**Proof of Concept: CI/CD Pipeline for Amazon EKS using Jenkins and Terraform**

**Objective:**

To demonstrate the creation and management of an Amazon EKS (Elastic Kubernetes Service) cluster using a Jenkins CI/CD pipeline, with Terraform as the Infrastructure as Code (IaC) tool. The pipeline automates the process of initializing, validating, planning, and applying Terraform configurations for EKS.

**Architecture Overview:**

1. **Version Control**: GitHub repository containing Terraform configuration files for EKS.
2. **CI/CD Tool**: Jenkins running on an EC2 instance.
3. **Infrastructure Management**: Terraform for provisioning and managing EKS clusters.
4. **Cloud Provider**: AWS (Amazon Web Services).

**Setup and Configuration:**

**1. Repository Setup**

**Commands Executed:**

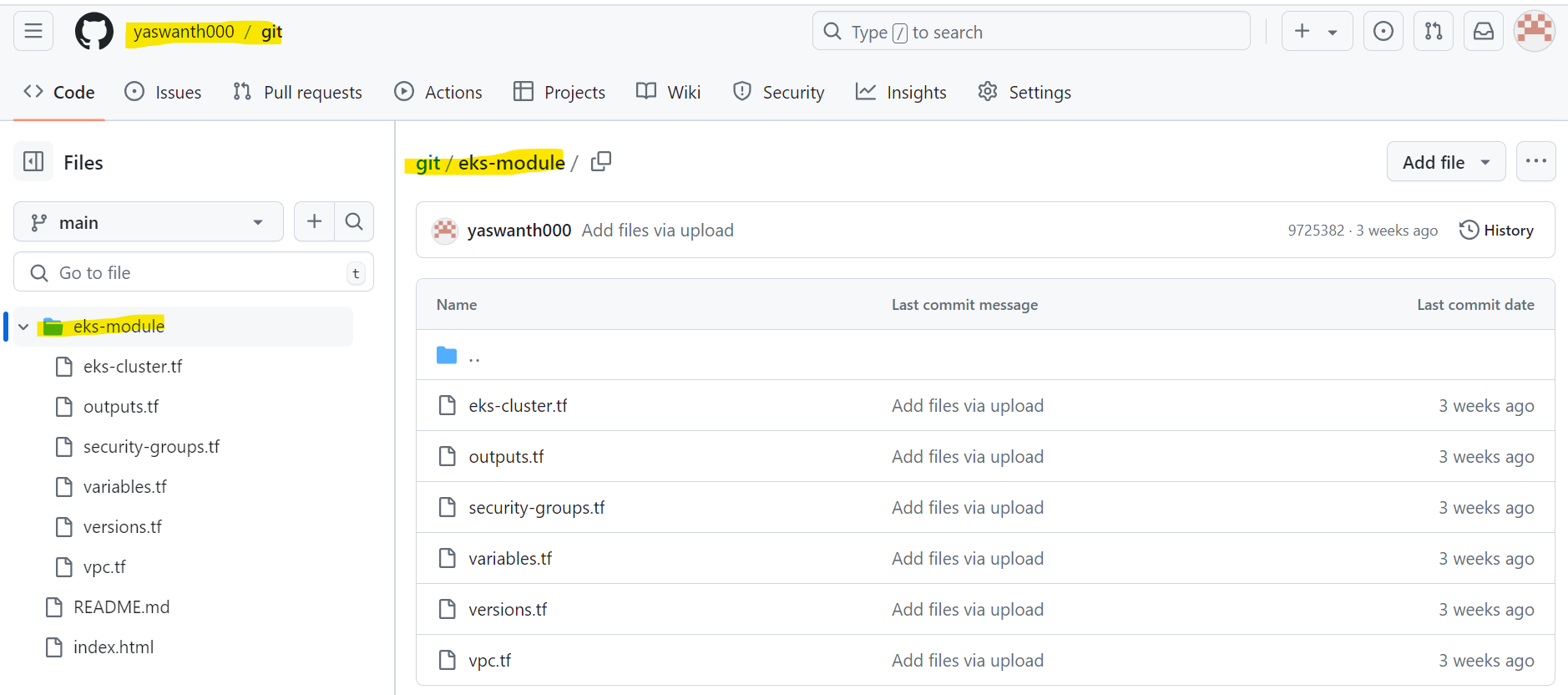
Git init

Git remote add origin [remote-url]

Git add .

