**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”)

}

**Demo 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.

}