THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

Multi-Region AWS Infrastructure Deployment Using Terraform

Project Description:

This project focuses on automating the deployment of a multi-region cloud infrastructure on **Amazon Web Services (AWS)** using **Terraform**, a powerful Infrastructure as Code (IaC) tool. The goal of this project was to understand and implement the core principles of cloud automation and build a functional and scalable environment that includes compute, storage, networking, and database components.

As part of the implementation, two **EC2 instances** were provisioned in separate regions — us-east-1 and us-east-2 — each using specific Amazon Machine Images (AMIs). The configuration required the use of **multiple provider blocks with aliases**, demonstrating how to manage resources across different regions in a single Terraform project. These instances represent the compute layer of the infrastructure and were configured to serve as general-purpose virtual machines.

To handle cloud-based file storage, a versioned **S3 bucket** was created with proper tagging and configuration. The use of **versioning** ensures that changes to files can be tracked and previous versions can be restored when needed — a critical feature in production-ready cloud environments.

The project also included the creation of a **custom Virtual Private Cloud (VPC)** and **subnet**, simulating a controlled network environment for secure communication between resources. This step demonstrated how to isolate and manage internal traffic within the AWS ecosystem.

Further, a **Relational Database Service (RDS)** instance running **MySQL** was added, highlighting the importance of persistent storage for application data. This involved provisioning a managed database instance with defined parameters such as engine version, storage size, and instance type.

All of these resources were defined in a reusable and readable Terraform configuration. The final result was a consistent, automated, and scalable cloud infrastructure that could be destroyed or recreated with simple CLI commands — greatly reducing manual effort and the chance of human error.

THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

- Install terraform

```
[thisara@localhost ~]$ sudo yum install -y yum-utils
[sudo] password for thisara:
Updating Subscription Management repositories.
Docker CE Stable - x86_64                                   7.9 kB/s | 3.5 kB     00:00
Jenkins                                                     2.7 kB/s | 2.9 kB     00:01
Jenkins                                                      52 kB/s | 126 kB     00:02
Kubernetes                                                  1.5 kB/s | 1.7 kB     00:01
Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)    5.1 kB/s | 4.5 kB     00:00
Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)    2.0 MB/s |  74 MB     00:37
Red Hat Enterprise Linux 8 for x86_64 - BaseOS (RPMs)       5.4 kB/s | 4.1 kB     00:00
Red Hat Enterprise Linux 8 for x86_64 - BaseOS (RPMs)       3.0 MB/s |  96 MB     00:31
Package yum-utils-4.0.21-25.el8.noarch is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[thisara@localhost ~]$ S
```

```
[thisara@localhost ~]$ sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/RHEL/has
hicorp.repo
Updating Subscription Management repositories.
Adding repo from: https://rpm.releases.hashicorp.com/RHEL/hashicorp.repo
[thisara@localhost ~]$
[thisara@localhost ~]$
```

```
[thisara@localhost ~]$ sudo yum -y install terraform
Updating Subscription Management repositories.
Hashicorp Stable - x86_64                                   1.7 MB/s | 1.8 MB     00:01
Last metadata expiration check: 0:00:01 ago on Sun 20 Jul 2025 09:56:55 AM PDT.
Dependencies resolved.
================================================================================
 Package                Architecture        Version                 Repository              Size
================================================================================
Installing:
 terraform              x86_64              1.12.2-1                hashicorp               28 M
```

```
[thisara@localhost ~]$ terraform -v
Terraform v1.12.2
on linux_amd64
[thisara@localhost ~]$
```

- **Create a file terraform demo**

```
[thisara@localhost ~]$ sudo mkdir terraform_demo
[sudo] password for thisara:
[thisara@localhost ~]$
[thisara@localhost ~]$ cd terraform_demo/
[thisara@localhost terraform_demo]$
[thisara@localhost terraform_demo]$
```

