

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. VS A [GitHub](#) account
2. An [Azure DevOps](#) setup
3. An active [Azure](#) subscription
4. [.NET 5.0 SDK](#)
5. [Git](#)

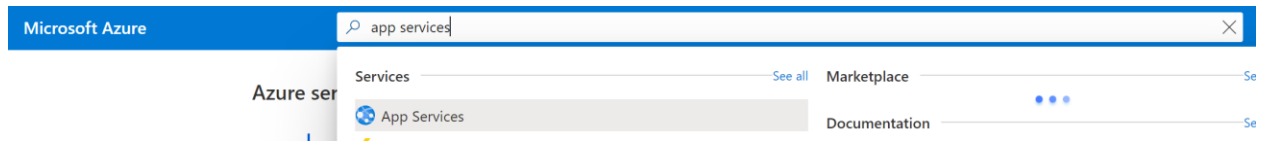
LAB 01: AZURE APP SERVICE	5
Create an Azure App Service	5
Fork the web project to your GitHub account	6
Create deployment slots	10
 LAB 02: MONITOR YOUR APP WITH AZURE APPLICATION INSIGHTS.....	 12
Enabling live telemetry through instrumentation key using VS Code	12
Monitor apps	15
Configure alerts for Azure App Service.....	16
 LAB 03: SCALE APPLICATIONS WITH AZURE APP SERVICE PLAN	17
Setting up a dashboard to review changes.....	17
Scale up your app service.....	17
Scale out your app service	17
 LAB 04: CONTINUOUS INTEGRATION WITH AZURE DEVOPS.....	19
Configure Azure DevOps project	19
Create a build pipeline with Azure Pipelines	20
Source Control with GitHub	29
Add unit tests	30
 LAB 05: ZERO-DOWNTIME APP DEPLOYMENT WITH RELEASE PIPELINE.....	 38
Deploy to staging slot from build pipeline	38
Build a release pipeline: zero-downtime deployment with slot swapping	43

OPTIONAL LABS.....	48
Optional Lab 01: Enabling continuous integration with YAML files	48
Optional Lab 02: Protect App Services with other WAF Options	50
Optional Lab 03: Azure Defender for App Services.....	51
Optional Lab 04: Managing technical debt with SonarQube and Azure DevOps.....	52
CLEAN-UP YOUR ENVIRONMENT	53
RESOURCES.....	54

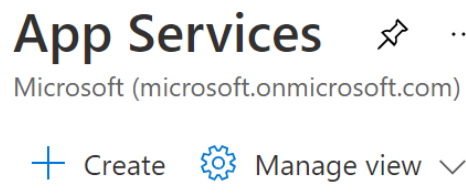
Lab 01: Azure App Service

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 **Basic**, enter the following details:
 - a. Select the relevant RG 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. Click on **Review + Create**
6. Click on **Create**

Create Web App ...

all your resources.

Subscription * ⓘ

Resource Group * ⓘ
[Create new](#)

Instance Details

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

Name *

Publish * ☒ Code ☐ Docker Container

Runtime stack *

Operating System * ☐ Linux ☒ Windows

Region *
[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) * ⓘ
[Create new](#)

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

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

Fork the web project to your GitHub account

- Go to [GitHub](#) and sign in
- Visit the [Space Game](#) web project (ASP.Net Core application)

MicrosoftDocs / mslearn-tailspin-spacegame-web

Watch 16 Star 74 Fork 16.6k

<> Code Issues 1 Pull requests 86 Actions Projects Wiki Security Insights

main 8 branches 0 tags

Go to file Add file Code

tpetchel Merge branch 'net5.0' #236243 on May 25 44 commits

File	Commit	Time
.vscode	Add tasks.json	3 months ago
Tailspin.SpaceGame.Web	Migrate from Newtonsoft.Json to System.Text.Json	3 months ago
.gitignore	Initial	3 years ago
LICENSE	Initial commit	3 years ago
LICENSE-CODE	Initial commit	3 years ago
README.md	Initial commit	3 years ago
Tailspin.SpaceGame.Web.sln	Rename solution file	3 years ago
gulpfile.js	Use gulp-clean-css	2 years ago
package-lock.json	Merge pull request #4937 from MicrosoftDocs/dependabot/npm_and_ya...	3 months ago
package.json	Bump node-sass version	15 months ago

Contributing

This project welcomes contributions and suggestions. Most contributions require you to agree to a Contributor License Agreement (CLA) declaring that you have the right to, and actually do, grant us the rights to use your contribution. For details, visit <https://cla.microsoft.com>.

About

Code used in Microsoft Learn modules to support Azure DevOps

[aka.ms/devops-build](#)

Readme

View license

Releases

No releases published

Packages

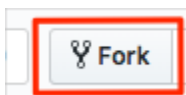
No packages published

Contributors 5

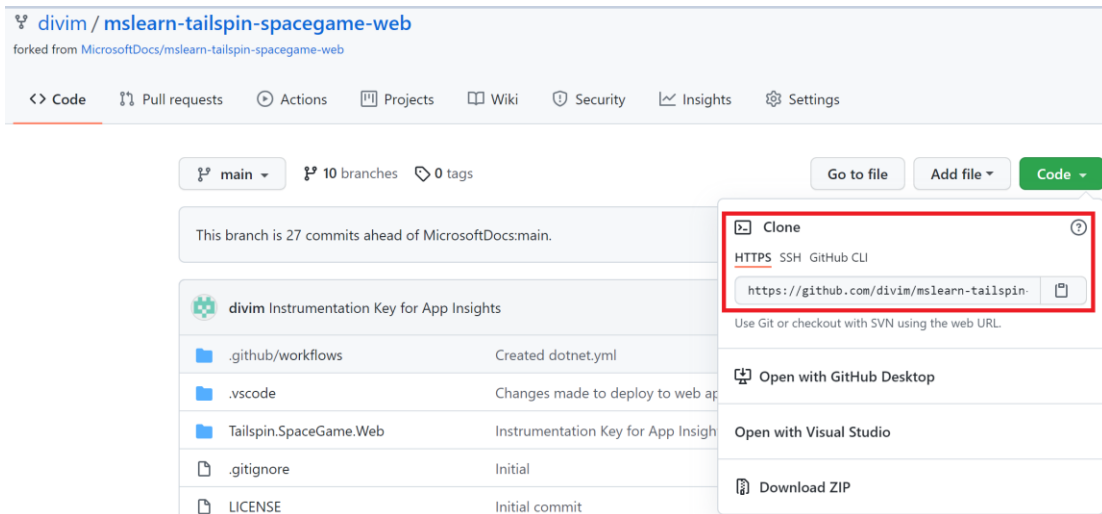
Languages

HTML 42.8% CSS 8.1% JavaScript 3.2%

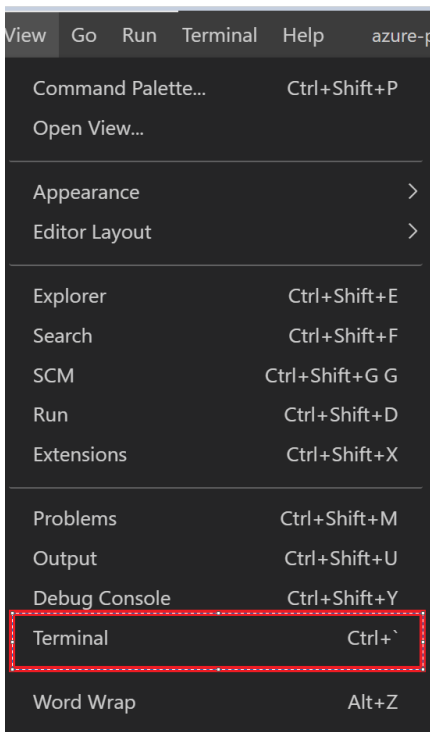
- From the top right corner, fork your own copy of the repo to your account.



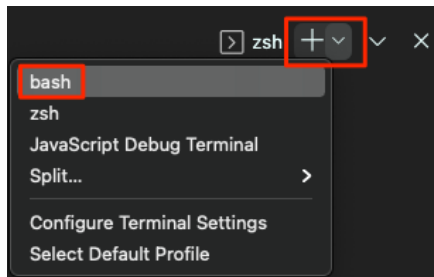
10. Go to your fork of the Space Game project. The forked repository will be saved as “<your-account-name>/mslearn-tailspin-spacegame-web”.
11. Select “Code”. Under the “HTTPS” tab, copy the URL.



12. Open Visual Studio code and open the terminal window. You can do so by going to View > Terminal. Alternatively, the keyboard shortcut is “Ctrl + `”. Ensure that your terminal is set to Git Bash.



