

VPC with Public Private Subnet in Production

About the Project

This example demonstrates how to create a VPC that you can use for servers in a production environment.

To improve resiliency, you deploy the servers in two Availability Zones, by using an Auto Scaling group and an Application Load Balancer. For additional security, you deploy the servers in private subnets. The servers receive requests through the load balancer. The servers can connect to the internet by using a NAT gateway. To improve resiliency, you deploy the NAT gateway in both Availability Zones.

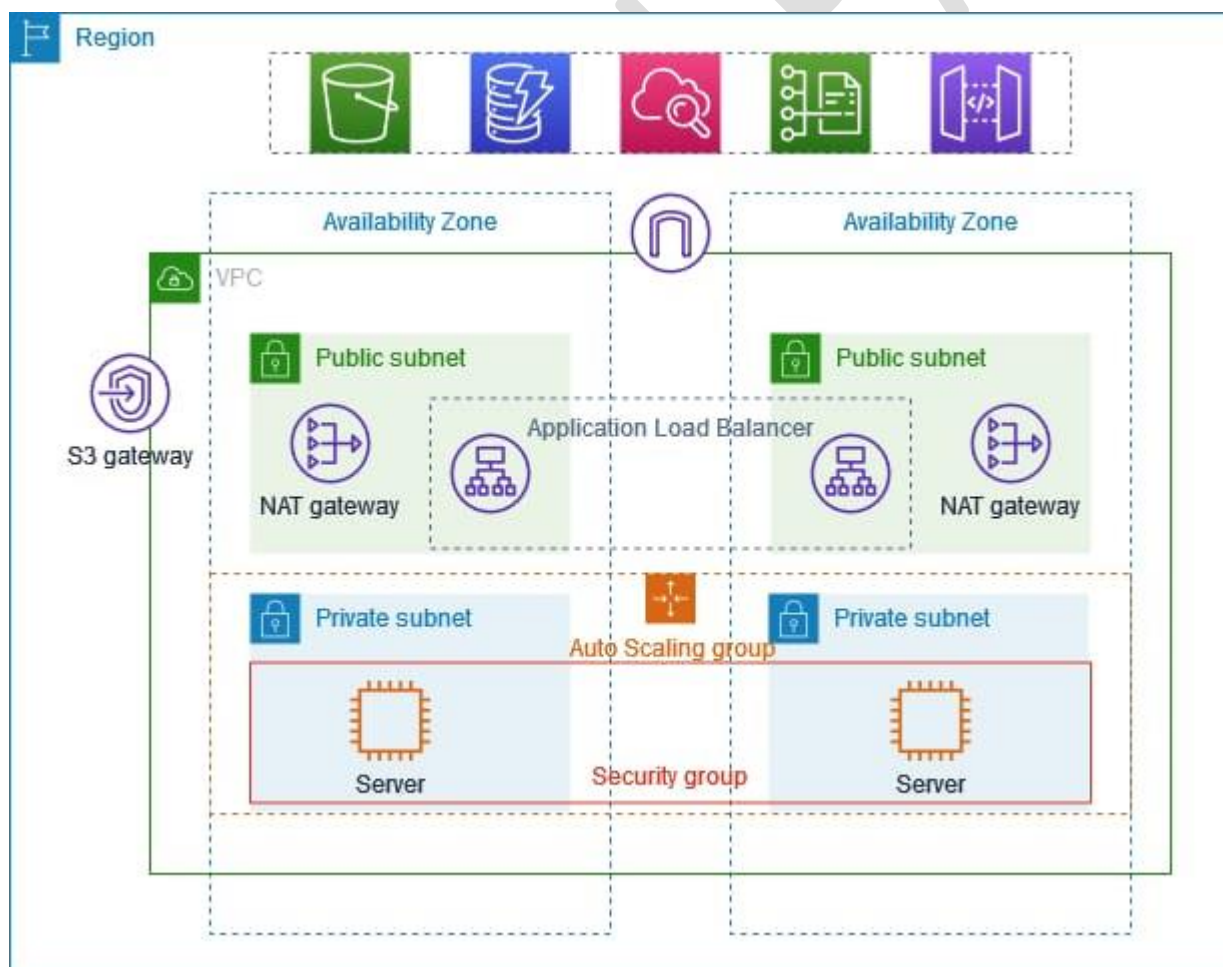


Fig1: VPC with Public-Private subnet

Overview

The VPC has public subnets and private subnets in two Availability Zones.

Each public subnet contains a NAT gateway and a load balancer node.

The servers run in the private subnets, are launched and terminated by using an Auto Scaling group, and receive traffic from the load balancer.

The servers can connect to the internet by using the NAT gateway.

Key steps in this Project

1. Auto scaling Groups
2. Load Balancer
3. Target Group
4. Bastion Host or Jump Server

Auto scaling Groups:

An Auto Scaling group in AWS is a collection of Amazon EC2 instances managed as a unit for the purposes of automatic scaling and management.

- 1. Automatic Scaling:** It can automatically increase or decrease the number of instances based on predefined policies, schedules, or health checks, ensuring that the right amount of compute capacity is available at all times.
- 2. Health Checks:** It performs regular health checks on instances within the group and automatically replaces any unhealthy instances to maintain the desired capacity.
- 3. Dynamic and Predictive Scaling:** Dynamic scaling responds to changing demand in realtime, while predictive scaling uses historical data to anticipate future demand and scale accordingly.
- 4. Load Balancing Integration:** It can be integrated with Elastic Load Balancing (ELB) to distribute incoming application traffic across multiple instances, improving application fault tolerance and availability.

5. Policies and Triggers: Scaling policies can be set to trigger based on CloudWatch alarms, which monitor various metrics like CPU utilization, network traffic, or custom metrics.

6. Cost Management: By automatically adjusting the number of running instances based on demand, it helps optimize costs by avoiding overprovisioning and underprovisioning of resources.

Load Balancer:

A load balancer in AWS is a service that distributes incoming network or application traffic across multiple targets, such as Amazon EC2 instances, containers, IP addresses, and Lambda functions, to ensure high availability and reliability of your applications.

