

Migrating a Two-Tier Application to Azure

A Hands-on Walkthrough of
Azure Infrastructure, Platform,
and Container Services

—
Peter De Tender

Apress®

Migrating a Two-Tier Application to Azure

**A Hands-on Walkthrough of Azure
Infrastructure, Platform,
and Container Services**

Peter De Tender

Apress®

Migrating a Two-Tier Application to Azure

Peter De Tender
Lokeren, Belgium

ISBN-13 (pbk): 978-1-4842-6436-2
<https://doi.org/10.1007/978-1-4842-6437-9>

ISBN-13 (electronic): 978-1-4842-6437-9

Copyright © 2021 by Peter De Tender

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Spandana Chatterjee
Development Editor: Matthew Moodie
Coordinating Editor: Divya Modi

Cover designed by eStudioCalamar

Cover image designed by Pixabay

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 New York Plaza, New York, NY 10004. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at <http://www.apress.com/bulk-sales>.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-6436-2. For more detailed information, please visit <http://www.apress.com/source-code>.

Printed on acid-free paper

I dedicate this book to Ivan De Rop, head teacher in my senior year in high school, for believing in my skills and passion for information technology, although I did business management studies.

Table of Contents

| | |
|---|-------------|
| About the Author | ix |
| About the Technical Reviewer | xi |
| Acknowledgments | xiii |
| Chapter 1: Introduction..... | 1 |
| Migrating a Two-Tier Application to Azure Using Different Architectures and DevOps Best Practices | 1 |
| Setting the Scene | 1 |
| Abstract and Learning Objectives | 2 |
| Technical Requirements..... | 4 |
| Azure Subscription | 4 |
| Naming Conventions..... | 5 |
| Other Requirements | 5 |
| Alternative Approach | 5 |
| Final Remarks..... | 6 |
| Chapter 2: Prerequisite Lab: Deploying Your Lab Virtual Machine | 7 |
| Prerequisite lab: Preparing your (Azure) environment | 7 |
| What You Will Learn..... | 7 |
| Time Estimate..... | 7 |
| Task 1: Deploying the lab jumpVM virtual machine using Azure Portal template deployment..... | 8 |
| Task 2: Cloning the setup scripts from GitHub | 20 |
| Summary..... | 21 |

TABLE OF CONTENTS

- Chapter 3: Lab 1: Deploying an Azure Virtual Machine Baseline Application Workload 23**
 - Lab 1: Deploying the baseline virtual machine environment using an ARM template from within Visual Studio 2019 23
 - What You Will Learn 23
 - Time Estimate 23
 - Prerequisites 24
 - Task 1: Understanding the ARM template building blocks 24
 - Task 2: Running an ARM template deployment from Visual Studio 2019..... 31
 - Summary..... 49

- Chapter 4: Lab 2: Performing Assessment of Your As-Is Situation 51**
 - Lab 2: Performing assessment of your as-is situation..... 51
 - What You Will Learn 51
 - Time Estimate 51
 - Prerequisites 51
 - Task 1: Running a SQL Server assessment using Data Migration Assistant..... 52
 - Task 2: Running a web server assessment using Azure App Service Migration Assistant.... 62
 - Summary..... 66

- Chapter 5: Lab 3: Deploying an Azure SQL Database and Migrating from SQLVM 67**
 - Lab 3: Deploying an Azure SQL database and migrating from SQLVM 67
 - What You Will Learn 67
 - Time Estimate..... 68
 - Prerequisites 68
 - Scenario Diagram 68
 - Task 1: Deploying a new Azure SQL Server instance..... 68
 - Task 2: Performing a SQL database migration from a SQL virtual machine to SQL Azure, using SQL Data Migration Assistant 78
 - Task 3 (Optional): Using SQL Server Management Studio to migrate from SQLVM to a SQL Azure instance..... 93
 - Task 4: Defining a hybrid connection from a WebVM to an Azure SQL database 102
 - Summary..... 107

| | |
|---|------------|
| Chapter 6: Lab 4: Deploying an Azure Web App and Migrating from WebVM..... | 109 |
| Lab 4: Deploying an Azure Web App and migrating from WebVM | 109 |
| What You Will Learn..... | 109 |
| Time Estimate..... | 109 |
| Prerequisites | 109 |
| Scenario Diagram..... | 110 |
| Task 1: Publish an ASP.NET project to Azure Web Apps from Within Visual Studio 2019..... | 110 |
| Task 2: Publishing the source code to Azure Web Apps..... | 117 |
| Task 3: Migrating a web application from Azure App Service Migration Assistant..... | 125 |
| Summary..... | 130 |
| Chapter 7: Lab 5: Deploying Docker and Running Azure Container Workloads.... | 131 |
| What You Will Learn | 131 |
| Time Estimate | 131 |
| Prerequisites..... | 132 |
| Scenario Diagram | 132 |
| Tasks..... | 132 |
| Task 1: Installing Docker Enterprise Edition (trial) for Windows Server 2019 on the lab jumpVM | 133 |
| Task 2: Validating and running basic Docker commands and containers | 139 |
| Task 3: Integrating Docker extension in Visual Studio Code | 149 |
| Task 4: Deploying and operating Azure Container Registry | 155 |
| Task 5: Deploying and running Azure Container Instance | 162 |
| Task 5: Running an Azure Container Instance from a Docker image in Azure Container Registry | 165 |
| Task 6: Deploying and operating Azure Web App for Containers..... | 181 |
| Summary..... | 186 |
| Chapter 8: Lab 6: Deploying and Running Azure Kubernetes Service (AKS) | 187 |
| What You Will Learn | 187 |
| Time Estimate | 187 |
| Prerequisites..... | 187 |

TABLE OF CONTENTS

- Scenario Diagram 188
- Task 1: Deploying Azure Kubernetes Service using Azure CLI 2.0 188
- Task 2: Configuring RBAC for managing Azure Kubernetes Service and ACR integration..... 193
- Task 3: Running a Docker container image from Azure Container Registry in Azure Kubernetes Service..... 196
- Summary..... 206
- Chapter 9: Lab 7: Managing and Monitoring Azure Kubernetes Service (AKS) 207**
 - What You Will Learn 207
 - Time Estimate 207
 - Task 1: Enabling container scalability in Azure Kubernetes Service (AKS) 208
 - Task 2: Monitoring Azure Kubernetes Service in Azure..... 215
 - Task 3: Managing Kubernetes from Visual Studio Code..... 225
 - Summary..... 231
- Chapter 10: Lab 8: Deploying Azure Workloads Using Azure DevOps 233**
 - What You Will Learn 233
 - Time Estimate 233
 - Prerequisites..... 234
 - Scenario Diagram 234
 - Task 1: Deploying an Azure DevOps organization 234
 - Task 2: Introduction to source control with Azure DevOps Repos..... 241
 - Task 3: Creating and deploying an Azure build pipeline for your application..... 254
 - Task 4: Building a release pipeline in Azure DevOps..... 263
 - Task 5: Creating and pushing a Docker container to ACR 279
 - Task 6: Creating a release pipeline for Docker containers from ACR..... 294
 - Task 7: Creating an Azure DevOps pipeline to deploy an ACR container to Azure Kubernetes Service (AKS) 307
 - Summary..... 316
- Index..... 317**

About the Author



Peter De Tender has more than 20 years of experience in architecting and deploying Microsoft datacenter technologies. Since early 2012, he started shifting to cloud technologies (Office 365, Intune) and quickly jumped onto the Azure platform, working as cloud solution architect and trainer, out of his own company. Since September 2019, Peter moved into an FTE role within Microsoft Corp in the prestigious Azure Technical Trainer team, providing Azure readiness workshops to larger customers and partners across the globe.

Peter was an Azure MVP for 5 years and IS a Microsoft Certified Trainer for more than 12 years and is still actively involved in the community as speaker, technical writer, and author.

You can follow Peter on Twitter @pdtit and check his technical blog, <https://www.007FFFlearning.com>.

About the Technical Reviewer



Amita Thukral is an IT professional, an NIIT degree holder, and ITIL certified. She has more than 16 years of extensive experience working with top IT organizations like Wipro Infotech, Dell India, Hughes Software Systems, and Xcad Agencies. She worked as a technical editor for Leanpub Publishing with author Peter De Tender (MVP) for a web book “Migrating a dotnetcore 2-tier application to Azure, using different architectures and DevOps best practices.”

As a service delivery manager, she has handled multiple IT instructor-led and online trainings across various global locations. As a project manager, she was responsible for running cloud computing projects, like Azure and Dynamics 365, and prepared comprehensive action plans, including resources, timeframes, and budgets for projects. She worked on updating, reviewing, and building documentation and content of the lab guides and ebooks for several cloud-based technical projects. She has performed coordinating tasks like planning and scheduling, along with administrative duties like maintaining project documentation, database management, and collaborating with clients and internal teams to deliver results. She ensured that all projects were completed on time and within budget and met high-quality standards.

Acknowledgments

After writing seven technical books, it's hard to come up with original thank-you words. Anyone reading this book knows this is a work of time, dedication, and passion for technology, as well as a passion for sharing knowledge. I am fortunate enough to have a wife supporting me in this. But I'm no longer allowed to thank her (her own words), as sharing knowledge and helping people is what makes me who I am.

That said, I owe a big thanks to Spandana Chatterjee and Divya Modi from Apress, who picked up my "Azure hands-on labs" self-published material from Leanpub and offered to take over the content and publish it through Apress. This was the best opportunity to update the technical content, make it current, and add new topics to the exercises. And your audience reach-out is much broader than what I could ever get myself, so you help in spreading the Azure knowledge.

I'd also like to thank my technical reviewer, Amita Thukral – my faraway friend from India, always eager to help where she can and at the same time living the "continuous learning" life. You are professional, have an amazing drive for details, and are overall a lovely person to work with.

And Wim Matthyssen, community buddy and fellow Azure expert, thanks for jumping in last minute to give your technical blessing on the flow, wording, and lab scenarios and overall validate them.

Both of you pushed up the level of quality.

CHAPTER 1

Introduction

Migrating a Two-Tier Application to Azure Using Different Architectures and DevOps Best Practices

Setting the Scene

You are part of an organization that is running an e-commerce platform application, at present using Windows Server on-premises infrastructure, based on a virtual Windows Server 2012 R2 web server running Internet Information Services (IIS) and a second Windows Server 2012 R2 virtual machine (VM) running Microsoft SQL Server 2014 database services.

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

- Infrastructure as a Service (IAAS), using Azure Virtual Machines
- Platform as a Service (PAAS), using Azure Web Apps and Azure SQL
- Containers as a Service (CAAS), using Azure Container Instance (ACI) and Azure Kubernetes Service (AKS)

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, together with other capabilities from Azure DevOps, can assist in optimizing the deployment as well as optimizing the 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 over migration to performing day-to-day operations.

Abstract and Learning Objectives

This book enables anyone to learn, understand, and build a Proof of Concept, by performing a platform migration of a two-tiered application workload 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 Service (AKS).

The focus of the book is having a true hands-on lab experience, by going through the following exercises and tasks:

- Deploying your “lab virtual machine”
- Deploying a two-tier Azure Virtual Machine (web server and SQL database server) using Infrastructure as Code (IAC) concepts with ARM (Azure Resource Manager) template automation in Visual Studio 2019
- Performing a proper assessment of the as-is WebVM and SQLVM infrastructure using Microsoft assessment tools
- Migrating a SQL Server 2014 database to Azure SQL PaaS (lift and shift)
- Migrating a .NET Core web application to Azure Web Apps (lift and shift)
- Containerizing a .NET Core web application using Docker and pushing to Azure Container Registry (ACR)
- Running a containerized application in Azure Container Instance (ACI) and Azure Web App for Containers
- Running a containerized application in Azure Kubernetes Service (AKS)

- Deploying Azure DevOps and building a CI/CD pipeline for the sample e-commerce application
- Managing and monitoring Azure Kubernetes Service (AKS) and other Azure Monitor capabilities

Starting from an (optional but highly recommended for consistency) ARM template-based deployment of a lab virtual machine, readers get introduced to the basics of automating Azure resource deployments using Visual Studio and Azure Resource Manager (ARM) templates, together with additional Infrastructure as Code concepts like Custom Script Extension and PowerShell Desired State Configuration (DSC).

Next, readers 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, readers learn about Microsoft SQL Server database migration to Azure SQL PAAS, as well as deploying and migrating web applications to Azure Web Apps.

After these foundational platform components, the following chapters 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 Web App for Containers, as well as how to enable container orchestration and cloud scale using Azure Kubernetes Service (AKS).

In the last part of the book, readers get introduced to Azure DevOps, the Microsoft application lifecycle environment, helping in building a CI/CD pipeline to publish workloads using the DevOps principles and concepts, showing the integration with the rest of the already-touched-on Azure services like Azure Web Apps and Azure Kubernetes Service (AKS), closing the exercises with a chapter on Azure monitoring and operations and what tools Azure has available to assist your IT teams in this challenge.

Note The Proof of Concept lab scenario is built in such a way that each lab exercise is building on top of the previous lab exercise in sequence. Given the dependencies across different labs, make sure you finish each lab exercise successfully, before continuing on to the next lab.

Technical Requirements

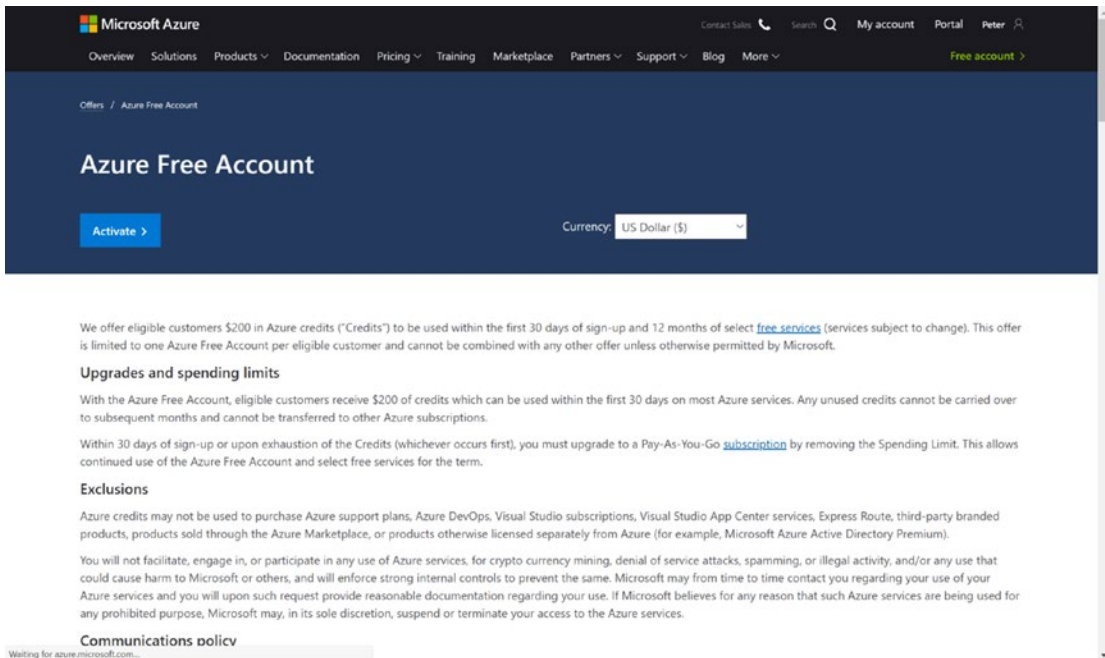
Before being able to perform the hands-on tasks in this book, make sure you meet each of the technical requirements:

- Azure subscription with full administrative permissions
- Naming conventions

Azure Subscription

Make sure you have (full administrative) access to an Azure subscription, allowing you to deploy the different Azure resources being used throughout the exercises. You can use an Azure free or trial subscription or use any paid subscription.

Signing up for a free/trial subscription can be done from here: <https://signup.azure.com/signup?offer=ms-azr-0044p&appId=102&l=en-gb&correlationId=37037FE60CF76B40251371B40DDF6AB9>



If you go through all exercises, estimate an average consumption of 20–30 USD, assuming you shut down or delete the resources that are no longer in use or required.

Naming Conventions

Important Most Azure resources require unique names. Throughout the lab steps, we will identify the naming convention for the given resources as “[SUFFIX]” as part of resource names. You should replace this with a unique string, e.g., your own initials, guaranteeing those resources get uniquely named and not blocking a successful deployment.

Other Requirements

Readers need a local client admin machine, running a recent Operating System, allowing them to

- Browse to <https://portal.azure.com> from a recent browser.
- Establish a secured Remote Desktop (RDP) session to a lab jumpVM running Windows Server 2019.

Alternative Approach

Where the lab scenario assumes all exercises will be performed from within a lab jumpVM (see Chapter 2 on how to get started with this deployment), readers could also execute (most, if not all) steps from their local client machine, if that is what they prefer.

The following tools are being used throughout the lab exercises:

- Visual Studio 2019 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)
- SimplCommerce Open Source e-commerce platform example (<http://www.simplcommerce.com>)

Note 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 in using these tools.

Final Remarks

Due to the continuously evolving nature of Azure, Azure services, the Azure Portal, and other tools we will be using for the exercises, it might be that some screenshots or wordings do not match what you will see on your end. We apologize for this already, although there isn't much we can do about it. If the differences are too many, it would be almost impossible to execute the exercises. Please have a look at our GitHub repository <http://www.apress.com/source-code> for any updates and errata.

We hope you enjoy the different exercises, learn from them, and find them useful in your day-to-day job or journey in which you explore Azure capabilities. Do not hesitate reaching out at peter@pdtit.be or [@pdtit](https://twitter.com/pdtit) (Twitter) in case you have any questions. We are here to help you making this a successful learning path.

CHAPTER 2

Prerequisite Lab: Deploying Your Lab Virtual Machine

Prerequisite lab: Preparing your (Azure) environment

What You Will Learn

In this first lab, you prepare the baseline for executing all hands-on lab exercises:

- Log on to your Azure subscription.
- Deploy the lab jumpVM within your Azure subscription.
- Download the required source files from GitHub to the lab jumpVM.

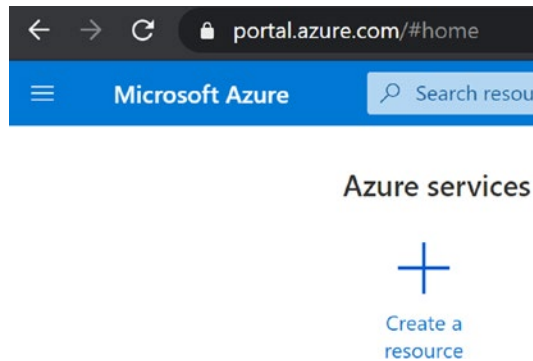
Time Estimate

This lab is estimated to take **45 min**, assuming your Azure subscription is already available.

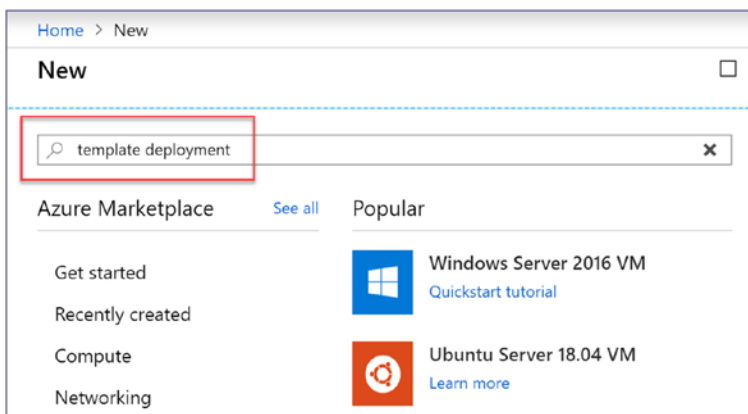
Task 1: Deploying the lab jumpVM virtual machine using Azure Portal template deployment

In this task, you start deploying the “lab jumpVM” virtual machine in your Azure environment. This machine becomes the starting point for all future exercises, as it has most required tools already installed. The deployment is based on an ARM (Azure Resource Manager) template in a publicly shared GitHub repository.

1. Once you are logged on to your Azure subscription, select **Create a Resource**.



2. In the Search Azure Marketplace field, type “template deployment”.



3. And select **Template deployment (deploy using custom templates)** from the list of Marketplace results, followed by clicking the **Create** button.

[Home](#) > [New](#) >

Template deployment (deploy using custom templates)

Microsoft



Template deployment (deploy using custom templates)

Microsoft

Create

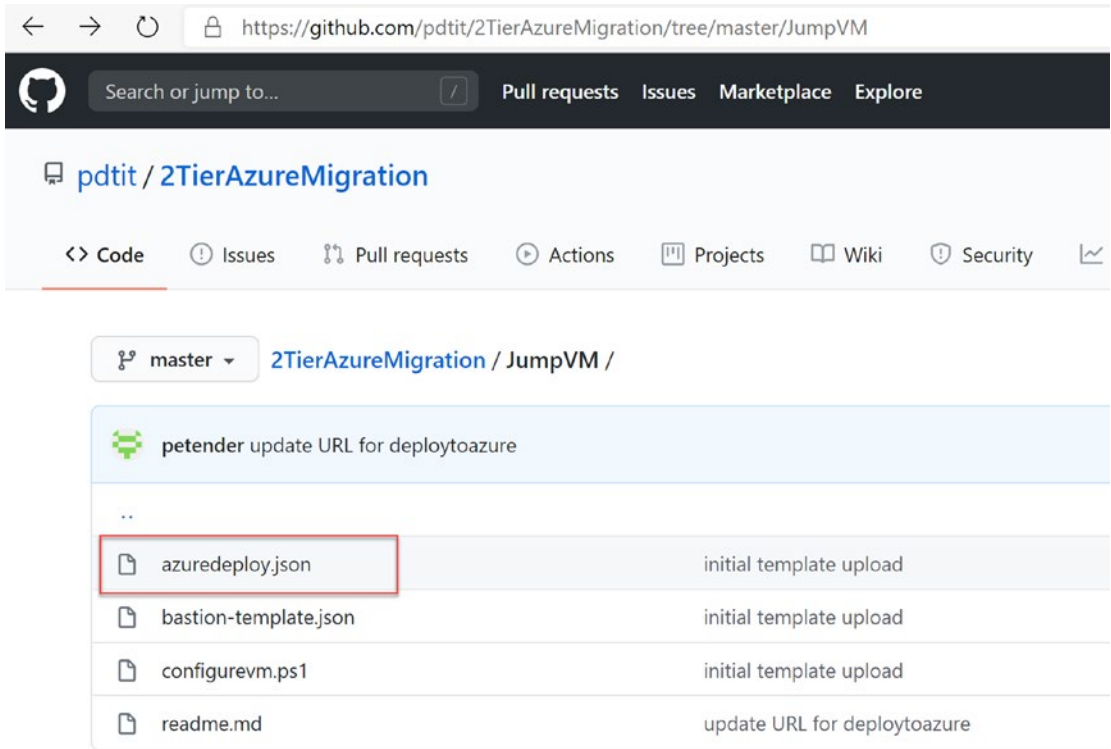
 Save for later

4. This opens the Custom deployment blade. Here, select “Build your own template in the editor.”

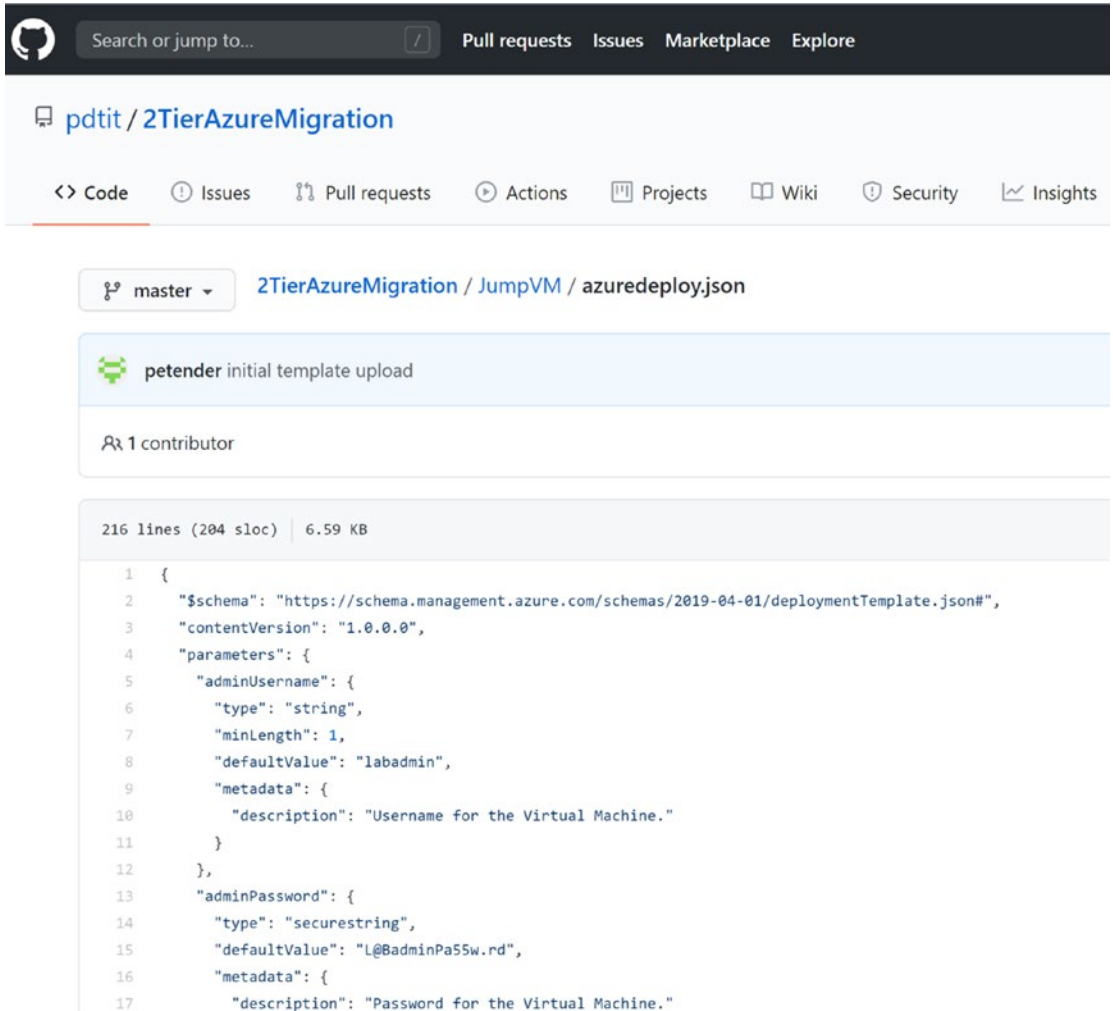
The screenshot shows the 'Custom deployment' blade in the Azure portal. The breadcrumb navigation at the top reads: [Home](#) > [New](#) > [Marketplace](#) > [Everything](#) > [Template deployment](#) > [Custom deployment](#). The main heading is 'Custom deployment' with the subtitle 'Deploy from a custom template'. Below this, there is a section 'Learn about template deployment' with two links: 'Read the docs' and 'Build your own template in the editor'. The 'Build your own template in the editor' link is highlighted with a red rectangular box. Underneath, there is a section 'Common templates' with four options: 'Create a Linux virtual machine', 'Create a Windows virtual machine', 'Create a web app', and 'Create a SQL database'. At the bottom, there is a section 'Load a GitHub quickstart template' with a dropdown menu labeled 'Select a template (disclaimer)' and a search input field containing the text 'Type to start filtering...'.

5. First, from a **second tab** in your browser window, go to the following URL on GitHub, browsing to the source files repository for this lab, specifically the JumpVM folder:

<http://www.apress.com/source-code>.



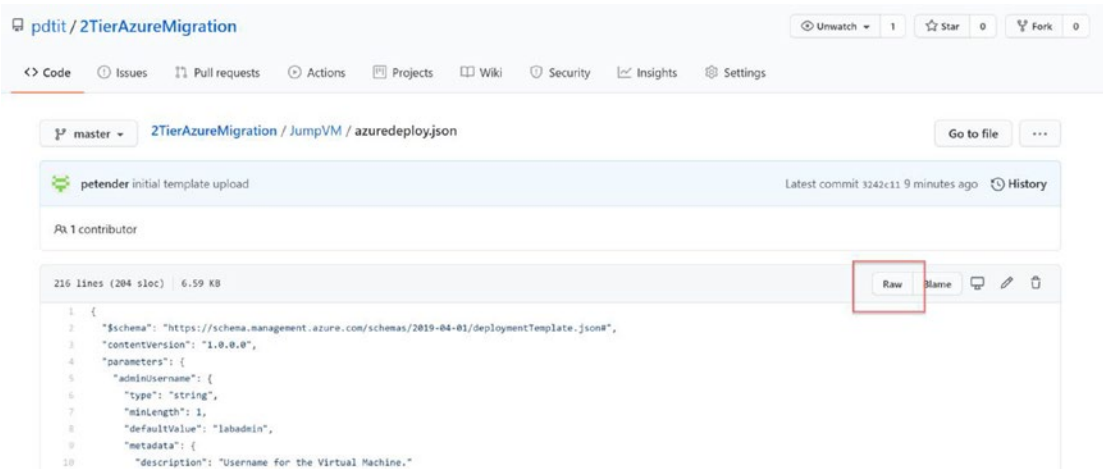
6. **Select** the `azuredeploy.json` object in there. This exposes the details of the actual JSON deployment file.



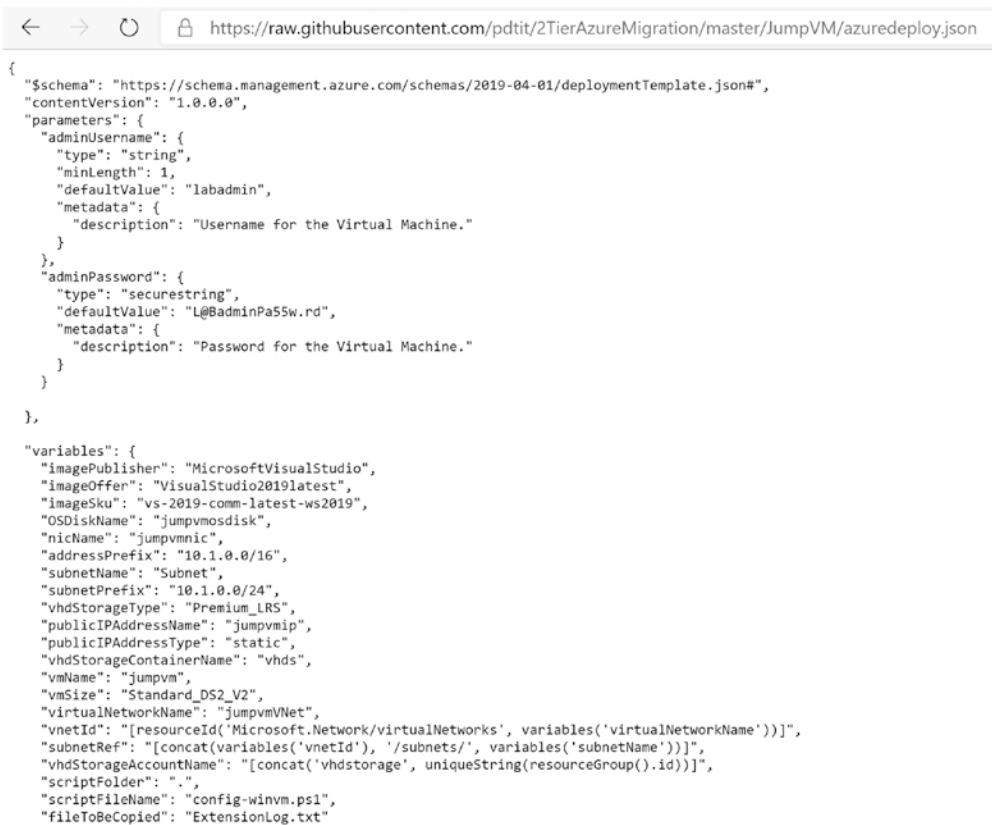
The screenshot shows the GitHub interface for the repository `pdtit / 2TierAzureMigration`. The file `JumpVM / azuredeploy.json` is selected, showing a commit by `petender` with the message "petender initial template upload". The file is 6.59 KB and contains 216 lines of JSON code. The code defines a deployment template with parameters for `adminUsername` and `adminPassword`.

```
1 {
2   "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
3   "contentVersion": "1.0.0.0",
4   "parameters": {
5     "adminUsername": {
6       "type": "string",
7       "minLength": 1,
8       "defaultValue": "labadmin",
9       "metadata": {
10        "description": "Username for the Virtual Machine."
11      }
12    },
13    "adminPassword": {
14      "type": "securestring",
15      "defaultValue": "L@BadminPa55w.rd",
16      "metadata": {
17        "description": "Password for the Virtual Machine."
18      }
19    }
20  }
21 }
```

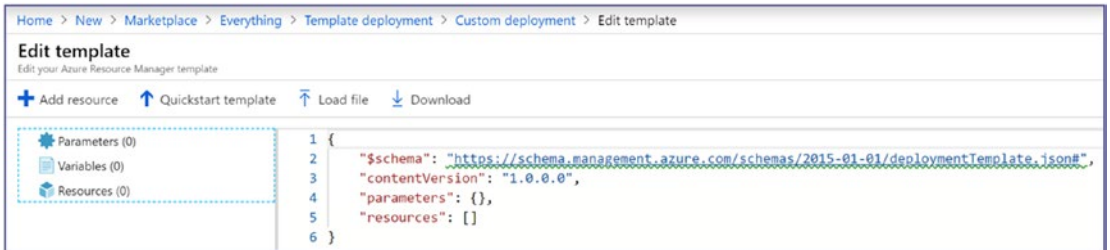
7. Click the **Raw** button, to open the actual file in your browser.



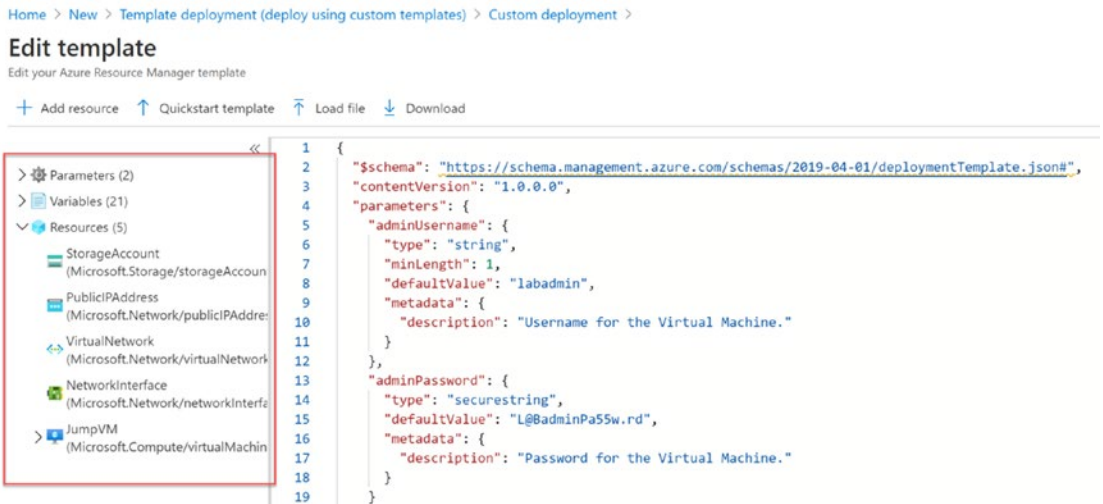
8. Your browser should show the content as follows:



9. Here, **select all lines in the JSON file**, and copy its content to the clipboard.
10. **Go back to the Azure Portal**. From “the **edit template**” blade, **remove the first six lines of code** you see in there, and **paste in the JSON content from the clipboard**.



11. “The edit template” blade should recognize the content of the JSON file, showing the details in the JSON Outline on the left.



12. **Click the Save button.**


13. This **redirects** you back to the Custom deployment blade, from where you will **execute** the actual template deployment, filling in the required fields as follows:
 - **Subscription: Your Azure subscription**
 - **Resource group: Create New/[SUFFIX]-JumpVMRG**
 - **Location: Your closest by Azure region**
 - **Admin Username: labadmin** (this information is picked up from the ARM template; although you could change this, we recommend you to not do so for consistency with the lab guide instructions and avoiding any errors during later deployment steps)
 - **Admin Password: L@BadminPa55w.rd** (this information is picked up from the ARM template; although you could change this, we recommend you to not do so for consistency with the lab guide instructions and avoiding any errors during later deployment steps)

[Home](#) > [New](#) > [Template deployment \(deploy using custom templates\)](#) >

Custom deployment

Deploy from a custom template

TEMPLATE

 Customized template
5 resources

 Edit template

 Edit paramet...

 Learn more

BASICS

Subscription *

Resource group *

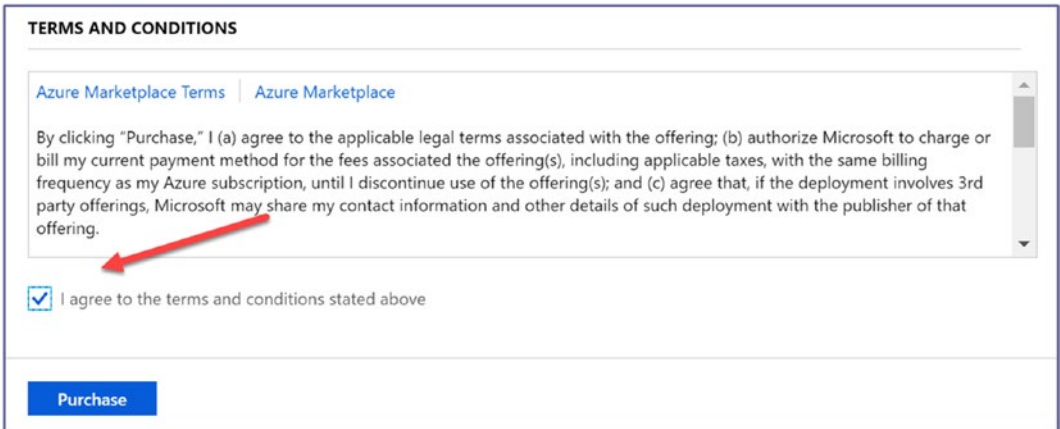
Location *

SETTINGS

Admin Username ⓘ

Admin Password ⓘ

- When all fields have been completed, scroll down in the blade. Under the Terms and Conditions section, **check “I agree to the terms and conditions stated above,”** and **click the Purchase** button.



TERMS AND CONDITIONS

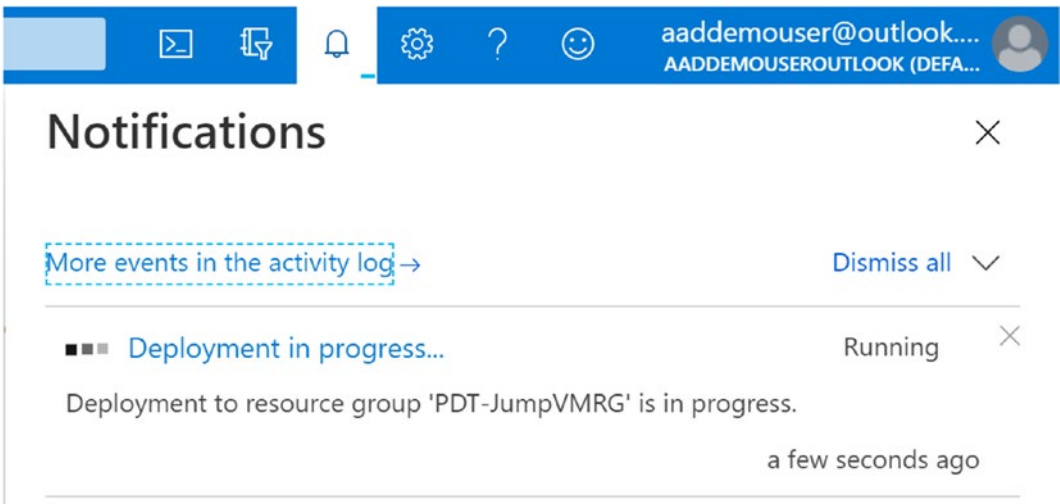
[Azure Marketplace Terms](#) | [Azure Marketplace](#)

By clicking "Purchase," I (a) agree to the applicable legal terms associated with the offering; (b) authorize Microsoft to charge or bill my current payment method for the fees associated the offering(s), including applicable taxes, with the same billing frequency as my Azure subscription, until I discontinue use of the offering(s); and (c) agree that, if the deployment involves 3rd party offerings, Microsoft may share my contact information and other details of such deployment with the publisher of that offering.

I agree to the terms and conditions stated above

Purchase

- This sets off the actual **Azure resource deployment process**. From the **Notifications** area, you can get update information about the deployment.



Notifications

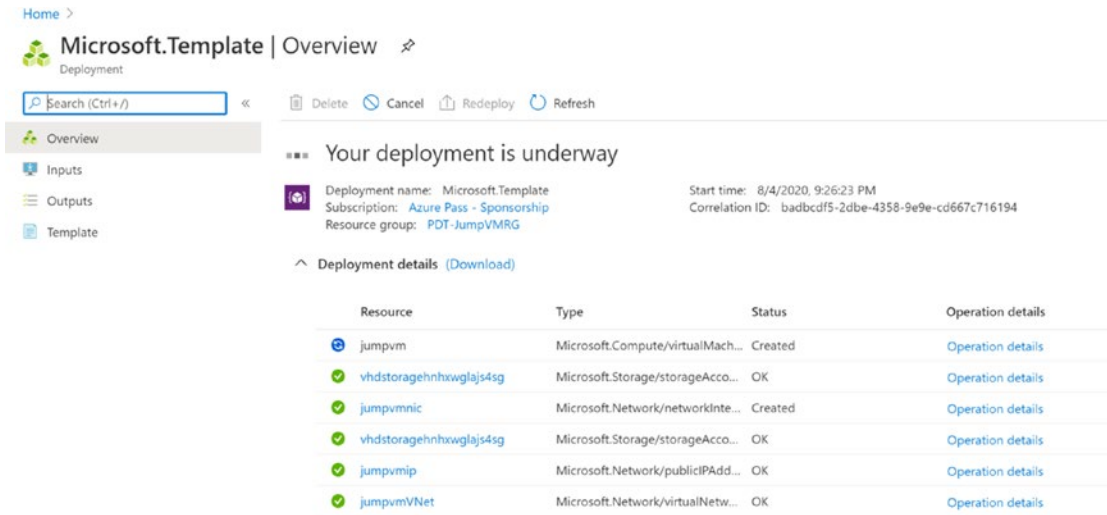
[More events in the activity log](#) → Dismiss all

■ ■ ■ **Deployment in progress...** Running

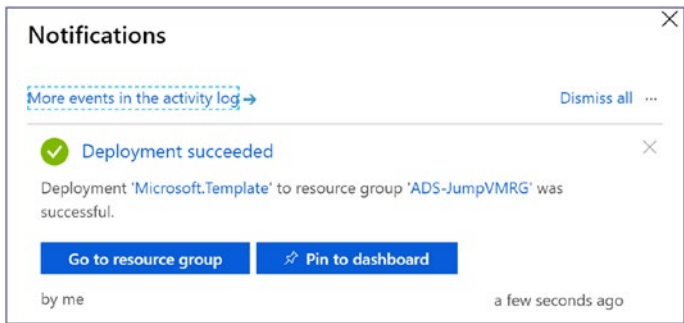
Deployment to resource group 'PDT-JumpVMRG' is in progress.

a few seconds ago

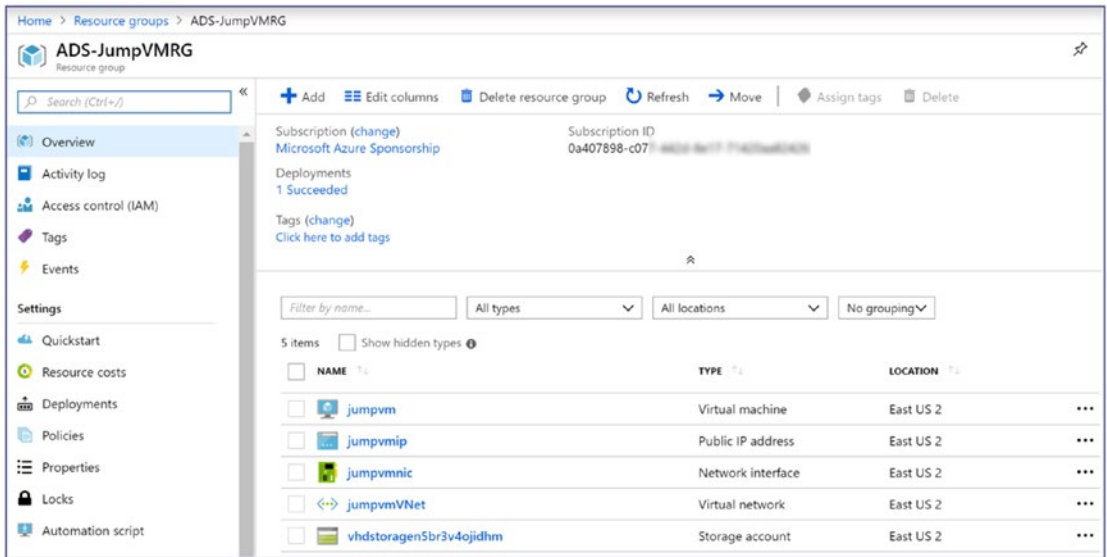
- 16. If you click “Deployment in progress...,” you will get redirected to the Microsoft.Template Overview blade, showing you the details of each Azure resource getting deployed.



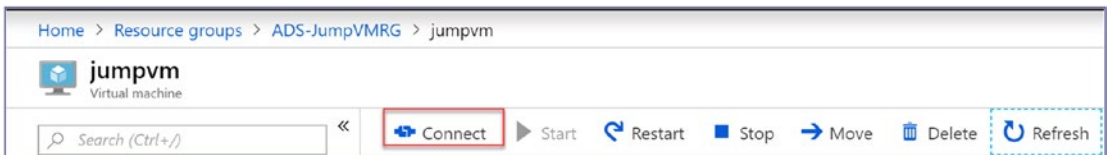
- 17. Wait for the deployment to complete successfully. **Note this could take up to 25–30 minutes, because of the custom scripts we run during the installation process...**, which you can see from this detailed view or from the Notifications area.



- 18. From the notification message, **click** “Go to resource group.” (If you already closed the notification message, from the Azure Portal navigation menu to the left, select Resource groups.)



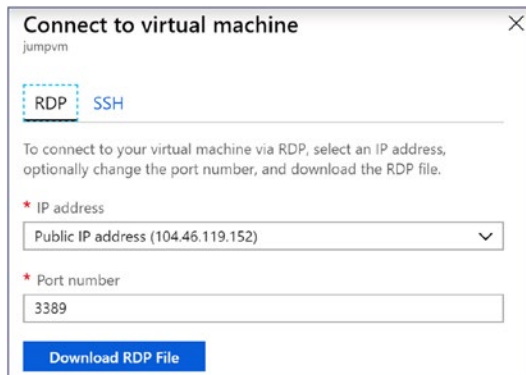
19. Click the **jumpvm Azure Virtual Machine** resource. This redirects you to the detailed blade for the jumpvm resource. Here, click the **Connect** button.



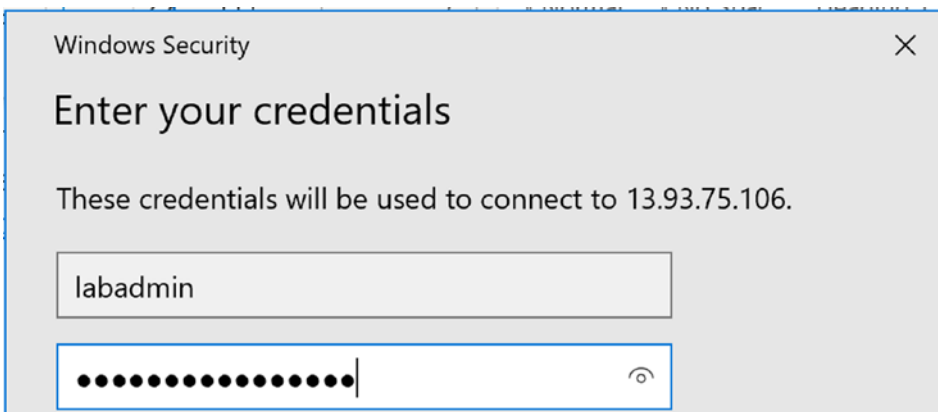
Note Because the VM is linked to a “basic” public IP address resource, all incoming TCPIP traffic is allowed. Therefore, incoming RDP is just working. In a real-life scenario, this VM would be configured with Network Security Group (NSG) rules, only allowing specific traffic.

20. From the **Connect to virtual machine** blade, notice the **public IP address and port 3389**. This allows you to establish an RDP session to the Azure VM. Do this by **clicking the Download RDP File** button.

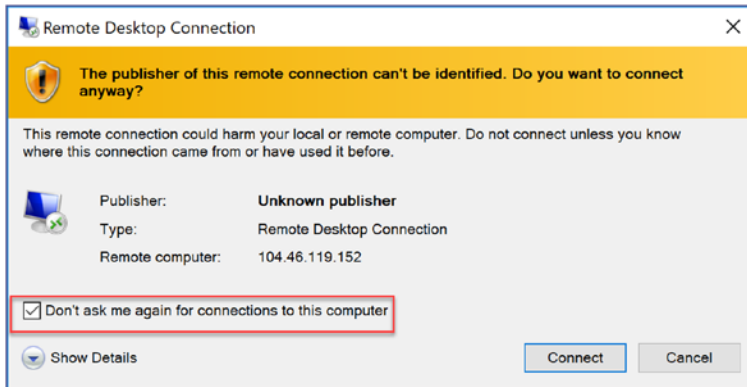
(**Note:** If your local network blocks direct RDP to Azure VMs, consider having a look at Azure Bastion, an Azure service performing HTML5 browser-based routing to RDP or SSH-enabled machines. Specifically for this JumpVM, we offer an ARM template in the same GitHub repo as the JumpVM: <https://github.com/pdtit/2TierAzureMigration/blob/master/JumpVM/bastion-template.json>.



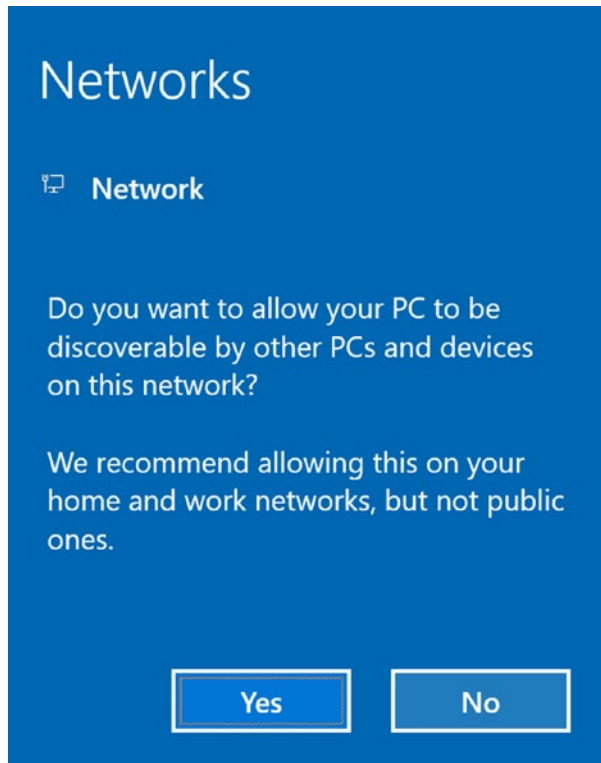
- 21. **Open** the downloaded RDP file; You are prompted for your credentials in the next step, provide the **VM administrator name (labadmin) and its password (L@BadminPa55w.rd), which are the default.**



22. From the appearing popup window, set the flag to “Don’t ask me again for connections to this computer.”



23. Your Remote Desktop session to this Azure VM gets established.
24. A popup message will appear, asking if you want to allow network discovery; close this popup using the **No** button.

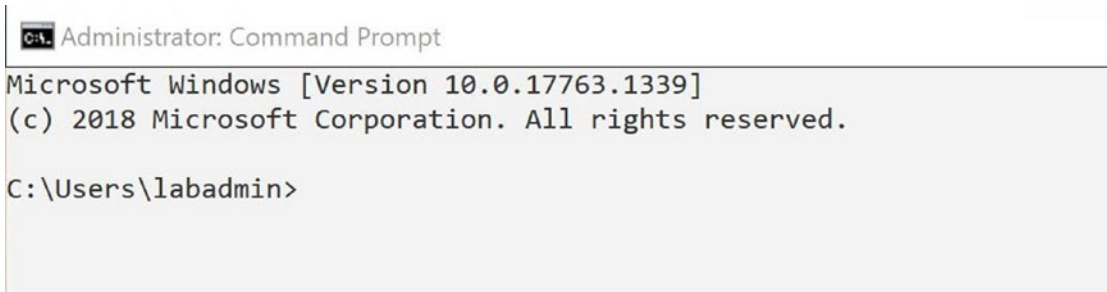


25. Next, “**Server Manager**” will open automatically. Close this for now. You will **arrive at the desktop**.

Task 2: Cloning the setup scripts from GitHub

In this task, you run Git command-line steps, to clone the necessary source files from GitHub to your lab jumpVM.

1. From the lab jumpVM, open a **command prompt**.

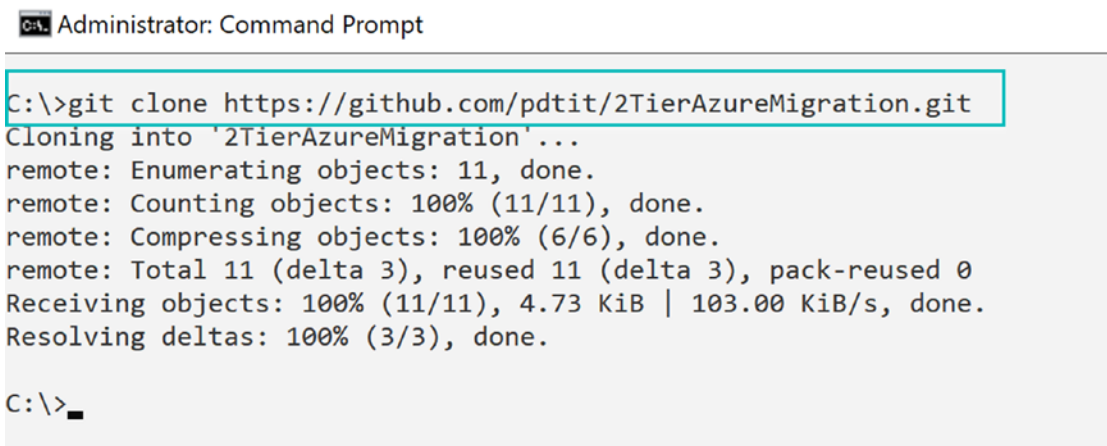


```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.17763.1339]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\labadmin>
```

2. Run the following command:

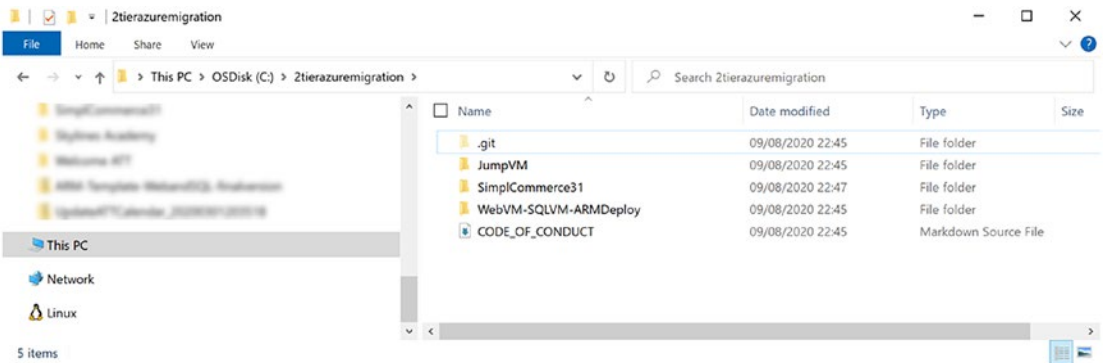
```
git clone http://www.apress.com/source-code
```



```
Administrator: Command Prompt
C:\>git clone https://github.com/pdtit/2TierAzureMigration.git
Cloning into '2TierAzureMigration'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 11 (delta 3), reused 11 (delta 3), pack-reused 0
Receiving objects: 100% (11/11), 4.73 KiB | 103.00 KiB/s, done.
Resolving deltas: 100% (3/3), done.

C:\>_
```

3. This downloads all lab-related source files to the C drive of the JumpVM, into the **2TierAzureMigration** folder.



Summary

This completes this prerequisite task, in which you deployed a Windows 2019 Azure VM as Jump server, by using Azure Resource Manager template-based deployment.

You will use this JumpVM for all future exercises requiring “tools” like Visual Studio, Docker, SQL Server Management Studio, and so on.

CHAPTER 3

Lab 1: Deploying an Azure Virtual Machine Baseline Application Workload

Lab 1: Deploying the baseline virtual machine environment using an ARM template from within Visual Studio 2019

What You Will Learn

In this task, you are guided through the definition of an ARM template, which is used to deploy the baseline virtual machine WebVM and SQLVM topology you need in the next lab. After you understand the core building blocks within the template, you run the actual template deployment from within Visual Studio 2019.

Time Estimate

This lab is estimated to take **60 min**, assuming your Azure subscription is already available.

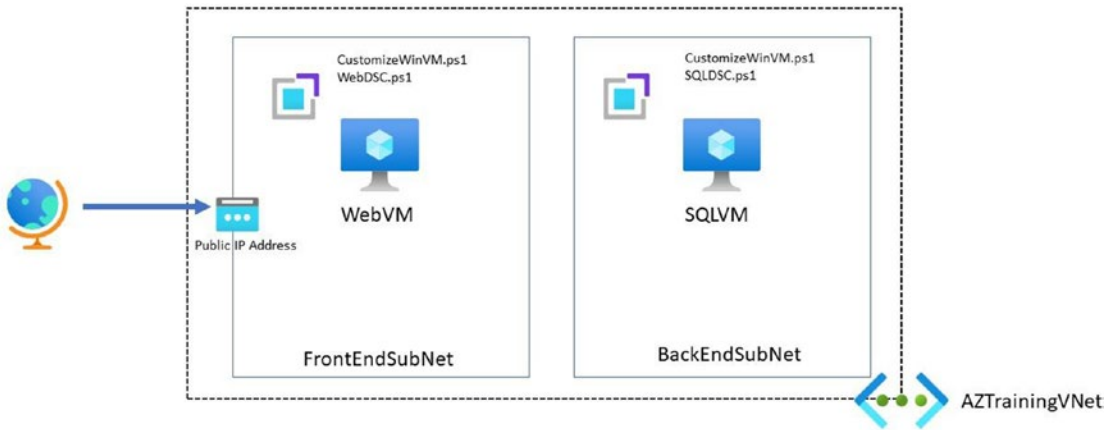
Prerequisites

Note The assumption is this lab will be performed from within the lab jumpVM, unless you choose to use your own administrative workstation for this. See Chapter 2 for instructions on how to deploy this VM if needed.

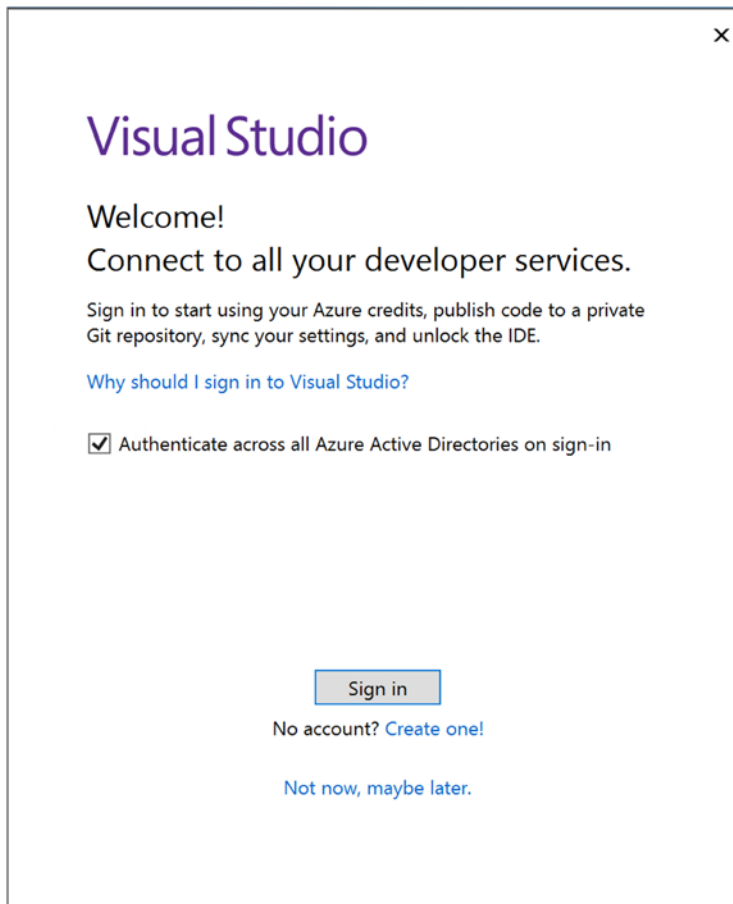
Task 1: Understanding the ARM template building blocks

The focus of this first task is becoming familiar with the baseline VM deployment for future labs, using ARM template building blocks. As part of Infrastructure as Code (IAC), ARM templates can be used to automate the deployment and configuration of Azure-running resources. Out of this template, you deploy the following Azure resources:

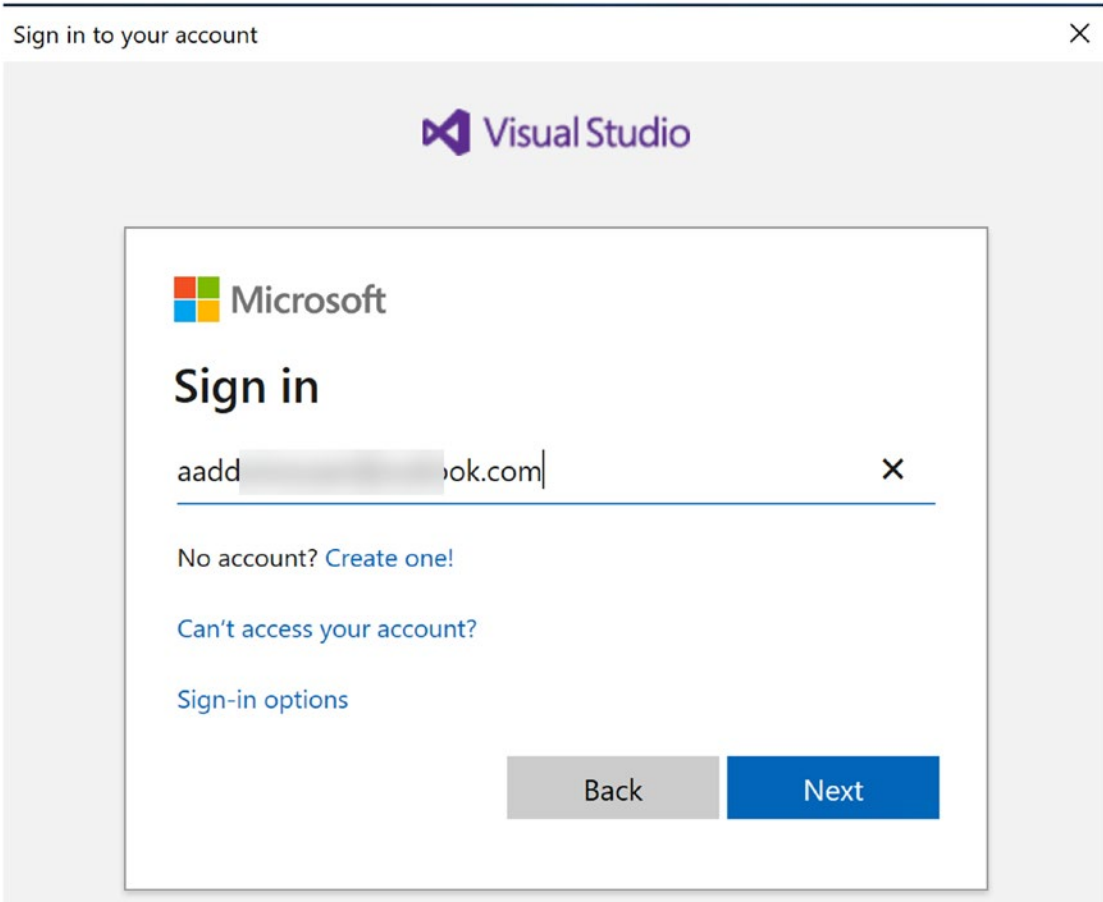
- Azure Virtual Network “AzTrainingVNET,” with two subnets
- WebVM virtual machine running IIS on Windows Server 2012 R2:
 - Azure resources themselves
 - WebDSC.ps1, as part of PowerShell DSC VM Extension
 - Customize-winVM.ps1, as part of Custom Script Extension
- SQLVM virtual machine running SQL Server 2014 on Windows Server 2012 R2:
 - Azure resources themselves
 - SQLDSC.ps1, as part of PowerShell DSC VM Extension
 - Customize-winVM.ps1, as part of Custom Script Extension



1. From the lab JumpVM desktop, **launch Visual Studio 2019**. You are prompted with a Welcome popup to authenticate.



2. **Click the Sign in** button, which will open the Microsoft “Sign in to your account” window; provide your Azure admin credentials here.



3. **Wait** for Visual Studio 2019 to launch. **Select** a theme of choice for the layout of Visual Studio 2019.

×

Visual Studio

Hello, aaddemouser@outlook.com



aaddemouser@outlook.com

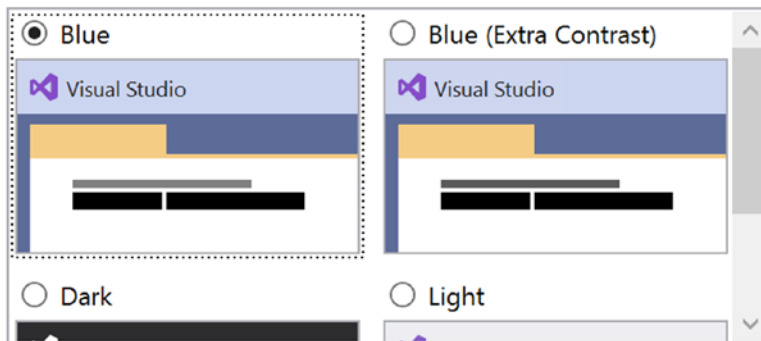
[View your Visual Studio profile](#)

Start with a familiar environment

Development Settings:

General

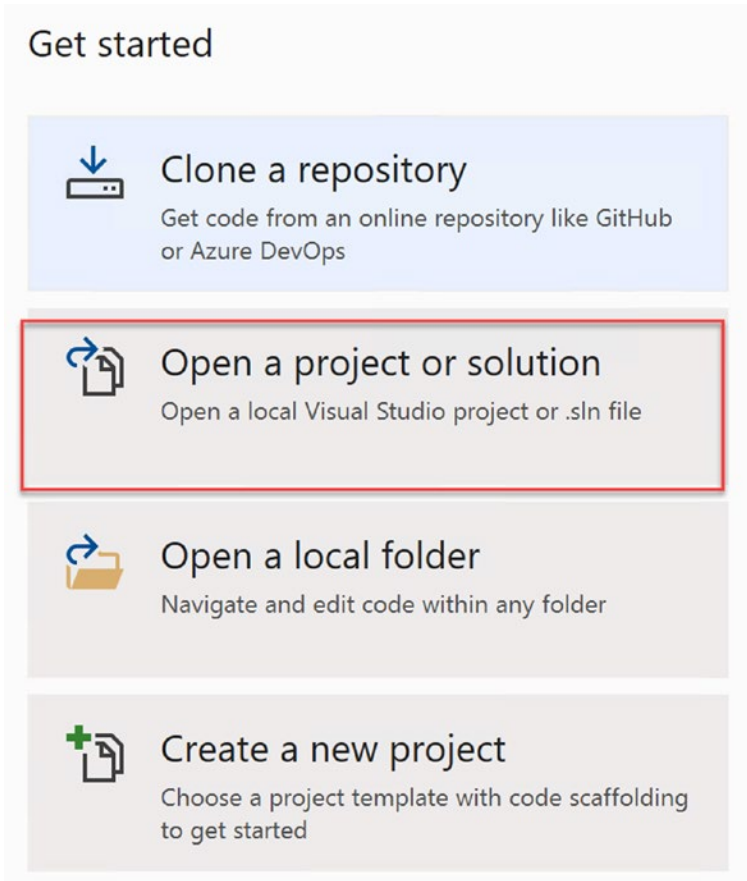
Choose your color theme



You can always change these settings later.

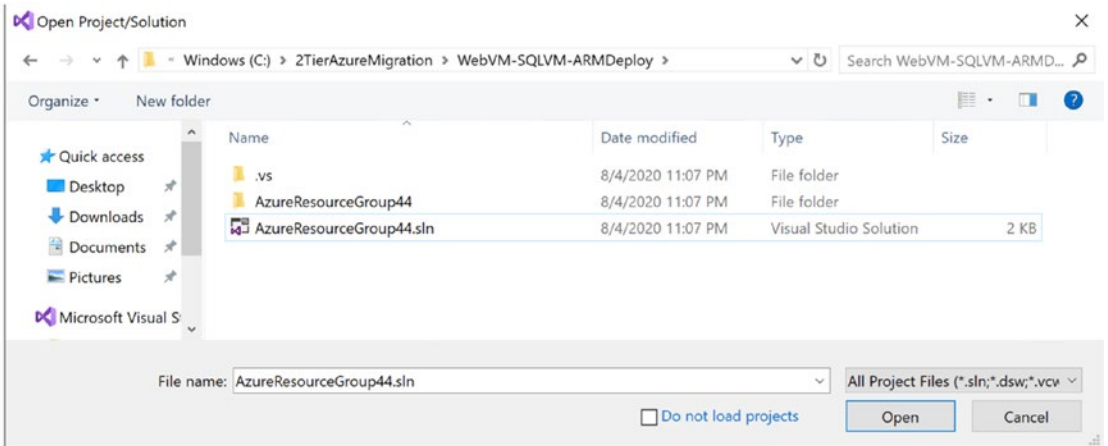
Start Visual Studio

4. **Continue** to Visual Studio by **clicking the “Start Visual Studio”** button.
5. **From the “Get started” window, select “Open a project or solution.”**

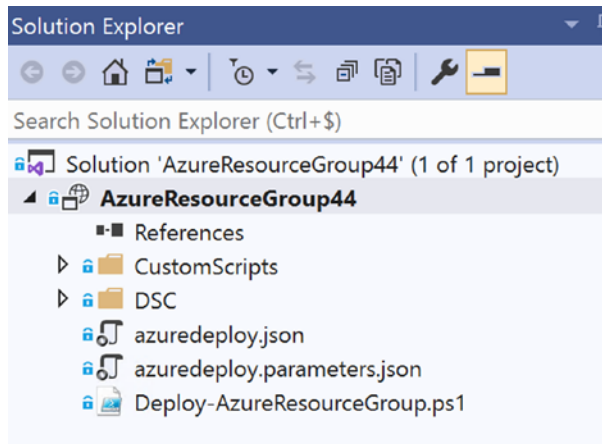


6. **Browse** to the 2TierAzureMigration folder on the JumpVM, and select the **WebVM-SQLVM-ARMDeploy** folder. From here, **select the AzureResourceGroup44.sln** file. **Click Open.**

Note If you don't have this source files folder, see “Task 2” in Chapter 2 to get the files.



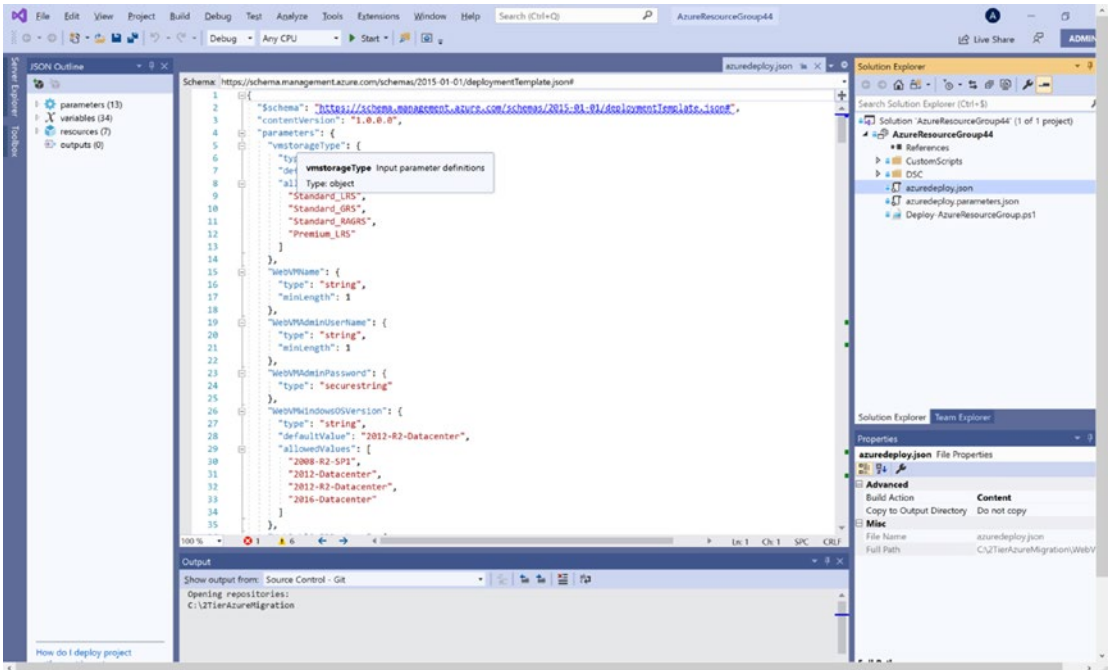
7. Make yourself familiar with the different files and folders in this project, using the **Solution Explorer** view.



8. In short, these files are doing the following:

| File | Purpose |
|---|---|
| Azuredeploy.json | The actual ARM template deployment file, which creates the different Azure resources for both WebVM and SQLVM infrastructure. |
| Azuredeploy.parameters.json | The ARM template parameters file. |
| \\CustomScripts\ Customize-WinVM.ps1 | A PowerShell script, containing specific settings that get applied to both VMs using PowerShell. |
| DSC\SQLDSC.ps1 | A PowerShell script that is used to customize the installation and configuration of SQL Server on the SQLVM: <ul style="list-style-type: none"> – Format disks. – Install SQL Server 2017 + mgmt. tools. – Download simplcommerce.bak from Azure Storage. – Run SQL database restore. |
| DSC\WebDSC.ps1 | A PowerShell script that is used to customize the installation and configuration of IIS web server on the WebVM: <ul style="list-style-type: none"> – Install IIS core components + mgmt. tools. – Install .NET framework 4.5. – Run silent install of the dotnetcore modules. |
| Deploy-AzureResourceGroup.ps1 | A PowerShell script that is used by VS2017 to run the actual deployment of the ARM template. |

9. **Select** the file **azuredeploy.json** to **open it**. This will load the details in a separate window, showing the JSON Outline for this ARM template.



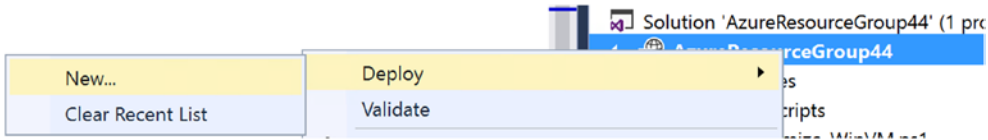
Note If the JSON Outline view is not visible, open it from the top menu. Click **View** ► **Other Windows** ► **JSON Outline**, to open the JSON viewer.

10. **Read** through the different files, to become familiar with the actual Azure resources getting deployed and the core settings used for this (VNET, subnets) as this will help in understanding the base VM landscape of our workload.

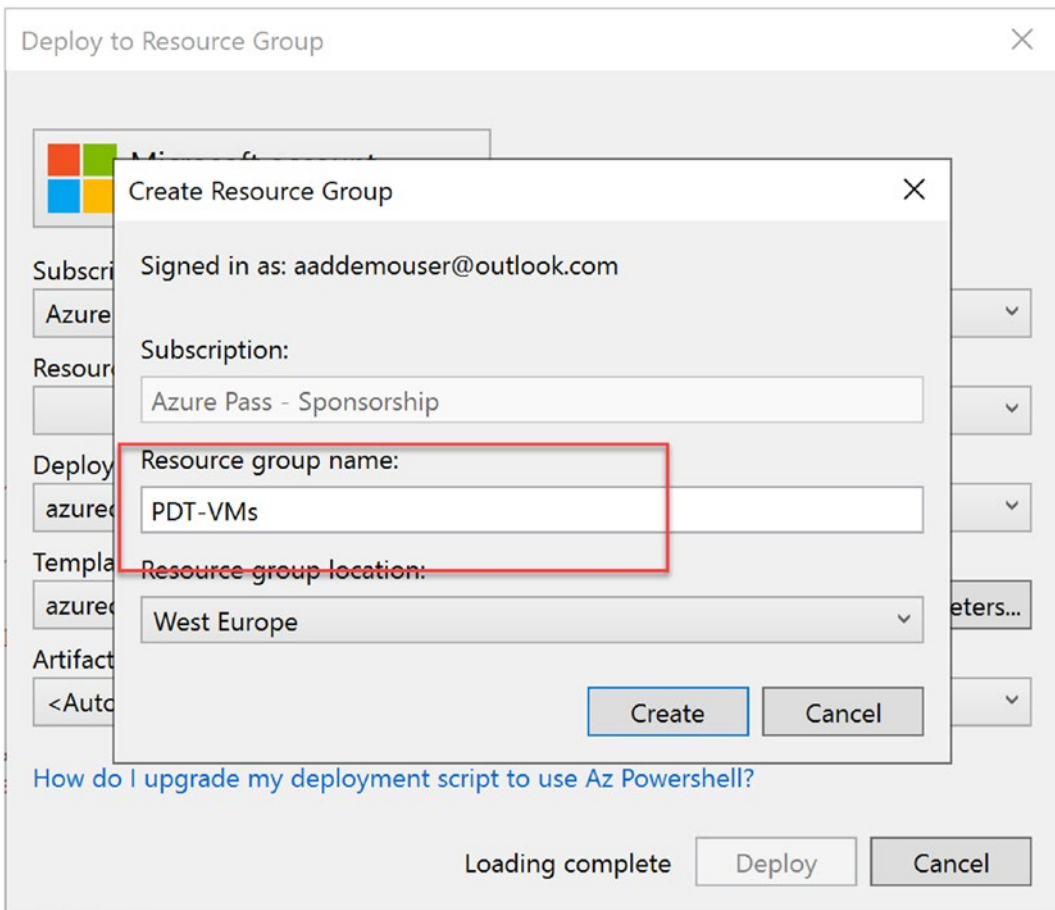
Task 2: Running an ARM template deployment from Visual Studio 2019

In this task, you start deploying the “lab jumpVM” virtual machine in your Azure environment. This machine becomes the starting point for all future exercises, as it has most required tools already.

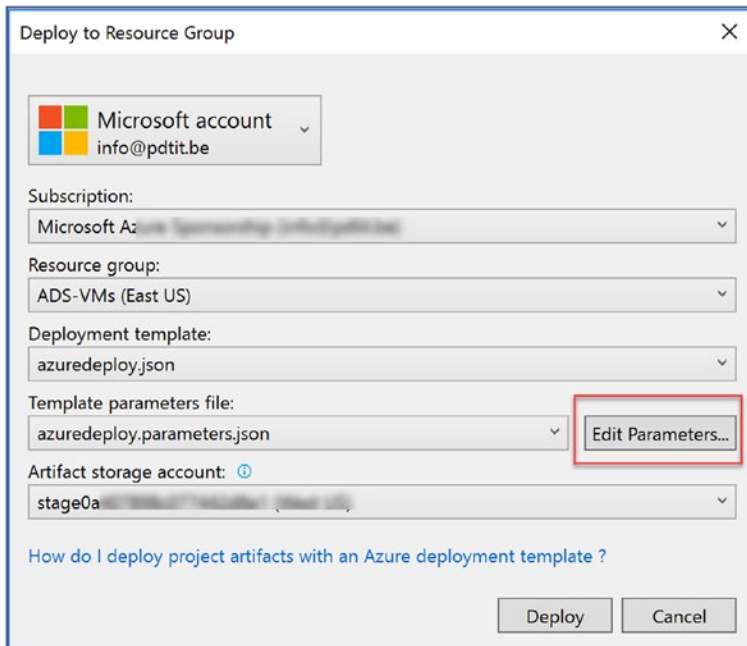
1. From within the Solution Explorer window, select the **AzureResourceGroup44** project, and **right-click** it; and from the context menu, select **Deploy** ► **New....**



2. In the appearing “Deploy to Resource Group” popup, complete the following settings:

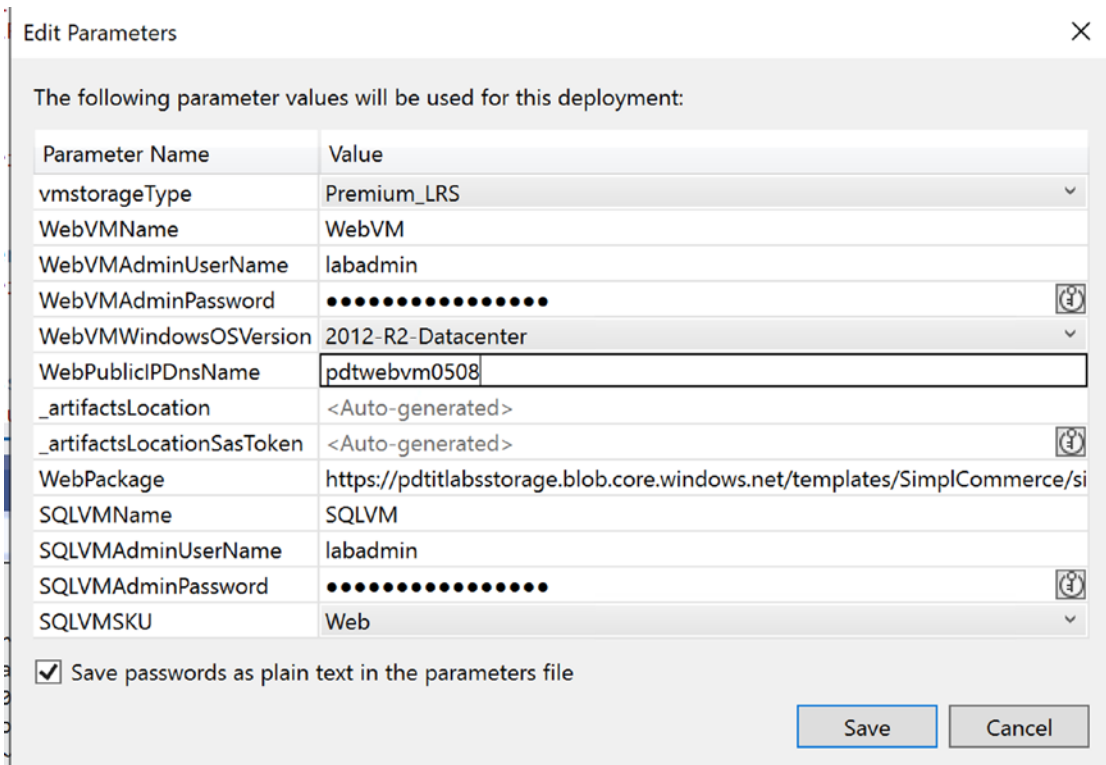


- **Subscription: Your Azure subscription**
- **Resource group: Create New/[SUFFIX]-VMs** – with location the one **closest to your location**
- **Deployment template: azuredeploy.json**
- **Template parameters file: azuredeploy.parameters.json**



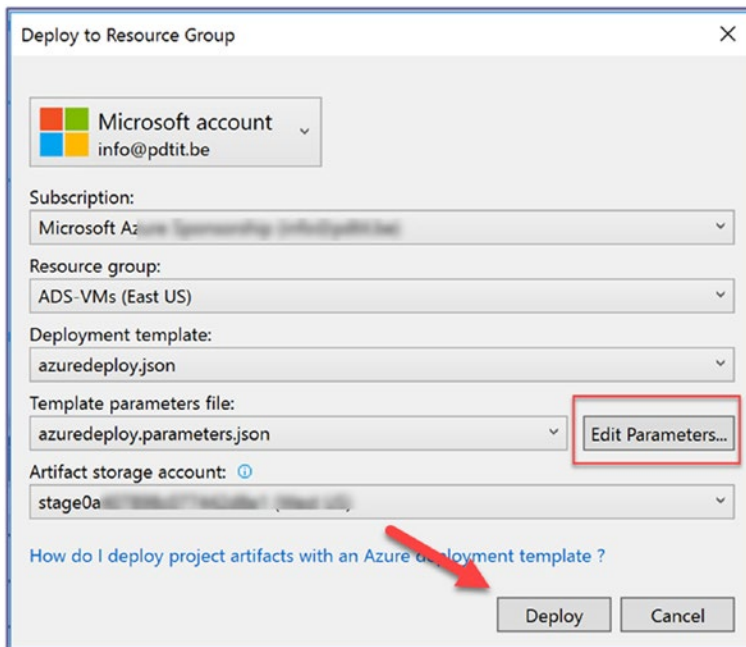
3. **Before clicking the Deploy** button, complete some additional deployment settings by **clicking the Edit Parameters... button**.