13. Ensure that your terminal is set to bash



14. Run the **git clone** command with the URL you copied. Your command will look like:

```
$ git clone https://github.com/<your-name>/mslearn-tailspin-spacegame-web.git
```

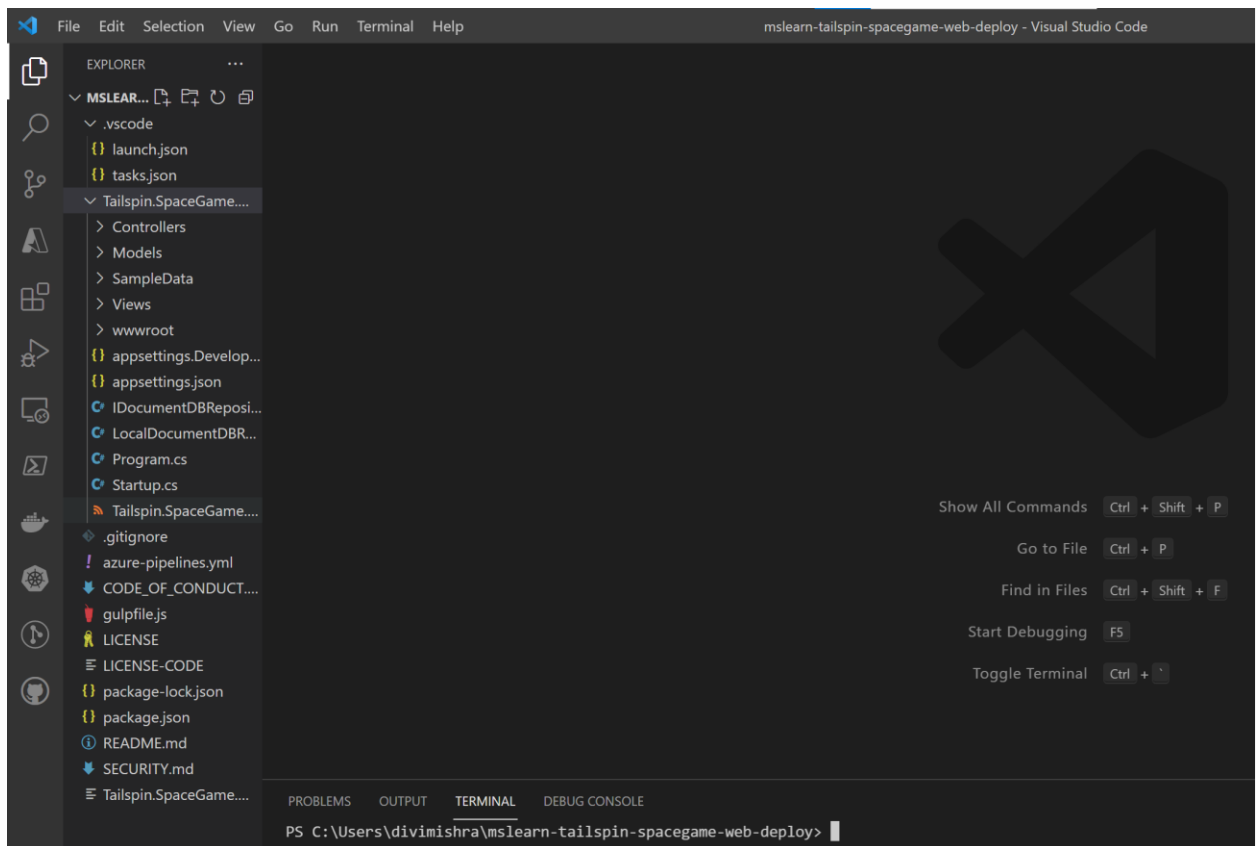
15. Move to the directory with the Space Game project by entering the following command:

```
$ cd mslearn-tailspin-spacegame-web
```

16. Open the project with the following command:

```
$ code -r .
```

17. You are now at the root of your web project. Reopen the terminal by going to “View > Terminal” or “Ctrl + `”



18. 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. Is 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"
```

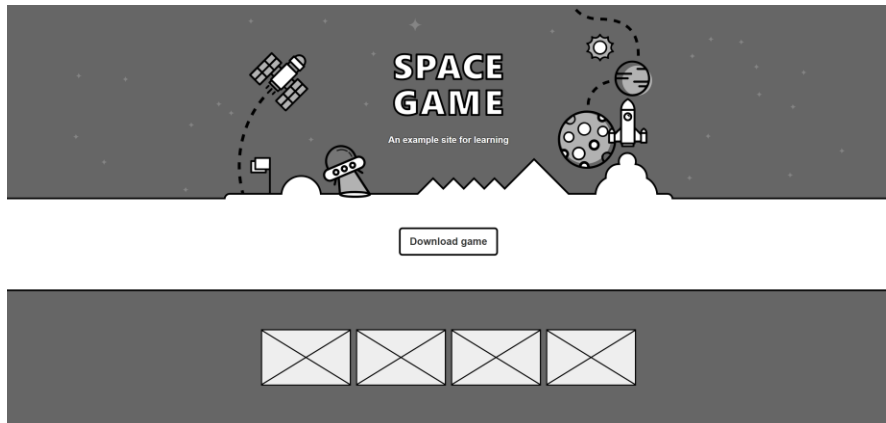
19. Enter the following command on the terminal to make sure the code runs on your local host:

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

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

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.
```



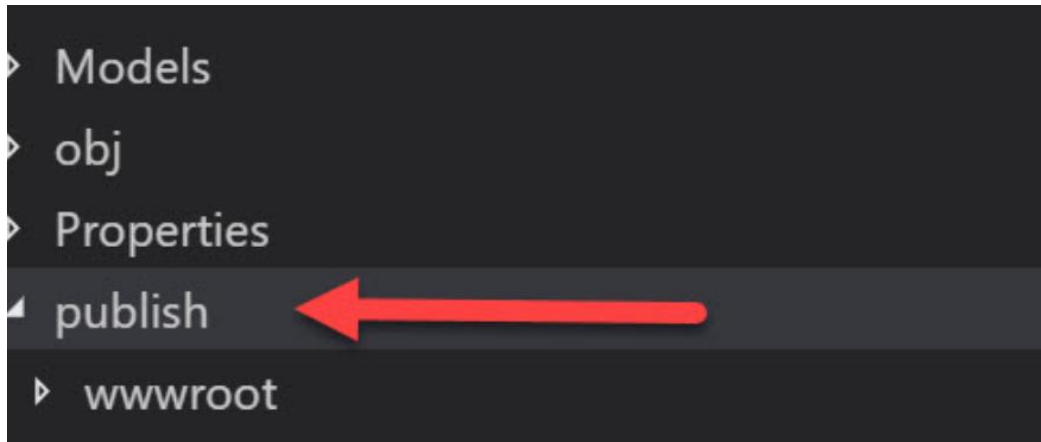
Ctrl+C when you're

done.

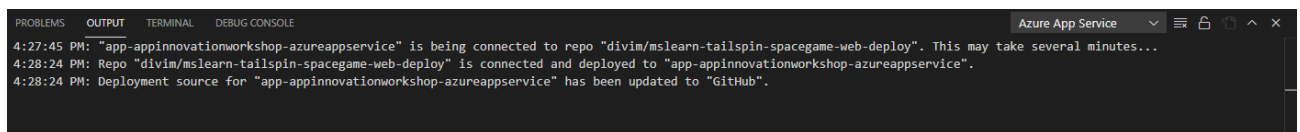
20. Go back to your terminal and enter the following command:

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

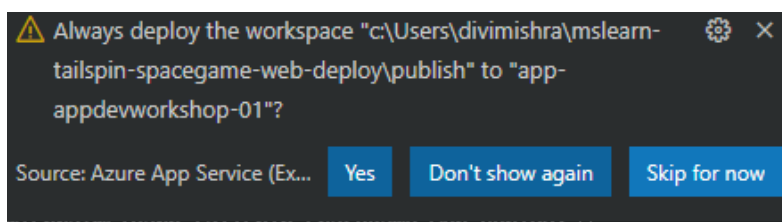
You will notice that a new **publish** folder has been created.



21. Right click on the **publish** folder.
22. Select **Deploy to web app...** and select the right subscription.
23. Select the Azure App service you had created.
24. Select **Deploy** to confirm.
25. To track the progress, open the terminal window. You can do so by going to View > Terminal.
Alternatively, the keyboard shortcut is "Ctrl + `". Navigate to **Output** and ensure that your selection is "**Azure App Service**" from the dropdown.



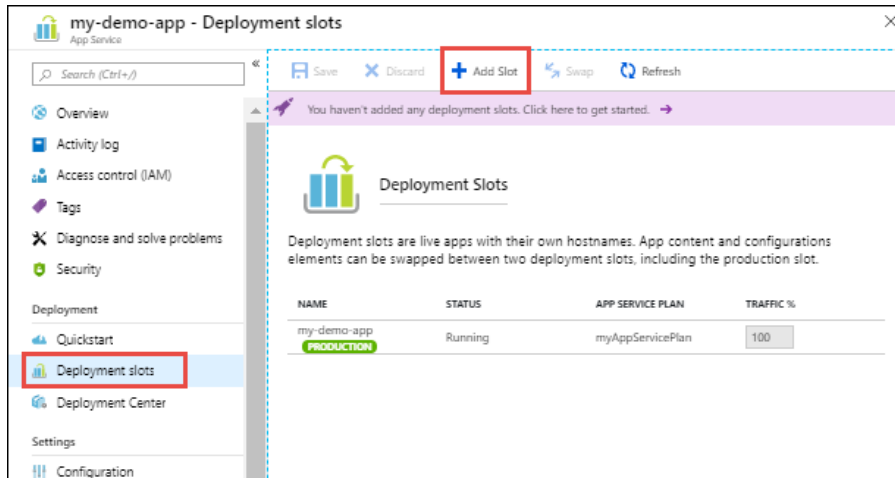
26. If you get the following prompt, click on **Skip for now**. We will be deploying Azure Pipelines for CI/CD in Lab 4.



27. 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.

Name
staging
azqatar-mslearn-tailspin-spacegame-web-staging.azurewebsites.net

Clone settings from:
azqatar-mslearn-tailspin-spacegame-web

Do not clone settings
azqatar-mslearn-tailspin-spacegame-web

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.
5. Go to [Cloud Shell](#).
6. Enter the following command to get the instrumentation key:

```
az resource show \  
  
--resource-group <resource_group_name> \  
  
--name <resource_name> \  
  
--resource-type "Microsoft.Insights/components" \  
  
--query properties.InstrumentationKey
```

Copy the output value.

