**LET’S SUMMARIZE THIS TERRAFORM CHEAT SHEET**

1. `**terraform init**`: Initialize a new or existing Terraform working directory.

2. `**terraform plan**`: Generate and show an execution plan for changes to infrastructure.

3. `**terraform apply**`: Apply the changes required to reach the desired state of the configuration.

4. `**terraform destroy**`: Destroy the Terraform-managed infrastructure.

5. `**terraform validate**`: Validate the configuration files in a directory.

6. `**terraform output**`: Read an output variable from a Terraform state file.

7. `**terraform refresh**`: Update the state file against real resources.

8. `**terraform import**`: Import existing infrastructure into Terraform.

9. `**terraform taint**`: Mark a resource instance as tainted, forcing it to be destroyed and recreated on the next `apply`.

10. `**terraform untaint**`: Remove the tainted state from a resource instance.

11. `**terraform state list**`: List resources within a Terraform state.

12. `**terraform state show**`: Show the attributes of a single resource in the Terraform state.

13. `**terraform state rm**`: Remove a resource from the Terraform state.

14. `**terraform state mv**`: Move an item within the Terraform state.

15. `**terraform fmt**`: Rewrites Terraform configuration files to a canonical format.

16. `**terraform get**`: Download and install modules for the configuration in the current directory.

17. `**terraform graph**`: Create a visual representation of a Terraform configuration or execution plan.

18. `**terraform import**`: Import existing infrastructure into Terraform.

19. `**terraform providers**`: Prints a tree of the providers used in the configuration.

20. `**terraform show**`: Inspect Terraform state or plan.

21. `**terraform force-unlock**`: Release a stuck lock on the current workspace/state.

22. `**terraform workspace**`: Workspace management.

23. `**terraform version**`: Prints the Terraform version.

24. `**terraform console**`: Interactive console for evaluating Terraform expressions.

25. `**terraform debug**`: Debug output management.

26. `**terraform state mv**`: Move an item within the Terraform state.

27. `**terraform state pull**`: Fetch and output the state from a remote backend.

28. `**terraform state push**`: Update remote state from a local state file.

29. `**terraform state replace-provider**`: Remove provider instances from the terraform.tfstate file.

30. `**terraform workspace new**`: Create a new workspace.

31. `**terraform workspace list**`: List workspaces.

32. `**terraform workspace show**`: Show the current workspace.

33. `**terraform workspace select**`: Select a workspace.

34. `**terraform workspace delete**`: Delete a workspace.

35. `**terraform providers**`: Prints a tree of the providers used in the configuration.

36. `**terraform graph**`: Create a visual representation of a Terraform configuration or execution plan.

Example of: **Multiple Resource Block**

resource "local\_file" "devops" {

filename = "/home/ubuntu/terraform/devopsautomated.txt"

content = "I want to be a rich citizen of USA"

}

resource "random\_string" "dollar" {

length = 24

special = true

override\_special = "!@#$%^&()\_+-=~:{}<>?,.;["

}

output "dollar" {

value = random\_string.dollar[\*].result

}

Example of: **Terraform Block**

terraform {

required\_providers {

docker = {

source = "kreuzwerker/docker"

version = "~> 2.21.0"

}

}

}

Example of: **Terraform Block**

terraform {

required\_providers {

docker = {

source = "kreuzwerker/docker"

version = "~> 2.21.0"

}

}

}

provider "docker" {}

terraform {

required\_providers {

docker = {

source = "kreuzwerker/docker"

version = "~> 2.21.0"

}

}

}

provider "docker" {}

resource "docker\_image" "nginx" {

name = "nginx:latest"

keep\_locally = false

}

resource "docker\_container" "nginx" {

image = docker\_image.nginx.latest

name = "nginx-tf"

ports {

internal = 80

external = 80

}

}

**Permission Command For Docker To Run Terraform File**

$ sudo chown $USER /var/run/docker.sock

**Variable Creation**

1. First Method

*main.tf*

resource "local\_file" "devops" {

filename = "/home/ubuntu/terraform-course/terraform-variable/devopstest.txt"

content = "I will become a rich citizen of USA"

