App1-install.sh

#! /bin/bash

# Instance Identity Metadata Reference - https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-identity-documents.html

sudo yum update -y

sudo yum install -y httpd

sudo systemctl enable httpd

sudo service httpd start

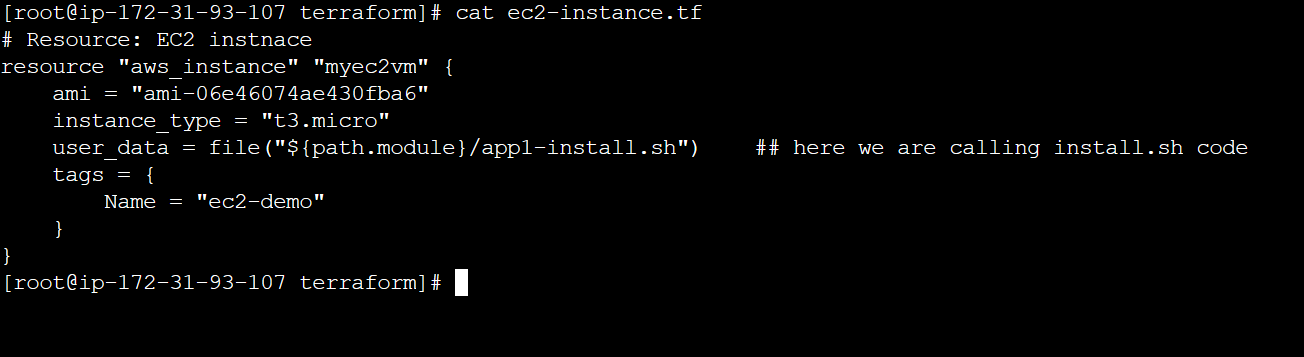
sudo echo '<h1>Welcome to StackSimplify - APP-1</h1>' | sudo tee /var/www/html/index.html

sudo mkdir /var/www/html/app1

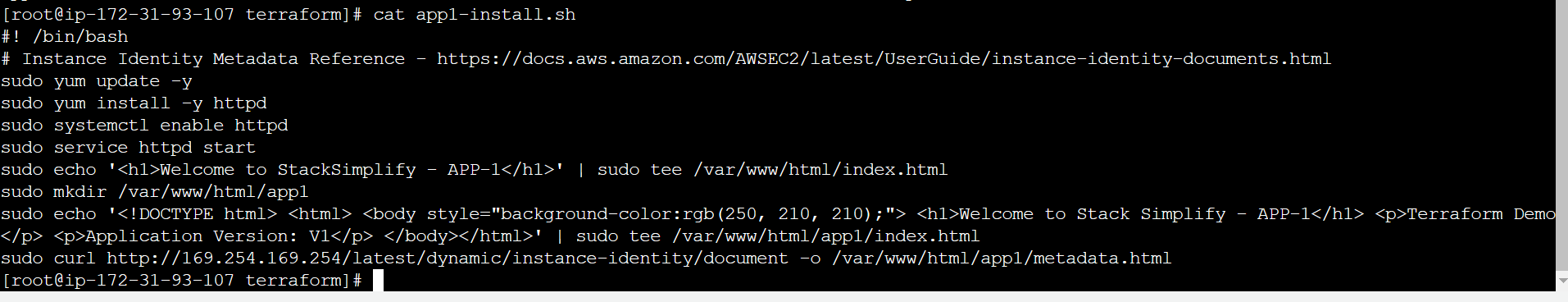
sudo echo '<!DOCTYPE html> <html> <body style="background-color:rgb(250, 210, 210);"> <h1>Welcome to Stack Simplify - APP-1</h1> <p>Terraform Demo</p> <p>Application Version: V1</p> </body></html>' | sudo tee /var/www/html/app1/index.html

sudo curl [http://169.254.169.254/latest/dynamic/instance-identity/document -o /var/www/html/app1/metadata.html](http://169.254.169.254/latest/dynamic/instance-identity/document%20-o%20/var/www/html/app1/metadata.html)

c2-ec2instnace.tf or ec2-instance.tf



App1-install.sh



Terraform init

Terraform validate

Terraform plan

Terraform apply

Letc check our index.html is created or not

If the server is not reached –security group – edit –in bound rule- add – HTTP – Anywhere



Check with index.html now

**Terraform state Basics:**

Terraform state:

=============

terraform state comes under

terraform resource

1. create resource- create resource with existing configuration but are not associated with in real infrastructure object in the state.

