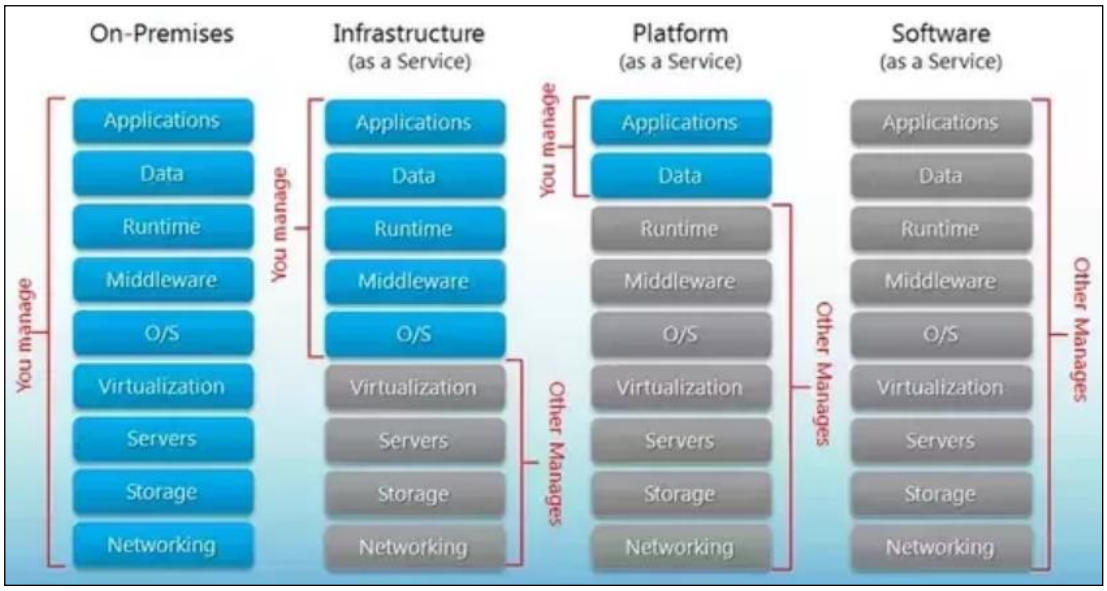
AWS

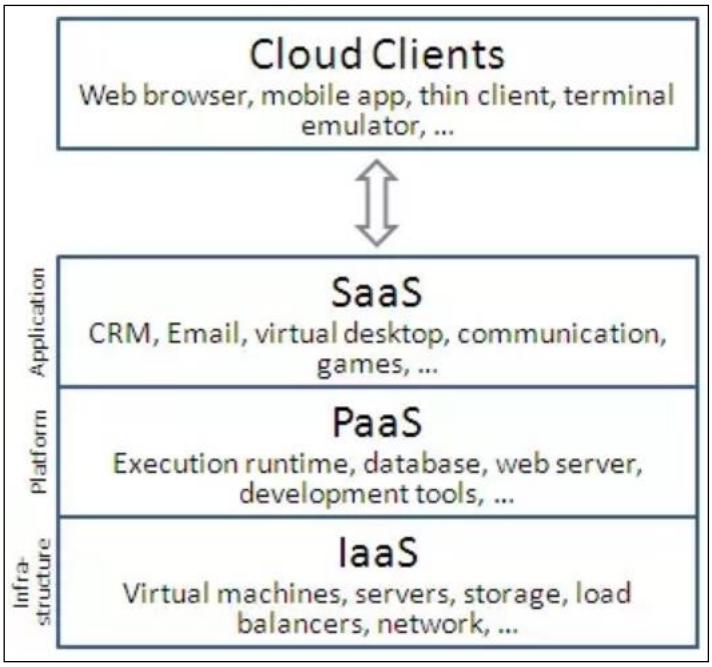
Contents

1. Cloud Introduction



1.1. Type of Cloud

- Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud.
- Amazon Web Services (AWS) EC2 is IaaS.
- You can get virtual machines of any size and configuration and run variants of Linux or Windows on it.



2. EC2-Instance

Steps to Create the EC2 machine on AWS.

A. Login and access to AWS services

- Login to your AWS account and go to the **AWS Services** tab at the top left corner.
- Open all the **services** and click on **EC2** under **Compute** services. This will launch the **dashboard of EC2**.
- On the top right corner of the EC2 dashboard, choose the **AWS Region** in which you want to provision the EC2 server.
- Once your desired Region is selected, come back to the EC2 Dashboard → Click on **'Launch Instance'** button.

B. Choose AMI

- It will be asked to choose an **AMI** of your choice → Choose **Ubuntu server 18.04** image.

C. Choose EC2 Instance Types

- choose **t2.micro - Free tier eligible** instance type, which is a 1vCPU and 1GB memory server offered by AWS
- Click on "**Next: Configure Instance Details**"

D. Configure Instance → Here you can choose how your instance should run as dedicated or as shared. But for now, you can leave all settings to be **as default**. → Following are the option explanation

- **Number of instances** – it can provision up to 20 instances at a time. **Keep default** 1
- **Purchasing Options**, keep the option of 'Request Spot Instances' **unchecked**
- **Network**: Select the existing VPC or it allow us to create new VPC. **Keep default**
- **Subnets**, it can choose the subnet where you want to place your instance. **Keep default**
- **Auto-assign Public IP:** AWS to assign it an IP automatically. **Keep default**
- **IAM role:** 'None'.
- **Shutdown Behaviour**: Stop
- **Enabled accidental termination**: Checkmark Protect against accidental termination
- **Monitoring**: Keep Default
- **Tenancy**: Shared – Run a shared hardware instance.
- Click **Next: Add Storage**

E. Add Storage: default settings – 8gb

F. Tag Instance

- tagged the instance as a **Web_Server_01**

G. Configure Security Groups

- Creating a new Security Group
- Naming our SG for easier reference
- Defining protocols which we want enabled on my instance.
  Assigning IPs which can access our instance on the said protocols.

