

AUTOMATED CONTINUOUS INTEGRATION USING CIRCLECI AND FIREBASE FOR ANDROID APPLICATION DEVELOPMENT

NGUYỄN THỊ PHƯƠNG GIANG¹, TRẦN THỊ MINH KHOA²

*Trường Đại học Công nghiệp Thành phố Hồ Chí Minh
nguyenthiphuonggiang@iuh.edu.vn; ttmkhoa@iuh.edu.vn*

Abstract—Continuous Integration (CI) is the most common practice among software developers where they integrate their work into a frequent baseline. The industry 4.0 is facing huge challenges while developing Software at multiple sites and tested on multiple platforms. Today, so many CI tools widely used for software development as CircleCI, Jenkins, Travis. CircleCI is one of the CI tools that can help in automating the complete process, reducing the works of a developer and check the development at each and every step of Software evolution. In this paper, we discuss the implementation of CircleCI for android application development. Firebase Test Lab will be used for some additional automation testing.

Keywords—*Android; Automated Continuous Integration; CircleCI; CircleCI and Firebase*

1. INTRODUCTION

Software Development in big ventures requires collaboration of multiple groups in different countries. This intercontinental development may have multiple teams working separately on different components or they may collaborate together to work on overlapped components. Such a complex architecture requires to be managed by a system/tool which can record the changes and revert back to previous whenever required i.e. Version Control System (VCS). VCS can be maintained using a local, central or a distributed architecture. With this Distributed VCS (DVCS), different teams can easily collaborate in multiple ways with in same project [2]. Git is a free open source tool used as DVCS and Source Code Management System. Git allows non-linear versioning through its branching and merging. Each time a commit is triggered, Git stores a set of snapshot of files at that particular instance and stores a reference to that file. For big complex projects, having multiple components, it gets difficult to manage multiple Gits for each component. So, requirement of a Repository Management Tool comes into picture i.e. Repo. Repo is maintained above Git, it combines multiple Git repositories as one Git and maintains a hidden repo directory containing all projects names and paths in an xml file. Once a developer makes changes into the code or generates a new source code, these changes need to be integrated into repository for other developers or for users. Then this new code repository is built again to get executable as per the project requirement.

Once the system is built successfully, then it needs to be tested against each scenario. If all test cases pass then the integrator proceeds to release the update to customer else notifies concerned teams for errors. Integration involves building, testing and validating processes as discussed above. These tasks consume a large amount of employee work hours. Building and testing frequently becomes very tedious as test cases fail more often. So these processes must be automated allowing integrator to build overnight automatically. All these steps can be automated with the help of continuous integration tools (CI). The best way to make CI faster and more efficient is to automate the build and testing process [1].

In recent years CI has become a best practice for software development and is guided by a set of key principles. Among them are revision controls, build automation and automated testing. Additionally, Continuous Deployment and Continuous Delivery have developed as best-practices for keeping your application deployable at any point or even pushing your main codebase automatically into production whenever new changes are brought into it. This allows your team to move fast while keeping high quality standards that can be checked automatically [2,3].

In this study, we used CircleCI for android application development. Firebase Test Lab will be used for some additional automation testing.

CircleCI is a cloud-based system—no dedicated server required, and you do not need to administrate it. CircleCI releases saved artifacts for testing. The result of the build is going to be an artifact or the group of artifacts. Artifacts could be a compiled application or executable files (e.g. android APK) or metadata (e.g. information about the tests success) [4].

Firebase Test Lab for Android is a Google Cloud Platform Services provides cloud-based infrastructure for testing Android apps. With one operation, you can initiate testing of your app across a wide variety of devices and device configurations. Test results—including logs, videos, and screenshots—are made available in your project in the Firebase console. Even if you haven't written any test code for your app, Test Lab can exercise your app automatically, looking for crashes [5].