Basically, the only required change here is providing a new unique DNS name for the **WebPublicIPDnsName** parameter:



- **WebVMName:** WebVM
- **WebVMAdminUserName:** labadmin
- **WebVMAdminPassword:** L@BadminPa55w.rd (**do not alter this password, as otherwise the customization script later on won't work**)
- **WebVMWindowsOSVersion:** 2012-R2-Datacenter
- **WebPublicIPDnsName:** [suffix]webvm<date> (**all lowercase, no complex characters**)
- **SQLVMName:** SQLVM
- **SQLVMAdminUserName:** labadmin
- **SQLVMAdminPassword:** L@BadminPa55w.rd (**do not alter this password, as otherwise the customization script later on won't work**)

4. **Check** the “Save passwords as plain text in the parameters file.”
(Note: This is ok in this lab environment, but not recommended in production deployments. If this option is not checked, you will get a PowerShell window appearing, asking you for this administrator password there.)
5. Once all settings have been completed in the Edit Parameters popup window, click **Save**. You are redirected to the “Deploy to Resource Group” window. Start the actual deployment by clicking the **Deploy** button.



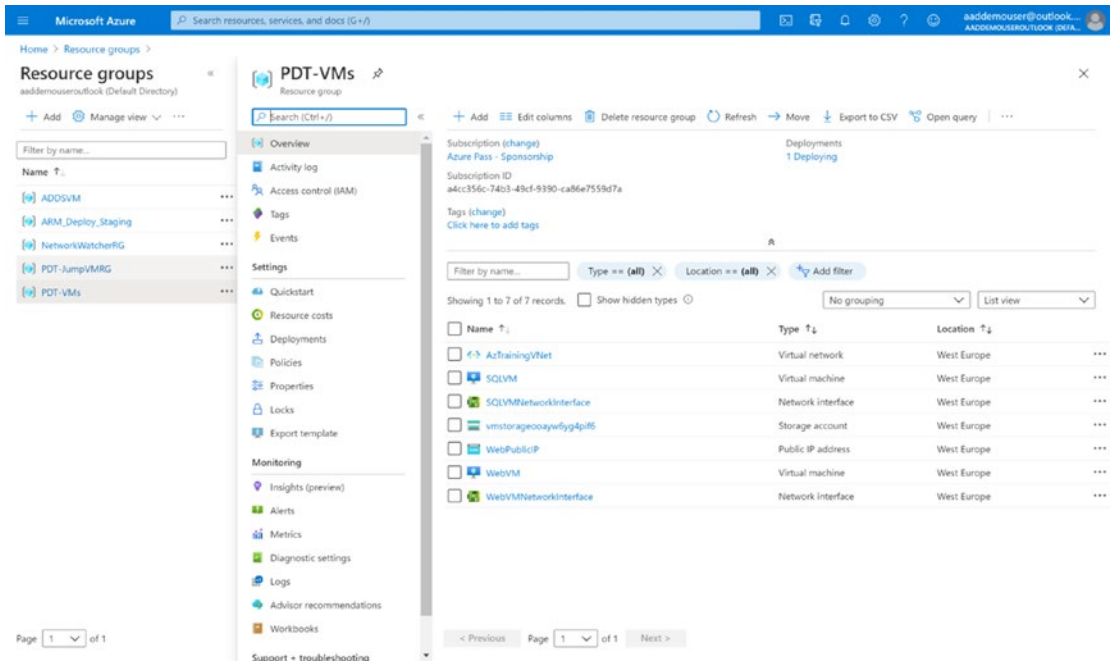
6. The Azure resource deployment kicks off, which can be followed from the Visual Studio **Output** window. (**For your info, this deployment takes about 15–20 min. It might be a good time for a break.**)

CHAPTER 3 LAB 1: DEPLOYING AN AZURE VIRTUAL MACHINE BASELINE APPLICATION WORKLOAD

```
Output
Show output from: PDT-VMs
23:21:51 - The following parameter values will be used for this operation:
23:21:51 - vmstorageType: Premium_LRS
23:21:51 - WebVMName: WebVM
23:21:51 - WebVMAdminUserName: labadmin
23:21:51 - WebVMAdminPassword: <securestring>
23:21:51 - WebVMWindowsOSVersion: 2012-R2-Datacenter
23:21:51 - WebPublicIPDnsName: pdtwebvm0508
23:21:51 - _artifactsLocation:
23:21:51 - _artifactsLocationSasToken: <securestring>
23:21:51 - WebPackage: https://pdtitlabstorage.blob.core.windows.net/templates/simplCommerce/simplcommerce\_iisource.zip
23:21:51 - SQLVMName: SQLVM
23:21:51 - SQLVMAdminUserName: labadmin
23:21:51 - SQLVMAdminPassword: <securestring>
23:21:51 - SQLVMSKU: Web
23:21:51 - Build started.
23:21:51 - Project "AzureResourceGroup44.deployproj" (StageArtifacts target(s)):
23:21:51 - Project "AzureResourceGroup44.deployproj" (ContentFilesProjectOutputGroup target(s)):
23:21:51 - Done building project "AzureResourceGroup44.deployproj".
23:21:51 - Done building project "AzureResourceGroup44.deployproj".
```

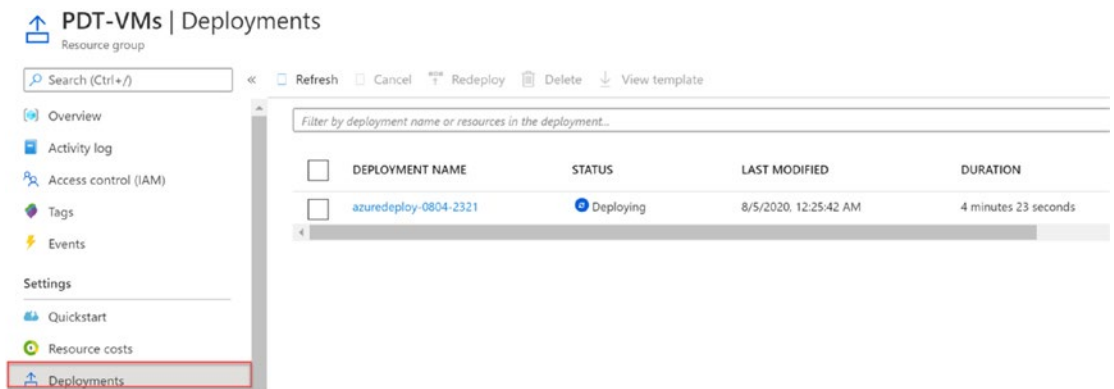
```
Output
Show output from: PDT-VMs
23:22:41 - VERBOSE: 11:22:41 PM - Checking deployment status in 5 seconds
23:22:46 - VERBOSE: 11:22:46 PM - Checking deployment status in 5 seconds
23:22:51 - VERBOSE: 11:22:51 PM - Checking deployment status in 5 seconds
23:22:56 - VERBOSE: 11:22:56 PM - Checking deployment status in 5 seconds
23:23:01 - VERBOSE: 11:23:01 PM - Checking deployment status in 5 seconds
23:23:07 - VERBOSE: 11:23:07 PM - Checking deployment status in 5 seconds
23:23:12 - VERBOSE: 11:23:12 PM - Checking deployment status in 5 seconds
23:23:17 - VERBOSE: 11:23:17 PM - Checking deployment status in 5 seconds
23:23:22 - VERBOSE: 11:23:22 PM - Checking deployment status in 5 seconds
23:23:27 - VERBOSE: 11:23:27 PM - Checking deployment status in 5 seconds
23:23:32 - VERBOSE: 11:23:32 PM - Checking deployment status in 5 seconds
23:23:37 - VERBOSE: 11:23:37 PM - Checking deployment status in 5 seconds
23:23:42 - VERBOSE: 11:23:42 PM - Resource Microsoft.Compute/virtualMachines/extensions 'WebVM/Microsoft.Powershell.DSC'
23:23:42 - provisioning status is running
23:23:42 - VERBOSE: 11:23:42 PM - Resource Microsoft.Compute/virtualMachines/extensions 'WebVM/Customize-WinVM' provisioning
23:23:42 - status is running
23:23:42 - VERBOSE: 11:23:42 PM - Resource Microsoft.Compute/virtualMachines 'WebVM' provisioning status is succeeded
23:23:42 - VERBOSE: 11:23:42 PM - Checking deployment status in 16 seconds
23:23:58 - VERBOSE: 11:23:58 PM - Checking deployment status in 5 seconds
```

7. While the deployment from Visual Studio is still running, open your **Internet browser**, connect to <http://portal.azure.com>, and **authenticate** with your Azure subscription credentials. **Go to Resource groups**, and open the [SUFFIX]-VMs resource group (RG). Here, you can see the different resources getting created.



8. From the **Resource groups** blade, **Settings** section, click **Deployments**.

9. This **shows** the actual running deployment task.



10. **Click the deployment name (e.g., azuredeploy-0804-2321),** which shows you more details about the actual deployment) process, including the already deployed resources.

CHAPTER 3 LAB 1: DEPLOYING AN AZURE VIRTUAL MACHINE BASELINE APPLICATION WORKLOAD

Home > Resource groups > PDT-VMs | Deployments >

azuredeploy-0804-2321 | Overview

Search (Ctrl+) Delete Cancel Redeploy Refresh

- Overview
- Inputs
- Outputs
- Template

Your deployment is underway

Deployment name: azuredeploy-0804-2321 Start time: 8/5/2020, 12:21:59 AM
Subscription: Azure Pass - Sponsorship Correlation ID: 5a82f1c6-3ce2-4669-9518-a5f0f6c7397f
Resource group: PDT-VMs



Deployment details (Download)

| Resource | Type | Status | Operation details |
|---------------------------------|----------------------------------|---------|-----------------------------------|
| SQLVM/Microsoft.PowerShell.DSC | Microsoft.Compute/virtualMach... | Created | Operation details |
| SQLVM/Customize-WinVM | Microsoft.Compute/virtualMach... | Created | Operation details |
| WebVM/Microsoft.PowerShell.D... | Microsoft.Compute/virtualMach... | Created | Operation details |
| WebVM/Customize-WinVM | Microsoft.Compute/virtualMach... | Created | Operation details |
| WebVM | Microsoft.Compute/virtualMach... | OK | Operation details |
| SQLVM | Microsoft.Compute/virtualMach... | OK | Operation details |
| vmstorageoayw6yg4pif6 | Microsoft.Storage/storageAcco... | OK | Operation details |
| WebVMNetworkInterface | Microsoft.Network/networkInte... | Created | Operation details |
| SQLVMNetworkInterface | Microsoft.Network/networkInte... | Created | Operation details |
| WebPublicIP | Microsoft.Network/publicIPAdd... | OK | Operation details |
| vmstorageoayw6yg4pif6 | Microsoft.Storage/storageAcco... | OK | Operation details |
| AzTrainingVNet | Microsoft.Network/virtualNetw... | OK | Operation details |





11. **Wait for the deployment to complete successfully.** This is noticeable from within the Visual Studio Output window or from within the Azure Portal Deployment blade you were in before.





```
Output
Show output from: ADS-VMs
21:16:49 - sqlvmAdminPassword SecureString
21:16:49 - sqlvmsku String Standard
21:16:49 -
21:16:49 - Outputs : {}
21:16:49 - OutputsString :
21:16:49 -
21:16:49 -
21:16:49 -
21:16:49 - Successfully deployed template 'azuredeploy.json' to resource group 'ADS-VMs'.
```



Home > Resource groups > PDT-VMs | Deployments >


 **azuredeploy-0804-2321 | Overview** 

Deployment

Search (Ctrl+/) <<  Delete  Cancel  Redeploy  Refresh

-  Overview
-  Inputs
-  Outputs
-  Template

 **Your deployment is complete**

 Deployment name: azuredeploy-0804-2321
Subscription: [Azure Pass - Sponsorship](#)
Resource group: [PDT-VMs](#)

∨ Deployment details ([Download](#))

∧ Next steps

[Go to resource group](#)

12. **Close Visual Studio** without saving changes to the project.

To verify all went fine during the deployment of the Azure resources, as well as the customization and configuration using PowerShell Desired State Configuration, we will validate if the e-commerce webshop sample workload is running fine.

13. From within the **Azure Portal**, go to **Resource groups**, and select the resource group where **you deployed the VMs ([SUFFIX]-VMs)**. In here, **select the WebVM** virtual machine by **clicking** it. This **opens the WebVM detailed blade**.

Home >

Virtual machines

Microsoft

+ Add ▾ ⌚ Reservations ≡ Edit columns ↻ Refresh | 🏷 Assign tags ▶

Subscriptions: 1 of 23 selected – Don't see a subscription? [Open Directory + Subscription set](#)

2 items

| <input type="checkbox"/> | Name ↑↓ | Type ↑↓ | Status |
|--------------------------|---------|-----------------|-----------------------|
| <input type="checkbox"/> | SQLVM | Virtual machine | Stopped (deallocated) |
| <input type="checkbox"/> | WebVM | Virtual machine | Stopped (deallocated) |

Home > Resource groups > PDT-VMs | Deployments > azuredeploy-0804-2321 | Overview > PDT-VMs >

WebVM

Virtual machine

Search (Ctrl+F) | Connect | Start | Restart | Stop | Move | Delete | Refresh | Share to mobile

WebVM is not using Managed Disks. Migrate to Managed Disks to get more benefits. →

| | | | |
|-------------------------|--|------------------------|---|
| Resource group (change) | : PDT-VMs | Operating system | : Windows (Windows Server 2012 R2 Datacenter) |
| Status | : Running | Size | : Standard_DS1_v2 (1 vcpu, 3.5 GiB memory) |
| Location | : West Europe | Public IP address | : 13.93.27.107 |
| Subscription (change) | : Azure Pass - Sponsorship | Virtual network/subnet | : AzTrainingVNet/FrontendNetwork |
| Subscription ID | : a4cc356c-74b3-49ef-9390-ca86e7559d7a | DNS name | : pdtwebvm0508.westeurope.cloudapp.azure.com |
| Tags (change) | : displayName : WebVM | | |

- 14. Notice the **Public IP address** of the WebVM resource. **Open your browser** and connect to this IP address. After a few seconds, the **SimpleCommerce** webshop home page should become visible.

The screenshot displays the SimpliCommerce website interface. At the top, there is a navigation bar with 'Log in' and 'Register' links. Below this is the 'SimpliCommerce' logo, a search bar with the placeholder 'Search here...', a dropdown menu for 'All Categories', and a shopping cart icon with a red notification badge. A secondary navigation bar lists 'WOMAN', 'MAN', 'SHOES', and 'WATCHES'. The main content area features a large hero image of a man in profile wearing a dark blue jacket, with the text 'Jackets & Coats' overlaid. Below the hero image are two product grids. The first grid, titled 'New products', contains four items: 'Square Neck Back' (a grey t-shirt), 'Pretty Little Thing' (a black t-shirt), 'T-Shirt with Sleeve' (a white t-shirt), and 'Herschel supply' (a brown belt). The second grid, titled 'Man collection', contains four items: 'Herschel supply' (a brown belt), 'Herschel supply' (a light blue button-down shirt), 'Vintage Inspired Classic' (a black watch), and 'Only Check Trouser' (a light blue button-down shirt). Each product card includes an image, a title, a price, a discount badge, a star rating, and a review count.

Log in Register

SimpliCommerce Search here... All Categories

WOMAN MAN SHOES WATCHES

Jackets & Coats

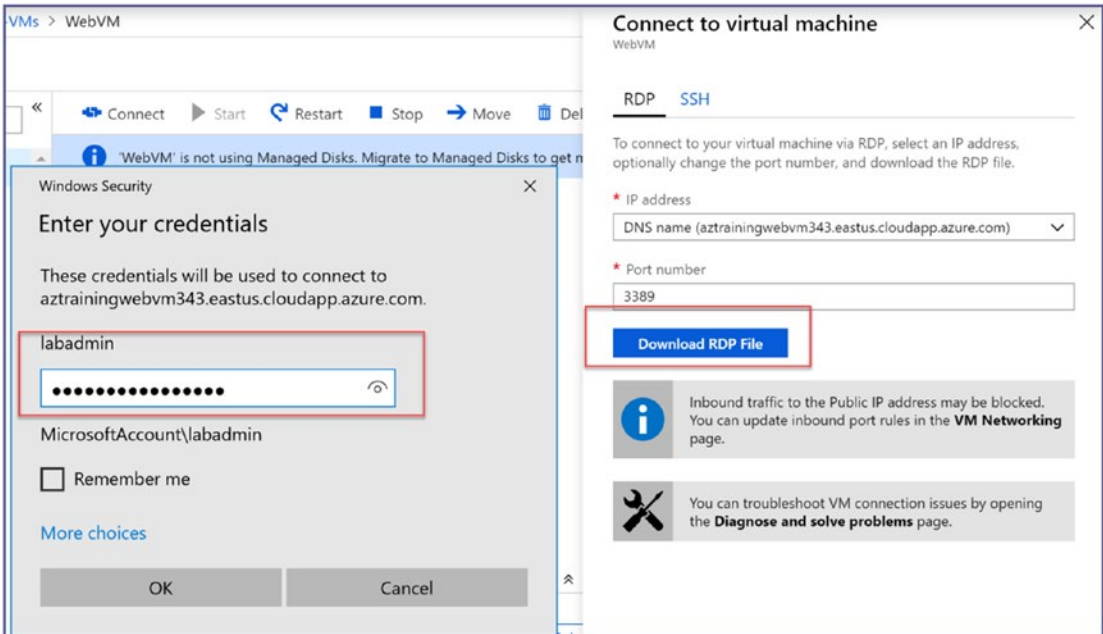
New products

- Square Neck Back**
\$29.64 **25%**
\$39.54
★★★★★ (1 reviews)
- Pretty Little Thing**
\$54.79
☆☆☆☆☆
- T-Shirt with Sleeve**
\$18.49
★★★★★ (1 reviews)
- Herschel supply**
\$63.15
☆☆☆☆☆

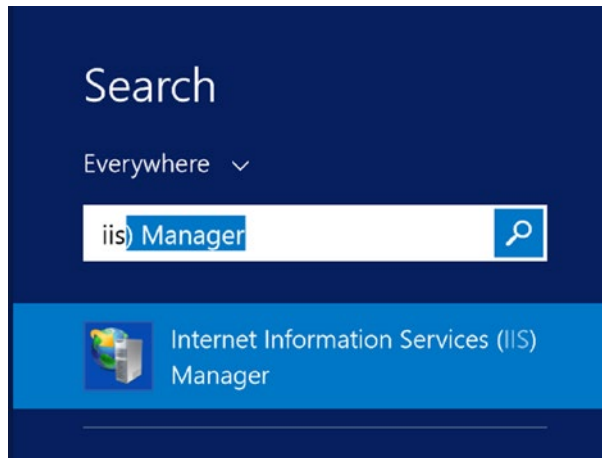
Man collection

- Herschel supply**
\$63.15
☆☆☆☆☆
- Herschel supply**
\$63.16
☆☆☆☆☆
- Vintage Inspired Classic**
\$93.20
☆☆☆☆☆
- Only Check Trouser**
\$25.50
☆☆☆☆☆

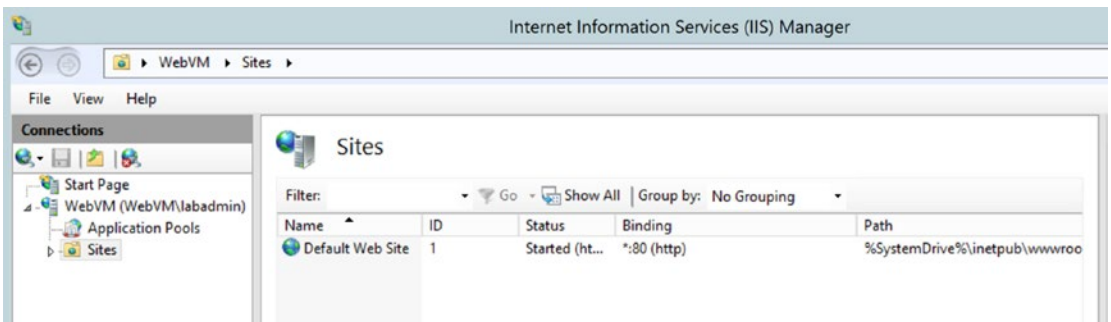
- 15. Back in the Azure Portal, from your WebVM blade (within the **Azure Portal**, go to **Resource groups**, and select the resource group where **you deployed the VMs ([SUFFIX]-VMs)**. In here, **select the WebVM virtual machine by clicking it**. This **opens the WebVM detailed blade**), **select Overview ► Connect**.
- 16. **Click the Connect button**, to open the Remote Desktop session to this WebVM virtual machine.



- 17. Here, log on with the credentials from the ARM template (labadmin, L@BadminPa55w.rd) unless you changed those before the deployment in Visual Studio.
- 18. From within the WebVM RDP session’s Start menu, search for “IIS,” which resolves **Internet Information Services Manager**.



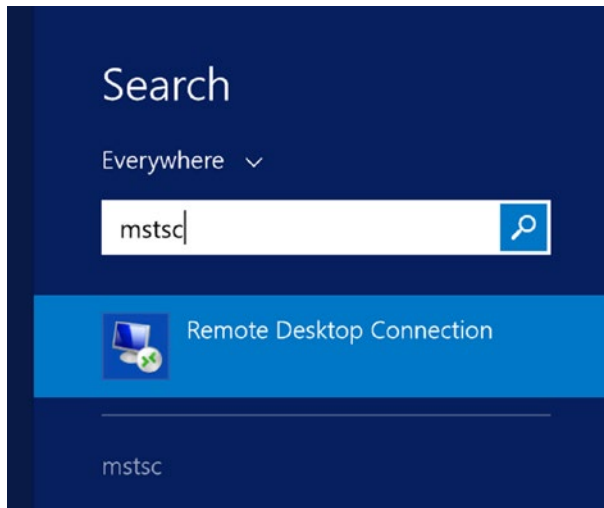
19. **Launch** Internet Information Services Manager.



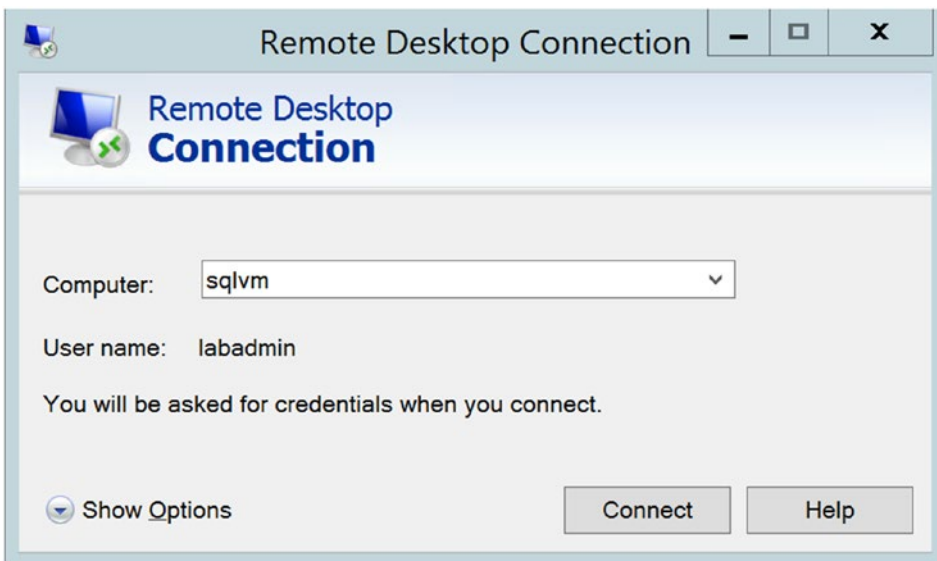
20. **This deployment has a Default Web Site** configured.

Close the Internet browser session on the WebVM.

21. **Still from within the WebVM RDP session**, start a new RDP session to the SQLVM (this needs to happen from within the WebVM, as the SQLVM has no public IP address attached to its NIC, thus not reachable from the Internet directly), by clicking the Start button and typing “**mstsc**”; this finds the Microsoft Remote Desktop Connection. **Launch it.**



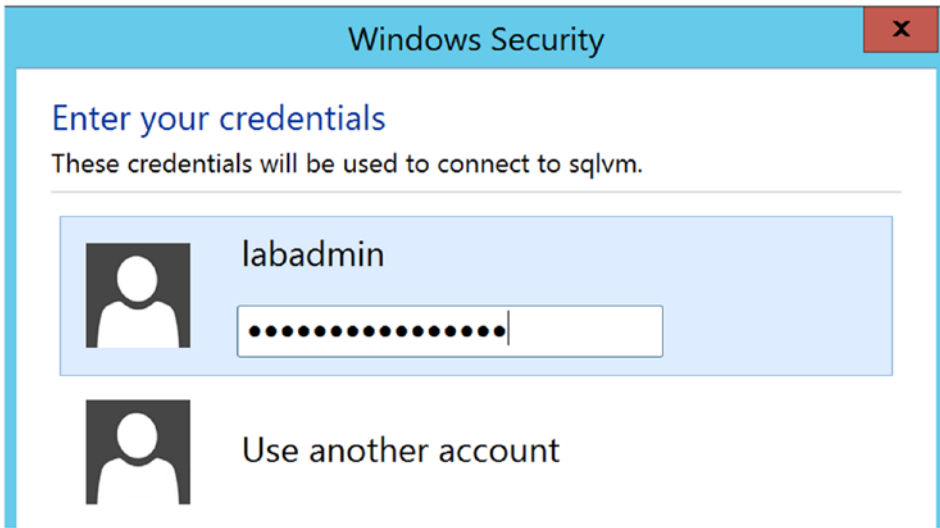
22. Enter “**sqlvm**” as computer name; **next**, click the **Connect** button.



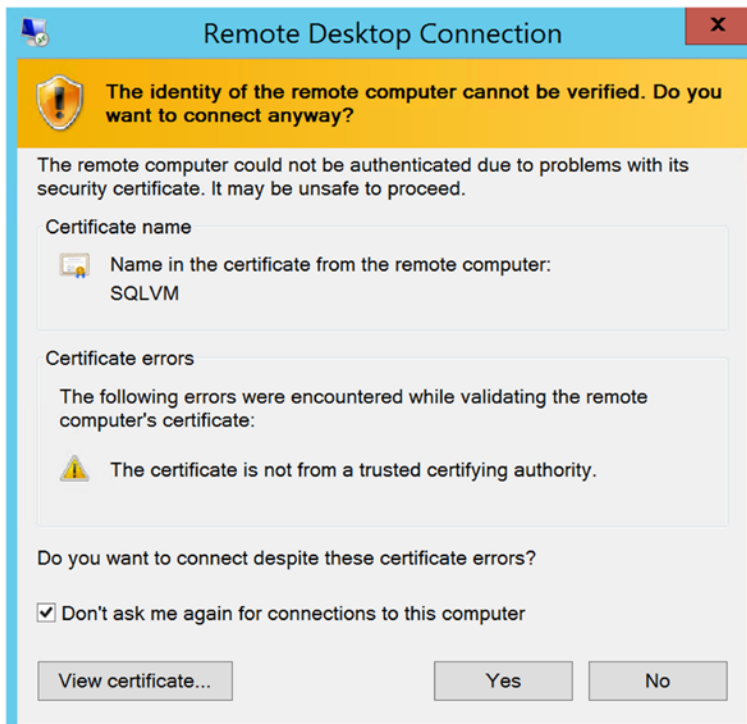
23. Provide the following credentials to authenticate:

User: labadmin

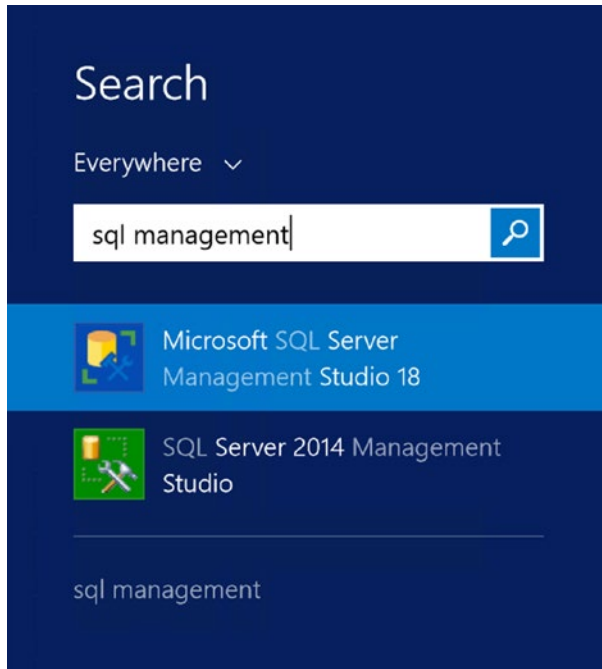
Password: L@BadminPa55w.rd



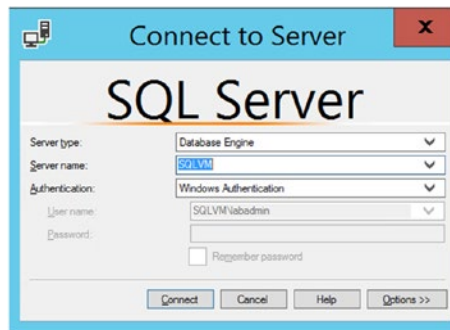
24. And **click OK** to continue.
25. When prompted with **“The identity of the remote computer cannot be verified”** error, select **“Don’t ask me again for connections to this computer.”**



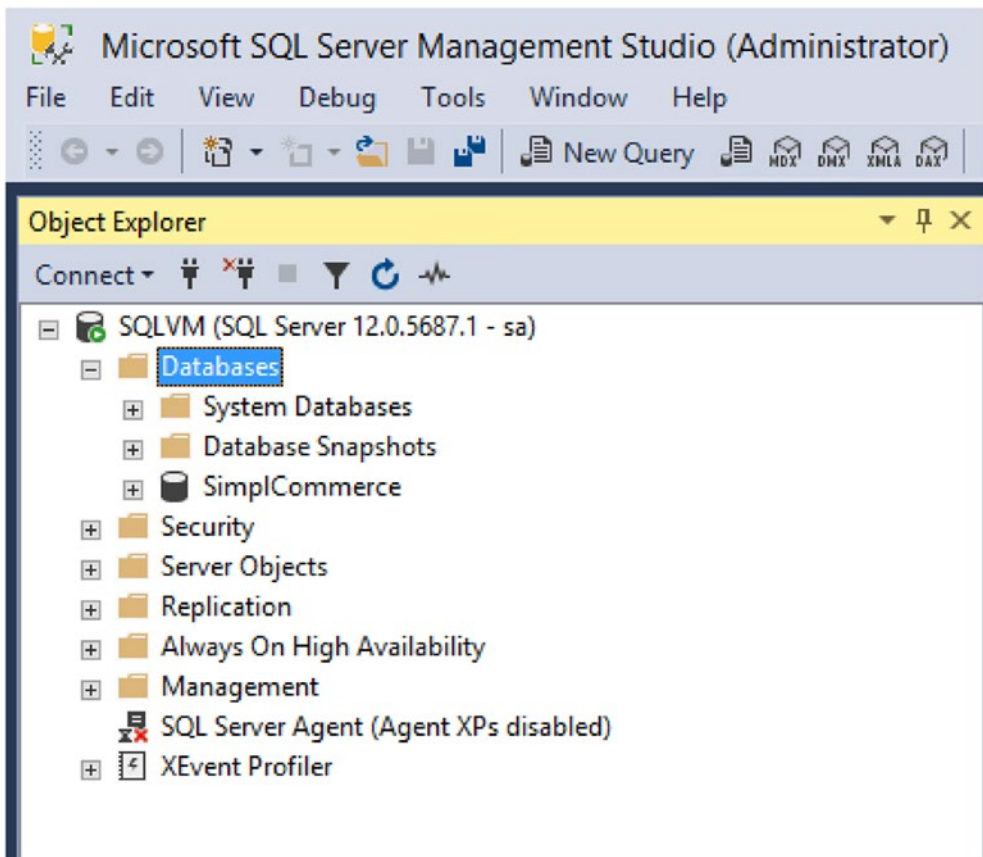
26. **Click Yes** to open the RDP session. Wait for the desktop of the SQLVM to load completely.
27. From the Start menu of the SQLVM, search for “SQL management,” which will resolve a list of keywords and applications. Here, select **Microsoft SQL Server Management Studio 18**.



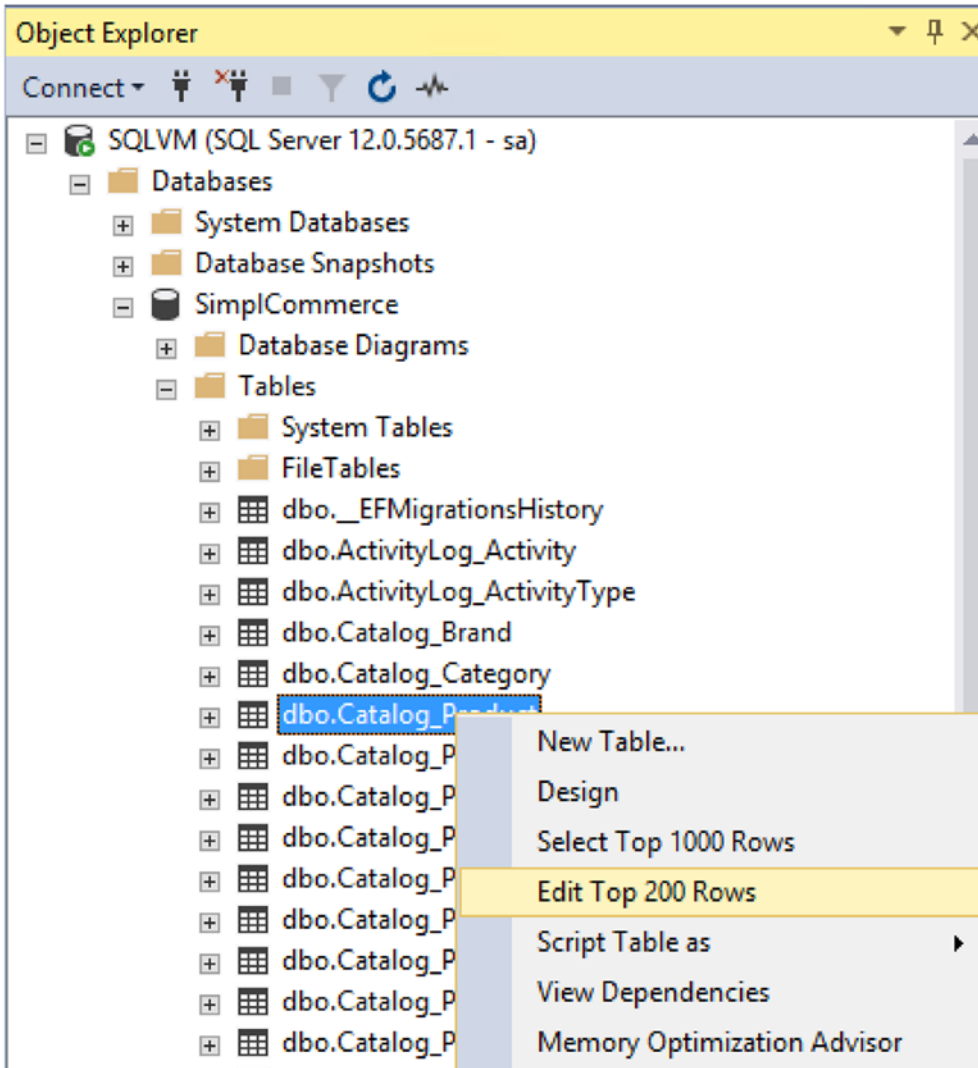
28. From SQL Server Management Studio, the “**Connect to Server**” popup opens. Provide the following information:
 - **Server type: Database Engine**
 - **Server name: SQLVM**
 - **Authentication: SQL Server Authentication**



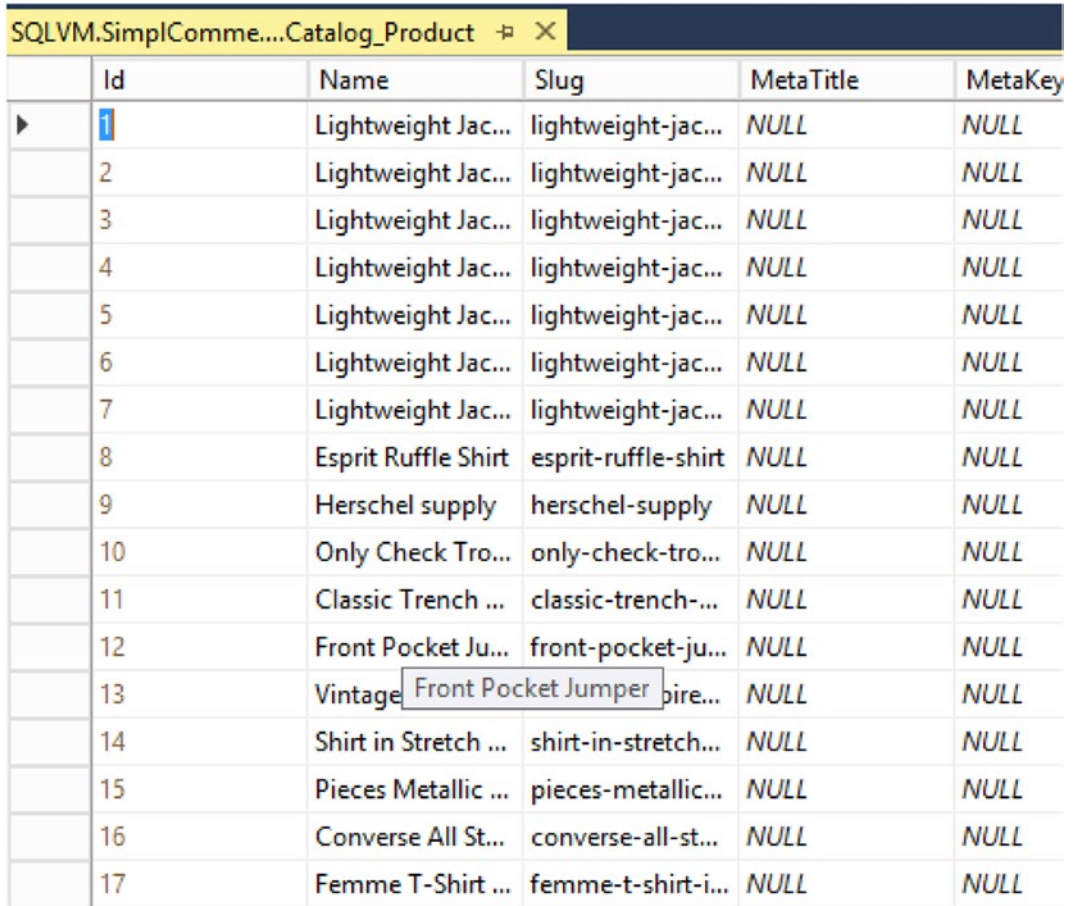
29. Click **Connect** to open the SQL Server connection.
30. Validate the **SimplCommerce** database object is available under the Databases section of the server.



- 31. Open the SimplCommerce database, by clicking the “+” in front of the name; browse to **Tables** and click the “+” again here. This opens a list of all tables within this database. Here, browse to **dbo.Catalog_Product** and **select it**.
- 32. Next, **right-click** this table, to open the context menu. Here, **select “Edit Top 200 Rows.”**



33. This shows a list of products in our sample e-commerce application.



The screenshot shows a SQL Server query result window titled "SQLVM.SimplComme...Catalog_Product". The table has six columns: Id, Name, Slug, MetaTitle, and MetaKey. The data is as follows:

| | Id | Name | Slug | MetaTitle | MetaKey |
|---|----|-----------------------------|---------------------|-----------|---------|
| ▶ | 1 | Lightweight Jac... | lightweight-jac... | NULL | NULL |
| | 2 | Lightweight Jac... | lightweight-jac... | NULL | NULL |
| | 3 | Lightweight Jac... | lightweight-jac... | NULL | NULL |
| | 4 | Lightweight Jac... | lightweight-jac... | NULL | NULL |
| | 5 | Lightweight Jac... | lightweight-jac... | NULL | NULL |
| | 6 | Lightweight Jac... | lightweight-jac... | NULL | NULL |
| | 7 | Lightweight Jac... | lightweight-jac... | NULL | NULL |
| | 8 | Esprit Ruffle Shirt | esprit-ruffle-shirt | NULL | NULL |
| | 9 | Herschel supply | herchel-supply | NULL | NULL |
| | 10 | Only Check Tro... | only-check-tro... | NULL | NULL |
| | 11 | Classic Trench ... | classic-trench-... | NULL | NULL |
| | 12 | Front Pocket Ju... | front-pocket-ju... | NULL | NULL |
| | 13 | Vintage Front Pocket Jumper | front-pocket-jum... | NULL | NULL |
| | 14 | Shirt in Stretch ... | shirt-in-stretch... | NULL | NULL |
| | 15 | Pieces Metallic ... | pieces-metallic... | NULL | NULL |
| | 16 | Converse All St... | converse-all-st... | NULL | NULL |
| | 17 | Femme T-Shirt ... | femme-t-shirt-i... | NULL | NULL |

34. This confirms the deployment of the SQL Server VM was successful.

This completes the task.

Summary

In this lab, you started with deploying an ARM template from within the Azure Portal, deploying a lab jumpVM virtual machine in Azure.

In the next task, you learned how to deploy a more complex Azure environment, again using an ARM template, where deployment was executed from within Visual Studio 2017/2019, using ARM templates to deploy Azure resources, as well as relying on Azure VM PowerShell DSC and Custom Script Extensions to fine-tune the configuration of the WebVM and SQLVM virtual machines.

CHAPTER 4

Lab 2: Performing Assessment of Your As-Is Situation

Lab 2: Performing assessment of your as-is situation

What You Will Learn

In this second lab, you focus on performing the necessary assessment phase in your simulated “on-premises” application landscape, by using Microsoft assessment tools:

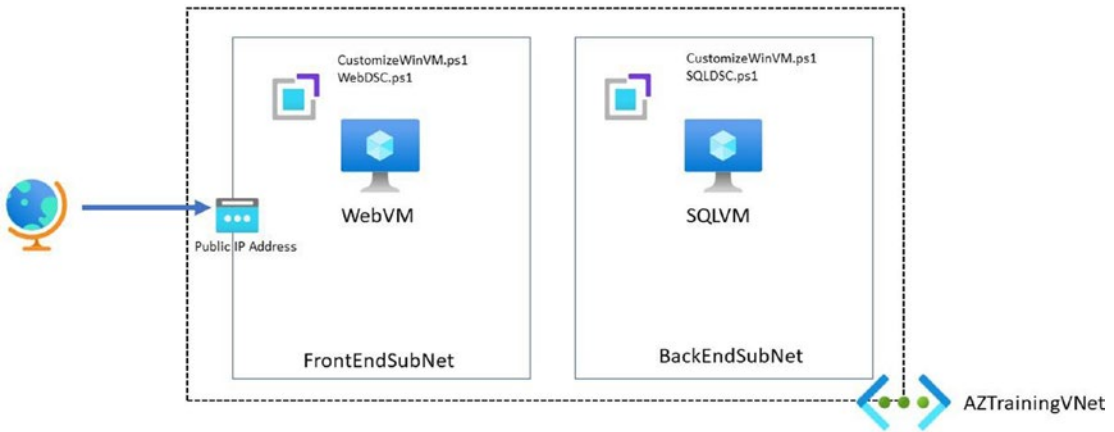
- Microsoft Data Migration Assistant (DMA)
- Azure App Service Migration Assistant

Time Estimate

This lab is estimated to take **30 min**, assuming your Azure subscription is already available and you successfully completed Lab 1, in which you deployed the baseline setup with the WebVM and SQLVM.

Prerequisites

Make sure you completed the ARM scenario deployment from Lab 1 before starting this exercise, as it is continuing on the infrastructure deployed out of that lab.



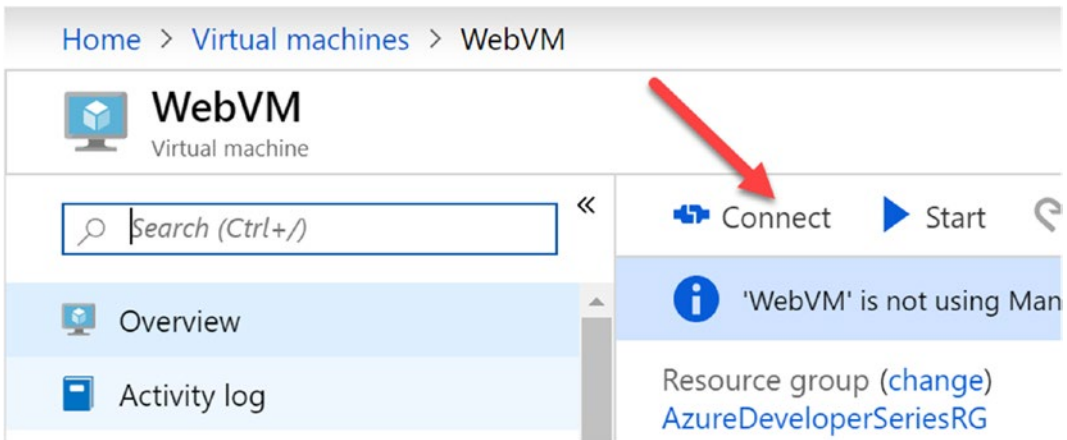
Task 1: Running a SQL Server assessment using Data Migration Assistant

In short, you will perform the following tasks:

1. Install the Azure Data Migration Assistant on the WebVM.
2. Perform an assessment of the to-be-migrated database.

In this task, you download and install the Azure Data Migration Assistant.

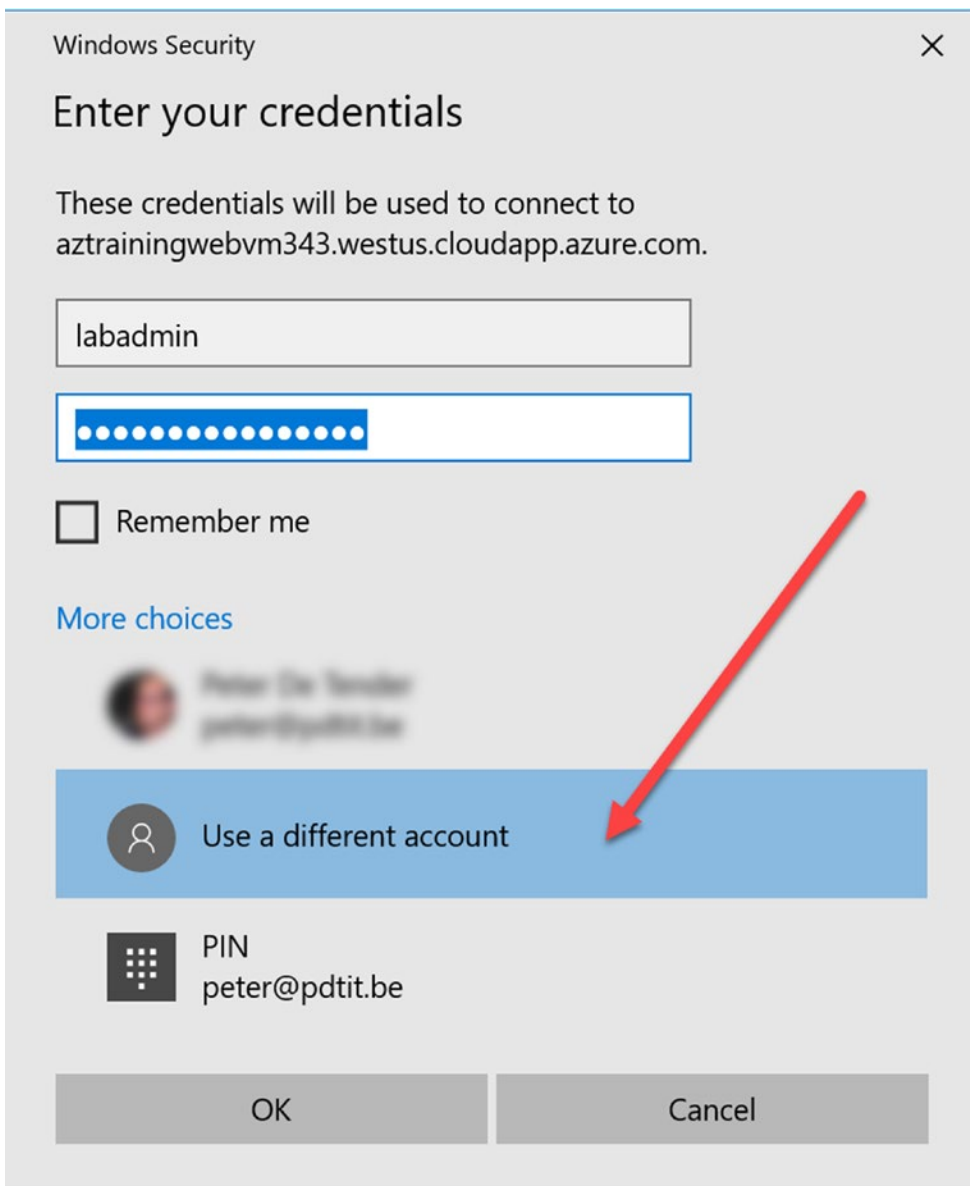
1. Connect to the WebVM virtual machine using RDP, by **selecting** the WebVM from the **Virtual machines** section in the Azure Portal followed by **selecting Connect**.



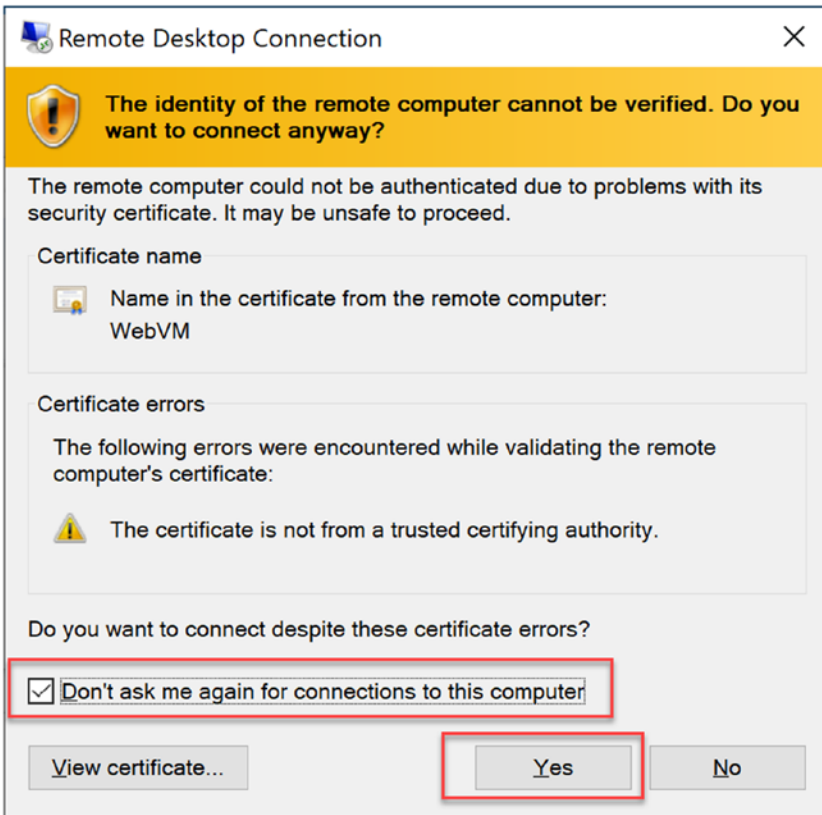
2. In the **Connect** blade, **click Download RDP File**. Once downloaded, open the file. This will start the **Remote Desktop**, asking for credentials. Here, **select “Use a different account”** and provide the following credentials:

User account: labadmin

Password: L@BadminPa55w.rd



- When you are **prompted for a certificate security warning**, select **Don't ask me again...** and click **Yes** to continue.



- Once logged on to the desktop of the WebVM, open the browser, and search for **Azure Data Migration Assistant** download, or connect directly to the following URL: www.microsoft.com/en-us/download/details.aspx?id=53595.

https://www.microsoft.com/en-us/download/details.aspx?id=53595

Important! Selecting a language below will dynamically change the complete page content to that language.

Language: English

Download

Data Migration Assistant (DMA) enables you to upgrade to a modern data platform by detecting compatibility issues that can impact database functionality on your new version of SQL Server. It recommends performance and reliability improvements for your target environment. It allows you to not only move your schema and data, but also uncontained objects from your source server to your target server.

+ Details

+ System Requirements

+ Install Instructions

5. Once the download is complete, launch the **DataMigrationAssistant.msi**. Click **Next** to continue.



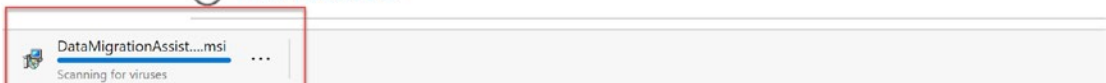
Thank you for downloading Microsoft® Data Migration Assistant v5.2

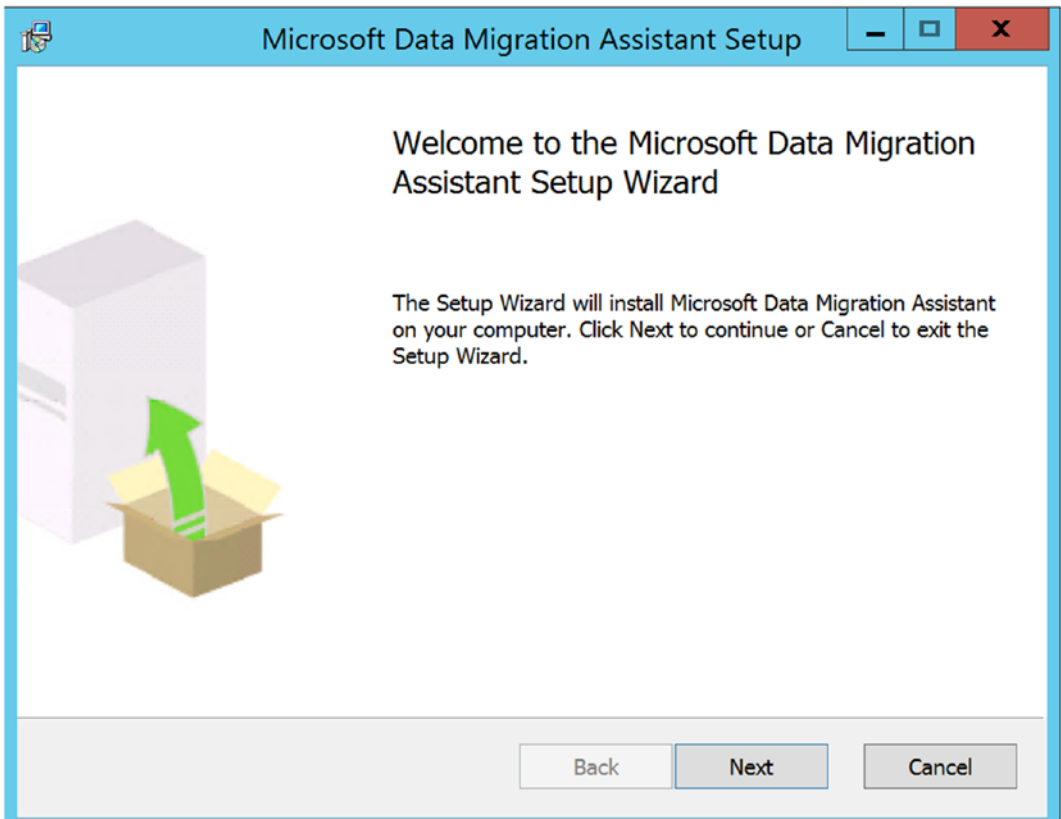
If your download does not start after 30 seconds, [click here to download manually](#).

Installation note:

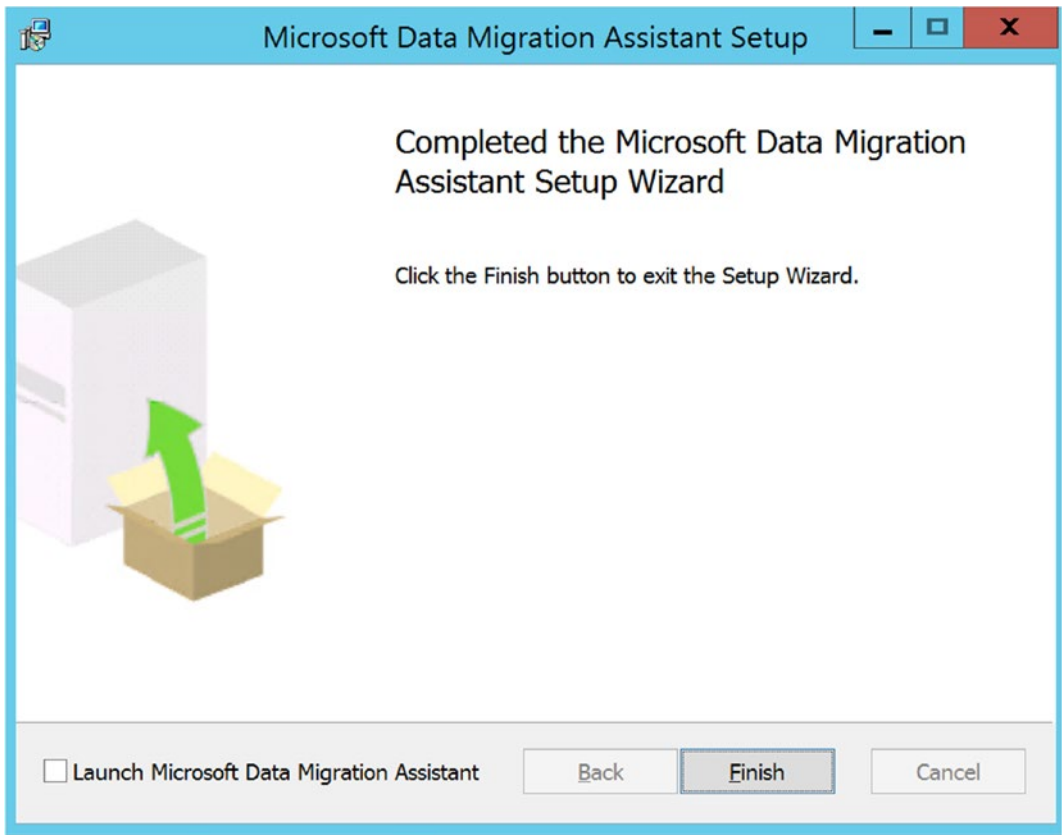
In the following Install Instructions, please start at the step after the mention of clicking the Download button.

+ Install Instructions

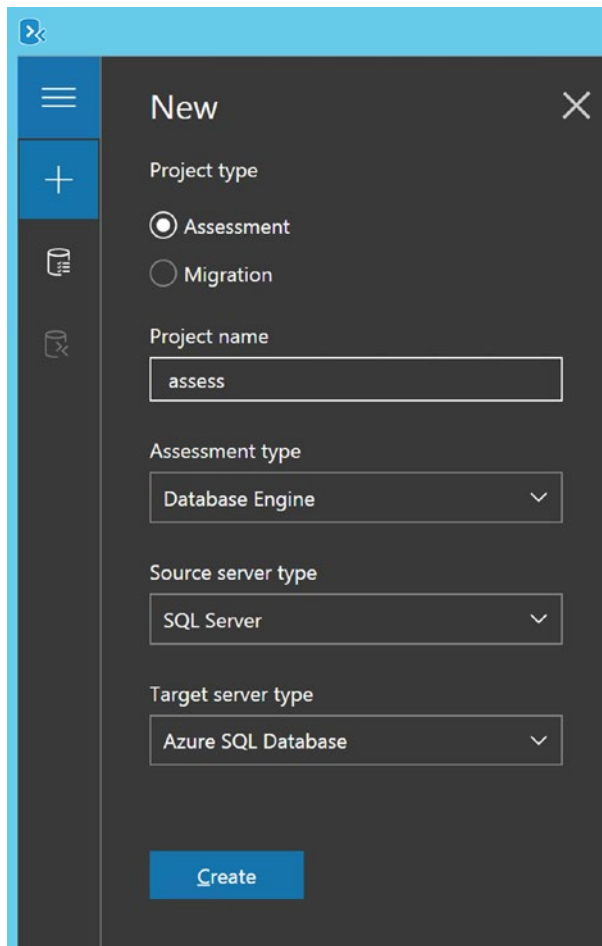




6. Accept the license terms agreement, click Next, and confirm by clicking the **Install** button. Wait for the install to complete successfully.

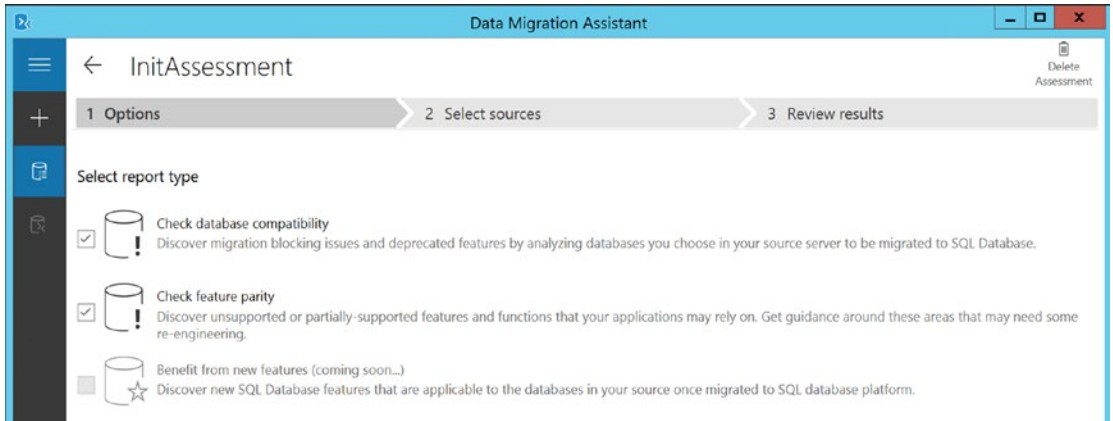


7. To open the DMA tool, select **“Launch Microsoft Data Migration Assistant.”**
8. From **Data Migration Assistant**, select the + on the side to **launch a new “Assessment” scenario.**



9. **We start** by running an **assessment**. Complete the wizard with the following parameters:
- **Project type: Assessment**
 - **Project name: assess**
 - **Assessment type: Database Engine**
 - **Source server type: SQL Server**
 - **Target server type: Azure SQL Database**
- And confirm these options by clicking “Create.”**

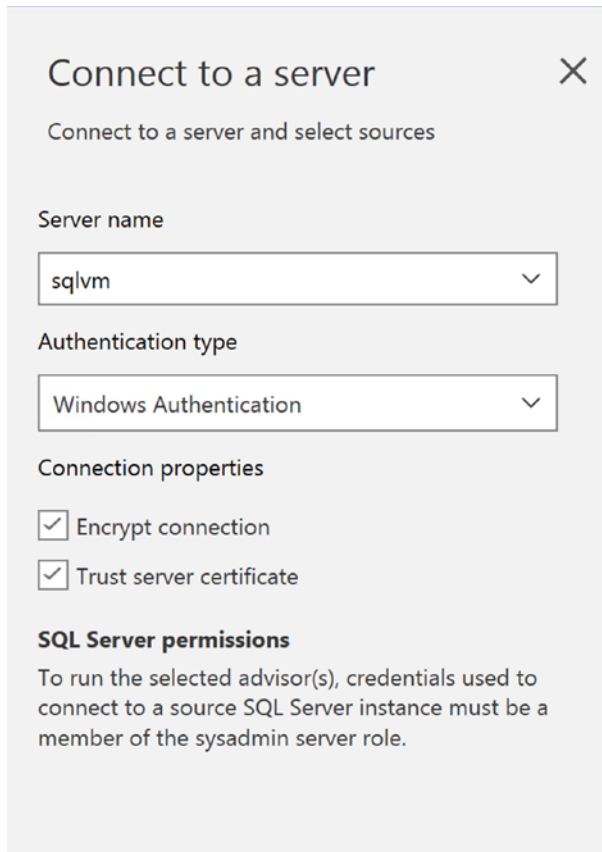
10. This launches the Data Migration Assistant selection window. Here, **click Next** to continue.



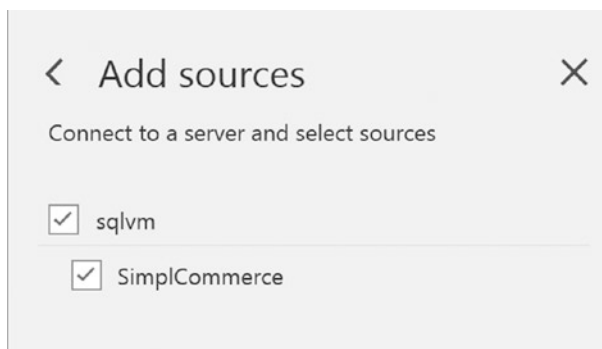
11. We now need to connect to our source SQL Server. Therefore, provide the following information in the wizard:

- **Server name: sqlvm**
- **Authentication type: Windows Authentication**
- **Username: labadmin**
- **Password: L@BadminPa55w.rd**

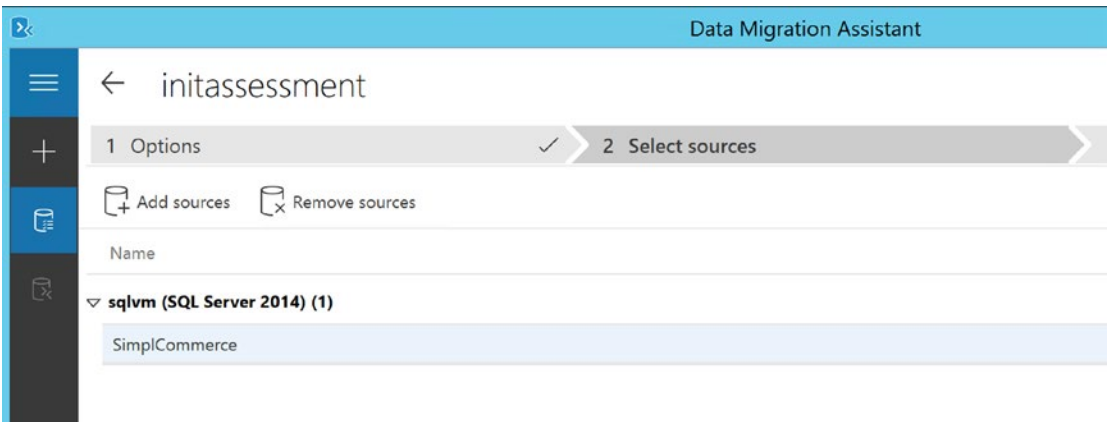
Also flag both options “Encrypt connection” and “Trust server certificate.”



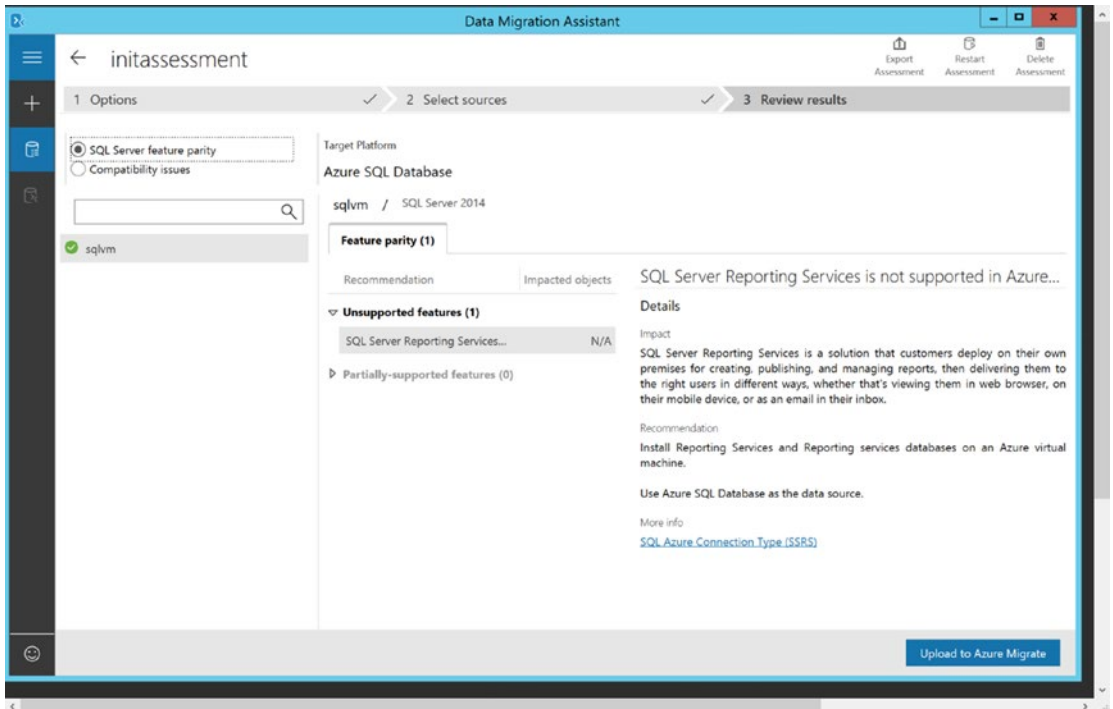
12. Click **Connect** to continue. This brings up the sources list.



13. **Select** SimplCommerce as source database, and **select Add**, to add this database to the list.



14. Next, click the “Start Assessment” button. This runs the assessment and should take a few minutes to complete. Take note of the several recommendations under **Unsupported features** and **Partially supported features**.



15. **Once** you are familiar with the reported features, you can close Data Migration Assistant.

This completes the task in which you deployed and ran Data Migration Assistant to validate compatibility of your source SQL Server database with Azure SQL target.

In a next lab, you will reuse this tool to perform the actual database migration.

Task 2: Running a web server assessment using Azure App Service Migration Assistant

In short, you will perform the following tasks:

1. Install the Azure App Service Migration Assistant on the WebVM.
2. Perform an assessment of the to-be-migrated web application.

In this task, you download and install the Azure App Service Migration Assistant. We are using the WebVM directly in this lab, but you can run this from any Windows Server in the same network as the WebVM virtual machine, meaning you don't have to install it on the web server VM itself.

1. Connect to the WebVM virtual machine using RDP, by **selecting** the WebVM from the **Virtual machines** section in the Azure Portal followed by **selecting Connect** and authenticating with labadmin and L@BadminPa55w.rd as credentials.
2. From within the **WebVM**, open an Internet browser, and connect to the following URL to download the latest version of the Azure App Service Migration Assistant:

<https://appmigration.microsoft.com/>

Microsoft App Service Migration Home Assess Download Readiness Checks Partners

Migrate to Azure App Service

Assess any app with an endpoint scan. Download the Migration Assistant and start your .NET and PHP app migration to Azure App Service.

Assess
Provide a public URL to assess to start your migration process.

Download
For internal .NET and PHP apps, download the Migration Assistant.

Start your assessment by entering a public facing endpoint to scan.

https:// Assess your site by entering a public URL **Assess**

3. **Click the Download option**, to get redirected to the download page. Here, continue with **clicking** the Download button.

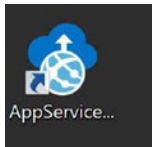
Download the Migration Assistant

Migration Assistant

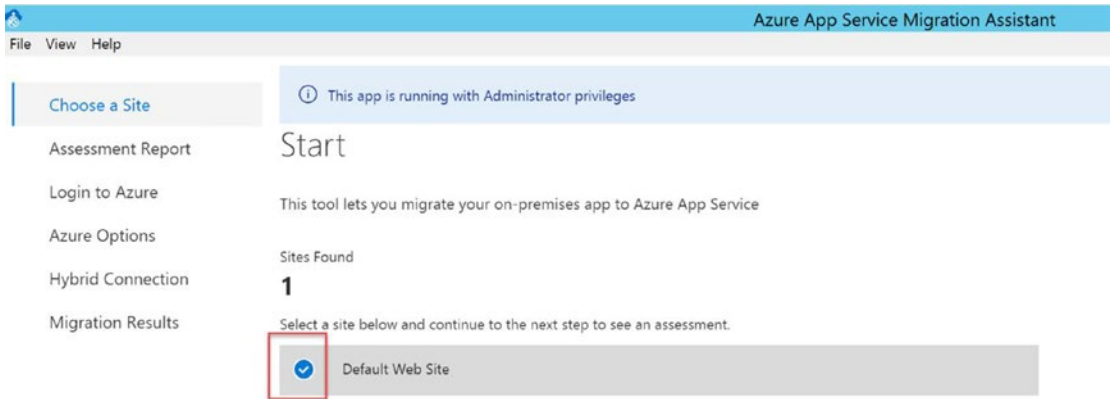


[Download](#)

4. **Once downloaded**, launch the **AppServiceMigrationAssistant.msi**, which will configure a shortcut on the desktop.



5. **Launch** the AppServiceMigrationAssistant. This brings up a five-step scenario. **Select Step 1 “Choose a Site”**; here, notice it has found one site, “Default Web Site.”



6. **Select “Default Web Site”** and click **Next** to continue.
7. **This results** in a detailed assessment report of the web application. Browse through this report to become familiar with the gathered information.

Assessment Report for 'Default Web Site'

| Success | Warning | Error | | |
|---------------------------|---|---|--|--|
| 13 | 0 | 0 | | |
| ^ Name | Description | Details | | |
| ^ Success (13) | | | | |
| ISAPI Filters | Checks if the site is using ISAPI filters not standard in App Service. | No unsupported ISAPI filters were detected. | | |
| Global Modules | Checks for global modules not standard in App Service. | No unsupported global modules were detected. | | |
| TCP Ports | Checks for TCP port bindings not supported in App Service | No unsupported port bindings were found. | | |
| Protocols | Checks for protocols not supported in App Service | No unsupported protocols were found. | | |
| Location Tags | Checks for location tags in applicationHost.config | No unsupported location tags were found. | | |
| Single Application Pool | Checks for multiple application pool usage by site. | The site is using a single application pool. | | |
| Authentication Type | Checks for authentication types not supported on App Service | No unsupported authentication types were found | | |
| Virtual Directories | Checks for virtual directories not compatible with App Service | No incompatible virtual directories were found. | | |
| PHP Versions | Checks for unsupported PHP configurations | No issues found. | | |
| Site Content Size | Checks if the site content is too large for automatic migration | The site content is ok. | | |
| Configuration Errors | Checks for errors in the IIS configuration | No configuration errors were found | | |
| Certificates | Checks to see if application is using HTTPS | The application does not use HTTPS | | |
| Application Pool Identity | Checks to see if application pool is running as a user supported by App Service | The site's application pool is running as a supported user. | | |

8. **Notice** we have no errors nor warnings.
9. **Leave this web app migration tool open for now, as you will reuse it in a following chapter to perform the actual web app migration. If you close it, you will need to run part of the assessment again later.**

This completes the task in which you installed and ran the App Service Migration Assistant tool, to identify compatibility and supportability issues of your existing web application workload, when being migrated to Azure Web Apps.

Summary

In this lab, you deployed the Data Migration Assistant as well as the App Service Migration Assistant, to validate your existing e-commerce application environment, being compatible with Azure Platform as a Service, as part of the assessment phase of your migration project.

In the next labs, you will reuse these tools to perform an actual migration of the SQL Server database to Azure SQL, as well as migrating the web application to Azure Web Apps.

CHAPTER 5

Lab 3: Deploying an Azure SQL Database and Migrating from SQLVM

Lab 3: Deploying an Azure SQL database and migrating from SQLVM

What You Will Learn

In this lab, you perform a migration from a SQL 2014 database running on the SQLVM to SQL Azure PaaS, using the SQL Data Migration Assistant (DMA), following these steps:

- Deploy a new Azure SQL Server instance.
- Authenticate to SSMS on the SQLVM virtual machine.
- Run the database migration wizard from within DMA.
- Verify the successful migration of the SQL database from the VM to Azure.
- Update the connection strings on the WebVM web application to point to the SQL Azure database instead of the on-premises one on SQLVM.
- Optional: Migrate the database using SQL Server Management Studio.

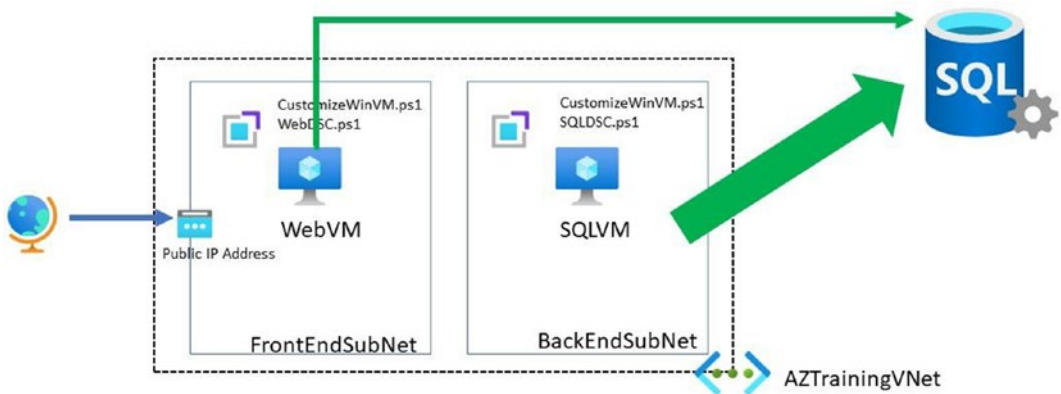
Time Estimate

This lab is estimated to take **60 min**, assuming your Azure subscription is already available.

Prerequisites

Make sure you completed Lab 1 and Lab 2 before starting this lab scenario, as it is building up on those.

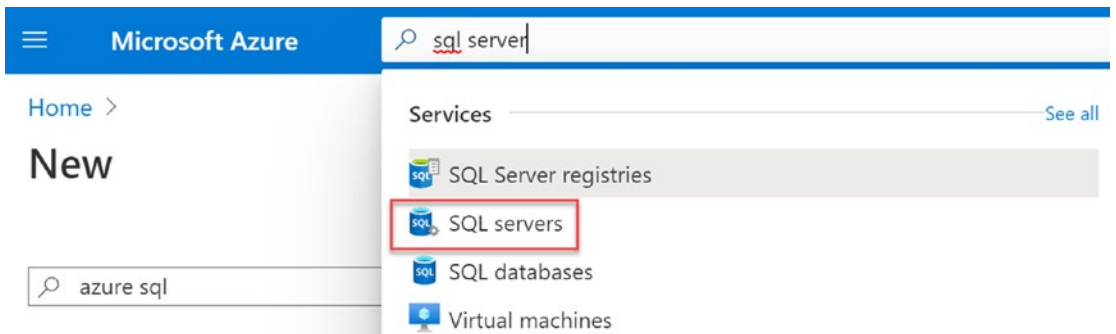
Scenario Diagram



Task 1: Deploying a new Azure SQL Server instance

In this task, you start deploying a new Azure SQL Server instance from within the Azure Portal, allowing you to migrate a database to it in the next task.

1. From within the **Azure Portal** “**Search resources, services, and docs (G+)**,” enter “**SQL servers.**” From the list of results, select **SQL servers.**



2. **Click** “Create a new SQL Server” or **click** the “+Add” button in the top menu. This launches the Create SQL Database Server deployment blade.

Home > SQL servers >

Create SQL Database Server

Microsoft


Basics

Networking

Additional settings

Tags

Review + create

SQL database server is a logical container for managing databases and elastic pools. Complete the Basic tab, then go to Review + Create to provision with smart defaults, or visit each tab to customize. [Learn more](#) 

3. **Complete** the different deployment settings as follows:

Basics tab:

- **Server name:** [suffix]sqlazure[date], for example, pdtsqlazure0508 (capitals are not allowed)
- **Server admin login:** labadmin
- **Password:** L@BadminPa55w.rd
- **Confirm password:** L@BadminPa55w.rd
- **Subscription:** Your Azure subscription
- **Resource group:** Create New/[SUFFIX]-SQLAzureRG
- **Location:** Same Azure location as where you deployed the WebVM and SQLVM


Note Although we define the same server admin credentials as the SQLVM SQL Server instance, these can be completely different in reality. We decide to define it this way for ease of the lab scenario. Same goes for the SQL Azure resource location, which can be any of the available Azure regions worldwide, irrelevant from where your SQL Server virtual machine is running today.

[Home](#) > [SQL servers](#) >

Create SQL Database Server

Microsoft

[Basics](#) [Networking](#) [Additional settings](#) [Tags](#) [Review + create](#)

SQL database server is a logical container for managing databases and elastic pools. Complete the Basic tab, then go to Review + Create to provision with smart defaults, or visit each tab to customize. [Learn more](#) 

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Resource group * ⓘ [Create new](#)

Server details

Enter required settings for this server, including providing a name and location.

Server name * [.database.windows.net](#)

Location *

Administrator account

Server admin login *

Password *

Confirm password *

Networking tab

- **Allow Azure services and resources to access this server: switch to “Yes.”**

[Home](#) > [SQL servers](#) >

Create SQL Database Server

Microsoft

Basics Networking Additional settings Tags Review + create

Configure networking access for your server.

Firewall rules

Allow Azure services and resources to access this server ⓘ

Yes No

- **We won't use the Additional settings tab for now.**
4. **Confirm** the creation of the Azure SQL Server by **clicking the “Review + create”** button.

Review + create

< Previous

Next : Additional settings >

5. **Validate the deployment summary, and confirm by clicking Create.**

Create SQL Database Server

Microsoft

- Basics
- Networking
- Additional settings
- Tags
- Review + create**

Product details

SQL Database Server
by Microsoft
[Terms of use](#) | [Privacy policy](#)

Estimated cost per month
No additional charges

Terms

By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the M same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my cont not provide rights for third-party offerings. For additional details see [Azure Marketplace Terms](#). [↗](#)

Basics

| | |
|--------------------|-----------------------|
| Subscription | MSFT ATT Subscription |
| Resource group | PDTSQLAzureRG |
| Server name | pdtsqlazure1708 |
| Server admin login | labadmin |
| Location | West Europe |

Networking

Allow Azure services to access server Yes

Additional settings

Enable advanced data security Not now

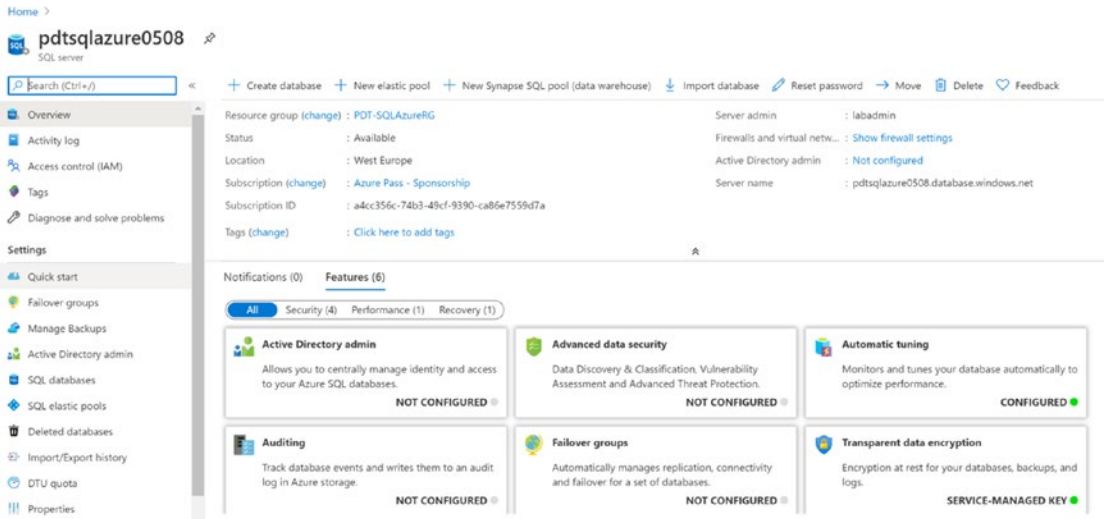
Tags

Create

< Previous

[Download a template for automation](#)

6. Wait for the deployment to complete.




- Once the Azure SQL Server has been deployed successfully, we can create a new database, by **clicking the “+ Create database”** button from the top menu. From here, we will define two settings, the database name and the database size:
 - Database name: [suffix]sqlazuredb**
 - Compute + storage: Standard S0, 10DTUs, 250 GB storage**

[Home](#) > [pdtsqlazure0508](#) >

Create SQL Database

Microsoft

[Basics](#) [Networking](#) [Additional settings](#) [Tags](#) [Review + create](#)

Create a SQL database with your preferred configurations. Complete the Basics tab then go to Review + Create to provision with smart defaults, or visit each tab to customize. [Learn more](#) 

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

| | | |
|------------------|--------------------------|---|
| Subscription ⓘ | Azure Pass - Sponsorship | ▼ |
| Resource group ⓘ | PDT-SQLAzureRG | ▼ |

Database details

Enter required settings for this database, including picking a logical server and configuring the compute and storage resources

| | | |
|-----------------------------------|---|---|
| Database name * | <input type="text" value="pdtazuredb"/> | ✓ |
| Server ⓘ | pdtsqlazure0508 (West Europe) | ▼ |
| Want to use SQL elastic pool? * ⓘ | <input type="radio"/> Yes <input checked="" type="radio"/> No | |
| Compute + storage * ⓘ | Standard S0 10 DTUs, 250 GB storage Configure database | |

- To modify the Compute + storage settings, **click “Configure database.”**

Home > pdtsqlazure0508 > Create SQL Database >

Configure

Feedback

| | | |
|---|--|--|
| Looking for basic, standard, premium? | General Purpose Scalable compute and storage options 500 - 20,000 IOPS 2-10 ms latency | Hyperscale On-demand scalable storage 500 - 204,800 IOPS 1-10 ms latency |
|---|--|--|

Compute tier

| | |
|---|--|
| <p style="text-align: center;">Provisioned</p> <p style="text-align: center;">Compute resources are pre-allocated Billed per hour based on vCores configured</p> | <p style="text-align: center;">Serverless</p> <p style="text-align: center;">Compute resources are auto-scaled Billed per second based on vCores used</p> |
|---|--|

Compute Hardware

Click "Change configuration" to see details for all hardware generations available including memory optimized and compute optimized options

Hardware Configuration

Gen5
 up to 80 vCores, up to 408 GB memory
[Change configuration](#)

Save money

Save up to 55% with a license you already own. Already have a SQL Server license? ⓘ

Yes No

vCores [How do vCores compare with DTUs? ↗](#)



9. Select "Looking for basic, standard, premium?"

Home > pdtsqlazure0508 > Create SQL Database >

Configure

Feedback

| | | | |
|--|--|---|--|
| Basic For less demanding workloads | Standard For workloads with typical performance requirements | Premium For IO-intensive workloads. | vCore-based purchasing options Click here to customize your performance using vCores > |
|--|--|---|--|

DTUs: What is a DTU? ↗

Data max size

10 (S0)
 250 GB

| | |
|-------------------------------|------------------|
| Cost summary | |
| Cost per DTU (in EUR) | 1.27 |
| DTUs selected | x 10 |
| ESTIMATED COST / MONTH | 12.65 eur |

10. Define **10 (S0)** for **DTUs**, and keep the **Data max size** to 250 GB (know the sample database is about 50 Mb in size, but data size isn't really impacting cost within the same allocated DTU size).
11. **Click "Next: Networking"**; notice you can't make any changes to the firewall or networking settings here. We will make the necessary changes once the database has been created.
12. **Click "Next: Advanced Settings"**; accept the default settings as is.

