

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	11

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 .. 25

Configure Azure DevOps project	25
Create a build pipeline with Azure Pipelines.....	26
Source Control with GitHub	35

LAB 104: TESTING WITH AZURE PIPELINES..... 37

Run unit tests locally	37
Run Code Coverage test locally	38
Add tests to Azure Pipeline.....	39

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

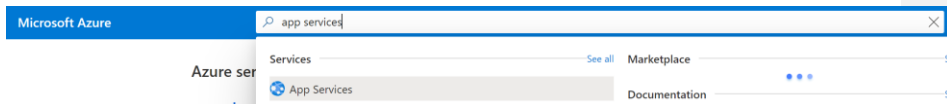
Deploy to staging slot from build pipeline	44
Build a release pipeline: zero-downtime deployment with slot swapping	51

LAB 201: UI TESTING WITH SELENIUM	60
Generate PAT for ADO	60
Configure the VM agent	61
Configure the pipeline	65
LAB 202: PROTECT YOUR APPLICATION	56
Deploy WAF with your application	56
Azure Defender for App Services	66
CLEAN-UP YOUR ENVIRONMENT	67
RESOURCES	68

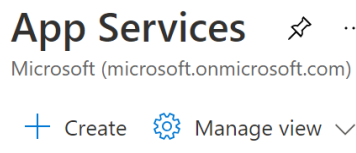
Lab 101: Azure App Service

Create an Azure App Service

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



3. Click on **+ Create**



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

Create Web App ...

all your resources.

Subscription * ⓘ

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](#)

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 *
100 total ACU, 1.75 GB memory

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

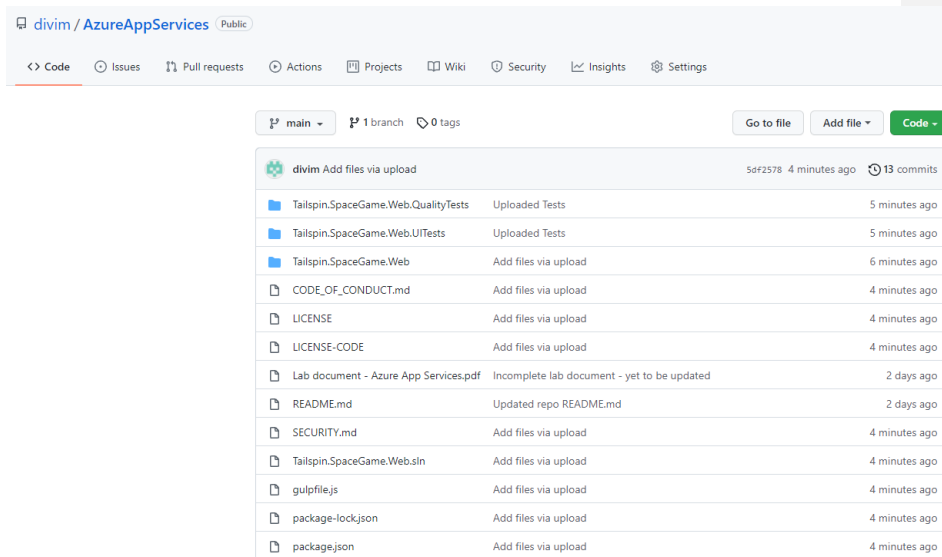
- Once the deployment is complete, click on “Go to resource”.
- From the overview tab, click on the “URL” to view the default web application that gets deployed.

Essentials			
Resource group (change)	: rg-appdevworkshop	URL	: https://appdevworkshop232.azurewebsites.net
Status	: Running	Health Check	: Not Configured
Location	: West Europe	App Service Plan	: asp-appdevworkshop (S1-1)

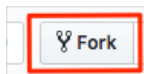
- You should be able to view a welcome page.

Fork the web project to your GitHub account

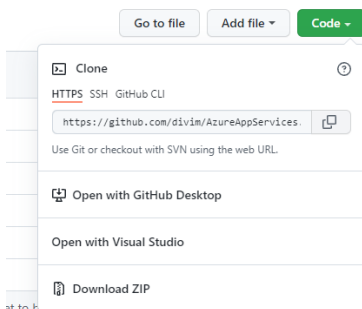
- Go to [GitHub](#) and sign in
- Visit the [Azure App Services workshop repo](#) (ASP.Net Core application)



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

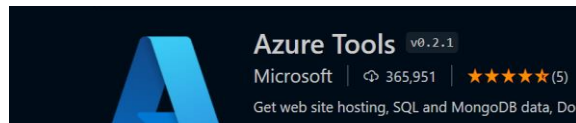


- Go to your fork of the Space Game project. The forked repository will be saved as “<your-account-name>/AzureAppServices”.
- Select “Code”. Under the “HTTPS” tab, copy the URL.

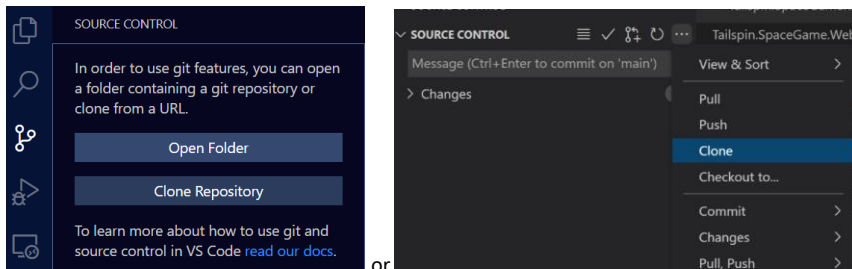


- Open Visual Studio code.
- Install the necessary extensions to visual studio code.

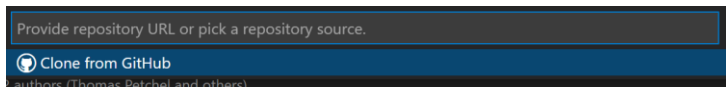
Click on the **Extensions** button (or **CTRL+Shift+X**) on the left side ribbon, search for “**Azure Tools**” extension (from Microsoft) and install it.



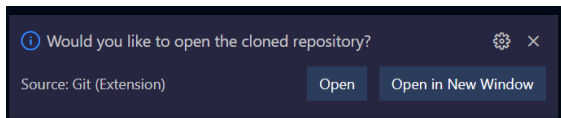
- Under **Source Control**. Click on **Clone Repository**.



