Terraform LAB:

**Terraform Installation Windows :**

Download the terraform file

<https://developer.hashicorp.com/terraform/downloads>

go to the pc

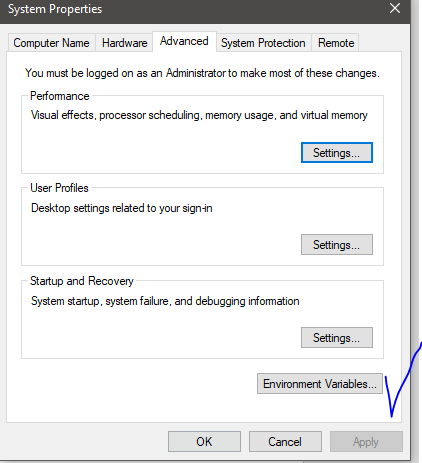
go to c folder

#Create a new folder

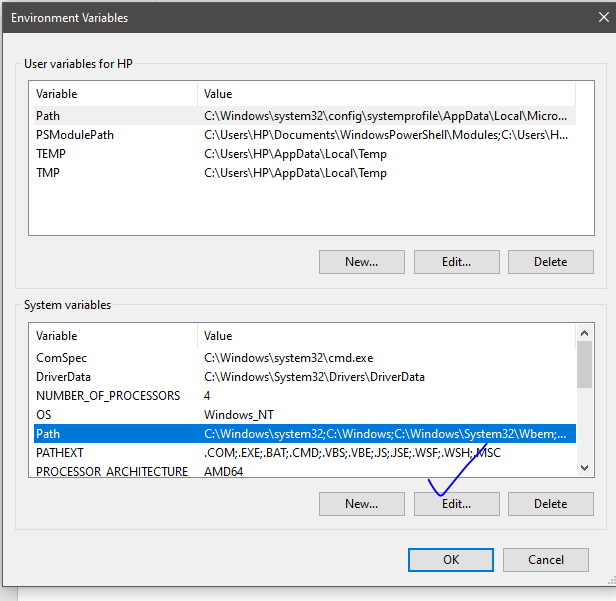
Software

Downloaded file ….Copy and paste the terraform file in softwares folder in local disk c

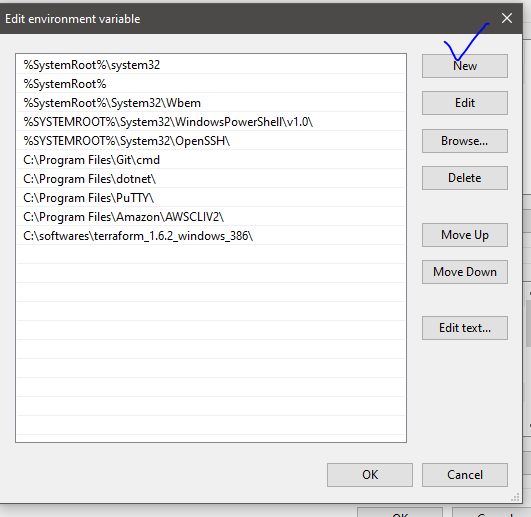
#widows search bar …search it edit the system environment variable



Click on environment variable



Click on Edit



Click on newrCopy and paste the terraform folder at the end forward \ enter example below

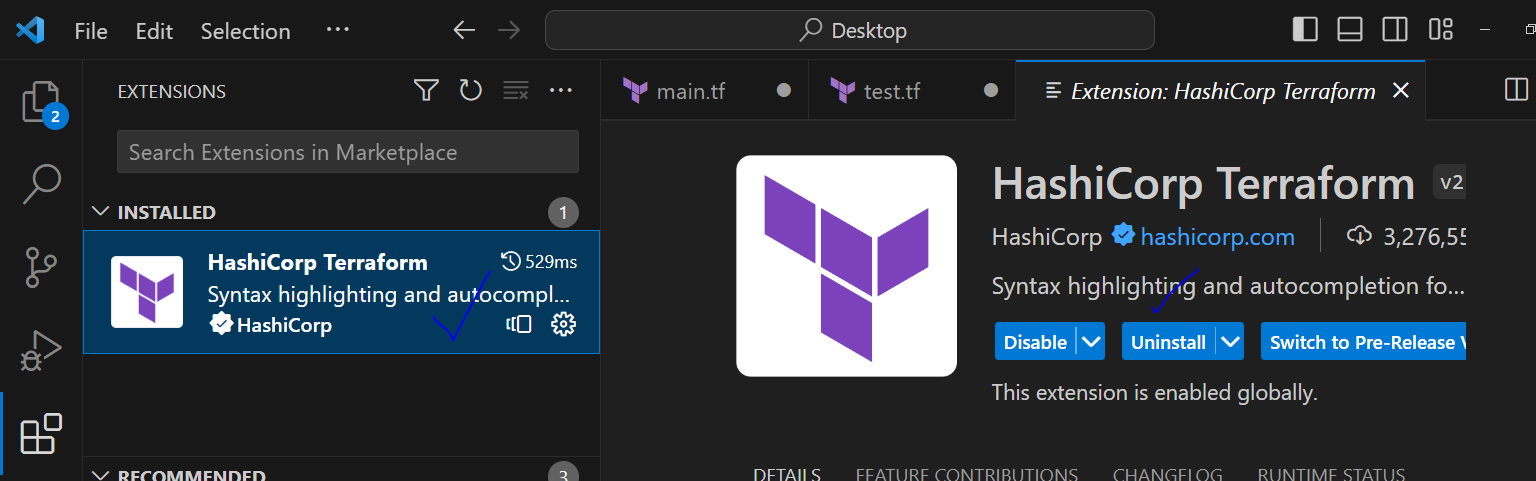
C:\softwares\terraform\_1.6.2\_windows\_386\

Go to cmd

Terraform –version type it

1Down load and install visual studio code Windows

2.go to visual studio…click on settings …Click on Extensions



Terrafrom Lab :

Example:

Local provder

terraform {

  required\_providers {

    local = {

        source = "hashicorp/local"

    }

  }

}

resource "local\_file" "demo" {

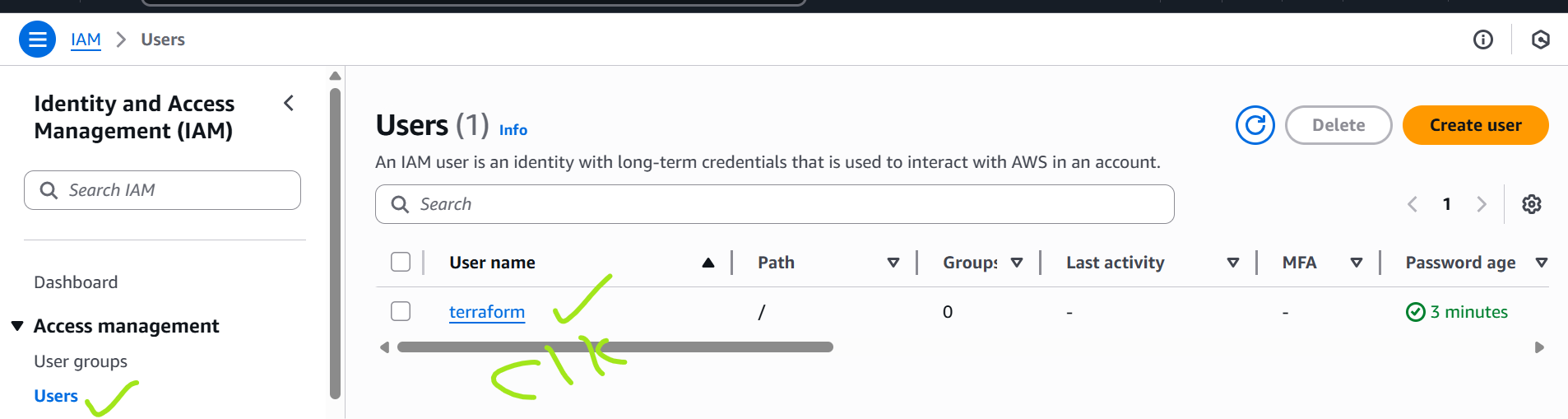
  filename = "sample.txt"

  content  = "This is terrafrom1"

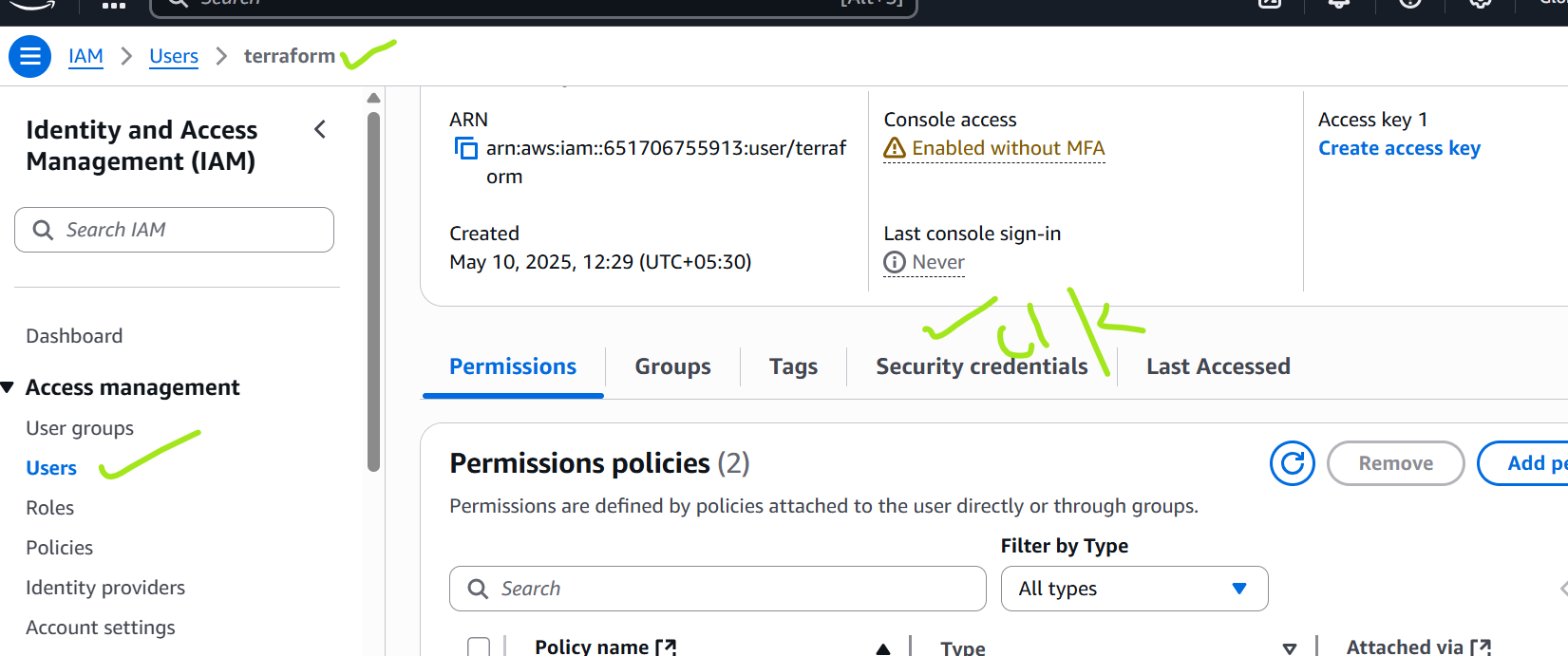
}

Terrafrom Aws provider

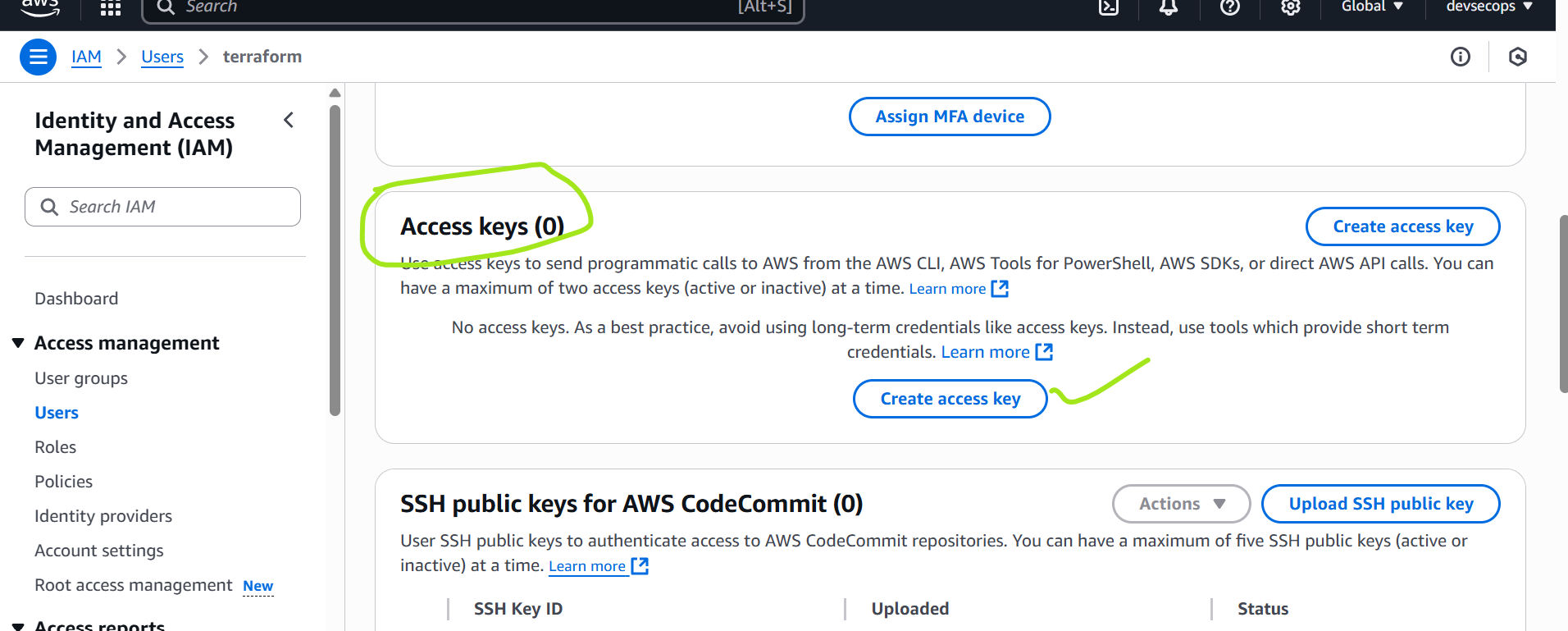
Terraform-ec2-sc3 lab and AWS IAM Credentials pass :

1. Go to IAM
2. Create user : ex terraform
3. Provide adminstraion permission
4. Download the credentials and store it
5. Go to the users in IAM in aws
6. 

