Terraform Registry:

<https://registry.terraform.io/>

Terraform Multi Provider:

$ cd ..

$ mkdir tf\_multiprovider

$ cd tf\_multiprovider

$ nano main.tf

terraform {

  required\_version = ">=1.0.0"

  required\_providers {

    aws = {

      source = "hashicorp/aws"

      version = "5.66.0"

    }

    random = {

      source = "hashicorp/random"

      version = "3.5.0"

    }

  }

}

$ terraform init

$ terraform providers

$ nano main.tf

terraform {

  required\_providers {

    aws = {

      source = "hashicorp/aws"

      version = "5.66.0"

    }

  }

}

provider "aws" {

  alias = "us\_east\_1"

  region = "us-east-1"

  access\_key = "YOUR\_AWS\_ACCESS\_KEY"

  secret\_key = "YOUR\_AWS\_SECRET\_KEY"

}

provider "aws" {

  alias = "us\_east\_2"

  region = "us-east-2"

  access\_key = "YOUR\_AWS\_ACCESS\_KEY"

  secret\_key = "YOUR\_AWS\_SECRET\_KEY"

}

resource "aws\_instance" "example\_east\_1" {

  provider = aws.us\_east\_1

  ami = "ami-0a0e5d9c7acc336f1"

  instance\_type = "t2.micro"

}

resource "aws\_s3\_bucket" "example\_east\_2" {

  provider = aws.us\_east\_2

  bucket = "my-unique-bucket-${random\_id.bucket\_id.hex}"

}

resource "random\_id" "bucket\_id" {

  byte\_length = 8

}

$ terraform init

$ terraform validate

$ terraform plan

$ terraform apply -auto-approve

$ terraform destroy -auto-approve

QKin1lE5McIHU0Y1Re6IWS9joNAAGUIzkkHx3A/f

TLS Provider:

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

$ mkdir tls\_test

$ cd tls\_test

$ nano main.tf

terraform {

  required\_version = ">= 1.0.0"

  required\_providers {

    http = {

      source  = "hashicorp/http"

      version = "2.1.0"

    }

    random = {

      source  = "hashicorp/random"

      version = "3.1.0"

    }

    local = {

      source  = "hashicorp/local"

      version = "2.1.0"

    }

    tls = {

      source  = "hashicorp/tls"

      version = "3.1.0"

    }

  }

}

resource "tls\_private\_key" "generated" {

  algorithm = "RSA"

}

resource "local\_file" "private\_key\_pem" {

  content  = tls\_private\_key.generated.private\_key\_pem

  filename = "MyAWSKey.pem"

}

$ terraform init

$ terraform validate

$ terraform plan

$ terraform apply -auto-approve

$ ls

$ cat MyAWSKey.pem

Upgrade Terraform to the latest version:

> Open browser within the lab machine:

> Browse: <https://developer.hashicorp.com/terraform/install#linux>

> Download AMD 64

$ cd ~/Downloads/

$ unzip terraform\_1.9.5\_linux\_amd64.zip

$ ls

$ which terraform

$ sudo mv terraform /usr/bin/

$ terraform -version

Local & Remote Exec Provisioners:

$ cd

$ mkdir provisioners\_test

$ cd provisioners\_test

$ nano main.tf

# Local values - to be used during resource creation

locals {

  ami\_id = "ami-0a0e5d9c7acc336f1"

  ssh\_user = "ubuntu"

}

# Provider section

provider "aws" {

  access\_key = "YOUR\_AWS\_ACCESS\_KEY"

  secret\_key = "YOUR\_AWS\_SECRET\_KEY"

  region = "us-east-1"

}

# Create AWS VPC

resource "aws\_vpc" "sl-vpc" {

  cidr\_block = "10.0.0.0/16"

  tags = {

    Name = "sl-vpc"

  }

}

# Create AWS Subnet

resource "aws\_subnet" "subnet-1" {

  vpc\_id     = aws\_vpc.sl-vpc.id

  cidr\_block = "10.0.1.0/24"

  map\_public\_ip\_on\_launch  = true

  depends\_on = [aws\_vpc.sl-vpc]

  tags = {

    Name = "sl-subnet"

