

# DOCKER NETWORKING

Docker networking is the system that enables communication between Docker containers, the host machine, and external networks (e.g., the internet). It provides the connectivity and isolation required for containerized applications to function effectively.

## Types of Docker Networks

### 1. Bridge Network (Default)

- Used by default when a container is created without specifying a network.
- Containers on the same bridge network can communicate with each other using IP addresses or container names.
- Ideal for standalone applications on a single host.

### 2. Host Network

- The container shares the host's network namespace.
- There is no network isolation between the host and the container.
- Used for performance-sensitive applications or when direct host networking is needed.

### 3. None Network

- The container is not attached to any network.
- Used for highly isolated containers.

## Created a EC2 instance and launch instance

The screenshot displays the AWS Management Console interface. The top navigation bar shows the user is logged in as 'Kavitha' in the 'Mumbai' region. The main content area is titled 'Instances (1/1) Info' and shows a table with one instance: 'Docker Networking' (Instance ID: i-0945d6dcca3da38fd). The instance is in a 'Running' state, using a 't2.micro' instance type, and its status is 'Initializing'. The console also shows the instance's public IPv4 address (13.201.83.91) and private IPv4 addresses (172.31.2.98). The left sidebar contains navigation links for 'Dashboard', 'EC2 Global View', 'Events', 'Instances', 'Instance Types', 'Launch Templates', 'Spot Requests', 'Savings Plans', 'Reserved Instances', 'Dedicated Hosts', 'Capacity Reservations', 'Images', 'AMI Catalog', and 'Elastic Block Store'. The bottom of the screen shows the Windows taskbar with the date and time as 11:01 PM on 21-12-2024.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
Docker Networking	i-0945d6dcca3da38fd	Running	t2.micro	Initializing	View alarms +	ap-south-11

**i-0945d6dcca3da38fd (Docker Networking)**

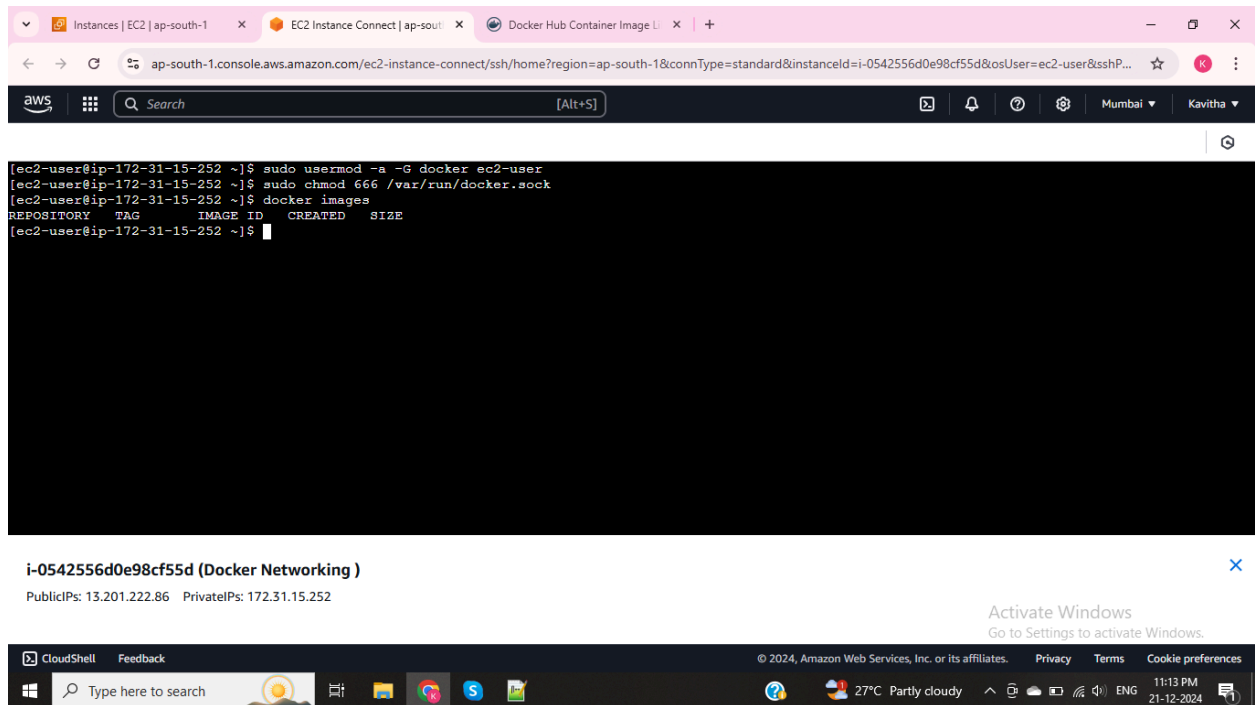
**Details** | Status and alarms | Monitoring | Security | Networking | Storage | Tags