}

resource "local\_file" "devops-var" {

filename = var.filename

content = var.content

}

output "devops\_trainer" {

value = var.devops\_trainer

}

*variable.tf*

variable "filename" {

default = "/home/ubuntu/terraform/terraform-variable/devops-automated.txt"

}

variable "content" {

default = "This is auto generated from a variable"

}

variable "devops\_trainer" {}

1. Second Method

$export TF\_VAR\_devops\_trainer=Iwillgotoamerica

**Data Types In Terraform**

*main.tf*

resource "local\_file" "devops" {

filename = "/home/ubuntu/terraform-course/terraform-variable/devopstest.txt"

content = var.content\_map[“content1”]

}

resource "local\_file" "devops-var" {

filename = var.filename

content = var.content\_map[“content2”]

}

output "devops\_trainer" {

value = var.devops\_trainer

}

*variable.tf*

variable "filename" {

default = "/home/ubuntu/terraform/terraform-variable/devops-automated.txt"

}

variable "content" {

default = "This is auto generated from a variable"

}

variable "devops\_trainer" {}

*variables.tf*

variable “filename” {

default = “/home/ubuntu/terraform-course/terraform-variable/devops.txt”

}

variable “content” {

default = “This is auto generated from a variable”

}

variable = “devopstrainer” {}

variable “content\_map” {

type = map

default = {

“content” = “I amit kothiyal”

“content” = “Data Engineer”

}

}

variable “file\_list” {

type = list

default = [“/home/ubuntu/terraform-course/terraform-variables/file\_1.txt”, “/home/ubuntu/terraform-course/terraform-variables/file\_2.txt”]

}

variable “aws\_ec2\_object” {

type = object({

name = string

instances = number

keys = list

ami = string

})

}

default = {

name = “test\_ec2\_instance”

instances = 4

keys = [“key1.pem”, “key2.pem”]

ami = “ubuntu-afed34”

}

}

*main.tf*

resource “local\_file” “devops” {

filename = “/home/ubuntu/terraform-course/terraform-variable/devops.txt”

content = var.content\_map[“content1”]

}

resource “local\_file” “devops-var” {

filename = var.filename

content = var.content\_map[“content2”]

}

output “devopsoptrainer” {

value = var. devopsoptrainer

}

output “aws\_ec2\_instances” {

value = var.aws\_ec2\_object.instances

}

$terraform plan

$terraform apply

*variables.tf*

variable “filename” {

default = “/home/ubuntu/terraform-course/terraform-variable/devops.txt”

}

variable “content” {

default = “This is auto generated from a variable”

}

variable = “devopstrainer” {}

variable “content\_map” {

type = map

default = {

“content” = “I amit kothiyal”

“content” = “Data Engineer”

}

}

variable “file\_list” {

type = list

default = [“/home/ubuntu/terraform-course/terraform-variables/file\_1.txt”, “/home/ubuntu/terraform-course/terraform-variables/file\_2.txt”]

}

variable “aws\_ec2\_object” {

type = object({

name = string

instances = number

keys = list

ami = string

})

}

default = {

name = “test\_ec2\_instance”

instances = 4

keys = [“key1.pem”, “key2.pem”]

ami = “ubuntu-afed34”

}

}

*main.tf*

resource “local\_file” “devops” {

filename = “/home/ubuntu/terraform-course/terraform-variable/devops.txt”

content = var.content\_map[“content1”]

}

resource “local\_file” “devops-var” {

filename = var.filename

content = var.content\_map[“content2”]

}

output “devopsoptrainer” {

value = var. devopsoptrainer

}

output “aws\_ec2\_instances” {

value = var.aws\_ec2\_object.instances

}

output “tf\_batch\_students\_isnt” {

value = var.no\_of\_students}

*vim prod.tfvars.json*

{

“batch” : 1,

“no\_of\_students” : 50

}

$terraform apply -var-file = prod.tfvar.json

If we apply

$rm terraform.tfstate

then backup file’ll be created. After that we can execute

$terraform apply

once again created same structure.

