.

Advantages - High Availability

Disadvantages - There will be application down time while changing the RAM and CPU

capacity where we may need shut down the server/EC2.

- If the hardware/motherboard does not work , application likely become

unavailable, leading to extended downtime for users.

- Increasing RAM capacity is directly tied to the hardware capabilities of

your device. RAM capacity is determined by the motherboard's

specifications and the limitations of the processor (CPU).

- Hence before upgrading, check the max RAM capacity supported by

your motherboard and the type of RAM (e.g., DDR3, DDR4) compatible

with your system.

1. **Horizontal scaling –**

Increasing application servers to handle increasing application requests.

Advantages - No down time even when one application server had hardware failure.

- High availability, fault tolerance,

- No limit on increasing application servers

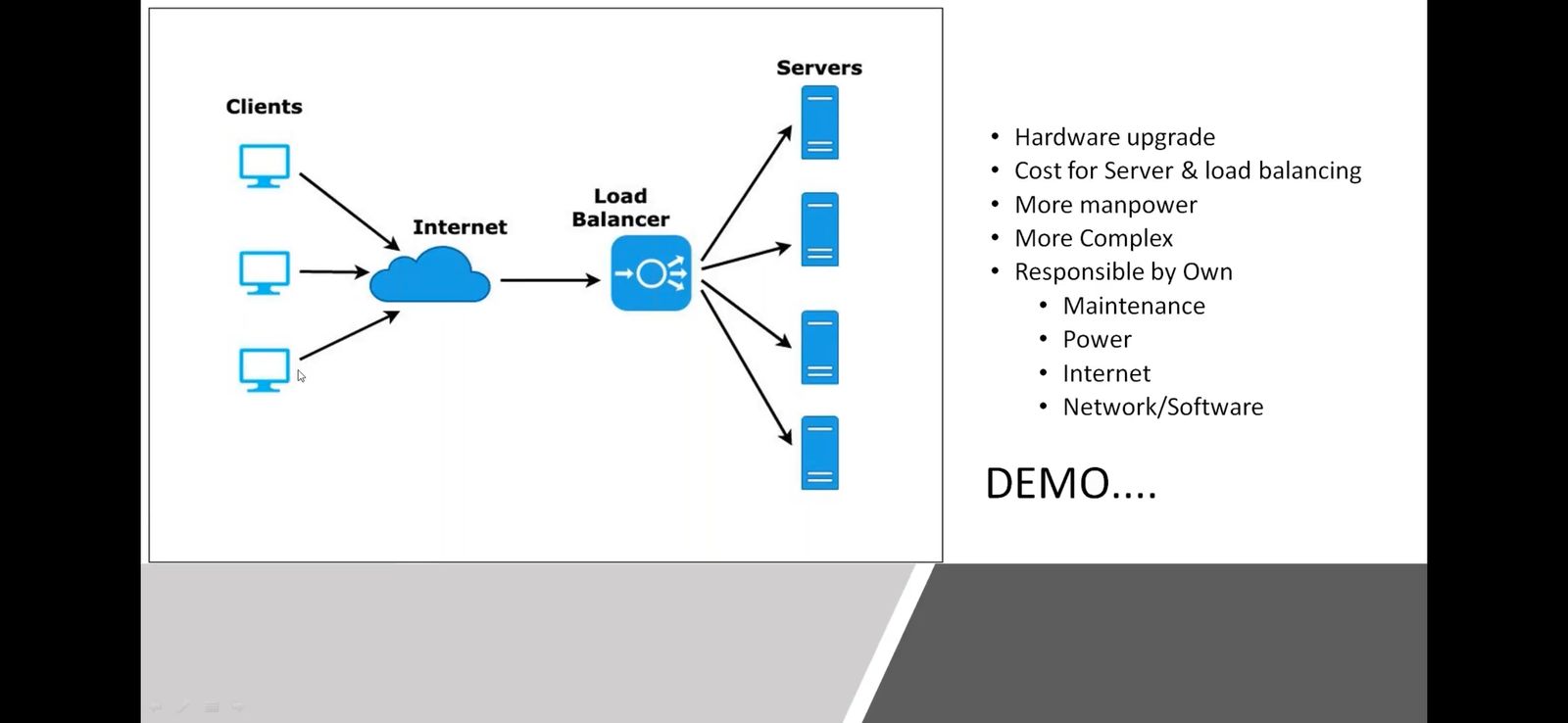
Disadvantages - Managing and coordinating with all servers ( Say, updating or installing

software ) can be more complex.

- Cost

**Elastic Load Balancer (AWS will create, install & maintain the Load Balancer)** :

The process of evenly distributing incoming network traffic or workload across multiple servers.



Load balancers act as intermediaries between clients (such as web browsers or applications) and servers.

Clients hit the IP address of a load-balancer which routes the client request to associated application server

They monitor incoming requests and determine how to distribute them among the available servers.

Uses Round Robin method, which is an algorithm that distributes work evenly across servers.

**NOTE:**

**If Load-Balancer**(which itself is a hardware) **experience a failure**, then all applications become unavailable.

Hence **AWS sets up multiple load balancer instances** in an active-passive or active-active setup.

In an active-passive setup, one load balancer handles traffic while the other serves as a standby backup. If the active load balancer fails, the standby backup takes over automatically.

In an active-active setup, both load balancers share the traffic load, and if one fails, the other can handle the entire workload.

**Remember,** AWS is providing Load balancer as its service called Elastic Load Balancer (ELB).

**Creating Load Balancer :**

**NOTE :**

For practice, either "**launch**" couple of new EC2 instances where web application is hosted or

"**start**" couple of existing EC2 instances where web application is hosted,

In order to attach them(EC2) to a Load Balancer.

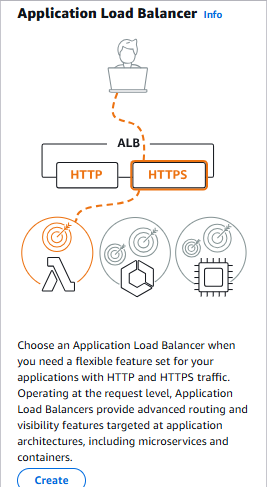
We have launched two EC2 instances in public subnets of two availability zones in order to

expose them(EC2) to world.

**Remember,** to have high application availability create EC2 instances at differently availability zones

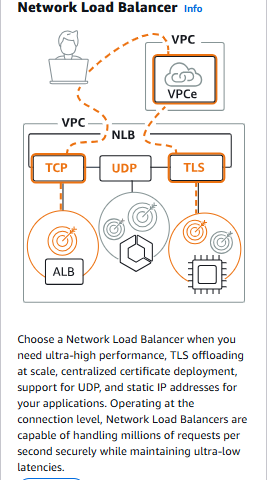
1. Open **Load Balancers** under **EC2 -> Load Balancing** dashboard
2. Click on "Create Load Balancer" button

Here we see, **4 Types of Load Balancers available in AWS**:

**Application Load Balancer :**

* + It works at Application Layer 7 in OSI model.
  + It uses Advance routing mechanism and directs incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, IP addresses, or Lambda functions, based on the content of the HTTP/HTTPS requests.
  + EC2 instance, IP addresses, Lambda function another Application Load balancer can be added to Application load balancer as its Target Groups
  + It is "region-based" – that means when you create an ALB, it is tied to a specific AWS region, and it can only route traffic to backend targets within the same region.

**Network Load Balancer : Gateway Load Balancer :**

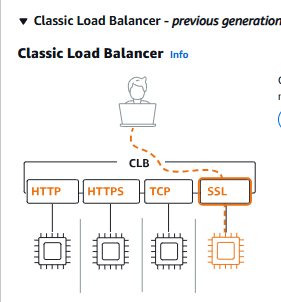
****

**Classic Load Balancer :** Choose a Classic Load Balancer when you have an existing application

running in the EC2-Classic network.

**NOTE :** **Classic Load Balancers** can't route to **target groups** as **Application**

**Load Balancer** does.

****

* + - It primarily operates at the Transport Layer (Layer 4) of the OSI model
    - Low in usage as it uses simple routing mechanism based on port number and protocol for incoming requests.
    - It uses old classic Round-Robin Algorithm which evenly distributes the workload to servers.
    - Distributes incoming application traffic across multiple EC2 instance targets in multiple Availability Zones.
    - Only EC2 instance be added to Classic Load Balancer

**Why is Classic Load Balancer (CLB) Considered Previous Generation?**

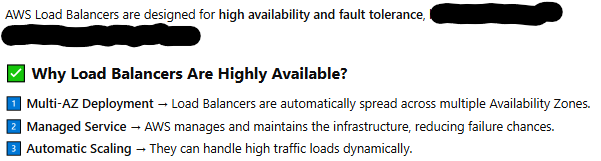
* + - Limited Layer Support - CLB supports both Layer 4 (TCP) and Layer 7 (HTTP/HTTPS) but lacks advanced features.
    - Lack of Advanced Routing - **CLB** cannot perform **host-based** or **path-based routing**. It **only distributes traffic based on Layer 4 (TCP/SSL) or Layer 7 (HTTP/HTTPS) protocols, but does not inspect the request path (e.g., /sales).**
    - Poor Scaling & Performance – **CLB s**lower in **handling millions of requests**.
    - **CLB does** **not** support **Web Sockets** or **HTTP/2**.
    - **No Static IP Support – CLB uses** a **dynamic IP**, making it harder to whitelist for security.
    - AWS Recommends ALB(for **modern web applications**) & NLB(for **low-latency, high-performance applications**) for New Workloads. Both provide **better security, performance, and scalability**.

**How Classic Load Balancers work :**

* + - Clients make requests to your application.
    - The listeners in your load balancer receive requests matching their protocol and port.
    - The receiving listener forwards requests to healthy registered instances using the routing algorithm of the listener protocol. You configure the health checks that monitor the health of the registered instances.
    - The healthy instances receive traffic. You can register or deregister instances to your load balancer either manually or by way of an autoscaling group without disrupting the overall flow of requests to your application.

**However, for learning purpose best practice to create classic load balancer:**

**Remember,** to have high application availability create EC2 instances in a VPC at differently availability zones.



1. Click on "create" button of **Classic Load Balancer – Which opens a template**
   1. **Basic configuration** 
      * Load balancer name : Give a name to load Balancer as you prefer.
      * Scheme - **Scheme can't be changed after the load balancer is created.**

Selectschema based on the requirement whether your load balancer exposed to world or within organisation:

* + - * select "Internet-facing" **load balancer** -

This routes the request from clients over internet to targets.

This requires to be in Public Subnet that has public IPs.

* + - * select "Internal" **load balancer** –

This routes the request from clients to targets/servers using

Private IP address.

* 1. **Network mapping -** The load balancer routes traffic to targets in the selected

subnets, and in accordance with your network settings.

* + - VPC - Select the VPC where your EC2 instances are running.

The selected VPC cannot be changed after the load balancer is created.

**NOTE:**  **Before you begin**

* When selecting a VPC for your load balancer, ensure each subnet has a CIDR block with at least a /27 bitmask and at least 8 free IP addresses.
* If the subnet is too small, the **Load Balancer might fail to launch** due to insufficient IP addresses.
* Subnet CIDR is **fixed** within the VPC and **cannot be changed**.
* Launch the EC2 instances that you plan to register with your load balancer. Ensure that the security groups for these instances **allow HTTP access on port 80.**

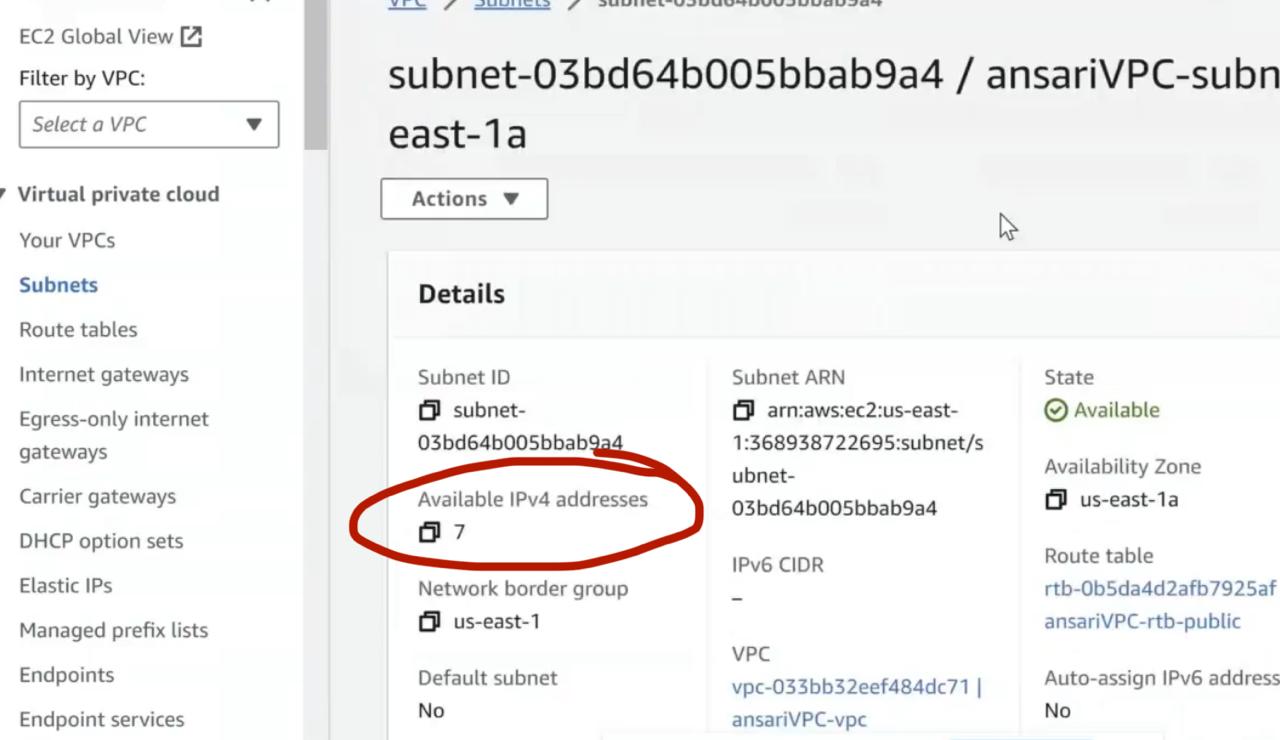
**How to ensure that** **enough IP space are available in** **Subnet(subnet\_ID) :**

Copy the **subnet\_ID** and went to "**subnets**" under **VPC dashboard**

Filtered the subnet from existing subnets list by pasting the **subnet\_ID**

Open that subnet where we see " 7 "Available IPv4 addresses and IPv4 CIDR

size is "10.0.0.0/28".