- Click on **Clone from GitHub**



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



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

- 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 "username"
$ git config --global user.email "email"
```

Commented [MS1]: git config --global user.name "username"

quotation mark should be used instead of left and right quotation mark.. This is a unicode confilct

Commented [MS2]: git config --global user.email "email"

Run and validate the Project Locally.

1. Open the terminal by going to “View > Terminal” or “Ctrl + `” and switch the directory to **Tailspin.SpaceGame.Web** folder using the **cd** command.

\$ **cd** <path to Tailspin.SpaceGame.Web>

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

\$ **dotnet build -c Release**

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

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

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



Ctrl+C when you're done.

Publish the project to Azure App Service

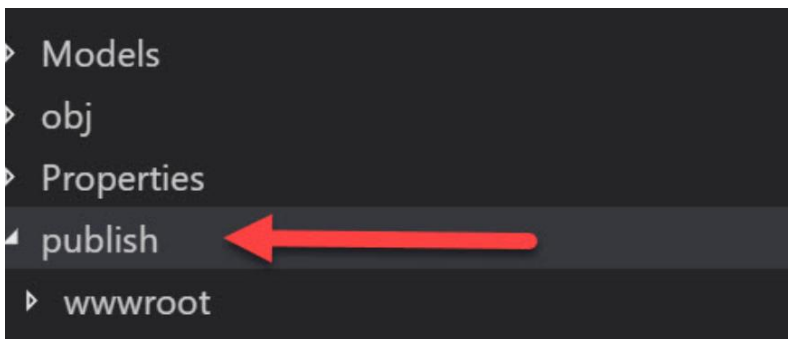
1. Click on the Azure icon in Visual Studio Code's left side ribbon



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:

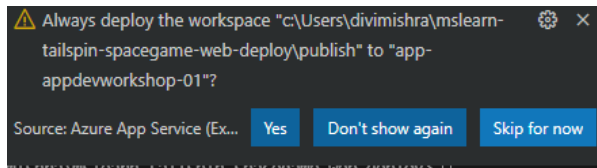
\$ dotnet publish -c Release -o ./publish

You will notice that a new publish folder has been created

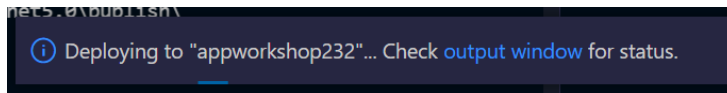


7. Right click on the **publish** folder.
8. Select **Deploy to web app...** and select the right subscription.
9. Select the Azure App service you have created.
10. Select **Deploy** to confirm.

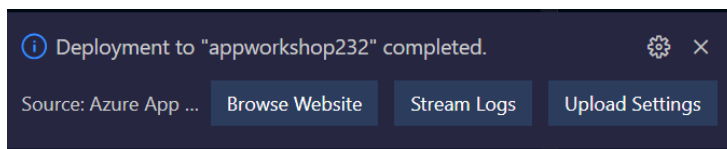
11. If you get the following prompt, click on “Don’t show again”.



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



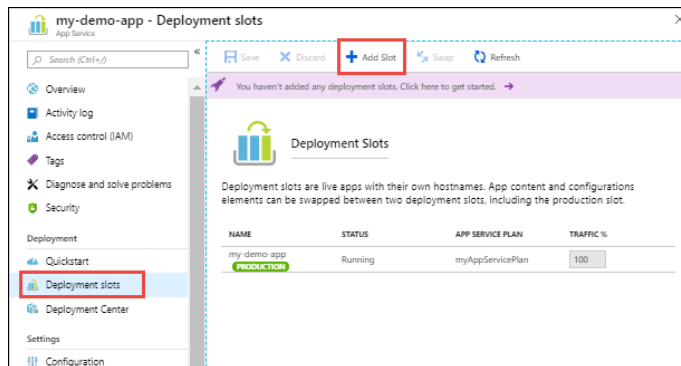
13. Once deployment is completed, select “Browse Website” button to open the azure app services website.



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

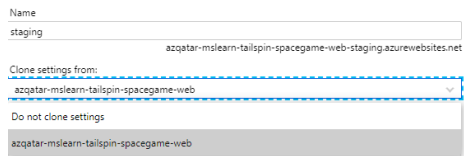
Create deployment slots

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



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

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



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.

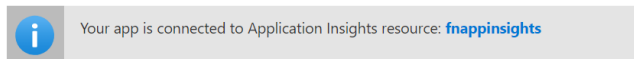
Note: If you would like to add an access rule restriction for the staging environment, please follow the steps from this document: [Azure App Service access restrictions - Azure App Service | Microsoft Docs](#)

Lab 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 : fnlawkspac

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).

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

```

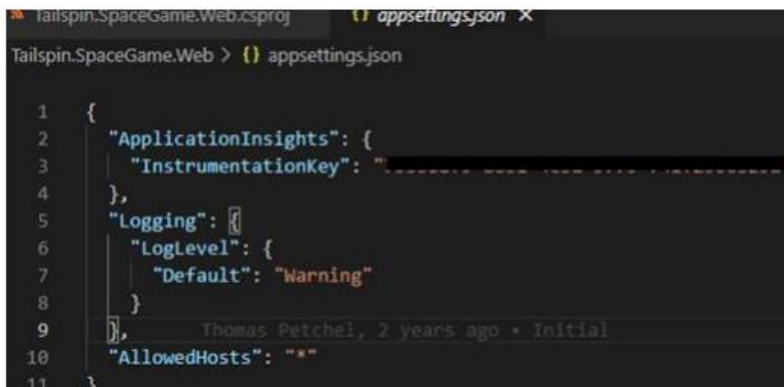
{
    "ApplicationInsights": {
        "InstrumentationKey": "putinstrumentationkeyhere"
    },
    "Logging": {
        "LogLevel": {
            "Default": "Warning"
        }
    }
}

```

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": "  
4   },  
5   "Logging": {  
6     "LogLevel": {  
7       "Default": "Warning"  
8     }  
9   },  
10  "AllowedHosts": "*" }  
11 }
```

11. Save all changes (Ctrl + S).

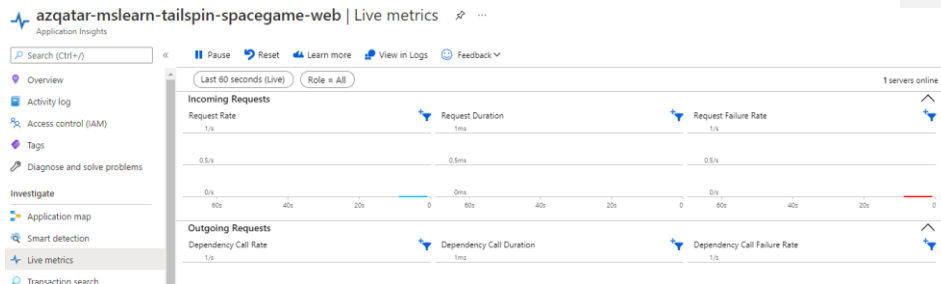
12. On the terminal:

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

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

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

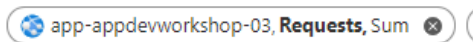
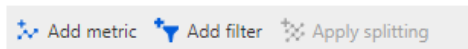
13. Right-click on the **publish** folder and click on **Deploy to web app** for the changes to be reflected. (Soon, you'll deploy a CI/CD pipeline using Azure Pipelines to automate this process and other processes for you).
14. To verify that the application insights package was configured accurately, navigate to **Live Metrics**. Please note that it may take a few minutes for the telemetry to appear.



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

Monitor apps

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



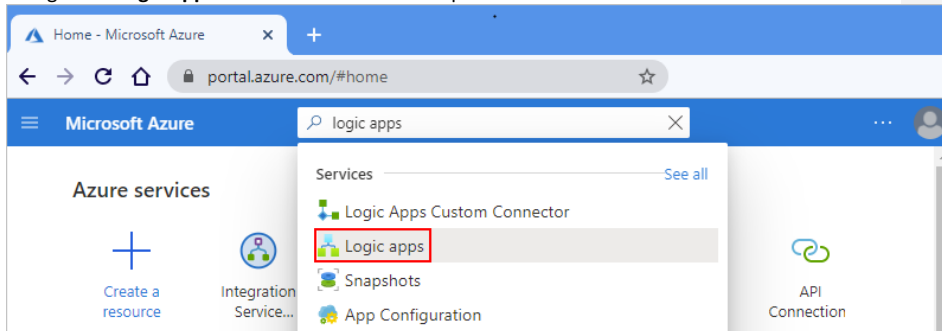
5. Select metrics of your choice and display them into meaningful bar representations. For example:
 - a. Select Add metric, and under the Metric dropdown list, select CPU Time. For Aggregation, select Sum.
 - b. Select Add metric, and under the Metric dropdown list, select Http Server Errors. For Aggregation,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



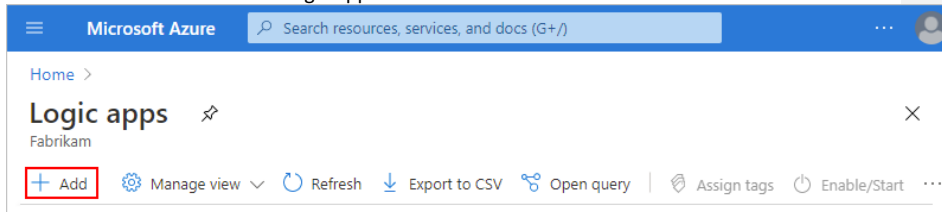
Configure alerts with Azure Logic Apps for Azure App Service

Creating the Logic App

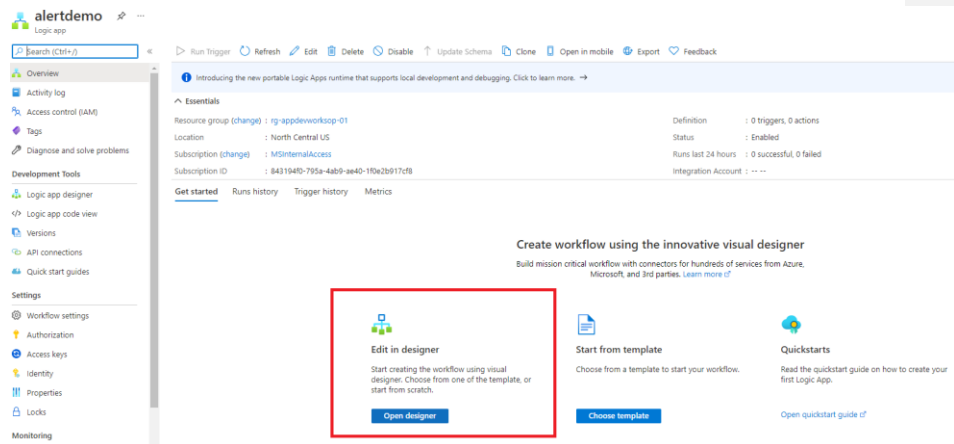
1. Open Azure Portal in a new tab
2. Navigate to **Logic Apps** from the search bar on top



