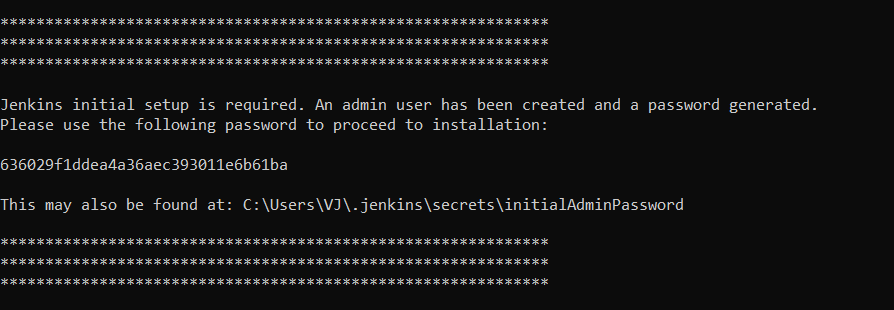
* Download latest Jenkins version from <https://updates.jenkins-ci.org/download/war/>
* Run the war file from cmd, command ‘java -jar Jenkins.war’
* After successfully running jenkin.war, in cmd below output will be generated.

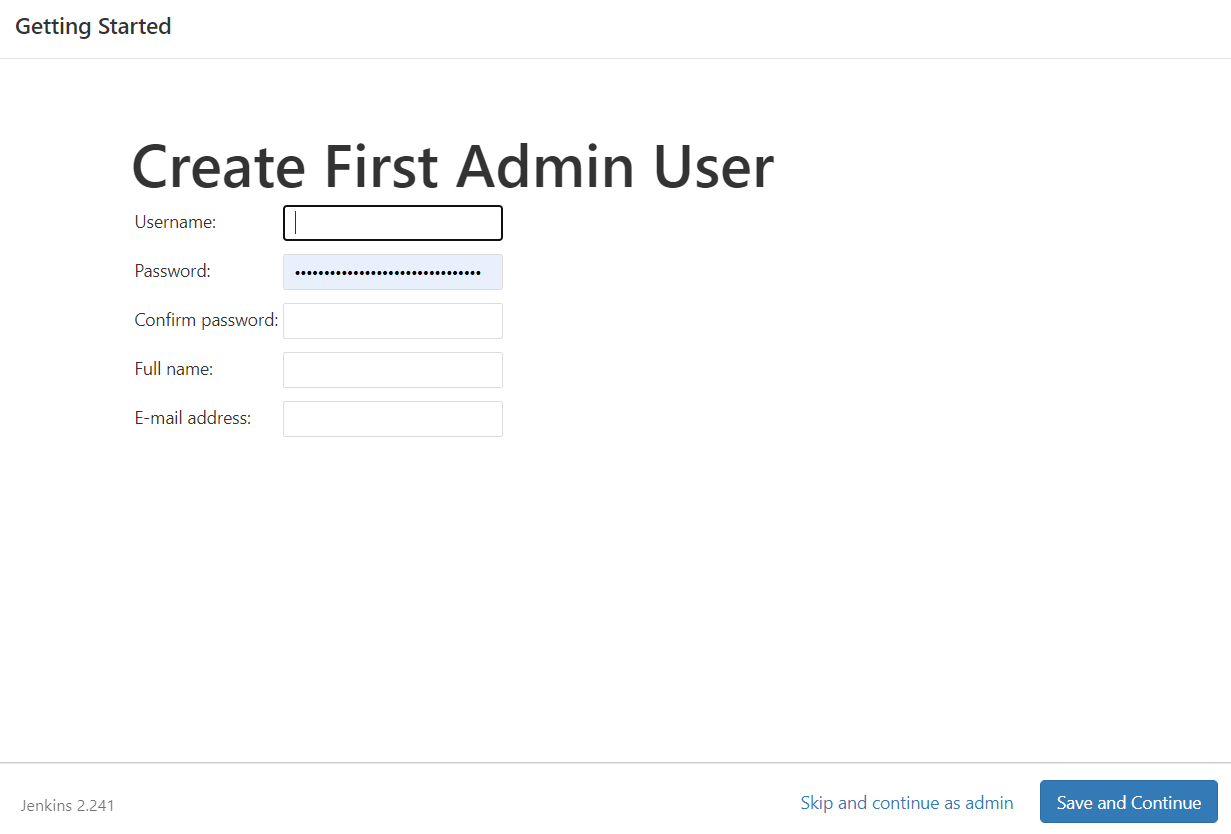


OR

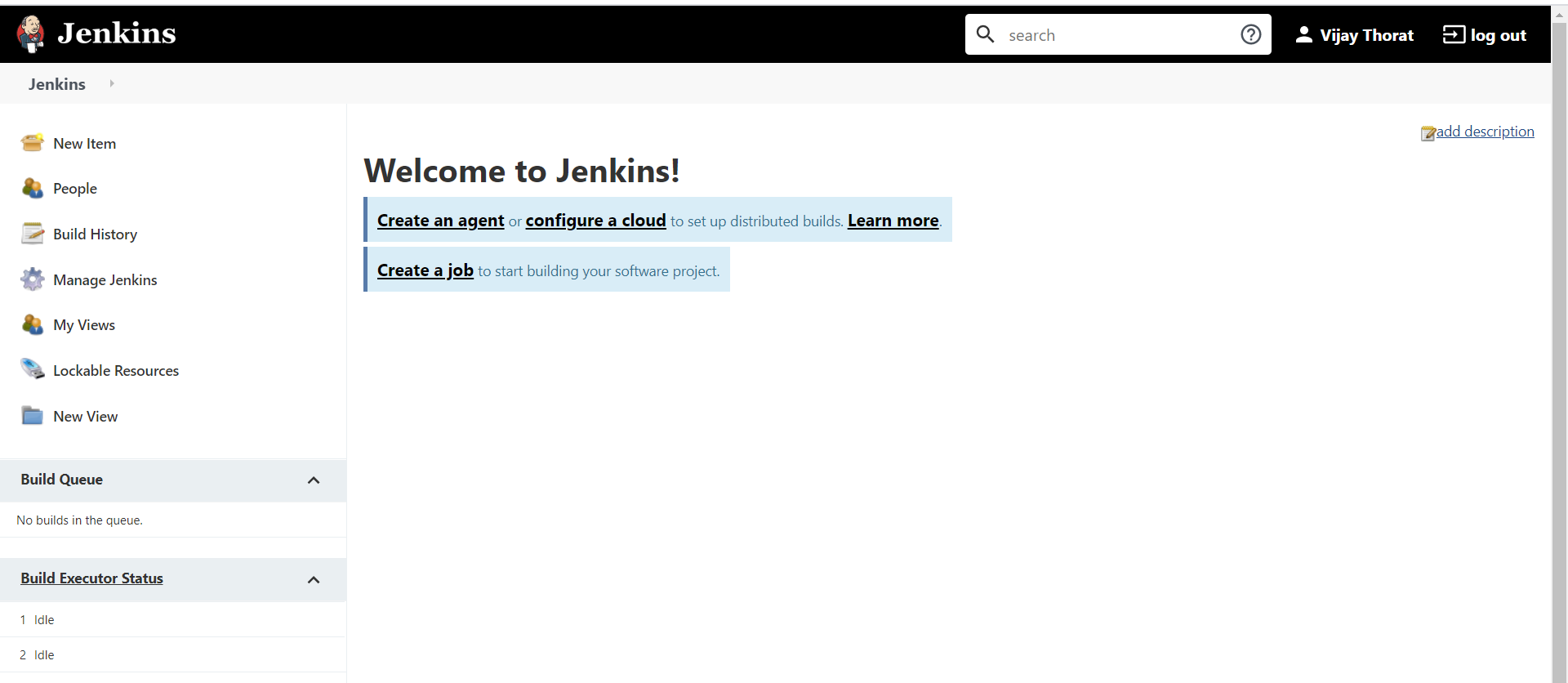
* Simply install the installer zip from <https://www.jenkins.io/download/thank-you-downloading-windows-installer/>

