Jenkins

Continuous Integration

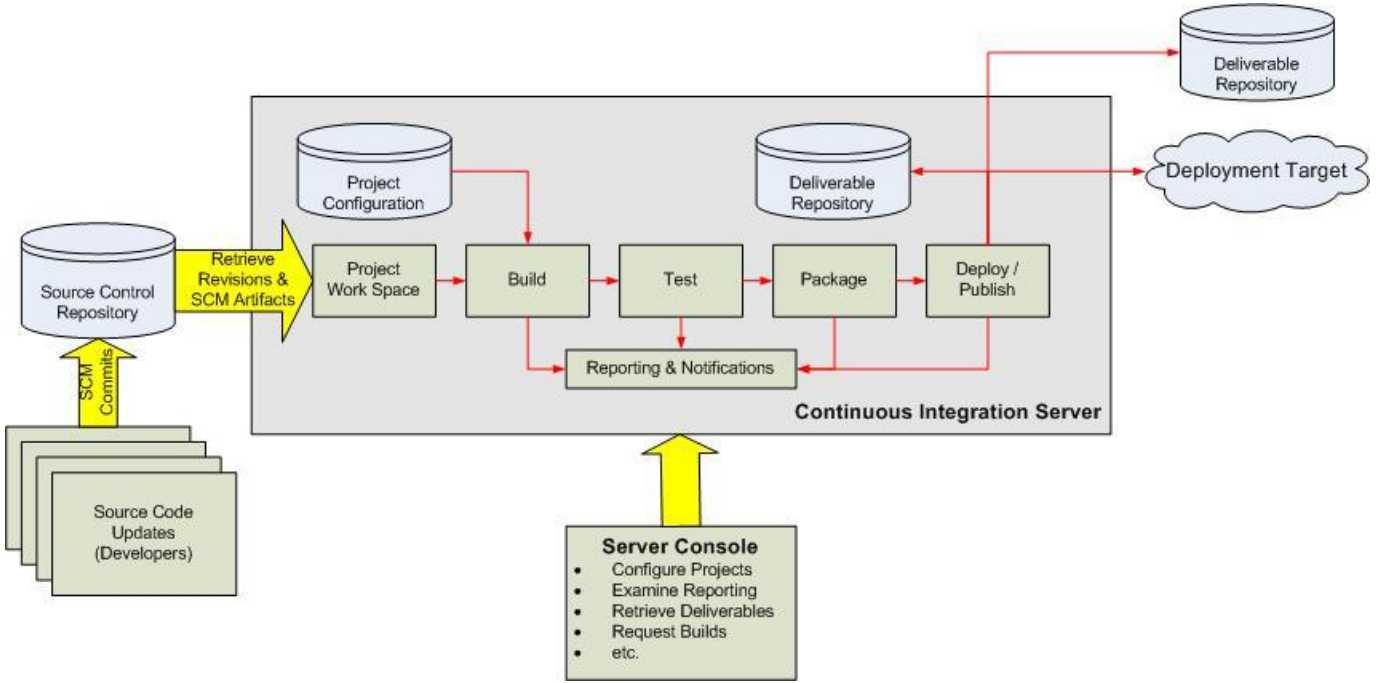
Avinash Patel

Senior Manager

Agenda

- Introduction – Jenkins
  - UI Walkthrough
  - Configure Git, Java, Maven and Email Notification
- Creating Jenkins Freestyle Job
  - Configure Build using batch commands & top level Maven goals
  - Configure Build using plugin based Maven goals
  - Configure Build, selenium test, deploy on server and notification
  - Automation of CI by defining poll SCM, hook trigger
- Creating Jenkins Pipeline Job
  - Construct groovy script to define simple pipeline job
  - Construct pipeline and monitor execution of defined stages

Continuous Integration Overview

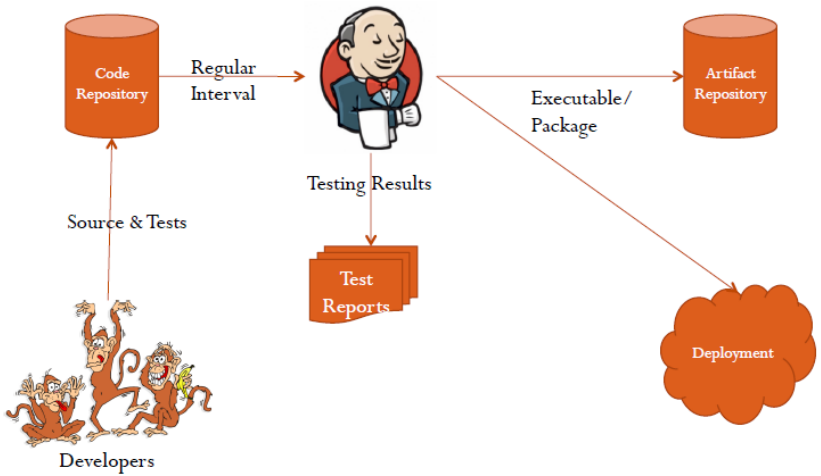


source: http://www.javaworld.com/javaworld/jw-12-2008/images/CIOverview.jpg

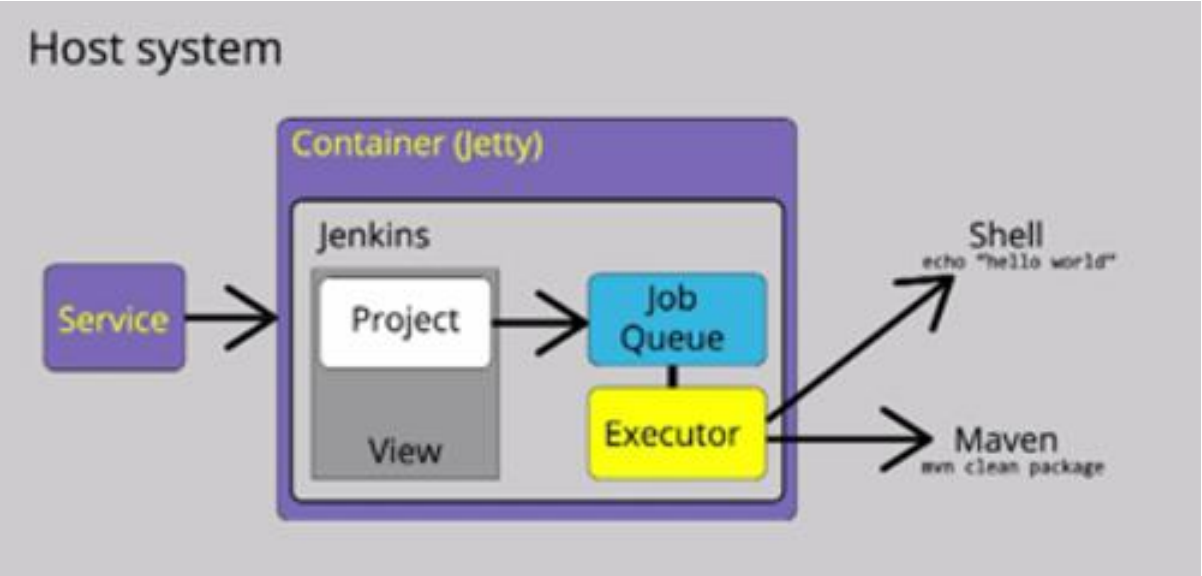
Jenkins - Introduction

- Java based Continuous Build System
- Jenkins is a highly configurable system by itself
- Jenkins monitors the whole build process and provides reports and notifications to alert maintainers on success or errors
- Runs in servlet container
  - Glassfish, Tomcat

Benefits
- Immediate bug detection
- No integration step in the lifecycle
- A deployable system at any given point
- Record of evolution of the project



Jenkins Architecture



How Jenkins works - Setup

- It associates with a version control server
- It triggers builds through polling, periodic and building based on other projects
- It executes Maven targets, bash scripts etc.,
- It does Artifact archival
- It publishes javadocs and JUnit test results
- It does email notifications
- Through plugins it expands the functionality even further

Jenkins for a Developer

- **Easy to install**
  - First download the file – jenkins.war
  - The Run the command – java –jar jenkins.war

  Note: You can alternatively use windows installer
- **Easy to use**
  - First create a new job – You could initially checkout and build a small project
  - Then check-in a change – watch it build
  - Further create a test – watch it build and run
  - Subsequently fix a test – check-in and watch it pass
- **Multi-technology**
  - You can develop code with C, Java, C#, Python, Perl, SQL, etc.
  - You can test with Junit, Nunit, MSTest, Selenium etc.

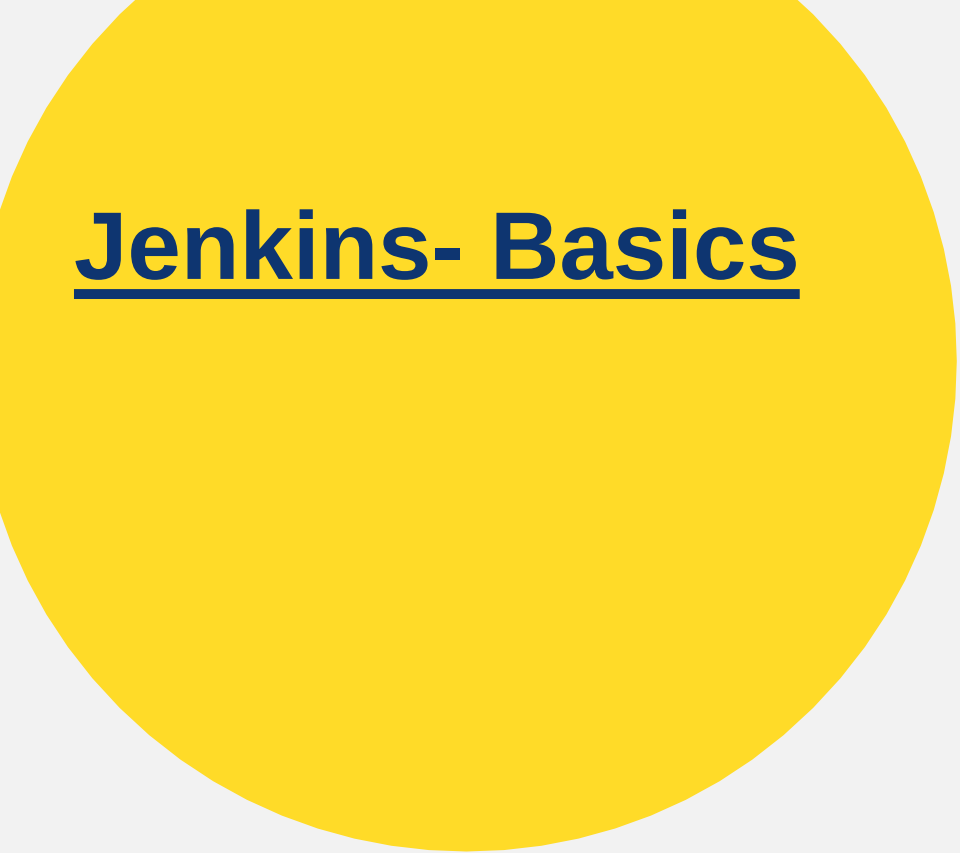
Jenkins Features

- Get source code from source code repository
- Trigger a build - Automatically build and test
- Post build, generate report & notify as Jenkins comes with basic reporting features that would help us to keep track of build status based on its last success and failure status
  - These features can be enhanced with use of pre-build plugins
    - Unit test coverage and Test result trending
- Deploy
- Distributed build

Jenkins terminologies

Term used in Jenkins	Description
Upstream project	A project can have one or several upstream projects, which means that current project is scheduled depending if the upstream project is built successfully or not. If the upstream project is successful the current project is added to the build queue. If the upstream project is broken the current project will not be added to the build queue.
Downstream project	A project can have one or several downstream projects. The downstream projects are added to the build queue if the current project is built successfully. It is an option to add the downstream project to the build queue even if the current project is unstable (default is off).
Stable build	A build is stable if it was built successfully and no publisher reports it as unstable.
Unstable build	A build is unstable if it was built successfully and one or more publishers report it unstable. For example if the JUnit publisher is configured and a test fails then the build will be marked unstable.
(Un)Stable project	A project is (un)stable if its most recent (completed) build is (un)stable.
Successful build	A build is successful when the compilation reported no errors.
Broken build Failed Build	A build is broken if it failed during building. That is, it is not successful.
Broken project	A project is broken if its most recent (completed) build is broken.
Slave	Slaves are computers that are set up to build projects for a master. Jenkins runs a separate program called "slave agent" on slaves.When slaves are registered to a master, a master starts distributing loads to slaves.
Publisher	A publisher is part of the build process other than compilation, for example JUnit test runs. A publisher may report *stable* or *unstable* result depending on the result of its processing. For example, if a JUnit test fails, then the whole JUnit publisher may report *unstable*.
Completed Build	A build is completed, if it was started and finished with any result, including failed builds.

Jenkins- Basics



Jenkins- Basic overview

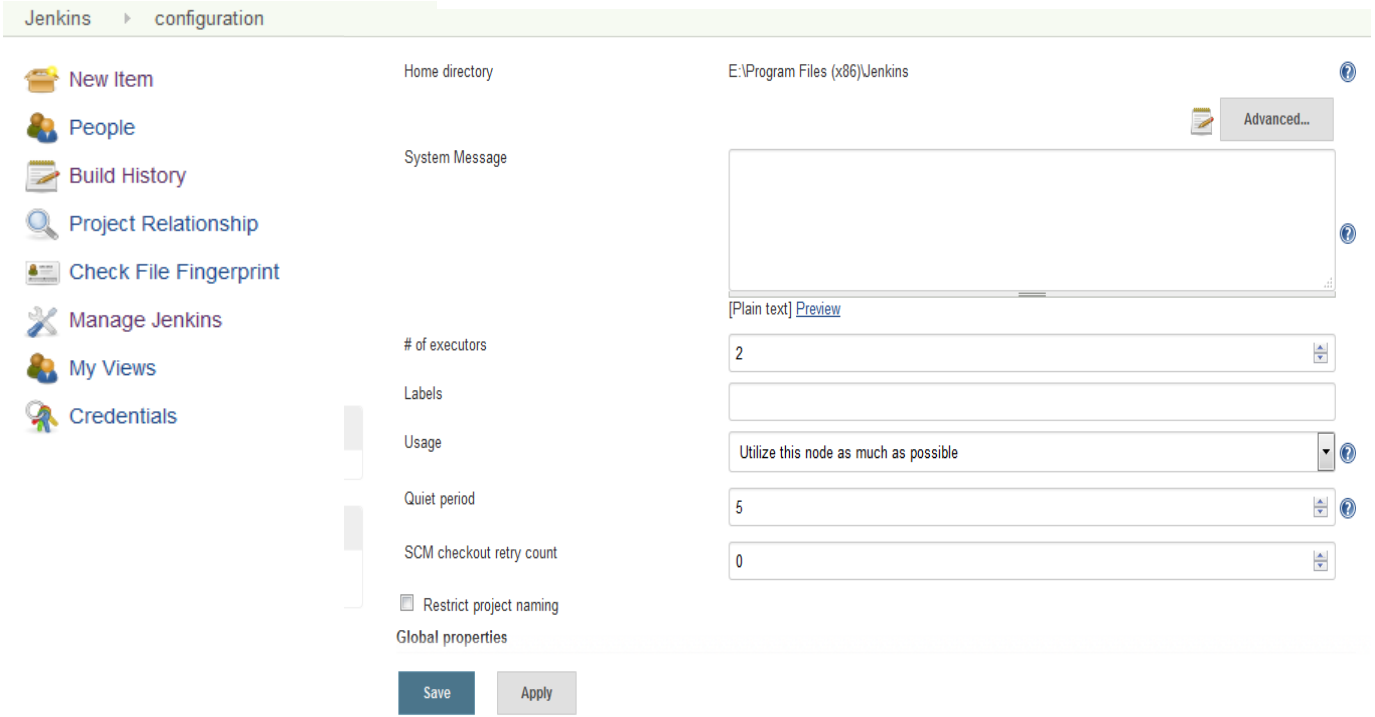
- Once Jenkins service is started, open any browser with localhost:8080
- Creating the job
  - New item can be used for creating a job or project
- Running the jobs
  - Build now option is used to build and run the jobs
- Viewing output
  - Results can be viewed in the console output format as well
- Jenkins dashboard
  - Dashboard provides the complete view of all the jobs and their status

