



Getting Started with Containers in Azure

Deploy, Manage, and Secure Containerized
Applications

—
Shimon Ifrah

Apress®

Getting Started with Containers in Azure

**Deploy, Manage, and Secure
Containerized Applications**

Shimon Ifrah

Apress®

Getting Started with Containers in Azure: Deploy, Manage, and Secure Containerized Applications

Shimon Ifrah
Melbourne, VIC, Australia

ISBN-13 (pbk): 978-1-4842-5752-4
<https://doi.org/10.1007/978-1-4842-5753-1>

ISBN-13 (electronic): 978-1-4842-5753-1

Copyright © 2020 by Shimon Ifrah

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: Smriti Srivastava
Development Editor: Matthew Moodie
Coordinating Editor: Shrikant Vishwakarma

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, email 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 email rights@apress.com, or visit <http://www.apress.com/rights-permissions>.

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/978-1-4842-5752-4. For more detailed information, please visit <http://www.apress.com/source-code>.

Printed on acid-free paper

Table of Contents

- About the Author xi
- About the Technical Reviewer xiii
- Introduction xv
- Chapter 1: Get Started with Microsoft Azure..... 1
 - Setting Up Your Azure Subscription..... 1
 - Sign Up for Azure..... 2
 - Assigning Permissions to Azure 4
 - Set Up Azure Cloud Shell..... 9
 - Azure CLI 12
 - Using PowerShell in Azure Cloud Shell..... 14
 - Azure Cloud Shell Code Editor 15
 - Uploading and Downloading Files to and from Azure Cloud Shell..... 17
 - Secure Your Microsoft Azure Account 18
 - Enable Multi-Factor Authentication (MFA) 19
 - Check Global Administrator Accounts..... 21
 - Azure Container Services..... 22
 - Azure Container Instances (ACI) 23
 - Azure Kubernetes Services (AKS)..... 24
 - Azure Container Registry (ACR) 25
 - Summary..... 26

Chapter 2: Store and Manage Docker Container Images on Azure Container Registry (ACR) 27

Overview of Azure Container Registry (ACR)..... 28

Set Up Microsoft Azure Container Registry (ACR) 29

 Install Docker Desktop..... 29

 Creating Azure Container Registry (ACR) Using the Portal 30

 Creating Azure Container Registry Using Azure CLI..... 33

Push Docker Images to Azure Container Registry (ACR) 34

 Connect to ACR Using Docker..... 35

Pull Images from Azure Container Registry (ACR)..... 38

Manage and Secure Azure Container Registry (ACR)..... 39

 Install VS Code..... 40

 Install Docker Extension 40

 Manage Containers with VS Code..... 43

 Manage Docker Images with VS Code..... 44

 Manage a Container Registry with VS Code 44

 Securing Azure Container Registry (ACR) 49

 Upgrade ACR SKU Plan 49

 Rotate ACR Access Keys..... 52

 Azure CLI 53

Summary..... 55

Chapter 3: Deploy Containerized Applications with Azure Container Instances (ACI) 57

Set Up Azure Container Instances (ACI) 58

 AZ Container Commands 59

 Container Groups..... 60

 ACI Limitations..... 60

Deploy Linux and Windows Containers to ACI..... 60

 Deploy Linux Containers..... 61

 Deploy Containers Using Azure CLI..... 67

 Deploy Windows Containers 68

Mount Storage Volume to an ACI Container	70
Mounting Azure File Share Volume to an ACI Container	71
Scale Containerized Applications in ACI.....	76
Deploy to Web App for Containers	81
Monitor and Manage Containerized Applications on ACI.....	83
Monitoring ACI with Azure Monitor	83
Managing ACI Deployments.....	89
Manage ACI with Azure Resource Explorer.....	95
Summary.....	96
Chapter 4: Deploy Containerized Applications with Azure Kubernetes Service (AKS).....	97
Kubernetes.....	98
Kubernetes Components.....	98
Kubernetes Master	98
Kubernetes Node	99
Kubernetes Add-ons	99
Getting Started with AKS.....	100
Creating an AKS Cluster.....	100
Creating an AKS Cluster Using Azure Cloud Shell.....	108
Connecting ACR to AKS.....	108
Deploying Containers Using an ACR Image to AKS	109
Deploy Kubernetes Web UI (Dashboard) on AKS	112
Enable Web UI (Dashboard) on AKS	113
Deploy with Dashboard (Web UI)	113
Mount Storage Volumes in AKS.....	116
Create a Storage Class	117
Create a Storage Role.....	118
Create a Volume Claim	119
Create a Deployment with Persistent Storage.....	119
Manage and Secure AKS.....	121
Summary.....	128

Chapter 5: Deploy Docker Container Host on Azure Virtual Machine 129

 Why Use a Container Host?..... 129

 Installing Docker Container Host on Ubuntu Linux VM..... 130

 Create a Linux Docker Host Using the Portal..... 130

 Create a Linux Docker Host Using Azure CLI and Azure Cloud Shell..... 132

 Get Azure VM Using Azure CLI..... 133

 Connect Using SSH from Azure Cloud Shell 134

 Install Docker Using the Snap Application Store 135

 Install the Latest Docker Version 136

 Install Docker Container Host on Windows Server 2019 VM..... 137

 Create a Windows Docker Container Using the Portal..... 137

 Create a Windows Docker Container Using Azure CLI 138

 Connect to the VM 139

 Update Docker on Windows Server 2019 140

 Manage Container Hosts on Azure 141

 Auto-Shutdown Container Host VM 141

 Configure Start/Stop Container Host VM 144

 Summary..... 160

Chapter 6: Secure Your Microsoft Azure Containers 161

 Protect and Manage Accounts and Hosts on Microsoft Azure Using Azure AD and Bastion 162

 Azure Active Directory Reporting..... 162

 Azure Bastion 167

 Azure Bastion Pricing 167

 Bastion in Action..... 168

 Enable Azure Bastion on an Azure VM 168

 Manage Bastion..... 175

 Create Bastion Using Azure PowerShell 177

Use Azure Security Center and Secure Score to Protect Your Tenant	177
Security Center	177
Secure Score	182
Secure Your Containers with Azure Firewall and Network Security Groups (NSG).....	184
Azure Firewall.....	184
Network Security Groups (NSGs).....	184
Summary.....	193
Chapter 7: Scale Containers and Containerized Applications on Azure	195
Scale Azure Kubernetes Service (AKS)	195
Create an AKS Cluster.....	196
Scaling an AKS Cluster	196
Autoscale AKS Using Azure CLI.....	201
Update Existing AKS Cluster to Autoscale	202
Delete an AKS Cluster Using Azure CLI	202
Scale Azure Container Instances (ACI)	202
Scale Azure Container Registry (ACR).....	205
Scale Azure Docker Container Host VM.....	205
Scale an Azure VM Using the Portal.....	206
Scale an Azure VM Using Azure CLI	207
Azure Spot Virtual Machines	207
Azure Spot Virtual Machines Pricing.....	208
Deploy a Spot Virtual Machine.....	208
Eviction Policy	209
Create a Spot VM Using Azure CLI	210
Summary.....	210
Chapter 8: Monitor Containers and Containerized Applications on Azure.....	211
Azure Monitor Overview.....	211
Azure Monitor Products	212
Azure Monitor Data.....	212
Azure Monitor Activity Logs	213

TABLE OF CONTENTS

Monitor Azure Kubernetes Service (AKS)	213
Log Analytics	214
Monitoring	215
Disable Azure Monitor for Containers	223
Monitor Azure Container Instances (ACI).....	224
Monitor ACI with Azure CLI	226
Monitor Azure Container Registry (ACR)	227
Monitor Azure Docker Container Host VM	230
Azure Monitor Insights for VMs	231
Connection Monitor	241
Disable Azure Monitoring for VMs	244
Summary.....	246
Chapter 9: Back Up and Restore Containers and Containerized Applications on Azure.....	247
Azure Backup Solutions	248
Azure Backup	248
Azure Site Recovery	249
Back Up Azure Kubernetes Service (AKS)	250
Azure Recovery Services.....	251
Back Up Azure File Shares.....	256
Restore Azure File Shares	260
Monitor Backup Jobs.....	264
Back Up Azure Container Instances (ACI).....	266
Back Up Azure Container Registry (ACR).....	268
Back Up Azure Docker Container Host VM	269
Back Up Azure VM Using Azure CLI.....	273
Restore a Virtual Machine.....	273
Delete Recovery Services Vault	276
Summary.....	278

Chapter 10: Troubleshooting Containers and Containerized Applications on Azure.....	279
Troubleshooting Azure Kubernetes Service (AKS).....	281
Capacity Issues.....	281
Storage Issues.....	281
AKS Limits	281
IP Address Limit.....	282
Pod Issues	282
Troubleshoot Azure Container Instances (ACI)	283
Slow Start.....	283
Support Versions	284
Bad Performance.....	284
Troubleshoot Azure Container Registry (ACR)	284
Storage Issues.....	284
Performance Issues.....	285
Can't Log In to ACR.....	285
Access Issues (Firewall)	285
Troubleshoot Azure Docker Container Host VM	286
Boot Diagnostics.....	287
Redeploy VM	290
Serial Console.....	292
Connection Troubleshooting	293
Summary.....	294
Index.....	295

About the Author



Shimon Ifrah is an IT professional with years of experience in the design, management, and deployment of information technology systems and networks. In the last few years, Shimon has been specializing in cloud computing and containerized applications on Amazon AWS and other public cloud providers. Shimon also holds more than twenty vendor certificates from Microsoft, AWS, VMware, and Cisco. During his career in the IT industry, he has worked for some of the largest managed services and technology companies in the world, helping them administer systems for the largest enterprises. He is based out of Melbourne, Australia.

About the Technical Reviewer



Samuel Rowe is a technologist who specializes in application innovation and solving problems. He is an agile advocate with extensive experience in DevOps and cloud practices. He is currently working at Microsoft as a solution architect, having previously worked for IBM and Codeweavers.

Introduction

Back in early 2019, while finishing my book on deploying containers on AWS, I decided I wanted to write a similar book about the same technology in Microsoft Azure.

In a multi-cloud world where companies use multiple cloud providers to run their workload, locking yourself to one platform is not an ideal solution. Because most cloud platforms offer similar services, transitioning between platforms is not hard if you already have experience with one platform.

There is no doubt that Microsoft Azure is a great platform, and ignoring it can only harm your career path to success. This book will fill in the gaps and give you a step-by-step guide on getting started with Azure and deploying containerized applications on Azure services.

Azure has a variety of cloud services that allow us to deploy containers, and in this book I will cover each of them in great detail.

When I planned this book, I wanted to achieve two things:

- Write a book that is not superficial and that touches topics on a deep technical level backed by a step-by-step approach.

- Write a book that covers all elements of running applications in a production environment and doesn't just focus on running code.

In this book, you will not find pages filled with back-to-back code. Instead, you will find chapters filled with technical information and explanations that will help you understand and take actions. My detailed, easy to understand, step-by-step approach will walk you through the labs without missing important information.

I hope you will find this book useful and helpful in growing your career and knowledge base. This book is the second book in my “Deploy Containers” book series, and I hope you like it.

The third book in the series is planned to be released in 2021.

I would like to take this opportunity to say thank you to my loving family for their amazing support and love that they have given me during the writing stages.

I would also like to say thank you to the amazing team at Apress for believing in me and giving me the opportunity to publish my books.

CHAPTER 1

Get Started with Microsoft Azure

This book will cover all Microsoft Azure Container services and support services needed to secure, back up, and maintain containerized applications on Microsoft Azure. In this chapter, we will focus on setting up our Microsoft Azure cloud environment and all the tools needed to deploy containers in Azure.

We will start with signing up for a Microsoft Azure subscription, and then we will assign permissions for our Administrator account, set up Azure Cloud Shell and Azure CLI, and finish with an overview of all the Container services on Microsoft Azure. To begin, we will cover the basics and build our capabilities from there using the following topics:

- Set up your Azure tenant and subscription.
- Secure and set up Administrative permissions.
- Set up Azure Cloud Shell.
- Perform Azure Container Services Overview.

Setting Up Your Azure Subscription

The first step in getting started with Azure is setting up an Azure account and billing.

Like other cloud providers, Microsoft Azure offers free Azure services to people signing up.

Currently, Azure offers the following for new accounts:

Twelve months of free services (see Table 1-1)

\$200 free credit to use any Azure service for thirty days

25+ Azure services that are always free to use

Table 1-1 shows each service that is free for twelve months and its free monthly limit.

Table 1-1. *Twelve Months Free Services*

Service	Details
Linux and Windows Virtual Machines	750 hours every month using B1S VM instance
Blob Storage	5 GB of LRS hot block storage
SQL Database	20 GB of storage
File Storage	5 GB of LRS file storage
Managed Disks	64 GB x 2
Azure Cosmos DB	5 GB and 400 requests units
Bandwidth	15 GB of outbound data transfer

Table 1-2 lists the main Azure products that are always free.

Table 1-2. *Azure Products and Services That Are Always Free*

Service	Details
App Services (Compute)	10 web, mobile or API apps
Azure Functions (Compute)	1,000,000 requests per month
AKS (Artificial Intelligence + Machine learning)	Free deployments of containers
Face API (Artificial Intelligence + Machine learning)	30,000 requests
DevTest Labs	Free developer tools

The main objective of the free services is getting you to explore, understand, and learn how to use them at no cost, and it is recommended you take advantage of the free services.

Sign Up for Azure

To get started and sign up for Azure, you will need to provide a billing method, regardless of whether you are using free services or not.

The billing method is used if you use Azure services that are not included in the free offer.

To sign up for an Azure account, use the following URL: <https://azure.microsoft.com/en-us/free/>. Fill in your details and follow the prompts. During the registration process, you will be asked to verify your identity using one of the following two verification methods.

Phone verification — using an SMS or automated phone call

Credit card — providing a credit card for billing purposes

Make sure you have these with you before signing into Azure.

Figure 1-1 shows the signup page with the free offers that are included for new account registrations.

Microsoft Azure shimoni4@outlook.com Sign out

Try Azure for free

Follow these steps to get started. We ask for these details to protect your account and information. There are no upfront charges or fees.

1 About you

Country/Region ⓘ

Australia ▼

First name

Last name

Email address ⓘ

Phone

Example: 499 123 456

ABN ⓘ

Optional

Next

What's included

- ✓ 12 months of free products**
 Get free access to popular products like *virtual machines*, *storage*, and *databases* in your first 30 days, and for 12 months after you upgrade your account to pay-as-you-go pricing.
- ✓ \$260 credit**
 Use your \$260 credit to experiment with any Azure service in your first 30 days—beyond the free product amounts.
- ✓ 25+ always-free products**
 Take advantage of more than 25 products, including *serverless*, *containers*, and *artificial intelligence*, that are always free. Get these in your first 30 days, and always—once you choose to upgrade.
- ✓ No automatic charges**
 You won't be charged unless you choose to upgrade. Before the end of your first 30 days, you'll be notified and have the chance to upgrade and start paying only for the resources you use beyond the free amounts.

Figure 1-1. Try Azure for free

After the registration process is complete, you will be redirected to the Azure portal and presented with an option to take a tour that will show you where everything is in the portal.

If you are new to Microsoft Azure, I recommend you take the tour.

Assigning Permissions to Azure

Now that we have our subscription and are logged in to the Azure portal, we will explore the steps needed to assign permissions in Azure.

By default, the user who creates the Azure subscription has the Global Administrator permission, which gives him or her full administrative rights to manage all resources and subscriptions.

This section will show you how to use role-based access control (RBAC) to assign permissions in Azure. Microsoft Azure uses RBAC to control access to Azure resources. RBAC is very powerful because it breaks down the permissions into roles and limits the number of permissions given to each person. Because the permissions are based on roles, the risk of giving too many permissions is limited.

In Azure, RBAC is implemented using access control, also known as identity and access management (IAM), which is available on a subscription level, resource-group level, and individual-resource level, which means we can control access to Azure resources on a few levels.

In most cases, you will assign your Global Administrators Owner permissions on the subscription level, which will give them access to all the resources on Azure; however, in an enterprise environment it is recommended you use management groups.

For other users, you can limit access to either the resource-group level or individual resources.

In Figure 1-2, you can see the access control (IAM) blade on the subscription level.

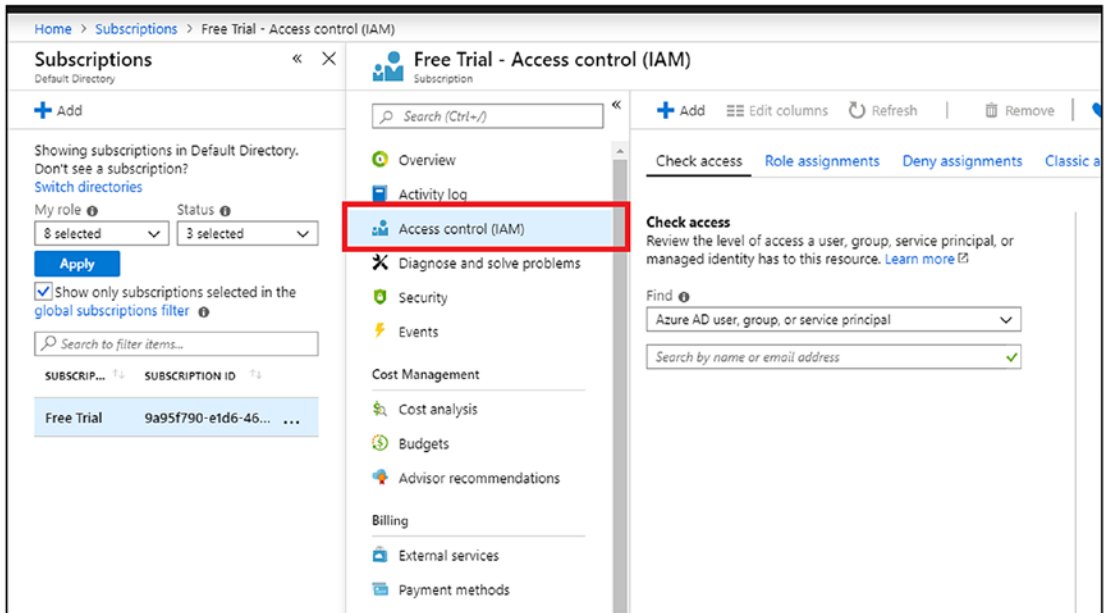


Figure 1-2. Access control (IAM)

Adding a user to an RBAC role on the subscription level will give them access to all the resources in the subscription.

To add a user to an RBAC role, from the left menu on the Microsoft Azure portal click on All Services, then click on Subscriptions, as shown in Figure 1-3.

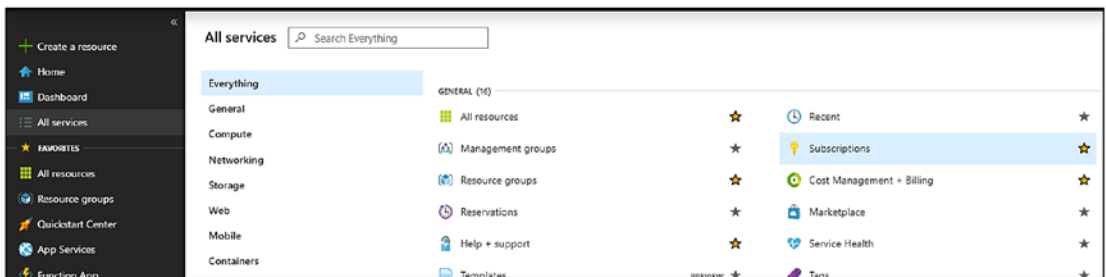


Figure 1-3. Azure subscriptions

From the Subscriptions page, select the Azure subscription you would like to assign permissions to.

In my case, I only have one subscription, which is the Free Trial subscription, as shown in Figure 1-4.

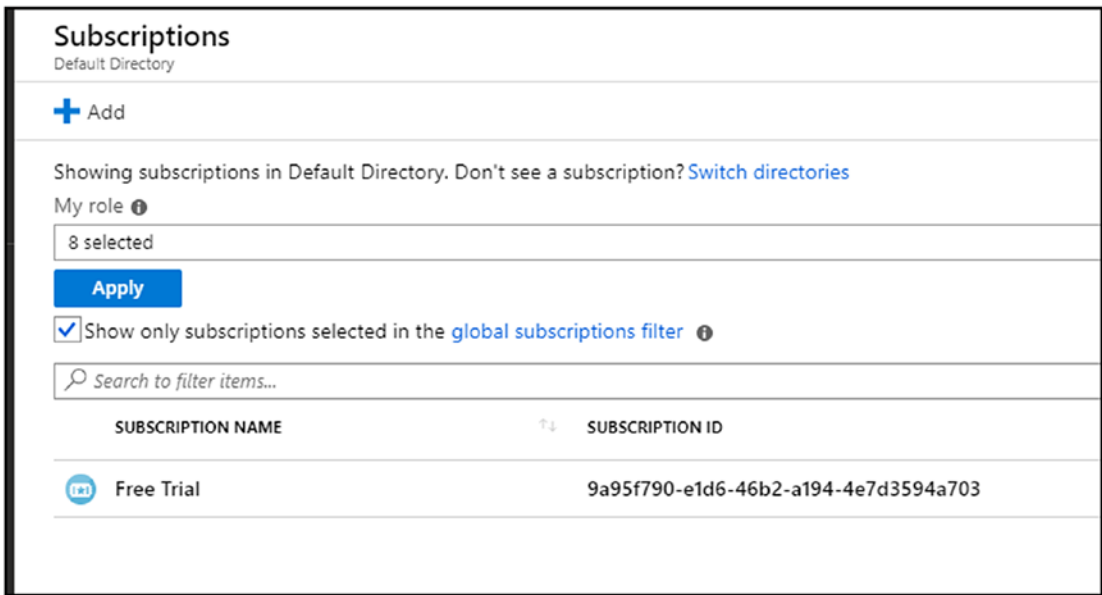


Figure 1-4. Azure Subscriptions page

From the Subscriptions page, I will click on the Access Control (IAM) menu on the left, as shown in Figure 1-5.

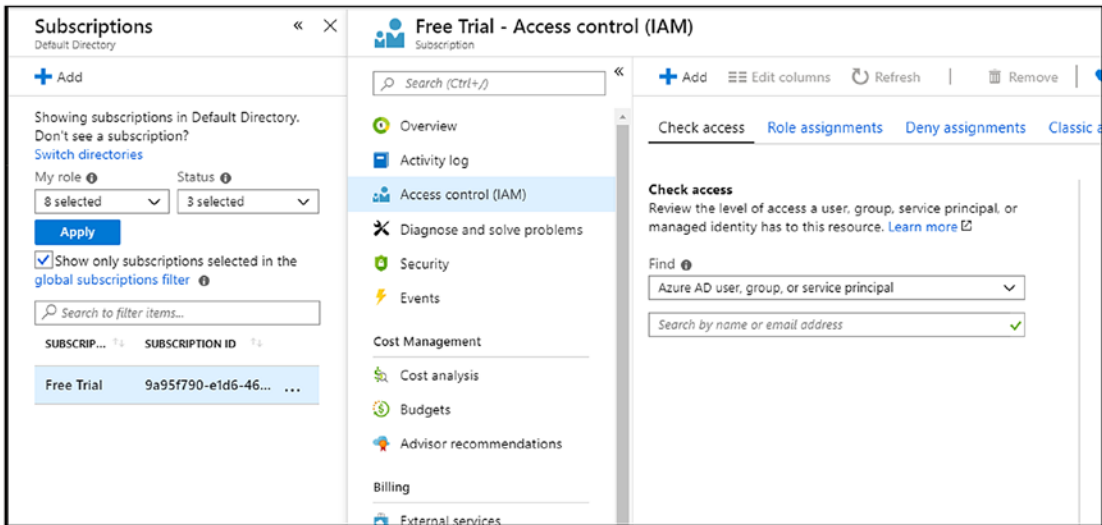


Figure 1-5. Access Control (IAM) page

To assign permissions to a user, click on Add and select “Add role assignment” from the drop-down menu, as shown in Figure 1-6.

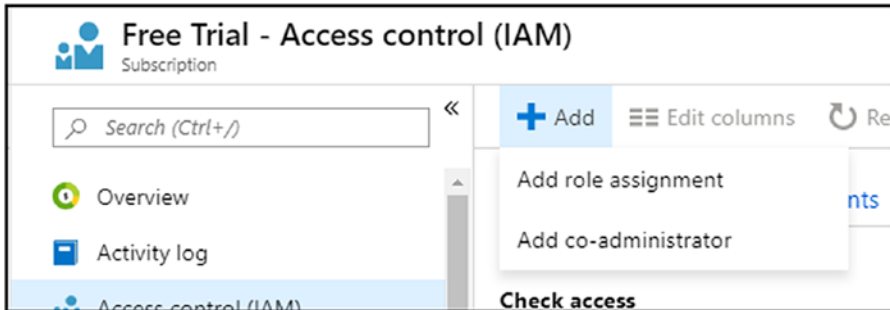


Figure 1-6. Add Role assignment

On the Add Role Assignment page, I have the option to select the role I would like to assign from the list.

Currently, Azure offers more than seventy assignment roles to select from.

The Owner role will give the user full administrative permissions for everything in Azure, including access to resources. You should avoid assigning this role to too many administrators, and Microsoft recommends that you have less than five administrators with Owner permissions.

Figure 1-7 shows the Add Role Assignment page and part of the list of roles available.

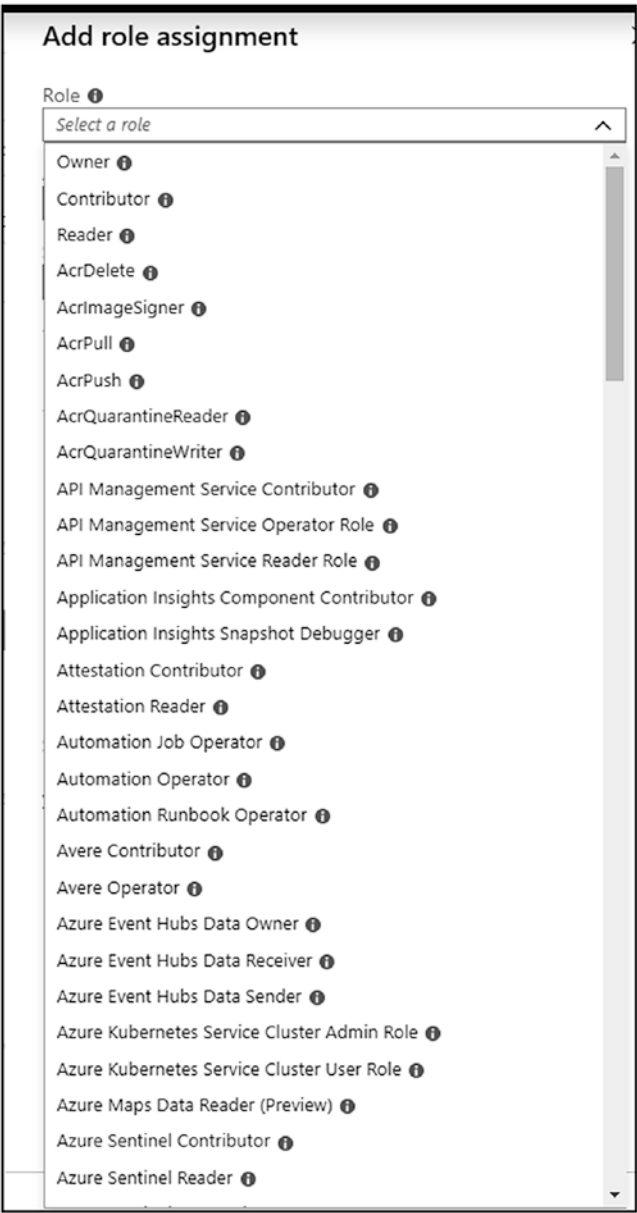


Figure 1-7. Add Role Assignment page

After selecting the role, I have the option to assign it to the user or groups.

From the Select search box, search for the user you would like to assign the owner role and click Save, as shown in Figure 1-8.

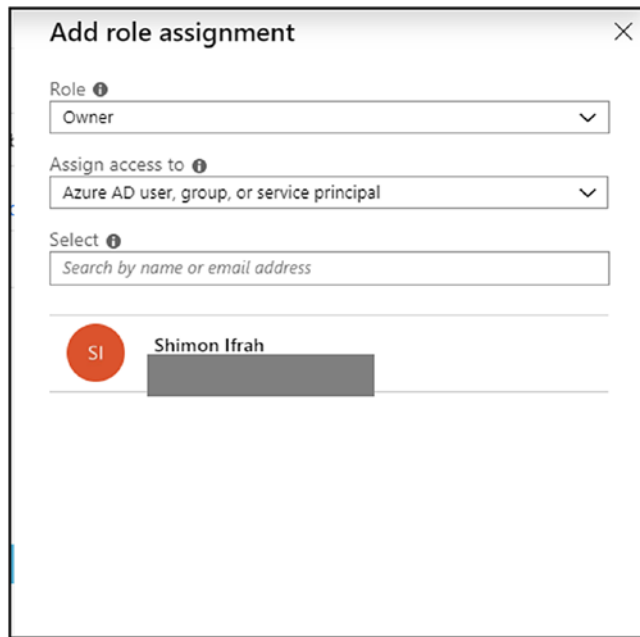


Figure 1-8. Select user from the Add Role Assignment page

Set Up Azure Cloud Shell

In this section, we will configure Azure Cloud Shell and learn how to use it and how to manage resources and services in Microsoft Azure.

Azure Cloud Shell is a browser-based command-line utility hosted in Microsoft Azure and pre-loaded with common tools and programming languages.

Azure Cloud Shell is available from anywhere and is always up-to-date with the latest tools. All the data, including the command-line history, is saved into a persistent Azure file share.

Azure Cloud Shell pricing is based on the Azure file share size where the data exist.

In Cloud Shell, we select between Bash and PowerShell as the shell we will run the command from.

When using the Bash Shell, the Azure file share will create a 5GB image that will be used to run it.

To set up Cloud Shell, from the Azure portal click on the Cloud Shell icon, as shown in Figure 1-9.



Figure 1-9. *Azure Cloud Shell icon*

When you open Azure Cloud Shell for the first time, it will ask you to select which shell you would like to start with.

In my case, I will select Bash.

Figure 1-10 shows the Welcome to Azure Cloud Shell page.

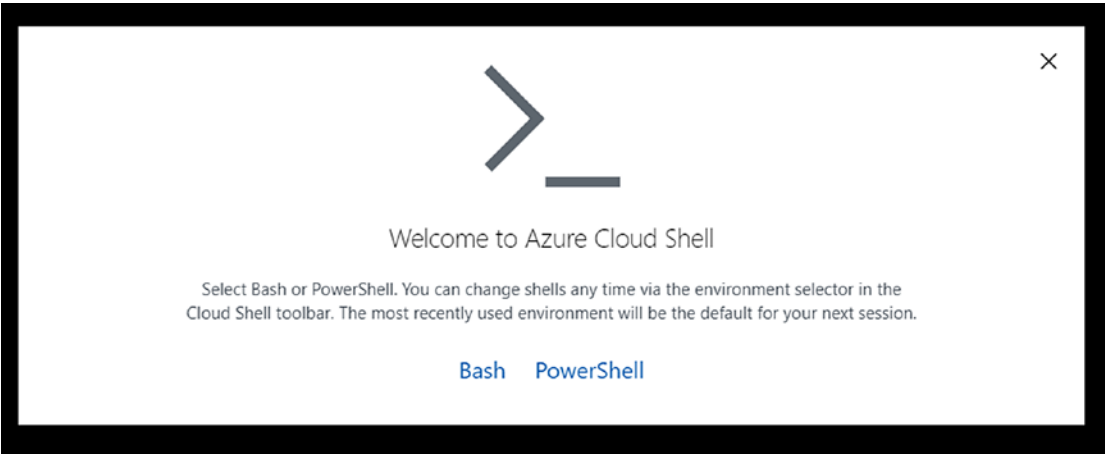


Figure 1-10. *Welcome to Azure Cloud Shell page*

Before we can start using Azure Cloud Shell, we need to create a storage account that will be used to keep our data access sessions.

Figure 1-11 shows the storage account setup for Azure Cloud Shell.

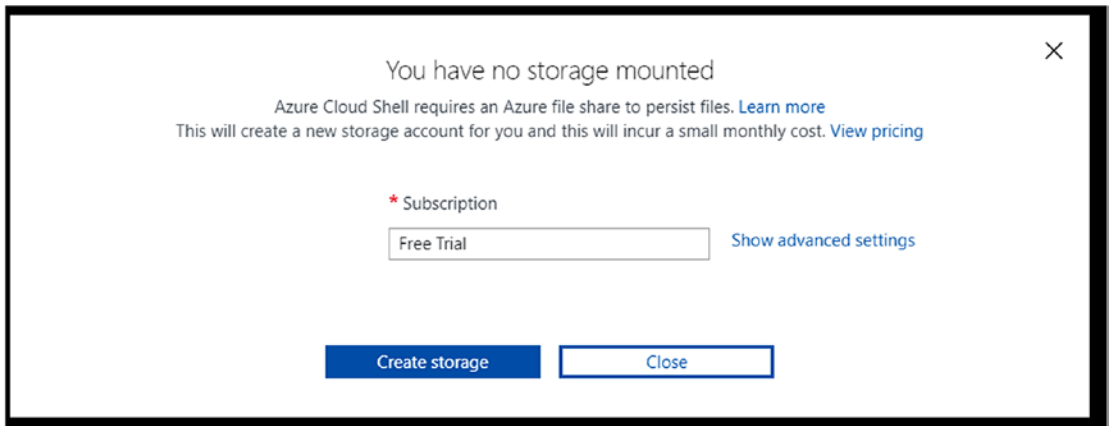


Figure 1-11. Create Azure file share for Azure Cloud Shell

After creating the storage account, you will be redirected Azure Cloud Shell, as shown in Figure 1-12.

You will also notice that Bash Shell is being used.

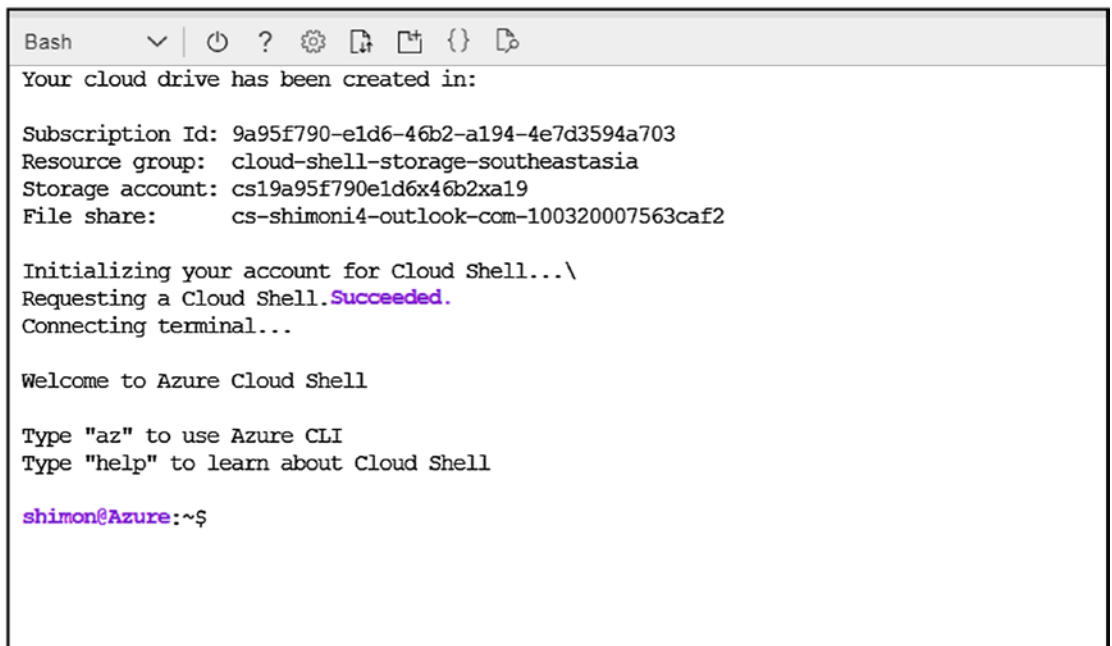


Figure 1-12. Azure Cloud Shell has been created screen

Azure CLI

Azure CLI is a command-line tool that Microsoft designed to automate, manage, and deploy Azure services quickly and effectively.

The tool can be installed on any platform, is preloaded into Azure Cloud Shell, and is always running the latest version.

Azure CLI is also available as a Docker image that can be run inside a container; to download the Azure CLI Docker image, please visit <https://hub.docker.com/r/microsoft/azure-cli/>.

To download and install Azure CLI for Windows, MacOS, and Linux, please visit <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest>.

To get started with Azure CLI inside Azure Cloud Shell, type the following command:

az -h

In Figure 1-13, you can see the Azure CLI Help menu and all the available commands.

```
shimon@Azure:~$ 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:

account	: Manage Azure subscription information.
acr	: Manage private registries with Azure Container Registries.
ad	: Manage Azure Active Directory Graph entities needed for Role Based Access

Figure 1-13. Azure CLI help

The Azure CLI command syntax is as follows:

`az command action`

In Figure 1-14, you can see an Azure CLI command that shows all the resources available inside an Azure subscription.

```
shimon@Azure:~$ az resource list
[
  {
    "id": "/subscriptions/9a95f790-e1d6-46b2-a194-4e7d3594a
    "identity": null,
    "kind": "StorageV2",
    "location": "southeastasia",
    "managedBy": null,
    "name": "cs19a95f790e1d6x46b2xa19",
    "plan": null,
    "properties": null,
    "resourceGroup": "cloud-shell-storage-southeastasia",
    "sku": {
      "capacity": null,
      "family": null,
      "model": null,
      "name": "Standard_LRS",
      "size": null,
      "tier": "Standard"
    },
    "tags": {
      "ms-resource-usage": "azure-cloud-shell"
    },
    "type": "Microsoft.Storage/storageAccounts"
  }
]
```

Figure 1-14. Azure CLI `az resource list` command

The command is:

`az resource list`

Azure Cloud Shell also offers a fast switch between the Bash and PowerShell experiences.

To switch between Bash Shell and PowerShell, use the top left corner drop-down menu to switch between the shells.

Figure 1-15 shows the drop-down menu that switches between Bash and PowerShell.

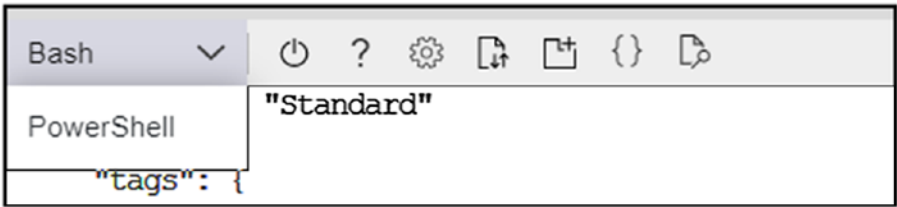


Figure 1-15. *Switch between Bash and PowerShell*

You can switch between the shells anytime, and all sessions’ information will be saved automatically.

It is important to note that if you have processing running in the Bash sessions, that information will continue running.

Figure 1-16 shows the confirmation screen when you switch between shells.

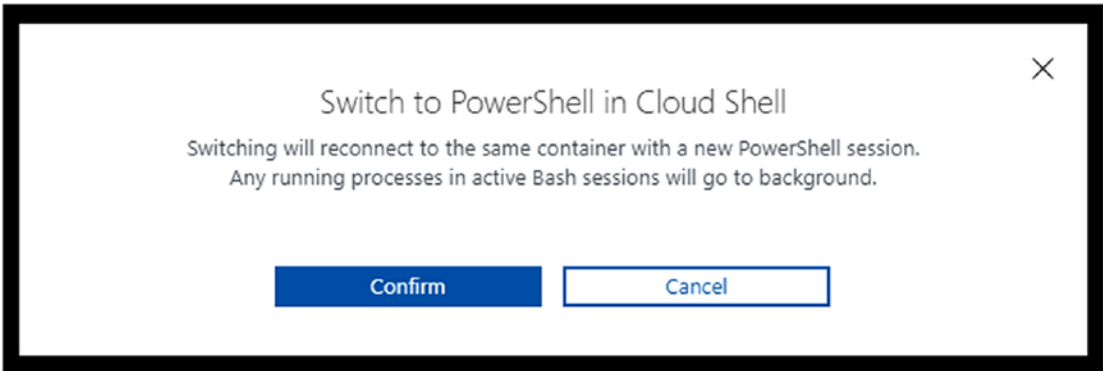


Figure 1-16. *Switch between shells*

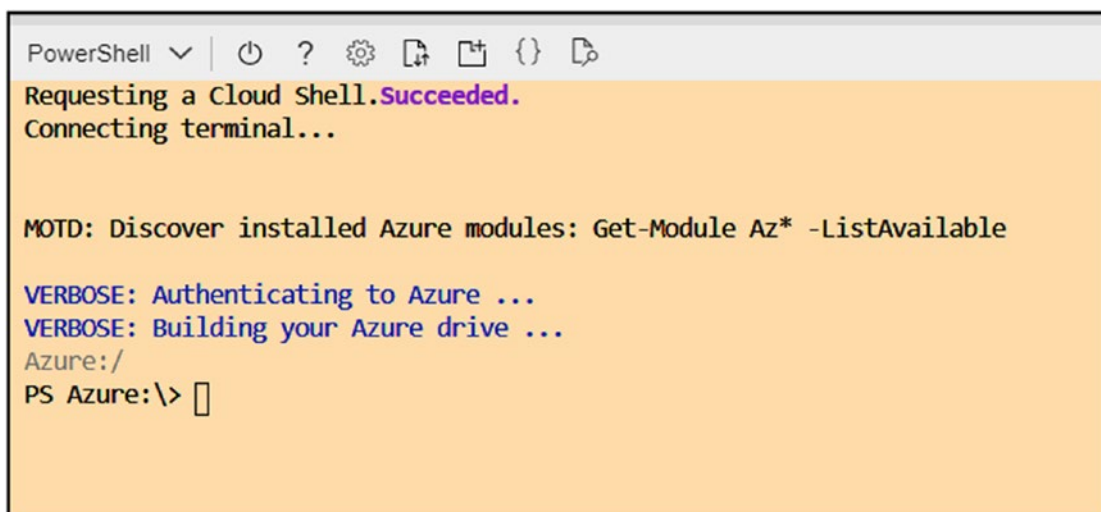
Using PowerShell in Azure Cloud Shell

If you decide to use PowerShell in Azure Cloud Shell, please note that the PowerShell version running in Cloud Shell is PowerShell Core.

PowerShell Core, also known as PowerShell 6.0, is the open-sourced and multi-platform version of PowerShell.

In Figure 1-17, you can see the PowerShell experience inside Azure Cloud Shell.

By default, Azure AD and Exchange Online PowerShell tools are also available from Azure Cloud Shell.



