SQLBITS

Deploy Databricks Components using Terraform

@annawykes

https://www.linkedin.com/in/anna-maria-wykes-31939454

AnnaWykes

@falekmiah

falekmiah.com

FalekMiah01

advancinganalytics.co.uk

Anna Wykes

Senior Consultant

- 16+ Years Microsoft Data Analytics
- Intensive Software & Data Engineering Experience
- Microsoft MVP, Data Consultant & Public Speaker

MVP Microsoft® Most Valuable Professional

Falek Miah

Principal Consultant

- 14+ Years Microsoft Data Analytics
- Intensive Data Engineering Experience
- Data, Cloud & DevOps Enthusiast
- Microsoft Azure, Databricks (Spark), Terraform (HashiCorp) certified

Terraform ASSOCIATE HashiCorp CERTIFIED

databricks PARTNER TRAINING DEVELOPER ESSENTIALS

databricks CERTIFIED ASSOCIATE DEVELOPER Apache Spark Release 3.0 APACHE SPARK 3.0

Microsoft CERTIFIED AZURE SOLUTIONS ARCHITECT EXPERT

Microsoft CERTIFIED AZURE DATA ENGINEER ASSOCIATE

Session Scope

Session Scope

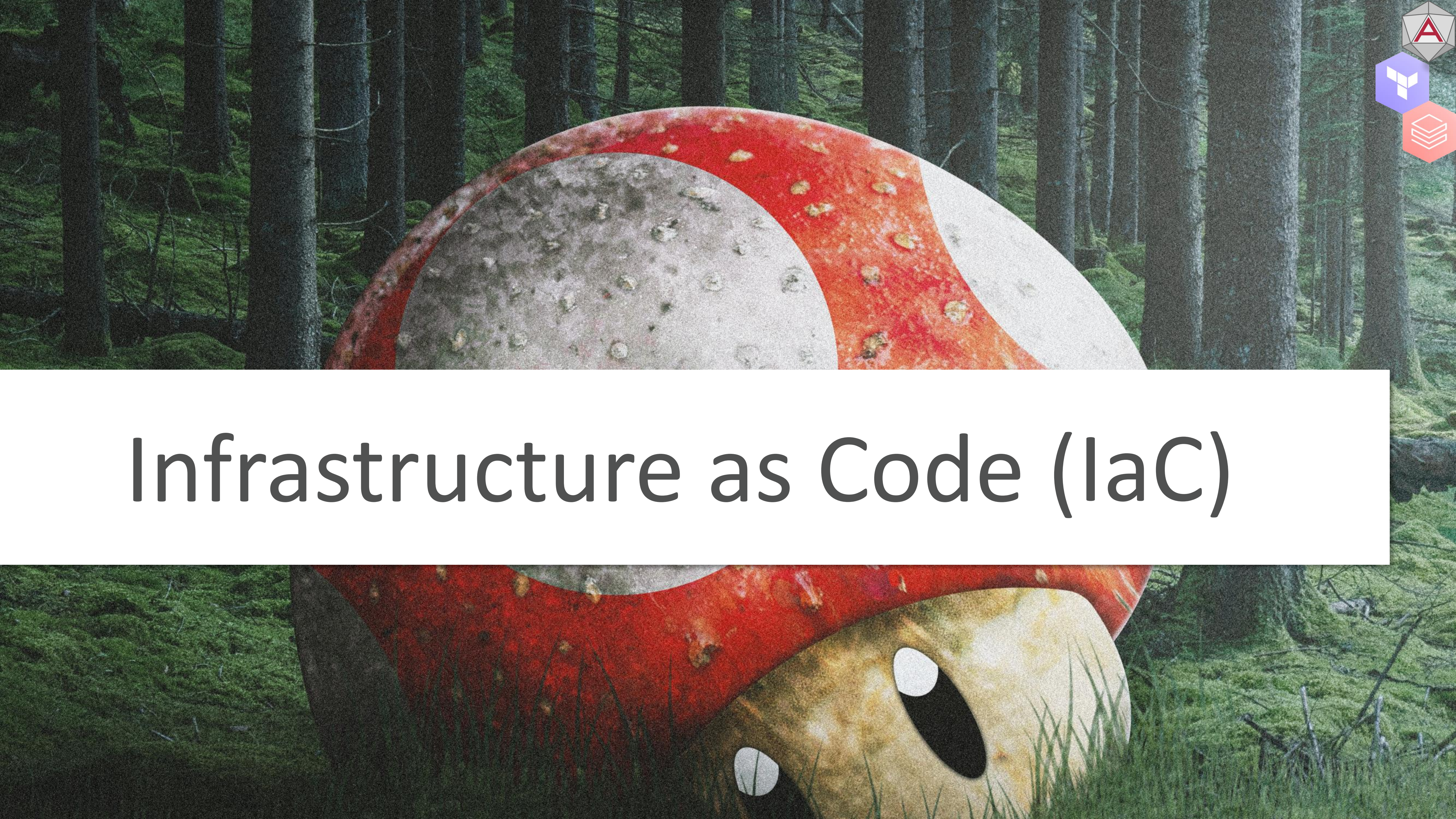
Manually provisioning Databricks workspace and its components (clusters, libraries, etc) in the cloud can be:

- time consuming
- hard to repeat
- becomes inconsistent between environments

Using Infrastructure as Code (IaC) tool like Terraform can minimize this.

Databricks Labs introduce a collection of Terraform Providers that gives you the ability to deploy nearly all resources onto your cloud platform using Terraform.

Reduces the need to use PowerShell, Databricks CLI or REST APIs.



Infrastructure as Code (IaC)

Infrastructure as Code (IaC)

- **Automate** the creation, update or destroy of cloud infrastructure
- Allows you to **version**, **share**, and **reuse** cloud infrastructure
- **Configure script** (blueprint) of cloud infrastructure

