D Y PATIL
INTERNATIONAL
UNIVERSITY
AKURDI PUNE

School of Computer Science, Engineering and Applications (SCSEA)
B.C.A. TY (CCSA)
Subject: Advanced Cloud Computing (P)

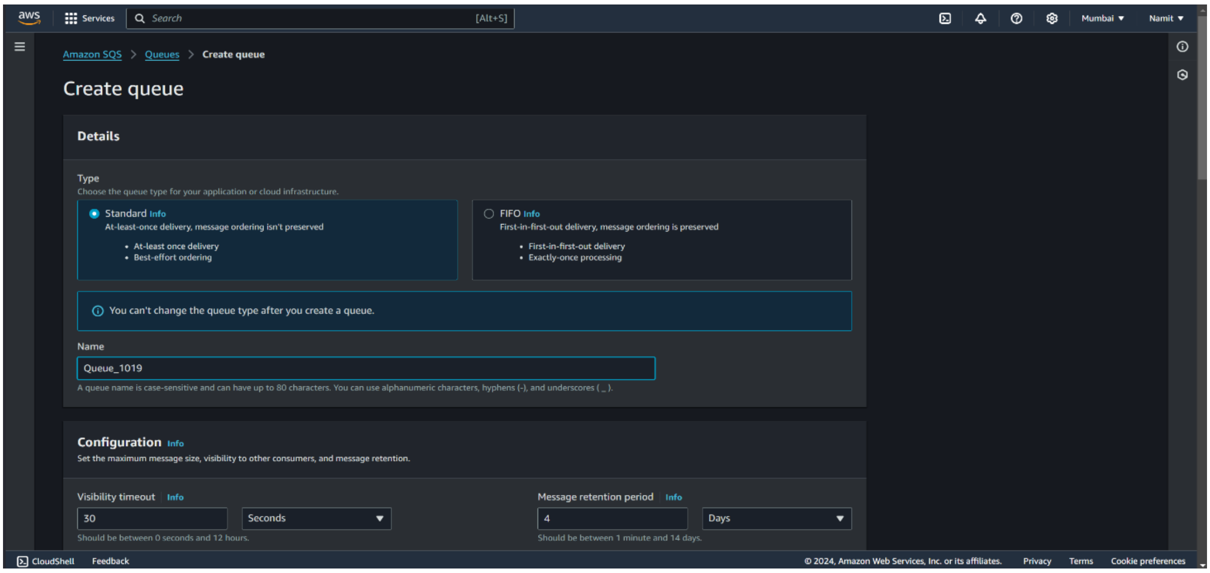
Name of the Student: **Namit Agarwal** **PRN: 20220801019**

Title of Practical: **Simple Queue Service (SQS) for sending messages**

1. Create an SQS Queue

1. Open the AWS Console, search for SQS, and click on Simple Queue Service (SQS).

2. Give a name to your queue.



Create queue

Details

Type
Choose the queue type for your application or cloud infrastructure.

● Standard Info
At-least-once delivery, message ordering isn't preserved
- At-least once delivery
- Best-effort ordering

○ FIFO Info
First-in-first-out delivery, message ordering is preserved
- First-in-first-out delivery
- Exactly-once processing

You can't change the queue type after you create a queue.

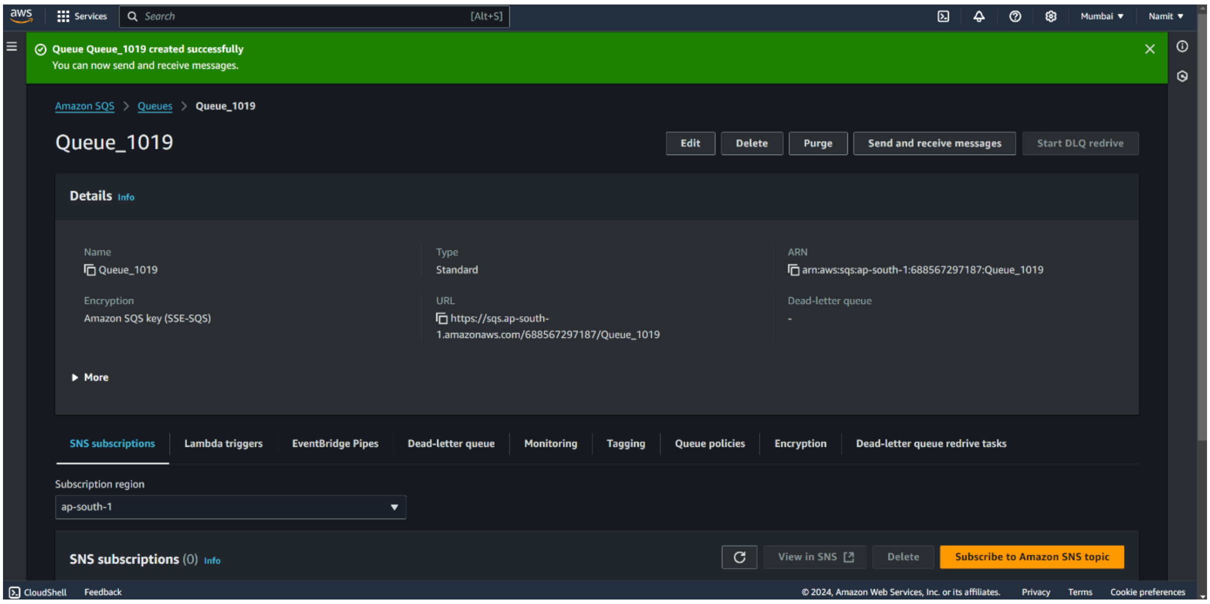
Name
Queue_1019
A queue name is case-sensitive and can have up to 80 characters. You can use alphanumeric characters, hyphens (-), and underscores (_).

Configuration Info
Set the maximum message size, visibility to other consumers, and message retention.

Visibility timeout Info
30 Seconds ▾
Should be between 0 seconds and 12 hours.

Message retention period Info
4 Days ▾
Should be between 1 minute and 14 days.

4. Scroll down and click **Create Queue**.



Queue Queue_1019 created successfully
You can now send and receive messages.

Queue_1019

Details Info

Name
Queue_1019

Type
Standard

ARN
arn:aws:sqs:ap-south-1:688567297187:Queue_1019

Encryption
Amazon SQS key (SSE-SQS)