  }

}

# Create AWS Route Table

resource "aws\_route\_table" "sl-route-table" {

  vpc\_id = aws\_vpc.sl-vpc.id

  tags = {

    Name = "sl-route-table"

  }

}

# Create AWS Route Table Association

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

  subnet\_id      = aws\_subnet.subnet-1.id

  route\_table\_id = aws\_route\_table.sl-route-table.id

}

# Create AWS Internet Gateway

resource "aws\_internet\_gateway" "gw" {

  vpc\_id = aws\_vpc.sl-vpc.id

  depends\_on = [aws\_vpc.sl-vpc]

  tags = {

    Name = "sl-gw"

  }

}

# Create AWS Route

resource "aws\_route" "sl-route" {

  route\_table\_id = aws\_route\_table.sl-route-table.id

  destination\_cidr\_block = "0.0.0.0/0"

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

}

# AWS security group resource block - 3 inbound & 1 outbound rule added

resource "aws\_security\_group" "sl-sg" {

  name = "sl-sg"

  vpc\_id = aws\_vpc.sl-vpc.id

  ingress {

    from\_port = 22

    to\_port = 22

    protocol = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    from\_port = 80

    to\_port = 80

    protocol = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    from\_port = 443

    to\_port = 443

    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"]

  }

  tags = {

    Name = "sl-sg"

  }

}

# TLS provider

resource "tls\_private\_key" "tls-key" {

  algorithm = "RSA"

}

# AWS Keypair with TLS

resource "aws\_key\_pair" "app-key" {

  key\_name   = "sl-key"

  public\_key = tls\_private\_key.tls-key.public\_key\_openssh

}

# Saving the key file

resource "local\_file" "sl-key" {

  content  = tls\_private\_key.tls-key.private\_key\_pem

  filename = "sl-key.pem"

  file\_permission = "0400"

}

# Change file permissions to 400

# resource "null\_resource" "set\_key\_permission" {

#   provisioner "local-exec" {

#     command = "chmod 400 sl-key.pem"

#   }

#   # Ensure this runs after the local\_file resource

#   depends\_on = [local\_file.sl-key]

# }

# AWS EC2 instance resource block

resource "aws\_instance" "web" {

  ami = local.ami\_id

  instance\_type = "t2.micro"

  associate\_public\_ip\_address = true

  subnet\_id = aws\_subnet.subnet-1.id

  security\_groups = [aws\_security\_group.sl-sg.id]

  key\_name = aws\_key\_pair.app-key.key\_name

  tags = {

    Name = "Abhi Test"

  }

  # SSH Connection block which will be used by the provisioners - remote-exec

  connection {

    type = "ssh"

    host = self.public\_ip

    user = local.ssh\_user

    # private\_key = file(local.private\_key\_path)

    private\_key = tls\_private\_key.tls-key.private\_key\_pem

    timeout = "2m"

  }

  # Remote-exec Provisioner Block - wait for SSH connection

  provisioner "remote-exec" {

    inline = [

      "echo 'wait for SSH connection to be ready...'",

      "touch /home/ubuntu/demo-file-from-terraform.txt"

    ]

  }

  # Local-exec Provisioner Block - create an Ansible Dynamic Inventory

  provisioner "local-exec" {

    command = "echo ${self.public\_ip} >> myhosts"

  }

  # Local-exec Provisioner Block - create an Ansible Dynamic Inventory

  provisioner "local-exec" {

    command     = "echo Hello\_Me > hello.txt"

    working\_dir = "${path.module}/"

    interpreter = ["/bin/bash", "-c"]

  }

}

# Output block to print the public ip of instance

output "instance\_ip" {

  value = aws\_instance.web.public\_ip

}

$ terraform init

$ terraform validate

$ terraform plan

$ terraform apply -auto-approve

$ ls

$ cat myhosts

$ cat hello.txt

SSH into the EC2 instance:

$ ssh -i "sl-key.pem" ubuntu@YOUR\_INSTANCE\_PUBLIC\_IP

$ ls

$ exit

Destroy/Delete the resources:

$ terraform destroy -auto-approve

Auto Formatting:

$ cd

$ mkdir test\_format

$ cd test\_format

$ nano main.tf

resource "random\_string" "random"    {

  length = 10

  special = true

min\_numeric = 6

      min\_special = 2

min\_upper = 3

}

$ cat main.tf

$ terraform fmt

$ cat main.tf

Terraform Workspaces:

$ cd

$ mkdir test\_workspace

$ cd test\_workspace

$ nano main.tf

resource "tls\_private\_key" "generated" {

  algorithm = "RSA"

}

resource "local\_file" "private\_key\_pem" {

  content  = tls\_private\_key.generated.private\_key\_pem

  filename = "my-aws-default-key.pem"

}

resource "random\_string" "random" {

  length      = 15

  special     = true

  min\_numeric = 6

  min\_special = 2

  min\_upper   = 3

}

# Output block to print the random value

output "random\_output" {

  value = random\_string.random.result

}

$ terraform init

$ terraform validate

$ terraform apply -auto-approve

$ terraform state list

$ terraform show

$ terraform workspace select default

$ terraform show

$ terraform destroy -auto-approve

$ terraform workspace select DEV

$ terraform destroy -auto-approve

$ terraform workspace select default

$ terraform workspace delete DEV

**This command will immediately terminate the EC2 instance**

aws ec2 terminate-instances --instance-ids ***i-0d7d813dc323fbbf8*** --region us-east-2

VGnDuhW1CI3J9g.atlasv1.5WzLZjamR16qXUZoygfzHSwphoj5GiVD3OeXgibND0xhiuXk0OfS6gMD0VKvC8NLsu8

Make an Account:

<https://app.terraform.io/public/signup/account>

Terraform State:

$ mkdir tf\_state\_test

$ cd tf\_state\_test

$ nano main.tf

provider "aws" {

  access\_key = "YOUR\_AWS\_ACCESS\_KEY"

  secret\_key = "YOUR\_AWS\_SECRET\_KEY"

  region = "us-east-1"

}

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

  bucket = "myterraformstatedemo-YOUR\_NAME"

}

$ terraform init

$ terraform validate

$ terraform plan

$ terraform apply -auto-approve

$ nano terraform.tf

terraform {

  backend "s3" {

    bucket = "myterraformstatedemo-YOUR\_NAME"

    key = "prod/aws\_infra"

    region = "us-east-1"

  }

}

$ terraform init -backend-config="access\_key=YOUR\_AWS\_ACCESS\_KEY" -backend-config="secret\_key=YOUR\_AWS\_SECRET\_KEY"

$ terraform validate

$ terraform apply -auto-approve

$ terraform destroy -auto-approve

Login:

<https://app.terraform.io/>

> Create Organization

  > Name: YOURNAME\_demo\_org

$ terraform login

Terraform State Locking:

$ cd

$ mkdir s3-dynamodb-state

$ cd s3-dynamodb-state

$ nano s3.tf

provider "aws" {

  access\_key = "YOUR\_AWS\_ACCESS\_KEY"

  secret\_key = "YOUR\_AWS\_SECRET\_KEY"

  region = "us-east-1"

}

resource "aws\_s3\_bucket" "backend" {

    bucket = "myterraformstatedemo-00-backend-YOURNAME"

    tags = {

        Name = "S3 Remote Terraform State Store"

    }

}

resource "aws\_s3\_bucket\_versioning" "enable\_versioning" {

  bucket = aws\_s3\_bucket.backend.id

  versioning\_configuration {

    status = "Enabled"

  }

}

resource "aws\_s3\_bucket\_server\_side\_encryption\_configuration" "s3\_encryption" {

  bucket = aws\_s3\_bucket.backend.id

  rule {

    apply\_server\_side\_encryption\_by\_default {

      sse\_algorithm     = "AES256"

    }

  }

}

resource "aws\_s3\_bucket\_object\_lock\_configuration" "s3\_lock" {

  bucket = aws\_s3\_bucket.backend.id

  object\_lock\_enabled = "Enabled"

  depends\_on = [aws\_s3\_bucket\_versioning.enable\_versioning]