Refer 🡪 <https://www.youtube.com/watch?v=gT9qnZi3f74>

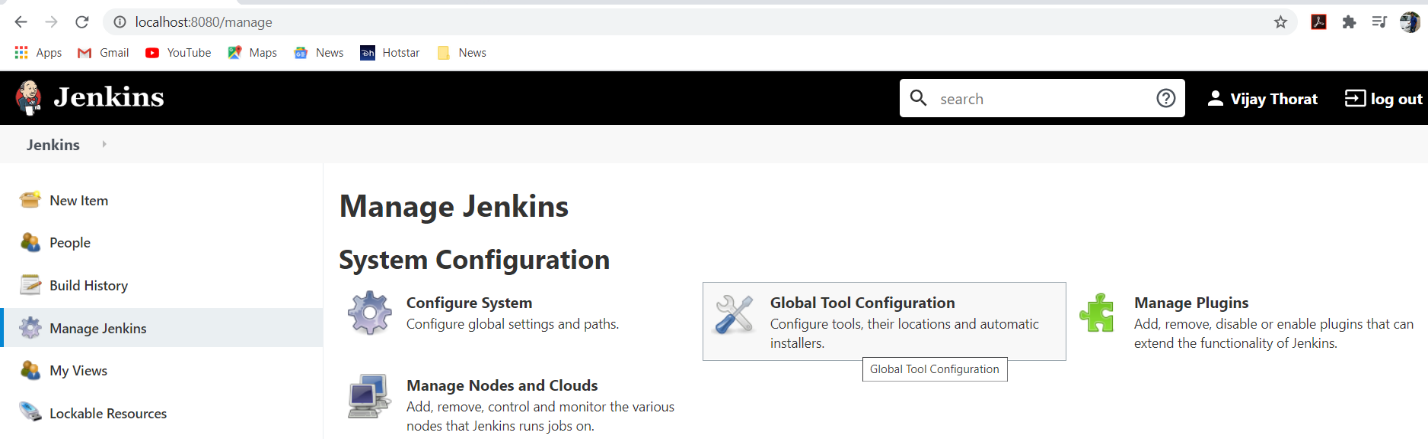
* Now open enter ‘localhost:8080’ in any browser, Jenkins sign in page will appear.
* Copy and paste above generated password.
* Click on ‘Install Suggested Plugin’. It will take some time to install all the plugins.
* After that ‘Admin User’ Creation window will appear. (vijay004 / rahul004)



