Azure Developer Series

Migrating a dotnetcore 2-tier application to Azure, using different architectures and DevOps best practices

Hands-On-Labs step-by-step guides

Prepared by:

Peter De Tender

CEO and Lead Technical Trainer
PDTIT and 007FFFLearning.com

@pdtit @007FFFLearning

Version: Sept 2019 – 1.0

Contents

Migrating a dotnetcore 2-tiered application to Azure using different architectures and DevOps best practices - Hands-On-Labs step-by-step

You are part of an organization that is running a dotnetcore e-commerce platform application, using Windows Server infrastructure on-premises today, comprising a WebVM running Windows Server 2012 R2 with Internet Information Server (IIS) and a 2nd SQLVM running Windows Server 2012 R2 and SQL Server 2014.

The business has approved a migration of this business-critical workload to Azure, and you are nominated as the cloud solution architect for this project. No decision has been made yet on what the final architecture should or will look like. Your first task is building a Proof-of-Concept in your Azure environment, to test out the different architectures possible:

- Infrastructure as a Service (IAAS)
- Platform as a Service (PAAS)
- Containers as a Service (CaaS)

At the same time, your CIO wants to make use of this project to switch from a more traditional mode of operations, with barriers between IT sysadmin teams and Developer teams, to a DevOps way of working. Therefore, you are tasked to explore Azure DevOps and determine where CI/CD Pipelines can assist in optimizing the deployment and running operations of this e-commerce platform, especially when deploying updates to the application.

As you are new to the continuous changes in Azure, you want to make sure this process goes as smooth as possible, starting from the assessment to migration to day-to-day operations.

Abstract and Learning Objectives

This workshop enables anyone to learn, understand and build a Proof of Concept, in performing a multi-tiered .Net Core web application using Microsoft SQL Server database, platform migration to Azure public cloud, leveraging on different Azure Infrastructure as a Service, Azure Platform as a Service (PaaS) and Azure Container offerings like Azure Container Instance (ACI) and Azure Kubernetes Services (AKS).

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 learn about the importance of performing proper assessments, and what tools Microsoft offers to help in this migration preparation phase. Once the application has been deployed on Azure Virtual Machines, students learn about Microsoft SQL database migration to SQL Azure PaaS, as well as deploying and migrating web applications to Azure Web Apps.

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

In the last part of the workshop, students get introduced to Azure DevOps, the new Microsoft Application Lifecycle environment, helping in building a CI/CD Pipeline to publish workloads using the DevOps principals and concepts, showing the integration with the rest of the already touched on Azure services like Azure Web Apps and Azure Kubernetes Services (AKS), closing the workshop with a module on overall Azure monitoring and operations and what tools Azure has available to assist your IT teams in this challenge.

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 2019;
- Publishing a .NET Core e-commerce application to an Azure Web Virtual Machine and SQL DB Virtual Machine;
- Performing a proper assessment of the as-is Web and SQL infrastructure using Microsoft Assessment Tools;
- Migrating a SQL 2014 database to Azure SQL PaaS (Lift & Shift);
- Migrating a .NET Core web application to Azure Web Apps (Lift & Shift);
- Containerizing a .NET Core web application using Docker, and pushing to Azure Container Registry (ACR);
- Running Azure Container Instance (ACI) and WebApp for Containers;
- Deploy and run Azure Azure Kubernetes Services (AKS);

- Deploying Azure DevOps and building a CI/CD Pipeline for the subject e-commerce application;
- Managing and Monitoring Azure Kubernetes Services (AKS);

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) 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.
b) 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.
c) 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); this could also be Visual Studio 2019 community edition - latest version
- Docker for Windows (updated to latest version)
- Azure CLI 2.0 (updated to latest version)
- Kubernetes CLI (updated to latest version)
- SimplCommerce Open Source e-commerce platform example (http://www.simplcommerce.com)

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 4: Deploying an Azure Web App and migrating from Web VM

What you will learn

In this lab, you will publish your dotnetcore application source code to an Azure Web App, out of Visual Studio 2017/2019.

In a 2nd task, you will continue on the path of the Azure App Services Migration Assistant, and running the actual web application migration from within that tool to a different Azure Web App.

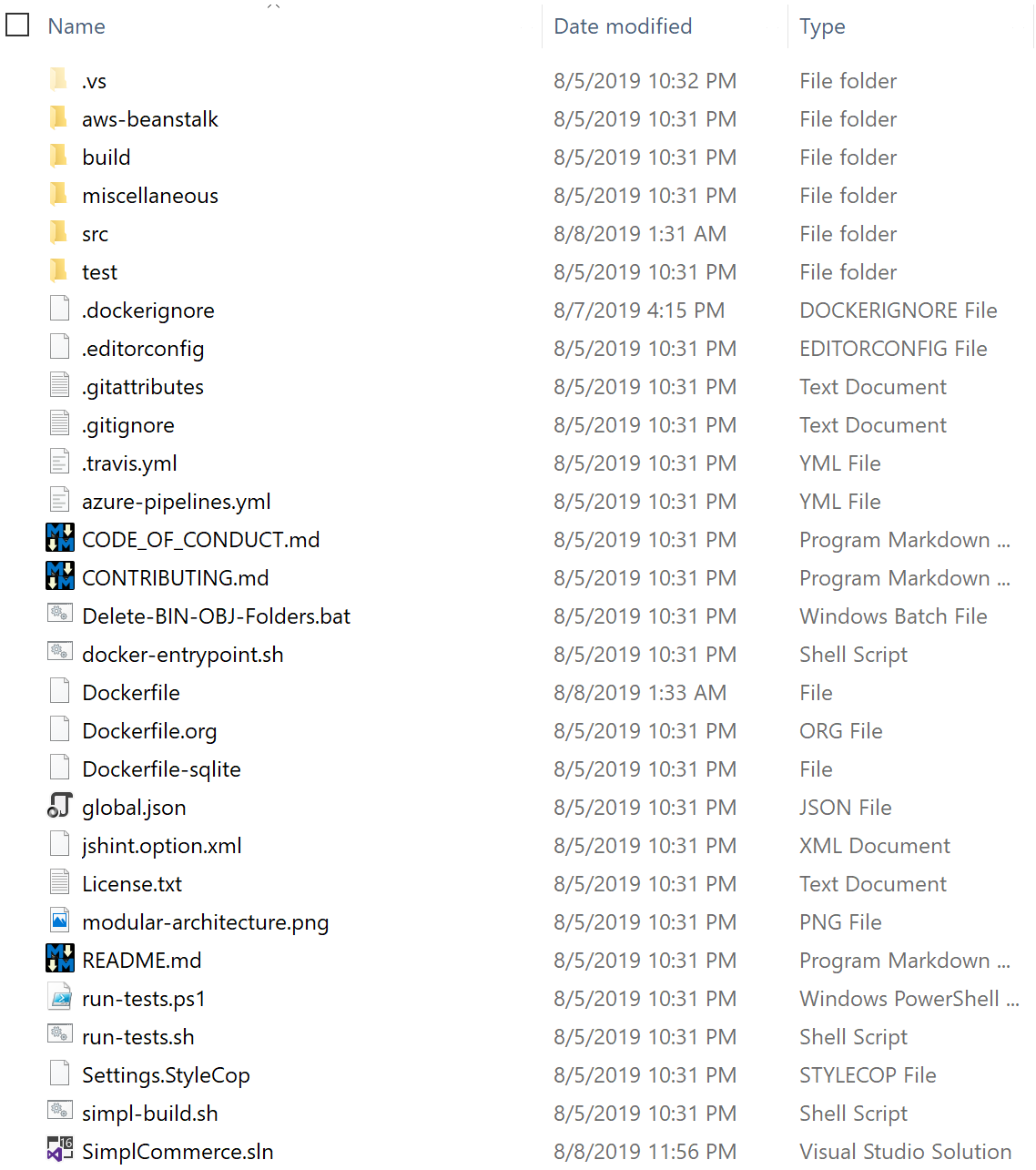
Time Estimate

This lab is estimated to take 45min in total.