$export AWS\_ACCESS\_KEY\_ID = \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

$export AWS\_SECRET\_ACCESS\_KEY = \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*Terraform code of Configuration for AWS instance*

*vi main.tf*

terraform {

required\_providers {

aws = {

source = “hashicorp/aws”

version = “~> 4.16”

}

}

required\_version = “>= 1.2.0”

}

provider “aws” {

region = “us-west-2”

}

resource “aws\_instances” “my\_ec2\_instance” {

count = 5

ami = “ami-08c40ec9ead489470”

instance\_type = “t2.micro”

tags = {

Name = “TerraformTestServerInstance”

}

}

resource “aws\_s3\_bucket” “my\_s3\_bucket” {

bucket = “terraform-trainwithshubham-batch-1234”

tags = {

Name = “terraform-trainwithshubham-batch-1234”

Environment = “Dev”

}

}

output “ec2\_public\_ips” {

value = aws\_instance.my\_ec2\_instance[\*].public\_ip

}

$terraform destroy

*main.tf*

terraform {

required\_providers {

aws = {

source = “hashicorp/aws”

version = “~> 4.16”

}

}

required\_version = “>= 1.2.0”

}

provider “aws” {

region = “us-east-1”

}

locals {

instances = toset([“oregon", “washington", “california", “texas"])

}

resource “aws\_instance” “aws\_ec2\_test” {

for\_each = local.instances

ami = “ami-08c4ec9ead89470”

instance\_type = “t2.micro”

tags = {

Name = each.key

}

}

*main.tf*

terraform {

required\_providers {

aws = {

source = “hashicorp/aws”

version = “~> 4.16”

}

}

required\_version = “>= 1.2.0”

}

provider “aws” {

region = “us-east-1”

}

locals {

instances = {“oregon”:”ami-0b0dcb5067f052a63”, “california”:”ami-08c40ec9ead489470”, “washington”:”ami-08759edfd968eed”, “texas”:”ami-96ghjgowxlm56”}

}

resource “aws\_instance” “awsinstance” {

for\_each = local.instances

ami = each.value

instance\_type = “t2.micro”

tags = {

Name = each.key

}

}

---------------------------------------------------------------------------------------

**Remote Backend**

*resources.tf*

resource “aws\_s3\_bucket” “my\_s3\_bucket” {

bucket = “terraweek-demo-state-bucket”

}

resource “aws\_dynamodb\_table” “my\_dynamo\_table” {

name = “terrafweek-demo-state-table”

billing\_mode = “PAY\_PER\_REQUEST”

hash\_key = “LockID”

attribute {

name = “LockID”

type = “S”

}

}

*terraform.tf*

terraform {

required\_providers {

aws = {

source = “hashicorp/aws”

version = “5.29.0”

}

}

}

*providers.tf*

provider “aws” {

region = “us-west-2”

}

**State Locking**

terraform {

required\_providers {

aws = {

source = “hashicorp/aws”

version = “5.29.0”

}

}

backend “s3” {

bucket = “[iwanttogoamerica](https://s3.console.aws.amazon.com/s3/buckets/iwanttogoamerica?region=us-west-2)”

key = “terraform.tfstate”

region = “us-west-2”

dynamodb\_table = “terraformremotestateblank”

}

}

*main.tf*

provider “aws” {

region = “us-east-1”

}

resource “aws\_instance” “my\_ec2\_instance” {

ami = “ami-0efcece6bed30fd98”

instance\_type = “t2.micro”

tags = {

Name = “[iwanttogoamerica](https://s3.console.aws.amazon.com/s3/buckets/iwanttogoamerica?region=us-west-2)”

}

}

$terraform state list

$terraform state pull

**Demo of Resources Blocks**

*provider.tf*

provider “aws” {

region = “us-west-2”

}

#VPC Block

resource “aws\_vpc” “myvpc1” {

cidr\_block = “10.0.0.0/16”

instance\_tenancy = “default”

tags = {

Name = “main”

}

}

#IGW Block

resource “aws\_internet\_gateway” “gw1” {

vpc\_id = aws\_vpc.myvpc1.id

tags = {

Name = “main\_igw”

}

}

#Subnet Block

resource “aws\_subnet” “subnet1” {

vpc\_id = aws\_vpc.myvpc1.id

cidr\_block = “10.0.1.0/24”

tags = {

Name = “subnet1”