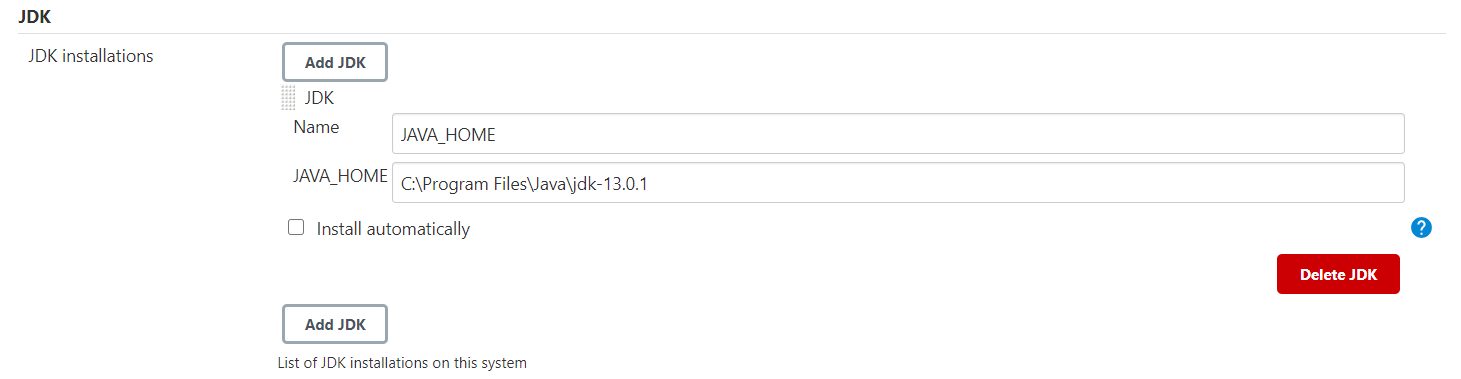
* After login, it will take to Jenkins home page.



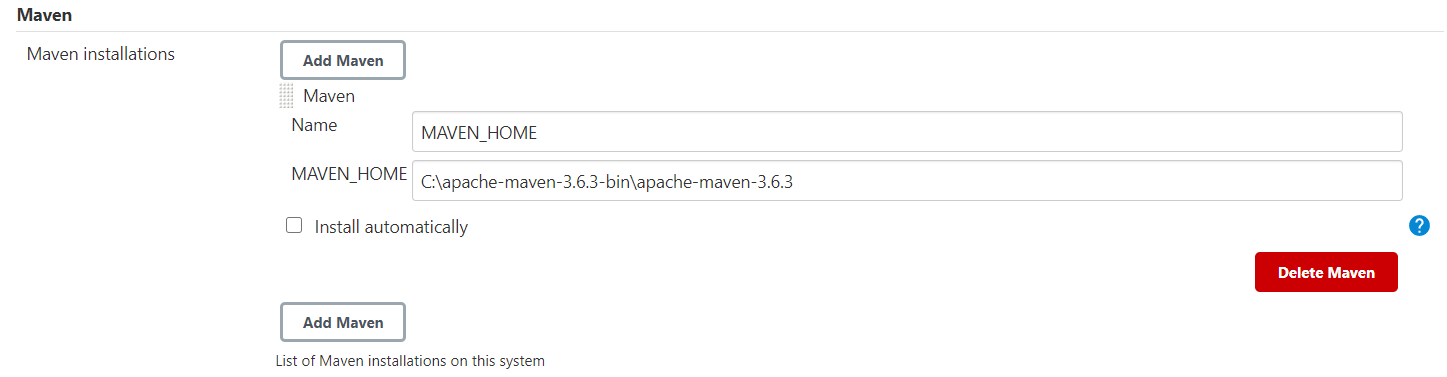
* Now we need to config JAVA and Maven in the project. It’s a one time activity.
* Go to Manage Jenkins -> Global Tool Configuration



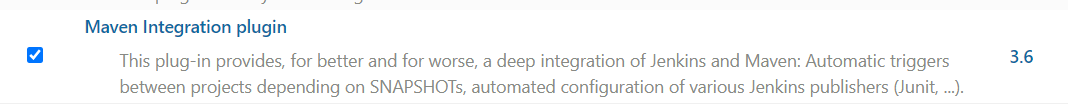
* Click on Add JDK, uncheck ‘install automatically’ option.
* Provide name as ‘JAVA\_HOME’ and provide system path of JDK.



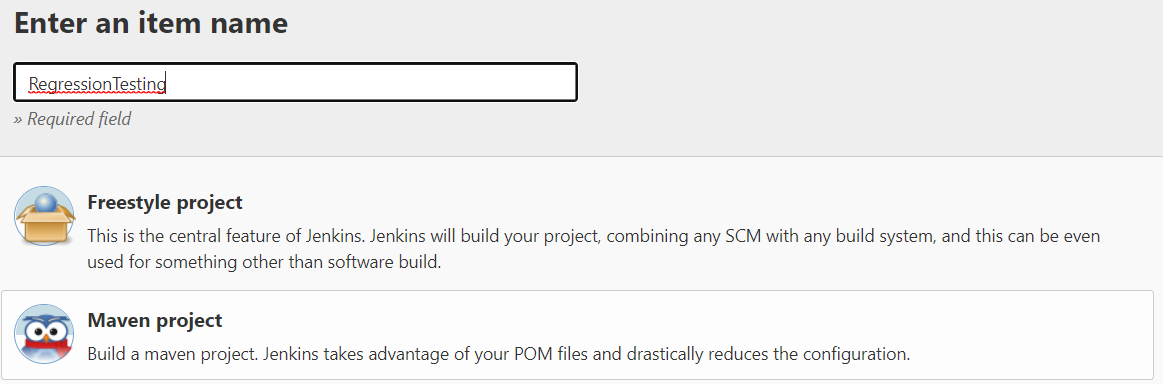
* Now in same way configure Maven as well.
* Maven path can be found out from system environment variable.



