# **BUILD ANDROID APPLICATION WITH JENKINS PIPELINE**

**[Setup Android SDK](https://medium.com/@prashant_48386/setup-android-sdk-on-centos-9a420b928e35)**

**[Setup Jenkins](https://medium.com/appgambit/setup-jenkins-on-centos-with-docker-for-selenium-b7dba07b9ffa)**

After setting up Jenkins we need to configure Android SDK on Jenkins.

## **Global Properties**

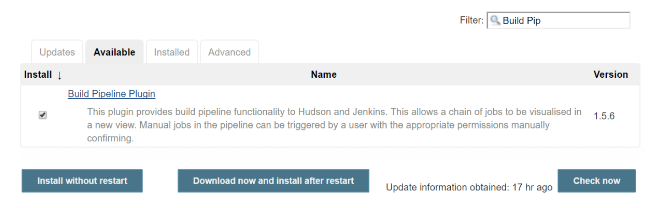
Open Jenkins: ****Manage Jenkins**** -> ****Configure System**** -> ****Global properties****. Mark “Environment variables” and add:

* ANDROID\_HOME : <Path of android-SDK directory>
* JAVA\_HOME : <Path of java-SDK>

## **Install the plugin from Manage Jenkins**

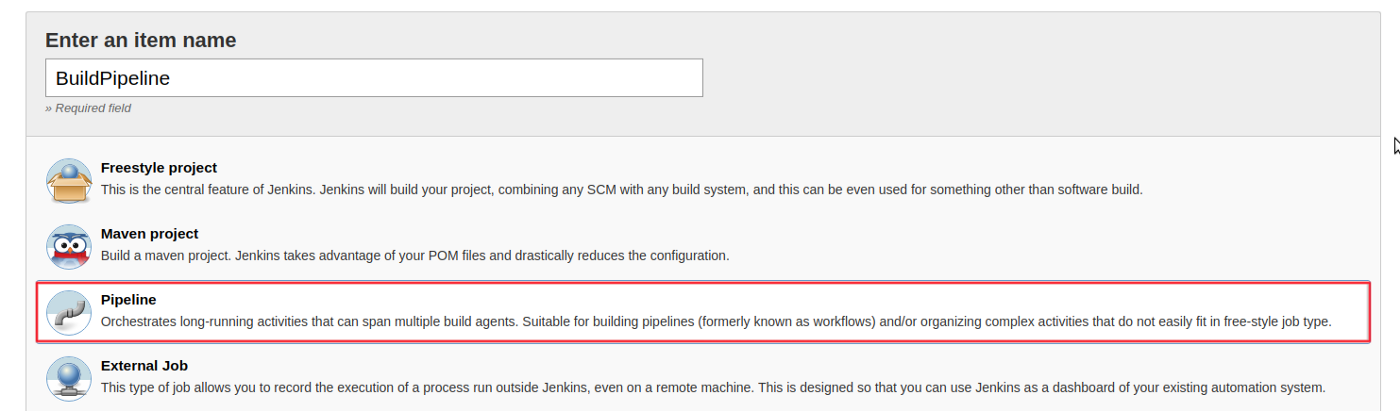
Go to Jenkins Dashboard > Manage Jenkins > Manage Plugins > Available > search for ****Build Pipeline**** > Install.

If you are already installed this plugin on your Jenkins it will display in the Installed section.



## **Create an Android build job**

* Open Jenkins -> ****New Item****. Enter any job name > Choose ****Pipeline >**** Click OK.



* ****Pipeline >****There 2 options for Jenkinsfile.

