

CICD [JENKINS]

AUTHOR : HARISH R

Welcome to possible

Contents

1. PREREQUISITIES OF GRADLE BUILDING IN JENKINS	3
2. GRADLE BUILD FOR NATIVE ANDROID APPS IN JENKINS USING SVN REPOSITORY	4-9
3. PUSHING SOURCE CODE FROM LOCAL DIRECTORY TO BITBUCKET REPOSITORY	10-12
4. GRADLE BUILD FOR ANDROID APPS IN JENKINS USING BITBUCKET REPOSITORY	13-14
5. ANDROID ANT BUILD FOR CORDOVA ANDROID APPS USING SVN REPOSITORY	15
6. ESTABLISHING CONNECTION TO REMOTE SERVER.....	16
7. INFORMATION RELATED TO PATHS AND ISSUES IN UBUNTU SERVER	17
8. SONARQUBE STATIC CODE ANALYSIS IN WINDOWS.....	18-19
9. MOUNTING WINDOWS SHARED FOLDER TO UBUNTU.....	20
10. RUNNING TESTNG USING IDE ECLIPSE.....	21
11. RUNNING TESTNG USING WINDOWS CMD.....	22
12. APPIUM TESTING FOR NATIVE ANDROID APPS.....	23-24
13. RUNNING JIRA PROGRAM USING ECLIPSE IDE	25
14. RUNNING JIRA PROGRAM USING WINDOWS CMD	26
15. LOGGING ISSUES TO JIRA CLOUD.....	26
16. CREATING JOBS PIPELINE IN SEQUENCE IN JENKINS	27
17. CREATING BUILD PIPELINE VIEW IN JENKINS	28-29
18. UPLOADING APK TO HOCKEY APP	30
19. ISSUES RELATED TO BUILD IN LINUX	31

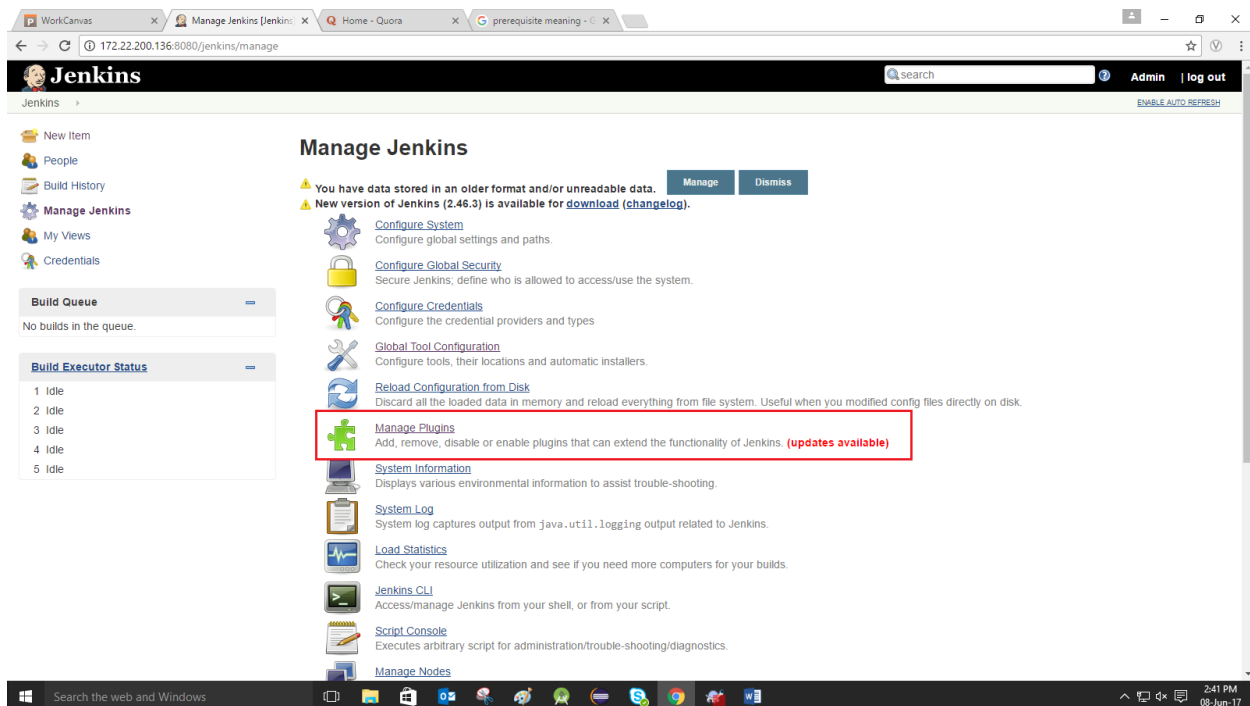
1. PREREQUISITES FOR GRADLE BUILDING IN JENKINS

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



The screenshot shows the Jenkins web interface. The top navigation bar includes the Jenkins logo, a search bar, and links for 'Admin' and 'log out'. The left sidebar contains a 'Manage Jenkins' section with links for 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', and 'Credentials'. The main content area is titled 'Manage Jenkins' and displays a list of configuration options. A red box highlights the 'Manage Plugins' option, which includes the text 'Add, remove, disable or enable plugins that can extend the functionality of Jenkins. (updates available)'. Other options include 'Configure System', 'Configure Global Security', 'Configure Credentials', 'Global Tool Configuration', 'Reload Configuration from Disk', 'System Information', 'System Log', 'Load Statistics', 'Jenkins CLI', 'Script Console', and 'Manage Nodes'. The bottom of the page shows a Windows taskbar with the date and time as 2:41 PM on 08-Jun-17.

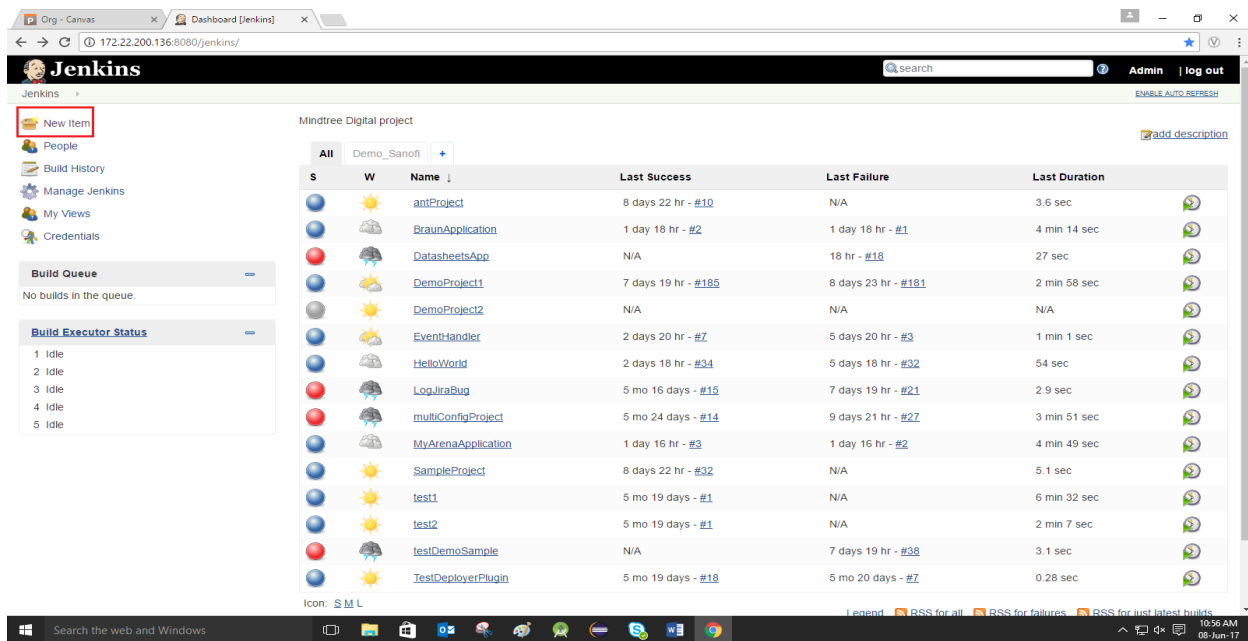
2. GRADLE BUILD FOR NATIVE ANDROID APPS IN JENKINS USING SVN REPOSITORY

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/>

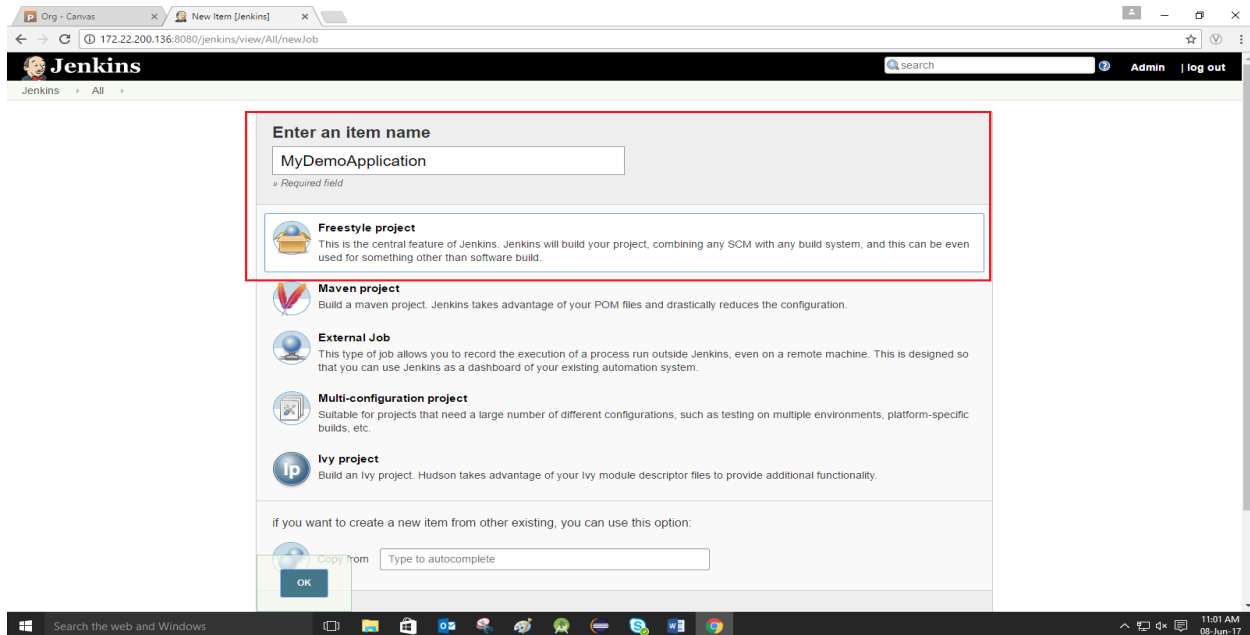
2.2. Click on **New Item**.



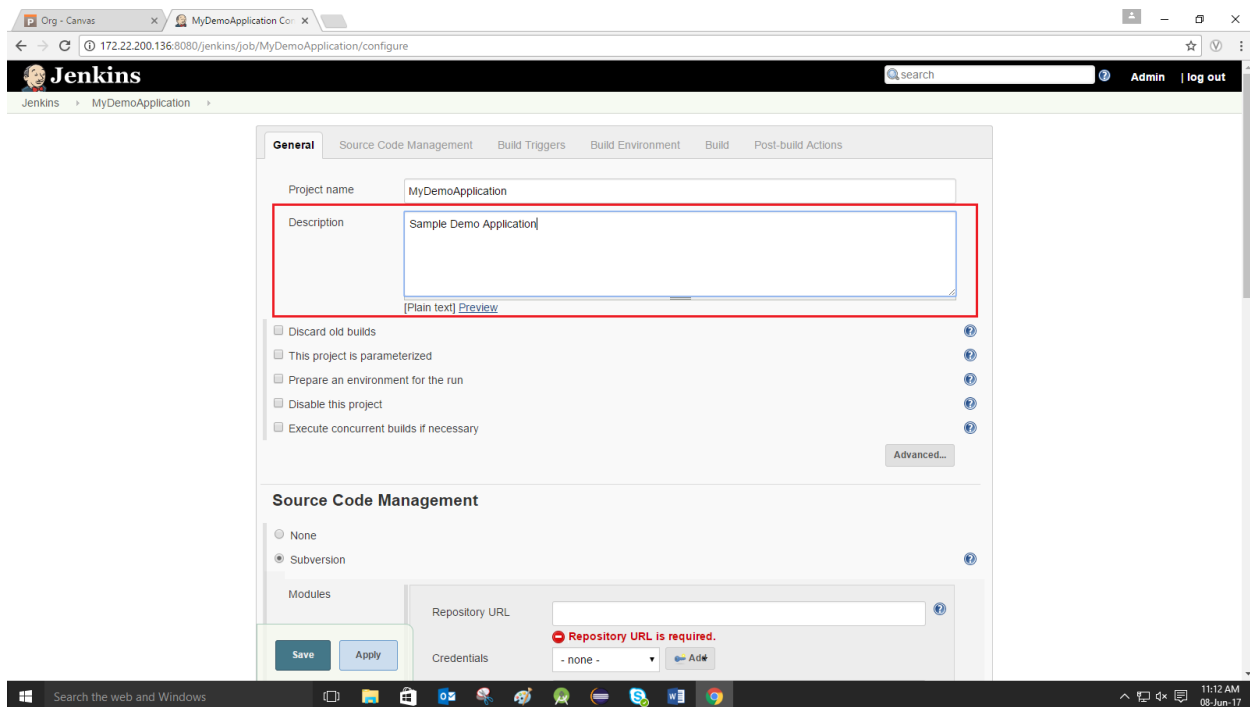
The screenshot shows the Jenkins web interface. In the left sidebar, the 'New Item' button is highlighted with a red box. The main area displays the 'Mindtree Digital project' dashboard for the 'Demo_Sanofi' job. It features a table of build history with columns for status, icon, name, last success, last failure, and last duration.