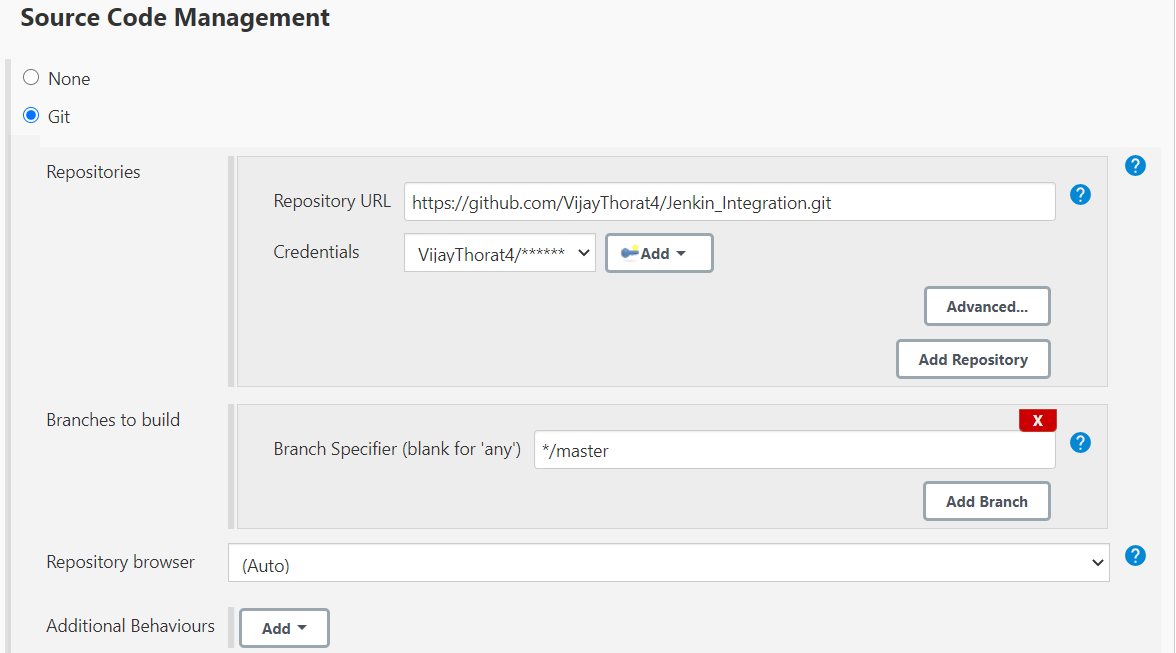
* Click Save at the bottom of the form.
* From Manage Jenkins -> Manage Plugins, install ‘Maven Integration Plugin’.



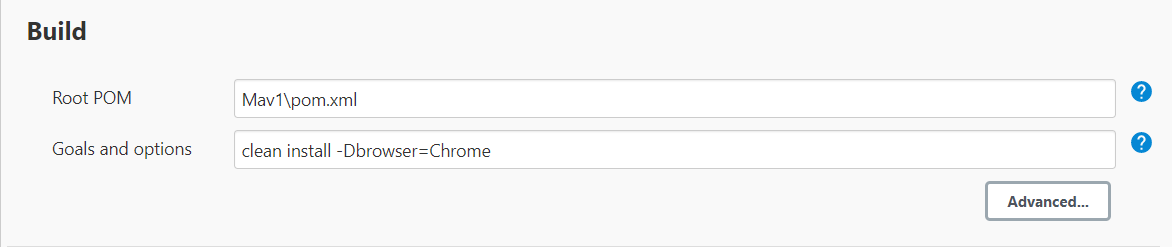
* Now from Jenkins home page, click on New Item and create new Maven project and click OK.



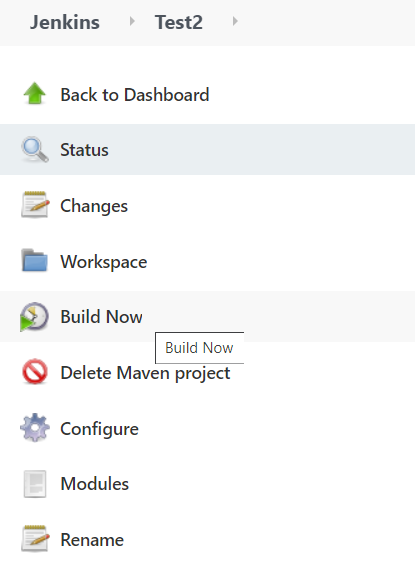
* In the ‘Source Code Management’ option select Git.
* Copy past GitHub repository URL and enter github credentials to access it.



* In the Build option, provide path of pom.xml and goals.



* Click on Save, project gets created.
* To start the job, click on build now.



Jenkins scheduling Build Periodically format 🡪

By setting the schedule period to 15 13 \* \* \* you tell Jenkins to schedule the build every day of every month of every year at the 15th minute of the 13th hour of the day.

Jenkins used a **Cron expression**, and the different fields are:

MINUTES Minutes in one hour (0-59)

HOURS Hours in one day (0-23)

DAYMONTH Day in a month (1-31)

MONTH Month in a year (1-12)

