# M.Sc C.S - II
# SEM IV
# Journal

| | |
|---|---|
| Roll No. | 075 |
| Name | Vikas Laljibhai Rathod |
| Subject | Cloud Computing |
| Subject Code | TPGCS P4021 |

THAKUR TRUSTS

Thakur Educational Trust's (Regd.)

THAKUR COLLEGE OF SCIENCE & COMMERCE

AUTONOMOUS COLLEGE, PERMANENTLY AFFILIATED TO UNIVERSITY OF MUMBAI

NAAC Accredited Grade 'A' (3rd Cycle) & ISO 9001: 2015 (Certified)

Best College Award by University of Mumbai for the Year 2018-2019

75 Azadi Ka Amrit Mahotsav

tcsc

CELEBRATING 25 YEARS OF GLORY

# CERTIFICATE

This is here to certify that Mr.Vikas Laljibhai Rathod Seat Number 075 of M.Sc. I Computer Science, has satisfactorily completed the required number of experiments prescribed by the THAKUR COLLEGE OF SCIENCE & COMMERCE AUTONOMOUS COLLEGE, PERMANENTLY AFFILIATED TO UNIVERSITY OF MUMBAI during the academic year 2022 – 2023.

Date:
Place: Mumbai

Teacher In-Charge Head of Department

External Examiner

# INDEX

# Practical No.: 1

**Aim:** Develop a private cloud using any suitable technology.

**Theory:**

**What is a Private Cloud?**

A **Private Cloud** is a model of cloud computing where the infrastructure is dedicated to a single user organization. A private cloud can be hosted either at an organization's own data centre, at a third party colocation facility, or via a private cloud provider who offers private cloud hosting services and may or may not also offer traditional public shared multi-tenant cloud infrastructure.

Typically, the end-user organization is responsible for the operation of a private cloud as if it were a traditional on-premises infrastructure, which includes ongoing maintenance, upgrades, OS patches, middleware, and application software management.

Private Cloud Solutions offer organizations more control over and better security of private cloud servers, although it does require a much higher level of IT expertise than utilizing a public cloud.

**Why use Private Clouds?**

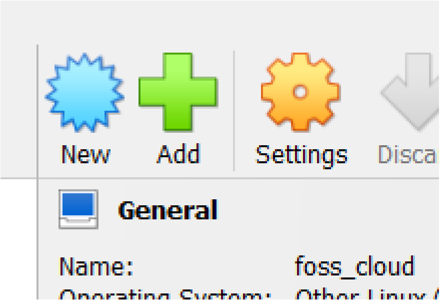
Private Clouds offer the same control and security as traditional on-premises infrastructure. Here are some reasons why organizations opt for private cloud computing:

- **Security**: Private cloud security is enhanced since traffic to a private cloud is typically limited to the organization's own transactions. Public cloud providers must handle traffic from millions of users and transactions simultaneously, thus opening a greater chance for malicious traffic. Since private clouds consist of dedicated physical infrastructure, the organization has better control over the server, network, and application security.
- **Predictable performance**: Because the hardware is dedicated rather than multi-tenant, workload performance is predictable and unaffected by other organizations sharing infrastructure or bandwidth.
- **Long-term savings**: While it can be expensive to set up the infrastructure to support a private cloud, it can pay off in the long term. If an organization already has the hardware and network required for hosting, a private cloud can be much more cost-effective over time compared to paying monthly fees to use someone else's servers on the public cloud.
- **Predictable costs**: Public cloud costs can be very unpredictable based on usage, storage charges, and data egress charges. Private cloud costs are the same each month, regardless of the workloads, an organization is running or how much data is moved.
- **Regulatory governance**: Regulations such as the EU's GDPR may dictate where data resides and where computing occurs. In those regions where public cloud providers cannot offer service, a private cloud may be required. Additionally, organizations with sensitive data such as financial or legal firms may opt for private cloud storage to

ensure they have complete control over personally identifiable or sensitive information.

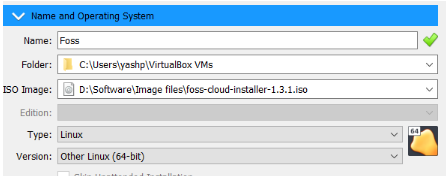
What is the difference between Private Cloud and Public Cloud?

In a private cloud, computing resources are dedicated and proprietary, and a single organization host and manages the system. What makes it private is the fact that the underlying hardware layer is segregated from any other client's infrastructure. In a public cloud, services are owned and managed by a provider who also hosts other tenants. Companies may combine a private cloud with a public cloud
in a hybrid or multi-cloud



Name: foss_cloud

Operating System: Other Linux (



Name: Foss

Folder: C:\Users\yashp\VirtualBox VMs

ISO Image: D:\Software\Image files\foss-cloud-installer-1.3.1.iso

Edition:

Type: Linux

Version: Other Linux (64-bit)

**FOSS cloud**

1. Install Oracle Virtual Box.
2. Click Add to create a new virtual machine.
3. Fill in the details.
4. Select base memory of 4GB.
5. Select 130GB of Hard Disk space.
6. Before opening Foss cloud open setting for the virtual machine just created.
7. Under System -> Processor enable "Nested Virtualization".
   If the option is greyed out then go "C:\Program Files\Oracle\VirtualBox" and open a command prompt and type the following command.

   **"vboxmanage modifyvm "vm_name" --nested-hw-virt on"**

   Where vm_name is the name of virtual machine just created **.**
8. Start the VM.
9. Follow through and install the as a demon system.
10. Once finished you have a private cloud to anything with.

**Conclusion:**

Developed a private cloud using any suitable technology.

# Practical No.: 2

**Aim:** Develop a public cloud using any suitable technology.

**Theory:**

**What is a Public Cloud?**

Public cloud is a type of computing where resources are offered by a third-party provider via the internet and shared by organizations and individuals who want to use or purchase them. Some public cloud computing resources are available for free, while customers may pay for other resources through subscription or pay-per-usage pricing models.

From artificial intelligence services and developer tools to the storage and computing capacity for virtually any workload, public cloud helps companies to harness cutting-edge technologies and achieve global scale without shouldering the costs and labour themselves.

Public clouds contrast with private cloud models, where the resources are available only to a single organization and the data centre is managed either on-premises or off-site by a vendor. For organizations looking for an alternative to traditional on-premises IT architectures or other types of cloud computing, the public cloud offers nearly infinite scalability and self-service provisioning to meet workload and user demands.

Public cloud defined

A public cloud is an IT model where public cloud service providers make computing services—including compute and storage, develop-and-deploy environments, and applications—available on-demand to organizations and individuals over the public internet.

Private cloud vs. public cloud

The main difference between public cloud and private cloud is where it is hosted and who is responsible for managing it. Public cloud uses shared infrastructure, while private clouds use your organization's own dedicated infrastructure. Other common cloud models include hybrid cloud, which combines public and private clouds, and multicloud, where customers employ cloud services spanning multiple public clouds.

Public cloud platforms, such as Google Cloud, pool resources in distributed data centers around the world that multiple companies and users can access from the internet. Rather than an in-house team, the public cloud providers are responsible for managing and maintaining the underlying infrastructure. As a result, leveraging public cloud services reduces IT operational costs and frees up time for teams to focus on valuable work that directly benefits the business.

**A common public cloud example is to think of it as similar to renting an apartment:**

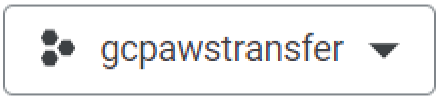
- You pay rent for a single unit
- The building manager handles the maintenance
- You share the overall space with other tenants, with security around your own belongings

Private cloud is more like owning a house, where you have your own personal space that belongs to you, but you're also personally responsible for the overall care and upkeep. However, there are cases where a business may choose a private cloud to fulfil certain requirements, such as industry or region-specific compliance and data sovereignty needs.

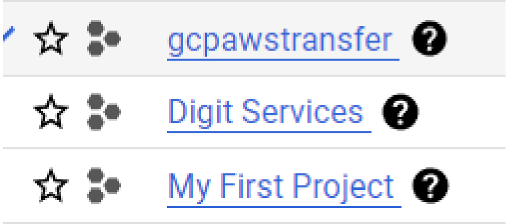
The same goes for a house you own: it can only be extended through renovation. Private cloud requires purchasing hardware to meet demand, as well as any licensing costs needed for software applications. Similarly, an on-premises IT stack can't easily accommodate surges in traffic without staff having to purchase and install new resources (which might go unused until the next surge or decay into technical debt).

Public cloud lets you automatically scale up your compute and storage resources along with the increased security and services you need.

**Google Cloud Console**



1. Head to https://console.cloud.google.com/
2. Login in with google account
3. Create a New project.
4. Decide the name of the project and follow the step to create a new project OR select one of the existing projects.



