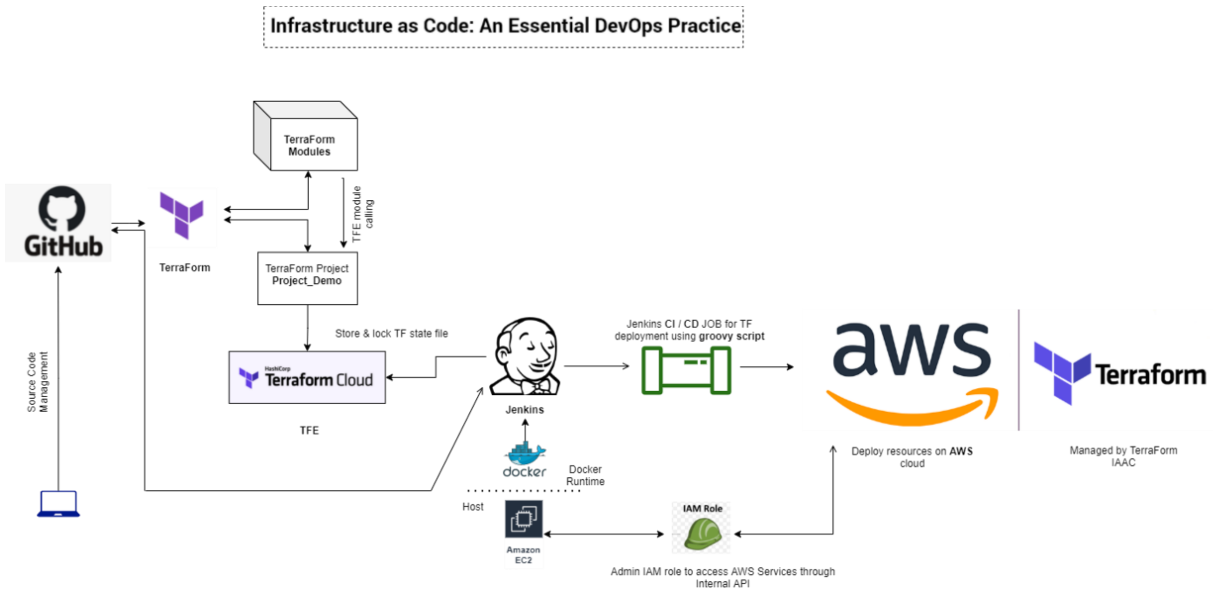
CI-CD project to deploy Infrastructure @ **AWS** using **Terraform** IaC.

Detailed Workflow >>>


Infrastructure as Code: An Essential DevOps Practice
TerraForm Modules
TFE module calling
GitHub
TerraForm
TerraForm Project Project_Demo
Source Code Management
Store & lock TF state file
Terraform Cloud
TFE
Jenkins
Jenkins CI / CD JOB for TF deployment using groovy script
aws
Terraform
docker
Docker Runtime
Host
Amazon EC2
IAM Role
Admin IAM role to access AWS Services through Internal API
Deploy resources on AWS cloud
Managed by TerraForm IAAC

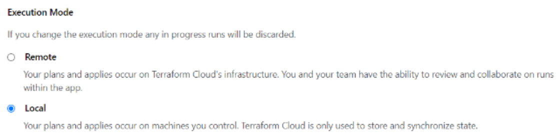
WorkFlow Components (High level) : GitOps , Terraform IaC , TF cloud(Free tier), TF modules, Project DIR to deploy resources to Provider: AWS, CI/CD: Jenkins(Hosting: EC2, Runtime Docker),EC2-IAM-ROLE for AWS-Services-API to deploy account Services-API access, CI-CD JOB: groovy scripted pipeline with business logic, Deploy and Destroy resources to AWS Cloud.

