

# EKS Provisioning using Terraform

---

- We will provision the EKS using Terraform as an Infrastructure as Code.
  - We will deploy it in a custom Virtual Private Cloud for isolation.
  - We will also deploy RDS MySQL Instance to store the relational data and connect it to EKS.
- 

## Prerequisites

---

1. AWS Account with an IAM User with administrative permissions.
  2. Terraform installed.
  3. Kubectl installed.
- 

## Write Terraform Configuration files

---

First, we will write Terraform configuration files for AWS resources using predefined modules available on the internet.

## Steps

1. Create the **eks-terraform** directory.
2. The folder structure for the above-created directory is as follows:

```
eks-terraform
├── .terraform.lock.hcl
├── locals.tf
├── main.tf
├── outputs.tf
├── providers.tf
├── terraform.tfstate
├── terraform.tfstate.backup
└── .terraform
```

We need to only create *providers.tf*, *main.tf*, *outputs.tf*, & *locals.tf* file. Other files are generated while initiating terraform.

3. Create a *providers.tf* file inside the above-created directory.
4. Inside it, define the following:
  - terraform
    - required\_providers
  - provider
    - aws
5. Click [code](#) for reference.
6. The definition of *providers.tf* file is complete.

7. Now, create the *main.tf* file.
8. Inside *main.tf* file, we will use the following predefined modules:
  - module.vpc
  - module.rds
  - module.eks
9. Click [code](#) for reference.
10. The definition of *main.tf* file is complete.
11. Now we will create *outputs.tf* file.
12. Inside it, define the following outputs.
  - output.DB\_HOST
  - output.bastion-host-ip
13. Click [code](#) for reference.
14. The definition of *outputs.tf* file is complete.
15. Now we will create *locals.tf* file.
16. Inside it, define the following variables:
  - local.vpc-properties
  - local.database-properties
  - local.bastion-properties
  - local.eks-properties
17. Click [code](#) for reference.
18. The definition of *locals.tf* file is complete.

Ensure you give the appropriate values to the variables defined in *locals.tf* file.

---

## Provisioning the Infrastructure

---

Now we will provision the AWS infrastructure by applying the above-created configuration files.

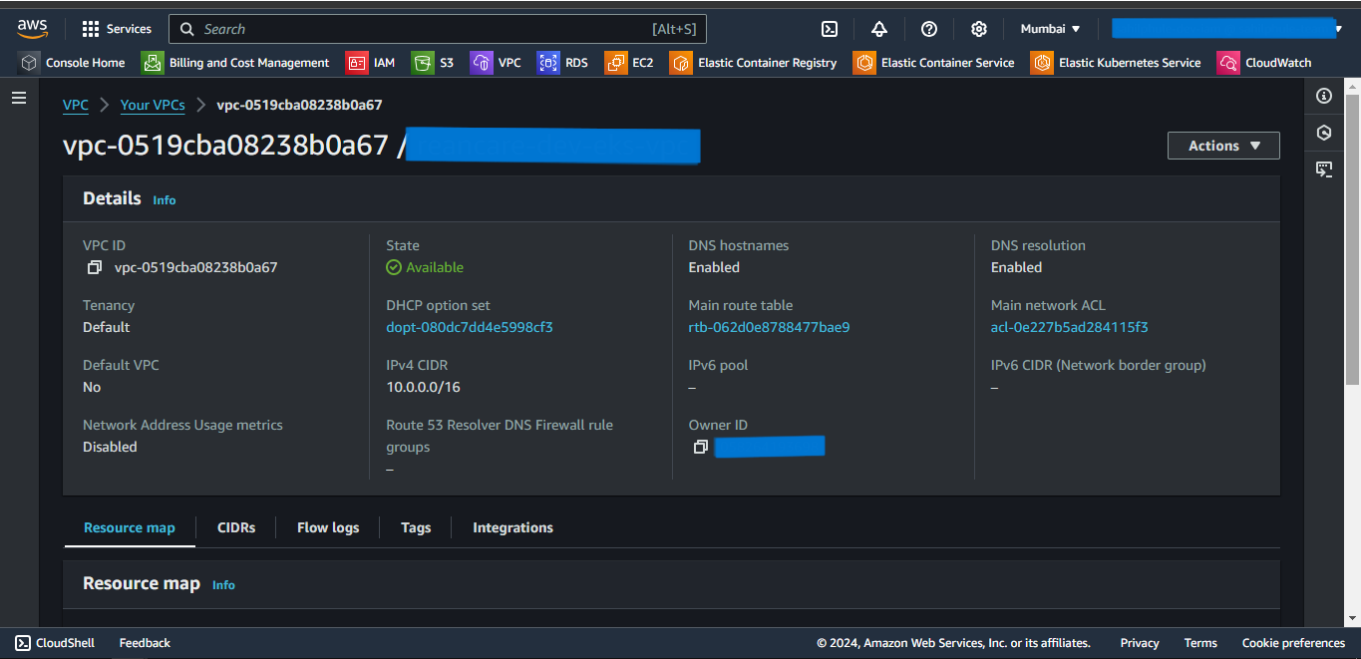
Ensure AWS CLI is configured with appropriate AWS user credentials and enough permissions.