2. Destroy resource -

3. update-in-place resource

4. Destroy and re-create

**Important Note: terraform.tf state file is having real infrastructure, what is present on the colud.**

What ever the code inside manifest file we are calling desired state

And what ever inside present in cloud called current state

**The primary purpose of terraform state is store binding b/w objects in remote system and resource instance declared in your configuration**.

When tf create remote object in response to change configuration, it will record the identity of that remote object against a particular resource instance, and then potentially update or delete the object in response to future configuration changes

**Desired state –means tf configuration files.**

**Current state – Real world Resources-EC2-Instnace**

**Cleanup:**

Terraform plan –destroy

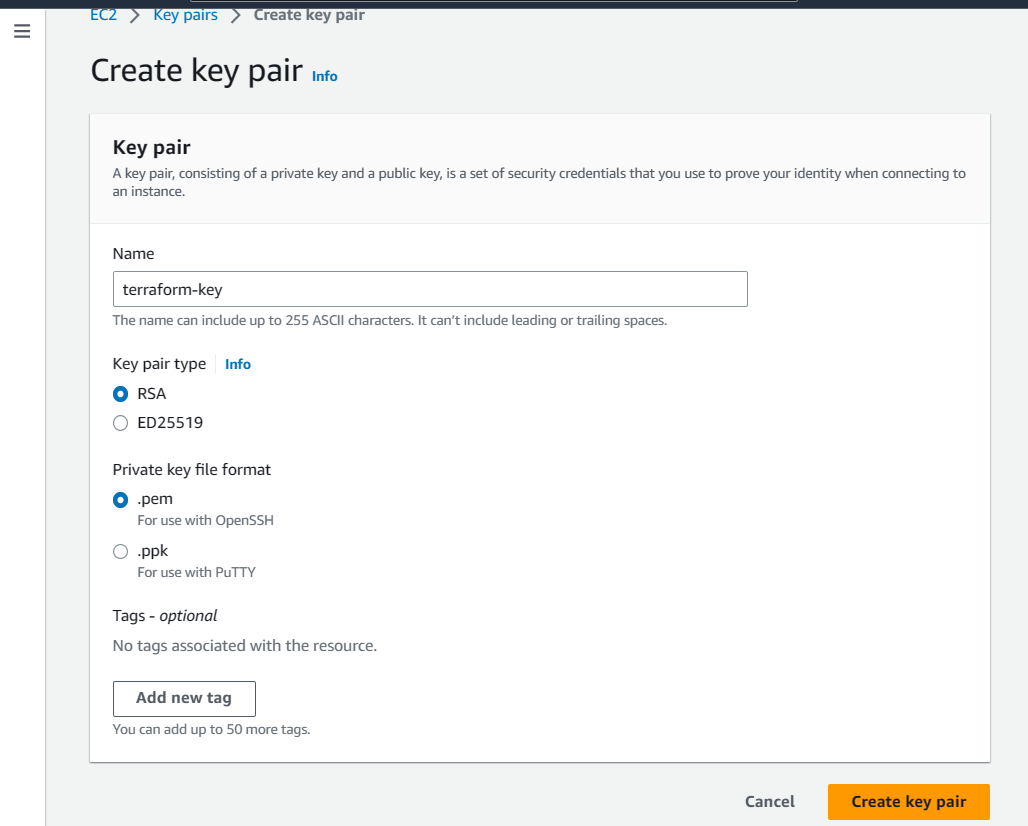
Terraform destroy

**Terraform Variables and Datasources:**

[terraform-on-aws-ec2/04-Terraform-Variables-and-Datasources at main · grvsobha/terraform-on-aws-ec2 (github.com)](https://github.com/grvsobha/terraform-on-aws-ec2/tree/main/04-Terraform-Variables-and-Datasources)