Type	Protocol:	Port Range	Source	
SSH	TCP	22	Anywhere:	0.0.0.0/0
HTTP	TCP	80	Anywhere:	0.0.0.0/0
HTTPS	TCP	443	Anywhere:	0.0.0.0/0

- Once, the firewall rules are set- Review and launch

H. Review Instances

- review all our choices and parameters and go ahead to launch our instance.

I. Key Pair:

- In the next step you will be asked to create a key pair to login to you an instance.
  - Create a new key pair
  - Give a name to your key
  - Download and save it in your secured folder
- Once you are done with downloading and saving your key → click **launch instance**.

Validation / Login to EC2 instance

A. Convert the downloaded pem key to ppk file

- From the **Start** menu, choose **All Programs**, **PuTTY**, **PuTTYgen**.
- Under **Type of key to generate**, choose **RSA**. If your version of PuTTYgen does not include this option,
- Choose **Load**. By default, PuTTYgen displays only files with the extension .ppk. To locate your .pem file, choose the option to display files of all types.
- Select your .pem file and choose **Open**. PuTTYgen displays a notice that the .pem file was successfully imported. Choose **OK**.
- To save the key → choose **Save private key**. PuTTYgen displays a warning about saving the key without a passphrase. Choose **Yes**.
- Specify the same name for the key and choose **Save**.

B. To connect to your instance using PuTTY

1. Start PuTTY (from the **Start** menu, choose **All Programs, PuTTY, PuTTY**).
2. In the **Category** pane, choose **Session** and complete the following fields:
   a. In the **Host Name** box, do one of the following:
      - (Public DNS) To connect using your instance's public DNS name, enter *ec2-user*@*my-instance-public-dns-name*.
   b. Ensure that the **Port** value is 22.
   c. Under **Connection type**, select **SSH**.
3. Expand **Connection**, expand **SSH**, and then choose **Auth**. Complete the following:
   a. Choose **Browse**.
   b. Select the .ppk file that you generated for your key pair and choose **Open**.
   c. (Optional) If you plan to start this session again later, you can save the session information for future use. Under **Category**, choose **Session**, enter a name for the session in **Saved Sessions**, and then choose **Save**.
   d. Choose **Open**.
4. Choose **Yes**. A window opens and you are connected to your instance.

C. Login to the EC2 using ppk file from Linux

```
# ssh -i /path/my-key-pair.pem ec2-user@public-dns-name
```

D. Transfer files to Linux instances using an SCP client

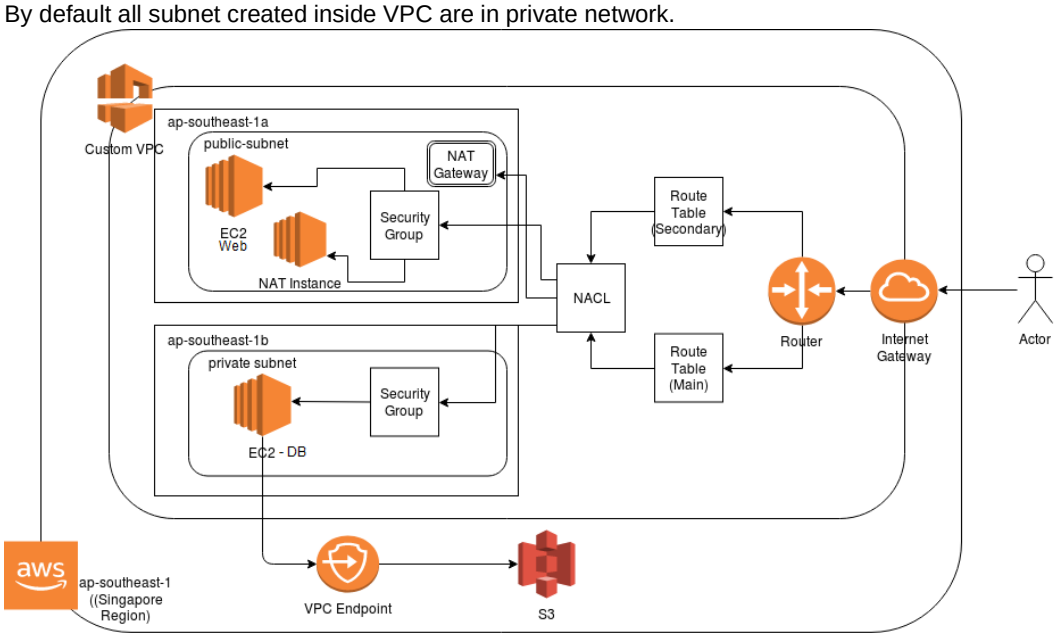
```
# scp -i /path/my-key-pair.pem source_file ec2-user@public-dns-name:dst/

# scp -i /path/my-key-pair.pem /user/my-file.txt ec2-user@public-dns-name:/user/file.txt
```

E. To transfer a file to a destination on your computer

```
# scp -i /path/my-key-pair.pem ec2-user@public-dns-name:/user/file.txt /user3/my-file2.txt
```

3. VPC

- Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources into a virtual network.
- Amazon VPC is the networking layer for Amazon EC2.
- There's no additional charge for using a VPC.
- There are charges for some VPC components, such as NAT gateways, Reachability Analyzer, and traffic mirroring.
- It is logically isolated from other virtual networks in the AWS Cloud.
- By default all subnet created inside VPC are in private network.



3.1. Concepts for VPCs:

- **Virtual private cloud (VPC)** — A virtual network dedicated to your AWS account.
- **Subnet** — A range of IP addresses in your VPC. Or Partition creation inside VPC called Subnet. It is a subdivision of a VPC. Breaking the network down into smaller networks (subnets) is called subnetting.
- **Route table** — A set of rules, called routes, that are used to determine where network traffic is directed.
- **Internet gateway** — A gateway that you attach to your VPC to enable communication between resources in your VPC and the internet. We use an internet gateway to make a subnet public.
- **VPC endpoint** — Enables you to privately connect your VPC to supported AWS services.
- **NAT Gateway** - When you want only a certain set of resources to be allowed publicly on the internet, you can use a NAT gateway. NAT is short for Network Address Translation, which means that it translates private IP addresses to public IPs.

