AWS – Infrastructure as Code (IaC)

Git • Jenkins • CloudFormation

Akhil Rao Bembera

19 July 2025

Overview:

This project demonstrates the use of Infrastructure as Code (IaC) using AWS CloudFormation to automate cloud resource provisioning. Git is utilized for version control, while Jenkins handles the CI/CD pipeline automation. The pipeline fetches YAML templates from GitHub and deploys AWS resources such as S3 buckets and their corresponding access policies. AWS credentials are securely managed through the Jenkins credentials plugin, ensuring secure and seamless deployments. CloudFormation enables consistent and repeatable infrastructure setup. Error handling and post-build steps are incorporated to enhance reliability. Overall, the project showcases best practices in DevOps, Site Reliability Engineering (SRE), and cloud automation.

Prerequisites:

1. Install and run Jenkins locally or on a server
2. Set up AWS and GitHub accounts
3. Install and configure the AWS CLI with credentials
4. Install Git and clone the project repository
5. Allocate some dedicated time and curiosity

Jenkins Configuration

After installation, start the service

Mac: `brew services start jenkins-lts`

Linux: `systemctl start jenkins`

Now, your Jenkins should be exposed on your browser. After logging in to Jenkins page:

Install Required Plugins

- Git plugin
- Pipeline
- AWS CloudFormation plugin (optional)
- Blue Ocean (optional)
- GitHub integration

Credentials

Go to Dashboard -> Manage Jenkins -> Credentials -> add credentials

- Type: Username and Password
  - ID: aws-cred
  - Username: your AWS Access Key
  - Password: your AWS Secret Key

Create a Pipeline Job in Jenkins

- Set Git URL
- Choose: Poll SCM (H/2 * * * *) – runs every 2 mins if changes detected

Pipeline-groovy

```
    pipeline {
        agent any

        environment {
            AWS_REGION = 'ap-south-1'
            STACK_NAME = 'free-s3-stack'
            TEMPLATE_FILE = 's3-bucket-template.yaml'
        }

        stages {
            stage('Checkout Code') {
                steps {
                    git branch: 'main', url: 'https://github.com/akhilrao199/
cloudformationProject1.git'
                }
            }

            stage('Deploy CloudFormation Stack') {
                steps {
                    withCredentials([usernamePassword(credentialsId: 'aws-cred',
usernameVariable: 'AWS_ACCESS_KEY_ID', passwordVariable: 'AWS_SECRET_ACCESS_KEY')]) {
                        sh '''
                            export AWS_ACCESS_KEY_ID=$AWS_ACCESS_KEY_ID
                            export AWS_SECRET_ACCESS_KEY=$AWS_SECRET_ACCESS_KEY

                            aws cloudformation deploy \
                              --template-file $TEMPLATE_FILE \
                              --stack-name $STACK_NAME \
                              --region $AWS_REGION \
                              --capabilities CAPABILITY_NAMED_IAM

                            echo "CloudFormation stack '$STACK_NAME' deployed
successfully."
                        '''
                    }
                }
            }
        }

        post {
            failure {
                echo "Deployment failed. Please check the CloudFormation template or
credentials."
            }
        }
    }
```

Git

- Clone Git repo

```
git clone https://github.com/akhilrao199/
cloudformationProject1.git
```

