Lab 1 – Terraform Setup

Task – Install Terraform and create an EC2

Prerequisite: EC2 Ubuntu instance you have connected to, AWS account credentials (access and secret_key).

1) Go to this link https://www.terraform.io/downloads and run the commands it specifies to install Terraform on Ubuntu

2) Create a new directory to contain your Terraform files and create a file called main.tf which contains the following (replace access_key and secret_key)

```
provider "aws" {
    access_key = "WERF864FE8EFE8FEASE"
    secret_key = "864wef684ef646ewf8ew6f4684wef"
    region = "eu-west-2"
}

resource "aws_instance" "demo1" {
    ami = "ami-0fb391cce7a602d1f"
    instance_type = "t2.micro"
}
```

IMPORTANT – DO NOT PUSH THIS CODE UP TO GITHUB OR ANY VERSION CONTROL SYSTEM WITH YOUR CREDENTIALS

3) Run the following commands to create your EC2 instance using AWS

```
`terraform init`
`terraform plan`
`terraform apply
```

Stretch goal – Try to connect to your EC2 instance, this will require configuration of your .tf file see what you can come up with.

Lab 2 – TF Resources

Task – Create Variables and simple resources

1) Create a file called 'variables.tf' and add the following to that file (replacing the default values with your keys):

```
variable "secret_key" {
    default = "684FE8F4EFW68FWEF684EFW"
    sensitive = true
}

variable "access_key" {
    default = "nregrg6g84erg864gre684erg6erg"
    sensitive = true
}
```

2) Create a .gitIgnore file and add variables.tf to the file. You have now stopped your keys being pushed up to VCS so you are free to use Git.

3) Modify the main.tf provider **access_key** and **secret_key** to equal **var.access_key** and **var.secret_key**

4) Use the TF docs to create an S3 bucket connected to your account with the name being set with a variable (add this to your variables.tf file)

5) Use **terraform destroy** to reset your AWS environment, then use **plan** and **apply** to build the new resources

Stretch goal – Use variables to set different default values for your aws_instance.

Lab 3 – TF Output

Task – Output a Public IP

Create a new file called output.tf and add the following to it:

```
output "vm_public_ip" {
    value = aws_instance.<instance label>.public_ip
}
```

Also create an output to return the S3 bucket domain, use **destroy, plan, apply**.

Lab 4 – Variable Configuration

Task – Use Vars properly and SSH connect to an EC2

You need to use a variety of -var, .tfvars, environment values, and default values to achieve the following environment:

- EC2 with specified key pair
- Key Pair
- Security Group that allows SSH access

All resources should use variables in some way for each value, and they should all use a tag with key = project and value = lab_4

Lab 5 – VPC Setup

Task – Create a VPC with all resources using Terraform

Using the Terraform docs and information from creating an AWS VPC from scratch, you should use terraform to provision a complete VPC with an EC2. You should use variables wherever possible and output any useful information:

- VPC
- 2x public subnets
- 1x security group (SSH, HTTP, SQL ports)
- Route table (and routes)
- Route table associations
- IGW
- EC2

Stretch goal - SSH into your EC2 without having to access AWS GUI at all or using any prebuilt resources.

Lab 6 – TF Modules

Task 1 – Create an EC2 module

Using the TF Module file structure below:

```
- main.tf
- output.tf
- variables.tf
/ EC2
    - main.tf
    - output.tf
    - variables.tf
```

1) Within main.tf of EC2 add your aws_instance code as well as variables in the EC2 variables.tf file

2) Within the root main.tf underneath your provider add the following:

```
module "ec2_1" {
    source = "./EC2"
}
```

3) Use the standard terraform destroy, plan and apply to rebuild the environment using the new module

4) Add a second ec2_2 module to your root main.tf and update the EC2 main.tf to allow you to specify the subnet_id it should connect to. Using the root main.tf, make the two EC2s connect to two different subnets

Task 2 – Modularise existing code

Using the Module file structure, split your VPC into sensible modules. You should ensure outputs and variables are being used and resources are not being duplicated.