Popular Infrastructure as Code (IaC) Tools

ARM (Azure Resource Manager) Templates

HashiCorp Terraform

pulumi

Azure Bicep

AWS CloudFormation

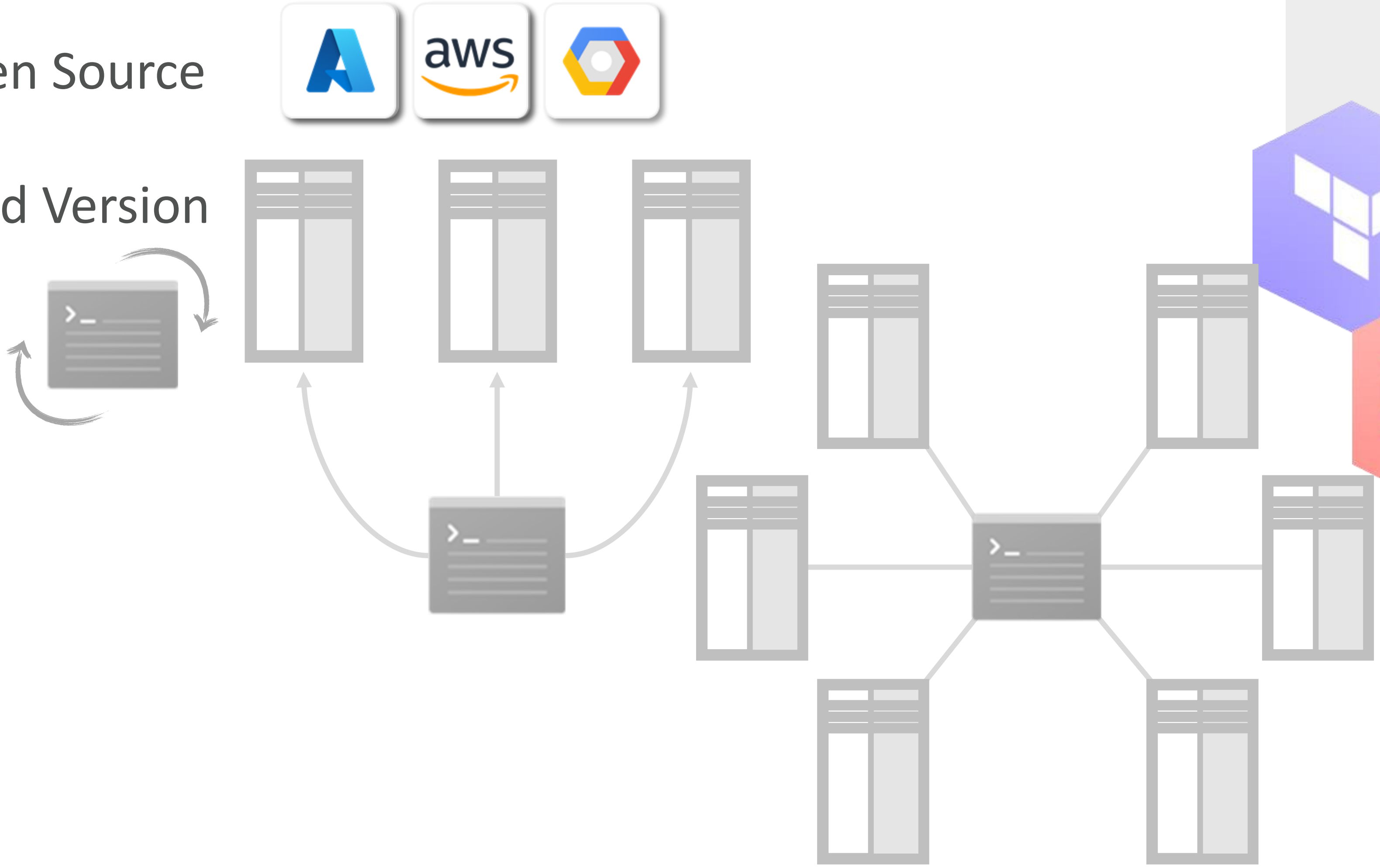
Cloud Deployment Manager

Terraform Overview

Terraform Overview

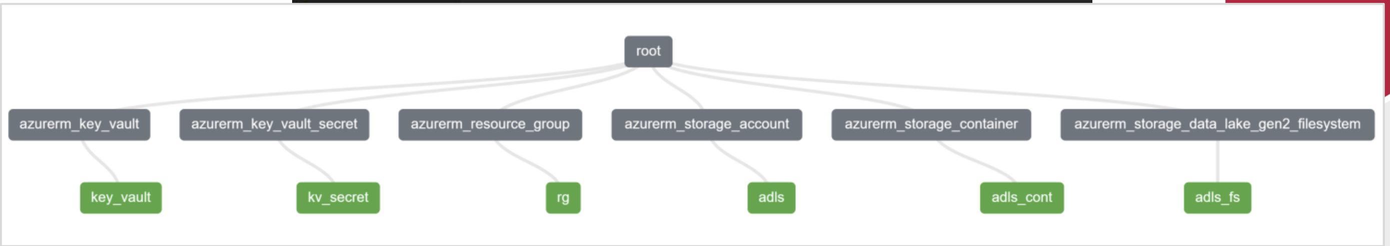
HashiCorp Terraform

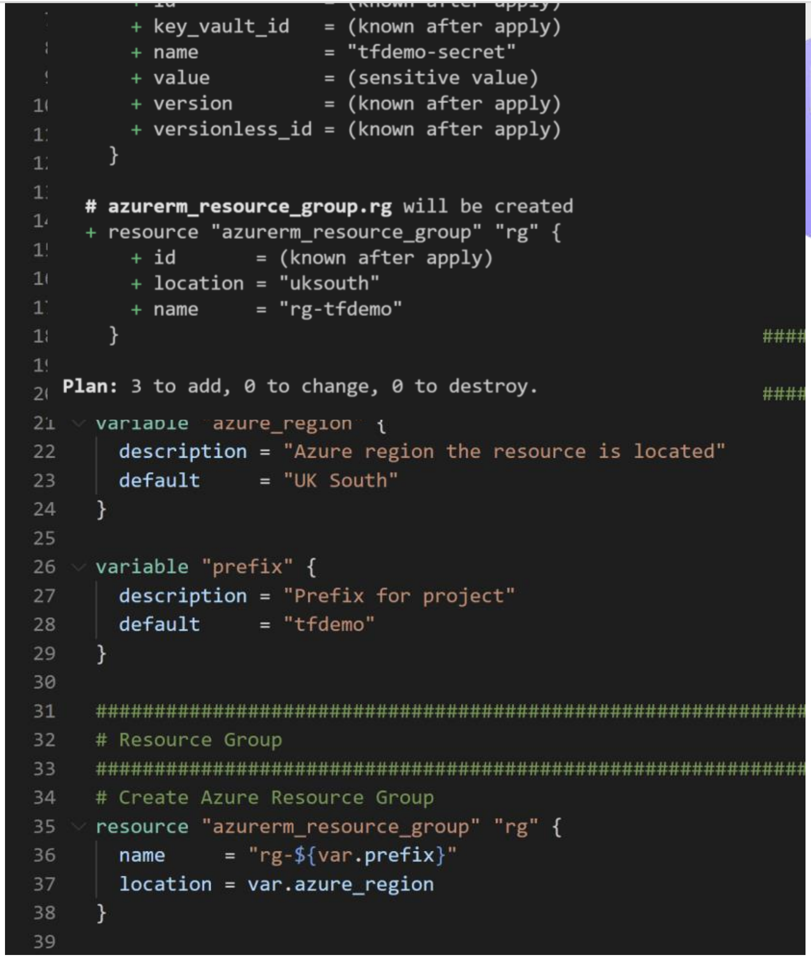
- Infrastructure as Code (IaC)
- Cloud Agnostic and Open Source
- Provisions, Manages and Version
- Consistent
- Reusable



Terraform Features

- Resource Graph
- Execution Plan
- Declarative Configure File
- HCL Terraform Language
  - Supports loops, dynamic blocks and local variables
  - Complex data structures (maps, collection)
- Supports JSON



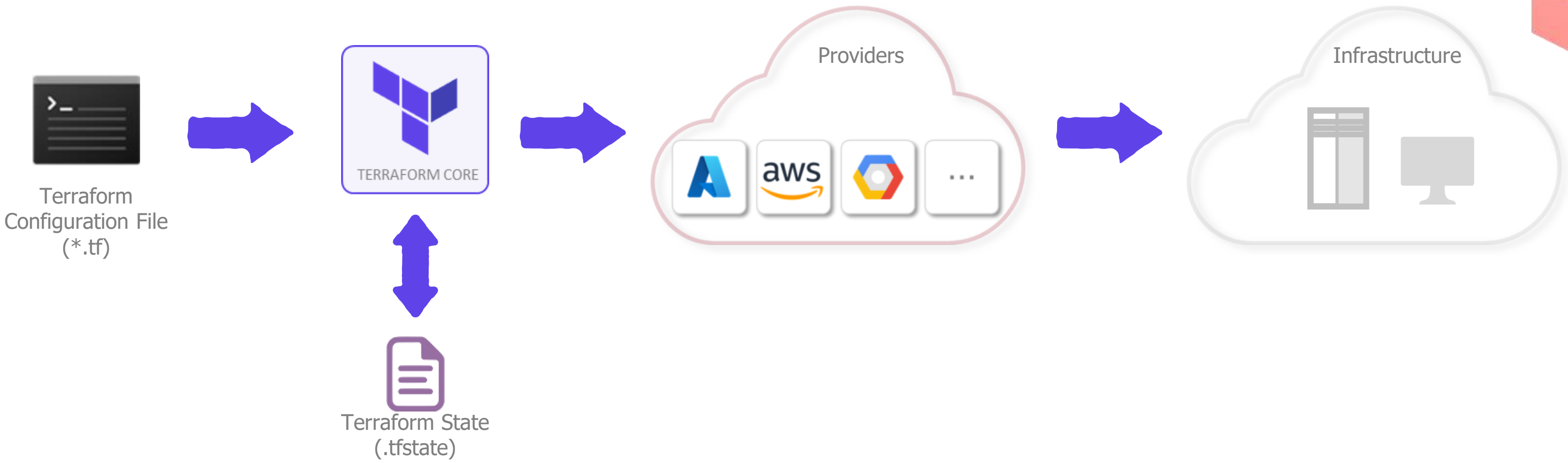


```
      + id              = (known after apply)
      + key_vault_id    = (known after apply)
      + name            = "tfdemo-secret"
      + value           = (sensitive value)
      + version         = (known after apply)
      + versionless_id  = (known after apply)
    }

  # azurerm_resource_group.rg will be created
  + resource "azurerm_resource_group" "rg" {
      + id       = (known after apply)
      + location = "uksouth"
      + name     = "rg-tfdemo"
    }

Plan: 3 to add, 0 to change, 0 to destroy.
```

```
variable "azure_region" {
  description = "Azure region the resource is located"
  default     = "UK South"
}

variable "prefix" {
  description = "Prefix for project"
  default     = "tfdemo"
}

##############################################################
# Resource Group
##############################################################
# Create Azure Resource Group
resource "azurerm_resource_group" "rg" {
  name     = "rg-${var.prefix}"
  location = var.azure_region
}

```

Terraform Architecture

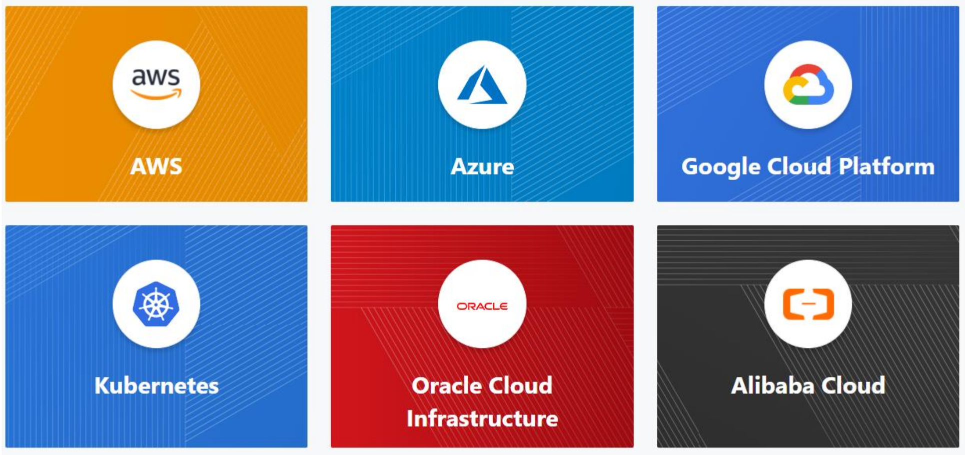
- Configuration File
- Terraform Core Commands
- Providers
- Terraform State
- Infrastructure


Terraform Configuration File (*.tf)
TERRAFORM CORE
Terraform State (.tfstate)
Providers
aws
...
Infrastructure

Terraform Providers

- Providers are required
- Plugins

Official Provider



AWS

Azure

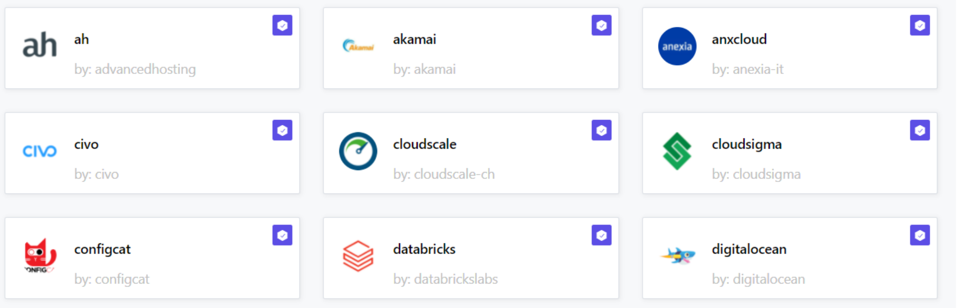
Google Cloud Platform

Kubernetes

Oracle Cloud Infrastructure

Alibaba Cloud

Verified Provider



ah
by: advancedhosting

akamai
by: akamai

anxcloud
by: anexia-it

civo
by: civo

cloudscale
by: cloudscale-ch

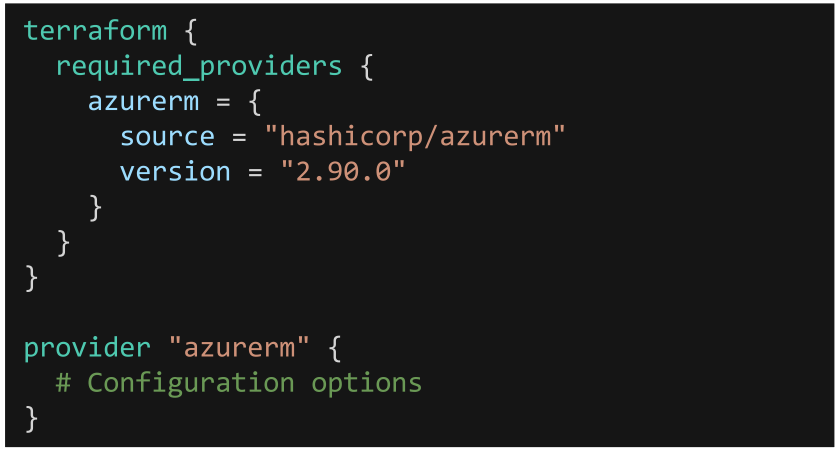
cloudsigma
by: cloudsigma

configcat
by: configcat

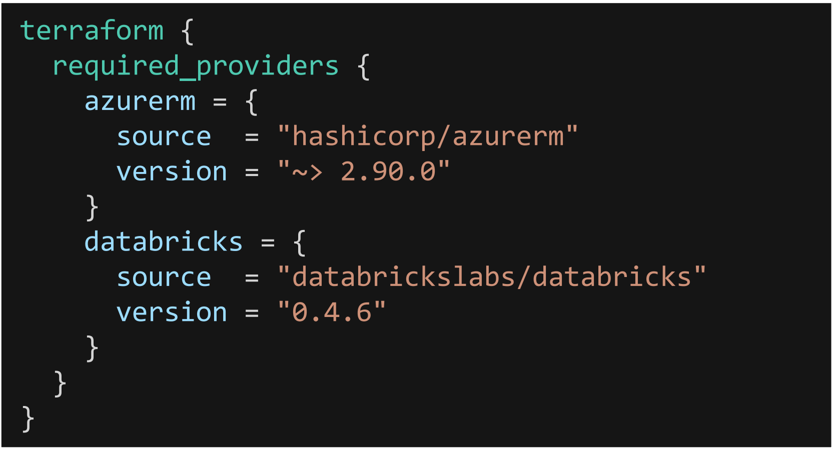
databricks
by: databrickslabs

digitalocean
by: digitalocean

Community Provider



```
terraform {
  required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
      version = "2.90.0"
    }
  }
}

provider "azurerm" {
  # Configuration options
}
```



```
terraform {
  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
      version = "~> 2.90.0"
    }
    databricks = {
      source  = "databrickslabs/databricks"
      version = "0.4.6"
    }
  }
}
```

