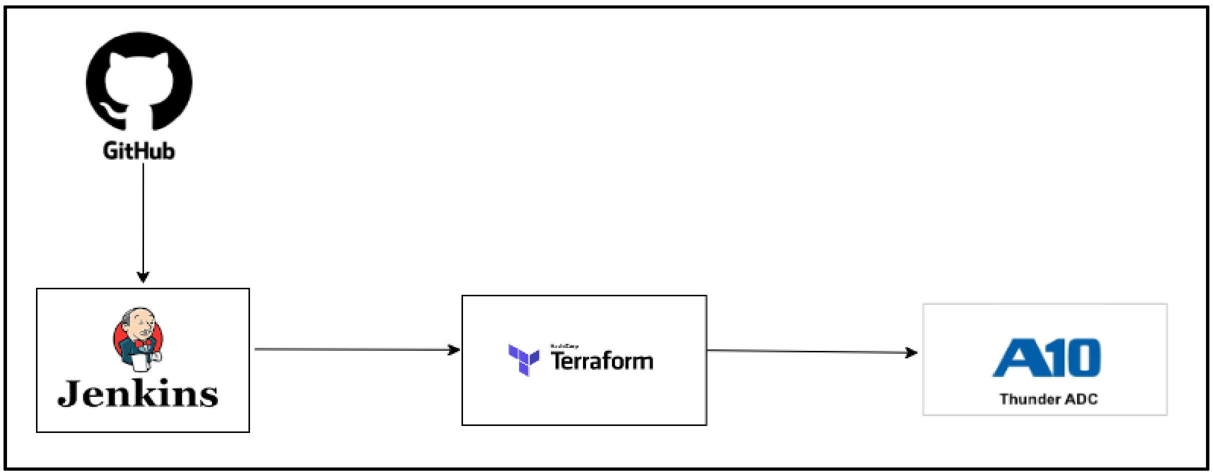
Jenkins Pipeline Configuration Guide

Architecture:



Pre-requisite:

- Jenkins
- Terraform

Install Terraform Software

To install Terraform software on your machine, perform the following steps:

On MacOS

To install Terraform on MacOS, perform the following steps:

1. Run the following commands to download and install the latest version of Terraform:
    a. brew tap hashicorp/tap
    b. brew install hashicorp/tap/terraform

On Ubuntu

1. Download the tool from the official downloads page Terraform Download.
2. Execute the below commands on your machine to install Terraform.
    a. wget https://releases.hashicorp.com/terraform/**{terraform_version}**/terraform_**{terraform_version}**_linux_amd64.zip
    b. unzip terraform_**{terraform_version}**_linux_amd64.zip
    c. mv terraform /usr/local/bin/
3. Verify installation using the below command:
    a. terraform -version

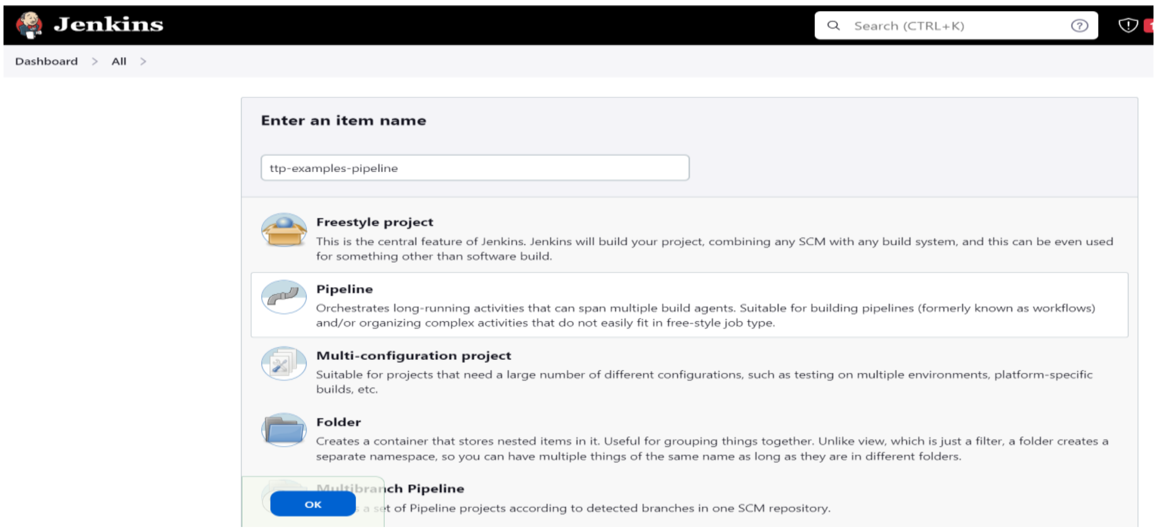
On Windows

1. Download the Terraform 1.5.6 Windows executable file:
    a. Windows 386
    b. Windows amd64
2. Extract the downloaded zip at **C:/Terraform1.5.6.**
3. Edit the **PATH** environment variable to add **C:/Terraform1.5.6/.**