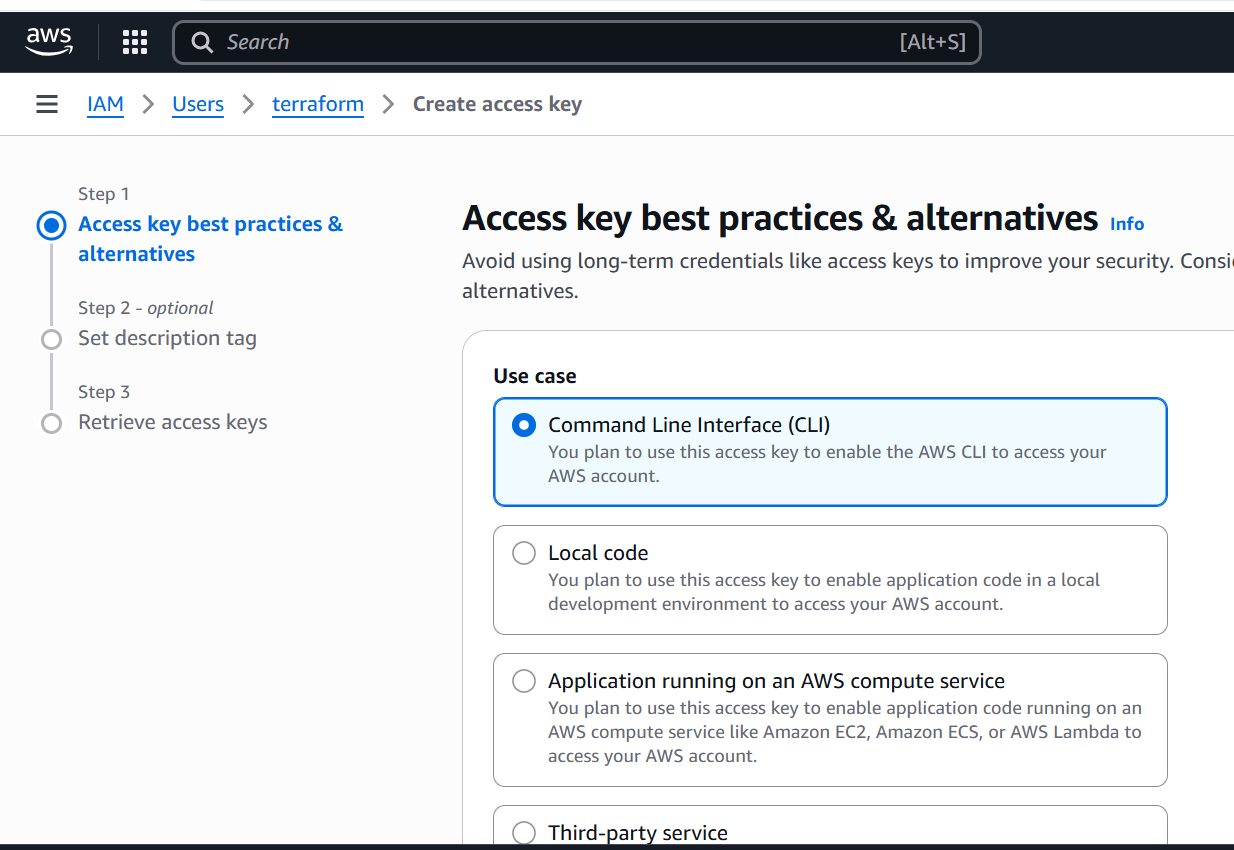
Click on user like tereafrom . Generate the credentials like access key and secret key to the particular user

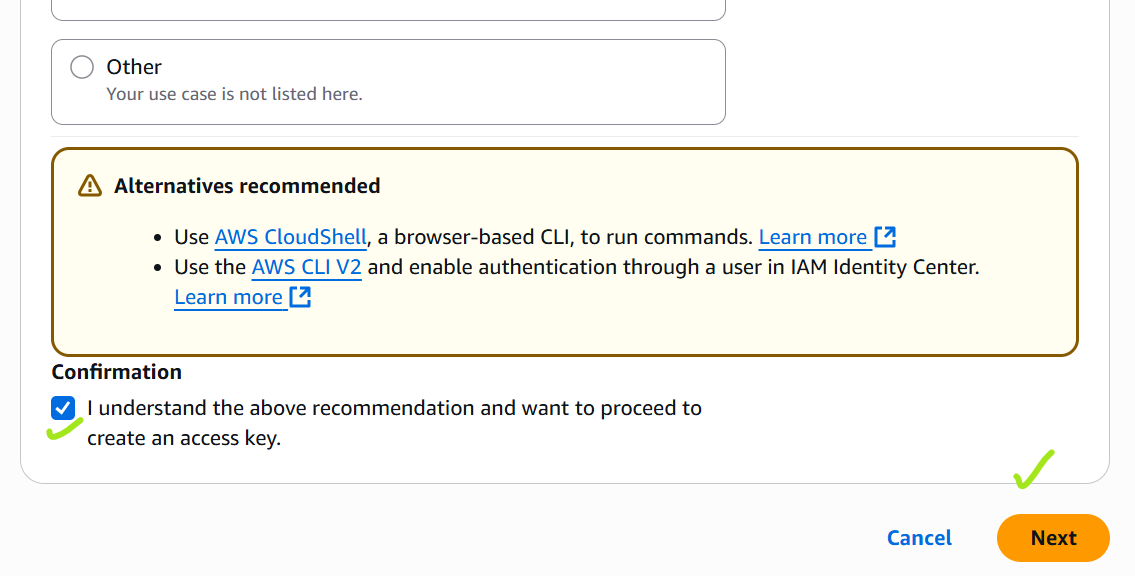
1. 

Click on security credentials

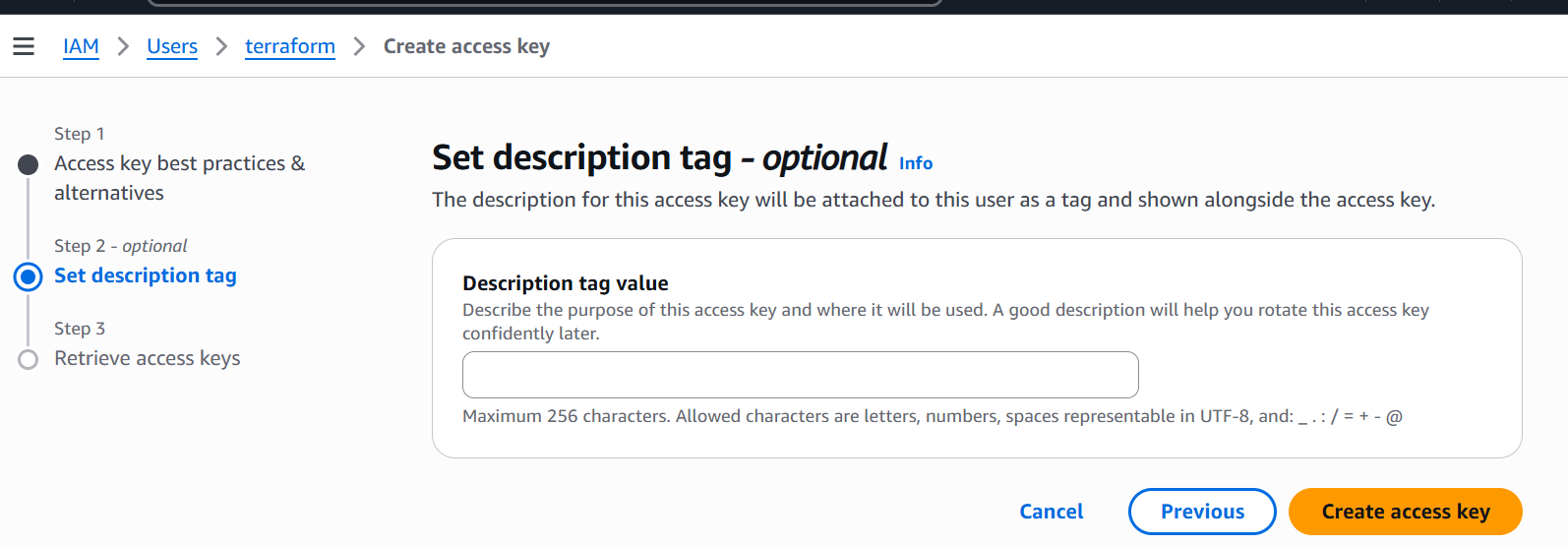


Click on create access key





Click on next



Click on create access key and download the credentials

#Terraform block Configuration like terraform version and

provider version

terraform {

#  required\_version = ">=1.3.0" #terraform Version

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 5.0" #provider Version like aws/azure/Gcp providers or if you don’t specify the version it will download latest version FYI

    }

  }

}

# Configure the AWS Provider

provider "aws" {

  region = "ap-south-1"

#  profile = "terraformprofile" #pass the profile name

# it you want pass the aws IAM role you can pass like these

/\*

   assume\_role {

    role\_arn     = "arn:aws:iam::123456789012:role/ROLE\_NAME"

}

\*/

#this one way to pass to the credentials aws iam user credentials  access\_key/secret\_key

/\*

  access\_key = "AKIA3XVUMBQA3F4KSXOP"

  secret\_key = "bhXioNO27Q2ufPTEq7q9XyGze5LnOfGvNe8Vn1U0"

\*/

#Another way of aws IAM user credentials using Environment Varibale

/\* In the linux terminal you canuser export below commands

export AWS\_ACCESS\_KEY\_ID="AKIA3XVUMBQA3F4KSXOP"

export AWS\_SECRET\_ACCESS\_KEY="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"

export AWS\_REGION="us-west-2"

In windows command prompt

Set AWS\_ACCESS\_KEY\_ID="AKIA3XVUMBQA3F4KSXOP"

Set AWS\_SECRET\_ACCESS\_KEY="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"

terraform plan

when the Terminal exits it will not work

\*/

/\*another way aws IAM user credetila usig aws configure command

install aws cli on terrform working instance/vm machine

#run these command pass access\_key&secret\_key and region, format

aws configure --profile "terraformprofile"

#check

cat .aws/credentials

cat .aws/config

shared\_config\_files = [“~/.aws/config”]

shared\_credentials\_files = [“/.aws/credentials”]

\*/

#if you aws credentials in different path use below syntax

provider "aws" {

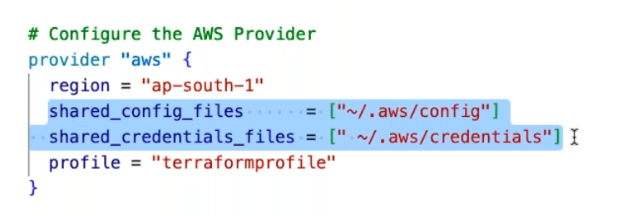
shared\_config\_files = ["/Users/tf\_user/.aws/conf"]

shared\_credentials\_files = ["/Users/tf\_user/.aws/creds"]

profile = "customprofile"

}

Or



  profile = "terraformprofile"

}

#resoucre block

resource "aws\_instance" "web" {

  ami           = "ami-05c0f5389589545b7"

  instance\_type = "t2.micro"

  availability\_zone = "ap-south-1a"

  count = "2"

#You can use multiple tags

  tags = {

    Name = "HelloWorld"

    Env = "SIT"

  }

}

#s3 resource block

resource "aws\_s3\_bucket" "web" {

  bucket = "aravindasameethabukcet-${count.index}" #Count indexing addedd the bucket like bukcet-0,bucket-2

  count = "2"

  tags = {

    Name = "web-images"

    Env  = "sit"

}

}

/\*

terraform  commands

terraform init --> initialize first it will install or download the prviders aws ,gcp and azure with the latest versions

terraform fmt -- >when you run terraform fmt it is arrange proper format of the code or it will set format of terraform code

terraform init –upgrade 🡪 >= 4.0 now your using ,Tommorow 5.1 is came that time you can use upgrade command or change the version >=5.0

terraform validate --> it will check the syntax and consistency

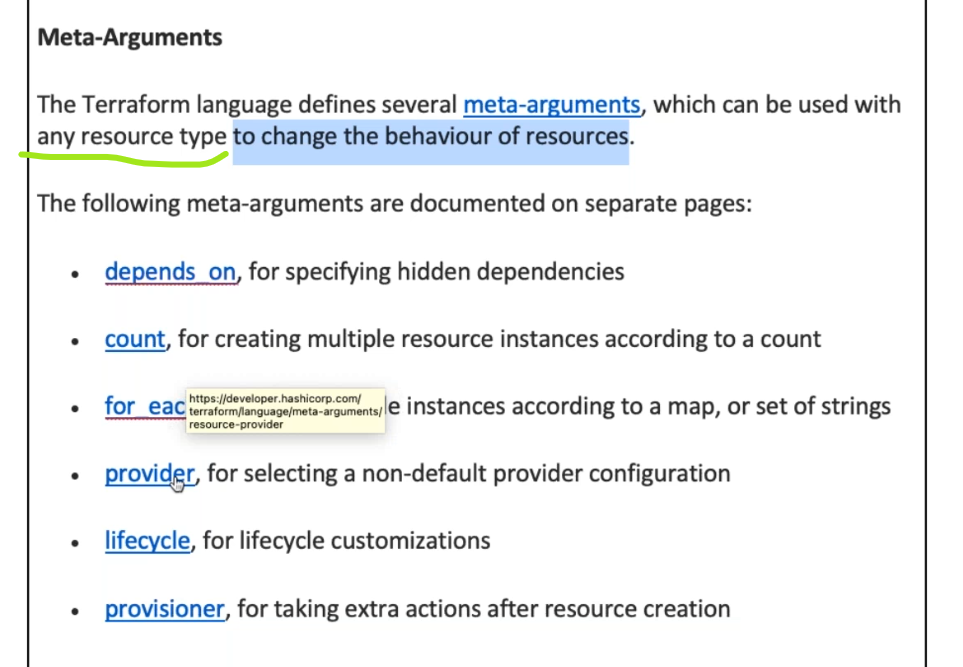
terraform plan --> show the changes before applying