[Home](#) > [pdtsqlazure0508](#) >

Create SQL Database

Microsoft

Basics Networking **Additional settings** Tags Review + create

Customize additional configuration parameters including collation & sample data.


Data source

Start with a blank database, restore from a backup or select sample data to populate your new database.

Use existing data *

None Backup Sample

Database collation


Database collation defines the rules that sort and compare data, and cannot be changed after database creation. The default database collation is SQL_Latin1_General_CP1_CI_AS. [Learn more](#) 

Collation * ⓘ

SQL_Latin1_General_CP1_CI_AS

[Find a collation](#)

Advanced data security

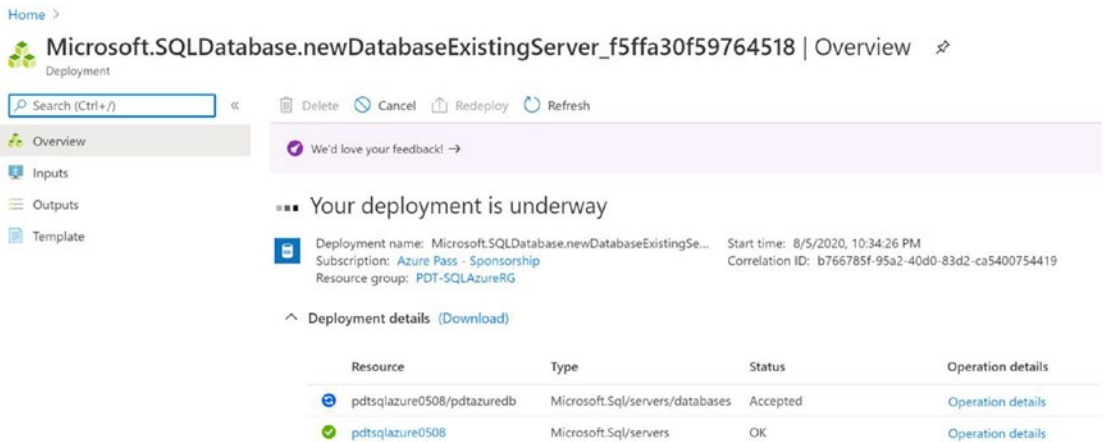
Protect your data using advanced data security, a unified security package including data classification, vulnerability assessment and advanced threat protection for your server. [Learn more](#) 

Get started with a 30 day free trial period, and then 12.6495 EUR/server/month.

Enable advanced data security * ⓘ

Start free trial **Not now**

13. **Confirm** the creation of the database by **clicking the “Review + create”** button. Validate the configuration settings, and confirm by **clicking “Create.”**



Home > Microsoft.SQLDatabase.newDatabaseExistingServer_f5ffa30f59764518 | Overview

Deployment

Search (Ctrl+/) Delete Cancel Redeploy Refresh

Overview

Inputs

Outputs

Template

We'd love your feedback →

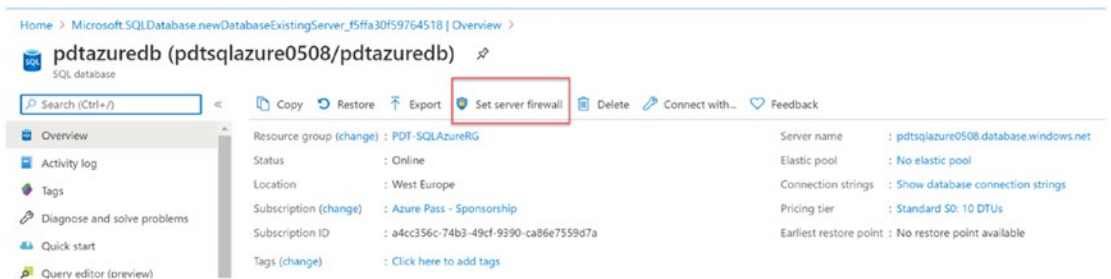
■ ■ ■ Your deployment is underway

Deployment name: Microsoft.SQLDatabase.newDatabaseExistingSe... Start time: 8/5/2020, 10:34:26 PM
 Subscription: Azure Pass - Sponsorship Correlation ID: b766785f-95a2-40d0-83d2-ca5400754419
 Resource group: PDT-SQLAzureRG

Deployment details (Download)

| Resource | Type | Status | Operation details |
|----------------------------|---------------------------------|----------|-----------------------------------|
| pdtsqlazure0508/pdtazuredb | Microsoft.Sql/servers/databases | Accepted | Operation details |
| pdtsqlazure0508 | Microsoft.Sql/servers | OK | Operation details |

14. **Wait** for the creation to complete. Once completed, **click** the **“Go to resource”** button, which redirects you to the SQL Azure database blade.



Home > Microsoft.SQLDatabase.newDatabaseExistingServer_f5ffa30f59764518 | Overview

pdtazuredb (pdtsqlazure0508/pdtazuredb)

SQL database

Search (Ctrl+/) Copy Restore Export Set server firewall Delete Connect with... Feedback

Overview

Activity log

Tags

Diagnose and solve problems

Quick start

Query editor (preview)

Resource group (change) : PDT-SQLAzureRG

Status : Online

Location : West Europe

Subscription (change) : Azure Pass - Sponsorship

Subscription ID : a4cc356c-74b3-49cf-9390-ca86e7559d7a

Tags (change) : [Click here to add tags](#)

Server name : pdtsqlazure0508.database.windows.net

Elastic pool : No elastic pool

Connection strings : Show database connection strings

Pricing tier : Standard S0: 10 DTUs

Earliest restore point : No restore point available

15. Here, we will modify the firewall settings, to allow the WebVM to connect to the Azure SQL Server database later on. **Click “Set server firewall”**.

16. Under Rule name, Start IP, and End IP, **enter the following parameters:**
 - **Rule name: allow_webVM.**
 - **Start IP: Enter the public IP address of the WebVM virtual machine.**
 - **End IP: Enter the public IP address of the WebVM virtual machine.**

| | | | |
|--|--|--|-----|
| Client IP address | 5.148.105.110 | | |
| Rule name | Start IP | End IP | |
| <input type="text" value="allow_webVM"/> ✓ | <input type="text" value="137.116.222.152"/> ✓ | <input type="text" value="137.116.222.152"/> ✓ | ... |

Note The reason we have the WebVM IP address here is because we will run the SQL database migration from this server.

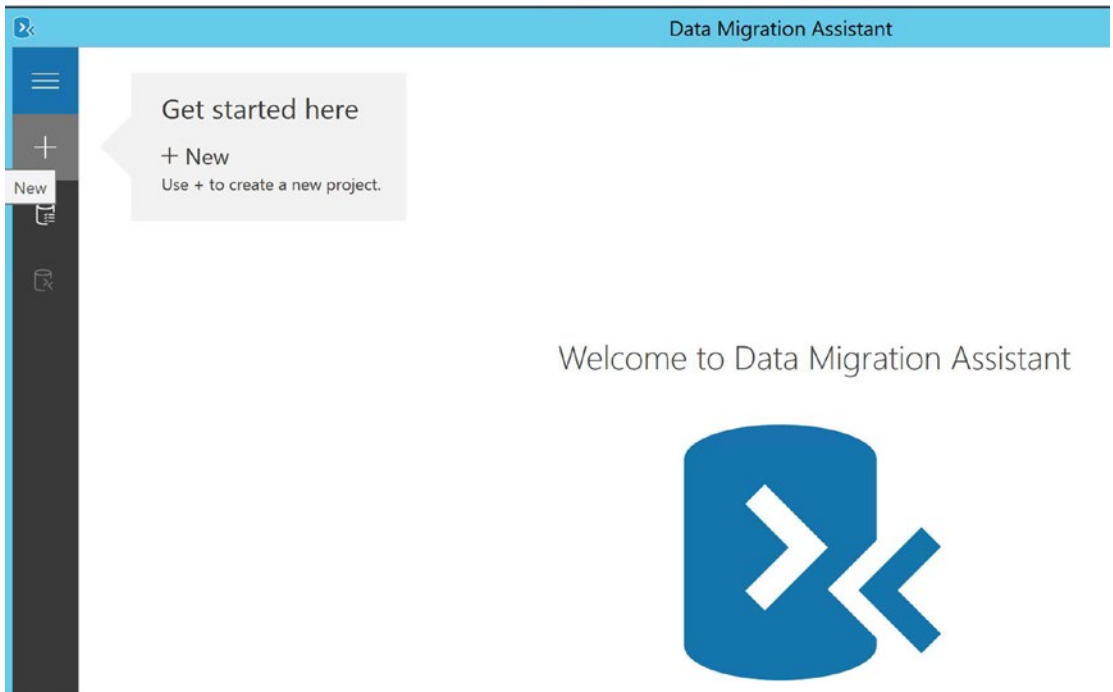
17. **Save** your settings.

This completes the first task, in which you deployed an Azure SQL Server instance and a new database. You also configured the necessary firewall settings to allow communication between the WebVM virtual machine and the Azure SQL Server.

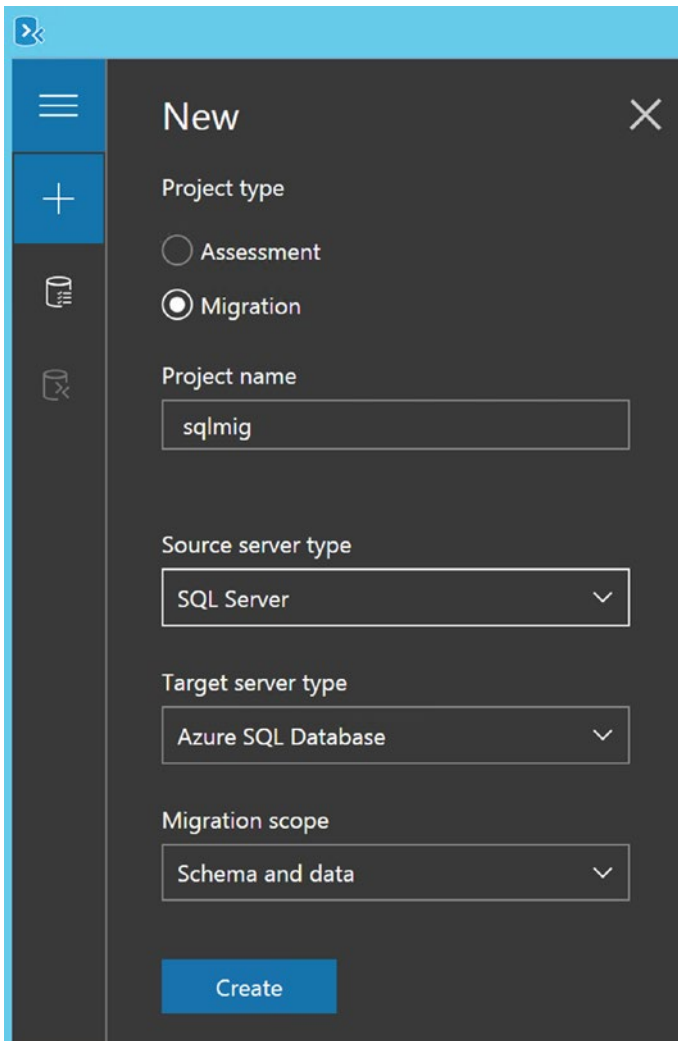
Task 2: Performing a SQL database migration from a SQL virtual machine to SQL Azure, using SQL Data Migration Assistant

In this task, you perform a SQL database migration from within a SQL virtual machine to SQL Azure. This approach is known as a lift and shift database migration, since no structure or data will be changed during the actual migration. Continuing on the path of the Azure migration tools available, you will use the Azure Data Migration Assistant you used earlier in the assessment phase to perform the actual migration.

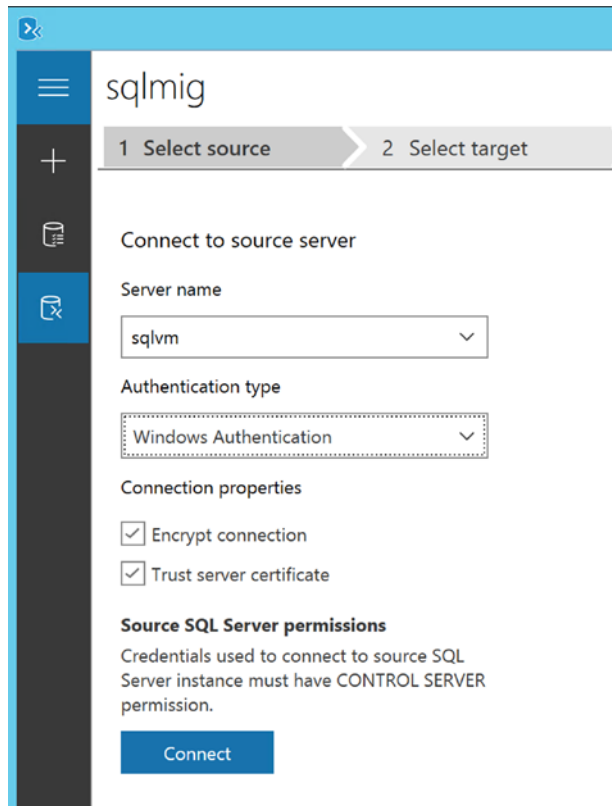
1. **Open an RDP session to the WebVM virtual machine** (using the same steps as described in the previous lab).
2. Once you are logged on to the WebVM RDP session, **launch Data Migration Assistant** (from a shortcut on the desktop or Start menu).



3. Click "+", to create a new project.
4. **Provide the following parameters:**
 - **Project type: Migration**
 - **Project name: SQLMig**
 - **Source server: SQL Server**
 - **Target server: Azure SQL Database**
 - **Migration scope: Schema and data**



5. **Click the Create button** to start this project.
6. **This opens the SQL migration dashboard; in Step 1,** complete the following parameters **to connect to the source server:**
 - **Server name: sqlvm**
 - **Authentication type: Windows Authentication**
 - **Encrypt connection: Yes**
 - **Trust server certificate: Yes**



7. **This will detect the SimplCommerce SQL database running on the SQLVM.** Since you already executed the assessment in the previous lab, **deselect the option to assess database.** Click **Next** to continue to the next step.

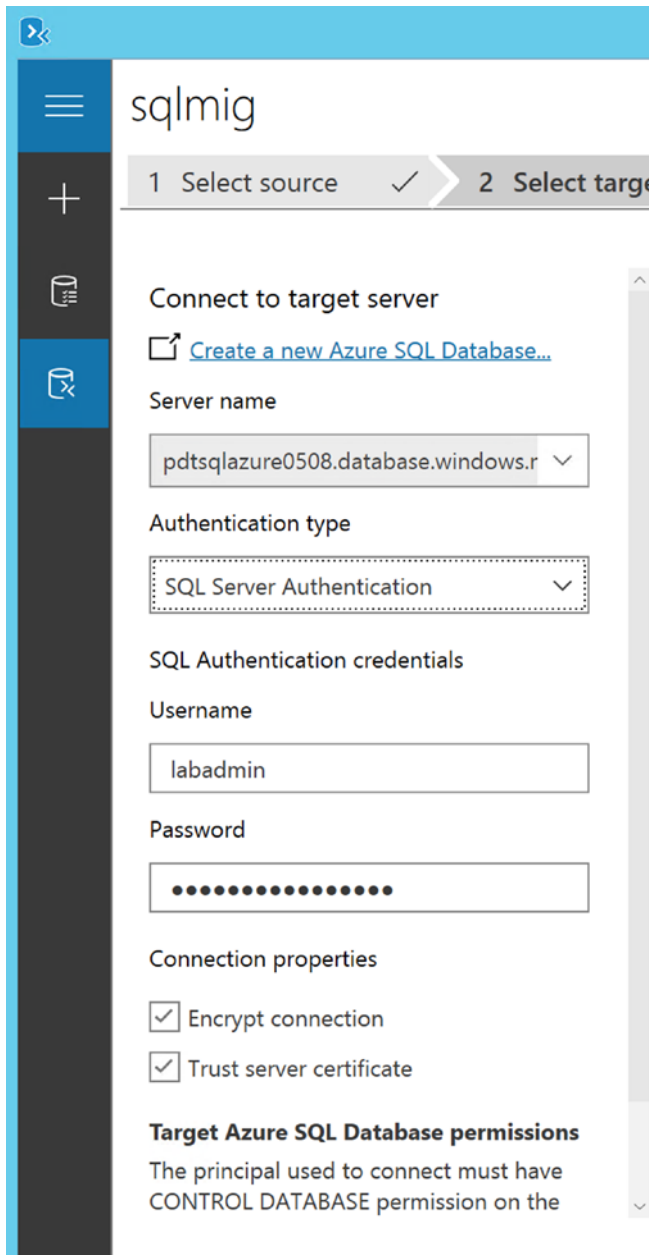
Select a single database from your source server to migrate to Azure SQL Database.

If you skip assessing the databases before migration, DMA will not be able to detect the specific schema objects that may fail to deploy on the Skip this option if you have already done the assessment and addressed the objects with breaking changes prior to the migration.

| Name | Compatibility Level | Assess database before migration? |
|--|---------------------|-----------------------------------|
| <input checked="" type="radio"/> SimplCommerce | 120 | <input type="checkbox"/> |

8. **In Step 2,** complete the following parameters:
 - **Server name:** SQL Azure server name ([suffix]sqlazure.database.windows.net)
 - **Authentication type:** SQL Server Authentication

- **Username:** labadmin
- **Password:** L@BadminPa55w.rd
- **Encrypt connection:** Yes
- **Trust server certificate:** Yes



9. This detects the SQL Azure database instance you created earlier.

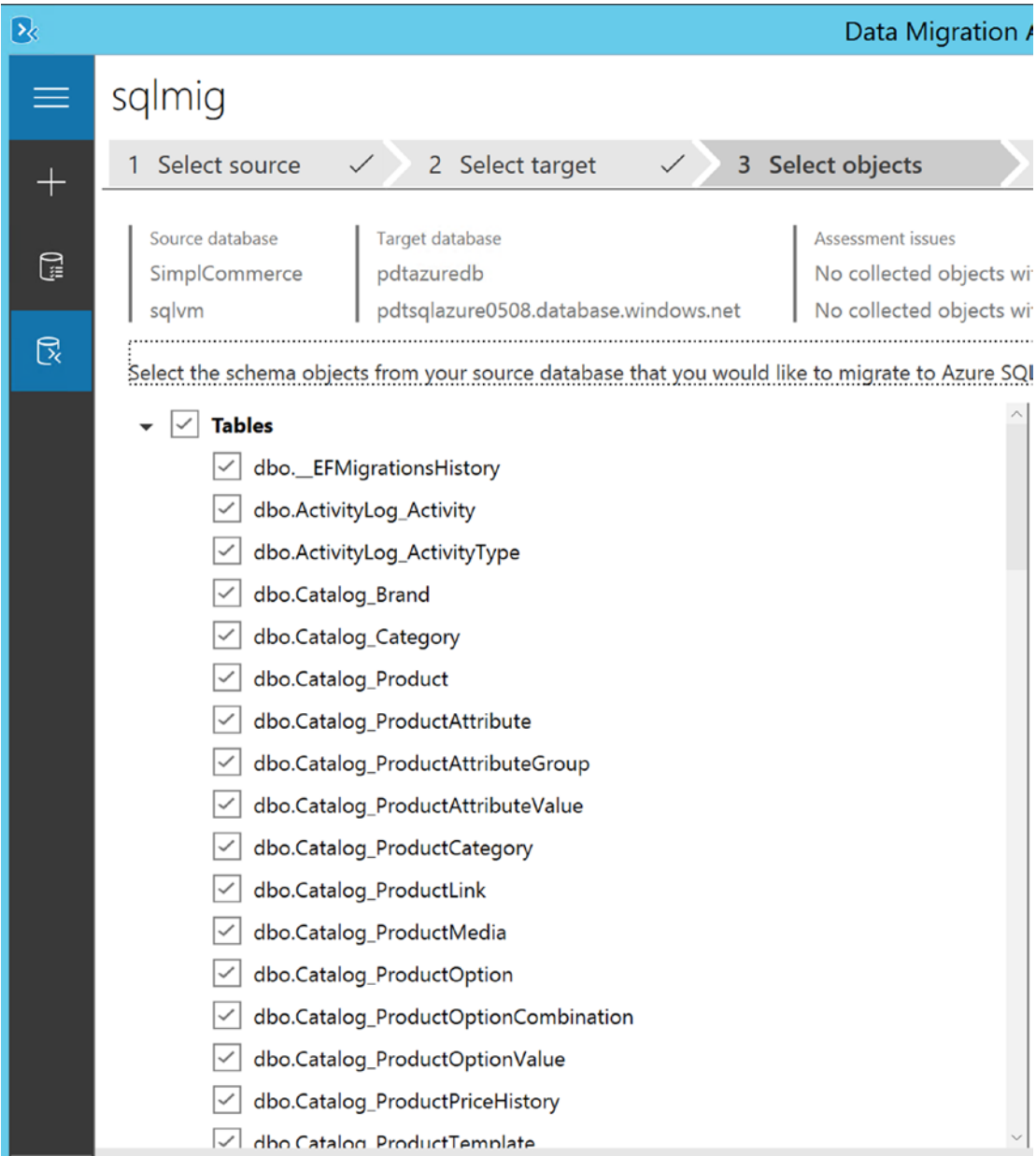
Select a single target database from your target Azure SQL Database server. If you intend to migrate Windows users, make sure the target exte

Target external user domain name

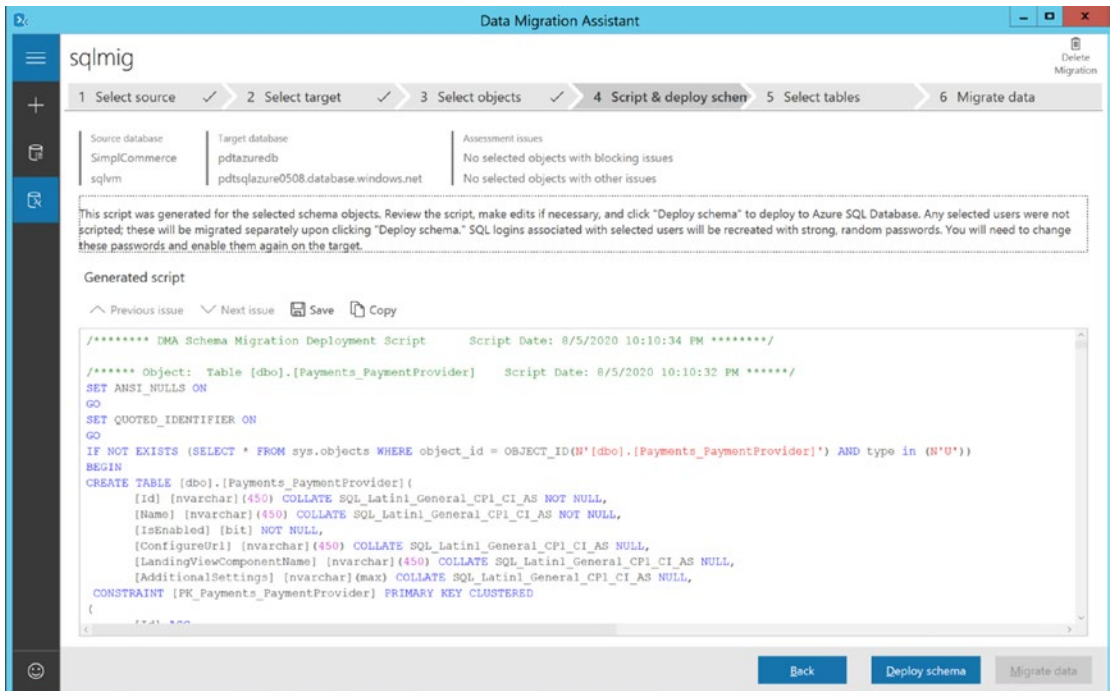
e.g. microsoft.com or contoso.com

| Name | Compatibility Level |
|---|---------------------|
| <input checked="" type="radio"/> pdtazuredb | 150 |

10. **Click “Next”** to continue.
11. This brings you to Step 3. By default, all tables are selected, which is ok for our scenario.



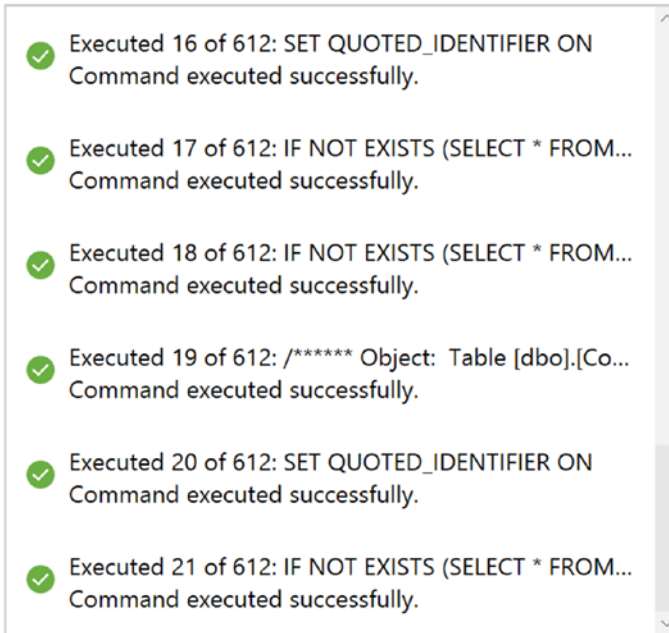
12. Click the “Generate SQL Script” button.



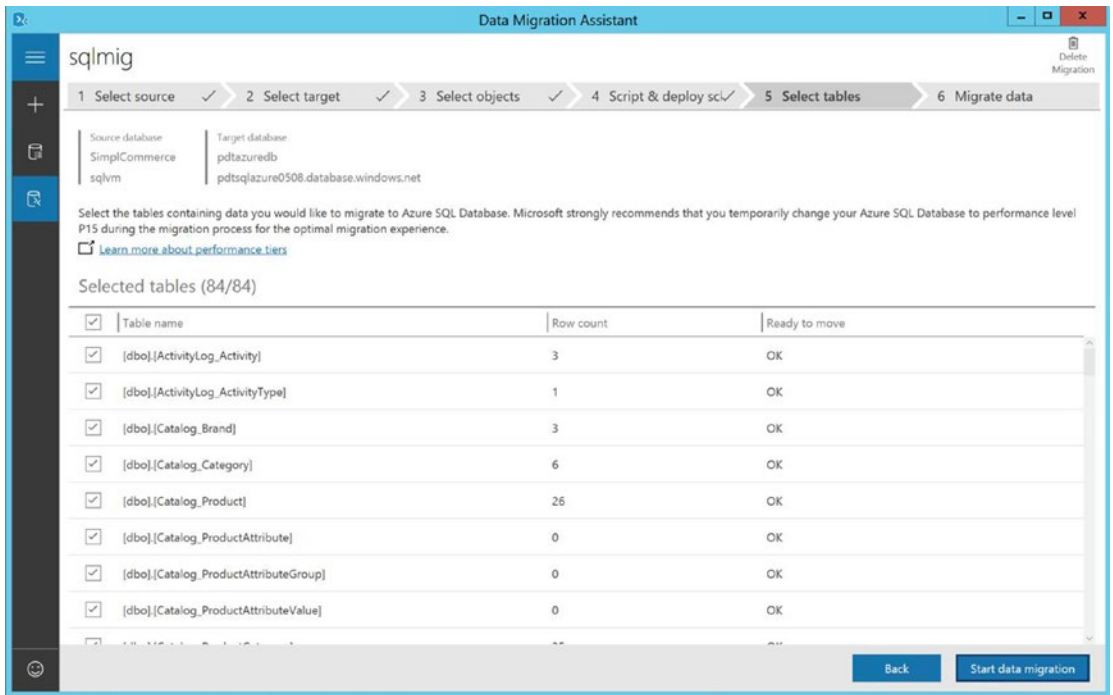
13. To run the actual migration, starting with the database schema, click “Deploy schema.”

Deployment results (21 commands executed, 0 err...

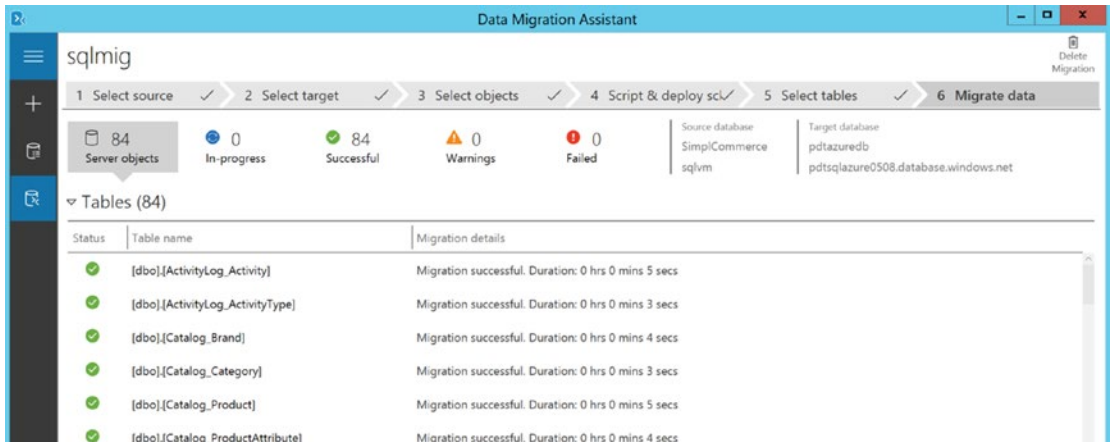
Previous error Next error Export



14. Wait for this step to complete successfully. This should take only a few minutes.
15. **Lastly**, click the **“Migrate Data”** button to start the actual database content migration. This will first show a list of tables; make sure all tables are selected here to not miss any data.

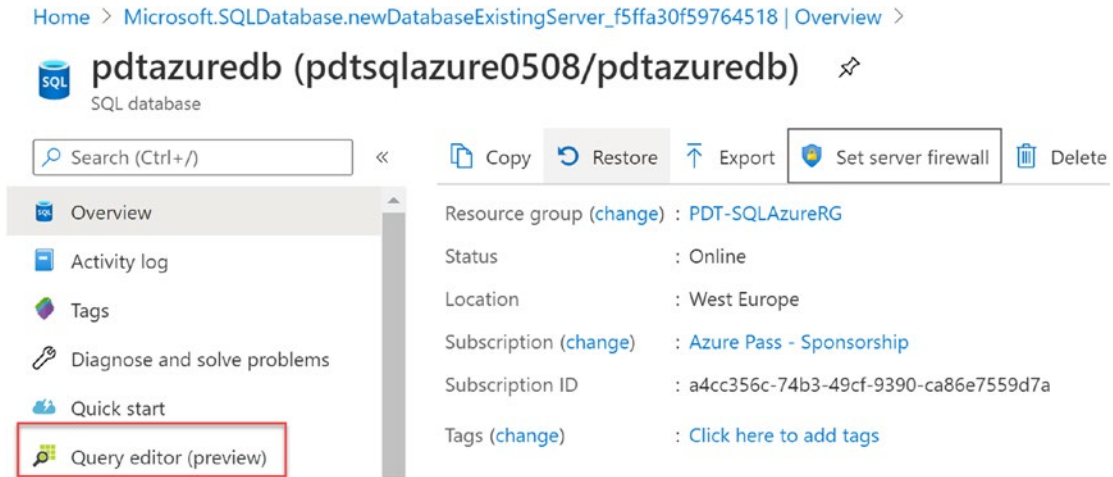


16. **And confirm**, by clicking the **Start data migration** button.



17. Wait for this process to complete successfully; this should only take a few minutes, given the rather small-sized sample database.

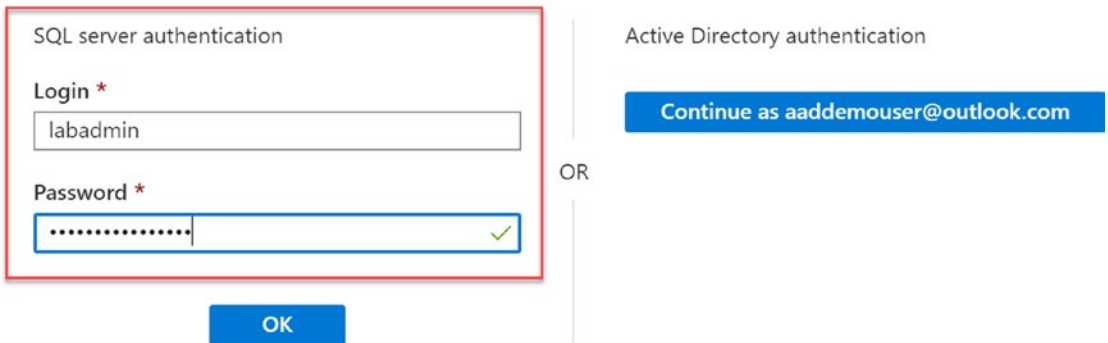
- 18. Once complete, **close the Data Migration Assistant**, without saving the changes.
- 19. **Return** to the Azure Portal, and **browse to the SQL Azure database** that just got migrated. From the SQL database blade, **select “Query editor (preview).”**



- 20. **Enter** the SQL Azure administrative credentials you defined earlier (default = labadmin and L@BadminPa55w.rd).



Welcome to SQL Database Query Editor



21. **You are prompted** with another security warning; although you are connecting from the browser, the SQL server and database connection is “seen” as a SQL connection (port 1433) and not an HTTPS (port 443) connection. Therefore, **you need to add your client IP to the list of firewall exceptions**, similar to what you did for the WebVM.

SQL server authentication

Login *

labadmin

Password *

..... ✓

✘ Cannot open server 'pdtsqlazure0508' requested by the login. Client with IP address '5.148.105.110' is not allowed to access the server. To enable access, use the Windows Azure Management Portal or run `sp_set_firewall_rule` on the master database to create a firewall rule for this IP address or address range. It may take up to five minutes for this change to take effect.

[Set server firewall \(pdtsqlazure0508\)](#)

OK

22. Click “Set server firewall” [suffixsqlazure].

Home > Microsoft.SQLDatabase.newDatabaseExistingServer_f5ffa30f59764518 | Overview >



Firewall settings

pdtsqlazure0508 (SQL server)



Save



Discard



Add client IP

- 23. Click “+ Add client IP,” which automatically detects your own client public IP address (JumpVM or your own Internet public IP address if running this from your own machine).

Client IP address 5.148.105.110

| Rule name | Start IP | End IP | |
|----------------------------|----------------------|----------------------|-----|
| <input type="text"/> | <input type="text"/> | <input type="text"/> | ... |
| ClientIPAddress_2020-8-... | 5.148.105.110 | 5.148.105.110 | ... |
| allow_webVM | 137.116.222.152 | 137.116.222.152 | ... |


- 24. The Rule base got updated with your ClientIPAddress rule; save the changes.
- 25. From the Azure Portal breadcrumbs link, select the SQL Azure database.



26. This brings you back to the SQL Azure database connection blade.

Click OK to set up the connection. This is successful this time.

Home > Microsoft.SQLDatabase.newDatabaseExistingServer_f5ffa30f59764518 | Overview


 **pdtazuredb (pdtsqlazure0508/pdtazuredb) | C**
SQL database


Search (Ctrl+/) << Login + New Query ↑ Open query

- Overview
- Activity log
- Tags
- Diagnose and solve problems
- Quick start
- Query editor (preview)**

Power Platform

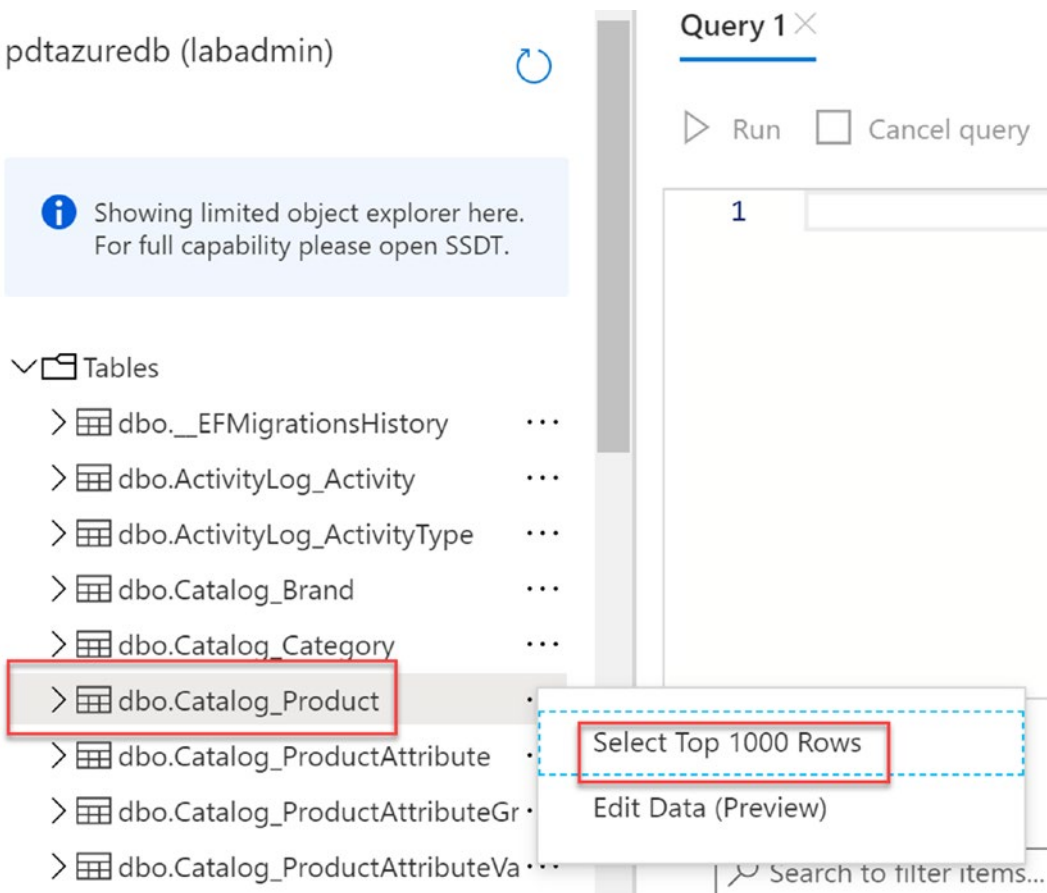
- Power BI (preview)

pdtazuredb (labadmin) 

 Showing limited object explorer here. For full capability please open SSDT.

- > Tables
- > Views
- > Stored Procedures

27. **Click the “>” sign** left to **Tables**, to open the list of tables in the database.



28. From the list of tables, **select `dbo.Catalog_Product`**. **Click the ellipsis (the three dots)** next to it, to open the context menu. Here, **click “Select Top 1000 Rows.”** This adds a new query2 item and runs it. The following shows the actual content of the products table more.

Query 1 × Query 2 ×

Run Cancel query Save query Export data as Show only Editor

```
1 SELECT TOP (1000) * FROM [dbo].[Catalog_Product]
```

Results Messages

Search to filter items...

| Id | Name | Slug | MetaTitle | MetaKe |
|----|----------------------------|----------------------------|-----------|--------|
| 1 | Lightweight Jacket | lightweight-jacket | | |
| 2 | Lightweight Jacket M Black | lightweight-jacket-m-black | | |
| 3 | Lightweight Jacket M Gray | lightweight-jacket-m-gray | | |
| 4 | Lightweight Jacket L Black | lightweight-jacket-l-black | | |
| 5 | Lightweight Jacket L Gray | lightweight-jacket-l-gray | | |
| 6 | Lightweight Jacket S Black | lightweight-jacket-s-black | | |

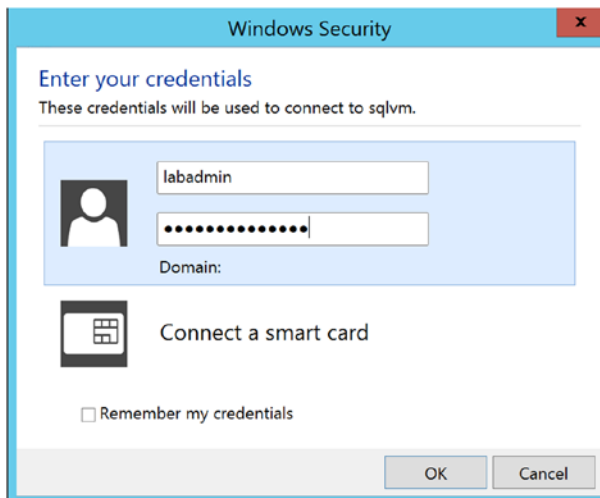
Query succeeded | 8s

29. This confirms the SQL Azure database is running as expected and confirms a successful migration once more.

Task 3 (Optional): Using SQL Server Management Studio to migrate from SQLVM to a SQL Azure instance

1. If your DBA team is familiar with SQL Server Management Studio, know they can keep using this tool to perform the actual SQL database migration as well. To use this method, open an **RDP session** to the **WebVM** (labadmin and L@BadminPa55w.rd).

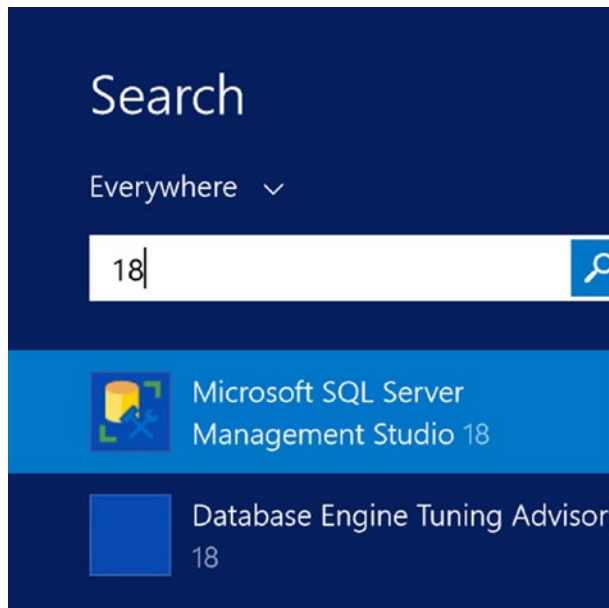
2. Next, from within the RDP session of the WebVM, open a second RDP session to the SQLVM machine (remember, the SQLVM has no public IP address, not making it reachable from the outside) by running **mstsc.exe** from the Start menu.
3. As server name, type **“SQLVM”**. (Since both virtual machines are in the same Azure Virtual Network and subnet, the server name resolution works.). **Click Connect**.



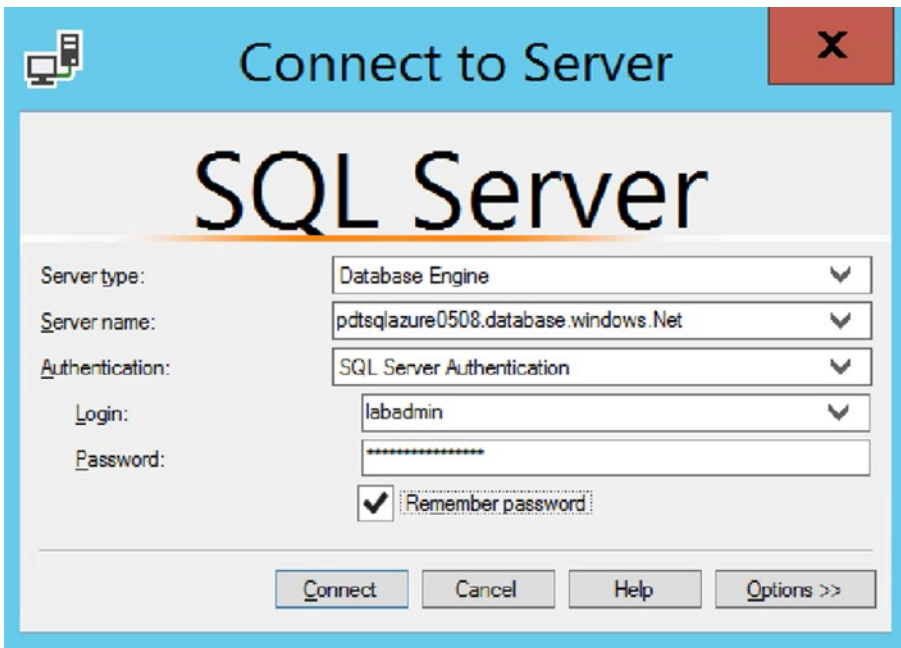
4. Provide the local admin credentials of the SQLVM virtual machine:
 - labadmin
 - L@BadminPa55w.rd

And confirm with **OK**.

5. Once you **are logged on** to the **SQL Server virtual machine** (notice the SQL Getting Started shortcut on the desktop), click the **Start button**. **Start typing “18”**; this will resolve several management tools available on the server. Notice **Microsoft SQL Server Management Studio 18**.

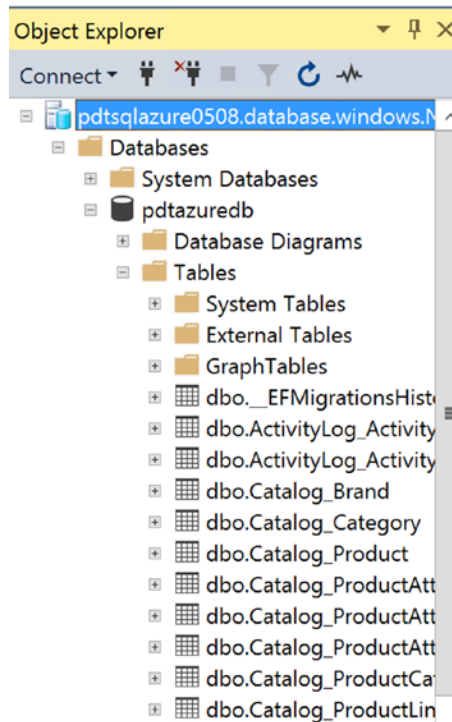


6. Select it to start the **SQL Server Management Studio 18 console**.
7. Once opened, you are asked for **server connection information**. Provide the following settings:
 - **Server name: SQL Azure server name ([suffix]sqlazure<date>.database.windows.net**
 - **Authentication: SQL Server Authentication**
 - **Login: labadmin**
 - **Password: L@BadminPa55w.rd**

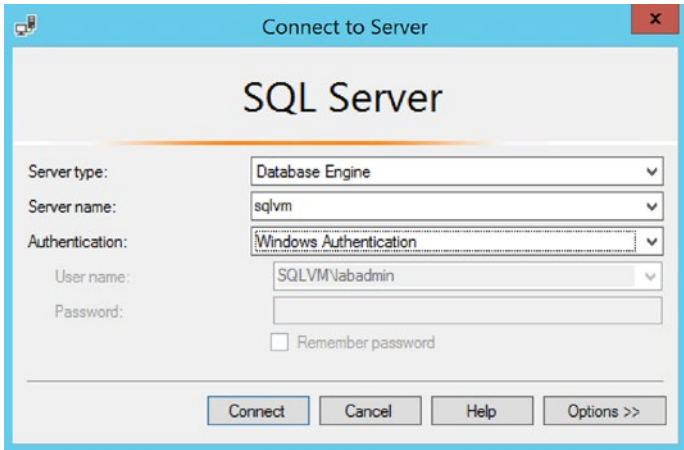


Note The reason this connection succeeds from an “internal” SQLVM that is not internet-facing is because we set the “Allow Azure services and resources to access this server” on SQL Azure level during the initial deployment. In a real-life scenario, you would need to configure the SQL Azure firewall and virtual network settings to allow hybrid connectivity between your on-premises infrastructure and SQL Azure, integrating with Site to Site VPN or ExpressRoute Networking.

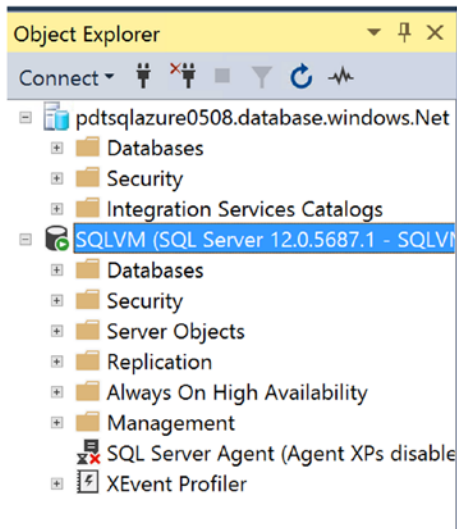
Click Connect to log on to this SQL Server instance.



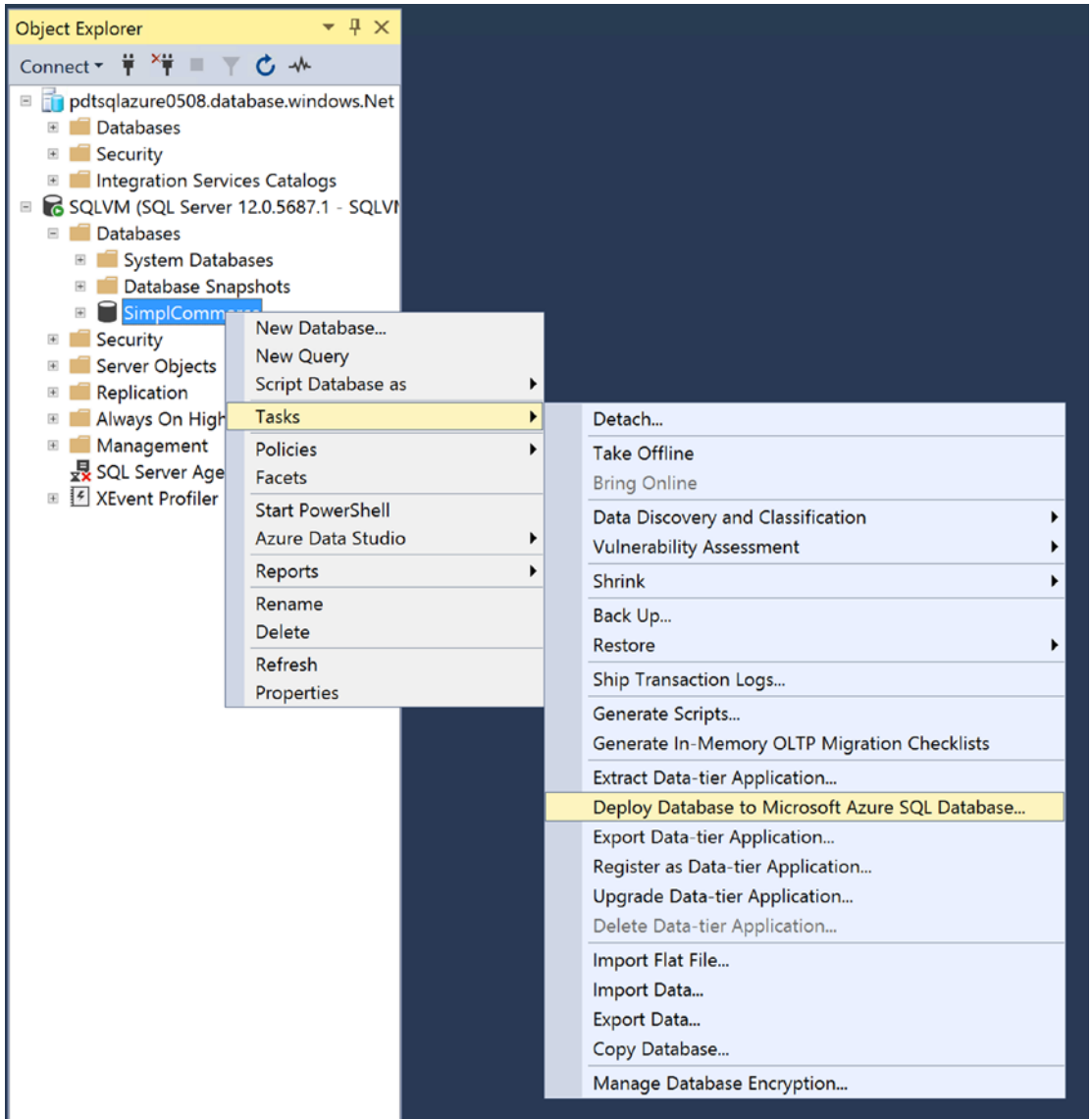
8. In order to have a connection to the SQLVM database instance, we need to add another connection. From the SQL Server Management Studio console, click **File ► Connect Object Explorer**. In the **Connect to server** popup that appears, this time provide the server credentials from the SQLVM:
 - **Server name: sqlvm**
 - **Authentication: Windows Authentication**



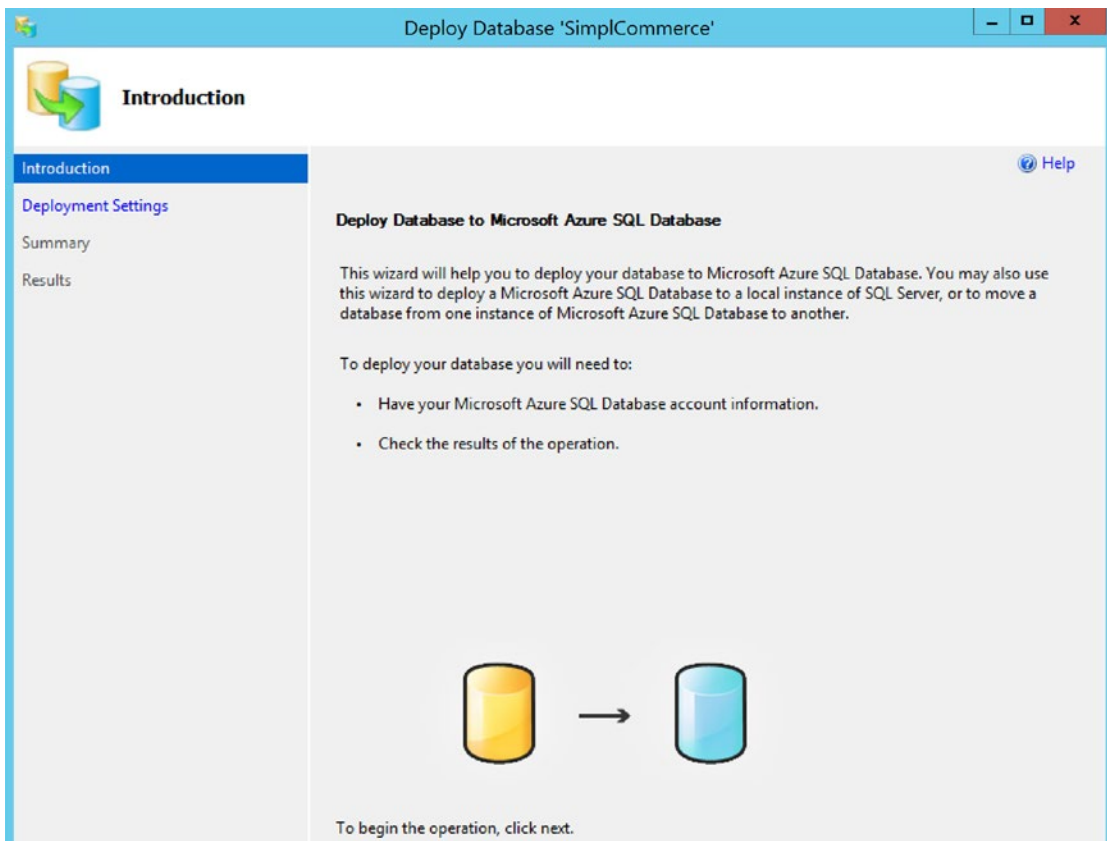
- 9. **Click the Connect button.** (If you get an unsuccessful connection error because of certificate chain not trusted, click the **Options** button and select to **Trust Certificate.**)
- 10. The **Object Explorer** shows a successful connection to both databases now. If you open the Databases level, you should see the **SimplCommerce** database.



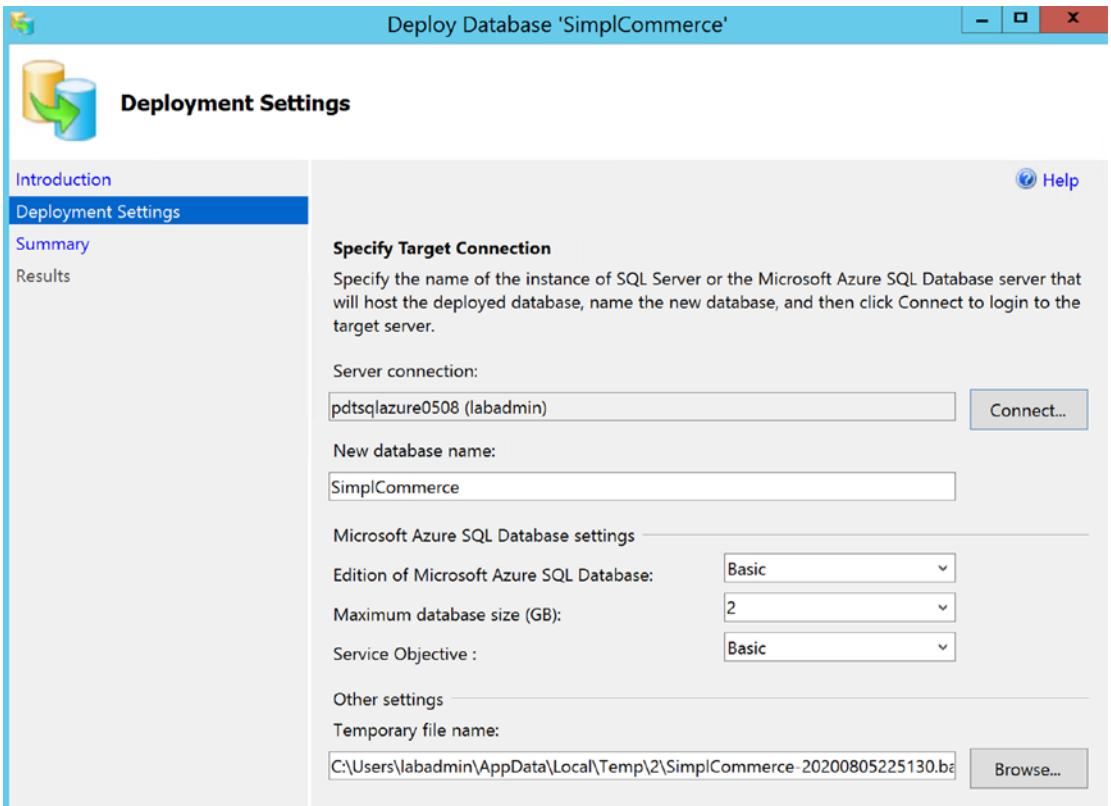
11. The next step is running the actual migration of the database. Therefore, **select** the **SimplCommerce** database on the **SQLVM**, **right-click** it, select **Tasks**, and select **Deploy Database to Microsoft SQL Azure Database**.



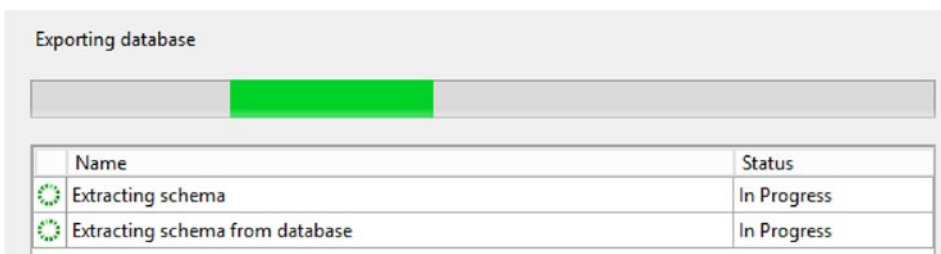
12. **Click** the **Next** button when you see the **Introduction** step showing up.



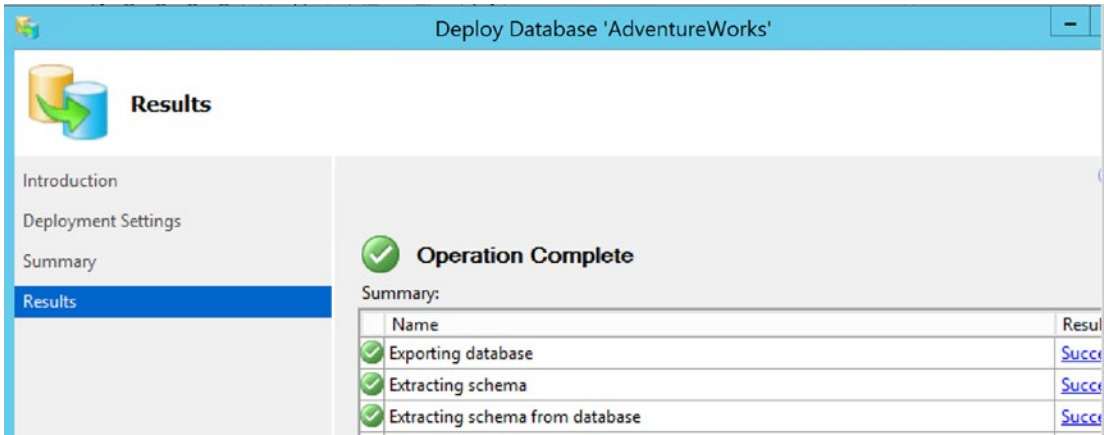
13. **In the Deployment Settings, provide the Server connection by clicking the Connect button. Provide the following details here:**
- **Server connection: <your SQL Server in Azure>[suffix] sqlazure<date>.database.windows.net**
 - **SQL Authentication (+provide credentials labadmin and L@BadminPa55w.rd)**
 - **New database name: SimplCommerce**
 - **Edition of Microsoft SQL Database: Basic**
 - **Max DB size: 2 GB**
 - **Service Objective: Basic**



14. **Read** through the settings in the summary step. **Click the Finish** button to start the actual move process.



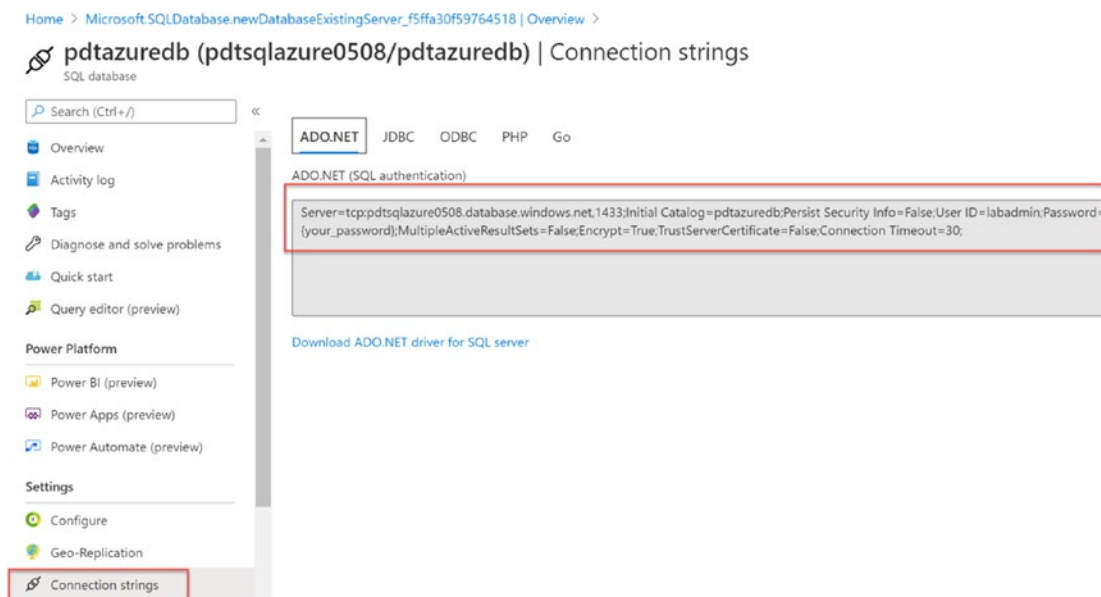
15. **Wait** for this process to complete – this should only take a few minutes.



16. Once completed, close the migration window.
17. This completes the task of migrating a SQL Server database to SQL Azure using SQL Server Management Studio.

Task 4: Defining a hybrid connection from a WebVM to an Azure SQL database

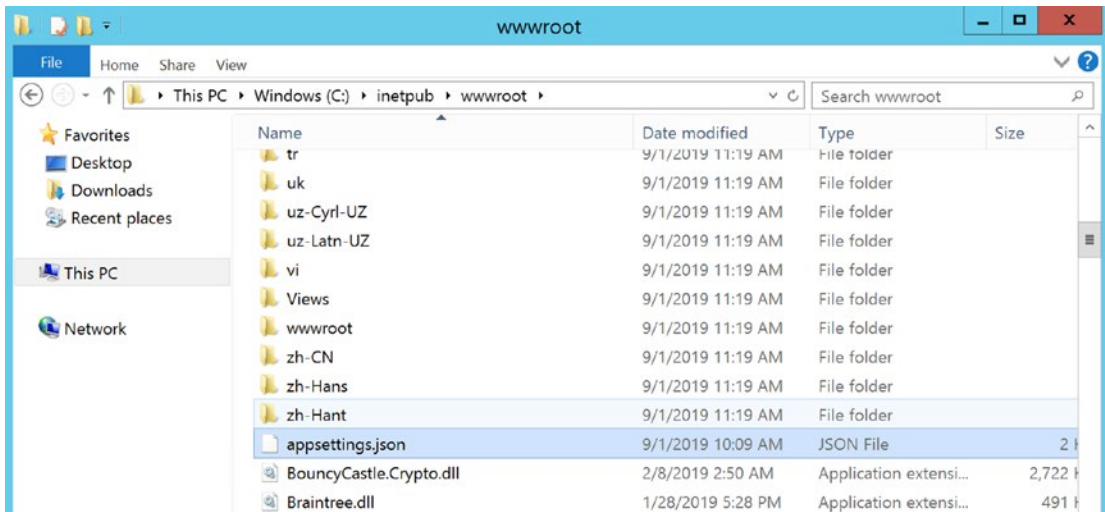
1. To complete our hybrid cloud migration, we will now update the Connection strings settings in the appsettings.json file of our WebVM web application. This information can be retrieved from the SQL database settings in the Azure Portal. **From within the SQL database detailed blade**, browse to **Connection strings** under the **Settings** section.



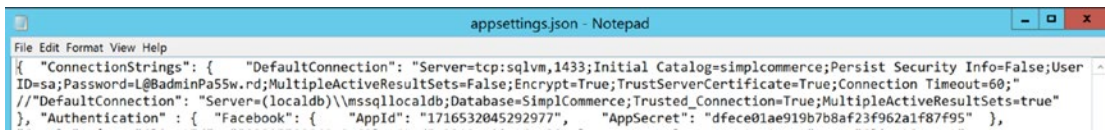
2. Leave this information on screen, or copy it into a temp text file, as you will need to copy parts of the ADO.NET connection string information into the web server's web.config file.
3. **Go back to the WebVM** virtual machine Remote Desktop session (or open it again when you already closed the WebVM RDP session).
4. Browse to the IIS web server folder that has the web application content:


```
c:\inetpub\wwwroot\
```

 Open the file **appsettings.json** with Notepad.



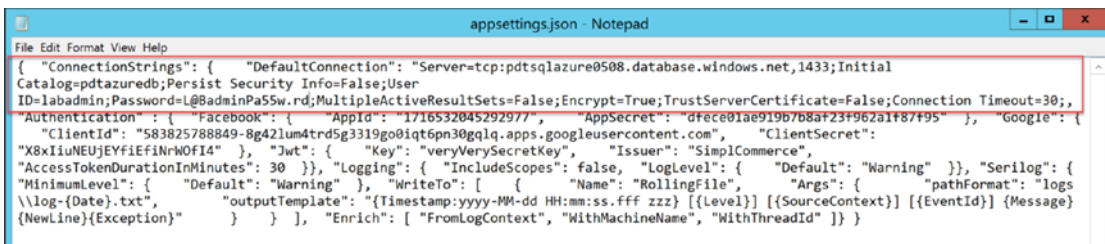
5. Go to the section that starts with “**ConnectionStrings**”.



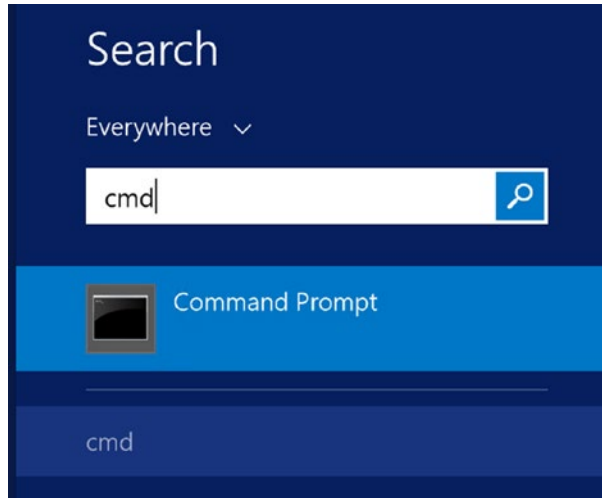
6. **Replace the** following settings with the parameters from the connection string information in the Azure Portal:

- **Server=tcp:sqlvm=>** **Change** the sqlvm to <Azure SQL server name>, nopsplus.database.windows.net in our example.
- **Uid=sa =>** **Change** the sa account to **labadmin**.

Save the changes to the **appsettings.json** file.

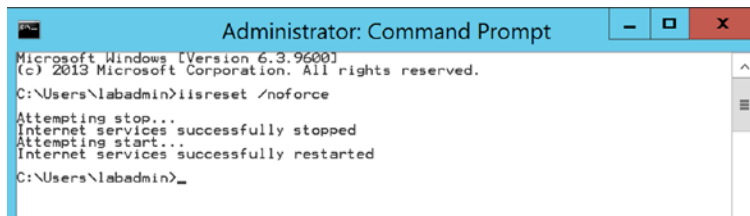


7. **From the Start screen on the WebVM, open a command prompt, by typing “CMD”.**

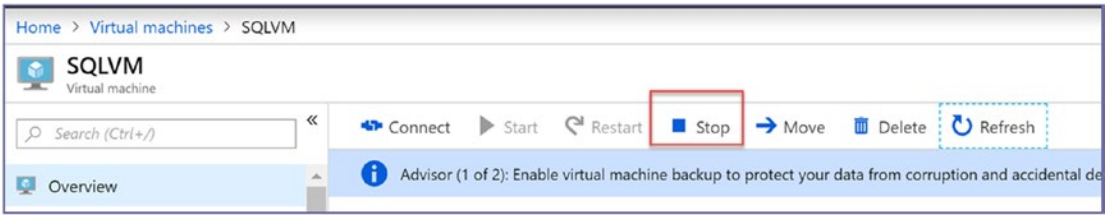


8. **In the command prompt, run the following command, to restart the IIS web server service:**

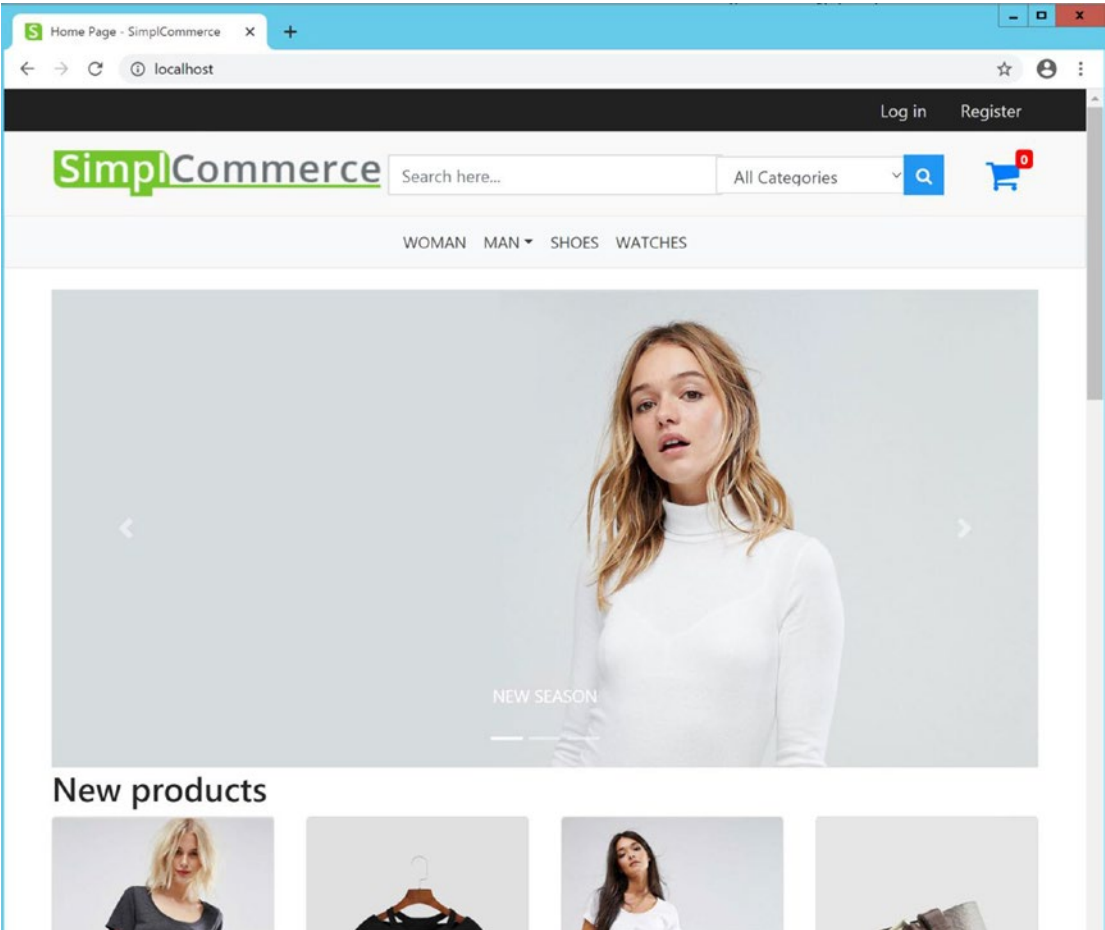
```
iisreset /noforce
```



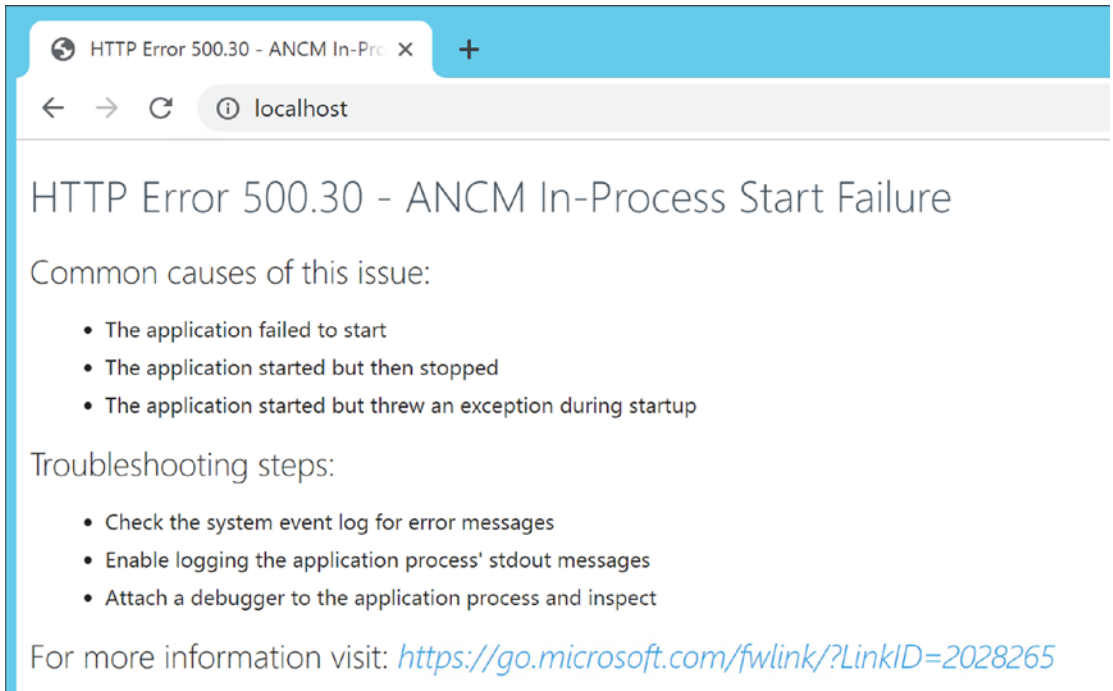
9. To prove that the web application is now connected to the Azure SQL database, let's shut down the SQLVM. **From the Azure Portal, navigate to Virtual machines, and click the SQLVM virtual machine.**
10. From the **SQLVM** detailed blade, **click the Stop** button in the top menu. Wait for the notification message, telling you the VM has shut down.



- 11. To test if the web application is now connected to the Azure SQL database, browse to the website from within the WebVM's browser, connecting to **localhost**.
- 12. The website should load successfully and show you the product catalog list.



13. If you receive an error message in the browser, similar to the following screenshot, it means there is something wrong with the SQL database connection. Verify your settings again in the **appsettings.json** file, and **run IISreset again** from the command prompt.



14. This completes this lab.

Summary

In this lab, you learned how to deploy an Azure SQL Server resource, as well as how to migrate a SQL database using Azure SQL Data Migration Assistant and/or the SQL Server Management Studio 18. You updated the IIS web server `appsettings.json` file and validated the web application is now running in a hybrid setup.

CHAPTER 6

Lab 4: Deploying an Azure Web App and Migrating from WebVM

Lab 4: Deploying an Azure Web App and migrating from WebVM

What You Will Learn

In this lab, you will publish your dotnetcore application source code to an Azure Web App, out of Visual Studio 2019, sometimes described as “right-click publish.”

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

In a later lab exercise, you will deploy the same web application using DevOps concepts.

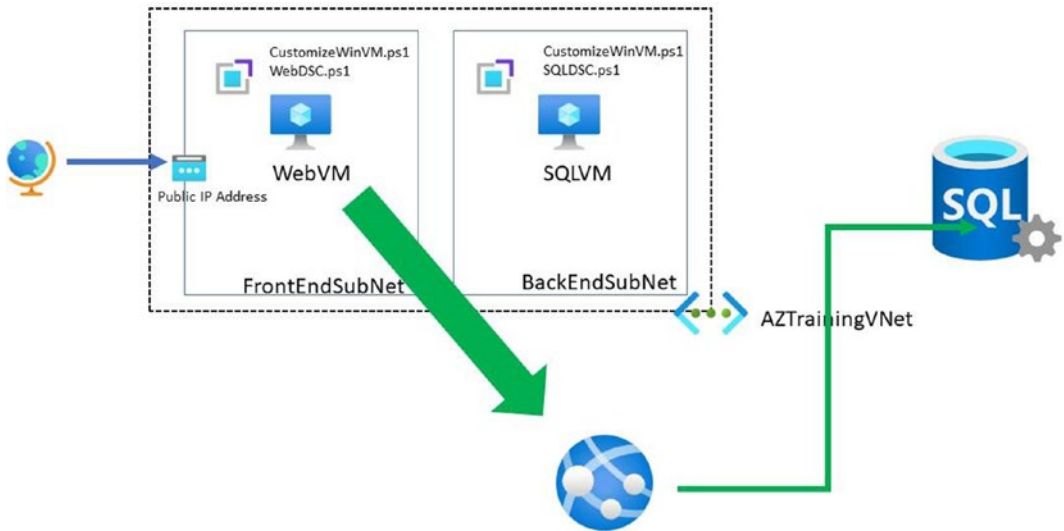
Time Estimate

This lab is estimated to take 45 min in total.

Prerequisites

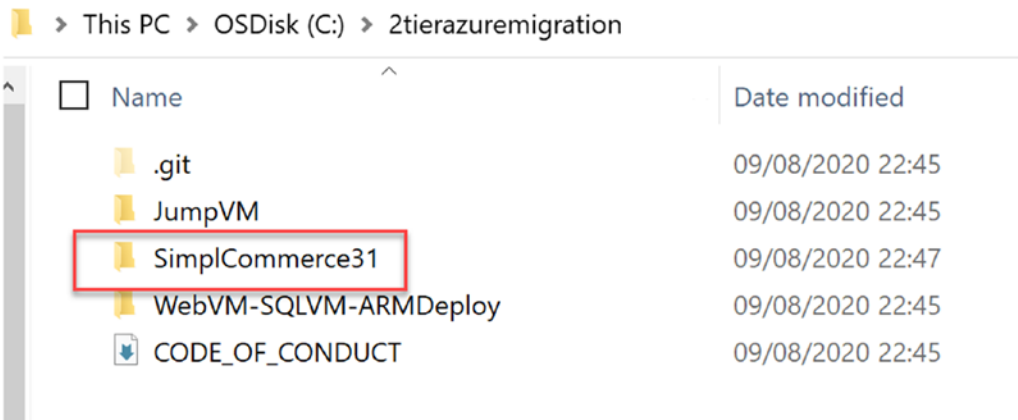
Make sure you completed Labs 1, 2, and 3 before starting this exercise.

Scenario Diagram



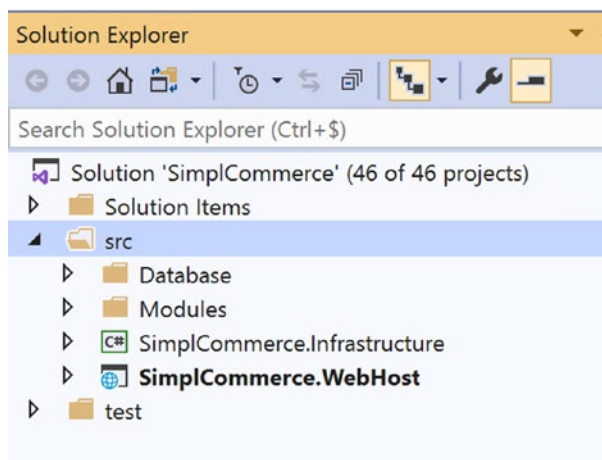
Task 1: Publish an ASP.NET project to Azure Web Apps from Within Visual Studio 2019

1. **Log on** to the lab jumpVM virtual machine (for your information, credentials labadmin and L@BadminPa55w.rd), or your own developer workstation, having Visual Studio 2019 with the latest updates running.
2. From the **lab jumpVM**, browse to the folder that holds the GitHub downloaded source files (default location = C:\2TierAzureMigration).
3. Here, **open the subfolder “SimplCommerce31”**; this folder contains all necessary coding files for the SimplCommerce webshop application we are using.

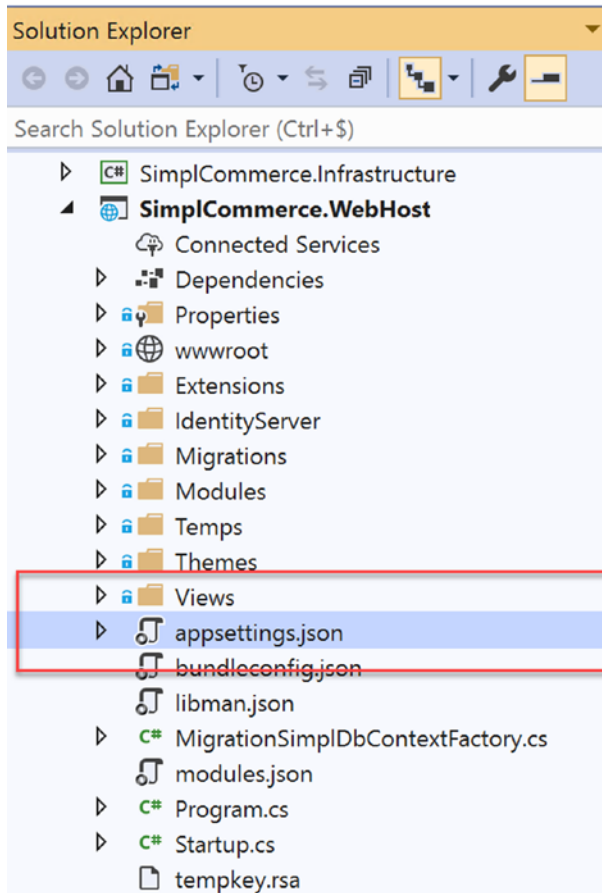


Note This folder contains more source files than what we need in this lab, but don't delete those, as you will use some of those in the labs coming.

4. **Open** the file **SimplCommerce.sln**, which should open your Visual Studio 2019 development environment.



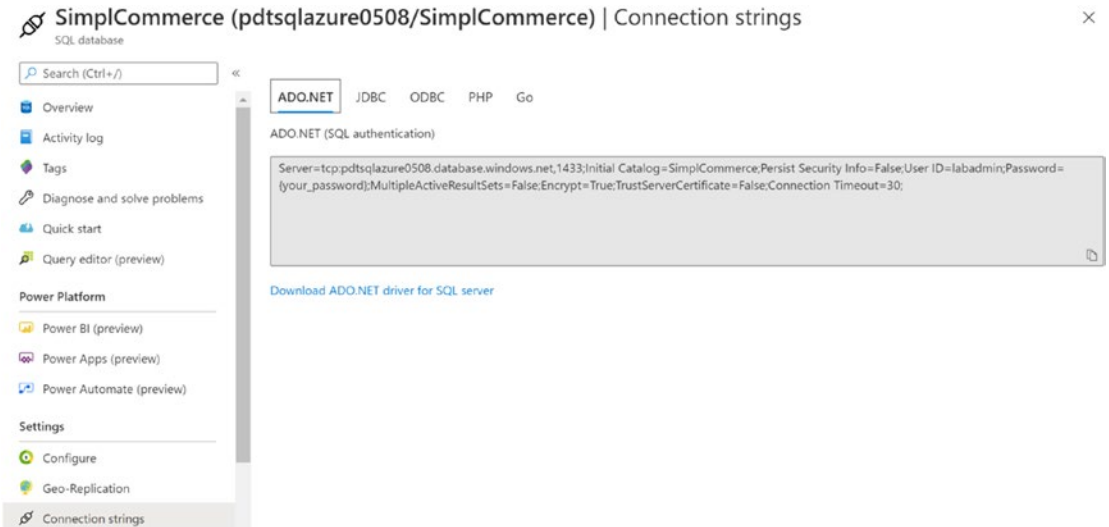
5. Under the **SimplCommerce.WebHost** solution, notice the **appsettings.json** file.



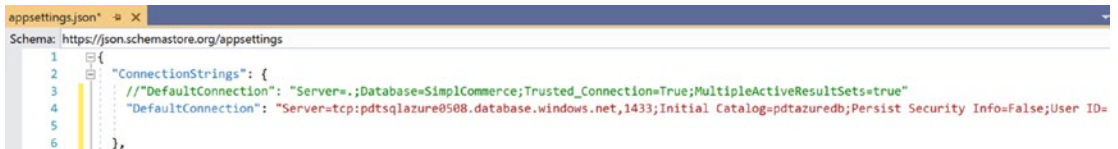
6. Open this file in the Visual Studio editor.

```
Schema: https://json.schemastore.org/appsettings
1  {
2    "ConnectionStrings": {
3      "DefaultConnection": "Server=.;Database=SimplCommerce;Trusted_Connection=True;MultipleActiveResultSets=true"
4    },
5    "Authentication": {
6      "Facebook": {
7        "AppId": "1716532045292977",
8        "AppSecret": "dfece01ae919b7b8af23f962a1f87f95"
9      },
10     "Google": {
11       "ClientId": "583825788849-8g42lum4trd5g3319go0iq6pn30gqlq.apps.googleusercontent.com",
12       "ClientSecret": "X8xIiuNEUjEYfiEFiNrW0FI4"
13     },
14   },
15   "Logging": {
16     "IncludeScopes": false,
17     "LogLevel": {
18       "Default": "Warning"
19     }
20   }
21 }
```

7. In order to make our webshop work, we need to update the database connection string from the current SQLite configuration to the SQL Azure database connection string.
8. **From the Azure Portal**, browse to the **SQL Azure database** you migrated earlier ([suffix]azuredb), and **open its Connection strings** settings.



9. **Copy the ADO.NET** connection string, and replace the **DefaultConnection** parameter in the appsettings.json file as shown in the following example.



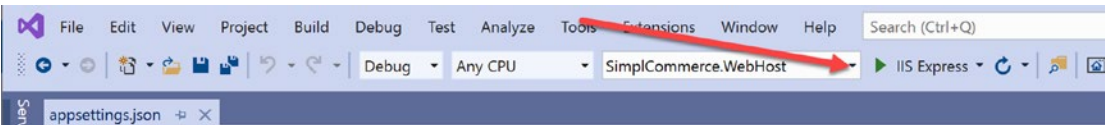
Note The formatting of the connection string might get “lost” when copying; easiest to bypass this issue is pasting it in Notepad first, before copy/pasting it directly into the VS editor.

As a reference, this is what the connection string should look like in full (all needs to be on one single line in the JSON), based on my setup:

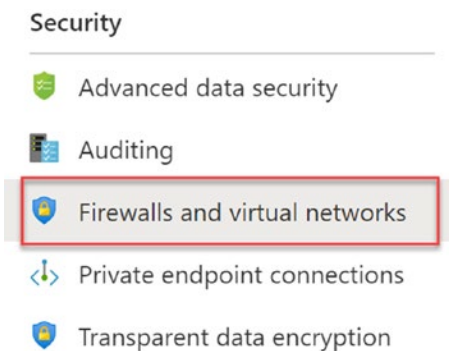
```
"DefaultConnection": "Server=tcp:simplcsqldt.database.windows.net,1433;Initial Catalog= simplcommercedb;Persist Security Info=False;User ID=pdtadmin;Password=L@BadminPa55w.rd;MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;"
```

Also make sure you replace the {yourpassword} string with the actual password as shown in the preceding example.

- 10. **Save** the changes made to the **appsettings.json**.
- 11. Let's validate the webshop app is working fine on the development station, by **starting it in Debug mode**.



Note If you are running this lab from within the JumpVM, you need to allow the public IP from this 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.

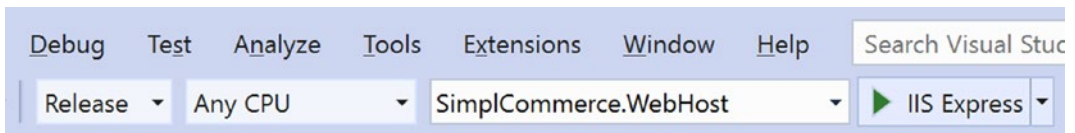


12. Add a new rule, named “allow_jumpVM,” having the JumpVM’s public IP address in the Start IP and End IP fields.

Client IP address 5.148.105.110

| Rule name | Start IP | End IP | |
|-------------------------------|----------------------|----------------------|-----|
| <input type="text"/> | <input type="text"/> | <input type="text"/> | ... |
| allow_jumpVM | 13.93.75.106 | 13.93.75.106 | ... |
| allow_webVM | 137.116.222.152 | 137.116.222.152 | ... |
| ClientIPAddress_2020-8-5_2... | 5.148.105.110 | 5.148.105.110 | ... |

13. Switch back to your Visual Studio environment, and run the application by pressing “F5” or clicking the “IIS Express” link in the top menu

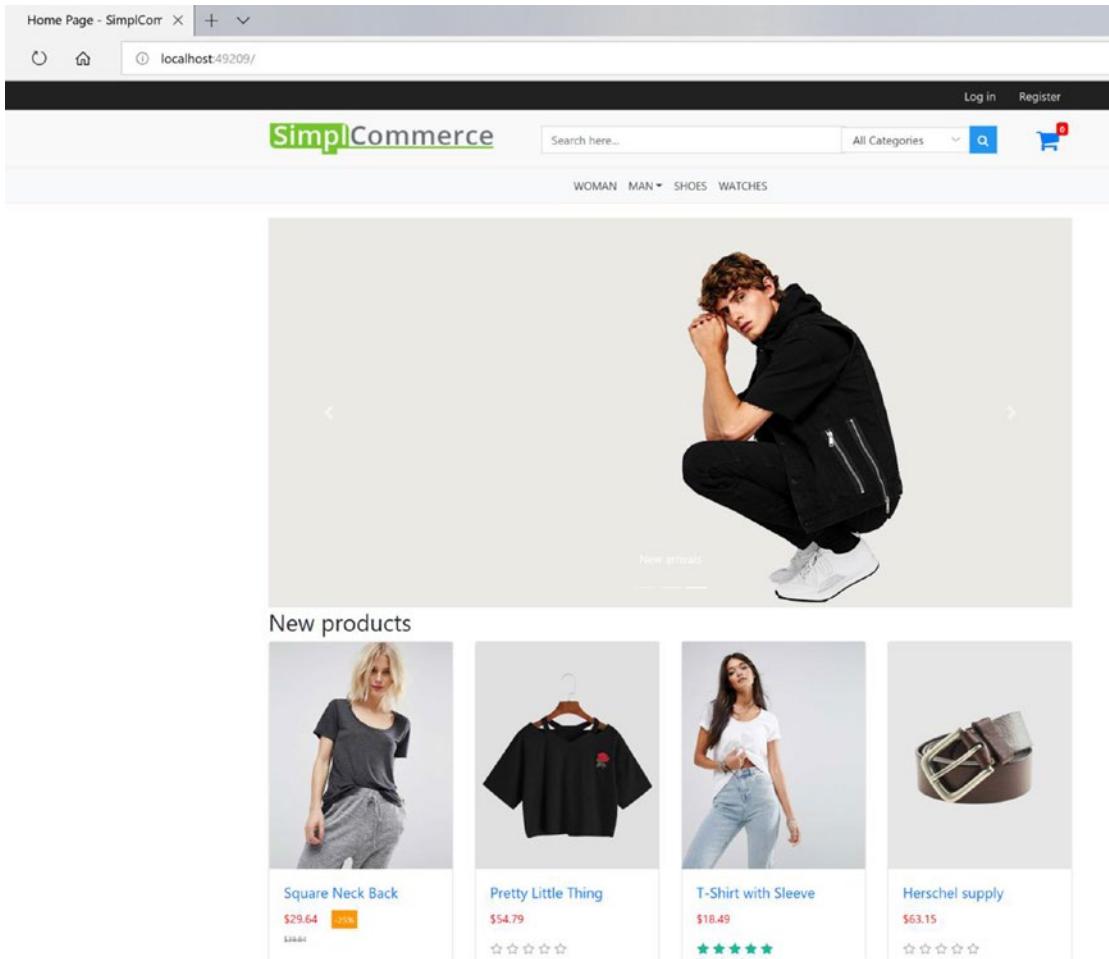


14. This compiles the application, showing debug information in the **Output window**, similar to the following screenshot (this is just a capture from during the debug; it doesn’t need to be the exact same).

```

Output
Show output from: Build
26>Done building project "SimplCommerce.Module.EmailSenderSmtplib.csproj".
27>----- Build started: Project: SimplCommerce.Module.ShippingTableRate, Configuration: Debug Any CPU -----
25>SimplCommerce.Module.PaymentCashfree -> C:\2TierAzureMigration\SimplCommerce-master\src\Modules\SimplCommerce.Module.PaymentCashfree\bin\Debug\netcoreapp3.1
\SimplCommerce.Module.PaymentCashfree.dll
25>SimplCommerce.Module.PaymentCashfree -> C:\2TierAzureMigration\SimplCommerce-master\src\Modules\SimplCommerce.Module.PaymentCashfree\bin\Debug\netcoreapp3.1
\SimplCommerce.Module.PaymentCashfree.View.dll
25>Done building project "SimplCommerce.Module.PaymentCashfree.csproj".
28>----- Build started: Project: SimplCommerce.Module.StorageLocal, Configuration: Debug Any CPU -----
27>C:\2TierAzureMigration\SimplCommerce-master\src\Modules\SimplCommerce.Module.ShippingTableRate\Services\TableRateShippingServiceProvider.cs(14,58,14,88): warning
CA1051: Do not declare visible instance fields
28>C:\2TierAzureMigration\SimplCommerce-master\src\Modules\SimplCommerce.Module.StorageLocal\LocalStorageService.cs(12,23,12,34): warning CA1055: Change the return
type of method LocalStorageService.GetMediaUrl(string) from string to System.Uri.
28>SimplCommerce.Module.StorageLocal -> C:\2TierAzureMigration\SimplCommerce-master\src\Modules\SimplCommerce.Module.StorageLocal\bin\Debug\netcoreapp3.1
\SimplCommerce.Module.StorageLocal.dll
27>SimplCommerce.Module.ShippingTableRate -> C:\2TierAzureMigration\SimplCommerce-master\src\Modules\SimplCommerce.Module.ShippingTableRate\bin\Debug\netcoreapp3.1
\SimplCommerce.Module.ShippingTableRate.dll
27>Done building project "SimplCommerce.Module.ShippingTableRate.csproj".
28>Done building project "SimplCommerce.Module.StorageLocal.csproj".
29>----- Build started: Project: SimplCommerce.Module.ShippingFree, Configuration: Debug Any CPU -----
30>----- Build started: Project: SimplCommerce.Module.Shipments, Configuration: Debug Any CPU -----
  
```

- 15. After about 30 seconds, the webshop will show up, confirming the application compiled fine, as well as having connectivity to the Azure SQL database we migrated earlier.



Note While off-topic for our lab scenarios, know this is a fully functional e-commerce application, allowing you to create new customers, place orders, update products, and so on if you want to extend the demo and also perform write operations to the database.

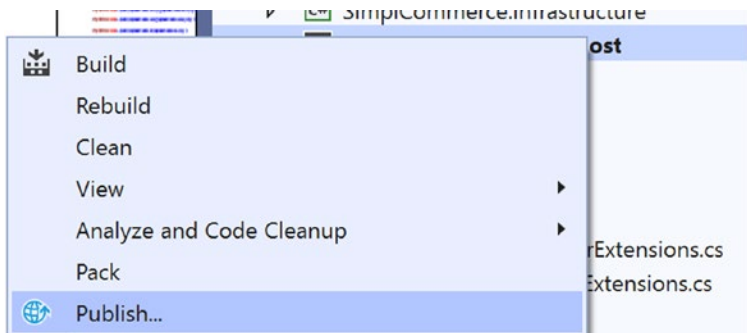
16. This confirms that our web application is working fine. You can close the browser session, which will also end the Visual Studio debugging.

This completes the first task in which you loaded a Visual Studio project, updated packages, made changes to the appsettings.json file database Connection strings settings, and ran a debug job to validate the e-commerce application is running fine.

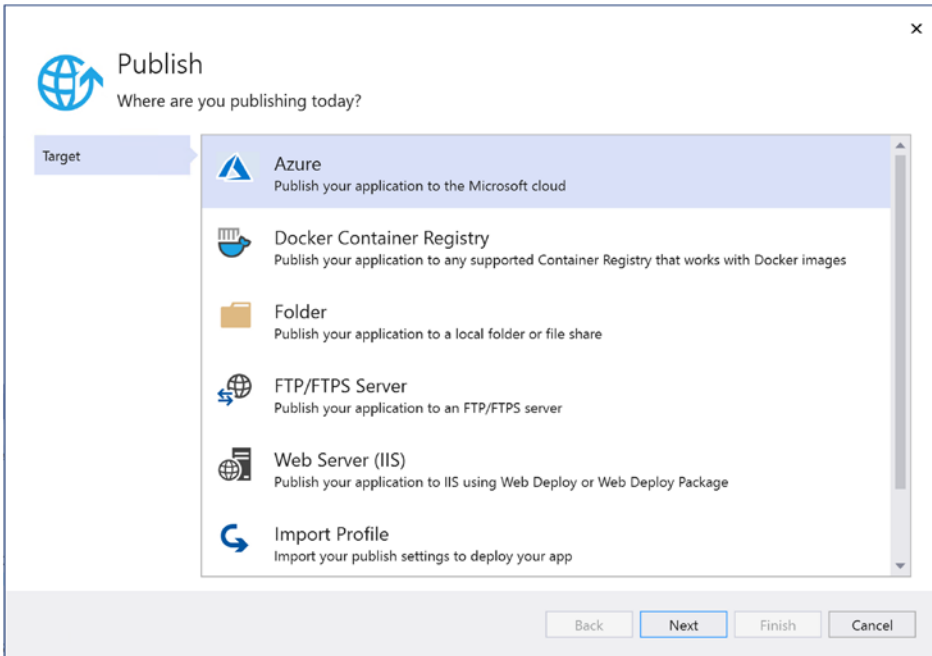
In the next task, you will publish the application to Azure Web Apps.

Task 2: Publishing the source code to Azure Web Apps

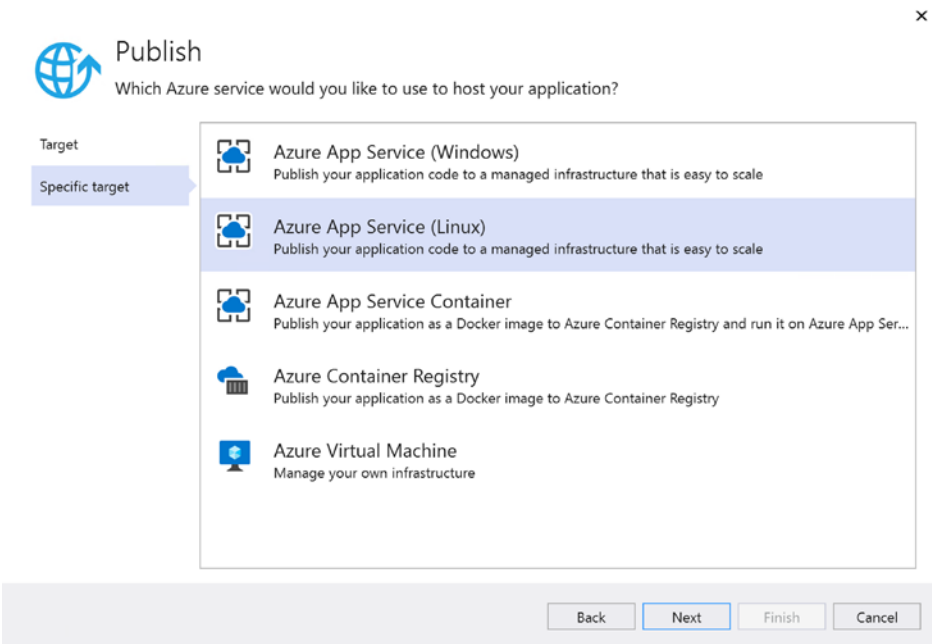
1. From within **Visual Studio Solution Explorer**, select **SimplCommerce.WebHost**, right-click it, and select **Publish....**



2. This starts the web application Publish wizard. In the **Where are you publishing today?** step, select **Azure**.



3. **Click Next.**



4. In the **Which Azure service would you like to use to host your application?** step, **select Azure App Service (Linux)**. This works because our application is based on .NET Core, which runs on both Windows and Linux.
5. This brings you to the **Select existing or create a new Azure App Service step** window.

Publish
Select existing or create a new Azure App Service

Microsoft account
aaddemouser@outlook.com

Target: Subscription
Azure Pass - Sponsorship

Specific target: View

App Service: Resource group

Search:

(No resources found)

[+ Create a new Azure App Service...](#) [Refresh](#)

Back Next Finish Cancel


6. **Click “+ Create a new Azure App Service...”**, which opens yet another popup window, in which you need to enter several details, related to the Azure Web App name, Azure region, and App Service plan.

Complete/validate the different parameters:

- **Name: Update the dynamically generated name with a more accurate one (e.g., [suffix]simplcommercefromvs2019).**

- **Subscription:** Select your Azure subscription.
- **Resource group:** Create a new resource group/[SUFFIX] **SimplwebAppRG.**
- **Hosting Plan:** Create a new Hosting Plan, specifying S1 and a close-by region.

×



Hosting Plan

Create new

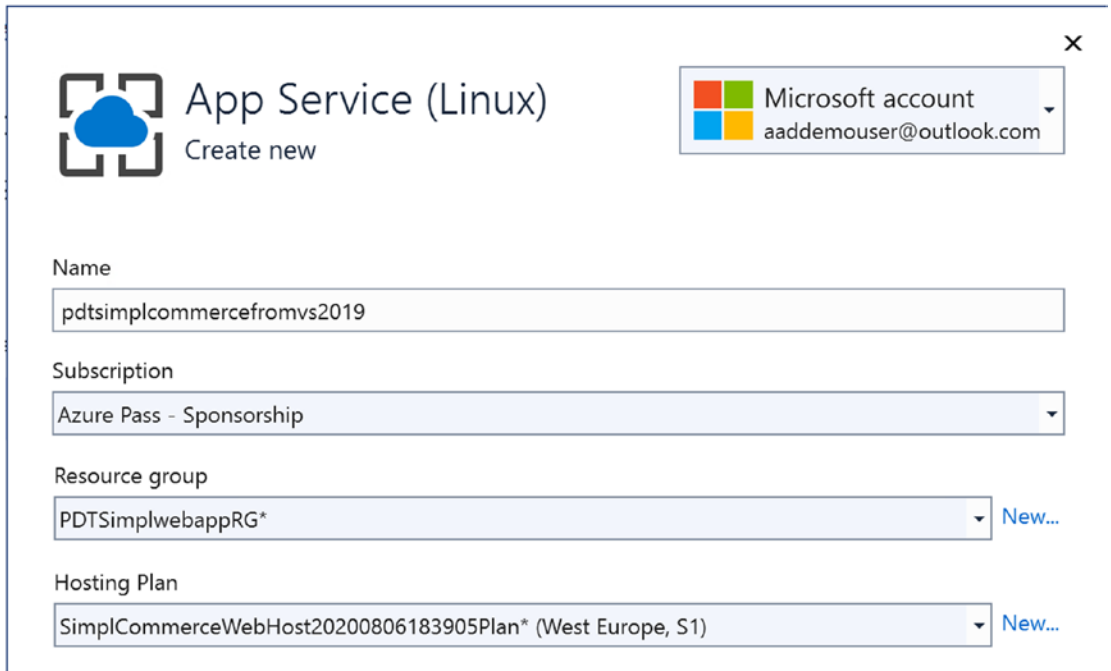
Hosting Plan


Location

West Europe▼

Size

S1 (1 core, 1.75 GB RAM)▼



 App Service (Linux)
Create new

Microsoft account
aaddemouser@outlook.com

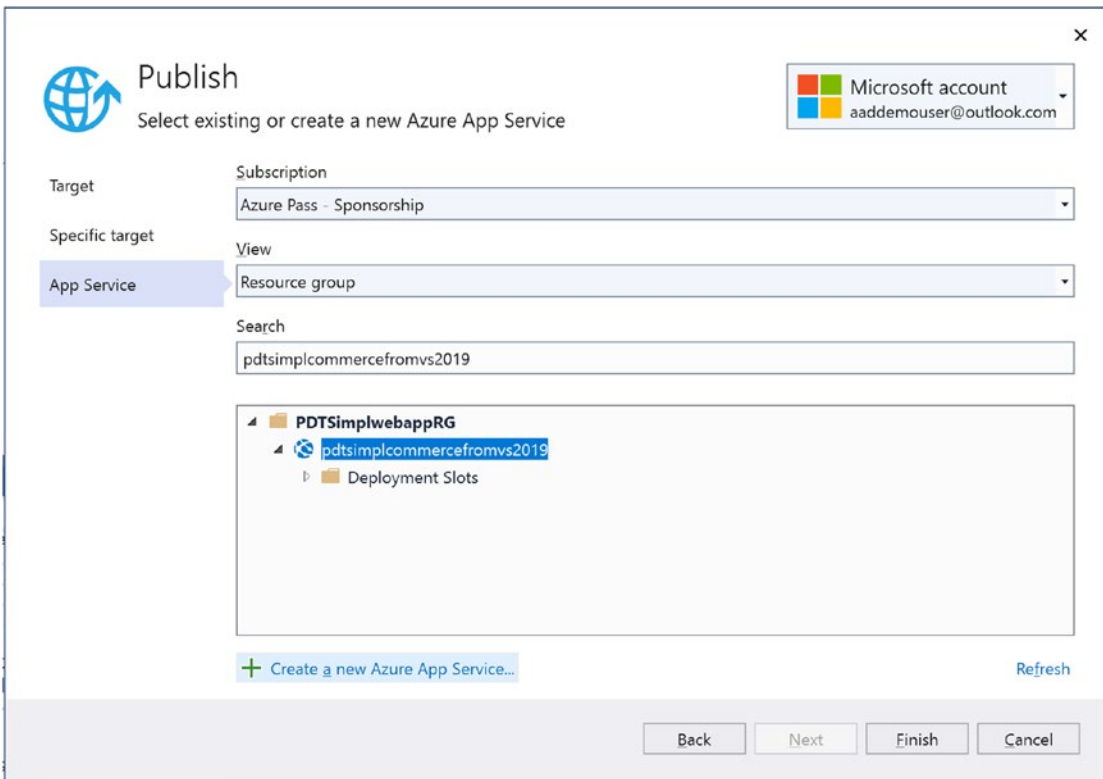
Name
pdtSimplcommercefromvs2019

Subscription
Azure Pass - Sponsorship

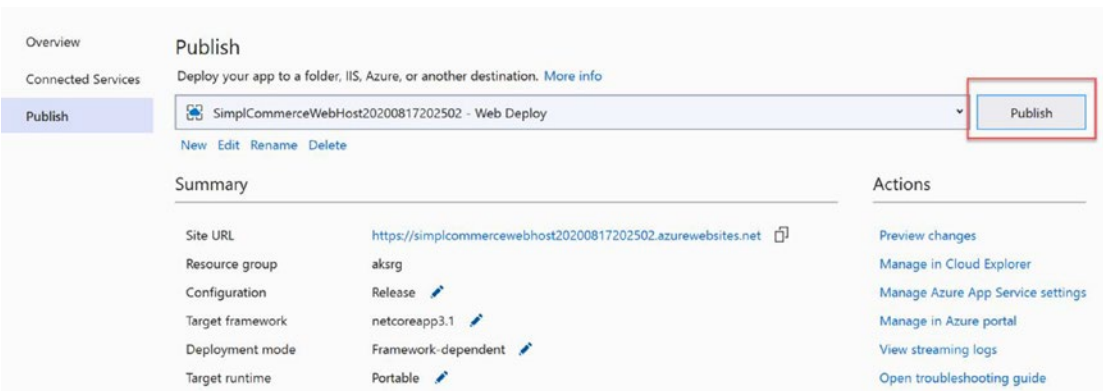
Resource group
PDTsimplwebappRG* [New...](#)

Hosting Plan
SimplCommerceWebHost20200806183905Plan* (West Europe, S1) [New...](#)

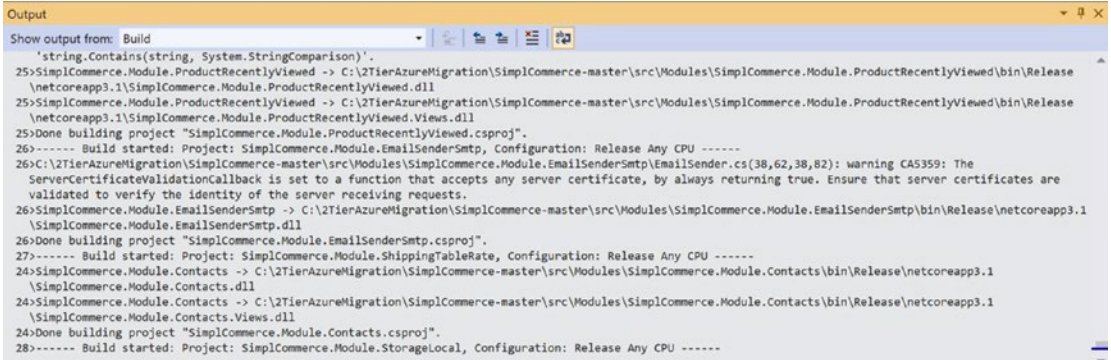
7. Confirm by clicking **Create**. The necessary Azure resources are getting created, which should take only about a minute. After that, the newly created app service will be listed as selected target for the web app.



8. **Confirm the deployment by clicking “Finish.”** This returns you to the Visual Studio 2019 Publish window, highlighting your web app as target for Web Deploy.



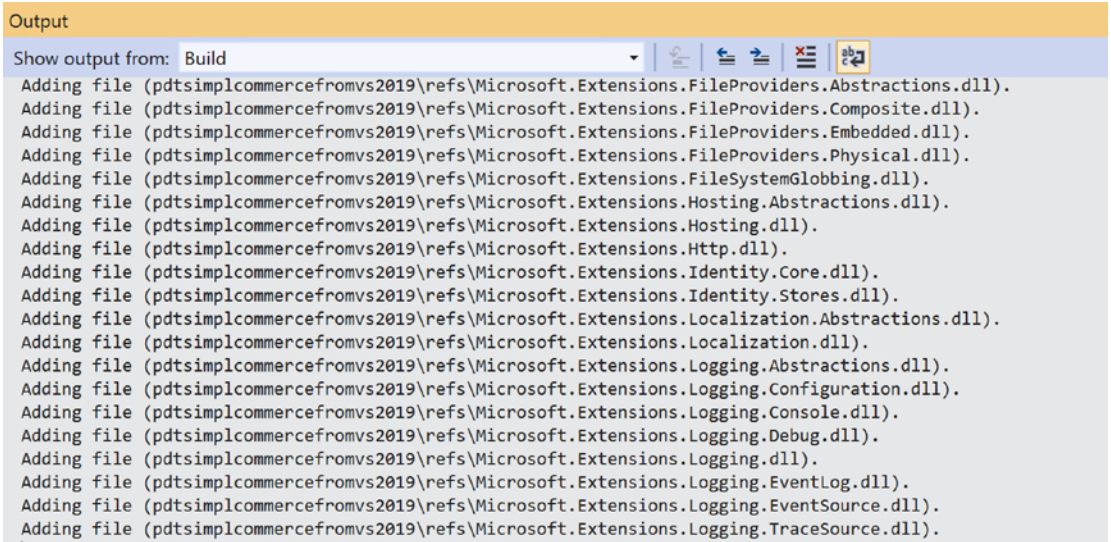
9. Click **“Publish”** to get the source files pushed to Azure Web Apps, and you can follow this process from the Visual Studio Output window.



```

Output
Show output from: Build
'string.Contains(string, System.StringComparison)'.
25>SimplCommerce.Module.ProductRecentlyViewed -> C:\2TierAzureMigration\SimplCommerce-master\src\Modules\SimplCommerce.Module.ProductRecentlyViewed\bin\Release\netcoreapp3.1\SimplCommerce.Module.ProductRecentlyViewed.dll
25>SimplCommerce.Module.ProductRecentlyViewed -> C:\2TierAzureMigration\SimplCommerce-master\src\Modules\SimplCommerce.Module.ProductRecentlyViewed\bin\Release\netcoreapp3.1\SimplCommerce.Module.ProductRecentlyViewed.Views.dll
25>Done building project "SimplCommerce.Module.ProductRecentlyViewed.csproj".
26>----- Build started: Project: SimplCommerce.Module.EmailSenderSmtplib, Configuration: Release Any CPU -----
26>C:\2TierAzureMigration\SimplCommerce-master\src\Modules\SimplCommerce.Module.EmailSenderSmtplib\EmailSender.cs(38,62,38,82): warning CA5359: The ServerCertificateValidationCallback is set to a function that accepts any server certificate, by always returning true. Ensure that server certificates are validated to verify the identity of the server receiving requests.
26>SimplCommerce.Module.EmailSenderSmtplib -> C:\2TierAzureMigration\SimplCommerce-master\src\Modules\SimplCommerce.Module.EmailSenderSmtplib\bin\Release\netcoreapp3.1\SimplCommerce.Module.EmailSenderSmtplib.dll
26>Done building project "SimplCommerce.Module.EmailSenderSmtplib.csproj".
27>----- Build started: Project: SimplCommerce.Module.ShippingTableRate, Configuration: Release Any CPU -----
24>SimplCommerce.Module.Contacts -> C:\2TierAzureMigration\SimplCommerce-master\src\Modules\SimplCommerce.Module.Contacts\bin\Release\netcoreapp3.1\SimplCommerce.Module.Contacts.dll
24>SimplCommerce.Module.Contacts -> C:\2TierAzureMigration\SimplCommerce-master\src\Modules\SimplCommerce.Module.Contacts\bin\Release\netcoreapp3.1\SimplCommerce.Module.Contacts.Views.dll
24>Done building project "SimplCommerce.Module.Contacts.csproj".
28>----- Build started: Project: SimplCommerce.Module.StorageLocal, Configuration: Release Any CPU -----

```



```

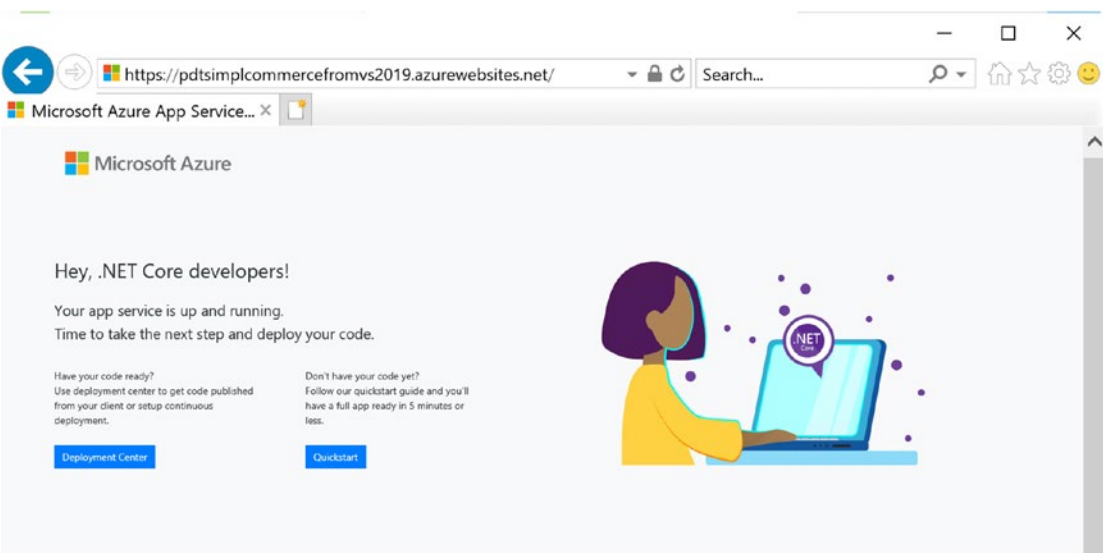
Output
Show output from: Build
Adding file (pdtsimplcommercefromvs2019\refs\Microsoft.Extensions.FileProviders.Abstractions.dll).
Adding file (pdtsimplcommercefromvs2019\refs\Microsoft.Extensions.FileProviders.Composite.dll).
Adding file (pdtsimplcommercefromvs2019\refs\Microsoft.Extensions.FileProviders.Embedded.dll).
Adding file (pdtsimplcommercefromvs2019\refs\Microsoft.Extensions.FileProviders.Physical.dll).
Adding file (pdtsimplcommercefromvs2019\refs\Microsoft.Extensions.FileSystemGlobbing.dll).
Adding file (pdtsimplcommercefromvs2019\refs\Microsoft.Extensions.Hosting.Abstractions.dll).
Adding file (pdtsimplcommercefromvs2019\refs\Microsoft.Extensions.Hosting.dll).
Adding file (pdtsimplcommercefromvs2019\refs\Microsoft.Extensions.Http.dll).
Adding file (pdtsimplcommercefromvs2019\refs\Microsoft.Extensions.Identity.Core.dll).
Adding file (pdtsimplcommercefromvs2019\refs\Microsoft.Extensions.Identity.Stores.dll).
Adding file (pdtsimplcommercefromvs2019\refs\Microsoft.Extensions.Localization.Abstractions.dll).
Adding file (pdtsimplcommercefromvs2019\refs\Microsoft.Extensions.Localization.dll).
Adding file (pdtsimplcommercefromvs2019\refs\Microsoft.Extensions.Logging.Abstractions.dll).
Adding file (pdtsimplcommercefromvs2019\refs\Microsoft.Extensions.Logging.Configuration.dll).
Adding file (pdtsimplcommercefromvs2019\refs\Microsoft.Extensions.Logging.Console.dll).
Adding file (pdtsimplcommercefromvs2019\refs\Microsoft.Extensions.Logging.Debug.dll).
Adding file (pdtsimplcommercefromvs2019\refs\Microsoft.Extensions.Logging.dll).
Adding file (pdtsimplcommercefromvs2019\refs\Microsoft.Extensions.Logging.EventLog.dll).
Adding file (pdtsimplcommercefromvs2019\refs\Microsoft.Extensions.Logging.EventSource.dll).
Adding file (pdtsimplcommercefromvs2019\refs\Microsoft.Extensions.Logging.TraceSource.dll).

```

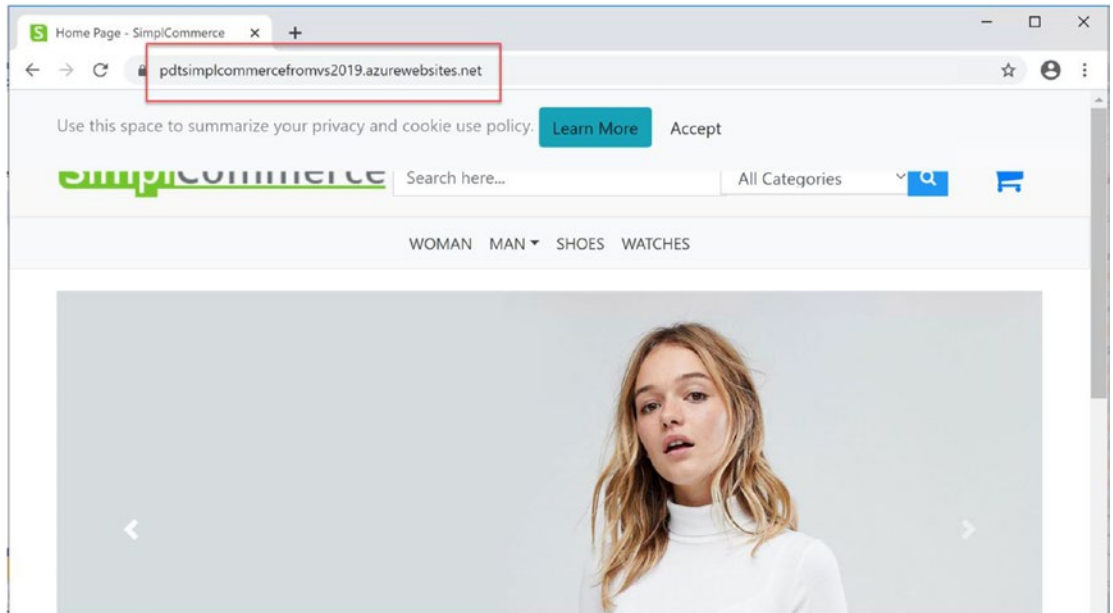
```
Output
Show output from: Build
Adding file (pdtssimplcommercefromvs2019\wwwroot\_content\SimplCommerce.Module.Vendors\admin\vendors\vendor-list.js).
Adding file (pdtssimplcommercefromvs2019\wwwroot\_content\SimplCommerce.Module.Vendors\admin\vendors\vendor-service.js).
Adding file (pdtssimplcommercefromvs2019\wwwroot\_content\SimplCommerce.Module.Vendors\admin\vendors\module.js).
Adding file (pdtssimplcommercefromvs2019\wwwroot\_content\SimplCommerce.Module.WishList\private-list.css).
Adding file (pdtssimplcommercefromvs2019\wwwroot\_content\SimplCommerce.Module.WishList\public-list.css).
Adding file (pdtssimplcommercefromvs2019\wwwroot\_content\SimplCommerce.Module.WishList\wishlist.js).
Adding file (pdtssimplcommercefromvs2019\zh-CN\Humanizer.resources.dll).
Adding file (pdtssimplcommercefromvs2019\zh-Hans\Humanizer.resources.dll).
Adding file (pdtssimplcommercefromvs2019\zh-Hans\Microsoft.CodeAnalysis.CSharp.resources.dll).
Adding file (pdtssimplcommercefromvs2019\zh-Hans\Microsoft.CodeAnalysis.resources.dll).
Adding file (pdtssimplcommercefromvs2019\zh-Hant\Humanizer.resources.dll).
Adding file (pdtssimplcommercefromvs2019\zh-Hant\Microsoft.CodeAnalysis.CSharp.resources.dll).
Adding file (pdtssimplcommercefromvs2019\zh-Hant\Microsoft.CodeAnalysis.resources.dll).
Publish Succeeded.
===== Build: 38 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
===== Publish: 1 succeeded, 0 failed, 0 skipped =====
Waiting for Web App to be ready...
Restarting the Web App...
Successfully restarted Web App.
Web App is ready.
```

- 10. **Wait** for the process to complete successfully. At the end, Visual Studio will open your default browser, where you can validate the web app is running successfully.

Note I freaked out at first, since my web app was not loading correctly in the browser – at least not in Internet Explorer 11 (which seemed the default on the JumpVM still). The following default web app page was shown:



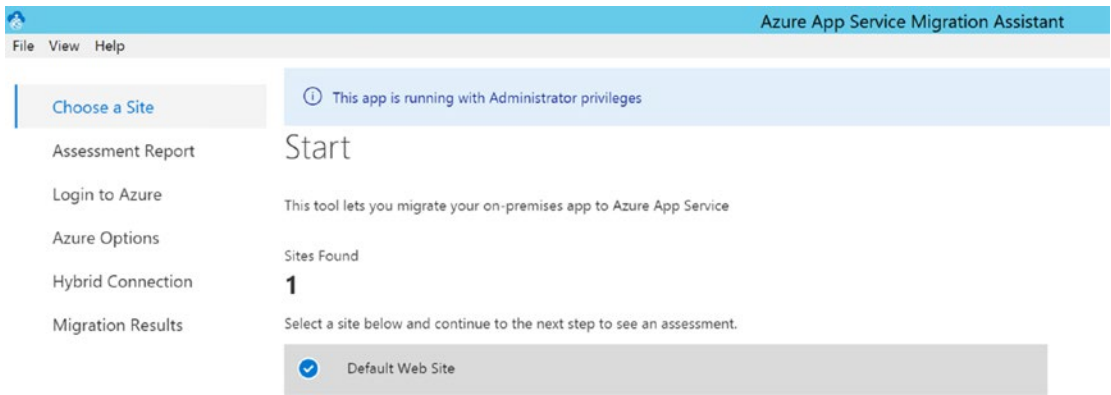
This could also be an issue with the code compilation itself (although we validated that in Visual Studio prior to publishing to Azure); however, when using **Microsoft Edge** or **Chrome** (which both are preinstalled on the JumpVM), the site was running as expected:



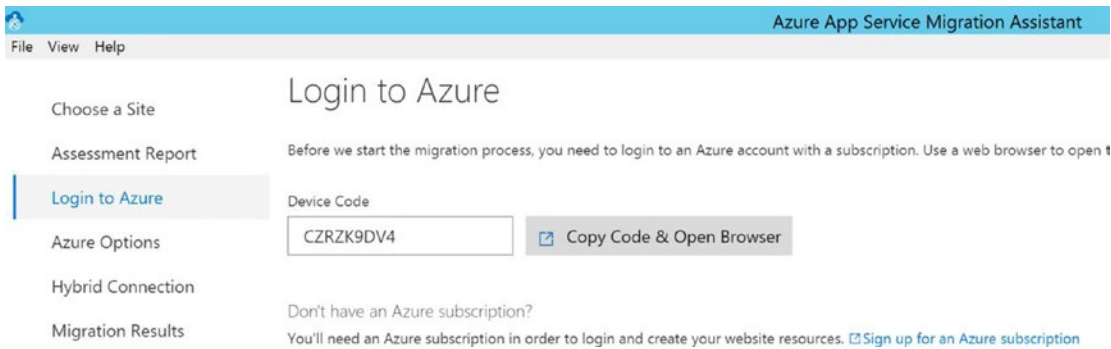
This completes the task, in which you published the webshop source code to Azure Web Apps using the Visual Studio Publish wizard integration.

Task 3: Migrating a web application from Azure App Service Migration Assistant

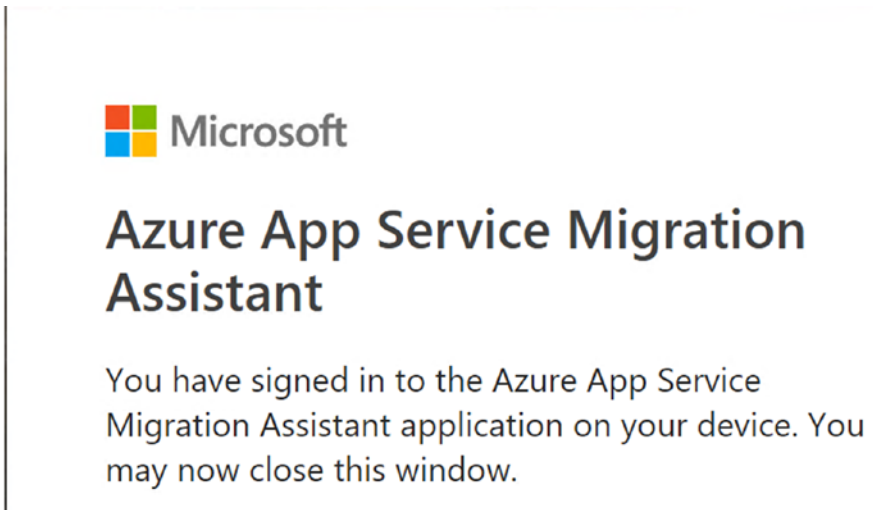
1. **Start an RDP session** to the **WebVM** you have running in Azure (labadmin and L@BadminPa55w.rd).
2. **From the desktop**, launch **Azure App Service Migration Assistant**. Since we used this tool for performing the web application assessment in a previous lab, it will remember some of those parameters.



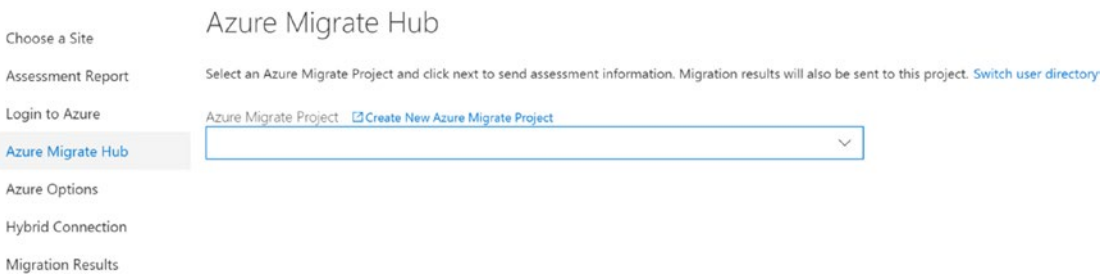
3. Select the detected **Default Web Site**, and click **Next**.
4. The tool will perform another assessment first; when complete, click **Next**. This is where you will launch and execute the actual web app migration, starting with authenticating to Azure.



5. Click **“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 Azure App Service Migration Assistant, you can immediately continue the migration process. The next step is **Azure Migrate Hub**, allowing you to add this project to **Azure Migrate**.



7. You can **skip** this step for now, which brings you to the **Azure Options** window. Here, you need to provide the necessary parameters to get the web app deployed and configured:
- **Resource Group: Create a new resource group** (the Migration Assistant will publish this application to a Windows-based web app, which cannot be mixed with the Linux-based web app service plan in the same resource group).
 - **Destination Site Name: Provide a unique name for the web app.**
 - **Region: Your Azure region of choice.**

Azure Options

We will create the required Azure resources for you to create and migrate your contents to a new app. Before we can do that we need some information.

Subscription *

Azure Pass - Sponsorship (a4cc356c-74b3-49cf-9390-ca86e7559d7a) ▾

Resource Group *

Create new Use existing

PDTsimplmigwebappRG

Destination Site Name *

pdtssimplwebappfrommig .azurewebsites.net

App service plan

Create new Use existing

Region *

West Europe ▾

A single Premium P1v2 instance will be created in the selected region. [Learn More about pricing tier](#)

Databases

Choose how to handle database connections [Learn More](#)

Skip database setup Set up hybrid connection to enable database connection

Note The Migration Assistant automatically allocates a “Premium P1” App Service plan; if needed, this can be changed from the web app settings once the migration is complete.

8. In the **database setup**, choose “Skip database setup.”

App service plan

A single Premium P1v2 instance will be created in the selected region. [Learn More about pricing tier](#)

Region *

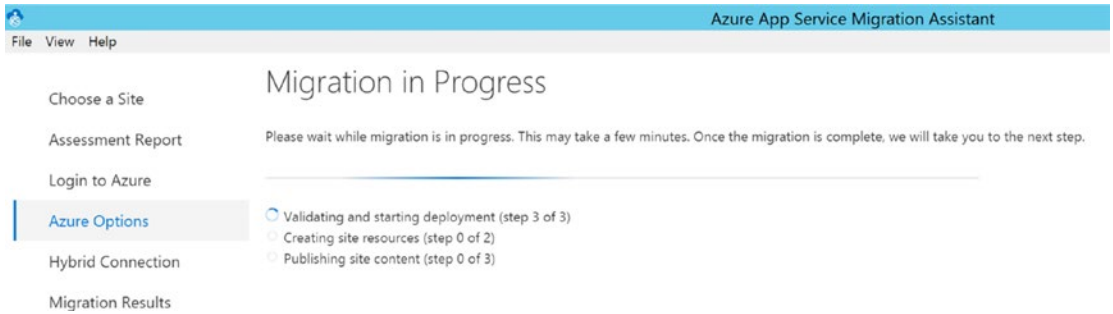
Central US ▾

Databases

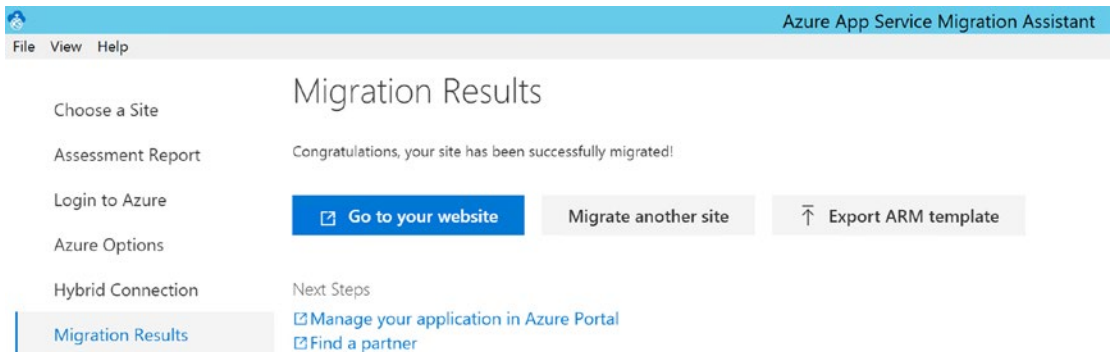
Choose how to handle database connections [Learn More](#)

Skip database setup Set up hybrid connection to enable database connection

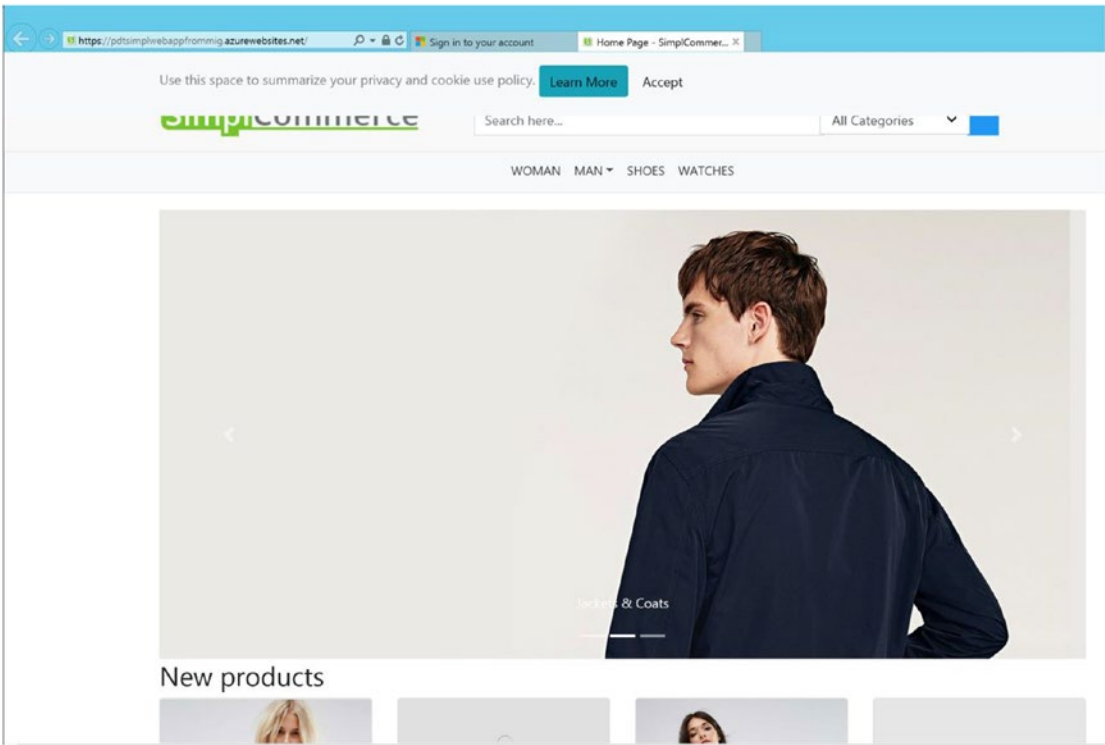
9. Confirm the settings by clicking the **Migrate** button.



10. 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.



- 11. Click **“Go to your website,”** which will open the newly deployed web app in the default browser.



- 12. 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.

CHAPTER 7

Lab 5: Deploying Docker and Running Azure Container Workloads

What You Will Learn

In this lab, we focus on deploying (a trial) edition of Docker Enterprise on Windows Server 2019, but using the LinuxKit rather than using Windows containers (just because we can and it is cool to showcase the mixed environment setup in my opinion). Starting with installing the Docker Enterprise Edition for Windows Server, you learn the basics of Docker commands using the Docker command-line interface. Next, you learn how to “Dockerize” the dotnetcore code that has been used in the former lab, using Visual Studio Code with Docker extensions.

In the next task, you learn about Azure Container Registry (ACR) and how to publish your new Docker container in there, as well as using this as a source for Azure Container Instance (ACI) and running your web application. We will also touch on deploying and running Azure Web App for Containers, allowing for advanced operations on containerized workloads, compared to Azure Container Instance.

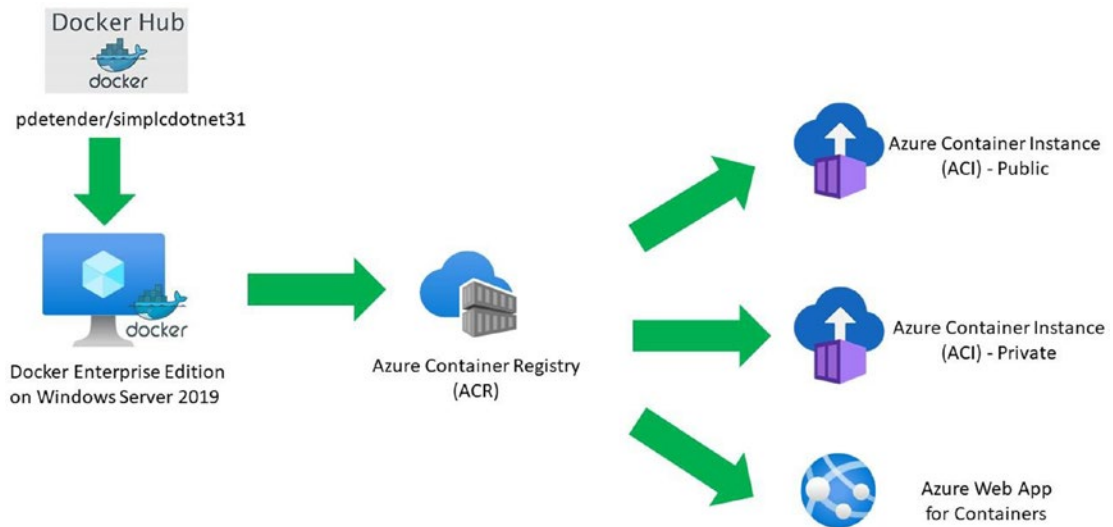
Time Estimate

This lab is estimated to take 90 min.

Prerequisites

There are no dependencies on previous lab exercises to start and complete this specific lab, outside of going through Chapter 2 to grab the necessary source files.

Scenario Diagram

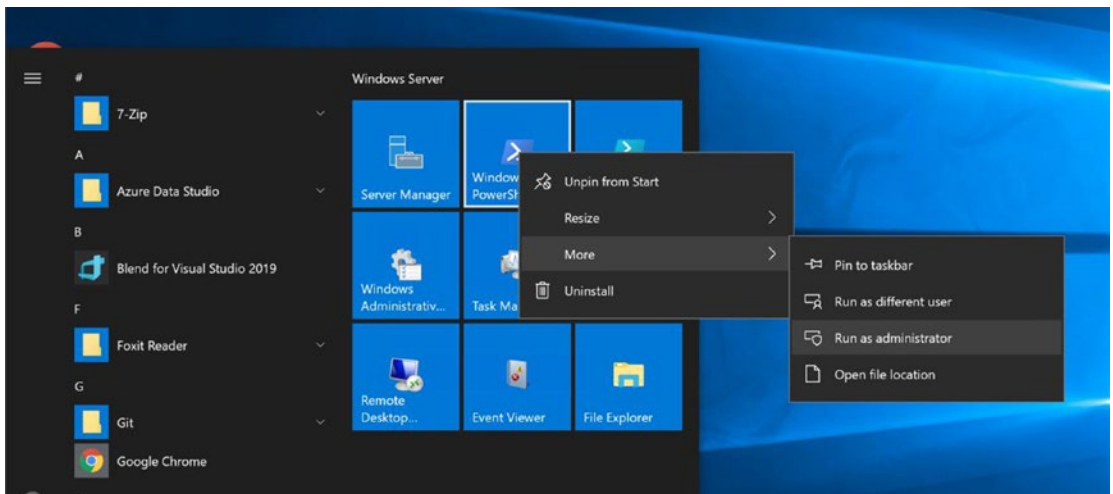


Tasks

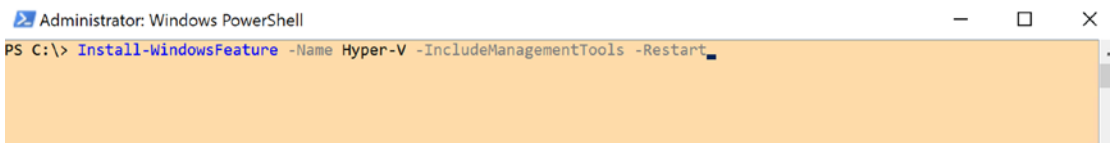
- Task 1:** Installing Docker Enterprise Edition on Windows Server 2019
- Task 2:** Validating and running basic Docker commands and containers
- Task 3:** Integrating Docker extension in Visual Studio Code
- Task 4:** Deploying and operating Azure Container Registry
- Task 5:** Deploying and running Azure Container Instance
- Task 6:** Deploying and operating Azure Web App for Containers

Task 1: Installing Docker Enterprise Edition (trial) for Windows Server 2019 on the lab jumpVM

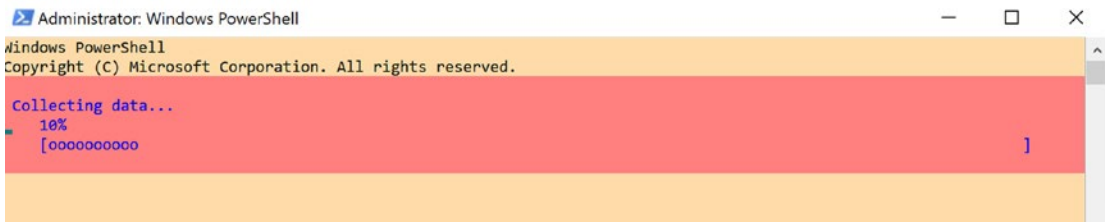
1. If not logged on anymore to the lab jumpVM, open an RDP session to this virtual machine, using labadmin and L@BadminPa55w.rd as credentials.
2. From the **Start menu**, launch **PowerShell** with **Run as administrator permissions**.



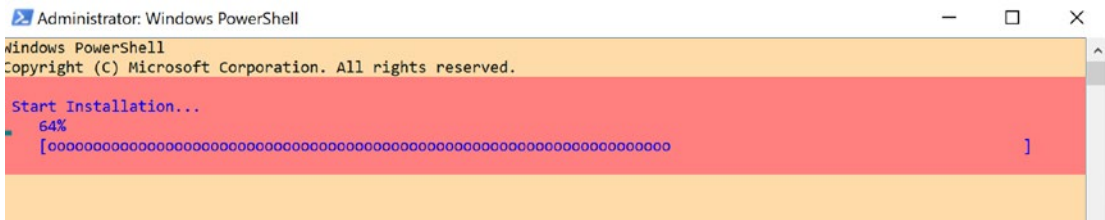
3. Run the following cmdlet:
Install-WindowsFeature -Name Hyper-V -IncludeManagement Tools -Restart



4. **Status information** will be shown.



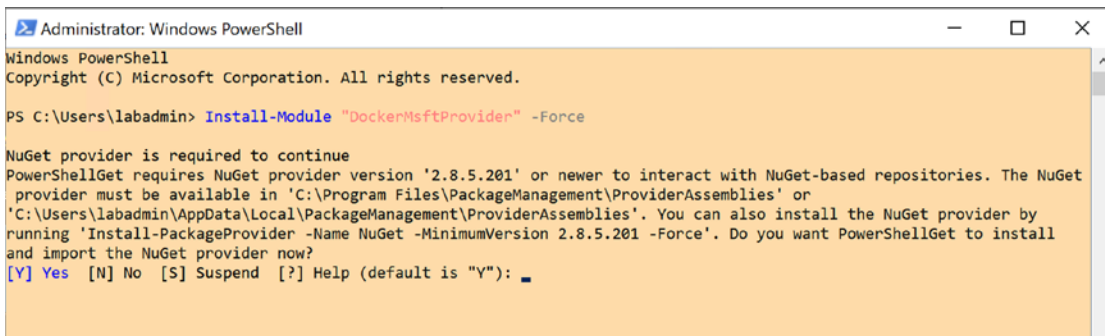
5. After which the **installation** starts.



6. Once the installation is complete, your **machine will restart (required!);** wait for it to reboot, and **log on using RDP again, reopening the PowerShell console (with Run as administrator permissions).**

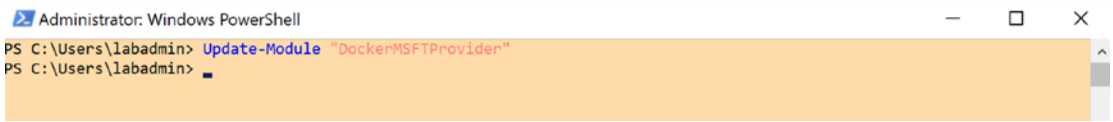
7. Next, we will install the **Docker Enterprise Edition** using the **PowerShell module “DockerMSFTProvider,”** using the following **cmdlet:**

Install-module “DockerMSFTProvider” -Force



8. This is followed by an update-cmdlet to make sure we have the latest bits:

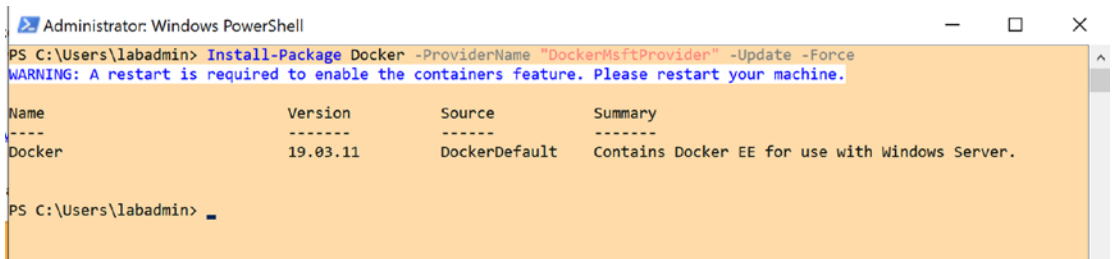
update-module "DockerMSFTProvider"



```
Administrator: Windows PowerShell
PS C:\Users\labadmin> Update-Module "DockerMSFTProvider"
PS C:\Users\labadmin>
```

9. Next, we will trigger the actual **Docker Enterprise** package installation, executing the following cmdlet:

Install-package Docker -ProviderName "DockerMSFTProvider" -Update -Force



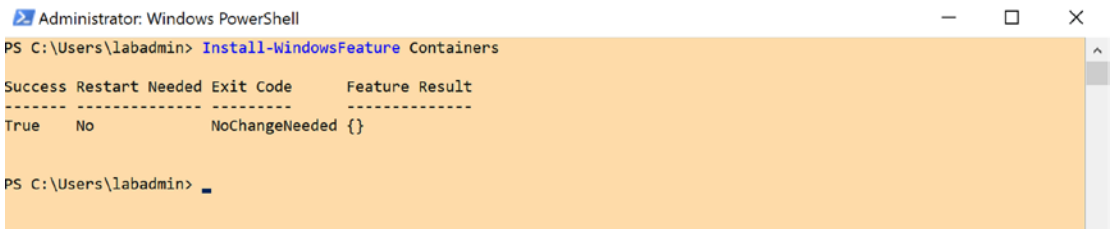
```
Administrator: Windows PowerShell
PS C:\Users\labadmin> Install-Package Docker -ProviderName "DockerMsftProvider" -Update -Force
WARNING: A restart is required to enable the containers feature. Please restart your machine.

Name                Version      Source          Summary
-----                -
Docker              19.03.11    DockerDefault   Contains Docker EE for use with Windows Server.

PS C:\Users\labadmin>
```

10. Once the installation of the package is complete, we also need to make sure we install the Windows Feature **Containers**, informing the host it will run as a container host, by **running the following cmdlet**:

Install-WindowsFeature Containers



```
Administrator: Windows PowerShell
PS C:\Users\labadmin> Install-WindowsFeature Containers

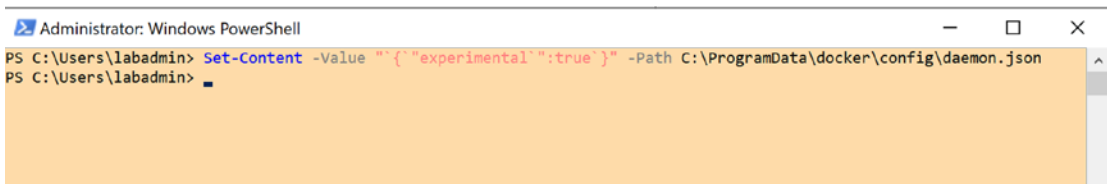
Success Restart Needed Exit Code      Feature Result
-----
True      No           NoChangeNeeded {}

PS C:\Users\labadmin>
```

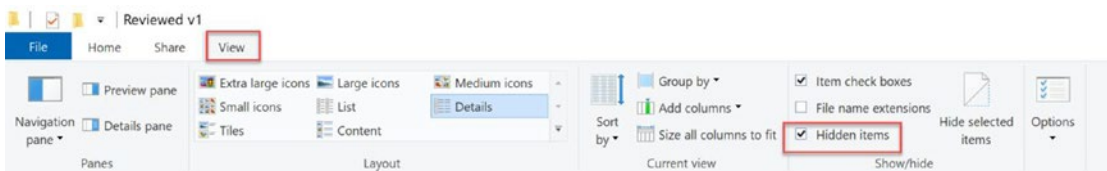

- 11. This is about it from a Windows Server and module perspective. However, we need to go through a few more steps to “enable” the **Linux/Linux Containers on Windows – LCOW**, starting with **creating a config JSON file for the experimental aspect of LCOW**.

Run the following cmdlet (this is on one line, but wrapped because of the layout):

```
Set-Content -Value "`{"experimental":true`}" -Path C:\ProgramData\docker\config\daemon.json
```



Note If you can’t complete this step successfully, verify if you have “Show Hidden items” enabled in your Windows Explorer.



- 12. **This is followed by restarting the Docker service, using restart-service Docker.**
- 13. **Confirm** the Docker engine is up and running, by **executing Docker version**
- 14. As well, **execute Docker info**

```

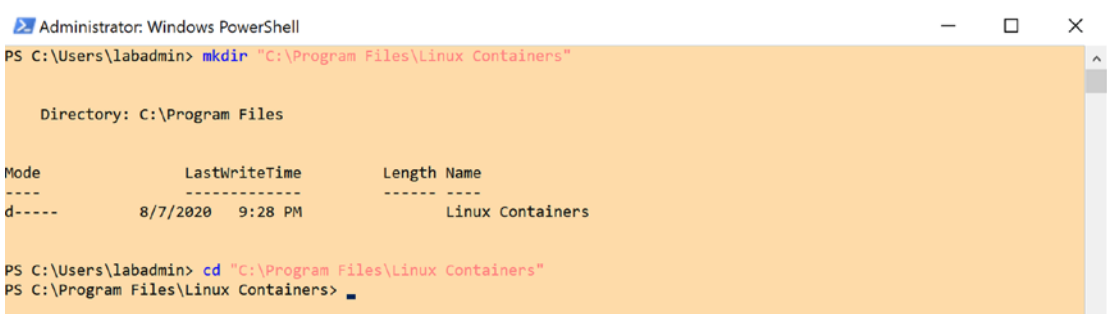
Administrator: Windows PowerShell
PS C:\Users\labadmin> Restart-Service docker
PS C:\Users\labadmin> docker version
Client: Docker Engine - Enterprise
Version: 19.03.11
API version: 1.40
Go version: go1.13.11
Git commit: 0da829ac52
Built: 06/26/2020 17:20:46
OS/Arch: windows/amd64
Experimental: false

Server: Docker Engine - Enterprise (Unlicensed - not for production workloads)
Engine:
Version: 19.03.11
API version: 1.40 (minimum version 1.24)
Go version: go1.13.11
Git commit: 0da829ac52
Built: 06/26/2020 17:19:32
OS/Arch: windows/amd64
Experimental: true
PS C:\Users\labadmin> docker info
Client:
Debug Mode: false
Plugins:
cluster: Manage Docker Enterprise clusters (Mirantis Inc., v1.4.0)

Server:
Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
Images: 0
Server Version: 19.03.11
Storage Driver: lcow (linux) windowsfilter (windows)
  LCOW:
  Windows:
Logging Driver: json-file
Plugins:
  Volume: local
  Network: ics internal l2bridge l2tunnel nat null overlay private transparent
  Log: awslogs etwlogs fluentd gcplogs gelf json-file local logentries splunk syslog
Swarm: inactive
Default Isolation: process
Kernel Version: 10.0 17763 (17763.1.amd64fre.rs5_release.180914-1434)
Operating System: Windows Server 2019 Datacenter Version 1809 (OS Build 17763.1339)
OSType: windows
Architecture: x86_64
CPUs: 2
Total Memory: 8GiB
Name: jumpvm

```

- The **Linux Containers on Windows** expects a specific folder to run in, so we need to create this folder first; easiest is using **mkdir <path>**:
mkdir "C:\Program Files\Linux Containers"



```
Administrator: Windows PowerShell
PS C:\Users\labadmin> mkdir "C:\Program Files\Linux Containers"

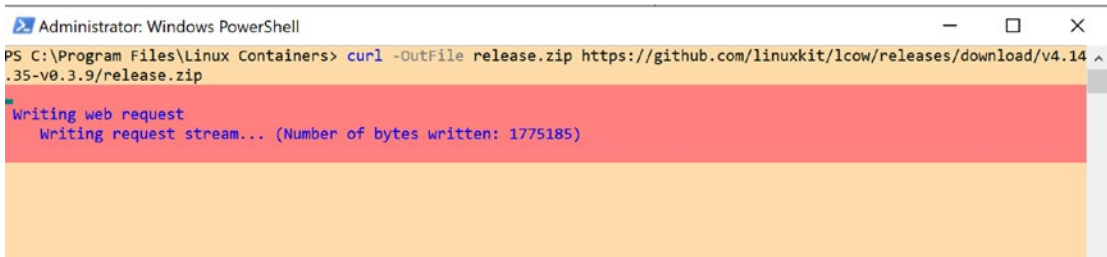
Directory: C:\Program Files

Mode                LastWriteTime         Length Name
----                -
d-----            8/7/2020   9:28 PM             Linux Containers

PS C:\Users\labadmin> cd "C:\Program Files\Linux Containers"
PS C:\Program Files\Linux Containers>
```

16. **This is followed** by downloading the “release” version of the kernel, by **launching the following cmdlet:**

```
curl -OutFile release.zip https://github.com/linuxkit/lcow/releases/download/v4.14.35-v0.3.9/release.zip
```



```
Administrator: Windows PowerShell
PS C:\Program Files\Linux Containers> curl -OutFile release.zip https://github.com/linuxkit/lcow/releases/download/v4.14.35-v0.3.9/release.zip
Writing web request
Writing request stream... (Number of bytes written: 1775185)
```

17. **Wait** for the download to complete; after which, we need to expand the archive file, **running the following cmdlet:**

```
Expand-Archive -DestinationPath . .\release.zip
```

```

Administrator: Windows PowerShell
PS C:\Program Files\Linux Containers> curl -OutFile release.zip https://github.com/linuxkit/lcow/releases/download/v4.14
.35-v0.3.9/release.zip
PS C:\Program Files\Linux Containers> dir

Directory: C:\Program Files\Linux Containers

Mode                LastWriteTime         Length Name
----                -
-a----            8/7/2020  9:29 PM       13840227 release.zip

PS C:\Program Files\Linux Containers> Expand-Archive -DestinationPath .\release.zip
PS C:\Program Files\Linux Containers> dir

Directory: C:\Program Files\Linux Containers

Mode                LastWriteTime         Length Name
----                -
-a----        11/15/2018  7:29 PM       6613996 initrd.img
-a----        11/15/2018  7:29 PM       7660304 kernel
-a----            8/7/2020  9:29 PM       13840227 release.zip
-a----        11/15/2018  7:29 PM           113 versions.txt

PS C:\Program Files\Linux Containers>

```

18. This **completes** the installation of the **LCOW** component; I'm pretty sure this process will become more straightforward in later builds of Windows Server 2019, although it is actually not too hard already.

This completes the first task, in which you installed Docker Enterprise Edition on Windows Server 2019, using the Linux Containers on Windows (LCOW) Kit. In the next task, you learn several Docker commands for managing and running container workloads.

Task 2: Validating and running basic Docker commands and containers

1. Let's try and **run a test Linux container, by executing the following command:**

docker run -it ubuntu

```

Administrator: Windows PowerShell
PS C:\Program Files\Linux Containers> docker run -it ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
3ff22d22a855: Pull complete
e7cb79d19722: Pull complete
323d0d660b6a: Pull complete
b7f616834fd0: Pull complete
Digest: sha256:5d1d5407f353843ecf8b16524bc5565aa332e9e6a1297c73a92d3e754b8a636d
Status: Downloaded newer image for ubuntu:latest
    
```

2. Since we don't have the image on our local machine yet, it needs to be downloaded first; the Docker engine relies on the Docker Hub, a public (and private) repository of images to pull the image from.
3. Once the download is complete, **Docker will “start up” the Ubuntu image and run it. This is expressed by giving us access to the Ubuntu system prompt (root@<containerID>#).**

From here, we can perform some basic Linux commands, for example, “LS,” which means “list,” showing a list of folders.

```

root@29ed466b60a3: /
PS C:\Program Files\Linux Containers> docker run -it ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
3ff22d22a855: Pull complete
e7cb79d19722: Pull complete
323d0d660b6a: Pull complete
b7f616834fd0: Pull complete
Digest: sha256:5d1d5407f353843ecf8b16524bc5565aa332e9e6a1297c73a92d3e754b8a636d
Status: Downloaded newer image for ubuntu:latest
root@29ed466b60a3:/# ls
bin  dev  home  lib32  libx32  media  opt  root  sbin  sys  usr
boot  etc  lib  lib64  lost+found  mnt  proc  run  srv  tmp  var
root@29ed466b60a3:/#
    
```

4. Or running the command “TOP” will show the list of running system processes and their performance counters.

```

root@29ed466b60a3: /
top - 21:31:41 up 0 min, 0 users, load average: 0.00, 0.00, 0.00
Tasks: 2 total, 1 running, 1 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni,100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 962.5 total, 856.8 free, 74.5 used, 31.2 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used, 793.7 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
   1 root        20   0   4112   552    4  S   0.0   0.1   0:00.08  bash
  10 root        20   0   6108   528    4  R   0.0   0.1   0:00.01  top
    
```


- To close the performance view, **press Ctrl-C**, which brings you back to the system prompt. If you want to shut down the container (= leaving the runtime), **type “exit”**.

```
PS C:\Program Files\Linux Containers> docker run -it ubuntu
root@80ee02bd9e27:/# dir
bin dev home lib32 libx32 media opt root sbin sys usr
boot etc lib lib64 lost+found mnt proc run srv tmp var
root@80ee02bd9e27:/# exit
exit
time="2020-08-07T21:42:07Z" level=error msg="Error waiting for container: failed to shutdown container: container 80ee02bd9e27d5ecee5ac9ba708e571f244909001c3c2e4e3b57855312cb05ba encountered an error during hcsshim::System::waitBackground: failure in a Windows system call: The virtual machine or container with the specified identifier is not running. (0xc0370110): subsequent terminate failed container 80ee02bd9e27d5ecee5ac9ba708e571f244909001c3c2e4e3b57855312cb05ba encountered an error during hcsshim::System::waitBackground: failure in a Windows system call: The virtual machine or container with the specified identifier is not running. (0xc0370110)"
PS C:\Program Files\Linux Containers> █
```

Note I received an error message here on-screen, informing me about “failed to shut down container.” This is presently listed as a known issue on the GitHub pages of the LCOW, although it is more of a bug in the status reporting, as the running container actually got shut down correctly.

- Validate** the running state of a container can be done by using the following **Docker command**:

Docker ps



```
Administrator: Windows PowerShell
PS C:\Program Files\Linux Containers> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS          PORTS
NAMES
PS C:\Program Files\Linux Containers> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS          PORTS
NAMES
29ed466b60a3  ubuntu   "/bin/bash"             2 minutes ago  Exited (4294967295) 19 seconds ago
trusting_bose
PS C:\Program Files\Linux Containers> █
```

- This shows **no running containers**; however, if you add the **-a** parameter to this command, it shows us “**history**” information about containers that ran on this host.

If you want to test with a few more **Linux-based** containers (e.g., Java, NGINX, Python, etc.) and several others that are available from hub.docker.com, feel free to do so.


Remember we have our own **DotnetCore 3.1 sample container**, based on the webshop application we used in the previous labs. To speed up the lab, as well as keeping the focus on running workloads on Azure, I am storing an up-to-date copy of the containerized application in my Docker Hub as well; so why not continue with this one from here, as well as for all remaining container-oriented lab exercises?

8. The SimplCommerce webshop container image in hub.docker.com is **pdetender/simplcdotnet31**. So similar to the “docker run ubuntu” example earlier, you can execute this command:

```
docker run -it -p 5000:80 pdetender/simplcdotnet31
```

Here is some explanation for the parameters:

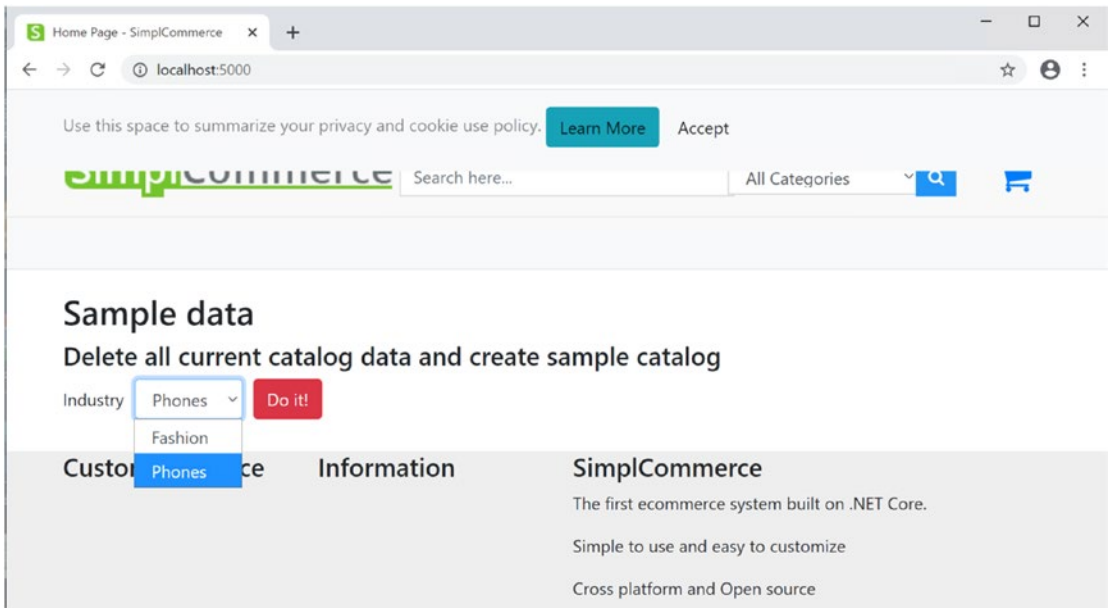
- **it:** Runs the container in interactive mode, which means it will show output (if any) in the console window.
- **p 5000:80:** This defines the container running on port 80, but mapping this to port 8000 in our local browser; this is handy when we have other applications or containers already running on port 80, as such avoiding any conflicts.