S	W	Name	Last Success	Last Failure	Last Duration
Success	Sun	antProject	8 days 22 hr - #10	N/A	3.6 sec
Success	Sun	BraunApplication	1 day 18 hr - #2	1 day 18 hr - #1	4 min 14 sec
Success	Sun	DatasheetsApp	N/A	18 hr - #18	27 sec
Success	Sun	DemoProject1	7 days 19 hr - #185	8 days 23 hr - #181	2 min 58 sec
Success	Sun	DemoProject2	N/A	N/A	N/A
Success	Sun	EventHandler	2 days 20 hr - #7	5 days 20 hr - #3	1 min 1 sec
Success	Sun	HelloWorld	2 days 18 hr - #34	5 days 18 hr - #32	54 sec
Success	Sun	LogIraBug	5 mo 16 days - #15	7 days 19 hr - #21	2.9 sec
Success	Sun	multiConfigProject	5 mo 24 days - #14	9 days 21 hr - #27	3 min 51 sec
Success	Sun	MyArenaApplication	1 day 16 hr - #3	1 day 16 hr - #2	4 min 49 sec
Success	Sun	SampleProject	8 days 22 hr - #32	N/A	5.1 sec
Success	Sun	test1	5 mo 19 days - #1	N/A	6 min 32 sec
Success	Sun	test2	5 mo 19 days - #1	N/A	2 min 7 sec
Success	Sun	testDemoSample	N/A	7 days 19 hr - #38	3.1 sec
Success	Sun	TestDeployerPlugin	5 mo 19 days - #18	5 mo 20 days - #7	0.28 sec

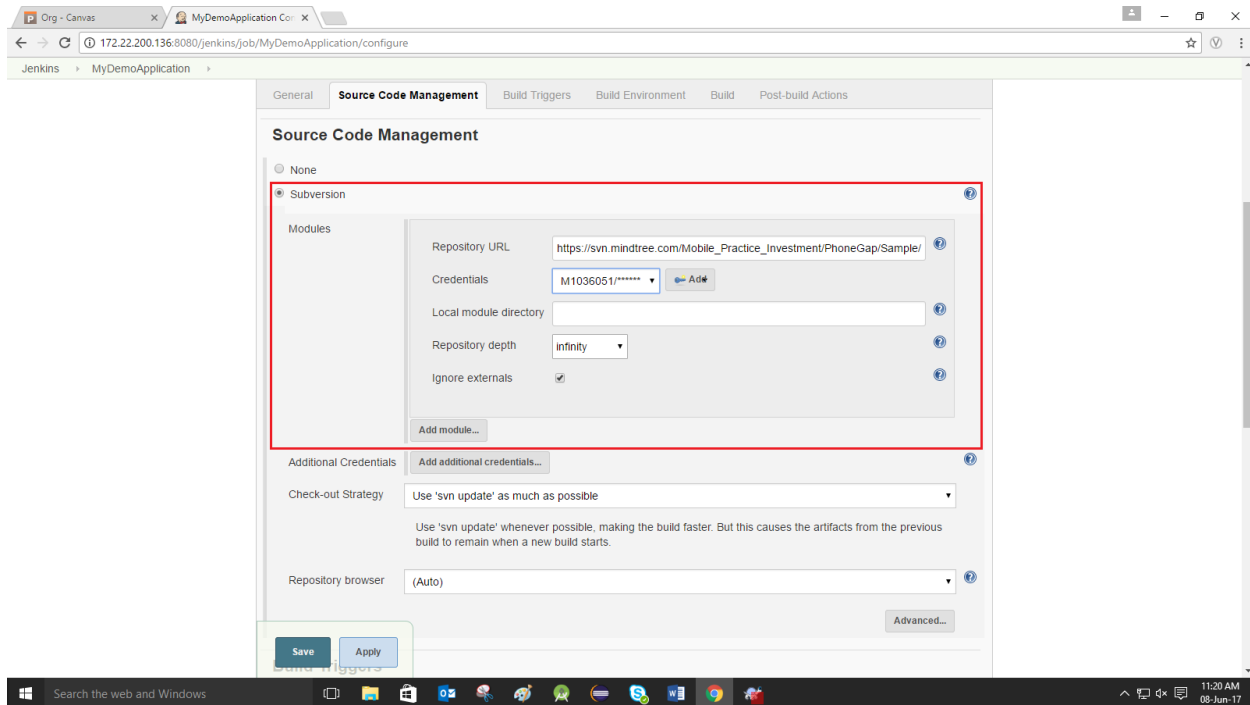
2.3. Enter an **item name** and select **Free style project**, click OK.



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



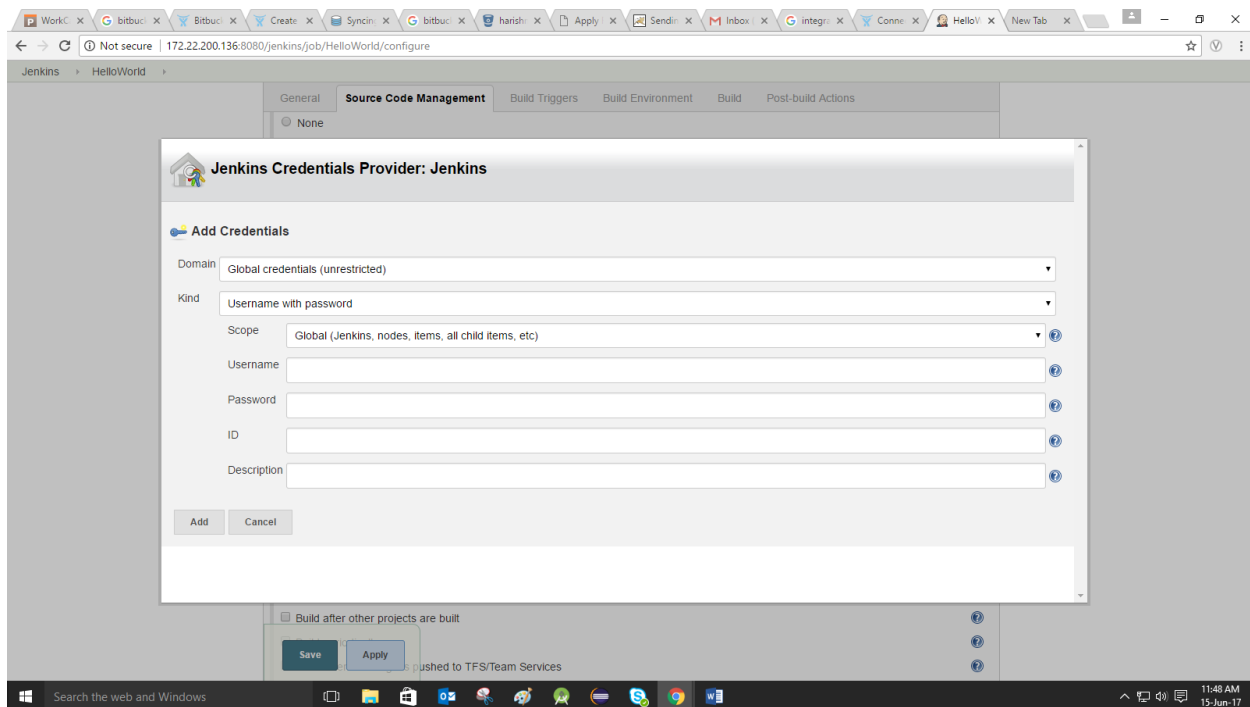
- 2.5. Select **subversion** in Source Code Management tab. Enter the path of the source code present in the **SVN repository**, add your respective **credentials** required for accessing the repo and make **Local module directory** empty.



The screenshot shows the Jenkins configuration page for a job named 'MyDemoApplication'. The 'Source Code Management' tab is selected. Under the 'Subversion' section, the following fields are visible:

- Repository URL:** `https://svn.mindtree.com/Mobile_Practice_Investment/PhoneGap/Sample/`
- Credentials:** A dropdown menu showing 'M1036051/*****' with an 'Add' button next to it.
- Local module directory:** An empty text field.
- Repository depth:** A dropdown menu set to 'Infinity'.
- Ignore externals:** A checked checkbox.
- Additional Credentials:** A section with an 'Add additional credentials...' button.
- Check-out Strategy:** A dropdown menu set to 'Use 'svn update' as much as possible'.
- Repository browser:** A dropdown menu set to '(Auto)'.

Buttons for 'Save' and 'Apply' are at the bottom left, and an 'Advanced...' button is at the bottom right.

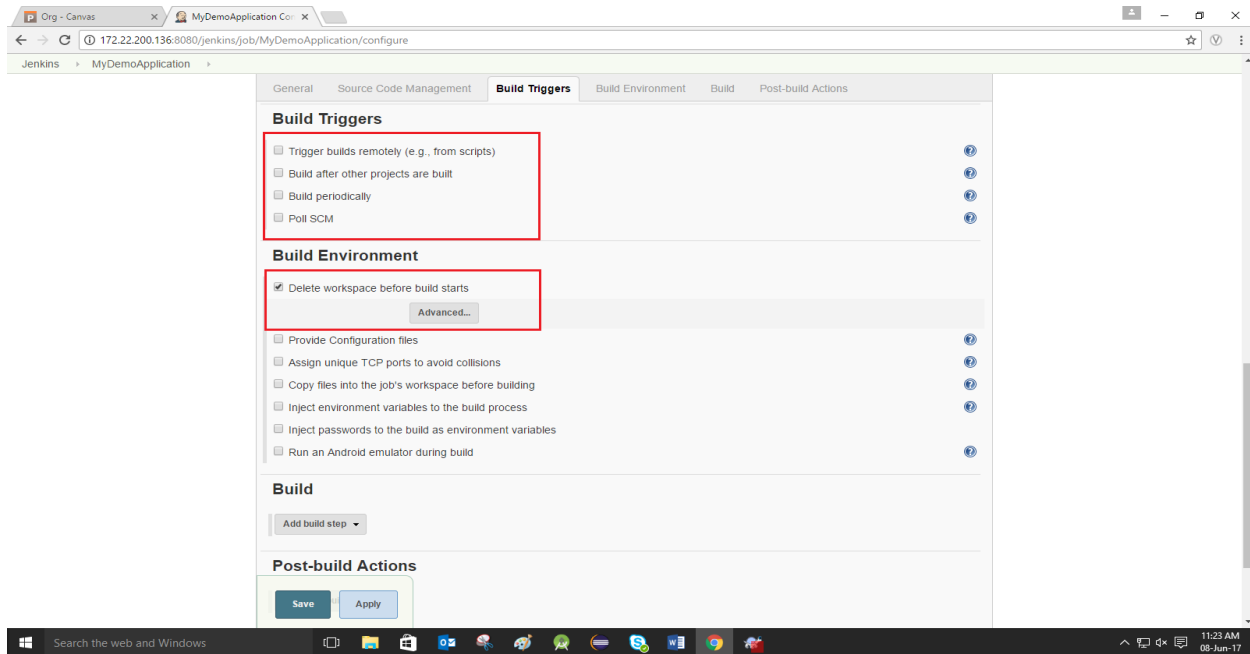


The screenshot shows a 'Jenkins Credentials Provider: Jenkins' dialog box with the 'Add Credentials' tab selected. The following fields are visible:

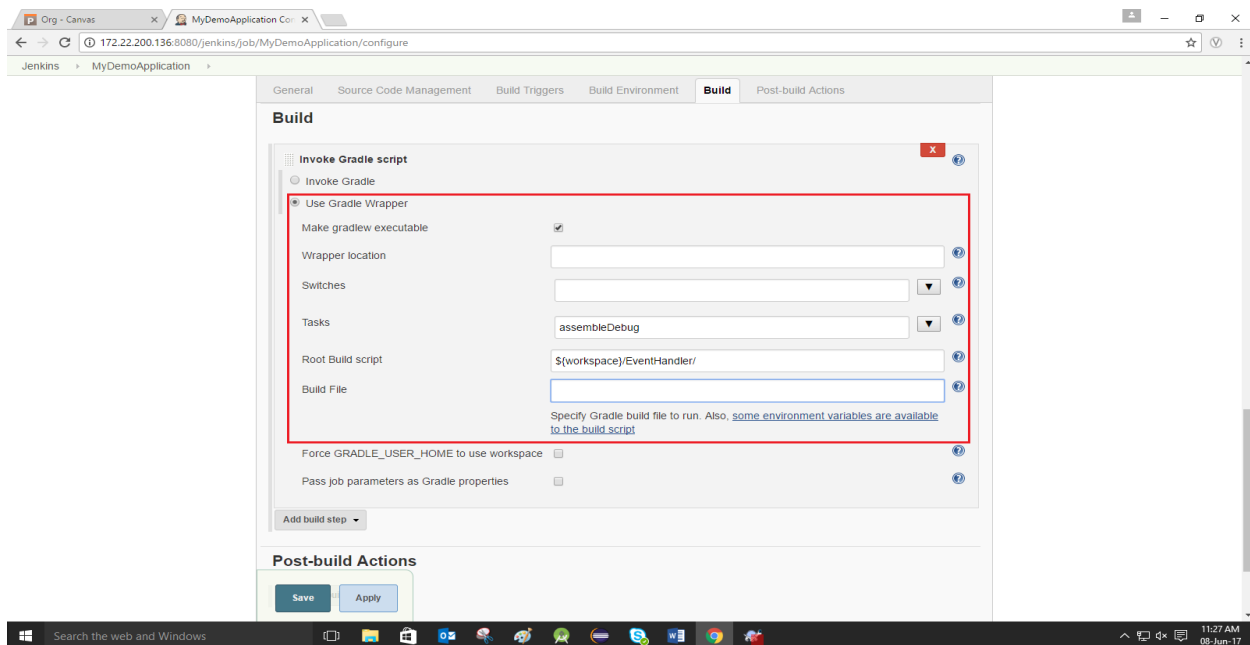
- Domain:** A dropdown menu set to 'Global credentials (unrestricted)'.
- Kind:** A dropdown menu set to 'Username with password'.
- Scope:** A dropdown menu set to 'Global (Jenkins, nodes, items, all child items, etc)'.
- Username:** An empty text field.
- Password:** An empty text field.
- ID:** An empty text field.
- Description:** An empty text field.