*Pipeline Script*

*Pipeline Script From SCM*

## **Pipeline Script:**

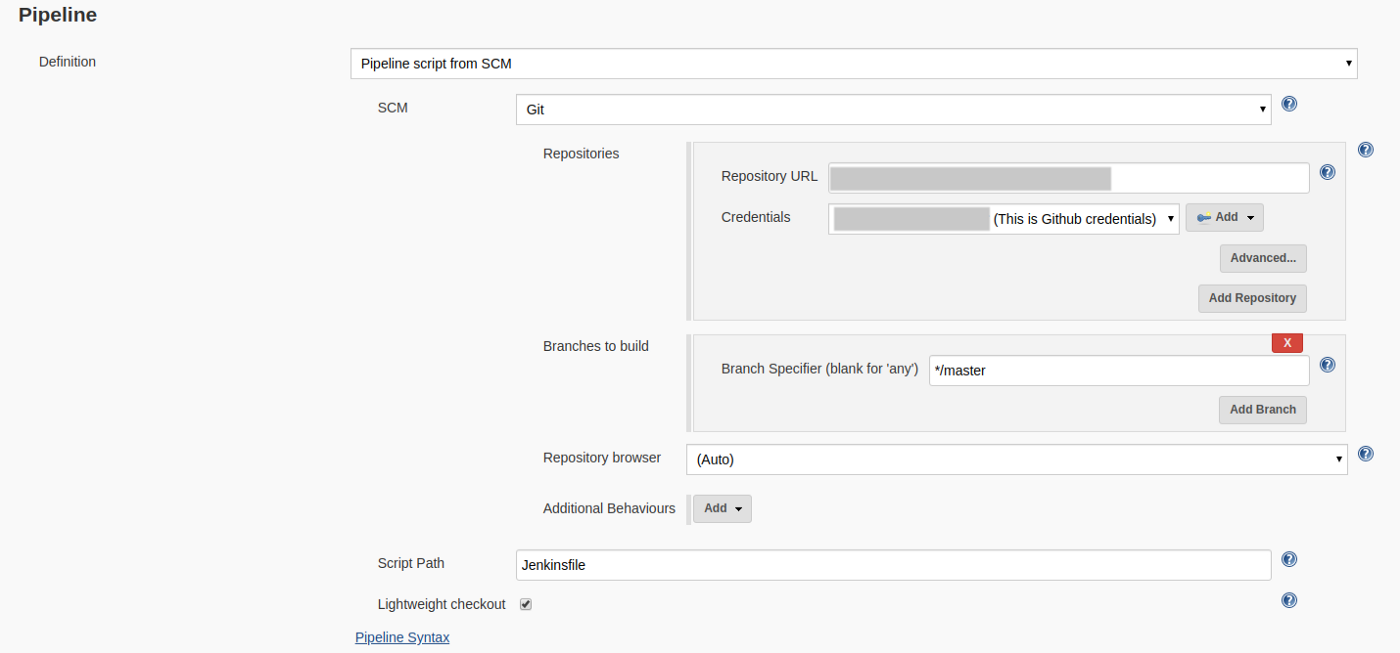
You can write your Pipeline code directly on Jenkins job.



## **Pipeline Script From SCM:**

Pipeline supports fetching the DSL (Domain Specific Language) script from the SCM. Typically called Jenkinsfile and located in the root of the project.

* Select “Pipeline script from SCM” from the definition.
* Select Git as SCM
* Git URL to your repo. Take this URL from Github. It should be the format of ****git@github.com:{username}/{repo}.git****
* Credentials: Select the one you created before.
* Branches to build: $branch



For both options, you can use this demo pipeline code.

def err = null

try {

node {

stage('Preparation') {

git credentialsId: '**your\_credentials\_id**', url: 'https://github.com/**username/repo**/'

}

stage('Dependencies') {

sh 'sudo npm install -g react-native-cli'

sh 'npm install'

sh 'react-native link'

sh 'export JAVA\_HOME=/opt/jdk1.8.0\_201'

sh 'export JRE\_HOME=/opt/jdk1.8.0\_201/jre'

sh 'export PATH=$PATH:/opt/jdk1.8.0\_201/bin:/opt/jdk1.8.0\_201/jre/bin'

sh 'echo $JAVA\_HOME'

}

stage('Clean Build') {

dir("android") {

sh "pwd"

sh 'ls -al'

sh './gradlew clean'

}

}

stage('Build release ') {

parameters {

credentials credentialType: 'org.jenkinsci.plugins.plaincredentials.impl.FileCredentialsImpl', defaultValue: '5d34f6f7-b641-4785-frd5-c93b67e71b6b', description: '', name: 'keystore', required: true