**Step-00: Pre-requisite Note**

* Create a terraform-key in AWS EC2 Key pairs which we will reference in our EC2 Instanc



Now lets get discuss about Input Variables Output Variables :

# Terraform Variables and Datasources

## Step-00: Pre-requisite Note

* Create a terraform-key in AWS EC2 Key pairs which we will reference in our EC2 Instance

## Step-01: Introduction

### Terraform Concepts

* Terraform Input Variables
* Terraform Datasources
* Terraform Output Values

### What are we going to learn ?

1. Learn about Terraform Input Variable basics

* AWS Region
* Instance Type
* Key Name

1. Define Security Groups and Associate them as a List item to AWS EC2 Instance

* vpc-ssh
* vpc-web

1. Learn about Terraform Output Values

* Public IP
* Public DNS

1. Get latest EC2 AMI ID Using Terraform Datasources concept
2. We are also going to use existing EC2 Key pair terraform-key
3. Use all the above to create an EC2 Instance in default VPC

## Step-02: c2-variables.tf - Define Input Variables in Terraform

Terraform variables are divided in 3 types

* 1. Input Variables
  2. Output Variables
  3. Local Variables

**Input variables:**

* Input variables are like function arguments.

Input variables serves as parameter for a terraform module, allowing aspects of the module to be customized without altering the module own source code, and allowing a module to be shared between configurations.

