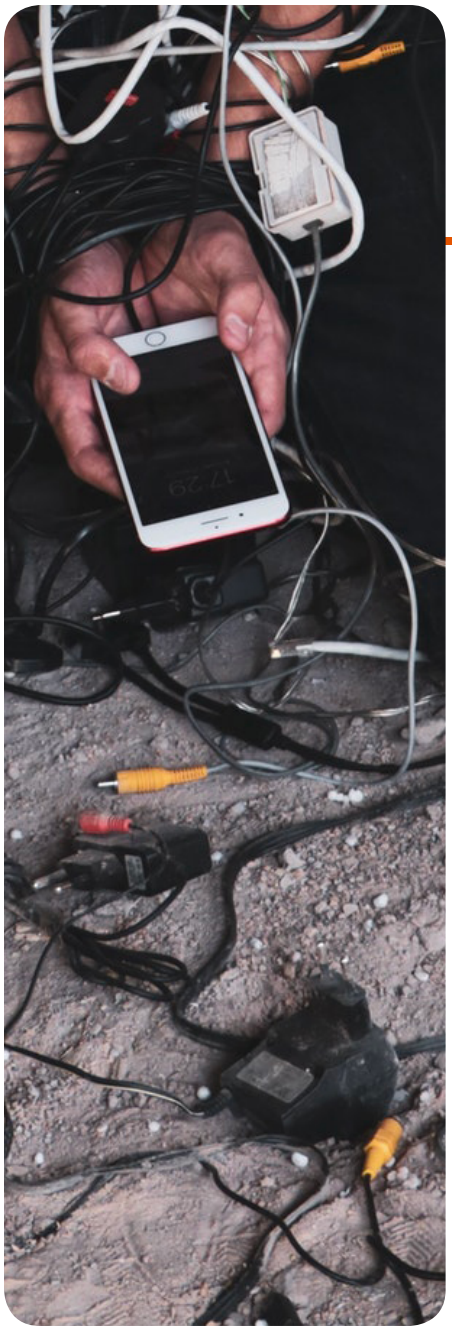
Jump Start Mobile Testing

MARCH 2019

MOBILE LABS

Contents

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



Introduction

The very spice of mobility – its rich variety of mobile platforms (devices, OS versions, and form factors) – can make app and web testing maddeningly difficult. The desire to test on "enough" different mobile platforms can cause chaos, particularly when team members are geographically dispersed. Worse, mobility renders our accustomed rapid access to production-equivalent testbeds, pillars of continuous integration and continuous testing, even more challenging. Unlike desktop web where we spin up virtual machines at will, we cannot spin up real mobile devices. Although simulators have a useful role in app development and checkout, complete testing requires real devices, which cannot be virtualized. Under such restrictions, progress in implementing a DevOps culture can be halted or reversed.

A cloud-based mobile testing lab can overcome most of these challenges. But how to get started? In more than eight years of working with the world's largest enterprises, Mobile Labs has had the opportunity to observe and to help create some of the most successful cloud-based mobile development and testing labs.

The best practitioners manage to retain agile goals while still overcoming mobility's challenges and meeting manufacturers' requirements. In this paper, we outline some of the decisions and programs we have seen in successful enterprises – key characteristics of their DevOps and app delivery programs.