Jenkins- Basic overview

- Disabling jobs
  - Disable project option can be chosen to disable the jobs
- Deleting a job
  - Delete project can be used for deleting a job
- Updating and adding plugins
  - Manage plug in option can be used for updating and adding plugins

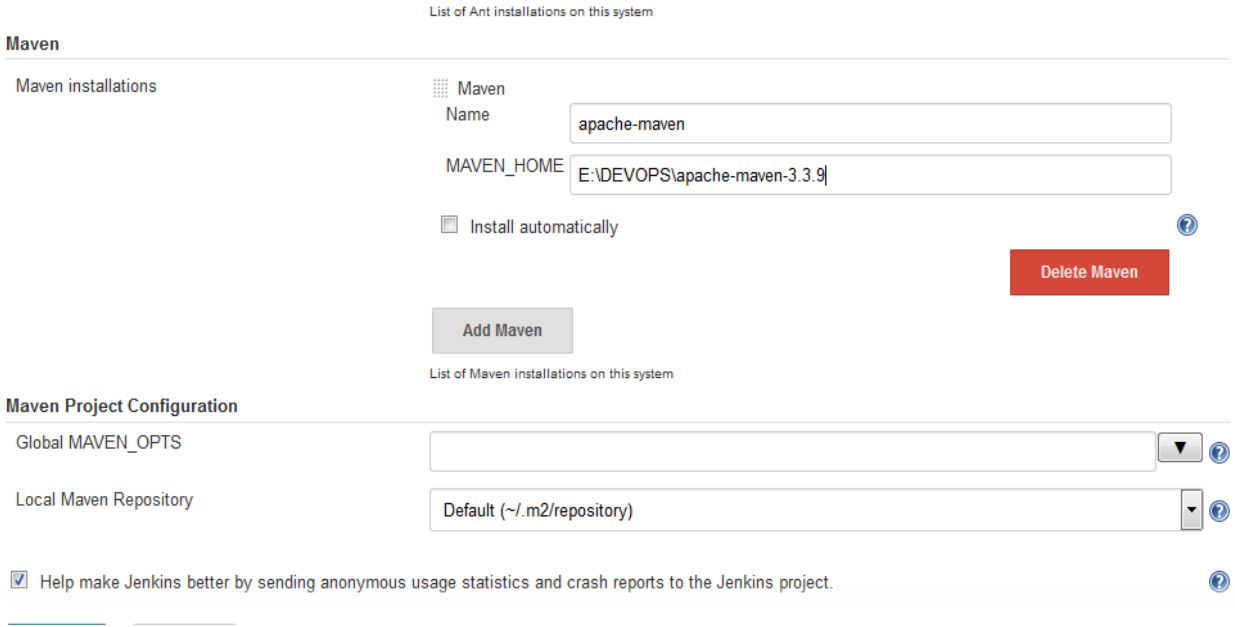
Jenkins configuration

- Manage Jenkins- configuration can be used



Jenkins-Maven

- Create a new project and set MAVEN_HOME variable



Build Tools

- **Java**
  - Maven (build-in), Ant, Gradle
- **.Net**
  - MSBuild, PowerShell
- **Shell script**
  - Python, Ruby, Groovy

Build Wrapper

- Build name (version no) setter
- Virtual machine (VMWare, Virtual Box)
- Set environment variable
- ClearCase release plugin

Build Notifier

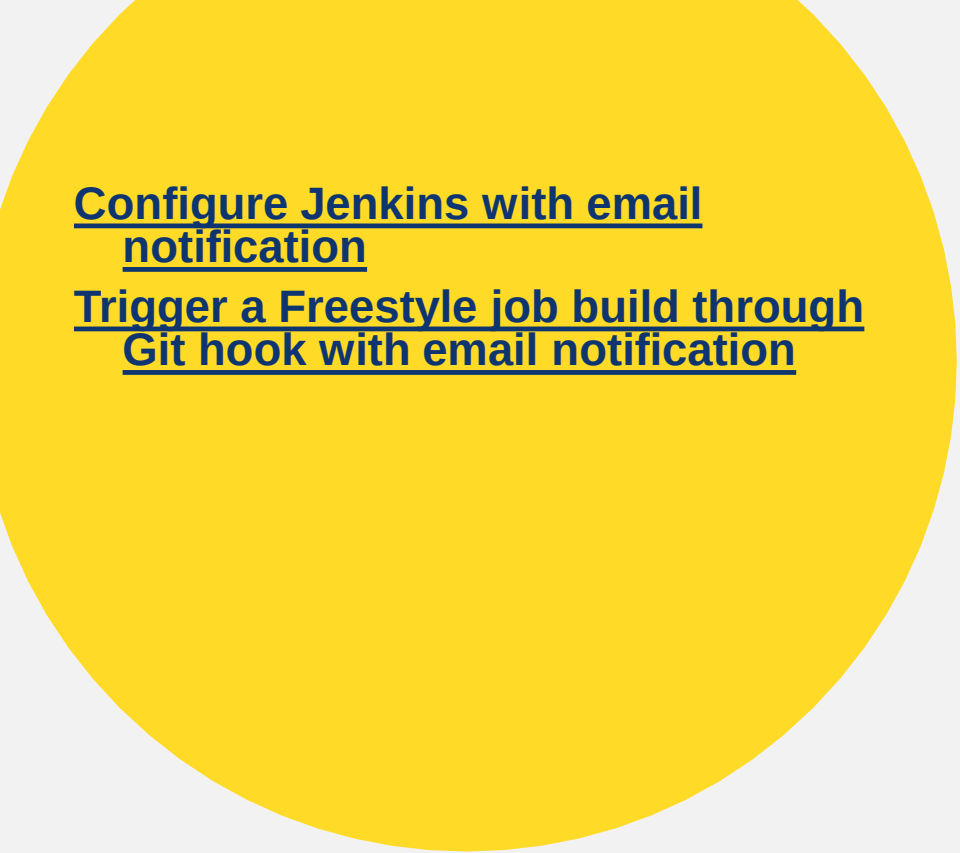
- The various options are :
  - E-mail
  - Twitter
  - Jabber
  - IRC
  - RSS
  - Google calendar

Build Report

- **Static Code Analysis**
  - Checkstyle, PMD, Findbugs, Compiler Warning
- **Test Report & Code Coverage**
  - JUnit, TestNG, Cobertura, Clover
- **Open Tasks**

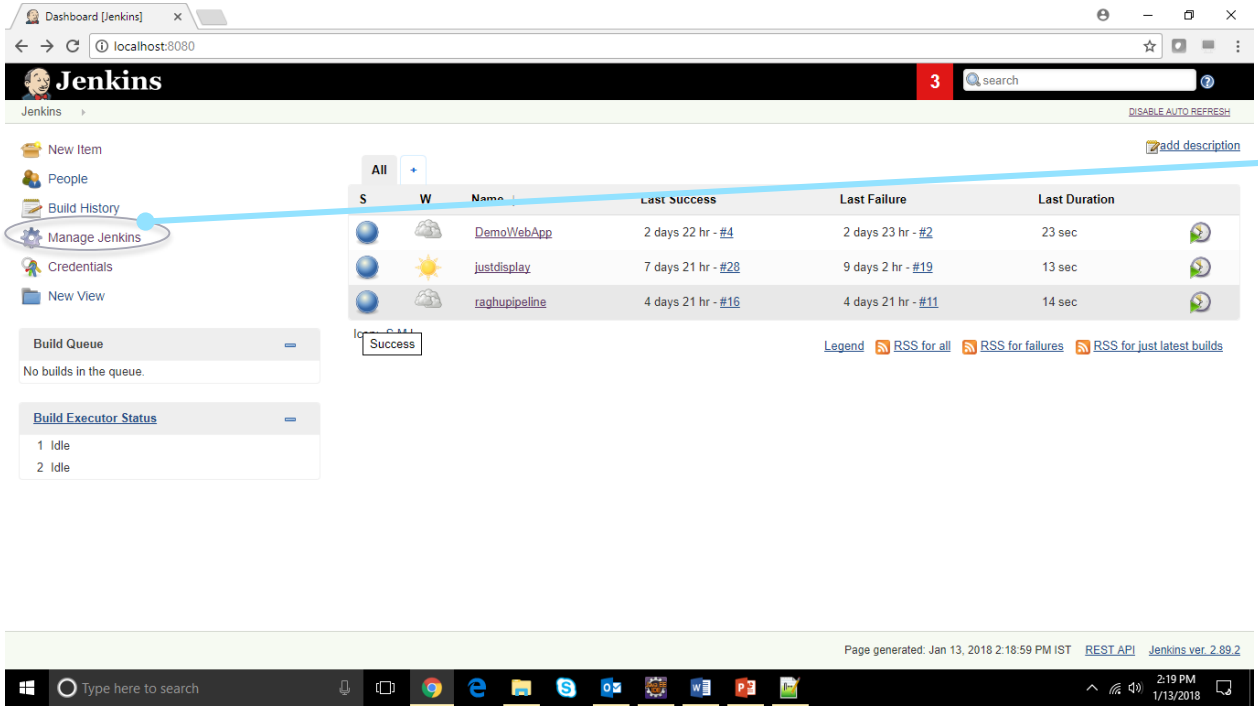
Configure Jenkins with email notification