### Steps:

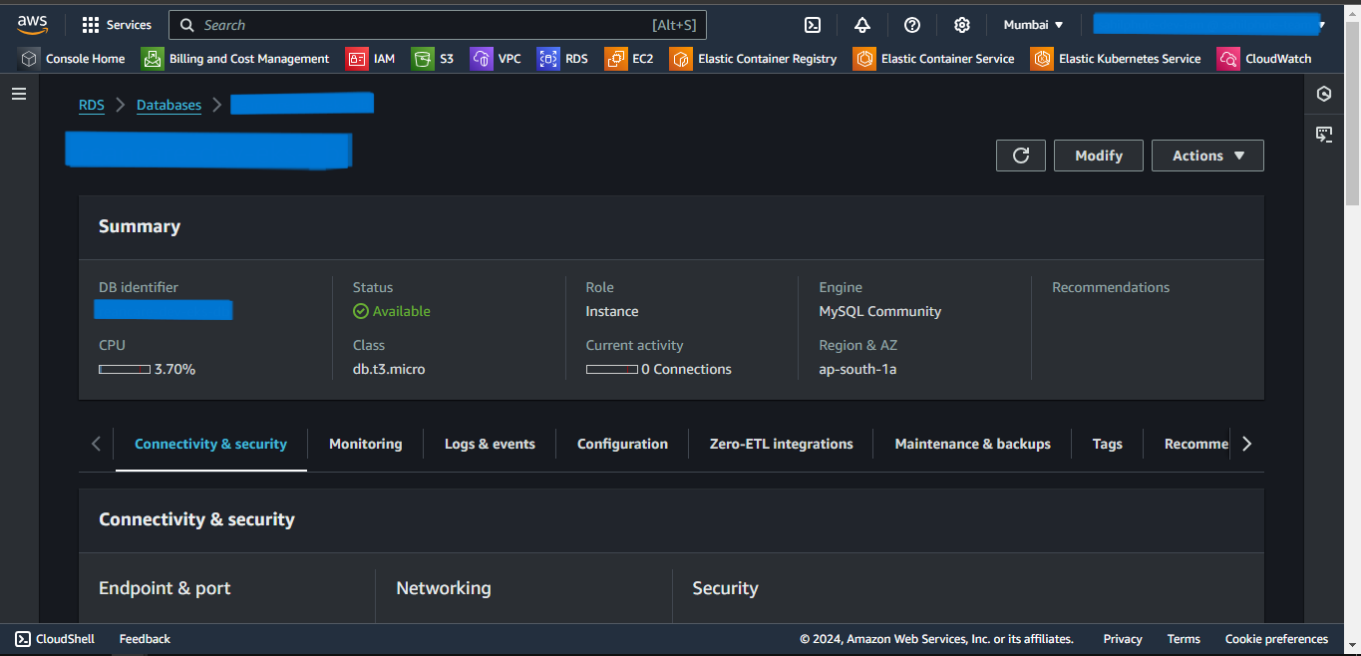
1. Open the PowerShell.
  2. Change the directory to the above-created **eks-terraform** directory using **cd** command.
  3. Run the **terraform fmt -recursive** command to format the syntax of the files.
  4. Run the **terraform init** command to initialize the *terraform*.
  5. Run the **terraform validate** command to validate the configuration files.
  6. Run the **terraform plan** command to plan the resources to be created.
  7. Run the **terraform apply** command and if prompted, type **yes** to provision the infrastructure.
  8. Run the **terraform output** command to get the values of defined variables in *outputs.tf* file.
  9. Head to the AWS Console, and verify the created resources.
-