**Instance summary**

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0945d6dcca3da38fd	13.201.83.91   open address	172.31.2.98

**IPV6 address** | **Instance state** | **Public IPv4 DNS**

## Install docker and added docker with user and give permission to access

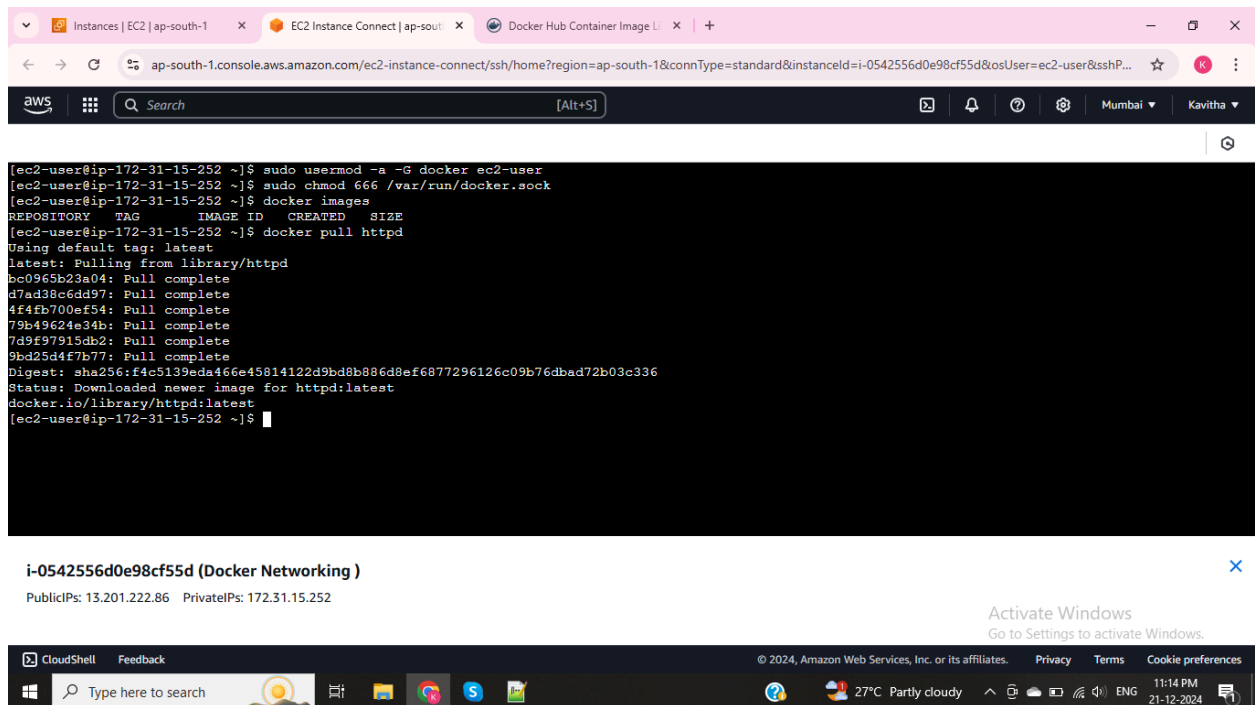


The screenshot shows the AWS CloudShell interface with a terminal window. The terminal output is as follows:

```
[ec2-user@ip-172-31-15-252 ~]$ sudo usermod -a -G docker ec2-user
[ec2-user@ip-172-31-15-252 ~]$ sudo chmod 666 /var/run/docker.sock
[ec2-user@ip-172-31-15-252 ~]$ docker images
REPOSITORY    TAG       IMAGE ID   CREATED   SIZE
[ec2-user@ip-172-31-15-252 ~]$
```

Below the terminal window, the instance ID **i-0542556d0e98cf55d (Docker Networking)** is displayed, along with PublicIPs: 13.201.222.86 and PrivateIPs: 172.31.15.252. An "Activate Windows" watermark is visible in the bottom right corner.

