Mindtree

CICD [JENKINS]

AUTHOR : HARISH R

Welcome to possible

Contents

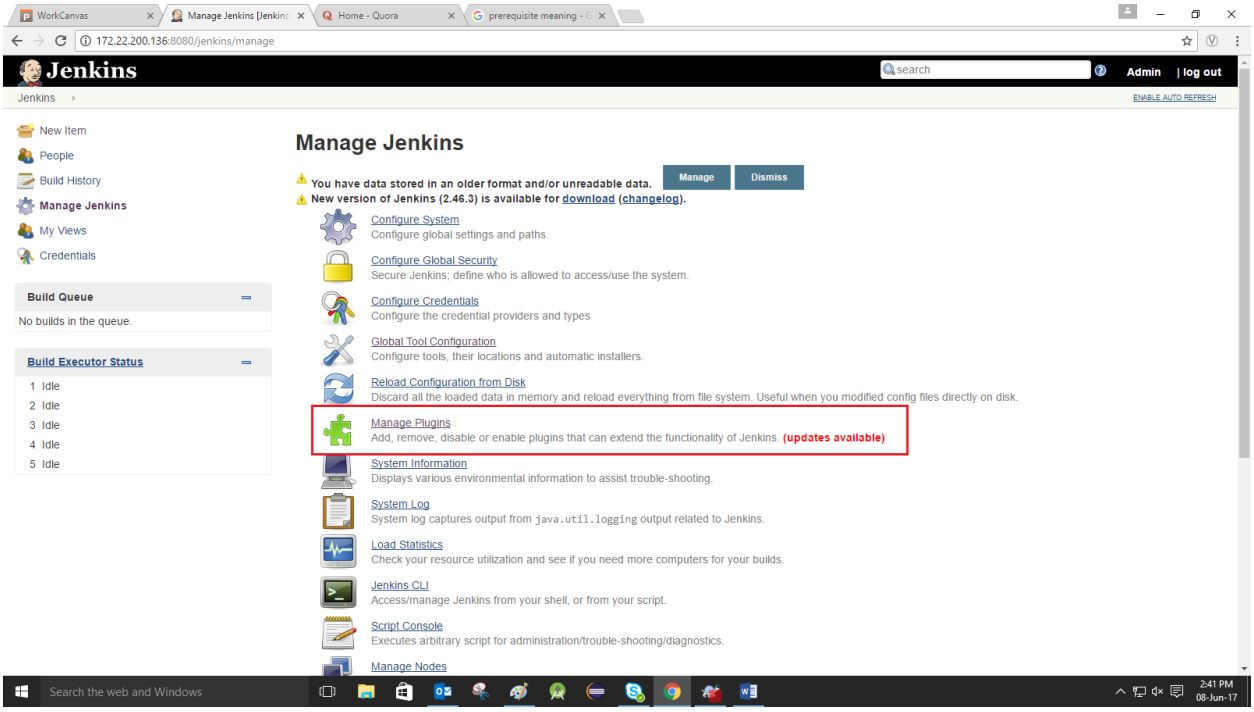
1. PREREQUISITES FOR GRADLE BUILDING IN JENKINS

Login into **Jenkins** using the provided credentials.

URL to access Jenkins: http://172.22.200.136:8080/jenkins/

Add below mentioned plugins in **Manage Plugins** under **Manage Jenkins** section:

- Gradle Plugin
- Android Emulator Plugin
- Artifact Deployer Plug-in

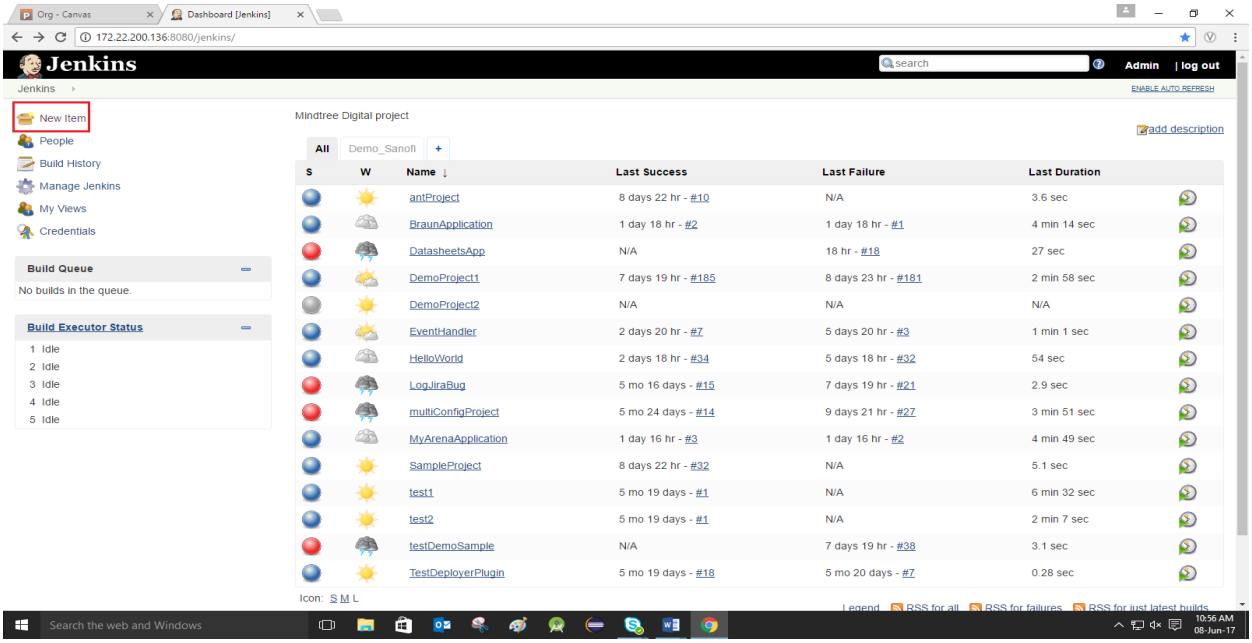


2. GRADLE BUILD FOR NATIVE ANDROID APPS IN JENKINS USING SVN REPOSITORY

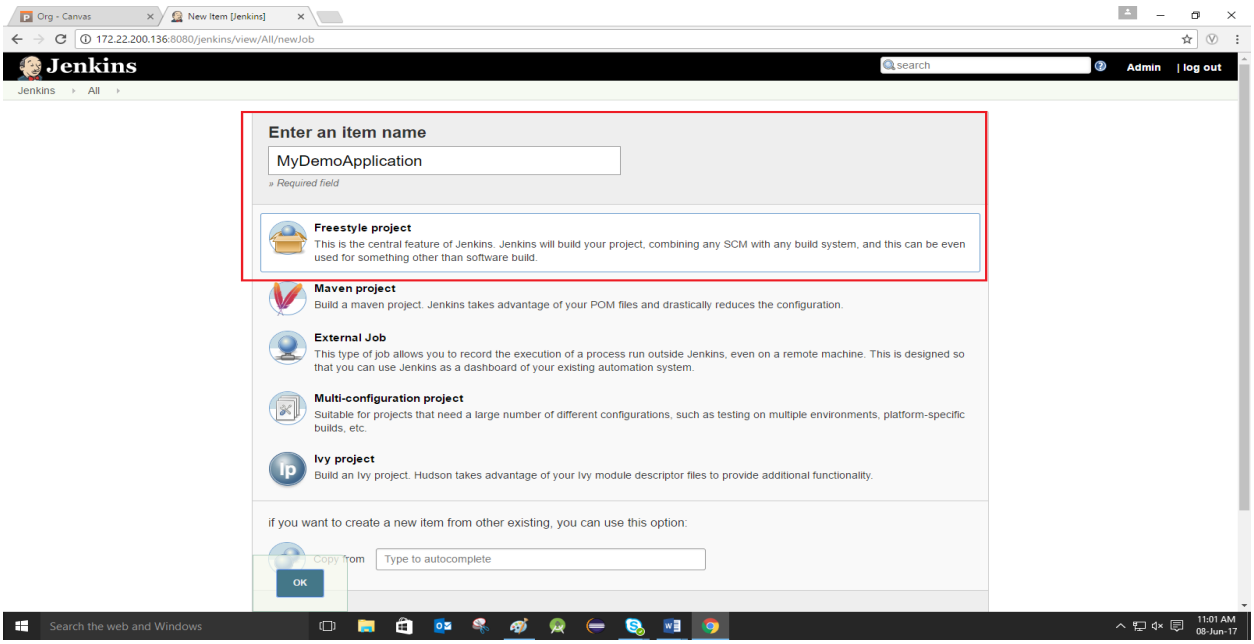
Steps to create a job in **Jenkins**

2.1. Login into **Jenkins** using the provided credentials.
URL to access Jenkins: http://172.22.200.136:8080/jenkins/

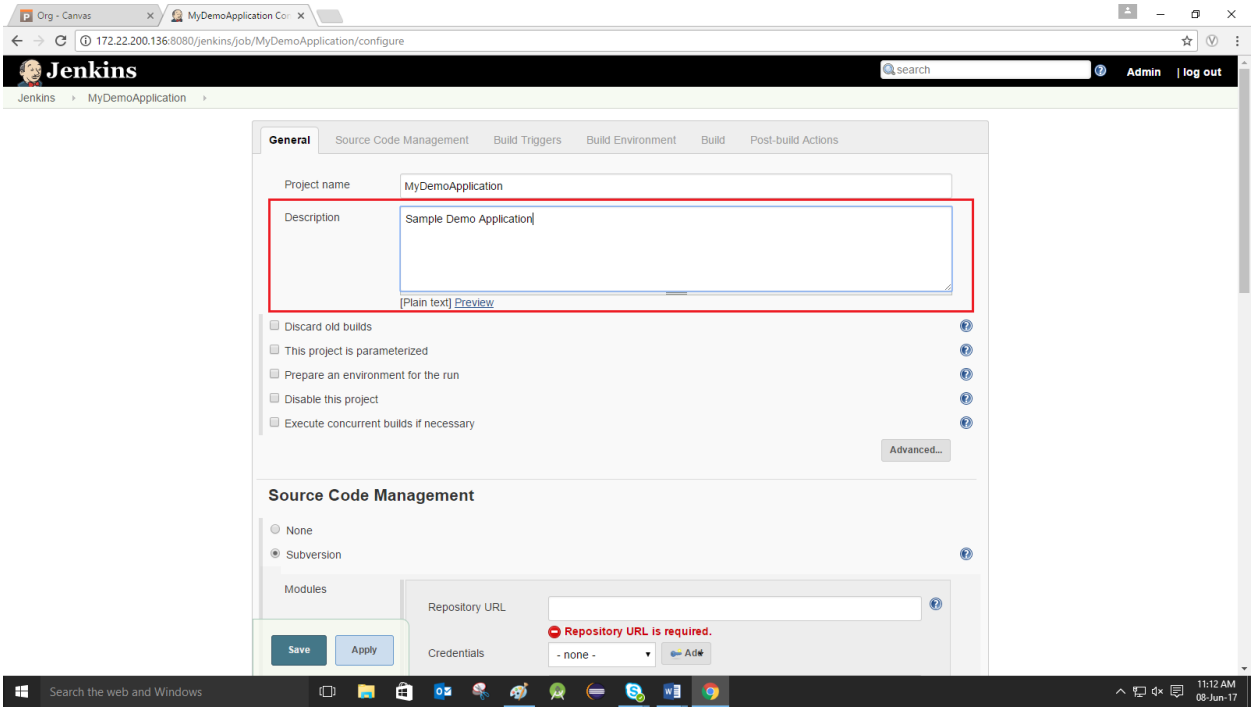
2.2. Click on **New Item**.



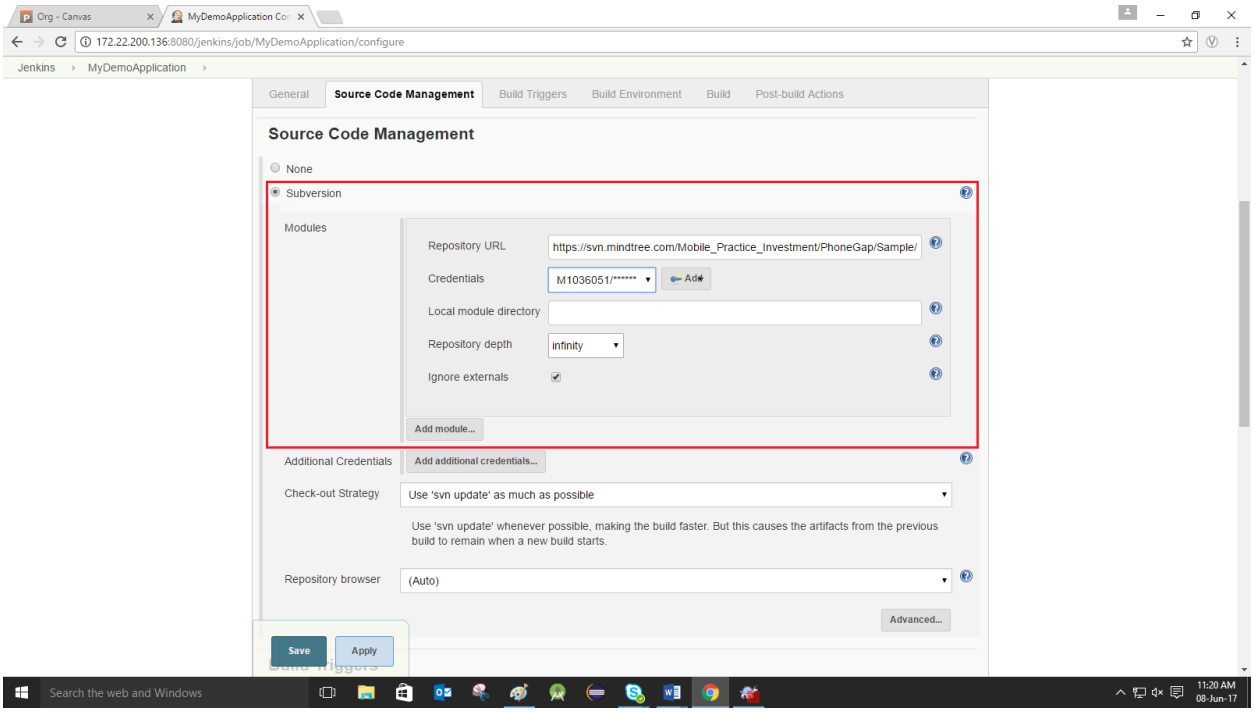
2.3. Enter an **item name** and select **Free style project**, click OK.