It is easier to create a CircleCI 2.0 script from scratch and iterate than to migrate from an existing CircleCI 1.0 script. As a prerequisite, It should be able to run this minimal CircleCI 2.0 Android build script for our project. [6] [7]

Continuous Integration (CI) is a widely established development practice in software development industry [11], in which members of a team integrate and merge development work (e.g., code) frequently, for example multiple times per day. CI enables software companies to have shorter and frequent release cycle, improve software quality, and increase their teams' productivity [11]. Due to the growing importance of continuous practices, an increasing amount of literature describing approaches, tools, practices, and challenges has been published through diverse venues. An evidence for this trend is the existence of five secondary studies on CI, rapid release, Contrinuous Delivery (CDE) and Contrinuous Deployment (CD) [12]. As a CI service, we will use CircleCI because they give you 1500 free build minutes per month and also it is possible to build projects that are hosted in the private repositories, but CircleCI only works with GitHub and Bitbucket. For running UI tests we will use Firebase Test Lab because they allow you to run up to 15 instrumentation tests per day. Dropbox to store the signing key and its credentials. Also, we will use Fastlane, because it simplifies some of the processes, like publishing the app on Google Play.

Historically organizations had to maintain a manual quality assurance (QA) teams represented the last security layer before any updates/changes could go live. It is worth to mention that several decades ago programming languages started to evolve exponentially and the broad variety of new and derivative languages appeared on the market. Depending on the application lan- guages differ from case-to-case, including database (DB) solutions (SQL vs. NoSQL), frameworks (for example, flask, Django, Node, and others), iOS, Android, Web front- end/back-end and automatization practices (with a custom syntax). [13]

2. CHALLENGES IN BUILD AND TESTING AUTOMATION

Though automation of build and testing tasks is essential for efficient software development process but, while automating these tasks one may face a number of challenges like:

- Software application being developed should be compatible with multiple platforms, different display units independent of the disk configuration in order to hit a larger market. For the development of an android application for example, requires be compatible with multiple android versions and different screen sizes. A web application on mobile phone /tablet should be as fully functional as on a desktop using any browser version.
- Developing such a multi-platform Software requires a large number of test cases for each of the configuration and devices.
- Additionally, the number of hardware devices connected to the test system also increases.
- For each new build, the Software has to be flashed on hardware. This task must be automated, which is difficult to achieve because it requires manually setting

- Up the connections. In this paper, we tried to overcome the challenges mentioned above by creating automation scripts for integration tasks which can be executed using CircleCI.

3. IMPLEMENTATION

Most of CI tools using the same architect that have a repository to store the project source code and CI tool will detect the change trigger on this repository. We define sub-tasks of us build task, tell CircleCI the dependencies between these tasks, and it will run all independent tasks in parallel and as soon as possible, as long as idle containers are available:

- When the new source code is committed, CI tools download the new source code and make a build. After testing process, the test result will be sent back to developer.
- CircleCI will ask for authentication of VCS like Github or Bitbucket at the first access. It automatic makes the first build when authentication is given.
- CircleCI allows user directly configure by project setting or via the circle.yml configuration file.

In this paper, we use build configuration by circle.yml configuration file and connect to Firebase Test Lab via project setting. Following is circle.yml file. The Fig. 1 showing CircleCI architect with linked with Google Firebase Test Lab.

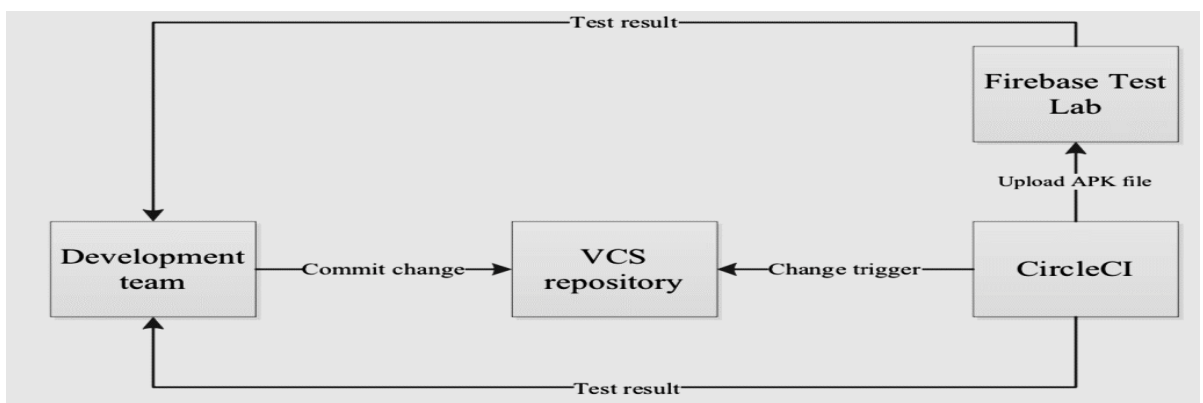


Fig. 1. CircleCI using Firebase Test Lab architect

Debug build only needs output APKs in later stages but build release, we need to maintain more files. For example, maps.txt generated is required when we upload the APK during the task check.