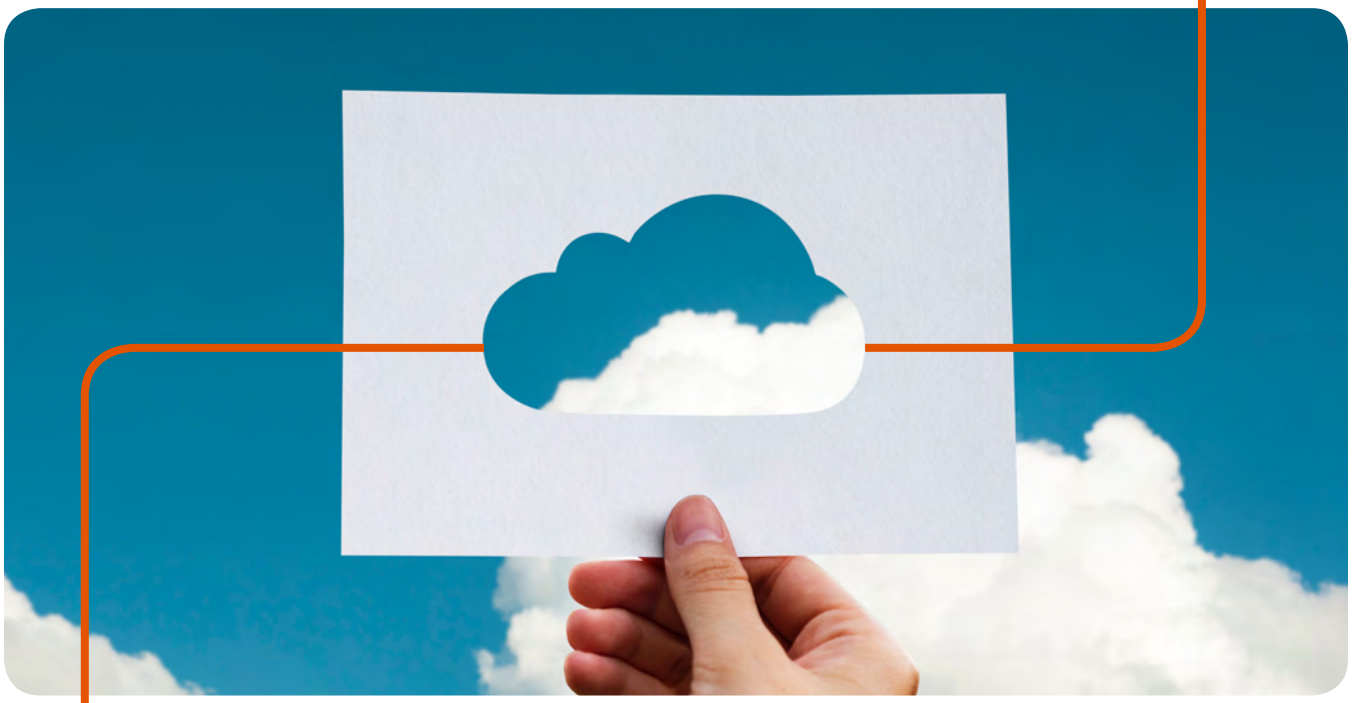
Use Real Devices

Whether using an in-hand device or one in a device cloud, plan to test on real devices. Simulators can be very useful and are often used for simple checks during development, but for hardened testing, simulators are widely recognized to have serious shortcomings.

Mobile apps built on Windows or Mac OS obviously will run on vastly different computing platforms. Smartphones and tablets have different and varying CPU, memory, graphics, and peripheral performance when compared to each other and to PC-based simulators. Mobile web browsers show variegated performance characteristics and may show subtle feature differences with each other and with their desktop namesakes.

Deciding to use real devices for testing is the most important step – plan for how the team will gain quick, easy access to real devices for development checkout, manual testing, and automated testing. Consider how to meet the needs of geographically-dispersed testers – whether they are work- from-home types or a continent away.

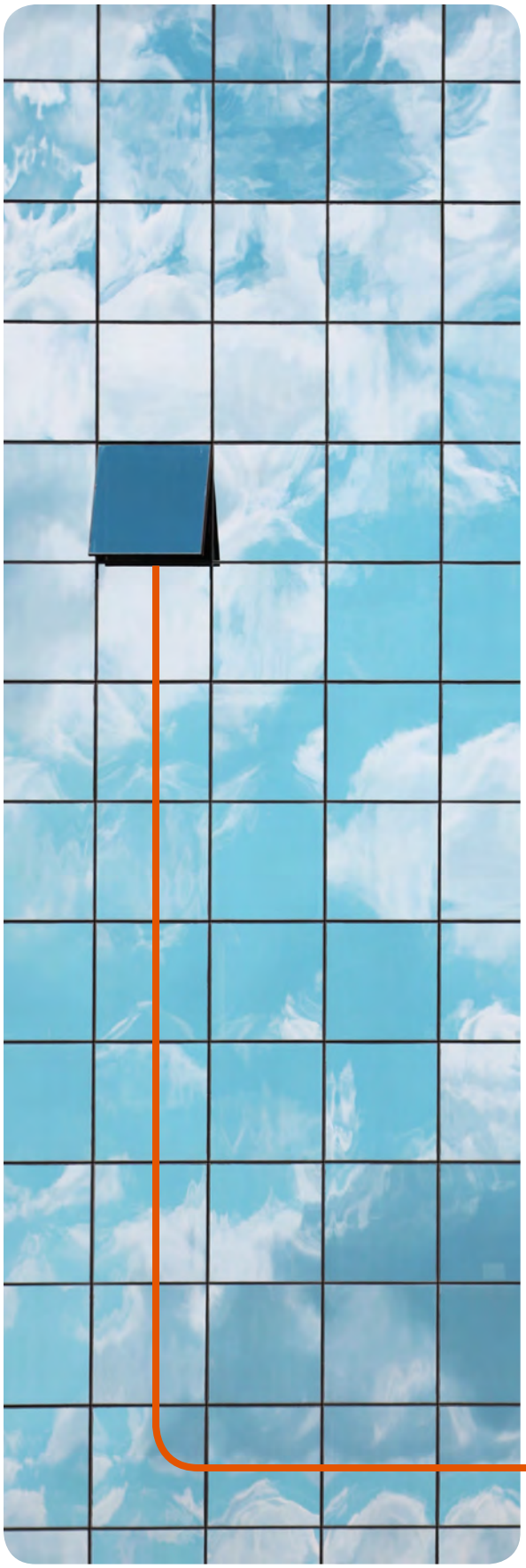
Perhaps for these reasons, TestingXperts[1] includes testing on real devices in its best practices for mobile application testing.