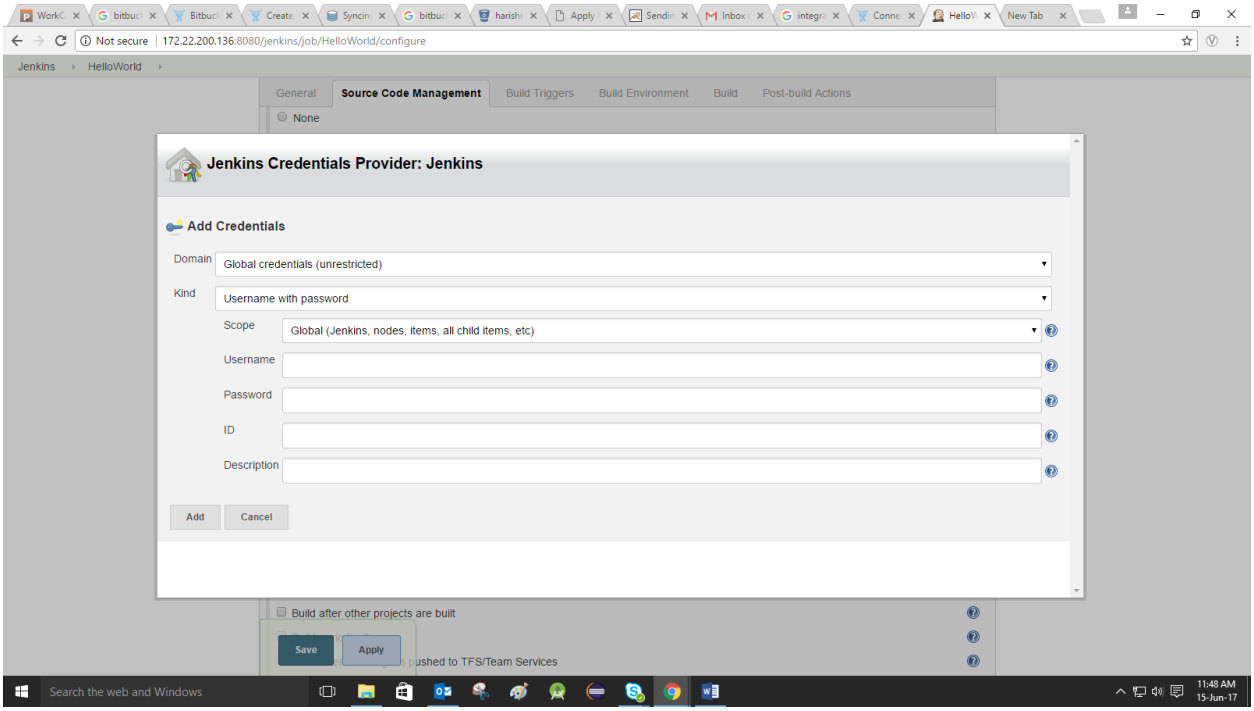


2.4. It automatically redirects to the job configuration page, enter the **description** of the job in the General tab.

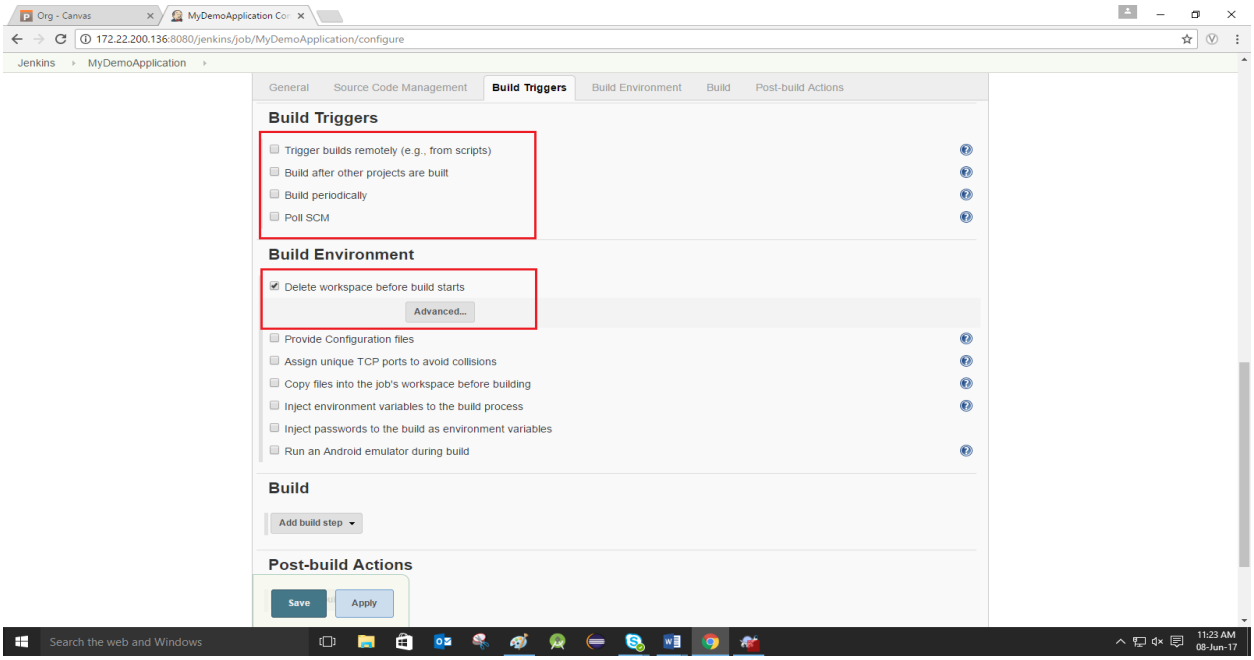


2.5. Select **subversion** in Source Code Management tab. Enter the path of the source code present in the **SVN repository**, add your respective **credentials** required for accessing the repo and make **Local module directory** empty.

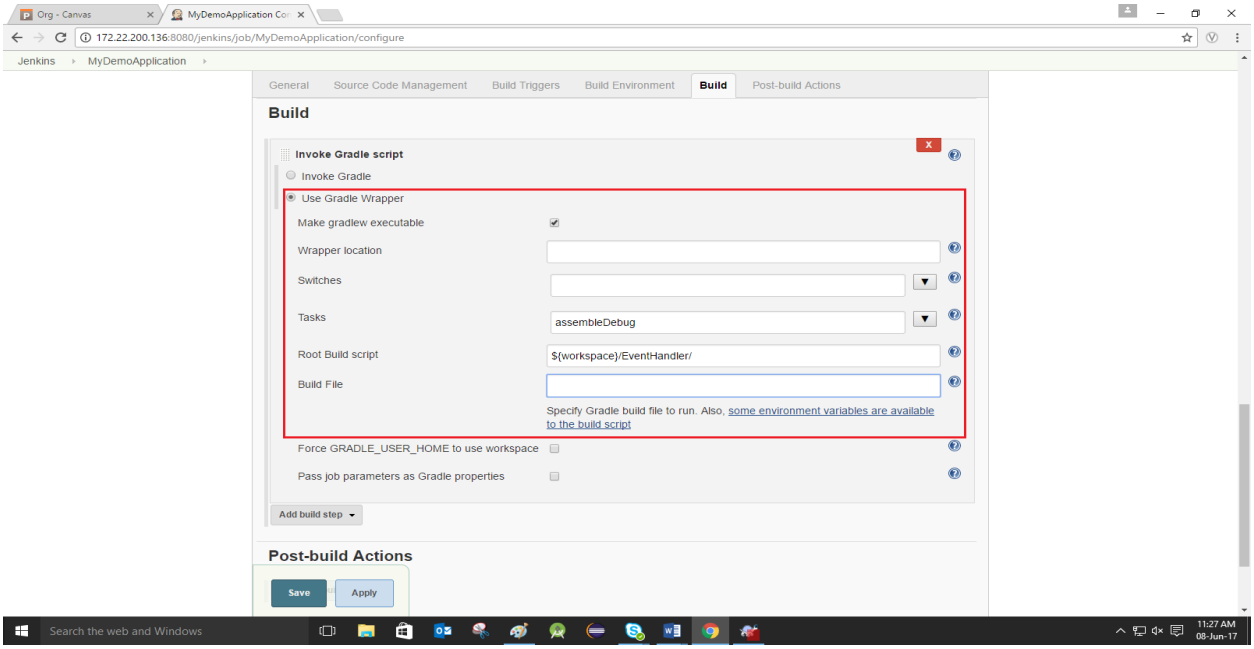




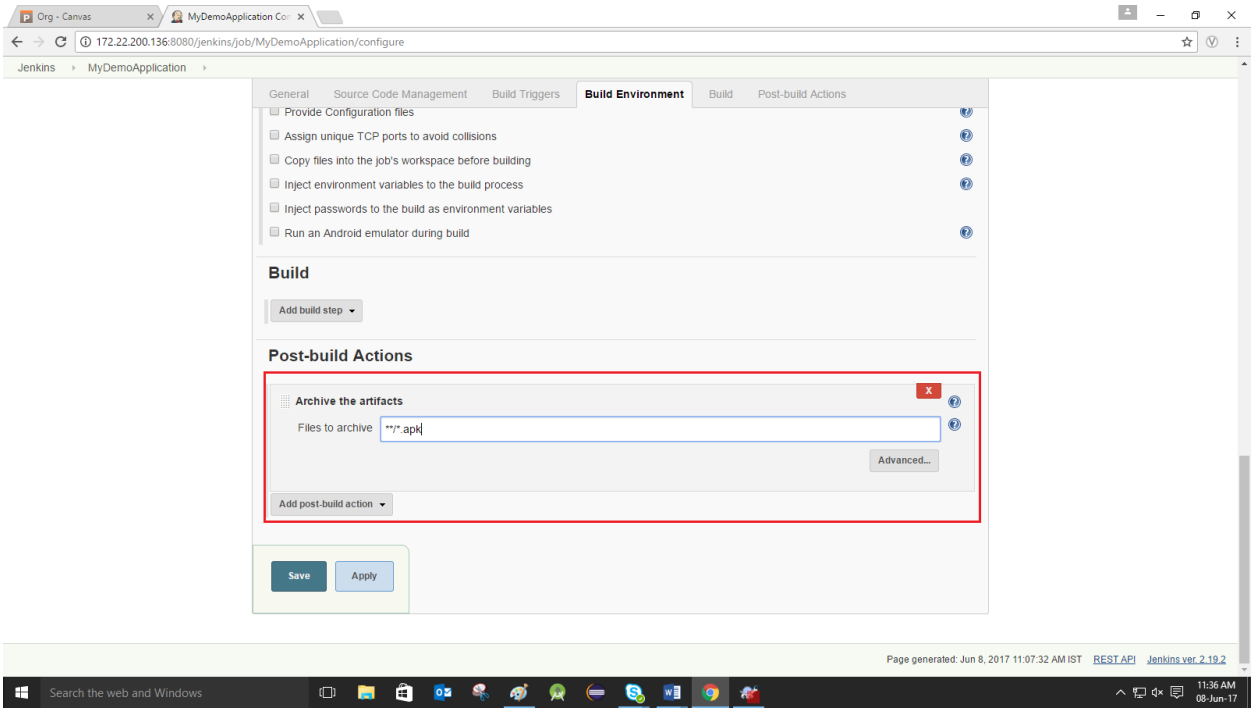
2.6. Don't make any changes in Build Triggers tab. Select **Delete workspace before build starts** in Build Environment tab.



