WHITE PAPER

WHY CONTINUOUS INTEGRATION SHOULD BE PART OF YOUR MOBILE DEVELOPMENT PROCESS

Mobile apps have taken center stage in the world of software development. This has pushed DevOps teams to consider new ways to ship apps faster and maintain the same commitment to quality. However, the way to build mobile apps faster is the same for mobile apps as it has always been for web apps – continuous integration.

This paper begins by highlighting the fragmented state of the mobile ecosystem that DevOps teams must grapple with. It suggests that the way to compete in today's mobile-first world is to adopt mobile continuous integration. It also discusses possible approaches, tactics, and tools available to DevOps teams as they consider mobile continuous integration.

SAUCELABS

TABLE OF CONTENTS

EXECUTIVE SUMMARY

DevOps teams have discovered that the explosive growth of mobile is a catalyst to speed up mobile development and drive better innovation. However, fragmentation in the mobile ecosystem has made this hard to achieve, and the issue is only worsening. Even as many organizations transition their web application development from waterfall to agile methodologies like continuous integration (CI) and continuous delivery (CD), they are faced with the elusive goal of further expanding CI to include their mobile app development as well. Despite the challenges, CI is still the answer to faster mobile app development and higher quality mobile apps.

The key to applying CI to mobile app development lies in automating the build and testing processes. The tools that enable automation have continuously evolved to keep pace with the rapid growth of mobile. Today, there is a wide range of tools that serve specific purposes along the entire CI pipeline.

This paper explores how organizations can approach their mobile app development from the perspective of continuous integration. It focuses on the two steps of automating build and test cycles, and features a variety of tools that help teams build the next generation of mobile apps.

SPEED AND QUALITY ARE AT THE CORE OF MOBILE APP DEVELOPMENT

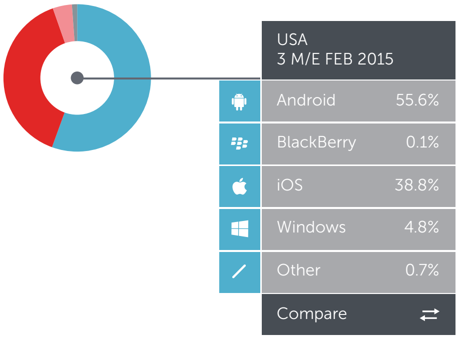
Mobile users demand innovation at much greater speed. Apps are installed, tried, and bought in seconds as opposed to hours or days for traditional apps. Apps that aren’t updated frequently risk becoming outdated and losing a competitive edge.

In addition, users have higher expectations from mobile apps in terms of personalization and user experience. Users demand that information be presented in context at the right time, in the right place, and in the right way. These demands, and the tremendous opportunity they afford, bring mobile app development to the forefront as businesses compete for leadership in today’s mobile-first world.

The need for faster releases and better applications were the two driving forces that caused organizations to transition from the traditional waterfall method of building software to agile methodologies like continuous integration (CI), and continuous delivery (CD). Today, as we make sense of a nascent mobile-first world, CI is even more relevant as organizations compete on the speed and quality of their mobile app development.

Platforms - iOS and Android dominate

iOS and Android dominate the mobile ecosystem with approximately 95% market share between them.



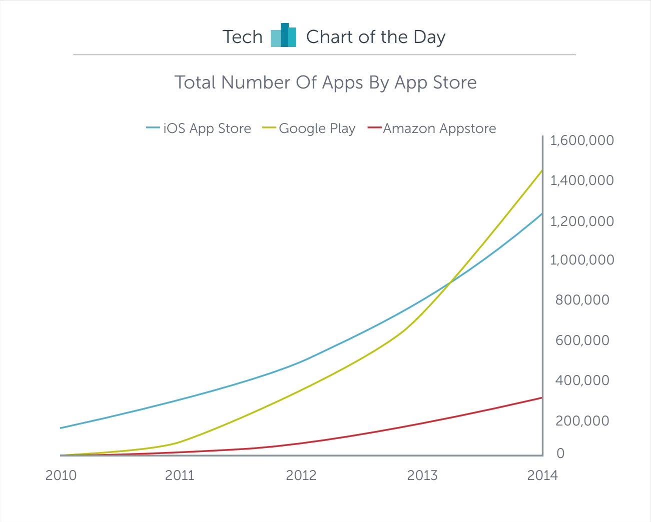
Source: Kantar Worldpanel

Operating System	2014 Unit Volumes*	2014 Market Share
Android	1,059.3	81.5%
iOS	192.7	14.8%
Windows Phone	34.9	2.7%
BlackBerry	5.8	0.4%
Others	7.7	0.4%
Total	1,300.4	100.0%

**Units in Millions* *Source: IDC, 2/24/2015*

Multiple sources attest to this fact:

The dominance of both these platforms has transformed the mobile ecosystem, bringing application development to the forefront. They each have over 1M unique apps listed in their app stores, and this number is still growing (by over 50% YOY in the case of Android).

