**Jenkins + Maven + Selenium (on AWS EC2 instances)**

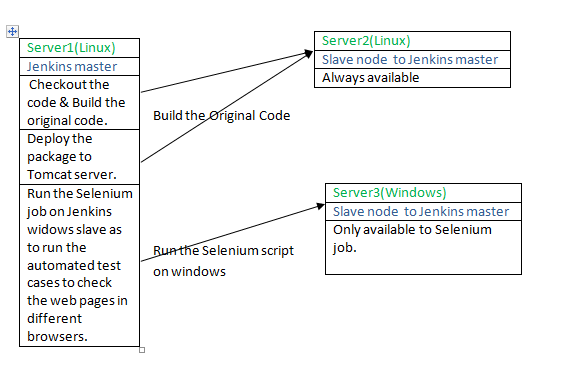
|  |
| --- |
| Note: For full image content view, change the layout of this doc to **Web Layout**. Right bottom of doc |

**Goal of this document:** Original Code check-in for web app 🡪 Prepare Selenium script to test the web pages in different browsers 🡪 Trigger a local maven build OR trigger a Jenkins job as to Build the original code 🡪 deploy the package to Tomcat server 🡪 build the Selenium job to run the automated test cases to check the web pages in different browsers.

Required Tools & servers:

|  |  |  |  |
| --- | --- | --- | --- |
| **Server/Type** | Server1(RHEL) | Server2 (RHEL) | Server3(Windows) |
| **Server Activity** | Jenkins master | Slave node to Jenkins master | Slave node to Jenkins master |
| **Tools** need to be installed | Java  Tomcat  Maven  Git  Jenkins | Java  Git  Maven  Tomcat | Java  Git  Maven  Selenium Grid  IE, Chrome, Firefox etc |

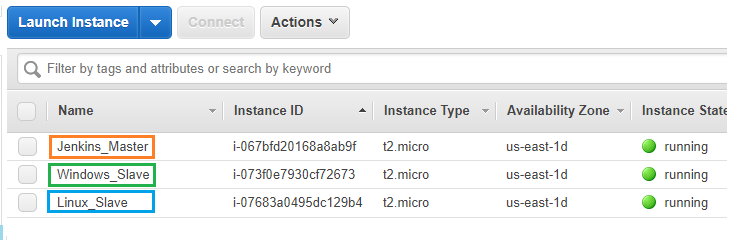
Communication between servers:



To implement the above task, we have to launch two RHEL, one windows machine. After that, we will Jenkins master on one of the RHEL(Linux) machines, and setup the other Linux machine as a salve node to Jenkins master. Also, setup the windows machines as slave node to the Jenkins master linux. Follow the below steps to complete this exercise.

1. **Launch one Windows instance in AWS**: As per the docuemnt **AWS-Windows-EC2-Launch-Instance.doc**, launch the AWS EC2 windows instances.
   1. As per the doc, **Jenkins+Maven+Selenium-OnLocalWindows.doc**, install JDK1.8, maven, install required browsers, setup **Selenium Grid** as well.
   2. Install “Git Scm” as well.
2. **Launch two RHEL instances on AWS**: Launch two REHL instances separately. Refer the doc, **AWS-TOMCAT-JENKINS.doc** and **Section.A** to know how to launch a Linux machine from Phase-1 docs.
   1. **Jenkins master on one RHEL** **instance**: As documented in **Java-Tomcat-maven-git-Jenkins.sh**, install the required tools to setup Jenkins master on one of the Linux instance. Also, proceed with Jenkins setup and launch the Jenkins master server & install required plugins (if any).
   2. **Jenkins Slave on one RHEL instance**: As documented **Java-Tomcat-maven-git-Jenkins.sh**, install JDK, Git, Maven tools on this Linux machine. We have to setup this Linux machines as slave node to the Jenkins master(Refer the below step-3 to setup the machines as nodes/slaves to Jenkins master).