5. You now have a workspace to use.

**Conclusion:** Developed a public cloud using any suitable technology.

# Practical No.: 4

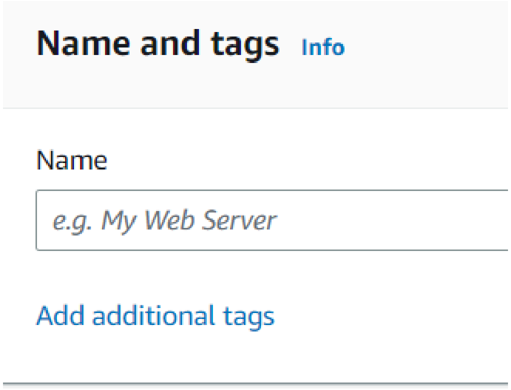
**Aim:** Explore Working of the following with Virtual Machines

- VM Lifecycle
- Creating VMs
- Accessing VMs
- Assigning VMs to Hosts

**Theory:**

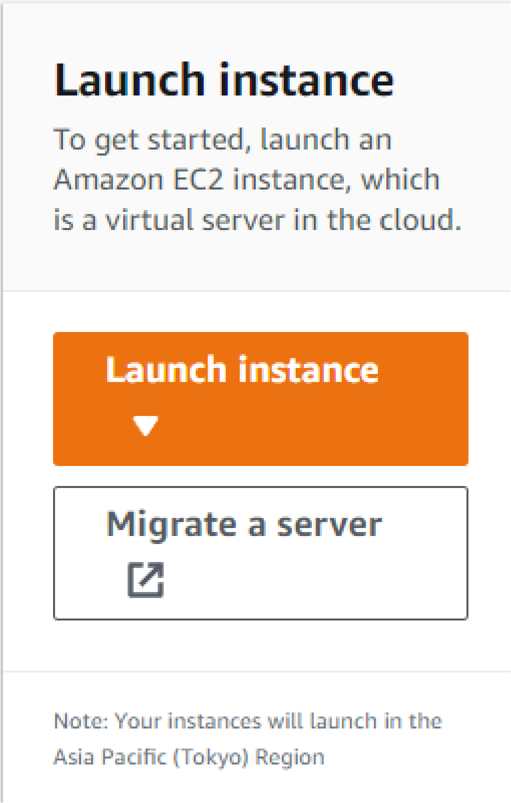
**Creating VM**

1. Log into Cloud console.
2. Open EC2 instances.
3. Launch instances.
4. Pick a name



Name: e.g. My Web Server

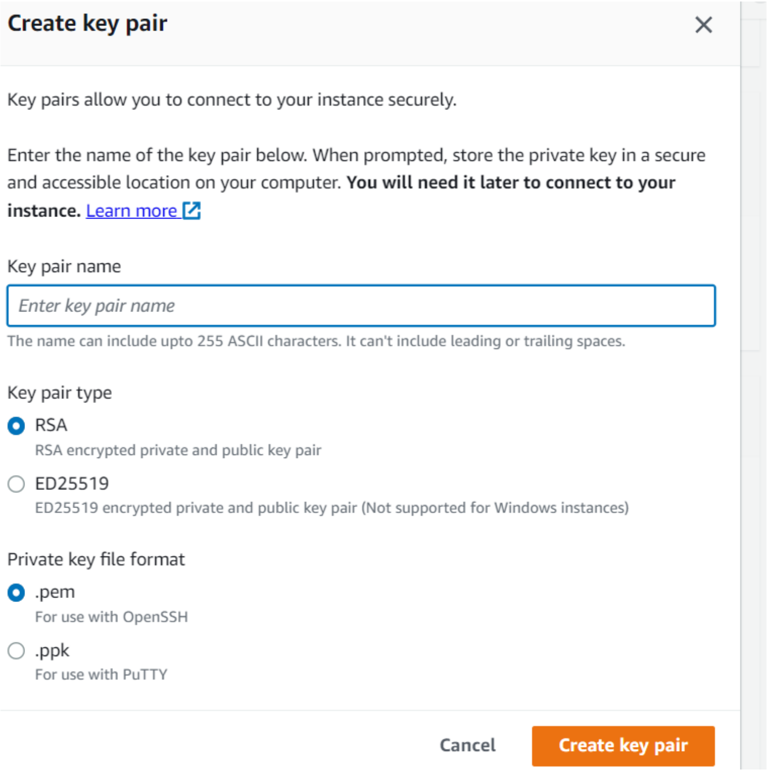
5. Select an Image.
6. Select storage as needed.
7. Create a new Key pair.
8. Store the key file in a secured location.
9. Click launch instance.



Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Note: Your instances will launch in the Asia Pacific (Tokyo) Region



**Create key pair** ✕

Key pairs allow you to connect to your instance securely.

Enter the name of the key pair below. When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** Learn more

Key pair name

*Enter key pair name*

The name can include upto 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

● RSA
RSA encrypted private and public key pair

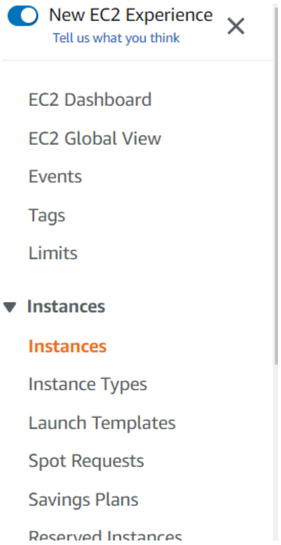
○ ED25519
ED25519 encrypted private and public key pair (Not supported for Windows instances)

Private key file format

● .pem
For use with OpenSSH

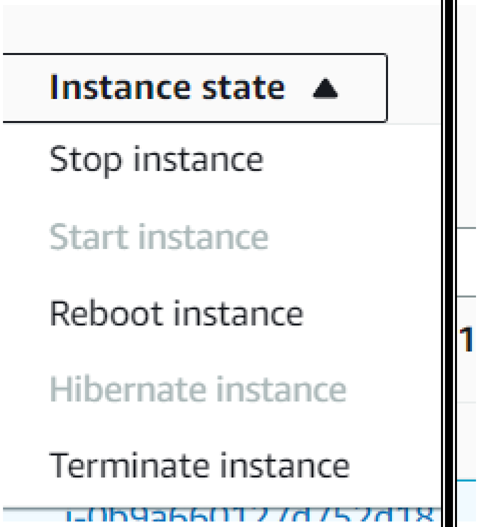
○ .ppk
For use with PuTTY

Cancel | Create key pair



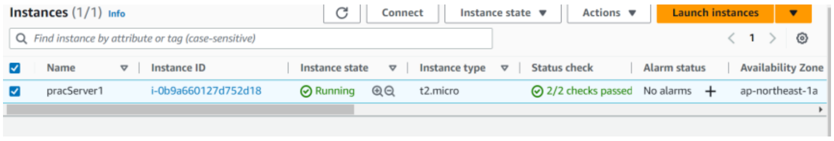
## VM states

1. Go to instances.
2. Select your instance and then pick instance state



## Accessing Vm

1. From Ec2 menu open instance

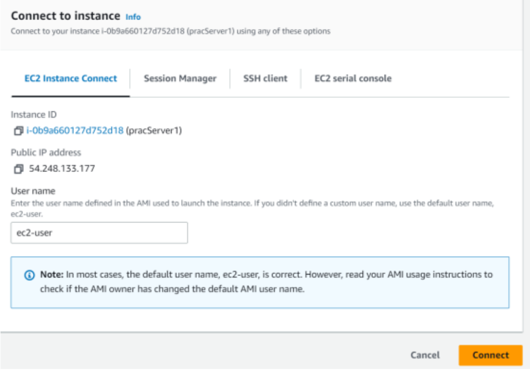


**Instances** (1/1) Info | Connect | Instance state ▼ | Actions ▼ | Launch instances ▼

*Find instance by attribute or tag (case-sensitive)*

| ☑ | Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone |
|---|---|---|---|---|---|---|---|
| ☑ | pracServer1 | i-0b9a660127d752d18 | ✓ Running | t2.micro | ✓ 2/2 checks passed | No alarms + | ap-northeast-1a |

2. Click connect



**Connect to instance** Info

Connect to your instance i-0b9a660127d752d18 (pracServer1) using any of these options

EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

Instance ID
i-0b9a660127d752d18 (pracServer1)

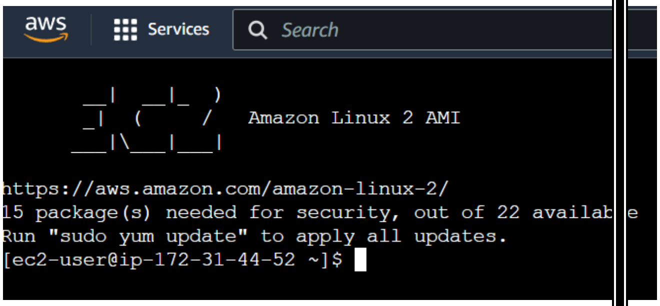
Public IP address
54.248.133.177

User name
Enter the user name defined in the AMI used to launch the instance. If you didn't define a custom user name, use the default user name, ec2-user.

ec2-user

**Note:** In most cases, the default user name, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Cancel | Connect



```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
15 package(s) needed for security, out of 22 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-44-52 ~]$
```

3. You will have you console in the tab opened.

**Conclusion:** Explored Working of the following with Virtual Machines

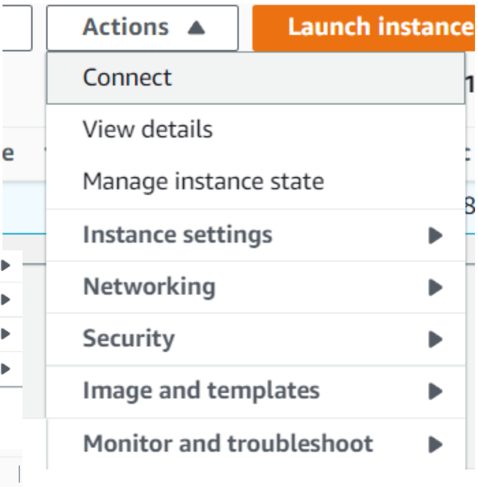
# Practical No.: 5

**Aim:** Explore Working of the following with Virtual Machines

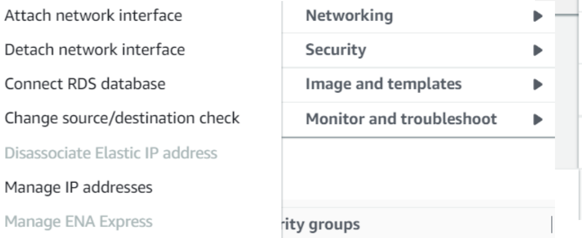
- Changing the Service Offering for a VM
- Using SSH Keys for Authentication