}

}

#route table block

resource “aws\_route\_table” “rt1” {

vpc\_id = aws\_vpc.myvpc1.id

route = []

tags = {

Name = “routetable1”

}

}

#route block

resource “aws\_route” “route1” {

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

destination\_cidr\_block = “0.0.0.0/0”

gateway\_id = aws\_internet\_gateway.gw1.id

depends\_on = [aws\_route\_table.rt1]

}

#route table association

resource “aws\_route\_table\_association” “association1” {

subnet\_id = aws\_subnet.subnet1.id

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

}

#security group block

resource “aws\_security\_group” “sg1” {

name = “sg1”

description = “Allow all inbound traffic security group”

vpc\_id = aws\_vpc.myvpc1.id

ingress { #for incoming traffic

description = “incoming from anywhere”

from\_port = 0

to\_port = 0

protocol = -1

cidr\_blocks = [“0.0.0.0/0”]

ipv6\_cidr\_blocks = null

}

egress { #For outgoing

from\_port = 0

to\_port = 0

protocol = “-1”

cidr\_blocks = [“0.0.0.0/0”]

ipv6\_cidr\_blocks = [“::/0”]

}

tags = {

Name = “sg1”

}

}

#EC2 block

resource “aws\_instance” “firstEC2” {

ami = “ami-a5fd5e5diheh”

instance\_type = “t2.micro”

tags = {

Name = “EC2usingTF”

}  
}

$terraform init

$terraform fmt

$terraform plan

$terraform apply

*provider.tf*

provider “aws” {

region = “us-east-1”

}

#Singleline Output

variable “singlelineinput” {

type = string

default = “Rajan Trivedi”

}

output “outputvalue1” {

value = “${var.singlelineinput}”

}

#Multiline Output

variable “multilineinput” {

type = string

default =<<EOH

yum update -y

yum install -y httpd.x86\_64

systemctl start httpd.service

systemctl enable httpd.service

echo “Hello World from $(hostname -f)” > /var/www/html/index.html

EOH

}

output “outputvalue2” {

value = var.multilineinput

}

#Map variable type block

variable “amiselection” {

type = map

default = {

“useast1” = “ami-0cff7528ff583bf9a”

“eucentral1” = “ami-0euf7528ff583bf9a”

}

}

output “mapoutput” {

value = “${var.amiselection[“useast1”]}”

}

#Variable list block

variable “environments” {

type = list

default = [“DataEngineer”, “DataScientist”, “RichUSACitizen”]

}

output “environmentop” {

value = “${var.environments[0]}”

}

$terraform plan

$terraform apply

**Demo of Data Blocks**

*provider.tf*

provider “aws” {

region = “us-east-1”

}

#Data block

Data “aws\_subnet” “newsubnet1” {

filter{

name = “tag:Name”

values = [“subnet1”]

}

}

#Create EC2 instance using above data block

resource “aws\_instance” “ec2usingdatablock” {

ami = “ami-0cff7528ff583bf9a”

instance\_type = “t2.micro”

subnet\_id = data.aws.subnet.newsubnet1.id

tags = {

Name = “ec2usingdatablock”

}

}

**Demo of creation of S3 bucket & an object to upload into S3 bucket.**

*provider.tf*

provider “aws” {

region = “us-east-1”

}

#S3 block

resource “aws\_s3\_bucket” “firsts3bucket” {

bucket = “firsts3bucketdemo”

tags = {

Name = “firsts3bucket”

Environment = “dev”

}

}

resource “aws\_s3\_bucket\_acl” “firsts3bucketacl” {

bucket = aws\_s3\_bucket. firsts3bucket.id

acl = “private”

}

#Object upload into S3 bucket

resource “aws\_s3\_bucket\_object” “firsts3object” {

bucket = aws\_s3\_bucket.firsts3bucket.id

key = “testdatafile.txt”

source = “./testdata.txt”

etag = filemd5(“./testdata.txt”)

}

**Demo1 of Terraform Modules**

*main.tf*

#Provider Block

provider “aws” {

region = “us-east-1”

}

#Module block for EC2

module “ec2\_module” {

source = “./ec2\_module” #To import the modules from *ec2\_module* directory in which *ec2\_module.tf* template file consist.