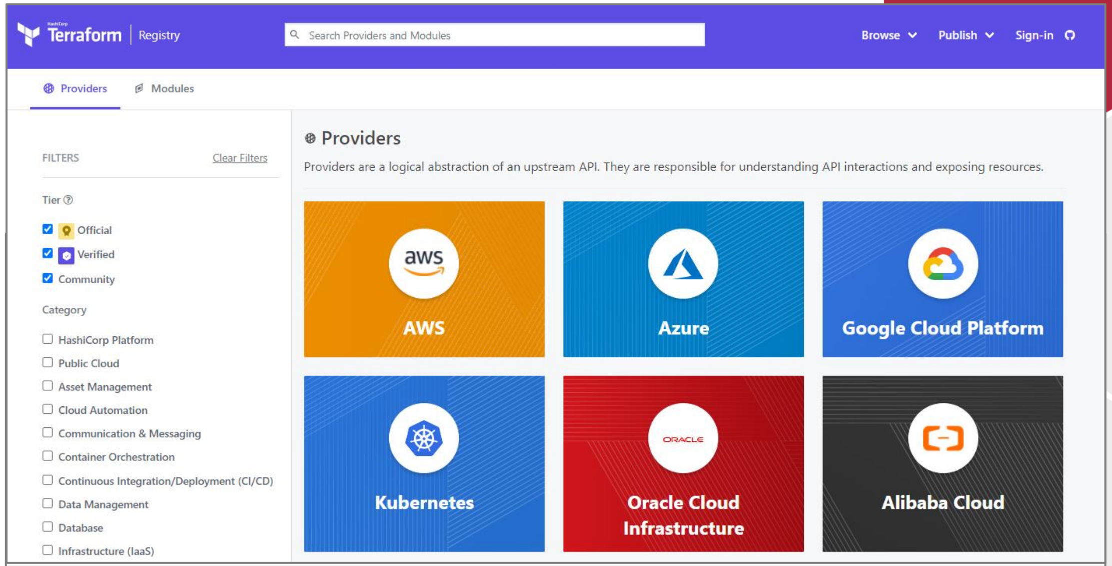
Terraform Registry

- **Terraform Registry** contains providers and modules. https://registry.terraform.io/

- **Example** codes and argument

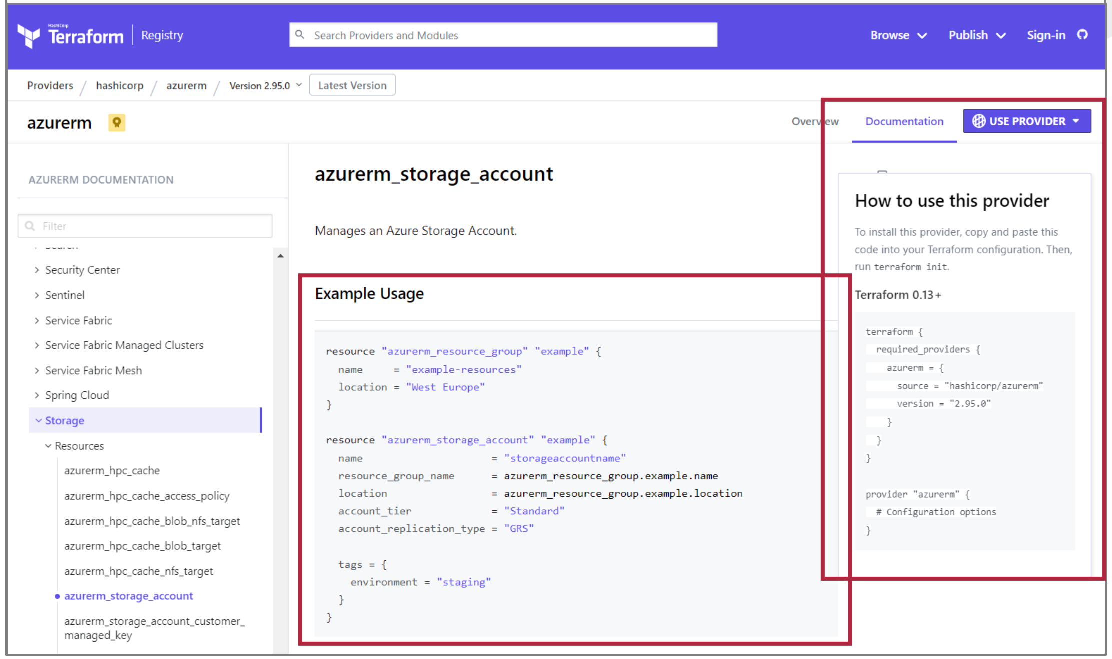
- **Providers** are plugins that are map to a cloud provider’s API

- **Modules** are reusable and self-containing groups of configurations files



Providers

Providers are a logical abstraction of an upstream API. They are responsible for understanding API interactions and exposing resources.


Terraform | Registry
Search Providers and Modules
Browse
Publish
Sign-in
Providers / hashicorp / azurerm / Version 2.95.0
Latest Version
azurerm
Overview
Documentation
USE PROVIDER
AZURERM DOCUMENTATION
Filter
Security Center
Sentinel
Service Fabric
Service Fabric Managed Clusters
Service Fabric Mesh
Spring Cloud
Storage
Resources
azurerm_hpc_cache
azurerm_hpc_cache_access_policy
azurerm_hpc_cache_blob_nfs_target
azurerm_hpc_cache_blob_target
azurerm_hpc_cache_nfs_target
azurerm_storage_account
azurerm_storage_account_customer_managed_key

azurerm_storage_account

Manages an Azure Storage Account.

Example Usage

```
resource "azurerm_resource_group" "example" {
  name     = "example-resources"
  location = "West Europe"
}

resource "azurerm_storage_account" "example" {
  name                     = "storageaccountname"
  resource_group_name      = azurerm_resource_group.example.name
  location                 = azurerm_resource_group.example.location
  account_tier             = "Standard"
  account_replication_type = "GRS"

  tags = {
    environment = "staging"
  }
}
```

How to use this provider

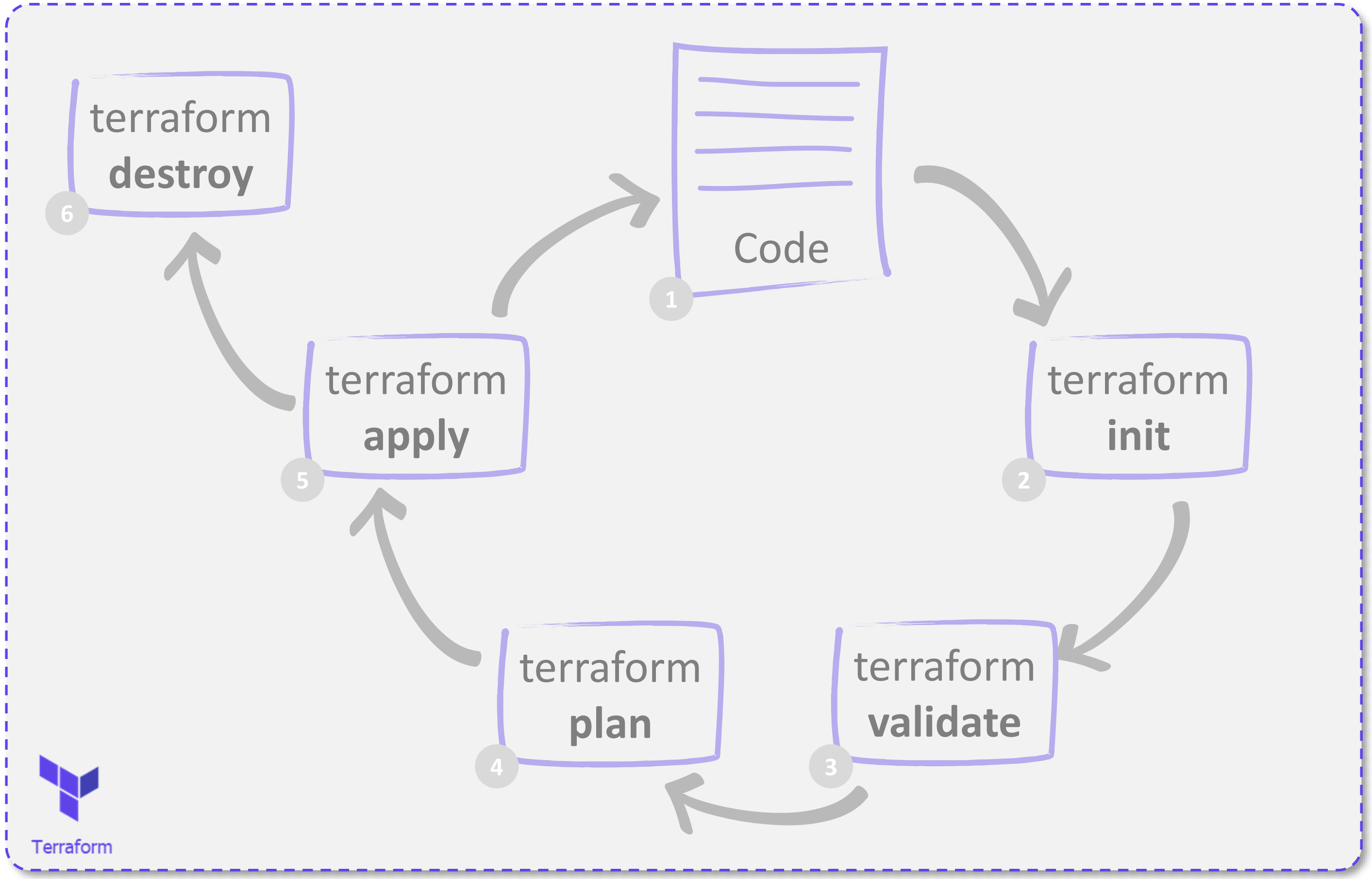
To install this provider, copy and paste this code into your Terraform configuration. Then, run `terraform init`.

