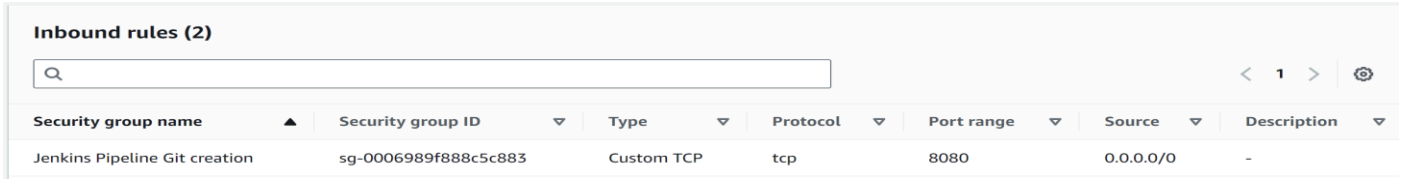
Name: Pranav Chaudhari

Task: Create a git repository and create 2 branches main and dev and add CODEOWNERS file on main branch, also enable branch protection on main branch all this with Jenkins pipeline.

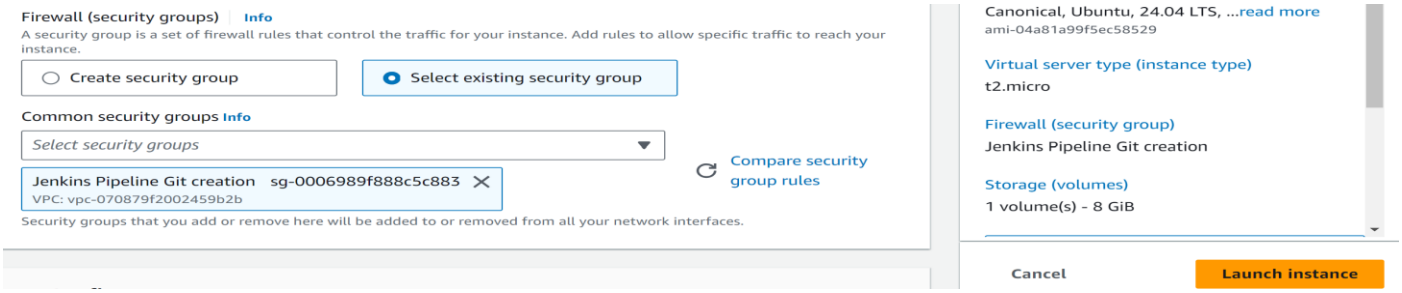
Github Repo: https://github.com/techdecipher/Jenkins-Auto-Repository22

Step 1-- Launch EC2 instance: Goto AWS > EC2> launch a new instance, > choose AMI > ensure port 8080 is enabled from everywhere as Jenkins work on the port 8080.

SG inbound Rules



Launch Instance



Step 2-- Install Jenkins: Jenkins requires Java to run, so installing OpenJDK before I install Jenkins. So connected to EC2 instance to install OpenJDK and Jenkins.

Install OpenJDK

```
sudo apt update -y
sudo apt install openjdk-11-jdk -y
sudo apt-get update
java -version
```

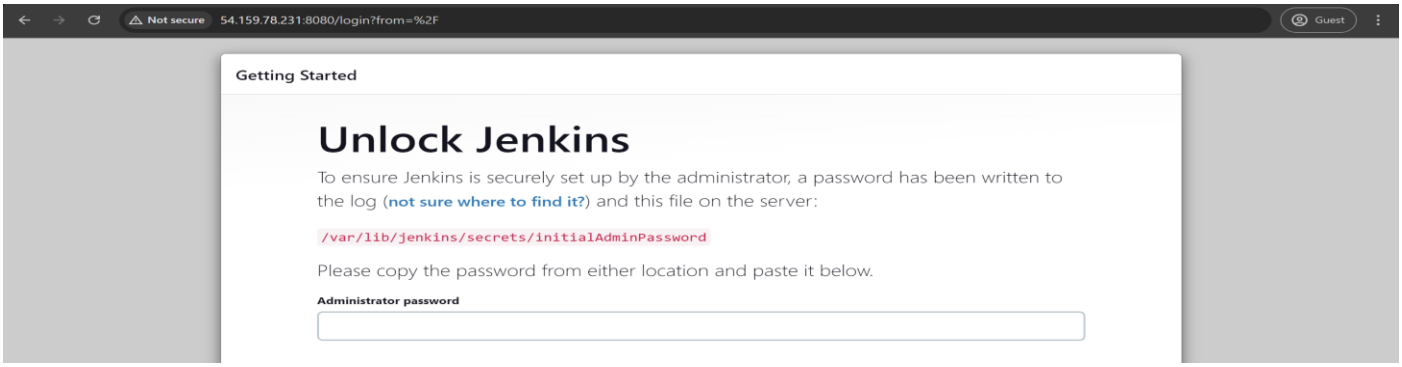
```
ubuntu@ip-172-31-69-211:~$ java -version
openjdk version "11.0.24" 2024-07-16
OpenJDK Runtime Environment (build 11.0.24+8-post-Ubuntu-1ubuntu324.04.1)
OpenJDK 64-Bit Server VM (build 11.0.24+8-post-Ubuntu-1ubuntu324.04.1, mixed mode, sharing)
ubuntu@ip-172-31-69-211:~$
```

