

Task-7.

Troubleshooting (Explain)

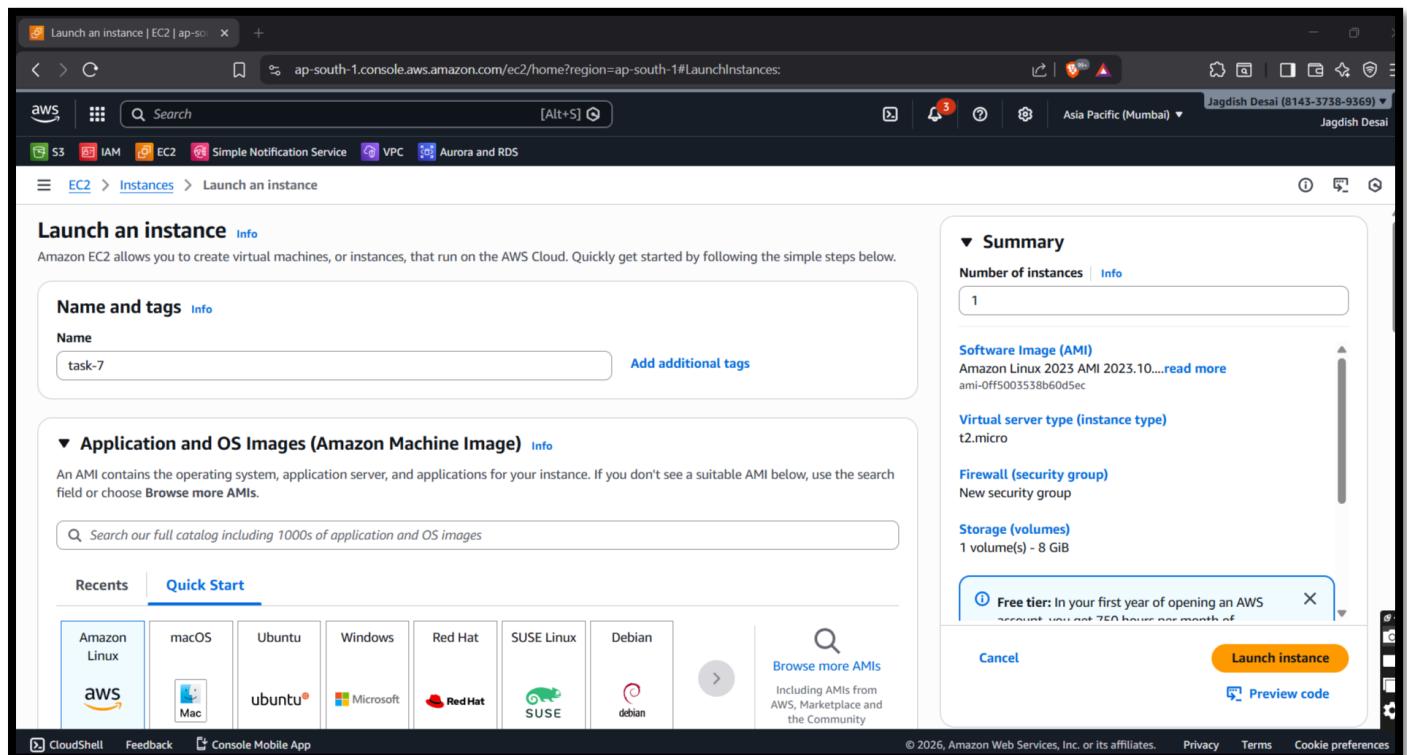
App not accessible

Container running but port not reachable

ALB health check failures

Steps

Step 1: Launch instance



Step 2: Select Default Ami

The screenshot shows the AWS EC2 'Launch an instance' interface. In the 'Amazon Machine Image (AMI)' section, 'Amazon Linux 2023 kernel-6.1 AMI' is selected. The details show it's a 64-bit (x86) instance with uefi-preferred boot mode, AMI ID ami-off5003538b60d5ec, published on 2026-01-23 by ec2-user (verified provider). The 'Description' panel notes it's a modern, general purpose Linux-based OS optimized for AWS. The 'Summary' panel on the right shows 1 instance being launched, using the selected AMI, instance type t2.micro, and other configurations like a new security group and 8 GiB storage. A 'Launch instance' button is prominent.

Step 3: Select Instance Type & key-pair

The screenshot shows the continuation of the 'Launch an instance' process. In the 'Instance type' section, 't2.micro' is selected, described as a current generation instance with 1 vCPU and 1 GiB Memory. The 'Summary' panel remains the same, showing 1 instance launch with the selected t2.micro instance type. The 'Key pair (login)' section allows selecting a key pair named 'geeta'. The 'Network settings' section shows the network interface 'vpc-052859de1479c19a9'. A 'Free tier' message indicates 750 hours per month of usage are available. The 'Launch instance' button is present.

Step 4: Keep all it is and launch instance.

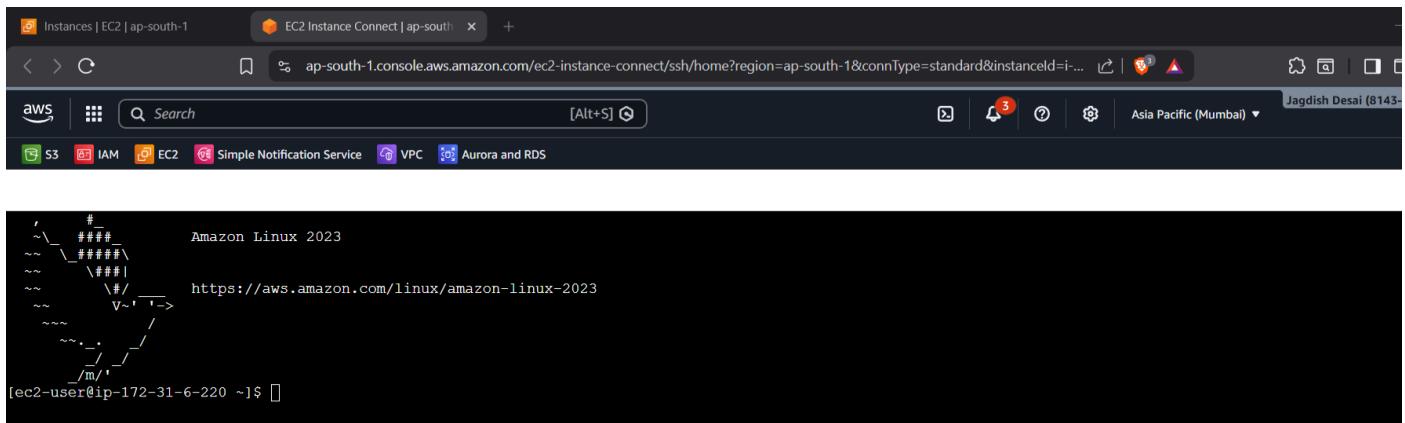
The screenshot shows the AWS EC2 Instances page. A green notification bar at the top indicates "Successfully initiated stopping of i-041888754f68c68ee". Below this, the "Instances (1/1) Info" section displays a single instance: task-7 (i-05779ebb49d30aba6). The instance is listed as "Running" with the instance type "t2.micro" and status "Initializing". The "Details" tab is selected, showing the instance ID, public IPv4 address (13.127.227.160), private IPv4 address (172.31.6.220), and public DNS. The sidebar on the left shows navigation links for EC2, Instances, Images, and Elastic Block Store.