 Administrator: Windows PowerShell

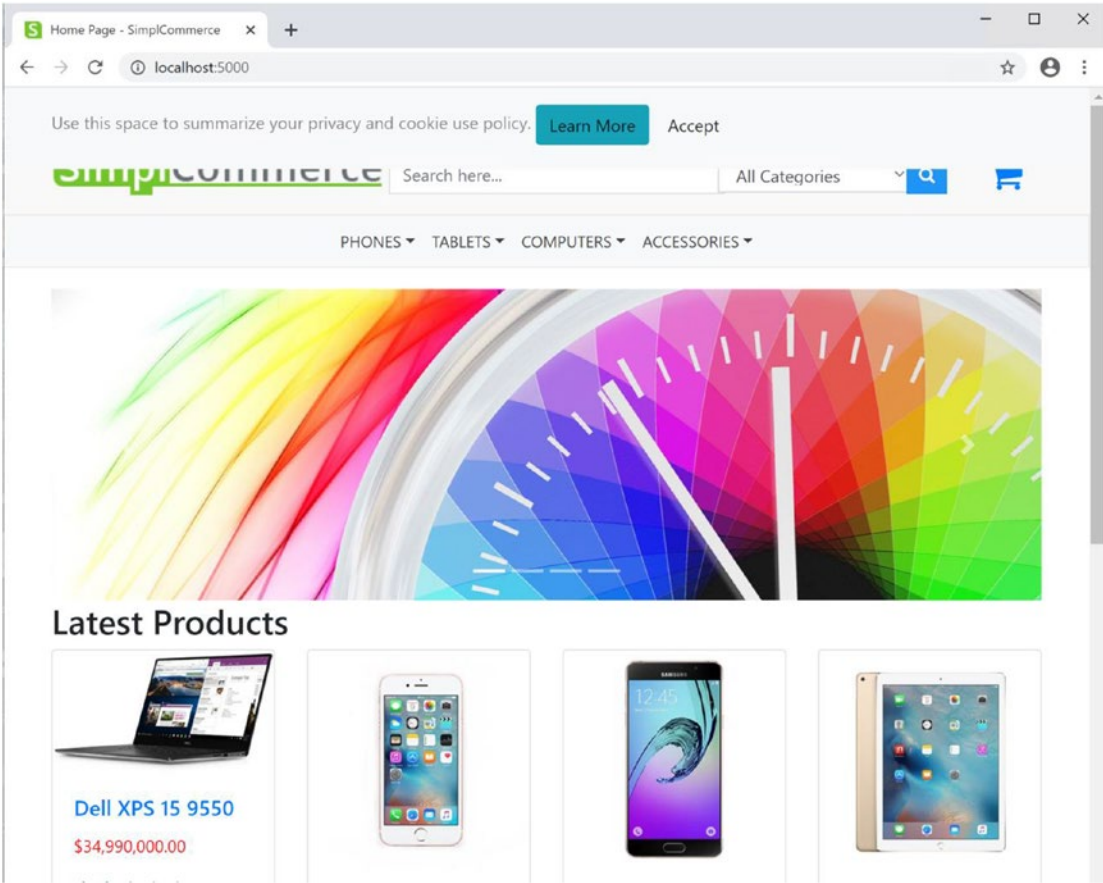
```
PS C:\2tierazuremigration\SimplCommerce31> docker run -it -p 5000:80 pdetender/simplcdotnet31
Unable to find image 'pdetender/simplcdotnet31:latest' locally
latest: Pulling from pdetender/simplcdotnet31
6ec8c9369e08: Already exists
fe8522826504: Already exists
658bf4619169: Already exists
0392978bbc2e: Already exists
33dd02257803: Already exists
f94d22bd253d: Already exists
00e296025eba: Already exists
16ec8d4b2b9e: Already exists
b34e8ab0f553: Already exists
754a119f24d9: Already exists
Digest: sha256:19816a782092ca2e6c27329c973490d90377adbbe3958ba1fc2cc7670923cc70
Status: Downloaded newer image for pdetender/simplcdotnet31:latest
```

9. Once the container is downloaded and running, **open** “localhost:5000” in your browser, which will show the “home page” of the SimplCommerce web application. Instead of expecting a full database like we used the Azure SQL earlier, this sample container image comes with its own built-in database engine. (If we want, we could update the container variables and actually point to an external database.)

Select “Phones” and click the “Do it!” button to confirm.



10. The webshop opens and shows **devices** available for buying.



11. **While this container instance is running, why not start another one?**
12. **Launch an additional instance of the PowerShell console (with Run as administrator permissions), and start a new container instance:**
docker run -it -p 4000:80 pdetender/simplcdotnet31
13. This time it is running on port 4000. **Since the image is already downloaded**, the container instance will kick off immediately.

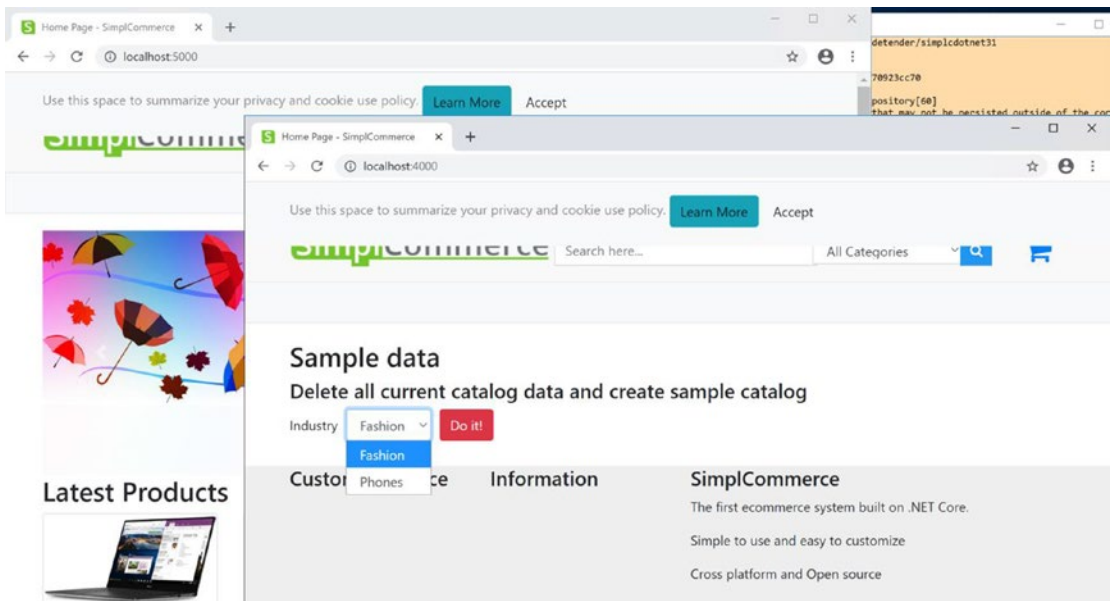
```

Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

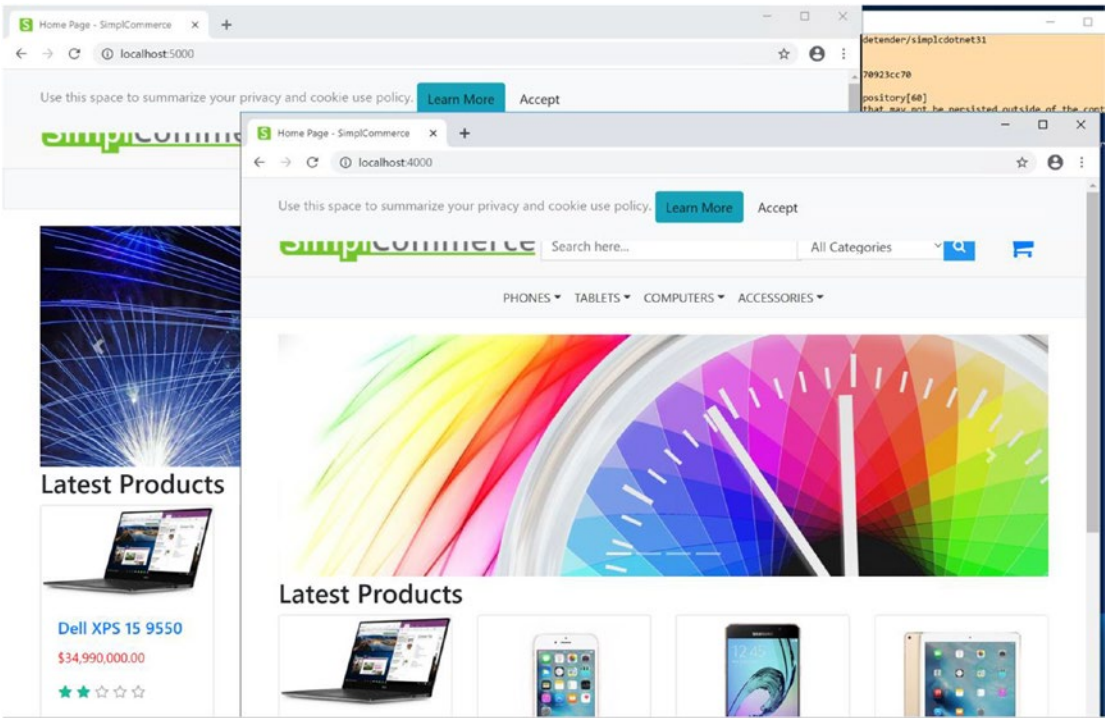
PS C:\Users\labadmin> docker run -it -p 4000:80 pdetender/simplcdotnet31
warn: Microsoft.AspNetCore.DataProtection.Repositories.FileSystemXmlRepository[60]
      Storing keys in a directory '/root/.aspnet/DataProtection-Keys' that may not be persisted outside of the container
      . Protected data will be unavailable when container is destroyed.
warn: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[35]
      No XML encryptor configured. Key {037bc59a-9bc6-43a8-901c-57d1ba0172b5} may be persisted to storage in unencrypted
      form.

```

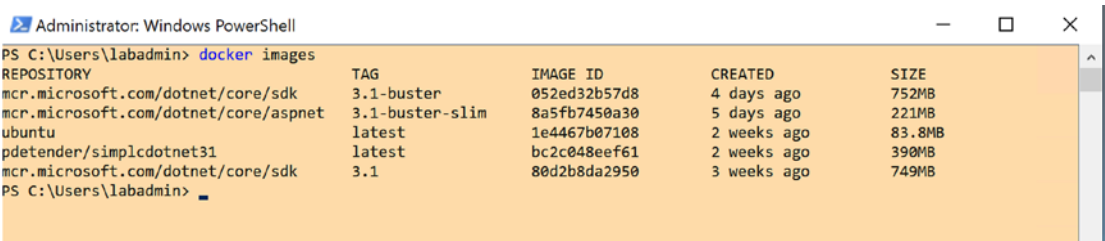
14. **Open your browser**, and connect to **localhost:4000**, which will show the webshop home page, confirming this is a new instance, since it is asking again to select the product database we want to use this time.



15. This loads the full application once selected.



- 16. **Switch back** to the **PowerShell window** (either of the open ones), and run **docker images**.



This shows a list of all current Docker images available on our machine. Note that besides the ubuntu and simplcdotnet31, I had a few additional ones, but you won't necessarily have these.

- 17. **Once more**, validate the “running” state of your container instance from a “Docker perspective,” but initiating the following command:

docker container ls

```

Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\labadmin> docker container ls
CONTAINER ID        IMAGE               COMMAND              CREATED        STATUS              PORTS
82d4479597f8      pdetender/simplcdotnet31  "dotnet SimplCommerc..."  43 seconds ago  Up 34 seconds      0.0.0.0:
4000->80/tcp      compassionate_pare
PS C:\Users\labadmin>

```

18. As you (should) still have the container instances running (port 4000 and port 5000), you **can take note of the (unique instance) container ID** and reuse this in other Docker commands, like **docker inspect 82d44** (where these are the first few characters of the container ID).

This provides a lot of additional details about our running container instance:

```

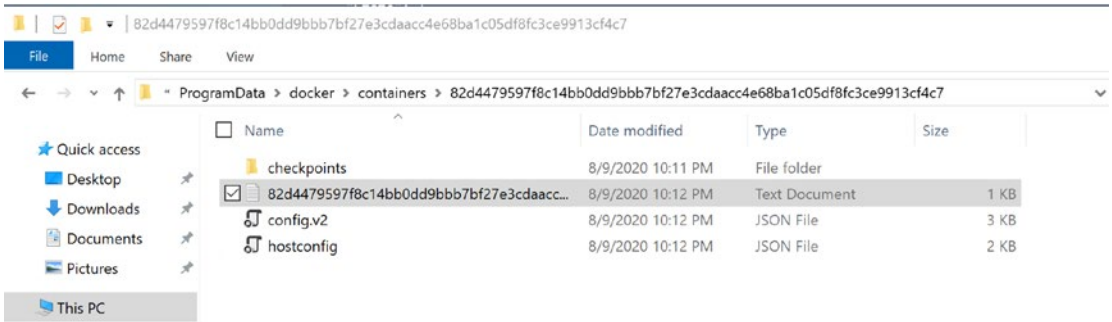
Administrator: Windows PowerShell
PS C:\Users\labadmin> docker inspect 82d44
[
  {
    "Id": "82d4479597f8c14bb0dd9bbb7bf27e3cdaacc4e68ba1c05df8fc3ce9913cf4c7",
    "Created": "2020-08-09T22:11:54.7835267Z",
    "Path": "dotnet",
    "Args": [
      "SimplCommerce.WebHost.dll"
    ],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 363,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2020-08-09T22:12:03.524782Z",
      "FinishedAt": "0001-01-01T00:00:00Z"
    },
    "Image": "sha256:bc2c048eef61c9c2c95783c19e7d378ef9208be52f1928e78b1cf4162b837fb3",
    "ResolvConfPath": "",
    "HostnamePath": "",
    "HostsPath": "",
    "LogPath": "C:\\ProgramData\\docker\\containers\\82d4479597f8c14bb0dd9bbb7bf27e3cdaacc4e68ba1c05df8fc3ce9913cf4c7\\82d4479597f8c14bb0dd9bbb7bf27e3cdaacc4e68ba1c05df8fc3ce9913cf4c7-json.log",
    "Name": "/compassionate_pare",
    "RestartCount": 0,
    "Driver": "lcow",
    "Platform": "linux",
    "MountLabel": "",
    "ProcessLabel": "",
    "AppArmorProfile": "",
    "ExecIDs": null,
    "HostConfig": {
      "Binds": null,

```

19. For example, consider “LogPath.”

```
    "RestartCount": 0,  
    "Driver": "lcow",  
    "Platform": "linux",  
    "MountLabel": "",  
    "ProcessLabel": "",  
    "AppArmorProfile": "",  
    "ExecIDs": null,  
    "HostConfig": {  
      "LogPath": "C:\\ProgramData\\docker\\containers\\82d4479597f8c14bb0dd9bbb7bf27e3cdaacc4e68ba1c05df8fc3ce9913cf47\\82d4479597f8c14bb0dd9bbb7bf27e3cdaacc4e68ba1c05df8fc3ce9913cf47-json.log",  
      "Name": "/compassionate_pare",  
      "RestartCount": 0,  
      "Driver": "lcow",  
      "Platform": "linux",  
      "MountLabel": "",  
      "ProcessLabel": "",  
      "AppArmorProfile": "",  
      "ExecIDs": null,  
      "HostConfig": {
```

20. This points to a log-JSON file, viewable from Windows Explorer, when browsing to the file location.



21. **Open the log-JSON file**, and notice the information stored in there is the same as what you saw earlier in the running container console (because you specified the “-it” parameter). Good to know this is not really required (although I personally prefer it, as it is a useful and easy mechanism to validate your container workload is running fine).

The screenshot shows two windows. The top window is 'Administrator: Windows PowerShell' with the following text:

```
PS C:\Users\labadmin> docker run -it -p 4000:80 pdetender/simplcdotnet31
warn: Microsoft.AspNetCore.DataProtection.Repositories.FileSystemXmlRepository[60]
      Storing keys in a directory '/root/.aspnet/DataProtection-Keys' that may not be persisted outside of the container
      . Protected data will be unavailable when container is destroyed.
warn: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[35]
      No XML encryptor configured. Key {b9cc1bc3-846c-40a8-aeb7-2ee3aa93759c} may be persisted to storage in unencrypted
      form.
```

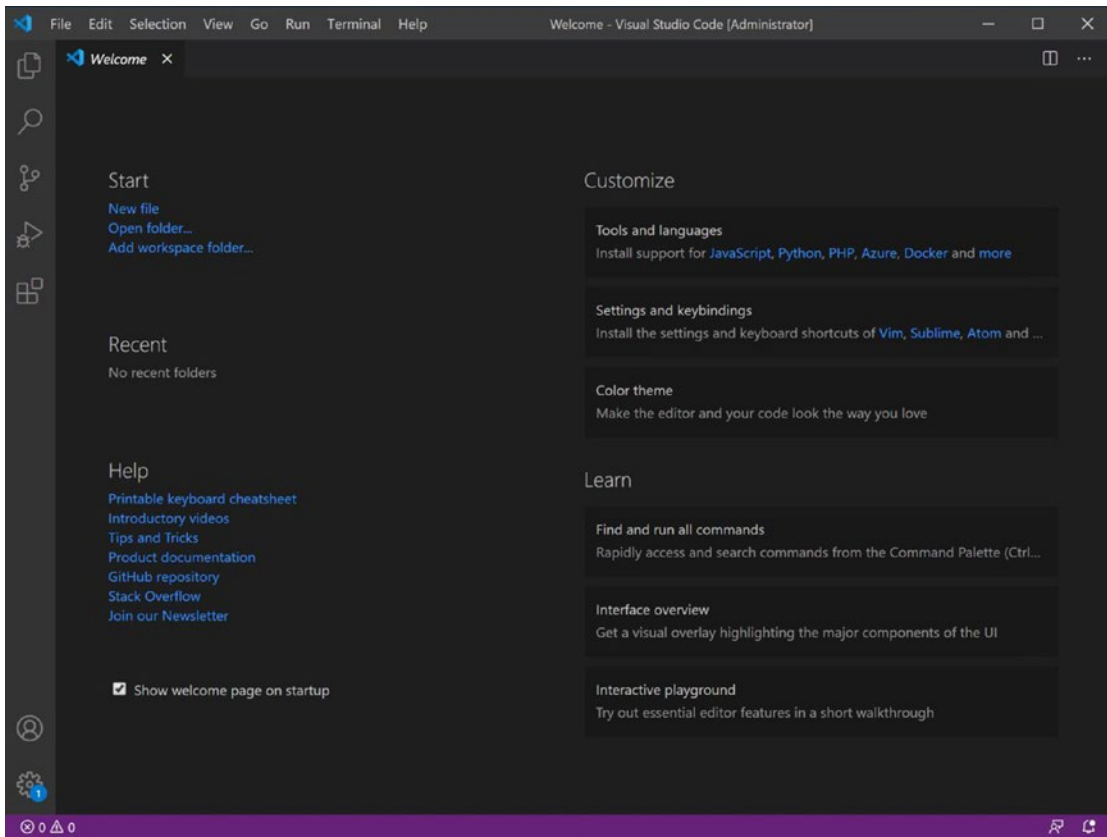
The bottom window is '82d4479597f8c14bb0dd9bbb7bf27e3cdaacc4e68ba1c05df8fc3ce9913cf4c7-json - Notepad' with the following text:

```
11b[49m: Microsoft.AspNetCore.DataProtection.Repositories.FileSystemXmlRepository[60]\r\n", "stream": "stdout", "time": "2020-08-09T11:11:11.111Z"}
: may not be persisted outside of the container. Protected data will be unavailable when container is destroyed.\r\n", "stream": "stdout", "time": "2020-08-09T11:11:11.111Z"}
: .AspNetCore.DataProtection.KeyManagement.XmlKeyManager[35]\r\n", "stream": "stdout", "time": "2020-08-09T11:11:11.111Z"}
: No XML encryptor configured. Key {b9cc1bc3-846c-40a8-aeb7-2ee3aa93759c} may be persisted to storage in unencrypted form.\r\n", "stream": "stdout", "time": "2020-08-09T11:11:11.111Z"}
: form.\r\n", "stream": "stdout", "time": "2020-08-09T11:11:11.111Z"}
: form.
```

This completes the second task, in which you learned several Docker commands, allowing you to run, validate, and troubleshoot containerized application instances. In the next task, I will show you another way to manage containers, using Visual Studio Code – Docker extensions.

Task 3: Integrating Docker extension in Visual Studio Code

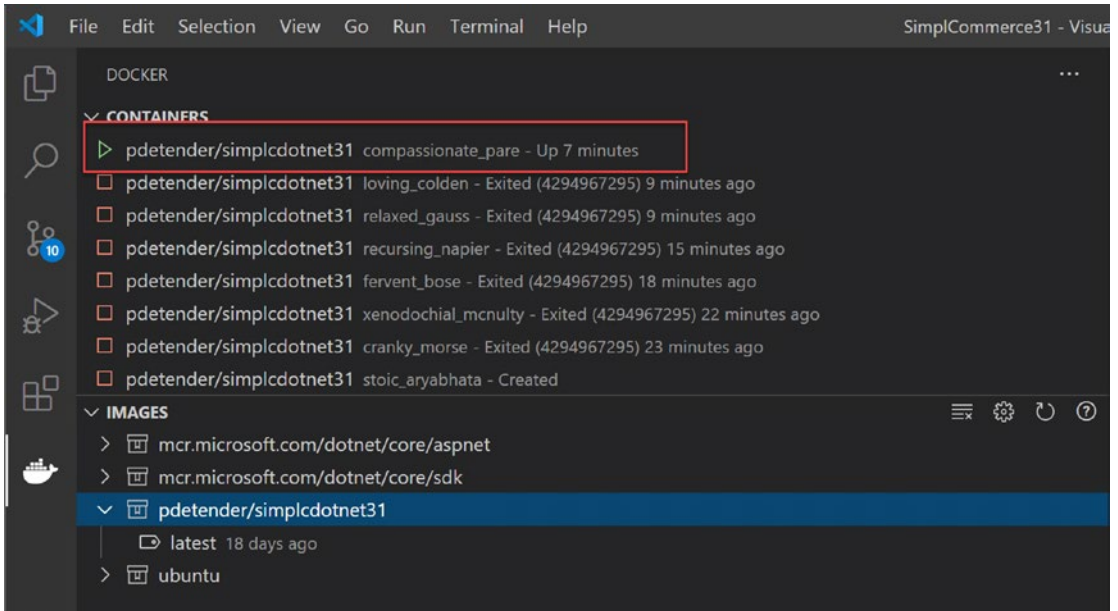
1. From the **Start menu**, launch **“Visual Studio Code.”**



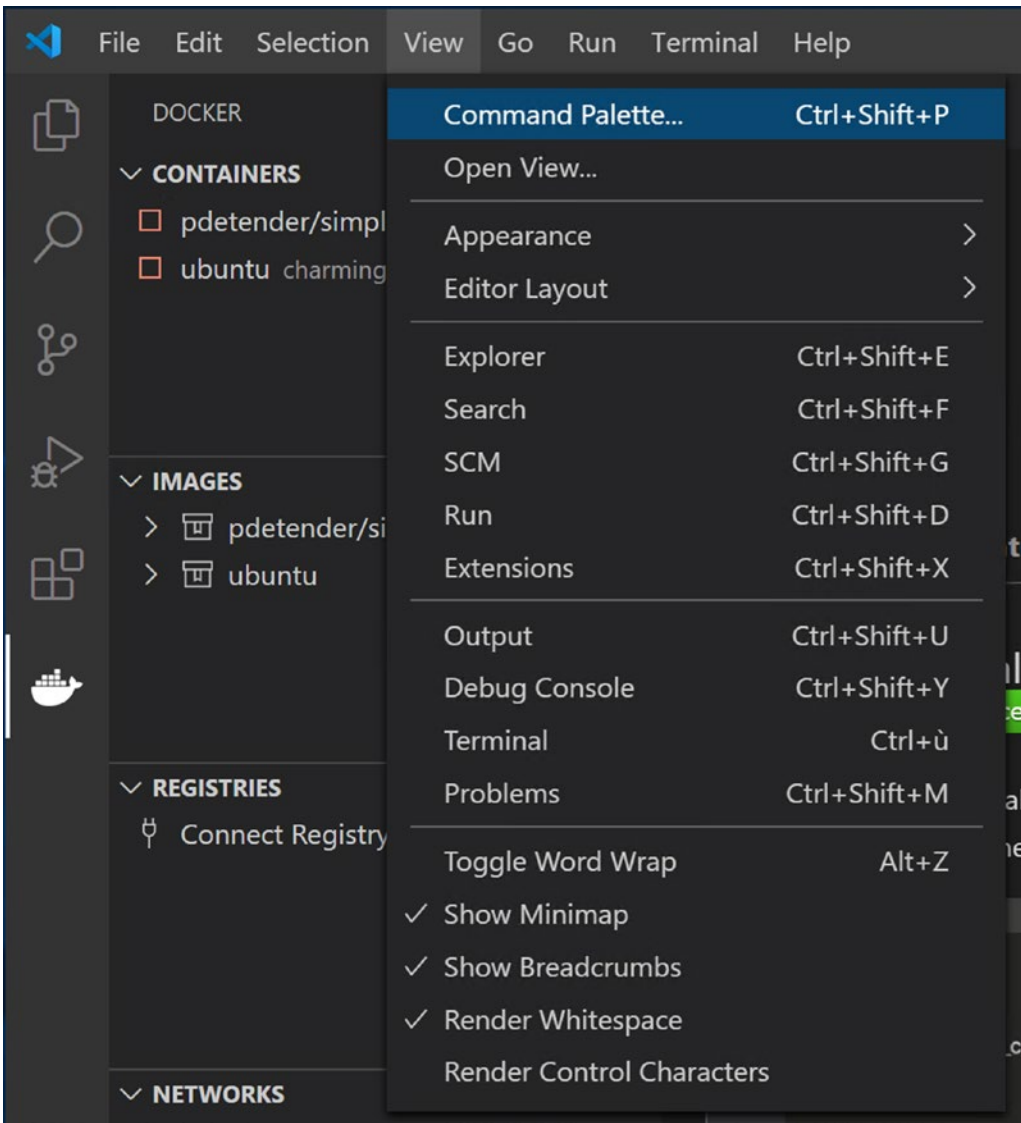
2. From the **Extensions** option, search for **“Docker.”**



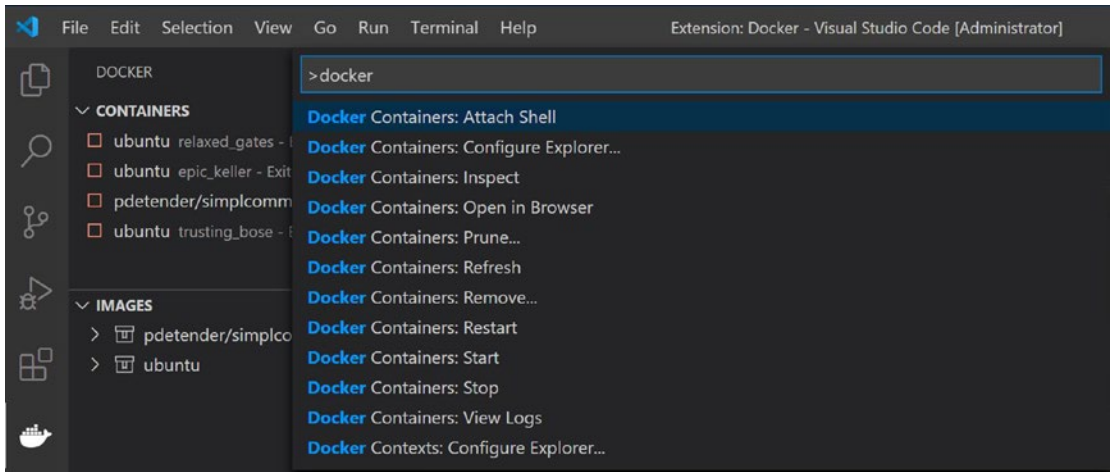
3. **Click the “Install” button**; while not (always) needed, I typically advise to **restart Visual Studio Code** after the installation, guaranteeing it loads successfully. This helped me tremendously in troubleshooting, or avoiding to needing to do that 😊.
4. **Notice the Docker extension installed successfully, by clicking the Docker icon.**



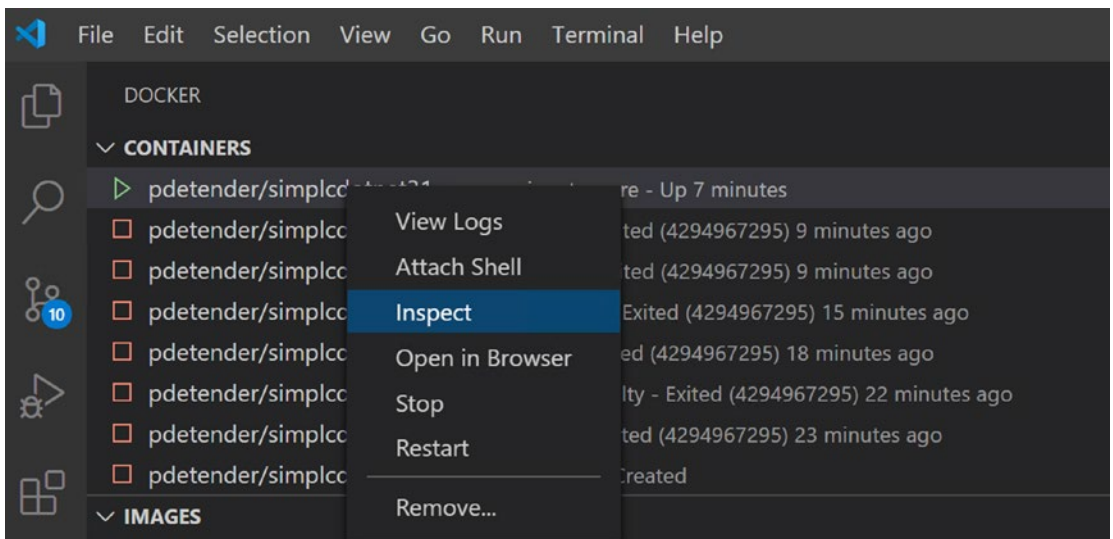
5. **From the left menu, it immediately exposes some information about the Docker environment that is running on the Docker Host:**
 - **Containers:** Lists up the running/previously running containers on this host.
 - **Images:** Lists up the container images.
 - **Registries:** Private Docker-compatible registries, for example, **Azure Container Registry**.
6. **Besides** the information here on the left menu, the extension also comes with **command palette options in the “View” menu**.



7. From “Command Palette,” start typing “docker,” showing a list of different commands available, similar to the ones you used in PowerShell earlier; but now you don’t (always 😊) have to remember them or know the correct syntax or parameters, but rather make use of this list.



- Remember the **docker inspect** command; you can run this now from the **Docker extension** menu.



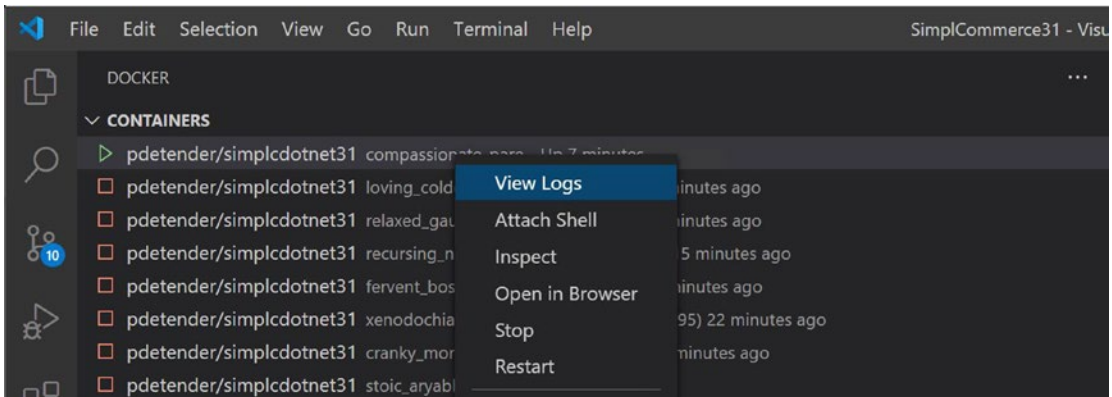
- This provides a similar **log-JSON** file, but directly published within Visual Studio Code.

```

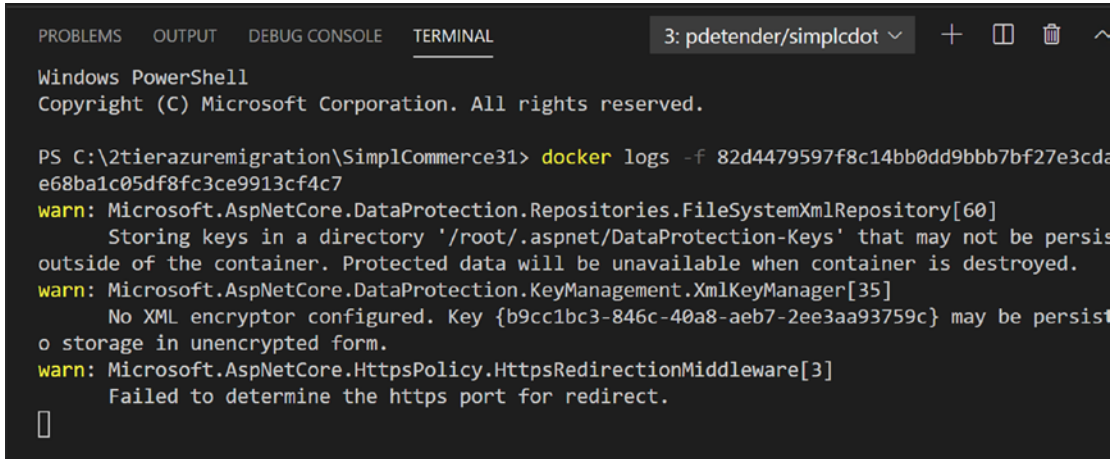
{} simplcdotnet31.json X
12     "Restarting": false,
13     "OOMKilled": false,
14     "Dead": false,
15     "Pid": 363,
16     "ExitCode": 0,
17     "Error": "",
18     "StartedAt": "2020-08-09T22:12:03.524782Z",
19     "FinishedAt": "0001-01-01T00:00:00Z"
20 },
21 "Image": "sha256:bc2c048eef61c9c2c95783c19e7d378ef9208be52f1928e78b1cf416
22 "ResolvConfPath": "",
23 "HostnamePath": "",
24 "HostsPath": "",
25 "LogPath": "C:\\ProgramData\\docker\\containers\\82d4479597f8c14bb0dd9bbb
26 "Name": "/compassionate_pare",
27 "RestartCount": 0,
28 "Driver": "lcow",
29 "Platform": "linux",
30 "MountLabel": "",
31 "ProcessLabel": "",
32 "AppArmorProfile": "",
33 "ExecIDs": null,
34 "HostConfig": {
35     "Binds": null,
36     "ContainerIDFile": "",
37     "LogConfig": {
38         "Type": "json-file",

```

10. Or select “View Logs.”



11. **This exposes** the logging information in a Visual Studio terminal window.



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 3: pdetender/simplcdot
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\2tierazuremigration\SimplCommerce31> docker logs -f 82d4479597f8c14bb0dd9bbb7bf27e3cda
e68ba1c05df8fc3ce9913cf4c7
warn: Microsoft.AspNetCore.DataProtection.Repositories.FileSystemXmlRepository[60]
      Storing keys in a directory '/root/.aspnet/DataProtection-Keys' that may not be persis
outside of the container. Protected data will be unavailable when container is destroyed.
warn: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[35]
      No XML encryptor configured. Key {b9cc1bc3-846c-40a8-aeb7-2ee3aa93759c} may be persis
o storage in unencrypted form.
warn: Microsoft.AspNetCore.HttpsPolicy.HttpsRedirectionMiddleware[3]
      Failed to determine the https port for redirect.

```

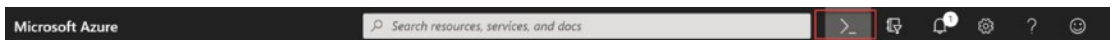
12. **There are a lot of interesting actions available from the Docker extension, giving DevOps teams an easy and single tool to manage their application workloads, from source code to containers and everything in between.**

This completes the third task in which I introduced you to the Docker extension in Visual Studio Code. As you know the basics of operating Docker and containerized workloads, let's move on and reuse this knowledge on Azure.

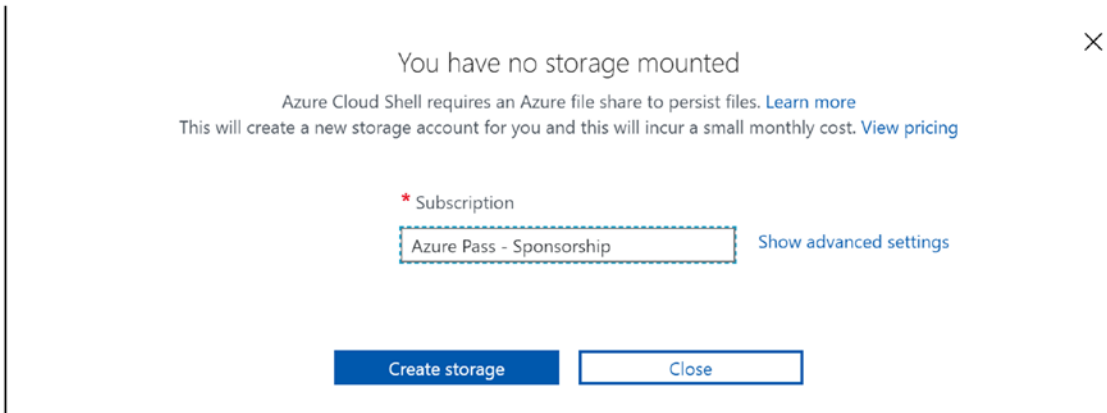
Task 4: Deploying and operating Azure Container Registry

As we have a successfully built Docker container out of the previous task, we can move on to the next step in the process, migrating this container to Azure, starting from pushing it into Azure Container Registry (ACR) and running it as an Azure Container Instance (ACI).

1. **Log on to the Azure Portal**, <http://portal.azure.com>, with your Azure admin credentials. From here, **open Cloud Shell**.



- 2. **Follow** the configuration steps if this is the first time you launched Cloud Shell, by **selecting your Azure subscription** and clicking **“Create storage.”**



Once you are in the shell environment itself, make sure you select **Bash**.



```

Bash
Requesting a Cloud Shell. Succeeded.
Connecting terminal...

Welcome to Azure Cloud Shell

Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell

imlearning@Azure:~$ █

```

```

Bash
}
]
imlearning@Azure:~$ az account list-locations --out table
-----
DisplayName          Name                RegionalDisplayName
-----
East US              eastus              (US) East US
East US 2            eastus2             (US) East US 2
South Central US     southcentralus      (US) South Central US
West US 2            westus2             (US) West US 2
Australia East       australiaeast       (Asia Pacific) Australia East
Southeast Asia       southeastasia       (Asia Pacific) Southeast Asia
North Europe         northeurope         (Europe) North Europe
UK South             uksouth            (Europe) UK South
West Europe          westeurope          (Europe) West Europe
Central US           centralus           (US) Central US
North Central US     northcentralus      (US) North Central US
West US              westus              (US) West US
South Africa North   southafricanorth    (Africa) South Africa North
Central India        centralindia        (Asia Pacific) Central India

```

- Execute the following Azure CLI commands, to **create a new Azure resource group**:

```
az group create --name [SUFFIX]-containersRG --location
<Azure Region Name of choice>
```

```

Bash
imlearning@Azure:~$ az group create --name PDT-containersRG --location westeurope
{
  "id": "/subscriptions/e373a65a-188d-48df-860d-604d07a5790a/resourceGroups/PDT-containersRG",
  "location": "westeurope",
  "managedBy": null,
  "name": "PDT-containersRG",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
imlearning@Azure:~$ █

```

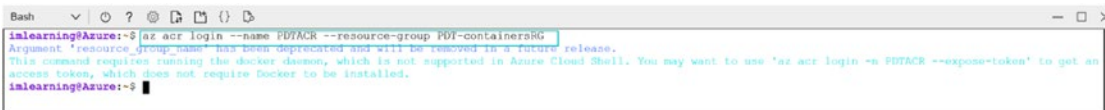
- 4. This is followed by another Azure CLI command to **create Azure Container Registry**:

```
az acr create --resource-group [Suffix]-containerRG
--name [SUFFIX]ACR --sku Basic --admin-enabled true
```



- 5. The next involves connecting to the Azure Container Registry we just created and pushing our Docker image into it. This relies on the following command:

```
az acr login --name [SUFFIX]ACR --resource-group
[SUFFIX]-containerRG
```



- 6. This means we have to execute the remaining commands from our local lab jumpVM, instead of the Azure Cloud Shell. Since we preloaded the Azure CLI on this machine, we can immediately make use of it (FYI, if you need to install this on your local machine when not using the JumpVM, use the following link: <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-windows?view=azure-cli-latest>).

- 7. To **validate** the Azure CLI is installed fine, **open a new PowerShell window**, and initiate the following command:

```
az
```

```

Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\labadmin> az

Welcome to Azure CLI!
-----
Use `az -h` to see available commands or go to https://aka.ms/cli.

Telemetry
-----
The Azure CLI collects usage data in order to improve your experience.
The data is anonymous and does not include commandline argument values.
The data is collected by Microsoft.

You can change your telemetry settings with `az configure`.



Welcome to the cool new Azure CLI!
Use `az --version` to display the current version.
Here are the base commands:

```

- This confirms Azure CLI 2.0 is running as expected. We can continue with our Azure Container Registry creation process. But first, we need to “authenticate” our session to Azure, by running the following command:

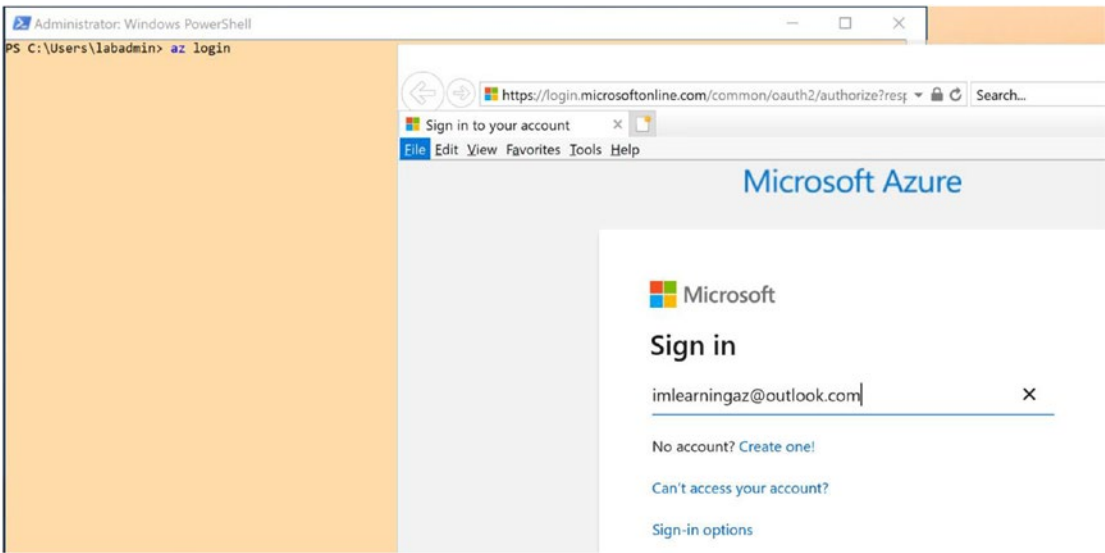
```
az login
```

```

Administrator: Windows PowerShell
PS C:\Users\labadmin> az login

```

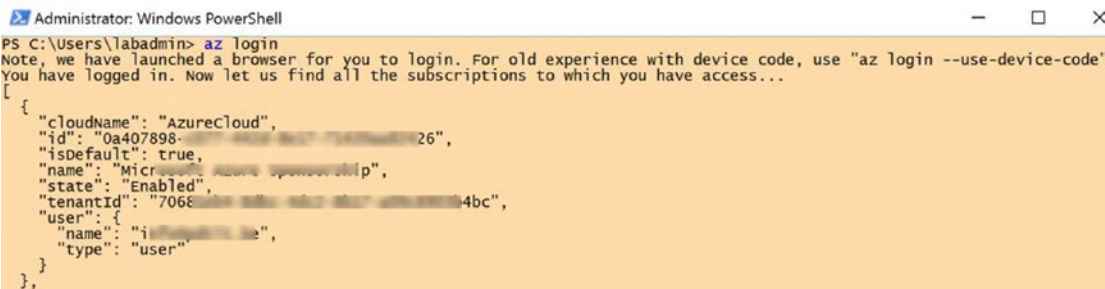
- This opens your Internet browser and prompts for your Azure admin credentials.



10. After successful login, the following information is displayed:



- 11. You can close the Internet browser.
- 12. When you go back to the PowerShell window, it will show you the JSON output of your Azure subscription, related to this Azure admin user.

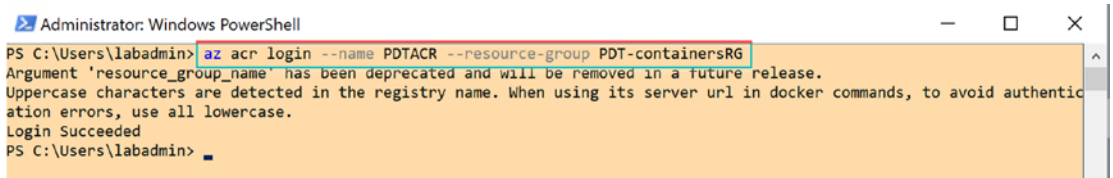


Note If you should have multiple Azure subscriptions linked to the same Azure admin credentials, run the following Azure CLI command to guarantee you are working in the correct subscription:

az account set --subscription "your subscription name here"

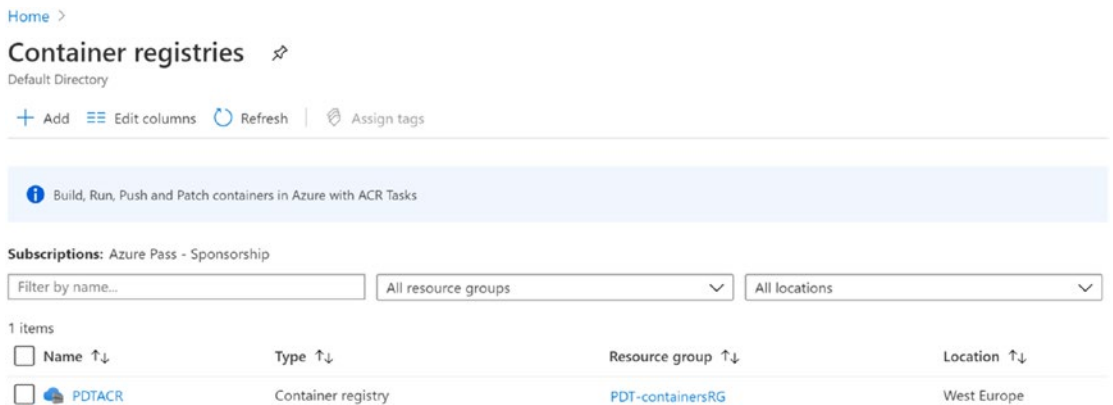
13. Let's try to redo our Azure Container Registry process, by executing the following command:

```
az acr login --name [SUFFIX]ACR --resource-group
[SUFFIX]-containerRG
```



```
Administrator: Windows PowerShell
PS C:\Users\labadmin> az acr login --name PDTACR --resource-group PDT-containersRG
Argument 'resource_group_name' has been deprecated and will be removed in a future release.
Uppercase characters are detected in the registry name. When using its server url in docker commands, to avoid authentication errors, use all lowercase.
Login Succeeded
PS C:\Users\labadmin>
```

14. **You can also validate the Azure Container Registry from the Azure Portal.**



Home >

Container registries

Default Directory

+ Add Edit columns Refresh Assign tags

Build, Run, Push and Patch containers in Azure with ACR Tasks

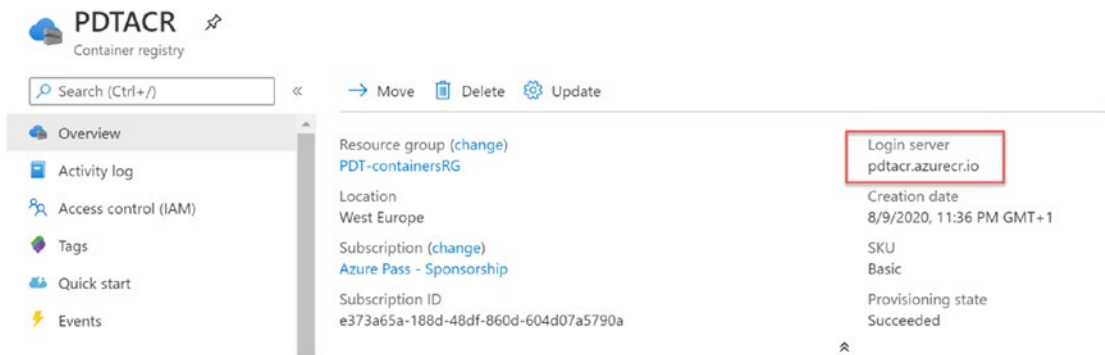
Subscriptions: Azure Pass - Sponsorship

Filter by name... All resource groups All locations

1 items

| <input type="checkbox"/> | Name ↑↓ | Type ↑↓ | Resource group ↑↓ | Location ↑↓ |
|--------------------------|---------|--------------------|-------------------|-------------|
| <input type="checkbox"/> | PDTACR | Container registry | PDT-containersRG | West Europe |

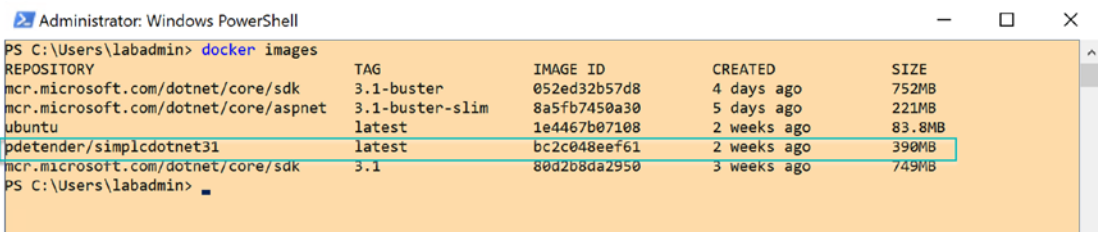
15. And validate the details of the Azure Container Registry resource.



Task 5: Deploying and running Azure Container Instance

- As we now have connectivity toward ACR, we can push our Docker image to it. There is however a dependency that the name of our Docker image needs to have the name of the Azure Container Registry in it. So we first need to update the Docker image tag for our Docker image, by executing the following command:

`docker images` (to get the image ID number)



`docker tag bc2c [SUFFIX]ACR.azurecr.io/<nameyouwanttogive>`

`docker images` (to validate the “new” image)

```

Administrator: Windows PowerShell
PS C:\Users\labadmin> docker tag bc2c pdtacr.azurecr.io/simplcdotnet31
PS C:\Users\labadmin> docker images
REPOSITORY                                TAG                IMAGE ID           CREATED           SIZE
mcr.microsoft.com/dotnet/core/sdk         3.1-buster        052ed32b57d8      4 days ago       752MB
mcr.microsoft.com/dotnet/core/aspnet     3.1-buster-slim   8a5fb7450a30      5 days ago       221MB
ubuntu                                    latest            1e4467b07108      2 weeks ago      83.8MB
pdtacr.azurecr.io/simplcdotnet31        latest            bc2c048eef61      2 weeks ago      390MB
pdetender/simplcdotnet31                latest            bc2c048eef61      2 weeks ago      390MB
mcr.microsoft.com/dotnet/core/sdk        3.1                80d2b8da2950      3 weeks ago      749MB
PS C:\Users\labadmin>

```

Notice the image ID is identical, as technically, all we did was create a clone with a new name.

- Execute the following command to upload this image to the Azure Container Registry:

```
docker push [SUFFIX]ACR.azurecr.io/<nameyouwanttogive>
```

```

Administrator: Windows PowerShell
PS C:\Users\labadmin> docker push pdtacr.azurecr.io/simplcdotnet31
The push refers to repository [pdtacr.azurecr.io/simplcdotnet31]
35c99434fc97: Retrying in 1 second
05da7522dc67: Pushing [=====>] 1.02MB/1.02MB
a30f768f6a34: Pushing [=====>] 9.913MB/94.54MB
ec564de22418: Pushing [=====>] 61.95kB/61.95kB
51ac662debb3: Pushing [=====>] 2.38MB/28.53MB
d86ec58d3137: Waiting
8c30868fe23a: Waiting
886801dff0ea: Waiting
49b759454bb2: Waiting
95ef25a32043: Waiting

```

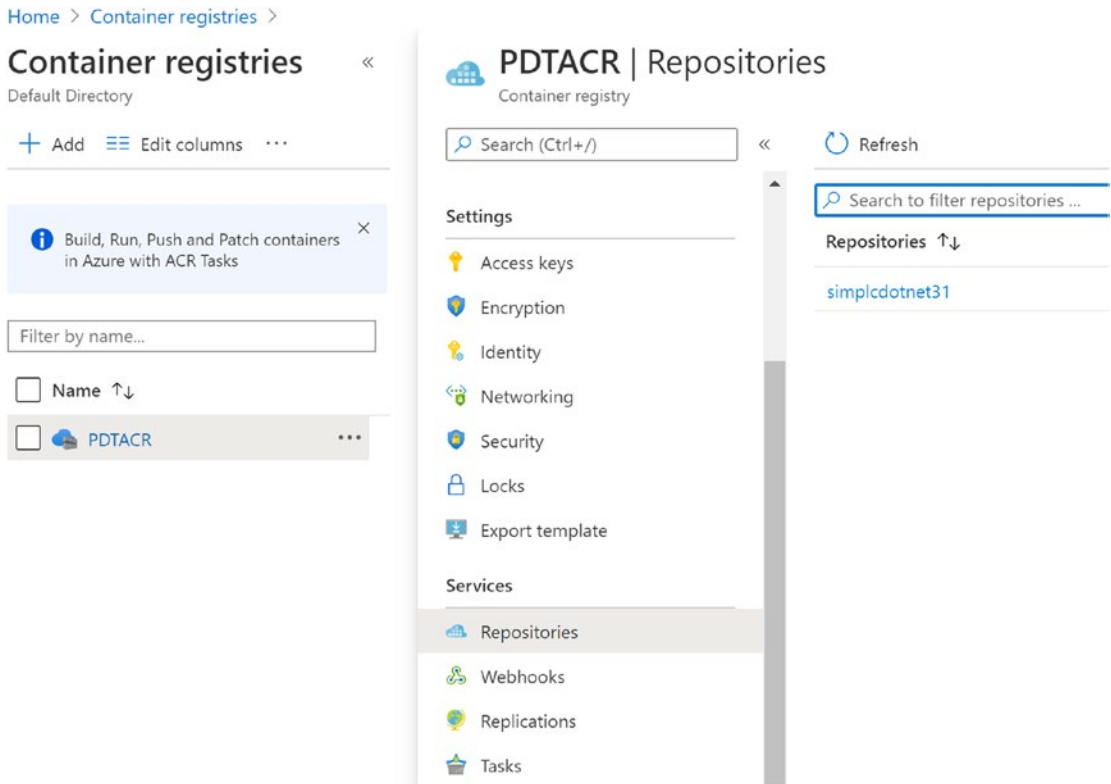
- Wait for this process to complete successfully; depending on Internet connection speed, this might take some time.

```

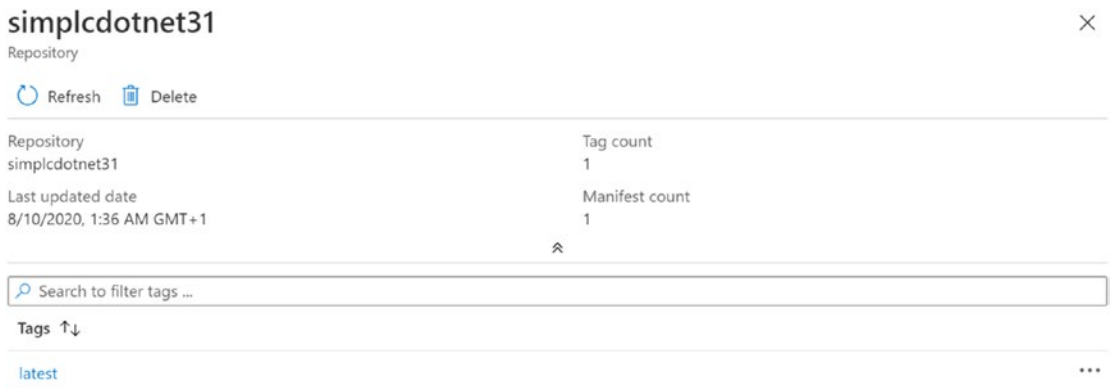
PS C:\Users\labadmin> docker push pdtacr.azurecr.io/simplcdotnet31
The push refers to repository [pdtacr.azurecr.io/simplcdotnet31]
35c99434fc97: Layer already exists
05da7522dc67: Layer already exists
a30f768f6a34: Pushing [=====>] 94.54MB/94.54MB
ec564de22418: Layer already exists
51ac662debb3: Layer already exists
d86ec58d3137: Layer already exists
8c30868fe23a: Layer already exists
886801dff0ea: Layer already exists
49b759454bb2: Layer already exists
95ef25a32043: Pushing [=====>] 80.07MB/80.07MB
latest: digest: sha256:19816a782092ca2e6c27329c973490d90377adbbe3958ba1fc2cc7670923cc70 size: 2425
PS C:\Users\labadmin>

```

4. From the Azure Portal ► All services ► Azure Container registries, select the ACR you created earlier.



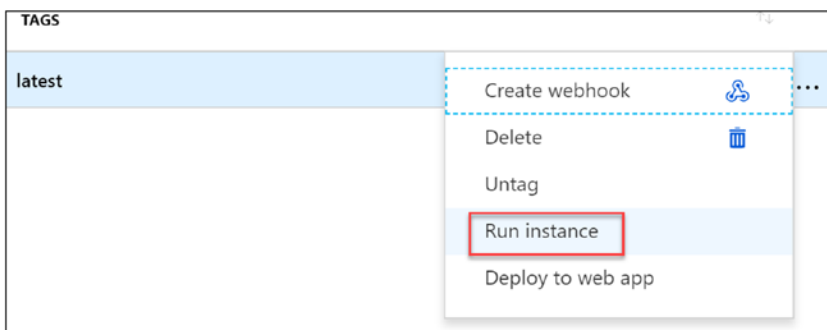
5. **Click the** <yourcontainername> repository, which opens the specific details for this image, exposing its version (we used the default version tag “latest,” but this could also be dev, test, v1.1, v2.5, etc. in a real-life scenario).



This completes this task, in which you created an Azure Container Registry (ACR), tagged a Docker container image, and uploaded this to Azure Container Registry repositories. In the next task, you will deploy this repository into a running state using Azure Container Instance (ACI).

Task 5: Running an Azure Container Instance from a Docker image in Azure Container Registry

1. From the **Azure Container Registry**, browse to Repositories, select your repository, and click “latest”; from here, click the ... next to latest, and **choose Run instance**.



2. This opens the **Create container instance blade**. Complete the parameter fields using the following information:
 - **Container name:** [suffix]simplcdotnet31 (or any other name you like)
 - **OS type:** Linux

- **Subscription:** Your Azure subscription
- **Resource group:** Select [SUFFIX]-containerRG as resource group
- **Location:** Same location as where you deployed ACR

Leave all other settings unchanged (one core, 1.5 GB memory, public IP address Yes, and port 80).

Home > Container registries > PDTACR | Repositories > simplcdotnet31 >

simplcdotnet31

Repository

Refresh Delete

Repository
simplcdotnet31

Last updated date
8/10/2020, 1:36 AM GMT+1

Tag count
1

Manifest count
1

Tags ↑↓

- latest

Create container instance

Container name *

Container image

OS type
 Linux Windows

Subscription *

Resource group *

[Create new](#)

Location *

Number of cores

Memory (GB) *

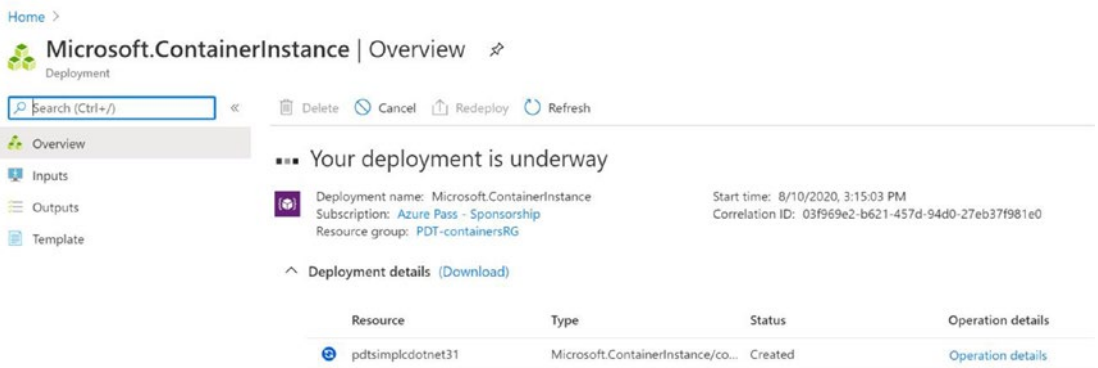
Public IP address
 Yes No

Port *

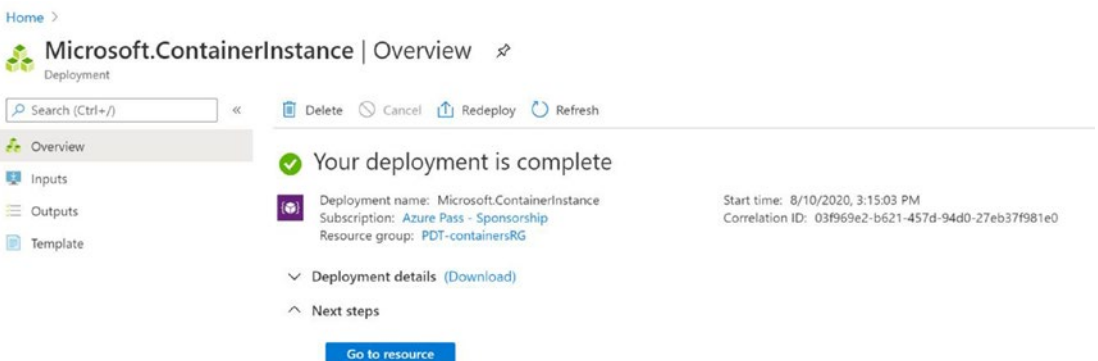
3. Click **OK** to have the container instance created. Deployment initialization kicks off.



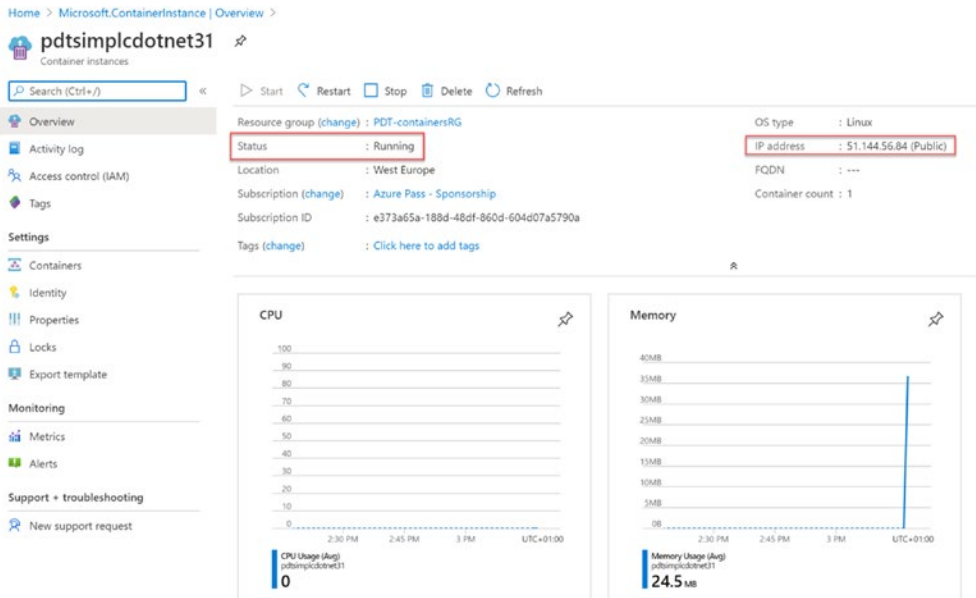
4. Follow the details by clicking the “Your deployment is underway” from the Notifications area.



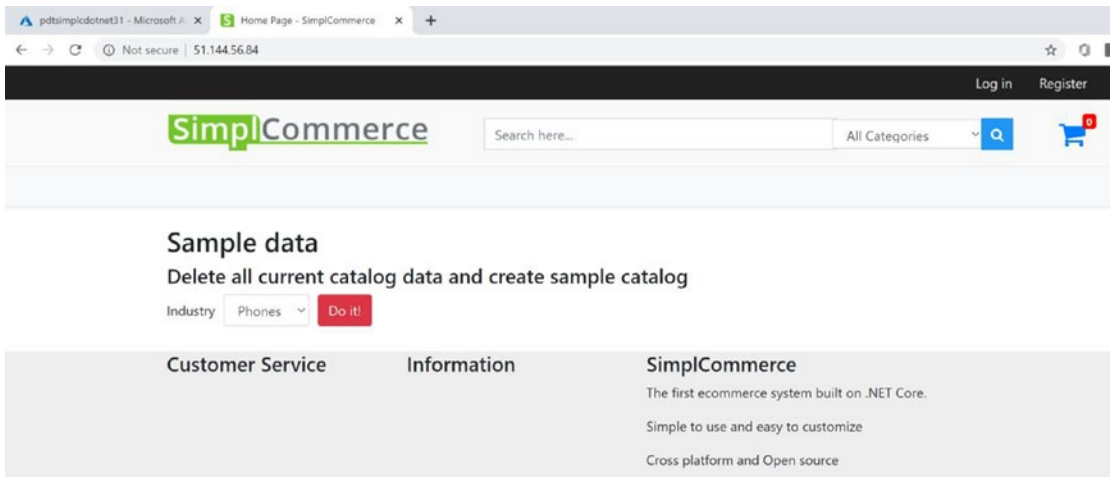
5. Wait for the deployment process to complete successfully, which should typically be within a minute.



6. Once the deployment is finished, **click Go to resource**; or open **the Azure Container Instance** in the portal (All services ► Container instances), and **browse to the ACI “instance”** that just got created.



7. **Copy the IP address** for this Azure Container Instance, or directly browse to it from your Internet browser, which should load your application successfully.



There's the webshop again; similar to the "local" Docker container behavior, it opens the home page, asking for a product offering. While I'm not showing the outcome here, you already know how this works.

- Back in the Azure Portal ► Azure Container instances blade, browse to **Containers** under **Settings**. Within the **Events** tab, there are more details about the running container itself, as well as providing a view on the process of pulling the image and running it.

Home > Microsoft.ContainerInstance | Overview >

pdtimplcdotnet31 | Containers

Container instances

Search (Ctrl+F) Refresh

1 container

| Name | Image | State | Previous state | Start time | Restart count |
|------------------|-----------------------------------|---------|----------------|----------------------|---------------|
| pdtimplcdotnet31 | pdtacr.azurecr.io/simplcdotnet... | Running | - | 2020-08-10T14:15:42Z | 0 |

Settings

- Containers
- Identity
- Properties
- Locks
- Export template

Monitoring

- Metrics
- Alerts

Support + troubleshooting

- New support request

Events Properties Logs Connect

Display time zone Local time UTC

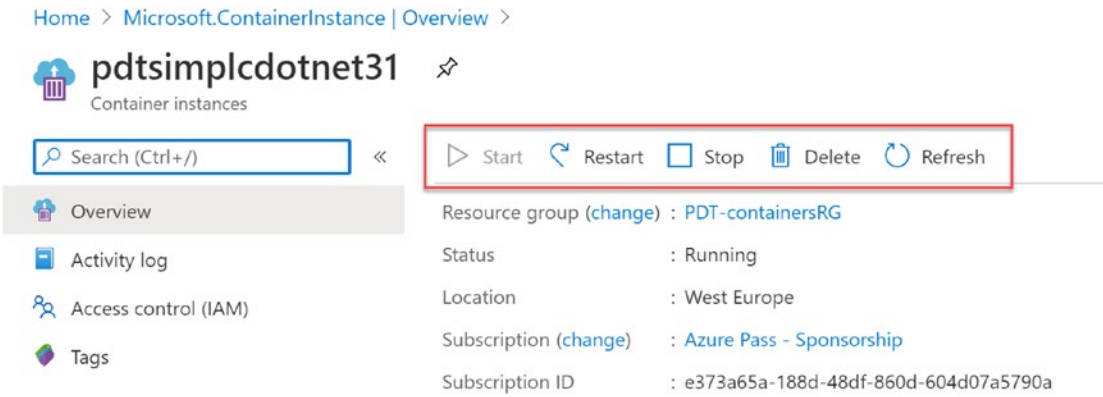
| Name | Type | First timestamp | Last timestamp | Message | Count |
|---------|---------|--------------------------|--------------------------|-----------------------------------|-------|
| Started | Normal | 8/10/2020, 3:15 PM GMT+1 | 8/10/2020, 3:15 PM GMT+1 | Started container | 1 |
| Created | Normal | 8/10/2020, 3:15 PM GMT+1 | 8/10/2020, 3:15 PM GMT+1 | Created container | 1 |
| Pulled | Normal | 8/10/2020, 3:15 PM GMT+1 | 8/10/2020, 3:15 PM GMT+1 | Successfully pulled image "pdt... | 1 |
| Pulling | Normal | 8/10/2020, 3:15 PM GMT+1 | 8/10/2020, 3:15 PM GMT+1 | pulling image "pdtacr.azurecr... | 2 |
| BackOff | Normal | 8/10/2020, 3:15 PM GMT+1 | 8/10/2020, 3:15 PM GMT+1 | Back-off pulling image "pdtacr... | 1 |
| Failed | Warning | 8/10/2020, 3:15 PM GMT+1 | 8/10/2020, 3:15 PM GMT+1 | Error: ImagePullBackOff | 1 |

- Next, click the **Logs** tab, showing you similar output from the log-JSON option you used earlier by executing "docker inspect" from the command line or selecting "Inspect" from the Docker extension in Visual Studio Code.

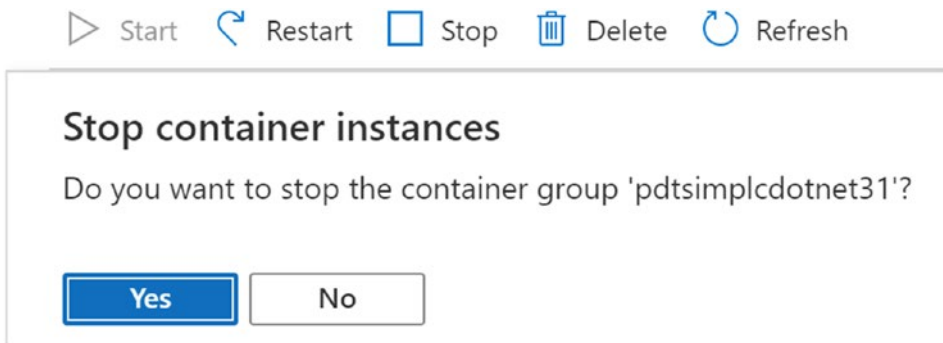
Events Properties Logs Connect

