**Lab Steps**

Task 1: Sign in to AWS Management Console

1. Click on the button, and you will get redirected to AWS Console in a new browser tab.
2. On the AWS sign-in page,

* Leave the Account ID as default. Never edit/remove the 12 digit Account ID present in the AWS Console. otherwise, you cannot proceed with the lab.
* Now copy your **User Name** and **Password** in the Lab Console to the **IAM Username and Password** in AWS Console and click on the **Sign in** button

     3. Once Signed In to the AWS Management Console, Make the default AWS Region as **US East (N. Virginia) us-east-1.**

Task 2: Setup Visual Studio Code

1. Open the visual studio code.
2. If you have already installed and using Visual studio code, open a new window.
3. A new window will open a new file and release notes page (only if you have installed or updated Visual Studio Code recently). Close the Release notes tab.
4. Open Terminal by selecting View from the Menu bar and choose Terminal.
5. It may take up to 2 minutes to open the terminal window.
6. Once the terminal is ready, let us navigate to the Desktop.

cd Desktop

1. Create a new folder by running the below command.

mkdir task\_10103

1. Change your present working directory to use the newly created folder by running the below command:

cd task\_10103

1. Get the location of the present working directory by running the below command:

pwd

1. Note down the location, as you will open the same in the next steps.
2. Now click on the first icon Explorer present on the left sidebar.
3. Click on the button called Open folder and navigate to the location of folder **task\_10103**.

Graphical user interface, application

Description automatically generated

1. (Optional) Click on Authorize button for allowing Visual Studio Code to use the task\_10103 folder. This will only be asked when you have been using Visual Studio code for a while as you are allowing a new folder to be accessed by VSC.
2. Visual Studio Code is now ready to use.

Task 3: Create a variable file

In this task, you will create variable files where you will declare all the global variables with a short description and a default value.

1. To create a variable file, expand the folder **task\_10103** and click on the **New** **File** icon to add the file.
2. Name the file as **variables.tf** and press **Enter** to save it.
3. **Note:** Don't change the location of the new file, keep it default, i.e. inside the **task\_10103** folder**.**
4. Paste the below contents in **variables.tf** file.

|  |
| --- |
| variable "access\_key" {      description = "Access key to AWS console"  }  variable "secret\_key" {      description = "Secret key to AWS console"  }  variable "region" {      description = "AWS region"  } |

1. In the above content, you are declaring a variable called, access\_key, secret\_key, and region with a short description of all 3.
2. After pasting the above contents, save the file by pressing **ctrl + S**.
3. Now expand the folder **task\_10103** and click on the **New File** icon to add the file.
4. Name the file as **terraform.tfvars** and press **Enter** to save it.
5. Paste the below content into the **terraform.tfvars** file.

|  |
| --- |
| region = "us-east-1"  access\_key = "<YOUR AWS CONSOLE ACCESS ID>"  secret\_key = "<YOUR AWS CONSOLE SECRET KEY>" |

1. In the above code, you are defining the dynamic values of variables declared earlier.
2. Replace the values of access\_key and secret\_key by copying from the lab page.
3. After replacing the values of access\_key and secret\_key, save the file by pressing Ctrl + S.

Graphical user interface

Description automatically generated

Task 4: Create a Security group and key pair for EC2 in main.tf file

 In this task, you will create a **main.tf** file where you will add details of the provider and resources.

1. To create a **main.tf** file, expand the folder **task\_10103** and click on the **New** **File** icon to add the file.
2. Name the file as **main.tf** and press **Enter** to save it.
3. Paste the below content into the **main.tf** file.

|  |
| --- |
| provider "aws" {      region     = "${var.region}"      access\_key = "${var.access\_key}"      secret\_key = "${var.secret\_key}"  } |

1. In the above code, you are defining the provider as aws.
2. Next, we want to tell Terraform to create a Security group and Key pair for EC2 Instance
3. To create a security group and key pair Paste the below content into the **main.tf** file after the provider

|  |
| --- |
| ############ Creating Security Group for EC2 ############  resource "aws\_security\_group" "web-server" {      name        = "MyEc2server-SG"      description = "Security for ec2 server to connect with RDS"      ingress {          from\_port   = 22          to\_port     = 22          protocol    = "tcp"          cidr\_blocks = ["0.0.0.0/0"]      }      egress {          from\_port   = 0          to\_port     = 0          protocol    = "-1"          cidr\_blocks = ["0.0.0.0/0"]      }  }  ############ Creating Key pair for EC2 ############  resource "tls\_private\_key" "example" {    algorithm = "RSA"    rsa\_bits  = 4096  }  resource "aws\_key\_pair" "whiz\_key" {    key\_name   = "WhizKey"    public\_key = tls\_private\_key.example.public\_key\_openssh  } |