- Used Terraform-Module - AWS: {SCM}/../AWS_modules/modules/*

- Deployed "project_demo" resources using TF modules: {SCM}/../AWS_modules/projects/project_demo/*

Note: z_backend-remote-tfcloud-state.tf is capable to auto create TF cloud workspace based on local workspace selection.

Created workspace at TF cloud has Default **Execution mode:** Remote. Here Called Module for this project **“../{Module_PATH}”**is **local source**, hence TF cloud workspace need to set > **Execution mode:** Local


Execution Mode

If you change the execution mode any in progress runs will be discarded.

○ Remote

Your plans and applies occur on Terraform Cloud's infrastructure. You and your team have the ability to review and collaborate on runs within the app.

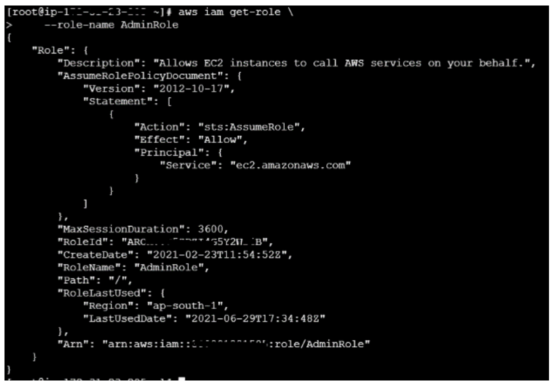
● Local

Your plans and applies occur on machines you control. Terraform Cloud is only used to store and synchronize state.

Jenkins (CI CD Tool) on Docker:

EC2: >>

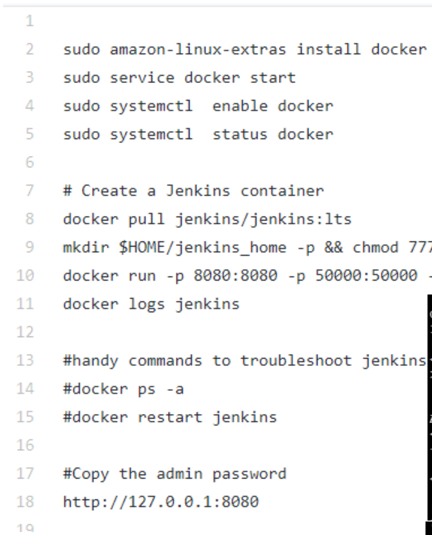
Step 1: Done Setup Ec2 instance with AL2 OS flavor and dependent configs. Attached Preconfigured AdminRole for EC2.


```
[root@ip-[redacted] ~]# aws iam get-role \
>     --role-name AdminRole
{
    "Role": {
        "Description": "Allows EC2 instances to call AWS services on your behalf.",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Action": "sts:AssumeRole",
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "ec2.amazonaws.com"
                    }
                }
            ]
        },
        "MaxSessionDuration": 3600,
        "RoleId": "[redacted]",
        "CreateDate": "2021-02-23T11:54:52Z",
        "RoleName": "AdminRole",
        "Path": "/",
        "RoleLastUsed": {
            "Region": "ap-south-1",
            "LastUsedDate": "2021-06-29T17:34:48Z"
        },
        "Arn": "arn:aws:iam::[redacted]:role/AdminRole"
    }
}
```

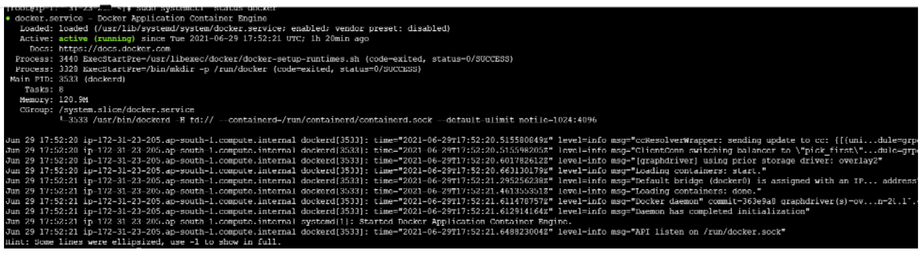
EC2-IAM-ROLE to get privilege on AWS Services-API to deploy AWS account Services-API access (Admin Role)