# Screenshots of Provisioned Infrastructure

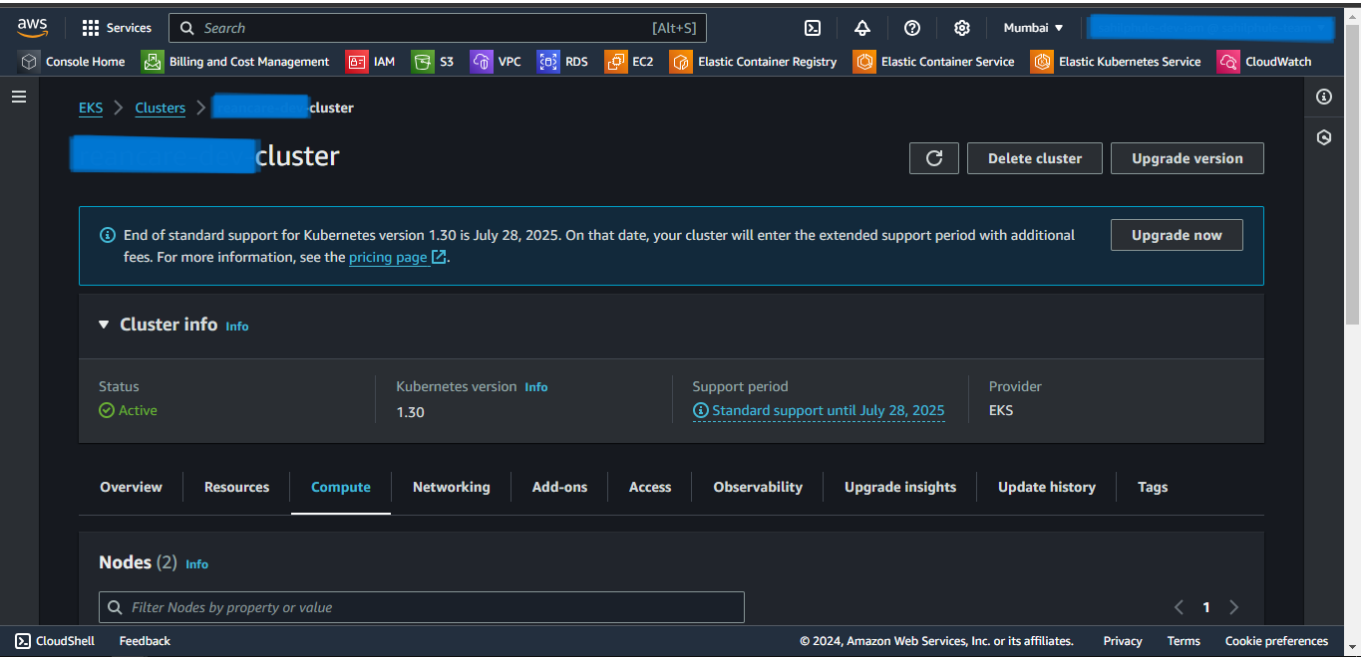
## VPC Image



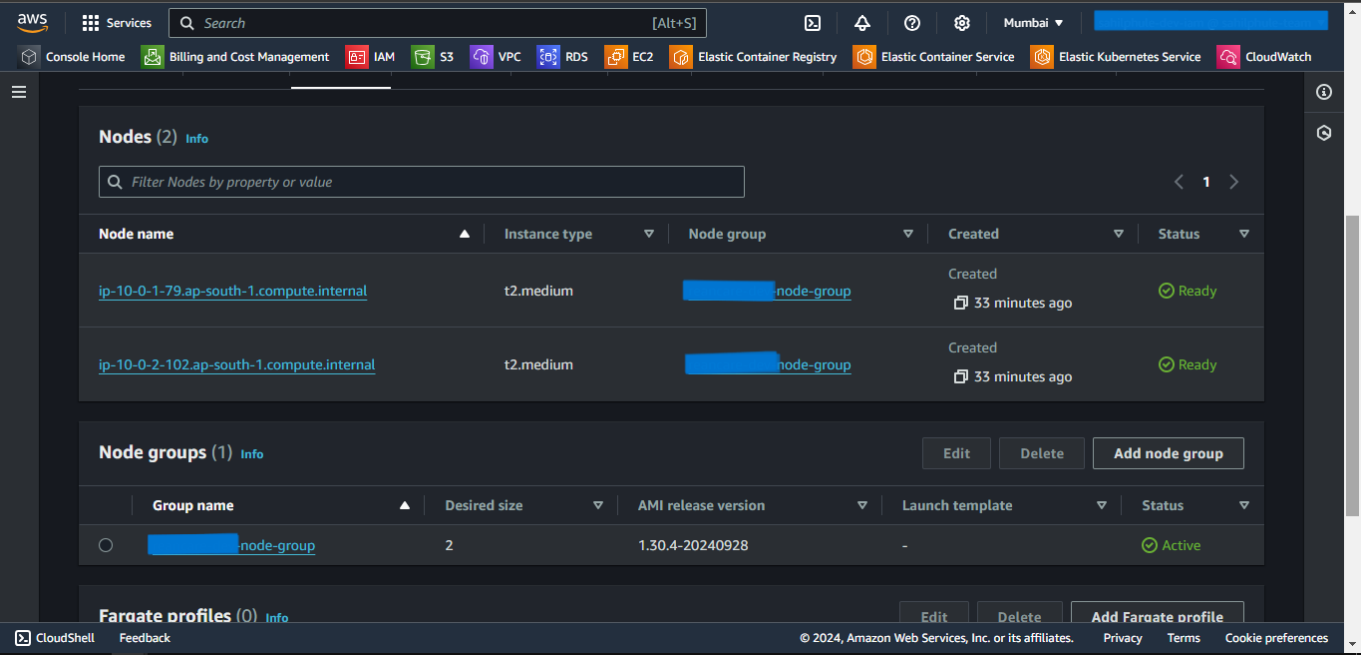
## RDS Image



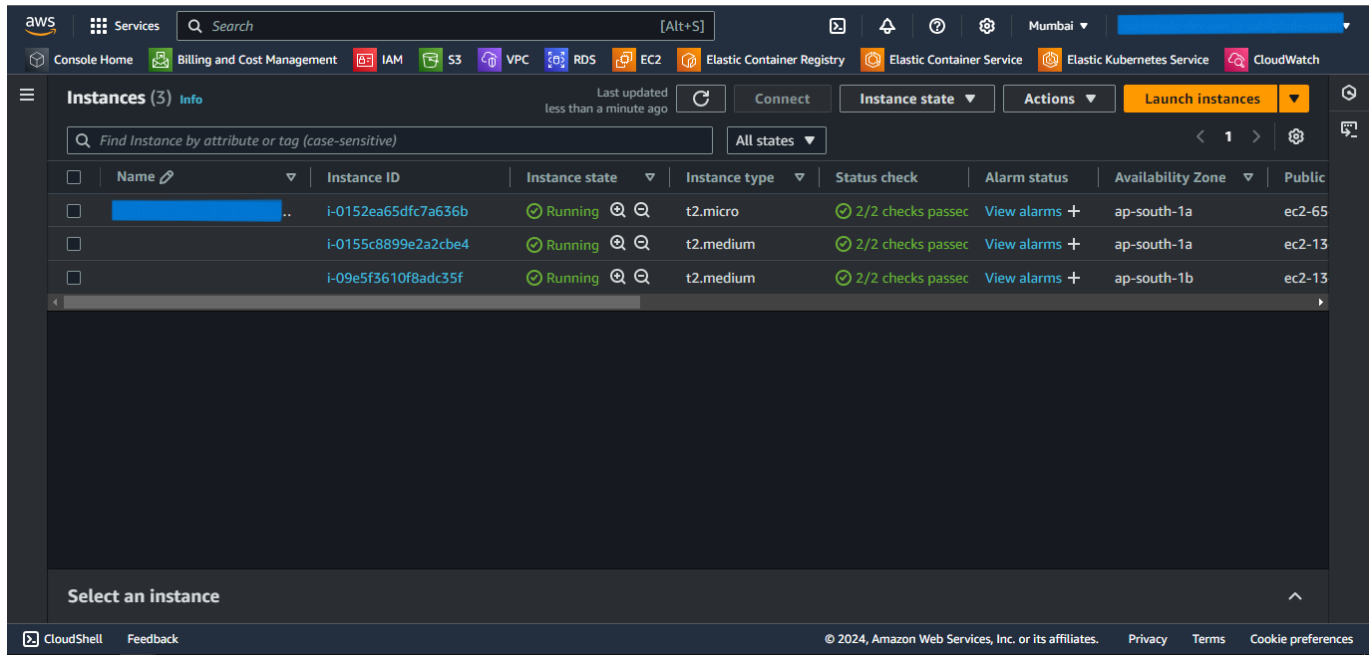
EKS Cluster Image



EKS Node Group Image



## EKS Nodes Image



## Connect to the EKS Cluster from Powershell

### Steps

1. Open a new Powershell window.
2. Run the following command to configure local kubectl with eks cluster:

```
aws eks --region <region-name> update-kubeconfig --name <cluster-name>
```

Substitute *<region-name>* and *<cluster-name>* with the values defined in the above-created *locals.tf* file.