Create Jenkins Project:

1. **Create a new Jenkins pipeline job**

   Goto Jenkins Home > New Item > Create Project with Pipeline.
   Enter the name of the new pipe. Select **Pipeline** and click **OK**.



Jenkins   Search (CTRL+K)

Dashboard > All >

Enter an item name

ttp-examples-pipeline

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

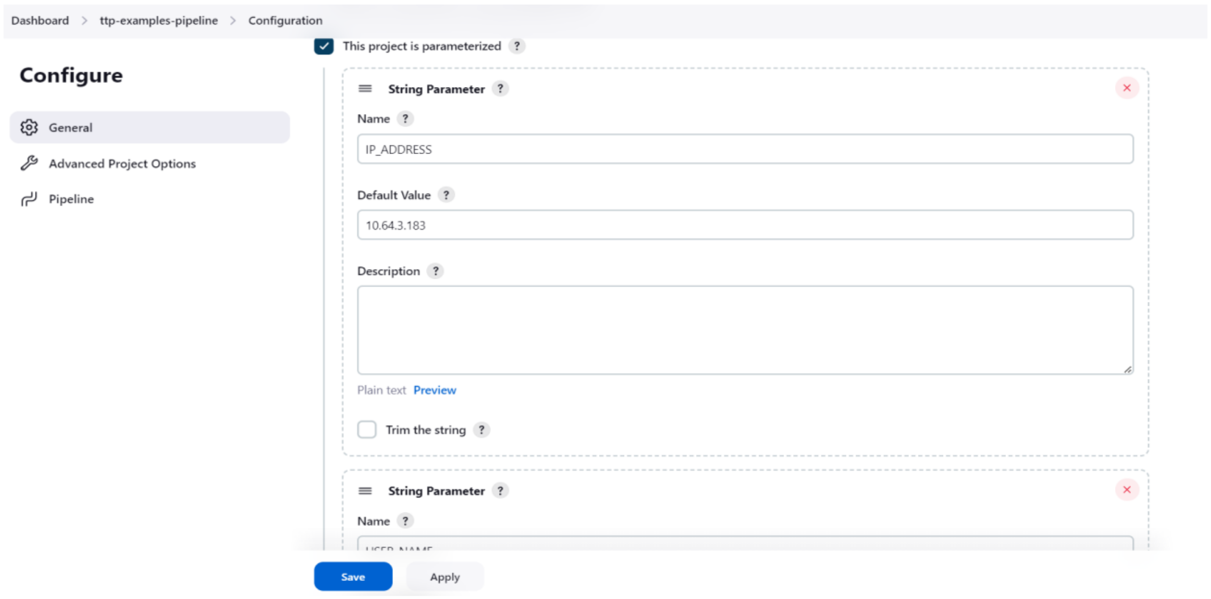
Multibranch Pipeline
a set of Pipeline projects according to detected branches in one SCM repository.

OK

2. **Configure pipeline settings**

   In the **Configure** screen, add the following parameter names in the **General** tab and click **Save** to save the changes:

   - IP_ADDRESS
   - PROVIDER_VERSION
   - USER_NAME
   - PASSWORD
   - TERRAFORM_DESTROY
   - STATE_DIR_PATH
   - EXAMPLE_DIR_PATH
   - GIT_REPOSITORY
   - GIT_BRANCH



Dashboard > ttp-examples-pipeline > Configuration

Configure

General
Advanced Project Options
Pipeline

✓ This project is parameterized

String Parameter
Name: IP_ADDRESS
Default Value: 10.64.3.183
Description:
Plain text Preview
Trim the string