Application Load Balancer (ALB):

Operates at the application layer (Layer 7) of the OSI model.

Ideal for HTTP and HTTPS traffic.

Supports advanced routing capabilities, such as pathbased and hostbased routing.

Integrates with AWS Web Application Firewall (WAF) for added security.

Features:

1. Automatic Distribution: Distributes incoming application traffic across multiple targets to balance the load and prevent any single instance from becoming a bottleneck.

2. Health Checks: Regularly checks the health of targets and only routes traffic to healthy instances, improving application reliability.

3. SSL/TLS Termination: Offloads SSL/TLS encryption and decryption to the load balancer, reducing the load on your application instances.

4. Scalability: Automatically scales to handle changes in traffic without manual intervention.

5. High Availability: Ensures your applications are always available by distributing traffic across multiple targets in multiple Availability Zones.

Target Group:

A target group in AWS is a configuration entity used with load balancers to direct incoming traffic to one or more registered targets, such as Amazon EC2 instances, containers, IP addresses, and Lambda functions. Target groups are associated with load balancers, and they determine how traffic is routed to these targets.

Key Features of Target Groups:

1. Target Types:

Instances: Individual Amazon EC2 instances.

IP Addresses: IP addresses that can be within a VPC or external.

Lambda Functions: AWS Lambda functions for handling serverless architectures.

Application Containers: Containers running on services such as Amazon ECS.

2. Health Checks:

Each target group performs health checks on its registered targets to ensure they are healthy and able to receive traffic.

Health check configurations can include parameters like protocol, path, port, interval, timeout, and the number of successes or failures required for a target to be considered healthy or unhealthy.

3. Port and Protocol Configuration:

Target groups allow you to specify the protocol and port for routing traffic (e.g., HTTP, HTTPS, TCP, UDP).

4. Advanced Routing Features:

For Application Load Balancers (ALBs), target groups support features like pathbased routing, hostbased routing, and advanced request routing rules.

For Network Load Balancers (NLBs) and Gateway Load Balancers (GWLBS), target groups can handle highthroughput and lowlatency requirements.

5. Load Balancer Integration:

A single load balancer can route traffic to multiple target groups, allowing for complex traffic management and distribution.

Different listeners on a load balancer can be configured to route to different target groups based on listener rules.

6. Sticky Sessions:

Target groups can be configured to use sticky sessions (session persistence) to ensure that a user's requests are consistently routed to the same target, useful for stateful applications.

How It Works?

- 1. Create a Target Group:** You define a target group and specify its configuration, including target type, protocol, port, and health check settings.
- 2. Register Targets:** Add targets to the target group by specifying the instances, IP addresses, or Lambda functions.
- 3. Attach to Load Balancer:** Associate the target group with a load balancer listener, which directs traffic to the appropriate target group based on routing rules.
- 4. Health Monitoring:** The load balancer uses the health check settings defined in the target group to continuously monitor the health of the targets and route traffic only to healthy targets.

Bastion Host or Jump Server

A bastion host, also known as a jump server or jump box, is a specialized instance in AWS (or any cloud environment) that acts as a secure gateway for accessing and managing other instances in a private network. It provides an additional layer

of security by serving as the only point of entry for administrators who need to manage instances within a Virtual Private Cloud (VPC).

Why Bastion Hosts are Used?

1. Enhanced Security:

Controlled Access: By funneling all administrative access through a single point, you can tightly control and monitor access to your instances.

Minimized Exposure: Instances in the private subnet do not have public IP addresses, reducing their exposure to the internet and potential attacks.

2. Centralized Logging and Monitoring:

You can set up logging and monitoring on the bastion host to track who is accessing the network and what actions they are performing, enhancing security and compliance.

3. Simplified Network Configuration:

Rather than opening SSH or RDP access to all instances, you only open it for the bastion host, simplifying firewall rules and security group configurations.

How Bastion Hosts Work?

1. Public Subnet Placement:

The bastion host is placed in a public subnet with a public IP address, making it accessible from the internet.

2. Private Subnet Access:

The instances you need to manage are placed in private subnets without direct internet access.

3. Secure Connectivity:

Administrators connect to the bastion host via SSH (for Linux instances) or RDP (for Windows instances). From the bastion host, they can then SSH/RDP into the private instances.

Best Practices for Using Bastion Hosts:

1. Strong Authentication:

Use key pairs (SSH keys) for authentication instead of passwords. Consider using multifactor authentication (MFA) for additional security.

2. Network Security:

Restrict access to the bastion host by configuring security groups to allow only specific IP addresses (e.g., administrator IP addresses) to connect.

3. Logging and Auditing:

Enable detailed logging on the bastion host to keep track of all login attempts and commands executed. Services like AWS CloudTrail and AWS CloudWatch Logs can help with this.

4. Regular Maintenance:

Keep the bastion host updated with the latest security patches and updates. Regularly review and rotate SSH keys.

5. Use of Session Manager:

AWS Systems Manager Session Manager can be used as an alternative to bastion hosts, providing a more secure and managed way to access instances without opening inbound ports.

Example Architecture in this Project

Public Subnet: Contains the bastion host with a security group allowing SSH/RDP access only from specific IP addresses.

Private Subnet: Contains the application instances that need to be managed, with security groups allowing SSH/RDP access only from the bastion host.

Step by step guide to create this Project

You can create a VPC using two methods:

1. VPC only
2. VPC and more

To create a VPC using the "VPC only" method:

Step 1: Create the VPC

1. Open the VPC Dashboard:

Sign in to the AWS Management Console.

Open the VPC dashboard by selecting Services and then VPC.

2. Create a VPC:

Select Your VPCs in the left navigation pane.

Click on Create VPC.

Provide the following details:

Name tag: Enter a name for your VPC, such as VPC-Project.

IPv4 CIDR block: Enter 10.0.0.0/16.

Tenancy: Select default.

Click on Create VPC.

Step 2: Create Subnets

3. Create Public Subnets:

Select Subnets in the left navigation pane.

