



The Ultimate Guide To a Successful Continuous Delivery Pipeline

Published by Mobile Labs Inc.



To learn more about Mobile Labs visit
mobilelabsinc.com

Click here for our other book on Amazon:
[Enterprise Mobile App Development & Testing](#)

Published by Mobile Labs Inc.
Copyright 2017 Mobile Labs Inc.

Thank you for downloading this ebook. This book remains the copyrighted property of the author, and may not be redistributed to others for commercial or non-commercial purposes.

Contents

Introduction.....	4
Manage Mobile Devices.....	5
3 Benefits of Using a Mobile Device Cloud.....	6
Test on Real Devices for Real Results.....	10
Set Up an Automation Strategy.....	13
How to Build a Continuous Delivery Pipeline.....	20
Important Sources for Feedback.....	24



Introduction

In this eBook, we will explore the 5 keys to building a successful continuous delivery pipeline. Our discussion will highlight the importance of using a mobile device cloud, testing on real devices, and explain how to set up a test automation strategy. We will also walk you through the steps to set up a continuous delivery pipeline and show you how to get effective feedback for your apps and mobile websites.

Manage Mobile Devices



Take a moment and picture the mobile devices in your mobile testing lab. Where are they stored? Are they kept in a locked drawer? Do they reside on a table, clearly labeled and waiting for action? Or, maybe they are in a haphazard pile that you must climb out of each day to test and to debug?

Even if your enterprise mobility team believes that they have found a solution that works for device sharing, consider what happens with increased demand. With more pressure to develop and test better and faster, teams must be flexible and agile to rise to these new challenges in developing, testing and releasing new updates to mission-critical apps to stay relevant and viable.

Having all your devices neatly spread out on a table, or in a central drawer is doable if your developers, testers and QA are working together in one office. But, what about remote employees? What about developers, testers and QA in other office buildings, or even in other countries? How can everyone on the team share the same devices for testing, but still work quickly and efficiently?

3 Benefits of Using a Mobile Device Cloud

At Mobile Labs, we advocate testing on real devices. Sometimes in life, there is not a good replacement for the “real thing.”

Certainly, simulators and emulators have their place in the development and testing process, but to expedite testing and debugging, why not work directly with the devices that will be employed by users to engage with your apps? One solution that can help enterprise mobility teams to manage devices for testing is a mobile device cloud. Here are 3 reasons why you should consider setting one up in your mobile testing lab. Setting up a mobile device cloud is the first step in creating a powerful continuous delivery pipeline.



#1 Resuscitate Engineering Practices To Boost Agility And Devops

Taken by itself, a robust continuous delivery strategy helps enterprise testing teams with agility and improved quality of apps.

With better apps comes increased customer satisfaction, whether your app is for external customers or used internally to increase employee productivity on the job.

Setting up a device cloud for mobile development and testing also provides these same benefits by boosting DevOps processes. By setting up a device cloud, whether on-premises or hosted in a secure data center, developers, testers

and QA can benefit from one central portal where devices are shared among all members of the respective teams.

By working with the same devices, development, testing and debugging are faster and more efficient. When paired with continuous integration, all teams can stay on top of fixing any issues with apps that may come up to keep new releases of apps on time and under budget.



#2 Increase Flexibility For Device Sharing



At Mobile Labs, one of our favorite questions to ask attendees at trade shows is how are they currently managing device sharing.

We have heard a diverse set of responses over the years, but most testing teams are either storing these devices in one central location such as a table, or under lock and key in a drawer. In the drawer scenario, typically a QA Manager must oversee the process of “checking out” devices to various team members. Imagine the time saved and the boost to productivity if there were an easier way to manage devices.

By connecting all devices in a device cloud, developers, testers and QA can quickly log

in to a central portal to test and run various scripts on the device they need. Without a device cloud setup, developers and testers may have to wait on each other for a certain device to become available, or they may have to physically get up to check out a physical device from a general location in the office. Naturally, this situation can cause even more delays if devices must be shared among locations in different cities, states, or countries. Just imagine the time delays (not to mention the astronomical costs) of shipping devices.

