

Managing tightly coupled architecture using Amazon SQS

Steps to be followed:

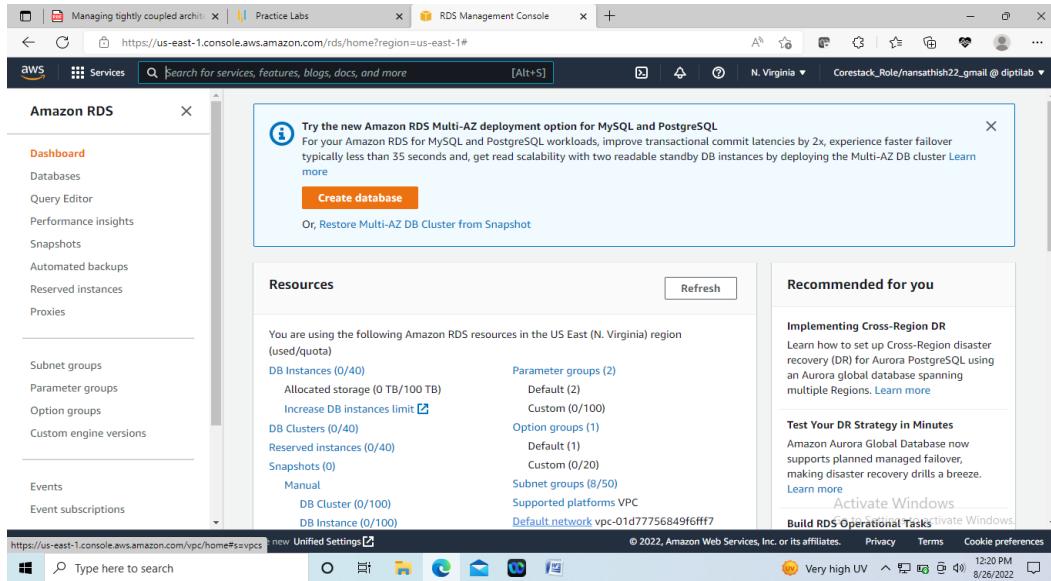
1. Create MySQL Database using RDS Service.
2. Create SQS Queue.
3. Connect through SQL Client and create a table.
4. Create a frontend and backend server using EC2 instance
5. Connect the SQS queue with the front end server to insert entries into MySQL dB in RDS service

Step 1: Make sure to create a Security Group with the port 3306 allowed in the Inbound rules.

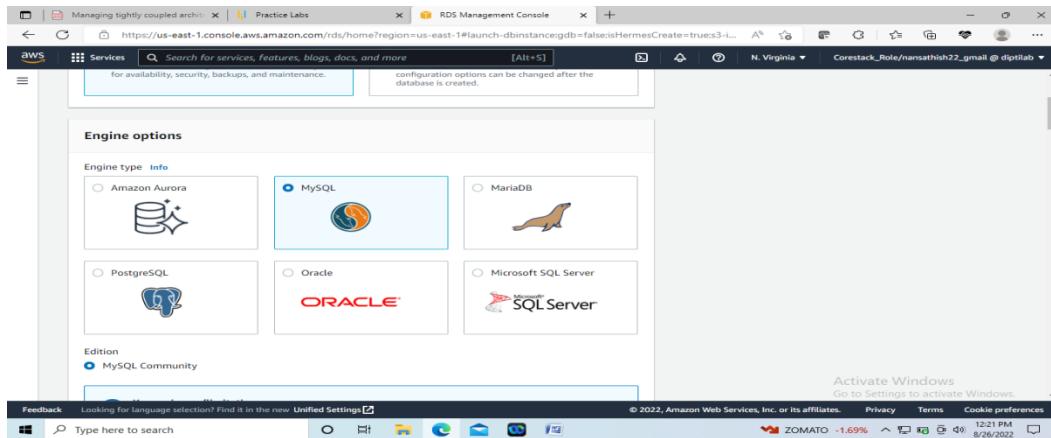
The screenshot shows two instances of the AWS EC2 Management Console. Both instances are displaying the 'Security Groups' page for a security group named 'sg-0696b3fb1c9ac4335 | AllowMySQL'. The security group has been created successfully. It is associated with a VPC ID 'vpc-080c1e7b73cca44d1'. The 'Inbound rules' tab is selected, showing two permission entries:

Security group name	IP version	Type	Protocol	Port range
sgr-0edead337adb5e3...	IPv6	MySQL/Aurora	TCP	3306
sgr-038a21abb5b1fe079	IPv4	MySQL/Aurora	TCP	3306

Step 2: Go to the RDS Management Console and click on Create Database.



Step 3: Select MySQL as the Database.



Step 4: Select Free tier for the templates. This will make sure all the settings for the RDS are for the AWS Free Tier (<https://aws.amazon.com/free/>). Specify the DB instance identifier as Customerdatabase1

The screenshot shows the AWS RDS Management Console for MySQL Community. The 'Known issues/limitations' section is visible, along with a dropdown for 'Version' set to MySQL 8.0.28. Under 'Templates', the 'Free tier' option is selected. In the 'Availability and durability' section, there are three options: 'Production', 'Dev/Test', and 'Free tier'. The 'Free tier' option is highlighted with a blue border. The status bar at the bottom indicates 'Activate Windows Go to Settings to activate Windows.'

Step 5: Specify the username and the password.

The screenshot shows the 'Settings' configuration page for a DB instance. It includes fields for 'DB instance identifier' (Customerdatabase1), 'Master username' (Nandini), 'Master password' (*****), and 'Confirm password' (*****). The status bar at the bottom indicates 'Activate Windows Go to Settings to activate Windows.'

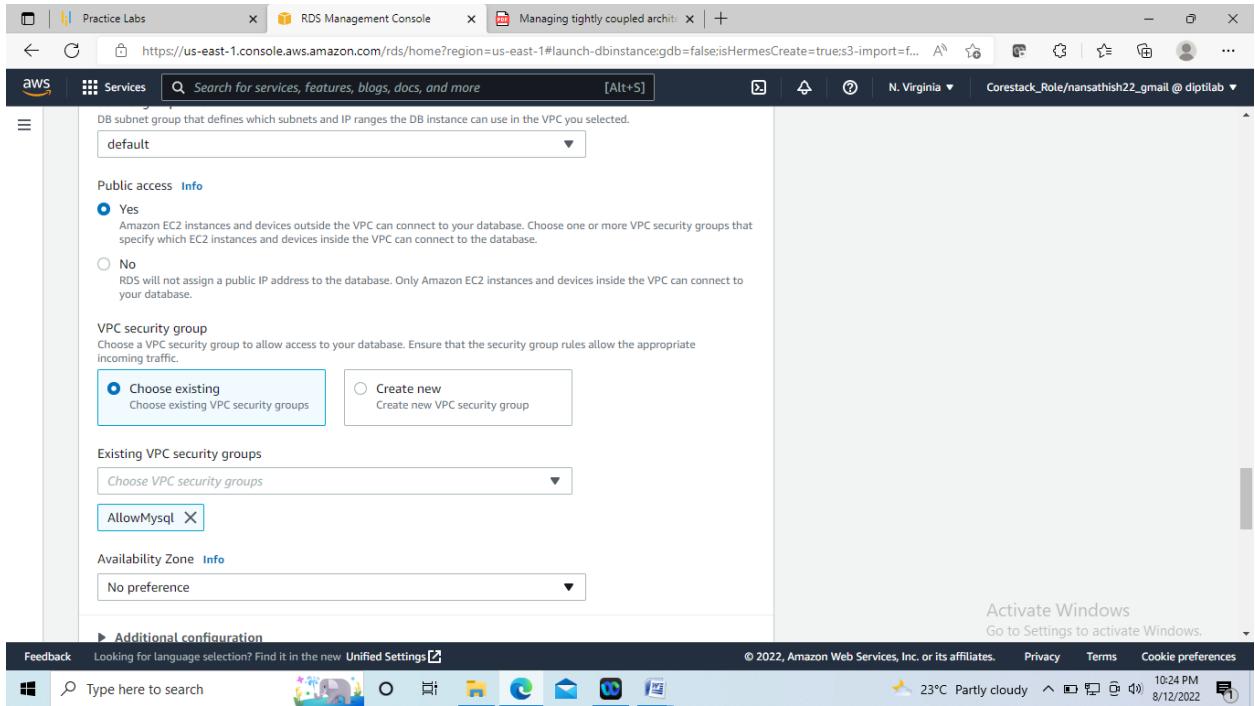
Step 6: Uncheck Storage autoscaling.

