Project Documentation: CI/CD Pipeline Using Jenkins and Terraform on AWS

---

1. Project Overview

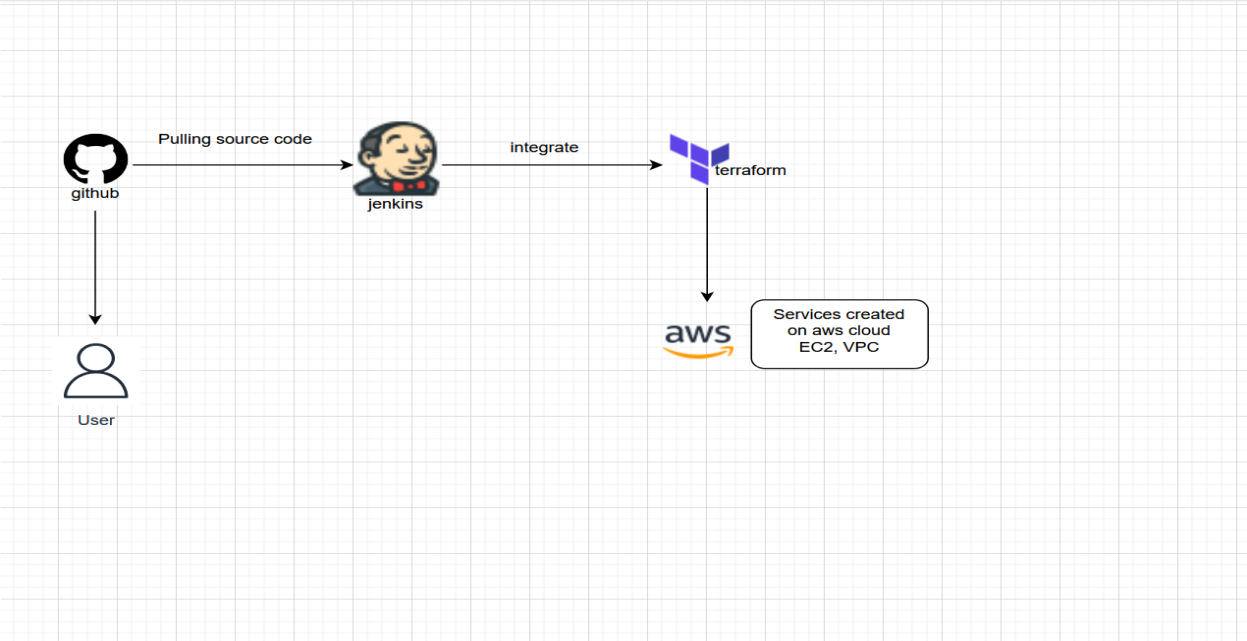
This project demonstrates how to automate the provisioning of AWS infrastructure using **Jenkins** and **Terraform** in a CI/CD pipeline.
The goal is to integrate Infrastructure as Code (IaC) into a continuous delivery workflow, allowing infrastructure to be created, validated, and managed automatically from a Jenkins pipeline using Terraform and AWS services.

Key Components:

- **Jenkins** → Automation server for CI/CD.
- **Terraform** → Infrastructure as Code tool for AWS resource provisioning.
- **AWS** → Cloud provider hosting the infrastructure (EC2, VPC, Subnet, S3).
- **S3 Bucket** → Used as Terraform backend for remote state management.
- **GitHub** → Version control system for storing Terraform and Jenkins files.

