AWS Configure

To configure **AWS for Terraform**, first set up AWS CLI with your credentials, then reference those credentials within your Terraform AWS provider block. This allows Terraform to authenticate with AWS and manage infrastructure resources automatically.

Step 1: Configure AWS CLI

Start by installing and configuring AWS CLI:

- Install AWS CLI for your OS.

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-aarch64.zip" -o "awscliv2.zip"
```

- Unzip the Installer.

```
unzip awscliv2.zip
```

- run the installer.

Navigate into the unzipped directory and run the install script.

```
sudo ./aws/install
```

This command installs the AWS CLI to /usr/local/bin by default. Verify the Installation.
Check the installed AWS CLI version to confirm the installation was successful.

```
aws --version
```

- Run aws configure in your terminal.

```
aws configure
```

Enter your AWS Access Key ID, Secret Access Key, default region, and output format (typically `json`).

This saves credentials in `~/.aws/credentials` and settings in `~/.aws/config`.

Step 2: Install Terrafrom:

```
sudo apt update && sudo apt upgrade -y
```

Install required dependencies.

```
sudo apt install -y gnupg software-properties-common curl
```

Add the HashiCorp GPG key.

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

Add the HashiCorp repository.

```
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
```

Update package list and install Terraform.

```
sudo apt update
sudo apt install -y terraform
```

Verify the installation.

```
terraform version
```

2. Manual Binary Download and Installation

Download the latest Terraform binary (example for version 1.13.1 amd64)

```
wget https://releases.hashicorp.com/terraform/1.13.1/terraform_1.13.1_linux_amd64.zip

# Unzip the binary

unzip terraform_1.13.1_linux_amd64.zip

# Move the terraform binary to /usr/local/bin

sudo mv terraform /usr/local/bin/

# Verify installation

terraform -version
```

Step 3: Reference Credentials in Terraform

Terraform's AWS provider reads credentials from environment variables or shared config files:

- Use the default profile, or specify a profile.
- Typical provider block example:

```
provider "aws" {
  region                  = "us-east-1"
  shared_config_files     = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
  profile                 = "default"  # or the name of your profile
}
```

Terraform automatically discovers credentials unless they are overridden in the provider block.

Step 4: Initialize Terraform Project

- In your project directory, write your infrastructure `.tf` files (e.g., `main.tf`) specifying AWS resources.
- Run `terraform init` to initialize Terraform and download the AWS provider plugin.
- Use main.tf file from documents

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.16"
    }
  }

  required_version = ">= 1.2.0"
}

provider "aws" {
  region = "ap-south-1"
```

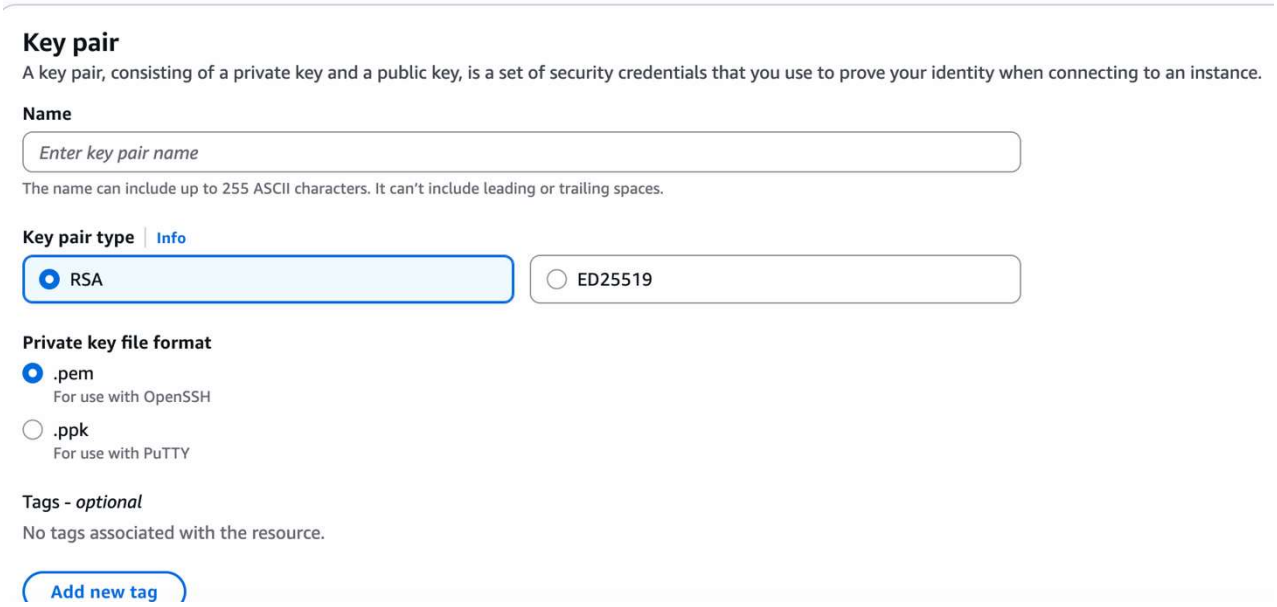
```
}

resource "aws_instance" "node1" {
  #ami           = "ami-0af9569868786b23a"
  ami = "ami-0f918f7e67a3323f0"
  instance_type = "t2.micro"
  key_name      = "yogeshpem". // Change to your keypair name
  tags = {
    Name = "node1"
  }
}

/* resource "aws_instance" "node2" {
  ami           = "ami-0af9569868786b23a"
  instance_type = "t2.micro"
  key_name      = "yogeshpem"
  tags = {
    Name = "node2"
  }
} */
```

Step 5: Create Key Pair

- Create a Key pair

https://ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#CreateKeyPair:

https://ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#KeyPairs:



- Download key pair
- Use keypair in Code

Step 6: Plan and Apply Infrastructure

- Validate configuration with `terraform validate`.
- Preview changes via `terraform plan`.
- Deploy with `terraform apply` and confirm when prompted.
- To tear down, run `terraform destroy`.

Alternatives: Using Environment Variables

Credentials can also be sourced from environment variables:

```
export AWS_ACCESS_KEY_ID="your-access-key"
export AWS_SECRET_ACCESS_KEY="your-secret-key"
export AWS_REGION="us-east-1"
```