```
PowerShell v | [Icons]
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

MOTD: Discover installed Azure modules: Get-Module Az* -ListAvailable

VERBOSE: Authenticating to Azure ...
VERBOSE: Building your Azure drive ...
Azure:/
PS Azure:\> [Cursor]
```

Figure 1-17. PowerShell in Azure Cloud Shell

To view all the available PowerShell modules in Azure Cloud Shell, type the following command:

```
Get-module az*
```

Azure Cloud Shell Code Editor

Azure Cloud Shell also comes pre-loaded with a code editor out of the box without your needing to install it.

The editor is based on the open-source code editor Monaco Editor, which is the same code editor used by Visual Studio Code.

The editor allows us to do the following:

1. Create scripts and notes
2. Edit existing or new scripts
3. Highlight language code
4. Use the command palette
5. Access File Explorer

You can access the editor in two ways; first, by using the following command:
code.

The second option is by using the editor icon, available from the top menu.

Figure 1-18 shows the editor icon in Azure Cloud Shell. The editor is available from both shells, PowerShell and Bash.

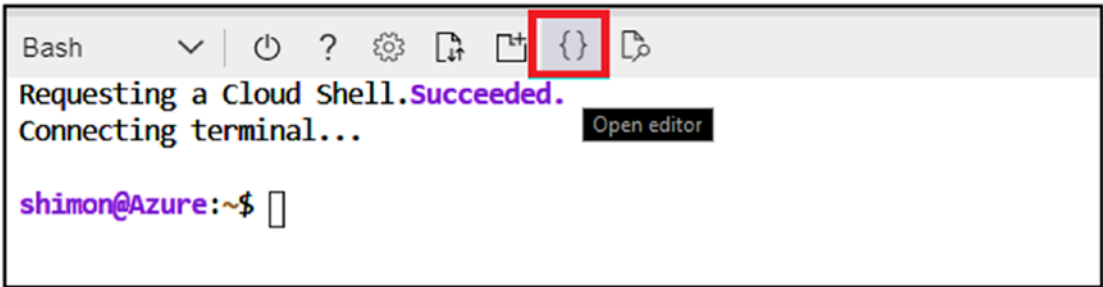


Figure 1-18. Azure Cloud Shell editor icon

If you click on the editor icon or use the command to start it, you will see it in action, as shown in Figure 1-19.

Once opened, you can start typing your code or paste it.

To create a new file, simply use the following command:

```
touch script.sh
```

The preceding command will create a Bash script, which we can load from the Files menu on the left and then start writing code into it.

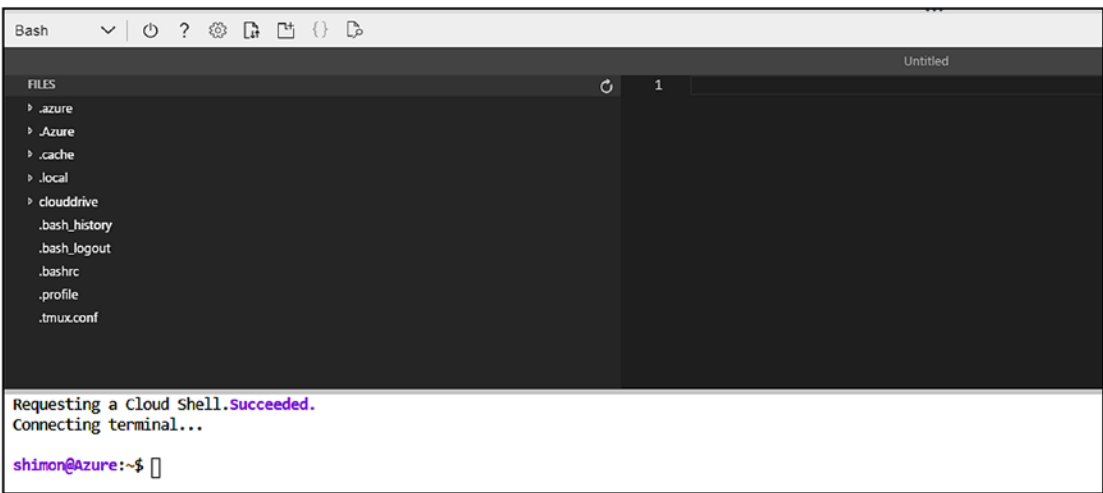


Figure 1-19. Editor in action

To run a script in Azure Cloud Shell, simply use the following command:

```
bash bashscript.sh
```

Uploading and Downloading Files to and from Azure Cloud Shell

What if you want to upload or download scripts to or from Azure Cloud Shell?

That's not a problem. Azure Cloud Shell comes with a built-in File Explorer that allows us to easily upload and download files that are stored on Azure Cloud Shell.

Figure 1-20 shows Azure Cloud Shell File Explorer and how easy it is to access it.

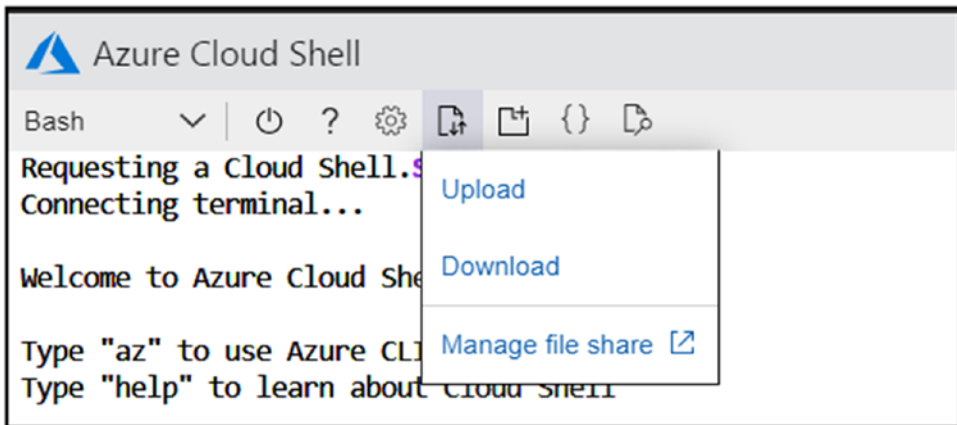


Figure 1-20. Azure Cloud Shell File Explorer

If you click on “Manage file share,” you will be redirected to the actual Azure file share that holds your Azure Cloud Shell sessions.

Figure 1-21 shows the Azure file share that stores the Azure Cloud Shell sessions with the following details:

- Storage account name

- Azure file share name

- Quota — in my case, the size is 6 GiB

File share Refresh

Storage account: cs19a95f790e1d6x46b2xa19

Search file shares by prefix

NAME	MODIFIED	QUOTA
cs-shimoni4-outlook-com-100320007563caf2	10/1/2019, 8:56:37 PM	6 GiB

Figure 1-21. Azure file share

If you would like to check which version of PowerShell Azure Cloud Shell is running, simply run the following command:

```
$PSVersionTable
```

As you can see in the output shown in Figure 1-22, Azure Cloud Shell runs PowerShell Core version 6.2.3.

```
PS Azure:\> $PSVersionTable

Name                           Value
----                           -
PSVersion                      6.2.3
PSEdition                      Core
GitCommitId                    6.2.3
OS                              Linux 4.15.0-1059-azure #64-Ubuntu SMP Fri Sep 13 17:02:44 UTC 2019
Platform                      Unix
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0...}
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1
WSManStackVersion              3.0

Azure:/
PS Azure:\> 
```

Figure 1-22. PowerShell Core version command

The operating system that runs Azure Cloud Shell is Linux Ubuntu.

Secure Your Microsoft Azure Account

Now that we have our Microsoft Azure subscription and account up and running and Azure Cloud Shell configured, it is time to move on to something very important: making sure your Azure account is secure enough.

Securing your Microsoft Azure Global Administrator account is an important task that should not be ignored.

If your Azure account is compromised and broken into your data can be exposed and exploited by malicious people.

Therefore, in this section I will show you methods that will help secure your Azure Global Administrator account and your tenant in general.

Enable Multi-Factor Authentication (MFA)

According to global security research that was released back in 2018, enabling multi-factor authentication (MFA) on your account and your user accounts can reduce the risk of their being broken by 80 percent. This is a quick win that goes a long way in improving the overall security status of your Azure account.

By default, Microsoft Azure doesn't charge extra fees for enabling MFA for the Global Administrator account, which shows the commitment Microsoft has for security and making sure customer data is safe.

To enable MFA on your Global Administrator account, open Azure Active Directory from the Azure portal.

Click on All Services, select Identity, and click on Azure Active Directory. Figure 1-23 shows the Azure Active Directory icon.

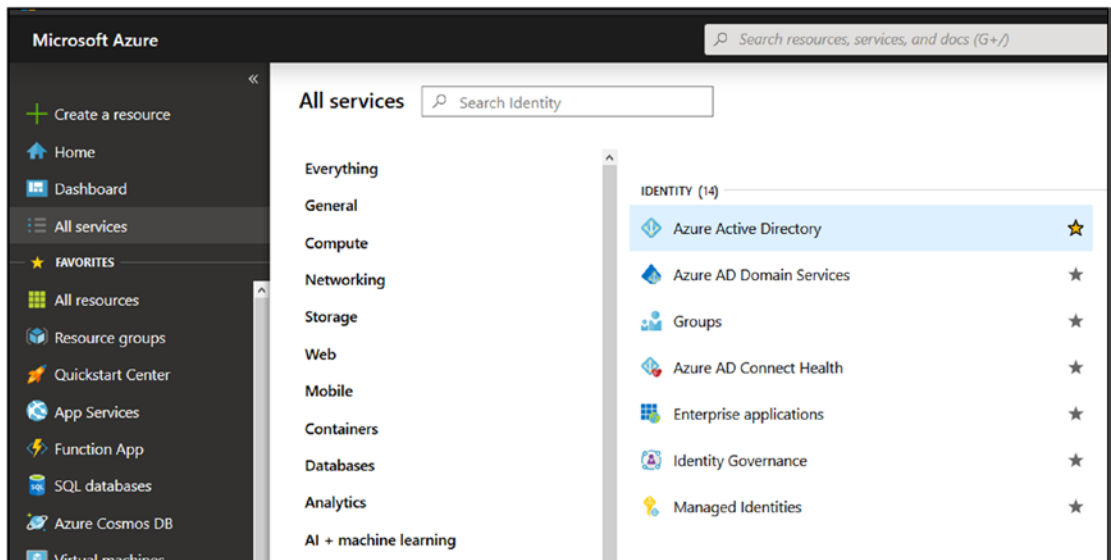


Figure 1-23. Azure Active Directory

From the Azure Active Directory main page, click on Users, as shown in Figure 1-24.

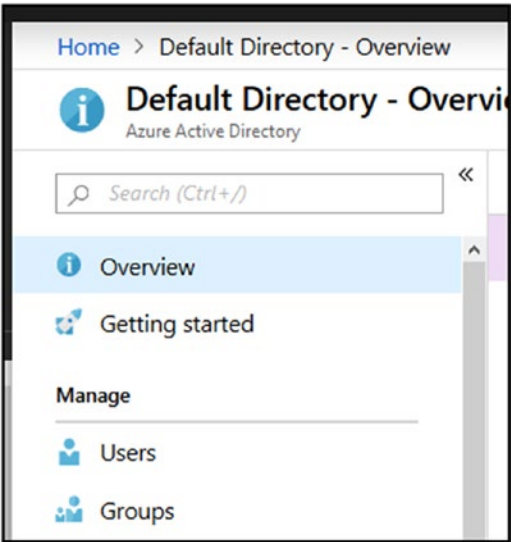


Figure 1-24. Azure Active Directory Users menu

From the Users menu, select your Global Administrator user and click on Multi-Factor Authentication in the top-left menu, as shown in Figure 1-25.

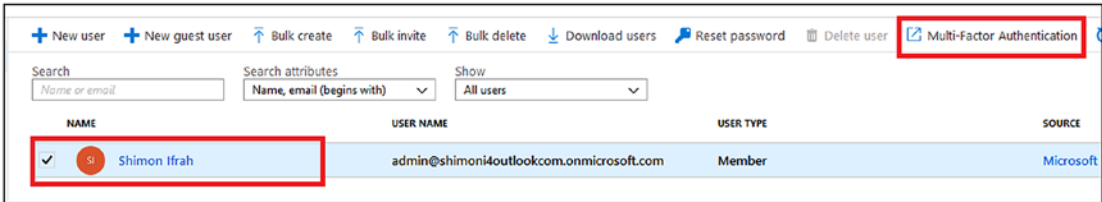


Figure 1-25. Enable Multi-factor authentication

After you click on the Multi-Factor Authentication link, you will be redirected to the Azure Active Directory Multi-Factor Authentication page, where you can use the right-hand menu to enable MFA.

Figure 1-26 shows the MFA page and that my account has MFA disabled.

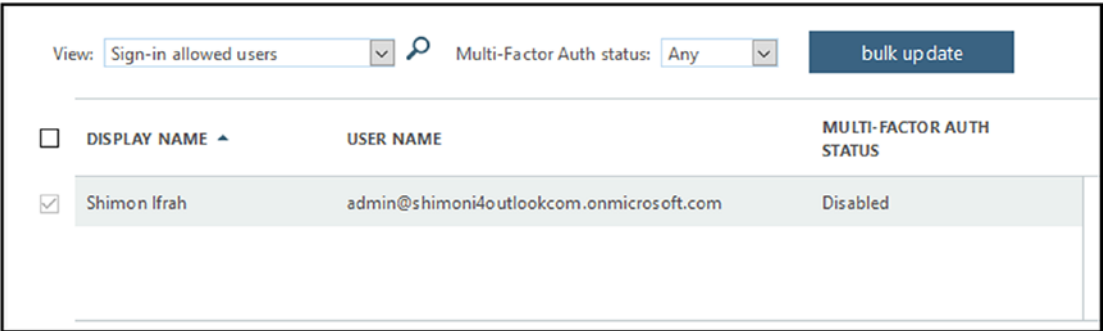


Figure 1-26. Azure MFA page

To enable MFA for my account, I need to click on Enable from the Quick Steps menu on the right-hand side.

Figure 1-27 shows the Quick Steps menu where you enable MFA.

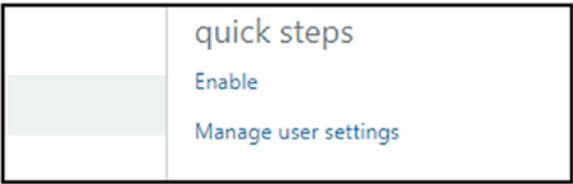


Figure 1-27. Enable MFA from the Quick Steps menu

Note If your Azure account is not part of an Office 365 subscription and is a Microsoft Account, please enable MFA from the Microsoft Account Security settings page using the following URL:

<https://account.microsoft.com/security/>

Check Global Administrator Accounts

Another good practice that can help you secure your Azure account and subscription is to check on a regular basis who has Global Administrator permission in your Azure tenant.

Microsoft strongly recommends you have less than five Global Administrator accounts inside your tenant.

You should not give administrators permissions they don't need, and always try to give them the exact permissions warranted by their role.

Note Global Administrator is also known as Company Administrator.

To review and check who has Global Administrator permissions for your Azure subscription, open Azure Cloud Shell and use the PowerShell experience.

From PowerShell, run the following command (Figure 1-28):

```
Get-AzureADDirectoryRole | Where { $_.DisplayName -eq "Company Administrator" } | Get-AzureADDirectoryRoleMember | Ft DisplayName
```



```
PS Azure:> Get-AzureADDirectoryRole | Where { $_.DisplayName -eq "Company Administrator" } | Get-AzureADDirectoryRoleMember | Ft DisplayName
DisplayName
-----
Shimon Ifrah
```

Figure 1-28. *Display Global Administrator accounts*

In Chapter 6, we will take a deep dive into Azure security best practices and how to mitigate risks and threats.

Azure Container Services

In this section, we will explore the four pillar services in Microsoft Azure that enable us to deploy containerized applications.

In the next few chapters, I will cover each of them in greater detail.

The last service, Web App for Containers, is an extension of the Azure Container Instances (ACI) service and will be covered within the ACI chapter.

Microsoft Azure allows small and large businesses the option to deploy containerized applications in both large and small scales.

Azure offers multiple services that cover any business needs and at any scale.

The core Microsoft Azure Container Services that we will cover in this book are:

Azure Kubernetes Services (AKS)

Azure Container Instances (ACI)

Azure Container Registry (ACR)

Web App for Containers

This section will introduce each chapter, which will have more details.

Azure Container Instances (ACI)

Azure Container Instances (ACI) is the most basic, simple, and easy-to-manage container service on Microsoft Azure.

With ACI, we can quickly deploy containers into an isolated environment without managing servers, storage networking, and container orchestration.

ACI is also integrated with Azure Container Registry (ACR) and can pull images directly from ACR without any complex configuration.

The start time of containerized applications on ACI is seconds compared to minutes when running your applications on a virtual machine.

Containers running in ACI have access to a public IP address and external access using a DNS hostname that is routable over the internet.

ACI also allows us to scale out containers in case usage goes up and the application needs more resources to perform well.

Billing is based on seconds and provides the most cost-effective solution for containerized application deployments; however, if you need your containers running 24/7 ACI is not the most cost-effective solution.

If you need to use persistent storage for your ACI application, you can mount a storage volume using the Azure file share and save the state of your containers in case they are restarted, crash, or stop.

All workloads running in ACI are secured and isolated using hypervisor-level security.

ACI supports both Windows and Linux containers.

You can deploy containers to ACI using the Azure portal and Azure CLI.

With ACI, we can also create multi-container groups that allow us to combine the application with multiple roles and services that share the same storage network host.

ACI also works with Web App for Containers, which uses Azure App Service to host web applications in the Azure shared infrastructure.

With Web App for Containers, we can run web applications inside Linux or Windows Containers.

In Chapter 3, we will deep dive into ACI and deploy containers.

Azure Kubernetes Services (AKS)

The Microsoft Azure Kubernetes Service (AKS) is Azure's fastest-growing service after virtual machines.

It provides state-of-the-art service-based container orchestration using the Kubernetes open-source project.

AKS is an enterprise-grade orchestration solution that can scale and handle extremely large deployments of containerized applications.

With autoscaling features, AKS can increase the number of nodes, storage volumes, and networking without manual or human intervention.

With AKS, we can automate daily, weekly, and monthly tasks and reduce the need to run jobs and tasks manually.

There is continuous integration and continuous delivery (CI/CD) of pipelines directly into AKS using API.

AKS tools are fully integrated with Visual Studio 2019, Visual Studio Code, and Azure DevOps, where you can connect your application directly to the preceding tools and deploy using a single click.

With Azure Active Directory integration support, we can control access to AKS using Azure AD users, groups, and RBAC roles.

With Dev Spaces, we can deploy code to a subsection in AKS that is dedicated to testing and development without affecting production code.

We can also get insight into AKS performance using Azure Monitoring.

Azure AKS is the only cloud solution (compared to AWS and GCP) that supports Windows Containers (currently in preview) workloads in Kubernetes.

AKS is fully integrated with Azure Container Registry (ACR) and allows fast retrieval of container images.

As of the time of writing, AKS is available in twenty-eight Microsoft Azure regions and growing.

Azure Container Registry (ACR)

The Azure Container Registry (ACR) service is an enterprise-grade private container registry.

ACR allows us to store Docker container images in a secure private registry and use advanced security features to protect the images.

Using Azure infrastructure, ACR is optimized and can easily scale up and down depending on usage and storage.

With ACR, we can opt to use security keys and admin account or use Azure Active Directory authentication and control access to the registry using users and groups.

Once enabled and configured, we can deploy Docker images to Azure AKS, ACI, and Web App for Containers using a single click.

ACR also supports high availability and geo-replication of a single ACR registry to multiple regions across the globe.

We manage ACR using the same familiar Docker CLI commands without needing to use other command-line tools.

ACR comes in three stock-keeping units (SKUs) and price points depending on size and features.

The three ACR SKUs are:

Basic

Standard

Premium

The main difference in the SKUs is the storage. However, the geo-replication option is only available in the Premium SKU.

Sometimes it is better to use the Premium SKU instead of deploying multiple ACRs in different regions.

Figure 1-29 shows the Azure Container Registry (ACR) pricing page and the three SKUs (Basic, Standard, and Premium).

Prices in the figure are for an ACR registry in a Central US datacenter and in USD.

Region:
Central US

Currency:
US Dollar (\$)

Pricing details

	BASIC	STANDARD	PREMIUM
Price per day	\$0.167 ¹	\$0.667 ¹	\$1.667 ¹
Included storage (GiB)	10	100	500 Premium offers enhanced throughput for docker pulls across multiple, concurrent nodes
Total web hooks	2	10	100 (additional available upon request)
Geo Replication	Not supported	Not supported	Supported \$1.667 ¹ per replicated region

Figure 1-29. ACR pricing page

Summary

In this chapter, we covered the ins and outs of setting up a Microsoft Azure subscription and how to use the free \$200 USD credit for thirty days.

We also learned how to get started and set up Azure Cloud Shell, a web-browser command-line utility running inside a container and hosted in Azure.

In the next few chapters, we will use Azure Cloud Shell to deploy many of the Azure Container services.

After setting up Azure Cloud Shell, we learned how to secure our Azure environment using quick-win methods, and the main one that we explored was using MFA to secure our Azure Global Administrators account. It is highly recommended you use MFA to secure all of your Global Administrator accounts.

And in the last section of this chapter, we explored the following main Azure Container services, which we will cover in this book:

- Azure Kubernetes Services (AKS)
- Azure Container Instances (ACI)
- Azure Container Registry (ACR)

In our next chapter, we will deep dive into the world of Azure Container Register (ACR) and use it to store Docker container images using Docker CLI.

We will learn how to push and pull Docker images to ACR and manage them using Azure CLI, Azure Cloud Shell, and the Azure Portal.

CHAPTER 2

Store and Manage Docker Container Images on Azure Container Registry (ACR)

In this chapter, we will learn how to get started with Azure Container Registry (ACR) and use it to store, upload, and download Docker images.

This chapter will cover the following:

- Overview of Azure Container Registry (ACR)
- Installing the Docker Desktop client
- Installing Visual Studio Code and the Docker extension
- Enabling the Azure Container Registry (ACR) service in Azure
- Authenticating to ACR from a local machine
- Tagging and pushing Docker images to ACR
- Pulling a Docker image from ACR to a local machine
- Securing our ACR container registry

Overview of Azure Container Registry (ACR)

Azure Container Registry (ACR) is a private container image registry used to store Docker images in a secure and high-availability environment. The underlying infrastructure and technology behind ACR are managed by Microsoft Azure, while the endpoints, access, and image repositories are managed by the tenant, who is us.

ACR is an elastic service that can grow easily based on our needs using the following plans:

- Basic
- Standard
- Premium

Figure 2-1 shows the pricing details of all the plans available in ACR.

Region:

Central US

Currency:

US Dollar (\$)

Pricing details

	BASIC	STANDARD	PREMIUM
Price per day	\$0.167 ¹	\$0.667 ¹	\$1.667 ¹
Included storage (GiB)	10	100	500 Premium offers enhanced throughput for docker pulls across multiple, concurrent nodes
Total web hooks	2	10	100 (additional available upon request)
Geo Replication	Not supported	Not supported	Supported \$1.667 ¹ per replicated region

Figure 2-1. ACR pricing details

Most developers will find the basic tier enough to test various applications and code; however, large businesses will probably opt for the Premium plan, which offers more storage and global replications.

We can manage ACR using Azure CLI (with Azure Cloud Shell) or the Azure portal.

Because the underlying infrastructure is managed by Azure, our role in managing ACR is limited and doesn't require complex operation procedures.

To push and pull Docker images to and from ACR, we use the Docker CLI command-line utility, which is great as it doesn't require us to learn how to use another tool.

Set Up Microsoft Azure Container Registry (ACR)

Before we can start using ACR, there are a few requirements we need to follow that will ensure we can connect to ACR and push and pull images. The tool that allows us to connect to ACR is the Docker CLI client.

Docker CLI allows us to do the following:

- Connect to ACR using the Docker login command

- Tag images that we need to push to ACR

- Push and pull images from and to ACR

We can use Azure CLI and Azure DevOps to connect to ACR.

Install Docker Desktop

Docker Desktop is a client application that is available on all major platforms (Windows, Linux, and macOS).

To get started, install the Docker client on your client operating system by going to the following URL:

<https://www.docker.com/products/docker-desktop>

From the Docker Desktop page, click “Download Desktop for Mac and Windows,” as shown in Figure 2-2, and follow the prompts.

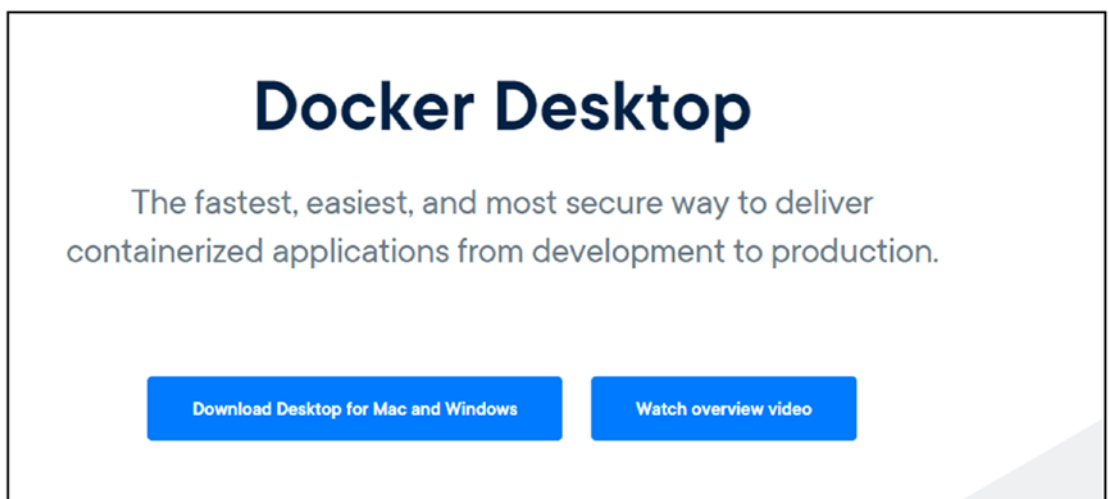


Figure 2-2. Docker Desktop download page

To install Docker on Linux, please refer to the following URL: <https://docs.docker.com/install/linux/docker-ce/ubuntu/>.

Creating Azure Container Registry (ACR) Using the Portal

In this section, we will create our Azure Container Registry (ACR) using the Microsoft Azure portal. Azure also allows for creating an ACR using Azure CLI directly from Azure Cloud Shell.

To create an Azure Container Registry (ACR), I log in to the Azure portal. From the portal, I click on “All services” in the left-hand menu.

In the Services list, I click on “Containers” and then on “Container registries.” Figure 2-3 shows the Container registries service in the Azure portal.

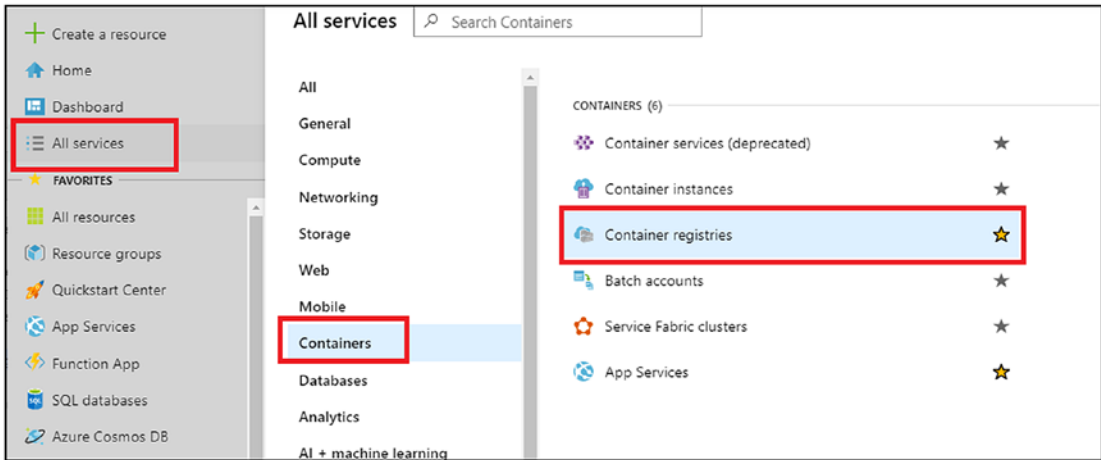


Figure 2-3. Azure Container registries service

On the Container Registries page, I click on “Add” to start the creation process, as shown in Figure 2-4.

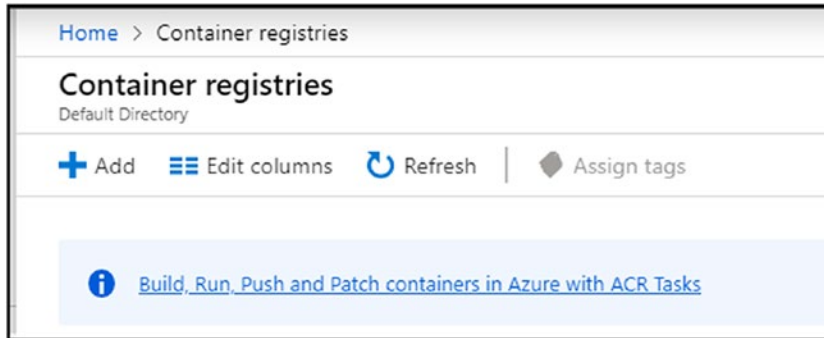


Figure 2-4. Add Container registry

To create my first ACR, I will fill in the details, as shown in Figure 2-5, and as outlined in the following list:

Registry Name — This is the name of the registry (needs to be globally unique) and the routable address to get to it.

Subscription — This is the Azure subscription the registry will be billed under; if you have only one subscription you can ignore this part.

Resource Group — Here we have the option to create a new resource group or use an existing one; in my case, I am creating a new one.

Location — This is the Azure region in which the registry will be hosted; this part is very important, and your ACR should be hosted in the same region in which your applications are located.

Admin User — This option needs to be enabled in order to access the registry.

SKU — This part is very important because, by default, Azure selects the Standard SKU, which costs more. I recommend you start with the Basic SKU and upgrade if needed.

Figure 2-5 shows the Create Container Registry page with the details I used.

Home > Container registries > Create container registry

Create container registry

* Registry name

DeployRG

✓

.azurecr.io

* Subscription

Free Trial

▼

* Resource group

(New) Apps

▼

Create new

* Location

West US

▼

* Admin user ⓘ

Enable

Disable

* SKU ⓘ

Basic

▼

Create

Automation options

Figure 2-5. Create Container Registry page

When the ACR is ready, it will show up on the Container Registries page, as shown in Figure 2-6.

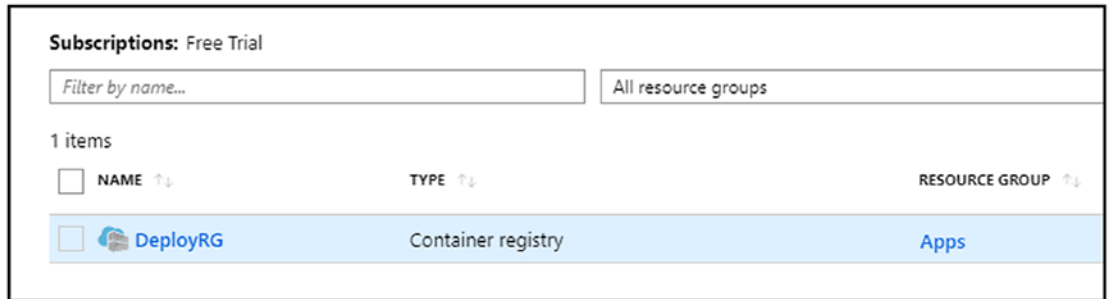


Figure 2-6. Azure Container Registries page

Creating Azure Container Registry Using Azure CLI

Microsoft Azure also gives us the option to create a new Azure Container Registry (ACR) using Azure CLI in case we want to create it programmatically without using the Azure portal.

To create a new ACR using the same details I used in the Azure portal, I will run the following command from Azure Cloud Shell. Because I am using Azure CLI, I can use both the Bash and PowerShell Shell experiences.

The following command will create a resource group called Apps in the Azure West US Region:

```
az group create --name Apps --location westus
```

The following command will create an ACR called DeployRG with admin access enabled in the Apps resource group and that uses the Basic SKU:

```
az acr create --resource-group Apps --name DeployRG --admin-enabled true
--sku Basic
```

To view the newly created ACR, I will run the following command:

```
az acr list --output table
```

Push Docker Images to Azure Container Registry (ACR)

Having created our first ACR, we will now push a Docker image to it using Docker CLI. Before we can push or pull images from ACR, we need to copy the login details to the ACR from the Azure portal.

The login details for our ACR are in the Settings section of the ACR under the Access Keys. Figure 2-7 shows the Access Keys' section with the details needed to log in to ACR. We will need to copy the following information:

Login Server — This is the address of our ACR registry.

Username — This is the username.

Password — This is the password used to log in to the ACR (only one password is needed).

Admin User — Make sure Admin User is set to “Enable.”

Copy all the details, and we will use them to log in and connect to ACR using the Docker CLI.

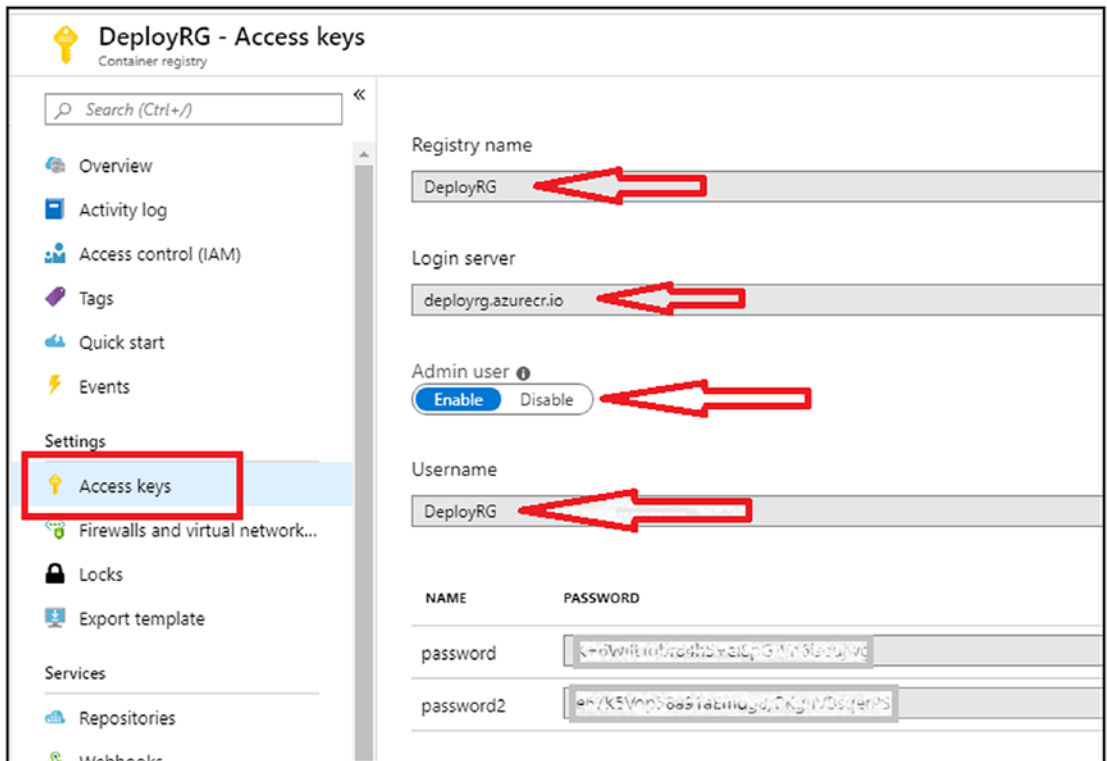


Figure 2-7. Access keys

Connect to ACR Using Docker

Now that we have enabled ACR and have the login details, we will use the Docker CLI to log in and push an image to ACR.

The process to push an image to ACR is as follows:

- Connect to ACR using Docker login

- Tag the image using Docker tag

- Push the image using Docker push

To log in to ACR, I will use the Docker CLI using PowerShell and use the following command, as shown in Figure 2-8.

Note Please use the login server details.