```
[40m][1m][33mwarn][39m][22m][49m: Microsoft.AspNetCore.DataProtection.Repositories.FileSystemXmlRepository[60]
Storing keys in a directory '/root/.aspnet/DataProtection-Keys' that may not be persisted outside of the container. Protected data will be unavailable when conta
iner is destroyed.
[40m][1m][33mwarn][39m][22m][49m: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[35]
No XML encryptor configured. Key {e545782a-b6d5-448a-839a-95077c3a88e9} may be persisted to storage in unencrypted form.
[40m][1m][33mwarn][39m][22m][49m: Microsoft.AspNetCore.Policy.HttpsPolicy.HttpsRedirectionMiddleware[3]
Failed to determine the https port for redirect.
```

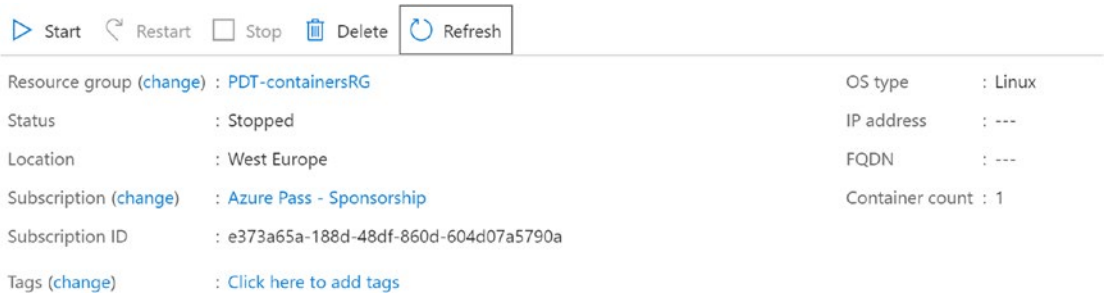
- 10. **Return** to the **Overview** section of the Azure Container instances blade, and notice the action buttons on top, allowing you to **start, restart, stop, or delete** the container instance.



- 11. **Nice to remember is that you don't pay anything for a "stopped" container**, so it could become handy to **stop** the container instance for now, saving a few bucks of your monthly Azure bill.



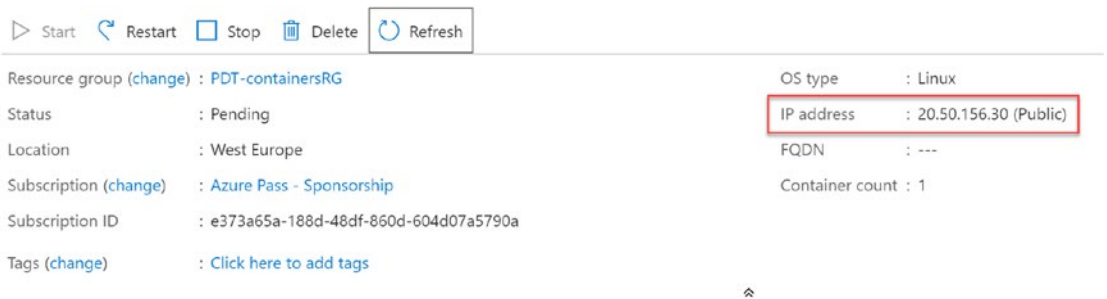
12. Checking back from the instance Overview tab, notice the public IP address is also “released” from the running instance.



Start Restart Stop Delete Refresh

| | |
|--|---------------------|
| Resource group (change) : PDT-containersRG | OS type : Linux |
| Status : Stopped | IP address : --- |
| Location : West Europe | FQDN : --- |
| Subscription (change) : Azure Pass - Sponsorship | Container count : 1 |
| Subscription ID : e373a65a-188d-48df-860d-604d07a5790a | |
| Tags (change) : Click here to add tags | |

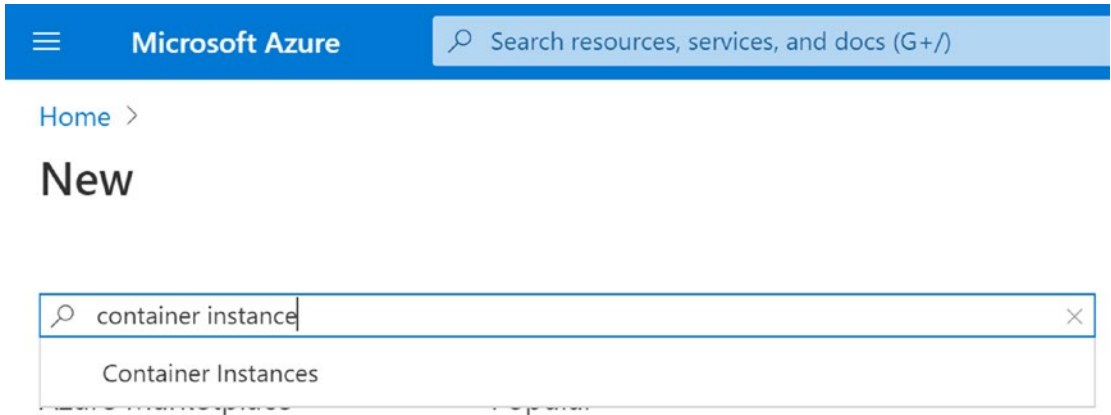
13. **Start** the container instance again, by clicking the Start button; wait a few seconds, and check on the updated settings. The container instance got a new public IP address.



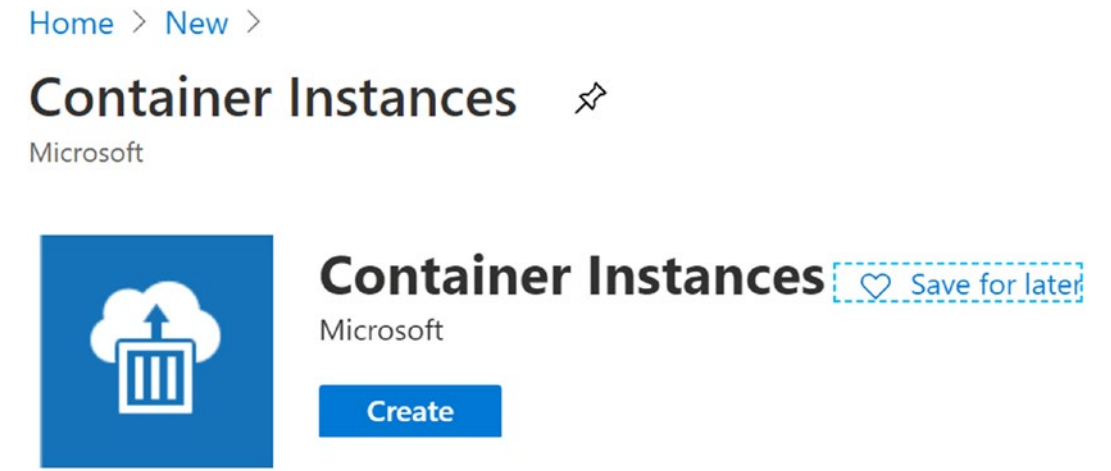
Start Restart Stop Delete Refresh

| | |
|--|------------------------------------|
| Resource group (change) : PDT-containersRG | OS type : Linux |
| Status : Pending | IP address : 20.50.156.30 (Public) |
| Location : West Europe | FQDN : --- |
| Subscription (change) : Azure Pass - Sponsorship | Container count : 1 |
| Subscription ID : e373a65a-188d-48df-860d-604d07a5790a | |
| Tags (change) : Click here to add tags | |

- 14. **This** is probably not something you want in a production environment, so let's spin up a new container instance, this time starting from the "+ Create Resource," and search for "**container instance.**"



- 15. **Confirm the creation, by clicking the Create button.**



16. Provide the necessary settings, following these information guidelines:

Create container instance

Azure Container Instances (ACI) allows you to quickly and easily run containers on Azure without managing servers or having to learn new tools. ACI offers per-second billing to minimize the cost of running containers on the cloud. [Learn more about Azure Container Instances](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

| | |
|--------------------|--------------------------|
| Subscription * ⓘ | Azure Pass - Sponsorship |
| Resource group * ⓘ | PDT-containersRG |

[Create new](#)

Container details

| | |
|--------------------|--|
| Container name * ⓘ | pdtaacimplcdotnet31 |
| Region * ⓘ | (Europe) West Europe |
| Image source * ⓘ | <input type="radio"/> Quickstart images <input checked="" type="radio"/> Azure Container Registry <input type="radio"/> Docker Hub or other registry |
| Registry * ⓘ | PDTACR |
| Image * ⓘ | simplcdotnet31 |
| Image tag * ⓘ | latest |
| OS type | Linux |
| Size * ⓘ | 1 vcpu, 1.5 GiB memory, 0 gpus |

[Change size](#)

Review + create

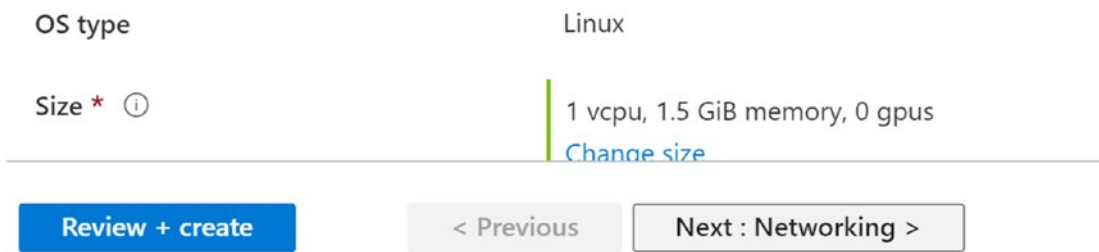
< Previous

Next : Networking >

- **Subscription: Your Azure subscription**
- **Resource group: [SUFFIX]-ContainersRG**
- **Container name: Unique name for this container instance**
- **Region: Same Azure region as Azure Container Registry**

- **Image source: Azure Container Registry**
- **Registry: <Your Azure Container Registry>**
- **Image: <Your Azure Container Repository>**
- **Image tag: latest**
- **OS type: Linux**
- **Size: 1 vcpu, 1.5 GiB memory**

17. Where this is similar to the previous way of deploying an Azure Container Instance, only driven directly from Azure Container Registry repositories, we take it a small step further by going through some additional configuration parameters. Continue by **clicking the Next: Networking button**



18. From the **Networking** tab, notice the default **networking type is “Public,”** allowing a direct connection from the Internet to your running container instance. Switching this to **“Private”** allows you to define **what Azure Virtual Network and subnet** you want to deploy this container instance into.

To see this in action, select the jumpvmVNet.

Basics **Networking** Advanced Tags Review + create

Choose between three networking options for your container instance:

- **'Public'** will create a public IP address for your container instance.
- **'Private'** will allow you to choose a new or existing virtual network for your container instance. This is not yet available for Windows containers.
- **'None'** will not create either a public IP or virtual network. You will still be able to access your container logs using the command line.

Networking type

Public Private None

Virtual network * ⓘ

jumpvmVNet

[Create new](#)

Subnet * ⓘ

Subnet (10.1.0.0/24)

[Manage subnet configuration](#)

Ports ⓘ

| Ports | Ports protocol | |
|----------------------|----------------------|---|
| 80 | TCP |  |
| <input type="text"/> | <input type="text"/> | <input type="text"/> |

19. Although the subnet is automatically pulled up from the JumpVMVNet settings, **we cannot use this subnet to mix container instances with virtual machines**. This is also emphasized from this error message (if you try to deploy this):

Errors



Summary

Raw Error

ERROR DETAILS



- ✓ The resource operation completed with terminal provisioning state 'Failed'. (Code: ResourceDeploymentFailure)
 - At least one resource deployment operation failed. Please list deployment operations for details. Please see <https://aka.ms/DeployOperations> for usage details. (Code: DeploymentFailed), { "error": { "code": "SubnetDelegationsCannotChangeWhenSubnetUsedByResource", "message": "Delegations of subnet /subscriptions/e373a65a-188d-48df-860d-604d07a5790a/resourceGroups/PDT-JumpVMRG/providers/Microsoft.Network/virtualNetworks/jumpvmVNet/subnets/Subnet cannot be changed from [] to [Microsoft.ContainerInstance/containerGroups] because it is being used by the resource /subscriptions/e373a65a-188d-48df-860d-604d07a5790a/resourceGroups/PDT-JumpVMRG/providers/Microsoft.Network/networkInterfaces/jumpvmnic/ipConfigurations/ipconfig1.", "details": [] } } (Code: BadRequest)

20. Instead, **click “Manage subnet configuration,”** which redirects you to the Azure VNet and Subnet settings. Here, **add a subnet**, by clicking the + **Subnet** button.

Home > New > Container Instances > Create container instance >

<.> jumpvmVNet | Subnets

Virtual network

Search (Ctrl+)

<< + Subnet + Gateway subnet Refresh

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems

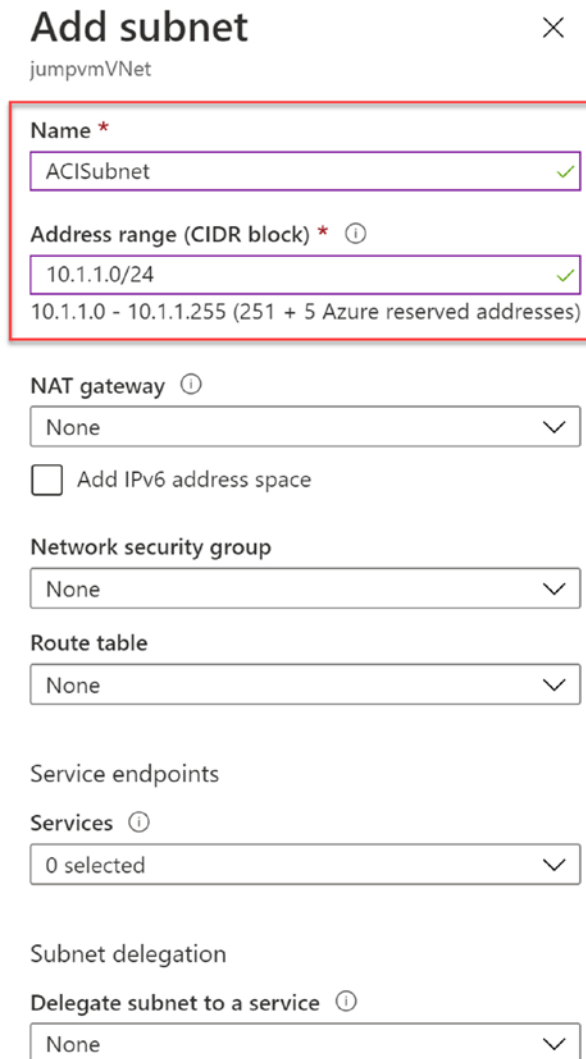
Search subnets

| Name ↑↓ | IPv4 ↑↓ |
|---------|-----------------------------|
| Subnet | 10.1.0.0/24 (250 available) |

21. From the **Add subnet** blade, provide the following parameters:

- **Name:** ACISubnet
- **Address range:** 10.1.1.0/24

Leave all other default settings, and confirm by **clicking OK**.



Add subnet ×

jumpvmVNet

Name *

ACISubnet ✓

Address range (CIDR block) * ⓘ

10.1.1.0/24 ✓

10.1.1.0 - 10.1.1.255 (251 + 5 Azure reserved addresses)

NAT gateway ⓘ

None ▾

Add IPv6 address space

Network security group

None ▾

Route table

None ▾

Service endpoints

Services ⓘ

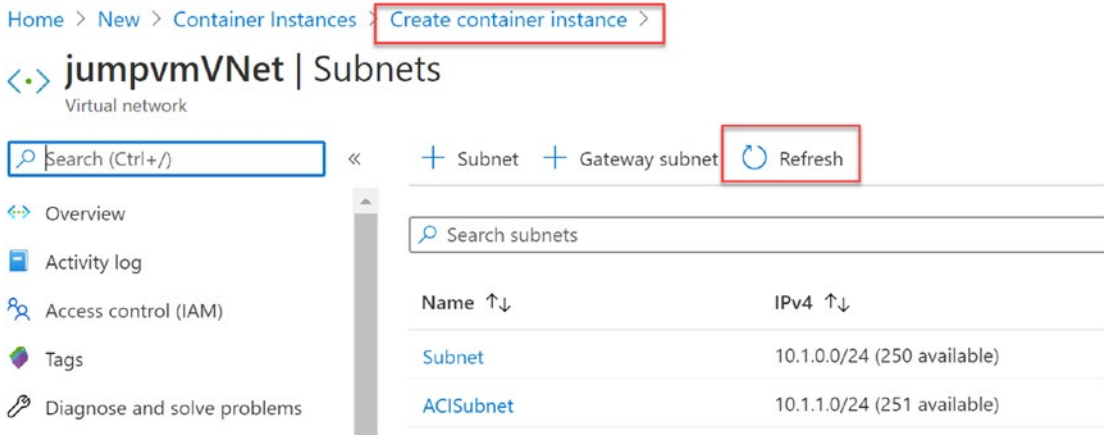
0 selected ▾

Subnet delegation

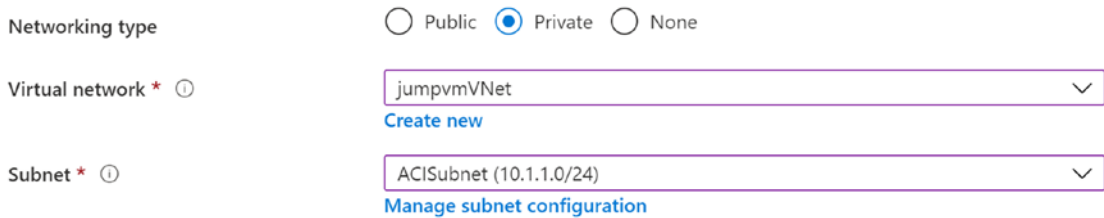
Delegate subnet to a service ⓘ

None ▾

- 22. **Refresh** the list of subnets; notice the **ACISubnet** will be in the list now. Next, click **“Create container instance”** from the breadcrumbs link in the portal, which brings you back to the Azure Container Instance creation wizard.



- 23. **This time**, select the **ACISubnet** in the Network and Subnet settings.



- 24. Move on to the next step in the ACI creation wizard, by **clicking the Next:Advanced button**. Here, one can specify when a container should restart, where the default is **On failure**, but could also be **Always** or **Never**.

In the **Environment variables** section, one could provide specific application variables, for example, to identify dev/test or production settings, database connection strings, and the like.

Basics Networking Advanced Tags Review + create

Configure additional container properties and variables.

Restart policy ⓘ On failure

Environment variables

Key

Always

Never

Command override ⓘ

Example: ["/bin/bash", "-c", "echo hello; sleep 100000"]

- That’s all we need to configure here; continue the deployment by **clicking the “Review + create” button and confirming “Create” once more by clicking the button.** This will kick off the creation of the second Azure Container Instance.

Microsoft.ContainerInstances-20200810184532 | Overview

Deployment

Search (Ctrl+F) Delete Cancel Redeploy Refresh

We'd love your feedback! →

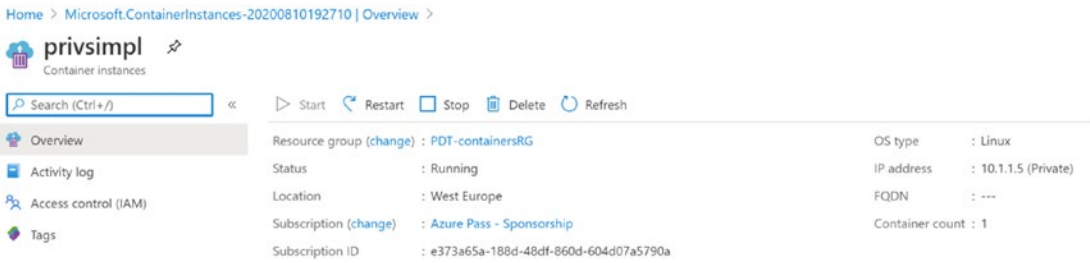
■ ■ ■ Your deployment is underway

Deployment name: Microsoft.ContainerInstances-20200810184532 Start time: 8/10/2020, 6:55:24 PM
 Subscription: Azure Pass - Sponsorship Correlation ID: aa1ccc0f-87ac-4848-b103-f4dd715982a9
 Resource group: PDT-containersRG

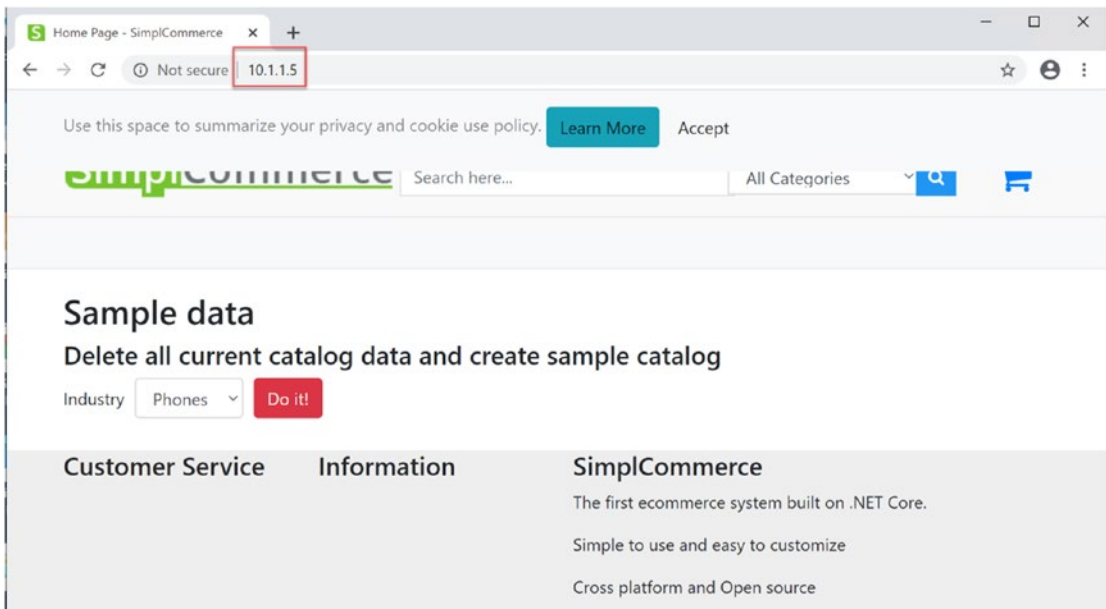
Deployment details (Download)

| Resource | Type | Status | Operation details |
|----------------------------------|-----------------------------------|---------|-----------------------------------|
| pdtsimplaci | Microsoft.ContainerInstance/co... | Created | Operation details |
| pdtsimplaci-networkProfile | Microsoft.Network/networkProf... | Created | Operation details |
| Microsoft.ContainerInstances-202 | Microsoft.Resources/deployme... | OK | Operation details |

- After about a minute, the private Azure Container Instance is ready; nothing is really different than before, besides that the **IP address is now an internal IP range-based one**; this would mean the containerized workload is reachable from within the JumpVM itself.



- 27. (If not already) **Open an RDP session** to the **JumpVM server**, and once logged on, **connect to the IP address** of this Azure Container Instance from your browser.



- 28. Nice, achievement unlocked!

This completes this task, in which you learned about Azure Container Instance for public Internet-facing running workloads, as well as internal/private running ones.

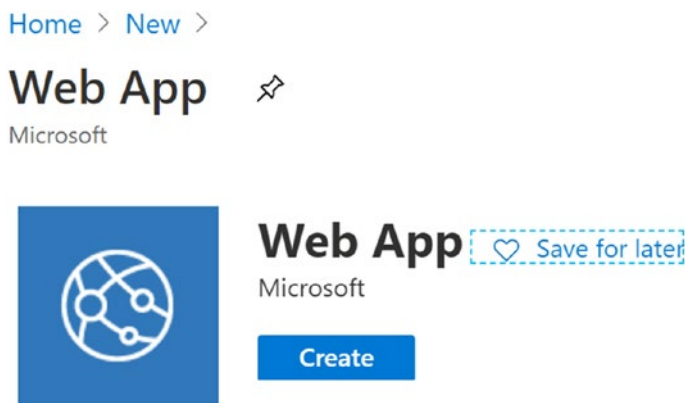
Task 6: Deploying and operating Azure Web App for Containers

Another method to run containerized workloads in Azure Platform as a Service outside of Azure Container Instance is **Azure Web App for Containers**. Easily said, it gives you all (or most) of the Azure Web Apps features, but instead of publishing source code, you publish and run a Docker container.

Main differences compared to Azure Container Instance are that it allows for scalability, supports deployment slot swapping, and is linked to App Service plan consumption costs, instead of ACI running costs.

That's what you will deploy and run in this task.


1. **Start from the Azure Portal ► Create New Resource ► Web App.**



2. **Click the Create** button to open the Create Web App blade. Complete the required parameters as follows:
 - **App name:** [suffix]contwebapp.azurewebsites.net
 - **- Resource group:** [SUFFIX]-ContainerRG
 - **- OS:** Linux
 - **- Publish:** Docker Image
 - **- Region:** Same region as Azure Container Registry


Create Web App


Basics Docker Monitoring Tags Review + create

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. [Learn more](#) 


Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ Azure Pass - Sponsorship 



Resource Group * ⓘ PDT-containersRG 
[Create new](#)

Instance Details

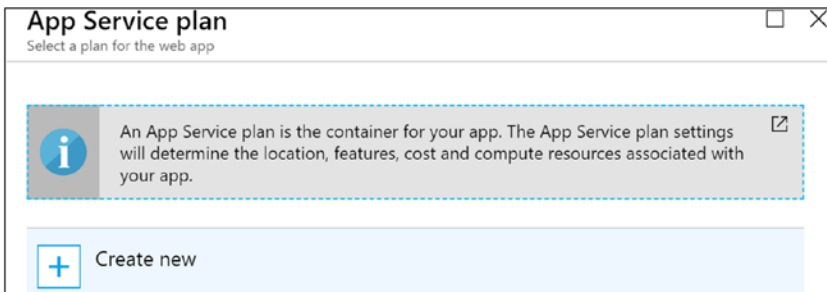
Name * pdtcontwebapp  .azurewebsites.net

Publish * Code Docker Container

Operating System * Linux Windows

Region * West Europe 
 Not finding your App Service Plan? Try a different region.


3. You also need to define the App Service plan parameters.
4. For the **Service plan** parameter, **click Create new**.



5. **Complete** the required parameters for the App Service plan as follows:

- **App Service plan:** [SUFFIX]contwebappPlan.
- **Location:** Same region as where you want to deploy the Azure Web App.
- **Pricing tier:** Select the Premium V2 P1v2 plan.

App Service Plan

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. [Learn more](#) 

Linux Plan (West Europe) * 

(New) pdtcontwebappplan 

[Create new](#)

Skus and sizes *

Premium V2 P1v2

210 total ACU, 3.5 GB memory

[Change size](#)

6. And confirm the plan with **OK**. **Click Next: Docker** to continue the configuration steps.
7. While we could use the same container from Azure Container Registry as in the previous task, let's try something with **Public Docker Hub** this time, showing you running container instances on Azure (in any supported way) doesn't require Azure Container Registry.

Complete the following settings and parameters:

- **Options: Single Container**
- **Image Source: Docker Hub**
- **Access Type: Public**
- **Image and tag: pdetender/simplcdotnet31**

CHAPTER 7 LAB 5: DEPLOYING DOCKER AND RUNNING AZURE CONTAINER WORKLOADS

Basics **Docker** Monitoring Tags Review + create

Pull container images from Azure Container Registry, Docker Hub or a private Docker repository. App Service will deploy the containerized app with your preferred dependencies to production in seconds.

Options

Image Source

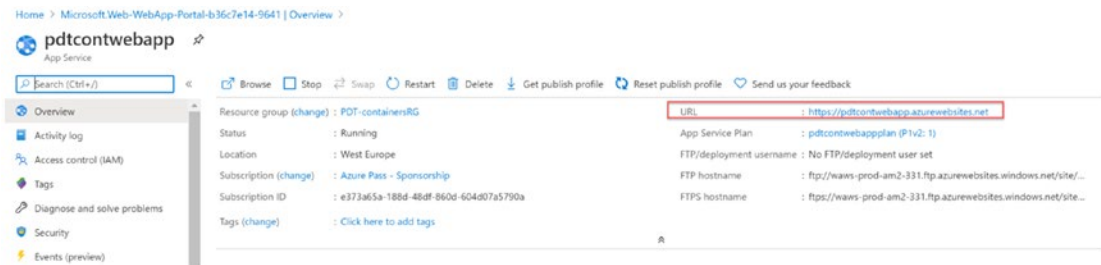
Docker hub options

Access Type *

Image and tag *

Startup Command ⓘ

8. **Confirm the creation** by clicking the **Review + create** button.
9. **Click the Create** button to start the deployment of the Azure Web App for Containers.
10. **Follow up** on the deployment from the Notifications area.
11. **Once deployed**, browse to the **[suffix]contwebapp Azure resource**, which opens the detailed blade.



12. **Click the URL** which opens your default Internet browser. The containerized webshop workload should be up and running once more. 😊

13. **Go back** to the Azure Portal, which still has your Azure Web App for Containers open; here, **browse to Settings ► Container settings** and **look at the Logs section**. This shows the different steps undergoing to get the container running.

14. For me, this is yet another benefit compared to Azure Container Instance, which is not giving you the same level of detail on what's happening with the container during the creation of the web app itself, or at least not this easy.

This completes this task, in which you got introduced to Azure Web App for Containers.

Summary

In this lab, you learned about installing Docker Enterprise for Windows Server. Next, you learned the basics of running Linux-based Docker images and containers, followed by executing several Docker commands that are common when operating Docker images and containers, as well as how Visual Studio Code extension for Docker could help you as well.

In the following tasks, you pushed the Docker container to Azure Container Registry and deployed a container instance running the image. You also learned how to deploy Azure Web App for Containers, validating each process was working fine and offering a running e-commerce platform.

CHAPTER 8

Lab 6: Deploying and Running Azure Kubernetes Service (AKS)

What You Will Learn

In this lab, you will learn what it takes to deploy an Azure Kubernetes Service (AKS), create a Kubernetes YAML deploy file, and run the Docker-containerized webshop application within the AKS cluster.

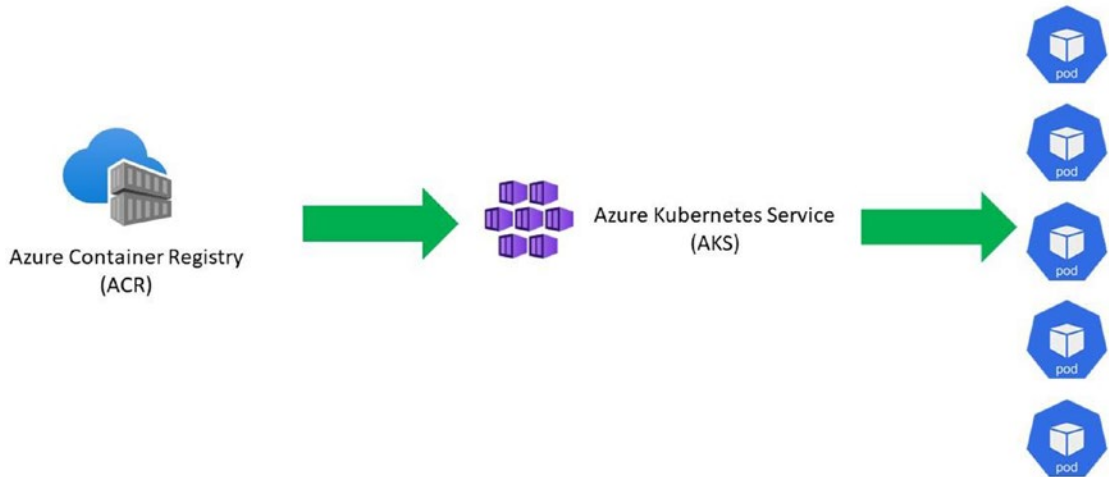
Time Estimate

This lab should take about 45 min to complete.

Prerequisites

This lab continues on the deployments from Lab 5; make sure you successfully completed that lab before starting with this one.

Scenario Diagram



Task 1: Deploying Azure Kubernetes Service using Azure CLI 2.0

Note AKS deployment is working awesome from the Azure Portal, as well as from Azure CLI. To make it easy, let's switch back to Azure Cloud Shell (Bash) and run the deployment from there.

1. **From the Azure Portal, open Azure Cloud Shell** and select **Bash**.



2. **Run the following command to** create a new Azure resource group:

```
az group create --name AKSNativeRG --location  
<yourregionofchoicehere>
```

```

Bash
imlearning@Azure:~$ az group create --name AKSNativeRG --location westeurope
{
  "id": "/subscriptions/e373a65a-188d-48df-860d-604d07a5790a/resourceGroups/AKSNativeRG",
  "location": "westeurope",
  "managedBy": null,
  "name": "AKSNativeRG",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
imlearning@Azure:~$

```

3. Next, run the following command to deploy the actual Azure Kubernetes Service resource:

```

az aks create --resource-group AKSNativeRG --name
AKScluster --node-count 2 --enable-addons
monitoring --generate-ssh-keys

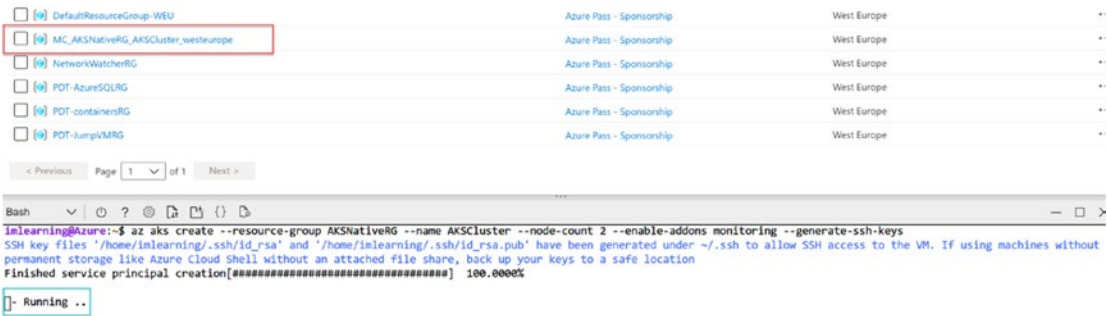
```

```

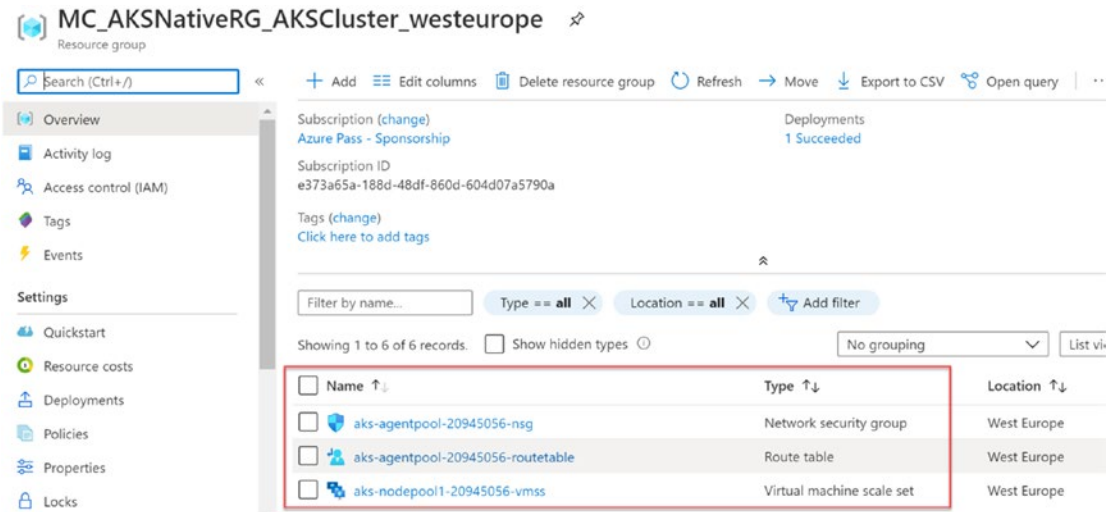
Bash
imlearning@Azure:~$ az aks create --resource-group AKSNativeRG --name AKScluster --node-count 2 --enable-addons monitoring --generate-ssh-keys
SSH key files '/home/imlearning/.ssh/id_rsa' and '/home/imlearning/.ssh/id_rsa.pub' have been generated under ~/.ssh to allow SSH access to the VM. If using machines without
permanent storage like Azure Cloud Shell without an attached file share, back up your keys to a safe location
Finished service principal creation[#####] 100.0000%

```

This command starts with creating the service principal, and moving on with the actual AKS deployment. **Note this first part of the process (after creating the service principal) is not showing any output and looks like it's hanging. But it is running fine in the background though.** After a few minutes, the status changes to **Running**, which means the actual AKS resources are getting created now. You can validate this from the **Azure Resource groups** view in the portal, where a new **RG** got created, **MC_<name of AKSRG>_<name of AKSCluster>_region**.



4. **Open this resource group**, where you can see the different Azure resources forming the Kubernetes cluster infrastructure getting created. (This might take away the magic of AKS a little bit, since technically it is a collection of traditional Azure IAAS components, like virtual machines, virtual network, load balancer, etc.)



5. After about **10 minutes**, the AKS resource has been created, **as you can notice** from the Cloud Shell window, showing you detailed JSON output with all related parameters and settings of the created service.

```

Bash
},
"loadBalancerSku": "Standard",
"networkMode": null,
"networkPlugin": "kubenet",
"networkPolicy": null,
"outboundType": "loadBalancer",
"podCidr": "10.244.0.0/16",
"serviceCidr": "10.0.0.0/16"
},
"nodeResourceGroup": "MC_AKSNativeRG_AKScluster_westeurope",
"privateFqdn": null,
"provisioningState": "Succeeded",
"resourceGroup": "AKSNativeRG",
"servicePrincipalProfile": {
  "clientId": "796c67a7-784a-46c7-a453-5075f2dc3162",
  "secret": null
},
"sku": {
  "name": "Basic",
  "tier": "Free"
},
},
"tags": null,
"type": "Microsoft.ContainerService/ManagedClusters",
"windowsProfile": null
}
imlearning@Azure:~$
imlearning@Azure:~$
imlearning@Azure:~$

```

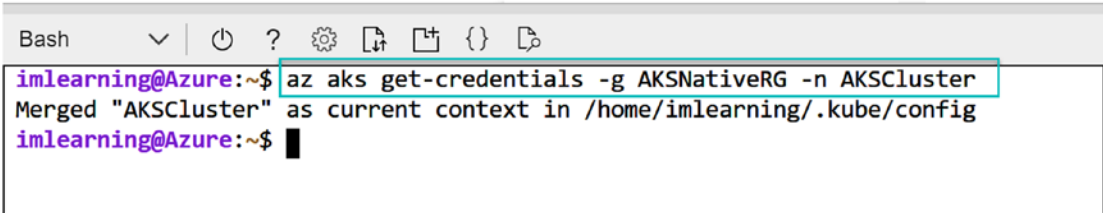
6. You can also validate this deployment from the Azure Portal, by browsing to **your Kubernetes Service**.

The screenshot shows the Azure Portal interface for the resource group 'AKSNativeRG'. The left sidebar contains navigation options like Overview, Activity log, Access control (IAM), Tags, Events, Settings, Quickstart, Resource costs, Deployments, and Policies. The main content area shows details for the 'AKSCluster' resource, including its subscription, ID, and tags. Below the details, there is a table with one record for the 'AKSCluster' resource, showing its type as 'Kubernetes service' and its location as 'West Europe'.

| Name | Type | Location |
|------------|--------------------|-------------|
| AKSCluster | Kubernetes service | West Europe |

- 7. Now that you have the Kubernetes cluster up and running, lets start with **connecting to the Kubernetes environment and validating** it is running ok, by **performing the following steps:**

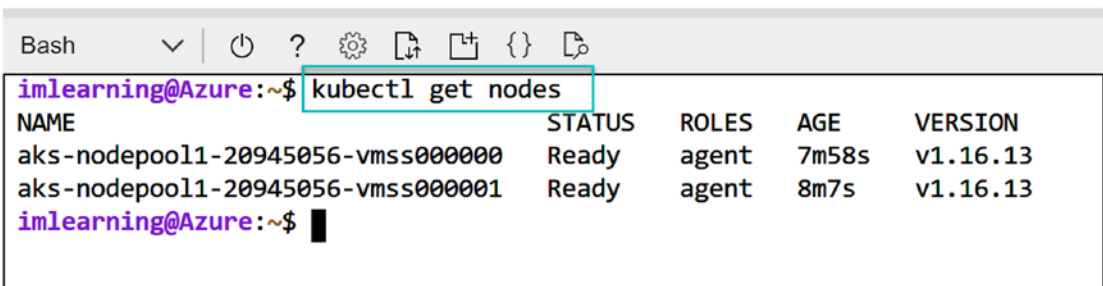
```
az aks get-credentials -g AKSNativeRG -n AKSCluster
```



(Notice how you got introduced to the shorter naming convention of Azure CLI parameters, -g instead of -resourcegroup or -n instead of -name. 😊)

- 8. Next, validate the functioning by checking the nodes, using **kubectl**. kubectl (Kube Control) is the command-line management and operations tool for Kubernetes and already integrated in Cloud Shell; if you want to manage your AKS cluster from your local machine, you need to install this kubectl tool first, following the guidelines in <https://kubernetes.io/docs/tasks/tools/install-kubectl/>:

```
kubectl get nodes
```



- As you can see here, we have two nodes running, identified with `vmss000000` and `vmss000001`; this is the default name for Azure Virtual Machine scale sets. This immediately tells you AKS is ready for scale. I'll guide you through how to do that in a later task.

This completes the task in which you deployed Azure Kubernetes Service using Azure Cloud Shell. In the next task, you learn how to integrate with Azure Container Registry, picking up your container image to have your containerized workload running in Kubernetes POD, which is the terminology for a running container in Kubernetes or a collection of containers.

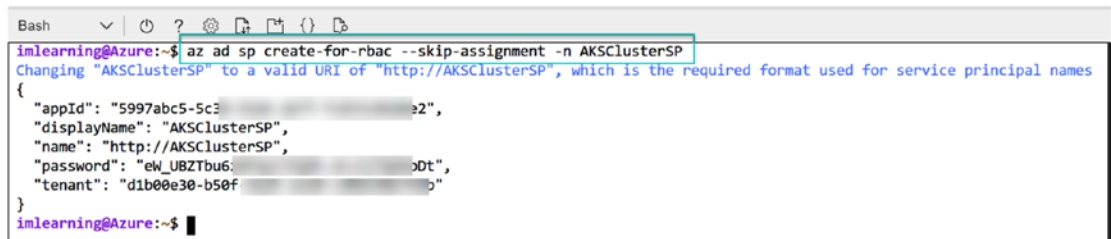
Task 2: Configuring RBAC for managing Azure Kubernetes Service and ACR integration

In the previous step, you deployed the AKS infrastructure and the AKS as a Service resource in Azure. Using the `kubectl get nodes`, you validated the underlying Kubernetes infrastructure is up and running.

Before we can have Kubernetes picking up Docker images from the Azure Container Registry you deployed earlier, we need to define Azure RBAC (Azure Role-Based Access Control) permissions for the Kubernetes resource to allow this. You need to create a service principal object in Azure Active Directory for this, which reflects an identity object for the AKS cluster.

- Create the service principal** as follows, from within your **Azure Cloud Shell** window:

```
az ad sp create-for-rbac --skip-assignment -n
AKSClusterSP
```

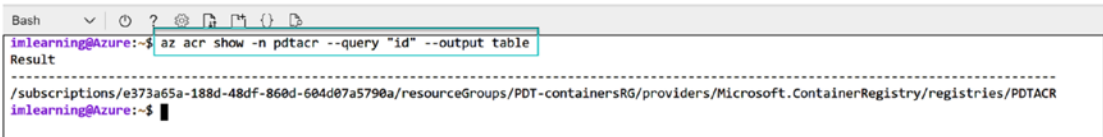


```
Bash
imlearning@Azure:~$ az ad sp create-for-rbac --skip-assignment -n AKSClusterSP
Changing "AKSClusterSP" to a valid URI of "http://AKSClusterSP", which is the required format used for service principal names
{
  "appId": "5997abc5-5c3e-4302-9000-000000000000",
  "displayName": "AKSClusterSP",
  "name": "http://AKSClusterSP",
  "password": "eW_UBZTbu6-00000000000000000000000000000000",
  "tenant": "d1b00e30-b50f-4000-9000-000000000000"
}
imlearning@Azure:~$
```

Since we need parts of this information later on, it might be good to **copy this to a Notepad doc** for easy retrieval.

2. This command creates an application ID and provides display name and tenant information that you'll need later on in the Kubernetes YAML file (similar to the Dockerfile we used earlier, but for Kubernetes deployments).
3. Next item information we need is the **full Azure resource ID for our Azure Container Registry**. This information **can be retrieved** using the following command:

```
az acr show --name [SUFFIX]ACR --query "id" --output table
```



```
Bash
imlearning@Azure:~$ az acr show -n pdtacr --query "id" --output table
Result
-----
/subscriptions/e373a65a-188d-48df-868d-684d07a5790a/resourceGroups/PDT-containersRG/providers/Microsoft.ContainerRegistry/registries/PDTACR
imlearning@Azure:~$
```

Copy this information into your Notepad doc as well, since you'll need this information later on.

4. Next, **assign the contributor role** for the previously created “appid” service principal object to this Azure Container Registry resource, by executing the following command:

```
az role assignment create --assignee "appid" --scope "ACRid" --role contributor
```

This maps like this in my environment (replaced some characters for security reasons):

```
az role assignment create --assignee "ae0ad426-af05-4a6a-0000-00000000" --scope "/subscriptions/0a407898-c077-0000-0000-714200000000/resourceGroups/ADS-dockerRG/providers/Microsoft.ContainerRegistry/registries/ADSACR" --role contributor
```

```
Bash
imlearning@Azure:~$ az role assignment create --assignee 5997abc5-5c38-4e11-8000-000000000002 --scope "/subscriptions/e373a65a-188d-48df-9122-000000000000/resourceGroups/PDT-containersRG/providers/Microsoft.ContainerRegistry/registries/PDTACR" --role contributor
{
  "canDelegate": null,
  "id": "/subscriptions/e373a65a-188d-48df-9122-000000000000/resourceGroups/PDT-containersRG/providers/Microsoft.ContainerRegistry/registries/PDTACR/providers/Microsoft.Authorization/roleAssignments/f6cd359c-c812-4608-8000-000000000000",
  "name": "f6cd359c-c812-4608-8000-000000000000",
  "principalId": "655ec812-4608-8000-0000-000000000000",
  "principalType": "ServicePrincipal",
  "resourceGroup": "PDT-containersRG",
  "roleDefinitionId": "/subscriptions/e373a65a-188d-48df-9122-000000000000/providers/Microsoft.Authorization/roleDefinitions/b24988ac-6180-42a0-ab88-20f7382dd24c",
  "scope": "/subscriptions/e373a65a-188d-48df-9122-000000000000/resourceGroups/PDT-containersRG/providers/Microsoft.ContainerRegistry/registries/PDTACR",
  "type": "Microsoft.Authorization/roleAssignments"
}
imlearning@Azure:~$
```

- We also will instruct **kubect1** (the Kubernetes cluster actually, by using kubect1) to use a secret, which will be used to get access to the Azure Container Registry, using the following command:

```
kubect1 create secret docker-registry acr-auth --docker-server <yourACR>.azurecr.io --docker-username 6956b3da-0000000 (Appid here) --docker-password a90497d6-69ea-000000 <app password here> --docker-email <your email address here>
```

```
Bash
imlearning@Azure:~$ kubect1 create secret docker-registry acr-auth --docker-server pdtacr.azurecr.io --docker-username 5997abc5-5c38-4e11-8000-000000000002 --docker-password eW_U8Z7bu6z --docker-email imlearning@outlook.com
secret/acr-auth created
imlearning@Azure:~$
```

Here is some explanation of the command information:

- **kubect1 create secret:** The command to create a secret.
- **docker-registry:** Secret is of type “docker registry.”
- **acr-auth:** A name you allocate to this secret.
- **docker-server:** Azure Container Registry is a docker-compatible registry.
- **docker-username:** Identifies the service principal object that has permissions.

- **docker-password:** Identifies the password of the service principal object.
- **docker-email:** The email account, which could be a Docker account, but I'm using the Azure admin account email here.

With all the back-end information and the RBAC service principal and permissions in place, we **can build our YAML deployment file for Kubernetes**. Key information in here is the name of your Azure Container Registry, the container image filename that you want to push to the Kubernetes cluster, and what port the container should run on, as well as specifying what kubectl credentials you want to use.

This will be performed in the next task.

Task 3: Running a Docker container image from Azure Container Registry in Azure Kubernetes Service

1. On the **lab jumpVM**, open **Visual Studio Code**. Browse to the source folder you used before, open the **Kubernetes subfolder**, and check for a file **kubernetes.yml**.

The content looks similar to this:

```
1  apiVersion: apps/v1beta1
2  kind: Deployment
3  metadata:
4  |   name: anothercontapp2
5  spec:
6  |   replicas: 5
7  |   strategy:
8  |     rollingUpdate:
9  |       maxSurge: 1
10 |       maxUnavailable: 1
11 |   minReadySeconds: 5
12 |   template:
13 |     metadata:
14 |       labels:
15 |         app: anothercontapp2
16 |     spec:
17 |       containers:
18 |         - name: anothercontapp2
19 |           image: pdtsimplacr.azurecr.io/simplcommerce:latest
20 |           ports:
21 |             - containerPort: 80
22 |           imagePullSecrets:
23 |             - name: acr-auth
24
25
26  ---
27  apiVersion: v1
28  kind: Service
29  metadata:
30 |   name: anothercontapp2
31  spec:
32 |   type: LoadBalancer
33 |   ports:
34 |     - port: 80
35 |   selector:
36 |     app: anothercontapp2
```

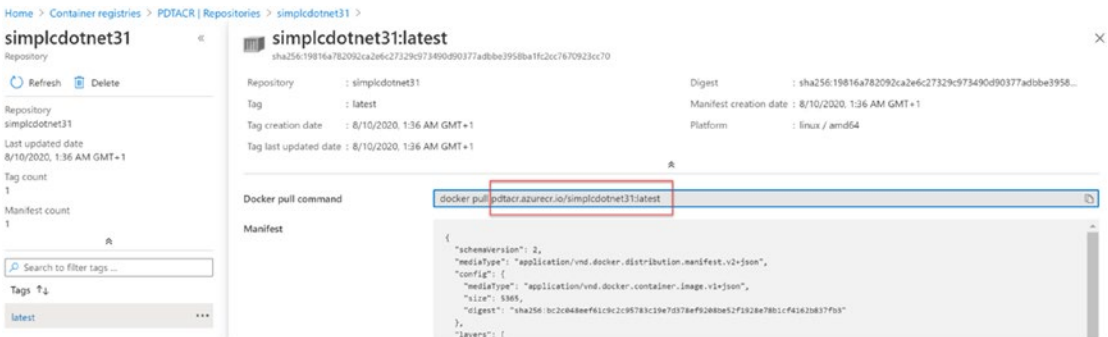
2. Note several parameters that are important for a successful deployment:

- **name: anothercontapp2** (this is just a random name you can decide on for the POD in AKS).
- **replicas: 5** (this defines how many instances of this container image we want to run within the AKS cluster).
- **image: pdtsimplacr...** points to the Azure Container Registry and Docker images we pushed earlier (and the same one we ran in Azure Container Instance).
- **port: 80** (specifies what port the app container should run on).
- **imagepullSecrets name:** The name of the RBAC contributor.

3. **Replace the** following sample parameters in this kubernetes.yml file with the actual values of your running environment:

- **name: firstsample (replace this consistently for all “name” parameter settings)**
- **image: [suffix]acr.azurecr.io/simplcdotnet31:latest**

(To find the correct image URL, go to the Azure Container Registry resource ► Repositories, select the pushed container image, and select latest.) You can grab the full URL from the Docker command option here.

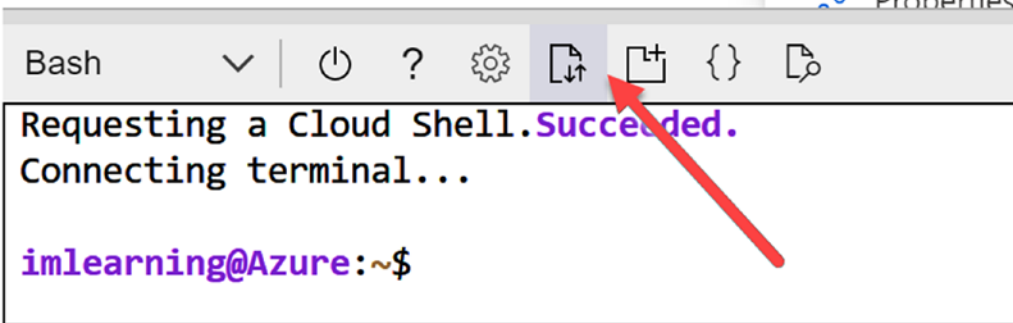


Note A full-production YAML file for Kubernetes is probably looking more complex than this, but this is the baseline you need to see it in action.

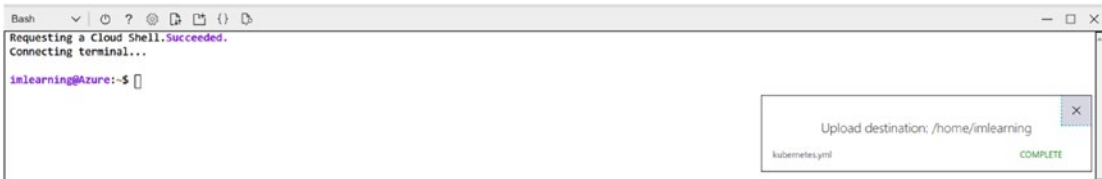
4. The updated `kubernetes.yml` file should now look similar to this (for my environment):

```
1  apiVersion: apps/v1beta1
2  kind: Deployment
3  metadata:
4    name: firstsample
5  spec:
6    replicas: 5
7    strategy:
8      rollingUpdate:
9        maxSurge: 1
10       maxUnavailable: 1
11    minReadySeconds: 5
12    template:
13      metadata:
14        labels:
15          app: firstsample
16      spec:
17        containers:
18          - name: firstsample
19            image: pdtacr.azurecr.io/simplcdotnet31:latest
20            ports:
21              - containerPort: 80
22            imagePullSecrets:
23              - name: acr-auth
24
25  ---
26  apiVersion: v1
27  kind: Service
28  metadata:
29    name: firstsample
30  spec:
31    type: LoadBalancer
32    ports:
33      - port: 80
34    selector:
35      app: firstsample
36
```

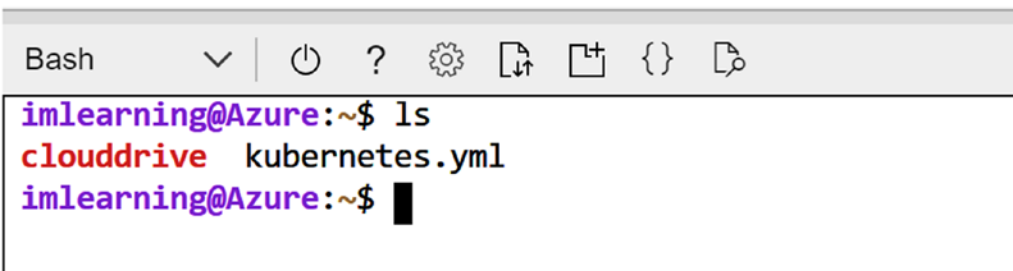

5. **Save** the updated file.
6. As this file was edited on the local JumpVM, but we are running the AKS cluster operations from within Azure Cloud Shell, you need to **upload this file** first. From the Azure Cloud Shell window in the browser, **select the “Upload/Download files” icon**.



7. **Browse** to the kubernetes.yml file on the JumpVM disk. c:\2tierazuremigration\kubernetes\kubernetes.yml is the default location.
8. **Wait for the upload to complete.**

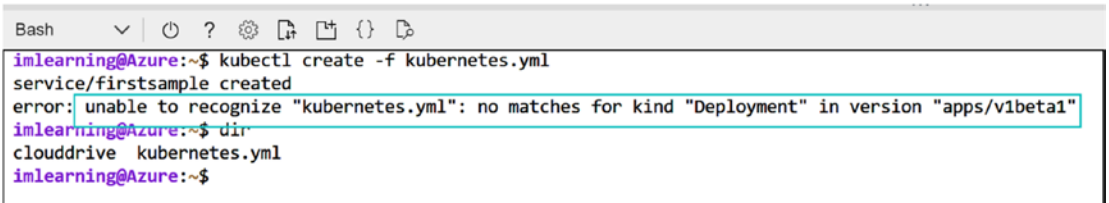


9. **Running “dir” or “ls”** in the Cloud Shell to get a list of items shows a successful upload.



10. Next, **run the deployment of this Kubernetes Service, by using the following command:**

```
kubectl create -f Kubernetes.yml
```

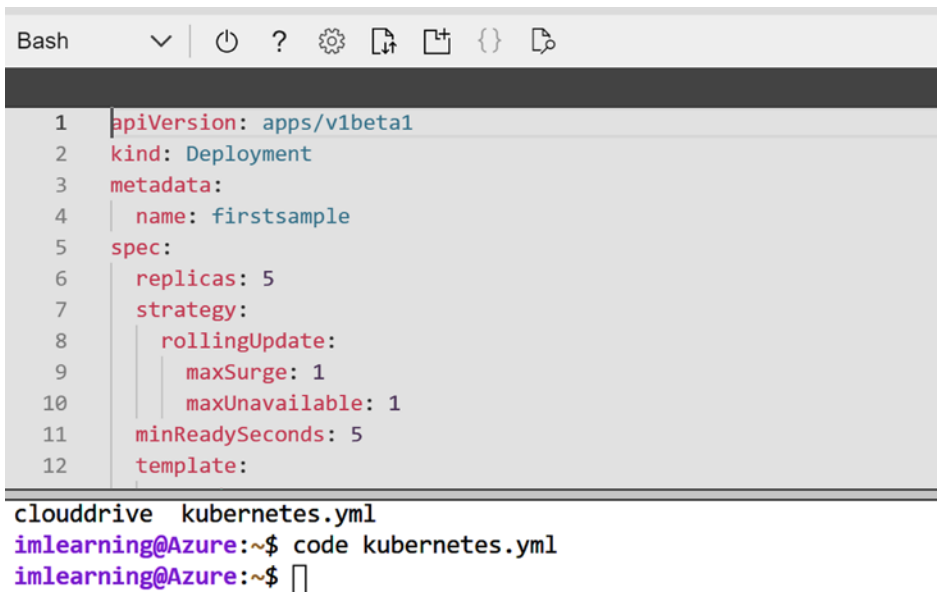


```
Bash
imlearning@Azure:~$ kubectl create -f kubernetes.yml
service/firstsample created
error: unable to recognize "kubernetes.yml": no matches for kind "Deployment" in version "apps/v1beta1"
imlearning@Azure:~$ dir
cloudrive kubernetes.yml
imlearning@Azure:~$
```

11. As you can see, this **throws an error message**, related to the version of the deployment being used. This means we need to update our kubernetes.yml file once again. Instead of going back to the JumpVM Visual Studio Code and uploading the file again to Cloud Shell, let me introduce you to some “cloud magic” 😊, running VS Code directly from within Azure Cloud Shell.
12. Run the following command in Cloud Shell:

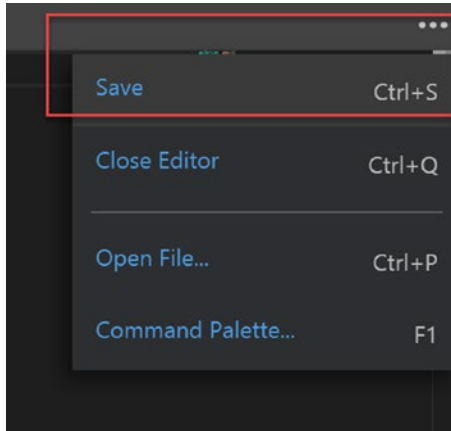
```
code Kubernetes.yml
```

This directly opens VS Code from within the shell itself! How nice!



```
Bash
1  apiVersion: apps/v1beta1
2  kind: Deployment
3  metadata:
4    name: firstsample
5  spec:
6    replicas: 5
7    strategy:
8      rollingUpdate:
9        maxSurge: 1
10       maxUnavailable: 1
11   minReadySeconds: 5
12   template:
cloudrive kubernetes.yml
imlearning@Azure:~$ code kubernetes.yml
imlearning@Azure:~$
```

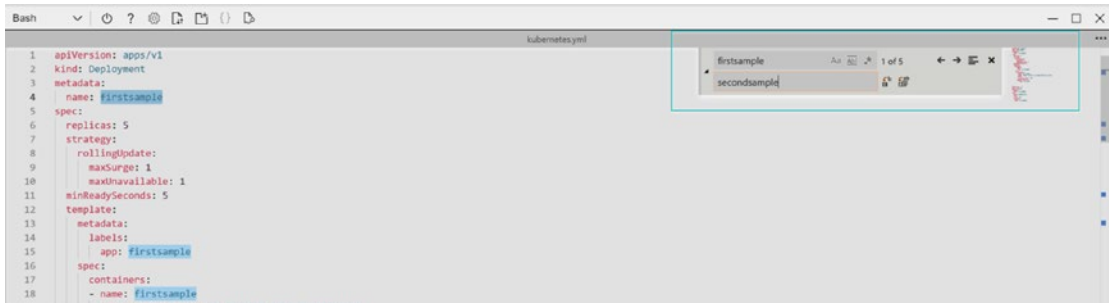
- 13. **Update** the parameter **apiVersion** to “apps/v1.”
- 14. **Once edited**, click the **ellipsis (three dots)** in the right-hand corner of Cloud Shell, and select **Save** (or press Ctrl-S).



- 15. **Before we can initiate a new deployment**, we need to make another update to this YAML file, that is, the name of the deployment. Although the earlier deployment failed, it is registered as a deployment in Kubernetes. Running this deployment again will throw another error, saying the name is already in use.

Therefore, replace the “firstsample” name to “secondsample” (in all locations).

The easiest way to do this is through **Find/Replace**; press **Ctrl-H**, which opens up the Find/Replace popup (similar to your local running instance of VS Code, but all done from within the Cloud Shell – yes, this is an almost full running instance in the browser!).



16. **Save the changes** from the Find/Replace, and once more **save the file**.
17. **Close the VS Code instance by pressing Ctrl-Q or selecting the ellipsis and choosing Close Editor**.
18. **Initiate a new deployment**, by running **kubectl create -f Kubernetes.yml** again; notice this time, the deployment succeeds.

```
Bash
imlearning@Azure:~$ kubectl create -f Kubernetes.yml
deployment.apps/secsample created
service/secsample created
imlearning@Azure:~$
```

19. While this confirms a successful “deployment” task, it doesn’t mean the containerized workload is already up and running. But you can follow/validate this process, running some other **kubectl** commands:

```
kubectl get pods
```

```
Bash
imlearning@Azure:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
secondsample-6dcc99ff4f-7cqvd      1/1     Running   0           11m
secondsample-6dcc99ff4f-87xch      1/1     Running   0           11m
secondsample-6dcc99ff4f-9zrkc      1/1     Running   0           11m
secondsample-6dcc99ff4f-ctwnk      1/1     Running   0           11m
secondsample-6dcc99ff4f-rpr7m      1/1     Running   0           11m
secsample-6fbfc856cf-8vgpb          1/1     Running   0           4m45s
secsample-6fbfc856cf-fwkk5         1/1     Running   0           4m45s
secsample-6fbfc856cf-jb57k         1/1     Running   0           4m45s
secsample-6fbfc856cf-m5bf8         1/1     Running   0           4m45s
secsample-6fbfc856cf-t4r4f         1/1     Running   0           4m45s
imlearning@Azure:~$
```

The reason why it shows five running PODs here is because we defined the **replicas** parameter in the YAML file (I'll drill down on this high availability/scalability in Chapter 9).

Note If you should see an error message here, it is most probably related to not having defined the ACR authentication correctly.

20. One can also **check the actual container services**, by running the following command

```
kubectl get services
```

```
Bash
imlearning@Azure:~$ kubectl get services
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
firstsample   LoadBalancer 10.0.179.30    20.50.161.123  80:30227/TCP     45m
kubernetes    ClusterIP     10.0.0.1       <none>         443/TCP          132m
secondsample  LoadBalancer 10.0.132.152   20.50.215.134  80:32405/TCP     31m
secsample     LoadBalancer 10.0.251.121   52.236.157.25  80:32477/TCP     5m26s
imlearning@Azure:~$
```

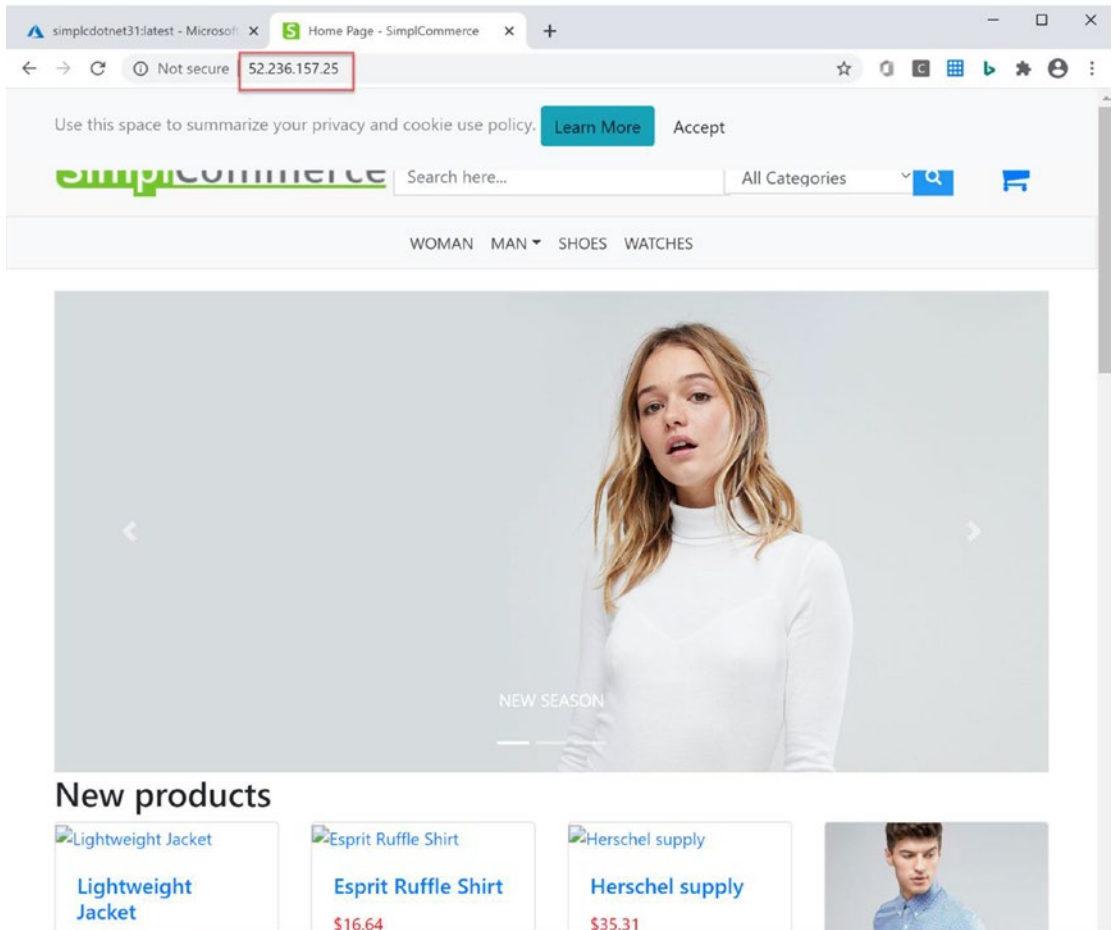
or checking for more details for a specific running service:

```
kubectl get service --watch
```

```
Bash
imlearning@Azure:~$ kubectl get services secsample --watch
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
secsample     LoadBalancer 10.0.251.121   52.236.157.25  80:32477/TCP     7m3s
```

21. **Wait for the service to receive an external IP** address, which would mean the POD is fully up and running in AKS. From here, you could open your browser and connect to the public IP address, revealing the e-commerce sample application!

Note This can take another few minutes before the app is actually fully loaded, no panic if it is not showing up immediately!



This confirms that our AKS service is fully operational, and the Docker container image that we pushed from the YAML file settings is also working correctly. Nice job! This completes the lab.

Summary

In this lab, you learned how to deploy Azure Kubernetes Service (AKS) using Azure CLI, as well as how to manage and validate the deployment using kubectl Kubernetes command line. Next, you configured RBAC and ACR authentication for a service principal. This was followed by the creation of a kubernetes.yml deployment file, having a pointer to the Azure Container Registry repository image to use. After deploying this container image within the AKS cluster, you validated the functioning using the EXTERNAL-IP of the AKS service and checked the PODs.

CHAPTER 9

Lab 7: Managing and Monitoring Azure Kubernetes Service (AKS)

What You Will Learn

In this next lab of this workshop, we will focus on common operations related to AKS. This includes enabling the basics of container scalability within the platform, as well as configuring the new Azure built-in monitoring capabilities for these services, using Azure Monitor for Azure Kubernetes Service.

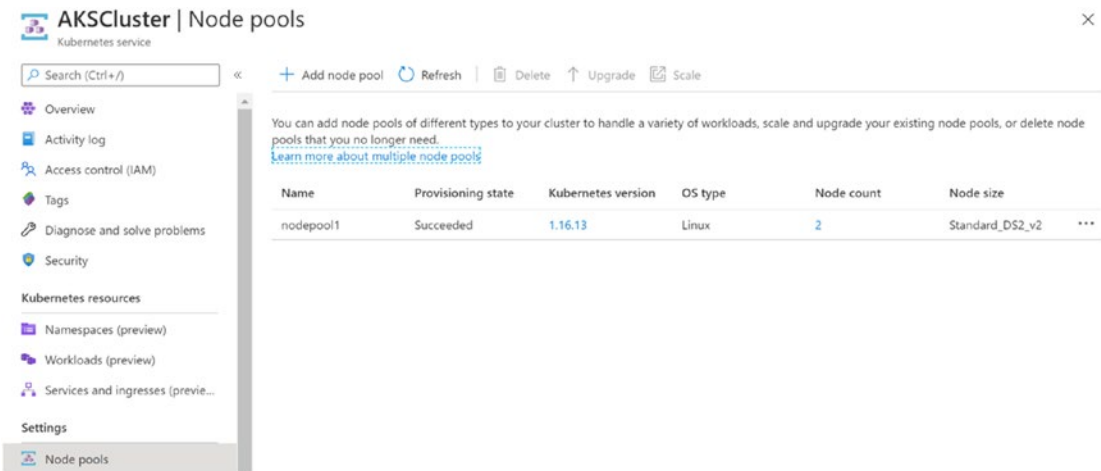
Time Estimate

This lab shouldn't take longer than 60 min.

Note Depending on the subscription type you are using (e.g., Azure Trial, Azure Pass, etc.), you might be limited in the number of cores still available for performing the scale operations discussed in this task. If you are OK with it, you could delete the WebVM and SQLVM virtual machines to free up cores.

Task 1: Enabling container scalability in Azure Kubernetes Service (AKS)

1. AKS provides some nice integration in the Azure Portal, for example, on how to **scale out** your Kubernetes Service. **From the Azure Portal**, browse to your **Azure Kubernetes Service**. In the detailed blade, **go to Settings ► Node pools**.



2. Here, you can “scale” in two different ways, **extending the amount of nodes in the existing pool** or **adding a new pool**. You will configure both, starting with adding additional nodes to an already existing pool. To do **this**, **click the number in the Node count column (2)**. This opens the “Scale” blade.

Scale



Scale method ⓘ

Manual
 Autoscale

Node count

x Standard DS2 v2 (2 vcpus, 7 GiB memory)

Total cluster capacity

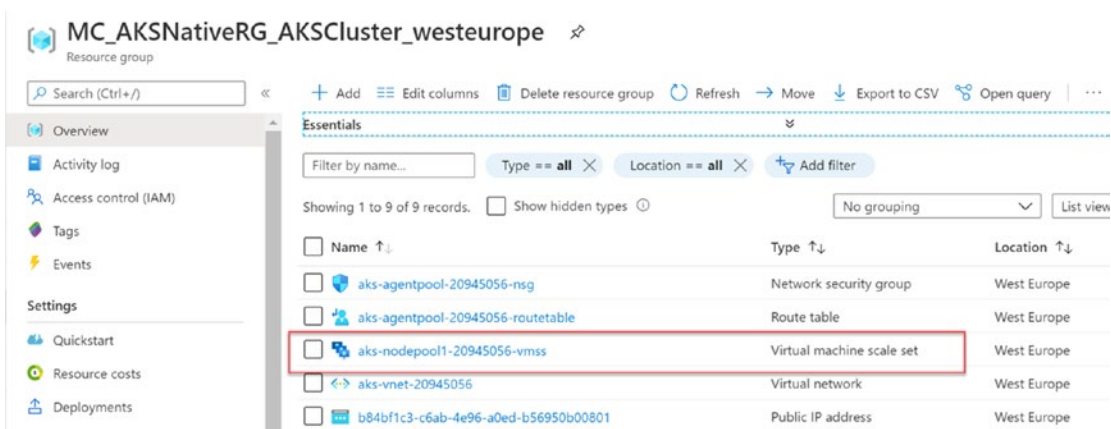
Cores 4 vCPUs

Memory 14 GiB

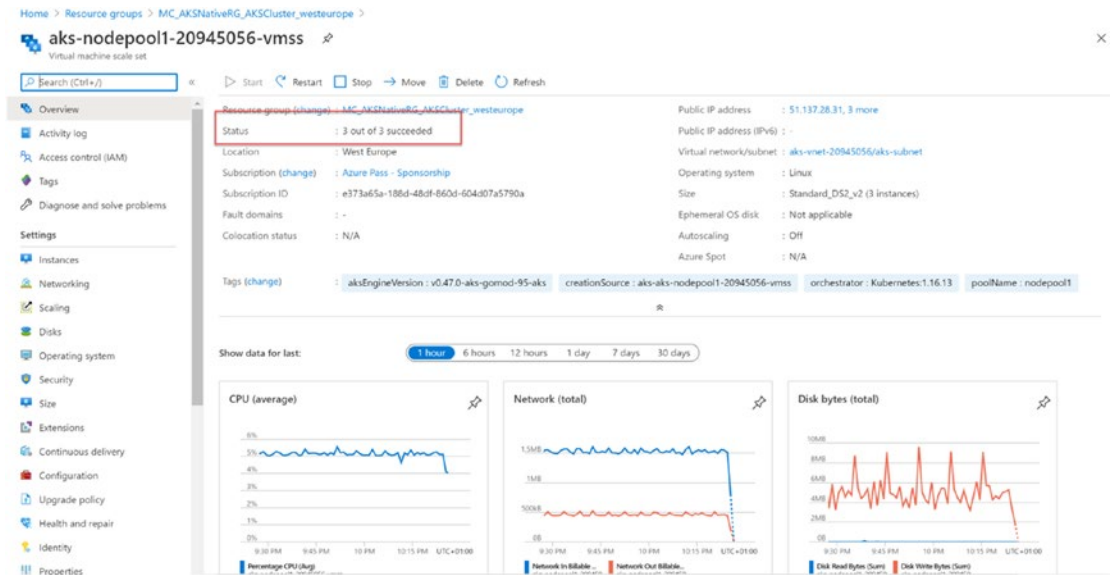
3. **Change the node count to “3,”** leaving the **Scale method as Manual**. Save the changes by **clicking Apply**. Wait for the changes to apply, **and validate by refreshing the blade**.

| Name | Provisioning state | Kubernetes version | OS type | Node count | Node size | |
|-----------|--------------------|--------------------|---------|------------|-----------------|-----|
| nodepool1 | Scaling | 1.16.13 | Linux | 3 | Standard_DS2_v2 | ... |

4. **From the Azure Portal**, browse to the resource group holding the Azure resources for Azure Kubernetes Service, identified as MC_AKSNativeRG_AKScluster_<region>; **here, select the “Virtual machine scale set”**



5. This redirects you to the individual scale set for the AKS Node Pool.



6. Notice how it identifies 3 out of 3 succeeded as Status; next, select Instances within the Settings pane.

aks-nodepool1-20945056-vmss | Instances
Virtual machine scale set

Search (Ctrl+/) << Start Restart Stop Reimage Delete

Search virtual machine instances

| Name | Status |
|--|-----------|
| <input type="checkbox"/> aks-nodepool1-20945056-vmss_0 | ✔ Running |
| <input type="checkbox"/> aks-nodepool1-20945056-vmss_1 | ✔ Running |
| <input type="checkbox"/> aks-nodepool1-20945056-vmss_2 | ✔ Running |

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems

Settings
Instances

- This shows** the three nodes running. Clicking any of these would show more details about the running instance, but mainly from an Azure infrastructure perspective, not from a Kubernetes perspective. More on that later...
- So now that you know how to extend the number of nodes in your cluster, let me show you the same, but using **kubectl command line**, again from **Azure Cloud Shell (Bash)**:

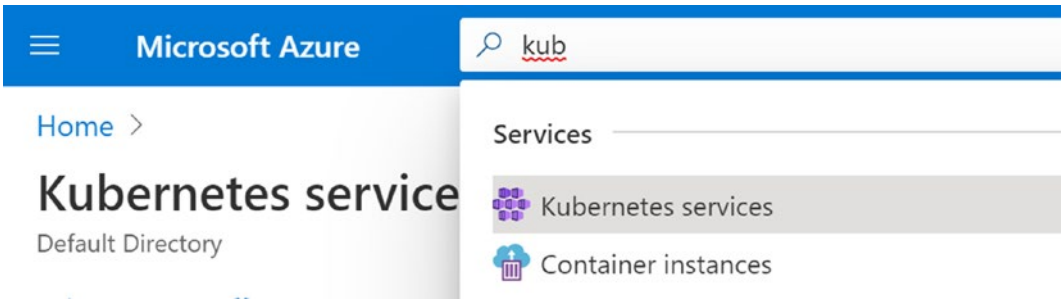
```
az aks scale -g AKSNativeRG -n AKSCluster --node-count 4
```

```
imlearning@Azure:~$ az aks scale -g AKSNativeRG -n AKSCluster --node-count 4
[- Running ..
```

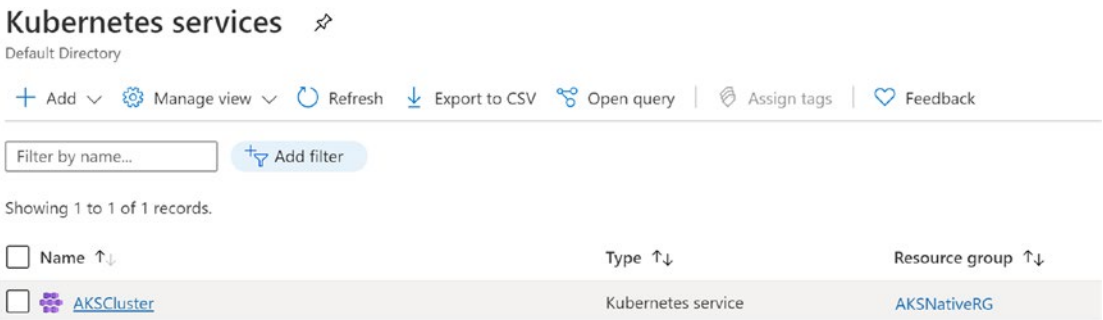
Note The command takes a couple of minutes to complete, without having impact on the already running PODs. The result is published in the JSON output.

```
Bash
(aks) ...
imlearning@azure:~$ az aks scale -g AKSNativeRG -n AKScluster --node-count 4
[- Finished ..]
{
  "aadProfile": null,
  "addonProfiles": {
    "KubeDashboard": {
      "config": null,
      "enabled": true,
      "identity": null
    },
    "omsagent": {
      "config": {
        "logAnalyticsWorkspaceResourceID": "/subscriptions/e373a65a-188d-48df-860d-604d07a5790a/resourcegroups/defaultresourcegroup-weu/providers/microsoft.operationalinsights/workspaces/defaultworkspace-e373a65a-188d-48df-860d-604d07a5790a-weu"
      },
      "enabled": true,
      "identity": null
    }
  }
}
```

9. Let’s switch back to the Azure Portal view and “scale” the AKS environment by **adding an additional node pool**. Switch back to your AKS cluster, by searching for “Kubernetes” in the Search resources, services, and docs (G+).



10. Select your cluster in the list of Kubernetes services.



11. **From the AKS cluster details, select Node pools under Settings; here, click “Add node pool.”**

You can add node pools of different types to your cluster to handle a variety of workloads, scale and upgrade your existing node pools, or delete node pools that you no longer need. [Learn more about multiple node pools](#)

| Name | Provisioning state | Kubernetes version | OS type | Node count | Node size | |
|-----------|--------------------|--------------------|---------|------------|-----------------|-----|
| nodepool1 | Succeeded | 1.16.13 | Linux | 4 | Standard_DS2_v2 | ... |

12. Complete the configuration of the new node pool, using the following parameters:

Add a node pool

Node pool name * ⓘ

OS type * ⓘ Linux Windows

i Windows node pools require a Windows authentication profile

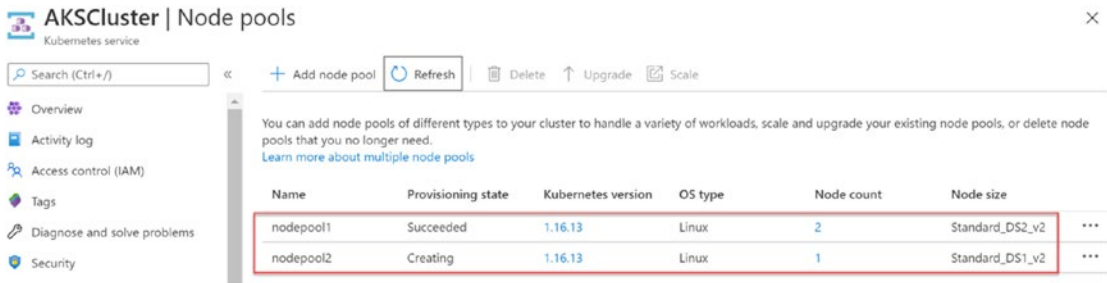
Kubernetes version * ⓘ

Node size * ⓘ **Standard DS1 v2**
1 vcpu, 3 GiB memory
[Choose a size](#)

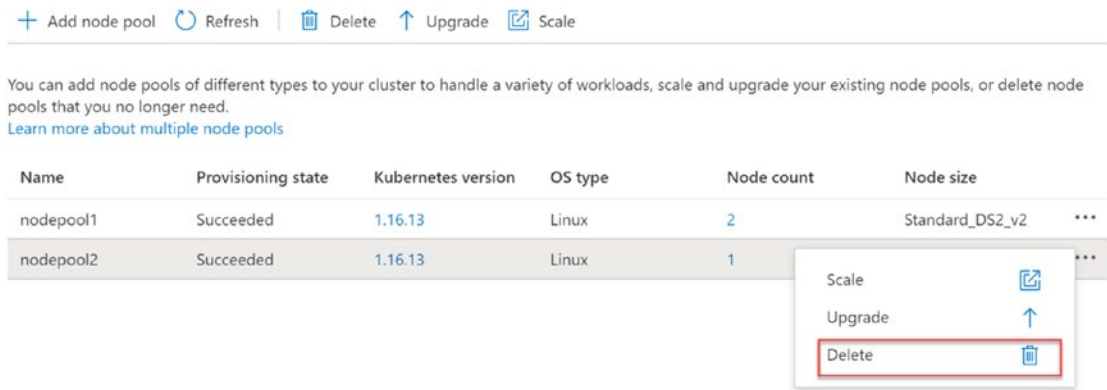
Node count * ⓘ

- **Node pool name:** A name of your choice.
- **OS type:** Linux.
- **Kubernetes version:** Leave default (know this doesn't need to be the same).
- **Node size:** Choose a size/DS1 v2.
- **Node count:** 1.

- 13. You are back in the Node pools blade, where you can see the second node pool getting created (you might need a refresh to speed this up).



- 14. From here, you would technically repeat the same process as earlier if you want to extend the number of nodes in this second pool. To free up some resources, **let's delete this second node pool again.**



- 15. Since adding pools is creating “separate” virtual machine scale set environments in Azure, it might not always be what you are looking for in terms of scaling. What if you want to run a larger amount of **identical instances**, but maxing out the capacity of your node pool? This is done using the kubernetes.YAML file (and something you actually already did). By scaling the actual number of PODs, using another update in the previously configured

kubernetes.yml file, you can identify how many identical instances you want to run.

This is done by running the following command:

```
kubectl scale --replicas=3 -f <path to yml file>
```

Which in this scenario scales down the number of replicas from five to three (remember we defined five replicas in the YAML file initially).

16. You can validate the operation using **kubectl get PODs** and **kubectl get services --watch**.

```
imlearning@Azure:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
secondsample-6dcc99ff4f-7cqvd      1/1     Running  0           7h52m
secondsample-6dcc99ff4f-87xch      1/1     Running  0           7h52m
secondsample-6dcc99ff4f-9zrkc      1/1     Running  0           7h52m
secondsample-6dcc99ff4f-ctwnk      1/1     Running  0           7h52m
secondsample-6dcc99ff4f-rpr7m      1/1     Running  0           7h52m
secsample-6fbfc856cf-8vgpb         1/1     Running  0           7h45m
secsample-6fbfc856cf-m5bf8         1/1     Running  0           7h45m
secsample-6fbfc856cf-t4r4f         1/1     Running  0           7h45m
imlearning@Azure:~$
```

```
imlearning@Azure:~$ kubectl get services
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
firstsample   LoadBalancer 10.0.179.30   20.50.161.123 80:30227/TCP     8h
kubernetes    ClusterIP      10.0.0.1      <none>         443/TCP          9h
secondsample  LoadBalancer 10.0.132.152 20.50.215.134 80:32405/TCP     8h
secsample     LoadBalancer 10.0.251.121 52.236.157.25 80:32477/TCP     7h45m
```

17. This completes the task on learning different scaling methods in AKS.

Task 2: Monitoring Azure Kubernetes Service in Azure

Azure provides a nice integration (Insights) between standard Azure monitoring capabilities and the AKS services.

1. **From the Azure Portal**, browse to **Azure Kubernetes Service**, and **select** your AKS service. **From the Overview** pane, you get a lot of important information about your AKS cluster setup, like Kubernetes version, amount of nodes/cores, and so on.

AKSCluster
Kubernetes service

Search (Ctrl+/) << Connect Delete Refresh

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Security

Kubernetes resources

- Namespaces (preview)
- Workloads (preview)
- Services and ingresses (previe...

Settings

- Node pools
- Upgrade
- Scale
- Networking
- Dev Spaces
- Deployment center (preview)
- Policies (preview)

Resource group ([change](#))
AKSNativeRG

Status
Succeeded

Location
West Europe

Subscription ([change](#))
[Azure Pass - Sponsorship](#)

Subscription ID
e373a65a-188d-48df-860d-604d07a5790a

Tags ([change](#))
[Click here to add tags](#)

Properties Capabilities

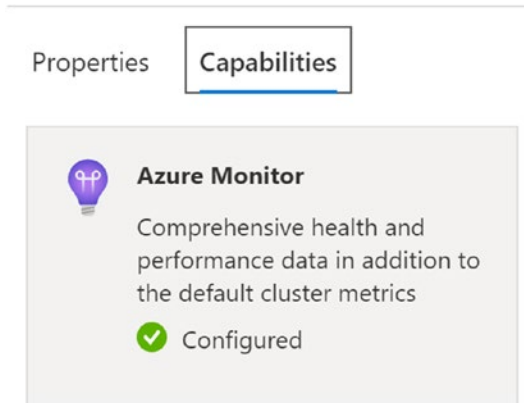
Kubernetes services

| | |
|----------------------|-------------|
| Kubernetes version | 1.16.13 |
| Azure AD integration | Not enabled |

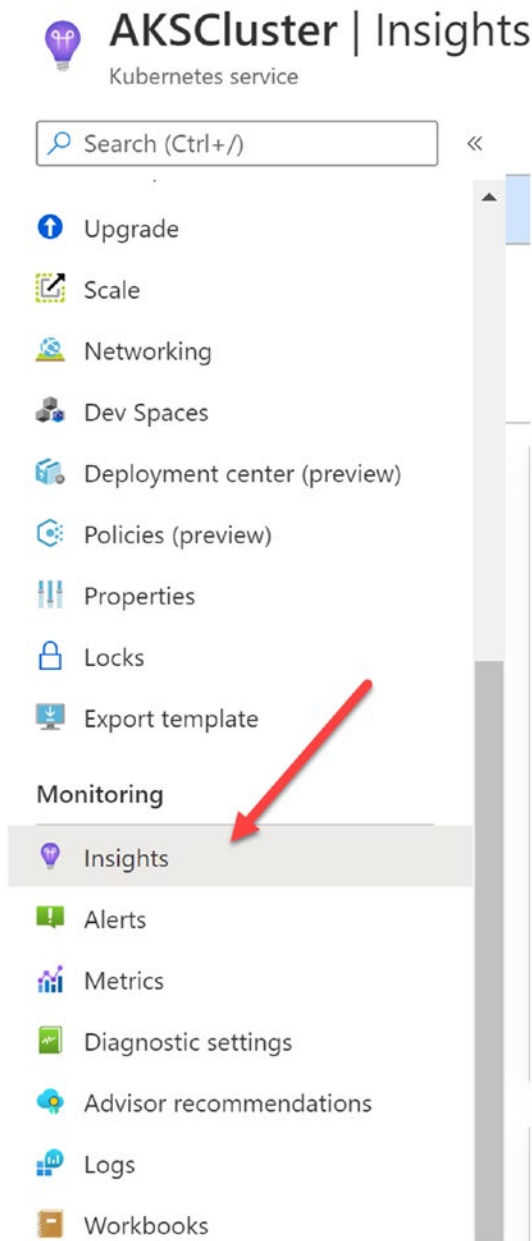
Node pools

| | |
|---------------------|-----------------|
| Node pools | 1 node pool |
| Kubernetes versions | 1.16.13 |
| Node sizes | Standard_DS2_v2 |
| Virtual node pools | Not enabled |

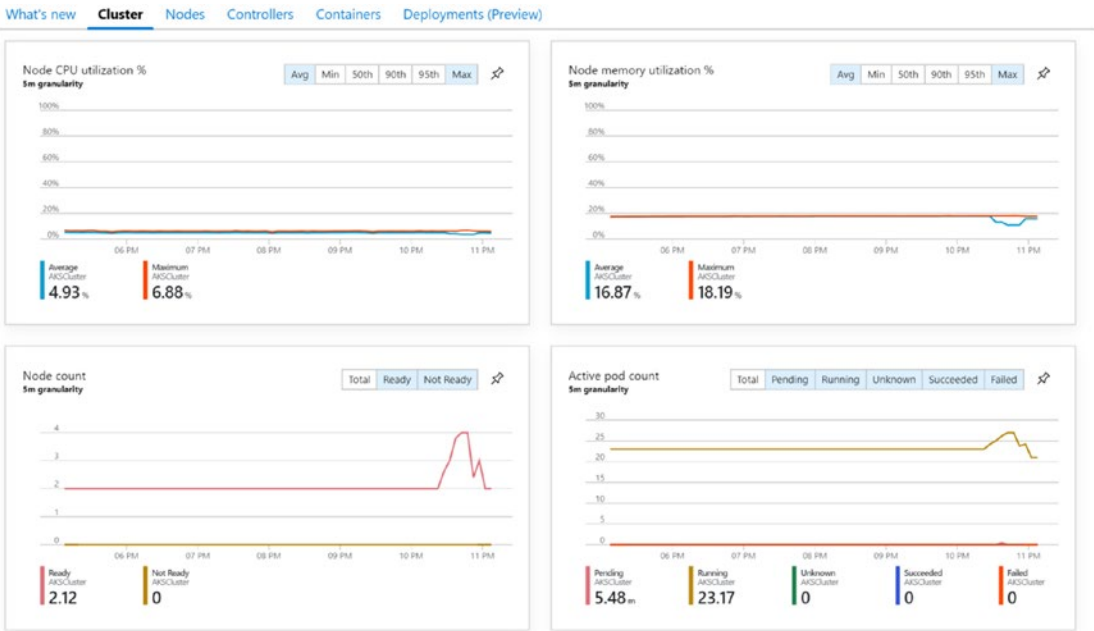
2. **Selecting the Capabilities pane next to Properties will open the detailed blade for this service. Here, select Azure Monitor.**



3. **You can reach the same by selecting Insights under the Monitoring section.**



4. **From here, you can get a more detailed view on nodes, containers, and overall system processes and performance indicators.**



5. Click “Nodes” to get a more detailed view on amount and status of nodes.

What's new Cluster **Nodes** Controllers Containers Deployments (Preview)

Search by name... Metric: CPU Usage (millicores) [v] [Min Avg 50th 90th 95th Max]

| NAME | STATUS | 95TH % | 95TH | CONTAINERS | UPTIME | CONTROLLER |
|----------------------------|--------|--------|--------|------------|---------|------------|
| ▶ aks-nodepool1-209450... | ✔ Ok | 7% | 136 mc | 14 | 9 hours | - |
| ▶ aks-nodepool1-209450... | ✔ Ok | 4% | 80 mc | 9 | 9 hours | - |
| ▶ aks-nodepool1-209450... | ✔ Ok | 3% | 56 mc | 2 | 43 mins | - |
| ▶ aks-nodepool1-209450... | ✔ Ok | 2% | 41 mc | 2 | 33 mins | - |
| ▶ aks-nodepool2-209450... | ✔ Ok | - | - | 2 | - | - |

6. Click “Containers” for a more detailed view of the running containers.

What's new Cluster Nodes Controllers **Containers** Deployments (Preview)

Search by name... Metric: CPU Usage (millicores) Min Avg 50th 90th 95th Max 29 items

| NAME | STATUS | 95TH... ↓ | 95TH | POD | NODE | RESTARTS | UPTIME | TREND 95TH % (1 BAR = 15M) |
|----------------|--------|-----------|--------|-------------------|-----------------|----------|---------|----------------------------|
| coredns | Ok | 0.2% | 4 mc | coredns-869c... | aks-nodepool... | 0 | 9 hours | |
| tunnel-probe | Ok | 0.2% | 0.2 mc | tunnelfront-6... | aks-nodepool... | 0 | 9 hours | |
| kube-proxy | Ok | 0.2% | 3 mc | kube-proxy-v... | aks-nodepool... | 0 | 9 hours | |
| kube-proxy | Ok | 0.1% | 3 mc | kube-proxy-fb... | aks-nodepool... | 0 | 33 mins | |
| metrics-server | Ok | 0.1% | 1 mc | metrics-server... | aks-nodepool... | 0 | 9 hours | |
| secondsample | Ok | 0% | 0.7 mc | secondsample... | aks-nodepool... | 0 | 7 hours | |

7. **From the list of containers**, notice there are a lot of other “system process” containers, besides the containerized application (firstsample, secondsample) you published yourself. To get a clearer view on your own application containers, add it to the **search field**.

What's new Cluster Nodes Controllers **Containers** Deployments (Preview)

second Metric: CPU Usage (millicores) Min Avg 50th 90th 95th Max

| NAME | STATUS | 95TH % | 95TH | POD | NODE | RESTARTS | UPTIME |
|--------------|--------|--------|--------|-----------------------|----------------------|----------|---------|
| secondsample | Ok | 0% | 0.7 mc | secondsample-6dccc... | aks-nodepool1-209... | 0 | 7 hours |
| secondsample | Ok | 0% | 0.7 mc | secondsample-6dccc... | aks-nodepool1-209... | 0 | 7 hours |
| secondsample | Ok | 0% | 0.6 mc | secondsample-6dccc... | aks-nodepool1-209... | 0 | 7 hours |
| secondsample | Ok | 0% | 0.7 mc | secondsample-6dccc... | aks-nodepool1-209... | 0 | 7 hours |
| secondsample | Ok | 0% | 0.6 mc | secondsample-6dccc... | aks-nodepool1-209... | 0 | 7 hours |

8. **This filters** the list of containers; **by hovering over the POD** names, you will notice that each “instance” of the secondsample replica is running in its own “POD,” where if you **hover over the node names**, part of them are running on VMSS000000, while some others are running on VMSS000001 (I couldn’t capture that little balloon popup in the screenshots).

9. While this might (should 😊) be already quite impressive, especially if you are already familiar with **Azure Monitor**, knowing you can read out all this information from within the Azure Portal, know that Microsoft is working on an even more detailed view, currently in preview. But that shouldn't stop me from showing you.

From the **AKSCluster** blade, browse to **Kubernetes resources** and select **Workloads (preview)**.

The screenshot displays the 'AKSCluster | Workloads (preview)' page in the Azure Portal. The left-hand navigation pane includes sections for 'Overview', 'Kubernetes resources', and 'Settings'. The 'Workloads (preview)' option is highlighted. The main content area shows a table of deployments with the following data:

| <input type="checkbox"/> | Name | Namespace | Ready | Up-to-date | Available | Age ↓ |
|-------------------------------------|---------------------------|-------------|-------|------------|-----------|----------|
| <input type="checkbox"/> | coredns | kube-system | 2/2 | 2 | 2 | 10 hours |
| <input type="checkbox"/> | coredns-autoscaler | kube-system | 1/1 | 1 | 1 | 10 hours |
| <input type="checkbox"/> | metrics-server | kube-system | 1/1 | 1 | 1 | 10 hours |
| <input type="checkbox"/> | omsagent-rs | kube-system | 1/1 | 1 | 1 | 10 hours |
| <input type="checkbox"/> | tunnelfront | kube-system | 1/1 | 1 | 1 | 10 hours |
| <input type="checkbox"/> | dashboard-metrics-scraper | kube-system | 1/1 | 1 | 1 | 10 hours |
| <input type="checkbox"/> | kubernetes-dashboard | kube-system | 1/1 | 1 | 1 | 10 hours |
| <input checked="" type="checkbox"/> | secondsample | default | 5/5 | 5 | 5 | 8 hours |
| <input type="checkbox"/> | secsample | default | 3/3 | 3 | 3 | 8 hours |

10. This again shows a list of all currently running **services** (remember `kubectl get services -watch?`), but nicely integrated in the Azure Portal. **Select “secondsample”** as our service workload, showing additional details for this service.

CHAPTER 9 LAB 7: MANAGING AND MONITORING AZURE KUBERNETES SERVICE (AKS)

secondsample | Overview

Deployment

Search (Ctrl+/) Refresh

Overview
YAML
Events
Insights

Namespace: default
Labels: -
Selector: app=secondsample

Creation time: 2020-08-11T14:10:07.000Z
Replicas: 5 desired, 5 updated, 5 total, 5 available, 0 unavailable
Revision history limit: 10
Min ready seconds: 0
Strategy type: RollingUpdate
Rolling update strategy: 25% max unavailable, 25% max surge

Pods Replica sets

Delete Show labels

| Name | Ready | Status | Restart count | Age | Pod IP | Node |
|------------------------------|-------|---------|---------------|---------|-------------|----------------------------|
| secondsample-6dcd9984f-7oqv4 | 1/1 | Running | 0 | 8 hours | 10.244.0.9 | aks-nodepool1-20945056-... |
| secondsample-6dcd9984f-87xch | 1/1 | Running | 0 | 8 hours | 10.244.1.6 | aks-nodepool1-20945056-... |
| secondsample-6dcd9984f-9ztkc | 1/1 | Running | 0 | 8 hours | 10.244.0.10 | aks-nodepool1-20945056-... |
| secondsample-6dcd9984f-ctwnk | 1/1 | Running | 0 | 8 hours | 10.244.1.7 | aks-nodepool1-20945056-... |

11. Next, click “YAML”; this exposes an actual YAML file configuration, almost similar to the one you used for the initial deployment.

secondsample | YAML

Deployment

Search (Ctrl+/) Refresh

Overview
YAML
Events
Insights

YAML JSON

