

App Modernization Labs

Azure App Services with Azure DevOps

Author: Divi Mishra

Cloud Solution Architect, Azure App Innovation, Microsoft Qatar

divimishra@microsoft.com | +974 50219105

Introduction to the Lab Document

Objective

This workshop lab is intended to materialize on theoretical Azure App Services learnings to have hands-on experience with end-to-end Azure App Services tools across the Developer Cloud. Through this workshop lab, you will have a basic yet broad understanding of how to realize value from the different offerings within and beyond Azure App Services.

Motive

When it comes to modernizing your web apps, Azure App Service is the best destination. A recent GigaOM study found out that Azure App Service offers potential total cost of ownership (TCO) savings of up to 54% over running on-premises while offering tangible benefits around streamlined operations, increased developer productivity, DevOps readiness and reduced friction.

Intended Audience

The intended audience for this workshop lab includes, but is not limited to: development team, application managers, enterprise architects, and technical managers. The difficulty level of this workshop is beginners.

Duration

This workshop lab followed step-by-step will take approximately 3 hours to complete.

Pre-requisites

1. Visual Studio Code application
2. A [GitHub](#) account
3. An [Azure DevOps](#) setup
4. An active [Azure](#) subscription
5. [.NET 5.0 SDK](#)
6. [Git](#)

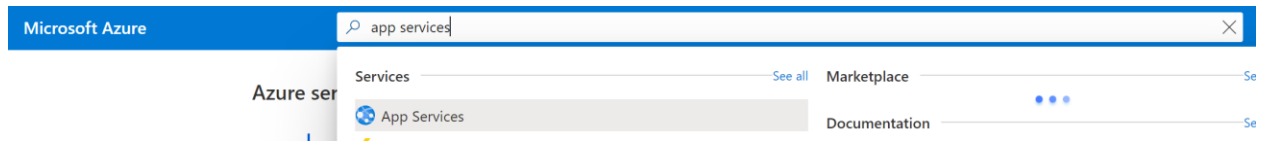
LAB 01: AZURE APP SERVICE	5
Create an Azure App Service	5
Fork the web project to your GitHub account	6
Create deployment slots	12
 LAB 02: MONITOR YOUR APP WITH AZURE APPLICATION INSIGHTS.....	 13
Enabling live telemetry through instrumentation key using VS Code	13
Monitor apps	15
Configure alerts with Azure Logic Apps for Azure App Service	16
 LAB 03: SCALE APPLICATIONS WITH AZURE APP SERVICE PLAN	22
Scale up your app service.....	22
Scale out your app service	22
 LAB 04: CONTINUOUS INTEGRATION WITH AZURE DEVOPS.....	23
Configure Azure DevOps project	23
Create a build pipeline with Azure Pipelines	24
Source Control with GitHub	34
 LAB 05: TESTING WITH AZURE PIPELINES.....	36
Quality tests	36
 LAB 05: ZERO-DOWNTIME APP DEPLOYMENT WITH RELEASE PIPELINE.....	 39
Deploy to staging slot from build pipeline	39
Build a release pipeline: zero-downtime deployment with slot swapping	47

OPTIONAL LABS.....	52
Optional Lab 01: Enabling continuous integration with YAML files	52
Optional Lab 02: Protect App Services with other WAF Options	54
Optional Lab 03: Azure Defender for App Services.....	55
Optional Lab 04: Managing technical debt with SonarQube and Azure DevOps.....	56
CLEAN-UP YOUR ENVIRONMENT	57
RESOURCES.....	58

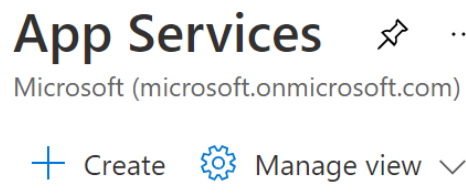
Lab 01: Azure App Service

Create an Azure App Service

1. Go to the [Azure Portal](#)
2. Search for **App Services** at the search bar on top



3. Click on **+ Create**



4. Under **Basics** tab, enter the following details:
 - a. Select the relevant resource group or create a new one
 - b. Type an app service name. Example: app-appdevworkshop-01
 - c. Publish: **Code**
 - d. Runtime Stack: **.NET 5**
 - e. Operating System: **Windows**
 - f. Region: **West Europe** (or any other region close to you)
 - g. App Service Plan:
 - i. Create new > Enter a name (example: asp-appdevworkshop)
 - ii. Select **Standard S1** under Production as your app service plan size
5. Go through the next tabs to understand settings. You don't have to change any other settings (accept defaults for all other settings).
6. Click on **Review + Create**
7. Click on **Create**

Create Web App ...

all your resources.

Subscription * ⓘ MSInternalAccess

Resource Group * ⓘ rg-appdevworksop-01
[Create new](#)

Instance Details

Need a database? [Try the new Web + Database experience.](#) ⓘ

Name * app-appdevworkshop-02

Publish * ☒ Code ☐ Docker Container

Runtime stack * .NET 5

Operating System * ☐ Linux ☒ Windows

Region * West Europe
 ⓘ Not finding your App Service Plan? Try a differen

App Service Plan

App Service plan pricing tier determines the location, features, cost and compute resources asso
[Learn more](#) ⓘ

Windows Plan (West Europe) * ⓘ ASP-mslearntailspinspacegameweb-9893 (S1)
[Create new](#)

Sku and size * **Standard S1**
 100 total ACU, 1.75 GB memory

[Review + create](#) < Previous Next: Deployment >

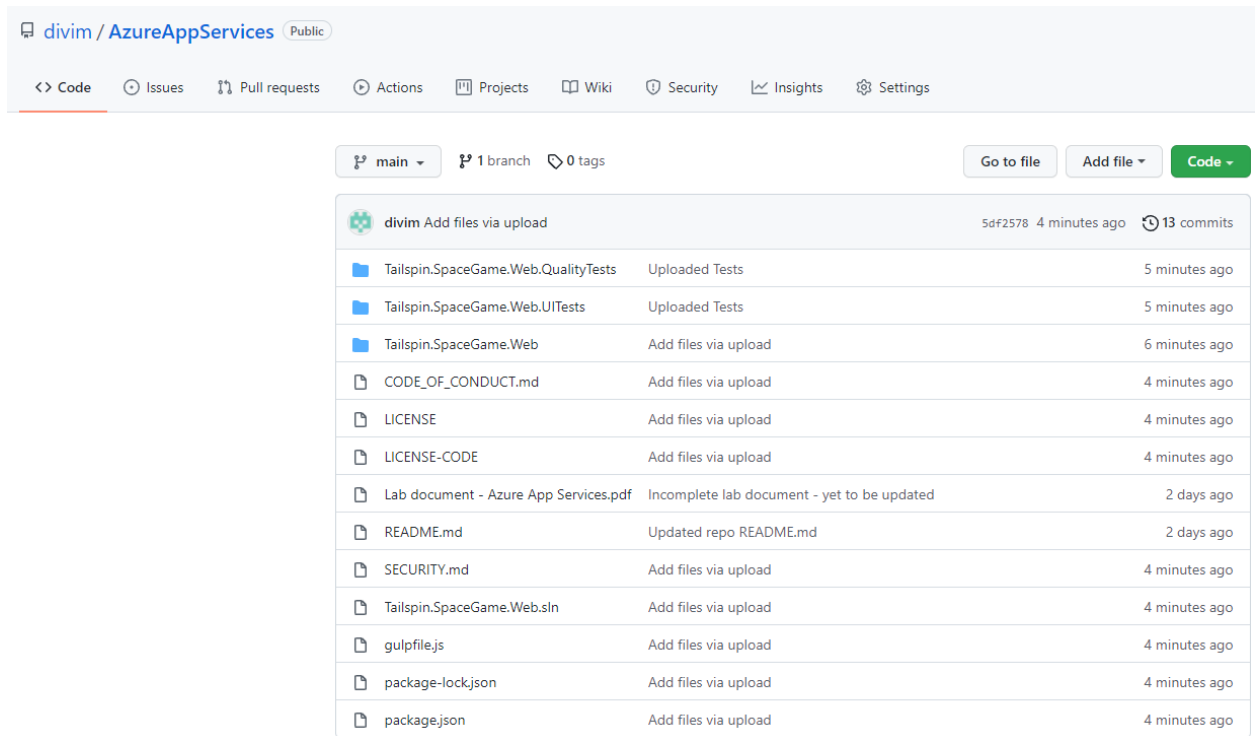
- Once the deployment is complete, click on “Go to resource”.
- From the overview tab, click on the “URL” to view the default web application that gets deployed.

^ Essentials	
Resource group (change) :	rg-appdevworkshop
Status :	Running
Location :	West Europe
URL :	https://appworkshop232.azurewebsites.net
Health Check :	Not Configured
App Service Plan :	asp-appdevworkshop (S1: 1)

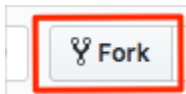
- You should be able to view a welcome page.

Fork the web project to your GitHub account

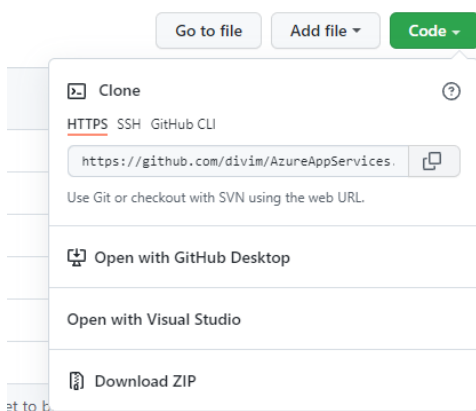
- Go to [GitHub](#) and sign in
- Visit the [Azure App Services workshop repo](#) (ASP.Net Core application)



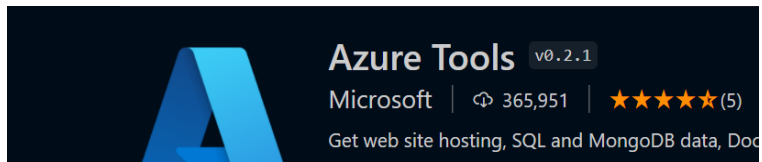
- From the top right corner, fork your own copy of the repo to your account.



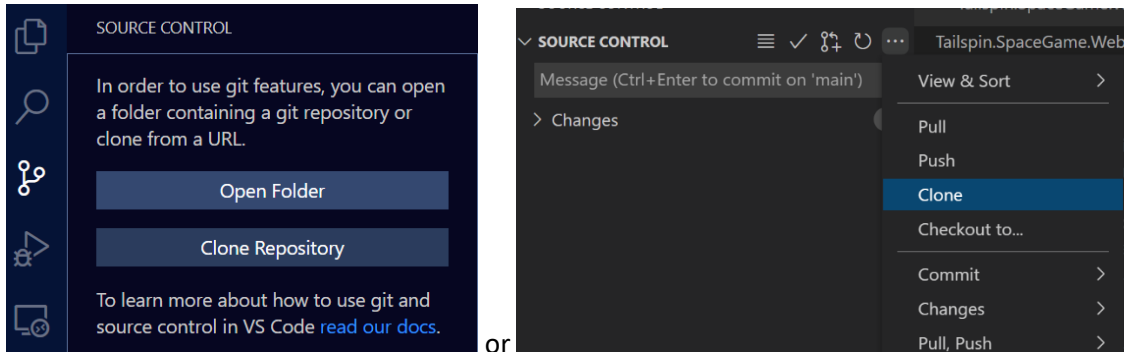
- Go to your fork of the Space Game project. The forked repository will be saved as “<your-account-name>/AzureAppServices”.
- Select “Code”. Under the “HTTPS” tab, copy the URL.



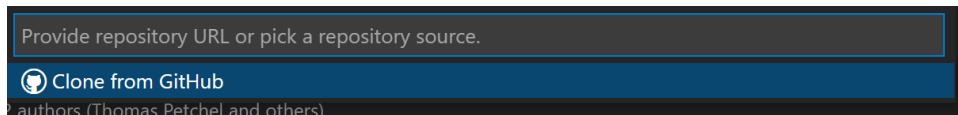
- Open Visual Studio code.
- Install the necessary extensions to visual studio code. Click on the **Extensions** button (or **CTRL+Shift+X**) on the left side ribbon, search for “**Azure Tools**” extension (from Microsoft), and install it.



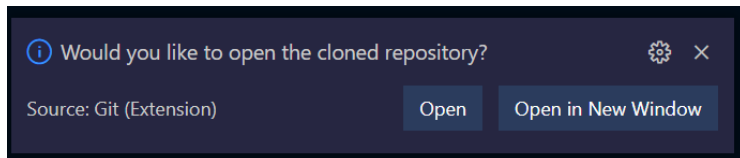
- Under **Source Control**. Click on **Clone Repository**.