THISARA KANDAGE

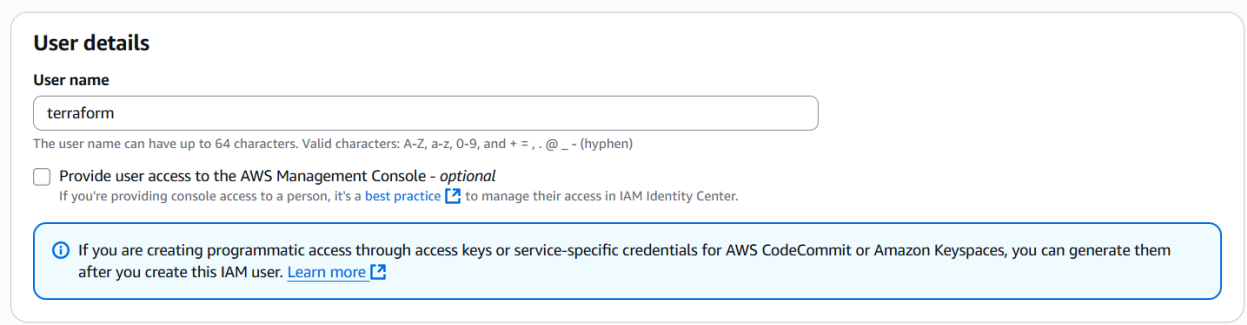
UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

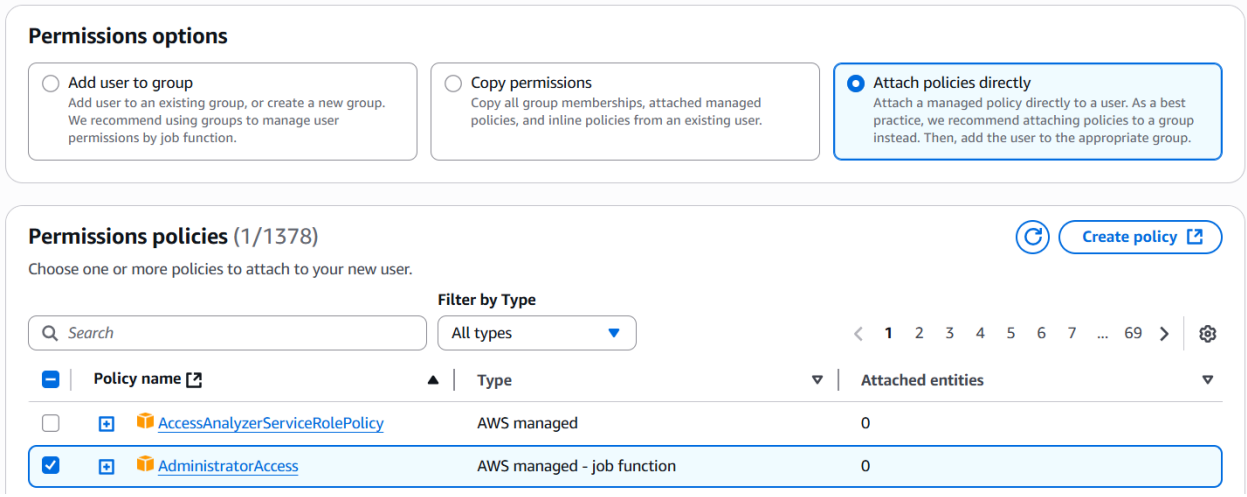
Create a Amazon EC2 instance and destroy it

