Terraform Multi-Region EC2 Deployment Guide

Terraform Multi-Region EC2 Deployment Guide

Objective

Set up a Terraform server on a GCP VM and use it to launch two EC2 instances in different AWS regions (**us-east-1** and **us-west-2**) using a single Terraform project.

1. Launch a Terraform Server on GCP

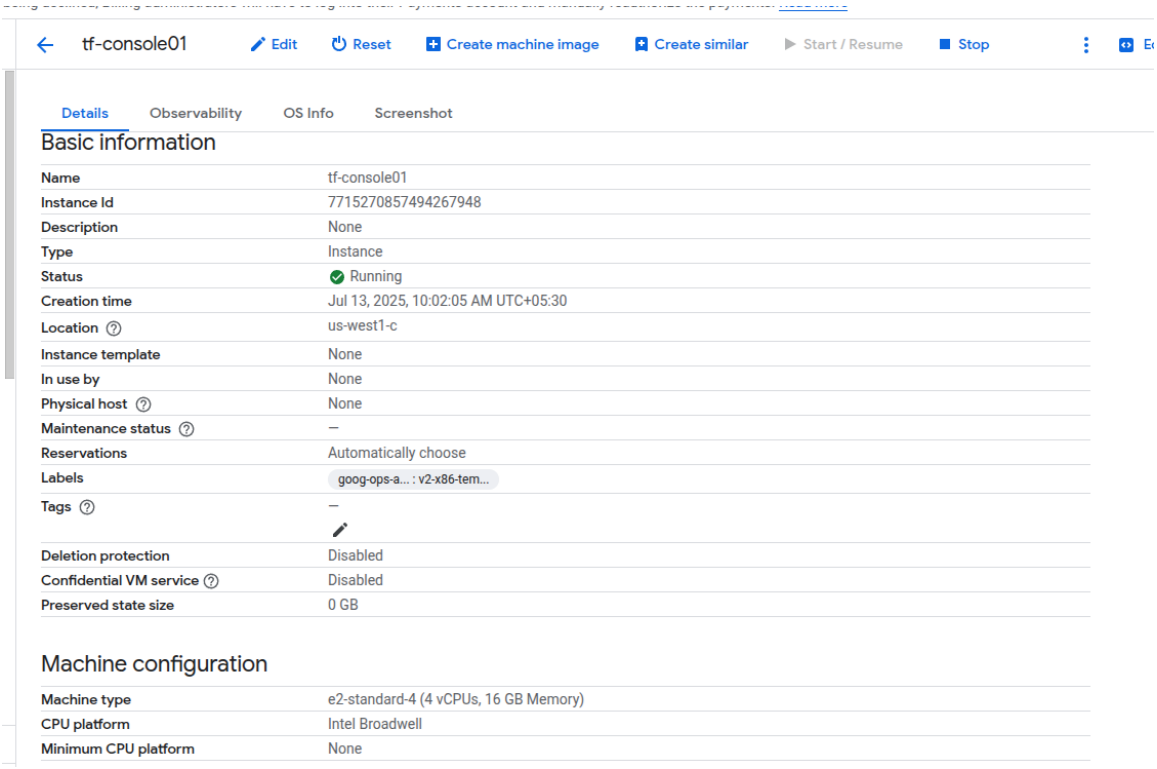
- Create a GCP Compute Engine VM:

  - OS: Ubuntu 22.04 LTS (or Debian-based)

  - Machine Type: 4 vcpu with 16 gb memory, and 30gb of storage

  - Firewall: Allow HTTP, HTTPS, and SSH

  - Ensure you can SSH into it



2. Install Terraform and AWS CLI on the GCP VM

2.1 Install Terraform

- Loginto the server either using putty or VS code via SSH. And follow the below steps to install terraform on the server.

sudo apt update && sudo apt install -y gnupg software-properties-common curl

curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg

echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list

sudo apt update && sudo apt install terraform -y

```
ubuntu@tf-console01:~/multi-region-ec2$ tf --version
Terraform v1.12.2
on linux_amd64
```

2.2 Install AWS CLI v2

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

2.3 Verify Installations

terraform -version

aws --version

```
ubuntu@tf-console01:~/multi-region-ec2$ aws --version
aws-cli/2.27.50 Python/3.13.4 Linux/6.8.0-1032-gcp exe/x86_64.ubuntu.22
ubuntu@tf-console01:~/multi-region-ec2$
```

3. Configure AWS CLI

Generate an AWS Access Key and Secret Key from the AWS console under IAM > Users.