This way we ensure that selected subnet’s available IP addresses**.**

**Solution:** As subnet CDIR is fixed create new subnet in selected VPC and enable load balancer to that subnet.

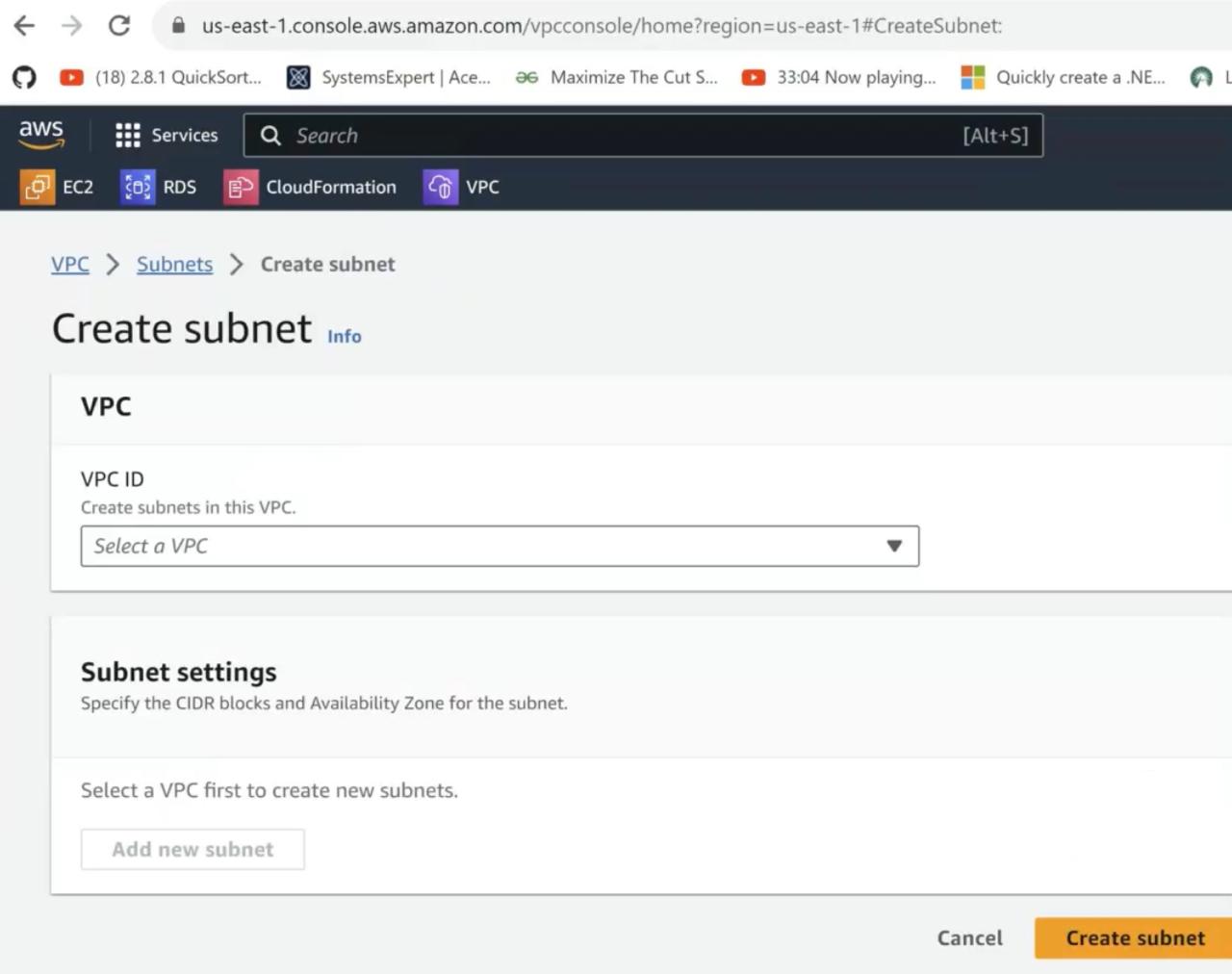


Fig : Open **Subnets** in **VPC** dashboard and create new subnet.

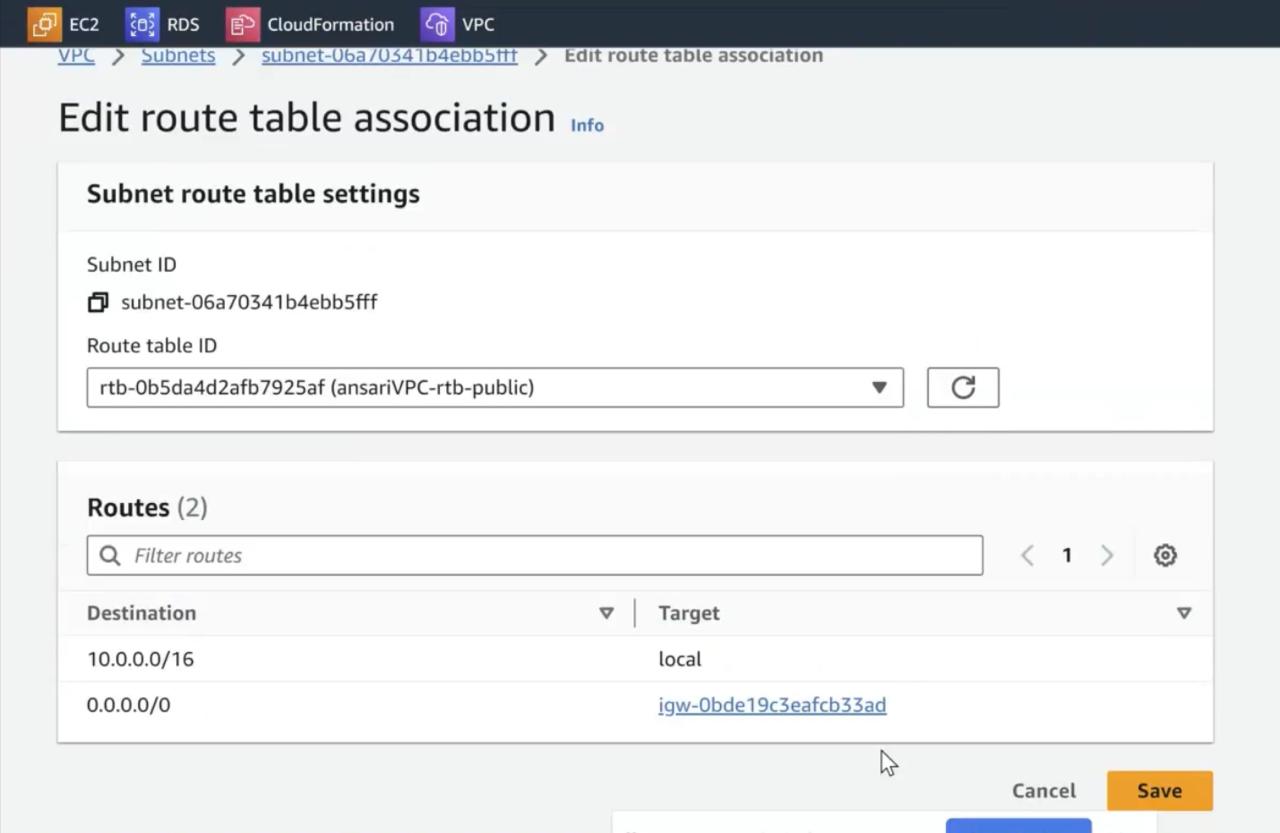


Fig : Open newly created subnet -> Round Table -> Edit route table association ->

Select the igw Route table ID as load-balancer require public subnet which has a

route to an Internet gateway(igw) then we must associate the new subnet with

route table(igw) which allows internet.

* + - Mappings - refers to how incoming requests are mapped or routed to specific backend resources by load balancer. **The load balancer will route traffic only to targets in the selected Availability Zones**.

Select **at least 1 or 2 Availability zones** where your EC2 instances are running and Select **1** **public subnet** for each Availability Zone.

**NOTE :**

* **AWS shows all the AZs**, even if they don't have any instances running in an AZ (let say AZ-c).
* However, for traffic routing to work, there must be at least **one** **public subnet in that AZ**.
* AWS assigns an **Elastic Load Balancer (ELB) node** in AZ-c.
* If there are no instances in AZ-c, the ELB will still **exist** but won’t have any targets to forward requests to.
* At Instances section in **classic Load balancer console** , AWS shows all instances in your **entire VPC**, even if they are in a different AZ (let say AZ-a, AZ-b).
* **Load balancer won’t allow traffic to reach the AZ-a & AZ-b instances unless AZ-a & AZ-b are enabled** in the Load Balancer.
* If you only enable AZ-c in the Load Balancer but have no instances in AZ-c, the Load Balancer will be active in AZ-c but **no traffic will be forwarded to your running instances**.

**Best Practices :**

* **Always select the AZs where your instances are running** (AZ-a and AZ-b in this case).
* Ensure that the subnets selected for the Load Balancer are **public subnets** with proper Internet Gateway (IGW) access.
  1. Security groups – Set of firewall rules that control the traffic to your load balancer.
     + **select an existing security group** that is configured to allow the required HTTP traffic on port 80 **or you can create new security group**.
     + If none are selected, the VPC’s default security group will be applied.
     + To ensure the security rules defined in VPC’s default security group
       - Copy the **security group ID** and goto "**Security Groups**" under "**Network & Security**" at **EC2 dashboard**
       - Paste the **ID** and view that security group by clicking on its **ID**
       - check the "**Inbound**" rules, where you see **all network type**s, **all protocols**, **and all port ranges** **are allowed**
  2. Listeners and Routing
     + A listener is a process that **checks for connection requests** using the **protocol and port you configure**.
     + The settings you define for a listener determine **how the load balancer routes requests to its registered targets**

When you add a listener, you specify a protocol and port used to check for connection requests(front-end), along with a protocol and port used to send those requests to your targets(back-end). When a connection request is received on the specified protocol and port, the listener forwards the request to a healthy, registered instance.

**Listeners for Classic Load Balancers support the following protocols and ports:**

* Protocols: HTTP, HTTPS, TCP, SSL

1. Ports: 1-65535

A **port number** is a unique identifier assigned to network connections, allowing devices and applications to communicate over the internet or a local network. **Firewalls & security groups** control access to ports.

Listener port : This is the port on which the load balancer listens for incoming

requests.

Instance port : This is the port to which the load balancer forwards incoming

requests to your EC2 instance

In security group, open the load balancer port number to which client can make a request.

**NOTE:**

We can configure the listener and instance ports differently if you want to keep the port numbers of your application hidden from clients.

**Example:** You can configure the load balancer to listen on a standard port like 80 for

HTTP traffic. This is the port that clients will connect to when accessing

your application.

You can configure the load balancer to forward incoming requests to your

EC2 instances on a diff port, such as 8080, where your app is running.

This way, clients won't directly interact with your application servers on port 8080;

instead, the load balancer will handle the communication.

* Health checks - The load balancer automatically performs health checks to test the availability of all registered instances. Traffic is only routed to healthy instances.

If the instance health fails, CLB will **stop sending traffic** to that instance.

* Ping Target : The health check ping is sent using the **protocol and port** you specify. If using HTTP/HTTPS protocol, you must also provide the destination **Ping path( a specific URL or endpoint on your backend servers that the load balancer sends health check requests.)**
* The **ping path (e.g., /index.html) is only used for health checks**, not for routing. CLB sends periodic **HTTP requests to this ping path** to check if the instance is healthy.
* This endpoint should return a specified response (e.g., HTTP 200 OK) to indicate that the server is healthy and able to handle incoming requests.
* **CLB health checks are performed on the port that your web application listens on.**

**NOTE : Health checks can be configured at the instance level (EC2) or application level**

**(specific path).**

You can configure health checks directly on the **EC2 application’s port (80 for**

**HTTP)** instead of specifying a **ping path**.

**Example:**

* if your backend servers are running a web application on port 80,

the load balancer may send health check requests to port 80 of each

server to verify their health and availability.

