A REPORT

on

Application Development & Deployment for Android Application

COMPUTER SCIENCE & ENGINEERING

Under Guidance of
Rajula Vineet Reddy
IMT2014045

Submitted by:
PANKAJ
MT2018075
…

INDEX

1. Abstract... **1**

2. Introduction **1-3**

2.1 Why Dev-Ops.. 1-3

2.2 About Application 3

3. SDLC Phases **4-12**

3.1 Scope 4

3.2 Architecture... 4-5

3.2.1 Jenkins in Docker Configuration 4-5

3.3 SCM 5-6

3.4 Build 6-7

3.5 Test..7-10

3.6 Deploy 10

3.7 Monitor...10-12

4. Results **12-13**

4.1 Scalability 13

5. Future Work **13**

6. Conclusion **13**

7. References **14**

1 Abstract

Now a day, it is obvious in our day to day life to have a great music playlist along with some great source of listening entertainment. For the sake of same, we came up with the idea to have such a player for our android phones that can play our favourite songs along with the built in internet FM in it.

In our app, we'll have toggle button that can allows users to switch between the two easily. Also, it'll play the first song in their respective domain.

We'll use Dev-Ops tools in order to track development, testing, deployment & monitoring tasks easily. This helps us to coordinate among us and come up with the best product for the end user in the narrow GTM window.

Following tools will be used for the above:

- Android Studio 3.2
- Jenkins in Docker
- Git, Github & GitAPI
- JUnit
- Go script for release
- Python script for monitoring

2 Introduction

We'll create development environment in local system and try to achieve Dev-Ops environment most of our routine task like Code Management, Testing, & Deployment to automate our tasks as much as we can. Following flow diagram shows the clear phase of SDLC.

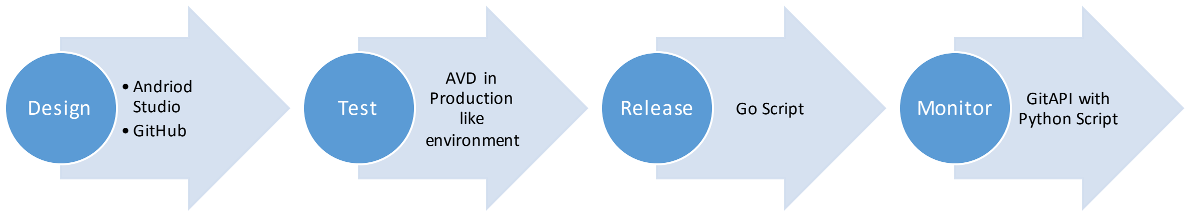


Figure 1. SDLC phases

2.1 Why Dev-Ops

- ➢ Development wants to add new features quickly
- ➢ Operations want stability

The biggest business problem in the new fully digitalized world is not the cost of IT operations or the DevOps team – it is the lost opportunity on not executing your business service delivery with enough quality and speed compared with the new breed of competitors attacking your business segment.

1. Everything needs software.
2. Software runs on a server to become a service.
3. Delivering a service from inception to its users is too slow and error-prone.
4. There are internal friction points.
5. Defects released into production.
6. Inability to diagnose production issues quickly.
7. Problems appear in some environments only.
8. Blame shifting or finger pointing.
9. Long delays while dev, QA, ops team waits on response from other teams.
10. Releases slip or fail - Delay causes money loss.
11. Traditional Thinking: Conflicting Perspectives and Lesser Collaboration

    Dev's job is to add new features; Ops' job is to keep the site stable and fast.

- DevOps emerged from an effort by businesses to respond more rapidly to market changes.
- It formed out of a fundamental need and is based on a simple philosophy- business works best, when efforts are coordinated and collaborative, so its growth has been rapid.
- The DevOps approach was designed to ensure that high-quality updated software gets into the hands of users more quickly. Though the DevOps is only a few years old, the DevOps movement is so widespread.

Temporary DevOps Team

❖ The members of the temporary team will ‘translate’ between Dev-speak and Ops-speak. If you start to see the value of bringing Dev and Ops together, then the temporary team has a real chance of achieving its aim.
❖ This type can be used as a precursor of type 1 (smooth collaboration) and it’s potential effectiveness is initially low and once it’s transformed to Type 1 then its potential effectiveness is high.
❖ DevOps principles involves **CALMS**:

Culture - people > process > tools

Automation - Infrastructure as code

Lean - Small batch sizes, focus on value for end user
Measure - Measure everything; Show improvement
Sharing - Collaboration between Dev and Ops

Due to this huge impact of Dev-Ops, we try to use it in our project. This helps us to get familiarize with the tools and also gives a great learning experience.

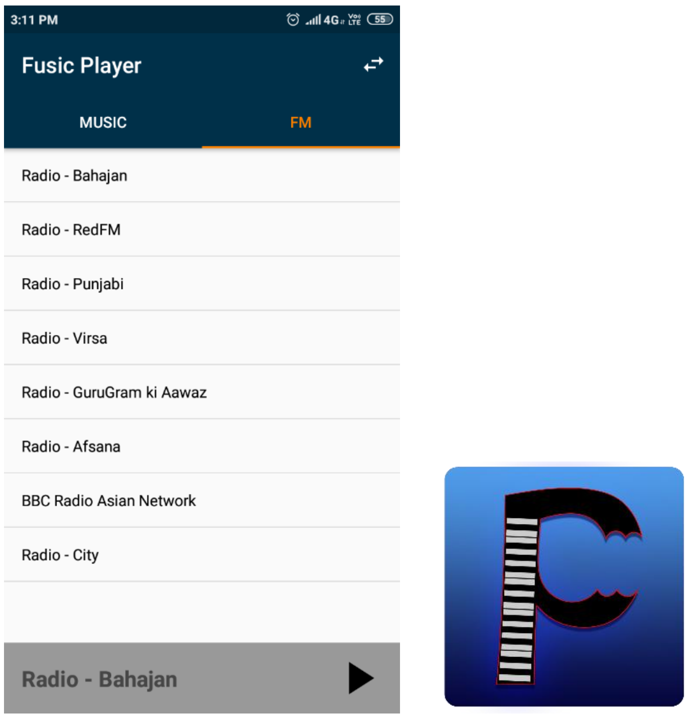
2.2 About Application

The application is based on android as of now and is compatible for API greater than 16. Following are the features of the app mentioned below:

- ✓ Uses Google's Open Source ExoPlayer for media functions.
- ✓ Toggle button on top-bar of player.
- ✓ Can access all *.mp3 file in phone. File READ access is required.
- ✓ Can play online RADIO channels that air on internet.

