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

Table of Contents

[1 OBJECTIVE 3](#_Toc113464161)

[2 What is Terraform 3](#_Toc113464162)

[3 PRE-REQUISISTE 4](#_Toc113464163)

[4 Setup Terraform 4](#_Toc113464164)

[5 Setup Azure Service Princiapl 6](#_Toc113464165)

[6 Terraform to Create Resource Group. 7](#_Toc113464166)

# OBJECTIVE

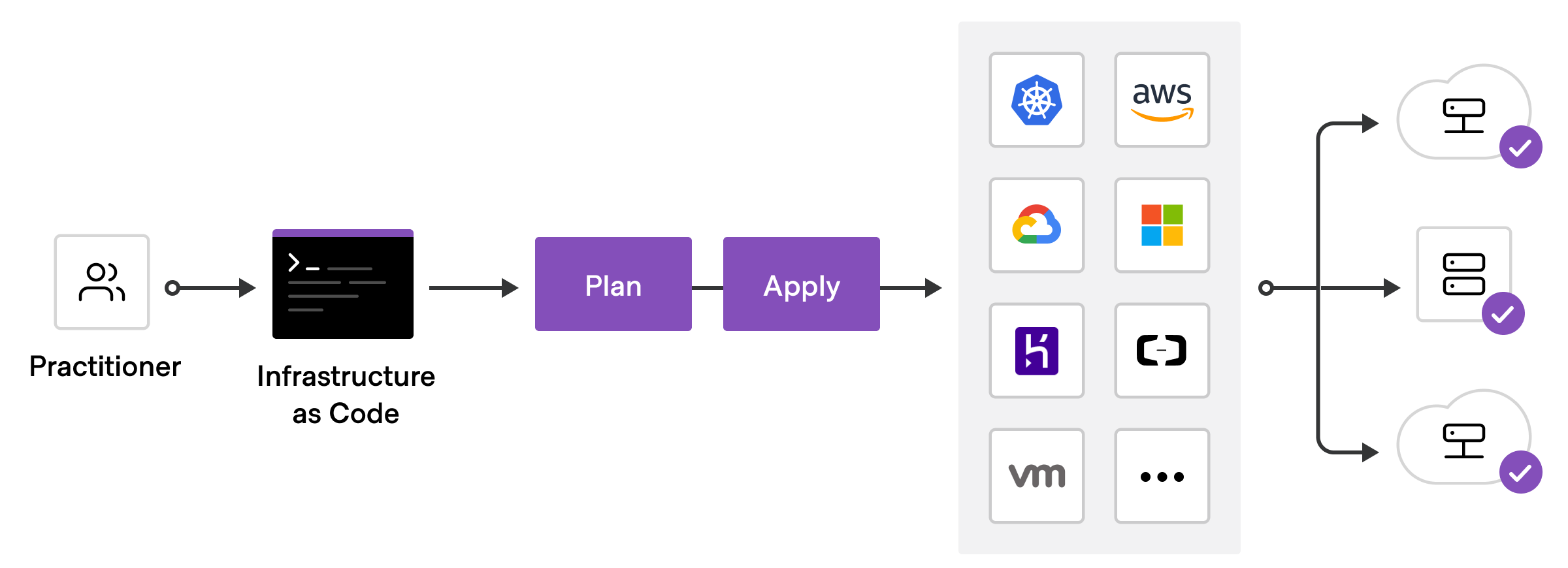
Terraform Azure providers enable you to manage all of your Azure infrastructure using the same declarative syntax and tooling. Using these providers you can:

* 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.

# Setup Terraform

1. To use Terraform you will need to install it. HashiCorp distributes Terraform as a [binary package](https://www.terraform.io/downloads.html). You can also install Terraform using popular package managers.

<https://www.terraform.io/downloads>



1. After downloading Terraform, unzip the package

Chart

Description automatically generated

1. Copy the Executable to some Folder in Local drive

Graphical user interface, text

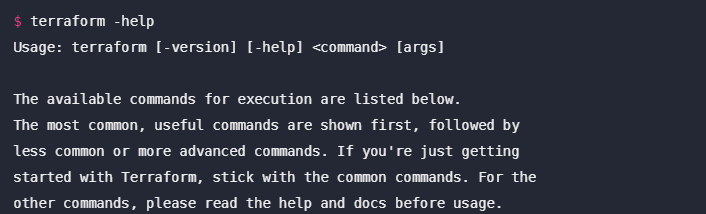
Description automatically generated

1. Setup the Path in System Variable. Add the folder location to your PATH variable, eg: **Control Panel -> System -> System settings -> Environment Variables**

A screenshot of a computer

Description automatically generated with medium confidence

1. Verify that the installation worked by opening a new terminal session and listing Terraform's available subcommands



# Setup Azure Service Princiapl

A Service Principal is an application within Azure Active Directory with the authentication tokens Terraform needs to perform actions on your behalf

1. Type below Command to Create Service Principal in Azure Cloud Shell

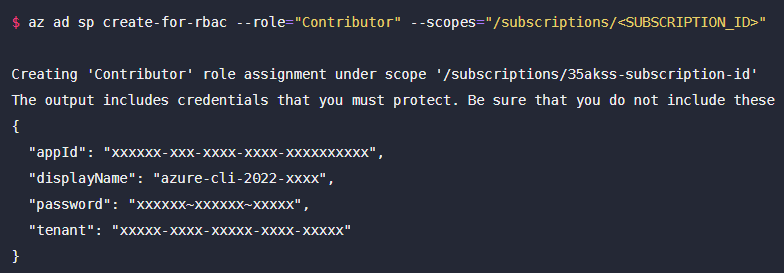
az ad sp create-for-rbac --name ServicePrincipalName

Graphical user interface, text, application

Description automatically generated

In case of Multiple Subscription you can also type below command

az ad sp create-for-rbac --role="Contributor" --scopes="/subscriptions/<SUBSCRIPTION\_ID>"



1. Copy the Output to Text Editor (Notepad)

# Terraform to Create Resource Group.

1. Create a directory in which to test the sample Terraform code and make it the current directory.
2. Create a file named main.tf and insert the following code:

=====Main.tf===

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 = "e6B8Q~37U30s\_tKTKi7ExeoU41G1O~KUT6tykcJz"

}

resource "azurerm\_resource\_group" "terraformctrls56" {

name = var.RGName

location = var.location

}

1. Create a file named **variables.tf** and insert the following code:

variable "RGName" {

type = string

description = "(Required) The name of the resource group. Must be unique on your Azure subscription"

}

variable "location" {

type = string

description = "Location of Resource group"

}

Graphical user interface, text, application, email

Description automatically generated

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.

[ Note : The terraform plan command creates an execution plan, but doesn't execute it. Instead, it determines what actions are necessary to create the configuration specified in your configuration files. This pattern allows you to verify whether the execution plan matches your expectations before making any changes to actual resources.]

**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 generated

A screenshot of a computer

Description automatically generated with medium confidence

1. Verify the output

**az group show --name <resource\_group\_name>**

1. When you no longer need the resources created via Terraform, do the following steps:

**Terraform destroy**

Text

Description automatically generated

Text

Description automatically generated