

1

Infrastructure Deployment - Jenkins

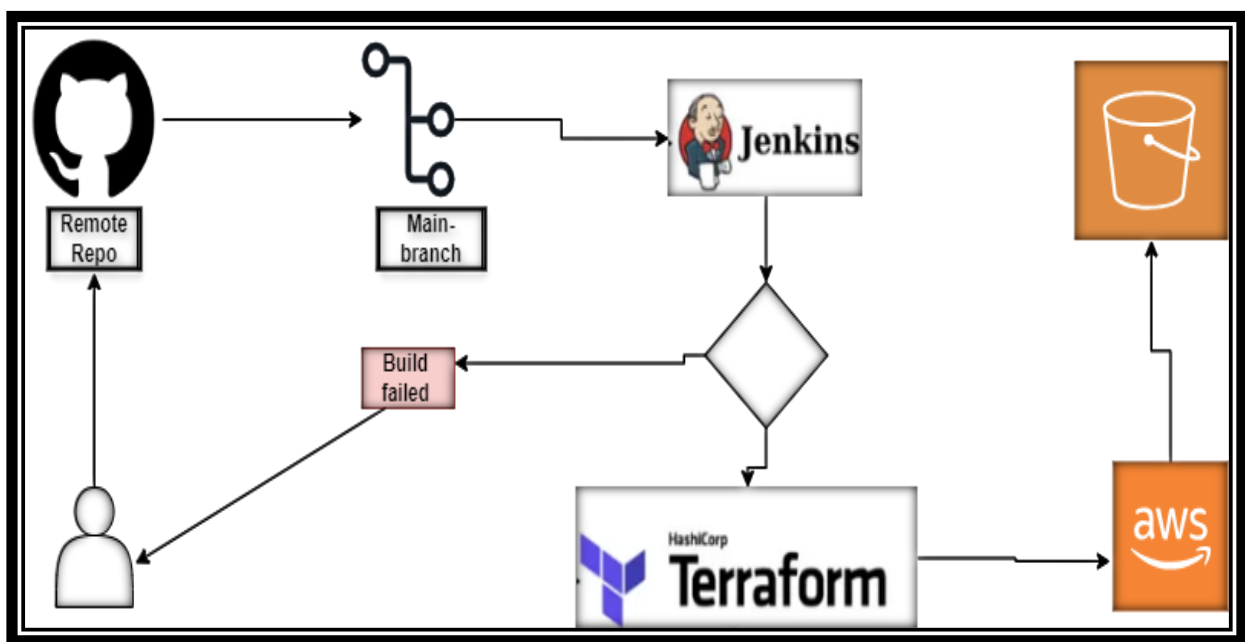
Scenario:

- Automating Simple Infrastructure Deployment with Terraform.

Given Scenario:

- In the given scenario, we have to setup the Jenkins build to perform and automate the simple infrastructure deployment to make the stage of terraform commands automated using the stage Jenkins setup.

Workflow – Flow Chart:



STEP 1: Create the Terraform IAC to make S3 Bucket in AWS.

Provider.tf:

- **Provider.tf:** providers. tf is responsible for managing the lifecycle of a particular set of resources, such as AWS, Azure, or GCP. Here, AWS is configured.

```
provider "aws" {
  region = "us-east-1"
}
```

Infrastructure Deployment - Jenkins

S3.tf:

- This file typically contains the terraform configuration to create the S3 bucket with other necessary items to create the simple S3 bucket.

```
terraform {  
  required_providers {  
    aws = {  
      source  = "hashicorp/aws"  
      version = "~> 5.0"  
    }  
  }  
}  
  
resource "aws_s3_bucket" "application_files" {  
  bucket = var.application_files_bucket_name  
  acl     = "private"  
  
  versioning {  
    enabled = true  
  }  
}  
  
resource "aws_s3_bucket" "static_assets" {  
  bucket = var.static_assets_bucket_name  
  acl     = "private"  
  
  versioning {  
    enabled = true  
  }  
}
```