```
docker login deployrg.azurecr.io
```

```
PS C:\> docker login deployrg.azurecr.io
Username: deployrg
Password:
Login Succeeded
PS C:\>
```

Figure 2-8. Docker login command

Once the login has succeeded, I will tag a local image I have on my machine. In my case, I have an Nginx image that I will tag.

Note To download the same Nginx image, I can use the following command:

```
docker pull nginx
```

```
docker tag nginx deployrg.azurecr.io/nginx
```

Once the image is tagged, I can use the following command, as shown in Figure 2-9, to push the image to ACR:

```
docker push deployrg.azurecr.io/nginx
```

```
PS C:\> docker login deployrg.azurecr.io
Username: deployrg
Password:
Login Succeeded
PS C:\> docker tag nginx deployrg.azurecr.io/nginx
PS C:\> docker push deployrg.azurecr.io/nginx
The push refers to repository [deployrg.azurecr.io/nginx]
ce3539cc1849: Pushing [=====>] 3.584kB
16d1b1dd2a23: Pushing [>] 536.1kB/56.77MB
2db44bce66cd: Pushing [>] 525.3kB/69.21MB
```

Figure 2-9. Docker push command to ACR

To view the image we just uploaded to ACR, perform the following steps:

Log in to the Azure portal and open the container registry.

In the registry, click on “Repositories.”

As shown in Figure 2-10, the Nginx image appears in the repository.

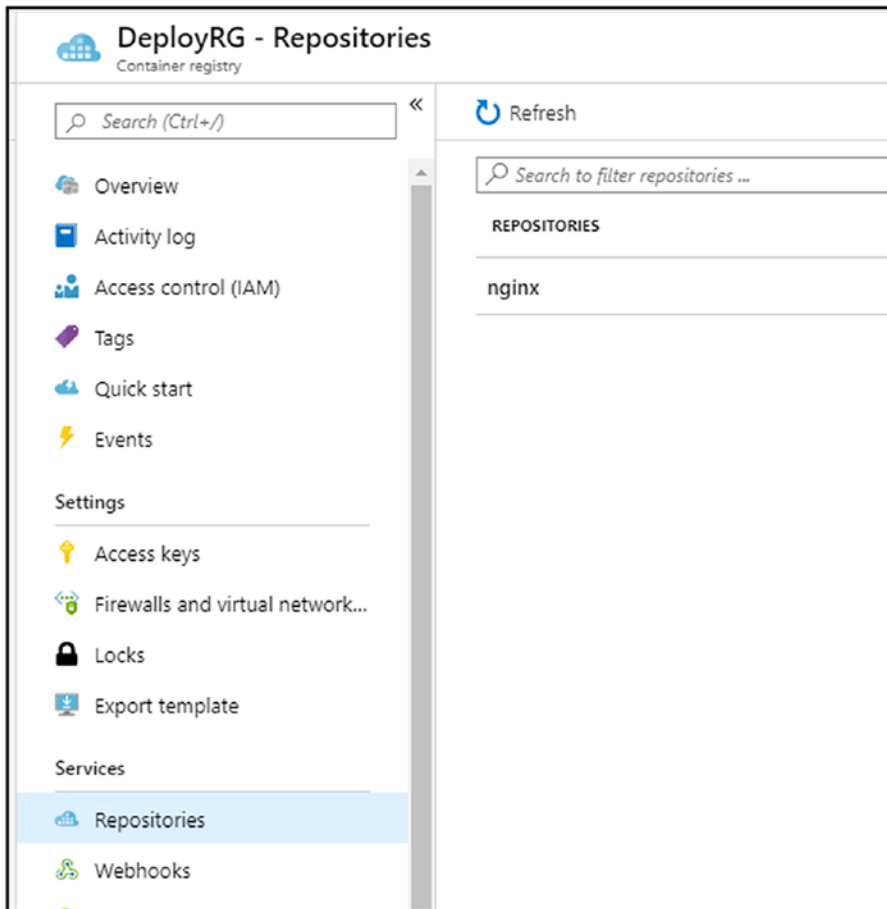


Figure 2-10. ACR repositories

If I click on the image, I will see all the available tags that belong to the image. In my case, and as shown in Figure 2-11, I have the “latest” tag available.

If I click on the action menu (...) on the left-hand side of the tag, I will see five options that are available for me to use. I will not cover them now; however, using this menu, we can deploy the image to ACI or to a web app.

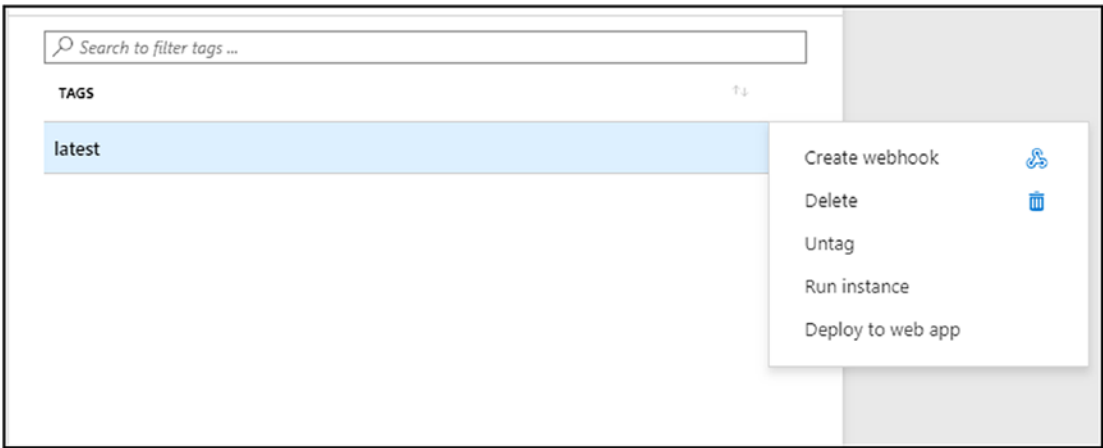


Figure 2-11. ACR tags and action menu

Pull Images from Azure Container Registry (ACR)

So far, we have learned how to enable ACR and use the Docker login to push a local image to an ACR.

Now, it is time to pull an image that is stored in ACR to our local machine using the Docker pull command. The process of pulling an image from ACR is much simpler than that for pushing an image because we only need to log in to ACR and use the pull command.

The process to pull an image from ACR is as follows:

- Use Docker login to log in to ACR.

- Use Docker pull command to download the image.

To get started, log in to the Azure portal and open your ACR. From the ACR, click on “Repositories” and locate the image you would like to download.

Click on the image tag; in my case, I will click on “latest.”

On the image blade page, you will see the full Docker pull command. Copy the command, which includes the image name and address.

Figure 2-12 shows the Docker pull command for my Nginx image.

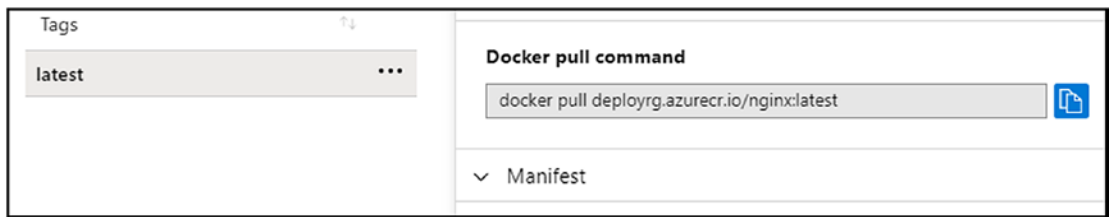


Figure 2-12. *Docker pull command*

Now that I have the image name, I will open PowerShell on my machine and log in to the registry using the following command:

```
docker login deployrg.azurecr.io
```

Once authenticated, I will use the command copied from Figure 2-12 to download the image:

```
docker pull deployrg.azurecr.io/nginx:latest
```

Note To log out of the registry from Docker CLI, I will use the following:

```
docker logout deployrg.azurecr.io
```

It is recommended you log out from ACR after finishing working with it.

At this point, we have covered the processes of pushing and pulling Docker images from to and from ACR. As shown, the process is not complex; however, it requires you to follow the discussed processes.

Manage and Secure Azure Container Registry (ACR)

In this section, we will learn how to manage Azure Container Registry (ACR) using Visual Studio Code (VS Code). VS Code is an open-source code editor developed by Microsoft. It works with major operating systems (Windows, Linux, and macOS).

It is highly customizable and allows us to install extensions, debug tools, and change keyboard shortcuts and other preferences. By default, it includes support for source code control (Git), GitHub, and intelligent code completion.

Install VS Code

To get started with VS Code, we need to download it from the following URL: <https://code.visualstudio.com/>.

You can see the VS Code homepage in Figure 2-13.

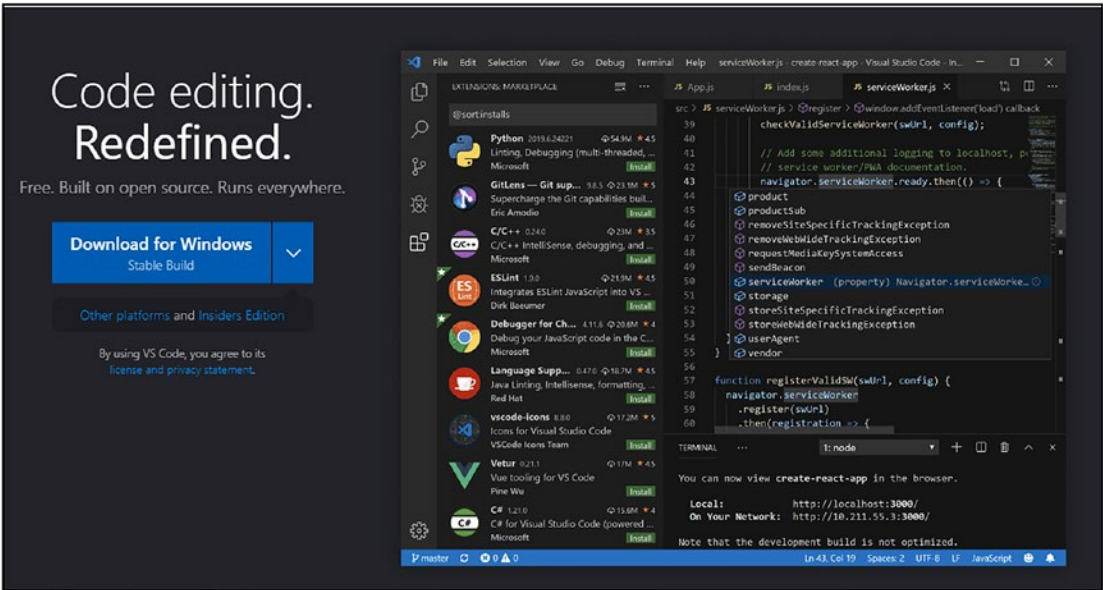


Figure 2-13. VS Code homepage

Install Docker Extension

To install it, download the support version and follow the installation steps. Once installed, start VS Code and click on the Extensions icon on the left-side menu.

From the Extensions page, type Docker in the search box, as shown in Figure 2-14.

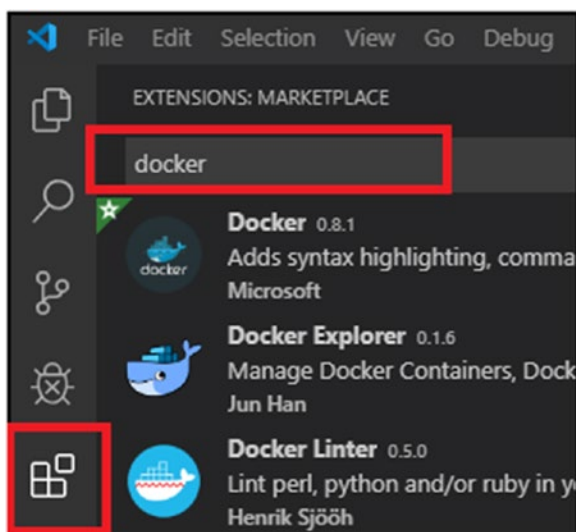


Figure 2-14. Docker extension in VS code

You should select the Docker extension released by Microsoft, as shown in Figure 2-15.

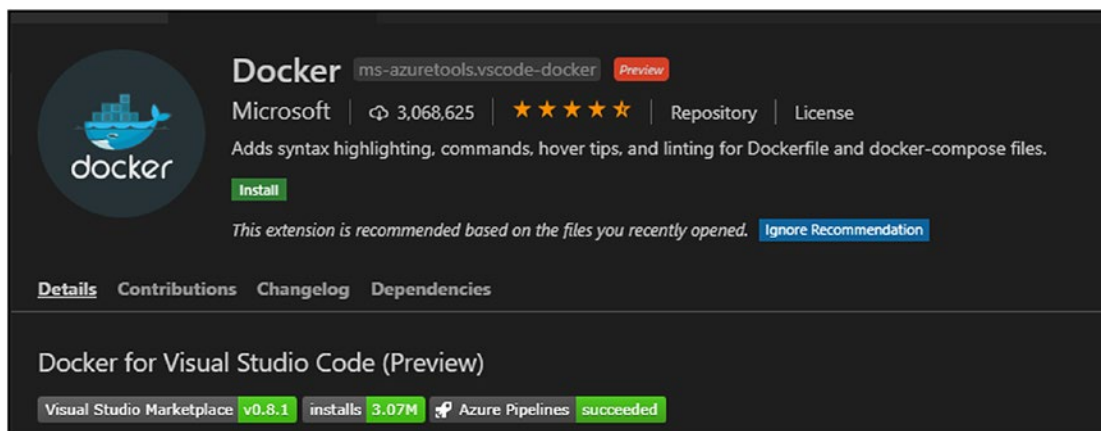


Figure 2-15. Docker extension by Microsoft

To install the extension, click “Install,” and once installed, restart VS Code. When you start VS Code, you will see the Docker icon on the left-side menu, as shown in Figure 2-16.

The Docker extension allows us to manage the following Docker components:

- Containers — running and stopped containers
- Images — Docker images
- Registries — Azure Container Registry (ACR) and other providers
- Networks — Local Docker networks
- Volumes — Docker volumes

The Docker extension allows us to better manage our local development environment and get better insight into our resources. Figure 2-16 shows all the Docker extension components.

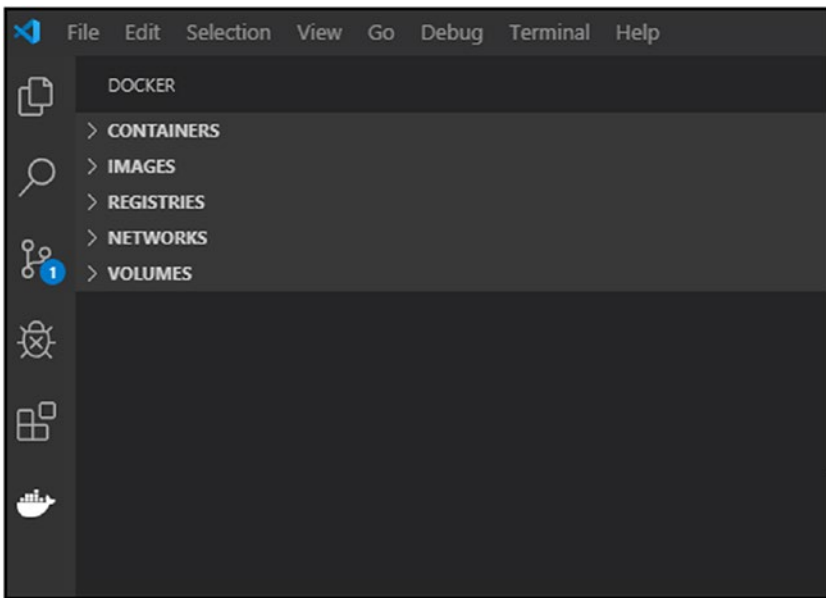


Figure 2-16. Docker extension components

Manage Containers with VS Code

Managing containers with VS Code is very easy because the main management command and utilities are integrated into the extension. In Figure 2-17, you can see my Nginx container. When I right-click on the container, I have the option to do the following:

- View the container logs

- Attach to the container using a shell and see the console

- Inspect the configuration of the container

- Stop the container

- Restart the container

- Remove\delete the container

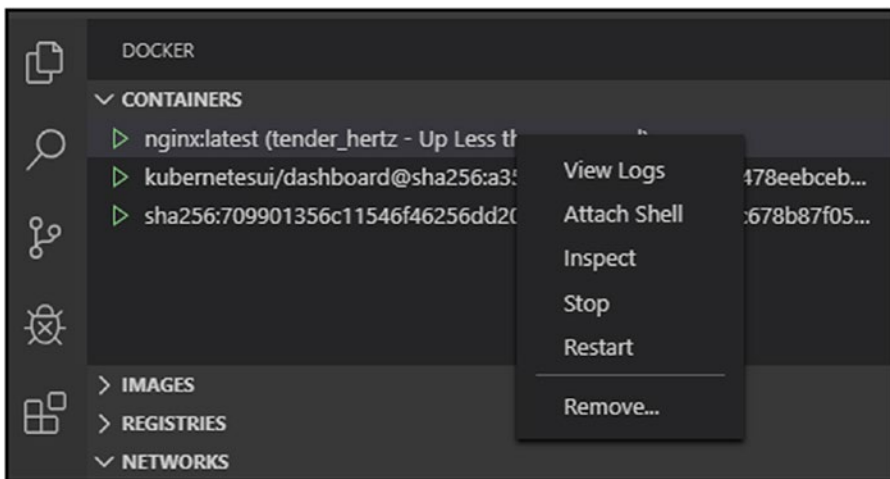


Figure 2-17. Container menu in VS Code

Manage Docker Images with VS Code

We can also manage our Docker images with VS Code in the same way we manage containers. In Figure 2-18 you can see the Images section. If I right-click on the image I get the following options:

- Run a container in detached mode
- Run a container and attach to the shell
- Inspect the image properties
- Push an image to a registry, including ACR
- Tag an image
- Remove\delete an image

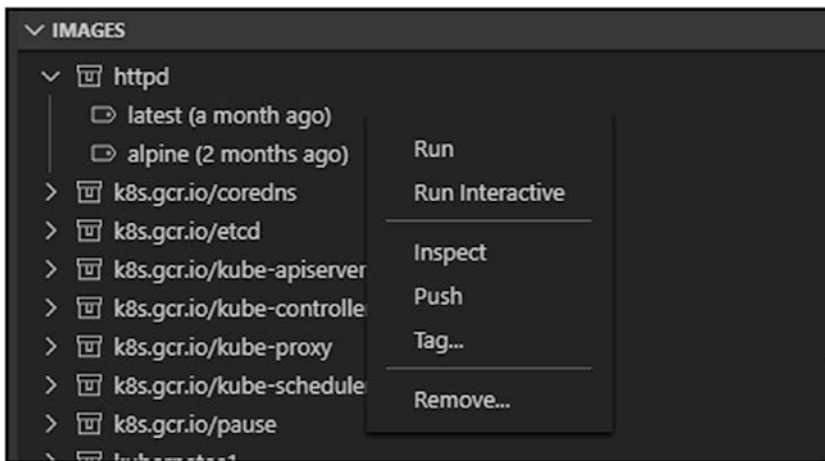


Figure 2-18. Manage images with VS Code Docker extension

Manage a Container Registry with VS Code

The Docker extension for VS Code gives us the option to manage our container registry directly from VS Code without using the portal or Azure CLI.

The process of connecting our ACR registry to VS Code is simple and takes just a few minutes. To connect VS Code to our ACR container registry, click on “Connect Registry...” in the Registries section, as shown in Figure 2-19.

Then, click on “Azure” in the menu.

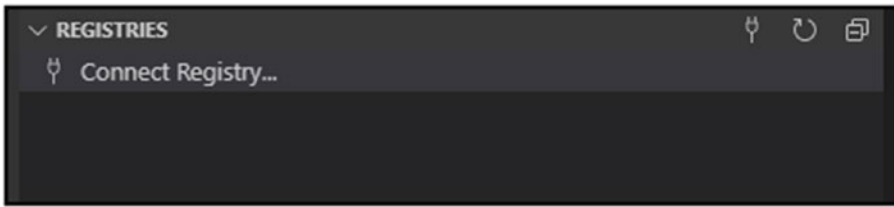


Figure 2-19. *Connect Registry*

Before we can connect the ACR registry, we need to install the Azure Account Extension. To do so, click on “Install Azure Account Extension...,” as shown in Figure 2-20.

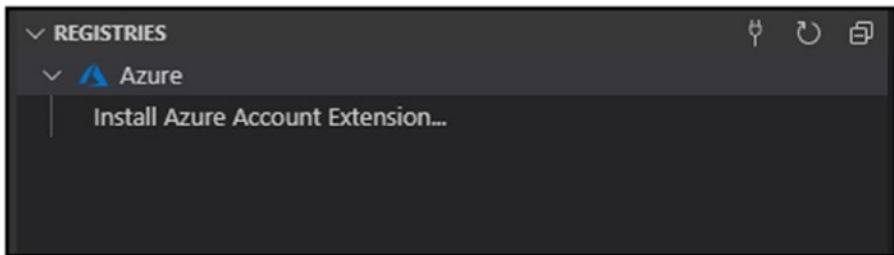


Figure 2-20. *Install Azure Account Extension*

From the Azure Account Extension screen, click on “Install” and wait for the installation to finish. Figure 2-21 shows the Azure Account Extension screen.

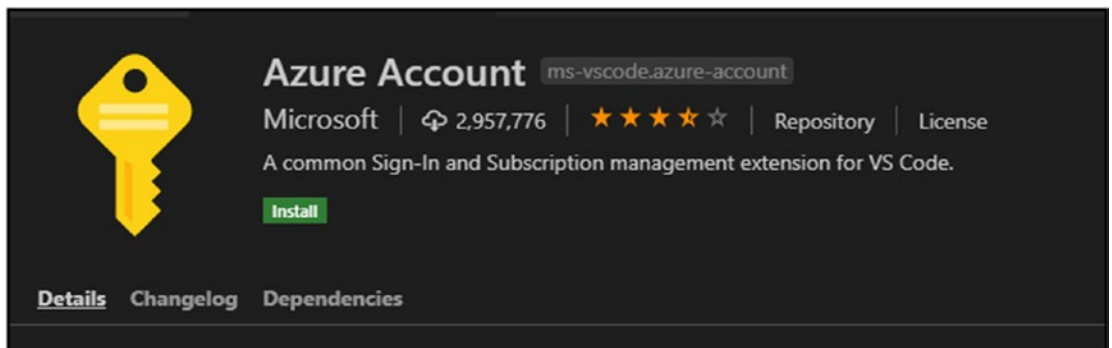


Figure 2-21. *Azure Account Extension installation screen*

After the installation is complete, click on the “Refresh” button next to the Registries section (located on the right side), as shown in Figure 2-22.

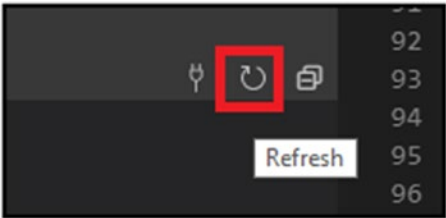


Figure 2-22. “Refresh” button

After refreshing the Registries section, you will see the option to sign into Azure, as shown in Figure 2-23.

Go ahead and click on Sign into Azure...

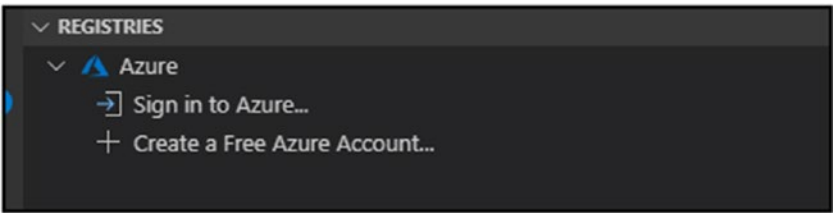


Figure 2-23. Sign in to Azure option

Once this is clicked, your browser will open a login page, where you will need to type your Azure username and password, as shown in Figure 2-24.

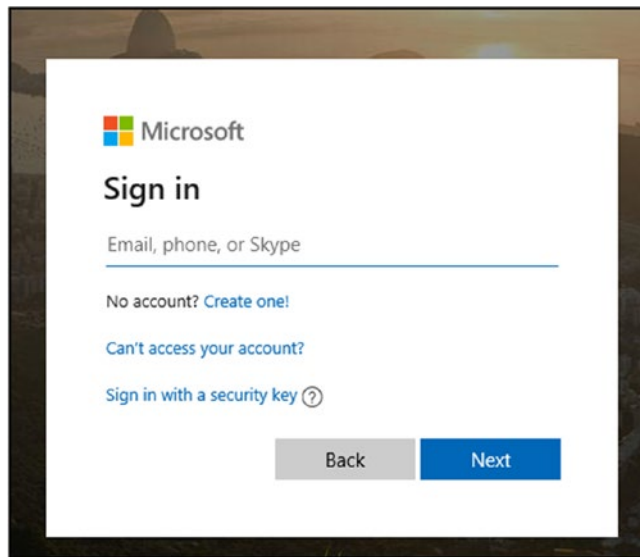


Figure 2-24. Login page

After you log in and authenticate successfully, you will see your ACR registry with all the images stored in it. Figure 2-25 shows my ACR registry and my Nginx Docker image that I pushed to the registry in the previous section.

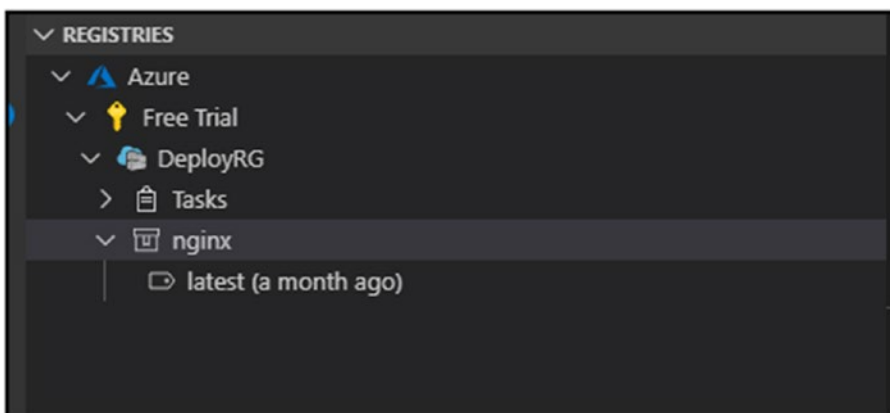


Figure 2-25. ACR registry in VS Code

If I right-click on the Nginx latest image tag, I will get the options menu, as shown in Figure 2-26. The Docker extension gives us many management options that allow easy management of our Docker images.

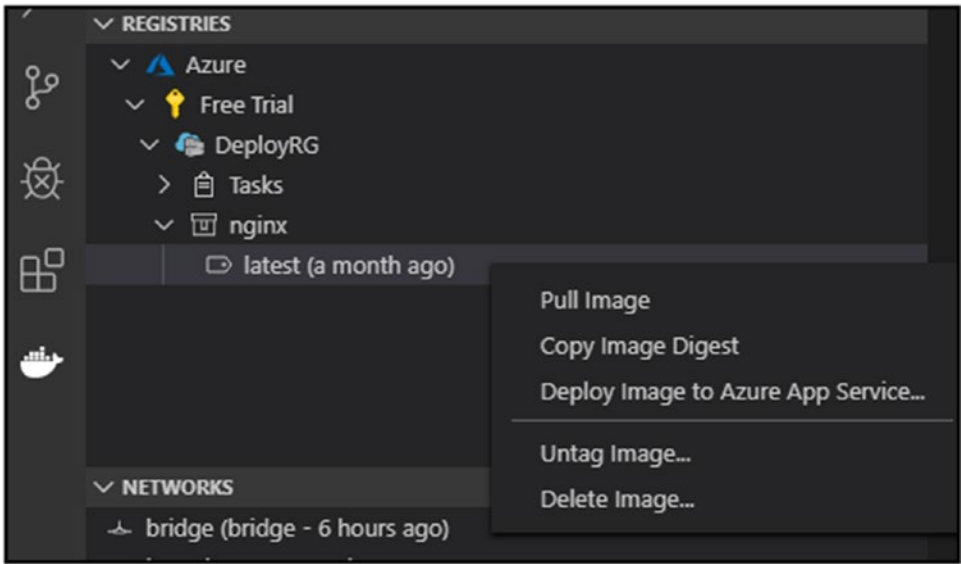


Figure 2-26. Registry right-click menu

With the VS Code Docker extension, we can do the following:

- Pull an image directly to my machine without using the Docker pull command
- Copy the image digest
- Deploy the image to Azure App Service as a web app
- Untag image that was tagged before
- Delete image from ACR without logging in to the portal

The last thing I would like to show you is how to disconnect and log off of your ACR registry from VS Code. This option is very important in case you are managing multiple registries across several tenants.

From VS Code, click on “View” in the top menu, or use Ctrl + Shift + P. Click on “Command Palette...,” as shown in Figure 2-27.

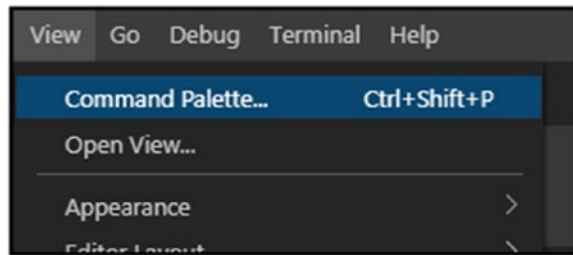


Figure 2-27. VS Code command palette

In the Command Palette text box, type `Azure: Sign Out`.

The command will log off and disconnect your Azure AD account from VS Code and will allow you to connect to another tenant. Figure 2-28 shows the command.

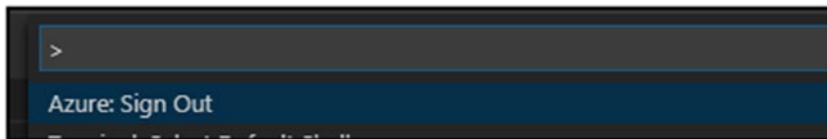


Figure 2-28. Azure Sign Out command

Securing Azure Container Registry (ACR)

In this subsection, we will learn how to secure our ACR registry using some advanced security features that are only available in the Premium SKU plan. Before we can show you the advanced security features of ACR, we need to upgrade the SKU plan.

Upgrade ACR SKU Plan

To upgrade the SKU plan, we will use the following steps.

From the ACR Overview page located in the portal, click on “Update,” as shown in Figure 2-29.

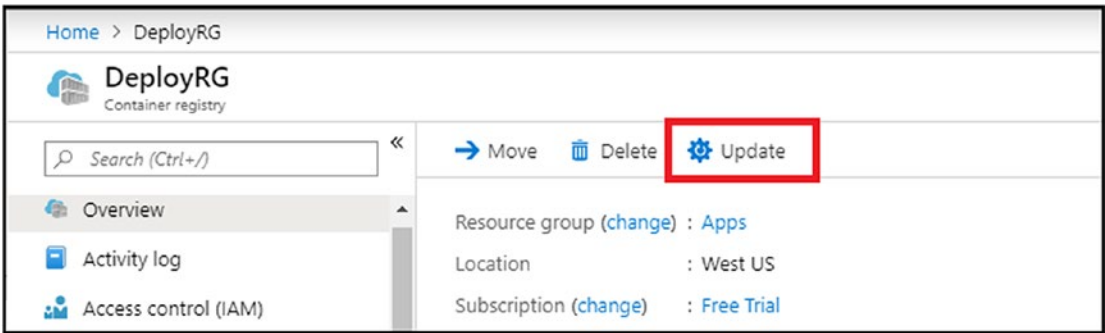


Figure 2-29. Update ACR

From the Update Container Registry page, I will select the Premium plan under the SKU drop-down menu and click on “Save.”

Figure 2-30 shows the Update Container Registry page.

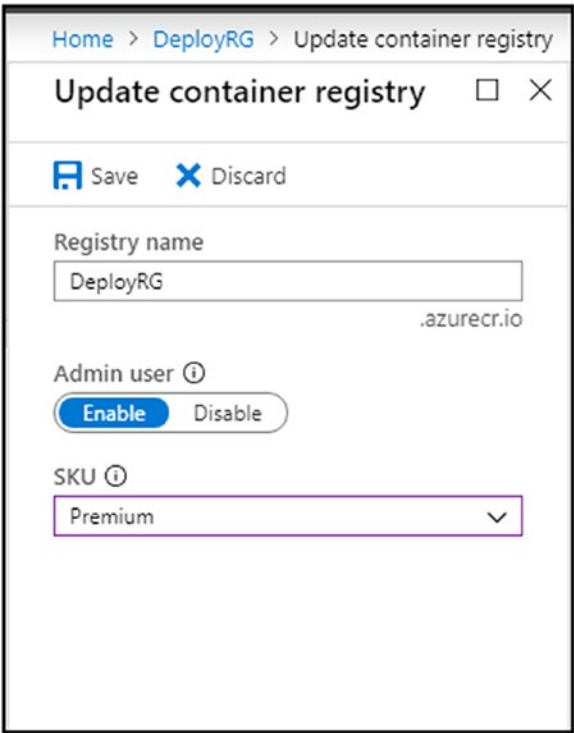


Figure 2-30. Update Container Registry page

After updating the ACR SKU plan, I will click on the “Firewalls and virtual networks (Preview)” option located under Settings in the ACR’s left-hand menu.

Figure 2-31 shows the Firewall and Virtual Networks (Preview) page.

Note The service is still in preview mode, but it is expected to change to general availability in the coming months.

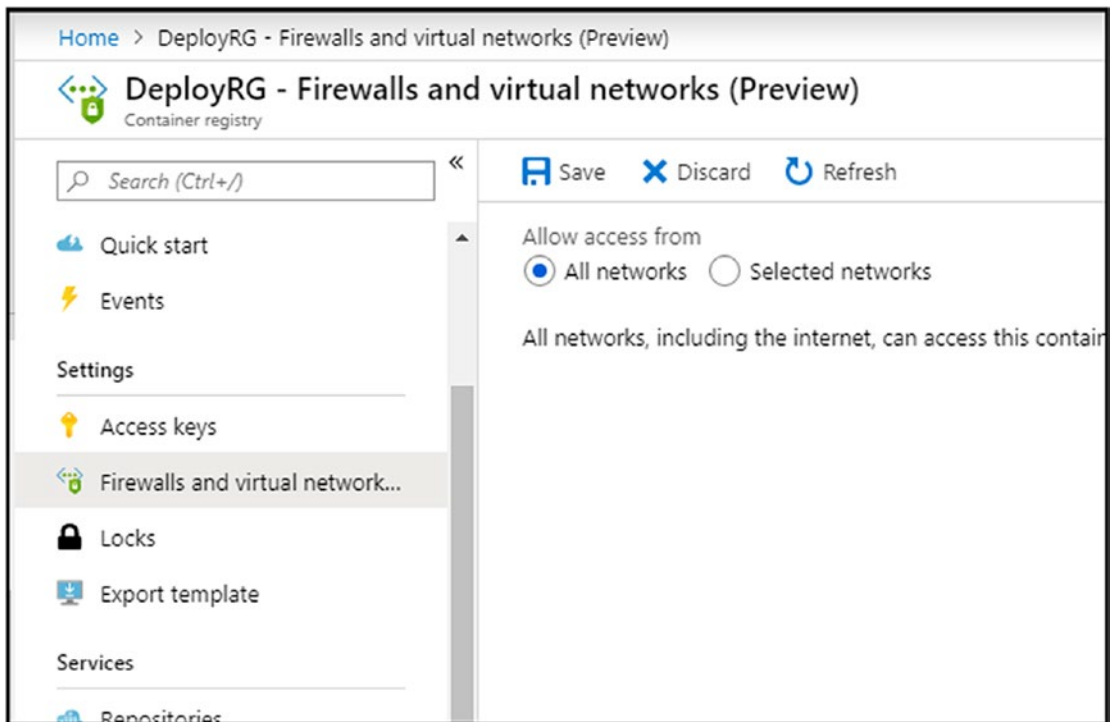


Figure 2-31. *Firewall and Virtual Networks page*

The Container Registry Firewalls and Virtual Networks feature allows us to limit access to our ACR from specific networks and subnets. By default, an ACR is accessible via internet from any address, and the only protection we have is the username and password that prevents unauthorized access.

This feature adds another layer of security on top of the password protection in case the security keys have been compromised; intruders will not be able to access the registry unless they are in an allowed network.

To enable the service and only allow specific networks or IP addresses access to the ACR, open the Firewalls and Virtual Networks settings.

Click on “Selected networks” for the “Allow access from” option.

To add a specific IP address, use the Firewall section and type the IP address under Address Range.

We also have the option to add existing virtual networks or new ones under the Virtual Networks option.

When completed, click on “Save.”

Figure 2-32 shows the Firewalls and Virtual Networks options.

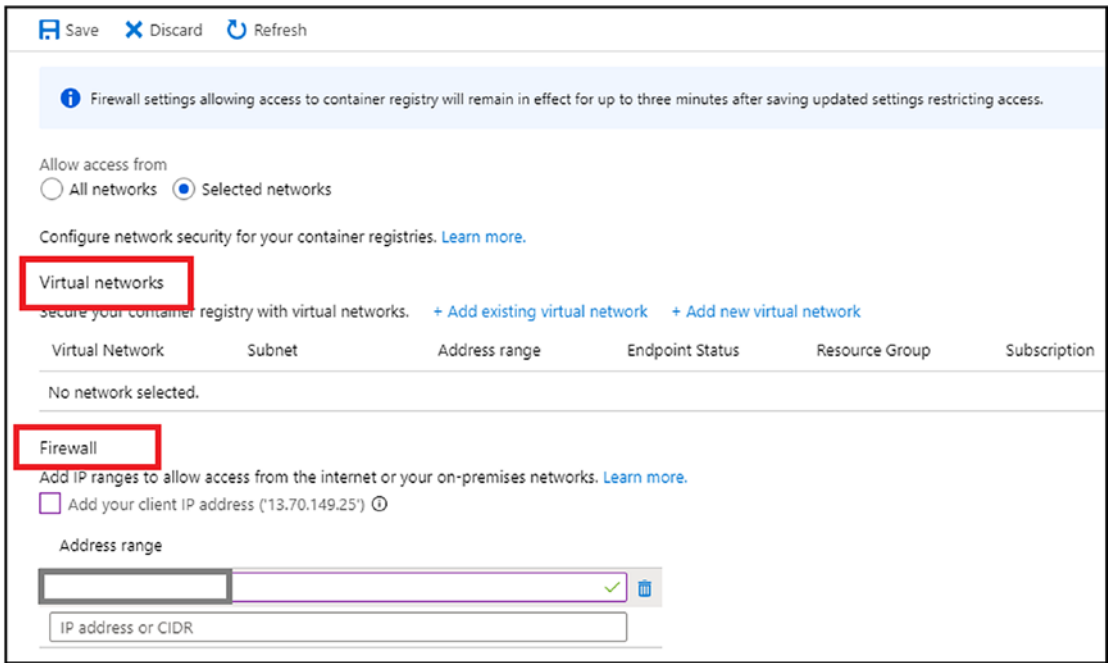


Figure 2-32. Firewall and Virtual Networks configuration page

Rotate ACR Access Keys

Another good security best practice in Azure and in general is to rotate and regenerate the access keys to the ACR registry.

Because the access keys act as a password with which to access the registry, over time the keys can swap hands and be used in the development and testing environment, where they can be mishandled.

Therefore, it is recommended you regenerate new keys every 90 days and distribute them to all stakeholders. By changing the keys, you prevent unauthorized access to the registry and reduce the risk of the keys' being mishandled.

To generate new access keys, click on “Access Keys” in the Settings menu in ACR.

In the access key's Password section, click on “Regenerate” to generate a new key and confirm the request.

Figure 2-33 shows the regenerate option available on the right side.

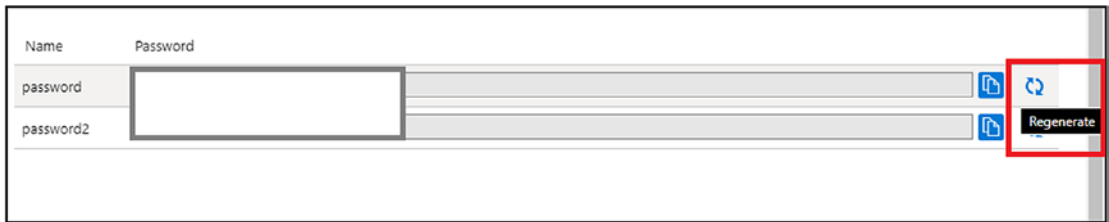


Figure 2-33. *Regenerate password option*

Confirm the request and note that the current password will stop working, so make sure you notify all the involved stakeholders beforehand. Figure 2-34 shows the confirmation screen.



Figure 2-34. *Regenerate access key*

Azure CLI

Before closing this chapter, I would like to show you how to install and use Azure CLI on your local machine.

Up until now, we have used Azure Cloud Shell, which comes pre-configured with Azure CLI and other utilities without our needing to install anything.

If you don't like Azure Cloud Shell, you can install Azure CLI on your machine and manage Azure resources using the Shell.

To get started, go to the following URL:

<https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest>

Azure CLI can be installed on the following platforms, as shown in Figure 2-35.

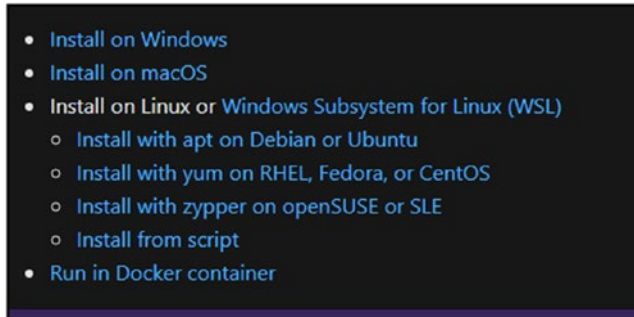


Figure 2-35. *Azure CLI platforms*

As shown in Figure 2-35, you can install Azure CLI on almost any platform. To install it on Windows 10 using PowerShell, I will use the following command:

```
Invoke-WebRequest -Uri https://aka.ms/installazurecliwindows -OutFile  
.\AzureCLI.msi; Start-Process msixexec.exe -Wait -ArgumentList '/I AzureCLI.  
msi /quiet'
```

Once installed, I can log in to Azure using the following command:

```
az login
```

To see the full list of the Azure CLI commands, type the following:

```
az help
```

The only disadvantage of installing Azure CLI on your local machine is that you will need to keep it up to date in order to use all the features in Azure.

Because Azure services are evolving daily, new Azure CLI commands are being released very often, and by using an outdated version of Azure CLI, you will not be able to utilize new commands with an outdated version of Azure CLI.

If you decide to use Azure CLI locally, make sure you check for new updates at least once a week.

One of my favorite ways to run Azure CLI is inside a Docker container using the Azure CLI Docker image. The Azure CLI image is running on Alpine Linux V3.7 with an overall disk size of 931 MB. To download the Azure CLI Docker image, we need to use the following command:

```
docker pull mcr.microsoft.com/azure-cli
```

After downloading the Docker image, we can start it by using the following command:

```
docker run -it mcr.microsoft.com/azure-cli
```

After running the image, we can use Azure CLI in the same way we use it in Azure Cloud Shell and Azure CLI locally.

To log in to Azure, use the following command:

```
Az login
```

Summary

This chapter was packed with technical hands-on changes and configurations that helped us enable and configure ACR.

In this chapter, we have learned how to enable and set up ACR using both the portal and Azure CLI.

We also pushed and pulled images from our ACR registry using the Docker CLI.

To simplify ACR management, we installed and configured VS Code with the Docker extension and connected it to our ACR.

We also explored two ways to secure the overall security status of our ACR using the Firewall and Virtual Networks feature and by changing the access keys' passwords.

In the next chapter, we will explore and deploy containers to Azure Container Instances (ACI).

CHAPTER 3

Deploy Containerized Applications with Azure Container Instances (ACI)

In this chapter, we will deep dive and explore the Microsoft Azure Container Instances (ACI) service. ACI allows us to deploy containerized applications quickly without extra configuration and provisioning of added infrastructure and is excellent for the following scenarios:

- Simple web applications
- Task containers
- Batch jobs

With ACI, we can publish containers and services with a public IP address and expose them to the public in a short time. We can quickly scale ACI containers and start them with a very low resources footprint running on both Linux and Windows containers.

The following topics will be covered in this chapter:

- Using `az container` Azure CLI commands to deploy containers
- Deploying Linux and Windows containers
- Mounting storage volumes using Azure file share
- Scaling ACI containers
- Deploying containers to web app for containers
- Monitoring and managing ACI containers

Set Up Azure Container Instances (ACI)

ACI is a service that doesn’t require a lot of setup or complex configuration before using it. In our case, I will show you first how to access ACI using the portal and then using Azure CLI.

To access ACI from the Azure portal side bar, click on “All services,” then click on “Containers” in the left pane. From the Containers pane, click on “Container instances.”

Figure 3-1 shows the container instances location in the Azure portal.

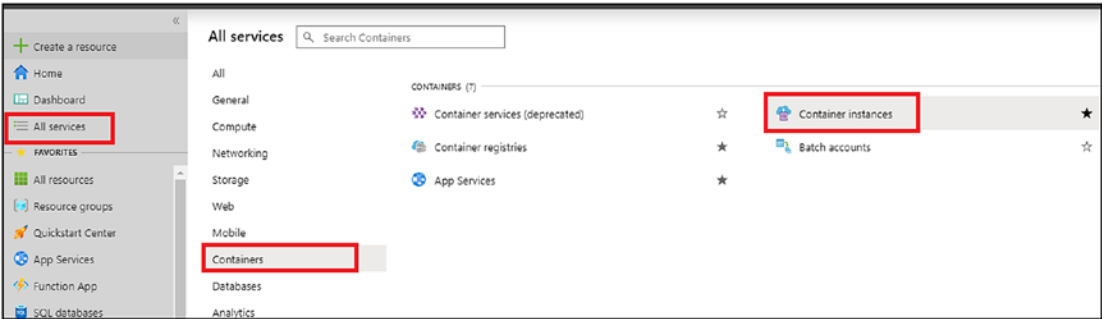


Figure 3-1. ACI in the Azure portal

To create a container instance, we click “Add” on the ACI page, as shown in Figure 3-2, and follow the wizard.

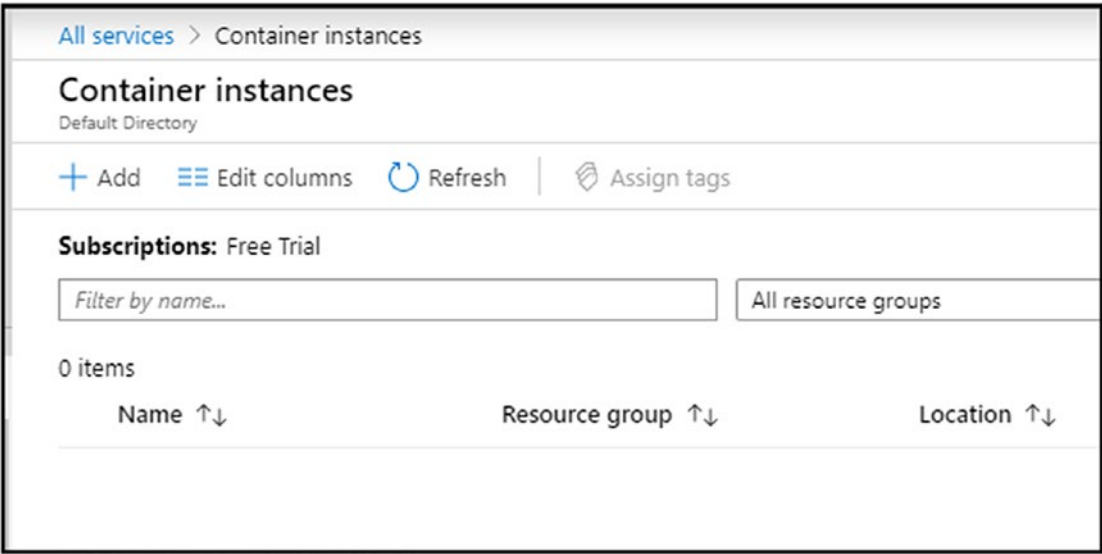


Figure 3-2. Create ACI instance

AZ Container Commands

We can also use Azure CLI via Azure Cloud Shell to create container instances using the following command syntax:

```
az container command
```

The `az container` commands are very powerful and allow us to fully manage ACI deployment from the CLI without using the portal.

The following command list outlines all the `az container` commands available:

<code>az container attach</code>	Attach to a running container
<code>az container create</code>	Create a new container or container group
<code>az container delete</code>	Delete a container or a container group
<code>az container exec</code>	Execute a command inside a running container
<code>az container export</code>	Export a container configuration to a YAML format.
<code>az container list</code>	List container or container group
<code>az container logs</code>	View container log
<code>az container restart</code>	Restart a container or a container group
<code>az container show</code>	Show container details
<code>az container start</code>	Start an ACI container or container group
<code>az container stop</code>	Stop an ACI container or container group

To view the latest list of `az container` commands, from Azure Cloud Shell run either

```
az container -h
```

or

```
az container --help
```

Container Groups

ACI also allows us to create container groups, which group multiple containers under the same deployment.

For example, if you have an application that is made up of a front-end proxy service and a backend service, you can group them together under the same ACI deployment.

Container groups are also good when your application has a single entry point using port 80 and there are other containers that are running on different points and are not accessible from the public-facing entry point.

ACI Limitations

Before you start your ACI deployment and scaling, it is important to note and understand the service limitations of ACI. ACI has a limit of running 60 containers inside a single container group.

We can only attach 20 storage volumes per container group.

An external IP address supports five ports only.

ACI has an image size limitation of 15 GB per image; therefore, it is crucial you keep your build small and efficient.

We can only create and delete 300 containers per hour and 100 per five minutes, If you are planning to use ACI for scheduling tasks and jobs, make sure you understand this limitation.

These limits are soft limits, and you can contact Azure support to increase them.

Deploy Linux and Windows Containers to ACI

ACI offers us the option to deploy both Linux and Windows containers' Docker images.

Most Linux containers are supported; however, it is recommended you use Linux Alpine because of its small footprint.

When it comes to Windows, it is important to note that ACI supports the following Windows container images:

Windows Nano Server 2016 (sac2016)

Windows Server Core 2016 (ltsc 2016)

Note Windows container images based on semi-annual channel releases are not supported and can't run on ACI.

The following Windows Server 2019 container images are also supported under preview mode:

Nano Server — 1809

Windows Server Core 2019 — 1809, ltsc2019

Deploy Linux Containers

To get started with ACI, I will show you how I deploy my Linux Nginx image from my Azure Container Registry (ACR).

This is the same image I used in Chapter 2 and pushed to my ACR. If you followed Chapter 2, you should have a stored image in ACR.

To run my first Linux container image, from the ACI home page I will click on “Add,” as shown in Figure 3-3.

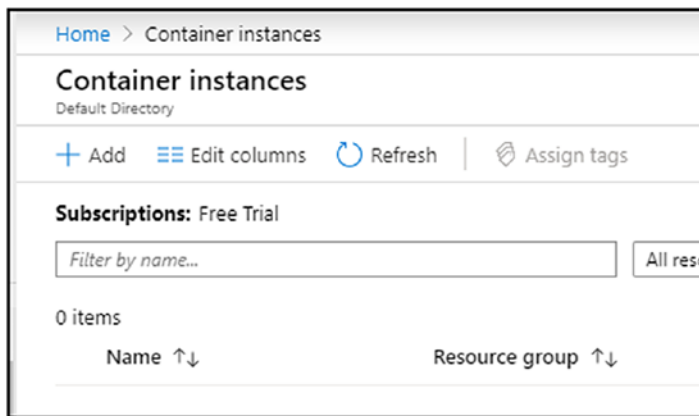


Figure 3-3. Add container instance

On the Create Container Instance page I will fill in the following details:

Resource Group — We can create a new one or select existing group.

Container Name — This is the name of the deployed container that will appear in the portal.

Region — This is the Azure region the container will run in.

Image type — In my case, I am selecting “Private” and using the login details of my ACR.

OS type — This is where we select Linux or Windows containers (default is Linux).

Fill in the details, and when you are ready click on “Create.” Figure 3-4 shows the Create Container Instance setup page.

Create container instance

[Basics](#) [Networking](#) [Advanced](#) [Tags](#) [Review + create](#)

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 * ⓘ

Resource group * ⓘ [Create new](#)

Container details

Container name * ⓘ ✓

Region * ⓘ

Image type * ⓘ ☐ Public ☒ Private

Image name * ⓘ ✓

Image registry login server * ⓘ ✓

Image registry user name * ⓘ ✓

Image registry password * ⓘ ✓

OS type * ☒ Linux ☐ Windows

[Review + create](#) [< Previous](#) [Next : Networking >](#)

Figure 3-4. *Create Container Instance page*

On the Networking screen, we have the following options:

Include a public IP address to make the container accessible from the internet.

Open specific ports (remember that we can only use five ports as per ACI limits).

Public DNS — We can also create a public DNS record.

Note We cannot add public DNS records after creating the container.

Figure 3-5 shows the Networking configuration page.

Basics

Networking

Advanced

Tags

Review + create

You can configure networking settings for your container, such as ports and protocols as well as a DNS name label. If you choose not to include a public IP address, you will still be able to access your container and logs using the command line.

[Learn more about Azure Container Instances networking](#)

Include public IP address

☒ Yes ☐ No

Ports ⓘ

Ports	Ports protocol
80	TCP

DNS name label ⓘ

☒

.westus.azurecontainer.io

Figure 3-5. Networking configuration page

Next, we have the option to add advanced features like the following:

- Restart policy — We can decide when to restart the container; the current options are on failure, always, never.
- Environment variables — Here, we can set variables for our application, like debugging, configuration files, etc.
- Command override — This is where we specify commands that will run when the container starts (same as CMD in Docker).

Figure 3-6 shows the Advanced configuration page.

Create container instance

Basics Networking **Advanced** Tags Review + create

Configure additional container properties and variables.

Restart policy ⓘ On failure ▼

Environment variables

Key	Value

Command override ⓘ [] ✓

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

Figure 3-6. Advanced configuration page

In the Tags configuration page, we can add custom tags for billing and management purposes, as shown in Figure 3-7.

Basics Networking Advanced **Tags** Review + create

Tags are name/value pairs that enable you to categorize resources and view consolidated billing by applying the same tag to multiple resources and resource groups. [Learn more about tags](#)

Note that if you create tags and then change resource settings on other tabs, your tags will be automatically updated.

Name ⓘ	Value ⓘ
containername ▼	nginx ▼
▼	▼

Figure 3-7. Tags configuration page

And, finally, we have the Review and Create screen, where we can review the configuration and see if it is valid.

When you are ready to deploy your container, click on “Create.”

Figure 3-8 shows the Review and Create page.

Create container instance

✓ Validation passed

Basics

Networking

Advanced

Tags

Review + create

Basics

Subscription

Free Trial

Resource group

Apps

Region

(US) West US

Container name

nginxwebapp

Image type

Public

Image name

deployrg.azurecr.io/nginx:latest

OS type

Linux

Memory (GiB)

1.5

Number of CPU cores

1

GPU type

None

Number of GPU cores

0

Networking

Include public IP address

Yes

Ports

80 (TCP)

DNS name label

nginxwebapp

Advanced

Restart policy

On failure

Command override

[]

Tags

Figure 3-8. Review and Create page

ACI will deploy the container. The deployment time depends on the image size and the ACR location the image is being pulled from. In my case, the deployment took less than two minutes.

Figure 3-9 shows the deployed container available from the DNS hostname I chose in the Networking configuration page.

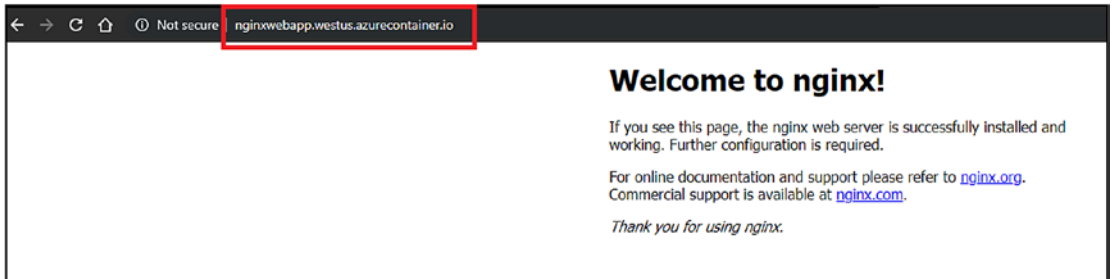


Figure 3-9. Deployed Nginx container

We can easily manage the container from the ACI Container Overview page, as shown in Figure 3-10.

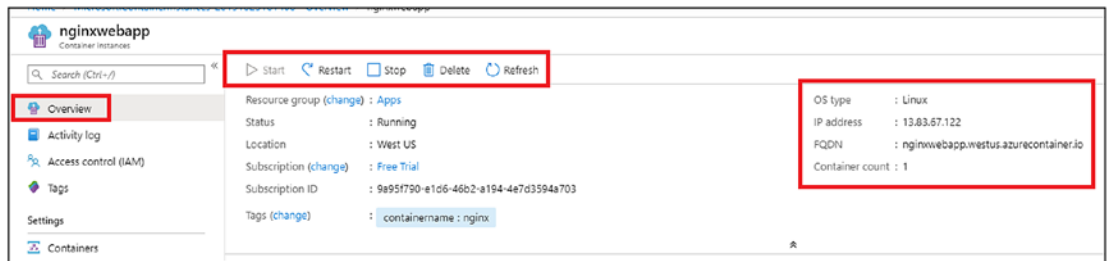


Figure 3-10. Deployed Container Overview page

Deploy Containers Using Azure CLI

I can also deploy the same container image using Azure CLI directly from Azure Cloud Shell by using the following code:

```
az container create -g apps --name nginxwebapp02 --location westus --dns-
name-label nginxwebappv --registry-username deployrg --registry-login-
server deployrg.azurecr.io --image deployrg.azurecr.io/nginx:latest
--registry-password typeaccesskeypassword
```

Using the `az container` command, I can deploy the same container I used in the GUI shown in the Deploy Linux Containers section, but this time the deployment is quicker.

The Azure CLI is excellent if you need to programmatically create containers quickly with zero chances of error. In the GUI, it is easier to miss some of the configurations when the task becomes repetitive.

The result can be seen in Figure 3-11, which is the same application with a different DNS name.

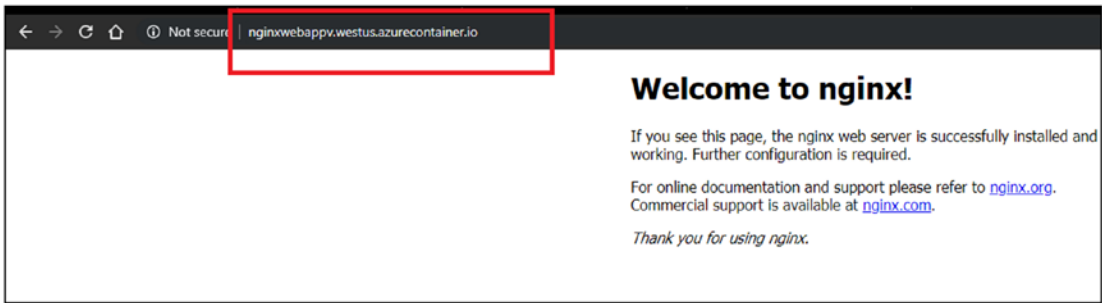


Figure 3-11. Azure CLI deployed container

To delete the container, I can simply use the following command:

```
az container delete --name nginxwebapp02 --resource-group apps
```

Once you run the command, it will take between one and three minutes for the container to be deleted.

Deploy Windows Containers

Up until now, we have deployed Linux containers, and now it is time to show you how to deploy Windows containers. Deploying Windows containers is almost the same as deploying Linux containers.

If we are using the portal, we need to select “Windows” under the “OS type” option and use a Windows-based Docker image. Figure 3-12 shows the OS type in the Azure portal ACI wizard.

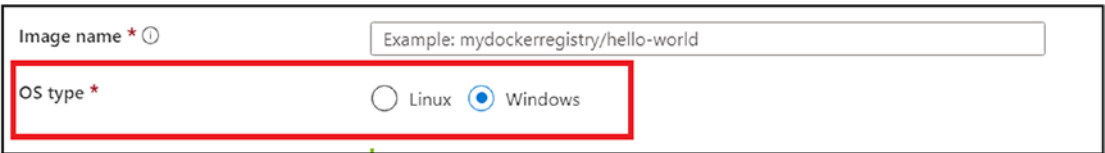


Figure 3-12. OS type

If we are using Azure CLI, we need to specify the OS type value; this is different than Linux containers, for which we don't need to specify this because it is the default option.

The following code shows an Azure CLI deployment of a Windows container. In the code you can see that I am specifying Windows in the `--os-type` value and using a public Windows IIS server image:

```
az container create --os-type windows -g apps --name iiswebapp --location westus --dns-name-label iiswebappv --image mcr.microsoft.com/windows/servercore/iis:windowsservercore-ltsc2019
```

After deploying the code using Azure CLI, head to the portal, copy the DNS name from the Overview section, and paste it into your browser.

Figure 3-13 shows the IIS homepage of the deployed container.

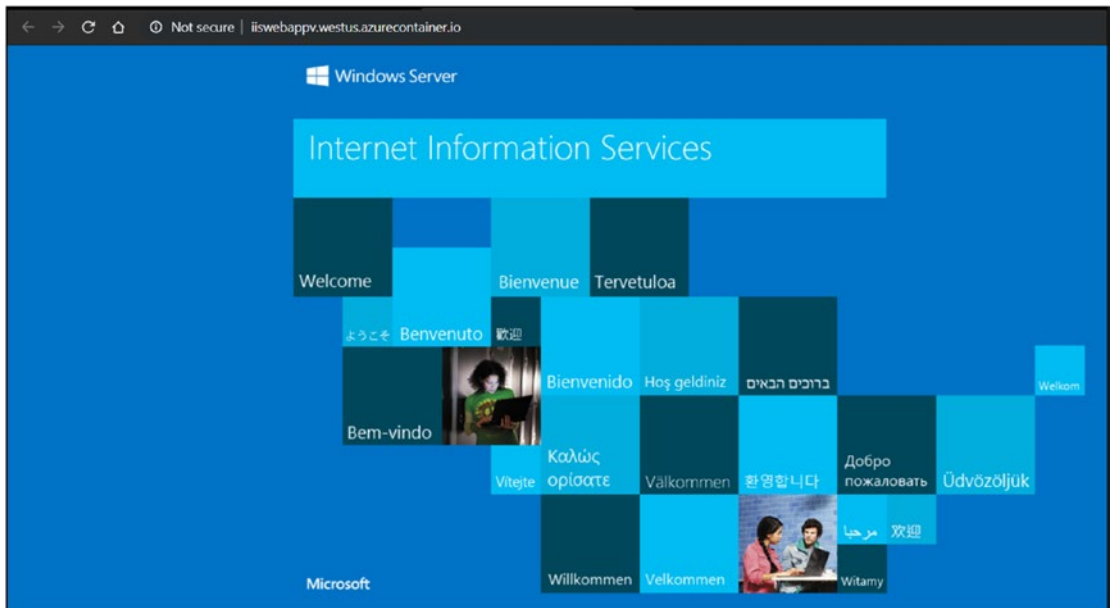


Figure 3-13. IIS homepage

Note Please make sure you use the following supported Windows container images on ACI, and remember that semi-annual channel releases are not supported and cannot run on ACI:

Windows Nano Server 2016 (sac2016)

Windows Server Core 2016 (ltsc 2016)

Nano Server — 1809 (Preview)

Windows Server Core 2019 — 1809, ltsc2019 (Preview)

Mount Storage Volume to an ACI Container

If you have worked with Docker containers, you already know that saving application data inside a container is not a good idea.

The reasons we don't save data inside our containers are as follows:

Data saved inside a container is lost when we delete the container.

The data that we store in the container is stored on the container host, which can lead to data loss if the host malfunctions.

Because the Docker image is using a software driver to manage storage to read in, write operations are slower. For that reason, it is recommended to use storage volumes that do not depend on the container host or the container.

Volumes are mounted to the containers using SMB protocol, and all data stored on the volumes are persistent.

If the container fails, is deleted, or is stopped, the data is not lost, and when the container restarts or is recreated, it picks up the configuration from the volume and continues from where it stopped.

Volumes can manage scaling quickly because multiple containers can access the same volume and retrieve the configuration. In our case, ACI allows us to use storage volumes and keep our configuration data persistent.

Next, we will mount a storage volume to our Nginx container. ACI uses Azure file share as the volume, and we will use Azure CLI using Azure Cloud Shell to configure our volume.

Mounting Azure File Share Volume to an ACI Container

In this example, we will use three Bash Shell scripts to mount our volume. The scripts that we will use are as follows:

1. `Create_Stroage.sh` — This script creates a storage account and an Azure file share for the volume.
2. `Get_Storage_Key.sh` — This script retrieves the storage key, which is like a password that allows the container to access the volume.
3. `Create_Container.sh` — This script creates a new container using an image stored on ACR and mounts the storage volume.

To get the scripts running, I will use Azure Cloud Shell and upload them using the “Upload” button. Please review Chapter 1 on how to use the Azure Cloud Shell upload option.

To run the bash scripts, we will use the following command:

```
. 1.Create_Strogae.sh
```

Note There is a space after the dot, and Bash is case sensitive.

To get started, I will run the first script, which will create an Azure storage account in the WestUS region. Please change the variables’ details to fit your environment.

1. `Create.storage.sh`

```
#!/bin/bash
Resource_Group=apps
Storage_Account=storageaccount$RANDOM
Location=westus
Share_Name=acifileshare
```

```
# Create the storage account
az storage account create \
    --resource-group $Resource_Group \
    --name $Storage_Account \
    --location $Location \
    --sku Standard_LRS

# Create the Azure file share
az storage share create --name $Share_Name --account-name
$Storage_Account --quota 5
```

Note To change the file share size, change the number next to quota (5 GB by default).

Once the storage account has been created, I run script number two, which will retrieve the storage account details and the storage access key needed to map the volume.

2. Get_Storage_Key.sh

```
#!/bin/bash
echo $Storage_Account

STORAGE_KEY=$(az storage account keys list --resource-group
$Resource_Group --account-name $Storage_Account --query "[0].
value" --output tsv)
```

Note If you are planning to redeploy this code to recreate the container in the future, please copy the output of script number two, which will output the storage account name and the storage account access key.

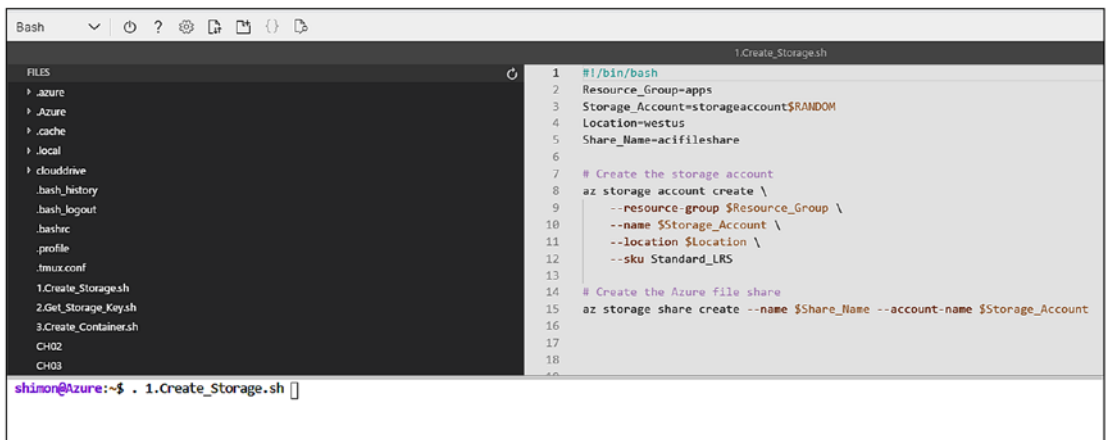
Once you have them, you can run script number three in the future and fill in the storage account and access key details without using the variables by changing the variables of the \$Storage_Account \$STORAGE_KEY:

```
echo $STORAGE_KEY
```


The third and last script will deploy a container using my Nginx image that is stored in ACR:

```
#!/bin/bash
az container create \
  --resource-group $Resource_Group \
  --name nginxvapp \
  --registry-username deployrg \
  --registry-login-server deployrg.azurecr.io \
  --image deployrg.azurecr.io/nginx:latest \
  --registry-password typeacraccesskeypassword \
  --dns-name-label nginxvol \
  --ports 80 \
  --azure-file-volume-account-name $Storage_Account \
  --azure-file-volume-account-key $STORAGE_KEY \
  --azure-file-volume-share-name $Share_Name \
  --azure-file-volume-mount-path /aci/
```

In Figure 3-14, you can see how I run the shell scripts from Azure Cloud Shell.



The screenshot shows the Azure Cloud Shell environment. On the left is a file explorer with a tree view containing folders like .azure, .Azure, .cache, .local, and files like .bash_history, .bash_logout, .bashrc, .profile, .tmux.conf, 1.Create_Storage.sh, 2.Get_Storage_Key.sh, 3.Create_Container.sh, CH02, and CH03. The main terminal window displays the contents of the script 1.Create_Storage.sh, which is a Bash script to create an Azure storage account and a file share. The script includes comments and uses the az CLI to create a storage account and a file share. The terminal prompt shows the user shimon@Azure:~\$ and the command 1.Create_Storage.sh being executed.

```
1 #!/bin/bash
2 Resource_Group=apps
3 Storage_Account=storageaccount$RANDOM
4 Location=westus
5 Share_Name=acifileshare
6
7 # Create the storage account
8 az storage account create \
9   --resource-group $Resource_Group \
10  --name $Storage_Account \
11  --location $Location \
12  --sku Standard_LRS
13
14 # Create the Azure file share
15 az storage share create --name $Share_Name --account-name $Storage_Account
```

Figure 3-14. Run Bash script in Azure Cloud Shell

When the container has deployed, I will connect to it using the following Azure CLI command directly from Azure Cloud Shell. As you can see in Figure 3-15, the ACI directory is visible to the container’s file system and the ACI folder is located on the Azure file share.



```
shimon@Azure:~$ az container exec -g apps --name nginxvapp --container-name nginxvapp --exec-command "/bin/bash"
root@wk-caas-fbe30925910c40268e1c05646c78576b-f8cc204a2796264ec297fc:/# ls
aci boot etc lib media opt root sbin sys usr
bin dev home lib64 mnt proc run srv tmp var
root@wk-caas-fbe30925910c40268e1c05646c78576b-f8cc204a2796264ec297fc:/#
```

Figure 3-15. ACI folder located on Azure file share

Any file created inside the ACI folder will be available after I delete this container and create a new one. If I delete this container and run script number three again there will be zero data loss and the data will be available again to a new container.

You must understand this process very well.

I will go ahead and create two new files called test and test 2 using the touch command:

```
touch test
touch test2
```

If I access the Azure file share from the Azure portal, these two files will appear inside the file share. Figure 3-16 shows the two files I created from inside the container, and they are available to me.

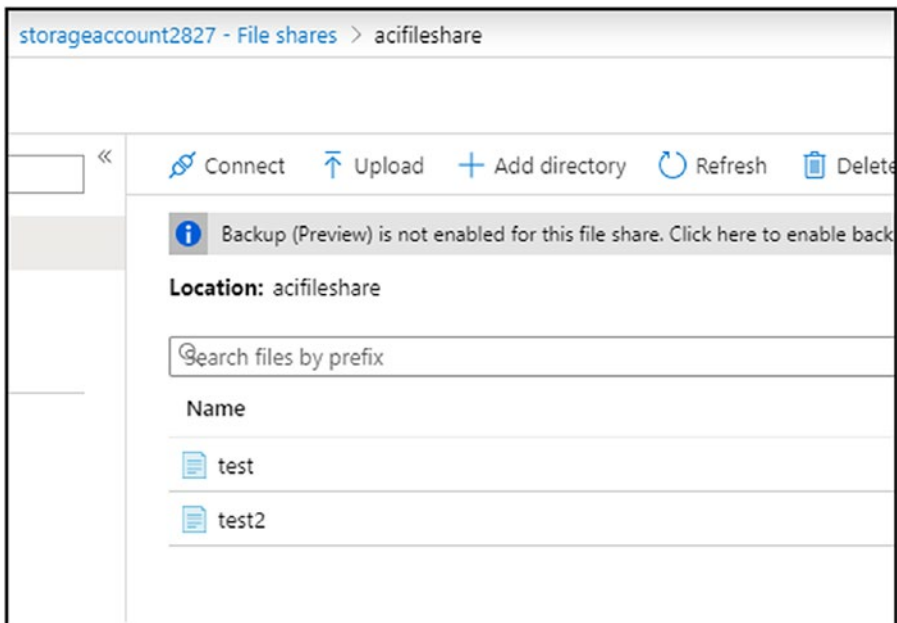


Figure 3-16. Azure file share contents

I will go ahead and delete the container and create a new one and test that the files are available for the new container.

To view my containers, I will run the following command:

```
az container list -o=table
```

To delete a container, I will run the following command:

```
az container delete --name nginxvapp --resource-group apps
```

I will run script number three again to create a new container and attach it to the Azure file share volume. As you can see in Figure 3-17, the two files are available to the new container. You will also see that the container ID is different than the ID seen in Figure 3-15.

```
shimon@Azure:~$ az container exec -g apps --name nginxvapp --container-name nginxvapp --exec-command "/bin/bash"
root@wk-caas-a2c1ecb770ac48f18a41532726c22bf8-38198eb2b2d07e55f79574:/# cd aci
root@wk-caas-a2c1ecb770ac48f18a41532726c22bf8-38198eb2b2d07e55f79574:/aci# ls
test test2
root@wk-caas-a2c1ecb770ac48f18a41532726c22bf8-38198eb2b2d07e55f79574:/aci#
```

Figure 3-17. New container with the same files

I hope that you managed to understand this process because it is important to know how to work with storage volumes for persistent data.

Note You cannot mount Azure file share to Windows containers in ACI.

Scale Containerized Applications in ACI

In this section, we will cover the process of scaling our containerized application using the Microsoft Azure portal and then using Azure CLI using Azure Cloud Shell.

The process of scaling containers in ACI is a bit different compared to other container services in Azure. ACI doesn't have a built-in process for scaling, so to scale containers in ACI we need to delete them and recreate them with more resources. This process is not ideal if you are planning to deploy enterprise solutions on ACI.

ACI is a service that was designed to deploy quickly and in a straightforward manner, almost like an entry-level service. For that reason, it is recommended we script and use Azure CLI for the scaling operations.

The default container in ACI comes with one core CPU and 1.5 GB of RAM. You can find the size of your container or container group in the ACI portal.

From the portal, open ACI, select your deployment, and click on the Containers icon on the left pane. From the bottom menu, click on the Properties tab, and you will see the size of your container.

Figure 3-18 shows the properties of my deployment.

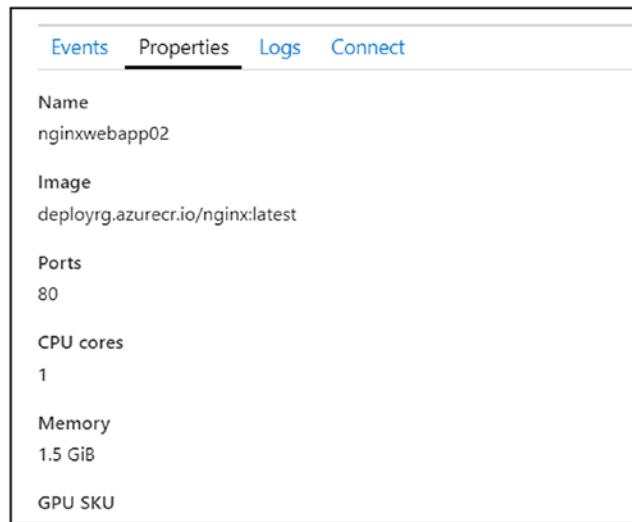


Figure 3-18. *Container group properties*

By using Azure CLI, we can control the size of our containers and not use the default values, which are used if we don't specify values.

The following two values in Azure CLI, in the `az container` command line, control the size of the container:

```
--memory
--cpu
```

In the following example, I will list my running containers, delete one from the list, and redeploy it.

First, I will run the following command to view my containers and note the name of the container I am going to scale:

```
az container list -o=table
```

In my case, I am going to scale the `nginxvapp` container and therefore will need to delete it before scaling it using the following command:

```
az container delete --name nginxvapp --resource-group apps
```

Once the container has been removed, I will run the following script, which is script number three.

Note This time I am using hard values because the variables we used in script numbers one and two are no longer in memory; please replace the values with your account details.

Look at lines 14 and 15, where I am specifying the number of CPU cores and amount of RAM:

```
#!/bin/bash
Storage_Account="storageaccountnamevalue"
STORAGE_KEY="storagekeyvalue"
Resource_Group="apps"
Share_Name="acifileshare"
az container create \
  --resource-group $Resource_Group \
  --name nginxvapp \
  --registry-username deployrg \
  --registry-login-server deployrg.azurecr.io \
  --image deployrg.azurecr.io/nginx:latest \
  --registry-password acrregistryaccesskeypassword \
  --dns-name-label nginxvol \
  --cpu 2 \
  --memory 3.5 \
  --ports 80 \
  --azure-file-volume-account-name $Storage_Account \
  --azure-file-volume-account-key $STORAGE_KEY \
  --azure-file-volume-share-name $Share_Name \
  --azure-file-volume-mount-path /aci/
```

To view the container and the specification, I will run the following command:

```
az container show --name nginxvapp --resource-group apps -o=table
```

Figure 3-19 shows the new size of the container.

Note The CPU option must come before the memory option; otherwise, the command will fail.

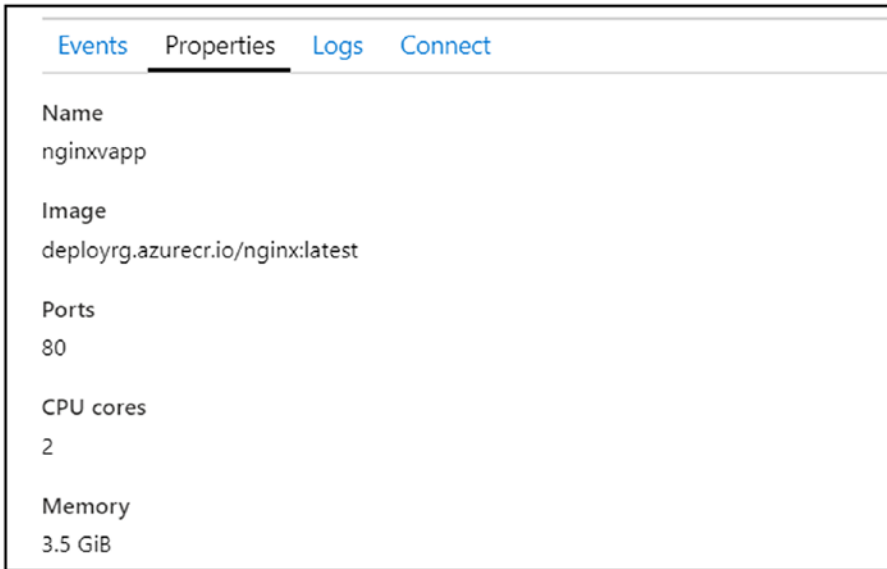


Figure 3-19. *New container size after scaling*

If you prefer to use the portal to scale your containers, simply delete the container from the ACI page and create a new one.

If you are scaling a container that is running an ACR image, perform the following steps.

Open ACR, select the image, and use the action menu, as shown in Figure 3-20.



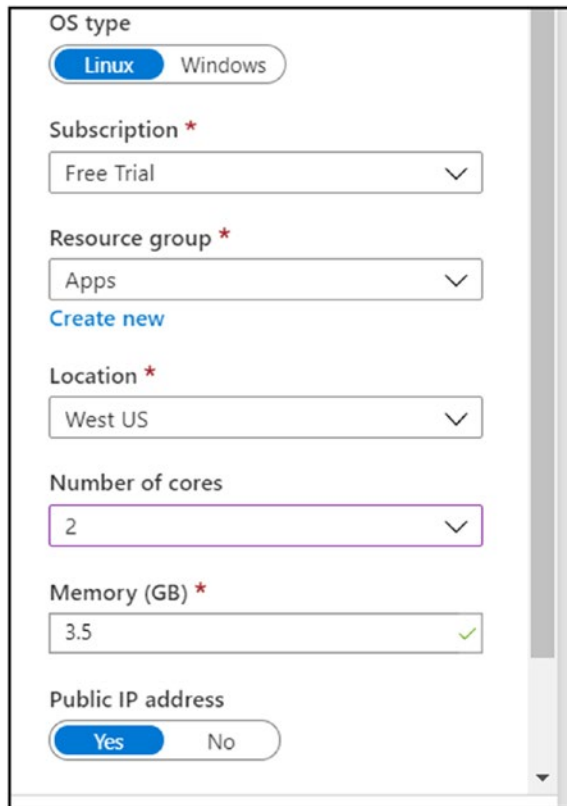
Figure 3-20. ACR image action menu

From the action menu, select “Run instance,” then modify the following values:

Number of cores

Memory (GB)

Figure 3-21 shows the CPU and memory options.



OS type

☒ Linux ☐ Windows

Subscription *

Free Trial

Resource group *

Apps

[Create new](#)

Location *

West US

Number of cores

2

Memory (GB) *

3.5

Public IP address

☒ Yes ☐ No

Figure 3-21. *Memory and CPU options*

Deploy to Web App for Containers

Another deployment option that we have, and which in some cases can act as a scaling option, is to deploy the image to a web app.

Azure Web Apps for Containers allows deploying container images stored in ACR to the Azure App Service. Azure App Service is a shared hosting service that allows us to run web applications in a shared or dedicated environment.

With Azure App Service, all the underlying infrastructure is managed by Azure, except our application. We can deploy any image to Azure App Service using Deploy to Web App and following the wizard, as shown in Figure 3-22.

Web App for Containers

Create

Site name *

Enter the name

Subscription *

Free Trial

Resource group *

Apps

Create new

*App service plan/Location

>

Image

nginx:latest

Operating system *

Linux

Windows

i

This enables continuous deployment by creating a webhook.

Create

Figure 3-22. Web App for Containers

The most crucial part of Azure App Service are the App Service Plans.

Azure App Service Plans are predefined groups of resources that are made up of CPU, memory, storage, and networking. Each App Service Plan has a price tier based on the resources included. The more resources, the more costly the plan.

As you can see in Figure 3-23, Azure offers a few pricing plans.

The screenshot shows the 'Spec Picker' window for Azure App Service. It has three tabs: 'Dev / Test' (selected), 'Production', and 'Isolated'. Below the tabs is a blue banner stating: 'The first Basic (B1) core for Linux is free for the first 30 days!'. Under the 'Recommended pricing tiers' section, three plans are listed in purple boxes:

Plan	ACU	Memory	Compute Equivalent	Price (AUD/Month)
P1V2	210 total	3.5 GB	Dv2-Series	111.25 (Estimated)
P2V2	420 total	7 GB	Dv2-Series	221.51 (Estimated)
P3V2	840 total	14 GB	Dv2-Series	443.01 (Estimated)

Below the plans is a link: 'See additional options'. At the bottom, there are two sections: 'Included features' and 'Included hardware'. 'Included features' lists 'Custom domains / SSL' with a sub-point 'Configure and purchase custom domains with SNI and IP SSL'. 'Included hardware' lists 'Azure Compute Units (ACU)' with a sub-point 'Dedicated compute resources used to run applications deployed'.

Figure 3-23. App Service Plans

The big difference between App Service and ACI is that web apps can be scaled up or down without needing to delete the deployment.

Monitor and Manage Containerized Applications on ACI

In our last section of this chapter, we will learn how to monitor and manage our containers in ACI by using the Azure portal and also using Azure CLI using Azure Cloud Shell.

Monitoring ACI with Azure Monitor

To monitor our containers in ACI, we will use the Azure Monitor service, which plugs directly into the ACI console. Microsoft Azure Monitor offers great monitoring tools and insights for free with any ACI deployment, directly from the ACI portal.

To get started, I will show you how to access the monitoring tools for my ACI deployment from the ACI console. To view the performance of my ACI deployment, I will click on my container group and click on “Metrics (Preview)” in the left-hand pane, as shown in Figure 3-24.

Note Azure Monitor for ACI is in preview mode, and some features might be missing until the service goes GA.

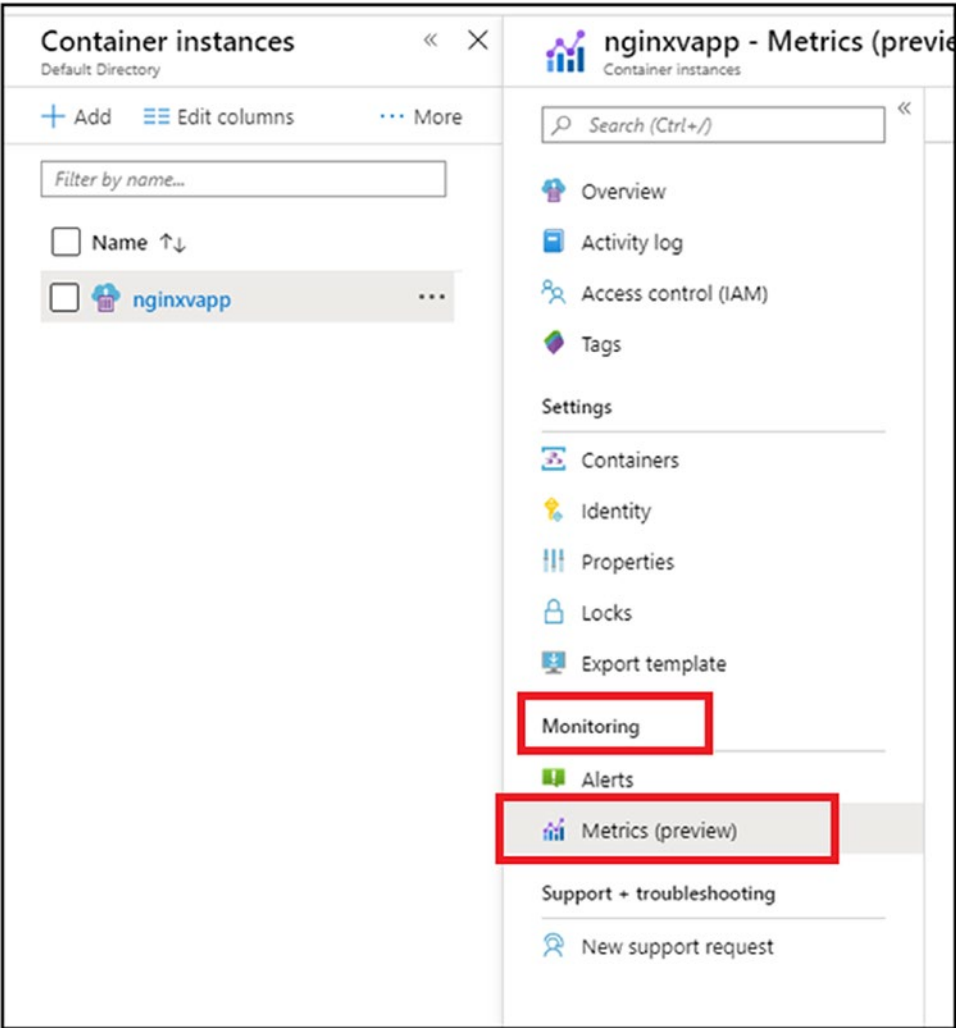


Figure 3-24. Metrics (Preview)

Using the drop-down menus on the Metrics page, I will select my container group and metric I would like to monitor.

Note Azure Monitor offers monitoring for three metrics:

CPU usage — Measured in millicores

Network — Outbound and inbound network traffic transmitted per second

RAM memory usage — Usage of RAM based on average bytes

Figure 3-25 shows the metrics page drop-down options.

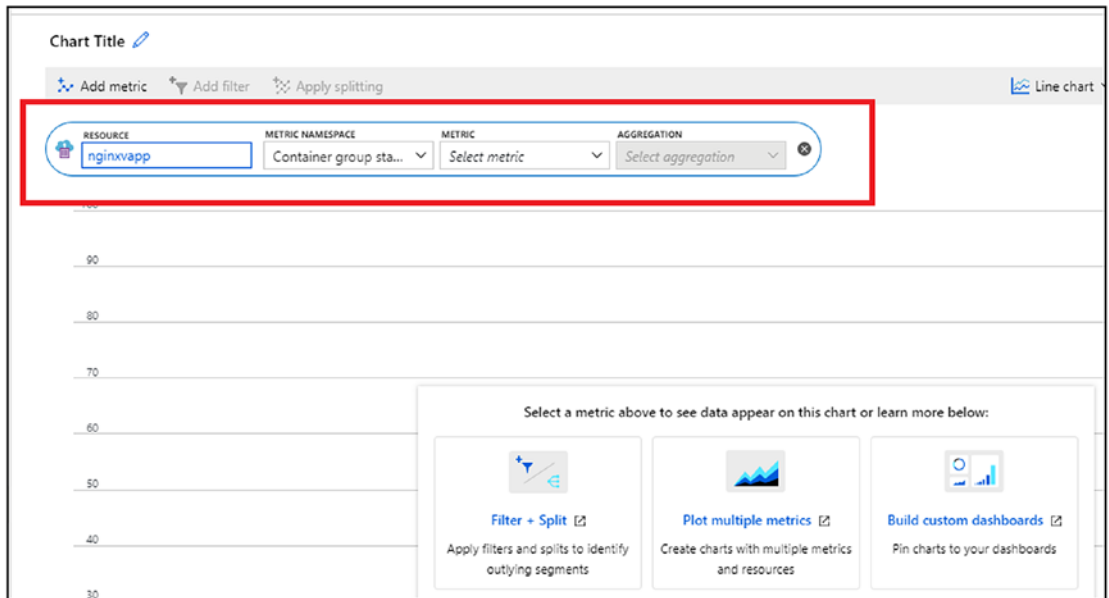


Figure 3-25. Metrics drop-down menus

In my case, I have selected the memory metric, which is based on average usage, and as shown in Figure 3-26 I can monitor my memory usage.

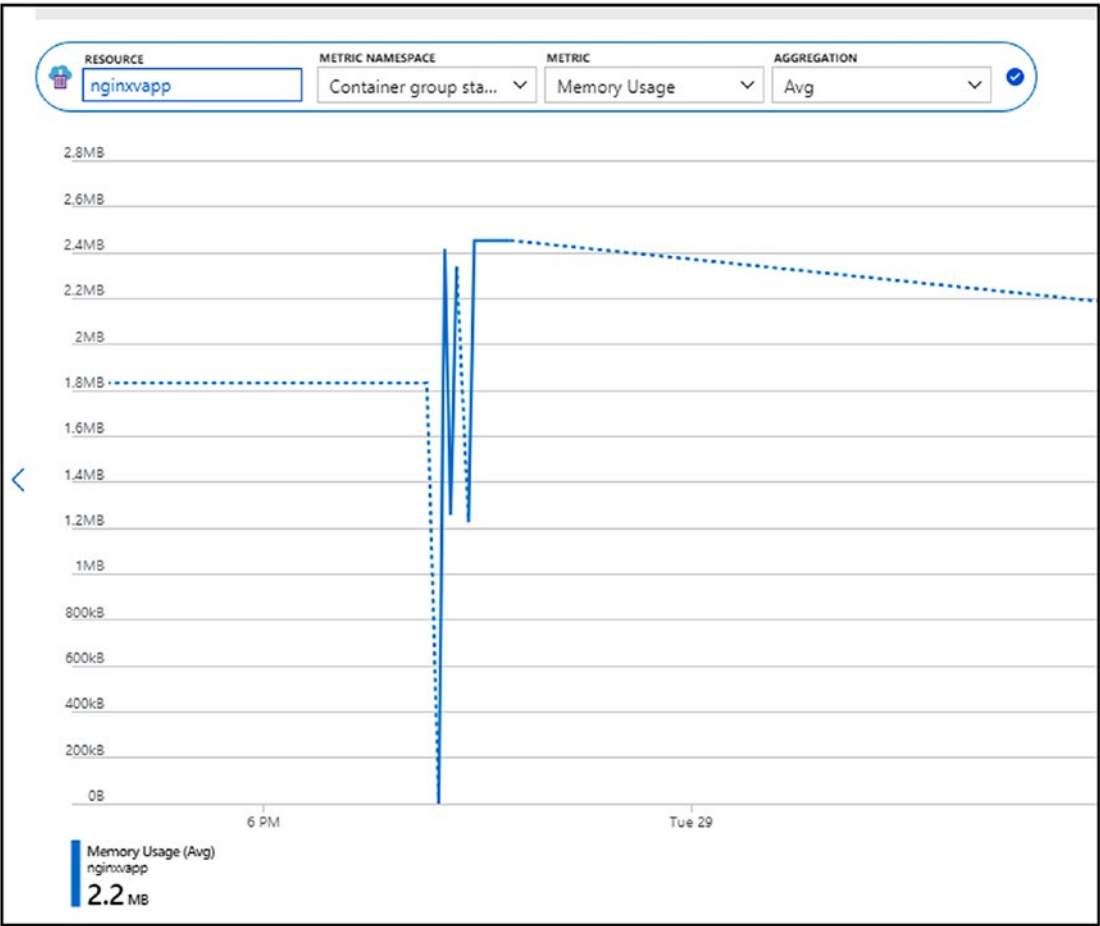


Figure 3-26. RAM usage based on average

I can also use Azure Monitor via the Azure CLI command line using Azure Cloud Shell to monitor my containers using the following commands:

```
CONTAINER_GROUP=$(az container show --resource-group apps --name nginxvapp
--query id --output tsv)
az monitor metrics list --resource $CONTAINER_GROUP --metric MemoryUsage
--output table
```

The preceding commands will produce the same output as that shown in [Figure 3-26](#).

To monitor CPU usage, I will use the following command:

```
az monitor metrics list --resource $CONTAINER_GROUP --metric CPUUsage
--output table
```

To change the metrics display format, I can use the “Area chart” drop-down menu and select the format from the list.

Azure Monitor offers the following display options:

- Line chart
- Area chart
- Bar chart
- Scatter chart
- Grid

Figure 3-27 shows the area chart options.

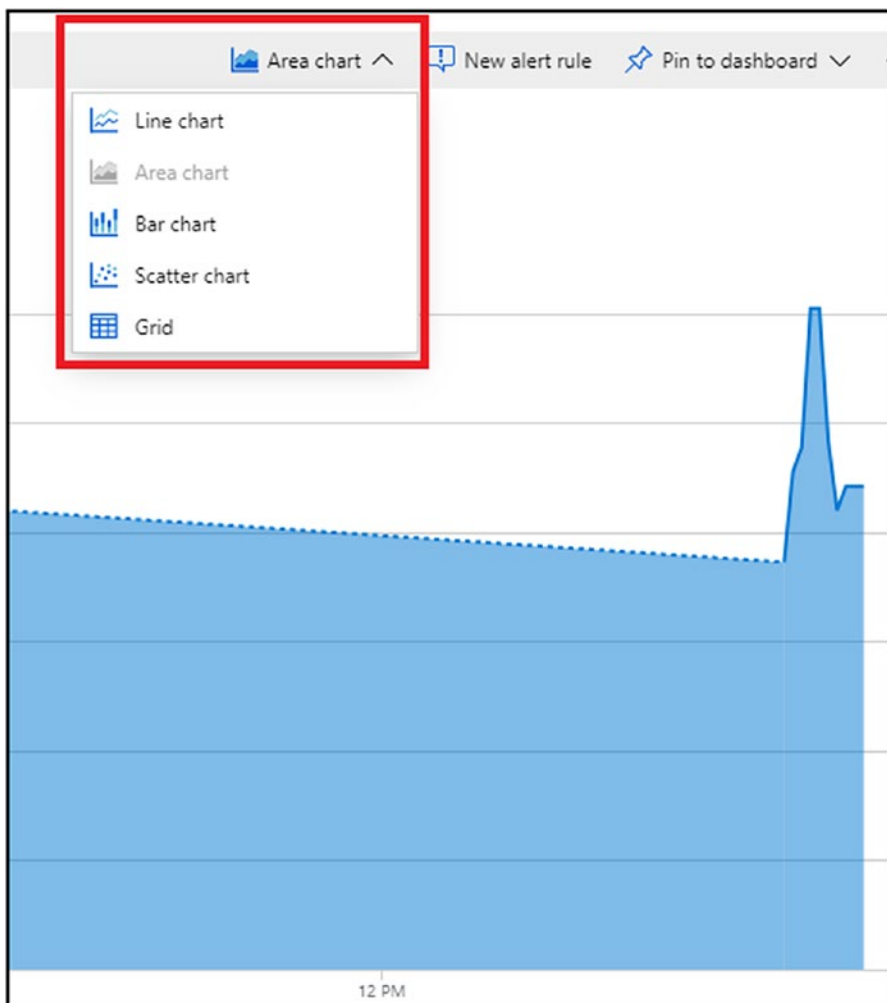


Figure 3-27. Area chart options

I can also export the results to Microsoft Excel using the “Share” option. Figure 3-28 shows the “Download to Excel” option.

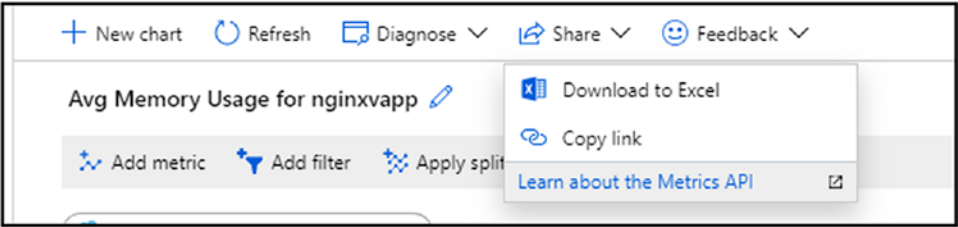


Figure 3-28. “Share” options

The exported data is shown in Figure 3-29 in Microsoft Excel.

	A	B	C
1	Chart Title:	Avg Memory Usage for nginxvapp	
2			
3	Mon Oct 28 2019 15:50:36 GMT+1100 (Australian Eastern Daylight Time)		
4	Tue Oct 29 2019 15:50:36 GMT+1100 (Australian Eastern Daylight Time)		
5	Grain: Automatic		
6	Aggregation type: Avg		
7	Grouping: None		
8	Filters: None		
9	Chart type: Line		
10			
11		nginxvapp, Memory Usage (Avg)	
12	28/10/2019 20:15	1832732.444	
13	28/10/2019 20:25	0	
14	28/10/2019 20:30	2416640	
15	28/10/2019 20:35	1257687.579	
16	28/10/2019 20:40	2338816	
17	28/10/2019 20:50	1226752	
18	28/10/2019 20:55	2453504	
19	28/10/2019 21:00	2453504	
20	28/10/2019 21:05	2453504	
21	28/10/2019 21:10	2453504	
22	28/10/2019 21:15	2453504	
23	28/10/2019 21:20	2453504	
24	28/10/2019 21:25	2453504	
25	29/10/2019 15:45	1864135.111	
26			

Figure 3-29. Exported data shown in Microsoft Excel

If you click on the Overview page of your container group, you will see that Azure Monitor displays by default matrices for CPU, memory, and network.

Figure 3-30 shows the Overview monitoring metrics page.

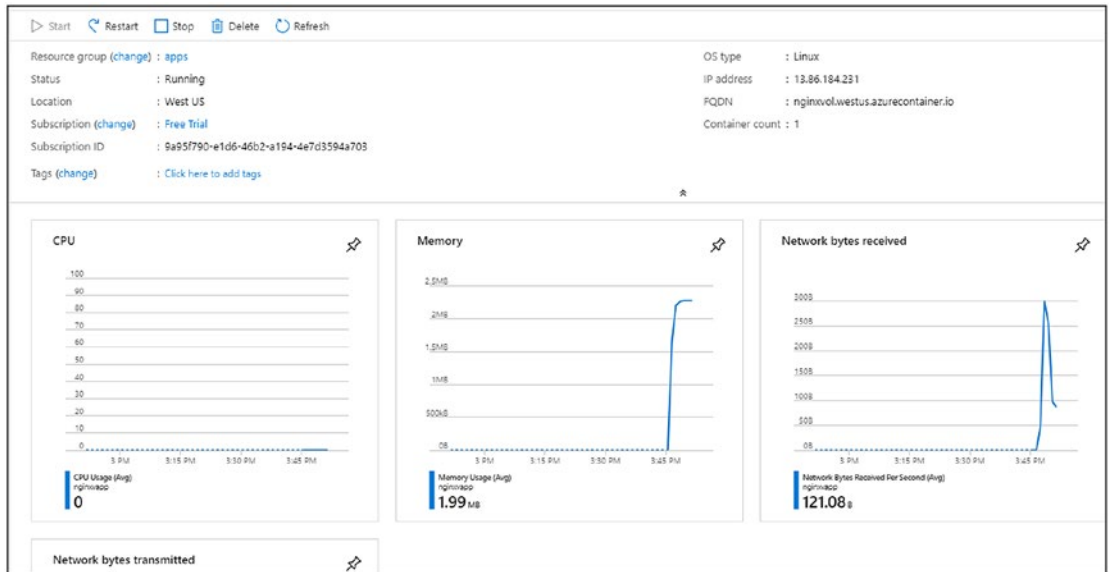


Figure 3-30. Overview page of Azure Monitor

Managing ACI Deployments

Out of the box and by default, Microsoft Azure gives us four management tools that allow us to manage our ACI deployments directly from the ACI console without using either Azure CLI or Azure Cloud Shell.

The following management options are available:

Event viewer — View every event directly from the ACI console.

Deployment properties — View deployment details like CPU, memory, etc.

Log — Container logs are shown directly from the console.

Shell (connect) — Access the container's Bash Shell directly from the Azure ACI console.

To access the ACI management options, on the ACI container group page click on “Containers” under Settings, located in the left-hand pane (as shown in Figure 3-31).

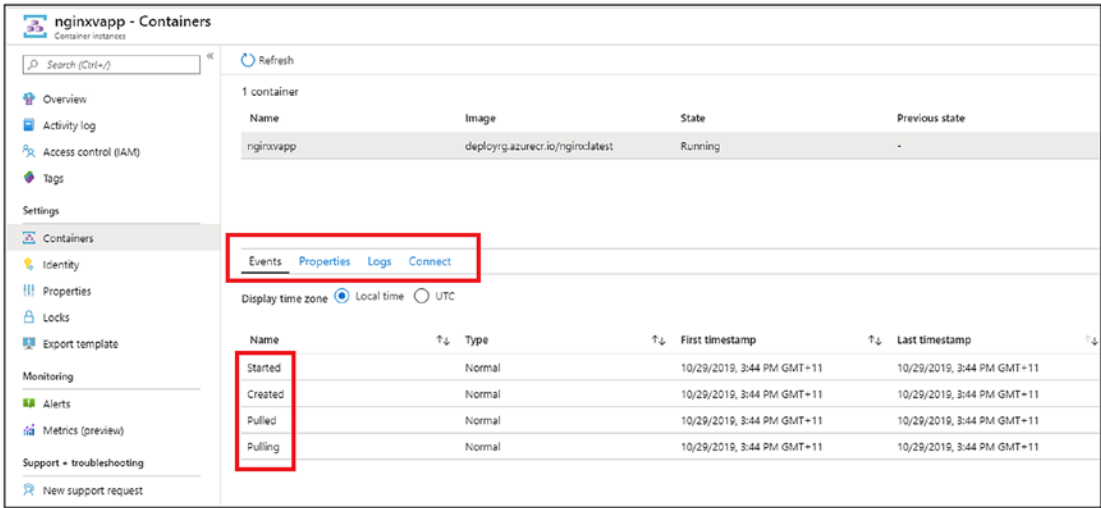


Figure 3-31. ACI management options

The “Events” option shows every event in the deployment process. In Figure 3-32, you can see the events that ACI exposes. Each event shows the event type, time started, time ended, and detailed message. These details can help us measure performance and troubleshoot issues.

Events

Properties

Logs

Connect

Display time zone ☒ Local time ☐ UTC

Name	↑↓	Type
Pulled		Normal
Created		Normal
Started		Normal
Pulling		Normal
Started		Normal
Created		Normal
Pulled		Normal
Pulling		Normal

Figure 3-32. *Container group events*

As you can see in Figure 3-33, all the deployment details are available. You can view and use the properties details for support purposes.

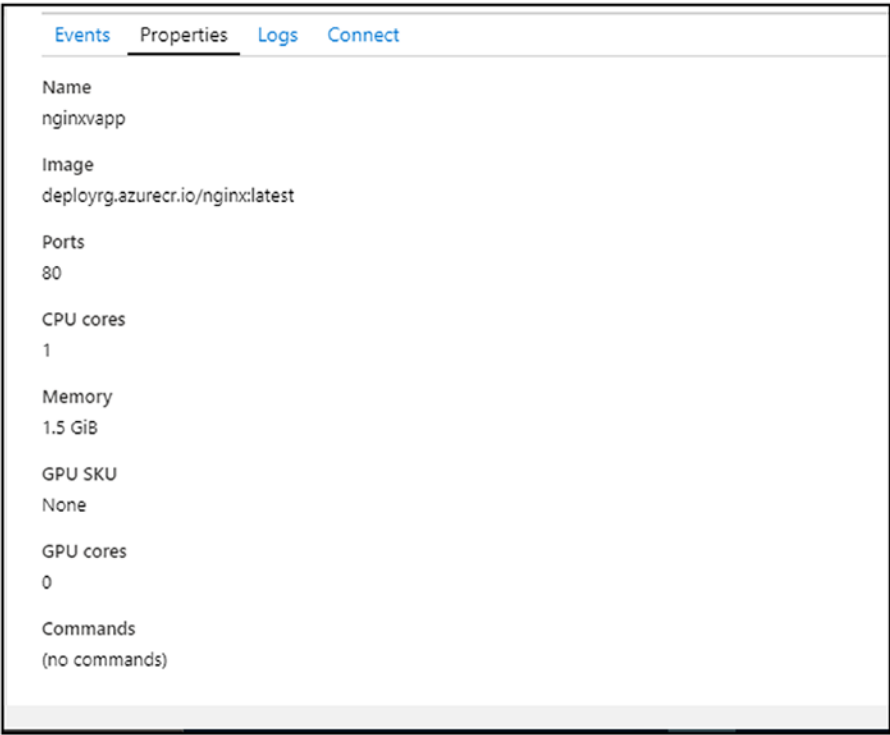


Figure 3-33. Container groups Properties tab

If we click on the Logs tab, we can see the actual logs the container produces during startup and during the normal running state. This view is like the Docker logs command.

Figure 3-34 shows the Logs output screen.

Events Properties Logs Connect			
10.240.255.56	-	[29/Oct/2019:04:46:37 +0000]	"GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0"
10.240.255.56	-	[29/Oct/2019:04:46:37 +0000]	"GET /favicon.ico HTTP/1.1" 404 555 "http://nginxvol.westus.azurecontainer.io/"
10.240.255.56	-	[29/Oct/2019:04:46:37 +0000]	2019/10/29 04:46:37 [error] 5#5: *37 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory) while serving request, client: 10.240.255.56, server: nginxvol.westus.azurecontainer.io, referer: "http://nginxvol.westus.azurecontainer.io/"
10.240.255.56	-	[29/Oct/2019:04:46:39 +0000]	"GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0"
10.240.255.56	-	[29/Oct/2019:04:46:39 +0000]	"GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0"
10.240.255.56	-	[29/Oct/2019:04:46:40 +0000]	"GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0"
10.240.255.56	-	[29/Oct/2019:04:46:40 +0000]	"GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0"
10.240.255.56	-	[29/Oct/2019:04:46:40 +0000]	"GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0"
10.240.255.56	-	[29/Oct/2019:04:46:41 +0000]	"GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0"
10.240.255.56	-	[29/Oct/2019:04:46:41 +0000]	"GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0"
10.240.255.56	-	[29/Oct/2019:04:46:42 +0000]	"GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0"
10.240.255.56	-	[29/Oct/2019:04:46:42 +0000]	"GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0"
10.240.255.56	-	[29/Oct/2019:04:46:42 +0000]	"GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0"
10.240.255.56	-	[29/Oct/2019:04:46:42 +0000]	"GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0"
10.240.255.56	-	[29/Oct/2019:04:46:43 +0000]	"GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0"
10.240.255.56	-	[29/Oct/2019:04:46:43 +0000]	"GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0"
10.240.255.56	-	[29/Oct/2019:04:46:43 +0000]	"GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0"
10.240.255.56	-	[29/Oct/2019:04:46:44 +0000]	"GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0"
10.240.255.56	-	[29/Oct/2019:04:46:44 +0000]	"GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0"
10.240.255.56	-	[29/Oct/2019:04:46:45 +0000]	"GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0"
10.240.255.56	-	[29/Oct/2019:04:46:45 +0000]	"GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:52.0) Gecko/20100101 Firefox/52.0"

Figure 3-34. Container groups Logs tab

The fourth and last option, which is my favorite, is the Connect tab, which allows us to connect to the container's shell using Bash directly from the browser.

This is a great feature. Earlier in this chapter, we used Azure CLI to do the same thing using Azure Cloud Shell.

Here, we can do the same thing but without Azure CLI and Azure Cloud Shell. This option is useful if you don't want to use Azure Cloud Shell.

Figure 3-35 shows the Connect option.

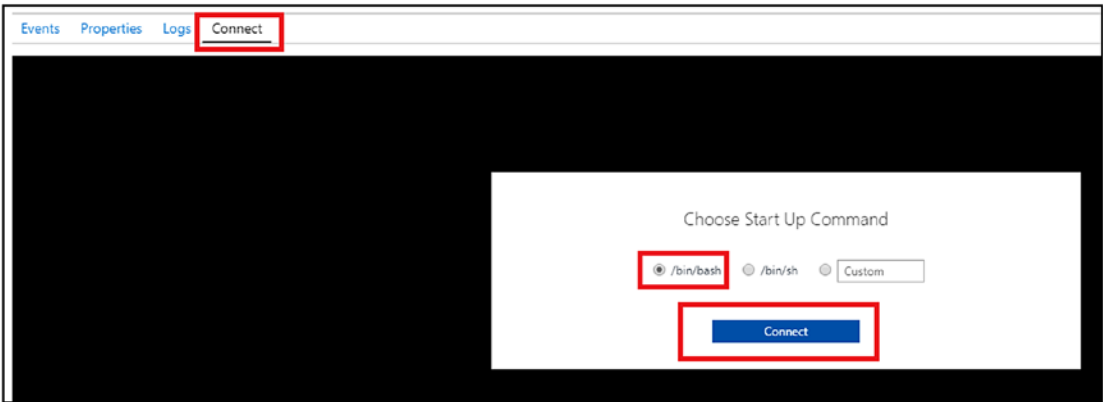


Figure 3-35. Container groups Connect tab

As you can see in Figure 3-36, I am connected to my container using the Bash Shell and issued the `ls` command.



Figure 3-36. Bash Shell `ls` command

To help you manage your deployment using Azure CLI and Azure Cloud Shell, I have listed a few `az container` commands that will help you manage your deployment.

The following command will list the details of a single container:

```
az container show --resource-group apps --name nginxvapp
```

To list all running container groups, I will use the following command:

```
az container list --resource-group apps -o=table
```

To connect to the container's console using Bash Shell, I can use the following command:

```
az container exec -g apps --name nginxvapp --container-name nginxvapp  
--exec-command "/bin/bash"
```

To restart a container, I would use the following command:

```
az container restart --resource-group apps --name nginxvapp
```

To stop a container, I can use the following command:

```
az container stop --resource-group apps --name nginxvapp
```

To start a container, I would use the following command:

```
az container start --resource-group apps --name nginxvapp
```

To delete a container, I can use the following command:

```
az container delete --name nginxvapp --resource-group apps
```

Manage ACI with Azure Resource Explorer

If you are a developer and would like to use APIs to manage your containers, you could use Azure Resource Explorer and use API calls to manage your resources.

If you would like to check all the available API calls, log in to the following URL with your Azure account: <https://resources.azure.com/>

For the portal, use the left-side menu to explore all the API options. Figure 3-37 shows the Azure Resource Explorer portal.

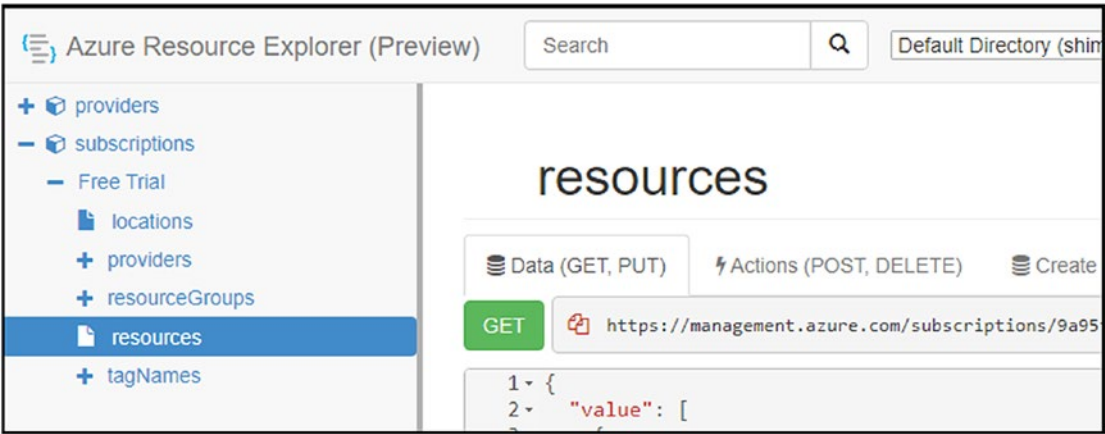


Figure 3-37. *Azure Resource Explorer portal*

Summary

In this chapter, we took a deep dive into the world of Azure Container Instances (ACI) and learned how to deploy Linux and Windows containers.

We also learned how to use Azure file share volumes and mount them to ACI Linux containers.

In the last section, we learned how to monitor and manage our containers using Azure Monitor and Azure CLI.

In the next chapter, we will learn how to deploy containerized applications with the Azure Kubernetes Service (AKS).

CHAPTER 4

Deploy Containerized Applications with Azure Kubernetes Service (AKS)

So far, we have covered the two main container services on Azure, and now we will move to the third service, which is considered an enterprise-ready solution. In this chapter, we will take a deep dive into the Microsoft Azure Kubernetes Service (AKS) and learn how to deploy and manage containerized applications with AKS. AKS is the most complex container service in Azure, and I recommend you follow all steps in this chapter.

The following topics will be covered in this chapter:

- Overview of Kubernetes
- Getting started with AKS
- Creating an AKS cluster using the Azure portal
- Creating an AKS cluster using Azure CLI and Azure Cloud Shell
- Connecting ACR to an AKS cluster
- Deploying containers using Docker images stored in ACR
- Deploying Kubernetes Web UI (Dashboard) on AKS
- Deploying persistent storage volumes with AKS
- Deploying pods with persistent storage
- Managing and securing AKS

Kubernetes

Kubernetes is an orchestration and automation container management system that was developed by Google and was open-sourced in 2014.

Kubernetes is so powerful that it was used by Google to run over one billion containers and to power many of their cloud services.

The scalability of Kubernetes is almost unlimited, which offers enterprises the ability to scale up or down in a short period of time.

It is currently considered the standard tool for container management and automation in both small and large organizations and has a market share of almost 80 percent and growing.

There is no doubt that Kubernetes is the go-to tool for orchestration and automation.

Kubernetes Components

In this section, I will cover each of the components that makes up Kubernetes so that once we move to AKS, we can better understand it.

Kubernetes is made up of three main components, each of which has subcomponents.

Kubernetes Master

This is the most important component in Kubernetes, and as the name implies, this is the master node, also known as the control plane.

The Kubernetes Master is made of five subcomponents:

Kube-apiserver — This is the API server that exposes all the APIs of Kubernetes to other components.

Etc — This is the key-value store component that stores the cluster data.

Kube-Scheduler — The scheduler makes sure new pods are matched with a node.

Kube controller-manager — The controller manager is responsible for managing nodes, replication between the nodes and the master, endpoints, and tokens. Each controller is a separate logical component. However, they are managed as a single process on the master.

Cloud-controller manager — This is a new component that was released with version 1.6, and it helps cloud providers control updates and releases of their Kubernetes deployments.

Kubernetes Node

In Kubernetes, the nodes are the actual servers that run the containers in pods. The nodes are the computing unit that performs the heavy lifting of deploying pods, volumes, and networking.

The node component has three subcomponents:

Kubelet — This is the agent service that runs on each node and communicates with the master node while making sure pods are running as planned.

Kube Proxy — This is the network component that makes sure services are operated according to the network policies and rules set by the master component.

Container Runtime — This is the runtime container software that runs our containers, and in our case this is where Docker is running.

Kubernetes Add-ons

Add-ons are features that are optional and can be added to a Kubernetes cluster.

The following add-ons are just a few of the many available:

DNS — This is a DNS service that serves DNS name resolution for Kubernetes; by default all Kubernetes components use this.

Web UI (Dashboard) — Web UI is a browser-based interface for Kubernetes management and allows us to perform administrative tasks using a web-based interface.

Container resource monitor — This logs metrics about running containers inside the cluster.

Cluster-level logging — This logs information about our Kubernetes cluster.

Now that we have some background information about Kubernetes, we can move to the next section, where we will cover Azure implementation of Kubernetes.

Getting Started with AKS

Before we get started with creating our first cluster on AKS, it is important we understand the management domain of AKS.

In the previous section, we learned about all the Kubernetes components and what each of them does.

In AKS, Microsoft Azure is responsible for managing the control plane. The control plane is beyond our reach, and we don't have access to it. This eliminates the complexity of managing Kubernetes and leaves us with the management of the nodes. This includes basic administrative tasks like scaling and updating them.

In this section, we will go over the deployment process of AKS using the portal and Azure CLI using Azure Cloud Shell.

Creating an AKS Cluster

To create our first AKS cluster, we will use the Azure portal.

From the Azure portal, click "Create a resource." From the resources list, click on "Containers." Click on "Kubernetes Service."

Figure 4-1 shows the Kubernetes Service (AKS) from the Azure portal.

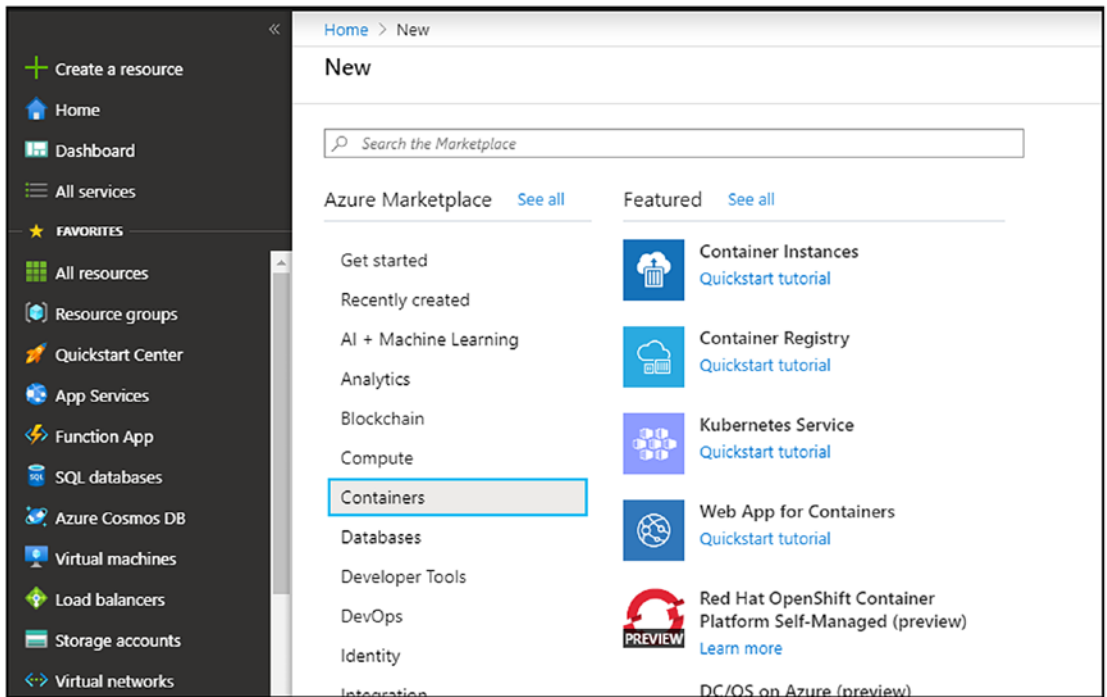


Figure 4-1. *Kubernetes service*

From the Kubernetes Service home page, click “Add,” and you will see the Create Kubernetes Cluster setup wizard, as shown in Figure 4-2.

The wizard gives you the option to customize the setup of the new cluster, and I will guide you through the process.

In the first section, you need to select your Azure subscription and an existing resource group.

In my case, I will use the same resource group I have ACR in.

The region will be set based on the resource group.

Figure 4-2 shows the Create Kubernetes Cluster setup page.

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 subscription

Resource group * ⓘ

Apps

Create new

Cluster details

Kubernetes cluster name * ⓘ

aks

Region * ⓘ

(US) West US

Kubernetes version * ⓘ

1.13.12 (default)

DNS name prefix * ⓘ

aks-dns

Figure 4-2. *Create Kubernetes Cluster setup page*

Under the Kubernetes version selection, you have the option to select the Kubernetes version you would like your cluster to run.

Note AKS only allows you to upgrade the Kubernetes version and not downgrade it, so make sure you start with a version that you know is compatible with your apps.

Figure 4-3 shows all the available Kubernetes versions.

Note Please avoid using preview versions in a production workload.

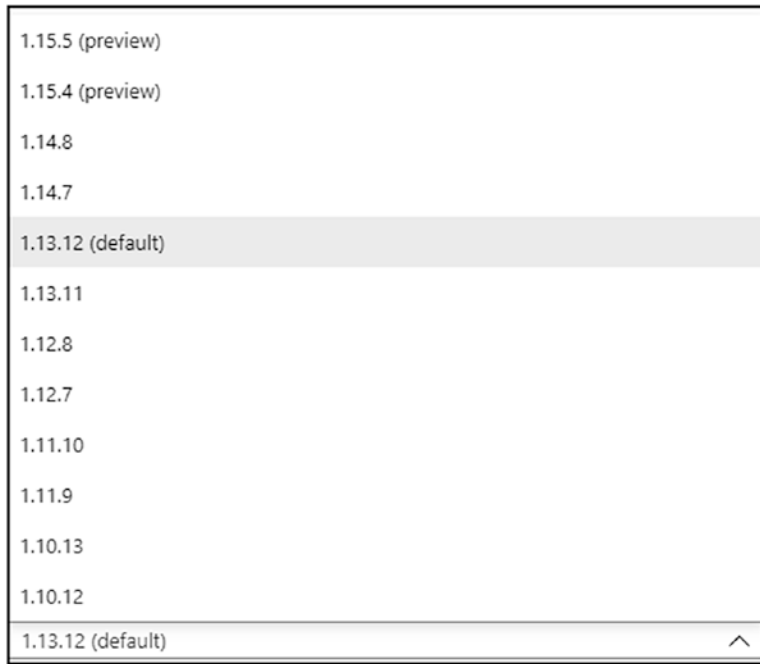


Figure 4-3. *Kubernetes software version*

In the lower section of the wizard, you have the option to select the number of nodes your cluster will run. By default, Azure configures the cluster with three nodes, which is good for production workloads; however, if you are deploying a development cluster you can change this value to 1.

Figure 4-4 shows the node count with the default node number. You can always scale the cluster later on as needed.

I also recommend you change the default node size to Standard DS1 V2 if you are not deploying a production cluster.



Figure 4-4. *Node count*

The next screen in the setup shows the scale options, and in my case, I will keep the default values, as shown in Figure 4-5.

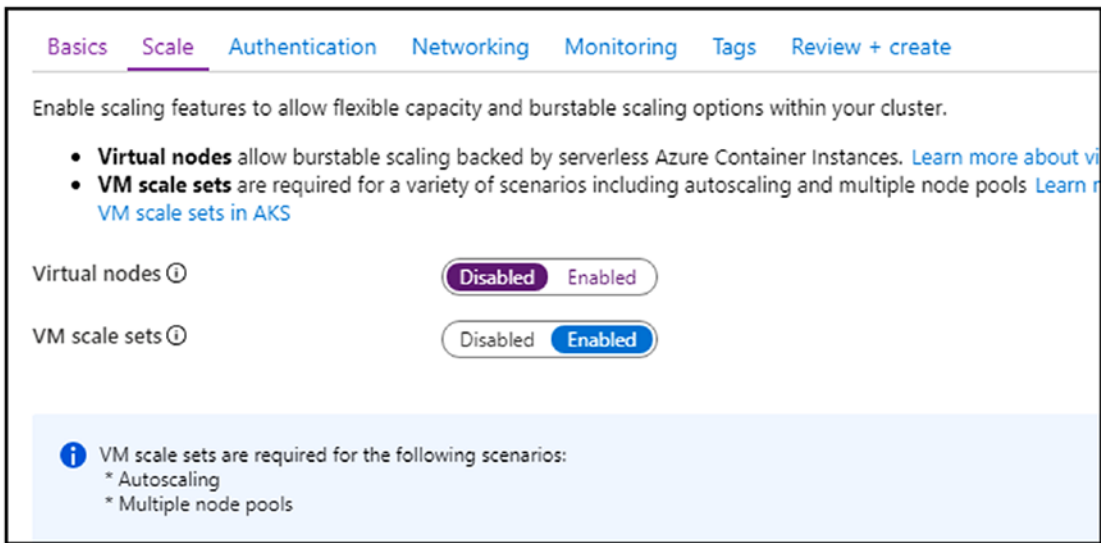


Figure 4-5. Scale options

In the Authentication screen, we have the option to enable or disable RBAC authentication, and I recommend you leave the default, which enables RBAC.

Figure 4-6 shows the Authentication screen.

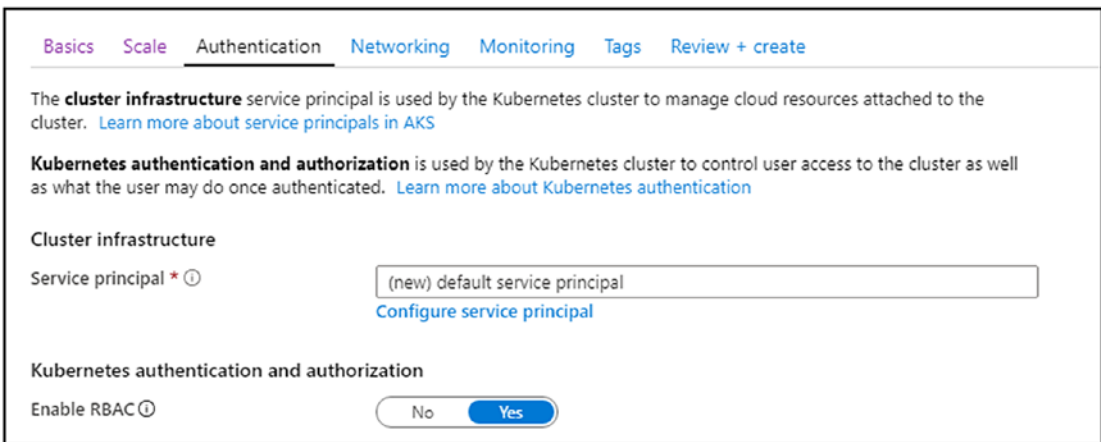


Figure 4-6. Authentication screen

In the Networking screen, we have the option to configure advanced networking or use basic. In my case, I will use the basic network configuration; however, for large deployments, you might need to configure an advanced network. See Figure 4-7.

Basics Scale Authentication **Networking** Monitoring Tags Review + create

You can change networking settings for your cluster, including enabling HTTP application routing and configuring your network using either the 'Basic' or 'Advanced' options:

- **'Basic'** networking creates a new VNet for your cluster using default values.
- **'Advanced'** networking allows clusters to use a new or existing VNet with customizable addresses. Application pods are connected directly to the VNet, which allows for native integration with VNet features.

[Learn more about networking in Azure Kubernetes Service](#)

HTTP application routing ☒ Yes ☐ No

Load balancer Standard

Network configuration ☒ Basic ☐ Advanced

Figure 4-7. *Networking configuration*

In the Monitoring screen, we have the option to enable container monitoring using Azure Monitor. This service is similar to the service we used in ACI; however, in AKS Azure will enable the service and create a log analytics workspace for you.

Note Log analytics is a paid service.

In my case, I will leave the default settings to enable monitoring, as shown in Figure 4-8.

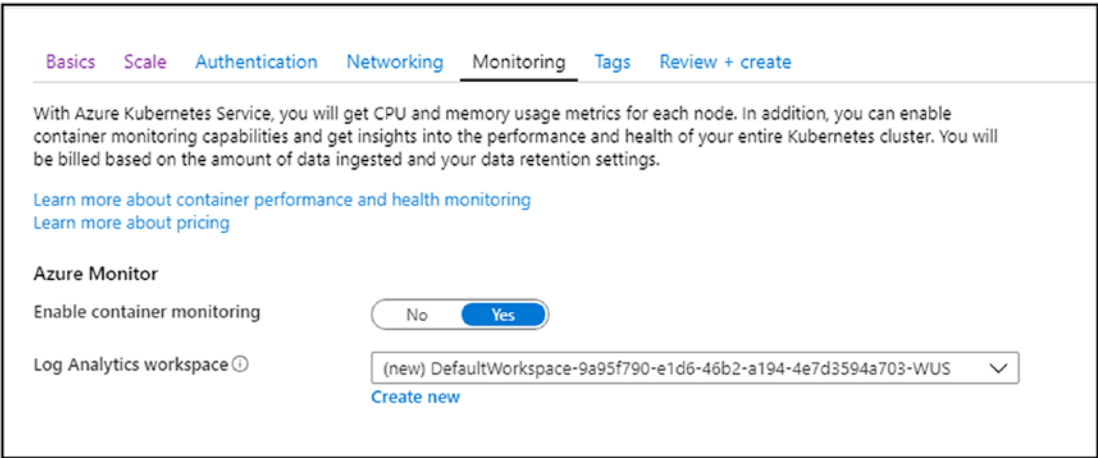


Figure 4-8. *Monitoring screen*

In the Tags screen, you can configure tags for billing purposes.
And finally, in our last screen, Review + Create, we can review the configuration before creating the cluster, as shown in Figure 4-9.

Basics	Scale	Authentication	Networking	Monitoring	Tags	Review + create
Basics						
Subscription	Azure subscription					
Resource group	Apps					
Region	(US) West US					
Kubernetes cluster name	aks					
Kubernetes version	1.13.12					
DNS name prefix	aks-dns					
Node count	1					
Node size	Standard_DS1_v2					
Scale						
Virtual nodes	Disabled					
VM scale sets	Enabled					
Authentication						
Enable RBAC	Yes					
Networking						
HTTP application routing	No					
Load balancer	Standard					
Network configuration	Basic					
Monitoring						
Enable container monitoring	Yes					

Figure 4-9. Review AKS setup configuration

After clicking on “Creating” it will take Azure around ten minutes to provision the cluster and make it fully ready. Once it is completed, you will see it in the Kubernetes Services page. Figure 4-10 shows my newly created AKS cluster.

Kubernetes services				
Default Directory				
+ Add	≡ Edit columns	↺ Refresh	↓ Export to CSV	🏷 Assign tags
	❤ Feedback	↺ Leave preview		
Filter by name...	Subscription == all	Resource group == all	Location == all	+ Add filter
Showing 1 to 1 of 1 records.				
☐ Name ↑↓	Type ↑↓	Resource group ↑↓	Location ↑↓	
☐  aks	Kubernetes service	Apps	West US	

Figure 4-10. AKS cluster

Creating an AKS Cluster Using Azure Cloud Shell

If you choose not to use the Azure portal, you can use Azure Cloud Shell and Azure CLI commands to create the same cluster we used the portal to create.

Azure CLI uses the `az aks` syntax to create and manage AKS clusters and deployments. The same process we used in the wizard can be done using the following Azure CLI script.

I called the following bash script, `1.Create_AKS.sh`, which you can run from Azure Cloud Shell.

Note Make sure you change the resource group name.

The script creates an AKS cluster called `aks` with a single node and a VM size `Standard_DS1_V2`.

It also enables Azure monitor. See the following:

```
#!/bin/bash
az aks create --resource-group apps \
  --name aks \
  --node-count 1 \
  --node-vm-size Standard_DS1_v2 \
  --enable-addons monitoring \
  --generate-ssh-keys $true
```

You can run the script by using the following command after you copy it to Azure Cloud Shell:

```
. 1.Create_AKS.sh
```

It will take the same amount of time for Azure to create the AKS cluster.

Connecting ACR to AKS

If you remember from Chapter 2, we created our first Azure Container Registry (ACR) and learned how to push and pull images from it.

Now that we have our AKS cluster up and running, we will connect it to our ACR image registry and use it to pull images and deploy them to our AKS cluster.

To connect the two services, we will use Azure Cloud Shell; we will need to use our ACR registry details. Once we have all the details, we will run a few commands from Azure Cloud Shell and connect the two services, as you will see.

We will use the following two Azure CLI commands; the first command will authenticate us with our AKS cluster.

Note Make sure you use your resource group name and AKS cluster name:

```
az aks get-credentials --resource-group apps --name aks
```

In the following command, I will update my AKS cluster with my ACR registry name. You will find your ACR name in the ACR portal on the Access Keys page.

