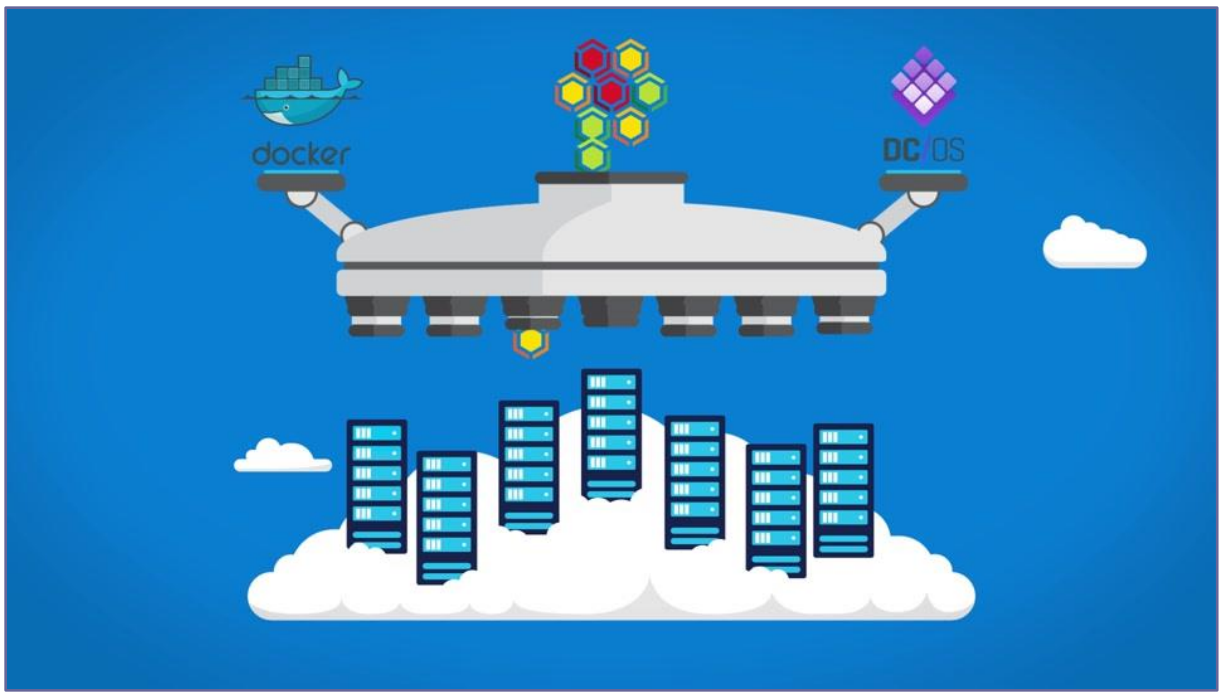


Azure Developer Series

Migrating a legacy ASP.NET 2-tier application to Azure using Container Services

Hands-On-Labs step-by-step guides

Prepared by:

Peter De Tender

CEO and Lead Technical Trainer
PDTIT and 007FFFLearning.com

@pdtit @007FFFLearning

Version: April – 1.0

Contents

Migrating a legacy ASP.NET 2-tiered application to Azure using Container Services - Hands-On-Labs step-by-step

Abstract and Learning Objectives

This workshop enables anyone to learn, understand and build a Proof of Concept, in performing a multi-tiered legacy ASP.NET web application using Microsoft SQL Server database, platform migration to Azure public cloud, leveraging on different Azure Platform Azure A Service (PaaS) and Azure Container Services.

After an introductory module on cloud app migration strategies and patterns, students get introduced to the basics of automating Azure resources deployments using Visual Studio and Azure Resource Manager (ARM) templates. Next, attendees will learn about Microsoft SQL database migration to SQL Azure PaaS, as well as deploying and migrating Azure Web Apps.

After these foundational platform components, the workshop will totally focus on the core concepts and advantages of using containers for running web apps, based on Docker, Azure Container Registry (ACR), Azure Container Instance (ACI), as well as how to enable container cloud-scale using Azure Container Services (ACS) with Kubernetes and Azure Kubernetes Service (AKS).