## From Docker hub pull httpd image



The screenshot shows the AWS CloudShell interface with a terminal window. The terminal output is as follows:

```
[ec2-user@ip-172-31-15-252 ~]$ sudo usermod -a -G docker ec2-user
[ec2-user@ip-172-31-15-252 ~]$ sudo chmod 666 /var/run/docker.sock
[ec2-user@ip-172-31-15-252 ~]$ docker images
REPOSITORY    TAG       IMAGE ID   CREATED   SIZE
[ec2-user@ip-172-31-15-252 ~]$ docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
bc0965b23a04: Pull complete
d7ad38c6dd97: Pull complete
4f4fb700ef54: Pull complete
79b49624e34b: Pull complete
7d9f97915db2: Pull complete
9bd25d4f7b77: Pull complete
Digest: sha256:f4c5139eda466e45814122d9bd8b886d8ef6877296126c09b76dbad72b03c336
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
[ec2-user@ip-172-31-15-252 ~]$
```

Below the terminal window, the instance ID **i-0542556d0e98cf55d (Docker Networking)** is displayed, along with PublicIPs: 13.201.222.86 and PrivateIPs: 172.31.15.252. An "Activate Windows" watermark is visible in the bottom right corner.

## Created a containers from image using bridge networking with particular port numbers

```
[ec2-user@ip-172-31-15-252 ~]$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
httpd         latest    494b2b45fd74   5 months ago   147MB
[ec2-user@ip-172-31-15-252 ~]$ docker run -itd --name sri --net bridge -p 8081:80 httpd
1dcd73be2e90de8520a70be7ae20e12aa316f1cfdb62733f25e74afc482bacb
[ec2-user@ip-172-31-15-252 ~]$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
1dcd73be2e90   httpd     "httpd-foreground"      6 seconds ago Up 5 seconds   0.0.0.0:8081->80/tcp, :::8081->80/tcp   sri
14f39017f39c   httpd     "httpd-foreground"      2 minutes ago Up 2 minutes   0.0.0.0:8080->80/tcp, :::8080->80/tcp   kavi
[ec2-user@ip-172-31-15-252 ~]$
```

i-0542556d0e98cf55d (Docker Networking)

PublicIPs: 13.201.222.86 PrivateIPs: 172.31.15.252

Activate Windows  
Go to Settings to activate Windows.

## In the security groups edited the Inbound rules and allow port numbers 8080 and 8081 to access the application

ModifyInboundSecurityGroupR... EC2 Instance Connect | ap-souti... Docker Hub Container Image L... +

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#ModifyInboundSecurityGroupRules?securityGroupId=sg-0ed5419ac28b0bd40

EC2 > Security Groups > sg-0ed5419ac28b0bd40 - launch-wizard-3 > Edit inbound rules

### Edit inbound rules [Info](#)

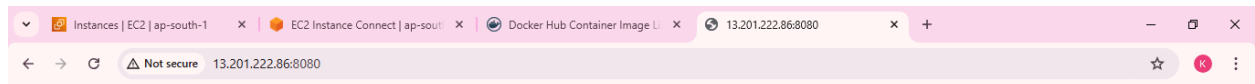
Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type <a href="#">Info</a>	Protocol <a href="#">Info</a>	Port range <a href="#">Info</a>	Source <a href="#">Info</a>	Description - optional <a href="#">Info</a>	
sg-01f7e342ff8caa2c8	SSH	TCP	22	Cu...	<input type="text"/>	<a href="#">Delete</a>
-	Custom TCP	TCP	8080	An...	<input type="text"/> 0.0.0.0/0	<a href="#">Delete</a>
-	Custom TCP	TCP	8081	An...	<input type="text"/> 0.0.0.0/0	<a href="#">Delete</a>

[Add rule](#)

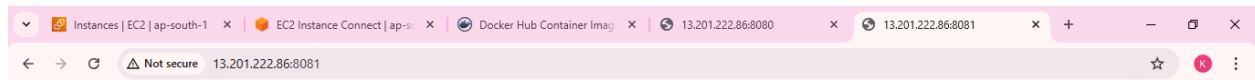
Activate Windows  
Go to Settings to activate Windows.