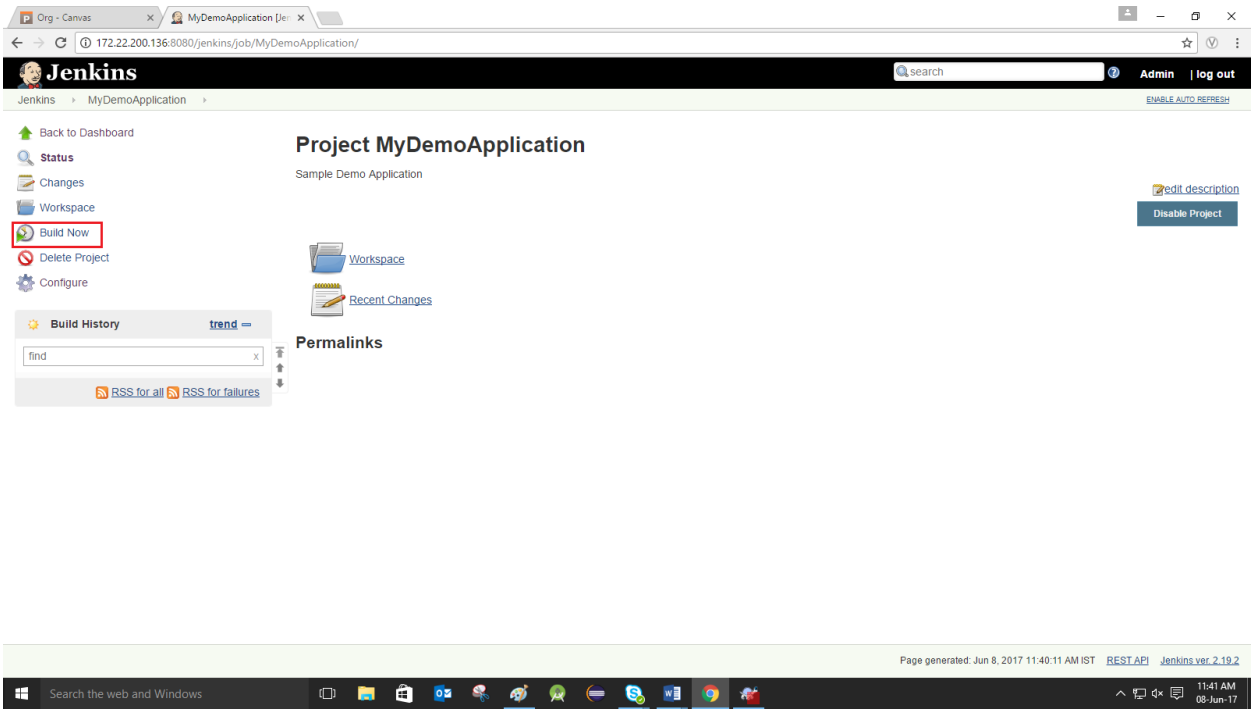
2.7. In Build tab, select **Invoke Gradle script** under **add build step**. Select **Use Gradle Wrapper** and check **make gradlew executable**. Give the tasks name as **assembleDebug** and root build script path as **${workspace}/<folder-name>/**.



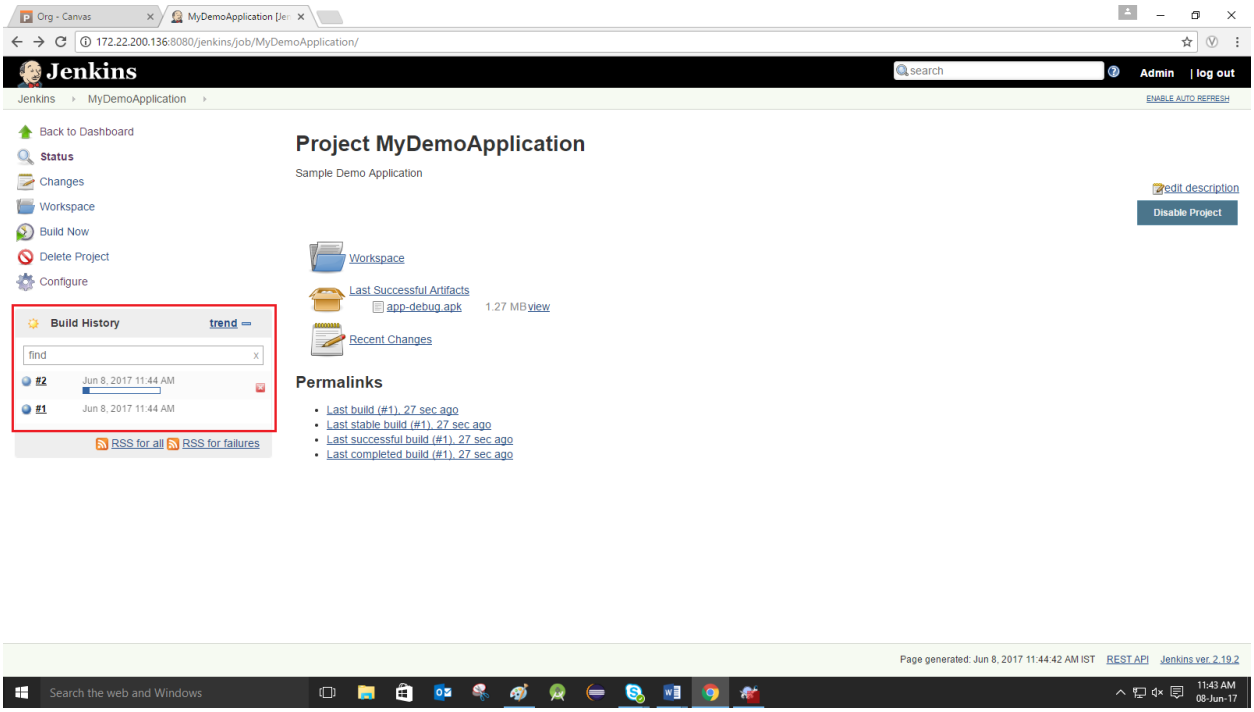
2.8. Select **Archive the Artifacts** under **add post build action**. Enter ****/*.apk** in the **files to archive** field for generating .apk file after build.

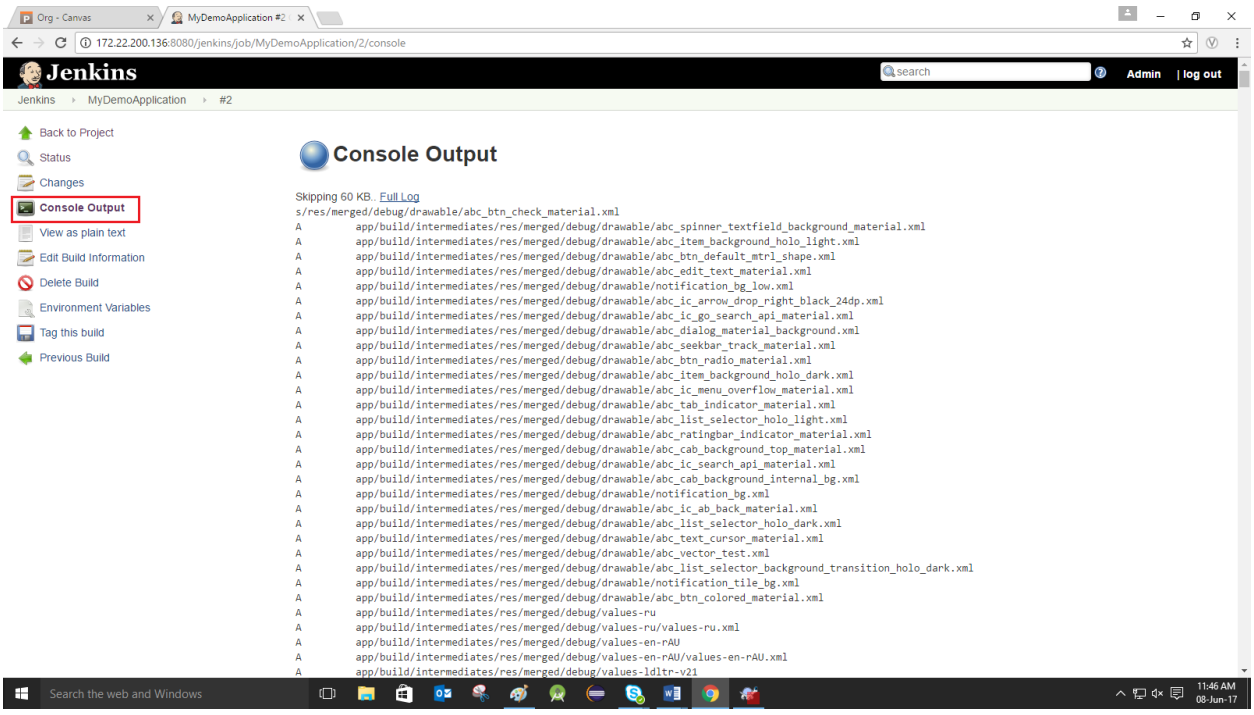


2.9. Click on **Save**, it redirects to build the job. Click on **Build Now**.



2.10. Once the build begins, it will be visible under **Build History**. Click on the build number, select **Console Output** to view the build result.



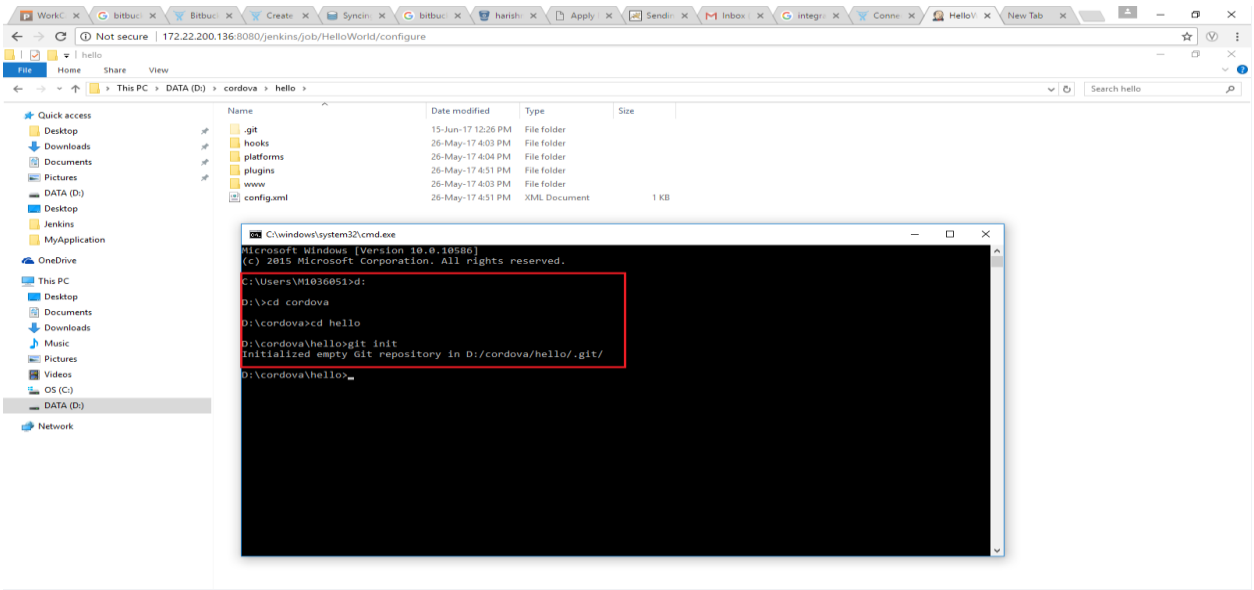


3. PUSHING SOURCE CODE FROM LOCAL DIRECTORY TO BITBUCKET REPOSITORY

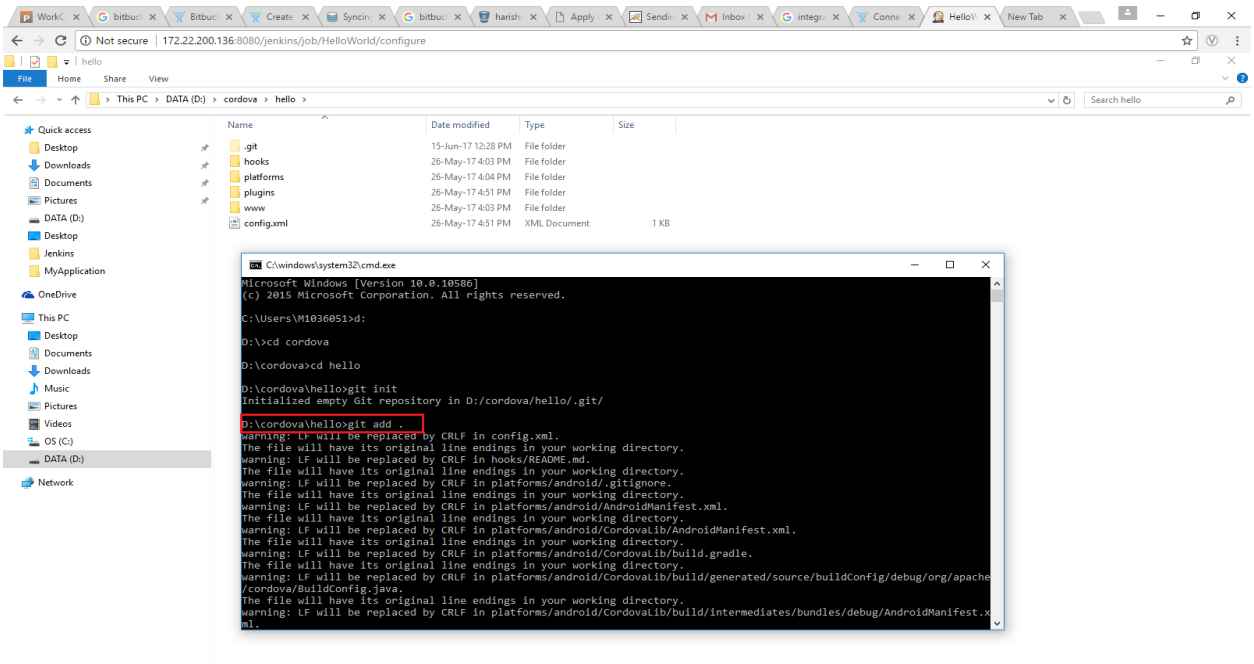
Follow the below steps:

3.1. After opening terminal window, navigate to your local project directory.

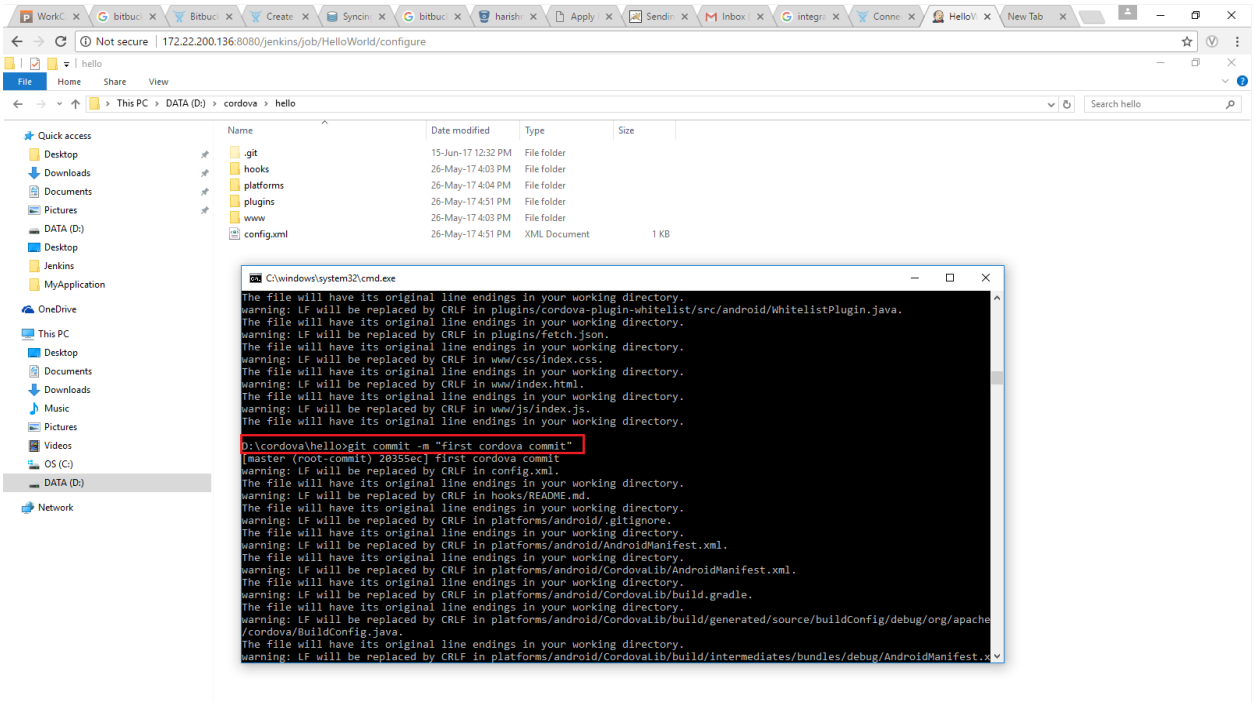
```
cd <project directory>
git init
```




```
git add .
```

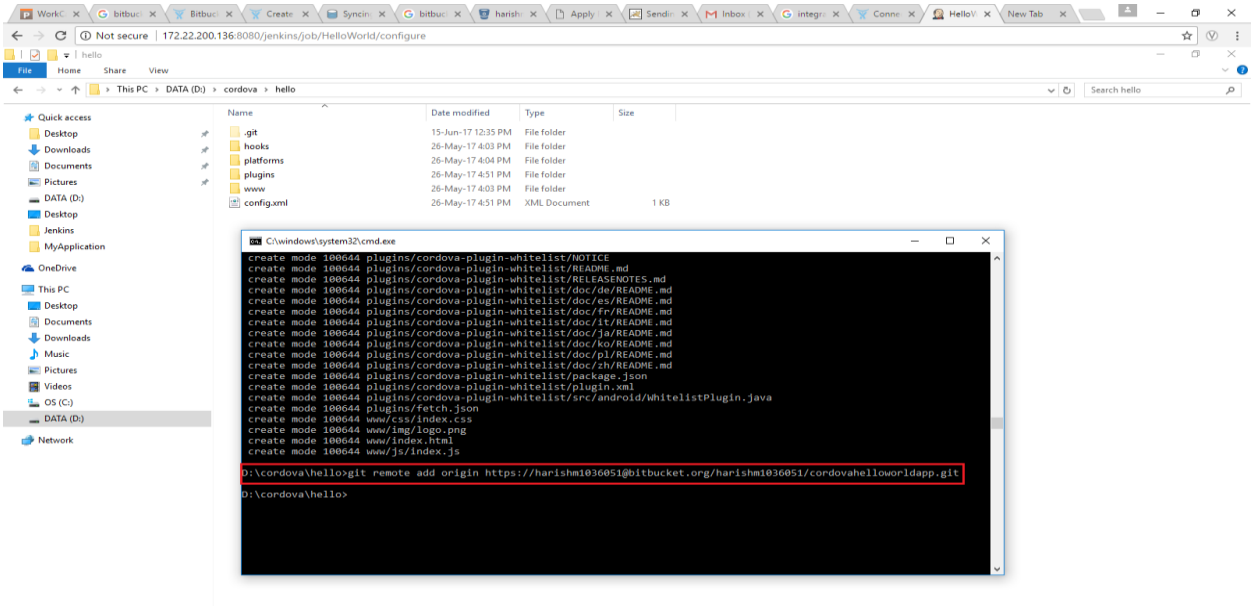


```
git commit -m "Initial commit"
```



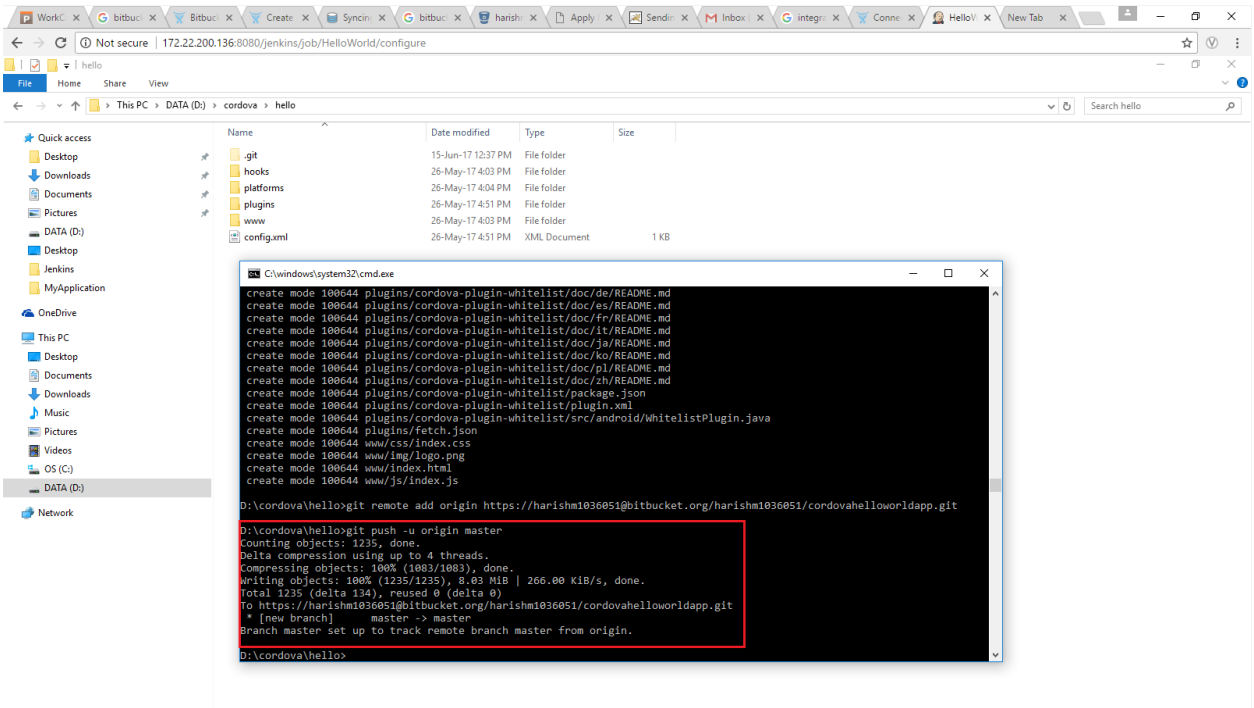
3.2. In the same window connect your existing repository to bitbucket repository.

```
git remote add origin https://harishm1036051@bitbucket.org/harishm1036051/cordovahelloworldapp.git
```

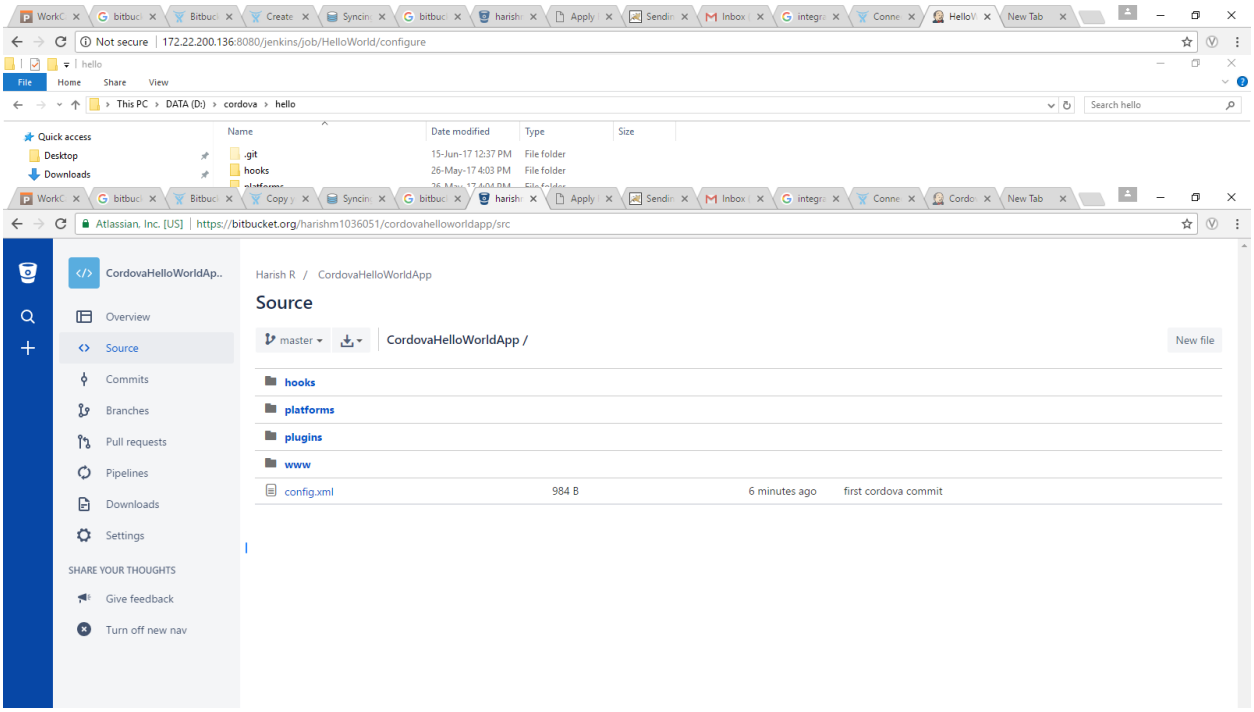


3.3. Push the source code into Bitbucket repository.

```
git push -u origin master
```



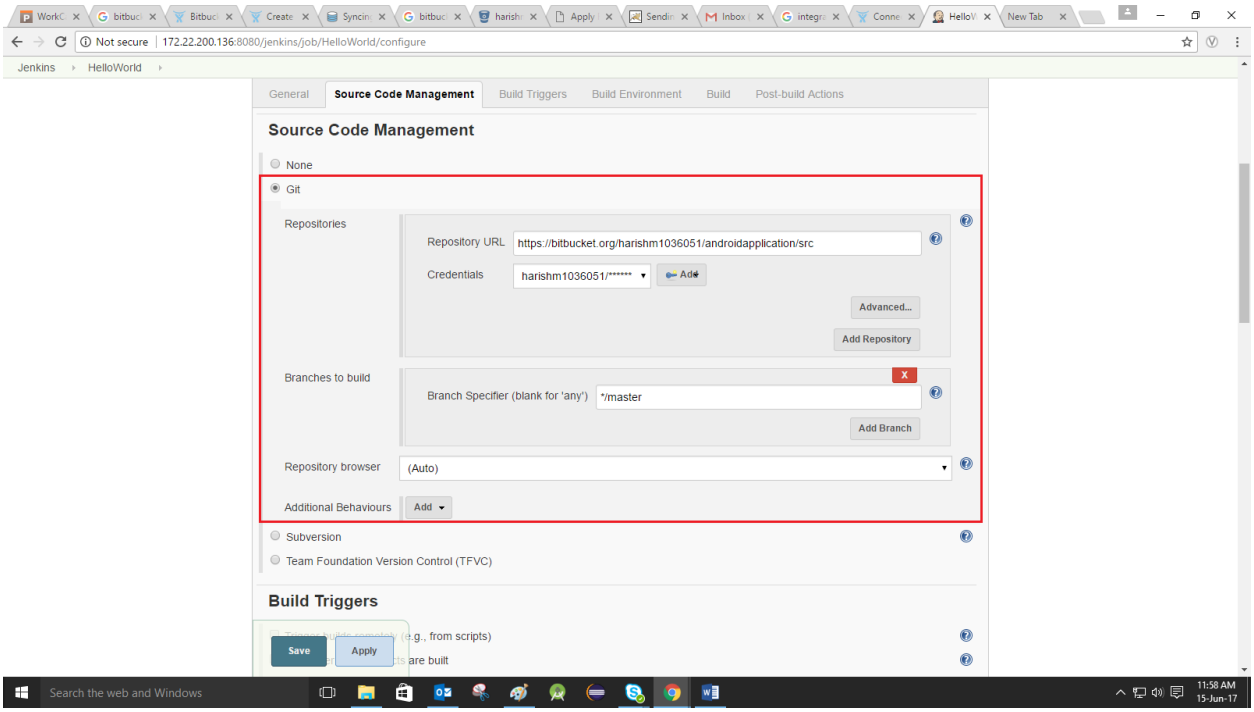
3.4. Bitbucket source code view after push from Local directory.