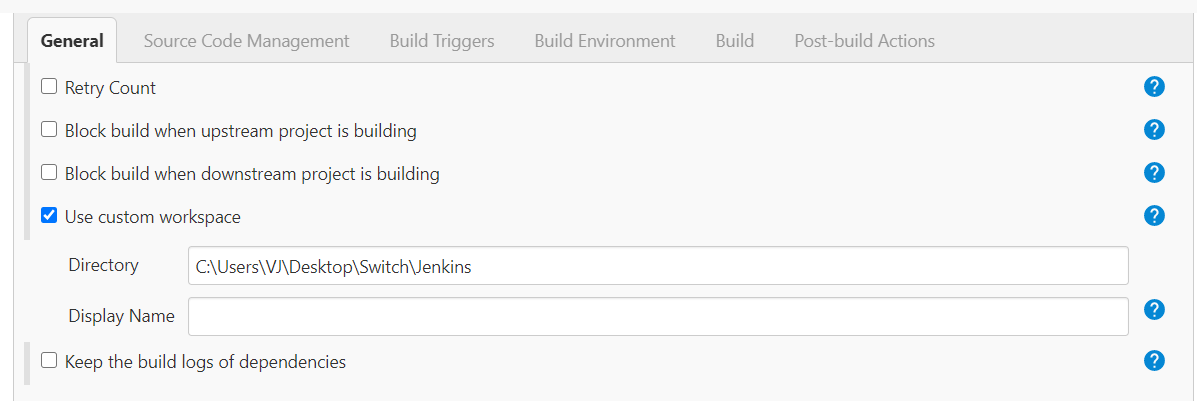
DAYWEEK Day of the week (0-7) where 0 and 7 are Sunday

Steps to schedule a build of Maven project through Jenkins 🡪

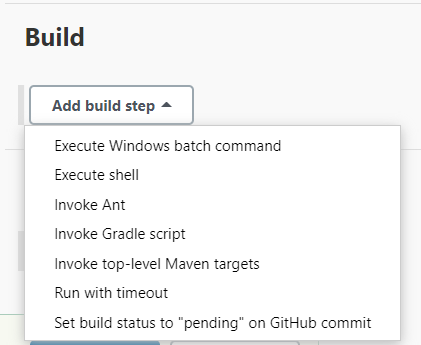
* Execute the project through Eclipse as Maven project (instead of testng)
* Execute this through command prompt
* Write a batch file for this execution
* Fire a scheduled build of this batch file through Jenkins.

Steps to execute a batch file through Jenkins 🡪

* On Jenkins dashboard, click on ‘New Item’ and create a Freestyle project.
* In the project configuration, General -> Advanced -> check box ON ‘use custom workspace’ and provide the system working directory.



* In Build, select Windows batch command.

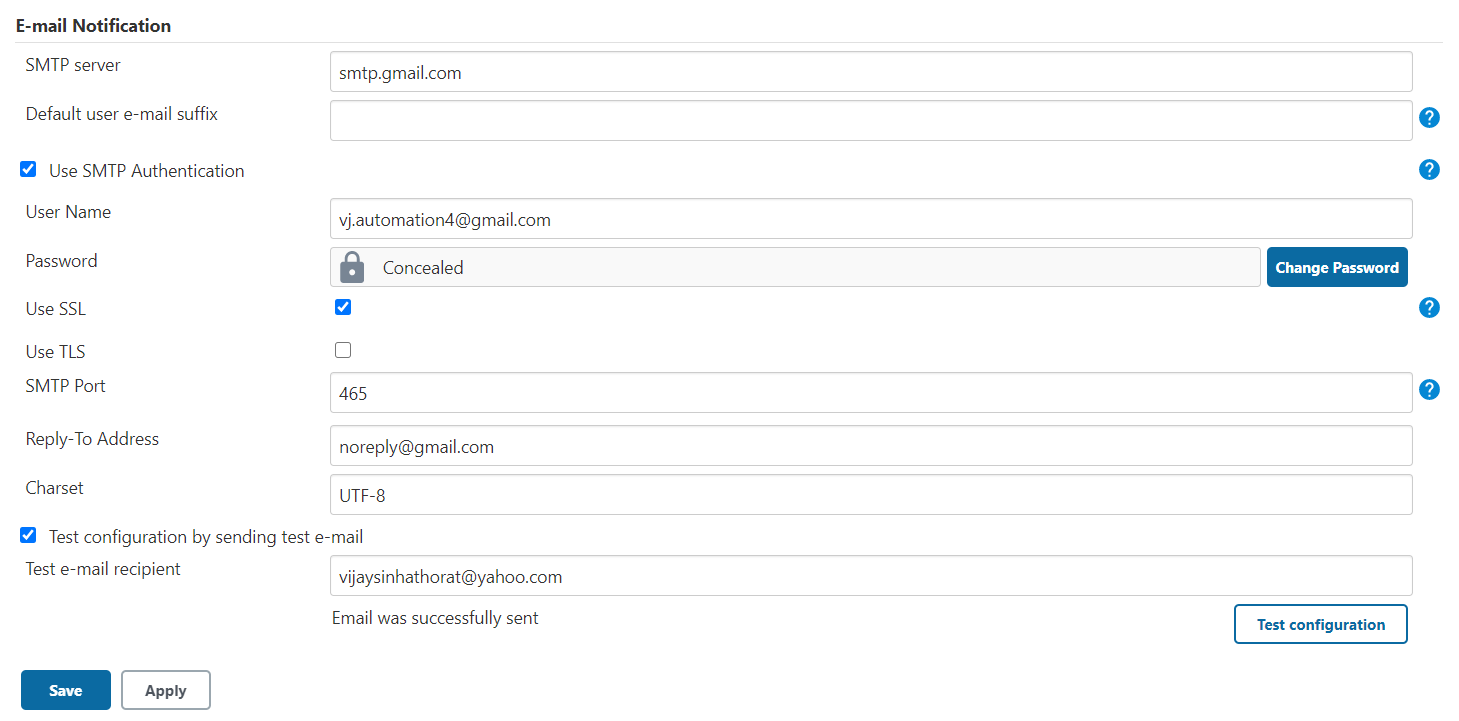


* Just provide the name of batch file.