GitHub Link

The layout of the app can be seen below:



3:11 PM

Fusic Player

MUSIC FM

Radio - Bahajan
Radio - RedFM
Radio - Punjabi
Radio - Virsa
Radio - GuruGram ki Aawaz
Radio - Afsana
BBC Radio Asian Network
Radio - City

Radio - Bahajan

Figure 2. App Main Screen & Logo

3 SDLC Phase

3.1 Scope

This app allows us to switch between the FM & Music player easily without any hassles. The design of the app is built in such a way that each module is separated from each other by OOP concepts in Android SDK. Also, the driver class is separate and each of the module is accessing it only one at a time.

Test cases are written in such a way that it covers most of the Integration Testing of the app. Like it checks for the availability of fragment and list of stations/music etc.

Overall, any module can be editable or removed in no amount of time. Code is developed with clarity and proper commenting is maintained. Proper format of coding is also taken care of.

3.2 Architecture

We've used following Dev-Ops tools for the app design.

- Android Studio as IDE
- AVD (Android Virtual Device) for basic testing
- Jenkins in Docker for Continuous Integration
- GitHub for SCM (Source Code Management)
- Gradle as Build tool to generate artifact
- JUnit for Integration Testing
- Go scripts for Continuous Deployment
- Python scripts with GitAPI for Monitoring no of downloads of Release

We only focus to design the code. As we pushed our new code to git, Jenkins automatically captures it with no input or just a single click and rest all process is automatic and proper reports are generated for the same. Monitoring can also be done in one click in Jenkins workspace.

3.2.1 Jenkins in Docker Configuration

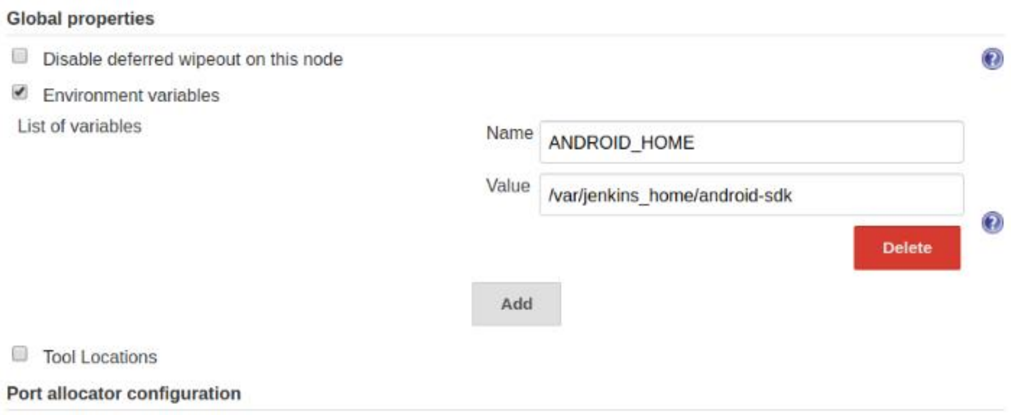
As our local system can be vulrable for data. So, Docker helps in order to save our useful data getting lost as it provides isolate environemrt for our particular tasks.

Following steps are used to configure docker for Continous Integration:

- [docker pull jenkins/jenkins] in Ubuntu Terminal of base machine
- [docker run --name JenkinsAndroid -p 8080:8080 -p 50000:50000 -v /home/panki/Jenkins:/var/jenkins_home jenkins/jenkins:lts] in base machine to run the docker jenkins container.
- Start jenkins in browser for initial setup @ localhost:8080
- After this, you need to setup Android SDK in the Jenkins Docker.
- Open another terminal to login into container.
- [docker exec --user root -it JenkinsAndroid /bin/bash] Login in docker container

From here every step is inside docker container

- Navigate to [cd /var/jenkins_home]
- Make directory [mkdir android-sdk]
- Download SDK[wget https://dl.google.com/android/repository/sdk-tools-linux-4333796.zip]
- Extract content [unzip sdk-tools-linux-4333796.zip]
- Accept License.
- Go to [cd /var/jenkins_home/android-sdk/bin]
- Run sdk update [./sdkmanager --update]
- Open License. [./sdkmanager --licenses]
- Accept all 5 by entering Y in terminal.
- Now, set the jenkins home variable in browser. Manage jenkins > Globle Configurations. Put values as screenshot.


Global properties

☐ Disable deferred wipeout on this node

☑ Environment variables

List of variables

Name: ANDROID_HOME

Value: /var/jenkins_home/android-sdk

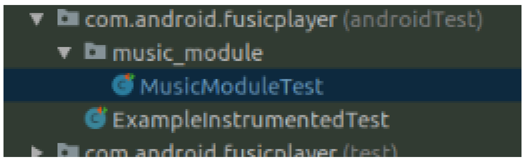
☐ Tool Locations

Port allocator configuration

Figure 3. Environment Variable Values

3.3 SCM

For Source Code Management, we are using in built VCS (Version Control System(**git**)) of Android Studio as it tracks the file changes that is modified or deleted in real time. See the below image, the .java file is modified but not committed to git. Hence, it is highlighted in blue.