Use a Secure Mobile Device Cloud

Testing on real mobile devices requires a means to efficiently and securely share them, especially when teams are geographically dispersed. Small, local groups may find some success passing devices along using 'sneaker net,' but for teams separated by even small distances, a device cloud becomes critical, more so as distances increase. A mobile device cloud can prevent delays caused by devices that are in transit, checked out but not in use, scarce and not locally available, or temporarily misplaced. Virtual machines and wide area networks have made it possible to support dispersed web deployment teams, and a mobile device cloud likewise enables a team offshore to test on real mobile devices even a continent away.

PC World[2] is unambiguous when it asserts device cloud necessity: "Testing needs to take place on many different devices and operating systems. It is not enough to simply have testers connect devices to their machines locally and test on them one by one. Enterprises should consider creating a device lab on their premises that testers can access remotely."



The cloud should be secure and high-performing and should not rely on either Wi-Fi or cellular data links to carry test control traffic. The latter is especially important when conducting network-based performance testing – a stream of test management data in and out of the device's Wi-Fi connection can invalidate results.

Both manual testing and automated testing should be supported, with the ability to view and control the device in real time. It should be possible to manually manipulate the entire UI of the device so that settings can be changed when needed. The cloud should offer de facto, industry-standard automation, such as Appium© , but also should be able to support most other tools the enterprise chooses.

Some applications will benefit from using devices in the vendor's hosted cloud, and some will have security profiles that require an on-premises cloud isolated from the outside world. Hosted and on-premises variations from the same vendor should be completely compatible, however. Many of our more successful customers use a mix of hosted and premises-based devices.

Whether through hosted and/or premises-based clouds, successful customers tend to use private, dedicated devices not shared with others.

Master Apple Provisioning For All Stakeholders

iOS provisioning, a common roadblock for testers wanting to use real devices, is the process by which Apple grants permission for an app to be installed and launched on a real device. Properly provisioning an app (code signing with an appropriate certificate that belongs to a valid provisioning profile) requires registering an Apple ID in Xcode (at no charge) or joining the Apple Developer Program. Organizations that enroll in the regular developer program are limited to deploying an application to no more than 100 different phones or tablets in any given calendar year. The enterprise-level program supports app deployment to an unlimited number of devices owned by the enterprise.

Successfully testing iOS apps requires mastering the provisioning process, whether through the free method or through one of the developer memberships. Our most successful customers create efficient workflows by sharing credentials between development and testing teams and by training both groups on how to obtain and apply credentials.

The logistics that enterprises use for this process are varied. Some use a single enterprise wildcard provisioning profile and ask developers to sign apps with Xcode. Other customers use individual app profiles, one per app.

Whether developers are in house or external, Apple[3] suggests using the enterprise's Apple developer membership to perform app provisioning. Apple also suggests adding external developers to the enterprise's membership roll rather than using an outsourcer's credentials, a move that keeps both companies in compliance with the Apple developer agreement.

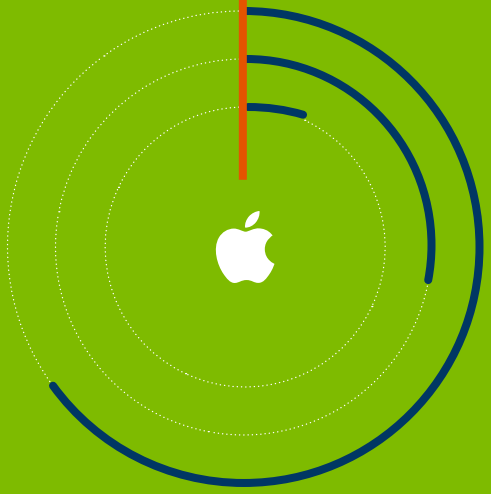
A device cloud's management tools should assist by automating the provisioning process and by providing informative analysis if problems exist. The QA group has as big a role in maintaining and applying proper credentials as any other stakeholder. Planning in advance how iOS apps will be provisioned and how credentials will be made available to all who need them gives a big head start.

Don't Get Crushed by the Numbers

Mobile metrics vendor Apteligent[4] points out that the numbers of OS versions, vendors, device models, app versions, backend services, carriers, types of connectivity, and variations in scale (numbers of users) can add up to **more than 100 million** potential app permutations. No one in the enterprise is likely to live long enough to achieve complete test coverage. An app bottlenecked on testing is not in use by customers or employees.