---

2. Architecture Diagram



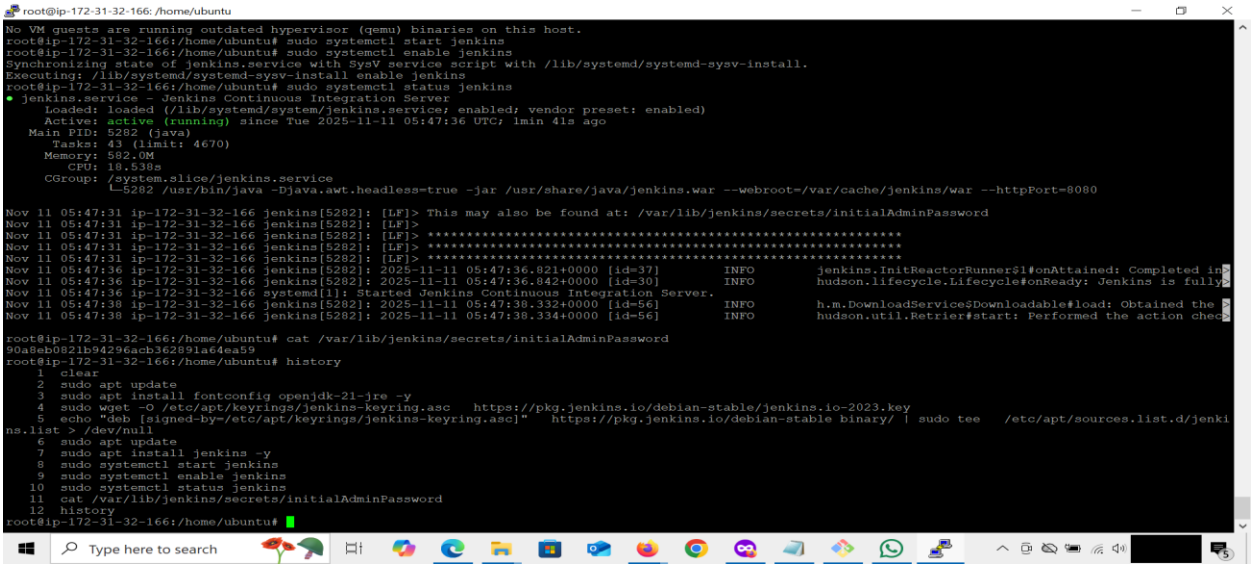
3. Prerequisites

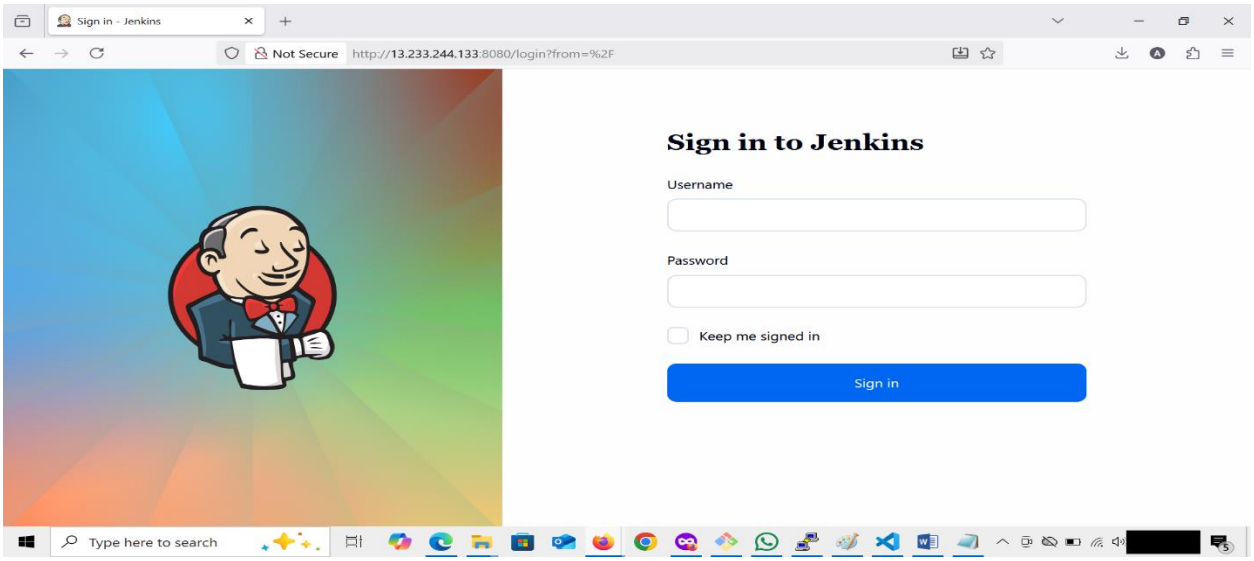
Before starting the setup, ensure the following requirements are met:

- **AWS Account**
- **IAM User** with permissions:
  - AdministratorAccess
  - AmazonS3FullAccess
- **S3 Bucket** (for Terraform state file) → infra-terraform-state-aniket-2025
- **EC2 Instance** (Ubuntu 22.04) for Jenkins and Terraform
- **Key Pair** → account-a (used by Terraform for SSH access)
- **Public GitHub Repository** → jenkins-terraform-aws-cicd