Successfully launched instance.

Step 5: Connect To Ec2 instance

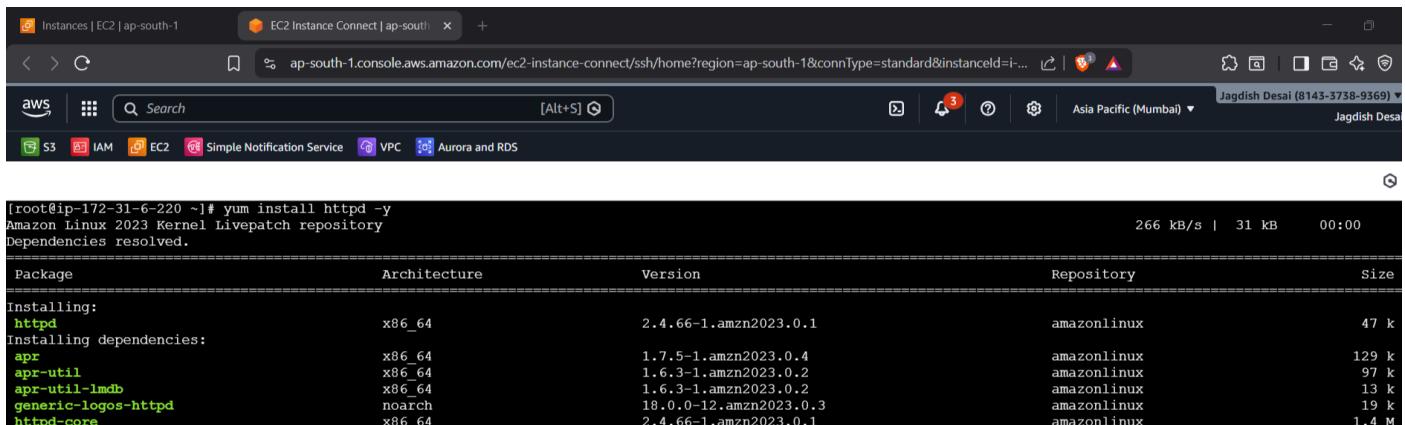
The screenshot shows the "EC2 Instance Connect" page for the instance i-05779ebb49d30aba6. The "Public IPv4 address" field is populated with "13.127.227.160". The "Username" field contains "ec2-user". A note at the bottom states: "Note: In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username." At the bottom right are "Cancel" and "Connect" buttons.

Successfully Connected Instance task-7:



```
'~\ _###_      Amazon Linux 2023
~~ \###\ \
~~ \##|
~~ \#/
~~ V~*-->
~~ .-
~/ \
/m/ \
[ec2-user@ip-172-31-6-220 ~]$
```

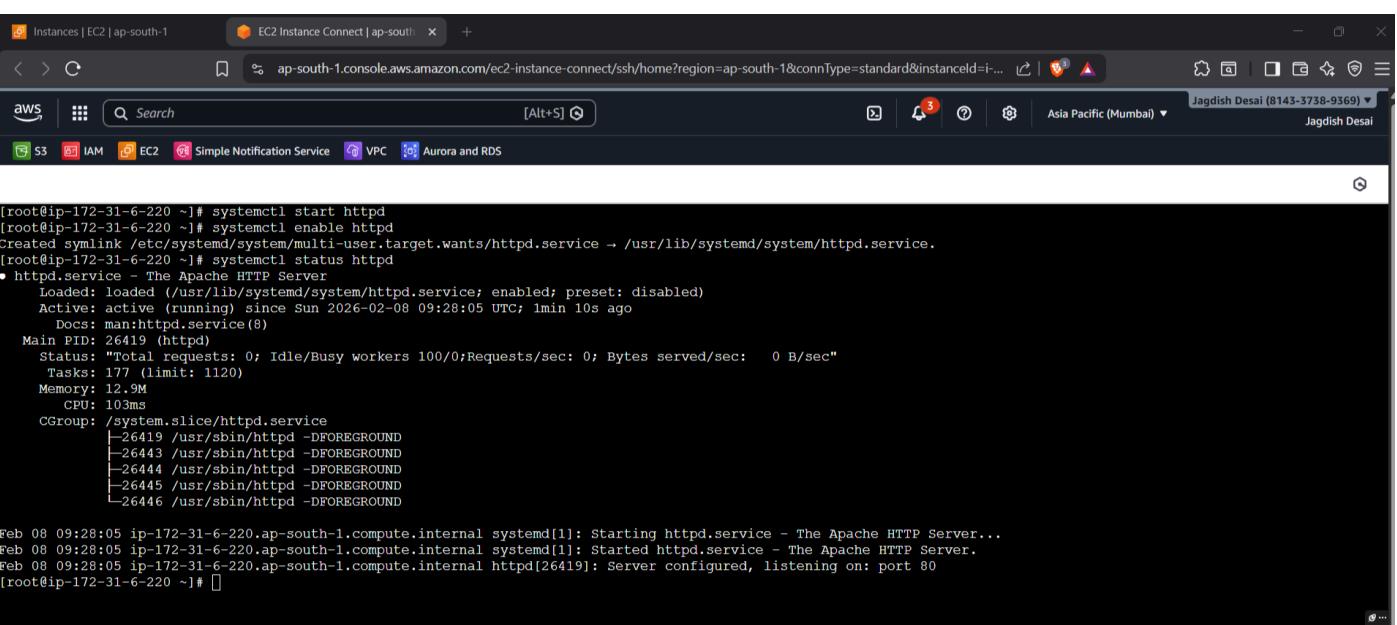
Step 6: We install httpd (Apache HTTP Server) to serve web content over the internet using command- yum install httpd



```
[root@ip-172-31-6-220 ~]# yum install httpd -y
Amazon Linux 2023 Kernel Livepatch repository
Dependencies resolved.
=====
Package           Architecture Version       Repository   Size
=====
Installing:
httpd            x86_64      2.4.66-1.amzn2023.0.1  amazonlinux  47 k
Installing dependencies:
apr              x86_64      1.7.5-1.amzn2023.0.4    amazonlinux  129 k
apr-util         x86_64      1.6.3-1.amzn2023.0.2    amazonlinux  97 k
apr-util-ldap   x86_64      1.6.3-1.amzn2023.0.2    amazonlinux  13 k
generic-logos-httpd noarch      18.0.0-12.amzn2023.0.3  amazonlinux  19 k
httpd-core       x86_64      2.4.66-1.amzn2023.0.1  amazonlinux  1.4 M
=====
266 kB/s | 31 kB     00:00
```

