Simple Terraform Configuration for AWS EC2 with S3 State

Created by Grok 3

June 20, 2025

1 Overview

This document provides a simple Terraform configuration to provision an AWS EC2 instance with a security group, storing the state file in an S3 bucket with DynamoDB locking. It covers the core Terraform concepts: Providers, Resources, State, Variables, Data Sources, and Outputs.

2 Directory Structure

```
main.tf
provider.tf
variables.tf
outputs.tf
```

3 Terraform Files

3.1 provider.tf

Configures the AWS provider and S3 backend for state storage.

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

terraform {
  backend "s3" {
    bucket         = "my-terraform-state-bucket" # Replace with your S3 bucket name
    key            = "state/terraform.tfstate"
    region         = "us-east-1"
    dynamodb_table = "terraform-state-lock" # Replace with your DynamoDB table name
  }
}
```

3.2 variables.tf

Defines input variables for parameterization.

```
variable "region" {
  description = "AWS region for resources"
  type        = string
  default     = "us-east-1"
}

variable "instance_type" {
  description = "EC2 instance type"
  type        = string
  default     = "t2.micro"
}
```

3.3 main.tf

Defines the infrastructure, including data sources and resources.

```
# Fetch the default VPC
data "aws_vpc" "default" {
  default = true
}

# Fetch available subnets in the default VPC
data "aws_subnets" "available" {
  filter {
    name   = "vpc-id"
    values = [data.aws_vpc.default.id]
  }
}

# Fetch the latest Ubuntu AMI
data "aws_ami" "latest_ubuntu" {
  most_recent = true
  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-
       server-*"]
  }
  owners = ["099720109477"] # Canonical
}

# Create a security group
resource "aws_security_group" "ec2_sg" {
  name        = "ec2-security-group"
  description = "Allow SSH and HTTP"
  vpc_id      = data.aws_vpc.default.id

  ingress {
    from_port   = 22
```

```
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "ec2-security-group"
  }
}

# Create the EC2 instance
resource "aws_instance" "web" {
  ami                    = data.aws_ami.latest_ubuntu.id
  instance_type          = var.instance_type
  vpc_security_group_ids = [aws_security_group.ec2_sg.id]
  subnet_id              = data.aws_subnets.available.ids[0]
  associate_public_ip_address = true

  tags = {
    Name = "web-server"
  }
}
```

3.4 outputs.tf

Defines the output for the EC2 instances public IP.

```
output "instance_public_ip" {
  description = "Public IP of the EC2 instance"
  value       = aws_instance.web.public_ip
}
```

4 Implementation Steps

1. **Set up AWS credentials**: Configure credentials in `~/.aws/credentials` or via environment variables:

```
export AWS_ACCESS_KEY_ID="your_access_key"
export AWS_SECRET_ACCESS_KEY="your_secret_key"
```

2. **Create an S3 bucket and DynamoDB table**:
   - **S3 Bucket**: Create a bucket named `my-terraform-state-bucket`:

```
aws s3 mb s3://my-terraform-state-bucket --region us-east-1
```

   - **DynamoDB Table**: Create a table named `terraform-state-lock` with `LockID` (string) as the primary key:

```
aws dynamodb create-table \
  --table-name terraform-state-lock \
  --attribute-definitions AttributeName=LockID,AttributeType=S \
  --key-schema AttributeName=LockID,KeyType=HASH \
  --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5 \
  --region us-east-1
```

3. **Create the Terraform files**: Save the provided `provider.tf`, `variables.tf`, `main.tf`, and `outputs.tf` in the project directory. Update the `bucket` and `dynamodb_tablenamesin`

4. **Navigate to the project directory**:

```
cd /path/to/project
```

5. **Initialize Terraform**:

```
terraform init
```

   Downloads the AWS provider and configures the S3 backend.

6. **Validate the configuration**:

```
terraform validate
```

   Checks for syntax errors.

7. **Format the configuration**:

```
terraform fmt
```

   Ensures consistent formatting.

8. **Generate an execution plan**:

```
terraform plan
```

   Previews the resources to be created.

9. **Apply the configuration**:

```
terraform apply
```

Provisions the resources (confirm with `yes`).

10. **View outputs**:

```
terraform output
```

Displays the EC2 instances public IP.

11. **Destroy resources** (when done):

```
terraform destroy
```

Removes all resources (confirm with `yes`).

5 Core Concepts Demonstrated

- **Providers**: Configured in `provider.tf` for AWS.
- **Resources**: EC2 instance and security group defined in `main.tf`.
- **State**: Stored in an S3 bucket with DynamoDB locking, configured in `provider.tf`.
- **Variables**: Defined in `variables.tf` for region and instance type.
- **Data Sources**: Used in `main.tf` to fetch VPC, subnets, and AMI.
- **Outputs**: Defined in `outputs.tf` for the public IP.

6 Notes

- The security group allows SSH (port 22) and HTTP (port 80) from `0.0.0.0/0`. In production, restrict `cidr`$_b$$lockstospecificIPs.EnsuretheS3bucketandDynamoDBtableexistbefo$

- Compile this LaTeX document with `latexmk -pdf terraform`$_e$$c2_s3_state.textogeneateaPDF.$