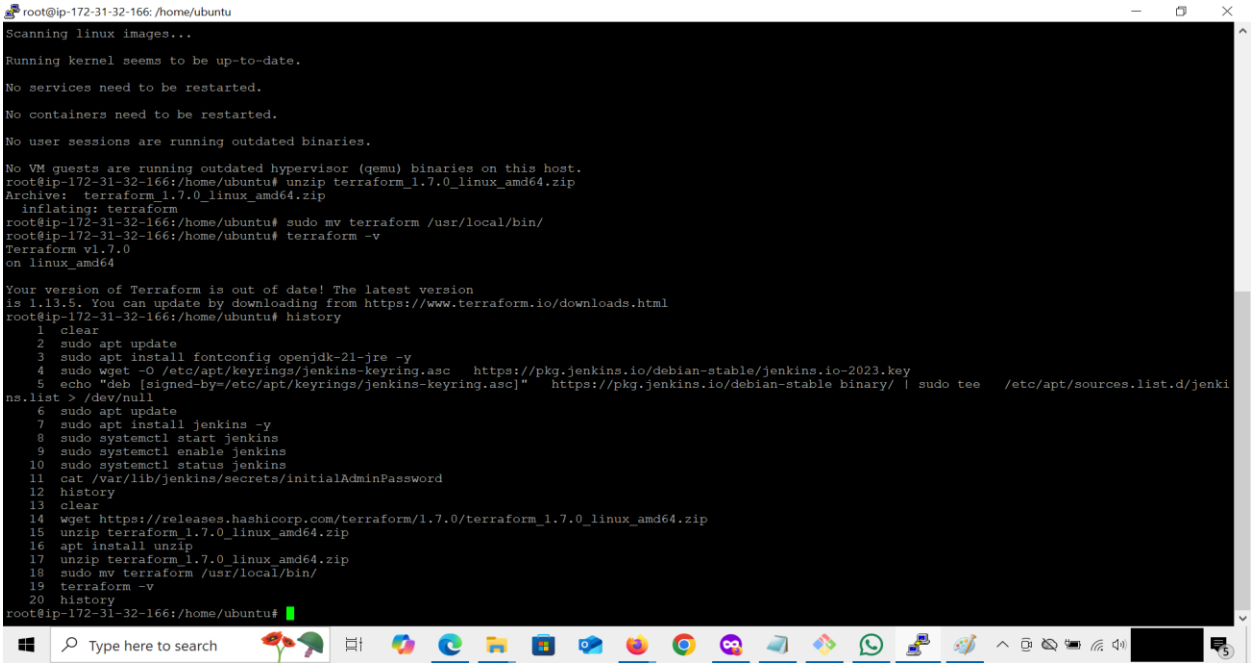
4. Environment Setup

Jenkins Installation on EC2

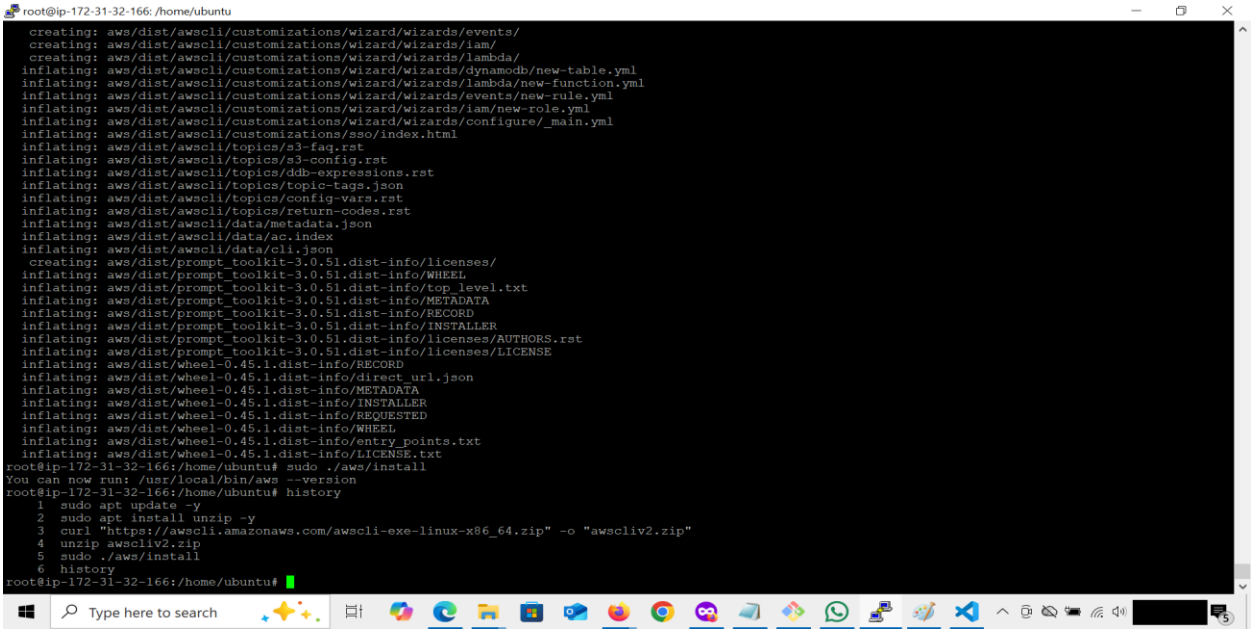




Terraform Installation on Jenkins EC2



AWS CLI Installation



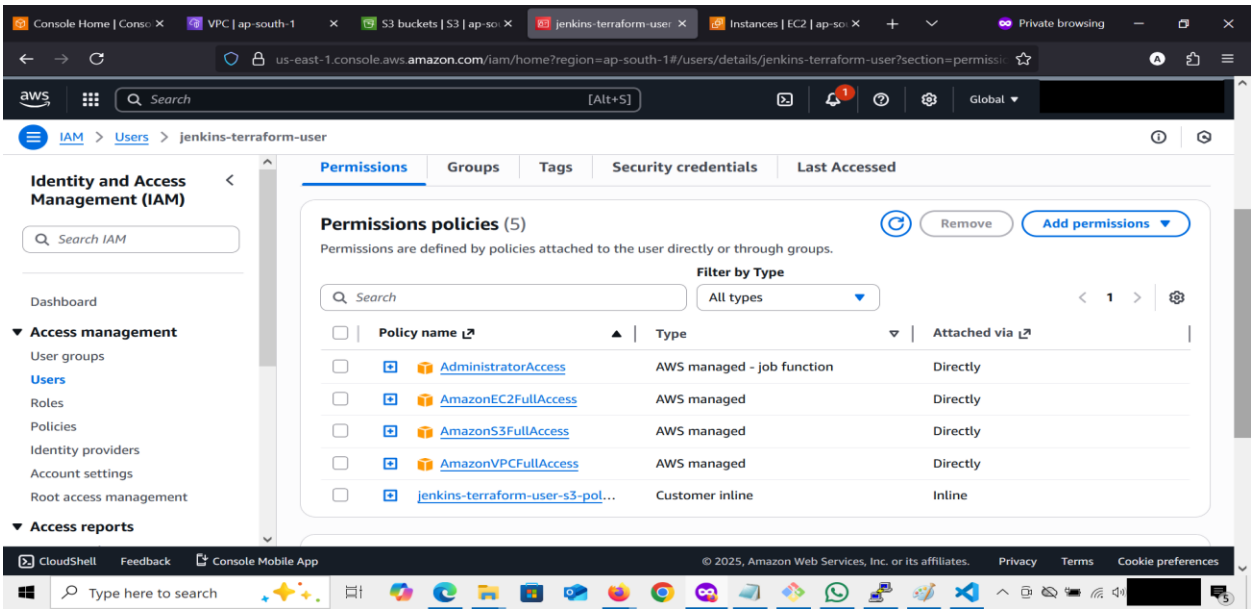
5. AWS Configuration

IAM User and Permissions

Created a dedicated IAM user jenkins-terraform-user with these policies:

- AdministratorAccess
- AmazonS3FullAccess

Used for Jenkins → Terraform AWS authentication.



S3 Bucket for Terraform State

What is a Terraform State File?

When Terraform creates or updates infrastructure, it keeps track of all deployed resources in a **state file** named terraform.tfstate.

This file acts as a **single source of truth** for Terraform, it records details about every AWS resource created (like EC2 IDs, subnet IDs, VPC IDs, etc.), allowing Terraform to understand the current infrastructure state and make only necessary changes in the future.

Without this file, Terraform would have no idea what's already deployed, leading to duplicated or mismatched resources.

Why Use a Remote Backend (S3)?

By default, Terraform stores its state file locally (on the Jenkins server).

However, this is not ideal for collaborative or CI/CD environments.