Steps

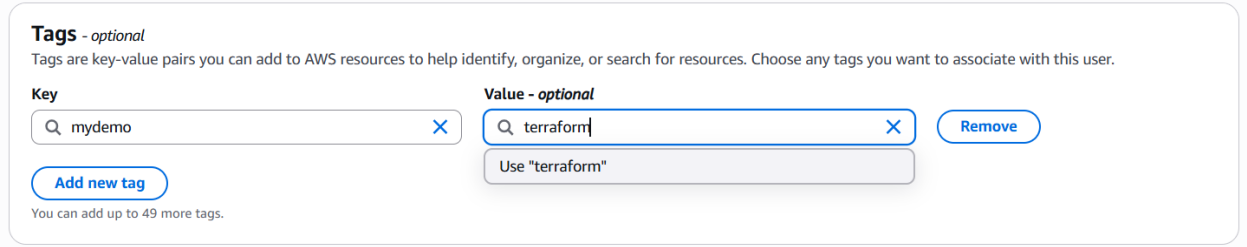
- ### Create AWS IAM user



- ### Set permitions



- ### Add tags

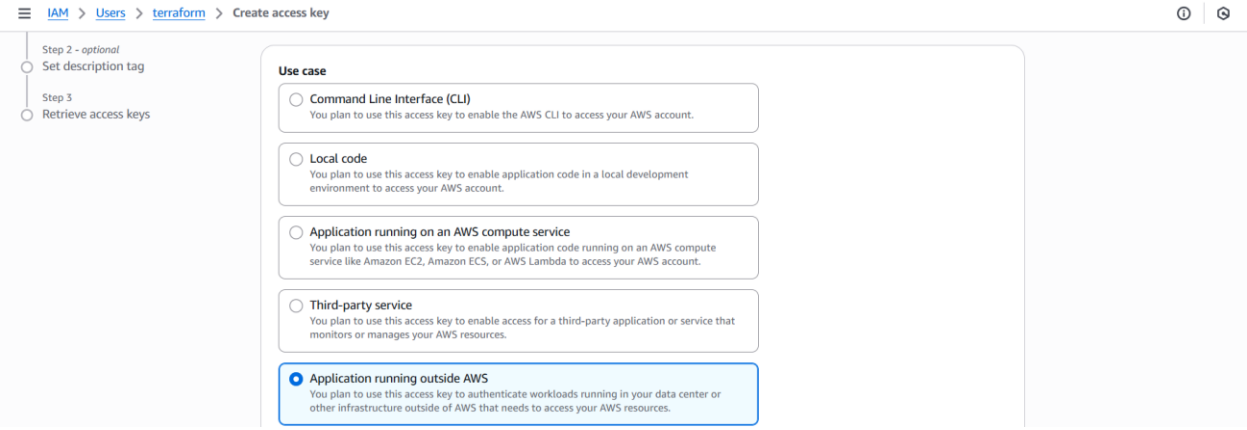


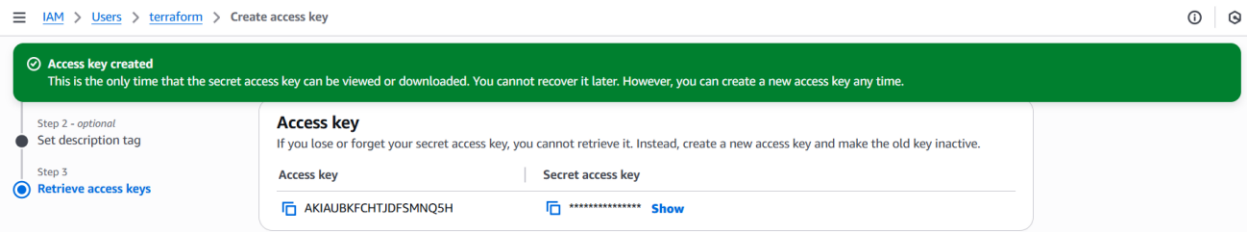
THISARA KANDAGE

UNDERGRADUATE - SLIIT

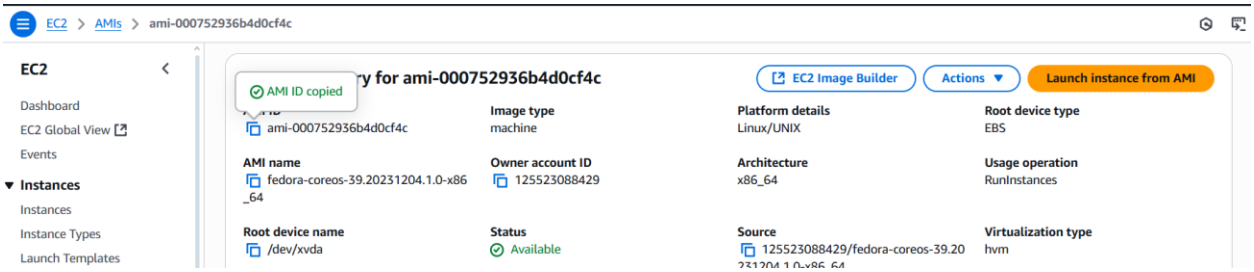
E-mail LinkedIn GitHub Website

- **Create access keys**





- **Find AMI for us-east1**



- **Create file permissions**

```
[thisara@localhost terraform_demo]$ sudo chown -R thisara:thisara .
[thisara@localhost terraform_demo]$
[thisara@localhost terraform_demo]$
```

THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

- **Terraform init**

```
[thisara@localhost terraform_demo]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.4.0...
- Installed hashicorp/aws v6.4.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!
```

- **Terraform plan**

```
[thisara@localhost terraform_demo]$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are
indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.us-east1 will be created
  + resource "aws_instance" "us-east1" {