Variable.tf

- This file is used to define input variables for terraform configuration. Input variables allow you to parameterize infrastructure so that we can reuse the same terraform configuration with different values.

```
variable "application_files_bucket_name" {  
  description = "Name of the application files S3 bucket"  
  type        = string  
}  
  
variable "static_assets_bucket_name" {  
  description = "Name of the static assets S3 bucket"  
  type        = string  
}
```

Infrastructure Deployment - Jenkins

Terraform. Tfvvars:

- This file sets values for the variables defined in variables.tf. It allows us to assign specific values to variables without directly modifying the terraform configuration files.

```
application_files_bucket_name = "application-files-arun-bucket"
static_assets_bucket_name = "static-assets-arun-bucket"
```

Output.tf

- The file is used to define output values that are helpful to interact with after terraforming applies our configuration.

```
output "application_files_bucket_name" {
  description = "Name of the application files S3 bucket"
  value       = aws_s3_bucket.application_files.bucket
}

output "application_files_bucket_arn" {
  description = "ARN of the application files S3 bucket"
  value       = aws_s3_bucket.application_files.arn
}

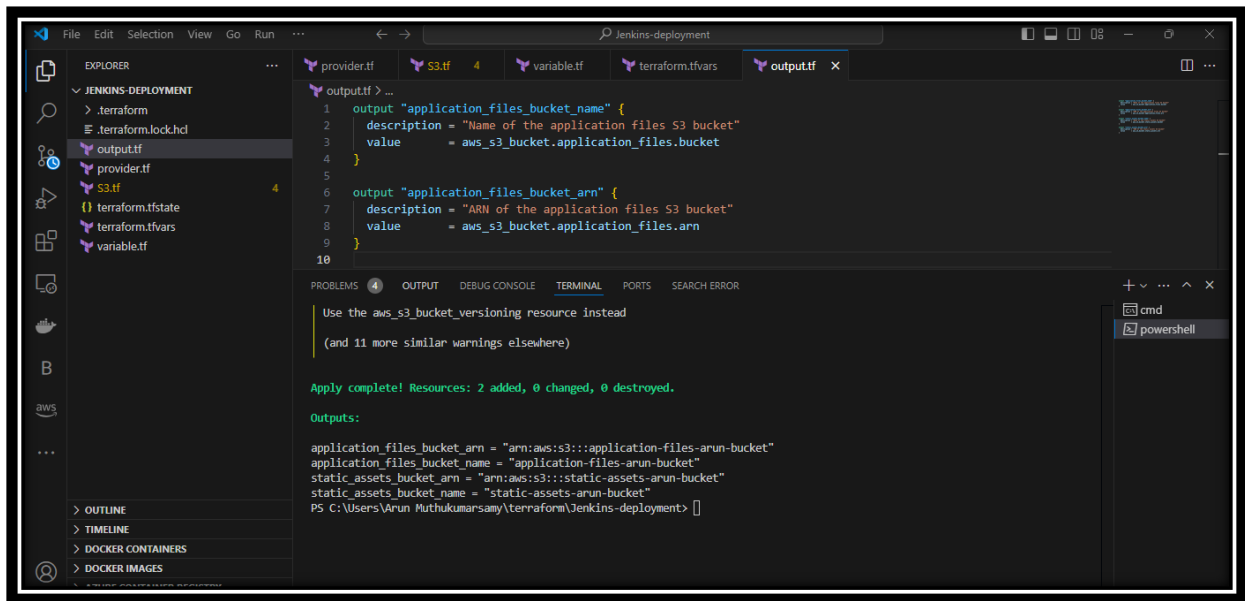
output "static_assets_bucket_name" {
  description = "Name of the static assets S3 bucket"
  value       = aws_s3_bucket.static_assets.bucket
}

output "static_assets_bucket_arn" {
  description = "ARN of the static assets S3 bucket"
  value       = aws_s3_bucket.static_assets.arn
}
```