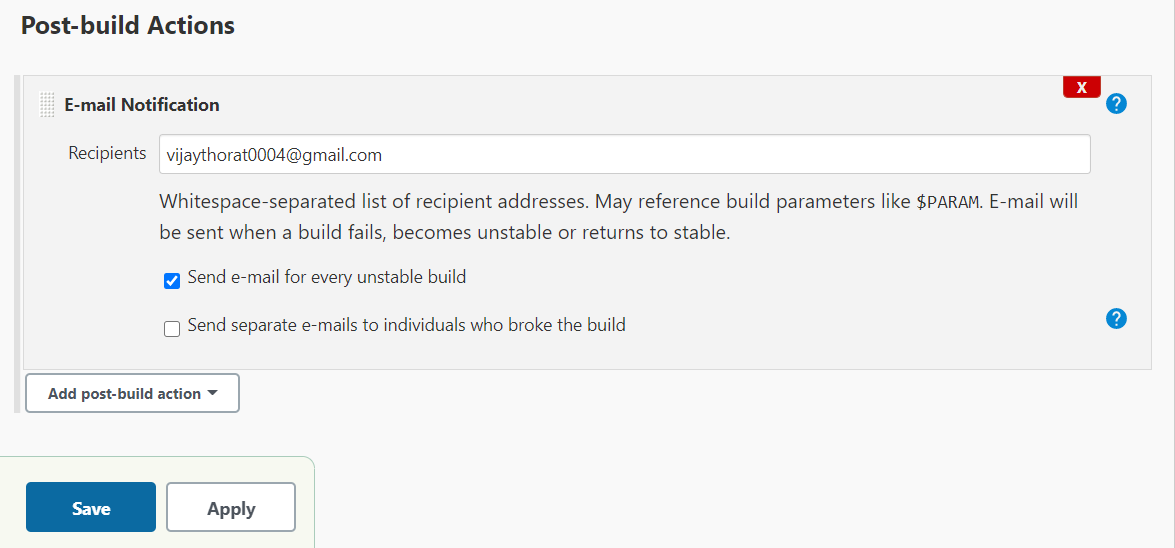


To send E-mail notification 🡪

* Go to Manage Jenkins -> Configure System.
* Scroll down to E-mail Notification

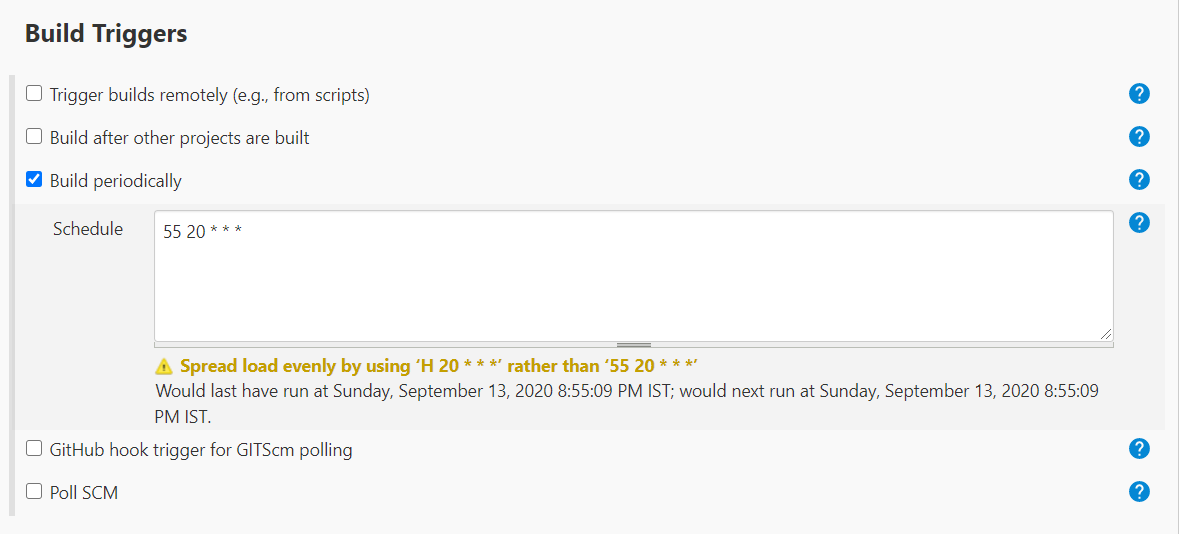


* Now go to any job -> Manage -> Add Post build Action -> Email Notification and add an email which is expected to receive mail. It will send the mail when job fails.

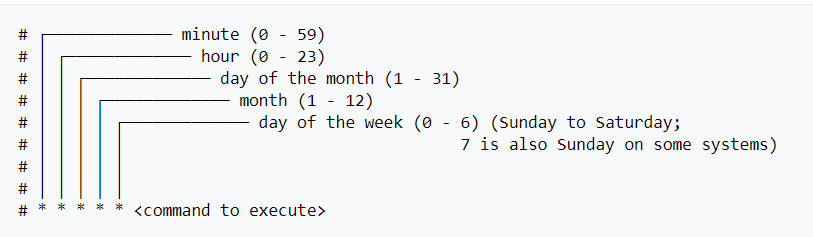


To schedule a job build 🡪

* Go to any job -> Manage -> Build Trigger -> checkbox ON Build Periodically.
* It is scheduled by **CRON** pattern.