7. On the VS Code terminal, navigate to ~/mslearn-tailspin-spacegame-web/Tailspin.SpaceGame.Web using the cd command.
8. Run the following command:
dotnet add package Microsoft.ApplicationInsights.AspNetCore --version 2.18.0
9. Navigate to Tailspin.SpaceGame.Web.csproj file. Add/ensure the following code sample:

```
<ItemGroup>  
  
  <PackageReference Include="Microsoft.ApplicationInsights.AspNetCore" Version="2.16.0" />  
  
</ItemGroup>
```

Your code will look something like this:

```
Tailspin.SpaceGame.Web.csproj X
Tailspin.SpaceGame.Web > Tailspin.SpaceGame.Web.csproj
You, 2 days ago | 2 authors (divim and others)
1 <Project Sdk="Microsoft.NET.Sdk.Web">
2
3   <PropertyGroup>
4     <TargetFramework>net5.0</TargetFramework>
5     <ProjectGuid>{A0C4E31E-AC75-4F39-9F59-0AA19D9B8F46}</ProjectGuid>
6   </PropertyGroup>
7
8   <ItemGroup>
9     <Folder Include="wwwroot\images\avatars\" />
10  </ItemGroup>
11
12  <ItemGroup>
13    <Content Remove="SampleData\profiles.json" />
14    <Content Remove="SampleData\scores.json" />
15    <Content Remove="SampleData\profiles.json" />
16    <Content Remove="SampleData\scores.json" />
17  </ItemGroup>
18  <ItemGroup>
19    <Content Update="SampleData\profiles.json">
20      <CopyToOutputDirectory>Always</CopyToOutputDirectory>
21    </Content>
22    <Content Update="SampleData\scores.json">
23      <CopyToOutputDirectory>Always</CopyToOutputDirectory>
24    </Content>
25  </ItemGroup>
26  <ItemGroup>
27    <EmbeddedResource Include="SampleData\profiles.json">
28      <CopyToPublishDirectory>PreserveNewest</CopyToPublishDirectory>
29    </EmbeddedResource>
30    <EmbeddedResource Include="SampleData\scores.json">
31      <CopyToPublishDirectory>PreserveNewest</CopyToPublishDirectory>
32    </EmbeddedResource>
33  </ItemGroup>
34  <ItemGroup>
35    <PackageReference Include="Microsoft.ApplicationInsights.AspNetCore" Version="2.18.0" />
36  </ItemGroup>
37 </Project>
```

10. Navigate to the Startup class under **Startup.cs**. Add the following command under the `ConfigureServices()` method:

```
services.AddApplicationInsightsTelemetry();
```

```
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();
        }
    }
}

```

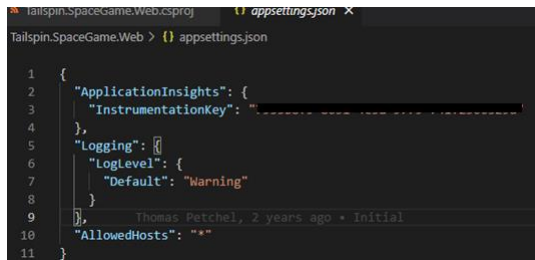
11. Although you can add the instrumentation key as an argument to `AddApplicationInsightsTelemetry`, we recommend that you specify it in configuration. Under **appsettings.json** file, edit the file to the following:

```

{
  "ApplicationInsights": {
    "InstrumentationKey": "putinstrumentationkeyhere"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Warning"
    }
  }
}

```

Screenshot:



```
1 {
2   "ApplicationInsights": {
3     "InstrumentationKey": "..."
4   },
5   "Logging": {
6     "LogLevel": {
7       "Default": "Warning"
8     }
9   },
10  "AllowedHosts": "*"
11 }
```

Save all the changes.

12. As done in the previous lab, Go back to your terminal and enter the following command:

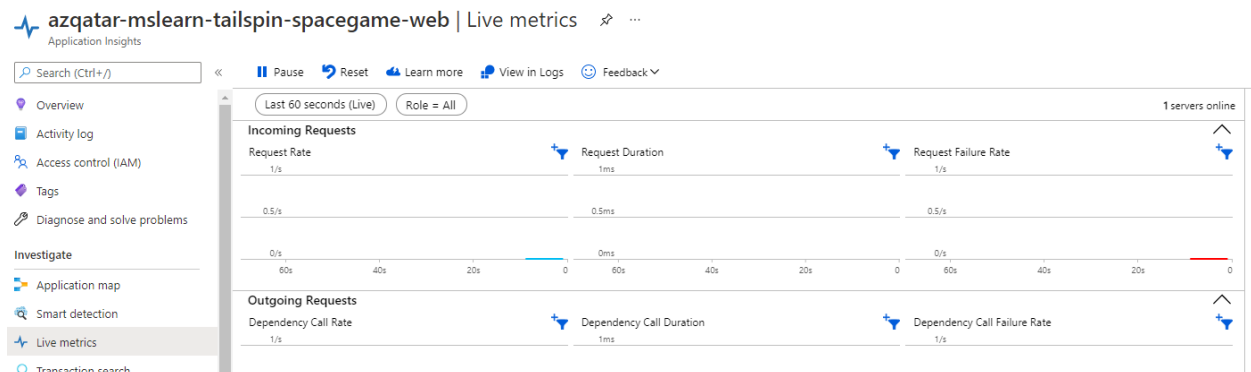
```
$ dotnet publish -c Release -o ./publish
```

13. Right-click on your app service 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).

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

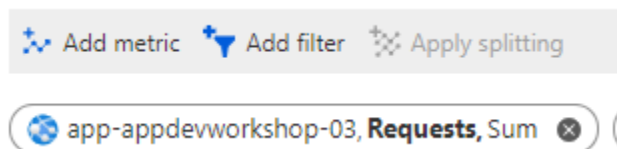
Please note that it may take a few minutes for the telemetry to appear.



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

Monitor apps

1. Under **Monitoring**, click on **Metrics**
2. Select **Requests** as your metric, and **Sum** as your aggregation.
3. Select **Add metric** from the top left.



4. Select metrics of your choice and display them into meaningful bar representations.

5. Explore the **Pin to Dashboard** option at the top right for developing shared dashboards to share with team members.



Select Pin to Dashboard to be able to easily monitor your desired queries and metrics.

Configure alerts for Azure App Service

1. Navigate 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 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 the Action Group: You can leave it as Application Insights Smart Detection or “Create Action Group” for your team.
7. Select the alert rule name, description, severity. 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

Setting up a dashboard to review changes

1. Navigate to your app service through Azure Portal.
2. Under Monitoring, select Metrics.
3. From the top left, change “Local time: Last 24 hours (Automatic)” to “Last 30 minutes”. Then, select Apply.
4. In the pane, under Chart Title, add the following metrics to the chart:
 - 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.
5. Select “Pin to Dashboard”. Click on the notification for viewing your dashboard. d

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 percentate is less than 10%.

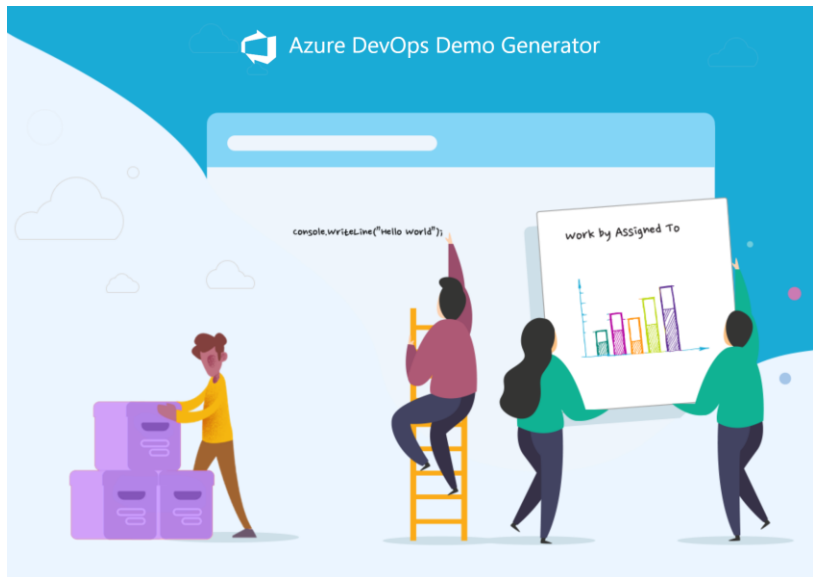
Operator *	Metric threshold to trigger scale action * ⓘ
Less than or equal to ▼	10 ✓ %
Duration (minutes) * ⓘ	
5 ✓	
Time grain (minutes) ⓘ	Time grain statistic * ⓘ
1	Average ▼
🔧 Action	
Operation *	Cool down (minutes) * ⓘ
Decrease count to ▼	5
Instance count *	
1 ✓	

- 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](#)



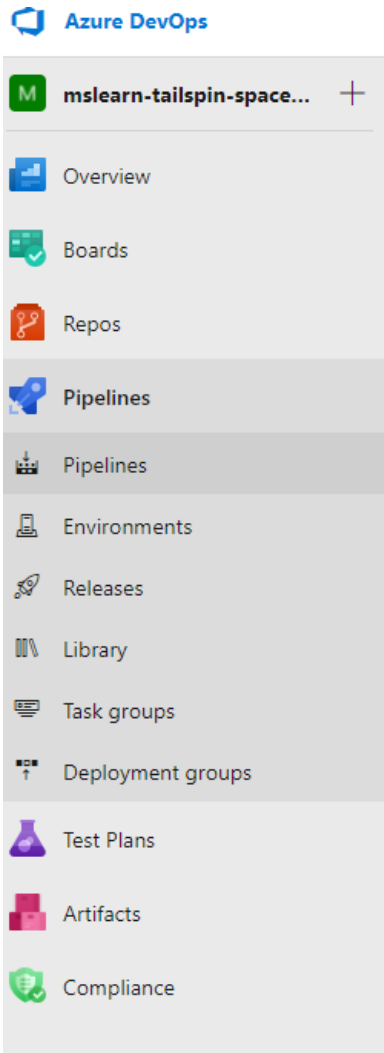
2. Sign in and accept the usage terms.
3. Create a new project with your Azure DevOps (ADO) organization and a project name. For Selected template: under MS learn, chose “Create a build pipeline with Azure Pipelines”. Finally, create project.