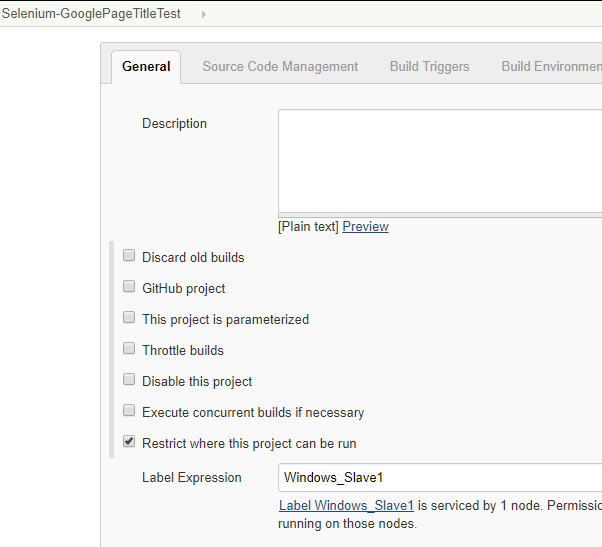
Once you launched all the three machines, you will see the machines as below. (Give the appropriate names for each machine).

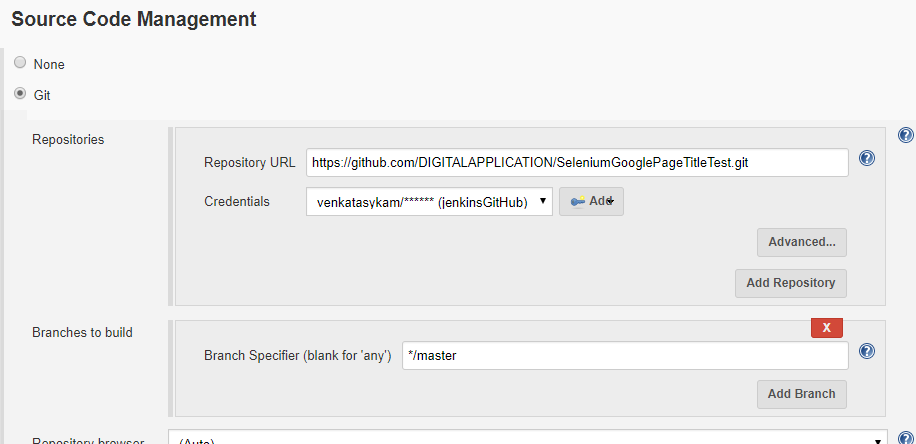


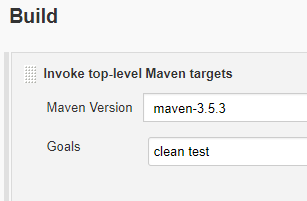
1. **Linux and windows node setup**: Refer **AWS-Jenkins-Slaves-Linux-And-Windows-Configuration.doc**.
2. **Create a Jenkins job in Jenkins master**: Now, let’s take an example selenium code for Google page title test, and configure a Jenkins job in Jenkins master, and then the selenium script run on windows slave.

4.1. **Testing Google page title**: Configure a job for the selenium script 🡪 <https://github.com/DIGITALAPPLICATION/SeleniumGooglePageTitleTest.git>

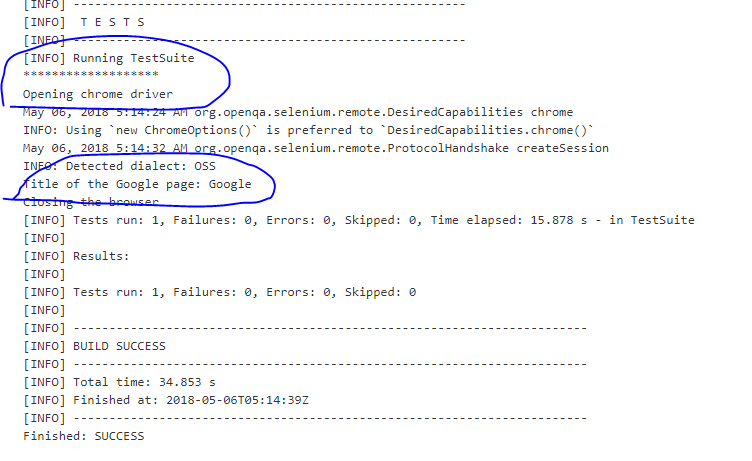
Job General section, restrict the job to run only on the **Windows\_Slave1** node.







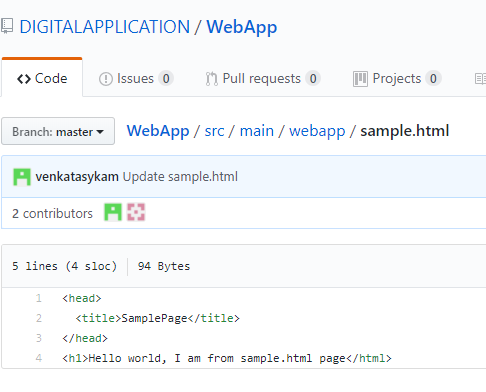
Save the job configuration & trigger the Jenkins job: Build is success.