Buttons for 'Add' and 'Cancel' are at the bottom left. The background shows the Jenkins configuration page for a job named 'HelloWorld'.

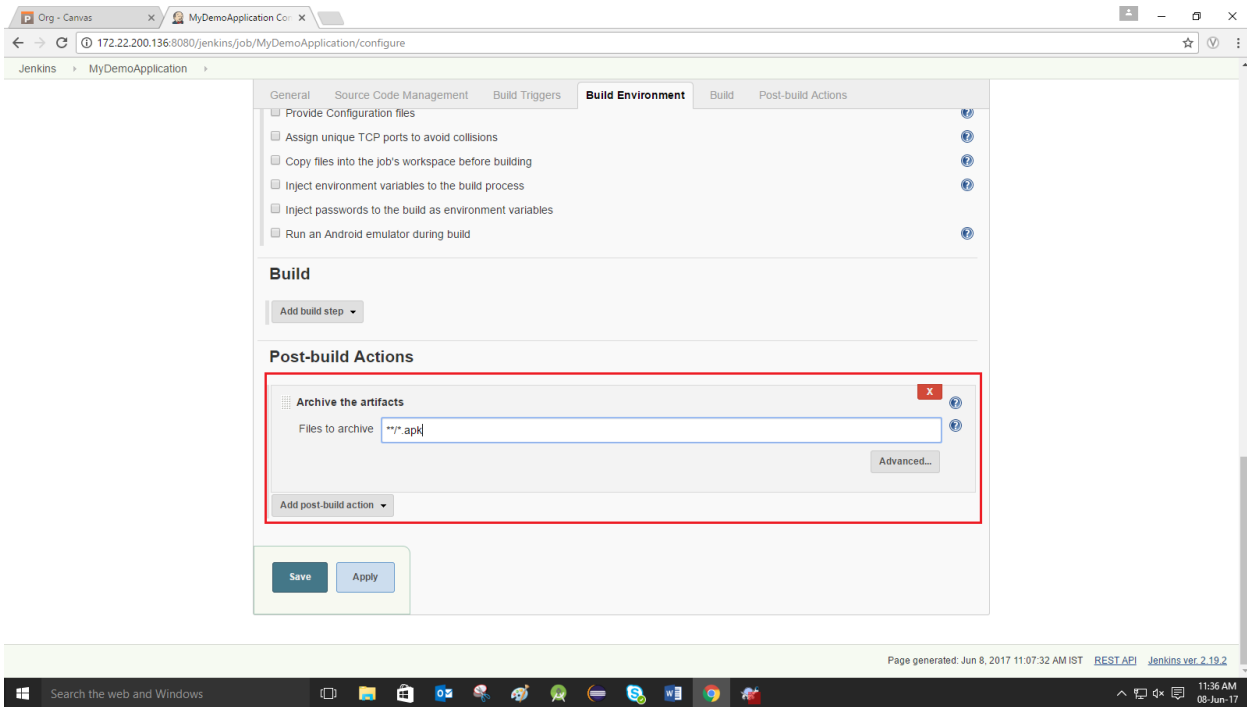
2.6. Don't make any changes in Build Triggers tab. Select **Delete workspace before build starts** in Build Environment tab.



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>/**.

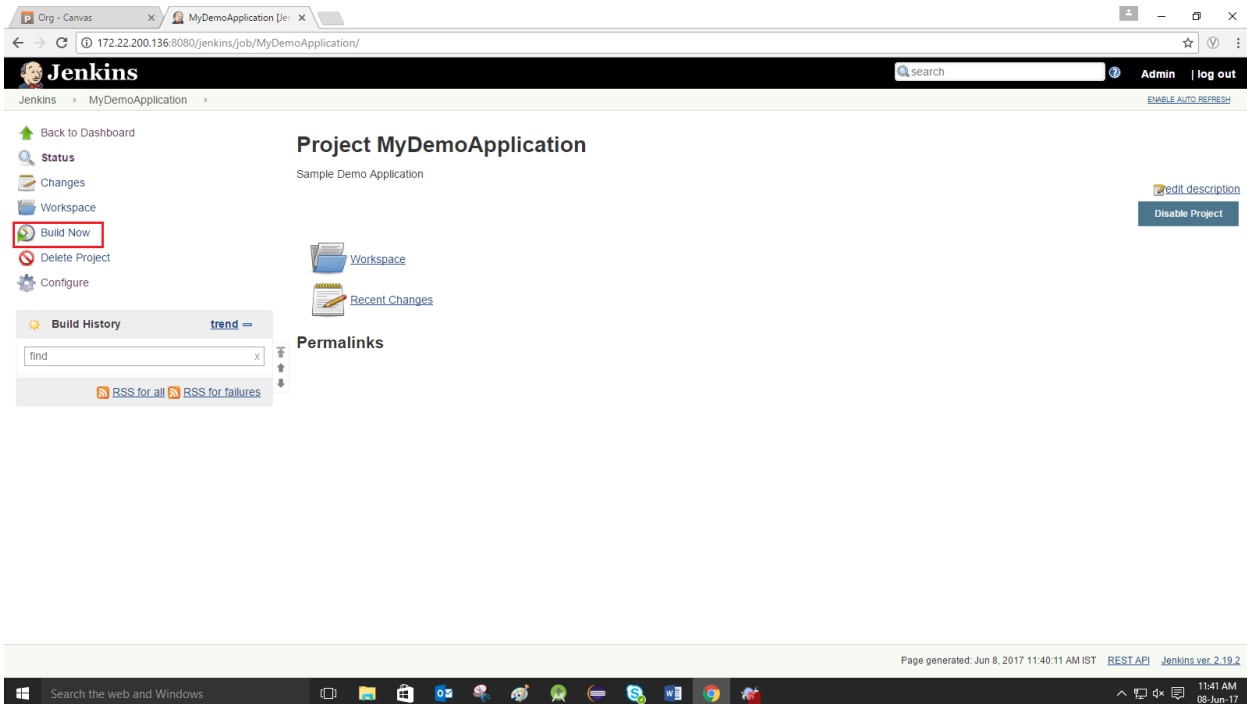


2.8. Select **Archive the Artifacts** under **add post build action**. Enter ****/*.apk** in the **files to archive** field for generating .apk file after build.



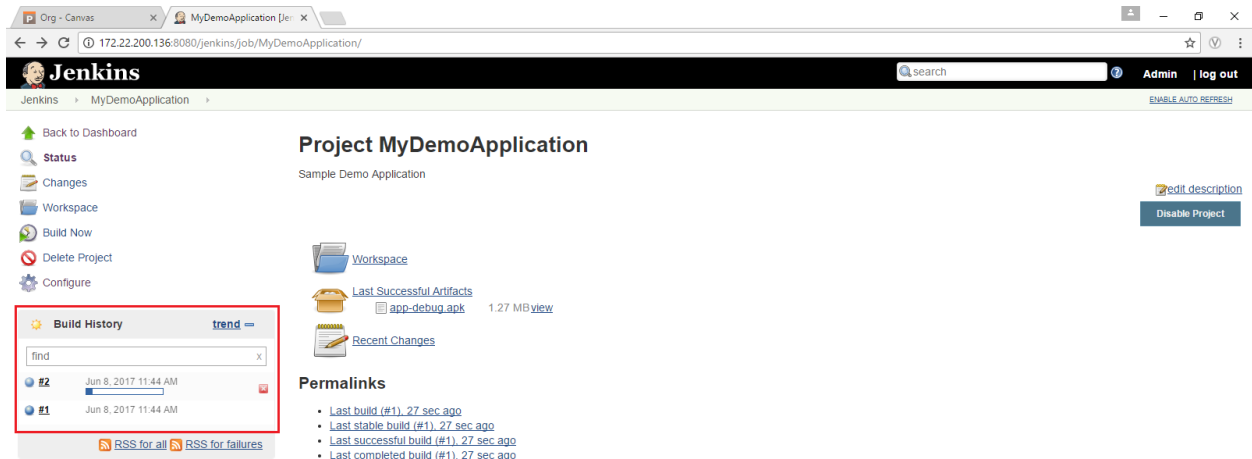
The screenshot shows the Jenkins configuration page for a job named 'MyDemoApplication'. The 'Post-build Actions' tab is active, and the 'Archive the artifacts' action is selected. The 'Files to archive' field is populated with '**/*.apk'. The 'Save' button is highlighted in blue. The page footer indicates it was generated on Jun 8, 2017, at 11:07:32 AM IST, using Jenkins version 2.19.2.

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



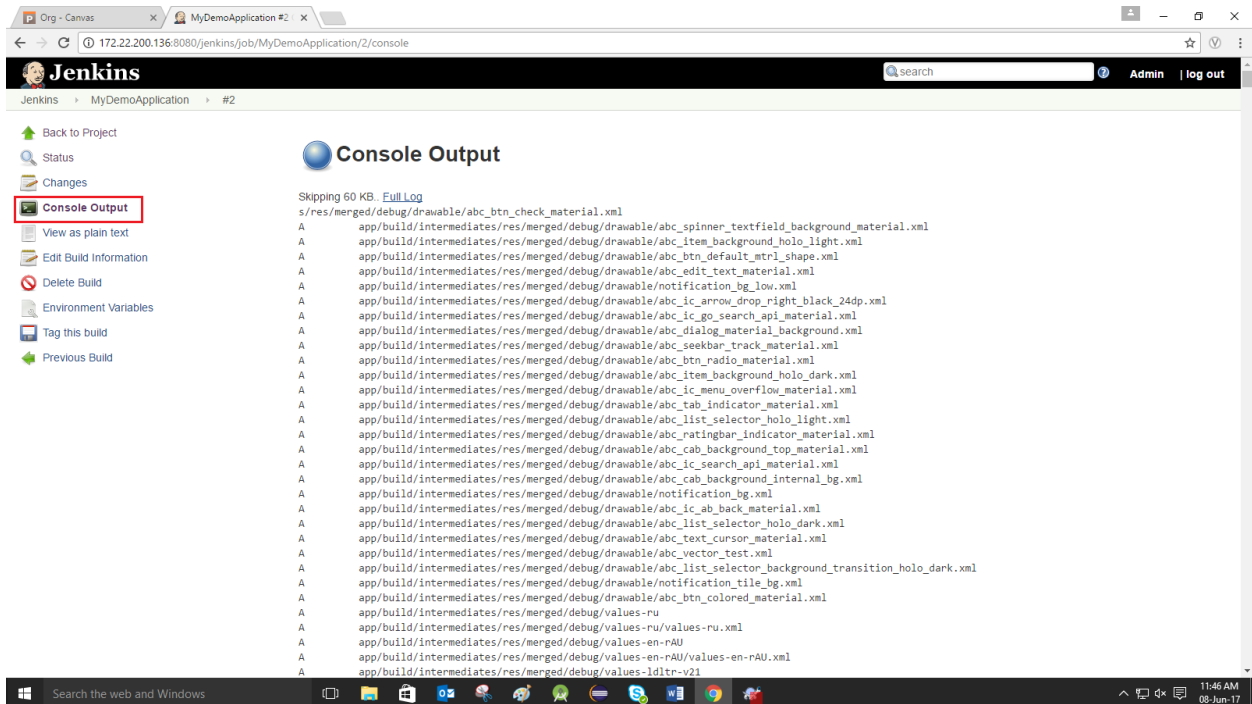
The screenshot shows the Jenkins main page for 'Project MyDemoApplication'. The 'Build Now' button is highlighted in the left sidebar. The 'Workspace' and 'Recent Changes' links are visible in the main content area. The page footer indicates it was generated on Jun 8, 2017, at 11:40:11 AM IST, using Jenkins version 2.19.2.

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.



The screenshot shows the Jenkins interface for 'Project MyDemoApplication'. The 'Build History' tab is active, displaying a list of builds. Build #2 is the most recent, dated Jun 8, 2017 11:44 AM. The left sidebar shows the 'Console Output' option highlighted. The top navigation bar includes links for 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', and 'Configure'.