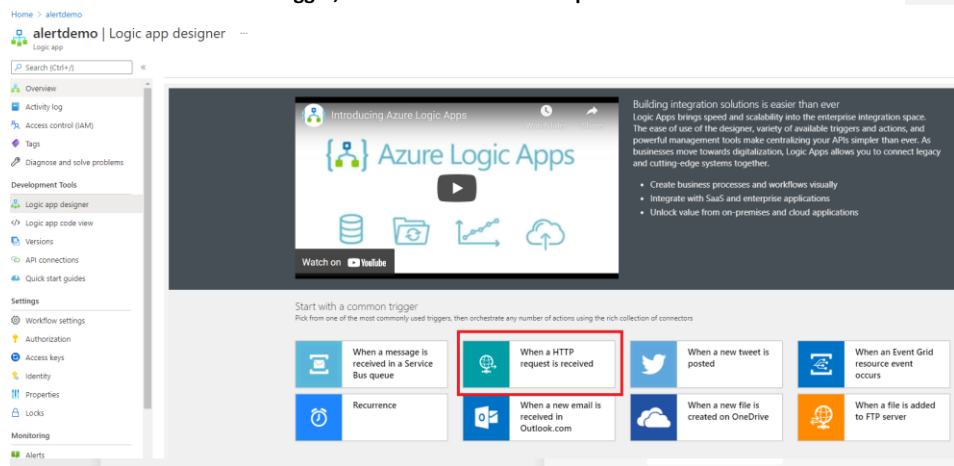
3. Click on + **Add** to create a new logic app



