

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:

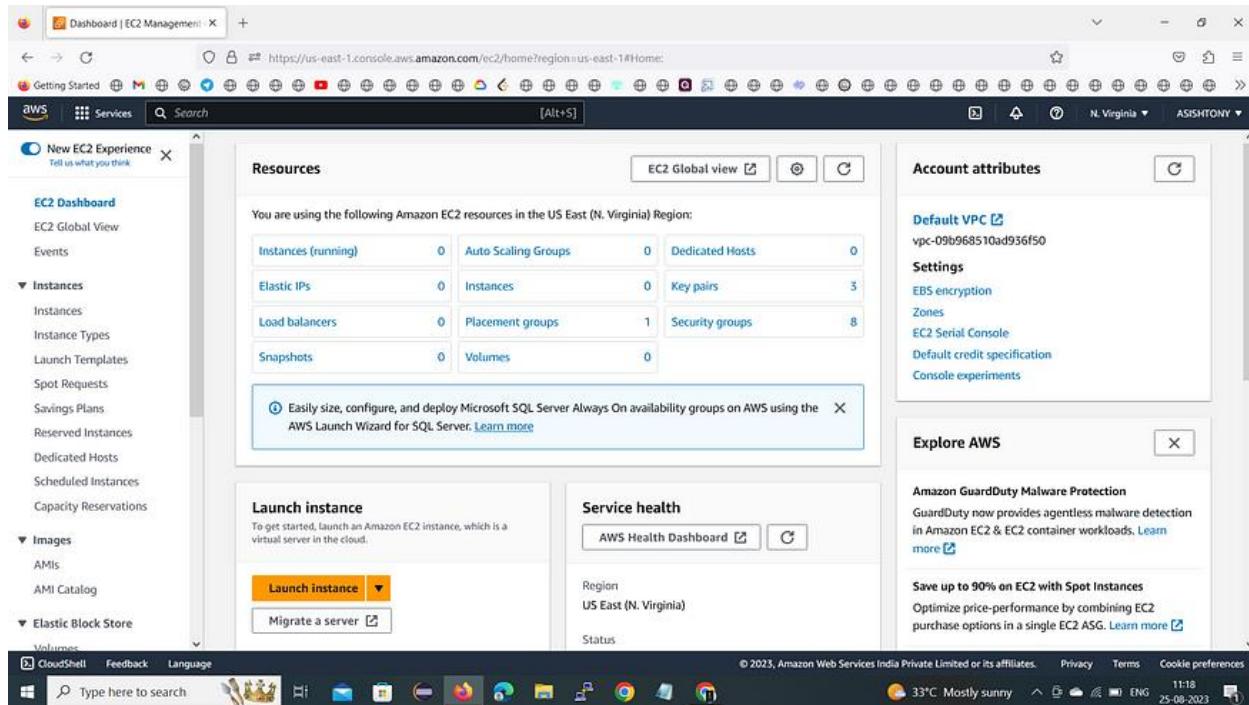
1. An Amazon Web Services (AWS) account: You'll need an active AWS account to create EC2 instances.
2. AWS Console Access: Log in to the AWS Management Console.

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.

The screenshot shows the AWS CloudWatch Metrics dashboard. At the top, there are two tabs: 'AWS Lambda' (selected) and 'Amazon API Gateway'. Below the tabs, there are two main sections: 'Metrics' and 'CloudWatch Metrics Insights'. The 'Metrics' section displays a chart for 'Lambda Metrics' over a 1-hour period, with data points for 'Invocations' and 'Errors'. The 'CloudWatch Metrics Insights' section shows a preview of a log entry from the 'AWS Lambda Metrics Insights' log stream.

Step 4: Name your Instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

The screenshot shows the 'Name and tags' step in the AWS EC2 instance launch wizard. It includes fields for 'Name' (with placeholder 'e.g. My Web Server') and 'Add additional tags'.

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.

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

▼ Application and OS Images (Amazon Machine Image) Info

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

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

Quick Start



Amazon
Linux




macOS




Ubuntu




Windows




Red Hat




SUSE Li




Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI
ami-08a52ddb321b32a8c (64-bit (x86)) / ami-0decaa87ef4609834 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Description

Amazon Linux 2023 AMI 2023.1.20230809.0 x86_64 HVM kernel-6.1

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.

- 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.
- Click on the instance type you want to use. You can find details about each instance type by clicking on the information icon.

▼ Instance type [Info](#)

Instance type

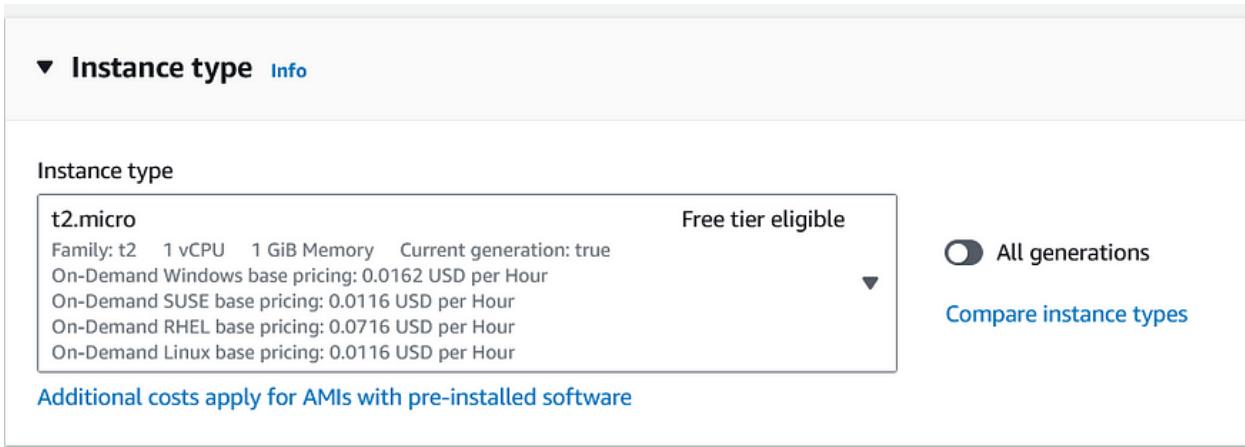
t2.micro Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true
 On-Demand Windows base pricing: 0.0162 USD per Hour
 On-Demand SUSE base pricing: 0.0116 USD per Hour
 On-Demand RHEL base pricing: 0.0716 USD per Hour
 On-Demand Linux base pricing: 0.0116 USD per Hour

Additional costs apply for AMIs with pre-installed software

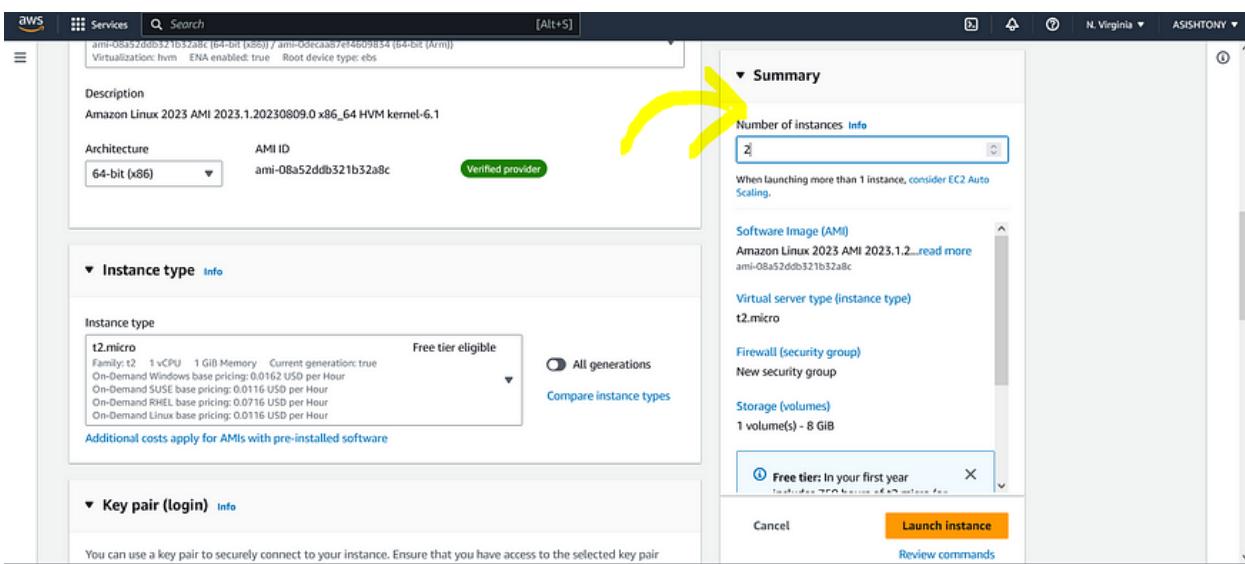
All generations

[Compare instance types](#)



Step 7: Configure Instance Details

- Configure the number of instances you want to launch.