```
az aks update -n aks -g apps --attach-acr DeployRG
```

Figure 4-11 shows the two commands and the output you should see when you connect your ACR registry to AKS.

```
shimon@Azure:~$ az aks get-credentials --resource-group apps --name aks
Merged "aks" as current context in /home/shimon/.kube/config
shimon@Azure:~$ az aks update -n aks -g apps --attach-acr DeployRG
{
  "aadProfile": null,
  "addonProfiles": {
    "omsagent": {
      "config": {
        "logAnalyticsWorkspaceResourceID": "/subscriptions/9a95f790-e1d6-46b2-a194-4e7d3594a7
.operationalinsights/workspaces/defaultworkspace-9a95f790-e1d6-46b2-a194-4e7d3594a703-wus"
      },
      "enabled": true
    }
  }
}
```

Figure 4-11. Commands to connect ACR to AKS

Deploying Containers Using an ACR Image to AKS

If you have followed all the preceding sections and subsections, you are ready to deploy your first container to AKS.

At this stage, we have an AKS cluster that is up and running and is also connected to our ACR container registry.

Now it is time to test and deploy the Nginx image I have stored on ACR to my AKS cluster. To do so, I will use Azure Cloud Shell using the `kubectl` command line.

In the following command, I create a deployment called `nginx` using an image stored in ACR. Please take note of the format I am using and change the details to match your ACR details.

```
kubectl create deployment nginx --image=deployrg.azurecr.io/nginx:latest
```

At this stage, the deployment has started and AKS will create a single pod to run the image. In the following line, I will expose port 80 to the Nginx web server and allow access to it using a web browser.

Note Without exposing the deployment, you can't access the service.

```
kubectl expose deployment nginx --type=LoadBalancer --port 80 --name=nginxhttp
```

After exposing the deployment, AKS will create a service called `nginxhttp` with port 80 allowed. This process is similar to opening a port on a firewall.

To check if the service is ready, I will use the following command:

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

When the service is ready, you will see the public IP address where the Nginx web server is available.

In Figure 4-12, you can see the end result of the deployment and service, where Nginx is now accessible via a public IP address.

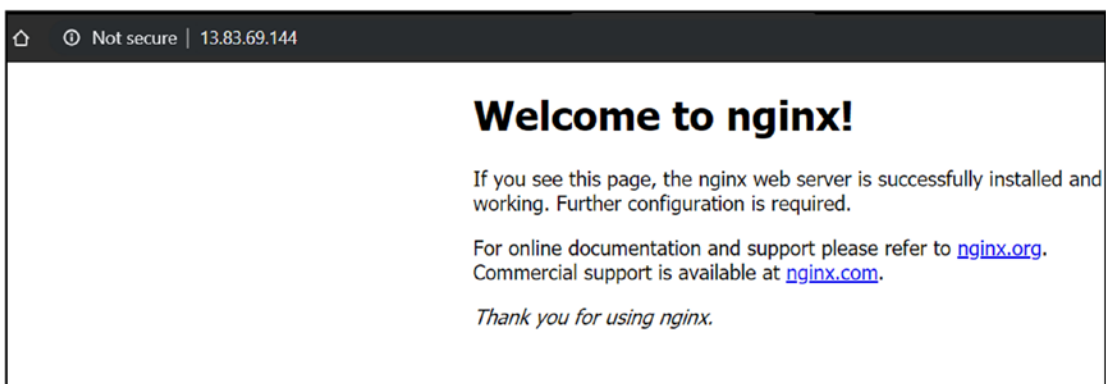


Figure 4-12. *Nginx web server*

We can achieve the same result using a YAML file deployment.

The following YAML file will create the same deployment and expose the port automatically. Copy the following code to Azure Cloud Shell and run it:

```
kubectl apply -f 4.3.nginx.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: deployrg.azurecr.io/nginx:latest
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginxhttp
spec:
  type: LoadBalancer
  ports:
    - port: 80
  selector:
    app: nginx
```

To get the external IP address of the deployment, run the following command:

```
Kubectl get service
```

To delete the deployment after you are done testing, run the following code:

```
Kubectl delete deployment nginx
```

To delete the service, run the following command:

```
Kubectl delete service nginxhttp
```

Deploy Kubernetes Web UI (Dashboard) on AKS

The Kubernetes Dashboard, also known as the Web UI, is a web-based management tool that allows us to manage and deploy containerized applications to Kubernetes regardless if it is hosted in the cloud or inside the corporate network.

By default, Dashboard is not installed with on-premises Kubernetes deployments; however, in AKS the dashboard is pre-loaded but is not available until we configure an RBAC user to access it and start it.

Personally, I recommend you install it and see how it works. However, I suggest you use it as a secondary tool and learn how to use the kubectl command-line tool.

Figure 4-13 shows the Dashboard of my AKS cluster.

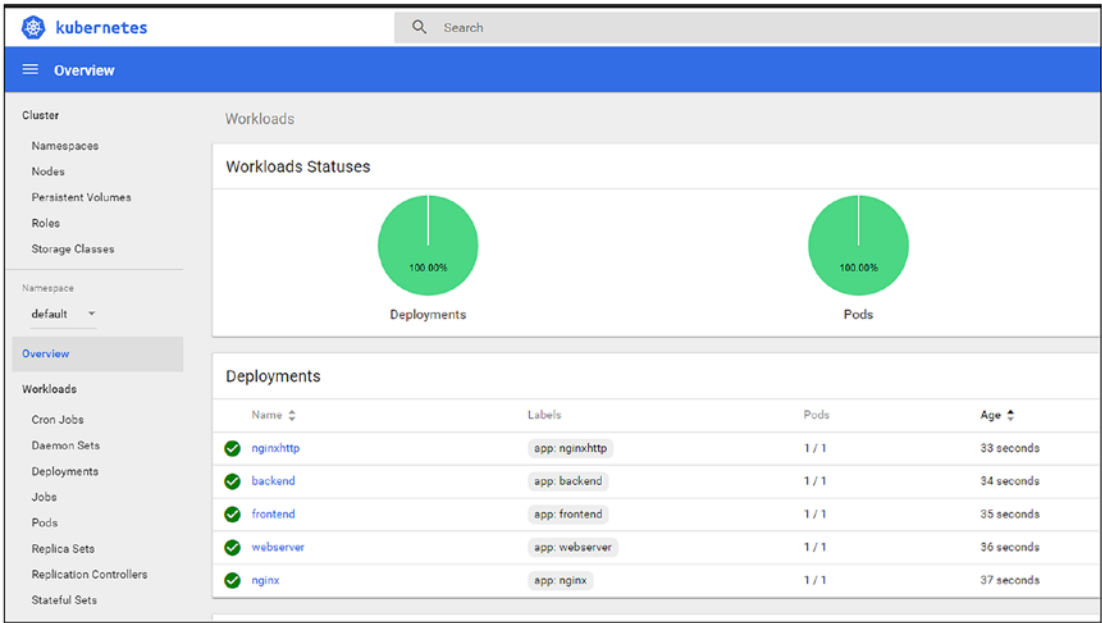


Figure 4-13. *Kubernetes Dashboard*

Enable Web UI (Dashboard) on AKS

Enabling Dashboard on AKS is much simpler than enabling it on an on-premises cluster.

It is important you understand how Kubernetes and AKS run via the Dashboard. The Dashboard, once started, gets deployed into the AKS cluster as a pod, and like any deployment as long as the pod runs the Dashboard is available.

Because AKS already has the Dashboard container image loaded, all I need to do is start it. Before we can start it in AKS, we need to run the following commands.

The first command will authenticate us to the AKS cluster:

```
az aks get-credentials --resource-group apps --name aks
```

The second command, which needs to run once, will create an RBAC user that we will use to read and write information to the cluster using the Dashboard:

```
kubectl create clusterrolebinding kubernetes-dashboard --clusterrole=cluster-admin --serviceaccount=kube-system:kubernetes-dashboard
```

The last command will start the Dashboard:

```
az aks browse --resource-group apps --name aks
```

By default, the command should start your browser and redirect you to the Dashboard; if this is not the case, click on the Dashboard link, as shown in Figure 4-14.

```
shimon@Azure:~$ kubectl create clusterrolebinding kubernetes-dashboard --clusterrole=cluster-admin --serviceaccount=kube-system:kubernetes-dashboard
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
shimon@Azure:~$ az aks browse --resource-group apps --name aks
Merged "aks" as current context in /tmp/tmp2d_vig36
To view the console, please open https://gateway15.southeastasia.console.azure.com/n/cc-203207ff/cc-203207ff/proxy/8001/ in a new tab
Press CTRL+C to close the tunnel...
```

Figure 4-14. Start Dashboard (Web UI)

Deploy with Dashboard (Web UI)

Web UI allows us to deploy containerized applications using the browser without the `kubectl` command line.

The Dashboard user interface for deployments is very intuitive and simple to follow.

I will go ahead and show you how I deploy my Nginx web server image, which is stored in ACR, into my AKS cluster using the Dashboard.

From the Dashboard, click on “+Create,” which is located on the top-right corner, as shown in Figure 4-15.

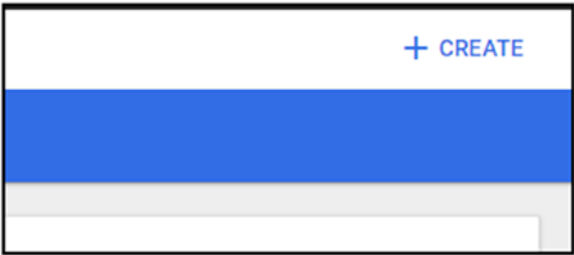


Figure 4-15. *Create*

The Create page will give you three options, as shown at the top of Figure 4-16.

The first option can create a deployment from a text file, where you can type or copy the content of a YAML or JSON file.

The second option allows you to upload a YAML and run it.

The third option, which is more user friendly, allows you to create an app and specify the Docker image, port, and other advanced options.

Figure 4-16 shows the Create an App option.

CREATE FROM TEXT INPUT

CREATE FROM FILE

CREATE AN APP

App name *

webdeploy

8 / 24

Container image *

deployrg.azurecr.io/nginx:latest

Number of pods *

2

Service *

External

Port *

80

Target port *

80

Protocol *

TCP

Port

Target port

Protocol *

TCP

Figure 4-16. *Create an app*

If you look closely, you will see that I have named the app and set it to use my Nginx image, which is stored in ACR.

I'm also opening port 80 to external access.

To complete the deployment, I will click on the “Deploy” button, as shown in Figure 4-17.

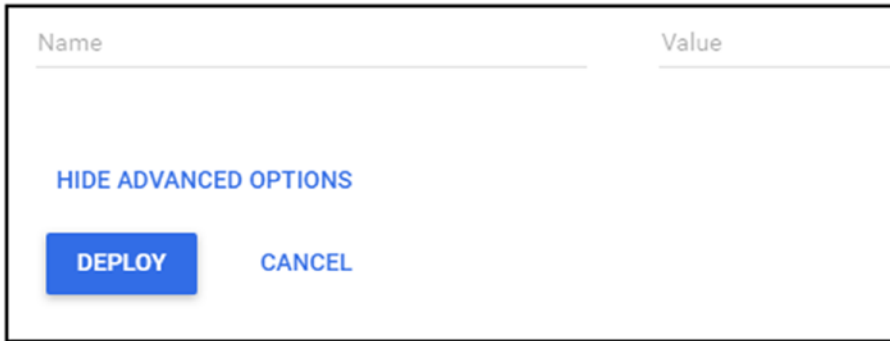


Figure 4-17. *Deploy*

After deploying the app, I can monitor the deployment and see if it was successful or not using the Workloads Statuses page, as shown in Figure 4-18.

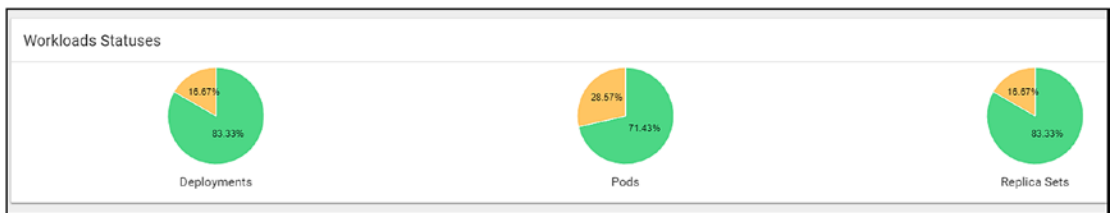


Figure 4-18. *Workloads Statuses page*

To test the deployment, I will click on “Services” in the Dashboard’s left-hand menu.

As you can see in Figure 4-19, my application, which is called webdeploy, is running, and if I click on the link under “External endpoints,” I will redirect the application.

Services				
Name ↕	Labels	Cluster IP	Internal endpoints	External endpoints
✔ webdeploy	k8s-app: webdeploy	10.0.71.68	webdeploy:80 TCP webdeploy:30196 TCP	13.86.255.219:80 ↗
✔ kubernetes	component: apiserver provider: kubernetes	10.0.0.1	kubernetes:443 TCP	-

Figure 4-19. *Services*

The end result can be seen in Figure 4-20; my Nginx web server is available via external domain.

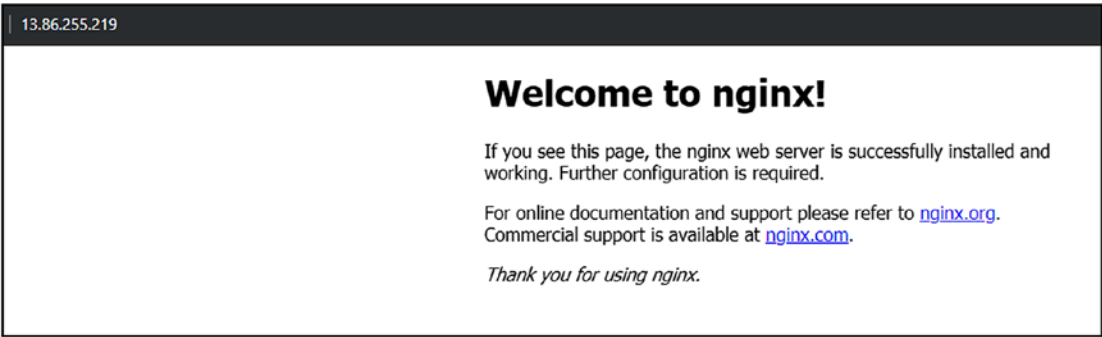


Figure 4-20. *Nginx web server*

Mount Storage Volumes in AKS

In this section, I will show you how to mount a persistent storage volume to AKS and use in containerized applications.

This process is similar to the process we used in Chapter 3, where we mounted a storage volume to our ACI deployment.

This process in AKS is a bit more complicated because of the complexity of the solution and the fact that it is an enterprise product.

In the following example, I will show you how I mount a persistent volume that can be dynamically provisioned to one or more pods. I will follow this process with the following YAML files:

1. Create a storage class using file 4.5.Create_Storage_Class.yaml
2. Create a storage role using file 4.6.Create_Storage_Role.yaml

3. Create a volume claim using file `4.7.Create_Volume_Claim.yaml`
4. Create a deployment with persistent storage using file `4.8.Create_Pod_With_Volume.yaml`

Once the storage is configured and deployed, I will deploy my Nginx web server and mount a persistent volume to it.

All the data that is saved in the mounted storage will remain intact after I delete the pods.

Create a Storage Class

To create a storage class, I will run the following YAML file, which holds the configuration of the storage class.

Note The storage class is using `Standard_LRS` storage, which is locally redundant storage (LRS); if you would like to use geo-redundant storage, use `Standard_GRS`.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: azurefile
provisioner: kubernetes.io/azure-file
mountOptions:
  - dir_mode=0777
  - file_mode=0777
  - uid=1000
  - gid=1000
  - mfsymlinks
  - nobrl
  - cache=none
parameters:
  skuName: Standard_LRS
```

Once you are ready, copy the file to Azure Cloud Shell and run it using the following command:

```
kubectl apply -f 4.5.Create_Storage_Class.yaml
```

Create a Storage Role

In this step, I will create an RBAC cluster role that will allow the AKS cluster to access the storage using an RBAC role.

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: system:azure-cloud-provider
rules:
- apiGroups: [""]
  resources: ['secrets']
  verbs:      ['get', 'create']
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: system:azure-cloud-provider
roleRef:
  kind: ClusterRole
  apiGroup: rbac.authorization.k8s.io
  name: system:azure-cloud-provider
subjects:
- kind: ServiceAccount
  name: persistent-volume-binder
  namespace: kube-system
```

Once you are ready, copy the file to Azure Cloud Shell and run it using the following command:

```
kubectl apply -f 4.6.Create_Storage_Role.yaml
```

Create a Volume Claim

In this step, I will create a volume claim, which is also known as a persistent volume claim (PVC). The claim uses the storage class, which is in Azure files behind the scenes.

Note The claim creates a 10 GB volume by default. To change it, change the storage value in the last line.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: azurefile
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: azurefile
resources:
  requests:
    storage: 10Gi
```

Once you are ready, copy the file to Azure Cloud Shell and run it using the following command:

```
kubectl apply -f 4.7.Create_Volume_Claim.yaml
```

Create a Deployment with Persistent Storage

Now that the persistent storage is configured, I will deploy my Nginx web server and mount a 10 GB storage volume to it. To do that, I will use the following YAML file to deploy to the pod.

The last four lines are responsible for mounting the storage volume to the pod. If you look closely, you will see that I am using the azurefiles volume claim, which I created in the preceding step.

Note The mount point is called aks.

```
kind: Pod
apiVersion: v1
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: deployrg.azurecr.io/nginx:latest
    resources:
      requests:
        cpu: 100m
        memory: 256Mi
      limits:
        cpu: 250m
        memory: 256Mi
    volumeMounts:
    - mountPath: "/aks"
      name: volume
  volumes:
  - name: volume
    persistentVolumeClaim:
      claimName: azurefile
```

And, finally, I will copy the file to Azure Cloud Shell and run it using the following command:

```
kubectl apply -f 4.8.Create_Pod_With_Volume.yaml
```

To test the deployment and see if the storage was mounted successfully, I will connect to the running pod using the `kubectl exec` command and start the Bash Shell utility using the following command:

```
kubectl exec -it nginx -- /bin/bash
```

After connecting to the pod, I will use the `ls` command to view all the directories.

As you can see in Figure 4-21, I can see my `aks` folder, which is mounted to the Azure files storage volume.


```

shimon@Azure:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
backend-7cbb968c7-2hgd6             1/1     Running   0           51m
frontend-7468cbc75c-mv58c           1/1     Running   0           51m
nginx                                1/1     Running   0           2m35s
nginx-67d795d89c-d56gp              1/1     Running   0           51m
nginxhttp-56fcdff869-ldzdd          1/1     Running   0           51m
webdeploy-7447768675-6bkt8          1/1     Running   0           41m
webdeploy-7447768675-gx4dj          1/1     Running   0           41m
webserver-64b5f8d8c4-dhv9f          1/1     Running   0           51m
shimon@Azure:~$ kubectl exec -it nginx -- /bin/bash
root@nginx:/# ls
aks bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
root@nginx:/#

```

Figure 4-21. Storage view inside the pod

Manage and Secure AKS

In the last section of this chapter, we will focus on the management options Azure gives us to manage and secure AKS. The main focus in this section will be the AKS portal inside the Azure portal and the primary management options.

The AKS cluster overview section gives us important information about our cluster, like its deployment status, location, Kubernetes version, and subscription details.

Figure 4-22 shows the AKS Overview page.

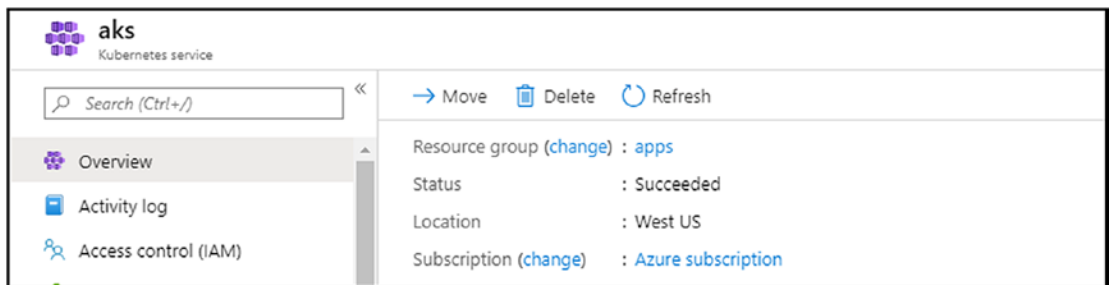


Figure 4-22. AKS Overview page

In the left pane menu, under the Settings section, you will see some of the configuration options that are configurable using the portal.

Some options are not available from the portal and require the use of Azure CLI; for example, connecting your AKS cluster to ACR, which can only be configured using the CLI.

Figure 4-23 shows the Settings section in the left pane.

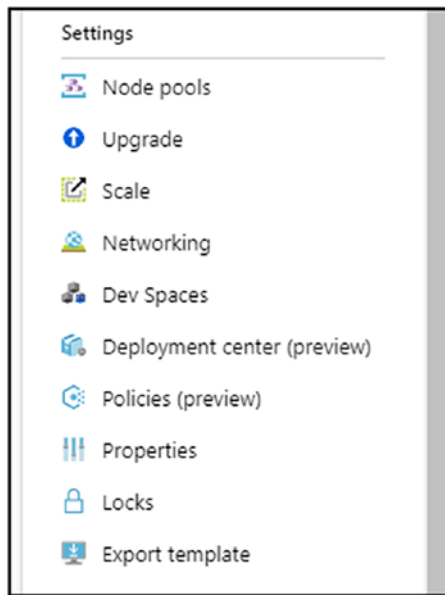


Figure 4-23. *Settings menu*

AKS node pools are where our AKS Kubernetes worker nodes are located.

Note If you remember, in the first section of this chapter, we learned that we could not control or access the master nodes.

Node pools contain virtual machines (nodes) that have the same configuration under a logical group. The use of node pools allows us to deploy multiple node pools with nodes that have different virtual machine size and allocate resources to them.

By default, Azure creates a node pool for the Kubernetes nodes, regardless of whether you have one VM or more. The portal gives the option to scale the number of nodes in the pool and even to use autoscaling, which is in preview mode.

Figure 4-24 shows the Node Pools page and the options available from the action menu.

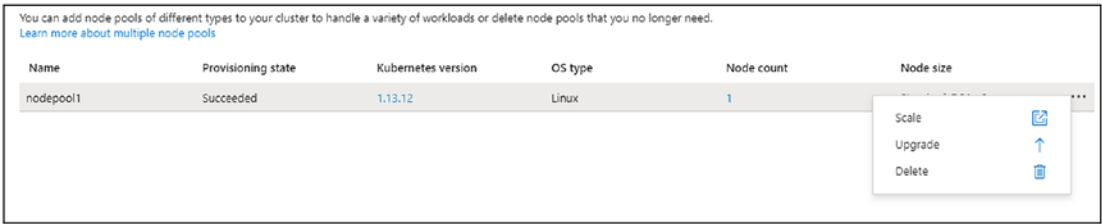


Figure 4-24. Node Pools page

From the Upgrade page, we can upgrade the Kubernetes version to a newer version; however, downgrading is not available. In Figure 4-25, you can see the Upgrade page and the drop-down menu with the available versions.

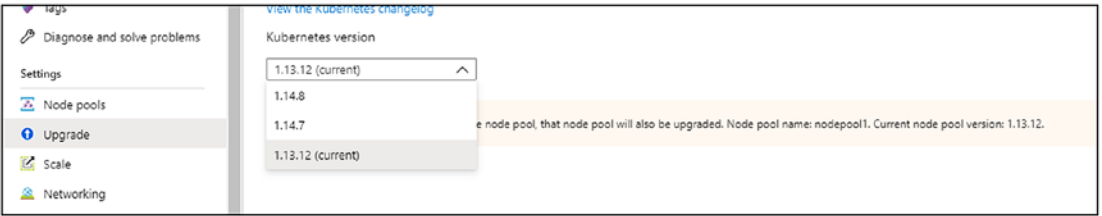


Figure 4-25. Upgrade AKS page

From the Scale menu, we can scale the number of nodes in the cluster, which is similar to the scale option in the Node Pools menu. See Figure 4-26.

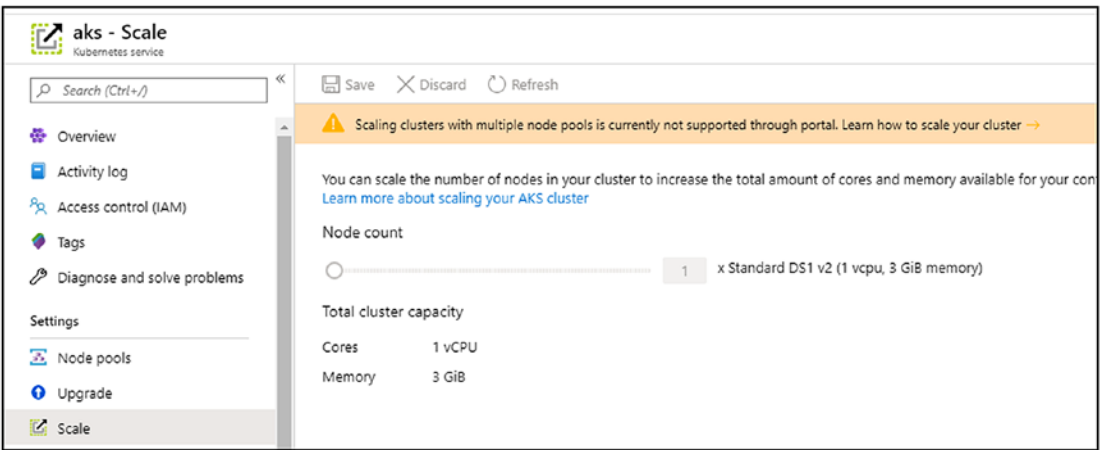


Figure 4-26. Scale AKS cluster

We can check if our AKS cluster needs to be scaled based on the cluster performance using the “Insights” link. You can see this in Figure 4-27.

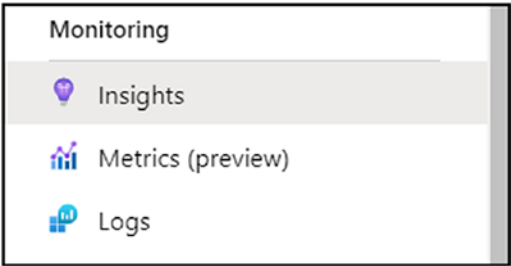


Figure 4-27. *Insights*

From the Insights page, we can check the cluster, nodes, controllers, and container performances and decide based on real-time and historical data if we need to scale our cluster.

Figure 4-28 shows the cluster CPU and memory performance.

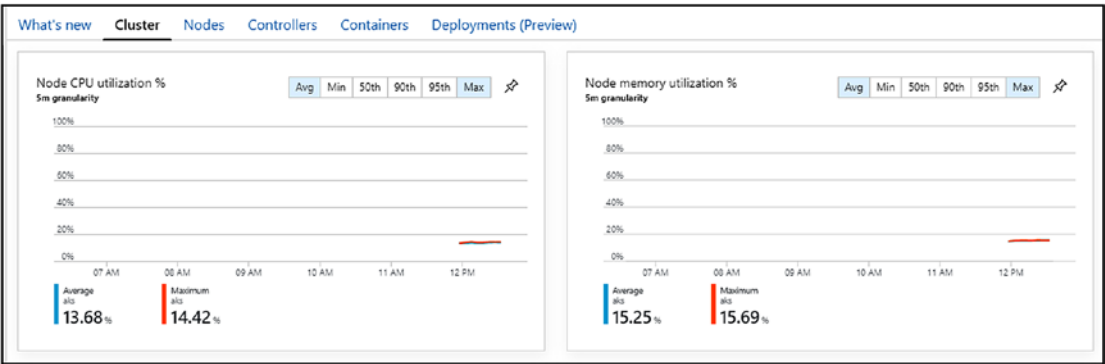


Figure 4-28. *Cluster CPU and memory*

The Nodes tab shows us the number of nodes we have in the cluster and also the node pools. We can also see the status of the nodes and their CPU utilization, uptime, and controllers running on the node.

On the right side, we have the option to view live data and real-time logs coming from the node. Figure 4-29 shows the Nodes tab.

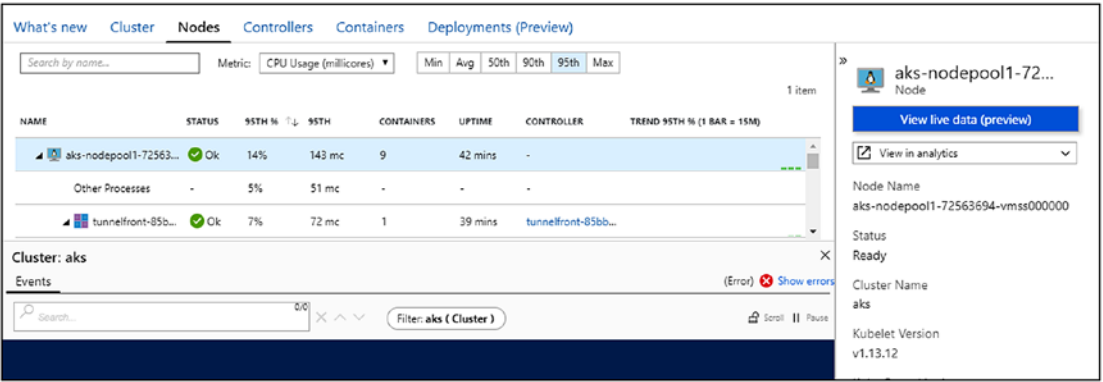


Figure 4-29. Nodes view

The Controller tab shows us all the running controllers and their statuses. If you remember, in the first section of this chapter we learned about them in the Kubernetes components section.

Figure 4-30 shows the Controllers tab with all running components.

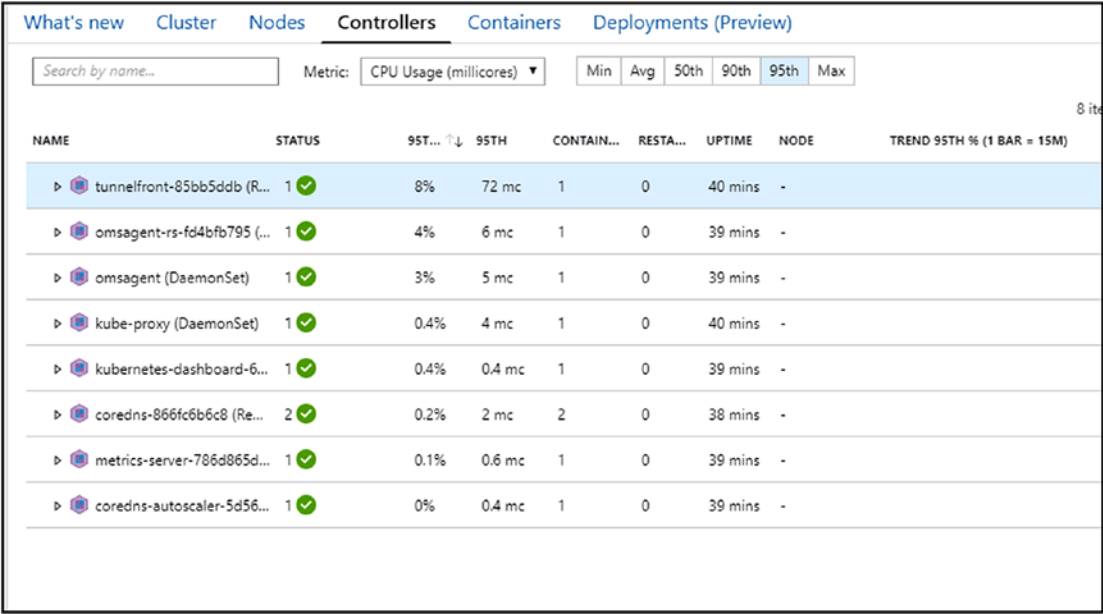


Figure 4-30. Controllers tab

We can also view all running containers in our AKS cluster from the Containers tab, which shows the number of running containers known as pods, as well as their usage.

If you look at Figure 4-31, you will see all the system containers and their statuses.

What's new Cluster Nodes Controllers Containers Deployments (Preview)									
Search by name...		Metric: CPU Usage (millicores)		Min Avg 50th 90th 95th Max					
NAME	STATUS	95TH...	95TH	POD	NODE	RESTARTS	UPTIME	TREND 95TH % (1 BAR = 15M)	
tunnel-front	Ok	8%	72 mc	tunnelfront-85...	aks-nodepool...	0	40 mins		
omsagent	Ok	4%	6 mc	omsagent-rs-f...	aks-nodepool...	0	39 mins		
omsagent	Ok	3%	5 mc	omsagent-574...	aks-nodepool...	0	39 mins		
kube-proxy	Ok	0.4%	4 mc	kube-proxy-g...	aks-nodepool...	0	40 mins		
main	Ok	0.4%	0.4 mc	kubernetes-da...	aks-nodepool...	0	40 mins		
coredns	Ok	0.2%	2 mc	coredns-866fc...	aks-nodepool...	0	39 mins		
coredns	Ok	0.2%	2 mc	coredns-866fc...	aks-nodepool...	0	40 mins		
metrics-server	Ok	0.1%	0.6 mc	metrics-server...	aks-nodepool...	0	40 mins		
autoscaler	Ok	0%	0.4 mc	coredns-autos...	aks-nodepool...	0	39 mins		

Figure 4-31. Containers tab

Lastly, I will show you how we can delegate access to other admins into our AKS cluster using RBAC.

If you click on “Access Control (IAM)” and click on “Add a role assignment,” you will be able to delegate a specific task to other admins. Figure 4-32 shows the Access Control page.

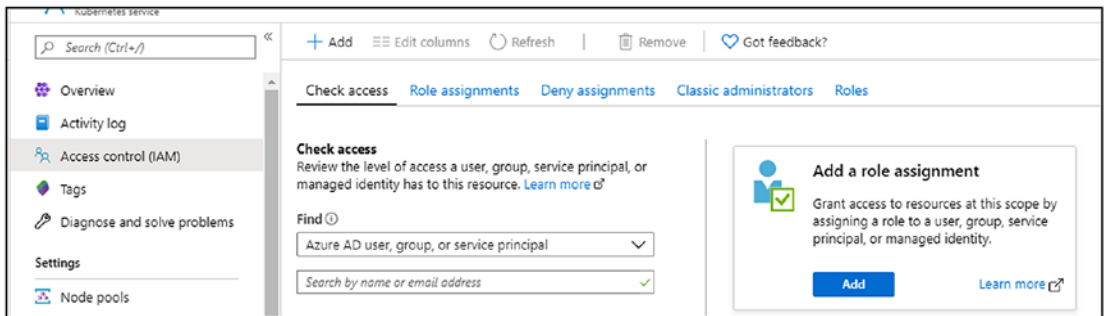


Figure 4-32. Access Control

When you click on “Add,” you will see the screen shown in Figure 4-33.

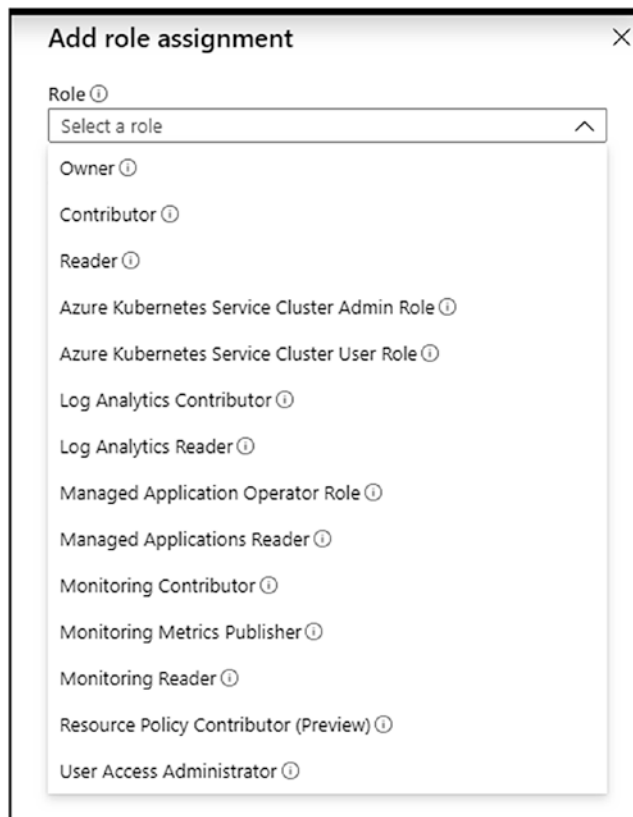


Figure 4-33. Add role assignments

From the Add Role Assignment page, you can select a specific role that has a set of tasks attached to it. If you would like to delegate monitoring permissions, you simply select one of the monitoring roles and then select the user.