Operating system coverage may require more devices, however, especially ones that run Android. As of January 2018, Apple[5] reports that 65 percent of iOS devices run iOS 11, 28 percent run iOS 10, and only seven percent run iOS 9 or earlier. The Android market is much more diverse according to Statista[6]. As of February 2018, only 28.1 percent of Android users are running the latest, Marshmallow, and three earlier versions are in double digits (Lollipop 24.6 percent, KitKat 12 percent, and Nougat 28.5 percent).

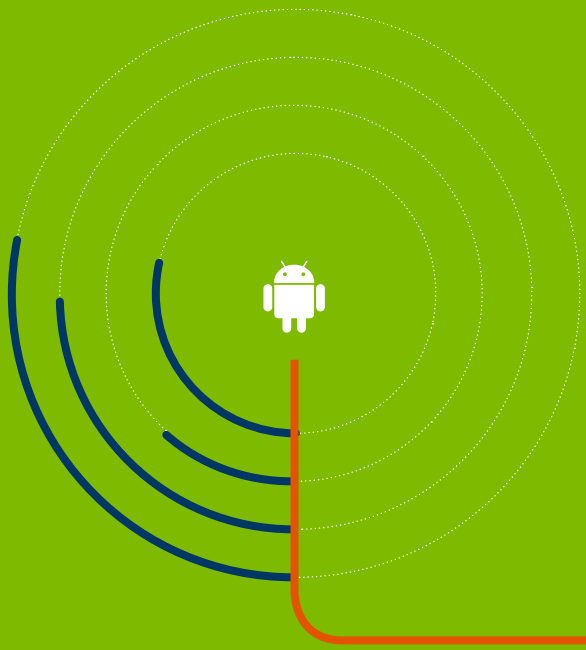
Companies with BYOD (bring your own device) policies also may need to increase device coverage. Insight[7] says that 59 percent of companies allow employees to use personal devices for work purposes with another 13 percent planning to do so within a year. More than 87 percent of companies allow their employees to access business apps using non-company-owned devices.