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"/>	

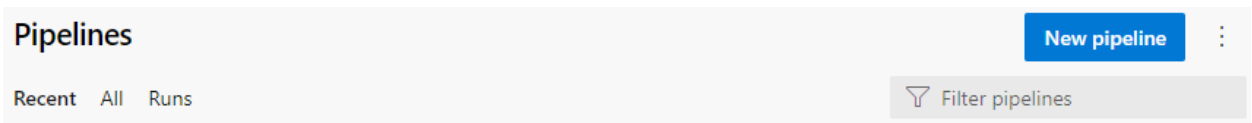
4. Once the deployment is done, select “Navigate to the project”

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


divimishra / mslearn-tailspin-spacegame... / Pipelines


Select a source


 Azure Repos Git

 GitHub

 GitHub Enterprise Server

 Subversion

 Bitbucket Cloud

 Other Git

Authorized using connection: divim Change

Repository * | Manage on GitHub [it](#)
divim/mslearn-tailspin-spacegame-web

Default branch for manual and scheduled builds *
main

Continue

Select "Continue".

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

Select a template

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.

6. 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

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

Agent job 1
Run on agent

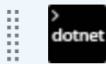
Add a task to Agent job 1

8. 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

9. 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 in blue and has 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)

10. 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 ⓘ

11. 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)'. Other sections like 'Advanced', 'Control Options', and 'Output Variables' are collapsed.

12. 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 task configuration on the right shows 'Display name' as 'Test' and 'Command' as 'test'. The 'Arguments' field is empty. The 'Publish test results and code coverage' checkbox is checked. In the top right corner, the 'Remove' button, represented by a trash icon, is highlighted with a red rectangle. The 'Test' task is also highlighted in the task list on the left.

13. 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 displays the Azure DevOps pipeline editor 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', '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', 'Control Options', and 'Output Variables' sections are collapsed.

14. 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 Azure Pipelines. The left sidebar lists tasks: 'Get sources', 'Agent job 1', and a list of tasks including 'Use .NET Core sdk 5.x', 'npm install', 'gulp', 'Restore', 'Build', 'Publish', and 'Publish Artifact' (highlighted with a blue box and a checkmark). The main panel shows the configuration for 'Publish build artifacts'. The 'Task version' is set to '1.*'. The 'Display name' is 'Publish Artifact'. The 'Path to publish' is '\$(build.artifactstagingdirectory)'. The 'Artifact name' is 'drop'. The 'Artifact publish location' is 'Azure Pipelines'. The 'Advanced' section is expanded, showing 'Control Options' and 'Output Variables'.

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

The screenshot shows the 'Triggers' section configuration in Azure Pipelines. The left sidebar lists triggers: 'Continuous integration', 'Pull request validation', 'Scheduled', and 'Build completion'. The 'Continuous integration' trigger is selected and highlighted. The main panel shows the configuration for 'divim/mslearn-tailspin-spacegame-web'. The 'Enable continuous integration' checkbox is checked. The 'Batch changes while a build is in progress' checkbox is unchecked. The 'Branch filters' section shows 'Type' set to 'Include' and 'Branch specification' set to 'main'. The 'Path filters' section has an 'Add' button.

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

17. 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

18. 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.

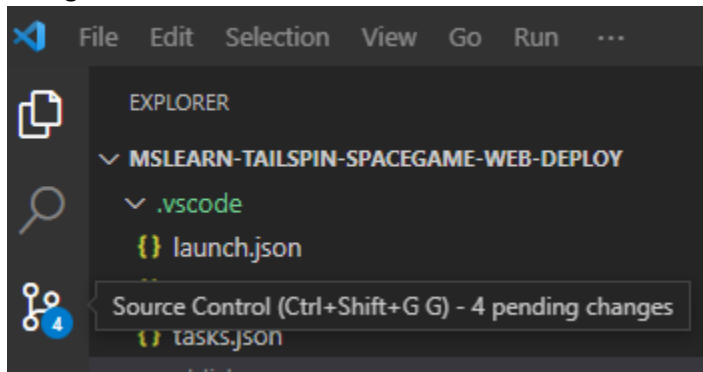
Source Control with GitHub

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.

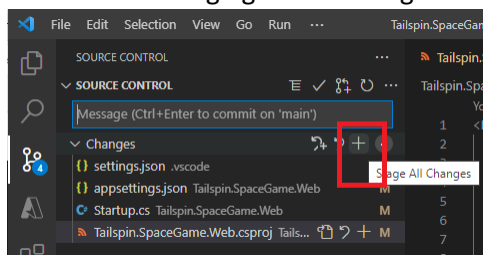
Recall that we had made some changes to the VS Code when we added the Application Insights instrumentation key.

Let's now push it to GitHub.

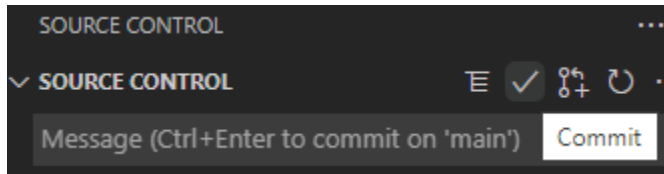
1. Navigate back to your VS Code and notice that your **Source Control** tab highlights pending changes.



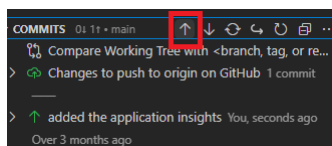
2. Click on the **Source Control** tab
3. Click on + for Staging all the changes



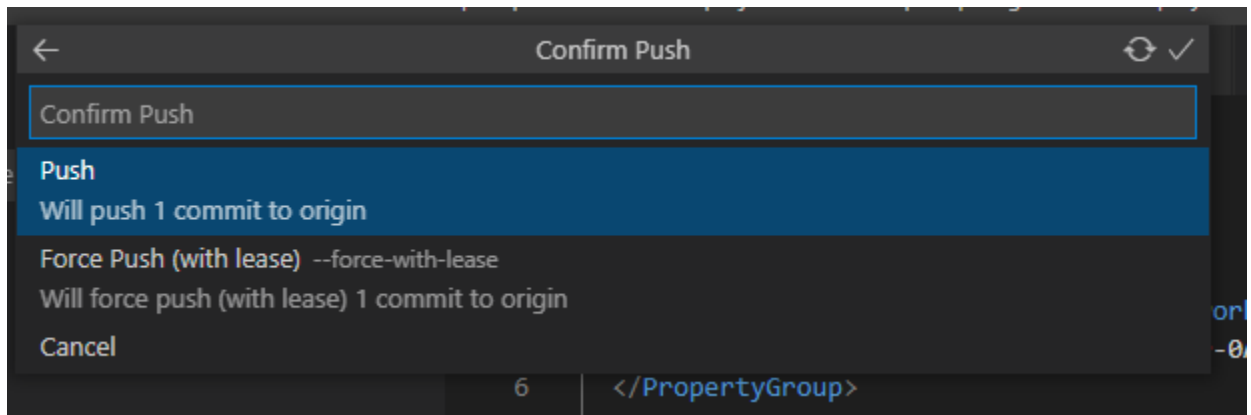
4. Click on the tic mark for Committing the changes



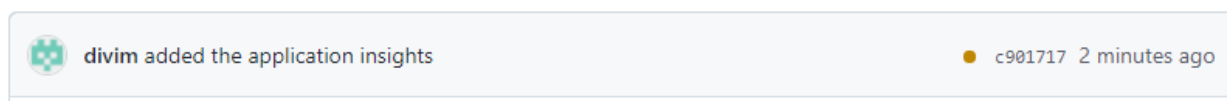
5. Enter a relevant message, such as "added app insights instrumentation key"
6. Under **COMMITTS** (at the bottom), click on the UP arrow. This will **Push** all the changes to GitHub.



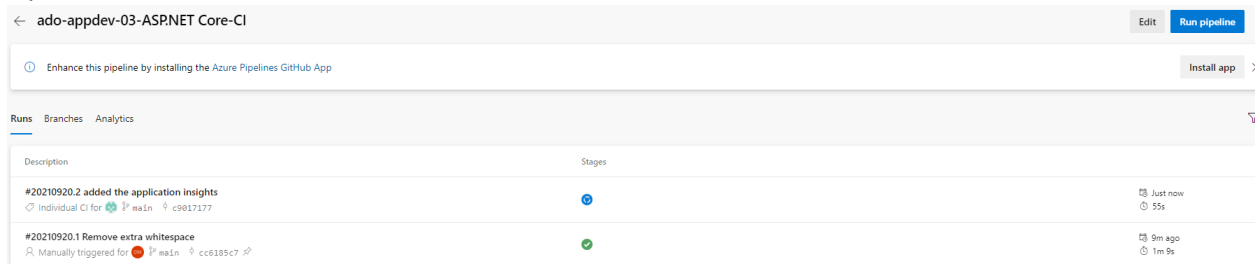
7. Confirm the Push.



