

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 101: AZURE APP SERVICE..... 5

Create an Azure App Service.....	5
Fork the web project to your GitHub account	6
Create deployment slots	12

LAB 102: MONITOR AND SCALE YOUR APPLICATION 13

Enabling live telemetry through instrumentation key using VS Code	13
Monitor apps.....	16
Configure alerts with Azure Logic Apps for Azure App Service	17
Scale up your app service	23
Scale out your app service	23

LAB 103: CONTINUOUS INTEGRATION WITH AZURE DEVOPS .. 24

Configure Azure DevOps project	24
Create a build pipeline with Azure Pipelines.....	25
Source Control with GitHub.....	34

LAB 104: TESTING WITH AZURE PIPELINES..... 36

Run unit tests locally.....	36
Run Code Coverage test locally.....	37
Add tests to Azure Pipeline.....	38

LAB 105: ZERO-DOWNTIME APP DEPLOYMENT WITH RELEASE PIPELINE 44

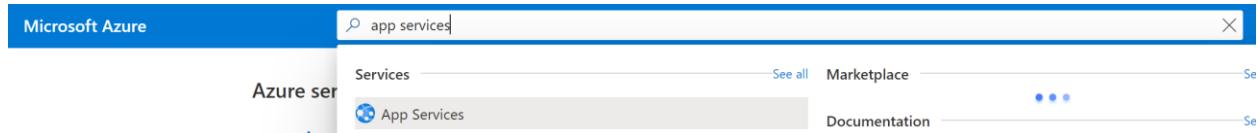
Deploy to staging slot from build pipeline	44
Build a release pipeline: zero-downtime deployment with slot swapping.....	51

LAB 201: UI TESTING WITH SELENIUM	61
Generate PAT for ADO	61
Configure the VM agent	62
Configure the pipeline.....	66
LAB 202: PROTECT YOUR APPLICATION	56
Deploy WAF with your application.....	56
Azure Defender for App Services	60
CLEAN-UP YOUR ENVIRONMENT	68
RESOURCES	69

Lab 101: Azure App Service

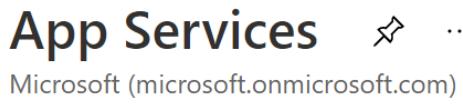
Create an Azure App Service

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



The screenshot shows the Microsoft Azure portal's search interface. A search bar at the top contains the text "app services". Below the search bar, there is a navigation bar with links for "Services", "See all", "Marketplace", "Documentation", and three dots. Under the "Services" link, there is a card for "App Services" with a blue icon.

3. Click on + Create



The screenshot shows the "App Services" creation page. At the top, it says "App Services" and "Microsoft (microsoft.onmicrosoft.com)". Below that is a "Create" button with a plus sign and a "Manage view" dropdown menu.

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 * 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 different region.](#)

App Service Plan

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. [Learn more](#) [?](#)

Windows Plan (West Europe) * Create new

Sku and size * 100 total ACU, 1.75 GB memory

[Review + create](#)

< Previous

Next : Deployment >

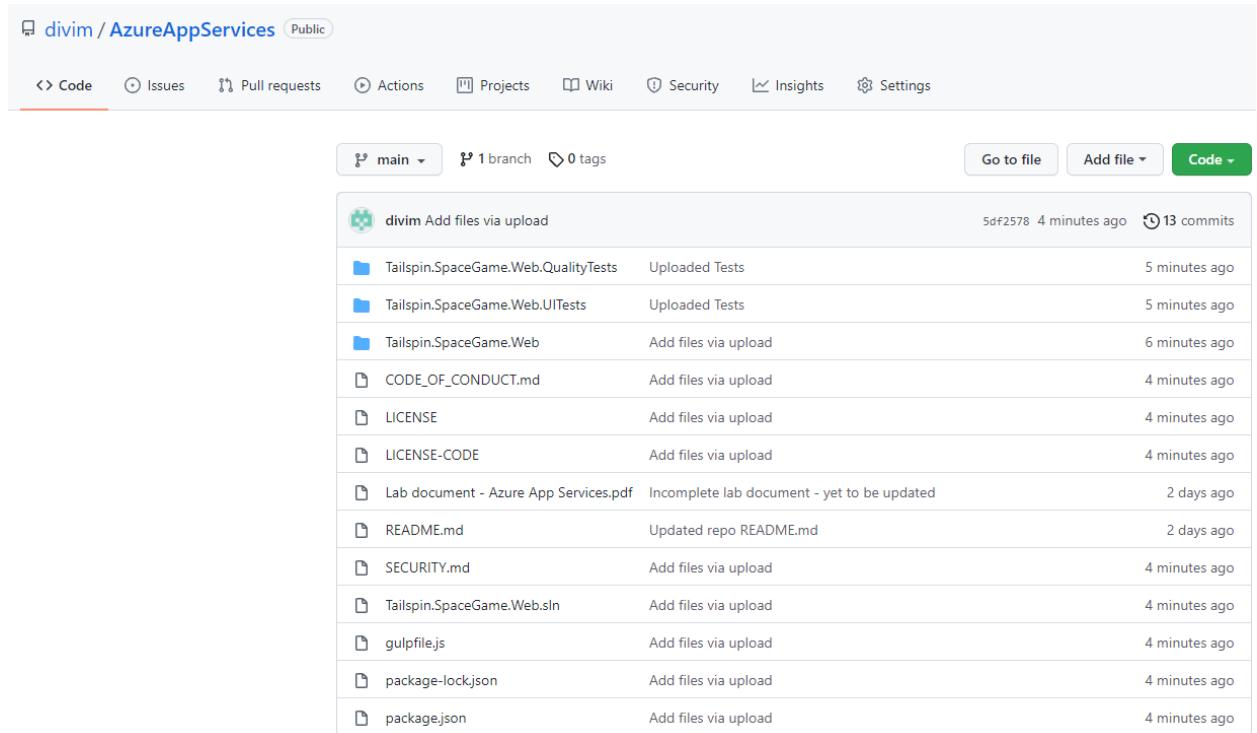
8. Once the deployment is complete, click on “Go to resource”.
9. From the overview tab, click on the “URL” to view the default web application that gets deployed.

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

10. You should be able to view a welcome page.

Fork the web project to your GitHub account

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



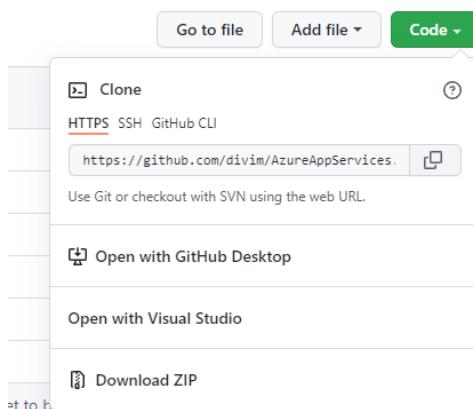
The screenshot shows a GitHub repository page for 'divim / AzureAppServices'. The 'Code' tab is active. At the top, there are navigation links: Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation, it shows 'main' branch, 1 branch, 0 tags, and three buttons: Go to file, Add file, and Code. The main content area displays a list of files and their upload history:

File	Description	Last Commit
Tailspin.SpaceGame.Web.QualityTests	Uploaded Tests	5 minutes ago
Tailspin.SpaceGame.Web.UTests	Uploaded Tests	5 minutes ago
Tailspin.SpaceGame.Web	Add files via upload	6 minutes ago
CODE_OF_CONDUCT.md	Add files via upload	4 minutes ago
LICENSE	Add files via upload	4 minutes ago
LICENSE-CODE	Add files via upload	4 minutes ago
Lab document - Azure App Services.pdf	Incomplete lab document - yet to be updated	2 days ago
README.md	Updated repo README.md	2 days ago
SECURITY.md	Add files via upload	4 minutes ago
Tailspin.SpaceGame.Web.sln	Add files via upload	4 minutes ago
gulpfile.js	Add files via upload	4 minutes ago
package-lock.json	Add files via upload	4 minutes ago
package.json	Add files via upload	4 minutes ago

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



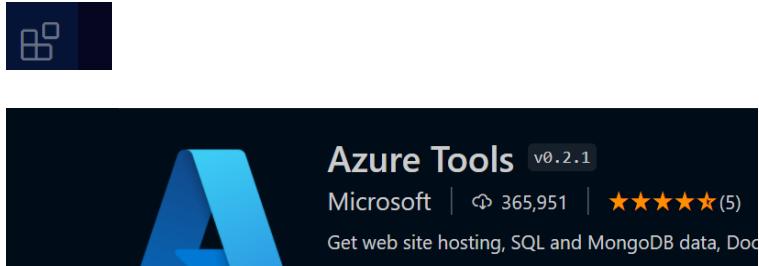
4. Go to your fork of the Space Game project. The forked repository will be saved as "*<your-account-name>/AzureAppServices*".
5. Select "Code". Under the "HTTPS" tab, copy the URL.



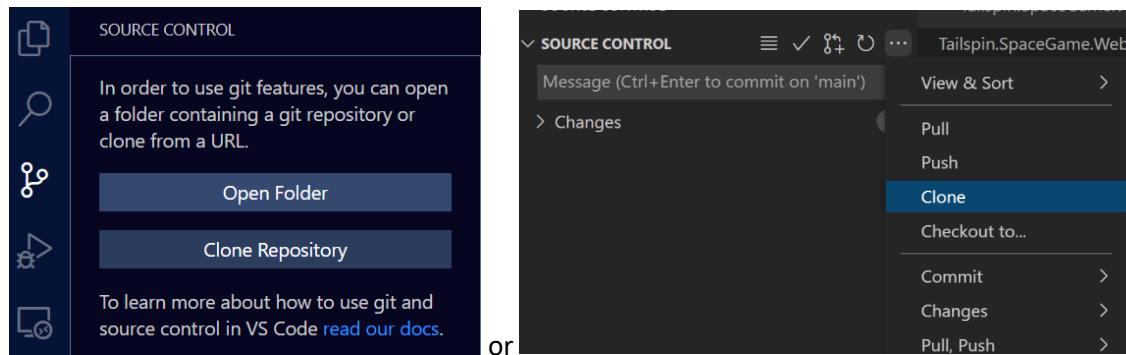
The screenshot shows the 'Code' dropdown menu for a forked repository. The 'Clone' section is expanded, displaying the following options:

- Clone
- HTTPS SSH GitHub CLI
- https://github.com/divim/AzureAppServices (highlighted with a red box)
- Use Git or checkout with SVN using the web URL.
- Open with GitHub Desktop
- Open with Visual Studio
- Download ZIP

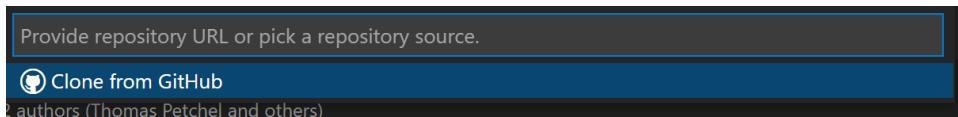
6. Open Visual Studio code.
7. Install the necessary extensions to visual studio code. Click on the **Extensions** button (or **CTRL+Shift+X**) on the left side ribbon, search for "Azure Tools" extension (from Microsoft), and install it.



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



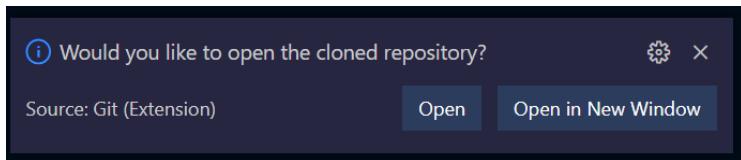
9. Click on **Clone from GitHub**



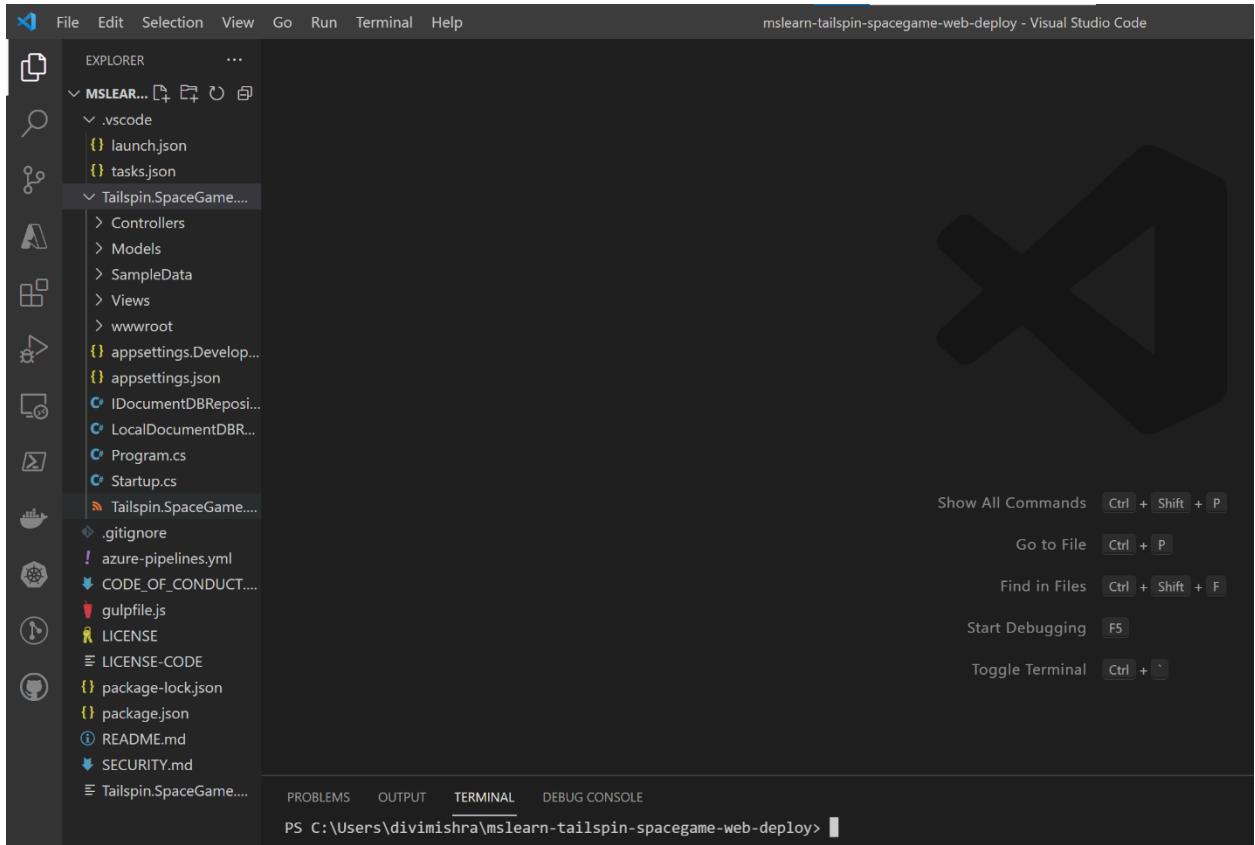
10. Sign-in and choose **AzureAppServices**.