3.2. Creating a VPC

- Login and access to AWS services → Network and Content Delivery → VPC
- Choose the option → Creating the VPC – on the right of the navigation bar.

- Click Start VPC. Now click VPC With a Single Public Subnet option on the left.
- fill in the required details – such as VPC name and subnet name and Leave the other boxes as **default** and click **Create VPC**.
  - IP CIDR block: **10.0.0.0/16**
  - VPC Name: **MyVPC**
  - Public subnet: 10.0.0.0/24
  - Availability Zone: No Preference – Default
  - Subnet name: public subnet – Default
  - Enable DNS hostnames: Yes
  - Hardware tenancy: Default

3.3. Create Subnet – WebSN

- Open the Amazon VPC console:
- Choose VPC Dashboard, then click on Subnets and finally click on Create Subnet.
- Enter the following values in the dialog box:
  - Name tag: **WebSN**
  - VPC: Select the VPC that you created above. – **MyVPC**
  - Availability Zone: Choose the Availability Zone.
  - IPv4 CIDR block: **10.0.1.0/24**
- Click on Create and Close on the confirmation page

3.4. Create Subnet – DB-SN

- Open the Amazon VPC console:
- Choose VPC Dashboard, then click on Subnets and finally click on Create Subnet.
- Enter the following values in the dialog box:
  - Name tag: **DB-SN**
  - VPC: Select the VPC that you created above. – **MyVPC**
  - Availability Zone: Choose the Availability Zone.
  - IPv4 CIDR block: **10.0.2.0/24**
- Click on Create and Close on the confirmation page

3.5. Make Subnet Public WebSN – through Internet Gateway

- To Make Subnet as Public 2 Steps need to follow

1. **Assign the Public IP to Subnet**
   - Choose VPC Dashboard, then click on Subnets → Select the Subnet.
   - Click on **Action** → Modify Auto Assign IP Setting → Enable Auto Assign Public (IPv4 Address) → Save
2. **Create Internet Gateway and attached to VPC**
   - Navigate to the AWS console → Services.
   - Under the Networking & Content Delivery section, choose VPC.
   - Navigate to Virtual Private Cloud -> Internet Gateways.

- Click **Create Internet Gateway**.
  - Name tag: **my-vpc-gateway**
- Click **Attach to VPC**.
- Select your VPC from the Name tag drop-down list and click **Yes, Attach**

3. **Attach the internet Gateway to VPC if not done**
   - Select the **my-vpc-gateway** → Click Action → Attach to VPC → Select VPC(MyVPC) → Attach

4. **Create the Route Table**
   - Navigate to the AWS console → Services.
   - Under the Networking & Content Delivery section, choose VPC.
   - Navigate to Virtual Private Cloud → Click Route Table.
   - Click Route Table
     - Name Tag: **InternetRT**
     - VPC**: MyVPC**
   - Click Create → Close

5. **Connect – First end Route Table to Subnet**
   - Select the Route Table (InternetRT)→ Subnet Associate tab → Edit Subnet Associate → Select the Subnet (10.0.1.0/24) - **WebSN** → Save

6. **Connect – Second end Route Table to Internet Gateway**
   - Select the Route Table (InternetRT) → Route tab → Edit Route → Add Route
     - Target Gateway: **my-vpc-gateway**
     - Destination: 0.0.0.0/0
   - Save Route

3.6. Provide Internet Access to DB-SN Subnet– through NAT

1. **Create the NAT Gateway**
   - Open the Amazon VPC console → choose **NAT Gateways**.
   - Choose **Create NAT Gateway**
     - Name: This is tag of NAT - **Optional**
     - Subnet: **10.0.1.0/24**
     - Create new **Elastic IP**
     - Choose **Add new tag** and enter the key name and value. – **Optional**
     - Choose **Create a NAT Gateway**.

2. **Create the Route Table**
   - Navigate to the AWS console → Services.
   - Under the Networking & Content Delivery section, choose VPC.
   - Navigate to Virtual Private Cloud → Click Route Table.
   - Click Route Table
     - Name Tag: **NAT-RT**
     - VPC**: MyVPC**
   - Click Create → Close

3. **Associate NAT to Subnet**

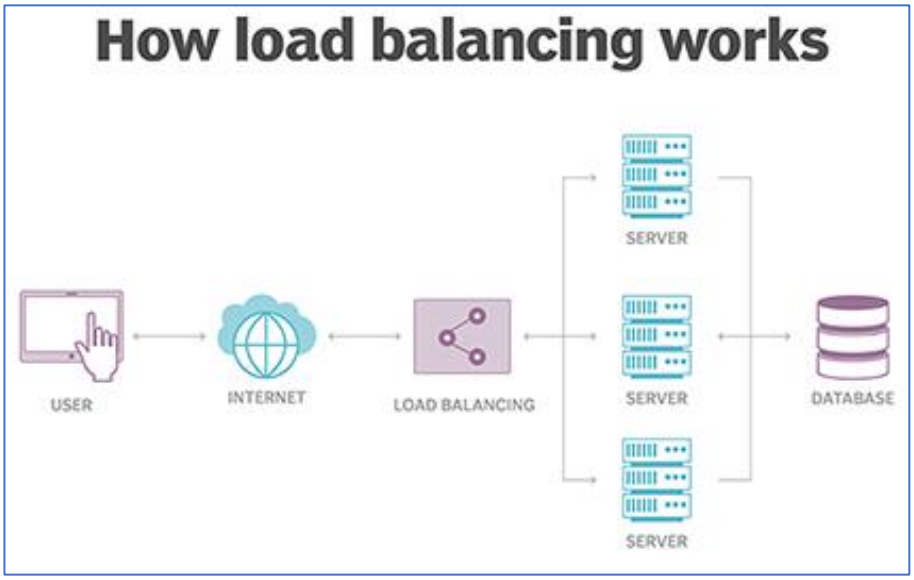
   Select NAT-RT → Subnet Associate → Edit Subnet Associate → Select Private Subnet → Save

4. **Add Route to NAT**
   - Select NAT-RT → Route → Edit Route → Add Route →
     - Target: NAT Gateway
     - Destination: 0.0.0.0/0
   - Save Route

4. Load Balancer

4.1. Introduction

- A **load balancer** distributes incoming application traffic across multiple **EC2** instances in multiple Availability Zones.
- This increases the fault tolerance of your applications.