Task 1: Publish an ASP.NET project to Azure Web Apps from within Visual Studio 2017/2019

1. **Log on** to the lab-JumpVM Virtual Machine (fyi, credentials labadmin / L@BadminPa55w.rd), or your own developer workstation, having VStudio 2017/2019 with the latest updates running.

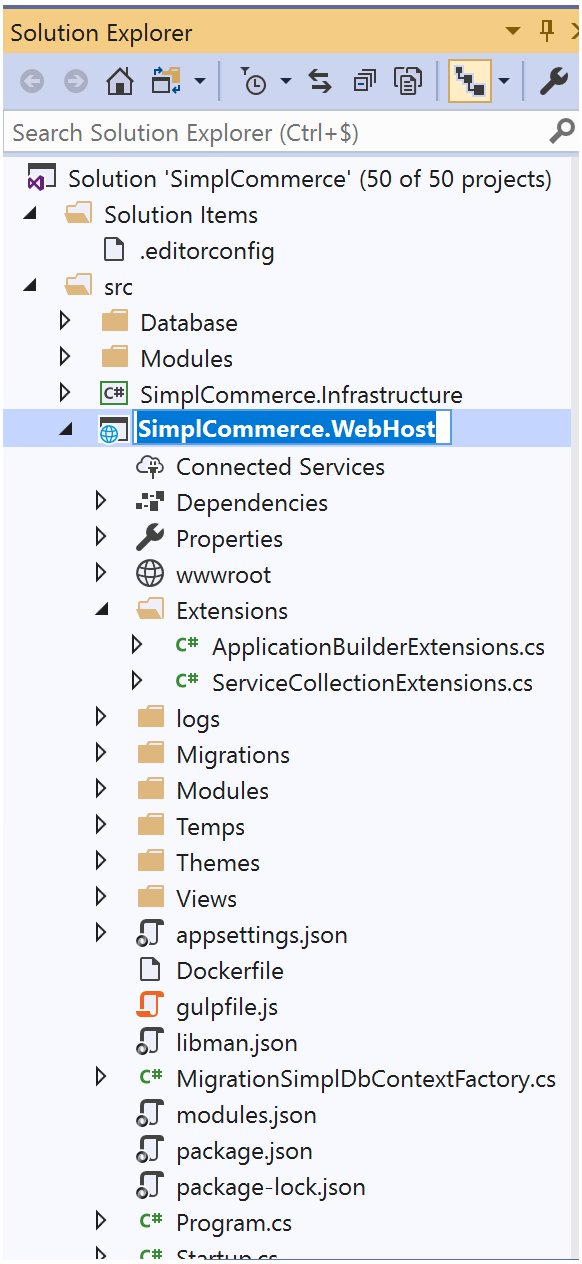
2. From the **lab-jumpVM**, browse to the folder that holds the GitHub downloaded source files. In here, find the SimplCommerceVSSRC.zip file. This is the source code of the dotnet SimplCommerce e-commerce sample app we are using. Extract this file to a folder on the local jumpVM, e.g. c:\SimplCommerceVSSRC



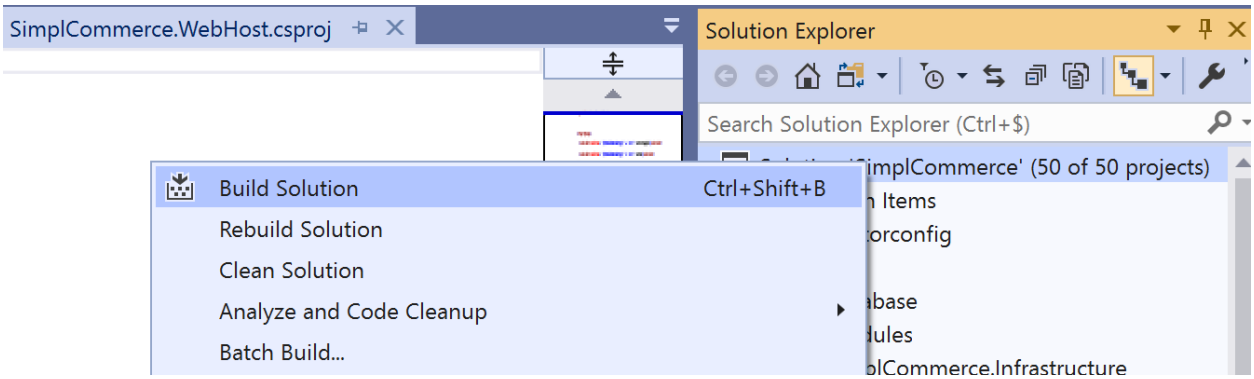
Name	Date modified	Type
.vs	8/5/2019 10:32 PM	File folder
aws-beanstalk	8/5/2019 10:31 PM	File folder
build	8/5/2019 10:31 PM	File folder
miscellaneous	8/5/2019 10:31 PM	File folder
src	8/8/2019 1:31 AM	File folder
test	8/5/2019 10:31 PM	File folder
.dockerignore	8/7/2019 4:15 PM	DOCKERIGNORE File
.editorconfig	8/5/2019 10:31 PM	EDITORCONFIG File
.gitattributes	8/5/2019 10:31 PM	Text Document
.gitignore	8/5/2019 10:31 PM	Text Document
.travis.yml	8/5/2019 10:31 PM	YML File
azure-pipelines.yml	8/5/2019 10:31 PM	YML File
CODE_OF_CONDUCT.md	8/5/2019 10:31 PM	Program Markdown ...
CONTRIBUTING.md	8/5/2019 10:31 PM	Program Markdown ...
Delete-BIN-OBJ-Folders.bat	8/5/2019 10:31 PM	Windows Batch File
docker-entrypoint.sh	8/5/2019 10:31 PM	Shell Script
Dockerfile	8/8/2019 1:33 AM	File
Dockerfile.org	8/5/2019 10:31 PM	ORG File
Dockerfile-sqlite	8/5/2019 10:31 PM	File
global.json	8/5/2019 10:31 PM	JSON File
jshint.option.xml	8/5/2019 10:31 PM	XML Document
License.txt	8/5/2019 10:31 PM	Text Document
modular-architecture.png	8/5/2019 10:31 PM	PNG File
README.md	8/5/2019 10:31 PM	Program Markdown ...
run-tests.ps1	8/5/2019 10:31 PM	Windows PowerShell ...
run-tests.sh	8/5/2019 10:31 PM	Shell Script
Settings.StyleCop	8/5/2019 10:31 PM	STYLECOP File
simpl-build.sh	8/5/2019 10:31 PM	Shell Script
SimplCommerce.sln	8/8/2019 11:56 PM	Visual Studio Solution

Note this folder contains more source files than what we need in this lab, but don't delete those, as you will use some of those in the labs coming.

