# Execution of Tests on Jenkins using Docker

1.1 Problem Statement

1.2 How did you overcome

1.3 Details of Dummy Project

1.4 Details of the Jenkins job

# 1.1 Problem Statement:

In Verification and Validation phase , dedicated Virtual VM’s are required to execute the test scripts which requires cost effective and due to adequate VM’s , every team member needs to wait to get their turn ( approx. 1 to 2 days ) to get VM’s for execution.

In addition to above , below are the additional problems faced during Verification and validation phase

* Every VM requires Microsoft Window Licenses ( More Cost Required to buy the Licenses )
* Automation Execution time is more due to less VM’s availability.
* Sometimes , it is required to execute the scripts across different browsers. Different Browser Setup in a single VM takes more time consuming.
* Browser Version upgrade / Degrade is required because of combability issue.
* Perform Browser Upgrade / Degrade manually takes time
* Firewall Issues in VM’s
* Port Issues in VM’s
* When Jenkins Jobs are triggered ,Every VM should always be UP and Running to execute the scripts ,otherwise Jenkins waits for the next available Executor
* When Multiple VM’s are opened in Local Machine , the performance of the Local Machine becomes slower.

# 1.2 How to Overcome:

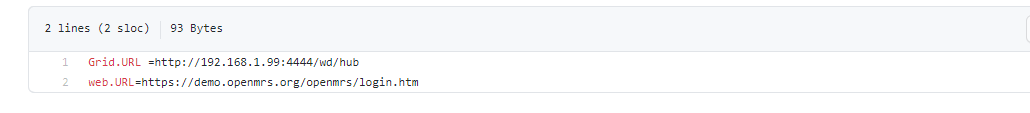
* Docker App is the best solution used to overcome the above problems
* Docker Containers are used to execute the Test Scripts
* Docker Provides the flexibility of creating Customized Docker Images for the Automation Project ( Irrespective of whether Test NG or JUNIT Framework ) and allows us to use the same Docker Image for Automation Execution
* Like Maven Repository , Docker also have Docker Hub where all the official Images for Selenium Grid , Selenium Standalone for Every Browser and Version are available in Docker Hub
* One can pull the Docker Images with simple commands in Command Prompt.
* Change of Browser Version and Browser setup is very easy in Docker (one simple command creates the appropriate Browser and Browser Version )
* No Licenses are required
* Based on Need , one can simply bring UP the Machines and can kill the VM’s after Successful Execution
* Docker creates Light weight Linux VM’s for execution and requires very less space ( size is in MB’s )

# 1.3 Details of Dummy Project

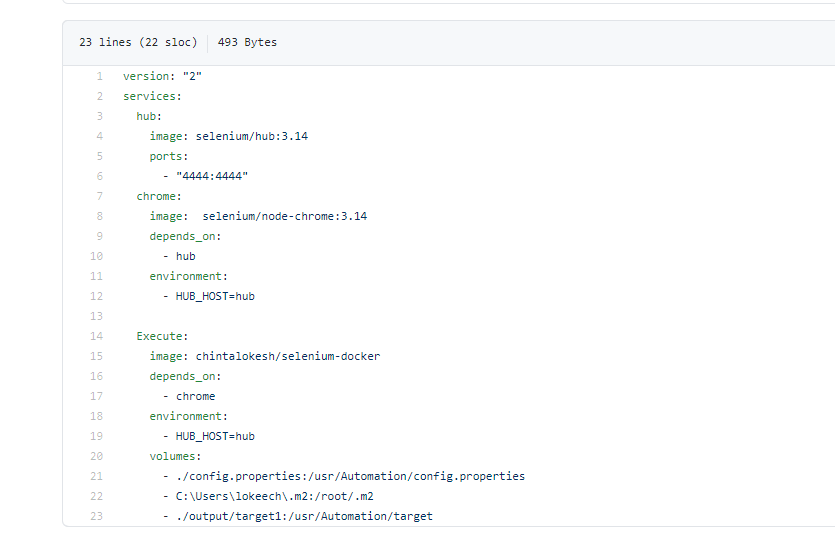
* Website : <https://demo.openmrs.org/openmrs/login.htm> is used to automate the feature
* Two Features are Automated which are given below
* Screenshots are captured and results are archived using Extent Report Library.
* Feature1:
* Login to Website using username as “***Admin***” and password as “***Admin123***”, select option as “***Inpatient Ward***” and click on “***Log In*** “button
* In Home Page, navigate to “Find ***Patient Record*** “page and search the patient’s name with “*John Smith* “
* Verify the Name is displayed in UI after successful search
* Logout and close the Browser
* Feature2:
* Login to Website using username as “***Admin***” and password as “***Admin123***”, select option as “***Inpatient Ward***” and click on “***Log In*** “button
* In Home Page, navigate to “Find ***Patient Record*** “page and search the patient’s name with “*John Smith* “
* Get the Vitals of the Patient displayed in UI
* Automation Code for the above Features are checked in to GIT HUB Path for reference: https://github.com/lokeshChinta/selenium-docker
* Jenkins Job Files are checked in the following GIT HUB Location: https://github.com/lokeshChinta/selenium-runner-testng-docker
* IMPORTANT Note: Requesting not to change the code in the above GIT HUB Project path
* please FORK the project and then use the GIT HUB Project location in Jenkins Job.