artifacts:

- /home/ubuntu/Test/app/build/outputs/apk/

We store the output APKs as CircleCI artifacts so we can download them from CircleCI when the task is completed. It works for other developers to test APKs without having to build or require development on their own. On the other hand, it is common to build APKs on a dedicated environment instead of machines developed on the system, which may be unstable and tend to change frequently.

```

# Build configuration for Circle CI
general:
  artifacts:
    - /home/ubuntu/Test/app/build/outputs/apk/
  machine:
    environment:
    
```

```

ANDROID_HOME: /usr/local/android --sdk --linux

dependencies:
  override:
    - chmod +x gradlew
    - echo y | android update sdk --no-ui --all --filter
tools, platform-tools, build-tools -25. 0. 2, android-25, extra --googlem2repository, extra
google --google_play_services, extra --android --support
    - echo y | $ANDROID_HOME/tools/bin/sdkmanager

"extras; m2repository; com; android; support;constraint;constraint-layoutsolver;1.0.2"
    - ANDROID_HOME=/usr/local/android-sdk-linux ./gradlew

dependencies test:
  override:
    - (./gradlew assemble):
        timeout: 370

```

Review last updates and push them up to the master branch of the repository. Now, head back to CircleCI and check out the Jobs tab. We see a passing build. It was able to take the settings from config.yml, set up the correct virtual environments, and run our tests just as we did locally when we first generated the project. CircleCI will automatic detect the circle.yml configuration file in the VCS repository and parsing it to project setting. Fig. 2 show build history for a test project.

By project	My builds	All builds
> project/FU1	<div style="background-color: green; color: white; padding: 2px;">SUCCESS</div> master #22 Daily commit 19 days ago 09:25 1ce8ec2	1.0
> project/FU2	<div style="background-color: green; color: white; padding: 2px;">SUCCESS</div> master #21 Merge remote-tracking branch 'origin/master' 19 days ago 09:09 8527520	1.0
	<div style="background-color: green; color: white; padding: 2px;">SUCCESS</div> master #20 Update circle.yml 19 days ago 08:33 f28ad40	1.0
	<div style="background-color: green; color: white; padding: 2px;">FIXED</div> master #19 Update circle.yml 19 days ago 08:01 4303429	1.0
	<div style="background-color: red; color: white; padding: 2px;">FAILED</div> master #18 Update circle.yml 19 days ago 03:49 4303429 <small>rebuild</small>	1.0
	<div style="background-color: red; color: white; padding: 2px;">FAILED</div> master #17 Update circle.yml 19 days ago 01:29 92f6bd6 <small>rebuild</small>	1.0