      + ami                                  = "ami-000752936b4d0cf4c"
      + arn                                  = (known after apply)
```

- **Terraform apply**

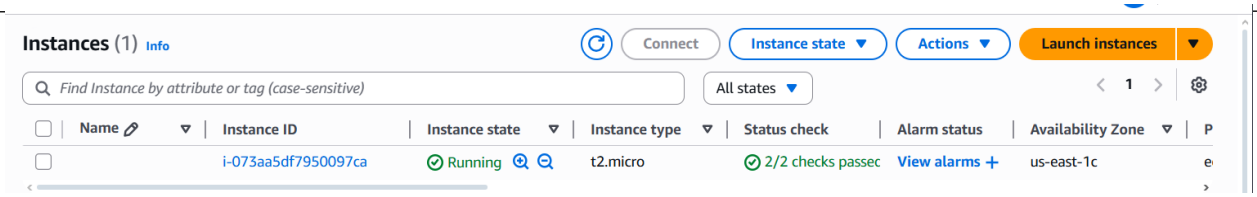
```
[thisara@localhost terraform_demo]$ terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are
indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.us-east1 will be created
  + resource "aws_instance" "us-east1" {
      + ami                                  = "ami-000752936b4d0cf4c"
```

```
  Enter a value: yes

aws_instance.us-east1: Creating...
aws_instance.us-east1: Still creating... [00m10s elapsed]
aws_instance.us-east1: Still creating... [00m20s elapsed]
aws_instance.us-east1: Still creating... [00m30s elapsed]
aws_instance.us-east1: Still creating... [00m40s elapsed]
aws_instance.us-east1: Still creating... [00m50s elapsed]
aws_instance.us-east1: Creation complete after 53s [id=i-073aa5df7950097ca]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[thisara@localhost terraform_demo]$
```