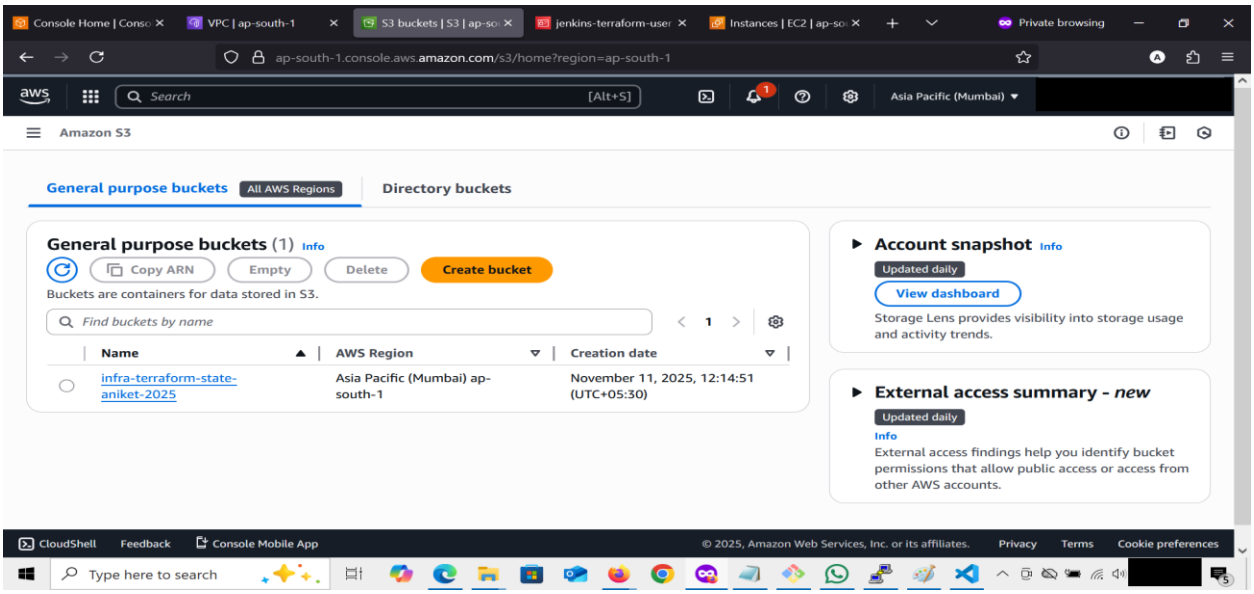
That's why we use an **AWS S3 bucket** as a **remote backend** to

S3 bucket name: **infra-terraform-state-aniket-2025**

Region: ap-south-1

Used to store Terraform state files remotely for consistency and collaboration.

S3 bucket view in AWS Console



6. Jenkins Configuration

Required Jenkins Plugins

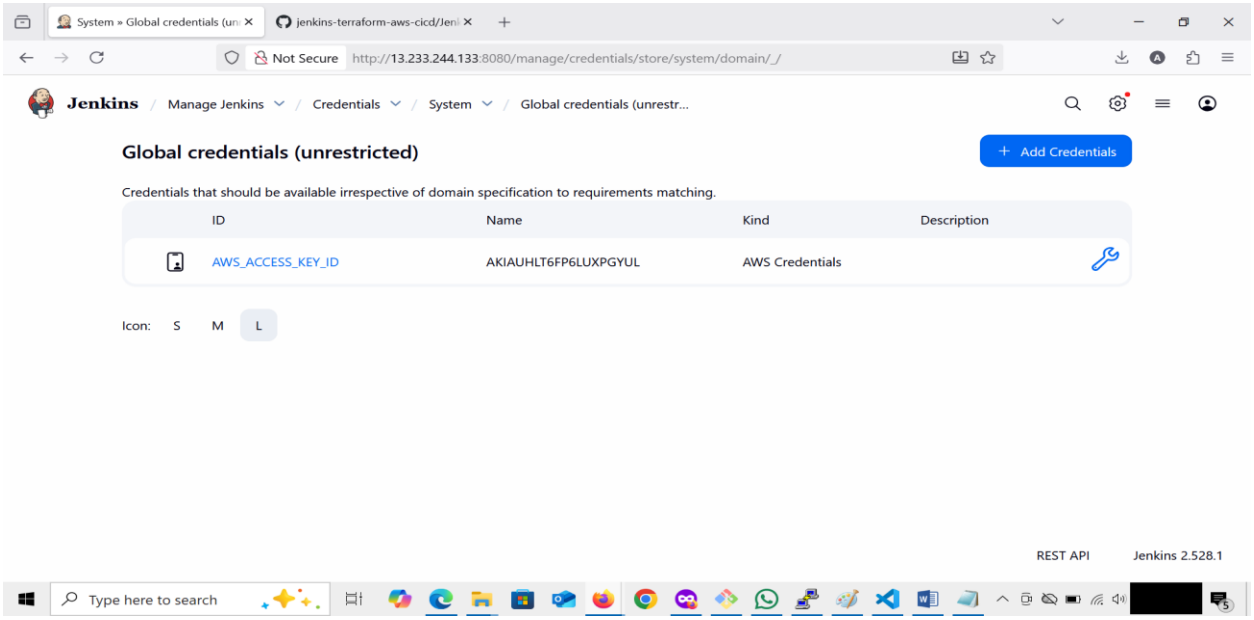
Installed via **Manage Jenkins → Manage Plugins → Available:**

- **AWS Credentials Plugin**
- **Git Plugin**
- **Pipeline Plugin**
- **Terraform Plugin (optional)**