* if your web application has a health check endpoint at "/health",

the load balancer would send ping requests to this path (e.g.,

http://example.com/health) to check the health of the backend servers.

* **According to our EC2 applications, set the ping path to "/index.html" or " / ".**
* **Ensure all the load balancer targets/instances at diff Availability Zones must have provided ping path otherwise load balancer will perform health checks only on targets that has provided ping path.**
* Click on "Advance health check settings" ,
  + Here we can increase the time between the health check sent to EC2 and time within the EC2 should respond to health checks.
  + We can specify the number of health checks should be performed in order to determine whether the instance is health/unhealthy
* At Instances - You can add instances to register as targets of the load balancer.

**You will see all instances running in the selected VPC**, even if they

are in private subnets.

It is **your responsibility** to select instances that are in the correct

subnets for load balancing.

* + - Click on "Add instances" button
      * Select EC2 instances to register to your load balancer.
      * Start the instances if they are stopped.
        + Requests will be routed to registered instances that meet the health check requirements.
        + For maximum fault tolerance, AWS recommend maintaining approximately equivalent numbers of instances in each Availability Zone enabled for the load balancer.
        + If demand on your instances changes, you can register or deregister instances without disrupting the flow of requests to your application.
    - Click on "confirm" button
* At Attributes - Creating your load balancer using the console gives you the opportunity

specify additional features at launch.

You can also find and adjust these settings in the load balancer’s “Attributes” section after your load balancer is created.

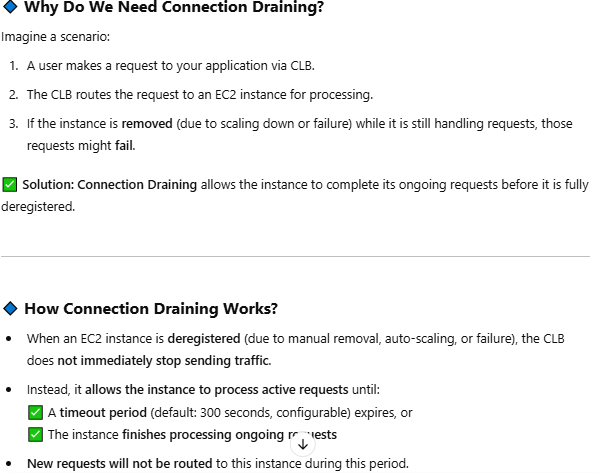
Here AWS auto enable the following two options,

* Enable cross-zone load balancing - This enables the load balancer to distributes requests evenly across registered instances in all enabled AZ, otherwise it only distributes the requests evenly across instances in its AZ.

However, my EC2 instances present in Two Availability Zone, my load balancer should applicable to both zones.

* Enable connection draining -

1. Timeout(draining interval) – AWS set it to 300 sec



* Load balancer tags – Tags helps you to categorize your AWS resources

Ex: Key = webserver, and value = production.

* Click on "create load balance" button.
* Click on "view load balancer" which takes us to "Load balancers" template at EC2 dashboard.
  + Open the newly created Load Balance where we see the metadata of it and its DNS name provided by AWS.
    - Client can use this DNS name to hit a request to Load balance which route the request to registered application servers.
    - At "Target instances", **refresh** it and **check** the instance's "Health status" should say "**In-service**" which means instance is healthy. If it says “Out-of-service” then it means instance is not working.
    - At "Monitoring ",provides health and performance of our load balancer in graphs.
    - **Copy the load balancer’s DNS name and run in browser**, hence the requests are evenly forwarded to registered instances via load balancer.
* **Copy the LB DNS name and run it in browser. LB will evenly forward the requests to registered EC2 instances. If you made continuous server requests to LB you can see the even responses from two registered EC2 instances. Make sure the registered instances are running.**

**NOTE:** Remember EC2 and Load Balancer are chargeable in AWS.

Terminate the running instances(open Instances at EC2 dashboard – select instance – Actions)

Delete Load balancer(open Load Balancers at EC2 dashboard – Select LB - Actions – Delete)

**Creating Application Load Balancer:**

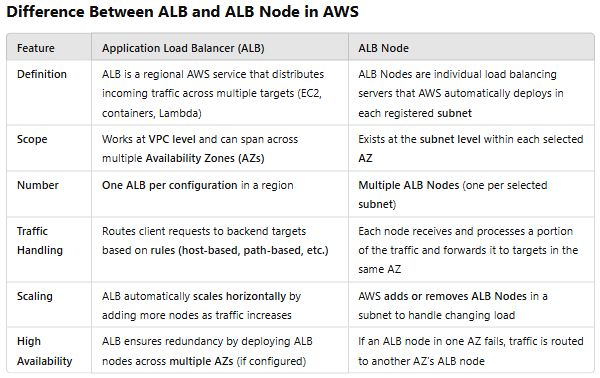
It works on layer7 which is an Application Layer in Open Systems Interconnection(OSI) model.

Uses the protocols : HTTP(default port 80) & HTTPS(default port is 443)

ALB is region based : ALB is deployed **within a region**, but it distributes traffic **only across the**

**Availability Zones (AZs) that you enable**. ALB **cannot distribute traffic across**

**multiple regions**.



* **Single ALB is created** at the **VPC level** and spans across multiple **Availability Zones (AZs)**.

ALB **ఒకే VPC లో పనిచేస్తుంది** కానీ **అదే ALB** **వివిధ Availability Zones (AZs) లో ALB Nodes ను create చేయగలదు**.

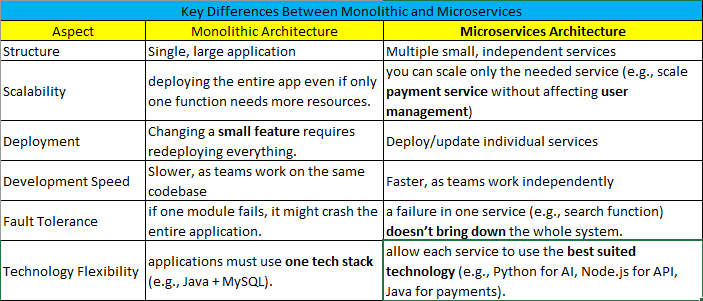
* You must register at least **one public subnet per selected AZ** for the ALB to distribute traffic properly.
* AWS **automatically provisions an ALB node** in each **selected subnet** to handle traffic. If traffic increases, AWS **adds more ALB nodes dynamically** within each registered subnet.
* You don’t **manually create or manage ALB Nodes**—AWS **automatically scales** them as needed.

### **Example Scenario**

🔹 Assume you have a VPC with **3 Availability Zones (AZ-a, AZ-b, AZ-c)**.  
🔹 You create an **ALB and** register **public subnets in AZ-a and AZ-b**.  
🔹 AWS automatically **creates an ALB node in each subnet**.  
🔹 When a user sends a request to the **ALB DNS name**, AWS **routes the request to one of the ALB nodes**.

🔹 If AZ-c **has no registered subnet**, it **won’t participate in load balancing**.

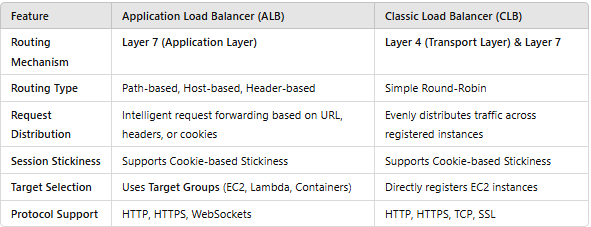
**Why do we choose Application load balancer over classic load balancer?**

****

In context of **microservices architecture**, the Application Load Balancer (ALB) is generally more suitable and accurate compared to the Classic Load Balancer (CLB).

Because **ALB supports path-based routing, content-based routing** allowing us to route traffic to different microservices based on **end-points/URL path**, **host header**, **HTTP request method content**, **query parameters**, or **headers**.

Whereas, **CLB doesn’t support Path-based Routing** and it supports routing based on Port numbers



**“ Target Group “** concept in ALB- a target group is a logical grouping of targets/EC2 instances to receive incoming traffic from the load balancer.

Target Group **Example Scenario :** In microservices architecture, **we can group same EC2 micro services**(say two “example.com/cart ” EC2 application and register them as target group to **ALB** in order to forward traffic to them(EC2).

**Creating Target Groups before creating ALB: You can either create TG before or while creating ALB**.

* 1. Open "**Target Groups**" under "Load Balancing" at **EC2 dashboard**

**Target Groups :** are used to route traffic to a group of targets.

**Example :**

* + - If you have multiple EC2 instances running a **Sales App**, you can create a **Sales Target Group** and attach it to the ALB. **The ALB will route traffic only to those instances.**
    - Suppose you have a multiple EC2 instances running **Sales App** and other EC2 instances running **Marketing App**. You can create two separate **Target Groups**(Sales-TG, Marketing-TG).
      1. Then, you can configure **Path-Based Routing** in the ALB to send **/sales** requests to Sales-TG and **/marketing** requests to Marketing-TG.
  1. Click on Create target group button.
     + A Specify group details template will open - Your load balancer routes requests to the targets in a target group and performs health checks on the targets.
       1. Basic Configuration – Settings can’t be changed after TG is created.
          - At Choose a target type –

we can add **EC2 instances** / **Ip addresses** / **Lambda Function** / **Application Load Balancer** as Target Groups to our Load Balancer.

**For practice,** we have selected "**instances**" option.

* + - * + Target group name - Give a name to your Target Group.
        + Protocol : Port - Choose a protocol for your target group.

Example: HTTP : 80

**NOTE :** The protocol and port you specify here should match the protocol and port that your backend application is listening on.

* + - * + IP address type –

Only targets with the selected IP address type can be registered to this target group.

IPv4

IPv6

* + - * + VPC - Select the VPC with the instances that you want to include in the target group.
        + Protocol Version – we have selected **HTTP1** option.
      1. At "Health checks" –
         * Health check protocol - Select protocol of application/service running in target.
         * Health check path - Use default path "**/** " to ping to root or specify custom path

like "**/index.html** ".

For one target group we configured as " / " as our EC2 APP has path "/ " and,

For another target group, we configure as "sales/sales.html" as our another

EC2 APP has path "sales/sales.html".

**NOTE:**

ALB health checks are typically performed at the application layer (Layer 7) of the OSI model.

ALB sends HTTP or HTTPS requests to a specified endpoint on the backend instances and

expects a specific response (e.g., HTTP 200 OK) within a defined timeout period.

CLB health checks operate at the transport layer (Layer 4) of the OSI model.

CLB sends TCP or SSL/TLS handshake requests to the backend instances on a specified port

and expects a successful response within a defined timeout period.

CLB doesn’t inspect the content of the responses from the backend instances. It only verifies

whether the TCP connection has established successfully within the defined timeout period.

* + - * + Advance health check settings –

**Health check port** - Enable "Traffic port" which is the default port in AWS that a load balancer uses when performing health checks on targets.

**Healthy threshold** – The number of consecutive health checks successes required before considering an unhealthy target healthy.

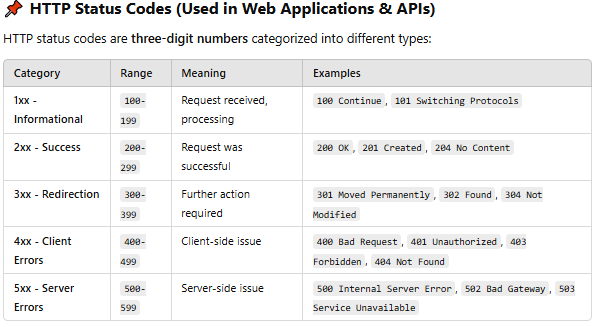
**Unhealthy threshold** – The number of consecutive health check failures required before considering a target unhealthy.

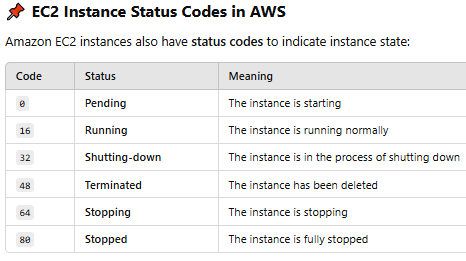
**Timeout – In specified time (Seconds)** response రాకపోతే, **అది Failed Health Check గా పరిగణించబడుతుంది**.

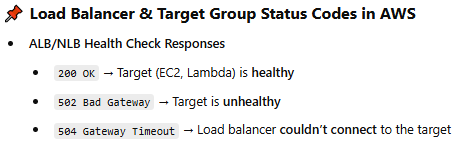
**Interval** - The amount of time between health checks of an individual target

**Success code –** Add the success response status code

A **status code** is a standardized numeric response from a server indicating the outcome of a request. These are commonly used in HTTP, networking, and system responses.