Click on Create subnet.

Provide the following details for the first public subnet:

Name tag: Enter a name for your subnet, such as PublicSubnet1.

VPC: Select the VPC you created earlier (VPC-Project).

Availability Zone: Select an availability zone (e.g: ap-northeast-1a).

IPv4 CIDR block: Enter 10.0.1.0/24.

Click on Create subnet.

Repeat the above steps to create the second public subnet with the following details:

Name tag: PublicSubnet2

Availability Zone: Select a different availability zone (e.g: ap-northeast-1c).

IPv4 CIDR block: 10.0.2.0/24.

4. Create Private Subnets:

Repeat the steps to create the first private subnet with the following details:

Name tag: PrivateSubnet1

VPC: Select VPC-Project.

Availability Zone: Select the same availability zone as PublicSubnet1 (e.g: ap-northeast-1a).

IPv4 CIDR block: 10.0.3.0/24.

Click on Create subnet.

Repeat the steps to create the second private subnet with the following details:

Name tag: PrivateSubnet2

Availability Zone: Select the same availability zone as PublicSubnet2 (e.g: ap-northeast-1c).

IPv4 CIDR block: 10.0.4.0/24.

Step 3: Create an Internet Gateway and Attach it to the VPC

5. Create Internet Gateway:

Select Internet Gateways in the left navigation pane.

Click on Create internet gateway.

Provide a name for the Internet gateway (e.g., VPCInternetGateway).

Click on Create internet gateway.

6. Attach Internet Gateway to VPC:

Select the Internet gateway you just created.

Click on Actions and then Attach to VPC.

Select your VPC (VPC-Project) and click on Attach internet gateway.

Step 4: Create Route Tables and Associate them with Subnets

7. Create a Route Table for Public Subnets:

Select Route Tables in the left navigation pane.

Click on Create route table.

Provide a name (e.g., PublicRouteTable) and select your VPC (VPC-Project).

Click on Create route table.

8. Create a Route to the Internet:

Select the route table you just created (PublicRouteTable).

Select the Routes tab and click on Edit routes.

Click on Add route.

Destination: 0.0.0.0/0

Target: Select the Internet gateway you created (VPCInternetGateway).

Click on Save routes.

9. Associate Public Subnets with the Route Table:

Select the Subnet associations tab and click on Edit subnet associations.

Select the checkboxes for PublicSubnet1 and PublicSubnet2.

Click on Save associations.

10. Create a Route Table for Private Subnets:

Repeat the steps to create another route table for the private subnets (e.g., PrivateRouteTable).

11. Associate Private Subnets with the Route Table:

Select the Subnet associations tab for the private route table (PrivateRouteTable).

Click on Edit subnet associations.

Select the checkboxes for PrivateSubnet1 and PrivateSubnet2.

Click on Save associations.

Step 5: Create a NAT Gateway (Optional but Recommended for Private Subnets)

12. Allocate an Elastic IP Address:

Select Elastic IPs in the left navigation pane.

Click on Allocate Elastic IP address.

Click on Allocate.

13. Create a NAT Gateway:

Select NAT Gateways in the left navigation pane.

Click on Create NAT gateway.

Provide the following details:

Subnet: Select one of your public subnets (e.g., PublicSubnet1).

Elastic IP allocation ID: Select the Elastic IP address you allocated.

Click on Create NAT gateway.

14. Update the Private Route Table:

Select the private route table (PrivateRouteTable).

Select the Routes tab and click on Edit routes.

Click on Add route.

Destination: 0.0.0.0/0

Target: Select the NAT gateway you created.

Click on Save routes.

To create a VPC using the "VPC and more" method:

1. Open the VPC dashboard and select "Create VPC."
2. In the VPC settings, choose "VPC and more."
3. Assign a name to your VPC.
4. For the IPv4 CIDR block, enter an IP address such as 10.0.0.0/16 (CIDR block size must be between /16 and /28).

5. In the IPv6 CIDR block section, select "No IPv6 CIDR block."
 6. Set the tenancy to "Default."
 7. Select 2 for the number of Availability Zones (AZs).
 8. Choose 2 for the number of public subnets.
 9. Choose 2 for the number of private subnets.
 10. Set NAT gateways to "1 per AZ."
 11. Select "None" for VPC endpoints.
 12. Under DNS options, enable both "DNS hostnames" and "DNS resolution."
- Finally, click "Create VPC." AWS will automatically create all the necessary resources for your VPC.