4.2. Development & selenium automation tests are in parallel (the main aim of the Agile methodology is to Dev team & Test team should work paralely & closely to make the software as error free).

There is a WebApp, developing a sample.html page and it has some tags title & h1 tags as shown below.

<https://github.com/DIGITALAPPLICATION/WebApp/blob/master/src/main/webapp/sample.html>



Refer the doc, **AWS-TOMCAT-JENKINS.doc** to build & deploy the WebApp on tomcat server. Once the deployment done, out application URL is http://<IPAdress>:8080/WebApp-1.0.0-SNAPSHOT/Sample.html

Now, we have to writ selenium script for this web page sample.html.

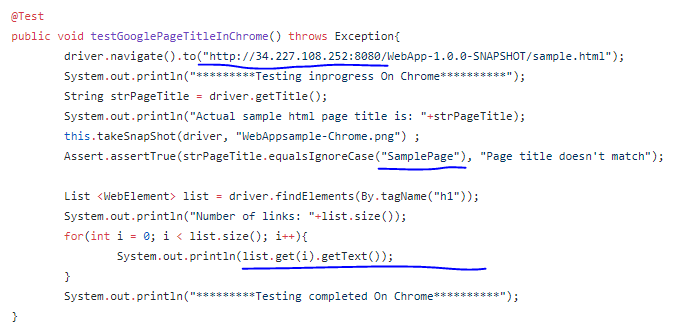
As per project requirement:

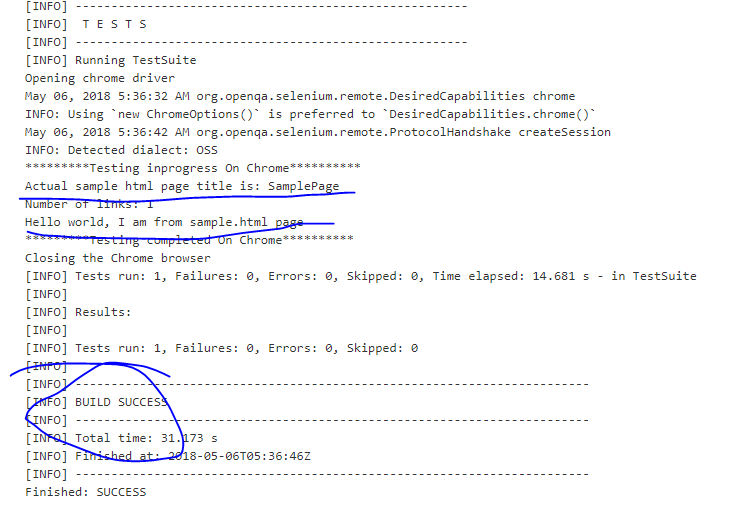
Sample.html web page title should be: SamplePage.

Now, let’s test the web page title should match with “SamplePage”, otherwise build should be failed. For that, Also, we are writing the selenium code as to comparing the actual title page “SamplePage” & expected title “SamplePage” as per the requirement.

Here is the Selenium test project: <https://github.com/DIGITALAPPLICATION/SeleniumWebAppTest.git>

Also, we are trying to print the title of sample.html page & print the h1 tag value.





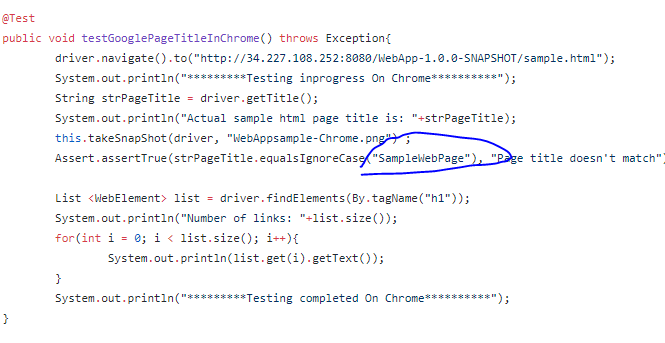
Now, let’s test it in failure case.