Using RBAC to secure your AKS cluster is a quick win that can go a long way in forming your security strategy and road map.

Summary

This chapter was packed with technical information and labs about how to get AKS up and running in Azure.

We covered the basic Kubernetes components and what each one does. We also discussed the management domain of Azure and the fact that we don't have access to the Kubernetes master components.

After understanding the core components, we deployed our first cluster using the portal and by using Azure CLI via Azure Cloud Shell. We then deployed Web UI using Azure CLI and mounted a storage volume using Azure files. In the last section, we learned how to manage AKS using the Azure portal.

CHAPTER 5

Deploy Docker Container Host on Azure Virtual Machine

In this chapter, we will focus on deploying a Docker container host inside a virtual machine in Microsoft Azure. We will deploy both Windows and Linux container hosts using Microsoft Azure–certified images that were tested and approved by Microsoft. After deploying our hosts, we will learn how to manage them and cost-effectively run them.

The main topics of this chapter will be:

- Why use a container host?
- Installing and configuring a Docker container host on Ubuntu Linux
- Installing and configuring a Docker container host on Windows Server 2019
- Using images stored on ACR with a Docker container host
- Controlling container hosts on Azure with automated VM start and stop

Why Use a Container Host?

Now that we have learned how to use ACR, ACI, and AKS to deploy and manage containers on Microsoft Azure, you are probably wondering why we should use a container host.

The first reason is that a container host allows us to build our images in a secure and stable environment.

It also allows us to test and optimize our builds while keeping costs under control.

We can also run multiple environments on one host without affecting the production workload.

The last reason is cost: ACI pricing is based on seconds for every vCPU and bit of memory consumed from the moment you start pulling the image.

If you need to run your test environment on ACI, your costs will skyrocket.

Having a dedicated AKS cluster for testing is good, but it is especially ideal from a cost-effectiveness perspective. A container host allows us to run our development environment in a cost-effective and efficient way, and only when we are ready do we push the final product, which is the image, to ACR and deploy it into ACI. This way, we are only running production workloads on ACI.

We can also run production servers on a Docker host, which was the only method available in the early days of containers.

Installing Docker Container Host on Ubuntu Linux VM

In this section, I will show you how to create a Docker container host that runs on Linux Ubuntu virtual machines in Azure.

As you know, we can use the Azure portal to create the VM, or we can use Azure CLI from Azure Cloud Shell.

My preferred method is to use Azure Cloud Shell because it allows me to reuse the code and be consistent with my deployments.

I will start by showing you how to create a VM using the portal, and then I will show you how to do so using Azure CLI.

Create a Linux Docker Host Using the Portal

To create my Docker container host using the portal, I will create a new virtual machine and set the virtual machine image to Ubuntu server 18.04, which at the time of writing is the latest long-term, stable version.

If you would like to use a newer version, you can select version 19 from the drop-down menu.

The other important section is the VM size; in our case, I recommend that you select Standard D2s V3, which will give a decent performance level.

Figure 5-1 shows the Create a Virtual Machine wizard with the two important details highlighted in red. To complete the process, follow the wizard.

Home > Virtual machines > Create a virtual machine

Create a virtual machine

Basics Disks Networking Management Advanced Tags Review + create

Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image. Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization. Looking for classic VMs? [Create VM from Azure Marketplace](#)

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 subscription

Resource group * ⓘ Apps
[Create new](#)

Instance details

Virtual machine name * ⓘ LinuxDockerHost ✓

Region * ⓘ (US) West US

Availability options ⓘ No infrastructure redundancy required

Image * ⓘ Ubuntu Server 18.04 LTS
[Browse all public and private images](#)

Size * ⓘ Standard D2s v3
2 vcpus, 8 GiB memory
[Change size](#)

Figure 5-1. Create a Virtual Machine wizard

Once the VM is ready, note down the virtual machine external IP address and connect to it using an SSH client.

If you don't want to use an SSH client to connect from your machine to the VM, you can use Azure Cloud Shell, which comes preloaded with the OpenSSL client.

To connect to the VM using SSH, open your command-line tool. If you are using Windows 10, version 1809 and above, OpenSSL is preloaded, and all you need to do is use the following command:

```
ssh username@external_ip_address
```

Create a Linux Docker Host Using Azure CLI and Azure Cloud Shell

To create a Docker container host on a Linux Ubuntu virtual machine using Azure CLI, use the following Bash script, which is loaded with Azure CLI commands.

The Azure CLI commands are based on the `az vm` (virtual machine management) module.

Note Make sure you change the code in the script to match your resource group, location, and admin username and password.

I name the file as follows:

```
5.1.Create_linux_host.sh
```

To run the script on Azure Cloud Shell, simply upload it to Cloud Shell or copy-paste it.

Note To create a new Bash script file on Azure Cloud Shell, use the following command: `touch script.sh`.

```
az vm create --name "LinuxHost" \  
  --resource-group "apps" \  
  --image "UbuntuLTS" \  
  --size Standard_DS2_v2 \  
  --location westus \  
  --admin-username vadmin \  
  --admin-password enterpassord \  
  --generate-ssh-keys
```

Note The default disk size comes with 30 GB hard disk; to change it, add the following file, which will set the disk size to 50 GB:

```
--data-disk-sizes-gb 50
```

Once the VM is ready, all we need to do is get the external IP address and connect to it. You can get the IP address from the portal or use Azure CLI, as you will see in the following subsection.

Get Azure VM Using Azure CLI

For your convenience, I have created a bit of Azure CLI code that lists all the IP addresses of the virtual machines in a resource group called apps.

To run, it changes the resource group name and runs the code. The script is as follows:

```
5.2.Get_IP.sh
az vm list-ip-addresses \
--resource-group apps \
-o table
```

In Figure 5-2 you can see the Get_IP script in action.

```
shimon@Azure:~$ . 5.2.Get_IP.sh
VirtualMachine   PublicIPAddresses   PrivateIPAddresses
-----
LinuxHost3       40.83.145.199       10.0.0.6
shimon@Azure:~$
```

Figure 5-2. Get IP addresses script

All I need to do next is use an SSH client or Azure Cloud Shell to connect to the VM. If you have followed this book thus far, you already know that my preferred method is to use Azure Cloud Shell.

Connect Using SSH from Azure Cloud Shell

To connect to the container host using Azure Cloud Shell, I will log in to it from the following URL:

<https://shell.azure.com>

From the Azure Cloud Shell, I will use the following command to connect:

```
ssh vadmin@external_ip_address
```

If you look at Figure 5-3, you will see the full command in action, but after I logged in, I typed `docker`.

By default, Docker is not installed on the Ubuntu image, and so we need to install it.

Luckily, Ubuntu made it easy for us to get started and listed the commands to install it; however, if you choose to install it using the method shown on the screen in Figure 5-3, it will install an older version of Docker (18.06).

To install the latest Docker version, we will need to run a few commands, which I will show you soon.

Next, I will show you how to install the version that is available in the Ubuntu Linux snap application store.

```

shimon@Azure:~$ ssh vadmin@40.83.145.199
vadmin@40.83.145.199's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 5.0.0-1025-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Thu Nov 14 09:20:33 UTC 2019

System load:  0.0               Processes:            118
Usage of /:   4.1% of 28.9GB    Users logged in:     1
Memory usage: 5%               IP address for eth0: 10.0.0.6
Swap usage:   0%

0 packages can be updated.
0 updates are security updates.

Last login: Thu Nov 14 09:15:34 2019 from 123.2.45.127
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

vadmin@LinuxHost3:~$ docker

Command 'docker' not found, but can be installed with:

sudo snap install docker      # version 18.06.1-ce, or
sudo apt install docker.io

```

Figure 5-3. SSH to Docker container host running on Linux

Install Docker Using the Snap Application Store

To install Docker using the snap Linux app store, we simply need to run the following command:

```
sudo snap install docker
```

The only issue with this installation method is that it will not install the latest version of Docker. In the next subsection, we will install the latest version.

Install the Latest Docker Version

Installing the latest Docker version on our Ubuntu Docker host involves a few extra commands. For your convenience, I have listed all the commands in the following base script:

5.3.Install_Latest_Docker.sh

I will start with the following command, which will remove older Docker versions from your host. If you get a message that nothing was found, it is OK.

```
sudo apt-get remove docker docker-engine docker.io containerd runc
```

Next, I will update the repository:

```
sudo apt-get update
```

The next lines will allow apt to use https:

```
sudo apt-get install \
  apt-transport-https \
  ca-certificates \
  curl \
  gnupg-agent \
  software-properties-common
```

The following command will add all the necessary security keys:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
sudo apt-key add -
```

The following lines will set the repository to install the stable release of Docker:

```
sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) \
stable"
```

The following code will run another update before the installation:

```
sudo apt-get update
```


And finally we will use the following line to install Docker (Version 19.03):

```
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

At this stage, Docker is up and running. To check which version was installed, run the following line:

```
Sudo docker version
```

Install Docker Container Host on Windows Server 2019 VM

In this section, we will learn how to install a Docker container host on a Windows Server 2019 virtual machine.

The good news about Windows Server and Azure is that Microsoft has pre-configured images with Docker already installed. Because it is already installed, we don't need to worry about installing Docker after deploying the VM, as we did with the Linux VM.

Create a Windows Docker Container Using the Portal

Like our Linux Docker host, we can use the portal. In Figure 5-4, you can see that I am using the same wizard as before, but this time I am selecting "Windows Server 2019 Datacenter with Containers."

I can also select "Windows Server 2019 Core," which comes without any GUI.

Home > New > Create a virtual machine

Create a virtual machine

Subscription * ⓘ Azure subscription

Resource group * ⓘ Apps
[Create new](#)

Instance details

Virtual machine name * ⓘ WindowsDocker ✓

Region * ⓘ (US) West US

Availability options ⓘ No infrastructure redundancy required

Image * ⓘ [smalldisk] Windows Server 2019 Datacenter with Containers
[Browse all public and private images](#)

Size * ⓘ **Standard DS2 v2**
2 vcpus, 7 GiB memory
[Change size](#)

Figure 5-4. Create a new virtual machine

To create the VM, all we need to do is follow the wizard and connect to the VM.

Create a Windows Docker Container Using Azure CLI

To create the same Windows Docker host using Azure CLI via Azure Cloud Shell, I have prepared a Bash script with Azure CLI commands. The script is called `5.4.Create_Windows_Host.sh`.

The script will spin a Windows Server 2019 server with Docker installed. The VM instance type is Standard DS2 V2, which comes with 2 vCPU and 8 GB of RAM.

Before running the script, set the admin password (minimum 12 characters) and the admin username (default admin username is vadmin).

By default, the Windows VM comes with 127 GB of disk space.

```
az vm create \
  --resource-group apps \
  --name WindowsDocker \
  --image "MicrosoftWindowsServer:WindowsServer:2019-Datacenter-with-Containers:2019.0.20190603" \
  --size Standard_DS2_v2 \
```

```
--location westus \  
--admin-username vadmin \  
--admin-password typeyoutopassword
```

Note To check the list of all Windows Server images, run the following command in Azure Cloud Shell:

```
az vm image list -f windows --all
```

Connect to the VM

Once the VM is up and running, you can connect to it using a remote desktop client. Simply note down the external IP address and connect to it using the following command from a Windows 10 machine:

```
mstsc /v <Public IP Address>
```

In Figure 5-5, you can see the remote desktop connection client.

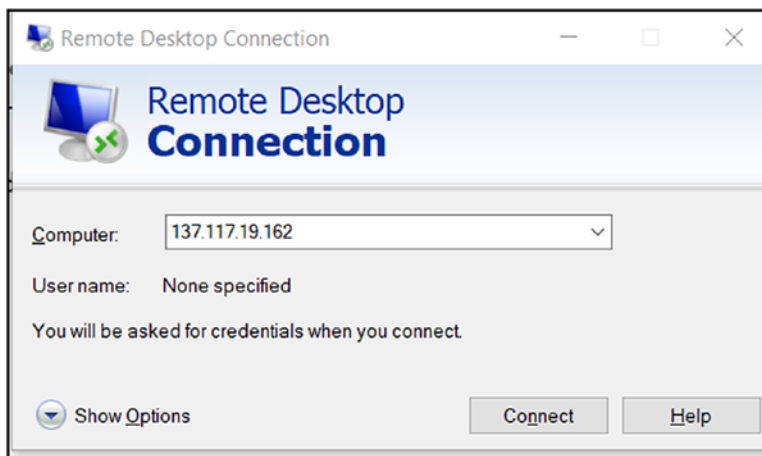


Figure 5-5. Remote desktop

Update Docker on Windows Server 2019

If you want to update Docker on your Windows Server 2019, you can perform the following steps.

First, type the following command, which will check the currently installed version:

```
docker version
```

To start the update process, run the following command:

```
Install-Package -Name Docker -ProviderName DockerMSFTProvider -Update -Force
```

Next, restart the Docker service:

```
Restart-service docker
```

After restarting the service, your Windows Server should run the latest Docker Enterprise version. In Figure 5-6, you can see all the commands and the before and after Docker version.

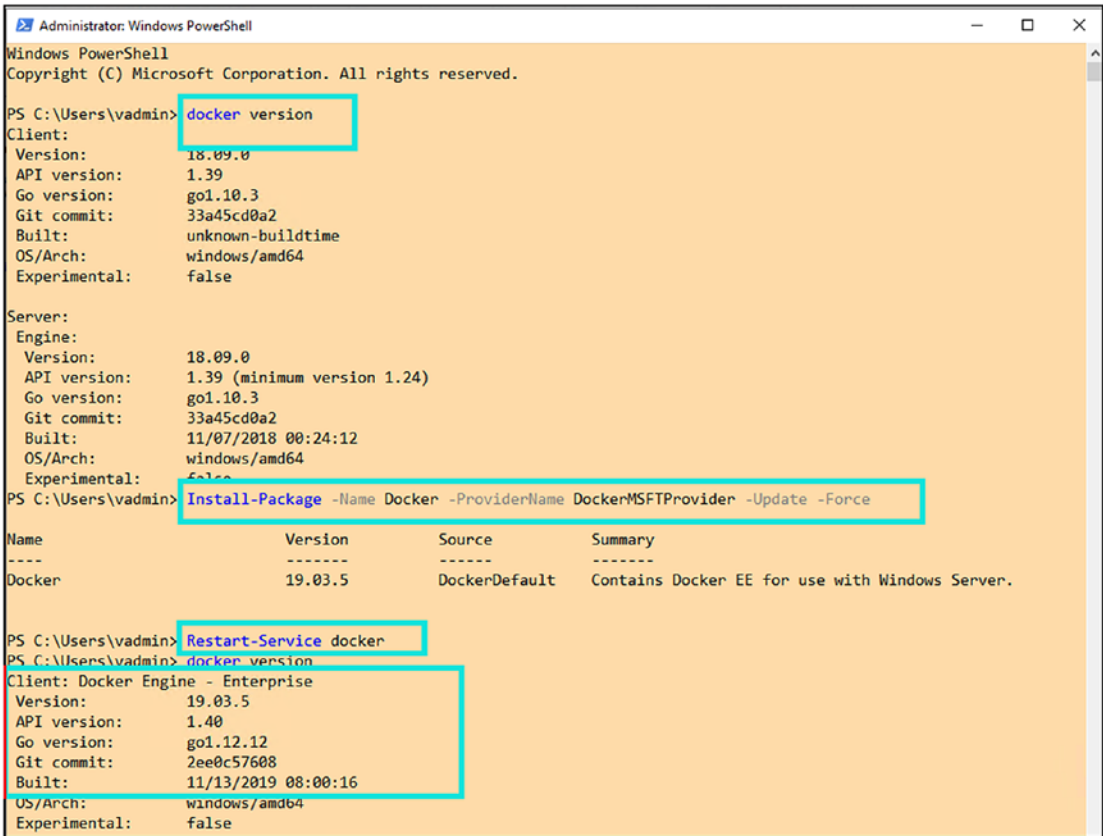


Figure 5-6. Update Docker on Windows Server

Note By default, Windows Server runs Docker Enterprise edition.

Manage Container Hosts on Azure

Now that our container host (Linux or Windows) is running on Azure, we will learn how to both manage it and make sure it is cost effective.

Azure offers many features that allow us to better manage the performance of our resources and at the same time make sure costs are under control.

In this section, I will focus on two options that allow us to better manage the cost of our virtual machines in Azure. Azure billing is based on seconds, and every time your VM runs when it doesn't need to, you are wasting funds you could have allocated to other resources.

Auto-Shutdown Container Host VM

One of the great features of Azure virtual machines is that we can configure them to auto-start and auto-shutdown.

Currently, Azure offers two features that allow us to control virtual machines' costs, which are:

Auto-shutdown — This feature configures the VM to automatically shut down at a specific time every day. However, it doesn't turn the machine on.

This feature is great for a VM that is not used every day for a specific time.

If you use your VM a few hours a day, this feature is good because it will automatically turn off at a specific time every day, and you can turn it on whenever you need it.

Start/Stop VM — This solution is for development and test environments where the VMs are not used after hours.

If your VM is used every day during business hours or until a specific time, this solution is great for you because you can automatically configure the VM to stop and start.

This is the most cost-effective solution, and it is recommended you use it when possible because of the huge savings it provides over the long term.

The following example outlines the potential savings of the Start/Stop VM solution when using a Standard DS2 V2 VM.

By default, if you decide to keep your VM running for twenty-four hours a day, the cost will be as follows:

730 hours per month costs \$102 USD

If you decide to run your VM twelve hours a day (8:00 AM to 20:00 PM) the cost will be as follows:

365 hours per month costs \$51 USD

If you decide to run your VM eighteen hours a day (6:00 AM to 12:00 AM) the cost will be as follows:

547 hours per month costs \$76 USD

As you can see from the preceding examples, costs can be reduced significantly if you keep the VM running for just twelve or eighteen hours a day.

By turning the VM off for six hours a day, seven days a week, the VM cost is reduced by 25 percent.

Configure Auto-Shutdown

I will start by showing you how to configure auto-shutdown of an Azure Linux Docker host virtual machine.

Auto-shutdown is available from the portal; to access it perform the following steps:

Open the Azure portal, then open virtual machines.

Click on the VM for which you would like to configure auto-shutdown.

From the VM left-hand menu pane, click on “Auto-shutdown,” as shown in Figure 5-7.

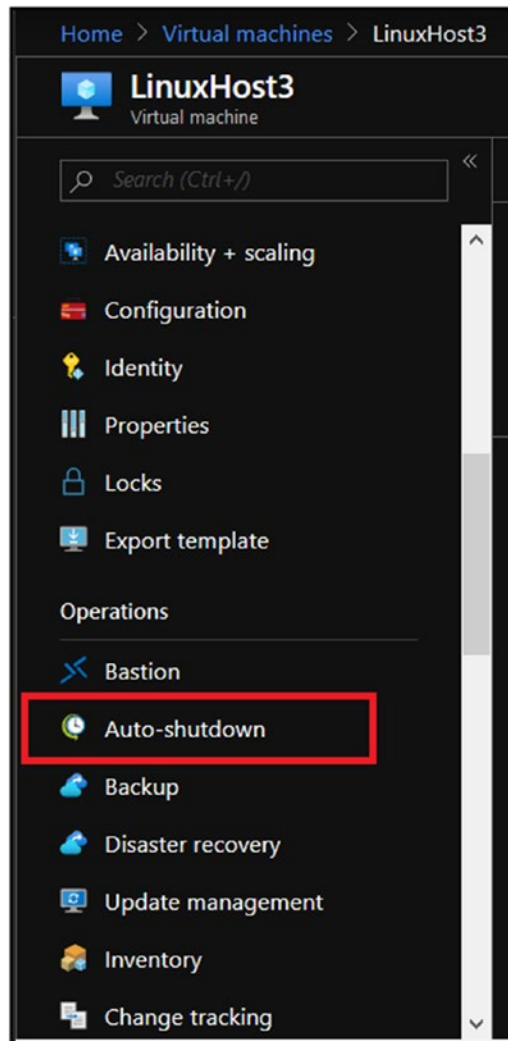


Figure 5-7. *Auto-shutdown*

From the Auto-shutdown configuration page, all you need to do is enable the feature, select shutdown time, and configure notification (email or webhook).

When you are ready, click “Save,” and you are done.

Figure 5-8 shows the auto-shutdown configuration.

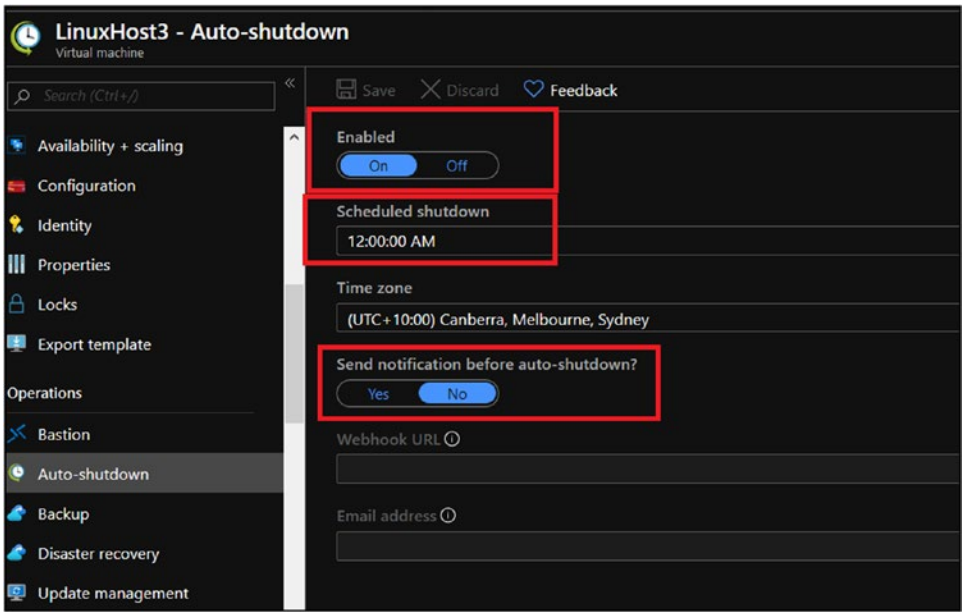


Figure 5-8. Auto-shutdown configuration

At this stage, you are done, and the VM will shut down automatically (if it is on at this time).

Just remember that you will need to turn it on when you need to use it because this feature only shuts the VM down.

Configure Start/Stop Container Host VM

The previous option was straightforward and easy to implement; however, configuring a VM to auto-shutdown and restart at a specific time is a bit more complicated.

To accomplish this task, we will use the following Azure solutions:

- Azure Automation account
- Azure log analytics
- Azure functions

As you can see, configuring this feature is not simple at all; however, by using the following process you will get your VM configured in just ten minutes.

To get started, open the Azure portal.

In the search bar, search for Azure automation, as shown in Figure 5-9.

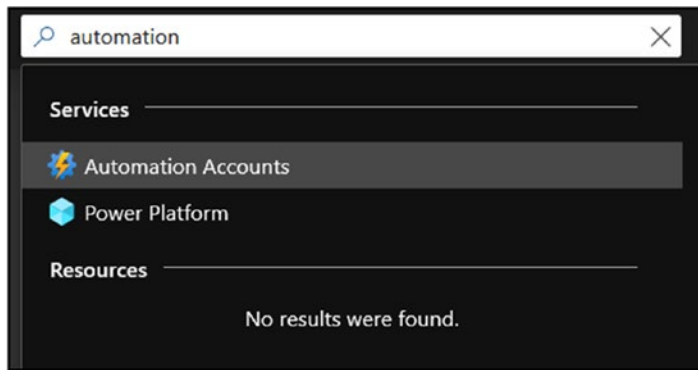


Figure 5-9. Automation accounts page

From the search bar, click on “Automation Accounts.” In the Automation Accounts page, click on “Add.” Figure 5-10 shows the Automation Accounts page.

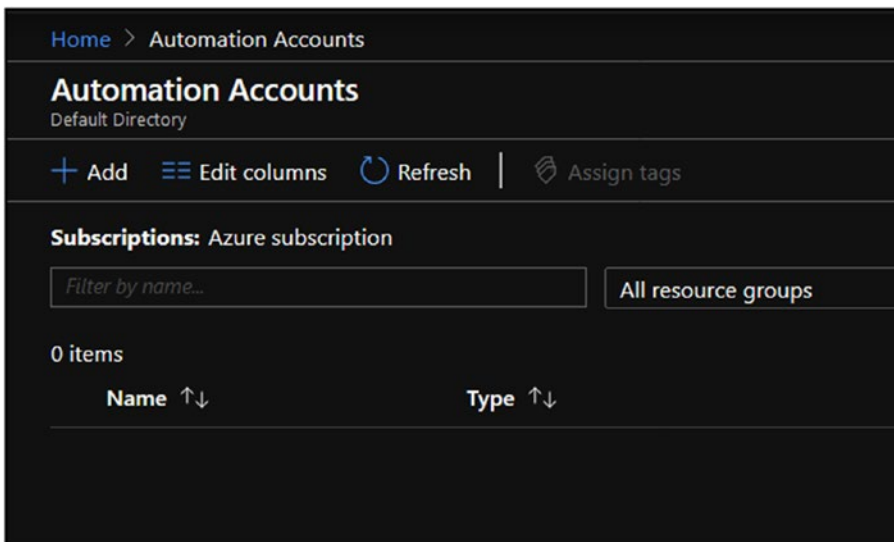


Figure 5-10. Automation Accounts page

We will go ahead and create an Azure automation account, which will run our solution. In the Add Automation Account page, fill in the following details:

1. Name the automation account.
2. Select the same subscription in which the VMs you need to Start/ Stop are located.

3. Select the same resource group in which the VMs are located.
4. Leave the default selection to create an Azure Run As account.
5. In the location drop-down, please select a location from the following supported regions: Australia Southeast, Canada Central, Central India, East US, Japan East, Southeast Asia, UK South, West Europe, and West US 2.

Figure 5-11 shows the Add Automation Account page.

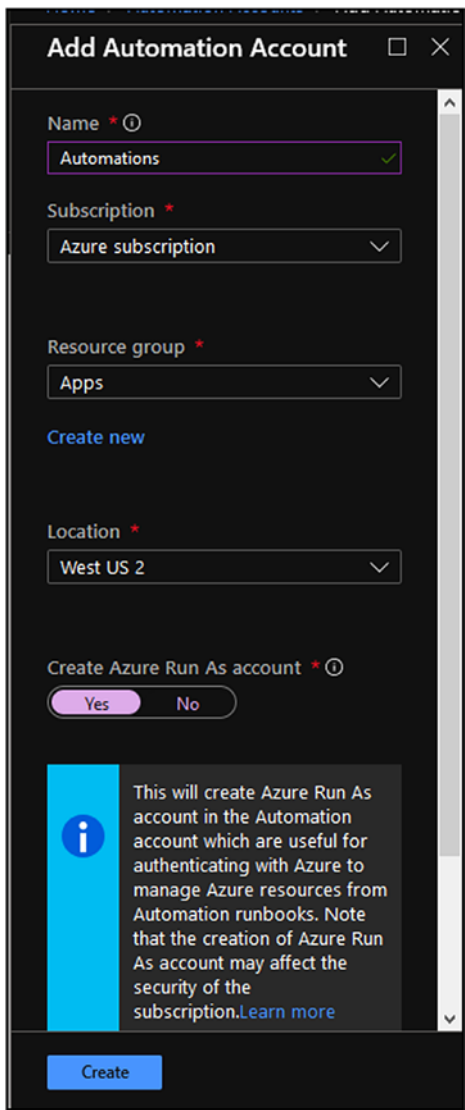


Figure 5-11. Add Automation Account page

Azure will go ahead and create the automation account, which will take a few minutes.

The next step will be adding the Start/Stop VM solution to the automation account.

Note Because of all the related services needed for Azure automation, the creation process can take up to ten minutes.

When the automation account is ready, you will see it under Automation Accounts, as shown in Figure 5-12.

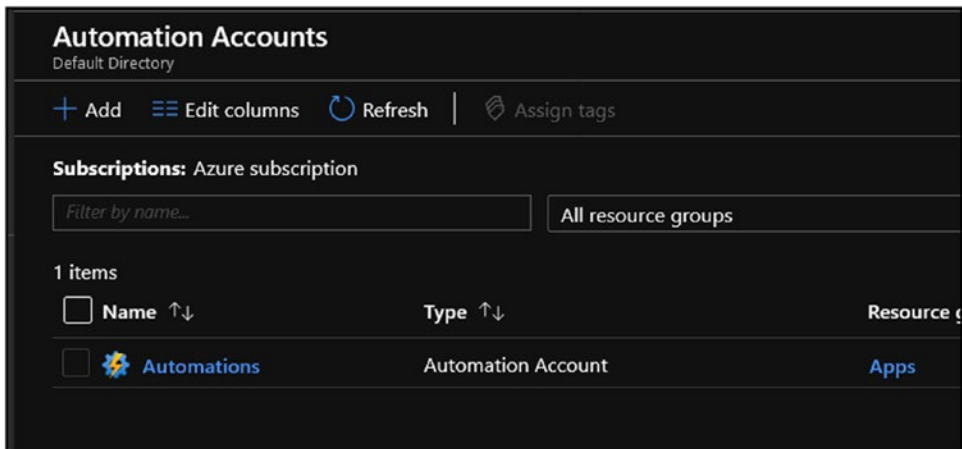


Figure 5-12. Automations accounts

To continue with our configuration, click on the newly created account. To enable the Start/Stop VM solution, use the following steps.

In the Automations left-hand menu pane, click on “Start/Stop VM.”

Note The Start/Stop VM option is located under the Related Resources menu.

Figure 5-13 shows the Start/Stop VM option.

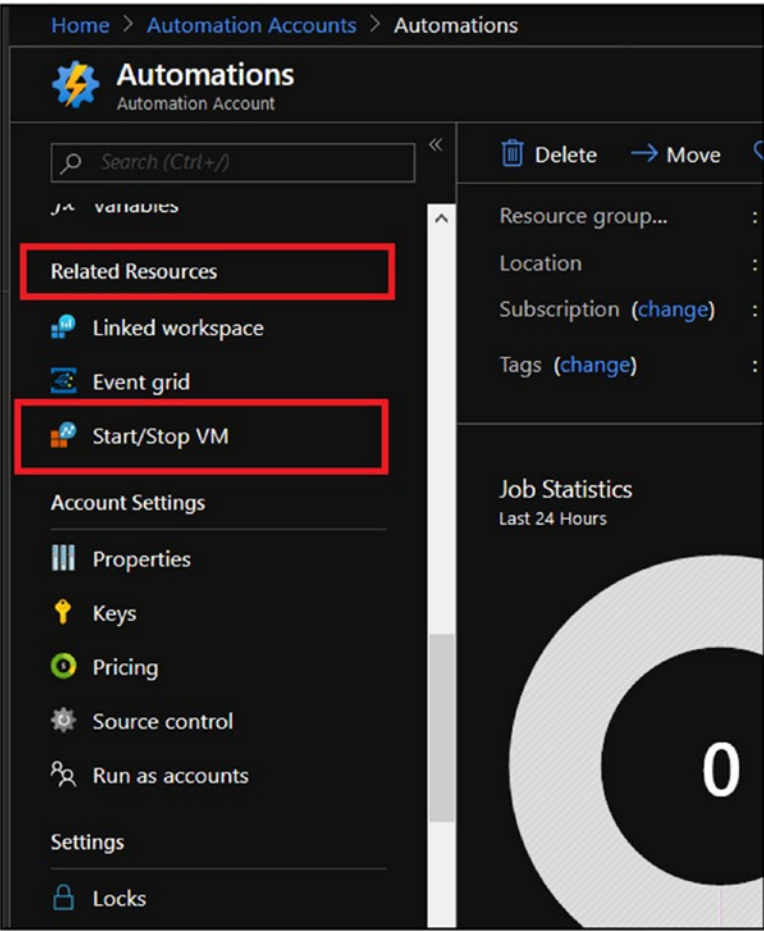


Figure 5-13. Start/Stop VM

From the Start/Stop VM page, we need to click on “Learn more about and enable the solution.” See Figure 5-14.

Note To manage an existing Start/Stop VM solution, click on “Manage the solution.”

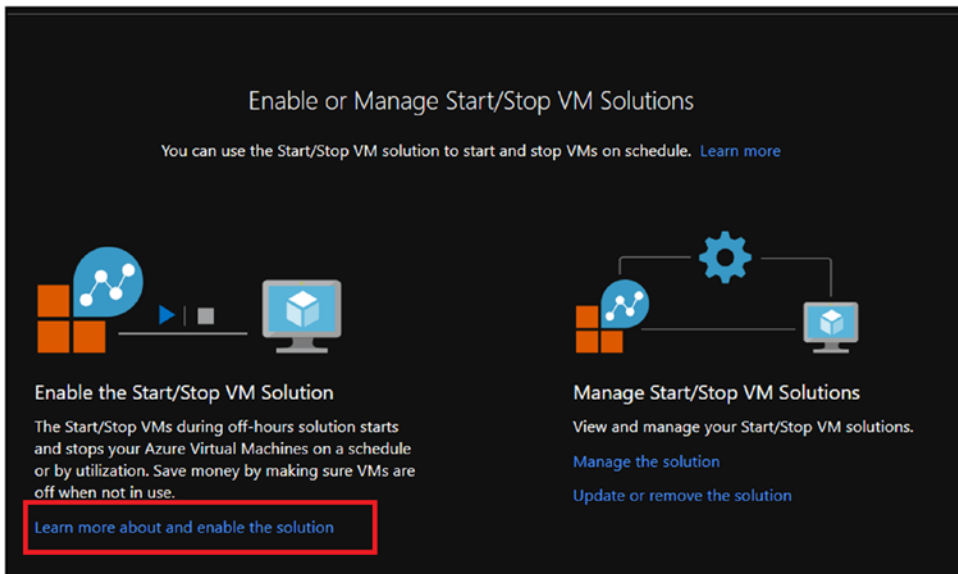


Figure 5-14. Enable Start/Stop VM

From the Start/Stop VMs During Off-hours page, click on the “Create” button to start the configuration process.

Figure 5-15 shows the Start/Stop VMs During Off-hours page.

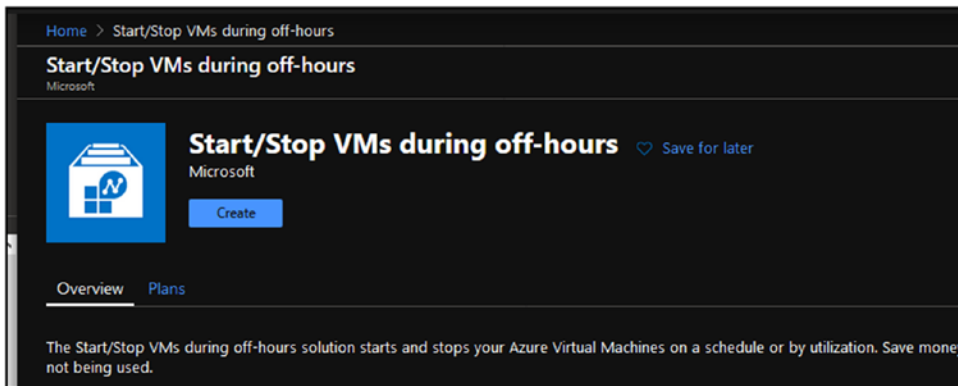


Figure 5-15. The Start/Stop VMs During Off-hours page

After clicking on the “Create” button, we will be redirected to the Add Solution page.

To create the solution, Azure will ask us to create a log analytics workspace to manage the logs, so we also need to link the automation account we created before and configure the schedule, which we will need to change after.

Figure 5-16 shows the Add Solution page.

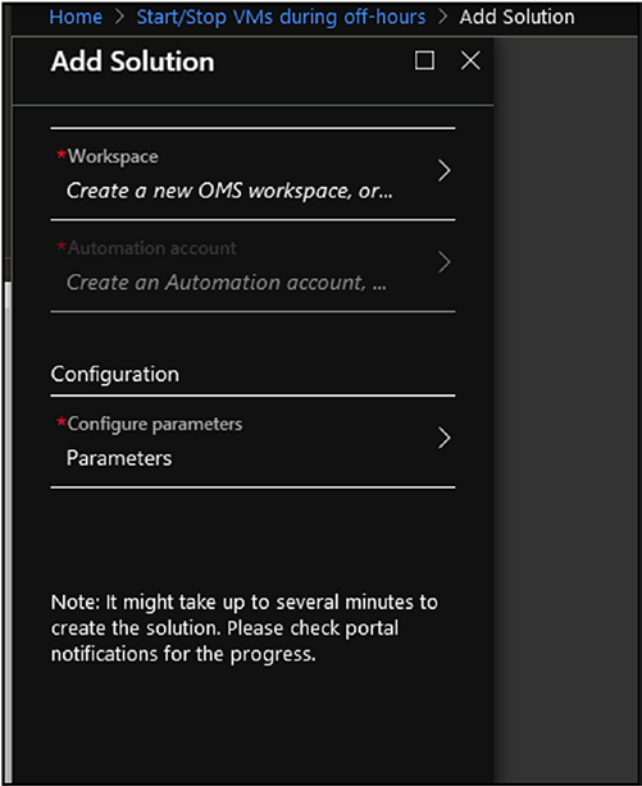


Figure 5-16. Add Solution page

To get started, click on “Workspace” and then click on “Create New Workspace,” as shown in Figure 5-17.

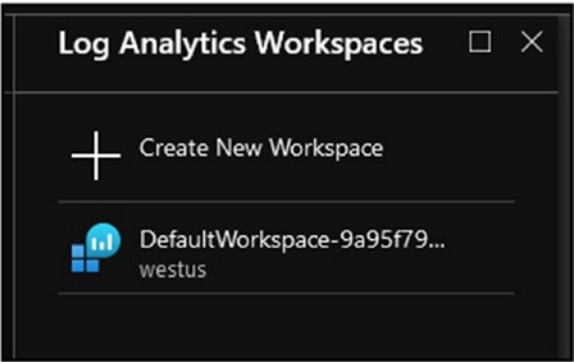


Figure 5-17. Create a new workspace

In the Create a New Workspace creation menu, fill in the details and make sure you use the same location and resource group as the VM or VMs you would like to auto-shutdown.

Figure 5-18 shows how to create a log analytics workspace. Go ahead and fill in the details.

Log Analytics workspace □ ×
Create new or link existing workspace

☒ Create New ☐ Link Existing

Log Analytics Workspace * ⓘ
automationworkspace0001 ✓

Subscription *
Azure subscription ▼

Resource group *
Apps ▼
[Create new](#)

Location *
West US 2 ▼

*Pricing tier
Pay-as-you-go (Per GB 2018) >

Figure 5-18. Create a log analytics workspace

After creating the log analytics workspace, click on “Automation account” and select the automation account we created before.

Figure 5-19 shows my automation account.

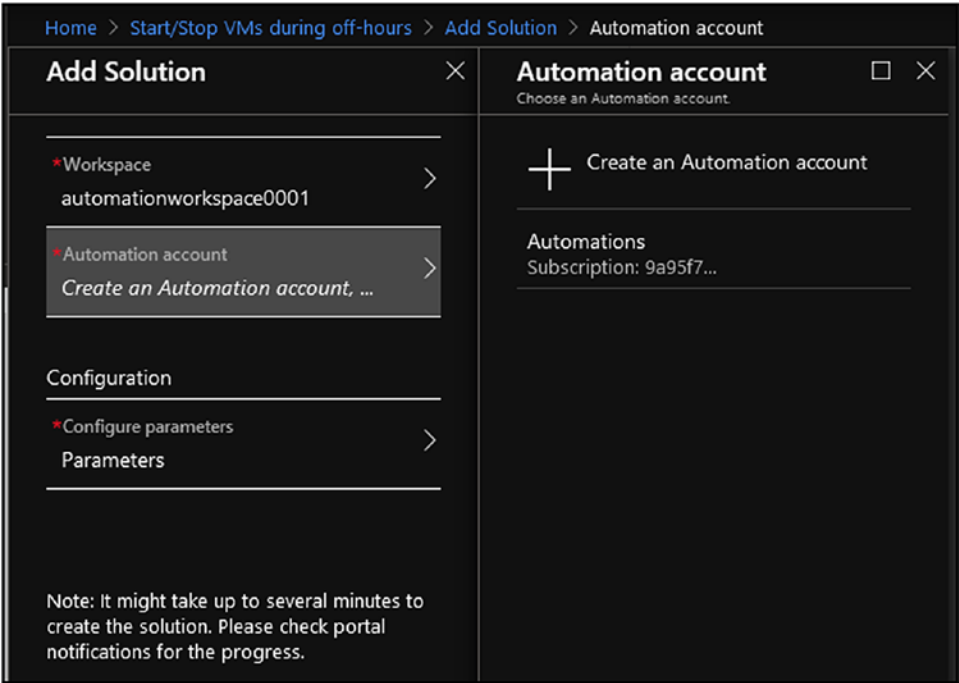


Figure 5-19. *Select the automation account*

In this last section, we need to configure the schedule and select the resource group affected by the solution.

I will go ahead and type the name of my resource group where I have my VMs; in my case, it’s called apps.

Note By default, all the virtual machines inside the resource group will be turned off. If you don’t want all the VMs inside the resource group to shut down, add them to the VM exclude list.

Figure 5-20 shows the Parameters menu.

Parameters

Vm runbook

Target ResourceGroup Names (string) ⓘ
apps

VM Exclude List (string) ⓘ
none

Schedule

Daily Start Time (datetime) * ⓘ
11/22/2019 6:00:00 AM

Daily Stop Time (datetime) * ⓘ
11/22/2019 6:00:00 PM

Email functionality

Receive Email Notifications (string) ⓘ
No

Email Addresses (string) ⓘ

OK

Figure 5-20. Parameters menu

Once we are ready and have configured all settings, we click “OK.” To finish the configuration, we review the settings and click “Create,” as shown in Figure 5-21.