Number of instances [Info](#)

When launching more than 1 instance, consider EC2 Auto Scaling.

Software Image (AMI)
 Amazon Linux 2023 AMI 2023.1.20230809.0 x86_64 HVM kernel-6.1

Virtual server type (instance type)
t2.micro

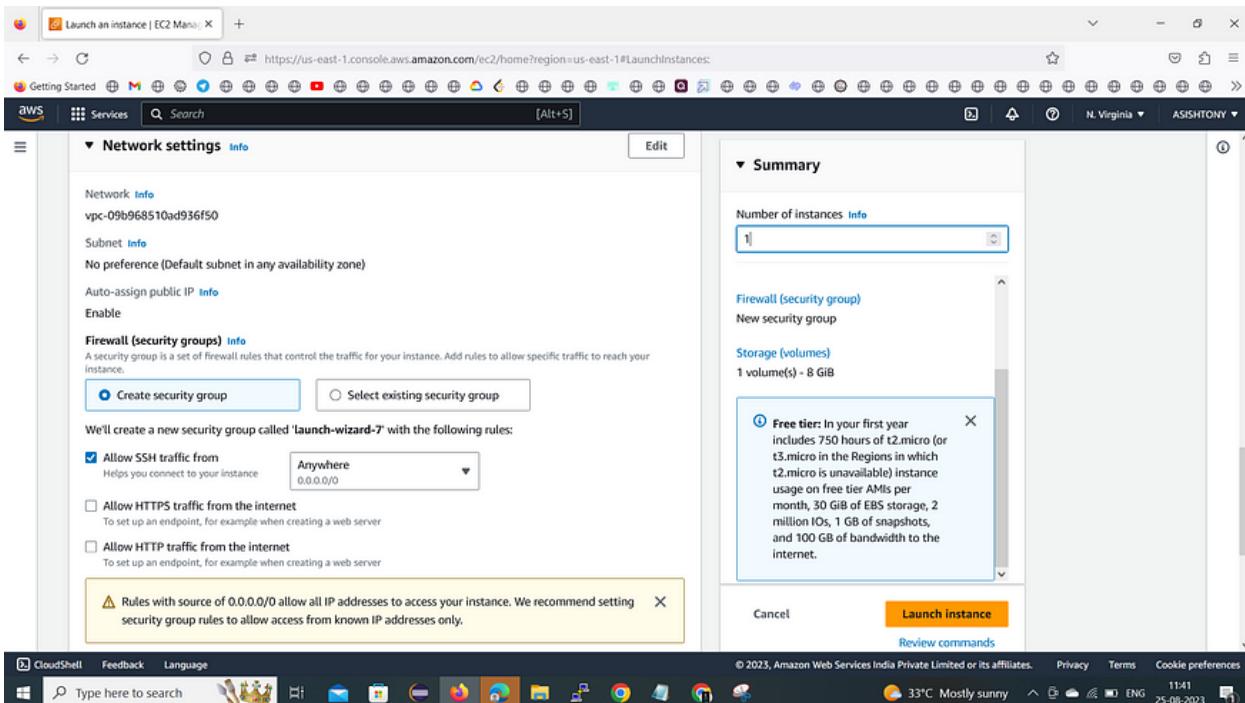
Firewall (security group)
 New security group

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

Free tier: In your first year

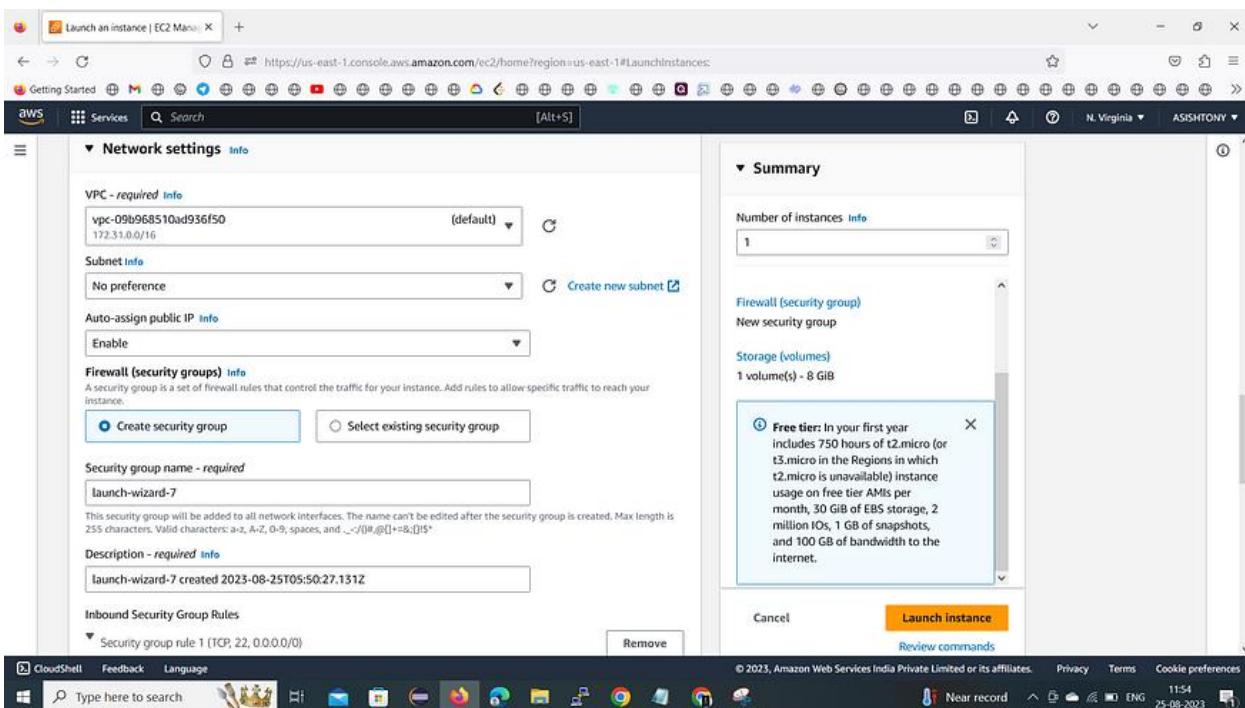
[Launch instance](#)