The focus of the workshop is having a Hands-On-Labs experience, by going through the following exercises and tasks:

- Deploying a 2-tier Azure Virtual Machine (Webserver and SQL database Server) using ARM-template automation with Visual Studio 2017;
- Migrating a legacy SQL 2012 database to Azure SQL PaaS (Lift & Shift);
- Migrating a legacy ASP.NET web application to Azure Web Apps (Lift & Shift);
- Containerizing a legacy ASP.NET web application using Docker;
- Running Azure Container Instance (ACI) from an Azure Container Registry (ACR) image;
- Deploy and run Azure Container Services (ACS) with Kubernetes;
- Deploy and run Azure Kubernetes Services (AKS);
- Managing and Monitoring Azure Container Services (ACS) and Azure Kubernetes Services (AKS);

Hands-On-Lab Scenario

The baseline of the hands-on-lab scenario is starting from an 8-year-old legacy ASP.NET application, developed around 2010, currently offering a product catalog browsing web page, pulling the product catalog list from a legacy Microsoft SQL Server 2012 database, running on dedicated Windows Server 2012 R2 Virtual Machines. (This could be seen as a subset of a larger application landscape within your organization, think of manufacturing database information, HR solutions, e-commerce platforms,… and alike). Imagine this application is running in our on-premises datacenter, where you want to perform an "application digital migration" to Azure Public cloud. You will use several Azure cloud-native services and solutions, ranging from Virtual Machines, Platform Services and different Container Solutions on Azure.

Requirements

Naming Conventions:

IMPORTANT: Most Azure resources require unique names. Throughout these steps you will see the word **"[SUFFIX]"** as part of resource names. You should replace this with your initials, guaranteeing those resources get uniquely named.

Azure Subscription:

Participants need a "pay-as-you-go", MSDN or other paid Azure subscription

a) Azure Trial subscriptions won't work
b) In one of the Azure Container Services tasks, you are required to create an Azure AD Service Principal, wich typically requires an Azure subscription owner to log in to create this object. If you don't have the owner right in your Azure subscription, you could ask another person to execute this step for you.
c) The Azure subscription must allow you to run enough cores, used by the baseline Virtual Machines, but also later on in the tasks when deploying the Azure Container Services, where ACS agent and master machines are getting set up. If you follow the instructions as written out in the lab guide, you need 12 cores.
d) If you run this lab setup in your personal or corporate Azure payable subscription, using the configuration as described in the lab guide, the estimated Azure consumption costs for running the setups during the 2 days of the workshop is $20.

Other requirements:

Participants need a local client machine, running a recent Operating System, allowing them to:

- browse to https://portal.azure.com from a most-recent browser;
- establish a secured Remote Desktop (RDP) session to a lab-jumpVM running Windows Server 2016;

Alternative Approach:

Where the lab scenario assumes all exercises will be performed from within the lab-jumpVM, (since several tools will be installed on the lab-jumpVM or are already installed by default), participants could also execute (most, if not all…) steps from their local client machine.

The following tools are being used throughout the lab exercises:

- Visual Studio 2017 community edition (updated to latest version)
- Docker for Windows (updated to latest version)
- Azure CLI 2.0 (updated to latest version)
- Kubernetes CLI (updated to latest version)

Make sure you have these tools installed prior to the workshop, if you are not using the lab-jumpVM. You should also have full administrator rights on your machine to execute certain steps within using these tools.

Final Remarks:

VERY IMPORTANT: You should be typing all of the commands as they appear in the guide, except where explicitly stated in this document. Do not try to copy and paste from Word to your command windows or other documents where you are instructed to enter information shown in this document. There can be issues with Copy and Paste from Word or PDF that result in errors, execution of instructions, or creation of file content.

IMPORTANT: Most Azure resources require unique names. Throughout these steps you will see the word "[SUFFIX]" as part of resource names. You should replace this with your initials, guaranteeing those resources get uniquely named.

Lab 3: Migrating an ASP.NET web application to Azure Web Apps

What you will learn

In this lab, you will publish your legacy ASP.NET application to an Azure Web App, out of Visual Studio 2017.

