Contents

[Installation 2](#_Toc187574063)

[Windows 2](#_Toc187574064)

[Linux 2](#_Toc187574065)

[Verify Installation 3](#_Toc187574066)

[Configure Terraform Provider 3](#_Toc187574067)

[Define provider 3](#_Toc187574068)

[Initialize terraform 4](#_Toc187574069)

[Demo 4](#_Toc187574070)

[Create S3 Bucket 4](#_Toc187574071)

[Apply resource 4](#_Toc187574072)

[EC2 Instance 5](#_Toc187574073)

[Apply resource 5](#_Toc187574074)

[Variables 6](#_Toc187574075)

[Local 6](#_Toc187574076)

[Input 6](#_Toc187574077)

[Output 6](#_Toc187574078)

[Terraform commands of interest 7](#_Toc187574079)

[State Management 7](#_Toc187574080)

[State Backend 7](#_Toc187574081)

[Use an S3 backend 7](#_Toc187574082)

[Migrate State 7](#_Toc187574083)

[Create a resource using default backend (current dir) 7](#_Toc187574084)

[Add a Configuration for S3 backend 8](#_Toc187574085)

[Migrate the resources from default backend to S3 backend 8](#_Toc187574086)

[Provisioners 9](#_Toc187574087)

[local-exec 9](#_Toc187574088)

[remote-exec 9](#_Toc187574089)

[file 9](#_Toc187574090)

[Workspaces 10](#_Toc187574091)

[commands 10](#_Toc187574092)

[Terraform manifest Example 10](#_Toc187574093)

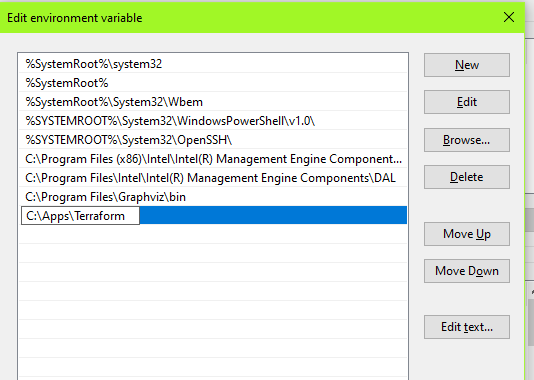
[check out the different statefiles 11](#_Toc187574094)

[Exercise: 12](#_Toc187574095)

# Installation

## Windows

1. To install Terraform, find the [appropriate package](https://developer.hashicorp.com/terraform/install) for your system and download it as a zip archive.
2. Extract the .exe from the ZIP file to a folder eg C:\Apps\Terraform
3. Copy this path location
4. Add the folder location to your PATH variable,
   1. Control Panel -> System -> System settings -> Environment Variables
   2. In System Variables, select Path > edit > new > Enter the location of the Terraform .exe, eg C:\Apps\Terraform
   3. click OK



1. Open a new CMD/PowerShell and the Terraform command should work

## Linux

Follow the Installation guide from the URL

<https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli>.

*Note: Please choose Linux Tab in the guideA screenshot of a computer

Description automatically generated*

## Verify Installation

Once finished, verify that terraform is installed properly:

|  |
| --- |
| terraform version |

Expected Output (Version Numbers may differ)

|  |
| --- |
| Terraform v1.8 on linux\_amd64 |

## Configure Terraform Provider

### Define provider

1. Create a directory terraform\_demo\_1 and cd into it

|  |
| --- |
| mkdir ~/terraform\_demo\_1 cd ~/terraform\_demo\_1/ |

1. Create terraform\_demo\_1/main.tf file with [aws provider information](https://registry.terraform.io/providers/hashicorp/aws/latest/docs)

terraform {

  required\_providers {

    aws = {

      source = "hashicorp/aws"

      version = "5.83.1"

    }

  }

}

provider "aws" {

  # Configuration options

  # eu-central-1

  region = "us-east-1"

  access\_key = "AKIAYS2NUG2MT3QEJMG3"

  secret\_key = "fjemIfmX3y20iUn838tCaLt/9Lm5UFNTnp6NXE+o"

### Initialize terraform

|  |
| --- |
| terraform init |

expected output:

A screenshot of a computer program

Description automatically generated

# Demo

## Create S3 Bucket

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

  bucket = "2023-08-19cdgvsdfbsbgfdb"

  tags = {

    Name        = "My bucket"

    Environment = "Dev"

  }

}

### Apply resource

terraform apply

## EC2 Instance

resource "aws\_instance" "ec2-1" {

  ami = "ami-080e1f13689e07408"

  instance\_type = "t2.micro"

  tags = {

    Name = "my-first-ec2-foo"

  }

}

### Apply resource

terraform apply

# Variables

## Local

NOTE: The value of the locals **can not** be overridden

# local variables

locals {

  bucket-tag = "foor"

}

resource "aws\_s3\_bucket" "example-1" {

  bucket = "2023-08-19cdgvsdfbsbgfdb"

  tags = {

    Name        = local.bucket-tag

  }

}

## Input

# input variables

# can be overridden using command: terraform apply -var env=pre-prod

variable "env" {

    default =   "prod"

}

resource "aws\_s3\_bucket" "example-1" {

  bucket = "2023-08-19cdgvsdfbsbgfdb"

  tags = {

    Environment = var.env

  }

}

Pass value to input variable: terraform apply -var env=pre-prod

## Output

resource "aws\_s3\_bucket" "example-1" {

  bucket = "2023-08-19cdgvsdfbsbgfdb"

  tags = {

    Environment = var.env

  }

}

# Output variable

# can be used to print a resource's property on stdout

output "bucket-id" {

  value = aws\_s3\_bucket.example-1.id

}

# Terraform commands of interest

# Format files for enhanced readability

terraform fmt

# State Management

# Show all resources

terraform show

# List all resource names currently managed by TF

terraform state list

# List details of a particular resource

terraform state show <resourceref>

# State Backend

## Use an S3 backend

terraform {

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "5.83.1"

    }

  }

  backend "s3" {

    region     = "us-east-1"

    access\_key = "AKIAWFHH2Q6WG4JWL6O3"

    secret\_key = "+ba9PbKjzpvDM4DxpeFQSj2zyJTYHbwgE4MD5qPn"

    bucket     = "tf-backend-616465156465151515"

    key        = "tf-demo-2"

  }

}

## Migrate State

### Create a resource using default backend (current dir)

terraform {

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "5.68.0"

    }

  }

}

provider "aws" {

  region     = "us-east-1"

  access\_key = "AKIA4YCZP4FW7XNXADGY"

  secret\_key = "uMKdlLkG4zLyIbRZeLgJDfUQOdI1nMZhZkfjbKdd"

}

resource "aws\_instance" "inst-1" {

  ami           = "ami-0e86e20dae9224db8"

  instance\_type = "t2.micro"

}

### Add a Configuration for S3 backend

terraform {

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "5.68.0"

    }

  }

  backend "s3" {

    bucket = "123ccvvffgg-backend"

    key    = "tf-backend-1"

    region = "us-east-1"

    access\_key = "AKIA6GBMA5GVID6OF5OP"

    secret\_key = "fdbjy/qwR1Pn0wxZZxx07/LWoCochbt5tlFJnBGW"

  }

}

provider "aws" {

  region     = "us-east-1"

  access\_key = "AKIA4YCZP4FW7XNXADGY"

  secret\_key = "uMKdlLkG4zLyIbRZeLgJDfUQOdI1nMZhZkfjbKdd"

}

resource "aws\_instance" "inst-1" {

  ami           = "ami-0e86e20dae9224db8"

  instance\_type = "t2.micro"

}

### Migrate the resources from default backend to S3 backend

terraform init -migrate-state

#### Output

Initializing the backend...

Do you want to copy existing state to the new backend?

  Pre-existing state was found while migrating the previous "local" backend to the

  newly configured "s3" backend. No existing state was found in the newly

  configured "s3" backend. Do you want to copy this state to the new "s3"

  backend? Enter "yes" to copy and "no" to start with an empty state.

  Enter a value: yes

Successfully configured the backend "s3"! Terraform will automatically

use this backend unless the backend configuration changes.

# Provisioners

## local-exec

resource "aws\_s3\_bucket" "bucket-1" {

  bucket = "tiutiygj875435uyyiuv"

  provisioner "local-exec" {

    command = "echo Bucket has domain: ${self.bucket\_regional\_domain\_name}"

  }

}

## remote-exec

resource "aws\_s3\_bucket" "bucket-1" {

  bucket = "tiutiygj875435uyyiuv"

  provisioner "remote-exec" {

    inline = [ "echo Hello World", "who am i" , "apt update" ]

    connection {

      host     = "172.65.58.69"

      user     = "ubuntu"

      password = "secretpass"

      type     = "ssh"

    }

  }

}

## file

resource "aws\_s3\_bucket" "bucket-1" {

  bucket = "tiutiygj875435uyyiuv"

  provisioner "file" {

    source      = "/path/to/file/on/local/tf/server/file.txt"

    destination = "/path/to/file/on/a/remote/server/file.txt"

    connection {

      host     = "172.65.58.69"

      user     = "ubuntu"

      password = "secretpass"

      type     = "ssh"

    }

  }

}

# Workspaces

## commands

# list all workspaces

terraform workspace list

# Create new workspace called production

terraform workspace new production

# Create new workspace called staging

terraform workspace new staging

# list all workspaces

terraform workspace list

# switch between workspaces

terraform workspace select production

terraform workspace select staging

## Terraform manifest Example

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

  ami = "ami-053b0d53c279acc90"

  instance\_type = "t2.micro"

  tags = {

    Name = "web-${terraform.workspace}"

  }

}

terraform {

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "5.45.0"

    }

  }

  backend "s3" {

    bucket = "somebucketname-vcdcvfeiuhefv-23443"

    key = "shared-statefile"

    region = "us-east-1"

    access\_key = "AKIA6GBMA5GVID6OF5OP"

    secret\_key = "fdbjy/qwR1Pn0wxZZxx07/LWoCochbt5tlFJnBGW"

  }

}

provider "aws" {

  # Configuration options

  # eu-central-1

  region     = "us-east-1"

    access\_key = "AKIA6GBMA5GVID6OF5OP"

    secret\_key = "fdbjy/qwR1Pn0wxZZxx07/LWoCochbt5tlFJnBGW"

}

## check out the different statefiles

A screenshot of a computer

Description automatically generated

# Resource Lifecycle

resource "aws\_instance" "inst-1" {

  ami = "ami-05576a079321f21f8"

  instance\_type = "t2.micro"

  tags = {

    Name = terraform.workspace

    env  = terraform.workspace

  }

 lifecycle {

   prevent\_destroy = true

   ignore\_changes = [ tags ]

   create\_before\_destroy = true

 }

}

# Exercise:

1. Create an S3 bucket using terraform
2. Create an EC2 Instance using TF
3. Write TF code to create an EC2 instance. Ensure that the user can provide the ami ID at runtime (during apply)
   1. Solution

variable "ami\_name" {

  default = "prod"

}

resource "aws\_instance" "ec2-1" {

  ami           = var.ami\_name

  instance\_type = "t2.micro"

  tags = {

    Name = "my-first-ec2-foo"

  }

}

terraform apply -var ec2-ami=ami-05576a079321f21f8