Step 2:

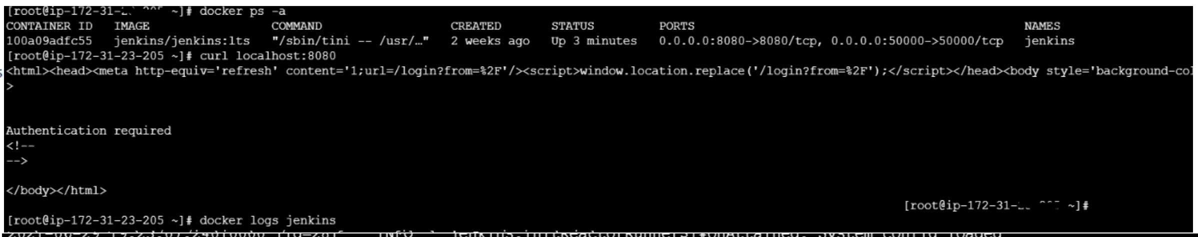
Install & configure Docker runtime & Docker PV & Jenkins setup on Docker.



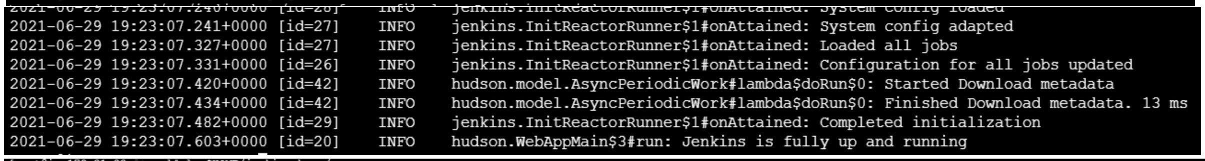
```
sudo amazon-linux-extras install docker -y
sudo service docker start
sudo systemctl  enable docker
sudo systemctl  status docker

# Create a Jenkins container
docker pull jenkins/jenkins:lts
mkdir $HOME/jenkins_home -p && chmod 777 $HOME/jenkins_home
docker run -p 8080:8080 -p 50000:50000 -d -v $HOME/jenkins_home:/var/jenkins_home --name jenkins jenkins/jenkins:lts
docker logs jenkins

#handy commands to troubleshoot jenkins
#docker ps -a
#docker restart jenkins

#Copy the admin password
http://127.0.0.1:8080
```