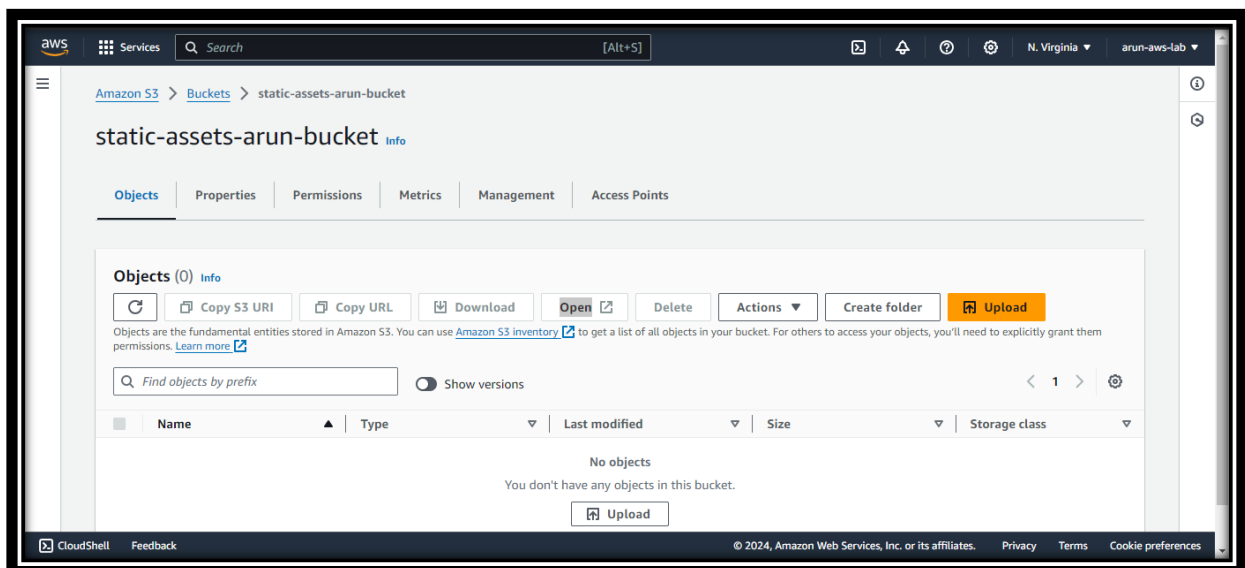
Infrastructure Deployment - Jenkins

Step 2: Executing the Terraform command to verify the template

- **terraform init** - command initializes a terraform configuration directory.
- **terraform plan** - command creates an execution plan. It compares the current state of the infrastructure to the desired state described in terraform files (.tf files) and determines what actions are necessary to achieve the desired state.
- **terraform apply** - command applies the changes required to reach the desired state of the configuration, as determined by the execution plan generated by terraform plan.



Verification on AWS console:



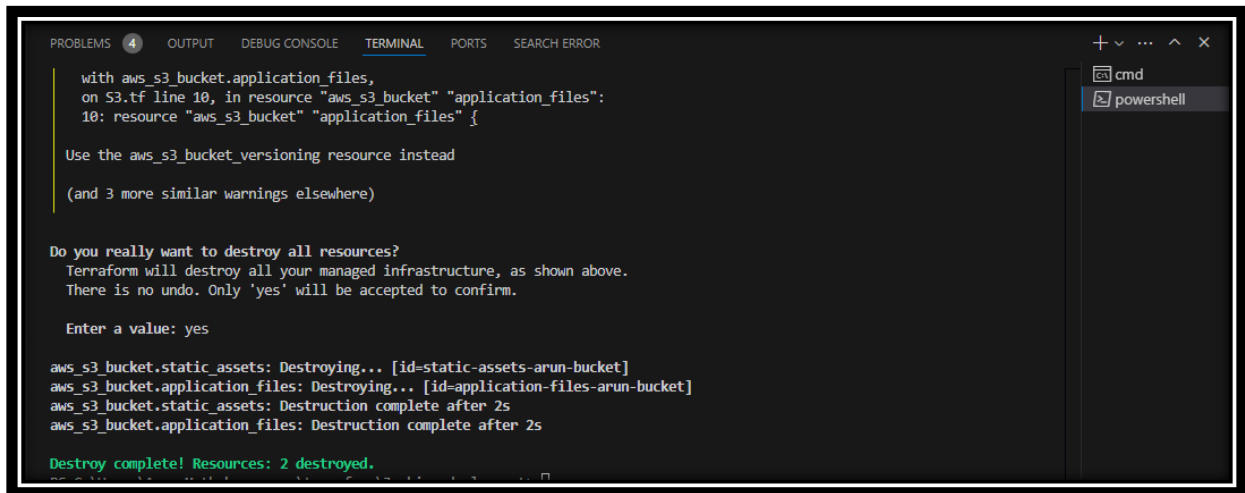
- Terraform code was created in a sample bucket named “static-assets-arun-bucket”. So, our template successfully worked. Without any issues.