THISARA KANDAGE

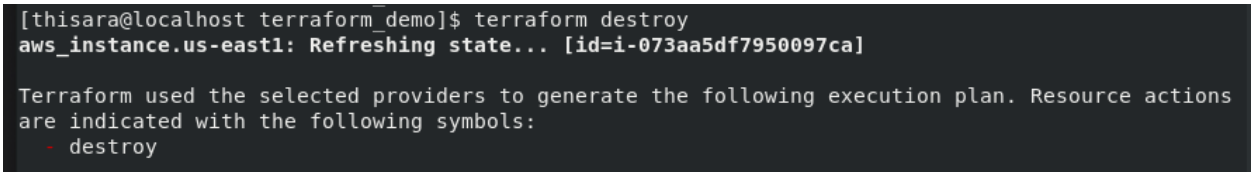
UNDERGRADUATE - SLIIT

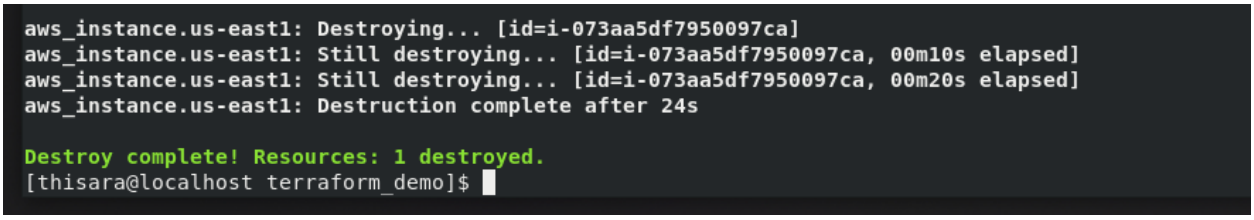
E-mail LinkedIn GitHub Website

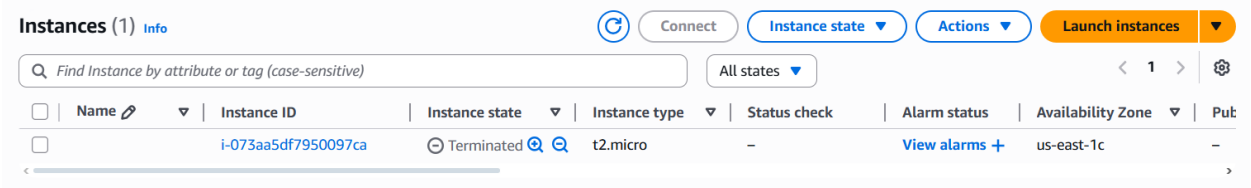


- **Tf file that I used**

- **Destroy the EC2 instance**







THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

- **Add code segment for creating another EC2 instance in Different Region**

```
thisara@localhost:~/terraform_demo
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.8                                  demo1.tf

provider "aws" {
  region     = "us-east-1"
  access_key = "AKIAUBKFCHTJDFSMNQ5H"
  secret_key = "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"
  alias      = "useast1"
}

provider "aws" {
  region     = "us-east-2"
  access_key = "AKIAUBKFCHTJDFSMNQ5H"
  secret_key = "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"
}

resource "aws_instance" "us-east1" {
  ami           = "ami-000752936b4d0cf4c"
  instance_type = "t2.micro"
  provider      = aws.useast1
}

resource "aws_instance" "us-east2" {
  ami           = "ami-0b8b44ec9a8f90422"
  instance_type = "t2.micro"
}
```

- **And also create S3 bucket**

```
resource "aws_s3_bucket" "myfirstbucket" {
  bucket = "thisara-my-tf-test-bucket"

  tags = {
    Name        = "My terraform bucket"
    Environment = "Dev-Env"
  }
}
```

- **Terraform init**

```
[thisara@localhost terraform_demo]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.4.0

Terraform has been successfully initialized!
```

THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

- ## Terraform plan

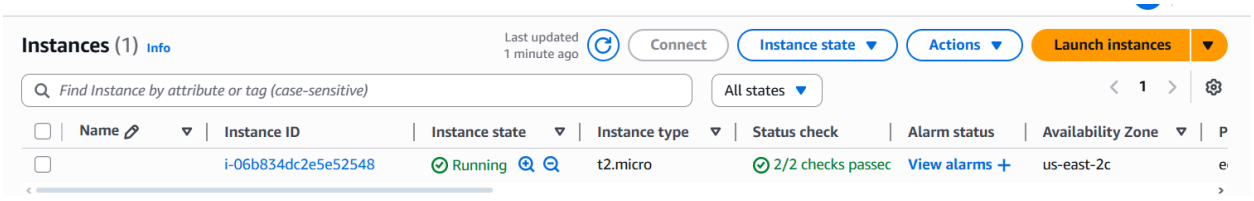
```
[thisara@localhost terraform_demo]$ terraform plan
aws_instance.us-east1: Refreshing state... [id=i-00962138c723bd638]
aws_s3_bucket.myfirstbucket: Refreshing state... [id=thisara-my-tf-test-bucket]
aws_s3_bucket_versioning.myfirstbucket_versioning: Refreshing state... [id=thisara-my-tf-test-bucket]

Terraform used the selected providers to generate the following execution plan. Resource actions are
indicated with the following symbols:
  + create
```

