

<b>Mithun Technologies</b> +91-9980923226	<b>Terraform</b> devopstrainingblr@gmail.com	<b>Author</b>	<b>Mithun Technologies</b>
		<b>Web site</b>	<b>http://mithuntechnologies.com</b>

## **Terraform**

### **Introduction**

Terraform is an open-source, infrastructure as code software (IaC) tool, created by HashiCorp and written in the Go programming language.

Infrastructure as code is the process of managing infrastructure in a file or files, rather than manually configuring resources in a user interface (UI).

Here resources are nothing but virtual machines, Elastic IP, Security Groups, Network Interfaces...

Terraform code is written in the HashiCorp Configuration Language (HCL) in files with the extension .tf

Terraform allows users to use HashiCorp Configuration Language (HCL) to create the files containing definitions of their desired resources on almost any provider (AWS, GCP, Azure, Digital Ocean, OpenStack, etc) and automates the creation of those resources at the time of apply.

**Reference URL:** <https://www.terraform.io/docs/providers/index.html>

### **Advantages of Terraform**

Platform Agnostic  
State Management  
Operator Confidence

### **Difference between Terraform and Cloud Formation**

<b>Terraform</b>	<b>Cloud Formation</b>
<b>Terraform is developed by HashiCorp</b>	<b>Cloud Formation is Developed by AWS</b>
<b>It will work for many Cloud providers like AWS, Azure, GCP, Digital Ocean...</b>	<b>Cloud Formation will support only AWS</b>
<b>Terraform uses HashiCorp Configuration Language (HCL), a language built by HashiCorp. It is fully compatible with JSON.</b>	<b>AWS Cloud Formation utilizes either JSON or YAML. Cloud Formation has a limit of 51,000 bytes for the template body itself</b>

<b>Mithun Technologies</b> +91-9980923226	<b>Terraform</b> devopstrainingblr@gmail.com	<b>Author</b> <b>Web site</b>	<b>Mithun Technologies</b> <a href="http://mithuntechnologies.com">http://mithuntechnologies.com</a>
--	---	----------------------------------	---

## Difference between Terraform and Ansible

### Pre-Requisites

- 1) Any Cloud Provider (AWS, GCP, Azure, Digital Ocean, OpenStack, etc)
- 2) User credentials (Secret Key and Access Key)

Add the following policies to IAM user if you are using AWS as cloud provider.

AmazonEC2FullAccess  
 AmazonS3FullAccess  
 AmazonDynamoDBFullAccess  
 AmazonRDSFullAccess  
 CloudWatchFullAccess  
 IAMFullAccess

### Terraform Installation

Follow my blog for installation.

<https://mithuntechnologies-devops.blogspot.com/2020/02/terraform-installation-linux-server.html>

Follow my below YouTube channel video url.

<https://youtu.be/kxOR-WrK4y8>

### Terraform Examples

**File Name:** AwsEC2InstanceCreation.tf

vim AwsEC2InstanceCreation.tf

The first step in terraform script is typically a provider.

```
provider "aws"{
  region = "us-east-2"
  access_key = "AKIA4UQE3BUQ6GQ3BAFO"
  secret_key = "5LAzj2tYFxFk1NNvmvz0Z1USAOEzDAZHlc6R5wHF"
}
```

This tells Terraform that you are going to be using AWS as your provider and that you want to deploy your infrastructure into the us-east-2 region.

If you use,  
 export AWS\_ACCESS\_KEY\_ID="AKIA4UQE3BUQ6GQ3BAEO"  
 export AWS\_SECRET\_ACCESS\_KEY="5LAzj2tYFxFk1NNvmvz0Z1UASOEzDAZHlc6R5wHF"

No need to mention access\_key and secret\_key keys in provider section.