65% iOS 11

28% iOS 10

7% iOS 9



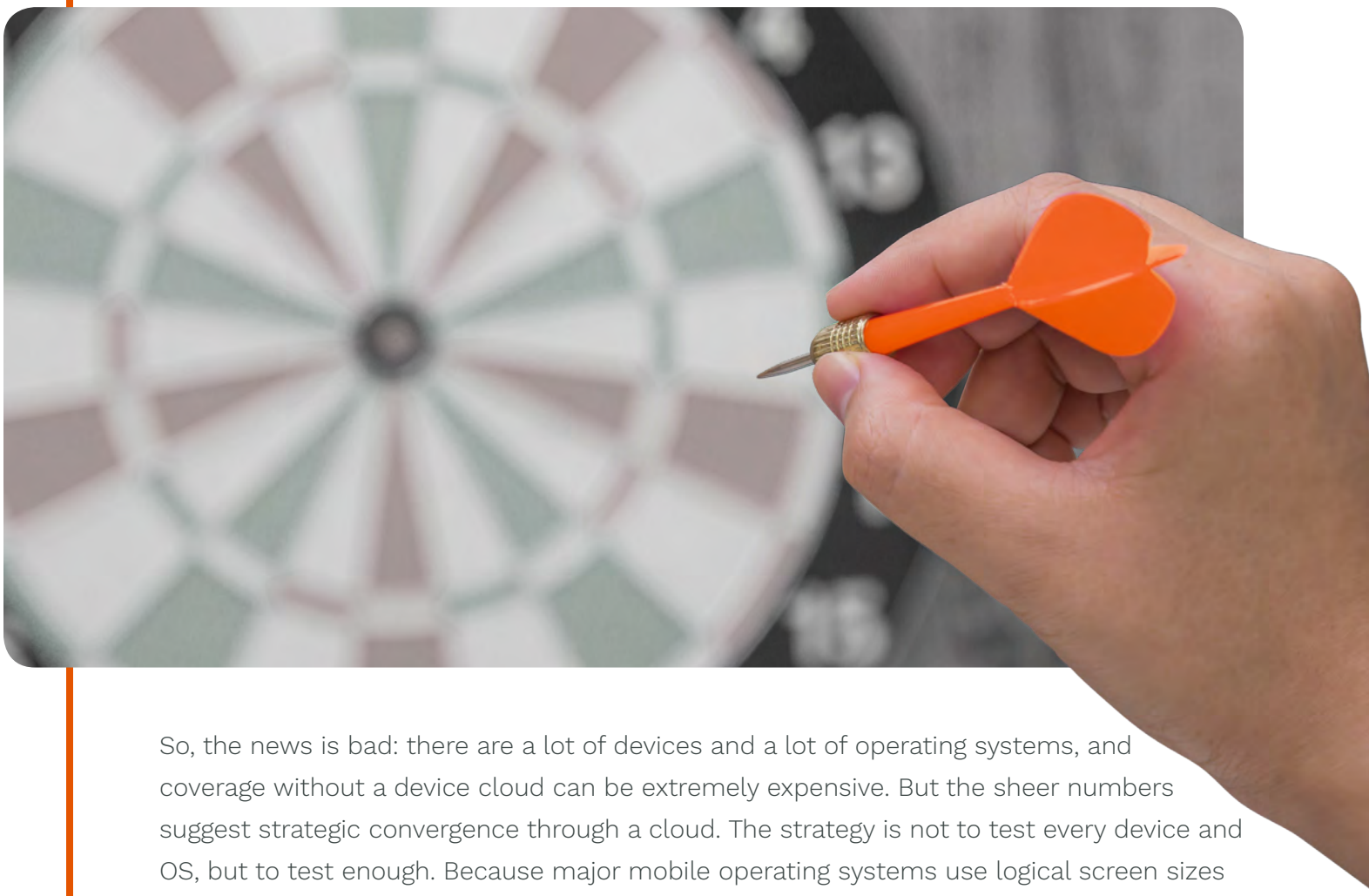
28.1% Marshmallow

12% KitKat

24.6% Lollipop

28.5% Nougat

The task of sharing a well-constructed pool of mobile devices is nearly impossible without a mobile device cloud. If the chosen set of devices is 48, there is no easy way to prevent testers and developers from becoming idle while waiting on devices. Buying one of everything for everybody could solve the problem, but a not-uncommonly-large pool of 100 testers and 50 developers could require 7,200 mobile devices, easily costing more than four million dollars at a cost of only $600 each. Such numbers ensure that flying without a device cloud risks that developers or testers may wait on a device to be relinquished, found, or worse – to be shipped and delivered.



So, the news is bad: there are a lot of devices and a lot of operating systems, and coverage without a device cloud can be extremely expensive. But the sheer numbers suggest strategic convergence through a cloud. The strategy is not to test every device and OS, but to test enough. Because major mobile operating systems use logical screen sizes mapped to physical screens, testing OS versions on representative phones and tablets (throwing in a set for each CPU architecture) can get a lot of coverage. HMG Strategy[8] sums it up nicely, “When looking at the number of mobile device combinations in the market, there may be thousands of combinations, and testing all of them is not practical. Testing only four is too little, testing 700 combinations is too many. If you can test on 40 to 60 combinations of Android and iOS phones and tablets, you will have great coverage.”

DevOps and Continuous Integration, Continuous Testing

Web-based continuous integration (CI) and continuous testing (CT) has dramatically increased workflow efficiencies by shortening cycle times and allowing function to be built and tested in smaller increments.

The idea is so popular a recent Forbes[9] article enthusiastically offers:

> "While the overhead of setting up a CI system might be daunting, it ultimately reduces risk, cost, and time for rework. If you introduce a bug and detect it quickly, it's far easier to get rid of. Since you've only changed a small bit of the system, you don't have far to look and only a small number of changes are lost. This would ultimately result in frequent automated testing and releases! And who doesn't like that?!"

Greatly increased velocity and quality are the benefits of CI. Yet non-mobile, web-app CI relies on cloud-based virtual systems to support an entire IT community. But without the benefits of a device cloud, mobile apps cannot easily benefit from setting up a CI system.

Mobility's inability to provide either satisfactory virtualization or emulation of real devices leads many teams to scatter a few real devices here and there among the team members therefore putting "manual interaction" front and center. Worse, the tools provided by Apple and Google can be difficult to use, time consuming, or may require an administrator's action, all factors that reduce workflow efficiency.

Requiring a developer or tester to remember and find the current app version is an error-prone process, more so if development is making the rapid commits that CI envisions. It is here that mobility, sans a device cloud, falls down on the job. Third parties have the means to move apps from cloud-based websites onto devices, but absent a full-blown MDM (mobile device management system), such an install may require manual intervention. Any attempt at automating access to a local USB device depends on the device being connected at the time it is needed, leading to more potential errors and delays.

Successful customers resolve these problems by using a device cloud as the path to automatically move apps from the build system to device(s) targeted for testing. Once CI automation installs and launches an app on one, more, or all devices in the cloud, test tools can automatically run test scripts. Manual testers also benefit because they know a device always has the current build. The device cloud thus enables continuous integration and continuous testing for mobility.