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



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



10. Click on Use sample payload to generate schema

When a HTTP request is received

HTTP POST URL

URL will be generated after save

Request Body JSON Schema

Use sample payload to generate schema

Add new parameter

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

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

```

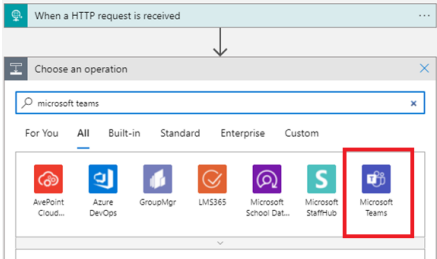
```
        "submissionTimestamp": "2018-04-03T22:33:36.1068742+00:00",
        "resourceType": "microsoft.insights/activityLogAlerts"
    },
    "properties": {}
}
```

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

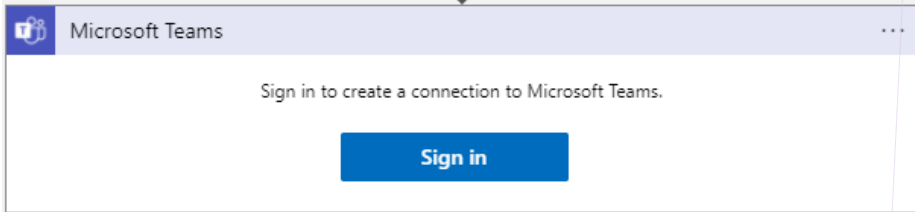
Remember to include a Content-Type header set to application/json in your request.

✓ 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

Commented [MS3]: I think it's good to mention the left pane where they can add the Dynamic content.

Add dynamic content from the used in this flow.

Dynamic content Expression

Search dynamic content

operationName

operationId

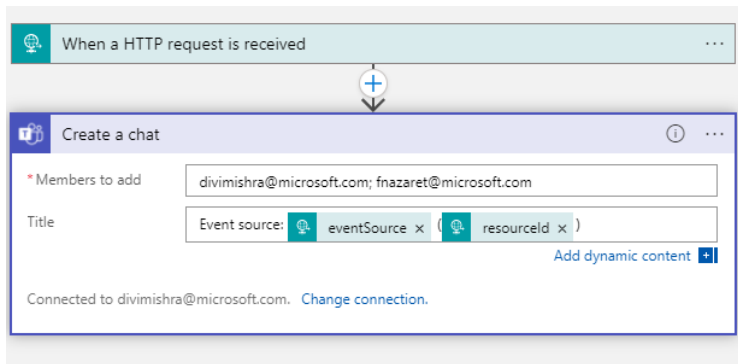
resourceId

resourceGroupName

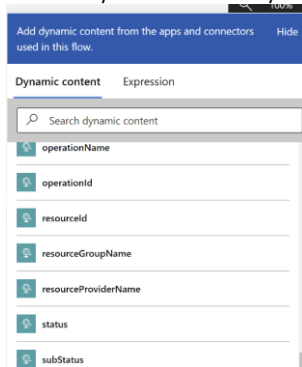
resourceProviderName

status

subStatus



Note that you can also add dynamic content as a part of this action:



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

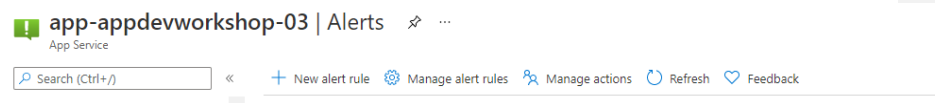
Creating the alert

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

Monitoring



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 ⓘ

Displaying 1 - 1 signals out of total 1 search results

http 4xx			
Signal name	↑↓	Signal type	↑↓
Http 4xx		Metric	Platform

- Set the Alert logic for “greater than 10”
- 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.

- Click on + Create action group
- Under **Basics > instance details**, name your action group something relevant. For example, **dotnetapp_alertgroup**
- Under **Actions**:
 - Action type: **Logic App**
 - Select the logic app you just created
 - Name: **msteamschat**
- Select **Review + Create**
- 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)”.

There are 2 ways to scale your application: manually or automatically. We certainly recommend autoscaling for unexpected spikes in traffic, errors, and to respond to your organization’s monitoring strategy.

2. Manual Scale:

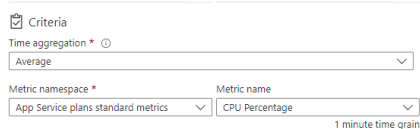
- a. Configure your instance count to your desired instance count.
- b. Select Save.
- c. Review your dashboard to monitor performance.

3. Automatic Scale (Custom Auto-scale):

When scaling automatically: decide if you want to scale based on a metric or scale to a specific instance count (when no other scale conditions are matched). You decide the minimum, maximum and default instances that you’d like to have for your ASP.

Here, we scale based on a metric:

- a. Add a rule to set instance count to 1 if the CPU percentage is less than 10%. Explore through the other standard metrics that you can also leverage.



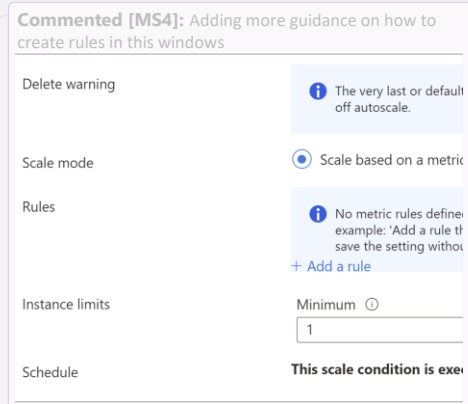
Criteria

Time aggregation * ⓘ
Average

Metric namespace * Metric name
App Service plans standard metrics CPU Percentage

1 minute time grain

You can define the operation to increase/decrease the count of instances according to your scaling rule.



Commented [MS4]: Adding more guidance on how to create rules in this windows

Delete warning ⓘ The very last or default off autoscale.

Scale mode Scale based on a metric

Rules ⓘ No metric rules defined. example: 'Add a rule to save the setting without...' + Add a rule

Instance limits Minimum ⓘ 1

Schedule This scale condition is ex

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	

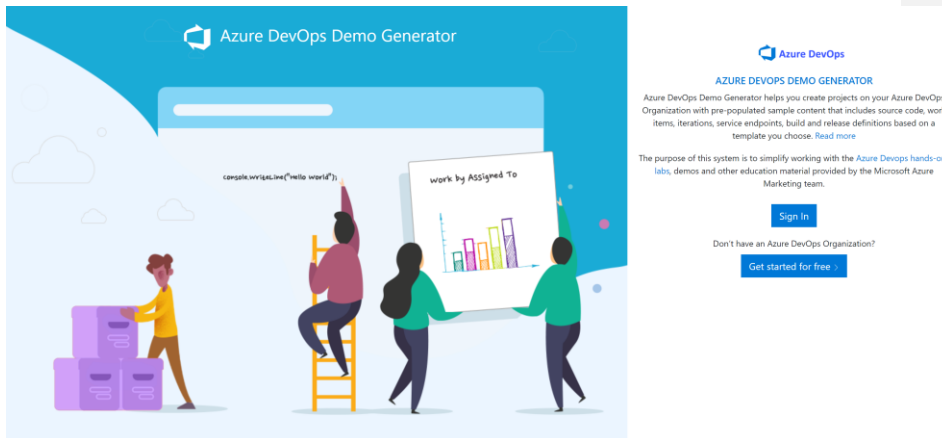
Best practice: for every scale out rule, there must be a scale in rule (and vice versa).

- 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](#)



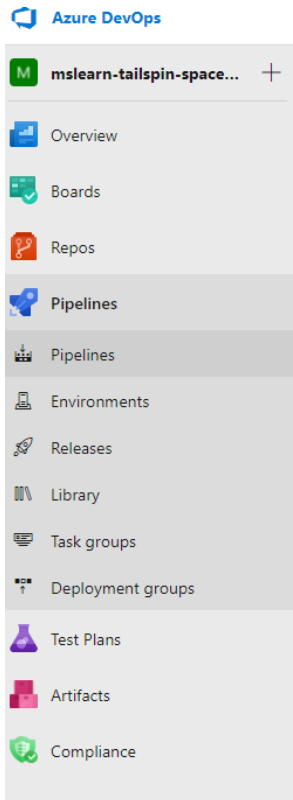
2. Sign in and accept the usage terms.
3. Create a new project with your Azure DevOps (ADO) organization and a project name. For Selected template: under MS learn, chose "Create a build pipeline with Azure Pipelines". Finally, create project.

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

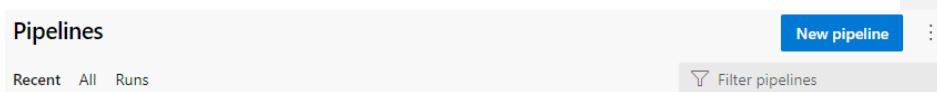
4. Once the deployment is done, select "Navigate to the project"

Create a build pipeline with Azure Pipelines

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



2. Create a new pipeline.



3. Use the classic editor option at the bottom.

Connect


Select


Configure


Review


New pipeline


Where is your code?


 Azure Repos Git YAML
Free private Git repositories, pull requests, and code search

 Bitbucket Cloud YAML
Hosted by Atlassian

 GitHub YAML
Home to the world's largest community of developers


 GitHub Enterprise Server YAML
The self-hosted version of GitHub Enterprise

 Other Git
Any generic Git repository

 Subversion
Centralized version control by Apache

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

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

 We need your authorization to access your repositories

Connection name *

GitHub connection 1

Authorize using OAuth

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

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

Select a source

Azure Repos Git

GitHub

GitHub Enterprise Server

Subversion

Bitbucket Cloud

Other Git

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

Repository * | [Manage on GitHub](#)

divim/AzureAppServices

Default branch for manual and scheduled builds *

main

[Continue](#)

Select “Continue”.

- 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.

- Your pipeline currently looks like this:

- Please ensure to use the “ubuntu-latest” specification for the Azure pipeline agent pool’s specification

Commented [MS5]: Ubuntu-v16 is not exist anymore, better to mention the latest version instead.

Pipeline
Build pipeline

Get sources
divim/AzureAppDevWorkshops main

Unit testing job
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

Agent job ⓘ

View YAML Remove

Display name *
Unit testing job

Agent selection ^

Agent pool ⓘ | Pool information | Manage ↗
Azure Pipelines

Agent Specification *
ubuntu-latest

Demands ⓘ

Name	Condition	Value
------	-----------	-------

+ Add

Execution plan ^

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

Agent job 1
Run on agent

+ ⋮

Add a task to Agent job 1

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

10. Your pipeline will now look like this:

Build the app Run on agent

Use .NET Core sdk 5.x

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 ①

[Link settings](#) [View YAML](#) [Remove](#)

Task version

Display name *

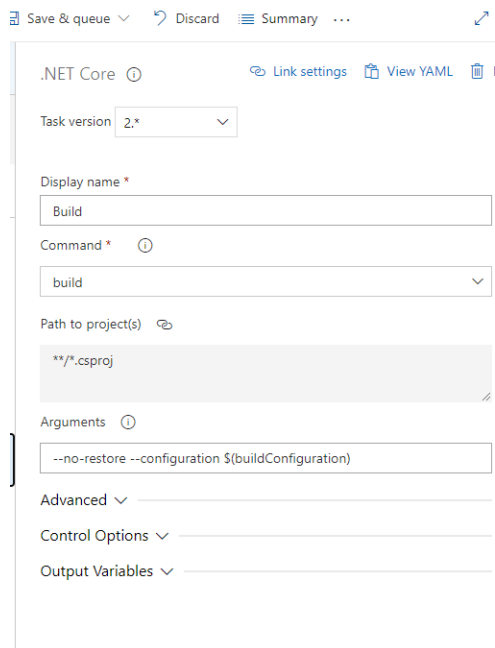
Package to install ①

☐ Use global json ①

Version ①

☐ Include Preview Versions ①

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



The screenshot shows the configuration for a .NET Core build task. The task version is set to 2.x. The display name is 'Build'. The command is 'build'. The path to project(s) is '**/*.csproj'. The arguments are '--no-restore --configuration \$(buildConfiguration)'. The advanced options are collapsed.

Save & queue ▾ Discard Summary ...

.NET Core ⓘ Link settings View YAML |

Task version 2.x ▾

Display name *
Build

Command * ⓘ
build ▾

Path to project(s) ⓘ
**/*.csproj

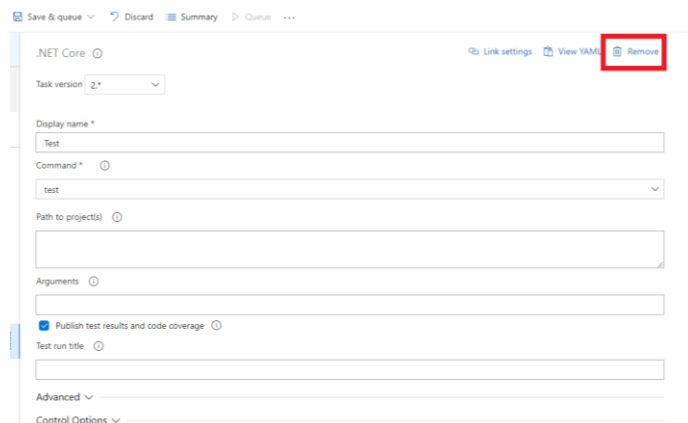
Arguments ⓘ
--no-restore --configuration \$(buildConfiguration)

Advanced ▾

Control Options ▾

Output Variables ▾

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



The screenshot shows the configuration for a .NET Core test task. The task version is set to 2.x. The display name is 'Test'. The command is 'test'. The path to project(s) is empty. The arguments are empty. The checkbox 'Publish test results and code coverage' is checked. The test run title is empty. The advanced options are collapsed. The 'Remove' button is highlighted with a red box.

Save & queue ▾ Discard Summary Queue ...

.NET Core ⓘ Link settings View YAML Remove

Task version 2.x ▾

Display name *
Test

Command * ⓘ
test ▾

Path to project(s) ⓘ

Arguments ⓘ

☒ Publish test results and code coverage ⓘ

Test run title ⓘ

Advanced ▾

Control Options ▾

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 interface for the ".NET Core" task in Azure DevOps. The task version is set to "2.*". The display name is "Publish". The command is set to "publish". The "Publish web projects" checkbox is unchecked. The path to project(s) is set to "**/*.csproj". The arguments field contains the following text: `--no-build --configuration $(buildConfiguration) --output $(Build.ArtifactStagingDirectory)/$(buildConfiguration)`. The "Zip published projects" and "Add project's folder name to publish path" checkboxes are both checked. The interface includes tabs for "Save & queue", "Discard", "Summary", and "View YAML".

Save & queue ▾ Discard Summary ...

.NET Core ⓘ Link settings View YAML R

Task version 2.* ▾

Display name *
Publish

Command * ⓘ
publish ▾

☐ Publish web projects ⓘ

Path to project(s) 📁
**/*.csproj

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

☒ Zip published projects ⓘ

☒ Add project's folder name to publish path ⓘ

Advanced ▾

Control Options ▾

Output Variables ▾

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

Save & queue ▾ ↶ Discard ☰ Summary ⋮ ↗

Publish build artifacts ⓘ 🔗 Link settings 📄 View YAML 🗑️ Re

Task version 1.* ▾

Display name *

Path to publish * ⓘ
 ⋮

Artifact name * ⓘ

Artifact publish location * ⓘ
 ▾

Advanced ▾

Control Options ▾

Output Variables ▾

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

Tasks Variables Triggers Options History | Save & queue ▾ ↶ Discard ☰ Summary ▷ Queue ⋮

Pipeline
Run pipeline ⋮

Get sources
🔗 GitHub Actions (github-actions) 📄 main

Build the app
🔗 Run on agent

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

Agent job ⓘ

View YAML 🗑️ Remove

Display name *

Agent selection ^

Agent pool ⓘ | Manage ⓘ

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

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

Continuous integration

divim/mslearn-tailspin-spacegame-web Enabled

Pull request validation

divim/mslearn-tailspin-spacegame-web Disabled

Scheduled + Add

No builds scheduled

Build completion + Add

Build when another build completes

divim/mslearn-tailspin-spacegame-web

☒ Enable continuous integration

☐ Batch changes while a build is in progress

Branch filters

Type Branch specification

Include main

+ Add

Path filters

+ Add

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

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

Run pipeline

Select parameters below and manually run the pipeline

Save comment

First build pipeline

Agent pool

Azure Pipelines

Agent Specification *

ubuntu-16.04

Branch/tag

main

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

Commit

Advanced options

Variables

3 variables defined

Demands

This pipeline has no defined demands

☐ Enable system diagnostics

Cancel Save and run

20. Observe the tasks running.

You have successfully created your build pipeline.

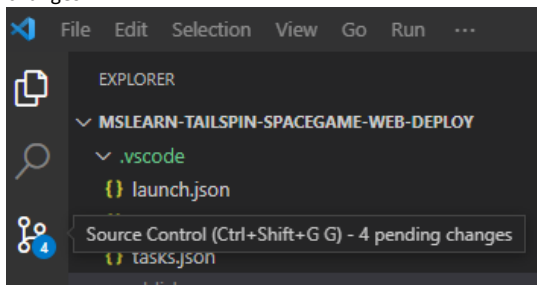
Source Control with GitHub

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

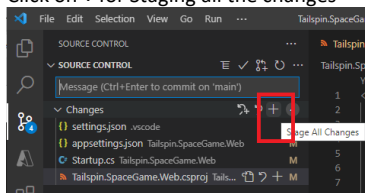
Recall that we had made some changes to the VS Code when we added the Application Insights instrumentation key.

Let's now push it to GitHub.

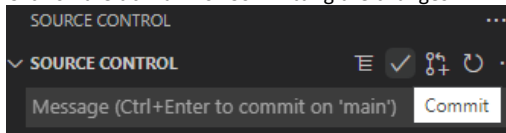
1. Navigate back to your VS Code and notice that your **Source Control** tab highlights pending changes.



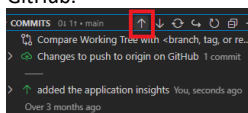
2. Click on the **Source Control** tab
3. Click on **+** for Staging all the changes



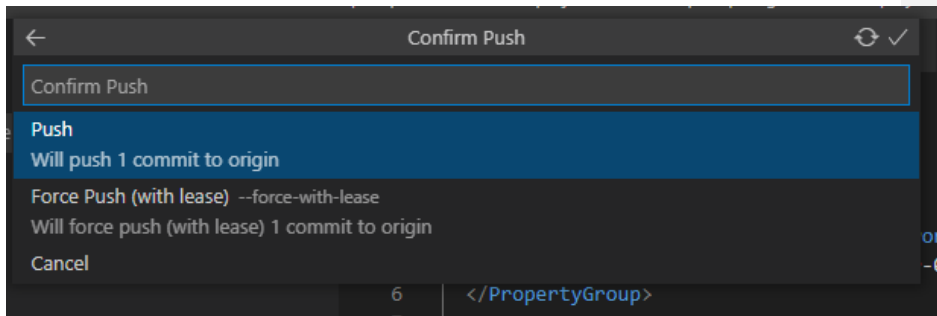
4. Click on the tic mark for Committing the changes



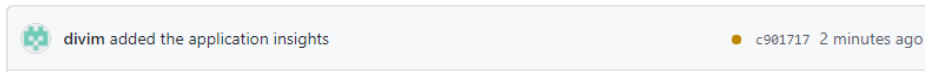
5. Enter a relevant message, such as "added app insights instrumentation key"
6. Under **COMMITTS** (at the bottom), click on the UP arrow. This will **Push** all the changes to GitHub.



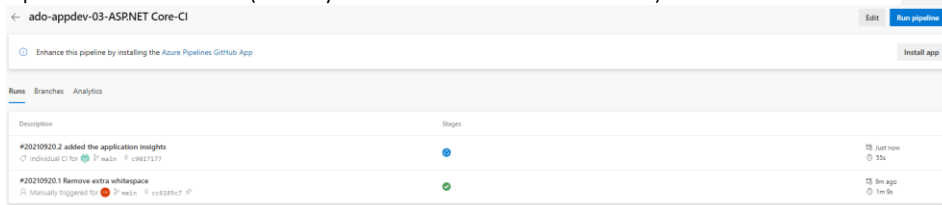
7. Confirm the Push.



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



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



Note:

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

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

Lab 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\Azure App Innovation Workshops\Personal\AzureAppServices\Tailspin.SpaceGame.Web.Tests\bin\Debug\net5.0\Tailspin.SpaceGame.Web.Tests.dll
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

- On your terminal, run the following commands:

- Return to the parent directory of **Azure App Service** using the `cd` command
- Create a local tool manifest file:
`$ dotnet new tool-manifest`
- Install **ReportGenerator**:
`$ dotnet tool install dotnet-reportgenerator-globaltool`

Warning: You may receive an error `System.UnauthorizedAccessException` error. You can ignore that and proceed with the next steps

- Add the `coverlet.msbuild` package to the project (please ensure that you're at the 'Azure App Service' directory for this):

`$ dotnet add Tailspin.SpaceGame.Web.Tests package coverlet.msbuild`

- Run the command for unit test and code coverage (the `/p:` tells coverlet what format to use and where to place the results):

`$ dotnet test Tailspin.SpaceGame.Web.Tests/Tailspin.SpaceGame.Web.Tests.csproj --configuration Release /p:CollectCoverage=true /p:CoverletOutputFormat=cobertura /p:CoverletOutput=../TestResults/Coverage/`

Commented [MS6]: This command gives the following error:

```

PS C:\Users\msoultan\source\repos\AzureAppDevWorkshops\Azure App Service\Tailspin.SpaceGame.Web.Tests> dotnet add package coverlet.msbuild

```

It should be replaced by
`dotnet add package coverlet.msbuild`

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: 302 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% | 10.91%
Tailspin.SpaceGame.Web.Views | 0% | 0% | 0%

Total | 7.77% | 0% | 14.58%
Average | 5.72% | 0% | 9.45%

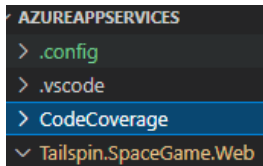
```

- 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`

Commented [MS7]: Wasn't able to execute this command, as it depends on the previous one

- 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 - 3:35:53 PM
Parser:	CoberturaParser
Assemblies:	2
Classes:	16
Files:	16
Covered lines:	15
Uncovered lines:	178
Coverable lines:	193
Total lines:	890
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

Source #1: Report #1

By severity

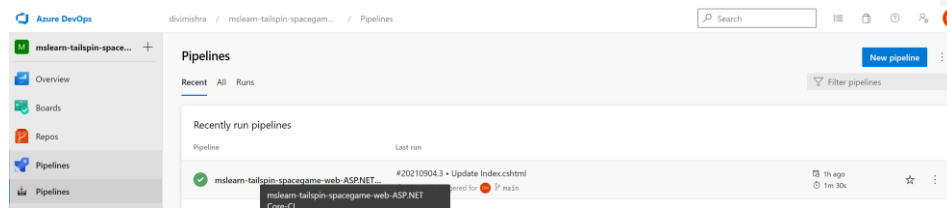
Grouping

Name	Covered	UI
Tailspin.SpaceGame.Web	15	
Tailspin.SpaceGame.Web\Controllers/HomeController	0	
Tailspin.SpaceGame.Web\LocalDocuments\HBBandwidthTx	10	

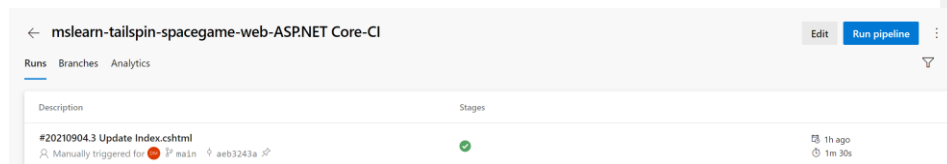
4. Commit your changes to your GitHub repository.