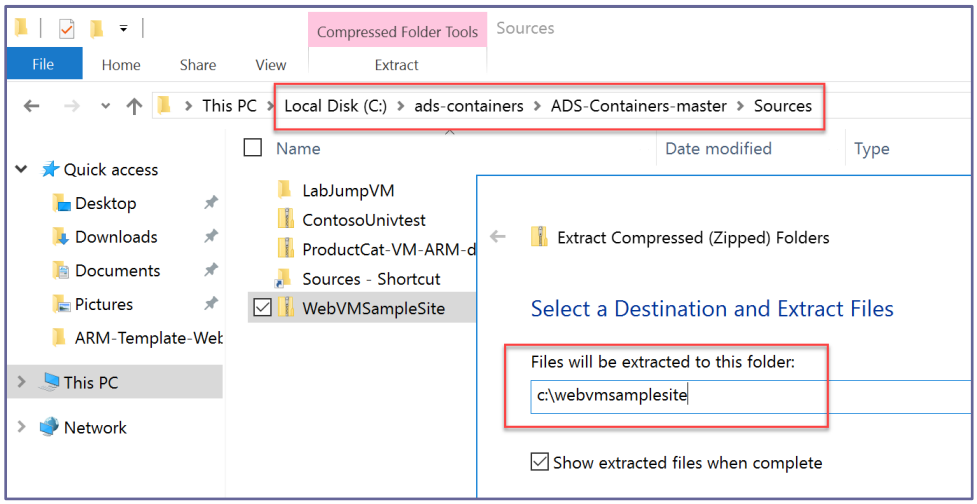
Time Estimate

This lab shouldn't take longer than 15 minutes.

Task 1: Publish an ASP.NET project to Azure Web Apps from within Visual Studio 2017

1. **Log on** to the lab-JumpVM Virtual Machine (fyi, credentials labadmin / L@BadminPa55w.rd).

2. **From the lab-jumpVM**, browse to the folder that holds the GitHub downloaded source files. In here, find the **WebVMSampleSite.zip** file. This is the source code of the legacy ASP.NET web application. **Extract** this file to a folder on the local jumpVM, e.g. c:\webvmsamplesite



3. **Once extracted**, open the file **web.config in the extracted folder.**

4. In here, edit the **<connectionStrings>** settings again, **Replacing the** following settings with the parameters from the Connection String information in the Azure Portal, in both lines starting with <add name:
   - Data Source=10.0.1.4 => change the 10.0.1.4 with your SQL Server name e.g. adssqlazure0923.database.windows.net in our example
   - **Uid=sa => change** the sa account to labadmin

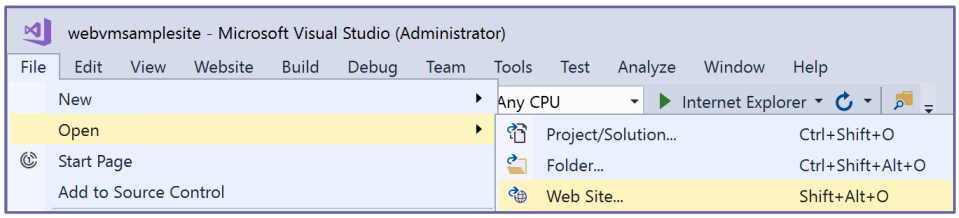
```
<connectionStrings>
```

```
<add name="DefaultConnection" connectionString="Data Source=adssqlazure0923.database.windows.net;
initial catalog=AdventureWorks;Uid=labadmin;Password=L@BadminPa55w.rd;MultipleActiveResultSets=True"
providerName="System.Data.SqlClient" />
<add name="AdventureWorksEntities" connectionString="metadata=res://*/Models.AdventureWorks.csdl
|res://*/Models.AdventureWorks.ssdl |res://*/Models.AdventureWorks.msl;provider=System.Data.SqlClient;
provider connection string=&quot;data source=adssqlazure0923.database.windows.net;initial
catalog=AdventureWorks;Uid=labadmin;Password=L@BadminPa55w.rd;multipleactiveresultsets=True;App=Entity
Framework&quot;" providerName="System.Data.EntityClient" />
  </connectionStrings>
```

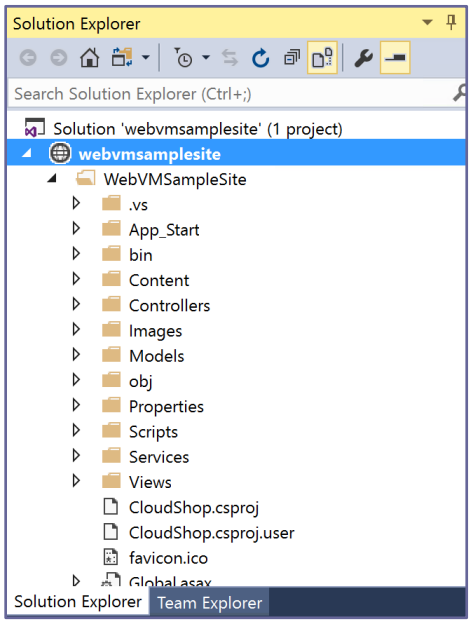
5. Save the changes to the web.config file.

6. Once this step is done, launch Visual Studio 2017.

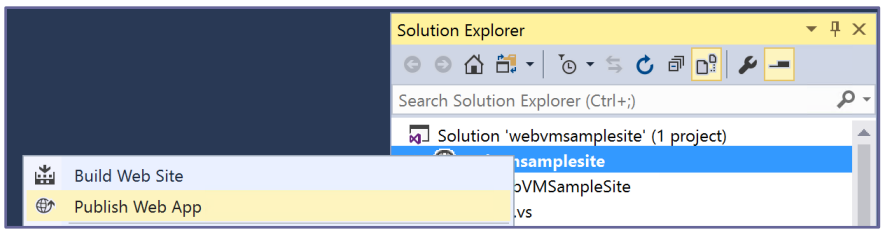
7. From the Visual Studio menu, click File / Open / Web Site... and browse to the location where you extracted the webvmsamplesite



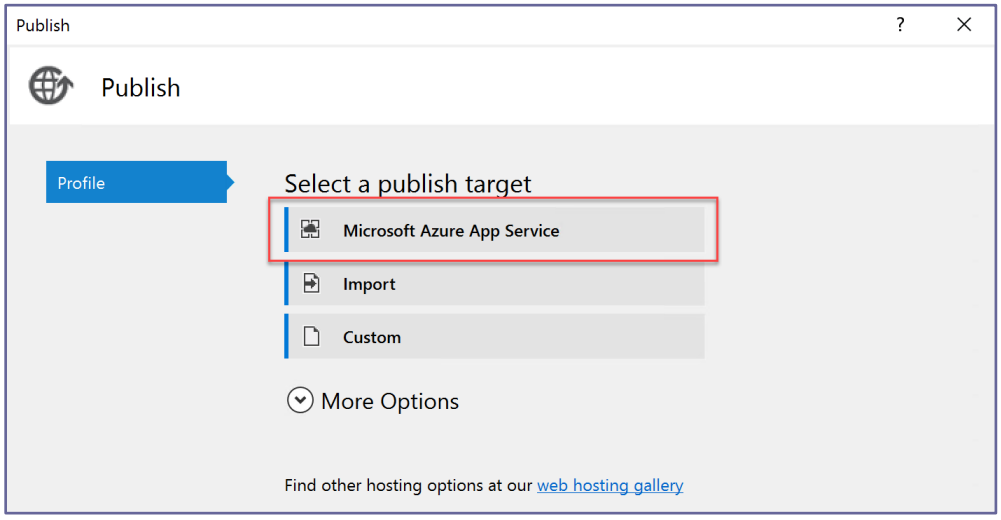
8. The Visual Studio Solution Explorer should show you the contents of the folder