**Theory:**

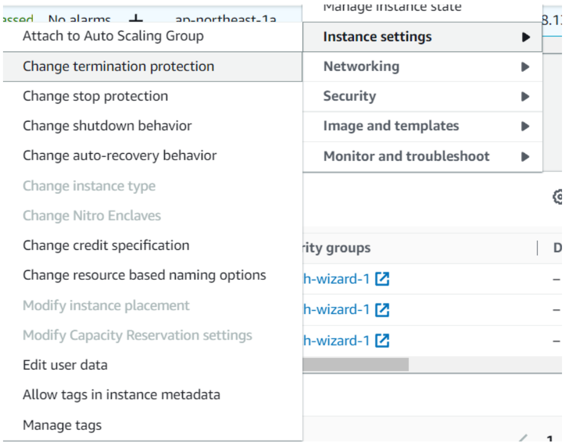
1. Open Instances in EC2
2. Select the instance you want to change



**Network settings**



**Instance settings**


Manage instance state
Attach to Auto Scaling Group
Change termination protection
Change stop protection
Change shutdown behavior
Change auto-recovery behavior
Change instance type
Change Nitro Enclaves
Change credit specification
Change resource based naming options
Modify instance placement
Modify Capacity Reservation settings
Edit user data
Allow tags in instance metadata
Manage tags
Instance settings
Networking
Security
Image and templates
Monitor and troubleshoot

**Security settings**

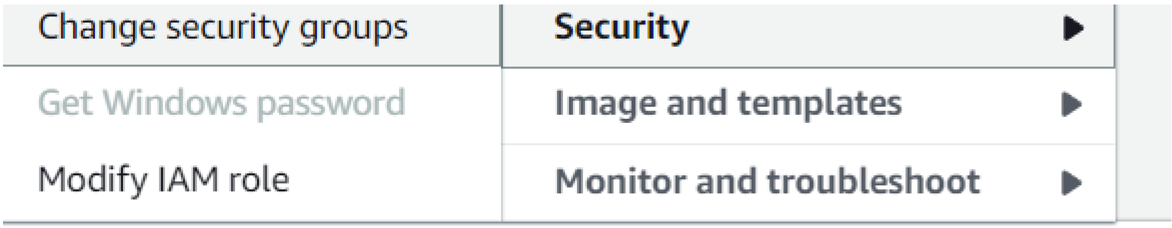
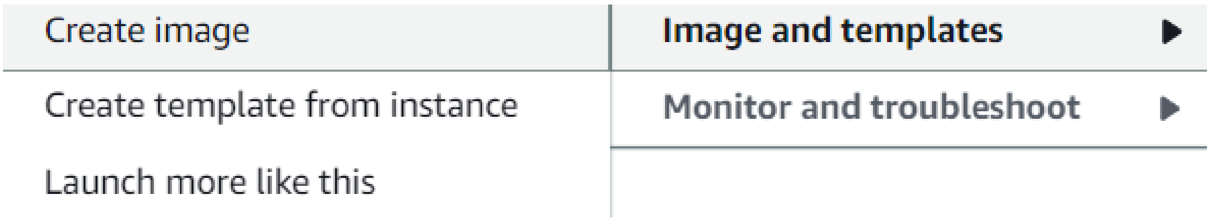
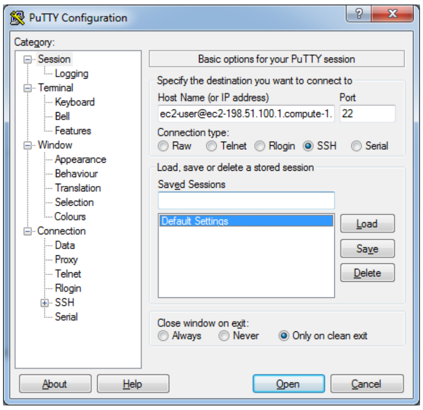


**Image and templets**



## SSH

1. Start PuTTY (from the **Start** menu, choose **All Programs, PuTTY, PuTTY**).
2. In the **Category** pane, choose **Session** and complete the following fields:
3. In the **Host Name** box, do one of the following:
4. (Public DNS) To connect using your instance's public DNS name, enter instance-user-name@instance-public-dns-name.
5. (IPv6) Alternatively, if your instance has an IPv6 address, to connect using your instance's IPv6 address, enter instance-user-name@instance-IPv6-address.
6. For information about how to get the user name for your instance, and the public DNS name or IPv6 address of your instance, see Get information about your instance.
7. Ensure that the **Port** value is 22.
8. Under **Connection type**, select **SSH**.
9. (Optional) You can configure PuTTY to automatically send 'keepalive' data at regular intervals to keep the session active. This is useful to avoid disconnecting from your instance due to session inactivity. In the **Category** pane, choose **Connection**, and then enter the required interval in the **Seconds between keepalives** field. For example, if your session disconnects after 10 minutes of inactivity, enter 180 to configure PuTTY to send keepalive data every 3 minutes.



Host Name (or IP address): ec2-user@ec2-198.51.100.1.compute-1.
Port: 22
Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial
Close window on exit: ○ Always ○ Never ● Only on clean exit

10. In the **Category** pane, expand **Connection**, expand **SSH**, and then choose **Auth**. Complete the following:
11. Choose **Browse**.
12. Select the .ppk file that you generated for your key pair and choose **Open**.
13. (Optional) If you plan to start this session again later, you can save the session information for future use. Under **Category**, choose **Session**, enter a name for the session in **Saved Sessions**, and then choose **Save**.
14. Choose **Open**.
15. If this is the first time you have connected to this instance, PuTTY displays a security alert dialog box that asks whether you trust the host to which you are connecting.
16. (Optional) Verify that the fingerprint in the security alert dialog box matches the fingerprint that you previously obtained in (Optional) Get the instance fingerprint. If these fingerprints don't match, someone might be attempting a "man-in-the-middle" attack. If they match, continue to the next step.
17. Choose **Yes**. A window opens and you are connected to your instance.

## Conclusion:

Explored Working of Virtual Machines

# Practical No.: 6

**Aim:** Explore the working of the following: Storage Overview

- Primary Storage
- Secondary Storage

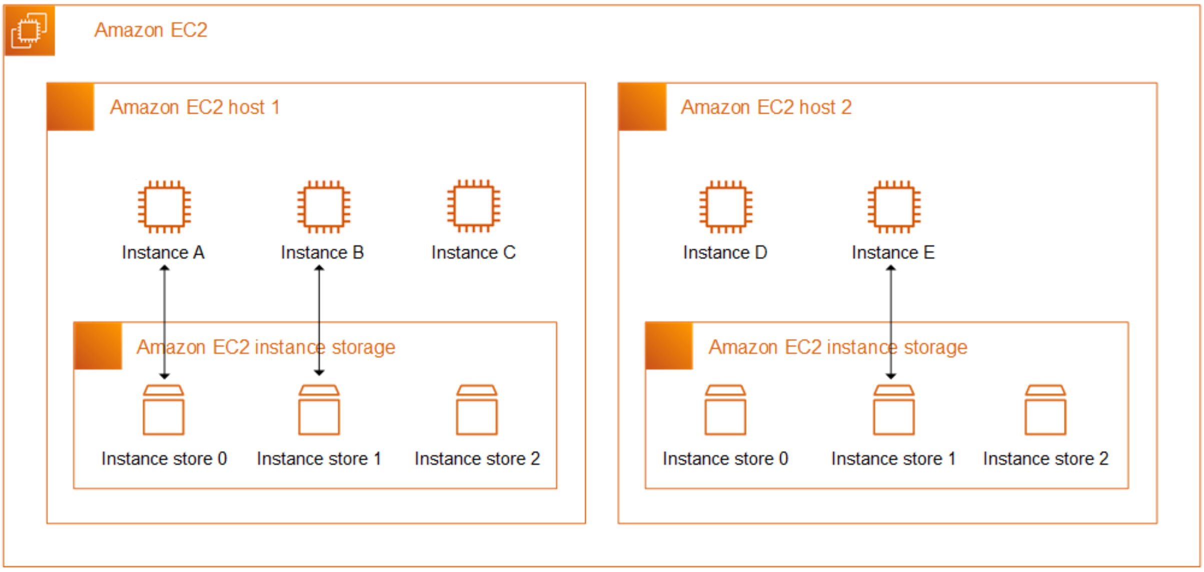
**Theory:**

**Amazon EC2 instance store**

An instance store provides temporary block-level storage for your instance. This storage is located on disks that are physically attached to the host computer. Instance store is ideal for temporary storage of information that changes frequently, such as buffers, caches, scratch data, and other temporary content, or for data that is replicated across a fleet of instances, such as a load-balanced pool of web servers.

An instance store consists of one or more instance store volumes exposed as block devices. The size of an instance store as well as the number of devices available varies by instance type.

The virtual devices for instance store volumes are ephemeral[0-23]. Instance types that support one instance store volume have ephemeral0. Instance types that support two instance store volumes have ephemeral0 and ephemeral1, and so on.



**Instance store lifetime**

You can specify instance store volumes for an instance only when you launch it. You can't detach an instance store volume from one instance and attach it to a different instance.

The data in an instance store persists only during the lifetime of its associated instance. If an instance reboots (intentionally or unintentionally), data in the instance store persists. However, data in the instance store is lost under any of the following circumstances:

- The underlying disk drive fails
- The instance stops
- The instance hibernates
- The instance terminates

Therefore, do not rely on instance store for valuable, long-term data. Instead, use more durable data storage, such as Amazon S3, Amazon EBS, or Amazon EFS.

When you stop, hibernate, or terminate an instance, every block of storage in the instance store is reset. Therefore, your data cannot be accessed through the instance store of another instance.