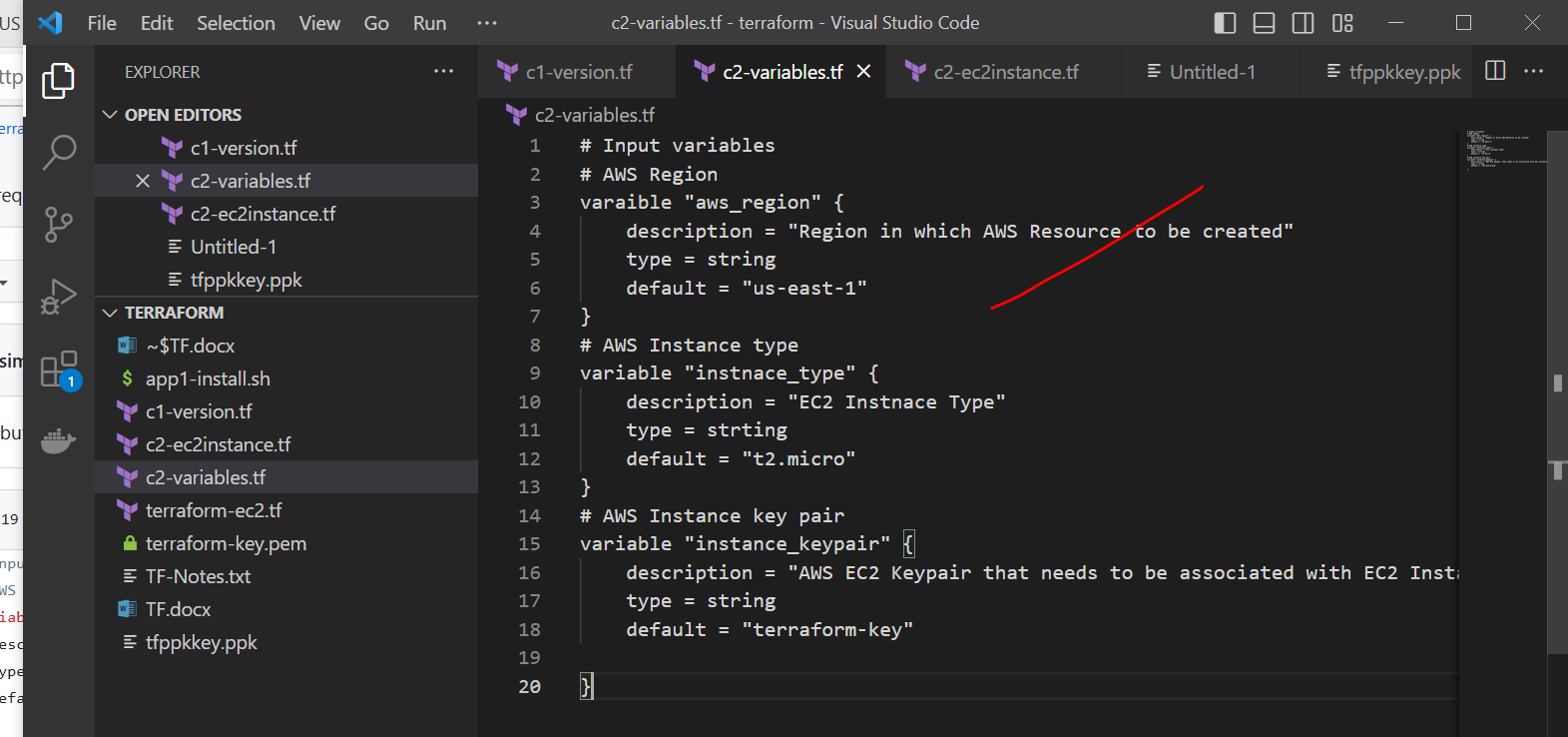
**We can pass 10diffrent ways**

* 1. Input Variables – Basics
  2. Provided input variables when prometed during terraform plan or apply
  3. Override default variable value using CLI argument –var
  4. Override default variable value using Environment Variables(TF\_var\_aa)
  5. Provided input variables using terraform.tfvars files
  6. Provided input variables using<any name>tfvars file with CLI argument –var-file
  7. Provided input variables using auto.tfvars file
  8. Implementation complex type constructions like List & Map In input variables
  9. Implementation custom validation rules in variables
  10. Protect Sensitive Input Variables