* + - 1. Click "Next" button.
      2. A "**Register targets**" template will open – Register your targets to ensure that your LB routes traffic to this TG.
         * At Available instances – Select the instances which you want to add them as 'target group'.
      3. Click on "Include as pending below" button.
      4. At Review targets template – The selected instance will be added as TG.

Subsequently, Load balancer begins health checks ( sending requests to a

specified endpoint of the registered instance (e.g., /health ).

Hence At "Health Status" - we can check the status of availability of an instance.

* + - 1. Click on Create target group button – A target group will be created.

**NOTE :** Repeat the same above steps to create another **Instance** typed Target group.

For one target group we configured as " / " as our EC2 APP has path "/ "

and,

For another target group, we configure as "sales/sales.html" as our another

EC2 APP has path "sales/sales.html".

**NOTE:** Additional charges on **Load Balancer** and **Target Groups** and **EC2** apply by AWS.

If TG associated with ALB you must first delete the Load Balancer at Actions dropdown,

Then delete the TG.

**How to check status of Target Group:**

1. Open **Target** groups under **EC2 Load balancing** dashboard.
2. Click on **Target Group name** you specified – Open a template showing TG details.

**TG details:**

* "None associated " at Load balancer tag means we haven’t attached this target group to a load balancer
* Shows NO. of **Total targets**, NO. of **Healthy** targets, NO. of **Unhealthy** targets, No. of targets Initialized.

**ALB creation steps:**

1. Open "**Load Balancers**" console at **EC2 dashboard**.

2. Click on "**Create**" button of **Application Load Balancer**

1. **Basic configuration**
   * + At **Load Balancer name** - Give a name to your Load Balancer
     + At **Scheme** – Select either of the following
       - Select "**internet-facing**" - if you want to expose your EC2 micro-services to public.
       - Select "**internal**" -if you want to expose your EC2 micro-service within organisation.
     + **Load Balancer IP address type** – Select the front-end **IP address type** to assign to the LB.

For bath practice and production level, best select is **IPv4.**

The **IP address type** determines the IP address versions(IPv4 & IPv6) that can be used to communicate with your load balancer. **Public IPv4 addresses have an additional cost.**

**The following IP address types are available for Application Load Balancers:**

* **IPv4** – includes only Public and private IPv4 addresses.
* **Dualstack – includes public and private of both IPv4 & IPv6 connections.**
* **Dualstack without public IPv4** – includes Public and private IPv6 connections, only

private IPv4 connections. Compatible with **internet-**

**facing** load balancers only.

1. At **Network mapping**
   * + - **VPC** - Select the VPC ( Select the VPC where yours load balancer's backend instances present)
       - At **Mappings** – Select At least two AZs and one subnet per zone. LB routes traffic to targets in these AZs only.

**NOTE:** **classic LB** allow us to add Instances **from at least 1 or 2 Availability Zone** to

our load Balancer.

Whereas, **Application LB** allow us to add Instances **from at least 2 Availability** **Zone** to

our load Balancer.

**Always ensure to select those AZs where your instances are running.**

1. At **Security Groups** – it is a set of firewall rules that control the traffic allowed to your **LB**.

For practice , we have selected "**Default**" Security Group of VPC which

allows all traffic.

1. At "**Listener and Routing**" – A listener is a process that checks for connection requests using the port and protocol you configure.

Here we configure the **Target Group** and **protocol & port of an application instance** to determine how the load balancer routes the http requests to its targets.

**NOTE :** The protocol and port you specify here should match the protocol and port that clients will use to access your application.

* + - **Here default** **Listener – HTTP:80**

**Here we configure, Default action** – Select a target group which receive the traffic.

**You can either create a target group while creating ALB or you**

**can create TG and attach after creating ALB.**

Click on Create target group – to create new TG.

* + - Click on **Add Listener** to add more listeners

1. Load Balancer tags – Tags helps you to categorize AWS services/resources to manage them.
2. Optimize with service integrations – Additional charges will be applied.
   1. **Amazon CloudFront + AWS Web Application Firewall (WAF)**

Integrating your Application Load Balancer with Amazon CloudFront. Its global network of edge locations optimizes content delivery, reducing latency for your users.

* 1. **AWS Web Application Firewall (WAF)**
  2. **AWS Global Accelerator**

1. **Summary -** Review the load balancer configurations and make changes if needed
2. Click on **create Load Balancer** button – Remember only the ALB will be created if no target

groups are Specified

**HOW ALB works :** **Client Hits ALB DNS name -> route to specified TG -> route to registered EC2**

**Instances.**

**How to associate another TG to an existing ALB:**

* Open **Load Balancers** under **EC2 Load balancing** dashboard.
* Select your Load Balancer and open it.
* Under Listeners and rules - You can see the one rule to attached TG which is listening on

HTTP : 80

* Click on **HTTP : 80** of associated TG – Open a template
* At Listener rules – You see the specified rules(we specified **Default** rule) of

associated TG.

Click on Add rule button

You see Listener details: HTTP:80 for IndrajaALB

Name and tags – Specify name to the rule.( Ex. **Item Rule**)

Click on Next button

A **Define rule conditions** template will open

You see **Listener details** : HTTP:80 for IndrajaALB

Click on Add Condition button

At Rule condition types : select **Path**

At path – specify any sequence of characters present in

ULR (Ex. \*item\*).

Click on Confirm button.

Click on Next button

At Actions – Select the Forward to target group

Routing actions.

At Target group – Select the TG to which you wan to forward the traffic when the URL contains above specified path pattern i.e., \*item\*

Weight – Ensure how equally should distribute the traffic

to TG

Click on Next button

A **set rule priority template** will open

You see Listener details: HTTP:80 for IndrajaALB

At Rule : Item Rule – **Item Rule** is the name you specified

above to a Rule.

Priority – specify **1**

At Listener rule – you see the **Item Rule** will be added.

Click on Next button

A Review and create template will open

Click on Create button

* Which take you to **EC2** -> **Load balancers** -> **IndrajaALB** -> **HTTP : 80**
  + Now click on **IndrajaALB** and refresh the Listeners and rule attribute of **IndrajaALB**

Please You see a new rule has added at Protocol : port listener HTTP : 80 of IndrajaALB associated TG.

* + Click on **2 rules** under Rules attribute.
  + At Listener rules - You see **Item Rule** and **Default** rule.

If you run the **ALB DNS name** in browser – which hits the **index.html** page of EC2 instance.

If you run the **ALB DNS name following sales/sales.html** in browser – which hits the

**sales.html** page of associated EC2 targets

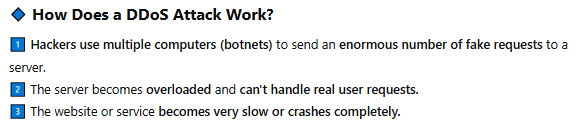
**NOTE:** Additional charges on **Load Balancer** and **Target Groups** and **EC2** apply by AWS.

If TG associated with ALB you must first delete the Load Balancer at Actions dropdown,

then delete the TG.

To protect our Load Balancer from **DDos attacks(**making websites or servers unreachable by flooding them with traffic. To protect against DDoS attacks, enable "AWS **Shield** " service under

"Integration" attribute of a "Load Balancer".



**Cloud watch:**

we can know who, for what ,and from where, the client is hitting our load balancer are logged at AWS "**Cloud Watch**" service.

All those logs will be stored in AWS service called "**S3**" storage

**How to specify the S3 URL to ALB to store details of all request made to ALB:**

* Open **Load balancers** console at **EC2** dashboard.
* Select and open your Load balancer.
* At Attribute – You see the **Monitoring** option have **Access** logs
  + Click on Edit button – A **Edit load balancer attributes** template will open
  + At Monitoring -> You see to enable **Access logs** and specify **S3 URL** to store request logs.

**Steps If we do not want to allow requests from specific region/IP address range to our Load Balancer**:

One- way by providing Subnet level security through Network ACL:

1. Open the Subnet where our Load Balancer's EC2 Instances are present (Click on instance ID, under its details you will find "Subnet ID". By clicking on its ID will take you to that Subnet )
2. At "Network ACL" where by default it will allow(both inbound and outbound) everyone.
3. Click on "Edit network ACL association" – a template will open
4. To edit the Inbound rules, click on "**Current network ACL ID**" which takes us to "Network ACLs"

template at **VPC’s** **Security** dashboard

1. Click on Network ACL ID which takes us inside of it.
2. Here we can edit the Inbound rules by clicking on "Edit inbound rules".

**Inbound rules control the incoming traffic that's allowed to reach the VPC.**

**NOTE :** Rules are evaluated starting with the lowest numbered rule.

1. Click on Add new rule button.
2. Here you can specify
   * Rule number
   * **Type –** The type of traffic; such as **SSH** or **HTTP**. You can also specify all traffic or a custom range. AWS default pre-fill it by **Custom TCP**
   * **Protocol – default selected as TCP(6)**
   * **Port** **range – select 80 for HTTP**
   * **Source -** మీ **VPC**లోకి **ప్రవేశించగల IP address ట్రాఫిక్** ఏది, మరియు అది **ఎక్కడి నుంచి రావచ్చు** అనే విషయాన్ని నిర్ణయిస్తుంది.
   * **Allow/Deny -** Whether or not to allow the specified IP address traffic to enter your VPC.

**What is Custom TCP option in AWS?**

* **TCP (Transmission Control Protocol)** is a widely used protocol for communication between computers over a network.
* AWS provides **predefined TCP types** such as:
  + **HTTP (Port 80)** → Web traffic
  + **HTTPS (Port 443)** → Secure web traffic
  + **SSH (Port 22)** → Secure shell access
  + **RDP (Port 3389)** → Remote Desktop Protocol for Windows
* However, if your **EC2 application** is running on a **custom port** (not a standard predefined port like 80, 443, 22, etc.), then **clients won't be able to reach your application unless you explicitly allow traffic on that port.**
* Predefined **TCP types** won't work for **custom ports**!
* **Custom TCP** lets you specify **non-standard TCP ports** that your application uses.
* Use it when your application runs on ports other than the default ones.

**Example:**

Imagine you have an application running on **port 5000** inside an **EC2 instance**. If you don’t allow traffic to **port 5000**, users won’t be able to connect to your application.

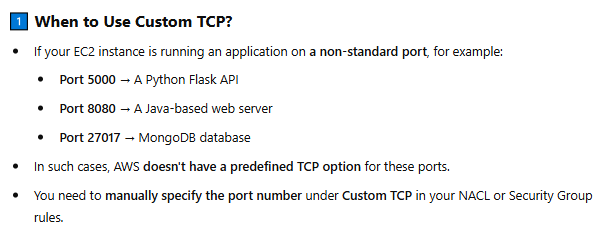
**To allow traffic at subnet level :**

**Go to NACL settings** in your AWS VPC -> **Add an inbound rule** ->

**Select "Custom TCP"** -> **Specify port 5000** ->

**Allow traffic from a specific IP range (CIDR block)**, e.g., 0.0.0.0/0 (allow

from anywhere).



**Auto scaling the Load Balancer:**

If your load balancer is handling **50times more requests** than usual, it may **struggle to distribute traffic efficiently**, causing slow performance or **website crashes.**

**Example:**

Imagine you run an **e-commerce website** that usually gets **1,000 visitors per hour.** Your website is behind an **Application Load Balancer (ALB)**, which distributes traffic across **5 EC2 instances**.

**Situation:** You announce a **big festive sale** (like Flipkart's Big Billion Days or Amazon's Great Indian

Festival).

**Normal traffic:** 1,000 users/hour

**Sale day traffic:** 50,000 users/hour

**AWS Handles auto scaling automatically:**

**Auto Scaling for Load Balancer Nodes:**

* If the **Load Balancer itself** is overwhelmed by **too many requests**, AWS **automatically adds more ALB nodes** in different Availability Zones. This ensures smooth **traffic distribution** even during heavy loads.

**Auto Scaling for Target Group Instances:**

* + To handle traffic AWS adds/removes EC2 instances based on CPU usage, requests, or latency. When traffic drops, Auto Scaling **removes extra instances** to save costs.
  + When traffic increases and CPU usage goes beyond 75%, **Auto Scaling Group (ASG)** automatically creates new EC2 instances using a **Launch Template. Then the new instances will have same AMI OS, Instance type(CPU, RAM), Security Groups, ESB Storage, User data, Networking (VPC, Subnet, IP settings) as you configure in Launch Template.**

**NOTE :**

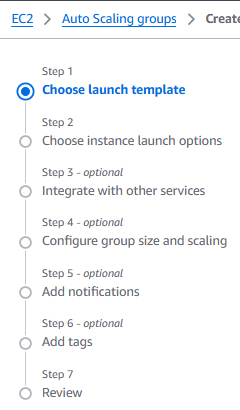
* When traffic increases AWS **does** **not** simply copy an Existing EC2 instance to launch more instances for scaling **because**
  + Your running EC2 might have **logs, cache, temporary files, or ongoing requests.**

If AWS directly clones the running instance, **it might include unnecessary or outdated data**, causing inconsistencies. **Launch Template ensures consistency** while scaling up.

* + The **existing EC2 might have manual changes** that AWS isn’t aware of (like updates, config changes, new software).

If AWS copies it, the new instances may have unexpected

differences.