3. **Open** the file **SimplCommerce.sln**, which should open your Visual Studio 2017/2019 development environment. From **Solution Explorer**, the structure should look similar to this:

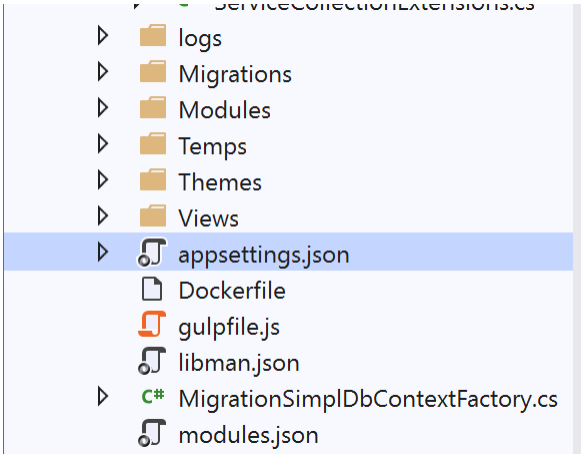


4. If the source files are not loading correctly, you might need to rebuild the project first before continuing. This can be done from **selecting the Solution 'SimplCommerce'**, **right-clicking** it and select **Build Solution**.

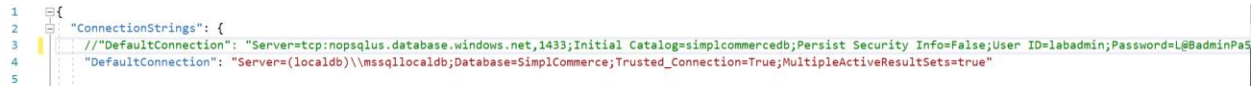


5. Before testing and validating the application to run successful, we need to update the **appsetings.json** file, providing a correct database connection string.This can be done directly from within the Visual Studio editor.

   From Solution Explorer, browse to the file **appsettings.json**, and open it.

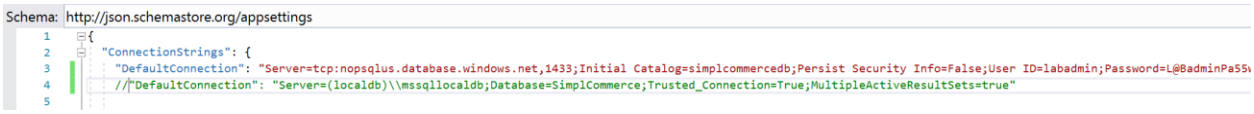


6. The file looks like this example:

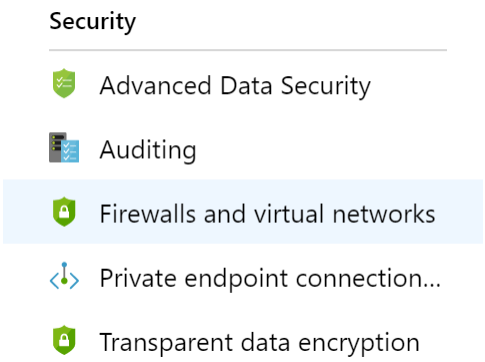


7. You can choose what database scenario you will use here, using a local SQL Express scenario, or updating it and pointing to the Azure SQL database instance you migrated in the previous lab scenario. The recommendation would be switching to the Azure SQL instance 😊

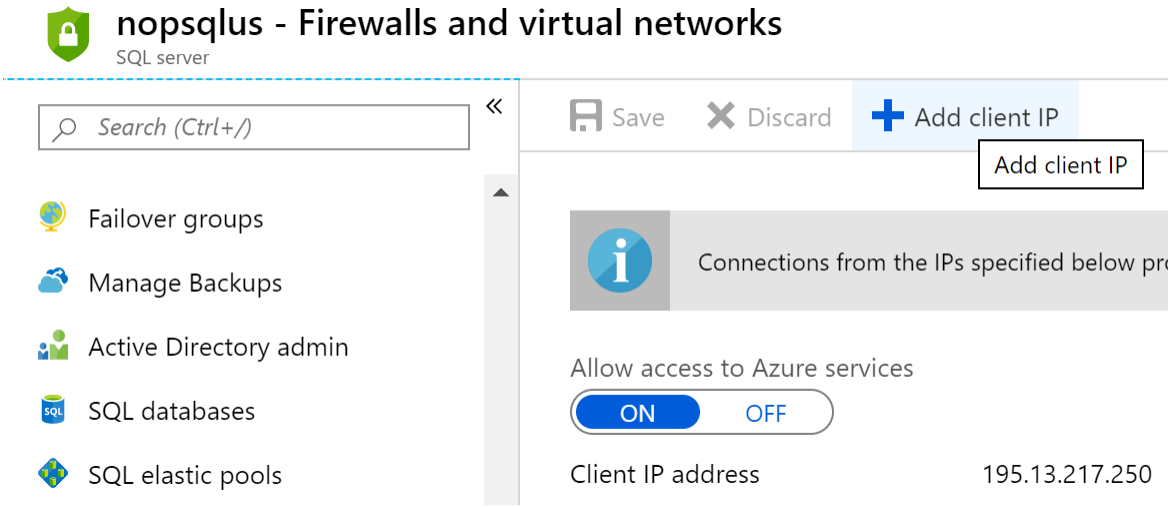
   Validate all the connection string parameters, and save the updated file.



8. Note: if you are running this lab from within your local development workstation, you need to allow your public IP from your internet connection, connecting to the SQL Server instance in Azure. To do this, browse to the Azure SQL Server in the Azure Portal / Security / Firewall and Virtual Networks