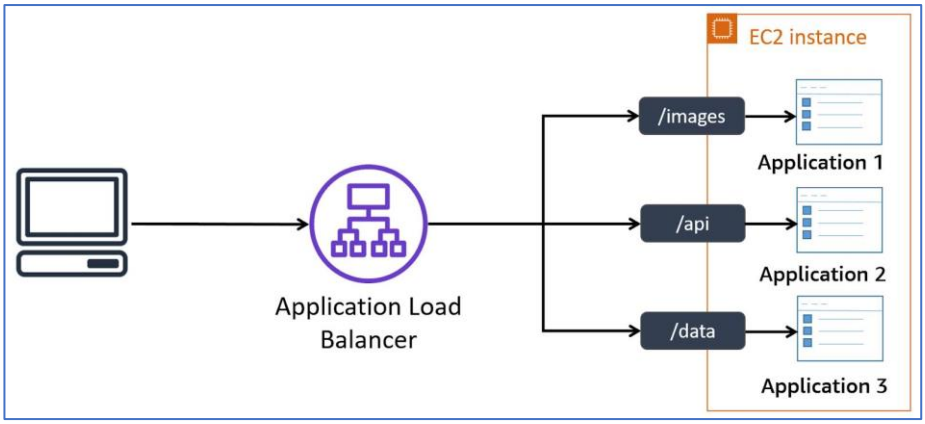


4.2. Type of Load Balancer

1. Application Load Balancer
2. Network Load Balancer
3. Classic Load Balancer
4. Gateway Load Balancer

4.2.1. Application Load Balancer

- This is Layer-7 load balancer, HTTP and HTTPS listeners only.
- Application Load Balancer is particularly useful for websites and mobile apps running in containers.
- It is ideal for microservices or container-based architectures where there is a need to route traffic to multiple services or load balance across multiple ports on the same EC2 instance.



Create an Application Load Balancer

- Ensure that the virtual private cloud (VPC) for your load balancer has at least one public subnet in each Availability Zone used by your targets.

Following Task are required to Create Application Load balancer

- Step 1: Configure Load balancer
- Step 2: Configure security settings
- Step 3: Configure security group
- Step 4: Configure Routing
- Step 5: Register Targets
- Step 6: Review

1. Step 1: Configure Load balancer

- Open the Amazon EC2 console at **https://console.aws.amazon.com/ec2/**
- On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
- Choose **Create Load Balancer**.
- For **Application Load Balancer**, choose **Create**.
  - For Name: **my-alb**
  - Scheme: **internet-facing**
  - Ip address Type: **ipv4**
- For **Listeners**, the default is a listener that accepts **HTTP** traffic on port **80**.
- **Availability Zones**
  - VPC: Select the VPC which created.
  - Availability Zones: Select **2 availabilit**y zones
- **Tags**

Key	Value
Name	**my-alb**

- Click on **Next: Configure Security Settings**

2. Step 2: Configure Security Settings

Note: When you use HTTPS for your load balancer listener then configure the required security settings, and it must deploy an SSL certificate on your load balancer.

- For **Select default certificate**, do one of the following:
  - If you created or imported a certificate using AWS Certificate Manager, select **Choose a certificate from ACM**, and then select the certificate from **Certificate name**.
  - If you uploaded a certificate using IAM, select **Choose a certificate from IAM**, and then select the certificate from **Certificate name**.
- For **Security policy**, we recommend that you keep the default security policy.
- Choose **Next: Configure Security Groups**.

3. Step 3: Configure Security Group

- Choose **Create a new security group** OR **Select and Existing Security Group**
  - For **Create a new security group**:- Enter a name and description for the security group or keep the default name and description. This new security group contains a rule that allows traffic to the port that you selected for your load balancer on the **Configure Load Balancer** page.
  - For **Select and Existing Security Group**: - Select the Security Group .
- Choose **Next: Configure Routing**.

4. Step 4: Configure Routing

- **Target Group**
  - Target Group: **New Target Group**
  - Name: **myalb**
  - Target Type: **instance**
  - Protocol: **HTTP**
  - Port: **80**
- **Health Check**
  - Protocol: **HTTP**
  - Path: **/index.html**
- **Advance Health Check setting**
  - Port: **traffic Port**
  - Healthy threshold: 5
  - Unhealthy threshold: 2
  - Timeout: 5
  - Internal: 30
  - Success Code: 200
- Click **Next: Register Targets**