}

**Demo2 of Terraform Modules**

*main.tf*

provider “aws” {

region = “us-east-1”

}

#Module block for sg\_module -> Sending output

module “sg\_module” {

source = “./sg\_module” #To import the modules from *sg\_module* directory in which *sg\_module.tf* template file consist.

}

#Module block for ec2\_module -> Getting output

module “ec2\_module” {

sg\_id = “${module.sg\_module.sg\_id\_output}” #sg\_id\_output to be created under sg\_module directory in which *sg\_module.tf* template file consist.

source = “./ec2\_module”

}

*sg\_module.tf*

#Create SG & send send SG id to other module

variable “vpcid” {

type = string

default = “vpc-0db85478”

}

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

name = "allow\_tls"

description = "Allow TLS inbound traffic."

vpc\_id = “${var.vpcid}”

ingress {

description = “TLS from VPC”

from\_port = 443

to\_port = 443

ip\_protocol = "tcp"

cidr\_blocks =[“0.0.0.0/0”]

}

egress {

from\_port = 0

to\_port = 0

protocol = “-1”

cidr\_blocks =[“0.0.0.0/0”]

ipv6\_cidr\_blocks = [“::/0”]

}

tags = {

Name = “allow\_tls”

}

}

output “sg\_id\_output” {

value = “${aws\_security\_group.allow\_tls.id}”

}

*ec2\_module.tf*

#Create EC2 & get SG id value from sg module and use it inside this file

variable “amiid” {

default = “ami-0cff7528ff583bf9a”

}

variable “sg\_id” {} #It’ll accept value from other moduler or files

resource “aws\_instance” “web” {

ami = “${var.amiid}”

instance\_type = “t2.micro”

vpc\_security\_group\_ids = [“${var.sg\_id}”] #Reusing the variable value of sg\_id

tags = {

Name = “FirstEC2instancemodule”

}

}

**Demo of Terraform Workspace**

*main.tf*

provider “aws” {

region = “us-east-1”

}

#Module block for sg\_module -> Sending output

module “sg\_module” {

sg\_name = “sg\_for\_ec2\_${local.evn}”

source = “./sg\_module” #To import the modules from *sg\_module* directory in which *sg\_module.tf* template file consist.

}

#Module block for ec2\_first\_module -> Getting output

module “ec2\_first\_module” {

sg\_id = “${module.sg\_module.sg\_id\_output}” #sg\_id\_output to be created under sg\_module directory in which *sg\_module.tf* template file consist.

ec2\_name = “ec2\_instance\_${local.env}”

source = “./ec2\_module”

}

#Module block for ec2\_second\_module -> Getting output

module “ec2\_second\_module” {

sg\_id = “${module.sg\_module.sg\_id\_output}” #sg\_id\_output to be created under sg\_module directory in which *sg\_module.tf* template file consist.

ec2\_name = “ec2\_instance\_${local.env}”

source = “./ec2\_module”

}

locals {

env = “${terraform.workspace}” #It will return the currently selected environment value.

}

*sg\_module.tf*

#Create SG & send send SG id to other module

variable “vpcid” {

type = string

default = “vpc-0db85478”

}

variable “sg\_name” {} #Expects a value from other module or a file

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

name = "allow\_tls"

description = "Allow TLS inbound traffic."

vpc\_id = “${var.vpcid}”

ingress {

description = “TLS from VPC”

from\_port = 443

to\_port = 443

ip\_protocol = "tcp"

cidr\_blocks =[“0.0.0.0/0”]

}

egress {

from\_port = 0

to\_port = 0

protocol = “-1”

cidr\_blocks =[“0.0.0.0/0”]

ipv6\_cidr\_blocks = [“::/0”]

}

tags = {

Name = “allow\_tls”

}

}

output “sg\_id\_output” {

value = “${aws\_security\_group.allow\_tls.id}”

}

*ec2\_module.tf*

#Create EC2 & get SG id value from sg module and use it inside this file

variable “amiid” {

default = “ami-0cff7528ff583bf9a”