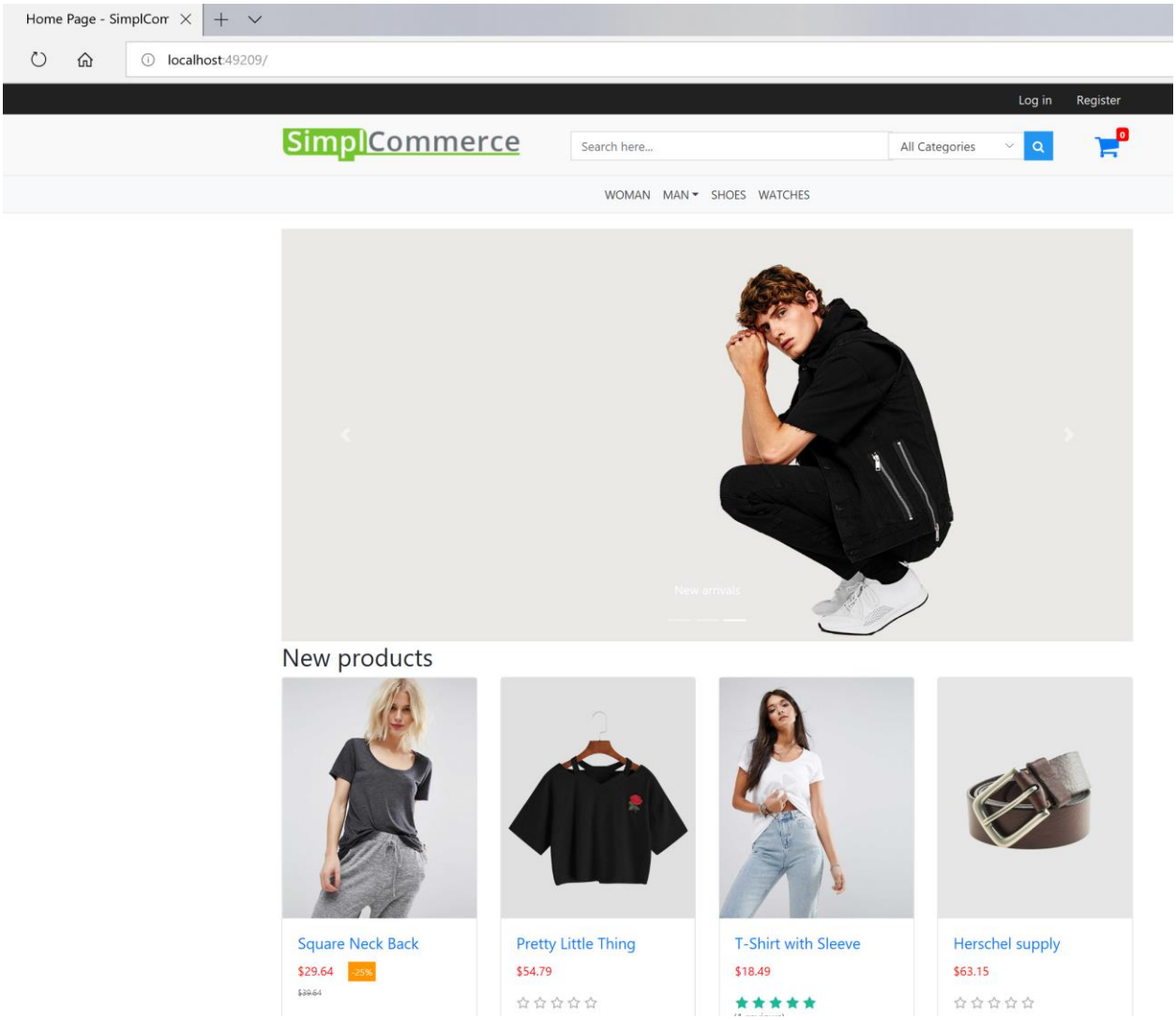


9. Select "Add Client IP" on top, and save the changes.



10. Switch back to your Visual Studio environment, and run the application by pressing "F5", or clicking the "IIS_Express" link in the top menu

11. This compiles the application, and after a few minutes, the e-commerce application start page should show up. The default port in the browser is 49209

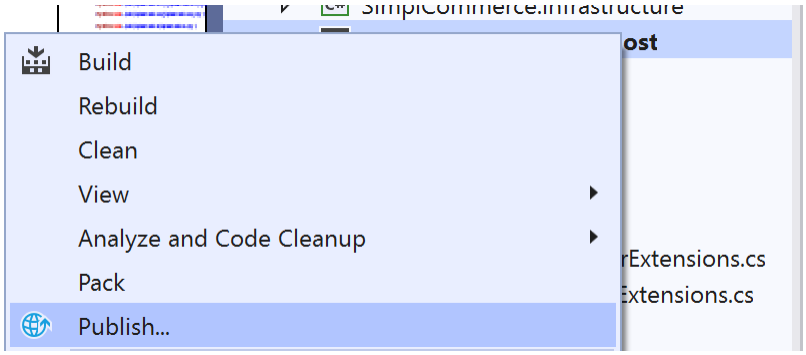


12. While off-topic for our lab scenarios, know this is a fully-functional e-commerce application, allowing you to create new customers, placing orders, updating products,…

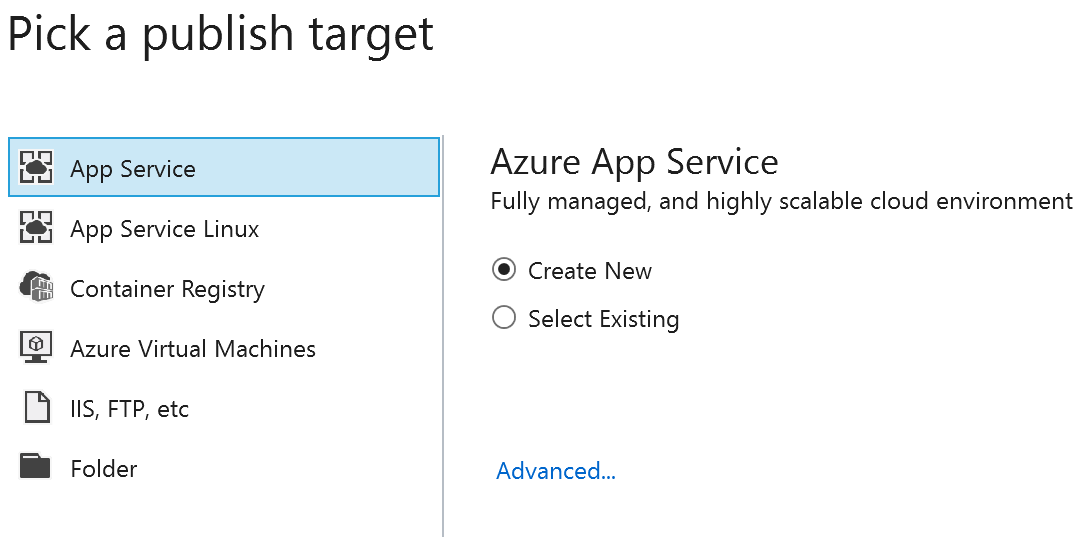
13. This confirms that our web application is working fine. You can close the browser session, which will also end the Visual Studio debugging.

