1 Demonstrate the Creation of an EC2 instance using the AWS management console

Amazon Elastic Compute Cloud (EC2) is a popular web service that provides resizable computing capacity in the cloud. With EC2, you can easily create and manage virtual servers, known as instances, to run your applications and workloads. In this guide, we will walk you through the process of creating an EC2 instance.

What is an Instance?

An instance in cloud computing is a server resource provided by third-party cloud services. While you can manage and maintain physical server resources on-premises, it is costly and inefficient to do so. Cloud providers maintain hardware in their data centers and give you virtual access to compute resources in the form of an instance. You can use the cloud instance for running compute-intensive workloads like containers, databases, microservices, and virtual machines.

Prerequisites

Before you begin, make sure you have the following:

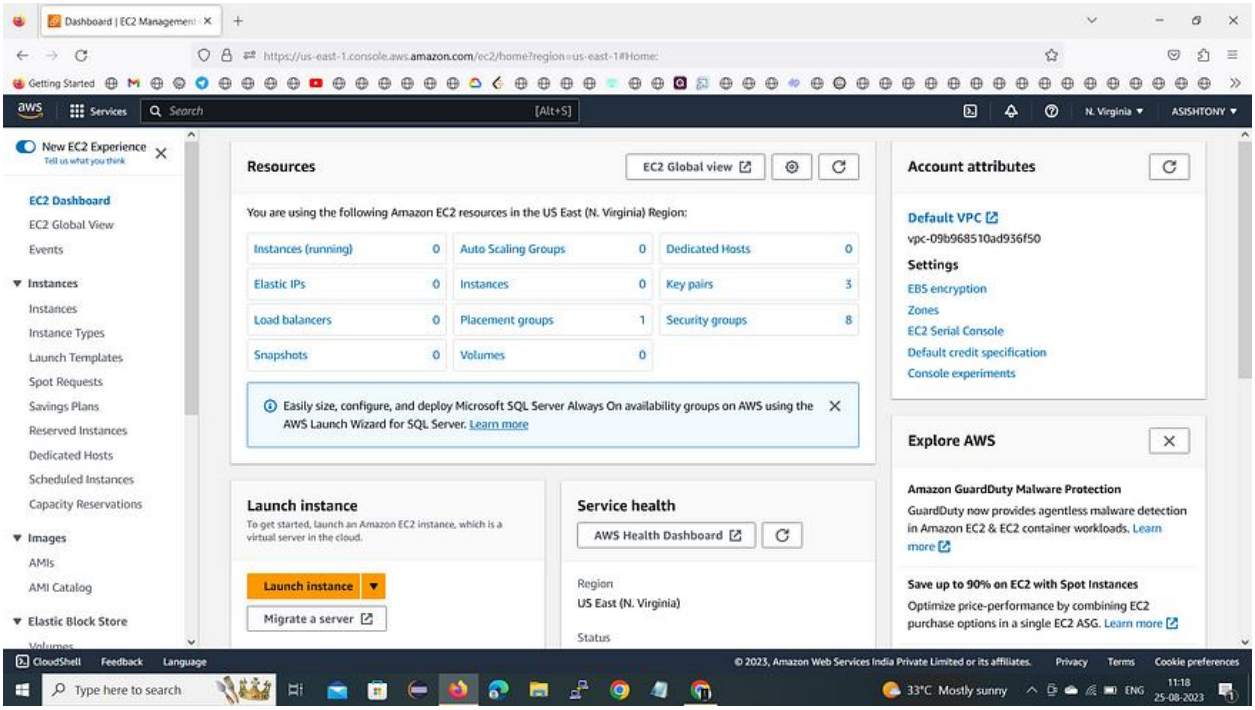
1. An Amazon Web Services (AWS) account: You'll need an active AWS account to create EC2 instances.
2. AWS Console Access: Log in to the AWS Management Console.

Step 1: Log in to the AWS Management Console

1. Open your web browser and navigate to the AWS Management Console.
2. Sign in with your AWS account credentials.

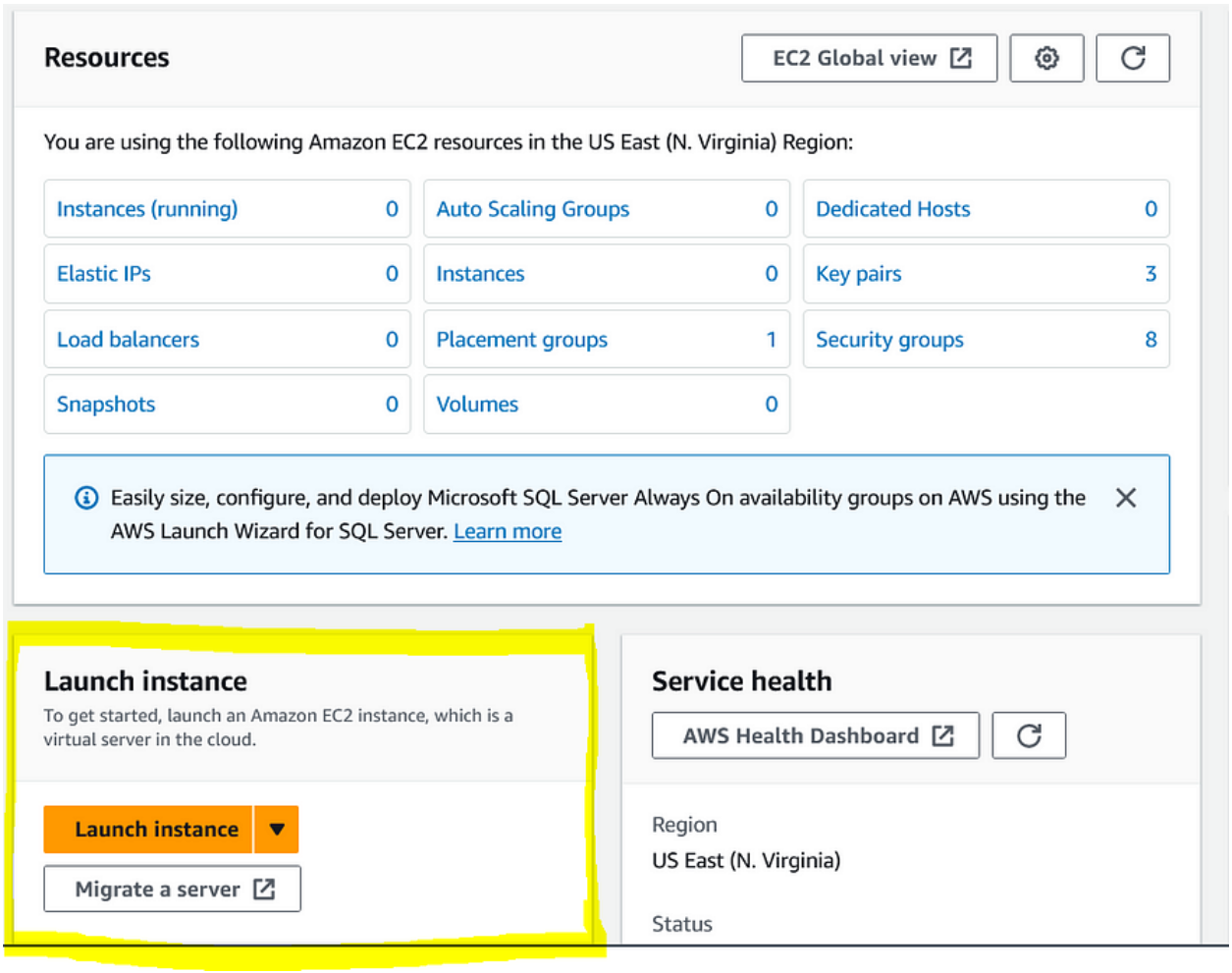
Step 2: Navigate to EC2 Dashboard

1. Once logged in, you'll be in the AWS Management Console. In the "Find Services" search bar, type "EC2" and select "EC2" from the results. This will take you to the EC2 Dashboard.

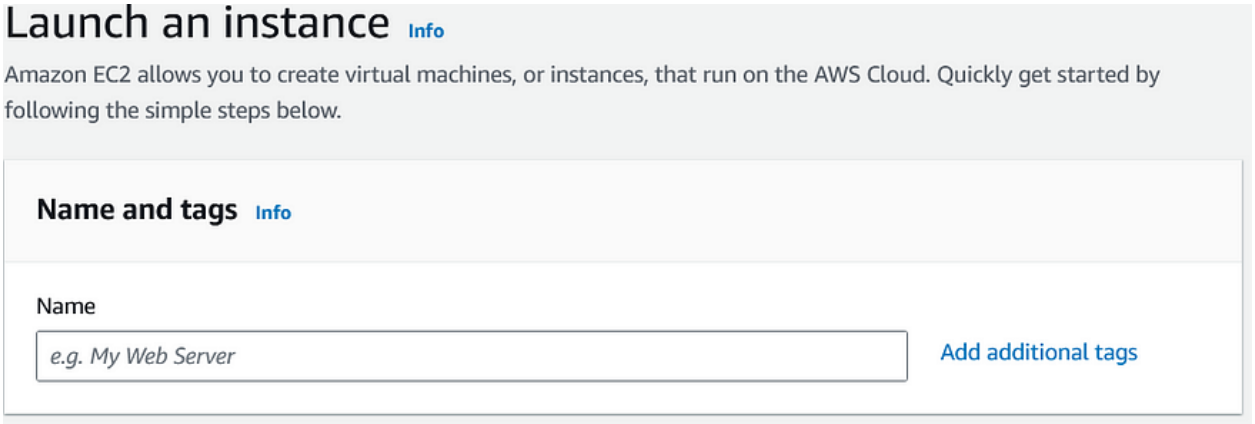


Step 3: Launch an Instance

1. On the EC2 Dashboard, click the "Launch Instance" button to begin the instance creation process.



Step 4: Name your Instance

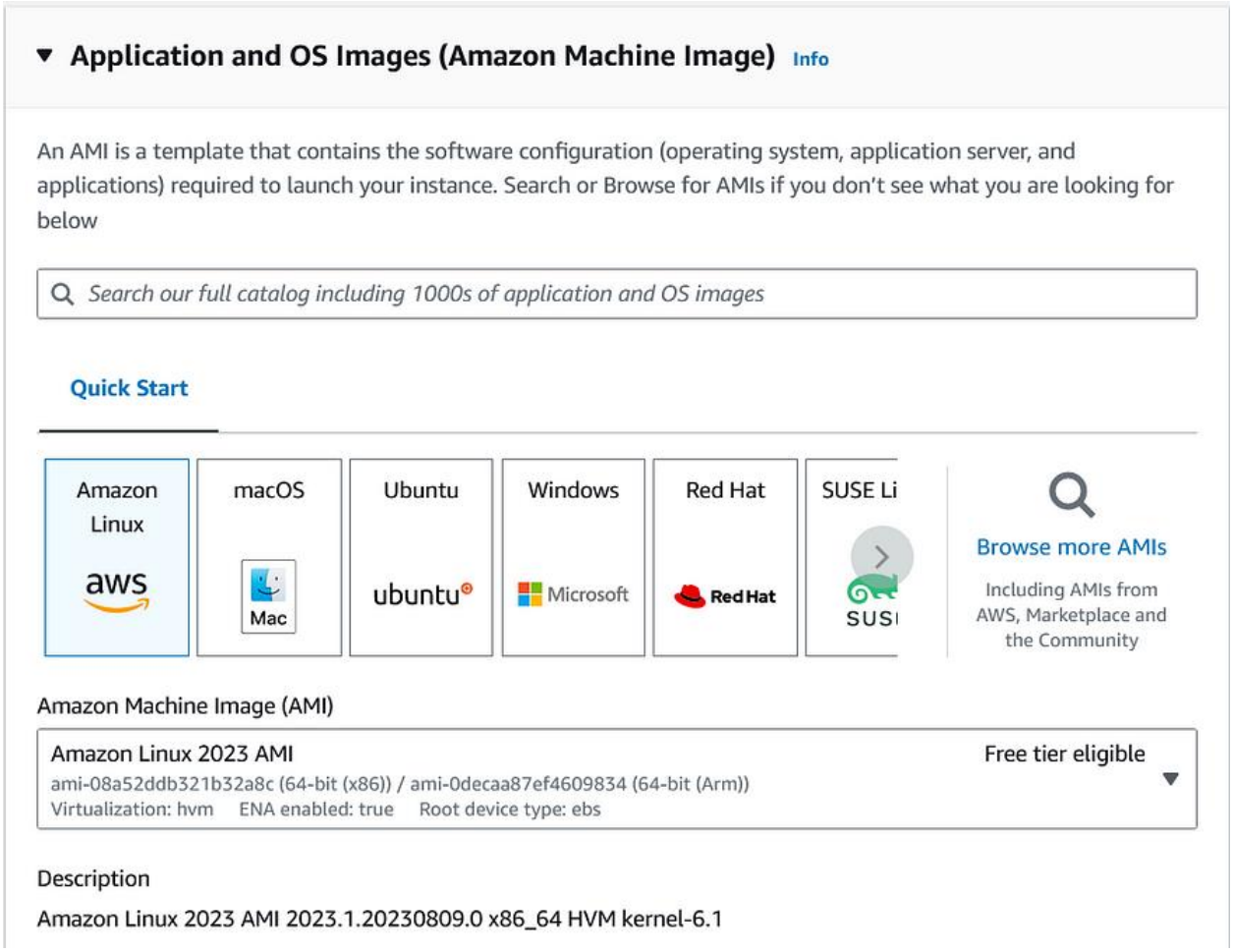


Step 5: Choose an Amazon Machine Image (AMI)

An Amazon Machine Image (AMI) is a supported and maintained image provided by AWS that provides the information required to launch an instance. You must specify an AMI when you launch an instance. You can launch multiple instances from a single AMI when you require multiple instances with the same configuration. You can use different AMIs to launch instances when you require instances with different configurations.

1. In the "Choose an Amazon Machine Image (AMI)" section, you'll need to select the operating system and software stack for your instance. AWS offers a wide range of pre-configured AMIs.

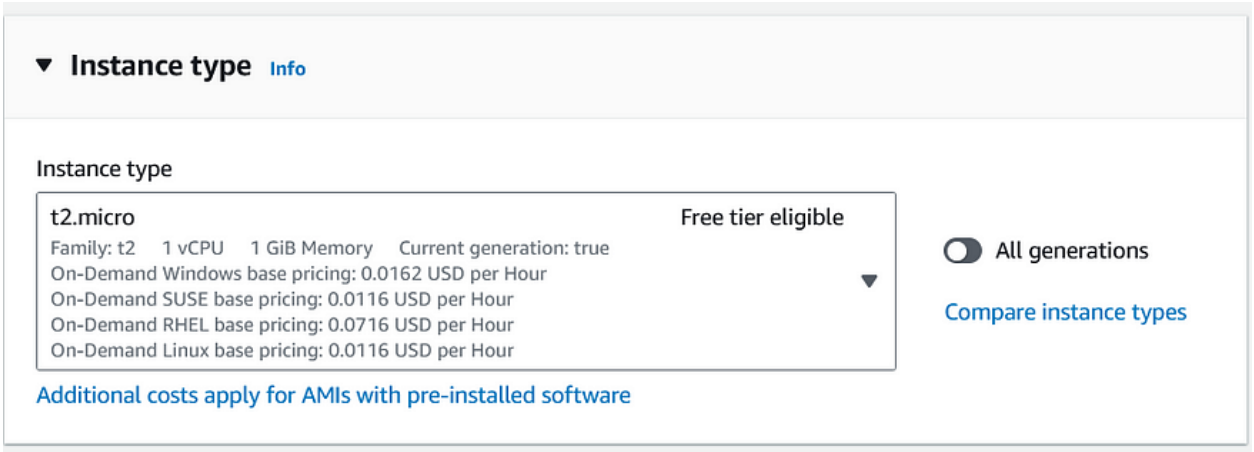
2. Choose an AMI based on your requirements, such as Amazon Linux, Ubuntu, Windows Server, etc.



Step 6: Choose an Instance Type

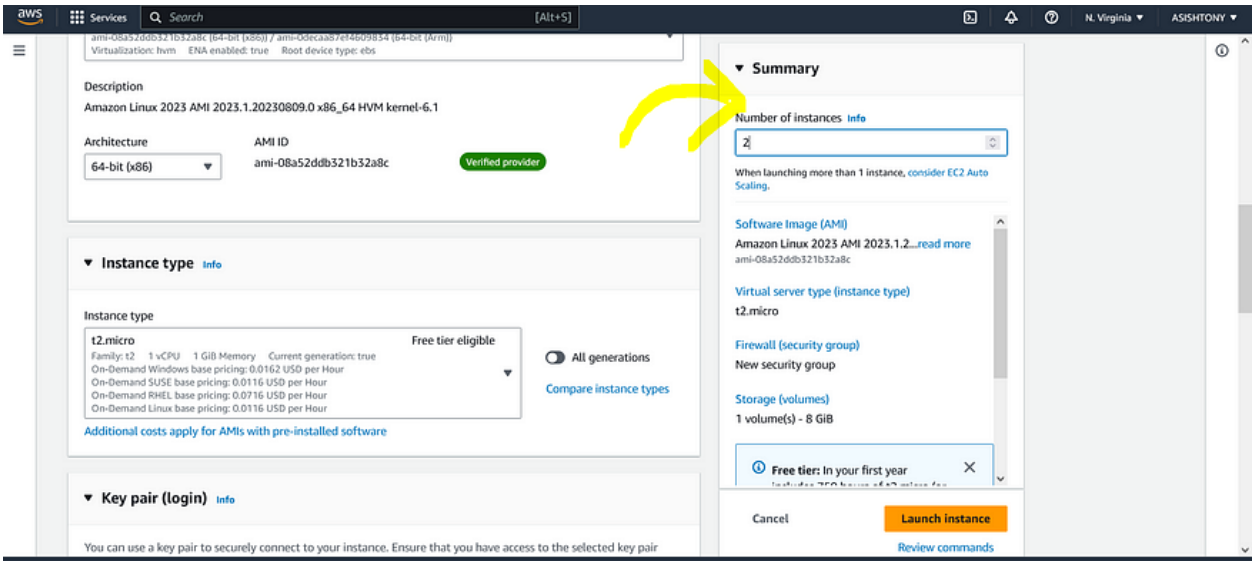
When you launch an instance, the ***instance type*** that you specify determines the hardware of the host computer used for your instance. Each instance type offers different **compute, memory, and storage capabilities**, and is grouped into an instance family based on these capabilities. Select an instance type based on the requirements of the application or software that you plan to run on your instance.

1. In the "Choose an Instance Type" section, you'll need to select the instance type based on your workload's resource requirements. Instances vary in terms of CPU, memory, storage, and networking capabilities.
2. Click on the instance type you want to use. You can find details about each instance type by clicking on the information icon.

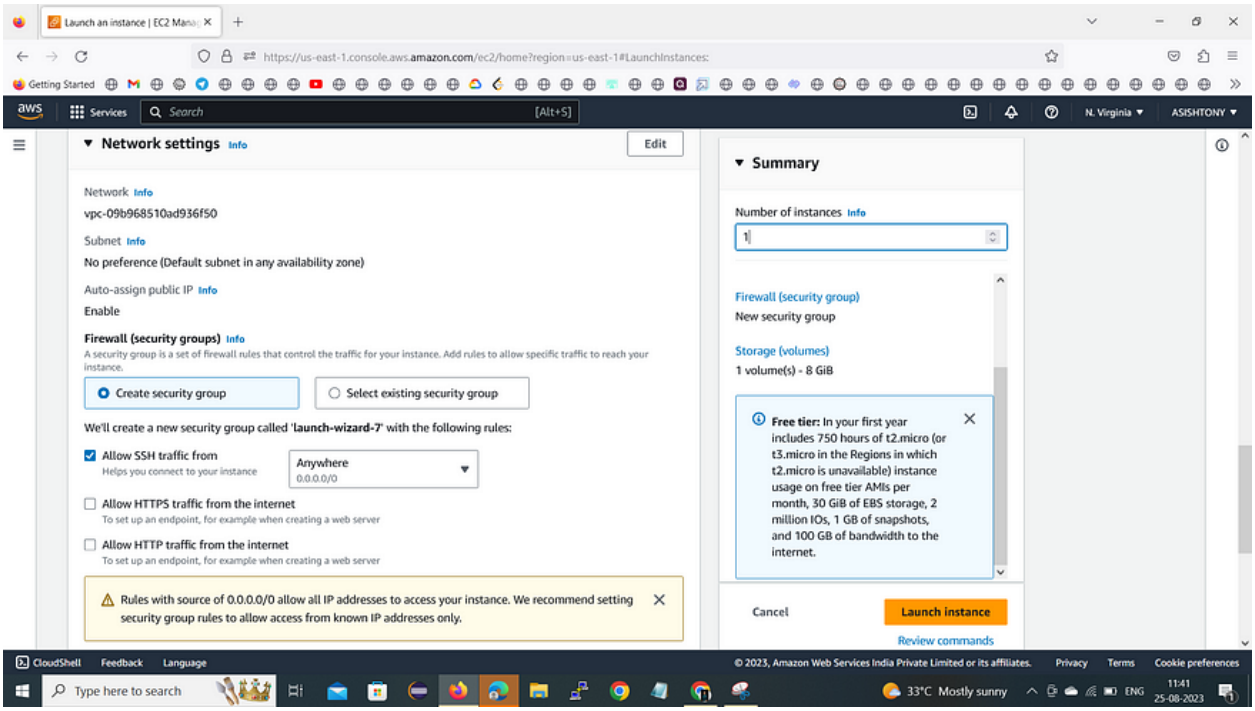


Step 7: Configure Instance Details

1. Configure the number of instances you want to launch.

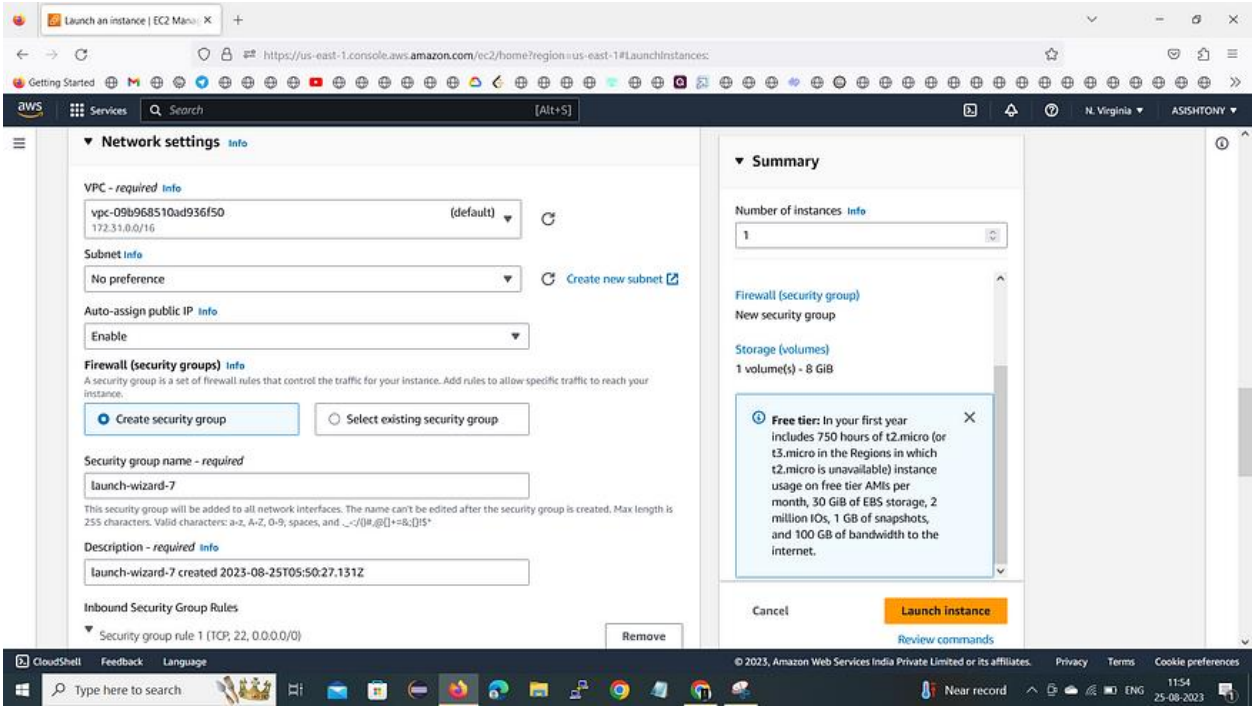


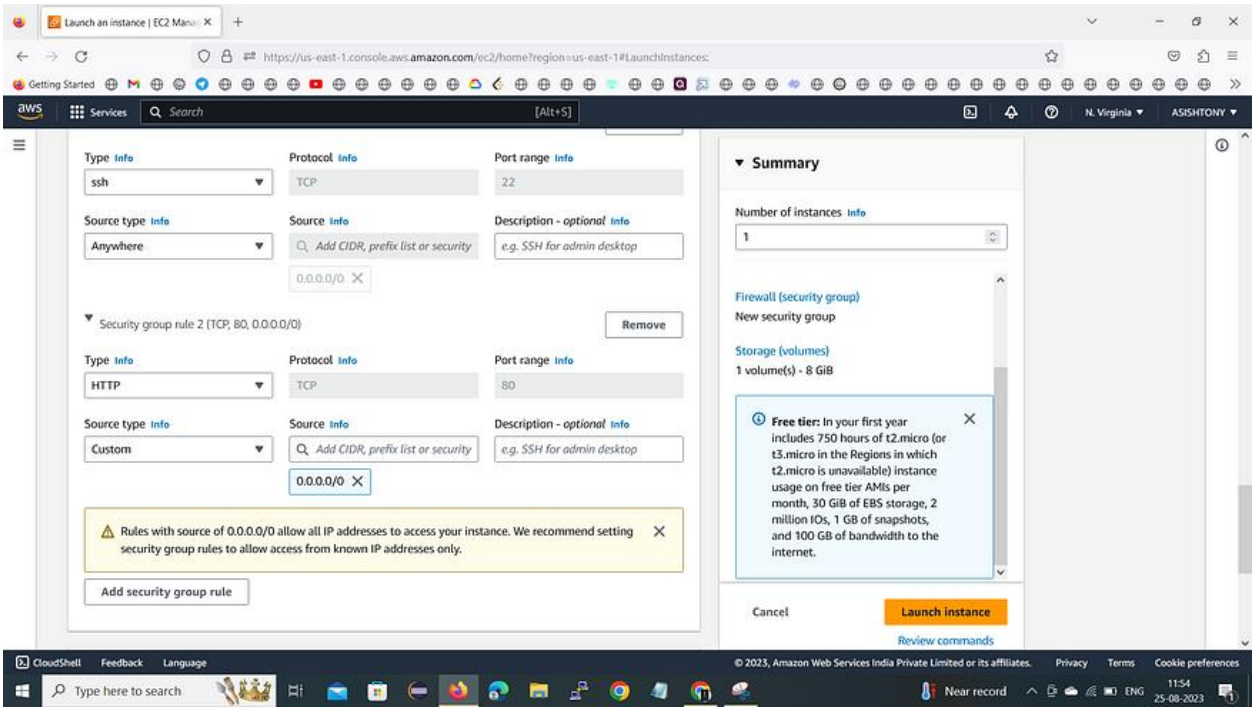
Step 8: Configure Network Settings



Network settings > edit

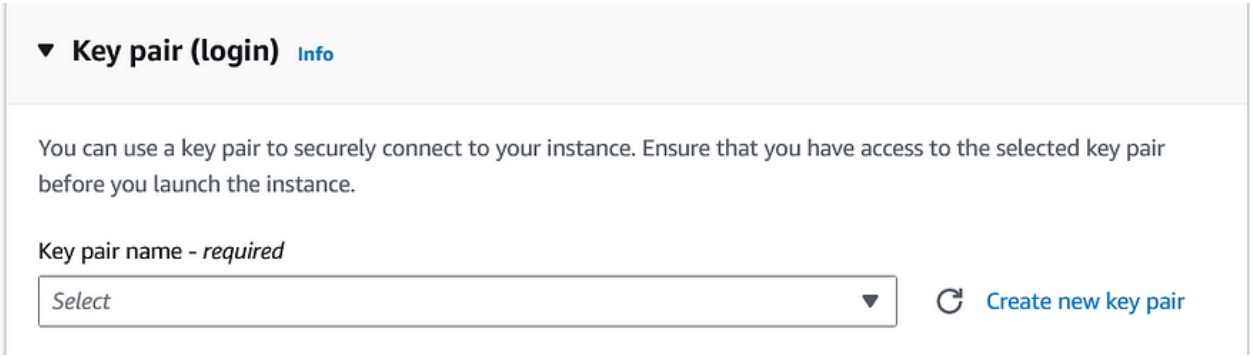
1. Choose a VPC (Virtual Private Cloud) and subnet for your instance.
2. In the “Network Settings” section, you’ll define the security rules for your instance. A security group acts as a virtual firewall, controlling inbound and outbound traffic.
3. You can create a new security group or select an existing one. Configure rules for SSH, HTTP, HTTPS, and other necessary protocols.