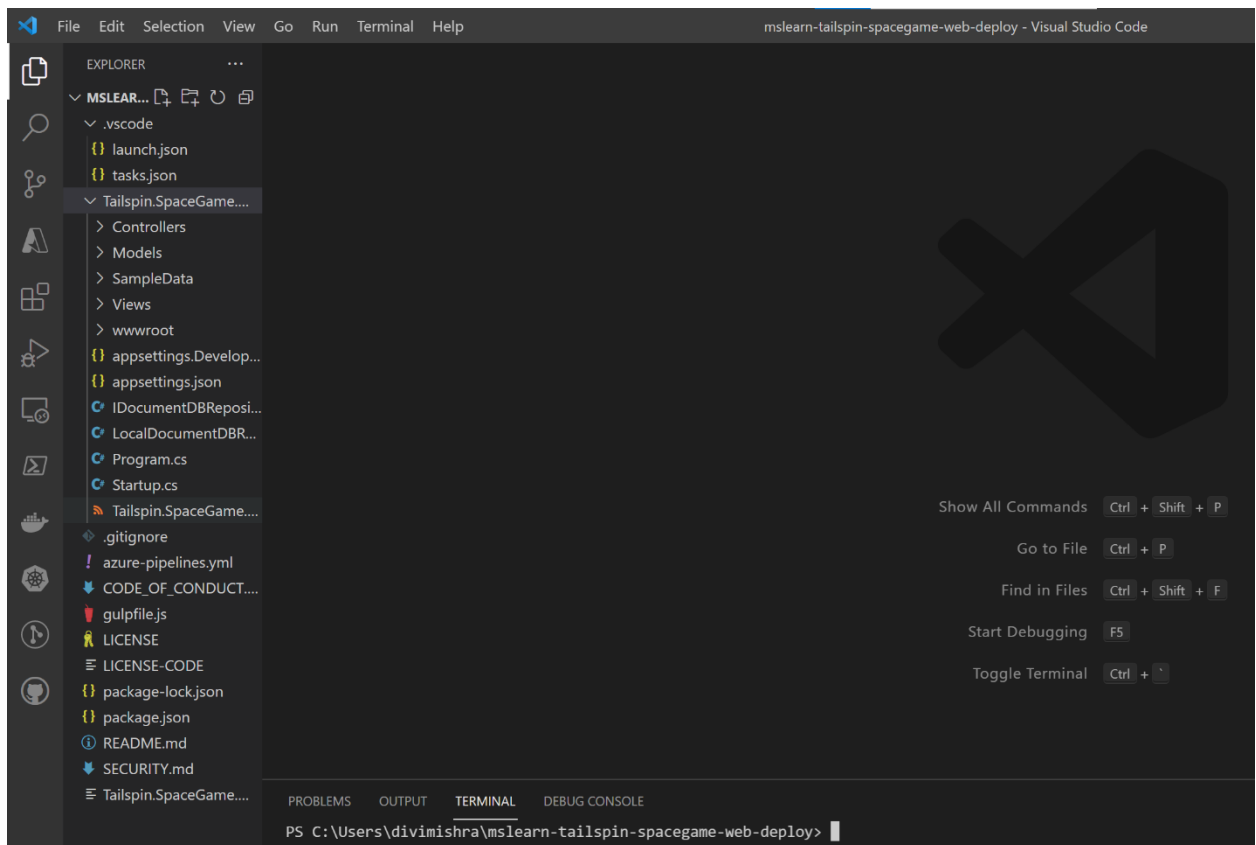
- Click on **Clone from GitHub**



- Sign-in and choose **AzureAppServices**.
- Select a folder on your local machine to clone the files.
- Once the clone is complete, in the visual studio code's prompt for opening the repository, click **"Open"**.



- You are now at the root of your web project. Open the terminal by going to **"View > Terminal"** or **"Ctrl + `"**



14. (Optional) Enter the following commands with “Sample Name” and sampleemail@abc.com replaced with your name and your commit email address.

```
$ git config --global user.name "Sample Name"
```

```
$ git config --global user.email "sampleemail@abc.com"
```

Note: The “--global” tag sets the entered username and email address for every repository on your computer. If you want to set your username or email address for a single repository, use:

```
$ git config user.name "Sample Name"
```

```
$ git config user.email "sampleemail@abc.com"
```

Run and validate the Project Locally.

1. Enter the following command on the terminal to make sure the code runs on your local host:

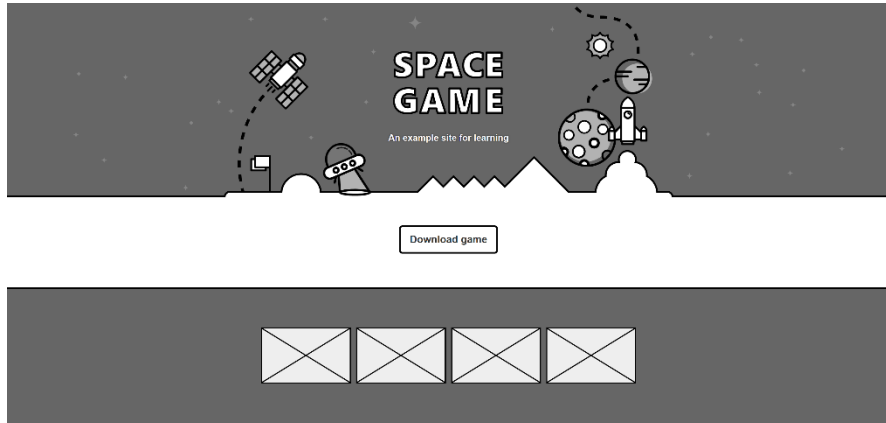
```
$ dotnet build --configuration Release
```

```
$ dotnet run --configuration Release --no-build --project .\Tailspin.SpaceGame.Web\
```

```
$ dotnet publish -c Release -o ./publish
```

Navigate to your <http://localhost:5000>

```
PS C:\Users\divimishra\mslearn-tailspin-spacegame-web-deploy> dotnet run --configuration Release --no-build --project .\Tailspin.SpaceGame.Web\
Hosting environment: Production
Content root path: C:\Users\divimishra\mslearn-tailspin-spacegame-web-deploy\Tailspin.SpaceGame.Web
Now listening on: http://localhost:5000
Now listening on: https://localhost:5001
Application started. Press Ctrl+C to shut down.
```



Ctrl+C when you're done.

Publish the project to Azure App Service

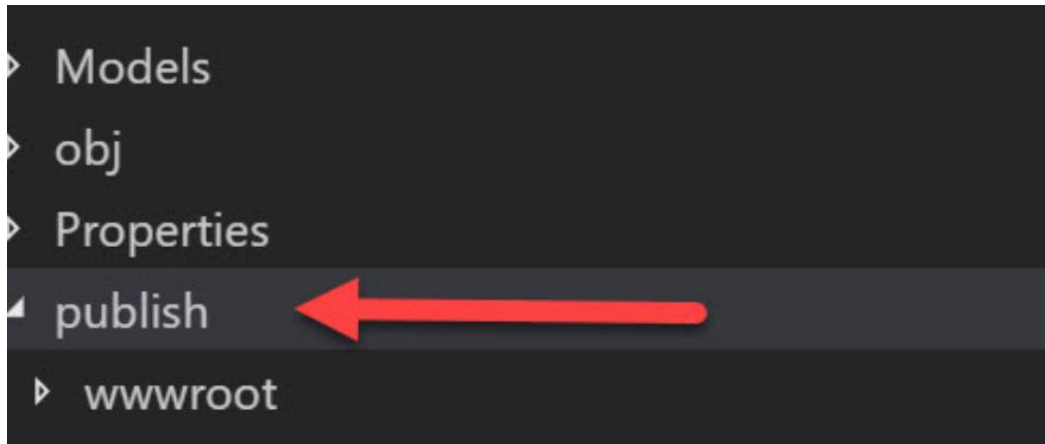
1. Install the Azure icon Click on the Azure icon in Visual Studio Code's left side ribbon



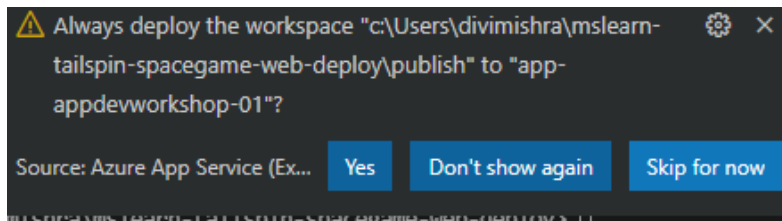
2. Sign in to your Azure Account when prompted.
3. Drill down into your subscription -> resource group -> App service (that you created in step 1).
4. Visual studio code will prompt you to install the app service extension. Click "Install" to Install the extension.
5. Once the extension is installed, go back to your file explorer.
6. Go back to your terminal and enter the following command:

```
$ dotnet publish -c Release -o ./publish
```

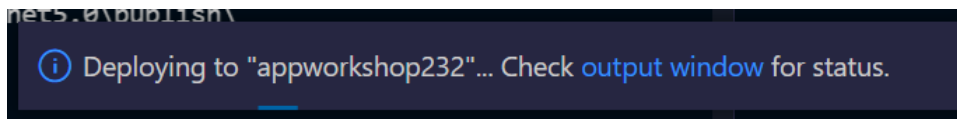
You will notice that a new publish folder has been created



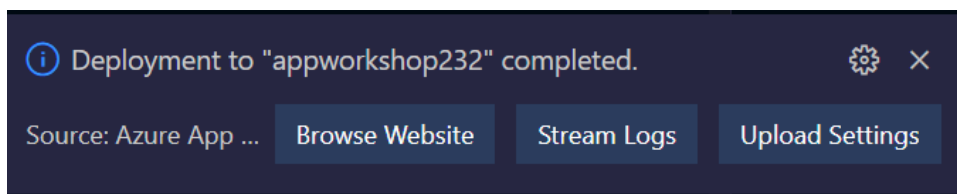
7. Right click on the publish folder.
8. Select Deploy to web app... and select the right subscription.
9. Select the Azure App service you had created.
10. Select Deploy to confirm.
11. If you get the following prompt, click on **Don't show again**. We will be deploying Azure Pipelines for CI/CD in Lab 4.



12. You can view the deployment status using the output window.



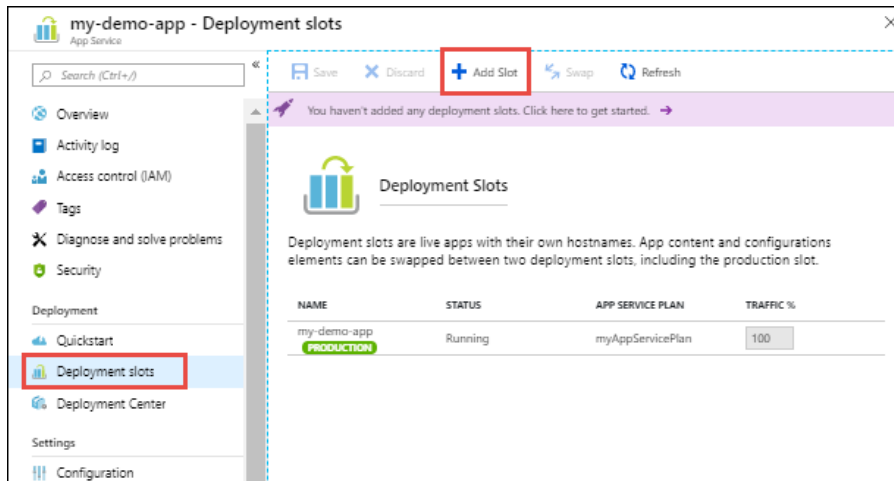
13. Once deployment is completed, select "Browse Website" button to open the azure app services website.



14. Once the deployment is done, click on Browse Website to validate the deployment.

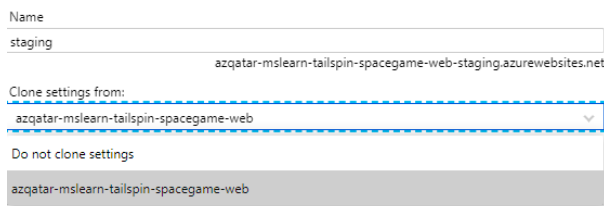
Create deployment slots

1. Navigate to your newly created app service from Azure Portal.
2. Under deployment, click on Deployment Slots. Then, add a slot.



Note: The app service tier must be Standard, Premium, or Isolated to enable staged publishing.

3. In the “Add Slot” dialog box, give the slot a name “**staging**” and select whether you want to clone an app configuration from another deployment slot.



4. Once the slot has been created, close the dialog box. You will notice that for the staging environment:
 - a. The default traffic is set to 0.
 - b. The slot's URL will be of the format `http://sitename-slotname.azurewebsites.net`
 - c. The web URL is empty even if we clone it from the production app. Hence, you can use a separate branch or a separate repository altogether to test the application.

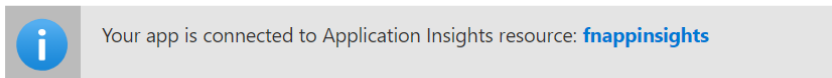
Note: If you would like to add an access rule restriction for the staging environment, please follow the steps from this document: [Azure App Service access restrictions - Azure App Service | Microsoft Docs](#)

Lab 02: Monitor your app with Azure Application Insights

Enabling live telemetry through instrumentation key using VS Code

1. Navigate to your Azure App Service through your Azure Portal.
2. Under settings, go to Application Insights.
3. Make sure that the collection is **Enabled**.
4. Click on Apply (if you enabled application insights just now).
5. Click on the application insights link to open the application insights instance.

Link to an Application Insights resource



6. Copy the instrumentation key from the application insights overview page.

A screenshot of the Application Insights overview page. It shows the "Instrumentation Key" as "30b1eded-d7d5-418e-9ed5-5044046bd6f2" with a copy icon. Below it, the "Connection String" is shown as "InstrumentationKey=30b1eded-d7d5-418e-9ed5-5044046bd6f2;IngestionEndp...". At the bottom, the "Workspace" is listed as "fnlawkspace".

Instrumentation Key : 30b1eded-d7d5-418e-9ed5-5044046bd6f2

Connection String : InstrumentationKey=30b1eded-d7d5-418e-9ed5-5044046bd6f2;IngestionEndp...

Workspace : [fnlawkspace](#)

7. On the VS Code terminal, navigate to /Tailspin.SpaceGame.Web using the cd command.

```
$ cd /Tailspin.SpaceGame.Web
```
8. Run the following command:
dotnet add package Microsoft.ApplicationInsights.AspNetCore --version 2.18.0
9. Navigate to the Startup class under **Startup.cs**. Add the following command under the ConfigureServices() method:

```
services.AddApplicationInsightsTelemetry("Insert instrumentation key");  
  
services.AddMvc();
```

The method will look something like this:

Screenshot:

```

namespace TailSpin.SpaceGame.Web
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddControllersWithViews();
            services.Configure<CookiePolicyOptions>(options =>
            {
                // This lambda determines whether user consent for non-essential cookies is needed for a given request.
                options.CheckConsentNeeded = context => true;
                options.MinimumSameSitePolicy = SameSiteMode.None;
            });

            // Add document stores. These are passed to the HomeController constructor.
            services.AddSingleton<IDocumentDBRepository<Score>>(new LocalDocumentDBRepository<Score>(@"SampleData/scores.json"));
            services.AddSingleton<IDocumentDBRepository<Profile>>(new LocalDocumentDBRepository<Profile>(@"SampleData/profiles.json"));

            // The following line enables Application Insights telemetry collection.
            services.AddApplicationInsightsTelemetry();

            // This code adds other services for your application.
            services.AddMvc();
        }
    }
}

```

Save all the changes (Ctrl + S).