If you create an AMI from an instance, the data on its instance store volumes isn't preserved and isn't present on the instance store volumes of the instances that you launch from the AMI.

If you change the instance type, an instance store will not be attached to the new instance type.

**Amazon Elastic Block Store (Amazon EBS)**

Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with EC2 instances. EBS volumes behave like raw, unformatted block devices. You can mount these volumes as devices on your instances. EBS volumes that are attached to an instance are exposed as storage volumes that persist independently from the life of the instance. You can create a file system on top of these volumes, or use them in any way you would use a block device (such as a hard drive). You can dynamically change the configuration of a volume attached to an instance.

We recommend Amazon EBS for data that must be quickly accessible and requires long-term persistence. EBS volumes are particularly well-suited for use as the primary storage for file systems, databases, or for any applications that require fine granular updates and access to raw, unformatted, block-level storage. Amazon EBS is well suited to both database-style applications that rely on random reads and writes, and to throughput-intensive applications that perform long, continuous reads and writes.

With Amazon EBS, you pay only for what you use.

## Features of Amazon EBS

- You create an EBS volume in a specific Availability Zone, and then attach it to an instance in that same Availability Zone. To make a volume available outside of the Availability Zone, you can create a snapshot and restore that snapshot to a new volume anywhere in that Region. You can copy snapshots to other Regions and then restore them to new volumes there, making it easier to leverage multiple AWS Regions for geographical expansion, data center migration, and disaster recovery.
- Amazon EBS provides the following volume types: General Purpose SSD, Provisioned IOPS SSD, Throughput Optimized HDD, and Cold HDD.
  The following is a summary of performance and use cases for each volume type.
  - General Purpose SSD volumes (gp2 and gp3) balance price and performance for a wide variety of transactional workloads. These volumes are ideal for use cases such as boot volumes, medium-size single instance databases, and development and test environments.
  - Provisioned IOPS SSD volumes (io1 and io2) are designed to meet the needs of I/O-intensive workloads that are sensitive to storage performance and consistency. They provide a consistent IOPS rate that you specify when you create the volume. This enables you to predictably scale to tens of thousands of IOPS per instance. Additionally, io2 volumes provide the highest levels of volume durability.
  - Throughput Optimized HDD volumes (st1) provide low-cost magnetic storage that defines performance in terms of throughput rather than IOPS. These volumes are ideal for large, sequential workloads such as Amazon EMR, ETL, data warehouses, and log processing.
  - Cold HDD volumes (sc1) provide low-cost magnetic storage that defines performance in terms of throughput rather than IOPS. These volumes are ideal for large, sequential, cold-data workloads. If you require infrequent access to your data and are looking to save costs, these volumes provides inexpensive block storage.
- You can create your EBS volumes as encrypted volumes, in order to meet a wide range of data-at-rest encryption requirements for regulated/audited data and applications. When you create an encrypted EBS volume and attach it to a supported instance type, data stored at rest on the volume, disk I/O, and snapshots created from the volume are all encrypted. The encryption occurs on the servers that host EC2 instances, providing encryption of data-in-transit from EC2 instances to EBS storage.
- You can create point-in-time snapshots of EBS volumes, which are persisted to Amazon S3. Snapshots protect data for long-term durability, and they can be used as

the starting point for new EBS volumes. The same snapshot can be used to create as many volumes as needed. These snapshots can be copied across AWS Regions.

- Performance metrics, such as bandwidth, throughput, latency, and average queue length, are available through the AWS Management Console. These metrics, provided by Amazon CloudWatch, allow you to monitor the performance of your volumes to make sure that you are providing enough performance for your applications without paying for resources you don't need.

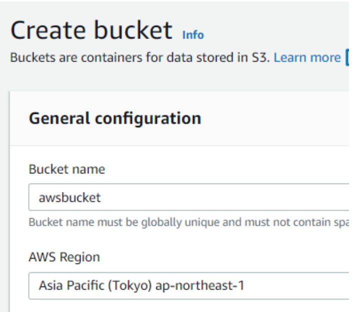
## S3

You can get started with Amazon S3 by working with buckets and objects. A bucket is a container for objects. An object is a file and any metadata that describes that file.

To store an object in Amazon S3, you create a bucket and then upload the object to the bucket. When the object is in the bucket, you can open it, download it, and move it. When you no longer need an object or a bucket, you can clean up your resources.

Prerequisite: Setting up Amazon S3

Step 1: Create your first S3 bucket



Create bucket Info

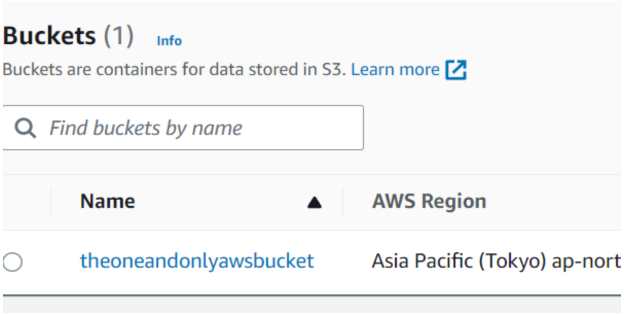
Buckets are containers for data stored in S3. Learn more

General configuration

Bucket name: awsbucket

Bucket name must be globally unique and must not contain spa

AWS Region: Asia Pacific (Tokyo) ap-northeast-1



Buckets (1) Info

Buckets are containers for data stored in S3. Learn more

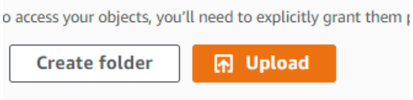
|  | Name ▲ | AWS Region |
|---|---|---|
| ○ | theoneandonlyawsbucket | Asia Pacific (Tokyo) ap-nort |

Step 2: Upload an object to your bucket

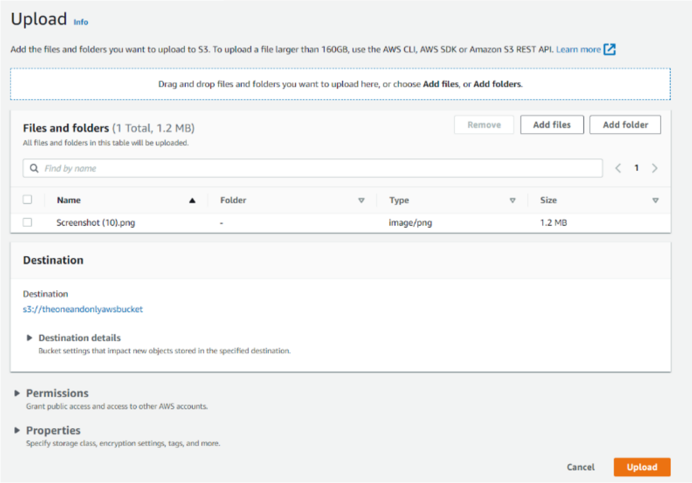
Step 3: Download an object

Step 4: Copy your object to a folder

Step 5: Delete your objects and bucket



o access your objects, you'll need to explicitly grant them



Upload Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. Learn more

Drag and drop files and folders you want to upload here, or choose **Add files**, or **Add folders**.

Files and folders (1 Total, 1.2 MB)

All files and folders in this table will be uploaded.

| □ | Name ▲ | Folder | Type | Size |
|---|---|---|---|---|
| □ | Screenshot (10).png | - | image/png | 1.2 MB |

Destination

Destination: s3://theoneandonlyawsbucket

▸ Destination details

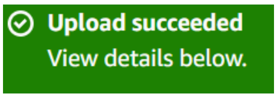
Bucket settings that impact new objects stored in the specified destination.

▸ Permissions

Grant public access and access to other AWS accounts.

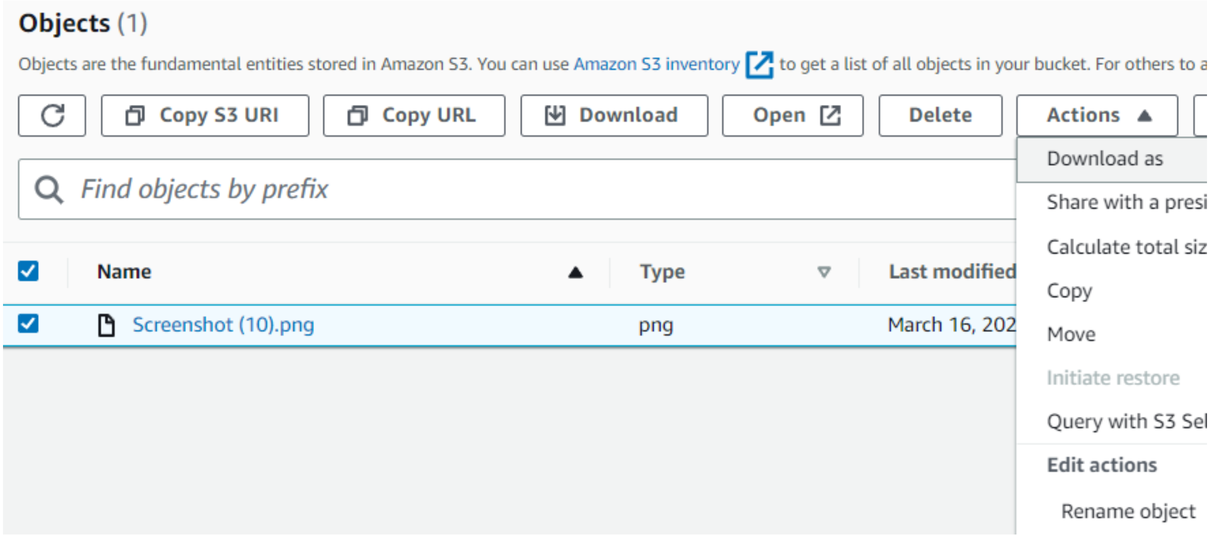
▸ Properties

Specify storage class, encryption settings, tags, and more.