Assume,

**Actual title**: SamplePage (which is actually developer coded in html page)

**Expected Title**: SampleWebPage (Assume, this is the requirement given by project team)

Update the Selenium script as to compare with expected value “SampleWebPage” as this is the original clients requirement. Test engineers write the selenium script as such, but the dev team un-knwonly or by mistake they coded the title in html page as “SamplePage”. Let’s see how to test this automatically.



Now trigger the selenium test Jenkins job:

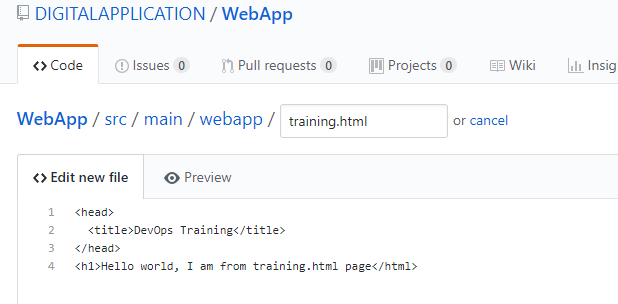


As per the client requirement, developer should needs to update the **sample.html** page title as to “SampleWebPage”.

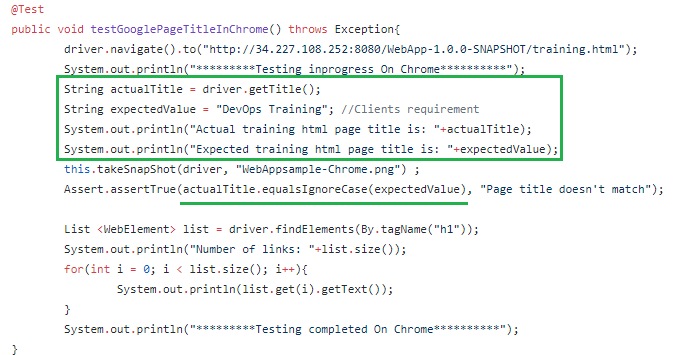
**Another Scenario**: Dev team & testing team working together. Since both the teams are working together, setup the pipeline in Jenkins as to build & deploythe original code to tomcat serever. And then automatically the selenium script Jenkins job should run to test the original code.

Add a new page called **training.html** to the WebApp project & the title should be “DevOps Training”.

**Development Team:** Dev team will add the html page for as per the requirement. In Parallel, dev team will inform to test team as they are going to add the taining.html age as per the new requirement. Test team start developing selenium script as per the requirement.



**Testing Team**: Writing automated test cases in selenium: Test team will get the complete test plan from project team what needs to test on the web pages of the original code.



The original Source code location is: [https://github.com/DIGITALAPPLICATION/**WebApp**/blob/master/src/main/webapp/training.html](https://github.com/DIGITALAPPLICATION/WebApp/blob/master/src/main/webapp/training.html)

The selenium test code location is: [https://github.com/DIGITALAPPLICATION/**SeleniumWebAppTest**/blob/master/src/test/java/com/test/WebAppTitleTestOnChrome.java](https://github.com/DIGITALAPPLICATION/SeleniumWebAppTest/blob/master/src/test/java/com/test/WebAppTitleTestOnChrome.java)

We need two separate jobs for each project. One for original & another one for Test project.

Here is the pipeline: First the WebApp (original source code) should be build 🡪 and then the Selenium project **SeleniumWebAppTest**. Earlier we triggered the jobs separately, now we are going to join them, they will run one after one.