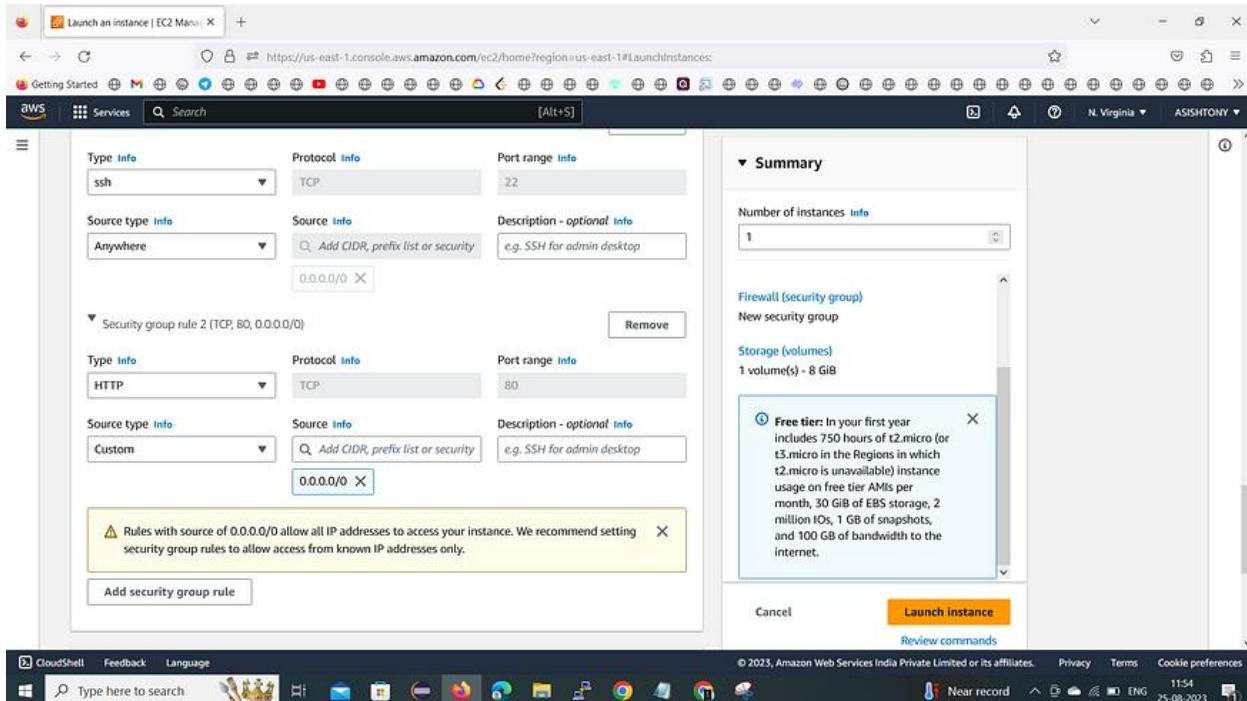
Step 8: Configure Network Settings



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.

▼ Key pair (login) [Info](#)

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

Key pair name - required

Select [Create new key pair](#)

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.

The screenshot shows the AWS EC2 Instances 'Launch an instance' page. At the top, there's a green success banner stating 'Successfully initiated launch of instance (I-05d915ae626952097)'. Below the banner, there's a 'Next Steps' section with a search bar and a navigation menu (1-6). The main area contains four cards: 'Create billing and free tier usage alerts', 'Connect to your instance', 'Connect an RDS database', and 'Create EBS snapshot policy'. Each card has a brief description, a primary action button, and a 'Learn more' link.

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.

Engine options

Engine type [Info](#)

- Aurora (MySQL Compatible) 
- Aurora (PostgreSQL Compatible) 
- MySQL 
- MariaDB 
- PostgreSQL 
- Oracle 
- Microsoft SQL Server 

[▼ Hide filters](#)

Show versions that support the Amazon RDS Optimized Writes [Info](#)
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.

Engine Version

MariaDB 10.6.14 [▼](#)

Templates

Choose a sample template to meet your use case.

- Production
Use defaults for high availability and fast, consistent performance.
- Dev/Test
This instance is intended for development use outside of a production environment.
- Free tier
Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS.
[Info](#)

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.

Settings

DB instance identifier [Info](#)
 Type a name for your DB instance. The name must be unique cross all DB instances owned by your AWS account in the current AWS Region.

tutorial-db-instance

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens (1 to 15 for SQL Server). First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

Credentials Settings

Master username [Info](#)
 Type a login ID for the master user of your DB instance.

tutorial_user

1 to 16 alphanumeric characters. First character must be a letter

Auto generate a password
 Amazon RDS can generate a password for you, or you can specify your own password

Master password [Info](#)
 Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), "(double quote) and @ (at sign).

Confirm password [Info](#)

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

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

Instance configuration

The DB instance configuration options below are limited to those supported by the engine that you selected above.

DB instance class [Info](#)

Standard classes (includes m classes)
 Memory optimized classes (includes r and x classes)
 Burstable classes (includes t classes)

db.t3.micro
 2 vCPUs 1 GiB RAM Network: 2,085 Mbps

Include previous generation classes

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**.

Compute resource
Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.

Don't connect to an EC2 compute resource
Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.

Connect to an EC2 compute resource
Set up a connection to an EC2 compute resource for this database.

EC2 instance Info
Choose the EC2 instance to add as the compute resource for this database. A VPC security group is added to this EC2 instance. A VPC security group is also added to the database with an inbound rule that allows the EC2 instance to access the database.

i-1234567890abcdef0
tutorial-ec2-instance-web-server

Some VPC settings can't be changed when a compute resource is added
Adding an EC2 compute resource automatically selects the VPC, DB subnet group, and public access settings for this database. To allow the EC2 instance to access the database, a VPC security group rds-ec2-X is added to the database and another called ec2-rds-X to the EC2 instance. You can remove the new security group for the database only by removing the compute resource.

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.

RDS > Databases > tutorial-db-instance

tutorial-db-instance

Summary

DB identifier	tutorial-db-instance	CPU	3.10%
Role	Instance	Current activity	0 Connections

Connectivity & security Monitoring Logs & events Configuration Maintenance

Connectivity & security

Endpoint & port

Endpoint: tutorial-db-instance.
west-2.rds.amazonaws.com

Port: 3306

Networking

Availability Zone: us-west-2a

VPC: tutorial-vpc (vpc-04badc20a546242e6)

Subnet group:

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

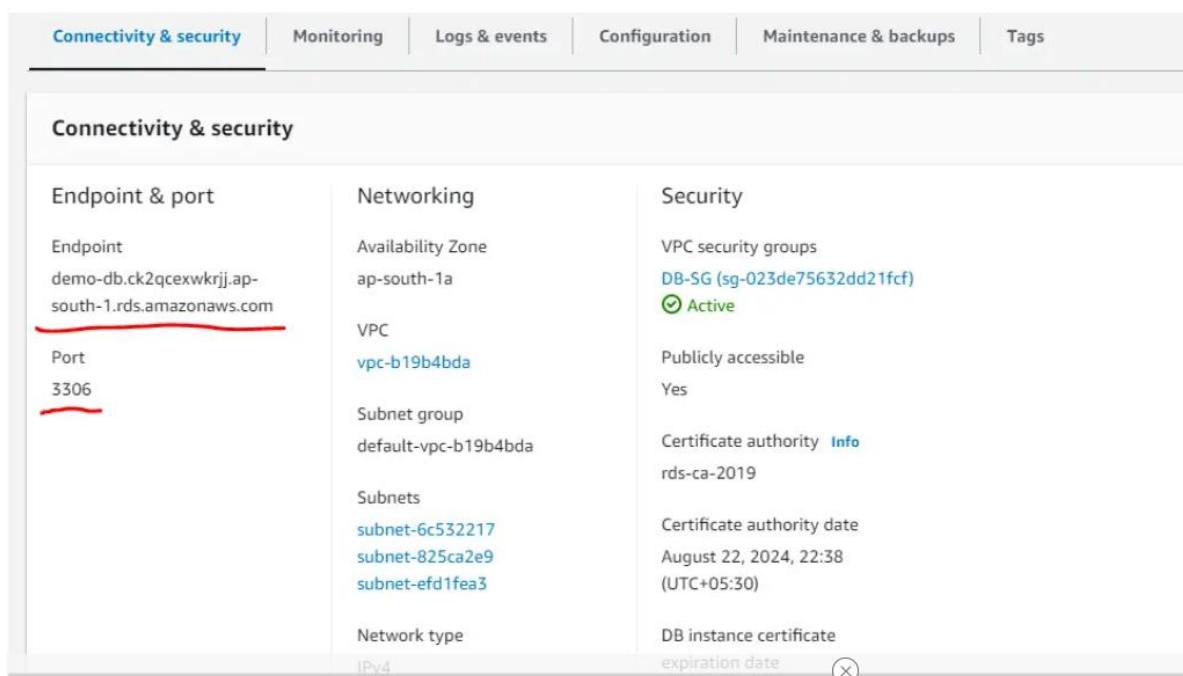
1. Create a MySQL Database on AWS RDS 
2. Install MySQL Workbench
3. Open MySQL Workbench
4. Setup New Connection
5. Verify Connectivity

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.