The screenshot shows the AWS RDS Management Console. In the top navigation bar, there are tabs for 'Managing tightly coupled archit...', 'Practice Labs', and 'RDS Management Console'. The main content area has a sidebar titled 'Allocated storage' with a value of '20 GiB'. Below it is a section for 'Storage autoscaling' with an option to 'Enable storage autoscaling'. Under 'Connectivity', there are sections for 'Compute resource' (with options for 'Don't connect to an EC2 compute resource' or 'Connect to an EC2 compute resource'), 'Virtual private cloud (VPC)' (set to 'Default VPC (vpc-01d77756849f6fff7)'), and 'Public access' (set to 'Yes'). The status bar at the bottom shows the date as 8/26/2022 and the time as 12:27 PM.

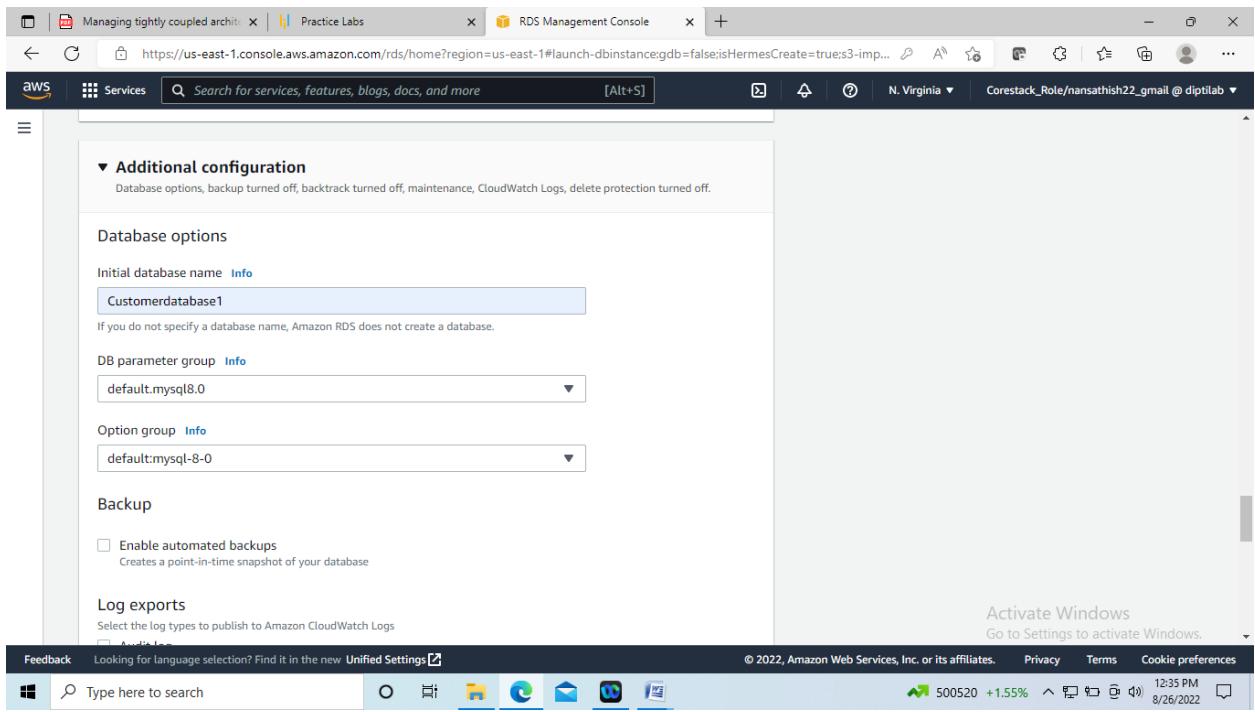
Step 7: Under Connectivity, for Publicly accessible as Yes.

This screenshot shows the 'Connectivity' configuration page. The 'Public access' section is highlighted, showing the 'Yes' option selected. A note below states: 'Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.' The status bar at the bottom shows the date as 8/12/2022 and the time as 10:23 PM.

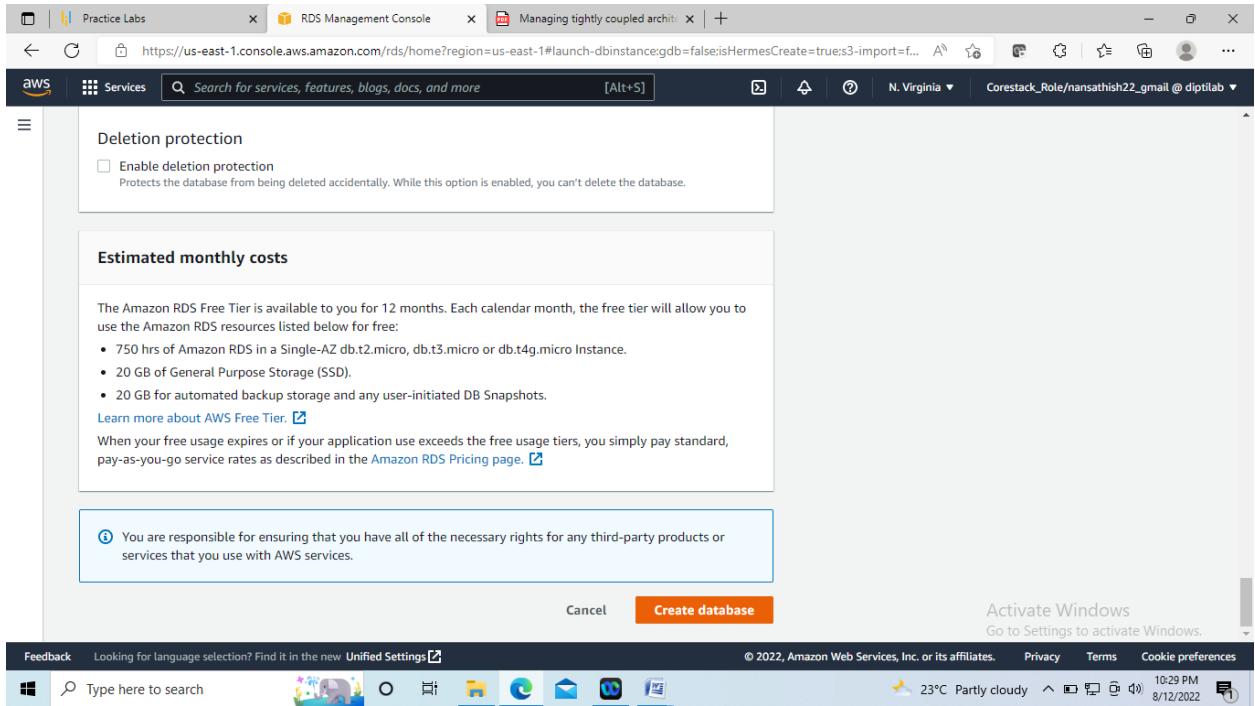
Step 8: Select the AllowMySQL Security Group created earlier.



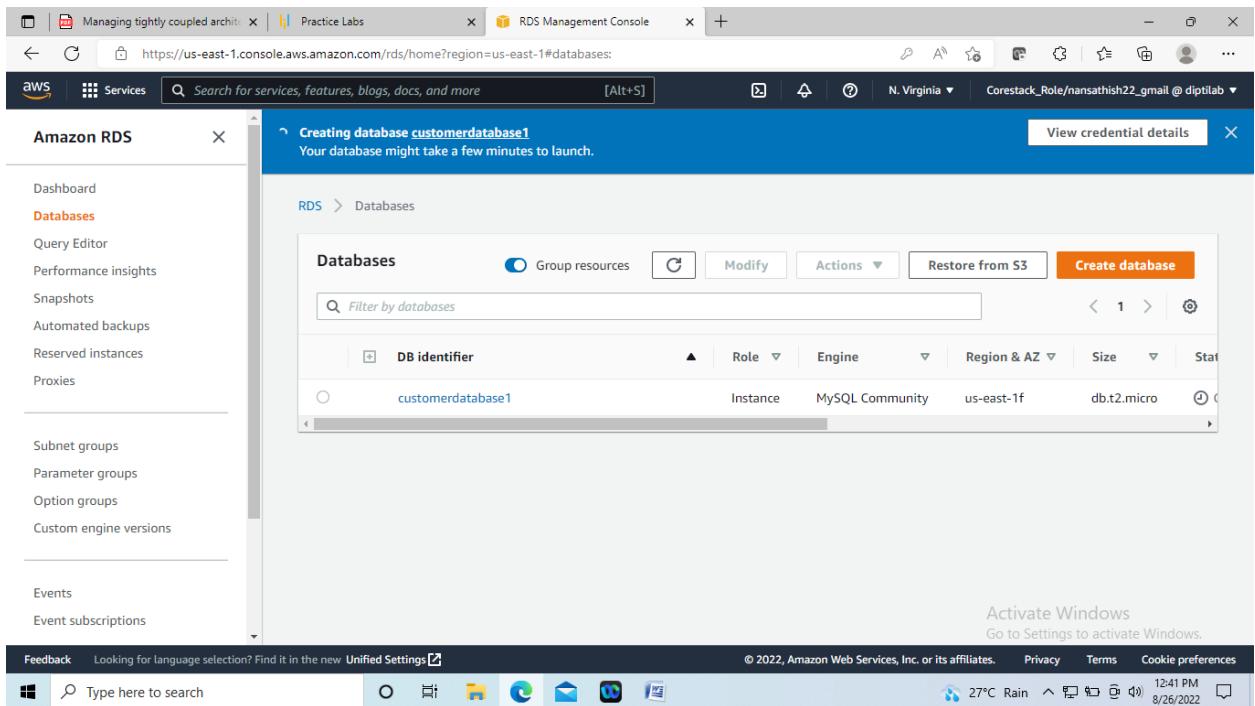
Step 9: For the Initial database name, specify Customerdatabase1. Disable the Backup.