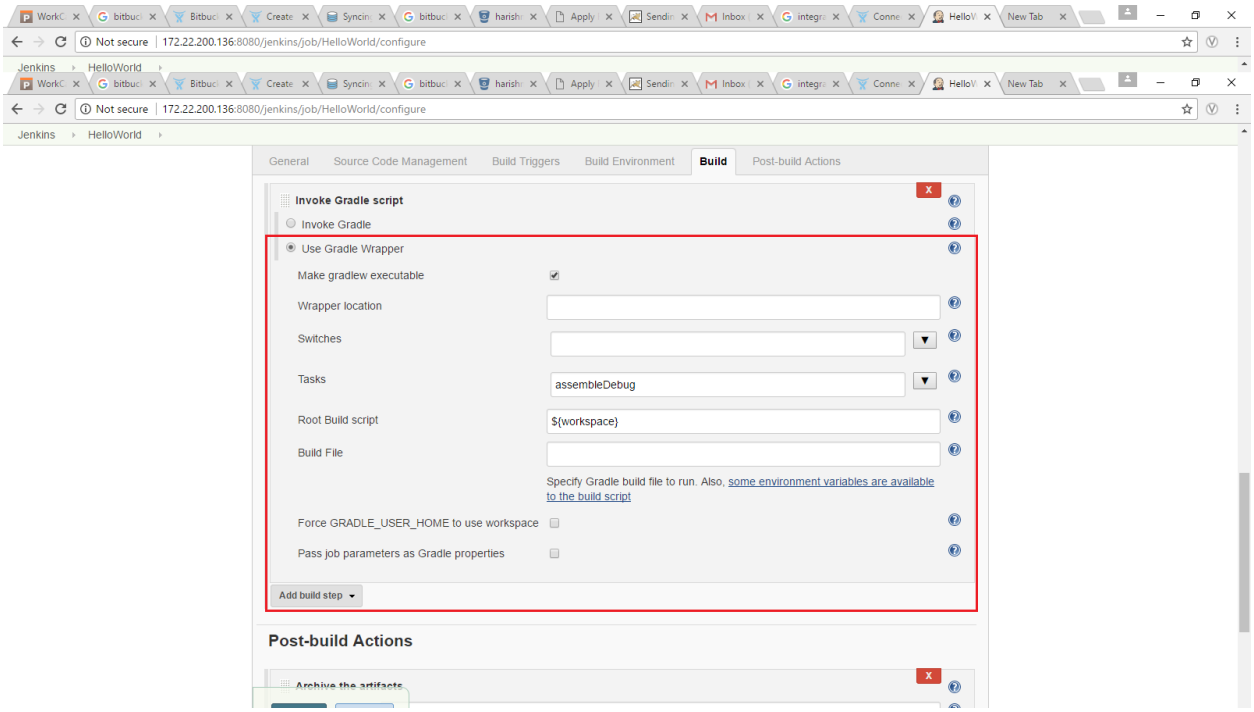
4. GRADLE BUILD FOR NATIVE ANDROID APPS IN JENKINS USING BITBUCKET REPOSITORY

Follow same steps that of **GRADLE BUILD FOR NATIVE ANDROID APPS IN JENKINS USING SVN REPOSITORY** mentioned above and the changes that must be done is as shown below:

4.1. Select **git** in Source Code Management tab. Enter the path of the source code present in the **Bitbucket repository**, add your respective **credentials** required for accessing the repo and make **Local module directory** empty.



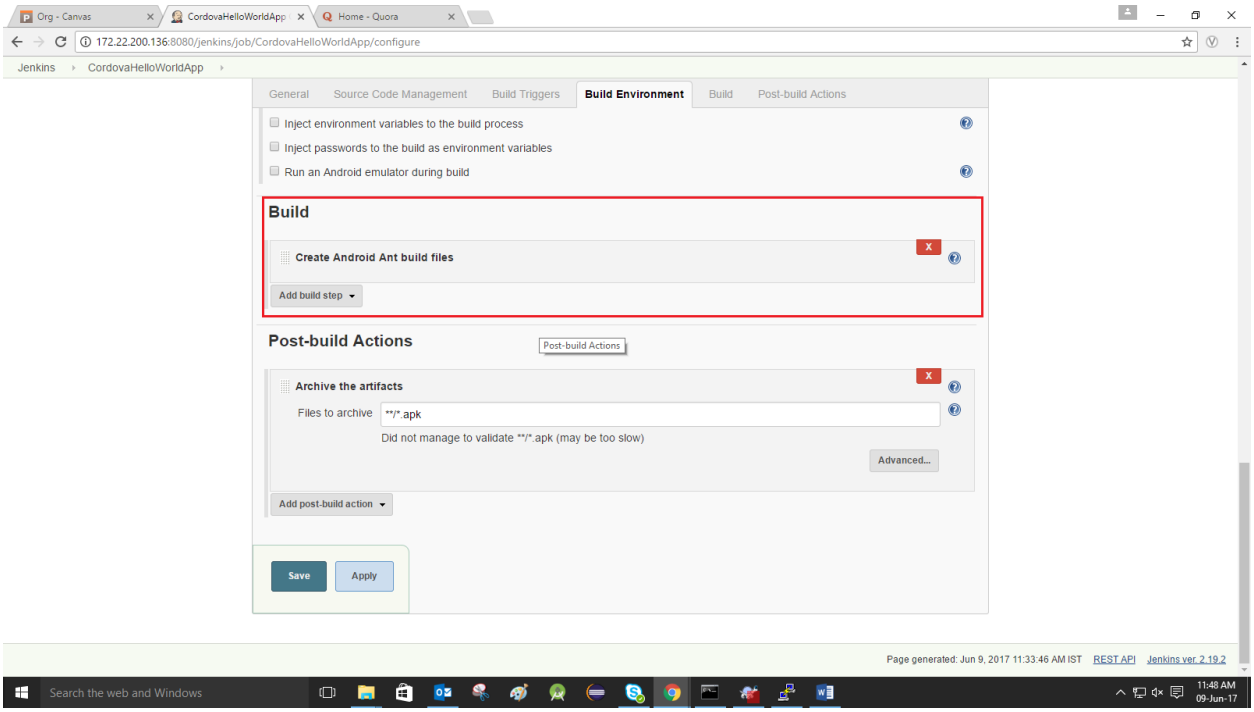
4.2. In Build tab, select **Invoke Gradle script** under **add build step**. Select **Use Gradle Wrapper** and check **make gradlew executable**. Give the tasks name as **assembleDebug** and root build script path as **${workspace}**.



5. ANDROID ANT BUILD FOR CORDOVA ANDROID APPS USING SVN REPOSITORY

Follow same steps that of **Gradle build for native android apps in Jenkins using SVN Repository** mentioned above and the only change that must be done is as shown below:

5.1. In **Build** tab of Jenkins job configuration, select **Create Android Ant build files** under **add build step**.

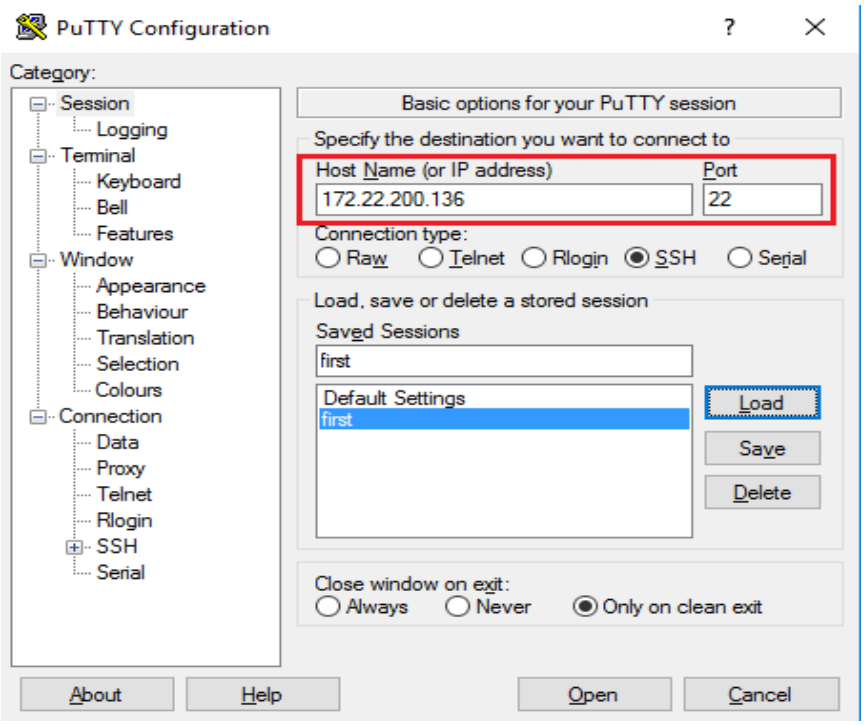


5.2. Click **Save** and build the job.

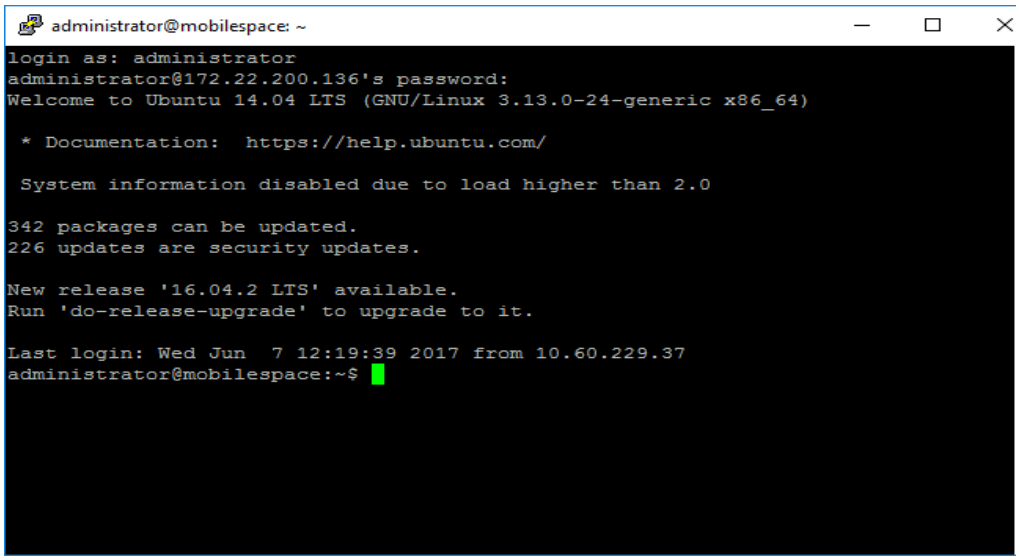
6. ESTABLISHING CONNECTION TO REMOTE SERVER

Steps for establishing connection to **remote server** using **Putty**:

6.1. Open putty configuration. Enter **hostname**, **port number** and click **open**.



6.2. It redirects to **putty CLI**, then enter the credentials for logging into remote server.



7. INFORMATION RELATED TO PATHS AND ISSUES IN UBUNTU SERVER

Path Information:

7.1. Android-sdk path: /usr/share/tomcat7/.jenkins/tools/android-sdk/
7.2. Jenkins workspace path: /usr/share/tomcat7/.jenkins/workspace/

Issues:

7.3. Permission issue: Use command sudo chmod 777 –R
7.4. Jenkins Dashboard access issue: Follow below commands to restart tomcat server

- cd /var/lib/tomcat7/
- sudo service tomcat7 stop
- cd logs/
- sudo rm –rf
- cd work/
- sudo rm –rf Catalina/
- cd webapps/
- sudo rm –rf jenkins/
- sudo service tomcat7 start

Android commands:

- android list sdk –all
- android update sdk –no-ui
- android update sdk –u –a –t <package number>

Note: Usage of this commands is in remote server after logging using putty.