The screenshot shows the 'Connectivity & security' tab selected in the AWS RDS console. It displays the following information:

Endpoint & port	Networking	Security
Endpoint demo-db.ck2qcexwkrjj.ap-south-1.rds.amazonaws.com	Availability Zone ap-south-1a	VPC security groups DB-SG (sg-023de75632dd21fcf)  Active
Port 3306	VPC vpc-b19b4bda	Publicly accessible Yes
	Subnet group default-vpc-b19b4bda	Certificate authority Info rds-ca-2019
	Subnets subnet-6c532217 subnet-825ca2e9 subnet-efd1fea3	Certificate authority date August 22, 2024, 22:38 (UTC+05:30)
	Network type IPv4	DB instance certificate expiration date 

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.

④ [MySQL Community Downloads](#)

◀ MySQL Workbench

MySQL Workbench 8.0.34

Select Operating System:

Microsoft Windows

Recommended Download:

MySQL Installer for Windows

All MySQL Products. For All Windows Platforms. In One Package.

Starting with MySQL 5.6 the MySQL Installer package replaces the traditional MSI packages.

Windows (x86, 32 & 64-bit), MySQL Installer MSI

[Go to Download Page >](#)

Other Downloads:

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-

MySQL Workbench

Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

Browse Documentation > Read the Blog > Discuss on the Forums >

MySQL Connections [+ ↗](#)

Step 4: Setup New Connection

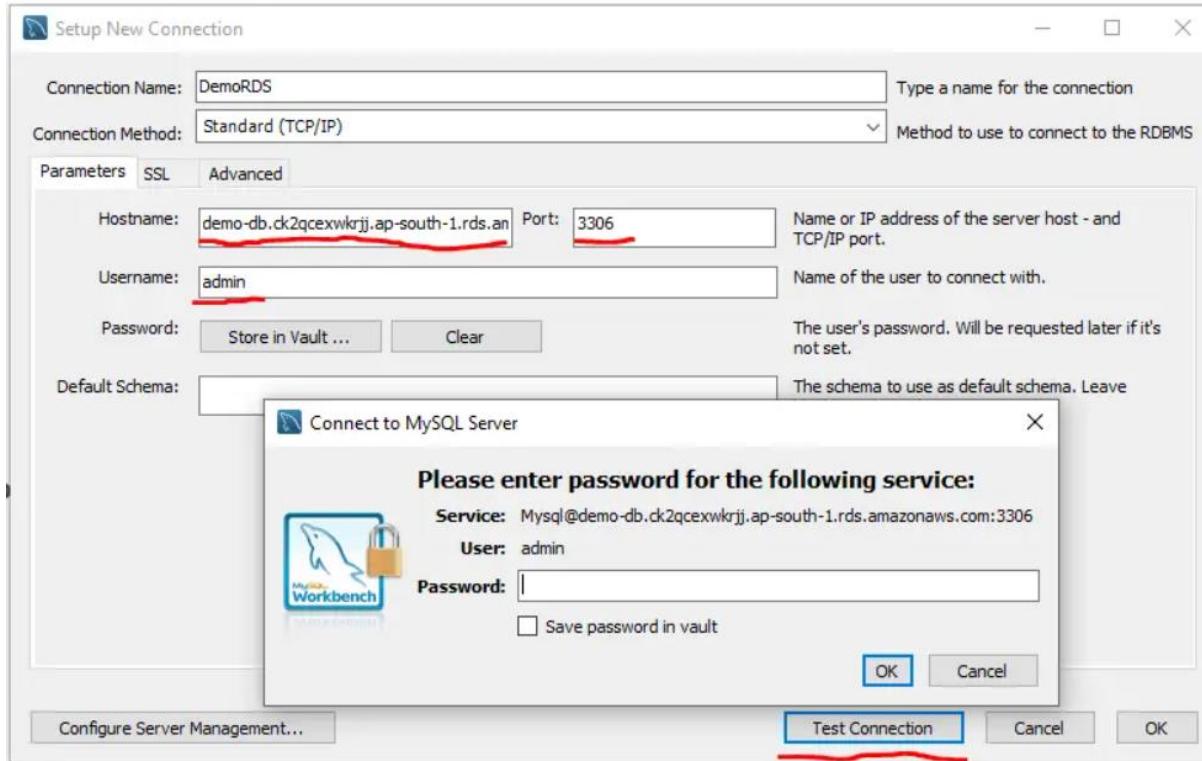
Once you click on the + icon, it will open the Setup new connection dialogue. We need to put up details like-

Hostname: Remember we noted down the RDS endpoint.

Port: We have the Port number as well.

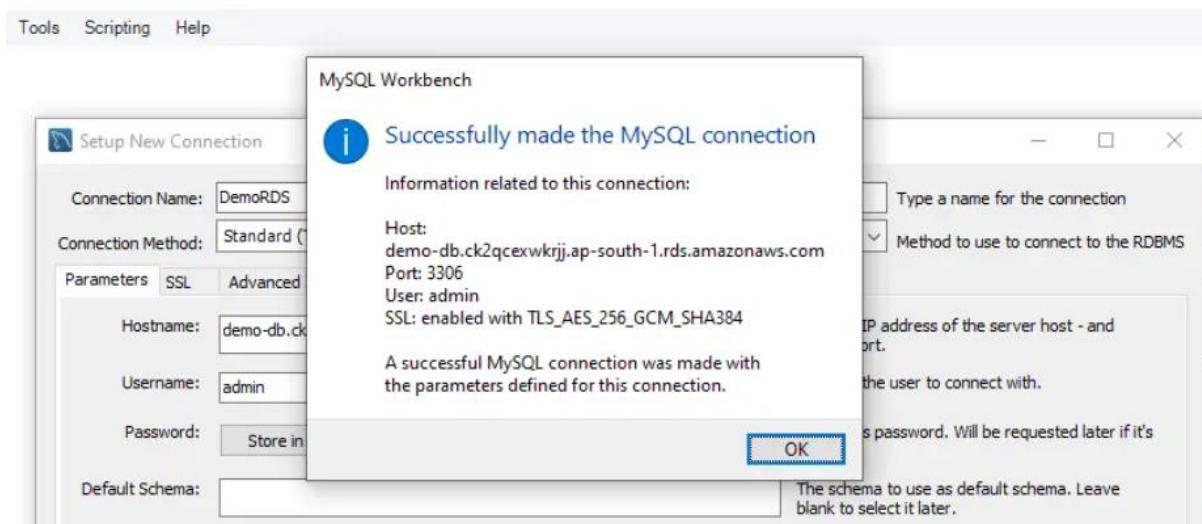
Username: User name that you created while setting up MySQL on RDS

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.

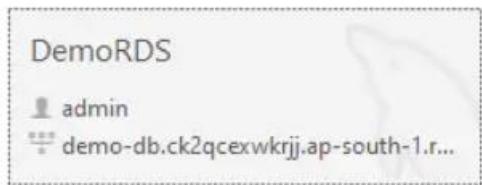
You will see, that you are successfully connected to AWS RDS from your MySQL workbench.