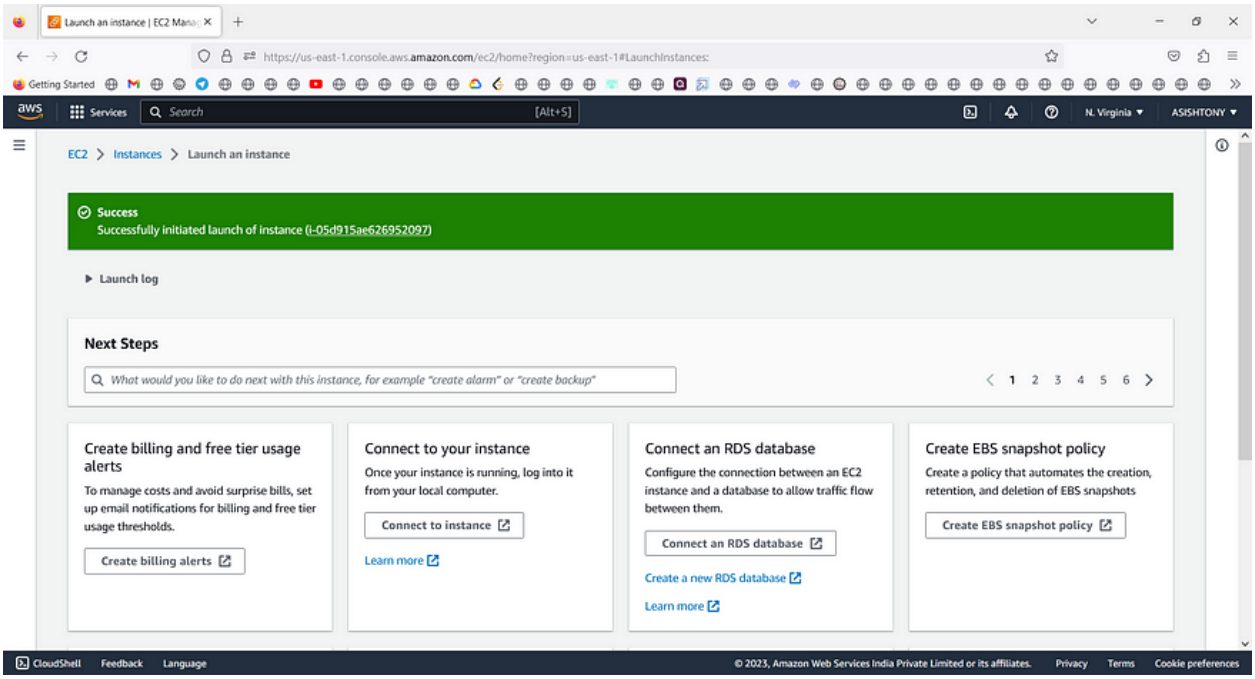
Step 9: Create a Key Pair

1. In this step, you'll need to create a key pair or use an existing one. Key pairs are used for securely connecting to your instance using SSH.
2. Select "Create a new key pair," give it a name, and download the private key file (.pem). Note: Keep this private key file secure, as it's required to access the instance.



Step 10: Review Instance Launch

1. Review all the configuration settings you've made for the instance.
2. Click the "Launch" button if everything looks correct.



Step 11: Accessing Your Instance

1. Once your instance is running, you can find its public IP address in the EC2 Dashboard under “Instances.”
2. To connect to your instance using SSH (for Linux/Unix) or Remote Desktop (for Windows), use the private key you downloaded in Step 10.

Congratulations! You’ve successfully created an Amazon EC2 instance. You can now start installing and configuring your applications on the instance to meet your specific needs.

Remember to monitor your instance’s usage and manage its security settings regularly to ensure optimal performance and data safety.

2 Demonstrate hosting a website using the EC2 instance directory.

Hosting a website on AWS EC2 is a great way to get started with web hosting. In this guide, we'll walk you through the steps you'll need to take to get your website up and running on AWS EC2. Here's what you'll need to do:

- **Set Up Your EC2 Instance:** Open your favorite web browser and sign in to your AWS Management Console. Next, search for and select EC2.
- **Connect Your Apache Web Server to Your DB Instance:** While still connected to your EC2 instance, change the directory to /var/www and create a new subdirectory.
- **Launch an EC2 Instance:** Through the AWS console, you can launch a "virtual laptop" to serve your website.
- **SSH into to the EC2 Instance:** You'll need to connect to your instance via Secure Shell (SSH).
- **Enter Your Instance IP Address:** Enter your Instance IP address (IPv4 Public IP address) on the browser to check if you can see the expected content.

  Linux Commands used in this process:

  sudo su -

  yum update -y

  yum install -y httpd

  systemctl status httpd

  mkdir temp

  cd temp

  wget https://www.free-css.com/assets/files...

  unzip complex.zip

  cd complex

  ls -lrt

  mv * /var/www/html

  systemctl enable httpd

  systemctl start httpd

You can also host a WordPress site on AWS. Here are the steps you'll need to take:

- **Create an EC2 Instance:** Create an AWS EC2 instance to host your WordPress site.
- **Install Apache and MySQL:** Install Apache and MySQL on the EC2 instance.
- **Configure Security Rules:** Click on Add Rule and from the dropdown select HTTP and HTTPs. Click on "Create a new pair" and give a name say "linuxkey" to it.

3 Demonstrate creation of RDS from AWS management console.

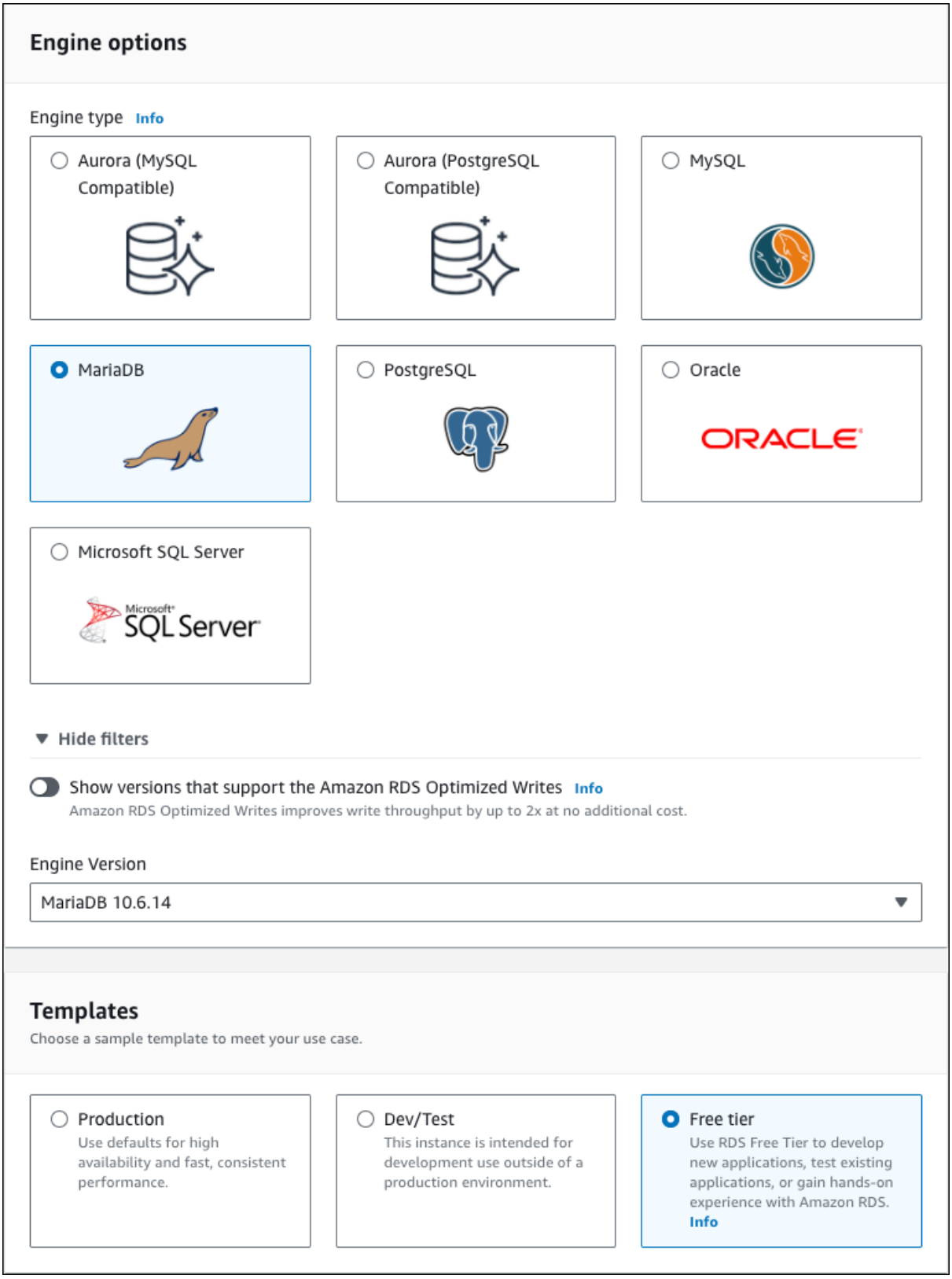
Create an RDS for MariaDB, RDS for MySQL, or RDS for PostgreSQL DB instance that maintains the data used by a web application.

- RDS for MariaDB
- RDS for MySQL
- RDS for PostgreSQL

To create a MariaDB instance

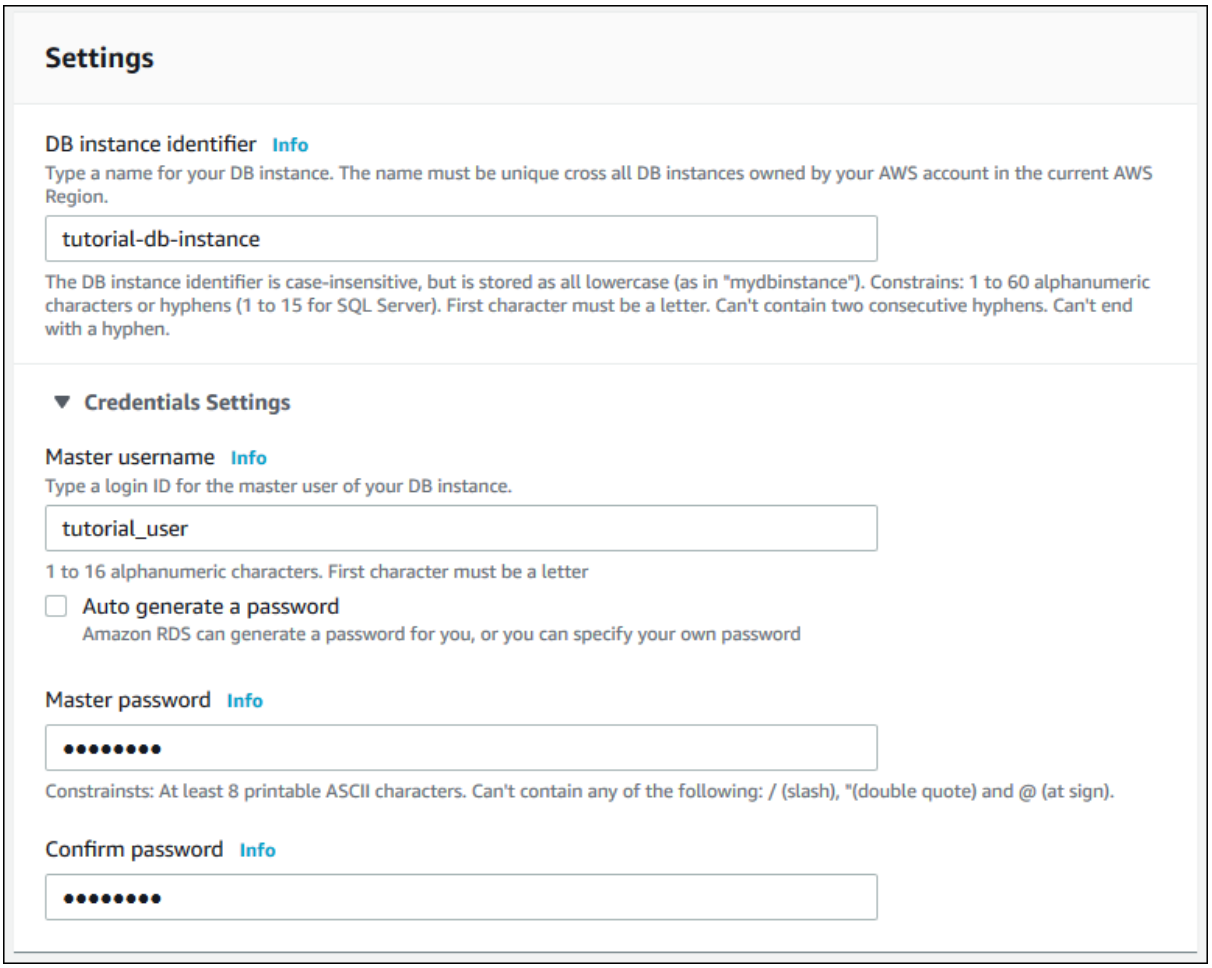
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the upper-right corner of the AWS Management Console, check the AWS Region. It should be the same as the one where you created your EC2 instance.
3. In the navigation pane, choose **Databases**.
4. Choose **Create database**.
5. On the **Create database** page, choose **Standard create**.
6. For **Engine options**, choose **MariaDB**.
7. For **Templates**, choose **Free tier**.

Your DB instance configuration should look similar to the following image.



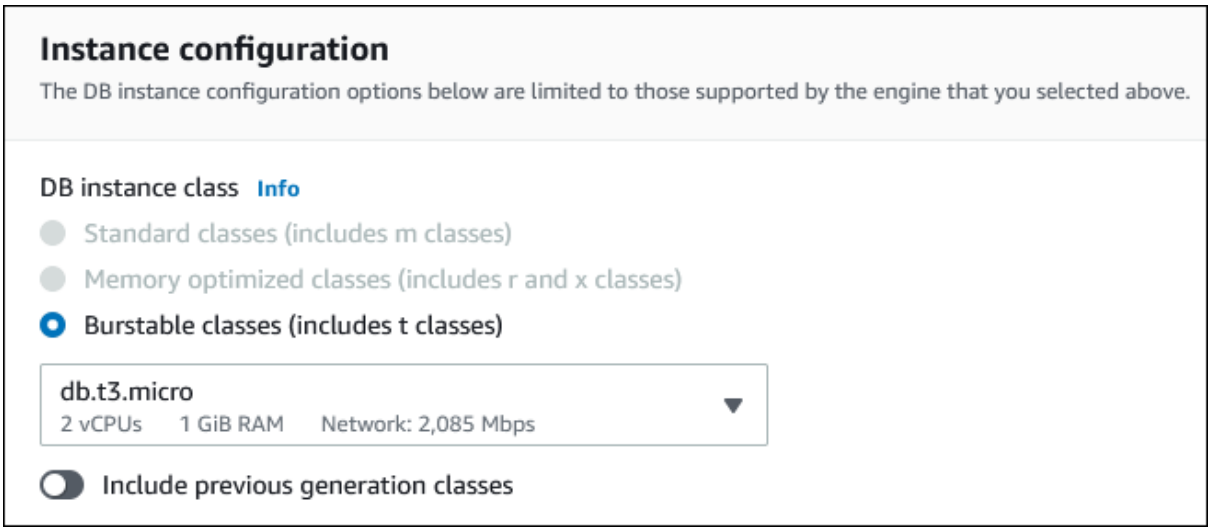
8. In the **Availability and durability** section, keep the defaults.
9. In the **Settings** section, set these values:
    - **DB instance identifier** – Type **tutorial-db-instance**.
    - **Master username** – Type **tutorial_user**.

- **Auto generate a password** – Leave the option turned off.
- **Master password** – Type a password.
- **Confirm password** – Retype the password.

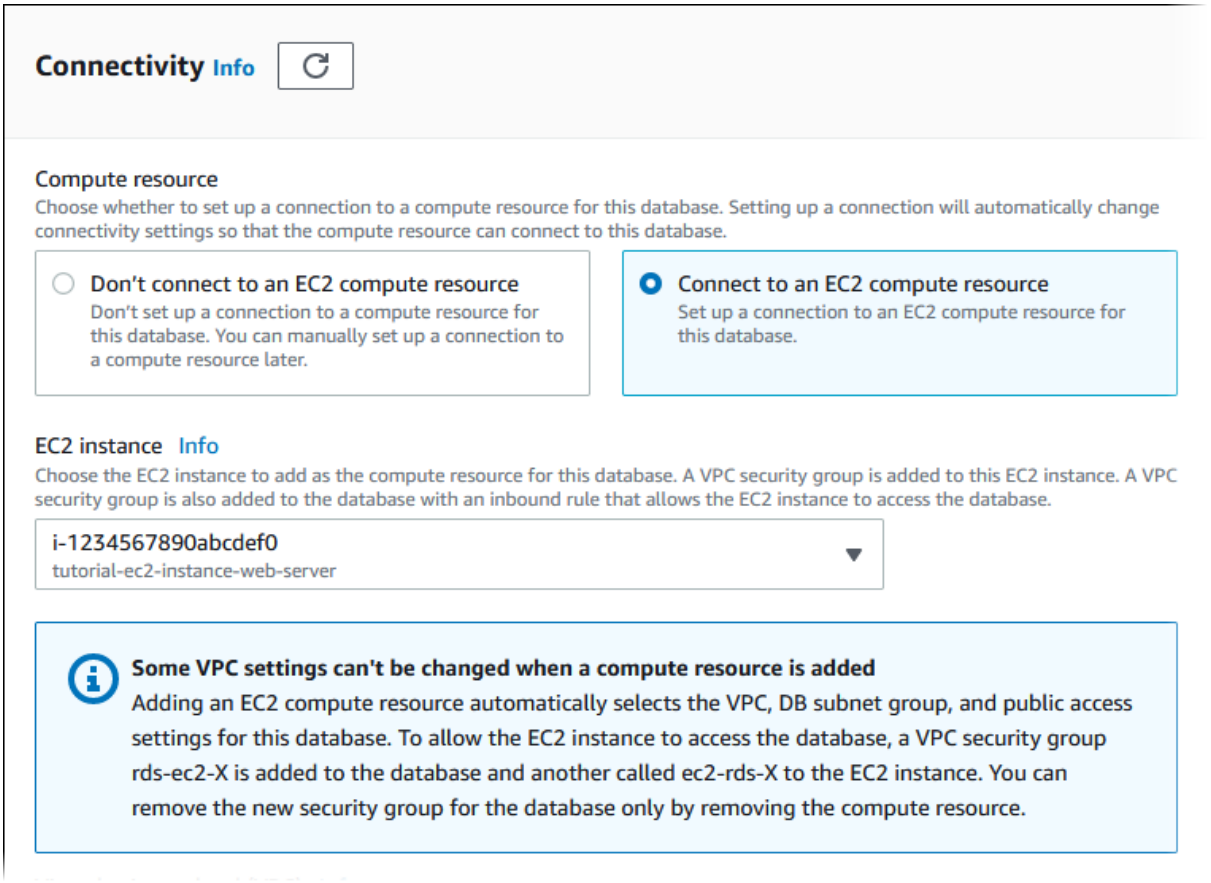


10. In the **Instance configuration** section, set these values:

- **Burstable classes (includes t classes)**
- **db.t3.micro**



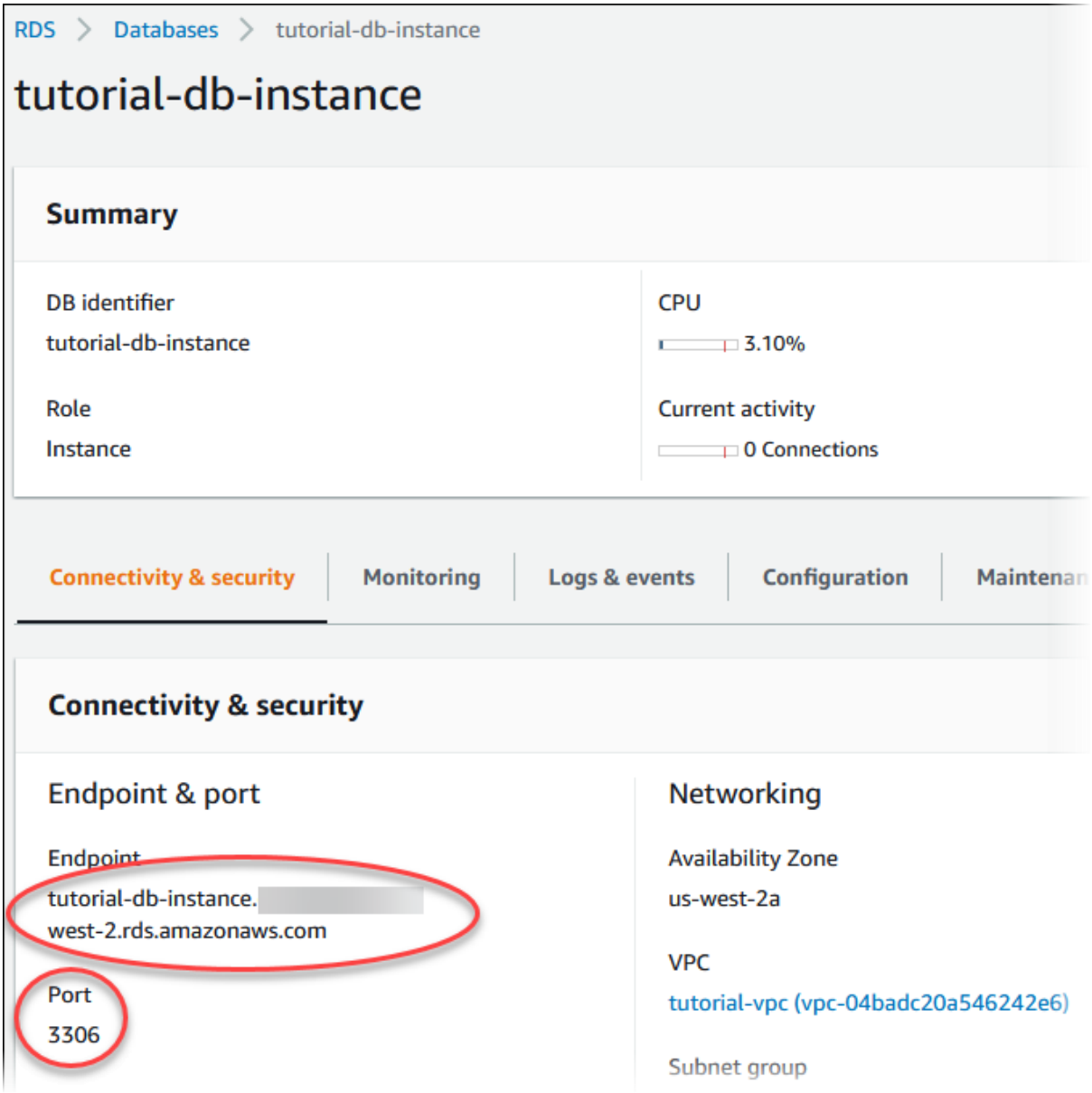
11. In the **Storage** section, keep the defaults.
12. In the **Connectivity** section, set these values and keep the other values as their defaults:
    - For **Compute resource**, choose **Connect to an EC2 compute resource**.
    - For **EC2 instance**, choose the EC2 instance you created previously, such as **tutorial-ec2-instance-web-server**.



13. In the **Database authentication** section, make sure **Password authentication** is selected.
14. Open the **Additional configuration** section, and enter **sample** for **Initial database name**. Keep the default settings for the other options.
15. To create your MariaDB instance, choose **Create database**.