Step 10: Finally, click on Create Database.



Step 11: Once the database is configured, it will show the status as creating, and then, after some time, it displays the status finally created.



Step 12: After a few minutes, the database should be in available status, and the endpoint would be populated.

The screenshot shows the AWS RDS Management Console for a MySQL Community instance named 'customerdatabase1'. The 'Connectivity & security' tab is selected. Key details shown include:

- Endpoint & port:** Endpoint: customerdatabase1.ciqaqfyzagmn.us-east-1.rds.amazonaws.com, Port: 3306.
- Networking:** Availability Zone: us-east-1f, VPC: vpc-01d77756849f6fff7, Subnet group: default-vpc-01d77756849f6fff7, Subnets: subnet-0a1d5ffab155a7599, subnet-0a77081462a566383, subnet-04de647e72db69a97, subnet-0bc444b21be1ba23b.
- Security:** VPC security groups: default (sg-0324abafca6f9bef8) (Active), Public accessibility: No, Certificate authority: rds-ca-2019, Certificate authority date: August 22, 2024, 22:38 (UTC+05:30).

The left sidebar lists various RDS management options like Databases, Query Editor, Performance insights, etc.

Step 13: Download the HeidiSQL (<https://www.heidisql.com/download.php>) by clicking on Installer, 32/64 bit combined and install it as any other application.

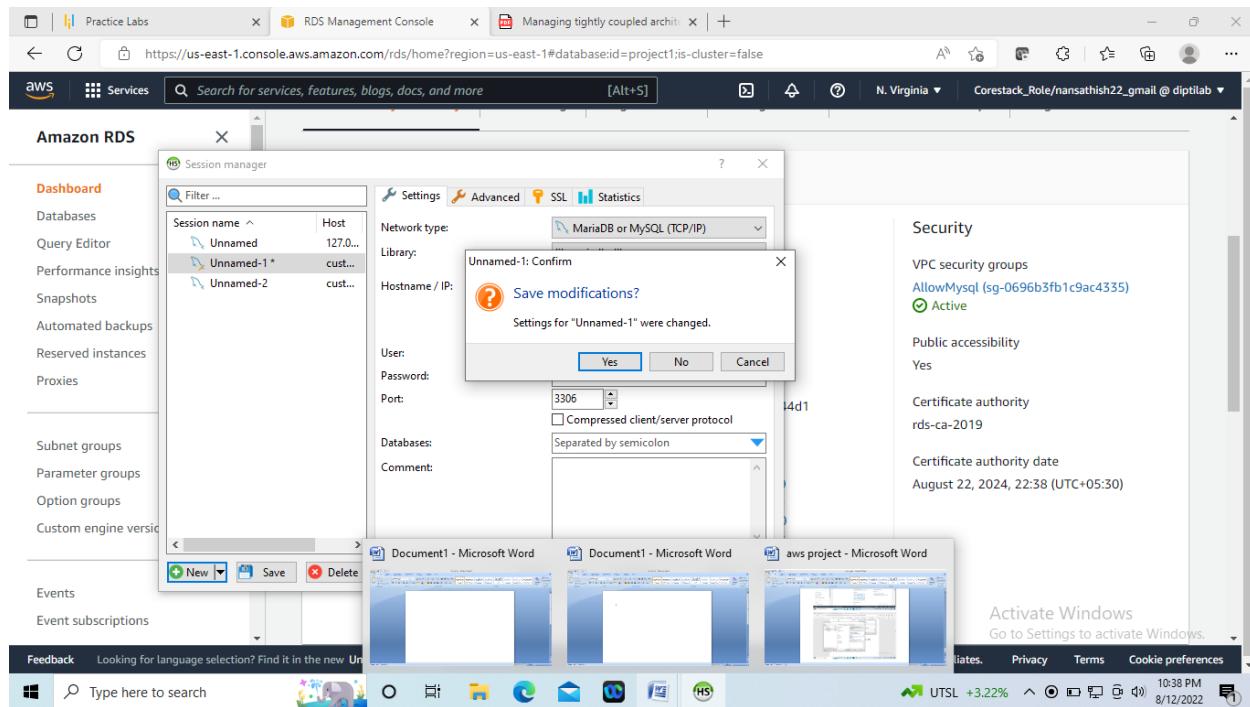
Step 14: Launch HeidiSQL once the installation is complete and enter the RDS/MySQL endpoint/user/password. Then, click on Open

The screenshot shows the HeidiSQL Session manager dialog box. A session named 'customerdb.cmeeo0ikklen.us-east-1.rds.amazonaws.com' is selected. The connection details are as follows:

- Network type: MariaDB or MySQL (TCP/IP)
- Library: libmariadb.dll
- Hostname / IP: ciqaqfyzagmn.us-east-1.rds.amazonaws.com
- User: nandini
- Password: (redacted)
- Port: 3306
- Databases: Separated by semicolon

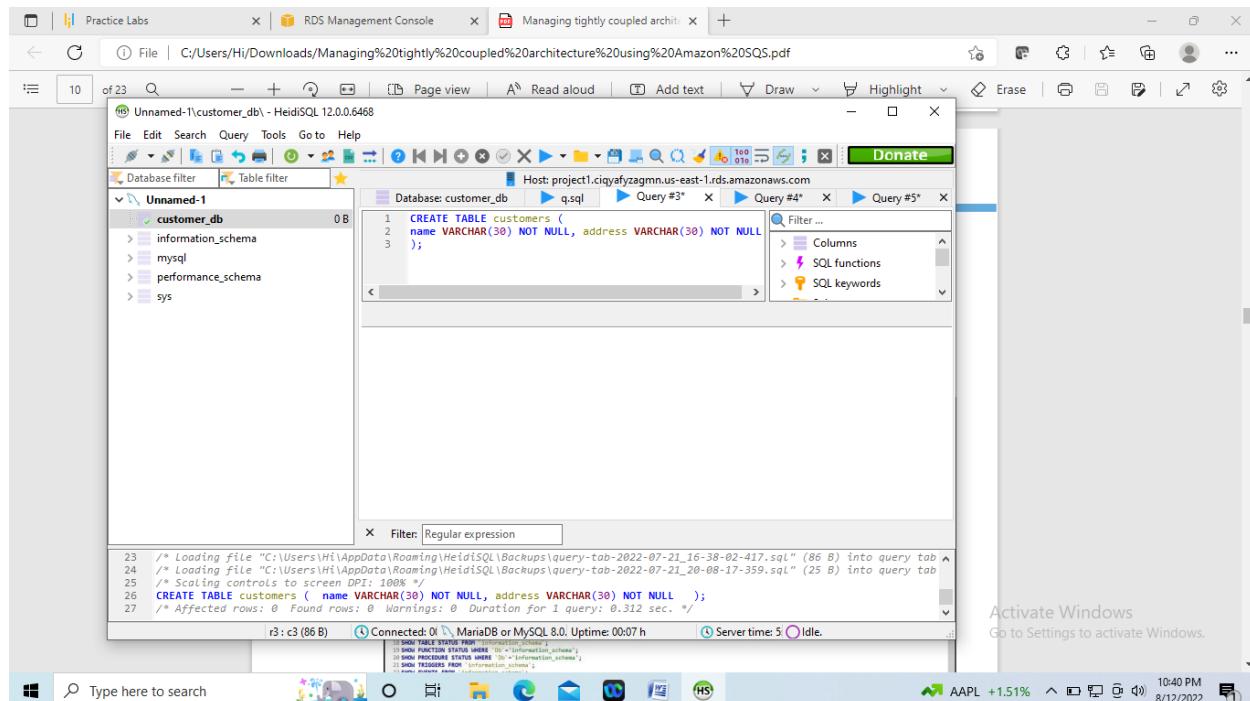
The 'Session manager' pane shows other sessions: 'Unnamed', 'Unnamed-1', and 'Unnamed-2'. The bottom status bar indicates 'Activate Windows Go to Settings to activate Windows.'