URL
https://sqs.ap-south-1.amazonaws.com/688567297187/Queue_1019

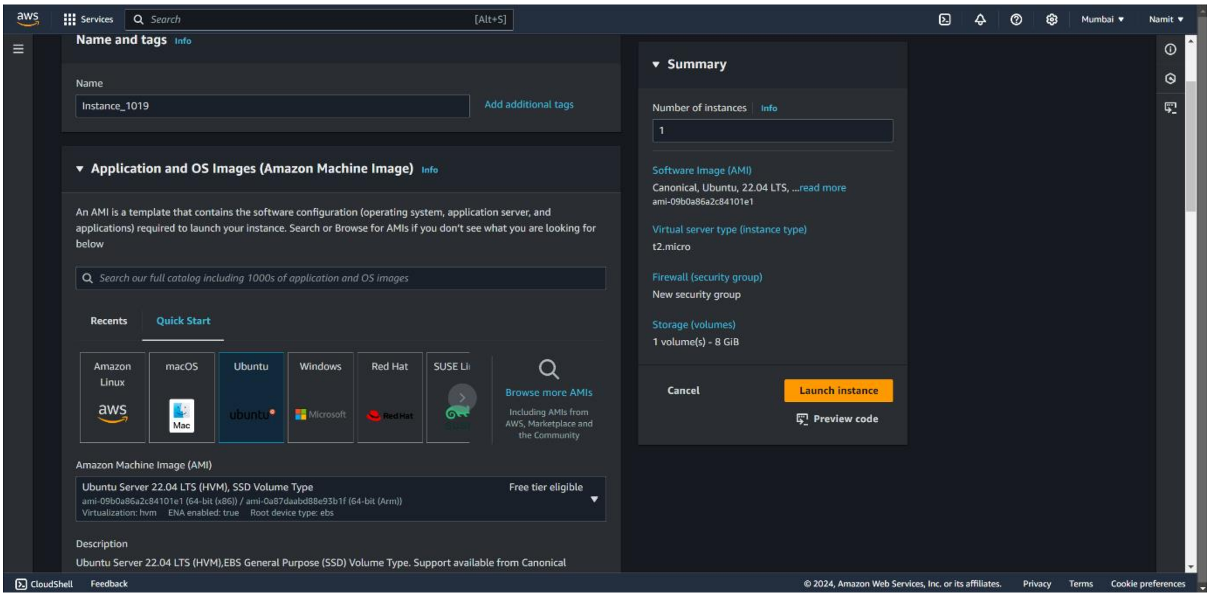
Dead-letter queue
-

Subscription region
ap-south-1

SNS subscriptions (0) Info

2. Launch an EC2 Instance

1. Search for EC2 and click on **Launch Instance**.
2. Give a name to your EC2 instance.
   - For AMI, select Ubuntu, and in the dropdown, select Ubuntu 22.04.



Name and tags

Name: Instance_1019

Application and OS Images (Amazon Machine Image)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-09b0a86a2c84101e1 (64-bit (x86)) / ami-0a87daabd88e93b1f (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Ubuntu Server 22.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical

Summary

Number of instances: 1

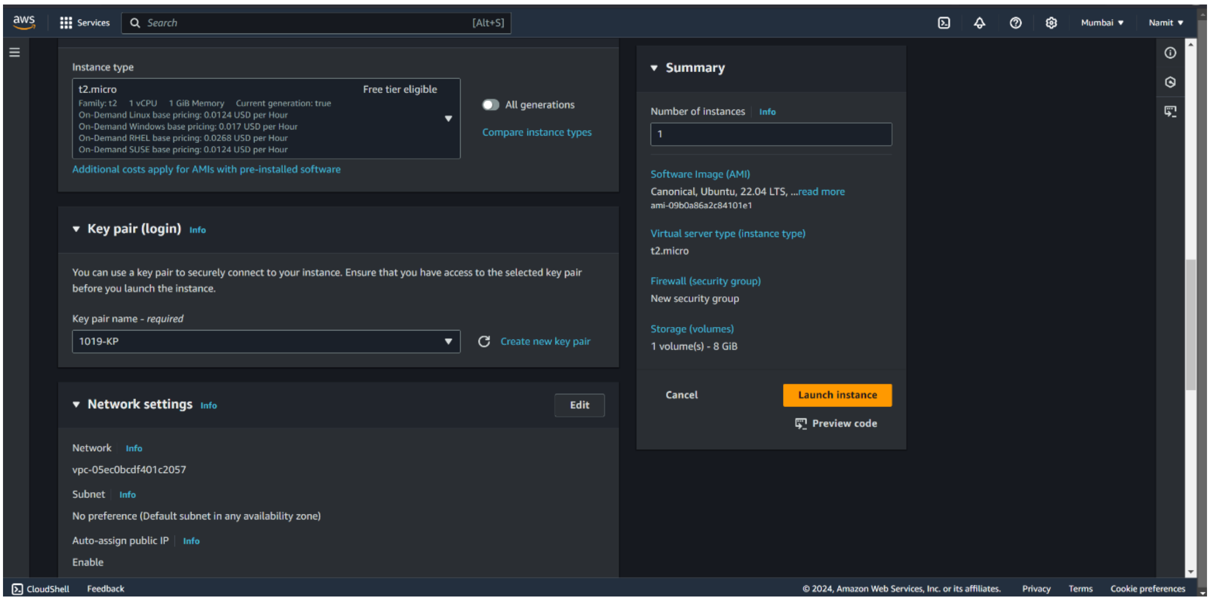
Software Image (AMI)
Canonical, Ubuntu, 22.04 LTS, ...read more
ami-09b0a86a2c84101e1

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

3. Create a key pair and save it.
4. In the Network section, click on **Edit** and select **Auto-assign IP**. Enable it and click **Add**.
5. Click on **Add Security Group Rule** in the Inbound Security Group Rules subsection of network settings.



Instance type

t2.micro — Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0124 USD per Hour
On-Demand Windows base pricing: 0.017 USD per Hour
On-Demand RHEL base pricing: 0.0268 USD per Hour
On-Demand SUSE base pricing: 0.0124 USD per Hour

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*: 1019-KP

Create new key pair

Network settings Info — Edit

Network: vpc-05ec0bcdf401c2057

Subnet: No preference (Default subnet in any availability zone)

Auto-assign public IP: Enable

Summary

Number of instances: 1

Software Image (AMI): Canonical, Ubuntu, 22.04 LTS, ...read more
ami-09b0a86a2c84101e1

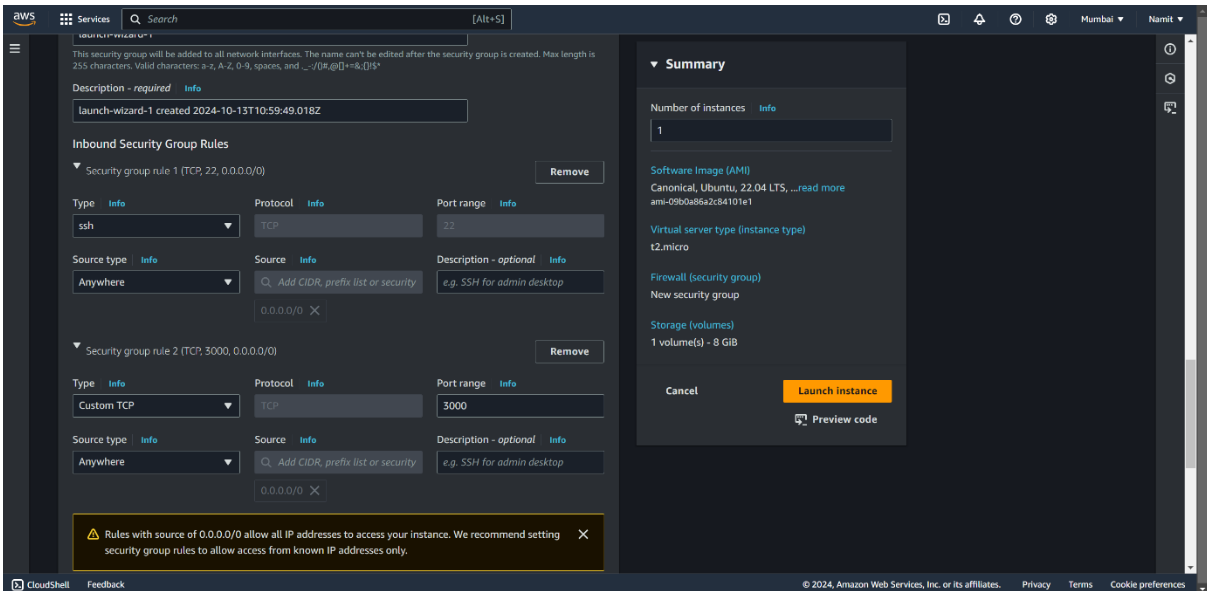
Virtual server type (instance type): t2.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Cancel — Launch instance — Preview code