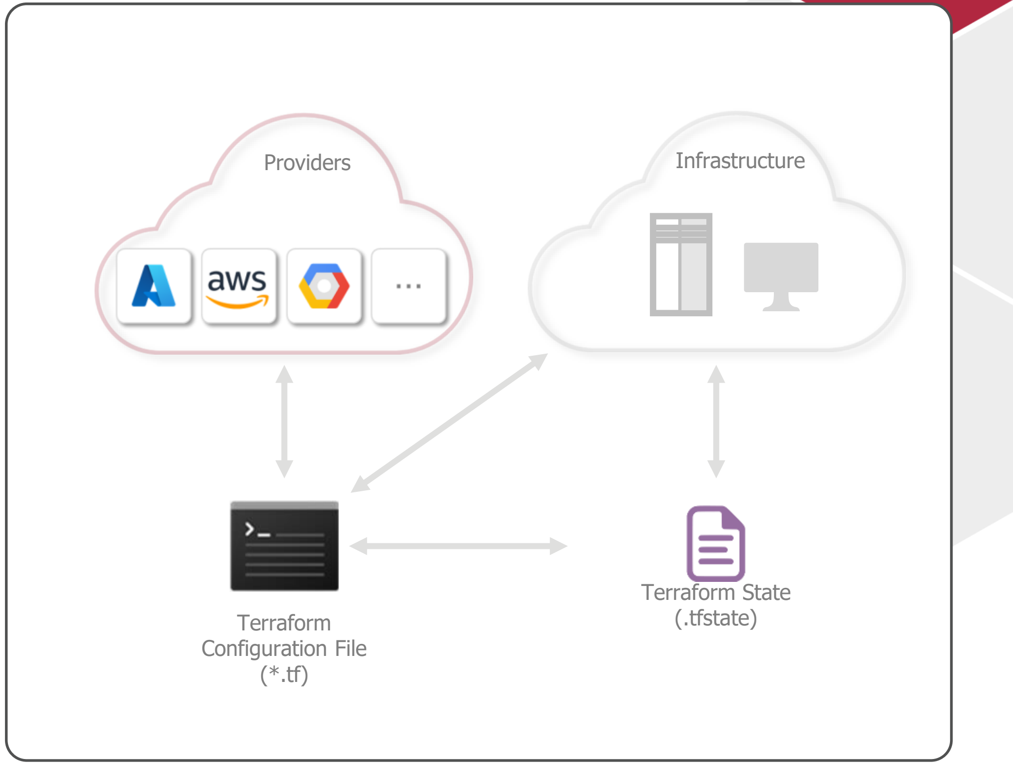
Terraform 0.13+

```
terraform {
  required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
      version = "2.95.0"
    }
  }
}

provider "azurerm" {
  # Configuration options
}
```

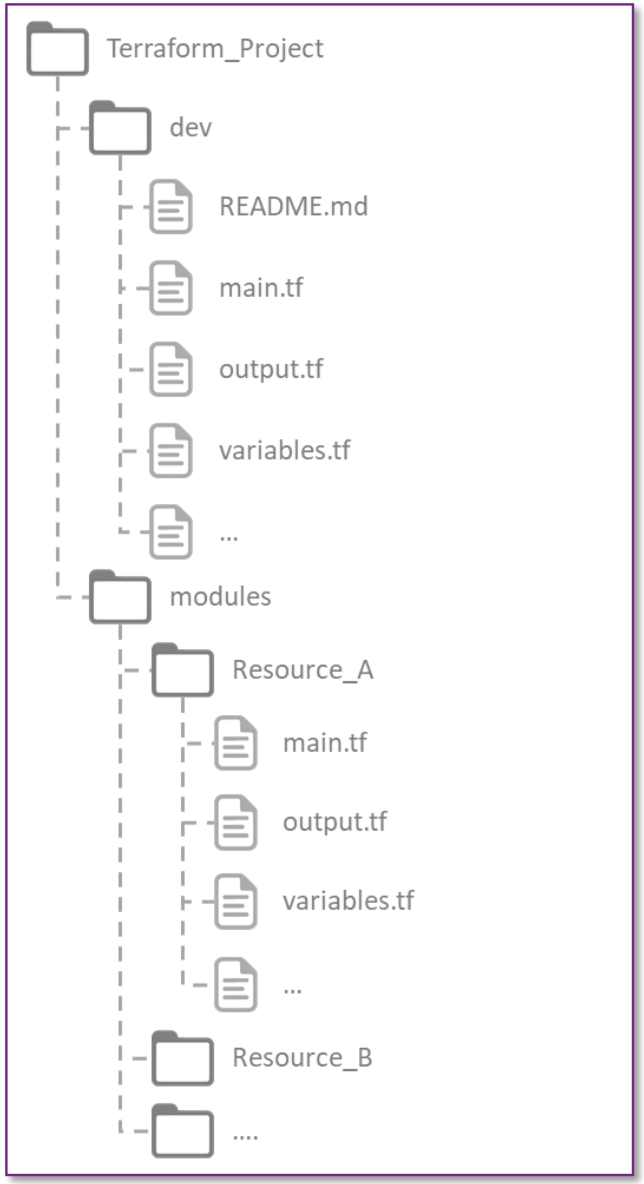
Terraform Concepts

Terraform Lifecycle





Terraform Structure



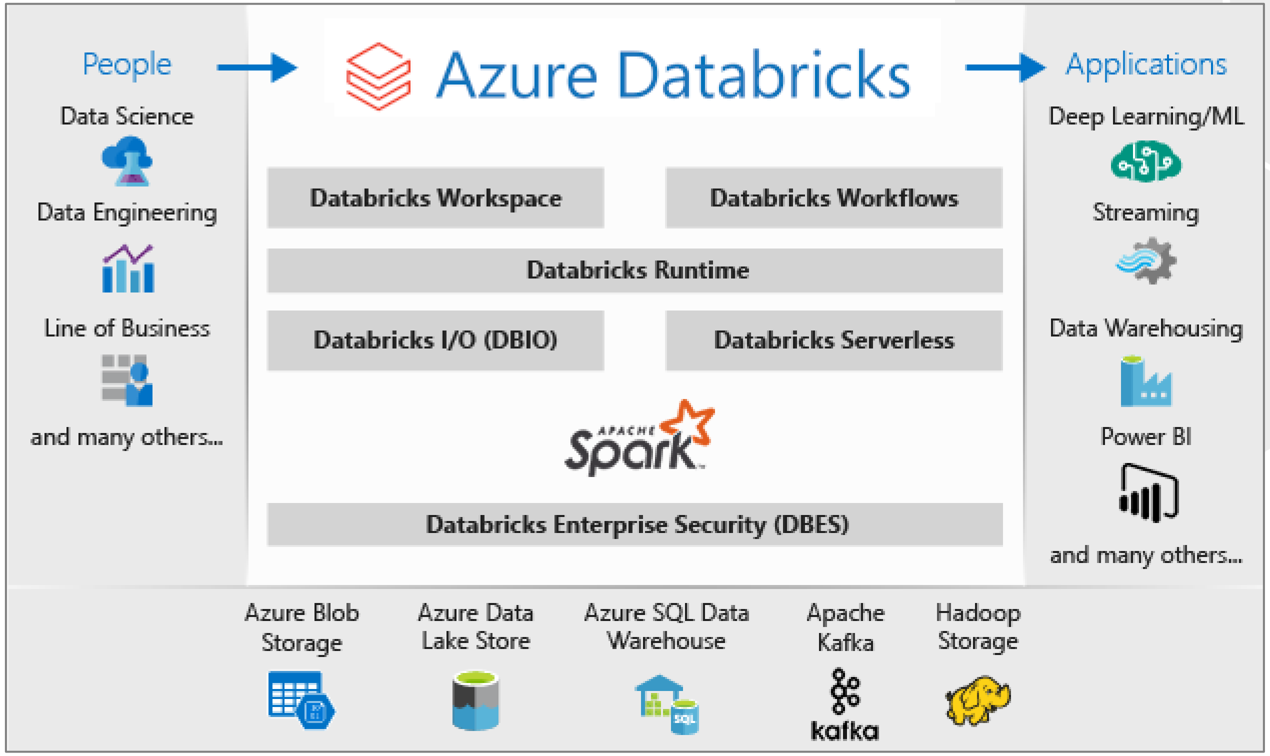
- Terraform_Project
  - dev
    - README.md
    - main.tf
    - output.tf
    - variables.tf
    - …
  - modules
    - Resource_A
      - main.tf
      - output.tf
      - variables.tf
      - …
    - Resource_B
    - ….

README.md	-	Description of what resources, modules and project
main.tf	-	Resource configuration for the infrastructure
output.tf	-	Output values of the resources created
variables.tf	-	Input variables for resources
*.tfvars	-	Variable definitions for the resource to apply
locals.tf	-	Define expression and use multiple times within a modules
data.tf	-	Return read-only view of pre-existing resource and components

Databricks

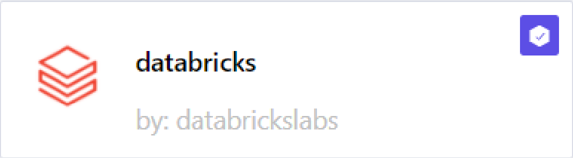
Databricks

- **Data analytics** platform for cloud services
- **Ingest**, **transform** and **explore** large amounts of data
- Based on **Apache-Spark** distributed system
  - **Multiple Processing**
  - **Scalable**



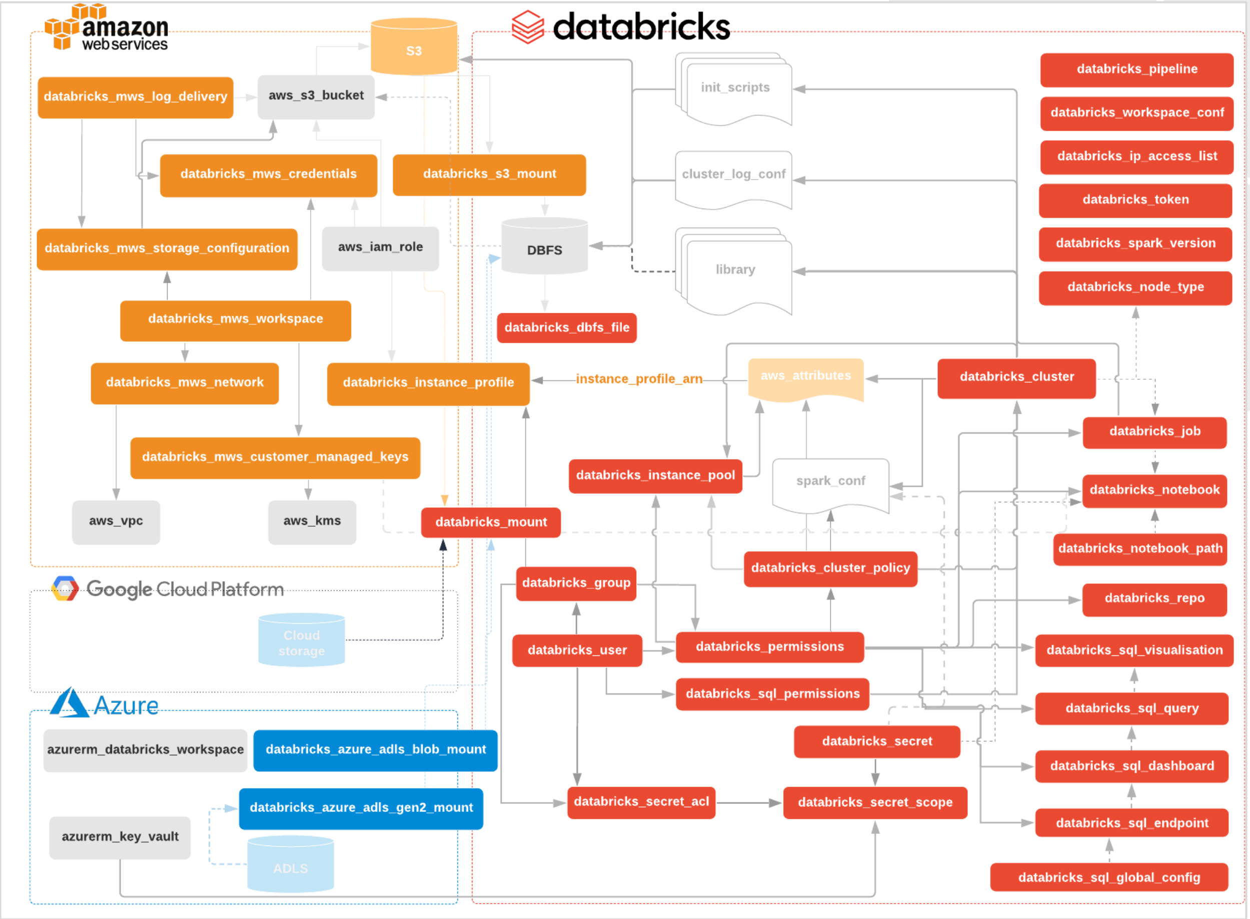
Databricks Labs

- Collection of Terraform providers
- Supports all Databricks REST APIs
- Create, update, and delete components
- Requires both a Cloud Service provider and the Databricks Terraform provider