Step 15: Click on Yes when prompted. This will store the connection details.



Step 16: In the HeidiSQL, make sure to select Customerdatabase1 in the top left pane, go to the Query tab, copy the below DDL statement and click on Execute to create a table.

CREATE TABLE customers (name VARCHAR(30) NOT NULL, address VARCHAR(30) NOT NULL);

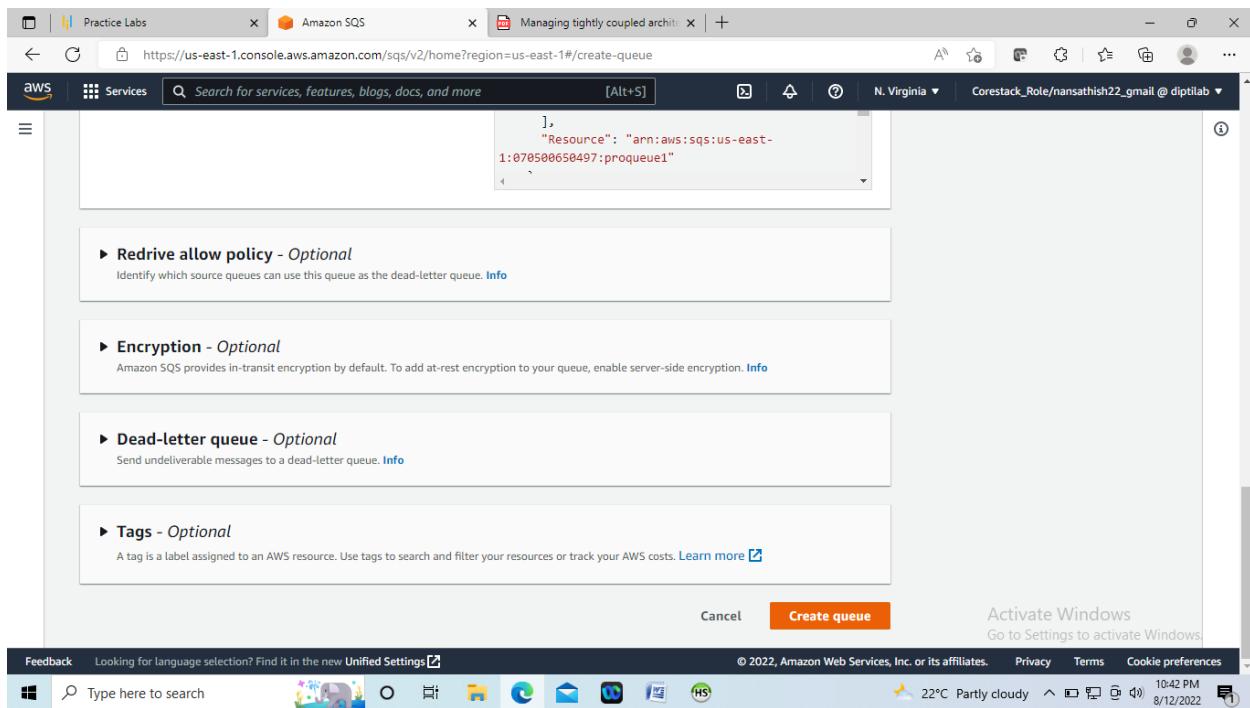


Step 17: Go to the SQS Management Console and click on Create Queue.

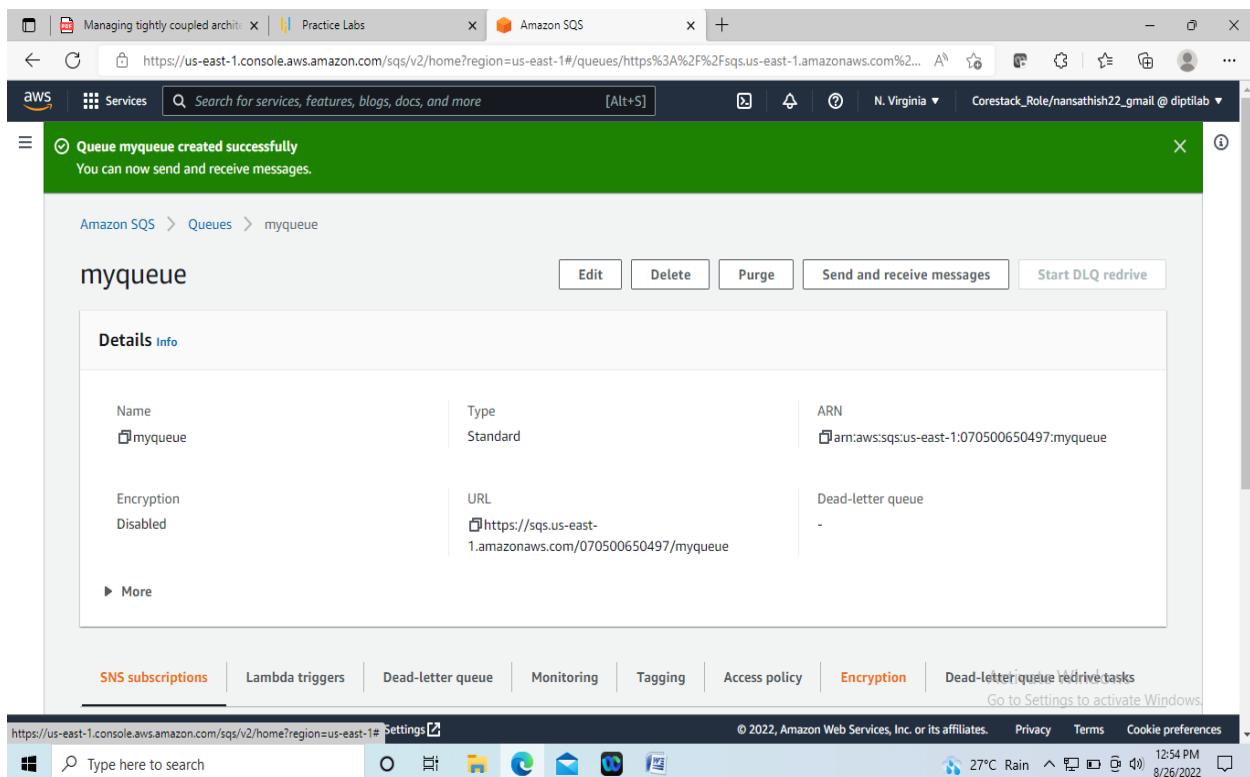
The screenshot shows the Amazon SQS Management Console homepage. At the top, there are three tabs: 'Practice Labs', 'Amazon SQS', and 'Managing tightly coupled archit...'. The URL in the address bar is https://us-east-1.console.aws.amazon.com/sqs/v2/home?region=us-east-1#. The top navigation bar includes 'Services' and a search bar. On the left, under 'Application integration', there's a section titled 'Amazon SQS' with the sub-section 'A message queuing service'. It describes how SQS provides queues for high-throughput, system-to-system messaging. Below this is a diagram titled 'How it works' showing a flow from 'Producers' to 'Consumers' through a queue. To the right, there's a 'Get started' box with a 'Create queue' button, and a 'Pricing (US)' section. The bottom of the page has a footer with links for 'Feedback', 'Unified Settings', 'Privacy', 'Terms', 'Cookie preferences', and weather information ('22°C Partly cloudy').

Step 18: Enter the Queue name as myqueue, proceed with the default options and click on Create queue.