**In Bridge networking Public ip address is same but applications are run in different port numbers - 8080 port output**



**It works!**

**Output Port number 8081**



**It works!**

## Docker inspect net bridge - Applications are run in different ip address and different port in internal output

The screenshot shows a terminal window with the following commands and output:

```
[ec2-user@ip-172-31-15-252 ~]$ docker run -itd --name sri --net bridge -p 8081:80 httpd
1dc473be2e90de8520a70bc7ae20e12ae316f1cfd62733f25e74afc482bacb
[ec2-user@ip-172-31-15-252 ~]$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
1dc473be2e90	httpd	"httpd-foreground"	6 seconds ago	Up 5 seconds	0.0.0.0:8081->80/tcp, :::8081->80/tcp	sri
14f39017f39c	httpd	"httpd-foreground"	2 minutes ago	Up 2 minutes	0.0.0.0:8080->80/tcp, :::8080->80/tcp	kavi

```
[ec2-user@ip-172-31-15-252 ~]$ docker inspect net bridge
[
  {
    "Name": "bridge",
    "Id": "0d81cc66809633981e8ad3a05d0a2a79a7e6ee00a851357fed1f4beffd99195f",
    "Created": "2024-12-21T17:40:06.285714094Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "172.17.0.0/16",
          "Gateway": "172.17.0.1"
        }
      ]
    }
  }
]
```

**i-0542556d0e98cf55d (Docker Networking)**

PublicIPs: 13.201.222.86 PrivateIPs: 172.31.15.252

Activate Windows  
Go to Settings to activate Windows.

## Release the particular ports and deleted the containers

The screenshot shows a terminal window with the following commands and output:

```
[ec2-user@ip-172-31-15-252 ~]$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
nginx latest 66f8bdd3810c 3 weeks ago 192MB
httpd latest 494b2b45fd74 5 months ago 147MB
[ec2-user@ip-172-31-15-252 ~]$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
1dc473be2e90	httpd	"httpd-foreground"	5 minutes ago	Up 5 minutes	0.0.0.0:8081->80/tcp, :::8081->80/tcp	sri
14f39017f39c	httpd	"httpd-foreground"	8 minutes ago	Up 8 minutes	0.0.0.0:8080->80/tcp, :::8080->80/tcp	kavi

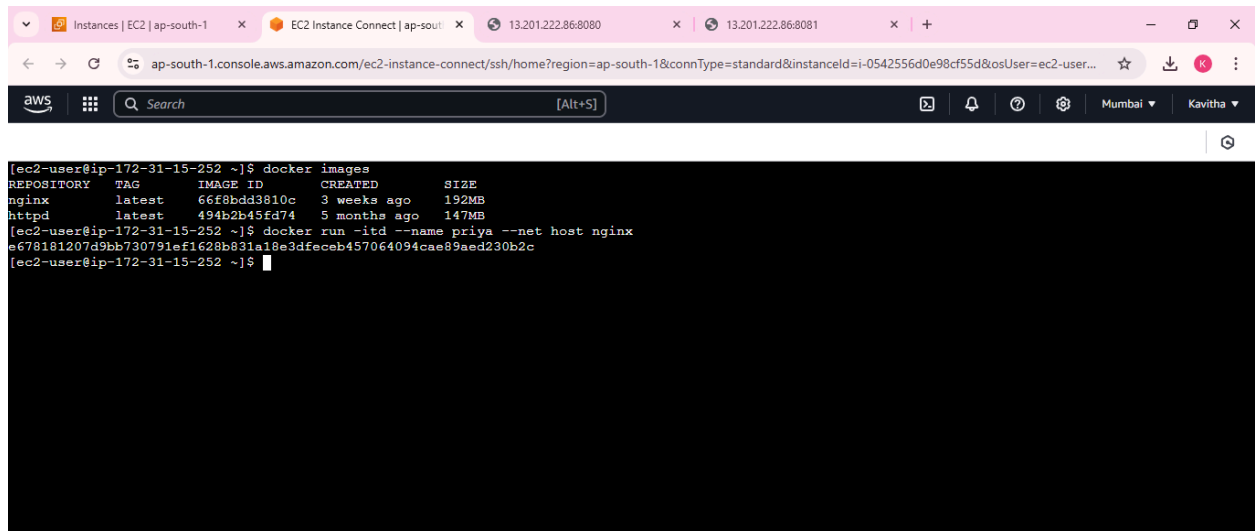
```
[ec2-user@ip-172-31-15-252 ~]$ docker kill sri kavi
sri
kavi
[ec2-user@ip-172-31-15-252 ~]$ docker rm sri kavi
sri
kavi
[ec2-user@ip-172-31-15-252 ~]$
```