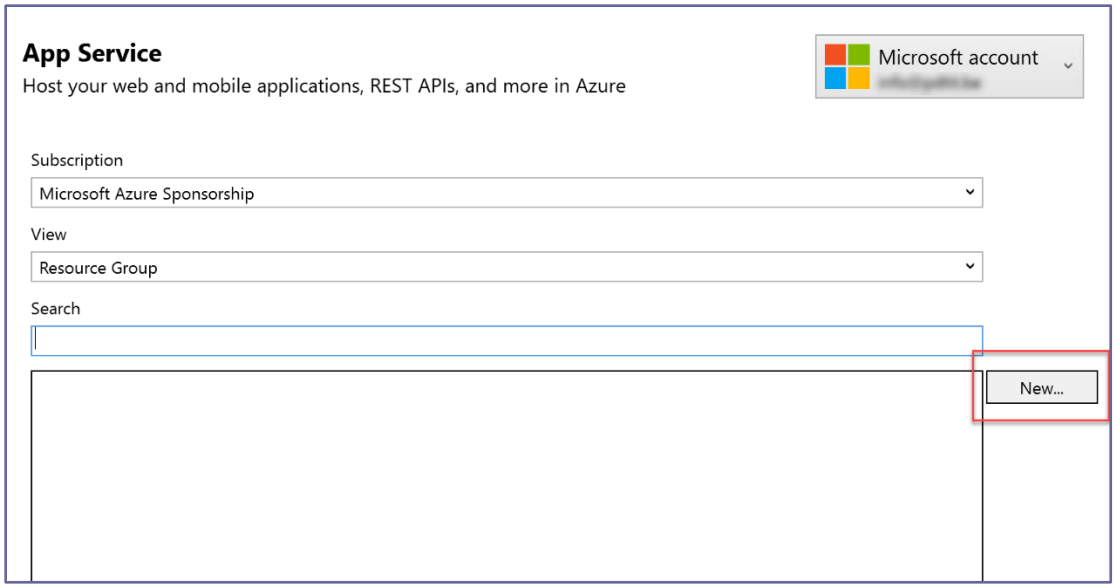
9. Now, right-click on the web site again, and this time choose Publish web app



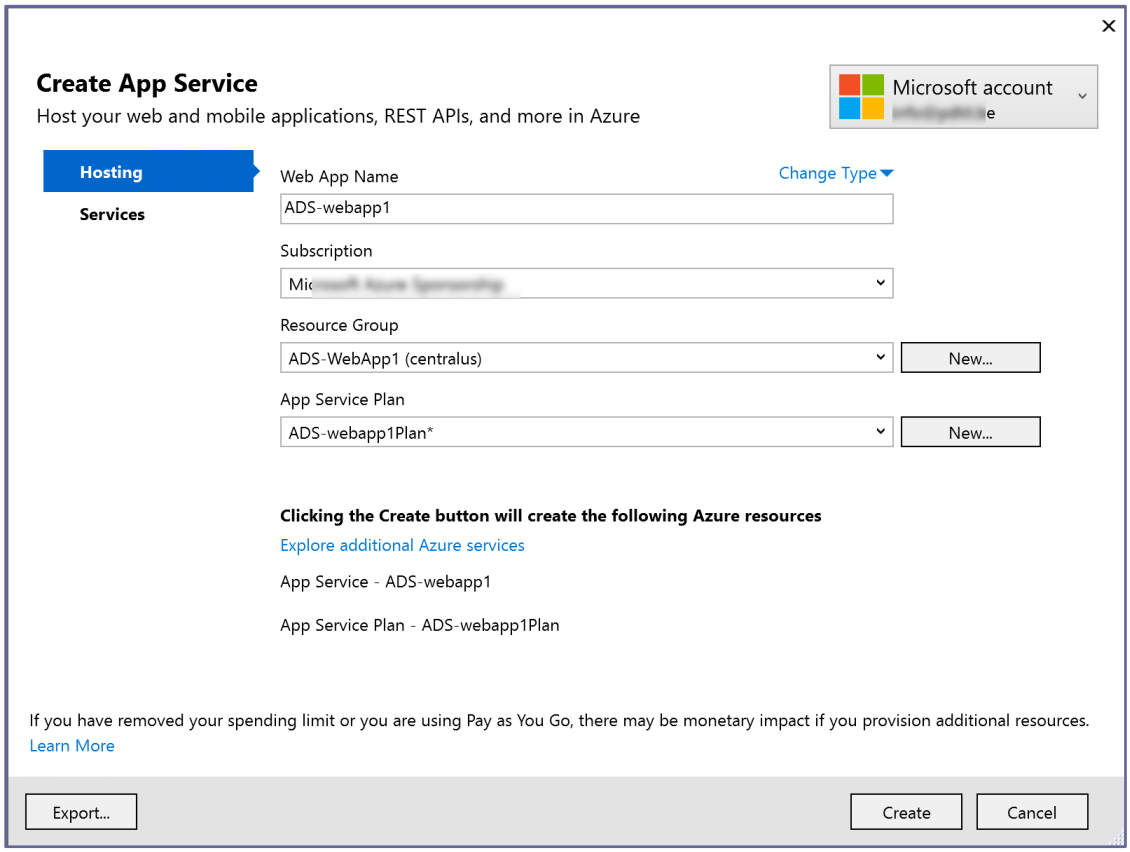
10. This launches the Publish wizard. In the **Select a publish target**, choose **Microsoft Azure App Service**.