The screenshot shows the 'Create Queue' wizard, Step 1: Set queue type and configuration. It has two main sections: 'Standard Info' and 'FIFO Info'. Under 'Standard Info', the radio button is selected and shows: 'At-least-once delivery, message ordering isn't preserved' with options 'At-least once delivery' and 'Best-effort ordering'. Under 'FIFO Info', the radio button is unselected and shows: 'First-in-first-out delivery, message ordering is preserved' with options 'First-in-first-out delivery' and 'Exactly-once processing'. Below these is a 'Name' field containing 'myqueue', with a note: 'A queue name is case-sensitive and can have up to 80 characters. You can use alphanumeric characters, hyphens (-), and underscores (_).'. A 'Configuration' section follows, containing 'Visibility timeout' (30 seconds), 'Message retention period' (4 days), 'Delivery delay' (0 seconds), and 'Maximum message size' (256 KB). The bottom of the page has a footer with links for 'Feedback', 'Unified Settings', 'Privacy', 'Terms', 'Cookie preferences', and weather information ('27°C Rain').



Step 19: Note down the Queue URL and the ARN.



Step 20: Go to the IAM Management Console and go to Policies. Click on Create Policy.

The screenshot shows the AWS IAM Management Console. On the left, there's a navigation sidebar with 'Identity and Access Management (IAM)' selected. Under 'Access management', 'Policies' is also selected. The main area shows a table of policies with columns for 'Policy name', 'Type', 'Used as', and 'Description'. One policy is highlighted: 'claas_policy_active' (Customer managed, Permissions policy). At the top right, there's a 'Create policy' button.

Step 21: Click on the JSON tab and paste the below JSON. Make sure to replace the Queue ARN with the one you obtained from the previous step. Click on Review Policy.

```
{  
  "Version": "2012-10-17", "Statement": [  
    {"Sid": "SQSSendMessage",  
     "Effect": "Allow",  
     "Action": "sns:SendMessage",  
     "Resource": "arn:aws:sns:us-east-1:304000509264:myqueue"  
  ]}
```

The screenshot shows the 'Create policy' wizard. It has three tabs at the top: 'Visual editor' (selected), 'JSON' (highlighted with a yellow background), and 'Import managed policy'. The JSON code from the previous step is pasted into the editor. A message at the bottom says 'You need permissions' because the user lacks the required permissions to perform the operation.

Step 22: Enter the policy name as SQS_SendMessage_Policy and click on Create policy.

Screenshot of the AWS IAM Management Console showing the creation of a new policy named "SQS_SendMessage_Policy".

Name: SQS_SendMessage_Policy

Description: (Empty)

Summary:

Service	Access level	Resource	Request condition
SQS	Limited: Write	QueueName string like CustomerQueue	None

Tags:

Key	Value

Feedback: Looking for language selection? Find it in the new Unified Settings.

Policies (962) Info: A policy is an object in AWS that defines permissions.

Policy name	Type	Used as	Description
claas_policy_active	Customer managed	Permissions policy (2)	
SQS_SendMessage_Policy	Customer managed	None	
STS	Customer managed	Permissions policy (2)	
AWSDirectConnectReadOnlyAccess	AWS managed	None	Provides read or
AmazonGlacierReadOnlyAccess	AWS managed	None	Provides read or

Feedback: Looking for language selection? Find it in the new Unified Settings.

Step 23: Click on Roles and click on Create Role.

The screenshot shows the AWS IAM Management Console with the URL <https://us-east-1.console.aws.amazon.com/iamv2/home#/roles>. The left sidebar is collapsed, and the main area shows the 'Roles' page with 12 entries. The first entry is 'AWS ServiceRoleForAmazonElasticFileSystem'. The 'Create role' button is visible at the top right. The status bar at the bottom indicates 'Activate Windows' and the date '8/12/2022'.

Step 24: Select EC2 as the service and click on Next Permissions.

The screenshot shows the 'Trusted entity type' step of the IAM role creation wizard. It has five sections: 'Step 2' (Add permissions), 'Step 3' (Name, review, and create), and three tabs: 'Common use cases' (selected), 'Use cases for other AWS services', and 'Choose a service to view use case'. Under 'Common use cases', 'EC2' is selected. The status bar at the bottom indicates 'Activate Windows' and the date '8/12/2022'.

Step 25: Select the policy created in the previous step and click on Next Tags.

The screenshot shows the AWS IAM Management Console with the URL https://us-east-1.console.aws.amazon.com/iamv2/home#/roles/create?commonUseCase=EC2&step=addPermission&trustedEntityType=AWS_SE.... The page title is "Add permissions". On the left, there are three steps: Step 1 (Select trusted entity), Step 2 (Add permissions, which is currently active), and Step 3 (Name, review, and create). The main area is titled "Permissions policies (Selected 1/760)" and contains a table with the following data:

Policy name	Type	Description
<input type="checkbox"/> <input checked="" type="checkbox"/> SQS_SendMessage_P...	Custom...	
<input type="checkbox"/> <input type="checkbox"/> STS	Custom...	
<input type="checkbox"/> <input type="checkbox"/> AWSDirectConnect...	AWS m...	Provides read only access to AWS Direct Connect via the AWS Managem...
<input type="checkbox"/> <input type="checkbox"/> AmazonGlacierRea...	AWS m...	Provides read only access to Amazon Glacier via the AWS Management C...
<input type="checkbox"/> <input type="checkbox"/> AWSMarketplaceFu...	AWS m...	Provides the ability to subscribe and unsubscribe to AWS Marketplace Soft...

At the bottom right of the table, there is a link "Activate Windows" and a note "Go to Settings to activate Windows." The status bar at the bottom shows "Feedback", "Type here to search", "22°C Partly cloudy", "11:03 PM", and "8/12/2022".

Step 26: Tags are optional. Click on Next Review.

Step 27: Enter the Role name as SQS_SendMessage_Role and click on Create Role.

The screenshot shows the AWS IAM Management Console with the URL https://us-east-1.console.aws.amazon.com/iamv2/home#/roles/create?commonUseCase=EC2&step=review&trustedEntityType=AWS_SE.... The page title is "Name, review, and create". On the left, there are three steps: Step 1 (Select trusted entity), Step 2 (Add permissions), and Step 3 (Name, review, and create, which is currently active). The main area is titled "Role details" and contains the following fields:

- Role name:** SQS_SendMessage_Role
- Description:** Allows EC2 instances to call AWS services on your behalf.

Below these fields, there is a section titled "Step 1: Select trusted entities" with an "Edit" button. The status bar at the bottom shows "Feedback", "Type here to search", "22°C Partly cloudy", "11:04 PM", and "8/12/2022".

The screenshot shows the AWS IAM Management Console. The left sidebar has 'Identity and Access Management (IAM)' selected. Under 'Access management', 'Roles' is selected. The main area shows a table of roles:

Role name	Trusted entities	Last activity
AWSServiceRoleForAmazonElasticFileSy	AWS Service: elasticfilesystem (Service-Linked)	-
AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Linked Role)	⚠ 390 days ago
AWSServiceRoleForBackup	AWS Service: backup (Service-Linked Role)	12 hours ago
AWSServiceRoleForCloudTrail	AWS Service: cloudtrail (Service-Linked Role)	-
AWSServiceRoleForElasticLoadBalancing	AWS Service: elasticloadbalancing (Service-Linked Role)	⚠ 390 days ago
AWSServiceRoleForOrganizations	AWS Service: organizations (Service-Linked Role)	-

Step 28: Create another Policy called SQS_ReceiveDeleteMessage_Policy with the below JSON.

Make sure to replace the SQS Queue ARN. Create a role called SQS_ReceiveDeleteMessage_Role and attach the policy to it.

```
{
"Version": "2012-10-17",
"Statement": [{ "Sid": "SQSReceiveDeleteMessage",
"Effect": "Allow",
"Action": ["sns:ReceiveMessage" ,
"sns:DeleteMessage"],
"Resource": "arn:aws:sns:us-east-1:304000509264:myqueue"
}] }
```

The screenshot shows the 'Create policy' wizard in the AWS IAM Management Console. The steps are:

- Visual editor (selected)
- Import managed policy
- Summary

The visual editor shows the JSON code:

```

1- {
2- "Version": "2012-10-17", "Statement": [
3- {"Sid": "SQSReceiveDeleteMessage",
4- "Effect": "Allow",
5- "Action": ["sns:ReceiveMessage" , "sns:DeleteMessage"],
6- "Resource": "arn:aws:sns:us-east-1:304000509264:myqueue"
7- } ]

```

A red box highlights the error message at the bottom:

You need permissions
You do not have the permission required to perform this operation. Ask your administrator to add permissions.