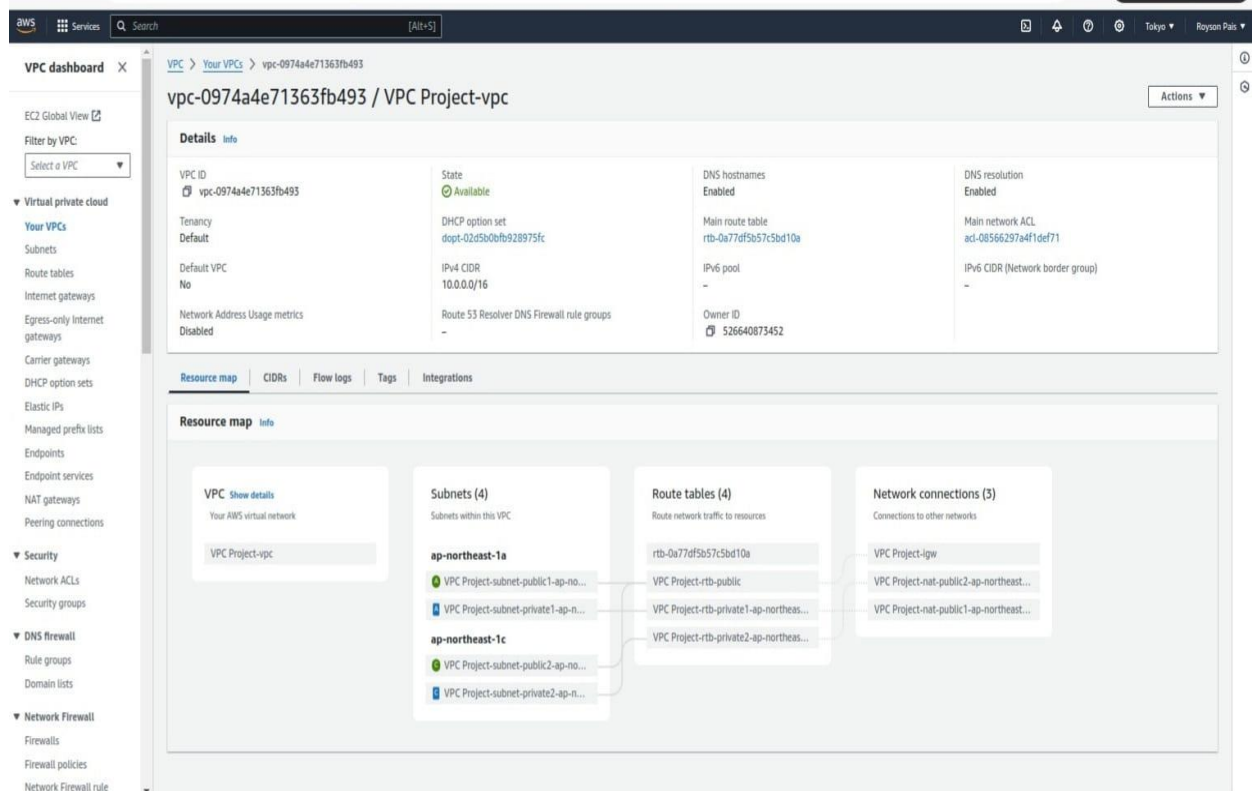


Fig2: VPC created using VPC and more

Auto scaling groups

Prerequisites:

- 1.VPC and Subnets:** Make sure you have a Virtual Private Cloud (VPC) and subnets configured.
- 2. Launch Template or Configuration:** Create a launch template or configuration with the details of the instances to be launched.

Step-by-Step Guide:

Step 1: Create a Launch Template

1. Navigate to the EC2 Dashboard:

Open the AWS Management Console.

Go to the EC2 Dashboard.

2. Create Launch Template:

In the left navigation pane, choose Launch Templates.

Click Create launch template.

Provide a name and description for the launch template.

Specify the Amazon Machine Image (AMI) ID.

Select the instance type.

Configure the instance details (key pair, network settings, storage, security groups, etc.).

create a new security group, give name to security group.

select VPC that you created.

in inbound Security group rules.

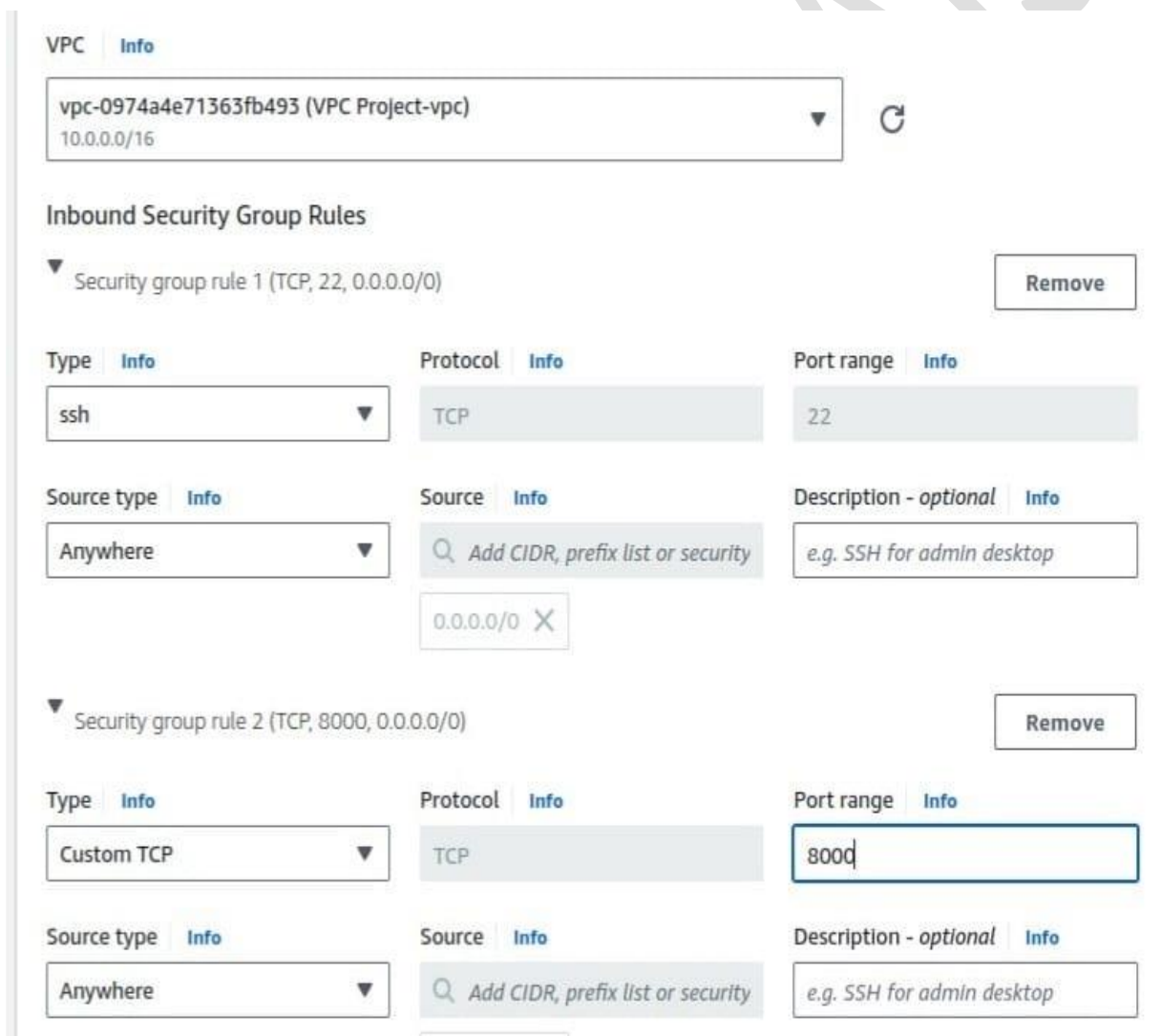
Add rules to allow SSH (port 22) access from your IP address.

