Deploying Docker Container Apps to Azure App Services

# Overview

Microsoft offers great support for building Docker container apps in Visual Studio that can be easily deployed and managed in Azure. In this Quick Start you will create an ASP.NET Core app hosted in a Docker container that you then deploy to Azure App Services.

# Objectives

* Create an ASP.NET Core web application.
* Test the application locally in Docker.
* Deploy the container application to Azure App Services.

# Prerequisites

* Windows 10
* Visual Studio 2017
* Docker for Windows
* Azure subscription (Sign up for a free account **https://azure.microsoft.com/free** Needs a Microsoft account)

# Intended Audience

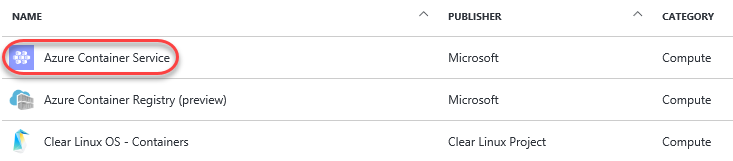
This Quick Start Challenge is intended for developers who are familiar with ASP.NET, C#, and Visual Studio. It does not require prior experience with Docker.

# Task 1: Create an Azure Container Service

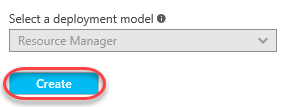
1. From the **Start Menu**, launch **Docker for Windows**. This takes a little time to get going, so kick it off now while you continue to the next step.
2. Open a browser window and log in to **http://portal.azure.com**.
3. Click **+New** and search for **“container”**.

C:\Users\Ed\AppData\Local\Temp\SNAGHTMLdd64375.PNG

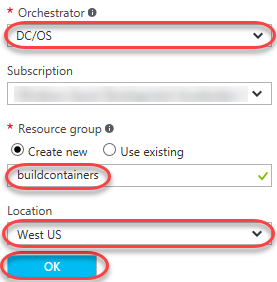
1. Click **Azure Container Service** in the search results.



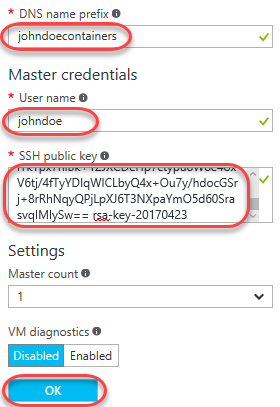
1. Click **Create** to begin the creation process.



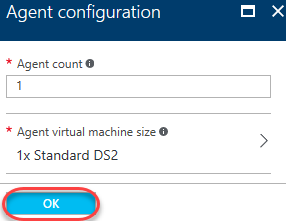
1. Select **DC/OS** as the **Orchestrator**, enter a **Resource group** name of **“buildcontainers”**, and select the **Location** of **West US**. Click **OK** to continue.



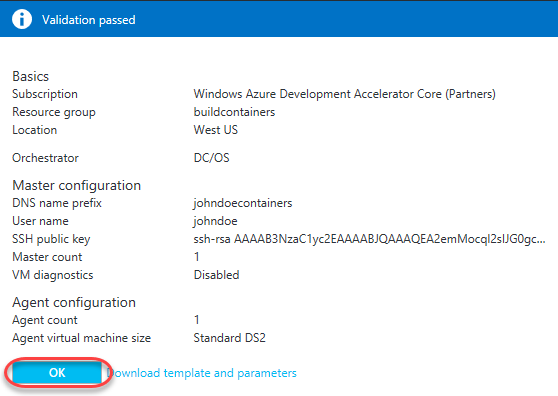
1. Enter a globally unique name for **DNS name prefix**. You may want to incorporate your name, such as **“johndoecontainers”**.
2. Enter any valid **User name**.
3. Paste the following **SSH public key**. You can alternatively generate your own using a tool like **PuTTY**. (Please note this may take a few minutes to complete)
   1. ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEA2emMocql2sIJG0gc8XC4aqGmNaQGyb27pYY0mhWhKAR7WujBu4qgNNKKH/VfeI8Qoa9RPBUIvqn5oCFONKPKyRT0lPN00MWInXeNsBqgHutq7gt8dpiVZpDFJLeBevydEccuzSkCu2HcI84ELMyXl2dWtLu06uJHEENk6LjxHX44enaT6MbpKRCijE+lAc1lzH2CCvh/Ol4C+23+z3HW1ghZMEaY00WfIKbhSMHiYkTpx7hlBk+1ZJXCDeHp7ctypu8W8e4oxV6tj/4fTyYDIqWICLbyQ4x+Ou7y/hdocGSrj+8rRhNqyQPjLpXJ6T3NXpaYmO5d60SrasvqIMIySw== rsa-key-20170423
4. Click **OK**.