#3 Break Down Geographical Barriers

Because many development, testing and QA teams are not all housed together in one central office, a mobile device cloud can help break down these geographical barriers many development, testing and QA teams are not all housed together in one central office. Although some teams are just separated by floors or city borders, others may have whole oceans between them. With barriers of this nature, enterprise mobility teams are presented with a unique set of challenges regarding device sharing.

For teams with remote members, and for those spread out all over the world, shipping the devices to various team members is not only costly, but extremely time consuming if not virtually impossible. Certain countries have laws that make it hard for devices to pass through customs and cannot be shipped to other countries. In today's agile, dynamic world, developers, testers and QA cannot afford to have these delays while they are working together to develop, test and release updates to apps.

But, setting up a device cloud eliminates this issue of shipping devices over borders, be they country or city borders. With a device cloud set up, the devices themselves are housed in one central location, but everyone on the team can log in to the portal and access the device they need for development or testing 24/7 regardless of where they physically reside.

Housing the devices in a device cloud not only makes the process of sharing devices easier for the entire team, but it is a boost to both productivity and cost savings by eliminating the issues of shipping.

Test on Real Devices for Real Results



Now that you understand the importance of setting up a device cloud for mobile development, testing and QA, it's time to move on to the next step in launching a continuous delivery strategy.

As an enterprise mobility team, all team members face diverse challenges regardless of their role. Naturally, due to the fast-paced, mobile first environment that we are operating in, a large portion of the tasks for an enterprise mobility team revolves around the customer experience. Customers, whether external or internal users, are

the ones who live in the apps you create daily. They are the ones whose objectives, expectations and mobile experiences color the types of apps and mobile experiences that your team has been tasked to create. In other words, mobile users have high expectations and your team better deliver.



No matter how you are testing today, with real devices or using simulators and/or emulators, it is important to remember that your users are interacting with the real thing. Therefore, your team should test on real mobile devices to accurately mimic the environment that your users are seeing when they interact with your app. As a nod to agility, if a problem should occur, debugging on real devices can save the team time by catching the problem at its source and fixing it where it lives.

Perhaps Motown legends Marvin Gaye and Tammi Terrell said it best in 1968 when they sang that there, “Ain’t nothing like the real thing, baby.” Although the duo was singing about love, and mobile devices did not exist, we can apply this sentiment to testing on real mobile devices.

Since your users are engaging with real devices, it only makes sense to also develop and test on real devices. Real devices produce real results. Real results enable your team to debug and fix any

issues that arise on the real platform quickly and efficiently.

Although it is tempting to use the free simulators and emulators that come with iOS and Android development environments, remember that you get what you pay for.



Simulators and emulators run on a desktop being processed by a desktop processor and memory, while the devices run ARM compatible chips, which is a completely different architecture from a real device. Running on a real device allows you to see exactly the type of computing power that is required to run your app.

Do you have a process that consumes the CPU? You may get an approximate guess on a simulator, but how does it run on the latest device, or even the entry level device?

If it uses a lot of the CPU, chances are your app will also consume a lot of battery. The same goes for memory. Various real devices have different memory configurations. Much like the simulator, the memory is an approximate guess. It is not the same memory that is on a real device. A real device can give you the answers you need.

A mobile device cloud solution that allows you to view the memory, CPU and the power of real devices allows you to know for sure that you have a performing application.

Set Up an Automation Strategy



Now that we have discussed the importance of setting up a device cloud for mobile development, testing and QA, and the benefits of testing and debugging on real devices, it is time to move on to the next step in launching a continuous delivery strategy - setting up an automated testing strategy by building a suite of automated tests to overcome testing challenges.

