



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.



- **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/hashicorp.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]$
```



Create a Amazon EC2 instance and destroy it

Steps

- **Create AWS IAM user**

User details

User name

terraform

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

☐ Provide user access to the AWS Management Console - *optional*

If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.

📘 If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

- **Set permissions**

Permissions options

☐ Add user to group

Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

☐ Copy permissions

Copy all group memberships, attached managed policies, and inline policies from an existing user.

☒ Attach policies directly

Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1/1378)

Choose one or more policies to attach to your new user.



[Create policy](#)

Search

Filter by Type

All types

< 1 2 3 4 5 6 7 ... 69 > ⚙️

Policy name	Type	Attached entities
<input type="checkbox"/> AccessAnalyzerServiceRolePolicy	AWS managed	0
<input checked="" type="checkbox"/> AdministratorAccess	AWS managed - job function	0

- **Add tags**

Tags - optional

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

Key

mydemo



Value - optional

terraform



[Remove](#)

[Add new tag](#)

Use "terraform"

You can add up to 49 more tags.



THISARA KANDAGE

UNDERGRADUATE - SLIIT

[E-mail](#) [LinkedIn](#) [GitHub](#) [Website](#)

• Create access keys

IAM > Users > terraform > Create access key

Step 2 - optional
Set description tag

Step 3
Retrieve access keys

Use case

- ☐ Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.
- ☐ Local code
You plan to use this access key to enable application code in a local development environment to access your AWS account.
- ☐ Application running on an AWS compute service
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.
- ☐ Third-party service
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.
- ☒ Application running outside AWS
You plan to use this access key to authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.

Access key created
This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

Access key
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key: AKIAUBKFCHTJDFSMNQSH

Secret access key: ***** [Show](#)

• Find AMI for us-east1

EC2 > AMIs > ami-000752936b4d0cf4c

AMI ID copied

AMI details for ami-000752936b4d0cf4c

Image type machine	Platform details Linux/UNIX	Root device type EBS
Owner account ID 125523088429	Architecture x86_64	Usage operation RunInstances
Status Available	Source 125523088429/fedora-coreos-39.20 x86_64	Virtualization type hvm

[EC2 Image Builder](#) [Actions](#) [Launch instance from AMI](#)

• Create file permissions

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



- 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](#)

Instances (1) [Info](#)

[Refresh](#) [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

[All states](#) < 1 > [Settings](#)

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	P
<input type="checkbox"/>		i-073aa5df7950097ca	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1c	e

- Tf file that I used

```
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  = "AKIAUBKFCHTJDFSMN05H"
  secret_key  = "uJ7NR/Nu4x5KRuh3jrlrEu3LJJ0vsc/R4JaezH/A"
}

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

- Destroy the EC2 instance

```
[thisara@localhost terraform_demo]$ terraform destroy
aws_instance.us-east1: Refreshing state... [id=i-073aa5df7950097ca]

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

```
aws_instance.us-east1: Destroying... [id=i-073aa5df7950097ca]
aws_instance.us-east1: Still destroying... [id=i-073aa5df7950097ca, 00m10s elapsed]
aws_instance.us-east1: Still destroying... [id=i-073aa5df7950097ca, 00m20s elapsed]
aws_instance.us-east1: Destruction complete after 24s

Destroy complete! Resources: 1 destroyed.
[thisara@localhost terraform_demo]$
```

Instances (1) [Info](#)

[Refresh](#) [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

[All states](#) < 1 > [Settings](#)

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	P
<input type="checkbox"/>		i-073aa5df7950097ca	Terminated	t2.micro	-	View alarms +	us-east-1c	-



- **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!
```



• 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

Instances (1) Info

Last updated
1 minute ago

Connect

Instance state ▾

Actions ▾

Launch instances ▾

All states ▾

< 1 >

<input type="checkbox"/>	Name	Instance ID	Instance state ▾	Instance type ▾	Status check	Alarm status	Availability Zone ▾	P
<input type="checkbox"/>		i-06b834dc2e5e52548	Running	t2.micro	2/2 checks passed	View alarms +	us-east-2c	e

Instances (1) Info

Connect

Instance state ▾

Actions ▾

Launch instances ▾

All states ▾

< 1 >

<input type="checkbox"/>	Name	Instance ID	Instance state ▾	Instance type ▾	Status check	Alarm status	Availability Zone ▾	P
<input type="checkbox"/>		i-00962138c723bd638	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1c	e



General purpose buckets (1) [Info](#)



Copy ARN

Empty

Delete

Create bucket

Buckets are containers for data stored in S3.

Find buckets by name

< 1 >

	Name	AWS Region	Creation date
<input type="radio"/>	thisara-my-tf-test-bucket	US East (Ohio) us-east-2	July 21, 2025, 12:09:47 (UTC+05:30)

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"
  }
}
```



- **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          = "foobabaz"
  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**

Your VPCs (2) [Info](#)

Last updated less than a minute ago [Actions](#) [Create VPC](#)

Find VPCs by attribute or tag

<input type="checkbox"/>	Name	VPC ID	State	Block Public...	IPv4 CIDR	IPv6 CIDR
<input type="checkbox"/>	dev-vpc	vpc-0d385f735a682a298	Available	Off	10.0.0.0/16	-
<input type="checkbox"/>	-	vpc-044d99e027bb1f462	Available	Off	172.31.0.0/16	-

Subnets (4) [Info](#)

Last updated less than a minute ago [Actions](#) [Create subnet](#)

Find subnets by attribute or tag

<input type="checkbox"/>	Name	Subnet ID	State	VPC	Block Public...	IPv4
<input type="checkbox"/>	-	subnet-08abb7740c220bfa6	Available	vpc-044d99e027bb1f462	Off	172
<input type="checkbox"/>	dev-subnet	subnet-0b208b0d25fb94dcd	Available	vpc-0d385f735a682a298 dev-...	Off	10.0
<input type="checkbox"/>	-	subnet-0bdf7a6362ac34525	Available	vpc-044d99e027bb1f462	Off	172
<input type="checkbox"/>	-	subnet-05a65fc137a1dfff65	Available	vpc-044d99e027bb1f462	Off	172



THISARA KANDAGE

UNDERGRADUATE - SLIIT

[E-mail](#) [LinkedIn](#) [GitHub](#) [Website](#)

Databases (1)

☒ Group resources

Modify

Actions

Create database

Filter by databases

< 1 > ⚙

DB identifier	Status	Role	Engine	Region ...	Size
terraform-2025072114394169800000000	Available	Instance	MySQL Co...	us-east-2a	db.t3.micro

• 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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"
```



THISARA KANDAGE

UNDERGRADUATE - SLIIT

[E-mail](#) [LinkedIn](#) [GitHub](#) [Website](#)

```
}

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  
}
```



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.