GIT commit -m "COMMENT"

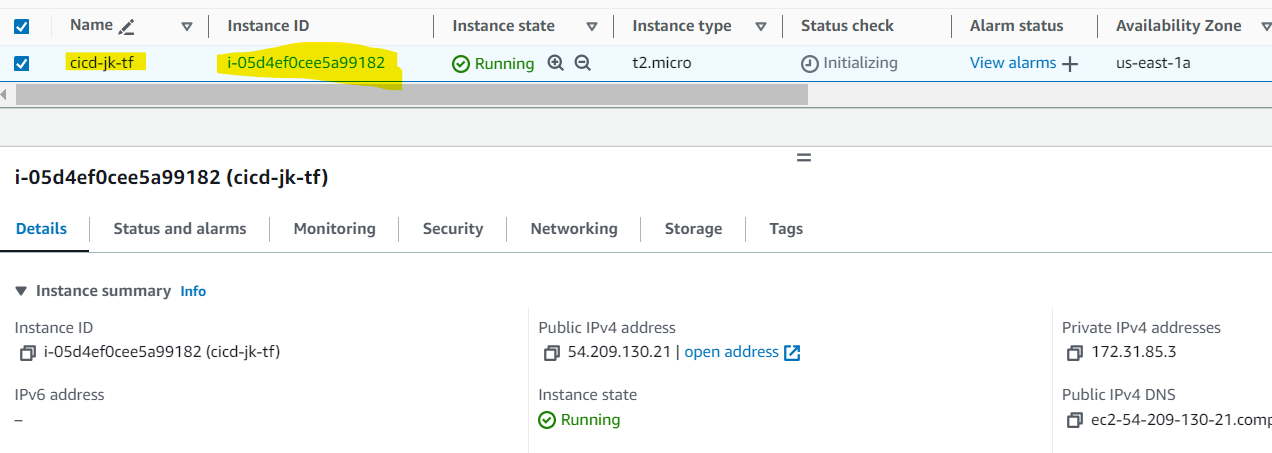
Git push -u origin main --force



**2. EC2 Instance Setup**

**Script: jk-tf.sh**

A shell script used to configure an EC2 instance with necessary software and tools.



$vi jk-tf.sh

#!/bin/bash

set -e # Exit on any error

# Install Java

sudo dnf update -y

sudo dnf install java-17-amazon-corretto -y

java -version

# Install Git

sudo yum install git -y

# Install Jenkins

sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo

sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key

sudo dnf install jenkins -y

sudo systemctl enable jenkins

sudo systemctl start jenkins

# Install Terraform

sudo yum install -y yum-utils shadow-utils

sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo

sudo yum -y install terraform

# Install Kubernetes CLI

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"

echo "$(cat kubectl.sha256) kubectl" | sha256sum --check

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

# Install AWS CLI

sudo yum install unzip -y

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

echo "Installation complete."