11. Select a folder on your local machine to clone the files.

12. Once the clone is complete, in the visual studio code's prompt for opening the repository, click "Open".



13. 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. Please switch the directory to **Tailspin.SpaceGame.Web** folder
2. Enter the following command on the terminal to make sure the code runs on your local host:

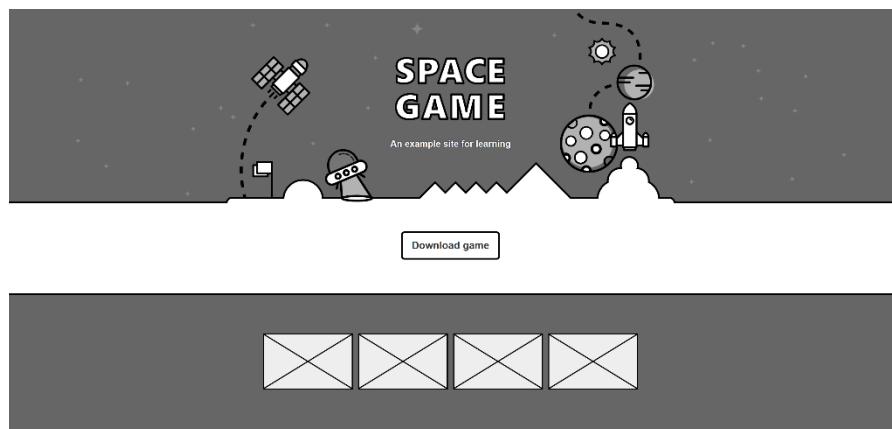
```
$ dotnet build --configuration Release
```

```
$ 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: ProductionContent root path: C:\Users\divimishra\mslearn-tailspin-spacegame-web-deploy\Tailspin.SpaceGame.WebNow listening on: http://localhost:5000Now listening on: https://localhost:5001Application started. Press Ctrl+C to shut down.
```



Ctrl+C when you're done.

Publish the project to Azure App Service

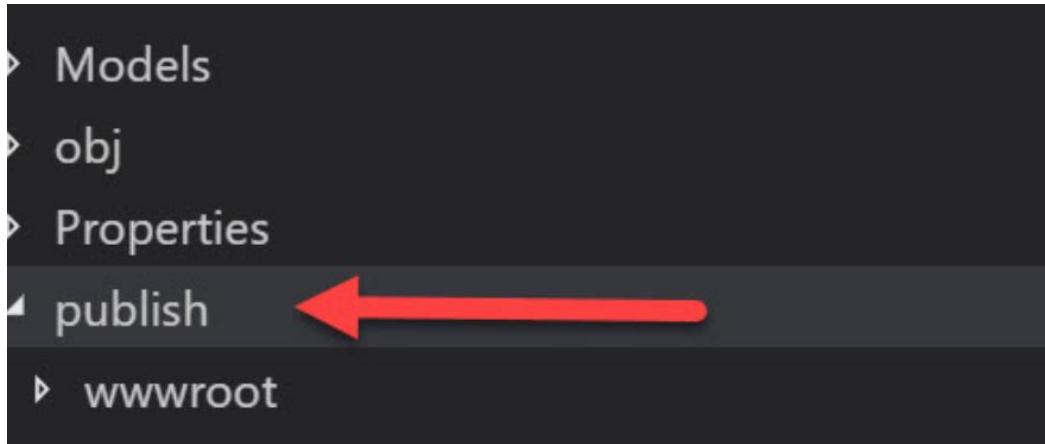
1. Install the Azure icon Click on the Azure icon in Visual Studio Code's left side ribbon



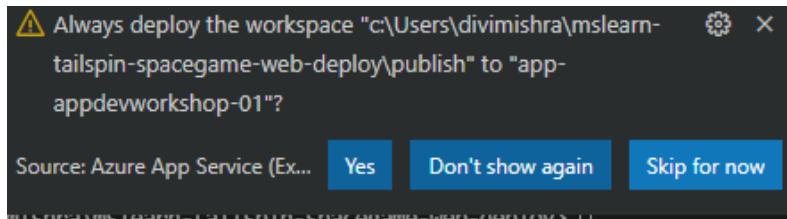
2. Sign in to your Azure Account when prompted.
3. Drill down into your subscription -> resource group -> App service (that you created in step 1).
4. Visual studio code will prompt you to install the app service extension. Click "Install" to Install the extension.
5. Once the extension is installed, 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



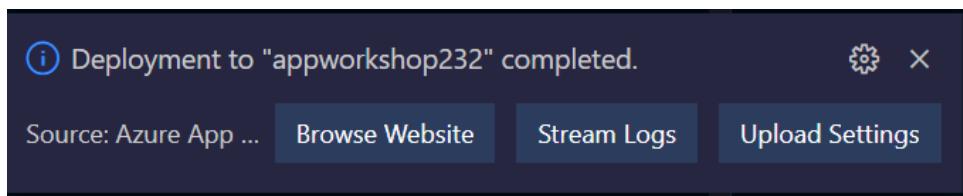
7. Right click on the publish folder.
8. Select Deploy to web app... and select the right subscription.
9. Select the Azure App service you had created.
10. Select Deploy to confirm.
11. If you get the following prompt, click on **Don't show again**. We will be deploying Azure Pipelines for CI/CD in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

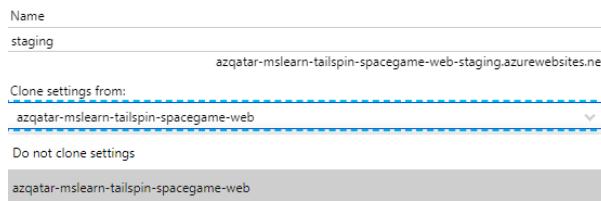
Create deployment slots

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

The screenshot shows the 'Deployment slots' page for an Azure App Service named 'my-demo-app'. On the left, there's a navigation menu with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Security, Deployment (Quickstart, Deployment slots, Deployment Center), Settings, and Configuration. The 'Deployment slots' option is selected and highlighted with a red box. In the main content area, there's a heading 'Deployment Slots' with a sub-instruction: '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.' Below this, a table lists the existing slot: NAME (my-demo-app), STATUS (Running), APP SERVICE PLAN (myAppServicePlan), and TRAFFIC % (100). At the top of the page, there are buttons for Save, Discard, Swap, and Refresh, with a red box around the 'Add Slot' button.

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 102: Monitor and Scale your application

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.

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
```

Note: You will now see a new package reference in Tailspin.SpaceGame.Web.csproj.

9. 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();
        ]
    }
}

```

Save all the changes (Ctrl + S).

10. You now specify the instrumentation key. Under the appsettings.json file, edit the following:

```

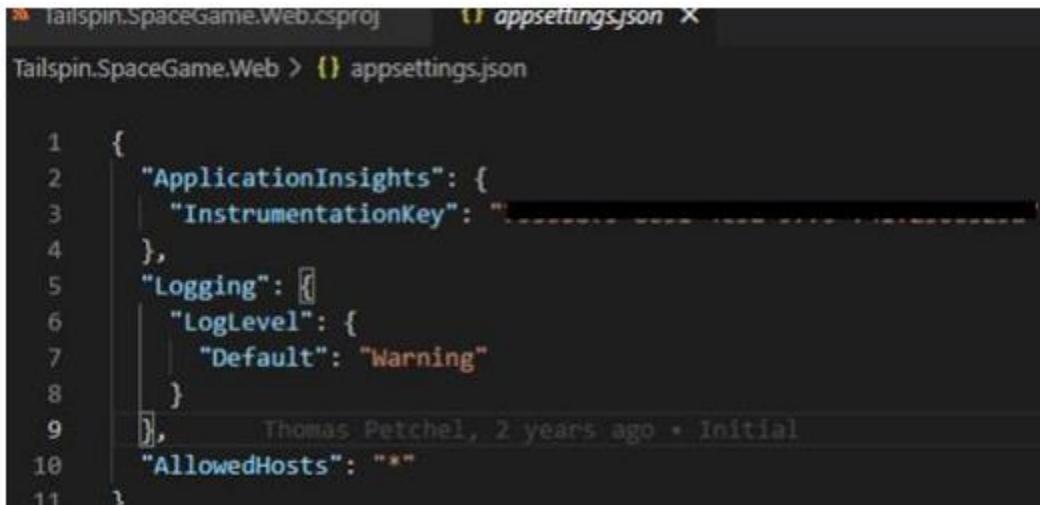
{
    "ApplicationInsights": {
        "InstrumentationKey": "putinstrumentationkeyhere"
    },
    "Logging": {
        "LogLevel": {
            "Default": "Warning"
        }
    }
}

```

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"];
```



```

1  {
2      "ApplicationInsights": {
3          "InstrumentationKey": "d6f4e333-425a-4a9d-a2a0-3c22d0e4a59a"
4      },
5      "Logging": {
6          "LogLevel": {
7              "Default": "Warning"
8          }
9      },
10     "AllowedHosts": "*"
11 }
```

11. Save all changes (Ctrl + S).

12. On the terminal:

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