Fig. 2. CircleCI build history

CircleCI can link with Google Firebase Test Lab to using some Google Cloud platform services like Robo test, Instrument test...To link with Firebase, we need to install Gcloud console by setting project dependency and test command:

```

Pre-dependency commands:

    sudo pip install -U crcmod

Post-dependency commands:

./gradlew :app:assembleDebug --PdisablePreDex
    echo $GCPLOUD_SERVICE_KEY | base64 --decode --ignore-garbage > ${HOME}/gcloud-
servicekey.json
    sudo /opt/google-cloud-sdk/bin/gcloud config set project project-79e99
    sudo /opt/google-cloud-sdk/bin/gcloud --quiet components update
    sudo /opt/google-cloud-sdk/bin/gcloud --quiet components install beta
    sudo /opt/google-cloud-sdk/bin/gcloud auth activate-service-account project
-79e99@appspot.gserviceaccount.com --key-file ${HOME}/gcloud-service-key.json

Test command
echo "y" | sudo /opt/google-cloud-sdk/bin/gcloud beta test android run --app

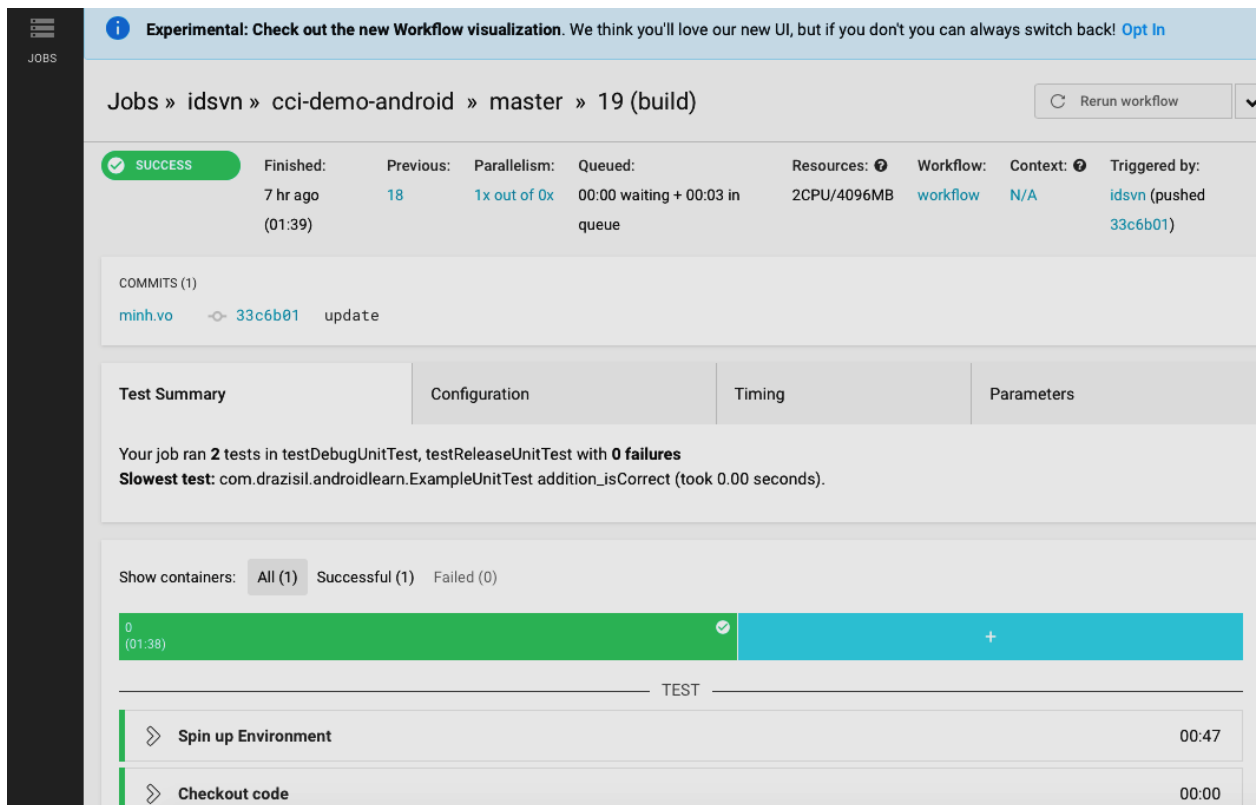
app/build/outputs/apk/app-debug.apk
    
```

And the test result is as follows:

```

Test results will be streamed to [https://console.firebase.google.com/project/project -
79e99/testlab/histories/bh.4ac642071abe70ee/matrices/5455937340372128030].
08:22:50 Test is Pending
08:23:15 Starting attempt 1
08:23:15 Test is Running
08:23:51 Logging into the device
08:23:51 Installing APK: android.com.project
08:24:28 Running Robo test. Package: android.com.project activity:
android.com.project.MainActivity
08:25:04 Robo test has finished
08:25:04 Generating video
08:25:04 Retrieving performance samples
08:25:29 Retrieving test artifacts
08:25:29 Retrieving any crash results
08:25:41 Retrieving logcat
08:26:05 Generating Robo ActivityMap
08:26:17 Done. Test time=36 (secs)
08:26:30 Test is Finished
Robo testing complete.
More details are available at [https://console.firebase.google.com/project/project-
79e99/testlab/histories/bh.4ac642071abe70ee/matrices/5455937340372128030].
    
```

OUTCOME	TEST_AXIS_VALUE	TEST_DETAILS
Passed	hammerhead-21-en-portrait	--



Experimental: Check out the new Workflow visualization. We think you'll love our new UI, but if you don't you can always switch back! [Opt In](#)

Jobs » idsvn » cci-demo-android » master » 19 (build) Rerun workflow

SUCCESS Finished: 7 hr ago (01:39) Previous: 18 Parallelism: 1x out of 0x Queued: 00:00 waiting + 00:03 in queue Resources: 2CPU/4096MB Workflow: workflow Context: N/A Triggered by: idsvn (pushed 33c6b01)

COMMITTS (1)
minh.vo 33c6b01 update

Test Summary Configuration Timing Parameters

Your job ran **2 tests** in testDebugUnitTest, testReleaseUnitTest with **0 failures**
Slowest test: com.drazisil.androidlearn.ExampleUnitTest addition_isCorrect (took 0.00 seconds).