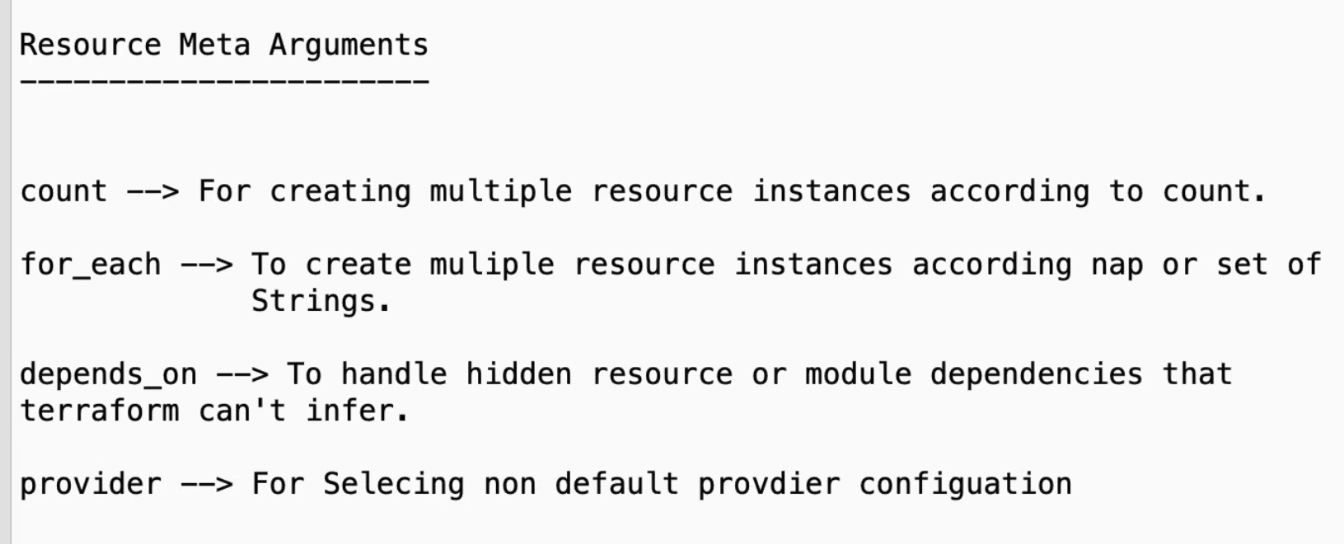
terraform apply --> apply changes it will ask yes

terraform apply --auto-approve --> it will not ask yes directly apply changes

\*/

#Meta arguments





Count and for each can’t be used both at a time or together in resource block

#Terraform block Configuration like terraform version and provider version

terraform {

  required\_providers {

    aws = {

      source = "hashicorp/aws"

      #      version = "~> 5.0"

    }

  }

}

# Configure the AWS Provider

provider "aws" {

  region  = "ap-south-1"

  profile = "terraformprofile"

}

resource "aws\_instance" "example" {

  ami           = "ami-062f0cc54dbfd8ef1"

  instance\_type = "t2.micro"

  count         = "2"

}

#Meta arguments pass here

resource "aws\_s3\_bucket" "demobucket" {

  bucket = "walmartbackupdb-${count.index}"

  count  = "2"

}

Or



[root@all-tools-master meta-arguments]# cat main.txt

#Terraform block Configuration like terraform version and provider version

terraform {

  #  required\_version = ">=1.3.0" #terraform Version

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 5.0" #provider Version like aws/azure/Gcp providers

    }

  }

}

# Configure the AWS Provider

provider "aws" {

  region  = "ap-south-1"

  profile = "terraformprofile" #pass the profile name

}

#Data Block get the information from the aws

data "aws\_ami" "this" {

  most\_recent = true

  owners      = ["amazon"]

  filter {

    name   = "name"

    values = ["RHEL-9.2.0\_HVM-20230905-x86\_64-\*"]

  }

  filter {

    name   = "root-device-type"

    values = ["ebs"]

  }

  filter {

    name   = "virtualization-type"

    values = ["hvm"]

  }

}

#output Block --display the output using output block

output "redhatlatestid" {

  value = data.aws\_ami.this.id

}

#resource block

resource "aws\_instance" "app" {

  ami           = data.aws\_ami.this.id

  instance\_type = "t2.micro"

  count         = 5

  tags = {

    "Name" = "App-${count.index}"

  }

}

Random

 ##

[root@all-tools-master meta-arguments]# cat random.tf

#Terraform block Configuration like terraform version and provider version

terraform {

#  required\_version = ">=1.3.0" #terraform Version

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 5.0" #provider Version like aws/azure/Gcp providers

    }

    random = {

      source = "hashicorp/random"

      version = "3.5.1"

    }

  }

}

# Configure the AWS Provider

provider "aws" {

  region = "ap-south-1"

  profile = "terraformprofile" #pass the profile name

}

#

#resource block

resource "random\_string" "random" {

  length           = 16

  count            = 2

  special          = false

  upper = false

#  count = 2

}

resource "aws\_s3\_bucket" "example" {

  bucket = "sai12345-bucket-${random\_string.random[count.index].id}"

  tags = {

    Name        = "My bucket"

    Environment = "Dev"

  }

}

Or

terraform {

#  required\_version = ">=1.3.0" #terraform Version

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 5.0" #provider Version like aws/azure/Gcp providers

    }

    random = {

      source = "hashicorp/random"

      version = "3.5.1"

    }

  }

}

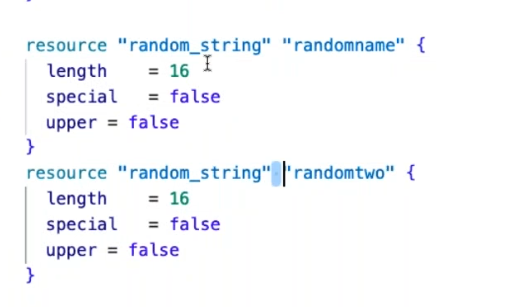
# Configure the AWS Provider

provider "aws" {

  region = "ap-south-1"

  profile = "terraformprofile" #pass the profile name

}





Create IAM user in aws :

Links of terraform modules ,Meta arguments ,variable declaration, data blocks,

<https://developer.hashicorp.com/terraform/language/meta-arguments/module-providers>

Create one file providers.tf terraform version, provider version, regions declare

Main.tf file resource block can declare

Example :

Vim provider.tf

#Terraform block Configuration like terraform version and provider version

terraform {

  #  required\_version = ">=1.3.0" #terraform Version

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 5.0" #provider Version like aws/azure/Gcp providers

    }

  }

}

# Configure the AWS Provider

provider "aws" {

  region  = "ap-south-1"

  profile = "terraformprofile" #pass the profile name

}

Vim main.tf

 resource "aws\_iam\_user" "iamusersainath" {

  name = "sainath"

  tags = {

    name = "sainath"

  }

}

resource "aws\_iam\_user" "iamusersainath1" {

  name = "sainath1"

  tags = {

    name = "sainath1"

  }

}

resource "aws\_iam\_user" "iamusersainath2" {

  name = "sainath2"

  tags = {

    name = "sainath1"

  }

}

Terraform init 🡪initaailse

Terraform fmt -- >format the code

Terraform validate 🡪syntaxt check

Terraform plan and apply

Meta arguments using for\_each strings

#Terraform block Configuration like terraform version and provider version

terraform {

#  required\_version = ">=1.3.0" #terraform Version

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 5.0" #provider Version like aws/azure/Gcp providers

    }

  }

}

# Configure the AWS Provider

provider "aws" {

  region = "ap-south-1"

  profile = "terraformprofile" #pass the profile name

}

resource "aws\_iam\_user" "iamusersainath" {

#Set of string usin for\_each arguments

  for\_each = toset (["sainath","sainath1","sainath2"])

  name = each.key

  tags = {

    "Name" = each.key

  }

}

For\_each Map key value pair :

 #Terraform block Configuration like terraform version and provider version

terraform {

#  required\_version = ">=1.3.0" #terraform Version

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 5.0" #provider Version like aws/azure/Gcp providers

    }

  }

}

# Configure the AWS Provider

provider "aws" {

  region = "ap-south-1"

  profile = "terraformprofile" #pass the profile name

}

resource "aws\_iam\_user" "iamusersainath" {

#Set of string usin for\_each arguments

  for\_each = toset(["sainath","sainath1","sainath2"])

  name = each.key

  tags = {

    "Name" = each.key

  }

}

#Map is like key value pairs

/\*

example:

for\_each = {

    "key" = "value"

 }

 \*/

resource "aws\_s3\_bucket" "example" {

  for\_each = {

    dev = "my-tf-dev-bucket"

    sit = "my-tf-test-bucket"

    prod = "my-tf-prod-bucket"

}

  bucket = "${each.key}-${each.value}"

  tags = {

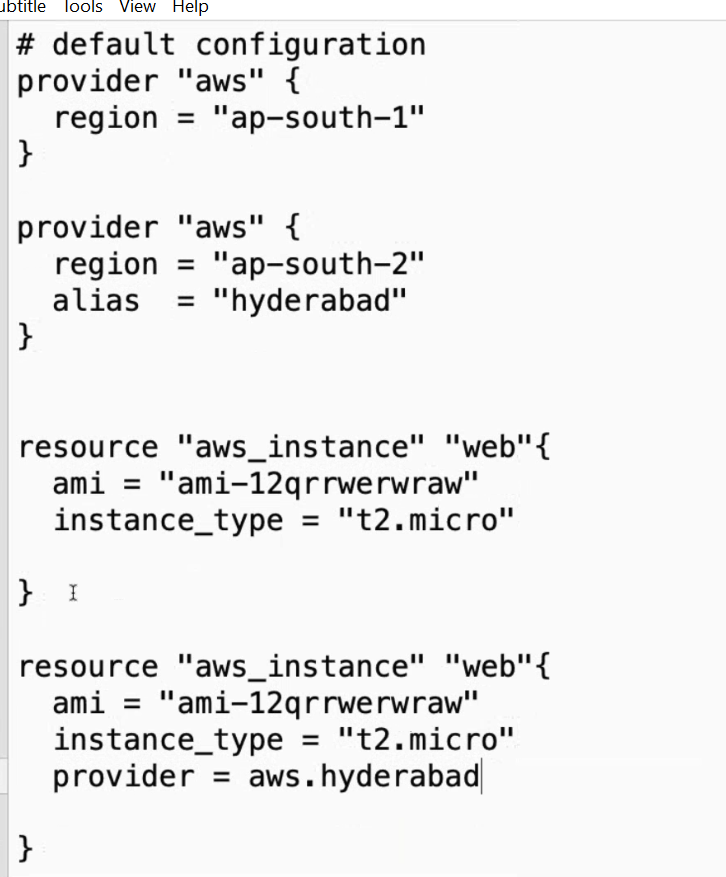
    Name        = "${each.key}-${each.value}"

    Environment = "${each.key}"

  }

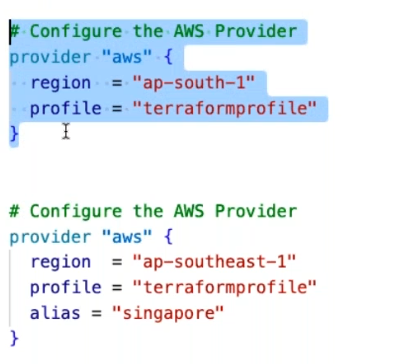
}

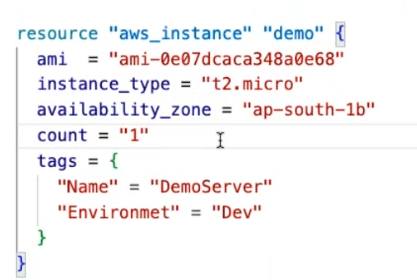
Provider arguments declaration :



Or

Or







30th video :VPC creation Without using variables

resource "aws\_vpc" "demovpc" {

  cidr\_block           = "192.168.0.0/24"

  instance\_tenancy     = "default"

  enable\_dns\_support   = true

  enable\_dns\_hostnames = true

  tags = {

    Name = "demovpc"

  }

}

#public subnet 1a

resource "aws\_subnet" "publicsubnet1a" {

  vpc\_id                  = aws\_vpc.demovpc.id

  cidr\_block              = "192.168.0.0/26"

  availability\_zone       = "ap-south-1a"

  map\_public\_ip\_on\_launch = true

  tags = {

    Name = "demovpc-publicsubnet-1a"

  }

}

#private subnet 1a

resource "aws\_subnet" "privatesubnet1a" {

  vpc\_id            = aws\_vpc.demovpc.id

  cidr\_block        = "192.168.0.64/26"

  availability\_zone = "ap-south-1a"

  tags = {

    Name = "demovpc-privatesubnet-1a"