**Steps to Set Up Launch template for EC2 Auto Scaling:**

* The **launch template** and **Auto Scaling group** that you create are tied to the Region that you specify.
* An **Auto Scaling group** is a collection of
* EC2 instances that are treated as a logical unit.
* **In AWS, EC2 Auto Scaling is the service** that automatically adjusts the number of EC2 virtual machine based on application workloads. It optimizes performance, availability and cost control. Users set the scaling policies using metrics like CPU utilization / Network traffic.
* **YouTube tutorial -** <https://www.youtube.com/watch?v=z8jraCN9Tf0.>

That explain only how to **create Launch Template** and how to launch EC2(irrespective of Auto scaling compatible) using launch template

1. Go to **EC2 Service** → Click on **Auto Scaling Groups**.
2. **Click on Create Auto Scaling Group button**

**Step -1 : Choose launch template console will open** - Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group.

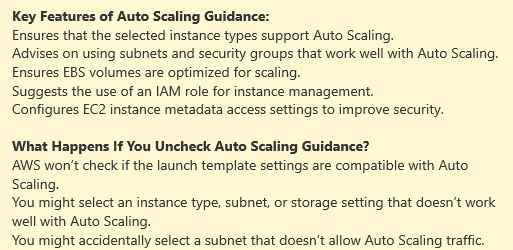
* 1. Name **-** Enter a name to identify the Auto scaling group.
  2. Launch template - Choose a launch template that contains the instance-level settings, such as AMI, instance type, key pair, and security groups.

**NOTE:** Instead of manually setting the EC2 configurations to launch an instance we can define a Launch Template and use it whenever we want to launch an EC2 instance.

* + 1. Click on create launch template
       1. Launch template name – Ex. Mytemplate
       2. Template version description – Ex. My production webserver for MyAPP.
       3. Enabling **Auto Scaling guidance** - helps you configure your EC2 instances efficiently for Auto Scaling Groups (ASG).

Ensures that the **launch template** settings are **compatible** with Auto Scaling Groups.

Suggests configurations that help in **dynamic scaling**.



* + - 1. Source template – You can use existing Launch Template to create new launch template.
    1. At Launch template contents.
       1. **AMI** – Select windows 2016 base AMI from

the **Quick Start** lists.

* + - 1. **Instance type** – Choose free tier 1CPU, 1RAM
      2. **Key Pair** - use a key pair to securely connect to your instance.
      3. At **Network settings,** expand **Advanced network configuration** and do the following:
         1. Choose **Add network interface** to configure the primary network interface.
         2. **Auto-assign public IP -** If you don't need to connect to your instance, choose **Disable**.
         3. For **Security group ID -**  If you don't specify your instance is automatically associated with the default security group for the VPC.
         4. For **Delete on termination**, choose **Yes** to delete the network interface when the instance is deleted.
      4. **Storage (volumes)**
      5. **Advanced details**
  1. **Click on create Launch template button –** To launch newer EC2 instances for auto scaling, you can use existing/previously created **Mytemplate** - Launch template which contains the instance-level settings.
  2. **A Next Steps console will open**
  3. On the confirmation page, choose **Create Auto Scaling group**.

**Step -2 :** **To create an Auto Scaling group,**

1. On the **Choose launch template page** 
   * + 1. for **Auto Scaling group name**, enter **my-first-asg**.
       2. Here **Mytemplate** Launch template will be pre-defined by AWS
       3. Click on **Next** button
2. **Choose instance launch options** page will open - Choose the VPC network environment that your instances are launched into
   * + 1. In the **Network** section
          1. VPC – Select your VPC
          2. For **Availability Zones and subnets -** choose a subnet from each Availability Zone that you want to include. Use subnets in multiple Availability Zones for high availability.
          3. In the **Instance type requirements** section - use the default setting to simplify this step.
          4. you will launch only one On-Demand Instance using the instance type specified in your launch template.
          5. choose **Skip to review**.
3. On the **Review** page, review the information for the group, and then choose

**Create Auto Scaling group**.

* + - * 1. Opens **Auto scaling groups** page of EC2 console.
        2. Select the check box next to the Auto Scaling group that you just created.
        3. Open a **Details** tab - shows information about the Auto Scaling group.
        4. At **Activity** tab, Under Activity history – Status column shows current status of instance

Not yet in service – shows instance is launching

The status changes to Successful after the instance is launched.

* + - * 1. On the **Instance management** tab, under **Instances**, you can view the status of the instance

The **Lifecycle** column shows the state of your instance. Initially, your instance is in the Pending state. After an instance is ready to receive traffic, its state is InService.

The **Health status** column shows the result of the Amazon EC2 Auto Scaling health checks on your instance.

* + - * 1. Do these steps to learn more about how Amazon EC2 Auto Scaling works.

1. On the **Instance management** tab, under **Instances**, select the **ID** of the instance. This takes you to the **Instances** page of the Amazon EC2 console, where you can terminate the instance.
2. Choose **Instance State**, **Terminate**.
3. At EC2 dashboard, under **Auto Scaling**, choose **Auto Scaling Groups**. Select your Auto Scaling group and choose the **Activity** tab.
4. In the **activity** **history**, when the scaling activity starts, you see an entry for the **termination** of the first instance and an entry for the **launch** of a new instance.
5. On the **Instance management** tab, the **Instances** section shows the new instance only.
6. At EC2, under **Instances**, choose **Instances**. This page shows both the terminated instance and the new running instance.

**NOTE:** The minimum size for the Auto Scaling group that you created in this tutorial is **one** instance. Therefore, if you terminate that running instance, Amazon EC2 Auto Scaling must launch a new instance to replace it.

**An Auto Scaling group always tries to maintain its desired capacity** . In cases where an instance terminates unexpectedly, the group automatically launches a new instance to maintain its desired capacity.

**Step -3 :** **Configure group size and scaling** - optional

1. If the load on your application changes, your Auto Scaling group can scale out (add instances) and scale in (run fewer instances) automatically by adjusting the desired capacity of the group between the minimum and maximum capacity limits.
   * 1. At **EC2** dashboard, under **Auto Scaling**, choose **Auto Scaling Groups**.
     2. On the **Auto Scaling groups** page, select the check box next to your Auto Scaling group. A split pane opens up in the bottom of the page.
     3. In the **Details** tab, you see group capacity overview, Click on **Edit** button

where you can edit **Group size** and **Scaling limits**.

**NOTE:** You can view the number of instances and the status of the currently running instances from the **Instance management** tab, under **Instances**.

**Step – 4 : Automatic Scaling**

1. AWS Auto Scaling **automatically adjusts the number of instances** in your Auto Scaling Group to maintain a specific metric target, like **CPU utilization, request count, or memory usage**.
   * 1. At **EC2** dashboard, under **Auto Scaling**, choose **Auto Scaling Groups**.
     2. select the check box next to your Auto Scaling group.
     3. Click on **Automatic Scaling** attribute → Click **create dynamic scaling policy**
        1. **Policy type** – Ex. Target tracking scaling
        2. **Scaling policy name** – Target tracking policy
        3. Metric type – defines **what aspect of your instances or load balancer is being monitored** to decide when to scale **up** or **down**.
           1. Choose Average CPU utilization to scale up the instances when specified CPU percentage has reached.
           2. Choose ALB request count per target to scale up the instances when specified **TG** has received specified request count.
        4. Target value – 85

**Step -5 :** **Add notifications – optional**

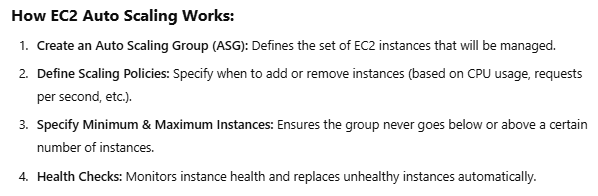
1. Configure an **Amazon SNS notification** to notify you whenever your Auto Scaling group launches or terminates instances. For more information, see [Amazon SNS notification options](https://docs.aws.amazon.com/autoscaling/ec2/userguide/ec2-auto-scaling-sns-notifications.html).
   * 1. At **EC2** dashboard, under **Auto Scaling**, choose **Auto Scaling Groups**.
     2. select the check box next to your Auto Scaling group.
     3. On the **Activity** tab, choose **Activity notifications**, **Create notification**.

**To delete your Auto Scaling group:**

1. At **EC2** dashboard, under **Auto Scaling**, choose **Auto Scaling Groups**.
2. select the check box next to your Auto Scaling group.
3. Click on **Actions** -> Choose **Delete**.

**To delete your launch template:**

1. Open the [Launch templates page](https://console.aws.amazon.com/ec2/v2/#LaunchTemplates) of the Amazon EC2 console.
2. Select your launch template (my-template-for-auto-scaling).
3. Choose **Actions**, **Delete template**.



**AWS SNS – Simple Notification Service:**

Amazon SNS is a fully managed service that **provides message delivery** from publishers (producers) to subscribers (consumers).

Publishers communicate asynchronously with subscribers by sending messages to a topic, which is a logical access point and communication channel.

**NOTE :** One **Topic** can have **multiple subscribers**.

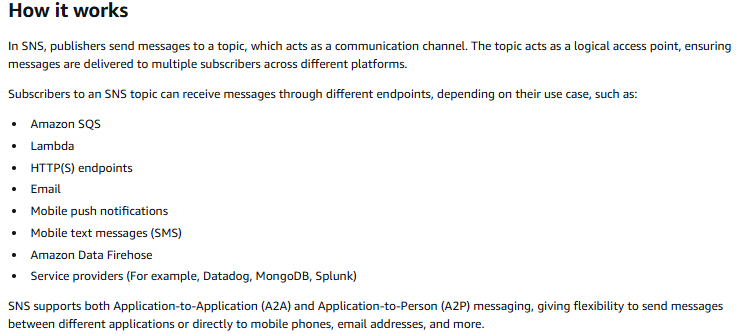
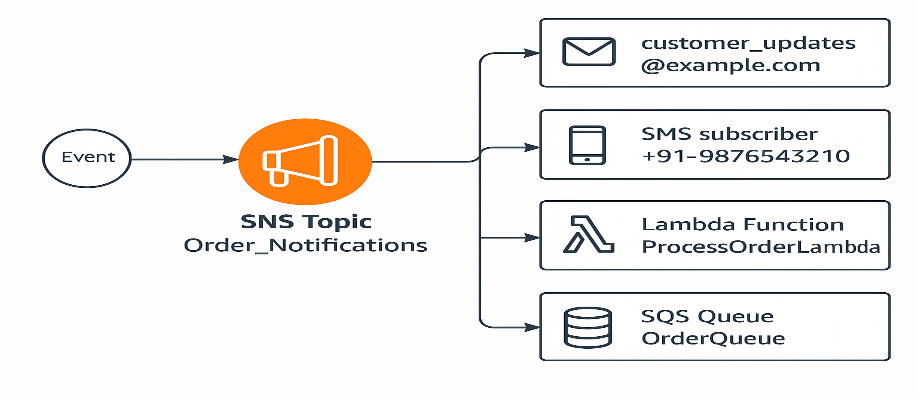


Diagram showing an SNS topic with multiple subscribers, including an email, SMS, Lambda function, SQS queue, and an HTTP endpoint.****

**Steps to create SNS:** [**https://docs.aws.amazon.com/sns/latest/dg/sns-create-topic.html**](https://docs.aws.amazon.com/sns/latest/dg/sns-create-topic.html)

An Amazon SNS topic is a **logical access point** that acts as a **communication channel.**

**A topic lets you group multiple endpoints** (such as AWS Lambda, Amazon SQS, HTTP/S, or an email address).

**Step -1 : Creating a Topic**

1. At AWS console search for **SNS**
2. Select **Simple Notification service** option
3. Click on **start with an overview.**
4. On the **Topics** page, choose **Create topic**.
5. On the **Create topic** page, in the **Details** section, do the following:
   1. For **Type**, choose a topic type (**Standard** or **FIFO** – first-in, first-out).

For practice choose **Standard.**

**NOTE :** After creating a topic, you **can't change** the topic type or name.

* 1. **Name** - Enter a **Name** for the topic. (Ex: *MyTopic*)

**NOTE** : For a [FIFO topic](https://docs.aws.amazon.com/sns/latest/dg/sns-fifo-topics.html), add **.fifo** to the end of the name.

* 1. (Optional) Enter a **Display name** for the topic.

For SMS subscribers to topic can only first 10 chars are displayed in SMS message.

1. **Access policy** – defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic. To configure additional access permissions, expand the **Access policy** section.
   1. For practice continue with AWS default select option **Basic.**
   2. Publishers - Specify who can publish messages to the topic.

Best practice, select **Everyone** – Anybody can push

* 1. Subscribers – Specify who can subscribe to the topic.

Best practice, select **Everyone** – Any AWS account can subscribe to the topic.

1. Click on **Create topic** – The topic is created and the ***MyTopic*** page is displayed.

The topic's **Name**, **ARN**, (optional) **Display name**, and **Topic owner**'s AWS account ID are displayed in the **Details** section.