▼ com.android.fusicplayer (androidTest)
▼ music_module
MusicModuleTest
ExampleInstrumentedTest
▶ com.android.fusicplayer (test)

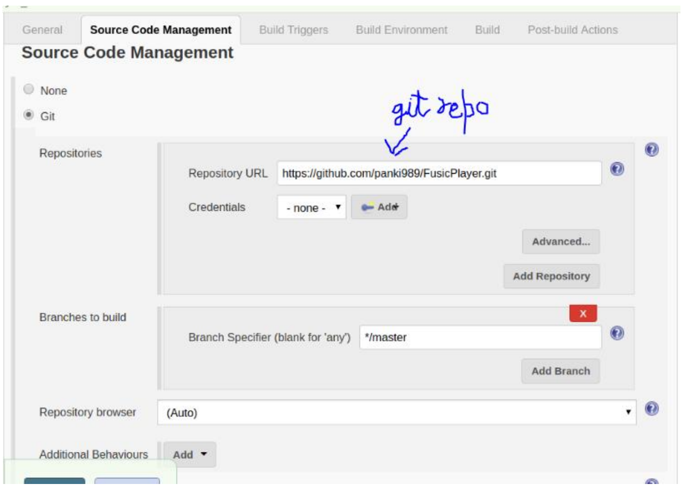
Figure 4. VCS in Android Studio

We can also use its GUI features to add, commit & push operations.

3.4 Build

We've now our Jenkins configured for the build activity. **Gradle** tool will be used in order to build the required artifact (*.apk). Let's create a new item in Jenkins for handling the build activity. Following are the steps.

- Click new item in Jenkins.
- Name it as FusicPlayer_build
- Select freestyle project & click OK.
- Now, fill the values as shown below:



Source Code Management

None
[✓ Git]

Repositories

Repository URL: https://github.com/panki989/FusicPlayer.git

Credentials: - none -

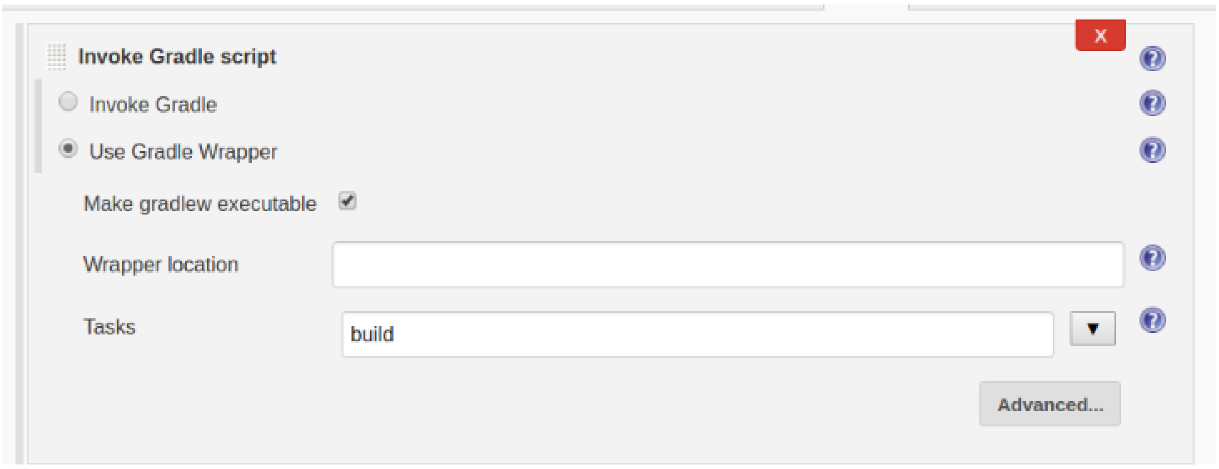
Branches to build

Branch Specifier (blank for 'any'): */master

Repository browser: (Auto)

Additional Behaviours

Figure 5(a). Build configuration values



Invoke Gradle script

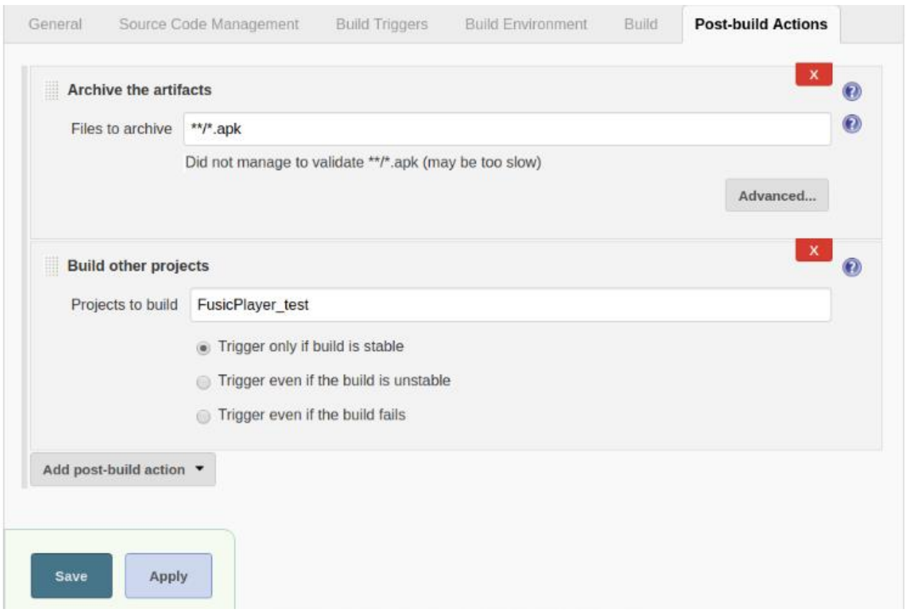
Invoke Gradle
[✓ Use Gradle Wrapper]

Make gradlew executable ✓

Wrapper location:

Tasks: build

Figure 5(b). Build Configuration values


General | Source Code Management | Build Triggers | Build Environment | Build | Post-build Actions

Archive the artifacts

Files to archive: **/*.apk

Did not manage to validate **/*.apk (may be too slow)

Advanced...