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-

MySQL Connections

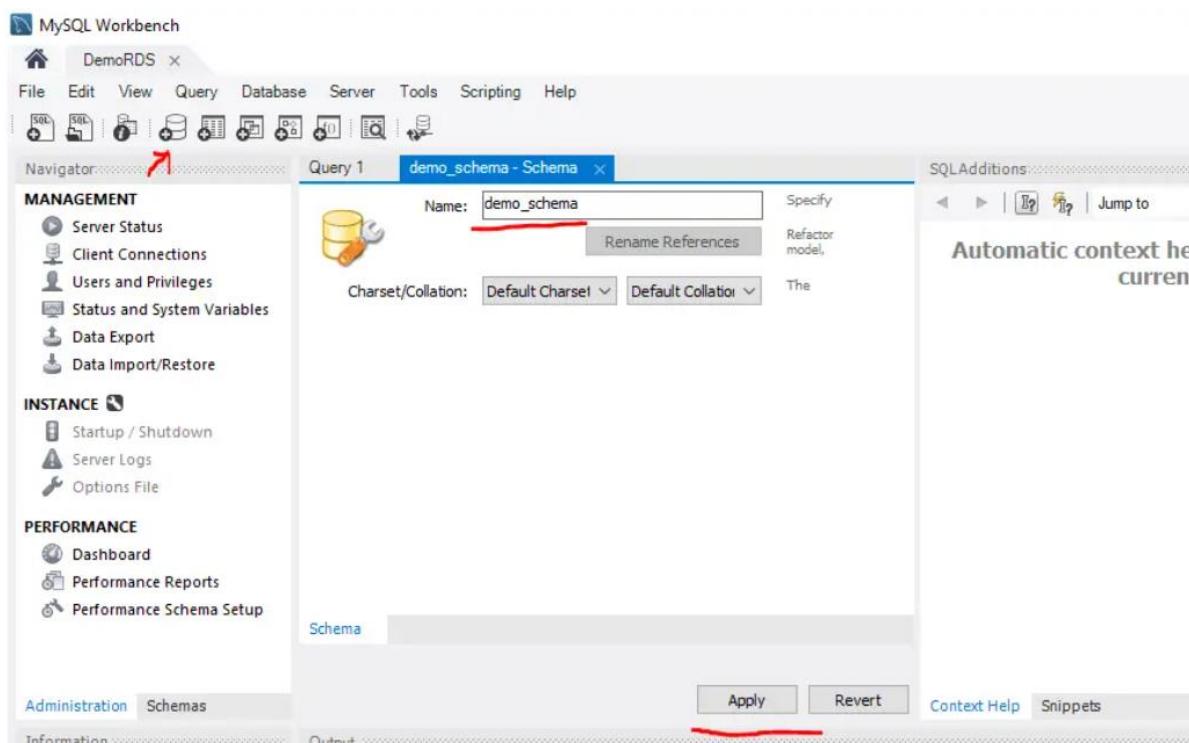


This setting is saved and you can connect to this AWS RDS database instance from your MySQL workbench anytime.

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 screenshot shows the AWS EC2 Dashboard. At the top, it displays resource counts: Instances (running) 3, Dedicated Hosts 0, Instances 3, Key pairs 5, Placement groups 0, Security groups 10, and Volumes 3. Below this, a callout box provides information about Microsoft SQL Server Always On availability groups. The main area features a large orange "Launch instance" button, which is circled in red. To the right, there are sections for "Service health" and "Zones".

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.

The screenshot shows the AWS Cloud9 interface. At the top, there's a search bar with placeholder text "Search our full catalog including 1000s of application and OS images". Below it, there are two tabs: "Recents" and "Quick Start", with "Quick Start" being active. A grid of icons represents different AMIs: Amazon Linux (selected), macOS, Ubuntu, Windows, Red Hat, and others partially visible. To the right of the grid is a search icon and a link "Browse more AMIs" with the note "Including AMIs from AWS, Marketplace and the Community". Below the grid, a section for "Amazon Machine Image (AMI)" displays the selected "Amazon Linux 2023 AMI". This section includes details like "Free tier eligible", the AMI ID "ami-0efa651876de2a5ce", and its configuration: "Virtualization: hvm", "ENA enabled: true", and "Root device type: ebs". Further down, there's a "Description" section with the text "Amazon Linux 2023 AMI 2023.0.20230322.0 x86_64 HVM kernel-6.1", and a table showing "Architecture: 64-bit (x86)", "Boot mode: uefi-preferred", and "AMI ID: ami-0efa651876de2a5ce", with a green "Verified provider" button next to it.

- 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.

Network settings [Info](#) [Edit](#)

Network [Info](#)
vpc-1a2b3c4d

Subnet [Info](#)
No preference (Default subnet in any availability zone)

Auto-assign public IP [Info](#)
Enable

Firewall (security groups) [Info](#)
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group Select existing security group

We'll create a new security group called '**launch-wizard-1**' with the following rules:

<input checked="" type="checkbox"/> Allow SSH traffic from	Helps you connect to your instance	My IP
<input type="checkbox"/> Allow HTTPS traffic from the internet	To set up an endpoint, for example when creating a web server	
<input type="checkbox"/> Allow HTTP traffic from the internet	To set up an endpoint, for example when creating a web server	

- 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.

EC2 > Instances > Launch an instance

Success
Successfully initiated launch of instance **(i-1234567890abcdef0)**

▶ Launch log

Next Steps

Create billing and free tier usage alerts **Connect t**

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**.

Details	Security	Networking	Storage	Status checks	Monitoring	Tags
▼ Instance summary Info						
Instance ID <input type="button" value="i-1234567890abcdef0"/>	Public IPv4 address <input type="button" value="open address"/>	Private IPv4 addresses <input type="button" value=""/>				
IPv6 address -	Instance state <input type="button" value="Pending"/>	Public IPv4 DNS <input type="button" value="ec2-12-345-67-890.compute-1.amazonaws.com"/> <input type="button" value="open address"/>				

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

Instance auto-recovery Default	Lifecycle normal	Stop-hibernate behavior disabled
AMI Launch index 0	Key pair name <input type="button" value="ec2-database-connect-key-pair"/>	State transition reason -
Credit specification standard	Kernel ID -	State transition message -

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

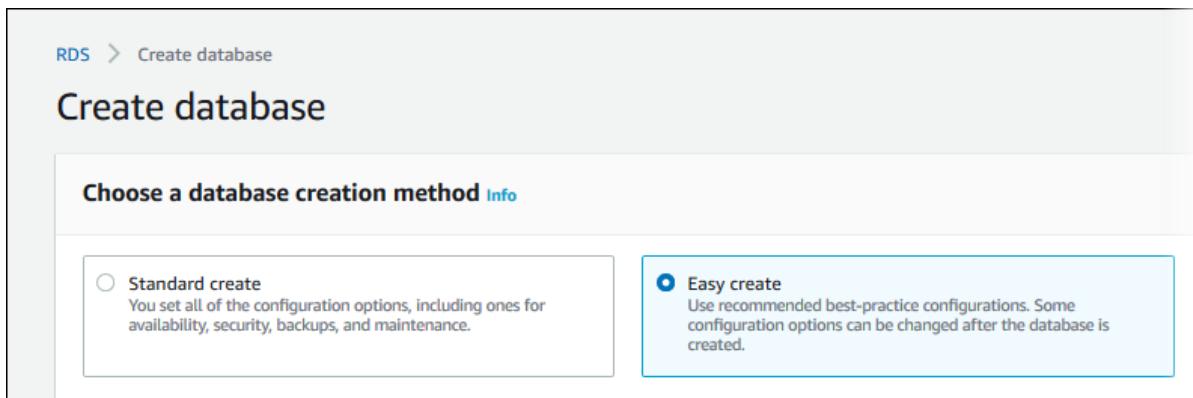
Step 2: Create a MySQL DB instance

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.

Configuration

Engine type [Info](#)

Aurora (MySQL Compatible)



Aurora (PostgreSQL Compatible)



MySQL



MariaDB



PostgreSQL



Oracle

ORACLE®

Microsoft SQL Server



Edition

MySQL Community

DB instance size

Production

db.r6g.xlarge
4 vCPUs
32 GiB RAM
500 GiB

Dev/Test

db.r6g.large
2 vCPUs
16 GiB RAM
100 GiB

Free tier

db.t3.micro
2 vCPUs
1 GiB RAM
20 GiB

DB instance identifier

Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

database-test1

- 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**.

- 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.

▼ Set up EC2 connection - optional

You can also set up a connection to an EC2 instance after creating the database. Go to the database list page or the database details page, choose **Actions**, and then choose **Set up to EC2 connection**.

Compute resource

Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.

 Don't connect to an EC2 compute resource

Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.

 Connect to an EC2 compute resource

Set up a connection to an EC2 compute resource for this database.

EC2 instance [Info](#)

Choose the EC2 instance to add as the compute resource for this database. A VPC security group is added to this EC2 instance. A VPC security group is also added to the database with an inbound rule that allows the EC2 instance to access the database.

i-
i-1234567890abcdef0



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.

Summary			
DB identifier database-test1	CPU -	Status  Creating	Class db.r6g.large
Role Instance	Current activity	Engine MySQL Community	Region & AZ us-east-1c

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.