Task 2: Publishing the source code to Azure Web Apps

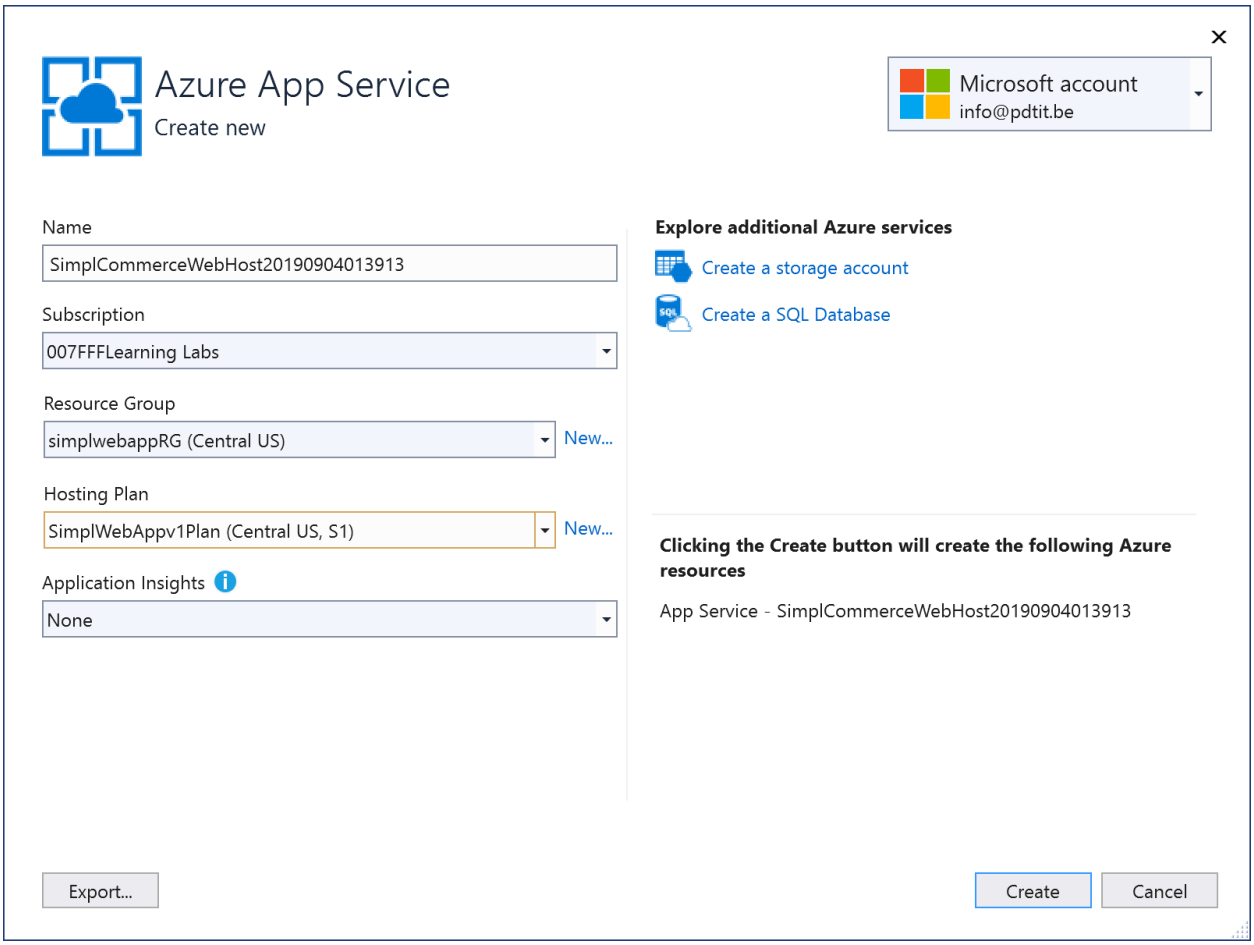
1. From within **Visual Studio Solution Explorer**, select **SimplCommerce.WebHost** , **right-click it**, and select **Publish...**



2. This starts the web application publishing wizard. In the **Pick a publish target** step, select **App Service / Create New** (given it's a dotnetcore application, you could also select App Service Linux as an alternative...)