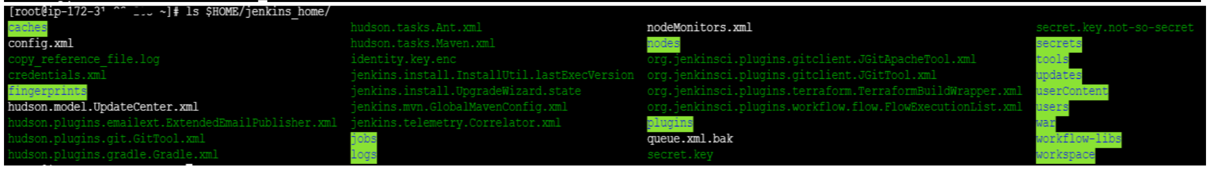
```
[root@ip-1[redacted]-31-23-[redacted] ~]# sudo systemctl  status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2021-06-29 17:52:21 UTC; 1h 20min ago
     Docs: https://docs.docker.com
  Process: 3440 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
  Process: 3328 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
 Main PID: 3533 (dockerd)
    Tasks: 8
   Memory: 120.9M
   CGroup: /system.slice/docker.service
           └─3533 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=1024:4096

Jun 29 17:52:20 ip-172-31-23-205.ap-south-1.compute.internal dockerd[3533]: time="2021-06-29T17:52:20.515580849Z" level=info msg="ccResolverWrapper: sending update to cc: {[{uni...dule=grp
Jun 29 17:52:20 ip-172-31-23-205.ap-south-1.compute.internal dockerd[3533]: time="2021-06-29T17:52:20.515598205Z" level=info msg="ClientConn switching balancer to \"pick_first\"...dule=grp
Jun 29 17:52:20 ip-172-31-23-205.ap-south-1.compute.internal dockerd[3533]: time="2021-06-29T17:52:20.601782612Z" level=info msg="[graphdriver] using prior storage driver: overlay2"
Jun 29 17:52:20 ip-172-31-23-205.ap-south-1.compute.internal dockerd[3533]: time="2021-06-29T17:52:20.663130179Z" level=info msg="Loading containers: start."
Jun 29 17:52:21 ip-172-31-23-205.ap-south-1.compute.internal dockerd[3533]: time="2021-06-29T17:52:21.295256238Z" level=info msg="Default bridge (docker0) is assigned with an IP... address
Jun 29 17:52:21 ip-172-31-23-205.ap-south-1.compute.internal dockerd[3533]: time="2021-06-29T17:52:21.461355351Z" level=info msg="Loading containers: done."
Jun 29 17:52:21 ip-172-31-23-205.ap-south-1.compute.internal dockerd[3533]: time="2021-06-29T17:52:21.611478757Z" level=info msg="Docker daemon" commit=363e9a8 graphdriver(s)=ov...n=20.1
Jun 29 17:52:21 ip-172-31-23-205.ap-south-1.compute.internal dockerd[3533]: time="2021-06-29T17:52:21.612914164Z" level=info msg="Daemon has completed initialization"
Jun 29 17:52:21 ip-172-31-23-205.ap-south-1.compute.internal systemd[1]: Started Docker Application Container Engine.
Jun 29 17:52:21 ip-172-31-23-205.ap-south-1.compute.internal dockerd[3533]: time="2021-06-29T17:52:21.648823004Z" level=info msg="API listen on /run/docker.sock"
Hint: Some lines were ellipsized, use -l to show in full.
```



```
[root@ip-172-31-[redacted] ~]# docker ps -a
CONTAINER ID   IMAGE                 COMMAND                  CREATED       STATUS         PORTS                                              NAMES
100a09adfc55   jenkins/jenkins:lts   "/sbin/tini -- /usr/…"   2 weeks ago   Up 3 minutes   0.0.0.0:8080->8080/tcp, 0.0.0.0:50000->50000/tcp   jenkins
[root@ip-172-31-23-205 ~]# curl localhost:8080
<html><head><meta http-equiv='refresh' content='1;url=/login?from=%2F'/><script>window.location.replace('/login?from=%2F');</script></head><body style='background-col
>


Authentication required
<!--
-->

</body></html>
                                                                                                   [root@ip-172-31-[redacted] ~]#
[root@ip-172-31-23-205 ~]# docker logs jenkins
```

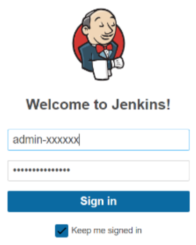


```
2021-06-29 19:23:07.240+0000 [id=28]    INFO    jenkins.InitReactorRunner$1#onAttained: System config loaded
2021-06-29 19:23:07.241+0000 [id=27]    INFO    jenkins.InitReactorRunner$1#onAttained: System config adapted
2021-06-29 19:23:07.327+0000 [id=27]    INFO    jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
2021-06-29 19:23:07.331+0000 [id=26]    INFO    jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
2021-06-29 19:23:07.420+0000 [id=42]    INFO    hudson.model.AsyncPeriodicWork#lambda$doRun$0: Started Download metadata
2021-06-29 19:23:07.434+0000 [id=42]    INFO    hudson.model.AsyncPeriodicWork#lambda$doRun$0: Finished Download metadata. 13 ms
2021-06-29 19:23:07.482+0000 [id=29]    INFO    jenkins.InitReactorRunner$1#onAttained: Completed initialization
2021-06-29 19:23:07.603+0000 [id=20]    INFO    hudson.WebAppMain$3#run: Jenkins is fully up and running
```