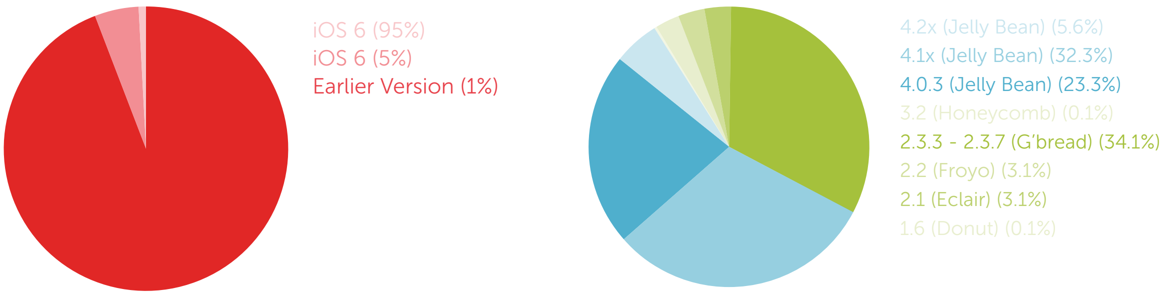


Source: Business Insider, 2/3/2015

While this may seem fairly consolidated at first, going a level deeper into the different operating system versions in use will reveal a high level of fragmentation, especially in the case of Android.

COMPARISON WITH IOS

Android fragmentation of all kinds is usually illustrated in comparison with iOS. These two pie charts clearly show the difference in API fragmentation between the two competing operating systems.



Source: OpenSignal, July 2013

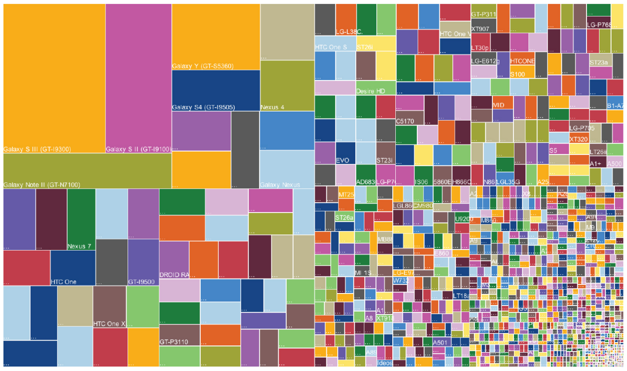
While iOS and Android are the undisputed leaders, there are a number of new mobile platforms that strive to challenge their dominance — Windows Phone, Firefox OS, Tizen, and Ubuntu to name a few. App developers need to be aware of the various operating systems and OS versions and decide which ones their apps should support. For most apps developed today, iOS and Android support are almost always a given, while support for other operating systems may vary.

Devices - A reflection of their platforms

There are only a few iOS device types, but they have gradually been increasing in number with pressure from competing Android devices. Apple has introduced cheaper and smaller versions of iPhone and iPad to compete with low-end Android devices.

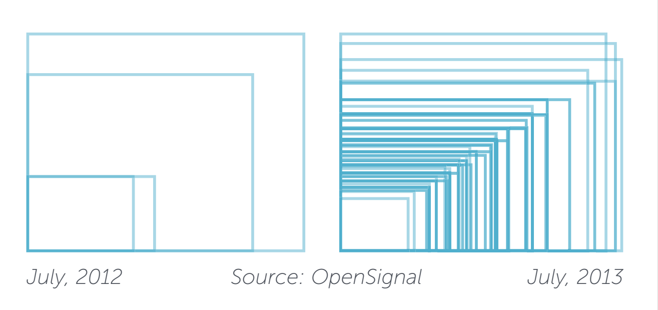
iOS vs Android screen sizes

The Android device ecosystem, on the other hand, is highly fragmented with devices ranging from phones, tablets, and watches, to TVs, cars, and video game consoles.



