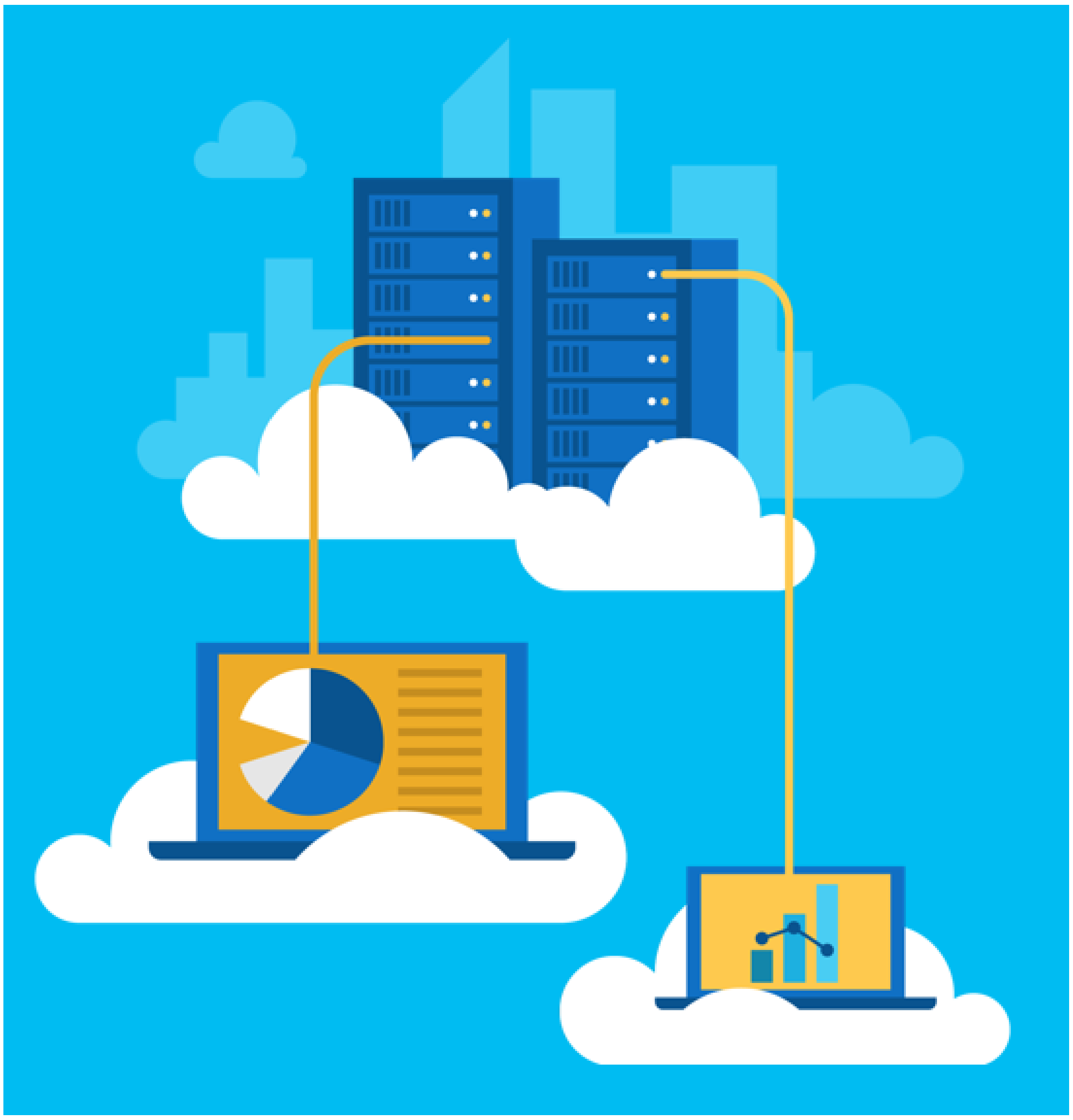
AZ-400 Course

Implement a Mobile DevOps Strategy



Microsoft

Mobile DevOps? DevOps? What’s the Difference?

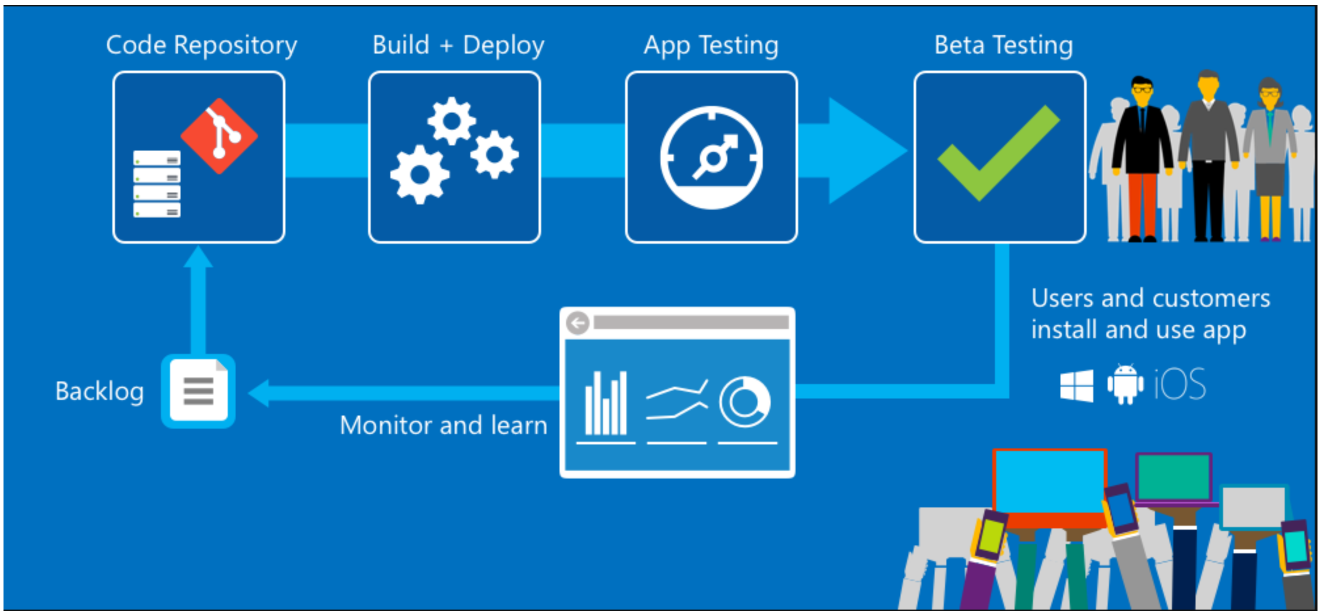
With all things DevOps, the answer is both simple and complex. The practices and the culture should be the same between DevOps and Mobile DevOps (simple), but the tooling required to successfully integrate a Mobile DevOps practice is going to be different than tooling for your DevOps practice (complex). Before we talk too much about the tooling for Mobile DevOps, let’s take a look at the core practices that make up a complete DevOps solution, and understand how those apply to Mobile DevOps.

A DevOps practice should start with a production first mindset, meaning that everything that is done is to enable the app to be successful once it is live for end users. You should not wait until your app is being deployed to start thinking about how to guarantee it is successful in production, that should be built into the app with the first few lines of code. For web apps, this might be integrating logging and application performance management SDK’s. For mobile apps you should you be integrating SDK’s that allow you track and respond to app crashes as their user impact dictates.

The DevOps team should always have clear goals, and the best way to do that is through a prioritised backlog. But how do you prioritise the backlog? With data driven decisions. To get that data, you need to instrument your app so you can understand how users are adopting and using your app. Ideally, you will find a way to enable conversations with your apps users to really understand the changes they want to see and what a great app looks like in their mind.

Now that you know where you should invest in your app, you need to quickly move those new features and enhancements out to your users. You can never sacrifice speed for quality or you simply trade problems, you never bring more value. Ensuring that you can move through all phases of testing to quickly ship a stable app is critical.

The final focal points of a DevOps practice should be around making sure your teams are cross functional, flexible, and enabled to respond based on their learning's from the above practices. They should focus on minimizing technical debt at every opportunity to maintain their flexibility and responsiveness.



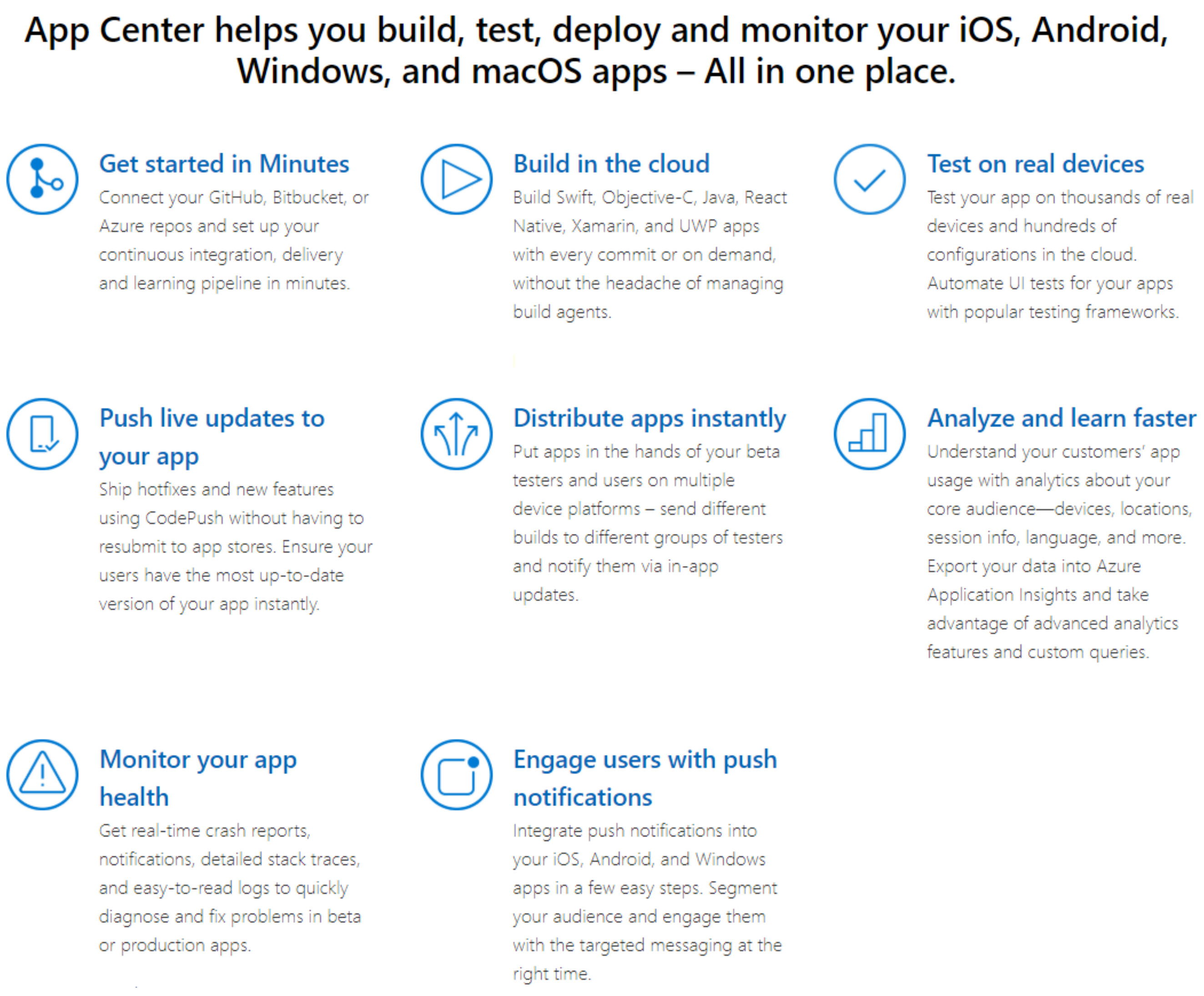
All of this is the same between a DevOps and a Mobile DevOps practice, and it should lead you to the same goals:

- Minimize work in progress
- Ensure highest quality user experience for new apps and updates
- Scale your delivery with automated process

To achieve these goals and maintain a strong Mobile DevOps practices requires a committed team, a cultural change and the right toolset. Microsoft offers this toolset through Visual Studio Mobile Center. Also called app center in short, it brings together multiple services commonly used by mobile developers into an integrated cloud solution.

Introduction to Visual Studio App Center

Visual Studio App Center is mission control for apps. App Center brings together multiple services commonly used by mobile developers into an integrated cloud solution. Developers use App Center to Build, Test, and Distribute applications. Once the app's deployed, developers monitor the status and usage of the app using the Analytics and Diagnostics services, and engage with users using the Push service.



App Center helps you build, test, deploy and monitor your iOS, Android, Windows, and macOS apps – All in one place.

Get started in Minutes
Connect your GitHub, Bitbucket, or Azure repos and set up your continuous integration, delivery and learning pipeline in minutes.

Build in the cloud
Build Swift, Objective-C, Java, React Native, Xamarin, and UWP apps with every commit or on demand, without the headache of managing build agents.

Test on real devices
Test your app on thousands of real devices and hundreds of configurations in the cloud. Automate UI tests for your apps with popular testing frameworks.

Push live updates to your app
Ship hotfixes and new features using CodePush without having to resubmit to app stores. Ensure your users have the most up-to-date version of your app instantly.

Distribute apps instantly
Put apps in the hands of your beta testers and users on multiple device platforms – send different builds to different groups of testers and notify them via in-app updates.

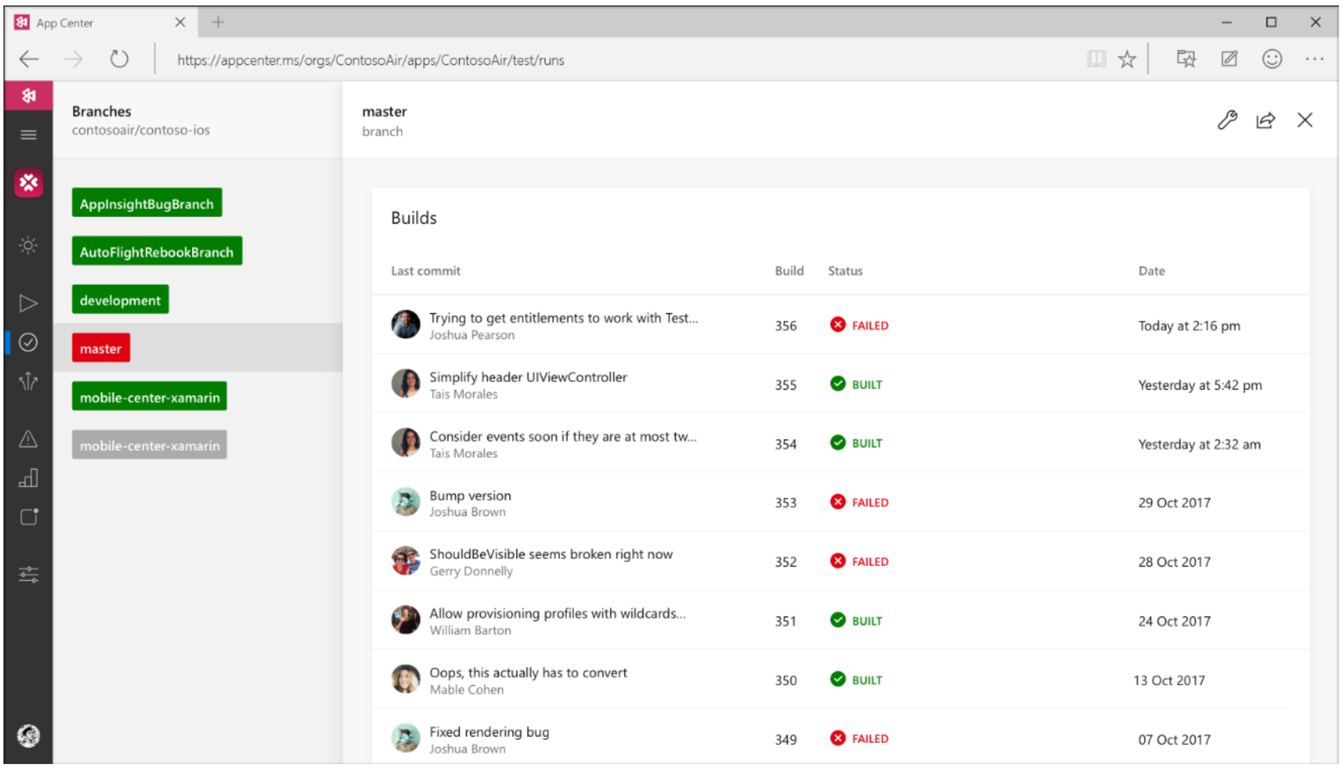
Analyze and learn faster
Understand your customers' app usage with analytics about your core audience—devices, locations, session info, language, and more. Export your data into Azure Application Insights and take advantage of advanced analytics features and custom queries.

Monitor your app health
Get real-time crash reports, notifications, detailed stack traces, and easy-to-read logs to quickly diagnose and fix problems in beta or production apps.

Engage users with push notifications
Integrate push notifications into your iOS, Android, and Windows apps in a few easy steps. Segment your audience and engage them with the targeted messaging at the right time.

Continuous Integration in Minutes

Build apps more frequently, faster. Take the pain out of building your iOS, Android, Windows, and macOS apps locally. Connect to your GitHub, Bitbucket or Azure repos and build your apps automatically with every pull request or on demand, without the headache of managing build agents. Create an installable app package automatically with every push to your repository. Supports GitHub, or Git repos on Bitbucket and Azure DevOps. No additional build hardware required.


App Center

https://appcenter.ms/orgs/ContosoAir/apps/ContosoAir/test/runs

Branches
contosoair/contoso-ios

- AppInsightBugBranch
- AutoFlightRebookBranch
- development
- master
- mobile-center-xamarin
- mobile-center-xamarin

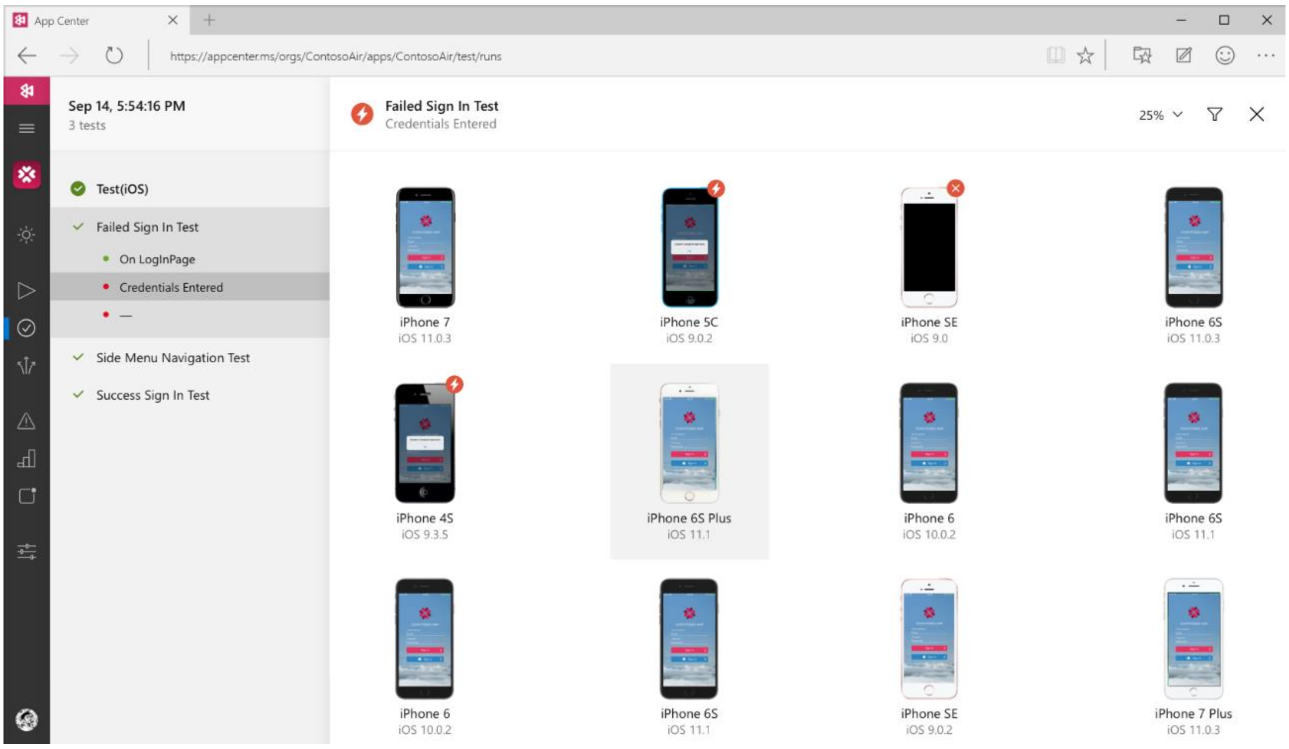
master
branch

Builds

Last commit	Build	Status	Date
Trying to get entitlements to work with Test... Joshua Pearson	356	FAILED	Today at 2:16 pm
Simplify header UIViewController Tais Morales	355	BUILT	Yesterday at 5:42 pm
Consider events soon if they are at most tw... Tais Morales	354	BUILT	Yesterday at 2:32 am
Bump version Joshua Brown	353	FAILED	29 Oct 2017
ShouldBeVisible seems broken right now Gerry Donnelly	352	FAILED	28 Oct 2017
Allow provisioning profiles with wildcards... William Barton	351	BUILT	24 Oct 2017
Oops, this actually has to convert Mable Cohen	350	BUILT	13 Oct 2017
Fixed rendering bug Joshua Brown	349	FAILED	07 Oct 2017

Continuous Quality on Real Devices

Ship higher-quality apps with confidence. Automate UI tests on thousands of real-world devices and hundreds of configurations in the cloud, using popular frameworks such as Appium, Espresso, and XCUITest. Test every UI interaction your users can do, and diagnose bugs and performance problems every time you build, with detailed step-by-step tracking reports, screenshots, and logs. Run your tests on more than 400 unique device configurations. Tests can be written for iOS and Android apps with Xamarin.UITest, Appium, Espresso (Android), and XCUITest (iOS). The next generation of Xamarin Test Cloud.


App Center

https://appcenter.ms/orgs/ContosoAir/apps/ContosoAir/test/runs

Sep 14, 5:54:16 PM
3 tests

- Test(iOS)
  - Failed Sign In Test
    - On LoginPage
    - Credentials Entered
    - —
  - Side Menu Navigation Test
  - Success Sign In Test

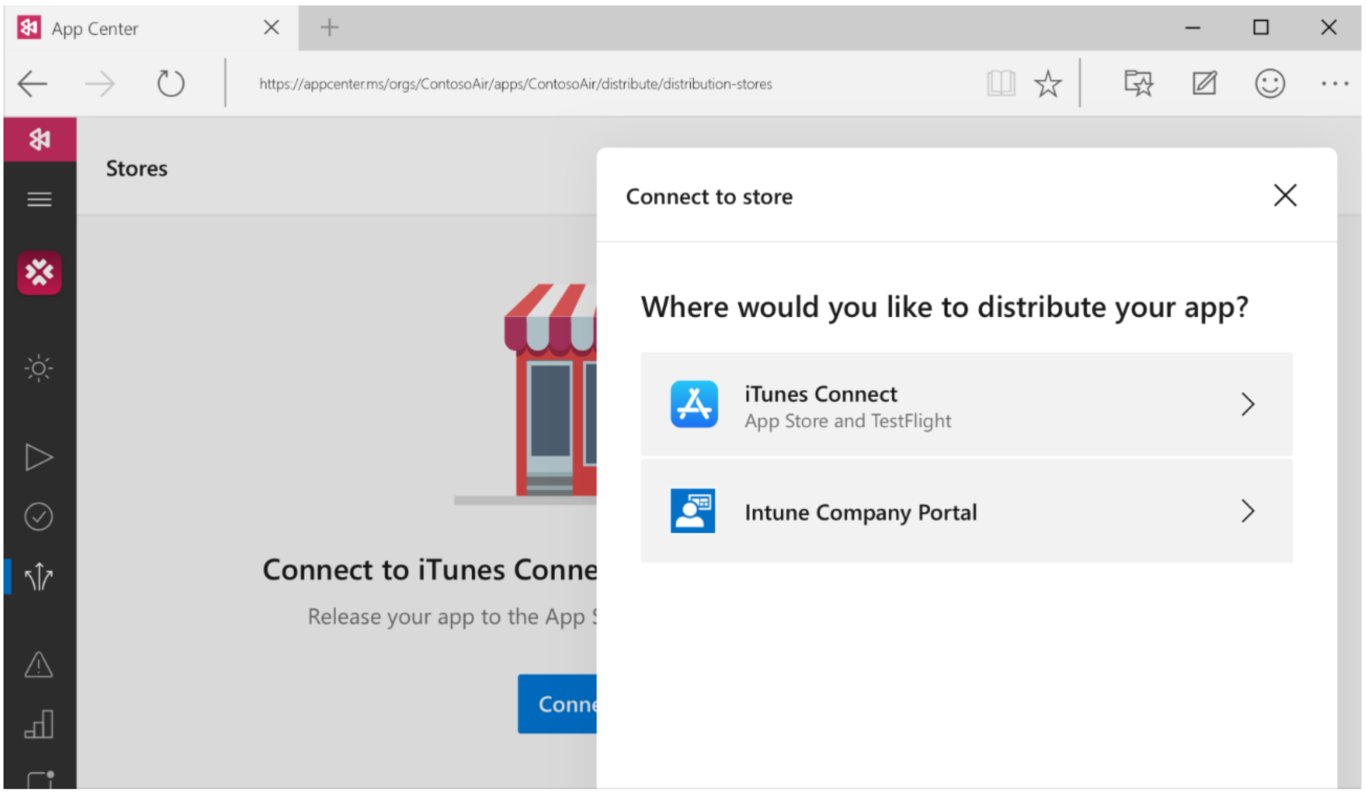
Failed Sign In Test
Credentials Entered

25%

iPhone 7 iOS 11.0.3	iPhone 5C iOS 9.0.2	iPhone SE iOS 9.0	iPhone 6S iOS 11.0.3
iPhone 4S iOS 9.3.5	iPhone 6S Plus iOS 11.1	iPhone 6 iOS 10.0.2	iPhone 6S iOS 11.1
iPhone 6 iOS 10.0.2	iPhone 6S iOS 11.1	iPhone SE iOS 9.0.2	iPhone 7 Plus iOS 11.0.3

Continuous Delivery that Works

Deploy everywhere with ease… Distribute your app to beta testers and users on Android, iOS, Windows, and macOS with every commit or on demand. Send different builds to different groups of testers and notify them via in-app updates. When ready, release to Apple's App Store, Google Play, and to Microsoft Intune. Users can install the app via email distribution lists for testing, much as they'd download an app from the app store.


App Center
https://appcenter.ms/orgs/ContosoAir/apps/ContosoAir/distribute/distribution-stores
Stores
Connect to iTunes Conne
Release your app to the App S
Conne
Connect to store

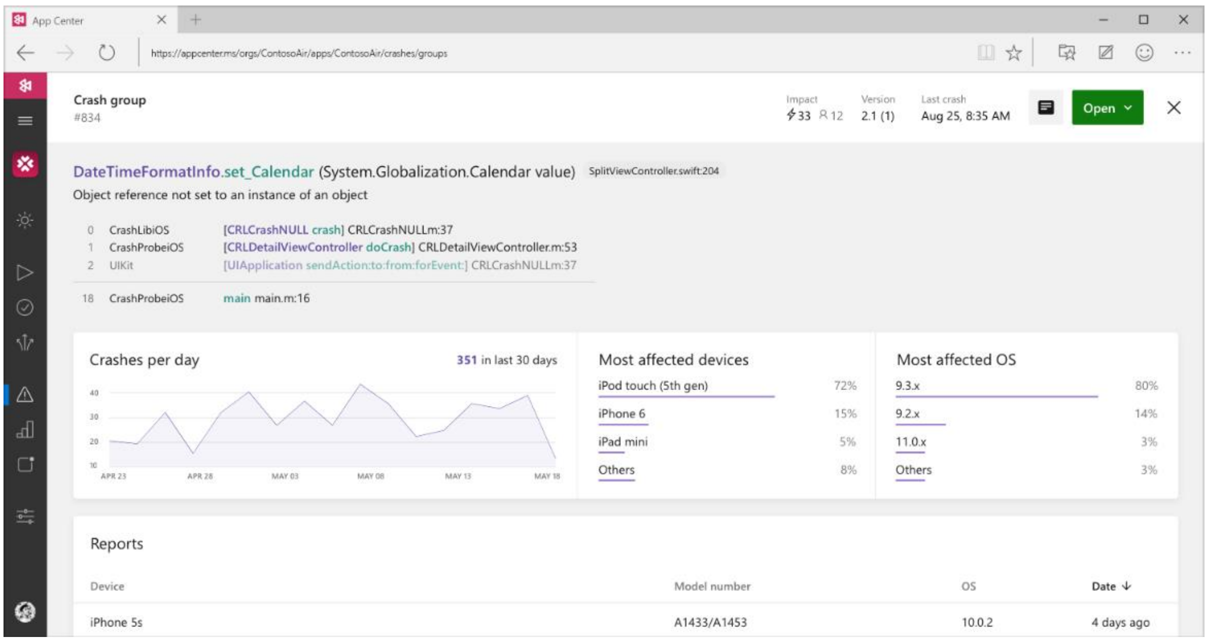
Where would you like to distribute your app?

- iTunes Connect — App Store and TestFlight
- Intune Company Portal

Continuous Learning for Growth

Insightful crash reports

Monitor the health of your app with advanced capabilities such as intelligent crash grouping and management, faster debugging with symbolication, and detailed crash reports. Get notified and fix issues as they come up.


App Center
https://appcenter.ms/orgs/ContosoAir/apps/ContosoAir/crashes/groups
Open

Crash group
#834

Impact ⚡33 👤12 | Version 2.1 (1) | Last crash Aug 25, 8:35 AM

DateTimeFormatInfo.set_Calendar (System.Globalization.Calendar value) SplitViewController.swift:204
Object reference not set to an instance of an object

```
0   CrashLibiOS      [CRLCrashNULL crash] CRLCrashNULLm:37
1   CrashProbeiOS    [CRLDetailViewController doCrash] CRLDetailViewController.m:53
2   UIKit            [UIApplication sendAction:to:from:forEvent:] CRLCrashNULLm:37
18  CrashProbeiOS    main main.m:16
```

Crashes per day — 351 in last 30 days

Most affected devices

iPod touch (5th gen)	72%
iPhone 6	15%
iPad mini	5%
Others	8%

Most affected OS

9.3.x	80%
9.2.x	14%
11.0.x	3%
Others	3%

Reports

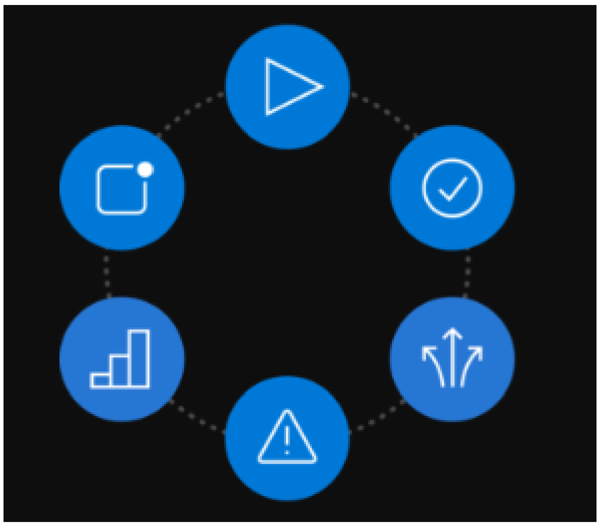
Device	Model number	OS	Date ↓
iPhone 5s	A1433/A1453	10.0.2	4 days ago

Real-time analytics

Grow your audience by focusing on what's important with deep insights about user sessions, top devices, OS versions, behavioral analytics and event trackers for your iOS, Android, Windows, and macOS apps. Easily create custom events to track anything.

Flexibility & choice

Choose only the App Center services you need Have your own analytics solution? Using a different beta distribution platform? No problem. Our open-sourced SDKs and APIs let you integrate just the services of App Center you need.



Continous Delivery Benefits and Challenges

Think about one of your mobile apps… How often do you publish updates to it? A couple of times per year? Quarterly? Monthly, bi-weekly? Every day? Multiple times per day? No! That's absurd, right?

Or is it? Some world-class app engineering teams are able to consistently release their apps weekly or every two weeks at high quality. How do they manage to do that? Part of the explanation is commitment to a continuous delivery process in some form. Let's look at some of the benefits and challenges of continuous delivery, and, more importantly, why this is accessible to everyone. You don't need to be a venture-backed startup or a huge company to do this stuff!

Benefits of Continuous Delivery for Mobile Apps

A continuous delivery process entails being able to release quickly, with high quality, at any given time. Let's contrast this with what we've seen a couple of times. The business wants a critical hotfix pushed out ASAP?

- NO! Fred is on vacation and he's the only guy on the team who can build, sign for distribution, and upload to iTunes.
- NO! We've not gone through the full QA cycle for the current master branch, and it will take three days to go through the manual regression tests, and, by the way, who's got the iPhone 4S running iOS 9.2?
- NO! We're not entirely sure how long the release process takes; reviews could take anything from 5-12 days.

Sound familiar?

With a continuous delivery process, things become easier. The business wants a critical hotfix pushed out ASAP? Not a problem! A developer makes a branch/pull-request with the change and gets peer review. An automated process builds and regression-tests the app using a CI server, test lab, and battery of automated regression tests. The developer merges the change to the master branch and pushes the release button! Continuous Delivery reduces your lead time, getting changes from idea to end-users faster. It reduces risk and variance in the release process because of automation and high repetition.

“Sure,” you might say, "but we don’t have the resources to maintain such an infrastructure." With Visual Studio App Center, this is accessible to everyone via the cloud at low cost.

Challenges for Continuous Delivery

App Store review times, a major barrier to continuous delivery in the past, have been declining steadily since 2015, and are now about one day on average. Still, there remain several challenges with implementing a continuous delivery process. For example, to build iOS apps, you need a Mac-based CI server, which is uncommon in most companies, and configuring this for CI can be time-consuming. With App Center, building in the cloud (even for iOS) takes less than five minutes to set up.

But what about quality? Thorough testing of apps is time-consuming and difficult. There are literally thousands of device models out there, with different and often limited hardware capabilities, screen sizes and resolutions. There are numerous platforms, including Android, iOS, macOS, and Windows, and your app likely supports multiple versions of each.

One way to address the challenge is to rely on outsourced manual testing with providers that offer specialist testers with access to large libraries of different mobile devices. This has benefits, but also drawbacks:

- You need to clearly communicate your requirements.
- You need to interpret the test results.
- The tests are performed by humans, who make mistakes.
- The turnaround for test passes can be slow, ranging from days to weeks.
- Integrating this testing process into an automated Continuous Integration/Continuous Delivery pipeline can be hard.

Instead, what if you had access to a large set of cloud-hosted devices on which you could run automated regression tests of your app with a single line script in your CI system? What if the tests produced visual test reports, so that not only could you perform automated tests, but also be able to review layout and compare differences across device models and OS versions? That’s what Visual Studio App Center Test offers.

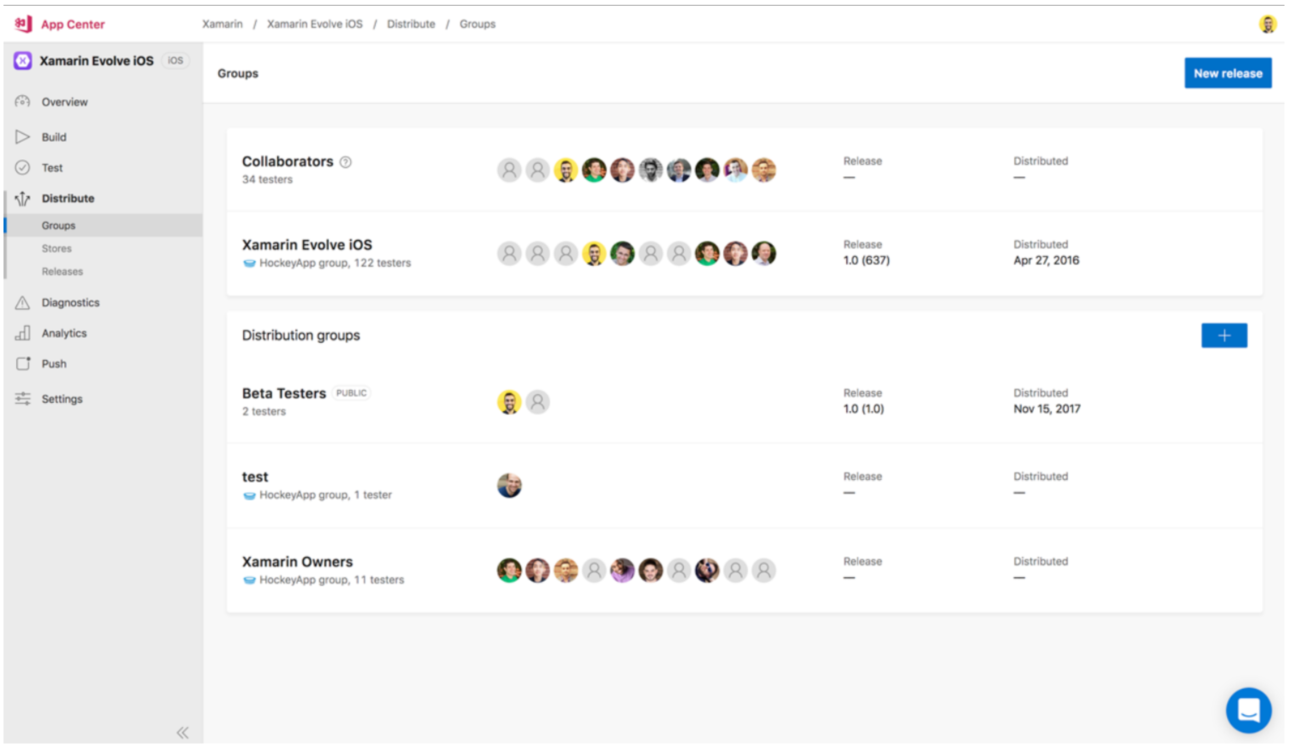
Distribution Groups

Getting Started: How to Extend Continuous Builds to Continuous Tests

Let's just quickly understand the terminology and how to enable it…

Distribution Groups

A distribution group could be based on types of users such as beta testers or internal stakeholders, or based on different stages of the development cycle such as nightly build or staging. Invite testers to your groups with just an email address. When distributing a new release in App Center, you'll select one of your distribution groups as the destination.



	Release	Distributed
Collaborators — 34 testers	—	—
Xamarin Evolve iOS — HockeyApp group, 122 testers	1.0 (637)	Apr 27, 2016

Distribution groups

	Release	Distributed
Beta Testers PUBLIC — 2 testers	1.0 (1.0)	Nov 15, 2017
test — HockeyApp group, 1 tester	—	—
Xamarin Owners — HockeyApp group, 11 testers	—	—

Manage Distribution Groups

Distribution Groups are used to control access to releases. A Distribution Group represents a set of users that can be managed jointly and can have common access to releases. Example of Distribution Groups can be teams of users, like the QA Team or External Beta Testers or can represent stages or rings of releases, such as Staging.

Creating a Distribution Group

To create a new Distribution Group log into the App Center portal, select your desired app, click Distribute in the navigation pane, and lastly select the New Group group button from the top of the screen. Provide a name for the Distribution Group. You can then add users to this group by email. You can also add additional users after the group has been created.

Managing Users in a Distribution Group

Clicking on a Distribution Group will allow for management of the group. You can use the invitation box to add additional users. Or select users from the table to remove them from the group. From this page you can also see the full release history for this Distribution Group by clicking on the releases tab.

It's incredibly easy to add automated UI tests into your App Center builds. App Center has thousands of physical Apple and Android devices on which to run your newly-built app, and App Center supports the most popular UI testing frameworks: Espresso, XCUITest, Appium, and Xamarin.UITest.

Adding Azure Active Directory (AAD) groups to a Distribution Group

In addition to adding testers via email, we now support the addition of AAD groups to a distribution group. Any member of an AAD group can link their organization's subscription to their AAD tenant in App Center. Doing so will enable members of an organization to start adding managed groups to their app's distribution groups. To get started, link your AAD tenant to your organization using the following steps:

- Login to the App Center portal, then select the organization to which you like to link to your AAD tenant
- On the navigator pane that opens, click Manage
- On the azure active directory panel on the Manage page, click the connect button
- You will be redirected to a Microsoft login page, login with your Microsoft/AAD credentials
- You will be redirected to the App Center portal, click the desired tenant you would like to attach.
- At the bottom right of the presented window, click the connect

Once your tenant is connected to the organization, you can add an AAD group as you would an individual tester.

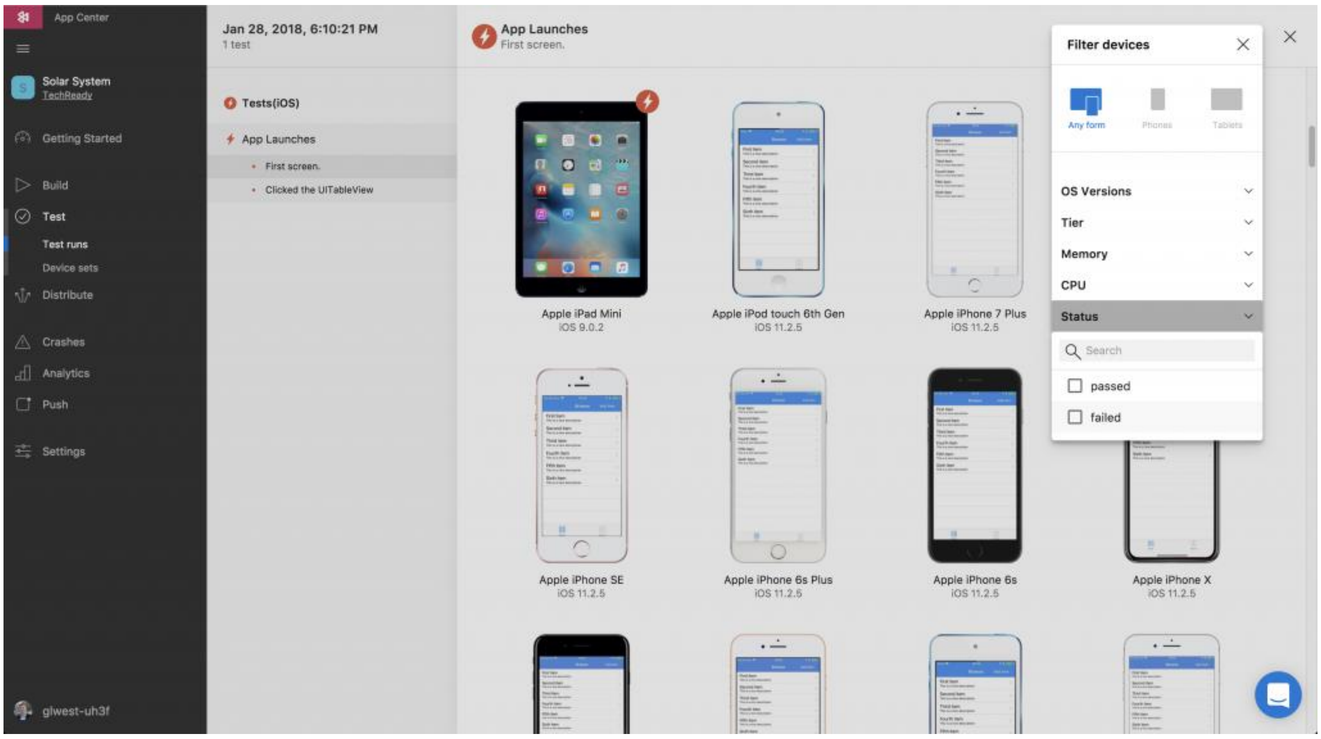
Run a UI Test

To run a UI test:

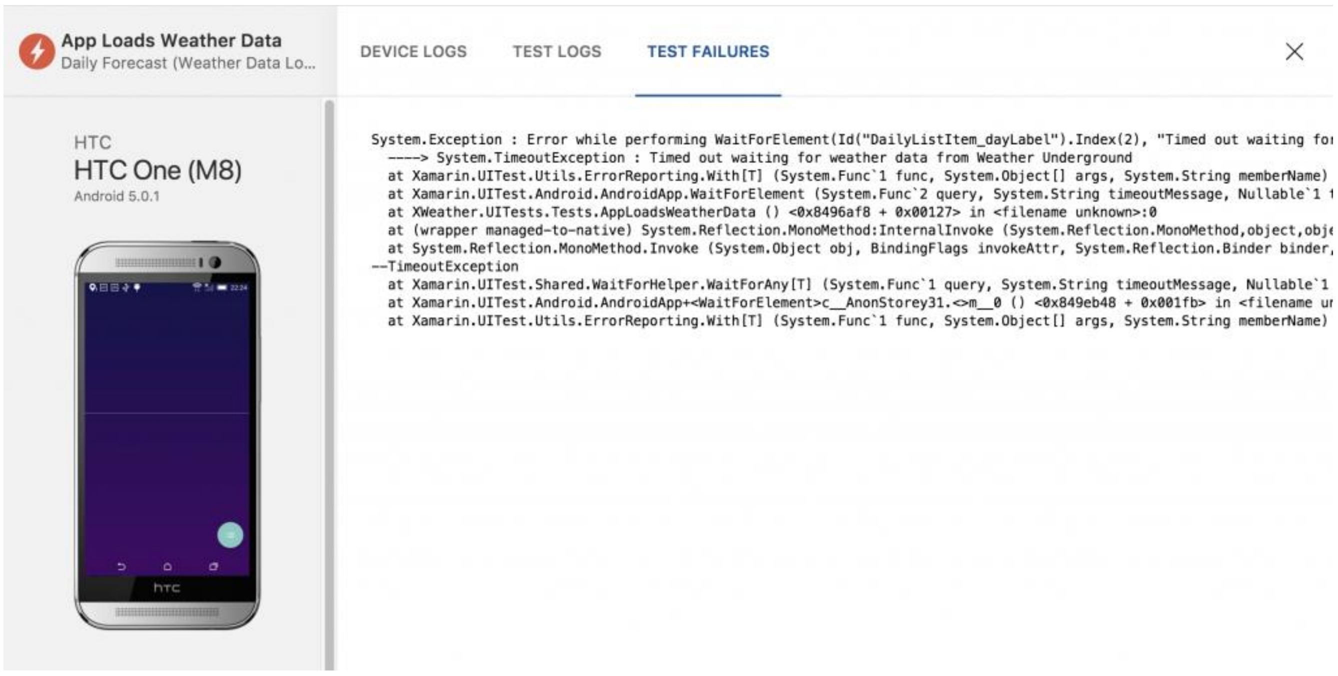
- Using the App Center Test CLI, upload your test suite automatically after completing a build.
- Once you generate a test upload shell command, add it to your app's repo at the solution or project level.

When your test code finishes executing, a detailed report will appear in the App Center Test and the CLI will return a URL linking directly to it. Please see this example command as a reference. Instrumenting your continuous builds with continuous testing is a sure way to shorten the feedback loop, greatly improving app quality.

When the test report is complete, you can easily identify the failed devices by the red "lightning" indicator. Additionally, you can filter just the failed devices or by other parameters by selecting the icon in the upper right corner of the test report.



For every test device, App Center provides device logs, test framework logs, and the line in which the failure occurred to help you easily identify the regression. Start by examining the test failures for clues about where in the test execution the failed assertion occurred. If this is not revealing, dive into the test and device logs for more thorough investigation to determine the root cause of the failure.



```
System.Exception : Error while performing WaitForElement(Id("DailyListItem_dayLabel").Index(2), "Timed out waiting fo
  ----> System.TimeoutException : Timed out waiting for weather data from Weather Underground
  at Xamarin.UITest.Utils.ErrorReporting.With[T] (System.Func`1 func, System.Object[] args, System.String memberName)
  at Xamarin.UITest.Android.AndroidApp.WaitForElement (System.Func`2 query, System.String timeoutMessage, Nullable`1 t
  at XWeather.UITests.Tests.AppLoadsWeatherData () <0x8496af8 + 0x00127> in <filename unknown>:0
  at (wrapper managed-to-native) System.Reflection.MonoMethod:InternalInvoke (System.Reflection.MonoMethod,object,obje
  at System.Reflection.MonoMethod.Invoke (System.Object obj, BindingFlags invokeAttr, System.Reflection.Binder binder,
--TimeoutException
  at Xamarin.UITest.Shared.WaitForHelper.WaitForAny[T] (System.Func`1 query, System.String timeoutMessage, Nullable`1
  at Xamarin.UITest.Android.AndroidApp+<WaitForElement>c__AnonStorey31.<>m__0 () <0x849eb48 + 0x001fb> in <filename un
  at Xamarin.UITest.Utils.ErrorReporting.With[T] (System.Func`1 func, System.Object[] args, System.String memberName)
```

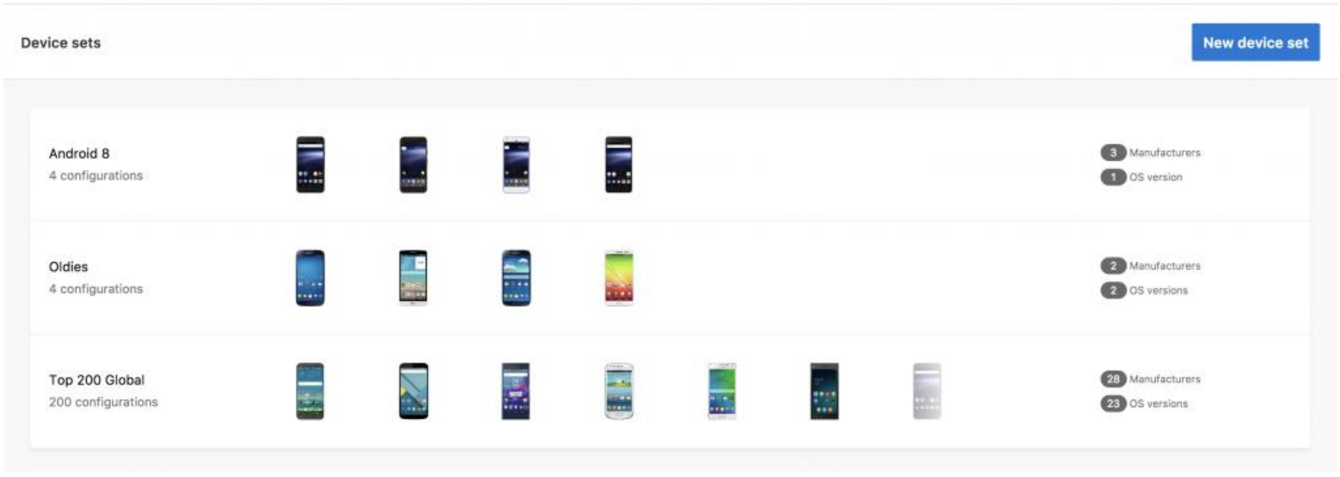
App Distribution Center

App Center Distribution is a tool for developers to release application binaries to their end user devices. The speed and integration capabilities make this tool perform admirably for release of pre-production or beta software to testing devices. App Center Distribution supports the distribution of application packages for iOS, Android, UWP, and macOS.