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.

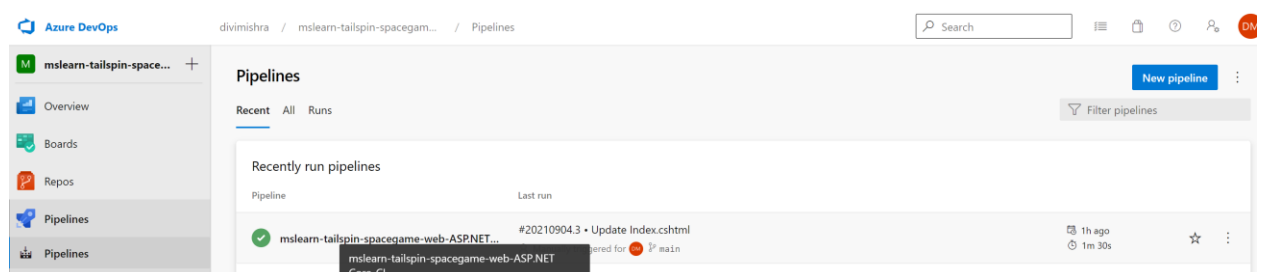


Note: You have now configured a way to reflect changes from your VS Code -> GitHub -> Azure Pipelines. In **Lab 05**, we will complete the pipeline as VS Code -> GitHub -> Azure Pipelines -> Azure App Services.

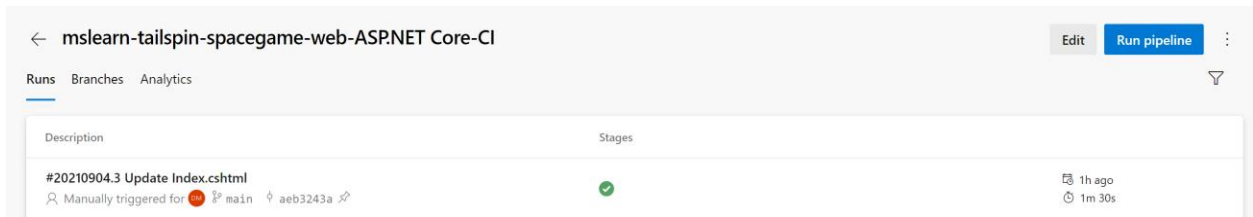
Add unit tests

Adding the test job to the pipeline

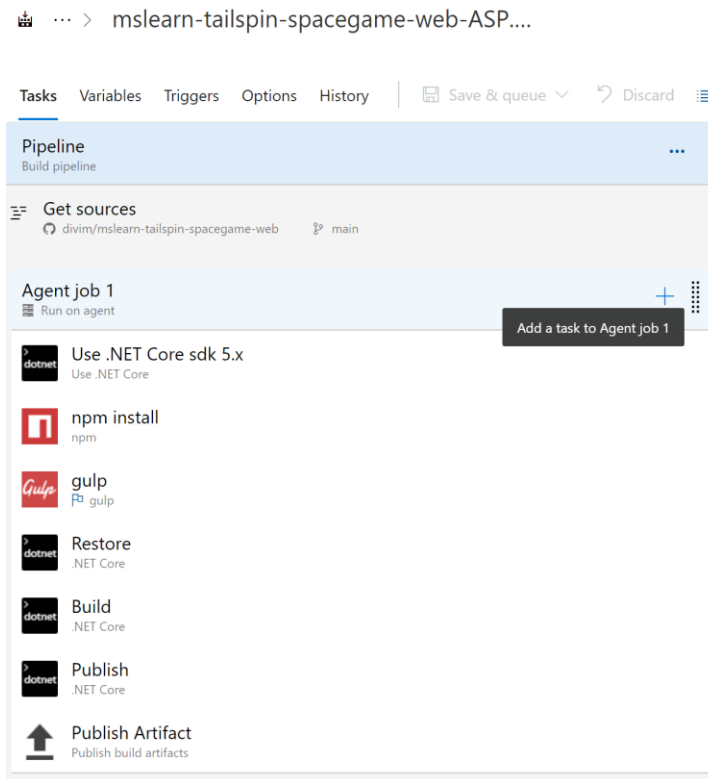
1. Open your Pipeline on dev.azure.com again.



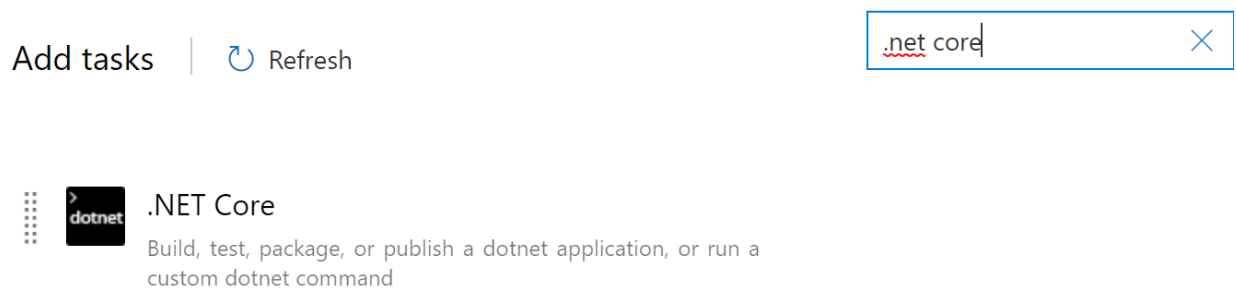
2. Click on **Edit** from the top right to edit the pipeline.



3. Click on “+” to add a task to the job agent.



4. Search for “.NET Core”. Click on “Add”.



5. Drag the task between “**Build**” and “**Publish**”. Your pipeline now looks like this:

Tasks Variables Triggers Options History | Save & queue Discard

Pipeline









Build pipeline

Get sources

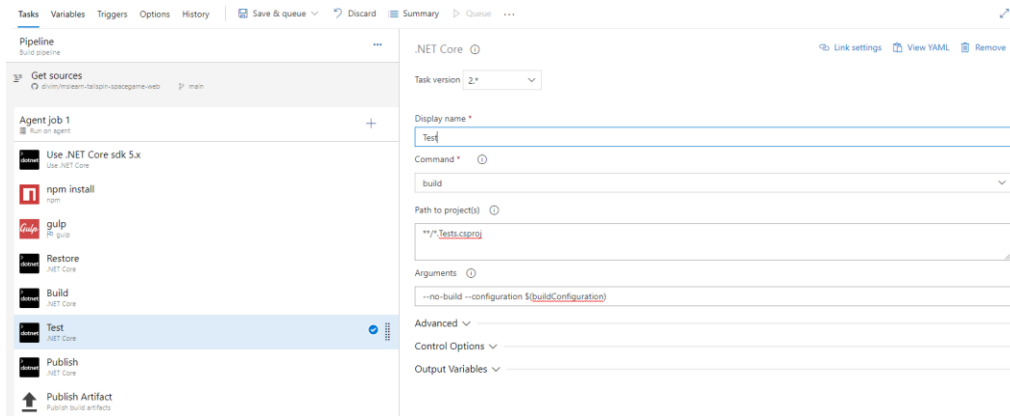
divim/mslearn-tailspin-spacegame-web main

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
-  dotnet build
.NET Core
-  Publish
.NET Core
-  Publish Artifact
Publish build artifacts

6. Click on the task to edit it.
 - a. Under Display name, type **Test**
 - b. Under Command, select **test**
 - c. Under path to project, type ****/*.Tests.csproj**
 - d. Under arguments, type **--no-build --configuration \$(buildConfiguration)**

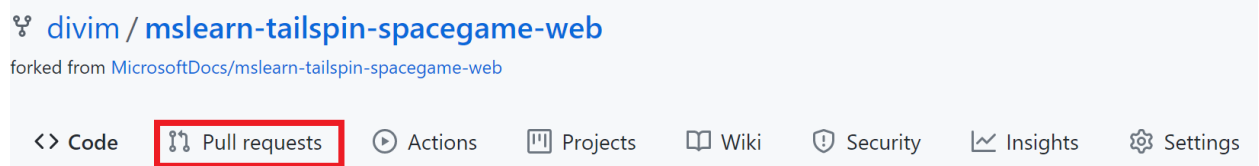


7. Click on **Save** (NOT Save & Queue).

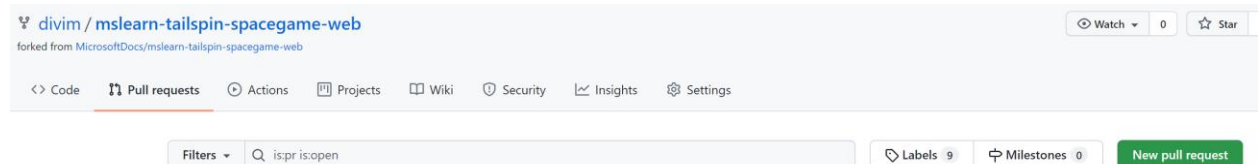
Pushing the tests files to the GitHub repository

The tests are stored in the **unit-tests** branch. Hence, we need to push these tests to the **main** branch.