}

variable “sg\_id” {} #It’ll accept value from other moduler or files

variable “ec2\_name” {} #Expects the value of the ec2\_instance name from other module or file

resource “aws\_instance” “web” {

ami = “${var.amiid}”

instance\_type = “t2.micro”

vpc\_security\_group\_ids = [“${var.sg\_id}”] #Reusing the variable value of sg\_id

tags = {

Name = “${var.ec2\_name}”

}

}

**Project Demo of Terraform**

*./shared\_config/main.tf #Path of main.tf*

*main.tf*

#Provider block

provider “aws” {

region = “us-east-1”

}

#Network module block

module “network\_module” {

source = “./network\_module”

}

#loadbalancer module block

module “loadbalancer\_module” {

source = “./loadbalancer\_module”

publicsg\_id = “${module.network\_module.publicsg\_id}”

}

#autoscaling module block

module “autoscaling\_module” {

source = “${./autoscaling\_module}”

privatesg\_id = “${module.network\_module.privatesg\_id}”

tg\_arn = “${module.loadbalancer\_module.tg\_arn}”

}

*./shared\_config/main.tf #Path of shared\_config.tf*

*shared\_config.tf*

#All shared configurations for the project

#Output blocks

output “env\_suffix” {

value = “${local.env}”

}

output “vpcid” {

value = “${local.vpcid}

}

output “publicsubnetid” {

value = “${local.publicsubnetid\_env}”

}

output “privatesubnetid” {

value = “${local.privatesubnetid\_env}”

}

output “amiid” {

value = “${local.amiid\_env}”

}

output “instancetype” {

value = “${local.instancetype}”

}

output “keypair” {

value = “${local.keypair}”

}

#Local variable block

locals {

#Environmentid

env = “${terraform.workspace}” #Possible values of staging,

production

#vpcid

vpcid\_env {

default = “vpc-0db9343”

staging = “vpc-0db9343”

production = “vpc-0db9343”

}

vpcid = “${lookup(local.vpcid\_env.local.env)}”

#publicsubnetid

publicsubnetid\_env{

default = “subnet-3598746”

staging = “subnet-3598746”

production = “subnet-3598746”

}

publicsubnetid = “${lookup(local.publicsubnetid\_env.local.env)}”

#privatesubnetid

privatesubnetid\_env {

default = “subnet-c5518ee4”

staging = “subnet-c5518ee4”

production = “subnet-c5518ee4”

}

privatesubnetid = “${lookup(local.privatesubnetid\_env.local.env)}”

#amiid

amiid\_env {

default = “ami-090fa75af13c156b4”

staging = “ami-090fa75af13c156b4”

production = “ami-090fa75af13c156b4”

}

amiid = “${lookup(local.amiid\_env.local.env)}”

#instancetype

instancetype \_env {

default = “t2.micro”

staging = “t2.micro”

production = “t2.medium”

}

instancetype = “${lookup(local.instancetype \_env.local.env)}”

#keypair

keypair\_env = {

default = “staging\_tf\_key\_pair”

staging = “staging\_tf\_key\_pair”

production = “production\_tf\_key\_pair”

}

keypair = “${lookup(local.keypair \_env.local.env)}”

}

*./network\_module/network\_module.tf #Path of main.tf*

*network\_module.tf*

#Network relates resources module

#Security\_group\_public

resource “aws\_security\_group” “publicsg” {

name = “publicsg\_${module.shared\_config.evn\_suffix}”

description = “Public security group for ALB”

vpc\_id = “${module.shared\_config.vpcid}”

ingress {

description = “Inbound traffic for port 80”

from\_port = 80

to\_port = 80

protocol = “tcp”

cidr\_blocks = [“0.0.0.0/0”]

}

egress {

from\_port = 80

to\_port = 80

protocol = “-1”

cidr\_blocks = [“0.0.0.0/0”]

}

tags = {

Name = “public\_sg”

}

}

#Security\_group\_private

resource “aws\_security\_group” “privatesg” {

name = “privatesg\_${module.shared\_config.evn\_suffix}”

description = “Private security group for EC2”

vpc\_id = “${module.shared\_config.vpcid}”

ingress {

description = “Inbound traffic for port 80”

from\_port = 80

to\_port = 80

protocol = “tcp”

security\_group = [“${aws\_security\_group.publicsg.id}”]