July, 2012 *Source: OpenSignal* *July, 2013*

This multitude of devices means their feature specifications, like screen size, are extremely varied, as shown by the following visualization:



Source: OpenSignal

On the reason for fragmentation within Android, OpenSignal comments, "Cheaper devices will struggle to run the most recent versions of Android." However, rather than seeing this as a problem, OpenSignal rightly suggests that "the fragmented operating system serves as an enabler of an ecosystem that is becoming more globally, and socio-economically, inclusive." Thus, fragmentation makes mobile app development more difficult, but it serves the greater good of increasing mobile adoption globally, and can't be ignored by mobile development shops.

With the dawn of the Internet of Things (IoT) and wearable devices, it is clear that device fragmentation is only in its beginning stages, and is bound to get more complex with time (See Appendix).

Apps - Blurring the lines between web and native

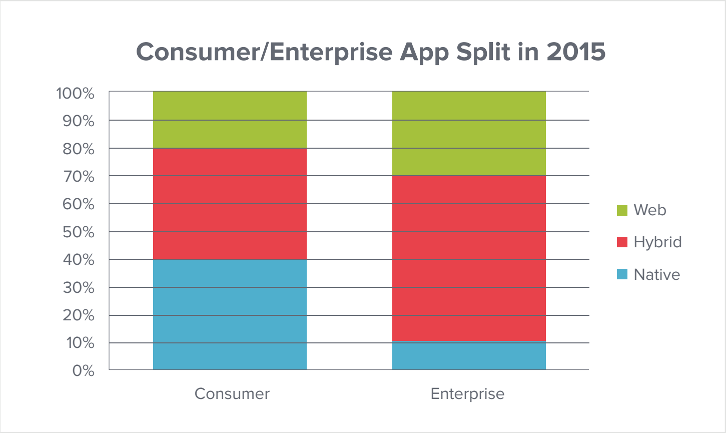
In addition to platforms and devices, there are multiple types of mobile apps as well. The two most popular app types are native and mobile web apps. Here is a comparison of these two types of apps:

	Mobile web app	Native mobile app
App store	Not necessary	Necessary
Mobile web browser	Necessary	Not necessary
Requires internet	Yes	No
Advanced functionality	No	Yes (leverages phone hardware)
User interface	Static	Interactive
Speed	Fast	Very fast
Development cost	Reasonable	Expensive
Approval process	None	Sometimes mandatory

A third type of app is a hybrid app. Telerik, maker of the popular KendoUI framework, says, "Hybrid apps are hosted inside a native application that utilizes a mobile platform's WebView. This enables them to access device capabilities such as the accelerometer, camera, contacts, and more." These apps are used by teams that want to avoid platform lock-in, and be able to utilize their developers' existing skills in web app development.

According to Gartner, developers of consumer apps tend to favor native apps, and developers of enterprise apps favor hybrid instead.

When creating a mobile app, developers need to decide what type of app is best suited for their business need. This can be a difficult decision when considering the fragmentation, lock-in issues with native apps, the lack of uniformity in HTML5, and a lack of consensus on how effective hybrid apps are.



Source: Gartner, April 2013

The fragmentation in the mobile ecosystem results in an extremely complex grid of operating systems, OS versions, devices, device capabilities, app types, app versions, browsers, and mobile networks. The challenge facing DevOps teams is the successful navigation of this maze, while out-maneuvering competition by releasing higher quality apps faster.

THE DEVOPS DILEMMA

Because of increased user expectations, organizations that compete in the mobile space end up with an aggressive roadmap, and their development teams are always trying to keep up. Mobile app development is slow and inefficient, often because it still follows the waterfall method of development. User experience is bolted onto a somewhat functional product, which becomes evident when users start to use the app.

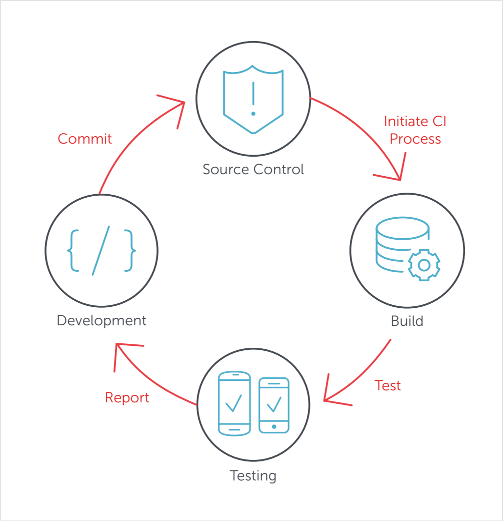
Additionally, mobile app testing can be difficult to scale and often becomes an afterthought. Even if testing on a single platform is easy, cross-platform testing is a challenge because of limitations with processes and tools. Functional tests are especially crucial to mobile apps, but are hard to scale on an unmanageable testing grid. Because of this added complexity with mobile, more things can and do go wrong — operating system, software, and hardware bugs, hardware limitations or issues — all of these can plague user experience.

