Step 1:

Write the Terraform Code Create a new directory for your Terraform project. Inside the directory, create a `main.tf` file with the following Terraform code:

```
provider "aws" {

    region = "your-aws-region"

}

resource "aws_instance" "example" {

    ami           = "your-ami-id"

    instance_type = "t2.micro"

    tags = {

        Name = "ExampleInstance"

    }

}

terraform {

    backend "s3" {
```

```
        bucket = "your-bucket-name"

        key    = "terraform.tfstate"

        region = "your-aws-region"

    }

}
```

Replace the placeholders such as “your-aws-region”, “your-ami-id”, and “your-bucket-name” with appropriate values for your AWS setup.

Step 2:

Install Required Plugins Ensure you have Jenkins installed, and then install the following plugins through the Jenkins Plugin Manager:

- •Jenkins Job DSL Plugin
- •AWS CLI Plugin
- •Terraform Plugin
- •Pipeline Utility Steps Plugin

Step 3:

Configure AWS Credentials To allow Jenkins to interact with AWS, you need to set up AWS credentials:

1. Go to “Manage Jenkins” > “Manage Credentials” > “Jenkins” Store.
2. Add a new “AWS Credentials” with your AWS Access Key ID and Secret Access Key.

Step 4:

Create the Seed Job using DSL

1. Create a new Freestyle project in Jenkins and name it “Seed Job — Terraform Pipeline.”

2.In the “Build” section, select “Process Job DSLs” and provide the DSL script below:

```
pipelineJob("Terraform Pipeline") {

    definition {

        cpsScm {

            scm {

                git('https://github.com/your-terraform-repo.git')

            }

            scriptPath('Jenkinsfile')

        }

    }

}
```

Step 5:

Set up your Terraform Code Repository Create a Git repository with your Terraform code, including the `main.tf` file that describes your desired EC2 instance.

Step 6:

Write the Jenkinsfile In your Terraform code repository, create a `Jenkinsfile` with the following stages:

```
pipeline {

    agent any

    environment {

        AWS_DEFAULT_REGION = 'your-aws-region'

    }

    stages {

        stage('Checkout Code') {

            steps {

                checkout scm

            }

        }

        stage('Terraform Init') {
```

```
        steps {

            script {

                sh 'terraform init'

            }

        }

    }

    stage('Terraform Plan') {

        steps {

            script {

                sh 'terraform plan -out=tfplan'

            }

        }

    }

    stage('Terraform Apply') {
```

```
                steps {

                    script {

                        sh 'terraform apply -auto-approve tfplan'

                    }

                }

            }

            stage('Upload State to S3') {

                steps {

                    script {

                        sh 'aws s3 cp terraform.tfstate s3://your-bucket-name'

                    }

                }

            }

        }
```

```
        post {

            always {

                cleanWs()

            }

        }

}
```

Step 7:

Configure Jenkins Job

1. In Jenkins, create a new pipeline job.
2. Under the “Pipeline” section, select “Pipeline script from SCM.”
3. Provide the repository URL and specify the Jenkinsfile location (e.g., `Jenkinsfile`).
4. Save the job configuration.

Step 8:

Run the Jenkins Job Now, you can run the pipeline job to execute your Terraform code and provision an EC2 instance. Jenkins will fetch the code from the repository, initialize Terraform, create a plan, apply the changes, and upload the Terraform state to an S3 bucket.