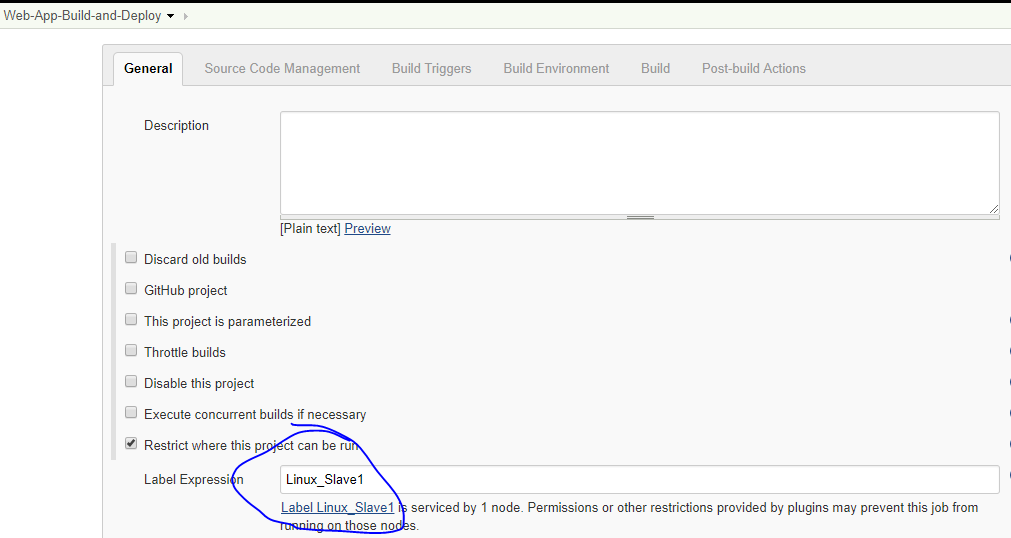
Jenkins job-1: Web-App-Build-and-Deploy

Jenkins job-2: Web-App-Automation-Test

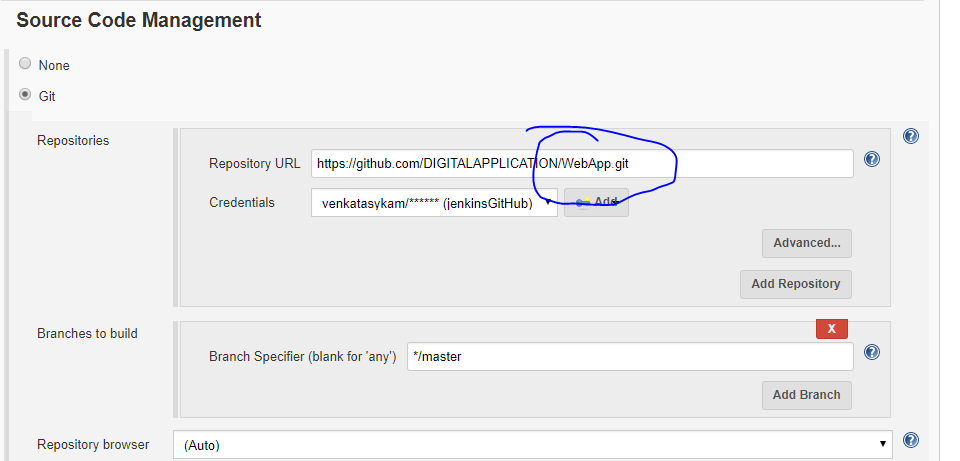
Configure the second job-2 **Web-App-Automation-Test** job as a downstream to job-1 “**Web-App-Build-and-Deploy**”.

**Jenkins job-1**: Web-App-Build-and-Deploy: Configuration

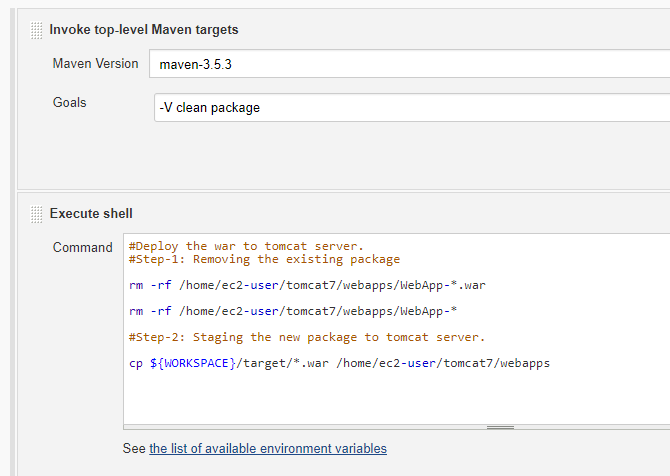
**General Section:** Restricting to Linux Slave1: Tomcat is installed & up in this slave. Build & deployment of original code will be executed on Linux\_Slave1 node. Part of the process, build the code, package the code, deploy the package to tomcat server which is running on the same machine.