Infrastructure Deployment - Jenkins

- Now, we destroy the bucket and then create the bucket using the Jenkins build pipeline.

Step 3: Destroy the bucket.

- **Terraform Destroy:** terraform destroy is used to destroy the created resources in the environment.



```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
with aws_s3_bucket.application_files,
on S3.tf line 10, in resource "aws_s3_bucket" "application_files":
10: resource "aws_s3_bucket" "application_files" {

Use the aws_s3_bucket_versioning resource instead

(and 3 more similar warnings elsewhere)

Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

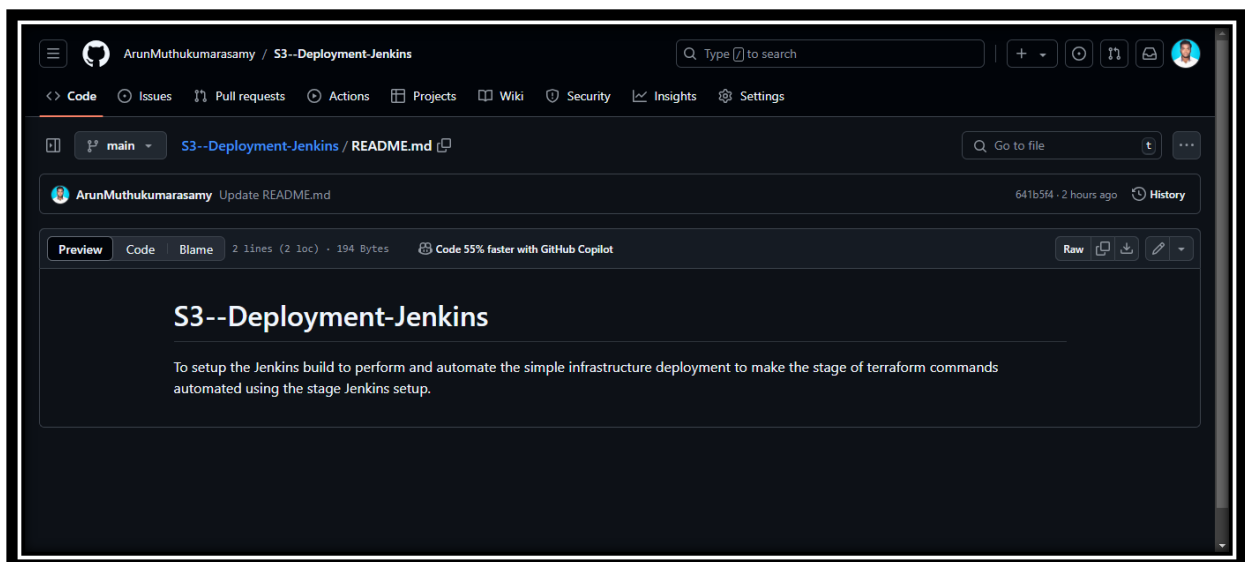
aws_s3_bucket.static_assets: Destroying... [id-static-assets-arun-bucket]
aws_s3_bucket.application_files: Destroying... [id-application-files-arun-bucket]
aws_s3_bucket.static_assets: Destruction complete after 2s
aws_s3_bucket.application_files: Destruction complete after 2s

Destroy complete! Resources: 2 destroyed.
  
```

Step 4: Jenkins Automated Build Setup

Step 1: GitHub Setup:

- To set up the GitHub to versioning the terraform sources code to make an automated build whenever the build was triggered.