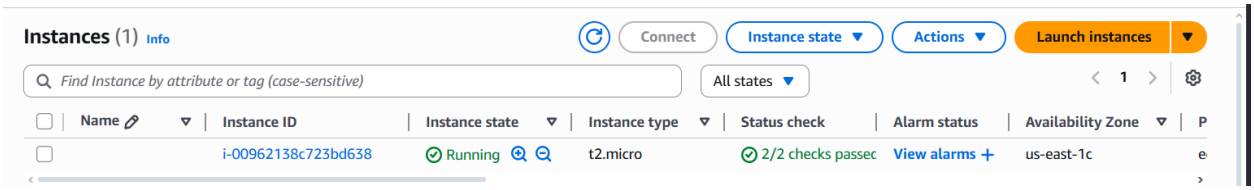
- ## Terraform apply

```
[thisara@localhost terraform_demo]$ terraform apply
aws_instance.us-east1: Refreshing state... [id=i-00962138c723bd638]
aws_s3_bucket.myfirstbucket: Refreshing state... [id=thisara-my-tf-test-bucket]
aws_s3_bucket_versioning.myfirstbucket_versioning: Refreshing state... [id=thisara-my-tf-test-bucket]

Terraform used the selected providers to generate the following execution plan. Resource actions are
indicated with the following symbols:
  + create
```

```
  Enter a value: yes

aws_instance.us-east2: Creating...
aws_instance.us-east2: Still creating... [00m10s elapsed]
aws_instance.us-east2: Still creating... [00m20s elapsed]
aws_instance.us-east2: Still creating... [00m30s elapsed]
aws_instance.us-east2: Still creating... [00m40s elapsed]
aws_instance.us-east2: Creation complete after 42s [id=i-06b834dc2e5e52548]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[thisara@localhost terraform_demo]$
```

- ## Outputs

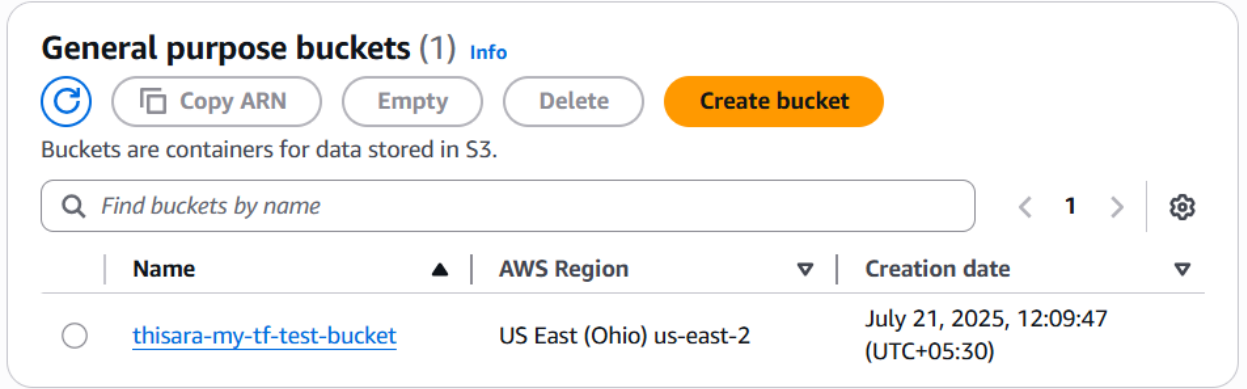




THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website



Add VPC one sub net and also Create relational Database Steps

- **VPC added**

```
resource "aws_vpc" "dev" {
  cidr_block       = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "dev-vpc"
  }
}
```