Jenkins Credentials

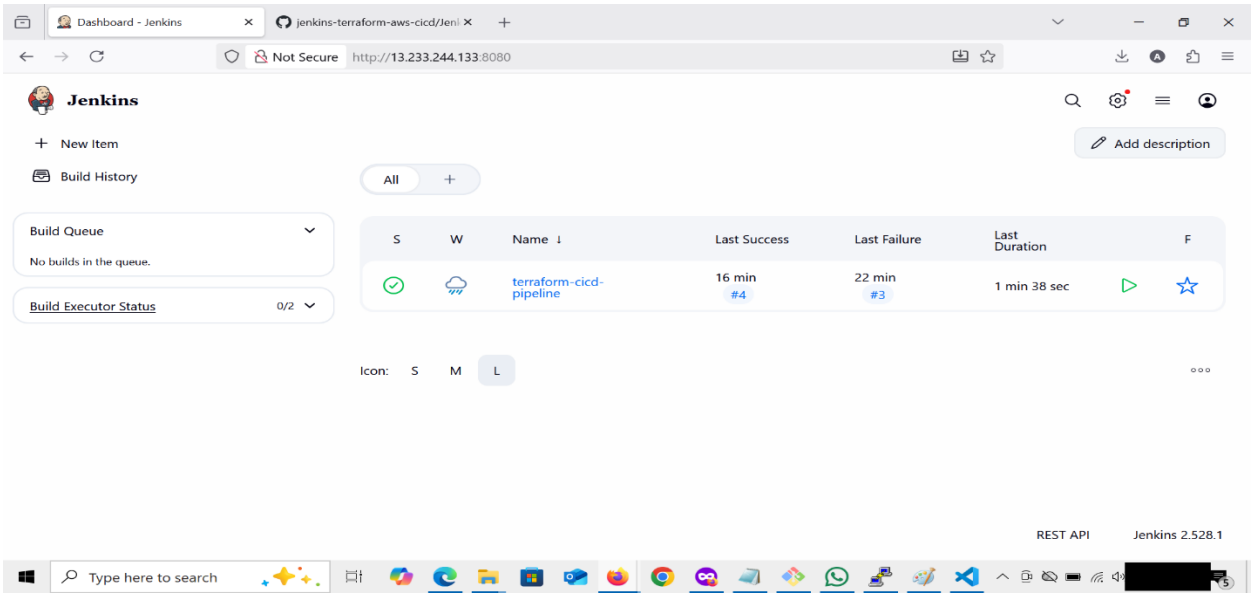
Stored AWS keys securely:

- Go to **Manage Jenkins → Manage Credentials**
- Add New:
  - **Kind:** AWS Credentials
  - **ID:** AWS_ACCESS_KEY_ID
  - **Access Key / Secret Key:** from IAM user



Jenkins Pipeline Configuration

1. Go to **Jenkins → New Item → Pipeline**
2. Set:
    - **Definition:** Pipeline script from SCM
    - **SCM:** Git
    - **Repository URL:** https://github.com/anikettalwekar/jenkins-terraform-aws-cicd.git
    - **Branch:** main



7. Terraform Configuration Files (Stored in GitHub)

- main.tf: Defines AWS VPC, Subnet, Security Group, and EC2 instance.
- variables.tf: Defines variable for key_name = "account-a".
- outputs.tf: Outputs instance details (ID, public IP, VPC ID).
- provider.tf: Configures AWS provider region and backend (S3).

File	Purpose
main.tf	Core infrastructure (VPC, subnet, EC2, SG)
variables.tf	Dynamic input parameters (key_name, region, etc.)
outputs.tf	Returns useful resource details
provider.tf	Connects Terraform to AWS region and backend

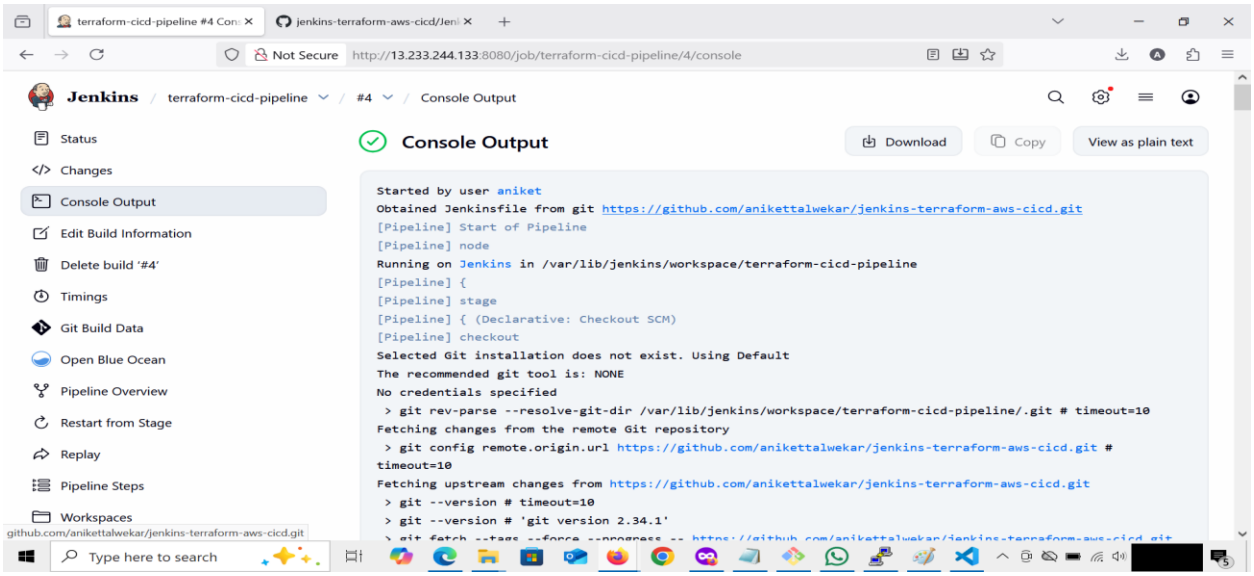
8. Jenkinsfile Explanation

Jenkinsfile defines all pipeline stages:

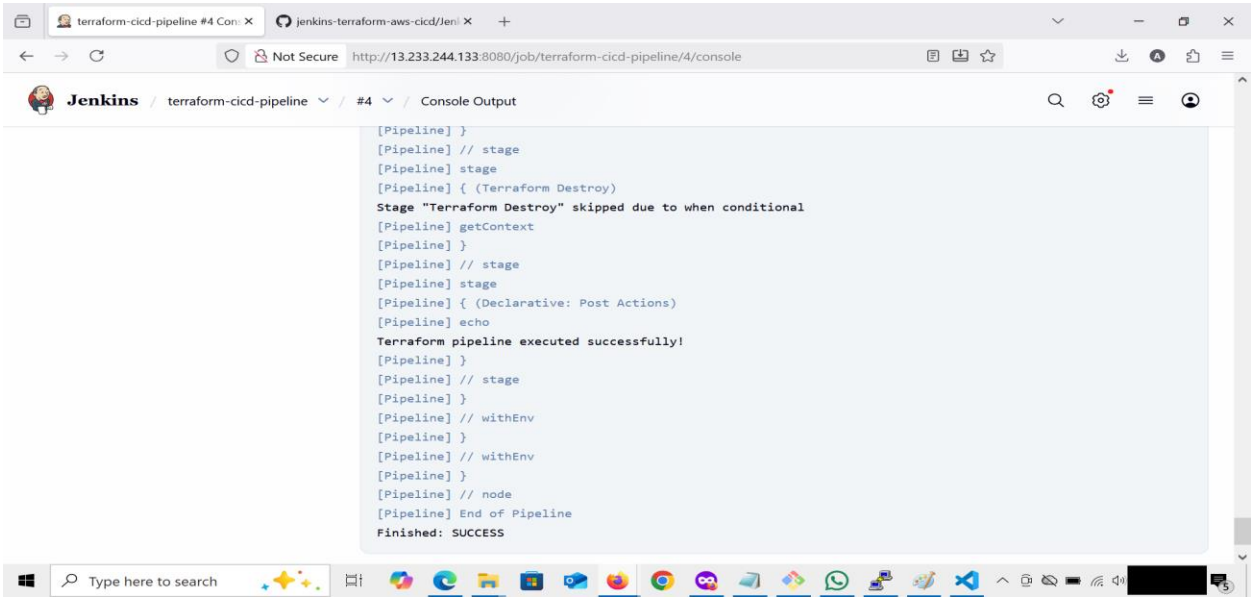
This provides **automated, repeatable infrastructure delivery**.

Stage	Purpose
Checkout Code	Pulls latest Terraform files from GitHub
Verify AWS Credentials	Validates IAM access using AWS CLI
Terraform Init	Initializes backend and providers
Terraform Validate	Checks syntax and configuration
Terraform Plan	Previews resource creation
Terraform Apply	Provisions AWS resources
Terraform Destroy (Optional)	Removes infrastructure

Jenkins console (pipeline start)



Jenkins console (pipeline end)



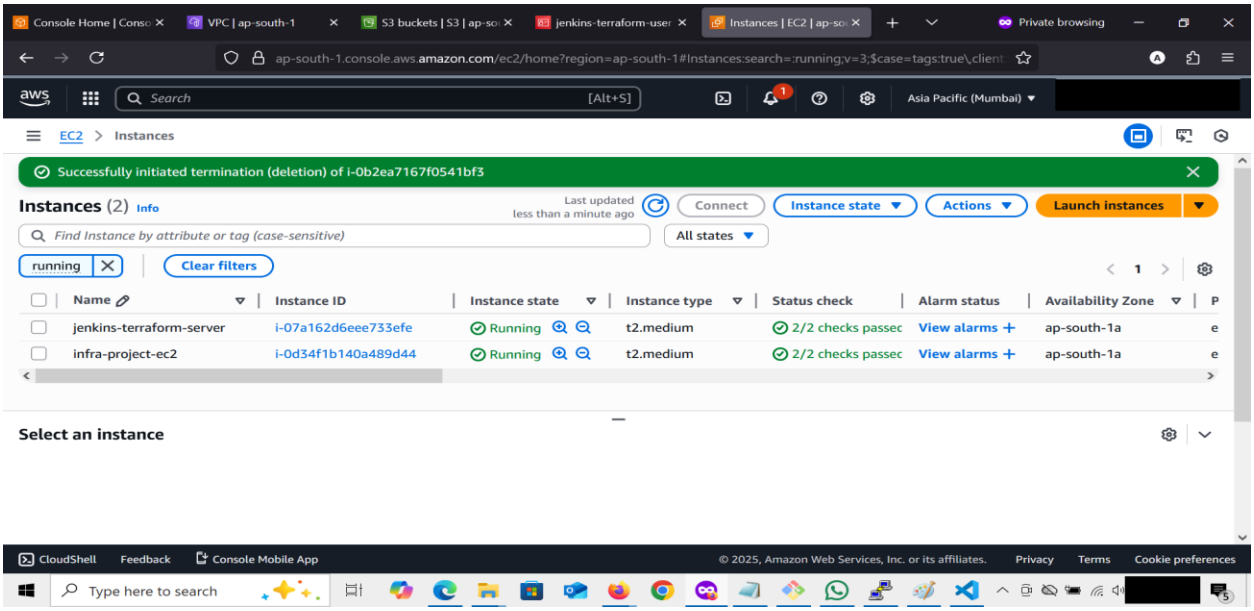
9. AWS Infrastructure Verification

After the successful Jenkins pipeline execution, Terraform automatically provisioned AWS resources as defined in the Terraform configuration files (main.tf, variables.tf, and provider.tf).