}

dir("android") {

sh './gradlew assembleRelease'

}

}

stage('Compile') {

archiveArtifacts artifacts: '\*\*/\*.apk', fingerprint: true, onlyIfSuccessful: true

}

}

} catch (caughtError) {

err = caughtError

currentBuild.result = "FAILURE"

} finally {

if(currentBuild.result == "FAILURE"){

sh "echo 'Build FAILURE'"

}else{

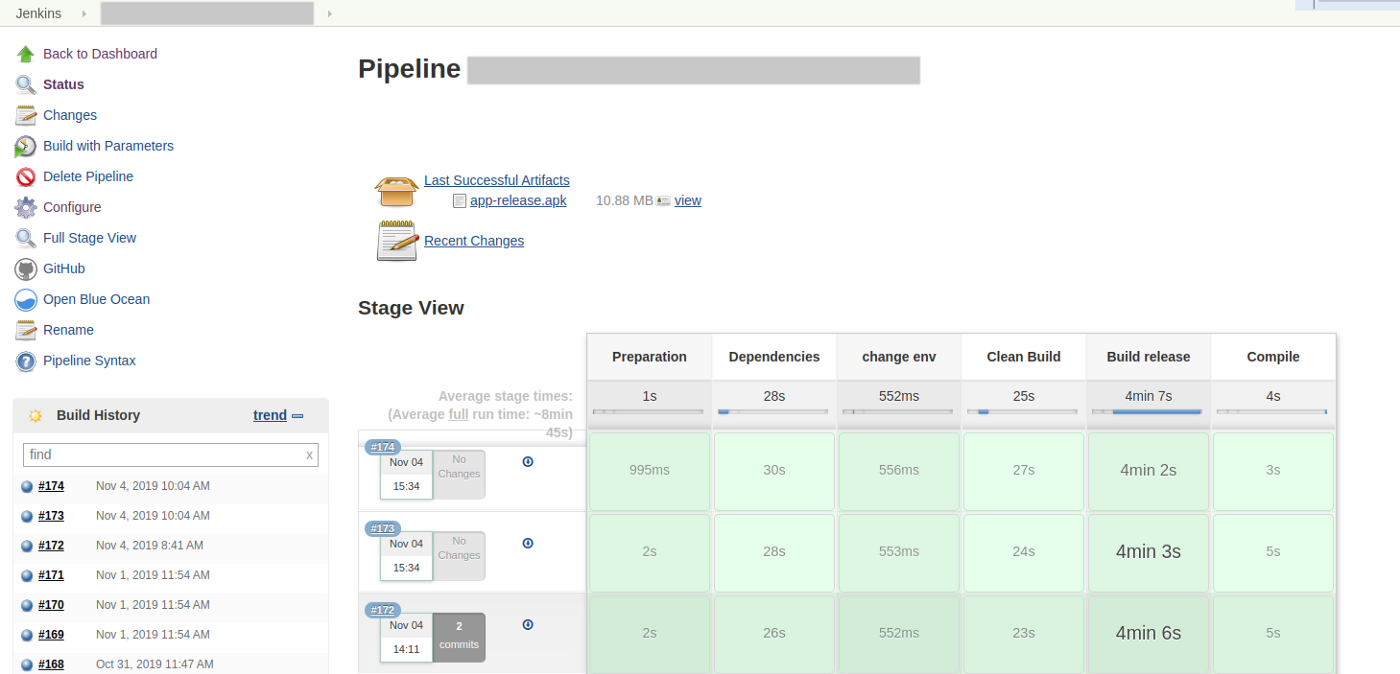
sh "echo 'Build SUCCESSFUL'"

}

}

Click on ****Build Now****.

Now, you can watch the progress in ****Console Output****. Once the job is finished successfully, you will see the APK.



You can also integrate your build with slack using the [Slack Notifications plugin](https://wiki.jenkins.io/display/JENKINS/Slack+Plugin" \t "https://medium.com/appgambit/_blank).

You can create a Slack trigger to run this job So you don’t need to open Jenkins whenever you create the build, Just execute the slack command and your build is up and running.