The screenshot shows the AWS IAM Management Console interface. On the left, there's a navigation sidebar with sections like 'Identity and Access Management (IAM)', 'Access management', 'Access reports', and 'Policies'. The 'Policies' section is currently selected. The main content area displays a table of policies with columns for 'Policy name', 'Type', 'Used as', and 'Description'. A blue banner at the top of the main area says 'Introducing the new Policies list experience'.

This screenshot shows the 'Create policy' wizard in the AWS IAM Management Console. It's on step 3, 'Review policy'. The 'Name*' field contains 'SQS_ReceiveDeleteMessage_Policy'. Below it, there's a note about character restrictions. The 'Description' field is empty. The 'Summary' section shows a table with columns 'Service', 'Access level', 'Resource', and 'Request condition'. One row is visible for SQS, allowing 'Read, Write' access to 'QueueName | string like | CustomerQueue' with 'None' request conditions. The status bar at the bottom indicates '11:06 PM 22°C Partly cloudy 8/12/2022'.

Maximum 1000 characters. Use alphanumeric and '+-_.' characters.

Service	Access level	Resource	Request condition
SQS	Limited: Read, Write	QueueName string like CustomerQueue	None

No tags associated with the resource.

*** Required**

Feedback Looking for language selection? Find it in the new Unified Settings 

© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 22°C Partly cloudy 11:08 PM 8/12/2022 

Type here to search  Practice Labs  IAM Management Console  IAM Management Console  Managing tightly coupled archi  +     

https://us-east-1.console.aws.amazon.com/iamv2/home#/policies

aws Services Search for services, features, blogs, docs, and more [Alt+S]    Global Corestack_Role/nansathish22_gmail @ diptilab 

Identity and Access Management (IAM)

- Dashboard
- Access management**
 - User groups
 - Users
 - Roles
 - Policies**
 - Identity providers
 - Account settings
- Access reports**
 - Access analyzer
 - Archive rules
 - Analyzers
 - Settings
 - Credential report
 - Organization activity
 - Service control policies (SCPs)

Introducing the new Policies list experience
We've redesigned the Policies list experience to make it easier to use. [Let us know what you think.](#)

The policy SQS_ReceiveDeleteMessage_Policy has been created.

IAM > Policies

Policies (963) Info
A policy is an object in AWS that defines permissions.

Create policy  Actions 

Policy name	Type	Used as	Description
claas_policy_active	Customer managed	Permissions policy (2)	
SQS_ReceiveDeleteMessage_Policy	Customer managed	None	
SQS_SendMessage_Policy	Customer managed	Permissions policy (1)	
STS	Customer managed	Permissions policy (2)	
AWSDirectConnectReadOnlyAccess	AWS managed	None	Provides read or write access to Direct Connect resources.

Feedback Looking for language selection? Find it in the new Unified Settings 

© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 22°C Partly cloudy 11:09 PM 8/12/2022  

Type here to search                  

Screenshot of the AWS IAM Management Console showing the selection of policies for a new role. The selected policy is 'SQS_ReceiveDeleteMessage_Role'.

Step 3: Name, review, and create

Policy name: SQS_ReceiveDeleteMessage_Role

Type: Custom...

Description: Allows EC2 instances to call AWS services on your behalf.

Other policies listed:

- sts
- AWSDirectConnect...
- AmazonGlacierRea...
- AWSMarketplaceFu...
- AWSSSOdirectory...
- AWSIoT1ClickRead...
- AutoScalingConsol...
- AmazonDMSRedsh...
- AWSQuickSightList...

Feedback: Looking for language selection? Find it in the new Unified Settings.

© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

22°C Partly cloudy 11:09 PM 8/12/2022

Screenshot of the AWS IAM Management Console showing the 'Role details' step of creating a new role.

Step 2: Add permissions

Step 3: Name, review, and create

Role details

Role name: SQS_ReceiveDeleteMessage_Role

Description: Allows EC2 instances to call AWS services on your behalf.

Step 1: Select trusted entities

JSON Policy Document:

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

Activate Windows
Go to Settings to activate Windows

Feedback: Looking for language selection? Find it in the new Unified Settings.

© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

22°C Partly cloudy 11:10 PM 8/12/2022

The screenshot shows the AWS IAM Management Console. On the left, the navigation menu includes 'Identity and Access Management (IAM)', 'Access management' (with 'Roles' selected), 'Policies', 'Identity providers', 'Account settings', 'Access reports' (with 'Analyzer' selected), 'Credential report', 'Organization activity', and 'Service control policies (SCPs)'. The main content area displays a message about securely accessing AWS services with IAM Roles Anywhere, followed by a table titled 'Roles (14)'. The table lists the following roles:

Role name	Trusted entities	Last activity
AWSServiceRoleForAmazonElasticFileSys	AWS Service: elasticfilesystem (Service-Linked)	-
AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Linked Role)	390 days ago
AWSServiceRoleForBackup	AWS Service: backup (Service-Linked Role)	12 hours ago
AWSServiceRoleForCloudTrail	AWS Service: cloudtrail (Service-Linked Role)	-
AWSServiceRoleForElasticLoadBalancing	AWS Service: elasticloadbalancing (Service-Linked Role)	390 days ago

Step 29: At the end, two policies and two roles should be created, as shown below.

The screenshot shows the AWS IAM Management Console. The navigation menu is identical to the previous screenshot. The main content area displays a message about the new Policies list experience, followed by a table titled 'Policies (963)'. The table lists the following policies:

Policy name	Type	Used as	Description
claas_policy_active	Customer managed	Permissions policy (2)	
SQS_ReceiveDeleteMessage_Policy	Customer managed	Permissions policy (1)	
SQS_SendMessage_Policy	Customer managed	Permissions policy (1)	
STS	Customer managed	Permissions policy (2)	
AWSDirectConnectReadOnlyAccess	AWS managed	None	Provides read-only access to Direct Connect resources.
AmazonGlacierReadOnlyAccess	AWS managed	None	Provides read-only access to Glacier resources.

Identity and Access Management (IAM)

- Dashboard
- Access management**
 - User groups
 - Users
 - Roles**
 - Policies
 - Identity providers
 - Account settings
- Access reports**
 - Access analyzer
 - Archive rules
 - Analyzers
 - Settings
 - Credential report
 - Organization activity
 - Service control policies (SCPs)

Roles Anywhere Info

Authenticate your non AWS workloads and securely provide access to AWS services.

	X.509 Standard	
Access AWS from your non AWS workloads	Use your own existing PKI infrastructure or use AWS Certificate Manager Private Certificate Authority to authenticate identities.	Temporary credentials
Operate your non AWS workloads using the same authentication and authorization strategy that you use within AWS.		Use temporary credentials with ease and benefit from the enhanced security they provide.

Manage

Feedback Looking for language selection? Find it in the new Unified Settings Z

© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Activate Windows Go to Settings to activate Windows. V

Step 30: Create two EC2 Instances and name them as CustomerWebApplication and BackendApplication with the below details.

Make sure to select the KeyPair. –

Amazon Linux 2 Kernel 5.10 AMI 2.0.20220805.0 x86_64 HVM gp2 - t2.micro - SecurityGroup with 22/SSH inbound