}

$ nano dynamo.tf

resource "aws\_dynamodb\_table" "terraform-lock" {

    name           = "terraform\_state"

    read\_capacity  = 5

    write\_capacity = 5

    hash\_key       = "LockID"

    attribute {

        name = "LockID"

        type = "S"

    }

    tags = {

        "Name" = "DynamoDB Terraform State Lock Table"

    }

}

$ nano main.tf

terraform {

  backend "s3" {

    bucket         = "myterraformstatedemo-00-backend-YOURNAME"

    key            = "terraform.tfstate"

    region         = "us-east-1"

    dynamodb\_table = "terraform\_state"

  }

}

$ mv main.tf main.tf\_test

$ terraform init

$ terraform init -backend-config="access\_key=YOUR\_AWS\_ACCESS\_KEY" -backend-config="secret\_key=YOUR\_AWS\_SECRET\_KEY"

$ mv main.tf main.tf\_test

$ terraform init

$ terraform validate

$ terraform plan

$ terraform apply -auto-approve

$ mv main.tf\_test main.tf

$ terraform init -backend-config="access\_key=YOUR\_AWS\_ACCESS\_KEY" -backend-config="secret\_key=YOUR\_AWS\_SECRET\_KEY"

$ nano s3.tf (>>>>>>>>APPEND the below content)

output "s3\_bucket\_id" {

  value = aws\_s3\_bucket.backend.id

}

$ terraform apply -auto-approve

Migrate the state to Terraform cloud:

> Login to https://app.terraform.io/ > Click on the Organization

> Create workspace > Terraform CLI > Name: my-aws-app

$ nano main.tf

terraform {

  backend "remote" {

    hostname       = "app.terraform.io"

    organization = "YOUR\_ORG\_NAME"

    workspaces {

      name = "my-aws-app"

    }

  }

}

$ terraform validate

$ terraform init -migrate-state

Sensitive Data:

$ cd

$ mkdir sensitive

$ cd sensitive

$ nano main.tf

variable "name" {

  type      = string

  sensitive = true

  default   = "Abhi"

}

variable "phone" {

  type      = string

  sensitive = true

  default   = "123-456"

}

locals {

  contact\_info = {

    name   = var.name

    phone = var.phone

  }

  my\_number = nonsensitive(var.phone)

}

output "name" {

  value = local.contact\_info.name

  sensitive = true

}

output "phone\_number" {

  value = local.contact\_info.phone

  sensitive = true

}

output "my\_number" {

  value = local.my\_number

  sensitive = true

}

$ terraform init

$ terraform validate

$ terraform apply -auto-approve

Terraform Variables - validation & sensitive data

$ mkdir test\_validate

$ cd test\_validate

$ nano variables.tf

# Validation for cloud provider choice

variable "cloud" {

  type        = string

  description = "Enter the cloud provider name (aws, azure, gcp, vmware)"

  validation {

    condition     = contains(["aws", "azure", "gcp", "vmware"], lower(var.cloud))

    error\_message = "Only aws, azure, gcp, and vmware are accepted as cloud providers."

  }

  validation {

    condition     = lower(var.cloud) == var.cloud

    error\_message = "The cloud provider name must be in lowercase."

  }

}

# Variable for no capital letters

variable "no\_caps" {

  type        = string

  description = "Enter a lowercase string."

  validation {

    condition     = lower(var.no\_caps) == var.no\_caps

    error\_message = "The string must be in lowercase."

  }

}

# Variable with character limit

variable "character\_limit" {

  type        = string

  description = "Enter a string of exactly 3 characters."

  validation {

    condition     = length(var.character\_limit) == 3

    error\_message = "This variable must contain exactly 3 characters."

  }

}

# IP address validation

variable "ip\_address" {

  type        = string

  description = "Enter a valid IP address."

  validation {

    condition     = can(regex("^(?:[0-9]{1,3}\\.){3}[0-9]{1,3}$", var.ip\_address))

    error\_message = "The input must be a valid IP address in the form X.X.X.X."