3. Now apply the Kubernetes manifest files of the application using the following command:

```
kubectl apply -f <file-path>
```

Substitute *<file-path>* with the Kubernetes manifest file path.

4. To list them all, run **kubectl get all**.
5. If a Load Balancer type Service is present then try accessing the External IP of that service in the browser.

---

## Connection to the RDS database through Bastion Host using MySQL Workbench

---

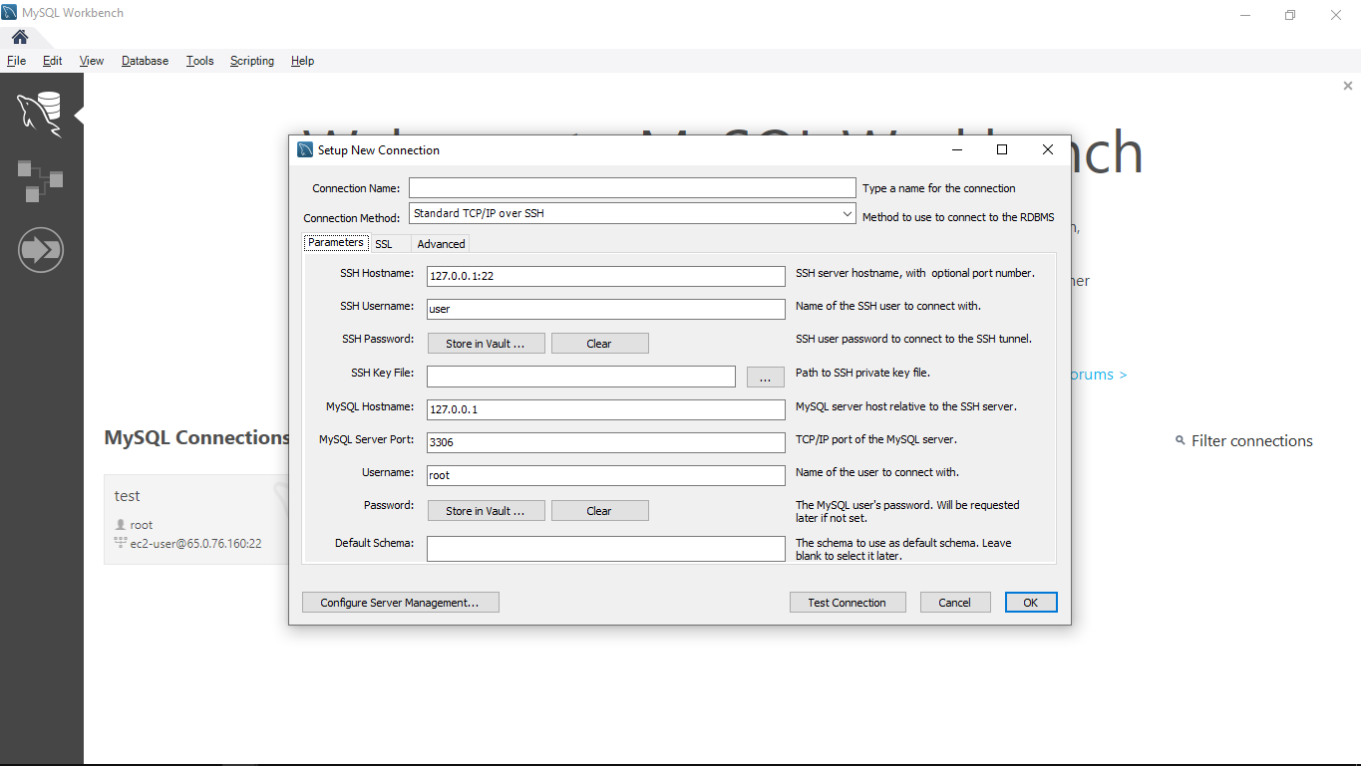
Now, we will use MySQL Workbench to connect and access the MySQL RDS Database through above created Bastion Host.

