**Lab Manual- Create and Update Azure AKS Cluster**

Table of Contents

[1 OBJECTIVE 3](#_Toc183860729)

[2 PRE-REQUISISTE 4](#_Toc183860730)

[3 Create AKS Cluster with Terraform 4](#_Toc183860731)

[4 Update Resource Group with Tag 11](#_Toc183860732)

[5 Delete AKS Cluster and Resources 14](#_Toc183860733)

# OBJECTIVE

The **objective** of the lab is to demonstrate how to deploy an **Azure Kubernetes Service (AKS) cluster** using **Terraform**. Specifically, it covers:

**1. Setting Up Infrastructure with Terraform**

* Learn how to use Terraform to create and manage Azure resources:
  + **Resource Group**: A container for all Azure resources.
  + **AKS Cluster**: Deploy and manage a Kubernetes cluster.
  + **Networking**: Set up virtual networks and subnets for secure communication between the AKS cluster components.

**2. Linking Resources**

* Understand how Terraform links different resources:
  + The **AKS cluster** is linked to a **resource group** and a **subnet** within a virtual network.
  + Authentication for the AKS cluster uses a **service principal**.
  + Outputs like the kube\_config are used to connect and manage the Kubernetes cluster.

**3. Working with Terraform State and Lock Files**

* Explore how Terraform uses state files to track infrastructure.
* Learn about the .terraform.lock.hcl file to manage provider versions for consistent deployments.

**4. Understanding Kubernetes Node Pools**

* Define and configure a **default node pool** for the AKS cluster:
  + Adjust the vm\_size based on workload requirements (e.g., ensuring a VM with >2 cores and >4GB memory for system node pools).
* Optimize the cluster for production or development environments.

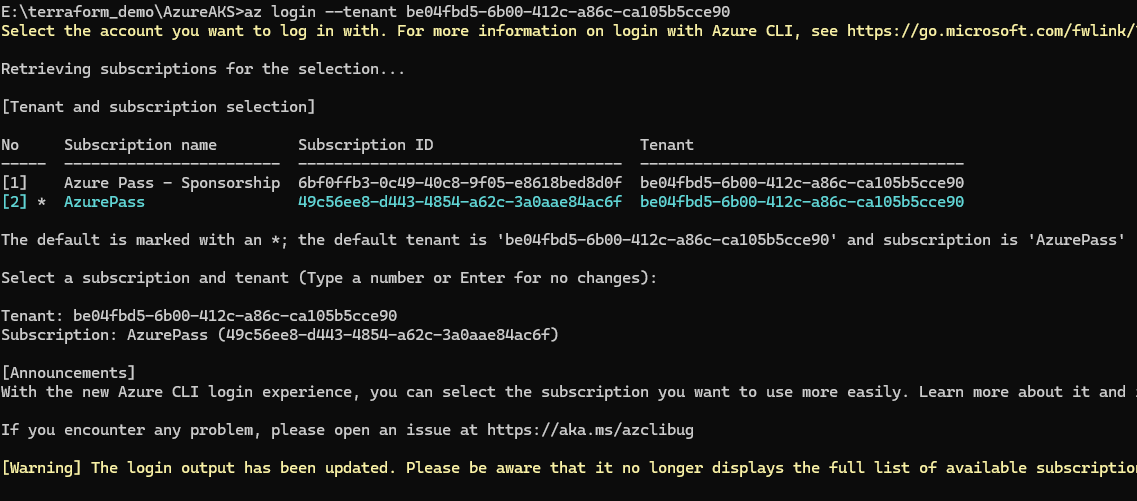
**7. Objective for Real-World Application**

* By completing this lab, participants will:
  + Be comfortable deploying and managing Kubernetes clusters on Azure using Terraform.
  + Understand Terraform's role in infrastructure as code (IaC) workflows.
  + Gain insights into Azure networking, authentication, and Kubernetes operations.