8. SONARQUBE STATIC CODE ANALYSIS IN WINDOWS

Steps to configure **SonarQube**:

8.1. Download the desired SonarQube version of your project requirement from the mentioned URL https://www.sonarqube.org/downloads/.

8.2. Create a local directory of your choice, unzip the downloaded folder and place inside it.

8.3. Traverse to the bin folder from the unzipped SonarQube folder, from there open command prompt. Type **startsonar**.

8.4. The process gets started and it will show SonarQube is Up at the end of the process.

8.5. Open any browser and access the URL http://localhost:9000 . You will find the SonarQube Dashboard. By default SonarQube runs on port 9000.

Steps to configure **SonarQube Scanner**:

8.6. Download the SonarQube Scanner of desired OS from the mentioned URL https://docs.sonarqube.org/display/SCAN/Analyzing+with+SonarQube+Scanner.

8.7. Unzip the downloaded folder and place it in the local directory created previously for SonarQube. We'll refer to it as <install_directory> in the next steps.

8.8. Add the <install_directory>/bin directory to your path [Edit in the environment variables].

8.9. You can verify your installation by opening a new command prompt and executing the command sonar-scanner –h. You should get output like this

```
usage: sonar-scanner [options]

Options:
 -D,--define <arg>     Define property
 -h,--help             Display help information
 -v,--version          Display version information
 -X,--debug            Produce execution debug output
```

8.10. Create a configuration file in the root directory of the project: sonar-project.properties

sonar-project.properties file

```
# must be unique in a given SonarQube instance
sonar.projectKey=my:project
# this is the name and version displayed in the SonarQube
UI. Was mandatory prior to SonarQube 6.1.
sonar.projectName=My project
sonar.projectVersion=1.0

# Path is relative to the sonar-project.properties file.
Replace "\" by "/" on Windows.
# This property is optional if sonar.modules is set.
sonar.sources=.

# Encoding of the source code. Default is default system
encoding
#sonar.sourceEncoding=UTF-8
```

8.11. Run the following command from the project base directory to launch the analysis:

sonar-scanner

8.12. Now you can view the project analysis report in the SonarQube Dashboard.

9. MOUNTING WINDOWS SHARED FOLDER TO UBUNTU

Steps to do in **Windows machine:**

9.1. Create a folder to be shared with Ubuntu in local directory.
9.2. Right click on the folder and select share with everyone.

Steps to do in **Ubuntu machine:**

9.3. Create a directory inside File System → tmp of your choice for ex : mountlcn.
9.4. Open terminal from desktop, type command
sudo apt-get install cifs-utils
sudo mount -t cifs //windows-IP-address/shared-folder -o username=M1036051 /linux/mountlcn -o vers=2.0
[Enter the password of respective user]
9.5. Place the files which needs to be shared from Ubuntu to Windows in the mountlcn directory
9.6. Shared apk file path must be mentioned in the appium selenium program with respect to windows.

Steps to **unmount** a directory**:**

9.7. Open terminal in Ubuntu machine, type command
sudo umount /folder-name

Permission for a directory to read, write and execute while mounting [Copy and remove]

sudo mount -t cifs //windows-IP-address/shared-folder -o username=M1036051,uid=administrator,gid=administrator,rw,dir_mode=0777 /linux/mountlcn -o vers=2.0

10. RUNNING TESTNG USING IDE ECLIPSE

Steps are shown below:

10.1. Create a Java testing project in eclipse.

10.2. Add **TestNG** library in Java Build Path of the project.
Project properties window → Java Build Path → Libraries → Add Library

10.3. Add External Jar files required for the project.
Project properties window → Java Build Path → Libraries → Add External Jars
Add the following Jar files:
Selenium-server-standalone-3.4.0.jar
AppiumForWindows-1.3.4.1.zip
Java-client-5.0.0-BETA9.jar

10.4. Create a lib folder in the current directory, add all these jar files in it. The primary purpose of adding these jar files in the lib folder is that, while executing from the command prompt you can tell the compiler that the required jar files for the execution of the program are present in this location. If you want to execute testng.xml from eclipse then this lib folder is not at all required.

10.5. Select the java file under package and right click. Select the option called **'TestNG'** and then click on **'Convert to TestNG'**. Proceed further and finish to generate testng.xml.

10.6. Launch the appium server and connect the device to the machine.

10.7. Right click on testing.xml and select **Run As** → **TestNG Suite** to execute the program.

10.8. Output of the program is shown in the console.

11. RUNNING TESTNG USING WINDOWS CMD

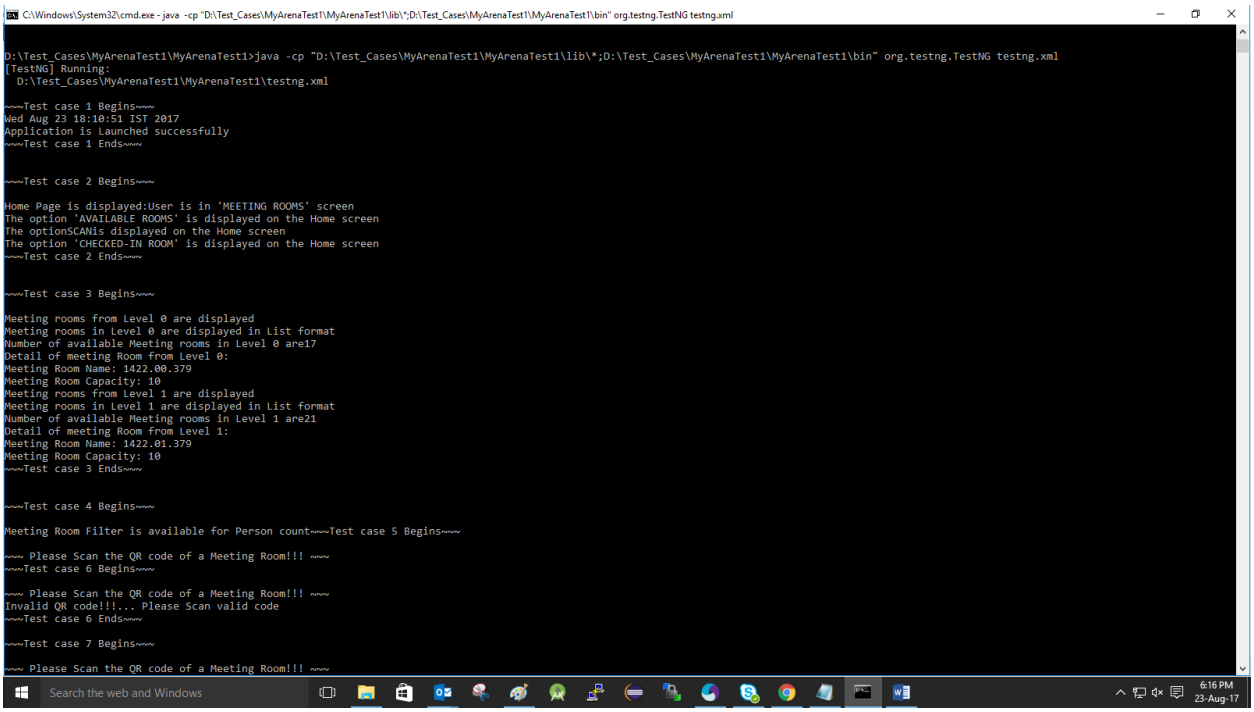
Steps are shown below:

11.1. Connect the device to the machine and launch the appium server.

11.2. Open Command Prompt in windows.

11.3. Traverse to the project directory.
For ex: **D:\Test_Cases\MyArenaTest1\MyArenaTest1>**

11.4. Run the command shown below respectively:
Java -cp "D:\Test_Cases\MyArenaTest1\MyArenaTest1\lib*; D:\Test_Cases\MyArenaTest1\MyArenaTest1\bin" org.testng.TestNG testng.xml



12. APPIUM TESTING FOR NATIVE ANDROID APPS

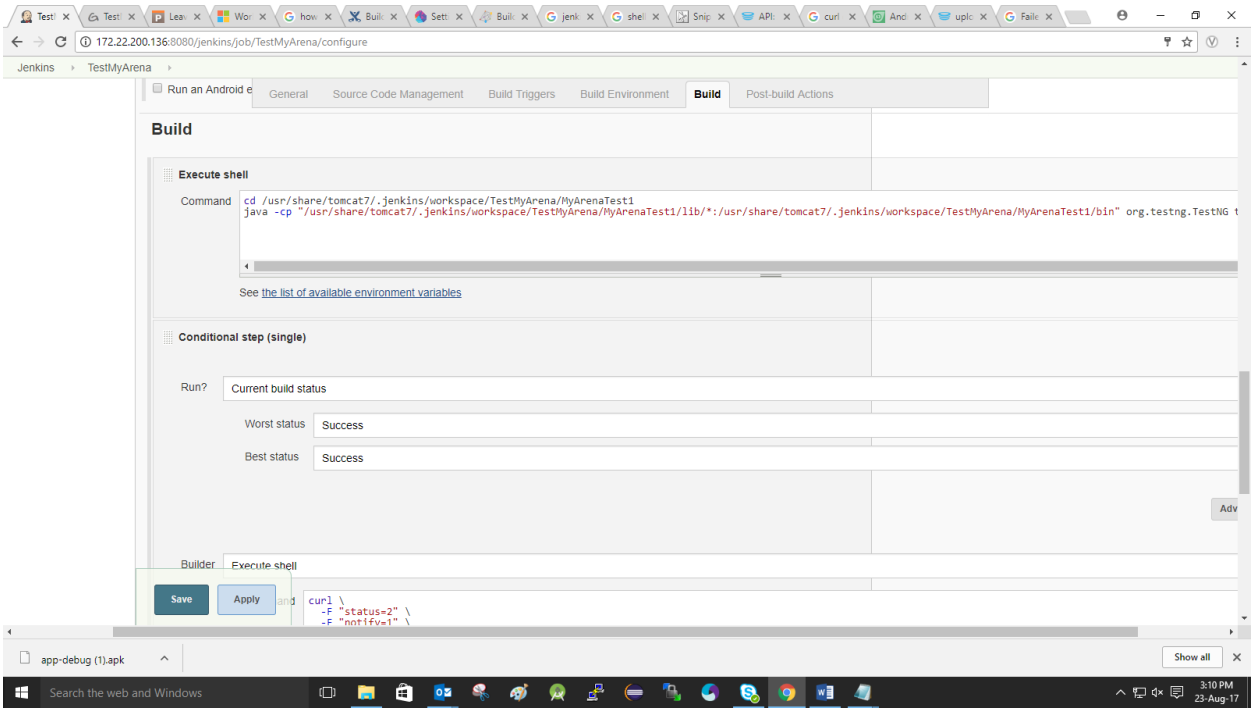
Steps to be done are shown below:

12.1. Create a job in Jenkins by following the same steps that of **GRADLE BUILD FOR NATIVE ANDROID APPS IN JENKINS USING SVN REPOSITORY** mentioned earlier until **Build Environment** section.

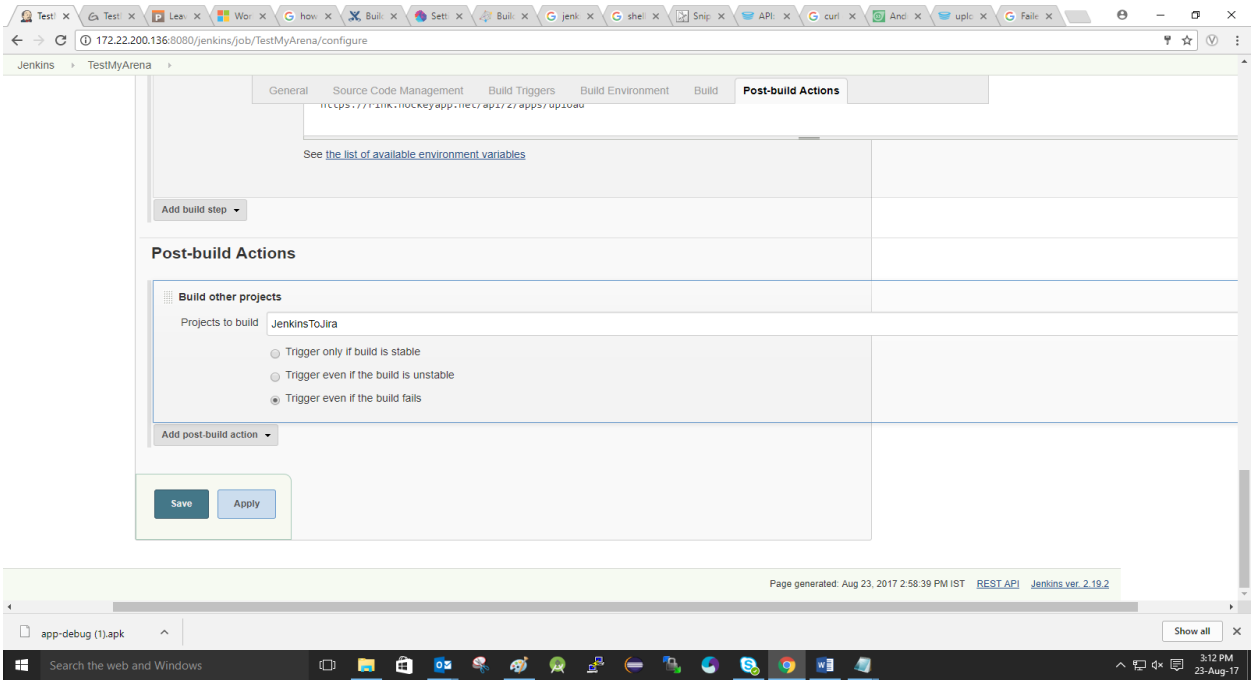
12.2. In the **Build** tab, select **Execute Shell**. Then provide the below commands in it

cd /usr/share/tomcat7/.jenkins/workspace/TestMyArena/MyArenaTest1

java -cp "/usr/share/tomcat7/.jenkins/workspace/TestMyArena /MyArenaTest1/lib/*:/usr/share/tomcat7/.jenkins/workspace /TestMyArena/MyArenaTest1/bin" org.testng.TestNG testng.xml