<b>Mithun Technologies</b> +91-9980923226	<b>Terraform</b> devopstrainingblr@gmail.com	<b>Author</b>	<b>Mithun Technologies</b>
		<b>Web site</b>	<a href="http://mithuntechnologies.com">http://mithuntechnologies.com</a>

```
resource "aws_instance" "MSS" {
  ami = "ami-0a74bfeb190bd404f"
  instance_type = "t2.micro"
  key_name = "mithuntechnologies"
  security_groups = ["launch-wizard-19"]
  tags = {
    Name = "Terraform Server"
  }
}
```

The general syntax for a Terraform resource is:

```
resource "<PROVIDER>_<TYPE>" "<NAME>" {
  [CONFIG ...]
}
```

Here PROVIDER is the name of a provider (e.g., aws), TYPE is the type of resource to create in that provider (e.g., instance), NAME is an identifier you can use throughout the Terraform code to refer to this resource (e.g., my\_instance), and CONFIG consists of one or more arguments that are specific to that resource."

**terraform init:** The terraform init command is used to initialize a working directory containing Terraform configuration files. This is the first command that should be run after writing a new Terraform configuration

**terraform fmt:** The terraform fmt command is used to rewrite Terraform configuration files to a canonical format and style.

**terraform validate:** The terraform validate command validates whether a configuration is syntactically valid or not.

**terraform plan:** The terraform plan command is used to create an execution plan. This command is a convenient way to check whether the execution plan for a set of changes matches your expectations without making any changes to real resources or to the state.

**terraform apply:** terraform apply to actually create the infrastructure on AWS.

**terraform apply -auto-approve:**

**terraform destroy:** The terraform destroy command is used to destroy the Terraform-managed infrastructure.

**terraform show:**

**terraform state list:**

**terraform graph:**

<b>Mithun Technologies</b> +91-9980923226	<b>Terraform</b> devopstrainingblr@gmail.com	<b>Author</b>	<b>Mithun Technologies</b>
		<b>Web site</b>	<b>http://mithuntechnologies.com</b>

### Create Multiple Resources

count = "15"

Using count argument, we can pass the number of resources we need.

### Comments

The Terraform language supports three different syntaxes for comments:

- a) # begins a single-line comment, ending at the end of the line.
- b) // also begins a single-line comment, as an alternative to #.
- c) /\* and \*/ are start and end delimiters for a comment that might span over multiple lines.

### AWS Credentials

The AWS provider offers a various method of providing credentials for authentication. The following methods are supported.

- a) Static credentials
- b) Environment variables
- c) Shared credentials file
- d) EC2 Role

#### Static credentials

Static credentials can be provided by adding an access\_key and secret\_key in-line in the AWS provider block, as follows.

```
provider "aws" {
  region    = "ap-south-1"
  access_key = "AKIA4UQE3BUQ6GQ3BAEO"
  secret_key = "5LAzj2tYFxFk1NNvmvz0Z1UASoEzDAZHlc6R5wHF"
}
```

#### Environment variables

You can provide your credentials via the AWS\_ACCESS\_KEY\_ID and AWS\_SECRET\_ACCESS\_KEY, environment variables, representing your AWS Access Key and AWS Secret Key, respectively.

```
export AWS_ACCESS_KEY_ID="accesskey"
export AWS_SECRET_ACCESS_KEY="secretkey"
export AWS_DEFAULT_REGION="ap-south-1"
```

#### Shared Credentials file

<b>Mithun Technologies</b> +91-9980923226	<b>Terraform</b> devopstrainingblr@gmail.com	<b>Author</b>	<b>Mithun Technologies</b>
		<b>Web site</b>	<b>http://mithuntechnologies.com</b>

You can use an AWS credentials file to specify your credentials. The default location is \$HOME/.aws/credentials on Linux and OS X, or "%USERPROFILE%\aws\credentials" for Windows users. If we fail to detect credentials inline, or in the environment, Terraform will check this location. You can optionally specify a different location in the configuration by providing the shared\_credentials\_file attribute, or in the environment with the AWS\_SHARED\_CREDENTIALS\_FILE variable. This method also supports a profile configuration and matching AWS\_PROFILE environment variable:

```
provider "aws" {
  region      = "ap-south-1"
  shared_credentials_file = "/home/mithun/.aws/creds"
  profile      = "customprofile"
}
```

#### **Possible Error:**

```
[mithun@mithuntechnologies iamuser]$ terraform apply
```

**Error: error configuring Terraform AWS Provider: no valid credential sources for Terraform AWS Provider found.**

Please see <https://registry.terraform.io/providers/hashicorp/aws> for more information about providing credentials.