The device cloud should make it easy for both developers and testers to obtain real devices for checking out builds and fixes, and the cloud's automation capabilities should make it possible to automatically set up production environments (correct device type, correct operating system version).

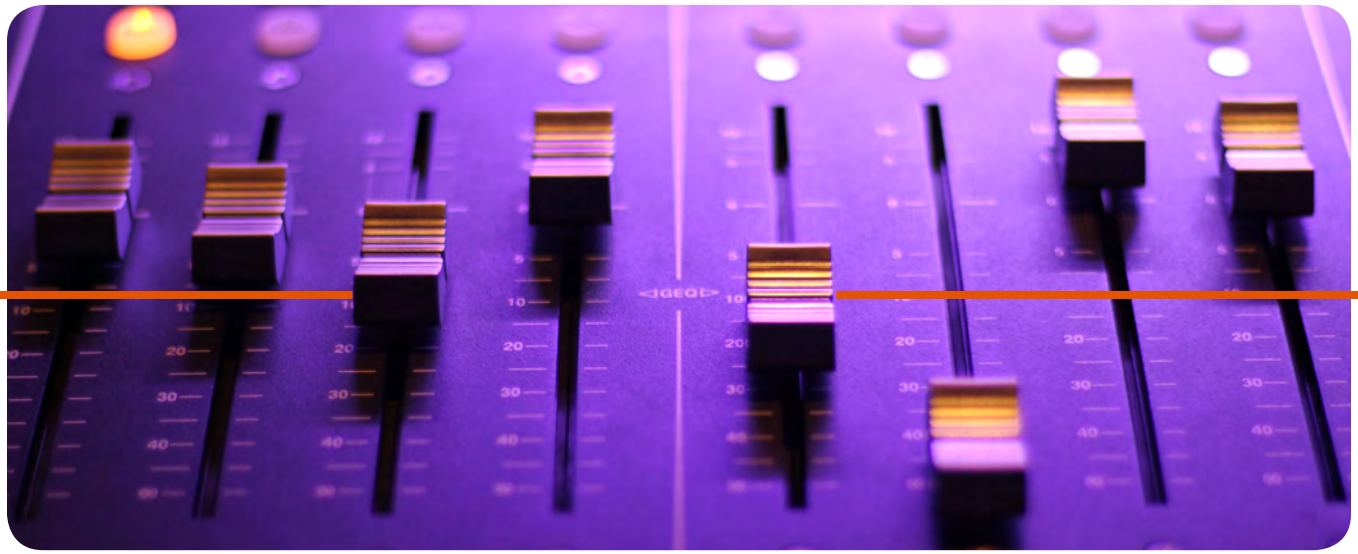
Use Cloud-Based Manual Testing

Jean Ann Harrison[10], a leading software testing and services consultant, says that a sound mobile testing strategy uses manual testing where appropriate and automated testing where appropriate. She suggests that trainability tests, configuration tests, mobile device performance and usability tests are "faster, easier, and less expensive to run manually than to automate."

Many of our customers use manual testing on cloud devices to test app usability, taking advantage of the cloud's ability to make a wide variety of operating system versions and device form factors instantly available. At issue is whether all the platforms successfully deliver an app's new requirements. Bloggers at Software Testing Help[11] note that manual testing is essential for new code where a tester must make continuous, real-time judgements vs. requirements, employing clear and constant observation, questioning, reasoning, and heuristic action.

Moreover, Harrison suggests the best candidates for manual testing are easy for a human being to carry forward onto new devices or new operating systems, but whose scripts might have both infrequent use and require rewrites for new platforms.

A majority of such tests can be performed using a remote-control viewer that connects to a cloud-based device. For example, one very large enterprise's compliance department uses a device cloud to rapidly access a variety of device form factors to check the company's apps for style compliance and proper spelling.



Create a Cloud-Based Community

Shared communication between the test and development teams can shorten cycle times and improve efficiency. Screen sharing sessions that view cloud-based devices make it easy to demonstrate performance problems, appearance issues, failure points, and logic problems over a live shared link, and if the device cloud provides a real-time view of the device, testers can record videos and share test run results with developers. For their part, developers can see the app at its point of failure and help the testers gather diagnostic information.

Accenture[12] also defines mobile DevOps in community terms, calling it a "collaborative approach that fosters close working between app developers, testing and QA teams, and IT operations. It emphasizes an ability to prepare, install, and launch mobile applications and test scripts across all target devices."

Just as important, Accenture describes key capabilities of mobile DevOps, all of which are provided by a device cloud, enabling testing teams to use real mobile devices to:

- Install or remove a mobile application and its different versions
- Select an appropriate variant of mobile device
- Conduct automated mobile app tests on particular features
- Run trusted test scripts on one or more variants of mobile device

Successful customers establish efficient workflows that leverage these capabilities. Developers can reduce the number of cases where an app has problems on a particular class of phone or tablet. Sharing cloud devices with all stakeholders in the app development and delivery process eliminates bugs before they must be creatively unearthed later. Sharing rarely-used devices thus makes it possible to find and fix problems earlier or prevent them in the first place.