# PRE-REQUISISTE

* + 1. Login on azure with cli

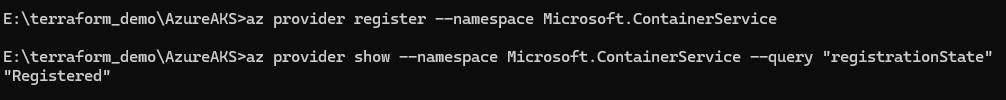
**az login --tenant be04fbd5-6b00-412c-a86c-ca105b5cce90**

****

* + 1. Register container service with subscription

**az provider register --namespace Microsoft.ContainerService**

**az provider show --namespace Microsoft.ContainerService --query "registrationState"**



# Create AKS Cluster with Terraform

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

terraform {

required\_providers {

azurerm = {

source = "hashicorp/azurerm"

version = "~> 3.70" # Adjust the version as needed

}

}

}

provider "azurerm" {

features {} # Required for the Azure provider to work

}

# Define the resource group for the AKS cluster

resource "azurerm\_resource\_group" "aks" {

name = "aks-resource-group"

location = "West US 2"

}

# Define the AKS cluster using kubenet networking

resource "azurerm\_kubernetes\_cluster" "aks\_cluster" {

name = "aks-cluster"

location = azurerm\_resource\_group.aks.location

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

dns\_prefix = "akscluster"

default\_node\_pool {

name = "default"

node\_count = 1

vm\_size = "Standard\_DS2\_v2" # Update to a valid VM SKU with >=2 cores and >=4GB RAM

}

# Specify the service principal for authentication

service\_principal {

client\_id = "0b381472-3197-49d4-a324-f1a96a23c8a7" # Service principal client ID

client\_secret = "PiX8Q~CqWaqiC4Bh~rALxAl56VFmkllYgNJmHc-D" # Service principal client secret

}

# Optional settings like Azure Active Directory integration or monitoring can be added here

tags = {

environment = "dev"