6. Click on **Launch Instance**.



This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&;{}!$*

Description - *required*: launch-wizard-1 created 2024-10-13T10:59:49.018Z

Inbound Security Group Rules

Security group rule 1 (TCP, 22, 0.0.0.0/0) — Remove

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

Security group rule 2 (TCP, 3000, 0.0.0.0/0) — Remove

Type: Custom TCP | Protocol: TCP | Port range: 3000
Source type: Anywhere | Source: 0.0.0.0/0 | Description - *optional*: e.g. SSH for admin desktop

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

Summary

Number of instances: 1

Software Image (AMI): Canonical, Ubuntu, 22.04 LTS, ...read more
ami-09b0a86a2c84101e1

Virtual server type (instance type): t2.micro

Firewall (security group): New security group

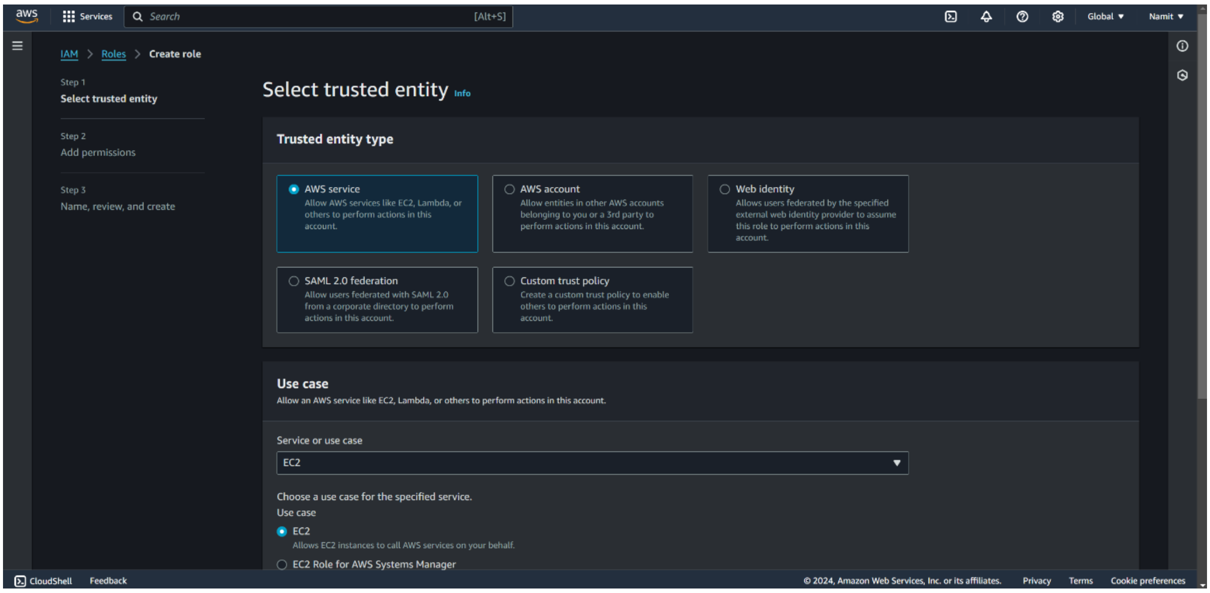
Storage (volumes): 1 volume(s) - 8 GiB

Cancel — Launch instance — Preview code

aws
Services
Search [Alt+S]
Mumbai
Namit
Add CIDR, prefix list or security
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

3. Create an IAM Role

1. Search for IAM (Identity Access Management).
2. In IAM, on the left-side panel, click on **Roles**.
3. Click on **Create Role**.
4. In the **Select trusted entity** section, choose **AWS Service** and select **EC2** in the use case.
5. Click **Next**, then search for the sqsfullaccess policy and select it.



IAM > Roles > Create role

Step 1
Select trusted entity

Step 2
Add permissions

Step 3
Name, review, and create

Select trusted entity Info

Trusted entity type

● AWS service
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

○ AWS account
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

○ Web identity
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

○ SAML 2.0 federation
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

○ Custom trust policy
Create a custom trust policy to enable others to perform actions in this account.

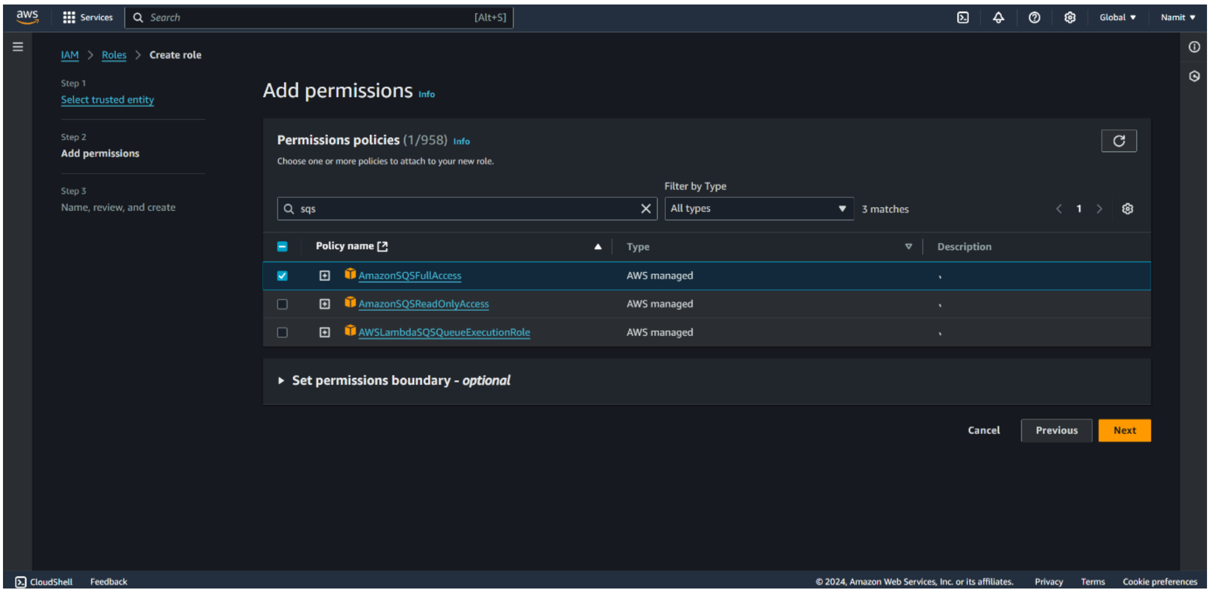
Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case
EC2 ▾

Choose a use case for the specified service.
Use case

● EC2
Allows EC2 instances to call AWS services on your behalf.

○ EC2 Role for AWS Systems Manager



IAM > Roles > Create role

Step 1
Select trusted entity

Step 2
Add permissions

Step 3
Name, review, and create

Add permissions Info

Permissions policies (1/958) Info

Choose one or more policies to attach to your new role.