Note:

- Ensure that your repository is a private repository as you include the instrumentation key within the code itself
- Alternatively, you can access the instrumentation as an environment variable (App Service > Configuration > App Settings). The syntax for accessing the app settings:

```

using System.Configuration;
string value = ConfigurationManager.AppSettings["key"];

```

10. On the terminal:

```
$ cd .. (to go back to the root directory of your web app code)
```

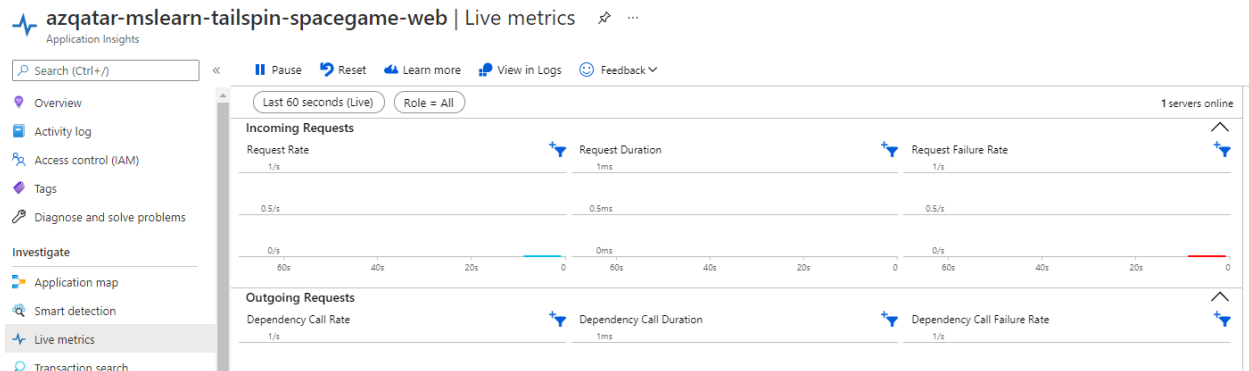
```
$ dotnet run -c Release --no-build --project .\Tailspin.SpaceGame.Web\
```

```
$ dotnet publish -c Release -o ./publish
```

11. Right-click on the **publish** folder and click on **Deploy to web app** for the changes to be reflected. (Soon, you'll deploy a CI/CD pipeline using Azure Pipelines to automate this process and other processes for you).

12. To verify that the application insights was configured accurately, navigate to **Live Metrics**.

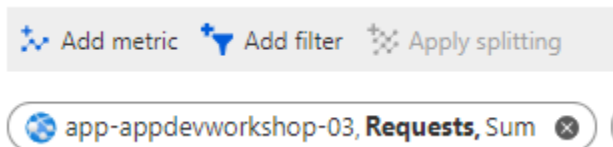
Please note that it may take a few minutes for the telemetry to appear.



This verifies that your application insights with instrumentation key is enabled accurately!

Monitor apps

1. Go back to your Azure App Service.
2. Under **Monitoring**, click on **Metrics**
3. Select **Requests** as your metric, and **Sum** as your aggregation.
4. Select **Add metric** from the top left.



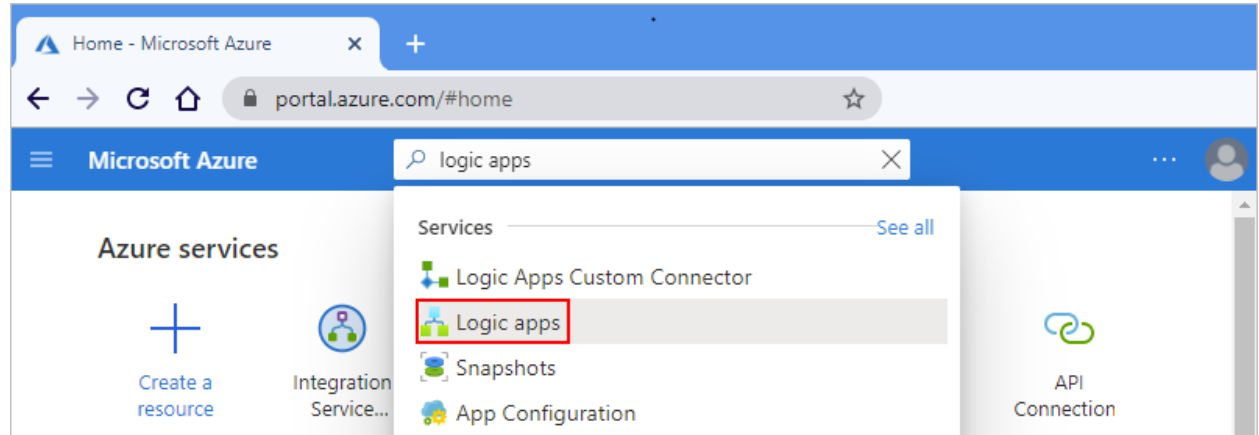
5. Select metrics of your choice and display them into meaningful bar representations. For example:
 - a. Select Add metric, and under the Metric dropdown list, select CPU Time. For Aggregation, select Sum.
 - b. Select Add metric, and under the Metric dropdown list, select Http Server Errors. For Aggregation, select Sum.
 - c. Select Add metric, and under the Metric dropdown list, select Http 4xx. For Aggregation, select Sum.
 - d. Select Add metric, and under the Metric dropdown list, select Response Time. For Aggregation, select Avg.
6. Select **"Pin to Dashboard"**. Click on the notification for viewing your dashboard. This is for developing shared dashboards to share with team members.



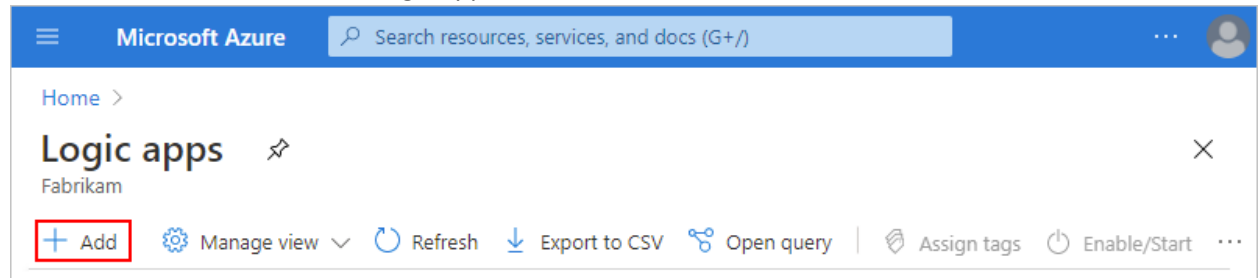
Configure alerts with Azure Logic Apps for Azure App Service

Creating the Logic App

1. Open Azure Portal in a new tab
2. Navigate to **Logic Apps** from the search bar on top



3. Click on + **Add** to create a new logic app



4. Select your resource group
5. Under **Instance Details: Type: Consumption**

Consumption vs Standard instance details:

 - **Consumption:**
 - This logic app resource type runs in global, multi-tenant Azure Logic Apps
 - Pay-for-what-you-use pricing model
 - Only one workflow
 - **Standard:** This logic app resource type runs in single-tenant Azure Logic Apps
 - Pricing is based on a hosting plan
 - Portable runtime runs anywhere (Azure, containers, on-prem, etc.)
 - Can have multiple stateful and stateless workflows
6. Select **Review + Create** and then **Create**.
7. Once the logic app resource has been created, click on **Go to resource**.
8. Click on **Open designer**

alertdemo Logic app

Search (Ctrl+/)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Development Tools

Logic app designer

Logic app code view

Versions

API connections

Quick start guides

Settings

Workflow settings

Authorization

Access keys

Identity

Properties

Locks

Monitoring

Run Trigger Refresh Edit Delete Disable Update Schema Clone Open in mobile Export Feedback

Introducing the new portable Logic Apps runtime that supports local development and debugging. Click to learn more. →

Essentials

Resource group (change) : rg-appdevworkshop-01

Location : North Central US

Subscription (change) : MSInternalAccess

Subscription ID : 843194f0-795a-4ab9-ae40-1f0e2b917cf8

Definition : 0 triggers, 0 actions

Status : Enabled

Runs last 24 hours : 0 successful, 0 failed

Integration Account : -- --

Get started Runs history Trigger history Metrics

Create workflow using the innovative visual designer

Build mission critical workflow with connectors for hundreds of services from Azure, Microsoft, and 3rd parties. [Learn more](#)

Edit in designer

Start creating the workflow using visual designer. Choose from one of the template, or start from scratch.

Open designer

Start from template

Choose from a template to start your workflow.

Choose template

Quickstarts

Read the quickstart guide on how to create your first Logic App.

Open quickstart guide

9. Under Start with a common trigger, Click on When a HTTP request is received

Home > alertdemo

alertdemo | Logic app designer

Search (Ctrl+/)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Development Tools

Logic app designer

Logic app code view

Versions

API connections

Quick start guides

Settings

Workflow settings

Authorization

Access keys

Identity

Properties

Locks

Monitoring

Alerts

Introducing Azure Logic Apps

Watch later Share

Azure Logic Apps

Watch on YouTube

Building integration solutions is easier than ever

Logic Apps brings speed and scalability into the enterprise integration space. The ease of use of the designer, variety of available triggers and actions, and powerful management tools make centralizing your APIs simpler than ever. As businesses move towards digitalization, Logic Apps allows you to connect legacy and cutting-edge systems together.

- Create business processes and workflows visually
- Integrate with SaaS and enterprise applications
- Unlock value from on-premises and cloud applications

Start with a common trigger

Pick from one of the most commonly used triggers, then orchestrate any number of actions using the rich collection of connectors

When a message is received in a Service Bus queue

When a HTTP request is received

When a new tweet is posted

When an Event Grid resource event occurs

Recurrence

When a new email is received in Outlook.com

When a new file is created on OneDrive

When a file is added to FTP server

10. Click on Use sample payload to generate schema

When a HTTP request is received

HTTP POST URL

URL will be generated after save

Request Body JSON Schema

Use sample payload to generate schema

Add new parameter

11. Replace the Request Body JSON Schema with the following code:

```
{
  "schemaId": "Microsoft.Insights/activityLogs",
  "data": {
    "status": "Activated",
    "context": {
      "activityLog": {
        "authorization": {
          "action": "microsoft.insights/activityLogAlerts/write",
          "scope": "/subscriptions/..."
        },
        "channels": "Operation",
        "claims": "...",
        "caller": "logicappdemo@contoso.com",
        "correlationId": "91ad2bac-1afa-4932-a2ce-2f8efd6765a3",
        "description": "",
        "eventSource": "Administrative",
        "eventTimestamp": "2018-04-03T22:33:11.762469+00:00",
        "eventDataId": "ec74c4a2-d7ae-48c3-a4d0-2684a1611ca0",
        "level": "Informational",
        "operationName": "microsoft.insights/activityLogAlerts/write",
        "operationId": "61f59fc8-1442-4c74-9f5f-937392a9723c",
        "resourceId": "/subscriptions/...",
        "resourceGroupName": "LOGICAPP-DEMO",
        "resourceProviderName": "microsoft.insights",
        "status": "Succeeded",
        "subStatus": "",
        "subscriptionId": "...",

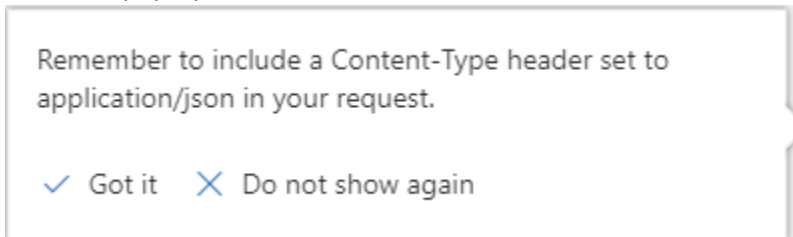
```

```

        "submissionTimestamp": "2018-04-03T22:33:36.1068742+00:00",
        "resourceType": "microsoft.insights/activityLogAlerts"
    },
    "properties": {}
}

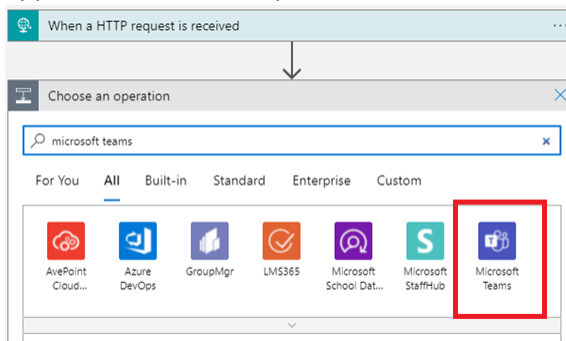
```

12. Click on **Done**.
13. Close the pop-up window. The Azure Monitor alert sets the header.

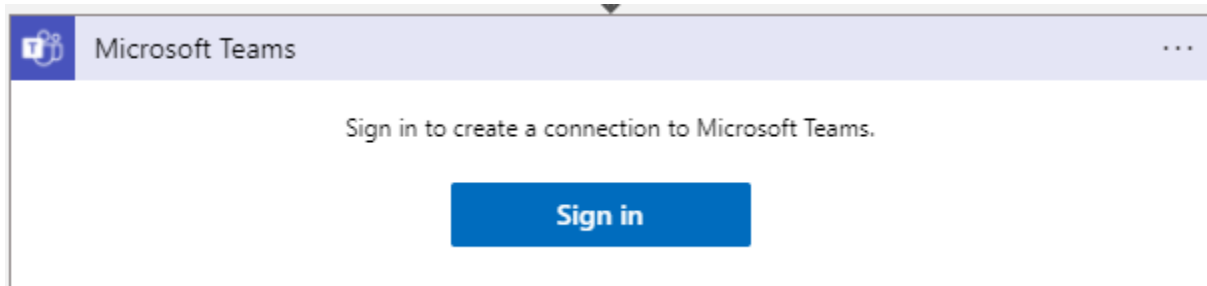


14. Click on + **New step**
15. Search for **Microsoft Teams**

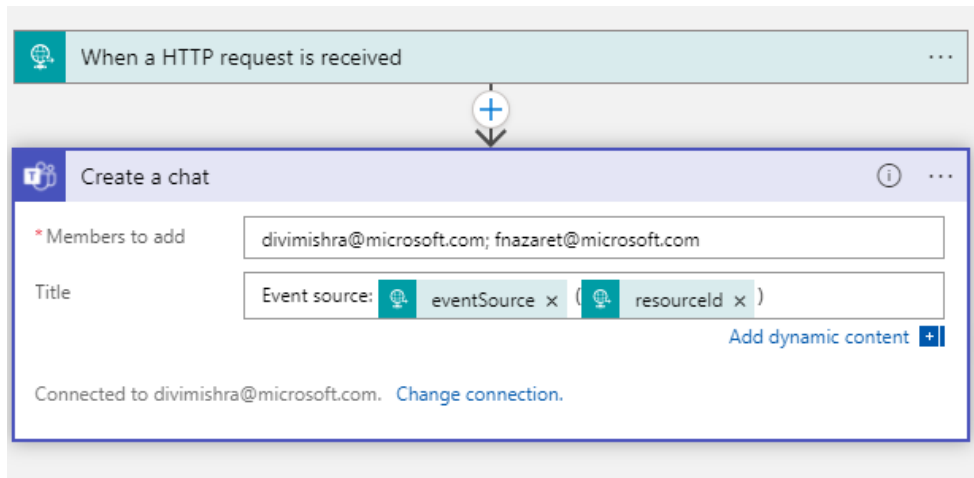
Note: Please search for your respective team collaboration platform. Example: Microsoft Teams, Skype for Business, Slack, etc. This lab shows steps for Microsoft Teams, however, the other apps have a similar step too.