  }

}

# Sensitive data variable

variable "phone\_number" {

  type        = string

  description = "Enter a sensitive phone number."

  sensitive   = true

}

$ nano main.tf

# Managing sensitive information with locals

locals {

  contact\_info = {

    cloud         = var.cloud

    department    = var.no\_caps

    cost\_code     = var.character\_limit

    phone\_number  = var.phone\_number

  }

}

output "cloud" {

  value = local.contact\_info.cloud

}

output "department" {

  value = local.contact\_info.department

}

output "cost\_code" {

  value = local.contact\_info.cost\_code

}

output "phone\_number" {

  description = "A sensitive output of the phone number"

  value       = local.contact\_info.phone\_number

  sensitive   = true

}

$ terraform init

$ terraform validate

$ terraform plan

cloud = "abhi"

cost\_code = "abhi"

ip\_address = 22.22.22.22

department = "abhi"

phone\_number = 12345

$ terraform apply -auto-approve

cloud = "aws"

cost\_code = "dev"

ip\_address = 22.22.22.22

department = "abhi"

phone\_number = 12345

$ terraform output

$ terraform output phone\_number

Dynamic Block:

$ cd

$ mkdir dynamic

$ cd dynamic

$ nano main.tf

#Configure the AWS provider

provider "aws" {

  # Replace with your actual AWS credentials

  access\_key = "YOUR\_ACCESS\_KEY"

  secret\_key = "YOUR\_SECRET\_KEY"

  region     = "us-east-1" # Replace with your desired region

}

resource "aws\_vpc" "my\_vpc" {

  cidr\_block = "10.0.0.0/16"

  tags = {

    Name = "MyVPC"

  }

}

locals {

  ingress\_rules = {

    ssh = {

      from\_port   = 22

      to\_port     = 22

      protocol    = "tcp"

      cidr\_blocks = ["0.0.0.0/0"]

    },

    http = {

      from\_port   = 80

      to\_port     = 80

      protocol    = "tcp"

      cidr\_blocks = ["0.0.0.0/0"]

    }

  }

}

resource "aws\_security\_group" "dynamic\_sg" {

  name   = "dynamic-sg"

  vpc\_id = aws\_vpc.my\_vpc.id

  dynamic "ingress" {

    for\_each = local.ingress\_rules

    content {

      from\_port   = ingress.value.from\_port

      to\_port     = ingress.value.to\_port

      protocol    = ingress.value.protocol

      cidr\_blocks = ingress.value.cidr\_blocks

    }

  }

}

locals {

  instance\_suffix = "001"

}

resource "aws\_subnet" "my\_subnet" {

  vpc\_id            = aws\_vpc.my\_vpc.id  # Ensures subnet is within the created VPC

  cidr\_block        = "10.0.1.0/24"      # Specify a valid CIDR block within the VPC's range

  availability\_zone = "us-east-1a"       # Adjust to your desired AZ

  tags = {

    Name = "MySubnet"

  }

}

resource "aws\_instance" "web\_server" {

  ami           = "ami-0a0e5d9c7acc336f1"

  instance\_type = "t2.micro"

  subnet\_id     = aws\_subnet.my\_subnet.id

  tags = {

    Name = "WebServer-${local.instance\_suffix}"

  }

}

$ terraform init

$ terraform validate

$ terraform plan

$ terraform apply -auto-approve

Terraform Graph:

$ terraform graph > graph.dot

$ cat graph.dot

Browse: <http://www.webgraphviz.com/?tab=map>

Lifecycle Block:

$ nano main.tf

#Configure the AWS provider

provider "aws" {

  # Replace with your actual AWS credentials

  access\_key = "YOUR\_ACCESS\_KEY"

  secret\_key = "YOUR\_SECRET\_KEY"

  region     = "us-east-1" # Replace with your desired region

}

resource "aws\_vpc" "my\_vpc" {

  cidr\_block = "10.0.0.0/16"

  tags = {

    Name = "MyVPC"

