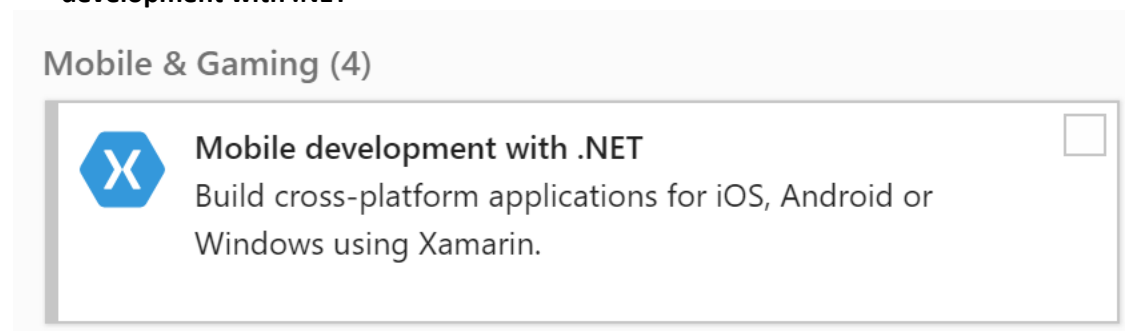


App Center Workshop

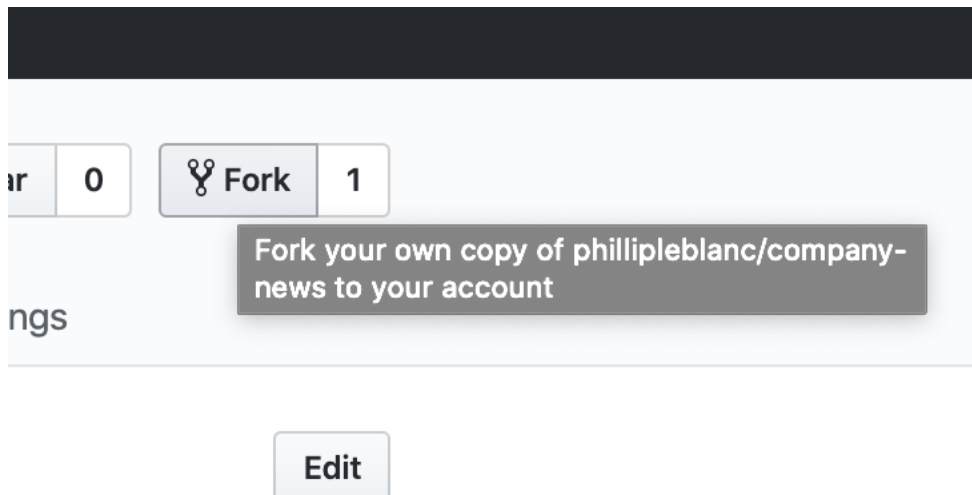
Pre-requisites	1
Module 1: Setting up an App Center app	3
Module 2: Build & Launch Test	7
Module 3: Distribution	13
Module 4: Analytics & Diagnostics	19
Module 5: Push	21

Pre-requisites

1. [Visual Studio 2019](#) – during installation you need to select the component **Mobile development with .NET**



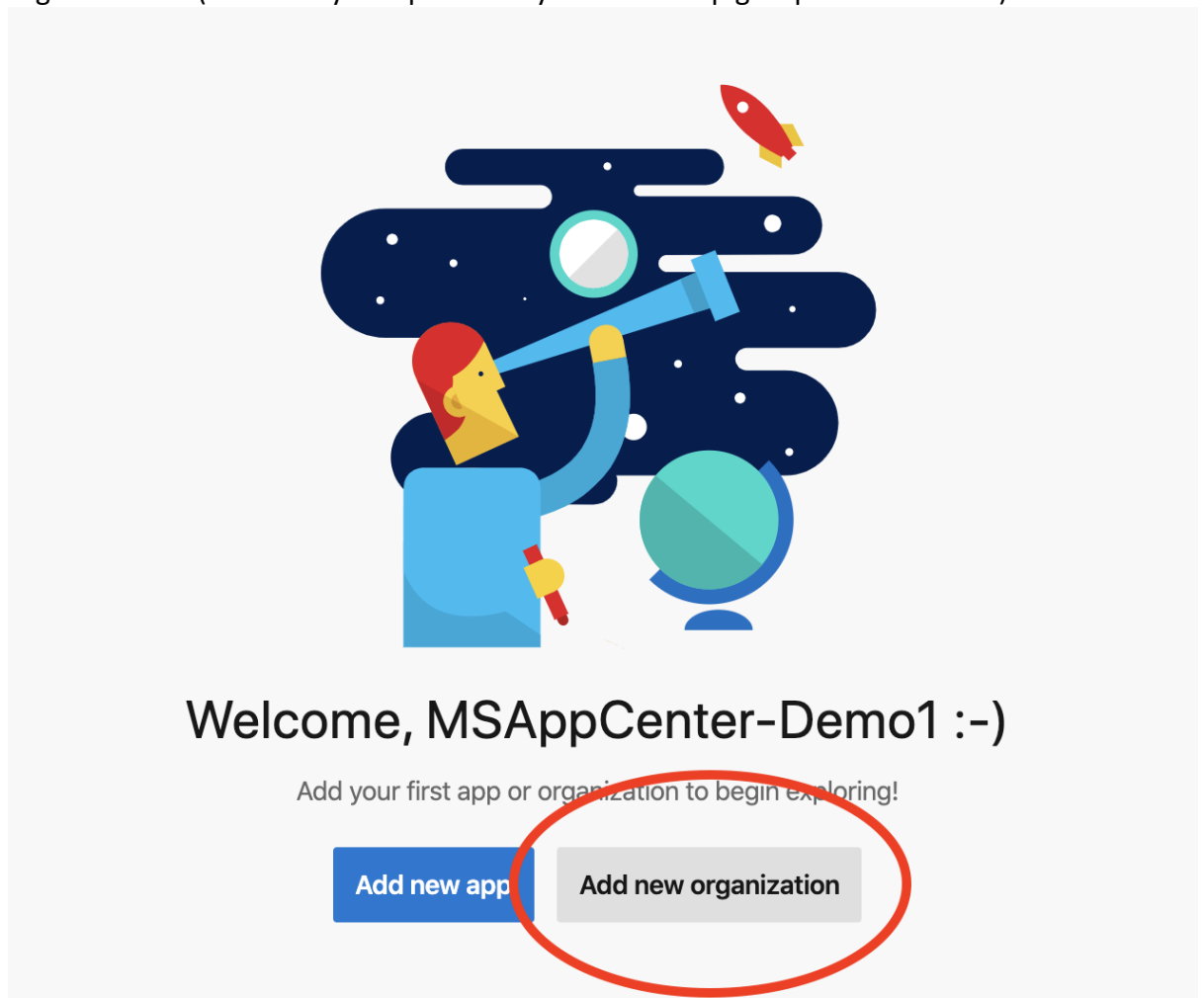
2. A GitHub account, you can sign up for one here: <https://github.com>
3. An App Center account, you can sign up for one here: <https://appcenter.ms>
4. Git for Windows - <https://git-scm.com/download/win>
5. Using your GitHub account, fork this repository: <https://github.com/phillipleblanc/company-news>



6. Open Visual Studio, login to the GitHub account and clone the repo locally.
7. Build the Solution, this should restore all the nuget packages.
8. Set the CompanyNews.Android project as the Startup project (right click, Set as Startup Project)
9. Run the project locally. This will ask you to create an emulator.

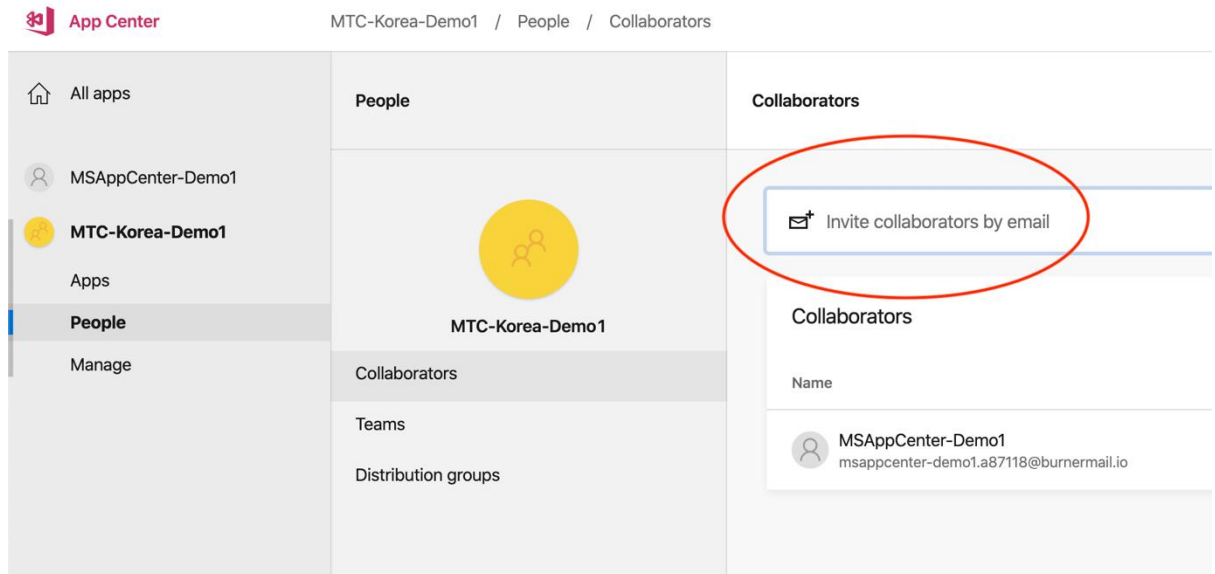
Module 1: Setting up an App Center app

1. Login to App Center, you will see a page that looks like the below. Click on “Add new organization”. (Note: Only one person in your workshop group should do this)



2. Give the organization a name. Note that App Center organization names are unique, so you can't have the same name as another organization. For this workshop, I have named mine "MTC Korea Demo1".

3. Navigate to the Collaborators section and invite everyone else from your workshop group.



4. Navigate to the Apps section in the organization and click "Add an app". Use the following fields for the app:
 - a. App name: CompanyNews.Android
 - b. Release Type: Beta
 - c. OS: Android
 - d. Platform: Xamarin

5. It should look like this:

Add new app



App name:

Icon:

CompanyNews.Android



Release Type:

Owner:

Beta



MTC-Korea-Demo1

OS:

☐

iOS

☒

Android

☐

Windows

☐

macOS

Preview

☐

tvOS

Platform:

☐

Java / Kotlin

☐

React Native

☐

Cordova

Preview

☒

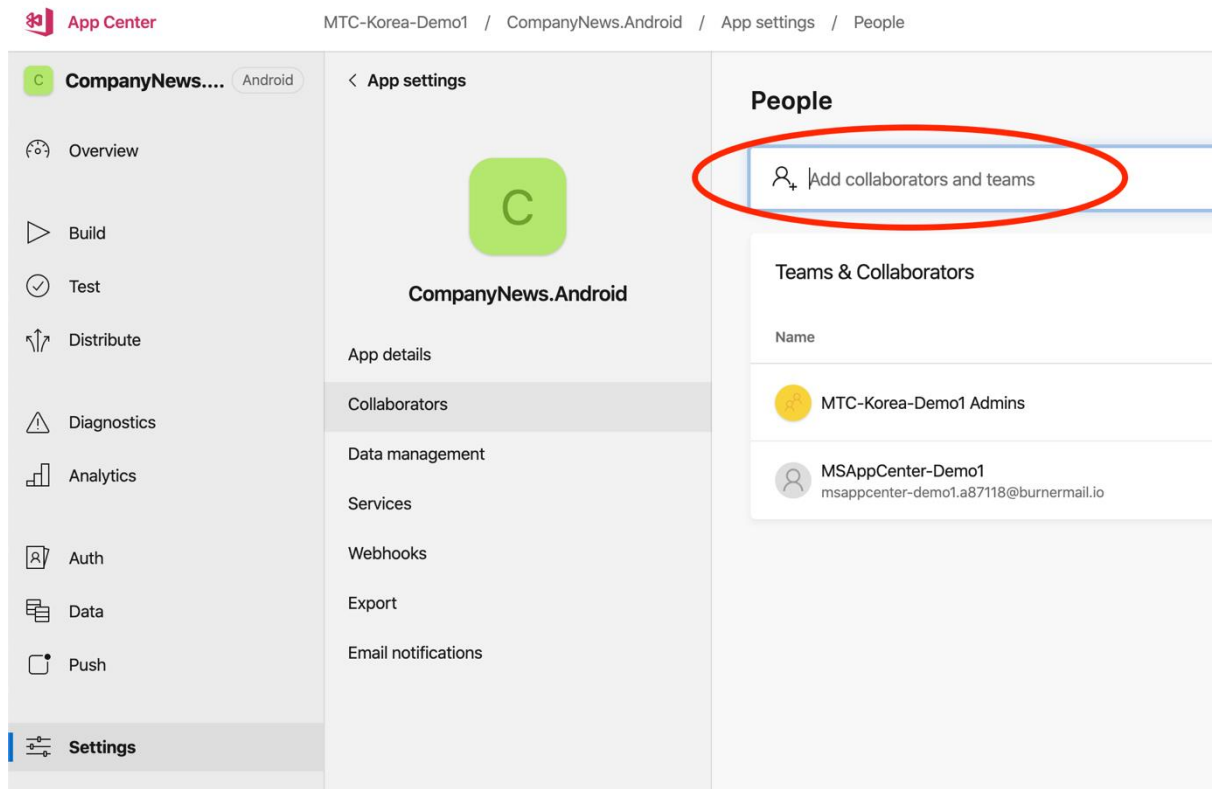
Xamarin

☐

Unity

6. By default, only the admins of the organization and the person who created the app can see this app. To add the rest of your workshop group, navigate to the app

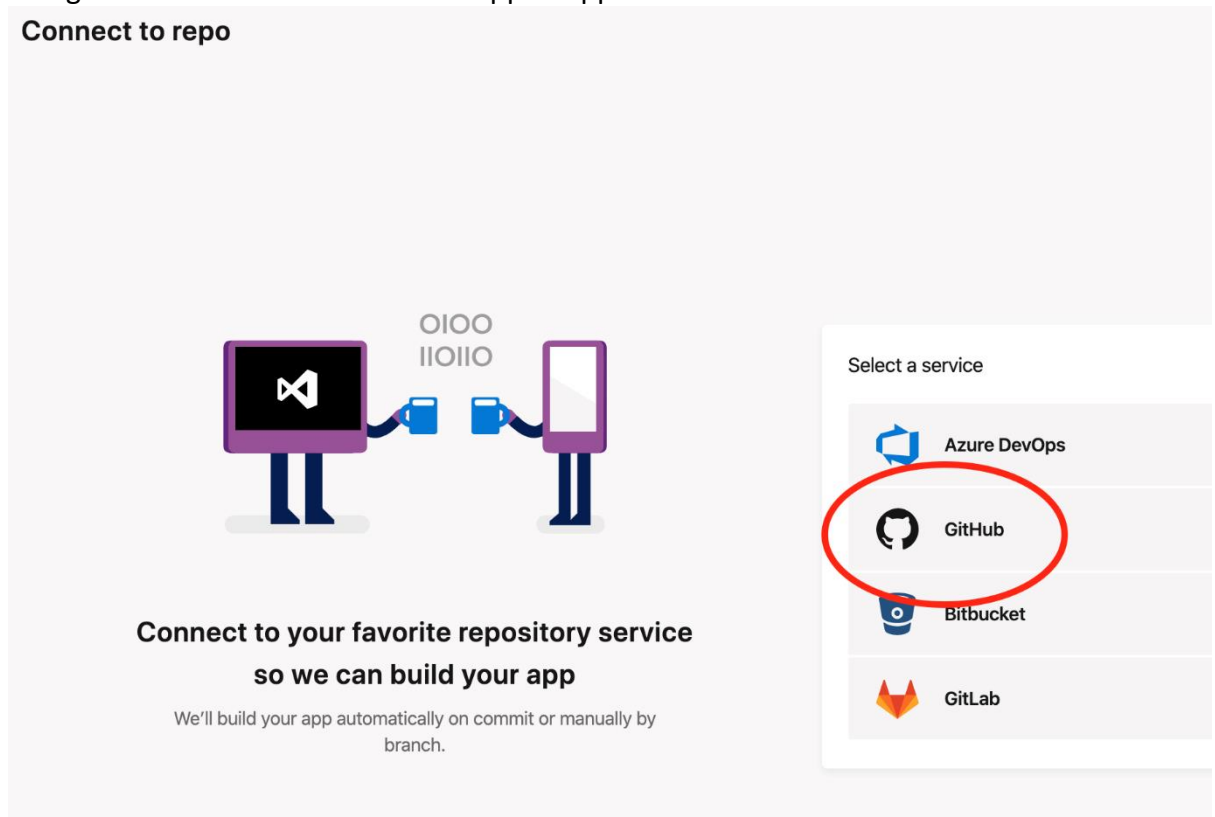
settings and add them as “Collaborators”.



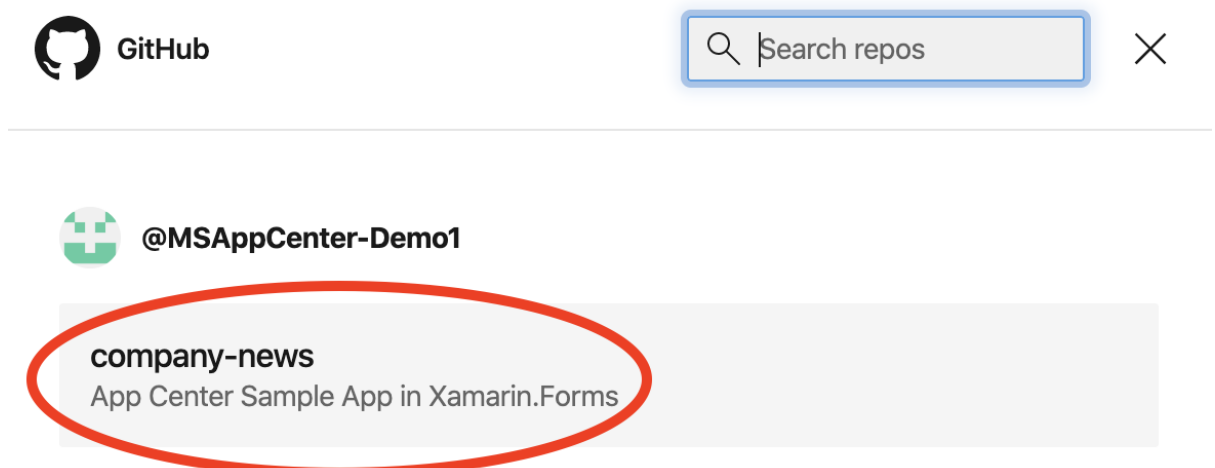
7. Congratulations! You have completed Module 1! We now have an app in App Center that represents our Company News app that we will be working on for this workshop. Move onto the next module to configure Build & Distribution.

Module 2: Build & Launch Test


1. Navigate to the Build section of the app in App Center and click the GitHub button.



2. Click on the company-news repo from your account to set-up the connection.




3. Click on the master branch, and then on configure build.



This branch has not been configured to build yet.

It sure looks intriguing, though!

LAST COMMIT



Changing commented code to remove app secret

Phillip LeBlanc

1 hour ago

Configure build

4. You can investigate the various options available in the build configuration, but for now we will just click Save & Build. We will revisit these options later.

Build configuration
master

Build app

Project: CompanyNews.Android.csproj

Configuration: Debug

SDK version: Xamarin.Android 10.0

Build scripts: ☒ Pre-build

Build frequency: ☒ Build this branch on every push

Build Android App Bundle: Off

Automatically increment version code: Off

Environment variables

Save

Save & Build

5. You will see a live view of the cloud build agent working to build the app. After a few minutes you will see that it has completed successfully.

< master

Changing commented code to rem... Phillip LeBlanc

Build 1
Manual build

DURATION
1 min 35 sec

LAUNCH TESTED
N/A

SIGNED
No

6. However, in order to distribute this app to real devices it needs to be signed by a keystore. So, let's create that keystore now. Open PowerShell on your machine as Administrator and find the path of the latest Java SDK or Runtime by running the command:

```
ls 'C:\Program Files (x86)\Java'
```

Once you find the name of the latest Java SDK, you can construct the path to the keystore tool we will use. Pass these arguments:

```
-genkeypair -v -keystore C:\company-news.keystore -alias key0  
keyalg RSA -keysize 2028 -validity 10000
```

An example of calling this with the full path is this:

```
C:\Users\phleblan> & 'C:\Program Files  
(x86)\Java\jre1.8.0_211\bin\keytool.exe' -genkeypair -v -  
keystore C:\company-news.keystore -alias key0 -keyalg RSA -  
keysize 2048 -validity 10000
```

```
C:\Users\phleblan> ls 'C:\Program Files (x86)\Java'
```

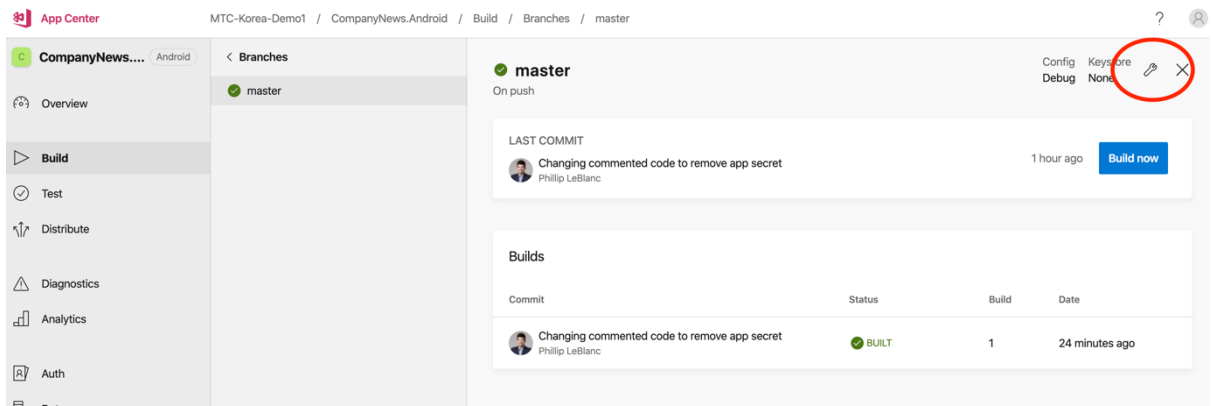
Directory: C:\Program Files (x86)\Java

Mode	LastWriteTime	Length	Name
d----	3/5/2019 11:11 AM		jre1.8.0_201
d-----	5/24/2019 11:17 AM		jre1.8.0_211

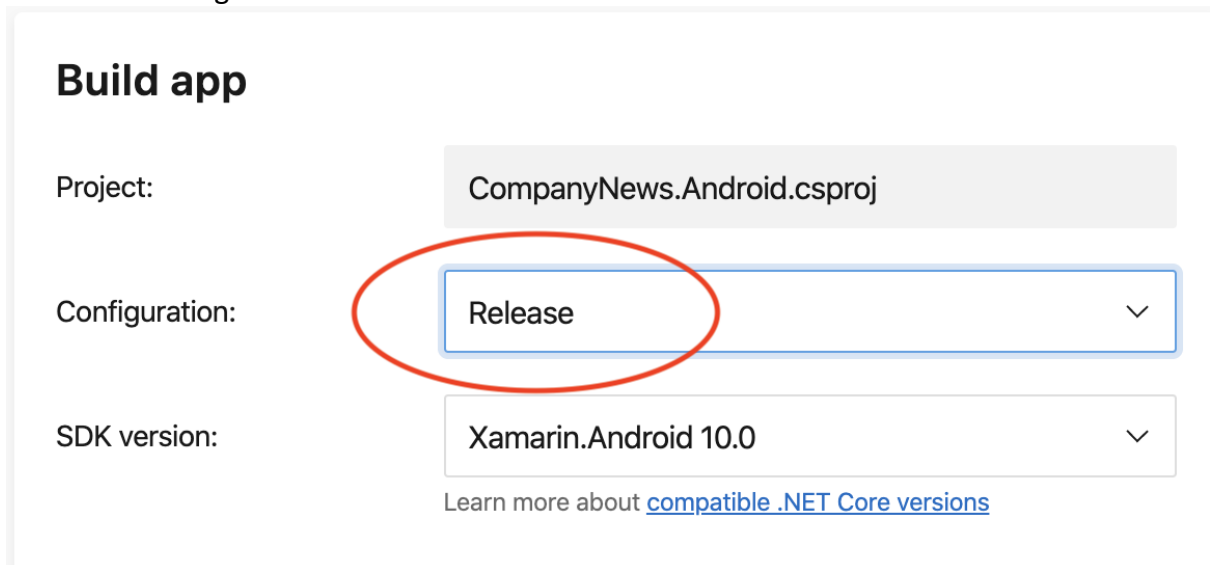
```
C:\Users\phleblan> & 'C:\Program Files (x86)\Java\jre1.8.0_211\bin\keytool.exe' -genkeypair -v  
-keystore company-news.keystore -alias key0 -keyalg RSA -keysize 2048 -validity 10000  
Enter keystore password:  
Re-enter new password:  
What is your first and last name?  
[Unknown]: Phillip LeBlanc  
What is the name of your organizational unit?  
[Unknown]:  
What is the name of your organization?  
[Unknown]:  
What is the name of your City or Locality?  
[Unknown]:  
What is the name of your State or Province?  
[Unknown]:  
What is the two-letter country code for this unit?  
[Unknown]:  
Is CN=Phillip LeBlanc, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown correct?  
[no]: yes  
  
Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity  
of 10,000 days  
for: CN=Phillip LeBlanc, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown  
Enter key password for <key0>  
(RETURN if same as keystore password):  
[Storing company-news.keystore]  
  
Warning:  
The JKS keystore uses a proprietary format. It is recommended to migrate to PKCS12 which is an  
industry standard format using "keytool -importkeystore -srckeystore company-news.keystore -d  
estkeystore company-news.keystore -deststoretype pkcs12".
```

Remember what value you used for the password, as you will need it when configuring the build.

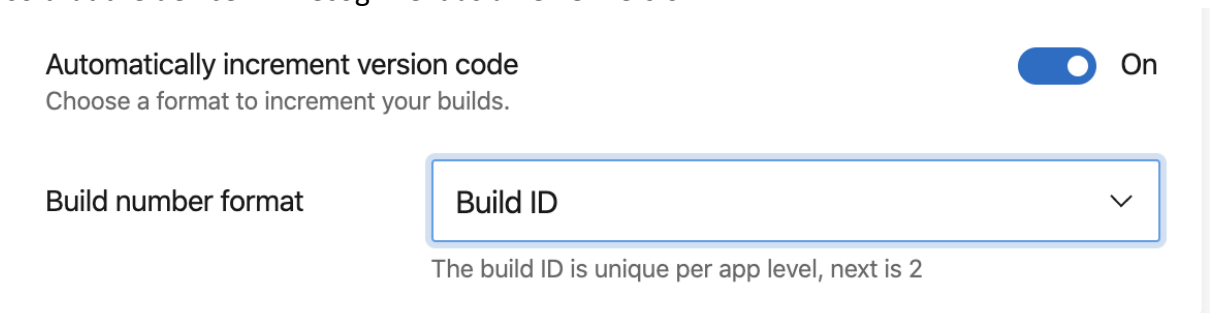
7. Now with the keystore created, we can add this to the App Center Build configuration page. Navigate back to the configuration page for this branch.



8. Switch the configuration to “Release”.



9. In the build details page, configure “Automatically increment version code”. This will tell App Center build to automatically bump the build version number on each build, so that the device will recognize it as a newer version.



10. Scroll down and configure the signing section by uploading the company-news.keystore file we generated as well as filling in the password and key alias. (The


key alias is key0, the password is whatever you set during creation)

Sign builds

☒ On

Builds must be signed to run on devices.

Keystore

 **Keystore file:**
company-news.keystore

×

Environment variables

Keystore password:

.....

Key alias:

....

Key password:

.....


11. And finally, enable the Launch Test. This will run this build on a real device from our App Center Test service to see that the app will launch properly. (Note: This does add a few minutes to the build).


Test on a real device Free ☒ On

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

[Do not use personal data in your tests](#)

12. Click Save & Build. After a few minutes the build will be completed, and the launch test finished. You can click on the “Results” link from the Launch Test to view the test run.



Changing commented code to remove app secret
 Phillip LeBlanc


Build 2
 Manual build

DURATION
 5 min 47 sec

LAUNCH TESTED
[Results](#)


SIGNED
 Yes



 Oct 24, 2019, 2:01:06 PM
 1 test

VERSION
 1.0 (2)

DEVICES
 1

RUN TIME
 3 min


100%
 1/1
 Test passed


100%
 1/1
 Device passed

Feature	Peak duration	Peak memory	Status
✓ Tests	20.00 sec	130.79 MB	Passed
✓ App Does Launch	20.00 sec	130.79 MB	Passed

13. Congratulations! You have completed the Build Module and learned how to configure App Center to build a signed app that you can distribute to devices. Now let's do that!

Module 3: Distribution

- In the previous module, we configured a launch test during build to demonstrate the functionality, but since it does add several minutes to the build let's go ahead and uncheck that for now. Navigate to the build configuration and uncheck the Launch Test setting. Click Save.

Test on a real device
Free
☐ Off

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

- Navigate to the Distribute section in App Center and click on Groups. You will see a pre-created group called "Collaborators" which automatically includes all the

collaborators from your app. Let's go ahead and create a new group and call it "Beta Testers". Add the testers to this group, include everyone from your workshop group. Note that testers can be people who are not Collaborators on the app, any email address will work.

New distribution group



Group name:

Beta Testers

Allow public access



Off

Allow anyone to download

Who would you like to invite to the group?

 Add testers

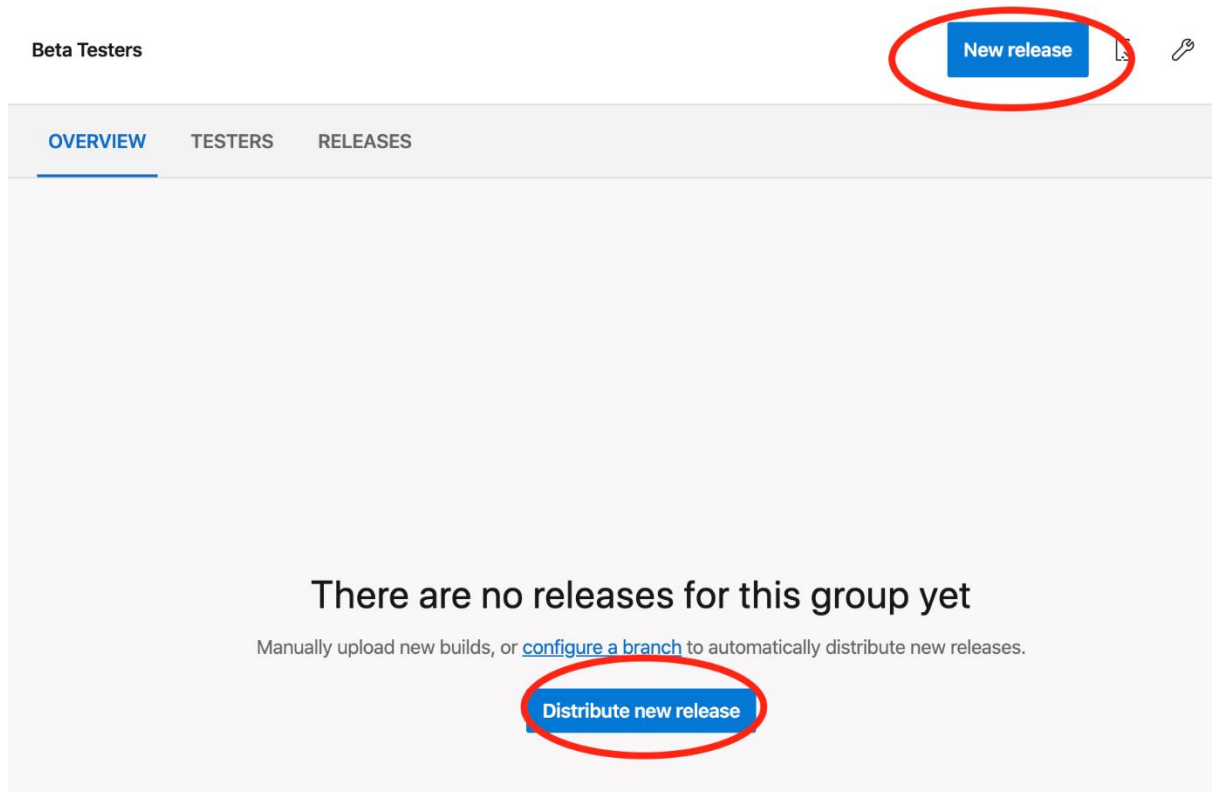


MSAppCenter-Demo1



Create Group

3. In the group screen, either click on “New Release” in the top right, or “Distribute new release”.



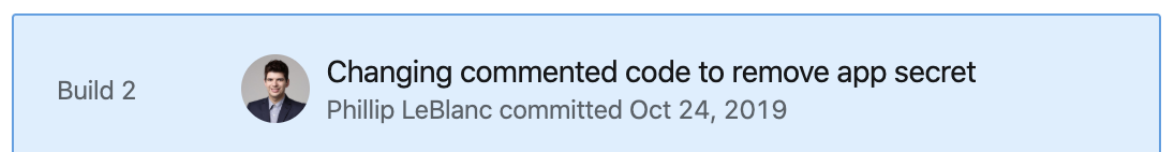
4. In the wizard that shows up you can choose to upload an .apk that you have already pre-built from your machine. But since we already have a build from App Center click on “Distribute an Existing Build” in the bottom left.



5. It should automatically find the master branch build that we did before, click on it and click next.
6. Choose the latest build from this branch and click Next.

Choose a build from `master` branch

Only the 5 most recent signed and successful non-simulator builds available for distribution are shown here.



7. On the next page, you can enter release notes that your testers will see. By default, it is pre-filled with the latest commit message. You can change this or leave it as is and press Next.
8. The last screen will let you review what is about to happen. You also have the option to make this a mandatory update and to not notify testers. We will leave those unchecked for now and click Distribute.

New release



1 Branch 2 Build 3 Notes 4 Review

Distributing release



Build 2:
master branch

- ☐ Mandatory update ⓘ
- ☐ Do not notify testers

Releasing to 2 testers

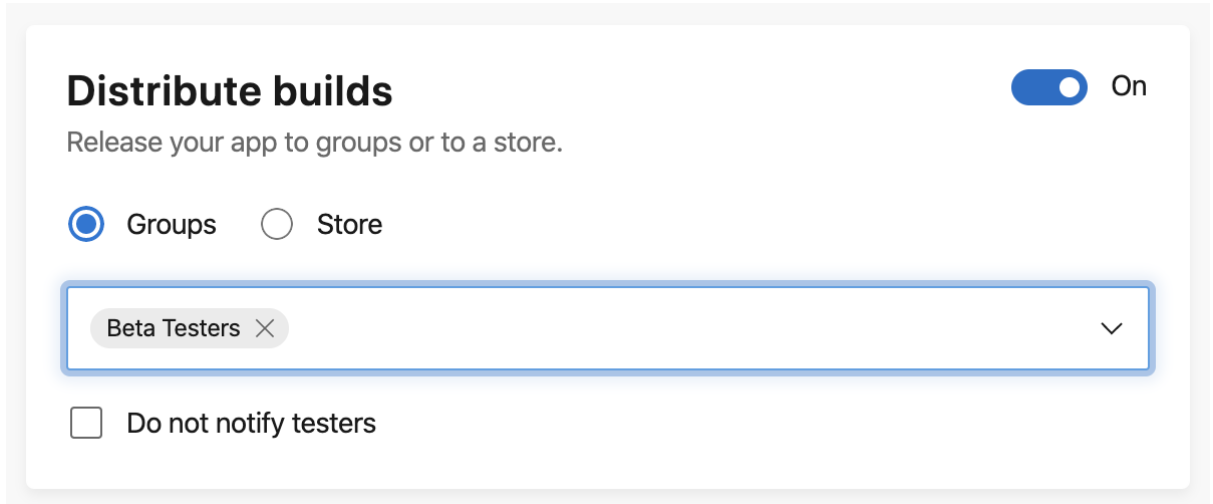


Beta Testers
Distribution group

2 testers

9. At this point, you should now receive an email from App Center along with a link to install this app. Using your Android device, open the email and click the link. Sign into App Center and then click the “Download” button. You may need to allow your phone to install from untrusted sources, but then the app will be running on your phone! 🎉
10. Now let’s modify our build settings to automatically distribute new builds whenever they are ready. Navigate back to the build configuration and in the section

“Distribute builds” configure the “Beta Testers” destination.



Distribute builds On

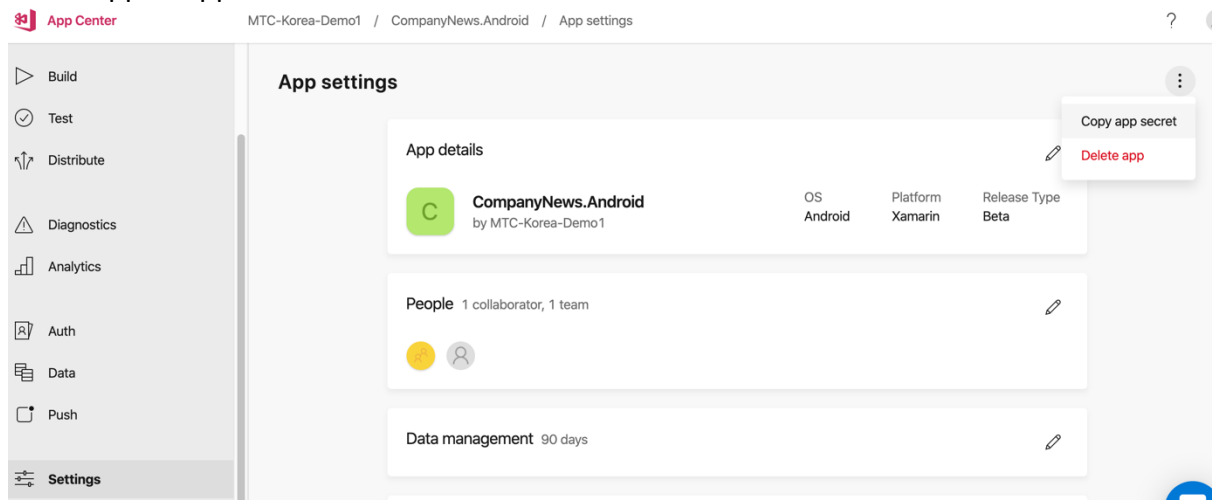
Release your app to groups or to a store.

☒ Groups ☐ Store

Beta Testers X

☐ Do not notify testers

11. Click Save & Build and in a few minutes the app will be built, a new email will be sent, and you can upgrade the installed app on your phone with this new build.
12. Now let's enable the App Center SDK integration. The App Center SDK allows access to some powerful diagnostics and analytics capabilities, as well as being able to configure in-app updates. Let's demonstrate in-app updates now. Navigate to the app settings and copy the app secret. This app secret is what allows the SDK to know which app in App Center to associate with.



13. Using the app secret that you have copied go back to the code. Open Visual Studio and navigate to the file MainActivity.cs in the CompanyNews.Android project. Find this commented line of code:

```
// APPCENTER
//AppCenter.Start("<app secret goes here>",
//    typeof(Analytics), typeof(Crashes), typeof(Push), typeof(Distribute));
```

And un-comment it by removing the two slashes at the front. Replace the text in “<app secret goes here>” with the app secret that you copied from the App Center portal. This is what my code looks like after doing this process (remember the App Secret will be different for your code):

```
// APPCENTER
AppCenter.Start("abaeed36-bd3a-4262-a2fe-c49a63c4fb2a",
    typeof(Analytics), typeof(Crashes), typeof(Push), typeof(Distribute));
```

Because the App Center SDK is modular, you can choose which services to integrate. We will integrate the Analytics, Crashes, Push and Distribute services in this app.

14. Build the project in Visual Studio to ensure there are no build issues, then commit these changes and push to your GitHub repo. This will automatically kick-off a new build and distribution.
15. After opening the app from these changes, you will notice a pop-up that visited the App Center site. This was to enable in-app updates. To demonstrate that, let's make another change to the app. Open the file MainPage.xaml.cs in the CompanyNews project. Find the below code blocks and uncomment them to enable additional App Center integration.

Code Block 1: Tracking a custom event when a user taps on the news item:

```
void OnListViewItemSelected(object sender, SelectedItemChangedEventArgs e)
{
    NewsItem selectedItem = e.SelectedItem as NewsItem;

    var properties = new Dictionary<string, string> {
        { "Title", selectedItem.Title },
        { "ContentBrief", selectedItem.ContentBrief }
    };

    // APPCENTER
    // Analytics.TrackEvent("News Item Selected", properties);
}
```

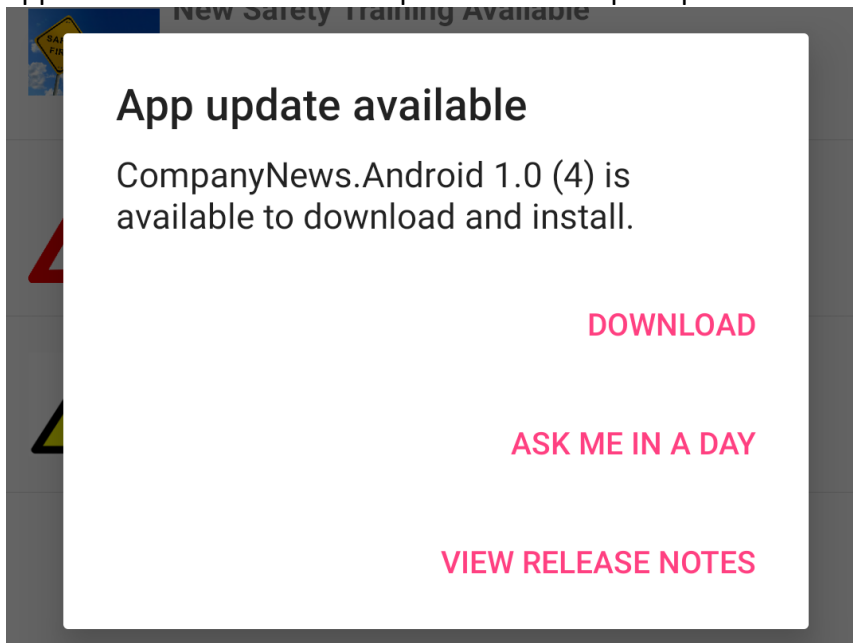
Code Block 2: Tracking an error that doesn't cause the app to crash, but is still something we want to be aware of:

```
if (tappedItem.ShouldHandledError)
{
    try
    {
        throw new InvalidOperationException("Handled Error");
    }
    catch (InvalidOperationException ex)
    {
        var properties = new Dictionary<string, string> {
            { "Title", tappedItem.Title },
            { "ContentBrief", tappedItem.ContentBrief }
        };

        // APPCENTER
        // Crashes.TrackError(ex, properties);
        return;
    }
}
```

After uncommenting both of those blocks, commit and push the changes.

16. App Center will automatically start building and distributing this app as before. However, instead of installing the app by clicking on a link in the email, all you need to do is open the app and see a prompt that a new build is available inside of the app itself. This is extremely useful, as it prompts users to install the latest version of the app even if they aren't looking at their email. You may need to close and re-open the app after the distribute is complete to see the prompt.

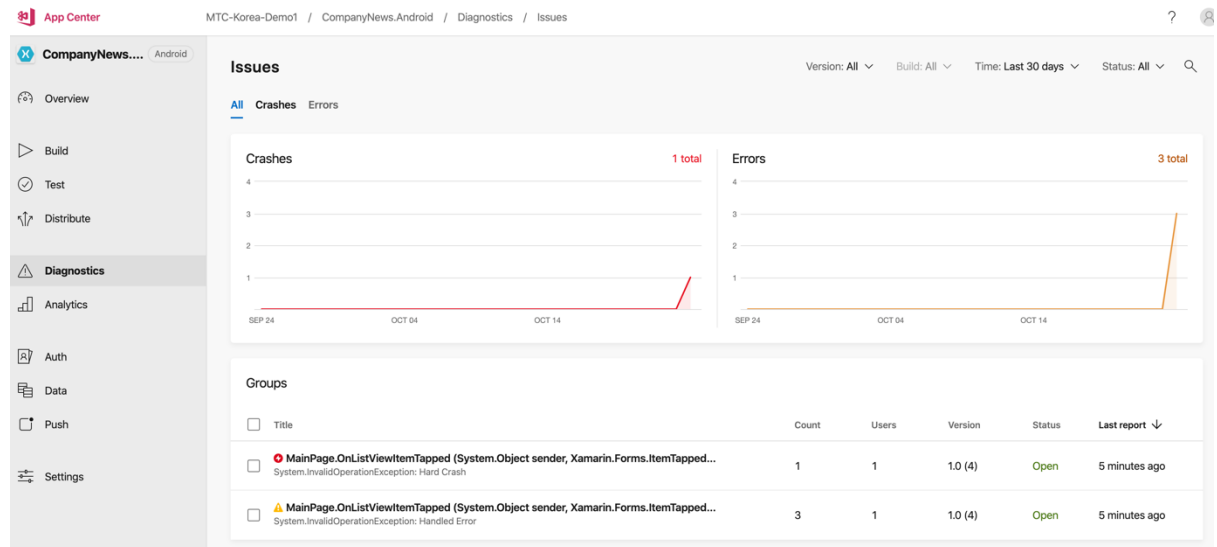


17. Congratulations! You have completed the Distribution module and learned how to distribute your app to beta testers and enable in-app updates.

Module 4: Analytics & Diagnostics

1. In the previous module we enabled the Analytics and Diagnostics integration with the App Center SDK. In this module we will show how to view that data. If you haven't already, open the latest version of the app on your phone and start clicking around. Open some news items and click both the Handled Error item as well as the Hard Crash item. The Hard Crash will cause the app to exit, so re-open it.
2. Navigate to the App Center Diagnostics page and you will see some data populated here. You can also intentionally crash the app or cause more handled errors to see these numbers rise. There is a short delay from when they occur to when they will

show up here.



3. Take some time to explore this UI. Notice that if you drill into a single report, you can see the additional details that we passed along as extra properties in the code for the Handled Error:

Mi 9T
Android 9

Version: 1.0 (4) | Occurrence: Oct 24, 3:14 PM | [Download](#)

MainPage.OnListViewItemTapped (System.Object sender, Xamarin.Forms.ItemTappedEventArgs e)
System.InvalidOperationException: Handled Error

Country	Device	Network
Korea, Republic Of	Mi 9T	N/A
Language	OS	User ID
English	Android 9	N/A

STACKTRACE

RAW

Main thread

CompanyNews MainPage.OnListViewItemTapped (System.Object sender, Xamarin.Forms.ItemTappedEventArgs e)

Error properties

title	contentBrief
Handled Error	Lorem ipsum dolor si...

4. Navigate to the Analytics page. You will see the different statistics that we are able to collect about the users. Take some time to familiarize yourself with this page.
5. Navigate to the Events sub-page, you will see the custom event that we added in the code earlier. Scrolling down to the bottom of that page, you will see the extra

properties we added to this event and in what percentage.

ContentBrief	Count	Trend
Lorem ipsum dolor si...	9 100%	N/A

Title	Count	Trend
Handled Error	3 33.3%	N/A
Hard Crash	3 33.3%	N/A
New Safety Training Available	2 22.2%	N/A
Public Holiday Next Week	1 11.1%	N/A

6. Navigate to the Log flow sub-page – this is where you can see a real-time event stream of what is happening inside your app. Open this page and then open the app on your device, you will be able to see the activity you are doing in real-time.

Log flow

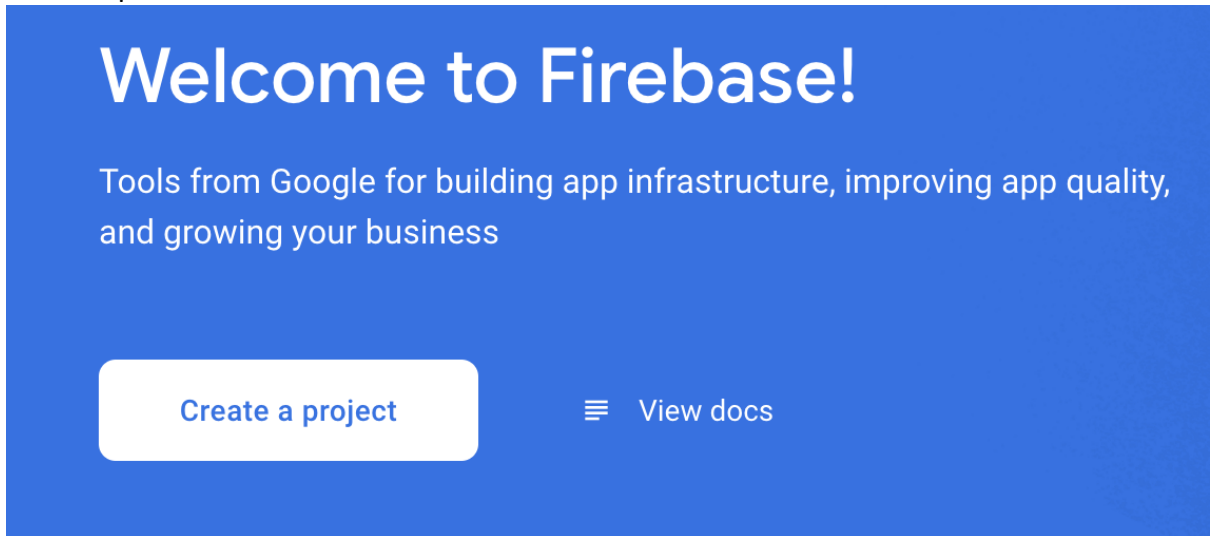
Id	Description	Time
• ab088581	STARTSESSION - 329a7727-bd46-4bf3-a4a1-de2c40c18c9f	06:28:08
• ab088581	EVENT - News Item Selected - {"ContentBrief":"Lorem ipsum dolor si...", "Title":"New Safety Training Av...	06:28:09
• ab088581	EVENT - News Item Selected - {"ContentBrief":"Lorem ipsum dolor si...", "Title":"Public Holiday Next We...	06:28:13
• ab088581	EVENT - News Item Selected - {"ContentBrief":"Lorem ipsum dolor si...", "Title":"Cafeteria is now open!...	06:28:19

7. Congratulations 🎉! You have completed the Analytics and Diagnostics module. You have learned how to view crashes affecting your app, as well as analytics for how people are using your app. Using custom events, you can add tracing in your code to get an understanding of how customers are using your app and make changes based on that.

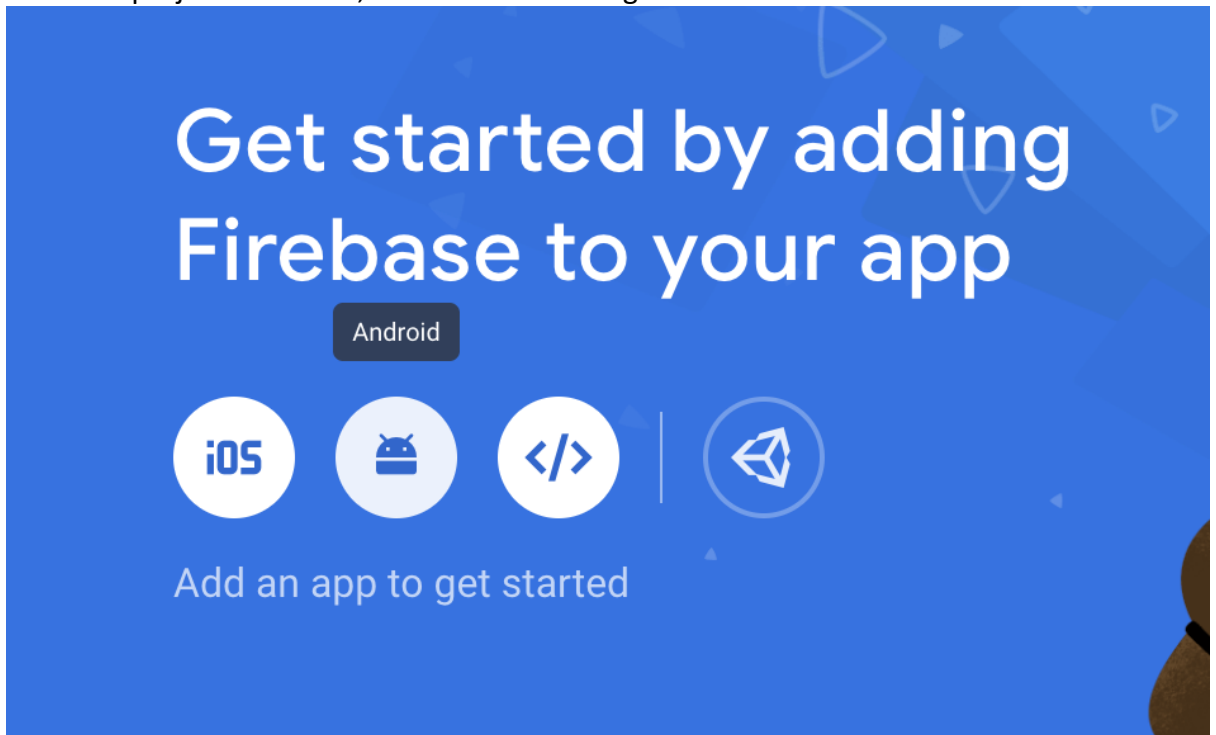
Module 5: Push

1. Now let's switch gears and integrate push notifications into the app. Using App Center you can send targeted push notifications to your app's users. To get started, navigate to the Push section in the portal. This first section shows how to integrate the App Center SDK, but we have already done that so click Next.
2. This next step requires creating a project on Firebase. Sign-in to the Firebase console using your existing Google account, or create a new Google account for this

workshop.



3. Once the project is created, click the Android logo.



4. Fill in the details for the app. For the Android package name fill in:
ms.appcenter.sample.companynews

×

Add Firebase to your Android app

1

Register app

Android package name ?

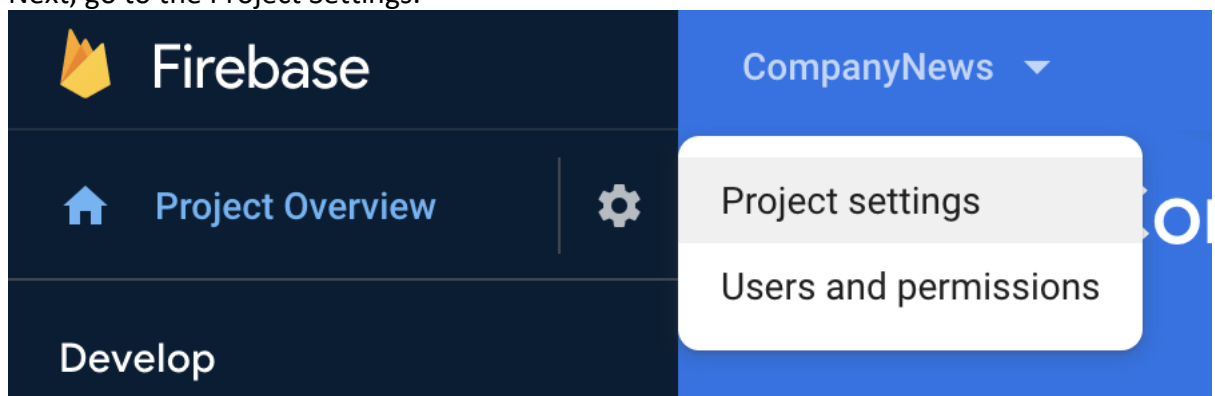
App nickname (optional) ?

Debug signing certificate SHA-1 (optional) ?

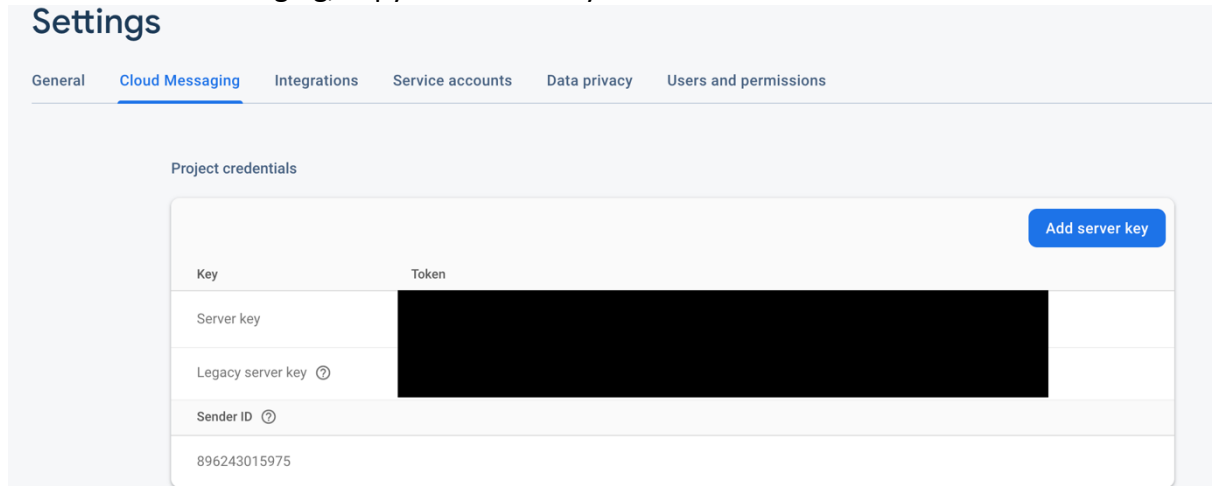
Required for Dynamic Links, Invites, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

Register app

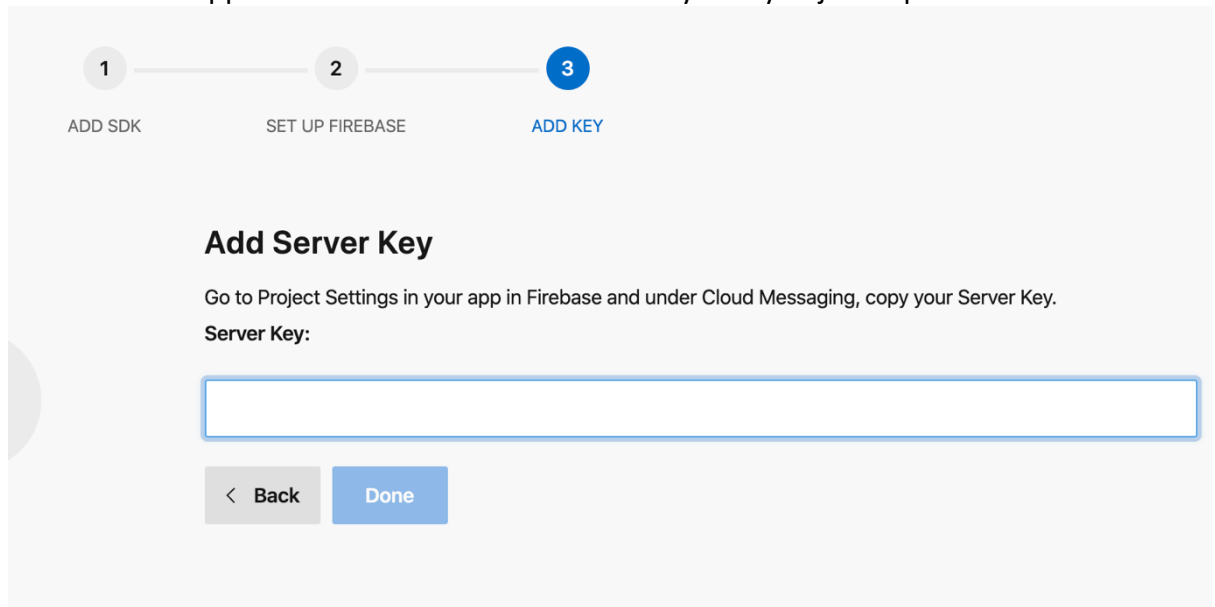
5. On the next page download the google-services.json.
6. Next, go to the Project Settings.



7. And on Cloud Messaging, copy the Server key.




8. Return to the App Center UI and fill in the Server Key that you just copied:




9. Now we need to add the google-services.json file into our app. Because this file has sensitive information, we don't want to check it into our git repository. Instead what we will do it utilize a feature of App Center Build for custom environment variables and custom build scripts to automatically add this file during build. The script is already set up, it is called appcenter-pre-build.sh and is in the CompanyNews.Android project folder. It is looking for a variable named GOOGLE_SERVICES_JSON, so we will set that variable. Because this is sensitive information we will also indicate to App Center that this should be treated as sensitive information. Navigate back to the Build configuration and add a new environment variable called GOOGLE_SERVICES_JSON with the contents of the

google-services.json file you downloaded earlier.

Environment variables

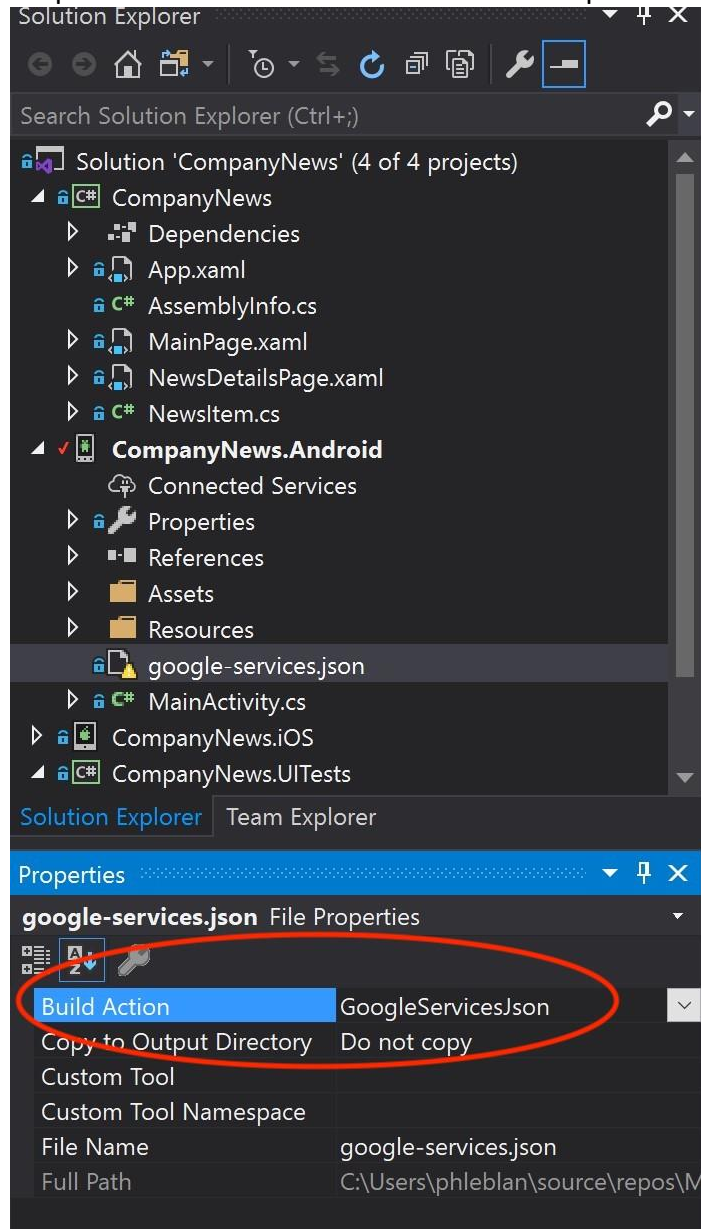
 On

Specify custom [environment variables](#) to use during the build process.

GOOGLE_SERVICES_JSON	<div>.....</div>	—
Name	Value	+

10. Click Save (not Save & Build – we have one more step before building).
11. Open Visual Studio and in the CompanyNews.Android project you will see a file for google-services.json. This file doesn't exist on disk, but since we will create it during build time this will work. Change the Build Action to GoogleServicesJson in the

Properties window for this file. Commit and push these changes.



12. The build will complete and automatically distribute to the Beta Testers group, open the app on your phone and use the in-app update notification to update to the latest version.