**TERRAFORM**

**What is Terraform?**

Terraform is an open source infrastructure as a code software tool created by Harshicorp that enables a developer to safely and predictably create, change and improve infractructure.

**Interview question:**

How do you store your statefile?

Its stored on the S3 in your AWS account and make a reference to it on your Visual Access Code (VSC) eg backend.tf

* Download Terraform using terraform.io <https://www.terraform.io/downloads.html>
* Extract to a folder
* Copy the path where you downloaded the terraform on.
* Go to Env or your system environment, click system variable, go to environment variable, click on the path.
* Paste the path you copied on the edit path.
* Go to cmd, to check the version of terraform installed by typing terraform -v
* Download vscode
* Launch the vscode.
* On your VScode, install Hashcorp terraform, AWS CLI configure and AWS Toolkit.
* Create an empty file in an already created folder. Example **xxfile.tf (Saving in .tf very important)**

**Step 1: Terraform AWS provider**

<https://registry.terraform.io/providers/hashicorp/aws/latest/docs>

* Under static credentials

provider "aws" {

region = "us-west-2"

access\_key = "my-access-key"

secret\_key = "my-secret-key"

}

* Edit the region
* Edit the access key
* Edit the secret key
* Remember to place all edited script in (“”)

**Step 2: terraform aws ec2-instance**

<https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/instance>

* Under Resources

resource "aws\_instance" "web" {

ami = data.aws\_ami.ubuntu.id

instance\_type = "t3.micro"

tags = {

Name = "HelloWorld"

}

}

* Edit from the data.aws\_ami...... with an Ubuntu from AMI in your console.
* Edit the t3.micro to t2.micro
* Remember to place all edited script in (“”)

Always remember to Save.

**Test your script**

Type: Terraform init : initialize

Type: Terraform Plan : To see if there are any changes to be made/ precautionary step

Type: Terraform Apply

Type: Yes

Type: Terraform Destroy: To delete the instance

Type yes.

**Step 3: Adding Security group**

Type aws security group: Download from the resource

resource "aws\_security\_group" "allow\_tls" {

name = "allow\_tls"

description = "Allow TLS inbound traffic"

vpc\_id = aws\_vpc.main.id

ingress = [

{

description = "TLS from VPC"

from\_port = 443

to\_port = 443

protocol = "tcp"

cidr\_blocks = [aws\_vpc.main.cidr\_block]

ipv6\_cidr\_blocks = [aws\_vpc.main.ipv6\_cidr\_block]

}

]

egress = [

{

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

ipv6\_cidr\_blocks = ["::/0"]

}

]

tags = {

Name = "allow\_tls"

}

}

Notes:

Add security\_groups = [aws\_security\_group.allow\_tls.name] under instance type (line 8) and space down.

* Edit cidr blocks = [“0.0.0.0/0”] on both ingress and egress
* Edit the egress to be the same as ingress i.e 443, to port 443 and protocol “tcp”
* Infront of the ingress, delete the = and [ and backspace.
* Infront of the egress, delete the = and [ and backspace.
* Delete the vpc id……
* Delete the description = allow TLS(line 17 & 20)
* Ensure to delete the ] thereby leaving } to end each command/references.
* Ensure you use [ ] not { }

**Final script should be like this:**

provider "aws" {

  region     = "us-east-1"

  access\_key = "AKIAU24EQI576F4VMEHI"

  secret\_key = "o+SansH1KIo2L7KhMuPlS6lUE+gGmYdZf9jVookG"

}

resource "aws\_instance" "web" {

  ami           = "ami-09e67e426f25ce0d7"

  instance\_type = "t2.micro"

Count = 5

security\_groups = [aws\_security\_group.allow\_tls.name]

  tags = {

    Name = "Welcome to my world"

  }

}

resource "aws\_security\_group" "allow\_tls" {

  name        = "allow\_tls"

  ingress {

      from\_port        = 443

      to\_port          = 443

      protocol         = "tcp"

      cidr\_blocks      = ["0.0.0.0/0"]

    }

  egress {

      from\_port        = 443

      to\_port          = 443

      protocol         = "tcp"

      cidr\_blocks      = ["0.0.0.0/0"]

    }

  tags = {

    Name = "allow\_tls"

  }

}

**Test your script**

* Type: Terraform init : initialize
* Type: Terraform Plan : To see if there are any changes to be made/ precautionary step
* Type: Terraform Apply
* Type: Yes
* Type: Terraform Destroy: To delete the instance/security group
* Type yes.