ArumMuthukumarasamy / S3--Deployment-Jenkins

< Code Issues Pull requests Actions Projects Wiki Security Insights Settings

S3--Deployment-Jenkins / README.md

ArumMuthukumarasamy Update README.md 641b5f4 · 2 hours ago History

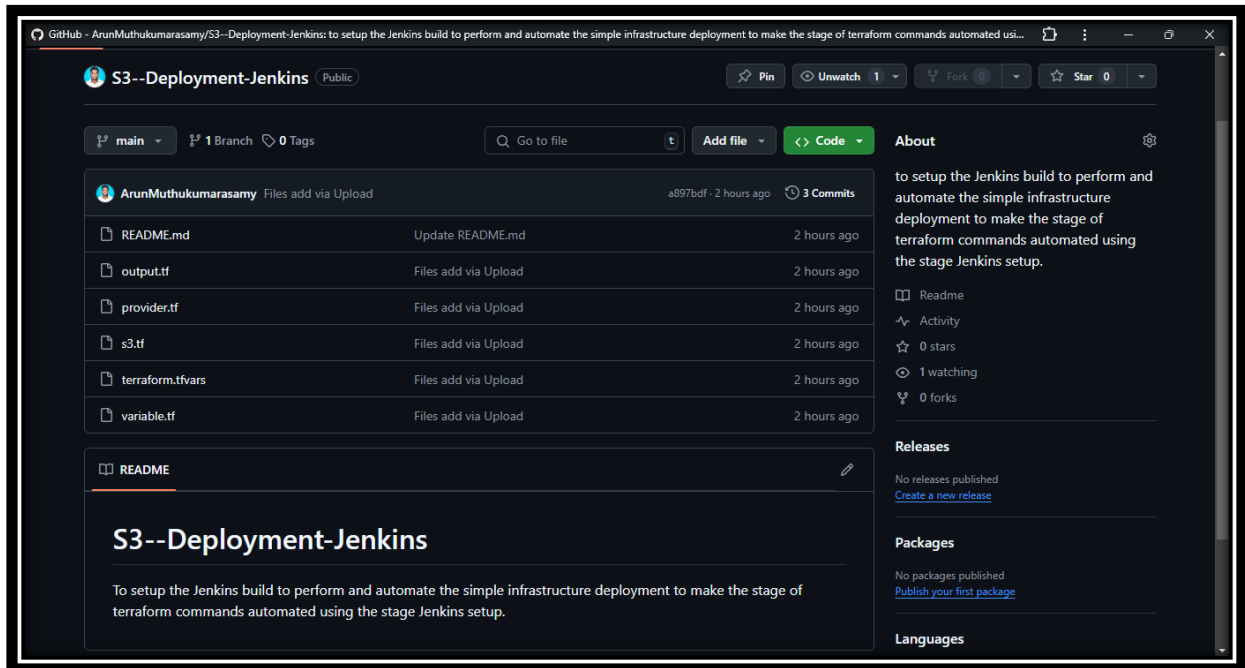
Preview Code Blame 2 lines (2 loc) · 194 Bytes Code 55% faster with GitHub Copilot Raw

S3--Deployment-Jenkins

To setup the Jenkins build to perform and automate the simple infrastructure deployment to make the stage of terraform commands automated using the stage Jenkins setup.