Example rule: Type = SSH, Protocol = TCP, Port Range = 22, Source = anywhere.

In this project, I will use port 8000 to deploy the application inside the EC2 instance.

Example rule: Type = Custom TCP, Protocol = TCP, Port Range = 8000, Source = anywhere.

Click Create launch template.



The screenshot displays the AWS VPC console interface for configuring Inbound Security Group Rules. At the top, the VPC is selected as 'vpc-0974a4e71363fb493 (VPC Project-vpc)' with a CIDR of '10.0.0.0/16'. Below this, two security group rules are listed. Rule 1 is for SSH access (Type: ssh, Protocol: TCP, Port range: 22, Source: Anywhere). Rule 2 is for a custom TCP application (Type: Custom TCP, Protocol: TCP, Port range: 8000, Source: Anywhere). Both rules have a description 'e.g. SSH for admin desktop'.

Type	Protocol	Port range	Source type	Source	Description - optional
ssh	TCP	22	Anywhere	0.0.0.0/0	e.g. SSH for admin desktop
Custom TCP	TCP	8000	Anywhere	0.0.0.0/0	e.g. SSH for admin desktop

Fig3: VPC selection and Providing Inbound Security Group Rules

Step 2: Create an Auto Scaling Group

1. Navigate to Auto Scaling Groups:

In the EC2 Dashboard, under Auto Scaling, choose Auto Scaling Groups.

2. Create Auto Scaling Group:

Click Create Auto Scaling group.

Provide a name for the Auto Scaling group.

Select the launch template created in the previous step.

Click Next.

3. Configure Network:

Select the VPC in which to create the Auto Scaling group.

Select the subnets where instances will be launched.(select both private subnets)

Click Next.

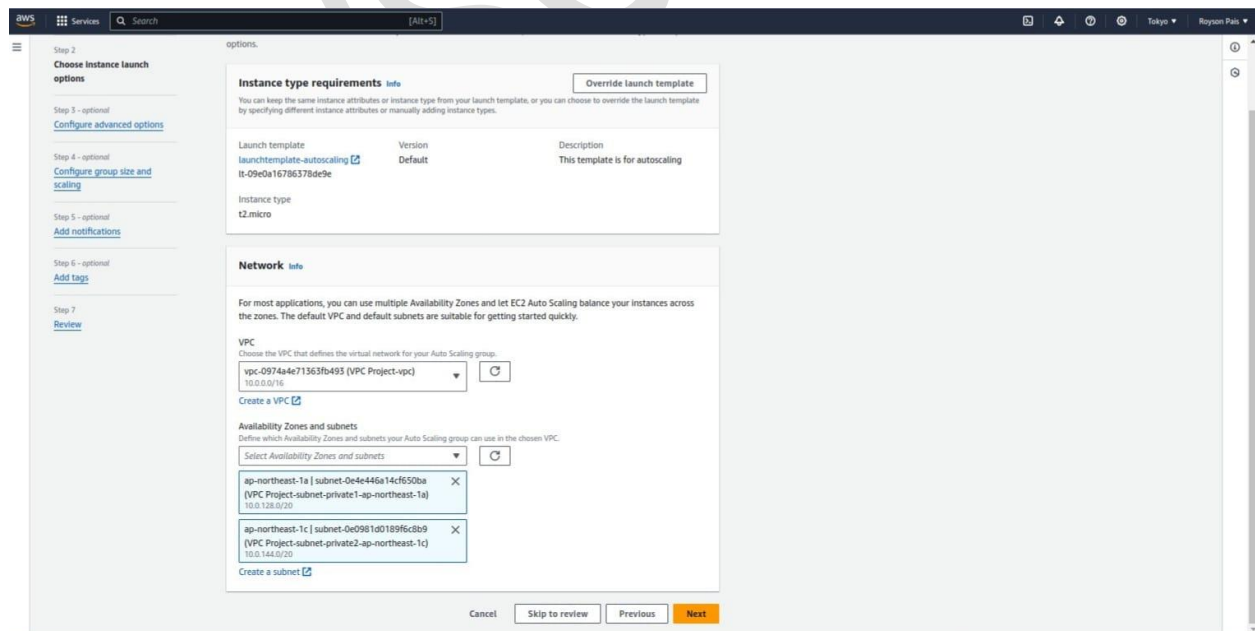


Fig4: Creating Auto scaling Group and Selecting both Private Subnets

4. We will add load balancer in public subnet. So here select no load balancer.

Configure health checks (300 seconds).

Click Next.

5. Configure Group Size and Scaling Policies:

Specify the desired(2), minimum(1), and maximum(4) number of instances.

Configure scaling policies to adjust the number of instances based on demand.

Target Tracking: Adjusts the number of instances to maintain a target value for a specific metric.

Step Scaling: Adjusts the number of instances based on a set of scaling adjustments.

Scheduled Scaling: Adjusts the number of instances based on a schedule.

Click Next.

The screenshot shows the 'Configure group size and scaling' step in the AWS Management Console. On the left, a navigation pane lists steps: 'Choose instance launch options', 'Step 3 - optional: Configure advanced options', 'Step 4 - optional: Configure group size and scaling' (which is the active step), 'Step 5 - optional: Add notifications', 'Step 6 - optional: Add tags', and 'Step 7: Review'. The main content area is divided into two sections. The first section, 'Group size', includes an 'Info' icon and text stating that the initial size can be changed manually or automatically. It features a 'Desired capacity type' dropdown set to 'Units (number of instances)' and a 'Desired capacity' input field with the value '2'. The second section, 'Scaling', also has an 'Info' icon and text explaining that the group can be resized manually or automatically. It contains a 'Scaling limits' section with two input fields: 'Min desired capacity' set to '1' and 'Max desired capacity' set to '4'. Below these fields are labels: 'Equal or less than desired capacity' for the minimum and 'Equal or greater than desired capacity' for the maximum.