**Error: NoCredentialProviders: no valid providers in chain. Deprecated.**  
For verbose messaging see aws.Config.CredentialsChainVerboseErrors

```
[mithun@mithuntechnologies iamuser]$ █
```

#### **Solution:**

Provide Valid Access Key and Secret key.

### **Variables**

In real-time world, the project has more variables, it is difficult maintain the variables in terraform script, instead we will put the variables in a separate file called vars.tf as follows.

**File Name:** vars.tf

```
variable "ami" {
  default = "ami-04169656fea786776"
}

variable "instance_type" {
  default = "t2.micro"
}
```

<b>Mithun Technologies</b> +91-9980923226	<b>Terraform</b> devopstrainingblr@gmail.com	<b>Author</b>	<b>Mithun Technologies</b>
		<b>Web site</b>	<a href="http://mithuntechnologies.com">http://mithuntechnologies.com</a>

Now Terraform script looks like below.

```

provider "aws" {
    region = "ap-south-1"
}

resource "aws_instance" "MT" {
    ami = "${var.ami}"
    instance_type = "${var.instance_type}"
    key_name = "mithuntechnologies"

    tags = {
        Name = "Terraform Server - MSS"
    }
}

```

### count parameter

**File Name: count.tf**

```

provider "aws" {
    region = "ap-south-1"
}

resource "aws_instance" "AWSServer" {
    count = "2"
    ami = "ami-052c08d70def0ac62"
    instance_type = "t2.micro"
    key_name = "devopsmssnovbatch"
    security_groups = ["launch-wizard-7"]
    tags = {
        Name = "Terraform Server - MSS"
    }
}

```

---

**File Name: vars.tf**

```

variable "ami" {
    description = "Amazon Machine Image type.."
    default = "ami-04169656fea786776"
}

variable "instance_type" {
    description = "Instance type, weather t2.micro, t2.medium..."
    default = "ap-south-1"
}

variable "instances" {
    description = "Total number of instances which we are going to create"
    default = 2
}

```

<b>Mithun Technologies</b> +91-9980923226	<b>Terraform</b> devopstrainingblr@gmail.com	<b>Author</b>	<b>Mithun Technologies</b>
		<b>Web site</b>	<a href="http://mithuntechnologies.com">http://mithuntechnologies.com</a>

### File Name: count.tf

```

provider "aws" {
  region = "ap-south-1"
}

resource "aws_instance" "AWSServer" {
  count = "${var.instances}"
  ami = "ami-052c08d70def0ac62"
  instance_type = "t2.micro"
  key_name = "devopsmssnovbatch"
  security_groups = ["launch-wizard-7"]
  tags = {
    Name = "Terraform Server - ${count.index}"
  }
}

```

---

### user\_data

The user\_data only runs at instance launch time.  
It will not execute after creation on instance.

### FileName: var.tf

```

variable "ami"{
  description = "AMI Name..."
  default = "ami-052c08d70def0ac62"
}

variable "instance_type"{
  description = "Instance Type..."
  default = "t2.micro"
}

```

### FileName: userdata.tf

```

provider "aws" {
  region = "ap-south-1"
}

resource "aws_instance" "AWSServer" {
  ami = "${var.ami}"
  instance_type = "${var.instance_type}"
  key_name = "devopsmssnovbatch"
  security_groups = ["launch-wizard-7"]
  user_data = "${file("installApacheServer.sh")}"
  tags = {
    Name = "Terraform Server - MSS "
  }
}