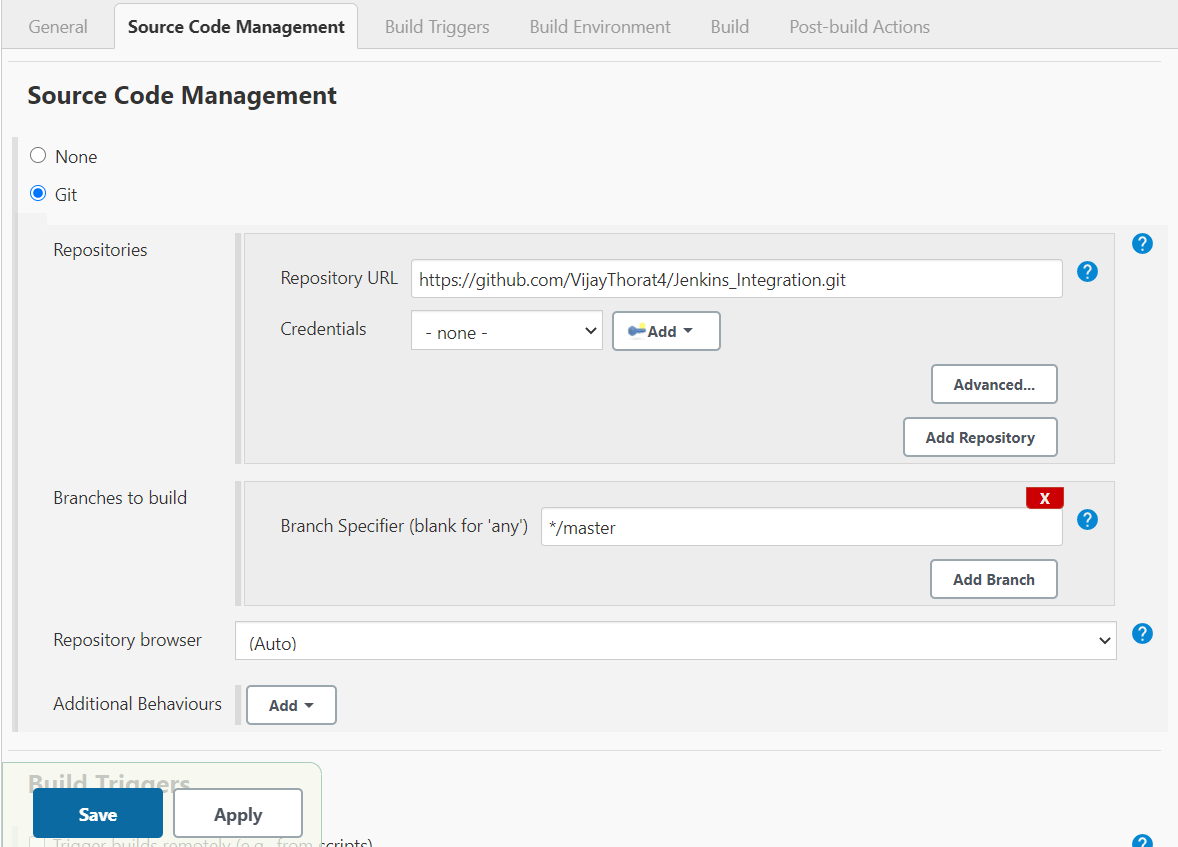


Note – This will execute job at 8.55pm daily



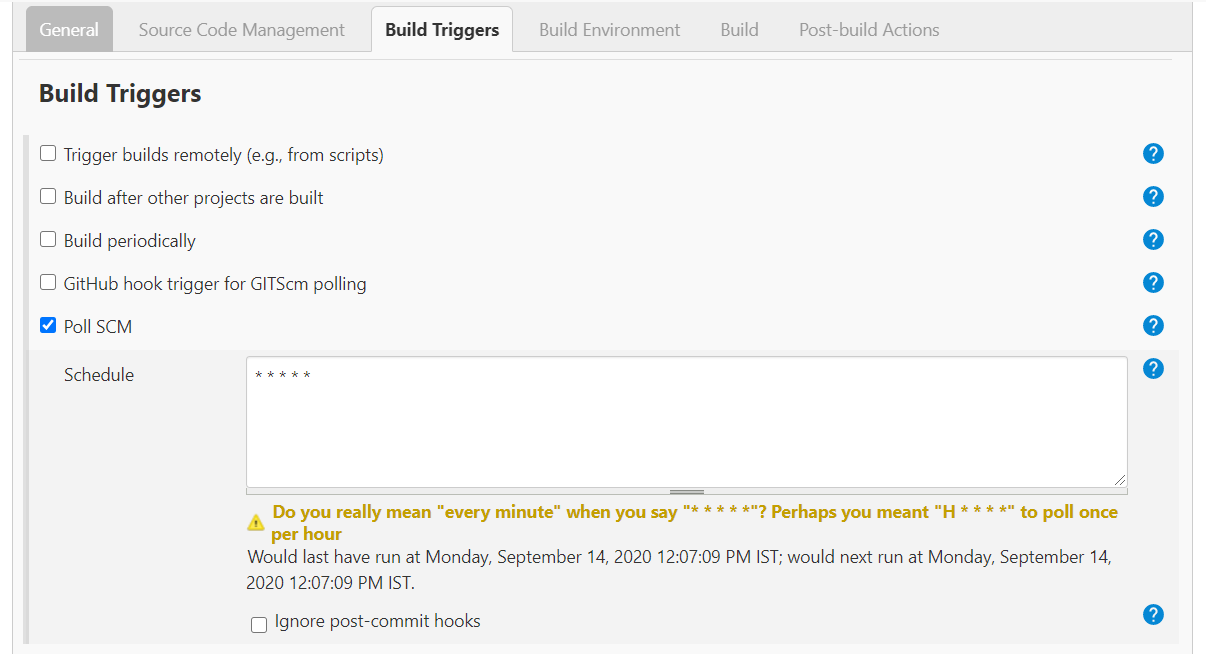
Integrating Jenkins with GitHub 🡪

Create a new Item and in the source code select Git option, provide the git repository URL, no need to provide any credentials, keep default branch as Master.