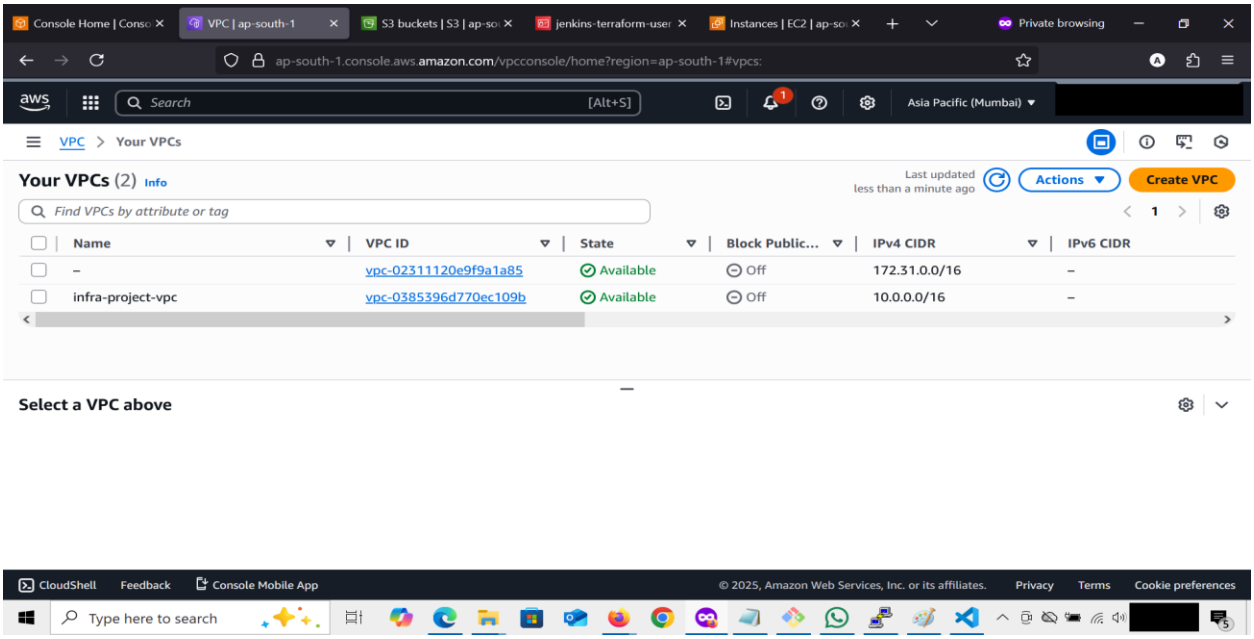
Resources Created by Terraform

Resource Type	Name	Description
VPC	infra-project-vpc	Custom Virtual Private Cloud created to host network resources
Subnet	infra-project-subnet	Public subnet within the VPC for EC2 instance deployment
Security Group	infra-project-sg	Allows inbound SSH (22) and HTTP (8080) traffic
EC2 Instance	infra-project-ec2	EC2 instance automatically created by Terraform through Jenkins pipeline

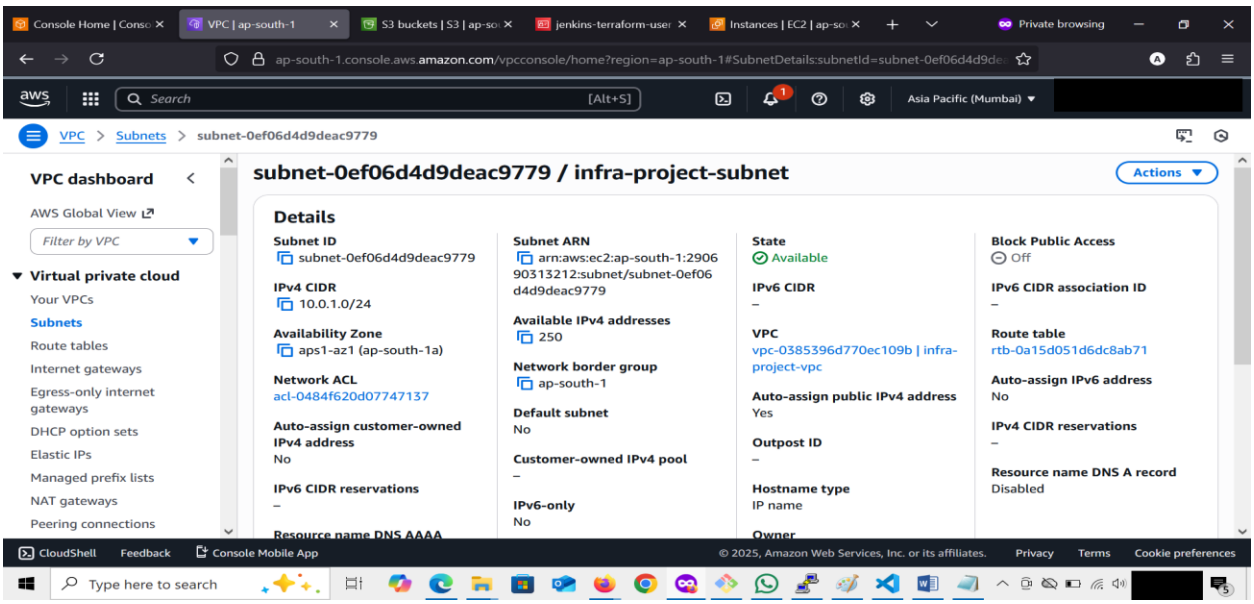
EC2


Successfully initiated termination (deletion) of i-0b2ea7167f0541bf3
Instances (2) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
running
Clear filters
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
jenkins-terraform-server
i-07a162d6eee733efe
Running
t2.medium
2/2 checks passed
View alarms
ap-south-1a
infra-project-ec2
i-0d34f1b140a489d44
Running
t2.medium
2/2 checks passed
View alarms
ap-south-1a
Select an instance