Systemctl start httpd

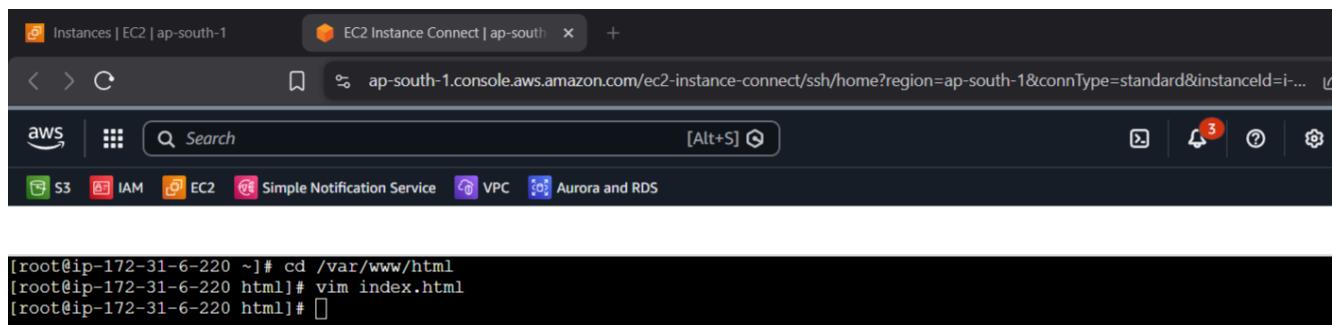
Systemctl enable httpd



```
[root@ip-172-31-6-220 ~]# systemctl start httpd
[root@ip-172-31-6-220 ~]# systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[root@ip-172-31-6-220 ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
   Active: active (running) since Sun 2026-02-08 09:28:05 UTC; 1min 10s ago
     Docs: man:httpd.service(8)
 Main PID: 26419 (httpd)
  Status: "Total requests: 0; Idle/Busy workers 100/0; Requests/sec: 0; Bytes served/sec: 0 B/sec"
   Tasks: 177 (limit: 1120)
  Memory: 12.9M
    CPU: 103ms
   CGroup: /system.slice/httpd.service
           ├─26419 /usr/sbin/httpd -DFOREGROUND
           ├─26443 /usr/sbin/httpd -DFOREGROUND
           ├─26444 /usr/sbin/httpd -DFOREGROUND
           ├─26445 /usr/sbin/httpd -DFOREGROUND
           └─26446 /usr/sbin/httpd -DFOREGROUND
Feb 08 09:28:05 ip-172-31-6-220.ap-south-1.compute.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
Feb 08 09:28:05 ip-172-31-6-220.ap-south-1.compute.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
Feb 08 09:28:05 ip-172-31-6-220.ap-south-1.compute.internal httpd[26419]: Server configured, listening on: port 80
[root@ip-172-31-6-220 ~]#
```

Step 7:Typing our code EC2 web server directory:

- For **httpd**: cd /var/www/html/index.html



```
[root@ip-172-31-6-220 ~]# cd /var/www/html
[root@ip-172-31-6-220 html]# vim index.html
[root@ip-172-31-6-220 html]# 
```

Index.html Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Public IP Viewer</title>
<style>
body {
    margin: 0;
    height: 100vh;
    font-family: 'Segoe UI', Tahoma, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    background: linear-gradient(135deg, #667eea, #764ba2);
    color: white;
}
.card {
    background: rgba(255, 255, 255, 0.15);
    padding: 40px 60px;
    border-radius: 15px;
    text-align: center;
    box-shadow: 0 10px 25px rgba(0,0,0,0.3);
    backdrop-filter: blur(8px);
```

```
}

h1 {
    margin-bottom: 15px;
    font-size: 32px;
}

.ip {
    font-size: 22px;
    font-weight: bold;
    color: #ffeb3b;
    margin-top: 10px;
}