Filter by Type: All types

sqs — 3 matches

	Policy name	Type	Description
☑	AmazonSQSFullAccess	AWS managed	,
□	AmazonSQSReadOnlyAccess	AWS managed	,
□	AWSLambdaSQSQueueExecutionRole	AWS managed	,

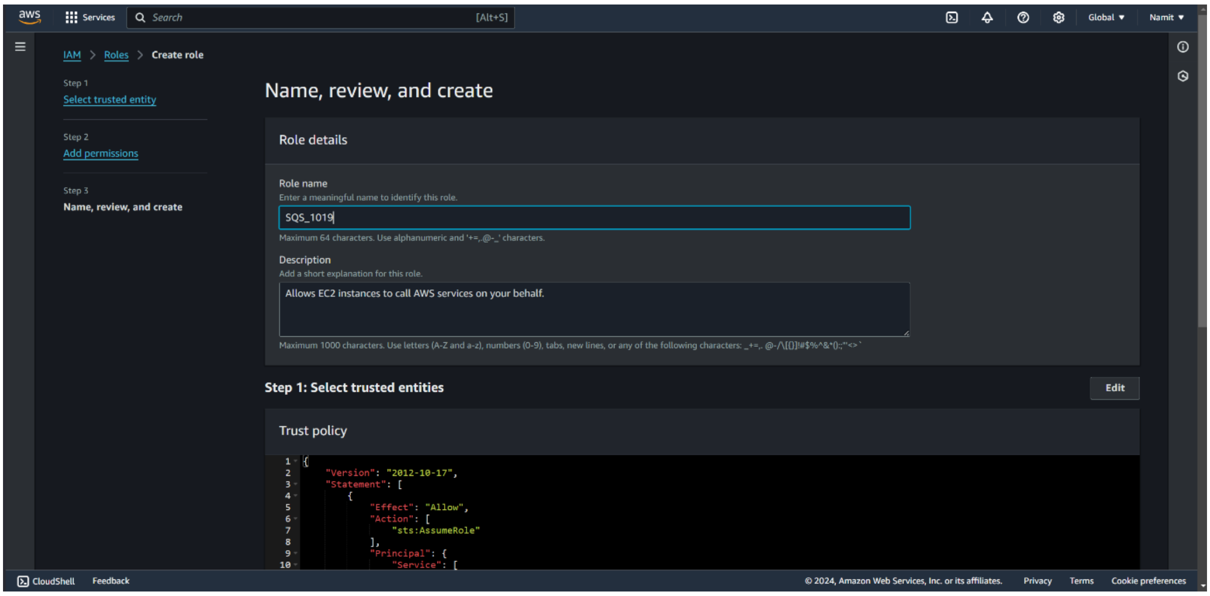
▸ Set permissions boundary - *optional*

6. Click **Next**, give a name to the role, and click **Create Role**.

7. After creating the role successfully, navigate back to your previously created EC2 instance.

8. Click on **Actions**, select **Security**, then **Modify IAM Role**.

9. Select the role you just created.



IAM > Roles > Create role

Step 1
Select trusted entity

Step 2
Add permissions

Step 3
Name, review, and create

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.
SQS_1019
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.

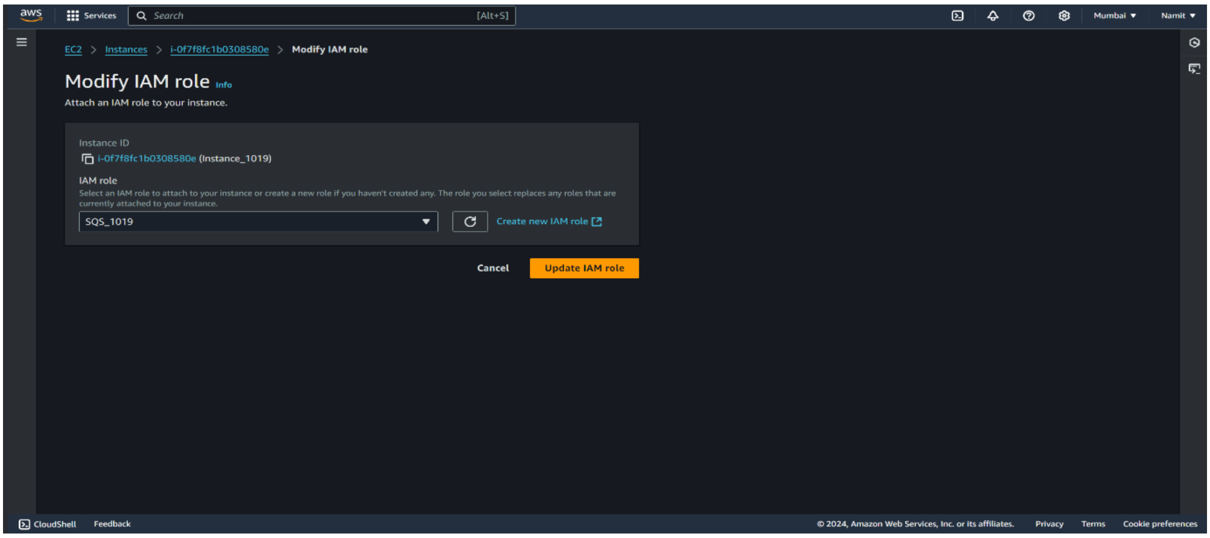
Description
Add a short explanation for this role.
Allows EC2 instances to call AWS services on your behalf.
Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,. @-/\[{}]!#$%^&*();:"'<>`