Your new DB instance appears in the **Databases** list with the status **Creating**.

16. Wait for the **Status** of your new DB instance to show as **Available**. Then choose the DB instance name to show its details.
17. In the **Connectivity & security** section, view the **Endpoint** and **Port** of the DB instance.



Note the endpoint and port for your DB instance. You use this information to connect your web server to your DB instance.

18. Complete Install a web server on your EC2 instance.

4 Demonstrate connecting to AWS RDS instance from MySQL workbench.

Steps to Connect to AWS RDS using MySQL Workbench

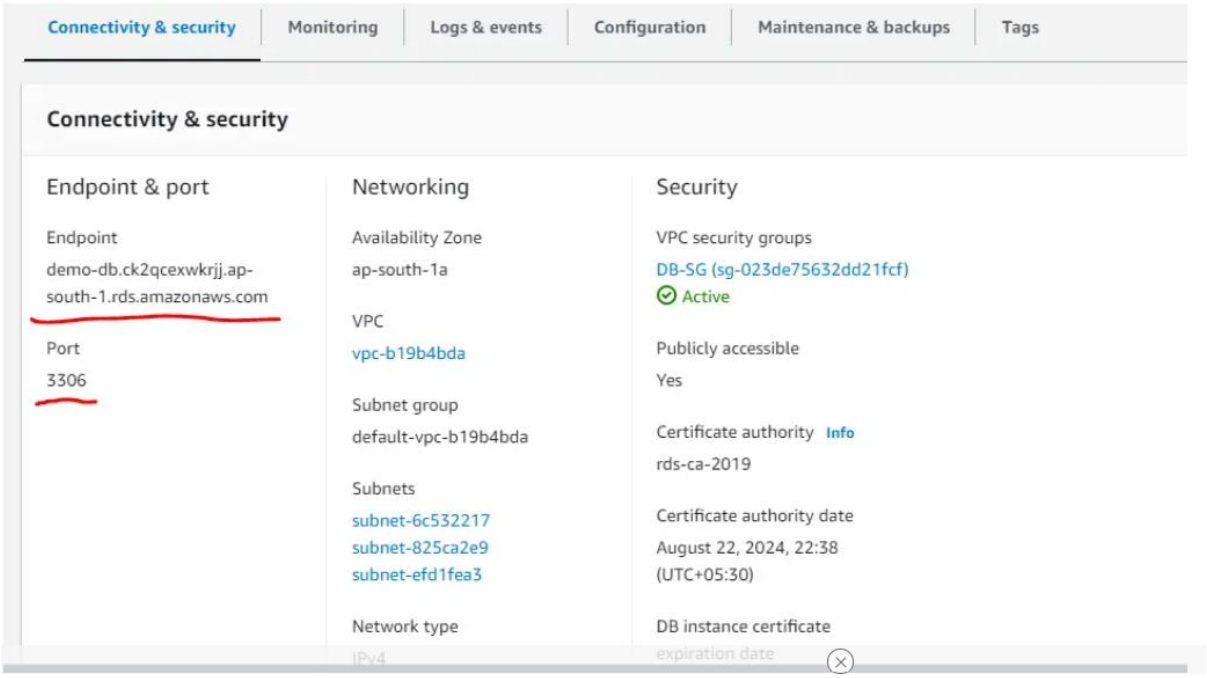
1. **Create a MySQL Database on AWS RDS**
2. **Install MySQL Workbench**
3. **Open MySQL Workbench**
4. **Setup New Connection**
5. **Verify Connectivity**

Step 1: Create a MySQL Database on AWS RDS

Before you can connect to AWS RDS using MySQL workbench, you need one database instance setup done. If you already have that, great.

If not, use tutorial to **setup a MySQL on AWS RDS**.

Once the database is up and running, grab the database endpoint info.

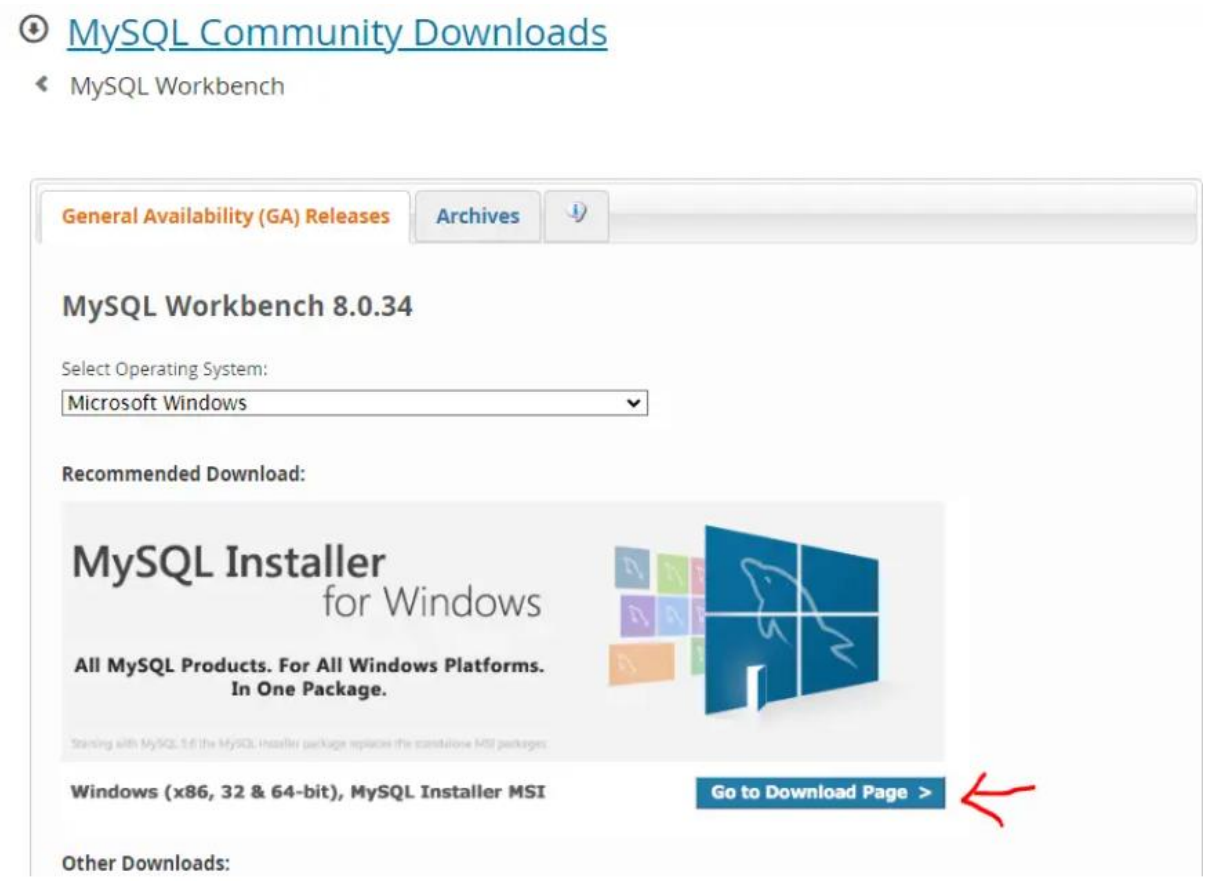


Can you notice the Endpoint and Port? Yeah?

Note them down as we'll need them while connecting to AWS RDS using MySQL workbench.

Step 2: Install MySQL Workbench

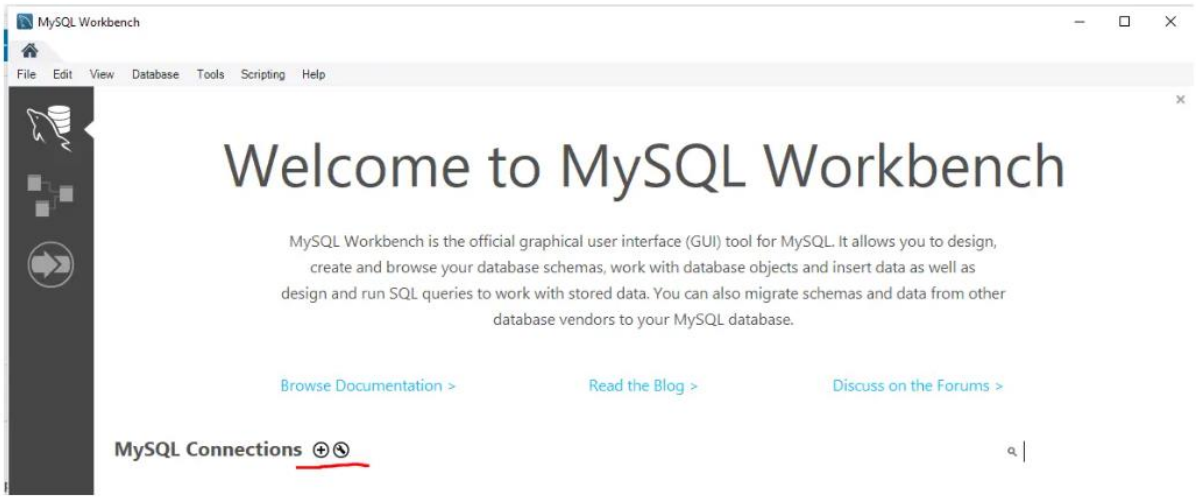
Navigate to MySQL workbench official **downloads page** and download it for your operating system.



Follow the wizard and complete the MySQL workbench installation.

Step 3: Open MySQL Workbench

Open the MySQL workbench and click on the + icon beside **MySQL Connections** as shown below-

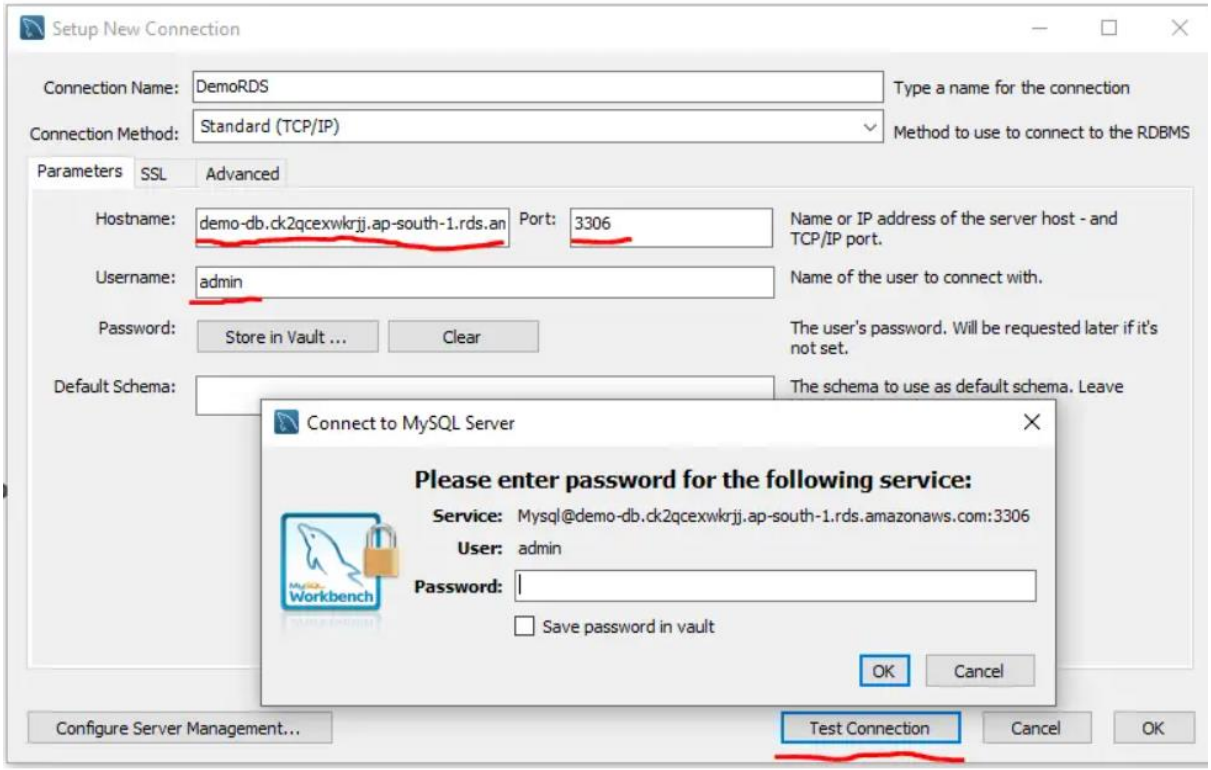


Step 4: Setup New Connection

Once you click on the + icon, it will open the Setup new connection dialogue. We need to put up details like-

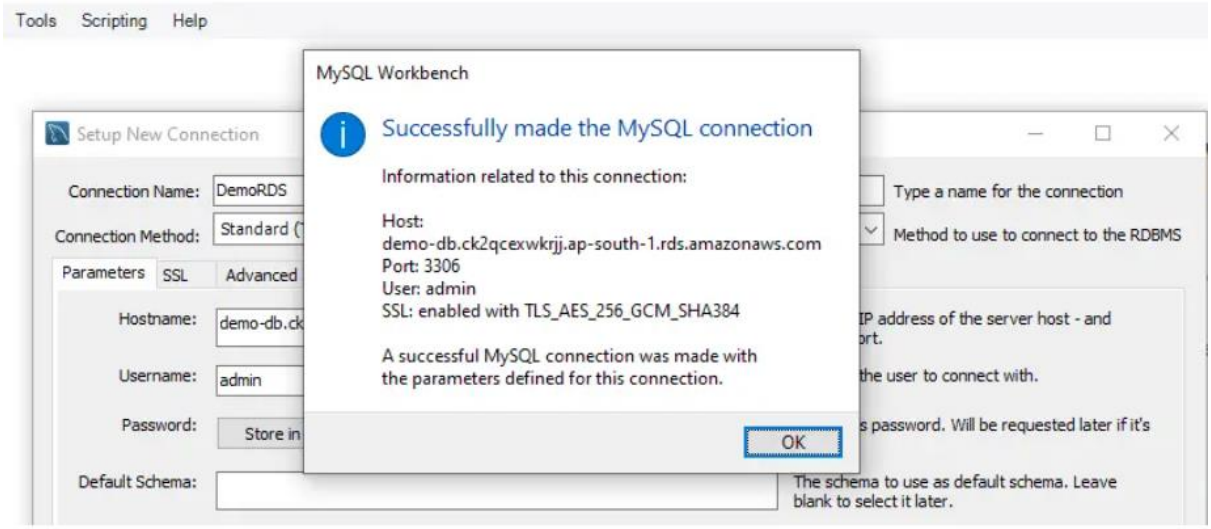
Hostname: Remember we noted down the RDS endpoint.
Port: We have the Port number as well.
Username: User name that you created while setting up MySQL on RDS

Fill up the details shown below-



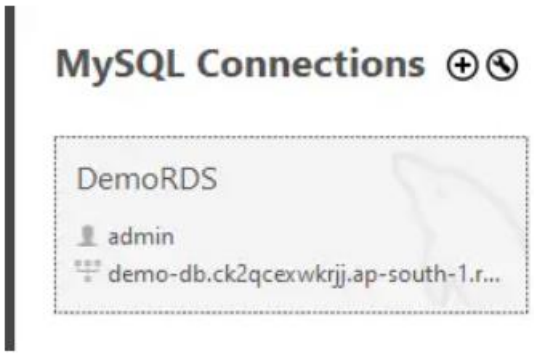
After entering the details, as soon as you click Test Connection, you will be prompted for the password as shown above. Provide the password you set for your DB user and click OK.

You will see, that you are successfully connected to AWS RDS from your MySQL workbench.



Once this dialogue closes, Click OK once again to add this setting to the MySQL workbench dashboard.

And, it gets added to your dashboard like below-

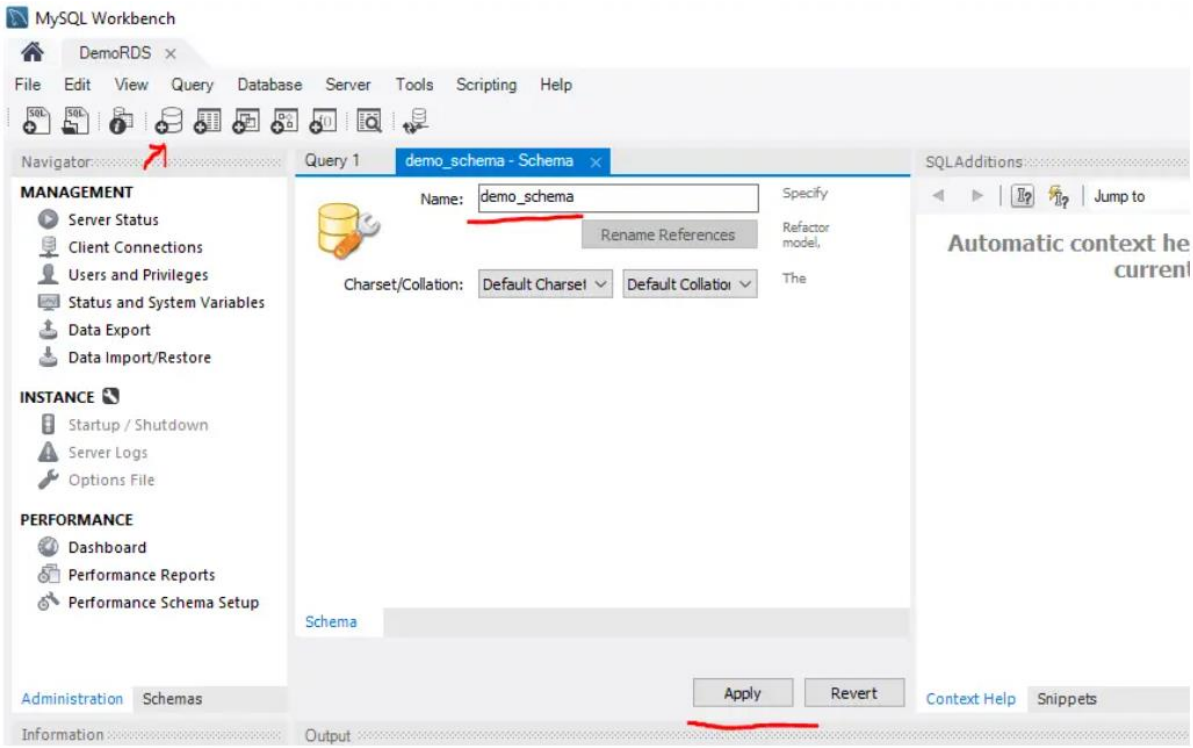


This setting is saved and you can connect to this AWS RDS database instance from your MySQL workbench anytime.

Step 5: Verify Connectivity

Double-click on the above setting and you will be presented with the management dashboard. Let's create a new schema as we don't have any as of yet.

Click on the database icon shown below, provide a schema name and click **Apply**.



Click Apply and click Finish.

Schema gets created as you can see. Next, you can go ahead and create a table, insert data and whatever you need.

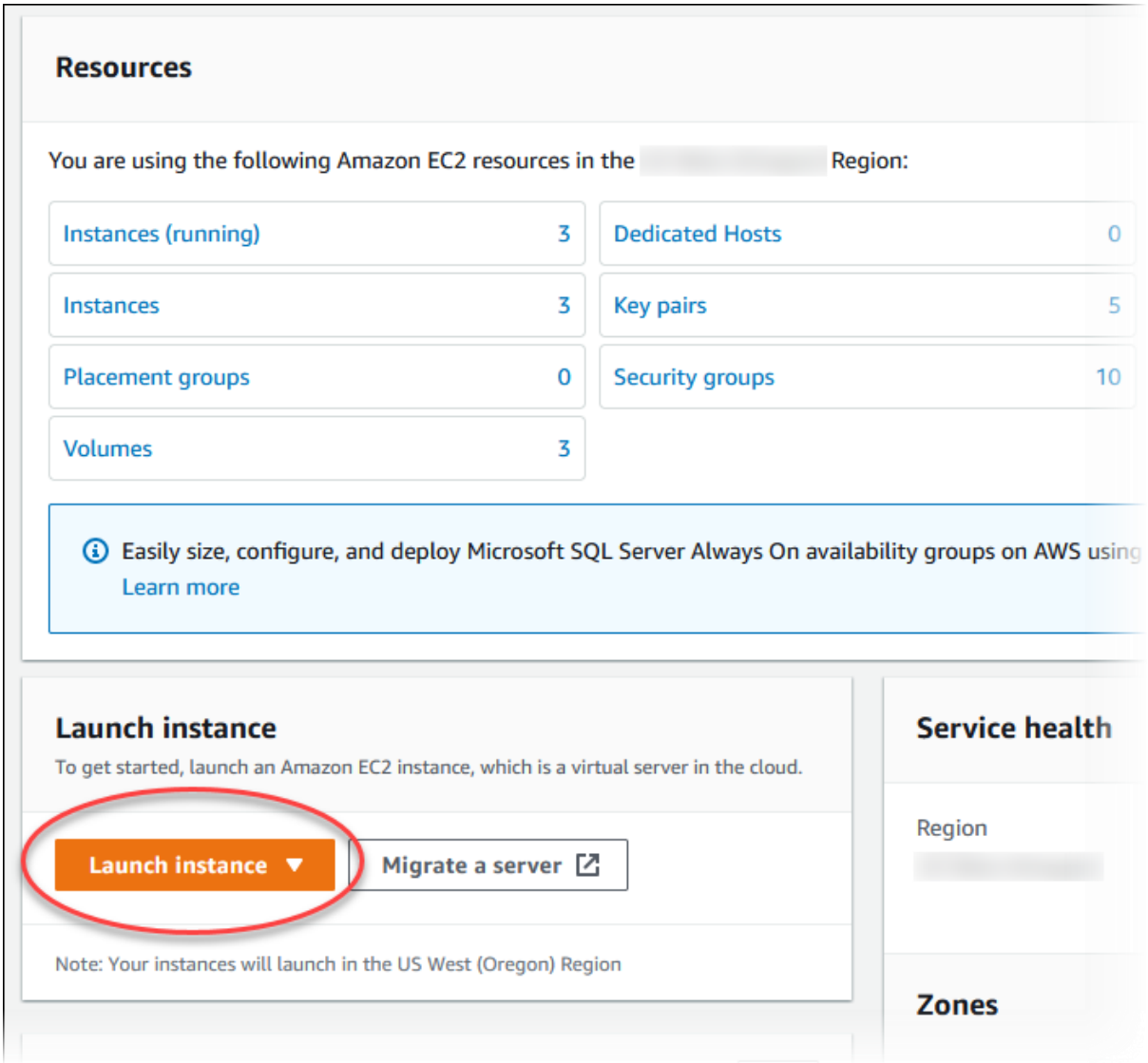
5 Demonstrate Connecting to AWS RDS instance from MariaDB

Step 1: Create an EC2 instance

Create an Amazon EC2 instance that you will use to connect to your database.

To create an EC2 instance

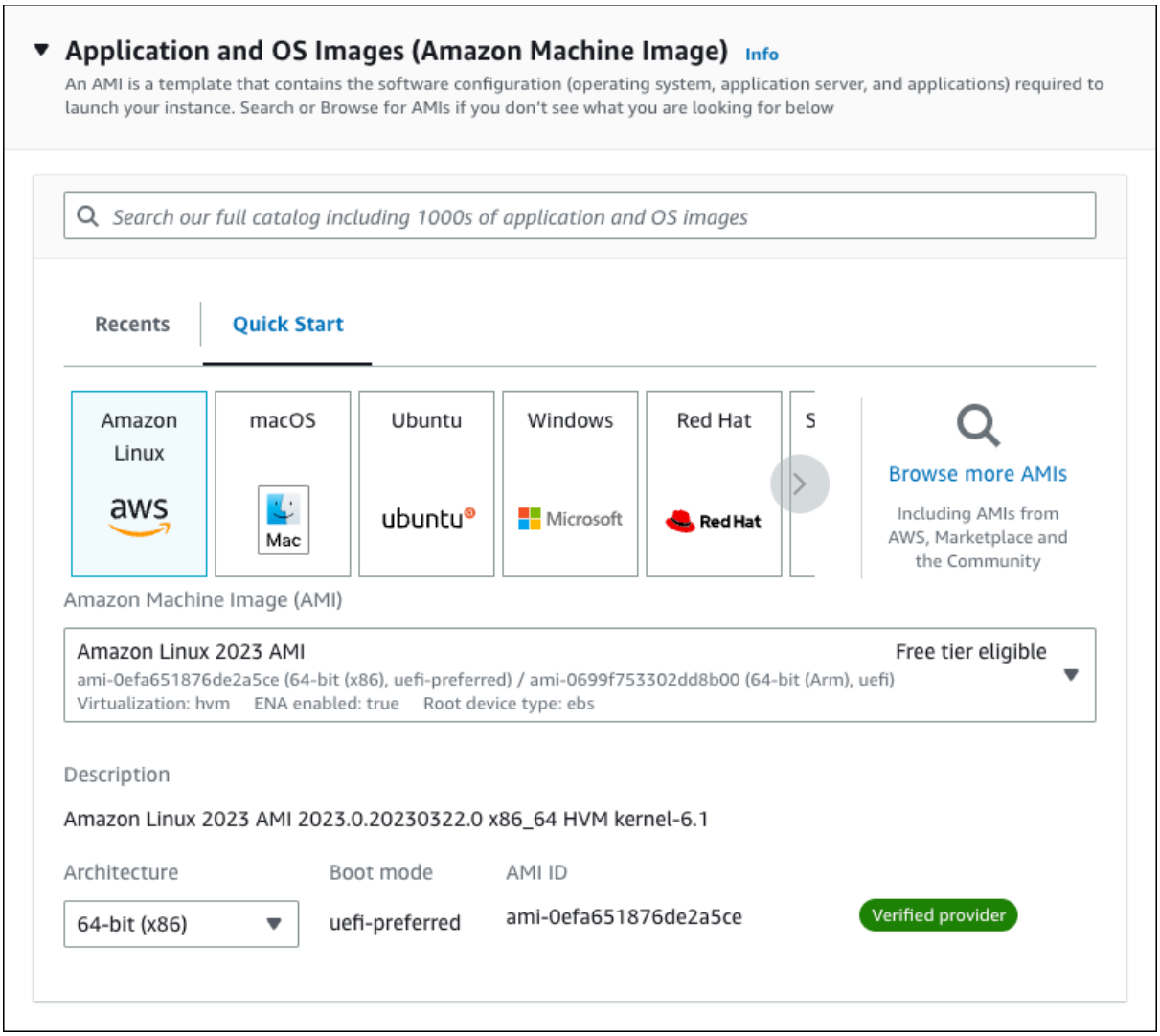
1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the upper-right corner of the AWS Management Console, choose the AWS Region in which you want to create the EC2 instance.
3. Choose **EC2 Dashboard**, and then choose **Launch instance**, as shown in the following image.



