Here are the **step-by-step instructions** to set up your **Jenkins container with AWS CLI and Terraform installed**, configure AWS credentials, and set up Jenkins for Terraform automation.

## **🚀 Step 1: Create/Edit the Dockerfile**

1. On your Jenkins server, navigate to the directory where you want to create the Dockerfile.

Create a new Dockerfile or edit an existing one:  
  
 nano Dockerfile

Add the following content to install **AWS CLI** and **Terraform** inside the Jenkins container:  
  
 FROM jenkins/jenkins:2.492.2-jdk17

USER root

# Install dependencies

RUN apt-get update && apt-get install -y \

curl \

unzip \

jq \

awscli \

lsb-release \

gnupg

# Install Terraform

RUN curl -fsSL https://apt.releases.hashicorp.com/gpg | gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg && \

echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb\_release -cs) main" | tee /etc/apt/sources.list.d/hashicorp.list && \

apt-get update && \

apt-get install -y terraform

USER jenkins

# Install Jenkins plugins

RUN jenkins-plugin-cli --plugins "blueocean docker-workflow"

1. Save and exit (CTRL+X, then Y, then Enter).

## **🚀 Step 2: Build and Run the Jenkins Container**

Run the following commands **one by one** to **rebuild** and **restart** Jenkins with AWS CLI and Terraform installed.

# Build the new Jenkins image

docker build -t myjenkins-blueocean:2.492.2-2 .

# Stop and remove the existing Jenkins container

docker stop jenkins-blueocean

docker rm jenkins-blueocean

# Run the new Jenkins container

docker run \

--name jenkins-blueocean \

--restart=on-failure \

--detach \

--network jenkins \

--env DOCKER\_HOST=tcp://docker:2376 \

--env DOCKER\_CERT\_PATH=/certs/client \

--env DOCKER\_TLS\_VERIFY=1 \

--publish 8080:8080 \

--publish 50000:50000 \

--volume jenkins-data:/var/jenkins\_home \

--volume jenkins-docker-certs:/certs/client:ro \

myjenkins-blueocean:2.492.2-2

✅ **Jenkins is now running with AWS CLI and Terraform inside the container.**

## **🚀 Step 3: Create AWS Access Key & Secret Key**

1. **Log in to AWS Console**
   * Navigate to **IAM (Identity & Access Management)**
   * Click **Users** → Select your Jenkins user (or create a new one)
   * Go to the **Security credentials** tab
2. **Create Access Key**
   * Click **Create Access Key**
   * Select **CLI** use case
   * Copy the **AWS Access Key** and **AWS Secret Key**

## **🚀 Step 4: Configure AWS Credentials in Jenkins**

1. **Go to Jenkins Dashboard → Manage Jenkins → Manage Credentials.**
2. **Under "Global Credentials", add:**
   * **AWS\_ACCESS\_KEY\_ID** → **ID: AWS\_ACCESS\_KEY**
   * **AWS\_SECRET\_ACCESS\_KEY** → **ID: AWS\_SECRET\_KEY**

## **🚀 Step 5: Configure Jenkins Pipeline**

1. Go to **Jenkins → New Item → Pipeline**

Set **Pipeline script from SCM** and enter your Git repository:  
  
 https://github.com/arjunachari12/jenkins-tf-pipeline

Use the following Jenkinsfile for **Terraform automation**:  
  
 pipeline {

agent any // Runs on Jenkins default agent

environment {

AWS\_ACCESS\_KEY\_ID = credentials('AWS\_ACCESS\_KEY')

AWS\_SECRET\_ACCESS\_KEY = credentials('AWS\_SECRET\_KEY')

}

stages {

stage('Checkout') {

steps {

git 'https://github.com/arjunachari12/jenkins-tf-pipeline.git'

}

}

stage('Terraform Init') {

steps {

sh 'terraform init'

}

}

stage('Terraform Plan') {

steps {

sh 'terraform plan -out=tfplan'

}

}

stage('Terraform Apply') {

steps {

sh 'terraform apply -auto-approve'

}

}

}

post {

success {

sh 'terraform destroy -auto-approve' // Optional cleanup

}

}

}

1. **Save and run the pipeline.**

## **✅ Final Outcome**

✅ **Jenkins container now includes AWS CLI & Terraform** ✅ **AWS credentials securely stored in Jenkins** ✅ **Terraform pipeline fully automated**

Now, every time Jenkins runs this pipeline, it will **provision AWS resources** using Terraform! 🚀 Let me know if you need any refinements.