1. Copy the topic **ARN** to the clipboard.

For example: arn:aws:sns:us-east-2:123456789012:MyTopic

**Step-2 : Scroll down the page, where you can create subscriptions.**

* **This step is essential for linking the topic to consumers**
* To receive messages that are published to [a topic](https://docs.aws.amazon.com/sns/latest/dg/sns-create-topic.html), you must subscribe an [endpoint](https://docs.aws.amazon.com/sns/latest/dg/sns-create-subscribe-endpoint-to-topic.html#sns-endpoints) to the topic.
* When you subscribe an endpoint to a topic, the endpoint begins to receive messages published to the associated topic.

**NOTE : HTTP(S) endpoints**, **email addresses**, and **AWS resources in other AWS accounts** require confirmation of the subscription before they can receive messages.

1. Click on **Create Subscription** button (or At AWS SNS console, at left navigation

pane choose **Subscriptions**.)

1. On the **Create subscription** page, in the **Details** section, do the following:
   * 1. **Topic ARN** – Specify the **ARN** name of the created topic.

AWS prefills with recently created *Mytopic*’s**ARN** name.

* + 1. **Protocol** – Specify the type of endpoint to subscribe.

Select **Email** option.

**NOTE :** To subscribe to an [SNS FIFO topic](https://docs.aws.amazon.com/sns/latest/dg/sns-fifo-topics.html),

choose **AWS lambda**, **Platform application endpoint**, **SMS**.

* + 1. **Endpoint –** Specify the email ID that can receive notification from SNS**.**

**NOTE :** After you subscribe your endpoint, Amazon SNS sends a subscription confirmation message to the endpoint. The code at the endpoint must retrieve the SubscribeURL from the subscription confirmation message.

**Remember,** if you select **SMS** as **protocol** and provide **your phone number** as an **Endpoint** then AWS would not send **SNS conformation message** to that **endpoint** because AWS ensures telecom regulations and **automatically confirms** the phone number when added as a subscriber.

1. Click on **Create Subscription** button - The console creates the subscription and opens the subscription's **Details** page.

* Subscription **ARN, Endpoint, Topic, Subscription principal, Status, Protocol** are displayed in the **Details** section.
* The **Status** here shows **pending confirmation** until the subscription is confirmed **at SNS confirmation email which is sent to provided endpoint.**
* Once you confirm the subscription the **Status** turns into **Confirmed.**

**Step-3 : Publishing a message. - https://docs.aws.amazon.com/sns/latest/dg/sns-publishing.html**

* After you [create an Amazon SNS topic](https://docs.aws.amazon.com/sns/latest/dg/sns-create-topic.html) and [subscribe](https://docs.aws.amazon.com/sns/latest/dg/sns-create-subscribe-endpoint-to-topic.html) an endpoint to it, you can ***publish***messages to the topic.
* When a message is published, Amazon SNS attempts to deliver the message to the subscribed [endpoints](https://docs.aws.amazon.com/sns/latest/dg/sns-create-subscribe-endpoint-to-topic.html#sns-endpoints).

1. Sign in to the [Amazon SNS console](https://console.aws.amazon.com/sns/home).
2. In the left navigation pane, choose **Topics**.
3. On the **Topics** page, select a topic, and then choose **Publish message**.

The console opens the **Publish message to topic** page.

1. In the **Message details** section, do the following:
   * 1. **Subject** - Enter a message **Subject**.
     2. **Time to Live** - This is the amount of time that a **push notification service** ( like Apple Push Notification Service or Firebase Cloud Messaging) has to deliver the message to the endpoint.
   1. In the **Message body** section, do one of the following:
      1. **Message** **structure** - Choose **Identical payload for all delivery protocols**, and then enter a message.
      2. At **Message body to send to the endpoint** – provide the message which to want to publish
   2. **Message attributes** - add any attributes that you want Amazon SNS to match with the subscription attribute **FilterPolicy** to decide whether the subscribed endpoint is interested in the published message.
      1. For **Type**, choose an attribute type, such as **String, String.Array** or **Number**.
      2. Enter an attribute **Name**, such as customer\_interests.
      3. Enter an attribute **Value**, such as ["soccer", "rugby", "hockey"].
   3. Click on **Publish message** button

## To delete an Amazon SNS topic and subscription:

* Sign in to the [Amazon SNS console](https://console.aws.amazon.com/sns/home).

1. In the left navigation pane, choose **Topics / Subscriptions**.
2. On the **Topics / Subscriptions** page, select a topic / subscription with a status of **Confirmed**, and then choose **Delete**.
3. In the **Delete topic / Delete Subscription** dialog box, enter ***delete me***, and then choose **Delete**.
4. The console deletes the topic / Subscription.

**SNS Use case:**

**Amazon SNS integrates with many AWS services for messaging, notifications, and event-driven architectures.**

**Example:** Amazon EC2 can use or integrate with SNS. SNS can send notifications about EC2 instance health, auto-scaling events, or alarms.

* Open EC2 -> Auto Scaling Groups-> Choose Create Auto Scaling group button

1. At Choose launch template or configuration page
   * 1. Name – specify auto scaling group name
     2. Launch Template - Either select the existing template or

Click on Create a launch template

* + 1. At **Create** **launch** **template** page
       - At **Launch template name and description**
         1. **Launch template name** – Specify the name to identify your template
         2. **Template Version description – Ex. A prod webserver for MyApp**
         3. Enabling **Auto Scaling guidance -** helps you configure your EC2 instances efficiently for Auto Scaling Groups (ASG).

**It ensures** that the **launch template** settings are **compatible** with Auto Scaling Groups.

If you **uncheck** Auto Scaling guidance, AWS won’t check if the launch template settings are compatible with Auto Scaling.

* + - * 1. Source template – You can use existing Launch Template to create new launch template.
      * At **Launch template contents**
        1. **Amazon Machine image** – At **Quick Start**, select free tier windows 2016 OS.
        2. **Instance type** – Select t2.medium
        3. **Key pair – you can use key pair to securely connect to your instance**
        4. At **Network settings**

**Subnet** - When you specify a subnet, a network interface is automatically added to your template.

Generally This determines where the EC2 instance will be launched within your AWS VPC.

**NOTE :** **Subnet** Option Not Applicable for EC2 auto scaling.

🔹If you **don’t** plan to use Auto Scaling, then while creating the **Launch Template**, you **must** select a **subnet** in the **Network Settings**.

When launching an EC2 instance manually from a launch template, AWS needs to know **which subnet** to place the instance in. Without a subnet, the instance **won’t have a network connection** and **can’t be launched**.  
🔹 If you use Auto Scaling, the **ASG selects the subnet, not the Launch Template**.

When using **EC2 Auto Scaling**, AWS **does not** launch instances in a specific subnet from the Launch Template itself. Instead, the subnet selection happens **at the Auto Scaling Group level**.

**Firewall (security groups) –** check box the Select existing security group option

**Common Security groups** – Select the security group

Expand **Advance network configuration**

**Auto-assign public IP -** If you don't need to connect to your instance, choose **Disable**.

For **Delete on termination**, choose **Yes** to delete the network interface when the instance is deleted.

For **Security group ID -**  If you don't specify your instance is automatically associated with the default security group for the VPC.

Choose **Add network interface** to configure the primary network interface.

**Storage (volumes)**

**Advanced details**

* + - * 1. **Click on create Launch template button**
        2. **A Next Steps console will open**
        3. On the confirmation page, choose **Create Auto Scaling group**.
        4. A **Choose launch template** page will open again.

**Name –** Specify the auto scaling group name. **Ex. *InduASG***

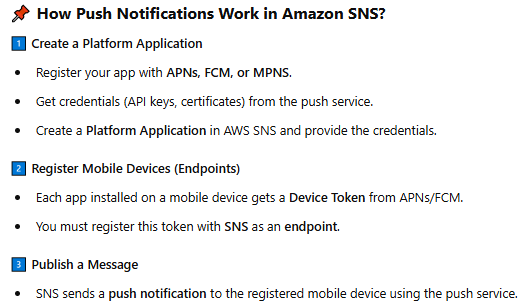
Click on **Next**

1. **Choose instance launch option** page will open
   * 1. At **Network**
        + **VPC –** Select VPC for your Auto scaling group
        + **Availability zones & subnets –** Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.
        + Click on **Next**
2. **Integrate with other services -** page will open
   * 1. Continue with AWS prefilled options i.e., **No load balancer**, **No VPC Lattice service**, **Health check grace period 300 sec**
     2. Click on **Next**
3. **Configure group size and scaling policies** page will open
   * 1. **At Group Size, Desired capacity –** Specify your group size. **For Ex.** if you specify 3 then total 3 EC2 instances will be created for autoscaling.
     2. **At Scaling, Min desired capacity, Max desired capacity** options allow us to set limits on how much your desired capacity can be increased or decreased.
     3. **At Automatic scaling** choose **Target tracking scaling policy**
        + **Metric type –**
          1. Choose **Average CPU utilization.** Specify **Target value Ex. 85** which means When CPU utilization reach 85% then increase scaling.
          2. Choose **Application load balancer request count per target.** Specify Target value
     4. Click on **next**
4. **Add notifications** page will open
   * 1. Click on **Add Notification**  button
     2. At **Notification 1**
        + **SNS Topic –** Specify the SNS topic. Ex. *MyTopic*
        + Click on **Next**
5. **Add tags** page will open
   * 1. Click on **Next**
6. A **Review** page will open. Click on **Create Auto Scaling Group** button.

Which successfully create ***InduASG***Auto scaling Group and **1 Notification**

Amazon **SNS** can send **push notifications** to mobile devices (iOS, Android, Windows, etc.) using **push notification services** like:

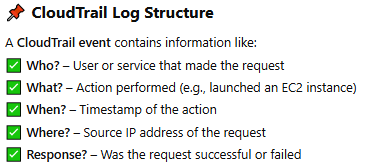
* **Apple Push Notification Service (APNs)** – for iOS devices
* **Firebase Cloud Messaging (FCM)** – for Android devices
* **Microsoft Push Notification Service (MPNS)** – for Windows devices
* **How Push Notifications Work in Amazon SNS:**



* 1. Open **Amazon SNS**
  2. At navigation page, Select Push Notifications.

**Cloud trail : Must enable at production level.**

* **AWS CloudTrail** is a service that helps you track and monitor **user activity and API usage** in your AWS account.
* It records actions taken by users, services, and roles, providing visibility into who did what, when, and from where.
* It Stores Event Logs in S3 & CloudWatch.
* Cloud Trail logs are **Read-only**, hence no one can edit them.
* **NOTE :** Creating trails are chargeable in AWS

****

1. Sign to AWS **CloudTrail** service
2. Click on **navigation pane**
3. Select **Dashboard**
4. Scroll down and check on **Event history**
5. Click on **View full Event history**
6. Console opens an **Event history** page - Event history shows you the last 90 days of management events.
7. Click on any **Event name** which open a **Details** page of selected Event.
8. At Dashboard, click on **Create trail** button.

Console opens **Choose trail attribute** page.

* 1. At **General details**
     1. Trail name – Enter display name for your trail
     2. **Storage location –** Stores your trail’s event logs

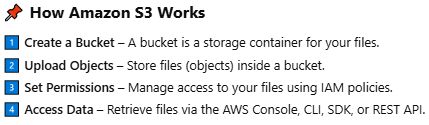
AWS auto selects **Create new S3 bucket** option

* + 1. At **Additional settings** 
       1. **SNS Notification delivery** - CloudTrail stores multiple events in a log file. **When you enable this option**, Amazon SNS notifications are sent for every log file delivery to your S3 bucket, not for every event.

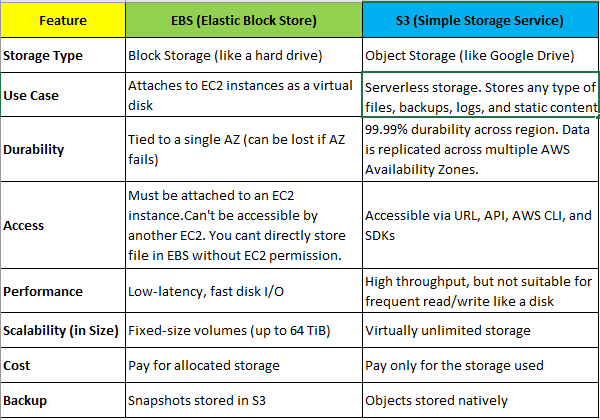
**Amazon S3 – Simple Storage Service :**

Amazon S3 is **secure object storage service** that allows you to store and retrieve any amount of data from anywhere on the web.

* Stores data as **objects** (files) inside **buckets** (containers).
* Each object has **metadata** (name, size, creation date, etc.).
* Objects can be from **0 bytes to 5 TB** in size. No limit on the amount of data you can store.
* Your data is **replicated across multiple AWS Availability Zones** to prevent data loss.
* **Designed for 99.99% availability** – Always accessible.
* Supports **Server-Side Encryption (SSE)** and **Client-Side Encryption**.
* **Versioning** keeps track of all changes to files, preventing accidental deletions
* **Lifecycle Rules** automatically move objects to cheaper storage classes.
* **Control access using IAM((Identity and Access Management)) Policies, Bucket Policies, and ACLs (Access Control Lists).**



**Key Differences Between EBS and S3:**

****

**S3 terminologies :**

**Example:** Assume the file path – D:/Movies/Telugu/2025/ABC.mp4

**In S3 the following are defined as,**

**D:/** - Parent Folder – Also called **S3 bucket** which **has static name** in AWS

world.