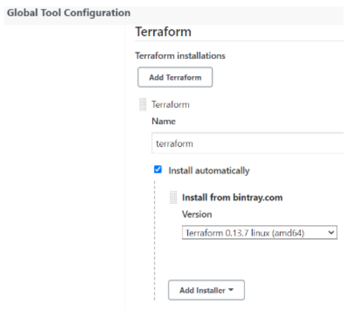
```
[root@ip-172-3[redacted] ~]# ls $HOME/jenkins_home/
caches                                            hudson.tasks.Ant.xml                         nodeMonitors.xml                                          secret.key.not-so-secret
config.xml                                        hudson.tasks.Maven.xml                       nodes                                                     secrets
copy_reference_file.log                           identity.key.enc                             org.jenkinsci.plugins.gitclient.JGitApacheTool.xml        tools
credentials.xml                                   jenkins.install.InstallUtil.lastExecVersion  org.jenkinsci.plugins.gitclient.JGitTool.xml              updates
fingerprints                                      jenkins.install.UpgradeWizard.state          org.jenkinsci.plugins.terraform.TerraformBuildWrapper.xml userContent
hudson.model.UpdateCenter.xml                     jenkins.mvn.GlobalMavenConfig.xml            org.jenkinsci.plugins.workflow.flow.FlowExecutionList.xml users
hudson.plugins.emailext.ExtendedEmailPublisher.xml jenkins.telemetry.Correlator.xml            plugins                                                   war
hudson.plugins.git.GitTool.xml                    jobs                                         queue.xml.bak                                             workflow-libs
hudson.plugins.gradle.Gradle.xml                  logs                                         secret.key                                                workspace
```

Step 3: Verify Jenkins & Docker PV path Up and running.



```
# Install suggested plugins
# Create a user
# Manage jenkins
# Manage plugins
# Search for Terraform in Available and install without restart
# Back to Manage jenkins
# Global Tool Configuration
# Add Terraform
# Name: terraform
# Install automatically
# Version - latest for linux (amd64)  (TF- 0.13 used)
# Click Save


# Go to credentials -> global

# Create a credential of type secret text with ID AWS_ACCESS_KEY_ID and the access key as the secret
# Create a credential of type secret text with ID AWS_SECRET_ACCESS_KEY and the access secret as the secret

#(OPTIONAL) for TF_API_TOKEN run base64 -w0 credentials.tfrc.json get the outpout in single string : paste the value after Create a credential of type secret text with ID TF_API_TOKEN


# Create a new item
# Name: TF-deploy
# Type pipeline
# Select poll SCM
# Definition: Pipeline script from SCM
# SCM: Git
# Repo URL: YOUR_REPO_URL
# Jenkinsfile Script path
# Uncheck lightweight checkout


# Run a new build WITH parameters
#verify parameters env values before build
```

Step 4: Setup Jenkins & GIT, terraform & AWS & Global creds based on this project requirement >>>>>