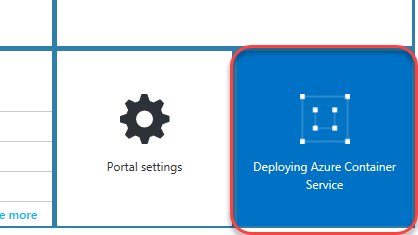
1. Click **OK** for the default **Agent configuration**.



1. Once validated, click **OK** on the summary page to start creation. This may take a while.

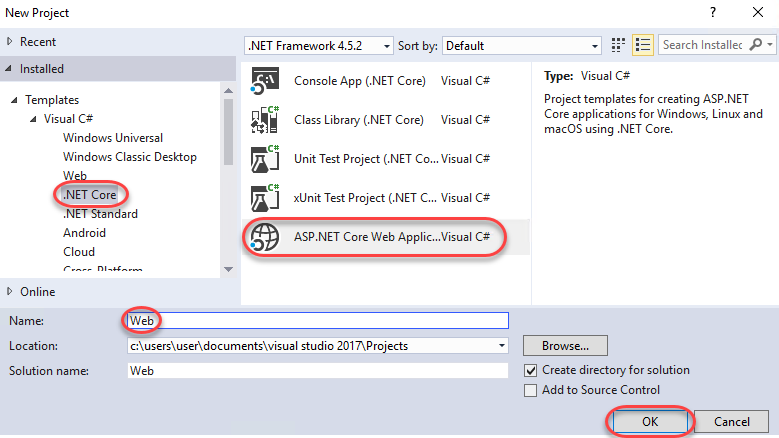


1. The deployment will take a few minutes, so leave the Azure portal window open to the dashboard and move on to the next task.

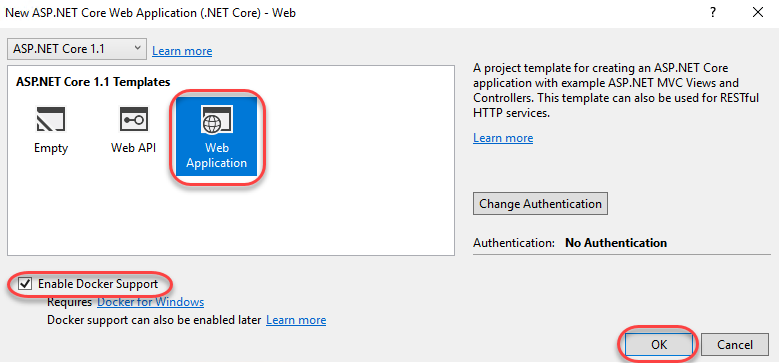


# Task 2: Create an ASP.NET Core application with Docker support

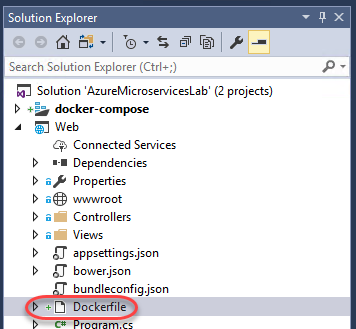
1. Launch **Visual Studio**.
2. From the main menu, select **File | New | Project**.
3. From the **Visual C# | .NET Core** category, select the **ASP.NET Core Web Application** template. Enter a **Name** of **“Web”** and click **OK**.



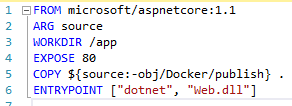
1. The first microservice will provide the web front-end to the application. Select the **Web Application** template and make sure **Enable Docker Support** is selected to ensure we’re ready to deploy to Docker from the start. Click **OK** to create.



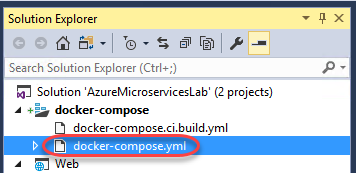
1. Now let’s take a look at some of the Docker files added to the solution. Double-click the **Dockerfile** added to the **Web** project to open it.



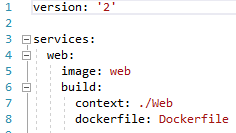
1. **Dockerfile** describes the application, including the base container, the port number to expose the application on, the entry point of the application, and more. You can learn more about this format [here](https://docs.docker.com/engine/reference/builder).



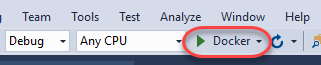
1. You’ll also notice that the Docker support added a set of solution files contained in a virtual **docker-compose** folder. Double-click **docker-compose.yml** to open it.



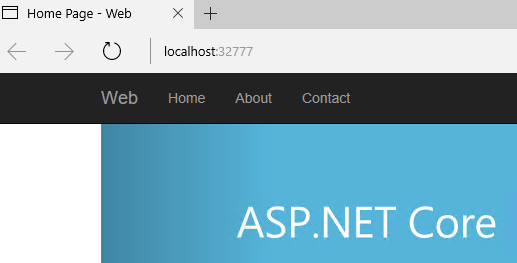
1. **docker-compose.yml** describes how the application should be composed of the required services to set up a given environment. Right now there’s just one service for the **Web** project we just created.



1. Also notice that the traditional **Debug** button has become a **Docker** button. Click it to build the project and deploy it to a container for testing.



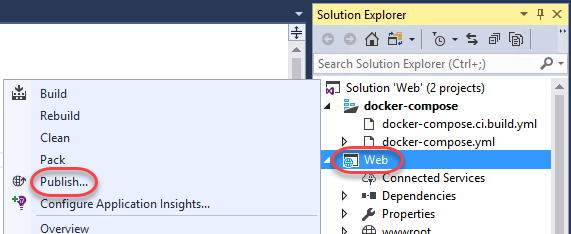
1. The site is just the default project right now. However, it’s important to recognize that it’s running from a Docker container, not local IIS or a simulated host. Close the browser window.



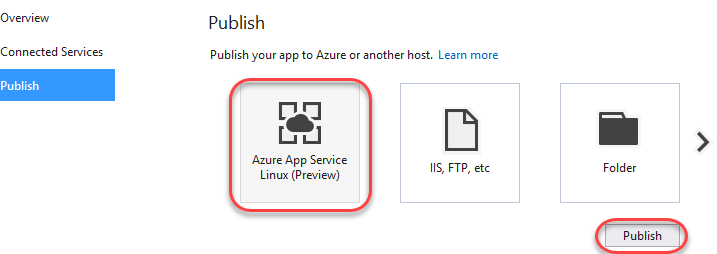
1. Return to **Visual Studio** and select **Debug | Stop Debugging**.

# Task 3: Deploy to Azure App Services

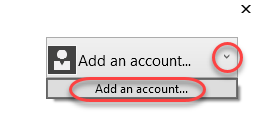
1. Return to **Visual Studio**.
2. From **Solution Explorer**, right-click **Web** and select **Publish**.



