**Lab Manual- Manage Azure Stoarge Infrastcrture Using Terraform**

Table of Contents

[1 OBJECTIVE 3](#_Toc113463654)

[2 What is Terraform 3](#_Toc113463655)

[3 PRE-REQUISISTE 4](#_Toc113463656)

[4 Create Storage Account with Terraform 4](#_Toc113463657)

[5 Update Storage Account with Blob and Queue 8](#_Toc113463658)

# OBJECTIVE

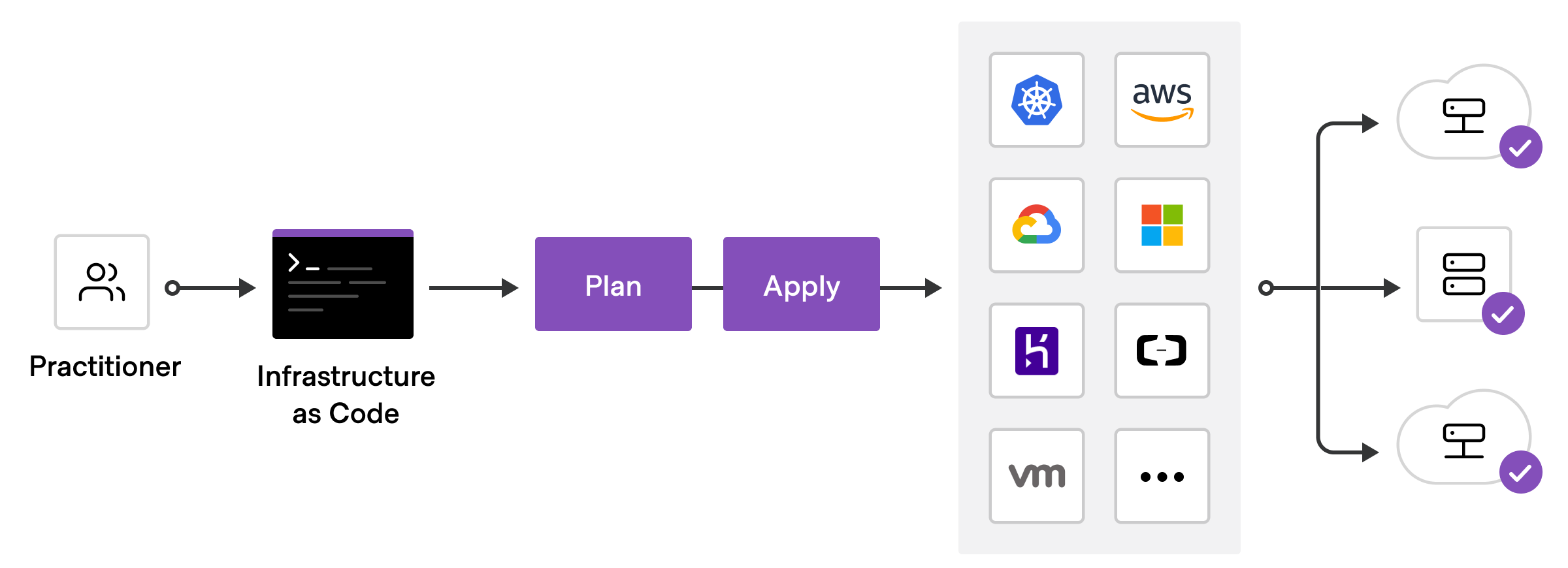
Terraform Azure providers enable you to manage all of your Azure infrastructure using the same declarative syntax and tooling. Using these providers you can:

* Provision core platform capabilities such as management groups, policies, users, groups, and policies. For more information, see [Terraform implementation of Cloud Adoption Framework Enterprise-scale](https://github.com/Azure/terraform-azurerm-caf-enterprise-scale#readme).
* Provision Azure DevOps Projects and pipelines to automate regular infrastructure and application deployments.
* Provision Azure resources required by your applications.