```
$ dotnet build -c Release
```

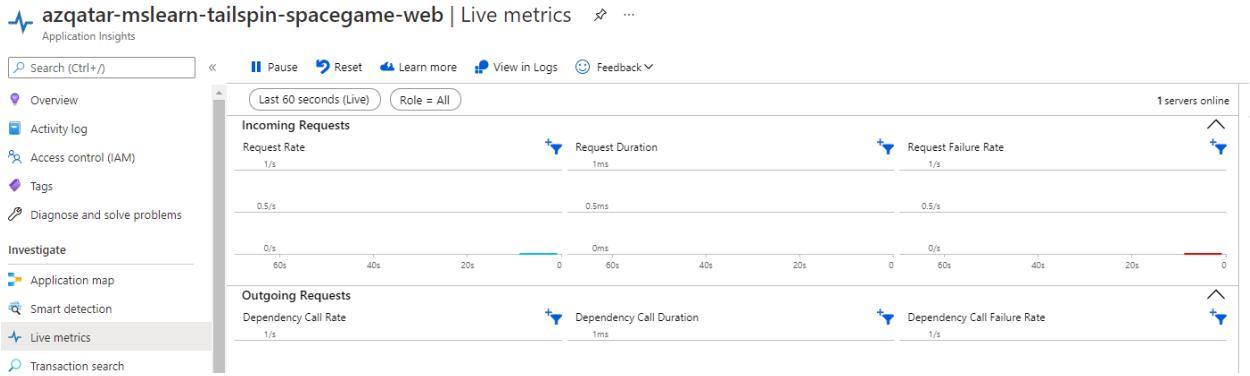
```
$ dotnet publish -c Release -o ./publish
```

13. 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).

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

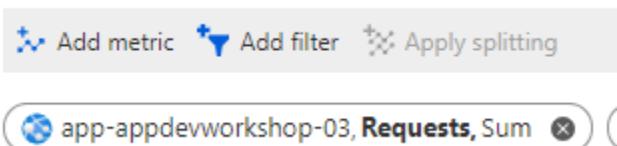
Please note that it may take a few minutes for the telemetry to appear.



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

Monitor apps

1. 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.

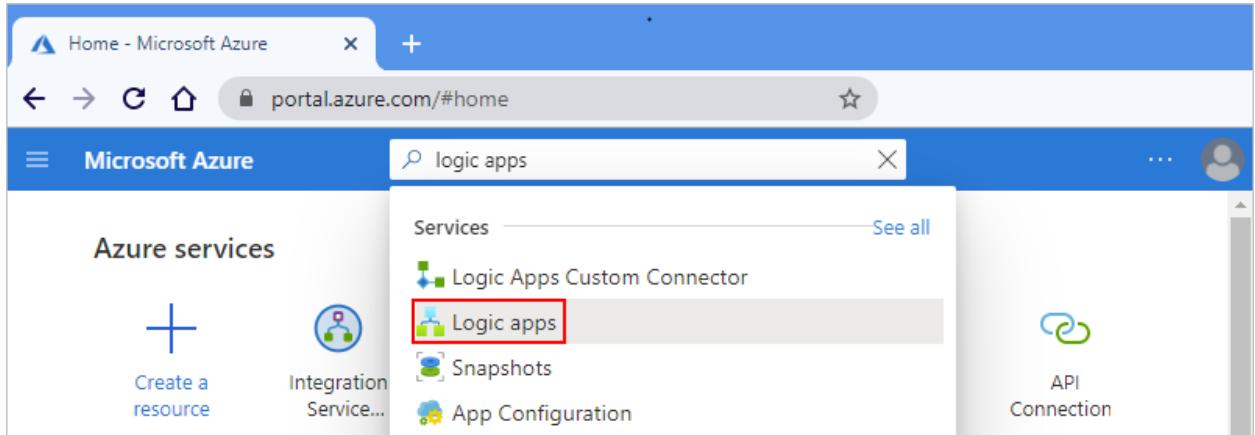
6. Select “**Pin to Dashboard**”. Click on the notification for viewing your dashboard. This is for developing shared dashboards to share with team members.



Configure alerts with Azure Logic Apps for Azure App Service

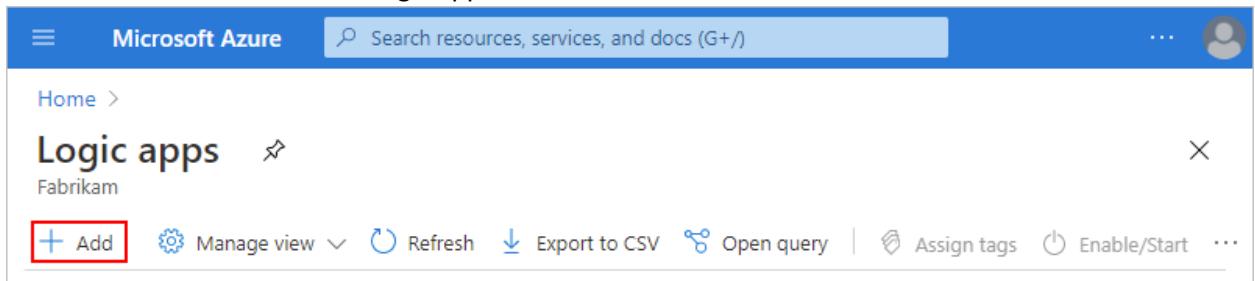
Creating the Logic App

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



The screenshot shows the Microsoft Azure portal interface. In the search bar at the top, 'logic apps' is typed. Below the search bar, the 'Services' section is visible, with 'See all' and several items listed: 'Logic Apps Custom Connector', 'Logic apps' (which is highlighted with a red box), 'Schemas', 'Snapshots', and 'App Configuration'. To the left, there's a sidebar with 'Azure services' and options like 'Create a resource' and 'Integration Service...'. On the right, there's an 'API Connection' section.

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



The screenshot shows the 'Logic apps' blade within the Azure portal. At the top, there's a breadcrumb navigation 'Home > Logic apps' and a search bar. Below that, it says 'Fabrikam'. There are several buttons: '+ Add' (which is highlighted with a red box), 'Manage view', 'Refresh', 'Export to CSV', 'Open query', 'Assign tags', 'Enable/Start', and three dots. The '+ Add' button is the focal point for creating 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**

Get started

Essentials

Resource group (change) : rg-ppdevworkshop-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 : ---

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.

Start from template

Choose from a template to start your workflow.

Quickstarts

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

Choose template

[Open quickstart guide](#)

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

Home > alertdemo

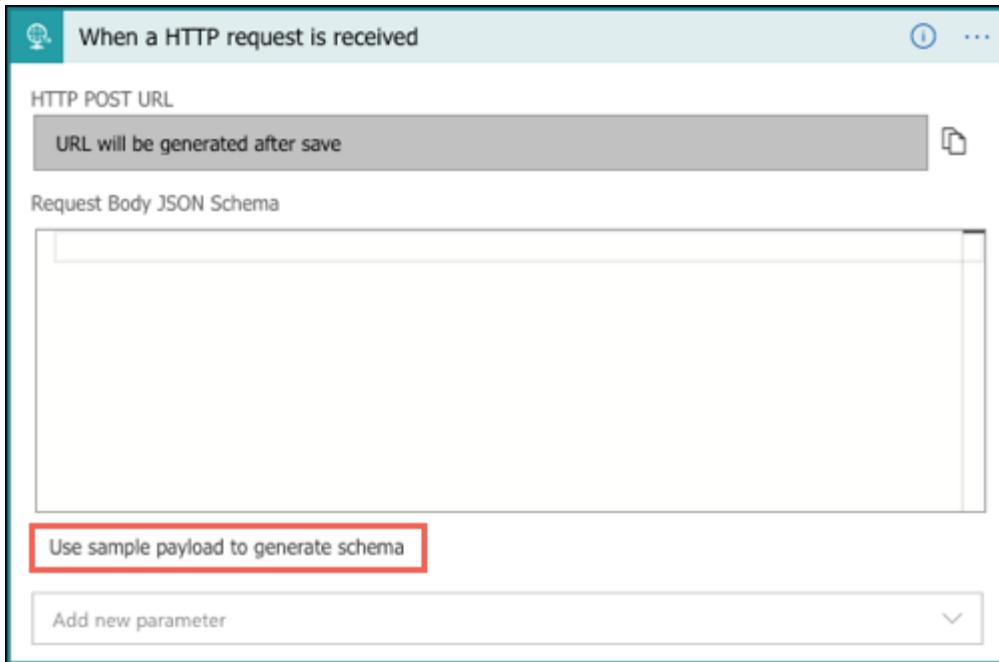
alertdemo | Logic app designer

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
Recurrence	When a new email is received in Outlook.com	When a new file is created on OneDrive
When an Event Grid resource event occurs	When a file is added to FTP server	

10. Click on Use sample payload to generate schema



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.

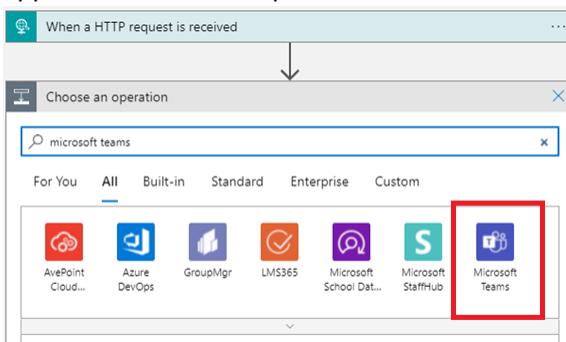
Remember to include a Content-Type header set to application/json in your request.

Got it Do not show again

14. Click on **+ New step**

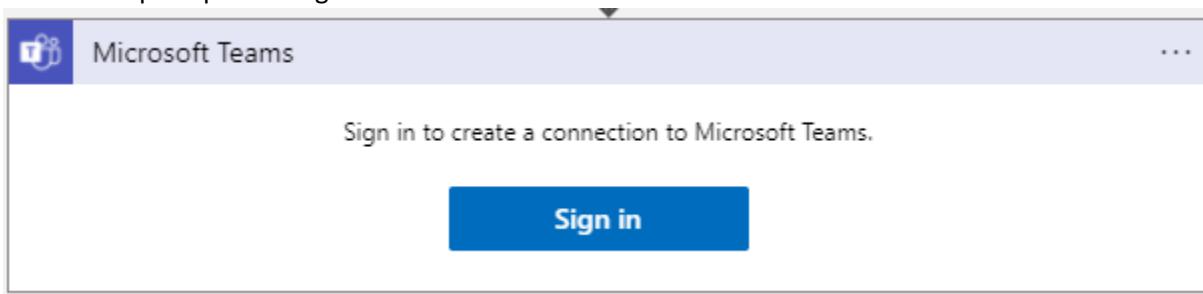
15. Search for **Microsoft Teams**

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

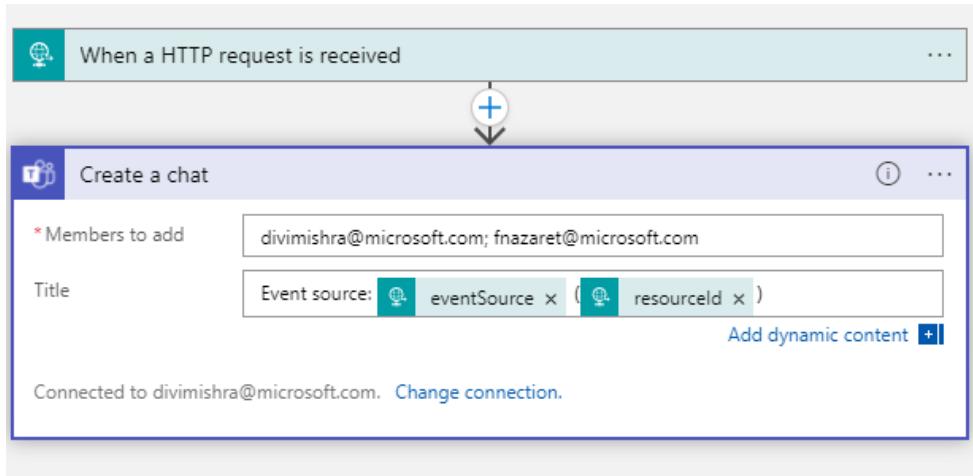


18. Enter your teammates email addresses.

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**.

Monitoring

! Alerts

app-appdevworkshop-03 | Alerts

App Service

Search (Ctrl+ /) New alert rule Manage alert rules Manage actions Refresh Feedback

3. Select “Add Alert Rule”.
4. Select the resource as the App Service.
5. Select the Condition as Http 400 or Http 4xx

Select a signal

Choose a signal below and configure the logic on the next screen to define the alert condition.

Signal type	Monitor service
All	All

Displaying 1 - 1 signals out of total 1 search results

Http 4xx

Signal name	Signal type	Monitor service
Http 4xx	Metric	Platform

- 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.

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%.

The screenshot shows the 'Custom Autoscale' configuration page. It displays a single scale rule with the following settings:

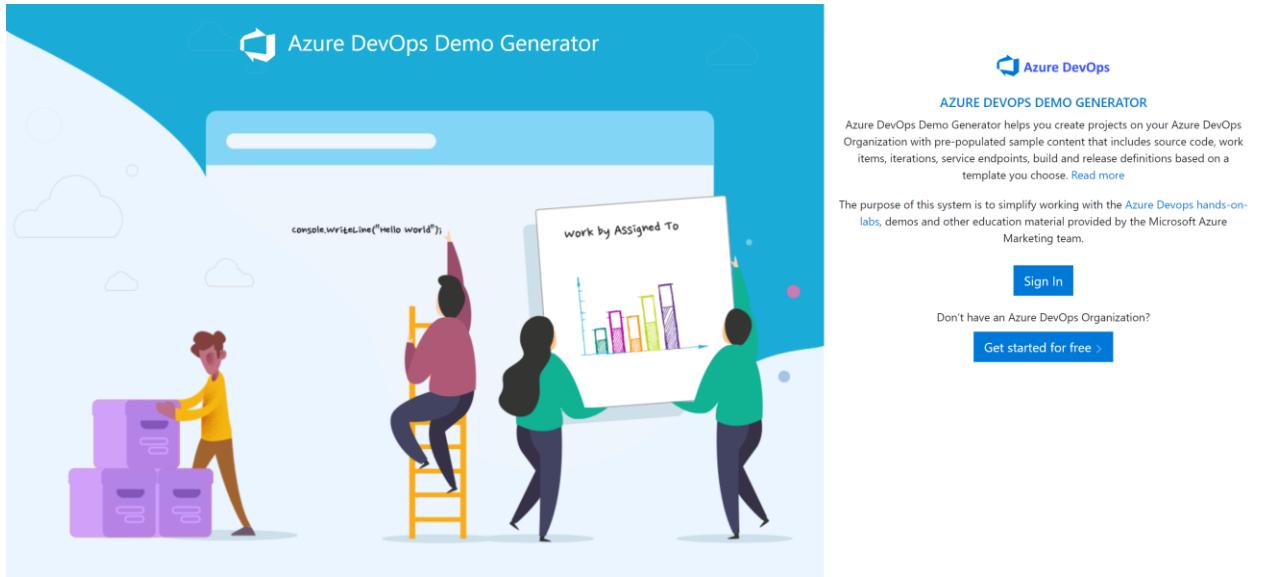
- Operator ***: Less than or equal to
- Metric threshold to trigger scale action ***: 10 %
- Duration (minutes) ***: 5
- Time grain (minutes)**: 1
- Time grain statistic ***: Average
- Action** section:
 - Operation ***: Decrease count to
 - Cool down (minutes) ***: 5
 - Instance count ***: 1

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

Lab 103: 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](#)



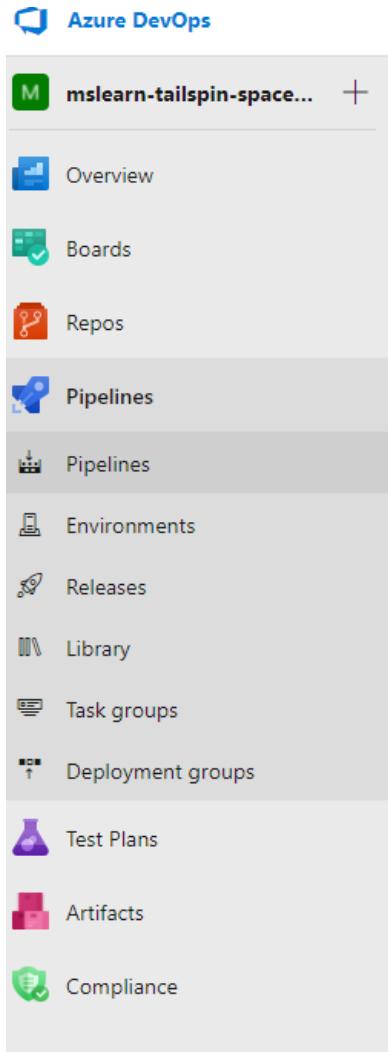
2. Sign in and accept the usage terms.
3. Create a new project with your Azure DevOps (ADO) organization and a project name. For Selected template: under MS learn, chose “Create a build pipeline with Azure Pipelines”. Finally, create project.

A screenshot of a web-based form for creating a new project. The form has three main fields: "New Project Name" with the value "Space Game - web - Pipeline", "Select Organization" with a dropdown menu showing "Select Organization", and "Selected Template" with a dropdown menu showing "Create a build pipeline with Azure Pip" and a "Choose template" button. At the bottom right of the form is a blue "Create Project" button, which is highlighted with a red border.

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

Create a build pipeline with Azure Pipelines

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



- Create a new pipeline.

A screenshot of the Azure Pipelines main page. At the top left is the word 'Pipelines'. To the right is a blue button with white text that says 'New pipeline'. Further to the right is a three-dot menu icon. Below this is a navigation bar with three tabs: 'Recent', 'All', and 'Runs'. On the far right of the navigation bar is a search bar with a magnifying glass icon and the placeholder text 'Filter pipelines'. The main area below the navigation bar is currently empty, showing a light grey background.

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**.

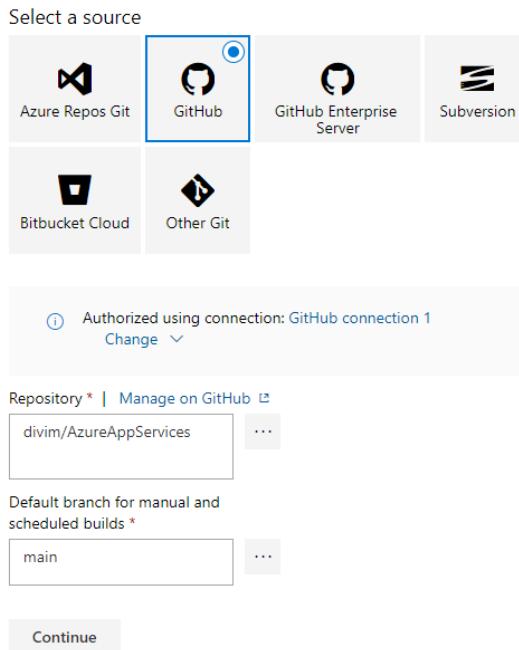
We need your authorization to access your repositories

Connection name *

Authorize using OAuth

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

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



Select “Continue”.

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

Select a template

Or start with an [Empty job](#)

asp.net core

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:

The screenshot shows the Azure Pipelines interface for a build pipeline named "mslearn-tailspin-spacegame-web-ASP....". The pipeline consists of three main stages:

- Get sources**: A task to fetch code from a "Space Game - web" repository using a "build-agent".
- Agent job 1**: An agent job set to run on an agent. It contains four tasks:
 - Restore**: dotnet restore
 - Build**: dotnet build
 - Test**: dotnet test
 - Publish**: dotnet publish
- Publish Artifact**: Publish build artifacts.

On the right side of the pipeline editor, there are configuration options for the pipeline, including the name "mslearn-tailspin-spacegame-web-ASP.NET Core-Cl", the agent pool "Azure Pipelines", and the agent specification "ubuntu-16.04". Parameters are also defined for the build and test steps.

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

The screenshot shows the configuration for the "Agent job 1" step. The job is set to "Run on agent". On the right side, there is a large black button with white text that reads "Add a task to Agent job 1". Above this button, there is a small blue plus sign icon.

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

10. Your pipeline will now look like this:

Build the app

Run on agent

+

Use .NET Core sdk 5.x

Use .NET Core

✓

Restore

.NET Core

Build

.NET Core

Test

.NET Core

Publish

.NET Core

↑ Publish Artifact

Publish build artifacts

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 *

Use .NET Core sdk 5.x

Package to install ⓘ

SDK (contains runtime) ▾

Use global json ⓘ

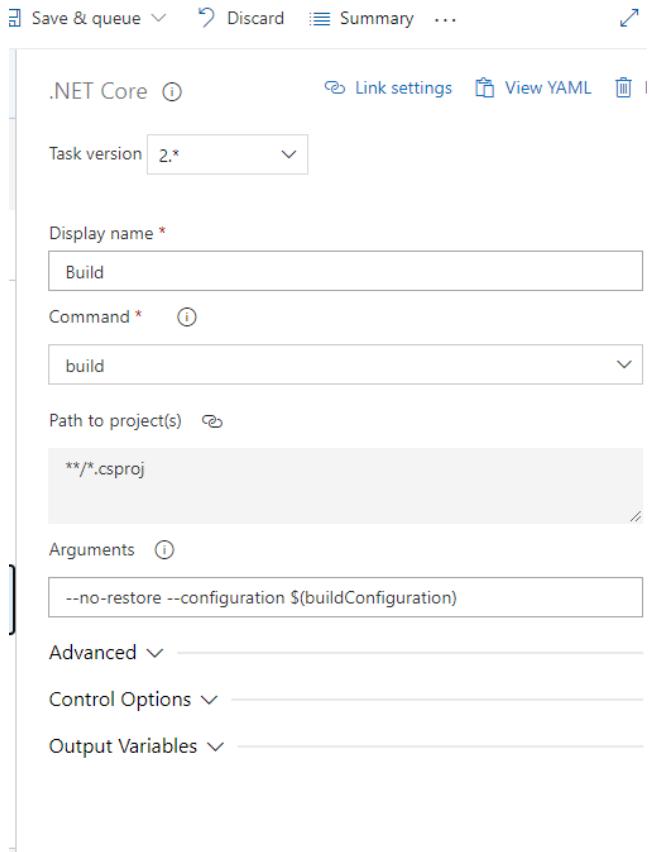
Version ⓘ

5.x

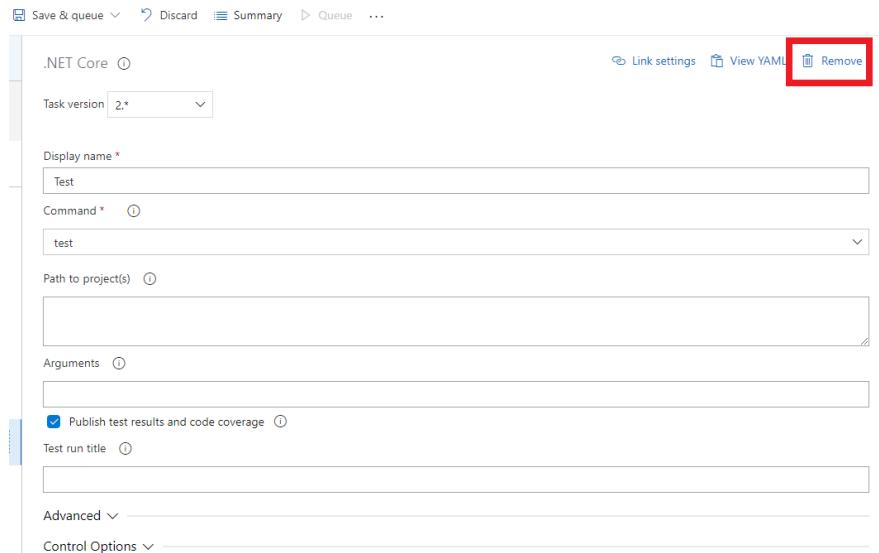
Include Preview Versions ⓘ

12. Click on the “Build” task and edit the “Arguments” as follows: `--no-restore --`

`configuration $(buildConfiguration)`



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.



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

a. Unselect the “**Publish web projects**” button.

b. Edit the arguments as follows: **--no-build --configuration**

\$(buildConfiguration) --output

\$(Build.ArtifactStagingDirectory)/\$(buildConfiguration)

The screenshot shows the configuration page for a .NET Core Publish task. At the top, there are buttons for Save & queue, Discard, Summary, and a copy icon. Below that, the task type is selected as ".NET Core". The Task version dropdown is set to "2.*". The Display name is "Publish". The Command dropdown is set to "publish". The "Publish web projects" checkbox is unchecked. The Path to project(s) field contains "**/*.csproj". In the Arguments section, the value is "--no-build --configuration \$(buildConfiguration) --output \$(Build.ArtifactStagingDirectory)/\$(buildConfiguration)". Under Advanced, two checkboxes are checked: "Zip published projects" and "Add project's folder name to publish path". There are also sections for Control Options and 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 configuration for the 'Publish build artifacts' task. The task version is set to 1.*. The display name is 'Publish Artifact'. The path to publish is set to \$(build.artifactstagingdirectory). The artifact name is 'drop'. The artifact publish location is set to 'Azure Pipelines'. Advanced, Control Options, and Output Variables sections are also visible.

Save & queue Discard Summary ...

Publish build artifacts ⓘ Link settings View YAML Re

Task version 1.*

Display name *

Publish Artifact

Path to publish *

\$(build.artifactstagingdirectory) ...

Artifact name *

drop

Artifact publish location *

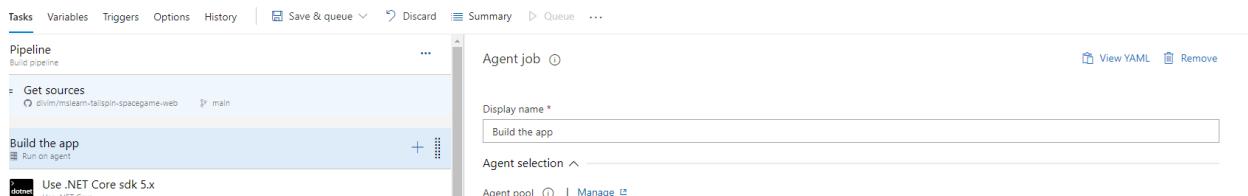
Azure Pipelines

Advanced

Control Options

Output Variables

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



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

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

Branch filters

Type	Branch specification
Include	main

Path filters

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

Agent pool

Azure Pipelines

Agent Specification *

ubuntu-16.04

Branch/tag

main

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.

You have successfully created your build pipeline.

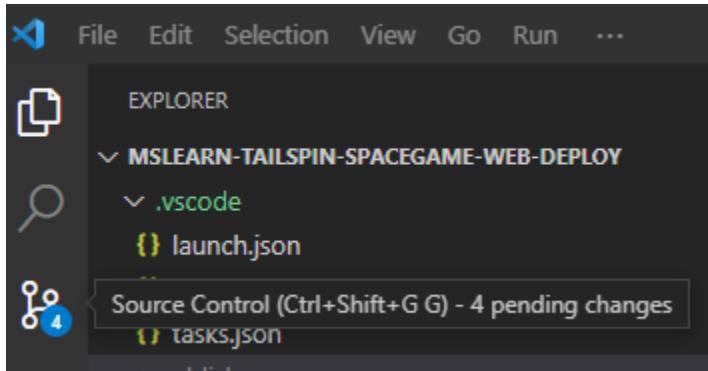
Source Control with GitHub

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.

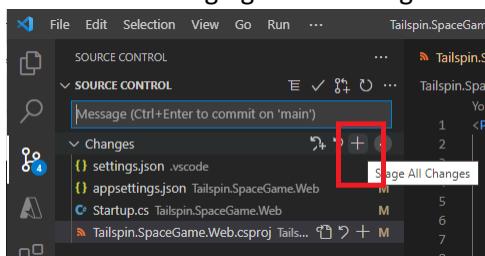
Recall that we had made some changes to the VS Code when we added the Application Insights instrumentation key.

Let's now push it to GitHub.

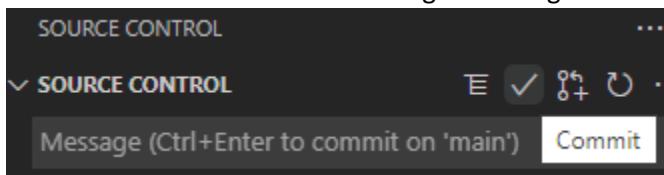
1. Navigate back to your VS Code and notice that your **Source Control** tab highlights pending changes.



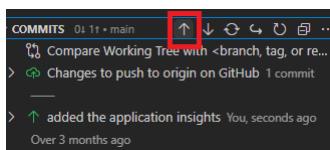
2. Click on the **Source Control** tab
3. Click on + for Staging all the changes



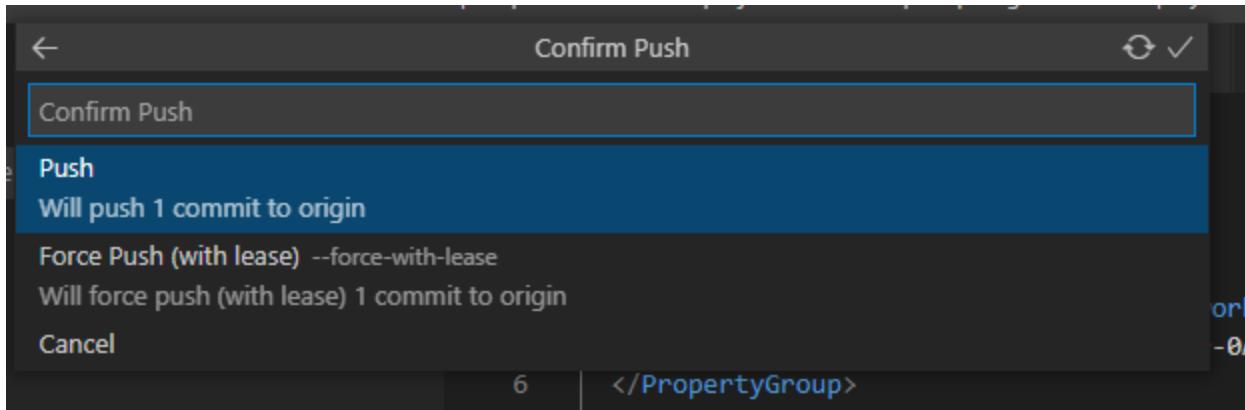
4. Click on the tic mark for Committing the changes



5. Enter a relevant message, such as "added app insights instrumentation key"
6. Under **COMMITS** (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 104: Testing with Azure Pipelines

Run unit tests locally

1. Open your VS Code workspace
2. Move to the testing directory:
\$ cd .\Tailspin.SpaceGame.Web.Tests\
3. Run the tests locally:

```
$ dotnet test
```

```

Test run for C:\Users\divimishra\OneDrive - Microsoft\Desktop\App Innovation Workshops\Personal\AzureAppServices\Tailspin.SpaceGame.Web.Tests\bin\Debug\net5.0\Ta
Microsoft (R) Test Execution Command Line Tool Version 16.11.0
Copyright (c) Microsoft Corporation. All rights reserved.

Starting test execution, please wait...
A total of 1 test files matched the specified pattern.

Passed! - Failed: 0, Passed: 5, Skipped: 0, Total: 5, Duration: 70 ms - Tailspin.SpaceGame.Web.Tests.dll (net5.0)

```

Run Code Coverage test locally

1. On your terminal, run the following commands:
 - a. *Create a local tool manifest file:*
\$ dotnet new tool-manifest
 - b. *Install ReportGenerator:*
\$ dotnet tool install dotnet-reportgenerator-globaltool
 - c. *Add the coverlet.msbuild package to the project:*
\$ dotnet add Tailspin.SpaceGame.Web.Tests package coverlet.msbuild
 - d. *Run the command for unit test and code coverage (the /p: tells coverlet what format to use and where to place the results):*
\$ dotnet build
\$ dotnet test --no-build --configuration Release /p:CollectCoverage=true
/p:CoverletOutputFormat=cobertura /p:CoverletOutput=./TestResults/Coverage/

After running this command, you'll see the following:

```

Starting test execution, please wait...
A total of 1 test files matched the specified pattern.

Passed! - Failed: 0, Passed: 5, Skipped: 0, Total: 5, Duration: 102 ms - Tailspin.SpaceGame.Web.Tests.dll (net5.0)

Calculating coverage result...
Generating report '.\TestResults\Coverage\coverage.cobertura.xml'

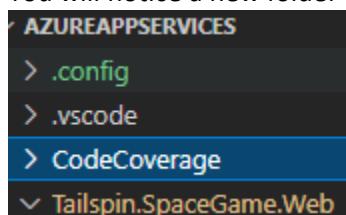
+-----+-----+-----+-----+
| Module          | Line | Branch | Method |
+-----+-----+-----+-----+
| Tailspin.SpaceGame.Web | 11.45% | 0%   | 18.91% |
+-----+-----+-----+-----+
| Tailspin.SpaceGame.Views | 0%   | 0%   | 0%   |
+-----+-----+-----+-----+

+-----+-----+-----+
|           | Line | Branch | Method |
+-----+-----+-----+
| Total    | 7.77% | 0%   | 14.58% |
+-----+-----+-----+
| Average  | 5.72% | 0%   | 9.45% |
+-----+-----+-----+

```

- e. *Use ReportGenerator to convert Cobertura to HTML*
\$ dotnet tool run reportgenerator -
reports:./Tailspin.SpaceGame.Web.Tests/TestResults/Coverage/coverage.cobertura.xml -targetdir:./CodeCoverage -reporttypes:HtmlInline_AzurePipelines

2. You will notice a new folder called CodeCoverage created



3. Right click on CodeCoverage's **Index.htm** file and open it in Explorer to view the file in your browser.

You will see your report:

Summary

Generated on:	9/28/2021 - 3:30:53 PM
Parser:	CoberturaParser
Assemblies:	2
Classes:	16
Files:	16
Covered lines:	15
Uncovered lines:	178
Coverable lines:	193
Total lines:	850
Line coverage:	7.7% (15 of 193)
Covered branches:	0
Total branches:	78
Branch coverage:	0% (0 of 78)
Covered methods:	7
Total methods:	47
Method coverage:	14.8% (7 of 47)

Risk Hotspots

No risk hotspots found.

Coverage

By assembly

Name	Covered	Uncovered
TailSpin.SpaceGame.Web	15	0
TailSpin.SpaceGame.Web.Controllers.HomeController	15	0
TailSpin.SpaceGame.Web\wwwroot\Environment\IRemoteControl.js	15	0

Add tests to Azure Pipeline

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

Azure DevOps

divimishra / mslearn-tailspin-spacegam... / Pipelines

Pipelines

Recent All Runs

Recently run pipelines

Pipeline	Last run
mslearn-tailspin-spacegame-web-ASP.NET Core-Cl	#20210904.3 • Update Index.cshtml Manually triggered for main aeb3243a 1m 30s

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

mslearn-tailspin-spacegame-web-ASP.NET Core-Cl

Runs Branches Analytics

Description	Stages
#20210904.3 Update Index.cshtml Manually triggered for main aeb3243a	1m 30s

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

Tasks Variables Triggers Options History Save & queue Discard Summary

Pipeline Build pipeline

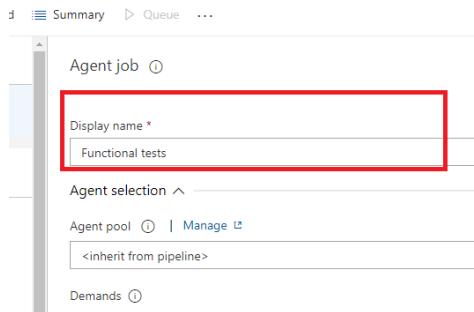
Get sources divim/mslearn-tailspin-spacegame-web main

...

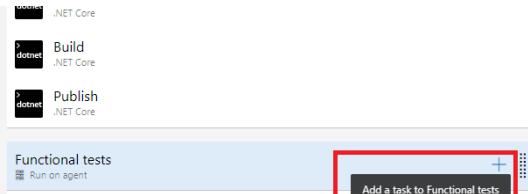
Add an agent job

Add an agentless job

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



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



6. Add the following jobs:

- a. Use .NET Core

Add tasks | Refresh

use .net core X

The screenshot shows a search result for 'use .net core'. It displays the 'Use .NET Core' task with its description: '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.' A blue 'Add' button is located on the right.

- b. .NET Core : Do this 4 times!

The screenshot shows a search result for '.NET Core'. It displays the '.NET Core' task with its description: 'Build, test, package, or publish a dotnet application, or run a custom dotnet command'. A blue 'Add' button is located on the right.

- c. Publish code coverage results

The screenshot shows a search result for 'Publish code coverage results'. It displays the 'Publish code coverage results' task with its description: 'Publish Cobertura or JaCoCo code coverage results from a build'. A blue 'Add' button is located on the right.

7. Click on the Use .NET Core task

- a. Version: 5.x

8. Click on the .NET Core #1 task

- a. Display Name: Restore tools from local manifest
- b. Command: custom
- c. Custom command: tool
- d. Arguments: restore

.NET Core ⓘ

Task version 2.* ▾

Display name *

Restore tools from local manifest

Command * ⓘ

custom

Path to project(s) ⓘ

Custom command * ⓘ

tool

Arguments ⓘ

restore

9. Click on the .Net Core #2 task:

- Display name: **Build**
- Command: **build**
- Path to project: ****/*.csproj**

.NET Core ⓘ

Link settings View YAML Remove

Task version 2.* ▾

Display name *

Build

Command * ⓘ

build ▾

Path to project(s) ⓘ

**/*.csproj

Arguments ⓘ

10. Click on the .Net Core #3 task:

- Display name: **Test the app**
- Command: **test**
- Path to project: ****/*.Tests.csproj**
- Arguments: **--no-build --configuration \$(buildConfiguration) /p:CollectCoverage=true /p:CoverletOutputFormat=cobertura /p:CoverletOutput=\$(Build.SourcesDirectory)/TestResults/Coverage/**
- Select the Publish test results and code coverage option

.NET Core ⓘ

Task version 2.* ▾

Display name *

Command *

Path to project(s) ⓘ

Arguments ⓘ

```
--no-build --configuration $(buildConfiguration) /p:CollectCoverage=true /p:CoverletOutputFormat=cobertura
/p:CoverletOutput=$(Build.SourcesDirectory)/TestResults/Coverage/
```

Publish test results and code coverage ⓘ

11. Click on the **.Net Core #4** task:

- Display name: **Run report**
- Command: **custom**
- Custom command: **tool**
- Arguments: **run reportgenerator -reports:\${Build.SourcesDirectory}/**/coverage.cobertura.xml -targetdir:\${Build.SourcesDirectory}/CodeCoverage -reporttypes:HtmlInline_AzurePipelines**

.NET Core ⓘ

Task version 2.* ▾

Display name *

Command *

Path to project(s) ⓘ

Custom command *

Arguments ⓘ