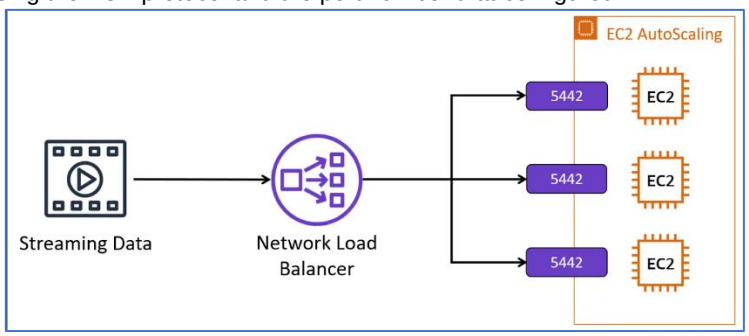
5. Step 5: Register Targets

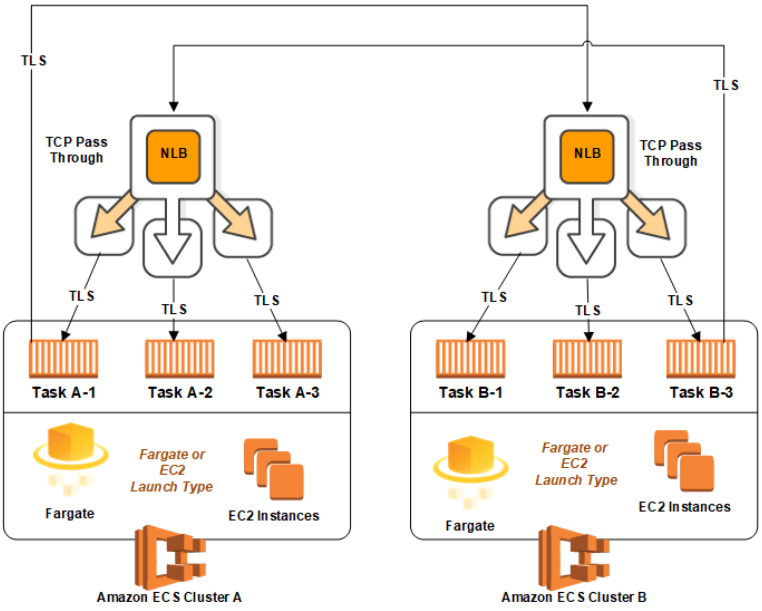
- For **Instances**, select one or more instances.
- Enter the instance listener port, and then choose **Add to registered**.
- When you have finished registering instances, choose **Next: Review**.

6. Review → Click Create

4.2.2. Network Load Balancer

5. A Network Load Balancer functions at the 4th layer of the Open Systems Interconnection (OSI) model.
6. Network Load Balancing (NLB) is a feature that distributes network traffic among multiple servers or virtual machines within a cluster.
7. The Network Load Balancing feature uses the TCP/IP networking protocol to route traffic to different hosts.
8. A listener checks for connection requests from clients, using the protocol and port that you configure, and forwards requests to a target group.
9. Each **target group** routes requests to one or more registered targets, such as EC2 instances, using the TCP protocol and the port number that configured.





Create a Network Load Balancer

- Ensure that the virtual private cloud (VPC) for your load balancer has at least one public subnet in each Availability Zone used by your targets.

Following Task are required to Create Network Load balancer

- Step 1: Configure Load Balancer
- Step 2: Configure Security Group
- Step 3: Configure Routing
- Step 4: Register Target
- Step 5: Review

1. **Step 1: Configure Load Balancer**
   - Open the Amazon EC2 console at **https://console.aws.amazon.com/ec2/**
   - On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
   - Choose **Create Load Balancer**.
   - For **Network Load Balancer**, choose **Create**.
   - **Basic configuration**
     - `Name: My-NLB`
     - `Scheme: internet facing`
   - **Listeners**

Load Balancer Protocol	Load Balancer Port
TCP	80

     - If the listener protocol is TCP, choose **TCP** or **TCP_UDP**.
     - If the listener protocol is TLS, choose **TCP** or **TLS**.
     - If the listener protocol is UDP, choose **UDP** or **TCP_UDP**.
     - If the listener protocol is TCP_UDP, choose **TCP_UDP**.

   - **Availability Zones**
     - `VPC: Select the Created VPC`
     - `Availability Zones: Select 2 zones`
   - Select on **Next: Configure Security Settings**

2. **Step2: Configure Security Settings**

   Note: When you use TLS for your load balancer listener then configure the required security settings.
   - For Target Group, keep the default setting **New Target Group.**
   - For **Name**, type in the name you would like your new Target Group to have.
   - Set **Protocol** and **Port** as required.

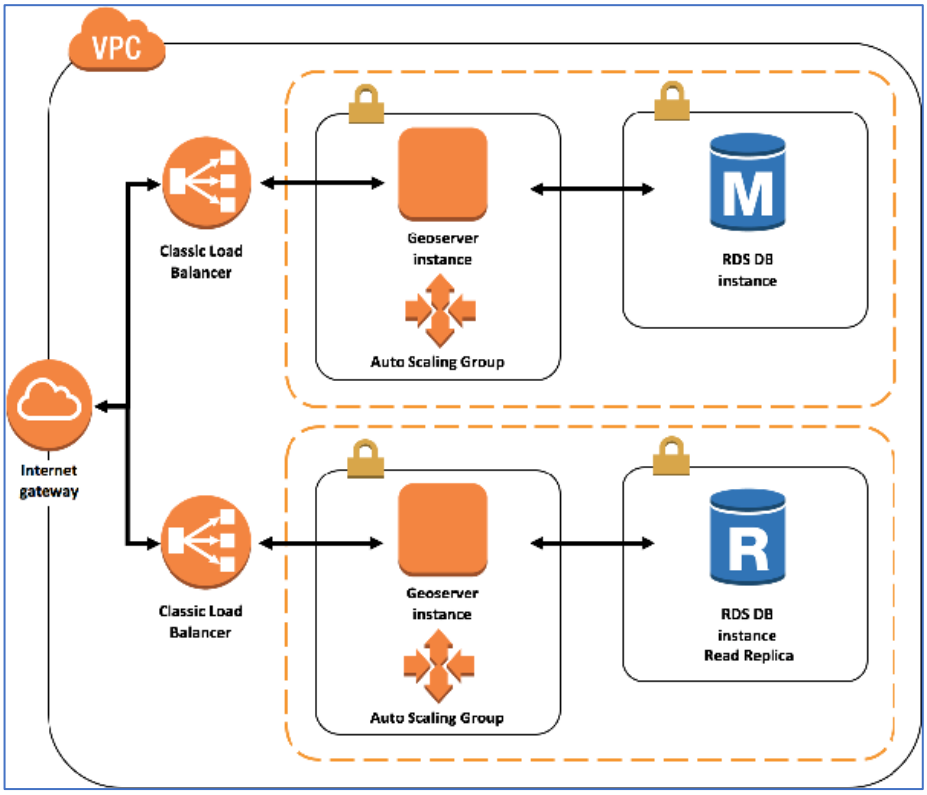
3. **Step 3: Configure Routing**
   - **Target Group**
     - `Target Group: **New Target Group**`
     - `Name: **mynlb**`
     - `Target Type: **instance**`
     - `Protocol: **HTTP**`
     - `Port: **80**`
   - **Health Check**
     - `Protocol: **TCP**`
   - **Advance Health Check setting**
     - `Port: **traffic Port**`
     - `Healthy threshold: 5`
     - `Unhealthy threshold: 2`
     - `Timeout: 5`
     - `Internal: 30`
   - Click **Next: Register Targets**