# What is Terraform

Terraform is HashiCorp's infrastructure as code tool. It lets you define resources and infrastructure in human-readable, declarative configuration files, and manages your infrastructure's lifecycle. Using Terraform has several advantages over manually managing your infrastructure:

* Terraform can manage infrastructure on multiple cloud platforms.
* The human-readable configuration language helps you write infrastructure code quickly.
* Terraform's state allows you to track resource changes throughout your deployments.
* You can commit your configurations to version control to safely collaborate on infrastructure.
* To deploy infrastructure with Terraform:
  + **Scope** - Identify the infrastructure for your project.
  + **Author** - Write the configuration for your infrastructure.
  + **Initialize** - Install the plugins Terraform needs to manage the infrastructure.
  + **Plan** - Preview the changes Terraform will make to match your configuration.
  + **Apply** - Make the planned changes.



# PRE-REQUISISTE

* Accounts in Azure
* A local Computer with 4 CPU, 16 GB RAM, 200 GB disk space
* An Azure tenant and access to a subscription, like **Owner** or **Contributor** rights.
* VS Code or other IDE. However, VS Code has a [Terraform extension](https://docs.microsoft.com/en-us/azure/developer/terraform/configure-vs-code-extension-for-terraform) to improve the authoring process.
* Terraform open-source command-line interface
* Azure CLI ([download](https://docs.microsoft.com/en-us/cli/azure/install-azure-cli)). This tutorial uses version 2.32.0.

# Create Storage Account with Terraform

1. Create Another Folder and Create Main.tf file inside it.

terraform {

required\_providers {

azurerm = {

source = "hashicorp/azurerm"

version = "~>2.0"

}

}

}

provider "azurerm" {

features {}

subscription\_id = "c49a4614-f368-4f5b-b72a-88bf82d12229"

tenant\_id = "be04fbd5-6b00-412c-a86c-ca105b5cce90"

client\_id = "0b381472-3197-49d4-a324-f1a96a23c8a7"

client\_secret = "QYd8Q~0fCdFMQW\_AtMZoNaiXHlGSbdGOq1fbTcui"

}

resource "azurerm\_resource\_group" "example" {

name = "aditi89io9"

location = "eastus"

}

resource "azurerm\_storage\_account" "example" {

name = "ctrls5678yt5ws"

resource\_group\_name = azurerm\_resource\_group.example.name

location = azurerm\_resource\_group.example.location

account\_tier = "Standard"

account\_replication\_type = "GRS"

tags = {

environment = "staging"

}

}

1. Run [terraform init](https://www.terraform.io/docs/commands/init.html) to initialize the Terraform deployment. This command downloads the Azure modules required to manage your Azure resources.

**terraform init**

Text

Description automatically generated

1. Run [terraform plan](https://www.terraform.io/docs/commands/plan.html) to create an execution plan.

**terraform plan**

Text

Description automatically generated

1. Run [terraform apply](https://www.terraform.io/docs/commands/apply.html) to apply the execution plan to your cloud infrastructure.

**terraform Apply**

Text

Description automatically generatedGraphical user interface, text, application

Description automatically generated

1. Go to Azuree Portal and Verify

**terraform Apply**

Graphical user interface, text, application

Description automatically generated

# Update Storage Account with Blob and Queue

1. Update the Main.tf created for Storage account AS Below

terraform {

required\_providers {

azurerm = {

source = "hashicorp/azurerm"

version = "~>2.0"

}

}

}

provider "azurerm" {

features {}

subscription\_id = "c49a4614-f368-4f5b-b72a-88bf82d12229"

tenant\_id = "be04fbd5-6b00-412c-a86c-ca105b5cce90"

client\_id = "0b381472-3197-49d4-a324-f1a96a23c8a7"

client\_secret = "QYd8Q~0fCdFMQW\_AtMZoNaiXHlGSbdGOq1fbTcui"

}

resource "azurerm\_resource\_group" "example" {

name = "aditi89io9"

location = "eastus"

}

resource "azurerm\_storage\_account" "example" {

name = "ctrls5678yt5ws"

resource\_group\_name = azurerm\_resource\_group.example.name

location = azurerm\_resource\_group.example.location

account\_tier = "Standard"

account\_replication\_type = "GRS"

tags = {

environment = "staging"

}

}

resource "azurerm\_storage\_container" "example" {

name = "content"

storage\_account\_name = azurerm\_storage\_account.example.name

container\_access\_type = "private"

}

resource "azurerm\_storage\_queue" "example" {

name = "queue1"

storage\_account\_name = azurerm\_storage\_account.example.name

}

1. Run terraform Plan

Text

Description automatically generated

1. Run Terraform Apply

Text

Description automatically generated

1. Go to the Portal and Check Both Container and Queue

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated