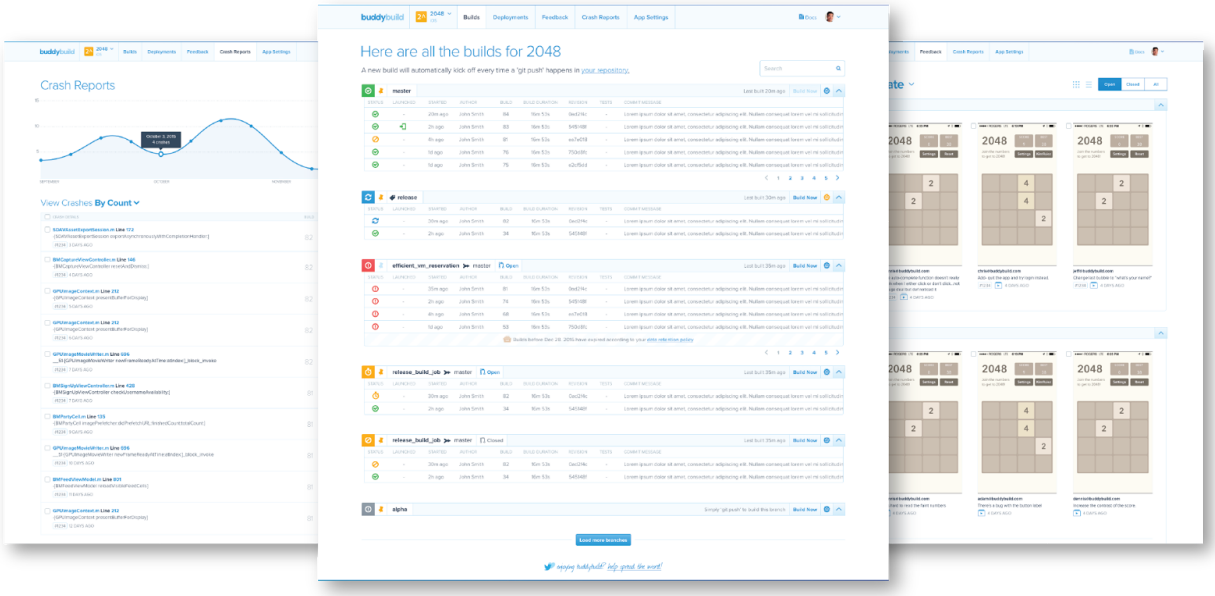
Increasing Mobile Developer Productivity with buddybuild



"After the first day of using buddybuild, our team's productivity quadrupled. Buddybuild has completely changed our workflow and how we do business."

-New York Times Engineering

December 2017

Table of Contents

Measuring Mobile Developer Productivity

A recent study found that “the most successful mobile apps release 1-4 updates a month.” Today’s consumers expect new features and bug fixes on a regular cadence and developer productivity is critical for teams focused on shipping better software, faster, to meet these increasing consumer demands.

To deliver on these demands, teams must make efficient use of their development resources and continue to measure and improve developer productivity. Development teams can measure their overall productivity in many different ways: total lines of code, time spent on task, number of features delivered, velocity of the team, issue close rate, and many more.

Regardless of the team’s preferred metric, the fastest way to improve a mobile developer’s productivity is by having them spend more time working on features and less time on non-feature related tasks. Teams that can improve developer productivity are better suited to deliver on the ultimate goal of shipping higher quality apps, faster that improve overall business results.

> “With buddybuild, we have a reliable build system that allows our engineering efforts to focus on developing new features and delivering quickly to our global markets.”
>
> *- Ian Bird, Director of iOS and Android at Plex*

Causes of Mobile Developer Productivity Loss

Today, mobile developers can spend up to 50% of their time on non-feature related tasks. This loss of productivity is often tied to a lack of proper tooling and infrastructure to support the unique needs of the mobile development process.

Instead of building features, mobile developers are spending their time:

Building and Maintaining Ad Hoc Tools

Development teams invest in 3rd party tools to automate repetitive and error-prone tasks. However, many of these tools are meant to address only a specific component of their workflow. Ultimately, teams spend hours cobbling together a collection of point-solutions, often never intended for mobile development, to try to solve the unique needs of mobile applications.

The result is typically a brittle, inefficient system that is hard to maintain and brings minimal overall productivity gains.

Procuring and Managing Hardware

IT departments are typically experienced in managing the physical or virtualized environments needed for server based applications, but many are not equipped to handle the new requirements of mobile application development.