The **Launch an instance** page opens.

4. Choose the following settings on the **Launch an instance** page.
    a. Under **Name and tags**, for **Name**, enter **ec2-database-connect**.

b. Under **Application and OS Images (Amazon Machine Image)**, choose **Amazon Linux**, and then choose the **Amazon Linux 2023 AMI**. Keep the default selections for the other choices.



c. Under **Instance type**, choose **t2.micro**.

d. Under **Key pair (login)**, choose a **Key pair name** to use an existing key pair. To create a new key pair for the Amazon EC2 instance, choose **Create new key pair** and then use the **Create key pair** window to create it.

For more information about creating a new key pair, see Create a key pair in the *Amazon EC2 User Guide for Linux Instances*.

e. For **Allow SSH traffic** in **Network settings**, choose the source of SSH connections to the EC2 instance.

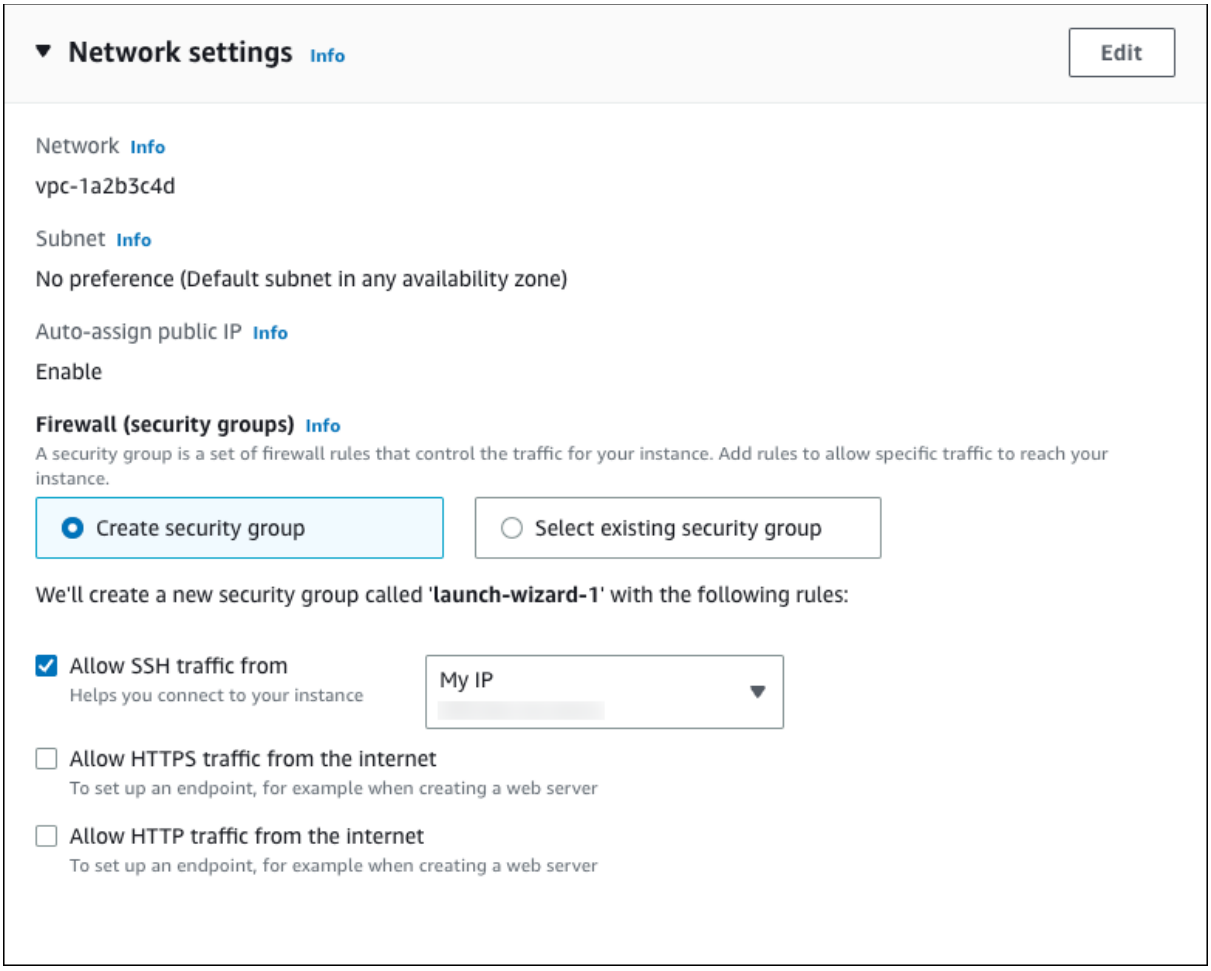
You can choose **My IP** if the displayed IP address is correct for SSH connections. Otherwise, you can determine the IP address to use to connect to EC2 instances in your VPC using Secure Shell (SSH). To determine your public IP address, in a different browser window or tab, you can use the service at https://checkip.amazonaws.com. An example of an IP address is 192.0.2.1/32.

In many cases, you might connect through an internet service provider (ISP) or from behind your firewall without a static IP address. If so, make sure to determine the range of IP addresses used by client computers.

Warning

If you use 0.0.0.0/0 for SSH access, you make it possible for all IP addresses to access your public EC2 instances using SSH. This approach is acceptable for a short time in a test environment, but it's unsafe for production environments. In production, authorize only a specific IP address or range of addresses to access your EC2 instances using SSH.

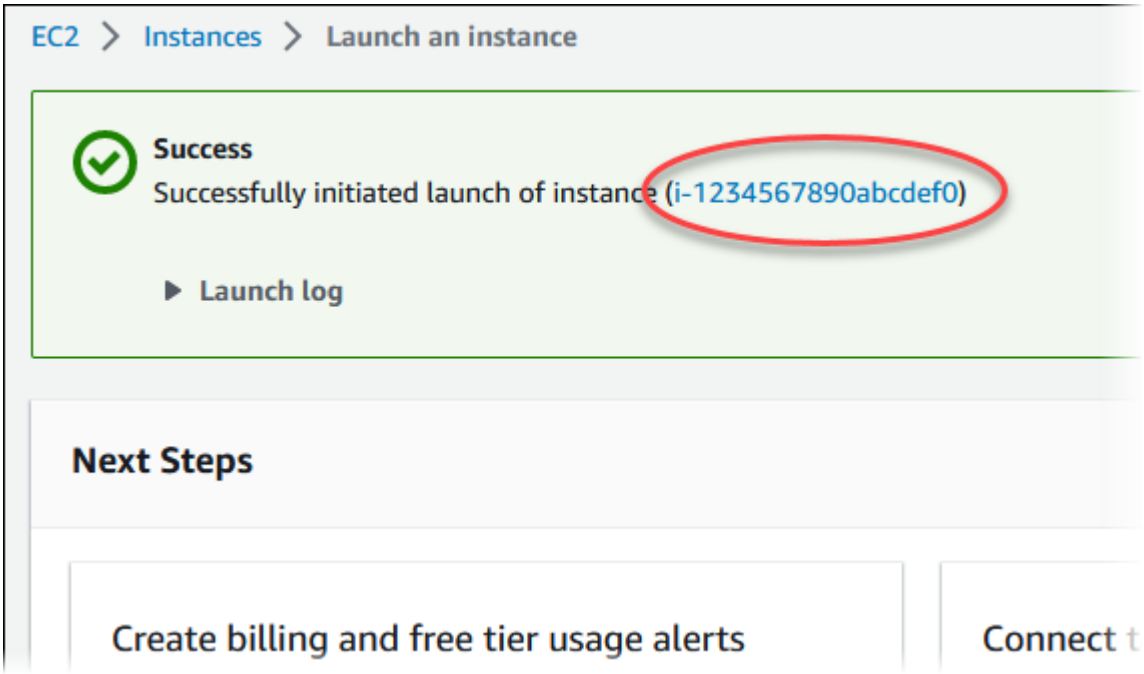
The following image shows an example of the **Network settings** section.



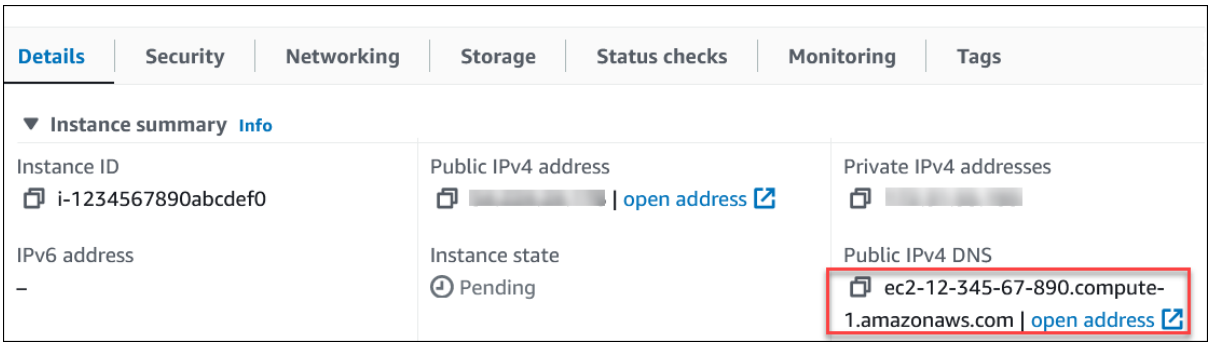
f. Leave the default values for the remaining sections.

g. Review a summary of your EC2 instance configuration in the **Summary** panel, and when you're ready, choose **Launch instance**.

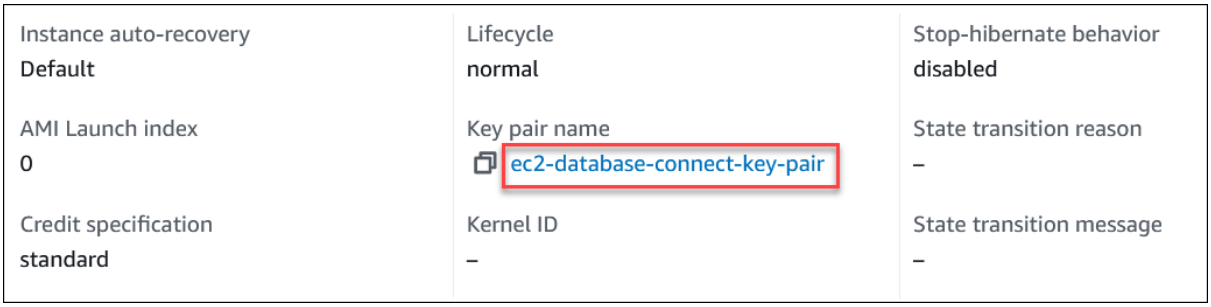
5. On the **Launch Status** page, note the identifier for your new EC2 instance, for example: i-1234567890abcdef0.



6. Choose the EC2 instance identifier to open the list of EC2 instances, and then select your EC2 instance.
7. In the **Details** tab, note the following values, which you need when you connect using SSH:
    a. In **Instance summary**, note the value for **Public IPv4 DNS**.



    b. In **Instance details**, note the value for **Key pair name**.



8. Wait until the **Instance state** for your EC2 instance has a status of **Running** before continuing.

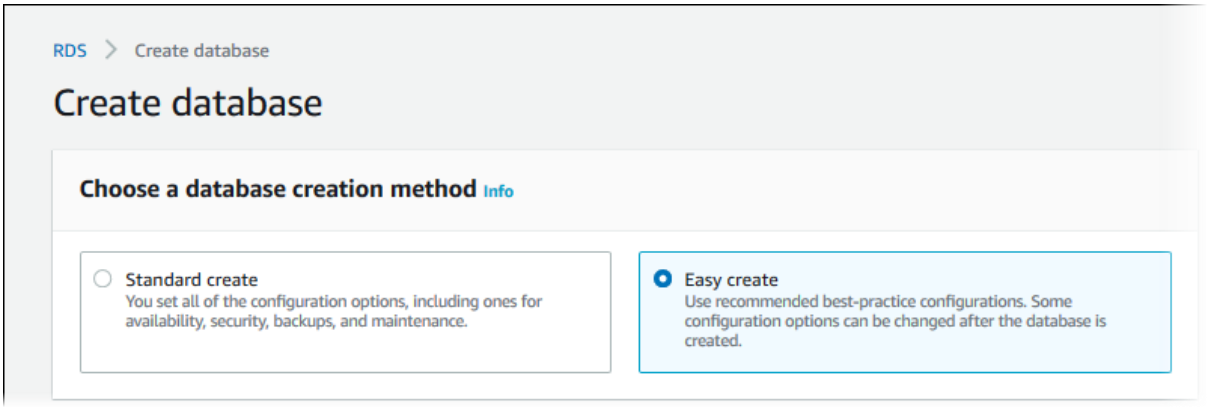
Step 2: Create a MySQL DB instance

The basic building block of Amazon RDS is the DB instance. This environment is where you run your MySQL databases.

In this example, you use **Easy create** to create a DB instance running the MySQL database engine with a db.t3.micro DB instance class.

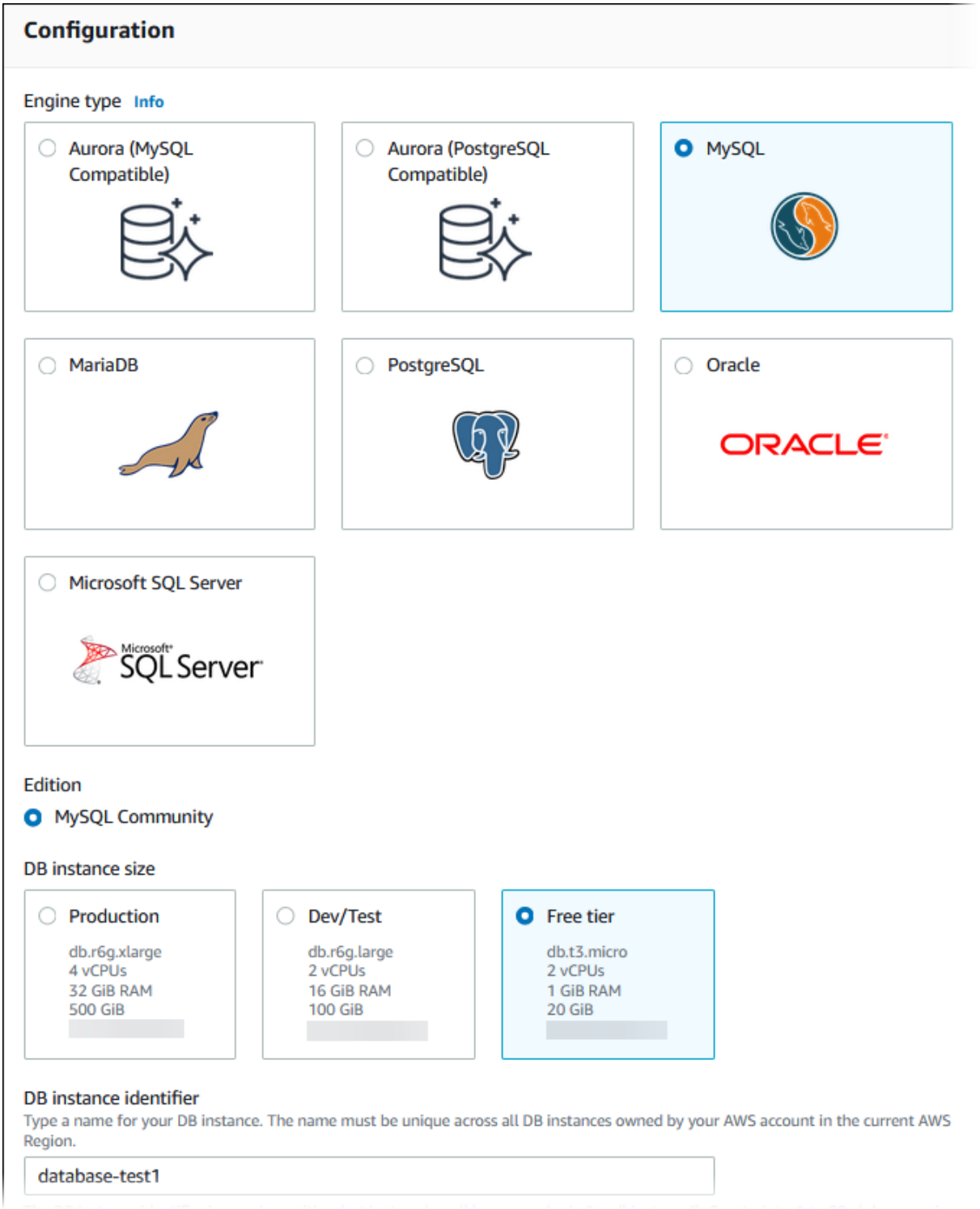
To create a MySQL DB instance with Easy create

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the upper-right corner of the Amazon RDS console, choose the AWS Region you used for the EC2 instance previously.
3. In the navigation pane, choose **Databases**.
4. Choose **Create database** and make sure that **Easy create** is chosen.



5. In **Configuration**, choose **MySQL**.
6. For **DB instance size**, choose **Free tier**.
7. For **DB instance identifier**, enter **database-test1**.
8. For **Master username**, enter a name for the master user, or keep the default name.

The **Create database** page should look similar to the following image.

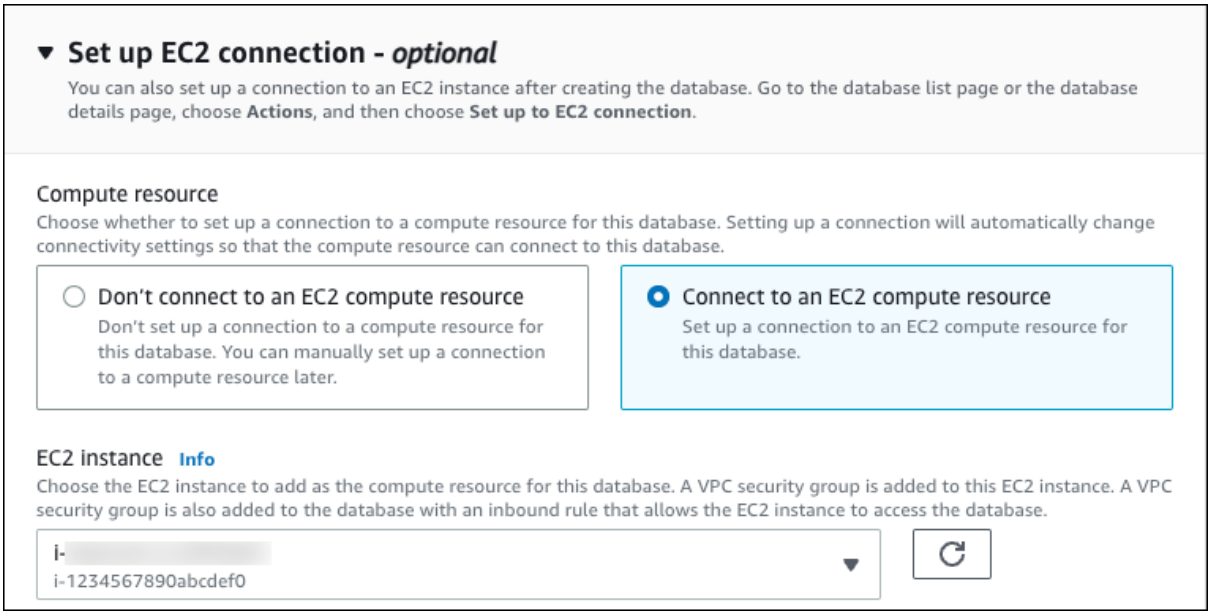


9. To use an automatically generated master password for the DB instance, select **Auto generate a password**.

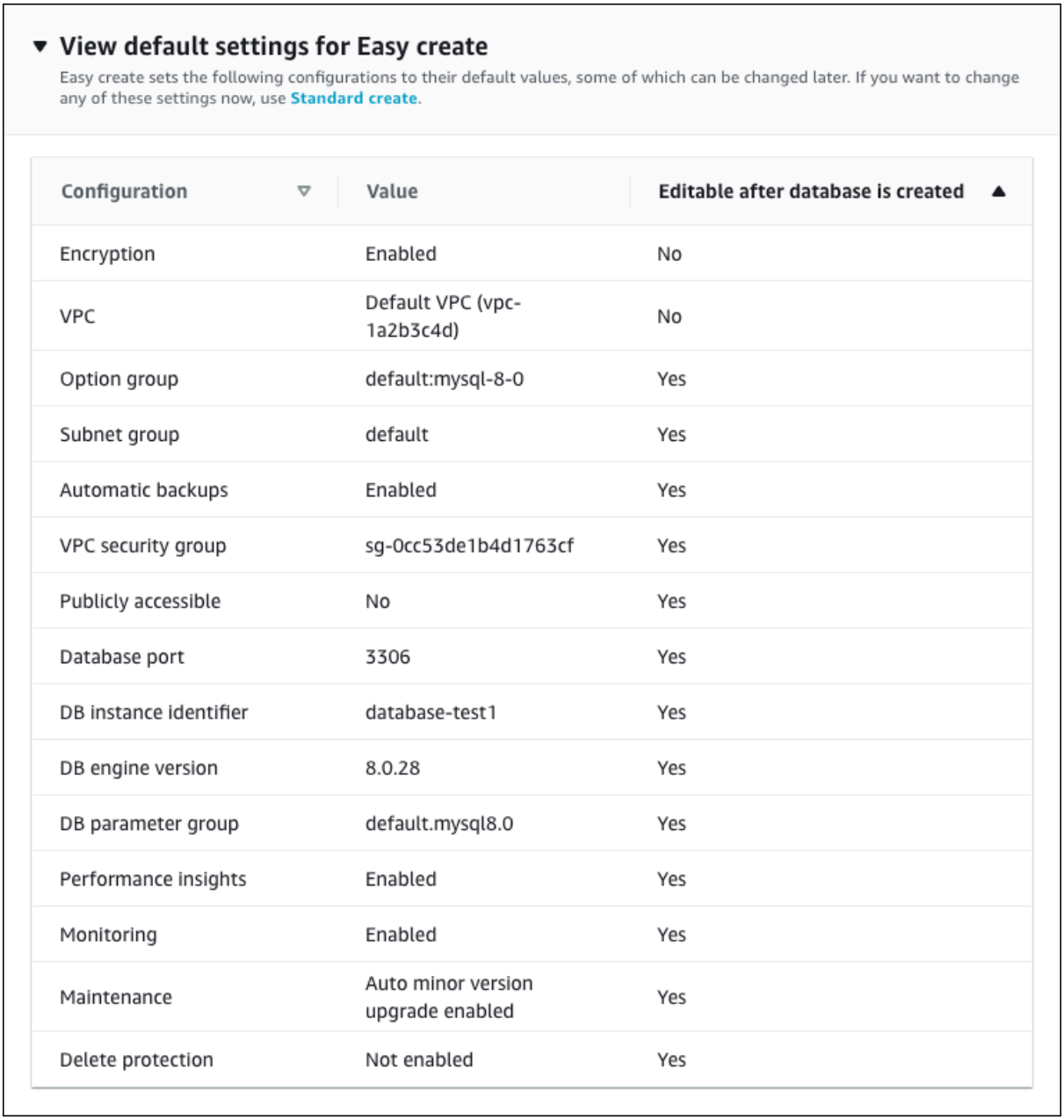
To enter your master password, make sure **Auto generate a password** is cleared, and then enter the same password in **Master password** and **Confirm password**.

10. To set up a connection with the EC2 instance you created previously, open **Set up EC2 connection - *optional***.

Select **Connect to an EC2 compute resource**. Choose the EC2 instance you created previously.



11. (Optional) Open **View default settings for Easy create**.



View default settings for Easy create

Easy create sets the following configurations to their default values, some of which can be changed later. If you want to change any of these settings now, use **Standard create**.

Configuration	Value	Editable after database is created
Encryption	Enabled	No
VPC	Default VPC (vpc-1a2b3c4d)	No
Option group	default:mysql-8-0	Yes
Subnet group	default	Yes
Automatic backups	Enabled	Yes
VPC security group	sg-0cc53de1b4d1763cf	Yes
Publicly accessible	No	Yes
Database port	3306	Yes
DB instance identifier	database-test1	Yes
DB engine version	8.0.28	Yes
DB parameter group	default.mysql8.0	Yes
Performance insights	Enabled	Yes
Monitoring	Enabled	Yes
Maintenance	Auto minor version upgrade enabled	Yes
Delete protection	Not enabled	Yes

You can examine the default settings used with **Easy create**. The **Editable after database is created** column shows which options you can change after you create the database.

- If a setting has **No** in that column, and you want a different setting, you can use **Standard create** to create the DB instance.
- If a setting has **Yes** in that column, and you want a different setting, you can either use **Standard create** to create the DB instance, or modify the DB instance after you create it to change the setting.

12. Choose **Create database**.

To view the master username and password for the DB instance, choose **View credential details**.

You can use the username and password that appears to connect to the DB instance as the master user.

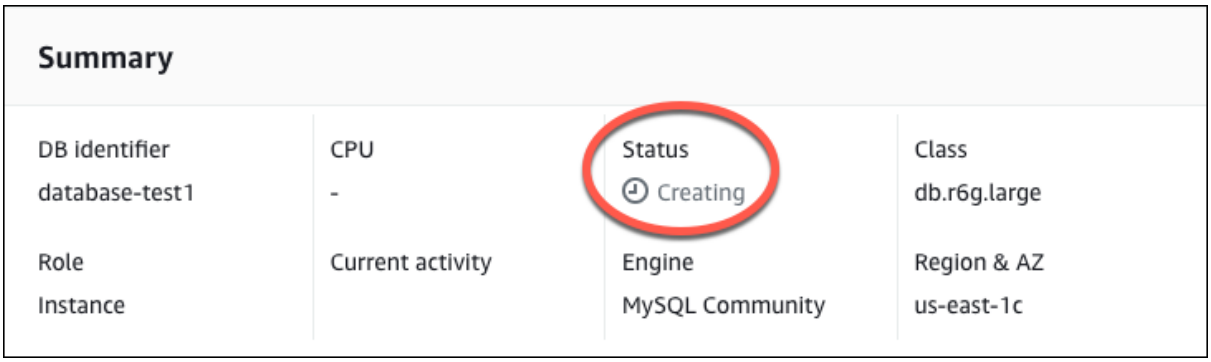
Important

You can't view the master user password again. If you don't record it, you might have to change it.