Page generated: Jun 8, 2017 11:44:42 AM IST [REST API](#) [Jenkins ver. 2.19.2](#)



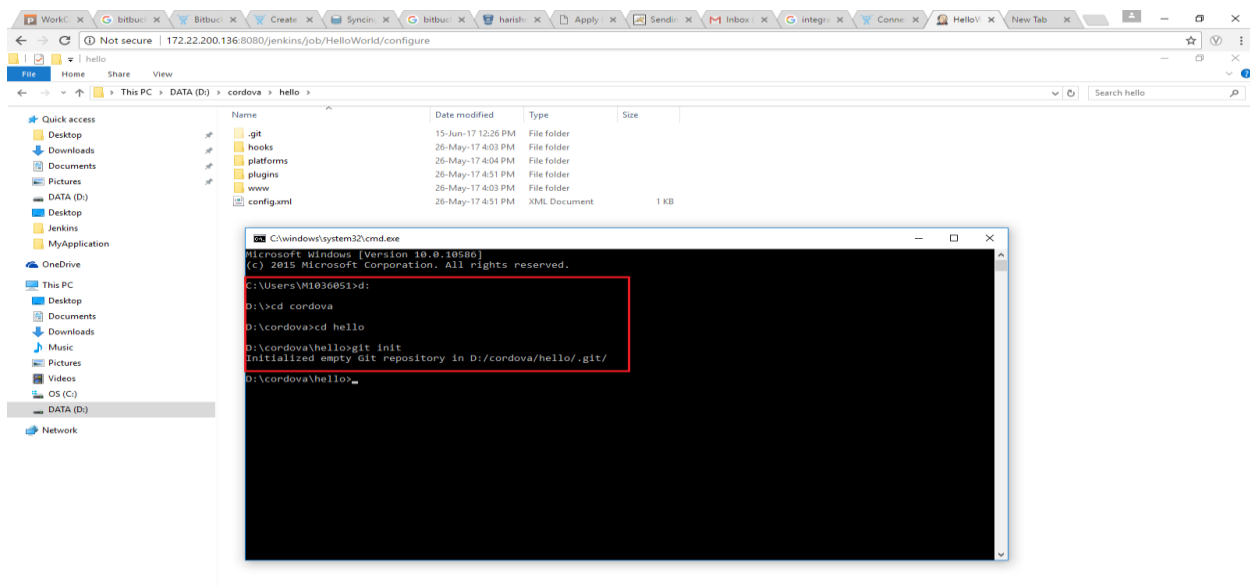
The screenshot shows the Jenkins 'Console Output' page for build #2. The 'Console Output' option is highlighted in the left sidebar. The main content area displays the output of the build, which is a list of files and directories. The output starts with 'Skipping 60 KB. Full Log' and then lists various XML files, including 'abc_btn_check_material.xml', 'abc_spinner_textfield_background_material.xml', 'abc_item_background_holo_light.xml', 'abc_btn_default_mtrl_shape.xml', 'abc_edit_text_material.xml', 'notification_bg_low.xml', 'abc_ic_arrow_drop_right_black_24dp.xml', 'abc_ic_go_search_api_material.xml', 'abc_dialog_material_background.xml', 'abc_seekbar_track_material.xml', 'abc_btn_radio_material.xml', 'abc_item_background_holo_dark.xml', 'abc_ic_menu_overflow_material.xml', 'abc_tab_indicator_material.xml', 'abc_list_selector_holo_light.xml', 'abc_ratingbar_indicator_material.xml', 'abc_cab_background_top_material.xml', 'abc_ic_search_api_material.xml', 'abc_cab_background_internal_bg.xml', 'notification_bg.xml', 'abc_ic_ab_back_material.xml', 'abc_list_selector_holo_dark.xml', 'abc_text_cursor_material.xml', 'abc_vector_test.xml', 'abc_list_selector_background_transition_holo_dark.xml', 'notification_tile_bg.xml', 'abc_btn_colored_material.xml', 'values-ru', 'values-ru-rU', 'values-en-rAU', and 'values-ldltr-v21'.

3. PUSHING SOURCE CODE FROM LOCAL DIRECTORY TO BITBUCKET REPOSITORY

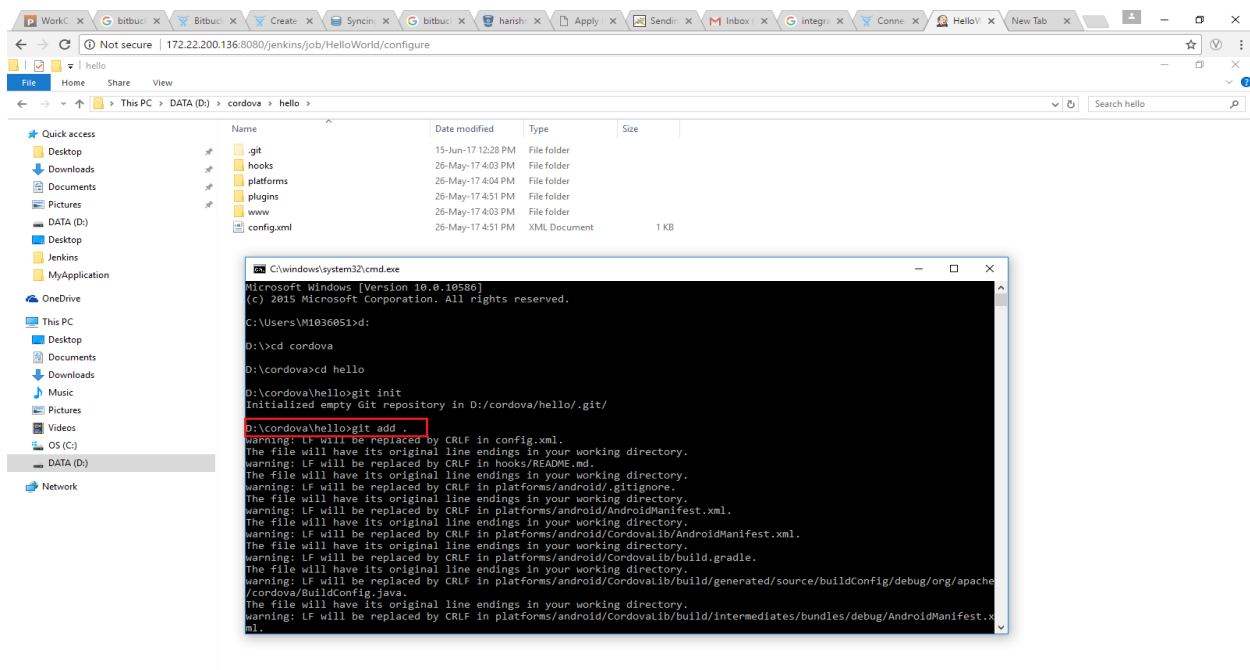
Follow the below steps:

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

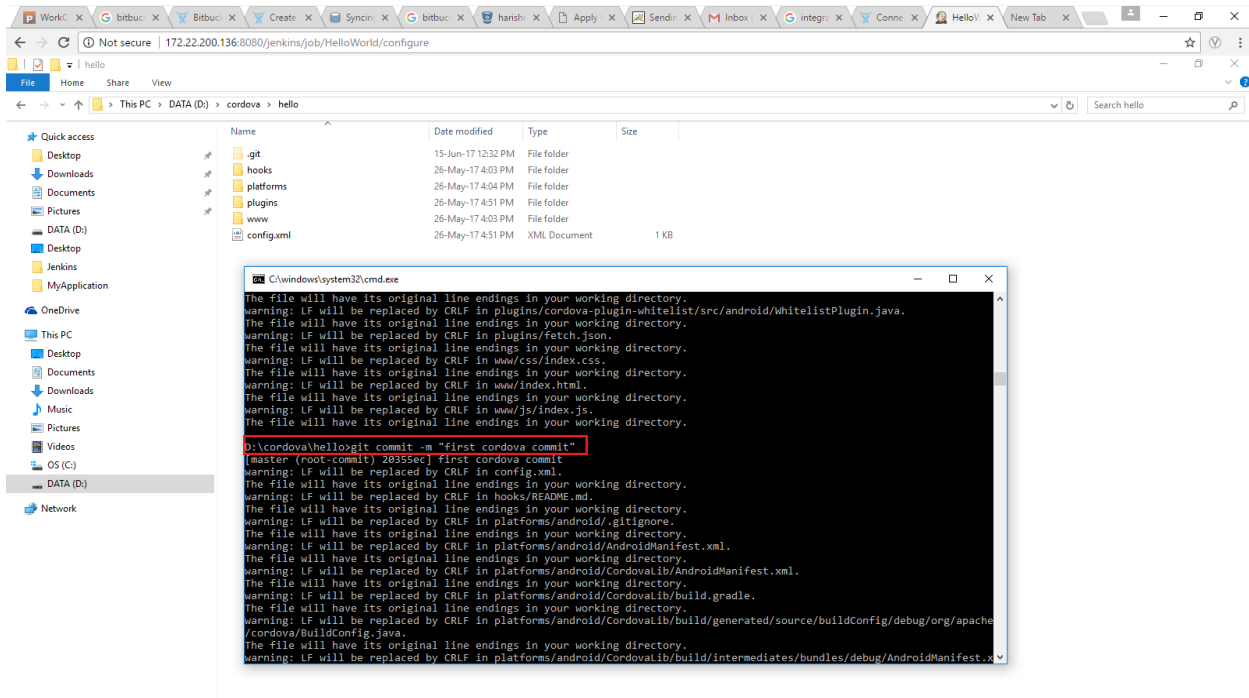
```
cd <project directory>
git init
```



```
git add .
```



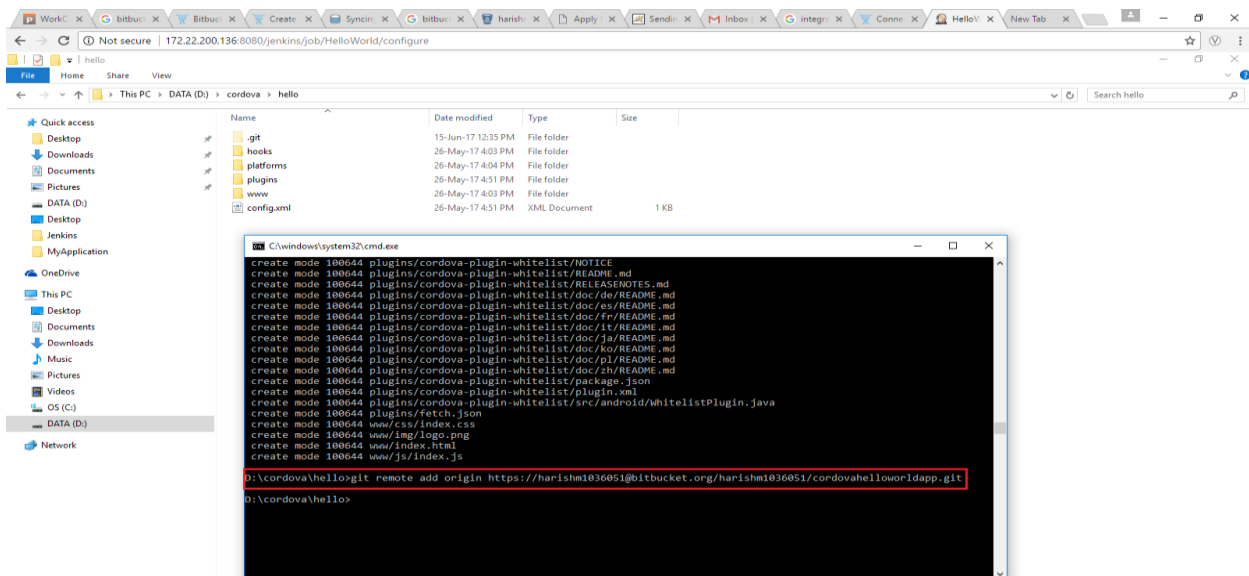
```
git commit -m "Initial commit"
```



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

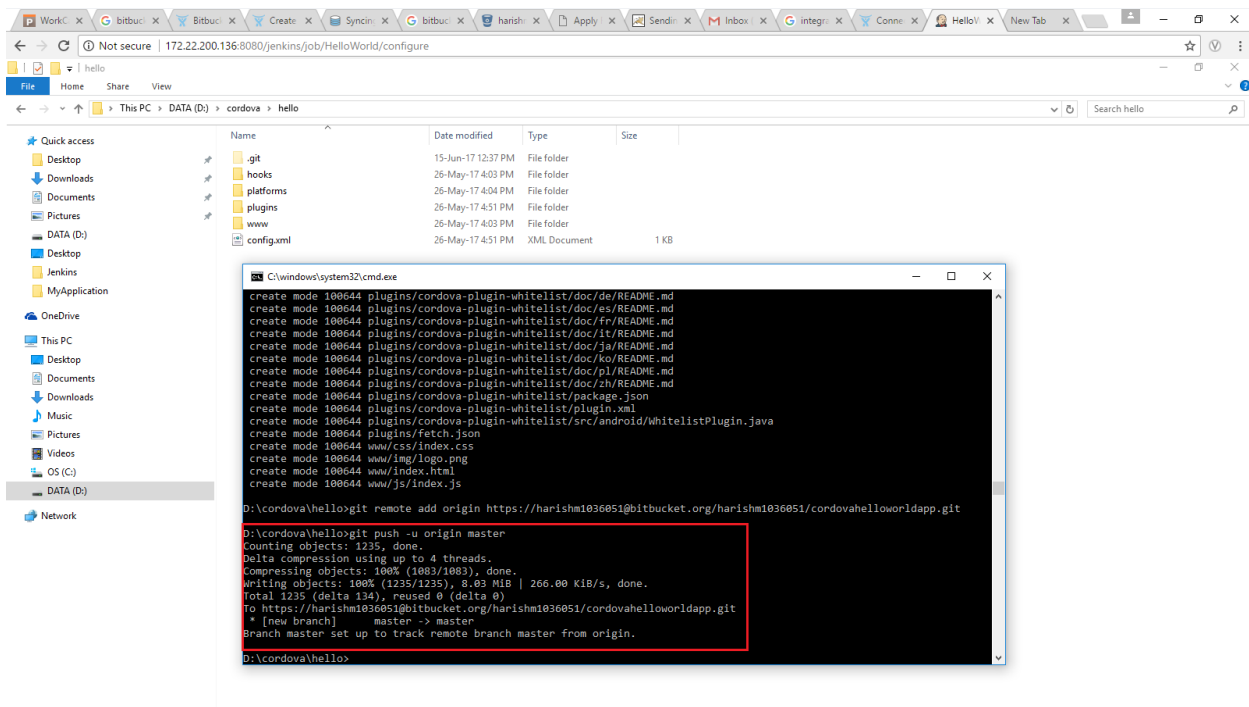
```
git remote add origin
```

```
https://harishm1036051@bitbucket.org/harishm1036051/cordovahelloworldapp.git
```