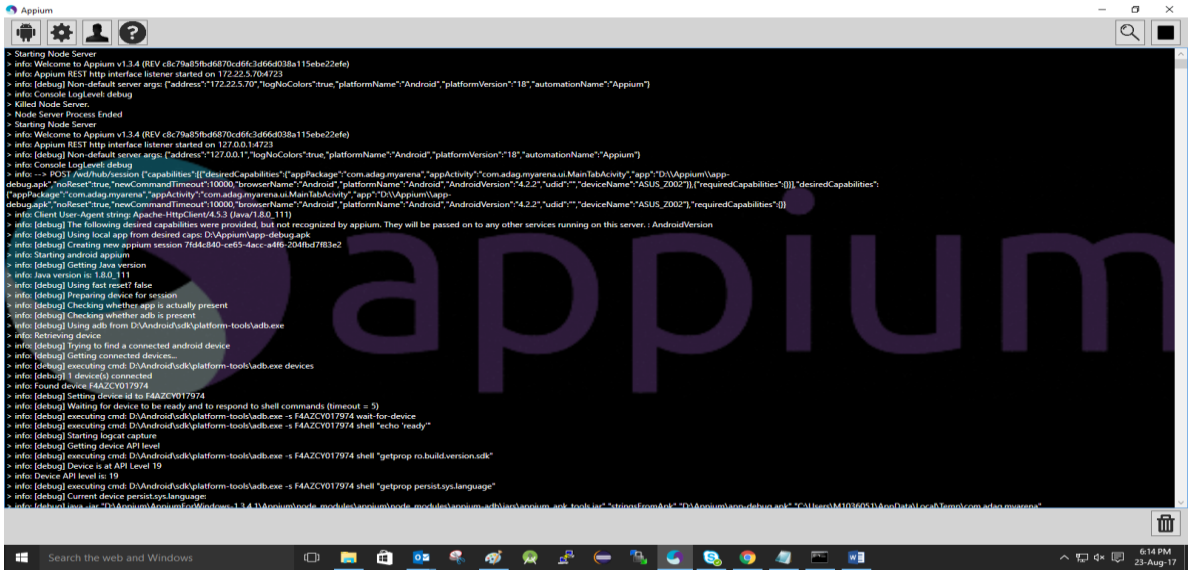


12.3. In the **Post-build** Actions tab, select **Build other projects** and provide the next job that has to be performed after testing the mobile application i.e logging jira bug Job.



12.4. Click on **Save**.

12.5. Launch the **Appium** server in your local machine and also connect your **device** to the same local machine.



12.6. **Build** the Job in Jenkins.

13. RUNNING JIRA PROGRAM USING ECLIPSE IDE

Steps are as shown below:

13.1. Create a Java Program which includes Java Rest Client api for posting a test issue to Jira Cloud account.

13.2. Add External Jar files required for the project.
Project properties window → Java Build Path → Libraries → Add External Jars

13.3. Create a lib folder in the current directory, add all these jar files in it. The primary purpose of adding these jar files in the lib folder is that, while executing from the command prompt you can tell the compiler that the required jar files for the execution of the program are present in this location. If you want to execute testng.xml from eclipse then this lib folder is not at all required.

13.4. Right click the Java file which contains the main method in it and click
Run As → Java Application

14. RUNNING JIRA PROGRAM USING WINDOWS CMD

Steps are shown below:

14.1. Open Command Prompt in windows.

14.2. Traverse to the project directory.
For ex: **D:\JenkinsJira_cmd\JenkinsJira>**

14.3. Run the command shown below respectively:

javac -d bin -sourcepath src -cp lib/* src/Appium.java

java -cp bin;lib/* Appium

15. LOGGING ISSUES TO JIRA CLOUD

Steps to be done are shown below:

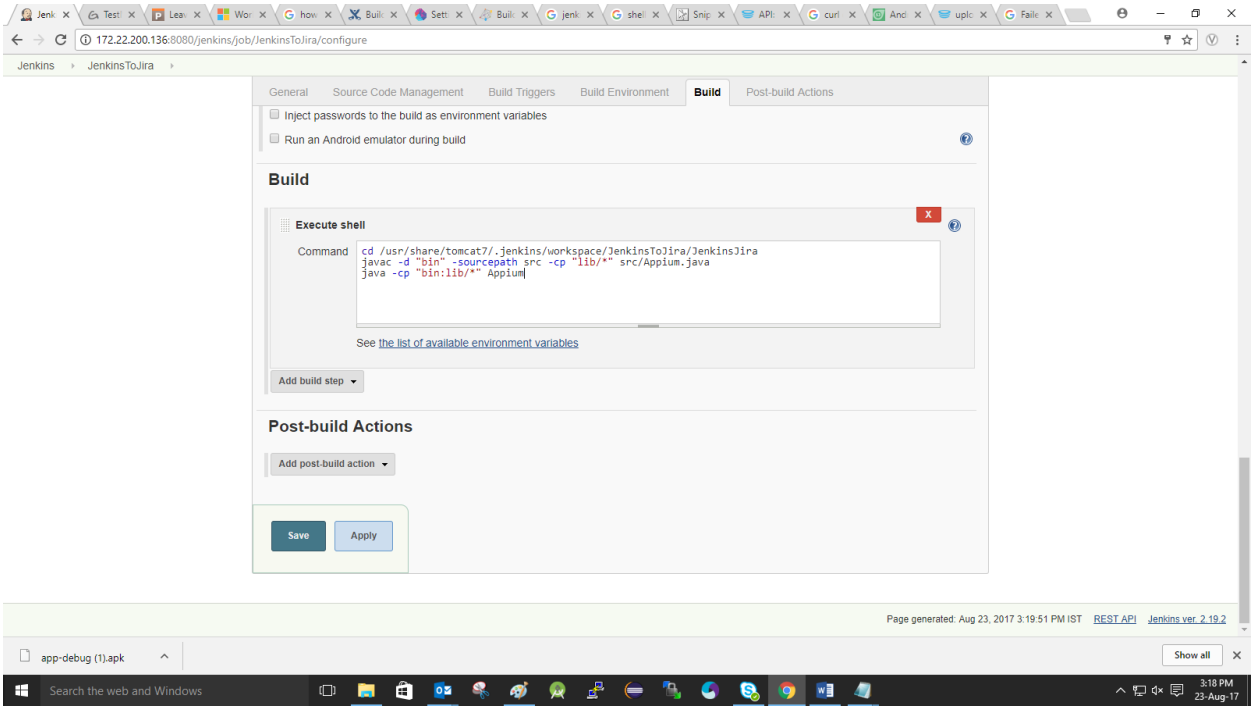
15.1. Create a job in Jenkins by following the same steps that of **Gradle build for native android apps in Jenkins using SVN Repository** mentioned earlier until **Build Environment** section.

15.2. In the **Build** tab, select **Execute Shell** and provide the below commands in it.

cd /usr/share/tomcat7/.jenkins/workspace/JenkinsToJira/JenkinsJira

javac -d "bin" -sourcepath src -cp "lib/*" src/Appium.java

java -cp "bin:lib/*" Appium



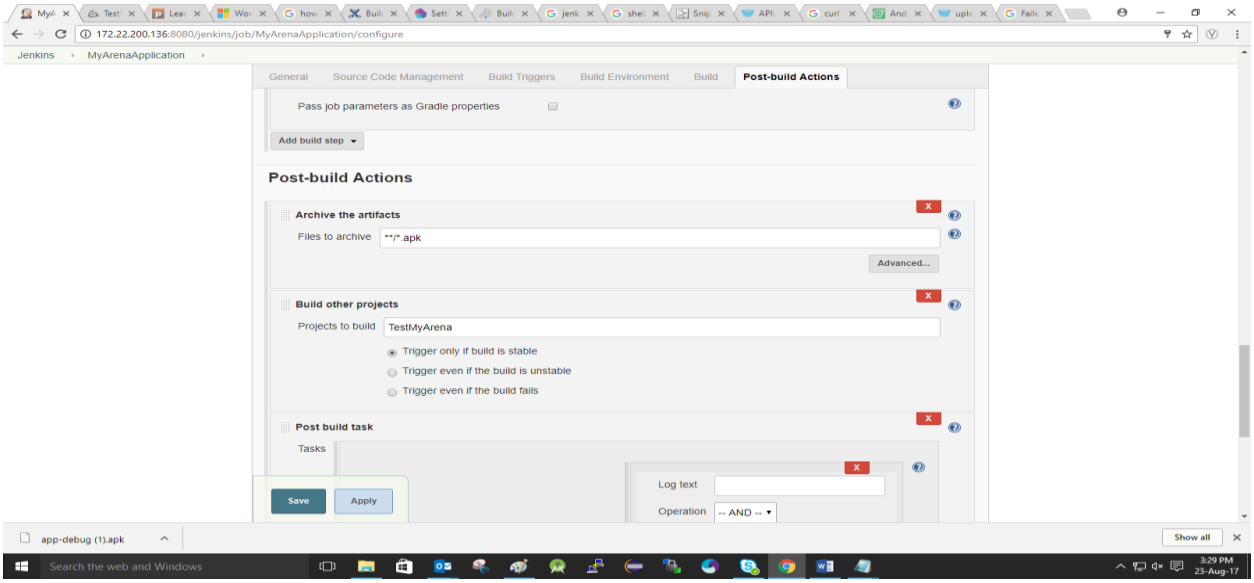
15.3. Click on **Save**.

15.4. **Build** the Job in Jenkins.

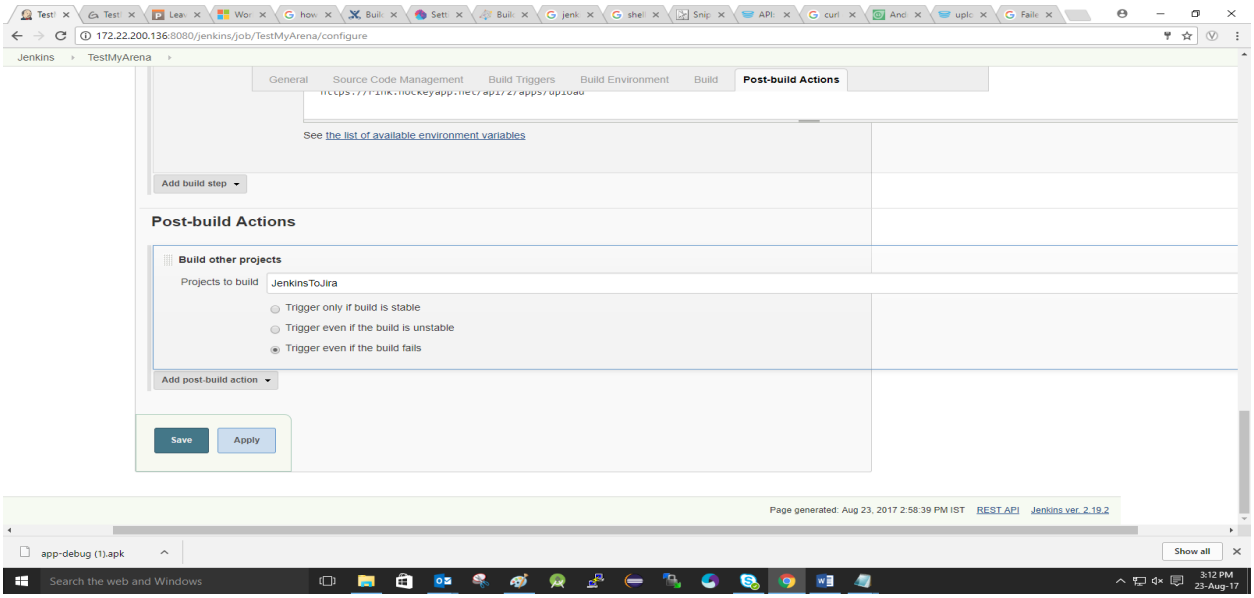
16. CREATING JOBS PIPELINE IN SEQUENCE IN JENKINS

Steps for creating pipeline of required jobs in the jenkins:

16.1. In the **Post-build** Actions tab of **apk file** generation Job, select **Build other projects** and provide the next job that has to be performed after generating the mobile application's apk file i.e **Automation testing** Job.



