**Introduction to Azure Functions**

In Create a release pipeline with Azure Pipelines , you built a basic release pipeline that deployed an ASP.NET Core application to Azure App Service. Although this typical scenario covers the needs of a basic application, modern solutions often require a combination of applications, services, and other components.

One of the unique benefits cloud computing has delivered on is the broad array of *serverless* application options. *Serverless* applications are those where the application's execution and resources are dynamically allocated by the cloud provider as needed. As a result, they are never dependent on, or constrained by, a specific configuration of servers. While these applications still technically run on servers, they drastically simplify the development model by automatically provisioning, scaling, and managing the infrastructure required.

In this module, you'll join the Tailspin Toys web team as they explore one of Azure's core serverless application offerings, Azure Functions. You'll learn about how an Azure Functions project can be added to an existing CI/CD pipeline in order to deploy to both Azure App Service and Azure Functions as part of the same process.

While this module focuses on the core build and deployment tasks, it's important to understand that all of the other features of Azure Pipelines are still available for multi-project and Azure Functions solutions. You can integrate testing, multiple stages, and other tasks just like you would for other scenarios. We omit these details here to help keep things focused.

**Learning objectives**

In this module, you will:

* Understand the benefits of Azure Functions and when to use them.
* Extend a basic release pipeline to also deploy an Azure Function app.
* Monitor the build, release, and deployment of the project.

**Prerequisites**

The modules in this learning path form a progression. To follow the progression from the beginning, be sure to first complete these learning paths:

* Evolve your DevOps practices
* Build applications with Azure DevOps

We also recommend you start at the beginning of this learning path: Deploy applications with Azure DevOps . This module picks up where Create a release pipeline in Azure Pipelines  leaves off.

**Note**

Keep in mind that you can use Azure DevOps to build and deploy almost any kind of application written in any language. In this module, you'll be working with a .NET Core application written in C#.

You don't need to be an expert in .NET or C# to complete this module. You can apply the patterns you learn here to your own projects that use your favorite programming languages and frameworks.

This module also assumes you have basic familiarity with Azure Functions, although that knowledge is not required to complete it. If you are new to the topic, it is recommended that you complete the Create serverless applications  learning path first.

**What is Azure Functions?**

Here you follow the Tailspin team as they discuss how to best deliver on new marketing requirements driven by management. As always, the team needs to navigate a combination of organizational, business, and technical challenges to deliver their software. In this scenario, the team needs to integrate some serverless technology, specifically Azure Functions, into their DevOps process.

The team has been making great strides in building out their automated processes. However, part of the reality of DevOps is that requirements can be fluid, so the ability to react to change is paramount. Let's check in with the team.

**The new requirements**

Irwin, the product manager, has called an urgent meeting with key team members in order to deliver news about some new requirements.

**Irwin:** Thanks for meeting on such short notice. Things are heating up and the marketing team is starting to make some promises that could make things complicated for all of us.

**Andy:** Well, you've got my attention now!

**Irwin:** They really like our site. They think the leaderboard is something that can be used to promote the game as we head toward the final launch and beyond. They're looking to scale it out way beyond what we had originally planned.

**Mara:** That's great, right? We're on Azure, so we can scale the service out as much as needed.

**Irwin:** Well, it's not that simple. They started off talking about "leveraging" and "synergy" and I sort of zoned out checking email. When I came back around they were asking me to task you with finding the most secure way for us to give anonymous third parties access to our database.

**Tim:** What? That's a terrible idea! Do they understand how many security risks that could create?

**Irwin:** I know, I know. That was my first reaction as well. I probed a bit deeper and it turns out that they really just want the ability for anyone to integrate leaderboard data into their own applications. They think it would be really cool to have the same leaderboard options on our site, from within the game, on gaming review sites run by partners, and even by our players who want to show it on their blogs and personal sites.

**Andy:** So then it's just an API they're after. We already kind of have what we need to expose that, and I'm sure we can make it developer-friendly and scale up using Azure.

**Irwin:** Yes, I know you can. However, there are a few potential challenges we need to work through. The game engine team currently owns the production database that we pull leaderboard data from. They think they should build and own the API moving forward. I told them that it's really a better fit for our team, but management isn't sure either way.

**Andy:** How can we show them that we have it under control?

**Irwin:** They're going to give us a week to put a proof of concept in place. However, they said that the final decision of ownership will be made based on the correct organizational structure. As a result, we will need to deliver a solution that is completely decoupled from the web site. Any team should be able to take over the project as a standalone component.

**Tim:** I'm not sure how I feel about this. I'm already spread too thin. I don't know if I have the bandwidth to add managing the infrastructure for yet another site. I'm also concerned about the scale requirements since we probably need to expect unpredictable surges in demand followed by lulls. We won't want to pay for peak demand the entire time.

**Andy:** Mara, do you have any thoughts?

**Mara:** This will take a bit of research, but I'm sure we can come up with something. Here are my notes:

* We need to deliver quickly, so we have to find a solution where we can leverage our code and skills.
* We need to refactor a small piece of functionality into an independent service.
* We need to minimize administrative overhead in production.
* We need to have massive scale without making a massive commitment.

The team nods in agreement.

**Andy:** Great, Mara and I will meet tomorrow to come up with a plan.

**It works on my machine**

The meeting breaks up. The next morning, Andy excitedly pops into Mara's office for an impromptu chat.

**Andy:** I've got it! I found a solution that meets all of our requirements for the Leaderboard API. It's a serverless offering called Azure Functions.

**Mara:** That's great! How long do you think it will take for us to have something we can demo?

**Andy:** I've already spiked something out. I forked our GitHub repo and refactored the leaderboard code into a new Azure Functions project. It exposes the leaderboard data via a simple HTTP GET request. I also updated the web site to now use that endpoint when requesting the leaderboard, and anyone else can do the same.

**Mara:** This sounds like exactly what we need. I can help extend the existing CI/CD pipeline to support building and deploying it like we do for the main site project.

**Note**

In this module, we won't focus on how to connect your app to Azure Functions. But if you're interested, you can **review the source code**. The code uses the **Azure Functions runtime for C#** to read leaderboard data from Azure Functions.

**What is Azure Functions?**

**Azure Functions** is one of the specific offerings within the broader spectrum of Azure serverless computing technologies. It provides an easy way for developers to build straightforward functions that exist in a stateless, serverless environment. They can be triggered using a variety of methods, including HTTP requests, changes to data in storage, receipt of a message from a queue, and more. You can learn more about the various trigger and bindings in Azure Functions triggers and bindings concepts .

The Tailspin team selected Azure Functions for this scenario because it met all their needs:

* We need to deliver quickly, so we have to find a solution where we can leverage our code and skills.
  + *We can build the solution quickly as an ASP.NET Core application leveraging code we already have.*
* We need to refactor a small piece of functionality into an independent service.
  + *We can deliver an independent function with no other application surface area to worry about.*
* We need to minimize administrative overhead in production.
  + *Azure Functions removes the need to worry about hosting environments.*
* We need to have massive scale without making a massive commitment.
  + *Serverless technologies have virtually unlimited scale that can automatically ramp up and down much faster than virtual machines or app services.*

Azure Functions and other serverless technologies are not always the right fit for every solution. Later on we will discuss where you can find resources to help you select the right technology for the job.