**Exercise - Deploy a Docker container to Azure**

Your project came with a release pipeline that builds the projects in the solution and deploys the web app to its App Service. Now it's time to update that pipeline build and deploy the project as a container instead.

In this part, you'll:

* Define some pipeline variables to make the build pipeline easier to maintain.
* Replace the existing **Build** tasks with a unified task to build and push a Docker container.
* Replace the existing **Deploy** task with one that updates the App Service Web App with the new container image.
* Save the pipeline to trigger a CI/CD workflow.

**Define variables to be shared within the pipeline**

Here you add a new pipeline variable to the existing CI/CD pipeline defined in *azure-pipelines.yml*.

1. From Azure DevOps, navigate to **Pipelines**.
2. Select the pipeline.
3. Select **Edit**. This will bring up the *azure-pipelines.yml* file that defines the existing CI/CD pipeline.

**Andy:** This was our previous build stage. I haven't made any changes yet, so we'll need to completely update it to build and push the container. We added some variables to the shared library that will make it easier to update the pipeline if we change our deployment settings. However, we have some other strings that will be reused across tasks, so we should set them as pipeline variables within the file itself.

1. Add the highlighted line below to add a pipeline variables named webRepository and tag. These will be used in multiple tasks to uniquely identify the specific version of the container being referenced. You may also remove the buildConfiguration variable as it won't be needed anymore.

**yml**

trigger:

- '\*'

variables:

buildConfiguration: 'Release'

webRepository: 'web'

tag: '$(Build.BuildId)'

**Replace the build stage tasks**

**Andy:** I don't think we need any of these build tasks anymore since the Dockerfile in the project folder already defines the build we want. However, I haven't had the chance to see what we can use to build the image using a Dockerfile yet. Any ideas?

**Mara:** I was just looking that up. It seems like should be able to build the container and even push it to the repository with a single task. Let's add it now.

**Docker task**

The Docker@2 task is designed to build and deploy Docker containers. It's ideal for this scenario because it accomplishes everything in a single task:

* command indicates the feature to run. In this case, buildAndPush does exactly what it sounds like.
* buildContext specifies the path to the build context.
* repository specifies the name of the repository.
* dockerfile specifies the path to the Dockerfile.
* containerRegistry specifies the name of the container registry connection to use.
* tags indicates which tags to apply to the container image.

You can learn more about the flexibility of this task in the [Docker task](https://docs.microsoft.com/en-us/azure/devops/pipelines/tasks/build/docker" \t "az-portal) documentation.

Replace the entire **Build** stage with the code below.

**yml**

- stage: 'Build'

displayName: 'Build and push'

jobs:

- job: 'Build'

displayName: 'Build job'

pool:

vmImage: 'ubuntu-18.04'

steps:

- task: Docker@2

displayName: 'Build and push the image to container registry'

inputs:

command: buildAndPush

buildContext: $(Build.Repository.LocalPath)

repository: $(webRepository)

dockerfile: '$(Build.SourcesDirectory)/Tailspin.SpaceGame.Web/Dockerfile'

containerRegistry: 'Container Registry Connection'

tags: |

$(tag)

**Replace the deploy stage task**

**Andy:** Well that seems pretty straightforward. Now all we need to find is a task that will instruct App Service to use the newly pushed version of the container image.

**Mara:** I'm already on it. It's a bit different from deploying a build local to the pipeline, but still direct enough that we can get the job done in one task. Let's add it now.

**Azure Web App for Container task**

The AzureWebAppContainer@1 task is designed to deploy Docker containers to App Service. Deployment can be accomplished with a single task:

* appName specifies the name of the App Service instance.
* azureSubscription specifies the name of the Azure connection.
* imageName specifies the full name of the Docker container image.

You can learn more about the flexibility of this task in the [Azure Web App for Container task](https://docs.microsoft.com/en-us/azure/devops/pipelines/tasks/deploy/azure-rm-web-app-containers) documentation.

Replace the entire **Deploy** stage with the code below.

