Multicloud DevOps & AI Challenge - Day 1: Automating AWS Provisioning with Terraform

Overview

This document serves as a comprehensive guide for automating AWS provisioning using Terraform. It includes step-by-step instructions for setting up an AWS environment, creating an S3 bucket, and provisioning DynamoDB tables. This guide ensures that users can quickly deploy infrastructure as code in AWS.

Step 1: Generating Terraform Code for S3 Bucket

1. Using Claude AI

- Ask Claude to generate Terraform code for an S3 bucket using the prompt:
  > "Please provide Terraform code to create an S3 bucket in AWS with a unique name."
- Claude will generate the following Terraform code:

```
provider "aws" {
  region = "us-west-2"  # Replace with your desired region
}
resource "random_id" "bucket_suffix" {
  byte_length = 8
}
resource "aws_s3_bucket" "my_bucket" {
  bucket = "my-unique-bucket-name-
${random_id.bucket_suffix.hex}"

  tags = {
    Name        = "My bucket"
    Environment = "Dev"
  }
}
resource "aws_s3_bucket_acl" "my_bucket_acl" {
  bucket = aws_s3_bucket.my_bucket.id
  acl    = "private"
}
```

2. Save this code as ‘main.tf’ for later use.

Step 2: Creating an IAM Role for EC2

- Log in to the AWS Management Console.
- Navigate to IAM -> Roles.
- Click Create Role.
- Select AWS service as the trusted entity and choose EC2.
- Attach the AdministratorAccess policy (Use restricted policies in production).
- Name the role ‘EC2Admin’ and add a description.
- Review and create the role.

Step 3: Launching an EC2 Instance

- Navigate to EC2 Dashboard -> Launch Instance.
- Choose Amazon Linux 2 AMI.
- Select t2.micro as the instance type.
- Configure instance details:
  - Network: Default VPC
  - Subnet: Any available
  - Auto-assign Public IP: Enabled
  - IAM Role: ‘EC2Admin’
- Keep default storage settings.
- Add a tag: Key: ‘Name’, Value: ‘workstation’
- Create a security group allowing SSH access from EC2 Instance Connect IP.
- Launch the instance with a key pair.

Step 4: Connecting to EC2 and Installing Terraform

- From EC2 Dashboard, select your ‘workstation’ instance.
- Click Connect- > ‘EC2 Instance Connect’(If it doesn’t work use SSH client).
- Execute the following commands:

```
sudo yum update -y
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo     https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
sudo yum -y install terraform
terraform version
```

Step 5: Applying Terraform Configuration

- Create a new directory and navigate to it:
  mkdir terraform-project && cd terraform-project
- Create and open `main.tf`:
  vi main.tf
- Paste the generated Terraform code from Step 1.
- Initialize Terraform:
  terraform init
- Review the execution plan:
  terraform plan
- Apply the configuration:
- terraform apply
- Type `yes` to confirm resource creation.

Step 6: Verifying S3 Bucket Creation

- Use AWS CLI to list all buckets:
  aws s3 ls
- Confirm that the newly created bucket appears in the list.

Step 7: Creating DynamoDB Tables

- Modify `main.tf` by replacing S3 resource definitions with the following DynamoDB configuration:
  - ‘’’

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_dynamodb_table" "cloudmart_products" {
  name         = "cloudmart-products"
  billing_mode = "PAY_PER_REQUEST"
  hash_key     = "id"

  attribute {
    name = "id"
    type = "S"
  }
}

resource "aws_dynamodb_table" "cloudmart_orders" {
  name         = "cloudmart-orders"
  billing_mode = "PAY_PER_REQUEST"
  hash_key     = "id"
```

```
  attribute {
    name = "id"
    type = "S"
  }
}

resource "aws_dynamodb_table" "cloudmart_tickets" {
  name           = "cloudmart-tickets"
  billing_mode   = "PAY_PER_REQUEST"
  hash_key       = "id"

  attribute {
    name = "id"
    type = "S"
  }
}
‘’’
```

- Apply the updated Terraform configuration:
  terraform apply
- Type `yes` to confirm resource creation.

Conclusion

- By following this guide, I have successfully:
  - Automated AWS infrastructure provisioning with Terraform.
  - Created an S3 bucket.
  - Launched and connected to an EC2 instance.
  - Installed and used Terraform on the instance.
  - Created DynamoDB tables.