Specifically, building and testing iOS applications requires Apple hardware. Apple hasn't produced a "rack and stack" server product in ~13 years and has made it clear not to expect server grade infrastructure in the foreseeable future. As such, development teams are relegated to creating an 'ad-hoc' datacenter center with Mac minis, Mac Pros or MacBook Pros. It is not uncommon to walk through an office filled with mobile app developers and find a stack of Mac mini's lying around on a desk or tucked away in a corner. Development teams are setting up "off-the-grid" hardware and IT and security teams are facing the challenges of securing and managing it.

> "Apart from the server itself being riddled with issues, the limited disk space, and time required to continuously update the keychain made our mac mini unusable. There was also a constant risk of spilling coffee on our build server, which added to the urgency of finding a new, secure solution."
>
> *- Jon Trainer, iOS Lead at Kinsa*

Fortunately, Android application development eases some aspects of the hardware challenges compared with iOS development. Specifically, the compilation of Android applications can be performed on traditional server hardware. That said, Android developers are faced with a different challenge — ensuring that their app runs seamlessly on the exact devices their userbase owns. Testing on physical devices is the best way to surface issues that are specific to a certain OS, device, or vendor permutation. However, the fragmentation of the Android ecosystem makes this an intractable problem. Purchasing, setting up, and maintaining potentially hundreds of phones falls outside the experience and scope of even the most adventurous and experienced IT teams.

Waiting on Builds and Test Runs

Developers can spend up to 20% of their time waiting on necessary builds and tests to finish. There are two primary causes for these delays: slow build times, and lack of hardware capacity.

Slow build times are a cause of frustration for many developers. As developers iterate on their application, they aren't able to merge new code changes into the master code line until

necessary build and test runs have completed. The longer it takes this process to run the longer the developer sits and waits.

Optimizing the build and test execution speed is possible, through better scripting and caching, but it can take considerable time and expertise that many developers may not have.

Additional time is also lost when limited hardware capacity is available to teams, as builds and test runs are queued until capacity becomes available.

> "Our in-house build machines weren't optimized for mobile projects and prevented us from moving at the speed we needed. More often than not, builds would fail because of insufficient resources, and a retry was required to solve the issue. However, a retry often means a developer would wait another 2 hours for the pull request to build before finding out if the new changes could be merged."
>
> *- Paul Yorke, iOS Lead at The New York TImes*

Diagnosing Bugs, Crashes, and Flakey/Failed Tests

Historically, creating a reliable testing framework for mobile projects has been a notoriously tedious process for development teams.

Unspecialized build infrastructure and flakey test simulators make Unit and UI test runs nondeterministic. What's more, the failure messages themselves don't provide the development team with enough context to reproduce and resolve the problem. Unfortunately, tracking down failures or erratic behaviour associated with test execution can be incredibly tedious, and can take hours of a mobile developer's time to rectify.

Similarly, the consumers of the mobile application are rarely equipped with the tools necessary to send actionable user feedback during beta testing. As a consequence, user feedback is often non-existent, unclear or sparse. It generally only comes from a small group of dedicated testers who may not be representative of the entire user base, and is submitted through many channels (like email or feedback forms) that lack enough information to reproduce the problem.

Development teams spend countless hours digging through the codebase and speaking with the end user to identify and resolve the reported issues. If mobile development teams were provided with the information that describes what path the user took to cause the reported issue, and the metadata of the device on which it occurred, many development hours could be saved with each new release.

Manually Deploying Builds to Stakeholders

For any application there are many potential stakeholders who need to receive builds and updates throughout development. Whether it's a group of external beta testers, an internal QA team, the Project Manager, or a Company Executive, developers must have the ability to send any build of their app out to stakeholders on demand.

For traditional web applications, sending a specific build is as easy as sharing a URL of webpage. However to share a mobile app, developers must ensure stakeholders can install and open the app on their device.

Because of Apple's added security requirements, this is a particularly time-consuming and often frustrating process for iOS developers. Distributing an iOS app to any new stakeholder or test device is an extremely convoluted process that introduces significant friction to the development process. Making a new stakeholder's device available for testing and sending the application to that stakeholder can take days, wasting the development team and the stakeholder's time. Furthermore, if you cannot send a build to a stakeholder and trust they will be able to reliably install it, a development team will be less likely to send out builds at all.

Continuous Integration and Deployment to Improve Developer Productivity

To increase productivity, teams must invest in tools and infrastructure that allow developers to build and distribute apps as quickly as possible while maintaining the quality of their codebase. This approach is commonly known as continuous integration (CI) and deployment (CD). Development teams that invest in the right tools are able to deliver far more with the same headcount.