footer {
    margin-top: 20px;
    font-size: 14px;
    opacity: 0.8;
}

</style>
</head>
<body>
<div class="card">

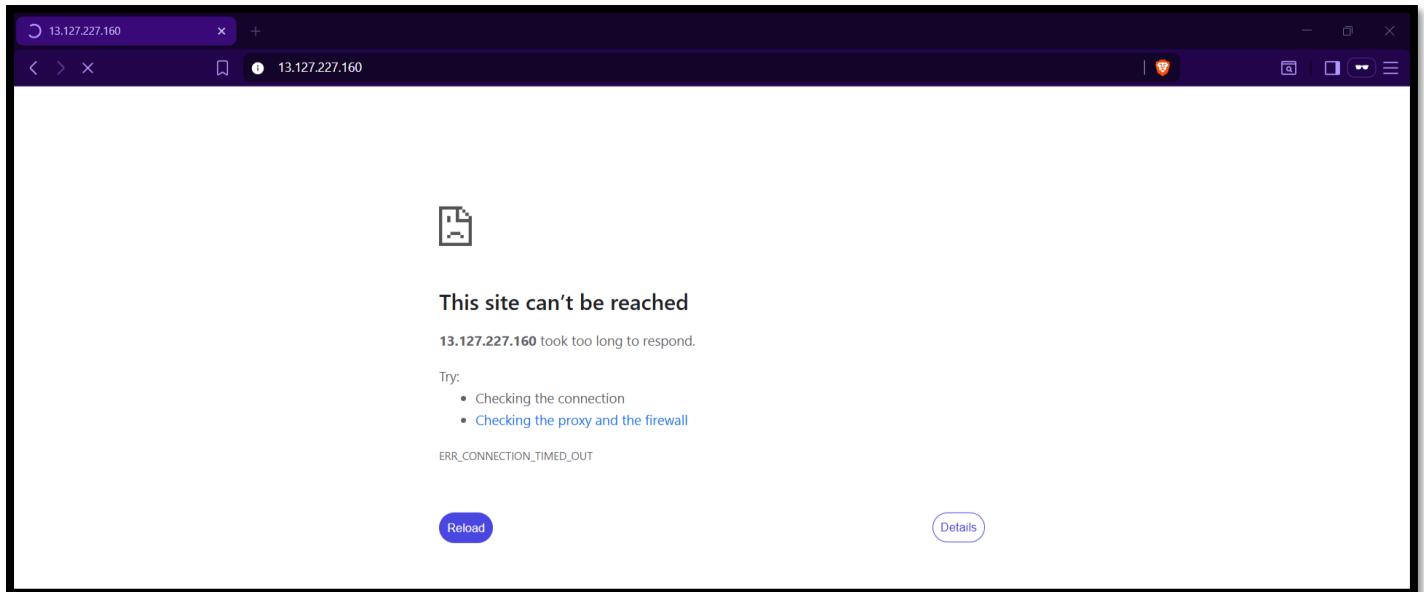
    <h1>Jagdish Desai EC2 Application</h1>
    <div>Public IP Address</div>
    <div class="ip" id="publicIp">Loading...</div>
    <footer>Running on AWS EC2 Instance</footer>
</div>
<script>
fetch('http://169.254.169.254/latest/meta-data/public-ipv4')
    .then(response => response.text())
    .then(ip => document.getElementById('publicIp').innerText = ip)
    .catch(() => document.getElementById('publicIp').innerText = 'Cannot fetch IP');
</script>
</body>
</html>
```

After That Code Copy public Ip pf our instance and check application is working or not.

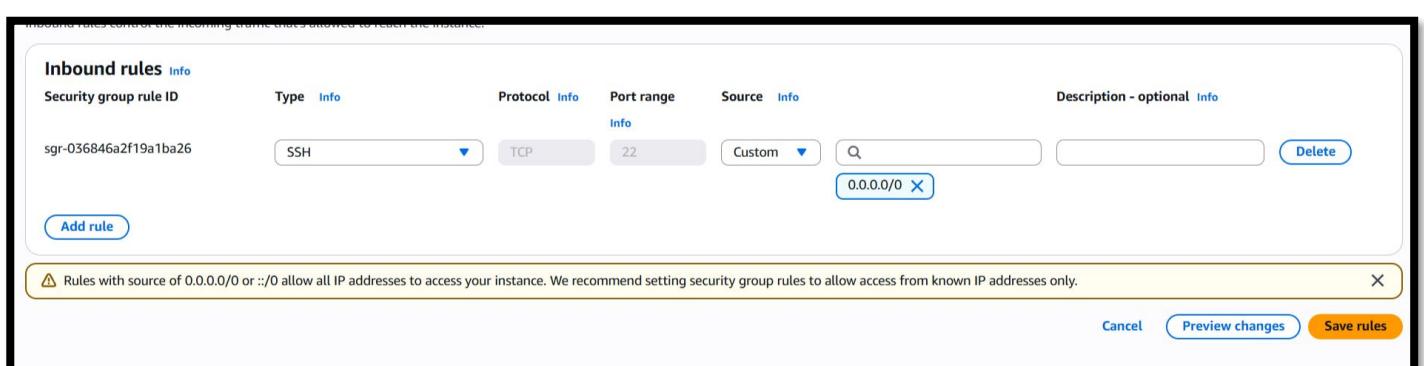
Public IPv4 address

13.127.227.160 |

Application not accessible because we doesn't allow http port in security group inbound rules.



Then we go to security groups and add http in inbound rule.



No http rule present.

Then click add rule

The screenshot shows the 'Inbound rules' section of the AWS Security Groups configuration. It lists two rules:

- Rule 1: Type: SSH, Protocol: TCP, Port range: 22, Source: Custom (0.0.0.0/0), Description: optional.
- Rule 2: Type: HTTP, Protocol: TCP, Port range: 80, Source: Anywhere (0.0.0.0/0), Description: optional.

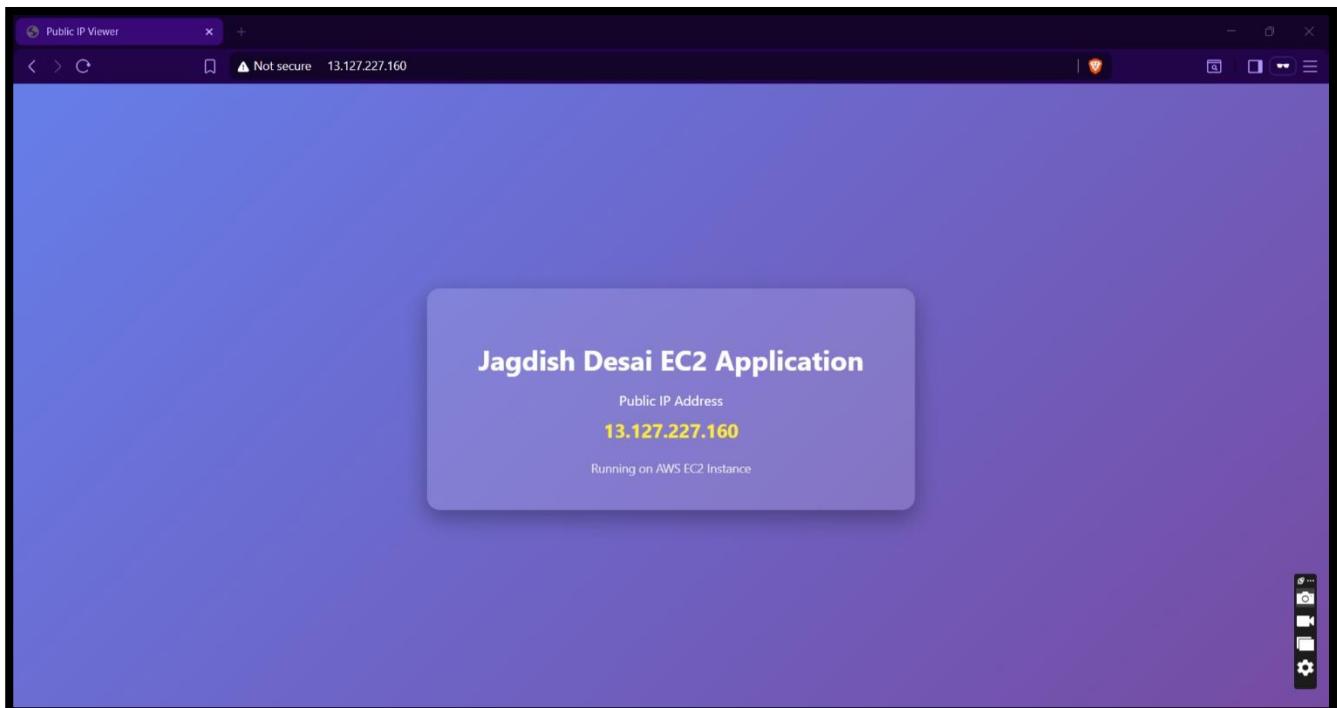
A button labeled 'Add rule' is visible at the bottom left. A warning message at the bottom states: '⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.' Buttons for 'Cancel', 'Preview changes', and 'Save rules' are at the bottom right.

Save rules

- Copy public Ip pf our instance and check application is working or not.

Public IPv4 address

13.127.227.160 |



After Allow Http Port in security group inbound rules it properly works.

2. Container Running but Port Not Reachable

Step 1: Install Docker