}

egress {

from\_port = 80

to\_port = 80

protocol = “tcp”

cidr\_blocks = [“0.0.0.0/0”]

}

tags = {

Name = “private\_sg”

}

}

./loadbalancer\_module/loadbalancer\_module.tf #Path of file

*variable block*

variable publicsg\_id{} #expecting value from other module or file

#module block

module “shared\_config” {

source = “../shared\_config”

}

#Load balancer block

resource “aws\_lb” “sampleapp\_alb” {

name = “sampleapp-alb-${module.shared\_config.env\_suffix}”

internal = false #because app is internet facing

load\_balancer\_type = “application”

security\_groups = [“${var.publicsg\_id}”]

subnets = [“${module.shared\_config.publicsubnetid}”, “${module.shared\_config.publicsubnetid2}”]

enable\_deletion\_protection = true

tags = {

Environment = “${module.shared\_config.env\_suffix}”

}

}

#ALB target group block

resource “aws\_lb\_target\_group” “sampleap\_http\_tg” {

name = “sampleapp-http-tg${module.shared\_config.env\_suffix}”

port = 80

protocol = “HTTP”

vpc\_id = “${module.shared\_config.vpcid}”

}

#Listner block

resource “aws\_lb\_listener” “http\_listener\_80” {

load\_balancer\_arn = “${aws\_lb.sampleapp\_alb.arn}”

port = “80”

protocol = “HTTP”

default\_action {

type = “forward”

target\_group\_arn = aws\_lb\_target\_group.sampleapp\_http\_tg.arn

}

}

#Output block for tg\_arn

output “tg\_arn” {

value = “${aws\_lb\_target\_group.sampleapp\_http\_tg.arn}”

}

./autoscaling\_module/autoscaling\_module.tf #Path of file

*autoscaling\_module.tf*

#Module block

module “shared\_config” {

source = “../shared\_config”

}

#Local variable block

locals {

#Environment

env = “${terraform.workspace}” #Possible values = staging, production

#assgmin

asgmin\_env = {

default = “1”

staging = “1”

production = “1”

}

asgmin = “${lookup(local.asgmin\_evn.local.env)}”

#assgmax

asgmax\_env = {

default = “1”

staging = “1”

production = “1”

}

asgmax = “${lookup(local.asgmax\_evn.local.env)}”

#asgdesired

asgdesired\_env = {

default = “1”

staging = “1”

production = “1”

}

asgdesired = “${lookup(local.asgdesired\_evn.local.env)}”

}

#variable block

variable “privatesg” {} #Expecting input from other module or file

variable “tg\_arn” {} #Expecting input from other module or file

#Launch configuration block

resource “aws\_launch\_configuration” “sampleapp\_lc” {

name = “sampleapp\_lc${module.shared\_config.env\_suffix}”

image\_id = “${module.shared\_config.amiid}”

instance\_type = “${module.shared\_config.instancetype}”

key\_name = “${module.shared\_config.keypair}”

user\_data = “${file(“assets/userdata.txt”)}”

security\_groups = [“${var.privatesg.id}”]

}

#Autoscaling group block

resource “aws\_autoscaling\_group” “sampleapp\_asg” {

name = “sampleapp\_asg\_${module.shared\_config.env\_suffix}”

max\_size = “${local.asgmin}”

min\_size = “${local.asgmax}”

health\_check\_grace\_period = 300

desired\_capacity = “${local.asgmax}”

launch\_configuration = aws\_launch\_configuration.foobar.name

vpc\_zone\_identifier = [“${module.shared\_config.publicsubnetid1}”, “${module.shared\_config.publicsubnetid2}”]

target\_group\_arns = [“${var.tg\_arn}”]

tag {

key = “environment”

value = “${module.shared\_config.env\_suffix}”

propagate\_at\_launch = true

}

}

./assets/userdata.txt #Path of file

*userdata.txt*

#!/bin/bash

sudo su

yum update -y

yum install httpd php -y

sudo service httpd start