RDS > Databases > database-test1

database-test1

Summary

DB identifier	CPU
database-test1	<div style="width: 2.58%; background-color: #ff9999; height: 10px;"></div> 2.58%
Role	Current activity
Instance	<div style="width: 0%; background-color: #ff9999; height: 10px;"></div> 0 Connections

- Connectivity & security**
- Monitoring
- Logs & events
- Configuration

Connectivity & security

Endpoint & port <div style="border: 2px solid red; border-radius: 10px; padding: 5px; margin-top: 10px;"> Endpoint database-test1.123456789012.us-east-1.rds.amazonaws.com </div> <div style="border: 2px solid red; border-radius: 10px; padding: 5px; margin-top: 10px;"> Port 3306 </div>	Networking <div style="margin-top: 10px;"> Availability Zone us-east-1c </div> <div style="margin-top: 10px;"> VPC vpc-<div style="background-color: #4f81bd; width: 100px; height: 10px; display: inline-block;"></div> </div> <div style="margin-top: 10px;"> Subnet group default </div>
---	--

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.

Queues (2)		<input type="button" value="Create queue"/>	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>	<input type="button" value="Send and receive messages"/>	<input type="button" value="Actions"/>	<input type="button" value="Create queue"/>
		<input type="text"/> Search queues by prefix				< 1 >	
Name	Type	Created	Messages available	Messages in flight	Encryption	Content-based deduplication	
<input type="radio"/> MyTestQueue	Standard	6/27/2022, 13:03:08 EDT	0	0	Disabled	-	
<input type="radio"/> testFifo1.fifo	FIFO	6/27/2022, 13:03:41 EDT	0	0	Disabled	Disabled	

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.

Queues (2)		<input type="button" value="Create queue"/>	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>	<input type="button" value="Send and receive messages"/>	<input type="button" value="Actions"/>	<input type="button" value="Create queue"/>
		<input type="text"/> Search queues by prefix				< 1 >	
Name	Type	Created	Messages available	Messages in flight	Encryption	Content-based deduplication	
<input checked="" type="radio"/> MyTestQueue	Standard	6/27/2022, 13:03:08 EDT	0	0	Disabled	-	
<input type="radio"/> testFifo1.fifo	FIFO	6/27/2022, 13:03:41 EDT	0	0	Disabled	Disabled	

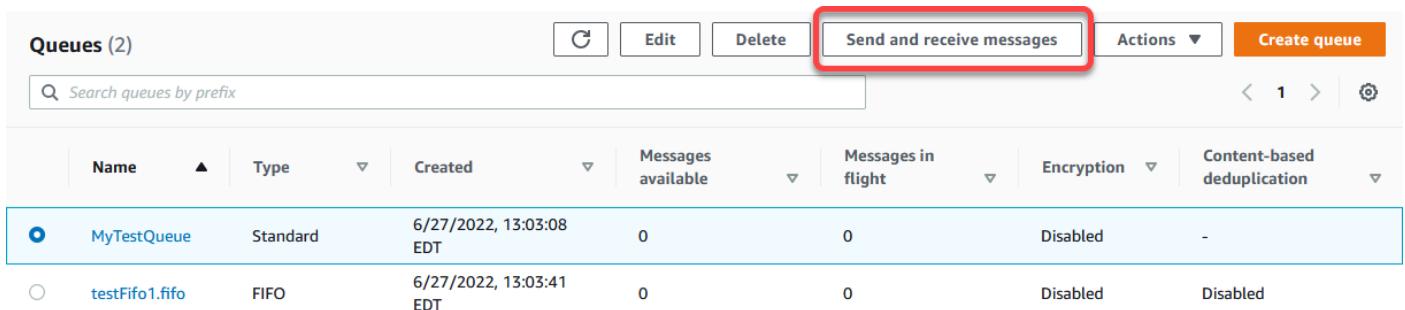
You can also choose the queue name, which opens the **Details** page for the queue. The **Details** page includes the same actions as the **Queues** page. In addition, you can choose one of the tabs below the **Details** section to view additional configuration details and actions.

MyTestQueue		<input type="button" value="Edit"/>	<input type="button" value="Delete"/>	<input type="button" value="Purge"/>	<input type="button" value="Send and receive messages"/>	<input type="button" value="Start DLQ rehive"/>
Details Info						
Name	<input type="text" value="MyTestQueue"/>	Type	Standard	ARN	<input type="text" value="arn:aws:sqs:us-east-1:269704527654:MyTestQueue"/>	
Encryption	Disabled	URL	<input type="text" value="https://sqs.us-east-1.amazonaws.com/269704527654/MyTestQueue"/>	Dead-letter queue	-	
<input type="button" value="More"/> SNS subscriptions Lambda triggers Dead-letter queue Monitoring Tagging Access policy Encryption Dead-letter queue rehive tasks						

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**.



Queues (2)		<input type="button" value="C"/>	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>	<input style="border: 2px solid red; border-radius: 5px; padding: 2px 10px; background-color: white; color: black; font-weight: bold; font-size: 10px; margin-right: 10px;" type="button" value="Send and receive messages"/>	<input type="button" value="Actions ▾"/>	<input type="button" value="Create queue"/>
	Name	Type	Created	Messages available	Messages in flight	Encryption	Content-based deduplication
<input checked="" type="radio"/>	MyTestQueue	Standard	6/27/2022, 13:03:08 EDT	0	0	Disabled	-
<input type="radio"/>	testFifo1 fifo	FIFO	6/27/2022, 13:03:41 EDT	0	0	Disabled	Disabled

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 CLI/ AWS 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.AmazonSQSEException;
```

```
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.AmazonSQSEException;  
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.

The screenshot shows the AWS Management Console search interface. The search term 'dy' has been entered into the search bar. Below the search bar, there is a 'Resource Groups & Tag Editor' button. The main area shows a list of services under 'Services (3)'. The 'DynamoDB' service is highlighted with a yellow star icon and labeled as a 'Managed NoSQL Database'. Other listed services include CloudFront (Global Content Delivery Network) and AWS Cloud Map (Build a dynamic map of your cloud). On the left sidebar, there are sections for 'Recent' (DynamoDB, S3, Support, AWS Lambda), 'Features (12)', 'Blogs (1,521)', 'Documentation (14)', 'Tutorials (7)', and 'Events (23)'.

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

a. In the DynamoDB console, choose Create table.

The screenshot shows the Amazon DynamoDB console homepage. The main heading is 'Amazon DynamoDB: A fast and flexible NoSQL database service for any scale'. Below the heading, it says 'DynamoDB is a fully managed, key-value, and document database that delivers single-digit-millisecond performance at any scale.' To the right, there is a 'Get started' section with a 'Create table' button, which is highlighted with a red box. On the left, there is a navigation sidebar with options like 'Dashboard', 'Tables', 'Update settings', 'Explore items', 'PartiQL editor', 'Backups', 'Exports to S3', 'Reserved capacity', 'DAX', and 'Clusters'.

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

The screenshot shows the 'Create table' wizard in the DynamoDB console. The first step is 'Table details'. The 'Table name' field is filled with 'Music' and is highlighted with a red box. Below the table name, there is a note: 'This will be used to identify your table.' and a character limit: 'Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).'. There is also a 'Partition key' section with a note: 'The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across'.

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.

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.
Music

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.
Artist String

1 to 255 characters and case sensitive.

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

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.
Music

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.
Artist String

1 to 255 characters and case sensitive.

Sort key - optional
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.
songTitle String

1 to 255 characters and case sensitive.

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**.

The screenshot shows the 'Settings' step of the DynamoDB CreateTable wizard. It features two main options: 'Default settings' and 'Customize settings'. The 'Customize settings' option is selected and highlighted with a red box. Below this, the 'Table class' section is shown, with 'DynamoDB Standard' selected. The 'Capacity calculator' section is also visible. Further down, the 'Read/write capacity settings' section is shown, with 'Provisioned' selected for capacity mode. Read capacity settings include auto scaling turned 'On' and target utilization set at 70%. The entire configuration page has a light gray background with white boxes for each section.

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.

The screenshot shows the AWS DynamoDB console. On the left, the navigation pane includes 'Dashboard', 'Tables' (with 'Explore items' highlighted in red), 'PartiQL editor', 'Backups', 'Exports to S3', 'Reserved capacity', and 'Preferences'. Under 'DAX', there are 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main area shows a table named 'Tables (1)' with one item selected, 'Music'. In the 'Scan/Query items' section, a scan is being run for the 'Music' table. The results table shows 'Items returned (0)' and a message: 'The query did not return any results.'