Upload succeeded

View details below.



**Objects** (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to a

| ☑ | Name ▲ | Type ▽ | Last modified |
|---|---|---|---|
| ☑ | Screenshot (10).png | png | March 16, 202 |

**Conclusion:**

Explored the working of Storage Options

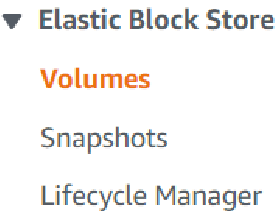
# Practical No.: 7

**Aim:** Explore the working of the following: Storage Overview

- Working With Volumes
- Working with Volume Snapshots

**Theory:**

1. Open EC2
2. Open Elastic Block Storage



▼ Elastic Block Store

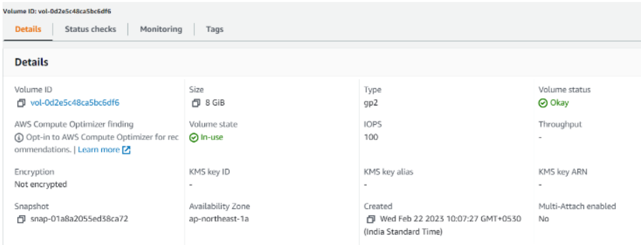
Volumes

Snapshots

Lifecycle Manager

3. Select volumes

**Check details of volume**

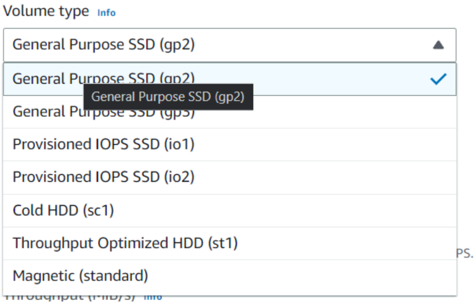


Volume ID: vol-0d2e5c48ca5bc6df6

Details

| Volume ID | Size | Type | Volume status |
|---|---|---|---|
| vol-0d2e5c48ca5bc6df6 | 8 GiB | gp2 | Okay |
| AWS Compute Optimizer finding | Volume state | IOPS | Throughput |
| Opt-in to AWS Compute Optimizer for recommendations. \| Learn more | In-use | 100 | - |
| Encryption | KMS key ID | KMS key alias | KMS key ARN |
| Not encrypted | - | - | - |
| Snapshot | Availability Zone | Created | Multi-Attach enabled |
| snap-01a8a2055ed38ca72 | ap-northeast-1a | Wed Feb 22 2023 10:07:27 GMT+0530 (India Standard Time) | No |

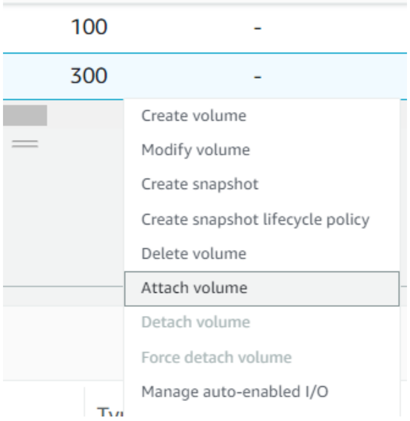
**Create volume**



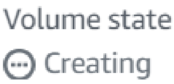
Volume type Info

General Purpose SSD (gp2)

General Purpose SSD (gp2)
General Purpose SSD (gp2)
General Purpose SSD (gp3)
Provisioned IOPS SSD (io1)
Provisioned IOPS SSD (io2)
Cold HDD (sc1)
Throughput Optimized HDD (st1)
Magnetic (standard)



1. Select volume type
2. Create volume
3. Wait for volume to be created
4. Select the created volume, right-click and select the “attach volume” option.
5. Select the ec2 instance from the instance text.

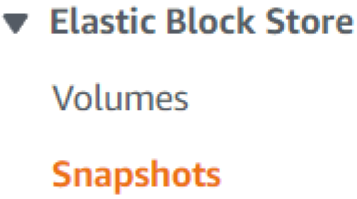


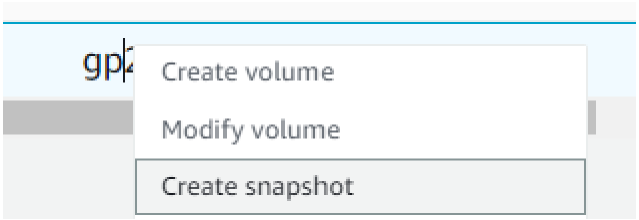
Volume state

Creating

**Snapshots**

1. Open Volumes





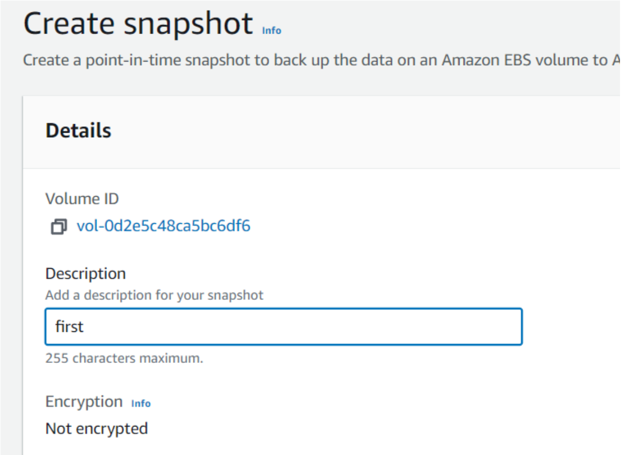
2. Select the volume of which you want to create snapshot of.

3. Right click on the volume.

4. Click create snapshot

5. Enter the snapshot name

6. Click create



Create snapshot Info

Create a point-in-time snapshot to back up the data on an Amazon EBS volume to A

Details

Volume ID

vol-0d2e5c48ca5bc6df6

Description

Add a description for your snapshot

first

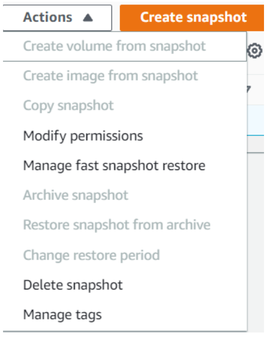
255 characters maximum.

Encryption Info

Not encrypted

7. Wait for the snapshot to be created.

8. Once created you can use the snapshot to create new volume or images from the snapshot.



**Conclusion:**

Explored the working of Volumes.

# Practical No.: 8

**Aim:** Explore managing the Cloud using following:

- Tags to Organize Resources in the Cloud
- Reporting CPU Sockets

**Theory:**

### Use Tagging to Organize Your Environment and Drive Accountability

Tagging your AWS resources lets you assign custom metadata to instances, images, and other resources. For example, you can categorize resources by owner, purpose, or environment, which helps you organize them and assign cost accountability.

### Define Mandatory Cost Tagging

An effective tagging strategy will give you improved visibility and monitoring, help you create accurate chargeback/showback models, and get more granular and precise insights into usage and spend by applications and teams. The following tag categories can help you achieve these goals:

- **Environment** – Distinguishes between development, test, and production infrastructure. Specifying an environment tag reduces analysis time, post-processing, and the need to maintain a separate mapping file of production versus non-production accounts.

- **Application ID** – Identifies resources that are related to a specific application for easy tracking of spend change and turn-off at the end of projects.

- **Automation Opt-In/Opt-Out** – Indicates whether a resource should be included in an automated activity such as starting, stopping, or resizing instances.

- **Cost Center/Business Unit** – Identifies the cost center or business unit associated with a resource, typically for cost allocation and tracking.

- **Owner** – Used to identify who is responsible for the resource. This is typically the technical owner. If needed, you can add a separate business owner tag. You can specify the owner as an email address. Using an email address supports automated notifications to both the technical and business owners as required (e.g., if the resource is a candidate for elasticity or right sizing).

### Enforce Quality of Tagging

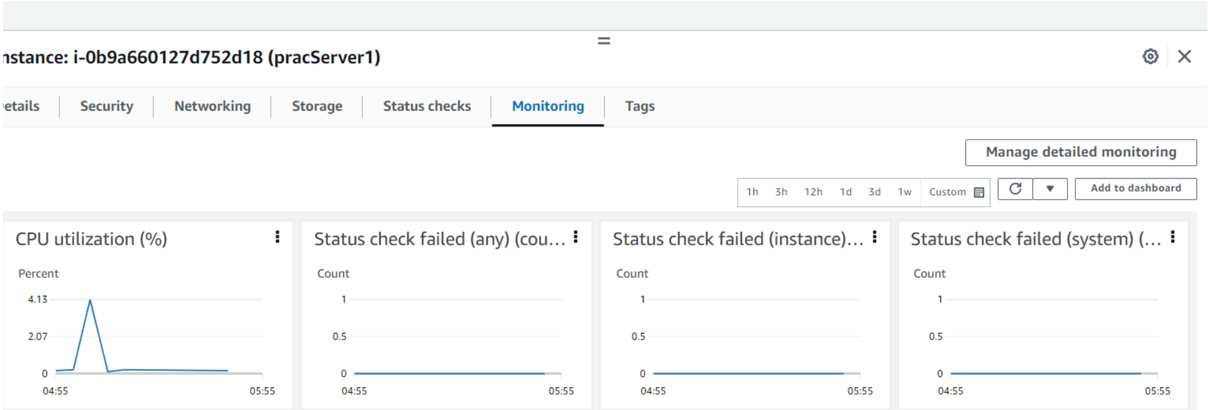
Without enforcement, tagging quality will be low, and reporting will be manual, time-consuming, and subject to debate. There are two general types of tagging enforcement: soft and hard. Soft enforcement notifies users when they have not followed policies. Hard enforcement terminates resources that are not tagged to the company standard (usually within hours after they're launched). Mature organizations find hard enforcement of tagging to be the best way to ensure that quality tagging is maintained.