Global Tool Configuration

Terraform

Terraform installations

Add Terraform

Terraform

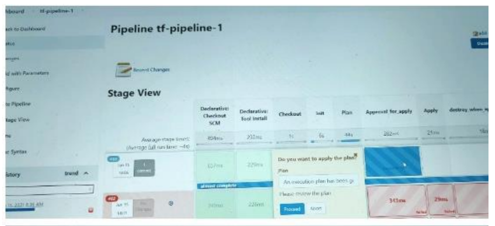
Name: terraform

☑ Install automatically

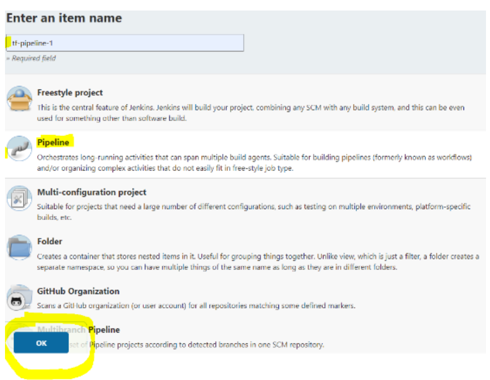
Install from bintray.com

Version: Terraform 0.13.7 linux (amd64)

Add Installer



Step 5: Jenkins Pipeline setup >>>



Enter an item name

tf-pipeline-1

» Required field

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.

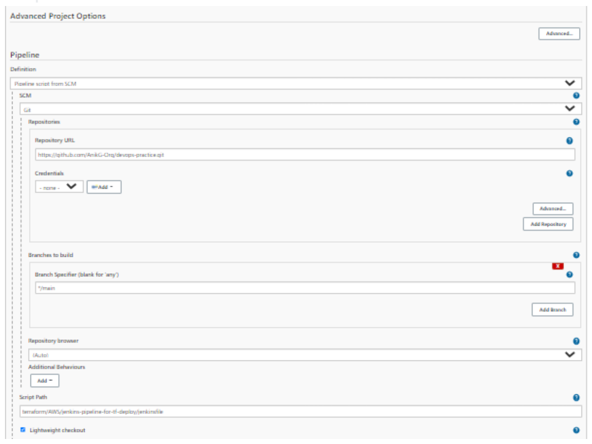
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

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.

GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

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

OK



Advanced Project Options

Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories

Repository URL: https://github.com/AnkG-Org/devops-practice.git

Credentials: - none -

Branches to build

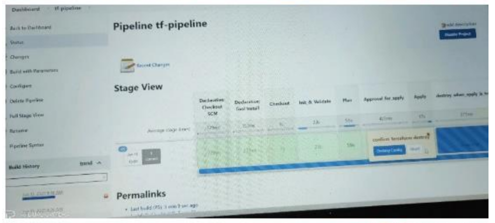
Branch Specifier (blank for 'any'): */main

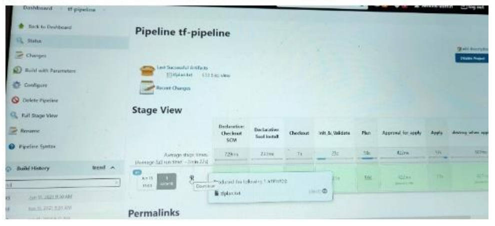
Repository browser: (Auto)

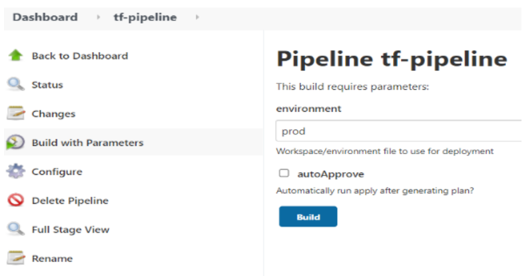
Additional Behaviours

Script Path: terraform/AWS/jenkins-pipeline-for-tf-deploy/Jenkinsfile

☑ Lightweight checkout







Pipeline tf-pipeline

This build requires parameters:

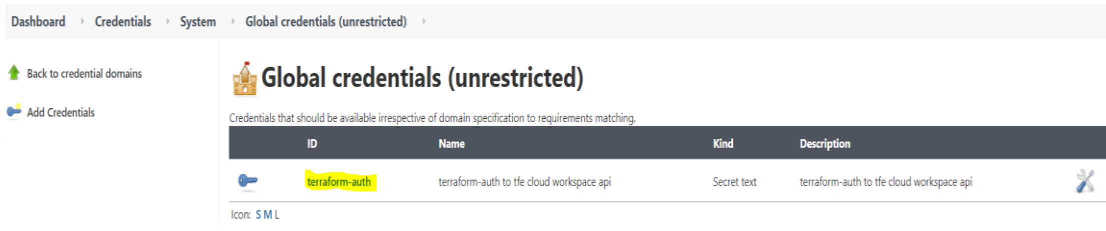
environment: prod

Workspace/environment file to use for deployment

☐ autoApprove

Automatically run apply after generating plan?

Build


Dashboard › Credentials › System › Global credentials (unrestricted) ›
Back to credential domains
Add Credentials

Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
terraform-auth	terraform-auth to tfe cloud workspace api	Secret text	terraform-auth to tfe cloud workspace api

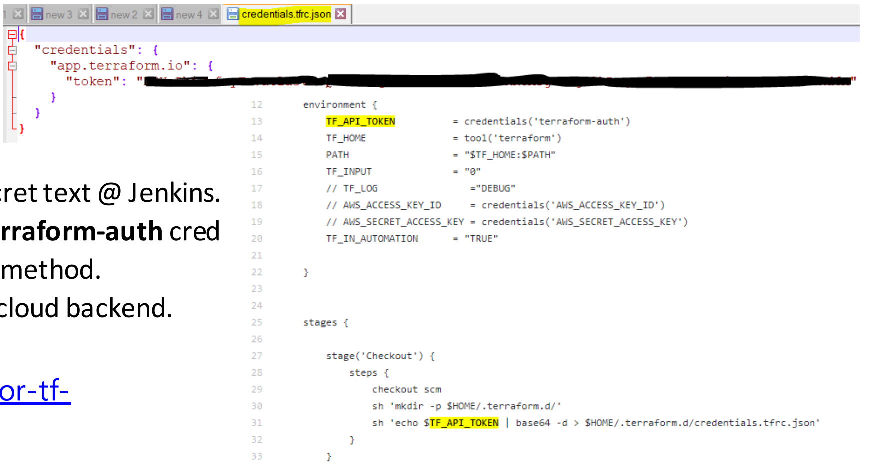
Icon: S M L

Key Note: Here I Configured **terraform-auth** to tf cloud resources from Jenkins pipeline via TF **API** token.

credentials.tfrc.json template with API Token >>

Encrypt credentials.tfrc.json

- base64 -w0 credentials.tfrc.json > secret_file.txt
- Will get a single sting secure bas64 output.
- That will output store as **terraform-auth** as secret text @ Jenkins.
- environment TF_API_TOKEN used to will get **terraform-auth** cred apply usage at stage Checkout with decryption method.
- Now Pipeline is capable to communicate to TF cloud backend.



```
{
  "credentials": {
    "app.terraform.io": {
      "token": "[redacted]"
    }
  }
}
```

```
    environment {
        TF_API_TOKEN          = credentials('terraform-auth')
        TF_HOME               = tool('terraform')
        PATH                  = "$TF_HOME:$PATH"
        TF_INPUT              = "0"
        // TF_LOG               ="DEBUG"
        // AWS_ACCESS_KEY_ID     = credentials('AWS_ACCESS_KEY_ID')
        // AWS_SECRET_ACCESS_KEY = credentials('AWS_SECRET_ACCESS_KEY')
        TF_IN_AUTOMATION      = "TRUE"

    }


    stages {

        stage('Checkout') {
            steps {
                checkout scm
                sh 'mkdir -p $HOME/.terraform.d/'
                sh 'echo $TF_API_TOKEN | base64 -d > $HOME/.terraform.d/credentials.tfrc.json'
            }
        }
```