When considering building a suite of automated tests, it is important to consider the types of tests that make sense for your various apps. Automated testing is a great fit for apps that continuously require updates and new releases to stay fresh and relevant. By setting up automated testing, your team can test quickly and efficiently to release on time to meet demand.

When test automation is done accurately and correctly, you can increase app quality while reducing app delivery time dramatically. This also holds true for regression testing to make sure that the software is still working properly.

#1 So Many Devices, So Many Operating Systems

Common Automation Challenges

Whether your enterprise mobile lab solely runs automated tests or a mix of manual and automated tests, here are 4 common challenges that you might run into when setting up your suite of automated tests for mobile.

It is commonly said that variety is the spice of life. Well, that's certainly true in mobility.

With so many different types of devices available today, from smartphones, to tablets, to smart speakers, and various wearables, it seems that new devices pop up daily. Along with this cornucopia of various mobile devices comes a slew of unique operating systems. Naturally, for a busy, resource-limited mobility team, manually testing mobile apps across each unique device and operating system can be time consuming.

Although many devices do run on the same OS platform, there is no guarantee,

for example, that an app created for the iPhone will behave the same way on an iPad. Sometimes a mobile app can have a different effect on devices in the same product family thanks to elements such as screen resolution, handset, memory level, battery, and manufacturer settings.

Automated testing in this scenario brings much needed agility to what could potentially be a time-consuming process for manual testing.

#2 Tools of the Trade – Open Source Tools vs. Commercial

One cannot begin automated testing without picking the right tools for the job.

Today's industry-standard tools typically fall into two camps – open source and commercial. With so many tools available, it is important to research and consider which tools work best for your team and that best suit its skill set.

Also, when choosing a device cloud solution, you should look for one that is tool-agnostic that can support today's industry standard testing tools for automated testing. A single automation tool may not work for

every application, therefore a device cloud should be able to support many different automation tools.

We also recommend looking for a vendor who provides a solution that is flexible and able to accommodate any new tools that may be introduced. This way your team is prepared for the future and can stay nimble.

But, ultimately, how do you know which tools to choose? What are the pros and cons?





Open source automation testing tools are becoming popular because they are developed to solve a particular problem. At the beginning of mobile app development, a lot of open source tools were created because there was a void of mobile support from the big automation vendors.

Today, there are many options to choose from in the open source community such as Appium, Selenium, Calabash, etc. One reason the open source tools are attractive is because they are free to implement. The most popular tool,

Appium builds on the most widely used open source automation tool for web testing, Selenium. This allows testers to utilize their existing automation framework design and skill sets for mobile. One thing to consider when using open source software is the lack of commercial support that you may get from a third-party vendor.

There are also commercial mobile automation tools available to purchase from vendors. These options usually give you a comprehensive product that fits into a larger testing suite of tools such

as performance and test management tools. They often have an easier implementation and can perform a wider set of testing than some of the open source tools. Some of the tools available are HP Unified Functional Testing, IBM Rational, and Tricentis Tosca.

Finding the right tool requires that it can perform the right tests and fit within your budget.

#3 High Volume of Automated Tests

Does automated testing make sense for all apps?