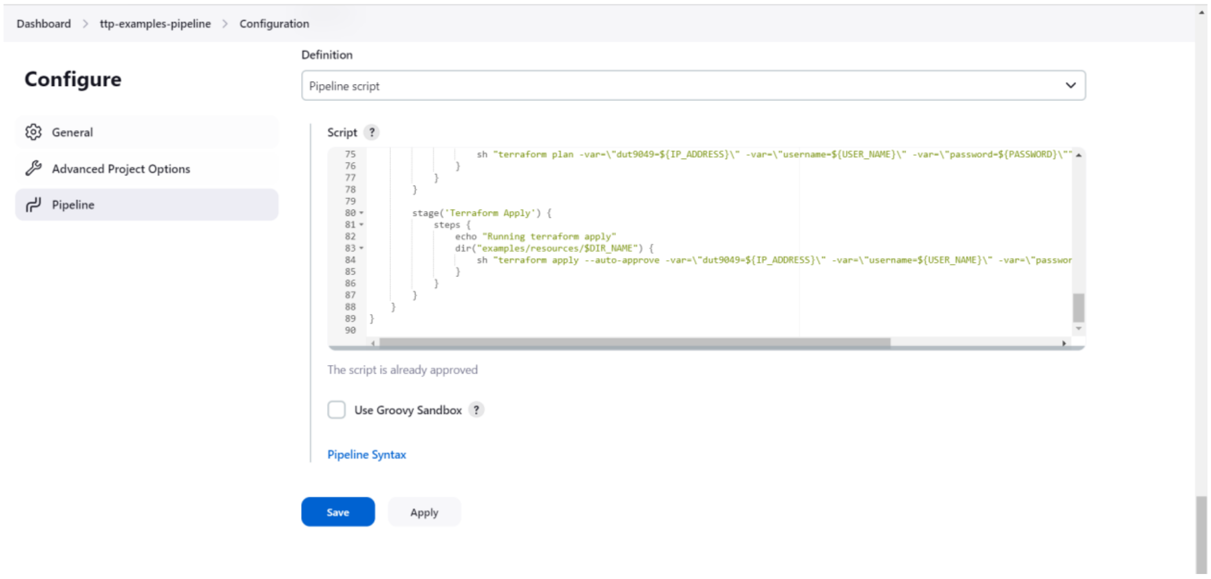
String Parameter
Name: USER_NAME

Save   Apply

Configure advanced settings in the **Advanced Project Options** tab as appropriate.

3. Define the stages and steps of the Jenkins pipeline

Download the pipeline script. Open the script to copy and paste it into the **Script** field of the **Pipeline** tab. Click **Save** to save the changes.



Definition: Pipeline script

Script

```
                    sh "terraform plan -var=\"dut9049=${IP_ADDRESS}\" -var=\"username=${USER_NAME}\" -var=\"password=${PASSWORD}\""
                }
            }
        }

        stage('Terraform Apply') {
            steps {
                echo "Running terraform apply"
                dir("examples/resources/$DIR_NAME") {
                    sh "terraform apply --auto-approve -var=\"dut9049=${IP_ADDRESS}\" -var=\"username=${USER_NAME}\" -var=\"passwor
                }
            }
        }
    }
}
```

