# Summary

Great work. You've successfully deployed the Space Game website to Azure App Service by using Azure Pipelines. If you've gone through the Build applications with Azure DevOps  learning path, you know you've taken a big step. You now have a live environment! Take advantage of it and run additional tests or make it available to your users.

Continuous delivery helps you release reliable software updates to your customers as rapidly as the business demands. Your customers can have the latest features and fixes as soon as you're ready to release them. By using the analytics provided by Azure Pipelines, your team can pinpoint hot spots and areas for improvement.

Setting up Azure Pipelines to publish the Space Game website to App Service is a great first step for the team. But this accomplishment is just the beginning. In the modules that follow, you help the team expand their release pipeline to include additional stages and tests. Each improvement to the pipeline will give the team more confidence in the quality of their releases.

## Learn more

Here are some additional resources you can use to learn more about Azure resources, App Service, and many other topics.

### **Choose the right compute option for your app**

In this module, you deployed a website to App Service. You can use Azure Pipelines to deploy to just about any kind of environment, including a virtual machine or containerized environment.

The Core Cloud Services - Azure compute options  module goes into further detail about the kinds of compute options Azure provides.

You'll work more with App Service in the modules that follow. You can also learn about App Service in the Deploy a website to Azure with Azure App Service  learning path.

### **Deploy to your preferred environment**

After you choose where to deploy your applications, learn how to use Azure Pipelines to deploy to that environment.

* Deploy an Azure web app
* Deploy to a Linux virtual machine
* Deploy to a Windows virtual machine
* Deploy a Docker container app to Azure Kubernetes Service
* Build and deploy Java to Azure Functions

### **Integrate with other CI/CD tools**

Azure DevOps enables you to build on your investment with other CI/CD tools. Learn how Azure DevOps integrates with Jenkins, CircleCI, and GitLab.

* Continuously deploy from a Jenkins build
* CircleCI artifacts for Release pipeline
* GitLab integration for Azure Pipelines

### **Go deeper**

These resources provide more information about some of the topics in this module.

* When should you right-click publish?
* Jobs
* Deployment jobs
* Environments
* Design for security in Azure
* Secure your Azure resources with RBAC
* Service connections
* Pipeline reports
* Explore the Analytics OData metadata

## Appendix: Implement a mobile DevOps strategy

The following sections are optional. They provide additional information beyond what's covered in this module.

So far, you've deployed the Space Game website to Azure App Service but there are, of course, other opportunities for deploying your app and one of them is to mobile devices. You'll need the same commitment and cultural changes as with any other DevOps project and, of course, you'll also need the right toolset. Microsoft offers this toolset through Visual Studio Mobile Center (also called App Center), which brings together multiple services commonly used by mobile developers into an integrated cloud solution. Let's look at some of these services.

### **Distribution groups**

Distribution groups control access to releases. A distribution group represents a set of users that can be managed jointly and can have common access to releases. Examples 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 then 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.

#### **Private distribution groups**

In App Center, distribution groups are private by default. This means only testers invited through 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.

#### **Managing users in a distribution group**

Clicking on a distribution group allows you to manage the group. For example, you can use the invitation box to add additional users. You can select users from the table to remove them from the group. You can also see the full release history for this distribution group by clicking on the **Releases** tab.

It's 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 such as Espresso, XCUITest, Appium, and Xamarin.UITest.

#### **Adding Azure Active Directory (Azure AD) groups to a distribution group**

In addition to adding testers though email, you can also add Azure AD groups to a distribution group. Any member of an Azure AD group can link their organization's subscription to their Azure AD tenant in App Center.

### **Manage target UI test device tests**

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 set up before it's ready to run the launch test. You can read more about configuring your branch in the Configure a build documentation for Android  and iOS .

Second, 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.

Once you have an active subscription you can enable testing. All you need to do is to toggle **Run a launch test on a device option** to **ON** and click the **Finish set up** button. During the new build, there will be two parts, a build part and a test part. 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**

When the test is completed, the app collaborators receive an e-mail with test results. You can also go to **Test** in the left-hand menu and follow along as the tests progress.

### **Provision tester devices for deployment**

Getting your app onto your own device, or in the hands of potential testers, can be a time-consuming challenge 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 repetition 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.

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 (iOS App Store Package) file, 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.

Once you've turned on the **Automatically manage devices** setting, the 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. Every time you release a new version of your build, testers will get an email notification to install the app and start using it.