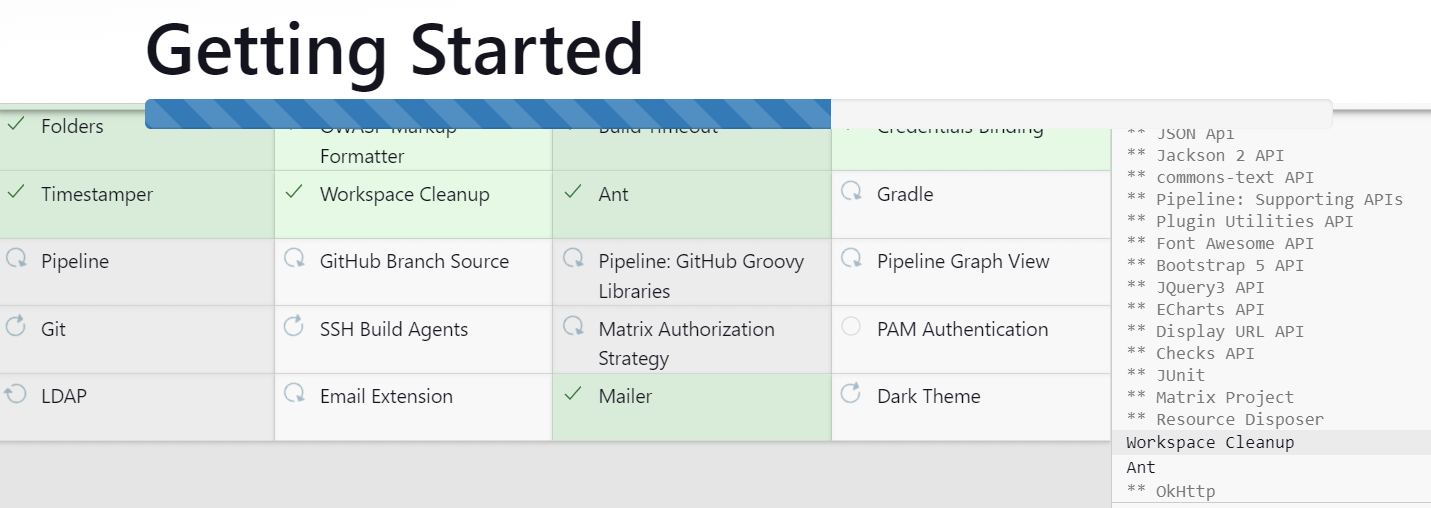
**3. Jenkins Configuration**

Plugins Installed:

* **Git Plugin**: For interacting with Git repositories.
* **Pipeline Plugin**: To support Jenkins pipeline jobs.
* **Credentials Binding Plugin**: For securely managing AWS credentials.

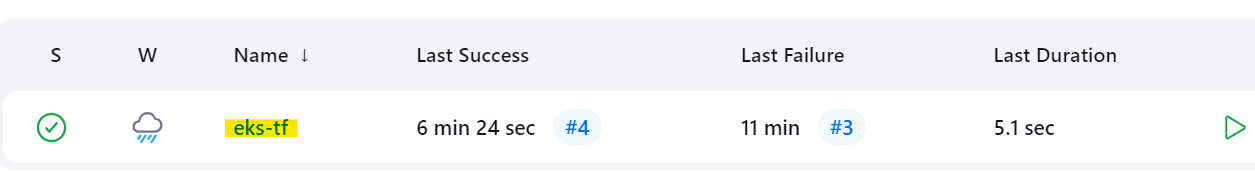
Secrets Configuration for best practices:

* **AWS\_ACCESS\_KEY\_ID**: Stored as a Jenkins secret.
* **AWS\_SECRET\_ACCESS\_KEY**: Stored as a Jenkins secret.





1. **Jenkins Pipeline Configuration**



**Pipeline Script:**

The pipeline script automates Terraform operations to manage the EKS cluster.

pipeline {

agent any

environment {

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

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

AWS\_DEFAULT\_REGION = 'us-east-1'

}

stages{

stage('Checkout SCM'){

steps{

script{

checkout scmGit(branches: [[name: '\*/main']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/yaswanth000/terraform.git']])

}

}

}

stage('Initializing Terraform'){

steps{

script{

dir('eks-module'){

sh 'terraform init'

}

}

}

}

stage('Validating Terraform'){

steps{

script{

dir('eks-module'){

sh 'terraform validate'

}

}

}

}

stage('Previewing the infrastructure'){

steps{

script{

dir('eks-module'){

sh 'terraform plan'

}

input(message: "Approve?", ok: "proceed")

}

}

}

stage('Create/Destroy an EKS cluster'){

steps{

script{

dir('eks-module'){

sh 'terraform $action --auto-approve'