//Project = "demoProject" # Added new tag

}

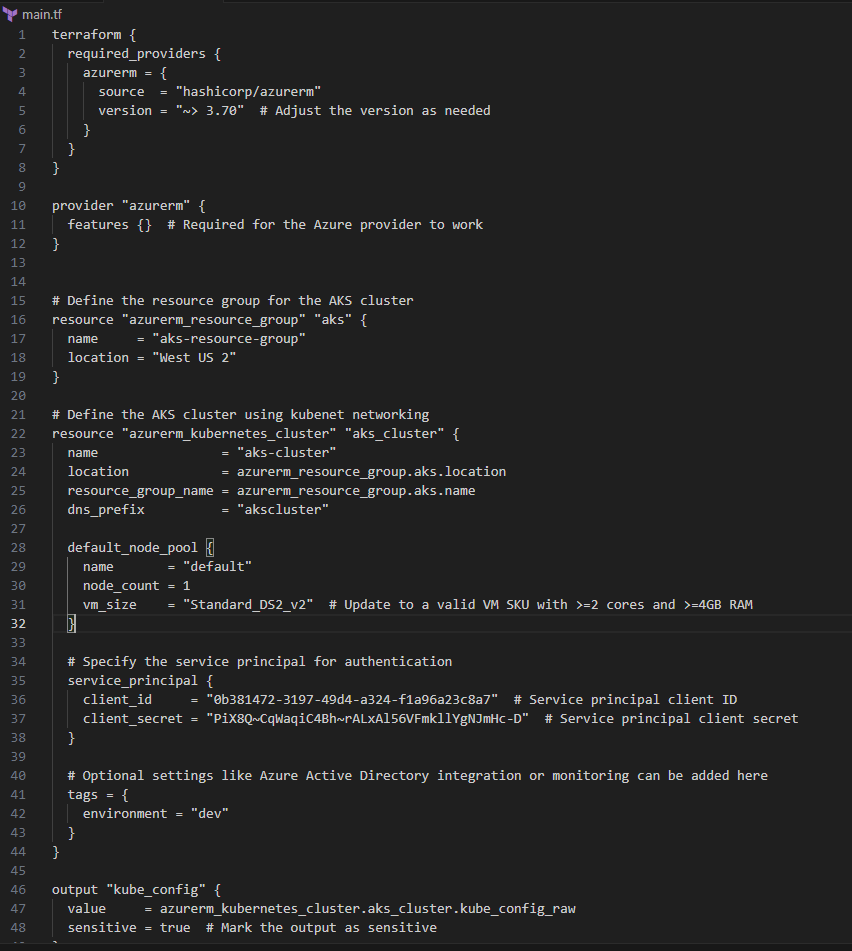
}

output "kube\_config" {

value = azurerm\_kubernetes\_cluster.aks\_cluster.kube\_config\_raw

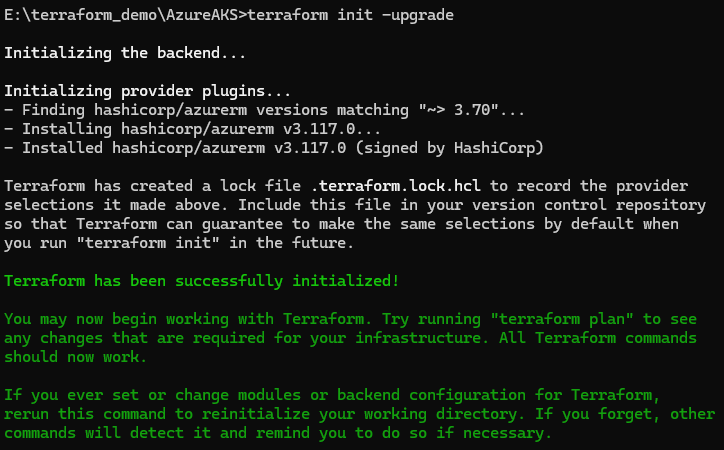
sensitive = true # Mark the output as sensitive

}



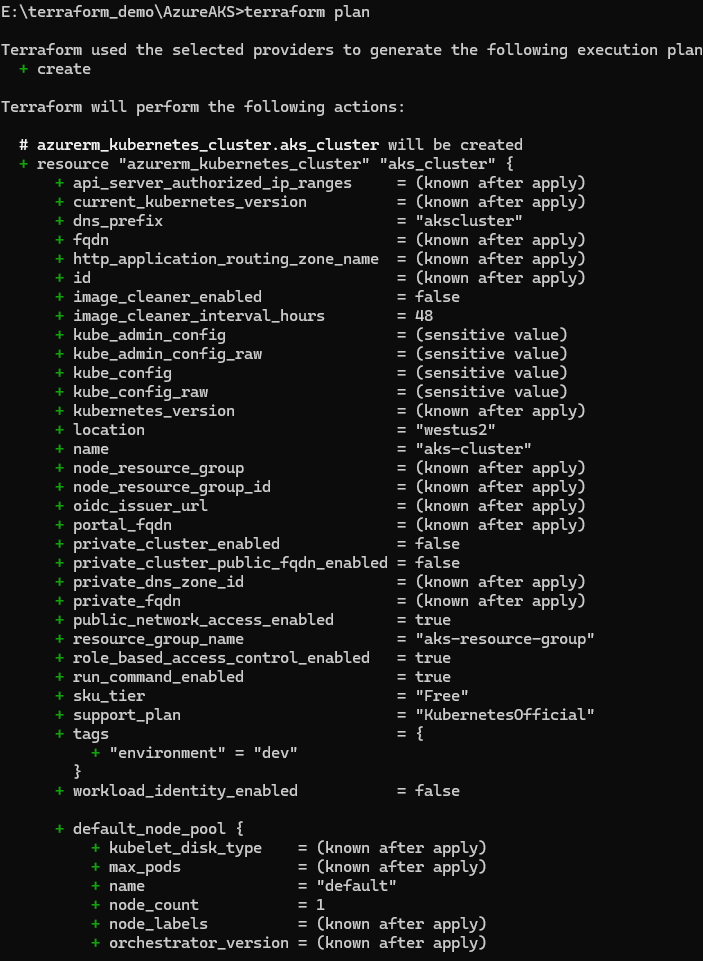
1. Run [terraform init](https://www.terraform.io/docs/commands/init.html) with upgreade to initialize the Terraform deployment. This command downloads the Azure modules required to manage your Azure resources. (Ensures that Terraform uses the latest versions of provider plugins that meet the version constraints specified in your configuration (**~> 3.70** for azurerm in your case).