  }

}

#public subnet 1b

resource "aws\_subnet" "publicsubnet1b" {

  vpc\_id                  = aws\_vpc.demovpc.id

  cidr\_block              = "192.168.0.128/26"

  availability\_zone       = "ap-south-1b"

  map\_public\_ip\_on\_launch = true

  tags = {

    Name = "demovpc-publicsubnet-1b"

  }

}

#private subnet 1b

resource "aws\_subnet" "privatesubnet1b" {

  vpc\_id            = aws\_vpc.demovpc.id

  cidr\_block        = "192.168.0.192/26"

  availability\_zone = "ap-south-1b"

  tags = {

    Name = "demovpc-privatesubnet1b"

  }

}

#internet gateway

resource "aws\_internet\_gateway" "demovpc-igw" {

  vpc\_id = aws\_vpc.demovpc.id #Internet gateway associated with vpc

  tags = {

    Name = "demovpc-igw"

  }

}

#Creaate Route table

resource "aws\_route\_table" "demovpcpublicrt" {

  vpc\_id = aws\_vpc.demovpc.id

  route {

    cidr\_block = "0.0.0.0/0"

    gateway\_id = aws\_internet\_gateway.demovpc-igw.id

  }

  tags = {

    Name = "demovpc-publicrt"

  }

}

#Route table association with public subnets 1a and 1b

resource "aws\_route\_table\_association" "demovpcigwassociation1a" {

  subnet\_id      = aws\_subnet.publicsubnet1a.id

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

}

resource "aws\_route\_table\_association" "demovpcigwassociation1b" {

  subnet\_id      = aws\_subnet.publicsubnet1b.id

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

}

#Create EIP public 1a

resource "aws\_eip" "eipone" {

  vpc = true

  tags = {

    Name = "ElasticipNAT"

  }

}

#Create NAT gateway 1a

resource "aws\_nat\_gateway" "demovpcnat1a" {

  allocation\_id = aws\_eip.eipone.id

  subnet\_id     = aws\_subnet.publicsubnet1a.id

  tags = {

    Name = "Natgateway1a"

  }

  depends\_on = [aws\_internet\_gateway.demovpc-igw]

}

#Create EIP public 1b

resource "aws\_eip" "eipfornat-two" {

  vpc = true

  tags = {

    Name = "ElasticipNAT"

  }

  depends\_on = [aws\_internet\_gateway.demovpc-igw]

}

#Create NAT gateway 1b

resource "aws\_nat\_gateway" "demovpcnat1b" {

  allocation\_id = aws\_eip.eipfornat-two.id

  subnet\_id     = aws\_subnet.publicsubnet1b.id

  tags = {

    Name = "Natgateway1b"

  }

  depends\_on = [aws\_internet\_gateway.demovpc-igw]

}

#Creaate private Route table

resource "aws\_route\_table" "privateroutetable1a" {

  vpc\_id = aws\_vpc.demovpc.id

  route {

    cidr\_block     = "0.0.0.0/0"

    nat\_gateway\_id = aws\_nat\_gateway.demovpcnat1a.id

  }

  tags = {

    Name = "demovpc-privatert1a"

  }

}

#Route table association with private subnets 1a and 1b

resource "aws\_route\_table\_association" "demovpcnatassociation1a" {

  subnet\_id      = aws\_subnet.privatesubnet1a.id

  route\_table\_id = aws\_route\_table.privateroutetable1a.id

}

resource "aws\_route\_table" "privateroutetable1b" {

  vpc\_id = aws\_vpc.demovpc.id

  route {

    cidr\_block     = "0.0.0.0/0"

    nat\_gateway\_id = aws\_nat\_gateway.demovpcnat1b.id

  }

  tags = {

    Name = "demovpc-privatert1b"

  }

}

#Route table association with private subnets  1b

resource "aws\_route\_table\_association" "demovpcnatassociation1b" {

  subnet\_id      = aws\_subnet.privatesubnet1b.id

  route\_table\_id = aws\_route\_table.privateroutetable1b.id

}

31st videos variables

Variables :

Input variable block

Output variable block

Local variable block

Local Variable block Lab :

Local values 🡪 a local values assigns name to an expression (value). You can use the name multiple time within that module

This is helpful to avoid repetating same values or expression multiple times in a configuration

#Terraform block Configuration like terraform version and provider version

terraform {

#  required\_version = ">=1.3.0" #terraform Version

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 5.0" #provider Version like aws/azure/Gcp providers

    }

  }

}

# Configure the AWS Provider

provider "aws" {

  region = "ap-south-1"

  profile = "terraformprofile" #pass the profile name

}

#Local Variable Declaration

locals {

  instancetypemicro = "t2.micro"

  instancetypemicro-t3med = "t3.medium"

  availabilityzone1a = "ap-south-1a"

  availabilityzone1b = "ap-south-1b"

  tags = {

    "Env" = "dev"

    "Name" = "demo"

}

}

resource "aws\_instance" "web" {

  ami           = "ami-05c0f5389589545b7"

  instance\_type = "local.instancetypemicro"

  availability\_zone = "local.availabilityzone1a"

  tags = local.tags

}

resource "aws\_s3\_bucket" "web" {

  bucket = "aravindasameethabukcet-1"

  tags = local.tags

}

#repeting the same local varibales with in that module

#example below same tags same instance type

resource "aws\_instance" "demo-ec2" {

  ami           = "ami-05c0f5389589545b7"

  instance\_type = "loca.instancetypemicro-t3med"

  availability\_zone = "local.availabilityzone1b"

  tags = local.tags

}

Input Variable :

Input variable are used as parameter to provide values at runtime to customize our deployments

We can use these input variable to pass certain values from outside of the configuration or module

Each input variable must be declared using variable block

Variables “<lable>” {

}

Each Variable block we are defining can have some arguments .

1. Type
2. String
3. Number
4. Bool
5. List
6. Set
7. Map

Description

Default

 #Terraform block Configuration like terraform version and provider version

terraform {

#  required\_version = ">=1.3.0" #terraform Version

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 5.0" #provider Version like aws/azure/Gcp providers

    }

  }

}

# Configure the AWS Provider

provider "aws" {

  region = "ap-south-1"

  profile = "terraformprofile" #pass the profile name

}

#input Variable Declaration

variable "amiid" {

  type = string

  description = "this ami id to be use for ec2 instance"

  default = "ami-05c0f5389589545b7"

}

variable "ec2count" {

  type = number

  description = "this count to be use for ec2 instance"

  default = "2"

}

resource "aws\_instance" "web" {

  ami           = var.amiid

  instance\_type = "t2.micro"

  availability\_zone = "ap-south-1"

  count = var.ec2count

  tags = {

    "Env" = "dev"

    "Name" = "Test"

}

}

#if You want to update on terminal level

terraform plan -var="ec2count=4" -var="amiid=ami-05c0f5389589545b7"

terraform apply -var="ec2count=4" -var="amiid=ami-05c0f5389589545b7"

Terraform format – it will set format of terraform code

Terraform init

Terraform validate

Terraform plan

Terraform apply --auto-approve

Terraform destroy –auto-approve

32 .Videos : variables explanation

Types of varaibales :

Primitive type or simple types

String 🡪 A squenece of character . ex: “hello” ,”ami-id-234df” or “ap-south-1”

Number 🡪 A numerica value . 15 or 20 or 2 or 6.2

Bool 🡪either true or false

Complex types

List

Set

Map

object

local variables :

 #Local Variable Declaration

locals {

  instancetypemicro = "t2.micro"

  instancetypemicro-t3med = "t3.medium"

  availabilityzone1a = "ap-south-1a"

  availabilityzone1b = "ap-south-1b"

  tags = {

    "Env" = "dev"

    "Name" = "demo"

}

}

resource "aws\_instance" "web" {

  ami           = "ami-05c0f5389589545b7"

  instance\_type = "local.instancetypemicro"

  availability\_zone = "local.availabilityzone1a"

  tags = local.tags

}

resource "aws\_s3\_bucket" "web" {

  bucket = "aravindasameethabukcet-1"

  tags = local.tags

}

#repeting the same local varibales with in that module

#example below same tags same instance type

resource "aws\_instance" "demo-ec2" {

  ami           = "ami-05c0f5389589545b7"

  instance\_type = "loca.instancetypemicro-t3med"

  availability\_zone = "local.availabilityzone1b"

  tags = local.tags

}

Input variables :

#input Variable Declaration

variable "amiid" {

  type = string

  description = "this ami id to be use for ec2 instance"

  default = "ami-05c0f5389589545b7"

}

variable "ec2count" {

  type = number

  description = "this count to be use for ec2 instance"

  default = "2"

}

variable "instancetype" {

  type = string

  description = "this count to be use for ec2 instance"

  default = "t2.micro"

}

variable "availabilityzone" {

  type = string

  description = "this count to be use for ec2 instance"

  default = "ap-south-1a"

}

resource "aws\_instance" "web" {

  ami           = var.amiid

  instance\_type = var.instancetype

  availability\_zone = var.availabilityzone

  count = var.ec2count

  tags = {

    "Env" = "dev"

    "Name" = "Test"

}

}

/\*

terraform plan -var="ec2count=4" -var="amiid=ami-05c0f5389589545b7"

terraform apply -var="ec2count=4" -var="amiid=ami-05c0f5389589545b7"

in command line you can override the variables or arguments

Terraform format – it will set format of terraform code

Terraform init

Terraform validate

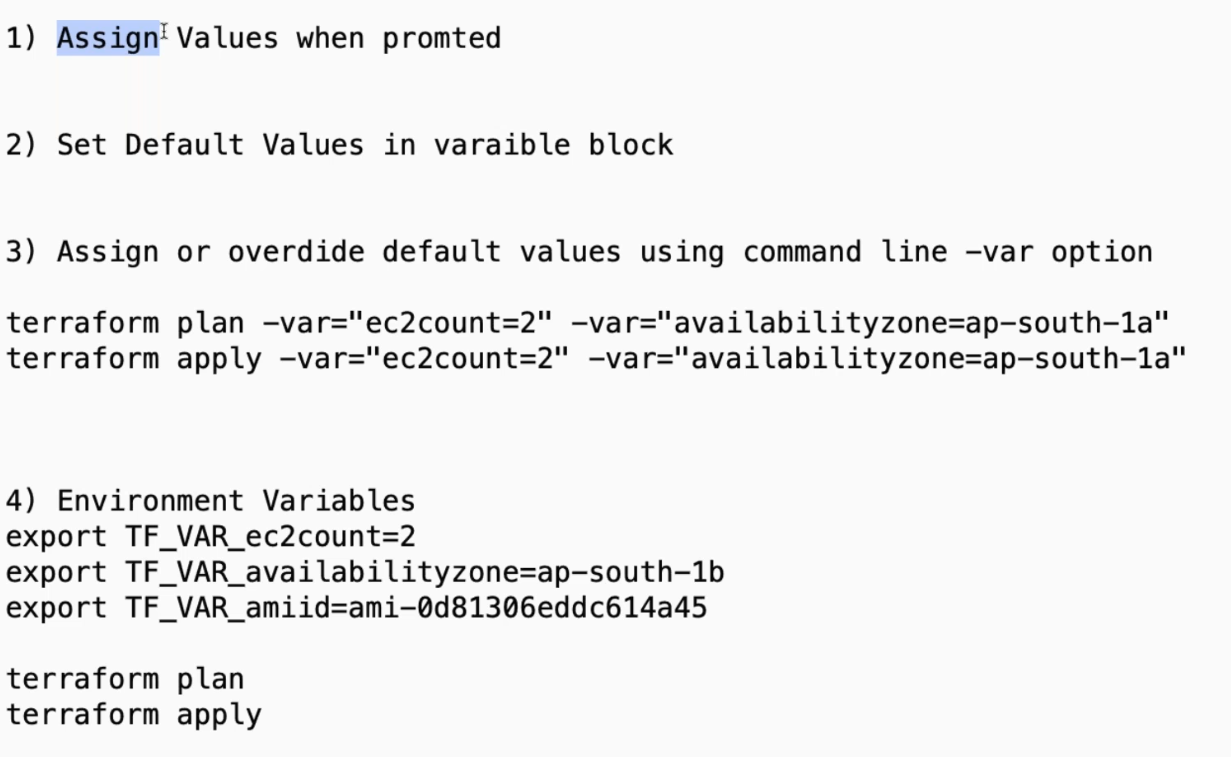
Terraform plan

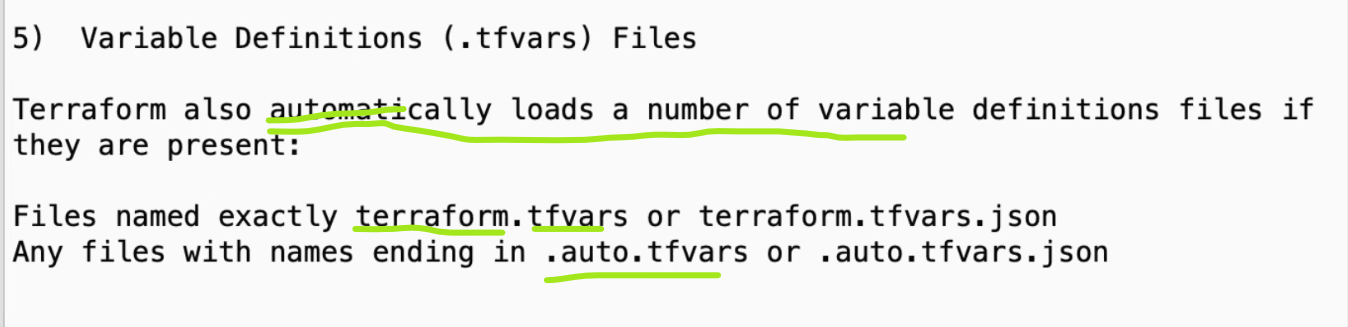
Terraform apply  --auto-approve

Terraform destroy –auto-approve

\*/

33 .tfvars and auto.tfvars





5) another way variables can be declare in one file filenaming convetion like ex : .tfvars or .tfvars.json

Terraform.tfvars default file

Vim terraform.tfvars

amiid            = "ami-05c0f5389589545b7"

ec2count         = "5"

instancetype     = "t2.medium"

availabilityzone = "ap-south-1a"

vim provider.tf

terraform {

  #  required\_version = ">=1.3.0" #terraform Version

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 5.0" #provider Version like aws/azure/Gcp providers

    }

  }

}

# Configure the AWS Provider

provider "aws" {

  region  = "ap-south-1"

  profile = "terraformprofile" #pass the profile name

}

Vim resource.tf or main.tf

#input Variable Declaration

variable "amiid" {

  type = string

  description = "this ami id to be use for ec2 instance"

  default = "ami-05c0f5389589545b7"

}

variable "ec2count" {

  type = number

  description = "this count to be use for ec2 instance"

  default = "2"

}

variable "instancetype" {

  type = string

  description = "this count to be use for ec2 instance"

  default = "t2.micro"

}

variable "availabilityzone" {

  type = string

  description = "this count to be use for ec2 instance"

  default = "ap-south-1a"

}

resource "aws\_instance" "web" {

  ami           = var.amiid

  instance\_type = var.instancetype

  availability\_zone = var.availabilityzone

  count = var.ec2count

  tags = {

    "Env" = "dev"

    "Name" = "Test"

}

}

Specifty the file in command line

Terraform apply -var-file=”dev.tfvars” 🡪filename is different execute these –var-file=”dev.tfvars” or “prod.tfvars”

* Any files with names ending in .auto.tfvars or .auto.tfvars.json.