Step 1: Select trusted entities

Trust policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
```

10. Click **Update IAM Role**.



EC2 > Instances > i-0f7f8fc1b0308580e > Modify IAM role

Modify IAM role

Attach an IAM role to your instance.

Instance ID
i-0f7f8fc1b0308580e (Instance_1019)

IAM role
Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

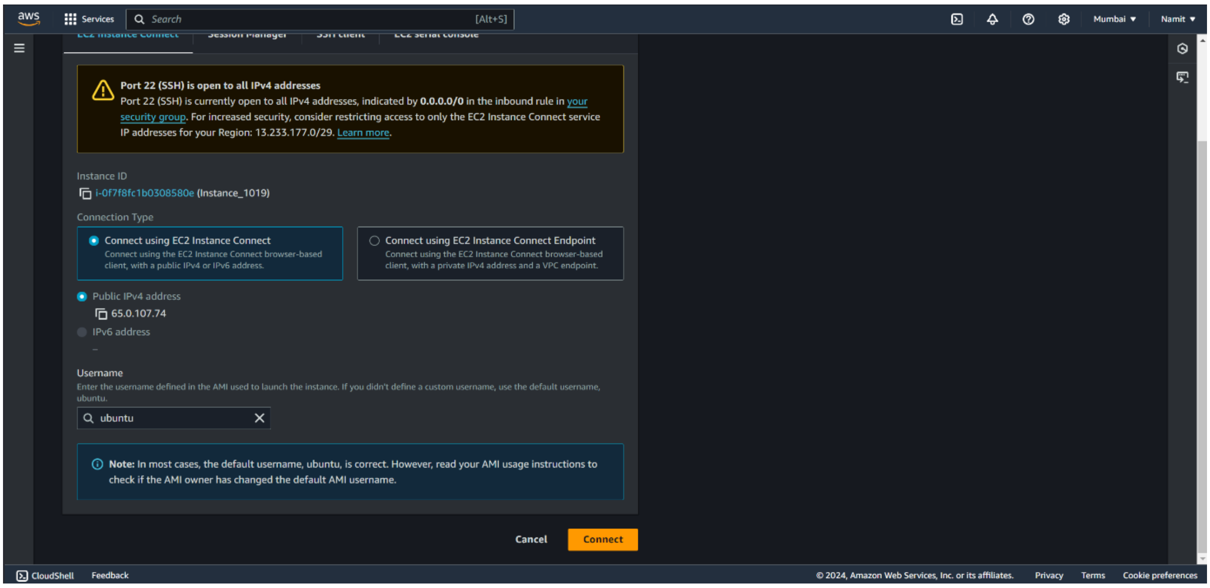
SQS_1019

4. Connect to the EC2 Instance

Connect to your EC2 instance using SSH.

5. Update and Set Up the EC2 Instance

5.1. Update and Upgrade Your Machine, and Install Node.js and npm



Port 22 (SSH) is open to all IPv4 addresses
Port 22 (SSH) is currently open to all IPv4 addresses, indicated by **0.0.0.0/0** in the inbound rule in your security group. For increased security, consider restricting access to only the EC2 Instance Connect service IP addresses for your Region: 13.233.177.0/29. Learn more.

Instance ID
i-0f7f8fc1b0308580e (Instance_1019)

Connection Type

● Connect using EC2 Instance Connect
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.

○ Connect using EC2 Instance Connect Endpoint
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

● Public IPv4 address
65.0.107.74

○ IPv6 address
–

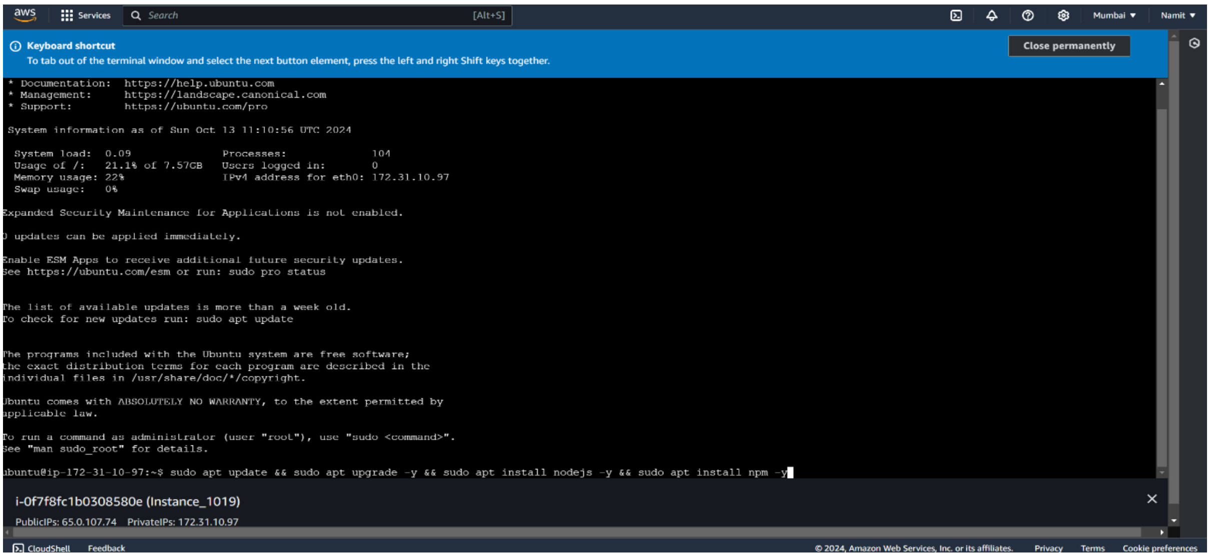
Username
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ubuntu.

ubuntu

Note: In most cases, the default username, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

After connecting to your EC2 instance, run the following commands:

sudo apt update && sudo apt upgrade -y && sudo apt install nodejs -y && sudo apt install npm -y


aws
Services
Search
[Alt+S]
Mumbai ▼
Namit ▼
Keyboard shortcut
To tab out of the terminal window and select the next button element, press the left and right Shift keys together.
Close permanently

```
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sun Oct 13 11:10:56 UTC 2024

  System load:  0.09               Processes:             104
  Usage of /:   21.1% of 7.57GB    Users logged in:       0
  Memory usage: 22%                IPv4 address for eth0: 172.31.10.97
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-10-97:~$ sudo apt update && sudo apt upgrade -y && sudo apt install nodejs -y && sudo apt install npm -y
```

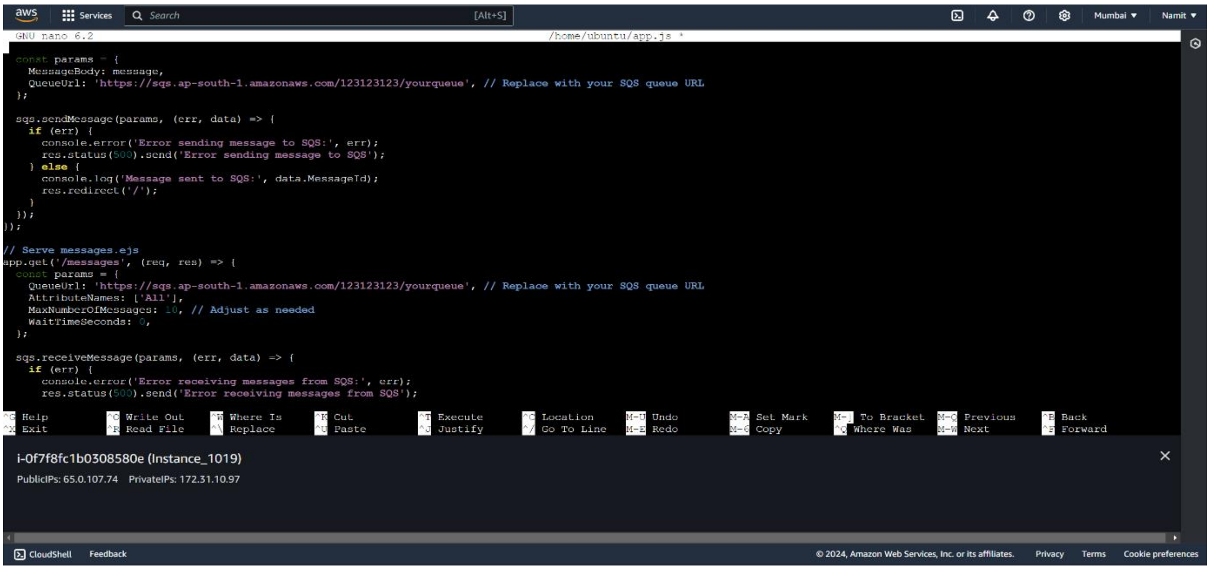
i-0f7f8fc1b0308580e (Instance_1019)

PublicIPs: 65.0.107.74 PrivateIPs: 172.31.10.97

5.2. Create app.js and Add Content

1. Create the app.js file:

nano ~/app.js



```
  GNU nano 6.2                                               /home/ubuntu/app.js *

  const params = {
    MessageBody: message,
    QueueUrl: 'https://sqs.ap-south-1.amazonaws.com/123123123/yourqueue', // Replace with your SQS queue URL
  };

  sqs.sendMessage(params, (err, data) => {
    if (err) {
      console.error('Error sending message to SQS:', err);
      res.status(500).send('Error sending message to SQS');
    } else {
      console.log('Message sent to SQS:', data.MessageId);
      res.redirect('/');
    }
  });
});