Add tests to Azure Pipeline

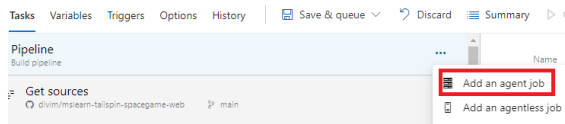
1. Open your Pipeline on dev.azure.com again.



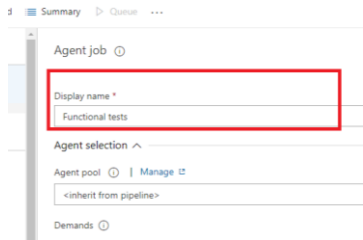
2. Click on **Edit** from the top right to edit the pipeline.



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

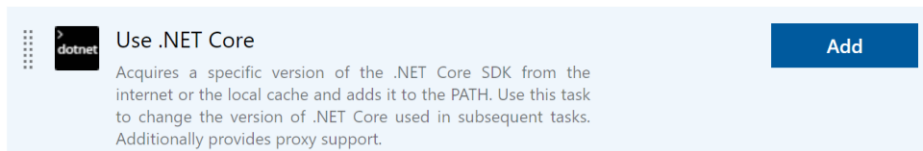


4. Click on **Agent job** and rename the job to **Test the app**. Leave the rest as default.

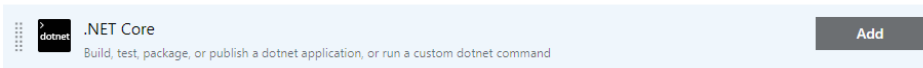


5. Click on the **+** to add a new task to the Functional tests job
6. Add the following jobs:
 - a. Use .NET Core

Add tasks | Refresh



- b. .NET Core : **Do this 2 times!**



- c. Publish code coverage results



7. Click on the **Use .NET Core** task
 - a. Version: **5.x**
8. Click on the **.NET Core #1** task
 - a. Display Name: **Restore**
 - b. Command: **restore**
 - c. Path to project: ****/*.csproj**

.NET Core 🔗 Link settings 📄 View YAML 🗑 Remove

Task version 2.*

Display name *

Command *

Path to project(s) 🔗

Arguments 🔗

Feeds and authentication ^

Feeds to use *

☒ Feeds I select here ☐ Feeds in my NuGet.config

Use packages from this Azure Artifacts feed 🔗

9. Click on the next **.Net Core** task:

- a. Display name: **Test the app**
- b. Command: **test**
- c. Path to project: ****/*.Tests.csproj**
- d. Arguments: **--configuration \$(buildConfiguration) /p:CollectCoverage=true /p:CoverletOutputFormat=cobertura /p:CoverletOutput=\$(Build.SourcesDirectory)/TestResults/Coverage/**
- e. **Select** the Publish test results and code coverage option

.NET Core 🔗 Link settings 📄 Vi

Task version 2.*

Display name *

Command *

Path to project(s) 🔗

Arguments 🔗

☒ Publish test results and code coverage 🔗

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

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

Publish code coverage results ⓘ

Task version 1.* ▼

Display name *

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

Code coverage tool * ⓘ

Cobertura

Summary file * ⓘ

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

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

Commented [MS8]: The testing task gives the following error

```
The argument /home/vsts/work/1/s/Azure App Service/Tailspin.SpaceGame.Web.Tests/bin/Release/net5.0/SpaceGame.Web.Tests.dll is not a valid file path.
##[error]Error: The process 'dotnet/dotnet' failed with exit code 1
##[warning]No test result files were found.
##[warning].NET 5 has some compatibility issues with older NuGet versions(<=5.7), so it is recommended to update to the latest version.
Info: Azure Pipelines hosted agents have been updated and now contain .Net 5.x SDK/Runtime
##[error]dotnet command failed with non-zero exit code on the following projects : /home/vsts/work/1/s/Azure App Service/Tailspin.SpaceGame.Web.Tests
Finishing: Test the app
```

✓ #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)

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

Summary

1 Run(s) Completed (1 Passed, 0 Failed)

9
Total tests
+9



9 ● Passed
0 ● Failed
0 ● Others

100%
Pass percentage
↑ 100%

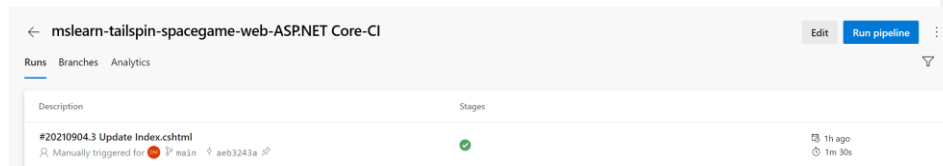
2m 7s
Run duration ⓘ
↑ +2m 7s

0
Tests not reported

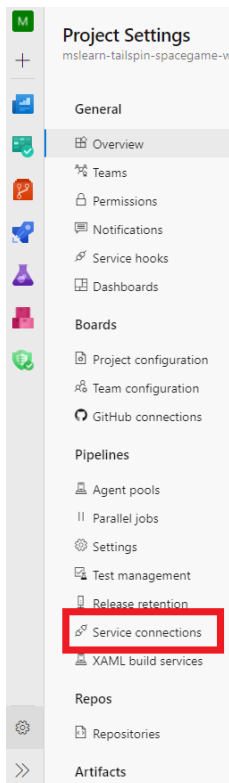
Lab 105: Zero-downtime app deployment with release pipeline

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.



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



4. Click on **New Service Connection**

Service connections

New service connection

Filter by keywords
Created by

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

Description (optional)

Security

☒ Grant access permission to all pipelines

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

Back
Save

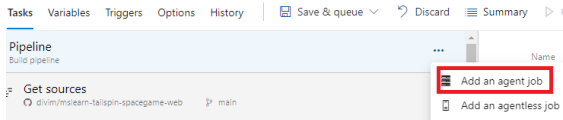
- Select **Save**.
- Navigate back to your pipeline and select **Edit**.

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

Edit
Run pipeline

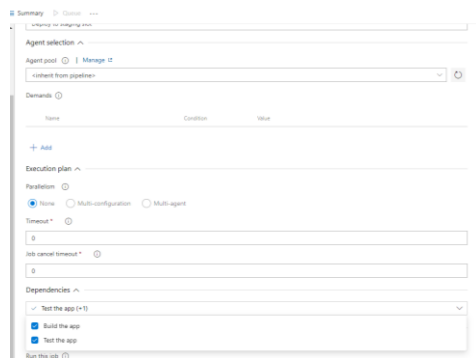
Runs
Branches
Analytics

- 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:

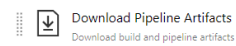


14. Add the following agent jobs:

- Download pipeline artifact

Add tasks | Refresh

download pipeline X



Download Pipeline Artifacts
Download build and pipeline artifacts

- Azure App Service Deploy

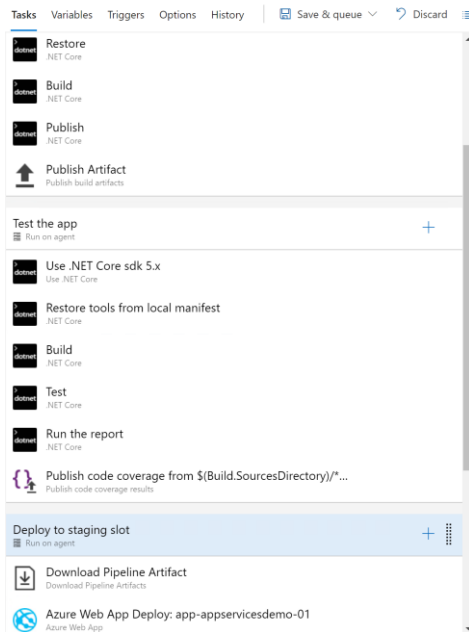
Add tasks | Refresh

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, Nodejs, PHP, Python, or Ruby

15. Ensure that your pipeline now looks like this:



16. Click on the **Download Pipeline Artifact** task to edit the task as follows:
 - a. Destination directory: \$(Pipeline.Workspace)
 - b. Leave the Artifact name and matching patterns empty

Summary Queue ...