Dev.auto.tfvars

Sit.auto.tfvars

Prod.auto.tfvars

No need to execute -var-file in command line below like these

Terraform apply -var-file=”dev.tfvars” 🡪filename is different execute these –var-file=”dev.tfvars” or “prod.tfvars”

34th video : video :same expaltion about tfvars and auto.tfvars

35 video :Complex type :

List

Set

Map

List : list will allow the duplicate values

Vim provider.tf

terraform {

  #  required\_version = ">=1.3.0" #terraform Version

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 5.0" #provider Version like aws/azure/Gcp providers

    }

  }

}

# Configure the AWS Provider

provider "aws" {

  region  = "ap-south-1"

  profile = "terraformprofile" #pass the profile name

}

Vim main.tf or resource.tf

variable "users" {

    type = list(string)

    description = "IAM users list"

}

resource "aws\_iam\_user" "iamuser" {

    for\_each = var.users

    name = each.value

}

Vim terraform.tfvars

users = ["sai", "test", "lord"]

Toset 🡪complext type

Vim terraform.tf

#Terraform block Configuration like terraform version and provider version

terraform {

  #  required\_version = ">=1.3.0" #terraform Version

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 5.0" #provider Version like aws/azure/Gcp providers

    }

  }

}

# Configure the AWS Provider

provider "aws" {

  region  = "ap-south-1"

  profile = "terraformprofile" #pass the profile name

}

Vim main.tf

variable "users" {

    type = list(string)

    description = "IAM users list"

}

resource "aws\_iam\_user" "iamuser" {

    for\_each = toset(["sai","sample","lord"])

    name = each.value

}

Vim terraform.tfvars

users = [ "sainath","test","lord" ]

Terraform plan

Terraform apply

35 videos: vpc LAB

Vim terraform.tf

terraform {

  #  required\_version = ">=1.3.0" #terraform Version

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 5.0" #provider Version like aws/azure/Gcp providers

    }

  }

}

# Configure the AWS Provider

provider "aws" {

  region  = "ap-south-1"

  profile = "terraformprofile" #pass the profile name

}

Vim variable.tf

#Vpc CIDR

variable "vpccidr" {

  type        = string

  default     = "192.168.0.0/16"

  description = "CIDR for vpc"

}

#PublicSubnet CIDR

variable "publicssubnetcidr" {

  type    = list(string)

  default = ["192.168.1.0/24"]

  description = "Public subnet CIDR "

}

#Private Subnet CIDR

variable "privatesubnetcidr" {

  type    = list(string)

  default = ["192.168.2.0/24"]

  description = "Priavte subnet CIDR "

}

variable "commontags" {

  type = map(string)

  default = {

    "Name" = "demo"

    "Env"  = "Dev"

  }

}

variable "enable\_dns\_support" {

  type    = bool

  default = true

}

variable "availability\_zones" {

  type    = list(string)

  default = ["ap-south-1a", "ap-south-1b"]

}

Vim main.tf

 resource "aws\_vpc" "demovpc" {

  cidr\_block           = var.vpccidr

  enable\_dns\_support   = true

  enable\_dns\_hostnames = true

  #  enable\_dns\_hostnames = true

  tags = var.commontags

}

#Public subnet cidr

resource "aws\_subnet" "demovpc-PublicSubnet" {

  vpc\_id            = aws\_vpc.demovpc.id

  count             = length(var.publicssubnetcidr)

  cidr\_block        = element(var.publicssubnetcidr, count.index)

  availability\_zone = element(var.availability\_zones, count.index)

  tags = merge(var.commontags,

    {

      "Name" = "demovpc-publicsubnet-${count.index}"

    }

  )

}

#private subnet CIDr

resource "aws\_subnet" "demovpc-privateSubnet" {

  vpc\_id            = aws\_vpc.demovpc.id

  count             = length(var.privatesubnetcidr)

  cidr\_block        = element(var.privatesubnetcidr, count.index)

  availability\_zone = element(var.availability\_zones, count.index)

  tags = merge(var.commontags,

    {

      "Name" = "demovpc-privatesubnet-${count.index}"

    }

  )

}

#Internet gateway

resource "aws\_internet\_gateway" "demovpc-igw" {

  vpc\_id = aws\_vpc.demovpc.id

  tags = merge(var.commontags,

    {

      "Name" = "demovpc-igw"

    }

  )

}

#public Route table

resource "aws\_route\_table" "demovpc-publirt" {

  vpc\_id = aws\_vpc.demovpc.id

  route {

    cidr\_block = "0.0.0.0/0"

    gateway\_id = aws\_internet\_gateway.demovpc-igw.id

  }

  tags = merge(var.commontags,

    {

      "Name" = "demopublcrt"

    }

  )

}

#Route table assosication

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

  count          = length(var.publicssubnetcidr)

  subnet\_id      = element(aws\_subnet.demovpc-PublicSubnet[\*].id, count.index)

  route\_table\_id = aws\_route\_table.demovpc-publirt.id

}

#Nat gateway require elastic ip

resource "aws\_eip" "nateip" {

  count = length(var.publicssubnetcidr)

  tags = merge(var.commontags,

    {

      "Name" = "Elastic-ip-${count.index}"

    }

  )

}

resource "aws\_nat\_gateway" "demovpc-natgateway" {

  count = length(var.publicssubnetcidr)

  #allocatio id nothig but elastic ip

  allocation\_id = element(aws\_eip.nateip[\*].id, count.index)

  subnet\_id     = element(aws\_subnet.demovpc-PublicSubnet[\*].id, count.index)

  tags = merge(var.commontags,

    {

      "Name" = "demovpc-nat-${count.index}"

    }

  )

}

#private Route table creation

resource "aws\_route\_table" "demovpc-privatert" {

  vpc\_id = aws\_vpc.demovpc.id

  count  = length(var.privatesubnetcidr)

  route {

    cidr\_block     = "0.0.0.0/0"

    nat\_gateway\_id = element(aws\_nat\_gateway.demovpc-natgateway[\*].id, count.index)

  }

  tags = merge(var.commontags,

    {

      "Name" = "demoprivatert"

    }

  )

}

#route table association

#Route table assosication to private subnets

resource "aws\_route\_table\_association" "demovpc-privatertassociation" {

  count          = length(var.privatesubnetcidr)

  subnet\_id      = element(aws\_subnet.demovpc-privateSubnet[\*].id, count.index)

  route\_table\_id = element(aws\_route\_table.demovpc-privatert[\*].id,count.index)

}

Terraform plan

Or another way two public subnets and two private subnets

Vim variable.tf

#Vpc CIDR

variable "vpccidr" {

  type        = string

  default     = "10.1.0.0/24"

  description = "CIDR for vpc"

}

#PublicSubnet CIDR

variable "publicssubnetcidr" {

  type    = list(string)

  default = ["10.1.0.0/26", "10.1.0.64/26"]

  description = "Public subnet CIDR "

}

#Private Subnet CIDR

variable "privatesubnetcidr" {

  type    = list(string)

  default = ["10.1.0.128/26", "10.1.0.192/26"]

  description = "Priavte subnet CIDR "

}

variable "commontags" {

  type = map(string)

  default = {

    "Name" = "demo"

    "Env"  = "Dev"

  }

}

variable "enable\_dns\_support" {

  type    = bool

  default = true

}

variable "availability\_zones" {

  type    = list(string)

  default = ["ap-south-1a", "ap-south-1b"]

}

Vim main.tf

 resource "aws\_vpc" "demovpc" {

  cidr\_block           = var.vpccidr

  enable\_dns\_support   = true

  enable\_dns\_hostnames = true

  #  enable\_dns\_hostnames = true

  tags = var.commontags

}

#Public subnet cidr

resource "aws\_subnet" "demovpc-PublicSubnet" {

  vpc\_id            = aws\_vpc.demovpc.id

  count             = length(var.publicssubnetcidr)

  cidr\_block        = element(var.publicssubnetcidr, count.index)

  availability\_zone = element(var.availability\_zones, count.index)

  tags = merge(var.commontags,

    {

      "Name" = "demovpc-publicsubnet-${count.index}"

    }

  )

}

#private subnet CIDr

resource "aws\_subnet" "demovpc-privateSubnet" {

  vpc\_id            = aws\_vpc.demovpc.id

  count             = length(var.privatesubnetcidr)

  cidr\_block        = element(var.privatesubnetcidr, count.index)

  availability\_zone = element(var.availability\_zones, count.index)

  tags = merge(var.commontags,

    {

      "Name" = "demovpc-privatesubnet-${count.index}"

    }

  )

}

#Internet gateway

resource "aws\_internet\_gateway" "demovpc-igw" {

  vpc\_id = aws\_vpc.demovpc.id

  tags = merge(var.commontags,

    {

      "Name" = "demovpc-igw"

    }

  )

}

#public Route table

resource "aws\_route\_table" "demovpc-publirt" {

  vpc\_id = aws\_vpc.demovpc.id

  route {

    cidr\_block = "0.0.0.0/0"

    gateway\_id = aws\_internet\_gateway.demovpc-igw.id

  }

  tags = merge(var.commontags,

    {

      "Name" = "demopublcrt"

    }

  )

}

#Route table assosication

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

  count          = length(var.publicssubnetcidr)

  subnet\_id      = element(aws\_subnet.demovpc-PublicSubnet[\*].id, count.index)

  route\_table\_id = aws\_route\_table.demovpc-publirt.id

}

#Nat gateway require elastic ip

resource "aws\_eip" "nateip" {

  count = length(var.publicssubnetcidr)

  tags = merge(var.commontags,

    {

      "Name" = "Elastic-ip-${count.index}"

    }

  )

}

resource "aws\_nat\_gateway" "demovpc-natgateway" {

  count = length(var.publicssubnetcidr)

  #allocatio id nothig but elastic ip

  allocation\_id = element(aws\_eip.nateip[\*].id, count.index)

  subnet\_id     = element(aws\_subnet.demovpc-PublicSubnet[\*].id, count.index)

  tags = merge(var.commontags,

    {

      "Name" = "demovpc-nat-${count.index}"

    }

  )

}

#private Route table creation

resource "aws\_route\_table" "demovpc-privatert" {

  vpc\_id = aws\_vpc.demovpc.id

  count  = length(var.privatesubnetcidr)

  route {

    cidr\_block     = "0.0.0.0/0"

    nat\_gateway\_id = element(aws\_nat\_gateway.demovpc-natgateway[\*].id, count.index)

  }

  tags = merge(var.commontags,

    {

      "Name" = "demoprivatert"

    }

  )

}

#route table association

#Route table assosication to private subnets

resource "aws\_route\_table\_association" "demovpc-privatertassociation" {

  count          = length(var.privatesubnetcidr)

  subnet\_id      = element(aws\_subnet.demovpc-privateSubnet[\*].id, count.index)

  route\_table\_id = element(aws\_route\_table.demovpc-privatert[\*].id,count.index)

}

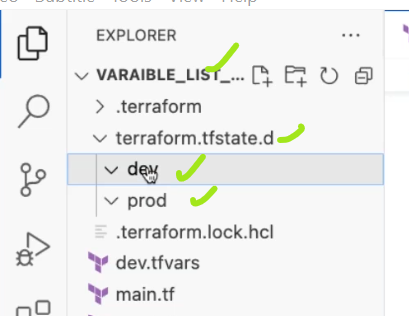
36 video : vpc explantion

Terraform workspace new dev

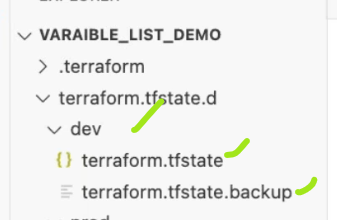
Terraform workspace new prod

Terraform workspace list

Terraform workspace select dev 🡪switch to dev



Terraform apply



37 videos : map variables using ec2 creations ,security grp , and output variables

Vim provider.tf

terraform {

  #  required\_version = ">=1.3.0" #terraform Version

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 5.0" #provider Version like aws/azure/Gcp providers

    }

  }

}

# Configure the AWS Provider

provider "aws" {

  region  = "ap-south-1"

  profile = "terraformprofile" #pass the profile name

}

Vim variables.tf

variable "aws\_region" {

  type    = string

  default = "ap-south-1"

}

variable "amiid" {

  type = map(string)

  default = {

    "redhat" = "ami-0645cf88151eb2007"

    "amazon" = "ami-02e94b011299ef128"

    "ubuntu" = "ami-0287a05f0ef0e9d9a"

  }

  description = "Amid id map based os types"

}

variable "commontags" {

  type = map(string)

  default = {

    "project" = "demo"

  }

}

Vim main.tf

resource "aws\_instance" "webserver" {

  ami           = var.amiid["amazon"]

  instance\_type = "t2.micro"

  tags = merge(var.commontags,

    {

      "Name" = "Webserver"

    }

  )

  #Key value pair name to passs

  key\_name = "saiphp"

  vpc\_security\_group\_ids = [aws\_security\_group.all-traffic-sg1.id]

  #user\_date = file("install-httpd.sh") --if you want pass the file here file path also

  user\_data = <<-EOF

    #!/bin/bash

    yum install git wget zip net-tools -y

    yum install httpd -y

    systemctl enable httpd

    systemctl start httpd

    echo "Weclome terraform ec2 sg crt" > /var/www/html/index.html

    EOF

}

#Security grp creation

resource "aws\_security\_group" "all-traffic-sg1" {

  name = "all-traffic-sg1"

  ingress {

    description = "TLS from VPC"

    from\_port   = 22

    to\_port     = 22

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    description = "TLS from VPC"

    from\_port   = 80

    to\_port     = 80

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  #Everry where we can open the ports outbound

  egress  {

    from\_port   = 0

    to\_port     = 0

    protocol    = "-1"

    cidr\_blocks = ["0.0.0.0/0"]

    description = "allow all ip & port outbound"

  }

}

#out put variable display the output

output "webserver-publicip" {

  value = aws\_instance.webserver.public\_ip

  description = "public ip of webserver vm"

}

output "webserver-sg" {

  value = aws\_security\_group.all-traffic-sg1.id

  description = "securiy grp id"

}

Terraform init

Terraform fmt

Terraform validate

Terraform plan

Terraform apply

38th video : aws RDS and data source or data block ec2 demo

Aws RDS:

Vim providers.tf you already knows or if you don’t copy above one

Vim variables.tf

variable "mysqlrds\_username" {

  type = string

  description = "mysql admin username "

}

variable "mysqlrds\_password" {

    type = string

    description = "mysql admin passowrd"

    sensitive = true

}

Vim terraform.tfvars or vim dev.auto.tfvars

#this Is one way apporve pass variable

mysqlrds\_username = "devdb"

mysqlrds\_password = "12345scm@"

#another way exportn terminal

#export TF\_VARS\_mysqlrds\_usernam=devdb

#export TF\_VARS\_mysqlrds\_password=12345scm@

Vim main.tf

resource "aws\_db\_instance" "default" {

  allocated\_storage    = 10

  db\_name              = "mydb"

  engine               = "mysql"

  engine\_version       = "5.7"

  instance\_class       = "db.t3.micro"

  username             = var.mysqlrds\_username

  password             = var.mysqlrds\_password

  parameter\_group\_name = "default.mysql5.7"

  skip\_final\_snapshot  = true

}

Terraform plan

Terraform apply

Data source or data block :

Vim provider.tf 🡪 you already know what to declare terraform version and provder version and credentials and region

Vim main.tf

data "aws\_ami" "redhat" {

  most\_recent = true

  filter {

    name   = "name"

    values = ["RHEL-9.5.0\_HVM-20250313-x86\_64-\*"]

  }

  filter {

    name   = "virtualization-type"

    values = ["hvm"]

  }

  filter {

    name   = "root-device-type"

    values = ["ebs"]

  }

  owners = ["amazon"] # Canonical

}

data "aws\_availability\_zones" "aws\_azs" {

  state                  = "available"

  all\_availability\_zones = true

}

output "aws\_azs\_list" {

  value = data.aws\_availability\_zones.aws\_azs.names

}

output "redhatami" {

  value = data.aws\_ami.redhat.id

}

resource "aws\_instance" "webserver1" {

  ami               = data.aws\_ami.redhat.id

  availability\_zone = data.aws\_availability\_zones.aws\_azs.names[0] # you use [\*] also for all fileds

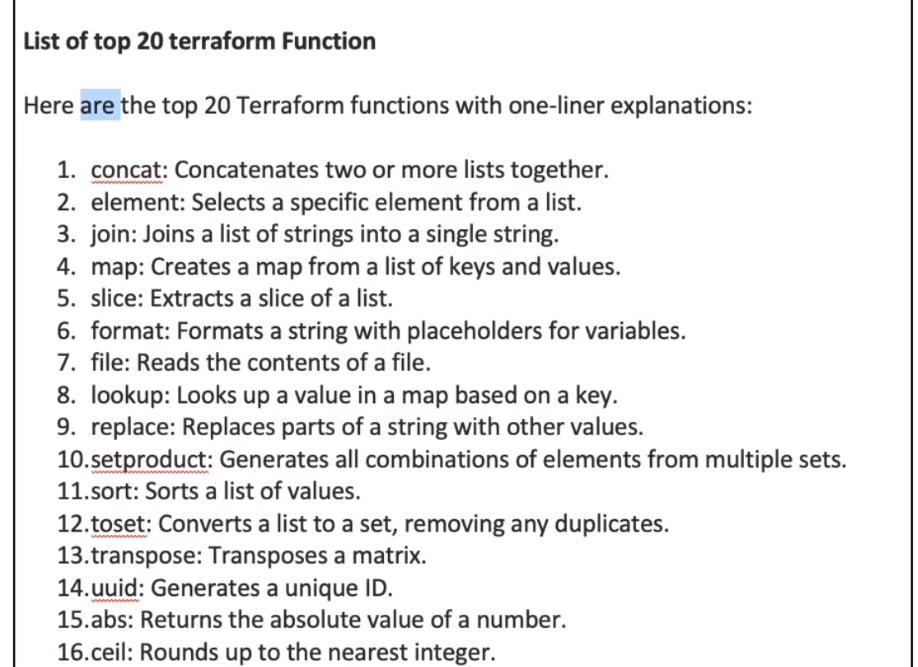
  instance\_type     = "t2.micro"

}

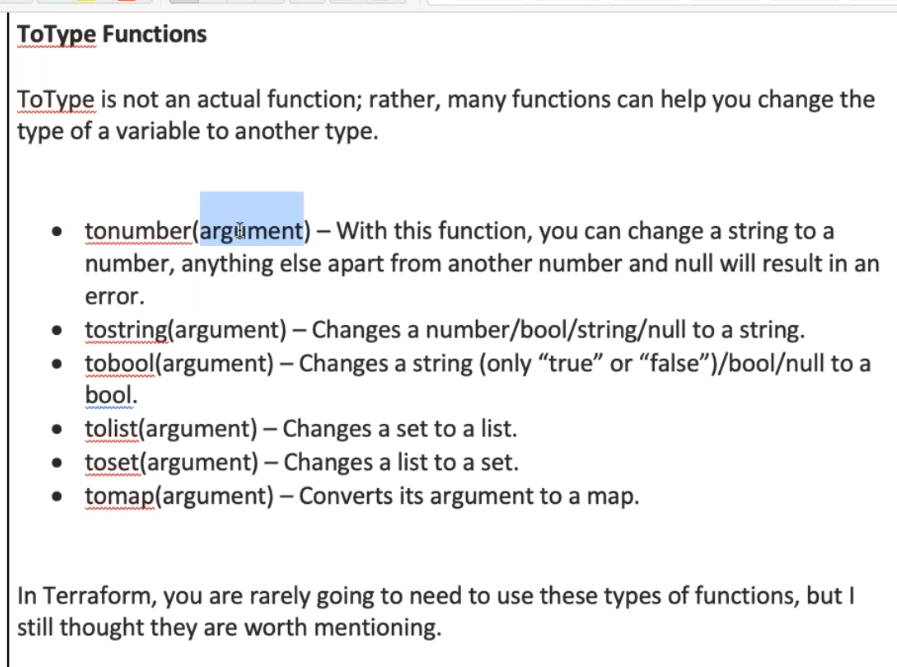
Terraform plan and apply

39 videos : terraform Function I think not much important just knowledge purposed

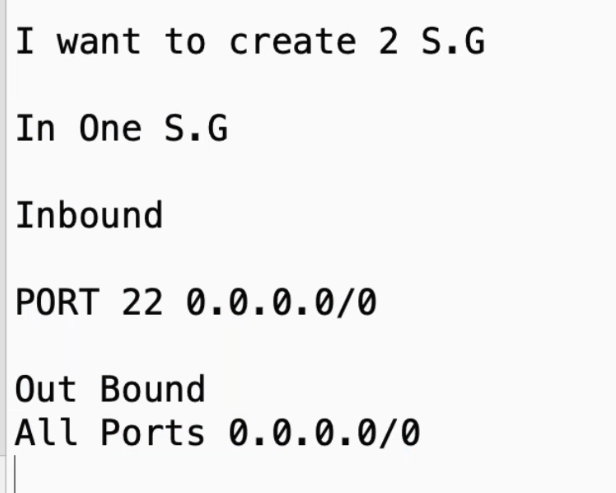
40 type conversion function







41 videos : for\_Expression





Vim provider.tf

Vim main.tf

variable "security\_group\_rules" {

    type = map(object({

      from\_port = string

      to\_port = string

      description = string

      cidr\_blocks = list(string)

      protocol  = string

    }))

  default = {

    "ssh" = {

      from\_port = 22

      to\_port = 22

      description = "allow ssh port 22"

      protocol = "tcp"

      cidr\_blocks = ["0.0.0.0/0"]

    },

    "http" = {

        from\_port = 80

        to\_port = 80

        description = "allow ssh port 80"

        protocol = "tcp"

        cidr\_blocks = ["0.0.0.0/0"]

    }

  }

}

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

    for\_each = var.security\_group\_rules

    name        = "allow\_${each.key}" # Allow ssh and http both etiration for loop

    description = "Allow Traffic for ${each.key} " #Allow traffic for ssh & http

  #vpc\_id      = aws\_vpc.main.id

  ingress {

    description      = each.value.description  #allow ssh port

    from\_port        = each.value.from\_port #22 &80

    to\_port          = each.value.to\_port #22 & 80

    protocol         = each.value.protocol # Tcp

    cidr\_blocks      = each.value.cidr\_blocks # ["0.0.0.0/0"]

  }

  egress {

    from\_port        = 0

    to\_port          = 0

    protocol         = "-1"

    cidr\_blocks      = ["0.0.0.0/0"]

  }

  tags = {

    Name = "allow\_tls"

  }

}

resource "aws\_instance"  "demo" {

  ami = "ami-02e94b011299ef128"

  instance\_type = "t2.micro"

  vpc\_security\_group\_ids = [for securitygroup in aws\_security\_group.allow\_ssh: securitygroup.id]

}

#output block

output "sg\_ids" {

  value = [ for securitygroup in aws\_security\_group.allow\_ssh: securitygroup.id]