## 1.3.1 Changes required for the FORKED GIT HUB Project

* In “ ***Config.properties*** “ file , change Ip address ( IP Address of the Machine where Docker is Installed) to the “Grid.URL” Property .



* In docker-compose.yaml file , Under ***Volumes*** Section , Change the .m2 folder location to Local Machine .m2 folder location
* The Main use of mapping .m2 folder location is to avoid downloading the dependencies again and again



# 1.4 Details of Jenkins Job

* Jenkins Team provided official Docker image to work with Jenkins named: jenkins/Jenkins:lts
* Use the command in command prompt to get the Jenkins : [ docker run -p 9090:8080 -p 50000:50000 -v “<Location in Host Machine>:/var/jenkins\_home” jenkins/jenkins:lts ] to get the Jenkins Docker image download in local machine

Note : ***Jenkins/Jenkins:lts*** where ***lts*** refers to Long Term Support

Note : /var/jenkins\_home is the Jenkins container location exposed by Jenkins Docker Image

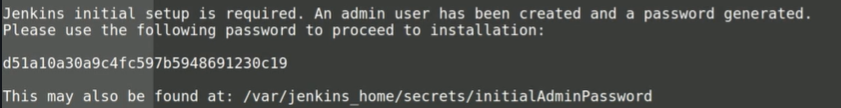
Note : <Location in Host Machine> can be anything like : “ **C:\Users\jenkins “**

## 1.4.1 Need of Volume Mapping

* As we know , Docker Container is just instance of an Image. So , when we run Jenkins ,we might be creating lot of jobs. Jenkins uses “ ***/var/Jenkins\_home***  “ location to create the jobs.
* As a reason , we are mapping the same in our host directory .In future , let’s say if the host machine is restarted (or) somehow the Jenkins container is killed and if we issue the command without volume mapping then Docker is going to create brand new Jenkins
* So , whatever we have done so far , everything lost including the jobs
* To avoid the above issue , and by using volume mapping , the host directory acts as a database
* Therefore , if we run the above command ,brand new container will gets creating by taking all the file from the HOST directory.
* Finally , Jenkins understands that this guy is created all the jobs and will use the same and behaves like **Existing Jenkins**.

## Installing Jenkins using Docker

* When the command is triggered in Command prompt , Jenkins will take few minutes to get the plugins installed in local machine for the very first time because Jenkins is a server.
* The Initial password for Jenkins will be displayed in the CMD.
* Below is the screenshot for reference
* Keep the password handy and the same password will be used while logging into the Jenkins for the first time.

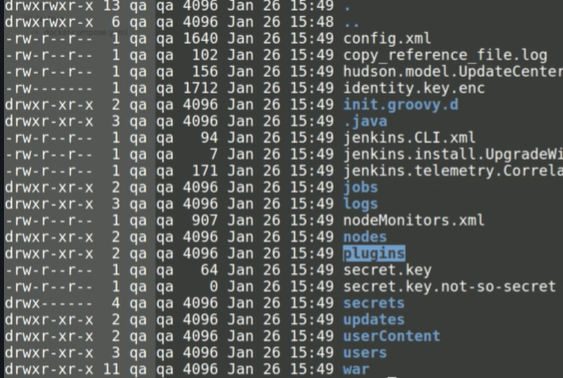


Important Note : When you do the volume mapping for Jenkins , for Instance, c:\users\jenkins

Requesting to please take care of below points

* It’s better to create the above directory under the “ ***users*** “ directory
* Create the Jenkins directory manually and give the full privileges to the directory.