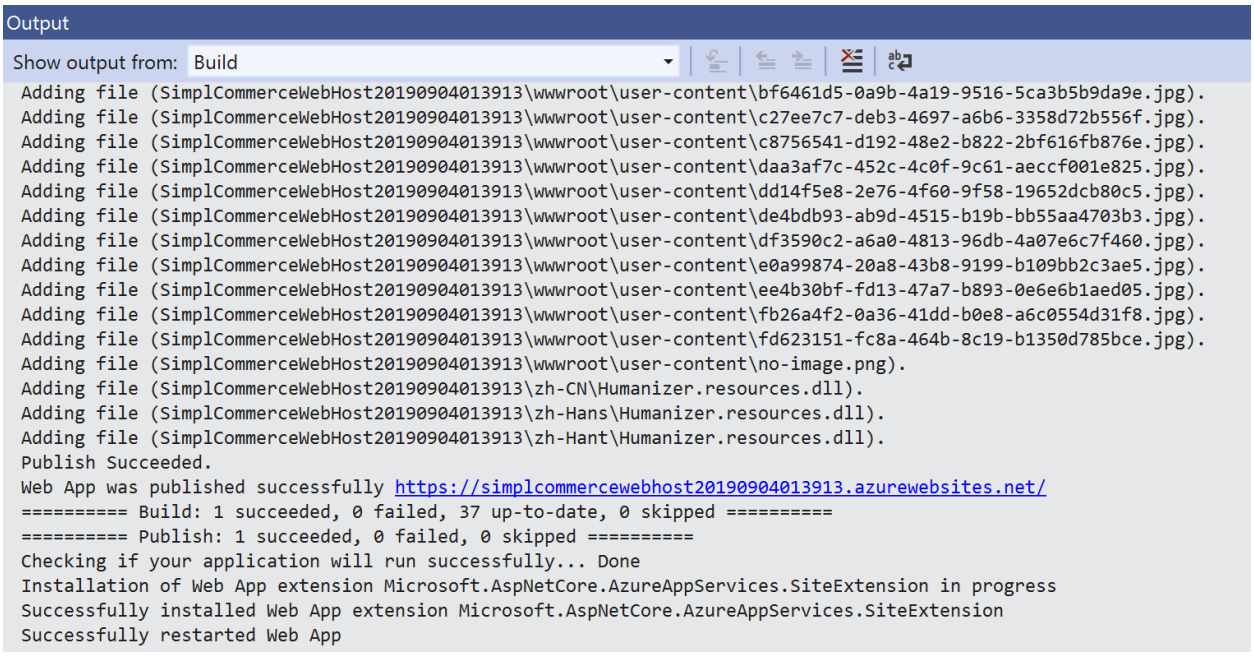
3. **Click Next**



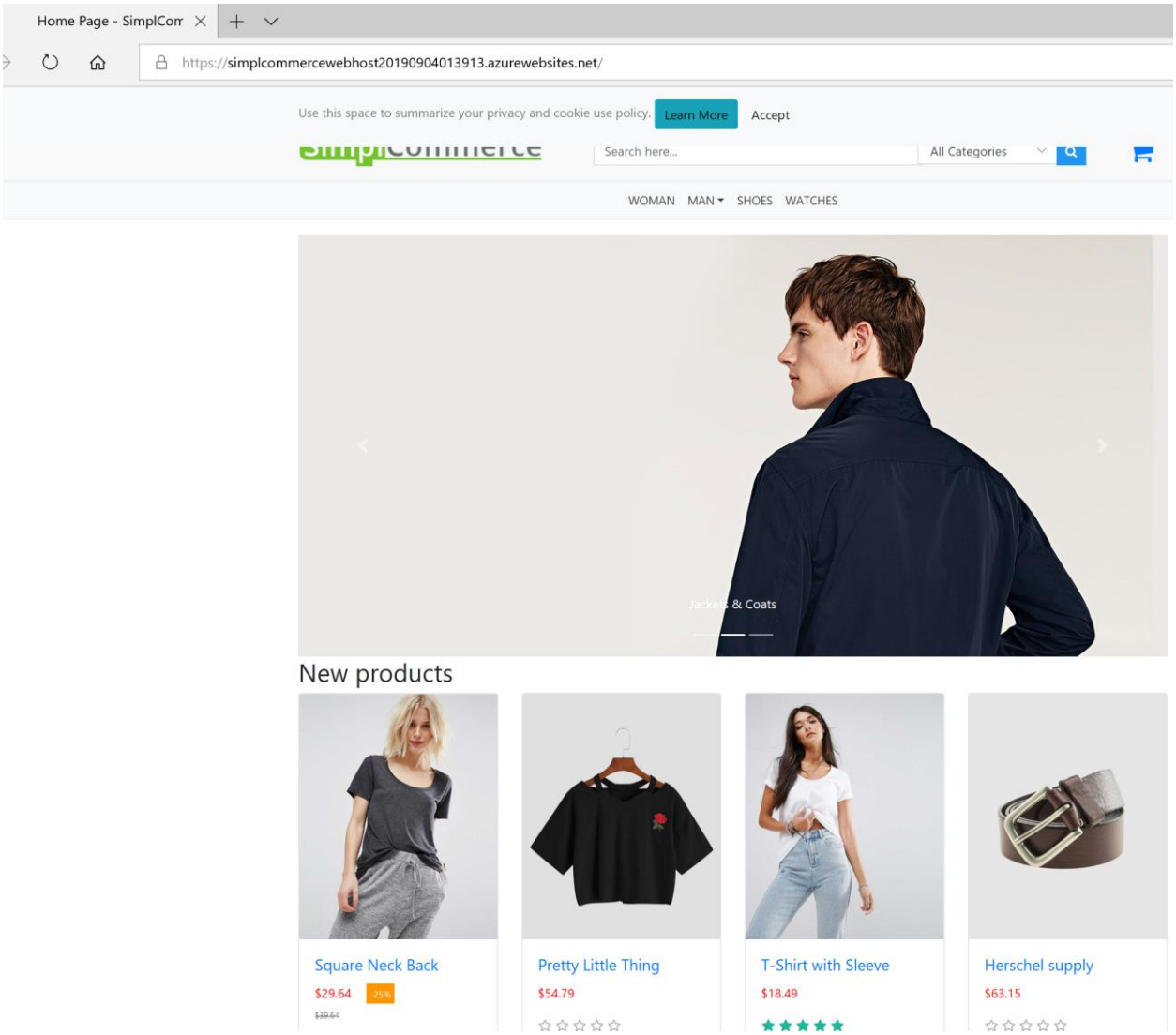
4. In th **Azure App Service** step, complete / validate the different parameters:

   - Name **accept the dynamically generated name, or update to a user-friendly unique**
   - Subscription **Select your Azure Subscription**
   - Resource Group **Select or create a new Resource Group**
   - Hosting Plan **Accept the default of S1, or modify where needed**

5. Confirm by clicking **Create;** the source files are getting pushed to the Azure Web App, where you can follow this process from the Visual Studio Output window:



6. Wait for the process to complete successfully. At the end, Visual Studio will open your default browser, where you can validate the webapp is running successfully.

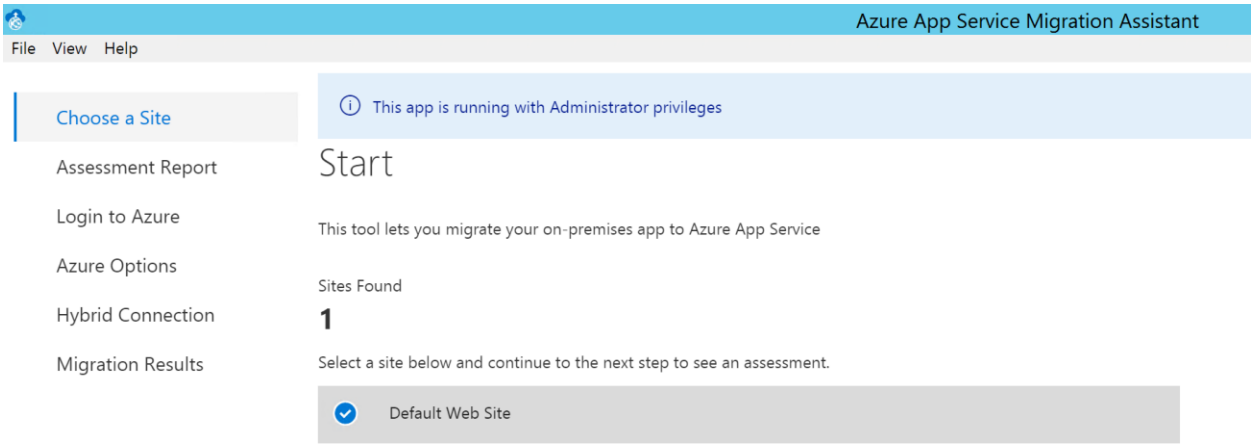


7. This completes the task, in which you loaded the source code into Visual Studio, validating and debugging the web application locally, and pushing it to Azure Web Apps using the Visual Studio wizard integration.

Task 2: Migrating a Web Application from Azure App Service Migration Assistant

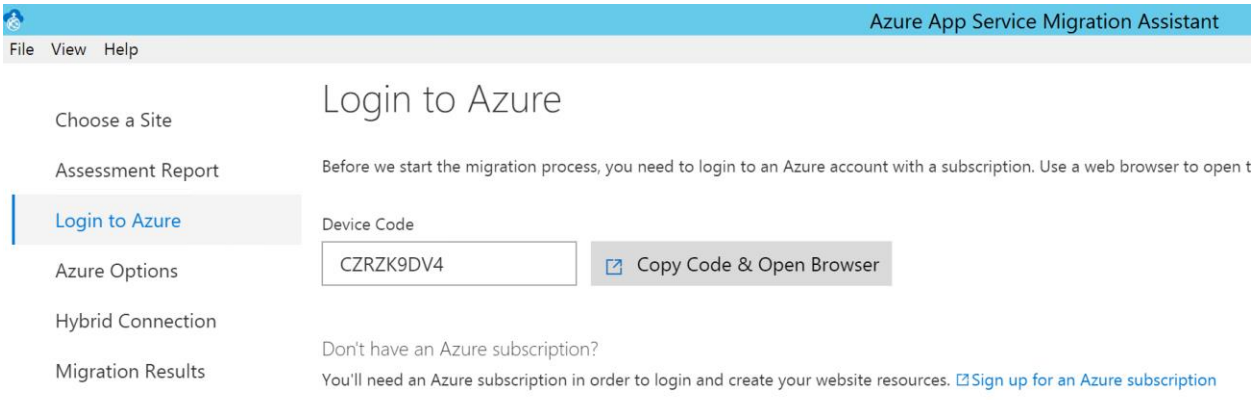
1. **Start an RDP Session** to the **WebVM** you have running in Azure (labadmin / L@BadminPa55w.rd)

2. **From the desktop**, launch **App Service Migration Assistant**. Since we used this tool for performing the web application assessment in a previous lab, it will remember some of those parameters.