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.

The screenshot shows the 'Item editor' for the 'Music' table. It displays a 'Create item' form with two attributes: 'Artist - Partition key' set to 'No One You Know' and 'songTitle - Sort key' set to 'Call Me Today'. There are 'Form' and 'JSON' tabs at the top right. At the bottom right are 'Cancel' and 'Create item' buttons, with 'Create item' highlighted by a red box.

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**

The item has been saved successfully.

DynamoDB > Items > Music

Tables (1)

Any table tag

Find tables by table name

Music

Music

Autopreview View table details

Scan/Query items

Expand to query or scan items.

Items returned (5)

Actions Create item

Artist	songTitle
The Acme Band	Look Out, World
The Acme Band	Still in Love
No One You Know	Somewhere Down The Road
No One You Know	My Dog Spot
No One You Know	Call Me Today

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".

Scan

Query

Artist (Partition key)

Enter partition key value

songTitle (Sort key)

Equal to Enter sort key val.

Sort descending

Filters

Run Reset

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.

The screenshot shows the AWS DynamoDB console interface. On the left, the navigation bar includes 'Services', 'Search for services, features, blogs, docs, and more' (with a keyboard shortcut '[Option+S]'), and 'Ohio' as the region. Below the navigation bar are links for 'Resource Groups & Tag Editor', 'DynamoDB', 'CloudFormation', and 'QuickSight'. The main left sidebar for 'DynamoDB' has sections for 'Dashboard', 'Tables', 'Update settings', 'Explore items' (which is selected and highlighted in blue), 'PartiQL editor', 'Backups', 'Exports to S3', 'Reserved capacity', and 'Preferences'. Under 'DAX', there are links for 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The central area shows a 'Tables (1)' list with one item, 'Music', which is selected and highlighted with a blue circle. To the right, under the 'Music' table, there is a 'Scan/Query items' section. The 'Artist (Partition key)' field contains 'The Acme Band', and the 'songTitle (Sort key)' dropdown is set to 'Equal to' with the value 'Enter sort key val.' The 'Run' button is highlighted with a red box. Below the query section, a results table titled 'Items returned (2)' shows two rows of data:

	Artist	songTitle
<input type="checkbox"/>	The Acme Band	Look Out, World
<input type="checkbox"/>	The Acme Band	Still in Love

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.

The screenshot shows the AWS DynamoDB console with the 'Music' table selected. In the 'Scan/Query items' section, the 'Artist' partition key is set to 'The Acme Band' and the 'songTitle' sort key is set to 'Still in Love'. The results table shows one item: 'Artist: The Acme Band, songTitle: Still in Love'.

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.

The screenshot shows the AWS DynamoDB console interface. On the left, there's a navigation sidebar with options like Dashboard, Tables, Update settings, Explore items, PartiQL editor, Backups, Exports to S3, Reserved capacity, DAX, Clusters, Subnet groups, Parameter groups, and Events. Below that are links to Tell us what you think, Return to the previous console experience, and Density settings.

The main area shows a 'Tables (2)' list with 'Music' selected. The 'Actions' menu is open, showing options like Edit item, Duplicate item, and Delete items (which is highlighted with a red box). Below that are Scan and Query buttons, and a table or index dropdown set to 'Music'. At the bottom are Run and Reset buttons.

Underneath, a 'Completed' message indicates 0.5 read capacity units consumed. The 'Items returned (5)' section displays a table with columns Artist and SongTitle. The rows are:

	Artist	SongTitle
<input type="checkbox"/>	No One You Know	Call Me Today
<input type="checkbox"/>	No One You Know	My Dog Spot
<input type="checkbox"/>	No One You Know	Somewhere Down The Road
<input checked="" type="checkbox"/>	The Acme Band	Look Out, World
<input type="checkbox"/>	The Acme Band	Still in Love

The row for 'The Acme Band' is highlighted with a red box.

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.

The screenshot shows the Amazon S3 console interface. On the left, there is a sidebar with various navigation options like Buckets, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, and Access analyzer for S3. The main area is titled 'Amazon S3' and shows an 'Account snapshot' with a 'View Storage Lens dashboard' button. Below that is a section for 'Buckets (0) Info' with a 'Create bucket' button highlighted with a red box. There is also a search bar labeled 'Find buckets by name' and a table header for 'Name', 'AWS Region', 'Access', and 'Creation date'. A message below the table says 'No buckets' and 'You don't have any buckets.' with a 'Create bucket' button.

- 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

The screenshot shows the 'Create bucket' configuration page. At the top, there is a 'Bucket name' field containing 'mybucket1' and an 'Info' link. Below it is a note about naming rules and a 'Choose bucket' button. Under 'Object Ownership', there are two options: 'ACLs disabled (recommended)' and 'ACLs enabled'. The 'ACLs enabled' option is selected and has a note: 'Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.' At the bottom, there is a warning message: '⚠ We recommend disabling ACLs, unless you need to control access for each object individually or to have the object writer own the data they upload. Using a bucket policy instead of ACLs to share data with users outside of your account simplifies permissions management and auditing.'

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).

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on **Block all public access**. These settings apply only to this bucket and its access points. AWS recommends that you turn on **Block all public access**, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- Block public access to buckets and objects granted through new access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- Block public access to buckets and objects granted through any access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.
- Block public access to buckets and objects granted through new public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- Block public and cross-account access to buckets and objects through any public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Turning off block all public access might result in this bucket and the objects within becoming public

AWS recommends that you turn on **Block all public access**, unless public access is required for specific and verified use cases such as static website hosting.

I acknowledge that the current settings might result in this bucket and the objects within becoming public.

Feedback English (US) ▾

© 2022, Amazon Internet Services Private Ltd. or its affil

Tags(0) : Optional

Default encryption: Disable

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning

Disable
 Enable

Tags (0) - optional

Track storage cost or other criteria by tagging your bucket. [Learn more](#)

No tags associated with this bucket.

Add tag

Default encryption

Automatically encrypt new objects stored in this bucket. [Learn more](#)

Server-side encryption

Disable
 Enable

Now, click on Create Bucket

Advanced settings

After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings.

Create bucket

Step 3: Now upload code files

Select Bucket and Click your Bucket Name.

The screenshot shows the AWS S3 console. At the top, there's an 'Account snapshot' section with a 'View Storage Lens dashboard' button. Below it is a 'Buckets (1) Info' section. A single bucket, 'oriyansh-blog-s3', is listed. The bucket details show: Name: 'oriyansh-blog-s3', AWS Region: 'Asia Pacific (Mumbai) ap-south-1', Access: 'Bucket and objects not public', and Creation date: 'March 1, 2022, 00:51:08 (UTC+05:30)'. There are buttons for 'Copy ARN', 'Empty', 'Delete', and 'Create bucket'.

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

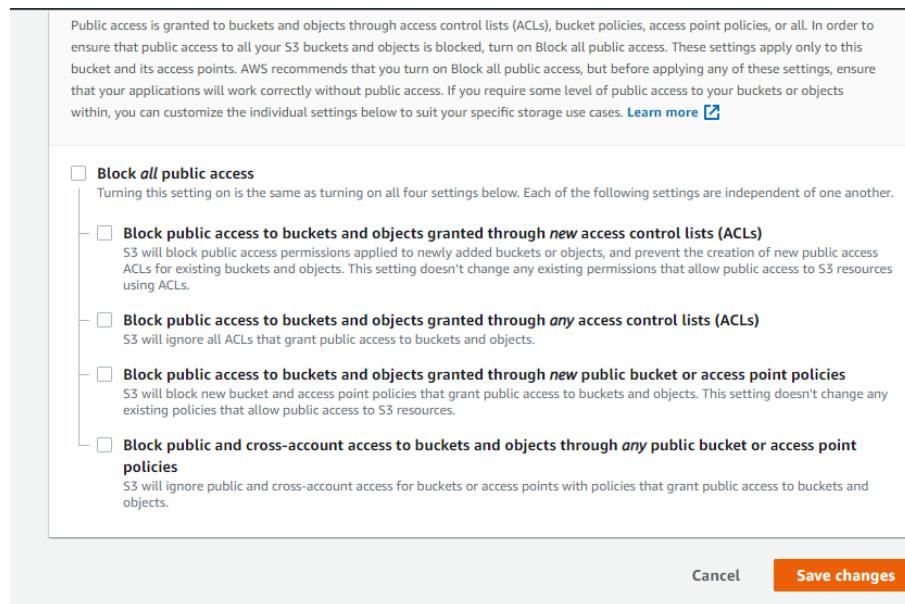
The screenshot shows the 'Upload' dialog box. It has a central area for 'Files and folders' (2 Total, 379.8 KB) with a list containing 'front-logo.png' (image/png, 375.6 KB) and 'index.html' (text/html, 4.1 KB). Both files are highlighted with a red box. Below this is a 'Destination' section with 's3://oriyansh-blog-s3' selected. Under 'Permissions', it says 'Grant public access and access to other AWS accounts.' Under 'Properties', it says 'Specify storage class, encryption settings, tags, and more.' At the bottom right are 'Cancel' and 'Upload' buttons, with 'Upload' highlighted with a red box.