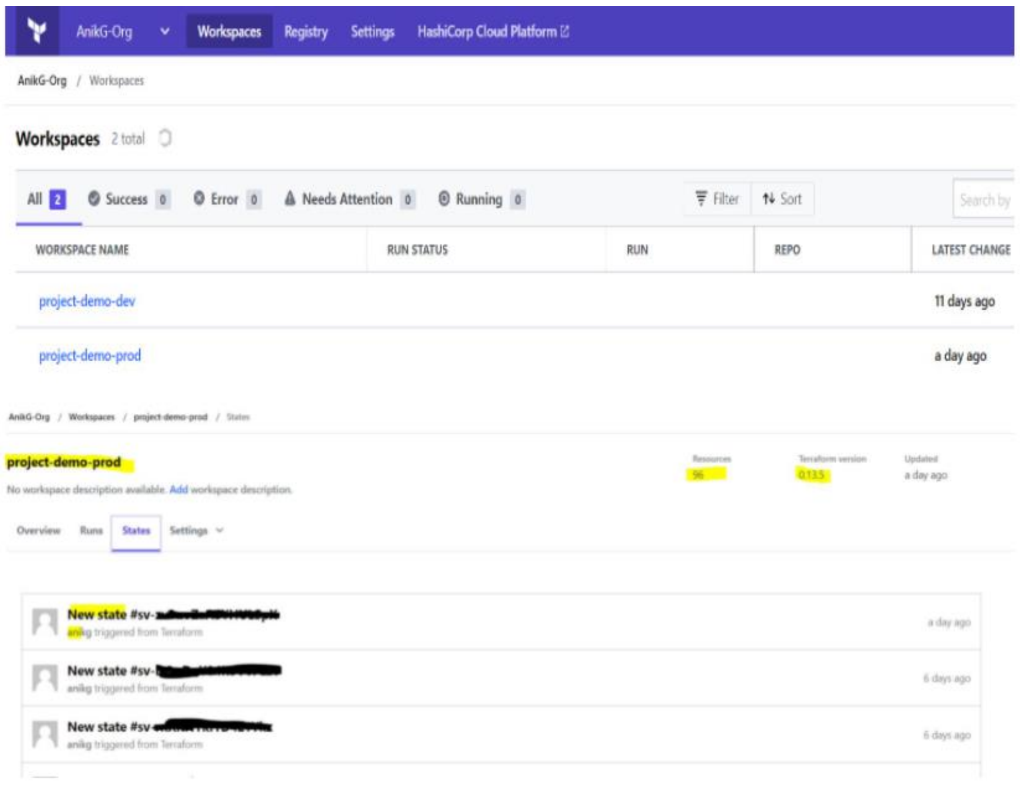
- Jenkins_Pipeline: [{SCM}/.../jenkins-pipeline-for-tf-deploy/jenkinsfile]

This Jenkinsfile(Groovy script) use custom installed plugin: `"org.jenkinsci.plugins.terraform.TerraformInstallation"  "terraform"` **custom name

This pipeline has validations, multiple conditions and approval stages , destroy condition, and artifact output.

Step 5: Verify your state file and status at TF CLOUD: https://app.terraform.io/

& Verify AWS deployments.



Workspaces 2 total

All 2 | Success 0 | Error 0 | Needs Attention 0 | Running 0

WORKSPACE NAME	RUN STATUS	RUN	REPO	LATEST CHANGE
project-demo-dev				11 days ago
project-demo-prod				a day ago

AnikG-Org / Workspaces / project-demo-prod / States

project-demo-prod

No workspace description available. Add workspace description.

Resources	Terraform version	Updated
96	0.13.5	a day ago

Overview | Runs | States | Settings

New state #sv-[redacted]
anikg triggered from Terraform — a day ago

New state #sv-[redacted]
anikg triggered from Terraform — 6 days ago

New state #sv[redacted]
anikg triggered from Terraform — 6 days ago