**i-0542556d0e98cf55d (Docker Networking)**

PublicIPs: 13.201.222.86 PrivateIPs: 172.31.15.252

Activate Windows  
Go to Settings to activate Windows.

## Host Networking - Created a container with host networking



```
[ec2-user@ip-172-31-15-252 ~]$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
nginx         latest    66f8bdd3810c   3 weeks ago    192MB
httpd         latest    494b2b45fd74   5 months ago    147MB
[ec2-user@ip-172-31-15-252 ~]$ docker run -itd --name priya --net host nginx
e678161207d9bb730791ef1628b831a10e3dfecb457064094cae89aed230b2c
[ec2-user@ip-172-31-15-252 ~]$
```

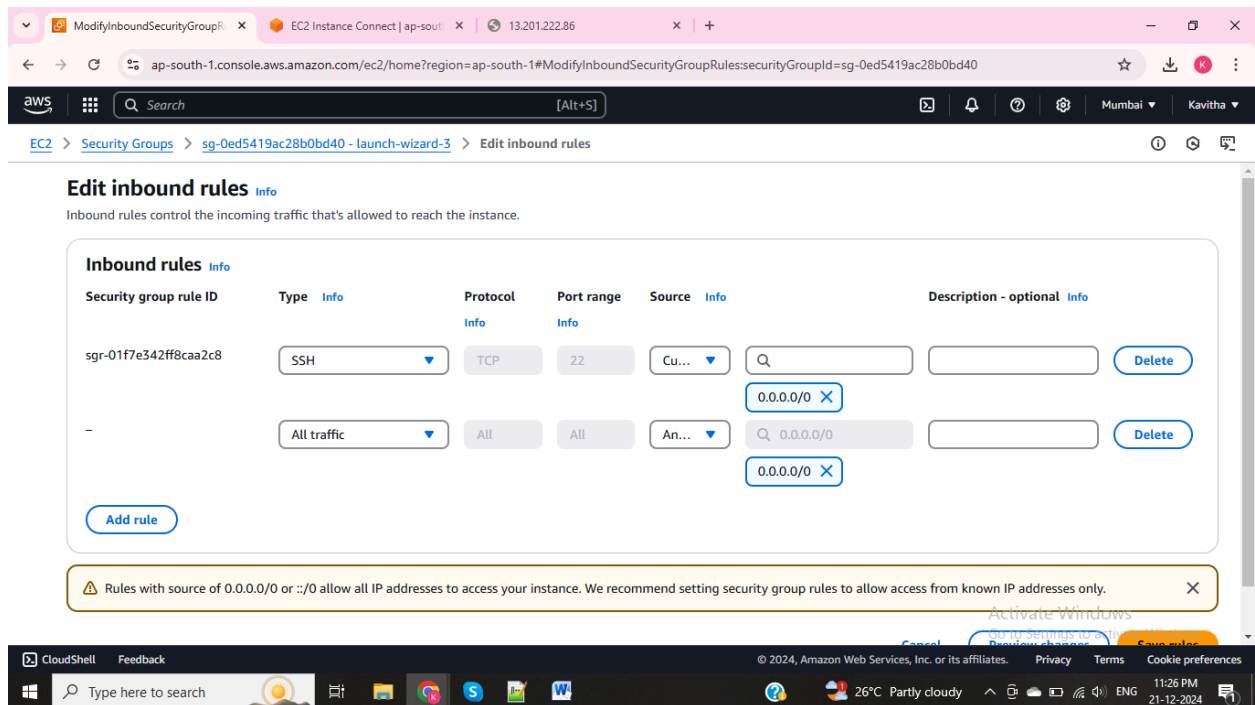
i-0542556d0e98cf55d (Docker Networking)

PublicIPs: 13.201.222.86 PrivateIPs: 172.31.15.252

Activate Windows  
Go to Settings to activate Windows.