}

}

}

}

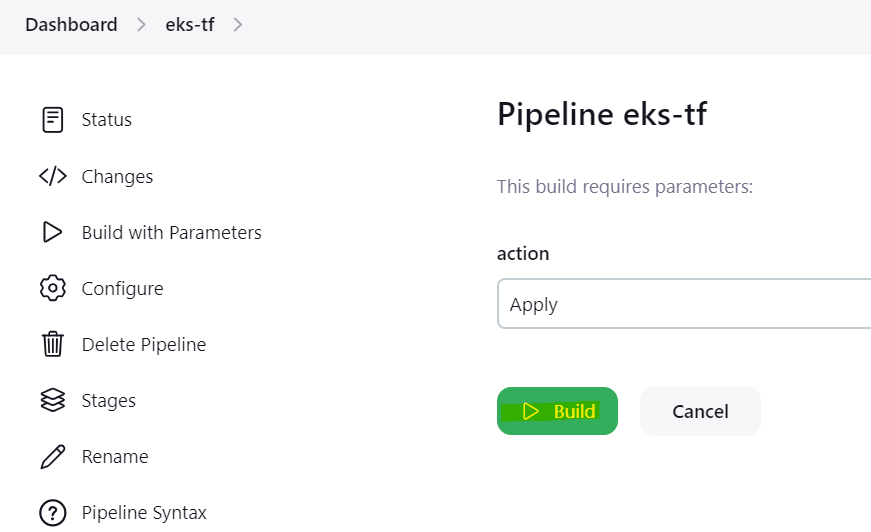
}

}

1. Execution

To execute the Jenkins pipeline:

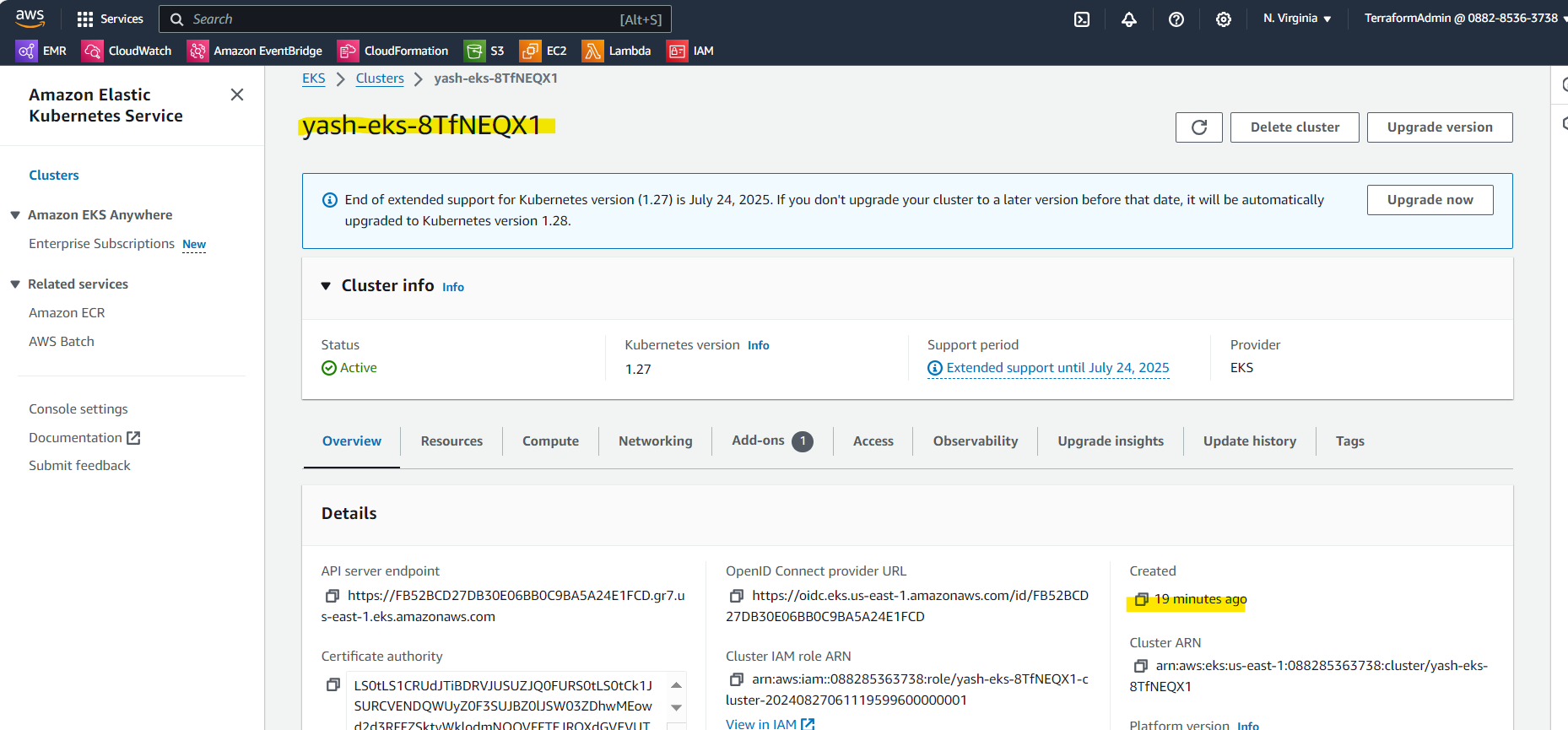
1. Trigger the Jenkins job.
2. Choose the action (apply or destroy) when prompted.
3. The pipeline will:
   * Check out the latest code from the GitHub repository.
   * Initialize and validate Terraform configurations.
   * Preview the infrastructure changes.
   * Apply or destroy the EKS cluster based on the selected action.