```
sudo amazon-linux-extras install docker -y
```

Start Docker service:

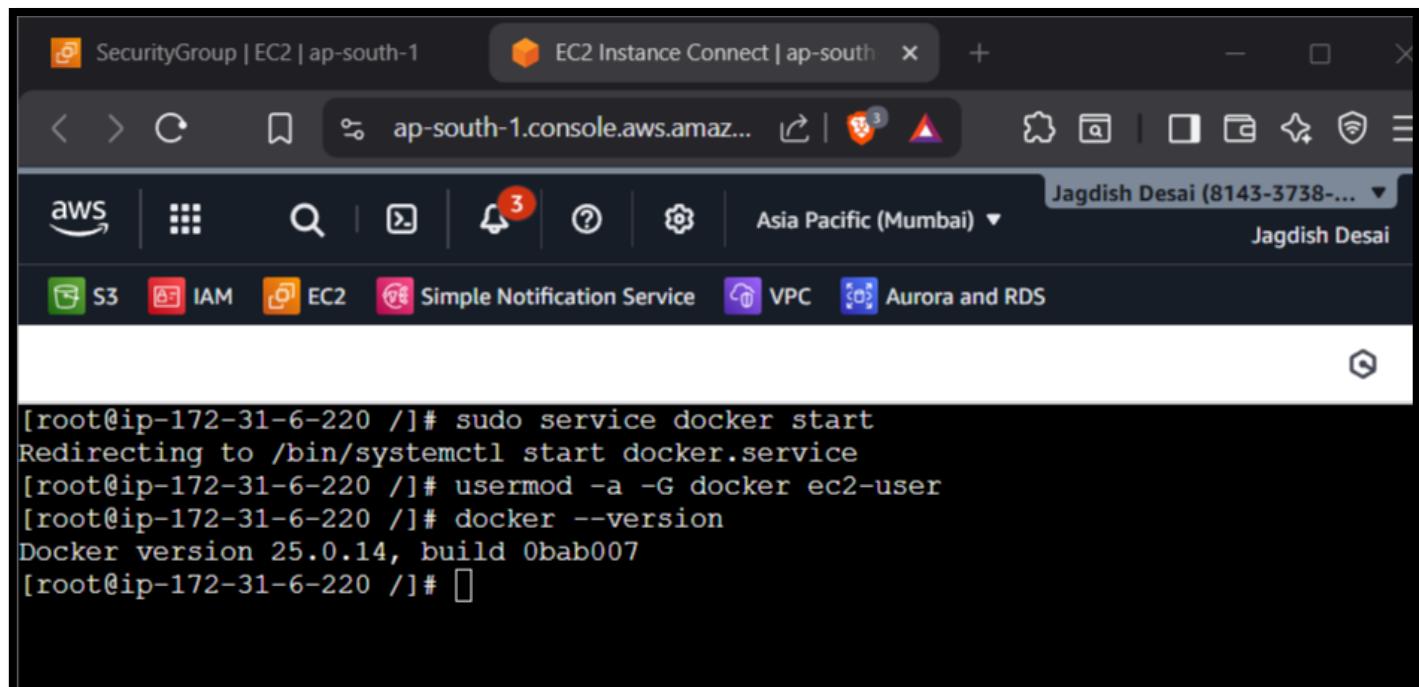
```
sudo service docker start
```

Allow EC2 user to run Docker without sudo (optional):

```
sudo usermod -a -G docker ec2-user
```

Test Docker:

```
docker --version
```

A screenshot of the AWS Management Console. The top navigation bar shows 'SecurityGroup | EC2 | ap-south-1' and 'EC2 Instance Connect | ap-south-1'. The main dashboard has sections for S3, IAM, EC2, Simple Notification Service, VPC, and Aurora and RDS. A user profile 'Jagdish Desai (8143-3738-...)' is visible on the right. Below the header is a toolbar with various icons. The main content area is a terminal window showing the following command history:

```
[root@ip-172-31-6-220 /]# sudo service docker start
Redirecting to /bin/systemctl start docker.service
[root@ip-172-31-6-220 /]# usermod -a -G docker ec2-user
[root@ip-172-31-6-220 /]# docker --version
Docker version 25.0.14, build 0bab007
[root@ip-172-31-6-220 /]#
```

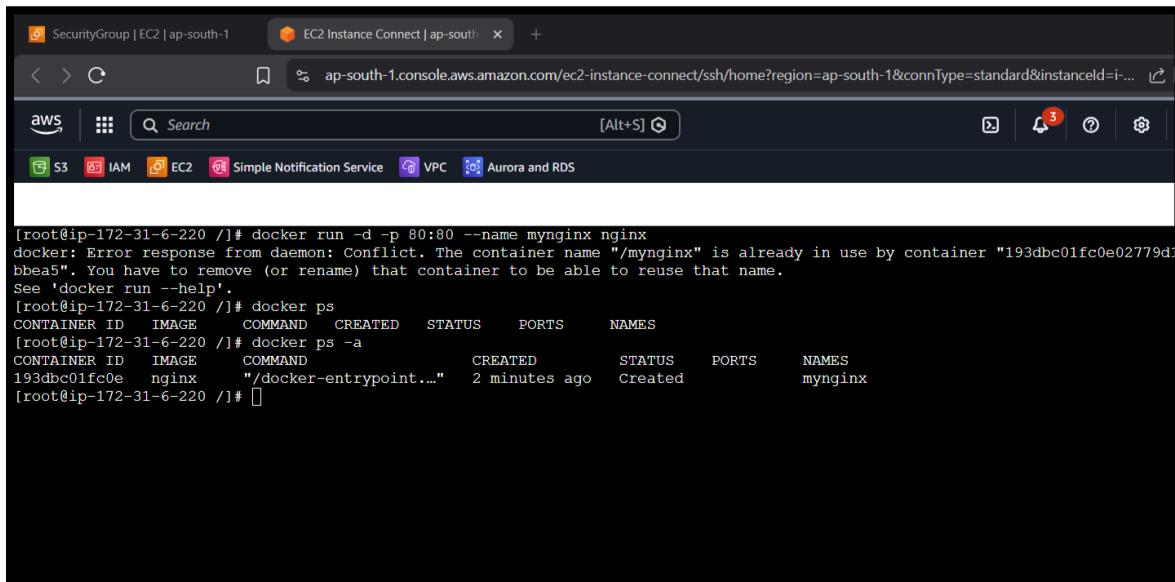
Step 2: Run a Docker Container

Example: Nginx web server on port 80