3.3. Push the source code into Bitbucket repository.

`git push -u origin master`

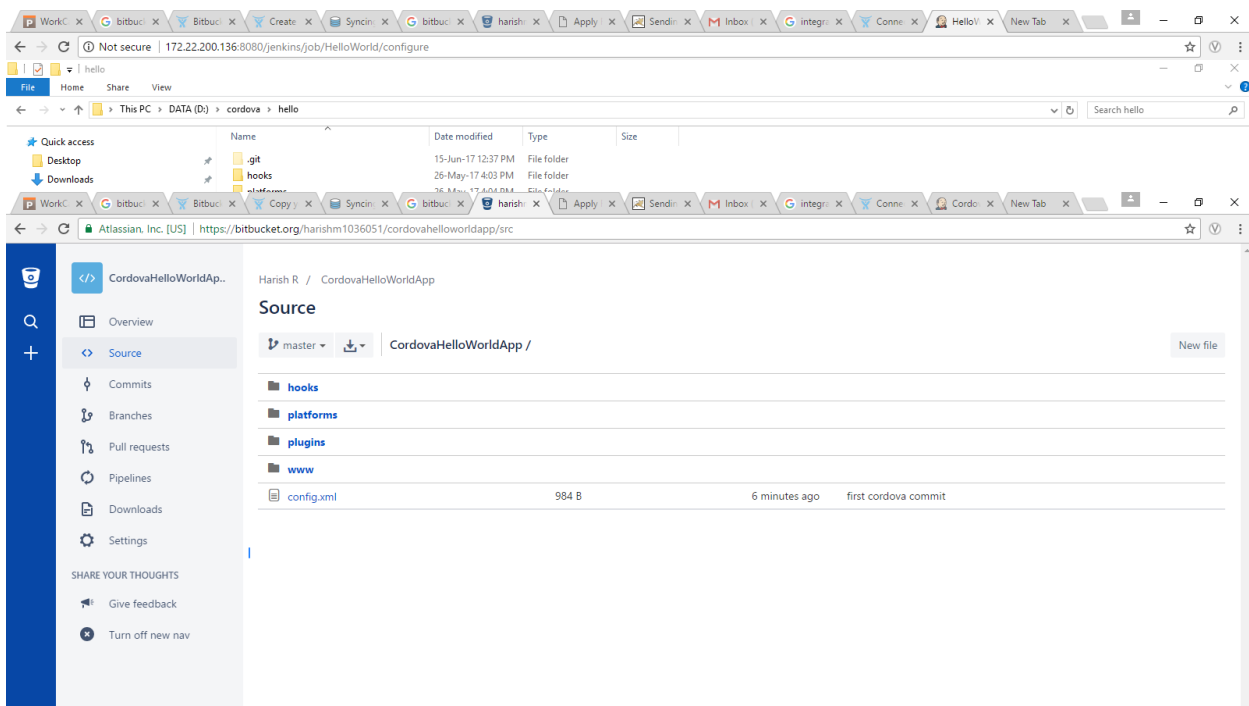


The screenshot shows a Windows File Explorer window displaying the contents of the 'D:\cordova\hello' directory. The files listed are: .git, hooks, platforms, plugins, www, and config.xml. Overlaid on this is a terminal window running the following commands:

```

D:\cordova\hello>git remote add origin https://harishm1036051@bitbucket.org/harishm1036051/cordova-hello-worldapp.git
D:\cordova\hello>git push -u origin master
Counting objects: 1235, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (1083/1083), done.
Writing objects: 100% (1235/1235), 8.03 MiB | 266.00 KiB/s, done.
Total 1235 (delta 134), reused 0 (delta 0)
To https://harishm1036051@bitbucket.org/harishm1036051/cordova-hello-worldapp.git
 * [new branch] master -> master
Branch master set up to track remote branch master from origin.
D:\cordova\hello>
  
```

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

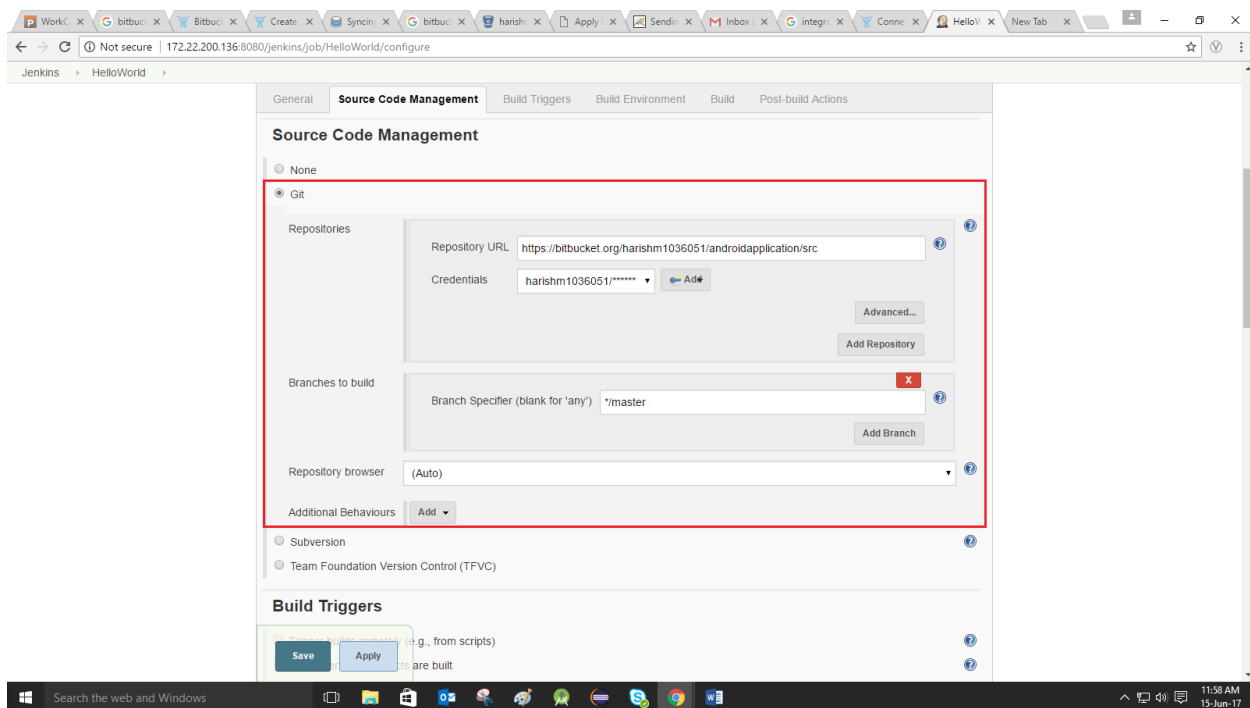


The screenshot shows the Bitbucket web interface for the repository 'CordovaHelloWorldApp'. The 'Source' tab is selected, displaying the file structure of the repository. The files listed are: hooks, platforms, plugins, www, and config.xml. The 'config.xml' file is highlighted, showing its size (984 B) and the commit message 'first cordova commit'.

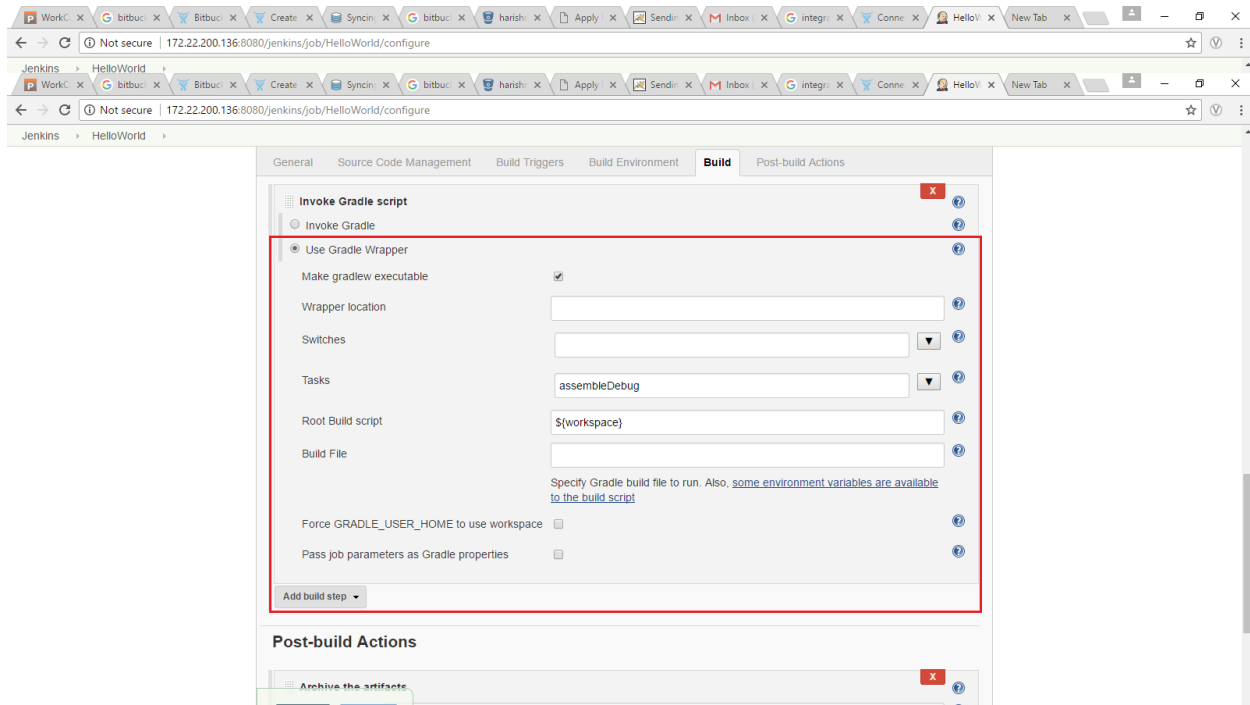
4. GRADLE BUILD FOR NATIVE ANDROID APPS IN JENKINS USING BITBUCKET REPOSITORY

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.



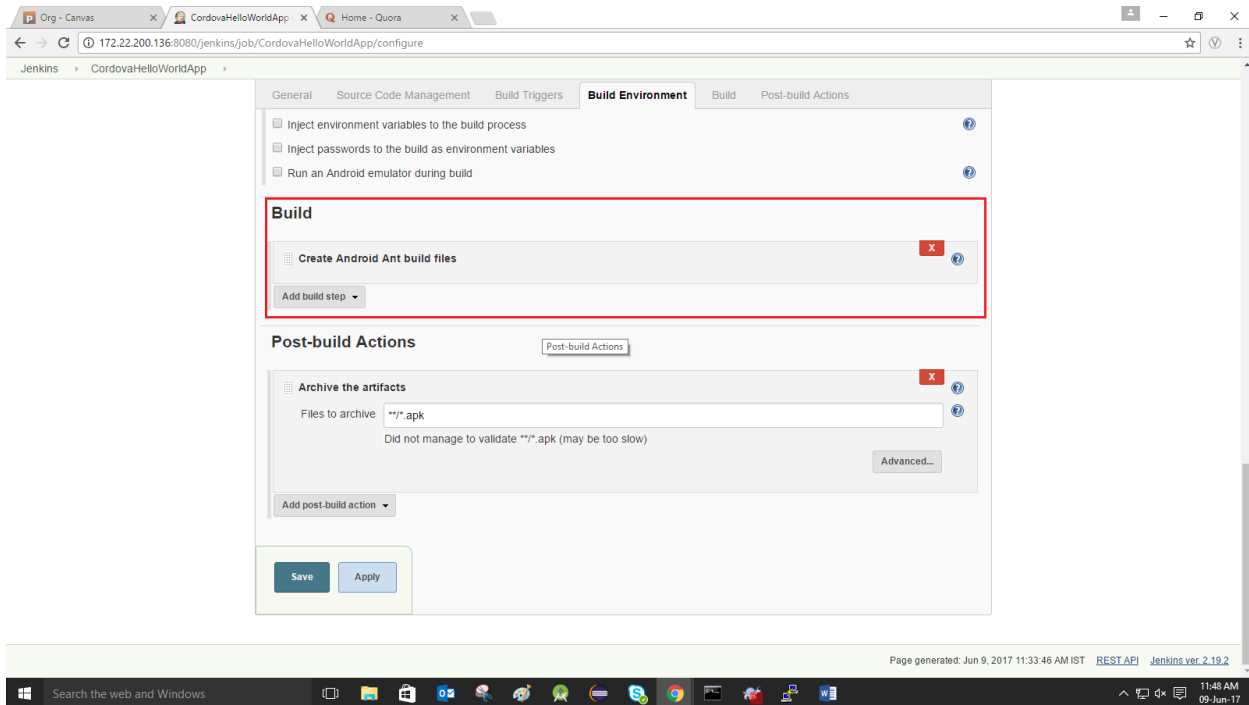
- 4.2. In Build tab, select **Invoke Gradle script** under **add build step**. Select **Use Gradle Wrapper** and check **make gradlew executable**. Give the tasks name as **assembleDebug** and root build script path as **\${workspace}**.