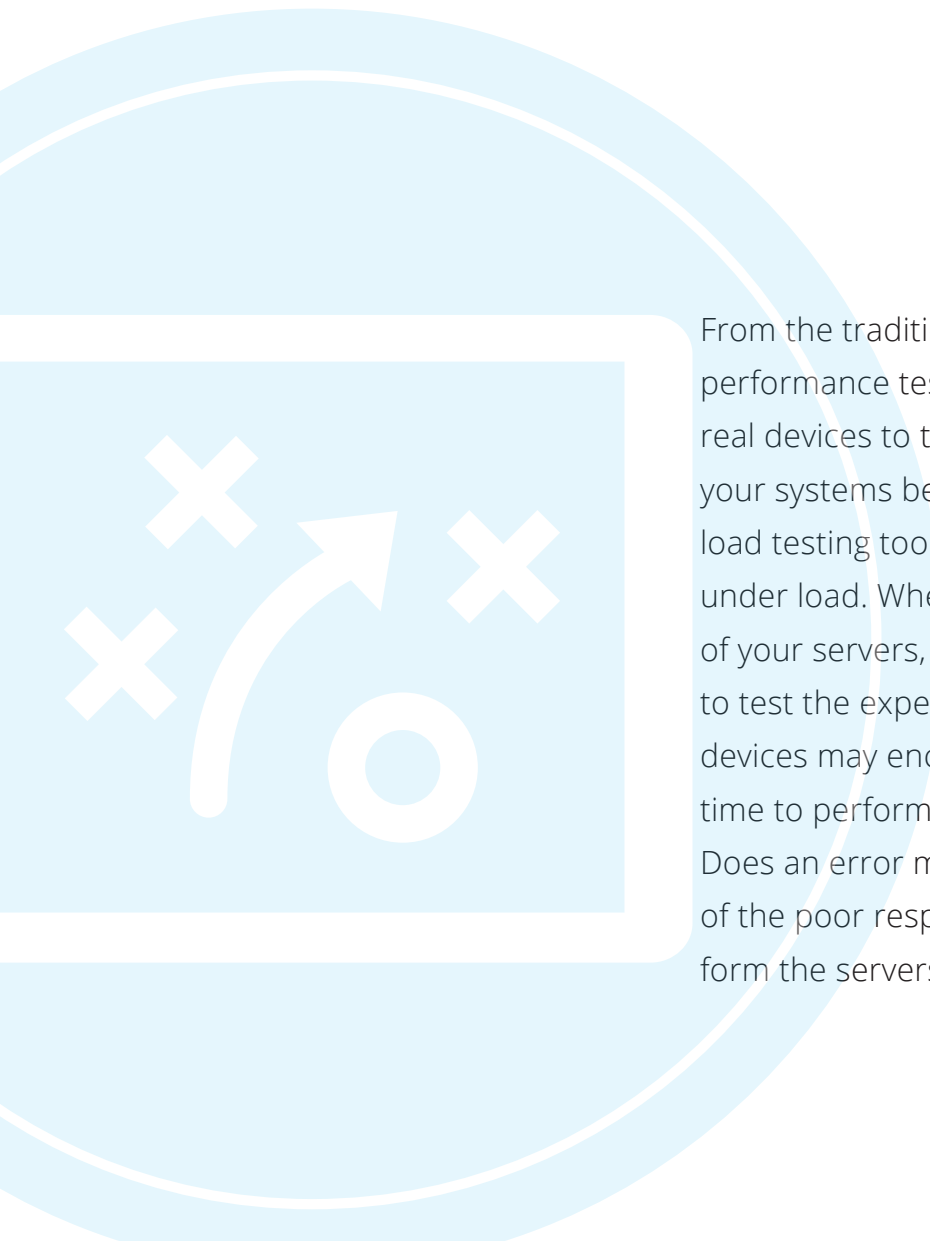
For example, if you have an app that is only deployed one time per year for a special event, such as a conference or meeting, or even an app that only requires a few yearly updates, then manual testing may be a better fit for your organization. If the application has a repeatable process, it is a very good candidate for automated testing.

Automated testing is done in various ways. You have automated unit tests, which should be created to cover the methods

within your code. Having a comprehensive unit test suite lets you know if changes to code break existing functionality. This is done at build time and you should target to have all of your code covered by unit tests.

Functional automated tests cover the flows and features of your application and often test the application UI. The focus of functional tests should cover as much of the application as possible that has a repeatable and consistent outcome.





From the traditional standpoint of performance testing, you would not need real devices to test the performance of your systems because you would use a load testing tool that can measure servers under load. When testing the performance of your servers, it is often a good idea to test the experience that users on real devices may encounter. Does it take a long time to perform a transaction on a device? Does an error message appear because of the poor responses the app is receiving from the servers?