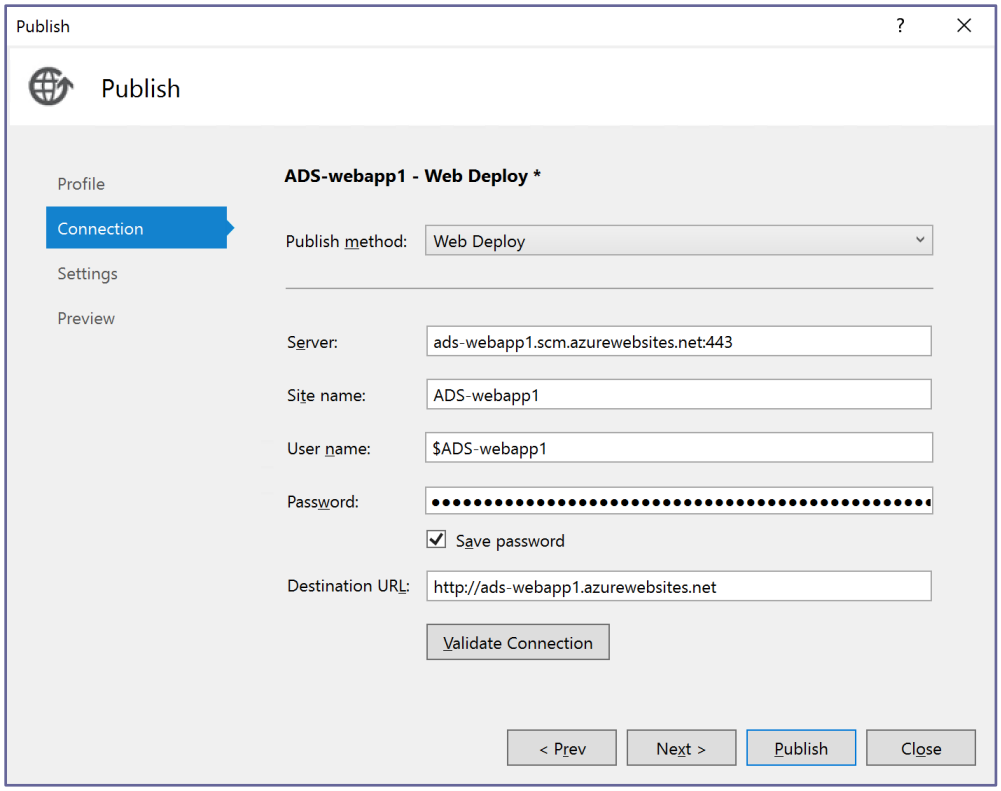
11. In the **App Service** step, validate your Azure credentials and subscription are the correct ones, and click the **New** button.



12. In the **Create App Service** step, define
    - the WebApp Name: **"[SUFFIX]-webapp1"**
    - a **New Resource Group "[SUFFIX]-webapp1"**,
    - a **New App Service Plan "[SUFFIX]-webapp1ServicePlan"**

