Terraform API

Perquisites:

1. Install the Jenkins HTTP Request plugin if it is not already installed.
2. Set up a TFE OAuth token with permissions to manage workspaces and runs in TFE. You can create a new token in the TFE user settings.
3. Store the TFE OAuth token as a Jenkins credential. You can create a new "Secret text" credential and store the token as the secret value.
4. Install the Terraform plugin for Jenkins. This plugin allows you to run Terraform commands as part of your Jenkins pipeline.
5. Connect your repo with Terraform enterprise.

To create a Terraform workspace using API via a Jenkins pipeline, you can use the following steps:

1. Install the Terraform plugin for Jenkins. This plugin allows you to run Terraform commands as part of your Jenkins pipeline.
2. Create a Jenkins pipeline that includes the necessary steps to create a Terraform workspace using the API. Here's an example:

HTTP Request plugin. Here is an example of what this might look like:

pipeline {

agent any

stages {

stage('Create Terraform Workspace') {

steps {

script {

def token = 'YOUR\_API\_TOKEN'

def organization = 'YOUR\_ORGANIZATION\_NAME'

def workspace = 'YOUR\_WORKSPACE\_NAME'

def response = sh(script: "curl -X POST \

-H 'Authorization: Bearer ${token}' \

-H 'Content-Type: application/vnd.api+json' \

-d '{\"data\":{\"type\":\"workspaces\",\"attributes\":{\"name\":\"${workspace}\"},\"relationships\":{\"organization\":{\"data\":{\"type\":\"organizations\",\"id\":\"${organization}\"}}}}}' \

https://app.terraform.io/api/v2/workspaces",

returnStdout: true)

echo response

}

}

}

}

}

This pipeline uses the **curl** command to make an API call to the Terraform API to create a new workspace. You'll need to replace **YOUR\_API\_TOKEN**, **YOUR\_ORGANIZATION\_NAME**, and **YOUR\_WORKSPACE\_NAME** with your own values.

1. Run the Jenkins pipeline to create the Terraform workspace. You should see the response from the API call in the Jenkins console output.
2. Verify that the Terraform workspace was created by logging in to the Terraform web interface and checking that the workspace exists. You can also use the Terraform CLI to list all workspaces:

terraform workspace list

Terraform workspace execution:

1. make a POST request to the **POST /workspaces/${workspaceId}/runs**
2. use a while loop to monitor the status of the run by making a GET request to the **GET /runs/${runId}** API endpoint

pipeline {

agent any

stages {

stage('Create TFE Run') {

steps {

script {

def tfeUrl = 'https://app.terraform.io'

def tfeOAuthToken = credentials('tfe-oauth-token')

def workspaceId = 'my-workspace-id'

def apiUrl = "${tfeUrl}/api/v2/workspaces/${workspaceId}/runs"

def requestBody = '''

{

"data": {

"type": "runs",

"attributes": {

"is-destroy": false

}

}

}

'''

def response = httpRequest customHeaders: [[name: 'Authorization', value: "Bearer ${tfeOAuthToken}"]], contentType: 'APPLICATION\_JSON', httpMode: 'POST', requestBody: requestBody, url: apiUrl

def runId = response.data.id

echo "TFE Run created with ID: ${runId}"

}

}

}

stage('Monitor TFE Run') {

steps {

script {

def tfeUrl = 'https://app.terraform.io'

def tfeOAuthToken = credentials('tfe-oauth-token')

def runId = 'my-run-id'

def apiUrl = "${tfeUrl}/api/v2/runs/${runId}"

while (true) {

def response = httpRequest customHeaders: [[name: 'Authorization', value: "Bearer ${tfeOAuthToken}"]], url: apiUrl

def status = response.data.attributes.status

if (status == 'planning' || status == 'applying') {

echo "TFE Run is still in progress with status: ${status}"

sleep(10)

} else if (status == 'completed') {

echo "TFE Run completed with status: ${response.data.attributes.status}"

break

} else {

echo "TFE Run failed with status: ${response.data.attributes.status}"

break

}

}

}

}

}

}

}