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**.



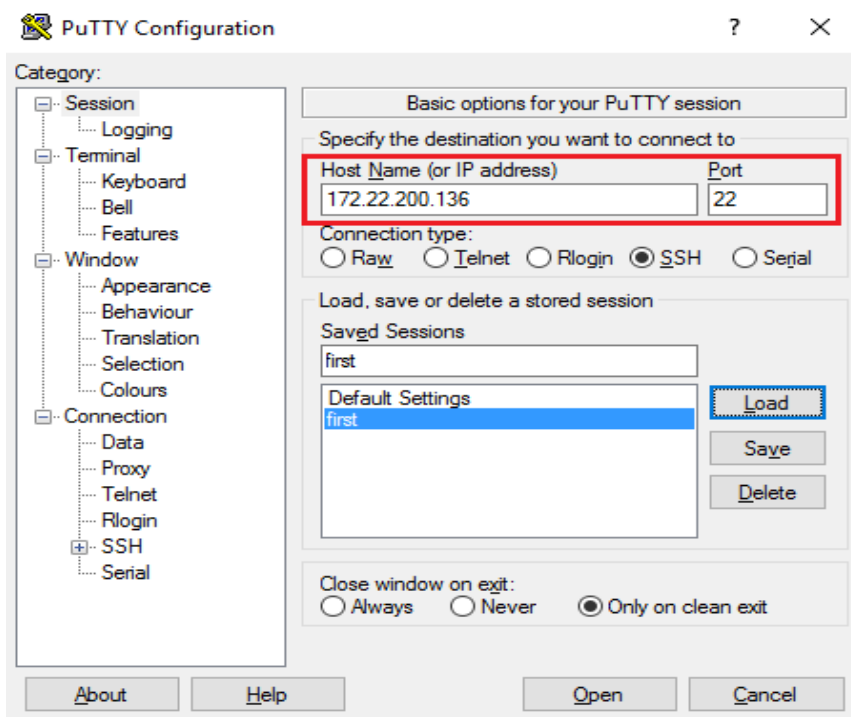
The screenshot shows the Jenkins job configuration page for 'CordovaHelloWorldApp'. The 'Build' tab is selected, and the 'Build' section is highlighted with a red box. In this section, 'Create Android Ant build files' is the selected build step. Below the 'Build' section, the 'Post-build Actions' section shows 'Archive the artifacts' with 'Files to archive' set to '**/*.apk'. The 'Save' button is visible at the bottom of the configuration page.

- 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.

```
administrator@mobilespace: ~  
login as: administrator  
administrator@172.22.200.136's password:  
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com/  
  
System information disabled due to load higher than 2.0  
  
342 packages can be updated.  
226 updates are security updates.  
  
New release '16.04.2 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
Last login: Wed Jun  7 12:19:39 2017 from 10.60.229.37  
administrator@mobilespace:~$
```


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:

<code>-D,--define <arg></code>	Define property
<code>-h,--help</code>	Display help information
<code>-v,--version</code>	Display version information
<code>-X,--debug</code>	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.

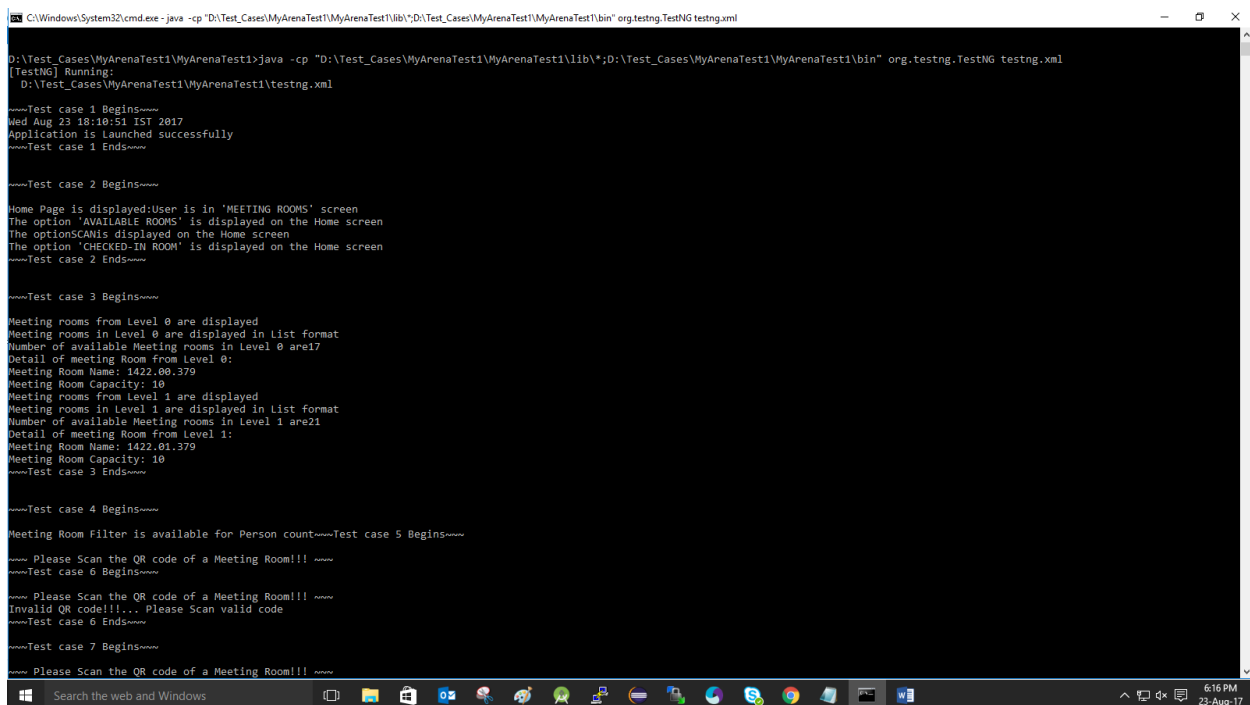
11.2. Open Command Prompt in windows.

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**



```
C:\Windows\System32\cmd.exe - java -cp "D:\Test_Cases\MyArenaTest1\MyArenaTest1\lib\*;D:\Test_Cases\MyArenaTest1\MyArenaTest1\bin" org.testng.TestNG testng.xml

D:\Test_Cases\MyArenaTest1\MyArenaTest1>java -cp "D:\Test_Cases\MyArenaTest1\MyArenaTest1\lib\*;D:\Test_Cases\MyArenaTest1\MyArenaTest1\bin" org.testng.TestNG testng.xml
[TestNG] Running:
  D:\Test_Cases\MyArenaTest1\MyArenaTest1\testng.xml

~~~Test case 1 Begins~~~
Wed Aug 23 18:10:51 IST 2017
Application is launched successfully
~~~Test case 1 Ends~~~

~~~Test case 2 Begins~~~
Home Page is displayed:User is in 'MEETING ROOMS' screen
The option 'AVAILABLE ROOMS' is displayed on the Home screen
The optionSCANis displayed on the Home screen
The option 'CHECKED-IN ROOM' is displayed on the Home screen
~~~Test case 2 Ends~~~

~~~Test case 3 Begins~~~
Meeting rooms from Level 0 are displayed
Meeting rooms in Level 0 are displayed in List format
Number of available Meeting rooms in Level 0 are17
Detail of meeting Room from Level 0:
Meeting Room Name: 1422.00.379
Meeting Room Capacity: 10
Meeting rooms from Level 1 are displayed
Meeting rooms in Level 1 are displayed in List format
Number of available Meeting rooms in Level 1 are21
Detail of meeting Room from Level 1:
Meeting Room Name: 1422.01.379
Meeting Room Capacity: 10
~~~Test case 3 Ends~~~

~~~Test case 4 Begins~~~
Meeting Room Filter is available for Person count~~~Test case 5 Begins~~~

~~~ Please Scan the QR code of a Meeting Room!!! ~~~
~~~Test case 6 Begins~~~

~~~ Please Scan the QR code of a Meeting Room!!! ~~~
Invalid QR code!!!... Please Scan valid code
~~~Test case 6 Ends~~~

~~~Test case 7 Begins~~~

~~~ Please Scan the QR code of a Meeting Room!!! ~~~
```

12. APPIUM TESTING FOR NATIVE ANDROID APPS

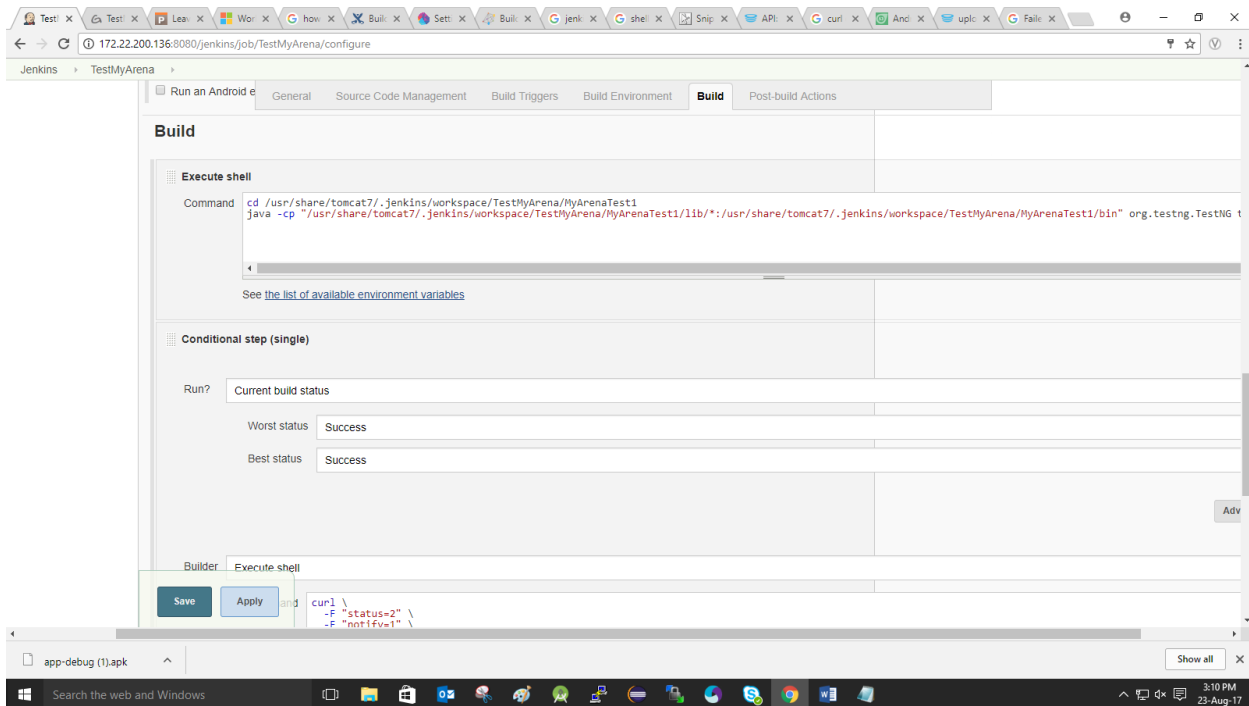
Steps to be done are shown below:

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.

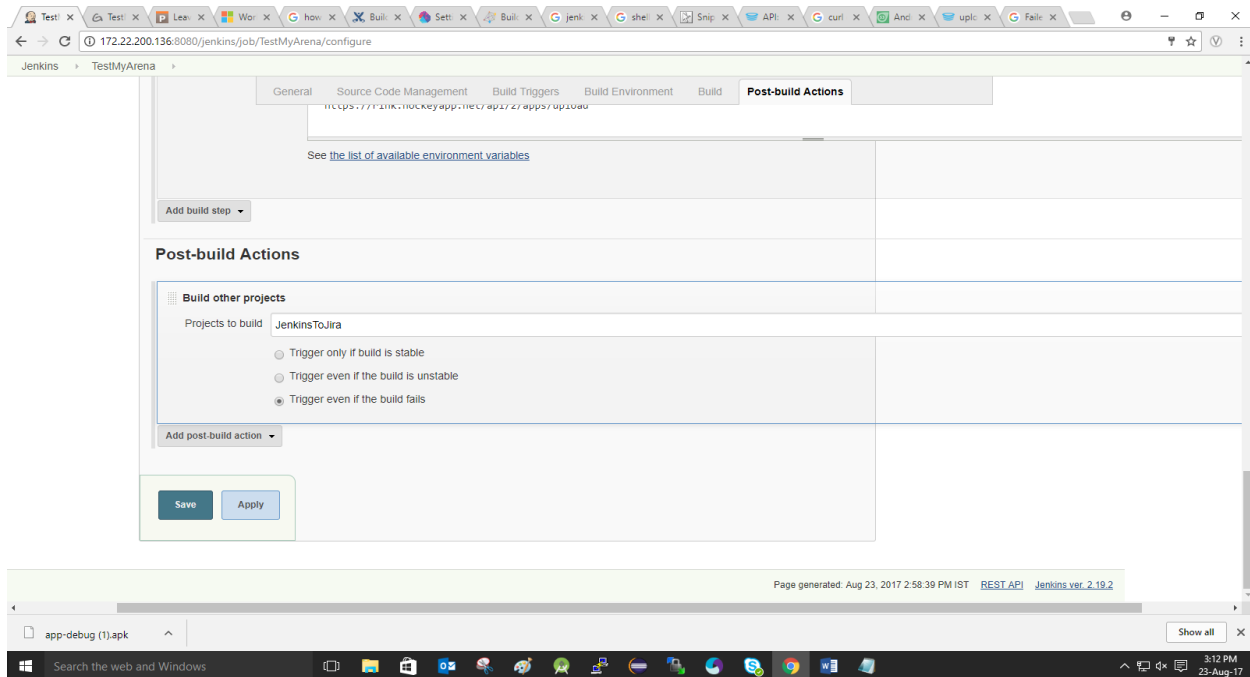
12.2. In the **Build** tab, select **Execute Shell**. Then provide the below commands in it