Not wanting to compromise on the quality of mobile app testing, some organizations set out to build a device lab with every possible platform, and device type. However, they soon end up with a maze of devices connected to desktops that are difficult maintain, don't produce the desired results, and are very expensive. Considering the limitations of traditional approaches, continuous integration (CI) is the key to adopting a DevOps approach to mobile app development.

EXTENDING CI TO INCLUDE MOBILE APP DEVELOPMENT

Continuous integration, as understood from web application development, involves two overarching steps:

- Automating builds so they are more frequent
- Automating tests to enable faster feedback



In the case of web applications, both these steps have resulted in better apps being shipped faster. Considering these benefits, CI should be naturally extended to include mobile app development and testing as well. However, for that to happen, we need to be clear about what continuous integration for mobile would look like in practice.

It starts with build automation

The first step in CI is to automate builds. According to Techopedia, "The term build may refer to the process by which source code is converted into a stand-alone form that can be run on a computer or to the form itself." Once a developer commits source code, it is stored in a software configuration management (SCM) system that maintains version control for the source code. From here it is automatically converted to executable code.

Builds and commits need to be frequent to enable faster feedback. The best way to increase frequency is to automate builds. Typically, the build process is automatically triggered by the CI server after every commit to the SCM. The CI server then executes a build script, which integrates the software. During this entire process, teams use multiple tools that perform specific functions and work well with each other.

Tools that enable mobile build automation

Build tools: Tools like Make, Ant, Maven, and Gradle help to automate the build process. Gradle is the newest of the lot, and has been adopted by Google as the default build tool for Android.

CI server: Jenkins has become the most popular CI server because of its ease of use and extensibility. Jenkins has deep support for mobile app development via plugins. Examples include plugins for Git, Gradle, Xcode, Android Emulator, and Android Lint. These plugins enable developers to automatically install SDKs and packages, spin up emulators and simulators, and much more.

Software configuration management (SCM): Subversion (SVN) was the original standard for SCM, but Git has replaced it as the preferred version control system for most developers as it decentralizes version control. This is especially well suited for projects that involve multiple developers who need version control even when they are on the go and may not be connected to the repository.

Platform-specific tools: XCode is the development kit provided by Apple for developing apps for iOS. It includes an IDE, a compiler, the most recent SDKs, a simulator, and other tools to enable development with iOS. Similarly, Android Studio, provided by Google, includes an IDE, the Android SDK, and an emulator for developing Android apps. They are indispensable when building native apps for these platforms.

There are many tools to choose from, and developers have no shortage of options when deciding which tools to use to automate the build process. With this momentum around build automation, there is no reason why developers should stick to traditional methods of building apps. The first step on the way to mobile continuous integration is build automation, and the landscape of modern development tools make this step possible, and inevitable.

Extending CI to include mobile app testing

After automating the build process, the next step is to automate mobile app testing. As defined by Techtarget, "Automated software testing is a process in which software tools execute pre-scripted tests on a software application before it is released into production."

In this step, test scripts are written and executed with an aim to improve the quality of the software before its release. In mobile testing, these scripts inspect and deploy compiled binaries to an emulator, simulator, or a physical mobile device. The two main types of tests in mobile apps are unit testing and functional testing.

In unit testing, the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Just as in web apps, this is the most basic type of testing for mobile apps, and a prime candidate for automation.

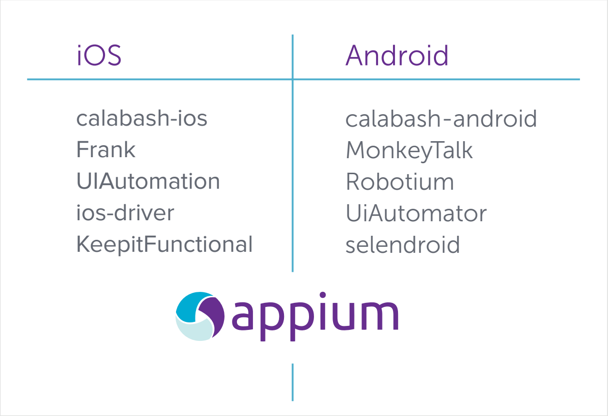
Functional testing, on the other hand, is a way of checking the functionality of an app to ensure it works as specified in the requirements. Given the importance of user experience in mobile, and the personalized experience that users expect, functional testing is even more important for mobile apps than for web apps.