Install Jenkins

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
sudo systemctl status jenkins
```

```
ubuntu@ip-172-31-69-211:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
     Active: active (running) since Fri 2024-08-09 08:09:15 UTC; 17s ago
   Main PID: 4728 (java)
      Tasks: 44 (limit: 1130)
     Memory: 289.3M (peak: 330.4M)
        CPU: 17.947s
     CGroup: /system.slice/jenkins.service
             └─4728 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins
/war --httpPort=80
```

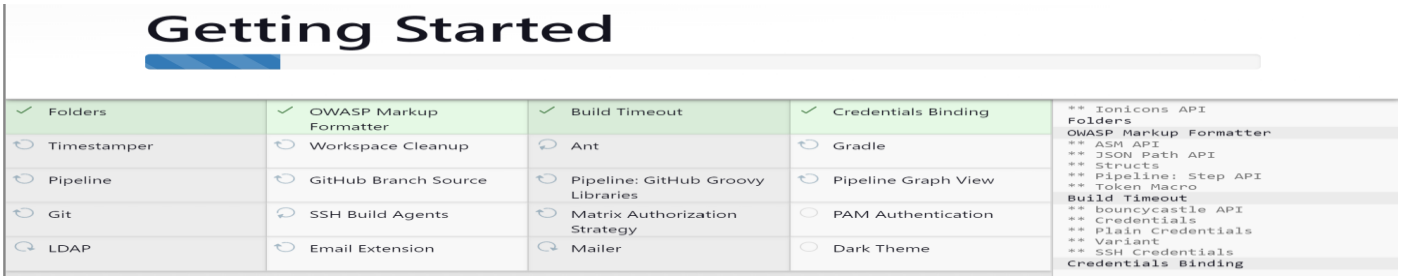
Step 3-- access Jenkins: Grabbed the public IP of EC2 instance and used the port 8080 to access it. To access the Jenkins on non-changeable URL or IP, we can use elastic IP or Load balancer however for now I am just using EC2 public IP.



Retrieve secrete text to login, later this can be changed. User would be admin.

```
ubuntu@ip-172-31-69-211:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
db04e4dc85ed4683bdf99a033e501f15
ubuntu@ip-172-31-69-211:~$
```

Custom Jenkins > install recommended packages >



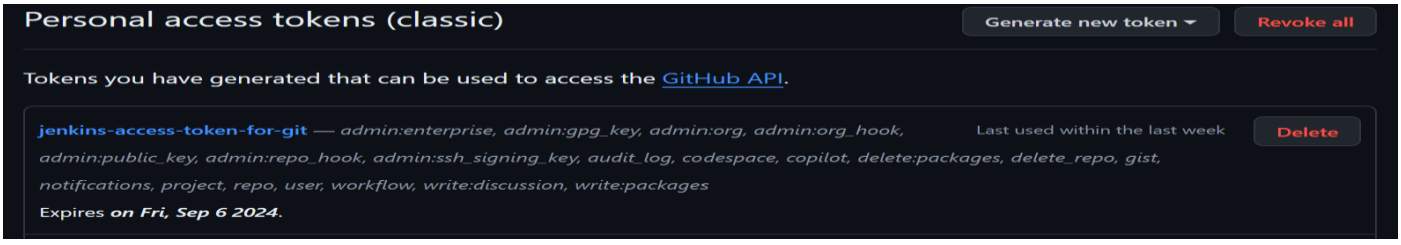
Step 4-- Install necessary plugins: Jenkins supports wider range of plugins, that makes Jenkins extensible. I would need 3 plugins to install for this task.

1. Pipeline plugin: for running Jenkins pipelines
2. Git plugin: allows Jenkins to interact with Git repositories [ensure base OS has git installed too]
3. HTTP Request plugin: to make HTTP requests to APIs

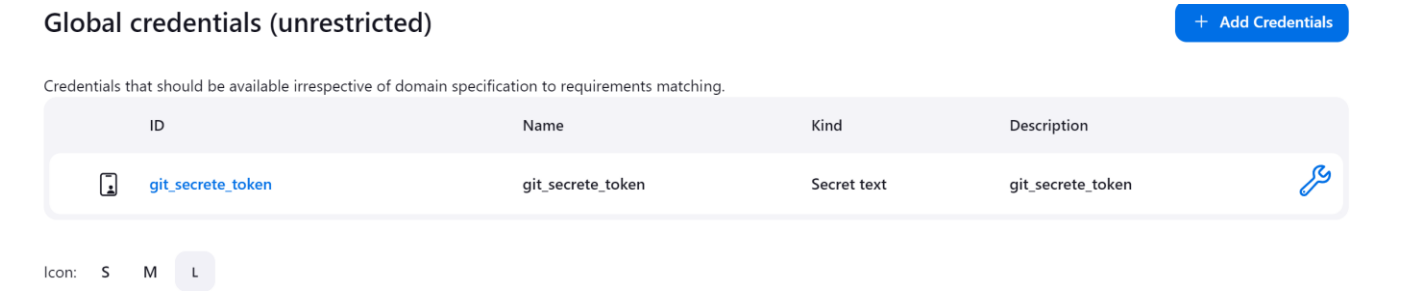
Manage Jenkins > Plugins > Available plugins >

Step 5-- Setup personal access token: from your git account into Jenkins manage Jenkins >

Github account > Settings > Developer Settings > Personal access token > Token (classic) (enabled admin access)



Setup token into the Jenkins > Manage Jenkins > Credentials > Global Credentials > Add Credentials > Secrete text>



Step 6-- Script stages:

Stage-1: Create Git Repository: Creating the script in groovy for the Jenkins pipeline. Define Env variable and Git creation stage

```
pipeline {
    agent any
    environment {
        GITHUB_TOKEN = credentials('classic-token-git') // GitHub token ID from jenkins
        REPO_NAME = 'Jenkins-Auto-Repository22'        // Repository name
        MAIN_BRANCH = 'main'                           // Main branch name
        DEV_BRANCH = 'dev'                             // Dev branch name
        YOUR_USERNAME = 'techdecipher'                 // Primary accounts GitHub username
        COLLABORATOR_USERNAME = 'waytopranav'          // Collaborator's GitHub username
        PERMISSION_LEVEL = 'push'                      // Permission level for the collaborator
    }
    stages {
        stage('Create Git Repository') {
            steps {
                script {
                    httpRequest(
                        httpMode: 'POST',
                        url: 'https://api.github.com/user/repos', // Endpoint
                        contentType: 'APPLICATION_JSON',
                        customHeaders: [
                            [name: 'Authorization', value: 'Bearer ' + GITHUB_TOKEN]
                        ],
                        requestBody: """{
                            "name": "$REPO_NAME",
                            "private": false
                        }"""
                    )
                }
            }
        }
```

Stage-2: Initial Commit with CODEOWNERS: Do initial commit with Readme file and CODONWERS file.

```
stage('Add Initial Commit with CODEOWNERS') {
            steps {
                script {
                    def readmeContent = "Welcome to ${REPO_NAME}"
                    def codeownersContent = """\
                    *.html @waytopranav
                    """

                    httpRequest(
                        httpMode: 'PUT',
                        url: 'https://api.github.com/repos/' + YOUR_USERNAME + '/' + REPO_NAME + '/contents/README.md',
                        contentType: 'APPLICATION_JSON',
                        customHeaders: [
                            [name: 'Authorization', value: 'Bearer ' + GITHUB_TOKEN]
                        ],
                        requestBody: """{
                            "message": "Initial commit with README.md and CODEOWNERS",
                            "content": "${readmeContent.bytes.encodeBase64().toString()}"
                        }"""
                    )

                    httpRequest(
                        httpMode: 'PUT',
                        url: 'https://api.github.com/repos/' + YOUR_USERNAME + '/' + REPO_NAME + '/contents/.github/CODEOWNERS',
                        contentType: 'APPLICATION_JSON',
                        customHeaders: [
                            [name: 'Authorization', value: 'Bearer ' + GITHUB_TOKEN]
                        ],
                        requestBody: """{
                            "message": "Add CODEOWNERS file",
                            "content": "${codeownersContent.bytes.encodeBase64().toString()}"
                        }"""
                    )
                }
            }
        }
```

Stage-3: Create Dev Branch: Create a main branch from the Dev branch.

```
stage('Create Dev Branch') {
        steps {
            script {
                def getMainBranchSHA = httpRequest(
                    httpMode: 'GET',
                    url: 'https://api.github.com/repos/' + YOUR_USERNAME + '/' + REPO_NAME + '/git/refs/heads/main',
                    customHeaders: [
                        [name: 'Authorization', value: 'Bearer ' + GITHUB_TOKEN]
                    ]
                )

                def shaMainBranch = new groovy.json.JsonSlurper().parseText(getMainBranchSHA.content).object.sha

                httpRequest(
                    httpMode: 'POST',
                    url: 'https://api.github.com/repos/' + YOUR_USERNAME + '/' + REPO_NAME + '/git/refs',
                    contentType: 'APPLICATION_JSON',
                    customHeaders: [
                        [name: 'Authorization', value: 'Bearer ' + GITHUB_TOKEN]
                    ],
                    requestBody: """{
                        "ref": "refs/heads/${DEV_BRANCH}",
                        "sha": "$shaMainBranch"
                    }"""
                )
            }
        }
    }
```

Stage-4: Add Collaborator: The user must be collaborator in order to do PR requests.

```
stage('Add Collaborator') {
        steps {
            script {
                httpRequest(
                    httpMode: 'PUT',
                    url: 'https://api.github.com/repos/' + YOUR_USERNAME + '/' + REPO_NAME + '/collaborators/' + COLLABORATOR_USERNAME,
                    contentType: 'APPLICATION_JSON',
                    customHeaders: [
                        [name: 'Authorization', value: 'Bearer ' + GITHUB_TOKEN]
                    ],
                    requestBody: """{
                        "permission": "${PERMISSION_LEVEL}"
                    }"""
                )
            }
        }
    }
```

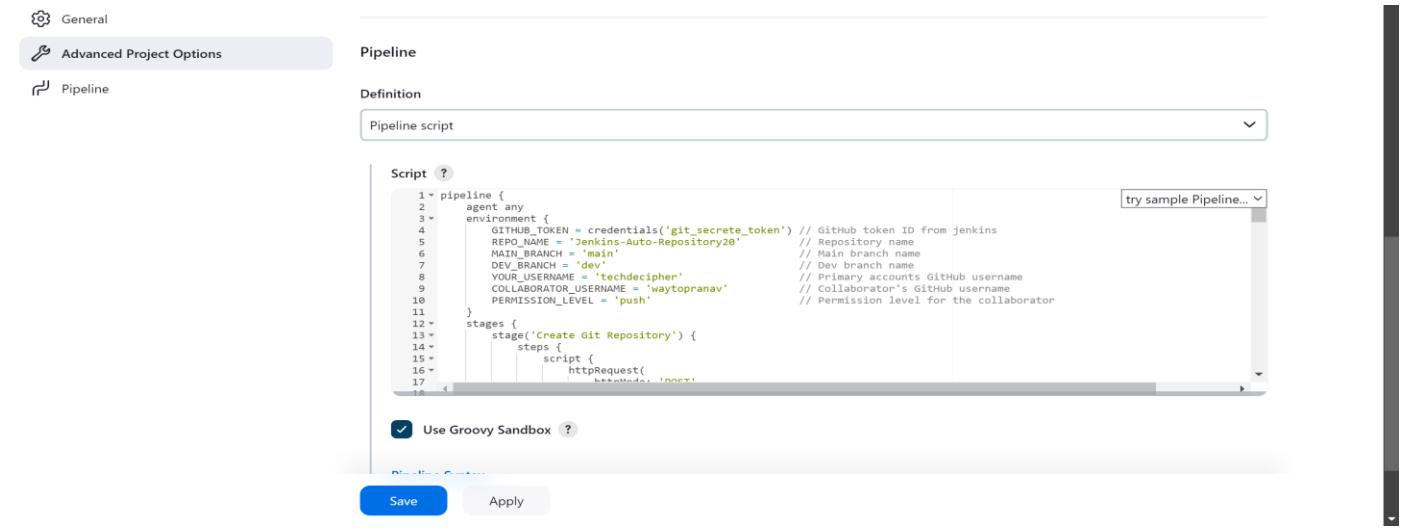
Stage-5: Branch Protection for Main: Setup branch protection with all html file to be reviewed by the CODEOWNER

```
stage('Set Branch Protection for Main') {
        steps {
            script {
                httpRequest(
                    httpMode: 'PUT',
                    url: 'https://api.github.com/repos/' + YOUR_USERNAME + '/' + REPO_NAME + '/branches/' + MAIN_BRANCH + '/protection',
                    contentType: 'APPLICATION_JSON',
                    customHeaders: [
                        [name: 'Authorization', value: 'Bearer ' + GITHUB_TOKEN],
                        [name: 'Accept', value: 'application/vnd.github.luke-cage-preview+json'] // Required for branch protection
                    ],
                    requestBody: """{
                        "required_status_checks": {
                            "strict": true,
                            "contexts": []
                        },
                        "enforce_admins": false,
                        "required_pull_request_reviews": {
                            "required_approving_review_count": 1,
                            "dismiss_stale_reviews": true,
                            "require_code_owner_reviews": true
                        },
                        "restrictions": null,
                        "allow_force_pushes": false,
                        "allow_deletions": false
                    }"""
                )
```

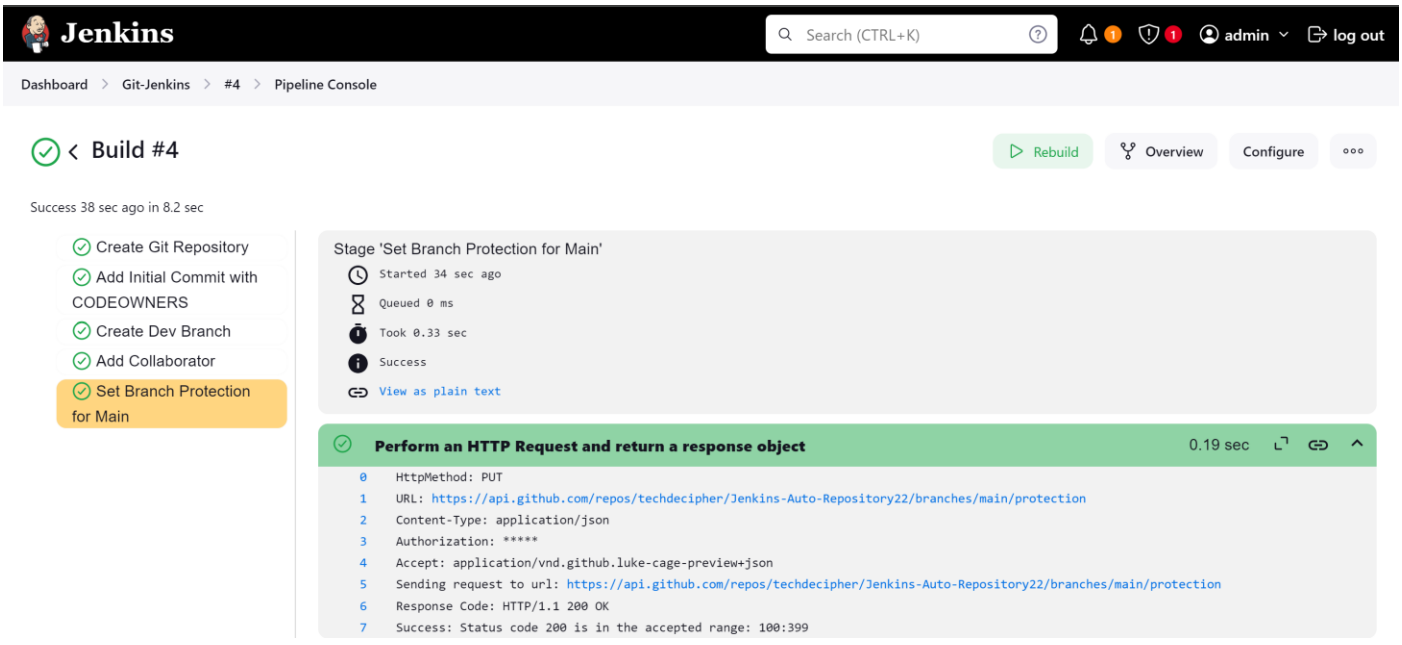
Step 7-- Final Script: https://github.com/techdecipher/cloudformation-templates/blob/main/Jenkins_pipepline

Step 8-- Create New Job on Jenkins: Going over to Jenkins dashboard > New Item > Give name and Select Pipeline >

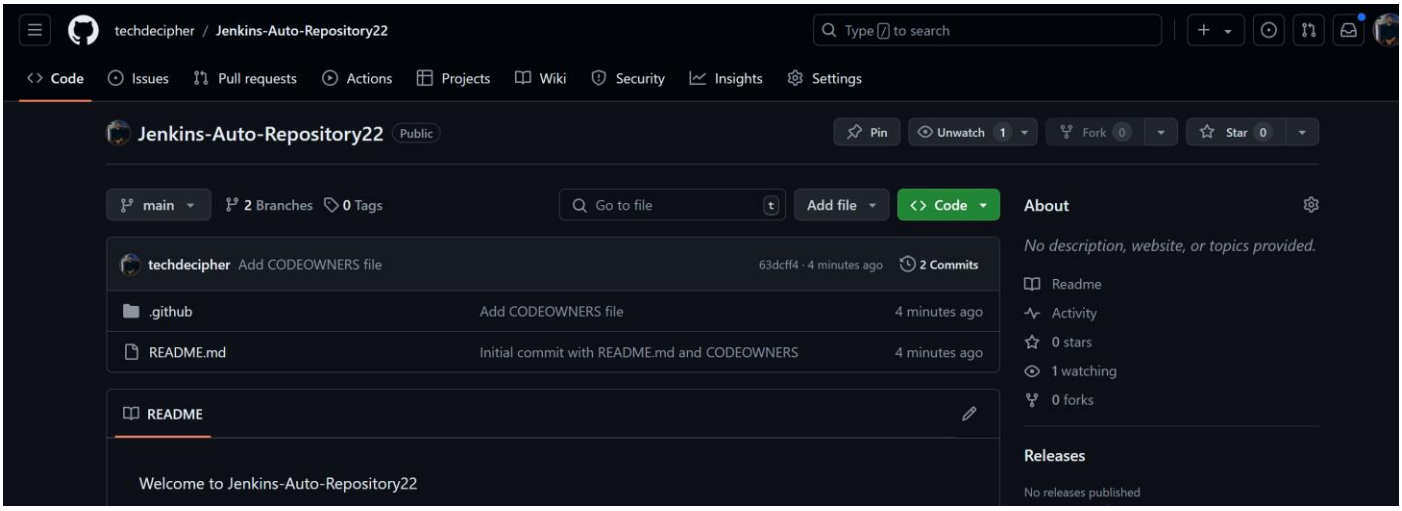
In next screen> paste the script and apply.



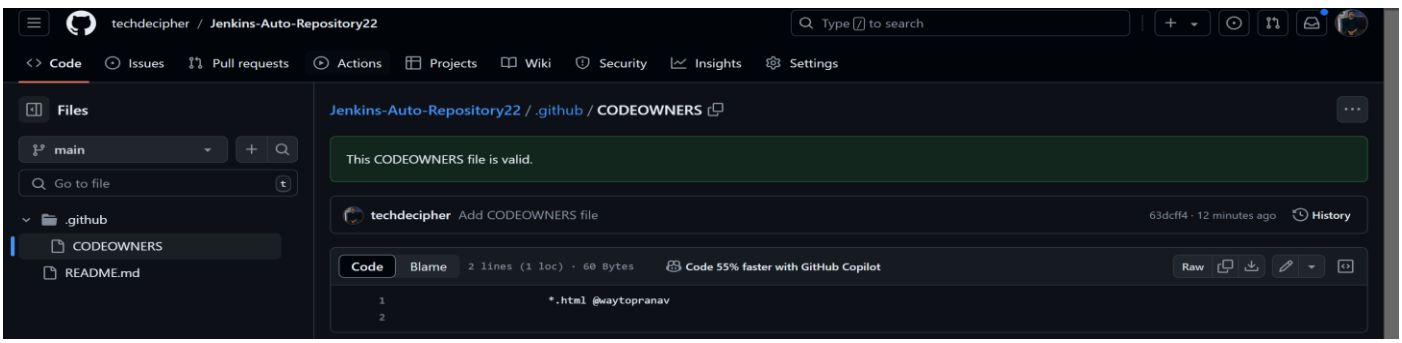
Pipeline Output



Step 9-- On the Git side: the branch is created with all the stages.



Got the invitation for the same.



Now any changes in the .html file will be notified to the user set in the CODEONWER file. Make changes to the file, Commit and push the changes, and do a PR request to analyse the brank protection.