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.

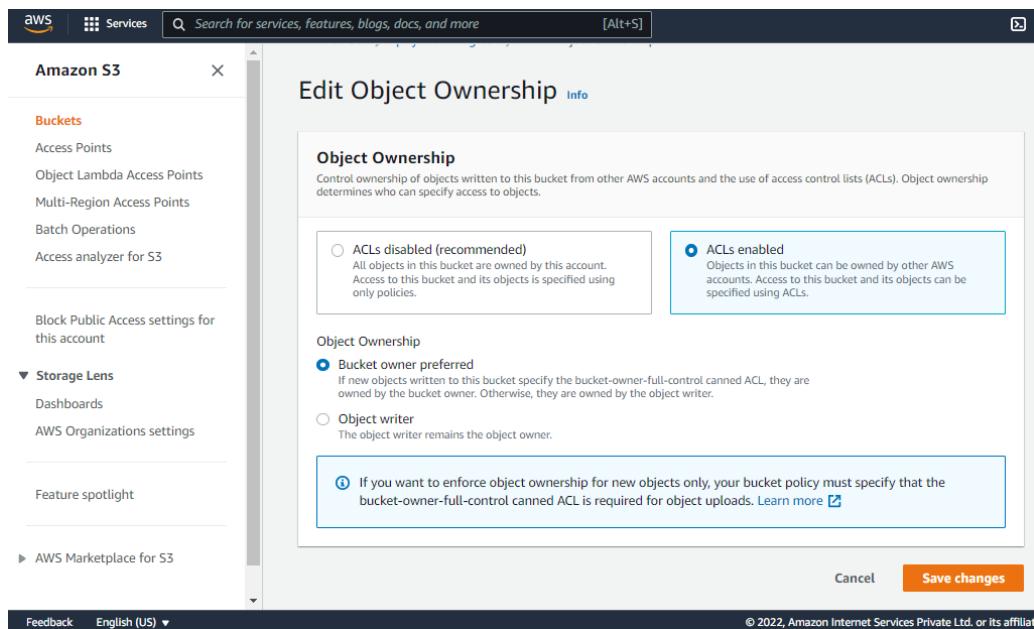
- Save changes then type “confirm”.



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

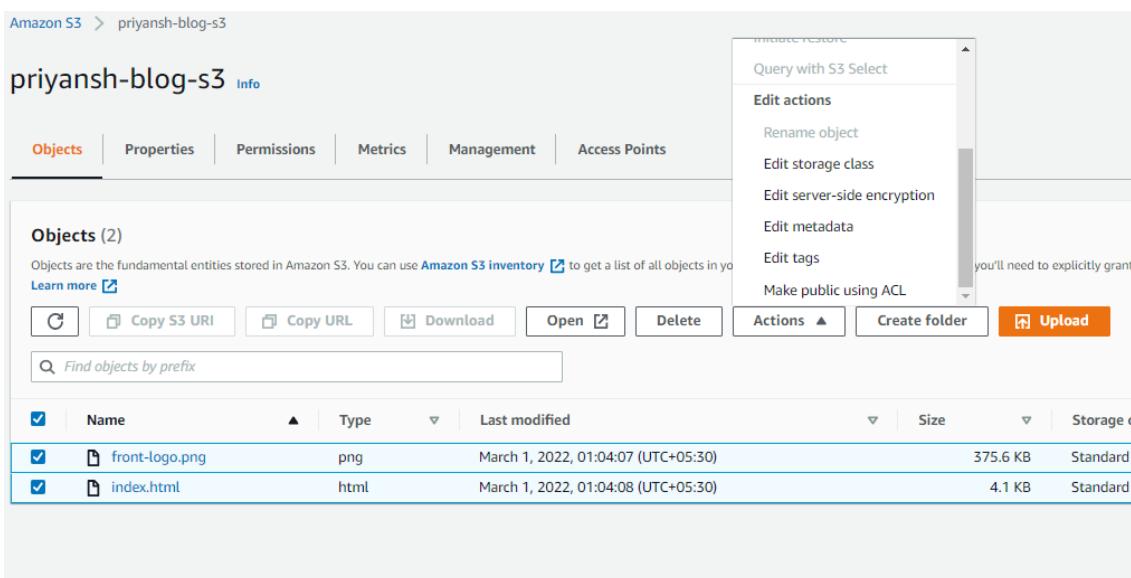
->Check I acknowledge restored.

Choose Save Changes.



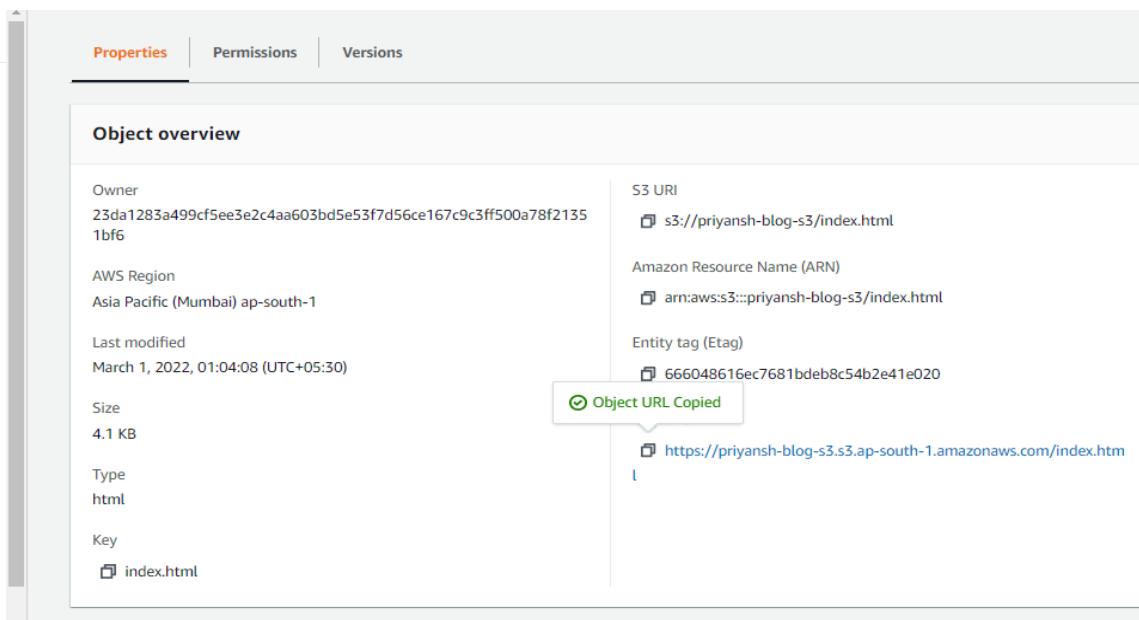
Step 5:- Make public Object

- 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>
```

```

43. <artifactId>junit</artifactId>
44. <version>3.8.1</version>
45. <scope>test</scope>
46. </dependency>
47. <dependency>
48.   <groupId>com.amazonaws</groupId>
49.   <artifactId>aws-java-sdk</artifactId>
50.   <version>1.11.330</version>
51. </dependency>
52. </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 groupId 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.amazonaws 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());
}
```