# **Terraform Install**

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

wget -O- https://apt.releases.hashicorp.com/gpg | \

gpg --dearmor | \

sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg

gpg --no-default-keyring \

--keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg \

--fingerprint

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 -y

sudo apt-get install terraform -y

terraform --version

## **Pre-req for Terraform to work**

The Terraform CLI (1.2.0+) installed. The AWS CLI installed. AWS account and associated credentials that allow you to create resources

## **AWS CLI**

<https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>

sudo apt install unzip -y

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

/usr/local/bin/aws --version

## **configure AWS**

grab aws ACCESS ID and SECRET key from aws profile top right side -> security crendentials -> create secret key, then paste below after command

$aws configure

## **confirm aws configuration**

aws sts get-caller-identity

## **Terraform script**

Script 1: Create ec2 instance

provider "aws" {

region = "ap-south-1"

}

resource "aws\_instance" "instance1" {

ami = "ami-0f5ee92e2d63afc18"

instance\_type = "t2.micro"

tags = {

Name = "terraform-instance"

}

}

terraform init

terraform plan

terraform apply

Script 2 - Add sshkey to instance

provider "aws" {

region = "ap-south-1"

}

resource "aws\_key\_pair" "sshkey" {

key\_name = "sshkey"

public\_key = file("/home/ubuntu/.ssh/id\_rsa.pub")

}

resource "aws\_instance" "instance1" {

ami = "ami-0f5ee92e2d63afc18"

instance\_type = "t2.micro"

key\_name = aws\_key\_pair.sshkey.key\_name

tags = {

Name = "terraform-instance"

}

}

Script 3 - Update the infrastructure

provider "aws" {

region = "ap-south-1"

}

resource "aws\_instance" "instance1" {

ami = "ami-0f5ee92e2d63afc18"

instance\_type = "t2.micro"

key\_name = "DevOps"

vpc\_security\_group\_ids = ["sg-00fbfd872907f4e0a"]

tags = {

Name = "terraform-instance"

}

provisioner "remote-exec" {

connection {

type = "ssh"

user = "ubuntu"

private\_key = file("/home/ubuntu/DevOps.pem")

host = self.public\_ip

}

inline = [

"echo This command is executed remotely on the EC2 instance.",

"sudo apt-get install -y nano",

"sudo apt-get install -y git",

# Add any additional commands here for remote execution

]

}

}

# script 4 - Multiple instances

provider "aws" {

region = "ap-south-1"

}

resource "aws\_key\_pair" "sshkey" {

key\_name = "sshkey"

public\_key = file("/home/ubuntu/.ssh/id\_rsa.pub")

}

resource "aws\_instance" "instance1" {

count = 2

ami = "ami-0f5ee92e2d63afc18"

instance\_type = "t2.micro"

key\_name = "DevOps"

vpc\_security\_group\_ids = ["sg-00fbfd872907f4e0a"]

tags = {

Name = "terraform-instance-${count.index + 1}"

Environment = "Development"

}

}

## To destroy specific instances

terraform destroy -target=aws\_instance.instance1[0]

General destroy command to destroy all:

terraform destroy

**End to End architecture:**

provider "aws" {

region = "ap-south-1" # Set your desired AWS region

}

# Create VPC

resource "aws\_vpc" "my\_vpc" {

cidr\_block = "10.0.0.0/16"

enable\_dns\_support = true

enable\_dns\_hostnames = true

}

# Create Subnet

resource "aws\_subnet" "my\_subnet" {

vpc\_id = aws\_vpc.my\_vpc.id

cidr\_block = "10.0.1.0/24"

availability\_zone = "ap-south-1a" # Set your desired availability zone

}

# Create Internet Gateway

resource "aws\_internet\_gateway" "my\_igw" {

vpc\_id = aws\_vpc.my\_vpc.id

}

# Create Route Table

resource "aws\_route\_table" "my\_route\_table" {

vpc\_id = aws\_vpc.my\_vpc.id

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.my\_igw.id

}

}

# Associate Subnet with Route Table

resource "aws\_route\_table\_association" "my\_subnet\_association" {

subnet\_id = aws\_subnet.my\_subnet.id

route\_table\_id = aws\_route\_table.my\_route\_table.id

}

# Create Security Group

resource "aws\_security\_group" "my\_security\_group" {

vpc\_id = aws\_vpc.my\_vpc.id

# Ingress rules for SSH and HTTP (port 22 and 80)

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"]

}

}

# Create EC2 Instance

resource "aws\_instance" "my\_instance" {

ami = "ami-0287a05f0ef0e9d9a" # Set your desired AMI ID

instance\_type = "t2.micro" # Set your desired instance type

subnet\_id = aws\_subnet.my\_subnet.id

key\_name = "DevOps" # Set your key pair name

security\_groups = [aws\_security\_group.my\_security\_group.id]

# User Data to run a shell script on instance launch

user\_data = file("/home/ubuntu/install\_nginx.sh")

tags = {

Name = "MyEC2Instance"

}

}