Testers and developers should be able to collaborate instantly – even when separated by an ocean.

Don't Try to Find One 'Swiss Army Knife' for Testing

Because enterprise mobile apps almost always use some form of back-end services, thorough testing may involve unit testing, manual testing, automated UI testing, security testing, service virtualization, load testing, performance profiling, and network condition testing, among others. Most enterprises also use tools that manage test cases – both automated and manual -- that include versioning, storage of results, and the ability to data-drive and to automate concurrent testing.

A TechBeacon[13] blog post noted that, "In 2016, software testers moved away from using broad suites for task automation in favor of best-of-breed tools seen as more capable of handling the demands of DevOps and agile development. That trend is likely to continue in the coming year."

Choose cloud support that is open to integrations among tool sets that show leadership in an area of specialty. The mobile tool landscape is a rapidly evolving locus of innovation; remain ready to bring those innovations to workflows.

Use What You Know and a Cloud with Staying Power

The most successful test labs do not limit teams to a vendor's vision of development, testing, or DevOps. The lab meets the organization "where it lives" on day one, but also delivers its benefits even as the organization evolves its sets of tools and workflows.

Maybe no better example of leveraging existing skills exists apart from the rise of Appium in mobile testing, fueled partly by the numbers of testers who previously used Selenium-based tools for web. Appium has a high adoption rate among our customers, even those who were previously using commercial tools.

We see our customers move from tools like UFT and Rational into Appium while still keeping their device-cloud-based testing labs in full operation. The story of mobile testing has been one of transition from the very beginning, and success or failure in keeping up with change often depends on whether the testing lab's core – the device cloud – helps or hinders the change.

The most common and successful transitions gain or maintain device cloud benefits for:

- Continuing to use an existing automation tool
- Transitioning from manual testing to an automation tool
- Moving from an older automation framework to a new one such as Appium

Our most successful customers have used the device cloud as a means to retain the efficiency of their workflows as they have moved from older commercial test suites into Appium or from manual testing to their first automation efforts. For them, the device cloud significantly improves manual testing efficiency while supporting new automation-based workflows. They did not need to 'step off a cliff' but made the transition in an orderly way.

A device cloud that supports both the old and new sides of the above transitions can give the organization the muscle needed not only to survive but to thrive. A strategy based on new tools or languages may mean training costs and delays while everyone gets up to speed on a new way of doing business. Training, education, and building infrastructure may represent significant hidden costs when implementing new tools. The right testing lab can smooth out these bumps in the road.

Know What You Are Testing

For native iOS and Android apps, the most challenging mobile DevOps issues arise because such apps dramatically diverge from the web-based delivery model. Desktop web tools test back-end services on a small set of desktop browsers that get always-fresh code served up by a web server -- making it more or less trivial to test current software.

Mobile apps, however, install each build on a real mobile device – where it stays until replaced. Some care is needed to ensure against testing an old build. The process of quickly building production test environments is also challenging, requiring matching the right app version to the right platform (device and operating system level). The best defense is to use a mobile device cloud that can version apps and can automate moving apps from the build system onto the right devices without human intervention.

Test scripts should be able to lean on the cloud for assistance in choosing the right device, for example, “obtain an iOS 10-based mobile phone,” and should not be locked to a specific device.

The need to maintain OS-specific app versions can arise from the object-based nature of the operating systems themselves. Changes in underlying frameworks can change an app’s behavior even when the app has not been changed. The device cloud gives the tools needed to master this complexity – by routing a new build to the proper platform.

Satisfy the CISO

Mobile devices are wireless and depend upon wireless network connectivity – public and private networks – for much of their function. So, testing must proceed not only on real devices, but also on real devices that have public network access. Mixing public network access by a mobile device with the ability to test the device from a system connected to the internal LAN is a source of potential concern for many CISOs. A device cloud must have a strategy to isolate both testing assets (apps, data, and network packets) from public access or exploitation and must have a strategy for isolating a device's wireless access to the WAN from systems and data on internal LANs. This is true whether the cloud is on-premises or hosted by a cloud vendor.

Additional caution is required for hosted cloud devices. The most secure strategy is to ensure that the devices are dedicated, restricted, and private, and that the hosting vendor provides a means to scrub the devices after they are used.

The most secure model for both on-premises and hosted clouds is for test traffic and control to be performed locally by the servers that control the mobile devices – reducing the data that moves from desktop to tester workstation. Communication between the devices and the test-driving software should occur on non-LAN-based USB links for greatest security from snooping attacks.