Text

Description automatically generated

1. Save the file by pressing **Ctrl + S**.

Task 5: Launch an EC2 Instance in main.tf file

1. To Launch an EC2 Instance add another block of code just below the key pair code into the **main.tf** file

|  |
| --- |
| ################## Launching EC2 Instance ##################  resource "aws\_instance" "web-server" {      ami             = "ami-01cc34ab2709337aa"      instance\_type   = "t2.micro"      key\_name        = aws\_key\_pair.whiz\_key.key\_name      security\_groups = ["${aws\_security\_group.web-server.name}"]      user\_data = <<-EOF  #!/bin/bash -ex  yum install mysql -y      EOF      tags = {          Name = "MyRdsEc2server"      }  } |

Text

Description automatically generated

1. Save the file by pressing **Ctrl + S**.

Task 6: Create a Security group for RDS in main.tf file

 In this task, you will create a security group for RDS database in main.tf file

1. To create a security group add another block of code just below the EC2 code into the **main.tf** file

|  |
| --- |
| ############ Creating Security Group for RDS ############  resource "aws\_security\_group" "rds-server" {      name        = "rds-maz-SG-SG"      description = "Security group for RDS Aurora"      ingress {          from\_port   = 3306          to\_port     = 3306          protocol    = "tcp"          cidr\_blocks = ["0.0.0.0/0"]      }      egress {          from\_port   = 0          to\_port     = 0          protocol    = "-1"          cidr\_blocks = ["0.0.0.0/0"]      }  } |

Text

Description automatically generated

1. Save the file by pressing **Ctrl + S**.

Task 7: Create RDS Database DB Cluster and Instance in main.tf file

In this task, you will create an Amazon Aurora database with Multi-AZ enabled

1. To create a RDS Database DB Cluster and Instance add another block of code just below the Security group into the **main.tf** file

|  |
| --- |
| ############ Creating Amazon Aurora Cluster ############  resource "aws\_rds\_cluster" "aurorards" {    cluster\_identifier      = "myauroracluster"    engine                  = "aurora-mysql"    database\_name           = "MyDB"    master\_username         = "WhizlabsAdmin"    availability\_zones = ["us-east-1a", "us-east-1b", "us-east-1c"]    master\_password         = "Whizlabs123"    vpc\_security\_group\_ids = [aws\_security\_group.rds-server.id]    storage\_encrypted = false    skip\_final\_snapshot   = true  }  ############ Launching Amazon Aurora DB Instance ############  resource "aws\_rds\_cluster\_instance" "cluster\_instances" {    count = 2    identifier         = "muaurorainstance${count.index}"    cluster\_identifier = aws\_rds\_cluster.aurorards.id    publicly\_accessible = true    instance\_class     = "db.t3.small"    engine             = aws\_rds\_cluster.aurorards.engine    engine\_version =      aws\_rds\_cluster.aurorards.engine\_version    } |

1. Save the file by pressing **Ctrl + S**.

Task 8: Create an Output file

In this task, you will create an **output.tf** file where you will add details of the provider and resources.

1. To create an **output.tf** file, expand the folder **task\_10103** and click on the **New** **File** icon to add the file.
2. Name the file as **output.tf** and press **Enter** to save it.
3. Paste the below content into the **output.tf** file.

|  |
| --- |
| output "instance\_id" {    description = "ID of the EC2 instance"    value       = aws\_instance.web-server.id  }  output "cluster\_endpoint" {    value       = aws\_rds\_cluster.aurorards.endpoint  } |

1. In the above code, we will extract details of resources created to confirm that they are created.

Task 9: Confirm the installation of Terraform by checking the version

1. In the Visual Studio Code, open Terminal by selecting **View** from the Menu bar and choose **Terminal**.
2. If you are not in the newly created folder change your present working directory by running the below command.

cd task\_10103

1. To confirm the installation of Terraform, run the below command to check the version:

terraform version

1. If you are getting output as command not found: terraform, this means that terraform is not installed on your system, To install terraform follow the official guide link provided in the Prerequisite section above.

