DEPLOPYMENT UTILIZING A COMPLETE SUITE OF AUTOMATION TOOLS IN JENKINS.

STEP 1: Launch the Jenkins Server with allowing the Jenkins default port number 8080.

- Switch to the root user by using the command “sudo su –“.
- Update the server by using the command “apt update –y”.
- Install the Java by using the command “apt install openjdk-17-jre –y”.
- Install the Maven by using the command “apt install maven –y”.
- Install and start the Jenkins in a Jenkins terminal.
- Login to the Jenkins server.

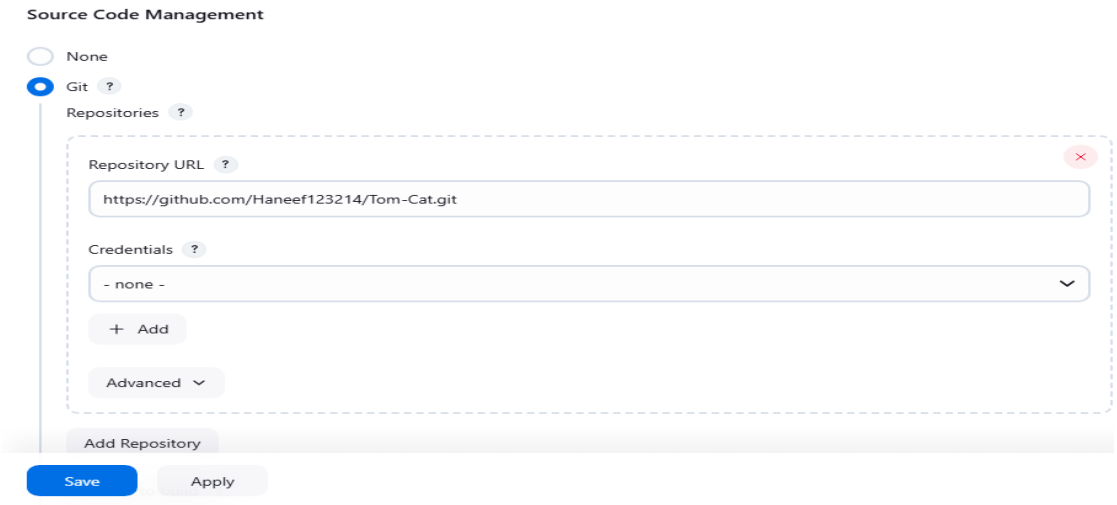
Step 2: Launch the Sonarqube Server with allowing the Sonarqube default port number 9000.

Step 3: Launch the Nexus Server with allowing the Nexus default port number 8081.

Step 4: Launch the Tomcat Server with allowing the port number 8088.

Step 5: Switch to the Jenkins Server.

- Install the recommended plugins and create a job using either a Freestyle project or a Pipeline.
- Once the Git configuration is complete, trigger the build. If the build is successful, it indicates that the task is finished as shown in below figure.



- Create a Pipeline job and select the **Checkout** stage.

- Generate the pipeline syntax for the checkout step using Jenkins' **Pipeline Syntax Generator**. Copy the generated syntax and paste it into the pipeline script to perform the build.

```
pipeline {
    agent any

    stages {
        stage('checkout') {
            steps {
                git branch: 'main', url: 'https://github.com/Haneef123214/Java.src.git'
            }
        }
```

- After the checkout stage, define a new stage for the **build** stage, and specify the build commands (e.g., `sh 'mvn clean package'`).
- Save the pipeline job and trigger the build to perform both the checkout and build steps.

```
stage('build') {
    steps {
        sh 'mvn clean package'
    }
}
```

Step 6: Log in to your SonarQube server, navigate to your user profile, and generate an authentication token. Copy the generated token.

- Go to the Jenkins **Manage Plugins** page, search for the **SonarQube Scanner** plugin, install it, and restart Jenkins if needed.
- **Configure SonarQube Credentials**: In Jenkins, go to **Manage Jenkins > Configure System**, find the SonarQube section.
- And add the SonarQube server details along with the authentication token as credentials.
- Generate the SonarQube pipeline syntax by configuring the SonarQube credentials in the **Pipeline Syntax** generator, then copy the generated syntax and paste it into your pipeline script as shown in the figure.

```
        stage('sonarqube scanner') {
            steps {
                withSonarQubeEnv('sonar') {
    sh 'mvn sonar:sonar'
}
            }
        }
```

Step 7: Switch to the Nexus Server and create the repository.

Switch to the Jenkins server.

- Installing the necessary plugins and restarting the Jenkins server, navigate to the **Build Steps** section in Jenkins and select the **Nexus Artifact Uploader** option.
- Enter the required credentials for Nexus Artifact Uploader and provide the necessary information related to the **POM file** in the uploader configuration.
- Generate the pipeline script using the Nexus Artifact Uploader configuration in Jenkins.
- Copy the generated script and paste it into the pipeline script section as shown in the provided figure.

```
        stage('sonarqube scanner') {
            steps {
                withSonarQubeEnv('sonar') {
    sh 'mvn sonar:sonar'
}
                }
            }
            stage('nexus uploader') {
                steps {
                    nexusArtifactUploader artifacts: [[artifactId: 'vprofile', classifier: '', file: 'target/vprofile-v2.war', type: 'war']], credentialsId: 'nexus', groupI
                }
            }
```

- Wait until the build get successful.

Step 8: Provide the Tomcat credentials in the Jenkins server configuration.

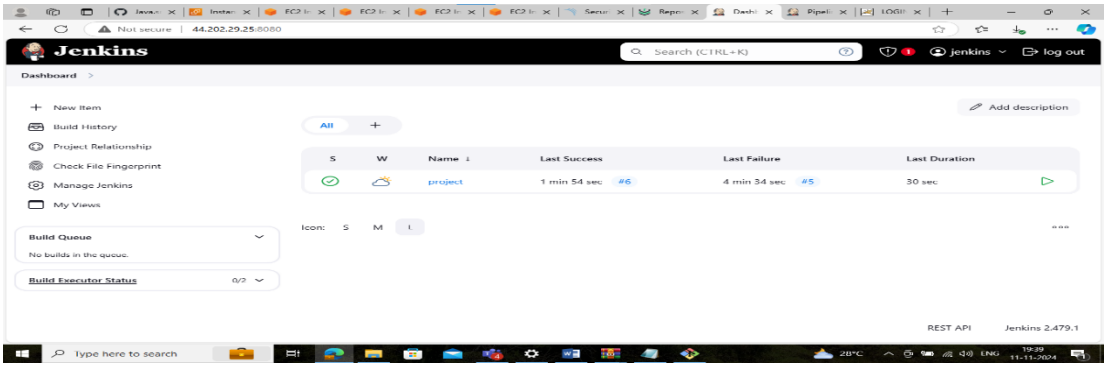
- Use the **Post-build Actions** option to select **Deploy WAR or EAR to a container**, and enter the Tomcat credentials.

- Generate the pipeline syntax.

```
pipeline {
    agent any

    stages {
        stage('checkout') {
            steps {
                git branch: 'main', url: 'https://github.com/Haneef123214/Java.src.git'
            }
        }
        stage('build') {
            steps {
                sh 'mvn clean package'
            }
        }
        stage('sonarqube scanner') {
            steps {
                withSonarQubeEnv('sonar') {
    sh 'mvn sonar:sonar'
}
            }
        }
        stage('nexus uploader') {
            steps {
                nexusArtifactUploader artifacts: [[artifactId: 'vprofile', classifier: '', file: 'target/vprofile-v2.war', type: 'war']], credentialsId: 'nexus', groupI
            }
        }
        stage('deploy') {
            steps {
                deploy adapters: [tomcat9(credentialsId: 'tomcat', path: '', url: 'http://18.234.63.217:8088/')], contextPath: 'Deployment', war: 'target/*.war'
            }
        }
    }
}
```

- Here you will see the job is successfully build as shown in the figure.



Step 9: Switch to Tomcat server.

- Here you will see the deployment of an application as shown in the figure.