```
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
```

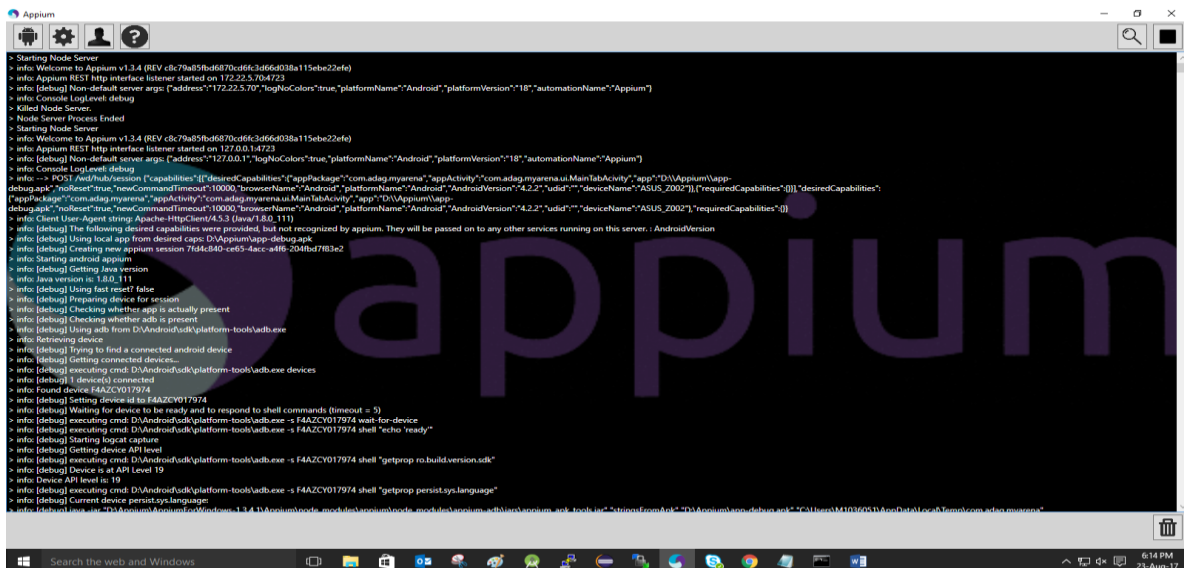


12.3. In the **Post-build** Actions tab, select **Build other projects** and provide the next job that has to be performed after testing the mobile application i.e logging jira bug Job.



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:

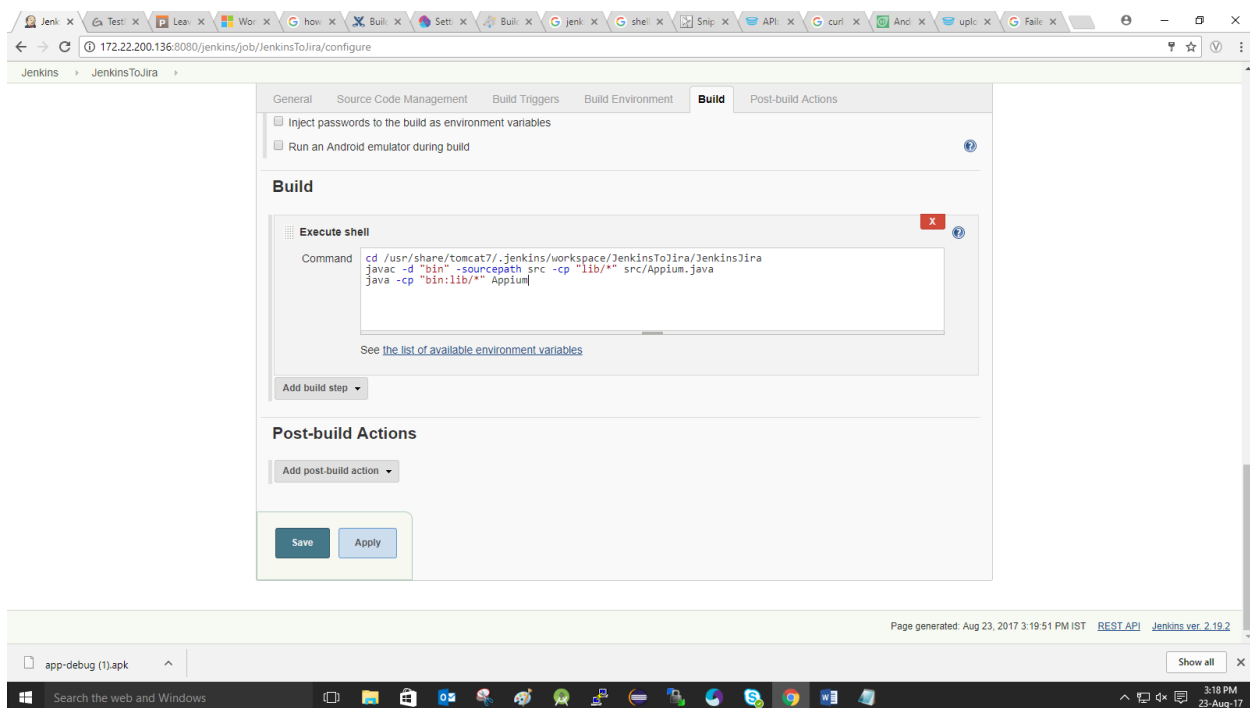
15.1. Create a job in Jenkins by following the same steps that of **Gradle build for native android apps in Jenkins using SVN Repository** mentioned earlier until **Build Environment** section.

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
```



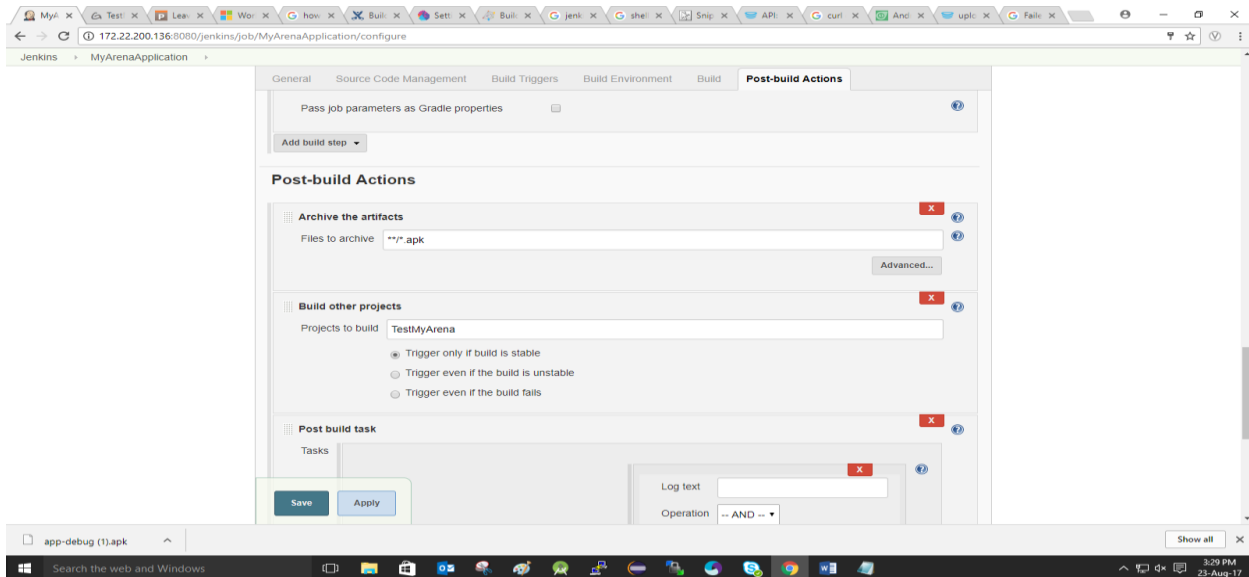
15.3. Click on **Save**.

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

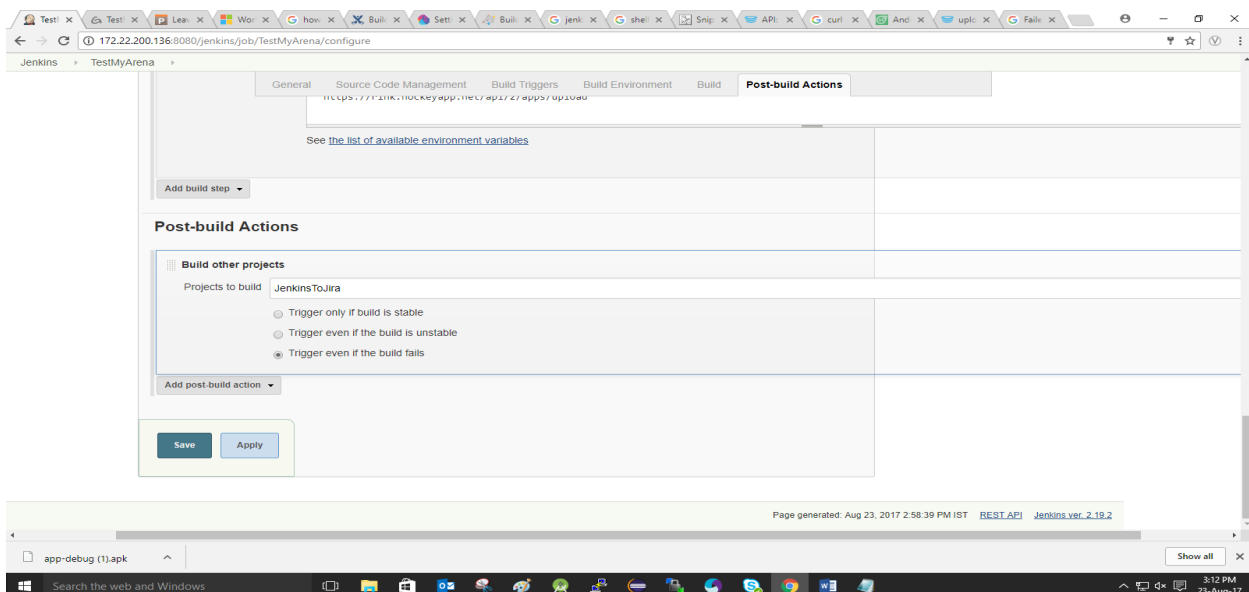
16. CREATING JOBS PIPELINE IN SEQUENCE IN JENKINS

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**.