Trigger a Freestyle job build through Git hook with email notification



Configure Jenkins

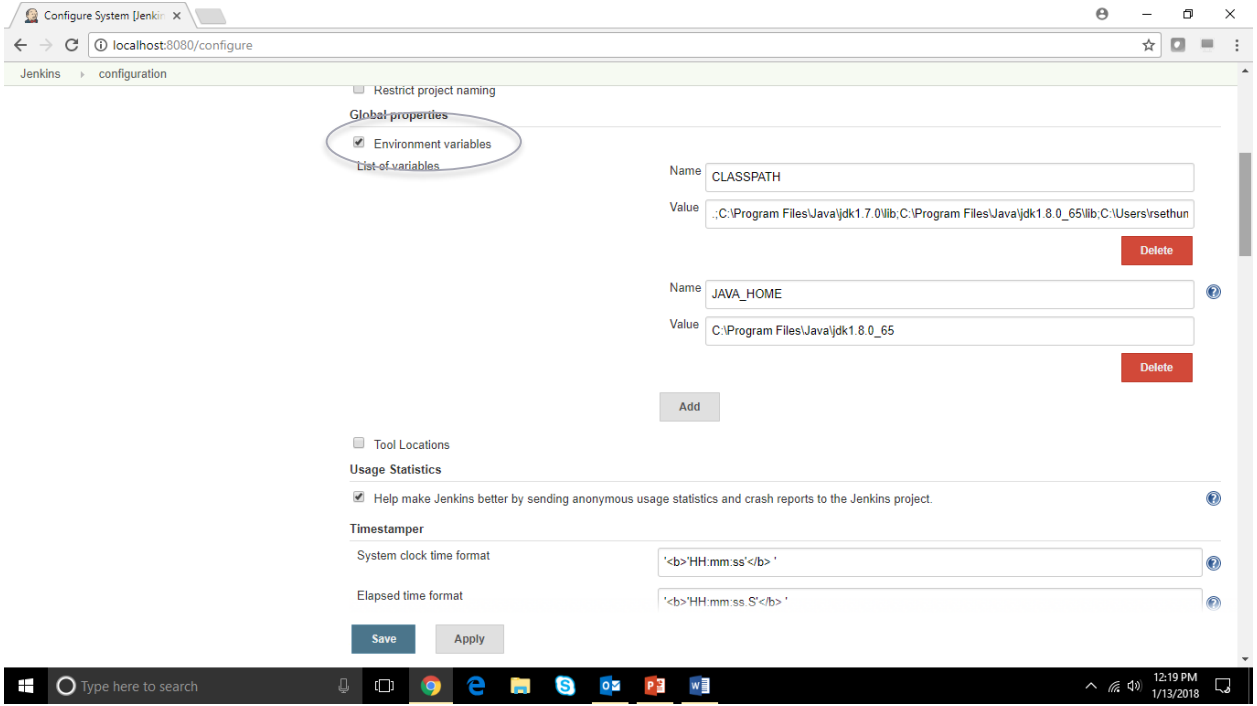
- **Configure Jenkins**



Click On Manage Jenkins

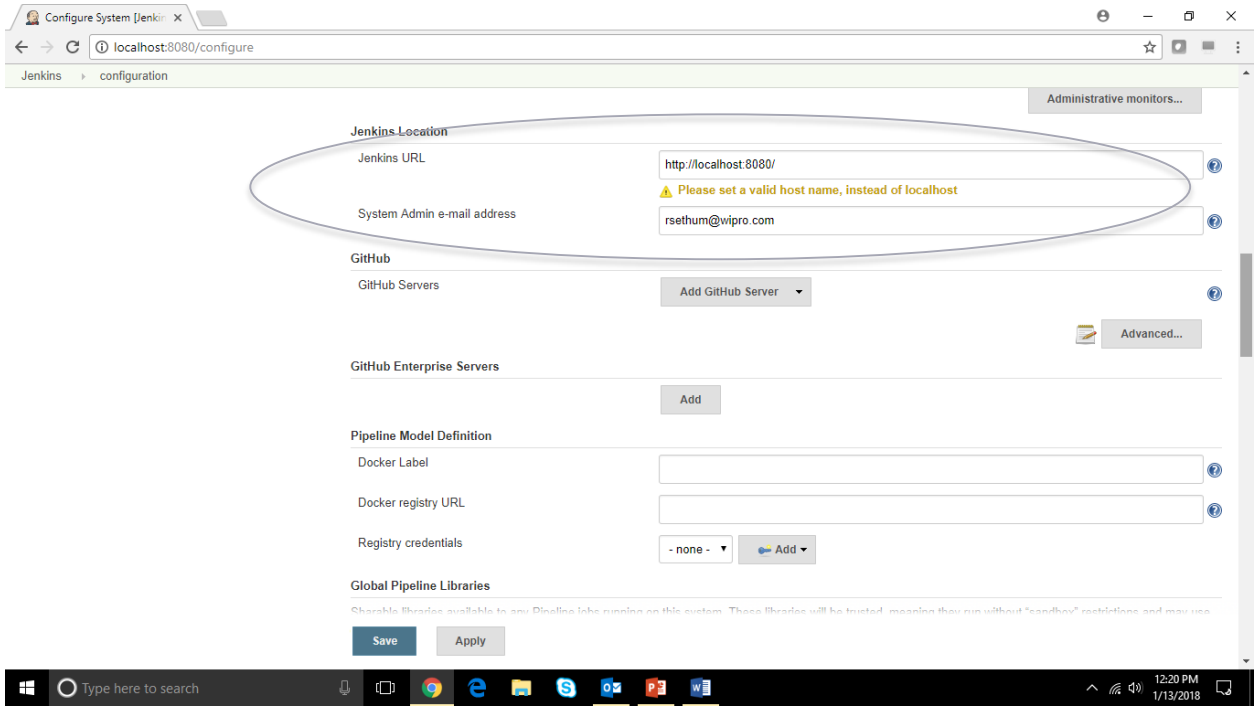
Configure Jenkins

- **Configure Environment Variables like CLASSPATH , JAVA_HOME in Jenkins**



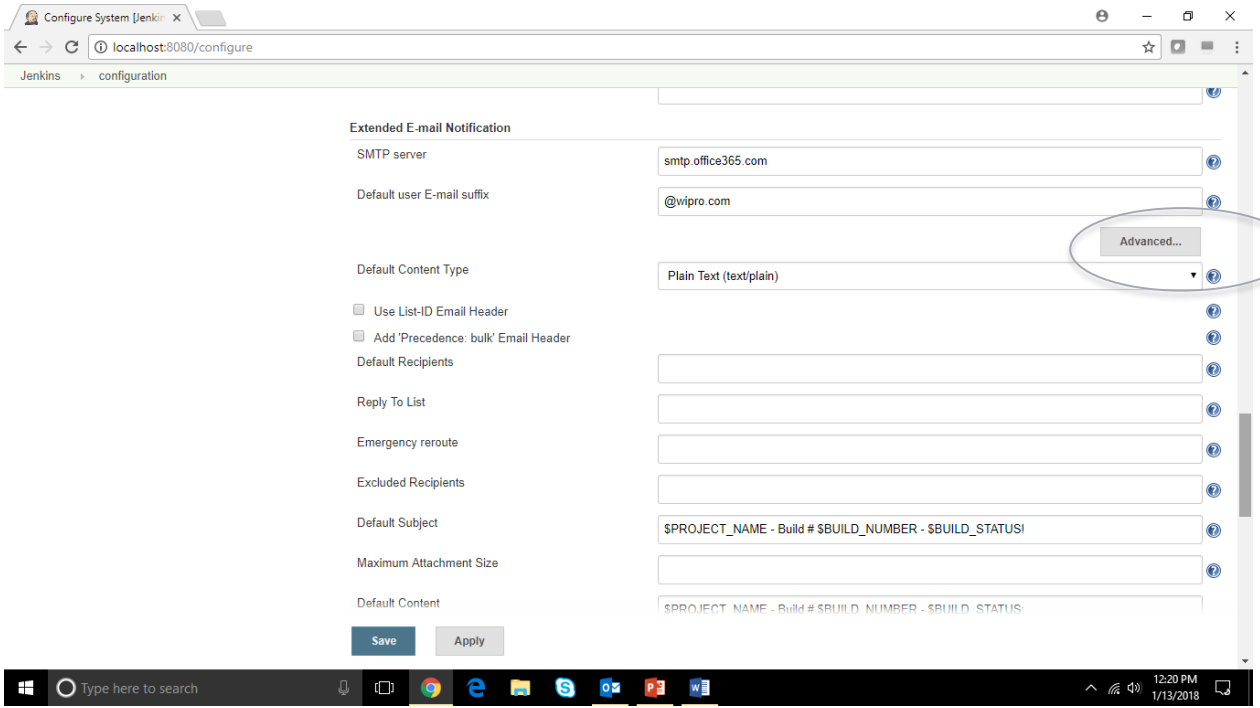
Configure Jenkins

- **Steps to define E-mail Notification:**



Configure Jenkins

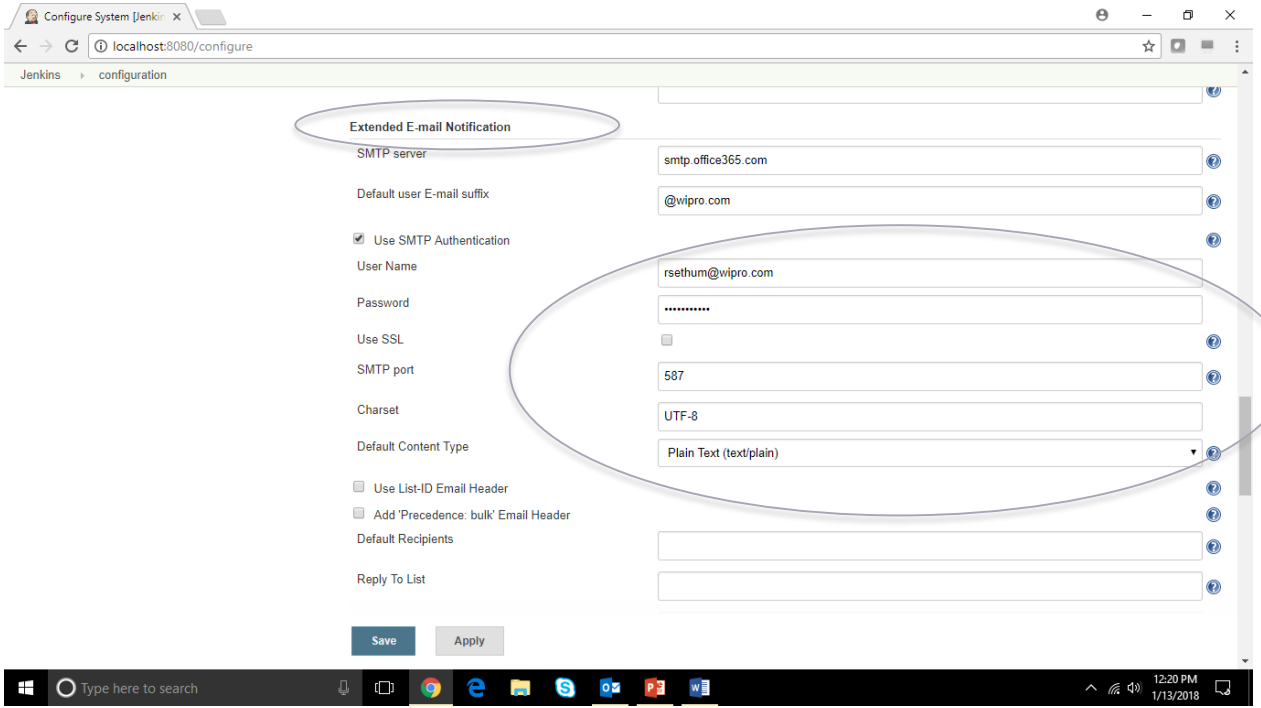
- **Steps to define E-mail Notification:**



Click on **Advanced** after entering SMTP server details

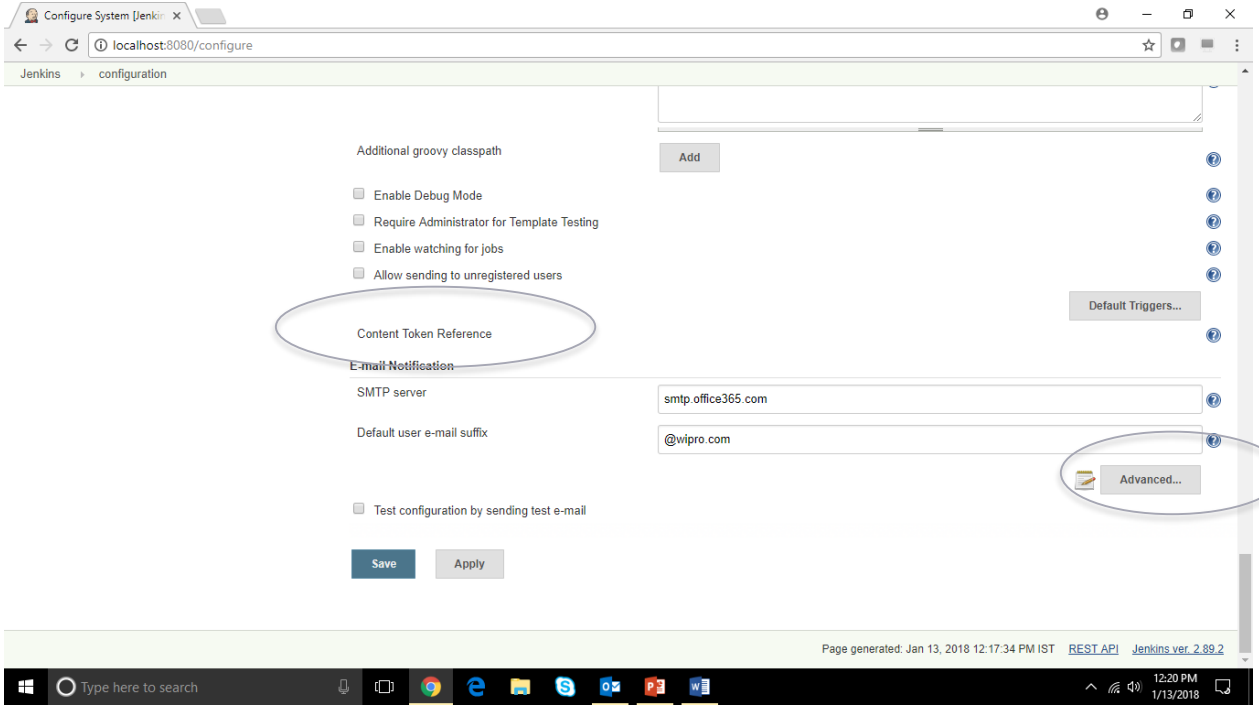
Configure Jenkins

- **Steps to define E-mail Notification:**



Configure Jenkins

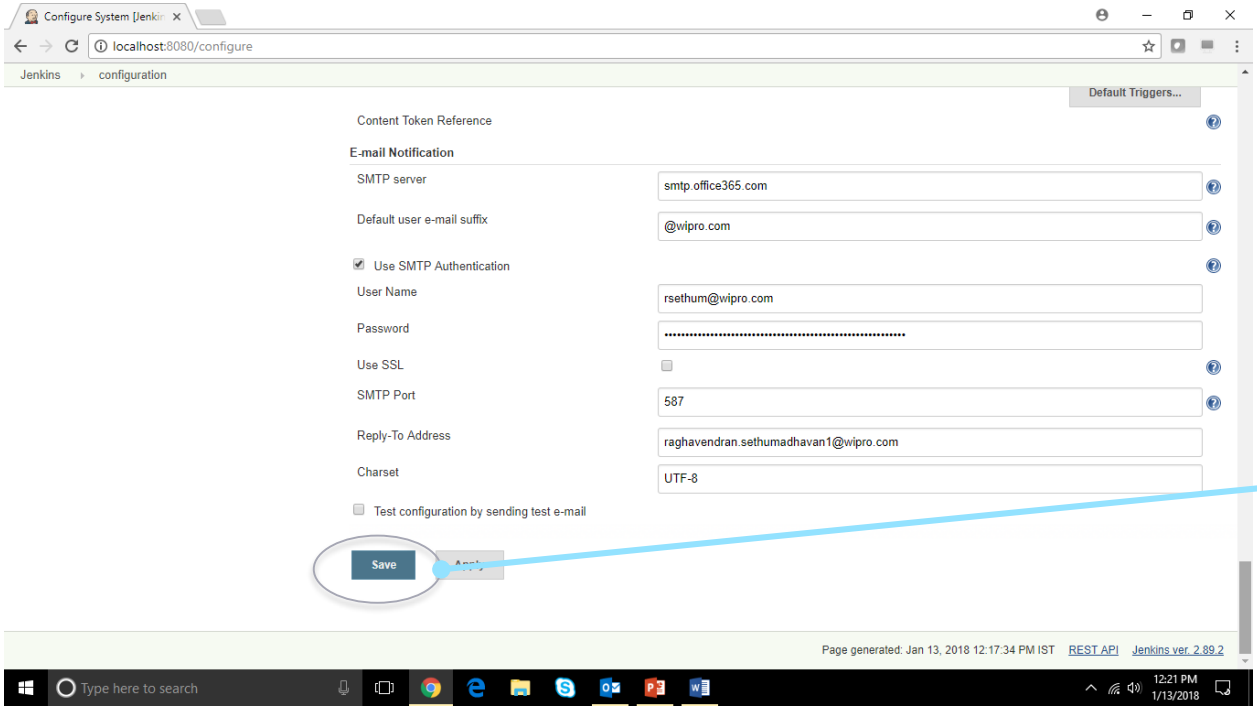
- **Steps to define E-mail Notification:**



Click on **Advanced** after entering SMTP server details

Configure Jenkins

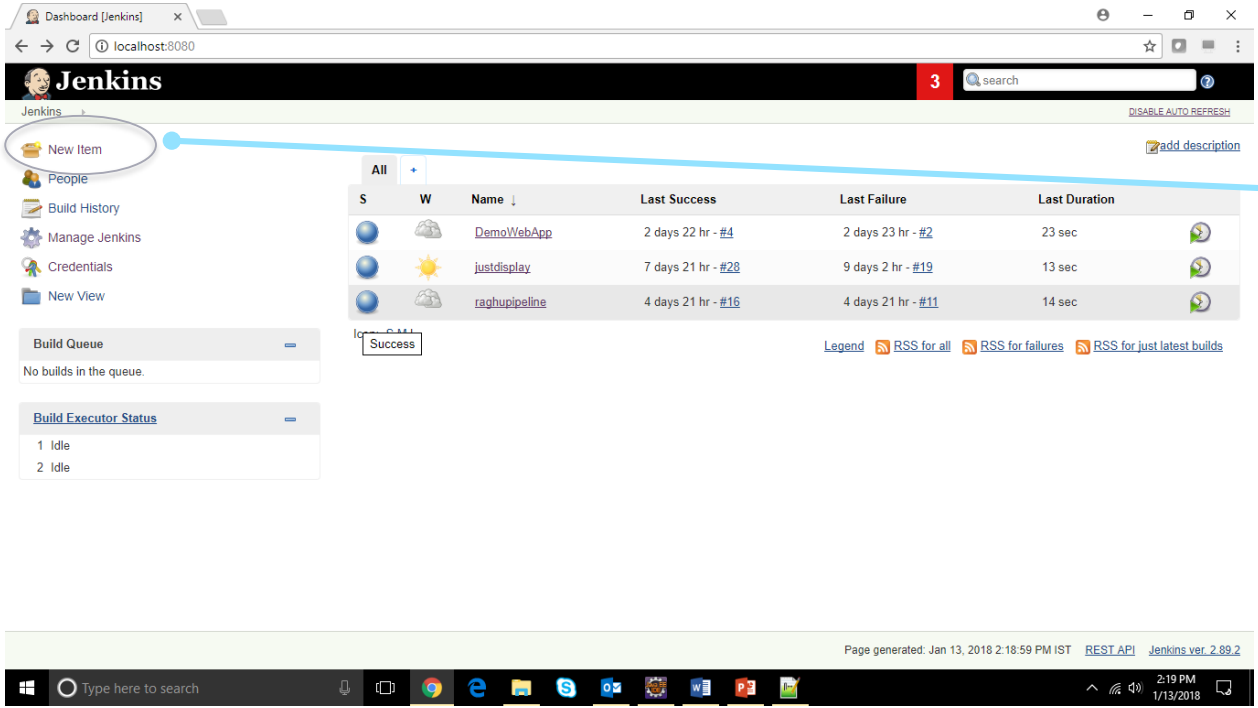
- **Steps to define E-mail Notification:**