```

<b>Mithun Technologies</b> +91-9980923226	<b>Terraform</b> devopstrainingblr@gmail.com	<b>Author</b> <b>Web site</b>	<b>Mithun Technologies</b> <a href="http://mithuntechnologies.com">http://mithuntechnologies.com</a>
--	---	----------------------------------	---

## installApacheServer.sh

```
#!/bin/bash
```

```
sudo yum install httpd -y
sudo systemctl enable httpd
sudo systemctl start httpd
```

---

A **security group** acts as a virtual firewall for your instance to control incoming and outgoing traffic. Inbound rules control the incoming traffic to your instance, and outbound rules control the outgoing traffic from your instance. ... **Security groups** are associated with network interfaces.

```
provider "aws" {
  region = "ap-south-1"
}

resource "aws_security_group" "SecurityGroupsMSS" {
  name      = "SecurityGroupsMSS"
  description = "Allow TLS inbound traffic"
  vpc_id    = "vpc-27ec054c"

  ingress {
    description = "TLS from VPC"
    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 = "SecurityGroupsMSS"
  }
}
```



<b>Mithun Technologies</b> +91-9980923226	<b>Terraform</b> devopstrainingblr@gmail.com	<b>Author</b>	<b>Mithun Technologies</b>
		<b>Web site</b>	<a href="http://mithuntechnologies.com">http://mithuntechnologies.com</a>

## Identity and Access Management (IAM)

AWS Identity and Access Management (IAM) enables you to manage access to AWS services and resources securely. Using IAM, you can create and manage AWS users and groups, and use permissions to allow and deny their access to AWS resources. IAM is a feature of your AWS account offered at no additional charge.

### FileName: iam.tf

```
provider "aws" {
  region = "ap-south-1"
}

resource "aws_iam_user" "IAMUSERS" {
  name = "mithuntechnologies"
}
```

---

### FileName: vars.tf

```
variable "user_names" {
  description = "Create IAM users"
  type        = list(string)
  default     = ["mithuntechnologies"]
}
```

### FileName: main.tf

```
provider "aws" {
  region = "ap-south-1"
}

resource "aws_iam_user" "IAMUsers" {
  count = length(var.user_names)
  name  = var.user_names[count.index]
}
```

Mithun Technologies +91-9980923226	Terraform devopstrainingblr@gmail.com	Author	Mithun Technologies
		Web site	http://mithuntechnologies.com

### Possible Error:

```
[mithun@mithuntechnologies iamuser]$ terraform apply
```

### Error: Invalid resource name

```
on main.tf line 5, in resource "aws_iam_user" "AWS IAM Users":
5: resource "aws_iam_user" "AWS IAM Users" {
```

A name must start with a letter or underscore and may contain only letters, digits, underscores, and dashes.

```
[mithun@mithuntechnologies iamuser]$ █
```

### Solution:

Give the Name without space like AWSIAMUser.

### Possible Error:

```
[mithun@mithuntechnologies iamuser]$ terraform apply
aws_iam_user.IAMUsers[0]: Refreshing state... [id=mithunreddy]
```

```
Error: Error reading IAM User mithunreddy: AccessDenied: User: arn:aws:iam::732957639877:user/mithuntechnologies is not authorized to perform: iam:GetUser on resource: user mithunreddy
status code: 403, request id: fe1e1d87-9a32-4180-bc9d-a39dc7f9cab8
```

```
[mithun@mithuntechnologies iamuser]$ █
```

### Solution:

Attach the **IAMFullAccess** to the user.

<b>Mithun Technologies</b> +91-9980923226	<b>Terraform</b> devopstrainingblr@gmail.com	<b>Author</b>	<b>Mithun Technologies</b>
		<b>Web site</b>	<b>http://mithuntechnologies.com</b>

### **Resources**

<https://www.terraform.io/docs/providers/aws/d/ami.html>  
<https://learn.hashicorp.com/terraform/getting-started/intro>  
<https://www.terraform.io/docs/providers/aws/index.html>  
<https://www.terraform.io/intro/index.html>