### Tagging Tools

The following tools can help you manage your tags:

- Tag Editor – Finds resources with search criteria (including missing and misspelled tags) and allows you to edit tags via the AWS Management Console
- AWS Config Managed Rules – Identifies resources that do not comply to tagging policies
- Capital One's Cloud Custodian (open source) – Ensures tagging compliance and remediation

### Select instance and move to monitoring tags



nstance: i-0b9a660127d752d18 (pracServer1)

### Conclusion:

Explored the working of tags and reporting.

# Practical No.: 9

**Aim:** Explore managing the Cloud using following:

- Changing the Database Configuration
- File encryption type

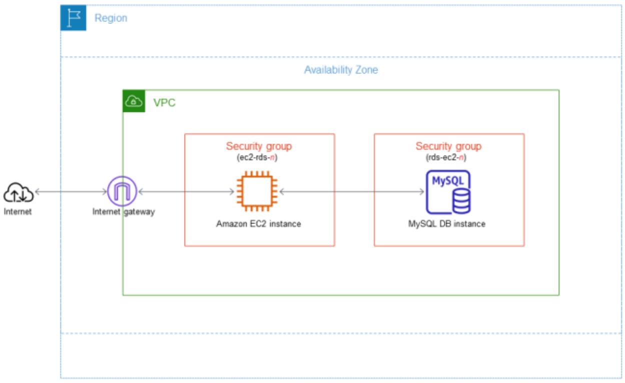
**Theory:**

### Creating and connecting to a MySQL DB instance

Here we create an EC2 instance and an RDS for MySQL DB instance. The tutorial shows you how to access the DB instance from the EC2 instance using a standard MySQL client. As a best practice, this tutorial creates a private DB instance in a virtual private cloud (VPC). In most cases, other resources in the same VPC, such as EC2 instances, can access the DB instance, but resources outside of the VPC can't access it.

The following diagram shows the configuration when the tutorial is complete.

This tutorial uses **Easy create** to create a DB instance running MySQL with the AWS Management Console. With **Easy create**, you specify only the DB engine type, DB instance size, and DB instance identifier. **Easy create** uses the default settings for the other configuration options. The DB instance created by **Easy create** is private.



When you use **Standard create** instead of **Easy create**, you can specify more configuration options when you create a DB instance, including ones for availability, security, backups, and maintenance. To create a public DB instance, you must use **Standard create**.

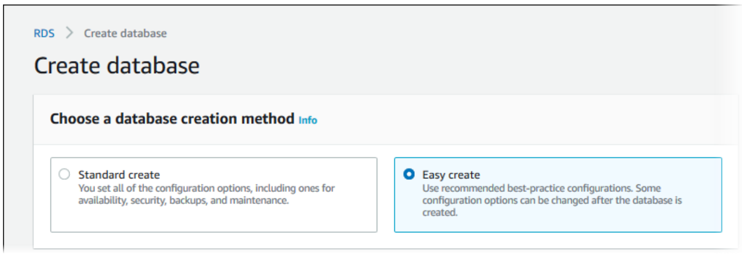
### Create a MySQL DB instance

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 in which you want to create the DB instance.
3. In the navigation pane, choose **Databases**.
4. Choose **Create database** and make sure that **Easy create** is chosen.



RDS > Create database

Create database

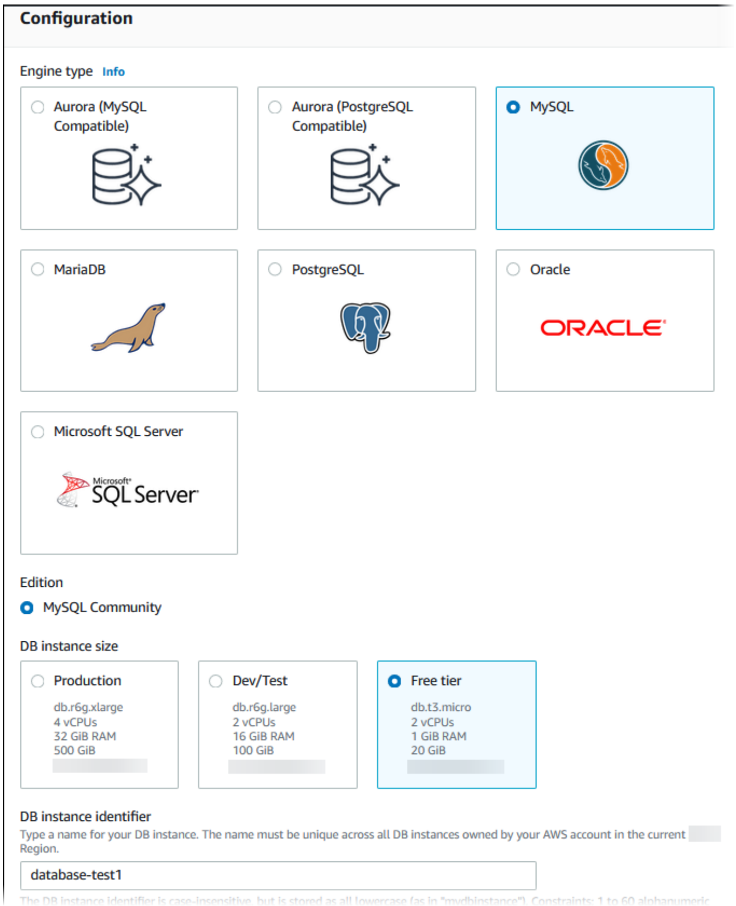
Choose a database creation method Info

○ Standard create
You set all of the configuration options, including ones for availability, security, backups, and maintenance.

● Easy create
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

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

○ 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 Region.

database-test1

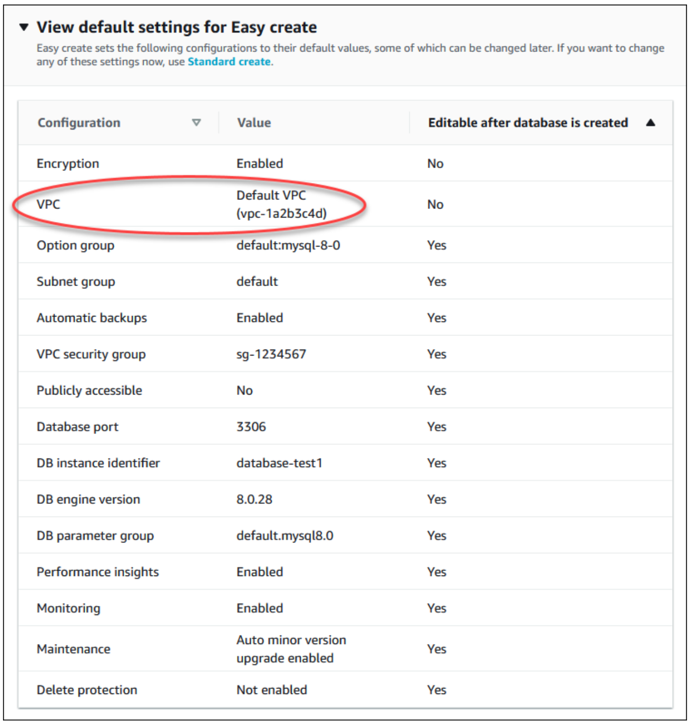
The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric

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. Open **View default settings for Easy create**.

    Note the setting for **VPC**. Your DB instance and EC2 instance must reside in the same VPC to set up connectivity between them automatically in a later step. If you didn't create a new VPC in the AWS Region, then the default VPC is selected.



▼ 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<br>(vpc-1a2b3c4d) | No |
| Option group | default:mysql-8-0 | Yes |
| Subnet group | default | Yes |
| Automatic backups | Enabled | Yes |
| VPC security group | sg-1234567 | 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<br>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.

11. Choose **Create database**.

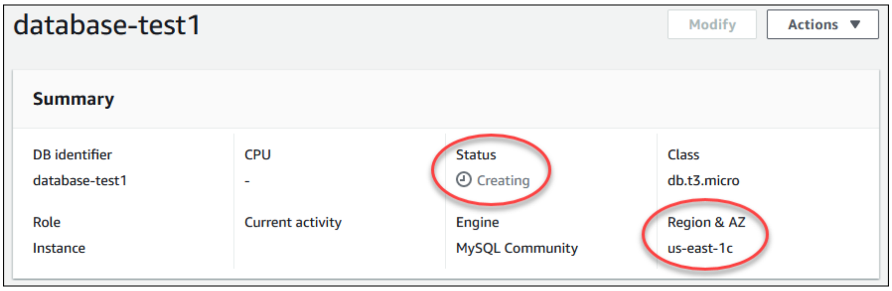
    To view the master user name and password for the DB instance, choose **View credential details**.

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

12. 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.

    Wait for the **Region & AZ** value to appear. When it appears, make a note of the value because you need it later. In the following image, the **Region & AZ** value is **us-east-1c**.



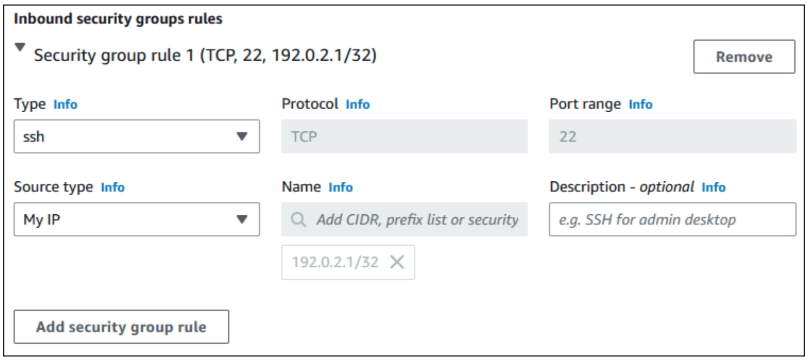
database-test1

Summary

| DB identifier | CPU | Status | Class |
|---|---|---|---|
| database-test1 | - | Creating | db.t3.micro |
| Role | Current activity | Engine | Region & AZ |
| Instance | | MySQL Community | 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. While the DB instance is being created, you can move on to the next step and create an EC2 instance.