// Serve messages.ejs
app.get('/messages', (req, res) => {
  const params = {
    QueueUrl: 'https://sqs.ap-south-1.amazonaws.com/123123123/yourqueue', // Replace with your SQS queue URL
    AttributeNames: ['All'],
    MaxNumberOfMessages: 10, // Adjust as needed
    WaitTimeSeconds: 0,
  };

  sqs.receiveMessage(params, (err, data) => {
    if (err) {
      console.error('Error receiving messages from SQS:', err);
      res.status(500).send('Error receiving messages from SQS');
^G Help      ^O Write Out  ^W Where Is  ^K Cut      ^T Execute  ^C Location    M-U Undo  M-A Set Mark  M-] To Bracket  M-Q Previous  ^B Back
^X Exit      ^R Read File  ^\ Replace   ^U Paste    ^J Justify  ^/ Go To Line  M-E Redo  M-6 Copy      ^Q Where Was    M-W Next      ^F Forward
```

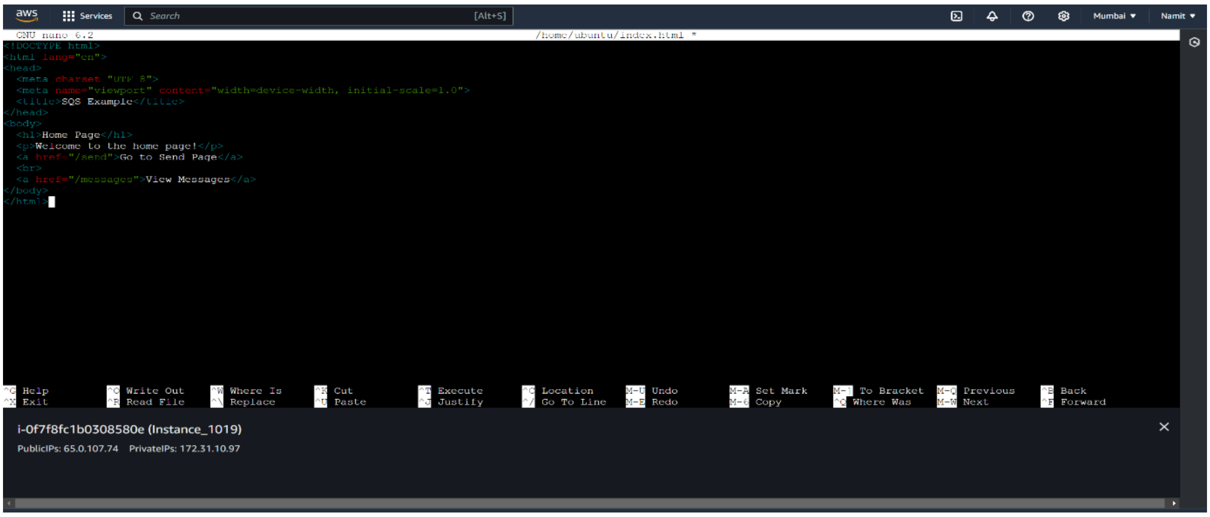
i-0f7f8fc1b0308580e (Instance_1019)

PublicIPs: 65.0.107.74 PrivateIPs: 172.31.10.97

5.3. Create index.html and Add Content

1. Create the index.html file:

nano ~/index.html



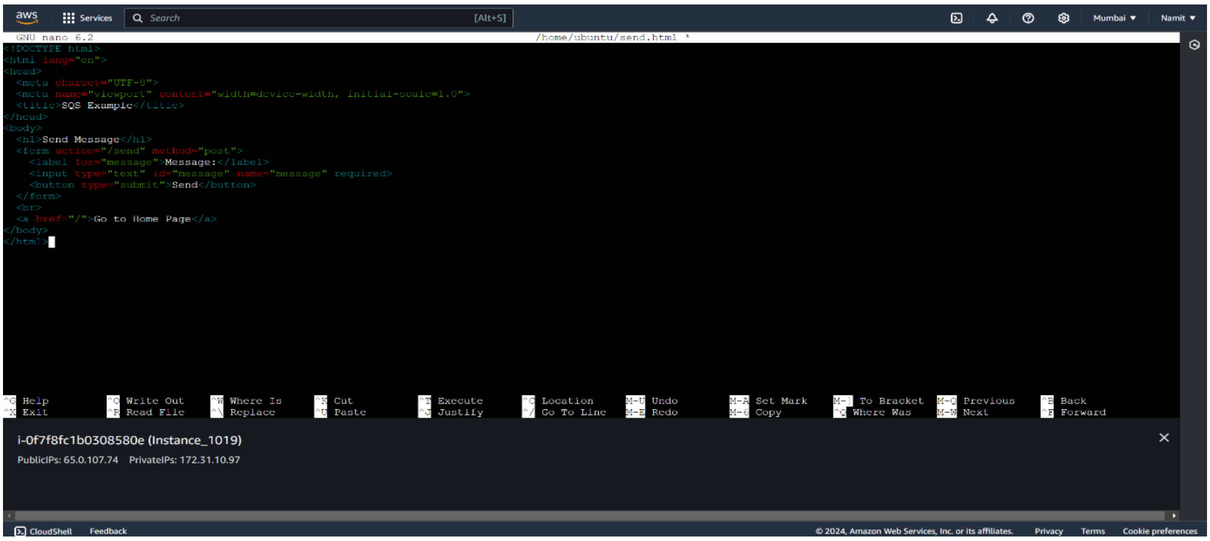
```
  GNU nano 6.2                               /home/ubuntu/index.html *
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>SQS Example</title>
</head>
<body>
  <h1>Home Page</h1>
  <p>Welcome to the home page!</p>
  <a href="/send">Go to Send Page</a>
  <br>
  <a href="/messages">View Messages</a>
</body>
</html>
```

i-0f7f8fc1b0308580e (Instance_1019)
PublicIPs: 65.0.107.74 PrivateIPs: 172.31.10.97

5.4. Create send.html and Add Content

1. Create the send.html file:

nano ~/send.html



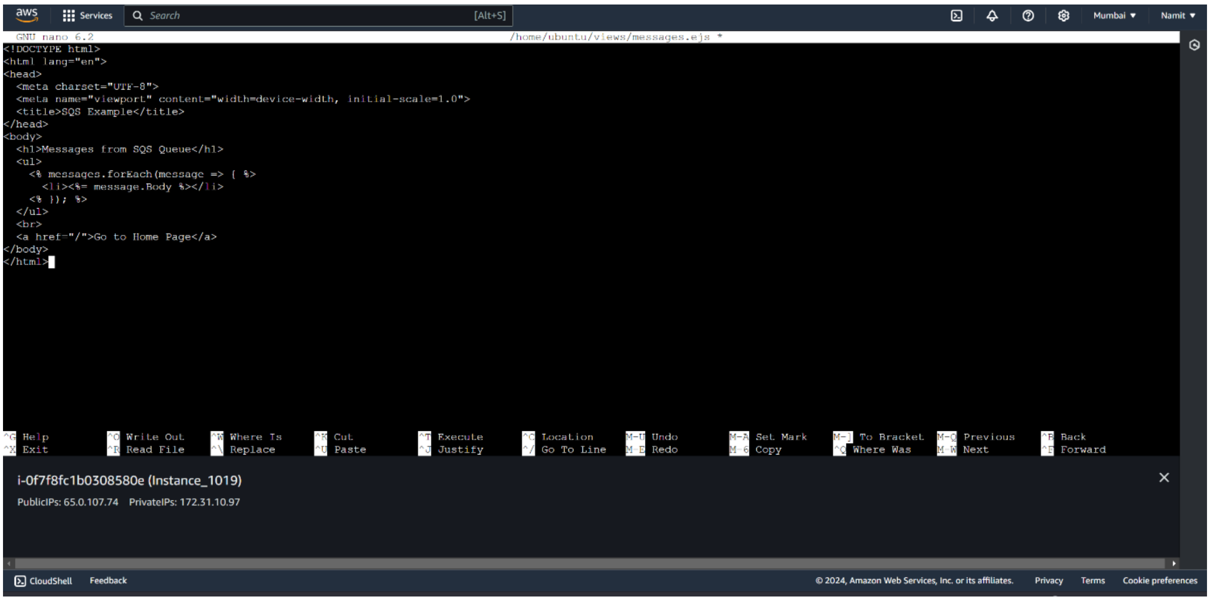
```
  GNU nano 6.2                               /home/ubuntu/send.html *
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>SQS Example</title>
</head>
<body>
  <h1>Send Message</h1>
  <form action="/send" method="post">
    <label for="message">Message:</label>
    <input type="text" id="message" name="message" required>
    <button type="submit">Send</button>
  </form>
  <br>
  <a href="/">Go to Home Page</a>
</body>
</html>
```

i-0f7f8fc1b0308580e (Instance_1019)
PublicIPs: 65.0.107.74 PrivateIPs: 172.31.10.97

5.5. Create the views Directory and messages.ejs File

1. Create the views directory and the messages.ejs file inside the views directory:

mkdir ~/views && nano ~/views/messages.ejs