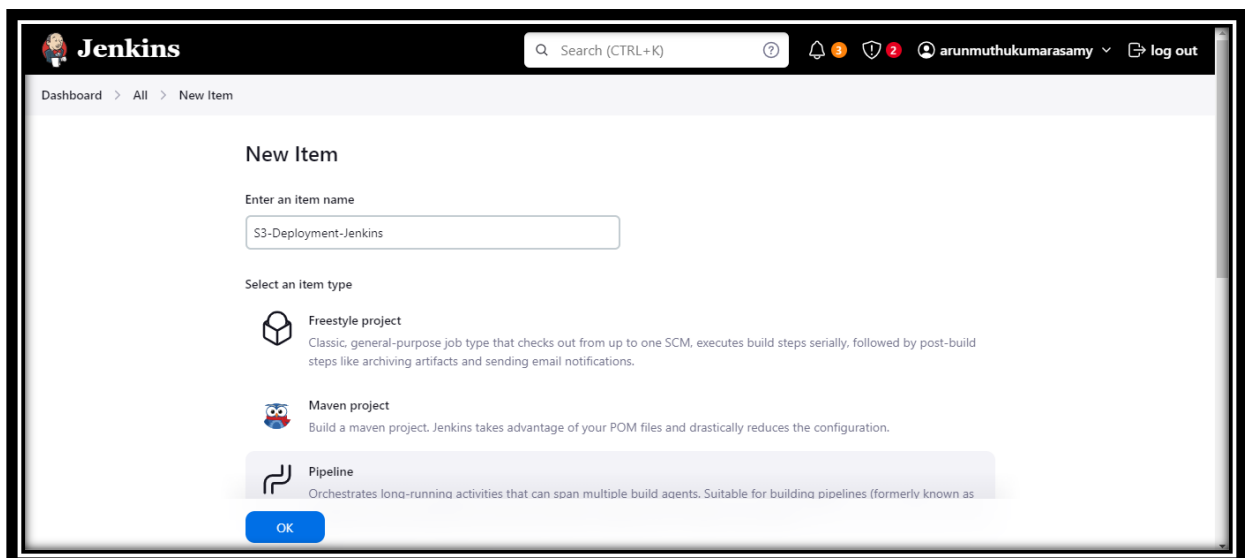
Infrastructure Deployment - Jenkins

- Created a new repository named as “S3-Deployment-Jenkins”.



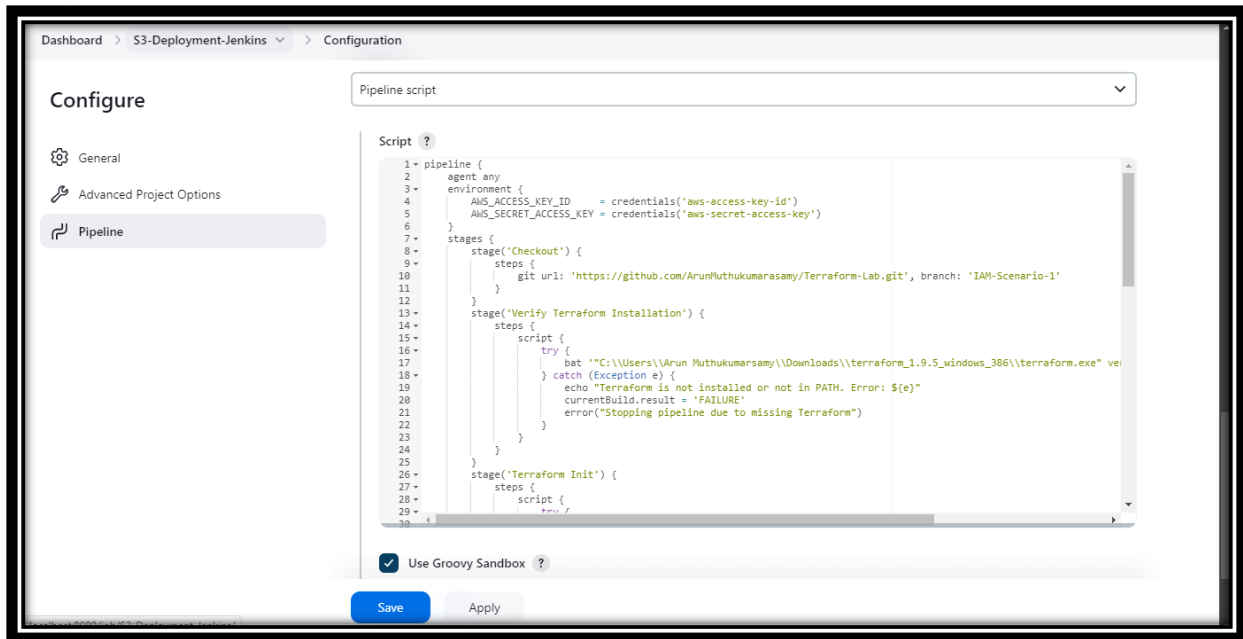
- And PUSH all the file sources of terraform in the GitHub repository.

Step 2: Setup the Jenkins build:



- Create a new item to set up the build named “S3-deployment-Jenkins” >> select item type as a “Pipeline”.