```
run reportgenerator -reports:${Build.SourcesDirectory}/**/coverage.cobertura.xml -targetdir:${Build.SourcesDirectory}/CodeCoverage -
reporttypes:HtmlInline_AzurePipelines
```

12. Click on the **Publish code coverage task**:

- Code coverage tool: **Cobertura**
- Summary file: **\$(Build.SourcesDirectory)/**/coverage.cobertura.xml**

Publish code coverage results ⓘ

Task version 1.* ▾

Display name *

Publish code coverage from \$(Build.SourcesDirectory)/**/coverage.cobertura.xml

Code coverage tool * ⓘ

Cobertura

Summary file * ⓘ

\$(Build.SourcesDirectory)/**/coverage.cobertura.xml

13. Finally, this is what your test job looks like:

The screenshot shows a pipeline configuration in a cloud-based CI/CD platform. The pipeline consists of the following steps:

- Use .NET Core sdk 5.x
- Restore tools from local manifest
- Build** (selected step, indicated by a blue checkmark)
- Test
- Run the report
- Publish code coverage from \$(Build.SourcesDirectory)/*... (Publish code coverage results)

14. Click on **Save & Queue** to save your progress and manually trigger the pipeline.

15. Navigate back to the **pipeline summary** of your latest run. Here, you will see the Test and coverage section:

#20210928.16 added code coverage features
on AzureAppServicesWorkshops-ASP.NET Core-CI

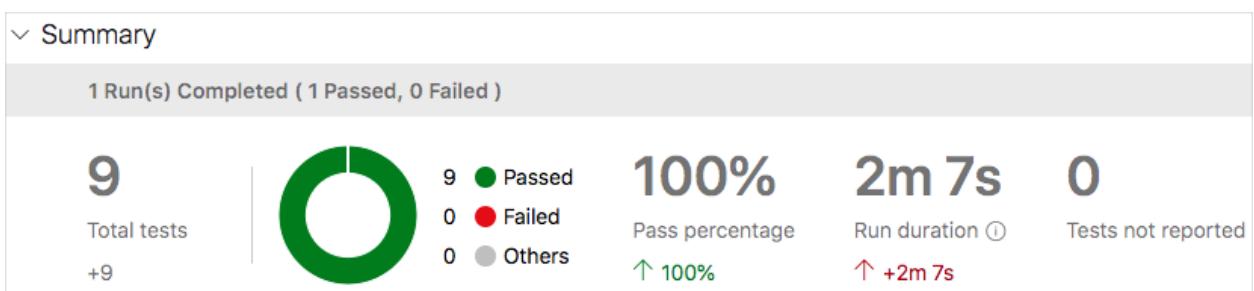
This run has been retained forever by 8 (Pipeline).

Summary **Code Coverage** Scans

Summary

Generated on:	9/28/2021 - 1:44:57 PM
Parser:	CoberturaParser
Assemblies:	1
Classes:	9
Files:	9
Covered lines:	15
Uncovered lines:	116
Coverable lines:	131
Total lines:	0
Line coverage:	11.4% (15 of 131)
Covered branches:	0
Total branches:	12
Branch coverage:	0% (0 of 12)

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



Lab 105: 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.

The screenshot shows the 'Runs' tab of a pipeline named 'mslearn-tailspin-spacegame-web-ASP.NET Core-CI'. A single run is listed: '#20210904.3 Update Index.cshtml', triggered manually for the 'main' branch. The run status is green, indicating success. It was completed 1 hour ago and took 1m 30s. There are 'Edit' and 'Run pipeline' buttons at the top right, and a three-dot menu icon at the top right corner of the table.

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

The screenshot shows the 'Project Settings' sidebar for the 'mslearn-tailspin-spacegame-w.' project. The sidebar includes sections for General (Overview, Teams, Permissions, Notifications, Service hooks, Dashboards), Boards (Project configuration, Team configuration, GitHub connections), Pipelines (Agent pools, Parallel jobs, Settings, Test management, Release retention, Service connections), Repos (Repositories), and Artifacts. The 'Service connections' option under Pipelines is highlighted with a red box.

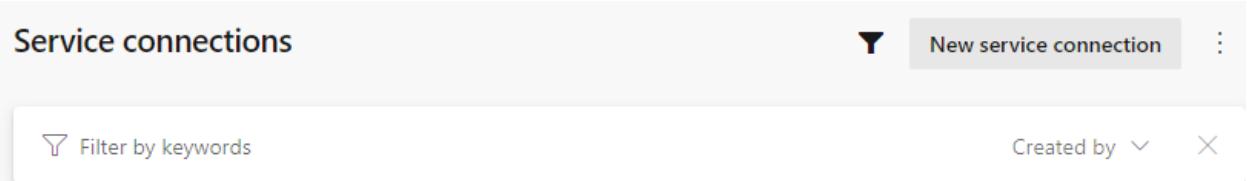
4. Click on **New Service Connection**

Service connections

New service connection

Filter by keywords

Created by



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

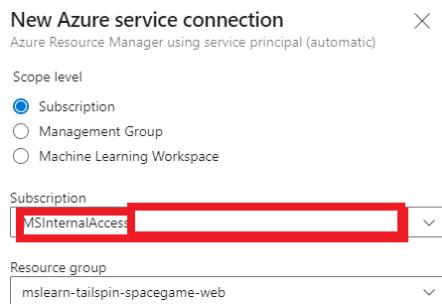
Machine Learning Workspace

Subscription

MSInternalAccess

Resource group

mslearn-tailspace-game-web

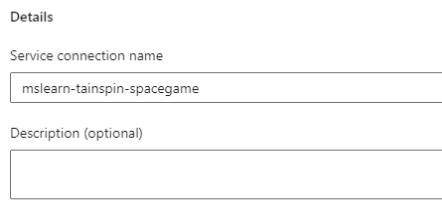


Details

Service connection name

mslearn-tailspace-game

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**.

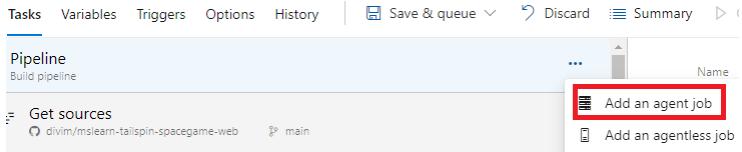
← mslearn-tailspace-game-web-ASP.NET Core-CI

Runs Branches Analytics

Edit Run pipeline

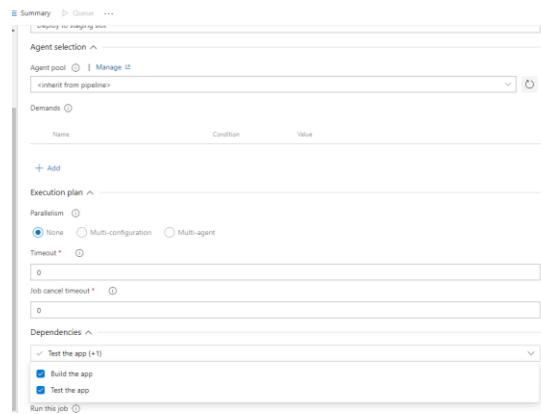


12. Select the three dots next to **Pipeline** and click on **Add an agent job**



13. Click on Agent job

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

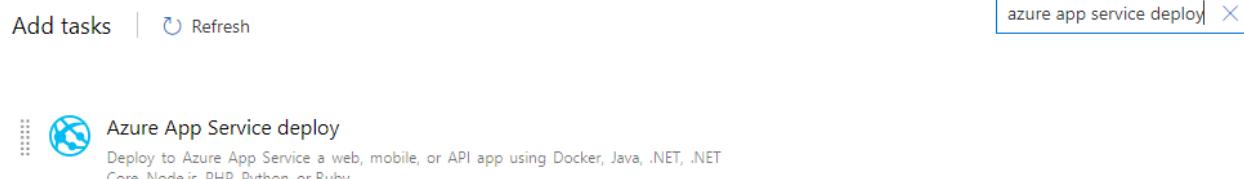


14. Add the following agent jobs:

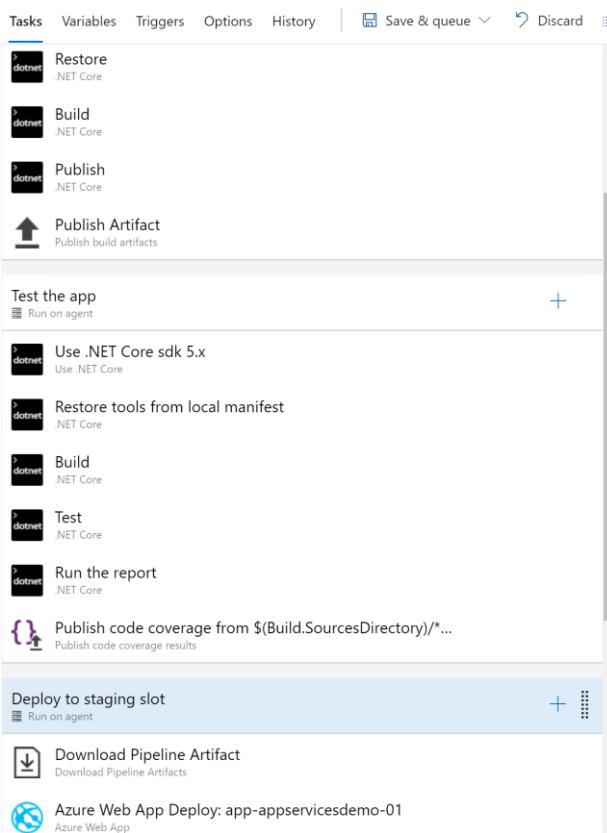
- Download pipeline artifact



- Azure App Service Deploy

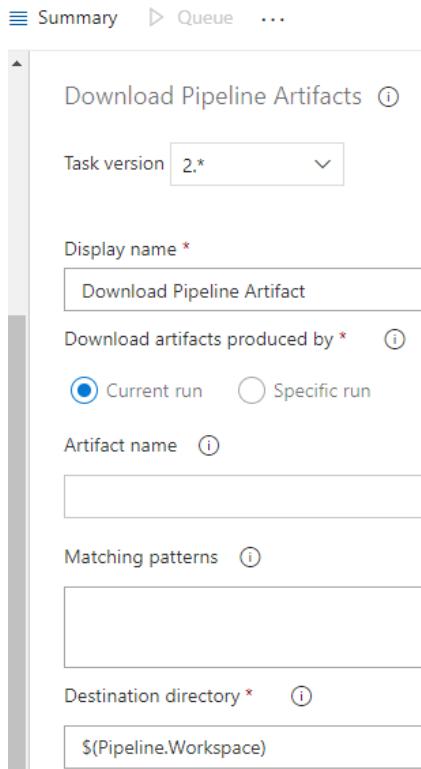


15. Ensure that your pipeline now looks like this:



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

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



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 * mslearn-tailspin-spacegame-rg

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

Link settings View YAML Remove

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 ...

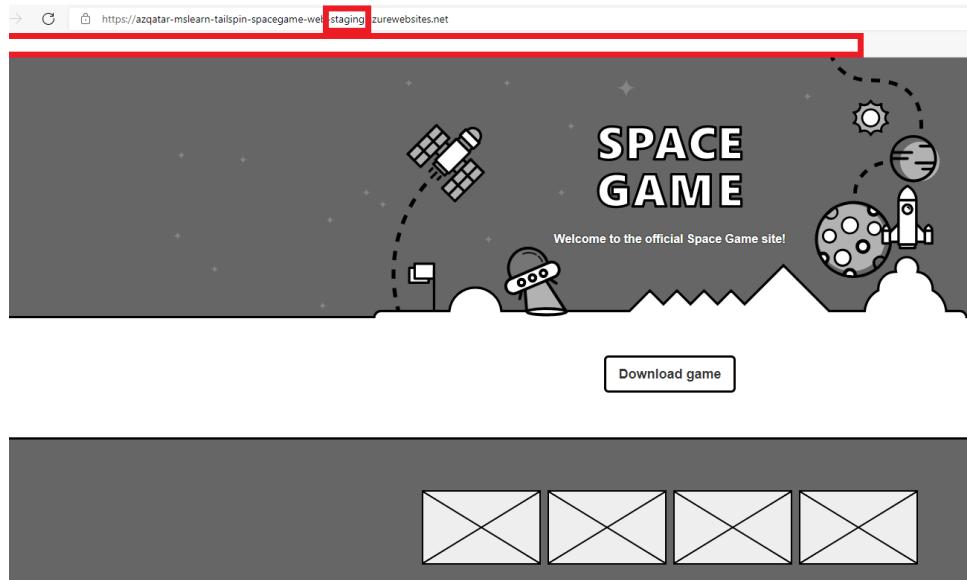
Overview Activity log Access control (IAM) Tags Diagnose and solve problems Security Events (preview) Deployment

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

Essentials JSON View

Resource group (change)	mslearn-tailspin-spacegame-web	URL	https://azqatar-mslearn-tailspin-spacegame-...
Status	Running	Health Check	Not Configured
Location	West Europe	App Service Plan	ASP-mslearntailsinspacegameweb-9893 (\$1:...
Subscription (change)	MSInternalAccess		

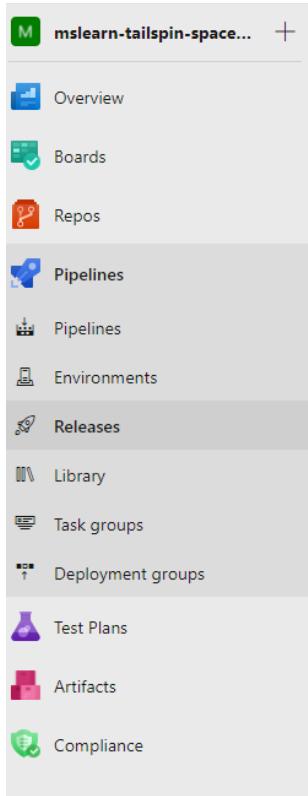
23. Your app has been deployed successfully to the staging slot. Note that the name of the URL is simply <app-service-name>-staging.azurewebsites.net



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**

Select a template

Search

0 start with an [Empty job](#)

Featured

- Azure App Service deployment
Deploy your application to Azure App Service. Choose from Web App on Windows, Linux, containers, Function Apps, or WebJobs.
- Deploy a Java app to Azure App Service
Deploy a Java application to an Azure Web App.
- Deploy a Node.js app to Azure App Service
Deploy a Node.js application to an Azure Web App.
- Deploy a PHP app to Azure App Service and Azure Database for MySQL
Deploy a PHP application to an Azure Web App and database to Azure Database for MySQL.
- Deploy a Python app to Azure App Service and Azure database for MySQL
Deploy a Python Django, Bottle, or Flask application to an Azure Web App and database to Azure Database for MySQL.
- Deploy to a Kubernetes cluster

4. Click on **Add an artifact**



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

A screenshot of the 'Tasks' tab for 'Stage 1'. The stage name is 'Stage 1 Deployment process'. Below it, there is a single task named 'Agent job' with the sub-task 'Run on agent'. To the right of the task list, there is a button labeled 'Add a task to Agent job' with a plus sign. Above the task list, there is a 'Stage name' input field containing 'Stage 1'.

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

A screenshot of a search interface. At the top, there is a search bar with the text 'Azure app service manage' and a close button. Below the search bar, there is a 'Add tasks' button and a 'Refresh' button. A list of search results is shown, with the first result being 'Azure App Service manage' which is highlighted with a blue border. The description for this result is: 'Start, stop, restart, slot swap, slot delete, install site extensions or enable continuous monitoring for an Azure App Service'.

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**.

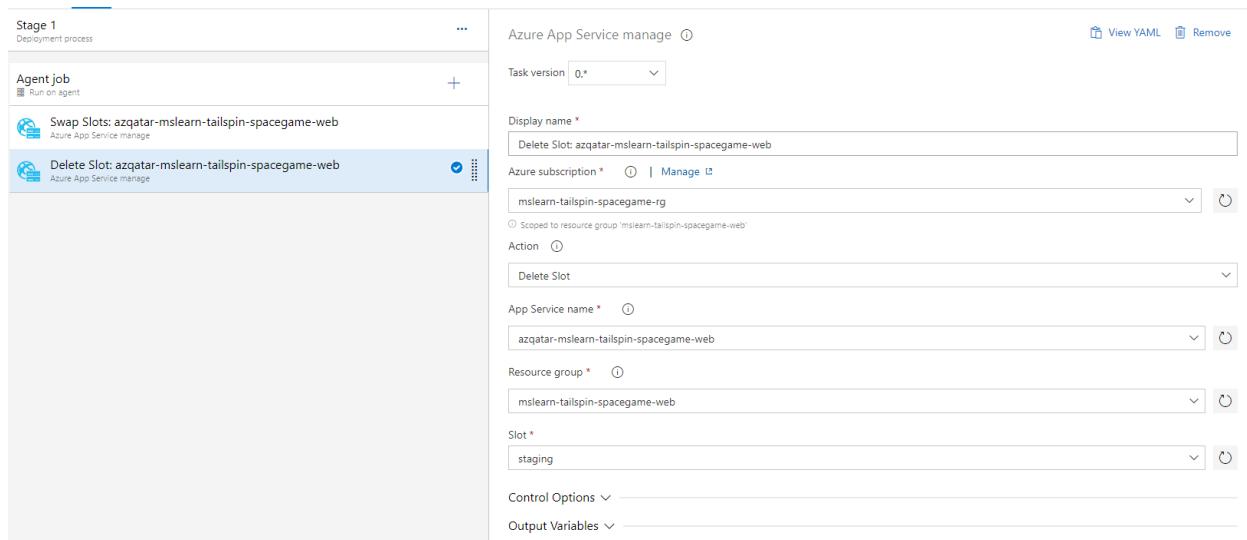
13. Add another task for **Azure App Service manage**.

The screenshot shows the Azure DevOps Pipeline editor interface. The top navigation bar includes Pipeline, Tasks (selected), Variables, Retention, Options, and History. The main area displays a pipeline stage named 'Stage 1 Deployment process' containing an 'Agent job' task and a 'Swap Slots' task. The 'Swap Slots' task is currently selected for editing. The configuration pane on the right shows the following settings:

- Task version: 0.*
- Display name: Swap Slots
- Azure subscription: (required, highlighted in red)
- Action: Swap Slots
- App Service name: (required, highlighted in red)
- Resource group: (required, highlighted in red)
- Source Slot: (required, highlighted in red)
 - Swap with Production
 - Swap with Staging

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**



15. Select **Save**.

16. Select **Create release**.



17. Observe the tasks on the release pipeline.

Task	Status	Duration
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.

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

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 trigger 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.

Lab 201: Protect your application

Deploy WAF with your application

Create WAF

1. Sign in to the Azure Portal
2. On the Azure Portal Menu, select **Create a resource**
3. Select **Networking > Application Gateway**
4. Under the Basics tab:
 - a. Resource group: Select the RG for your app service or create a new one
 - b. App Gateway Name: agw-demo
 - c. Tier: WAF V2
 - d. Virtual network: Create new virtual network
 - i. Name: vnet-agw
 - ii. Address space: Leave as default
 - iii. Subnets (x = value from address range):
 1. snet-agw: 10.x.0.0/24
 2. snet-backend: 10.x.1.0/24

Here is an example:

Create virtual network

The Microsoft Azure Virtual Network service enables Azure resources to securely communicate with each other in a virtual network. It provides logical isolation of the Azure cloud dedicated to your subscription. You can connect virtual networks to other virtual networks, on-premises network. [Learn more](#)

Name

ADDRESS SPACE
The virtual network's address space, specified as one or more address prefixes in CIDR notation (e.g. 192.168.1.0/24).

<input type="checkbox"/> Address range	Addresses	Overlap
<input type="checkbox"/> 10.2.0.0/16	10.2.0.0 - 10.2.255.255 (65536 addresses)	None
<input type="checkbox"/>	(0 Addresses)	None

SUBNETS
The subnet's address range in CIDR notation. It must be contained by the address space of the virtual network.

<input checked="" type="checkbox"/> Subnet name	Address range	Addresses
<input checked="" type="checkbox"/> snet-agw	10.2.0.0/24	10.2.0.0 - 10.2.0.255 (256 addresses)
<input type="checkbox"/> snet-backend	10.2.1.0/24	10.2.1.0 - 10.2.1.255 (256 addresses)
		(0 Addresses)

- e. Click on **OK**
5. **Frontends tab:**
 - a. Type: Public
 - b. Public IP Address: **Add new**

Traffic enters the application gateway via its frontend IP address(es). An application gateway can use a public IP address, private IP address, or one of each type.

Frontend IP address type Public Private Both

Public IP address

6. Backends tab:

- a. Add a backend pool
- b. Name: **be-pool-demo**
- c. Add backend pool without targets: **Yes**
- d. **Add** the backend pool

7. Configuration tab:

- a. Select **Add a routing rule**
- b. Rule name: **rr-demo**
- c. Listener:
 - i. Listener name: **listener-demo**
 - ii. Frontend IP: **Public**
 - iii. Protocol: **HTTP**
 - iv. Leave the rest as defaults
- d. Backend targets:
 - i. Target type: **Backend pool**
 - ii. Backend target: **be-pool-demo**
 - iii. HTTP Settings:
 1. **Add new**
 2. HTTP Settings name: **HTTP-rule**
 3. Leave the rest as default

Add a HTTP setting

[← Discard changes and go back to routing rules](#)

HTTP settings name *	<input type="text" value="HTTP-rule"/>
Backend protocol	<input checked="" type="radio"/> HTTP <input type="radio"/> HTTPS
Backend port *	<input type="text" value="80"/>
Additional settings	
Cookie-based affinity	<input type="radio"/> Enable <input checked="" type="radio"/> Disable
Connection draining	<input type="radio"/> Enable <input checked="" type="radio"/> Disable
Request time-out (seconds) *	<input type="text" value="20"/>
Override backend path	<input type="checkbox"/>
Host name	
By default, Application Gateway does not change the incoming HTTP host header from backend. Multi-tenant services like App service or API management rely on a specific endpoint. Change these settings to overwrite the incoming HTTP host header.	
Override with new host name	<input checked="" type="radio"/> Yes <input type="radio"/> No
Host name override	<input type="radio"/> Pick host name from backend target <input checked="" type="radio"/> Override with specific domain name
e.g. contoso.com	
Create custom probes	<input type="radio"/> Yes <input checked="" type="radio"/> No