Task 10: Apply terraform configurations

1. Initialize Terraform by running the below command,

terraform init

**Text

Description automatically generated**

**Note:** terraform init will check for all the plugin dependencies and download them if required, this will be used for creating a deployment plan

1. To generate the action plans run the below command,

terraform plan

Text

Description automatically generated

1. To create all the resources declared in main.tf configuration file, run the below command:

terraform apply

1. Approve the creation of all the resources by entering **yes**.
2. It may take up to 15-20 minutes for the terraform apply command to create the resources.

A screenshot of a computer

Description automatically generated with medium confidence

1. Id’s of all the resources created by terraform will be visible there.

Task 11: Check the resources in AWS Console

1. Make sure you are in the **US East (N. Virginia) us-east-1** Region.
2. Navigate to **EC2** by clicking on **Services** on the top, then click on **EC2** in the **Compute** section.
3. Click on the **Instances** on the left navigation panel. You can see the instance created successfully.
4. To Check RDS Creation Select **RDS** under **Database** section.
5. Click on the **Database** on the left navigation panel. You can see that the Database is created successfully.

Graphical user interface, text, application, email

Description automatically generated

1. Click on the **RDS cluster name** and then navigate to **Connectivity & security** to find the endpoint of your **Master(Writer) and Reader** instances, with which you can connect to your DB instance.
2. Copy the Endpoints for later use

Graphical user interface, text, application, email

Description automatically generated

Task 12: Execute Database Operations via SSH

1. Navigate to the EC2 Dashboard and select the **MyRdsEc2Server**
2. Click on button to SSH into the EC2 instance
3. In Connect to Instance page select EC2 Instance Connect tab and click on **Connect** button

Graphical user interface, text, application, email

Description automatically generated

1. Switch to the root user using the command :

sudo -s

1. Log into the RDS instance using the below command:

mysql -h myauroracluster.cluster-cuglym6vfuml.us-east-1.rds.amazonaws.com -u WhizlabsAdmin -p

* **Syntax: mysql -h** **<Hostname>** **-u <username>** **-p**

**Note:** Make sure to change the above **Master(Writer)Cluster endpoint** and **Username** with yours.

* Host name : **(Master(Writer)cluster endpoint)**
* Username : Enter ***WhizlabsAdmin***
* Password :Enter ***Whizlabs123***
* You should now be able to log into the database, as shown below:

Text

Description automatically generated

1. List all Databases:

Show databases;

1. Now you will see the database **MyDB** created while launching the RDS cluster.

Text

Description automatically generated

1. Now create the database in the **Master(Writer) RDS** as given in the screenshot. We'll create a demo database named **auroro\_db.**

|  |
| --- |
| Create database auroro\_db; |

1. Select the newly-created database:

|  |
| --- |
| use  auroro\_db; |

**Text

Description automatically generated**

1. Next we'll create a table named **students** and insert few rows of data using list of commands:

|  |
| --- |
| CREATE TABLE students    (       subject\_id   INT auto\_increment,       subject\_name VARCHAR(255) NOT NULL,       teacher      VARCHAR(255),       start\_date   DATE,       lesson       TEXT,       PRIMARY KEY (subject\_id)    ); |

1. Insert data into the table:

INSERT INTO students(subject\_name, teacher) VALUES ('English', 'John Taylor');

INSERT INTO students(subject\_name, teacher) VALUES ('Science', 'Mary Smith');

INSERT INTO students(subject\_name, teacher) VALUES ('Maths', 'Ted Miller');

INSERT INTO students(subject\_name, teacher) VALUES ('Arts', 'Suzan Carpenter');

**Text

Description automatically generated with low confidence**

1. Now you can view the contents of the table student using the below command:

select \* from students;

**A screenshot of a computer

Description automatically generated with medium confidence**

1. Exit from mysql console use the below command:

exit

Task 13: Delete AWS Resources

1. To delete the resources, open Terminal again.
2. Run the below command to delete all the resources.

terraform destroy

1. Approve the deletion of all the resources by entering **yes**. You can see the **Destroy complete!** message.

**Completion and Conclusion**

* You have successfully set up the Visual Studio Code editor.
* You have successfully created variables.tf and terraform.tfvars files.
* You have successfully Launched an EC2 Instance using the terraform.
* You have successfully created an Amazon Aurora MySQL database with Multi-Az enabled.
* You have successfully created output.tf
* You have successfully executed the terraform configuration commands to create the resources.
* You have successfully checked all the resources created by opening the Console.
* You have successfully Executed Database operations via ssh
* You have successfully deleted all the resources.

**End Lab**