### Steps

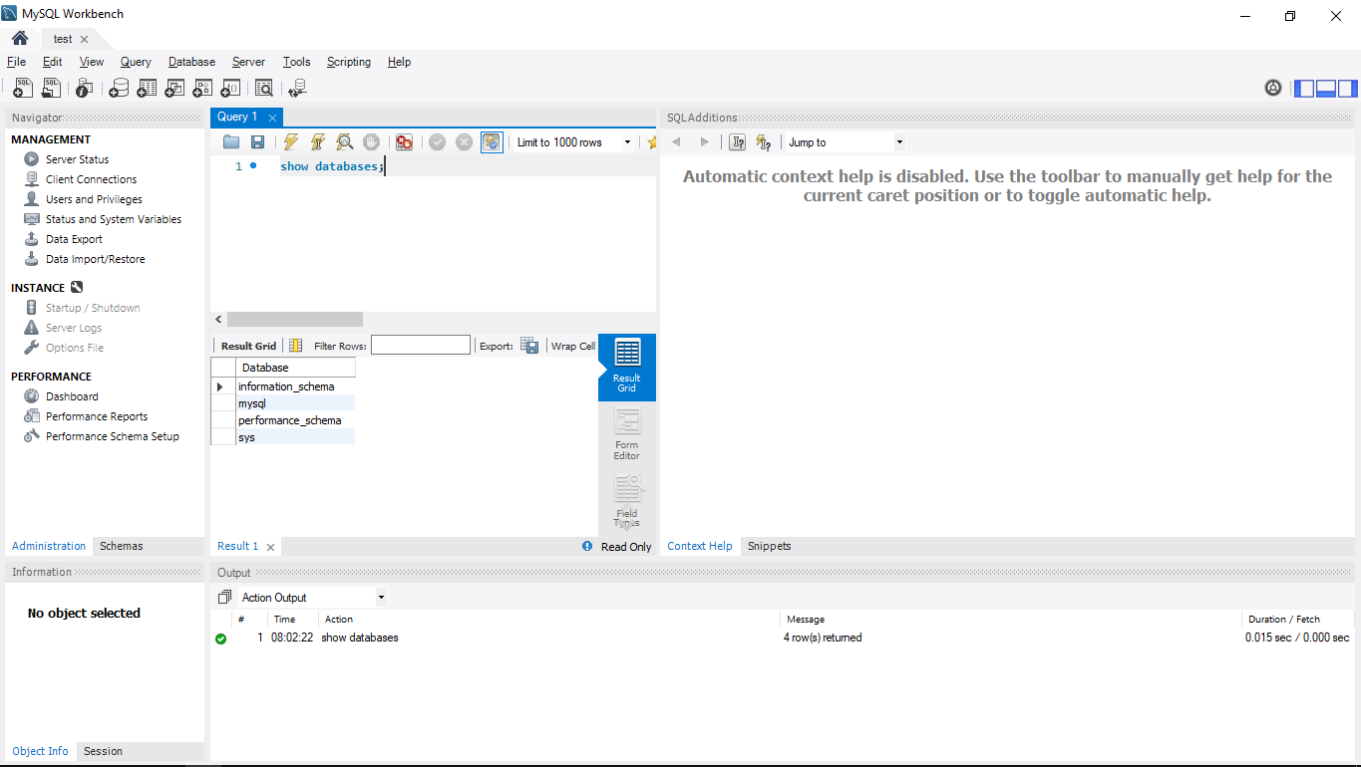
1. Open MySQL Workbench.
  2. Click Add Connection.
  3. Select connection method as **Standard TCP/IP over SSH**.
  4. In SSH Hostname, enter *bastion-host-ip:22* where bastion-host-ip is received from the **terraform output** command.
  5. In SSH Username, enter *ec2-user*.
  6. In SSH Key File, select *bastion-key.pem* file passed in above *locals.tf* file from your local computer.
  7. In MySQL Hostname, enter *DB\_HOST* where DB\_HOST is received from the **terraform output** command.
  8. In the Password section, select *Store in Vault*, and enter the password passed in above-created *locals.tf* file.
  9. Click *OK* and open the connection.
  10. Now you can run MySQL commands to access databases and verify the successful connection of *eks-nodes*.
-

# Screenshots of MySQL Workbench

## Connection Page



## Commands Page



---

## Destroy the provisioned infrastructure

---

Lastly, we will destroy the above-created resources.

### Steps

1. Firstly, delete all the Kubernetes Deployments using:

```
kubectl delete -f <file-path>
```

Substitute *<file-path>* with the Kubernetes manifest file path.

2. To destroy infrastructure, change the directory to the above-created **eks-terraform** directory using the **cd** command.
  3. Run **terraform destroy** & if prompted, type **yes**.
  4. Infrastructure will be destroyed.
-