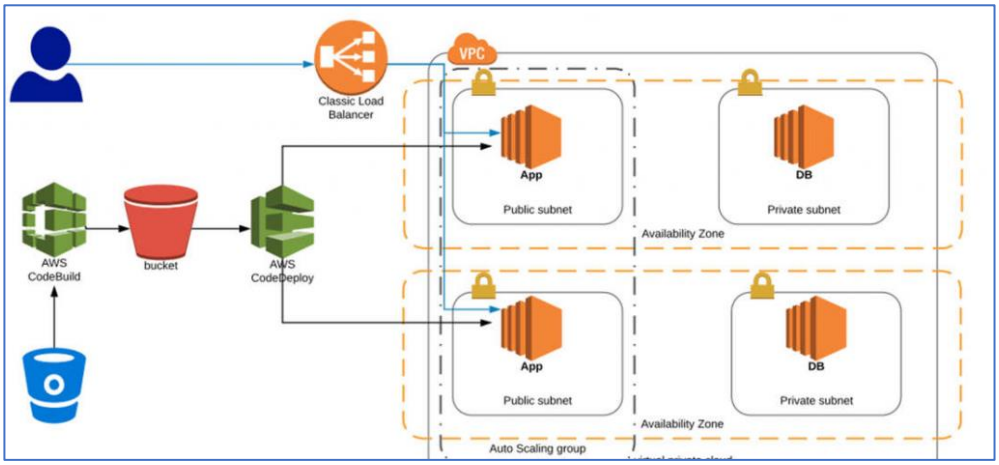
4. **Step 4: Register Target**
   - Register your instances with the target group and click on **Next: Review**

5. **Step 5: Review** and Click **Create**

4.2.3. Classic Load Balancer

- A Classic Load Balancer makes routing decisions at either the transport layer (TCP/SSL) or the application layer (HTTP/HTTPS).
- Classic Load Balancers currently require a fixed relationship between the load balancer port and the container instance port.
- For example, it is possible to map the load balancer port 80 to the container instance port 3030 and the load balancer port 4040 to the container instance port 4040.
- However, it is not possible to map the load balancer port 80 to port 3030 on one container instance and port 4040 on another container instance.





Following Task are required to Create Classic Load balancer

- Step 1: Define Load Balancer
- Step 2: Assign Security Group
- Step 3: Configure Security Settings
- Step 4: Configure Health Check
- Step 5: Add EC2 instance
- Step6: Add Tags
- Step7: Review

1. Step1: Define Load Balancer

- Open the Amazon EC2 console at **https://console.aws.amazon.com/ec2/**
- On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
- Choose **Create Load Balancer**.
- For **Classic Load Balancer**, choose **Create**.
- **Basic configuration**
  - Load Balancer Name: **My-CLB**
  - Create LB inside: **Select the VPC**
  - Create an internal load balancer: default – **No Tick**
  - Enable advance VPC configuration: Tick Mark
  - **Listener Configure:**
    1. Load Balancer Protocol: **HTTP**
    2. Load Balancer Port: **80**
    3. Instance Protocol: **HTTP**
    4. Instance Port: **80**
- **Select Subnet**
  - select at least one available subnet using it add icon.
- Choose **Next: Assign Security Groups**.

2. Step 2: Assign Security Group

- If you selected a VPC as your network, you must assign your load balancer a security group that allows inbound traffic to the ports that you specified for your load balancer
- Assign a security group: Create a new security group
- Security Group Name: my-clb-sg
- Description: For Classic Load Balancer
- Add the Rule

Type	**Protocol**	**Port Range**	**Source**	
Custom TCP Rule	TCP	80	Anywhere	0.0.0.0/0
Custom TCP Rule	TCP	443	Anywhere	0.0.0.0/0

- Choose Next: Configure Security Settings.

3. Step 3: Configure Security Settings

- When you use HTTPS or SSL for your front-end listener, you must deploy an SSL certificate on your load balancer.
- If you configured HTTPS/SSL on the back-end connection, you could enable authentication of your instances.
- For **Select Certificate**, do one of the following:
  - If you created or imported a certificate using AWS Certificate Manager, select **Choose an existing certificate from AWS Certificate Manager (ACM)**, and then select the certificate from **Certificate**.
  - If you imported a certificate using IAM, select **Choose an existing certificate from AWS Identity and Access Management (IAM)**, and then select your certificate from **Certificate**.
- For **Select a Cipher**, verify that **Predefined Security Policy** is selected and set to **ELBSecurityPolicy-2016-08**.
- If you configured the HTTPS listener to communicate with instances using an encrypted connection, you can optionally set up authentication of the instances.
  - For **Backend Certificate**, select **Enable backend authentication**.
  - For **Certificate name**, type the name of the public key certificate.
  - For **Certificate Body (pem encoded)**, copy and paste the contents of the certificate.
- Choose **Next: Configure Health Check**.