Though the number of available testing tools for mobile are fewer than those for build automation, the ecosystem has been growing in recent years. Automated testing for mobile helps avoid errors and omissions caused by manual testing, and greatly reduces the time it takes to ship an app. Because of this, it is important to carefully consider available tools for mobile test automation.

Tools that enable mobile testing automation

Unit testing: XCTest is the unit testing framework for iOS, which replaces the older OCUnit framework. For Android apps, jUnit is a built-in open source framework that automates unit testing.

Functional testing: Tools for functional testing in mobile apps fall either in the iOS or Android camp. Here's a list of the most popular functional testing tools for iOS and Android:



iOS	Android
calabash-ios	calabash-android
Frank	MonkeyTalk
UIAutomation	Robotium
ios-driver	UiAutomator
KeepitFunctional	selendroid

Most existing solutions support either iOS or Android. Even if they do support both platforms, they don't use the same API. Appium is the only automated testing tool that supports both iOS and Android equally well. Let's take a closer look at Appium.

APPIUM - THE LEADING CROSS-PLATFORM TEST AUTOMATION FRAMEWORK

Appium is the mobile counterpart of the leading automated testing framework for web apps — Selenium. Based on the WebDriver API that powers Selenium, Appium is well on its way to becoming an industry standard in mobile app testing.

Appium is built on 4 core philosophies:

1. Test the same app you submit to the app store
   Some test frameworks require you to recompile your app to automate it. However, Appium uses vendor-provided automation frameworks. This means you won't need to compile any Appium-specific or third-party code or frameworks to your app — you're testing the same app you're shipping. This way you won't miss any bugs.

2. Write your tests in any language, using any framework
   With Appium you can write automated tests in a programming language different from the application code. This means you can write programs in Java for Android and Objective C for iOS. In fact, you can write automated test scripts in just about any modern language.

3. Use a standard automation specification and API
   To extend your existing test automation to mobile, it makes sense to start with something you're already familiar with — Selenium's WebDriver API. As a model, WebDriver is a mature and good place to start for mobile automation. In fact, Appium is soon to become a core feature of Selenium for testing mobile apps.

4. Have a large and thriving open-source community effort
   Lots of users and contributors help make the tool better for everyone. With the community making decisions on the development of Appium, vendor lock-in isn't a concern.

With web browsers you can write one Selenium test that runs on all the different Web browsers, so you can uncover cross-platform issues, but so far this hasn't been possible or easy with mobile devices. Appium opens the door to true cross-platform mobile testing. It enables you to write one test in a programming language of your choice and to run the test across both iOS and Android. Therefore, any organization evaluating the ideal test automation solution for mobile apps should take a close look at Appium.

CONCLUSION

The fragmentation of the mobile ecosystem can cause DevOps to shy away from adopting CI for their mobile software development life cycle. However, because of the complexity of mobile app development, and high stakes for the companies involved, CI is even more essential for mobile app development than web apps. Contrary to what some may believe, mobile CI is not completely different from CI as we've always known it. The two most important goals of mobile CI are still faster releases and improved code quality. These goals can be met by automating the two vital phases of continuous integration — build, and testing.

The necessary tools for mobile continuous integration have been evolving at a slower pace than the growth of the mobile economy, but today, there are many capable tools that meet specific needs at every step of the CI process. Be it a build tool like Gradle, or a test automation tool like Appium, mobile developers who are serious about adopting CI for mobile can be well-equipped for the task. By seeking to extend CI to also include mobile development, teams can make a smoother transition to the DevOps culture, and give their organizations an edge in the fierce mobile battles that have just begun.

ABOUT SAUCE LABS

Sauce Labs ensures the world's leading apps and websites work flawlessly on every browser, OS and device. Its award-winning Continuous Testing Cloud provides development and quality teams with instant access to the test coverage, scalability, and analytics they need to deliver a flawless digital experience. Sauce Labs is a privately held company funded by Toba Capital, Salesforce Ventures, Centerview Capital Technology, IVP and Adams Street Partners. For more information, please visit saucelabs.com.

SAUCELABS

SAUCE LABS INC. - HQ
116 NEW MONTGOMERY STREET, 3RD FL
SAN FRANCISCO, CA 94105 USA

SAUCE LABS EUROPE GMBH
NEUENDORFSTR. 18B
16761 HENNIGSDORF GERMANY

SAUCE LABS INC. - CANADA
134 ABBOTT ST #501
VANCOUVER, BC V6B 2K4 CANADA