Build other projects

Projects to build: FusicPlayer_test

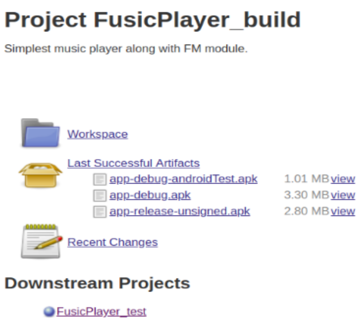
- [✓ Trigger only if build is stable]
- Trigger even if the build is unstable
- Trigger even if the build fails

Add post-build action

Save | Apply

Figure 5(c). Build Configuration values

- As of now, Leave the Build other project section. As it'll be used later to build pipeline.
- After this click on Build Now. Your *.apk file must be generated in the workspace of Jenkins.


Project FusicPlayer_build

Simplest music player along with FM module.

Workspace

Last Successful Artifacts

app-debug-androidTest.apk	1.01 MB	view
app-debug.apk	3.30 MB	view
app-release-unsigned.apk	2.80 MB	view

Recent Changes

Downstream Projects

FusicPlayer_test

Figure 6. Workspace & artifact for Build

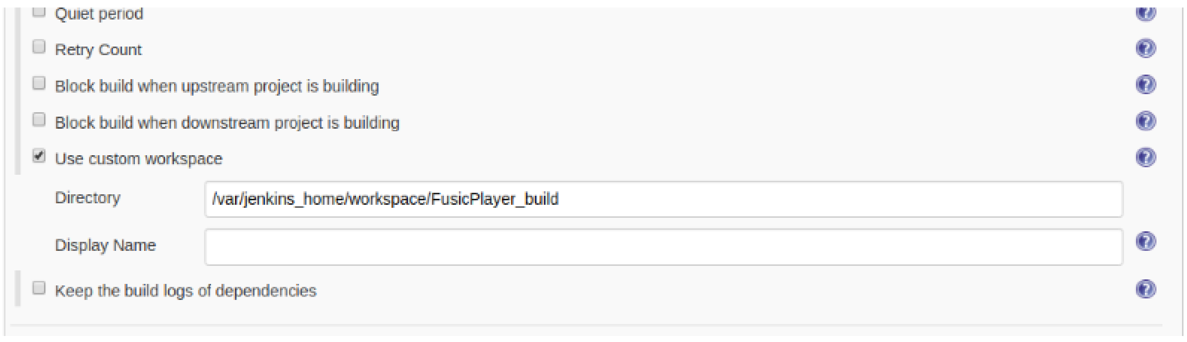
3.5 Test

We've now created our required *.apk file. But to test that in Jenkins, we need to create an **Android Virtual Device** in the Jenkins Docker so that out **JUnit** test-cases can run on that. Follow the below steps in to create the AVD:

From here every step is inside docker container

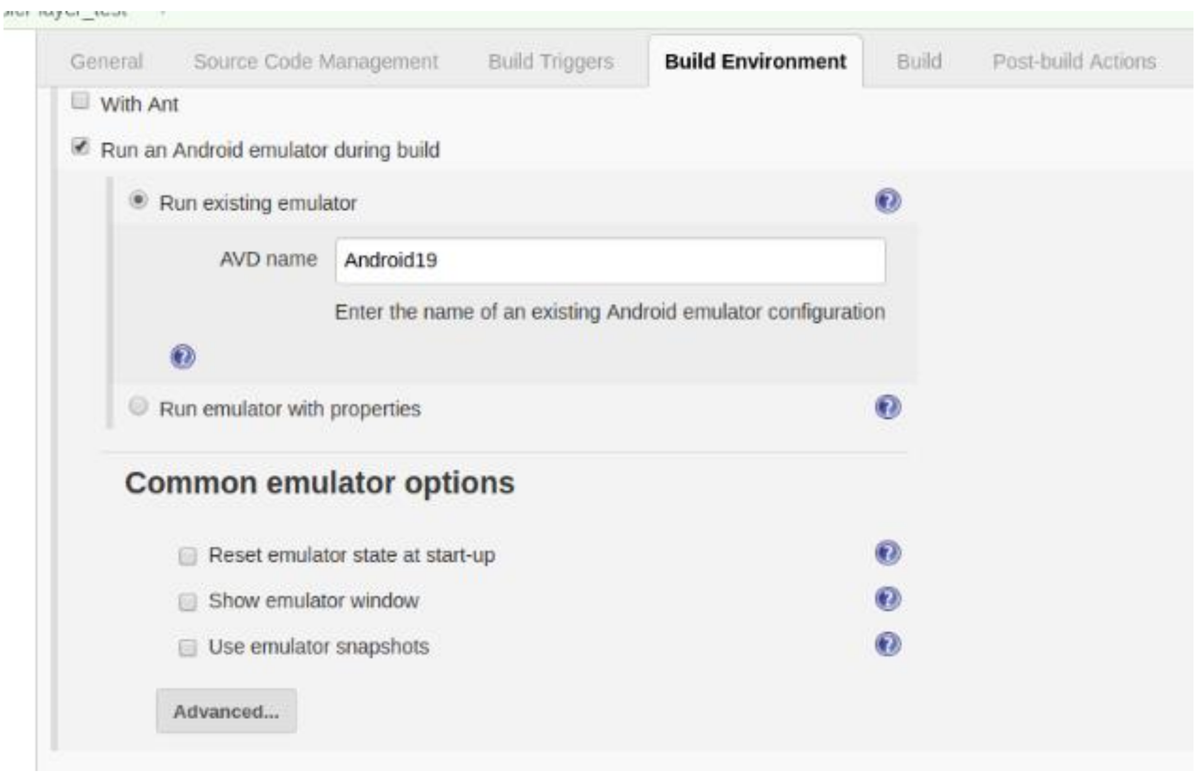
- Go to bin directory of sdk [cd /var/jenkins_home/android-sdk/tools/bin]
- Run [./avdmanager -v create avd -k 'system-images;android-19;default;x86' -n Android19]
- This will create a new AVD named Android19