An on-premises cloud also offers the ability to establish a test-only wireless network for WAN access that uses enterprise standard proxy and intrusion detection mechanisms while keeping that net separate and unreachable from internal systems.

Directing test data traffic (events, screen shots, object inventories) through the mobile device's USB data port helps isolate the wireless mobile devices from the internal LAN or the WAN by restricting data flow to known ports and formatted messages. The resulting configuration can satisfy corporate security standards as well as satisfy the CISO.

Sources

1 Henderson, Alisha. (2017, June 15). "**8 Best Practices for Mobile Application Testing.**" TESTINGXPERTS Blog. Retrieved from https://testingxperts.wordpress.com/2017/06/15/8-best-practices-for-mobile-application-testing/

2 Arieli, Guy. (2017, April 12). "**6 tips to solve Mobile App Testing problems." PC World**. Retrieved from https://www.pcworld.idg.com.au/article/617576/6-tips-solve-mobile-app-testing-problems/

3 "**Support: Membership.**" Apple Developer Program. Retrieved from https://developer.apple.com/support/membership/

4 Levy, Andrew. (2014, May 14). "**Newbies Guide to Mobile APM.**" Apteligent Blog. Retrieved from http://www.apteligent.com/2014/05/newbies-guide-to-mobile-apm/

5 "**Support: App Store.**" Apple Developer Program. Retrieved from https://developer.apple.com/support/app-store/

6 "**Distribution of Android operating systems used by Android phonz owners in February 2018, by platform versions.**" Statista. Retrieved from https://www.statista.com/statistics/271774/share-of-android-platforms-on-mobile-devices-with-android-os/

7 Lazar, Michael. (2017, November 16). "**BYOD Statistics Provide Snapshot of Future.**" Insight Blog. Retrieved from https://www.insight.com/en_US/learn/content/2017/01182017-byod-statistics-provide-snapshot-of-future.html

8 Sweet, Thomas & Hoffman, Tom. "**Testing for a Great Mobile Application Experience.**" HMG Strategy. Retrieved from http://hmgstrategy.com/resource-center/articles/2017/11/08/testing-for-a-great-mobile-application-experience

9 Zhou, Anne. (2018, January 9). "**The Principles of Continuous Integration and How It Maintains Clean Code and Increases Efficiency.**" Forbes. Retrieved from https://www.forbes.com/sites/forbesproductgroup/2018/01/09/the-principles-of-continuous-integration-how-it-maintains-clean-code-and-increases-efficiency/#284bacc11920

10 Denman, James A. "**Mobile App Quality Takes More than Just Software Test Automation.**" Tech Target. N.p., May 2013. Web 16 May 2014 http://searchsoftwarequality.techtarget.com/tip/Mobile-app-quality-takes-more-than-just-software-testing-automation

11 (2017, March 27). "**Manual Testing Tutorial (Free Course with 100+ Tutorials). Software Testing Help.** Retrieved from http://www.softwaretestinghelp.com/manual-testing-tutorial-1/

12 "**Bridging the Gap: DevOps in mobile app development.**" Accenture. Retrieved from https://www.accenture.com/us-en/insight-bridging-gap

13 Mello Jr., John P. "**Testing engineers look to best-in-breed tools to automate.**" TechBeacon. Retrieved from https://techbeacon.com/testing-engineers-look-best-breed-tools-automate

About the Author

Michael Ryan

Senior Vice President & Chief Technology Officer

Michael Ryan has been the CTO of Mobile Labs since early 2012 and has successfully managed Architecture, R&D, and Support during the development and delivery of the company's flagship products. Since 2016, he has helped oversee organizational, product, and go-to-market strategies.

Ryan's career spans 42 years in Enterprise Computing as a software engineer, manager, and executive in large-scale computer system and operating system support and development, computer system emulation, application lifecycle automation, and mobility lifecycle automation. He holds a B.A. Summa Cum Laude in American Literature and has completed PhD coursework.

About Mobile Labs

Mobile Labs remains the leading supplier of in-house mobile device clouds that connect remote, shared mobile devices to Global 2000 mobile web, gaming, and app engineering teams. Its patented GigaFox™ is offered on-premises or hosted, and solves mobile device sharing and management challenges during development, debugging, manual testing, and automated testing. A pre-installed and pre-configured Appium server provides "instant on" Appium test automation. GigaFox enables scheduling, collaboration, user management, security, mobile DevOps, and continuous automated testing for teams spread across the globe and can connect cloud devices to an industry-leading number of third-party tools. For more information please visit mobilelabsinc.com.

Contact Mobile Labs

Email **info@mobilelabsinc.com** Phone **1 (404) 214 5804** Web **mobilelabsinc.com**