- Commit changes - update yaml file with latest cloud formation template

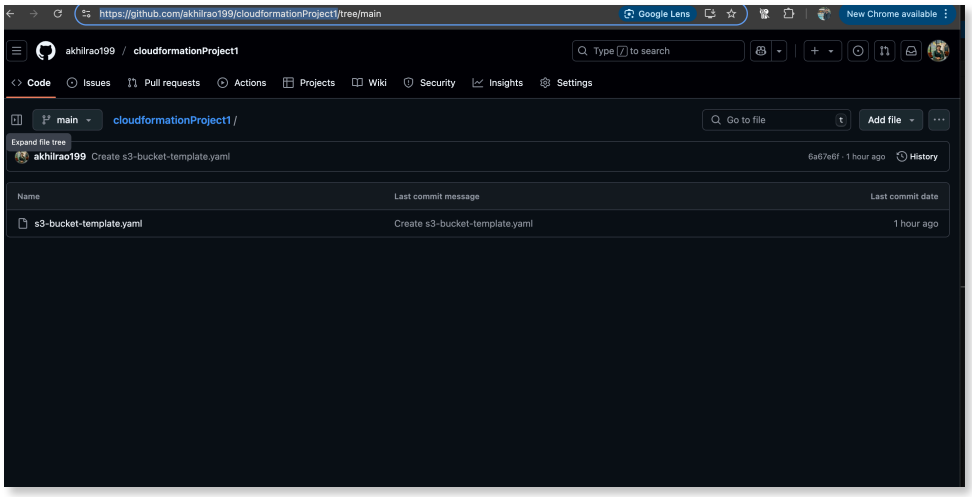
```
vi s3-bucket-template.yaml
```

- Add

```
git add s3-bucket-template.yaml
```

- Commit

```
git commit -m "Updated template - added second S3 bucket"
```




```
akhilrao@Akhils-MacBook-Air cloudformationProject1 % vi s3-bucket-template.yaml
akhilrao@Akhils-MacBook-Air cloudformationProject1 % git add s3-bucket-template.yaml
akhilrao@Akhils-MacBook-Air cloudformationProject1 % git commit -m "re-visited- addition of second bucket"
[main 723279c] re-visited- addition of second bucket
 1 file changed, 33 insertions(+), 33 deletions(-)
akhilrao@Akhils-MacBook-Air cloudformationProject1 % git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 647 bytes | 647.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/akhilrao199/cloudformationProject1.git
   4dab99e..723279c  main -> main
```

CloudFormation templates

1. Template to create a S3 bucket

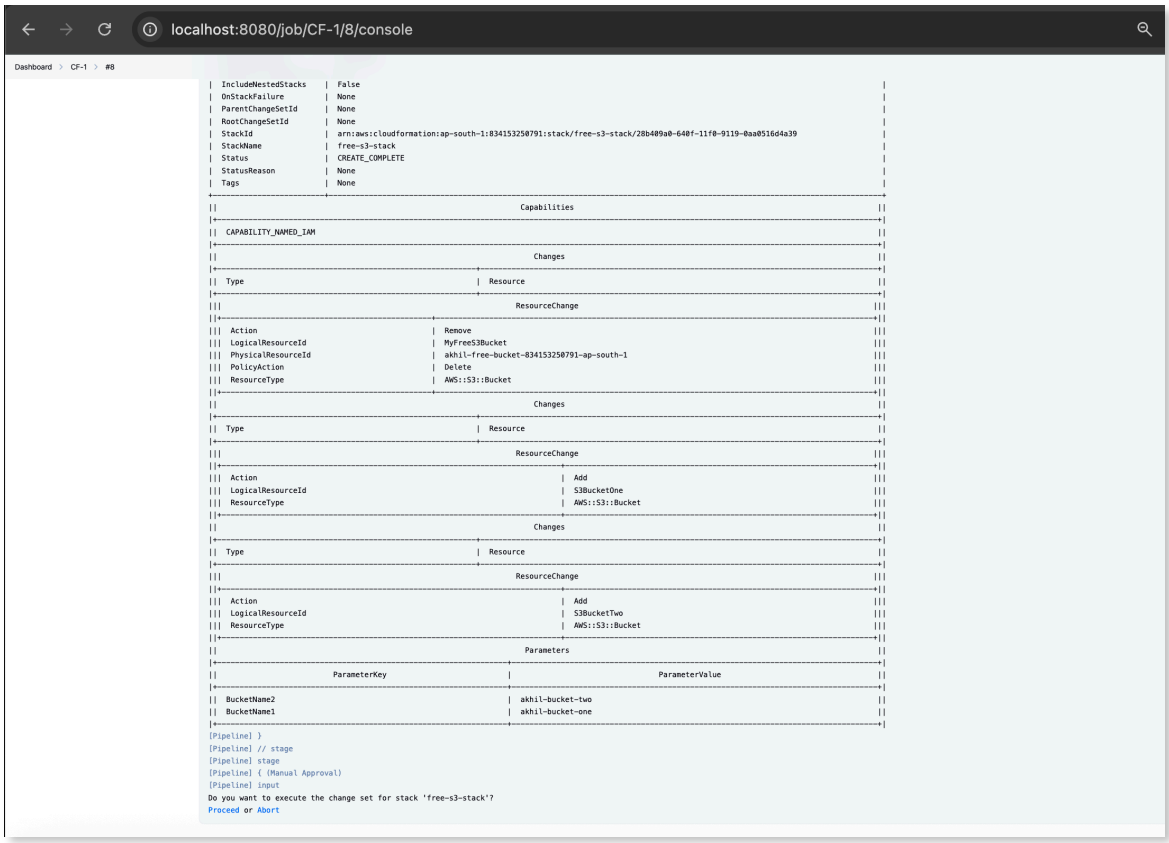
```
AWSTemplateFormatVersion: '2010-09-09'
Description: CloudFormation template to create a versioned S3
bucket

Resources:
  FreeS3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: akhil-free-bucket-834153250791-ap-south-1
      VersioningConfiguration:
        Status: Enabled

Outputs:
  BucketName:
    Value: !Ref FreeS3Bucket
    Description: Name of the created S3 bucket
```