Verify the build logs and check the status of the build project

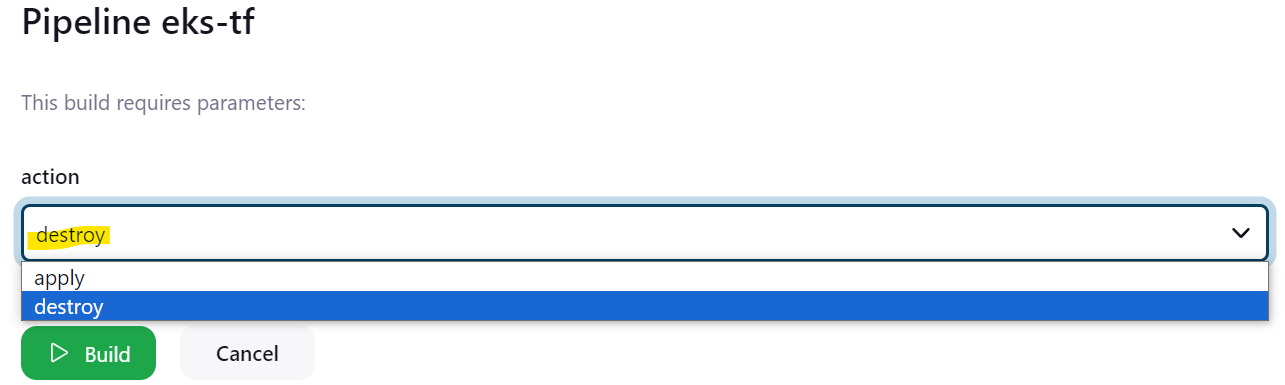


Verification : After build completed connect to aws management console check the resources are created or not

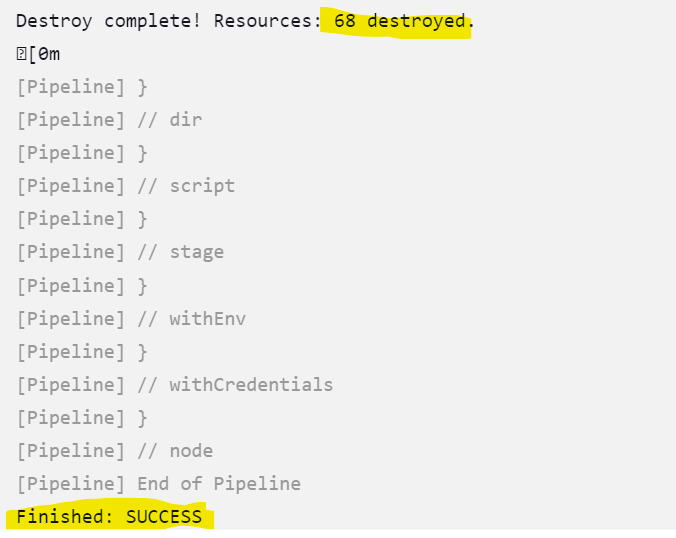


My EKS cluster has been created successfully

1. To destroy the cluster start build and change the parameter



Resources has been successfully destroyed



**Best Practices**

* **Security**: Ensure sensitive data such as AWS credentials are managed securely using Jenkins credentials.
* **Validation**: Regularly review and test the pipeline to ensure it handles various scenarios effectively.
* **Error Handling**: Add error handling and notifications to manage and alert on any issues during pipeline execution.