If you need to change the master user password after the DB instance is available, you can modify the DB instance to do so. For more information about modifying a DB instance, see Modifying an Amazon RDS DB instance.

13. In the **Databases** list, choose the name of the new MySQL DB instance to show its details.

The DB instance has a status of **Creating** until it is ready to use.



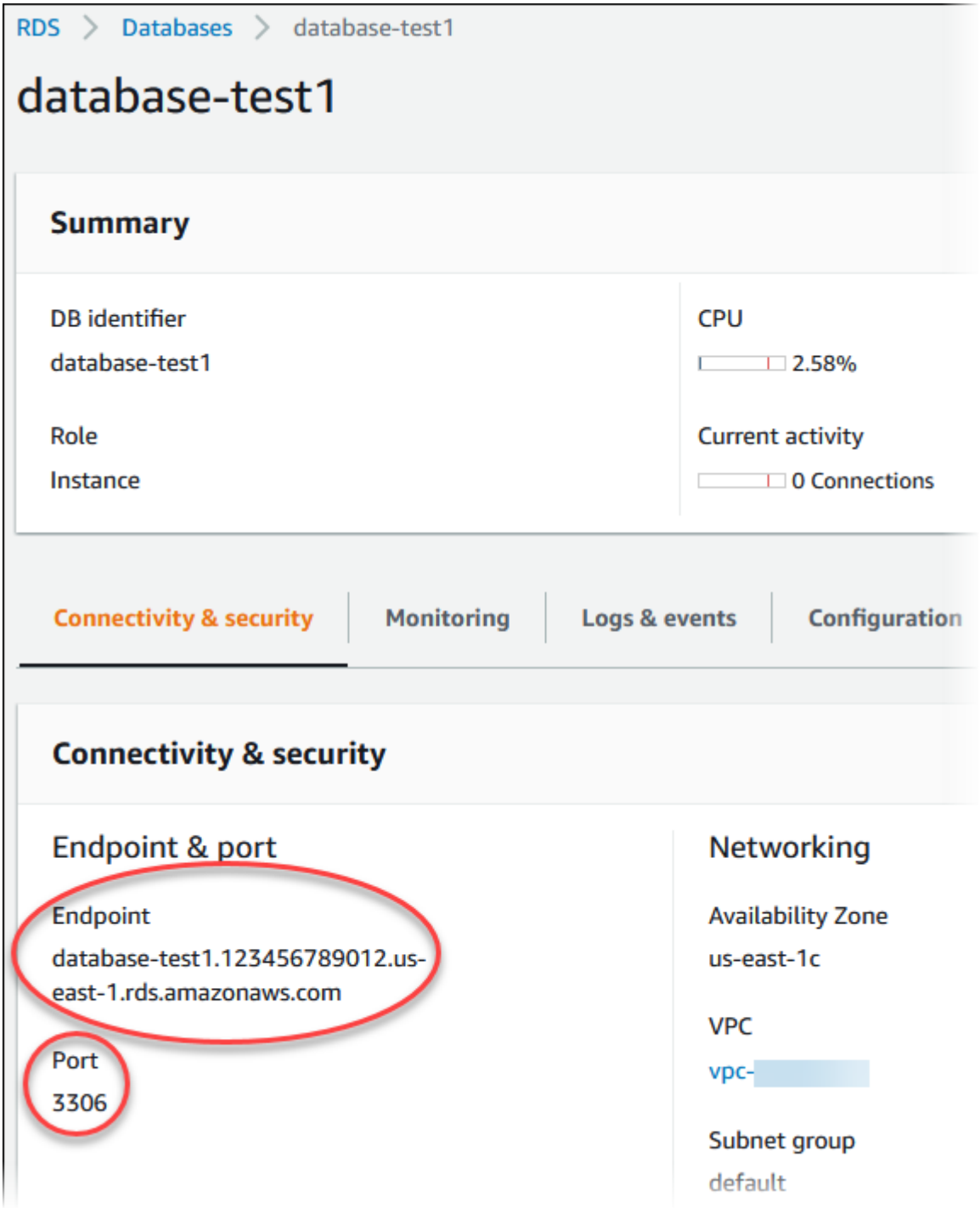
When the status changes to **Available**, you can connect to the DB instance. Depending on the DB instance class and the amount of storage, it can take up to 20 minutes before the new instance is available.

Step 3: Connect to a MySQL DB instance

You can use any standard SQL client application to connect to the DB instance. In this example, you connect to a MySQL DB instance using the mysql command-line client.

To connect to a MySQL DB instance

1. Find the endpoint (DNS name) and port number for your DB instance.
    a. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
    b. In the upper-right corner of the Amazon RDS console, choose the AWS Region for the DB instance.
    c. In the navigation pane, choose **Databases**.
    d. Choose the MySQL DB instance name to display its details.
    e. On the **Connectivity & security** tab, copy the endpoint. Also, note the port number. You need both the endpoint and the port number to connect to the DB instance.



2. Connect to the EC2 instance that you created earlier by following the steps in Connect to your Linux instance in the *Amazon EC2 User Guide for Linux Instances*.

We recommend that you connect to your EC2 instance using SSH. If the SSH client utility is installed on Windows, Linux, or Mac, you can connect to the instance using the following command format:

ssh -i *location_of_pem_file* ec2-user@*ec2-instance-public-dns-name*

For example, assume that ec2-database-connect-key-pair.pem is stored in /dir1 on Linux, and the public IPv4 DNS for your EC2 instance is ec2-12-345-678-90.compute-1.amazonaws.com. Your SSH command would look as follows:

ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-user@ec2-12-345-678-90.compute-1.amazonaws.com

3. Get the latest bug fixes and security updates by updating the software on your EC2 instance. To do this, use the following command.

Note

The -y option installs the updates without asking for confirmation. To examine updates before installing, omit this option.

sudo dnf update -y

4. To install the mysql command-line client from MariaDB on Amazon Linux 2023, run the following command:

sudo dnf install mariadb105

5. Connect to the MySQL DB instance. For example, enter the following command. This action lets you connect to the MySQL DB instance using the MySQL client.

Substitute the DB instance endpoint (DNS name) for *endpoint*, and substitute the master username that you used for *admin*. Provide the master password that you used when prompted for a password.

mysql -h *endpoint* -P 3306 -u *admin* -p

After you enter the password for the user, you should see output similar to the following.

Welcome to the MariaDB monitor. Commands end with ; or \g.

Your MySQL connection id is 3082

Server version: 8.0.28 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>

For more information about connecting to a MySQL DB instance, see Connecting to a DB instance running the MySQL database engine. If you can't connect to your DB instance, see Can't connect to Amazon RDS DB instance.

For security, it is a best practice to use encrypted connections. Only use an unencrypted MySQL connection when the client and server are in the same VPC and the network is trusted. For information about using encrypted connections, see Connecting from the MySQL command-line client with SSL/TLS (encrypted).

6. Run SQL commands.

For example, the following SQL command shows the current date and time:

SELECT CURRENT_TIMESTAMP;

Step 4: Delete the EC2 instance and DB instance

After you connect to and explore the sample EC2 instance and DB instance that you created, delete them so you're no longer charged for them.

To delete the EC2 instance

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Instances**.
3. Select the EC2 instance, and choose **Instance state, Terminate instance**.
4. Choose **Terminate** when prompted for confirmation.

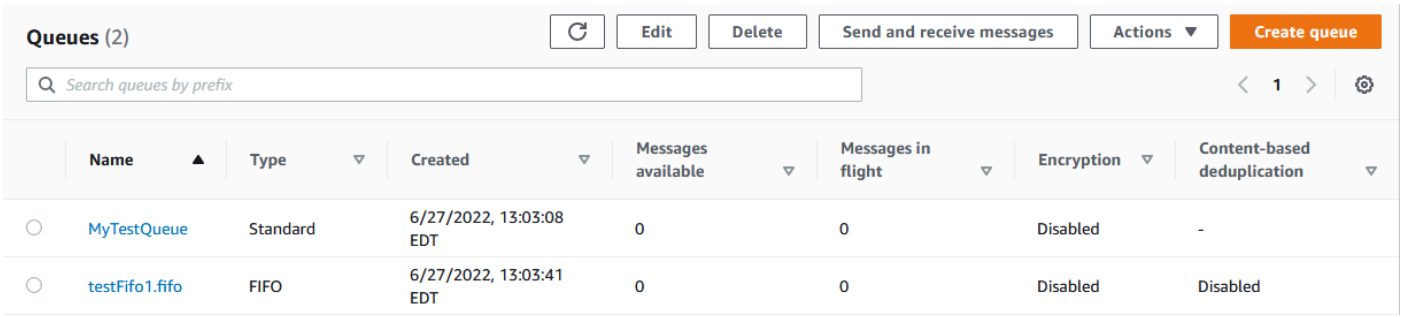
For more information about deleting an EC2 instance, see Terminate your instance in the *Amazon EC2 User Guide for Linux Instances*.

To delete the DB instance with no final DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Databases**.
3. Choose the DB instance that you want to delete.
4. For **Actions**, choose **Delete**.
5. Clear **Create final snapshot?** and **Retain automated backups**.
6. Complete the acknowledgement and choose **Delete**.

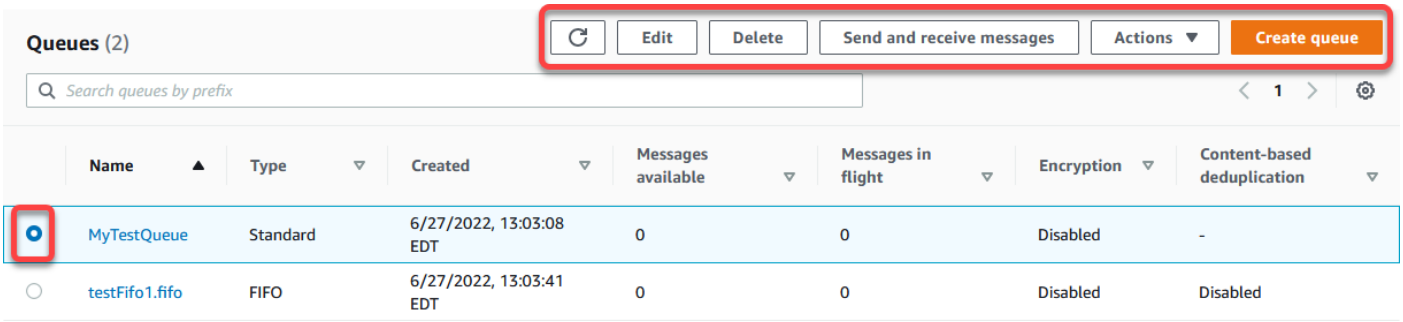
6 Demonstrate the Creation of a simple Queue service using the AWS management console.

When you open the console, choose **Queues** from the navigation pane to display the **Queues** page. The **Queues** page provides information about all of your queues in the active region.

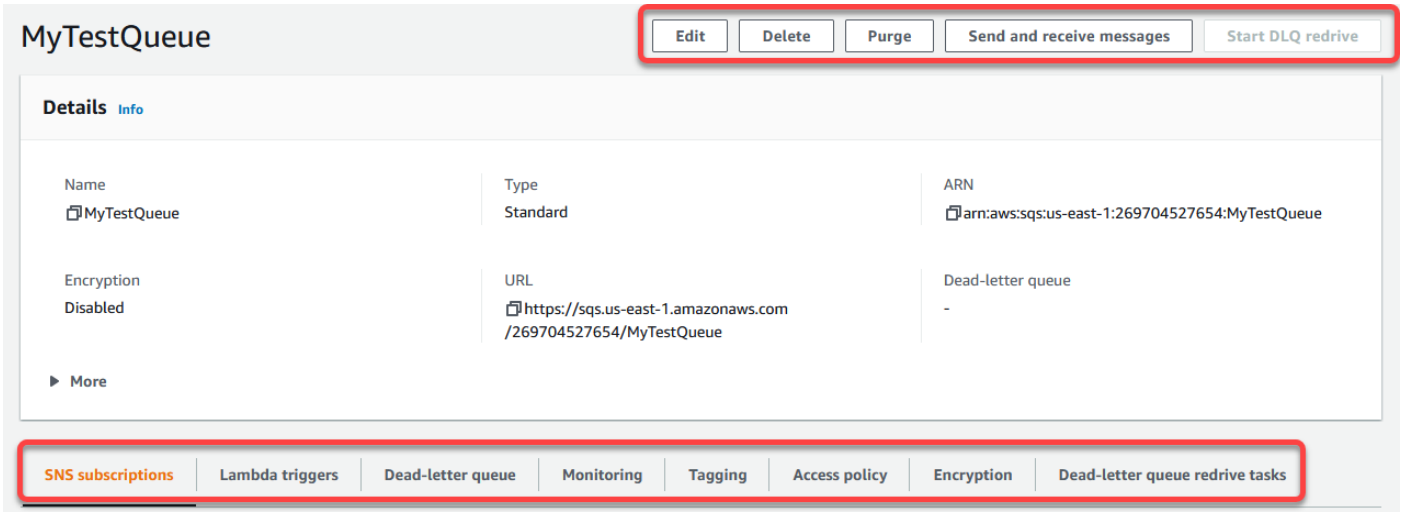


The entry for each queue shows the queue type and other information about the queue. The **Type** column helps you distinguish standard queues from First-In-First Out (FIFO) queues at a glance.

From the **Queues** page, there are two ways to perform actions on a queue. You can choose the option next to the queue name and then choose the action you want to perform on the queue.



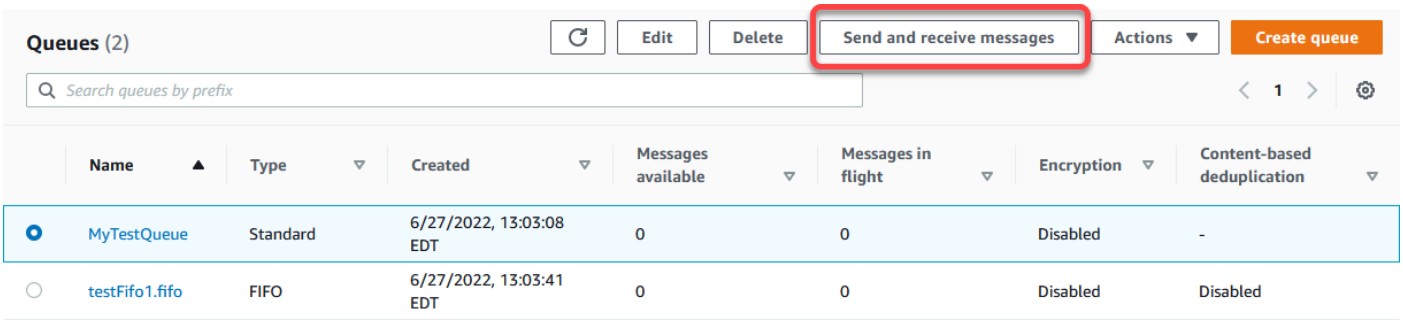
You can also choose the queue name, which open the **Details** page for the queue. The **Details** page includes the same actions as the **Queues** page. In addition, you can choose one of the tabs below the **Details** section to view additional configuration details and actions.



To receive and delete a message (console)

1. Open the Amazon SQS console at https://console.aws.amazon.com/sqs/.
2. In the navigation pane, choose **Queues**.

3. On the **Queues** page, choose a queue.
4. Choose **Send and receive messages**.



The console displays the **Send and receive messages** page.

5. Choose **Poll for messages**.

Amazon SQS begins to poll for messages in the queue. The progress bar on the right side of the **Receive messages** section displays the duration of polling.

The **Messages** section displays a list of the received messages. For each message, the list displays the message ID, Sent date, Size, and Receive count.

6. To delete messages, choose the messages that you want to delete and choose **Delete**.
7. In the **Delete Messages** dialog box, choose **Delete**.

To delete a queue (console)

1. Open the Amazon SQS console at https://console.aws.amazon.com/sqs/.
2. In the navigation pane, choose **Queues**.
3. On the **Queues** page, choose the queue to delete.
4. Choose **Delete**.
5. In the **Delete queue** dialog box, confirm the deletion by entering **delete**.
6. Choose **Delete**.

To delete a queue (AWS CLIAWS API)

You can use one of the following commands to delete a queue:

- AWS CLI: aws sqs delete-queue
- AWS API: DeleteQueue

Demonstrate managing SQS with AWS SDK using Java (Cloud 9).

```
import com.amazonaws.services.sqs.AmazonSQS;

import com.amazonaws.services.sqs.AmazonSQSClientBuilder;

import com.amazonaws.services.sqs.model.AmazonSQSException;
```

```
import com.amazonaws.services.sqs.model.CreateQueueRequest;
```

create a Queue:

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();
CreateQueueRequest create_request = new CreateQueueRequest(QUEUE_NAME)
        .addAttributesEntry("DelaySeconds", "60")
        .addAttributesEntry("MessageRetentionPeriod", "86400");


try {
    sqs.createQueue(create_request);
} catch (AmazonSQSException e) {
    if (!e.getErrorCode().equals("QueueAlreadyExists")) {
        throw e;
    }
}
sqs.createQueue("MyQueue" + new Date().getTime());
```

Listing Queues:

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.ListQueuesResult;
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();
ListQueuesResult lq_result = sqs.listQueues();
System.out.println("Your SQS Queue URLs:");
for (String url : lq_result.getQueueUrls()) {
    System.out.println(url);
}


import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.ListQueuesRequest;
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();
String name_prefix = "Queue";
```

```
lq_result = sqs.listQueues(new ListQueuesRequest(name_prefix));
System.out.println("Queue URLs with prefix: " + name_prefix);
for (String url : lq_result.getQueueUrls()) {
   System.out.println(url);
}
```

Send a Message

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.SendMessageRequest;
SendMessageRequest send_msg_request = new SendMessageRequest()
      .withQueueUrl(queueUrl)
      .withMessageBody("hello world")
      .withDelaySeconds(5);
sqs.sendMessage(send_msg_request);
```

Send Multiple Messages at Once

```
import com.amazonaws.services.sqs.model.SendMessageBatchRequest;
import com.amazonaws.services.sqs.model.SendMessageBatchRequestEntry;
SendMessageBatchRequest send_batch_request = new SendMessageBatchRequest()
      .withQueueUrl(queueUrl)
      .withEntries(
            new SendMessageBatchRequestEntry(
                  "msg_1", "Hello from message 1"),
            new SendMessageBatchRequestEntry(
                  "msg_2", "Hello from message 2")
                  .withDelaySeconds(10));
sqs.sendMessageBatch(send_batch_request);
```

Receive Messages:

```
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.AmazonSQSException;
import com.amazonaws.services.sqs.model.SendMessageBatchRequest;
List<Message> messages = sqs.receiveMessage(queueUrl).getMessages();
Delete Messages after Receipt
for (Message m : messages) {
   sqs.deleteMessage(queueUrl, m.getReceiptHandle());
}
```

7 Demonstrate the creation of an AWS Lambda function

You can run Java code in AWS Lambda. Lambda provides runtimes for Java that run your code to process events. Your code runs in an Amazon Linux environment that includes AWS credentials from an AWS Identity and Access Management (IAM) role that you manage.

Lambda provides the following libraries for Java functions:

- com.amazonaws:aws-lambda-java-core (required) – Defines handler method interfaces and the context object that the runtime passes to the handler. If you define your own input types, this is the only library that you need.
- com.amazonaws:aws-lambda-java-events – Input types for events from services that invoke Lambda functions.
- com.amazonaws:aws-lambda-java-log4j2 – An appender library for Apache Log4j 2 that you can use to add the request ID for the current invocation to your function logs.
- AWS SDK for Java 2.0 – The official AWS SDK for the Java programming language.

To create a Java function

1. Open the Lambda console.
2. Choose **Create function**.
3. Configure the following settings:
   - **Function name**: Enter a name for the function.
   - **Runtime**: Choose **Java 17**.
4. Choose **Create function**.
5. To configure a test event, choose **Test**.
6. For **Event name**, enter `test`.
7. Choose **Save changes**.
8. To invoke the function, choose **Test**.

The console creates a Lambda function with a handler class named `Hello`. Since Java is a compiled language, you can't view or edit the source code in the Lambda console, but you can modify its configuration, invoke it, and configure triggers.

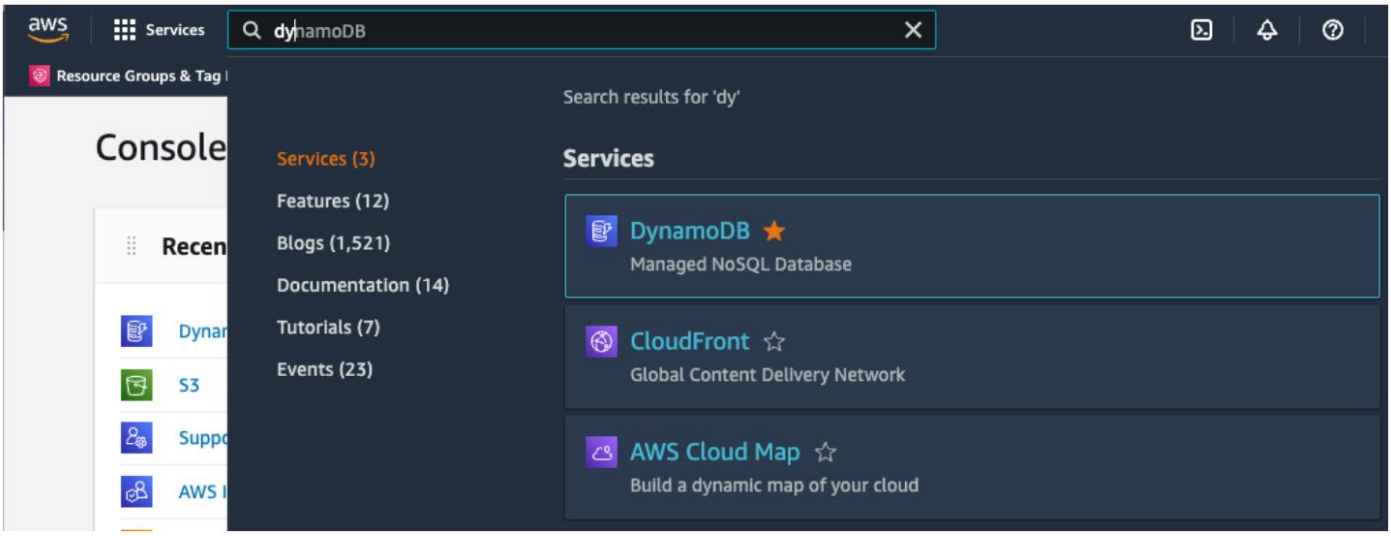
The Hello class has a function named handleRequest that takes an event object and a context object. This is the handler function that Lambda calls when the function is invoked. The Java function runtime gets invocation events from Lambda and passes them to the handler. In the function configuration, the handler value is example.Hello::handleRequest.

To update the function's code, you create a deployment package, which is a .zip file archive that contains your function code. As your function development progresses, you will want to store your function code in source control, add libraries, and automate deployments. Start by creating a deployment package and updating your code at the command line.

The function runtime passes a context object to the handler, in addition to the invocation event. The context object contains additional information about the invocation, the function, and the execution environment. More information is available from environment variables.

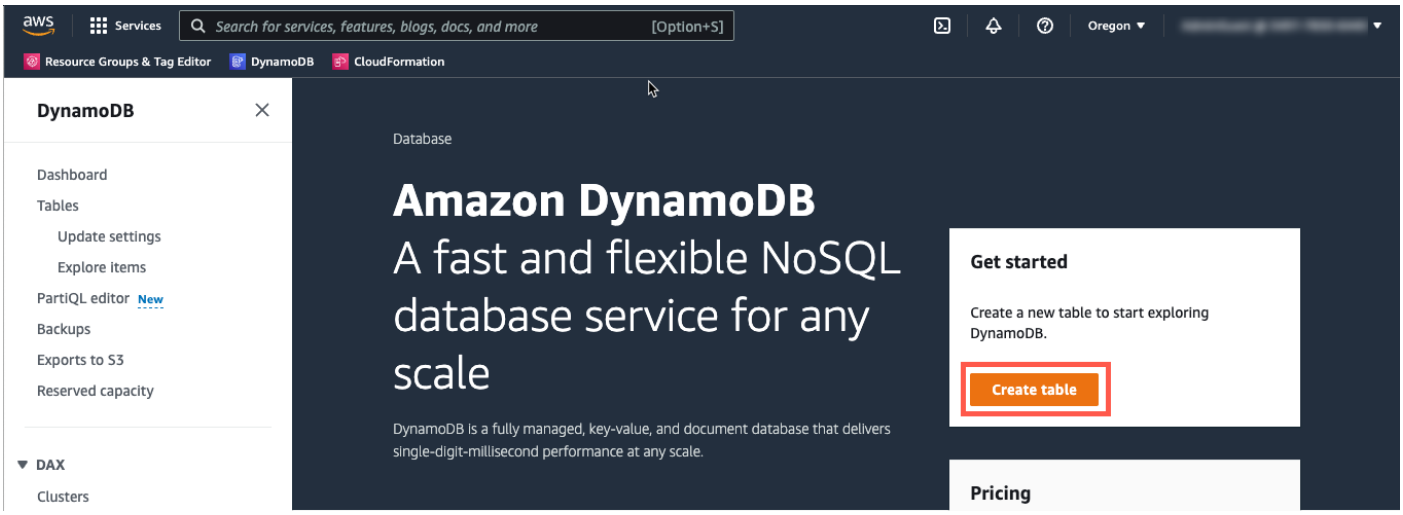
Your Lambda function comes with a CloudWatch Logs log group. The function runtime sends details about each invocation to CloudWatch Logs. It relays any logs that your function outputs during invocation. If your function returns an error, Lambda formats the error and returns it to the invoker.

8 Demonstrate creation of DynamoDB using AWS management console.

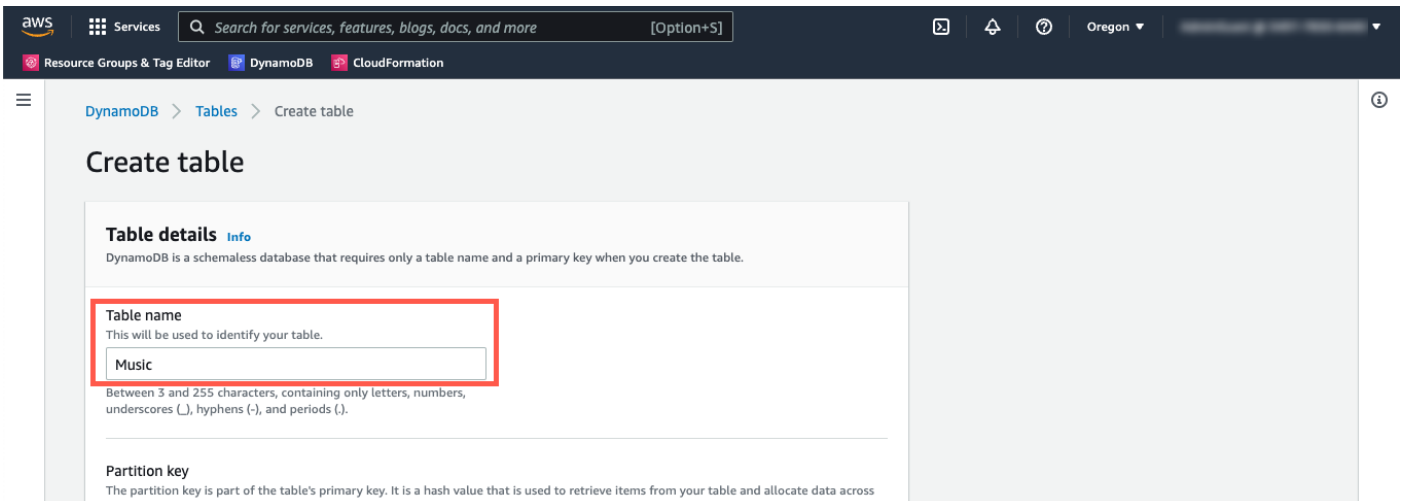


-
- In this step, you will use the DynamoDB console to create a table.

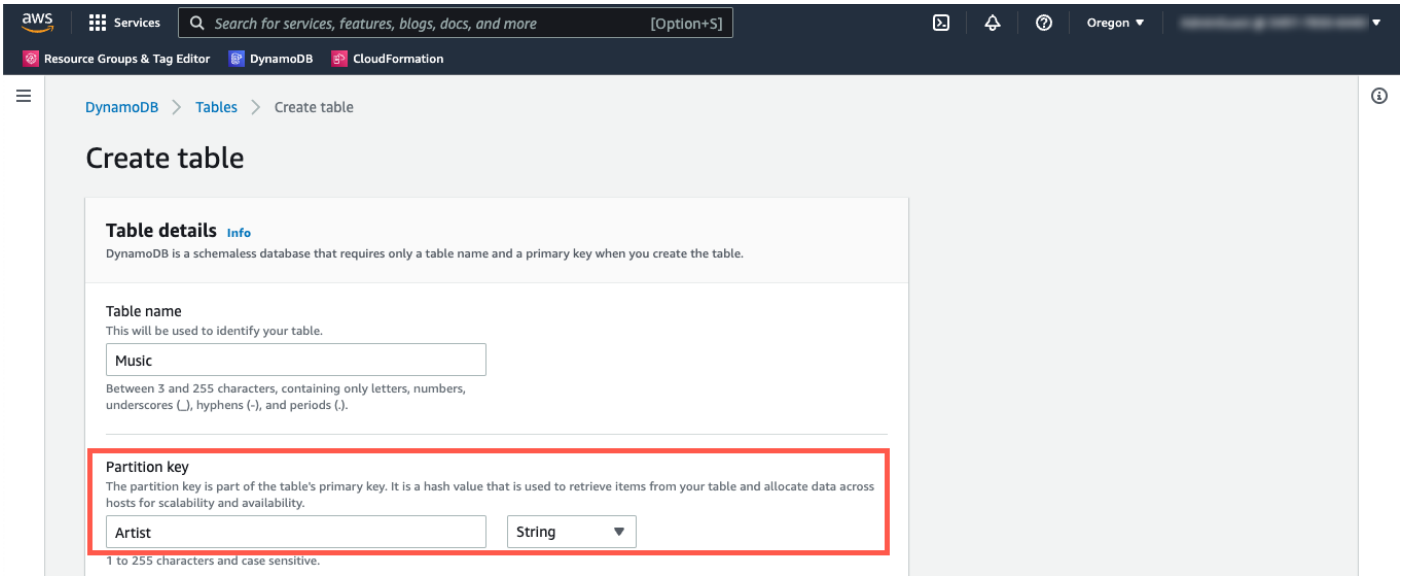
a. In the DynamoDB console, choose Create table.



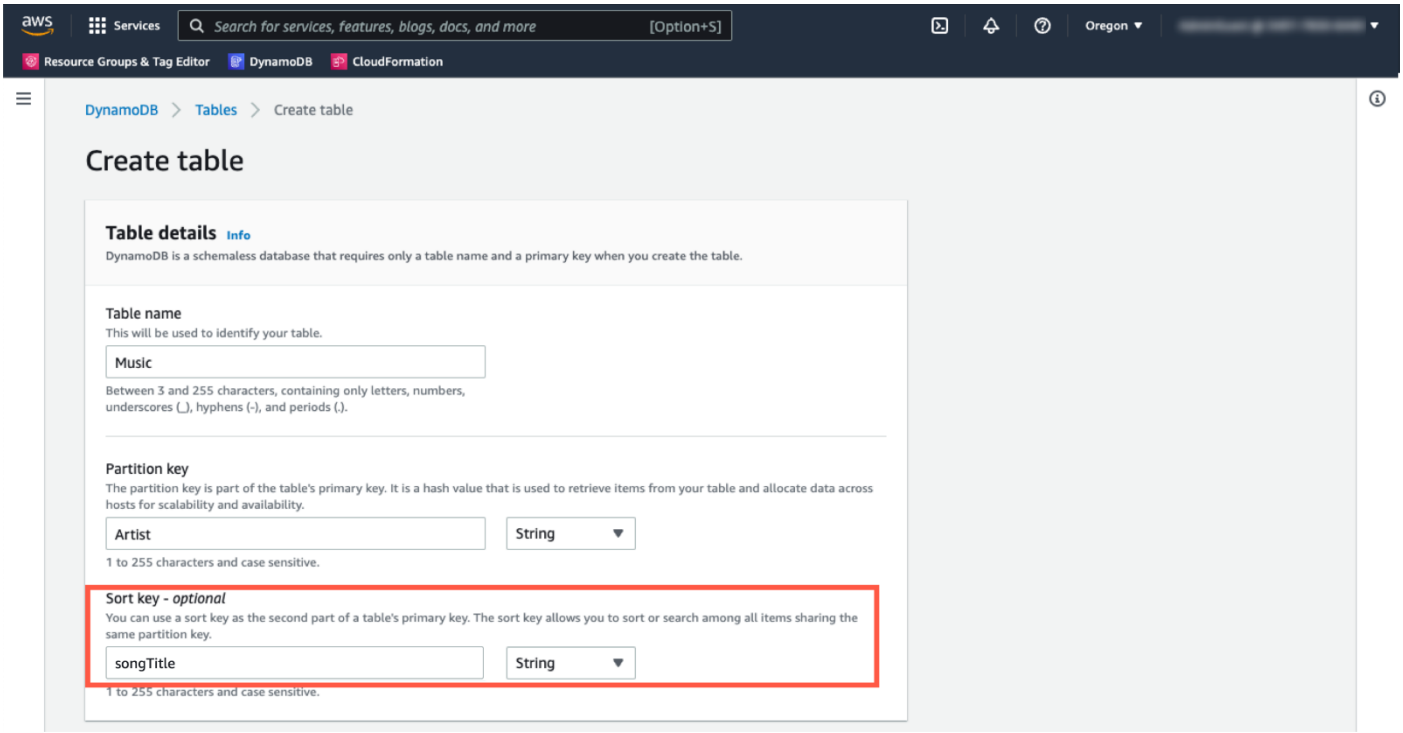
b. We will use a music library as our use case for this tutorial. In the Table name box, type *Music*.



c. The partition key is used to spread data across partitions for scalability. It's important to choose an attribute with a wide range of values and that is likely to have evenly distributed access patterns. Type *Artist* in the Partition key box.



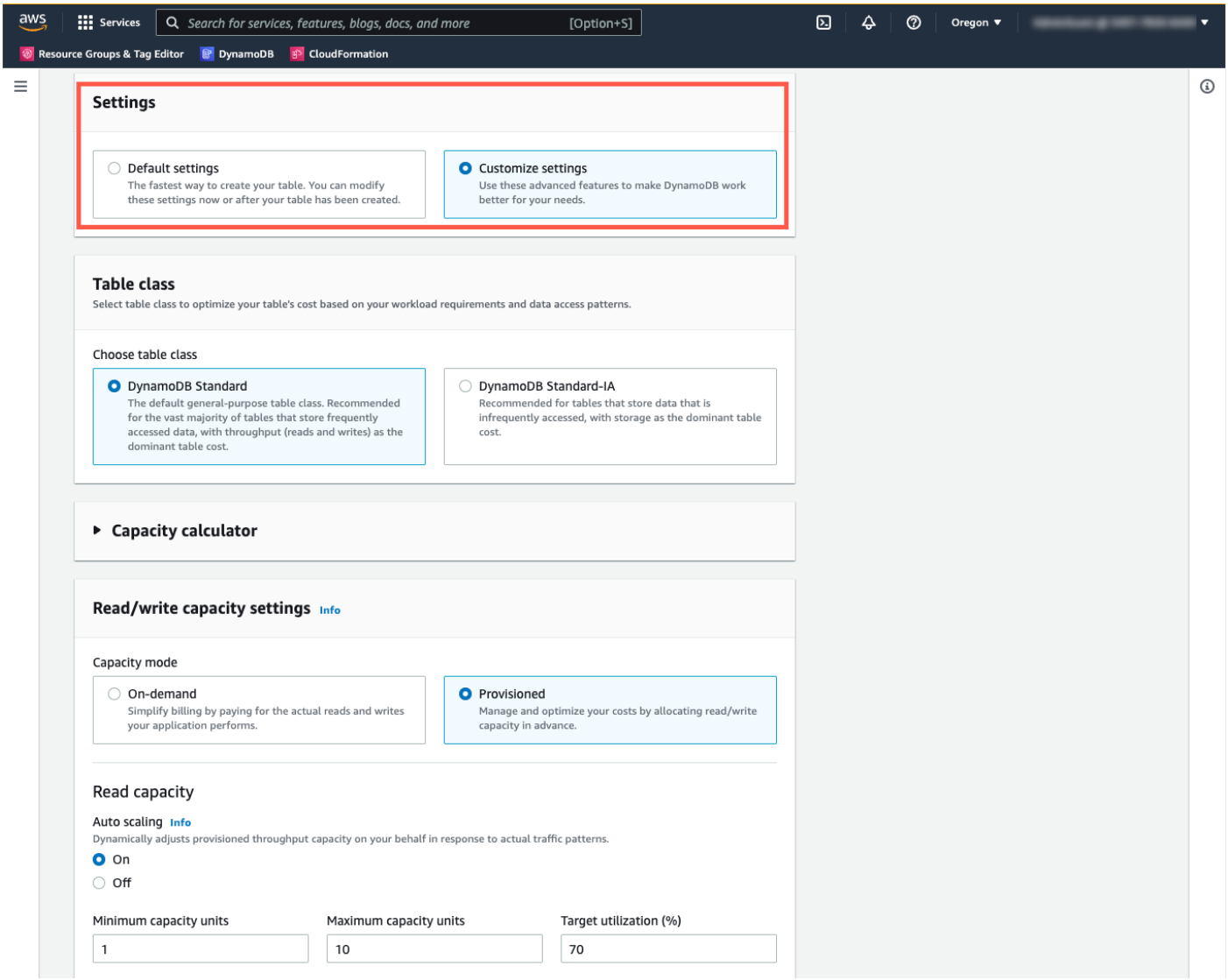
d. Because each artist may write many songs, you can enable easy sorting with a sort key. Enter *songTitle* in the Sort key box.



e. Next, you will enable DynamoDB auto scaling for your table.

DynamoDB auto scaling will change the read and write capacity of your table based on request volume. Using an AWS Identity and Access Management (AWS IAM) role called *DynamoDBAutoscaleRole*, DynamoDB will manage the scaling process on your behalf. DynamoDB creates this role for you the first time you enable auto scaling in an account.

To enable DynamoDB auto scaling for your table, select Customize settings.

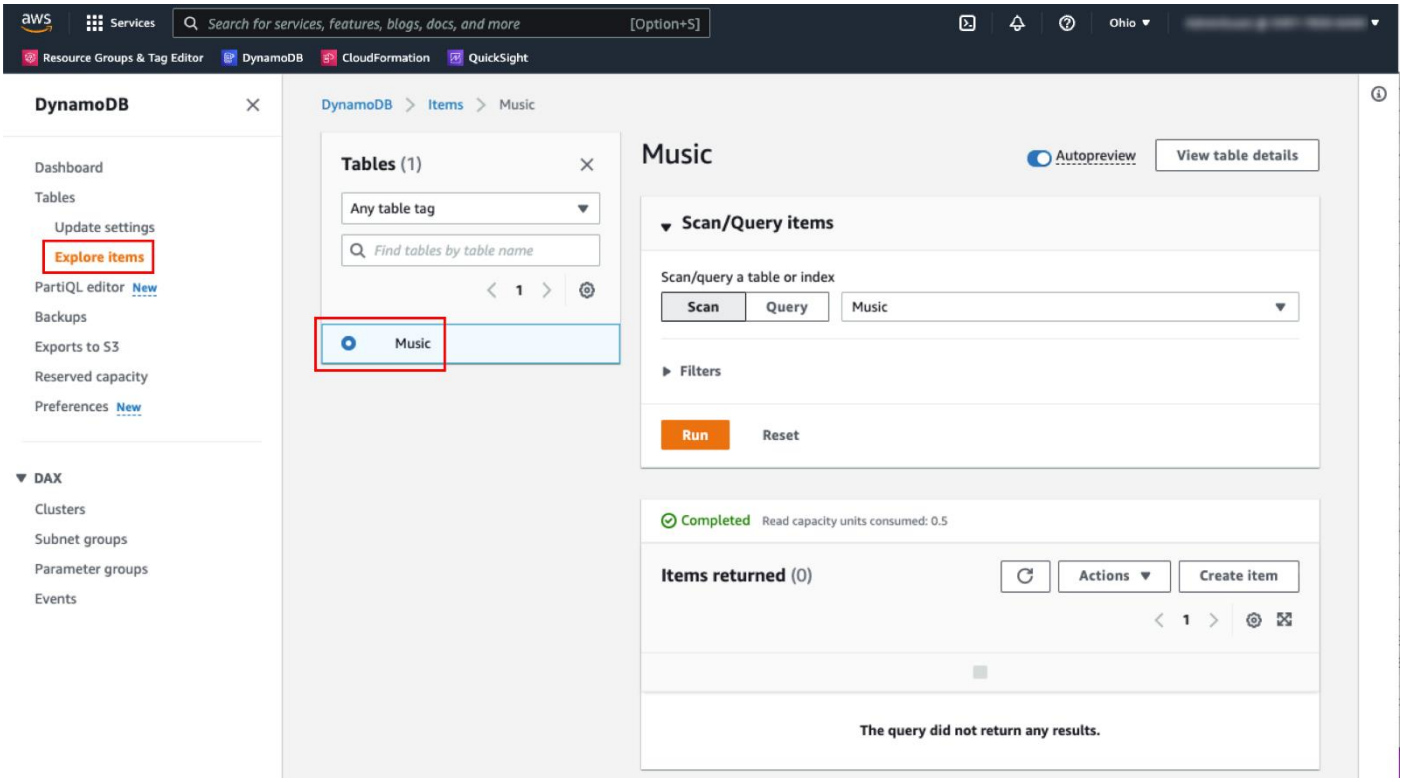


f. Scroll down the screen past Secondary indexes, Estimated read/write capacity cost, Encryption at rest, and Tags to the Create table button. We won't change these settings for the tutorial.

Now choose Create table.

Add data to NoSQL Table

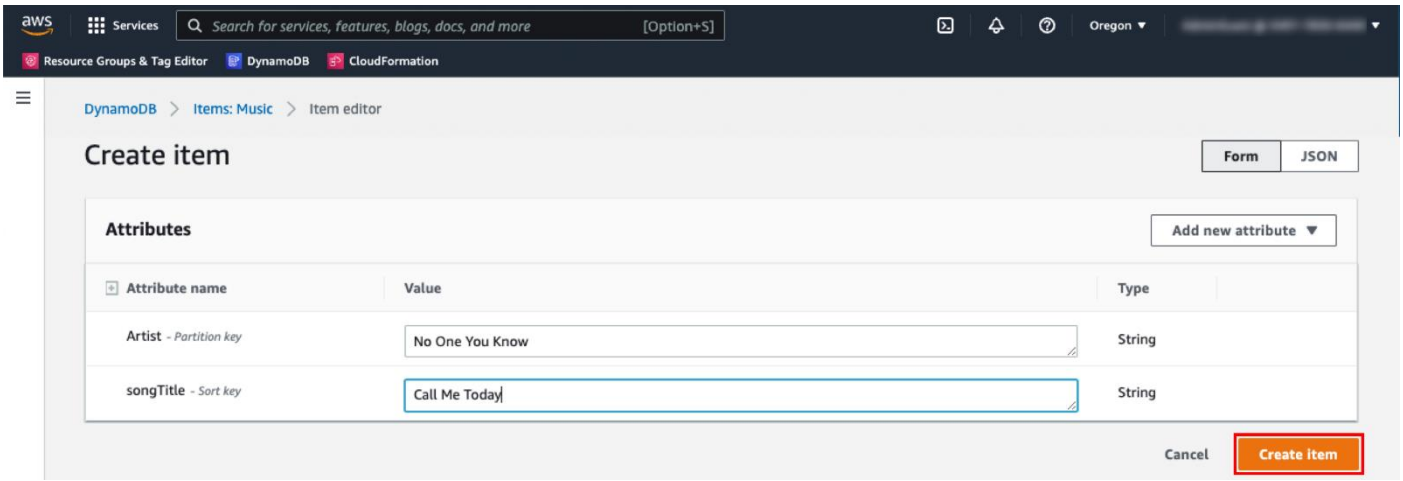
- **Select Explore items from the left menu, then select the radio button next to the Music table.**



b. In the data entry window, type the following:

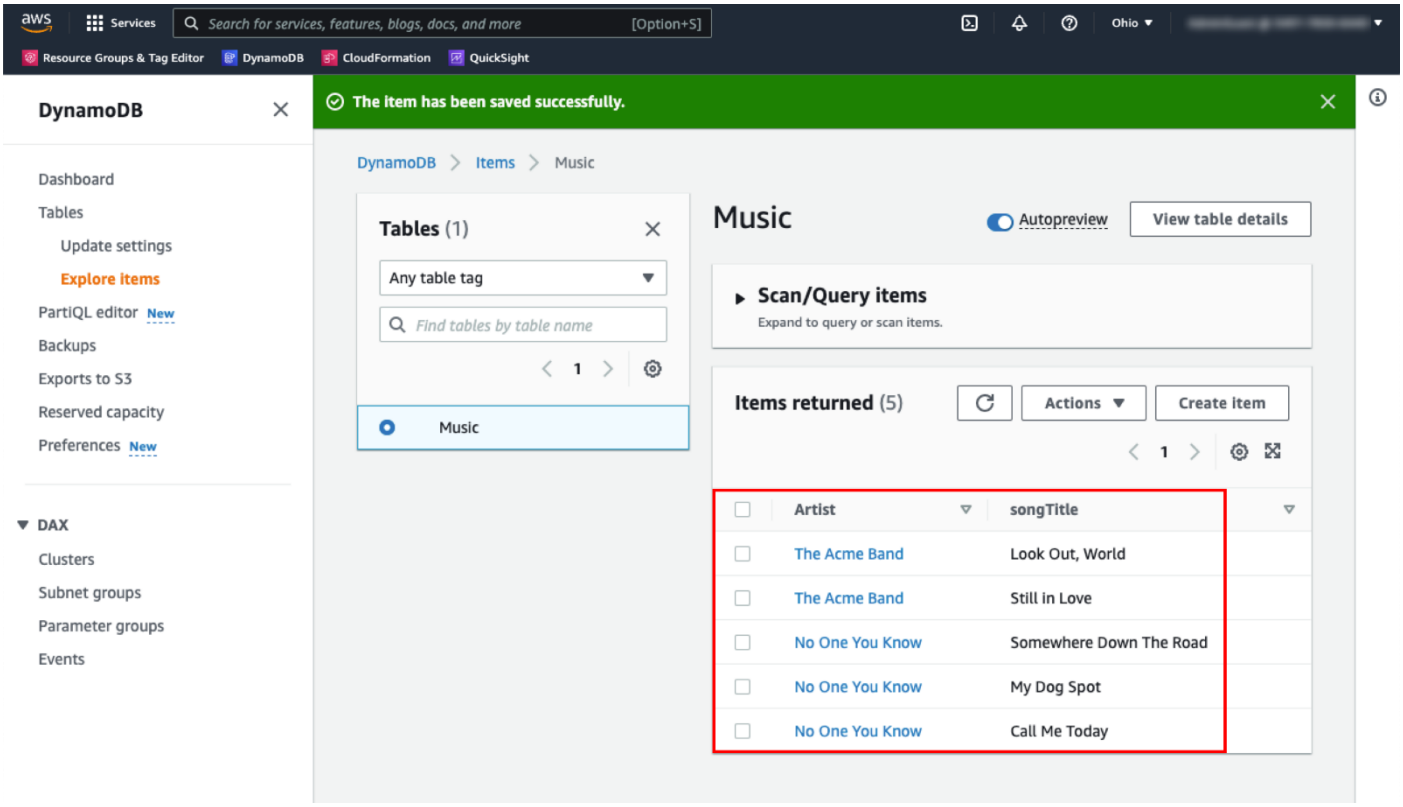
- **For the Artist attribute, type *No One You Know*.**
- **For the songTitle attribute, type *Call Me Today*.**

Choose Create item.



c. Repeat the process to add a few more items to your *Music* table:

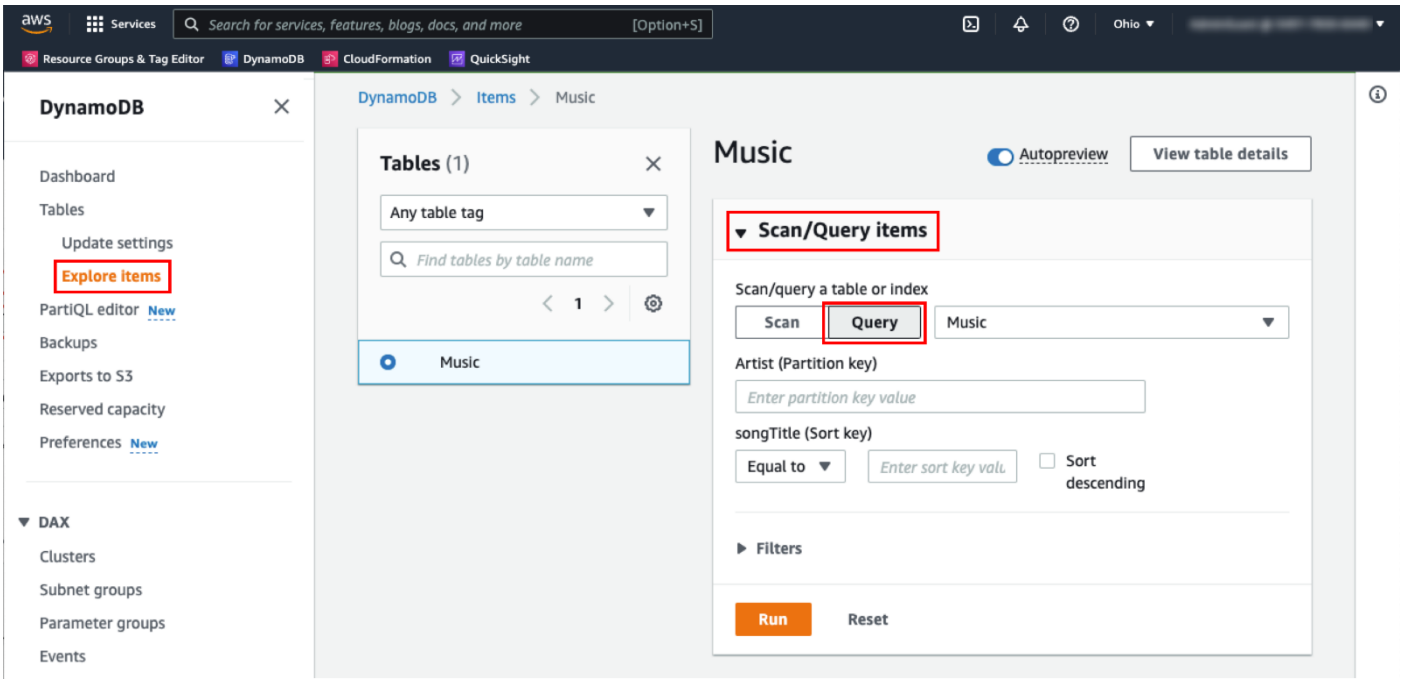
- **Artist: *No One You Know*; songTitle: *My Dog Spot***
- **Artist: *No One You Know*; songTitle: *Somewhere Down The Road***
- **Artist: *The Acme Band*; songTitle: *Still in Love***
- **Artist: *The Acme Band;* songTitle: *Look Out, World***



Query the NoSQL table

- In this step, you will search for data in the table using query operations. In DynamoDB, query operations are efficient and use keys to find data. Scan operations traverse the entire table.

a. Select the arrow next to "Scan/Query items". Then select "Query".

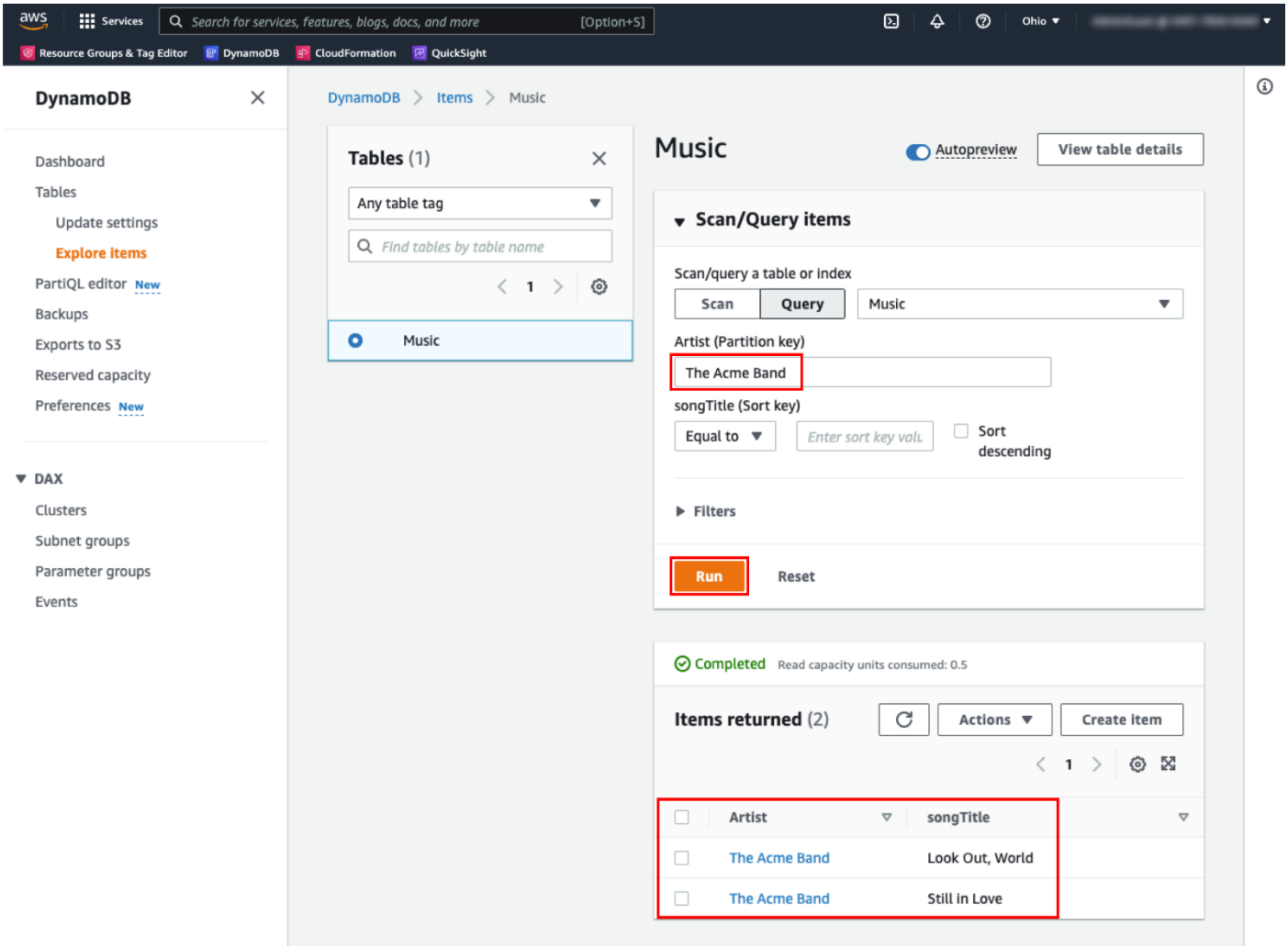


b. You can use the console to query the *Music* table in various ways. For your first query, do the following:

- In the Artist box, type *No One You Know*, and choose Run. All songs performed by *No One You Know* are displayed.

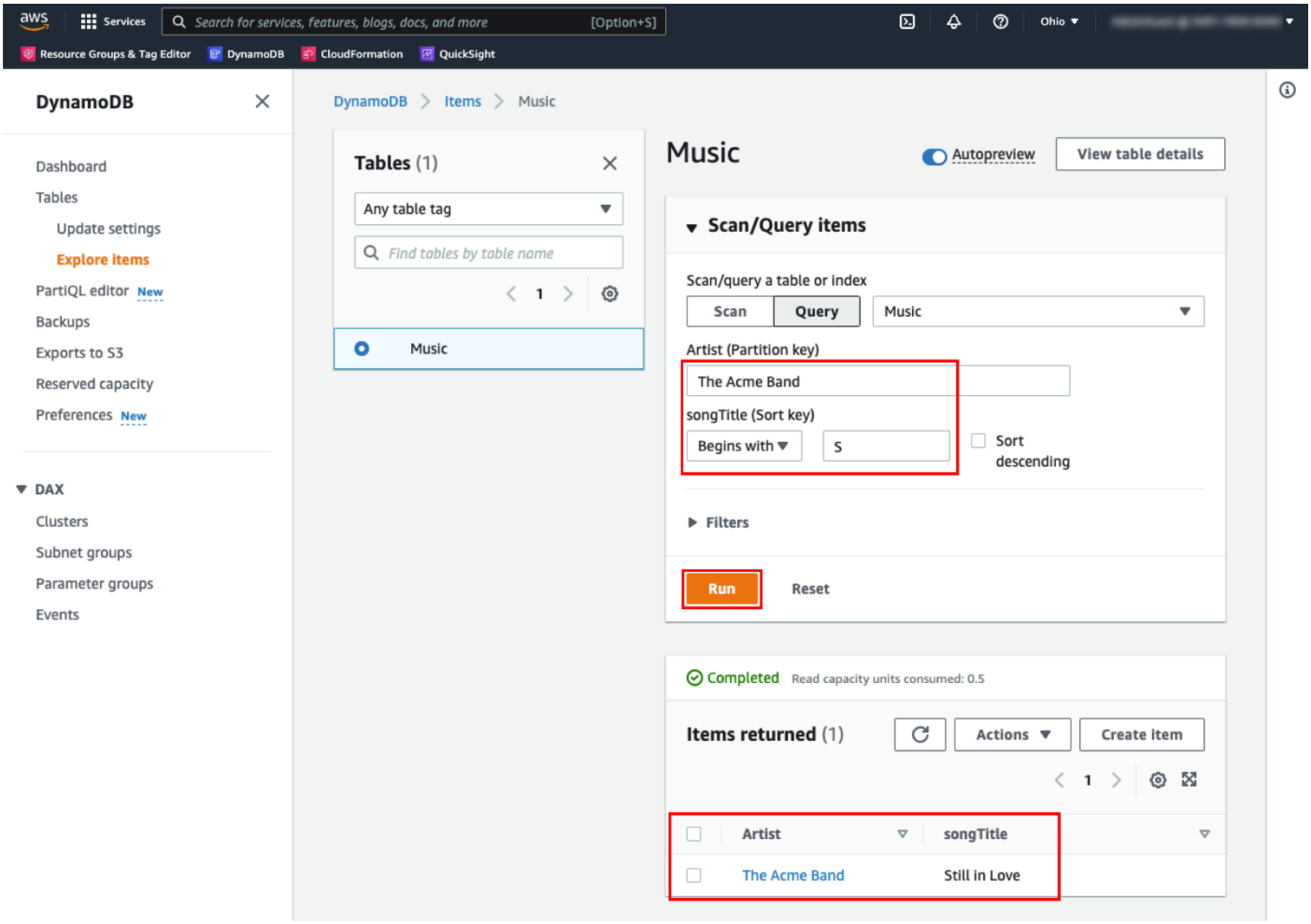
Try another query:

- In the Artist box, type *The Acme Band*, and choose Run. All songs performed by *The Acme Band* are displayed.



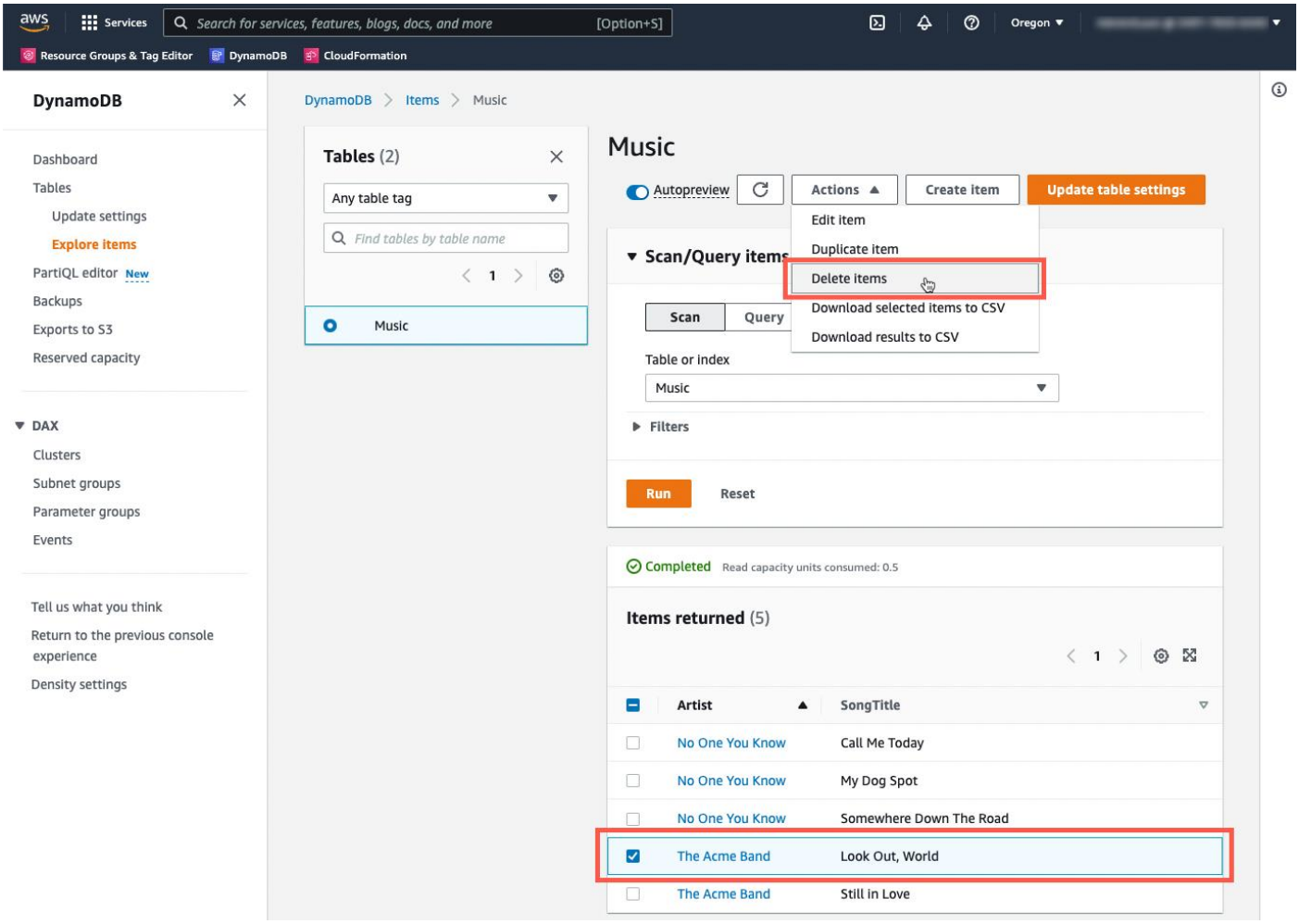
c. Try another query, but this time narrow down the search results:

- In the Artist box, type *The Acme Band.*
- In the songTitle box, select Begins with from the dropdown list and type *S*.
- Choose Run. Only "Still in Love" performed by *The Acme Band* is displayed.



Delete an existing item