[Choose instance launch options](#)

Step 3 - optional
[Configure advanced options](#)

Step 4 - optional
Configure group size and scaling

Step 5 - optional
[Add notifications](#)

Step 6 - optional
[Add tags](#)

Step 7
[Review](#)

Group size Info
Set the initial size of the Auto Scaling group. After creating the group, you can change its size to meet demand, either manually or by using automatic scaling.

Desired capacity type
Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups configured with a set of instance attributes.

Units (number of instances) ▼

Desired capacity
Specify your group size.

2

Scaling Info
You can resize your Auto Scaling group manually or automatically to meet changes in demand.

Scaling limits
Set limits on how much your desired capacity can be increased or decreased.

Min desired capacity
1
Equal or less than desired capacity

Max desired capacity
4
Equal or greater than desired capacity

Fig5: Configure Group Size and Scaling

6. Configure Notifications and Tags:

(Optional) Configure notifications for scaling events.

(Optional) Add tags to the Auto Scaling group.

Click Next.

7. Review and Create:

Review all the configurations.

Click Create Auto Scaling group.

Step 3: Verify and Test

1. Verify:

Ensure the Auto Scaling group has been created successfully.

Check if instances are launched according to the desired capacity.

2. Test Scaling Policies:

Manually adjust the load or use test scripts to trigger scaling policies.

Monitor the scaling activities in the Auto Scaling group.

Bastion Host or Jump Server

Prerequisites:

1. VPC and Subnets: Ensure you have a VPC with both public and private subnets configured.

Step-by-Step Guide:

1. Navigate to the EC2 Dashboard:

Open the AWS Management Console.

Go to the EC2 Dashboard.

2. Create Key Pair:

In the left navigation pane, choose Key Pairs under Network & Security.

Click Create key pair.

Provide a name for the key pair.

Select the key pair file format (PEM for Linux/Mac, PPK for Windows).

Click Create key pair and download the key pair file.

Step 2: Launch a Bastion Host

1. Launch Instance:

In the EC2 Dashboard, click Launch Instance.

Choose an Amazon Machine Image (AMI) for the bastion host (e.g., Amazon Linux 2, Ubuntu, etc.).

Select an instance type (e.g., t2.micro for free tier).

2. Configure Instance Details:

Choose your VPC.

Select a public subnet.

Ensure Autoassign Public IP is enabled.

5. Configure Security Group:

Create a new security group for the bastion host.

Add rules to allow SSH (port 22) access from your IP address.

Example rule: Type = SSH, Protocol = TCP, Port Range = 22, Source = anywhere.

6. Review and Launch:

Review the instance settings.

Click Launch.

Open Terminal on local machine

To copy pem file from local machine to remote server (bastion host) using SCP.

Syntax

```
scp i path/to/privatekey.pem localfilepath  
username@hostname_or_IP:remotedirectory
```

Example command:

```
scp i /home/royson/Downloads/vpcproject-keypair.pem  
/home/edureka_std/Downloads/vpcproject-keypair.pem  
ubuntu@54.95.59.78:/home/ubuntu
```

scp: The command to securely copy files between hosts on a network.

i /home/royson/Downloads/vpcproject-keypair.pem: The option to specify the private key file used for authentication.

/home/royson/Downloads/vpcproject-keypair.pem: The local file path of the file you want to copy.

ubuntu: The username to log in as on the remote server.

54.95.59.78: The IP address of the remote server.

/home/ubuntu: The remote directory where the file will be copied.

This command is for connecting to a remote server using SSH with a specific private key. (To connect bastion host to local machine)

Syntax

```
ssh i "path/to/vpcproject-keypair.pem " username@hostname_or_IP
```

Example command:

```
ssh i "vpcproject-keypair.pem " ubuntu@54.95.59.78
```

ssh: The command to initiate an SSH connection.

i "vpcproject-keypair.pem": The option to specify the private key file used for authentication.

ubuntu: The username to log in as on the remote server.

54.95.59.78: The IP address of the remote server.

This command is for copying a file to a remote server using SCP with a specific private key. (copying pem file from bastion host to a EC2 in a private subnet)

Syntax

```
scp i path/to/vpcproject-keypair.pem localfilepath  
username@hostname_or_IP:remotedirectory
```

Example command:

```
scp i /home/ubuntu/vpcproject-keypair.pem /home/ubuntu/vpcproject-  
keypair.pem ubuntu@10.0.138.249:/home/ubuntu
```

scp: The command to securely copy files between hosts on a network.

i /home/ubuntu/vpcproject-keypair.pem: The option to specify the private key file used for authentication.

/home/ubuntu/vpcproject-keypair.pem: The local file path of the file you want to copy.

ubuntu: The username to log in as on the remote server.

10.0.138.249: The IP address of the remote server.

/home/ubuntu: The remote directory where the file will be copied.

You can create an index.html file on a private subnet EC2 instance using an editor like vi, vim, or nano. If you want to copy an index.html file or any other file from your local machine to a private subnet EC2 instance, you need to first copy the file to the bastion host. From the bastion host, you can then copy the file to the private subnet EC2 instance.

This command is for copying a file to a remote server using SCP with a specific private key. (from local machine to a bastion host)

Syntax

```
scp i path/to/vpcproject-keypair.pem localfilepath  
username@hostname_or_IP:remotedirectory
```

Example command:

```
scp i /home/royson/Downloads/vpcproject-keypair.pem  
/home/royson/index.html ubuntu@54.95.59.78:/home/ubuntu
```

scp: The command to securely copy files between hosts on a network.

i /home/royson/Downloads/vpcproject-keypair.pem: The option to specify the private key file used for authentication.

/home/royson/index.html: The local file path of the file you want to copy.

ubuntu: The username to log in as on the remote server.