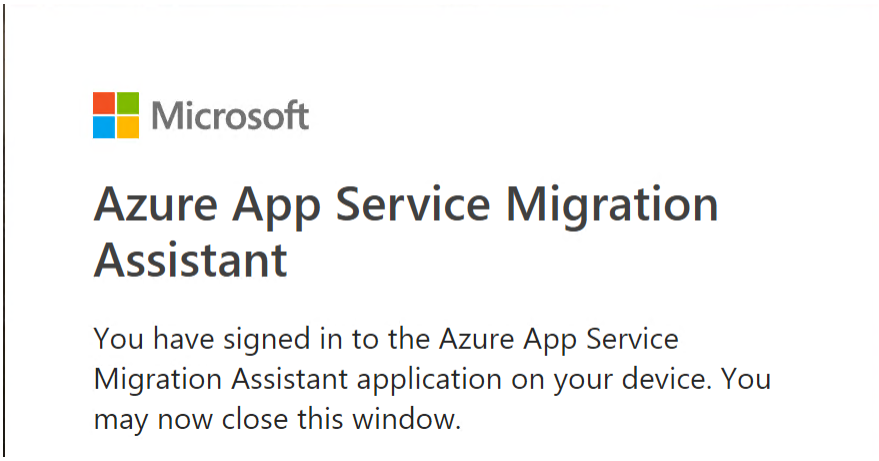
3. Select the detected **Default Web Site**, and click **Next**.

4. The tool will perform another assessment first; when complete, press **Next.** This is where you will launch and execute the actual web app migration. Starting with authenticating to Azure:

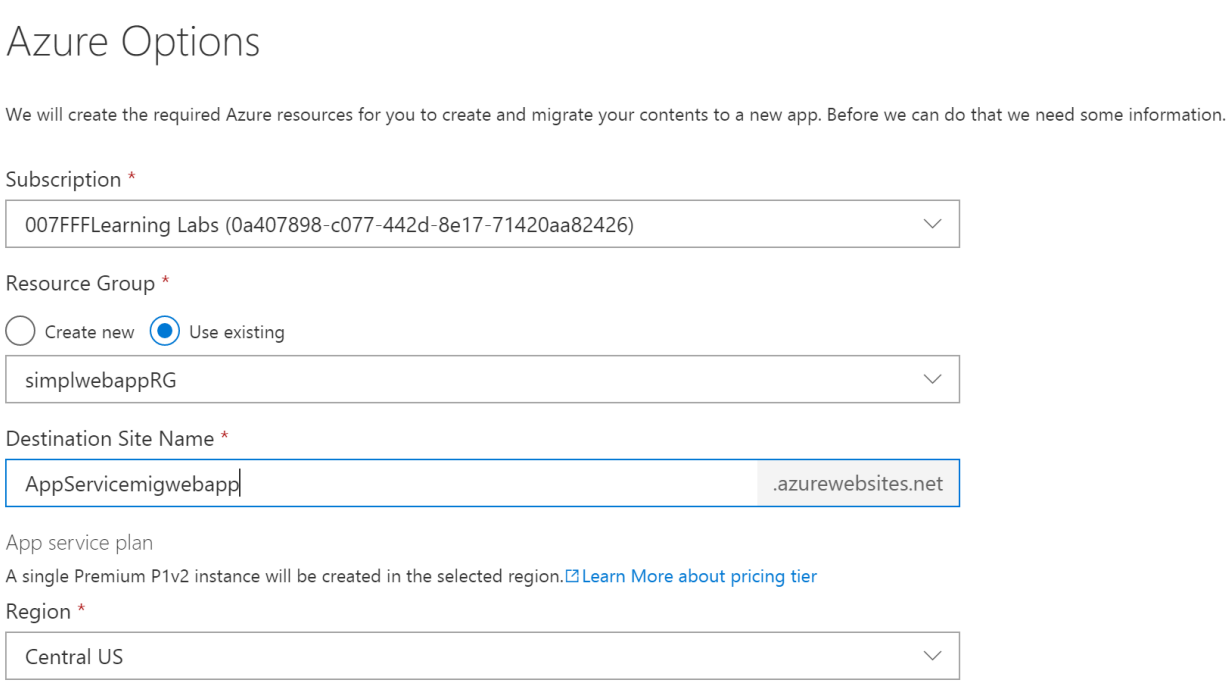


5. **Press "Copy Code & Open Browser"**, and paste in this **Device** Code in the popup window. Next, log on to Azure with your Azure admin credentials in the appearing popup. After a

successful authentication, you are prompted to close your browser session

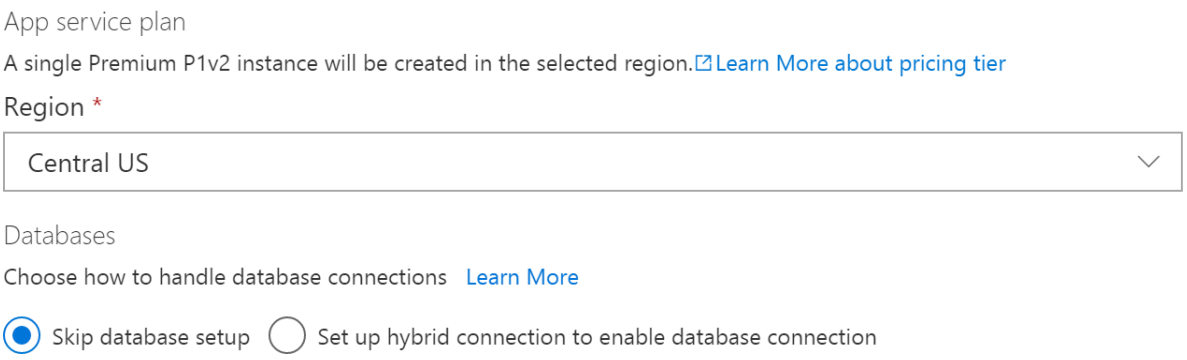


6. **Back** in the App Service Migration Assistant, you can continue with providing the necessary parameters to get the web app deployed and configured, in the **Azure Options** step:

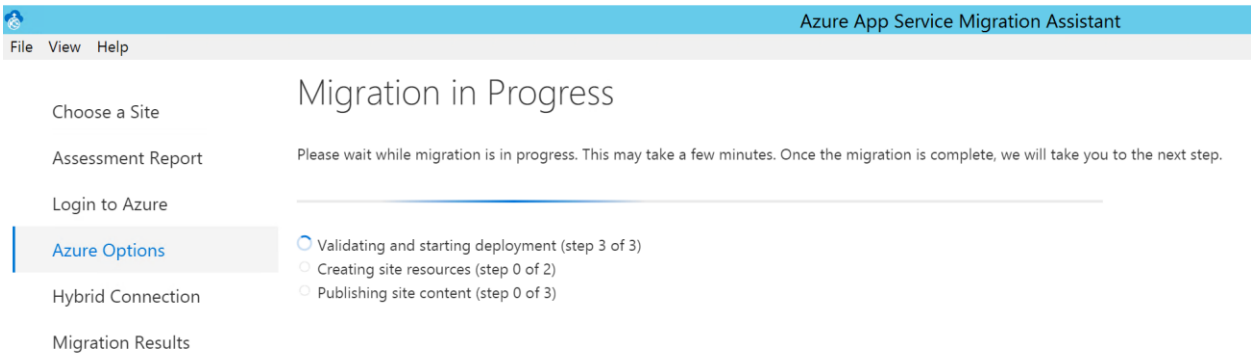


- Resource Group — **select the same one from task 1 earlier**
- Destination Site Name — **provide a unique name for the webapp**
- Region — **your Azure region of choice**

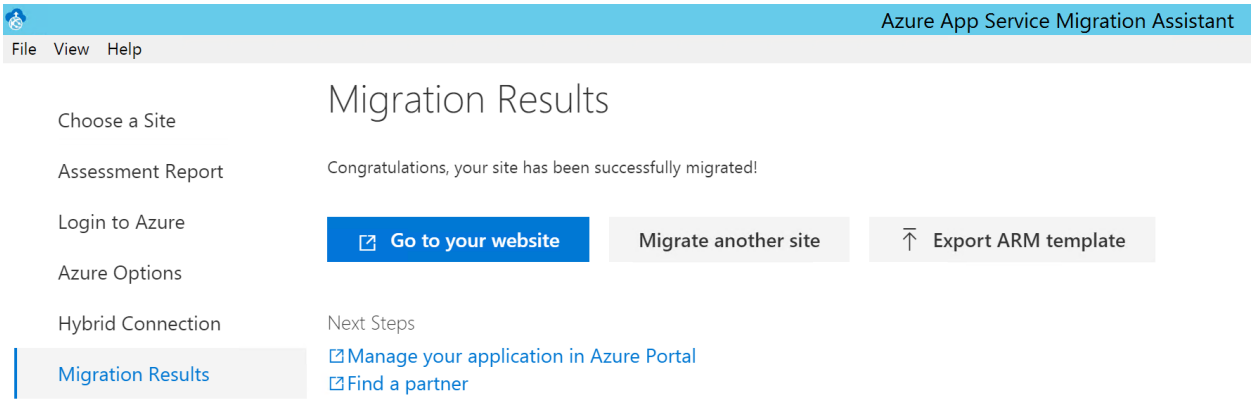
7. In the **Database setup**, choose "**Skip database setup**"



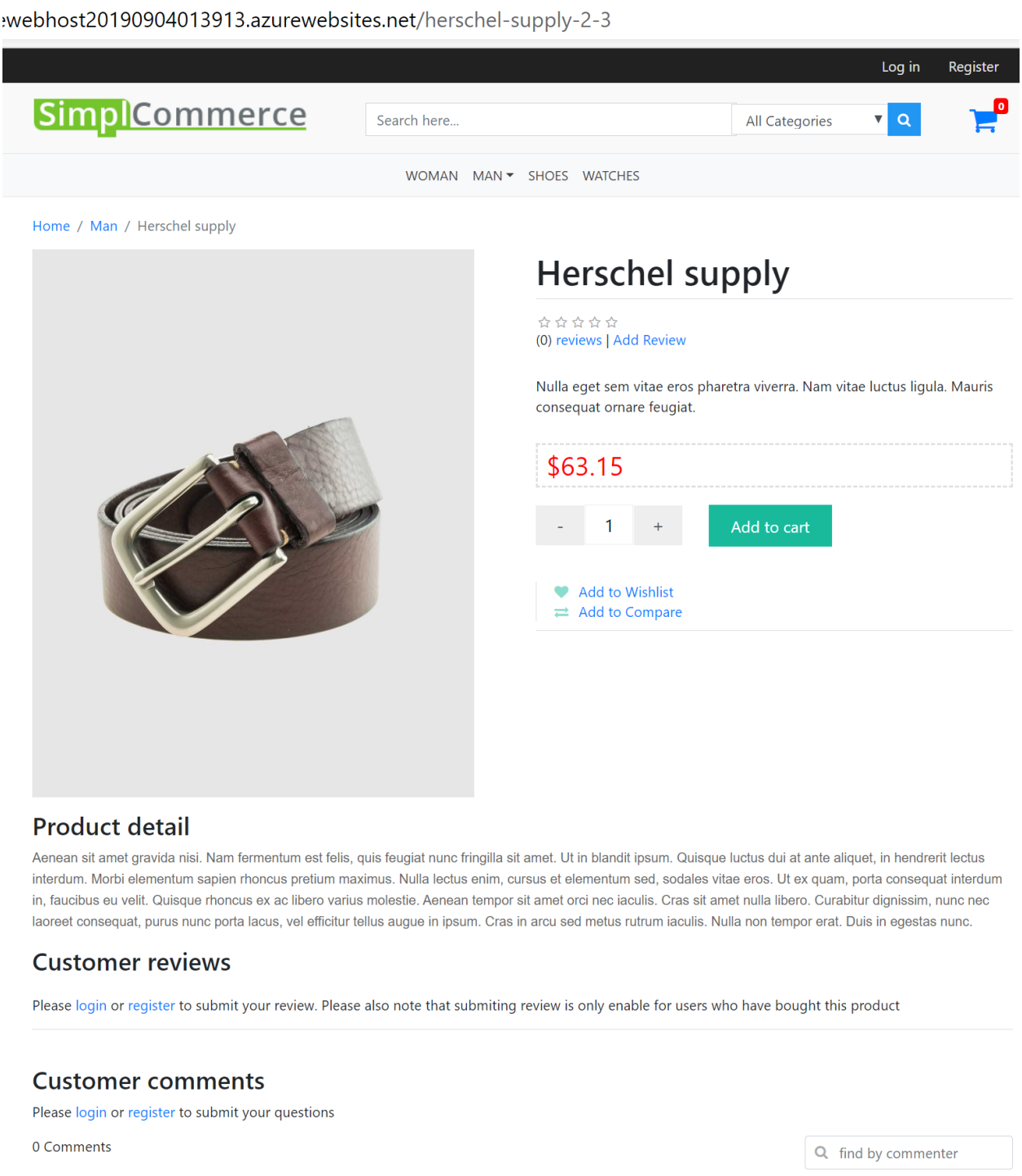
8. Confirm the settings by pressing the **Migrate** button



9. This kicks off the actual Azure Web App deployment, followed by creating and copying the content. Wait a few minutes for this process to complete.



10. **Click "Go to your website"**, which will open the newly deployed web app in the default browser.



11. This completes this lab.

Summary

In this lab, you learned how to deploy a web application from source code in Visual Studio to Azure Web Apps, as well as by using the Azure App Service Migration Assistant.