Below are the files created by Jenkins and can be seen in Host Directory

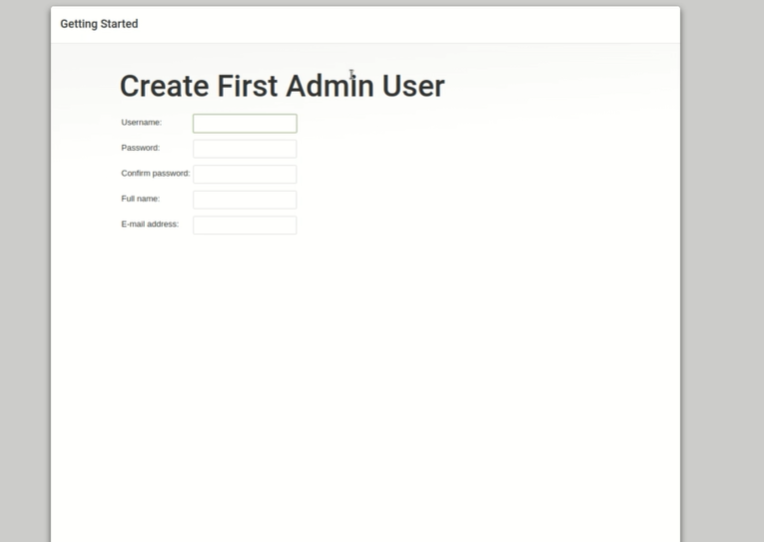


## 1.4.3 setting up Initial Plugins

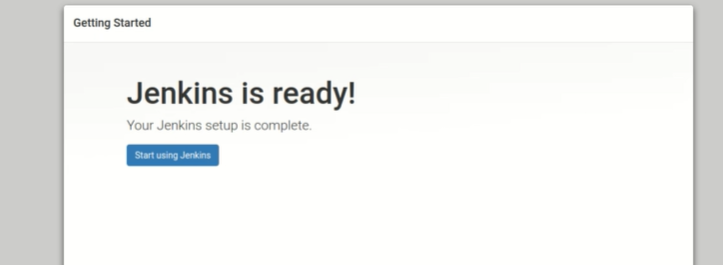
* Open the Browser and hit the URL : <http://localhost:9090> ( or) <http://192.168.1.6:9090> to check the Jenkins home page as the port was mapped to 9090 in the above Docker Command.



* Above is the very first screen of Jenkins
* Provide the initial password that got saved earlier and click on continue
* Jenkins will take few minutes to install the plugins for the very first time
* Once the plugins are downloaded , the Jenkins will take us to the below screen

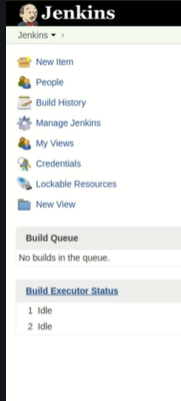
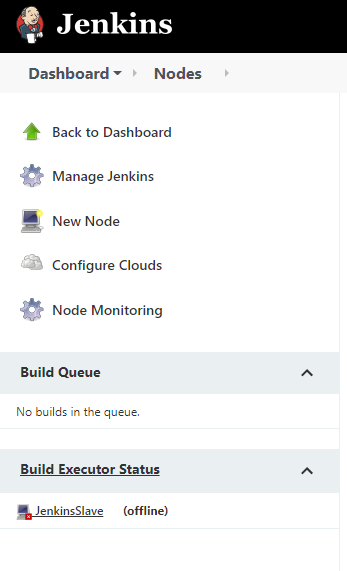


* Fill all the required details and click on “ ***Save and Continue*** “ and finally Jenkins will show the message stating that “Jenkins is ready! “