}

43 videos: Dynamic block -🡪 I think not required

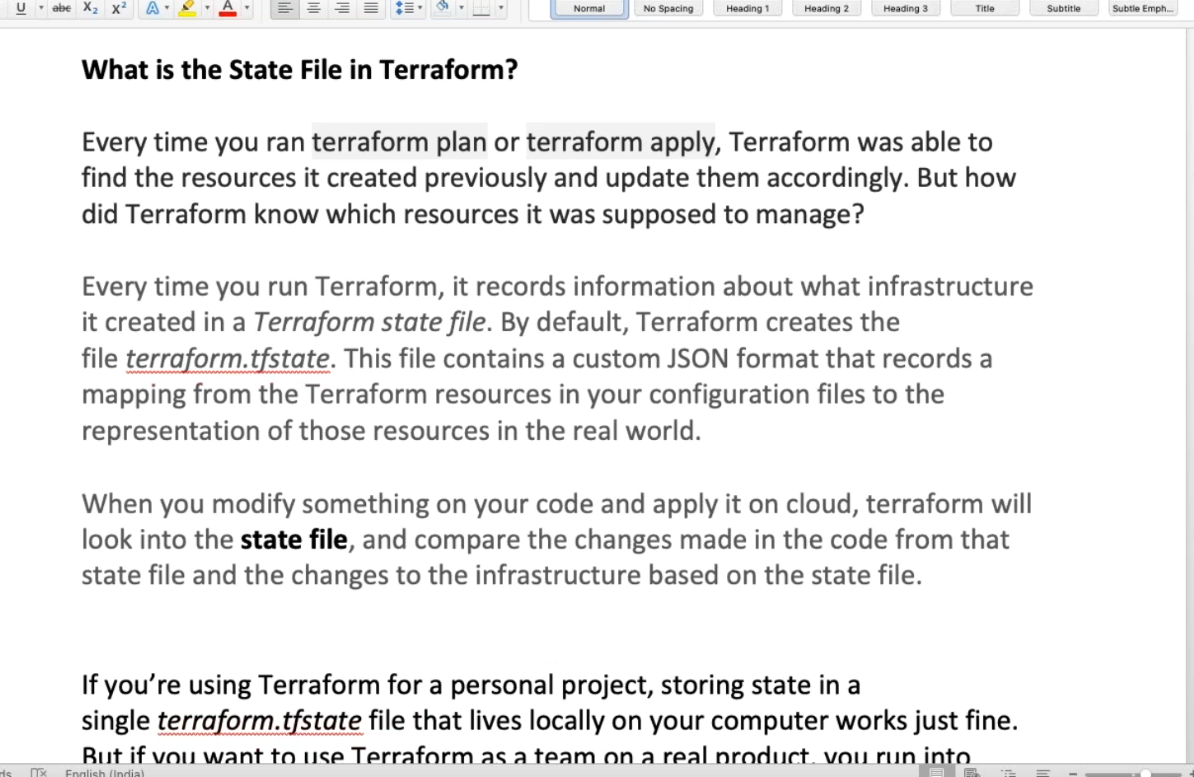
44: vpc lab for EKS setup

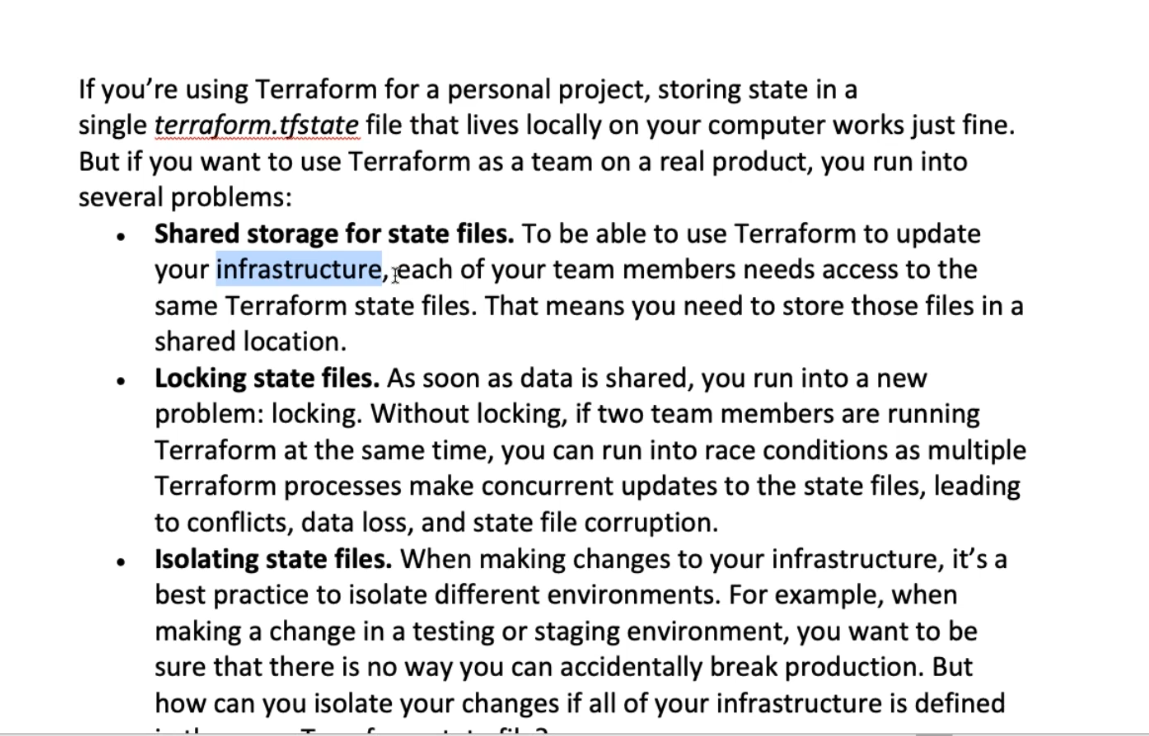
45. EKS cluster setup in terraform

46 . Terraform state file

Terraform.tfstate new updates in these file

Terraform.tfstate.backup is previous state it is maintaining in the backup file





Available backend configuration terraform state files

S3

Consul

Gcs

Azure storage

Remote

Pg

Cos

Oss

--etc

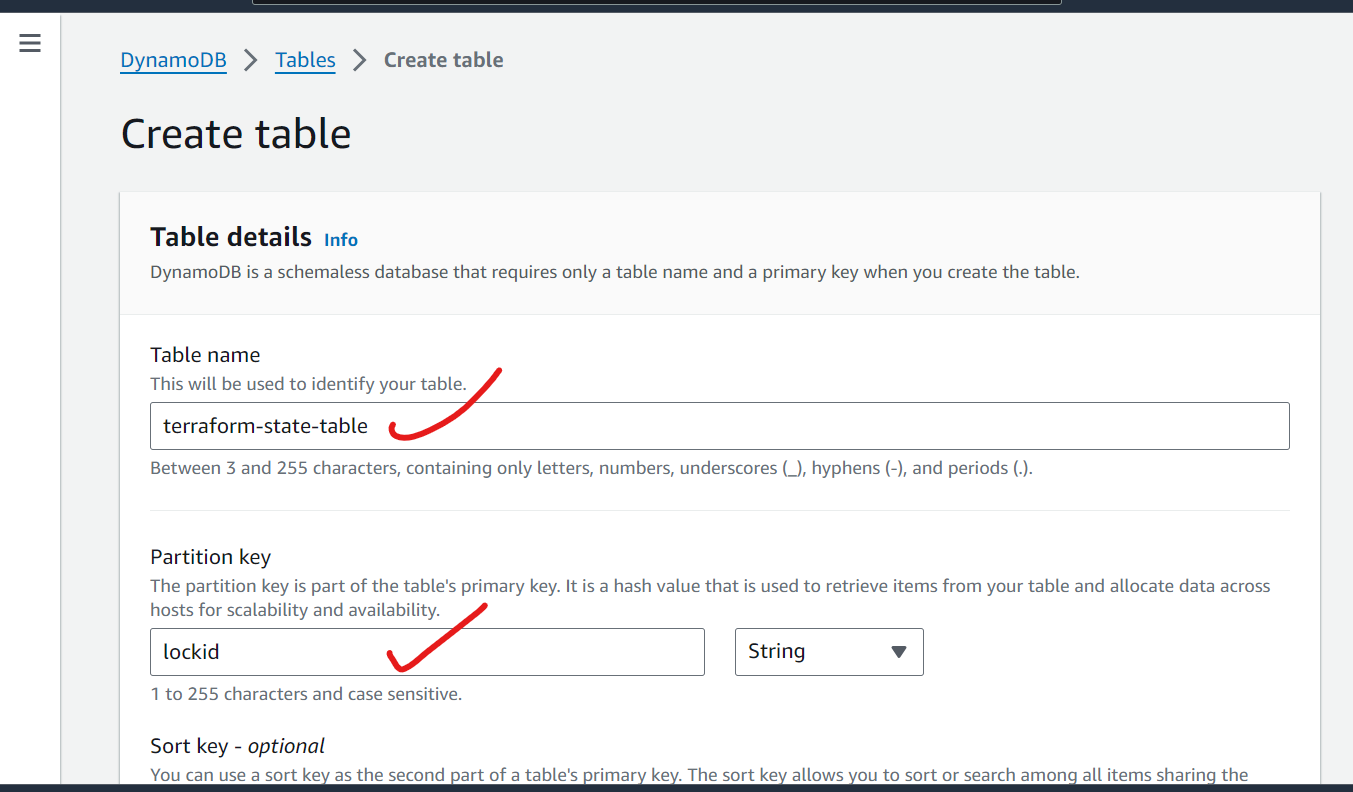
Create a s3 bucket

Bucket name terraform-state

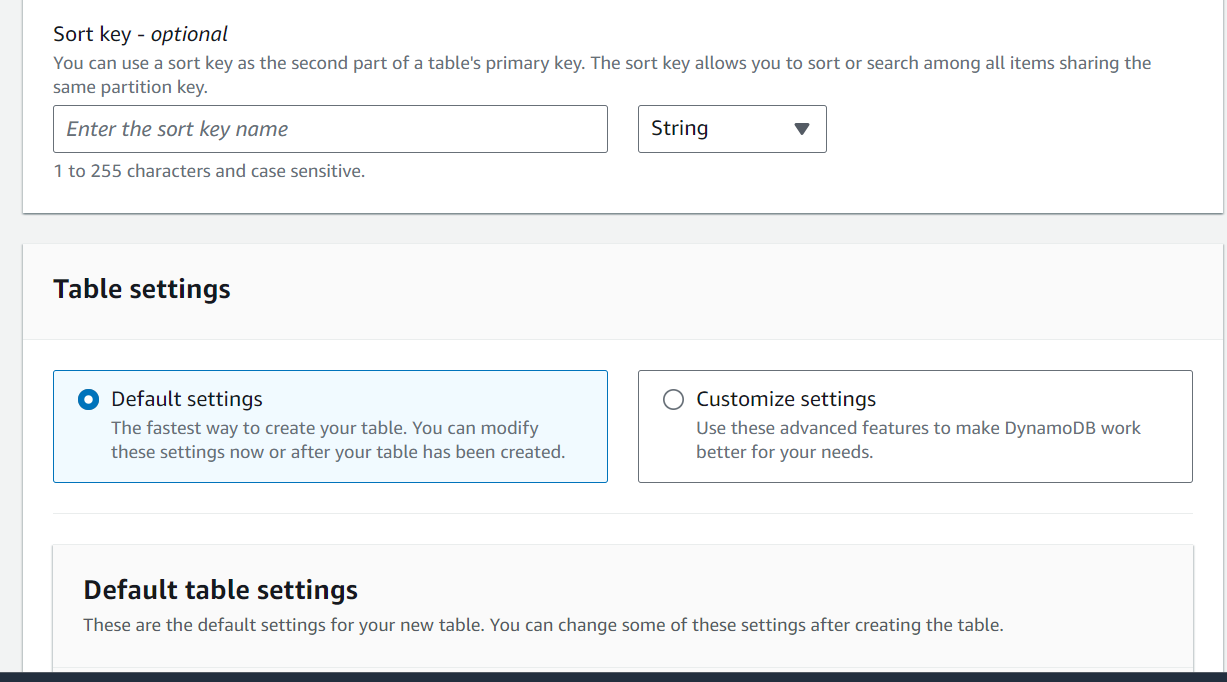
Bucket name : terraform-state-bucketsai

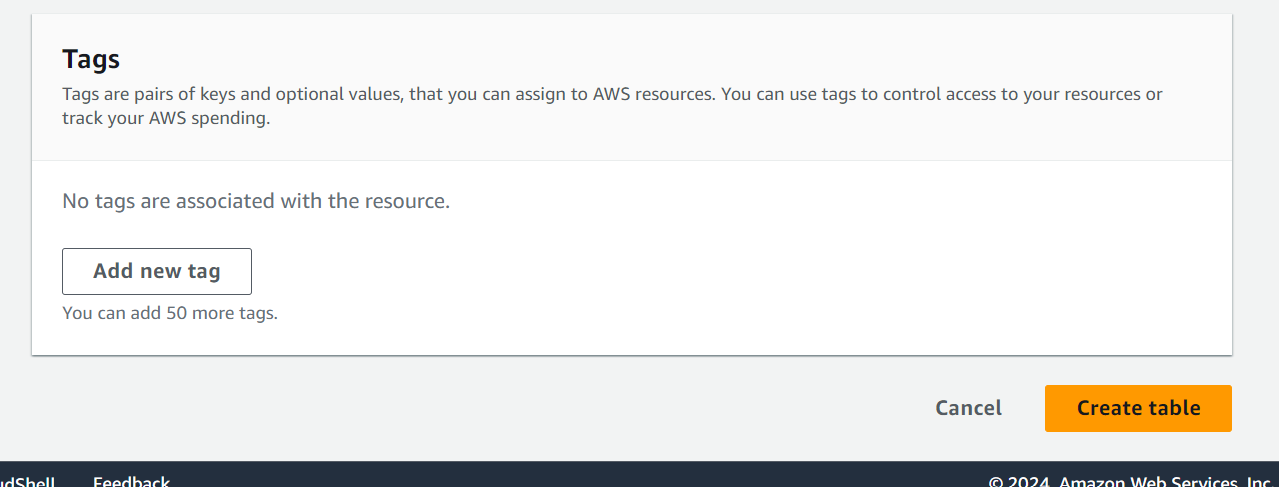
Under the bucket create a folder dev and sit

Create a dynamodb table



LockID use these name exactly same





Click on create table

Vim terraform.tf

terraform {

  required\_version = ">= 1.3.0"

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 5.0"

    }

  }

#Shared storage for terraform state files

  backend "s3" {

    bucket         = "terraform-state-bucketsai-fresh"  #Get the state file from these here and update the state file from these here

    key            = "dev/terraform.tfstate"

    region         = "ap-south-1"

    dynamodb\_table = "terraform-state-lock"  #this for state file locking

    profile        = "terraformprofile"

  }

}

# AWS Provider Configuration

provider "aws" {

  region  = "ap-south-1"

  profile = "terraformprofile"

}

Vim main.tf

resource "aws\_instance" "demo" {

  ami           = "ami-05a5bb48beb785bf1"

  instance\_type = "t2.micro"

  key\_name      = "vpc-demp"

  tags = {

    Name = "dev"

  }

}

47 declare the varibales

48 .user the varibales

terraformprofile

terraform init --backend-config=dev.tfbackend

terraform init --reconfigure --backend-config=dev.tfbackend

terraform plan –var-file=dev/dev.tfvars

terraform apply –auto-approve –var-file=dev/dev.tfvars

49 : Video terraform import

terraform show 🡪 it will read the terraform state file it will show human readable format

terraform -help

terraform state list 🡪 it is maintaingin resource /data source the information maintaining

terraform state show aws\_instance.demo

terraform refresh 🡪 you want change in aws console level ,terraform state don’t know about what you change

you want get that changes into terraform state file use terraform refresh command

terraform import command :

create default instance in aws console

vim terraform.tf

terraform {

  required\_version = ">= 1.3.0"

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 5.0"

    }

  }

  backend "s3" {

    bucket         = "terraform-state-bucketsai-fresh"

    key            = "import/terraform.tfstate"

    region         = "ap-south-1"

    dynamodb\_table = "terraform-state-lock"

    profile        = "terraformprofile"

  }

}

# AWS Provider Configuration

provider "aws" {

  region  = "ap-south-1"

  profile = "terraformprofile"