Click on **Save** and complete the basic configuration

Create Jenkins Freestyle Job

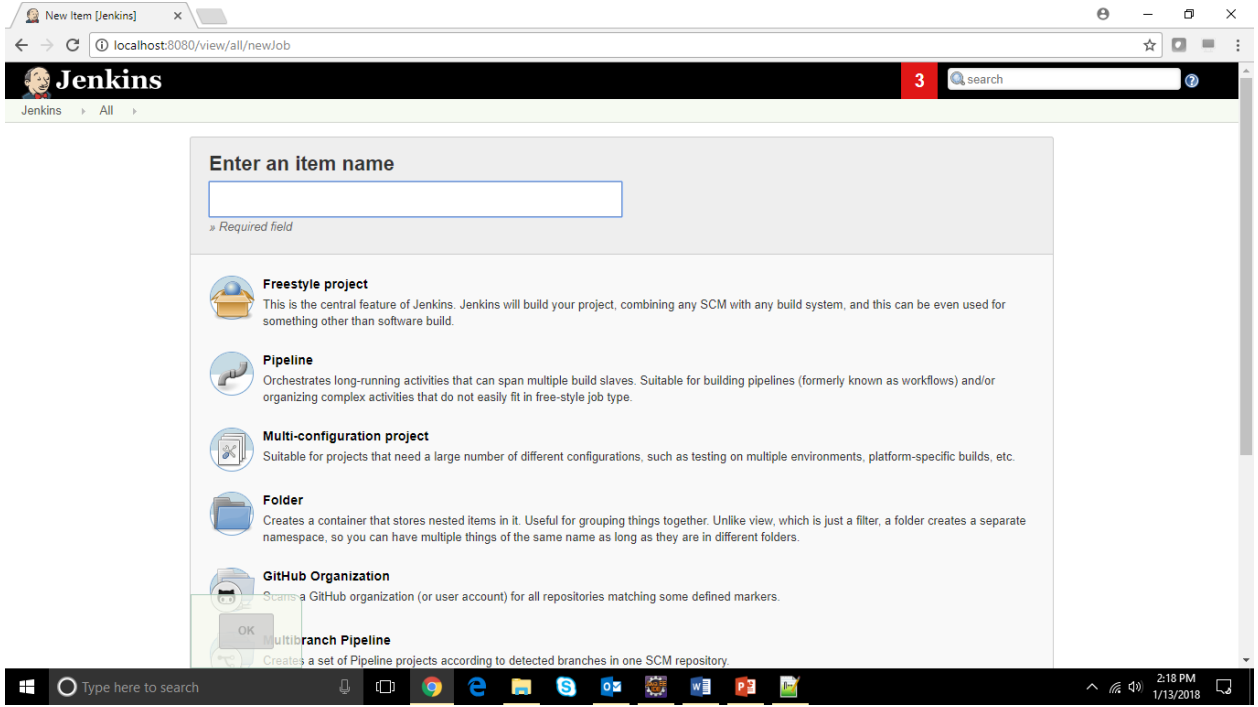
Create a New Job in Jenkins



Click on New Item to create a job

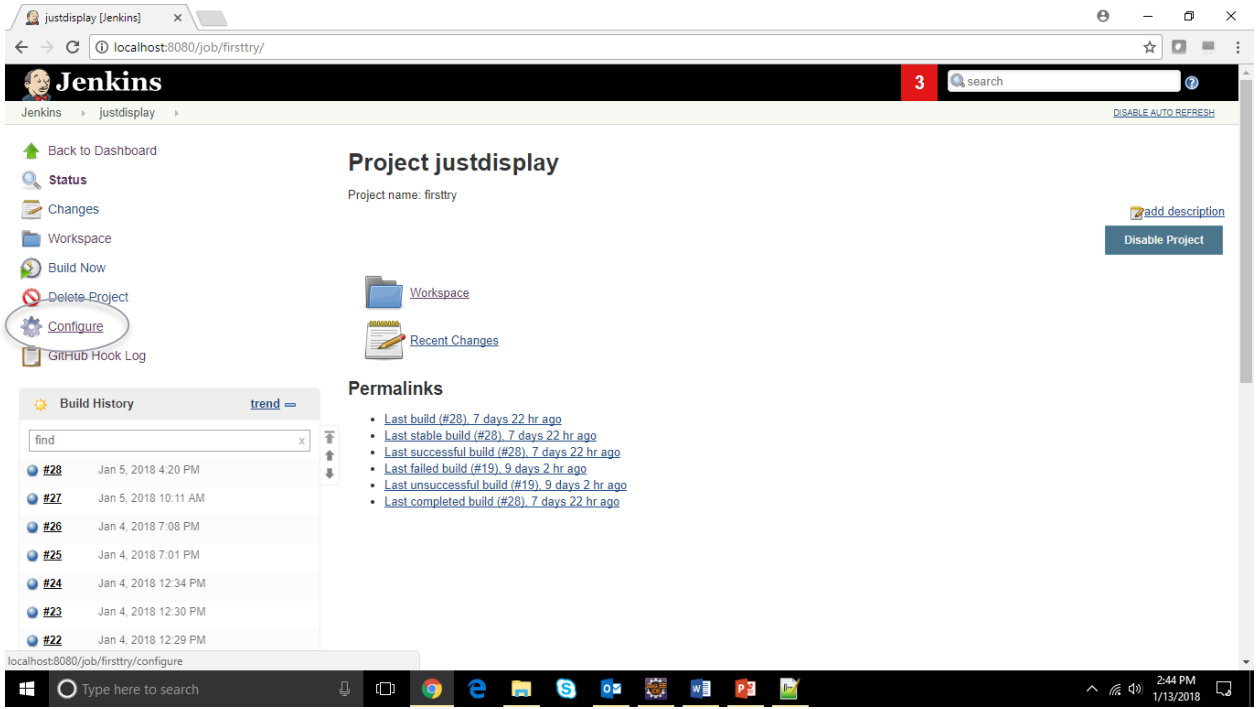
Create Jenkins Freestyle Job

Create a New Job



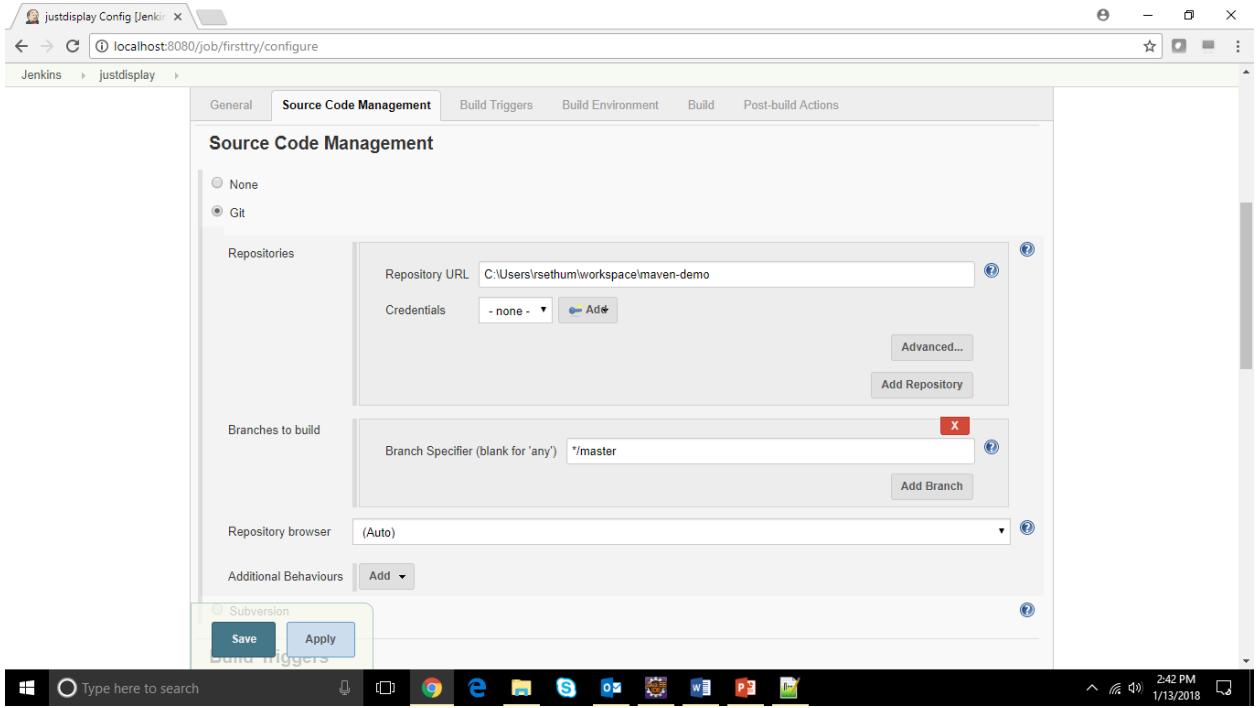
Configure Jenkins Freestyle Job

Configure a new Job



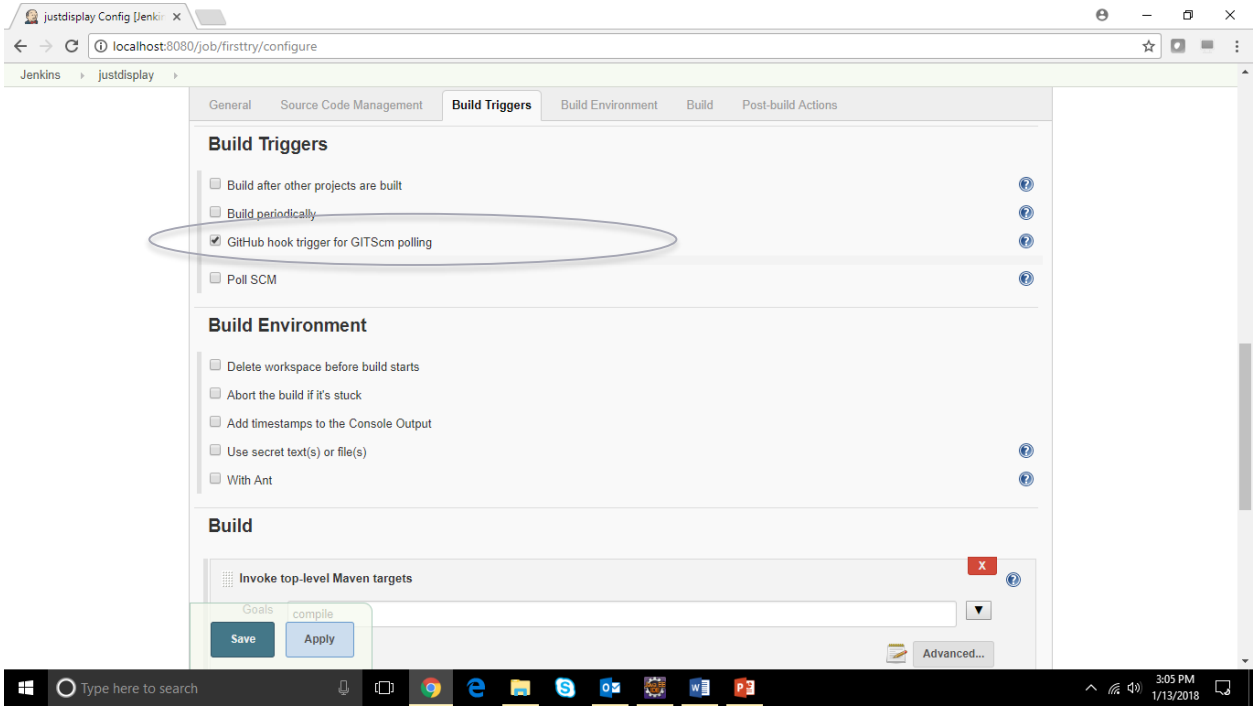
Configure Jenkins Freestyle Job

Declare that the source will be pulled from a Git repository



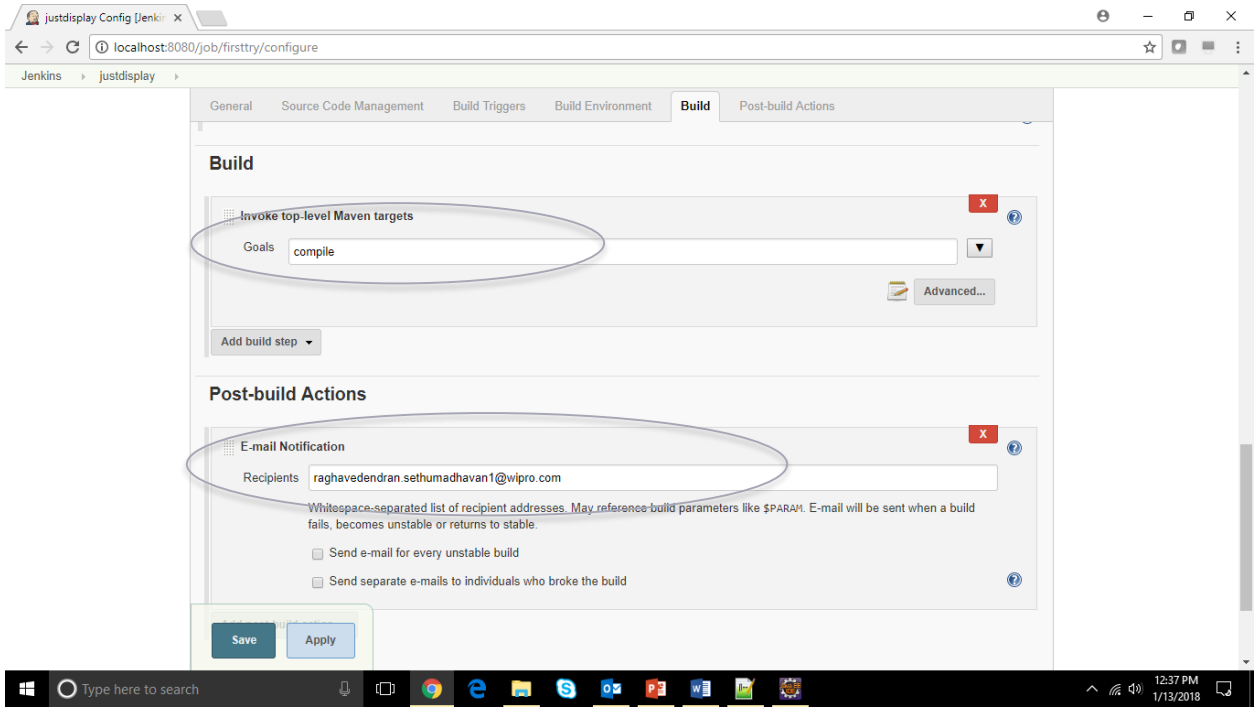
Configure Jenkins Freestyle Job

Declare that the source will be pulled from a Git repository



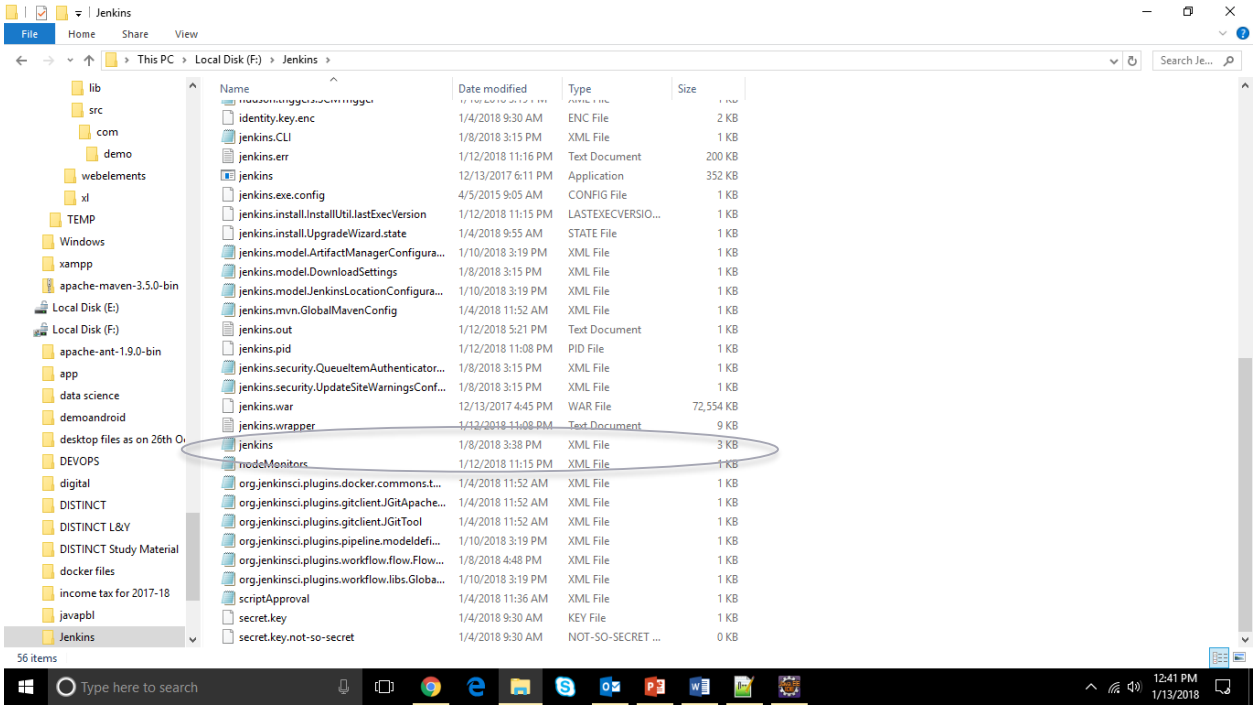
Create Jenkins Freestyle Job

- **Configure Build using top level Maven goal with customizable email notification trigger at Job Level:**