**terraform init -upgrade**

****

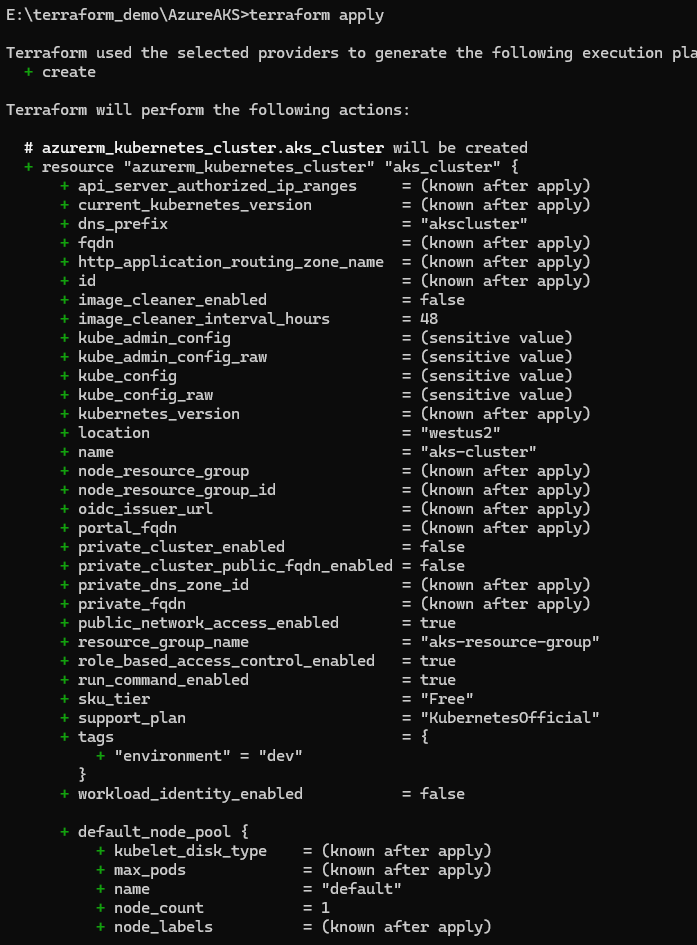
1. Run [terraform plan](https://www.terraform.io/docs/commands/plan.html) to create an execution plan.

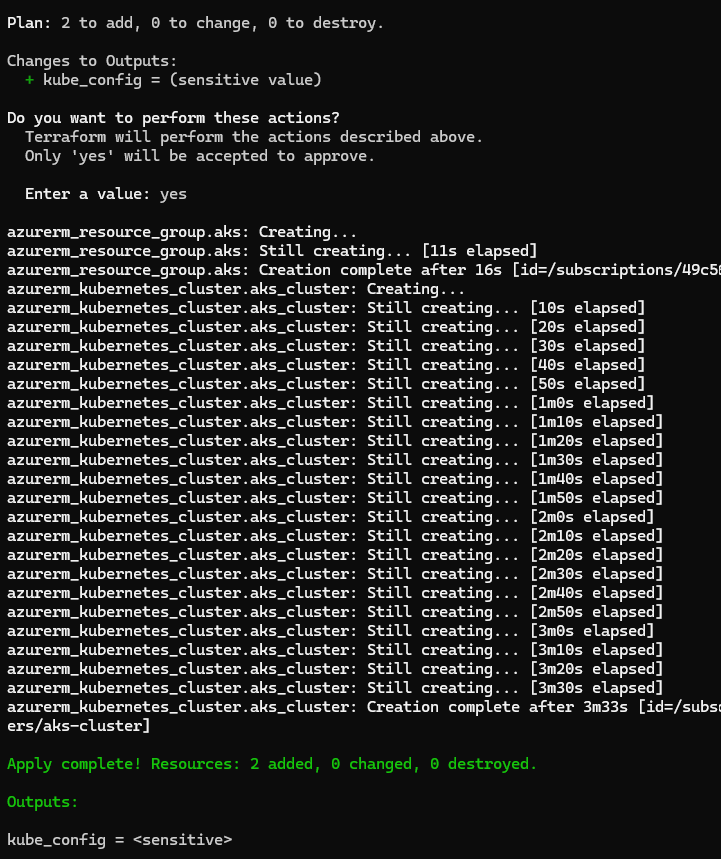
**terraform plan**



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

**terraform Apply**





1. Go to Azuree Portal and Verify

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Update Resource Group with Tag

1. Update the Main.tf created for Resource Group AS highlighted Below. **Uncomment** the **Tag Part**

terraform {

required\_providers {

azurerm = {

source = "hashicorp/azurerm"

version = "~> 3.70" # Adjust the version as needed

}

}

}

provider "azurerm" {

features {} # Required for the Azure provider to work

}

# Define the resource group for the AKS cluster

resource "azurerm\_resource\_group" "aks" {

name = "aks-resource-group"

location = "West US 2"

}

# Define the AKS cluster using kubenet networking

resource "azurerm\_kubernetes\_cluster" "aks\_cluster" {

name = "aks-cluster"

location = azurerm\_resource\_group.aks.location

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

dns\_prefix = "akscluster"

default\_node\_pool {

name = "default"

node\_count = 1

vm\_size = "Standard\_DS2\_v2" # Update to a valid VM SKU with >=2 cores and >=4GB RAM

}

# Specify the service principal for authentication

service\_principal {

client\_id = "0b381472-3197-49d4-a324-f1a96a23c8a7" # Service principal client ID

client\_secret = "PiX8Q~CqWaqiC4Bh~rALxAl56VFmkllYgNJmHc-D" # Service principal client secret