16. Choose **Microsoft Teams – Create a chat** action
17. You will be prompted to sign in



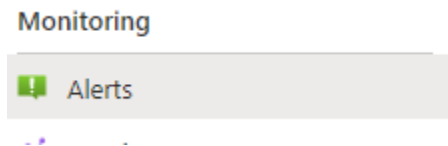
18. Enter your teammates email addresses.
For Title, type:
Event Source: <eventSource> <resourceID>
Here, **eventSource** and **resourceID** is dynamic content



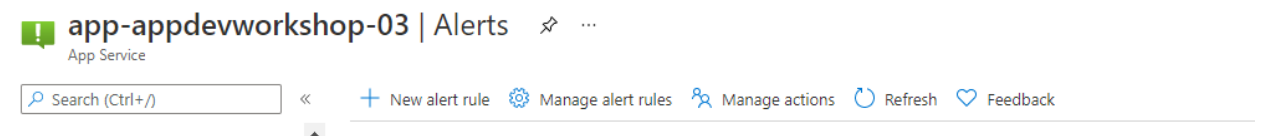
19. Click on **Save**. (You can click on **Run Trigger** for a manual run to check if it works).

Creating the alert

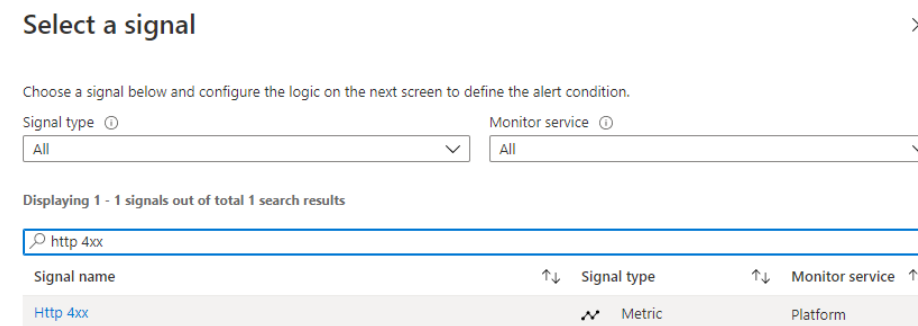
1. Navigate back to your App Service from the Azure Portal.
2. Under Monitoring, select **Alerts**.



3. Select "Add Alert Rule".



4. Select the resource as the App Service.
5. Select the Condition as Http 400 or Http 4xx



- a. Set the Alert logic for "greater than 10"
- b. Leave the granularity and evaluation as their default values.

Note: You can add up to 5 conditions with a static threshold. The alerts will be evaluated with a logical AND.

6. Select **Add Action Groups**.
 - a. Click on **+ Create action group**
 - b. Under **Basics > instance details**, name your action group something relevant. For example, **dotnetapp_alertgroup**
 - c. Under **Actions**:
 - i. Action type: **Logic App**
 - ii. Select the logic app you just created
 - iii. Name: **msteamschat**
 - d. Select **Review + Create**
 - e. Select **Create**
7. Select the alert rule name, description, severity.

Note: The following is a table that describes severity:

- Sev 0 = Critical
- Sev 1 = Error
- Sev 2 = Warning
- Sev 3 = Informational
- Sev 4 = Verbose

8. Select "Create Alert Rule". It may take a few minutes to be configured.

Lab 03: Scale applications with Azure App Service Plan

Scale up your app service

1. Navigate to “Scale up (App Service plan)”.
2. Choose your tier and select **Apply**.

Scale out your app service

1. Navigate to “Scale out (App Service plan)”.
2. **Manual Scale:**
 - a. Configure your instance count to your desired instance count.
 - b. Select Save.
 - c. Review your dashboard to monitor performance.
3. **Automatic Scale (Custom Autoscale):**
 - a. Add a rule to set instance count to 1 if the CPU percentage is less than 10%.

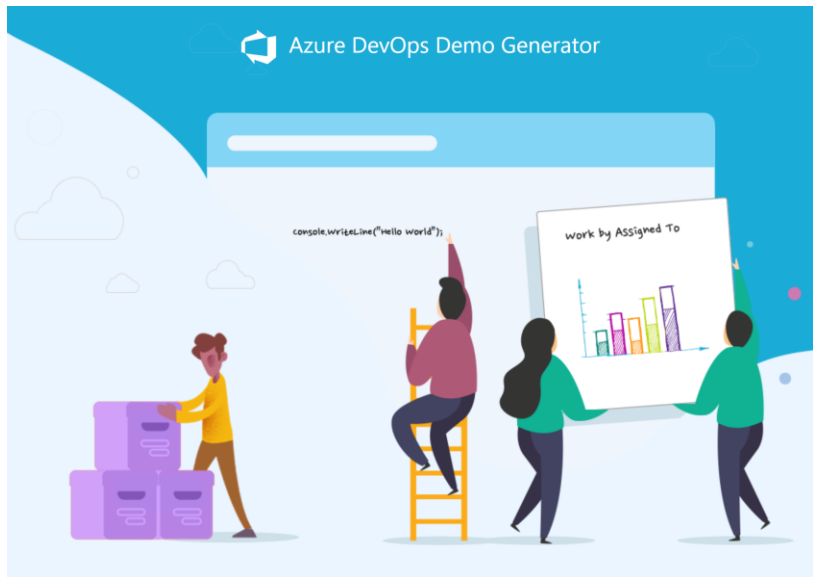
Operator *	Metric threshold to trigger scale action * ⓘ
Less than or equal to ▼	10 ✓
	%
Duration (minutes) * ⓘ	
5 ✓	
Time grain (minutes) ⓘ	Time grain statistic * ⓘ
1	Average ▼
🔧 Action	
Operation *	Cool down (minutes) * ⓘ
Decrease count to ▼	5
Instance count *	
1 ✓	

- b. Click on Save.
- c. Review your dashboard to monitor performance.

Lab 04: Continuous Integration with Azure DevOps

Configure Azure DevOps project

1. We will be using a template that sets everything up in your Azure DevOps organization. Run the template using this link: [Azure DevOps Demo Generator](#)



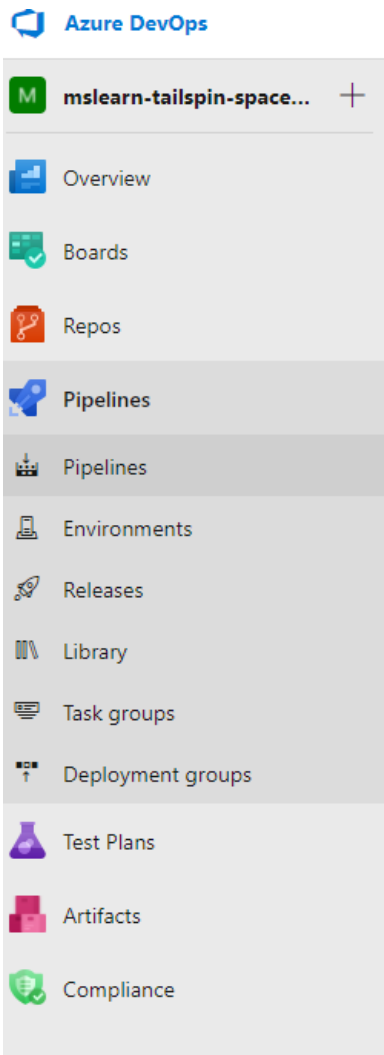
2. Sign in and accept the usage terms.
3. Create a new project with your Azure DevOps (ADO) organization and a project name. For Selected template: under MS learn, chose “Create a build pipeline with Azure Pipelines”. Finally, create project.

New Project Name :	<input type="text" value="Space Game - web - Pipeline"/>
Select Organization :	<input type="text" value="Select Organization"/>
Selected Template :	<input type="button" value="Create a build pipeline with Azure Pip"/> <input type="button" value="Choose template"/>
<input type="button" value="Create Project"/>	

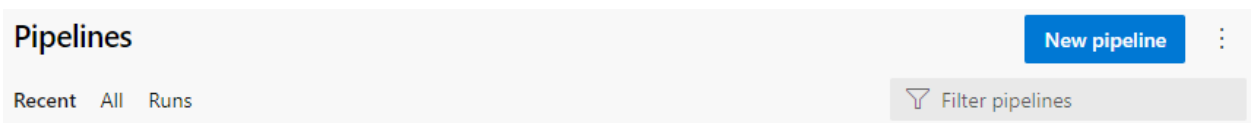
4. Once the deployment is done, select “Navigate to the project”

Create a build pipeline with Azure Pipelines

1. Once your project is deployed, navigate to **Pipelines**.



2. Create a new pipeline.



3. Use the classic editor option at the bottom.

Connect


Select

Configure

Review


New pipeline

Where is your code?

 Azure Repos Git


YAML

Free private Git repositories, pull requests, and code search

 Bitbucket Cloud


YAML

Hosted by Atlassian

 GitHub


YAML


Home to the world's largest community of developers

 GitHub Enterprise Server

YAML


The self-hosted version of GitHub Enterprise

 Other Git
Any generic Git repository

 Subversion
Centralized version control by Apache

[Use the classic editor to create a pipeline without YAML.](#)

4. Select your source as GitHub. You will be required to sign in. You can use **OAuth**.

 We need your authorization to access your repositories

Connection name *


GitHub connection 1


Authorize using OAuth


Or [Authorize with a GitHub personal access token](#)


5. Select your forked repository and the default branch as **main**.


Select a source



Azure Repos Git


GitHub



GitHub Enterprise Server


Subversion


Bitbucket Cloud


Other Git

① Authorized using connection: [GitHub connection 1](#)
[Change](#) ▾

Repository * | [Manage on GitHub](#) 

divim/AzureAppServices

...

Default branch for manual and scheduled builds *


main


...

[Continue](#)


Select “Continue”.

6. Search for **ASP.NET Core**. Apply that as your template.

Select a template asp.net core 


Or start with an  [Empty job](#)

Configuration as code




YAML
Looking for a better experience to configure your pipelines using YAML files?
Try the new YAML pipeline creation experience. [Learn more](#)

Others



ASP.NET Core
Build and test an ASP.NET Core web application.



ASP.NET Core (.NET Framework)
Build an ASP.NET Core web application that targets the full .NET Framework.

7. Your pipeline currently looks like this:

mslearn-tailspin-spacegame-web-ASP...

Tasks Variables Triggers Options History Save & queue Discard Summary Queue

Pipeline
Build pipeline

Get sources
Space Game - web build-agent

Agent job 1
Run on agent

Restore
NET Core

Build
NET Core

Test
NET Core

Publish
NET Core

Publish Artifact
Publish build artifacts

Name *
mslearn-tailspin-spacegame-web-ASP.NET Core-CI

Agent pool [Pool information](#) | [Manage](#)
Azure Pipelines

Agent Specification *
ubuntu-16.04

Parameters | [Unlink all](#)

Project(s) to restore and build
**/*.csproj

Project(s) to test
**/[Tt]ests/*.csproj

8. Click on the “+” to add an agent job.

Agent job 1
Run on agent


+

Add a task to Agent job 1

9. Search and click on **“Add”** for the following tasks:

a. Use .NET Core

Add tasks

 Refresh

 use .net core




Use .NET Core

Acquires a specific version of the .NET Core SDK from the internet or the local cache and adds it to the PATH. Use this task to change the version of .NET Core used in subsequent tasks. Additionally provides proxy support.

Add

b. npm

Add tasks

 Refresh

npm




npm

Install and publish npm packages, or run an npm command. Supports npmjs.com and authenticated registries like Azure Artifacts.

Add

c. gulp

Add tasks

 Refresh

gulp



gulp

Run the gulp Node.js streaming task-based build system

Add

10. Your pipeline will now look like this:

The screenshot shows a build pipeline titled "Agent job 1" with a sub-label "Run on agent". The pipeline consists of the following tasks in order:

- Use .NET Core sdk 5.x** (icon: dotnet) - This task is highlighted with a blue background and a checkmark icon.
- npm install** (icon: npm)
- gulp** (icon: gulp)
- Restore** (icon: dotnet)
- Build** (icon: dotnet)
- Test** (icon: dotnet)
- Publish** (icon: dotnet)
- Publish Artifact** (icon: upload)

11. Click on the “Use .NET Core sdk” task and write the version as “5.x”.

Use .NET Core ⓘ [Link settings](#) [View YAML](#) [Remove](#)

Task version

Display name *

Package to install ⓘ

☐ Use global json ⓘ

Version ⓘ

☐ Include Preview Versions ⓘ

12. Click on the “**Build**” task and edit the “**Arguments**” as follows: `--no-restore --configuration $(buildConfiguration)`

The screenshot shows the Azure DevOps pipeline editor. On the left, a list of tasks is displayed under 'Agent job 1'. The 'Build' task, which uses the '.NET Core' provider, is selected and highlighted. On the right, the configuration for the 'Build' task is shown. The 'Task version' is set to '2.*'. The 'Display name' is 'Build'. The 'Command' is set to 'build'. The 'Path to project(s)' is '**/*.csproj'. The 'Arguments' field contains the text '--no-restore --configuration \$(buildConfiguration)'. Below the arguments, there are expandable sections for 'Advanced', 'Control Options', and 'Output Variables'.

13. Click on the **Test** task and delete it by clicking on “**Remove**”. We will be creating unit tests and adding it to the pipeline in the lab after this.

The screenshot shows the Azure DevOps pipeline editor with the 'Test' task selected. The 'Test' task is highlighted in the task list on the left. On the right, the configuration for the 'Test' task is displayed. The 'Display name' is 'Test'. The 'Command' is set to 'test'. The 'Path to project(s)' is empty. The 'Arguments' field is empty. The 'Publish test results and code coverage' checkbox is checked. The 'Test run title' is empty. At the top right of the configuration panel, there is a red box around the 'Remove' button, which is used to delete the task from the pipeline.