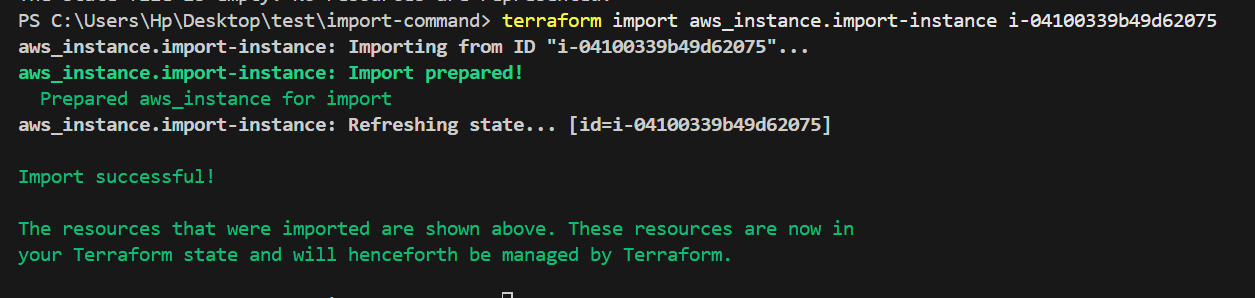
}

Vim main.tf

resource "aws\_instance" "dev" {

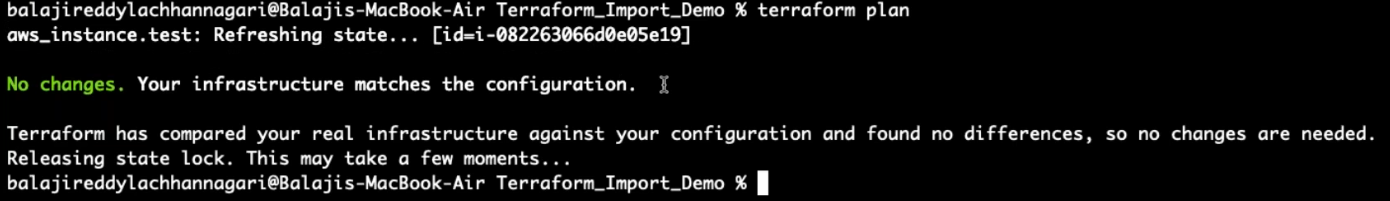
}

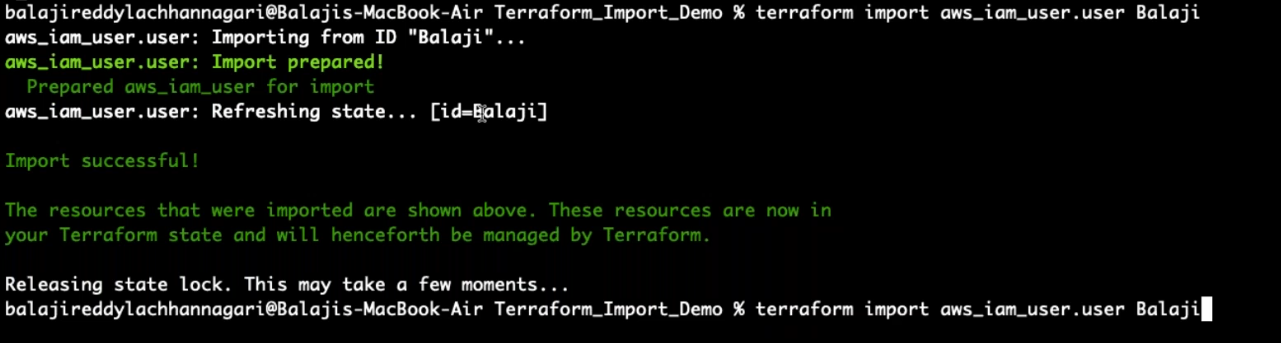
terraform import aws\_instance.dev i-04100339b49d62075(instance-id) 🡪



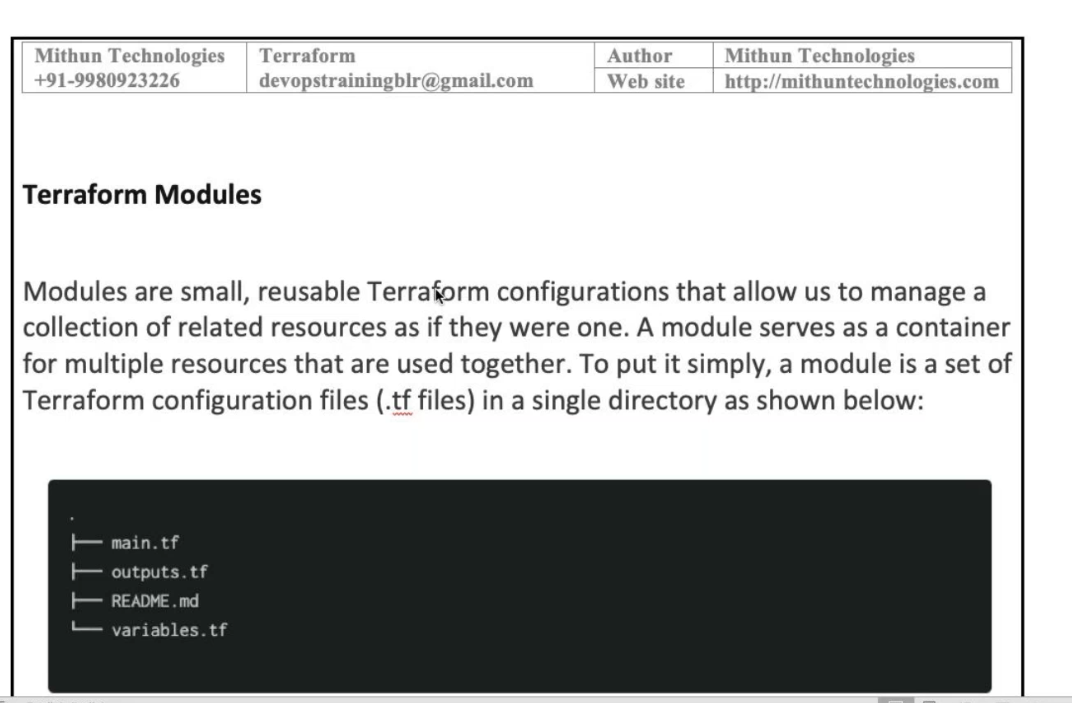
Terraform show

Terraform state list





50 .Terraform Modules



Type of modules :

Root module

Child module

Published mpdule 🡪 Terraform registry published modules

If you calling same module multiple time you can use

Terraform init

Or

Terraform get

Mkdir terraform-module

Vim terraform.tf

terraform {

  required\_version = ">= 1.3.0"

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 5.0"

    }

  }

 # AWS Provider Configuration

provider "aws" {

  region  = "ap-south-1"

  profile = "terraformprofile"

}

Vim main.tf

Terraform registry published modules

module "ec2\_instance" {

  source  = "terraform-aws-modules/ec2-instance/aws"

  name = "single-instance"

  instance\_type = "t2.micro"

  key\_name      = "2025-key"

  monitoring    = true

  ami           = "ami-0c55b159cbfafe1f0" # Replace with your desired AMI ID

  subnet\_id = "subnet-05245b7de3874e860"

  tags = {

    Terraform   = "true"

    Environment = "dev"

  }

}

module "jenkins\_instance" {

  source  = "terraform-aws-modules/ec2-instance/aws"

  name = "single-instance"

  instance\_type = "t2.micro"

  key\_name      = "2025-key"

  monitoring    = true

  ami           = "ami-0c55b159cbfafe1f0" # Replace with your desired AMI ID

  subnet\_id = "subnet-05245b7de3874e860"

  tags = {

    Terraform   = "true"

    Environment = "dev"

  }

}

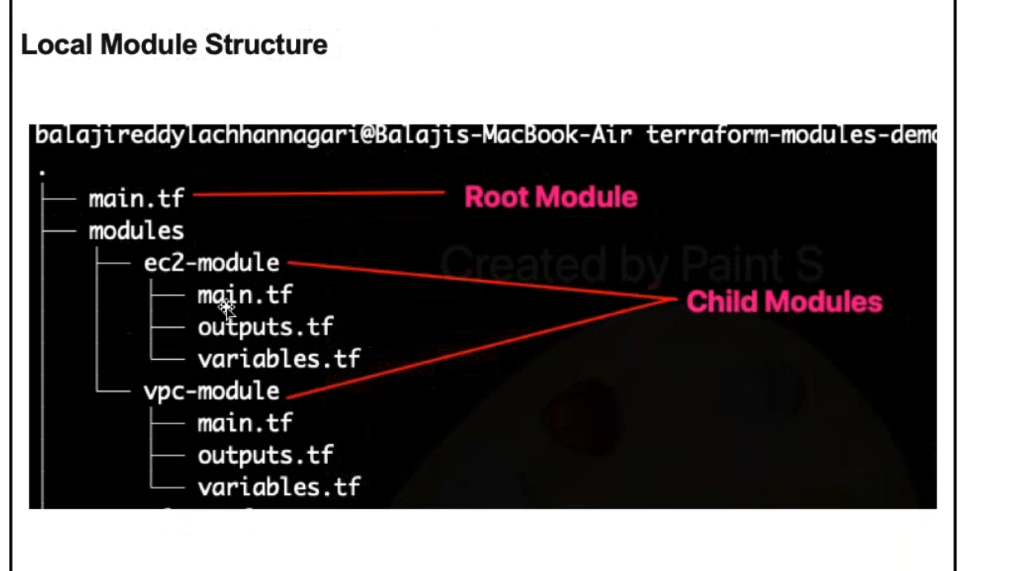
\*\*\*Note :If you calling same module multiple time you can use

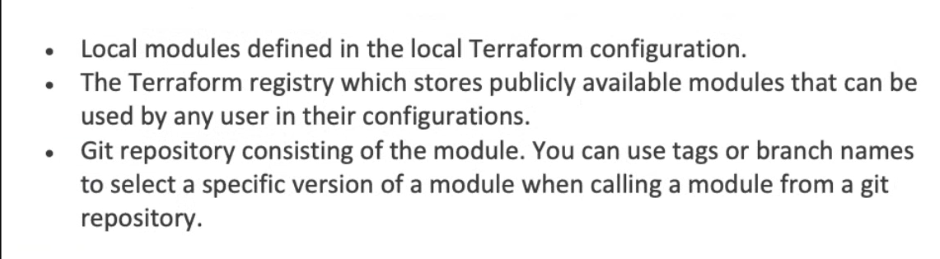
Terraform init

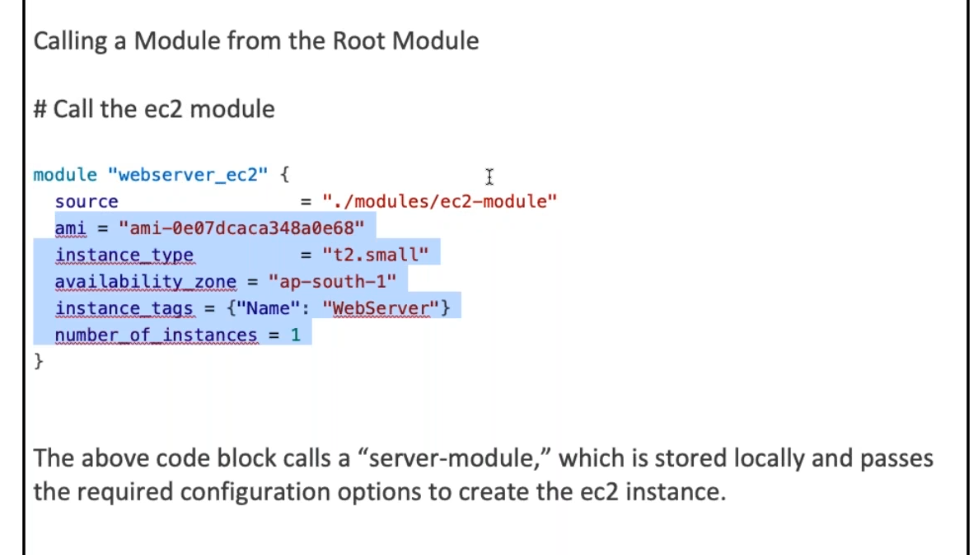
Or

Terraform get

51 : Local modules







Local module

Mkdir terraform-local-module 🡪cd terraform-local-module

Mkdir ec2

Mkdir s3

Mkdir iam

Cd terraform-local-module

Vim terraform.tf

terraform {

  required\_version = ">= 1.3.0"

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 5.0"

    }

  }

 # AWS Provider Configuration

provider "aws" {

  region  = "ap-south-1"

  profile = "terraformprofile"

}

Vim main.tf

provider "aws" {

  region  = "ap-south-1"

  profile = "terraformprofile"

}

module "ec2\_instance" {

  source = "./ec2"

  # other inputs...

}

module "s3\_bucket" {

  source = "./s3"

  # other inputs...

}

module "iam\_user" {

  source = "./iam"

  # other inputs...

}

Cd terraform-local-module 🡪 mkdir ec2

Vim main.tf

resource "aws\_instance" "demo" {

  ami           = "ami-05a5bb48beb785bf1"

  instance\_type = "t2.micro"

  key\_name      = "2025-key"

  tags = {

    Name = "dev"

  }

}

Cd terraform-local-module 🡪 mkdir s3

Vim main.tf

resource "aws\_s3\_bucket" "example\_bucket" {

  bucket = "sainath-unique-bucket-2025"  # Bucket name must be globally unique

  tags = {

    Name        = "MyBucket"

    Environment = "dev"

  }

}

Cd terraform-local-module 🡪 mkdir iam

Vim main.tf

resource "aws\_iam\_user" "example\_user" {

  name = "sainath-iam-user"

  tags = {

    Environment = "dev"

  }

}

You can pass the module in github/bitbucket in these below way

module "consul" {

source = "github.com/hashicorp/example"

}

module "consul" {

source = "git@github.com:hashicorp/example.git"

}

module "consul" {

source = "bitbucket.org/hashicorp/terraform-consul-aws"

}