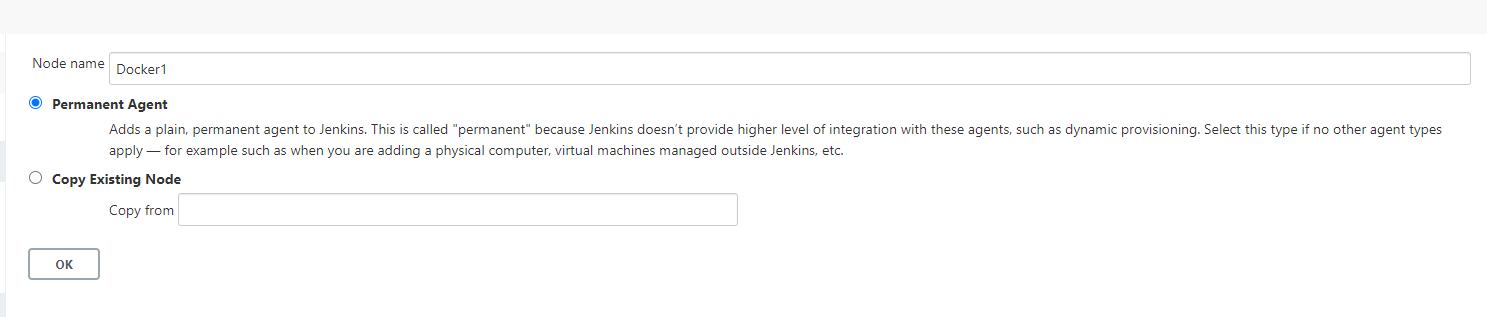


## Connecting to an Agent

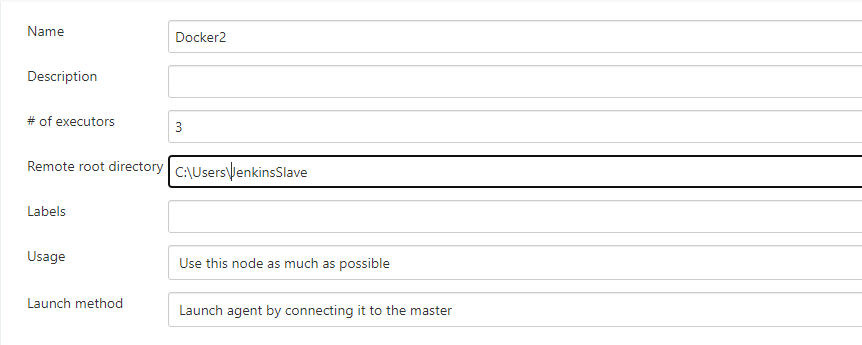
* Navigate to “ ***Build Executor status*** “ and click on “ ***New Node*** “ to create the New Node

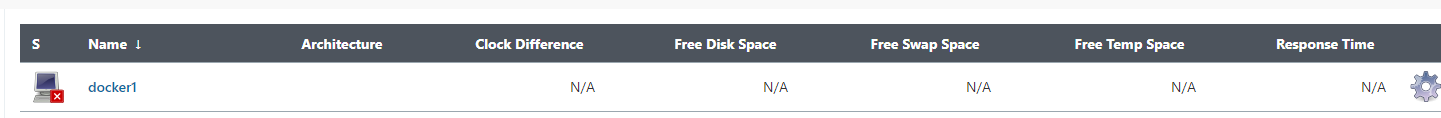
 

* Give Node name and select “ ***Permanent Agent*** “ and click on “ ***OK*** “ button

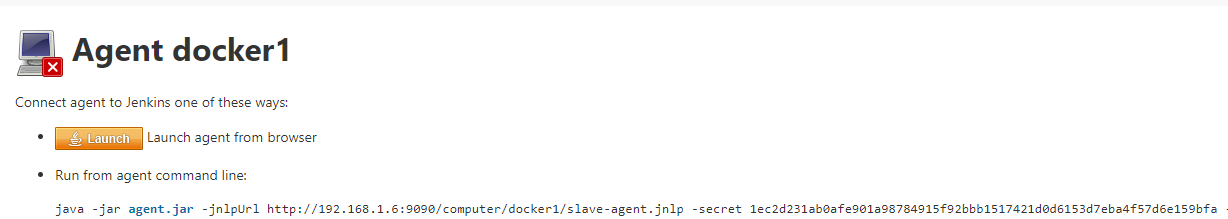


* # of executors is “ ***3*** “ , Remote root directory as “ ***C:\users\jenkinsSlave*** “ and click on “ ***SAVE*** ”
* Jenkins will keep all the project related files(.JAVA Files ) in the above Host directory (i.e., **jenkinsSlave** )





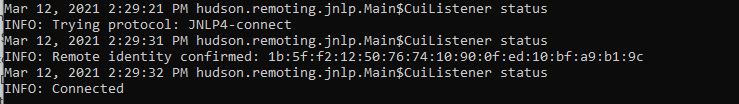
* Agent “**docker1** “ is created but the default status will be shown as “ **OFFLINE** “



