

Terraform AWS EC2 Provisioning

This Terraform configuration provisions multiple EC2 instances in AWS using dynamic AMI lookup and tagging based on user-defined input variables. It's structured to be region-aware, scalable, and reusable.

■ Overview

This project demonstrates how to: - Use variables (region, AMI map, and dynamic tags) - Dynamically create EC2 instances based on the number of tag values - Select the appropriate AMI for the region using `lookup` - Use `count` to scale resources based on input - Output essential information like instance IDs and public IPs

■■ Terraform Blocks Explained

`terraform` Block

```
`` terraform { required_providers { aws = { source = "hashicorp/aws" version = "~>6.0" } } } ``
```

- This block specifies the required provider: AWS from HashiCorp. - The version constraint `~>6.0` ensures compatibility with all 6.x versions.

`provider` Block

```
`` provider "aws" { region = var.region } ``
```

- Configures the AWS provider to use a region defined by a variable. - `access_key` and `secret_key` are commented for security and should ideally be handled via environment variables or IAM roles.

■ Input Variables

`region`

```
`` variable "region" { default = "ap-south-1" type = string } ``
```

- Specifies the region where resources will be deployed.

`ami`

```
`` variable "ami" { type = map default = { "us-east-1" = "ami-123", "us-east-2" = "ami-345", "ap-south-1" = "ami-084e7e1456028650e" } } ``
```

- A map of AMIs by region. - Selected dynamically using `lookup(var.ami, var.region)`.

`tags`

```
`` variable "tags" { type = list default = ["first-ec2", "second-ec2", "third-ec2"] } ``
```

- List of tags that determine the number of EC2 instances to launch.

■■ EC2 Resource

```
`` resource "aws_instance" "demo-ec2" { ami = lookup(var.ami, var.region) instance_type = "t2.micro" count = length(var.tags) tags = { "Name" = element(var.tags, count.index) } } ``
```

- Launches one EC2 instance per tag name. - Dynamically selects the correct AMI using region-based lookup. - Tags each instance uniquely with names like `first-ec2`, `second-ec2`, etc.

■ Outputs

`ec2_public_ip`

```
``` output "ec2_public_ip" { value = aws_instance.demo-ec2[*].public_ip } ```
```

- Returns a list of public IPs for the created instances.

### `instance\_ids`

```
``` output "instance_ids" { value = aws_instance.demo-ec2[*].id } ```
```

- Returns the EC2 instance IDs created in this deployment.

■ How to Use

1. ****Initialize Terraform**** ```bash terraform init ```
2. ****Plan the Infrastructure**** ```bash terraform plan ```
3. ****Apply the Configuration**** ```bash terraform apply ```
4. ****Destroy Resources**** ```bash terraform destroy ```

■ Notes

- For secure access, avoid hardcoding credentials. - Use `terraform.tfvars` to override default variable values if needed. - Ensure the AMI ID used is valid for the selected region.