The other type of performance test, and one that is often overlooked, is how does the app perform in general. For instance, does the user wait for images and data to load as they scroll down the screen? Testing the performance of the application itself is important and when an application is performing well, it usually leads to a good user experience.

#4 Skilled Automation Specialists



Hiring the right automation engineers is an important aspect to making sure that your automation strategy is robust and efficient.

Some mobile automation tools allow experienced automation engineers to use their same skill sets. For instance, Appium builds upon Selenium and therefore your Selenium engineers can quickly ramp up Appium automation.

Some automation tools use basic scripting languages and some use an object-

oriented language such as Java and C#. Make sure that your automation engineers are experienced using the languages needed for automation. It may not be a good idea to turn manual testers into automation specialists because they may not be experienced in writing efficient code.

How to Build a Continuous Delivery Pipeline



So far we have discussed the importance of setting up a device cloud, the benefits of testing and debugging on real devices, and the reasons to build a suite of automated tests. Although these are important steps on their own, it is the integration of these steps into a continuous delivery pipeline flow that can take your enterprise mobility strategy to the next level.

In other words, the three previous steps are the building blocks that you need to explore and plan before launching a continuous delivery strategy. Each part of the process, with its various functions are all interrelated and dependent on each other to properly function.

But, why does it matter?

Do you remember the board game Mousetrap? If you are familiar with this game, then you know that each action in the game sets off an entire chain reaction that is necessary for you to capture your opponents' mice. Each step in the game is dependent on the step right before it and right after it.

Let's compare this scenario to building a continuous delivery process flow. You know that you have a series of steps that must be created and successfully executed to build a mobile application. But, along the way your team may find issues that need to be addressed quickly.

By setting up a continuous delivery process flow, every member of the enterprise mobility team can have the agility and flexibility to debug and to keep the app functioning correctly.



Build a Continuous Delivery Pipeline:



1. Create a Delivery Process

The first step is choosing the right tool to create a delivery process. There are many CI/CD tools that you can choose, from commercial options (Team City, Bamboo, Urban Code) and open-source (Jenkins, GoCD, Concourse) that your team can explore. If you already have a CI/CD tool, then it is probably best to extend this tool to include your mobile delivery.



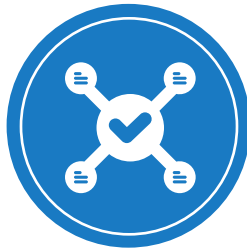
2. Create the Pipeline Needed to Build the App

The next step is to create the pipeline needed to build the application. This means checking out the source code and compiling the app. During the build, it is recommended to make your unit tests blocking, meaning that the entire build process will fail if a unit test fails. This gives the fastest feedback to the development team to create a fix.



3. Install the App on a Device For Testing

Once the unit tests are passed, the next step would be to install the application to the device for testing. With a device cloud, this can be done through add-ins or API calls to upload the application to the cloud infrastructure and install it on the appropriate devices.



4. Run Automated Tests Against the Device

Now that the application is built, the unit tests are passed, and the application is uploaded and installed on the device, it is time to run automated tests against the device. To do so, simply use the built-in mechanism or add-in to execute the tests.

For Appium tests, this usually means to have your test code pulled from the repository or test management system and executed through the build process of the test framework you are using. For those that are using Java, simply execute the JUnit or TestNG tests of your Maven or Gradle builds.



5. Manual Testing and QA

The test results are then used to determine if the build has passed and is ready for internal teams to manually run through the application to find any errors or admissions. This is usually the final step before declaring it ready to go out.

The good thing about this process, is that it makes it easy to repeat for every change because the whole process is automated. This speeds up delivery and provides faster feedback. The faster it is to detect an issue, usually the faster it is to fix. The goal is to speed up delivery and increase quality.