Select the checkbox next to *The Acme Band*. In the Actions dropdown list, choose Delete items. You will be asked whether to delete the item. Choose Delete and your item is deleted.

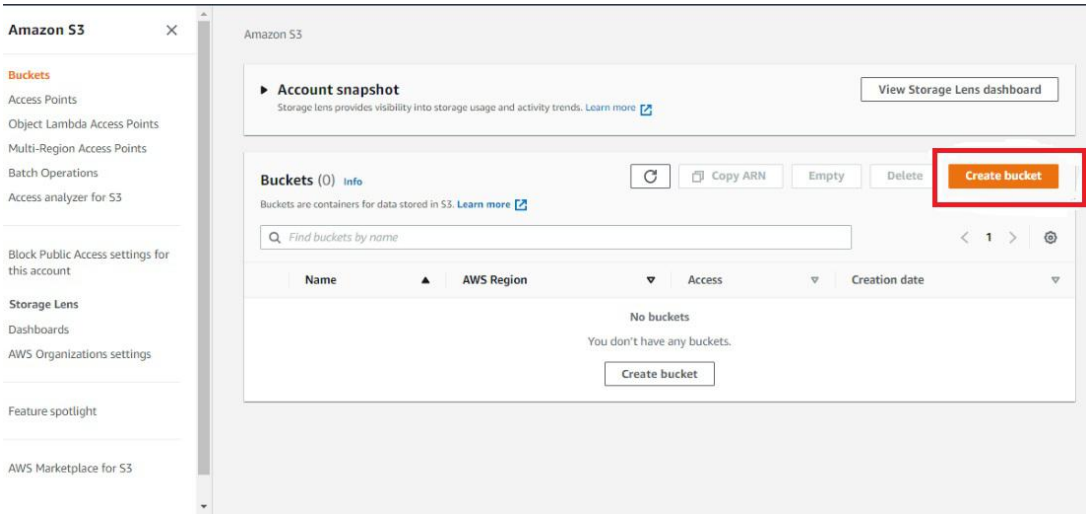

aws
Services
Search for services, features, blogs, docs, and more
[Option+S]
Oregon
Resource Groups & Tag Editor
DynamoDB
CloudFormation
DynamoDB
Dashboard
Tables
Update settings
Explore items
PartiQL editor New
Backups
Exports to S3
Reserved capacity
DAX
Clusters
Subnet groups
Parameter groups
Events
Tell us what you think
Return to the previous console experience
Density settings
DynamoDB > Items > Music
Tables (2)
Any table tag
Find tables by table name
1
Music
Music
Autopreview
Actions
Create item
Update table settings
Edit item
Duplicate item
Delete items
Download selected items to CSV
Download results to CSV
Scan/Query items
Scan
Query
Table or index
Music
Filters
Run
Reset
Completed
Read capacity units consumed: 0.5
Items returned (5)
1
Artist
SongTitle
No One You Know
Call Me Today
No One You Know
My Dog Spot
No One You Know
Somewhere Down The Road
The Acme Band
Look Out, World
The Acme Band
Still in Love

9 Demonstrate hosting a static website using AWS S3 bucket.

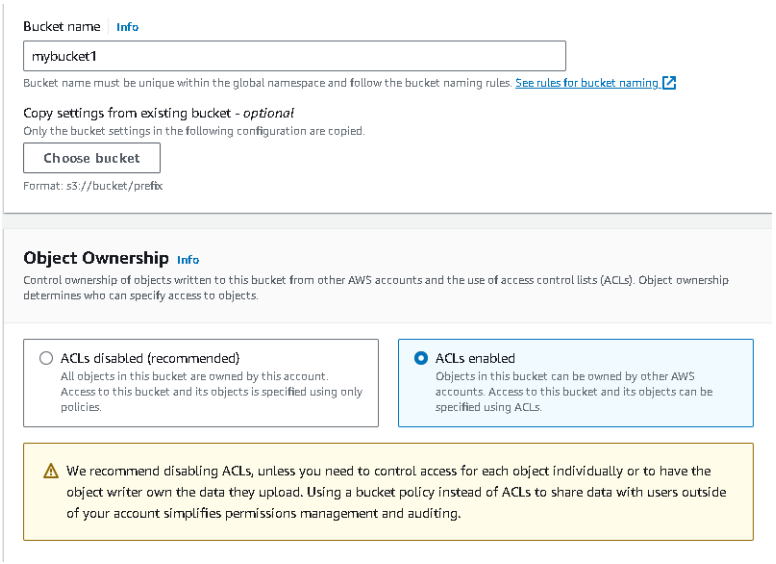
Step 1: Creating a Bucket.

First, we have to launch our S3 instance. Follow these steps for creating a Bucket

- Open the Amazon S3 console by logging into the AWS Management Console at https://console.aws.amazon.com/s3/.
- Click on Create Bucket.

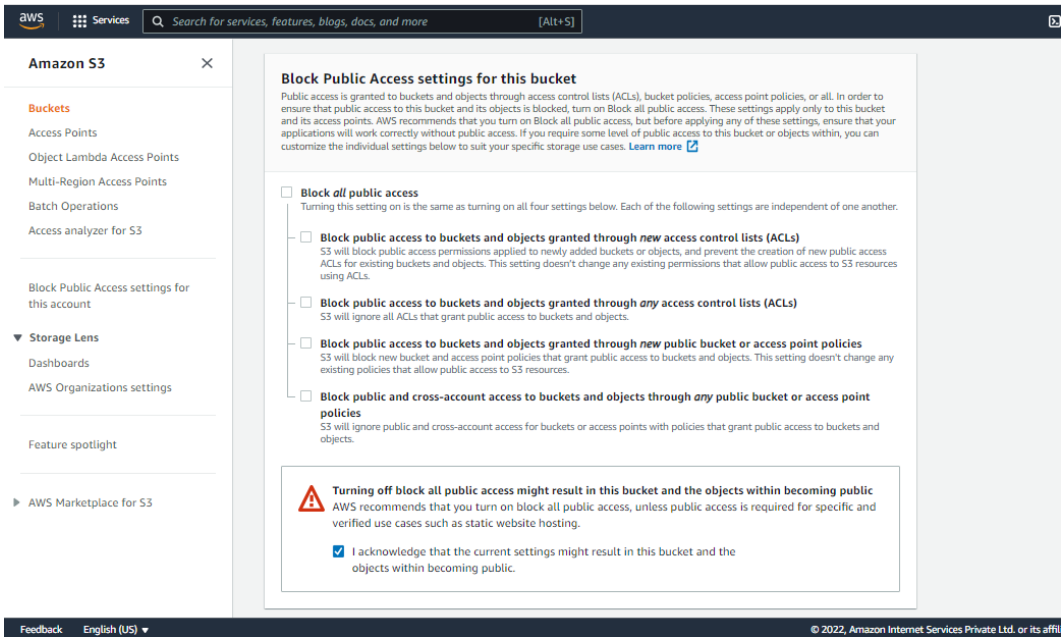


- Choose Bucket Name – Bucket Name Should be Unique
- AWS Region – Choose a region close to you or the region where you want to create the bucket (Example — Mumbai)
- Object Ownership – Enable for making Public, Otherwise disable



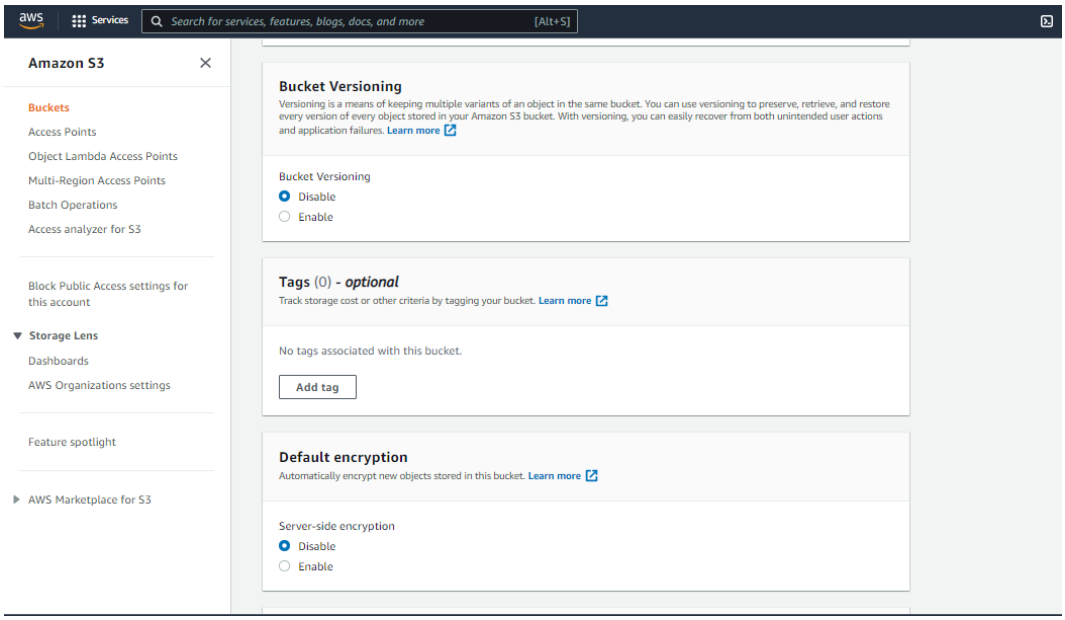
Step 2: Block Public Access settings for the bucket

Uncheck (Block all public access) for the public, otherwise set default. If you uncheck (Block all public keys).

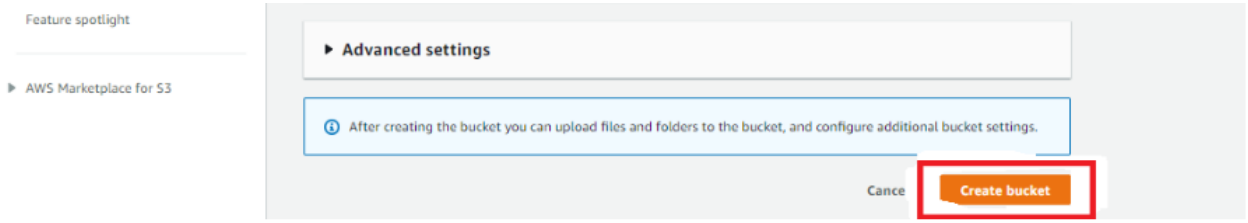


Tags(0) : Optional

Default encryption: Disable

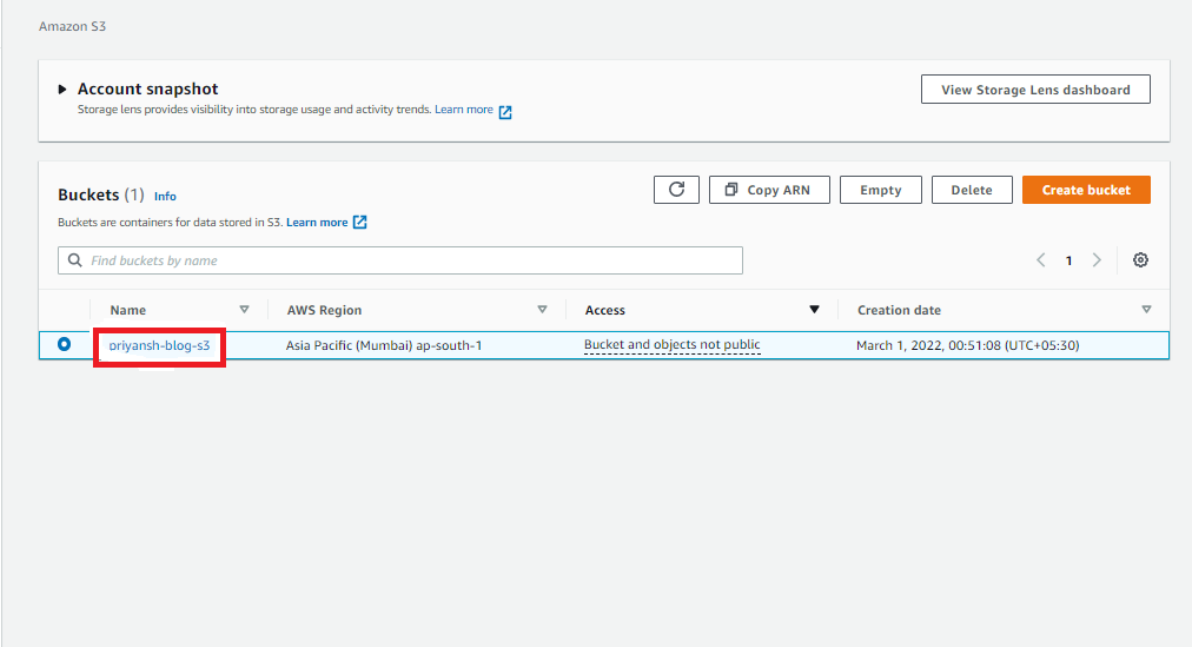


Now, click on Create Bucket

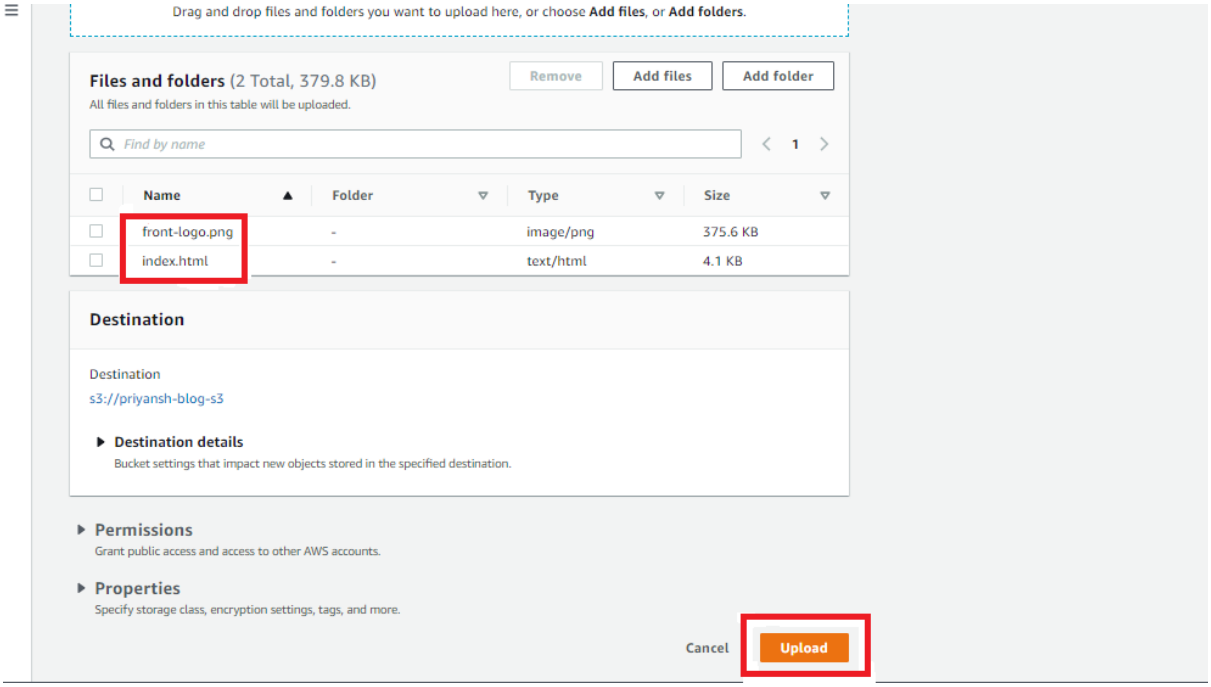


Step 3: Now upload code files

Select Bucket and Click your Bucket Name.



Now, click on upload (then click add File/folder) and select your HTML code file from your PC/Laptop.

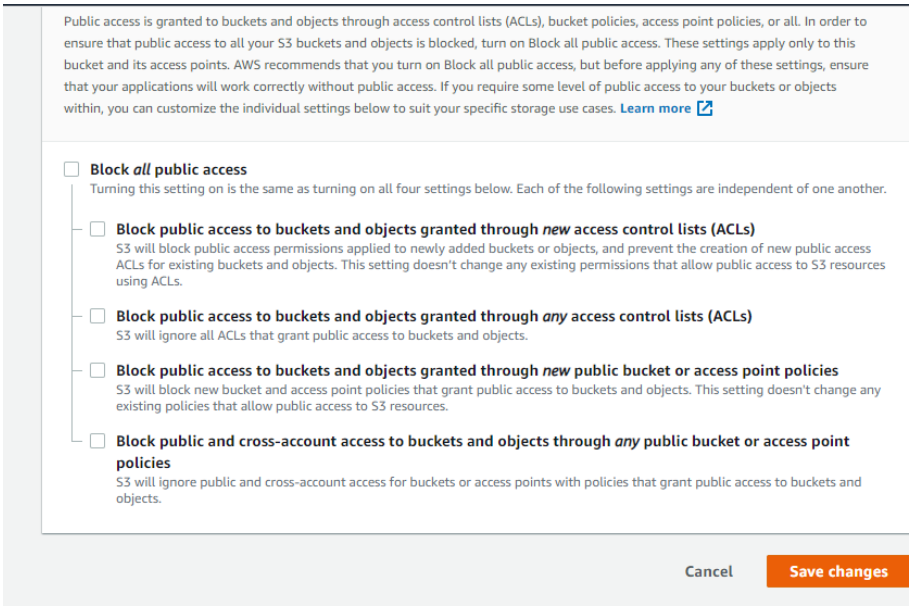


After uploading, click on Close.

Step 4: Once the Files are uploaded successfully, click on Permissions and now follow this Process –

- Block public access:
- Click on Edit, under Bucket Policy.
- Uncheck Block all public access.

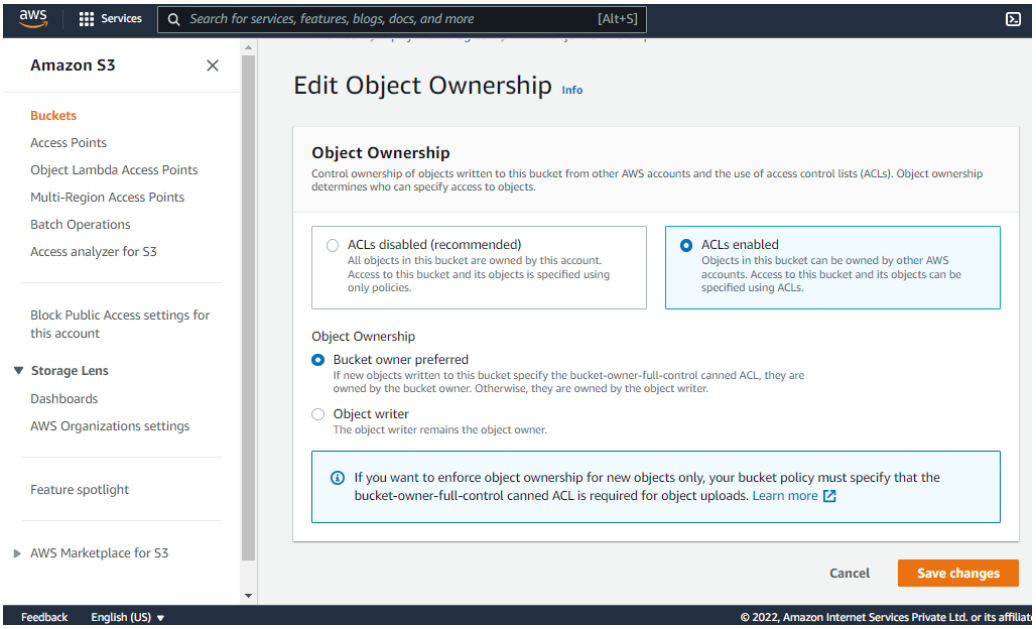
- Save changes then type “confirm”.



- Object Ownership: Click on Edit.->Click on ACLs Enabled.

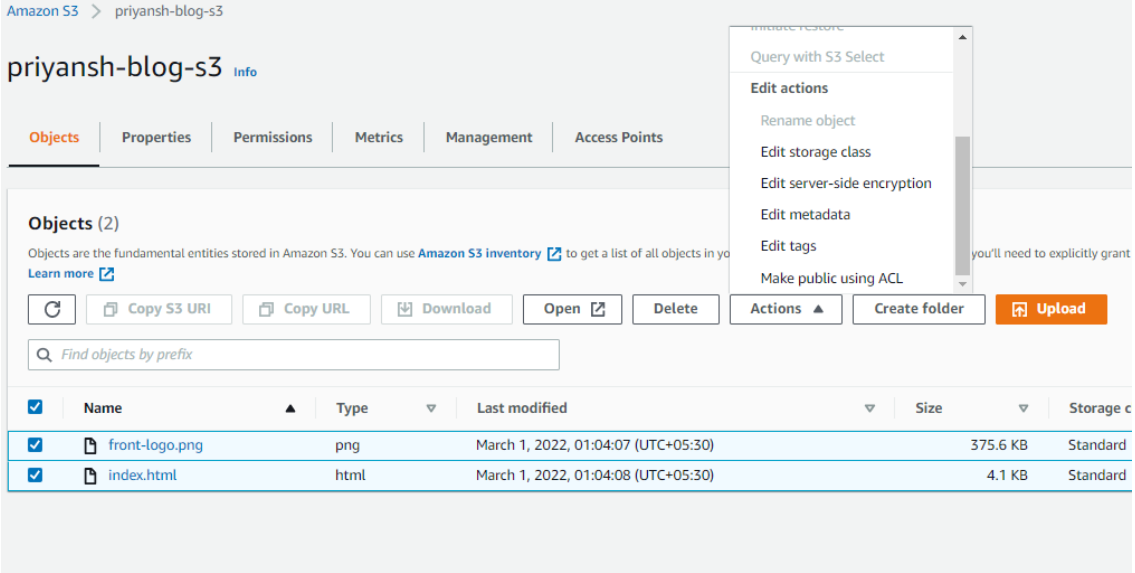
->Check I acknowledge ….. restored.

Choose Save Changes.



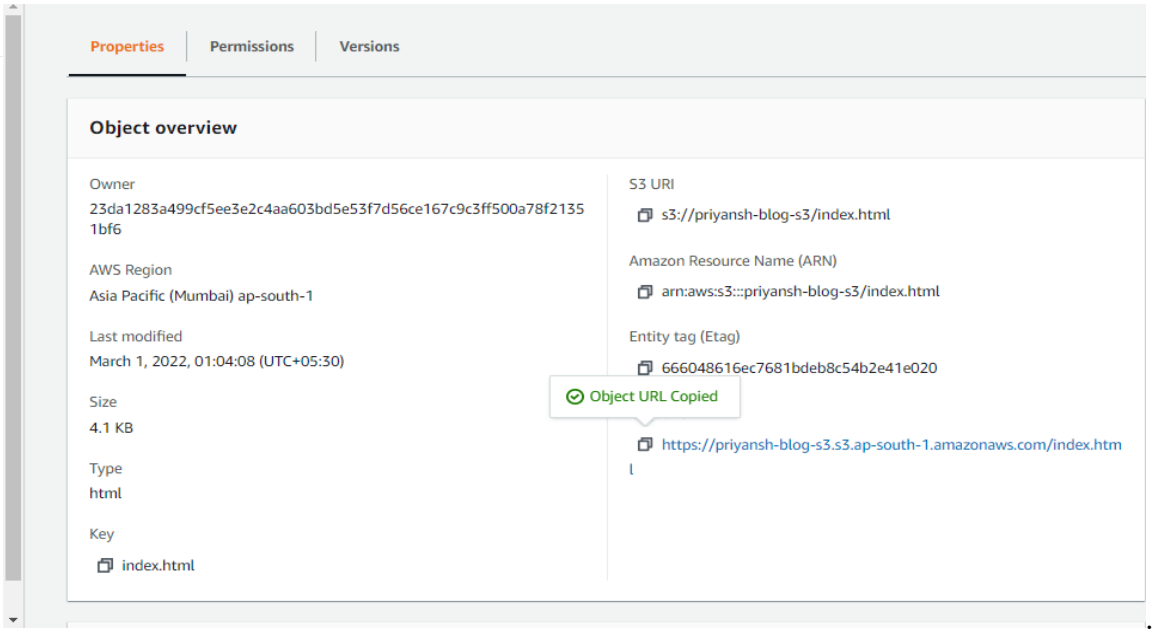
Step 5:- Make public Object

- Now, Click on Objects.
- Select your All Objects.
- Now, Click on Actions.
- Select Make Public Using ACL.
- Now, Click on Make Public and Close.



Step 6: Copy your Object URL

- Now, click on your HTML File Object Name.
- Copy the Object URL



Step 7: Check out your Website!

- Directly Paste this URL into the Other Tab or your other System.
- Congratulation, Now Your Website is available in the Public.

10 Demonstrate Managing AWS S3 bucket with AWS SDK using Java (Cloud 9)

Step 1: Install required tools

In this step, you install a set of Java development tools in your AWS Cloud9 development environment. If you already have a set of Java development tools such as the Oracle JDK or OpenJDK installed in your environment, you can skip ahead to Step 2: Add code. This sample was developed with OpenJDK 8, which you can install in your environment by completing the following procedure.

1. Confirm whether OpenJDK 8 is already installed. To do this, in a terminal session in the AWS Cloud9 IDE, run the command line version of the Java runner with the **-version** option. (To start a new terminal session, on the menu bar, choose **Window**, **New Terminal**.)

java -version

Based on the output of the preceding command, do one of the following:

- If the output states that the java command isn't found, continue with step 2 in this procedure to install OpenJDK 8.
- If the output contains values starting with Java(TM), Java Runtime Environment, Java SE, J2SE, or Java2, the OpenJDK isn't installed or isn't set as the default Java development toolset. Continue with step 2 in this procedure to install OpenJDK 8, and then switch to using OpenJDK 8.
- If the output contains values starting with java version 1.8 and OpenJDK, skip ahead to Step 2: Add code. OpenJDK 8 is installed correctly for this sample.
- If the output contains a java version less than 1.8 and values starting with OpenJDK, continue with step 2 in this procedure to upgrade the installed OpenJDK version to OpenJDK 8.

2. Ensure the latest security updates and bug fixes are installed. To do this, run the yum tool (for Amazon Linux) or the apt tool (for Ubuntu Server) with the **update** command.

For Amazon Linux:

sudo yum -y update

For Ubuntu Server:

sudo apt update

3. Install OpenJDK 8. To do this, run the yum tool (for Amazon Linux) or the apt tool (for Ubuntu Server) with the **install** command, specifying the OpenJDK 8 package.

For Amazon Linux:

sudo yum -y install java-1.8.0-openjdk-devel

For Ubuntu Server:

sudo apt install -y openjdk-8-jdk

For more information, see How to download and install prebuilt OpenJDK packages on the OpenJDK website.

4. Switch or upgrade the default Java development toolset to OpenJDK 8. To do this, run the **update-alternatives** command with the **--config** option. Run this command twice to switch or upgrade the command line versions of the Java runner and compiler.

5. sudo update-alternatives --config java

sudo update-alternatives --config javac

At each prompt, type the selection number for OpenJDK 8 (the one that contains java-1.8).

6. Confirm that the command line versions of the Java runner and compiler are using OpenJDK 8. To do this, run the command line versions of the Java runner and compiler with the -version option.

7. java -version

javac -version

If OpenJDK 8 is installed and set correctly, the Java runner version output contains a value starting with openjdk version 1.8, and the Java compiler version output starts with the value javac 1.8.

Step 2: Add code

In the AWS Cloud9 IDE, create a file with the following code, and save the file with the name hello.java. (To create a file, on the menu bar, choose **File**, **New File**. To save the file, choose **File**, **Save**.)