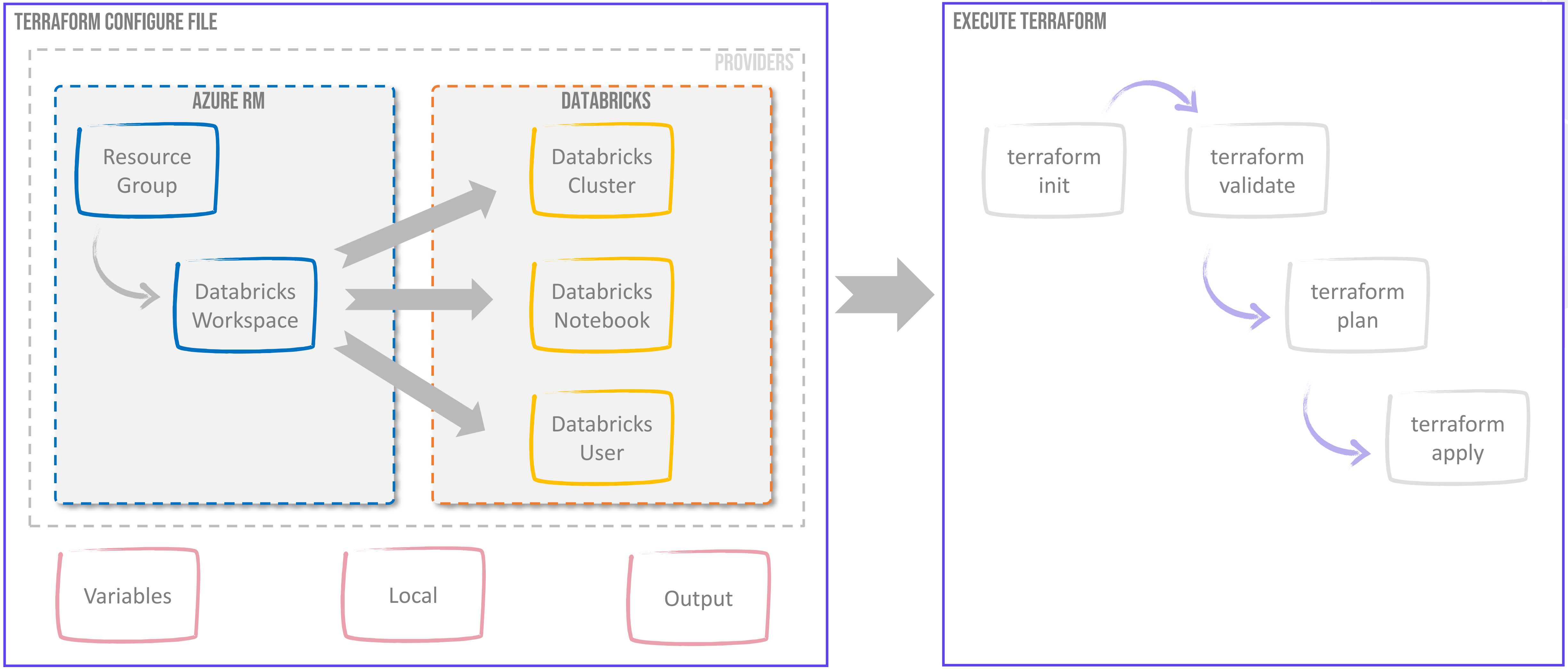
databricks

by: databrickslabs



DEMO

Deploy Databricks Components using Terraform



Demo

- Using Terraform for Databricks IaC

CI/CD Deployment Tools

CI/CD Deployment Tools

- Deploy Terraform using:
  - DevOps CI/CD
  - GitHub Actions
  - Jenkins
  - AWS CodeBuild
  - GitLab
  - CircleCI
  - etc…

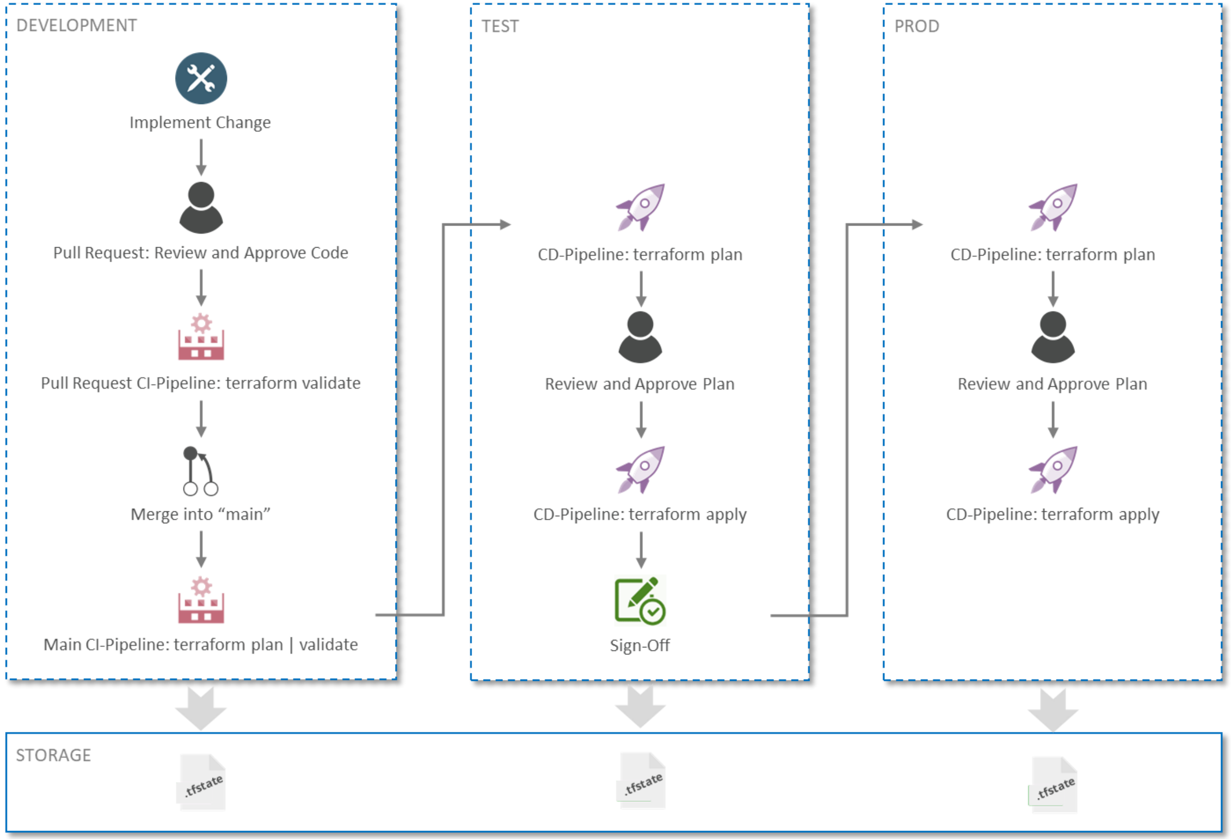
Publish Terraform

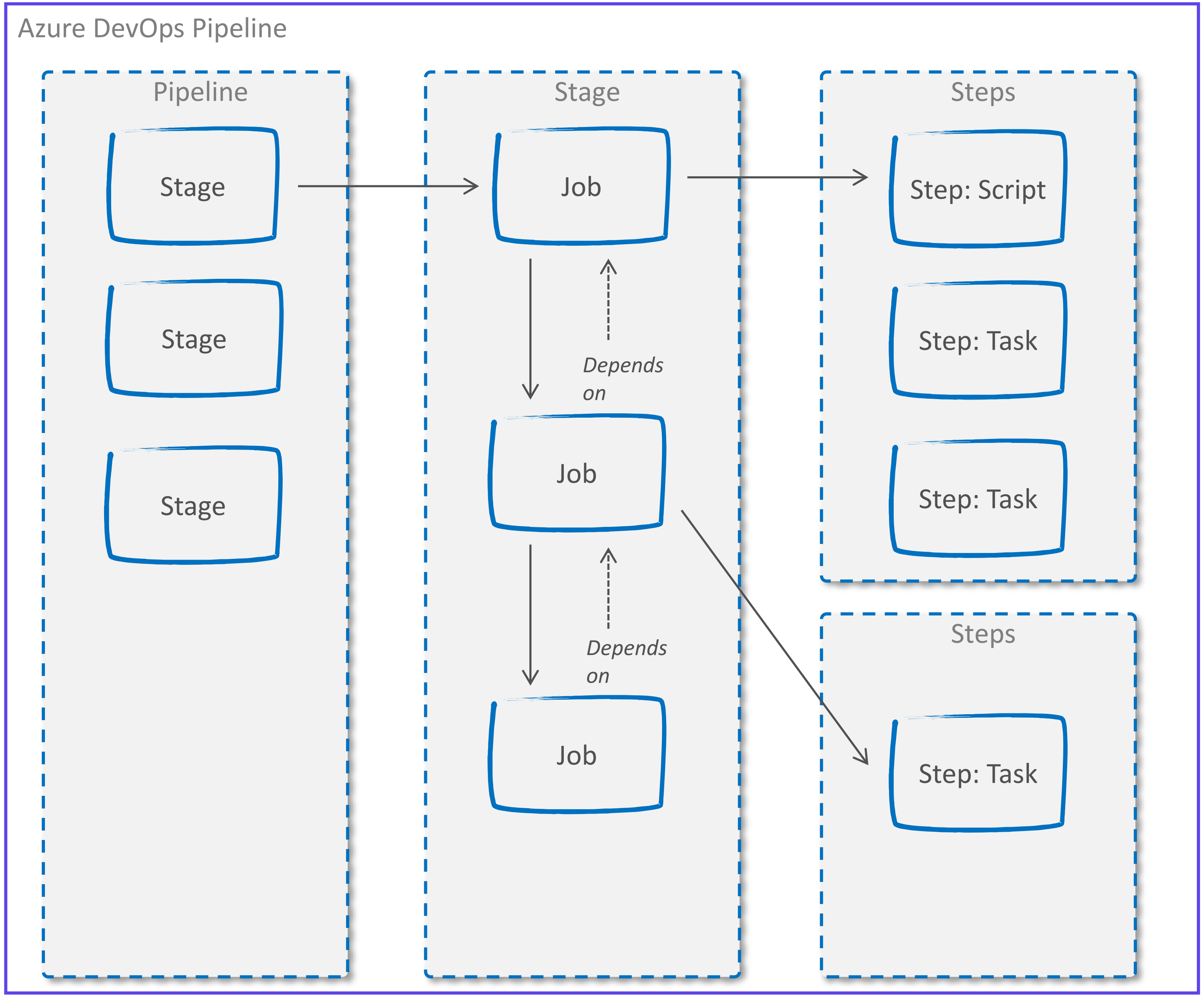
Introducing changes into a cloud environment without manual intervention and having explicit permissions