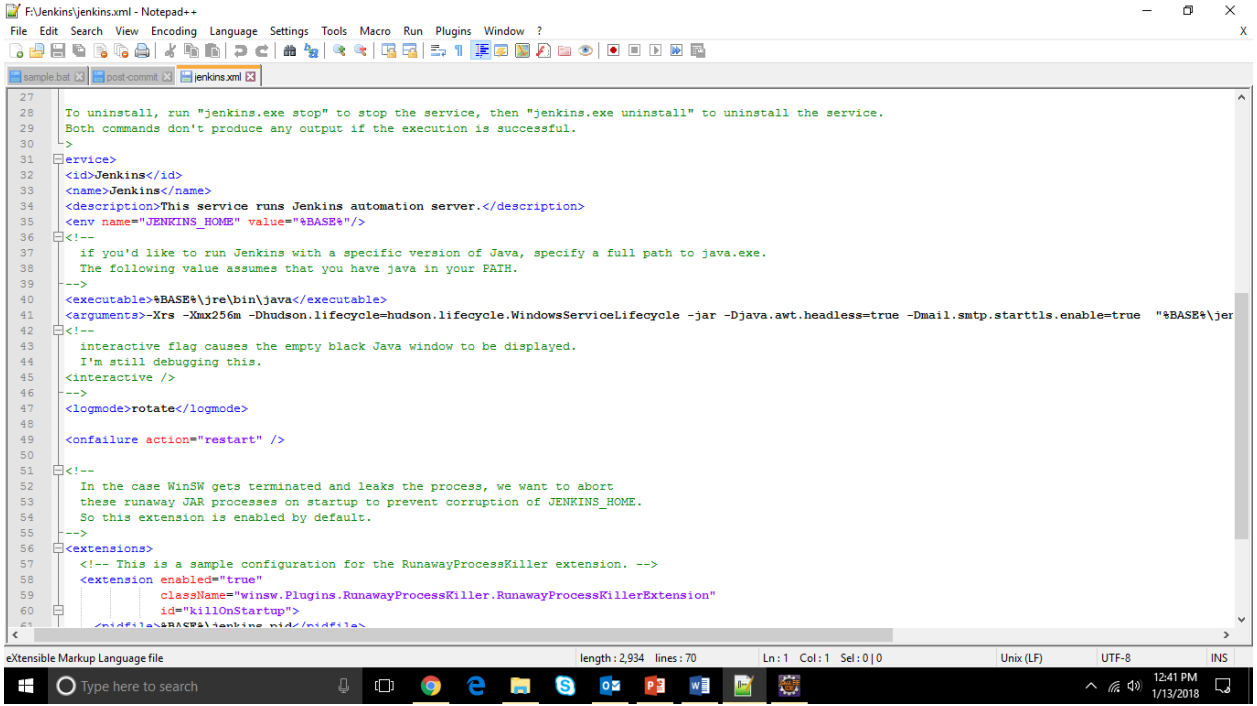
Update Jenkins.xml

- **Make Changes in Jenkins.xml :**



Update Jenkins.xml

- **Make Changes in Jenkins.xml :**

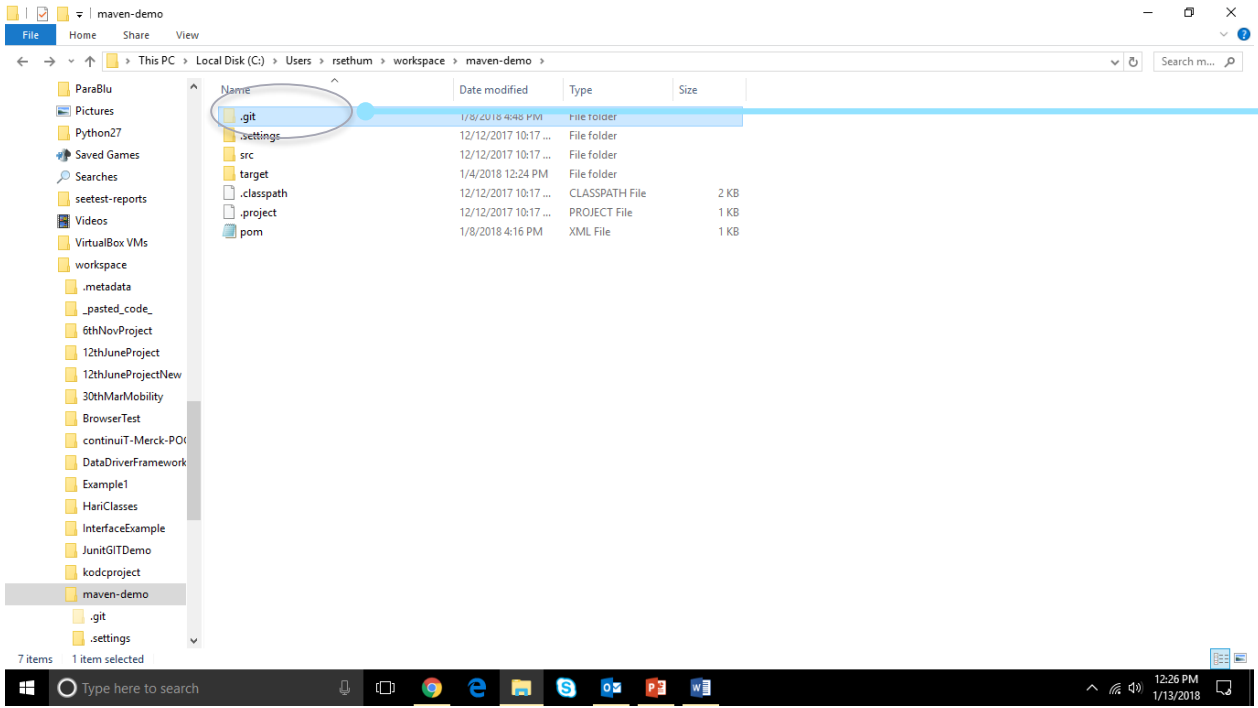


Add the below two parameters in <Arguments> tag as indicated:

-Djava.awt.headless=true

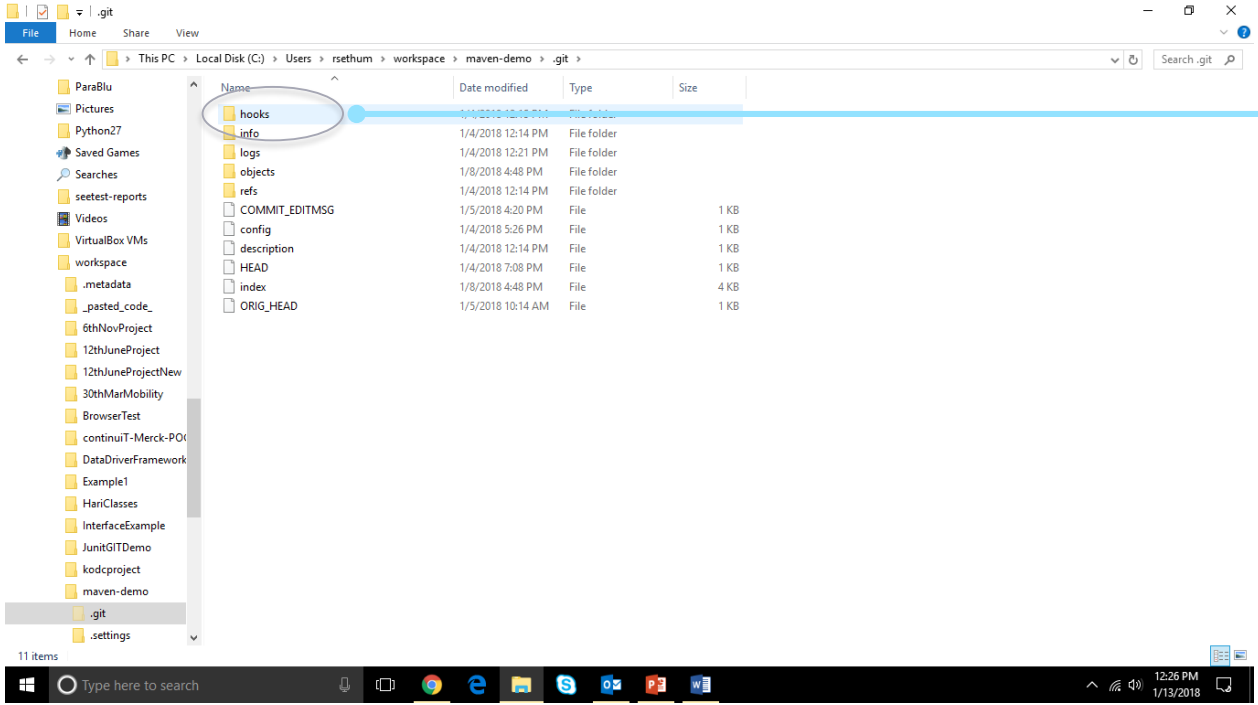
-Dmail.smtp.starttls.enable=true

Create a hook in Git repository



Click on .git folder

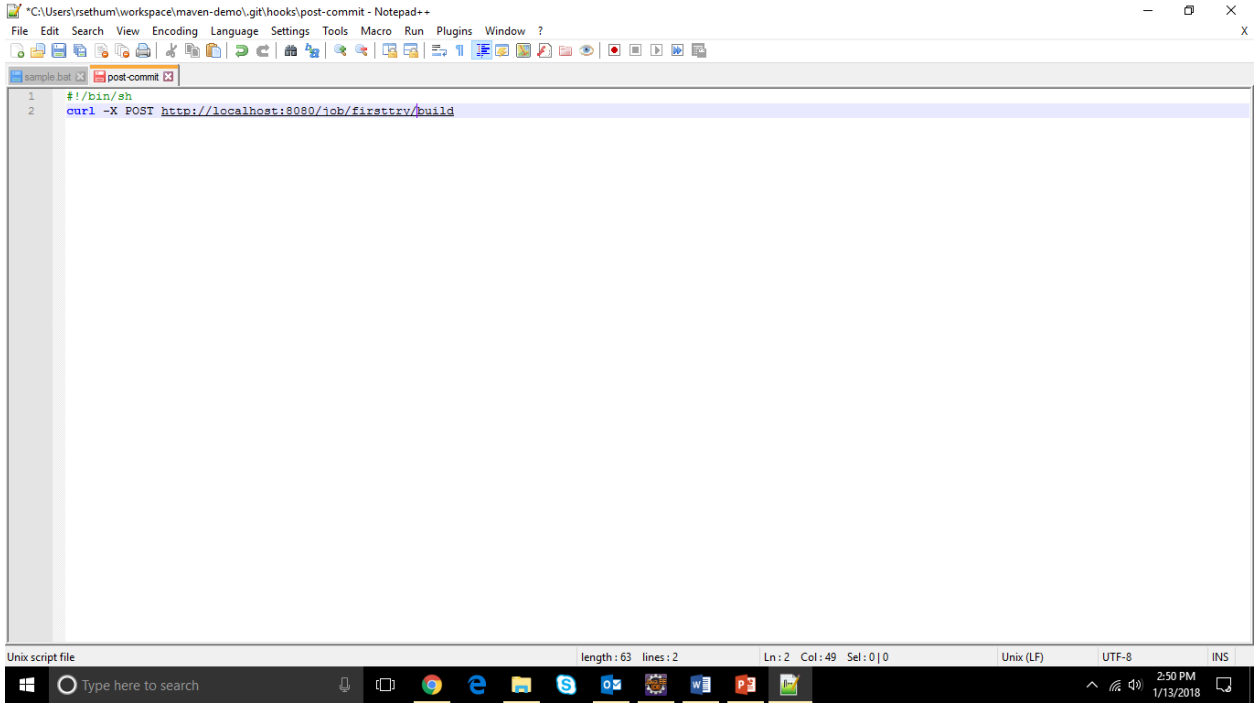
Create a hook in Git repository



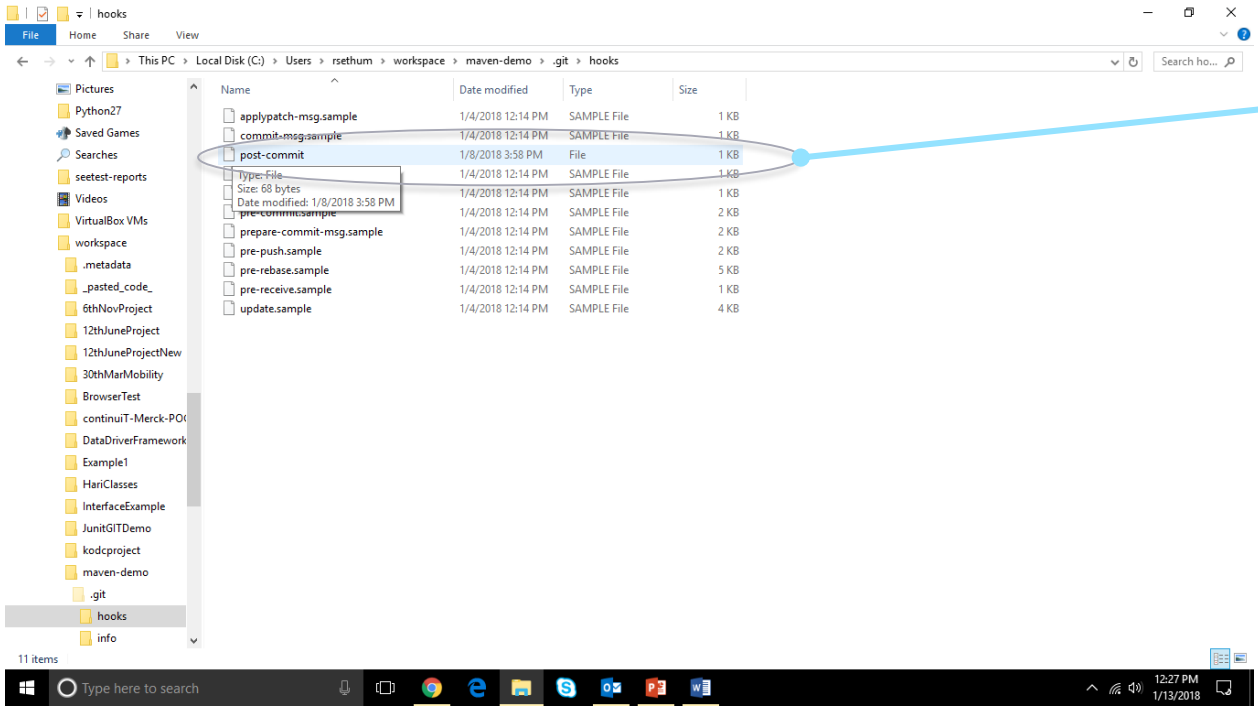
Click on hooks folder

Create a hook in Git repository

- **Create a post-commit file**



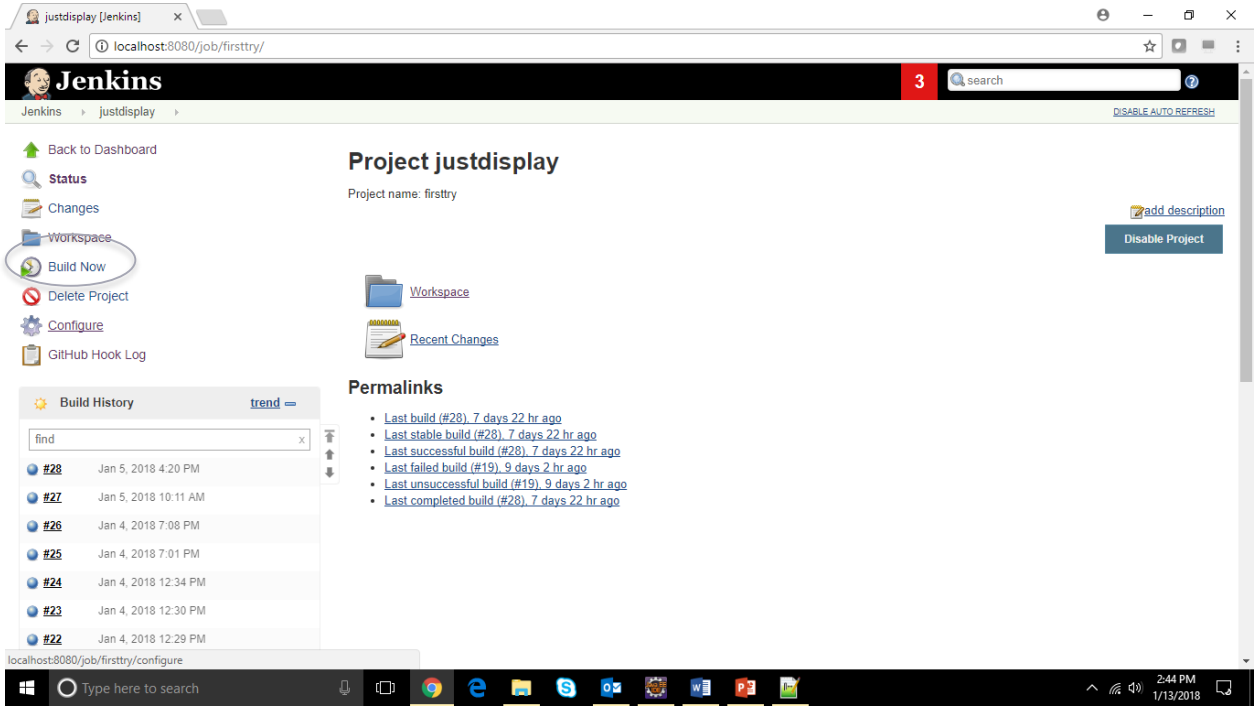
Create a hook in Git repository



Create a post-commit file

Auto trigger of build by commiting changes in Git

- **After the configuration, by committing changes in Git repository, the build will be auto triggered as define**

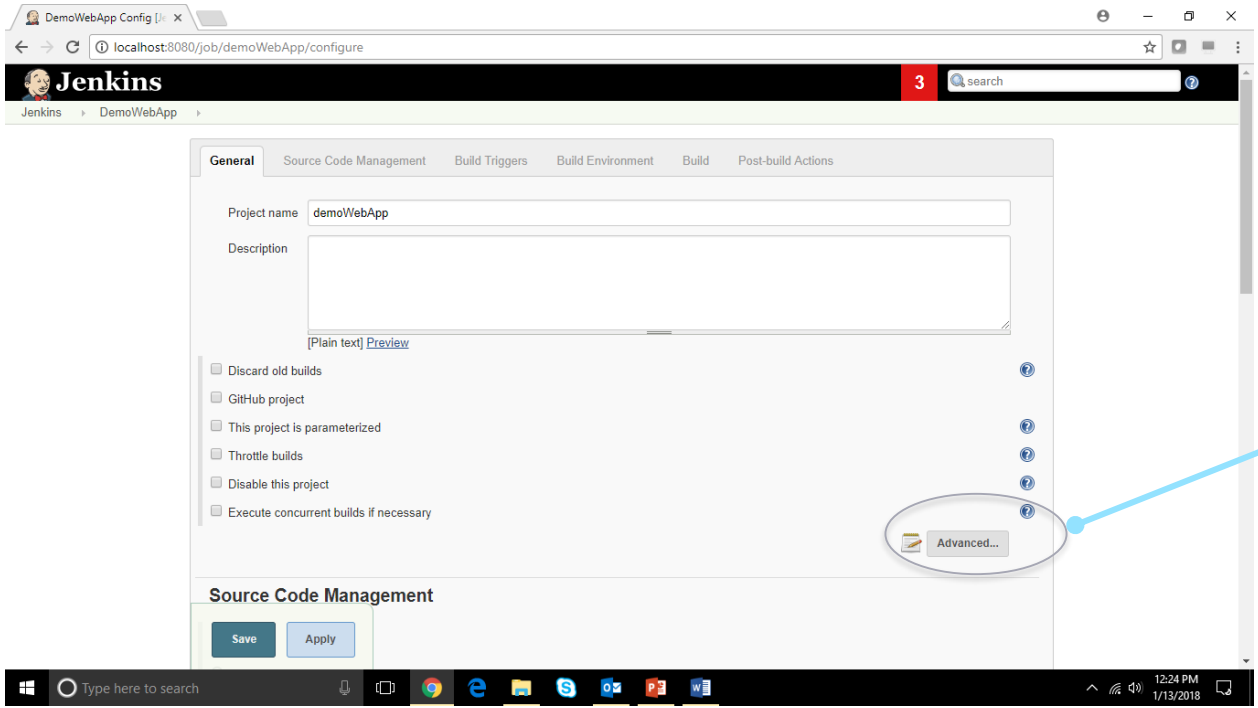


Build a selenium test job using batch file




Build a selenium test with a batch file

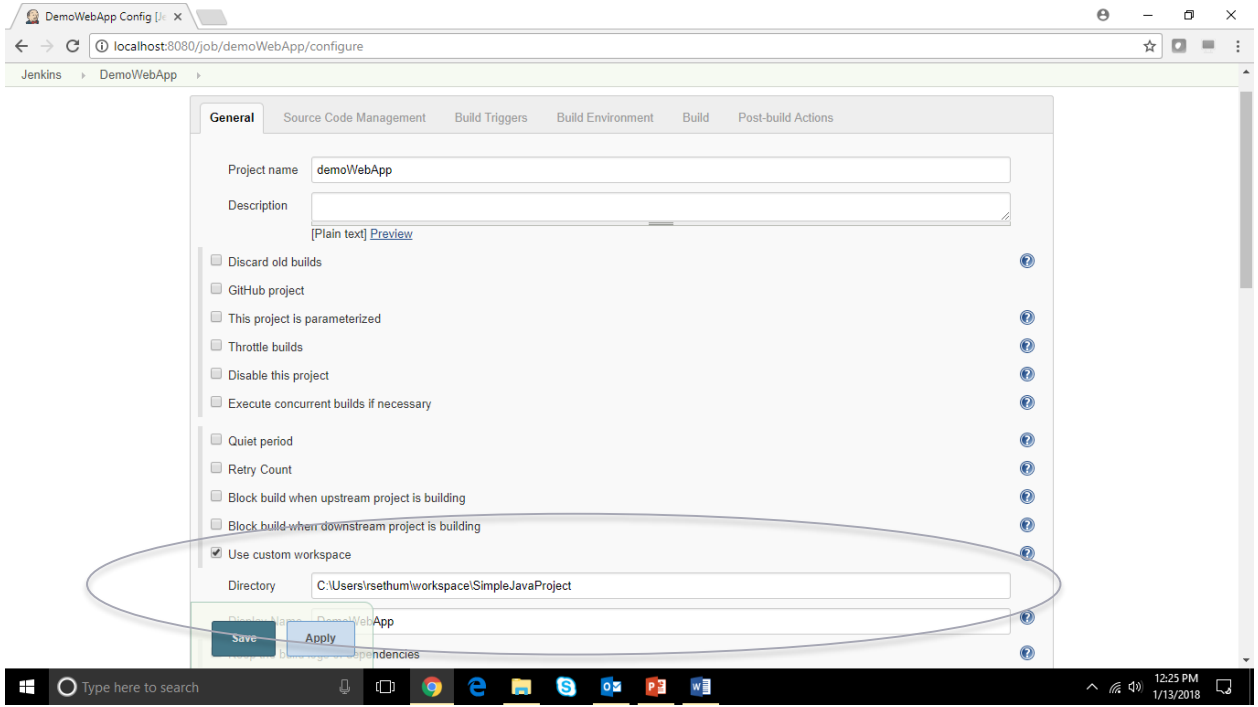
- **Declare a customer workspace in a Freestyle job newly created:**