  }

}

locals {

  ingress\_rules = {

    ssh = {

      from\_port   = 22

      to\_port     = 22

      protocol    = "tcp"

      cidr\_blocks = ["0.0.0.0/0"]

    },

    http = {

      from\_port   = 80

      to\_port     = 80

      protocol    = "tcp"

      cidr\_blocks = ["0.0.0.0/0"]

    }

  }

}

resource "aws\_security\_group" "dynamic\_sg" {

  name   = "dynamic-sg"

  vpc\_id = aws\_vpc.my\_vpc.id

  dynamic "ingress" {

    for\_each = local.ingress\_rules

    content {

      from\_port   = ingress.value.from\_port

      to\_port     = ingress.value.to\_port

      protocol    = ingress.value.protocol

      cidr\_blocks = ingress.value.cidr\_blocks

    }

  }

}

locals {

  instance\_suffix = "001"

}

resource "aws\_subnet" "my\_subnet" {

  vpc\_id            = aws\_vpc.my\_vpc.id  # Ensures subnet is within the created VPC

  cidr\_block        = "10.0.1.0/24"      # Specify a valid CIDR block within the VPC's range

  availability\_zone = "us-east-1a"       # Adjust to your desired AZ

  tags = {

    Name = "MySubnet"

  }

}

resource "aws\_instance" "web\_server" {

  ami           = "ami-0a0e5d9c7acc336f1"

  instance\_type = "t2.micro"

  subnet\_id     = aws\_subnet.my\_subnet.id

  tags = {

    Name = "WebServer-${local.instance\_suffix}"

  }

  lifecycle {

    prevent\_destroy = true

  }

}

$ terraform apply -auto-approve

$ terraform destroy -auto-approve

$ nano main.tf

#Configure the AWS provider

provider "aws" {

  # Replace with your actual AWS credentials

  access\_key = "YOUR\_ACCESS\_KEY"

  secret\_key = "YOUR\_SECRET\_KEY"

  region     = "us-east-1" # Replace with your desired region

}

resource "aws\_vpc" "my\_vpc" {

  cidr\_block = "10.0.0.0/16"

  tags = {

    Name = "MyVPC"

  }

}

locals {

  ingress\_rules = {

    ssh = {

      from\_port   = 22

      to\_port     = 22

      protocol    = "tcp"

      cidr\_blocks = ["0.0.0.0/0"]

    },

    http = {

      from\_port   = 80

      to\_port     = 80

      protocol    = "tcp"

      cidr\_blocks = ["0.0.0.0/0"]

    }

  }

}

resource "aws\_security\_group" "dynamic\_sg" {

  name   = "dynamic-sg"

  vpc\_id = aws\_vpc.my\_vpc.id

  dynamic "ingress" {

    for\_each = local.ingress\_rules

    content {

      from\_port   = ingress.value.from\_port

      to\_port     = ingress.value.to\_port

      protocol    = ingress.value.protocol

      cidr\_blocks = ingress.value.cidr\_blocks

    }

  }

}

locals {

  instance\_suffix = "001"

}

resource "aws\_subnet" "my\_subnet" {

  vpc\_id            = aws\_vpc.my\_vpc.id  # Ensures subnet is within the created VPC

  cidr\_block        = "10.0.1.0/24"      # Specify a valid CIDR block within the VPC's range

  availability\_zone = "us-east-1a"       # Adjust to your desired AZ

  tags = {

    Name = "MySubnet"

  }

}

resource "aws\_instance" "web\_server" {

  ami           = "ami-0a0e5d9c7acc336f1"

  instance\_type = "t2.micro"

  subnet\_id     = aws\_subnet.my\_subnet.id

  tags = {

    Name = "ABHI-WebServer-${local.instance\_suffix}"

  }

  lifecycle {

    create\_before\_destroy = true

  }

}

$ terraform apply -auto-approve

$ terraform destroy -auto-approve

Security BestPractice:

$ cd

$ mkdir security01

$ cd security01

$ nano ec2.tf

provider "aws" {

  region = "us-east-1"

}

resource "aws\_instance" "web" {

  ami           = "ami-0a0e5d9c7acc336f1"

  instance\_type = "t2.micro"

  tags = {

    Name = "MyMachine"

