Terraform LAB:

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

#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

# 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 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 terminal export below commands

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

export AWS\_SECRET\_ACCESS\_KEY="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"

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

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”]

\*/

  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

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

terraform plan -->

terraform apply --> apply changes it will ask yes

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

\*/

#Meta arguments

[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

}

#output Block

output "redhatlatestid" {

  value = data.aws\_ami.this.id

}

#Data Block

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"]

  }

}

#resource block

resource  "aws\_instance" "app" {

  ami = data.aws\_ami.this.id

  instance\_type = "t2.micro"

  count = 5

  tags = {

    "Name" = "App-${count.index}"

}

}

 ##

[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"

  }

}

Create IAM user in aws

 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"

  }

}

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}"

  }

}

Varables :

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

35 video :Complex type :

List

Set

Map

Create a folder

Mkdir test

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

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

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

    name = each.value

}

Terraform plan

Terraform apply

Create Security Group for instance :

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]

}