```
docker run -d -p 80:80 --name mynginx nginx
```

Check container status:

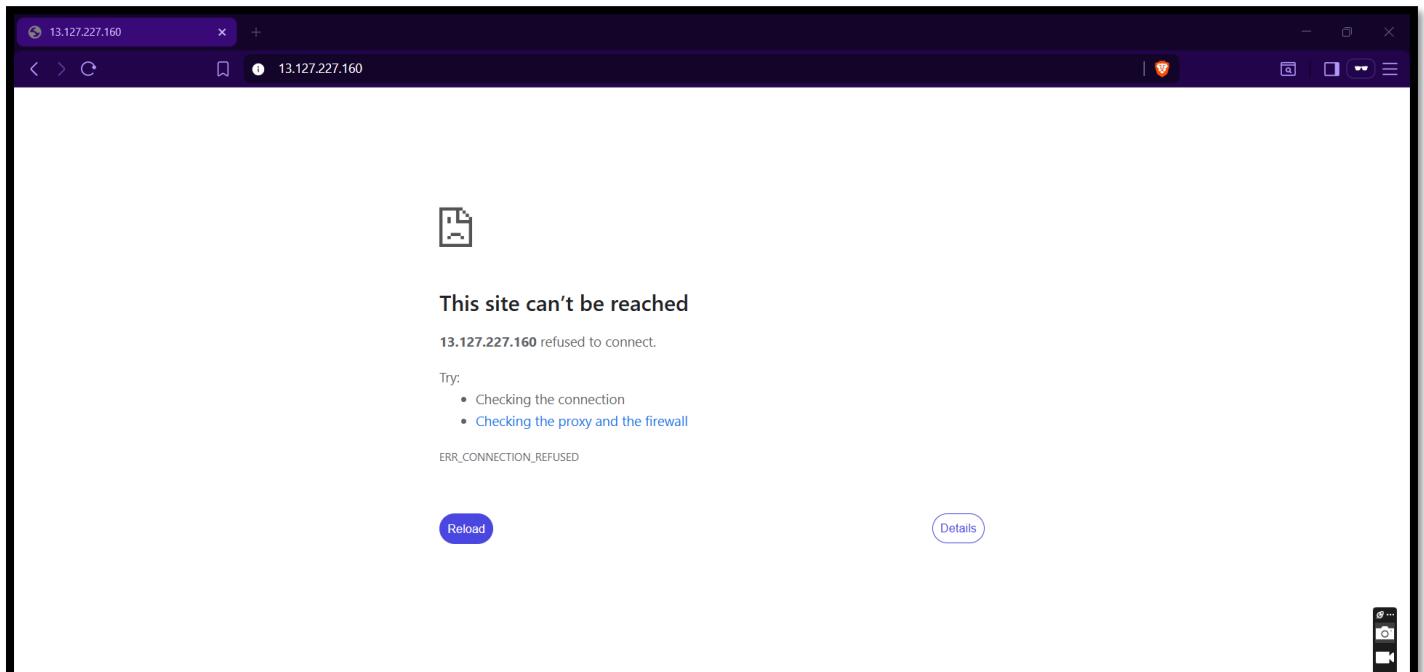
```
docker ps
```



The screenshot shows a terminal window within an AWS EC2 Instance Connect session. The user has run several commands to manage a Docker container named 'mynginx'. The first command, `docker run -d -p 80:80 --name mynginx nginx`, failed because the container name was already in use by another container. The user then checked the status of all containers with `docker ps -a` and found the existing container 'mynginx' running.

```
[root@ip-172-31-6-220 /]# docker run -d -p 80:80 --name mynginx nginx
docker: Error response from daemon: Conflict. The container name "/mynginx" is already in use by container "193dbc01fc0e02779dbbea5". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
[root@ip-172-31-6-220 /]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
[root@ip-172-31-6-220 /]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
193dbc01fc0e        nginx              "/docker-entrypoint..."   2 minutes ago    Created           mynginx
[root@ip-172-31-6-220 /]#
```

Step 3: Try Accessing From Browser



Step 7: Troubleshooting “Port Not Reachable”

1. Checking Security Group

- Port 80 must be open for **0.0.0.0/0**.

The screenshot shows the AWS Management Console interface for managing security groups. The user is on the 'Edit inbound rules' page for a specific security group. There are two rules listed:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0f555dbe1080a573d	HTTP	TCP	80	Custom	0.0.0.0/0
sgr-036846a2f19a1ba26	SSH	TCP	22	Custom	0.0.0.0/0

A yellow warning box at the bottom states: "⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only." Below the warning are buttons for 'Cancel', 'Preview changes', and 'Save rules'.

remove container and rerun with:

```
docker rm -f mynginx
```

```
docker run -d -p 80:80 --name mynginx nginx
```

Check EC2 Security Group

- HTTP port 80 → 0.0.0.0/0

Test locally on EC2

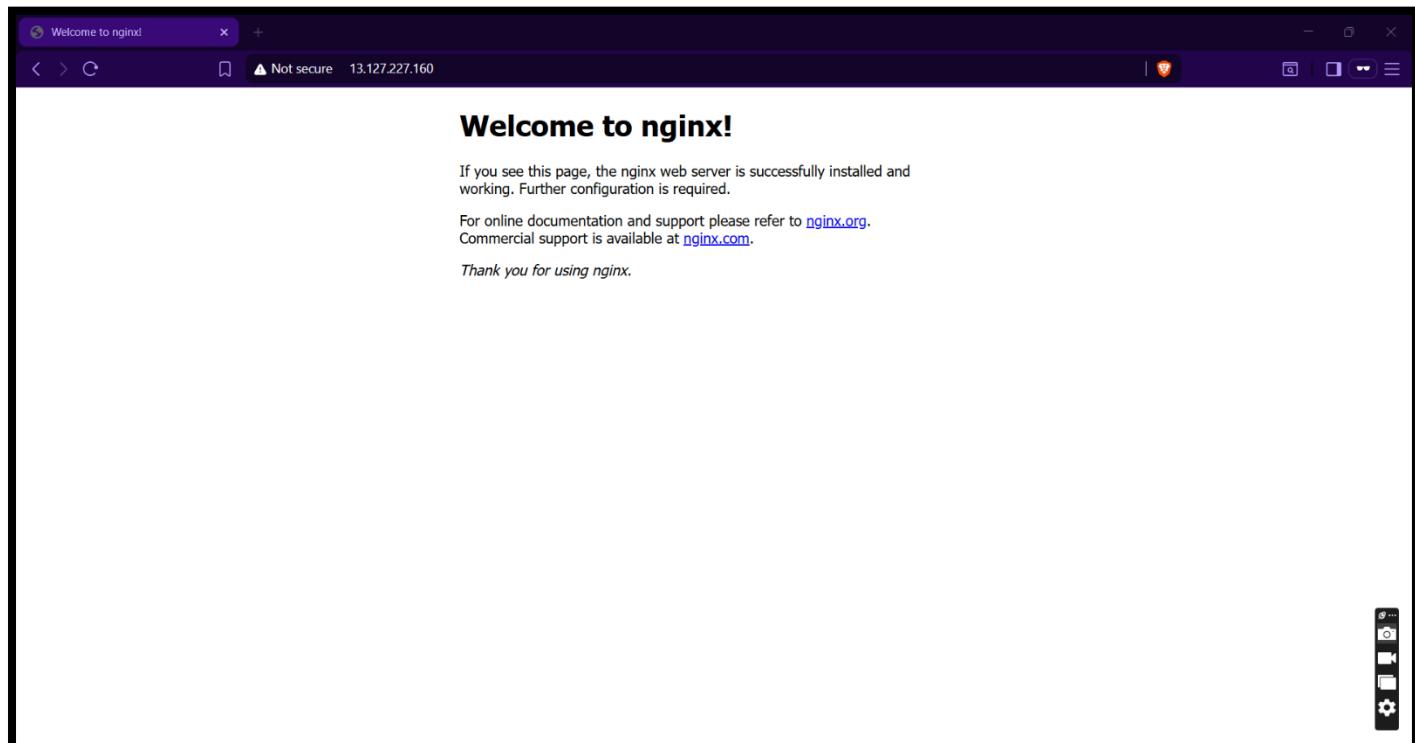
```
curl localhost:80
```