- **Subnet added**

```
resource "aws_subnet" "sub" {
  vpc_id     = aws_vpc.dev.id
  cidr_block = "10.0.1.0/24"

  tags = {
    Name = "dev-subnet"
  }
}
```

THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

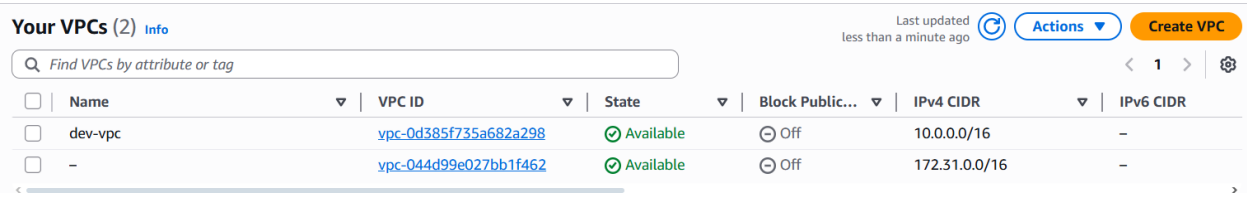
- **RDS added**

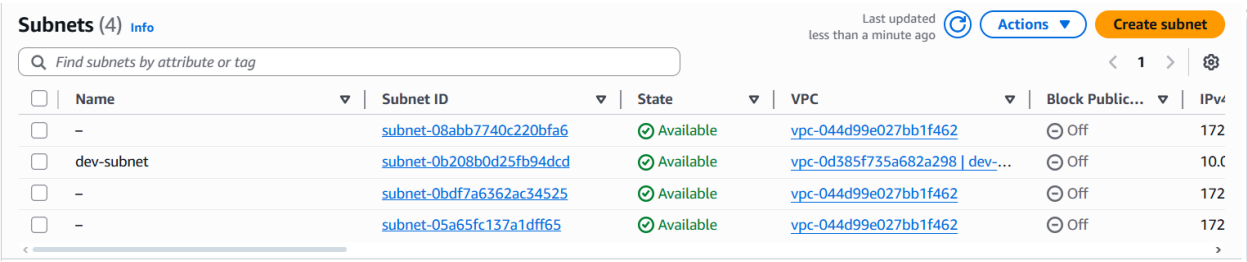
```
resource "aws_db_instance" "projectdb" {
  allocated_storage    = 10
  db_name              = "mydb"
  engine               = "mysql"
  engine_version       = "8.0"
  instance_class       = "db.t3.micro"
  username             = "foo"
  password             = "foobarbaz"
  parameter_group_name = "default.mysql8.0"
  skip_final_snapshot  = true
}
```

- **After init ,plan and apply terraform**

```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[thisara@localhost terraform_demo]$ sudo nano demo1.tf
[sudo] password for thisara:
[thisara@localhost terraform_demo]$
```