4. Step 4: Configure Health Check

- `Ping Protocol: TCP`
- `Pong Port: 22`
- `Advance Details:`
  - `Response Timeout: 5`
  - `Health Check Internal: 10`
  - `Unhealth threshold: 2`
  - `Health threshold: 5`
- Choose **Next: Add EC2 Instances**.

5. Step 5: Add EC2 instance

- On the **Add EC2 Instances** page, select the instances to register with your load balancer.
- Leave cross-zone load balancing and connection draining enabled.
- Choose **Next: Add Tags**.

6. Step 6: Add Tags

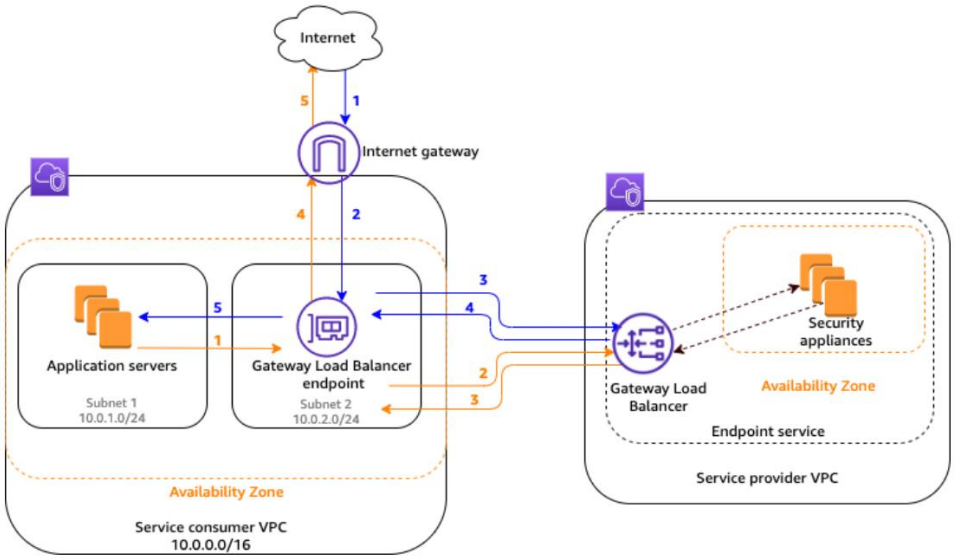
- On the **Add Tags** page, specify a key and a value for the tag.
- To add another tag, choose **Create Tag** and specify a key and a value for the tag.
- After you are finished adding tags, choose **Review and Create**

7. Step 7: Review

- On the **Review** page, check your settings. If you need to make changes, choose the corresponding link to edit the settings.
- Choose **Create**.
- After you are notified that your load balancer was created, choose **Close**.

4.2.4. Gateway Load Balancer

- A Gateway Load Balancer operates at the third layer of the Open Systems Interconnection (OSI) model, the network layer.
- It listens for all IP packets across all ports and forwards traffic to the target group that's specified in the listener rule.
- The Gateway Load Balancer and its registered virtual appliance instances exchange application traffic using the GENEVE protocol on port 6081.
- It supports a maximum transmission unit (MTU) size of 8500 bytes.
- Gateway Load Balancers use Gateway Load Balancer endpoints to securely exchange traffic across VPC boundaries.
- deploy the Gateway Load Balancer in the same VPC as the virtual appliances.



Create a Network Load Balancer

- ensure that the virtual private cloud (VPC) for your Gateway Load Balancer has at least one subnet in each Availability Zone where you have targets.

Following Task are required to Create Network Load balancer

- Step 1: Configure Load Balancer
- Step 2: Configure Routing
- Step 3: Register Target
- Step 4: Review

1. Step 1: Configure Load Balancer

- Open the Amazon EC2 console at **https://console.aws.amazon.com/ec2/**
- On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
- Choose **Create Load Balancer**.
- For **Gateway Load Balancer**, choose **Create**.
- **Basic configuration**
  - `Name: My-GLB`
  - `IP address type: IPv4`
- **Availability Zone**
  - `VPC: Select the VPC`
  - `Availability Zone: Select the availability zone`
- Choose **Next: Register target**.

2. Step 2: Configure Routing

- **Target group**
  - `Target group: New target group`
  - `Name: app-target-group`
  - `Target type: Instance`
  - `Protocol: Port: GENEVE: 6081`
- **Health Check**
  - `Protocol: TCP`
- **Advanced health check settings** choose the health check port, count, timeout, and interval, and then specify success codes.
  - `Port: 80`
  - `Count: 10`
  - `Timeout: 5`
  - `Interval: 5`
  - `Success codes: 200`

3. Step 3: Register Target

- If the target type is **Instances**, select one or more instances, enter one or more ports, and then choose **Include as pending below**.
- If the target type is **IP addresses**, select the network, enter the IP address and ports, and then choose **Include as pending below**.
- Choose **Next: Review**

4. Step 4: Review

- On the **Review** page, check your settings. If you need to make changes, choose the corresponding link to edit the settings.
- Choose **Create**.
- After you are notified that your load balancer was created, choose **Close**.

5. Auto Scaling

5.1. Introduction

- **Autoscaling** is a cloud computing feature that enables organizations to scale cloud services such as server capacities or virtual machines up or down automatically, based on defined situations such as traffic ir utilization levels.

5.2. Benefits of Auto Scaling

- Better fault tolerance
- High availability of resources
- Better cost management
- High reliability of resources
- The high flexibility of resource

5.3. Snapshots vs. AMI

- In Auto Scaling, creating a backup, and restoring the data is an essential part.
- This can be done by creating an EBS instance.
- EBS (elastic block store) is responsible for creating volume backups.
- It consists of two backups, namely, snapshots and AMI.