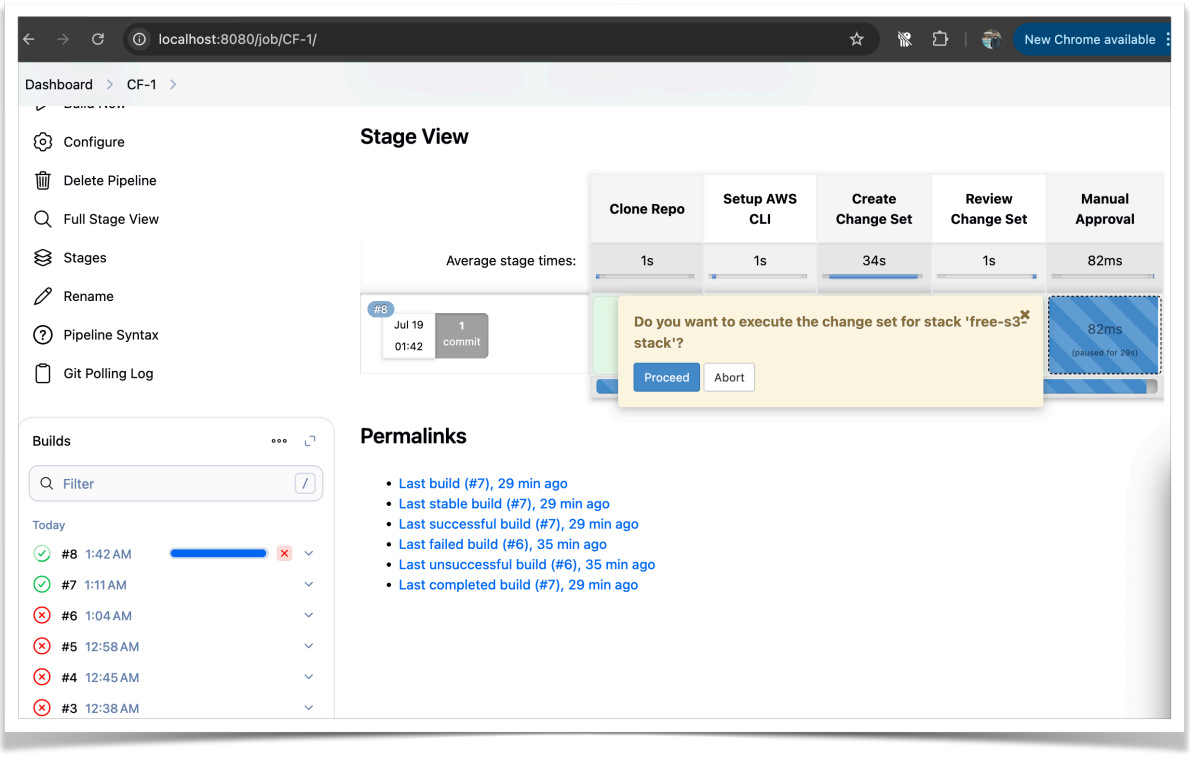
2. Template to add S3AccessPolicy to existing S3 bucket