- AVD will be created in directory [/root/.android/avd]
- We need to copy the files of avd folder from above to [/var/jenkins_home/.android/avd] as Jenkins will look for AVD here only.
- [cd /var/jenkins_home/.android/] Go in directory.
- [chown -R jenkins:jenkins avd/] Change the owner from root to Jenkins.
- Now create new item as "FusicPlayer_test"in Jenkins with following details:



☐ Quiet period
☐ Retry Count
☐ Block build when upstream project is building
☐ Block build when downstream project is building
☑ Use custom workspace
Directory: /var/jenkins_home/workspace/FusicPlayer_build
Display Name:
☐ Keep the build logs of dependencies

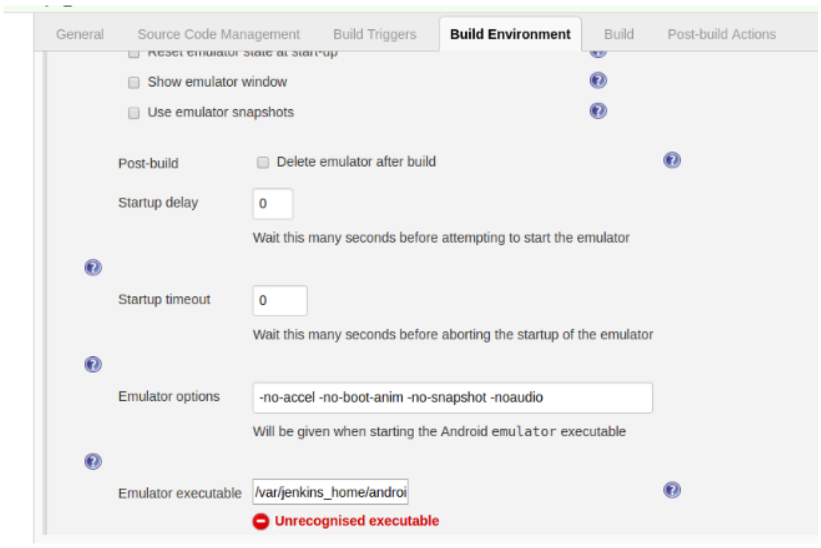
Figure 7(a). Test Configuration Value



☐ With Ant
☑ Run an Android emulator during build
● Run existing emulator
AVD name: Android19
Enter the name of an existing Android emulator configuration
○ Run emulator with properties

Common emulator options
☐ Reset emulator state at start-up
☐ Show emulator window
☐ Use emulator snapshots

Figure 7(b). Test Configuration Value


General | Source Code Management | Build Triggers | **Build Environment** | Build | Post-build Actions

☐ Reset emulator state at start-up
☐ Show emulator window
☐ Use emulator snapshots

Post-build: ☐ Delete emulator after build

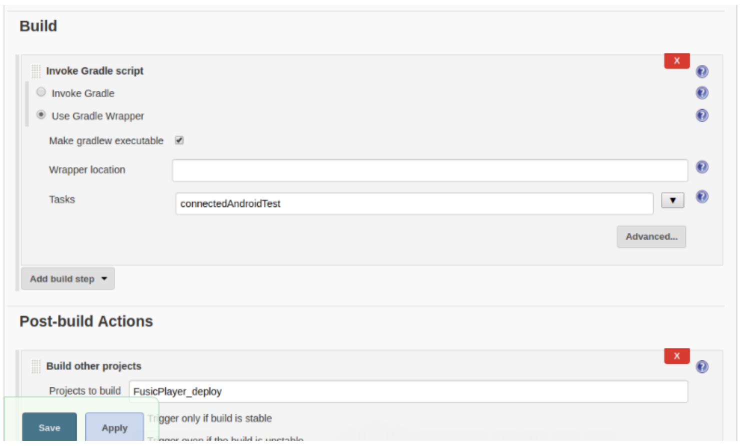
Startup delay: 0
Wait this many seconds before attempting to start the emulator

Startup timeout: 0
Wait this many seconds before aborting the startup of the emulator

Emulator options: -no-accel -no-boot-anim -no-snapshot -noaudio
Will be given when starting the Android `emulator` executable

Emulator executable: /var/jenkins_home/androi
Unrecognised executable

Figure 7(c). Test Configuration Value [/var/jenkins_home/android-sdk/emulator]


Build

Invoke Gradle script
○ Invoke Gradle
● Use Gradle Wrapper
Make gradlew executable ☑
Wrapper location:
Tasks: connectedAndroidTest
Advanced...

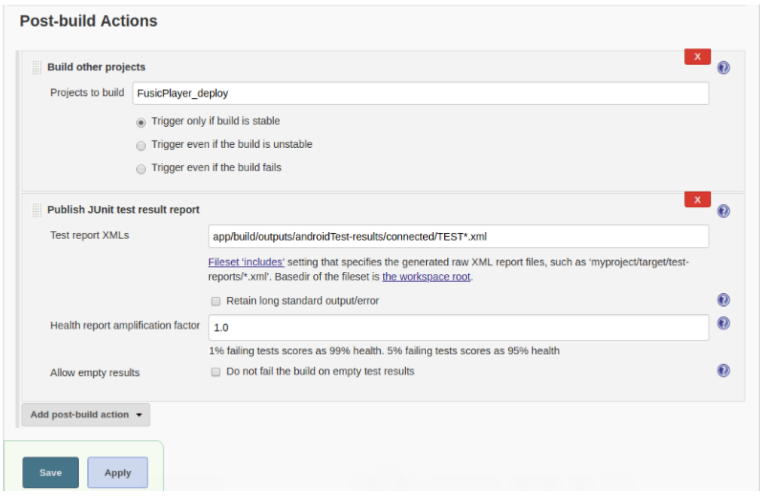
Add build step

Post-build Actions

Build other projects
Projects to build: FusicPlayer_deploy
Trigger only if build is stable
Trigger even if the build is unstable