Important Sources for Feedback



But what happens next? After successfully setting up a continuous delivery flow with all necessary tests in place, you need a way to collect results and to receive real time reports. After all, how can you catch and fix any issues if you are unaware of any trouble in the first place?

The answer lies in setting up a continuous feedback loop. By setting up a closed loop process, your team will receive feedback and notifications about any issues as soon as they occur. When you have quick feedback, your mobile developers, testers and QA can isolate issues and debug quickly to keep releases on track.

But, what should you include in your feedback loop?

For your enterprise mobility team to really be agile, you need to receive feedback in a way that is easy to digest. With so many tests running, your team could potentially receive “feedback overload” making it difficult, if not virtually impossible, for your team to sort through. To make feedback meaningful and easy for your team to act on, you want to put solutions in place to give you targeted, focused feedback. Here are 4 of the most important sources for your team to monitor.

#1 Automated Test Results

The automated tests that you set up, whether unit, functional or performance will feed their results back into your continuous delivery pipeline or to a test case management system.

Regardless of where you are getting your results, they must be managed and analyzed by your team. Any execution issues or defects must be addressed by your team quickly to stay on track with releases. Make

sure to make your automated tests rock solid so that in the event of a failure, there is no guessing if the test failed due to issues with the test or with the product.



#2 APM Monitoring



When making sure that your app is running at its peak performance, it is important to measure how the system is running.

APM monitoring tools can help provide real time performance metrics about the health of the system. APM covers a lot of different areas such as performance and availability of the systems, transaction timing on the device and the network, as well as bottleneck detection. All of the data provided gives you insight into how your application is working from the app side all the way back to the systems that support the app. If an issue arises you can get an immediate alert if something is going

haywire. Once an alert is triggered, everyone can jump in and diagnose the issue quickly. Making sure that your critical transactions are successful is key to having a quality app.

One way to get earlier feedback from APM is to leverage the APM tools during pre-production testing. Having this data before the app hits the users can help narrow down issues that can be fixed quickly, reducing costly issues once the app has been deployed.

#3 User Monitoring and Feedback

If you want a better understanding of how your users are interacting with your mobile apps or websites, then user monitoring reports can help you better understand their behavior.

By reviewing these reports, you can see which features of an app are being used, or how users are engaging with different areas on a mobile website.

User monitoring delivers actual, real time information about key trends, user-timing and other metrics. You can use these findings to innovate and improve the mobile experience.



#4 Direct End User Feedback



Test results are not the only feedback your team receives about apps or mobile websites. Do not forget about your users, or mobile consumers. This audience has high expectations for mobile experiences, and checking out any feedback on your app or mobile website is beneficial for you team to consider when making improvements.

One way to get feedback is to monitor the app store reviews. Figuring out why your “star” rating is moving up or down gives valuable feedback about the quality of your application. Reviews can give insights into problems with the application, but it can also give you insights to what customers want more of in the app or on the mobile website.

Early feedback can also come from early release programs for your application. Apple has a program called TestFlight to get early releases of your application out to end users so you can get feedback earlier. You can also use crowd sourced testing programs, like Rainforest and Applause to get users to run through your application and provide valuable feedback.



Enterprise mobility teams face many challenges thanks to digital transformation and consumer demand. There is mounting pressure to develop apps that are better, faster and more innovative than ever before. Naturally with increased development comes the heightened demand for more testing to push out apps that are high-performing and that fulfill the expectations of savvy consumers. In today's mobile world, consumers are now conditioned to expect user experiences that quite simply work with no issues right out of the box.

At Mobile Labs, we believe that mobile teams can thrive in an agile, DevOps world by leveraging a mobile device cloud in conjunction with a continuous delivery strategy. A continuous delivery strategy with continuous feedback will enable mobility teams to rise to the demands of increased development without negatively affecting testing and new releases.

To learn more about Mobile Labs visit www.mobilelabsinc.com