**Movies/Telugu/2025** – Sub-folder.

**ABC**.mp4 – S3 object which is a file.

**File path** – S3 key / key prefix.

**S3 pricing :**

​Amazon S3 offers flexible, pay-as-you-go pricing based on several factors, including storage class, data retrieval, and data transfer.

* **Storage Pricing based on storage class (per GB per Month).**
* **Duration**
* **Region**
* **Request**(PUT, COPY, POST, LIST) **and Retrieval**(GET and Other Requests) **Pricing.**
* **Data Transfer Pricing**
  + - **Data Transfer IN**: Free across all AWS regions.​
    - **Data Transfer OUT** to the Internet: chargeable

**Creating S3 bucket :**

**Remember,** S3 is region based

**S3 pricing :** You are not charged for creating a bucket. You are charged only for storing objects in the bucket and for transferring objects in and out of the bucket.

1. Open AWS console
2. Search **S3** service – Scalable storage in the cloud
3. Every object in S3 is stored in a bucket. To upload files and folders to S3, you’ll need to create a bucket where the objects will be stored.
4. In the navigation bar on the top of the page choose the Region in which you want to create a bucket.
5. In the left navigation pane, choose **General purpose buckets**.
6. Choose **Create bucket**. The **Create bucket** page opens.

Buckets are containers for data stored in S3.

* + For **Bucket name**, enter a name for your bucket where no one in AWS will have same name. Remember, don't include sensitive information in the bucket name because it is visible in the URLs that point to the objects in the bucket.

**NOTE :** Be unique within a partition. AWS currently has three partitions :

aws (commercial Regions), aws-cn (China Regions), and

aws-us-gov (AWS GovCloud (US) Regions).

AWS recommend to avoid using periods (.) in bucket names, except for buckets that are used only for static website hosting.

* + At **Object ownership** – Enable **ACLs enabled**
  + At **Block Public Access settings for this bucket –** uncheck the **Block all public access**
  + Click on **Create bucket** button
  + At **General purpose buckets** page – open your bucket Ex. *indu-aws-us-bucket*
  + At ***indu-aws-us-bucket*** page -
  + Click on **create folder** button – creates folder in S3 bucket
    - Folder – give a name to folder Ex. *movies*
    - Click on Create Folder button
  + Select the ***movies*** folder to create a folder inside it. Ex. *telugu*
    - Select ***telugu*** folder and click on Upload button to upload a file in it.
    - Click on Add files button, add a file
    - Scroll down, click on **Permission** option - **Access control list (ACL)** pagewill open whichGrant basic read/write permissions to other AWS accounts.
      * At **Choose from predefined ACLs** select **Grant public-read access** option, where Anyone in the world will be able to access the specified objects. The object owner will have read and write access.
    - Click on Upload button – which adds the uploaded files to s3
    - Close the review page.
    - Click the file **Ex. image.png** which you uploaded to ***telugu***folder.
    - Steps to create **presigned S3 URL** to **grant temporary public access** to your files securely — without making the object always open to public.
      * At **Object Actions** button select **shared with a presigned URL** – a windows will open, where you can mention the number of min/hours you would like to provide public access to your **S3 URL**
      * Click on **Create presigned URL** button.
        1. Copy the presigned URL and can share with public which can only accessible to the S3 object for specified time.
    - A **Properties** attribute page of selected file **Ex. *image.png*** will open which shows **Object overview** of selected object/file.

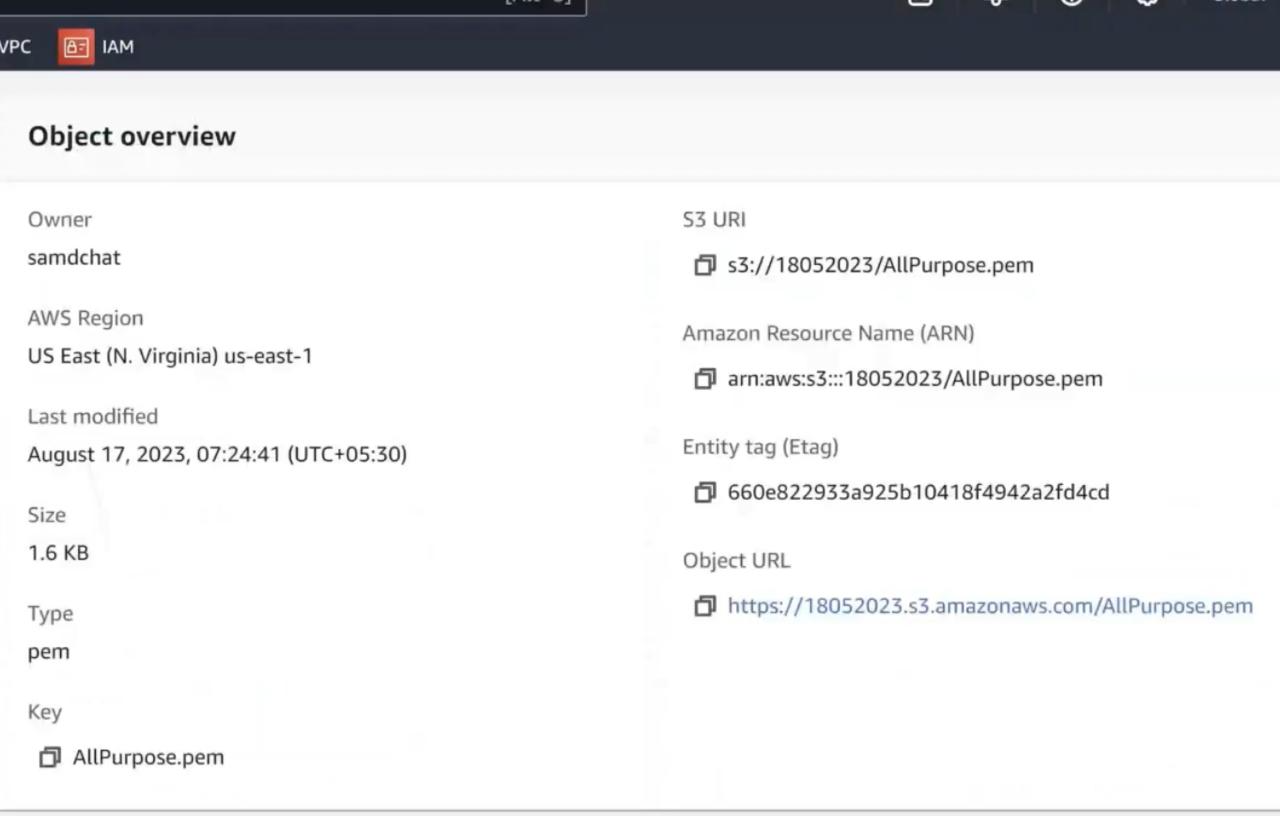


Fig: similar to **S3**->**Buckets**->**Indu-aws-us-bucket**->**movies** folder->**telugu** folder ->**image.png**

* + - * + **Amazon Resource Name (ARN) –** All AWS accounts లోని resources (EC2, S3, Lambda, IAM policies, etc.) ను unique గా identify చేయడానికి ఉపయోగించే structure.

**ARN structure** - arn:partition:service:region:account-id:resource

**Remember, in context of S3** as default S3 name is unique in AWS world hence there is no need to define region, account-id in **ARN**

**Ex.** arn:aws:s3:::indu-aws-us-bucket/movies/telugu/Cloudtrail.png

* + - * + **Entity tag(Etag)** - **ETag** అనేది object content ఆధారంగా ఏర్పడే **MD5 Checksum** (for single-part uploads).Internet through file transfer jariginappudu Upload చేసిన file ఏమాత్రం corrupt అవుతుందా అని verify చేయడానికి and Download చేసిన file correct గానే ఉందా అని check చేయడానికి file yokka checksum value chepthundi.

**Experiment :**

<https://emn178.github.io/online-tools/md5_checksum.html> - upload local system file which generates MD5 checksum code and upload same file in S3. Change the content in local file and upload in above github site and compare MD5 checksum value of both uploaded file and s3 file.

* + - * + **Object URL** – it’s a public access URL to your S3 file object.

**Ex**. <https://indu-aws-us-bucket.s3.us-west-2.amazonaws.com/movies/telugu/Cloudtrail.png>

* + - * If you scroll down the page you will see the **Metadata** which is an information associated with object provided as **key-value** pair.

**Example:**

* + - * + **type** : System defined
        + **key** : Content-Type
        + **value**: application/pdf
      * Click on **Permission** attribute of an object

At **Access control list(ACL)** page, Click on **Edit** button.

Grant **Objects** **Read** access to **Everyone(public access). Checkbox the acknowledgement**

Click on Save changes button

Hence everyone in the world can use endpoint(URL) of an S3 object.

* + - * At **Properties** attribute now copy the **Object URL** and run in browser. You can have the S3 object open in your browser.

**How to remove public access permissions to S3 object**

* + - * Go to **Permissions** attribute and **Edit** the **ACL** **Read** permissions to deny the public access to S3 **object URL**

**S3 Storage Classes:**

Amazon S3 లో **Storage Classes** అనేవి objects ను వేరే వేరే usage patterns, availability, durability, cost ఆధారంగా categorize చేయడానికి ఉపయోగిస్తారు. మీ data ఎంత frequently access అవుతుందో, retention period ఎంత వుంటుందో బట్టి మీరు correct storage class ని ఎంపిక చేసుకోవచ్చు.

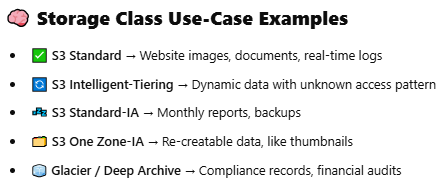
**Setting the storage class while uploading a new object/file to S3 or to a folder in S3:** <https://docs.aws.amazon.com/AmazonS3/latest/userguide/sc-howtoset.html>

**NOTE:**  If you don't specify storage class for an object when you upload it, AWS S3 uses the default **S3** **Standard storage class** for objects in **general purpose buckets**.

* You can also change the storage class of an existing object.

1. Sign in to the AWS Management Console and open the Amazon S3 console at: [https://console.aws.amazon.com/s3/](https://console.aws.amazon.com/s3).
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of the bucket that you want to upload your folders or files to.
4. Choose **Upload**.
5. In the **Upload** window, scroll down the page and choose **Properties**.
6. Under **Storage class**, choose a storage classes for the files you're uploading.

**NOTE :** Reduced redundancy storage class is not AWS recommended as S3 Standard is more cost effective



1. In the Upload window, do one of the following:
   1. Drag files and folders to the Upload window.
   2. Choose **Add file** or **Add folder**, choose the files or folders to upload, and choose **Open**.
2. At the bottom of the page, Choose **Upload**.

**AWS S3 feature ‘ Multi-part Upload ’:**

When you're uploading **large files** (say, > 100 MB or several GBs), instead of uploading it as **one big file**, S3 lets you: ✅ **Split the file into parts** ✅ **Upload each part in parallel** ✅ **Combine them into one file on S3.**

* **S3 Size limitation**
* Number of allowed S3 buckets per AWS account is **100.**
* Max Total data store in a single bucket is **Unlimited.**
* **On Manual uploads**, Max object/file size is **5 GB.**

**Ex**. Uploading in one single PUT request via drag and drop or simple API call

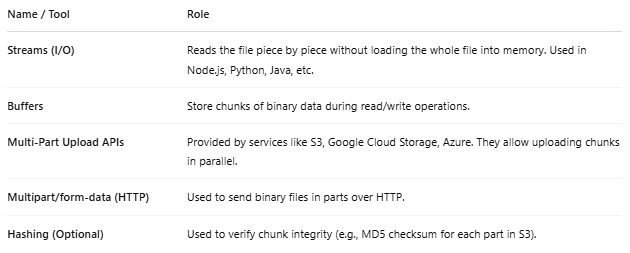
* **On Automated uploads**, Max object/file size is **5 TB**

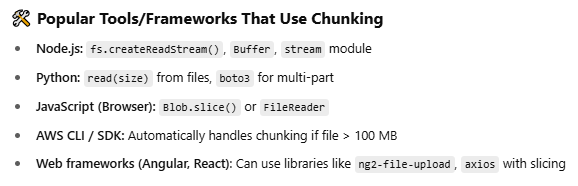
**Ex. Multi-part upload -** Best for big files — breaks file into parts and uploads in parallel.

**Each part** in a multi-part upload must be at **least 5 MB**.

However, Maximum parts allowed are **10,000 parts**.

**NOTE** : Companies uses AWS **multi-part upload** feature instead of relying on 3rd party File

splitting tools to cut the file**.** 

****

**S3 feature ‘Strong Consistency i.e., Strong Read-After-Write Consistency and Strong List-After-Write’ :**

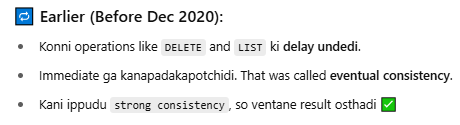
[**https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html#ConsistencyModel**](https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html#ConsistencyModel)**.**

Consistency in storage means - “If I upload a file or change existing file, when will I be able to **see that change** when I read or list files?”