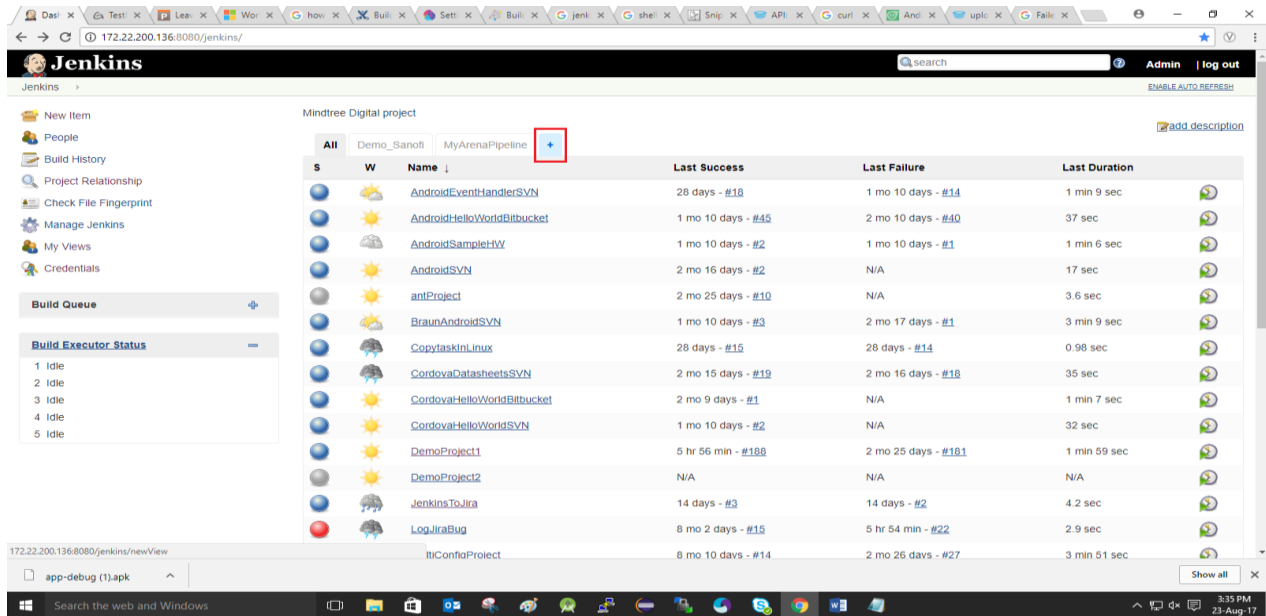
16.2. In the **Post-build** Actions tab of **Automation testing Job**, select **Build other projects** and provide the next job that has to be performed after testing the mobile application i.e **logging jira bug Job**.



17. CREATING BUILD PIPELINE VIEW IN JENKINS

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

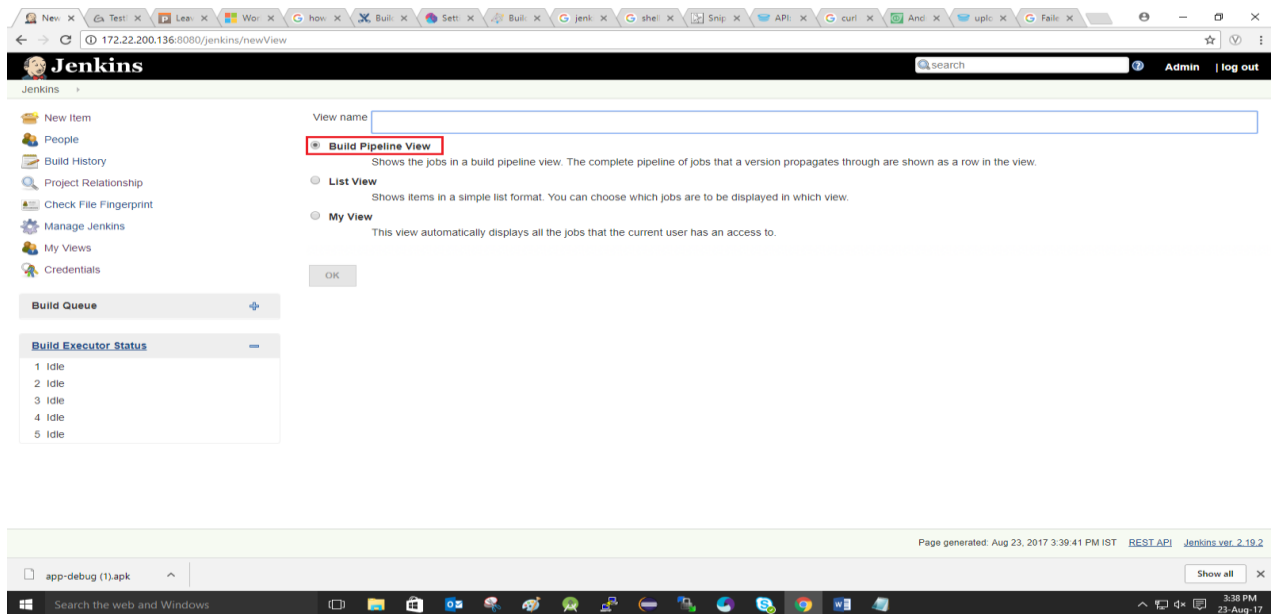
17.1. Click on the '+' icon in the Jenkins Dashboard.



The screenshot shows the Jenkins Dashboard. On the left sidebar, there is a 'New Item' button. In the main content area, under the 'Mindtree Digital project' section, there is a table of build jobs. A red box highlights the '+' icon in the top right corner of the table, which is used to create a new build pipeline.

S	W	Name ↓	Last Success	Last Failure	Last Duration
		AndroidEventHandlersSVN	28 days - #18	1 mo 10 days - #14	1 min 9 sec
		AndroidHelloWorldBitbucket	1 mo 10 days - #45	2 mo 10 days - #40	37 sec
		AndroidSampleHW	1 mo 10 days - #2	1 mo 10 days - #1	1 min 6 sec
		AndroidSVN	2 mo 16 days - #2	N/A	17 sec
		antProject	2 mo 25 days - #10	N/A	3.6 sec
		BraunAndroidSVN	1 mo 10 days - #3	2 mo 17 days - #1	3 min 9 sec
		CopytaskInLinux	28 days - #15	28 days - #14	0.98 sec
		CordovaDataSheetsSVN	2 mo 15 days - #19	2 mo 16 days - #18	35 sec
		CordovaHelloWorldBitbucket	2 mo 9 days - #1	N/A	1 min 7 sec
		CordovaHelloWorldSVN	1 mo 10 days - #2	N/A	32 sec
		DemoProject1	5 hr 56 min - #188	2 mo 25 days - #181	1 min 59 sec
		DemoProject2	N/A	N/A	N/A
		JenkinsToJira	14 days - #3	14 days - #2	4.2 sec
		LogJiraBug	8 mo 2 days - #15	5 hr 54 min - #22	2.9 sec
		ItiConfoProject	8 mo 10 days - #14	2 mo 26 days - #27	3 min 51 sec

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



The screenshot shows the Jenkins 'New Item' dialog. The 'View name' field is empty. The 'Build Pipeline View' option is selected and highlighted with a red box. Below the options, there is an 'OK' button.

View name:

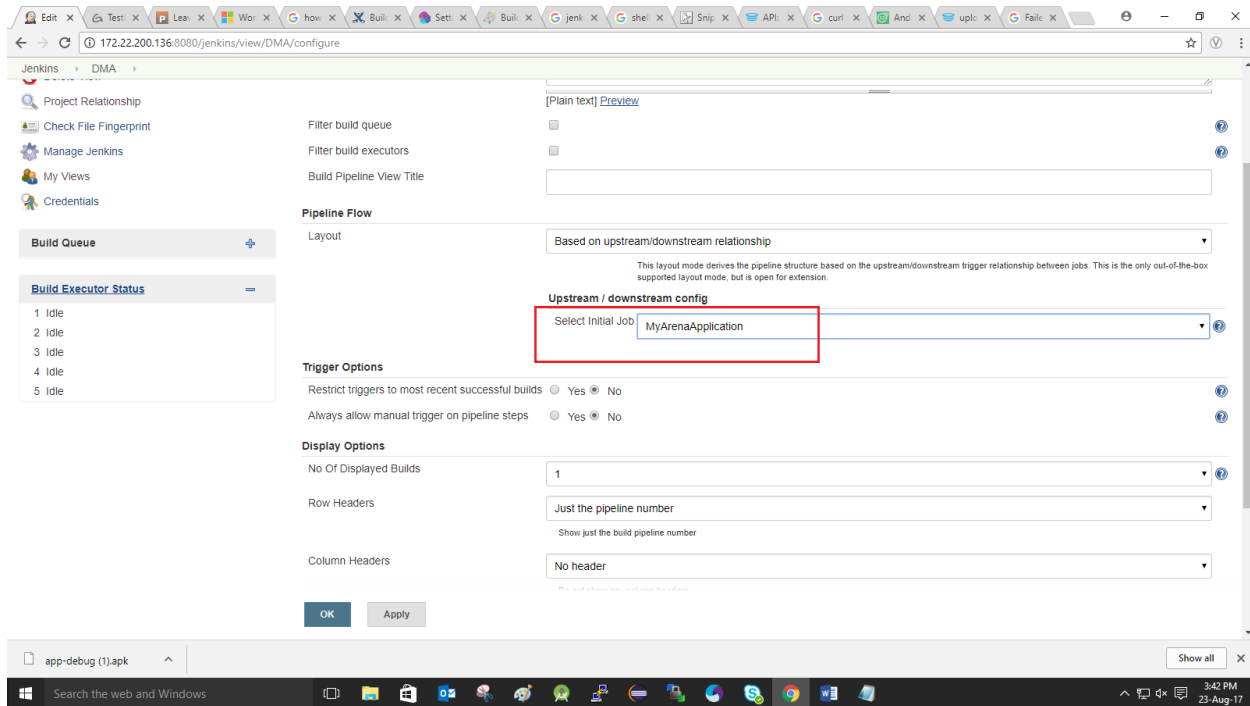
☒ **Build Pipeline View**
Shows the jobs in a build pipeline view. The complete pipeline of jobs that a version propagates through are shown as a row in the view.

☐ List View
Shows items in a simple list format. You can choose which jobs are to be displayed in which view.

☐ My View
This view automatically displays all the jobs that the current user has access to.

OK

17.3. Set the initial Job to be performed.



The screenshot shows the Jenkins Pipeline Configuration page for a job named 'DMA'. The 'Upstream / downstream config' section is highlighted with a red box, indicating the 'Select Initial Job' is set to 'MyArenaApplication'. The 'Build Queue' section shows 5 idle executors. The 'Pipeline Flow' section is set to 'Based on upstream/downstream relationship'. The 'Trigger Options' section has 'Restrict triggers to most recent successful builds' set to 'No' and 'Always allow manual trigger on pipeline steps' set to 'No'. The 'Display Options' section has 'No Of Displayed Builds' set to '1', 'Row Headers' set to 'Just the pipeline number', and 'Column Headers' set to 'No header'. The 'OK' and 'Apply' buttons are at the bottom.

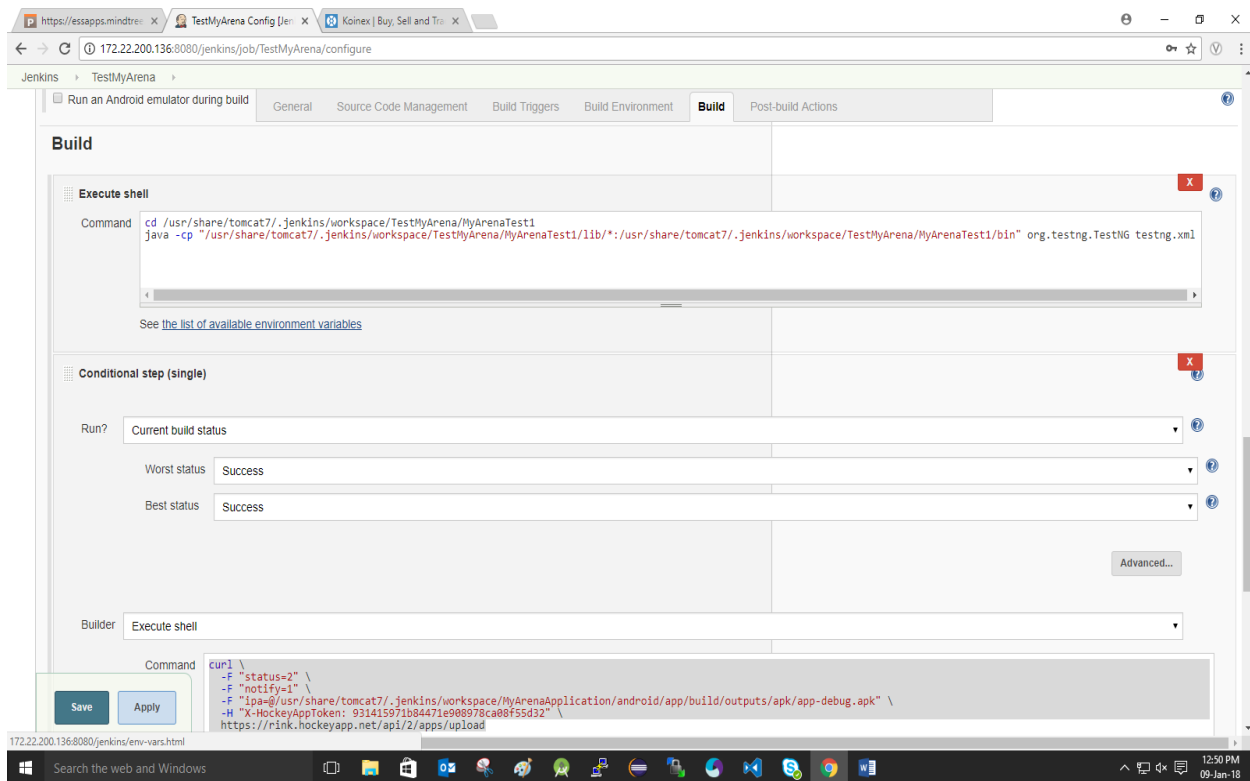
17.4. Click on **Save**.

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
```