54.95.59.78: The IP address of the remote server.

/home/ubuntu: The remote directory where the file will be copied.

This command is for copying a file to a remote server using SCP with a specific private key. (Coping from bastion host to a private Subnet EC2)

Syntax

```
scp -i path/to/vpcproject-keypair.pem localfilepath  
username@hostname_or_IP:remotedirectory
```

Example command:

```
scp -i /home/ubuntu/vpcproject-keypair.pem /home/ubuntu/index.html  
ubuntu@10.0.138.249:/home/ubuntu
```

scp: The command to securely copy files between hosts on a network.

-i /home/ubuntu/vpcproject-keypair.pem: The option to specify the private key file used for authentication.

/home/ubuntu/index.html: The local file path of the file you want to copy.

ubuntu: The username to log in as on the remote server.

10.0.138.249: The IP address of the remote server.

/home/ubuntu: The remote directory where the file will be copied.

This command is for connecting to a remote server using SSH with a specific private key. (From bastion host to connect EC2 instance in the private subnet)

Syntax

```
ssh i "path/to/vpcproject-keypair.pem" username@hostname_or_IP
```

Example command:

```
ssh i "vpcproject-keypair.pem" ubuntu@10.0.138.249
```

ssh: The command to initiate an SSH connection.

i "vpcproject-keypair.pem": The option to specify the private key file used for authentication.

ubuntu: The username to log in as on the remote server.

10.0.138.249: The IP address of the remote server.

Sample index.html file

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
  <meta charset="UTF8">
```

```
  <meta name="viewport" content="width=devicewidth, initialscale=1.0">
```

```
  <title>Sample Index Page</title>
```

```
  <style>
```

```
    body {
```

```
      fontfamily: Arial, sansserif;
```

```
      margin: 0;
```



```
padding: 0;

backgroundcolor: f4f4f4;

color: 333;
}

header {

backgroundcolor: 4CAF50;

color: white;

padding: 1em;

textalign: center;

}

main {

padding: 1em;

}

footer {

backgroundcolor: 333;

color: white;

textalign: center;

padding: 1em;

position: fixed;

bottom: 0;

width: 100%;

}

</style>

</head>
```

```
<body>
```

```
  <header>
```

```
    <h1>VPC with Public-Private Subnet in Production</h1>
```

```
  </header>
```

```
  <main>
```

```
    <h2>About the Project</h2>
```

```
    <p>This example demonstrates how to create a VPC that you can use for  
servers in a production environment. </p>
```

```
  </main>
```

```
  <footer>
```

```
    <p>ROYSON PAIS</p>
```

```
  </footer>
```

```
</body>
```

```
</html>
```

After logging into a private subnet EC2 instance, you need to run a web server to host the application. In this example, we'll use a Python web server.

Command