```
1 kind: Deployment
2 apiVersion: apps/v1
3 metadata:
4   name: secondsample
5   namespace: default
6   selfLink: /apis/apps/v1/namespaces/default/deployments/secondsample
7   uid: c4acc977-ee26-47c0-be8a-92f018f56c24
8   resourceVersion: '10472'
9   generation: 1
10  creationTimestamp: '2020-08-11T14:10:07Z'
11  annotations:
12    deployment.kubernetes.io/revision: '1'
13  spec:
14    replicas: 5
15    selector:
16      matchLabels:
17        app: secondsample
18    template:
19      metadata:
20        creationTimestamp: null
21      labels:
22        app: secondsample
23      spec:
24        containers:
25          - name: secondsample
26            image: 'pdtacr.azurecr.io/simplcdotnet31:latest'
```

- This could become handy for documenting your AKS cluster setup, including running nodes. **Or why not make changes to the actual running state of your service?** To show this, **edit the number of replicas from five to two (line 14 in my example), and save the changes.**

```
13 spec:
14   replicas: 2
15   selector:
16     matchLabels:
17       app: secondsample
18   template:
19     metadata:
20       creationTimestamp: null
21     labels:
22       app: secondsample
23     spec:
24       containers:
25         - name: secondsample
26           image: 'pdtacr.azurecr.io/simplcdotnet31:latest'
```

[Review + save](#)[Discard](#)

- The YAML file is getting updated, highlighting the change, **asking you to confirm the manifest changes and saving these once more.**


```

14  -- replicas: 5
14+ replicas: 2
15  15 selector:
16  16 |   matchLabels:
17  17 |     app: secondsample
18  18 template:
19  19 |   metadata:
20  20 |     creationTimestamp: null
21  21 |   labels:
22  22 |     app: secondsample
    
```

Confirm manifest changes

- After saving the change and waiting for it to get applied, switch back to the Overview pane of this detailed view. See how the number of replicas of the “secondsample” has now gone down to only two.

| Pods | | Replica sets | | | | | |
|---|--|--------------|---------------|---------|------------|----------------------------|--|
| Name | Ready | Status | Restart count | Age ↓ | Pod IP | Node | |
| <input type="checkbox"/> secondsample-6dcc99f4f-87xch | ● 1/1 | Running | 0 | 8 hours | 10.244.1.6 | aks-nodepool1-20945056-... | |
| <input type="checkbox"/> secondsample-6dcc99f4f-ctwnk | ● 1/1 | Running | 0 | 8 hours | 10.244.1.7 | aks-nodepool1-20945056-... | |

- The same goes for the **Services and ingresses** topic within the **Kubernetes resources** section; by selecting it, a list of currently running services will be shown, similar to the `kubectl get services` command you ran earlier.

| Service Name | Namespace | Status | Type | IP Address | Public IP |
|------------------------------|-------------|--------|--------------|--------------|---------------|
| kubernetes | default | Ok | ClusterIP | 10.0.0.1 | |
| kube-dns | kube-system | Ok | ClusterIP | 10.0.0.10 | |
| metrics-server | kube-system | Ok | ClusterIP | 10.0.172.64 | |
| healthmodel-replicaset-se... | kube-system | Ok | ClusterIP | 10.0.171.235 | |
| dashboard-metrics-scraper | kube-system | Ok | ClusterIP | 10.0.76.41 | |
| kubernetes-dashboard | kube-system | Ok | ClusterIP | 10.0.241.243 | |
| firstsample | default | Ok | LoadBalancer | 10.0.179.30 | 20.50.161.123 |
| secondsample | default | Ok | LoadBalancer | 10.0.132.152 | 20.50.215.134 |
| secsample | default | Ok | LoadBalancer | 10.0.251.121 | 52.236.157.25 |

16. **This time**, it also shows the public IP address for each of the running services (that's what the ingresses refer to, incoming traffic from the public Internet).

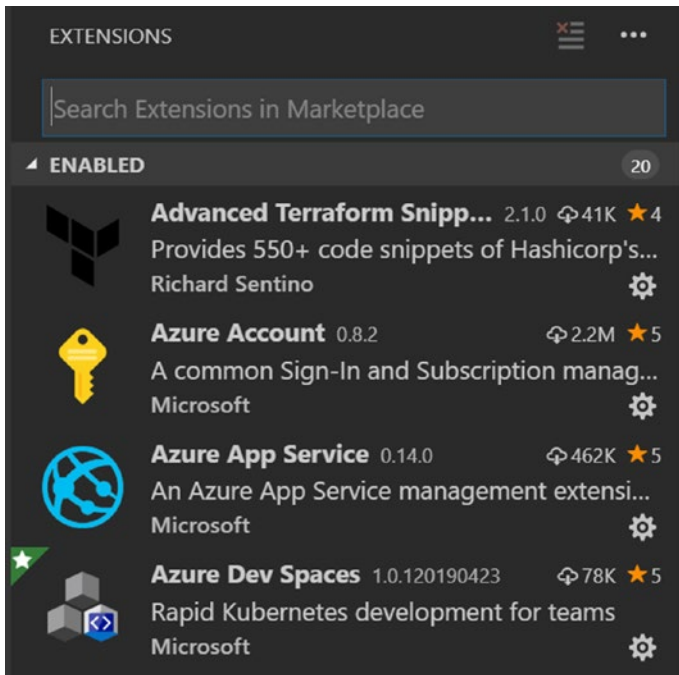
Note These last few screens and options are still in preview, which means they might have changed by the time you go through this exercise yourself. It's yet another nice improvement, trying to help Azure customers in managing the AKS environment, without requiring to be an expert on **kubectl** commands.

This concludes this part of the task, in which you learned how to manage your AKS environment from the Azure Portal.

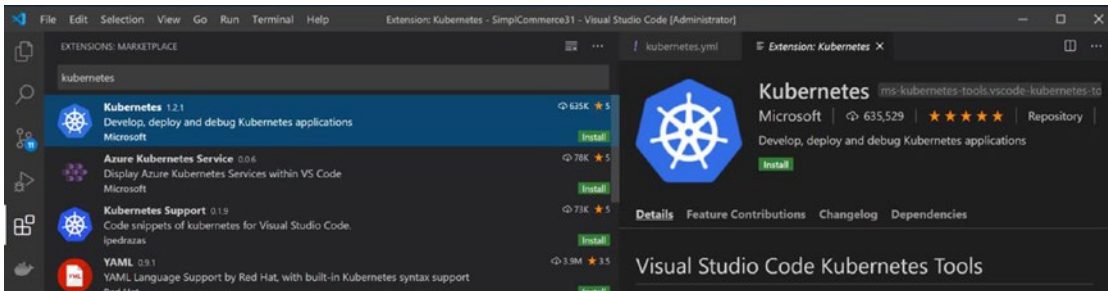
Task 3: Managing Kubernetes from Visual Studio Code

Besides the Azure built-in monitoring tools in the previous task, one can also manage the AKS cluster using Visual Studio Code.

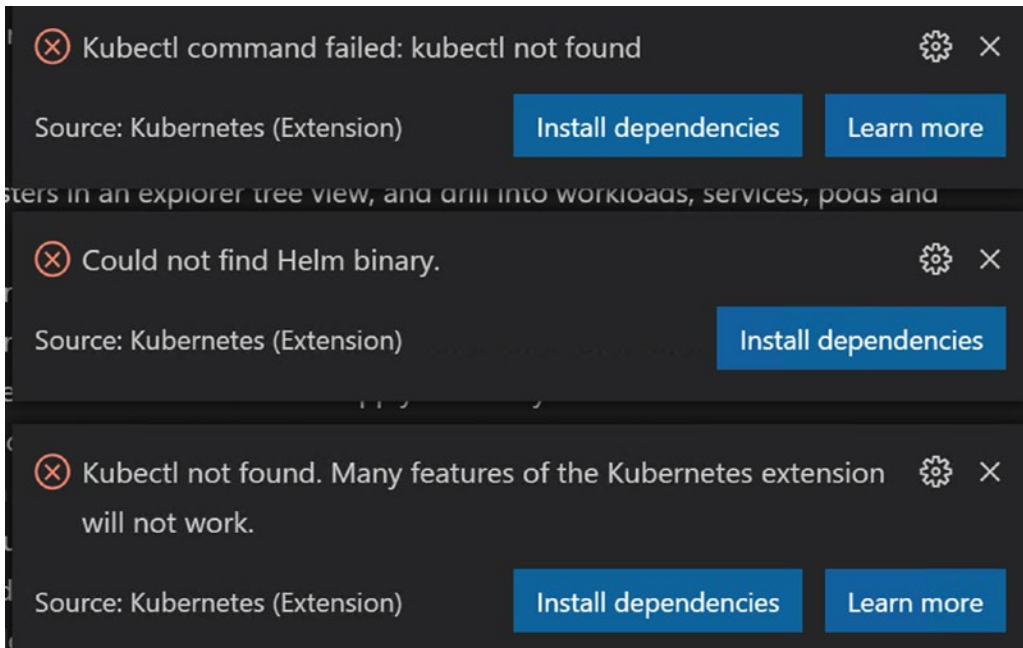
1. Once Visual Studio Code is launched, from the menu, go to **File ► Preferences ► Extensions**. This shows a list of community and third-party vendor-provided extensions.



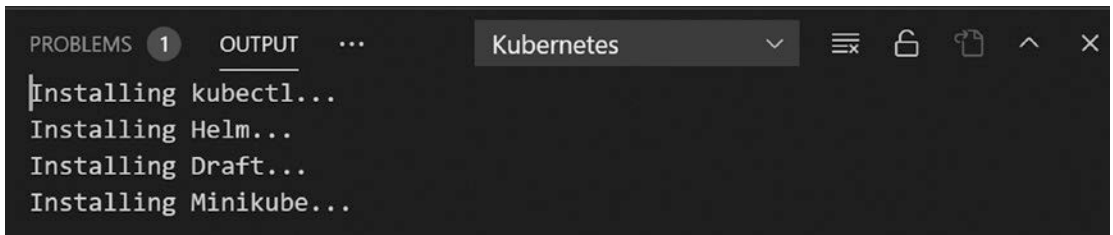
2. In the **Search Extensions Marketplace**, type “**kubernetes**”.



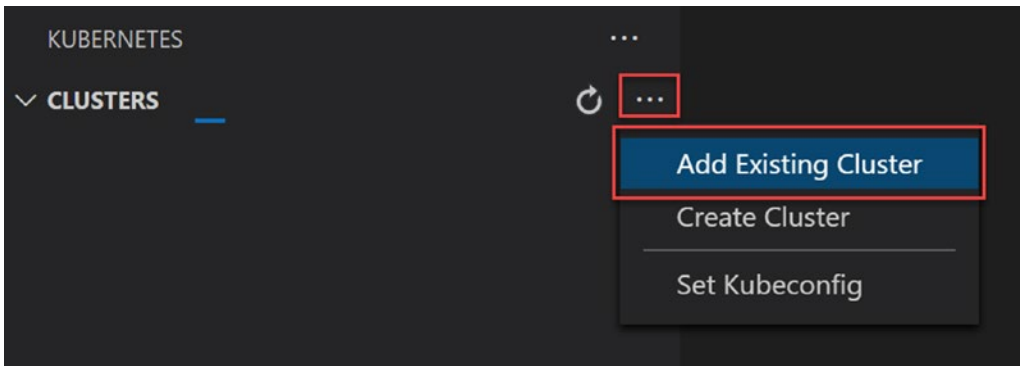
3. **Click Install** and wait for the extension to get installed successfully. You will see a shortcut to it in the left menu sidebar. **Click it.** Since this extension requires kubectl, which is not preloaded on the VM, the extension fails.



4. Click **“Install dependencies”** (only once is ok) and follow the progress from the terminal window.



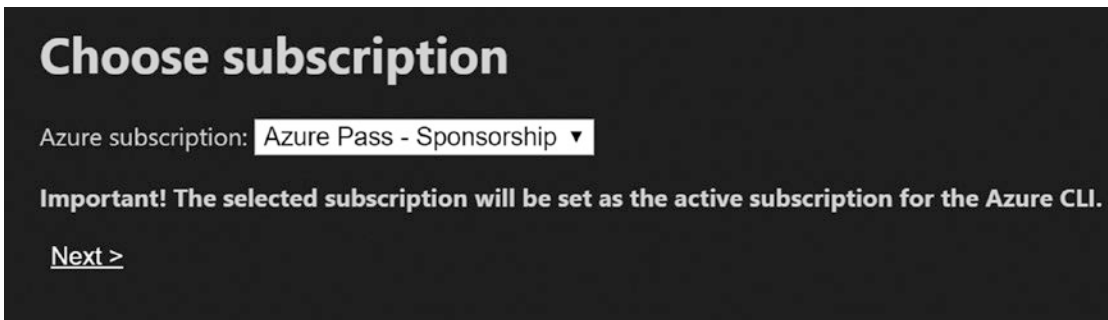
5. Once the tools and dependencies have installed, refresh the Kubernetes extension by clicking its icon in the sidebar. This time no error messages show up anymore. From the Kubernetes section, click the ellipsis, and select **Add Existing Cluster**.



- 6. You will need to go through a series of questions, to make sure the extension picks up the correct information.



- 7. Cluster type is Azure Kubernetes Service. Click Next >.



- 8. Select your Azure subscription then click Next >.

Choose cluster

Kubernetes cluster: AKSCluster (in AKSNativeRG) ▼

[Add this cluster >](#)

9. Select AKSCluster as Kubernetes cluster. Click “Add this cluster >”

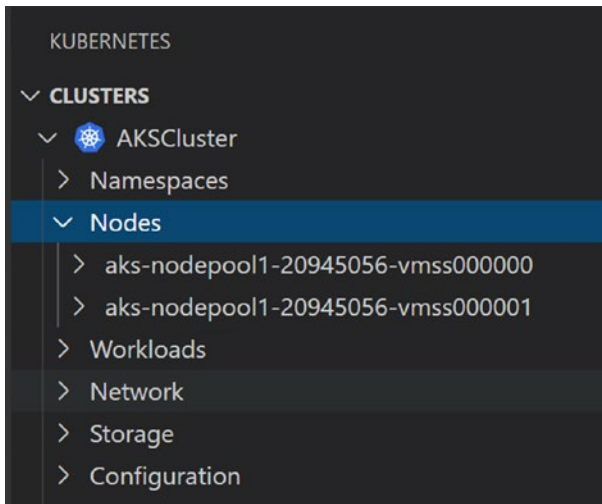
Cluster added

kubectl installed at C:\Users\labadmin\AppData\Local\kubectl\kubectl.exe

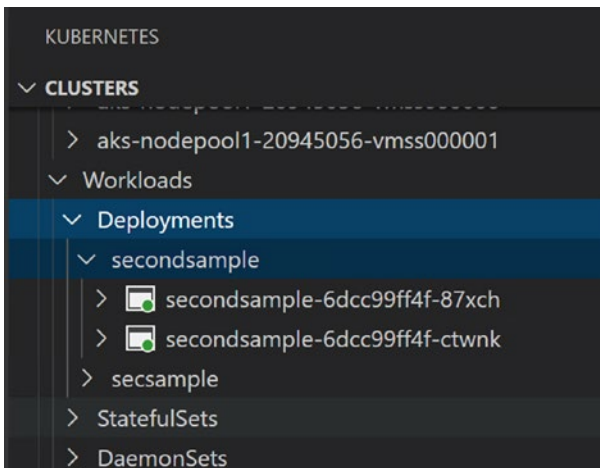
This location is not on your system PATH. Add this directory to your path, or set the VS Code **vs-kubernetes.kubectl-path** config setting.

Successfully configured kubectl with Azure Kubernetes Service cluster credentials.

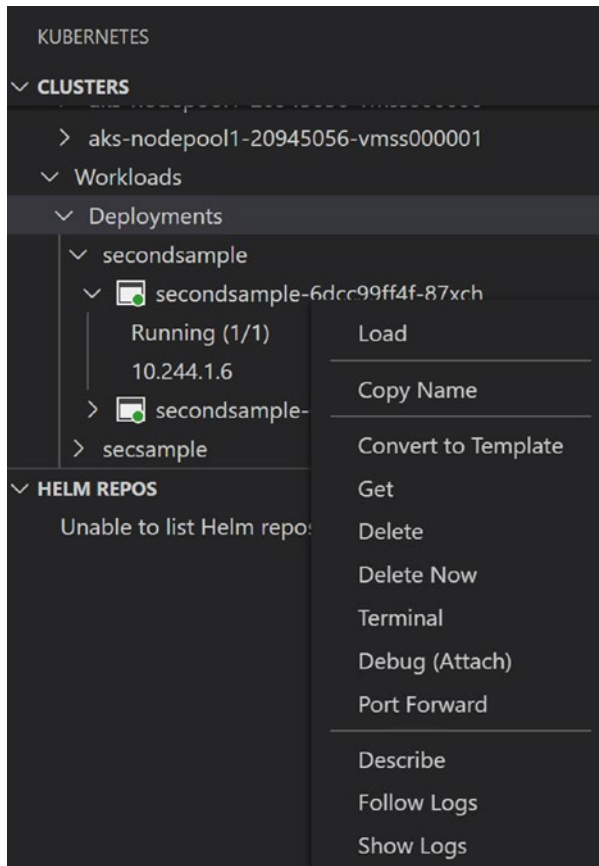
10. The cluster got added successfully. From the Kubernetes pane, notice the AKSCluster resource is visible, allowing you to “manage” the cluster services and components.



11. Click Deployments.



12. This shows the earlier pushed deployment for service “secondsample,” where we have two replicas running. Right-click any of the running PODs, to see an action menu.



13. Feel free to perform an action against a running POD, for example, Delete Now.

This completes the lab exercise.

Summary

In this lab, you learned the basic admin tasks about scaling Azure Kubernetes Service, using both the Azure Portal and `kubectl` command line. Next, you became familiar with the Azure Monitor capabilities of Kubernetes monitoring, as well as how the built-in standard Kubernetes dashboard can be used besides the Azure monitoring capabilities. Last, you deployed the Kubernetes extension in Visual Studio Code and performed some basic operations against the AKS cluster from there.

CHAPTER 10

Lab 8: Deploying Azure Workloads Using Azure DevOps

What You Will Learn

In this next lab, you get introduced to Azure DevOps, Microsoft's tooling which allows for CI/CD pipeline deployments of application workloads to Azure (as well as other platforms). Starting from creating our Azure DevOps organization and project, you kick off the process by importing the source code of our sample e-commerce application from the GitHub repo into Azure DevOps Repos and learn the basics of Git and branching. Next, you get introduced to creating a build pipeline using the Azure DevOps classic editor as well as the newer pipeline.yml approach. From here, you will also learn how to deploy the previously built Docker container and run this in Azure Container Instance, but deployed using Azure DevOps release pipelines. Lastly, you will deploy the Docker container to the AKS cluster you deployed earlier, again using Azure DevOps release pipelines.

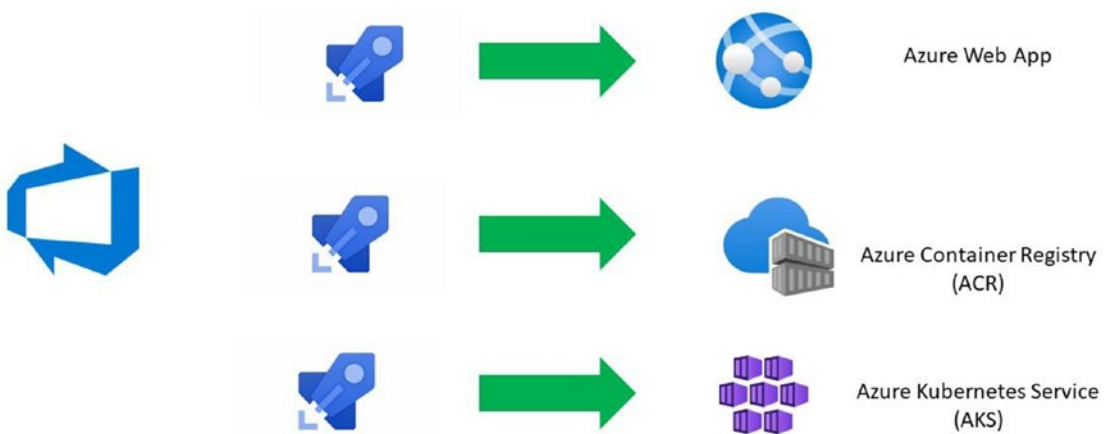
Time Estimate

This lab is estimated to take 90 min.

Prerequisites

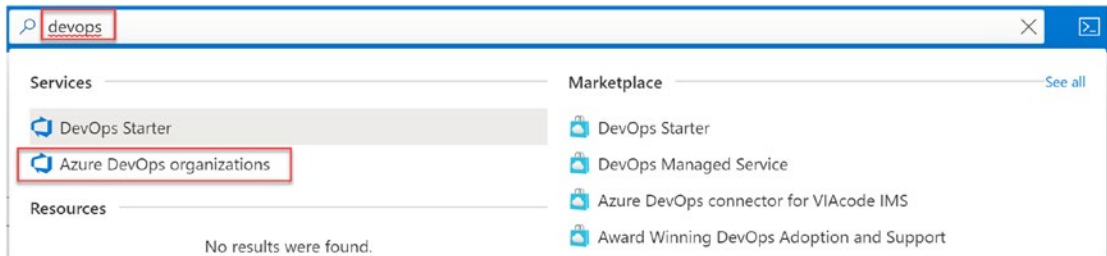
This lab continues on the deployments from Labs 3, 4, 5, 6, and 7. Make sure you successfully completed those, before starting this lab.

Scenario Diagram



Task 1: Deploying an Azure DevOps organization

1. From the “Search for resources, services, and docs (G+/)” field, search for **devops**.



2. Select **Azure DevOps organizations**.

[Home](#) >

Azure DevOps

We've made it easier to manage Azure DevOps billing and subscriptions. You can [set up billing](#), [change your subscription](#) or pay for more users and resources within Azure DevOps. [Learn more](#).

Azure DevOps

Plan smarter, collaborate better, and ship faster with a set of modern dev services

[My Azure DevOps Organizations](#)

Get started using Azure DevOps
Billing management for Azure DevOps



3. **From the Azure DevOps start screen, click “My Azure DevOps Organizations”**; this redirects you to the dev.azure.com portal, where you need to provide some additional details about your user and organizational profile.

We need a few more details

Your name:

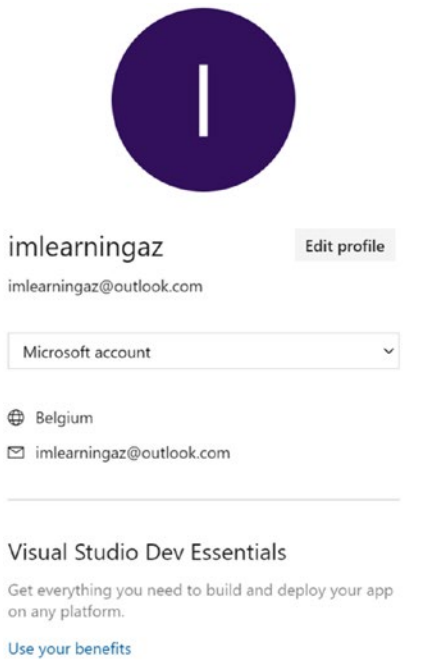
We'll reach you at:

From:

- I would like to receive information, tips, and resources related to Microsoft developer tools and services, including Azure DevOps, Visual Studio, Visual Studio Subscriptions, and other Microsoft products and services.

Continue

4. **Click Continue.**



imlearninggaz [Edit profile](#)

imlearninggaz@outlook.com

Microsoft account

🌐 Belgium

✉ imlearninggaz@outlook.com

Visual Studio Dev Essentials

Get everything you need to build and deploy your app on any platform.

[Use your benefits](#)

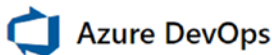


Get started with Azure DevOps

Plan better, code together, ship faster with Azure DevOps

[Create new organization](#)

5. Click **Create new organization**.



imlearninggaz@outlook.com [Switch directory](#)

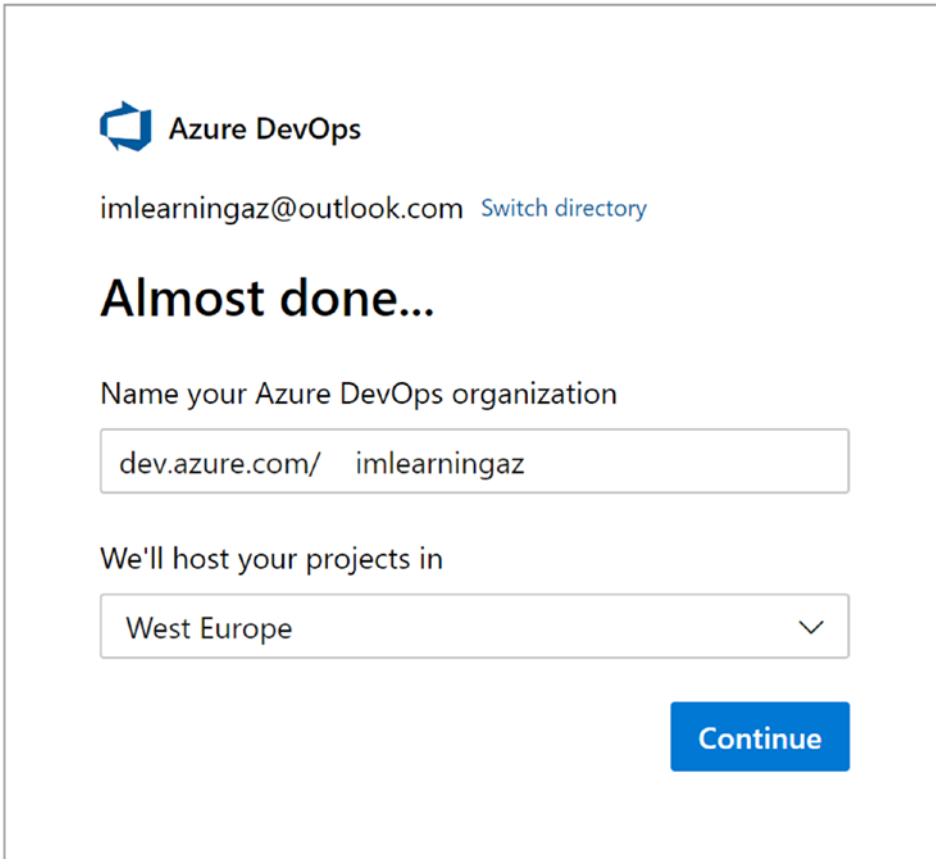
Get started with Azure DevOps

Choosing **Continue** means that you agree to our [Terms of Service](#), [Privacy Statement](#), and [Code of Conduct](#).

- I would like information, tips, and offers about Azure DevOps and other Microsoft products and services. [Privacy Statement](#).

[Continue](#)

6. **Click Continue.**

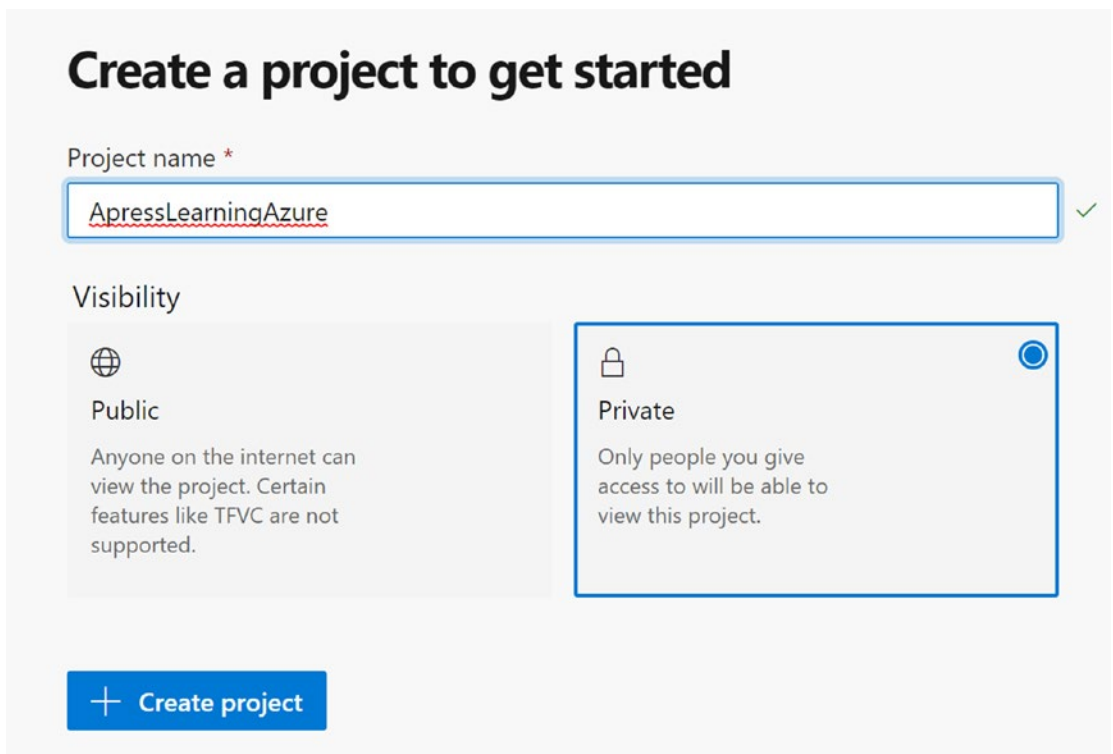


The screenshot shows the Azure DevOps organization creation interface. At the top left is the Azure DevOps logo. Below it, the user's email address 'imlearningaz@outlook.com' is displayed with a 'Switch directory' link. The main heading is 'Almost done...'. Below this, there is a prompt 'Name your Azure DevOps organization' followed by a text input field containing 'dev.azure.com/ imlearningaz'. Underneath is another prompt 'We'll host your projects in' followed by a dropdown menu currently set to 'West Europe'. A blue 'Continue' button is located at the bottom right of the form area.

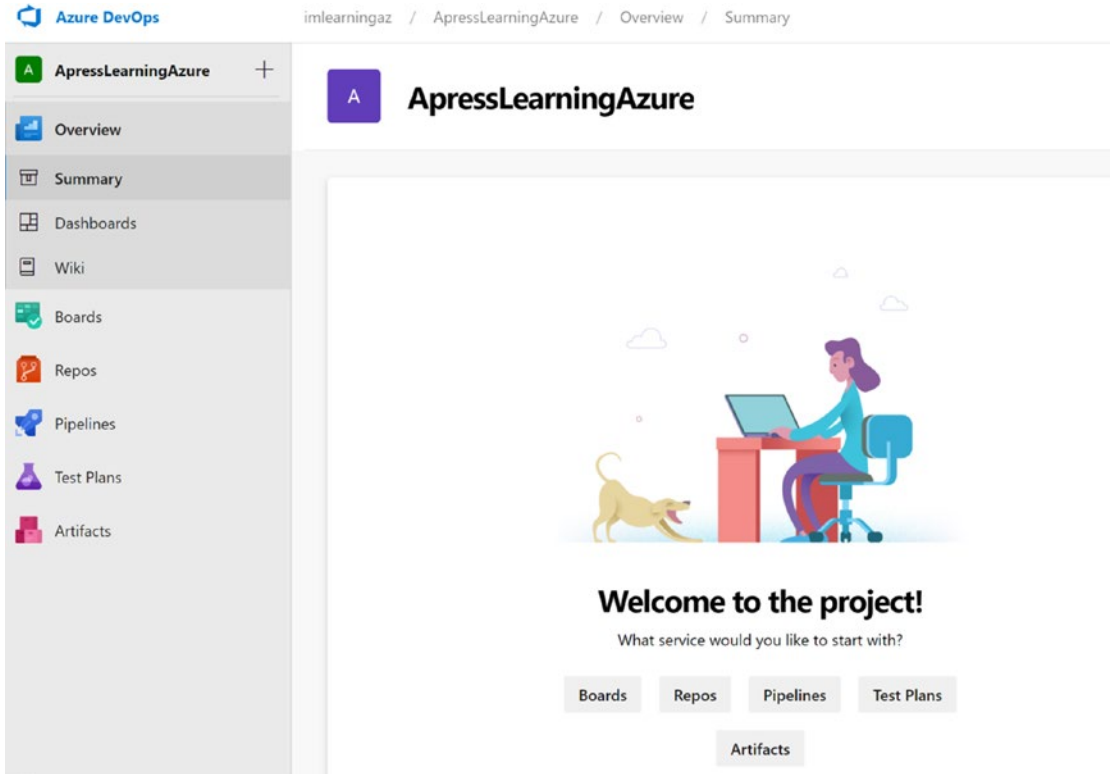
7. **Provide a unique name for your Azure DevOps organization and what Azure region you want to use for hosting the projects. Confirm by clicking Continue.**



8. Wait for this process to complete, after which you are redirected to the Azure DevOps portal (dev.azure.com/<organizationname>), where you are asked to create a new project.



- 9. **Define a project name, and set visibility to private (which means that only users within your organization can get access to it).** Confirm by **clicking the “+ Create project”** button; your Azure DevOps “Workspace” gets created.



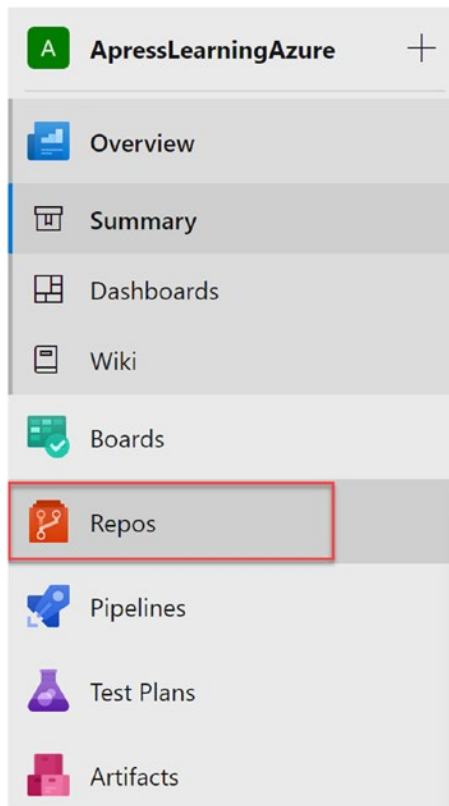
This completes the task, in which you deployed Azure DevOps and configured an Azure DevOps organization. In the next task, you will start using Azure DevOps Repos as a source control/version control mechanism.

Task 2: Introduction to source control with Azure DevOps Repos

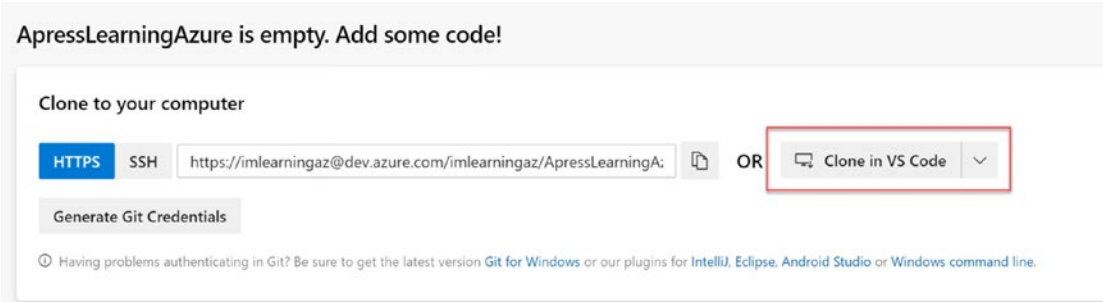
The starting point of many successful deployments is source code. This can be application source code like a dotnetcore web app, but could also be used for Azure templates, PowerShell scripts, or basically any other data source facing regular updates. A popular source control solution today is GitHub (www.github.com), which by itself is based on Git, a distributed source control/version control solution. While GitHub is very useful, it is mainly used for public and community-based source code publishing. But what if you want to keep your source code “internal”? Like within your DevOps projects themselves? That’s what Azure DevOps Repos offers: a Git-compatible source control service.

This task introduces you to the basics of source control, guiding you through cloning a public GitHub repo into Azure DevOps Repos, from where you will work with versioning and branching. These changes will be used later on for the build and release pipelines.

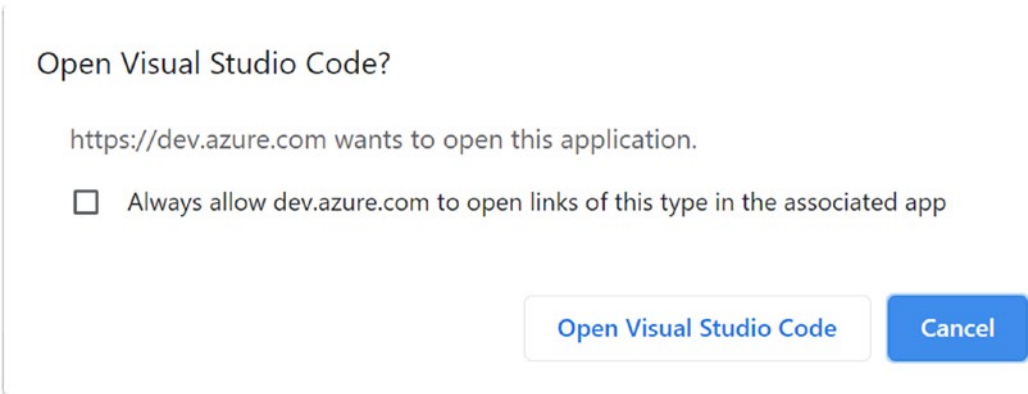
1. **From the Azure DevOps portal, select Repos.**



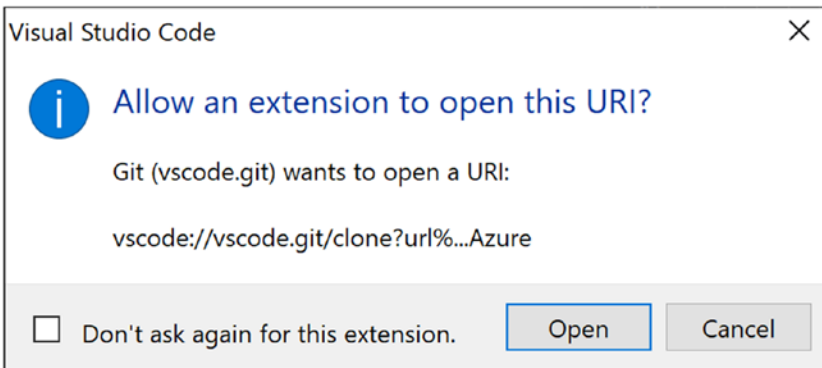
- 2. **This gives you several options to choose from, specifying how this repo will be used. Select “Clone in VS Code.”**



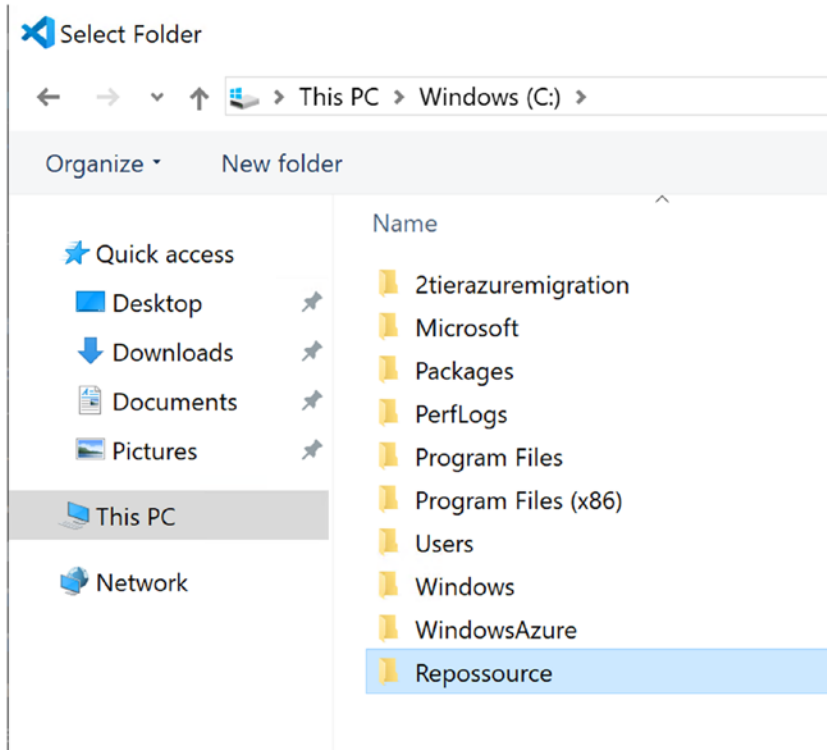
- 3. **Confirm to open this repo in VS Code from the popup box.**



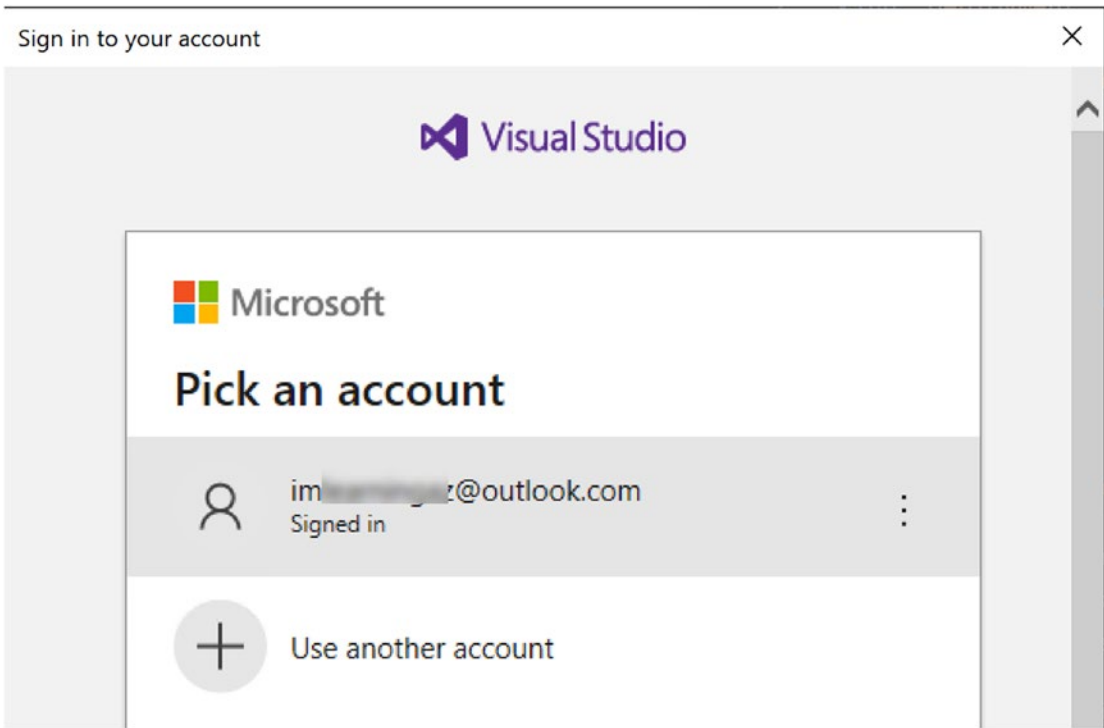
- 4. **This opens VS Code, asking you a confirmation to open the URI link; confirm this by clicking “Open.”**



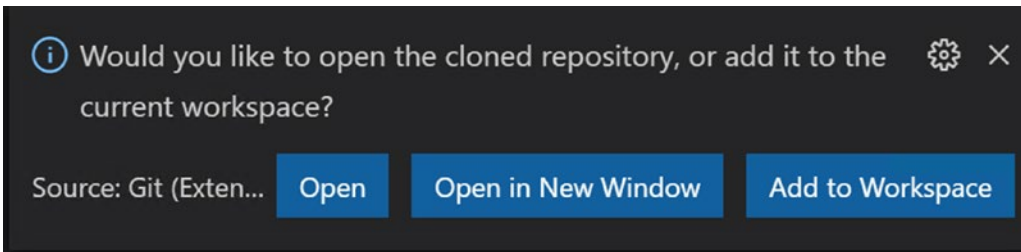
5. **Once opened, you need to specify where VS Code needs to clone the Azure DevOps Repos folder. Browse to the local C drive, and create a new folder, named Repossource.**



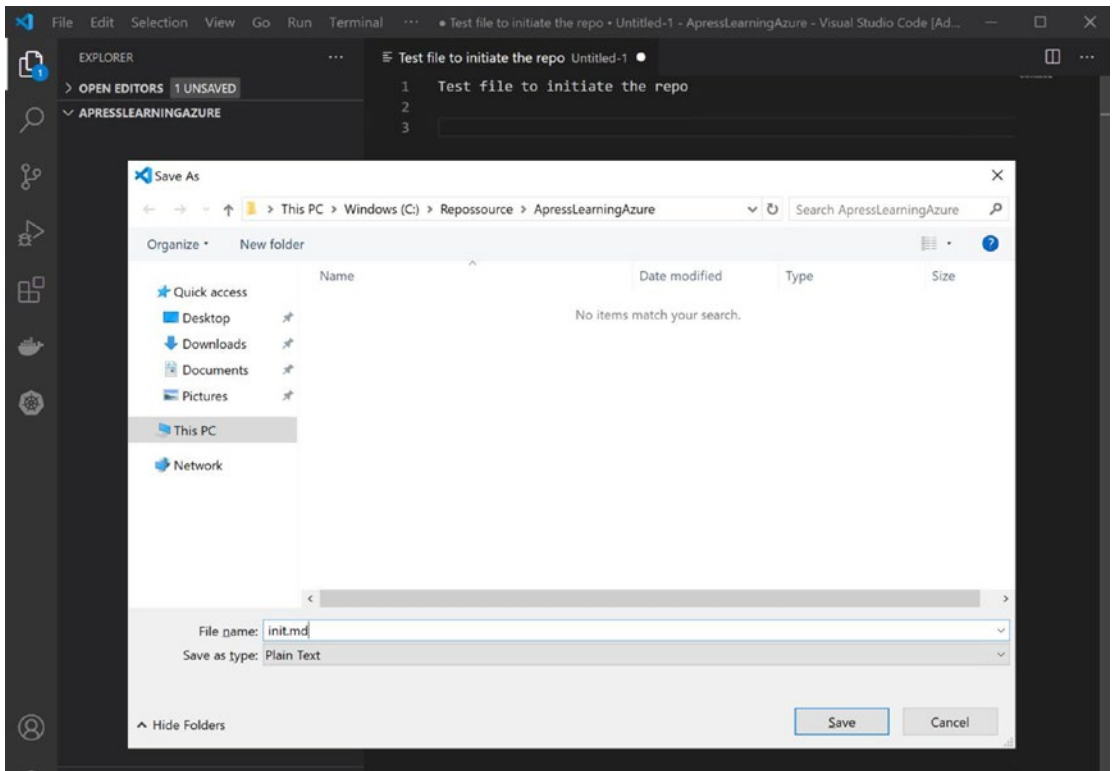
6. **In order to be able to clone, you need to provide your Azure DevOps credentials.**



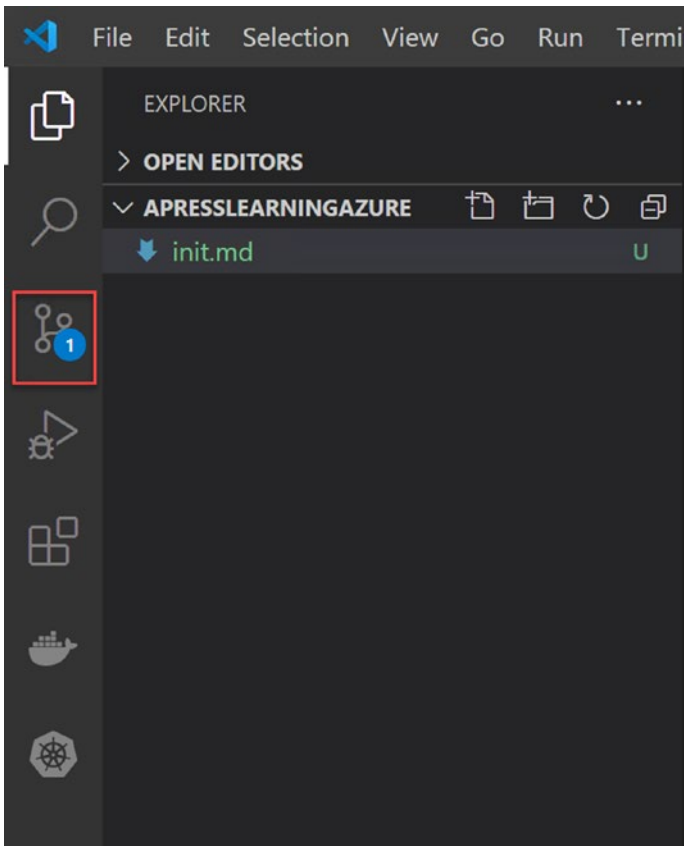
- 7. After which, VS Code will provide you a prompt, asking if you want to open this folder; select **Open in New Window**.



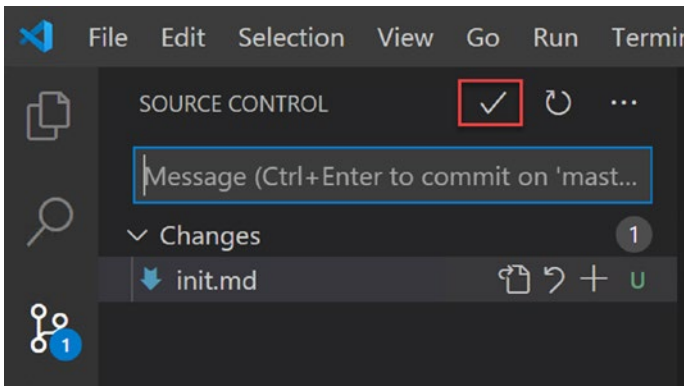
- 8. From here, let's at some "source code," by creating a new file, typing some text (e.g., *Test file to initiate the repo*), and save the file in the root of the Repossource folder; I called my example "init.md", but this is not that important. It can be saved as text file as well, with a name of your choice.



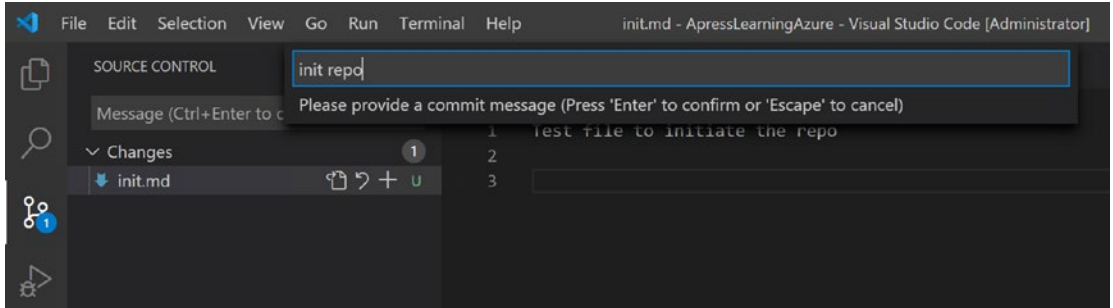
9. **Since the Reposource folder is automatically “Git-enabled,” thanks to Azure DevOps Repos, we can make use of the source control extension as part of VS Code. Click the source control icon.**



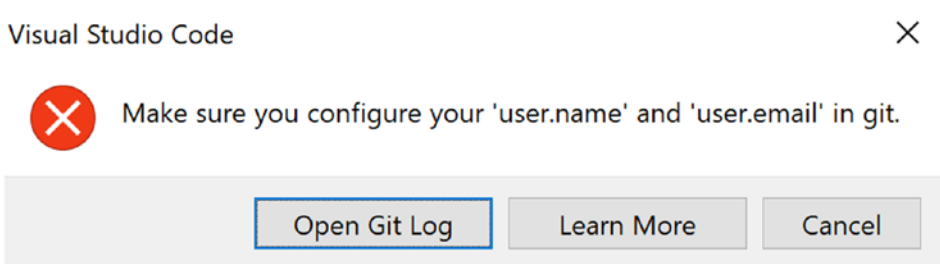
- 10. Notice how it picked up the “init.md” as a change, waiting to be “pushed” back to Azure DevOps Repos. To do this, click the “Commit” button.



11. It will ask you to provide a “message,” which typically refers to the updates done to the repository (e.g., init repo or anything).



12. This throws an error message.

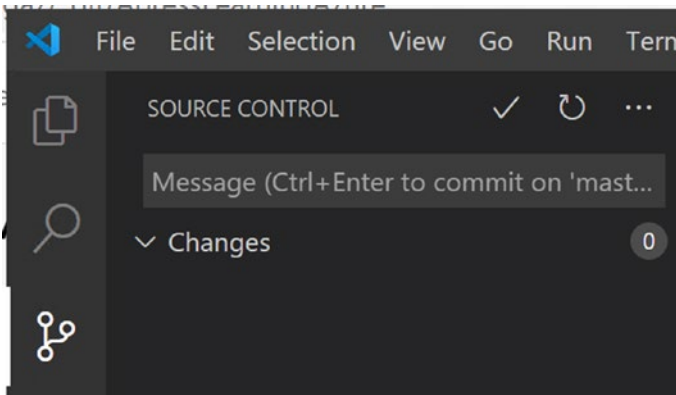


13. What this refers at is that each “**git commit**” must be linked to an individual person, in order to trace back who made changes. This is done by setting “Git variables,” which you didn’t do yet. **Click “Open Git Log,”** which redirects you to the “**Output**” window of VS Code; for now, the relevant information is executing the following two commands from the “**terminal**”:

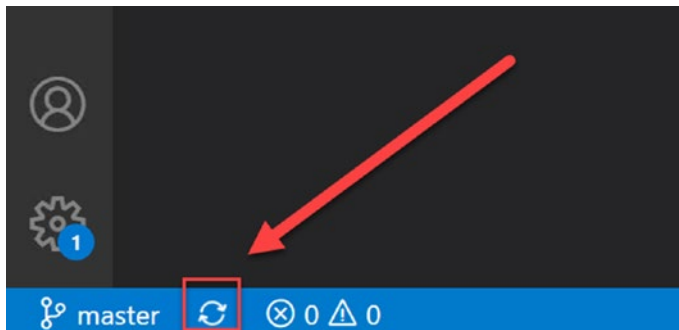
```
git config --global user.email "your email address"
git config --global user.name "your name"
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: powershell + [ ] [ ] ^ x
PS C:\Reposource\ApressLearningAzure> git config --global user.email "you@example.com"
PS C:\Reposource\ApressLearningAzure> git config --global user.email "imlearningaz@outlook.com"
PS C:\Reposource\ApressLearningAzure> git config --global user.name "Az Learner"
PS C:\Reposource\ApressLearningAzure> |
```

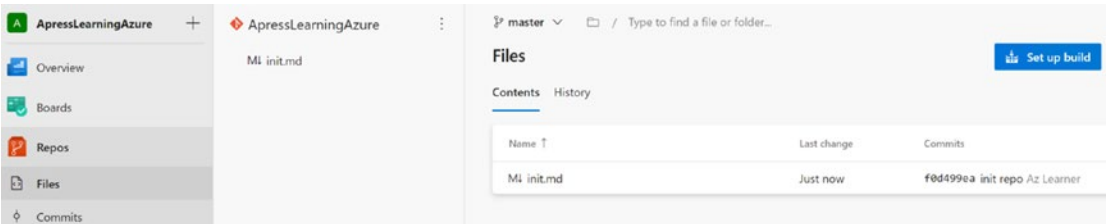
- 14. **Once these variables are set, return to the source control view, and commit your changes again.**



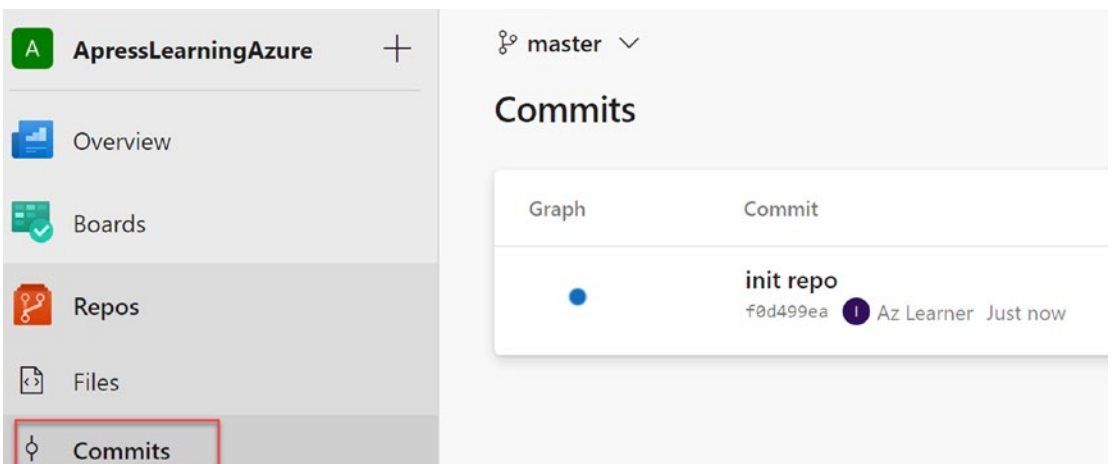
- 15. **While the commit was done successfully, it doesn't mean the file has been uploaded to Azure DevOps Repos yet; VS Code has a built-in "safety net" (as I call it) to not sync immediately, but rather waiting for you to trigger this automatically (this could be handy when you detect mistakes in your source control, allowing you to edit and commit the change again - all this happens locally, without impacting the actual Azure DevOps Repos). To force the sync, click the "sync changes icon."**



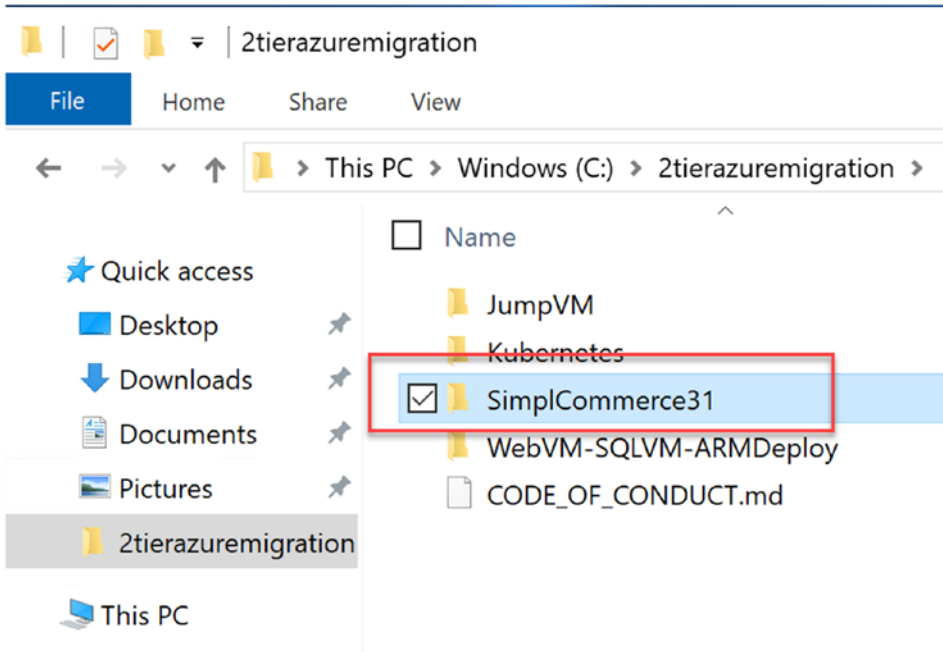
16. **After a few seconds, you can check back in the Azure DevOps Repos portal and see the new file you created showing up there.**



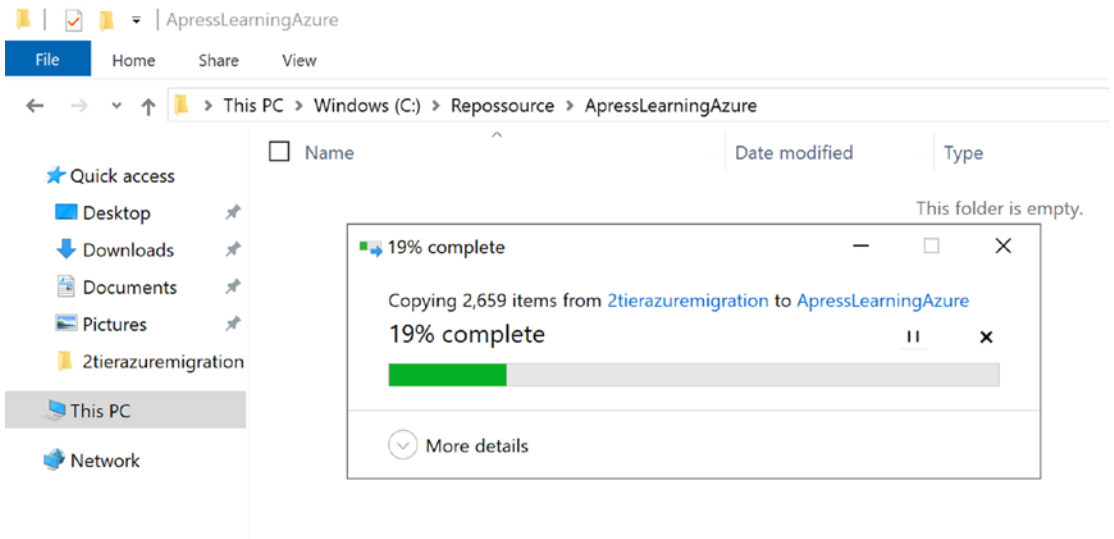
17. **From Repos, select “Commits”; this shows the trace of your previous commits, triggered from VS Code. Notice how it recognizes your “name,” as well as showing the “message” you provided.**



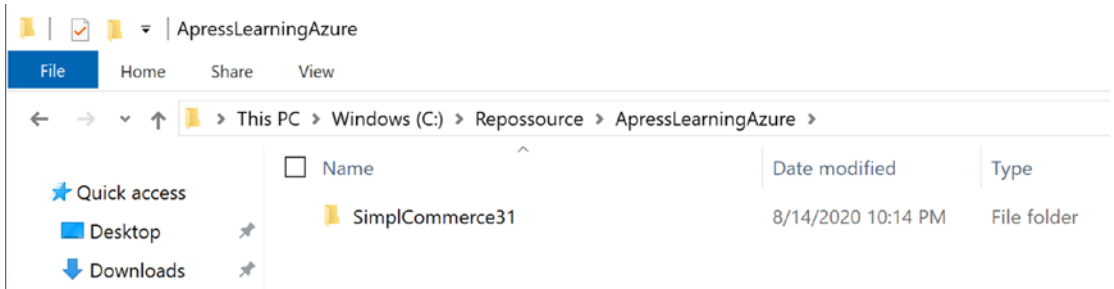
- 18. **As you now know the sync is working, thanks to Git integration out of Azure DevOps Repos, we can “upload” the source folder we used earlier into this Repos. To do this, open your File Explorer, and browse to the 2tierAzureMigrate source folder. Select ONLY the SimplCommerce31 folder.**



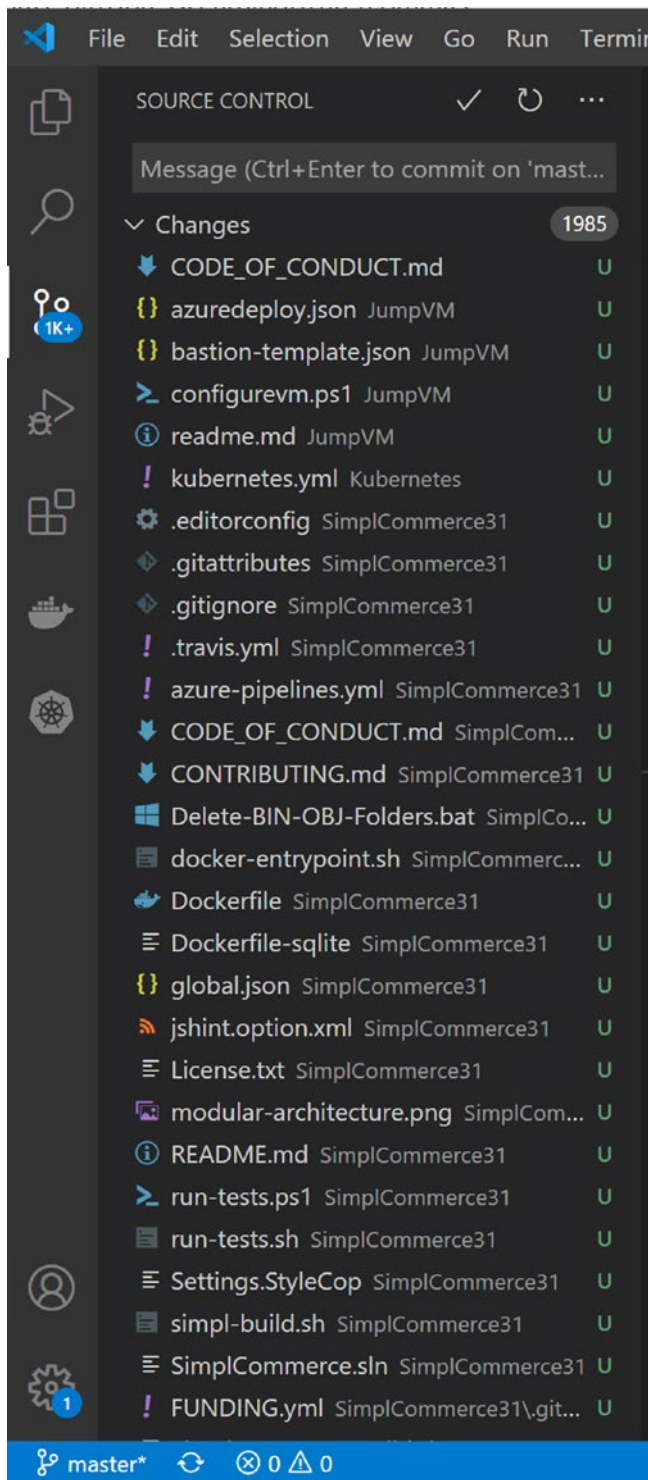
- 19. **Copy this folder to the target directory “Reposource,” noticing there is already a subfolder, named after the Azure DevOps Project.**



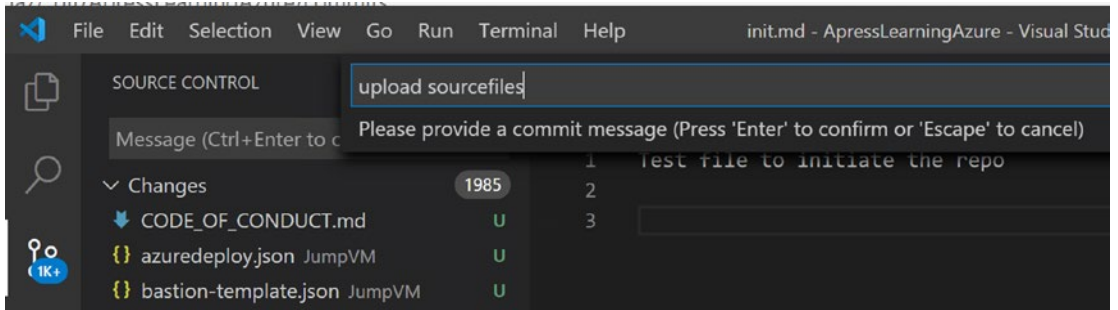
20. **The content should look like this now:**



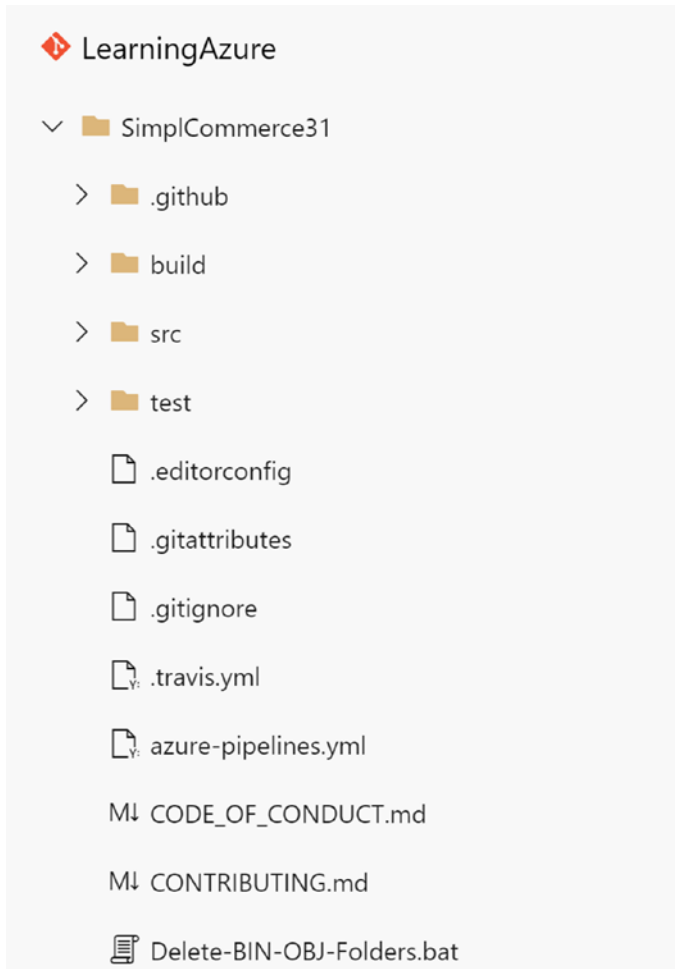
21. **Once the copy operation is complete, switch back to VS Code, and open the source control extension. This has picked up all the changes, ready to be “committed.”**



22. **Commit the changes, and provide a descriptive message.**



23. **Next, click the “sync changes” icon again, and wait for all files to get pushed into Azure DevOps Repos. After about a minute, this should be completed.**



This completes the task in which you learned about source control, based on Azure DevOps Repos. Given the integration with Git, it allows a clone to VS Code (among other development tools), providing DevOps engineers with the necessary integration to enable source control, commit changes, and keep source code in sync. In the following task, you will create a build pipeline, based on this source control repository.

Task 3: Creating and deploying an Azure build pipeline for your application

While Azure DevOps gives you an end-to-end solution to manage your application development and deployment lifecycle, this lab focuses mainly on the **Azure Pipelines** service within.

1. **Select “Pipelines,” and within, select “Pipelines” once more.**



2. **You are greeted to create your first pipeline.**



Create your first Pipeline

Automate your build and release processes using our wizard, and go from code to cloud-hosted within minutes.





3. **Click Create Pipeline. This launches the Pipeline wizard, starting with the source code location.**


Connect Select Configure


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 “Azure Repos Git,” followed by selecting the repository you created earlier.**

✓ Connect **Select** Configure Review

New pipeline

Select a repository

Filter by keywords ApressLearningAzure ▾ ✕









 ApressLearningAzure

- This brings you to the “Configure your pipeline” blade; based on the source code, it will offer you different selections. Since our application is a dotnetcore app, select ASP.NET Core.**

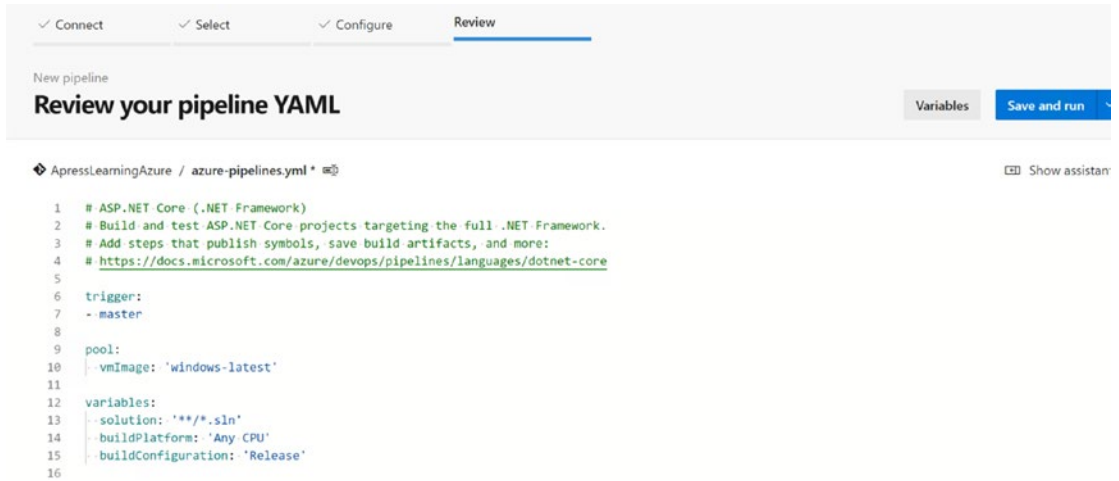
✓ Connect ✓ Select **Configure** Review

New pipeline

Configure your pipeline

-  **Docker**
Build a Docker image
-  **Docker**
Build and push an image to Azure Container Registry
-  **Deploy to Azure Kubernetes Service**
Build and push image to Azure Container Registry; Deploy to Azure Kubernetes Service
-  **Deploy to Kubernetes - Review app with Azure DevSpaces**
Build and push image to Azure Container Registry; Deploy to Azure Kubernetes Services and setup Review App with Azure DevSpaces
-  **ASP.NET**
Build and test ASP.NET projects.
-  **ASP.NET Core (.NET Framework)**
Build and test ASP.NET Core projects targeting the full .NET Framework.
-  **.NET Desktop**
Build and run tests for .NET Desktop or Windows classic desktop solutions.
-  **Universal Windows Platform**
Build a Universal Windows Platform project using Visual Studio.

6. This results in an azure-pipelines.YML file, storing the actual configuration of the build pipeline.



Connect Select Configure **Review**

New pipeline

Review your pipeline YAML Variables Save and run

◆ ApressLearningAzure / azure-pipelines.yml * #

```

1 # ASP.NET Core (.NET Framework)
2 # Build and test ASP.NET Core projects targeting the full .NET Framework.
3 # Add steps that publish symbols, save build artifacts, and more:
4 # https://docs.microsoft.com/azure/devops/pipelines/languages/dotnet-core
5
6 trigger:
7   - master
8
9 pool:
10  vmImage: 'windows-latest'
11
12 variables:
13   - solution: '**/*.sln'
14   - buildPlatform: 'Any CPU'
15   - buildConfiguration: 'Release'
16

```

Show assistant

7. While this file is already quite useful, we are going to make a few changes to the tasks, outside of the default configuration offered here. Scroll down to line 24, where you find the task “VSBuild@1”:

```

Settings
24 - task: VSBuild@1
25   inputs:
26     solution: '$(solution)'
27     msbuildArgs: '/p:DeployOnBuild=true /p:WebPublishMethod=Package /p:PackageAsSingleFile=true /p:SkipInvalidConfigurations=true /p:DesktopBuildPa
28     platform: '$(buildPlatform)'
29     configuration: '$(buildConfiguration)'

```

8. Replace the msbuildArgs line with the following update:

msbuildArgs: /p:DeployOnBuild=true /p:DeployDefaultTarget=WebPublish /p:WebPublishMethod=FileSystem /p:publishUrl="\$(Agent.TempDirectory)\WebAppContent\\"

Note this should all be on a single line.

9. The new layout should be similar to this:

```

Settings
24 - task: VSBuild@1
25   inputs:
26     solution: '$(solution)'
27     msbuildArgs: /p:DeployOnBuild=true /p:DeployDefaultTarget=WebPublish /p:WebPublishMethod=FileSystem /p:publishUrl="$(Agent.TempDirectory)\WebApp
28     platform: '$(buildPlatform)'
29     configuration: '$(buildConfiguration)'
30

```

10. Also verify the indentation of the line is at the same level as solutions, platform, and configuration; these actually define the parameters (input) for this task. If the indentation is wrong, these will not be recognized however.
11. Next, below the VSBuild@1 task, add a new task, by inserting the following lines:

```
- task: ArchiveFiles@2
  displayName: Archive Files
  inputs:
    rootFolderOrFile: $(Agent.TempDirectory)\WebAppContent
    includeRootFolder: false
```

12. The file structure should look as in the following:

```
Settings
24 - task: VSBuild@1
25   inputs:
26     solution: '$(solution)'
27     msbuildArgs: /p:DeployOnBuild=true /p:DeployDefaultTarget=WebPublish /p:WebPublishMethod=FileSystem /p:publishUrl="$(Agent.TempDirectory)\WebAp
28     platform: '$(buildPlatform)'
29     configuration: '$(buildConfiguration)'
30
31 Settings
32 - task: ArchiveFiles@2
33   displayName: Archive Files
34   inputs:
35     rootFolderOrFile: $(Agent.TempDirectory)\WebAppContent
36     includeRootFolder: false
```

13. Validate the indentation of “- task,” making sure it is in line with the level of the previous task, as well as for the displayName and inputs.

Last, paste in the following new task “PublishBuildArtifacts@1,” based on the following lines, at the end of the current file:

```
- task: PublishBuildArtifacts@1
  inputs:
    PathtoPublish: $(Build.ArtifactStagingDirectory)
    ArtifactName: drop
    publishLocation: Container
```

14. The file structure should look like the following:

```
Settings
31 -- task: ArchiveFiles@2
32 | . displayName: Archive Files
33 | . inputs:
34 | . . . rootFolderOrFile: $(Agent.TempDirectory)\WebAppContent
35 | . . . includeRootFolder: false
36
Settings
37 -- task: VSTest@2
38 | . inputs:
39 | . . . platform: '$(buildPlatform)'
40 | . . . configuration: '$(buildConfiguration)'
41
Settings
42 -- task: PublishBuildArtifacts@1
43 | . inputs:
44 | . . . PathToPublish: $(Build.ArtifactStagingDirectory)
45 | . . . ArtifactName: drop
46 | . . . publishLocation: Container
47 |
```

15. Click Save and run; accept the defaults and confirm once more.

Save and run ✕

Saving will commit azure-pipelines.yml to the repository.

Commit message

Set up CI with Azure Pipelines

Optional extended description

Add an optional description...

Commit directly to the master branch
 Create a new branch for this commit

16. This creates your pipeline and initiates the build job against the Azure DevOps Agents.

#20200814.1 Set up CI with Azure Pipelines Cancel ⋮
on LearningAzure

Summary

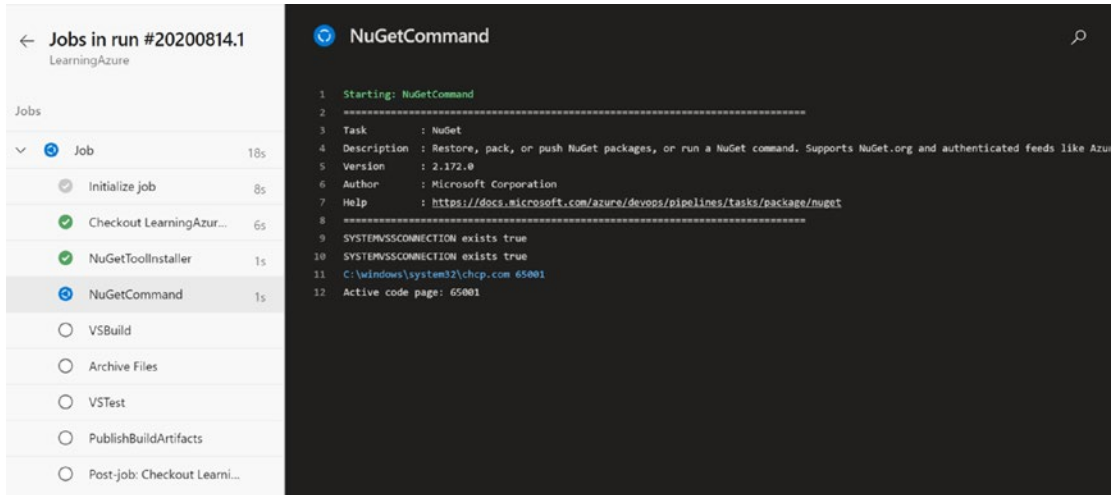
Triggered by i imlearningaz View 2 changes

| | | | |
|------------------------|--------------------------|--------------|-----------------------------|
| Repository and version | Time started and elapsed | Related | Tests and coverage |
| LearningAzure | Just now | 0 work items | Get started |
| master 1089627 | - | 0 artifacts | |

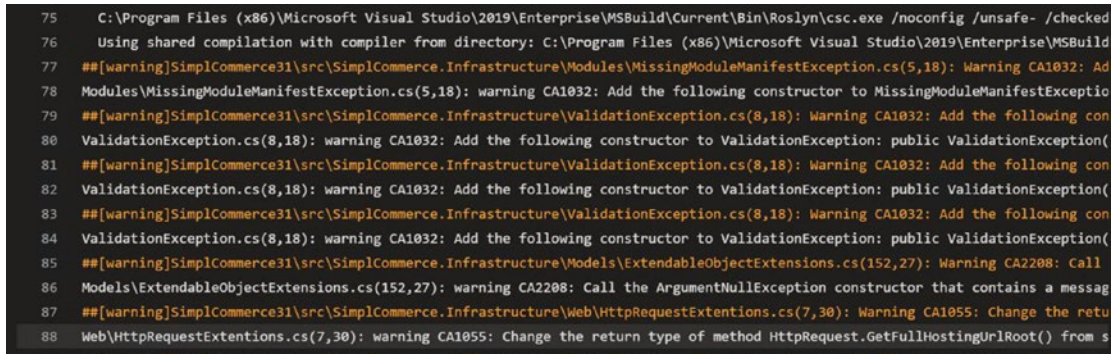
Jobs

| Name | Status | Duration |
|--|--------|----------|
| i Job | Queued | |

17. Click the job item itself, which opens the more detailed view of the running job, showing the different steps in the build process.



18. Wait for the process to complete. Notice there are several warnings visible during the VSBuild stage; these can be ignored for now.



19. After about 5–6 minutes, the job completes successfully.

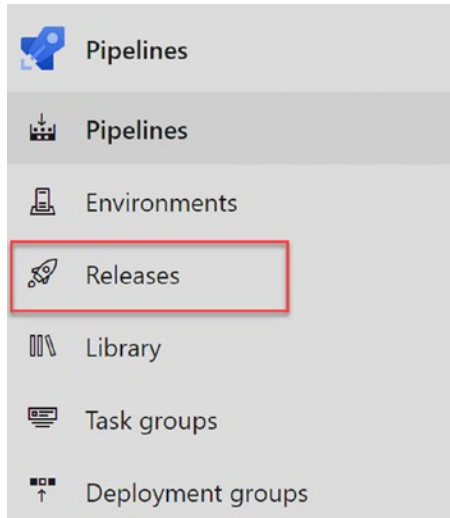
The screenshot shows a build job summary for 'Jobs in run #20200814.1' under the 'LearningAzure' project. The job is marked as successful with a green checkmark. Below the job name, a list of tasks is shown, each with its own status icon and duration.

| Job | Duration |
|--------------------------|----------|
| Job | 3m 51s |
| Initialize job | 8s |
| Checkout LearningAzur... | 6s |
| NuGetToolInstaller | 1s |
| NuGetCommand | 1m 17s |
| VSBuild | 1m 46s |
| Archive Files | 6s |
| VSTest | 16s |
| PublishBuildArtifacts | 5s |
| Post-job: Checkout Le... | <1s |
| Finalize Job | <1s |
| Report build status | <1s |

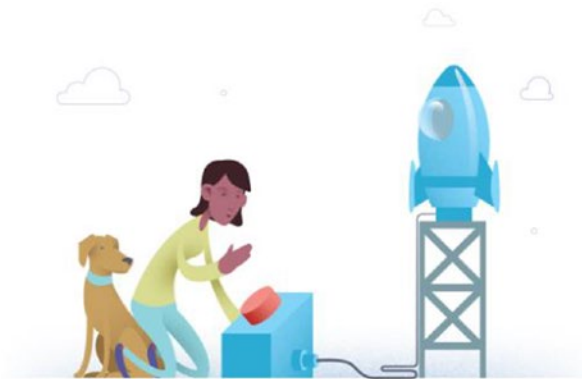
This completes the task in which you set up a build pipeline, based on application source code in Azure DevOps Repos. In the next task, we will continue the process, by creating and running a release pipeline, publishing the code to Azure.

Task 4: Building a release pipeline in Azure DevOps

1. From Azure DevOps ► Pipelines, select **Releases**.



2. Next, select **New pipeline**.

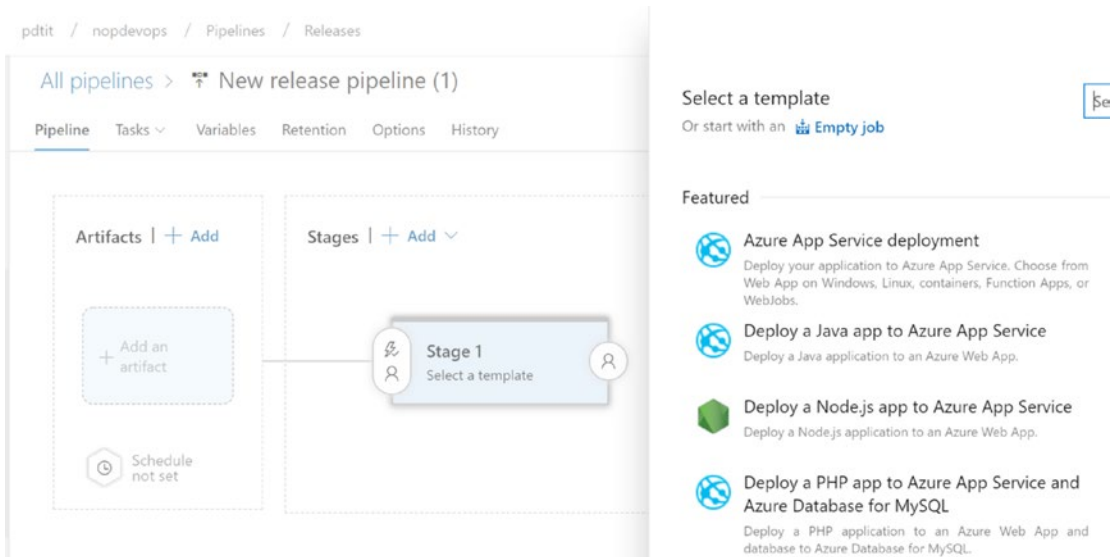


No release pipelines found

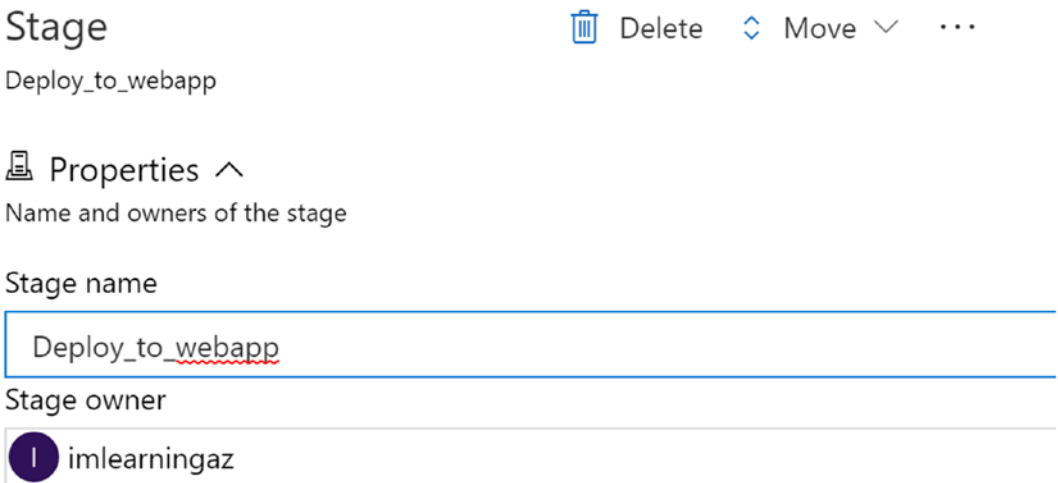
Automate your release process in a few easy steps with a new pipeline

[New pipeline](#)

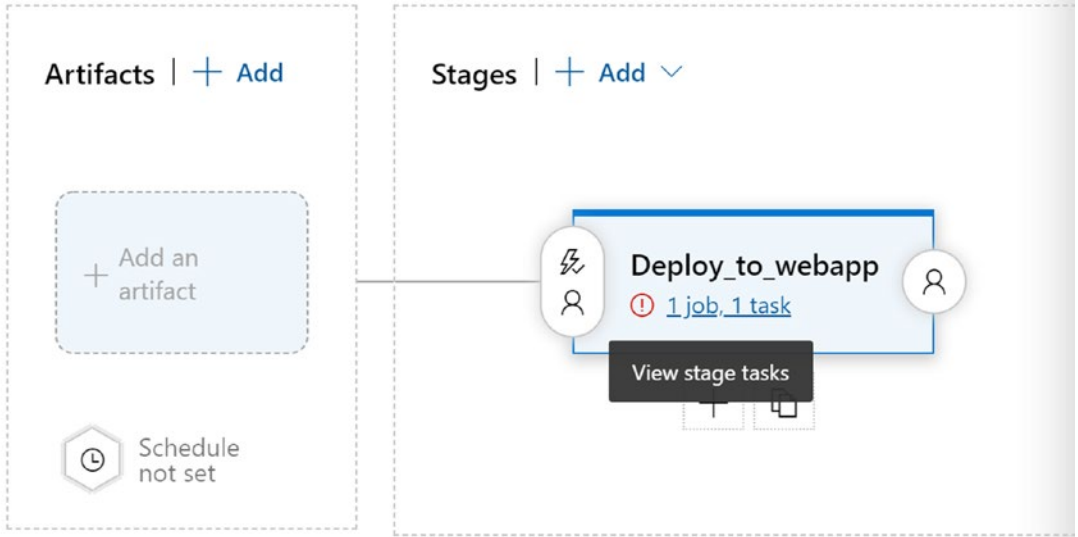
3. This launches the New release pipeline creation wizard.



4. From the **template list**, select **Azure App Service deployment**. Provide a description for the **Stage name**, for example, **Deploy_to_webapp**.



5. **Close the Stage window.**
6. The pipeline now looks like this:

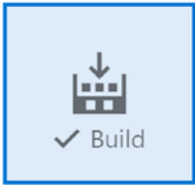
All pipelines >  New release pipeline (1)Pipeline  Tasks  Variables Retention Options History

The screenshot shows the Azure DevOps pipeline editor interface. It is divided into two main sections: 'Artifacts' and 'Stages'. The 'Artifacts' section on the left contains a dashed box with a '+ Add an artifact' button and a 'Schedule not set' indicator. The 'Stages' section on the right contains a dashed box with a '+ Add' button and a 'Deploy_to_webapp' task. The 'Deploy_to_webapp' task is highlighted with a blue border and has a 'View stage tasks' dropdown menu. The task details show '1 job, 1 task' and a warning icon. The 'View stage tasks' dropdown menu is open, showing a list of tasks.

7. Before defining the actual deployment task, let's add the artifact; this is the source package for the actual deployment, which you created during the previous build task. **Click "Add an artifact."**

Add an artifact

Source type



5 more artifact types 

Project * 

Source (build pipeline) * 

 This setting is required.

Add

8. Click the **Source (build pipeline)** drop-down icon, and select **“LearningAzure,”** which is the source build pipeline you created earlier. This will complete some additional parameters.

Source (build pipeline) * ⓘ

LearningAzure

Default version * ⓘ

Latest

Source alias * ⓘ

_LearningAzure

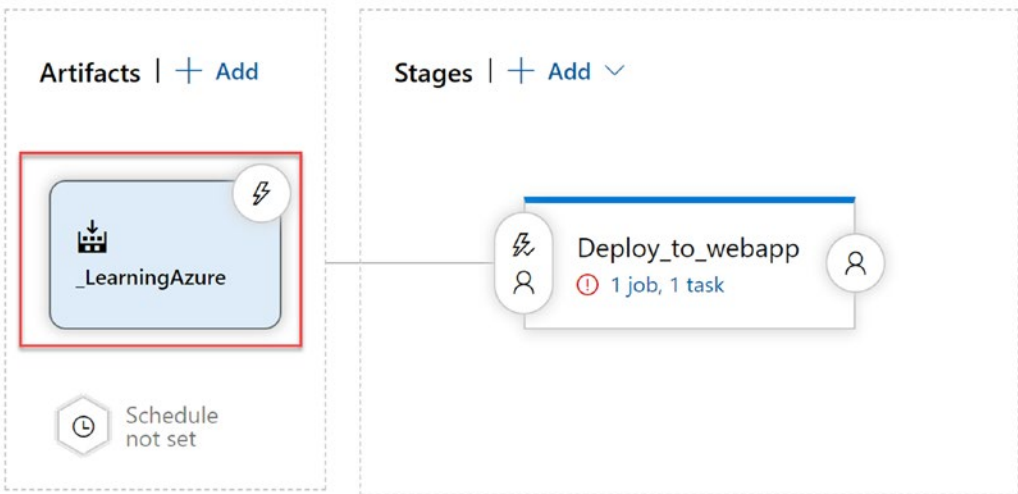
ⓘ The artifacts published by each version will be available for deployment in release pipelines. The latest successful build of **LearningAzure** published the following artifacts: **drop**.

Add

9. **Confirm by clicking “Add.”** The updated pipeline looks like this:

All pipelines > 🚀 New release pipeline

Pipeline ⓘ Tasks ▾ Variables Retention Options History



10. In the **Stages** field, select “**1 job, 1 task**”; this is where you will provide the settings of the Azure Web App environment you will use for the actual deployment.

Stage name

Parameters ⓘ | [Unlink all](#)

Azure subscription * [Manage](#)



ⓘ This setting is required.

App type [Manage](#)

App service name * [Manage](#)



ⓘ This setting is required.

11. Since it’s the first time we integrate Azure DevOps pipelines with Azure itself, you need to authorize this from the Azure subscription topic (since your Azure admin and Azure DevOps admin accounts are the same and have full permissions, this “just works”; in a production environment, you would configure a “Service ConnectionPoint” for this, using a service principal (remember you did something similar for RBAC in the AKS lab?).

Parameters ⓘ

| [Unlink all](#)

Azure subscription *

| [Manage](#)Azure Free - Sponsorship | 1000 4000 0000 0000 d0 ▾

Authorize | ▾

Click Authorize to configure an Azure service connection. A new Azure service principal will be created and added to the Contributor role, having access to all resources in the selected subscription. To restrict the scope of the service principal to a specific resource group, see [connect to Microsoft Azure](#)

App type

Web App on Windows

App service name *



ⓘ This setting is required.


12. Click Authorize, and authenticate using your Azure admin credentials. You will notice the list of App Service names is **empty**. **This makes sense, since we didn't deploy the Azure Web App resource** yet. While Azure Pipelines could do this from an ARM template or Azure PowerShell or CLI, let's do it a bit more manual for now. (Think of the Ops team providing the Azure resources and the Dev team (= you) providing the source code for the web app...)
13. Switch to **the Azure Portal, and create a new resource "web app," using the following parameters for the deployment:**
 - **Resource Group: Create New/FromAzureDevopsRG**
 - **Name: Unique name of your choice**
 - **Publish: Code**
 - **Runtime stack: .NET Core 3.1**

- **Operating System: Windows**
- **Region: Region of choice**

[Home](#) > [New](#) >

Create Web App

[Basics](#) [Monitoring](#) [Tags](#) [Review + create](#)

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. [Learn more](#) 

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Resource Group * ⓘ [Create new](#)

Instance Details

Name * .azurewebsites.net ✓

Publish * Code Docker Container

Runtime stack *

Operating System * Linux Windows

Region *
 ⓘ Not finding your App Service Plan? Try a different region.

14. Accept the defaults for the App Service plan.

App Service Plan

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app.

[Learn more](#) 

Windows Plan (West Europe) * 

ASP-FromDevOpsRG-92b4 (S1) 

[Create new](#)

Sku and size *

Standard S1

100 total ACU, 1.75 GB memory

15. Confirm the creation by clicking Review + create and once more Create. Wait for the deployment to complete.



The screenshot shows the Azure portal interface for an App Service plan. The breadcrumb path is 'Home > Microsoft.Web-WebApp-Portal-4cb7a15-86bd | Overview >'. The app name is 'fromdevopswebapp'. The left sidebar contains navigation options: Overview (selected), Activity log, Access control (IAM), Tags, Diagnose and solve problems, and Security. The main content area shows the following details:

- Resource group (change): FromDevOpsRG
- Status: Running
- Location: West Europe
- Subscription (change): Azure Pass - Sponsorship
- Subscriptions ID: e373a65a-188d-48df-860d-604d07a5790a
- Tags (change): Click here to add tags
- URL: https://fromdevopswebapp.azurewebsites.net
- App Service Plan: pdtcontwebappplan (P1v2: 1)
- FTP/deployment username: No FTP/deployment user set
- FTP hostname: ftp://waaws-prod-am2-331.ft.azurewebsites.windows.net/site/...
- FTPS hostname: ftps://waaws-prod-am2-331.ft.azurewebsites.windows.net/site/...

16. The baseline is ready, so let's switch back to Azure DevOps Pipelines and complete the following settings:

- App type: Web App on Windows
- App service name" <name of the web app you just created>

Stage name

Parameters ⓘ | [Unlink all](#)

Azure subscription * [Manage](#)

ⓘ Scoped to subscription 'Azure Pass - Sponsorship'

This field is linked to 1 setting in 'Deploy Azure App Service'

App type [Unlink](#)

App service name * [Unlink](#)

This field is linked to 1 setting in 'Deploy Azure App Service'

17. Next, click the task “Deploy Azure App Service.”

Run on agent

Run on agent

Deploy Azure App Service
Azure App Service deploy

18. Validate the setting “Package or folder” looks similar to this:

Package or folder * ⓘ

 ...

File Transforms & Variable Substitution Options ▾

Additional Deployment Options ▾

This refers to the webdeploy package you created out of the build pipeline.

19. When done, **click Save** in the top menu of your Azure Pipelines project, and click **OK** for the popup showing the folder (“\”) where to store this information, followed by **Create release**.



20. **Accept the default settings.**

Create a new release



New release pipeline

Pipeline ^

Click on a stage to change its trigger from automated to manual.



Stages for a trigger change from automated to manual.

Artifacts ^

Select the version for the artifact sources for this release

| Source alias | Version | |
|----------------|------------|--|
| _LearningAzure | 20200814.1 | |

Release description

21. **And confirm the creation.**

[All pipelines](#) > New release pipeline

Release **Release-1** has been created

22. Click “Release-X” in the confirmation bar.

↑ New release pipeline > Release-1 ▾

Pipeline Variables History | + Deploy ▾ ⊘ Cancel ↻ Refresh ✎ Edit ▾ ...

Release

Manually triggered
by imlearninggaz
8/15/2020, 1:03 AM

Artifacts

_LearningAzure
20200814.1
 master

Stages

Deploy_to_webapp
⊙ Queued
Waiting in **Azure Pipelines** q...

New release pipeline > Release-1

Pipeline Variables History | + Deploy Cancel Refresh Edit ...

Release

Manually triggered
by imlearningaz
8/15/2020, 1:03 AM

Artifacts

_LearningAzure_20200814.1
master

Stages

Deploy_to_webapp
In progress

3/3 tasks
Deploy Azure App Service
00:02

23. Click the “In progress” status, and wait for this process to initialize.

New release pipeline (1) > Release-1 > Deploy_to_webapp In progress

Pipeline Tasks Variables Logs Tests | Deploy Cancel Refresh Download all logs Edit ...

Deployment process
In progress

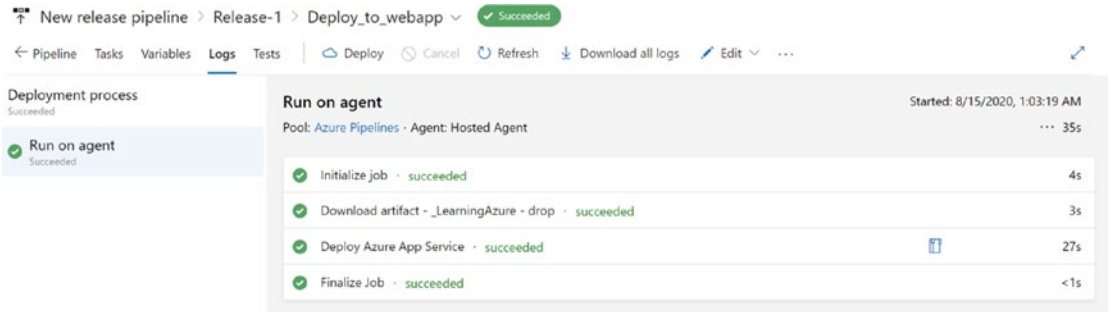
Run on agent
In progress

Run on agent
Pool: Azure Pipelines · Agent: Hosted Agent

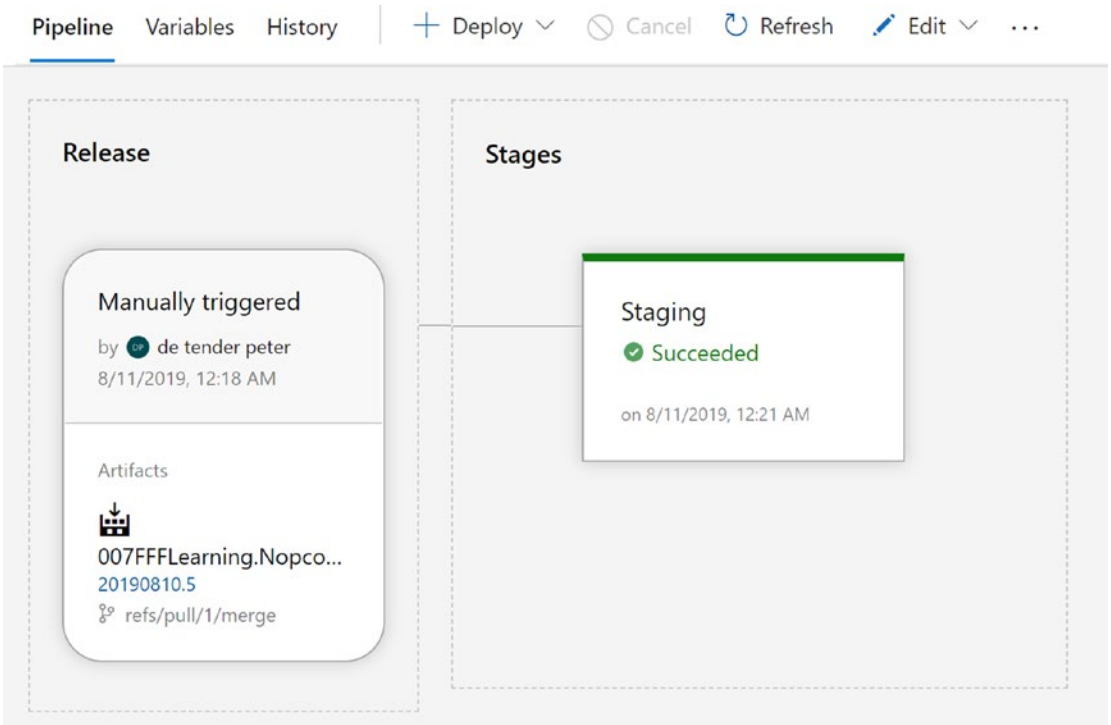
Initialize job

```
Waiting for console output from an agent...
```

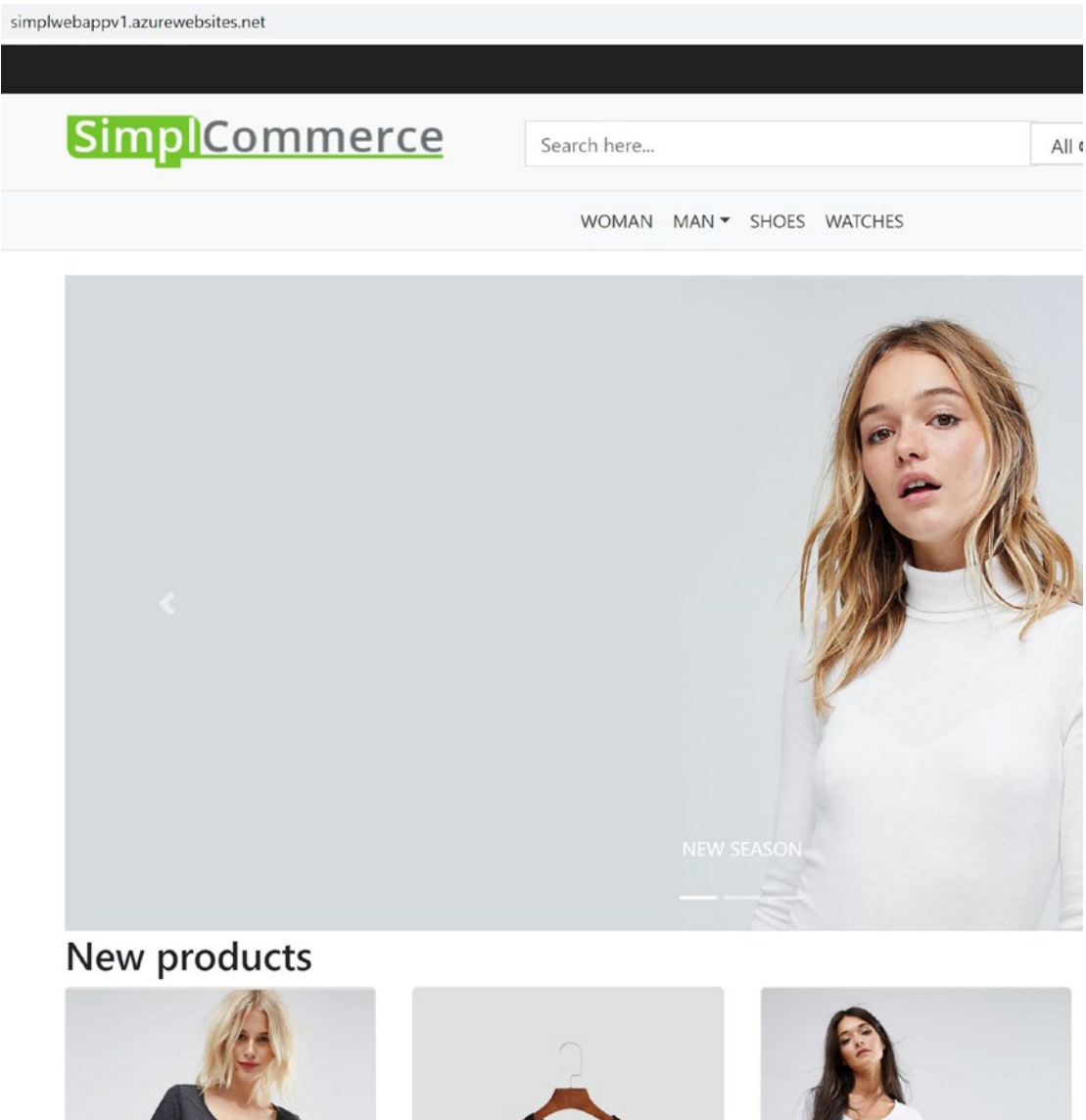
24. This shows the detailed progress for each and every step in the process; wait for the task to complete.



25. The release pipeline shows this Succeeded status as well.



26. From the Azure Portal, browse to the **Azure Web App** you selected in the **release pipeline** as target, and validate it is running as expected.



- 27. **If you should receive the following web page instead of the webshop itself, it typically means there is no database connectivity; validate your SQL Azure database is present, as well as checking if you (still) have the database connection string in the appsettings.json file in the Azure DevOps Repos SimplCommerce31\src\Simplcommerce.webhost\ folder.**

HTTP Error 500.30 - ANCM In-Process Start Failure

Common solutions to this issue:

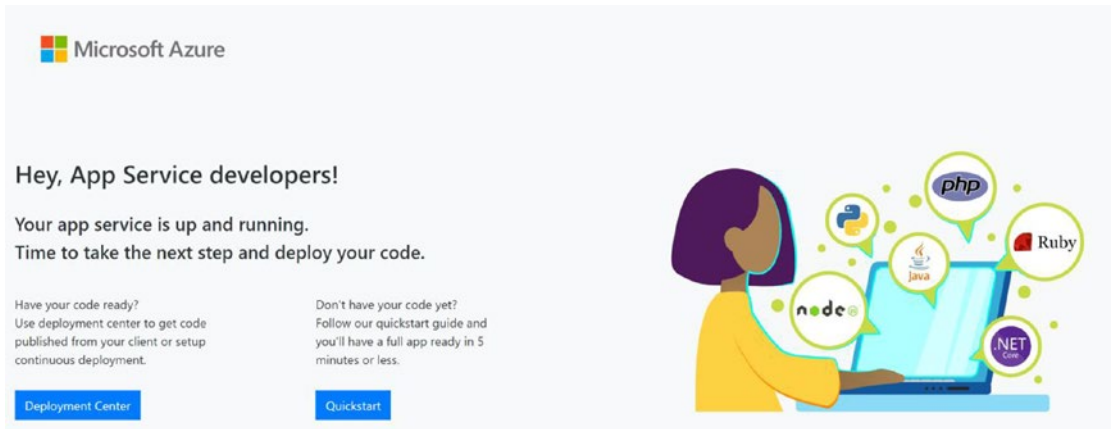
- The application failed to start
- The application started but then stopped
- The application started but threw an exception during startup

Troubleshooting steps:

- Check the system event log for error messages
- Enable logging the application process' stdout messages
- Attach a debugger to the application process and inspect

For more information visit: <https://winwebappfromdevops.scm.azurewebsites.net/detectors?type=tools&name=eventviewer> and <https://go.microsoft.com/fwlink/?LinkId=2028265>

28. **If you receive the following web page instead of the webshop itself or the HTTP Error 500.30, it means you were a bit too fast 😊; waiting for a few seconds and refreshing the website typically fixes this. You could also try to stop and start the web app again to force the publishing.**



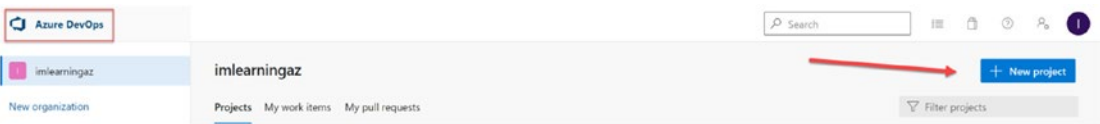
This completes the task in which you created a release pipeline, based on a previous build pipeline configuration, allowing you to publish an Azure Web App.

Task 5: Creating and pushing a Docker container to ACR

In one of the previous labs, you learned the basics of Docker commands and how to push an existing Docker Hub container image to Azure Container Registry. Most probably at that time, you were wondering how to create a Docker container yourself, right? Since this felt a bit more “DevOps” in character, I decided to keep it for the Azure

DevOps module. So here we are, where I will guide you through creating a Dockerized container image, based on the webshop source code, and pushing this container image to Azure Container Registry, all done by Azure DevOps.

1. Let us start with creating a new Azure DevOps Project (this is not really required out of Azure DevOps itself, but just feels more organized to me), **by clicking “Azure DevOps” in the upper-left corner in the Azure DevOps portal and clicking “+ New project.”**



2. **Provide a project name**, keep visibility to private, and confirm by clicking **Create**.

Create new project ✕

Project name *

 ✓

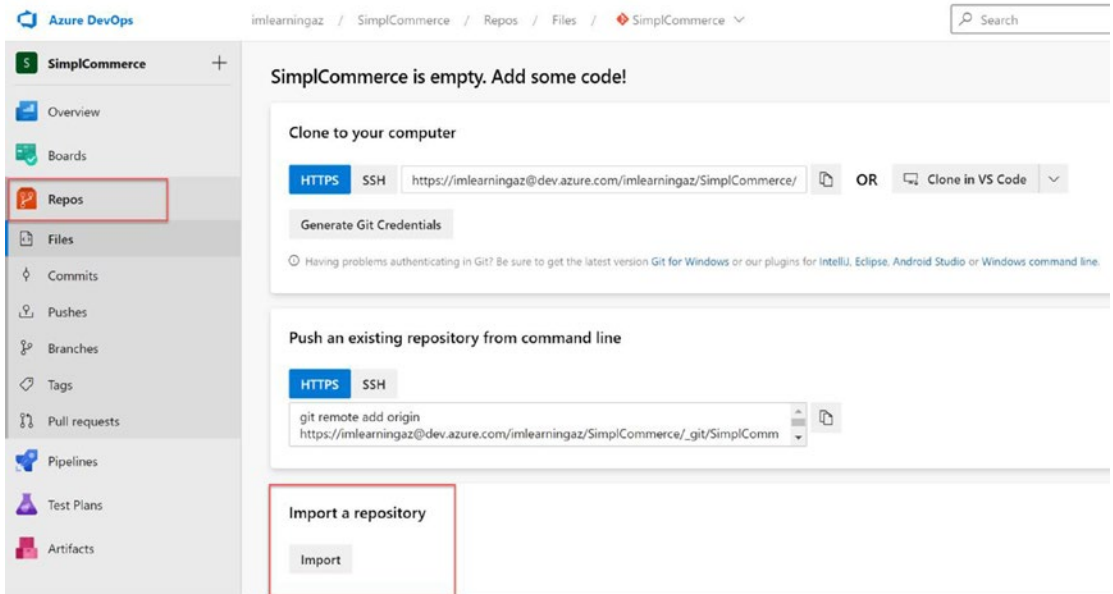
Description

Visibility

| | |
|---|--|
| <input type="radio"/> Public Anyone on the internet can view the project. Certain features like TFVC are not supported. | <input checked="" type="radio"/> Private Only people you give access to will be able to view this project. |
|---|--|

∨ **Advanced**

- Once the project got created, select Repos, where you will “import a repository.”




- Provide the following URL:**

<https://github.com/simplcommerce/simplcommerce.git> (know this repo is managed by SimplCommerce itself, not by me; since it is getting continuously updated, I thought it was more safe to provide this one, to make sure the container build steps keep working)

Import a Git repository ✕

Repository type

 Git ▾

Clone URL *

Requires Authentication

5. **The import process starts.**

On its way!

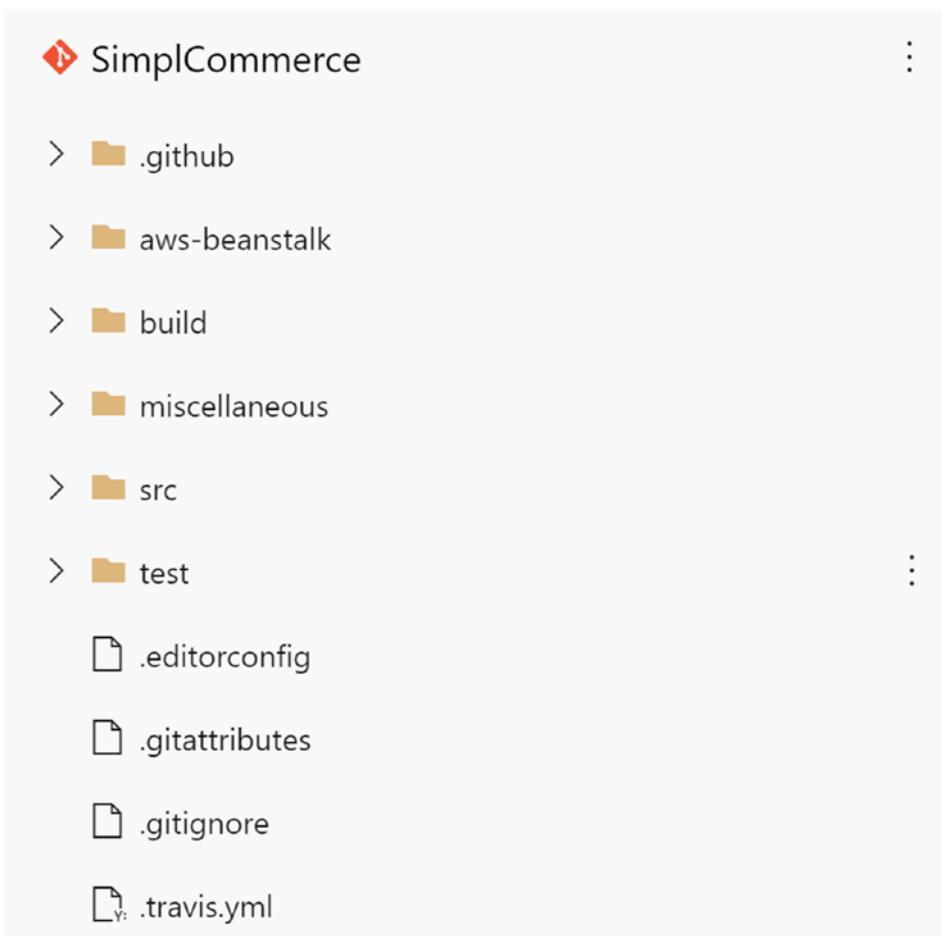


 Processing request

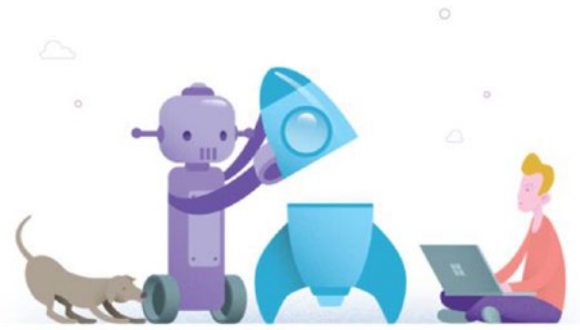
Importing <https://github.com/simplcommerce/simplcommerce.git>

We'll send you a notification when it's ready. For now, you can work on some other project or just take a moment to sit back, relax and enjoy your day.

6. **Once the import succeeded, the Repos structure looks like this:**



7. **Next, create a new (build) pipeline, by selecting Pipelines ► Create Pipeline.**



Create your first Pipeline


Automate your build and release processes using our wizard, and go from code to cloud-hosted within minutes.




- 8. This launches the Create Pipeline wizard. In the “Where is your code?” step, select Azure Repos Git.**

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.

9. **Click Next. Select “SimplCommerce” as the repo to use.**

New pipeline

Select a repository

🔍 Filter by keywords
SimplCommerce ▾ ×


SimplCommerce

- 10. **This builds up an azure-pipelines.yml file, looking similar to this one:**

New pipeline

Review your pipeline YAML

◆ SimplCommerce / azure-pipelines.yml

```
1 # ASP.NET Core
2 # Build and test ASP.NET Core projects targeting .NET Core.
3 # Add steps that run tests, create a NuGet package, deploy, and more:
4 # https://docs.microsoft.com/azure/devops/pipelines/languages/dotnet-core
5
6 trigger:
7   - master
8
9 jobs:
10  - job: Linux
11    pool:
12      vmImage: 'ubuntu-18.04'
13    steps:
14      - script: dotnet build ./SimplCommerce.sln
15      - displayName: 'dotnet build'
```

- 11. Notice it offers different jobs, for different Operating System Build Agents (Mac, Linux, Windows); this is because the application is developed in dotnetcore, which is supported to run on each of those platforms.
- 12. **This creates the new job.**

← SimplCommerce

Runs Branches Analytics

| Description | Stages |
|---|--------|
| #20200816.1 Technical Change Request in OrderService #926 (#927) Manually triggered for master b90189d | |

- 13. **Select the job, which shows more details for the running job(s).**

#20200816.1 Technical Change Request in OrderService #926 (#927) on SimplCommerce



Summary Tests

Manually run by  imlearningaz

Repository and version


 SimplCommerce
📁 master 🔄 b90189d


Time started and elapsed

 Today at 4:24 PM
 15m 55s


Jobs

Name

 Linux

 macOS

 Windows

 LinuxRelease

14. **You could select any of the jobs to get even more details about the build process itself;** since you already did that in earlier tasks, I'll skip that for now.
15. Return to Azure DevOps Pipelines, and create yet another one. When you **are asked where the source code is**, select **“Azure Repos Git.”**

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.

- Next, you need to **select the repository to use. Here, select "SimplCommerce."**

New pipeline

Select a repository


SimplCommerce ▾ ✕





- Next, in the **Configure your pipeline** step, select **“Docker – Build and push an image to Azure Container Registry.”**


New pipeline


Configure your pipeline

 Docker
Build a Docker image

 Docker
Build and push an image to Azure Container Registry

 Deploy to Azure Kubernetes Service
Build and push image to Azure Container Registry; Deploy to Azure Kubernetes Service

 Deploy to Kubernetes - Review app with Azure DevSpaces
Build and push image to Azure Container Registry; Deploy to Azure Kubernetes Services and setup Review App with Azure DevSpaces

 ASP.NET
Build and test ASP.NET projects.

- Next, select your **Azure subscription and confirm by clicking “Continue”**; this will prompt you for your Azure admin credentials.

Docker



Build and push an image to Azure Container Registry

Select an Azure subscription

- Azure Pass - Sponsorship
e373a65a-188d-48df-860d-604d07a5790a

19. **Once authenticated, select your Azure Container Registry from the list, and update the container name to “devopssimpl[suffix].”**

Docker



Build and push an image to Azure Container Registry

Container registry

PDTACR



Image Name

devopssimplpdt


Dockerfile

\$(Build.SourcesDirectory)/Dockerfile

20. **Click “Validate and Configure,” which produces an azure-pipelines.yml file, looking like the following screenshot:**

New pipeline

Review your pipeline YAML

SimplCommerce / azure-pipelines-1.yml * 

```

1  # Docker
2  # Build and push an image to Azure Container Registry
3  # https://docs.microsoft.com/azure/devops/pipelines/languages/docker
4
5  trigger:
6  - master
7
8  resources:
9  - repo: self
10
11  variables:
12  - # Container registry service connection established during pipeline creation
13  - dockerRegistryServiceConnection: '9ae90406-f08f-4c17-ae1-c01633562d46'
14  - imageRepository: 'devopssimplpdt'
15  - containerRegistry: 'pdtacr.azurecr.io'
16  - dockerfilePath: '$(Build.SourcesDirectory)/Dockerfile'
17  - tag: '$(Build.BuildId)'
18  -
19  - # Agent VM image name
20  - vmImageName: 'ubuntu-latest'
21

```

21. **Confirm by clicking “Save and run.”**

Save and run



Saving will commit azure-pipelines-1.yml to the repository.

Commit message

DockerPush_from_DevOps

Optional extended description

Add an optional description...

- Commit directly to the master branch
- Create a new branch for this commit

22. Provide a descriptive name in the Commit message field, and confirm by clicking **Save and run** again, which **creates the pipeline**.

#20200816.1 DockerPush_from_DevOps
on SimplCommerce (1)

Cancel

Summary

Triggered by **imlearningaz** [View 250 changes](#)

| Repository and version | Time started and elapsed | Related | Tests and coverage |
|---------------------------------|--------------------------|-----------------------------|-----------------------------|
| SimplCommerce master 754c431 | Just now | 0 work items 0 artifacts | Get started |

Jobs

| Name | Status | Duration |
|--------------|--------|----------|
| Build | Queued | |

23. Click the “Build” job, to open more details about the job. Notice how each step in the Dockerfile gets processed.

The screenshot shows the details of a build job in Azure DevOps. On the left, a sidebar lists the job steps: 'Build and push stage', 'Build' (33s), 'Initialize job' (2s), 'Checkout SimplComme...' (3s), 'Build and push an ima...' (28s), and 'Post-job: Checkout Simpl...'. The main area displays the terminal output for the 'Build and push an image to container registry' job, showing the execution of Docker commands and the successful tagging and pushing of the container image.

```

24 b7b0f8dc0c5c: Waiting
25 e4af0a0b092f: Verifying Checksum
26 e4af0a0b092f: Download complete
27 c958d65b3090: Verifying Checksum
28 c958d65b3090: Download complete
29 7b9b07889c2a: Verifying Checksum
30 7b9b07889c2a: Download complete
31 d6ff36c9ec48: Verifying Checksum
32 d6ff36c9ec48: Download complete
33 80931cf68816: Verifying Checksum
34 80931cf68816: Download complete
35 b7b0f8dc0c5c: Verifying Checksum
36 b7b0f8dc0c5c: Download complete
37 4f2c2524f197: Verifying Checksum
38 4f2c2524f197: Download complete
39 d6ff36c9ec48: Pull complete
40 c958d65b3090: Pull complete
41 e4af0a0b092f: Pull complete
42 80931cf68816: Pull complete
43 7b9b07889c2a: Pull complete
44 4f2c2524f197: Pull complete
45 b7b0f8dc0c5c: Pull complete
46 Digest: sha256:4805a0ec2e1acae2be9554679acca0ec368aaf50cabdd028c0aa3cbff34a751
47 Status: Downloaded newer image for mcr.microsoft.com/dotnet/core/sdk:3.1
48 ---> 9ab567a29502
49 Step 2/31 : WORKDIR /app
50 ---> Running in 19043062d46a
51 Removing intermediate container 19043062d46a
52 ---> 69494c05d097
53 Step 3/31 : COPY . ./
54 ---> c3f840f95814
55 Step 4/31 : RUN sed -i 's:<PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="3.1.0" />:<PackageReferenc
56 ---> Running in 8914c3c28d46
57 Removing intermediate container 8914c3c28d46
58 ---> e0132ad7303e
59 Step 5/31 : RUN sed -i 's:/UseSqlServer/UseNpgsql/' src/SimplCommerce.WebHost/Program.cs
60 ---> Running in aad76839f320
  
```

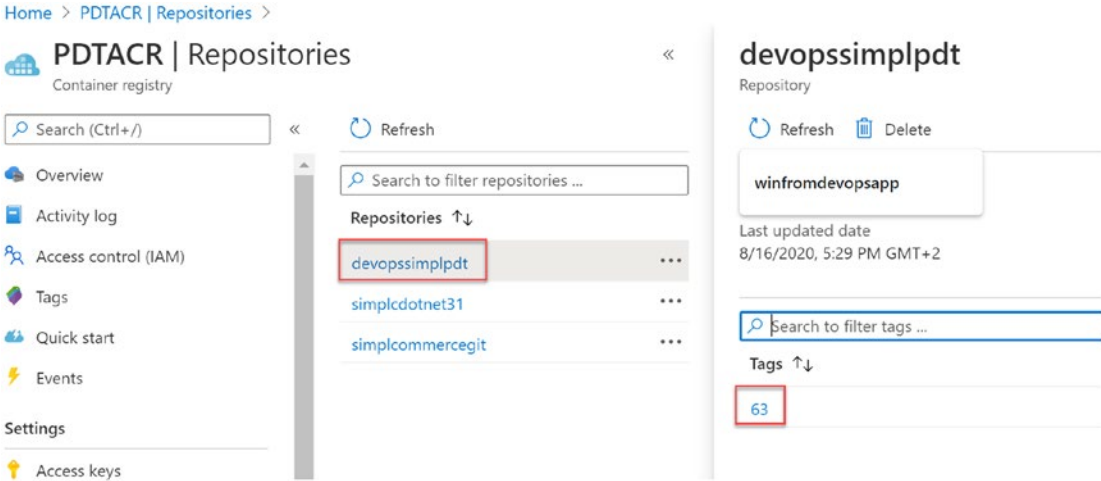
24. If you follow along in the container build process, you will notice that all the way at the end how the name gets tagged to the container, followed by the Docker Push command for this container image.

The screenshot shows the final steps of the container build process in a terminal window. The output indicates that the container image has been successfully built and tagged, and then pushed to the container registry.

```

1916 Removing intermediate container 1d498b02bad0
1917 ---> f51b4713e60a
1918 Successfully built f51b4713e60a
1919 Successfully tagged ***/devopssimplpdt:63
1920 /usr/bin/docker images
1921 /usr/bin/docker push ***/devopssimplpdt:63
  
```

25. **Validating this process from the Azure Portal itself shows the successful push as well.**

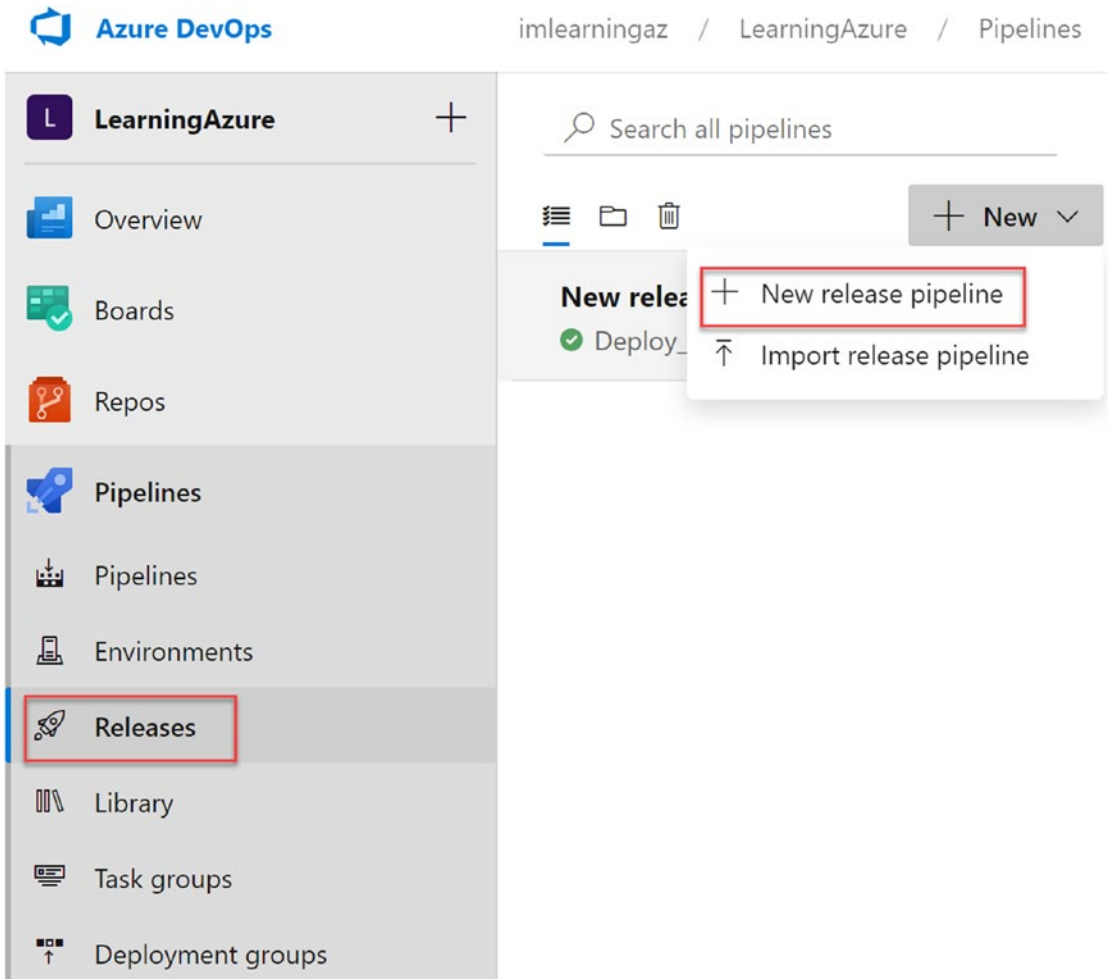


This completes the task in which you learned how to containerize an application using Docker build pipelines.

Task 6: Creating a release pipeline for Docker containers from ACR

Similar to the previous release pipeline from source code in GitHub to a published Azure Web App, we can use the same concept to create a release pipeline, based on a Docker container in Azure Container Registry. This is similar to the manual task you ran in Lab 4 earlier.

1. From **Azure DevOps**, select **Pipelines** > **Releases** > **New release pipeline**.

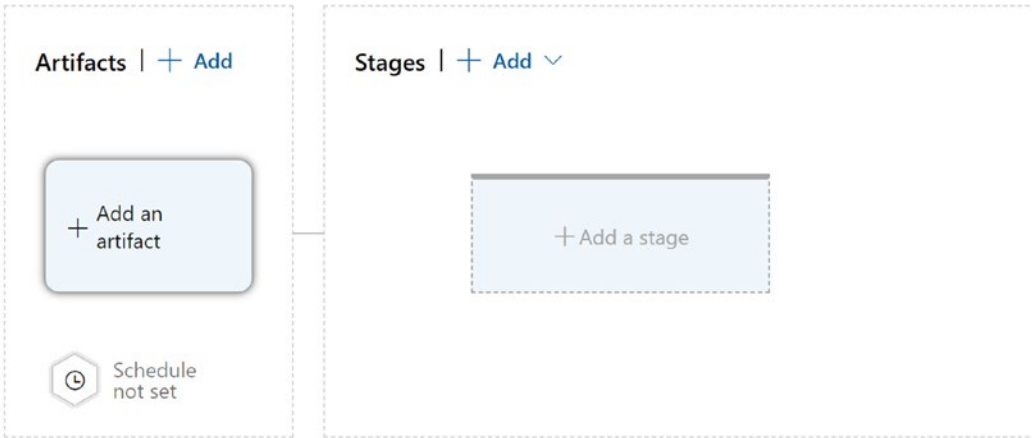


2. **When the template window appears, close it, and select “Add an artifact” first.**

All pipelines > New release pipeline (2)

Pipeline Tasks Variables Retention Options History

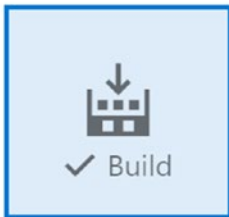
You cannot save a release pipeline that contains zero stages.



3. From the Add an artifact blade, click “5 more artifact types,” to extend the list of artifacts to choose from.

Add an artifact

Source type

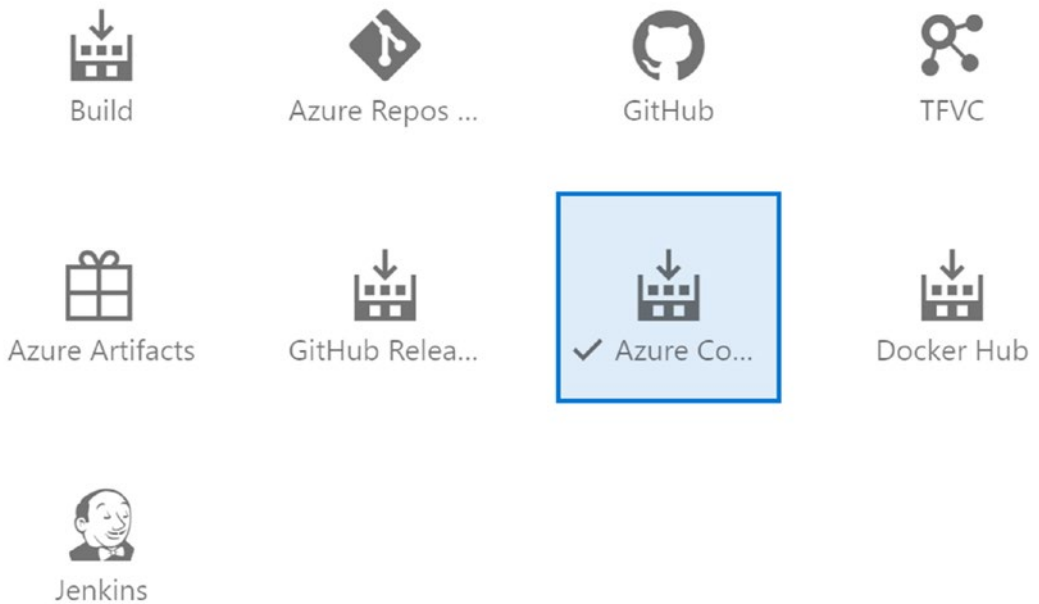


5 more artifact types

4. Select “Azure Container Registry.”

Add an artifact

Source type




5. Complete the parameters according to the existing resources in your Azure subscription, reusing the resources from previous lab exercises (Azure Container Registry, repository, etc.).

Service connection * | [Manage](#) 

Azure Pass - Sponsorship (e373a65a-188d-48df-860d-604d07a5790a)  

Resource Group * 

PDT-containersRG 

Azure Container Registry * 

PDTACR 

Repository * 

simplcdotnet31 

Default version * 

Latest 

Source alias * 

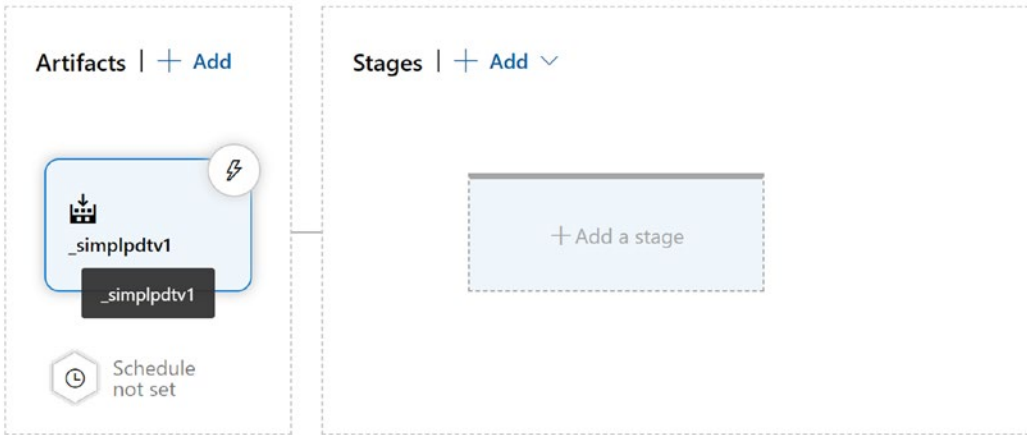
_simplcdotnet31

Add

6. Confirm the artifact selection, by **clicking Add**.
7. Your artifact will be completed.

Pipeline Tasks Variables Retention Options History

i You cannot save a release pipeline that contains zero stages.



8. Next, click **Stages** ► **Add a stage**, and select **Azure App Service deployment**.

Select a template

Or start with an **Empty job**

Others



Azure App Service deployment with continuous monitoring

Deploy your Web applications to Azure App Service and enable continuous monitoring using Application Insights.

Featured



Azure App Service deployment

Deploy your application to Azure App Service. Choose from Web App on Windows, Linux, containers, Function Apps, or WebJobs.

Apply

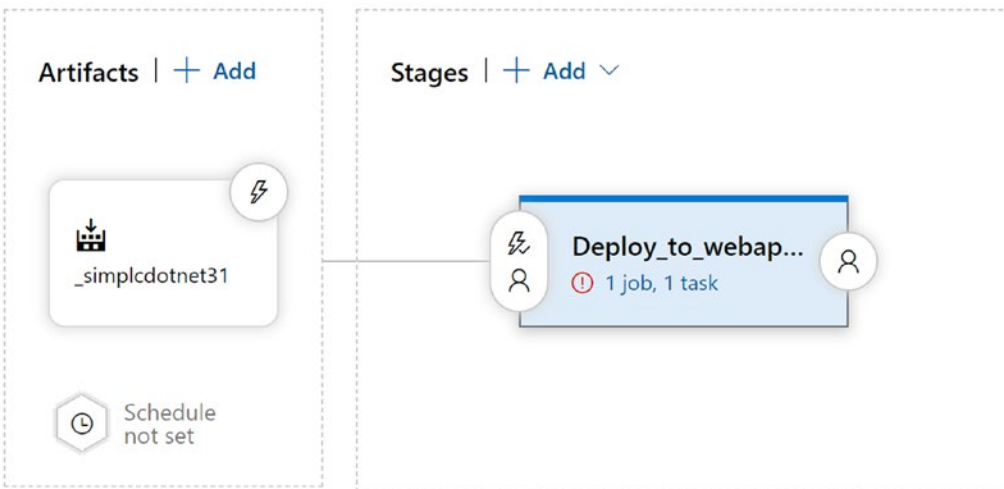
- 9. **Confirm with Apply; provide a descriptive name for the stage, for example, “Deploy to webapp for containers.”**

The screenshot shows a 'Stage' configuration popup. At the top, it says 'Stage' with a close button (X) in the top right. Below that are 'Delete' (trash icon), 'Move' (diamond icon), and a menu icon (three dots). The stage name is 'Deploy_to_webapp_for_containers'. Under 'Properties', there is a 'Stage name' input field containing 'Deploy_to_webapp_for_containers' and a 'Stage owner' dropdown menu showing 'imlearningaz' with a close button (X).

- 10. **Close the Stage popup, followed by selecting the “1 job, 1 task” item in the pipeline view.**

All pipelines > 🚀 New release pipeline (1)

Pipeline ⚠️ Tasks ▾ Variables Retention Options History



11. **Provide the required parameter for your Azure subscription, and specify “Web App for Containers (Linux)” for App type.**

Stage name

Deploy_to_webapp_for_containers

Parameters ⓘ | [Unlink all](#)

Azure subscription * [Manage](#)

Azure Pass - Sponsorship (e373a65a-188d-48df-860d-604d07a5790a) ▼



ⓘ Scoped to subscription 'Azure Pass - Sponsorship'

App type [Unlink](#)

Web App for Containers (Linux) ▼

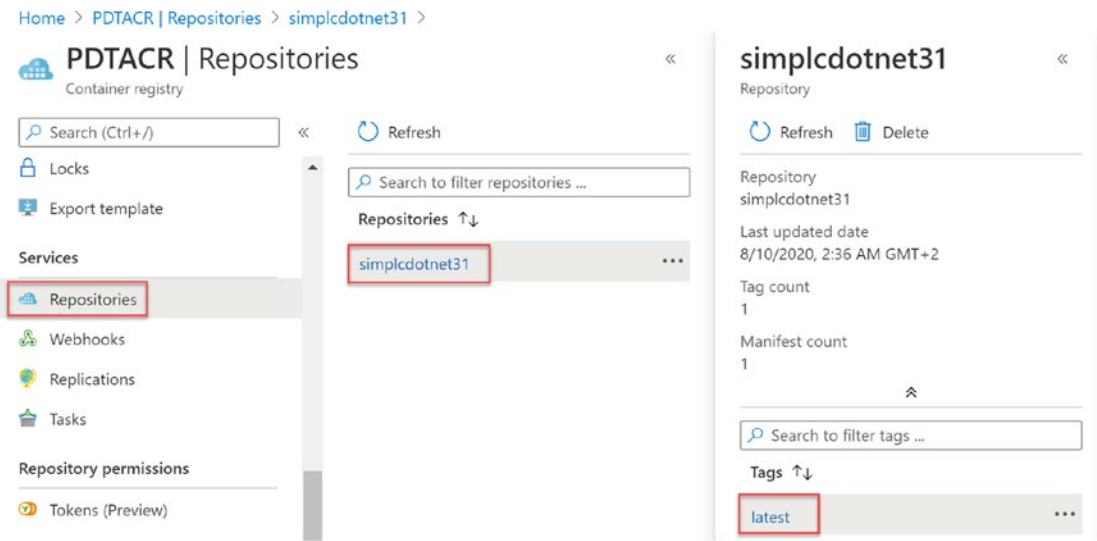
12. **Complete the additional parameters for Azure Container Registry and image. Note you have to provide these values yourself; they are not pulled from a list box like in the previous task when publishing the web app. These are the screenshots from the Azure Portal to help you in finding this information:**

Home > PDTACR Container registry


Search (Ctrl+) | Move | Delete | Update

| | | | |
|-------------------------|--|--------------------|-----------------------------|
| Resource group (change) | : PDT-containersRG | Login server | : pdtacr.azurecr.io |
| Location | : West Europe | Creation date | : 8/10/2020, 12:36 AM GMT+2 |
| Subscription (change) | : Azure Pass - Sponsorship | SKU | : Basic |
| Subscription ID | : e373a65a-188d-48df-860d-604d07a5790a | Provisioning state | : Succeeded |


Overview | Activity log | Access control (IAM) | Tags | Quick start



13. The deployment parameters should look similar to my screenshot:

App service name * 

Registry or Namespace * 

Repository * 

Startup command 

14. **Click “Save,” and confirm the popup as OK, followed by “Create release.”**

 Save  Create release

15. **Validate the settings for this new release pipeline.**

Create a new release ×

New release pipeline (1)

Pipeline ^

Click on a stage to change its trigger from automated to manual.

 Deploy_to_we

Stages for a trigger change from automated to manual. 

Artifacts ^

Select the version for the artifact sources for this release

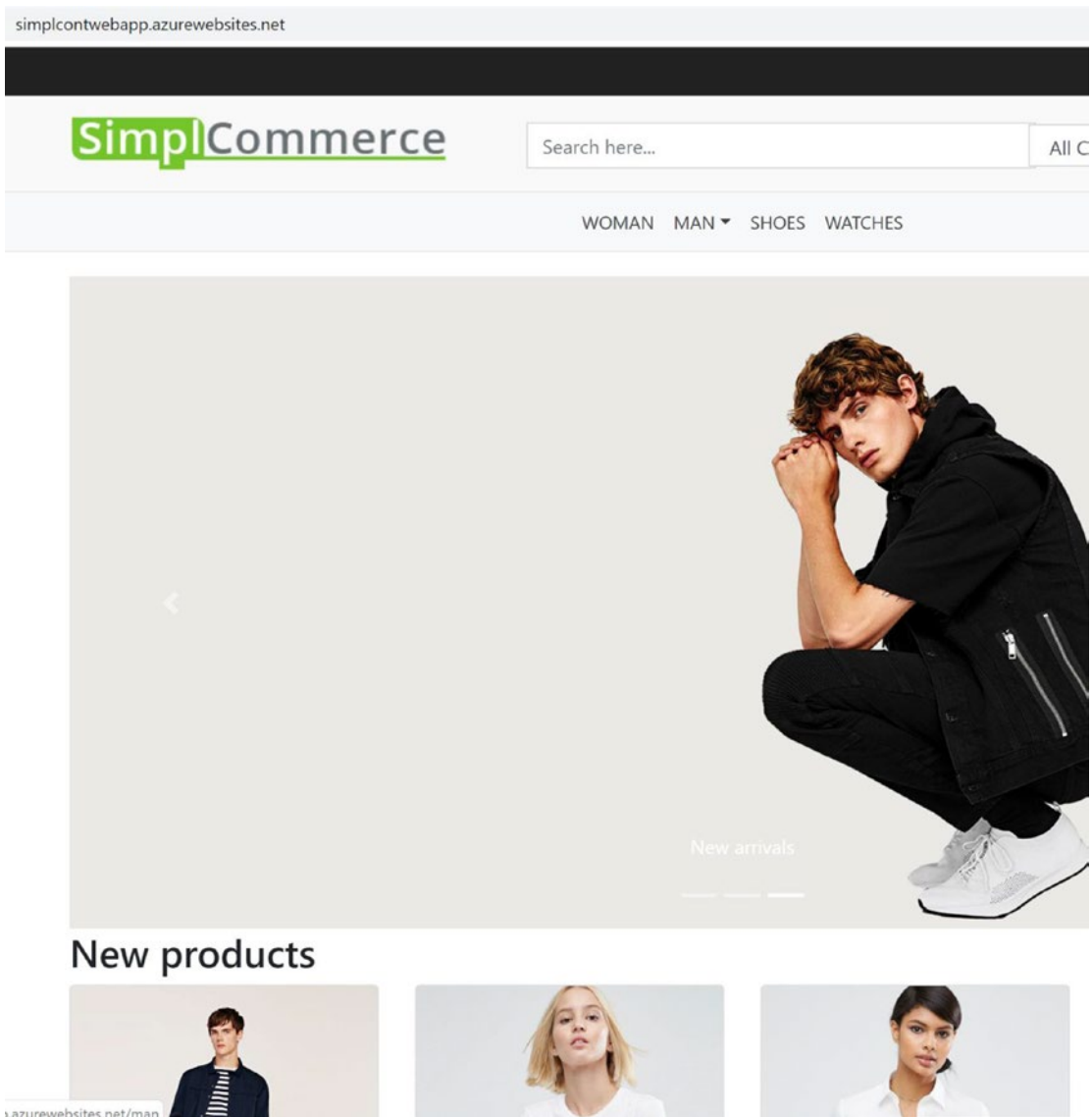
| Source alias | Version |
|-----------------|---|
| _simplcdotnet31 | latest ▼ |

16. **And confirm the deployment by pushing the Create button. This creates Release-1.**

- Once the task is complete, you can see its overall status from the Pipeline window.

The screenshot displays the Azure DevOps Pipeline interface. At the top, the breadcrumb navigation shows 'New release pipeline (1) > Release-1'. Below this is a toolbar with options: 'Pipeline' (selected), 'Variables', 'History', '+ Deploy', 'Cancel', 'Refresh', 'Edit', and a menu icon. The main content area is divided into two panels: 'Release' and 'Stages'. The 'Release' panel shows a 'Manually triggered' release by user 'imlearningaz' on 8/15/2020 at 1:28 AM. Underneath, it lists an artifact named '_simplcdotnet31 latest'. The 'Stages' panel shows a single stage named 'Deploy_to_webapp_fr' which has a status of 'Succeeded' (indicated by a green checkmark) and was completed on 8/15/2020 at 1:29 AM. A line connects the release to the stage.

- Check back in Azure Web Apps if your app is running successfully, by connecting to the Azure Web App URL for this Azure resource.

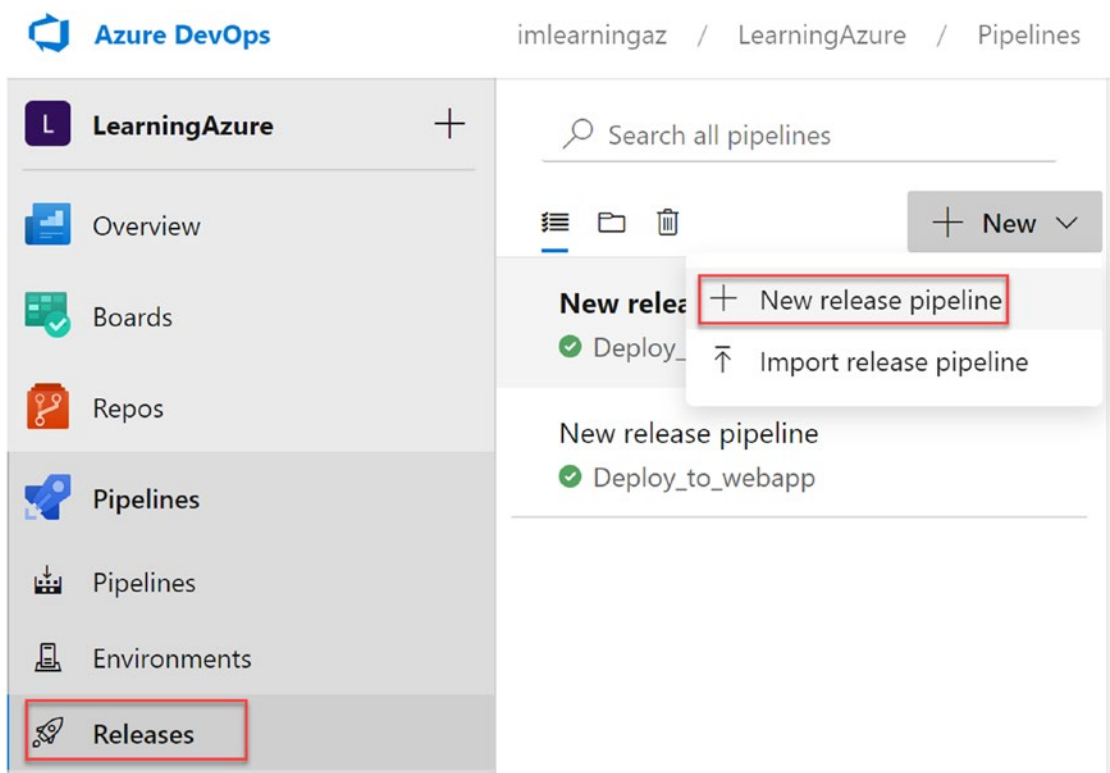


This completes the task in which you created a new Azure Pipelines release, deploying an Azure Web App for Containers, relying on a repository in Azure Container Registry.

Task 7: Creating an Azure DevOps pipeline to deploy an ACR container to Azure Kubernetes Service (AKS)

In this scenario, you will create yet another Azure release pipeline, this time pushing a container from ACR into the earlier deployed Azure Kubernetes Service cluster.

1. From **Azure DevOps**, select **Pipelines** ► **Releases** ► **New release pipeline**.

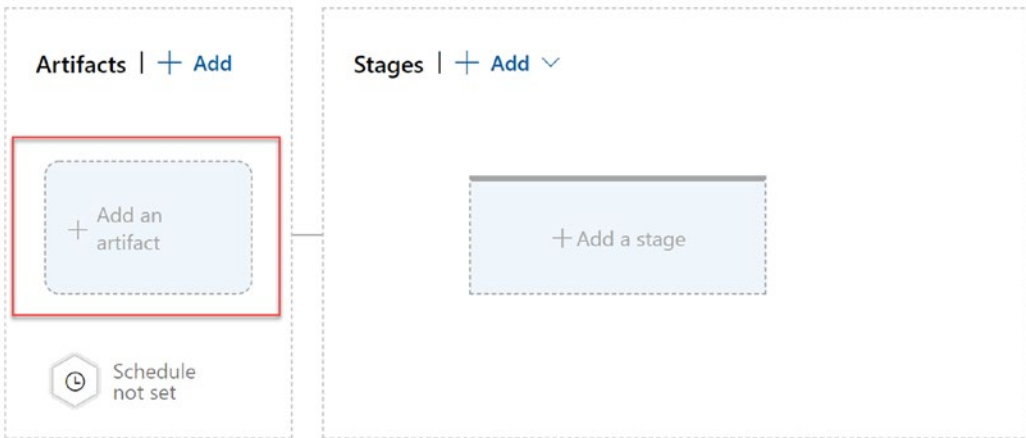


2. Close the appearing **template window**, and return to **Artifacts**; click **“Add an artifact.”**

All pipelines > New release pipeline (2)

Pipeline Tasks Variables Retention Options History

You cannot save a release pipeline that contains zero stages.



3. Repeat the steps from the previous task, selecting **Azure Container Registry** as source and selecting the **ACR and container repository you want to use** for this deployment. I show you my settings as illustration:



Artifact

 Delete ...

AzureContainerRepository - _simplcdotnet31

Service connection | [Manage](#) 

Azure Pass - Sponsorship (e373a65a-188d-48df-860d-604d07a5790a)

Resource Group * 

PDT-containersRG

Azure Container Registry * 

PDTACR

Repository * 

simplcdotnet31

Default version * 

Latest 

Source alias * 


_simplcdotnet31

4. Next, select **“Add a stage,”** and select the **Deploy to a Kubernetes cluster** template.

Select a template

Or start with an  [Empty job](#)

Featured

 **Deploy to a Kubernetes cluster**
Deploy, configure, update your containerized applications to a Kubernetes cluster. [Apply](#)

Others

5. Confirm by clicking **Apply**; provide a descriptive name for the stage, for example, **Deploy_to_AKS**.



Stage

 Delete  Move  ...

Deploy_to_AKS

Properties

Name and owners of the stage

Stage name

Stage owner


6. Close the Stage popup, which returns you to the Release Pipeline window. Click “1 job, 1 task” under Stages.

All pipelines >  New release pipeline (3)Pipeline **Tasks** ▾ Variables Retention Options History

Stage 2

Deployment process

Agent job

 Run on agent

kubectl

 Deploy to Kubernetes

7. Select “**kubectl**,” and provide the necessary settings and parameters of the AKS cluster you deployed in a previous lab, knowing you **only need to provide the Kubernetes service connection** name.

Kubectl  View YAML  Remove Task version ▾

Display name *

Kubernetes service connection  | [Manage](#)  ▾   NewNamespace 

- To create this one, click “+ New” next to it, which opens the New service connection blade; your Azure subscription will get resolved, as well as asking you for your Azure credentials. After successful logon, you can complete the Kubernetes cluster information and namespace, similar to what it looks like in my setup:**

New service connection ✕

Authentication method

KubeConfig

Service Account

Azure Subscription

Azure Subscription

Azure Pass - Sponsorship (e373a65a-188d-48df-860d-604d07... ▾)

Cluster

AKSCluster (AKSNativeRG) ▾

Namespace

default ▾

Use cluster admin credentials

Details

Service connection name

Azure_AKS

Description (optional)

Security



Grant access permission to all pipelines


[Learn more](#)

[Troubleshoot](#)

[Save](#)



9. **Click Save; the pipeline definition shows the Kubernetes service connection now. Since you already defined the namespace in the service connection settings, you can leave that field blank here.**

Kubectl ⓘ  View YAML  Remove

 Task version ▼

Display name *

Kubernetes service connection ⓘ | [Manage](#) ↗

▼   New

Namespace ⓘ

10. Confirm the settings using Save and Release.

 Save  Create release

11. Click Create release from the pipeline confirmation window.

Create a new release



New release pipeline (2)

Pipeline ^

Click on a stage to change its trigger from automated to manual.

Deploy_to_AK

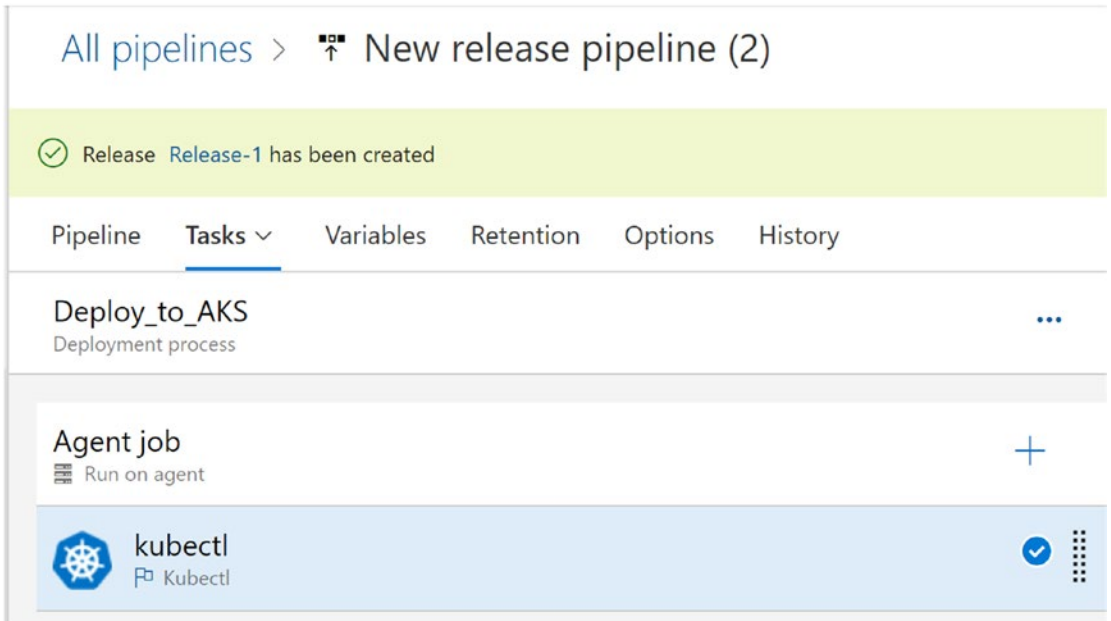
Stages for a trigger change from automated to manual.

Artifacts ^

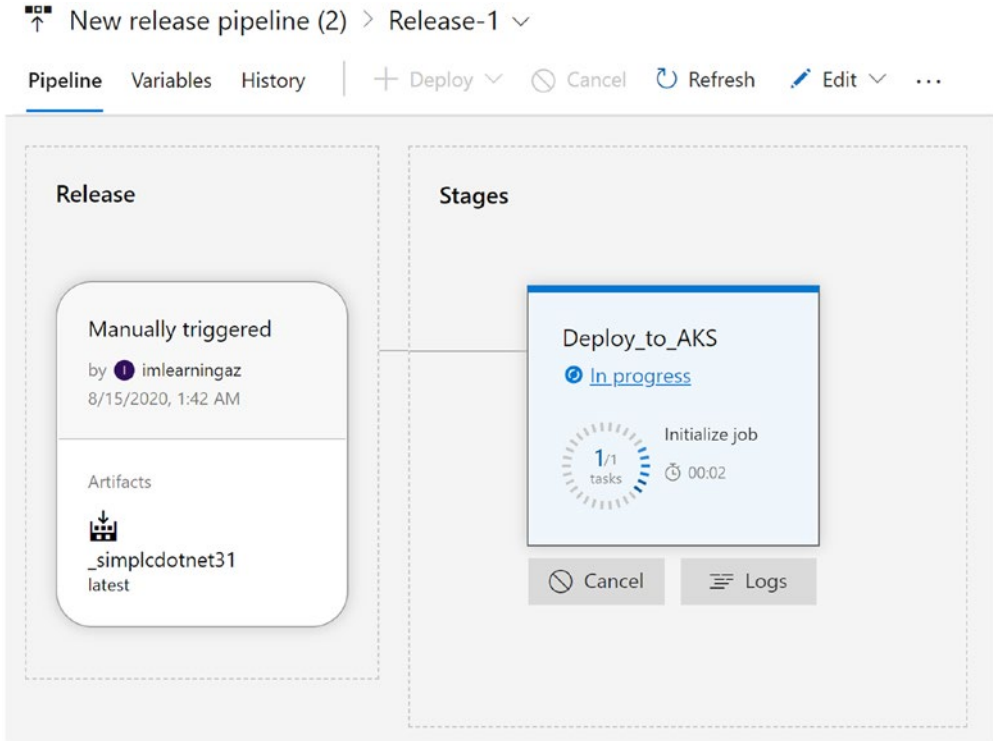
Select the version for the artifact sources for this release

| Source alias | Version | |
|-----------------|---------|--|
| _simplcdotnet31 | latest | |

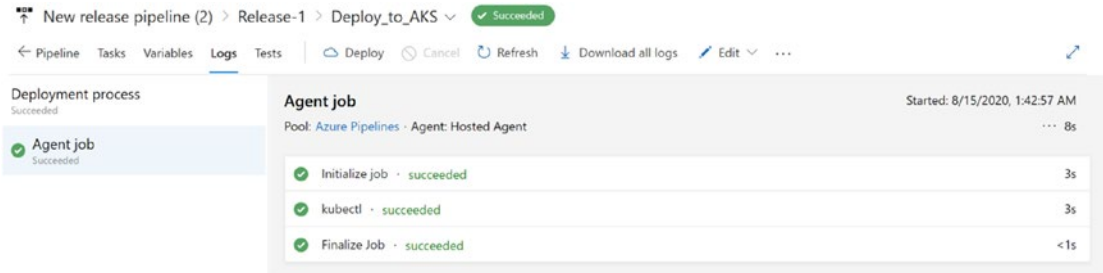
12. The release is getting created.



13. Click “Release-1,” to open the detailed view of the release task.



14. This results in a successful job.



Note The full deployment process is much much much more powerful and provides many more settings than we had here, but this is mainly to allow you to experience what a base deployment release pipeline can do and how to configure it.

This completes the task in which you created a new Azure release pipeline for a deployment of an ACR-stored repository to an existing Azure Kubernetes cluster.

Summary

In this lab, you performed several tasks around Azure DevOps, starting from the initial creation of an Azure DevOps organization, followed by creating an Azure DevOps build pipeline, using a GitHub repository with an application’s source code. In the next task, you created a release pipeline, deploying the build from the previous task, publishing an Azure Web App.

The following tasks involved creating an Azure DevOps release pipeline to publish an Azure Container Registry repository image to Azure Container Instance, as well as to publish to Azure Kubernetes Service.

Congrats if you completed all labs with all tasks from all modules. You should now have a real good understanding of Azure and where it can help in your overall digital transformation. Reach out when having any questions or concerns or wanting to share overall feedback about the lab content used in this book (peter@pdtit.be or @pdtit on Twitter). Have a nice day!

Index

A, B

App Service Migration Assistant, [62–66](#)

Azure Container Instance (ACI),
[1–3](#), [131](#), [132](#), [155](#), [165](#), [169](#), [174](#)

Azure Container Registry
(ACR), [2](#), [3](#), [131](#), [279](#)

ACI, [155](#)

admin credentials, [159](#), [160](#)

Azure Portal, [161](#)

CLI commands, [157](#), [158](#)

PowerShell window, [160](#)

Azure Container Workloads

ACR, Docker image

ACI instance, [168](#)

ACISubnet, [178](#)

choose run instance, [165](#)

connect to IP address, [180](#)

container instance, [165](#), [167](#), [172](#),
[173](#), [179](#)

copy IP address, [168](#)

logs tab, [169](#)

manage subnet configuration, [176](#)

networking button, [174](#)

stop container, [170](#)

deploying/operating, [181](#), [183–185](#)

deploying/running ACI, [162–164](#)

Docker-extensions,

VSC, [149](#), [151](#), [152](#), [154](#), [155](#)

Azure DevOps

ACR container, AKS, [307](#), [310](#), [311](#),
[313](#), [316](#)

build/release pipeline, [2–279](#)

creating/deploying pipeline, [254](#),
[256–259](#), [261](#)

creating/publishing Docker container,
ACR, [279–281](#), [283](#), [285](#), [286](#), [288](#),
[289](#), [292–294](#)

creating/release pipeline, Docker
containers, [294](#), [297](#), [298](#), [300–302](#),
[304](#), [306](#)

deploying organization, [234](#), [237–240](#)

Repos, source control, [241](#), [242](#), [244](#),
[247–249](#), [253](#), [254](#)

build/release pipelines, [242](#)

Azure Kubernetes Service (AKS), [1–3](#)

ACR integration, [193–196](#)

Azure CLI 2.0, [188–191](#), [193](#)

container scalability, [208–212](#), [214](#), [215](#)

Docker container image, [196](#), [198](#),
[200–205](#)

monitoring, [215](#), [217–221](#), [223](#), [225](#)

RBAC, [193–196](#)

Visual Studio Code, [225–227](#), [229–231](#)

Azure Resource Manager (ARM)

template, [3](#)

admin credentials, [26](#)

azuredeploy.json, [30](#)

INDEX

Azure Resource Manager (ARM)
 template (*cont.*)
 AzureResourceGroup44.sln file, [28](#)
 Azure resources, [24](#)
 deployment, Visual Studio (*see* Visual Studio 2019)
 files, [30](#)
 open project/solution, [28](#)
 Solution Explorer view, [29](#)
 theme, [27](#)
 time estimate, [23](#)
 Visual Studio, [25](#)

Azure Role-Based Access Control (Azure RBAC), [193](#)

Azure Web App
 migrating web application, [125](#), [127](#), [129](#), [130](#)
 publish ASP.NET project, [110](#), [111](#), [113-117](#)
 publish source code, [117](#), [119](#), [121](#), [124](#), [125](#)

C

CI/CD pipeline deployments, [233](#)
Containers as a Service (CAAS), [1](#)

D, E, F

Data Migration Assistant (DMA), *see* SQL server assessment

Data Migration Assistant (DMA), [57](#), [59](#), [62](#), [67](#)

Desired State Configuration (DSC), [3](#), [39](#)

Docker commands and containers
 validating/running
 docker images, [146](#)
 instance, [147](#)

log-JSON file, [148](#)
SimplCommerce webshop
 container image, [142](#), [143](#), [145](#)
 test Linux container, [139-141](#)

Docker Enterprise Edition, lab jumpVM
 DockerMSFTProvider, [134](#)
 installation, [134](#)
 Install-WindowsFeature
 Containers, [135](#), [136](#)
 LCOW component, [139](#)
 Linux Containers, [137](#)
 PowerShell, [133](#)
 update-cmdlet, [135](#)

G, H

GitHub setp scripts, [20](#), [21](#)

I, J, K

Infrastructure as a Service (IAAS), [1](#), [2](#)
Infrastructure as Code (IAC), [2](#), [3](#), [24](#)
Internet Information Services (IIS), [1](#), [42](#)

L

lab jumpVM
 azuredeploy.json, [11](#)
 Azure Marketplace, [8](#)
 Azure subscription, [8](#)
 content, [12](#)
 credentials, [18](#)
 custom deployment, [9](#)
 Edit template blade, [13](#)
 Microsoft.Template, [16](#)
 networks, [19](#)
 notifications area, [15](#), [16](#)
 RDP file, [18](#)

- required fields, 14
 - resource groups, 16
 - source files, 10
 - template deployment, 9
 - terms and conditions, 15
 - warning message, 19
 - Lab virtual machine (VM)
 - exercises, 7
 - time estimate, 7
 - Linux Containers on
 - Windows (LCOW), 136, 137, 139
- ## M
- Microsoft assessment tools, 2, 51
- ## N, O
- Network Security Group (NSG) rules, 17
- ## P, Q
- Platform as a Service (PAAS), 1, 2, 66, 181
- ## R
- Remote Desktop (RDP), 5, 19, 53, 103
- ## S
- Server assessment
 - scenario diagram, 51
 - SQL, 52
 - time estimate, 51
 - web server (*see* Web server assessment)
 - SQL database migration
 - administrative credentials, 88
 - breadcrumbs link, 90
 - client IP address, 90
 - connection blade, 91
 - data migration assistant, 79
 - deploy schema, 85
 - firewall setting, 89
 - migrate data, 86
 - parameters, 79–81
 - query, 92
 - query editor (preview), 88
 - RDP session, 79
 - scenario diagram, 68
 - script, 85
 - security warning, 89
 - SimplCommerce, 81
 - steps, 67
 - tables, 83, 91
 - time estimate, 68
 - SQL server assessment
 - connect server, 60
 - credentials, 53
 - download DMA, 54
 - features, 61
 - launch DMA, 55, 57
 - parameters, 58, 59
 - security warning, 54
 - SimplCommerce, 60
 - tasks, 52
 - WebVM, 52
 - SQL Server instance
 - advanced settings, 76
 - basic tab, 69
 - configuration settings, 77
 - configure database, 74
 - create SQL Server, 69
 - database blade, 77
 - database name/size, 73
 - firewall settings, 77
 - networking tab, 71, 76

INDEX

SQL Server instance (*cont.*)

parameters, 78

validation, 71

SQL Server Management Studio

admin credentials, 94

18 console, 95

mstsc.exe, 94

Object Explorer, 98

RDP session, 93

server connection

information, 95, 100

server credentials, 97

SimplCommerce, 99

SQLVM, 94

T, U

Technical requirements

Azure subscription, 4

GitHub repository, 6

local client admin machine, 5

naming conventions, 5

Operating System, 5

tools, 5

V

Virtual machine (VM), 1, 110, 133, 175

Visual Studio 2019

AzureResourceGroup44

project, 32

connect to Server, 46

credentials, 42, 44

dbo.Catalog_Product, 48

edit parameters, 33, 34

Internet Information Services

Manager, 43

list of products, 49

Microsoft Remote Desktop

Connection, 43

name, 37

Output window, 35, 38

resource groups, 36

settings, 32

SimplCommerce, 40, 47

SQL management, 46

sqlvm, 44

WebVM virtual machine, 39

W, X, Y, Z

Web server assessment

choose a site, 64

download App Service Migration

Assistant, 62, 63

launch App Service Migration

Assistant, 64

report, 65

tasks, 62

WebVM, 62

WebVM

appsettings.json file, 104

command prompt, 105

connection strings, 102

error message, 107

IIS web server, 103

parameters, 104

product catalog list, 106

SQLVM, 105

web.config file, 103