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Create two S3 buckets with access policies

Resources:
  FirstBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: akhil-free-bucket-834153250791-ap-south-1

  SecondBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: akhil-bucket-two

  FirstBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref FirstBucket
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal: "*"
            Action:
              - s3:GetObject
              - s3:PutObject
              - s3:DeleteObject
            Resource: !Sub "${FirstBucket.Arn}/*"

  SecondBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref SecondBucket
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal: "*"
            Action:
              - s3:GetObject
              - s3:PutObject
              - s3:DeleteObject
            Resource: !Sub "${SecondBucket.Arn}/*"
```

Workflow

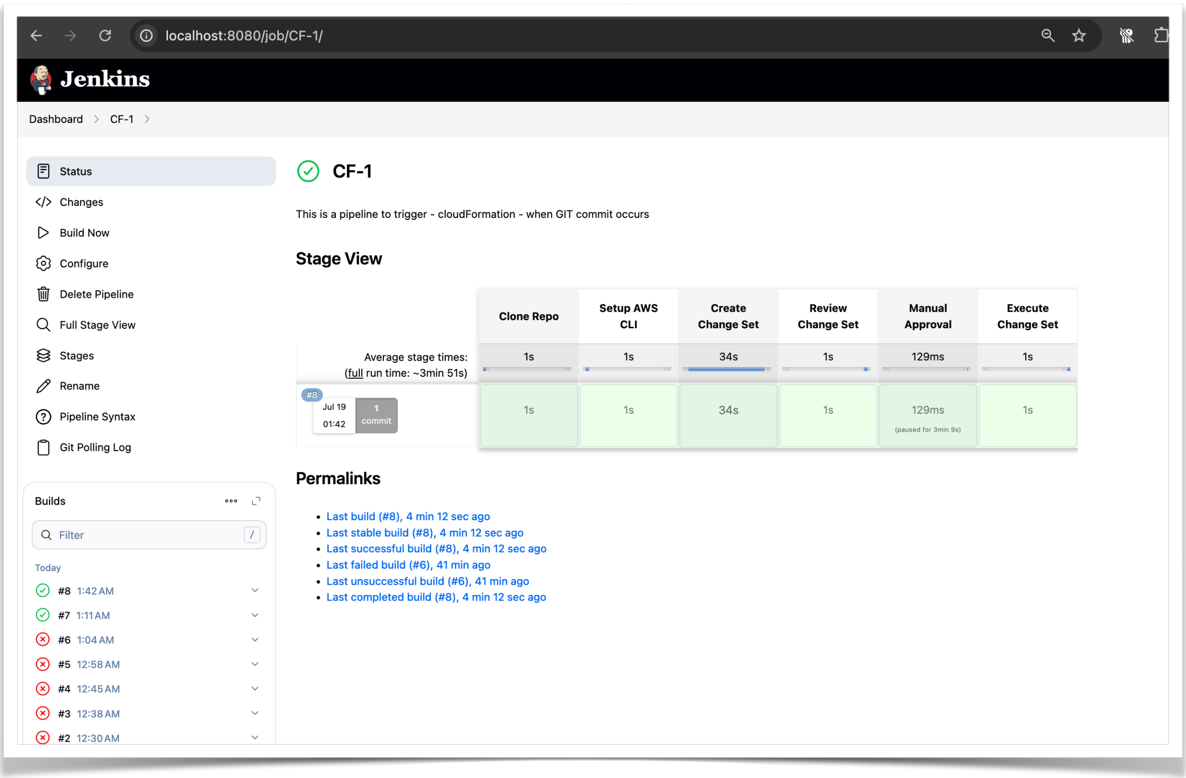