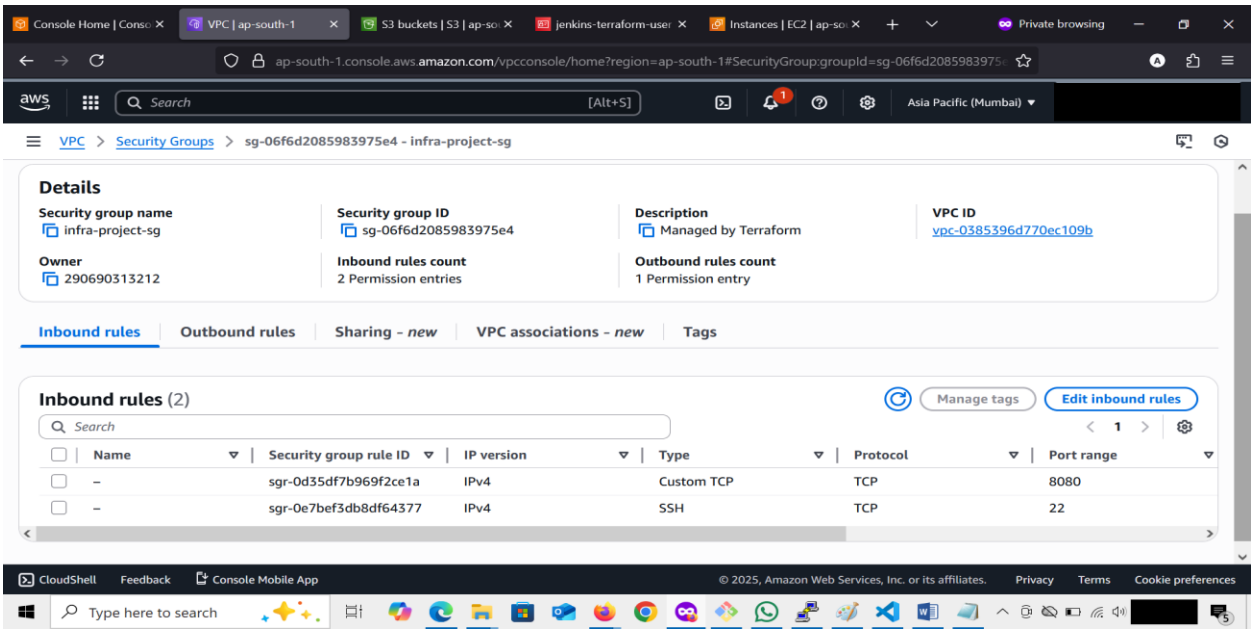
VPC


Your VPCs (2) Info
Last updated less than a minute ago
Actions
Create VPC
Find VPCs by attribute or tag
Name
VPC ID
State
Block Public...
IPv4 CIDR
IPv6 CIDR
–
vpc-02311120e9f9a1a85
Available
Off
172.31.0.0/16
infra-project-vpc
vpc-0385396d770ec109b
Available
Off
10.0.0.0/16
Select a VPC above

Subnet



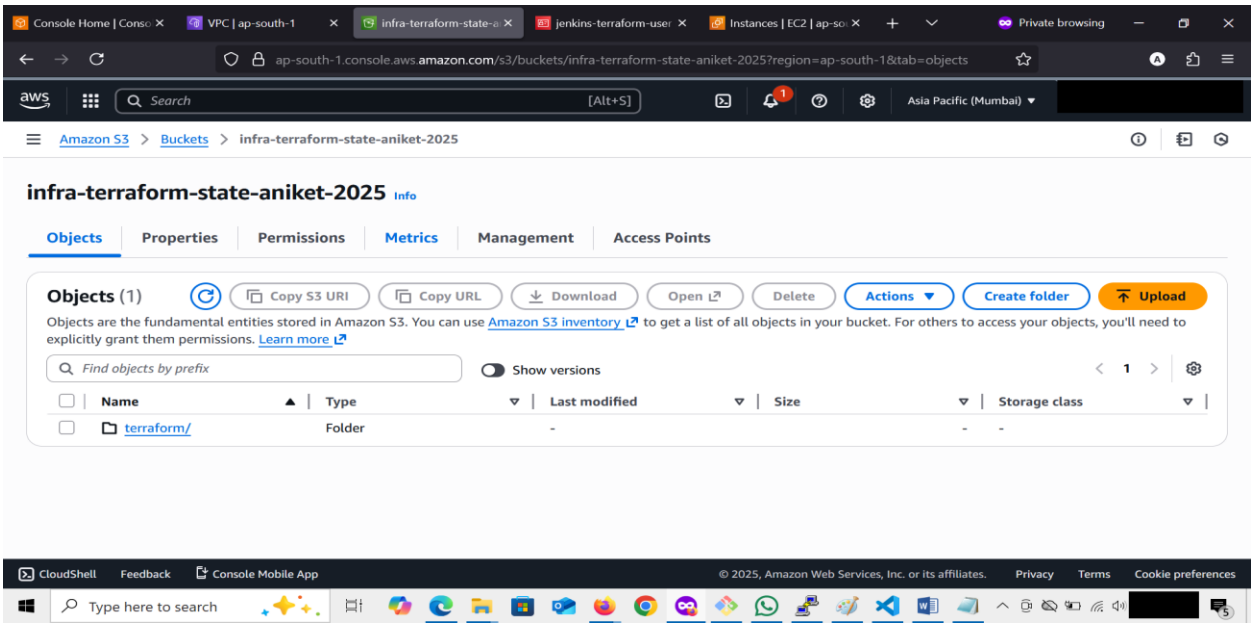
Security Group



S3 Bucket: infra-terraform-state-aniket-2025

Key: terraform/state.tfstate

Region: ap-south-1



his ensures consistent state even if the Jenkins instance is recreated.

10. CI/CD Execution Workflow

1. Developer pushes Terraform code to GitHub.
2. Jenkins pipeline is triggered.
3. Jenkins clones repo → verifies AWS creds → initializes Terraform.
4. Terraform provisions EC2, VPC, subnet, and SG on AWS.
5. Jenkins shows the build result and stores logs.
6. State file is uploaded to S3.
7. Destroy can be triggered for cleanup.

11. Destroy Infrastructure

Terraform destroy stage (optional) removes all created resources automatically:
terraform destroy -auto-approve

12. Conclusion

This project demonstrates a complete CI/CD workflow integrating **Jenkins**, **Terraform**, **GitHub**, and **AWS**.
The system automates infrastructure deployment and management using Infrastructure as Code (IaC).

Key Achievements:

- Jenkins automates Terraform workflows.
- Infrastructure is stored, versioned, and auditable.
- Remote state management with S3 ensures team consistency.
- Scalable foundation for multi-environment (Dev/Stage/Prod) CI/CD pipelines.