- Develop a deployment pipeline using CI/CD tool for release management
- Adopt GitOps approach and use repositories, branches, and enforce pull requests

Example Delivery Workflow:

- *Change is reviewed and merged with a Pull Request*
- *Pull Request validated using CI-Pipeline*
- *Branch merged into "main" and validated using CI-Pipeline*
- *Deploy to Test environment using CD-Pipeline, approve plan and apply*
- *Unit testing and sign-off environment*
- *Deploy to Prod environment using CD-Pipeline, approve plan and apply*


DEVELOPMENT
Implement Change
Pull Request: Review and Approve Code
Pull Request CI-Pipeline: terraform validate
Merge into "main"
Main CI-Pipeline: terraform plan | validate
TEST
CD-Pipeline: terraform plan
Review and Approve Plan
CD-Pipeline: terraform apply
Sign-Off
PROD
CD-Pipeline: terraform plan
Review and Approve Plan
CD-Pipeline: terraform apply
STORAGE
.tfstate
.tfstate
.tfstate

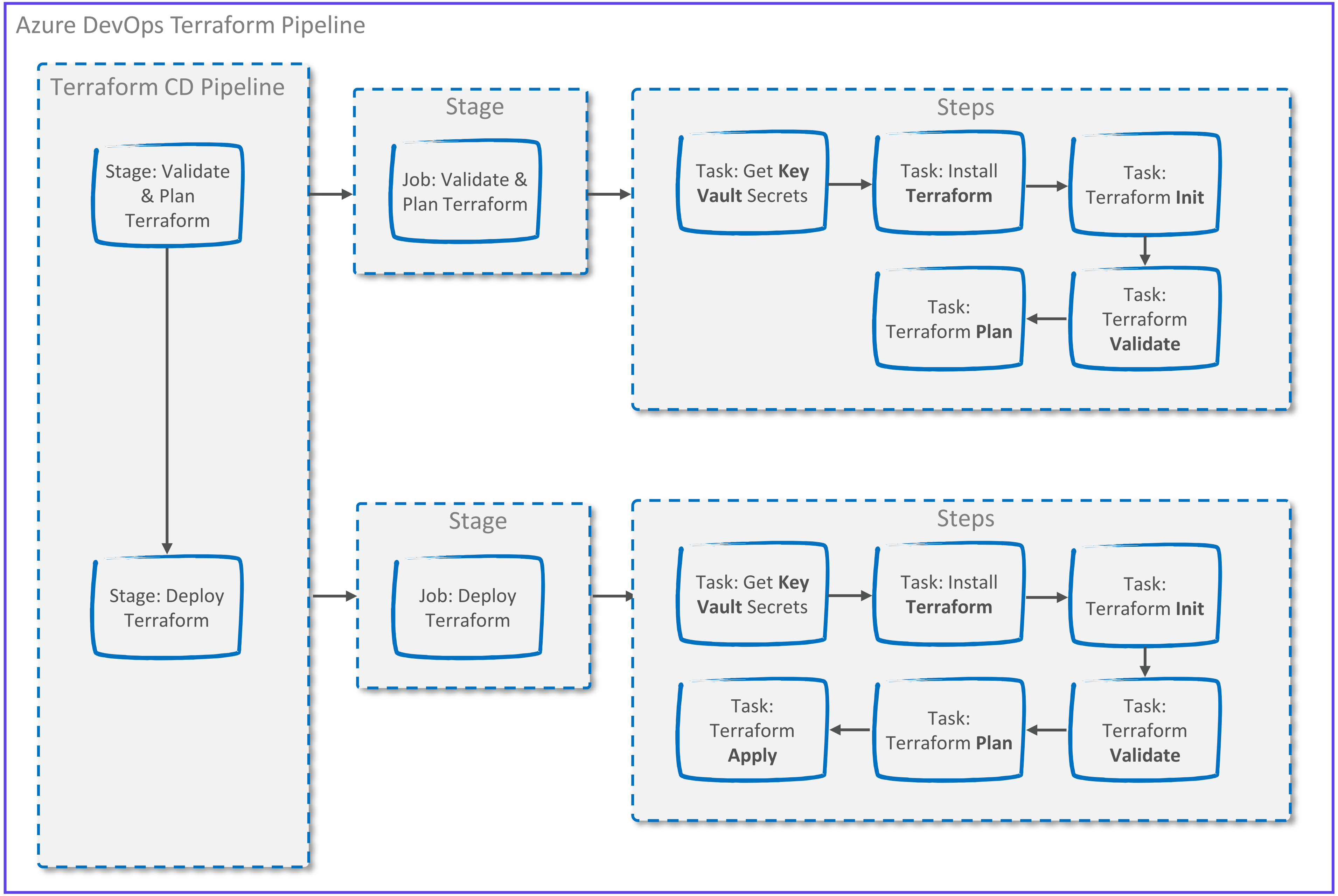


YML Pipelines consist of:

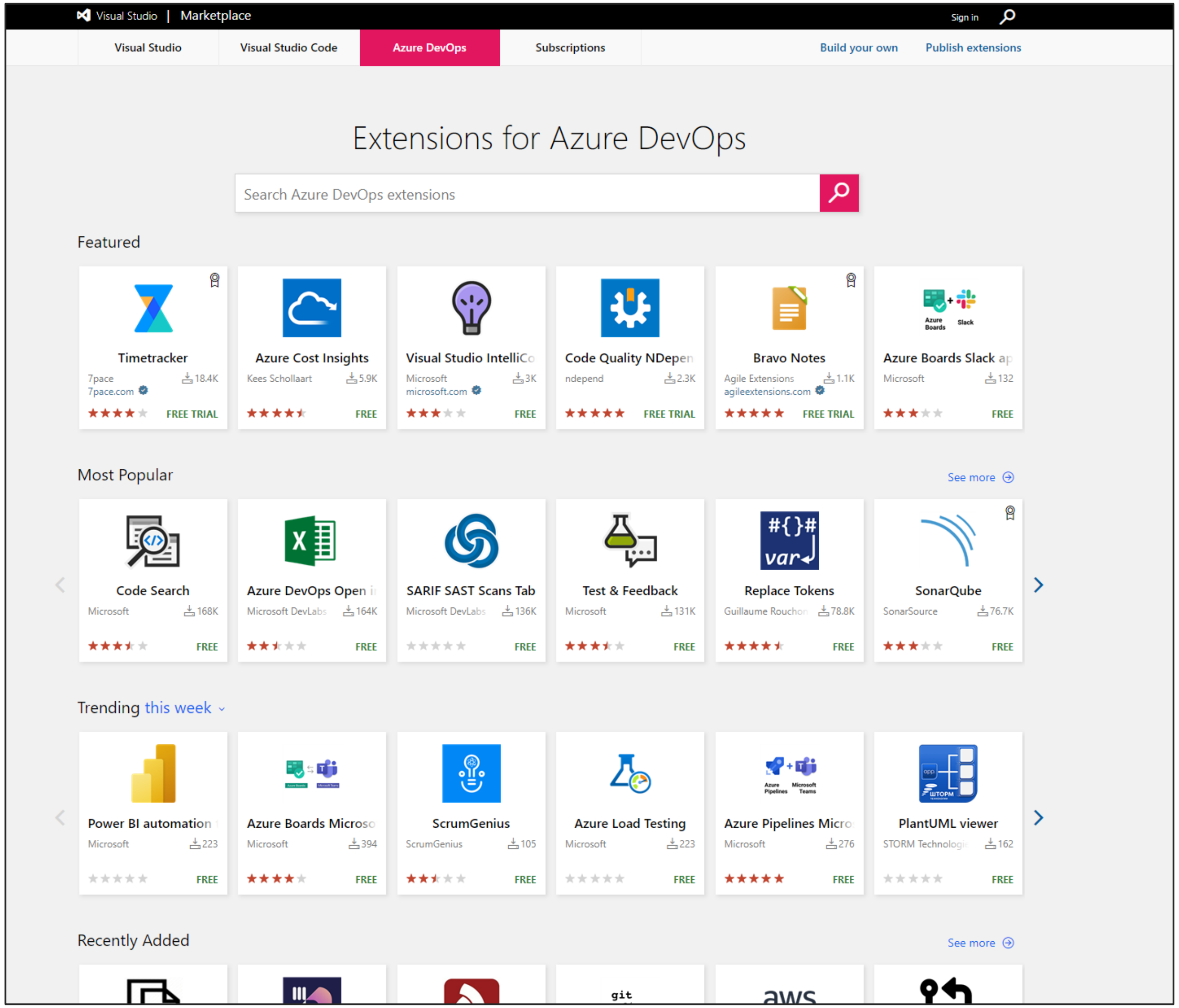
- Stages
- Jobs
- Steps

Tasks/Scripts may include:

- Install components
- Calls to Rest APIs
- CLI Commands
- Running code (e.g. Python script)
- Connecting to Key Vault
- Publishing Build Artifacts


Azure DevOps Terraform Pipeline
Terraform CD Pipeline
Stage: Validate & Plan Terraform
Stage: Deploy Terraform
Stage
Job: Validate & Plan Terraform
Stage
Job: Deploy Terraform
Steps
Task: Get Key Vault Secrets
Task: Install Terraform
Task: Terraform Init
Task: Terraform Validate
Task: Terraform Plan
Steps
Task: Get Key Vault Secrets
Task: Install Terraform
Task: Terraform Init
Task: Terraform Validate
Task: Terraform Plan
Task: Terraform Apply

Azure DevOps Extensions



Visual Studio | Marketplace

Sign in

Visual Studio | Visual Studio Code | Azure DevOps | Subscriptions

Build your own | Publish extensions

Extensions for Azure DevOps

Search Azure DevOps extensions

Featured

- Timetracker, 7pace, 7pace.com, 18.4K, ★★★★☆, FREE TRIAL
- Azure Cost Insights, Kees Schollaart, 5.9K, ★★★★½, FREE
- Visual Studio IntelliCo, Microsoft, microsoft.com, 3K, ★★★☆☆, FREE
- Code Quality NDepen, ndepend, 2.3K, ★★★★★, FREE TRIAL
- Bravo Notes, Agile Extensions, agileextensions.com, 1.1K, ★★★★★, FREE TRIAL
- Azure Boards Slack ap, Microsoft, 132, ★★★☆☆, FREE

Most Popular

See more

- Code Search, Microsoft, 168K, ★★★½☆, FREE
- Azure DevOps Open i, Microsoft DevLabs, 164K, ★★½☆☆, FREE
- SARIF SAST Scans Tab, Microsoft DevLabs, 136K, ☆☆☆☆☆, FREE
- Test & Feedback, Microsoft, 131K, ★★★½☆, FREE
- Replace Tokens, Guillaume Rouchon, 78.8K, ★★★★½, FREE
- SonarQube, SonarSource, 76.7K, ★★★☆☆, FREE