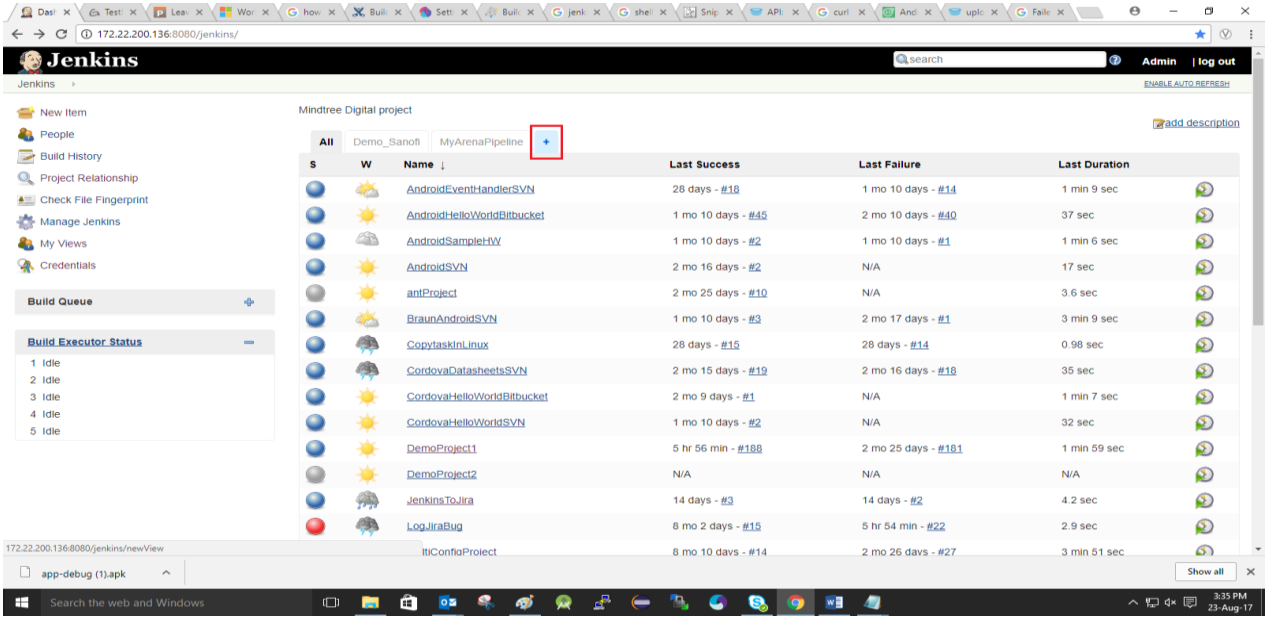
16.2. In the **Post-build** Actions tab of **Automation testing** Job, select **Build other projects** and provide the next job that has to be performed after testing the mobile application i.e **logging jira bug** Job.



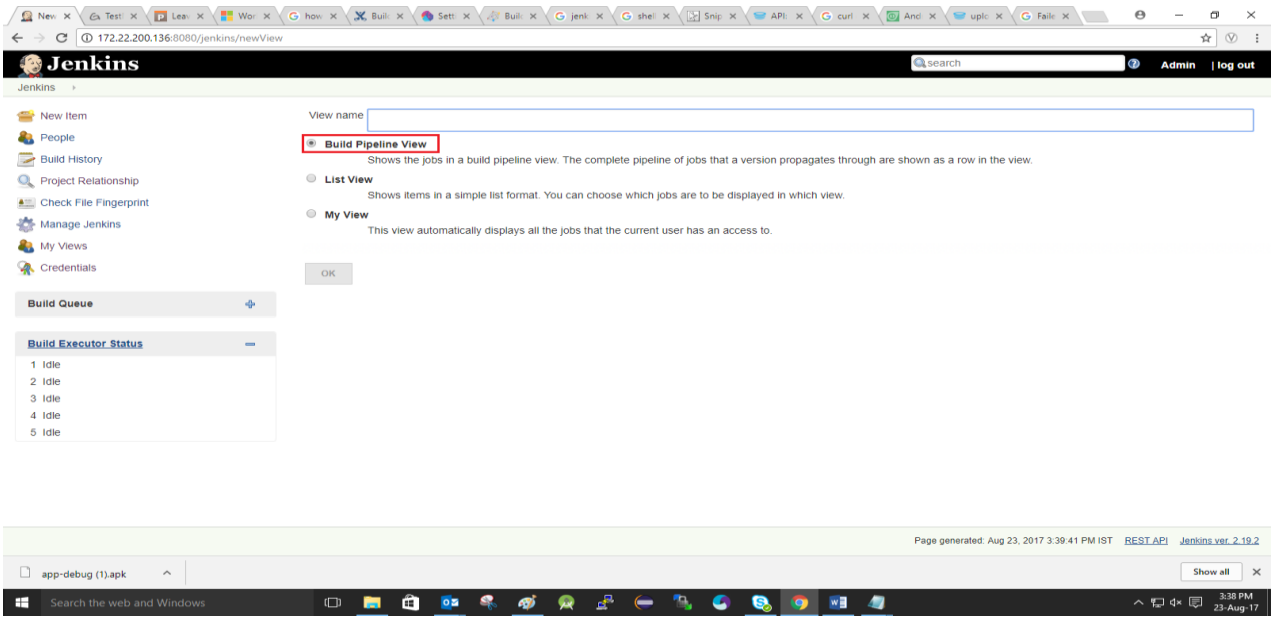
17. CREATING BUILD PIPELINE VIEW IN JENKINS

Steps to be followed for building pipeline view in the jenkins:

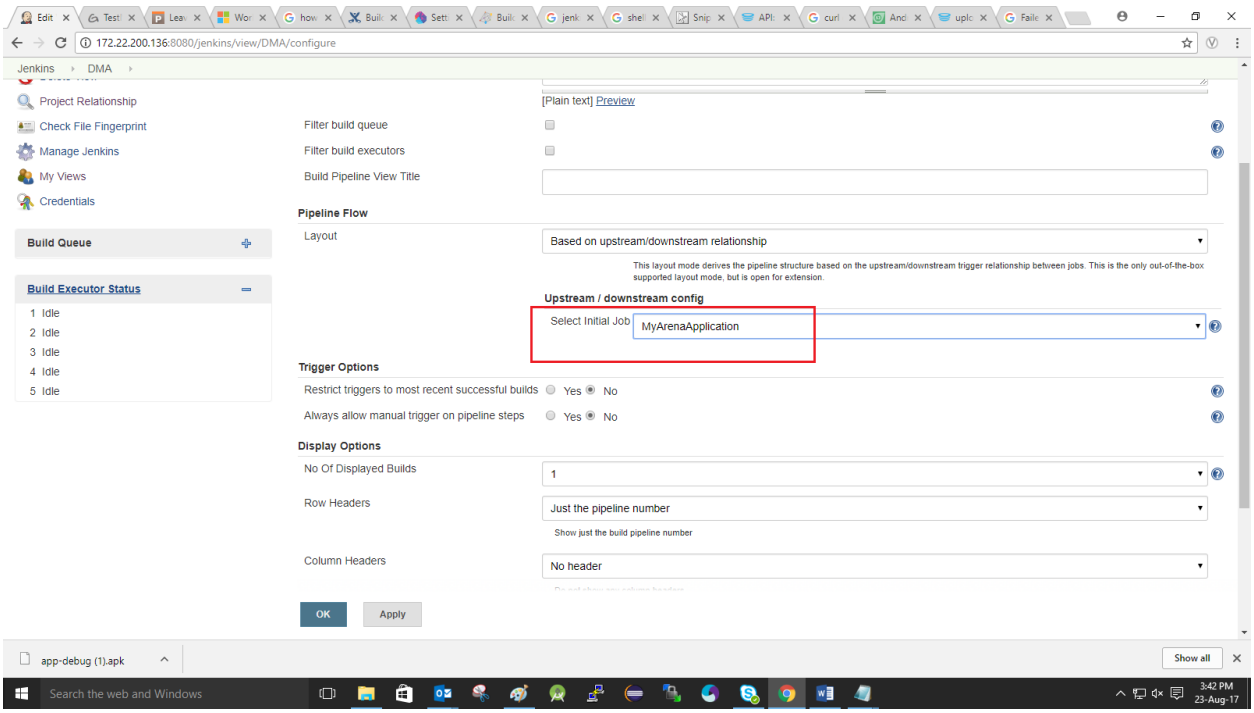
17.1. Click on the '+' icon in the jenkins Dashboard.



17.2. Name the Pipeline View and check the **Build Pipeline View** option.



17.3. Set the initial Job to be performed.



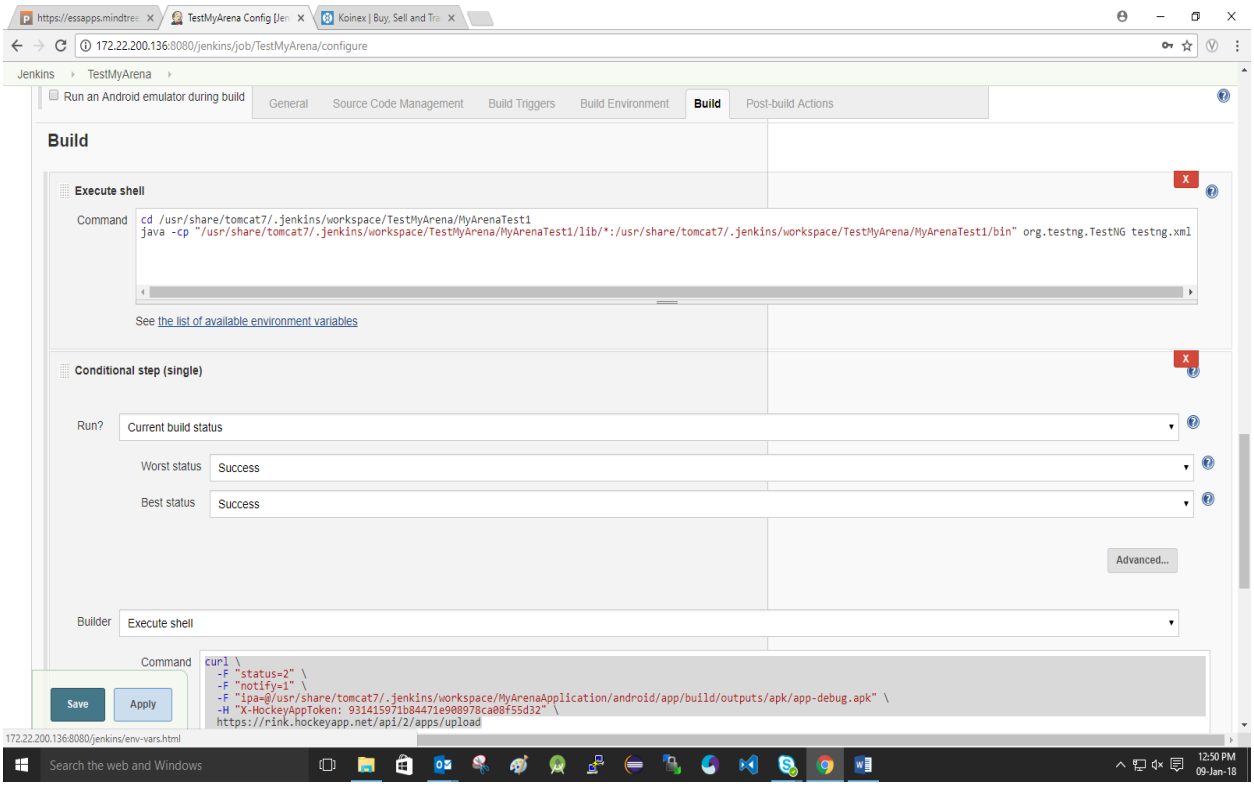
17.4. Click on **Save**.

17.5. **Build** the Job in Jenkins.

18. UPLOADING AN APK TO HOCKEYAPP

18.1. In case if the testing is stable, then execute the below shell commands in build tab in **Jenkins** to upload apk to Hockey App

```
curl \
 -F "status=2" \
 -F "notify=1" \
 -F "ipa=@/usr/share/tomcat7/.jenkins/workspace/MyArenaApplication/
     android/app/build/outputs/apk/app-debug.apk" \
 -H "X-HockeyAppToken: 931415971b84471e908978ca08f55d32" \
 https://rink.hockeyapp.net/api/2/apps/upload
```



19. ISSUES RELATED TO BUILD IN LINUX

19.1. To start Jenkins server running in the port 8000

sudo /etc/init.d/jenkins start

19.2. Android license agreement related issues

cd /usr/share/tomcat7/.jenkins/tools/

sudo chmod –R uga+rwx android-sdk

cd /usr/share/tomcat7/.jenkins/tools/android-sdk/tools/bin

android update sdk –no-ui

19.3. Issue with Min Max version of JDK

sudo vi /etc/environment

add below line:

/usr/lib/jvm/java-8-openjdk-amd64/

Ctrl+x, y

source /etc/environment

echo $JAVA_HOME

(OR)

sudo update-alternatives --config java

sudo vi /etc/environment

add below line:

/usr/lib/jvm/java-8-openjdk-amd64/

: x is used to save the changes in linux

source /etc/environment && export PATH

set the JDK 8 path in 'Manage Jenkins' -> 'Global Tool Configuration' -> JDK

19.4. To restart tomcat server

cd /var/lib/share/tomcat7

sudo service tomcat7 start