14. Click on the **Publish** task and edit it as follows:

- Unselect the “**Publish web projects**” button.
- Edit the arguments as follows: `--no-build --configuration $(buildConfiguration) --output $(Build.ArtifactStagingDirectory)/$(buildConfiguration)`

The screenshot shows the Azure DevOps task configuration interface. On the left, the 'Pipeline' view shows a sequence of tasks: 'Get sources', 'Agent job 1' (containing 'Use .NET Core sdk 5.x', 'npm install', 'gulp', 'Restore', 'Build', and 'Publish'), and 'Publish Artifact'. The 'Publish' task is selected. On the right, the configuration for the '.NET Core' task is shown. The 'Task version' is set to '2.*'. The 'Display name' is 'Publish'. The 'Command' is 'publish'. The 'Publish web projects' checkbox is unchecked. The 'Path to project(s)' is '**/*.csproj'. The 'Arguments' field contains the command: `--no-build --configuration $(buildConfiguration) --output $(Build.ArtifactStagingDirectory)/$(buildConfiguration)`. The 'Zip published projects' and 'Add project's folder name to publish path' checkboxes are checked. The 'Advanced' section is expanded, showing 'Control Options' and 'Output Variables'.

Tasks Variables Triggers Options History | Save & queue Discard Summary ...

Pipeline Build pipeline

Get sources
divin/mslearn-tailspin-spacegame-web main

Agent job 1
Run on agent

Use .NET Core sdk 5.x
Use .NET Core

npm install
npm

gulp
gulp

Restore
.NET Core

Build
.NET Core

Publish
.NET Core

Publish Artifact
Publish build artifacts

.NET Core Link settings View YAML R

Task version 2.*

Display name *
Publish

Command *
publish

☐ Publish web projects

Path to project(s)
/**/*.csproj

Arguments
--no-build --configuration \$(buildConfiguration) --output \$(Build.ArtifactStagingDirectory)/\$(buildConfiguration)

☒ Zip published projects

☒ Add project's folder name to publish path

Advanced

Control Options

Output Variables

15. Click on the **Publish Artifact** task. Under **Control Options** > **Run this task**, click on the option **“Only when all previous tasks have succeeded”**.

The screenshot shows the 'Publish Artifact' task configuration in the Azure Pipelines interface. The left sidebar lists the pipeline tasks: 'Get sources', 'Agent job 1', 'Use .NET Core sdk 5.x', 'npm install', 'gulp', 'Restore', 'Build', 'Publish', and 'Publish Artifact' (highlighted). The right pane shows the configuration for 'Publish Artifact'.

Task configuration details:

- Task version:** 1.*
- Display name:** Publish Artifact
- Path to publish:** \$(build.artifactstagingdirectory)
- Artifact name:** drop
- Artifact publish location:** Azure Pipelines
- Advanced:** (expanded)
- Control Options:** (expanded) - This section contains the 'Run this task' dropdown, which is set to 'Only when all previous tasks have succeeded'.
- Output Variables:** (expanded)

16. Click on **Agent Job 1** and rename it to **Build the app**

The screenshot shows the 'Build the app' agent job configuration in the Azure Pipelines interface. The left sidebar lists the pipeline tasks: 'Get sources', 'Build the app' (highlighted), 'Use .NET Core sdk 5.x', 'npm install', 'gulp', 'Restore', 'Build', 'Publish', and 'Publish Artifact'. The right pane shows the configuration for 'Build the app'.

Agent job configuration details:

- Display name:** Build the app
- Agent selection:** (expanded)
 - Agent pool:** <inherit from pipeline>
 - Demands:**

Name	Condition	Value
npm	exists	
node.js	exists	
- Execution plan:** (expanded)

17. Under the **Triggers** section, **enable continuous integration**

Tasks Variables **Triggers** Options History | Save & queue Discard Summary Queue ...

Continuous integration

divim/mslearn-tailspin-spacegame-web

Enabled

Pull request validation

divim/mslearn-tailspin-spacegame-web

Disabled

Scheduled

No builds scheduled

Build completion

Build when another build completes

divim/mslearn-tailspin-spacegame-web

☒ Enable continuous integration
☐ Batch changes while a build is in progress

Branch filters

Type

Branch specification

Include

main

+ Add

Path filters

+ Add

18. Under the “Save & Queue”, click on **Save & Queue**.

19. Add a relevant comment and click on **Save & Run**.

Run pipeline

Select parameters below and manually run the pipeline

Save comment

First build pipeline

Agent pool

Azure Pipelines

Agent Specification *

ubuntu-16.04

Branch/tag

main

Select a branch from the list or enter the name of a tag as refs/tags/<tagname>

Commit

Advanced options

Variables

3 variables defined

Demands

This pipeline has no defined demands

☐ Enable system diagnostics

Cancel

Save and run

20. Observe the tasks running:

← Jobs in run #20210904.3 mslearn-tailspin-spacegame-web-ASP.NET Core-CI		
Jobs		
▼	✓ Agent job 1	1m 24s
	✓ Initialize job	5s
	✓ Checkout divim/mslear...	2s
	✓ Nuget Security Analysis...	1s
	✓ Use .NET Core sdk 5.x	7s
	✓ npm install	22s
	✓ gulp	1s
	✓ Restore	10s
	✓ Build	5s
	✓ Publish	3s
	✓ Publish Artifact	<1s
	✓ Component Detection...	23s
	✓ Post-job: Checkout di...	<1s
	✓ Finalize Job	<1s

You have successfully created your build pipeline.

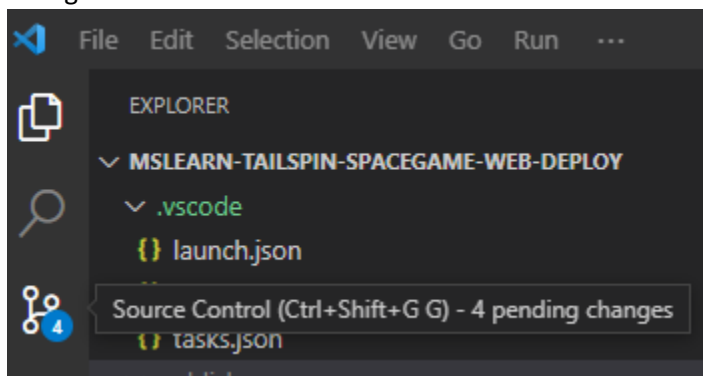
Source Control with GitHub

We just ran the pipeline manually by clicking on Save & Queue. In a real life scenario, we would like this pipeline to run every time we commit changes to our source code on GitHub.

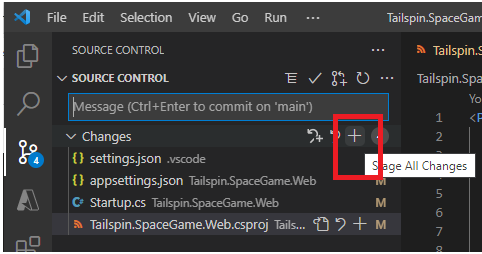
Recall that we had made some changes to the VS Code when we added the Application Insights instrumentation key.

Let's now push it to GitHub.

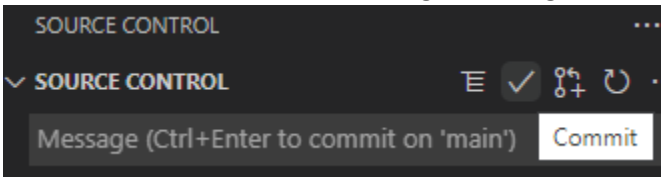
1. Navigate back to your VS Code and notice that your **Source Control** tab highlights pending changes.



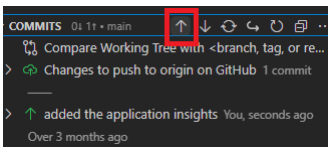
2. Click on the **Source Control** tab
3. Click on + for Staging all the changes



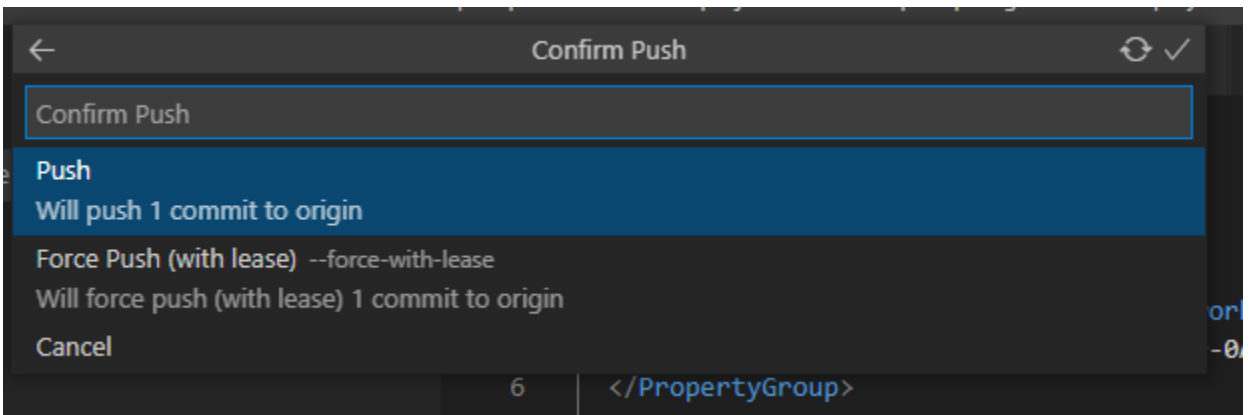
4. Click on the tic mark for Committing the changes



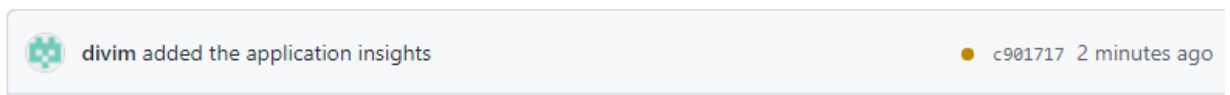
5. Enter a relevant message, such as “added app insights instrumentation key”
6. Under **COMMITTS** (at the bottom), click on the UP arrow. This will **Push** all the changes to GitHub.



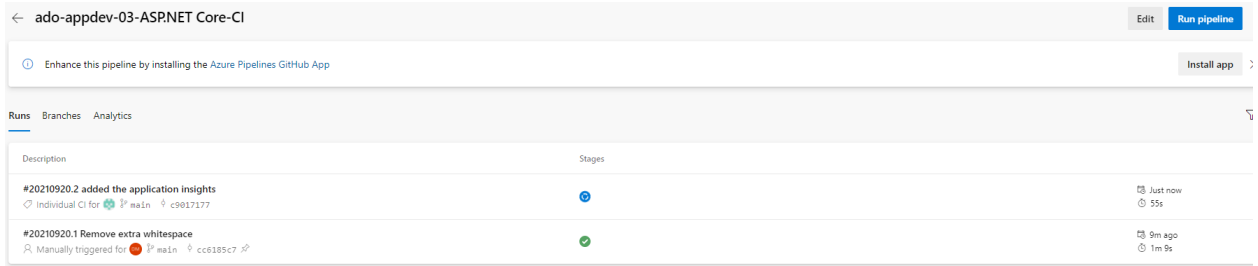
7. Confirm the Push.



8. Navigate to your GitHub repo on github.com. Validate the push.



9. Now that you have pushed new code to GitHub, your build pipeline has been triggered. Navigate back to Azure Pipelines. You will see that your pipeline has been triggered – this time because of a push to the main branch (You may need to refresh to see the new run).



Note:

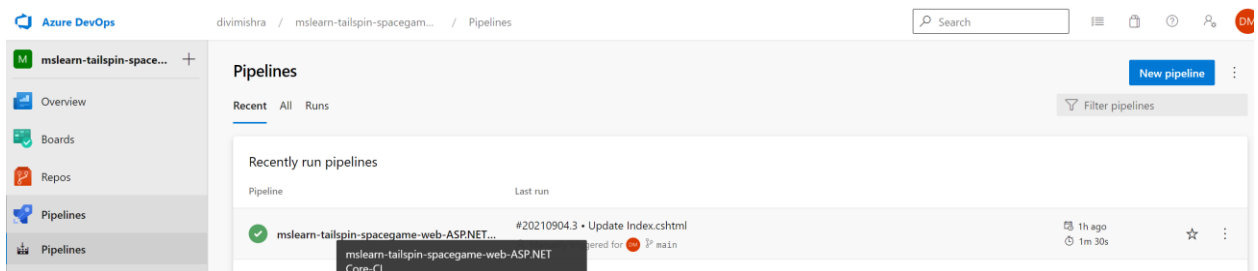
You have now configured a way to reflect changes from your VS Code -> GitHub -> Azure Pipelines.

In the next two labs, we will complete the pipeline as VS Code -> GitHub -> Azure Pipelines -> **Azure App Services**.

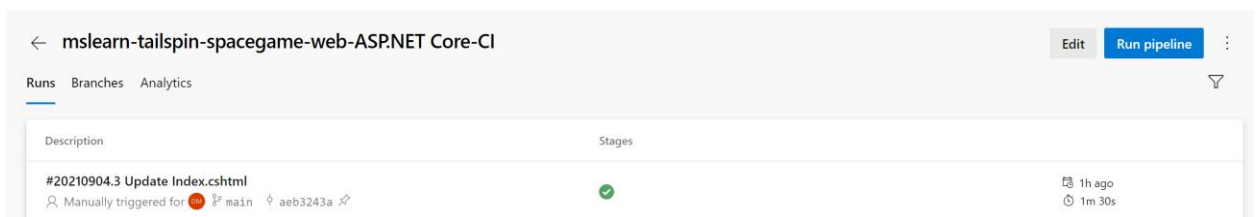
Lab 05: Testing with Azure Pipelines

Quality tests

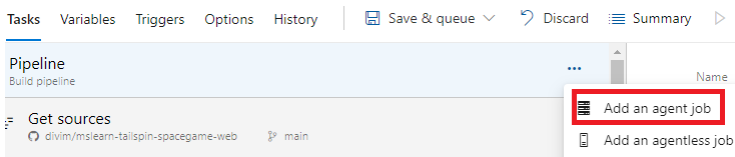
1. Open your Pipeline on dev.azure.com again.