4. Select **Add**

e. Select Add

The screenshot shows the Azure portal interface with the 'Configuration' tab selected. There are three main sections:

- Frontends:** Shows a list with 'Public: (new) pip-agw' and a 'Manage HTTP settings' button.
- Routing rules:** Shows a list with 'rr-demo' and a 'Manage HTTP settings' button.
- Backend pools:** Shows a list with 'be-pool-demo' and a 'Manage HTTP settings' button.

Below each section is a '+ Add' button for creating new items.

8. **Tags tab:** Add the tags relevant to your organization convention
9. Select **Review + Create**. Then, select **Create**.

Note: The deployment of the AGW may take 20-25 mins.

Add backend pool

1. Once the deployment is done, click on **Backend pools**. Select **be-pool-demo**.

The screenshot shows the 'Backend pools' blade for the 'waf-demo-01' application gateway. The left sidebar has 'Backend pools' selected. The main area shows a table with one row:

Name
bepool-demo

2. Under backend targets:
 - a. Target types: **App Services**
 - b. Target: Your app service
 - c. Save.
3. Select **HTTP Settings** for your AGW. Select the existing HTTP setting.
 - a. Under **Override with new host name**, select **Yes**.
 - b. Under **Host name override**, select **Pick host name from backend target**.

Add HTTP setting

HTTP settings name: http-set

Backend protocol: HTTP HTTPS

Backend port *: 80

Additional settings:

- Cookie-based affinity:
- Enable Disable
- Connection draining:
- Enable Disable

Request time-out (seconds) *: 20

Override backend path:

Host name:

By default, Application Gateway does not change the incoming HTTP host header from the client and sends the header unaltered to the backend. Multi-tenant services like App service or API management rely on a specific host header or SNI extension to resolve to the correct endpoint. Change these settings to overwrite the incoming HTTP host header.

Override with new host name: Yes No

Host name override:

- Pick host name from backend target
- Override with specific domain name

e.g. contoso.com

Use custom probe: Yes No

- Once the deployment of the backend target is done, visit the frontend public IP address to see the app service deployed to it.

Set service endpoint-based rule

- On the Azure Portal, navigate to the app service.
- On the left pane, select **Networking**.
- On the networking pane, select **Access Restrictions** under **Inbound traffic**.
- Select **Add rule**

Access Restrictions

Access restrictions allow you to define lists of allow/deny rules to control traffic to your app. Rules are evaluated in prior

app-appservicesdemo-01.azurewebsites.net	app-appservicesdemo-01.scm.azurewebsites.net
+ Add rule	
<input type="checkbox"/> Priority	Name
<input type="checkbox"/> 1	Allow all
	Source
	Any

- Enter the following details:
 - Name: agw-service-endpoint
 - Action: Allow
 - Priority: 400
 - Type: **Virtual Network**
 - Select the virtual network and the agw's default subnet.
 - Leave the rest as default
- Note that if you try to now access the web app, you'll receive Error 403.

Error 403 - Forbidden

The web app you have attempted to reach has blocked your access.

Now, you can only access the web app through the AGW.

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).

The screenshot shows the Azure Portal's "Settings | Azure Defender plans" page. At the top, there is a search bar and a "Save" button. On the left, a sidebar titled "Settings" lists "Azure Defender plans" as the selected item, along with other options like "Auto provisioning", "Email notifications", "Integrations", "Workflow automation", "Continuous export", and "Cloud connectors". A purple banner at the top right says "Azure Defender provides enhanced security. Learn more >". The main content area is divided into two sections: "Azure Defender off" (left) and "Azure Defender on" (right). Both sections list several features with green checkmarks or red X's. In the "Azure Defender off" section, the features listed are: Continuous assessment and security recommendations (green), Azure Secure Score (green), Just in time VM Access (red X), Adaptive application controls and network hardening (red X), Regulatory compliance dashboard and reports (red X), Threat protection for Azure VMs and non-Azure servers (including Server EDR) (red X), and Threat protection for supported PaaS services (red X). In the "Azure Defender on" section, the features listed are: Continuous assessment and security recommendations (green), Azure Secure Score (green), Just in time VM Access (green), Adaptive application controls and network hardening (green), Regulatory compliance dashboard and reports (green), Threat protection for Azure VMs and non-Azure servers (including Server EDR) (green), and Threat protection for supported PaaS services (green). Below these sections, a yellow key icon indicates that "Azure Defender plan will apply to: 9 resources in this subscription". At the bottom, there is a link "Select Azure Defender plan by resource type" and a blue "Enable all" button.

5. Click on “Save” to enable Azure Defender.

6. Your app service is now under Azure Defender.

The screenshot shows the Azure Defender interface with various alert categories and counts: Active alerts (823), Affected resources (55), and Active alerts by severity (High: 19, Medium: 620, Low: 138). A search bar and filter options are visible. A detailed view of an alert is shown on the right, highlighting a 'Dangling DNS Record detected on App' entry.

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

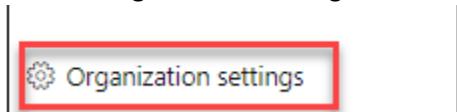
- the VM instance in which your App Service is running, and its management interface
- the requests and responses sent to and from your App Service apps
- the underlying sandboxes and VMs
- App Service internal logs - available thanks to the visibility that Azure has as a cloud provider

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

Lab 202: UI Testing with Selenium

Generate PAT for ADO

1. Navigate to your Azure DevOps homepage
2. Click on Organization settings in the bottom left.



3. At the top right, click on User Settings



4. Click on Personal Access Tokens
- a. Name: **Selenium Agent**
- b. Expiration as desired
- c. Scopes:
 - Full access**
 - (OR) Agent pools: Read & Manage; Auditing: Read Audit Log

Create a new personal access token X

⚠️ Your ability to create global personal access tokens (PATs) is restricted by your organization. [Learn more.](#)

Name

Organization

Expiration (UTC) Calendar icon

Scopes

Authorize the scope of access associated with this token

Scopes Full access Custom defined

Scopes
Authorize the scope of access associated with this token
Scopes Full access Custom defined

Agent Pools
Manage agent pools and agents
 Read Read & manage

Analytics
Read data from the analytics service
 Read

Auditing
Read audit log events, manage and delete streams.
 Read Audit Log

Build
Artifacts, definitions, requests, queue a build, and update build properties
 Read Read & execute

[Show less scopes](#)

Create

Cancel

5. Copy the PAT and store it somewhere safe as you won't be able to see it again. You will need this for the upcoming lab.

Configure the VM agent

1. Click on the Deploy to Azure button below to provision a Windows Server 2016 VM along with SQL Express 2017 and browsers (Chrome and Firefox). Deploy this on the same Azure RG as you used for your web app in the previous labs.

[**DEPLOY TO AZURE**](#)

This will take about 20 mins to deploy.

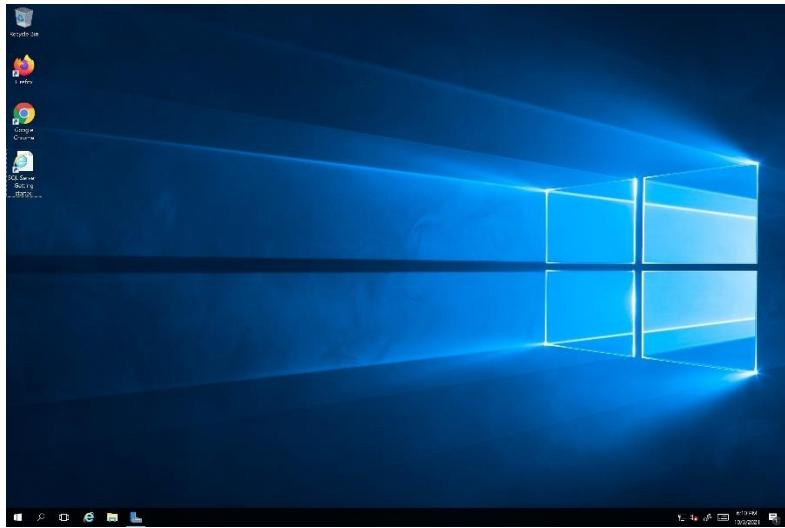
2. Navigate to your Azure Portal and open the resource group. You will notice the resources deployed:

Resources Recommendations (10)			
<input type="text"/> Filter for any field... <input type="button" value="Type == all"/> <input type="button" value="Location == all"/> <input type="button" value="Add filter"/>			
<input type="checkbox"/> Show hidden types <input type="button" value="No grouping"/>			
<input type="button" value="List view"/>			
<input type="checkbox"/> Name ↑↓	Type ↑↓	Location ↑↓	...
<input type="checkbox"/> vm-selenium	Network interface	West Europe	...
<input type="checkbox"/> vm-selenium-testing	Network interface	West Europe	...
<input type="checkbox"/> vm-selenium-testingsg	Network security group	West Europe	...
<input type="checkbox"/> vm-selenium-testingPip	Public IP address	West Europe	...
<input type="checkbox"/> vm-selenium-testingVnet	Virtual network	West Europe	...
<input type="checkbox"/> vm-seleniumumng	Network security group	West Europe	...
<input type="checkbox"/> vm-seleniumPip	Public IP address	West Europe	...
<input type="checkbox"/> vm-seleniumVnet	Virtual network	West Europe	...
<input type="checkbox"/> vmselenium	Disk	West Europe	...
<input type="checkbox"/> vmselenium	Virtual machine	West Europe	...
<input type="checkbox"/> vmselenium	Network interface	West Europe	...

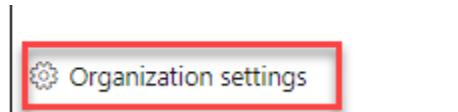
3. Select **vmselenium** virtual machine.
4. Download the RDP File for connecting to the VM.

The screenshot shows the Microsoft Azure portal interface for a virtual machine named 'vmselenium'. The left sidebar has a 'Connect' section selected. Under the 'RDP' tab, there's a note about enabling just-in-time access. The 'IP address' field is set to 'Public IP address (52.166.219.55)' and the 'Port number' is set to '3389'. A blue 'Download RDP File' button is at the bottom of this section.

5. Connect with the following credentials:
 - a. Username: vmadmin
 - b. Password: P2ssw0rd@123



6. In the VM, open any web browser and sign in to your Azure DevOps organization.
7. Click on **Organization Settings** at the bottom left.



8. Select **Agent pools** under Pipelines

A screenshot of the Azure DevOps 'Agent pools' settings page. The left sidebar shows 'Pipelines' with 'Agent pools' selected, indicated by a red box. The main area shows a table titled 'Agent pools' with one entry: 'Name: Azure Pipelines, Type: Azure Pipelines, Status: Enabled'. Below the table is a 'Default' button.

9. Select **Default** pool and select **Agents tab**. Click on **New agent**.

A screenshot of the 'Default' Agent pool settings page. The top navigation bar shows 'divimishra / Settings / Agent pools / Default'. The left sidebar shows 'Organization Settings' for 'divimishra'. The main area has tabs for 'Jobs', 'Agents' (which is highlighted with a blue box), 'Details', 'Security', 'Settings', 'Maintenance History', and 'Analytics'. The 'Agents' tab displays a table with columns 'Name', 'Status', and 'Actions'.

10. Under **Windows**, click on **Download**.

Get the agent

Windows (selected)

x64

x86

System prerequisites

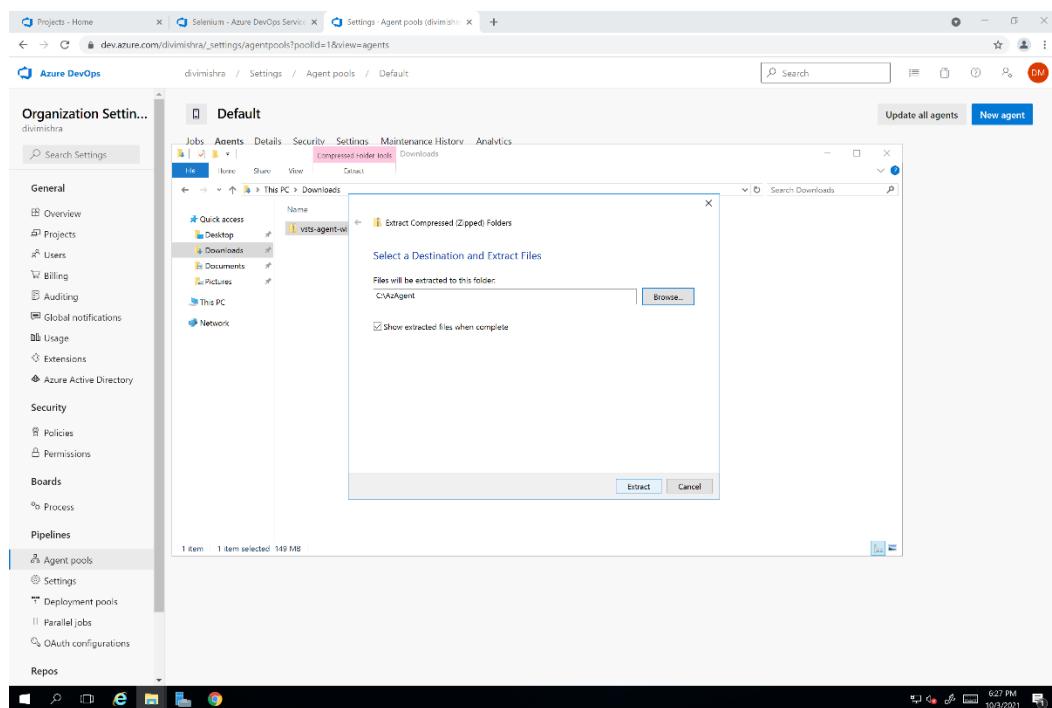
Configure your account

Configure your account by following the steps outlined [here](#).

Download the agent

Download

11. Make a directory in the C Drive named **AzAgent**.
12. Go back to your **Downloads** directory and extract the zipped folder from Step 10 on **AzAgent**.



This PC > Windows (C:) > AzAgent >

Name	Date modified	Type	Size
bin	5/8/2020 8:17 AM	File folder	
externals	5/8/2020 8:17 AM	File folder	
config	5/8/2020 8:16 AM	Windows Comma...	3 KB
run	5/8/2020 8:16 AM	Windows Comma...	4 KB

13. Open **Powershell** in administrator mode. Change to the **C:\AzAgent** path and type **.\config.cmd**
14. Press Enter.
15. Provide the following details:

a. **Enter server URL:** Your Azure DevOps Organization URL

- b. **Authentication type:** Press the **enter** key for **PAT** as the authentication type and paste the PAT you had configured in the next prompt.
 - c. Let us use the default options for the rest of the configuration. Press **Enter** for all prompts until the command execution completes.
16. Once the agent is registered, type **.\run.cmd** and hit **Enter** to start the agent.