Snapshots	AMI
It is used as a backup of a single EBS volume attached to the EC2 instance	It is used as a backup of an EC2 instance
opt for this when the instance contains multiple static EBS volumes	This is widely used to replace a failed EC2 instance
Here, pay only for the storage of the modified data	Here, pay only for the storage that you use
It is a non-bootable image on EBS volume	It is a bootable image on an EC2 instance

5.4. Steps to Configure AutoScale

1. Create Load Balancer
2. Create Launch configuration
3. Create Auto Scaling Group – Min, Max
4. Create Topic in SNS (Simple Notification Server)
5. Create Alarm in CloudWatch Service
6. Add Policy in Auto Scaling.

6. CLI Access

Configure the cli into desktop

```
Create IAM user and provide permission / role = AdministratorAccess

C:\> pip install awscli

C:\> aws configure
```

```
PS C:\> aws configure
AWS Access Key ID [None]: AKIAID3WXTZEC3JCAKOA
AWS Secret Access Key [None]: 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
Default region name [None]: us-west-1
Default output format [None]:
```

```
PS C:\> aws ec2 describe-regions --output table
-----------------------------------------------------------
|                     DescribeRegions                     |
+---------------------------------------------------------+
||                        Regions                        ||
|+-----------------------------------+-------------------+|
||             Endpoint              |    RegionName     ||
|+-----------------------------------+-------------------+|
||  ec2.eu-west-1.amazonaws.com      |  eu-west-1        ||
||  ec2.ap-southeast-1.amazonaws.com |  ap-southeast-1   ||
||  ec2.ap-southeast-2.amazonaws.com |  ap-southeast-2   ||
||  ec2.eu-central-1.amazonaws.com   |  eu-central-1     ||
||  ec2.ap-northeast-2.amazonaws.com |  ap-northeast-2   ||
||  ec2.ap-northeast-1.amazonaws.com |  ap-northeast-1   ||
||  ec2.us-east-1.amazonaws.com      |  us-east-1        ||
||  ec2.sa-east-1.amazonaws.com      |  sa-east-1        ||
||  ec2.us-west-1.amazonaws.com      |  us-west-1        ||
||  ec2.us-west-2.amazonaws.com      |  us-west-2        ||
|+-----------------------------------+-------------------+|
```

```
PS C:\> aws ec2 describe-regions --query Regions[].RegionName
[
    "eu-west-1",
    "ap-southeast-1",
    "ap-southeast-2",
    "eu-central-1",
    "ap-northeast-2",
    "ap-northeast-1",
    "us-east-1",
    "sa-east-1",
    "us-west-1",
    "us-west-2"
]
```

EC2-instance creation

```
PS C:\> aws ec2 run-instances --image-id ami-1b0f7d7b --instance-type t2.micro --key-name
norcal
```

To validate ec2

```
PS C:\> aws ec2 describe-instances --instance-ids i-0d477480792637 0a9
```

```
PS C:\> aws ec2 describe-instances --instance-ids i-0d4774807926370a9 --query 'Reservation
s[].Instances[].State'
[
    {
        "Name": "running",
        "Code": 16
    }
]
```

Create tag

```
PS C:\> aws ec2 create-tags --resources i-0d4774807926370a9 --tags 'Key=Name,Value=SRV01'
```

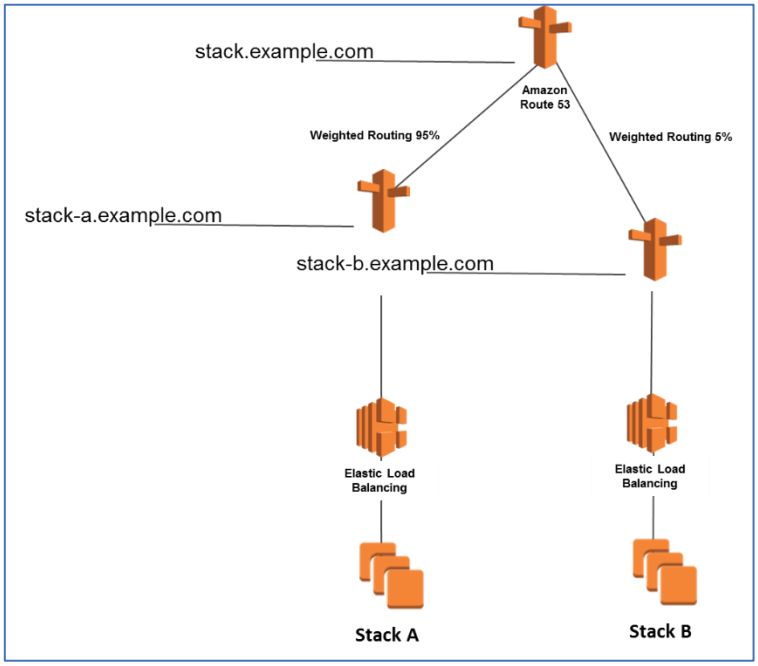
Validate tag

```
PS C:\> aws ec2 describe-tags
{
    "Tags": [
        {
            "Key": "Name",
            "Value": "SRV01",
            "ResourceId": "i-0d4774807926370a9",
            "ResourceType": "instance"
        }
    ]
}
```

7. SSL Certificate

To configure an SSL certificate for HTTPS Protocol and security policy

- For **Select default certificate**, do one of the following:
  - If you created or imported a certificate using AWS Certificate Manager, select **Choose a certificate from ACM**, and then select the certificate from **Certificate name**.
  - If you uploaded a certificate using IAM, select **Choose a certificate from IAM**, and then select the certificate from **Certificate name**.
- For **Security policy**, we recommend that you keep the default security policy.
- Choose **Next: Configure Security Groups**.


stack.example.com
Amazon Route 53
Weighted Routing 95%
Weighted Routing 5%
stack-a.example.com
stack-b.example.com
Elastic Load Balancing
Elastic Load Balancing
Stack A
Stack B