Show containers: All (1) Successful (1) Failed (0)

0 (01:38) [Success] +

TEST

- Spin up Environment 00:47
- Checkout code 00:00

The one more detail

```

:app:generateReleaseSources
:app:javaPreCompileRelease
:app:compileReleaseJavaWithJavac
:app:lint
Download https://dl.google.com/dl/android/maven2/com/android/tools/lint/lint-gradle/26.2.1/lint-gradle-26.2.1.pom
Download
https://dl.google.com/dl/android/maven2/com/android/tools/external/org-jetbrains/uast/26.2.1/uast-26.2.1.pom
Download https://jcenter.bintray.com/org/codehaus/groovy/groovy-all/2.4.12/groovy-all-2.4.12.pom
Calling mockable JAR artifact transform to create file:
/home/circleci/.gradle/caches/transforms-1/files-1.1/android.jar/f87b27d40560caf2cc53dafb2067c3ec/android.jar with input
/opt/android/sdk/platforms/android-25/android.jar
Ran lint on variant debug: 5 issues found
Ran lint on variant release: 5 issues found
Wrote HTML report to
file:///home/circleci/androidlearn/app/build/reports/lint-results.html
Wrote XML report to
file:///home/circleci/androidlearn/app/build/reports/lint-results.xml
:app:generateDebugUnitTestSources
:app:preDebugUnitTestBuild UP-TO-DATE
:app:javaPreCompileDebugUnitTest
:app:compileDebugUnitTestJavaWithJavac
:app:processDebugJavaRes NO-SOURCE
:app:processDebugUnitTestJavaRes NO-SOURCE

```

```
:app:testDebugUnitTest
:app:generateReleaseUnitTestSources
:app:preReleaseUnitTestBuild
:app:javaPreCompileReleaseUnitTest
:app:compileReleaseUnitTestJavaWithJavac
:app:processReleaseJavaRes NO-SOURCE
:app:processReleaseUnitTestJavaRes NO-SOURCE
:app:testReleaseUnitTest
:app:test

BUILD SUCCESSFUL in 12s
39 actionable tasks: 39 executed
```

Other way, online available [8], [9] discussed about automated CI/CD process for a new idea. The goals of this process are version control, multiple levels of testing, and automated deployment. They'll use them Azure CLI implement as it is small and simple enough to be fairly easily understood, yet also a bit complex, as it does integrate with Microsoft Azure, a third-party cloud provider.

Contribution of the article presenting solutions to solve difficulties in software development in environments where applications can run on many different platforms (phones, computers ...), software developers have many different groups in different geographical locations. Furthermore, The article execute the solution by script code to automate to build, test and test the android application using the cloud-based system and the circleCIs teamed with Firebase to set up self-test customizations dynamic.

4. CONCLUSION

Integration is a vital component of software development in the Software industry. Automation of these integration tasks is of at-most importance as these tasks need to be executed continuously. CircleCI provides a better solution for CI automation. This paper presented the implementation of automation script for build and running compatibility test suite by android using CircleCI. The challenges faced during automation of these tasks have been discussed in detail in this paper. We considered a case study of build and testing for android environment. Also, Algorithms and process work-flow for automation for android environment have been discussed in this paper.

Although there has been an opinion on the implementation in another cloud environment, Azure CircleCI and no conclusive conclusions for this implementation. However, we recommend choosing any kind of for integration of CircleCI automated in the cloud to be convenient and useful.

5. FUTURE WORK

These automation scripts can be optimized in future, allowing merging and rebasing different patch branches using Git and test and then pushing to remote repository for deployment using CircleCI.

