**Azure CI/CD Pipeline Documentation**

This YAML script defines a CI/CD pipeline in Azure DevOps for deploying infrastructure using Terraform. The pipeline triggers on commits to the main branch, uses an Ubuntu-based VM for execution, and consists of a single stage that includes multiple jobs and steps for managing Terraform operations.

**Trigger**

A black background with green text

Description automatically generated

* This section specifies that the pipeline should be triggered whenever there is a commit to the main branch.

**Pool**

****

* Defines the environment in which the pipeline runs. Here it specifies that the pipeline should use the latest version of Ubuntu as the virtual machine image.

**Stages**

A black screen with white text

Description automatically generated

* This section defines a stage named TerraformInitPlanApply, which is responsible for deploying infrastructure using Terraform. Within this stage, a job called DeployTerraform is created.

**Steps**

The following are the steps included in the DeployTerraform job:

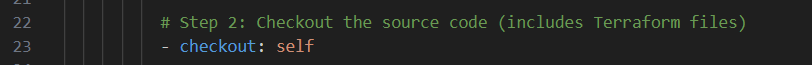
**Step 1: Install Terraform**

**A black screen with text

Description automatically generated**

* This step uses the built-in TerraformInstaller task to install Terraform version 1.9.6 on the agent.

**Step 2: Checkout Source Code**

****

* This step checks out the source code from the repository, which includes the Terraform configuration files needed for deployment.

**Step 3: Initialize Terraform**

**A black screen with white text

Description automatically generated**

* This step runs the terraform init command, which initializes the Terraform working directory. It prepares the backend and downloads any necessary plugins or modules defined in the Terraform configuration files.

**Step 4: Terraform Plan**

**A black screen with white text

Description automatically generated**

* This step executes the terraform plan command, which creates an execution plan and previews the changes that will be made to the infrastructure. The output is saved to a file named tfplan.

**Step 5: Apply Terraform Plan**

**A screen shot of a computer

Description automatically generated**

* This step runs the terraform apply command, which applies the changes required to reach the desired state of the infrastructure as specified in the tfplan file. The -auto-approve flag bypasses the interactive approval prompt.

**Step 6: Publish Terraform State**

**A screen shot of a computer

Description automatically generated**

* This step publishes the Terraform state file (terraform.tfstate) as an artifact of the pipeline. This state file is essential for tracking the current state of the deployed infrastructure and can be used for future operations.

**Conclusions:**

This YAML script automates the deployment of infrastructure using Terraform through Azure DevOps. It handles installation, initialization, planning, and applying changes to the infrastructure, and ensures that the Terraform state is saved for future reference.