If you want you can trigger its build periodically using above method.

However in industry after integrating with git repository, Poll SCM option is used. We provide CRON patter in schedule field of Poll SCM. As per below pic, it will check is there new code promoted to provided github repository every minute and if yes then it will trigger the build.



So by this Poll SCM option (using five star cron patter) you can immediately trigger a build as soon as you push any code / make changes in any file in central repository.

**Groovy Script**

<https://www.youtube.com/watch?v=Ozna0MS5D1E&t=1888s>

**Node** – A node is a machine which is a part of Jenkins environment & is capable of executing pipeline. If we are running Jenkins pipeline on a single machine then it there will be only one node & agent. But in real industry projects, there are multiple nodes.

**Stage** – Stage is nothing but a task like build, test, deploy etc

**Step** – A stage contains multiple steps. These are nothing but mini tasks.

Note – If Jenkins pipeline is running on local system then it will start with ‘node’ keyword. If it is supposed to run on multiple systems (remote system) then script will start with ‘pipeline’ keyword. Also on local system it is not necessary to use ‘step’ keyword. It can be written directly under stage.

* To access the parameters created within Jenkins Config, we use below syntax (say ‘Name’ is the string parameter in Jenkins pipeline).

${env.Name}

* We can create a parameter inside ‘parameters’ which is under ‘pipeline’ keyword. We can create any type of parameter inside groovy script that we can create under Config option in pipeline.

*pipeline {*

*agent any*

*parameters {*

*string(name: 'PERSON', defaultValue: 'Mr Jenkins', description: 'Who should I say hello to?')*

*text(name: 'BIOGRAPHY', defaultValue: '', description: 'Enter some information about the person')*

*booleanParam(name: 'TOGGLE', defaultValue: true, description: 'Toggle this value')*

*choice(name: 'CHOICE', choices: ['One', 'Two', 'Three'], description: 'Pick something')*

*password(name: 'PASSWORD', defaultValue: 'SECRET', description: 'Enter a password')*

*}*

* To access above parameters (which are created using groovy script), we need to use below syntax.

*steps{*

*echo "Code Deployed & informed to ${params.PERSON}"*

*}*

* If you are using parameters, then make sure to use them inside double quotes (“”) & not inside single quote (‘’). If you use inside single quote then it will not fetch value.

Example of Groovy Script 🡪

*def fun1 (){*

*echo "${env.Name}"*

*String var1 = "Ranjit Drives"*

*echo var1.split(' ')[1]*

*}*

*pipeline {*

*agent any*

*parameters{*

*string(name: 'PERSON', defaultValue: 'Mr Kohli')*

*}*

*stages {*

*stage('Build') {*

*steps {*

*echo "Clonning the Build ISO for ${env.Name}"*

*}*

*}*

*stage('Deploy'){*

*steps{*

*echo "Code Deployed & informed to ${params.PERSON}"*

*}*

*}*

*stage('Release'){*

*steps{*

*echo 'Product ready to release'*

*fun1()*

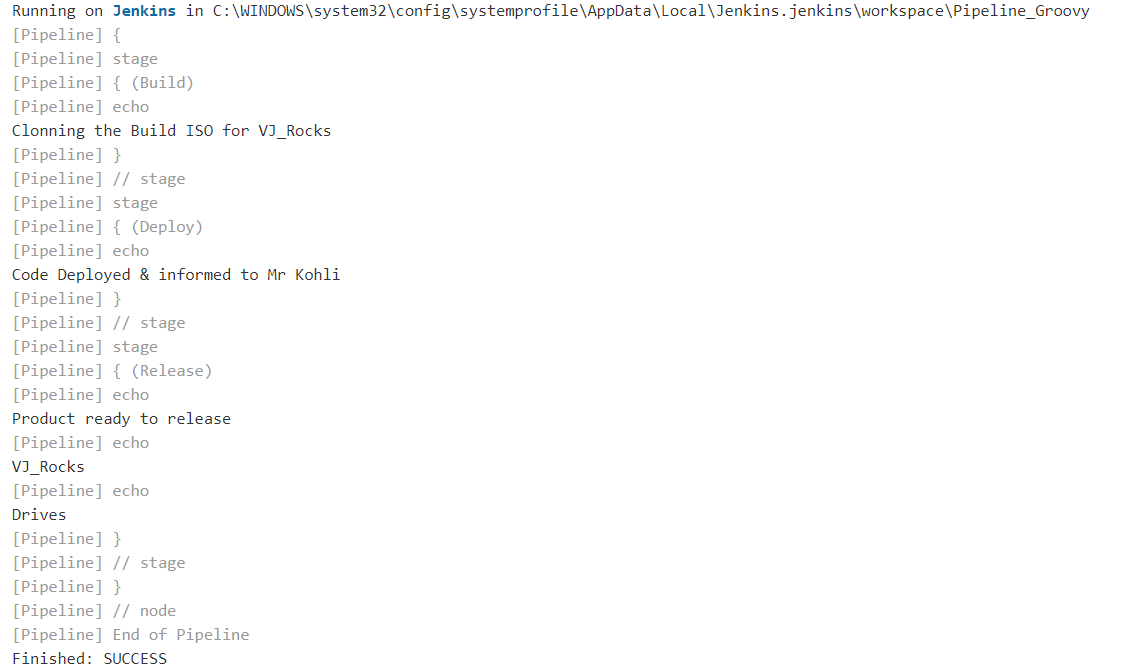
*}*

*}*

*}*

*}*

Output of above script 🡪



* *While providing file path in groovy, always use single forward slash ‘/’*