**Update the inbound rules of Ec2 instance**



Inbound security groups rules

Security group rule 1 (TCP, 22, 192.0.2.1/32)

Type: ssh
Protocol: TCP
Port range: 22
Source type: My IP
Name: 192.0.2.1/32
Description - *optional*

**Connect your EC2 instance and MySQL DB instance automatically**

You can automatically connect an existing EC2 instance to a DB instance using the RDS console. The RDS console simplifies setting up the connection between an EC2 instance and your MySQL DB instance. For this tutorial, set up a connection between the EC2 instance and the MySQL DB instance that you created previously.

Before setting up a connection between an EC2 instance and an RDS database, make sure you meet the requirements described in Overview of automatic connectivity with an EC2 instance.

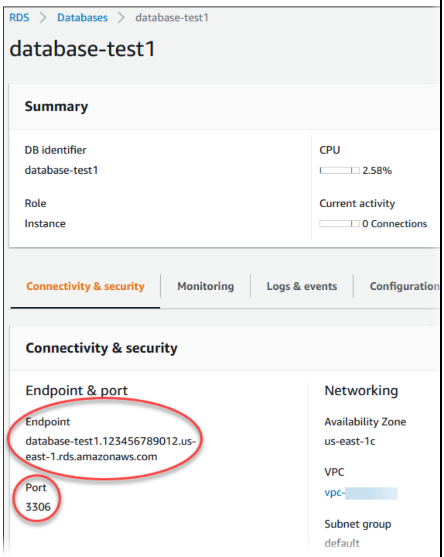
If you change these security groups after you configure connectivity, the changes might affect the connection between the EC2 instance and the RDS database

**Connect your EC2 instance and MySQL DB instance automatically**

You can automatically connect an existing EC2 instance to a DB instance using the RDS console. The RDS console simplifies setting up the connection between an EC2 instance and your MySQL DB instance. For this tutorial, set up a connection between the EC2 instance and the MySQL DB instance that you created previously.

Before setting up a connection between an EC2 instance and an RDS database, make sure you meet the requirements described in Overview of automatic connectivity with an EC2 instance.

If you change these security groups after you configure connectivity, the changes might affect the connection between the EC2 instance and the RDS database



database-test1

Summary

DB identifier: database-test1

Role: Instance

CPU: 2.58%

Current activity: 0 Connections

Connectivity & security

Endpoint & port

Endpoint: database-test1.123456789012.us-east-1.rds.amazonaws.com

Port: 3306

Networking

Availability Zone: us-east-1c

VPC: vpc-[redacted]

Subnet group: default

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

```
sudo yum update -y
```

Install the mysql command-line client from MariaDB

Most Linux distributions, including Amazon Linux 2, include the mysql command-line client from MariaDB instead of the mysql command-line client from Oracle. To install the mysql command-line client from MariaDB on Amazon Linux 2, run the following command:

```
sudo yum install mariadb
```

4. Connect to the MySQL DB instance. For example, enter the following command at a command prompt on a client computer. 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 user name 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 12662