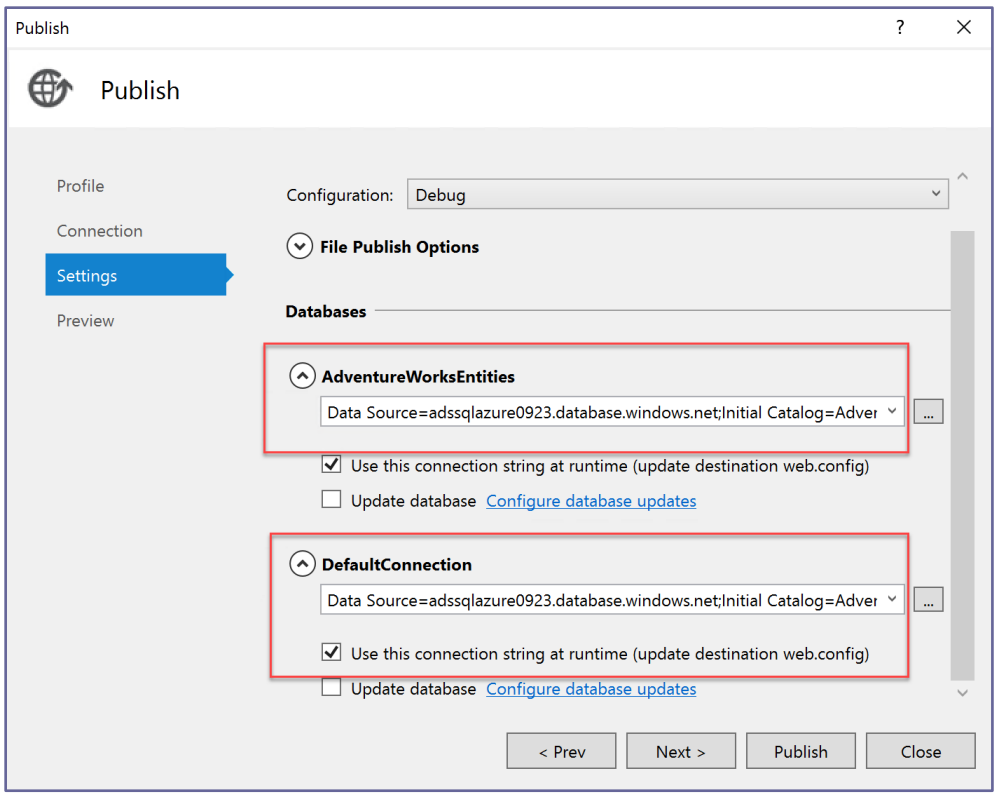


13. **Click** the **Create button** to kick off the actual Web App resource creation. This creates the different Azure Resource objects (Resource Group and App Service Plan) in the back-end. After which you get redirected back to the **Publish** wizard.



14. Click Next to continue to the next step in the Publish wizard.

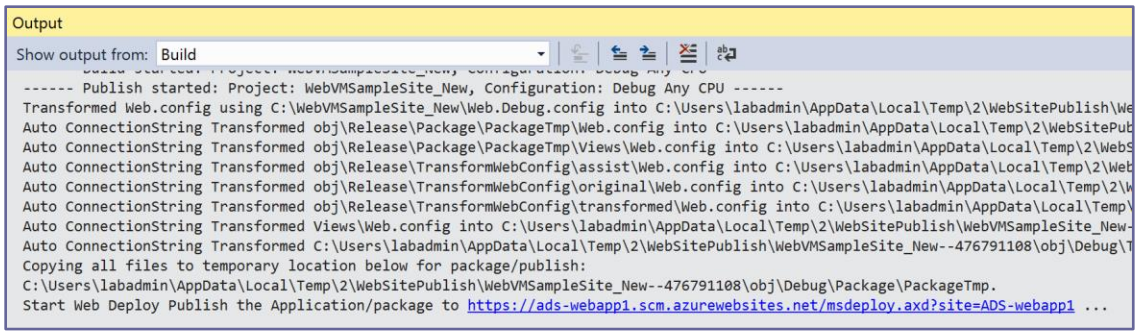
15. Out of the integrated Web App Publishing Wizard, the website specific settings from the web.config file are being analyzed, more specific the database connection strings.