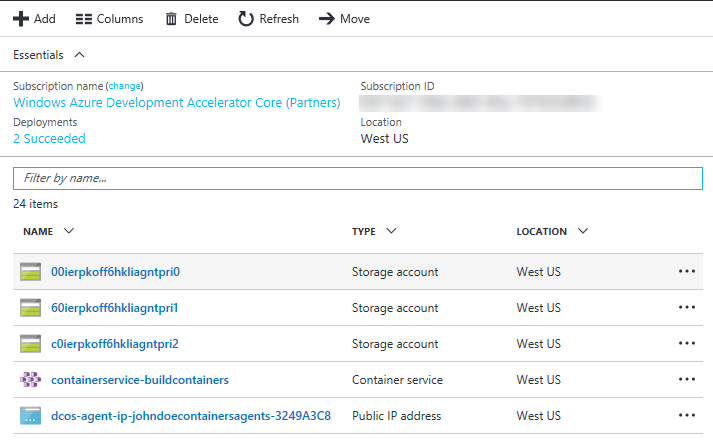
1. On the **Publish** tab, Select **Azure App Service Linux (Preview)** and click **Publish**.



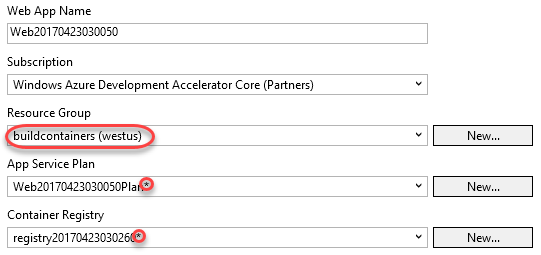
1. From the account dropdown, select **Add an account**. Sign in with the Microsoft account associated with your Azure account.



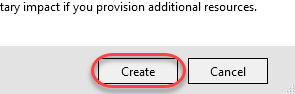
1. Check back in on the status of the container service. If it hasn’t finished deploying yet, wait until it’s done. You’ll know it’s ready when you view the notifications or if the newly created **buildcontainers** resource group blade has been loaded.



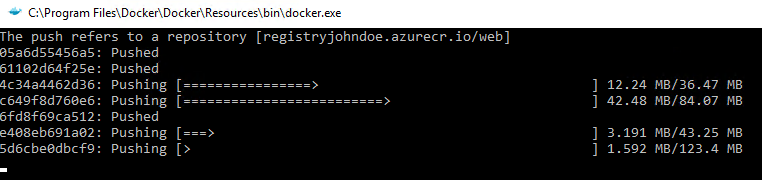
1. Back in **Visual Studio**, select **buildcontainers** as the **Resource Group** to deploy to. Note that if you see **asterisks (\*)**, that means this particular resource will be created for this deployment.



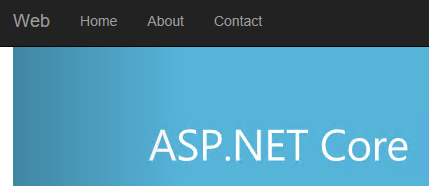
1. To start the publish, click **Create**. If an error occurs, try renaming the **App Service Plan** and **Container Registry** resources and clicking **Create** again.



1. The initial part of the build may take a few minutes before it even starts writing to the log. If there’s an error, you will be notified immediately. Otherwise, please be patient. As the build and publish process progresses, you’ll eventually see the push of the bits specific to this solution. Note that the full container isn’t deployed, but rather just the differential changes needed to update the container from its base image layers.



1. After deployment completes, a browser window will open to your app hosted in a container hosted in Azure App Services.



# Summary

Congratulations on completing this Quick Start Challenge! In this lab, you’ve learned how to build ASP.NET Core container applications and deploy them to Azure App Services.

# Additional Resources

If you are interested in learning more about this topic, you can refer to the following resources:

**Documentation**: <https://azure.microsoft.com/en-us/services/container-service/>

**GitHub**: <https://github.com/Azure/acs-engine>

**Team blog**: <https://azure.microsoft.com/en-us/blog/tag/azure-container-service/>