```
PS C:\AzAgent> .\config.cmd

agent v2.166.4          (commit efdfb40)

>> Connect:
Enter server URL > https://dev.azure.com/ [REDACTED]
Enter authentication type (press enter for PAT) >
Enter personal access token > *****
Connecting to server ...

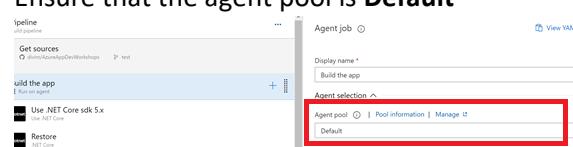
>> Register Agent:
Enter agent pool (press enter for default) >
Enter agent name (press enter for selenium) > [REDACTED]
Scanning for tool capabilities.
Connecting to the server.
Successfully added the agent
Testing agent connection.
Enter work folder (press enter for _work) >
2020-05-08 08:24:14Z: Settings Saved.
Enter run agent as service? (Y/N) (press enter for N) >
Enter configure autologon and run agent on startup? (Y/N) (press enter for N) >
PS C:\AzAgent> .\run.cmd
Scanning for tool capabilities.
Connecting to the server.
2020-05-08 08:24:27Z: Listening for Jobs
```

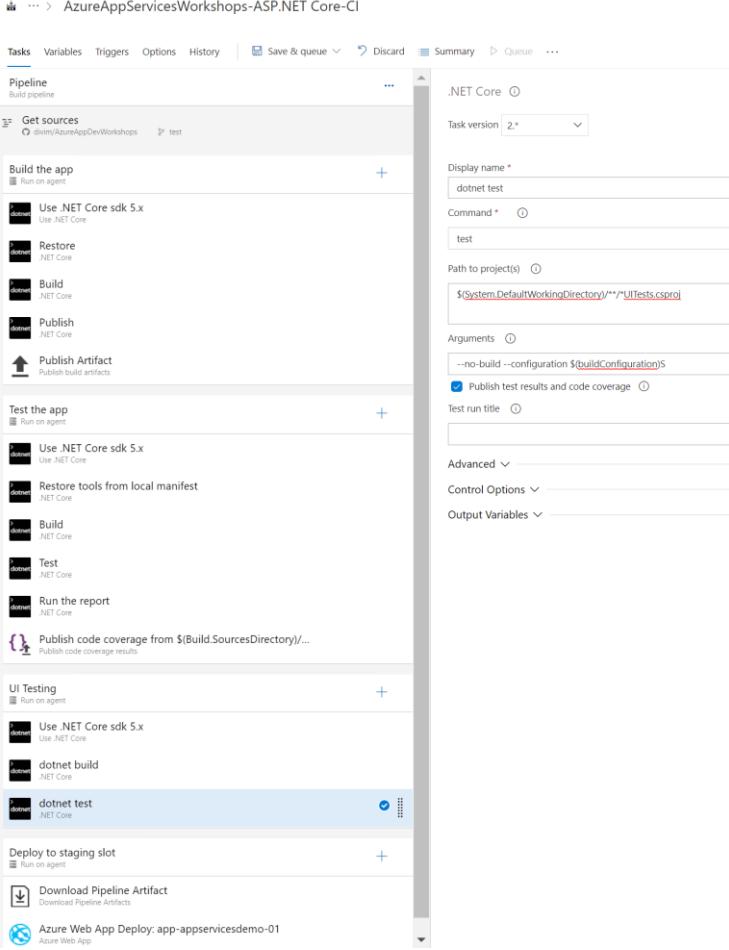
Configure the pipeline

1. Go to the pipeline on Azure DevOps.
2. For all agent jobs, under Agent Selection, select the **Agent Pool as Default**.
3. Under **Variables**, add the following variables:

SITE_URL	<code>https://<your-app-service-name>-01.azurewebsites.net</code>
CHROMEWEBDRIVER	<code>C:\SeleniumWebDrivers\ChromeDriver</code>
EDGEWEBDRIVER	<code>C:\SeleniumWebDrivers\EdgeDriver</code>
GECKOWEBDRIVER	<code>C:\SeleniumWebDrivers\GeckoDriver</code>

4. Add another agent job.
 - a. Name it **UI Testing**
 - b. Ensure that the agent pool is **Default**



- c. Add the following tasks:
- Use .NET Core**
 - Specify the SDK as 5.x
 - Dotnet Build**
 - Path to project:
`$(System.DefaultWorkingDirectory)/**/*UITests.csproj`
 - Dotnet Test**
 - Path to project:
`$(System.DefaultWorkingDirectory)/**/*UITests.csproj`
 - Arguments: `--no-build --configuration $(buildConfiguration)`
 - Select “Publish test results and code coverage”
5. Ensure that this is what your pipeline looks like
- 
6. Select **Save & Queue**
7. On the connected VM, see the UI tests executing. Note: It takes around 10 mins the first time you run it.
- In this lab, 4 UI tests are run on Chrome, Edge and Gecko.

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**.

Delete project

This will affect all contents and members of this project.
[Learn more about deleting projects](#)

Delete

6. Enter the project name and confirm deletion.
7. Navigate to Azure Portal and delete the resource group you have been working on.

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](#)
- 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 \(azuredavopslabs.com\)](#)