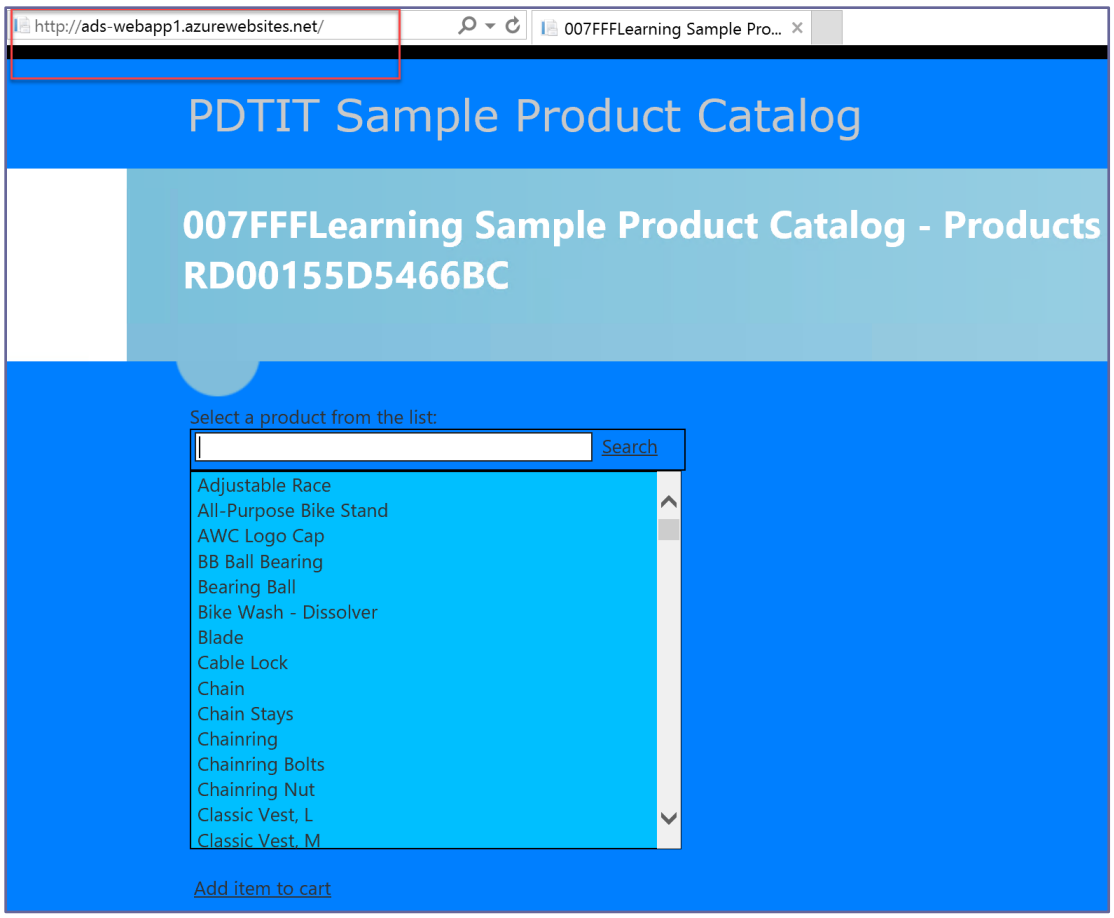
16. Accept the presented information. (Note this gives interesting options towards developers to test the web app with test/dev databases, right from within this web app publishing wizard).

17. Click Publish to finalize this publish wizard step, and wait for the Web App publish steps to be completed successfully.

18. From the Visual Studio Output window, you can follow the publishing sequence in a log file.



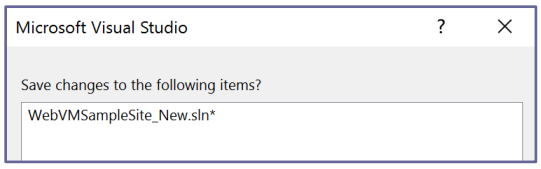
19. Once the publish step is complete, Visual Studio will open your internet browser and connect to the URL of the new Azure Web App.



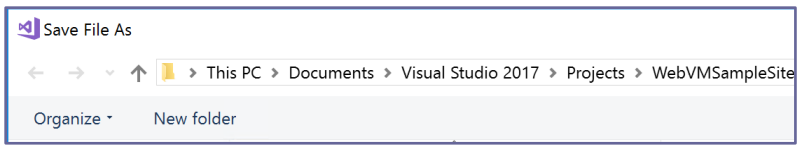
20. This confirms our Azure Web is published successfully, communicating with the SQL Azure database.

21. Close your internet browser.

22. Go back to **Visual Studio 2017**, and **close** the application. This will **prompt you to save** the Visual Studio application (.sln).



23. **Click Yes**, and accept the default location that is offered.



24. This closes Visual Studio 2017.

This completes this lab.

Summary

In this lab exercise, you learned how to update connection string settings in a web.config file, and how to publish your website code to an Azure Web App, by using the Visual Studio Web App Publishing wizard. Lastly, you saved your web site content as a new Visual Studio Project.

Migrating a legacy ASP.NET 2-tier application to Azure using Container Services

Hands-On-Labs step-by-step guides

Prepared by:

Peter De Tender

CEO and Lead Technical Trainer
PDTIT and 007FFFLearning.com

@pdtit @007FFFLearning

Version: April 2019 – 1.0