   Server version: 8.0.28 Source distribution
   ```

   Run SQL commands.

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

   ```
   SELECT CURRENT_TIMESTAMP;
   ```

**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.

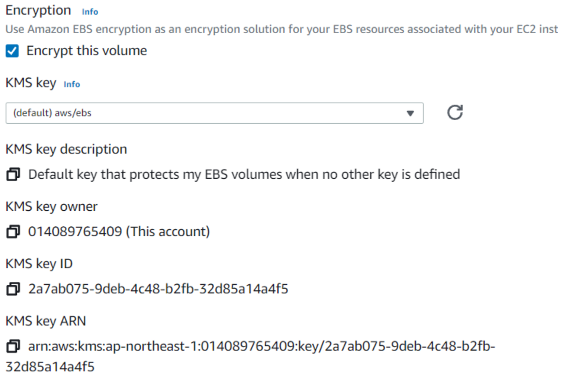
**To delete the EC2 instance**

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.

**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**.



Encryption Info

Use Amazon EBS encryption as an encryption solution for your EBS resources associated with your EC2 inst

☑ Encrypt this volume

KMS key Info

(default) aws/ebs ▼

KMS key description

Default key that protects my EBS volumes when no other key is defined

KMS key owner

014089765409 (This account)

KMS key ID

2a7ab075-9deb-4c48-b2fb-32d85a14a4f5

KMS key ARN

arn:aws:kms:ap-northeast-1:014089765409:key/2a7ab075-9deb-4c48-b2fb-32d85a14a4f5

**Conclusion:**

Explore database and file encryption.

# Practical No.: 10

**Aim:** Explore managing the Cloud using following:

- Administrator Alerts
- Customizing the Network Domain Name

**Theory:**

You can create metric and composite alarms in Amazon CloudWatch.

- A metric alarm watches a single CloudWatch metric or the result of a math expression based on CloudWatch metrics. The alarm performs one or more actions based on the value of the metric or expression relative to a threshold over a number of time periods. The action can be sending a notification to an Amazon SNS topic, performing an Amazon EC2 action or an Amazon EC2 Auto Scaling action, or creating an OpsItem or incident in Systems Manager.
- A composite alarm includes a rule expression that takes into account the alarm states of other alarms that you have created. The composite alarm goes into ALARM state only if all conditions of the rule are met. The alarms specified in a composite alarm's rule expression can include metric alarms and other composite alarms.

Using composite alarms can reduce alarm noise. You can create multiple metric alarms, and also create a composite alarm and set up alerts only for the composite alarm. For example, a composite might go into ALARM state only when all of the underlying metric alarms are in ALARM state.

Composite alarms can send Amazon SNS notifications when they change state, and can create Systems Manager OpsItems or incidents when they go into ALARM state, but can't perform EC2 actions or Auto Scaling actions.

**Note**

You can create as many alarms as you want in your AWS account.

You can add alarms to dashboards, so you can monitor and receive alerts about your AWS resources and applications across multiple regions. After you add an alarm to a dashboard, the alarm turns gray when it's in the INSUFFICIENT_DATA state and red when it's in the ALARM state. The alarm is shown with no color when it's in the OK state.

You also can favorite recently visited alarms from the Favorites and recents option in the CloudWatch console navigation pane. The Favorites and recents option has columns for your favorited alarms and recently visited alarms.

**Metric alarm states**

A metric alarm has the following possible states:

- OK – The metric or expression is within the defined threshold.
- ALARM – The metric or expression is outside of the defined threshold.

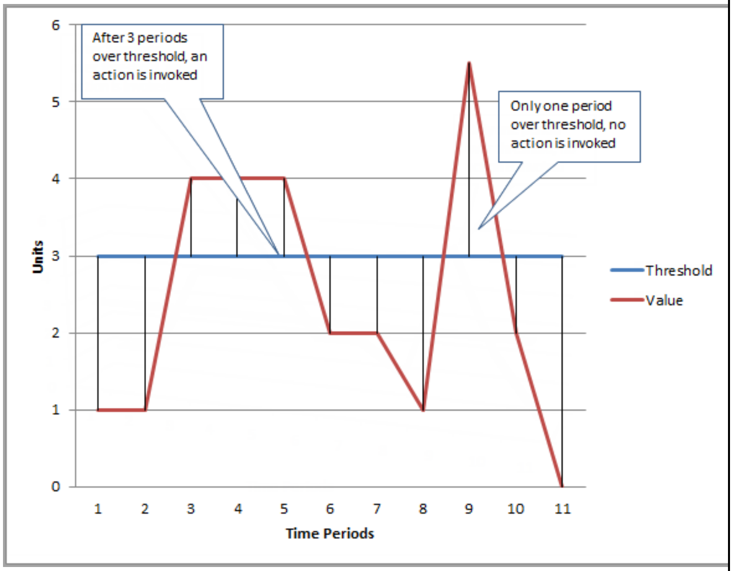
- INSUFFICIENT_DATA – The alarm has just started, the metric is not available, or not enough data is available for the metric to determine the alarm state.

### Evaluating an alarm

When you create an alarm, you specify three settings to enable CloudWatch to evaluate when to change the alarm state:

- **Period** is the length of time to evaluate the metric or expression to create each individual data point for an alarm. It is expressed in seconds. If you choose one minute as the period, the alarm evaluates the metric once per minute.
- **Evaluation Periods** is the number of the most recent periods, or data points, to evaluate when determining alarm state.
- **Datapoints to Alarm** is the number of data points within the Evaluation Periods that must be breaching to cause the alarm to go to the ALARM state. The breaching data points don't have to be consecutive, but they must all be within the last number of data points equal to **Evaluation Period**.

In the following figure, the alarm threshold for a metric alarm is set to three units. Both **Evaluation Period** and **Datapoints to Alarm** are 3. That is, when all existing data points in the most recent three consecutive periods are above the threshold, the alarm goes to ALARM state. In the figure, this happens in the third through fifth time periods. At period six, the value dips below the threshold, so one of the periods being evaluated is not breaching, and the alarm state changes back to OK. During the ninth time period, the threshold is breached again, but for only one period. Consequently, the alarm state remains OK.



When you configure **Evaluation Periods** and **Datapoints to Alarm** as different values, you're setting an "M out of N" alarm. **Datapoints to Alarm** is ("M") and **Evaluation Periods** is ("N"). The evaluation interval is the number of data points multiplied by the period. For example, if you configure 4 out of 5 data points with a period of 1 minute, the evaluation interval is 5 minutes. If you configure 3 out of 3 data points with a period of 10 minutes, the evaluation interval is 30 minutes.

An alarm invokes actions only when the alarm changes state. The exception is for alarms with Auto Scaling actions. For Auto Scaling actions, the alarm continues to invoke the action once per minute that the alarm remains in the new state.

An alarm can watch a metric in the same account. If you have enabled cross-account functionality in your CloudWatch console, you can also create alarms that watch metrics in other AWS accounts. Creating cross-account composite alarms is not supported. Creating cross-account alarms that use math expressions is supported, except that

the ANOMALY_DETECTION_BAND, INSIGHT_RULE,
and SERVICE_QUOTA functions are not supported for cross-account alarms.

**Alarm actions**

You can specify what actions an alarm takes when it changes state between the OK, ALARM, and INSUFFICIENT_DATA states. The most common type of alarm action is to notify one or more people by sending a message to an Amazon Simple Notification Service topic.

**Common features of CloudWatch alarms**

The following features apply to all CloudWatch alarms:

- There is no limit to the number of alarms that you can create. To create or update an alarm, you use the CloudWatch console, the PutMetricAlarm API action, or the put-metric-alarm command in the AWS CLI.
- Alarm names must contain only UTF-8 characters, and can't contain ASCII control characters
- You can list any or all of the currently configured alarms, and list any alarms in a particular state by using the CloudWatch console, the DescribeAlarms API action, or the describe-alarms command in the AWS CLI.
- You can disable and enable alarms by using the DisableAlarmActions and EnableAlarmActions API actions, or the disable-alarm-actions and enable-alarm-actions commands in the AWS CLI.
- You can test an alarm by setting it to any state using the SetAlarmState API action or the set-alarm-state command in the AWS CLI. This temporary state change lasts only until the next alarm comparison occurs.
- You can create an alarm for a custom metric before you've created that custom metric. For the alarm to be valid, you must include all of the dimensions for the custom metric in addition to the metric namespace and metric name in the alarm definition. To do this, you can use the PutMetricAlarm API action, or the put-metric-alarm command in the AWS CLI.
- You can view an alarm's history using the CloudWatch console, the DescribeAlarmHistory API action, or the describe-alarm-history command in the AWS CLI. CloudWatch preserves alarm history for two weeks. Each state transition is marked with a unique timestamp. In rare cases, your history might show more than one notification for a state change. The timestamp enables you to confirm unique state changes.
- You can favorite alarms from the Favorites and recents option in the CloudWatch console navigation pane by hovering over the alarm that you want to favorite and choosing the star symbol next to it.

- The number of evaluation periods for an alarm multiplied by the length of each evaluation period can't exceed one day.

Customize Domain name

Domain Name System (DNS) is a standard by which names used on the internet are resolved to their corresponding IP addresses. A DNS hostname is a name that uniquely and absolutely names a computer; it's composed of a host name and a domain name. DNS servers resolve DNS hostnames to their corresponding IP addresses.

**Amazon DNS server**

The Route 53 Resolver (also called "Amazon DNS server" or "AmazonProvidedDNS") is a DNS Resolver service which is built into each availability zone within an AWS Region. The Route 53 Resolver is located at 169.254.169.253 (IPv4), fd00:ec2::253 (IPv6), and at the primary private IPV4 CIDR range provisioned to your VPC plus two. For example, if you have a VPC with an IPv4 CIDR of 10.0.0.0/16 and an IPv6 CIDR of fd00:ec2::253, you can reach the Route 53 Resolver at 169.254.169.253 (IPv4), fd00:ec2::253 (IPv6), or 10.0.0.2 (IPv4).

When you launch an instance into a VPC, we provide the instance with a private DNS hostname. We also provide a public DNS hostname if the instance is configured with a public IPv4 address and the VPC DNS attributes are enabled.

The format of the private DNS hostname depends on how you configure the EC2 instance when you launch it. For more information on the types of private DNS hostnames, see EC2 instance naming.

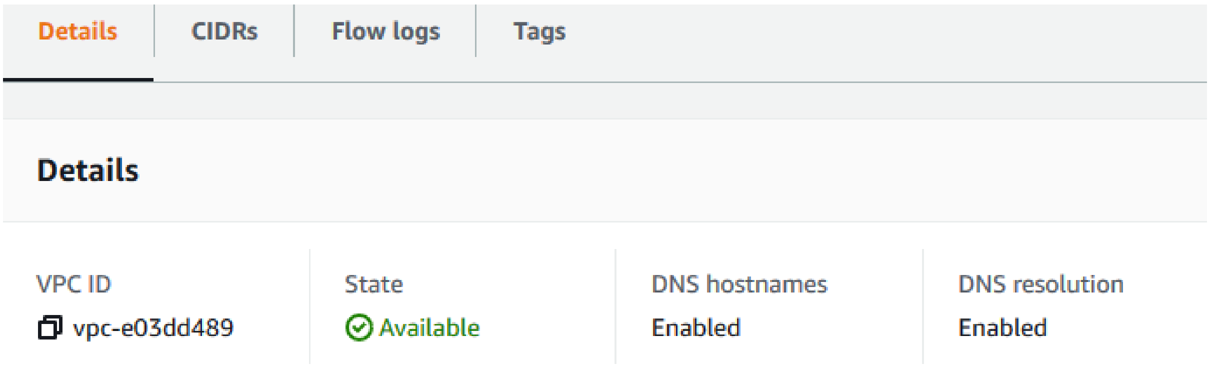
The Amazon DNS server in your VPC is used to resolve the DNS domain names that you specify in a private hosted zone in Route 53. For more information about private hosted zones, see Working with private hosted zones in the Amazon Route 53 Developer Guide.

**View and update DNS attributes for your VPC**

You can view and update the DNS support attributes for your VPC using the Amazon VPC console.

**To describe and update DNS support for a VPC using the console**

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Your VPCs**.
3. Select the checkbox for the VPC.
4. Review the information in **Details**. In this example, both **DNS hostnames** and **DNS resolution** are enabled.



Details

| VPC ID | State | DNS hostnames | DNS resolution |
|---|---|---|---|
| vpc-e03dd489 | ✓ Available | Enabled | Enabled |

5. To update these settings, choose **Actions** and then choose **Edit VPC settings**. Select or clear **Enable** on the appropriate DNS attribute and choose **Save changes**.

**To describe DNS support for a VPC using the command line**

You can use one of the following commands. For more information about these command line interfaces, see Working with Amazon VPC.

- describe-vpc-attribute (AWS CLI)
- Get-EC2VpcAttribute (AWS Tools for Windows PowerShell)

**To update DNS support for a VPC using the command line**

You can use one of the following commands. For more information about these command line interfaces, see Working with Amazon VPC.

- modify-vpc-attribute (AWS CLI)
- Edit-EC2VpcAttribute (AWS Tools for Windows PowerShell)

**Private hosted zones**

To access the resources in your VPC using custom DNS domain names, such as example.com, instead of using private IPv4 addresses or AWS-provided private DNS hostnames, you can create a private hosted zone in Route 53. A private hosted zone is a container that holds information about how you want to route traffic for a domain and its subdomains within one or more VPCs without exposing your resources to the internet. You can then create Route 53 resource record sets, which determine how Route 53 responds to queries for your domain and subdomains. For example, if you want browser requests for example.com to be routed to a web server in your VPC, you'll create an A record in your private hosted zone and specify the IP address of that web server. For more information about creating a private hosted zone, see Working with private hosted zones in the Amazon Route 53 Developer Guide.

To access resources using custom DNS domain names, you must be connected to an instance within your VPC. From your instance, you can test that your resource in your private hosted zone is accessible from its custom DNS name by using the ping command; for example, ping mywebserver.example.com. (You must ensure that your instance's security group rules allow inbound ICMP traffic for the ping command to work.)

**Configuring Amazon Route 53 to route traffic to an Amazon EC2 instance**

To configure Amazon Route 53 to route traffic to an EC2 instance, perform the following procedure.

**To route traffic to an Amazon EC2 instance**

1. Get the IP address for the Amazon EC2 instance:
    a. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
    b. In the Regions list in the upper right corner of the console, choose the Region that you launched the instance in.
    c. In the navigation pane, choose **Instances**.
    d. In the table, choose the instance that you want to route traffic to.
    e. In the bottom pane, on the **Description** tab, get the value of **Elastic IPs**.

If you didn't associate an Elastic IP with the instance, get the value of **IPv4 Public IP**.

2. Open the Route 53 console at https://console.aws.amazon.com/route53/.
3. In the navigation pane, choose **Hosted zones**.
4. Choose the name of the hosted zone that matches the name of the domain that you want to route traffic for.
5. Choose **Create record**.
6. Specify the following values:

    **Routing policy**

    Choose the applicable routing policy. For more information, see Choosing a routing policy.

    **Record name**

    Enter the domain name that you want to use to route traffic to your EC2 instance. The default value is the name of the hosted zone.

    For example, if the name of the hosted zone is example.com and you want to use acme.example.com to route traffic to your EC2 instance, enter **acme**.

    **Value/Route traffic to**

    Choose **IP address or another value depending on the record type**. Enter the IP address that you got in step 1.

    **Record type**

    Choose **A – IPv4 address**.

    **TTL (seconds)**

    Accept the default value of **300**.

7. Choose **Create records**.

Changes generally propagate to all Route 53 servers within 60 seconds. When propagation is done, you'll be able to route traffic to your EC2 instance by using the name of the record that you created in this procedure.

**Conclusion:**

Explored Administrator alerts and network domain names