Download Pipeline Artifacts ⓘ

Task version

Display name *

Download artifacts produced by * ⓘ

☒ Current run ☐ Specific run

Artifact name ⓘ





Matching patterns ⓘ


Destination directory * ⓘ

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   Link settings  View YAML  Remove




Task version  4*


Display name *


Azure App Service Deploy: azqatar-mslearn-tailspin-spacegame-web

Connection type *

Azure Resource Manager

Azure subscription *  |  Manage 


mslearn-tailspin-spacegame-rg 


 Scoped to resource group mslearn-tailspin-spacegame-web

App Service type *


Web App on Windows

App Service name *


azqatar-mslearn-tailspin-spacegame-web 


☒ Deploy to Slot or App Service Environment 

Resource group *

mslearn-tailspin-spacegame-web 

Slot *

staging 

Virtual application 

18. Select **Save & Queue**.




19. Select **Save & Run**.

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








21. Click on the **staging slot**


22. Click on **Browse**


Home > azqatar-mslearn-tailspin-spacegame-web >

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

App Service (Slot)

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

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

^ Essentials  JSON View

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

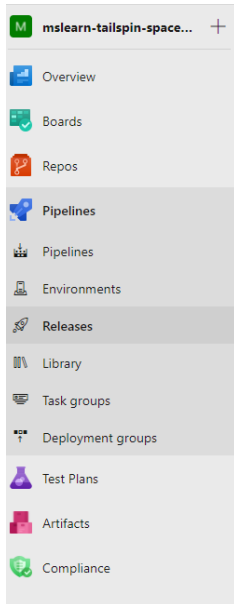
23. Your app has been deployed successfully to the staging slot. Note that the name of the URL is simply <app-service-name>-staging.azurewebsites.net



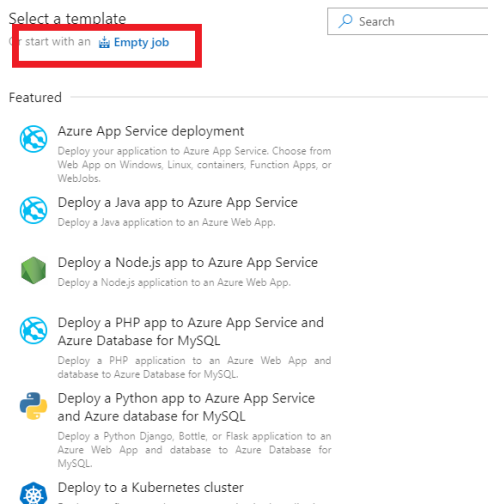
24. Simply remove the “-staging” from the URL to view the production slot website.

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

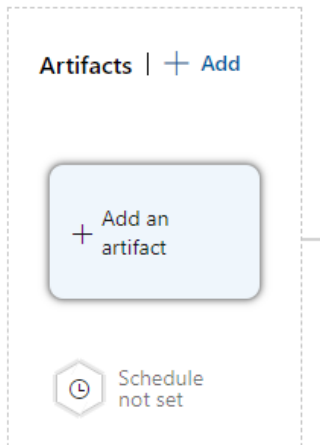
1. Navigate to your Azure DevOps project > **Pipelines > Releases**



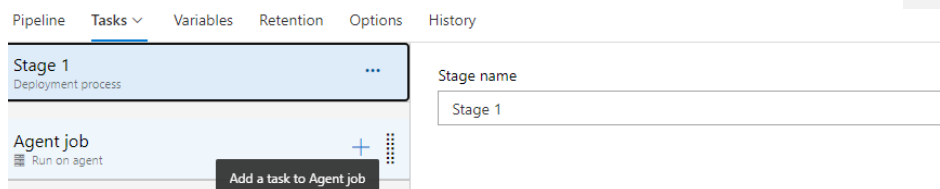
2. Select “New pipeline”.
3. For the template, start with an empty job



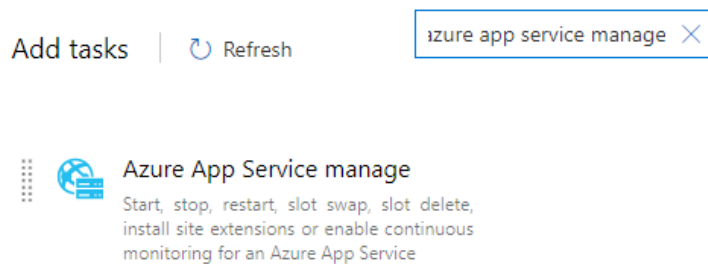
- Click on **Add an artifact**



- 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' includes '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 named 'Azure App Service manage' with version '0.*'. The configuration fields are as follows:

- Display name ***: Swap Slots
- Azure subscription ***: A dropdown menu with a red border and a warning icon. Below it, a message says 'This setting is required.'
- Action**: Swap Slots
- App Service name ***: A dropdown menu with a red border and a warning icon. Below it, a message says 'This setting is required.'
- Resource group ***: A dropdown menu with a red border and a warning icon. Below it, a message says 'This setting is required.'
- Source Slot ***: A dropdown menu with a red border and a warning icon. Below it, a message says 'This setting is required.'

At the bottom, there is a checkbox labeled '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 0.0

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

Azure subscription *
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

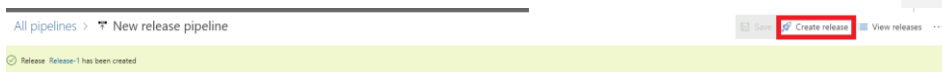
Slot *
staging

Control Options

Output Variables

15. Select **Save**.

16. Select **Create release**.



17. Observe the tasks on the release pipeline.

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

Pipeline Tasks Variables Logs Tests Deploy Cancel Refresh Download all logs

Deployment attempt #2
Succeeded

Agent job
Succeeded

Agent job
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	Running	ASP-mslearn-tailspinspacegame-web-9893	100
PRODUCTION			

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

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.

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 logical isolation of the Azure cloud dedicated to your subscription. You can connect virtual networks to other virtual networks, 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="text" value=""/>	(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	<input type="text" value="10.2.1.0/24"/> ✓	<input type="text" value="10.2.1.0 - 10.2.1.255 (256 addresses)"/> ✓
<input type="text" value=""/>	<input type="text" value=""/>	(0 Addresses)

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

✓ Basics **2 Frontends** 3 Backends 4 Configuration 5 Tags 6 Review + create

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 (New) pip-agw 
[Add new](#)

6. Backends tab:

- Add a backend pool
- Name: **be-pool-demo**
- Add backend pool without targets: **Yes**
- Add** the backend pool

7. Configuration tab:

- Select **Add a routing rule**
- Rule name: rr-demo
- Listener:
 - Listener name: **listener-demo**
 - Frontend IP: **Public**
 - Protocol: **HTTP**
 - Leave the rest as defaults
- Backend targets:
 - Target type: **Backend pool**
 - Backend target: **be-pool-demo**
 - HTTP Settings:
 - Add new**
 - HTTP Settings name: **HTTP-rule**
 - Leave the rest as default

Add a HTTP setting

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


HTTP settings name *	HTTP-rule
Backend protocol	<input checked="" type="radio"/> HTTP <input type="radio"/> HTTPS
Backend port *	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) *	20
Override backend path: ⓘ	
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 type="radio"/> Yes <input checked="" type="radio"/> No
Host name override	<input checked="" type="radio"/> Pick host name from backend target
	<input 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

✓ Basics ✓ Frontends ✓ Backends **4 Configuration** 5 Tags 6 Review + create

Create routing rules that link your frontend(s) and backend(s). You can also add more backend pools, add a second frontend IP configuration if you have already, or edit previous configurations.


Frontends
+ Add a frontend IP
Public: (new) pip-agw


Routing rules
+ Add a routing rule
rr-demo
[Manage HTTP settings](#)



Backend pools
+ Add a backend pool
be-pool-demo

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**.