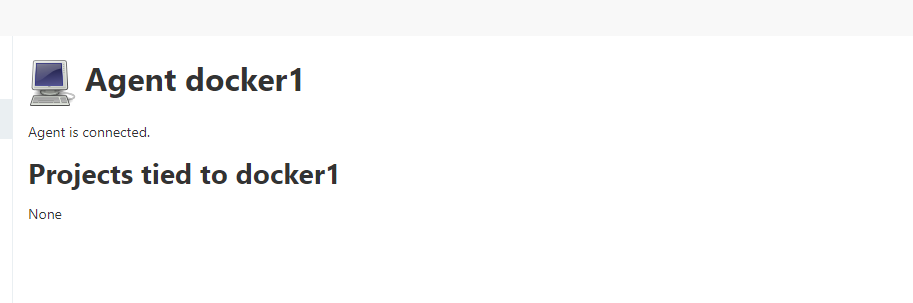
* To bring UP the Agent , click the ***agent.jar*** file link and the .jar file gets downloaded in the “ ***Downloads*** “ directory of Host Machine
* In CMD , Navigate to Downloads directory and fire the java command :

[ java -jar [agent.jar](http://192.168.1.6:9090/jnlpJars/agent.jar) -jnlpUrl http://localhost:9090/computer/docker1/slave-agent.jnlp -secret 1ec2d231ab0afe901a98784915f92bbb1517421d0d6153d7eba4f57d6e159bfa -workDir "C:\Users\JenkinsSlave" ]

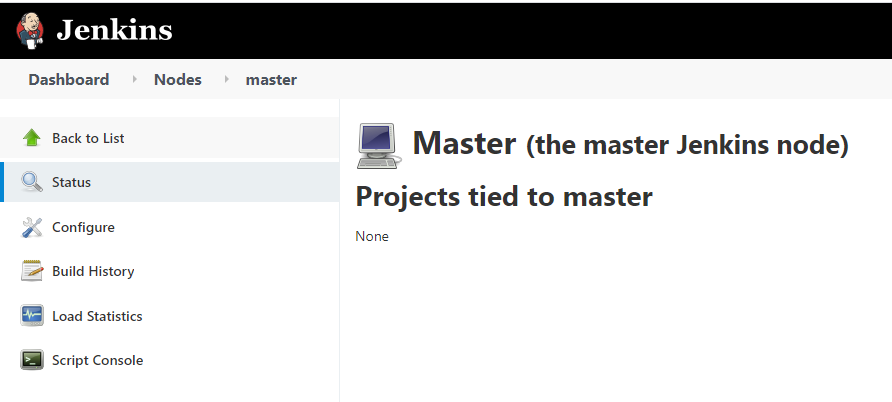
* In CMD , the status will be shown as “Connected “



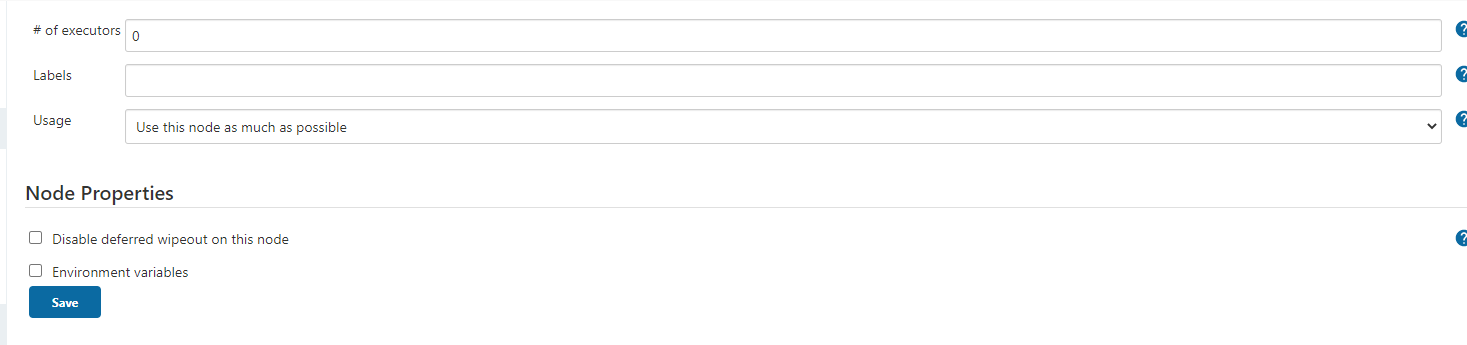
* Refresh the browser and you can see the Agent status as “Connected ”