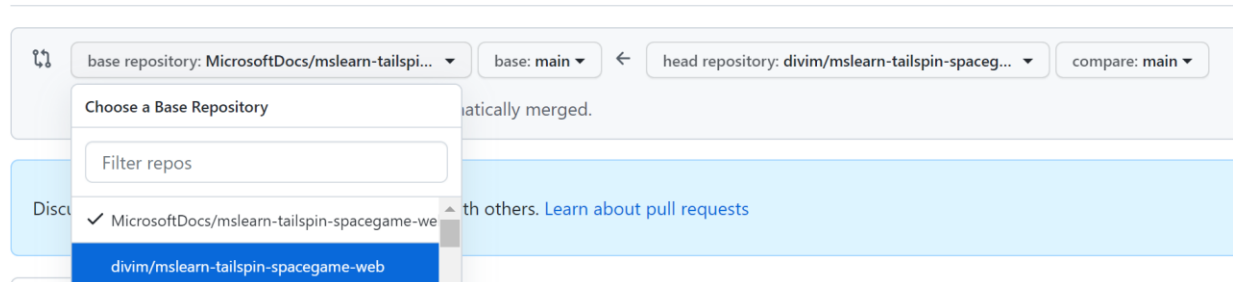
1. Navigate to your GitHub repository. Click on **Pull requests**.



2. Click on **New Pull Request**




3. Change the default base repository. Select your forked repo. (Example: <yourname>/mslearn-tailspin-spacegame-web



4. Leave the base branch as **main** and change the compare branch to **unit-tests**

Comparing changes


Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

 base: main ← compare: unit-tests ✗ Can't automatically merge. Don't worry, you can still create the pull request.







Discuss and review the changes in this comparison with others. [Learn about pull requests](#) Create pull request

(If you get an error “Can’t automatically merge.”: ignore and proceed creating pull request.)


5. Write a relevant message and click on **Create pull request**.

 Unit tests

Write Preview

H B I  <>     @  ↶

Request to pull from unit-tests to main.

Attach files by dragging & dropping, selecting or pasting them. 

Create pull request

6. (*OPTIONAL STEP*) If you have conflicts on your pull request, scroll down and click on **Resolve conflicts**.



 All checks have passed
1 successful check [Show all checks](#)

 This branch has conflicts that must be resolved
Use the [web editor](#) or the [command line](#) to resolve conflicts.

Resolve conflicts

Conflicting files

Tailspin.SpaceGame.Web/Tailspin.SpaceGame.Web.csproj
Tailspin.SpaceGame.Web/Views/Home/Index.cshtml

Merge pull request

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

When you open the file with conflicts in your text editor, you'll see the changes from the HEAD or base branch after the line <<<<<< HEAD. Next, you'll see =====, which divides your changes from the changes in the other branch, followed by >>>>>> BRANCH-NAME.

```

11 <<<<< unit-tests
12 <ItemGroup>
13 <Content Remove="SampleData\profiles.json" />
14 <Content Remove="SampleData\scores.json" />
15 <Content Remove="SampleData\profiles.json" />
16 <Content Remove="SampleData\scores.json" />
17 </ItemGroup>
18 <ItemGroup>
19 <Content Update="SampleData\profiles.json">
20 <CopyToOutputDirectory>Always</CopyToOutputDirectory>
21 </Content>
22 <Content Update="SampleData\scores.json">
23 <CopyToOutputDirectory>Always</CopyToOutputDirectory>
24 </Content>
25 </ItemGroup>
26 <ItemGroup>
27 <EmbeddedResource Include="SampleData\profiles.json">
28 <CopyToPublishDirectory>PreserveNewest</CopyToPublishDirectory>
29 </EmbeddedResource>
30 <EmbeddedResource Include="SampleData\scores.json">
31 <CopyToPublishDirectory>PreserveNewest</CopyToPublishDirectory>
32 </EmbeddedResource>
33 =====
34
35 <ItemGroup>
36 <PackageReference Include="Microsoft.ApplicationInsights.AspNetCore" Version="2.18.0" />
37 >>>>>> main

```

Edit the file and choose the part of the file you'd like to keep. Then, click on **Mark as resolved**.

Tailspin.SpaceGame.Web/Tailspin.SpaceGame.Web.csproj
1 conflict
Prev
Next
Mark as resolved

```

1 <Project Sdk="Microsoft.NET.Sdk.Web">
2
3 <PropertyGroup>
4 <TargetFramework>net5.0</TargetFramework>
5 <ProjectGuid>{ABC4E31E-AC75-4F39-9F59-0AA19D988F46}</ProjectGuid>
6 </PropertyGroup>
7
8 <ItemGroup>
9 <Folder Include="wwwroot\images\avatars\" />
10 </ItemGroup>
11
12 <ItemGroup>
13 <Content Remove="SampleData\profiles.json" />
14 <Content Remove="SampleData\scores.json" />
15 <Content Remove="SampleData\profiles.json" />
16 <Content Remove="SampleData\scores.json" />
17 </ItemGroup>
18 <ItemGroup>
19 <Content Update="SampleData\profiles.json">
20 <CopyToOutputDirectory>Always</CopyToOutputDirectory>
21 </Content>
22 <Content Update="SampleData\scores.json">
23 <CopyToOutputDirectory>Always</CopyToOutputDirectory>
24 </Content>
25 </ItemGroup>
26 <ItemGroup>
27 <EmbeddedResource Include="SampleData\profiles.json">
28 <CopyToPublishDirectory>PreserveNewest</CopyToPublishDirectory>
29 </EmbeddedResource>
30 <EmbeddedResource Include="SampleData\scores.json">
31 <CopyToPublishDirectory>PreserveNewest</CopyToPublishDirectory>
32 </EmbeddedResource>
33 </ItemGroup>
34 <PackageReference Include="Microsoft.ApplicationInsights.AspNetCore" Version="2.18.0" />
35 </ItemGroup>
36 </Project>

```

Then, click on **Commit merge**.

Unit tests #7

Resolving conflicts between `unit-tests` and `main` and committing changes → `unit-tests`

Commit merge

2 conflicting files		Tailspin.SpaceGame.Web/Views/Home/Index.cshtml	✓ Resolved
Tailspin.SpaceGame.Web.csproj	✓	<pre> 1 @model Tailspin.SpaceGame.Web.Models.LeaderboardViewModel 2 @{ 3 ViewData["Title"] = "Home Page"; 4 } 5 <section class="Intro"> 6 <div class="container"> 7 8 <p>Welcome to the official Space Game site!</p> </pre>	
Index.cshtml	✓		

Your test files are now on the main branch.

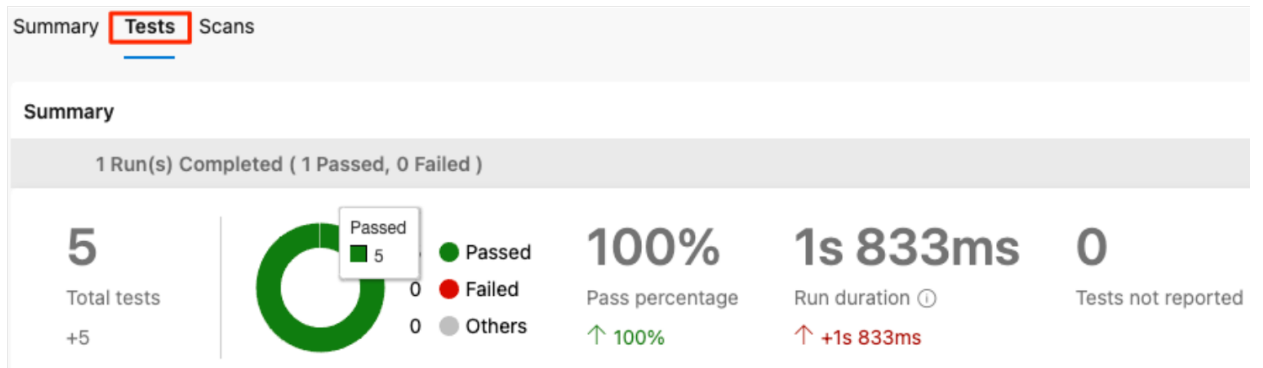
8. Since you committed the tests to the main branch, the pipeline has been triggered again. This time, it includes the **Test** job.
9. Navigate to the latest pipeline run. Click on the **Agent Job** and observe the tasks run.

The screenshot shows the 'Jobs' section of a pipeline run. The header indicates the pipeline is 'mslearn-tailspin-spacegame-web-ASP.NET Core-CI'. Under the 'Jobs' heading, 'Agent job 1' is expanded, showing a list of tasks. Each task is preceded by a green checkmark icon, indicating successful completion. The tasks and their durations are as follows:

Task	Duration
Initialize job	5s
Checkout divim/...	2s
Nuget Security A...	1s
Use .NET Core sd...	7s
npm install	10s
gulp	3s
Restore	12s
Build	8s
Test	6s
Publish	4s
Publish Artifact	2s
Component Det...	36s
Post-job: Check...	<1s
Finalize Job	<1s

Your pipeline now contains **unit testing**.