Create key pair | EC2 Management

Name: prokp

Key pair type: RSA

Private key file format: .pem

Tags - optional

Add new tag

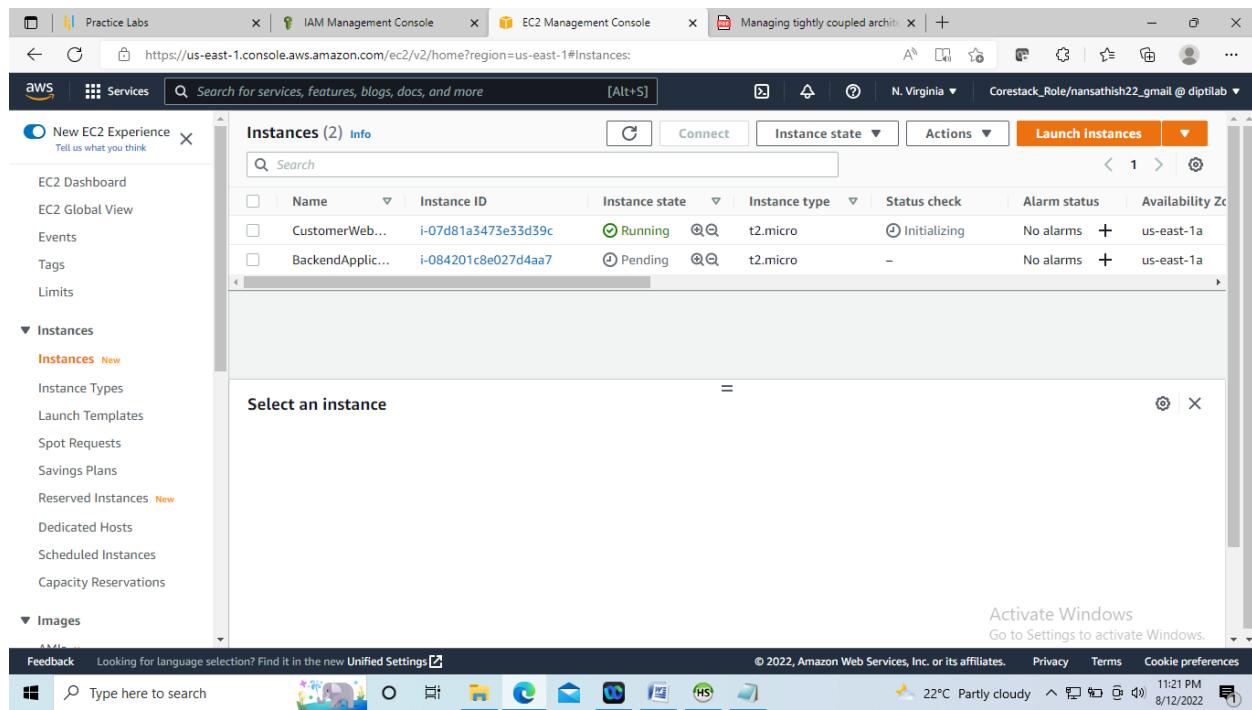
Create key pair

Activate Windows
Go to Settings to activate Windows. V

Feedback Looking for language selection? Find it in the new Unified Settings Z

© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

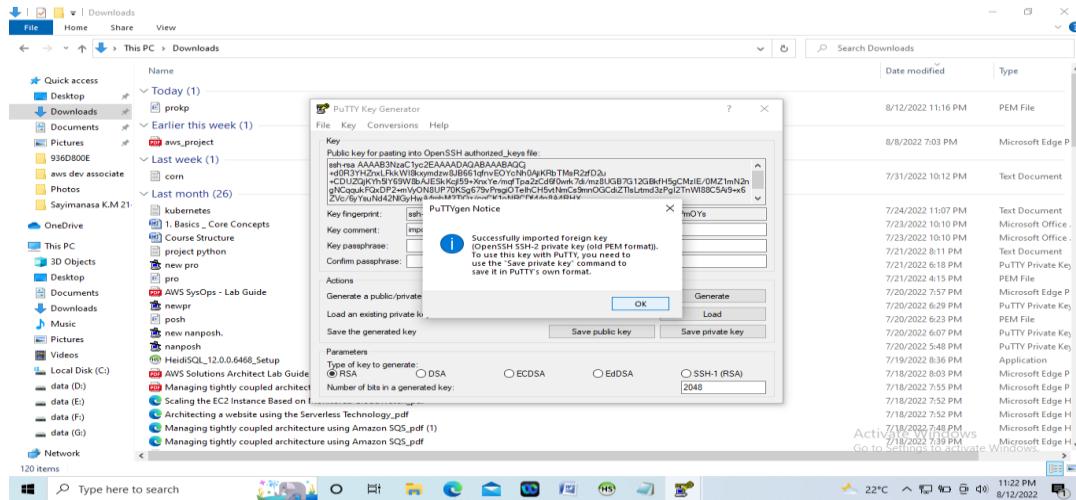
22°C Partly cloudy 11:16 PM 8/12/2022



Step 31: Login to the EC2 instances via PuTTY. Execute the below commands on both the EC2 instances. See the exception.

Step 32: Attach the SQS_SendMessage_Role Role to the CustomerWebApplication EC2 Instance. Attach the SQS_ReceiveDeleteMessage_Role Role to the BackendApplication EC2 Instance.

Generate private key using puttykeygen



#become root

```
sudo su -  
#get the list of softwares apt-get update  
#install python  
Sudo yum update -y
```

```

Sudo yum install python3 -y
#create env and activate
Python3 -m venv python_env
Source python_env/bin/activate
#install python lib
Pip install - --upgrade pip && pip install requests
#To execute pip commands apt install python-pip
#install Python AWS SDK and MySQL drivers pip install boto3 pip install mysql-connector-python (only
on the "BackendApplication" EC2 Instance) exit
mkdir .aws
echo -e "[default]\nregion=us-east-1" > .aws/config
Check :
Python
Import boto3
Import botocore
Exit()

```

```

ec2-user@ip-172-31-31-8:~ [6/7]: kernel-tools-5.10.135-122.509.amzn2.x86_64.rpm      | 176 kB  00:00
[7/7]: kernel-5.10.135-122.509.amzn2.x86_64.rpm          | 32 MB   00:00
Total                                         46 MB/s | 34 MB  00:00
-----  

Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Updating : 12:dhcp-libs-4.2.5-79.amzn2.1.1.x86_64           1/13
Updating : 12:dhcp-common-4.2.5-79.amzn2.1.1.x86_64         2/13
Updating : 12:dhclient-4.2.5-79.amzn2.1.1.x86_64          3/13
Updating : kernel-tools-5.10.135-122.509.amzn2.x86_64       4/13
Updating : chrony-4.2-5.amzn2.0.1.x86_64                  5/13
Installing : kernel-5.10.135-122.509.amzn2.x86_64          6/13
Updating : gnupg2-2.0.22-5.amzn2.0.5.x86_64                7/13
Updating : 12:dhclient-4.2.5-77.amzn2.1.6.x86_64          8/13
Cleanup   : 12:dhcp-common-4.2.5-77.amzn2.1.6.x86_64        9/13
Cleanup   : 12:dhcp-libs-4.2.5-77.amzn2.1.6.x86_64         10/13
Cleanup   : kernel-tools-5.10.130-118.517.amzn2.x86_64      11/13
Cleanup   : chrony-4.0-3.amzn2.0.2.x86_64                 12/13
Cleanup   : gnupg2-2.0.22-5.amzn2.0.4.x86_64              13/13
Verifying : 12:dhcp-libs-4.2.5-79.amzn2.1.1.x86_64          1/13
Verifying : 12:dhclient-4.2.5-79.amzn2.1.1.x86_64          2/13
Verifying : gnupg2-2.0.22-5.amzn2.0.5.x86_64              3/13
Verifying : 12:dhcp-common-4.2.5-79.amzn2.1.1.x86_64          4/13
Verifying : kernel-5.10.135-122.509.amzn2.x86_64          5/13
Verifying : chrony-4.2-5.amzn2.0.1.x86_64                 6/13
Verifying : kernel-tools-5.10.135-122.509.amzn2.x86_64       7/13
Verifying : gnupg2-2.0.22-5.amzn2.0.4.x86_64              8/13
Verifying : 12:dhcp-libs-4.2.5-77.amzn2.1.6.x86_64          9/13
Verifying : 12:dhclient-5.10.130-118.517.amzn2.x86_64      10/13
Verifying : 12:dhcp-common-4.2.5-77.amzn2.1.6.x86_64         11/13
Verifying : chrony-4.0-3.amzn2.0.2.x86_64                 12/13
Verifying : kernel.x86_64 0:5.10.135-122.509.amzn2.x86_64      13/13

Installed:
  kernel.x86_64 0:5.10.135-122.509.amzn2.x86_64

Updated:
  chrony.x86_64 0:4.2-5.amzn2.0.1      dhclient.x86_64 12:4.2.5-79.amzn2.1.1      dhcp-common.x86_64 12:4.2.5-79.amzn2.1.1      dhcp-libs.x86_64 12:4.2.5-79.amzn2.1.1
  gnupg2.x86_64 0:2.0.22-5.amzn2.0.5    kernel-tools.x86_64 0:5.10.135-122.509.amzn2.x86_64

Complete!
[ec2-user@ip-172-31-31-8 ~]$
```