```
[root@ip-172-31-6-220 ~]# docker rm -f mynginx
mynginx
[root@ip-172-31-6-220 ~]# docker run -d -p 80:80 --name mynginx nginx
c21a8c5292271af03bee9a7819adec679ce8b9fledda257c65fcddb9dde1d674
[root@ip-172-31-6-220 ~]# curl localhost:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
<p>For online documentation and support please refer to
<a href="http://nginx.org">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
[root@ip-172-31-6-220 ~]# 
```

CloudShell Feedback Console Mobile App © 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

4. Test externally from browser

<http://13.127.227.160/>



It works Properly.

Troubleshooting ALB health check failures

Step 1: using our existing Ec2 instance

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with sections like EC2, Instances, Images, and Elastic Block Store. The main area displays a table titled 'Instances (1/1) Info' with one row. The row details are:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
task-7	i-05779ebb49d30aba6	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1b	ec2-13-1...

Below the table, there's a detailed view for the instance i-05779ebb49d30aba6 (task-7). It shows tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. Under the Details tab, you can see the Instance ID (i-05779ebb49d30aba6), Public IPv4 address (13.127.227.160), Private IPv4 addresses (172.31.6.220), Instance state (Running), and Public DNS.

Step 2: Verified our existing nginx app is running:

```
curl http://localhost:80
```

The screenshot shows an SSH session on an EC2 instance. The terminal window displays the following command and its output:

```
[ec2-user@ip-172-31-6-220 ~]$ sudo -i
[root@ip-172-31-6-220 ~]# curl http://localhost:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
<p>For online documentation and support please refer to
<a href="http://nginx.org/">http://nginx.org/.<br/>
Commercial support is available at
<a href="http://nginx.com/">http://nginx.com/.</p>
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
[root@ip-172-31-6-220 ~]# ]
```

At the bottom of the terminal, it says "i-05779ebb49d30aba6 (task-7)". Below the terminal, there's a summary bar showing PublicIPs: 13.127.227.160 and PrivateIPs: 172.31.6.220.

Step 2: Created Target Group

Step 1
Create target group
Step 2 - recommended
Register targets
Step 3
Review and create

Create target group

A target group can be made up of one or more targets. Your load balancer routes requests to the targets in a target group and performs health checks on the targets.

Settings - immutable

Choose a target type and the load balancer and listener will route traffic to your target. These settings can't be modified after target group creation.*

Target type

Indicate what resource type you want to target. Only the selected resource type can be registered to this target group.

Instances

Supports load balancing to instances in a VPC. Integrate with Auto Scaling Groups or ECS services for automatic management.

Suitable for: **ALB** **NLB** **GWLB**

IP addresses

Supports load balancing to VPC and on-premises resources. Facilitates routing to IP addresses and network interfaces on the same instance. Supports IPv6 targets.

Suitable for: **ALB** **NLB** **GWLB**

Lambda function

Supports load balancing to a single Lambda function. ALB required as traffic source.

Suitable for: **ALB**

Application Load Balancer

Allows use of static IP addresses and PrivateLink with an Application Load Balancer. NLB required as traffic source.

Suitable for: **NLB**

Target group name

Name must be unique per Region per AWS account.

my-tg

Accepts: a-z, A-Z, 0-9, and hyphen (-). Can't begin or end with hyphen. 1-32 total characters; Count: 5/32

Register Targets:

Step 1
Create target group
Step 2 - recommended
Register targets
Step 3
Review and create

Register targets - recommended

This is an optional step to create a target group. However, to ensure that your load balancer routes traffic to this target group you must register your targets.

Available instances (1/1)

Instance ID	Name	State	Security groups	Zone
i-05779ebb49d30aba6	task-7	Running	launch-wizard-10	ap-south-1b

1 selected

Ports for the selected instances

Ports for routing traffic to the selected instances.

80

1-65535 (separate multiple ports with commas)

Include as pending below

Click Create Target Group.

Step 4: Create Application Load Balancer (ALB)

The screenshot shows the AWS CloudFront console with the URL ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#SelectCreateELBWizard. The page is titled "Compare and select load balancer type".

Application Load Balancer

Choose an Application Load Balancer when you need a flexible feature set for your applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers.

[Create](#)

Network Load Balancer

Choose a Network Load Balancer when you need ultra-high performance, TLS offloading at scale, centralized certificate deployment, support for UDP, and static IP addresses for your applications. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second securely while maintaining ultra-low latencies.

[Create](#)

Gateway Load Balancer

Choose a Gateway Load Balancer when you need to deploy and manage a fleet of third-party virtual appliances that support GENEVE. These appliances enable you to improve security, compliance, and policy controls.

[Create](#)

select **internet-facing**, IPv4, and subnets

Create application load balancer | ModifyInboundSecurityGroupRules | EC2 Instance Connect | ap-south-1 | +

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#CreateALBWizard:

aws [Alt+S] Ask Amazon Q

S3 IAM EC2 Simple Notification Service VPC Aurora and RDS

Jagdish Desai (8143-3758-9369) Asia Pacific (Mumbai) Jagdish Desai

EC2 > Load balancers > Create Application Load Balancer

Create Application Load Balancer Info

The Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets such as Amazon EC2 instances, microservices, and containers, based on request attributes. When the load balancer receives a connection request, it evaluates the listener rules in priority order to determine which rule to apply, and if applicable, it selects a target from the target group for the rule action.

▶ How Application Load Balancers work

Basic configuration

Load balancer name
Name must be unique within your AWS account and can't be changed after the load balancer is created.
 A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Scheme Info
Scheme can't be changed after the load balancer is created.

Internet-facing
• Services internet-facing traffic.
• Has public IP addresses.
• DNS name resolves to public IPs.
• Requires a public subnet.

Internal
• Services internal traffic.
• Has private IP addresses.
• DNS name resolves to private IPs.
• Compatible with the IPv4 and Dualstack IP address types.

Load balancer IP address type Info
Select the front-end IP address type to assign to the load balancer. The VPC and subnets mapped to this load balancer must include the selected IP address types. Public IPv4 addresses have an additional cost.

IPv4
Includes only IPv4 addresses.

Dualstack
Includes IPv4 and IPv6 addresses.

Dualstack without public IPv4

CloudShell Feedback F1 Console Mobile App

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Configure Security Group: allow HTTP (80)

Availability Zones and subnets [Info](#)
Select at least two Availability Zones and a subnet for each zone. A load balancer node will be placed in each selected zone and will automatically scale in response to traffic. The load balancer routes traffic to targets in the selected Availability Zones only.

ap-south-1a (az1)
Subnet
Only CIDR blocks corresponding to the load balancer IP address type are used. At least 8 available IP addresses are required for your load balancer to scale efficiently.
subnet-04cff1c02b7d4f454
IPv4 subnet CIDR: 172.31.32.0/20

ap-south-1b (az3)
Subnet
Only CIDR blocks corresponding to the load balancer IP address type are used. At least 8 available IP addresses are required for your load balancer to scale efficiently.
subnet-087f57a4647fd624
IPv4 subnet CIDR: 172.31.0.0/20

ap-south-1c (az2)
Subnet
Only CIDR blocks corresponding to the load balancer IP address type are used. At least 8 available IP addresses are required for your load balancer to scale efficiently.
subnet-0b794ec7dbb17441d
IPv4 subnet CIDR: 172.31.16.0/20

Security groups [Info](#)
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](#).

Security groups
Select up to 5 security groups

default
sg-0a219884179efdf6a4 VPC: vpc-052859de1479c19a9

Click on Create Load Balancer.

Step 5: Checking Health Status

1. Go to Target Group → Targets tab
2. The Status column shows:

Target groups (1/1) [Info](#) | [What's new?](#)

Actions [Create target group](#)

Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
my-tg	arn:aws:elasticloadbalancin...	80	HTTP	Instance	None associated	vpc-052859de1479c19a9

Target group: my-tg

Registered targets (1/1) [Info](#)

Anomaly mitigation: Not applicable

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

Instance ID	Name	Port	Zone	Health status	Health status details	Admini...	Overri...	Launch...
i-05779ebb49d30aba6	task-7	80	ap-south-1b (a...	Healthy	-	-	-	February ...

- **Healthy** → working fine
- **Unhealthy** → health check is failing

After Removing Http inbound rule:

The screenshot shows the AWS CloudWatch Metrics Insights interface. At the top, there's a navigation bar with tabs for 'CloudWatch Metrics' (selected), 'Logs', 'Metrics Insights', and 'Metrics Analytics'. Below the navigation is a search bar with the placeholder 'Search metrics' and a dropdown menu for 'Metric names'. The main content area displays a table of metrics with columns: Metric name, Metric namespace, Metric type, and Last value. One row is highlighted in blue, showing 'aws.ec2.instance.status' with 'Value' as the metric type and '1' as the last value. At the bottom right, there's a 'Create new metric' button.

Unhealthy

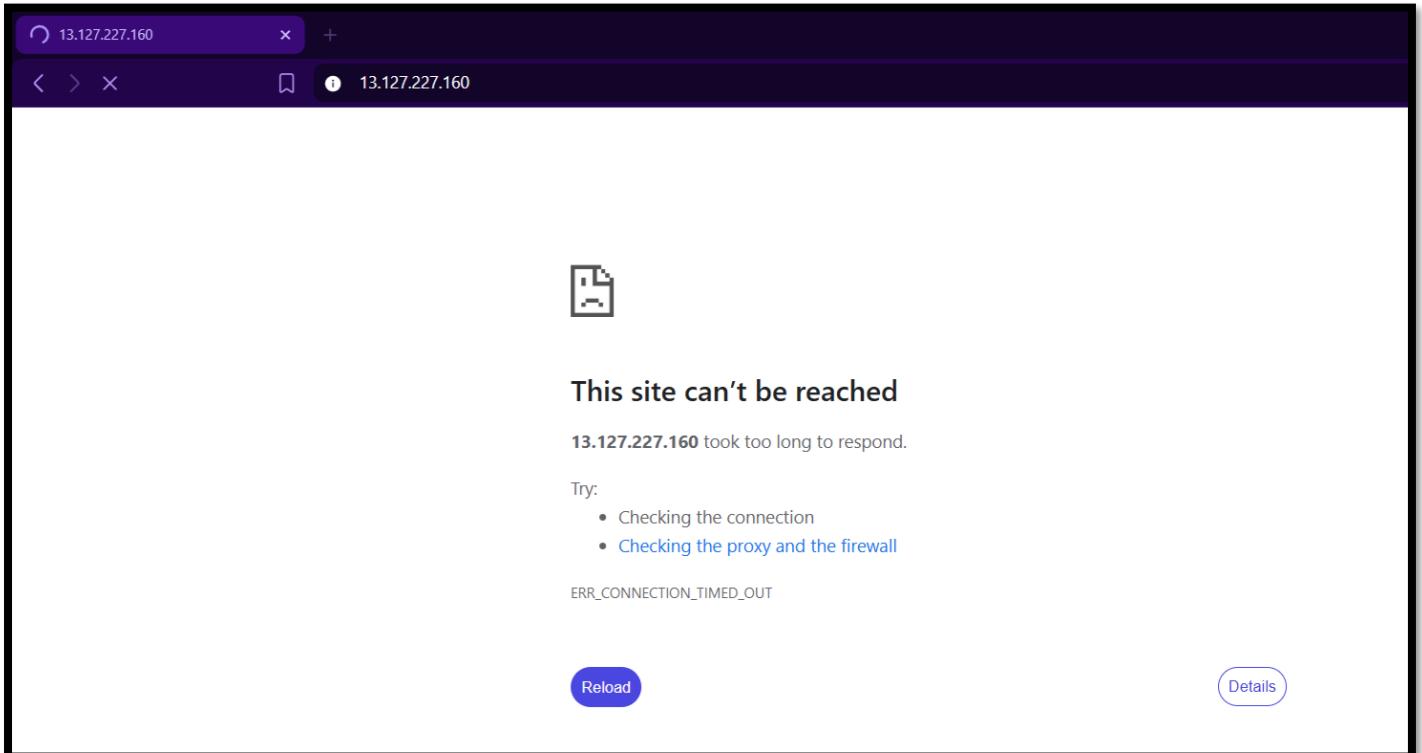
Command:

Curl <http://localhost:80/>

This accesses port 80 on your EC2 host locally.

After Target will become **healthy**, and traffic will be routed correctly.

Not working using public ip after removing http rule-



Allow Http Port in inbound rules

A screenshot of the AWS Management Console showing the "Edit inbound rules" page for a security group. The top navigation bar includes "EC2", "Security Groups", "sg-03f4d494b718e1436 - launch-wizard-10", and "Edit inbound rules".

Edit inbound rules Info

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

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-036846a2f19a1ba26	SSH	TCP	22	Custom	0.0.0.0/0
-	HTTP	TCP	80	Anywhere	0.0.0.0/0

Add rule

⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Preview changes Save rules

Save Rules

Again Go To Your Target Group Targets:

The screenshot shows the AWS CloudWatch Metrics console with the 'Target groups' page. At the top, there is a search bar labeled 'Filter target groups' and a table header with columns: Name, ARN, Port, Protocol, Target type, Load balancer, and VPC ID. A single row is selected, showing 'my-tg' as the name, its ARN, port 80, HTTP protocol, Instance target type, alb load balancer, and vpc-052859de1479c19a9 VPC ID. Below the table, a section titled 'Target group: my-tg' is shown with tabs for Details, Targets (which is selected), Monitoring, Health checks, Attributes, and Tags. Under the Targets tab, it says 'Registered targets (1)'. A table lists one target: 'i-05779ebb49d30aba6' with the name 'task-7', port 80, zone 'ap-south-1b (a...)', health status 'Healthy', and a note 'No override.' The status is also highlighted in green. At the bottom of the page, there are links for 'Console Mobile App', '© 2026, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

Now Showing Health Status Healthy. After allow 80 port.

Trying to access browser using public ip:
<http://13.127.227.160/>

The screenshot shows a web browser window with the address bar displaying 'Not secure 13.127.227.160'. The main content of the page is 'Welcome to nginx!'. The text on the page reads: 'If you see this page, the nginx web server is successfully installed and working. Further configuration is required.' followed by 'For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com. Thank you for using nginx.' The browser interface includes standard navigation buttons (back, forward, search) and a toolbar with various icons.

It Works Properly.