Integrating Jenkins with Terraform

Integrating Jenkins with Terraform allows you to automate infrastructure provisioning and management using Infrastructure as Code (IaC) practices. This integration helps in achieving continuous delivery (CD) for infrastructure by leveraging Terraform's ability to provision and manage cloud resources declaratively.

Overview

Jenkins: Jenkins is an open-source automation server that helps automate parts of the software development lifecycle, including building, testing, and deploying applications.

Terraform: Terraform is an open-source IaC tool that allows you to define, provision, and manage infrastructure resources across various cloud platforms in a consistent manner.

By integrating Jenkins with Terraform, you can:

- Automate the provisioning and de-provisioning of infrastructure resources.
- Manage infrastructure changes through version-controlled code.
- Create reusable infrastructure modules and automate testing and deployments.

Step-by-Step Setup

1. Prerequisites

- **Ubuntu machine**: A server or VM running Ubuntu.
- **Jenkins installed**: Jenkins must be installed and running. (Refer to previous setup steps if Jenkins is not installed).
- **Terraform installed**: Terraform must be installed on the Jenkins server.

2. Install Jenkins on Ubuntu

If Jenkins is not already installed, follow these steps:

1. **Update System Packages**:
   ```
   sudo apt update
   sudo apt upgrade -y
   ```

2. **Install Java**:
   Jenkins requires Java to run. Install OpenJDK:
   ```
   sudo apt install openjdk-11-jdk -y
   ```

3. **Add Jenkins Repository and Install Jenkins**:
   ```
   wget -q -O -
   https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo
   apt-key add -

   sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable
binary/ > /etc/apt/sources.list.d/jenkins.list'

   sudo apt update

   sudo apt install jenkins -y
   ```

4. **Start and Enable Jenkins Service**:
   ```
   sudo systemctl start jenkins

   sudo systemctl enable jenkins
   ```

5. **Access Jenkins**:
   Open your web browser and go to
   `http://your_server_ip_or_domain:8080` to complete the Jenkins setup.

3. Install Terraform on Ubuntu

1. **Download the Terraform Binary**:
   Find the latest version of Terraform from the Terraform downloads page. Replace the version number in the command below if necessary.
   ```
   wget
   https://releases.hashicorp.com/terraform/1.6.0/terraform_1.
   6.0_linux_amd64.zip
   ```

2. **Install Unzip Utility**:
   If you don't have unzip installed, install it using:
   ```
   sudo apt install unzip -y
   ```

3. **Unzip and Install Terraform**:
   ```
   unzip terraform_1.6.0_linux_amd64.zip

   sudo mv terraform /usr/local/bin/
   ```

4. **Verify Terraform Installation**:
   ```
   terraform --version
   ```

This should display the installed Terraform version and confirm that Terraform is installed correctly.

4. Configure Jenkins for Terraform Integration

1. **Install Terraform Plugin in Jenkins**:
   - Go to Jenkins Dashboard → Manage Jenkins → Manage Plugins.
   - In the "Available" tab, search for "Terraform" and install the "Terraform" plugin.
   - Restart Jenkins after the plugin is installed.
2. **Configure Terraform in Jenkins**:
   - Go to Jenkins Dashboard → Manage Jenkins → Global Tool Configuration.
   - Scroll down to the "Terraform" section and click "Add Terraform".
   - Provide a name for this Terraform installation (e.g., "Terraform").
   - Ensure that the "Install automatically" checkbox is **unchecked** because Terraform is already installed on the server.
   - Specify the path to the Terraform executable (default is `/usr/local/bin/terraform`).
3. **Add Credentials for Terraform Providers**:
   - Go to Jenkins Dashboard → Manage Jenkins → Manage Credentials.
   - Add new credentials for the cloud provider (e.g., AWS, Azure, Google Cloud) that Terraform will use to manage infrastructure.
   - Choose the appropriate credential type (e.g., "AWS Credentials" for AWS, "Google Service Account from private key" for GCP, etc.) and provide the required information.

5. Create a Jenkins Job to Run Terraform

1. **Create a New Jenkins Job**:
    - Go to Jenkins Dashboard → New Item → Freestyle Project.
    - Enter a name for the job and select "Freestyle project".
    - Click "OK" to create the job.
2. **Configure Source Code Management**:
    - If your Terraform configuration files are stored in a version control system (like Git), configure the source code management section to pull the configuration from the repository.
    - Select "Git" and enter the repository URL and credentials if needed.
3. **Add Build Steps to Initialize and Apply Terraform**:
    - In the "Build" section, click "Add build step" and select "Invoke Terraform".
    - Configure the Terraform step:
        - **Terraform Version**: Select the Terraform installation you configured earlier.
        - **Terraform Command**: Set this to `init` to initialize the Terraform environment.
    - Add another build step to apply the Terraform configuration:
        - **Terraform Version**: Select the same Terraform installation.
        - **Terraform Command**: Set this to `apply` to apply the Terraform configuration.
        - **Options**: Add `-auto-approve` to automatically approve the changes without manual intervention.
4. **Save and Build the Job**:
    - Save the Jenkins job configuration.
    - Click "Build Now" to run the job.

6. Create a Jenkins Pipeline to Run Terraform (Optional)

1. **Create a New Pipeline Job**:
    - Go to Jenkins Dashboard → New Item → Pipeline.
    - Enter a name for the pipeline job and select "Pipeline".
    - Click "OK" to create the pipeline job.
2. **Define the Pipeline Script**:
   Here is a simple pipeline script to run Terraform commands:
   groovy

```
pipeline {

    agent any
```

```
        environment {
           TF_CREDENTIALS = credentials('your-credential-id')
    }

        stages {
             stage('Checkout') {
                  steps {
                       git url:
'https://github.com/your-repo/your-terraform-config.git',
credentialsId: 'your-credentials-id'
                  }
             }

        stage('Terraform Init') {
            steps {
                sh 'terraform init'
            }
        }

        stage('Terraform Apply') {
            steps {
                sh 'terraform apply -auto-approve'
            }
        }
    }
}
```

- Replace `'https://github.com/your-repo/your-terraform-config.git'` with the URL of your Git repository.
- Replace `'your-credentials-id'` with the credentials ID for SCM access.
- Adjust the environment variable and steps according to your cloud provider and setup.

3. **Save and Run the Pipeline**:
    - Save the pipeline configuration.
    - Click "Build Now" to run the pipeline.

7. Example Terraform Script (Optional)

Here's a basic example of a Terraform configuration script to deploy an AWS EC2 instance:

```
provider "aws" {
  region = "us-west-2"
}

resource "aws_instance" "example" {
  ami           = "ami-0c94855ba95c71c99"
  instance_type = "t2.micro"

  tags = {
    Name = "Jenkins-Terraform-Example"
  }
}
```

This script specifies the AWS provider and creates an EC2 instance of type `t2.micro` in the `us-west-2` region.

8. Running and Monitoring Terraform Jobs

- **Trigger the Jenkins Job or Pipeline**: Manually trigger the job or configure it to run automatically on a schedule or based on events (like a Git commit).
- **Monitor the Build**: Check the Jenkins console output for detailed information about the Terraform execution. Jenkins will display the output of each Terraform command, including any errors or warnings.

Conclusion

Integrating Jenkins with Terraform on Ubuntu allows you to automate infrastructure provisioning and management in a CI/CD pipeline. By leveraging Jenkins' automation capabilities and Terraform's declarative infrastructure management, you can efficiently

manage cloud resources, enforce consistency across environments, and streamline your DevOps workflows.