10. Navigate back to the **pipeline summary**.
11. Move to the **Tests** tab to view a summary of the test run.

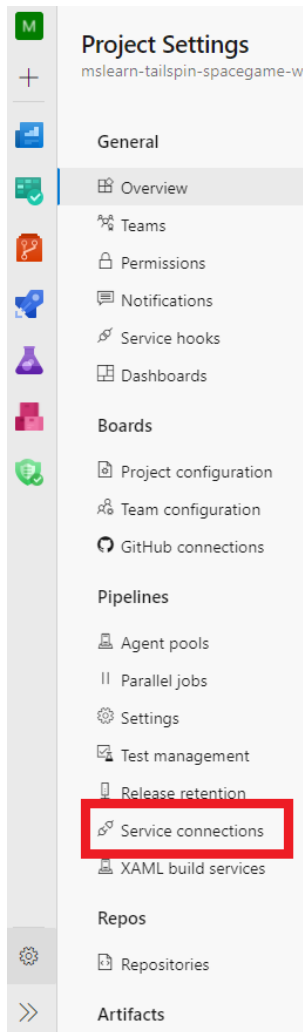


Note: You must ideally add functional and non-functional test to your build pipeline. Read more about such tests: [Run Functional Tests task - Azure Pipelines | Microsoft Docs](#)

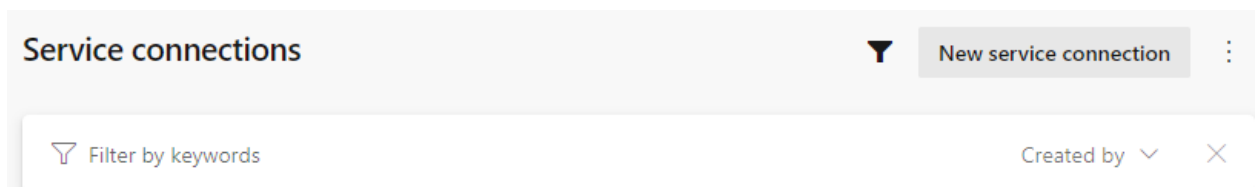
Lab 05: Zero-downtime app deployment with release pipeline

Deploy to staging slot from build pipeline

1. Go back to your build pipeline in dev.azure.com.
5. At the bottom left, click on **Project Settings**. Under Pipelines, navigate to Service connections.



6. Click on **New Service Connection**



7. Click on **Azure Resource Manager**. Select Next.
8. Click on **Service Principal (Automatic)**. Select Next.
9. Define your Subscription and Resource group that contains your app service.
10. Give a relevant service connection name.

11. 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

12. Select **Save**.

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

← mslearn-tailspin-spacegame-web-ASP.NET Core-CI

Edit

Run pipeline

⋮

Runs

Branches

Analytics

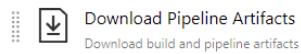
⌵

14. Add the following agent jobs:

a. Download pipeline artifact

Add tasks | Refresh

download pipeline



b. Azure App Service Deploy

Add tasks | Refresh

azure app service deploy X



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. Adjust the tasks for it to be in the following order:

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



Test

.NET Core



Publish

.NET Core



Publish Artifact: drop

Publish build artifacts



Download Pipeline Artifact

Download Pipeline Artifacts

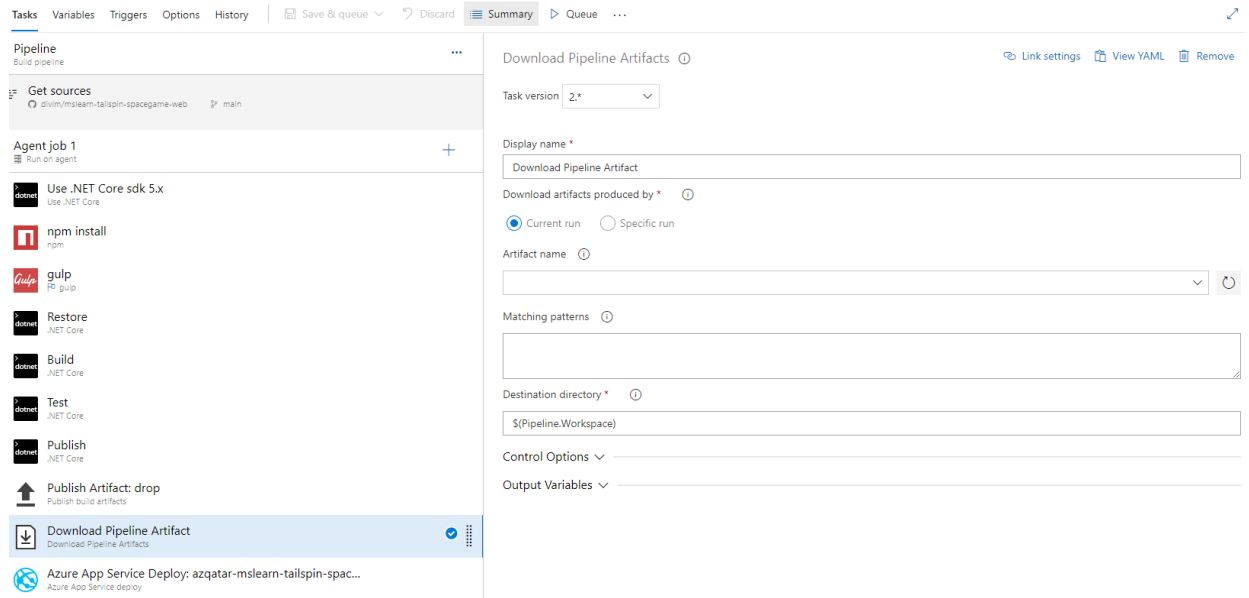


Azure App Service Deploy: i

Azure App Service deploy

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



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

- Connection type: Azure Resource Manager
- Susbcription: <name-of-your-service-connection>
- App Service type: *as defined during your app service definition above*
- App Service name: *as defined during your app service definition above*
- Select **Deploy to Slot or App Service Environment**
- Resource group: RG of your app service
- Slot: staging
- Package or folder: **\$(Pipeline.Workspace)/drop/Release/Tailspin.SpaceGame.Web.zip**

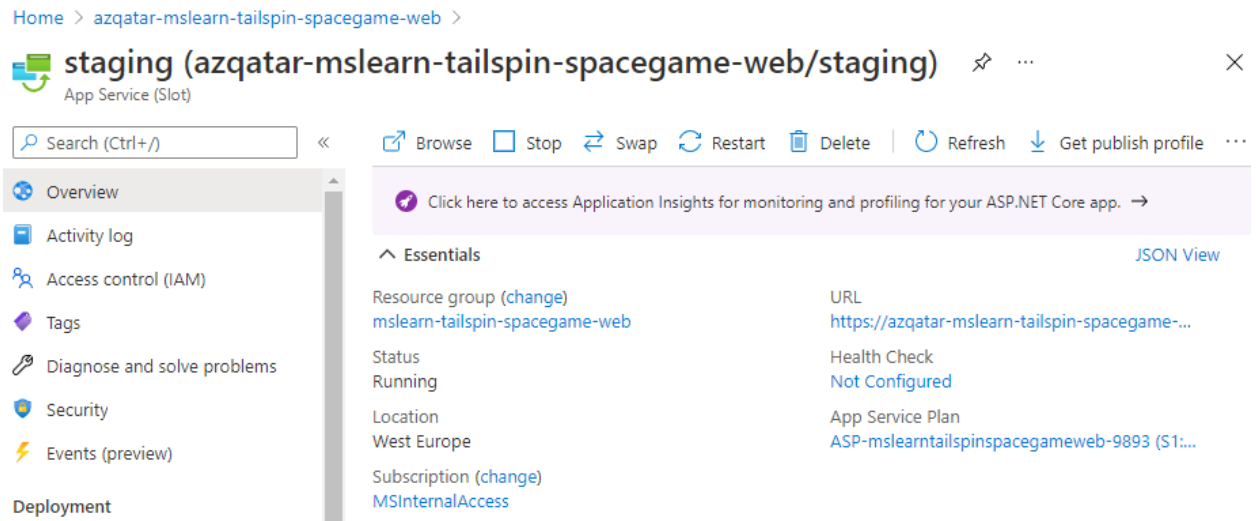
18. Select **Save & Queue**.

19. Select **Save & Run**.

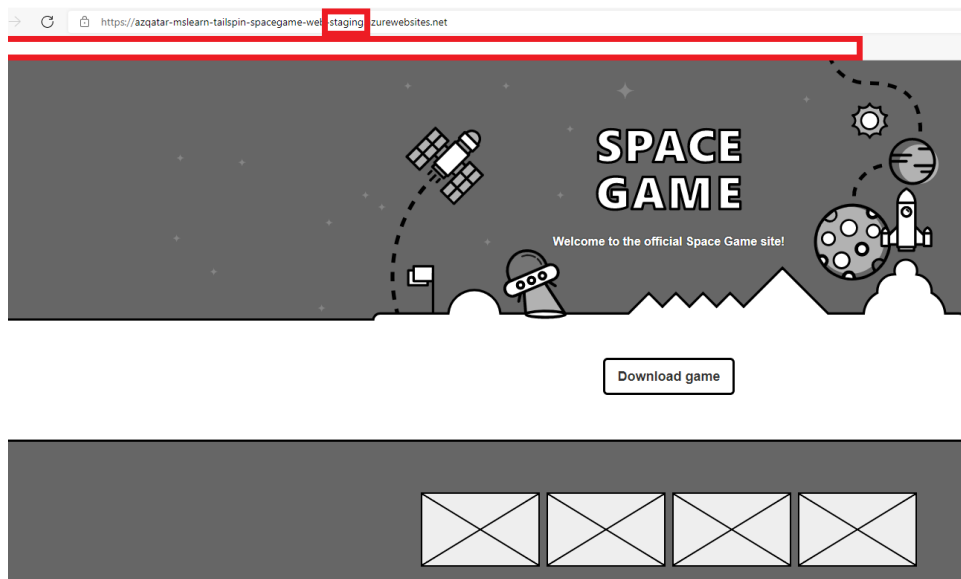
20. Once the pipeline is done running, go back to Azure Portal > App Service > your app service > Deployment Slots