Trending this week

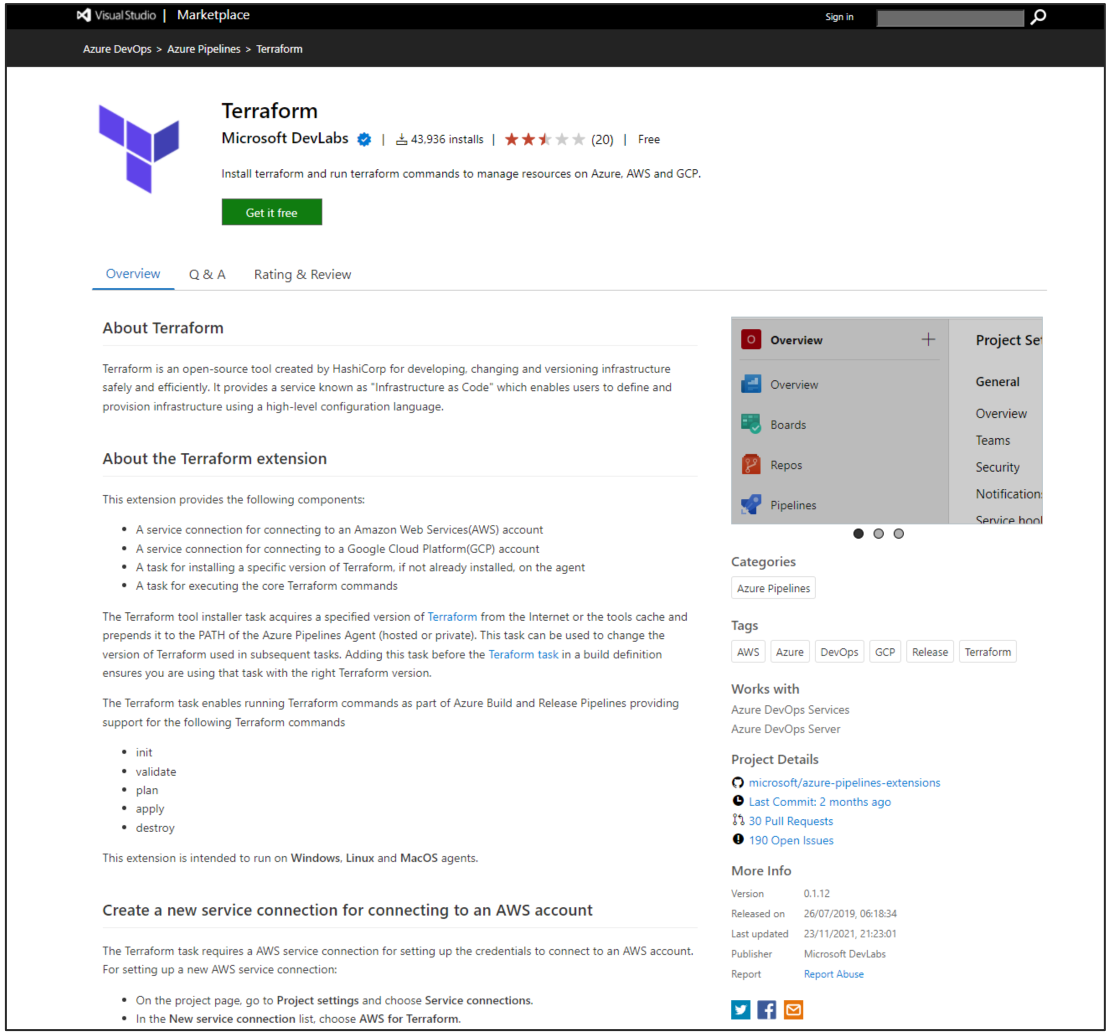
- Power BI automation, Microsoft, 223, ☆☆☆☆☆, FREE
- Azure Boards Microso, Microsoft, 394, ★★★★☆, FREE
- ScrumGenius, ScrumGenius, 105, ★★½☆☆, FREE
- Azure Load Testing, Microsoft, 223, ☆☆☆☆☆, FREE
- Azure Pipelines Micro, Microsoft, 276, ★★★★★, FREE
- PlantUML viewer, STORM Technologi, 162, ☆☆☆☆☆, FREE

Recently Added

See more

Azure DevOps Terraform Extension

Recommended:



Terraform

Microsoft DevLabs | 43,936 installs | ★★⯪☆☆ (20) | Free

Install terraform and run terraform commands to manage resources on Azure, AWS and GCP.

Get it free

Overview Q & A Rating & Review

About Terraform

Terraform is an open-source tool created by HashiCorp for developing, changing and versioning infrastructure safely and efficiently. It provides a service known as "Infrastructure as Code" which enables users to define and provision infrastructure using a high-level configuration language.

About the Terraform extension

This extension provides the following components:

- A service connection for connecting to an Amazon Web Services(AWS) account
- A service connection for connecting to a Google Cloud Platform(GCP) account
- A task for installing a specific version of Terraform, if not already installed, on the agent
- A task for executing the core Terraform commands

The Terraform tool installer task acquires a specified version of Terraform from the Internet or the tools cache and prepends it to the PATH of the Azure Pipelines Agent (hosted or private). This task can be used to change the version of Terraform used in subsequent tasks. Adding this task before the Teraform task in a build definition ensures you are using that task with the right Terraform version.

The Terraform task enables running Terraform commands as part of Azure Build and Release Pipelines providing support for the following Terraform commands

- init
- validate
- plan
- apply
- destroy

This extension is intended to run on **Windows**, **Linux** and **MacOS** agents.

Create a new service connection for connecting to an AWS account

The Terraform task requires a AWS service connection for setting up the credentials to connect to an AWS account. For setting up a new AWS service connection:

- On the project page, go to **Project settings** and choose **Service connections**.
- In the **New service connection** list, choose **AWS for Terraform**.

Categories

Azure Pipelines

Tags

AWS Azure DevOps GCP Release Terraform

Works with

Azure DevOps Services
Azure DevOps Server

Project Details

microsoft/azure-pipelines-extensions
Last Commit: 2 months ago
30 Pull Requests
190 Open Issues

More Info

Version	0.1.12
Released on	26/07/2019, 06:18:34
Last updated	23/11/2021, 21:23:01
Publisher	Microsoft DevLabs
Report	Report Abuse

Getting started

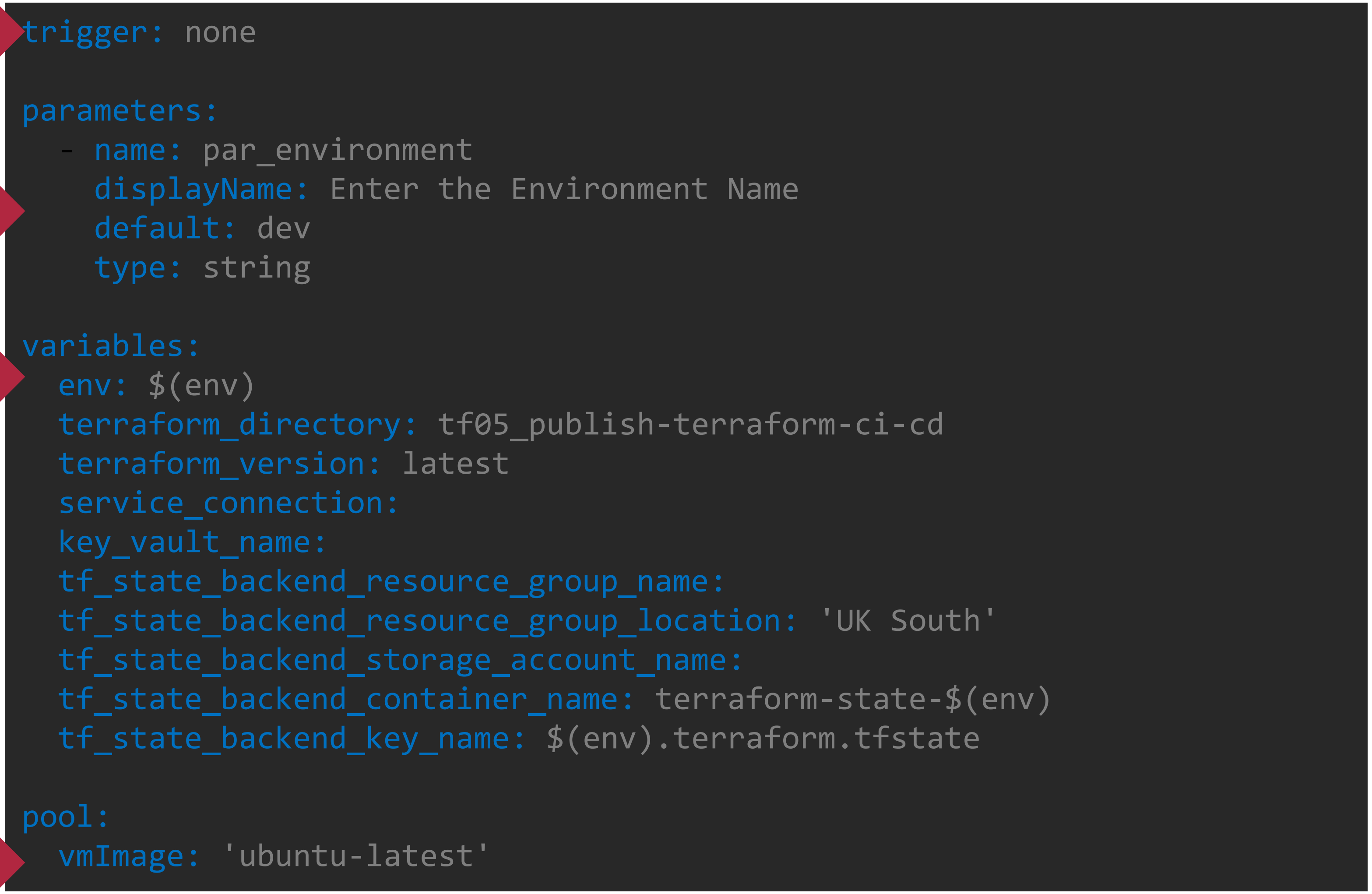
pipeline.yml

No trigger

Parameters entered on pipeline execution

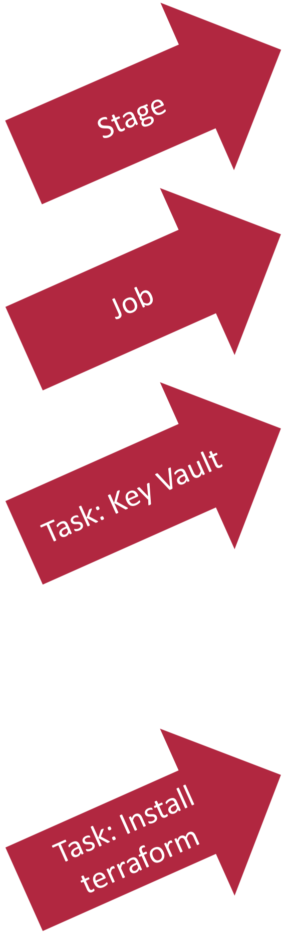
Variables: static, dynamic and key Vault