  }

}

$ aws configure

$ terraform init

$ terraform validate

$ terraform plan

$ terraform apply -auto-approve

Alt Method:

$ cd

$ mkdir security02

$ cd security02

Install Checkov

$ pip install checkov

$ nano credentials.tf

provider "aws" {

  access\_key = var.AWS\_ACCESS\_KEY

  secret\_key = var.AWS\_SECRET\_KEY

  region     = "us-east-1"

}

variable "AWS\_ACCESS\_KEY" {}

variable "AWS\_SECRET\_KEY" {}

Set the environment variables:

$ export AWS\_ACCESS\_KEY=YOUR\_AWS\_ACCESS\_KEY

$ export AWS\_SECRET\_KEY=YOUR\_AWS\_SECRET\_KEY

$ find ~/.local/bin -name checkov

$ ~/.local/bin/checkov -f credentials.tf

Open the Lab browser:

> Login to Github account

> Login to Terraform cloud: app.terraform.io

$ mkdir tfcloud\_test

$ cd tfcloud\_test

$ terraform -version

Upgrade Terraform to the latest version:

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

$ wget https://releases.hashicorp.com/terraform/1.9.6/terraform\_1.9.6\_linux\_arm64.zip

$ unzip terraform\_1.9.6\_linux\_arm64.zip

$ ls

$ which terraform

$ sudo mv terraform /usr/local/bin/

$ terraform -version

> Create a new TF cloud Workspace

  > Choose the organization > Workspaces > New > Workspace

  > Select - Default project > Create

  > Choose CLI Driven Workflow

Workspace name: demo\_testws\_01

    > Create

$ terraform login

  > type yes

  > Generate token

  > Paste the token in the console and press Enter

$ nano main.tf

terraform {

  cloud {

    organization = "abhi\_demo\_org"

    workspaces {

      name = "demo\_testws\_01"

    }

  }

}

variable "my\_string" {

  description = "My string input variable"

  type        = string

}

variable "listofstrings" {

  type    = list(string)

  default = ["my-string-1", "my-string-2", "my-string-3"]

}

variable "iplist" {

  type    = map(string)

  default = {

    prod = "100.100.100.100",

    dev  = "200.200.200.200"

  }

}

variable "env" {

  type    = map(any)

  default = {

    prod = {

      ip = "100.100.100.100",

      loc = "us"

    },

    dev = {

      ip = "200.200.200.200",

      loc = "in"

    }

  }

}

output "print\_string\_var" {

  value = var.my\_string

}

output "print\_full\_list" {

  value = var.listofstrings

}

output "print\_first\_list\_item" {

  value = var.listofstrings[0]

}

output "print\_prod\_ip" {

  value = var.iplist["prod"]

}

output "print\_prod\_env" {

  value = [var.env.prod.ip, var.env.prod.loc]

}

output "print\_dev\_env" {

  value = [var.env.dev.ip, var.env.dev.loc]

}

$ terraform init

> Set the value for variable my\_string in Terraform cloud:

  > Go to Terraform Cloud Workspace > Click on variables

> Add variable:

    key: my\_string

    value: MY HCP custom string value

    > Add Variable

$ terraform plan

$ terraform apply -auto-approve

Terraform Registry: https://registry.terraform.io/

Private Registry on Terraform Cloud:

> Create a new - non empty repository on github (initialize with readme.md file):

  > Name: terraform-local-abtftest-sep24

  Format: terraform-<PROVIDER>-<NAME>

> Create a file: main.tf

variable "my\_string" {

  description = "My string input variable"

  type        = string

  default     = "ABHI - Custom String - Default"

}

output "print\_string\_var" {

  value = var.my\_string

}

> Go to Terraform Cloud > Go to Organization > Registry

  > Click Publish > Module

  > Github.com - Connect

  > Choose a repo: terraform-local-abtftest-sep24

  > Confirm selection:

    Module publishing type: Branch

    Branch Name: main

    Module version: 1.0.0

> Enable testing

> Publish module