Here are the top three features to try in App Center Test:

1. Named Device Sets

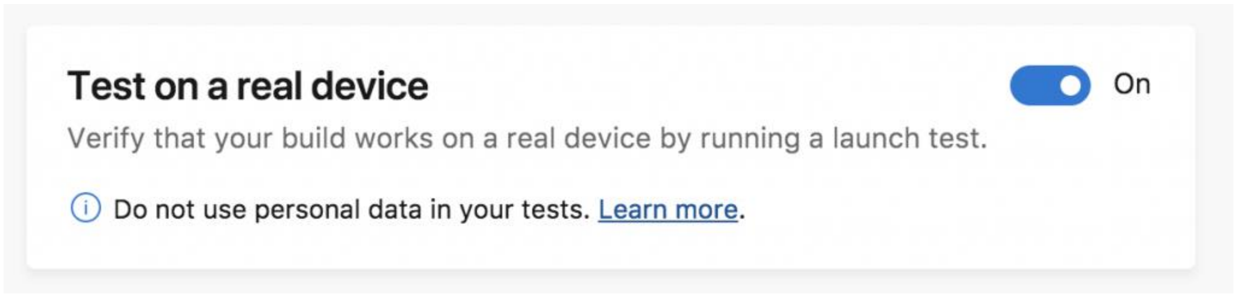
It’s best to run your app on as many devices as possible before shipping to your users. App Center offers named device sets to help you organize and reuse collections of mobile hardware in our labs. Try this out today by adding a “Top 20” set that you can update as you see trends in your users' devices changing.



Android 8 4 configurations	3 Manufacturers	1 OS version
Oldies 4 configurations	2 Manufacturers	2 OS versions
Top 200 Global 200 configurations	28 Manufacturers	23 OS versions

2. Launch Test

To ensure your latest app build runs and opens on our devices before you have any test code written, you can enable launch test in your build configuration. Validate your app’s ability to start on our wide variety of hardware today!



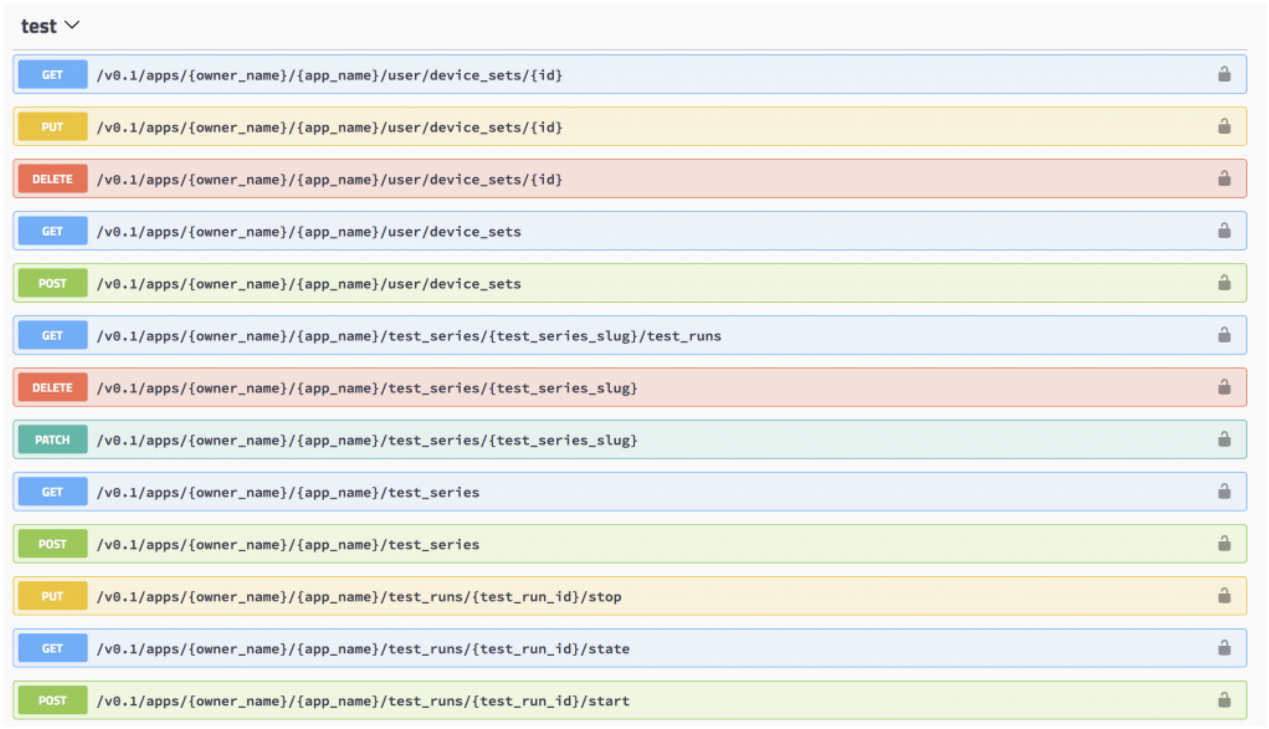
Test on a real device

On

Verify that your build works on a real device by running a launch test.

Do not use personal data in your tests. Learn more.

3. Improved API App Center Test provides a robust REST API giving full access to all data generated during your test run. The data in your test report is accessible via our API, including screenshots, device logs, test runner logs, and failed device specifics. To get started with App Center API today, please see the following documentation.



test

GET	/v0.1/apps/{owner_name}/{app_name}/user/device_sets/{id}
PUT	/v0.1/apps/{owner_name}/{app_name}/user/device_sets/{id}
DELETE	/v0.1/apps/{owner_name}/{app_name}/user/device_sets/{id}
GET	/v0.1/apps/{owner_name}/{app_name}/user/device_sets
POST	/v0.1/apps/{owner_name}/{app_name}/user/device_sets
GET	/v0.1/apps/{owner_name}/{app_name}/test_series/{test_series_slug}/test_runs
DELETE	/v0.1/apps/{owner_name}/{app_name}/test_series/{test_series_slug}
PATCH	/v0.1/apps/{owner_name}/{app_name}/test_series/{test_series_slug}
GET	/v0.1/apps/{owner_name}/{app_name}/test_series
POST	/v0.1/apps/{owner_name}/{app_name}/test_series
PUT	/v0.1/apps/{owner_name}/{app_name}/test_runs/{test_run_id}/stop
GET	/v0.1/apps/{owner_name}/{app_name}/test_runs/{test_run_id}/state
POST	/v0.1/apps/{owner_name}/{app_name}/test_runs/{test_run_id}/start

Manage Target UI Test Device Sets

You can use App Center Build's Launch Test feature to run your latest successful build on a real device using App Center Test. Launching your app after each build validates that your latest build is working.

Configuring your branch

First, the branch needs to be setup before it's ready to run the launch test. You can read more about configuring your branch in the Configure a build respectively for Android and iOS.

Secondly, you need a subscription to App Center Test before you can enable launch test. If you're a first-time user of App Center Test, there's a free trial so you can see how it works.

When you have an active subscription it's time to enable testing, all you’ll need to do is to toggle Run a launch test on a device option to on and click Finish setup button. During the new build, there will be two parts, a build and a test part, so it's normal to have prolonged build time. The benefit is that you know whether your app starts on a physical device.

Finding your launch test result can be done in two ways:

- When the test is completed the app collaborators will receive an e-mail with test results
- Go to Test in the left-hand menu and follow along in the progress

Unsupported configurations

When a build is running, the build configuration is composed of several parts: the build definitions you made in the code or in the IDE, and the settings you have chosen in App Center. The way configurations work is platform and language specific. Below are some known configurations where launch test is not supported.

- Some Android configurations can create several APK files. Which makes it unclear to the build service which binary to test. If you disable Generate one package(.apk) per selected ABI in your Android Build options, only one APK is created.
- Simulator builds for Xamarin apps don't output a binary executable file, and therefore can't be tested. This is the case for both Android and iOS.
- Android apps without Internet permissions cannot be tested using App Center Test. Internet permissions are needed for the test infrastructure to communicate with your app. In order to use launch test for your Android project, make sure to declare internet permissions in your app manifest
- iOS unsigned builds don't generate an .ipa file and they can't be tested on a real device.

Provision Tester Devices for Deployment

Getting your app onto your own device, or in the hands of potential testers, can be one of the most time-consuming challenges when developing an iOS app. To install an app on a single device, you must register the device with Apple and sign the app with the relevant provisioning profile. While this isn’t much of an issue for a single device, you need to register and re-sign the app each time you install the app on a new set of devices. For a developer, this repetitive action is a huge loss of time; for a tester, it’s a disruptive experience to receive an app and not be able to install it because your device hasn’t been provisioned, yet.