aws configure

Enter Access Key, Secret Key

Default region: us-east-1

Output format: json

4. Setup Terraform Project

4.1 Create Project Directory

mkdir ~/multi-region-ec2 && cd ~/multi-region-ec2

Create the following files:

- `main.tf`

- `providers.tf`

- `variables.tf`

- `outputs.tf`

4.2 providers.tf

```
provider "aws" {

  region = var.primary_region

}


provider "aws" {

  alias  = "secondary"

  region = var.secondary_region

}
```

4.3 variables.tf

```
variable "primary_region" {

  default = "us-east-1"

}


variable "secondary_region" {

  default = "us-west-2"

}


variable "instance_type" {

  default = "t2.micro"

}


variable "ami" {

  description = "Amazon Linux 2 AMI for each region"

  type = map(string)

  default = {
```

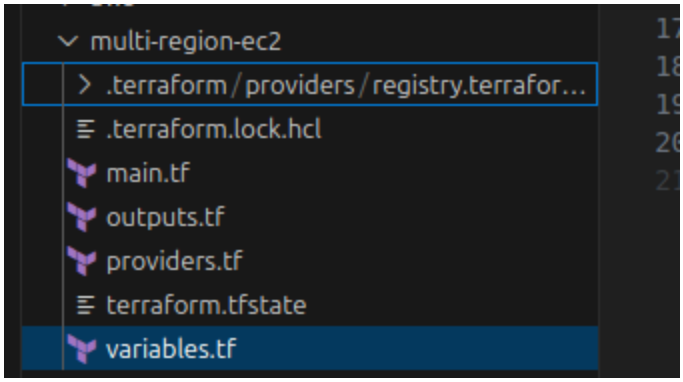
```
    us-east-1 = "ami-0c02fb55956c7d316"
    us-west-2 = "ami-0e34e7b9ca0ace12d"
  }
}
```

4.4 main.tf

```
resource "aws_instance" "ec2_primary" {
  provider      = aws
  ami           = var.ami[var.primary_region]
  instance_type = var.instance_type

  tags = {
    Name = "EC2-Primary"
  }
}

resource "aws_instance" "ec2_secondary" {
  provider      = aws.secondary
  ami           = var.ami[var.secondary_region]
  instance_type = var.instance_type

  tags = {
    Name = "EC2-Secondary"
  }
}
```

4.5 outputs.tf

```
output "primary_instance_ip" {

  value = aws_instance.ec2_primary.public_ip

}



output "secondary_instance_ip" {

  value = aws_instance.ec2_secondary.public_ip

}
```



5. Run Terraform Commands

5.1 Initialize Terraform

terraform init

```
ubuntu@tf-console01:~/multi-region-ec2$ tf init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.3.0...
- Installed hashicorp/aws v6.3.0 (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!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

5.2 Check the Plan

terraform plan

```
ubuntu@tf-console01:~/multi-region-ec2$ tf plan

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

Terraform will perform the following actions:

  # aws_instance.ec2_primary will be created
  + resource "aws_instance" "ec2_primary" {
      + ami                                  = "ami-0c02fb55956c7d316"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
```

```
      + tenancy                              = (known after apply)
      + user_data_base64                     = (known after apply)
      + user_data_replace_on_change          = false
      + vpc_security_group_ids               = (known after apply)

      + capacity_reservation_specification (known after apply)

      + cpu_options (known after apply)

      + ebs_block_device (known after apply)

      + enclave_options (known after apply)

      + ephemeral_block_device (known after apply)

      + instance_market_options (known after apply)

      + maintenance_options (known after apply)

      + metadata_options (known after apply)

      + network_interface (known after apply)

      + private_dns_name_options (known after apply)

      + root_block_device (known after apply)
    }

Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + primary_instance_ip   = (known after apply)
  + secondary_instance_ip = (known after apply)
```

5.3 Apply the Plan

terraform apply

Type 'yes' to confirm

```
ubuntu@tf-console01:~/multi-region-ec2$ tf apply

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

Terraform will perform the following actions:

  # aws_instance.ec2_primary will be created
  + resource "aws_instance" "ec2_primary" {
      + ami                                  = "ami-0c02fb55956c7d316"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
```

```
Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + primary_instance_ip   = (known after apply)
  + secondary_instance_ip = (known after apply)

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```

5.4 View Outputs

You'll see:

Outputs:

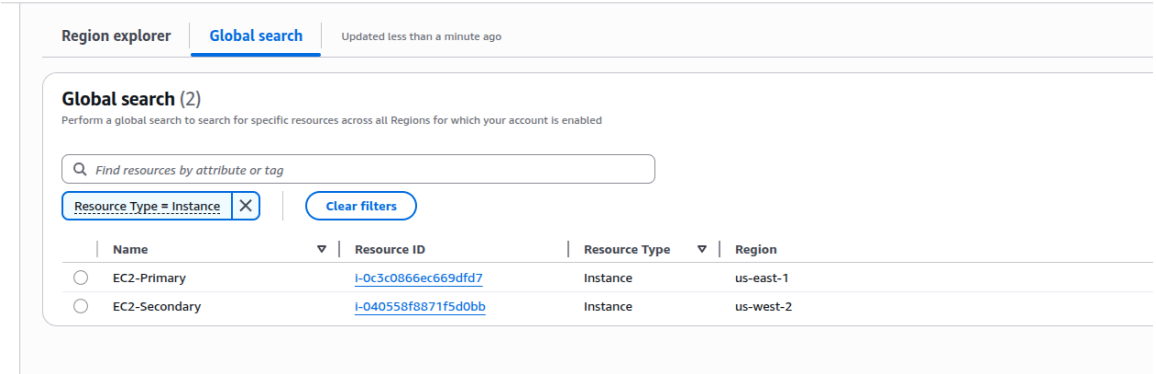
primary_instance_ip = "3.92.x.x"

secondary_instance_ip = "34.221.x.x"

```
aws_instance.ec2_primary: Creating...
aws_instance.ec2_secondary: Creating...
aws_instance.ec2_primary: Still creating... [00m10s elapsed]
aws_instance.ec2_secondary: Still creating... [00m10s elapsed]
aws_instance.ec2_primary: Still creating... [00m20s elapsed]
aws_instance.ec2_secondary: Still creating... [00m20s elapsed]
aws_instance.ec2_primary: Still creating... [00m30s elapsed]
aws_instance.ec2_secondary: Still creating... [00m30s elapsed]
aws_instance.ec2_secondary: Creation complete after 32s [id=i-040558f8871f5d0bb]
aws_instance.ec2_primary: Creation complete after 33s [id=i-0c3c0866ec669dfd7]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

primary_instance_ip = "44.203.183.117"
secondary_instance_ip = "52.41.252.199"
```

6. Verify EC2 Instances

Use AWS Console or CLI to check instances in both regions.



7. Destroy Resources

terraform destroy

Type 'yes' to confirm

```
ubuntu@tf-console01:~/multi-region-ec2$ tf destroy
aws_instance.ec2_primary: Refreshing state... [id=i-0c3c0866ec669dfd7]
aws_instance.ec2_secondary: Refreshing state... [id=i-040558f8871f5d0bb]

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

Terraform will perform the following actions:

  # aws_instance.ec2_primary will be destroyed
  - resource "aws_instance" "ec2_primary" {
      - ami                                  = "ami-0c02fb55956c7d316" -> null
      - arn                                  = "arn:aws:ec2:us-east-1:391070786986:instance/i
0866ec669dfd7" -> null
      - associate_public_ip_address          = true -> null
```

```
  Enter a value: yes

aws_instance.ec2_secondary: Destroying... [id=i-040558f8871f5d0bb]
aws_instance.ec2_primary: Destroying... [id=i-0c3c0866ec669dfd7]
aws_instance.ec2_secondary: Still destroying... [id=i-040558f8871f5d0bb, 00m10s elapsed]
aws_instance.ec2_primary: Still destroying... [id=i-0c3c0866ec669dfd7, 00m10s elapsed]
aws_instance.ec2_secondary: Still destroying... [id=i-040558f8871f5d0bb, 00m20s elapsed]
aws_instance.ec2_primary: Still destroying... [id=i-0c3c0866ec669dfd7, 00m20s elapsed]
aws_instance.ec2_secondary: Still destroying... [id=i-040558f8871f5d0bb, 00m30s elapsed]
aws_instance.ec2_secondary: Destruction complete after 30s
aws_instance.ec2_primary: Still destroying... [id=i-0c3c0866ec669dfd7, 00m30s elapsed]
aws_instance.ec2_primary: Still destroying... [id=i-0c3c0866ec669dfd7, 00m40s elapsed]
aws_instance.ec2_primary: Destruction complete after 41s

Destroy complete! Resources: 2 destroyed.
```

Notes

- Make sure you use the correct and updated AMI IDs for Amazon Linux 2 in each region.

- This setup uses only **local state**, no locking, no S3/DynamoDB.

- Everything runs from your GCP VM (Terraform server).