* In Jenkins , there are two nodes namely Master and docker1 .open the Master node and click on Configure



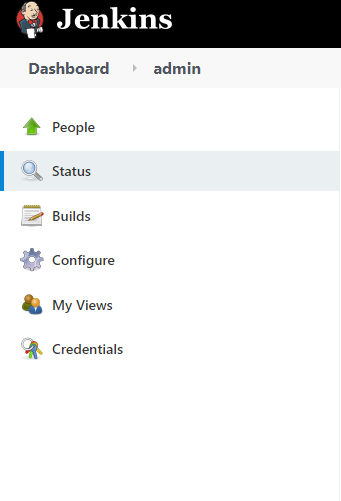
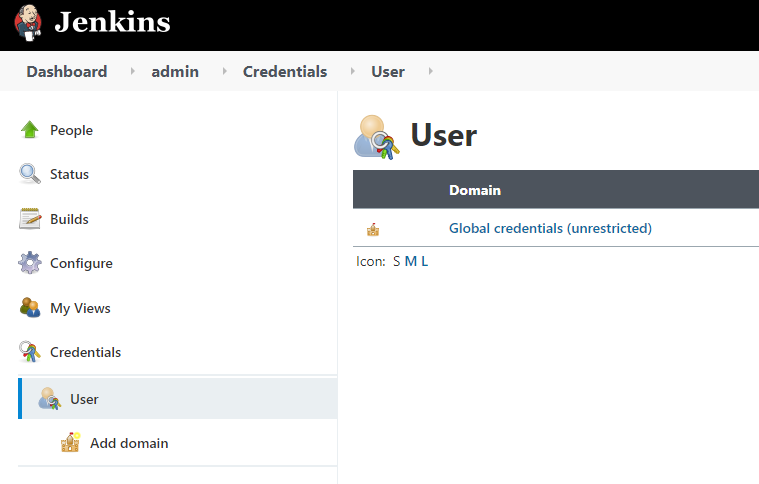
* Change “ # of executors “ to “ 0 “ and click on Save button

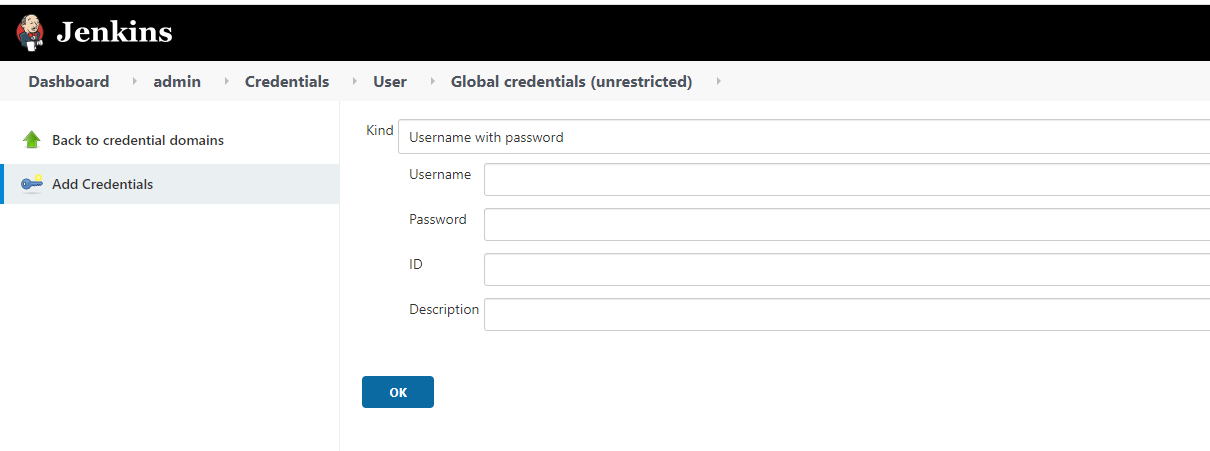


* The purpose of changing “ # of executors “ to “ 0 “ to Master is to balance the load so that Master doesn’t get overloaded.