- **AWS console proofs**

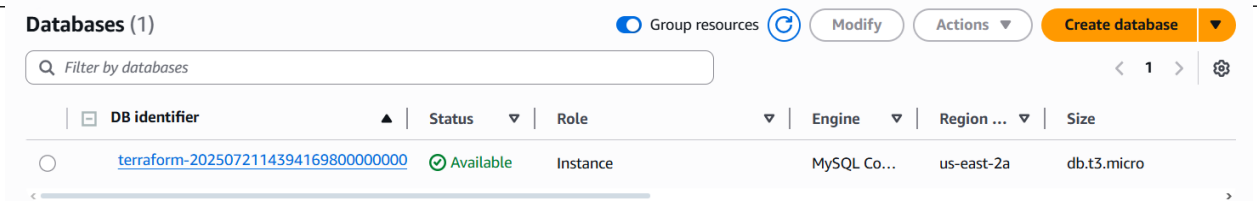




THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website



- **Destroy all things that I created**

```
[thisara@localhost terraform_demo]$ terraform destroy
aws_instance.us-east1: Refreshing state... [id=i-00962138c723bd638]
aws_vpc.dev: Refreshing state... [id=vpc-0d385f735a682a298]
aws_s3_bucket.myfirstbucket: Refreshing state... [id=thisara-my-tf-test-bucket]
aws_instance.us-east2: Refreshing state... [id=i-06b834dc2e5e52548]
aws_db_instance.projectdb: Refreshing state... [id=db-GYZFXR3RTDXCE3XJ53D6MOHMT4]
aws_s3_bucket_versioning.myfirstbucket_versioning: Refreshing state... [id=thisara-my-tf-test-bucket]
aws_subnet.sub: Refreshing state... [id=subnet-0b208b0d25fb94dcd]
```

```
aws_db_instance.projectdb: Destruction complete after 4m7s

Destroy complete! Resources: 7 destroyed.
[thisara@localhost terraform_demo]$
```

Full Terraform file that I used

```
provider "aws" {
 region     = "us-east-1"
 access_key = "AKIAUBKFCHTJDFSMNQ5H"
 secret_key = "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"
 alias      = "useast1"
}

provider "aws" {
 region     = "us-east-2"
 access_key = "AKIAUBKFCHTJDFSMNQ5H"
 secret_key = "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"
```

```
}

resource "aws_instance" "us-east1" {
 ami           = "ami-000752936b4d0cf4c"
 instance_type = "t2.micro"
 provider      = aws.useast1
}

resource "aws_instance" "us-east2" {
 ami           = "ami-0b8b44ec9a8f90422"
 instance_type = "t2.micro"
}

resource "aws_s3_bucket" "myfirstbucket" {
 bucket = "thisara-my-tf-test-bucket"
 tags = {
   Name        = "My terraform bucket"
   Environment = "Dev-Env"
 }
}

resource "aws_s3_bucket_versioning" "myfirstbucket_versioning" {
 bucket = aws_s3_bucket.myfirstbucket.id

 versioning_configuration {
   status = "Enabled"
 }
}
```

THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

```
resource "aws_vpc" "dev" {
  cidr_block       = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "dev-vpc"
  }
}

resource "aws_subnet" "sub" {
  vpc_id     = aws_vpc.dev.id
  cidr_block = "10.0.1.0/24"

  tags = {
    Name = "dev-subnet"
  }
}

resource "aws_db_instance" "projectdb" {
  allocated_storage    = 10
  db_name              = "mydb"
  engine               = "mysql"
  engine_version       = "8.0"
  instance_class       = "db.t3.micro"
  username             = "foo"
  password             = "foobarbaz"
  parameter_group_name = "default.mysql8.0"
  skip_final_snapshot  = true
}
```

THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

Summary of What I Learned:

- **Gained deep practical exposure to Infrastructure as Code (IaC) with Terraform, understanding how to write, manage, and apply .tf files to build real infrastructure.**
- **Learned how to configure and manage multiple AWS providers with region-specific resources, enabling multi-region deployments within a single project.**
- **Developed the ability to provision EC2 instances, define AMI IDs, assign tags, and control instance behavior using Terraform.**
- **Learned how to create and enable versioning for S3 buckets, ensuring safer and more reliable cloud-based storage operations.**
- **Understood the structure and implementation of custom VPCs, subnets, and availability zones, providing experience in building isolated network environments in AWS.**
- **Built hands-on experience provisioning an RDS MySQL instance, defining its specifications, and managing its configuration via Terraform.**
- **Enhanced my problem-solving and debugging skills while handling syntax errors, provider issues, and dependency relationships between resources.**
- **Strengthened my understanding of Terraform commands, resource lifecycle (init, plan, apply, destroy), and state management in infrastructure automation.**