Save | Apply

Figure 7(d). Test Configuration Value


Post-build Actions

Build other projects
Projects to build: FusicPlayer_deploy
● Trigger only if build is stable
○ Trigger even if the build is unstable
○ Trigger even if the build fails

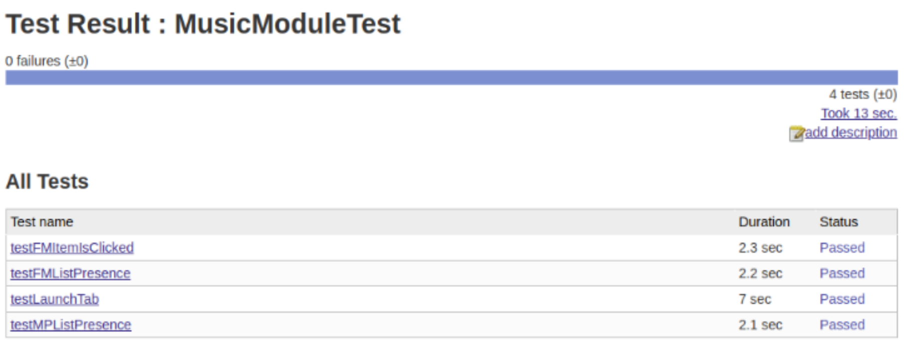
Publish JUnit test result report
Test report XMLs: app/build/outputs/androidTest-results/connected/TEST*.xml
Fileset 'includes' setting that specifies the generated raw XML report files, such as 'myproject/target/test-reports/*.xml'. Basedir of the fileset is the workspace root.
☐ Retain long standard output/error
Health report amplification factor: 1.0
1% failing tests scores as 99% health. 5% failing tests scores as 95% health
Allow empty results: ☐ Do not fail the build on empty test results

Add post-build action

Save | Apply

Figure 7(e). Test Configuration Value

The XML reports will be saved in above location in workspace. These can be used to show the result in Jenkins.



Test Result : MusicModuleTest

0 failures (±0)

4 tests (±0)
Took 13 sec.
add description

All Tests

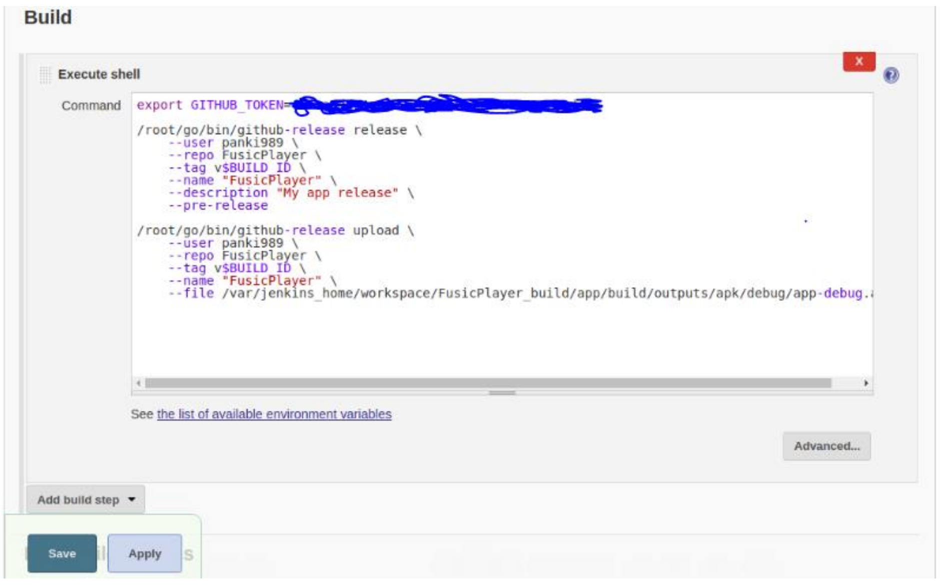
Test name	Duration	Status
testFMItemIsClicked	2.3 sec	Passed
testFMListPresence	2.2 sec	Passed
testLaunchTab	7 sec	Passed
testMPListPresence	2.1 sec	Passed

Figure 8. Integration Testing Results

3.6 Deploy

We use the **Go script** to upload the *.apk file to GitHub release. Also, Go binaries need to be installed in the Jenkins Docker. Following link is for the scripts. Following are the steps:

- Create new item FusicPlayer_deploy.
- Create GitHub token.
- Configure script according to your directory structure.