```
python3 -m http.server 8000
```

Explanation of the command:

python3: is the command to run Python version 3.

-m http.server: tells Python to run the builtin HTTP server module.

8000: specifies the port number on which the server will listen for incoming requests.

Load Balancer

Step 1: Create a Target Group

1. Navigate to the EC2 Dashboard

In the top navigation bar, click on "Services" and then select "EC2" under the "Compute" category.

2. Create a Target Group

In the left navigation pane, under "Load Balancing," click on "Target Groups." Click on the "Create target group" button.

3. Configure the Target Group

Choose a target type: Select "Instances" as the target type.

Target group name: Enter a name for your target group.

Protocol: Select "HTTP."

Port: Enter 8000.

VPC: Select the VPC where your instances are located.

Health checks: Configure health check settings (default values are usually fine for basic setups).

Protocol: HTTP

Path: Leave it as / (or specify a health check path if needed).

Click on "Next."

Target group name

Targetgroup-VPC-Project

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Protocol : Port

Choose a protocol for your target group that corresponds to the Load Balancer type that will route traffic to it. Some protocols now include anomaly detection for the targets and you can set mitigation options once your target group is created. This choice cannot be changed after creation

HTTP 8000

1-65535

IP address type

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

☒ IPv4

Each instance has a default network interface (eth0) that is assigned the primary private IPv4 address. The instance's primary private IPv4 address is the one that will be applied to the target.

☐ IPv6

Each instance you register must have an assigned primary IPv6 address. This is configured on the instance's default network interface (eth0). [Learn more](#)

VPC

Select the VPC with the instances that you want to include in the target group. Only VPCs that support the IP address type selected above are available in this list.

VPC Project-vpc
vpc-0974a4e71363fb493
IPv4 VPC CIDR: 10.0.0.0/16

Fig6: Select Target Group port and VPC

4. Register Targets

Select the instances you want to include in the target group.

Click on "Include as pending below."

Click on "Create target group."

Step 2: Create an Application Load Balancer (ALB)

1. Navigate to Load Balancers

In the left navigation pane, click on "Load Balancers."

2. Create Load Balancer

Click on the "Create Load Balancer" button.

Choose "Application Load Balancer" and click on "Create."

3. Configure Load Balancer

Name: Enter a name for your load balancer.

Scheme: Choose "internetfacing" for a publicly accessible load balancer or "internal" for internal use.

IP address type: Choose "ipv4."

Listeners:

By default, an HTTP listener on port 80 is added. Make sure it's listed.

The screenshot displays two configuration sections in the AWS Management Console. The top section, titled "Security groups", includes an information link and a description: "A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can create a new security group". Below this is a dropdown menu labeled "Security groups" with the placeholder text "Select up to 5 security groups". A selected item is shown in a box: "securitygroups-vpcproject" with ID "sg-0006bba55bd6c03c7" and VPC "vpc-0974a4e71363fb493". The bottom section, titled "Listeners and routing", also has an information link and a description: "A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets." It shows a single listener configuration for "HTTP:80". The "Protocol" is set to "HTTP" and the "Port" is "80". The "Default action" is "Forward to" "Targetgroup-VPC-Project" with "Target type: Instance, IPv4". A "Remove" button is located at the top right of the listener configuration area. A "Create target group" link is at the bottom of the default action dropdown.

Fig7: Select Target Group and Listeners and Routing

4. Configure Availability Zones

Select the VPC and the Availability Zones where your target instances are located.

Ensure that at least one subnet in each selected Availability Zone is chosen.

VPC Project-vpc
vpc-0974a4e71363fb493
IPv4 VPC CIDR: 10.0.0.0/16

Mappings [Info](#)
Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection.

☒ **ap-northeast-1a (apne1-az4)**
Subnet
subnet-0a1cc2b41fade7a76 VPC Project-subnet-public1-ap-northeast-1a
IPv4 address
Assigned by AWS

☒ **ap-northeast-1c (apne1-az1)**
Subnet
subnet-0b166de95e7f51fb5 VPC Project-subnet-public2-ap-northeast-1c
IPv4 address
Assigned by AWS

Fig8: Configure Availability Zones (both Public)

5. Configure Security Settings

Skip this step for HTTP (port 80) since it doesn't require SSL/TLS configuration.

6. Configure Security Groups

Select an existing security group or create a new one.

Ensure the security group allows inbound traffic on port 80 (HTTP).

7. Configure Routing

Target group: Select the target group you created in Step 1.

Name: Ensure it's the correct target group name.

Protocol: HTTP

Port: 8000

8. Register Targets

This step is already handled during the target group creation, so you can proceed.

9. Review and Create

Review all the settings you configured.

Click on "Create load balancer."

Target Group: Configured to use HTTP on port 8000.

Load Balancer: Configured to use HTTP on port 80, routing requests to the target group on port 8000.

EC2 > Security Groups > sg-0006bba55bd6c03c7 - securitygroups-vpcproject

sg-0006bba55bd6c03c7 - securitygroups-vpcproject

Actions

Details

Security group name

securitygroups-vpcproject

Security group ID

sg-0006bba55bd6c03c7

Description

allow ssh

VPC ID

vpc-0974a4e71363fb493

Owner

526640873452

Inbound rules count

3 Permission entries

Outbound rules count

1 Permission entry

Inbound rules

Outbound rules

Tags

Inbound rules (3)

Manage tags

Edit inbound rules

Search

	Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
<input type="checkbox"/>	-	sgr-0896c709ee14c5602	IPv4	Custom TCP	TCP	8000	0.0.0.0/0	-
<input type="checkbox"/>	-	sgr-0dd939e378aecbae8	IPv4	HTTP	TCP	80	0.0.0.0/0	-
<input type="checkbox"/>	-	sgr-04035434d28db4...	IPv4	SSH	TCP	22	0.0.0.0/0	-

Fig9: Configure HTTP Port 80 in the Security Groups

Copy load balancer URL (DNS name) to access page on web

Ex:

<https://loadbalancerforprivate1800066924.ap-northeast-1.elb.amazonaws.com>

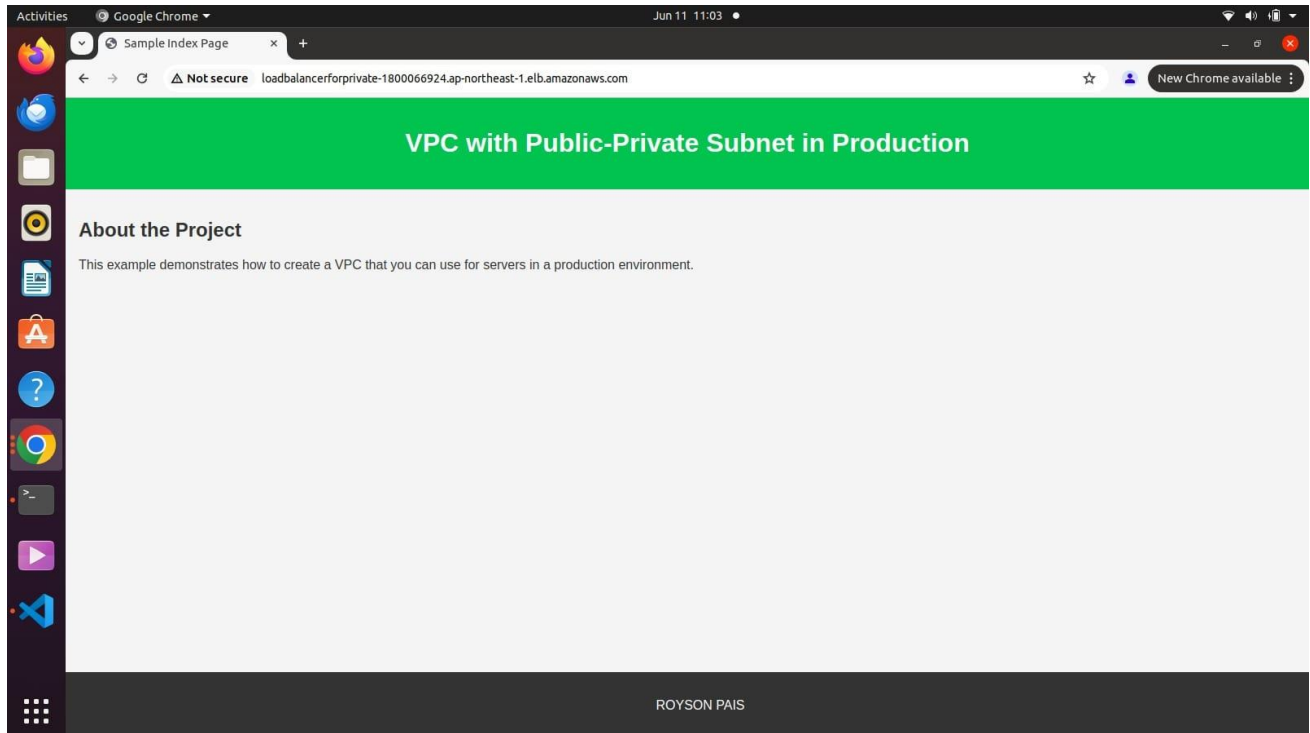


Fig10: Accessing index.html on the Web