```
  GNU nano 6.2                                                          /home/ubuntu/views/messages.ejs *
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>SQS Example</title>
</head>
<body>
  <h1>Messages from SQS Queue</h1>
  <ul>
    <% messages.forEach(message => { %>
      <li><%= message.Body %></li>
    <% }); %>
  </ul>
  <br>
  <a href="/">Go to Home Page</a>
</body>
</html>

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark   M-] To Bracket M-Q Previous   ^B Back
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy       ^Q Where Was   M-W Next       ^F Forward
```

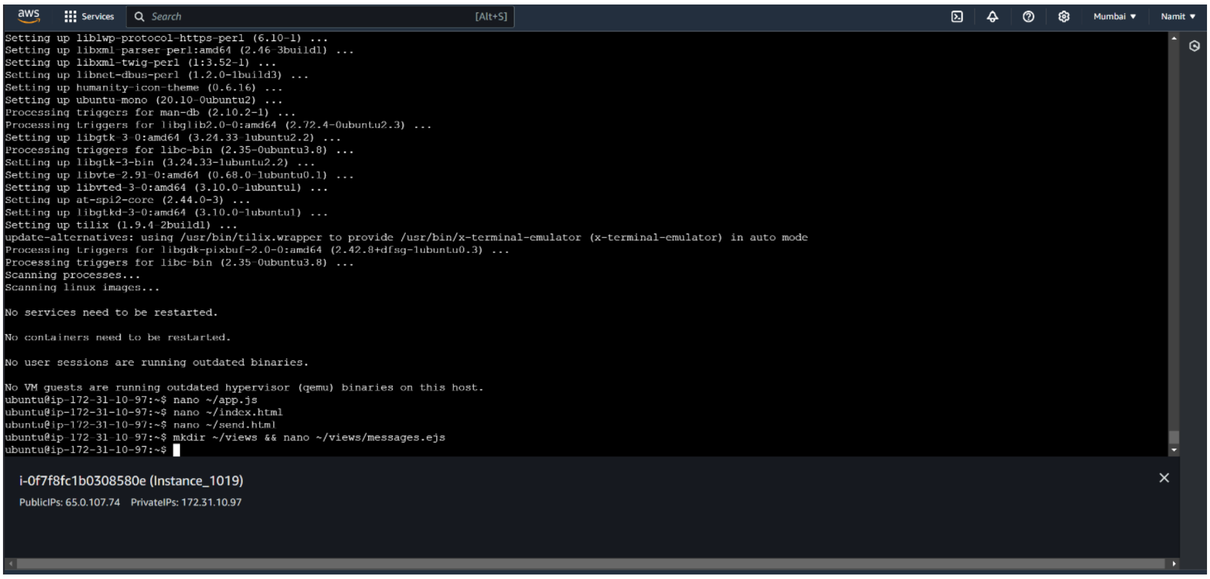
i-0f7f8fc1b0308580e (Instance_1019)

PublicIPs: 65.0.107.74 PrivateIPs: 172.31.10.97

5.6. Install Required Packages

Install the necessary Node.js packages:

npm install express aws-sdk body-parser ejs



```
Setting up liblwp-protocol-https-perl (6.10-1) ...
Setting up libxml-parser-perl:amd64 (2.46-3build1) ...
Setting up libxml-twig-perl (1:3.52-1) ...
Setting up libnet-dbus-perl (1.2.0-1build3) ...
Setting up humanity-icon-theme (0.6.16) ...
Setting up ubuntu-mono (20.10-0ubuntu2) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libglib2.0-0:amd64 (2.72.4-0ubuntu2.3) ...
Setting up libgtk-3-0:amd64 (3.24.33-1ubuntu2.2) ...
Processing triggers for libc-bin (2.35-0ubuntu3.8) ...
Setting up libgtk-3-bin (3.24.33-1ubuntu2.2) ...
Setting up libvte-2.91-0:amd64 (0.68.0-1ubuntu0.1) ...
Setting up libvted-3-0:amd64 (3.10.0-1ubuntu1) ...
Setting up at-spi2-core (2.44.0-3) ...
Setting up libgtkd-3-0:amd64 (3.10.0-1ubuntu1) ...
Setting up tilix (1.9.4-2build1) ...
update-alternatives: using /usr/bin/tilix.wrapper to provide /usr/bin/x-terminal-emulator (x-terminal-emulator) in auto mode
Processing triggers for libgdk-pixbuf-2.0-0:amd64 (2.42.8+dfsg-1ubuntu0.3) ...
Processing triggers for libc-bin (2.35-0ubuntu3.8) ...
Scanning processes...
Scanning linux images...