## Go to security group edited the inbound rules type is All traffic



**Edit inbound rules** [Info](#)

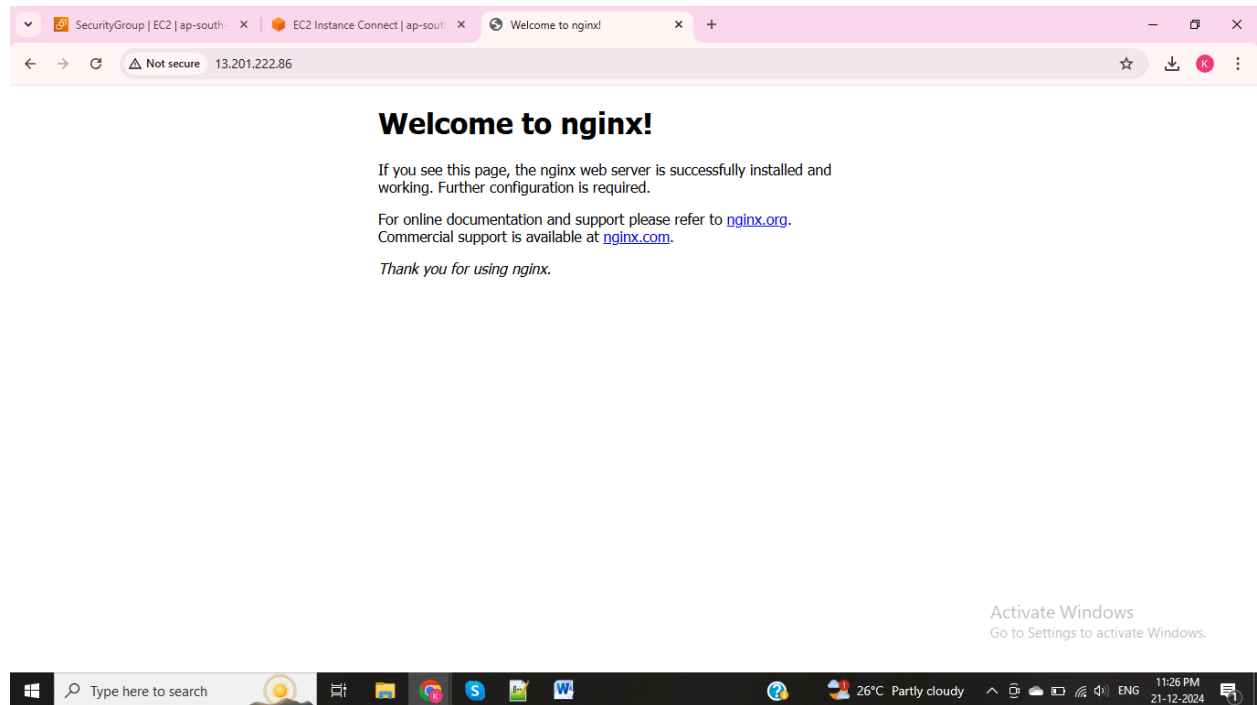
Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type <a href="#">Info</a>	Protocol <a href="#">Info</a>	Port range <a href="#">Info</a>	Source <a href="#">Info</a>	Description - optional <a href="#">Info</a>	
sg-01f7e342ff8caa2c8	SSH	TCP	22	Cu...		<a href="#">Delete</a>
-	All traffic	All	All	An...	0.0.0.0/0	<a href="#">Delete</a>