21. Click on the **staging slot**

22. Click on **Browse**



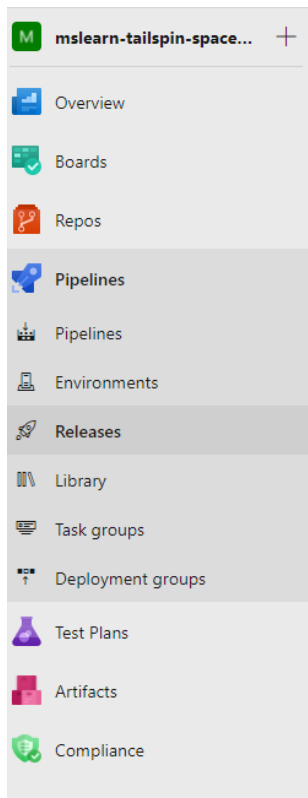
23. Your app has been deployed successfully to the staging slot. Note that the name of the URL is simply <app-service-name>-staging.azurewebsites.net



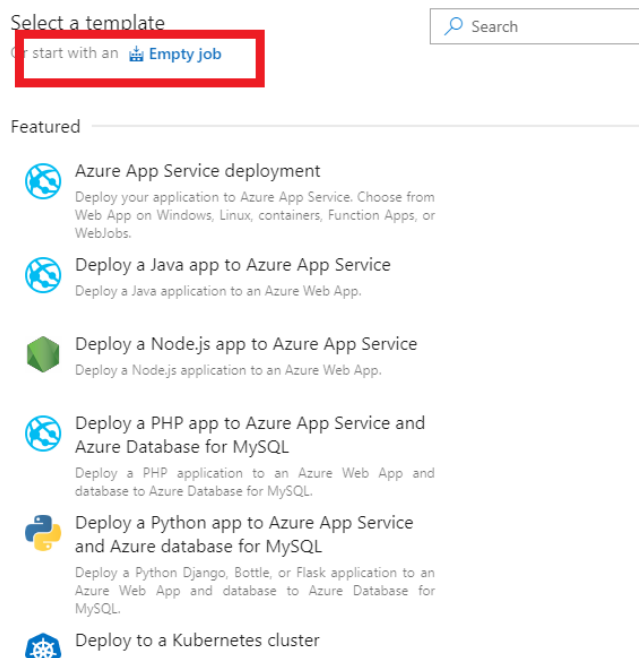
24. Simply remove the “-staging” from the URL to view the production slot website.

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

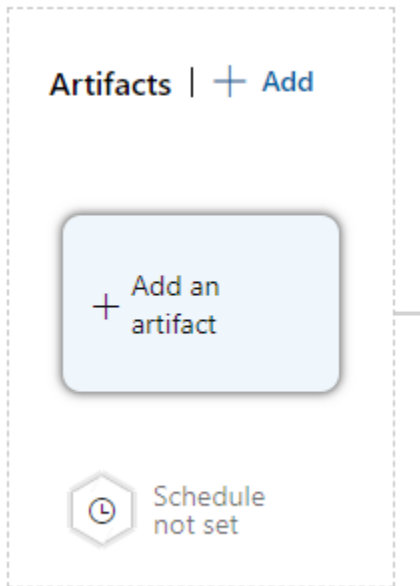
1. Navigate to your Azure DevOps project > **Pipelines** > **Releases**



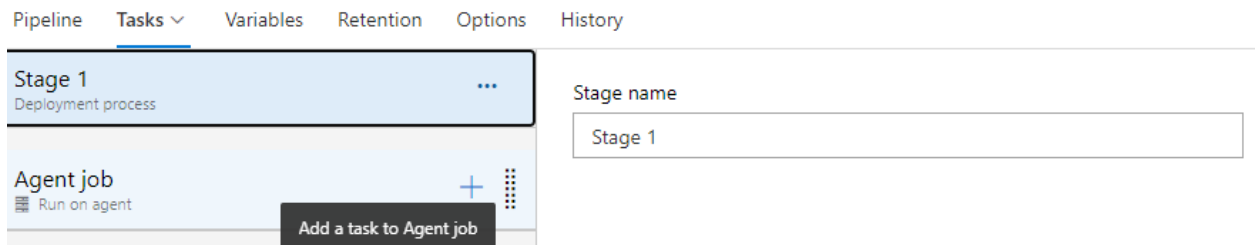
2. Select “New pipeline”.
3. For the template, **start with an empty job**



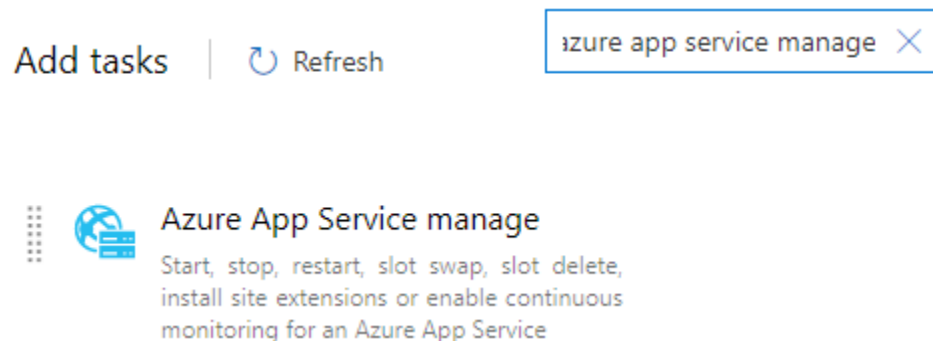
- Click on **Add an artifact**



- 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 'Tasks' tab in the Azure DevOps pipeline editor. On the left, a list of tasks for 'Stage 1' is shown, including 'Agent job', 'Swap Slots: azqatar-mslearn-...', and 'Swap Slots:'. The 'Swap Slots' task is selected, and its configuration is shown on the right. The task is titled 'Azure App Service manage'. 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 warning icon, with the message 'This setting is required.' below it. The 'Action' is 'Swap Slots'. The 'App Service name' is set to a dropdown menu with a red border and a warning icon, with the message 'This setting is required.' below it. The 'Resource group' is set to a dropdown menu with a red border and a warning icon, with the message 'This setting is required.' below it. The 'Source Slot' is set to a dropdown menu with a red border and a warning icon, with the message 'This setting is required.' below it. At the bottom, there is a checkbox for 'Swap with Production' which is currently unchecked.

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

Started: 9/5/2021, 12:00:10 AM
Pool: Azure Pipelines · Agent: Hosted Agent
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.

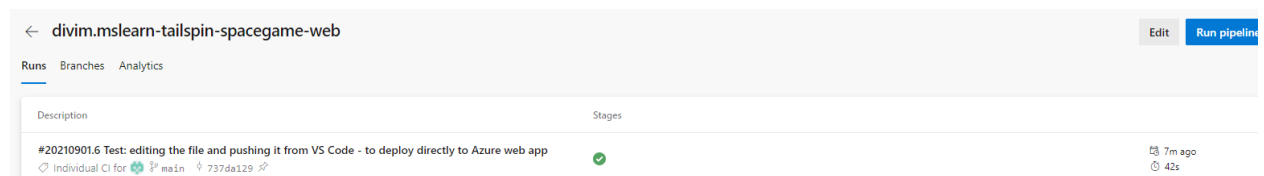
Optional Lab 01: Enabling continuous integration with YAML files

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:

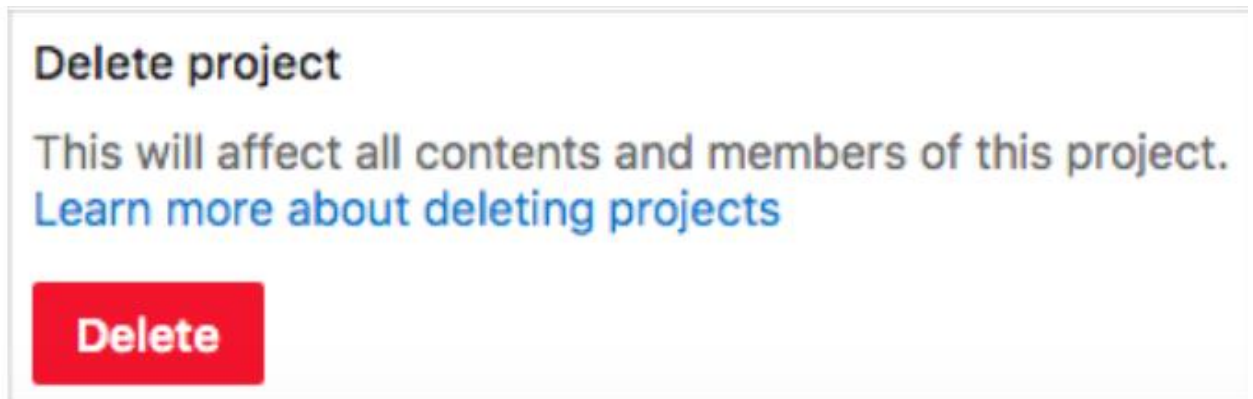
- Detect bugs
- Code smells
- Security vulnerabilities
- Centralize qualities

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/sonarqube-managing-technical-debt/)

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](#)