1. Commit YAML (s3-bucket-template.yaml) to GitHub.
2. Jenkins polls repo every 2 minutes.
3. Change set is created.
4. Jenkins waits for manual approval.

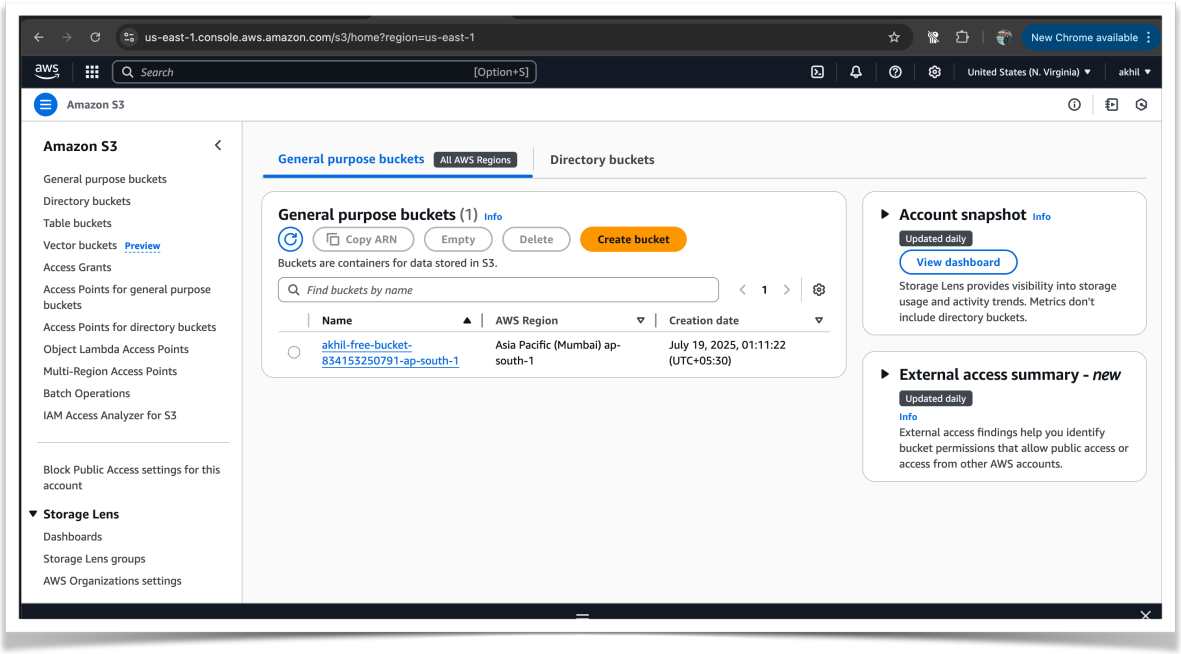




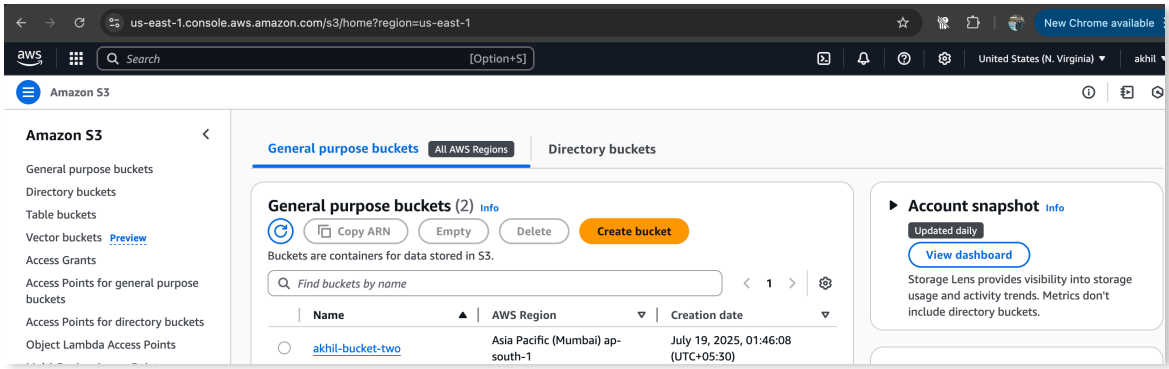
5. On approval → Change set is executed → Infra provisioned.

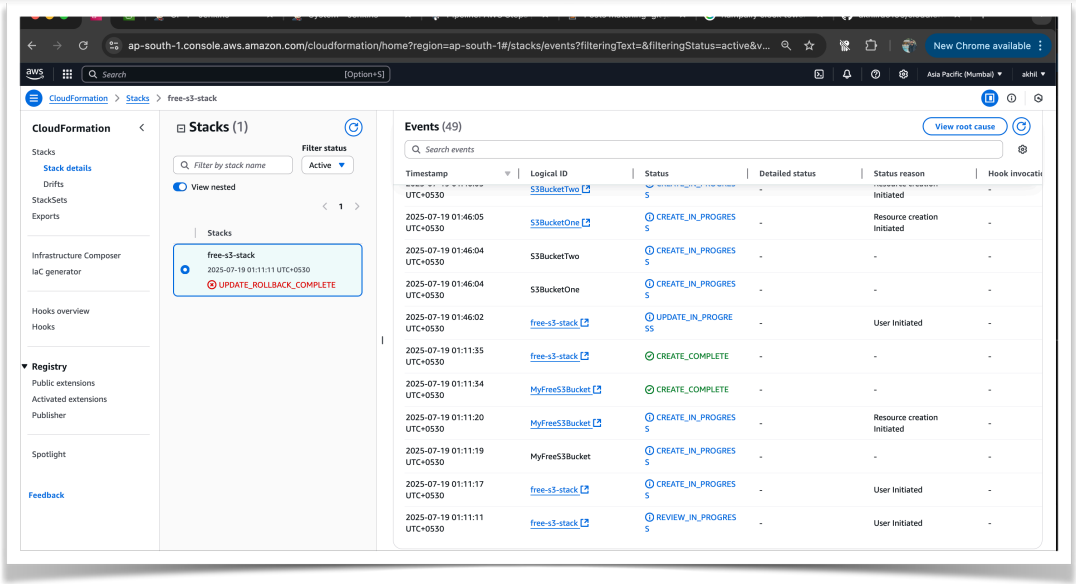
```
|| BucketName1                                         | akhil-bucket-one
|+-----------------------------------------------------+------------------------------------------
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Manual Approval)
[Pipeline] input
Do you want to execute the change set for stack 'free-s3-stack'?
Proceed or Abort
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Execute Change Set)
[Pipeline] sh
+ echo '⚙ Executing change set...'
⚙ Executing change set...
+ /opt/homebrew/bin/aws cloudformation execute-change-set --stack-name free-s3-stack --change-set-name manual-approval-changeset
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```





6. Next commit → same workflow continues (modular change sets).





7. Last step → manually delete stack to remove all infra.

```
$~ aws cloudformation list-stacks --stack-status-filter
CREATE_COMPLETE UPDATE_COMPLETE

$~ aws cloudformation delete-stack --stack-name ar-single-bucket-
stack
```

Learning:

Yes, I did face numerous errors during the course of this project, but each one turned out to be a valuable learning experience. These challenges helped me understand what can go wrong, why it happens, and how to troubleshoot effectively. From Git issues to Jenkins misconfigurations and CloudFormation template errors, every setback deepened my understanding of real-world CI/CD processes. Instead of seeing errors as roadblocks, I began to view them as stepping stones that improved my confidence in automating infrastructure and managing cloud deployments.