## Step-03: c3-ec2securitygroups.tf - Define Security Group Resources in Terraform

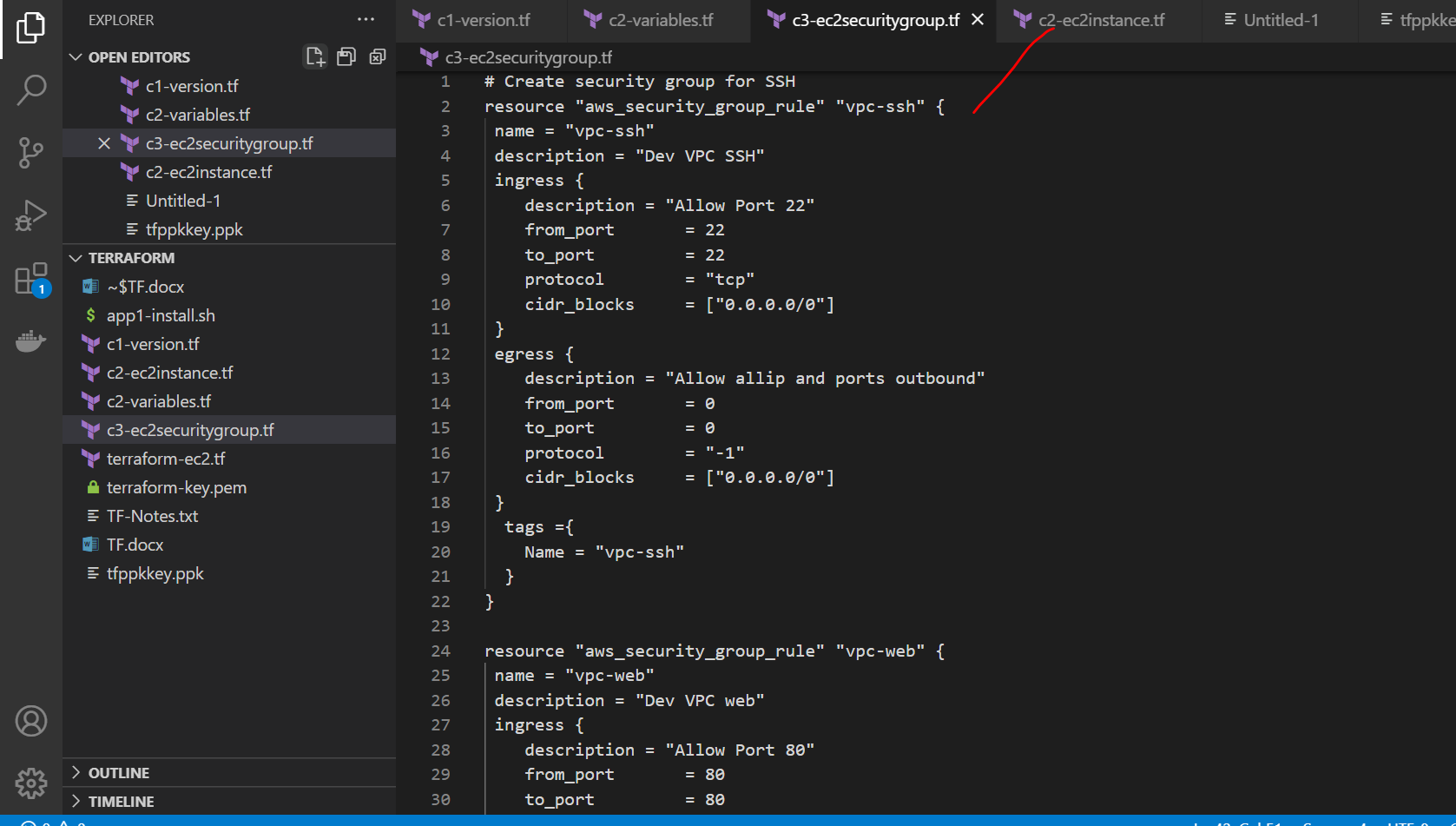
Implement Variables in TF

[terraform-on-aws-ec2/c2-variables.tf at main · grvsobha/terraform-on-aws-ec2 (github.com)](https://github.com/grvsobha/terraform-on-aws-ec2/blob/main/04-Terraform-Variables-and-Datasources/terraform-manifests/c2-variables.tf)



Implementation of security Group:

[terraform-on-aws-ec2/c3-ec2securitygroups.tf at main · grvsobha/terraform-on-aws-ec2 (github.com)](https://github.com/grvsobha/terraform-on-aws-ec2/blob/main/04-Terraform-Variables-and-Datasources/terraform-manifests/c3-ec2securitygroups.tf)



## Step-04: c4-ami-datasource.tf - Define Get Latest AMI ID for Amazon Linux2 OS

Terraform Datasource:

Data sources allows data to be fetched or computed for use elsewhere in Terraform configuration.

Use of data sources allows a Terraform configuration to make use of information defined outside of terraform, or defined by another terraform configuration.

A data source is accessed via a special kind of resource known as a data resource, declared using a data block.

Each data resource is associated with a single data source, which determines the kind of object(or, objects), it reads and what query constraint arguments are available.

Data resource have the same dependency resolution behaviour as defined for managed resource. setting the depends\_on\_meta\_argument which data blocks **defers reading** of the data source until after all changes to the dependencies have been applied.

## Step-05: c5-ec2instance.tf - Define EC2 Instance Resource

[terraform-on-aws-ec2/04-Terraform-Variables-and-Datasources at main · grvsobha/terraform-on-aws-ec2 (github.com)](https://github.com/grvsobha/terraform-on-aws-ec2/tree/main/04-Terraform-Variables-and-Datasources)

## Step-06: c6-outputs.tf - Define Output Values

Output vales are like the return values of a Terraform module and have several uses

* 1. A root module can use output to print certain values in the CLI output after running terraform apply.
  2. A child module can se output to expose a subset of its resource attributes to a parent module.
  3. When sing remote state, root module output can be accessed by other configuration via a terraform\_remote\_state data source

[terraform-on-aws-ec2/04-Terraform-Variables-and-Datasources at main · grvsobha/terraform-on-aws-ec2 (github.com)](https://github.com/grvsobha/terraform-on-aws-ec2/tree/main/04-Terraform-Variables-and-Datasources)

## Step-07: Execute Terraform Commands

Terraform init

Terraform validate

Terraform plan

Terraform apply

Chmod 400 private-key

Terraform destroy

**Terrafrom Loops Meta Arguments**:

Terrafrom list:

A **list variable** holds a list of values (for example, name of users) to be used. Each list variable specifies the order.

What is a Terraform map?

A map value is a lookup table from string keys to string values. This is useful for selecting a value based on some other provided value.