2. Click on **Edit** from the top right to edit the pipeline.



3. Select the three dots next to **Pipeline** and click on **Add an agent job**



4. Click on **Agent job** and rename the job to **Test the app**. Leave the rest as default.

Tasks Variables Triggers Options History | Save & queue Discard Summary Queue ...

Pipeline
Build pipeline

Get sources
divim/mslearn-tailspin-spacegame-web main

Build the app
Run on agent

Use .NET Core sdk 5.x
Use .NET Core

npm install
npm

gulp
gulp

Restore
.NET Core

Build
.NET Core

Publish
.NET Core

Functional tests
Run on agent

Agent job

Display name *
Functional tests

Agent selection ^

Agent pool | Manage

<inherit from pipeline>

Demands

Name	Condition
+ Add	

Execution plan ^

Parallelism

☒ None ☐ Multi-configuration ☐ Multi-agent

Timeout *

5. Click on the + to add a new task to the Functional tests job

Tasks Variables Triggers Options History | Save & queue Discard

Pipeline
Build pipeline

Get sources
divim/mslearn-tailspin-spacegame-web main

Build the app
Run on agent

Use .NET Core sdk 5.x
Use .NET Core

npm install
npm

gulp
gulp

Restore
.NET Core

Build
.NET Core

Publish
.NET Core

Functional tests
Run on agent

Add a task to Functional tests

6. Add the following jobs:

- Use .NET Core

Add tasks | Refresh

use .net core

Use .NET Core

Acquires a specific version of the .NET Core SDK from the internet or the local cache and adds it to the PATH. Use this task to change the version of .NET Core used in subsequent tasks. Additionally provides proxy support.

Add

- .NET Core

.NET Core

Build, test, package, or publish a dotnet application, or run a custom dotnet command

Add

7. Click on the **Use .NET Core** task
 - a. Version: **5.x**
8. Click on the **.NET Core** task
 - a. Display Name: **Quality Test**
 - b. Command: **test**
 - c. Path to project: ****/*Tests.csproj**
 - d. Arguments: **--configuration \$(buildConfiguration)**
 - e. **Select** the Publish test results and code coverage option

.NET Core ⓘ
[Link settings](#)
[View YAML](#)

Task version

2.x ▼

Display name *

Quality Test

Command * ⓘ

test ▼

Path to project(s) ⓘ

**/*QualityTests.csproj

Arguments ⓘ

--configuration \$(buildConfiguration)

☒ Publish test results and code coverage ⓘ

Test run title ⓘ

Advanced ▼

Control Options ▼

Output Variables ▼

9. Click on **Save & Queue** to save your progress and manually trigger the pipeline.
10. Navigate back to the **pipeline summary** of your latest run. Here, you will see the Test and coverage section:

Related

0 work items

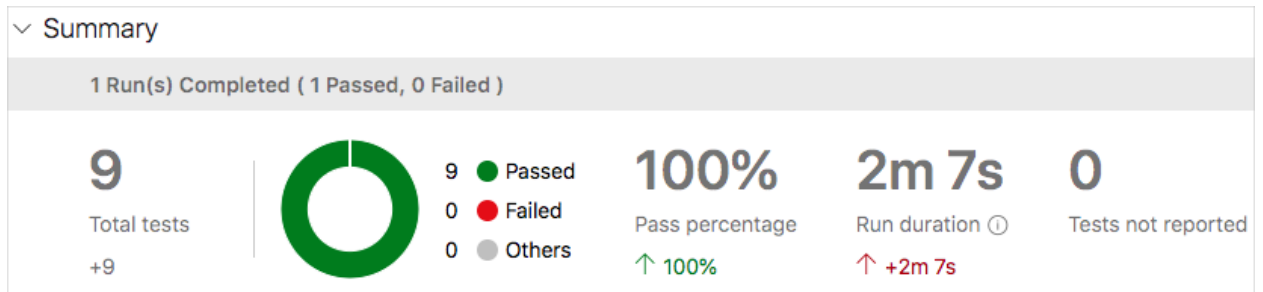
1 published

Tests and coverage

100% passed

Setup code coverage

11. Move to the **Tests** tab to view a summary of the test run.

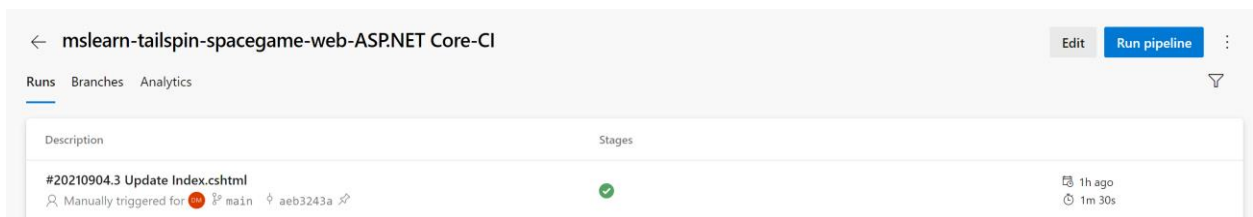


Learn more on how to run UI tests with Selenium: [Perform UI tests with Selenium - Azure Pipelines | Microsoft Docs](#)

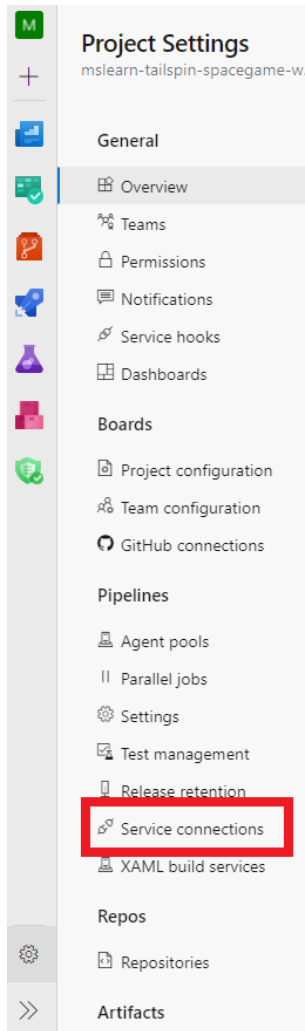
Lab 05: Zero-downtime app deployment with release pipeline

Deploy to staging slot from build pipeline

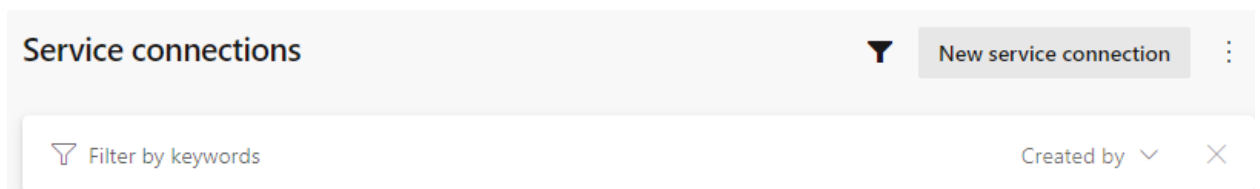
1. Go back to your build pipeline in dev.azure.com.
2. Click on **Edit** from the top right to edit the pipeline.



3. At the bottom left, click on **Project Settings**. Under Pipelines, navigate to Service connections.



4. Click on **New Service Connection**



5. Click on **Azure Resource Manager**. Select Next.
6. Click on **Service Principal (Automatic)**. Select Next.
7. Define your Subscription and Resource group that contains your app service.
8. Give a relevant service connection name.
9. Select **Grant access permission to all pipelines**.

New Azure service connection ✕
 Azure Resource Manager using service principal (automatic)

Scope level

☒ Subscription
☐ Management Group
☐ Machine Learning Workspace

Subscription

MSInternalAccess ▼

Resource group

mslearn-tailspin-spacegame-web ▼

Details

Service connection name

mslearn-tainspin-spacegame

Description (optional)

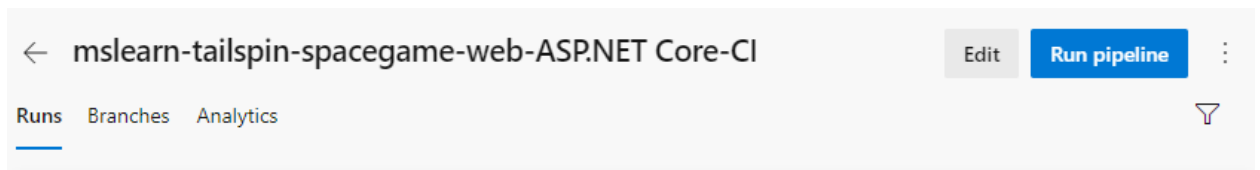
Security

☒ Grant access permission to all pipelines

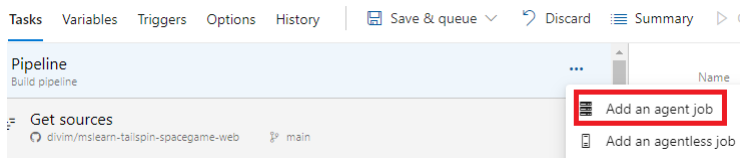
[Learn more](#) [Troubleshoot](#) Back Save

10. Select **Save**.

11. Navigate back to your pipeline and select **Edit**.

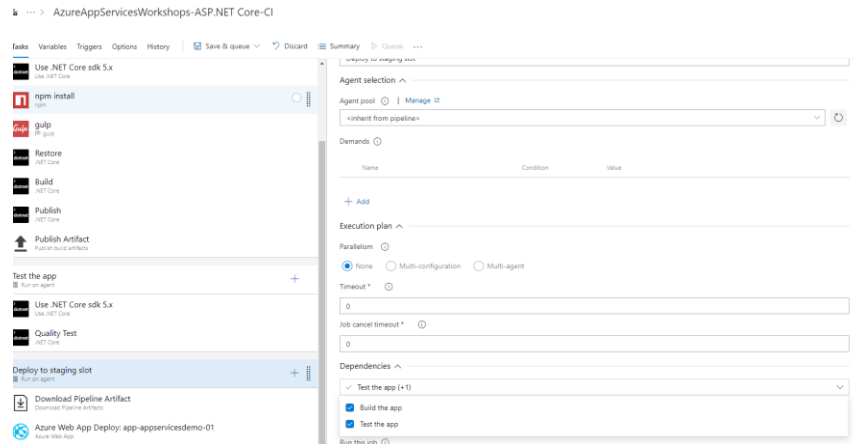


12. Select the three dots next to **Pipeline** and click on **Add an agent job**



13. Click on **Agent job**

- Rename the job to **Deploy to Staging environment**
- Choose dependencies as both the other build pipelines:

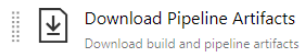


14. Add the following agent jobs:

a. Download pipeline artifact

Add tasks | Refresh

download pipeline



Download Pipeline Artifacts
Download build and pipeline artifacts

b. Azure App Service Deploy

Add tasks | Refresh

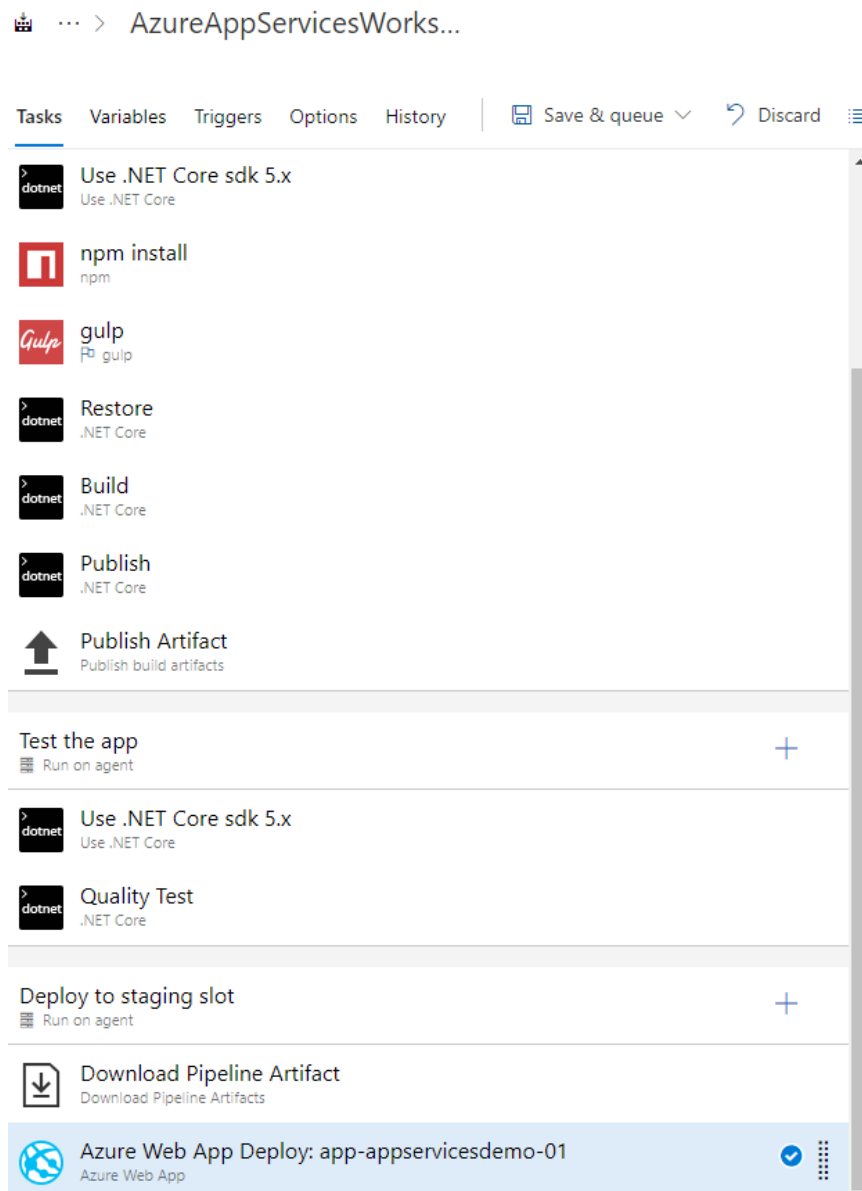
azure app service deploy



Azure App Service deploy

Deploy to Azure App Service a web, mobile, or API app using Docker, Java, .NET, .NET Core, Node.js, PHP, Python, or Ruby

15. Ensure that your pipeline now looks like this:



16. Click on the **Download Pipeline Artifact** task to edit the task as follows:

- Destination directory: \$(Pipeline.Workspace)
- Leave the Artifact name and matching patterns empty

Summary Queue ...