What is Continuous Integration and Deployment (CI/CD)

Continuous Integration (CI) is a development practice that requires teams to integrate code into a shared source code repository several times a day. Each check-in is validated by an automated build and generally includes the execution of Unit and UI tests.

By increasing the frequency of submitting code that implements features, addresses design tweaks, and fixes bugs, teams can identify and resolve issues earlier, saving them significant costs of dealing with issues later in the development lifecycle.

Continuous Deployment (CD) refers to the automated release of new code to testers, stakeholders, and customers.

By automating the deployment process, development teams are able to send new features or bug fixes to beta testers and into production faster, and easier than ever. Using a complete CD system, testers can review changes and provide feedback, stakeholders can sign off on the changes, and customers receive improvements as soon as they are made available without necessitating an engineer to drive this process.

Why Mobile Development Teams use CI/CD

Fix Broken Builds Faster

Developers can waste hours trying to identify which code changes caused an application to break. With multiple team members working on the same code base, it is not uncommon for one person's change to break another person's code or vice versa. When team members wait too long to check in code or only run unit tests on occasion, it becomes hard to figure out which code changes “broke the build”.

Mobile teams implement CI processes to encourage team members to check-in their code on a regular basis, triggering automatic builds and the execution of Unit and UI tests. This ensures that new additions don’t break the functioning work that came before them, and developers whose code does in fact “break the build” can be notified quickly and resolve the issue.

Deploy Features and Bug Fixes More Frequently

Deploying mobile applications can be slow, confusing, and error prone, due to the complex requirements mandated by Apple and Google. Fewer deployments mean less feedback from stakeholders which can result in lower quality applications.

Mobile development teams implement continuous deployment processes to minimize the manual steps it takes to distribute and install mobile applications. Allowing them to deploy any version of an app to an individual, a group of testers or stakeholders, or to the Apple App Store or Google Play Store.

Challenges of Maintaining your own CI/CD System

Many teams are eager to implement CI/CD solutions to automate repetitive tasks and improve developer productivity. However teams often underestimate the level of effort and expertise needed to establish and maintain these system and tools. As a result teams run into challenges that may ultimate decrease gains and increase costs.

Many teams try to implement CI/CD systems by setting up an in-house system that is tailormade to their development process. However, these in-house systems often consist of retrofitted legacy web infrastructure and custom scripts which constantly need to be updated, and require extensive maintenance by the development team to remain functional. In fact, these systems are often so “hands-on” that teams will hire a build engineer to dedicate his/her time to building and maintaining the team’s CI / CD system.

Ultimately, teams must account for the added complexities and challenges associated with building and maintaining a CI / CD system for mobile applications.

Some of the challenges teams must consider are:

Addressing Hardware Needs & Configuring Build Machines

Teams must consider the different hardware requirements for iOS and Android applications; Android apps can be built on traditional server hardware, but iOS apps need to be compiled on Mac hardware.

And, before building their application on the necessary hardware, teams must set up a working CI system for their app. This setup is very involved, and requires a series of custom scripts and steps to configure. For even the most adventurous and experienced mobile development teams, manually setting up and continually maintaining a CI system is outside their experience and scope.

> “All the CI/CD systems we tried weren’t able to accommodate the speed and agility needed to keep up with our release cycle. Our previous release process, a retrofitted web CI aided with a handful of open source scripts, would fail without warning and without any context. One engineer dedicated many hours each month managing our build scripts within and between projects. This often became the single point of failure when delivering sprints on time."
>
> *- Arthur Sabintsev, Lead iOS Engineer at The Washington Post*

Finally, as development teams grow, so do their hardware needs. Teams not only need access to the proper hardware, and the skillset to configure it correctly for their applications, they also need enough hardware configured properly to quickly build and test their apps to meet the requirements of the business.

Keeping up with Platform Vendor Changes

As an added complexity, Apple and Google regularly release updates to their operating systems (OS) and development tools. Developers can spend days making sure their apps will not only work with these updates but can also take advantage of new features released.

For example, Apple releases updates to Xcode, the IDE used by developers to build IOS apps, anywhere from 5-7 times a year. Developers can spend hours downloading, installing and testing their apps with not only the production version released, but also each Beta version, to make sure there are no breaking changes.

> Our development slowed whenever Apple released a new version of Xcode. As a result, the work required to fix or update our build infrastructure was the equivalent of losing an engineer. “One of the team would need to stop what they were working on and dedicate their time solely to monitoring and patching the build system."
>
> *- Ian Bird, Director of iOS and Android at Plex*

Additionally with each platform OS update, developers again must test their apps to ensure they work as expected. For iOS apps this is pretty straight forward given Apple has tight control over the hardware, but for Android, OS changes need to be tested across many permutations of physical devices.