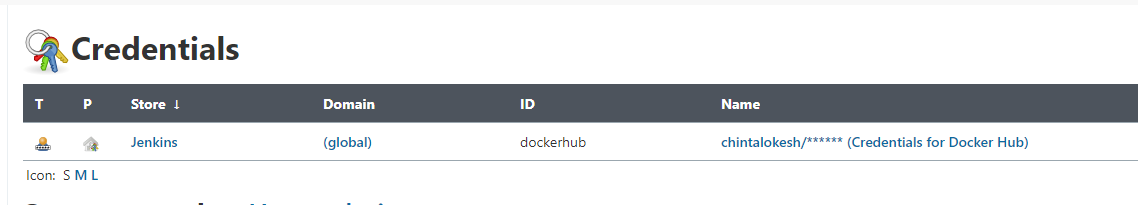
## Adding Docker hub Credentials

* To add Credentials, Navigate to Credentials -> Users -> [Global credentials (unrestricted)](http://192.168.1.6:9090/user/admin/credentials/store/user/domain/_) -> Add Credentials



* Docker Hub Credentials are required to push/Pull the Customized Docker Image (created for Automation Framework ) to Docker Repository.
* Provide Docker “ ***Username*** “ and “ ***Password*** “
* Keep ID as unique ( ID shouldn’t be having any spaces in between ) and click on OK
* For Instance, give ID : dockerhub

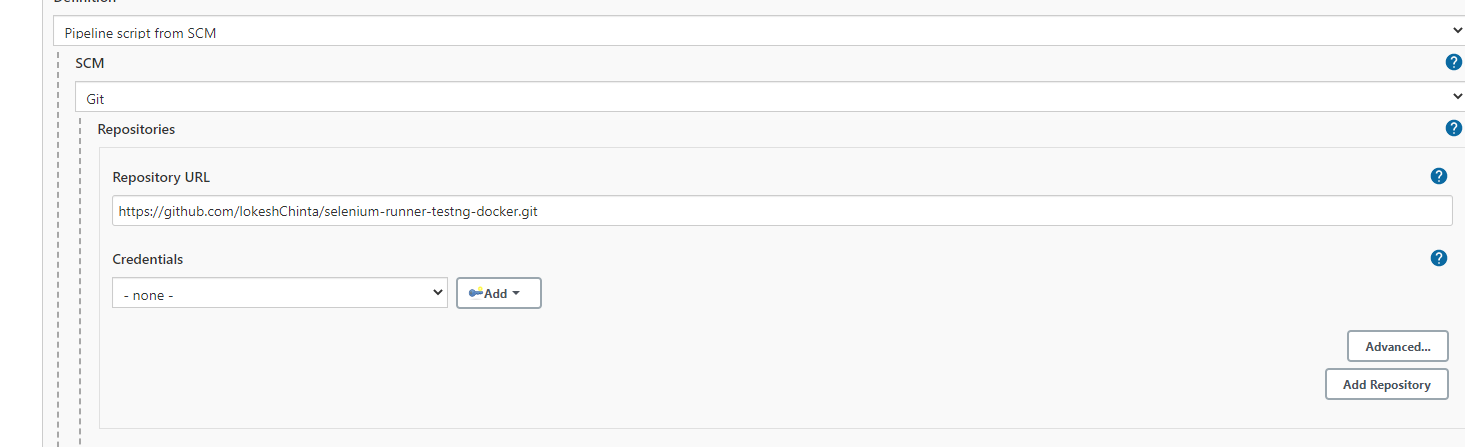


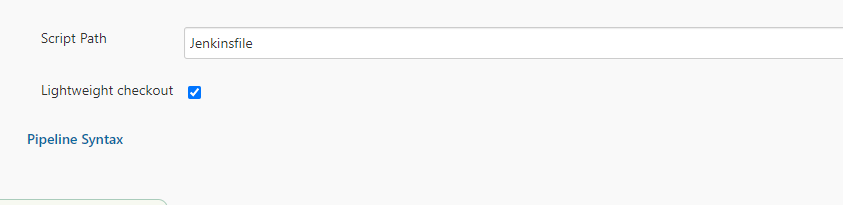
## 1.4.6 Create Jenkins Job for Running the Test Scripts

* To Create a New Job in Jenkins, navigate to “**New Item** “and provide name as “**SELENIUM\_ DOCKER\_RUNNER** “, select Pipeline option and click on “**OK** “

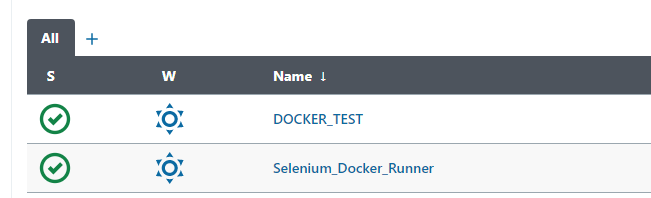
**Advantage of Pipeline option in Jenkins** :