Infrastructure Deployment - Jenkins

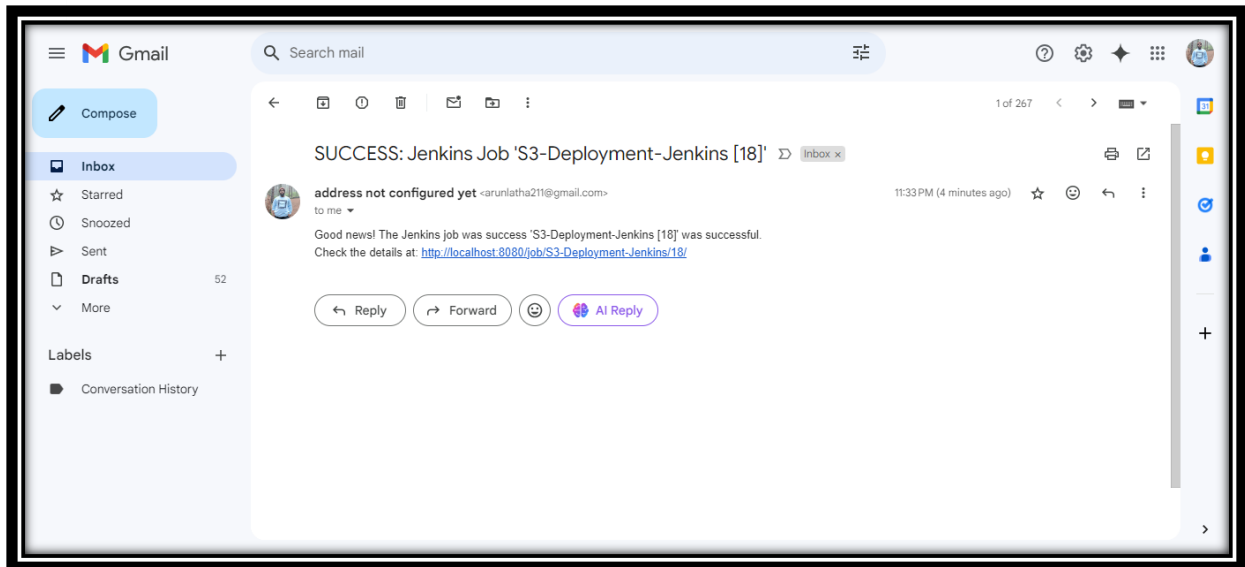


- Setup Groovy script for configuring the stage of the build.



- The build successfully finished without any errors.

Infrastructure Deployment - Jenkins



- Email verification is also done as per requirements. By using SMTP server configuration with the help of the mail extension plugin.

Groovy Script of the build:

```
pipeline {
  agent any
  environment {
    AWS_ACCESS_KEY_ID      = credentials('aws-access-key-id')
    AWS_SECRET_ACCESS_KEY = credentials('aws-secret-access-key')
  }
  stages {
    stage('Checkout') {
      steps {
        git url: 'https://github.com/ArunMuthukumarasamy/S3--
Deployment-Jenkins.git', branch: 'main'
      }
    }
    stage('Verify Terraform Installation') {
      steps {
        script {
          try {
            bat '"C:\\Users\\Arun
Muthukumarasamy\\Downloads\\terraform_1.9.5_windows_386\\terraform.exe"
version'
          } catch (Exception e) {
            echo "Terraform is not installed or not in PATH.
Error: ${e}"
            currentBuild.result = 'FAILURE'
            error("Stopping pipeline due to missing Terraform")
          }
        }
      }
    }
  }
}
```