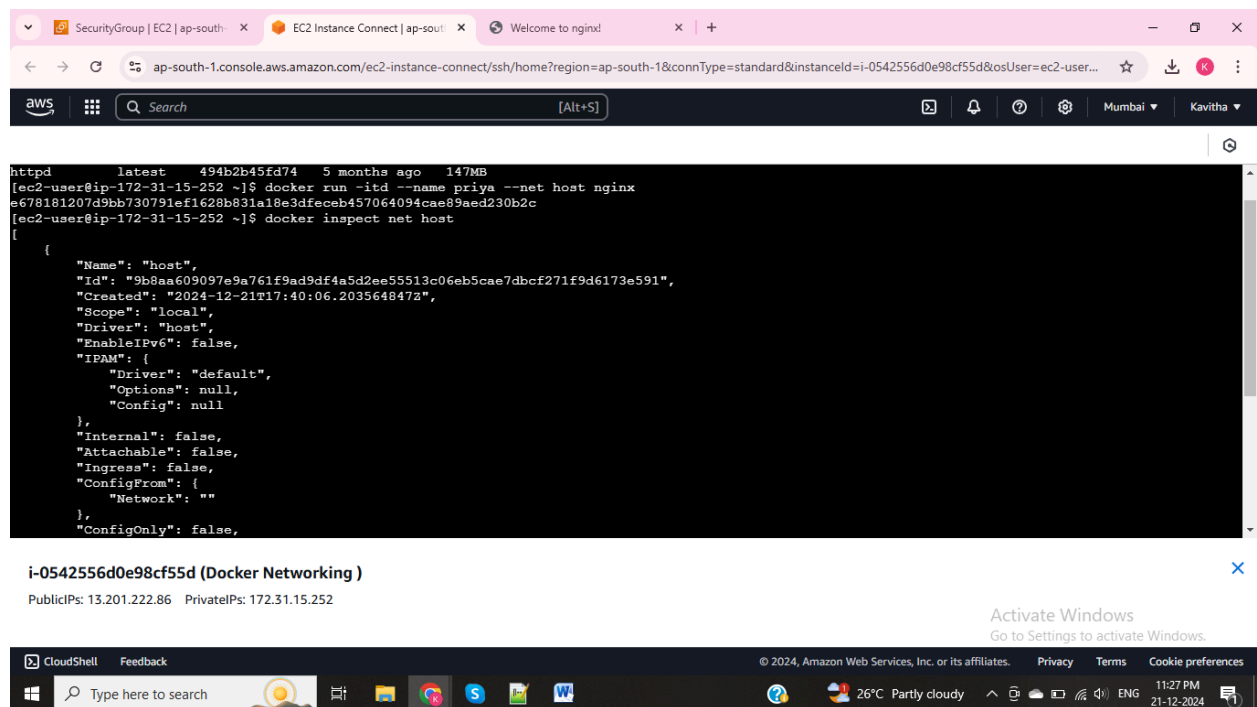
[Add rule](#)

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.

## Host Networking output - Application running only in public ip address



## Checking docker inspect net host



## None Networking - Disable the networking with conatiners

The screenshot shows the AWS CloudShell interface. The terminal window displays the following commands and output:

```
[ec2-user@ip-172-31-15-252 ~]$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
nginx         latest    66f8bdd3810c   3 weeks ago    192MB
httpd         latest    494b2b45fd74   5 months ago    147MB
[ec2-user@ip-172-31-15-252 ~]$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS          NAMES
e678161207d9   nginx     "/docker-entrypoint..." 2 minutes ago   Up 2 minutes   80/tcp         priya
[ec2-user@ip-172-31-15-252 ~]$ docker run -itd --name third --net none nginx
e0d647897decbabf594e42c952fff23a5b88fef8f933e6ff0c08374d07927e3a
[ec2-user@ip-172-31-15-252 ~]$
```

Below the terminal, the console output for instance **i-0542556d0e98cf55d (Docker Networking)** is shown. It includes the PublicIPs (13.201.222.86) and PrivateIPs (172.31.15.252). The console also displays the "Activate Windows" message and the "Go to Settings to activate Windows" link.

## None Networking output

The screenshot shows a web browser window with the address bar set to **13.201.222.86**. The page displays a connection error message:

**This site can't be reached**

**13.201.222.86** refused to connect.

Try:

- Checking the connection
- [Checking the proxy and the firewall](#)

ERR\_CONNECTION\_REFUSED

Buttons for **Reload** and **Details** are visible at the bottom of the error message.

The browser window also shows the "Activate Windows" message and the "Go to Settings to activate Windows" link.