Amazon S3 provides **strong read-after-write consistency** for **PUT**(upload) and **DELETE** requests of objects in your Amazon S3 bucket in all AWS Regions.

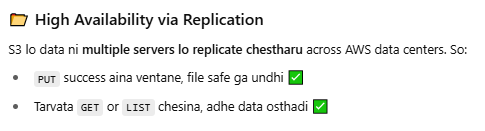
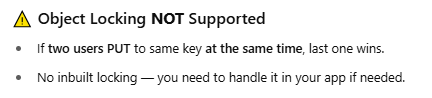
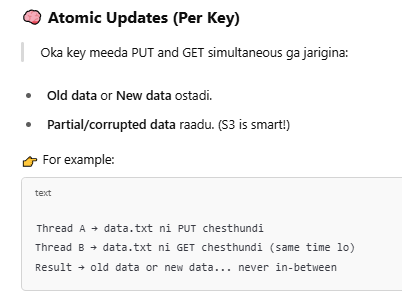
✅ Nuvvu oka file ni **PUT** (upload) or **DELETE** chesinaka, Ventane nuvvu **GET** or **LIST** chesina, **Correct result** vastundi — delay lekunda.

✅ Ante nuvvu file upload(**PUT**) chesina ventane, 📄 **GET** chesi chudachu, 📜 **LIST** chesi adhi file lo unda ani check chesi chudachu, 🗑️ **DELETE** chesaka adhi ventane remove ayyindho ledo chudachu.



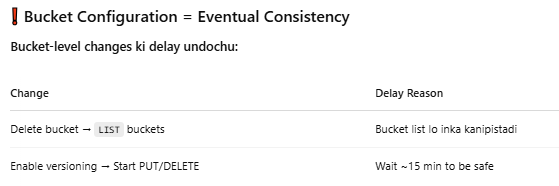
In **Amazon S3**, every file (object) is **uniquely identified** inside a bucket by a **key**.

3 things which helps to identify object/file in S3

* **Bucket Name** → **eg:** indraja-files
* **Key** (file path inside that bucket/folder)→ **eg:** photos/2024/jan/birthday.png
* **Version ID** (only if versioning is enabled)

**Bucket configurations have an eventual consistency model.** Specifically, this means that: If you delete a bucket and immediately list all buckets, the deleted bucket might still appear in the list.

If you enable versioning on a bucket for the first time, it might take a short amount of time for the change to be fully propagated. We recommend that you wait for 15 minutes after enabling versioning before issuing write operations (PUT or DELETE requests) on objects in the bucket.



**Expected behaviour from S3 when multiple clients are writing to the same items:**

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html#ConsistencyModel> **- Follow this link**

**NOTE:** S3 offers strong read-after write consistency for all GET, PUT, and LIST request operations. This means, after a successful write(PUT) request, any subsequent read(GET) request will return the latest version of the object. **Behaviour-1:** Whenboth W1 (write 1) and W2 (write 2) finish before the start of R1 (read 1) and R2 (read 2). Because S3 is strongly consistent, any subsequent read operations(R1,R2, …) will return the data from the latest write i.e., W2 content.

Which means W1 and W2 rendu Simultaneous ga perform jarigithe, latest completed write W2 ni S3 consider chesthundi.

**Behaviour-2:** When W2 starts before R1 or W2 might complete **DURING** R1 execution. Therefore, R1 might return W1 or W2 content. (S3 strong consistency ante Only AFTER SUCCESSFUL WRITE COMPLETE data matrame read chesthundi )However, because W1 and W2 finish before the start of R2, R2 returns W2 content.

**Behaviour-2: Concurrent writes** ante W1 finish Avvakapoina W2 start avvadam. **However**, S3 internally follows **LAST-WRITE-WINS** rule but due to Network latency(distance) request order will become unpredictable so Final READ lo W1 or W2 renditilo yedaina data read ayyae chance undi.

**S3 Bucket’s “properties” attribute:**

1. Open Amazon S3
2. Select a **bucket** under **General Purpose buckets.**
3. Select **properties** attribute of selected bucket. Here you see
   1. Bucket’s **AWS Region**
   2. Bucket’s **Amazon Resource Name**
   3. Bucket’s **Creation date**
4. At Bucket Versioning – Versioning means keeping multiple variants of an object in the same bucket.
   1. **Versioning explanation** – Suppose user upload 1GB **file.txt** to S3 as **LATEST** later he modified **file.txt** to 2GB and upload it to s3 as another new **LATEST** file by keeping the old **1GB** remain in S3 i.e., previous upload 1GB **file.txt** will remain exists in S3 and won’t get overwritten by modified 2GB **file.txt** then the last upload 2GB **file.txt** will become **LATEST** and last previous upload 1GB file.txt will become **Version Vn**.
   2. **With versioning you can easily recover deleted object/file.**
   3. **NOTE:** Once you enable Bucket versioning you can not Disable it but suspend it.
   4. **Steps to recover delete file from a bucket if versioning is enabled on the bucket:**
      * Open bucket.
      * Find the folder where the file was deleted.
      * Enable **Show Versions,** which shows a list of
        + **Latest** with type = Delete marker
        + **Previous versions of file**
      * Select the **Delete marker** file type and click on **Delete** button at top
      * Close and Disable **Show version**
      * Now you could see deleted file recovered to Folder.
5. Scroll down where you see Server access logging – Which logs the request for access to your bucket.
   1. Click on **Edit** button
   2. Enable **server access logging**
   3. At Destination use **Browse S3** to select your bucket.

**NOTE:** **Should not specify same bucket** to save logs because if ‘**X’** bucket removed the log folder in it also get removed then how could we know who deleted ‘**X’** bucket.

* 1. Click on Choose Destination button
  2. Click on Save Changes button

1. Aws CloudTrail data events – By configuring S3 to CloudTrail you can know who did what ( like who add file, who delete file etc).
2. Event notifications – Sends notification when specific events(Ex. Deleting file) occur in your Bucket.
   1. Click on Create Event Notification button
   2. Event name – Ex. Delete Notification
   3. At Object Creation – You can select the request events (like Object PUT, POST etc)
   4. At Object removal, enable **Permanently deleted** and **Delete marker created** options.
   5. At **Destination**, enable **SNS topic** -> select **Choose from your SNS topic** to Specify SNS topic. Select the topic at SNS topic
3. Requester pays – If you the Buckets owner Enables the Request Pays setting. If You are hosting public data then you don’t have to bear the cost of others(Requester) downloading large amounts of data from your bucket. **However, for uploads, still paid by bucket owner.**

**Important :** Requester must use his own AWS account(Where bank details were provided) to download an object.

**AWS S3 Bucket’s “Cross Region Replication (CRR)”:**

**CRR** = Automatically copy files from one S3 bucket in one region to another S3 bucket in another region.

**Cross-Region** = Different AWS regions.

**Usage** : Disaster recovery – if Mumbai region goes down, data is safe in London region

**Example**:

**Source bucket** = Mumbai

**Destination bucket** = London

**How to Enable CRR:**

**Rules:**

1. Versioning must be enabled on both source and destination.
2. IAM role must allow replication.
3. Open Amazon S3
4. Select/open a bucket = source bucket **Ex**: *MovieBucket*
   * 1. At **properties** attribute, Enable **Bucket Versioning.**
5. Create a new bucket in a different region ( or use an existing one in different region)

Enable versioning here too.

1. Open “*MovieBucket*” Source Bucket -> **Management** attribute at top -> **Replication rules** -> Create rule.
   * 1. Choose scope (i.e., Entire bucket or specific prefix/folder)
     2. Choose destination bucket (Another region bucket)

Example: same region lo source destination **buckets** use chesi test cheyyi.

* + 1. AWS will ask if you want to create a new IAM role automatically (say yes)
    2. Save the rule, then replication starts automatically for new objects.
    3. **NOTE:** Old files (already existing) will not be replicated unless you copy them manually.

**We can S3 bucket as serverless website:**

1. Open S3 console
2. Create a bucket **Ex:** *AmazonKart* at **General Purpose Buckets**.
3. Upload a **index.html** file in that bucket
4. At **Permission**, Allow public access to **index.html.**
5. open *AmazonKart* bucket, at **properties** attribute, at Static website hosting, click on **Edit** button.
6. Enable the “**Static website hosting**”.
7. Add the static html file i.e., above **index.html** file at Index document.
8. Click on save changes button.
9. Finally, website link will be created which host the static **html**, **CSS**, **JS** files

**S3 Bucket “Management” attribute:**

1. **Open S3 console**
2. Open a Bucket
3. At Management ,

Lifecycle Rules: Used to automatically manage the objects/files stored in bucket over time.

* It helps you to **movie** files to **cheaper storage classes**(like **Archive**, **Infrequent** **Access**)

after a certain period of time.

* It helps you **Delete** old files automatically after they are no longer needed.
* **Example**: Suppose you store daily log files in a bucket.

Here you can set a **Lifecycle Rule** like

After 30 days, move logs to a cheaper “**Infrequent Access**” storage.

After 365 days, delete the logs automatically.

* **Advantage:** **Save storage costs** by moving data to cheaper storage. **Manage storage automatically without manual deletion**. Keep the bucket clean by removing old files.

Steps to create Lifecycle Rule:

1. Lifecycle rule name : **Ex**: Delete after 90 days
2. Choose a rule scope: You can limit the scope to specific folder in bucket or you can apply to all objects in a bucket.
3. Lifecycle rule actions : Select “**move current versions of object btw storage classes.”**

andselect “**Expire current versions of objects**”

1. Transition current versions of object btw storage classes :
   1. Choose storage class transitions: **Ex:** standard-IA
   2. Days after object creation: **Ex:** 31 days
      * Click on Add transition button to add more transitions
2. Choose storage class transitions: **Ex:** Glacier Instant Retrieval
3. Days after object creation: **Ex:** 45days
4. Expire current versions of objects:
   1. Days after object creation : **Ex:** 90 days

**IAM role:**

* **IAM** = **Identity and Access Management(**AWS security service)
* Role = Oka AWS service (like S3,Ec2) use chesi vere service mide pani cheyadaaniki permissions ivvadam
* AWS services cannot automatically access other AWS services for security reasons.

**Example:**

* + X-region S3 wants to copy data to another Y-region s3 bucket
  + Ec2 wants to access s3 bucket.
  + **When setting Cross Region Replication**. AWS asks – “which IAM Role should be used to do replication?”. Then you usually select “Create new IAM Role automatically”. Then AWS will create a safe IAM Role with correct permissions(behind the scenes).

**S3 bucket policies:**

**S3 Bucket Policy** is **a set of permissions/rules** in **JSON / YAML** format.

Ee rules dwara evaru (IAM role, AWS account, Aws service) yeh bucket lo **READ**, **WRITE**, **DELETE** chese access ivvali anedi decide cheyachu.  
**NOTE:** Ivi Bucket-level permissions ni manage chestayi but not individual object-level kaadu.

**In a Simple note** – **AWS S3 bucket ki access ni control cheyadam kosam vadatharu.**

**Use Cases:**

* Bucket ni public cheyadam.
* Specific IAM users/accounts ki access ivvadam.
* Read/Write ni separate ga restrict cheyadam.
* Ip address base cheskoni access control Cheydam.

Phone lo json format examples unai copy past them or Official AWS S3 documentation(docs.aws.amazon.com/AmazonS3/latest/userguide/example-bucket-ploicies.html) lo kuda format example undi

**JSON Format Explanation:**

‘ \* ‘ means all/Everybody  
value of **Version** property is fixed.

We can provide multiple values to **statement** object parameter

**Principal** parameter specify **to whom** we are providing permission. [**Ex**. IAM account, where ‘\*’ specifies all, ]  
**Effect** property specifies **Allow** or **Deny**

**Action** property specifies **READ**, **WRITE**, **DELETE** requests(Ex: S3:\* - allows/Deny all requests)  
**Resource** property specified which specific bucket or files in it.(where your-bucket-name/\* specifies all files in your bucket).

**Sid** parameter specifies the meta data of permission.

**Condition** parameter specifies conditional object parameters.

**Steps to add S3 Bucket policy:**

AWS provides a service **(awspolicygen.s3.amazon.com/policygen.html)** which generates JSON object based on your provided values Instead of manually writing JSON object.

**Important:** Remove “**Id**” parameter if the object generated in BSON format.

**NOTE:** You can even check the generated format is following the JSON syntax or not at **www.jsonformatter.org**

1. Copy the generated JSON object.
2. Open S3 console
3. Choose and open the bucket
4. Click on **Permissions** attribute.
5. Scroll down to **Bucket policy**
6. Click on **Edit** button.
7. Paste the JSON object we copied earlier inside the **Policy** input box.
8. You can also generate JSON format in S3 console itself by clicking on **Add new statement** , **Add Actions**, **Add** **resource**, **Add condition** buttons you see beside **Policy** input box.
9. **NOTE:** AWS also provide policy Examples option at top of page. We can click on it which redirects to **AWS S3 documentation** where you could see multiple pre-define permission policies in JSON format. Copy any of them and paste. Remember to change the Bucket names as per your AWS account before saving it.
10. Click on **save changes** button**.**