```
public class hello {


  public static void main(String []args) {
    System.out.println("Hello, World!");


    System.out.println("The sum of 2 and 3 is 5.");


    int sum = Integer.parseInt(args[0]) + Integer.parseInt(args[1]);


    System.out.format("The sum of %s and %s is %s.\n",
      args[0], args[1], Integer.toString(sum));
  }
}
```

Step 3: Build and run the code

1. Use the command line version of the Java compiler to compile the hello.java file into a hello.class file. To do this, using the terminal in the AWS Cloud9 IDE, from the same directory as the hello.java file, run the Java compiler, specifying the hello.java file.

javac hello.java

2. Use the command line version of the Java runner to run the hello.class file. To do this, from the same directory as the hello.class file, run the Java runner, specifying the name of the hello class that was declared in the hello.java file, with two integers to add (for example, 5 and 9).

java hello 5 9

3. Compare your output.
4. Hello, World!
5. The sum of 2 and 3 is 5.

The sum of 5 and 9 is 14.

Step 4: Set up to use the AWS SDK for Java

You can enhance this sample to use the AWS SDK for Java to create an Amazon S3 bucket, list your available buckets, and then delete the bucket you just created.

In this step, you install Apache Maven or Gradle in your environment. Maven and Gradle are common build automation systems that can be used with Java projects. After you install Maven or Gradle, you use it to generate a new Java project. In this new project, you add a reference to the AWS SDK for Java. This AWS SDK for Java provides a convenient way to interact with AWS services such as Amazon S3, from your Java code.

Topics

- Set up with Maven
- Set up with Gradle

Set up with Maven

1. Install Maven in your environment. To see whether Maven is already installed, using the terminal in the AWS Cloud9 IDE, run Maven with the **-version** option.

mvn -version

If successful, the output contains the Maven version number. If Maven is already installed, skip ahead to step 4 in this procedure to use Maven to generate a new Java project in your environment.

2. Install Maven by using the terminal to run the following commands.

For Amazon Linux, the following commands get information about the package repository where Maven is stored, and then use this information to install Maven.

sudo wget http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo

sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo

sudo yum install -y apache-maven

For more information about the preceding commands, see Extra Packages for Enterprise Linux (EPEL) on the Fedora Project Wiki website.

For Ubuntu Server, run the following command instead.

sudo apt install -y maven

3. Confirm the installation by running Maven with the **-version** option.

mvn -version

4. Use Maven to generate a new Java project. To do this, use the terminal to run the following command from the directory where you want Maven to generate the project (for example, the root directory of your environment).

mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

The preceding command creates the following directory structure for the project in your environment.

```
my-app
  |- src
  |   `- main
  |       `- java
  |           `- com
  |               `- mycompany
  |                   `- app
  |                       `-App.java
  |- test
  |   `- java
  |       `- com
  |           `- mycompany
  |               `- app
  |                   `- AppTest.java
  `- pom.xml
```

For more information about the preceding directory structure, see Maven Quickstart Archetype and Introduction to the Standard Directory Layout on the Apache Maven Project website.

5. Modify the Project Object Model (POM) file for the project. (A POM file defines a Maven project's settings.) To do this, from the **Environment** window, open the my-app/pom.xml file. In the editor, replace the file's current contents with the following code, and then save the pom.xml file.
6. <project xmlns="http://maven.apache.org/POM/4.0.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
7. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
8. <modelVersion>4.0.0</modelVersion>
9. <groupId>com.mycompany.app</groupId>
10. <artifactId>my-app</artifactId>
11. <packaging>jar</packaging>

```
12.  <version>1.0-SNAPSHOT</version>
13.  <build>
14.    <plugins>
15.      <plugin>
16.        <groupId>org.apache.maven.plugins</groupId>
17.        <artifactId>maven-assembly-plugin</artifactId>
18.        <version>3.6.0</version>
19.        <configuration>
20.          <descriptorRefs>
21.            <descriptorRef>jar-with-dependencies</descriptorRef>
22.          </descriptorRefs>
23.          <archive>
24.            <manifest>
25.              <mainClass>com.mycompany.app.App</mainClass>
26.            </manifest>
27.          </archive>
28.        </configuration>
29.        <executions>
30.          <execution>
31.            <phase>package</phase>
32.              <goals>
33.                <goal>single</goal>
34.              </goals>
35.          </execution>
36.        </executions>
37.      </plugin>
38.    </plugins>
39.  </build>
40.  <dependencies>
41.    <dependency>
42.      <groupId>junit</groupId>
```

```
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk</artifactId>
    <version>1.11.330</version>
  </dependency>
 </dependencies>
</project>
```

The preceding POM file includes project settings that specify declarations such as the following:

- The artifactid setting of my-app sets the project's root directory name, and the group-id setting of com.mycompany.app sets the com/mycompany/app subdirectory structure and the package declaration in the App.Java and AppTest.java files.
- The artifactId setting of my-app, with the packaging setting of jar, the version setting of 1.0-SNAPSHOT, and the descriptorRef setting of jar-with-dependencies set the output JAR file's name of my-app-1.0-SNAPSHOT-jar-with-dependencies.jar.
- The plugin section declares that a single JAR, which includes all dependencies, will be built.
- The dependency section with the groupId setting of com.amazon.aws and the artifactId setting of aws-java-sdk includes the AWS SDK for Java library files. The AWS SDK for Java version to use is declared by the version setting. To use a different version, replace this version number.

Skip ahead to Step 5: Set up AWS credentials management in your environment.

Set up with Gradle

1. Install Gradle in your environment. To see whether Gradle is already installed, using the terminal in the AWS Cloud9 IDE, run Gradle with the **-version** option.

```
gradle -version
```

If successful, the output contains the Gradle version number. If Gradle is already installed, skip ahead to step 4 in this procedure to use Gradle to generate a new Java project in your environment.

2. Install Gradle by using the terminal to run the following commands. These commands install and run the SDKMAN! tool, and then use SDKMAN! to install the latest version of Gradle.
3. curl -s "https://get.sdkman.io" | bash
4. source "$HOME/.sdkman/bin/sdkman-init.sh"

```
sdk install gradle
```

For more information about the preceding commands, see Installation on the SDKMAN! website and Install with a package manager on the Gradle website.

5. Confirm the installation by running Gradle with the **-version** option.

gradle -version

6. Use Gradle to generate a new Java project in your environment. To do this, use the terminal to run the following commands to create a directory for the project, and then switch to that directory.
7. mkdir my-app

cd my-app

8. Run the following command to have Gradle generate a new Java application project in the my-app directory in your environment.

gradle init --type java-application

The preceding command creates the following directory structure for the project in your environment.

```
my-app
 |- .gradle
 |   `- (various supporting project folders and files)
 |- gradle
 |   `- (various supporting project folders and files)
 |- src
 |   |- main
 |   |   `- java
 |   |       `- App.java
 |   `- test
 |       `- java
 |           `- AppTest.java
 |- build.gradle
 |- gradlew
 |- gradlew.bat
 `- settings.gradle
```

9. Modify the AppTest.java for the project. (If you do not do this, the project might not build or run as expected). To do this, from the **Environment** window, open the my-app/src/test/java/AppTest.java file. In the editor, replace the file's current contents with the following code, and then save the AppTest.java file.
10. import org.junit.Test;

```
11. import static org.junit.Assert.*;
12.
13. public class AppTest {
14.   @Test public void testAppExists () {
15.     try {
16.       Class.forName("com.mycompany.app.App");
17.     } catch (ClassNotFoundException e) {
18.       fail("Should have a class named App.");
19.     }
20.   }
}
```

21. Modify the build.gradle file for the project. (A build.gradle file defines a Gradle project's settings.) To do this, from the **Environment** window, open the my-app/build.gradle file. In the editor, replace the file's current contents with the following code, and then save the build.gradle file.

```
22. apply plugin: 'java'
23. apply plugin: 'application'
24.
25. repositories {
26.   jcenter()
27.   mavenCentral()
28. }
29.
30. buildscript {
31.   repositories {
32.     mavenCentral()
33.   }
34.   dependencies {
35.     classpath "io.spring.gradle:dependency-management-plugin:1.0.3.RELEASE"
36.   }
37. }
38.
39. apply plugin: "io.spring.dependency-management"
```

```
40.
41. dependencyManagement {
42.   imports {
43.     mavenBom 'com.amazonaws:aws-java-sdk-bom:1.11.330'
44.   }
45. }
46.
47. dependencies {
48.   compile 'com.amazonaws:aws-java-sdk-s3'
49.   testCompile group: 'junit', name: 'junit', version: '4.12'
50. }
51.
52. run {
53.   if (project.hasProperty("appArgs")) {
54.     args Eval.me(appArgs)
55.   }
56. }
57.
mainClassName = 'App'
```

The preceding build.gradle file includes project settings that specify declarations such as the following:

- The io.spring.dependency-management plugin is used to import the AWS SDK for Java Maven Bill of Materials (BOM) to manage AWS SDK for Java dependencies for the project. classpath declares the version to use. To use a different version, replace this version number.
- com.amazonaws:aws-java-sdk-s3 includes the Amazon S3 portion of the AWS SDK for Java library files. mavenBom declares the version to use. If you want to use a different version, replace this version number.

Step 5: Set up AWS credentials management in your environment

Each time you use the AWS SDK for Java to call an AWS service, you must provide a set of AWS credentials with the call. These credentials determine whether the AWS SDK for Java has the appropriate permissions to make that call. If the credentials don't cover the appropriate permissions, the call will fail.

In this step, you store your credentials within the environment. To do this, follow the instructions in Calling AWS services from an environment in AWS Cloud9, and then return to this topic.

For additional information, see Set up AWS Credentials and Region for Development in the *AWS SDK for Java Developer Guide*.

Step 6: Add AWS SDK code

In this step, you add code to interact with Amazon S3 to create a bucket, list your available buckets, and then delete the bucket you just created.

From the **Environment** window, open the my-app/src/main/java/com/mycompany/app/App.java file for Maven or the my-app/src/main/java/App.java file for Gradle. In the editor, replace the file's current contents with the following code, and then save the App.java file.

```
package com.mycompany.app;


import com.amazonaws.auth.profile.ProfileCredentialsProvider;

import com.amazonaws.services.s3.AmazonS3;

import com.amazonaws.services.s3.AmazonS3ClientBuilder;

import com.amazonaws.services.s3.model.AmazonS3Exception;

import com.amazonaws.services.s3.model.Bucket;

import com.amazonaws.services.s3.model.CreateBucketRequest;


import java.util.List;


public class App {


   private static AmazonS3 s3;


   public static void main(String[] args) {
      if (args.length < 2) {
         System.out.format("Usage: <the bucket name> <the AWS Region to use>\n" +
               "Example: my-test-bucket us-east-2\n");
         return;
      }


      String bucket_name = args[0];
      String region = args[1];
```

```
s3 = AmazonS3ClientBuilder.standard()
        .withCredentials(new ProfileCredentialsProvider())
        .withRegion(region)
        .build();

// List current buckets.
ListMyBuckets();

// Create the bucket.
if (s3.doesBucketExistV2(bucket_name)) {
    System.out.format("\nCannot create the bucket. \n" +
            "A bucket named '%s' already exists.", bucket_name);
    return;
} else {
    try {
        System.out.format("\nCreating a new bucket named '%s'...\n\n", bucket_name);
        s3.createBucket(new CreateBucketRequest(bucket_name, region));
    } catch (AmazonS3Exception e) {
        System.err.println(e.getErrorMessage());
    }
}

// Confirm that the bucket was created.
ListMyBuckets();

// Delete the bucket.
try {
    System.out.format("\nDeleting the bucket named '%s'...\n\n", bucket_name);
    s3.deleteBucket(bucket_name);
} catch (AmazonS3Exception e) {
    System.err.println(e.getErrorMessage());
```

```
        }

        // Confirm that the bucket was deleted.
        ListMyBuckets();

    }

    private static void ListMyBuckets() {
        List<Bucket> buckets = s3.listBuckets();
        System.out.println("My buckets now are:");

        for (Bucket b : buckets) {
            System.out.println(b.getName());
        }
    }

}
```

Step 7: Build and run the AWS SDK code

To run the code from the previous step, run the following commands from the terminal. These commands use Maven or Gradle to create an executable JAR file for the project, and then use the Java runner to run the JAR. The JAR runs with the name of the bucket to create in Amazon S3 (for example, my-test-bucket) and the ID of the AWS Region to create the bucket in as input (for example, us-east-2).

For Maven, run the following commands.

```
cd my-app
mvn package
java -cp target/my-app-1.0-SNAPSHOT-jar-with-dependencies.jar com.mycompany.app.App my-test-bucket us-east-2
```

For Gradle, run the following commands.

```
gradle build
gradle run -PappArgs="['my-test-bucket', 'us-east-2']"
```

Compare your results to the following output.

My buckets now are:

Creating a new bucket named 'my-test-bucket'...

My buckets now are:

my-test-bucket

Deleting the bucket named 'my-test-bucket'...

My buckets now are

Java code for some operations on AWS S3 bucket:

Create Bucket:

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import com.amazonaws.services.s3.model.Bucket;

import java.util.List;
if (s3.doesBucketExistV2(bucket_name)) {
   System.out.format("Bucket %s already exists.\n", bucket_name);
   b = getBucket(bucket_name);
} else {
   try {
      b = s3.createBucket(bucket_name);
   } catch (AmazonS3Exception e) {
      System.err.println(e.getErrorMessage());
   }
}
return b;
```

List Buckets:

```
import com.amazonaws.regions.Regions;
```

```
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.Bucket;

import java.util.List;
List<Bucket> buckets = s3.listBuckets();
System.out.println("Your {S3} buckets are:");
for (Bucket b : buckets) {
   System.out.println("* " + b.getName());
}
```