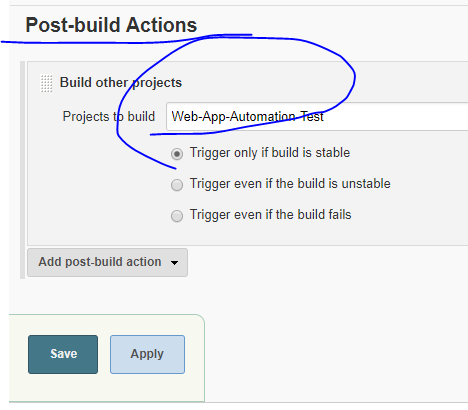
**SCM Section**:



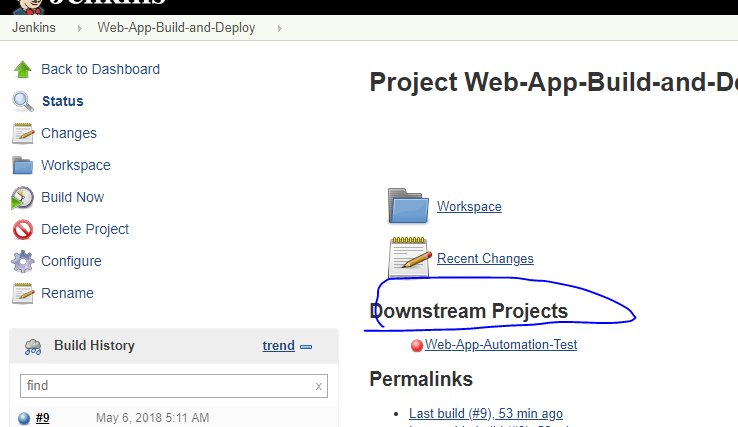
**Build section:** Build goals &Deployment script



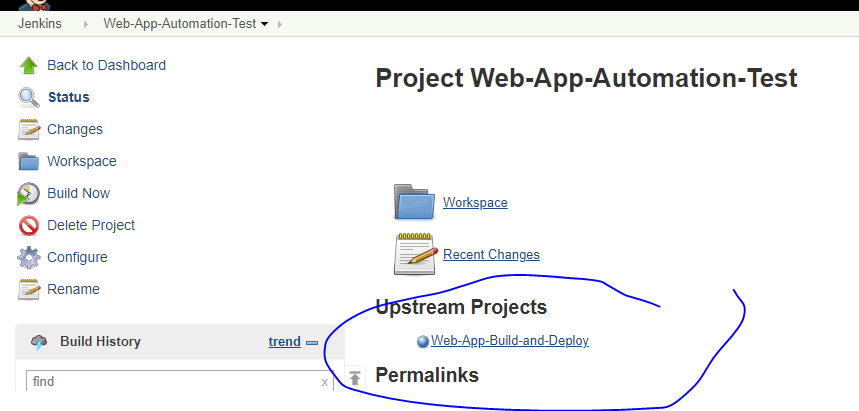
**Downstream job configuration**:



Observe the Job-1 once you saved the configuration: Job-2 is showing as downstream job for this job-1.