App Center supports the release of auto-provisioning capabilities, this enables iOS developers to spend more time creating and shipping great apps, rather than managing device provisioning. With the auto-provisioning capability, you no longer need to know the device IDs of testers, coworkers, or stakeholders when building a new beta version of your app for testing. App Center integrates this capability directly into the install portal, so you can automate the distribution process and enable testers and team members to install your app with one click.

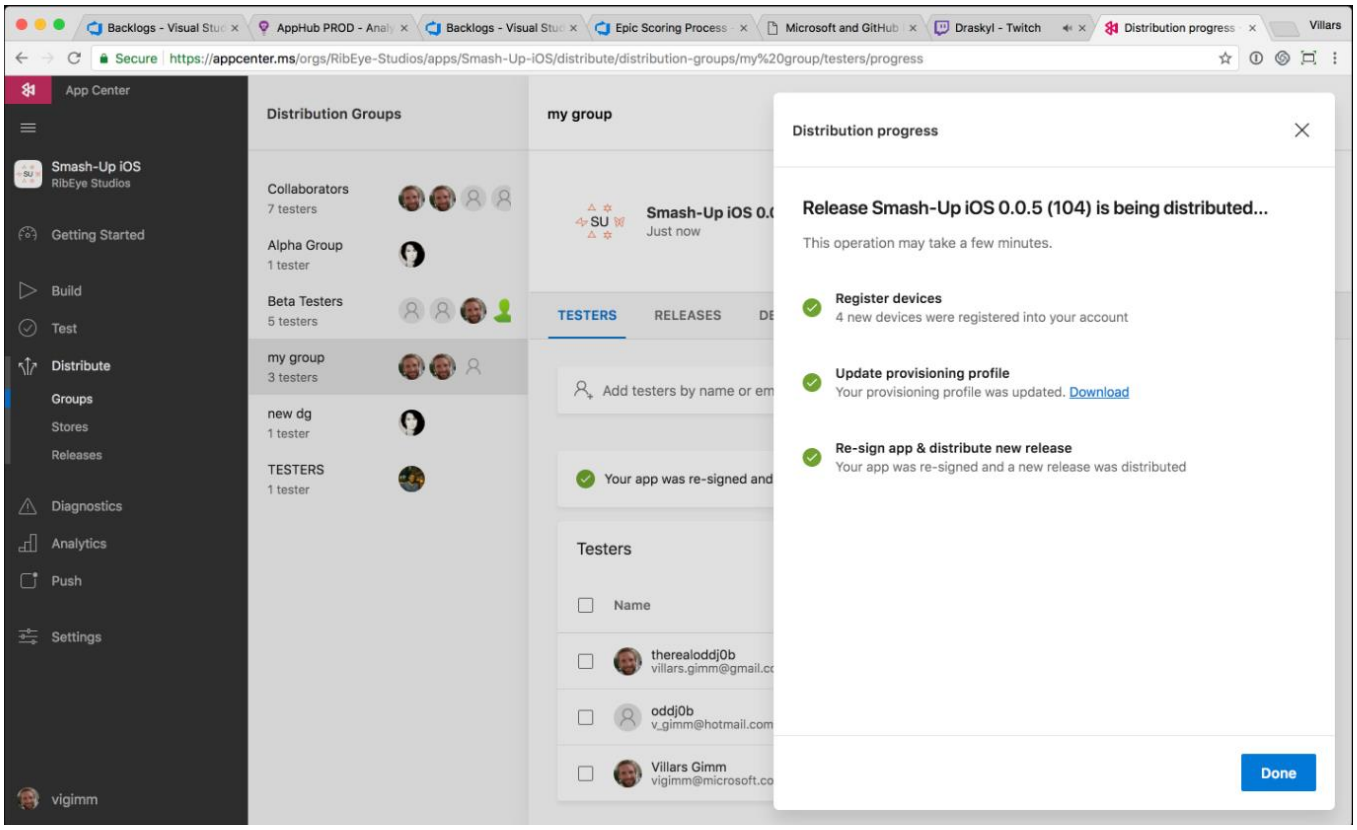
Setting Up Auto-Provisioning

All you need to get started with auto-provisioning is an ad-hoc provisioning profile and a production certificate, both of which are created in the Apple Developer portal, to produce a signed app ready for our distribution service.

Once you have an .ipa, select your app in App Center and navigate to the distribution service in the App Center portal to choose the distribution group you want to set up. Click on your distribution group settings and turn on Automatically manage devices. You'll need your Apple ID credentials and your distribution certificate to complete this step.

Now that you've turned on the Automatically manage devices setting, next time you release a new version of your app using the App Center's Build service or by manually uploading the app binary, App Center will automate the process for you and run the following steps without any action from you or your testers:

- Log into your developer portal.
- Register new device IDs.
- Add them to the app's provisioning profiles.
- Download all updated profiles.
- Re-sign the binary using the selected certificate and the updated profiles.
- Enable all testers in the distribution group to install your app.



Distribution progress

Release Smash-Up iOS 0.0.5 (104) is being distributed...

This operation may take a few minutes.

Register devices
4 new devices were registered into your account

Update provisioning profile
Your provisioning profile was updated. Download

Re-sign app & distribute new release
Your app was re-signed and a new release was distributed

Every time you release a new version of your build, testers will get an email notification to install the app and start using it.

Types of Distribution Groups

Private Distribution Groups

In App Center, distribution groups are private by default. This means only testers invited via email can access the releases available to this group. Testers added to this group will receive a notification that they have been added to the app to test. After a release, testers that were previously added to this group will receive the new release notification email and will be required to login to their App Center account in order to access and download releases.

Public Distribution Groups

Distribution groups must be public to enable unauthenticated installs from public links. When creating a new distribution group, the options is available during the setup process. After giving your group a name, you can enable Allow public access.

To make an existing distribution group public, open the distribution group and click on the settings icon in the upper right-hand corner of the screen. From the settings modal, you can enable Allow public access.

As with private distribution groups, testers will receive an email notifying them that they've been invited to test the app and when a release is available to them. In order to access the app from here, testers will be required to login with their App Center account.

In addition to this, a public download link is displayed underneath the distribution group name at the top of the distribution group page. Anyone, including testers who aren't explicitly added to the distribution group, can access the release without signing in using the public download link.

Shared Distribution Group Shared distribution groups are private or public distribution groups that are shared across multiple apps in a single organization. Shared distribution groups eliminate the need to replicate distribution groups across multiple apps. With a few clicks, you can give a shared distribution group access to any combination of apps belonging to a particular organization.

Unlike traditional public and private distribution groups, shared distribution groups are at the organization level rather than the app level. Due to this, the steps to create one are slightly different:

1. Login to the App Center portal, then select the organization to which you would like to add a shared group
2. On the navigator pane that opens, click People
3. On the People page, click the distribution group link to open the shared distribution groups page
4. On the top right corner of the shared distribution groups page, select the blue create new group button.
5. Once you have a distribution group, click the group entry in the table to add testers and apps to the group.

To add testers to your new shared distribution group, select the Testers tab and enter the emails of the desired testers.

To add apps to your new shared distribution group, select the Apps tab and enter the name of the desired apps that this group should have access to.

Other Considerations

Automatically manage devices

When releasing an iOS app signed with an ad-hoc or development provisioning profile, you must obtain tester's device IDs (UDIDs), and add them to the provisioning profile before compiling a release. When you enable the distribution group's Automatically manage devices setting, App Center automates the before mentioned operations and removes the constraint for you to perform any manual tasks. As part of automating the workflow, you must provide the user name and password for your Apple ID and your production certificate in a .p12 format.

App Center starts the automated tasks when you distribute a new release or one of your testers registers a new device. First, all devices from the target distribution group will be registered, using your Apple ID, in your developer portal and all provisioning profiles used in the app will be generated with both new and existing device ID. Afterward, the newly generated provisioning profiles are downloaded to App Center servers. Second, App Center uses your saved certificate and the downloaded provisioning profiles to re-sign your app so it can be installed on any device in the releases distribution group. Your testers will not have to wait for you to create a new release as the existing release is being updated and a download link will be available through the App Center Install portal.

Releasing a Build to a Distribution Group

To release a new build to a Distribution Group. Make use of the Distribute new Release button at the top of the screen and choose the Distribution Group from the list during the steps.

User Download Analytics

User download analytics allow you to monitor how testers within a distribution group are interacting with a release. Total and unique download counts for a release are available both at the top of each distribution group page as well as on the Release Details page. The total download count represents the total number of times the Install button has been hit by a tester. The unique download count represents the number of users that have installed the release. As an example, if one tester has downloaded a release to two different test devices, this would equal two total downloads and one unique download. Note that public distribution groups will only include the total download count and not the unique download count.