**Step 4: Creating multiple Instances**

resource "aws\_instance" "web" {

ami = data.aws\_ami.ubuntu.id

instance\_type = "t3.micro"

tags = {

Name = "HelloWorld"

}

}

Note:

* Repeat scripts while changing the resource name i.e web
* Rename the tags
* Change the instance type to t2 micro
* Always remember to close with} every time to end a reference.

**OR**

**Input Count = 3(depending on how many instance you intend to run) beneath instance type = t2 micro. Close the script with }and run it.**

**Deploying VPC**

Google AWS terraform VPC resources.

resource "aws\_vpc" "main" {

cidr\_block = "10.0.0.0/16"

}

**To Learn about Terraform command, on the terminal, type terraform and enter**

**Deploying EC2 with bootstrap attached.**

resource "aws\_instance" "web" {

ami = data.aws\_ami.ubuntu.id

instance\_type = "t3.micro"

tags = {

Name = "HelloWorld"

}

}

**After the script,**

* **Type user\_data = <<-EOF**
* **Paste the apache file.**
* **End the script with EOF “meaning End of file”**

**To delete just a resource form the script: type terraform destroy -target aws\_instance.web**

**To create internet gateway**

Google Terraform internet gateway resources

resource "aws\_internet\_gateway" "gw" {

vpc\_id = aws\_vpc.main.id

tags = {

Name = "main"

}

}

To list all resources deployed on the visual studio code (VSC): TYPE terraform state list then enter.

**Coded run**

provider "aws" {

region     = "us-east-1"

}

resource "aws\_vpc" "vpc1" {

  cidr\_block       = "10.0.0.0/16"

  tags = {

    Name = "Test1"

  }

}

resource "aws\_subnet" "public" {

  vpc\_id     = aws\_vpc.vpc1.id

  cidr\_block = "10.0.0.0/24"

  tags = {

    Name = "Main"

  }

}

resource "aws\_subnet" "private" {

  vpc\_id     = aws\_vpc.vpc1.id

  cidr\_block = "10.0.1.0/24"

  tags = {

    Name = "Main"

  }

}

resource "aws\_internet\_gateway" "gw" {

  vpc\_id = aws\_vpc.vpc1.id

  tags = {

    Name = "main"

  }

}

resource "aws\_route\_table" "rt1" {

  vpc\_id = aws\_vpc.vpc1.id

  route {

      cidr\_block = "0.0.0.0/0"

      gateway\_id = aws\_internet\_gateway.gw.id

    }

  tags = {

    Name = "example"

  }

}

resource "aws\_route\_table\_association" "rta" {

  subnet\_id      = aws\_subnet.public.id

  route\_table\_id = aws\_route\_table.rt1.id

}

resource "aws\_instance" "Test" {

  ami           = "ami-02e136e904f3da870"

  instance\_type = "t2.micro"

  availability\_zone = "us-east-1c"

  subnet\_id      = aws\_subnet.public.id

  security\_groups = [aws\_security\_group.allow\_tls.id]

  tags = {

    Name = "Test1"

  }

}

resource "aws\_security\_group" "allow\_tls" {

  name        = "allow\_tls"

  description = "Allow TLS inbound traffic"

  vpc\_id      = aws\_vpc.vpc1.id

  ingress {

      description      = "ssh from VPC"

      from\_port        = 22

      to\_port          = 22

      protocol         = "tcp"

      cidr\_blocks      = ["0.0.0.0/0"]

    }

    ingress {

      description      = "http from VPC"

      from\_port        = 80

      to\_port          = 80

      protocol         = "tcp"

      cidr\_blocks      = ["0.0.0.0/0"]

    }

    ingress {

      description      = "https from VPC"

      from\_port        = 443

      to\_port          = 443

      protocol         = "tcp"

      cidr\_blocks      = ["0.0.0.0/0"]

    }

  tags = {

    Name = "allow\_tls"

  }

}