The terminal window shows the progress of a yum update command. It lists the packages being updated, their versions, and the total download speed. After the update is complete, it shows the installed packages and the updated packages. The bottom of the window shows the Windows taskbar with icons for File Explorer, Task View, Start, Taskbar Help, and a system tray icon.

```
ec2-user@ip-172-31-31-8:~
```

```
Installing collected packages: urllib3, idna, charset-normalizer, certifi, requests, certifi, charset-normalizer, idna-3.3, requests
Successfully installed certifi-2022.6.15 charset-normalizer-2.1.1 idna-3.3 requests-2.28.1 urllib3-1.26.12
(python env) [ec2-user@ip-172-31-31-8 ~]$ python
Python 3.7.10 (default, Jun 3 2021, 00:02:01)
[GCC 7.3.1 20180712 (Red Hat 7.3.1-13)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import boto3
Exception: No module named 'boto3'
File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'boto3'
>>> exit()
(python env) [ec2-user@ip-172-31-31-8 ~]$ pip install boto3
Collecting boto3
  Downloading boto3-1.24.58-py3-none-any.whl (132 kB)
Collecting botocore<1.28.0,>=1.27.58
  Downloading botocore-1.27.58-py3-none-any.whl (9.1 MB)
    0.1/9.1 MB 62.6 MB/s eta 0:00:00
Collecting s3transfer<0.7.0,>=0.6.0
  Downloading s3transfer-0.6.0-py3-none-any.whl (79 kB)
    79.6/79.6 kB 10.3 MB/s eta 0:00:00
Collecting jmespath<2.0.0,>=0.7.1
  Downloading jmespath-1.0.1-py3-none-any.whl (20 kB)
Collecting python-dateutil<3.0.0,>=2.1
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
    247.7/247.7 kB 32.8 MB/s eta 0:00:00
Requirement already satisfied: urllib3<1.27,>=1.25.4 in ./python_env/lib/python3.7/site-packages (from botocore<1.28.0,>=1.27.58->boto3) (1.26.12)
Collecting six<1.5
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, jmespath, python-dateutil, botocore, s3transfer, boto3
Successfully installed boto3-1.24.58 botocore-1.27.58 jmespath-1.0.1 python-dateutil-2.8.2 s3transfer-0.6.0 six-1.16.0
(python env) [ec2-user@ip-172-31-31-8 ~]$ python
Python 3.7.10 (default, Jun 3 2021, 00:02:01)
[GCC 7.3.1 20180712 (Red Hat 7.3.1-13)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import boto3
>>> import botocore
>>>
```

Step 33: Create a file send_message.py with the below code on the CustomerWebApplication EC2 instance.

```
send_message.py
```

```
import sys

import boto3

sns = boto3.client('sns')

queue_url = 'https://sns.us-east-1.amazonaws.com/070500650497/myqueue'

response = sns.send_message( QueueUrl=queue_url, MessageBody=(sys.argv[1]) )

print(response['MessageId'])
```

Step 34: Execute the below commands to put the messages in the Queue.

```
python send_message.py nan,bangalore
```

```

ec2-user@ip-172-31-31-8:~$ Python 3.7.10 (default, Jun 3 2021, 00:02:01)
[GCC 7.3.1 20180712 (Red Hat 7.3.1-13)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import boto3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'boto3'
>>> exit()
(ec2-user@ip-172-31-31-8) pip install boto3
Collecting boto3
  Downloading boto3-1.24.58-py3-none-any.whl (132 kB)
    132.5/132.5 kB 14.4 MB/s eta 0:00:00
Collecting botocore<1.28.0,>=1.27.58
  Downloading botocore-1.27.58-py3-none-any.whl (9.1 kB)
    9.1/9.1 kB 62.6 MB/s eta 0:00:00
Collecting s3transfer<0.7.0,>=0.6.0
  Downloading s3transfer-0.6.0-py3-none-any.whl (79 kB)
    79.6/79.6 kB 10.3 MB/s eta 0:00:00
Collecting jmespath<2.0.0,>=0.7.1
  Downloading jmespath-1.0.1-py3-none-any.whl (20 kB)
Collecting python-dateutil<3.0.0,>=2.1
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
    247.7/247.7 kB 32.5 MB/s eta 0:00:00
Requirement already satisfied: urllib3<1.27,>=1.25.4 in ./python_env/lib/python3.7/site-packages (from botocore<1.28.0,>=1.27.58->boto3) (1.26.12)
Collecting six<1.5
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, jmespath, python-dateutil, botocore, s3transfer, boto3
Successfully installed boto3-1.24.58 botocore-1.27.58 jmespath-1.0.1 python-dateutil-2.8.2 s3transfer-0.6.0 six-1.16.0
(ec2-user@ip-172-31-31-8) python
Python 3.7.10 (default, Jun 3 2021, 00:02:01)
[GCC 7.3.1 20180712 (Red Hat 7.3.1-13)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import boto3
>>> import botocore
>>> exit()
(ec2-user@ip-172-31-31-8) mkdir .aws
(ec2-user@ip-172-31-31-8) echo -e "[default]\nregion=us-east-1" > .aws/config
(ec2-user@ip-172-31-31-8) vim send_message.py
(ec2-user@ip-172-31-31-8) python send_message.py nandini,bangalore
fe185152-84a9-4132-9651-d755a4a14998
(ec2-user@ip-172-31-31-8) $ 

```

Activate Windows
Go to Settings to activate Windows.

```

ec2-user@ip-172-31-31-8:~$ python send_message.py nandini,bangalore
d77B80d-a54d-e7f6-9548-3fb05e1def7
(ec2-user@ip-172-31-31-8) $ 

```

Step 35: Create a file `write_to_rds.py` with the below code on the `BackendApplication` EC2 instance. Make sure to change the `queue_url` and the database details. The below program reads the message from Queue and inserts a row in the RDS/MySQL Database.

`write_rds.py`

```

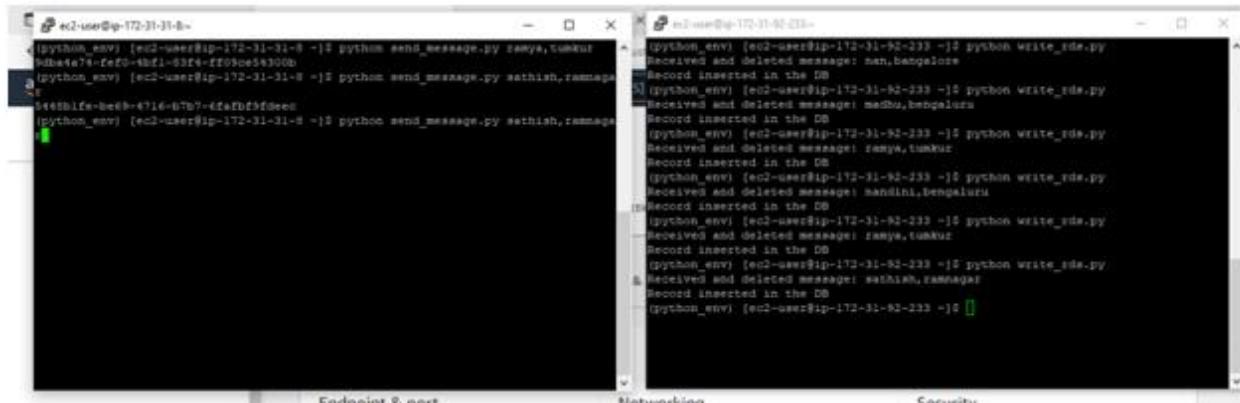
import time
import boto3
import mysql.connector
sns = boto3.client('sns')
queue_url = 'https://sns.us-east-1.amazonaws.com/070500650497/myqueue'
host = 'customerdatabase1.ciqyafyzagmn.us-east-1.rds.amazonaws.com'
user = 'Nandini'
password = 'handini1'
database='Customerdatabase1'
mydb = mysql.connector.connect(host=host, user=user, password=password, database=database)
mycursor = mydb.cursor()
response = sns.receive_message(QueueUrl=queue_url)
message = response['Messages'][0]
receipt_handle = message['ReceiptHandle']
sns.delete_message( QueueUrl=queue_url, ReceiptHandle=receipt_handle )
print('Received and deleted message: %s' % message["Body"])
customerDetails = message["Body"]

```

```
customerDetailsList = customerDetails.split(',')
name = customerDetailsList[0]
address = customerDetailsList[1]
val = (name, address)
sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
mycursor.execute(sql, val)
mydb.commit()
print("Record inserted in the DB")
```

Step 36: Wait for a few minutes and execute the python write_to_rds.py command to get the message from the Queue and insert a row in the database.

Here, we are trying to mimic the backend application being down. Notice that the messages between the web application are not lost despite waiting for a few minutes or the backend application being down. This is how highly available applications are built.



Step 37: Go back to the HeidiSQL and execute the select * from customers query to fetch the customers details.