Download Pipeline Artifacts ⓘ

Task version 2.* ▾

Display name *
Download Pipeline Artifact

Download artifacts produced by * ⓘ
☒ Current run ☐ Specific run

Artifact name ⓘ

Matching patterns ⓘ

Destination directory * ⓘ
\$(Pipeline.Workspace)




Control Options ▾

Output Variables ▾

17. Click on **Azure App Service Deploy** to edit it as follows:


- a. Connection type: Azure Resource Manager
- b. Subscription: <name-of-your-service-connection>
- c. App Service type: *as defined during your app service definition above*
- d. App Service name: *as defined during your app service definition above*
- e. Select **Deploy to Slot or App Service Environment**
- f. Resource group: RG of your app service
- g. Slot: **staging**
- h. Package or folder: **\$(Pipeline.Workspace)/**/Tailspin.SpaceGame.Web.zip**



Summary Queue ...


Azure App Service deploy   



Task version 4.*


Display name *
Azure App Service Deploy: azqatar-mslearn-tailspin-spacegame-web



Connection type * 
Azure Resource Manager



Azure subscription *  | [Manage](#) 
mslearn-tailspin-spacegame-rg
Scoped to resource group 'mslearn-tailspin-spacegame-web'


App Service type * 
Web App on Windows

App Service name * 
azqatar-mslearn-tailspin-spacegame-web 

☒ Deploy to Slot or App Service Environment 

Resource group * 
mslearn-tailspin-spacegame-web 

Slot * 
staging 

Virtual application 

18. Select **Save & Queue**.




19. Select **Save & Run**.

20. Once the pipeline is done running, go back to Azure Portal > App Service > your app service > Deployment Slots








21. Click on the **staging slot**


22. Click on **Browse**

Home > [azqatar-mslearn-tailspin-spacegame-web](#) >

 **staging (azqatar-mslearn-tailspin-spacegame-web/staging)**  ... 

App Service (Slot)

Search (Ctrl+ /) <<  Browse  Stop  Swap  Restart  Delete |  Refresh  Get publish profile ...

 Click here to access Application Insights for monitoring and profiling for your ASP.NET Core app. →

^ Essentials [JSON View](#)

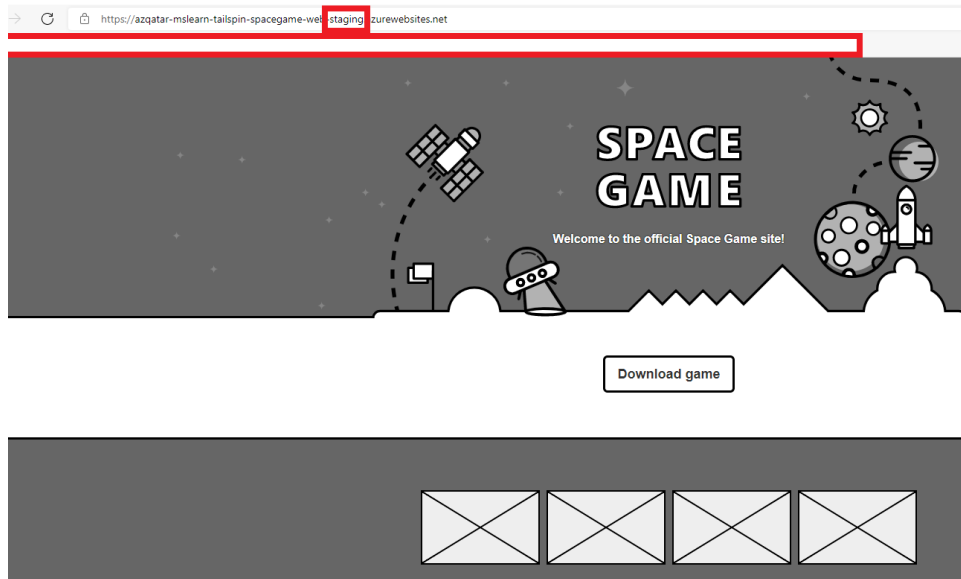
Resource group (change)	URL
mslearn-tailspin-spacegame-web	https://azqatar-mslearn-tailspin-spacegame-...
Status	Health Check
Running	Not Configured
Location	App Service Plan
West Europe	ASP-mslearntailspinspacegameweb-9893 (S1:...
Subscription (change)	
MSInternalAccess	

Overview

- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Security
- Events (preview)

Deployment

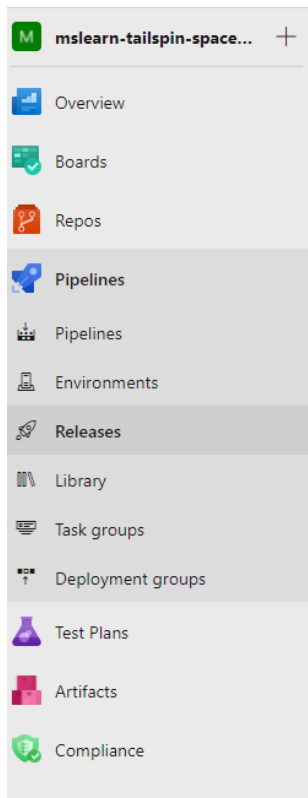
23. Your app has been deployed successfully to the staging slot. Note that the name of the URL is simply <app-service-name>-staging.azurewebsites.net



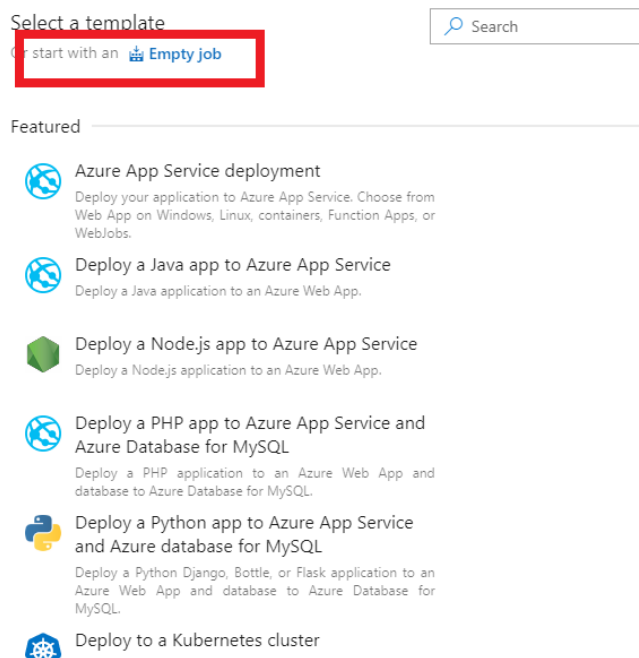
24. Simply remove the “-staging” from the URL to view the production slot website.

Build a release pipeline: zero-downtime deployment with slot swapping

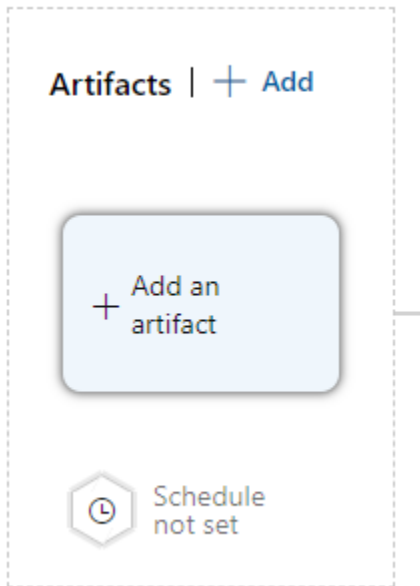
1. Navigate to your Azure DevOps project > **Pipelines** > **Releases**



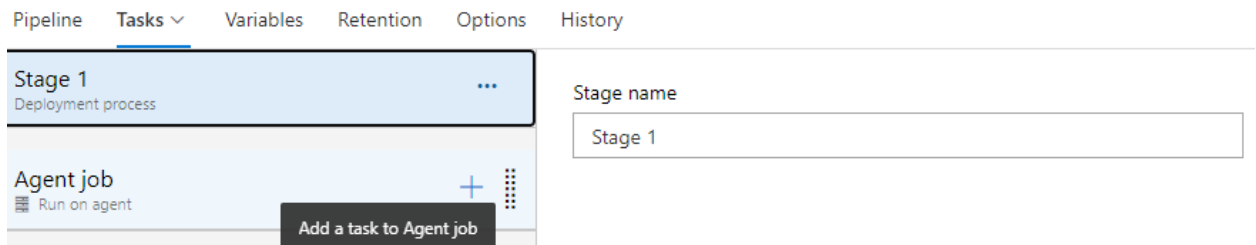
2. Select “New pipeline”.
3. For the template, **start with an empty job**



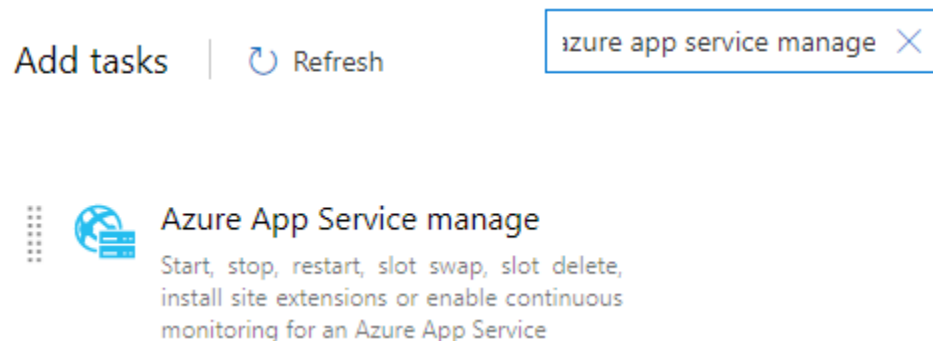
- Click on **Add an artifact**



- Select your source type as **Build**.
- Select your Source build pipeline from the previous lab.
- Leave the default values for the default version and source alias.
- Select **Add**.
- Select the **Stage 1**.
- Add a task to agent job by clicking on the “+”.



- Search for **Azure App Service manage** and click **Add**.



12. Click on the task to edit the task as follows:

- a. Select your service connection under Azure Subscription.
- b. Select the action as **Swap slots**.
- c. Select your app service for the app service name and its associated resource group.
- d. Select **staging** as your source slot.
- e. Select **Swap with production**.

13. Add another task for **Azure App Service manage**.

The screenshot shows the Azure DevOps pipeline editor interface. On the left, a sidebar displays the pipeline structure: 'Stage 1' (Deployment process), 'Agent job' (Run on agent), and two 'Swap Slots' tasks. The first 'Swap Slots' task is selected, showing its details: 'azqatar-mslearn-...' and 'Azure App Service manage'. The second 'Swap Slots' task is also shown, with a status icon indicating 'Some settings need attention'. The main panel on the right shows the configuration for the 'Azure App Service manage' task. The task version is set to '0.*'. The display name is 'Swap Slots:'. The Azure subscription is set to a dropdown menu, with a red border and a message 'This setting is required.' The action is set to 'Swap Slots'. The App Service name is set to a dropdown menu, with a red border and a message 'This setting is required.' The Resource group is set to a dropdown menu, with a red border and a message 'This setting is required.' The Source Slot is set to a dropdown menu, with a red border and a message 'This setting is required.' The 'Swap with Production' checkbox is checked.

14. Select the task to edit it as follows:

- a. Select your service connection for the Azure subscription
- b. Select your action as **Delete Slot**
- c. Select your app service name and resource group
- d. Select the slot to be deleted as **staging**

Stage 1
Deployment process

Agent job
Run on agent

Swap Slots: azqatar-mslearn-tailspin-spacegame-web
Azure App Service manage

Delete Slot: azqatar-mslearn-tailspin-spacegame-web
Azure App Service manage

Azure App Service manage ⓘ

Task version 0.*

Display name *
Delete Slot: azqatar-mslearn-tailspin-spacegame-web

Azure subscription * ⓘ | Manage ⓘ
mslearn-tailspin-spacegame-rg ⓘ

ⓘ Scoped to resource group 'mslearn-tailspin-spacegame-web'

Action ⓘ
Delete Slot

App Service name * ⓘ
azqatar-mslearn-tailspin-spacegame-web ⓘ

Resource group * ⓘ
mslearn-tailspin-spacegame-web ⓘ

Slot *
staging ⓘ

Control Options ▾

Output Variables ▾

15. Select **Save**.

16. Select **Create release**.

All pipelines > New release pipeline

Save Create release View releases ...

Release Release-1 has been created

17. Observe the tasks on the release pipeline.

New release pipeline > Release-1 > Stage 1 ✓ Succeeded

← Pipeline Tasks Variables Logs Tests | ☁ Deploy ⏸ Cancel ↺ Refresh ⬇ Download all logs ...

Deployment attempt #2
Succeeded

Agent job
Succeeded

Agent job
Pool: Azure Pipelines · Agent: Hosted Agent
Started: 9/5/2021, 12:00:10 AM
... 2m 55s

✓ Initialize job · succeeded	8s
✓ Download artifact - _mslearn-tailspin-spa... · succeeded	1m 8s
✓ Swap Slots: azqatar-mslearn-tailspin-spa... · succeeded	1m 29s
✓ Delete Slot: azqatar-mslearn-tailspin-spa... · succeeded	9s
✓ Finalize Job · succeeded	<1s

18. Once the pipeline has been completed, navigate back to Azure Portal.

19. Under deployment slots, you will notice that the slot is now deleted.

azqatar-mslearn-tailspin-spacegame-web | Deployment slots

App Service

Search (Ctrl+/) << Save Discard Add Slot Swap Logs Refresh

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Security
Events (preview)

Deployment

Quickstart
Deployment slots
Deployment Center

You haven't added any deployment slots. Click here to get started. →

Deployment Slots

Deployment slots are live apps with their own hostnames. App content and configurations elements can be swapped between two deployment slots, including the production slot.

NAME	STATUS	APP SERVICE PLAN	TRAFFIC %
azqatar-mslearn-tailspin-spacegame-web PRODUCTION	Running	ASP-mslearntailspinspacegameweb-9893	100

Now, when you push a commit to GitHub's main branch, your build pipeline will be triggered and deploy the app to your staging slot. If successful, your release pipeline will triggered to swap the staging and production slot. Finally, the pipeline will delete the staging slot to stop incurring any charges.

Notes:

You can choose to enable or disable continuous deployment for your release pipeline. This will allow you to choose whether you want the release to happen automatically or with a manual check.