Build

Execute shell

Command

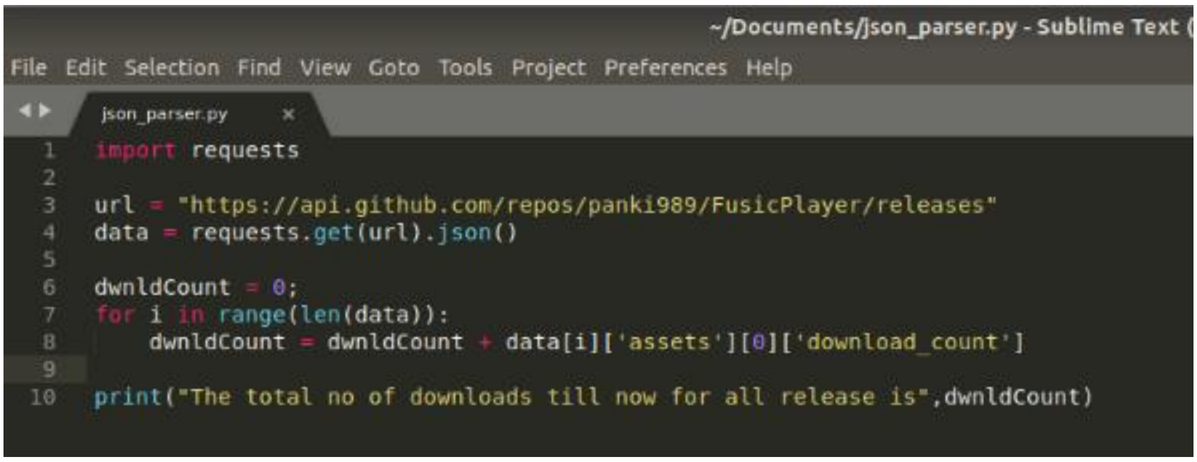
```
export GITHUB_TOKEN=[redacted]

/root/go/bin/github-release release \
    --user panki989 \
    --repo FusicPlayer \
    --tag v$BUILD_ID \
    --name "FusicPlayer" \
    --description "My app release" \
    --pre-release

/root/go/bin/github-release upload \
    --user panki989 \
    --repo FusicPlayer \
    --tag v$BUILD_ID \
    --name "FusicPlayer" \
    --file /var/jenkins_home/workspace/FusicPlayer_build/app/build/outputs/apk/debug/app-debug.
```

See the list of available environment variables

Figure 9. Deploy Script

3.7 Monitor

We can get the total no of downloads by using GitHub API which returns the data in .json format. As of now, we are not putting this in pipeline but we can always trigger this from Jenkins & get the total no of downloads for our release. We can get the count by below simple **python script**.

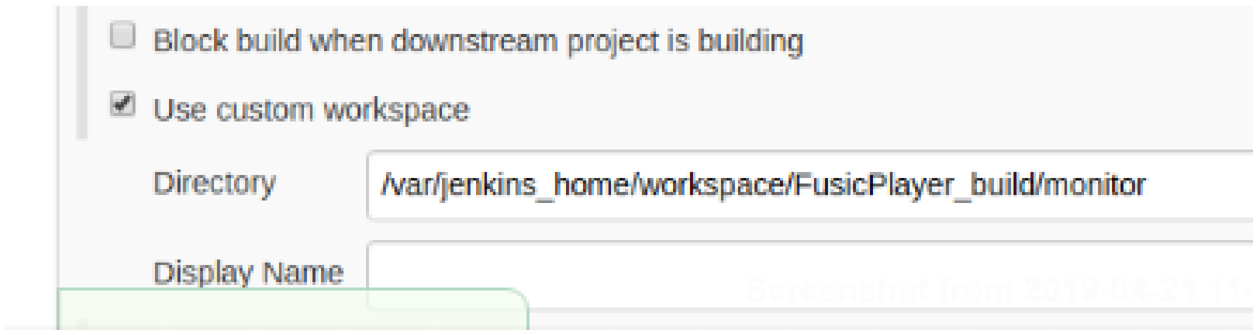

~/Documents/json_parser.py - Sublime Text (
File Edit Selection Find View Goto Tools Project Preferences Help
json_parser.py

```python
import requests

url = "https://api.github.com/repos/panki989/FusicPlayer/releases"
data = requests.get(url).json()

dwnldCount = 0;
for i in range(len(data)):
    dwnldCount = dwnldCount + data[i]['assets'][0]['download_count']

print("The total no of downloads till now for all release is",dwnldCount)
```

Figure 10. Monitoring Total no of Downloads

Following steps to create new item in Jenkins:

- Create new item as FusicPlayer_monitor.
- requests python library must be installed in Docker.
- Json_parser.py must be saved in Docker & its directory path must be given in workspace in Jenkins.
- Configure it as shown:



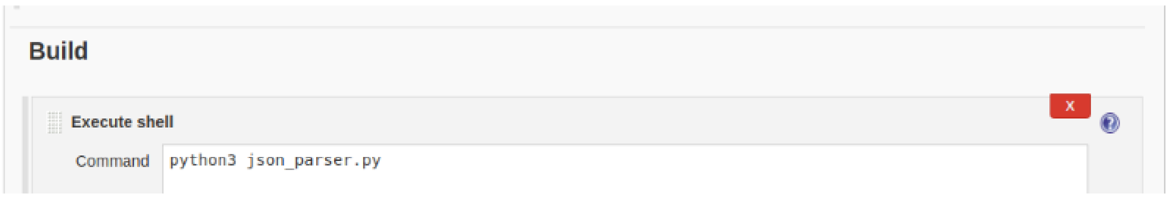
☐ Block build when downstream project is building

☑ Use custom workspace

Directory: /var/jenkins_home/workspace/FusicPlayer_build/monitor

Display Name:

Figure 12(a). Monitor Configuration



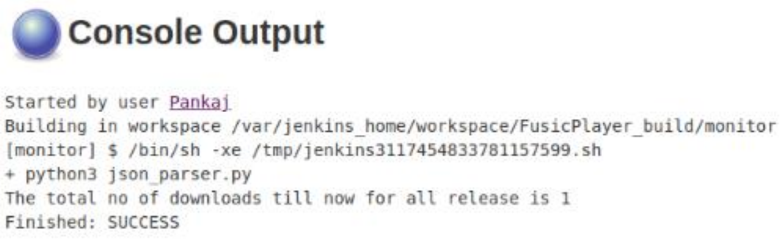
Build

Execute shell

Command: `python3 json_parser.py`

Figure 12(b). Monitor Configuration

The results will be displayed as follows:



Console Output

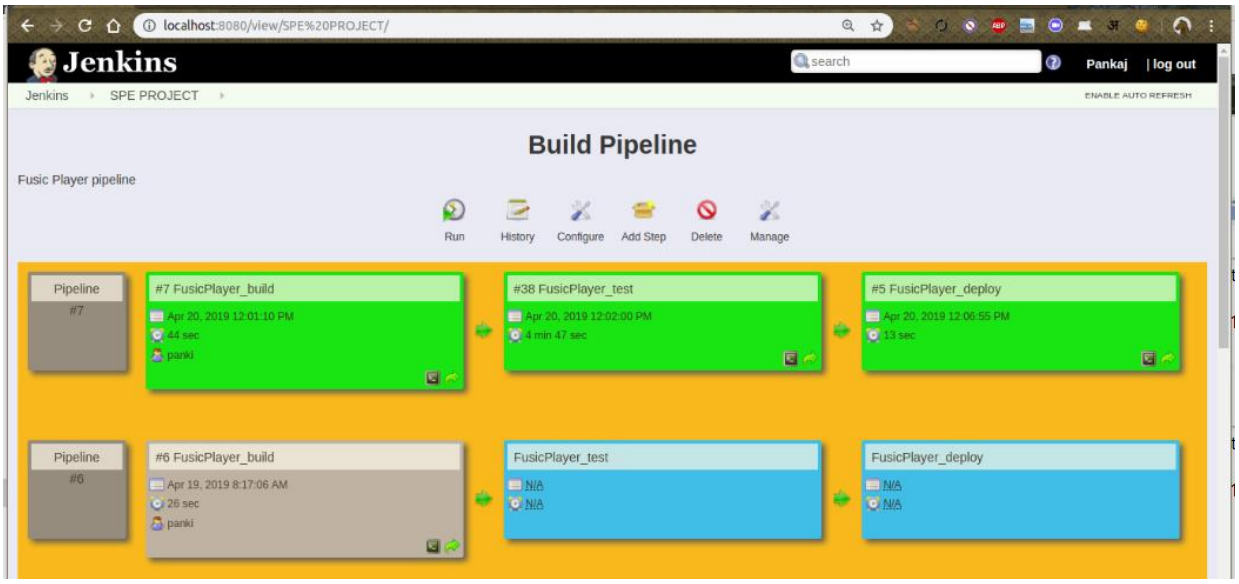
```
Started by user Pankaj
Building in workspace /var/jenkins_home/workspace/FusicPlayer_build/monitor
[monitor] $ /bin/sh -xe /tmp/jenkins3117454833781157599.sh
+ python3 json_parser.py
The total no of downloads till now for all release is 1
Finished: SUCCESS
```

Figure 13. Monitoring Downloads

4 Results

Now, we can create the pipeline for the same to process each job one after another. In post-build actions, we can allow to start the next job to create pipeline. As mentioned earlier, Download counts can be monitored by manual job running in Jenkins. The output will be displayed in console.

The total amount of time taken for all build, test & deploy with Dev-Ops for this is near about **7 min**. But setting up the environment might be time consuming.

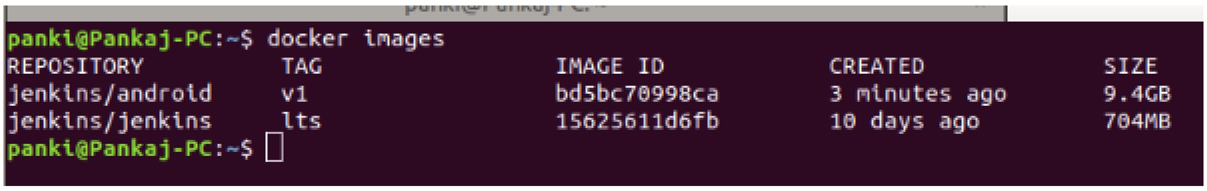


Pipeline	Build	Test	Deploy
Pipeline #7	#7 FusicPlayer_build Apr 20, 2019 12:01:10 PM 44 sec panki	#38 FusicPlayer_test Apr 20, 2019 12:02:00 PM 4 min 47 sec	#5 FusicPlayer_deploy Apr 20, 2019 12:06:55 PM 13 sec
Pipeline #6	#6 FusicPlayer_build Apr 19, 2019 8:17:06 AM 26 sec panki	FusicPlayer_test N/A N/A	FusicPlayer_deploy N/A N/A

Figure 14. Pipeline

Finally, everything that is done in Docker is saved using following command:

“docker commit -a "pankaj pankaj.075@iiitb.org" JenkinsAndroid jenkins/android:v1”



```
panki@Pankaj-PC:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
jenkins/android     v1                  bd5bc70998ca        3 minutes ago       9.4GB
jenkins/jenkins     lts                 15625611d6fb        10 days ago         704MB
panki@Pankaj-PC:~$
```

Figure 15. Docker Images

If anyone needs this Docker image, please mail me as due to its huge size, we're not going to push it on dockerHub.

4.1 Scalability

The app is highly scalable as:

- Developed using OOPS concept
- Separate package for Radio and Music modules
- Separate module for the ExoPlayer
- In each package, Utilities and Functionality are in separate class.
- Proper getters and setters for each variable is given.

5 Future Work

We can work on following points for this project:

- Automating the toggle button based on RJ voice.
- More features can be added in music player.
- Real FM module access. (Till now not allowed due to security reason).

6 Conclusion

Overall, It's a great experience to learn. Dev-Ops truly works as a magic in this. Initially, it took some time to setup but once done, it took few minutes to do everything we've ever imagined.

END OF REPORT

7 References

- https://hub.docker.com/r/jenkins/jenkins
- https://github.com/google/ExoPlayer
- https://github.com/aktau/github-release
- https://github.com/panki989/FusicPlayer
- https://medium.com/temy/android-ci-with-jenkins-docker-and-kitematic-step-by-step-b40ddbcaf5ff
- https://hub.docker.com/r/bertrandmartel/docker-jenkins-android
- https://onlineradiobox.com/in/redfm935/?cs=in.redfm935
- YouTube