Home > waf-demo-01

 **waf-demo-01** | Backend pools ...

Application gateway

Search (Ctrl+F)

Diagnose and solve problems

Settings

- Configuration
- Web application firewall
- Backend pools**

+ Add Refresh

Search backend pools

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

Backend protocol
☒ HTTP ☐ HTTPS

Backend port *

Additional settings

Cookie-based affinity ☐ Enable ☒ Disable

Connection draining ☐ Enable ☒ Disable

Request time-out (seconds) * ☐

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 priority order.

app-appservicesdemo-01.azurewebsites.net app-appservicesdemo-01.scm.azurewebsites.net

+ Add rule

Priority	Name	Source
1	Allow all	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

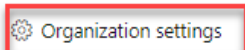
The web app you have attempted to reach has blocked your access.

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

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:
 - i. **Full access**
 - ii. **(OR)** Agent pools: Read & Manage; Auditing: Read Audit Log

Create a new personal access token

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

Name

Selenium Agent

Organization

divimishra

Expiration (UTC)

90 days

1/1/2022

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 updated 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:

Commented [MS9]: The deployment gives error Resource

✓	vm selenium/SqllaaSExtension
!	vm selenium/CustomScriptExtension
✓	vm selenium
✓	vm selenium
✓	vm seleniumdyvk2wqikgizy
✓	vm seleniumPip
✓	vm seleniumVnet
✓	vm seleniummsg

Resources Recommendations (10)

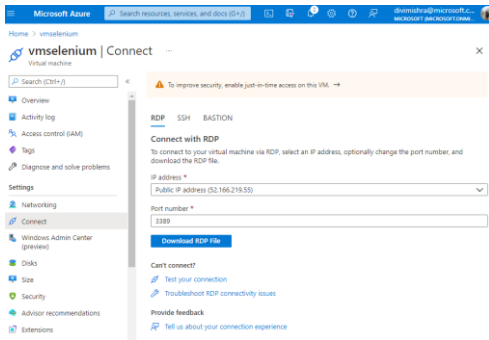
Filter for any field... Type == all Location == all Add filter

Showing 1 to 30 of 30 records ☐ Show hidden types No grouping

List view

Name	Type	Location
vm-selenium	Network interface	West Europe
vm-selenium-testing	Network interface	West Europe
vm-selenium-testingsg	Network security group	West Europe
vm-selenium-testingpip	Public IP address	West Europe
vm-selenium-testingvnet	Virtual network	West Europe
vm-seleniumsg	Network security group	West Europe
vm-seleniumpip	Public IP address	West Europe
vm-seleniumvnet	Virtual network	West Europe
vm-selenium	Disk	West Europe
vm-selenium	Virtual machine	West Europe
vm-selenium	Network interface	West Europe

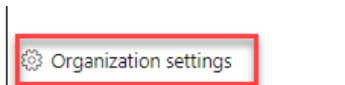
3. Select **vm-selenium** virtual machine.
4. Download the RDP File for connecting to the VM.



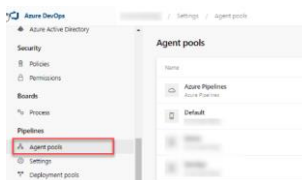
5. Connect with the following credentials:
 - a. Username: vmadmin
 - b. Password: P2ssw0rd@123



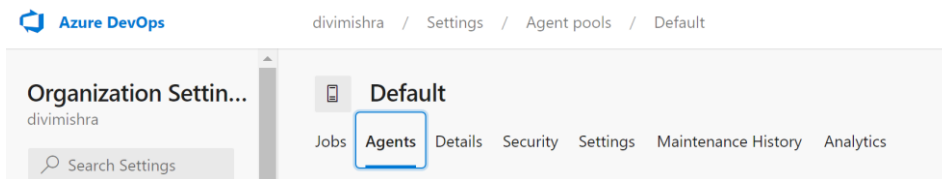
6. In the VM, open any web browser and sign in to your Azure DevOps organization.
7. Click on **Organization Settings** at the bottom left.



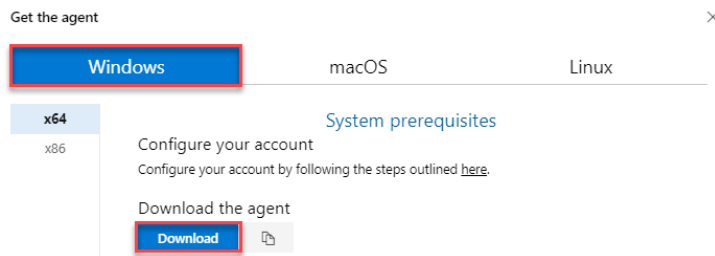
8. Select **Agent pools** under Pipelines



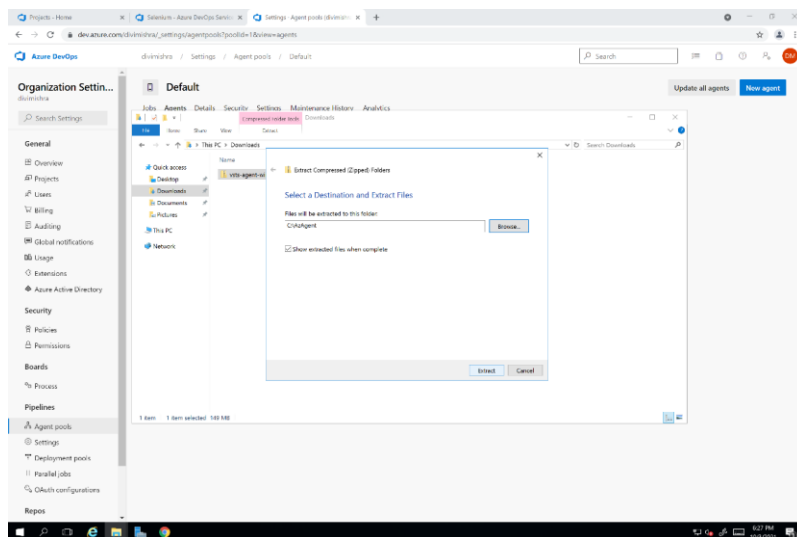
9. Select **Default** pool and select **Agents** tab. Click on **New agent**.



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

AZURE PIPELINES
agent v2.166.4 (commit efdfb40)

>> Connect:

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

>> Register Agent:

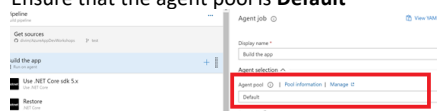
Enter agent pool (press enter for default) >
Enter agent name (press enter for selenium) >
Scanning for tool capabilities.
Connecting to the server.
Successfully added the agent
Testing agent connection.
Enter work folder (press enter for _work) >
2020-05-08 08:24:14Z: Settings Saved.
Enter run agent 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	https://<your-app-service-name>-01.azurewebsites.net
CHROMEWEBDRIVER	C:\SeleniumWebDrivers\ChromeDriver
EDGEWEBDRIVER	C:\SeleniumWebDrivers\EdgeDriver
GECKOWEBDRIVER	C:\SeleniumWebDrivers\GeckoDriver

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



c. Add the following tasks:

i. Use .NET Core

1. Specify the SDK as 5.x

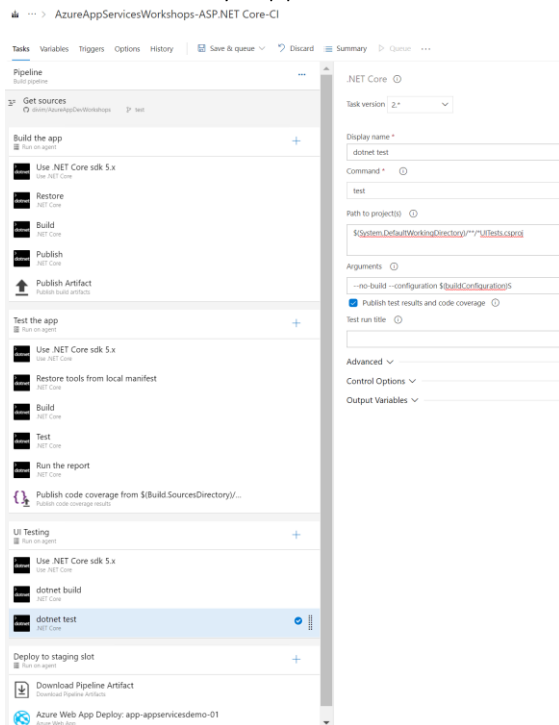
ii. Dotnet Build

1. Path to project:
`$(System.DefaultWorkingDirectory)/**/*.UITests.csproj`

iii. Dotnet Test

1. Path to project:
`$(System.DefaultWorkingDirectory)/**/*.UITests.csproj`
2. Arguments: `--no-build --configuration $(buildConfiguration)`
3. 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 \(azuredevopslabs.com\)](#)