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-10-97:~$ nano ~/app.js
ubuntu@ip-172-31-10-97:~$ nano ~/index.html
ubuntu@ip-172-31-10-97:~$ nano ~/send.html
ubuntu@ip-172-31-10-97:~$ mkdir ~/views && nano ~/views/messages.ejs
ubuntu@ip-172-31-10-97:~$
```

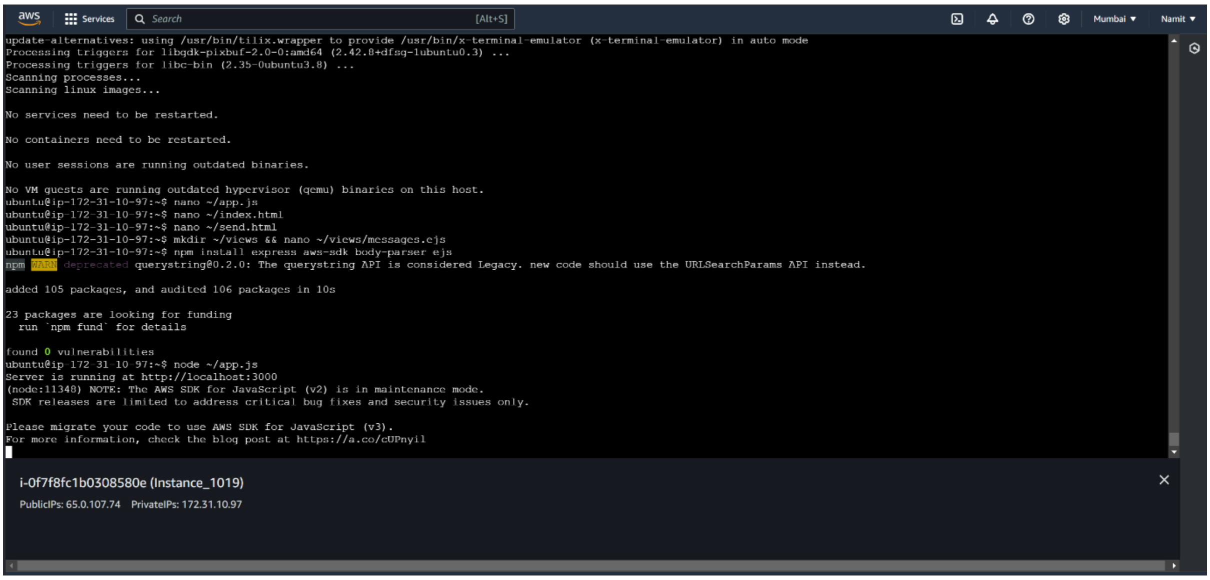
i-0f7f8fc1b0308580e (Instance_1019)

PublicIPs: 65.0.107.74 PrivateIPs: 172.31.10.97

5.7. Start the Application

Start the application using Node.js:

node ~/app.js

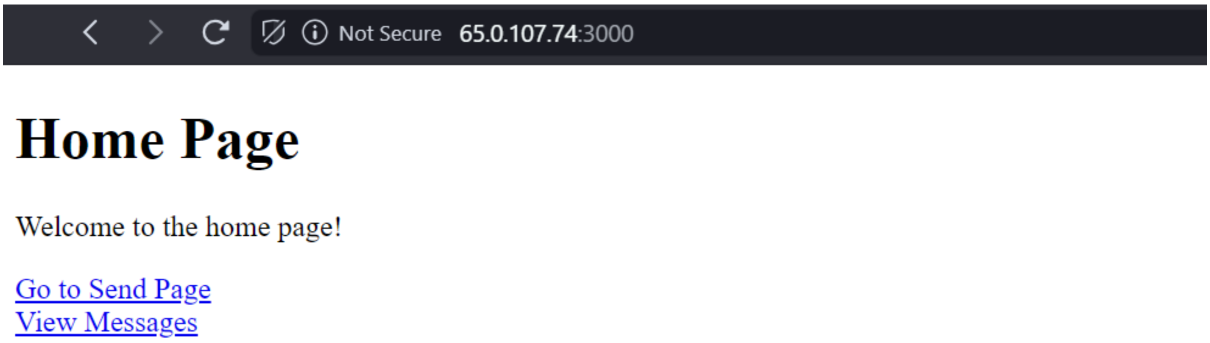


```
update-alternatives: using /usr/bin/tilix.wrapper to provide /usr/bin/x-terminal-emulator (x-terminal-emulator) in auto mode
Processing triggers for libgdk-pixbuf-2.0-0:amd64 (2.42.8+dfsg-1ubuntu0.3) ...
Processing triggers for libc-bin (2.35-0ubuntu3.8) ...
Scanning processes...
Scanning linux images...

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-10-97:~$ nano ~/app.js
ubuntu@ip-172-31-10-97:~$ nano ~/index.html
ubuntu@ip-172-31-10-97:~$ nano ~/send.html
ubuntu@ip-172-31-10-97:~$ mkdir ~/views && nano ~/views/messages.ejs
ubuntu@ip-172-31-10-97:~$ npm install express aws-sdk body-parser ejs
npm WARN deprecated querystring@0.2.0: The querystring API is considered Legacy. new code should use the URLSearchParams API instead.

added 105 packages, and audited 106 packages in 10s

23 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
ubuntu@ip-172-31-10-97:~$ node ~/app.js
Server is running at http://localhost:3000
(node:11348) NOTE: The AWS SDK for JavaScript (v2) is in maintenance mode.
 SDK releases are limited to address critical bug fixes and security issues only.

Please migrate your code to use AWS SDK for JavaScript (v3).
For more information, check the blog post at https://a.co/cUPnyil
```

i-0f7f8fc1b0308580e (Instance_1019)

PublicIPs: 65.0.107.74 PrivateIPs: 172.31.10.97

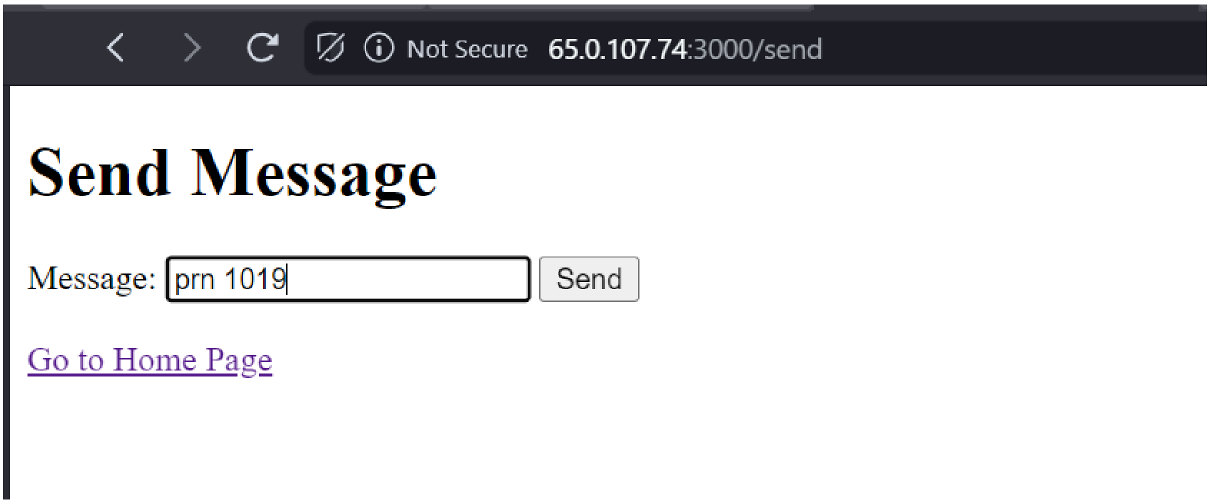
Application works, paste your ip address with port 3000



Home Page

Welcome to the home page!

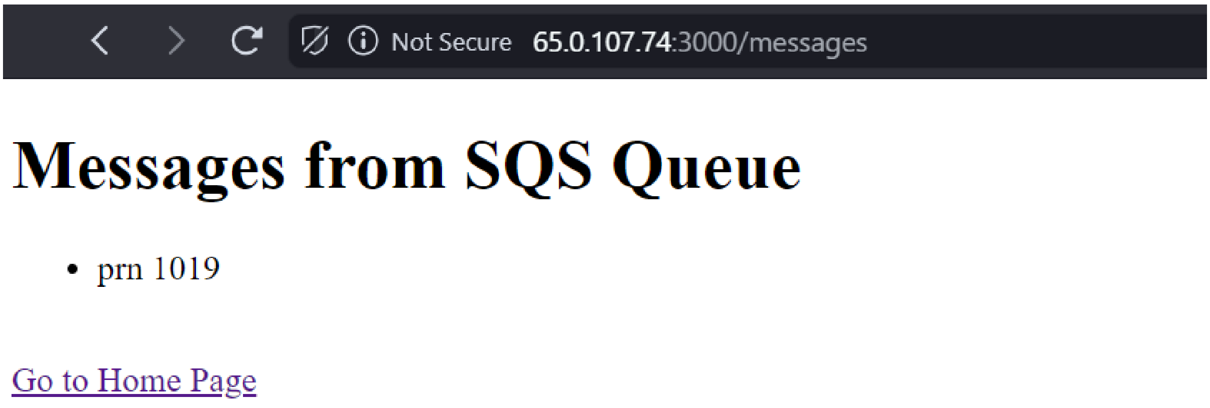
Go to Send Page
View Messages



Send Message

Message: prn 1019 Send

Go to Home Page



Messages from SQS Queue

- prn 1019

Go to Home Page