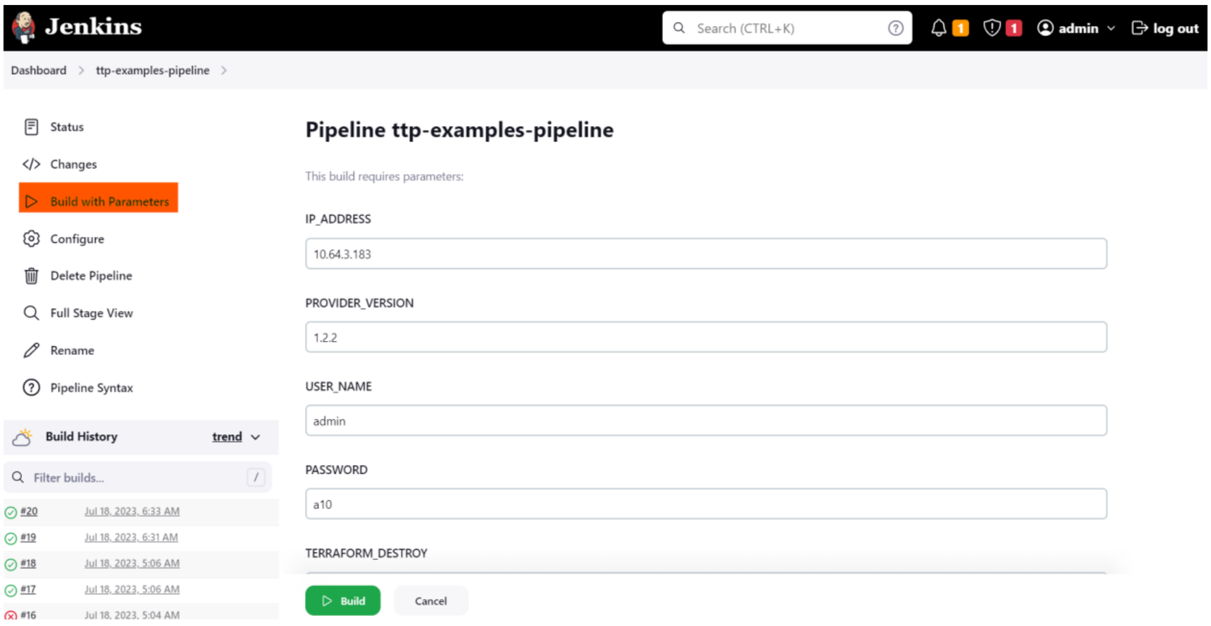
The script is already approved

☐ Use Groovy Sandbox

The new pipeline is created.

4. Execute the pipeline

From the new pipeline page, click **Build with Parameters** to start your pipeline.



Pipeline ttp-examples-pipeline

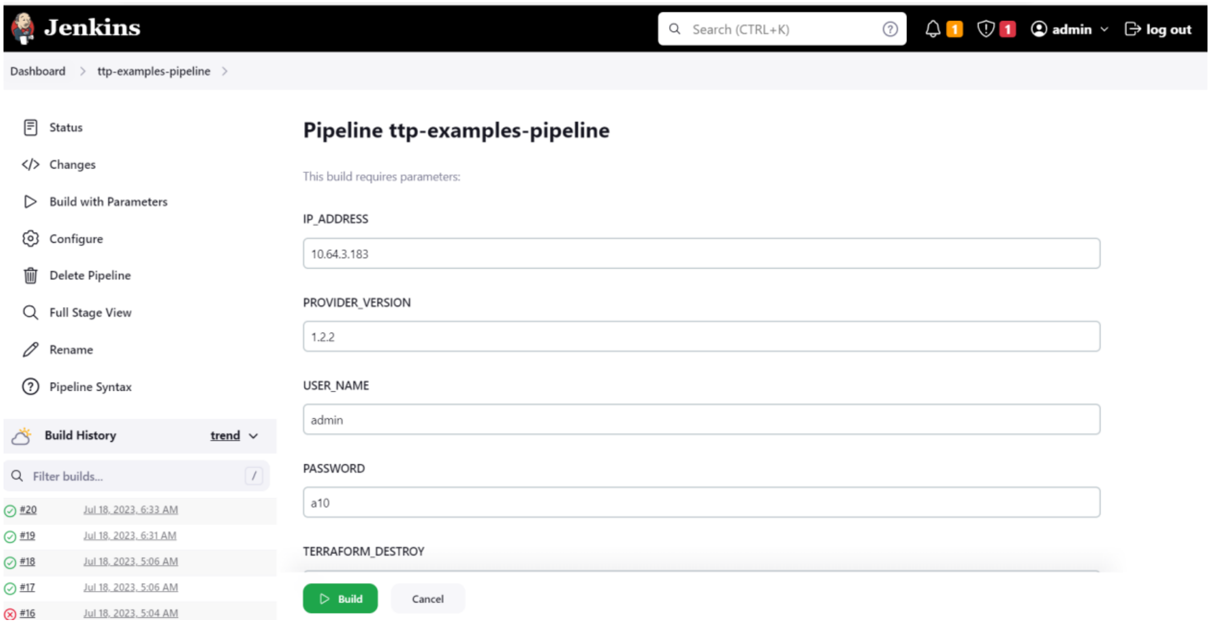
This build requires parameters:

IP_ADDRESS	10.64.3.183
PROVIDER_VERSION	1.2.2
USER_NAME	admin
PASSWORD	a10
TERRAFORM_DESTROY	

✓ #20	Jul 18, 2023, 6:33 AM
✓ #19	Jul 18, 2023, 6:31 AM
✓ #18	Jul 18, 2023, 5:06 AM
✓ #17	Jul 18, 2023, 5:06 AM
✗ #16	Jul 18, 2023, 5:04 AM

5. Update the pipeline parameter values

Enter the parameter values as required and click **Build**.