Infrastructure Deployment - Jenkins

```

    }
    stage('Terraform Init') {
        steps {
            script {
                try {
                    bat '"C:\\Users\\Arun
Muthukumarsamy\\Downloads\\terraform_1.9.5_windows_386\\terraform.exe" init -
input=false'
                } catch (Exception e) {
                    echo "Error during Terraform init: ${e}"
                    currentBuild.result = 'FAILURE'
                    error("Stopping pipeline due to init error")
                }
            }
        }
    }
    stage('Plan') {
        steps {
            script {
                try {
                    bat '"C:\\Users\\Arun
Muthukumarsamy\\Downloads\\terraform_1.9.5_windows_386\\terraform.exe" plan -
input=false'
                } catch (Exception e) {
                    echo "Error during Terraform plan: ${e}"
                    currentBuild.result = 'FAILURE'
                    error("Stopping pipeline due to plan error")
                }
            }
        }
    }
    stage('Apply') {
        steps {
            script {
                try {
                    bat '"C:\\Users\\Arun
Muthukumarsamy\\Downloads\\terraform_1.9.5_windows_386\\terraform.exe" apply -
input=false -auto-approve'
                } catch (Exception e) {
                    echo "Error during Terraform apply: ${e}"
                    currentBuild.result = 'FAILURE'
                    error("Stopping pipeline due to apply error")
                }
            }
        }
    }
}
post {

```

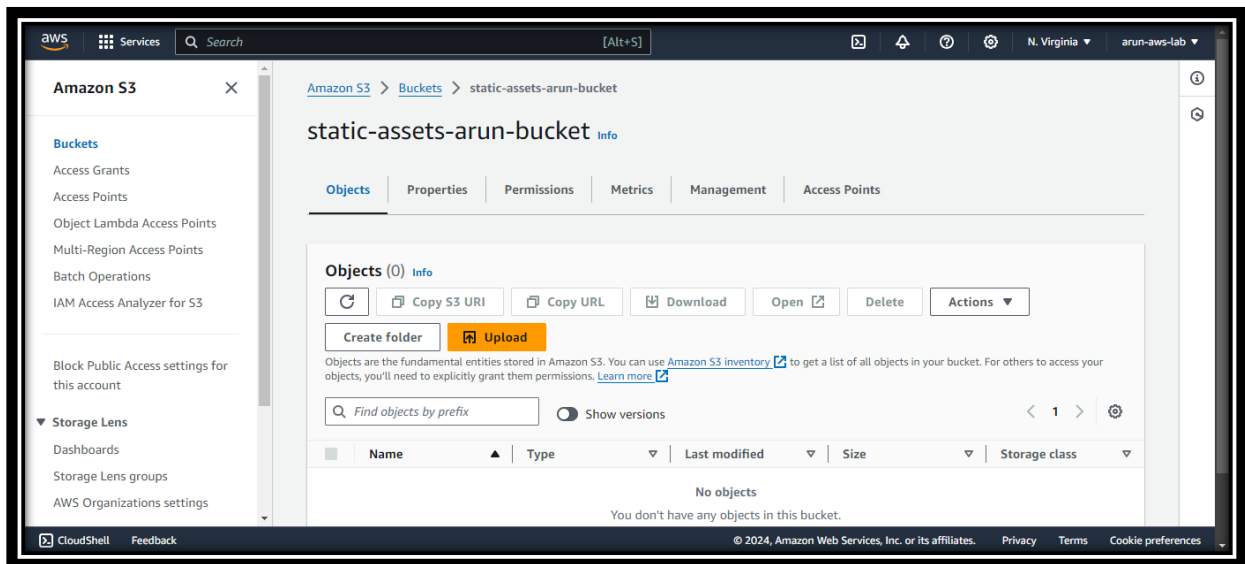
Infrastructure Deployment - Jenkins

```

always {
    cleanWs()
}
success {
    mail to: 'arunlatha211@gmail.com',
        subject: "SUCCESS: Jenkins Job '${env.JOB_NAME}'
[${env.BUILD_NUMBER}]",
        body: "Good news! The Jenkins job was success
'${env.JOB_NAME}' [${env.BUILD_NUMBER}]' was successful. \nCheck the details
at: ${env.BUILD_URL}"
}
failure {
    mail to: 'arunlatha211@gmail.com',
        subject: "FAILURE: Jenkins Job '${env.JOB_NAME}'
[${env.BUILD_NUMBER}]",
        body: "Unfortunately, the Jenkins job was
failure'${env.JOB_NAME}' [${env.BUILD_NUMBER}]' failed. \nCheck the details at:
${env.BUILD_URL}"
}
}
}

```

Step 4: Verification of AWS resource



- Static-assets-arun-bucket named bucket was created successfully.