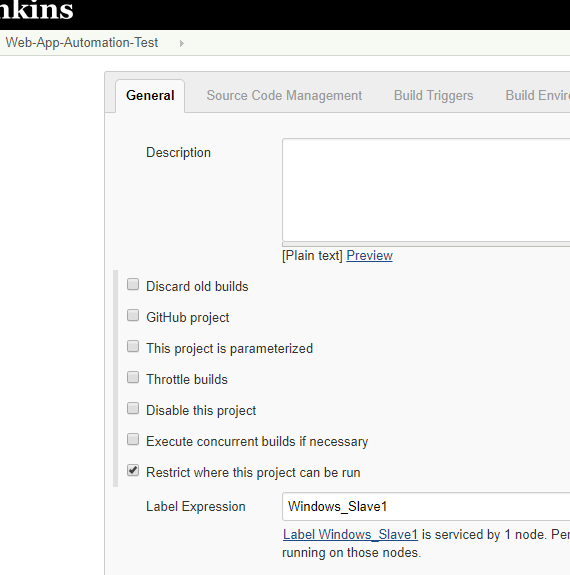


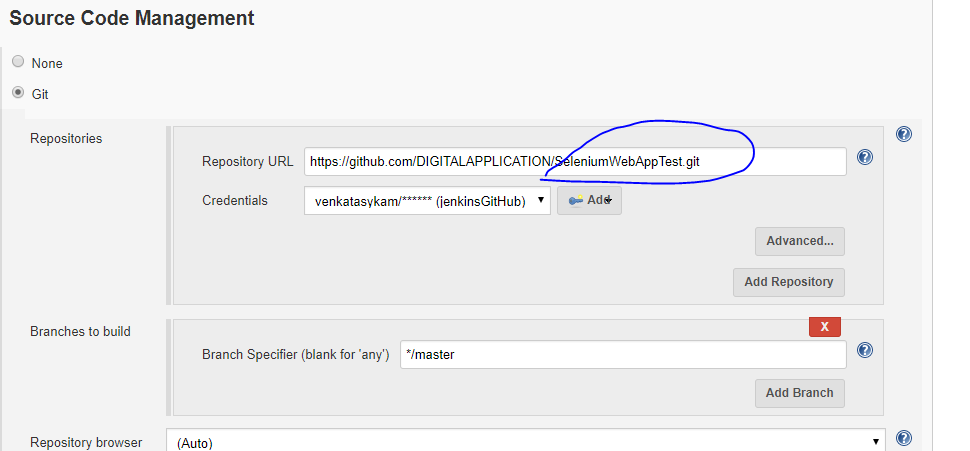
Observe Job-2, the job-1 is showing as the upstream job to Job-2.

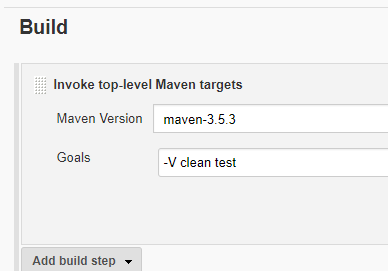


**Jenkins job-2: Web-App-Automation-Test: Configuration**

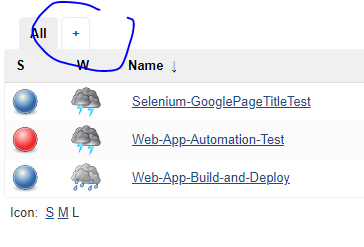
**General Section**: Restricted to windows slave since Selenium Grid is available here.

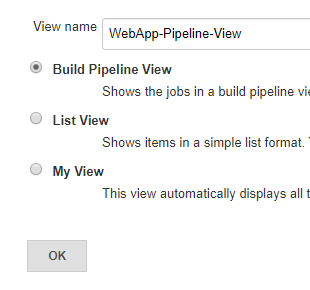




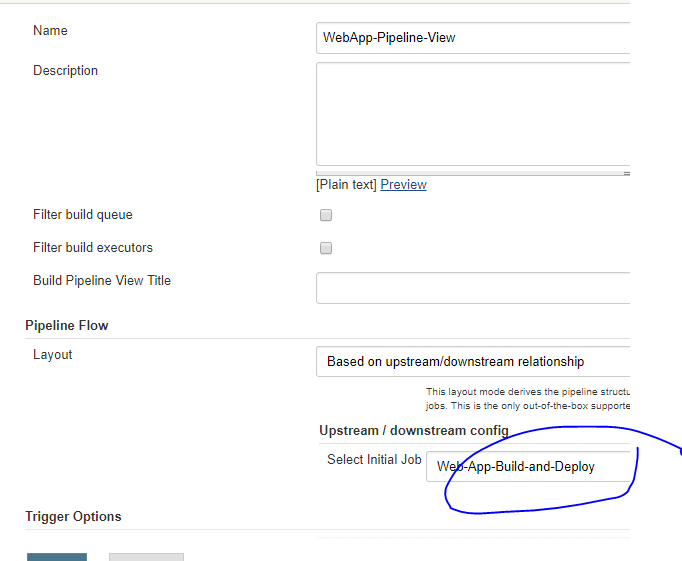


Note: Install the “Build Piepline” plugin under “Manage Jenkins🡪 Manage Plugins” & then create a new view with the template pipeline as below.