Click on Advanced

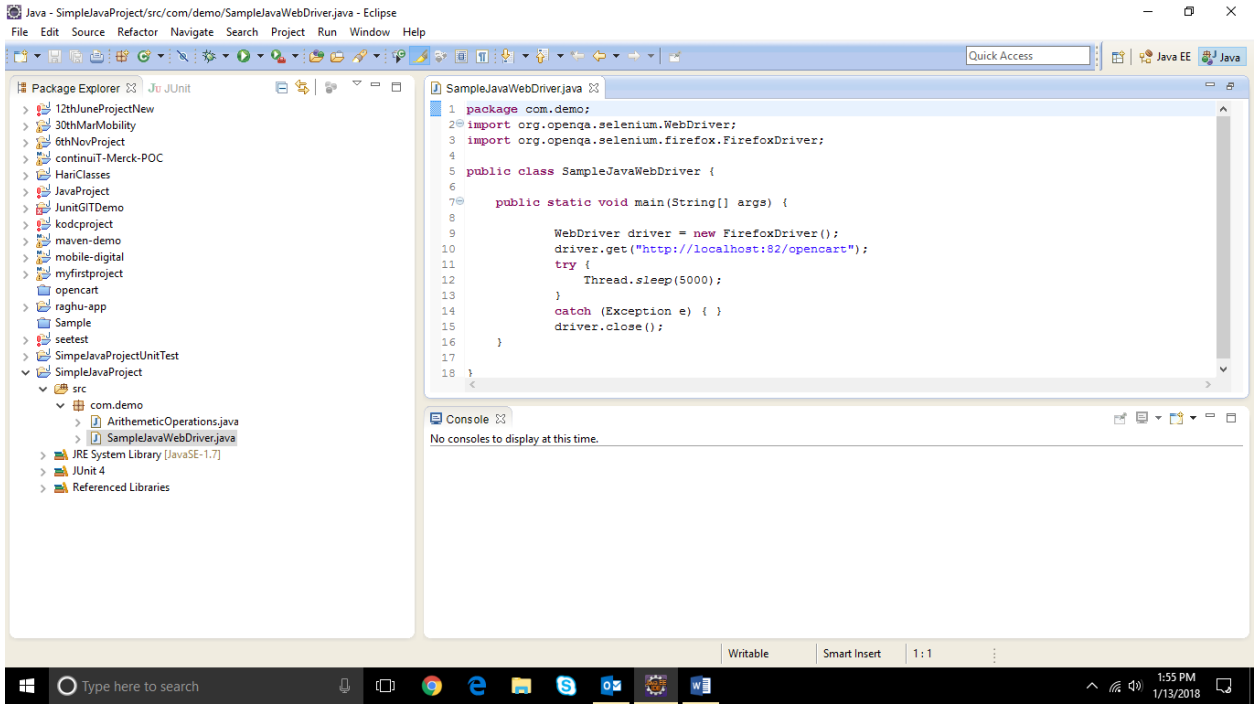
Build a selenium test with a batch file

- **Declare a customer workspace:**



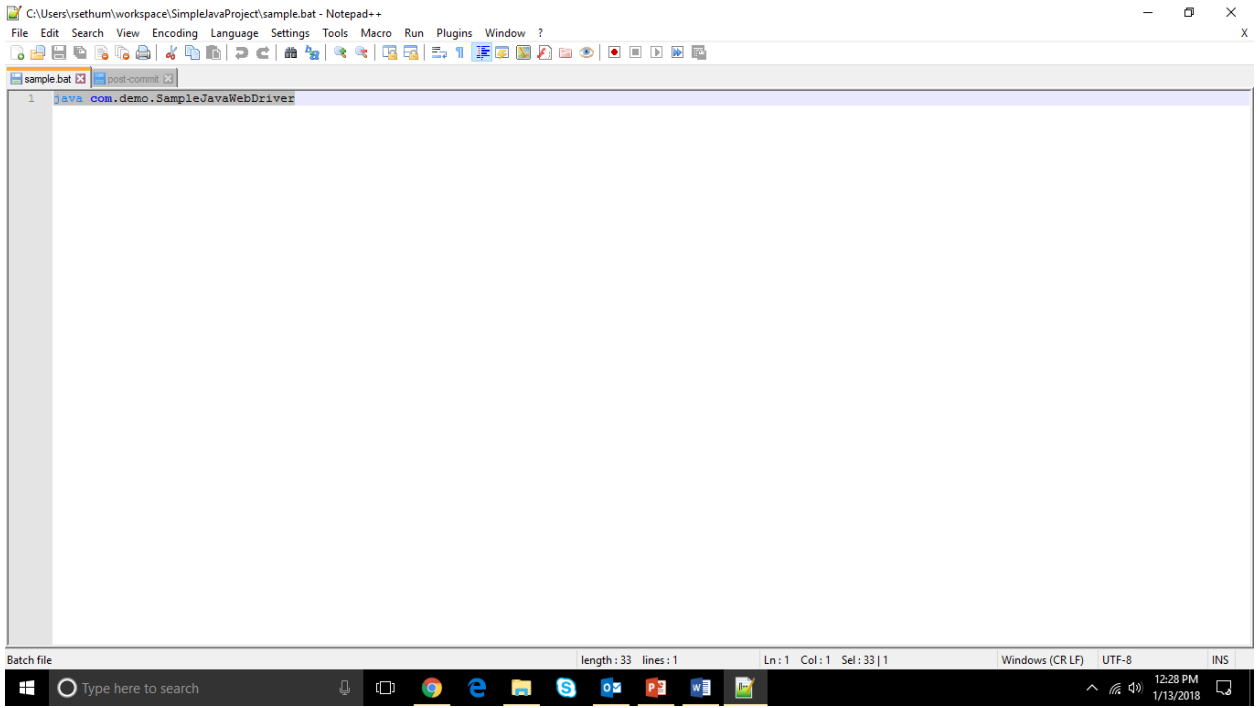
Build a selenium test with a batch file

- **Write a selenium test in Eclipse IDE**



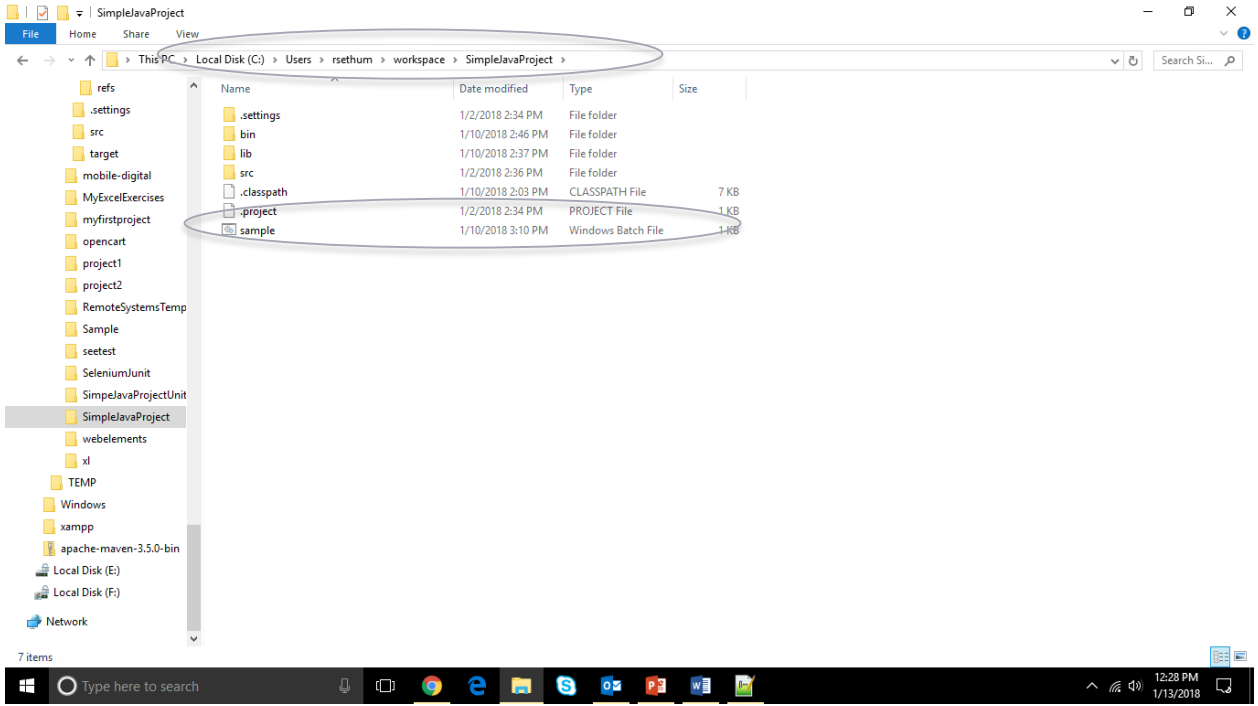
Build a selenium test with a batch file

- **Write a batch file**



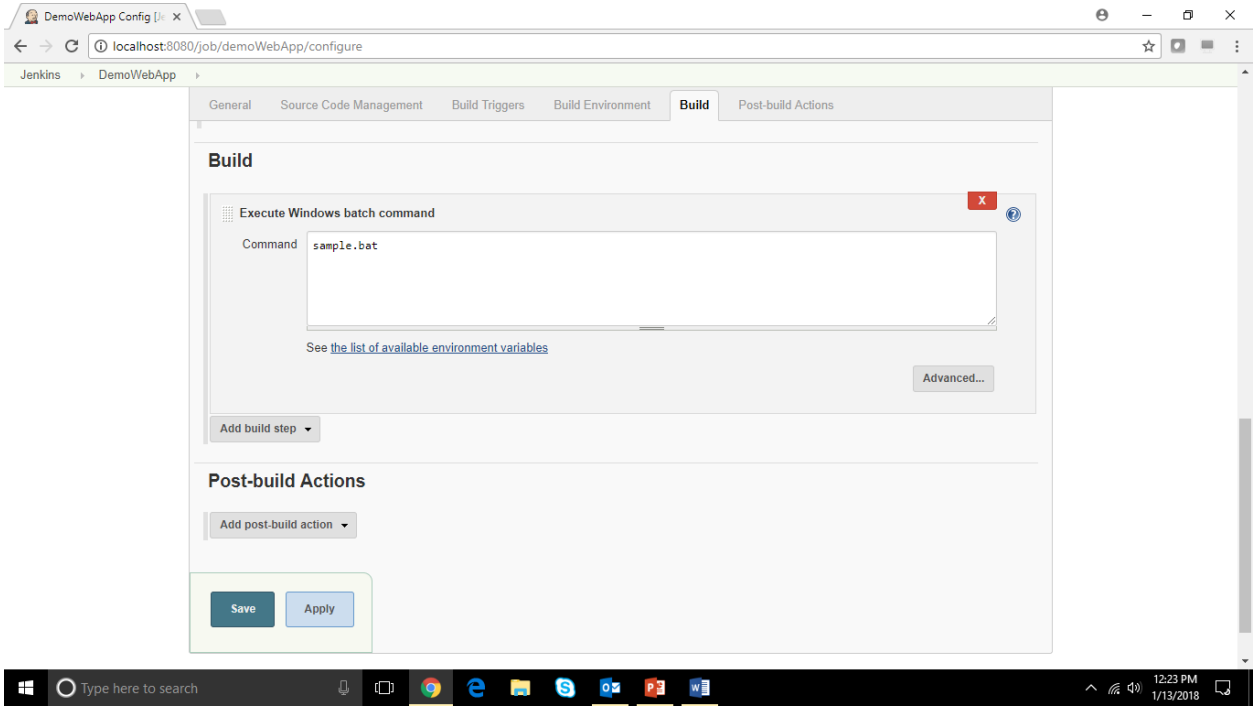
Build a selenium test with a batch file

- **Store the batch file in the project folder defined as customer workspace**



Build a selenium test with a batch file

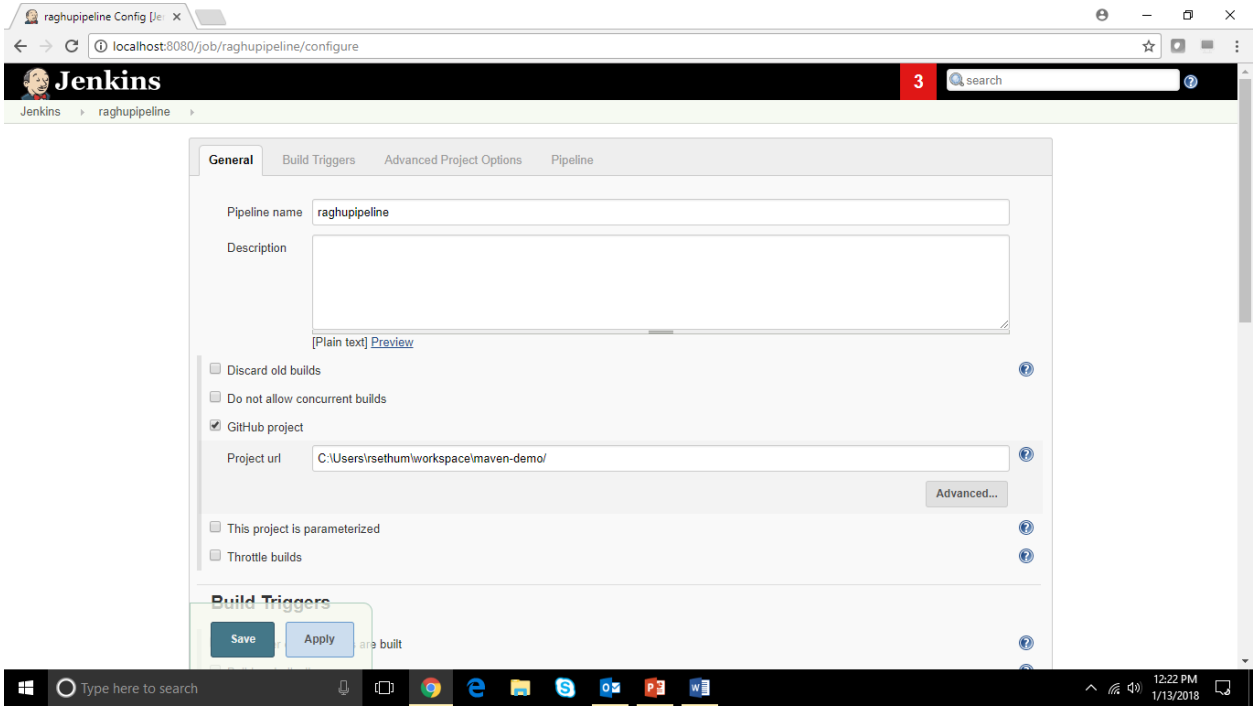
- **In the build section of the job, mention the name of the batch file created and then perform Build now operation in Jenkins.**



Create Jenkins Pipeline Job

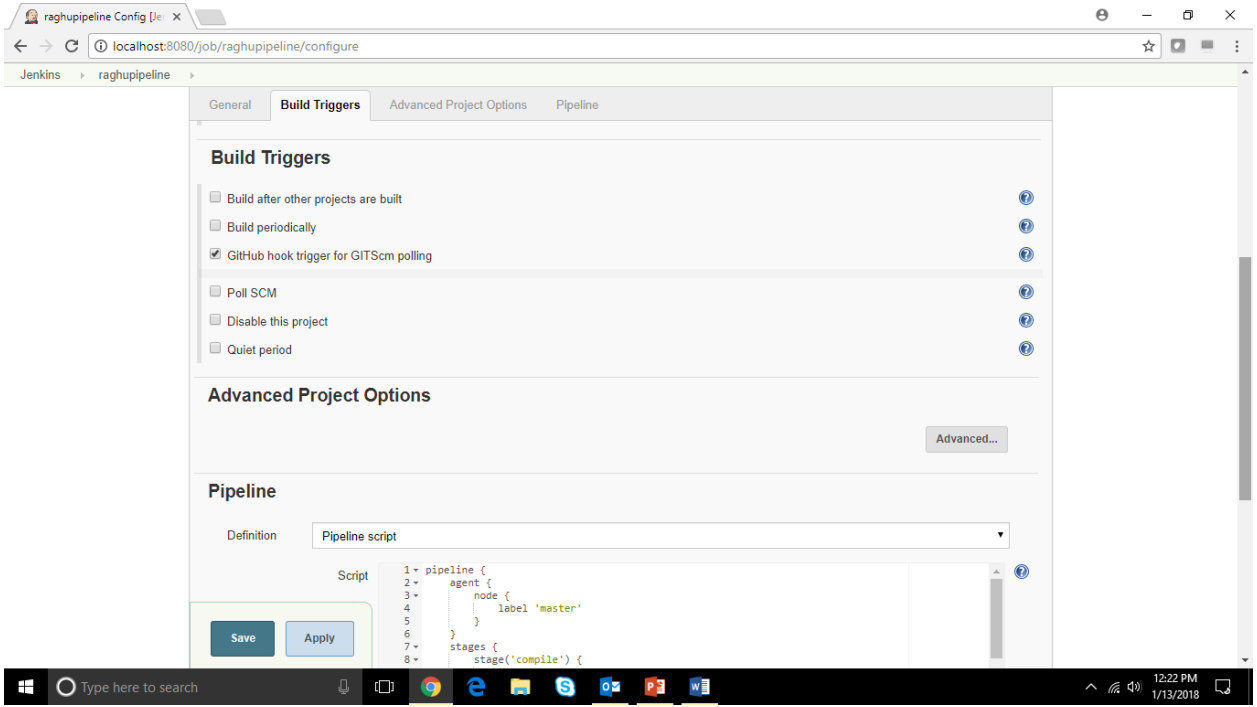
Create Jenkins Pipeline Job

- **Create a pipeline job in jenkins**



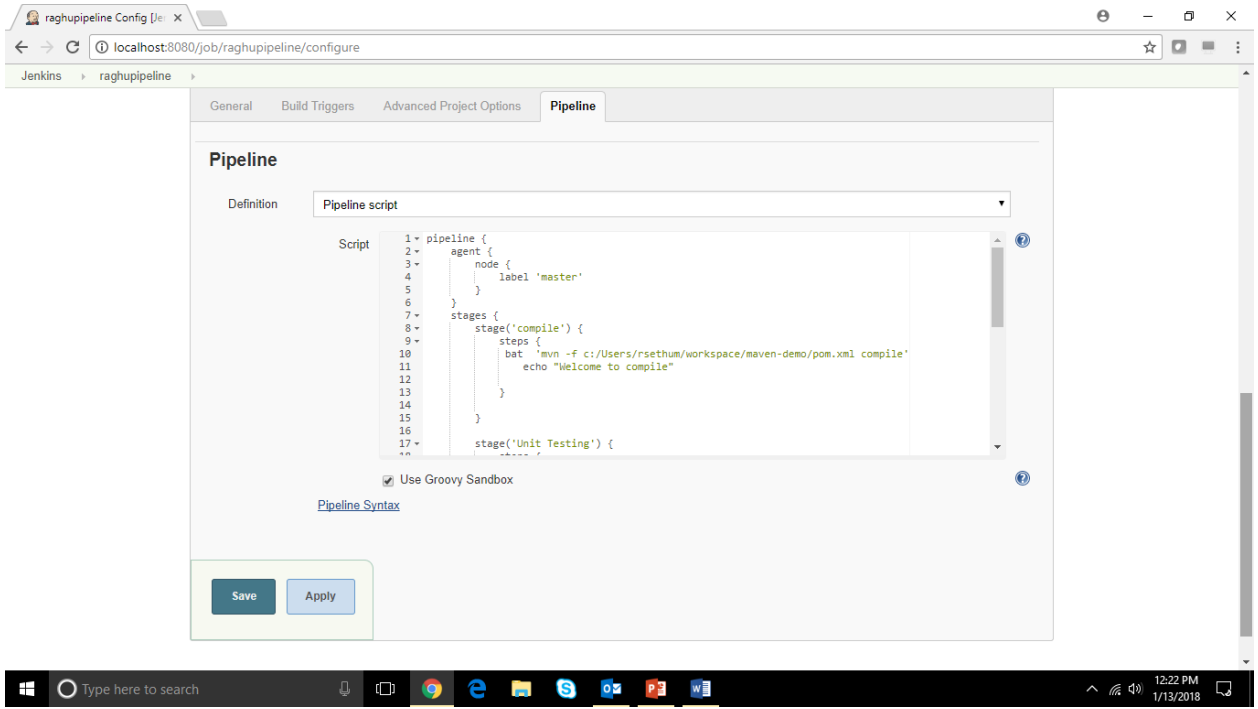
Create Jenkins Pipeline Job

- **Configure the job to perform build whenever there is a new commit in Git repository**



Create Jenkins Pipeline Job

- **Write a simple Groovy script**



Create Jenkins Pipeline Job

- **Write a groovy script (Continued)**

```
pipeline {
        agent {
            node {
                label 'master'
            }
        }
        stages {
            stage('compile') {
                steps {
                 bat  'mvn -f c:/Users/rsethum/workspace/maven-demo/pom.xml compile'
                    echo "Welcome to compile"
                }
            }
```