Pipeline Tasks Variables Retention Options History

Artifacts | + Add

Stages | + Add

Build artifact: _mslearn-tailspin-spacegame-web- (icon highlighted with a red box)

Schedule not set

Stage 1
1 job, 2 tasks

Continuous deployment trigger
Build: _mslearn-tailspin-spacegame-web-ASP.NET Core-CI
☒ Disabled
☐ Enabling the trigger will create a new release every time a new build is available.

Pull request trigger
Build: _mslearn-tailspin-spacegame-web-ASP.NET Core-CI
☒ Disabled
☐ Enabling this will create a release every time a selected artifact is available as part of a pull request workflow

Optional labs

Please note that these optional labs will take more time than the other labs. These labs may also require a higher SKU for the app service plan used.

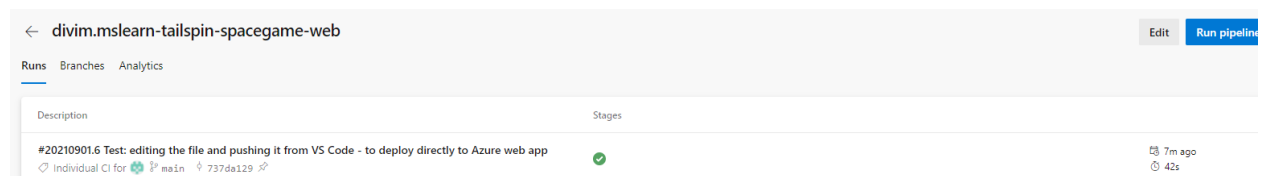
Optional Lab 01: Enabling continuous integration with YAML files

Create a build pipeline without classic editor

1. Go back to your Azure DevOps project created in Lab 01.
2. Navigate to **Pipelines** and create a pipeline.
3. Under Connect, click on **GitHub**.
4. Under Select, browse to your forked repository.
5. Under Configure, select **"ASP.NET Core"**. (Please don't select ASP.NET Core (.NET Framework))
6. Select **"Save and Run"**. Set a commit message. Select **"Save and run"** again.
7. Observe as all the tasks run. Your first pipeline has been created successfully.

VS Code to Azure App Services: enabling Continuous Integration

1. Open the Azure App Service resource from the Azure portal.
2. Under Deployment, go to the Deployment Center.
3. Select your source as GitHub.
4. Authorize AzureAppService to access your GitHub and select your enterprise, repository and branch.
5. Go to VS Code, make a change to the code (say index.cshtml). Then, stage, commit and push the change alongside a relevant commit message.
6. Navigate to your app in the VS Code Azure App Services Extension. Right-click and select **"Browse website"**. You will see the changes reflected.
7. You can now edit files in VS Code. Once you commit your changes to the main branch, the build pipeline will get triggered and your changes will be reflected on successful build.



You now have a basic CI/CD pipeline from VS Code, to GitHub, and finally to Azure App Service.

Note: As an alternative to Azure Repos, you can use GitHub Actions. Please follow this [documentation on steps to configure it: Configure CI/CD with GitHub Actions - Azure App Service | Microsoft Docs](#)

8. Modify **azure-pipelines.yml** file as follows to publish an artifact as a part of your build pipeline:

```
trigger:
- '*'

pool:
  vmImage: 'ubuntu-20.04'
  demands:
  - npm

variables:
  buildConfiguration: 'Release'
  wwwrootDir: 'Tailspin.SpaceGame.Web/wwwroot'
  dotnetSdkVersion: '5.x'

steps:
- task: UseDotNet@2
  displayName: 'Use .NET SDK $(dotnetSdkVersion)'
  inputs:
    version: '$(dotnetSdkVersion)'

- task: Npm@1
  displayName: 'Run npm install'
  inputs:
    verbose: false

- script: './node_modules/.bin/node-sass $(wwwrootDir) --output $(wwwrootDir)'
  displayName: 'Compile Sass assets'

- task: gulp@1
  displayName: 'Run gulp tasks'

- script: 'echo "$(Build.DefinitionName), $(Build.BuildId), $(Build.BuildNumber)" > buildinfo.txt'
  displayName: 'Write build info'
  workingDirectory: $(wwwrootDir)

- task: DotNetCoreCLI@2
  displayName: 'Restore project dependencies'
  inputs:
    command: 'restore'
    projects: '**/*.csproj'
```

```




- task: DotNetCoreCLI@2
  displayName: 'Build the project - $(buildConfiguration)'
  inputs:
    command: 'build'
    arguments: '--no-restore --configuration $(buildConfiguration)'
    projects: '**/*.csproj'

- task: DotNetCoreCLI@2
  displayName: 'Publish the project - $(buildConfiguration)'
  inputs:
    command: 'publish'
    projects: '**/*.csproj'
    publishWebProjects: false
    arguments: '--no-build --configuration $(buildConfiguration) --output $(Build.ArtifactStagingDirectory)/$(buildConfiguration)'
    zipAfterPublish: true

- task: PublishBuildArtifacts@1
  displayName: 'Publish Artifact: drop'
  condition: succeeded()

```

9. Push the change as a commit to the main branch. This will trigger the Azure Pipeline.
10. Go to Azure Pipelines and observe the build through the steps. You'll see that you now have a published artifact.
11. Select the artifact and expand the drop folders. You'll see a .zip file that contains your build app and its dependencies.

Published	
Name	Size
▼  drop	850 KB
▼  Release	850 KB
 Tailspin.SpaceGame.Web.zip	850 KB

Note: You can build multiple configurations within the pipeline by defining a **template**. Read more:

[Exercise - Build multiple configurations by using templates - Learn | Microsoft Docs](#)

Optional Lab 02: Protect App Services with other WAF Options

Alternatively, you can set up other WAF options from the [Azure Marketplace](#). Here is a documentation

on how you can integrate [a Barracuda WAF for Azure: Configure a WAF - Azure App Service Environment](#)

[| Microsoft Docs](#)

Optional Lab 03: Azure Defender for App Services

1. Navigate to the Security Center from the left menu on the Azure Portal.
2. Select “Pricing & Settings” under Management.
3. Select the subscription with your application.
4. Enable Defender for all resources (strongly recommended).

Settings | Azure Defender plans MSInternalAccess

Search (Ctrl+/) Save

Settings

- Azure Defender plans
- Auto provisioning
- Email notifications
- Integrations
- Workflow automation
- Continuous export
- Cloud connectors

Azure Defender provides enhanced security. Learn more >

Azure Defender off	Azure Defender on
✓ Continuous assessment and security recommendations	✓ Continuous assessment and security recommendations
✓ Azure Secure Score	✓ Azure Secure Score
✗ Just in time VM Access	✓ Just in time VM Access
✗ Adaptive application controls and network hardening	✓ Adaptive application controls and network hardening
✗ Regulatory compliance dashboard and reports	✓ Regulatory compliance dashboard and reports
✗ Threat protection for Azure VMs and non-Azure servers (including Server EDR)	✓ Threat protection for Azure VMs and non-Azure servers (including Server EDR)
✗ Threat protection for supported PaaS services	✓ Threat protection for supported PaaS services

Azure Defender plan will apply to: 9 resources in this subscription

Select Azure Defender plan by resource type **Enable all**

5. Click on “Save” to enable Azure Defender.
6. Your app service is now under Azure Defender.

823 Active alerts 55 Affected resources

Active alerts by severity: High (50) Medium (626) Low (138)

Search by ID, title, ... Subscription: ASC DEMO Status: Active

Severity: Low, Medium, High Time: Last month Add filter

Severity	Alert title	Affected resource	Activity start time	MITRE ATT&C	Status
High	Sample alert	Sample-Storage	01/25/21, 11:13 AM	Pre-attack	Active
High	Sample alert	Sample-Storage	01/25/21, 11:13 AM	Exfiltration	Active
High	Dangling DNS Record	dangling	01/24/21, 12:41 PM		Active
High	Azure Security Center...	OpenShift-Cluster-1	01/20/21, 01:26 PM	Persistence	Active
High	Azure Security Center...	ASC-Arc-OpenShift...	01/19/21, 07:22 PM	Persistence	Active
High	Azure Security Center...	ASC-Arc-K8S-demo	01/19/21, 01:16 PM	Persistence	Active
High	Azure Security Center...	ASC-Arc-Demo-clust...	01/14/21, 04:50 PM	Persistence	Active
High	Azure Security Center...	aks-engine-arc-test-2	01/14/21, 01:26 PM	Persistence	Active
High	Azure Security Center...	aks-engine-arc-test-2	01/14/21, 01:26 PM	Persistence	Active
High	Azure Security Center...	aks-engine-arc-test-2	01/14/21, 01:26 PM	Persistence	Active
High	Azure Security Center...	microsoftazuredefe...	01/14/21, 11:12 AM	Persistence	Active
High	Digital currency mini...	app-ix	01/13/21, 12:38 PM	Execution	Active
High	Digital currency mini...	app-ix	01/13/21, 12:38 PM	Execution	Active

Dangling DNS Record detected on App Service

High Severity Active Status Activity time: 01/24/21, 12:41 PM

Alert description

A DNS record that points to a recently deleted App Service resource has been detected. This is also known as 'Dangling DNS' entry and leaves you susceptible to a subdomain takeover. Subdomain takeovers enable malicious actors to redirect traffic intended for an organization's domain to a site performing malicious activity.

Affected resource

dangling Web application base
Playground Subscription

Azure Defender detects a multitude of threats to your App Service resources by monitoring:

- a. the VM instance in which your App Service is running, and its management interface

- b. the requests and responses sent to and from your App Service apps
- c. the underlying sandboxes and VMs
- d. App Service internal logs - available thanks to the visibility that Azure has as a cloud provider

You can detect DNS dangling and threats by MITRE ATT&CK tactics.

Optional Lab 04: Managing technical debt with SonarQube and Azure DevOps

Add SonarQube to your build pipeline to:

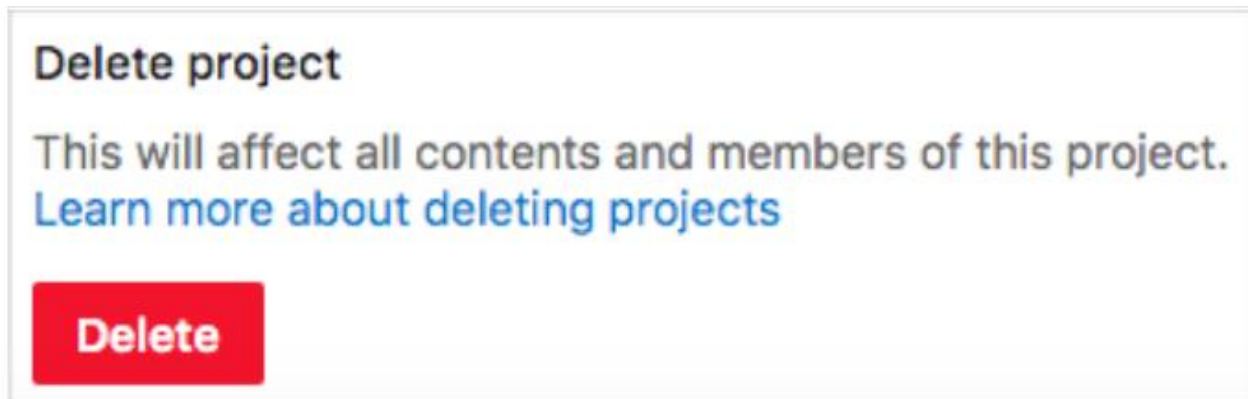
- Detect bugs
- Code smells
- Security vulnerabilities
- Centralize qualities

To analyze SonarQube projects, you must provision up your SonarQube server and set up your project.

Step-by-step lab for adding SonarQube to your build pipeline for increased testing: [Managing technical debt with SonarQube and Azure DevOps | Azure DevOps Hands-on-Labs \(azuredevopslabs.com\)](https://azuredevopslabs.com/labs/04-managing-technical-debt-with-sonarqube-and-azure-devops/)

Clean-up your environment

1. Delete the Azure DevOps project, including what's on Azure Boards and Azure Pipelines.
2. Navigate to dev.azure.com.
3. Navigate to your project.
4. Select Project Settings in the lower corner.
5. In the project details area, select **Delete**.



6. Enter the project name and confirm deletion.
7. Navigate to Azure Portal and delete the app service or the resource group.

Your project is now deleted.

Congratulations! You just reached the end of workshop labs.

Resources

- Intro to GitHub Lab: [Introduction to GitHub | GitHub Learning Lab](#)
- Creating pull requests, reviewing, creating issues, and other features on VS Code: [Working with GitHub in Visual Studio Code](#).
- Importing your project to GitHub: [How do I migrate an existing project to GitHub? - Learn | Microsoft Docs](#)
- For enabling live telemetry through instrumentation key on other code frameworks: [What is Azure Application Insights? - Azure Monitor | Microsoft Docs](#)
- For enabling continuous monitoring through Azure DevOps pipeline directly: [Continuous monitoring of your DevOps release pipeline with Azure Pipelines and Azure Application Insights - Azure Monitor | Microsoft Docs](#)
- Learn more about continuous monitoring: [Continuously monitor applications and services - Learn | Microsoft Docs](#)
- Best practices for Autoscale: [Best practices for autoscale - Azure Monitor | Microsoft Docs](#)
- Confused about whether or not to choose Azure App Service? Choose your candidate with this document: [Choosing an Azure compute service - Azure Architecture Center | Microsoft Docs](#)
- Choose the right App Service plan: [App Service plans - Azure App Service | Microsoft Docs](#)
- Security recommendations for App Service: [Security recommendations - Azure App Service | Microsoft Docs](#)
- Tutorial on how to deploy ASP.NET Core with Azure SQL Database app in Azure App Service: [Tutorial: ASP.NET Core with Azure SQL Database - Azure App Service | Microsoft Docs](#)
- Tutorials to use your App Gateway to:
 1. Secure by SSL: [Tutorial: Configure TLS termination in portal - Azure Application Gateway | Microsoft Docs](#)
 2. Host multiple sites: [Tutorial: Hosts multiple web sites using the Azure portal - Azure Application Gateway | Microsoft Docs](#)
 3. Route by URL: [Tutorial: URL path-based routing rules using portal - Azure Application Gateway | Microsoft Docs](#)
 4. Redirect web traffic: [Tutorial: URL path-based redirection using CLI - Azure Application Gateway | Microsoft Docs](#)