}

# Optional settings like Azure Active Directory integration or monitoring can be added here

tags = {

environment = "dev"

Project = "demoProject" # Added new tag

}

}

output "kube\_config" {

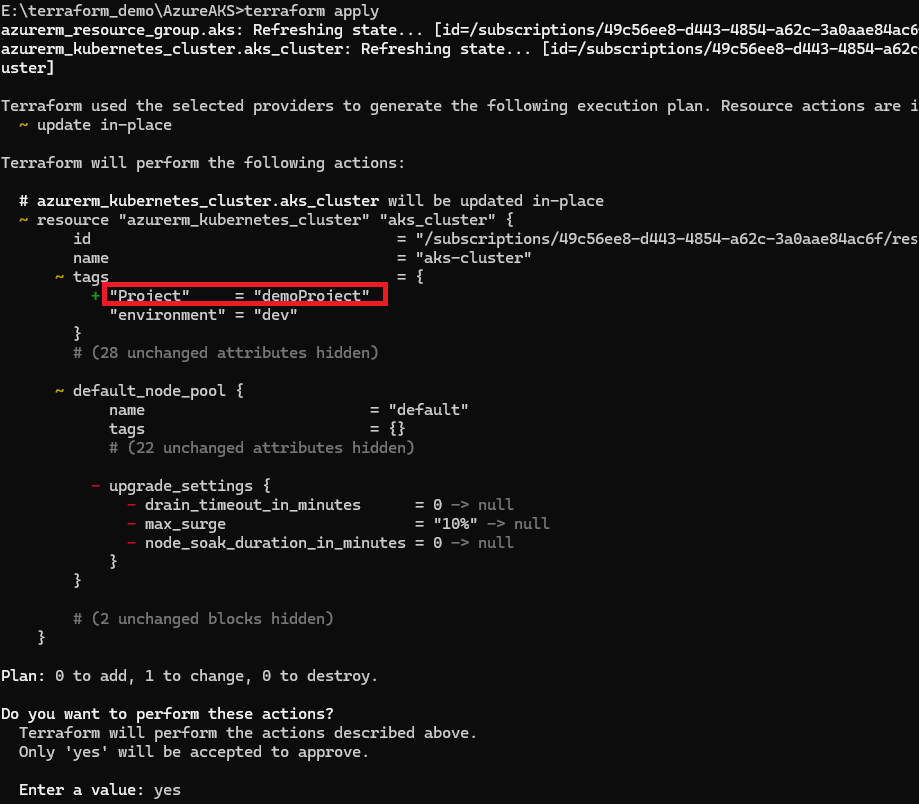
value = azurerm\_kubernetes\_cluster.aks\_cluster.kube\_config\_raw

sensitive = true # Mark the output as sensitive

}

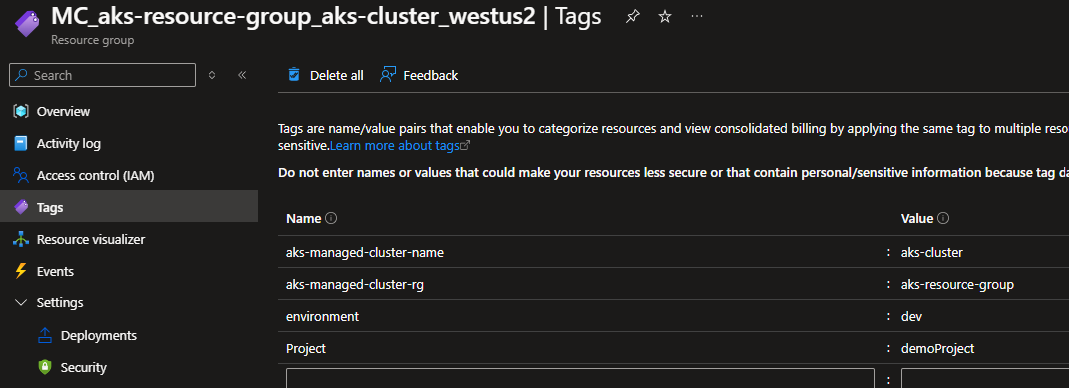
1. Run [terraform apply](https://www.terraform.io/docs/commands/apply.html) to apply the execution plan to your cloud infrastructure.

**terraform Apply**

****

**Note the difference it add new Tag**

1. Go to Azuree Portal and Verify



# Delete AKS Cluster and Resources

1. Run [terraform destroy](https://www.terraform.io/docs/commands/apply.html) to destroy cloud infrastructure created

A screenshot of a computer screen

Description automatically generated