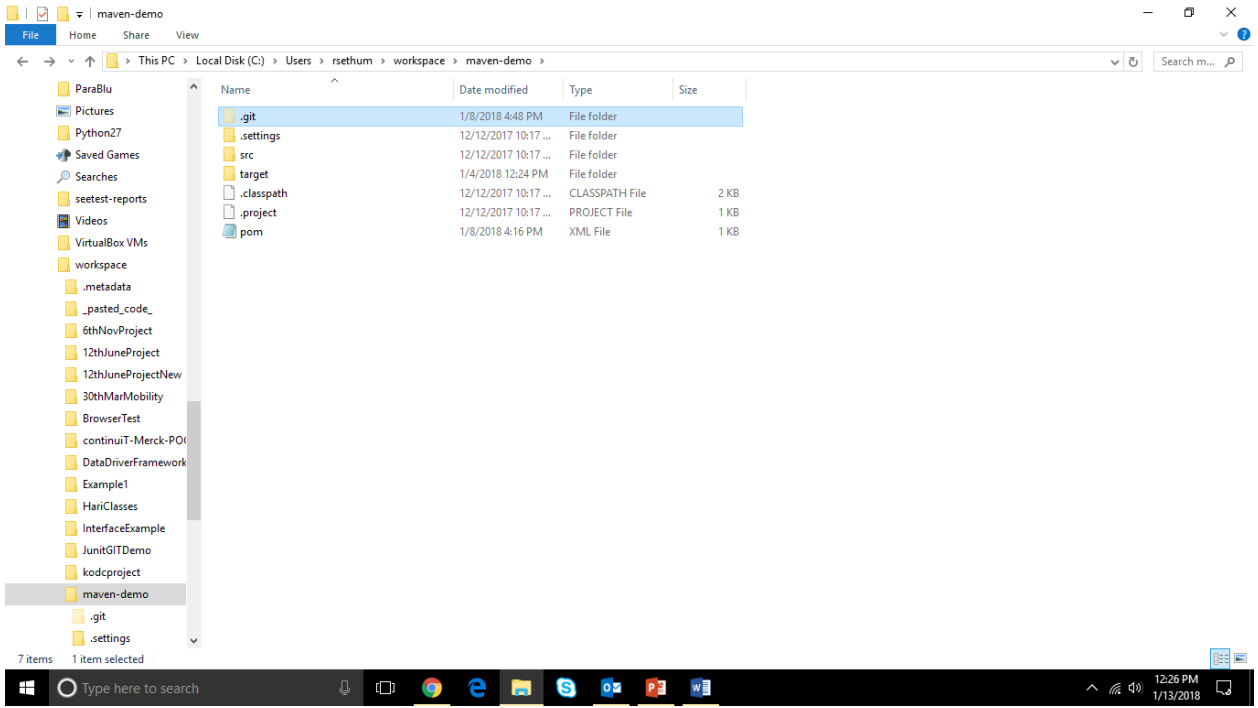
Create Jenkins Pipeline Job

- **Write a groovy script (Continued)**

```
        stage('Unit Testing') {
        steps {

          bat  'mvn -f c:/Users/rsethum/workspace/maven-demo/pom.xml test'
            echo "Welcome to Unit Testing"
        }
      }

    }



  post {
     always {
        echo 'it works always'
     }
     success {
        echo 'It will get executed only if successful'
     }
     failure {
        echo 'It will get executed only if failed'
     }
  }
}
```

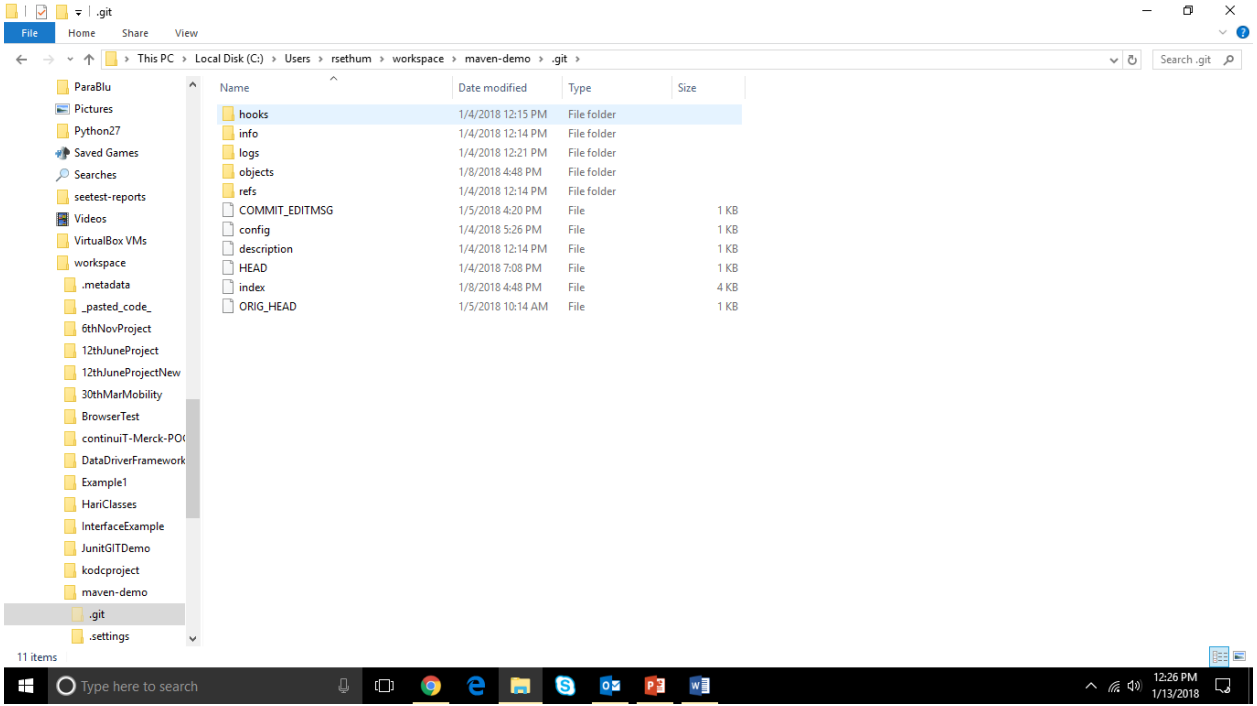
Create Jenkins Pipeline Job

- **In the git repository create the post-commit file**



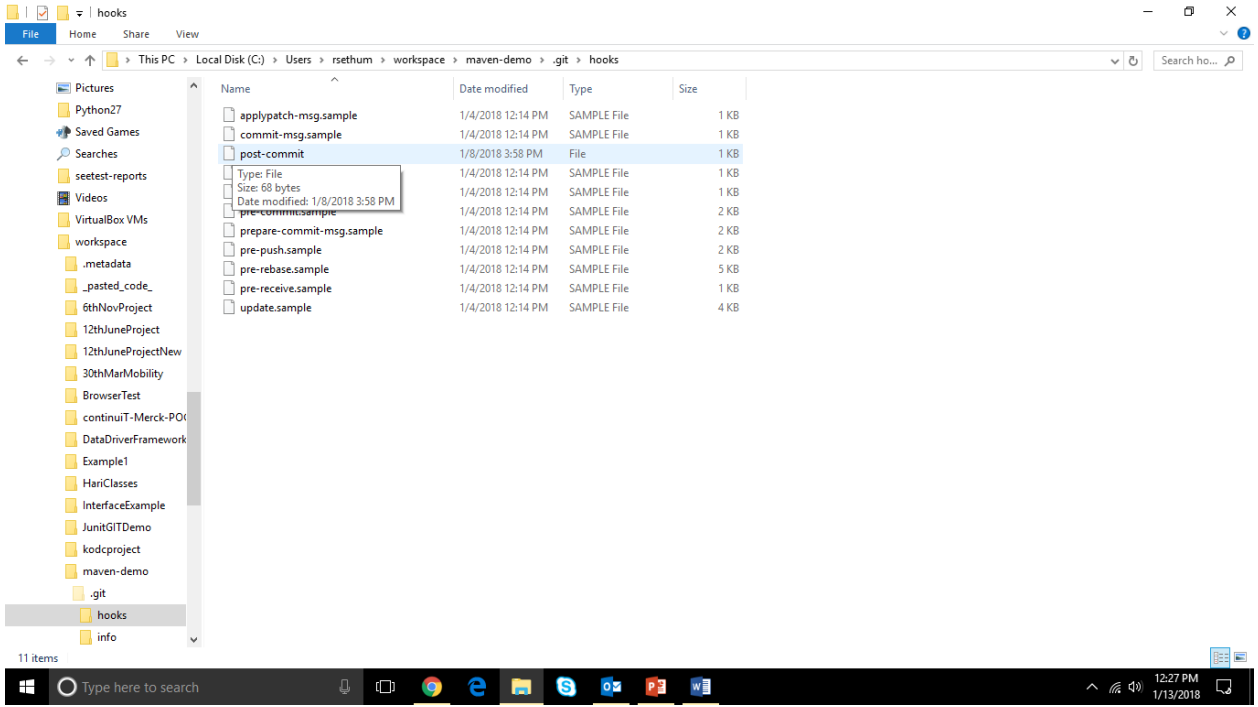
Create Jenkins Pipeline Job

- **In the git repository create the post-commit file (Continued)**



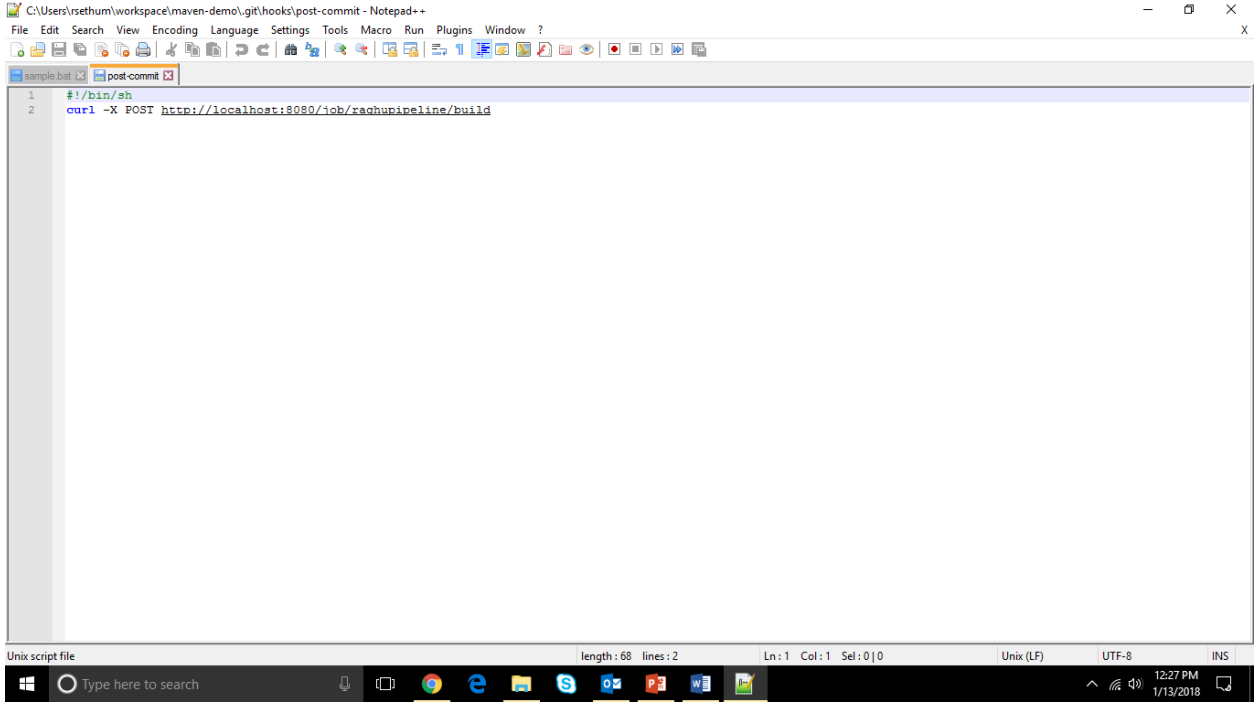
Create Jenkins Pipeline Job

- **In the git repository create the post-commit file (Continued)**



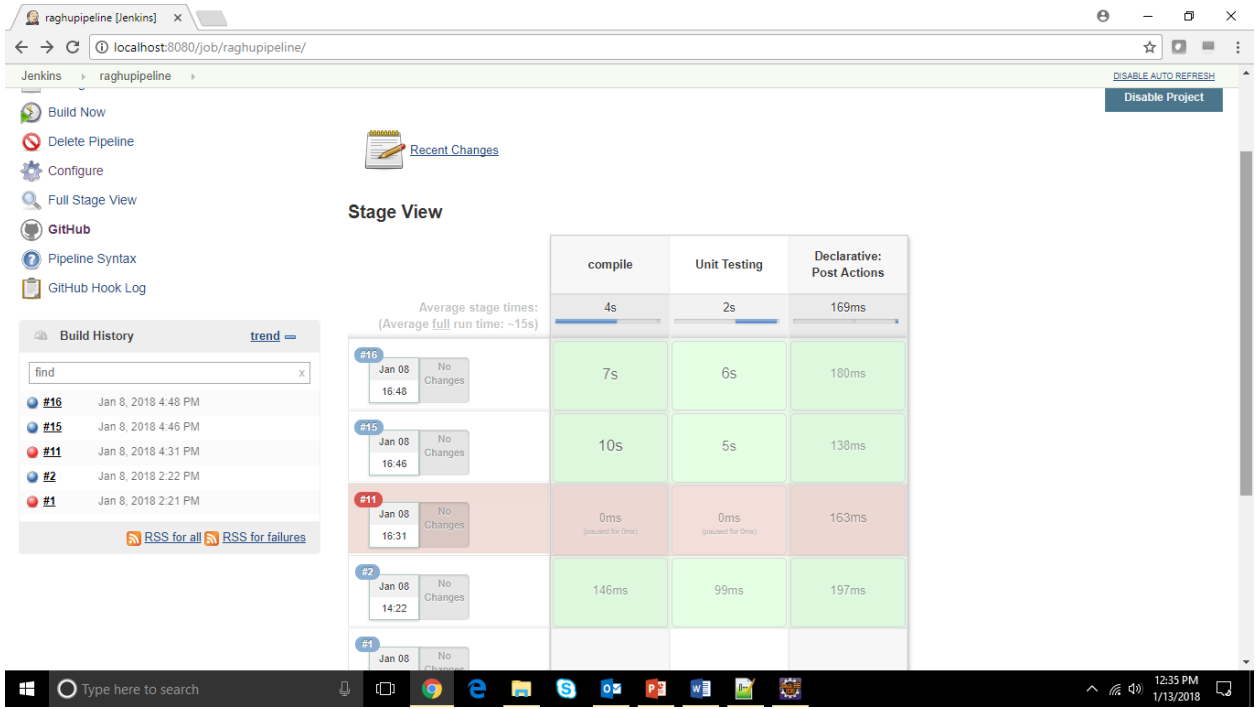
Create Jenkins Pipeline Job

- **In the git repository create the post-commit file (Continued)**



Create Jenkins Pipeline Job

Commit a new change in the Git repository. It would trigger a new build. You can monitor different stages of build process as demostrated in the below picture



Create Jenkins Pipeline Job

You could also view the console output as given below

References:

1. http://www.javaworld.com/javaworld

Thank You

Avinash Patel

Senior Manager

avinash.patel@wipro.com