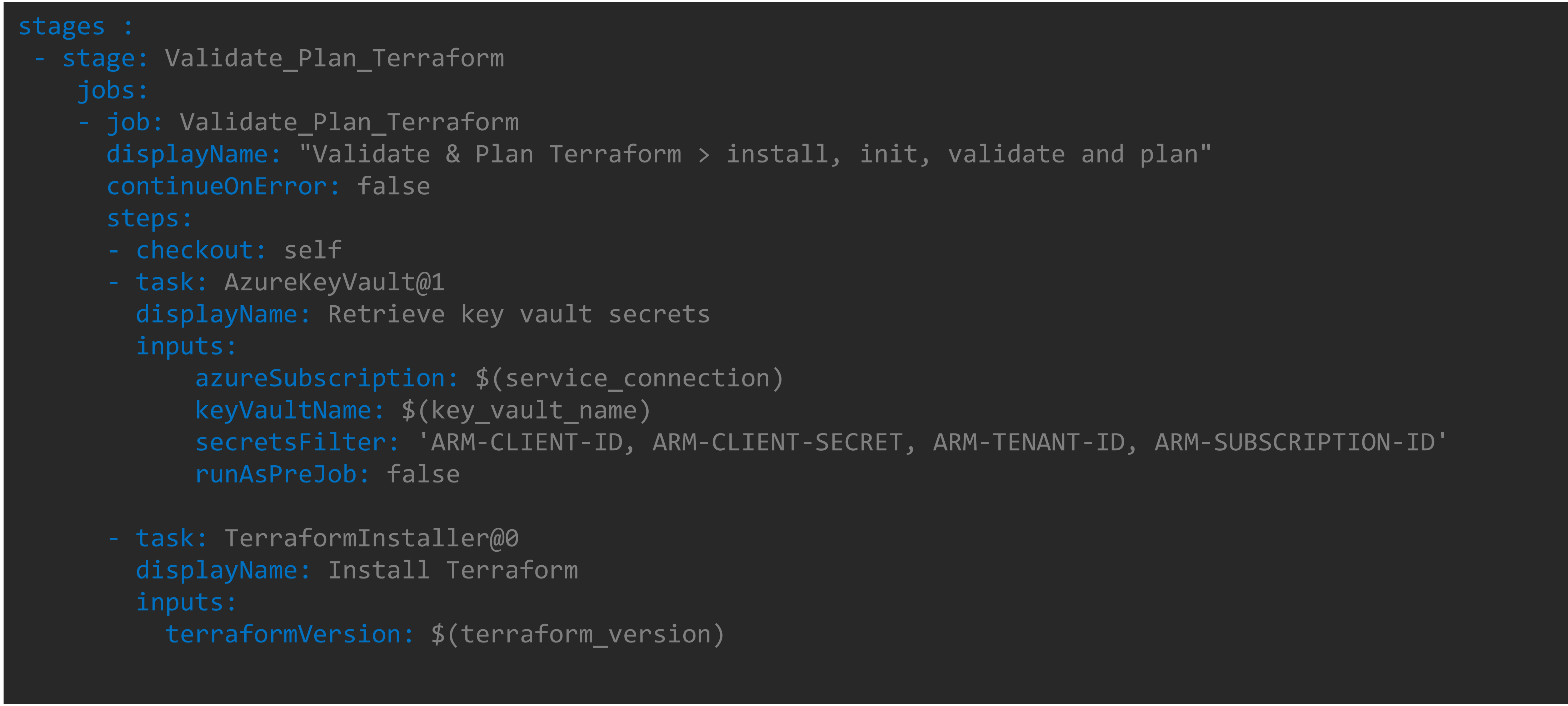
Microsoft hosted image for pool



```yaml
trigger: none

parameters:
  - name: par_environment
    displayName: Enter the Environment Name
    default: dev
    type: string

variables:
  env: $(env)
  terraform_directory: tf05_publish-terraform-ci-cd
  terraform_version: latest
  service_connection:
  key_vault_name:
  tf_state_backend_resource_group_name:
  tf_state_backend_resource_group_location: 'UK South'
  tf_state_backend_storage_account_name:
  tf_state_backend_container_name: terraform-state-$(env)
  tf_state_backend_key_name: $(env).terraform.tfstate

pool:
  vmImage: 'ubuntu-latest'
```

Stage



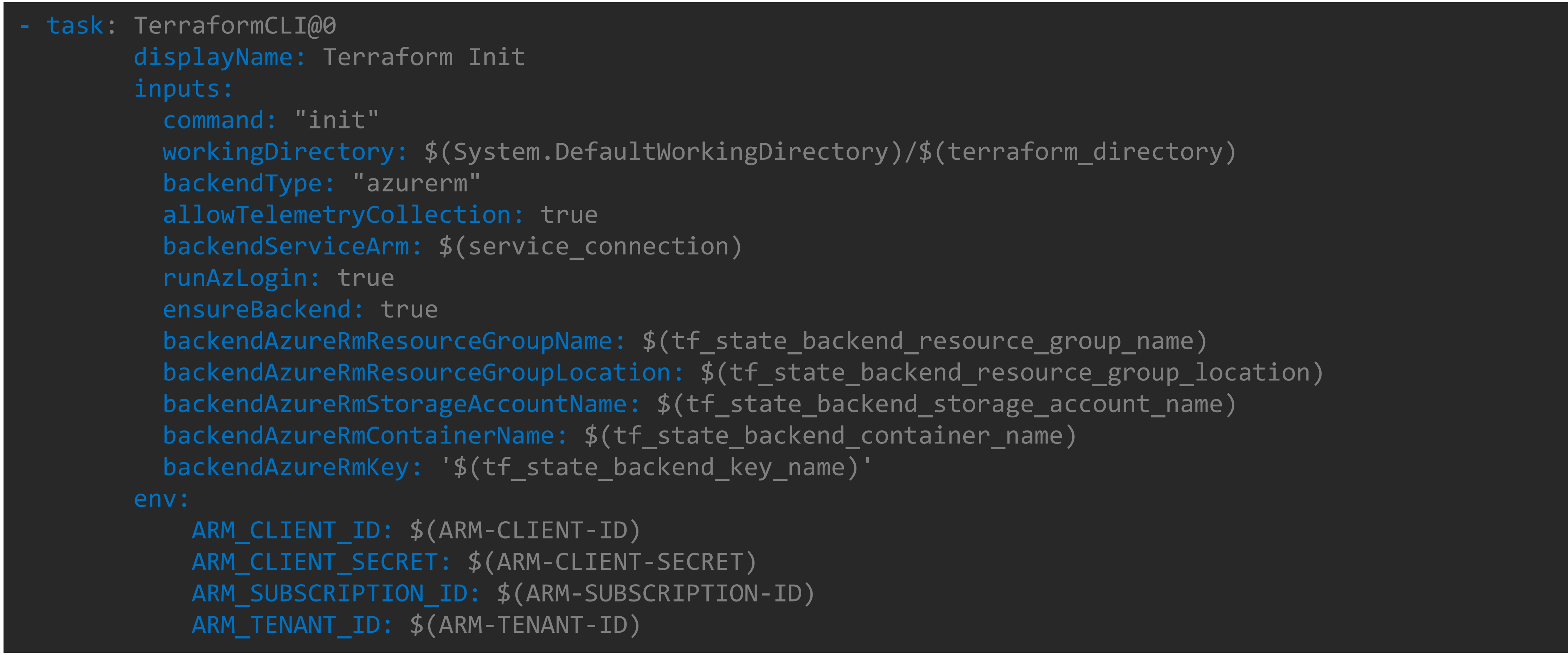


```
stages :
 - stage: Validate_Plan_Terraform
    jobs:
    - job: Validate_Plan_Terraform
      displayName: "Validate & Plan Terraform > install, init, validate and plan"
      continueOnError: false
      steps:
      - checkout: self
      - task: AzureKeyVault@1
        displayName: Retrieve key vault secrets
        inputs:
            azureSubscription: $(service_connection)
            keyVaultName: $(key_vault_name)
            secretsFilter: 'ARM-CLIENT-ID, ARM-CLIENT-SECRET, ARM-TENANT-ID, ARM-SUBSCRIPTION-ID'
            runAsPreJob: false

      - task: TerraformInstaller@0
        displayName: Install Terraform
        inputs:
          terraformVersion: $(terraform_version)
```

Init

Terraform task

State Configuration



```yaml
- task: TerraformCLI@0
        displayName: Terraform Init
        inputs:
          command: "init"
          workingDirectory: $(System.DefaultWorkingDirectory)/$(terraform_directory)
          backendType: "azurerm"
          allowTelemetryCollection: true
          backendServiceArm: $(service_connection)
          runAzLogin: true
          ensureBackend: true
          backendAzureRmResourceGroupName: $(tf_state_backend_resource_group_name)
          backendAzureRmResourceGroupLocation: $(tf_state_backend_resource_group_location)
          backendAzureRmStorageAccountName: $(tf_state_backend_storage_account_name)
          backendAzureRmContainerName: $(tf_state_backend_container_name)
          backendAzureRmKey: '$(tf_state_backend_key_name)'
        env:
            ARM_CLIENT_ID: $(ARM-CLIENT-ID)
            ARM_CLIENT_SECRET: $(ARM-CLIENT-SECRET)
            ARM_SUBSCRIPTION_ID: $(ARM-SUBSCRIPTION-ID)
            ARM_TENANT_ID: $(ARM-TENANT-ID)
```

Apply

Changed the command

```
- task: TerraformCLI@0
    displayName: Terraform Apply
    inputs:
     command: “apply”
     workingDirectory: $(System.DefaultWorkingDirectory)/$(terraform_directory)
     backendType: "azurerm"
     allowTelemetryCollection: true
     backendServiceArm: $(service_connection)
     runAzLogin: true
     ensureBackend: true
     backendAzureRmResourceGroupName: $(tf_state_backend_resource_group_name)
     backendAzureRmResourceGroupLocation: $(tf_state_backend_resource_group_location)
     backendAzureRmStorageAccountName: $(tf_state_backend_storage_account_name)
     backendAzureRmContainerName: $(tf_state_backend_container_name)
     backendAzureRmKey: '$(tf_state_backend_key_name)'
    env:
      ARM_CLIENT_ID: $(ARM-CLIENT-ID)
      ARM_CLIENT_SECRET: $(ARM-CLIENT-SECRET)
      ARM_SUBSCRIPTION_ID: $(ARM-SUBSCRIPTION-ID)
      ARM_TENANT_ID: $(ARM-TENANT-ID)
```

Demo

- Running a basic pipeline
- Implementing gating

Recap

Recap

Deploy Quickly and Effectively

Easy to Read, Manage and Maintain

Faster Development

Repeatable

Consistent

Various CI/CD Tools

Governance and Controls

GitOps

TRY IT OUT

databricks

HashiCorp Terraform

Azure DevOps

BLOG

Check out our blogs on Databricks, Terraform and DevOps

SQLBITS

https://sqlb.it/?7101

@annawykes

https://www.linkedin.com/in/anna-maria-wykes-31939454

AnnaWykes

@falekmiah

falekmiah.com

FalekMiah01

Code base can be found on GitHub:
https://github.com/FalekMiah01/sqlbits2022

Deploy Databricks Components using Terraform

advancinganalytics.co.uk