#20 Jul 18, 2023, 6:33 AM
#19 Jul 18, 2023, 6:31 AM
#18 Jul 18, 2023, 5:06 AM
#17 Jul 18, 2023, 5:06 AM
#16 Jul 18, 2023, 5:04 AM

Pipeline ttp-examples-pipeline

This build requires parameters:

IP_ADDRESS: 10.64.3.183

PROVIDER_VERSION: 1.2.2

USER_NAME: admin

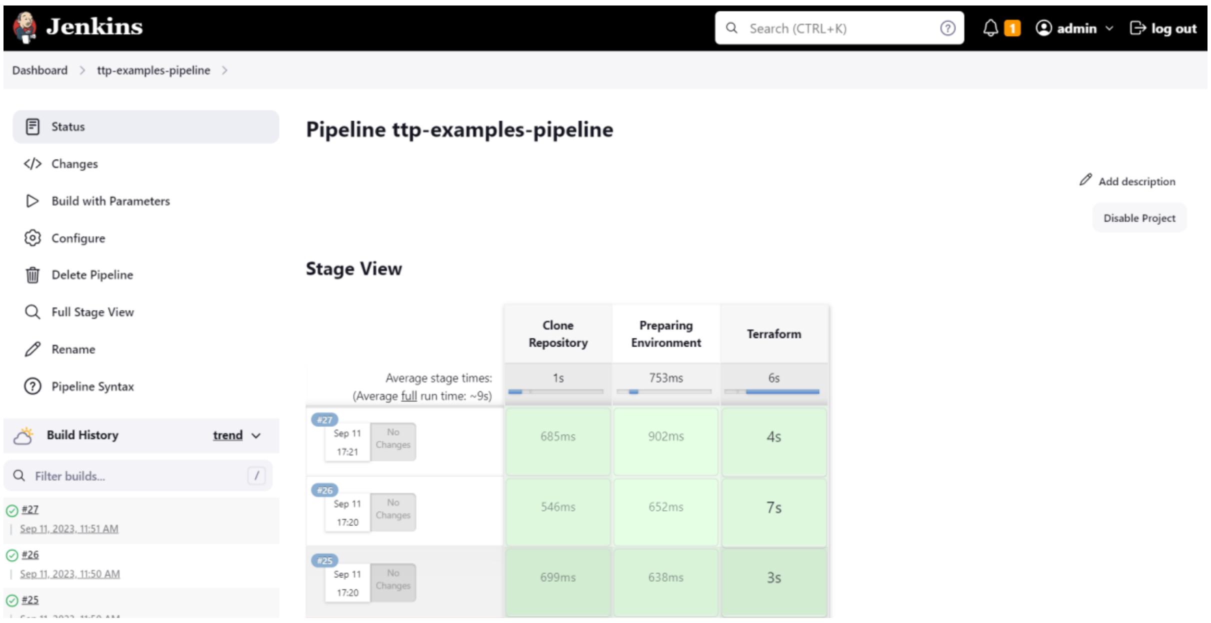
PASSWORD: a10

TERRAFORM_DESTROY

Note: To apply configurations on ACOS, set **"TERRAFORM_DESTROY = 0"**. To remove configurations from ACOS, set **"TERRAFORM_DESTROY = 1"**.

6. View pipeline results

Once the pipeline execution is complete, select the specific build to view the result. The result shows the console output and any generated reports or artifacts.



#27 Sep 11, 2023, 11:51 AM
#26 Sep 11, 2023, 11:50 AM
#25

Pipeline ttp-examples-pipeline

Stage View

	Clone Repository	Preparing Environment	Terraform
Average stage times: (Average full run time: ~9s)	1s	753ms	6s
#27 Sep 11 17:21 No Changes	685ms	902ms	4s
#26 Sep 11 17:20 No Changes	546ms	652ms	7s
#25 Sep 11 17:20 No Changes	699ms	638ms	3s

7. **Pipeline Stages:**
    1. The **"Clone Repository"** stage clones the terraform repository from GitHub.
    2. The **"Prepare Environment"** stage changes the directory, removes state files, and creates necessary Terraform configuration files.
    3. The **"Terraform"**
        - i. Initializes the Terraform environment.
        - ii. Performs the Terraform plan operation.
        - iii. Applies the Terraform changes with auto-approval.