Developer Productivity Benefits with **buddybuild**

Maximizing developer productivity is critical for companies focused on shipping higher quality software faster. Buddybuild was designed specifically to help iOS and Android development teams iterate on their projects.

> "Buddybuild has the best and most unique set of features out there. Nearly every other service is focused on generic build tasks. Most other CI / CD solutions we tried were so unspecialized and non-mobile focused it became unuseful, but with buddybuild, it's quite the opposite."
>
> *- Monzo Engineering*

By tying together continuous integration, continuous delivery and an iterative feedback solution into a single, seamless platform, buddybuild is increasing the productivity of thousands of development teams by allowing them to:

Spend more time on Feature Development

Buddybuild eliminates the need for developers to build and maintain their own CI/CD systems, giving teams more time to build valuable features for their customers.

Developers no longer need to spend time building and maintaining ‘ad hoc’ systems and tools, procuring and managing hardware, and trying to optimize builds. Developers can simply check-in their code and buddybuild will do the rest. Buddybuild will inspect their apps, provision the right hardware on the fly, download and install required dependencies, process the build, and execute all tests.

> “Buddybuild saves us an hour of review time for each pull request. Time we now put towards building new features and improving our code coverage."
>
> *- Firefox team, Mozilla Engineering*

For teams building iOS applications, buddybuild will also take their most recent successful build and run it against the newest version of Xcode, including the betas, to give developers early insights into any changes that may break their app.

> “For all 20 iOS applications and libraries we have under active development, we’re able to focus solely on improving our codebase and iterating on our projects with buddybuild”
>
> *- Arthur Sabintsev, Lead iOS Engineer at The Washington Post*

Release Quality Builds Faster

WIth buddybuild, distributing a mobile app is as simple as sending an email. Buddybuild integrates directly with the Apple Development Portal and Google Play Console to automate the manual, error prone steps needed to distribute and install mobile applications.

Buddybuild’s continuous deployment system allows teams to configure and manage which builds are deployed, how often, and to which set of users.

Additional time savings come from buddybuild’s CD system handling the code signing process for iOS applications. Buddybuild automatically provisions new stakeholders’ devices on behalf of

the development team to save them the time previously wasted collecting, entering, and troubleshooting devices ID's. With buddybuild's CD system, teams have a "one-click" experience to deploy their application, and the involved stakeholders have a "one-click" install experience.

> "Buddybuild consistently builds our complex projects. With no scripts required, it manages our dependencies, signing certificates, and provisioning profiles across all projects for us. This benefit alone freed up one engineer's time to start working on new features again."
>
> *- Arthur Sabintsev, Lead iOS Engineer at The Washington Post*

Gather Actionable Feedback from Stakeholders

To provide developers with the actionable feedback needed to resolve issues faster, buddybuild offers a feedback and crash reporting Software Development Kit (SDK).

The SDK makes it easy for users to send annotated feedback directly to developers using the screenshot feature. In addition, developers can more easily reproduce issues by gaining access to the actionable information they need such as:

- **Actionable Bug Reports from Testers** in the form of annotated screenshots along with all of the metadata from the actual device
- **Crash Reports with Source Context** that can guide developers to the exact line of source code that caused the crash
- **Video Replays of Crashes** providing developers with the video of their users interaction with the application to provide insight into the exact sequence of events that led up to the crash

> "Our workflow for gathering feedback and crash reports from your internal team and external users is seamless with buddybuild's continuous deployment and user feedback system. There's nothing we need to do to instruct team members or end users on how to use the feedback mechanism. Sending with buddybuild is a frictionless experience. As a result our end users send us more meaningful feedback than ever before, allowing us to make informed decisions throughout development."
>
> *-New York Times Engineering*

In summary, buddybuild's mobile iteration platform has increased the productivity of thousands of development teams. By focusing on solving the unique needs of the mobile development process, buddybuild makes it possible for developers to spend less time on non-feature related tasks and stay focused on building value added features that improve your overall business results.

> "After the first day of using buddybuild, our team's productivity quadrupled. Buddybuild has completely changed our workflow and how we do business."
>
> *-New York Times Engineering*

Buddybuild is trusted by companies like News Corp, Slack, Monzo, The New York Times, John Lewis, and thousands of others to provide a reliable and robust infrastructure on which they can build, test, deploy, and gather feedback on their mobile applications.

Do you want to improve your mobile development team's productivity with buddybuild? Sign-up today for a free 21-day trial or contact us at team@buddybuild.com to set up time to speak directly with one of our engineers.