REFERENCES

- [1] M. Shahin, M. A. Babar and L. Zhu, "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices", IEEE Access, vol. 5, pp. 3909 – 3943, March 2017.
- [2] S. Li, H. Tsukiji and K. Takano, "Analysis of Software Developer Activity on a Distributed Version Control System", Advanced Information Networking and Applications Workshops (WAINA), 2016 30th International Conference, March 2016.

- [3] N. Seth, R. Khare, “ACI (automated Continuous Integration) using Jenkins: Key for successful embedded Software development”, 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS), Dec. 2015.
- [4] circleci.com, ‘Test Android Applications’, 2017. [Online]. Available: <https://circleci.com/docs/1.0/android/>. [Accessed: 27- May - 2017].
- [5] firebase.google.com, ‘Firebase Test Lab for Android Overview’, 2017. [Online]. Available: <https://firebase.google.com/docs/testlab/overview>. [Accessed: 27- May - 2017]
- [6] Alan Tai, ‘All you need to know about CircleCI 2.0 with Firebase Test Lab’[Online]. Available: <https://medium.com/@ayltai/all-you-need-to-know-about-circleci-2-0-with-firebase-test-lab-2a66785ff3c2>, Dec 5, 2017
- [7] Bruno Correia, ‘Android Continuous Integration using Fastlane and CircleCI 2.0 — Part II’ [Online]. Available: <https://medium.com/pink-room-club/android-continuous-integration-using-fastlane-and-circleci-2-0-part-ii-7f8dd7265659> , Mar 2, 2018
- [8] Rose Kaplan-Bomberg, ‘Creating automated build, test, and deploy workflows for orbs, part 2’[Online]. Available: <https://circleci.com/blog/how-to-output-junit-tests-through-circleci-2-0-for-expanded-insights/>, Mar 5, 2019
- [9] Rose Kaplan-Bomberg, ‘Creating automated build, test, and deploy workflows for orbs, part 1’[Online]. Available: <https://circleci.com/blog/how-to-output-junit-tests-through-circleci-2-0-for-expanded-insights/>, Feb 26, 2019
- [10] B. Fitzgerald, and K.-J. Stol, —Continuous Software Engineering: A Roadmap and Agenda, Journal of Systems and Software, vol. 123, 2017.
- [11] P. Rodríguez, A. Haghghatkah, L. E. Lwakatare, S. Teppola, T. Suomalainen, J. Eskeli, T. Karvonen, P. Kuvaja, J. M. Verner, and M. Oivo, —Continuous deployment of software intensive products and services: A systematic mapping study, Journal of Systems and Software, vol. 123, pp. 263-291, 2017.
- [12] Valery Ponomareko, Test and publish your apps with CircleCI + Fastlane + Firebase Test Lab
<https://proandroiddev.com/test-and-publish-your-apps-with-circleci-fastlane-firebase-test-lab-e716c075b99b> , Oct 22, 2018
- [13] Sergejs Bobrovskis, Aleksejs Jurenoks - A Survey of Continuous Integration, Continuous Delivery and Continuous Deployment, CEUR – WS, Vol – 2018 , pp. 314 – 322

TÍCH HỢP LIÊN TỤC TỰ ĐỘNG BẰNG CIRCLECI VÀ FIREBASE ĐỂ PHÁT TRIỂN ỨNG DỤNG ANDROID

Tóm tắt. Tích hợp liên tục (CI) là cách phổ biến để các nhà phát triển phần mềm đưa vào công việc thường xuyên của họ. Ngành công nghiệp 4.0 đang phải gặp thách thức lớn trong phát triển phần mềm ở nhiều trang web và được thử nghiệm trên nhiều nền tảng. Ngày nay, có rất nhiều công cụ tích hợp liên tục được sử dụng rộng rãi để phát triển phần mềm như CircleCI, Jenkins, Travis... CircleCI là một trong những công cụ tích hợp liên tục có thể tự động hóa quy trình hoàn chỉnh, giảm công việc của nhà phát triển và kiểm tra

sự phát triển của mỗi quá trình phát triển phần mềm. Trong bài báo này, chúng tôi thảo luận về việc triển khai CircleCI để phát triển ứng dụng Android. Firebase Test Lab sẽ được sử dụng cho một số thử nghiệm tự động hóa bổ sung.

Từ khóa. Tích hợp liên tục tự động, Android, CircleCI, CircleCI and Firebase

Ngày nhận bài: 9/10/2019

Ngày chấp nhận đăng: 10/07/2020