* For Freestyle project , one must configure everything in Jenkins and in future , if we require another version of Jenkins then again, we need to configure right from the scratch.
* To overcome the above problem, Separate Jenkins file is created that contains all the instructions as part of the same file itself.
* In other words, All Jenkins Instructions are placed in Jenkins’s place and Jenkins will read all the Instructions from the file and process it accordingly which is the main advantage of Pipeline.
* In the Next window, ignore everything and go to pipeline section, Select GIT in SCM dropdown, provide GIT Automation project repository in Repositories and give credentials as None, select “Clean **before checkout** “in Additional behaviors, Script path as “Jenkinsfile” and click on “**SAVE** “button
* Note : As the GIT project is public one , no need to have credentials.
* Note : Other Credentials are required for Private projects
* GIT Automation project repository URL: [**https://github.com/lokeshChinta/selenium-runner-testng-docker**](https://github.com/lokeshChinta/selenium-runner-testng-docker) **(FORK the Project ,do the above required changes and use the GIT HUB Project URL)**

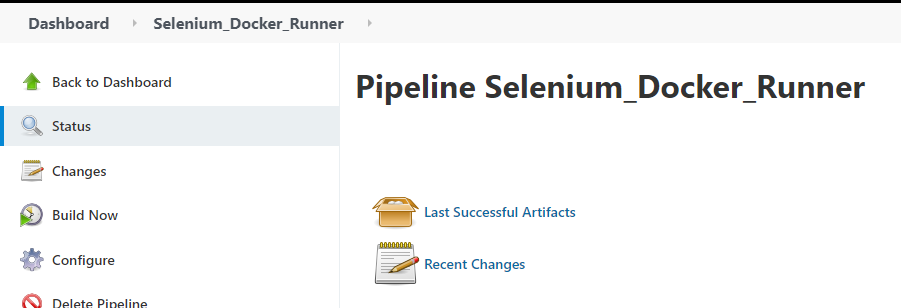




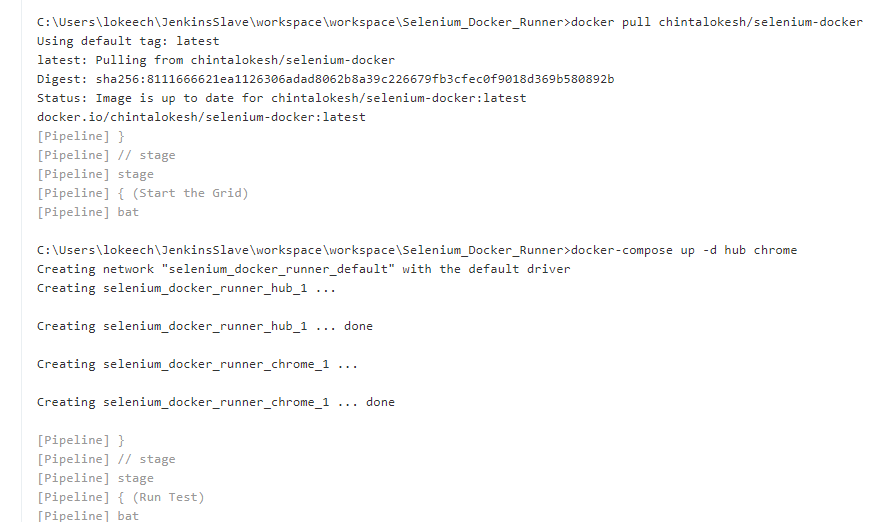
* Jenkins will use the file name called “**Jenkinsfile** “ (i.e., Pipeline script ) given in the “ **Script path** “ text box to build the project.
* Once it is saved successfully then SELENIUM\_DOCKER\_BUILDER can be seen in the Home page of Jenkins



* To aim of the “**Docker\_Builder\_RUNNER** “Job is to create the Selenium Grid Docker Containers, Run the Automation Scripts inside the Containers, Achieve the Results and kill the containers with Single Click
* Click on “Build **Now** “link to build the project



* The above **JOB** will pull the source code from GIT HUB, download all the Maven dependencies, Build the Project, Run the Scripts and kill the containers



## Achieve the Test Scripts

Once the Automation execution is successfully completed, go back to Main page of Jenkins Job and see the Results along with the screenshots

Below are the screenshots for reference