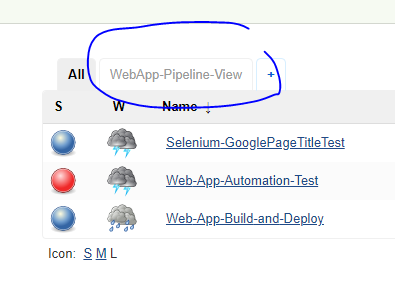




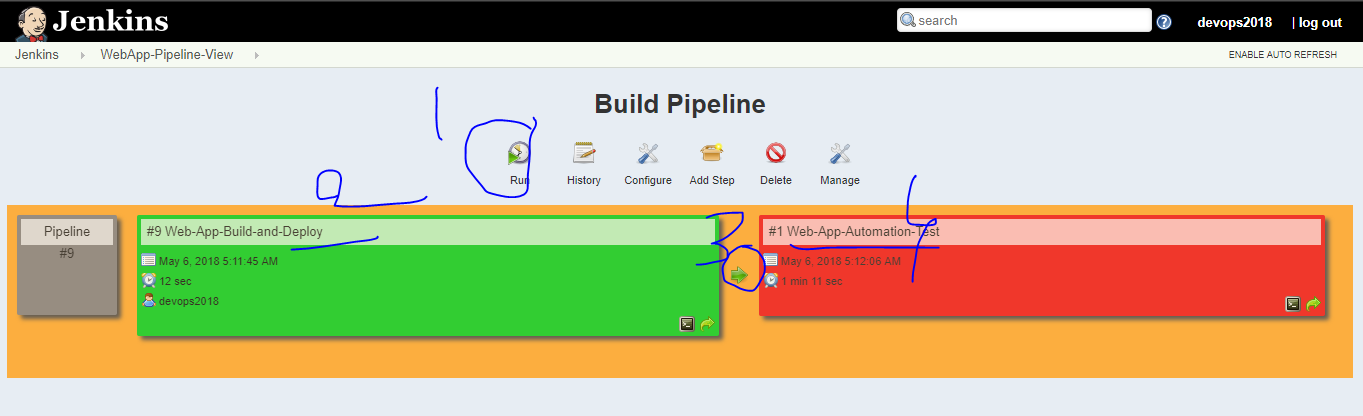
Select the upstream job (the job-1 which should trigger first)



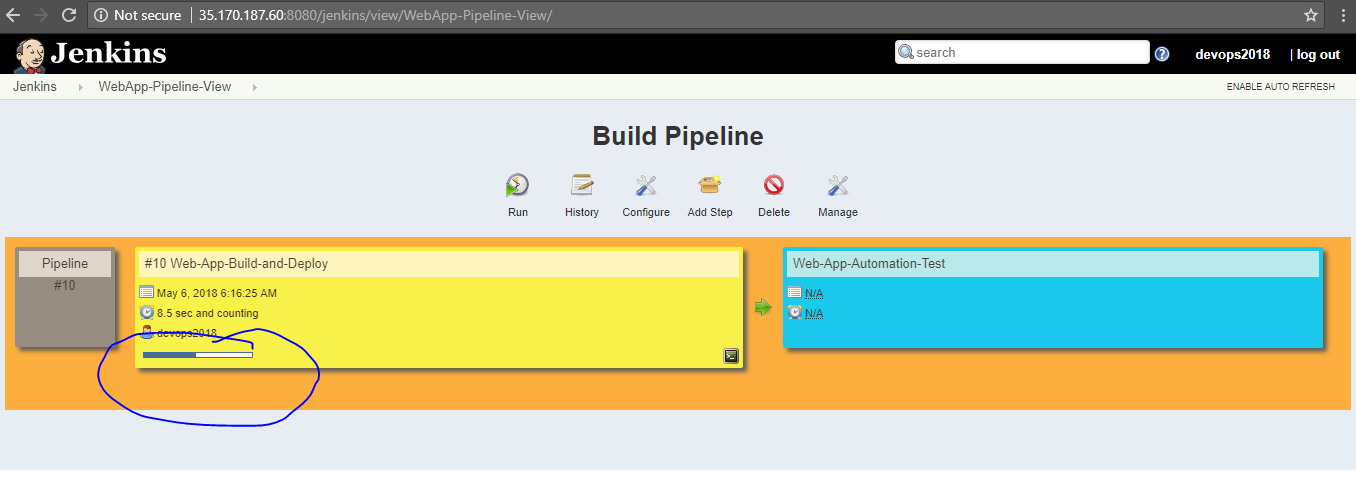
Save the configuration & go to build pipeline view.



1. To initiate the pipeline. Once we clicked on “Run”, the first job-1 will be triggered.
2. The first job with some details build number, date time last executed etc.
3. After the first job execution done successfully, then it will go to the downstream job to initiate.
4. The second job is in pipeline.



Click on “Run” 🡪 see the below image 🡪 yellow color represents the build in-progress 🡪 the status bar of the each project 🡪 blue color means yet to trigger the job in the same pipeline.



Both the jobs are success.