**yml**

- stage: 'Deploy'

displayName: 'Deploy the container'

dependsOn: Build

jobs:

- job: 'Deploy'

displayName: 'Deploy job'

pool:

vmImage: 'ubuntu-18.04'

variables:

- group: Release

steps:

- task: AzureWebAppContainer@1

inputs:

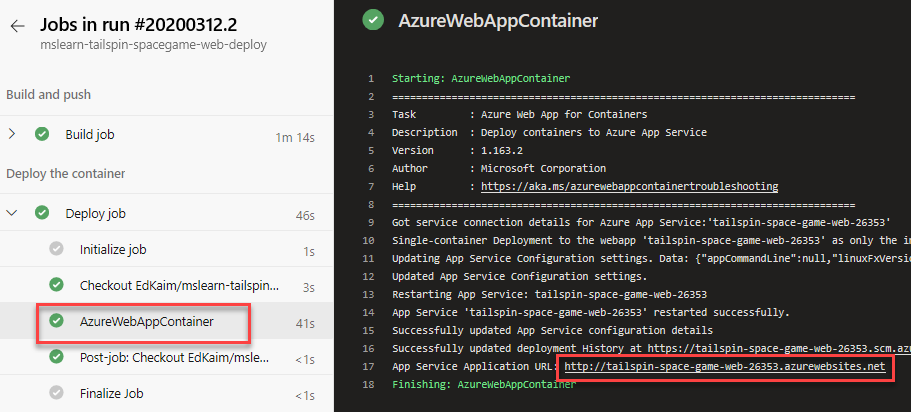
appName: $(WebAppName)

azureSubscription: 'Resource Manager - Tailspin - Space Game'

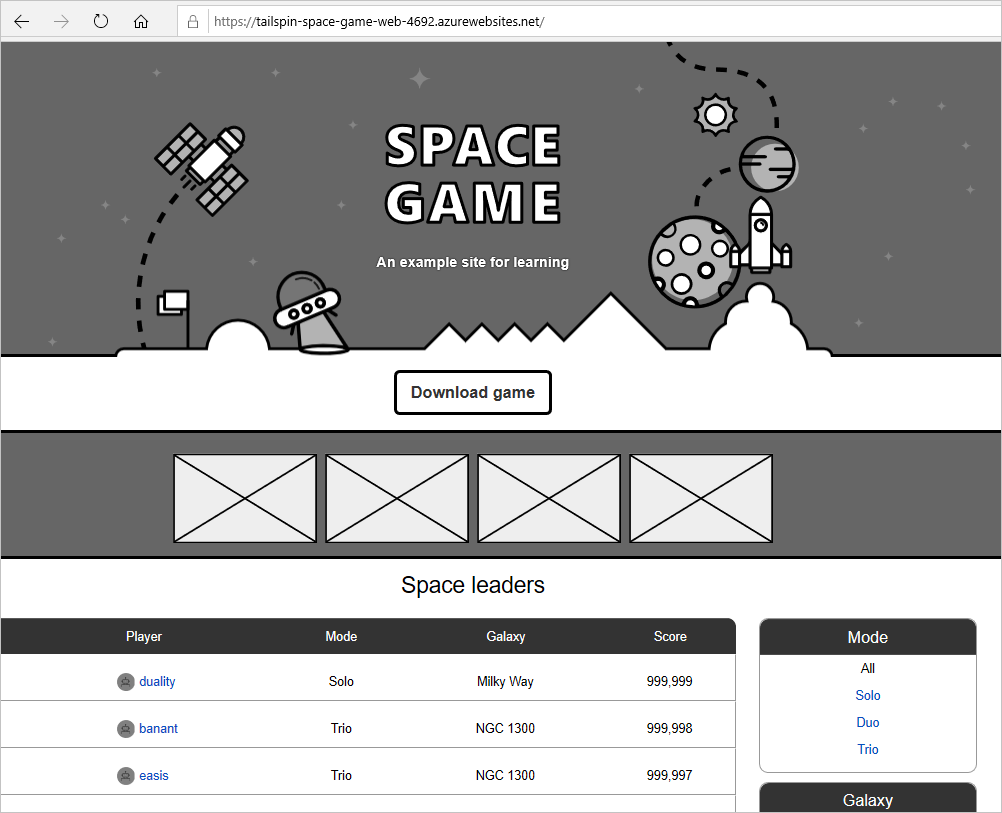
imageName: $(RegistryName)/$(webRepository):$(build.buildId)

**Save the pipeline to trigger a build and release**

1. Select **Save** from the top right corner of the page. Confirm the **Save** to trigger a run.
2. In Azure Pipelines, go to the build. Trace the build as it runs.
3. After the build has succeeded, select the web app container's deploy task and select the URL to view the deployed site.



1. You see the site running on App Service.



**Andy:** This turned out great! I think adopting containers could be a huge win for everyone.