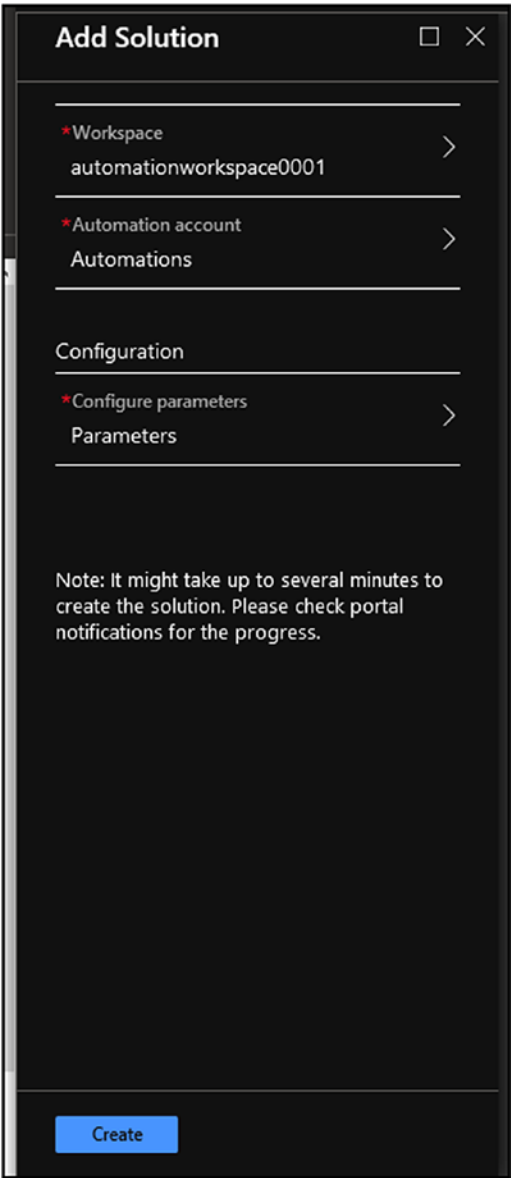


Figure 5-21. *Create a solution*

At this stage, Azure will go ahead and deploy the solution, which will take five to ten minutes.

After the solution has finished, we will need to configure the schedule again and set the time zone. Azure configured the parameters using UTC, and the values are acting as placeholders.

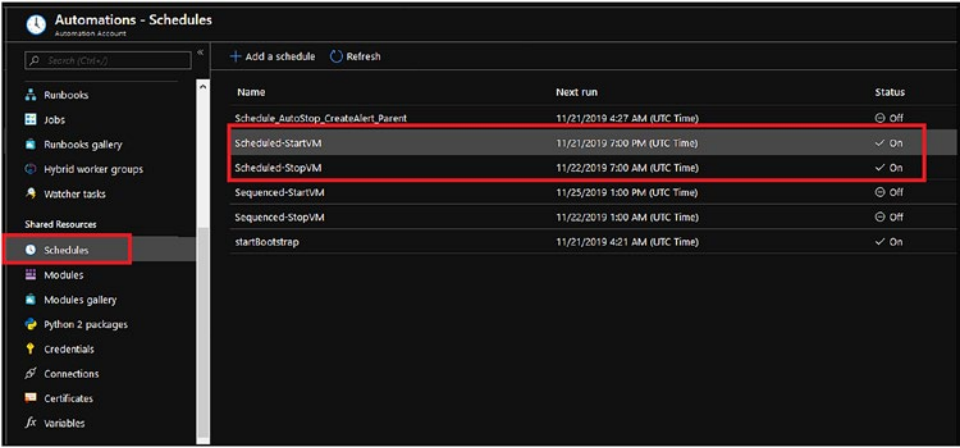
Change Schedule

To change the default schedule, which is set to UTC, in the Azure Automation Account left-hand menu, we click on “Schedules,” as shown in Figure 5-22.

In the Schedules page, we will see the two schedules we set in the Parameters page, and they are as follows:

Schedule-StartVM

Schedule-StopVM



Name	Next run	Status
Schedule_AutoStop_CreateAlert_Parent	11/21/2019 4:27 AM (UTC Time)	Off
Scheduled-StartVM	11/21/2019 7:00 PM (UTC Time)	On
Scheduled-StopVM	11/22/2019 7:00 AM (UTC Time)	On
Sequenced-StartVM	11/25/2019 1:00 PM (UTC Time)	Off
Sequenced-StopVM	11/22/2019 1:00 AM (UTC Time)	Off
startBootstrap	11/21/2019 4:21 AM (UTC Time)	On

Figure 5-22. Schedules

We should go ahead and change them to the correct time zone and the time that we need. In my case, I will set the start time to 6:00 AM and the stop time to 6:00 PM.

Figure 5-23 shows the Edit Schedule page.

Enabled *

Yes

No

Description

Schedule for ScheduledStartStop runbook - stop action

Starts *

11/22/2019

6:00 PM

Time zone

Australia - Eastern Australia Time (Sydney)

Recurrence

Once

Recurring

Recurrence Details

Recur every *

1

Day

Set expiration

Yes

No

Expires

Never

Figure 5-23. *Edit Schedule page*

Having edited the start and stop schedules, we can see both are now set correctly, as shown in [Figure 5-24](#).

Name	Next run	Status
Schedule_AutoStop_CreateAlert_Parent	11/21/2019 4:27 AM (UTC Time)	⊖ Off
Scheduled-StartVM	11/22/2019 6:00 AM (AET)	✓ On
Scheduled-StopVM	11/22/2019 6:00 PM (AET)	✓ On
Sequenced-StartVM	11/25/2019 1:00 PM (UTC Time)	⊖ Off
Sequenced-StopVM	11/22/2019 1:00 AM (UTC Time)	⊖ Off

Figure 5-24. Schedules after the change

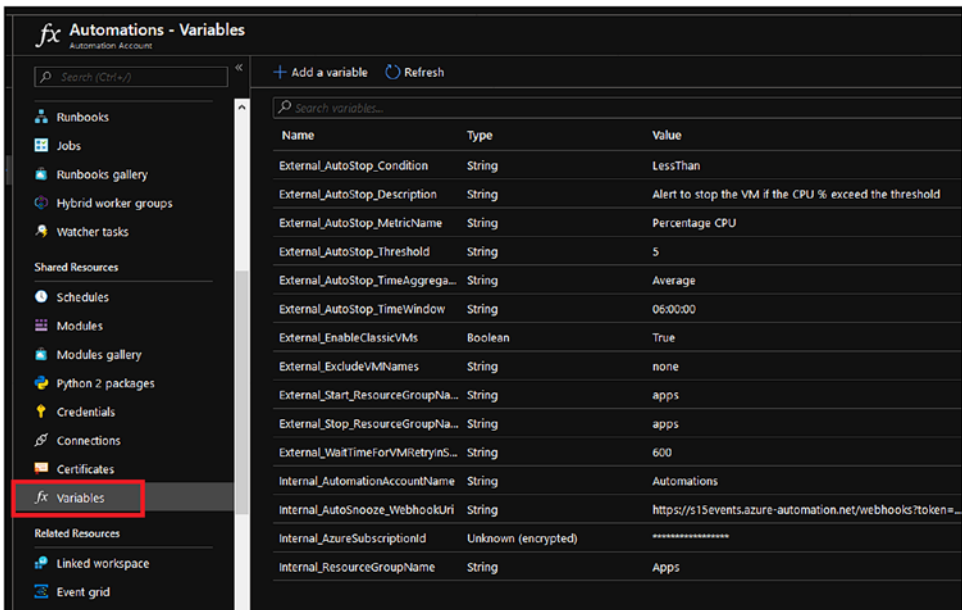
The last setting I will show you is how to exclude a VM from the schedule.

By default, the Start/Stop solution will turn off all the VMs in the specified resource group, which is not always the best idea. You might have one VM in the resource group that needs to be on twenty-four hours. In a case like this, we can exclude VMs from the solution.

Exclude Virtual Machines

To exclude single or multiple virtual machines from auto-shutdown, we can edit the solution parameters using the following steps.

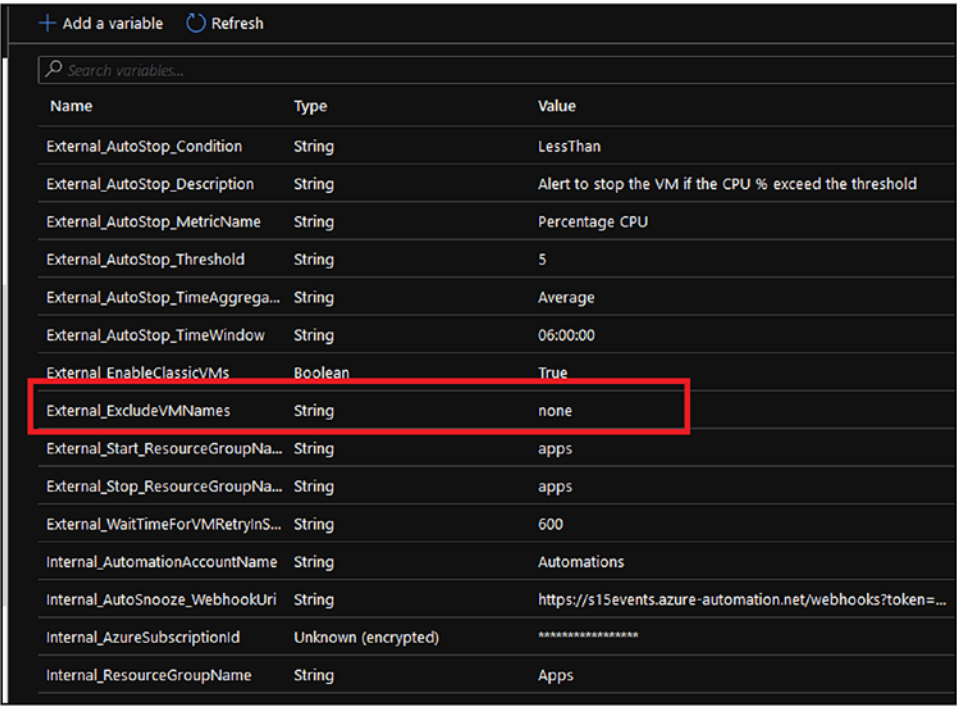
Open the automation account. In the left-hand menu, click on “Variables,” as shown in Figure 5-25.



fx Automations - Variables		
Automation Account		
Search (Ctrl+J) + Add a variable Refresh		
Search variables...		
Name	Type	Value
External_AutoStop_Condition	String	LessThan
External_AutoStop_Description	String	Alert to stop the VM if the CPU % exceed the threshold
External_AutoStop_MetricName	String	Percentage CPU
External_AutoStop_Threshold	String	5
External_AutoStop_TimeAggrega...	String	Average
External_AutoStop_TimeWindow	String	06:00:00
External_EnableClassicVMs	Boolean	True
External_ExcludeVMNames	String	none
External_Start_ResourceGroupNa...	String	apps
External_Stop_ResourceGroupNa...	String	apps
External_WaitTimeForVMretryinS...	String	600
Internal_AutomationAccountName	String	Automations
Internal_AutoSnooze_WebhookUri	String	https://s1sevents.azure-automation.net/webhooks/token=...
Internal_AzureSubscriptionId	Unknown (encrypted)	*****
Internal_ResourceGroupName	String	Apps

Figure 5-25. Variables

From the Variables page, click on “External_ExcludeVMnames,” as shown in Figure 5-26.



The screenshot shows the 'Variables' page in the Azure portal. At the top, there are buttons for '+ Add a variable' and 'Refresh'. Below these is a search bar labeled 'Search variables...'. The main area contains a table of variables. The variable 'External_ExcludeVMNames' is highlighted with a red rectangle. The table has three columns: 'Name', 'Type', and 'Value'.

Name	Type	Value
External_AutoStop_Condition	String	LessThan
External_AutoStop_Description	String	Alert to stop the VM if the CPU % exceed the threshold
External_AutoStop_MetricName	String	Percentage CPU
External_AutoStop_Threshold	String	5
External_AutoStop_TimeAggrega...	String	Average
External_AutoStop_TimeWindow	String	06:00:00
External_EnableClassicVMs	Boolean	True
External_ExcludeVMNames	String	none
External_Start_ResourceGroupNa...	String	apps
External_Stop_ResourceGroupNa...	String	apps
External_WaitTimeForVMRetryInS...	String	600
Internal_AutomationAccountName	String	Automations
Internal_AutoSnooze_WebhookUri	String	https://s15events.azure-automation.net/webhooks?token=...
Internal_AzureSubscriptionId	Unknown (encrypted)	*****
Internal_ResourceGroupName	String	Apps

Figure 5-26. External_ExcludeVMNames

To exclude VMs called VM01 and VM02, I will use the format that is shown in Figure 5-27.

Save Discard Delete

Name
External_ExcludeVMNames

Last modified
11/21/2019, 3:11 PM

Description
Excluded VMs as comma separated list: vm1,vm2,vm3

Encrypted
No

Type ⓘ
String

Value
vm01, vm02 ✓

Figure 5-27. *Exclude VMs*

To exclude the VMs, all I need to do now is save the configuration.

Monitor Jobs

To monitor completed jobs in our automation account we can use the “Jobs” link in the left-hand menu. This screen is useful for troubleshooting and keeping track of completed jobs.

Figure 5-28 shows the completed automated jobs in my automation account.

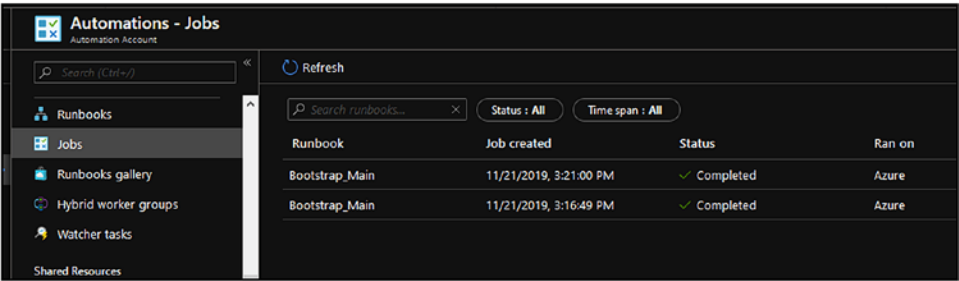


Figure 5-28. Jobs

Summary

In this chapter, we learned a lot about how to deeply Docker container hosts running on Windows and Linux virtual machines. We used both the portal and Azure CLI using Azure Cloud Shell to deploy the VMs.

For the Linux VM, we used the Azure Cloud Shell built-in OpenSSH client to connect to it. In the last section of this chapter, we deployed the Start/Stop VM automation solution, which helps us control a VM’s running costs by automatically turning it on and off.

CHAPTER 6

Secure Your Microsoft Azure Containers

In this chapter, we will focus on the security side of things with our Azure tenant and see how to use some of the built-in and free Azure services to help us protect our resources inside Azure. Security on the Microsoft Azure platform is based on multiple layers, where each layer comes with some built-in security features that are provided by Azure and cannot be changed. Some features can be deployed by the tenant owner as a paid service.

In this chapter, we will cover the following topics:

- Azure Active Directory reporting
- Managing Azure AD with Azure CLI
- Using a Bastion host to manage virtual machines
- Using Azure Security Center and Secure Score
- Learning about Azure Firewall and Network Security Groups (NSG)

The good thing about Azure is that the starting point of each service security state is high because Azure comes with built-in security controls. The security layers in Azure are as follows:

- Users and Identity (Azure AD)
- Network security (subnets, ports, etc.)
- Infrastructure (physical)
- Virtual machines (compute)
- Data security (disk encryption, database security)
- Application security (PaaS)

Protect and Manage Accounts and Hosts on Microsoft Azure Using Azure AD and Bastion

At the heart of our Microsoft Azure tenant, we have our Identity infrastructure, which is powered by Azure Active Directory. Azure Active Directory manages the User layer, also known as the Identity layer, where we control who can access what.

Azure Active Directory provides the following services:

- Enterprise Application — Single sign-on (SSO)
- App Registration — Use Azure AD for custom applications
- Conditional Access — Control access to Microsoft Azure using rules like location, browser, and more
- Multi-factor authentication
- Device registration
- Self-service password management for users
- The Azure AD platform is always evolving, and I recommend you review the product page for the latest services.

In this section, we will focus on identity management and reporting.

Azure Active Directory Reporting

By default, Azure Active Directory reporting is not available with a basic subscription, and so to tap into login activities reporting we would need to upgrade our license level to Azure Active Directory P1.

In a development environment, you might not consider upgrading your license level; however, in a production environment with multiple administrators and users, you might have to enable reporting for compliance reasons.

Sign-ins Information

To access Azure Active Directory reporting, log in to the Azure portal. Search for Azure Active Directory.

From the Azure Active Directory pane, click on “Sign-ins” in the Monitoring section. Figure 6-1 shows this link.

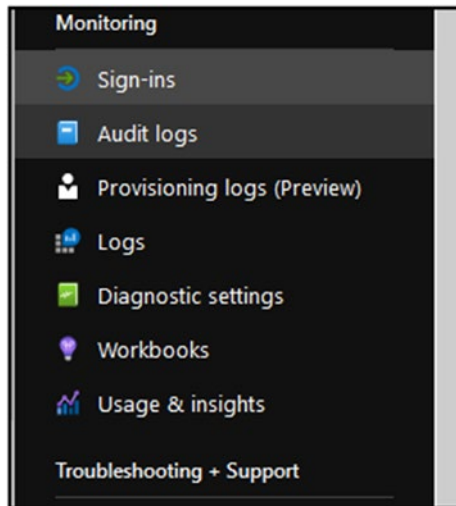


Figure 6-1. *Sign-ins*

If you don't have an Azure AD P1 license, you will see the error message shown in Figure 6-2.

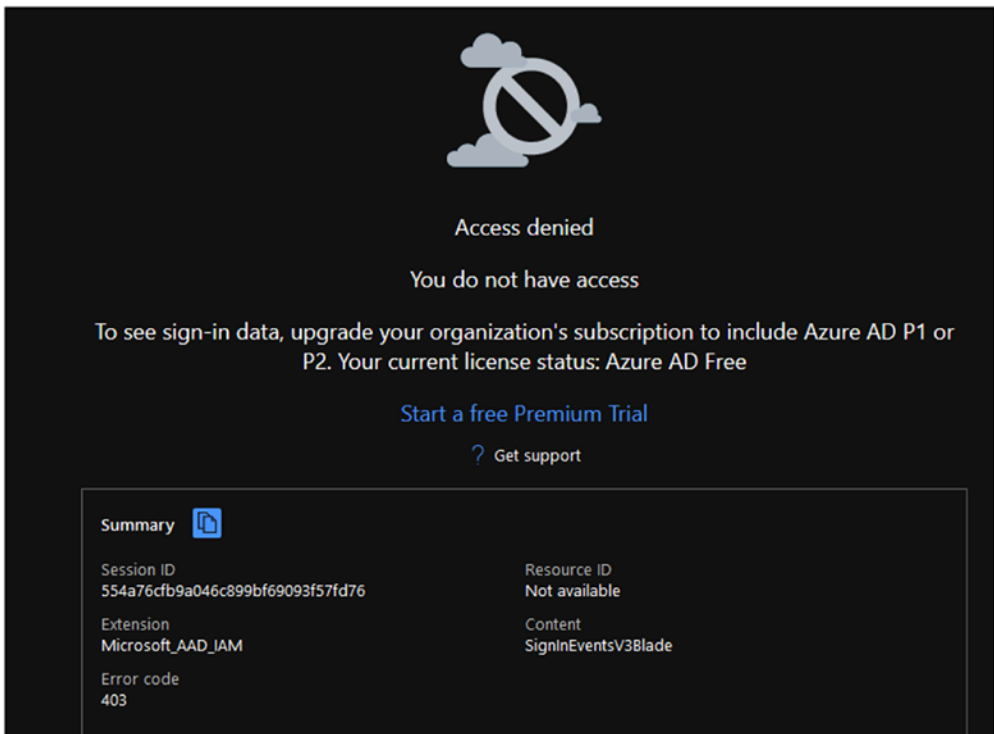


Figure 6-2. *No Azure AD Premium licensing*

To start a few Azure AD premium trials, click on the “Start a free Premium Trial” link. For this section, I have licensed my account for Azure AD P1.

After upgrading to P1, Azure AD will start logging all the user and application sign-in activities and will show detailed information.

Note Azure AD keeps sign-in information for thirty days.

In Figure 6-3, you can see how the Sign-ins page looks when the Azure AD P1 license is enabled.

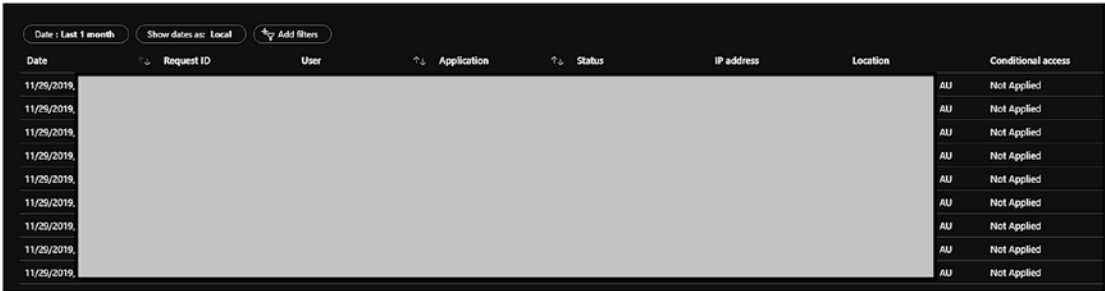


Figure 6-3. Azure AD Sign-ins page

If I click on each sign-in event, Azure will display detailed information about the login activity, which includes the following:

- User information — User ID, date and time, Azure resource
- Location — City and country and IP address
- Device info — Browser version and OS
- Authentication — MFA details, if used
- Conditional access - If used and which policy

Figure 6-4 shows the Login Details section.

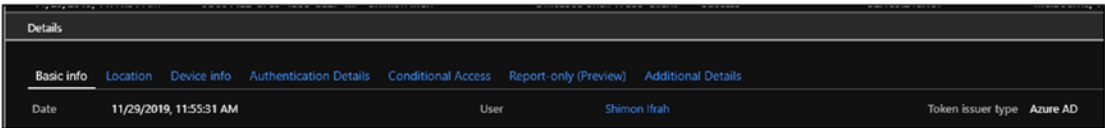


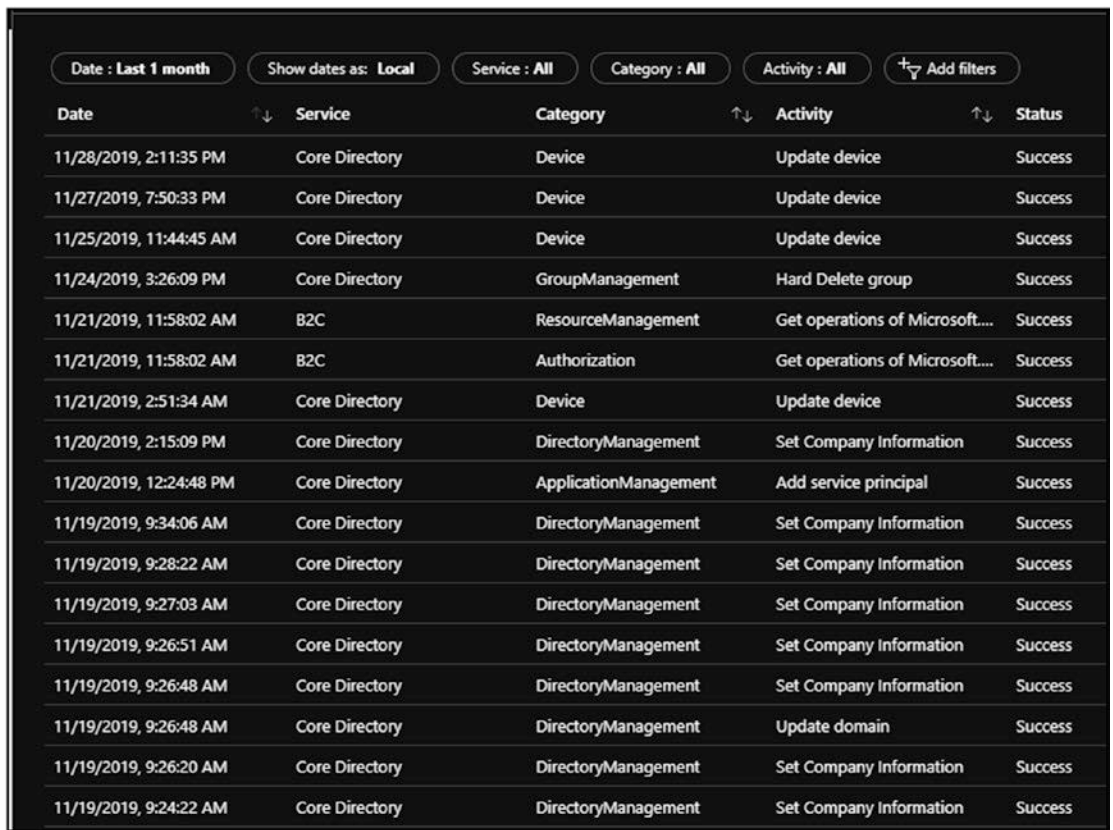
Figure 6-4. Login Details section

Audit Logs

Another great feature in Azure Active Directory is the audit log. Audit logs give us visibility into all changes made in Active Directory using a GUI-based interface.

For example, audit logs can show step-by-step changes administrators make in Active Directory. The good news about audit logs is that the feature is free and is provided at no added cost.

Figure 6-5 shows the audit logs and all the information provided by them. Audit logs also show changes made by Azure and not just by administrators.



Date	Service	Category	Activity	Status
11/28/2019, 2:11:35 PM	Core Directory	Device	Update device	Success
11/27/2019, 7:50:33 PM	Core Directory	Device	Update device	Success
11/25/2019, 11:44:45 AM	Core Directory	Device	Update device	Success
11/24/2019, 3:26:09 PM	Core Directory	GroupManagement	Hard Delete group	Success
11/21/2019, 11:58:02 AM	B2C	ResourceManagement	Get operations of Microsoft...	Success
11/21/2019, 11:58:02 AM	B2C	Authorization	Get operations of Microsoft...	Success
11/21/2019, 2:51:34 AM	Core Directory	Device	Update device	Success
11/20/2019, 2:15:09 PM	Core Directory	DirectoryManagement	Set Company Information	Success
11/20/2019, 12:24:48 PM	Core Directory	ApplicationManagement	Add service principal	Success
11/19/2019, 9:34:06 AM	Core Directory	DirectoryManagement	Set Company Information	Success
11/19/2019, 9:28:22 AM	Core Directory	DirectoryManagement	Set Company Information	Success
11/19/2019, 9:27:03 AM	Core Directory	DirectoryManagement	Set Company Information	Success
11/19/2019, 9:26:51 AM	Core Directory	DirectoryManagement	Set Company Information	Success
11/19/2019, 9:26:48 AM	Core Directory	DirectoryManagement	Set Company Information	Success
11/19/2019, 9:26:48 AM	Core Directory	DirectoryManagement	Update domain	Success
11/19/2019, 9:26:20 AM	Core Directory	DirectoryManagement	Set Company Information	Success
11/19/2019, 9:24:22 AM	Core Directory	DirectoryManagement	Set Company Information	Success

Figure 6-5. Audit logs

If you click on each item, you will be able to see detailed information about the event.

You can also use Azure CLI to manage Azure AD.

Azure CLI az ad

Up until now, I have shown you how to use the Microsoft Azure portal to access some of the Azure AD features. As with most Microsoft Azure services, we can also use the Azure CLI Active Directory module to manage users, groups, and applications.

To manage Azure Active Directory with Azure CLI, we use the `az ad` command syntax.

If you are using Azure Cloud Shell, log in to the shell from the browser using the following address:

<https://shell.azure.com>

To view all of your users, use the following command:

```
az ad user list -o=table
```

To view a list of all apps, use the following command:

```
az ad app list
```

To view all of the Azure AD user groups, use the following command:

```
az ad group list
```

We can also use Azure CLI to create users; in the following example, I will create a user called dbadministrator using Azure CLI.

```
az ad user create --display-name dbadministrator --password  
aDiPfjE2kVyy --user-principal-name dbadministrator@shimoni4outlookcom.  
onmicrosoft.com --force-change-password-next-login {true}
```

We can also delete the user using Azure CLI using the following command:

```
az ad user delete --id f7ca7b58-63b7-4766-b55e-2a59b1c14b29
```

Where the ID of the dbadministrator is f7ca7b58-63b7-4766-b55e-2a59b1c14b29.

As shown in the examples, Azure CLI allows us to programmatically create and manage users, groups, and applications using commands.

I recommend you review the preceding commands and get familiar with the `az ad` Azure CLI module.

Azure Bastion

In this section, we will cover Azure Bastion, a new managed platform as a service (PaaS) remote desktop and SSH secure connectivity service for virtual machines hosted on Azure.

If you remember from Chapter 5, we created Docker container hosts that ran on Windows and on Linux. By default, remote connectivity using remote desktop protocol (Windows) and SSH (Linux) was used to connect to them. The exposed protocols are available via the internet, and authentication was done using a username and password.

The issue with this remote connectivity solution is that anyone could try to break into the virtual machines if they found the IP address. Even though no one can log in without knowing the login details, the servers are exposed directly over the internet and are vulnerable to hackers.

Azure Bastion addresses this issue by providing a managed service that allows us to manage servers directly from the portal using a browser and then connect to the VMs using RDP or SSH.

With Bastion, our VMs don't need a public IP address to be accessible for management.

Azure Bastion Pricing

Similar to Azure Cloud Shell, pricing is based on hourly usage, and you are only charged when connected to Azure VMs.

Managing a VM in the West US region with Bastion will cost \$0.095 per hour. The first 5 GB of outbound is free; however, the next 5 GB in the Azure Zone 1 region will cost \$0.087 per GB.

In a Zone 2 region, you will pay \$0.12 per GB.

Zone 1 regions are West Europe, East US, South Central US, and West US. Zone 2 regions are Australia East and Japan East.

Bastion in Action

Before we get into the Bastion configuration, I would like you to have a look at the Windows Server Docker host virtual machine I created in the previous chapter.

As shown in Figure 6-6, my Windows host by default has a public IP address assigned to it, and RDP protocol is enabled.

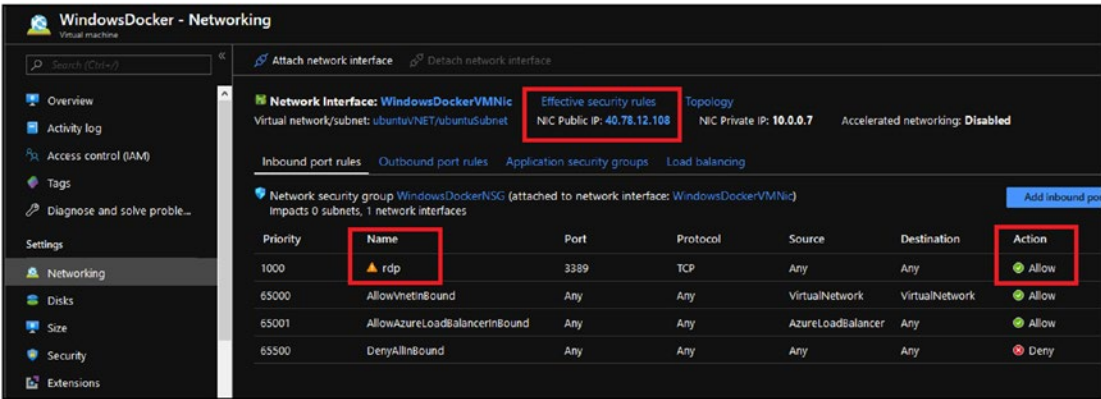


Figure 6-6. Windows Docker host networking

We will go ahead and enable Azure Bastion on this host. After Bastion is fully configured, we can disable RDP or SSH access.

Enable Azure Bastion on an Azure VM

To enable Bastion on my host, I will click on the Overview page of my Windows Azure VM.

From the Overview page, I will click on “Connect,” as shown in Figure 6-7.

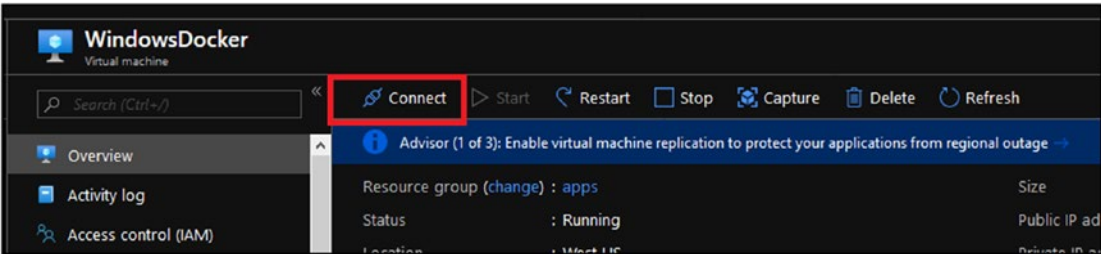


Figure 6-7. Connect option

After I click on the “Connect” button, Azure will give me the option to connect to my VM using RDP, SSH, or Bastion.

You can see the Connect menu in Figure 6-8.

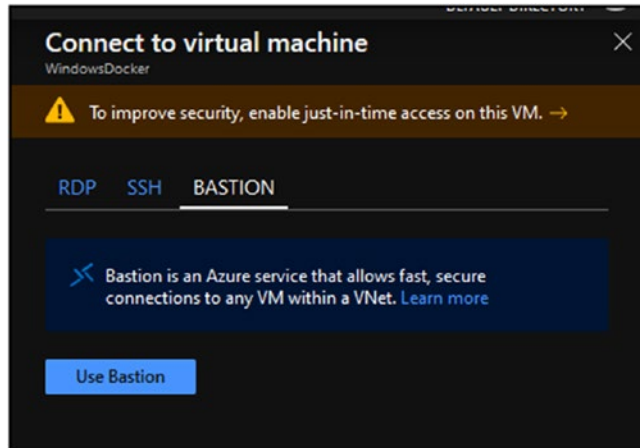


Figure 6-8. *Connect to Virtual Machine menu*

To use Bastion, I will click on the “Use Bastion” option. I am then asked to configure the Bastion service.

Most of the configuration is self-explanatory. However, the important part of the configuration is the Bastion subnet. As shown in Figure 6-9, I can’t complete the configuration without creating a management subnet called `AzureBastionSubnet`. To create the subnet, I will click on “Manage subnet configuration.”

Name *

ubuntuVNET-bastion

Virtual network ⓘ

ubuntuVNET

Subnet *

AzureBastionSubnet

Manage subnet configuration

✖ To associate a virtual network with a Bastion, it must contain a subnet with name AzureBastionSubnet with prefix of at least /27.

Public IP address

Public IP address * ⓘ

☒ Create new

☐ Use existing

Public IP address name *

ubuntuVNET-ip

Public IP address SKU

Standard

Assignment *

☐ Dynamic

☒ Static

Resource group * ⓘ

Apps

Create new

Subscription ⓘ

Create

Figure 6-9. Configure Bastion

In the Create Subnet page, I click on the plus sign next to “Subnet” to create a new subnet. Figure 6-10 shows the Subnet page and the option to create a subnet.

+ Subnet + Gateway subnet

🔍 Search subnets

Name	↕	Address range
ubuntuSubnet		10.0.0.0/24

Figure 6-10. Create subnet

After clicking on the Subnet page, I will fill in the subnet details. The most important part is to name the new subnet as follows (this is how Azure knows where to deploy the Bastion resource): AzureBastionSubnet.

Figure 6-11 shows the Add Subnet page. I leave all the details with the default values.

Add subnet
ubuntuVNET

Name *
AzureBastionSubnet ✓

Address range (CIDR block) * ⓘ
10.0.1.0/24 ✓
10.0.1.0 - 10.0.1.255 (251 + 5 Azure reserved addresses)

☐ Add an IPv6 address space

Network security group
None ▾

Route table
None ▾

Service endpoints
Services ⓘ
0 selected ▾

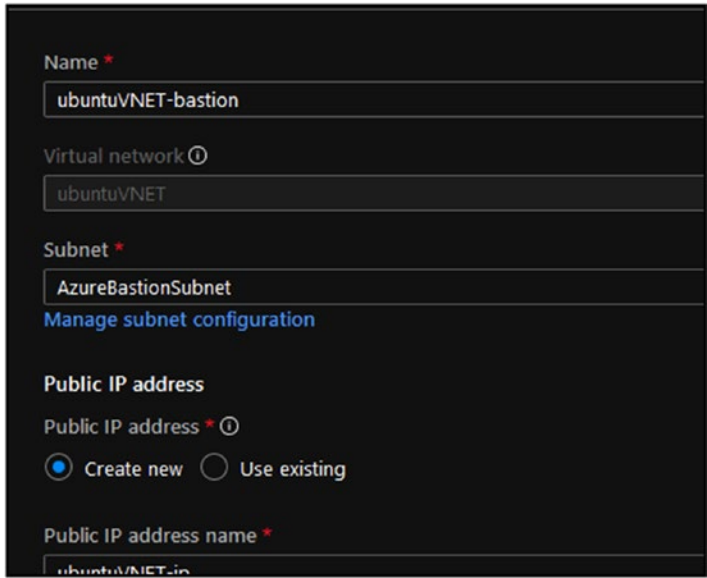
Subnet delegation
Delegate subnet to a service ⓘ
None ▾

OK

Figure 6-11. Add Subnet page

After creating the subnet, I go back to the Bastion menu.

From the menu, I select the newly created subnet and create a new IP address, as shown in Figure 6-12.

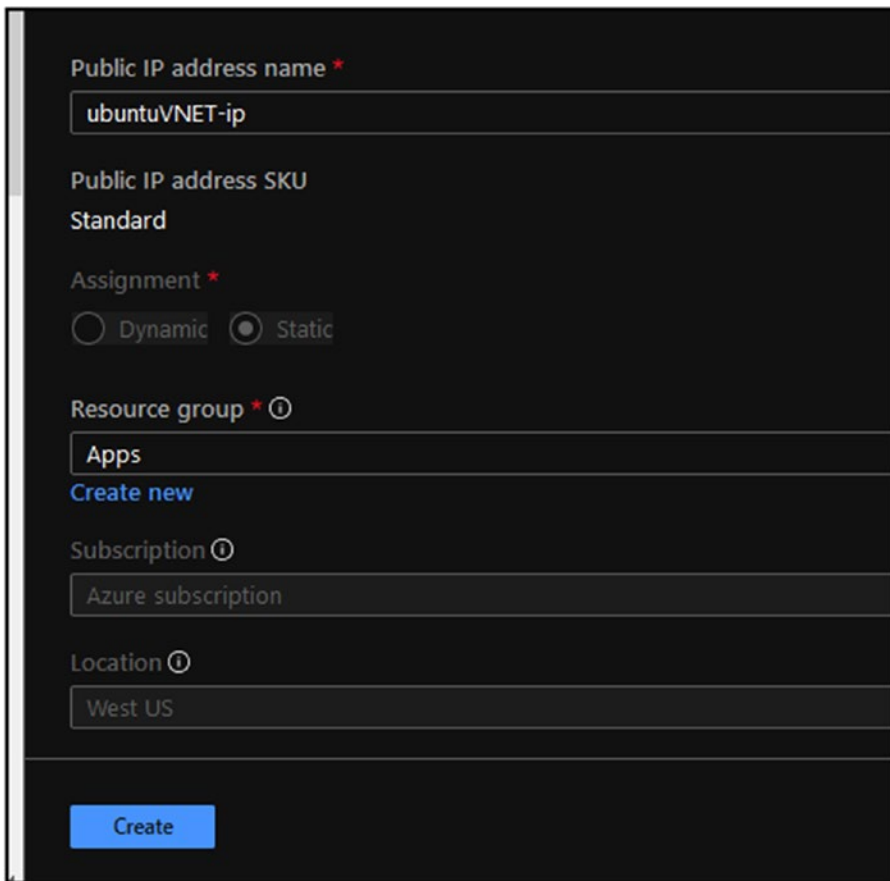
The image shows a configuration form for an Azure Bastion resource. The form has a dark theme. It includes the following fields and options:

- Name ***: A text box containing 'ubuntuVNET-bastion'.
- Virtual network ⓘ**: A dropdown menu showing 'ubuntuVNET'.
- Subnet ***: A dropdown menu showing 'AzureBastionSubnet'. Below this dropdown is a blue link that says 'Manage subnet configuration'.
- Public IP address**: A section header.
- Public IP address * ⓘ**: A label for the radio button options.
- Radio buttons**: Two radio buttons are present. The first is labeled 'Create new' and is selected (indicated by a blue dot). The second is labeled 'Use existing' and is not selected.
- Public IP address name ***: A text box containing 'ubuntuVNET-ip'.

Figure 6-12. Subnet and public IP address configuration

Make sure you select an existing resource group. In my case, I will select my apps resource group.

To complete the wizard, click on “Create,” as shown in Figure 6-13.



Public IP address name *

ubuntuVNET-ip

Public IP address SKU

Standard

Assignment *

☐ Dynamic ☒ Static

Resource group * ⓘ

Apps

Create new

Subscription ⓘ

Azure subscription

Location ⓘ

West US

Create

Figure 6-13. Select resource group option and click “Create”

Once Bastion is ready, I will click again on “Connect” and use the Bastion tab to connect to my VM. This time, I will get a different option to connect to my VM, as shown in Figure 6-14. To connect, I will type the VM username and password and click “Connect.”

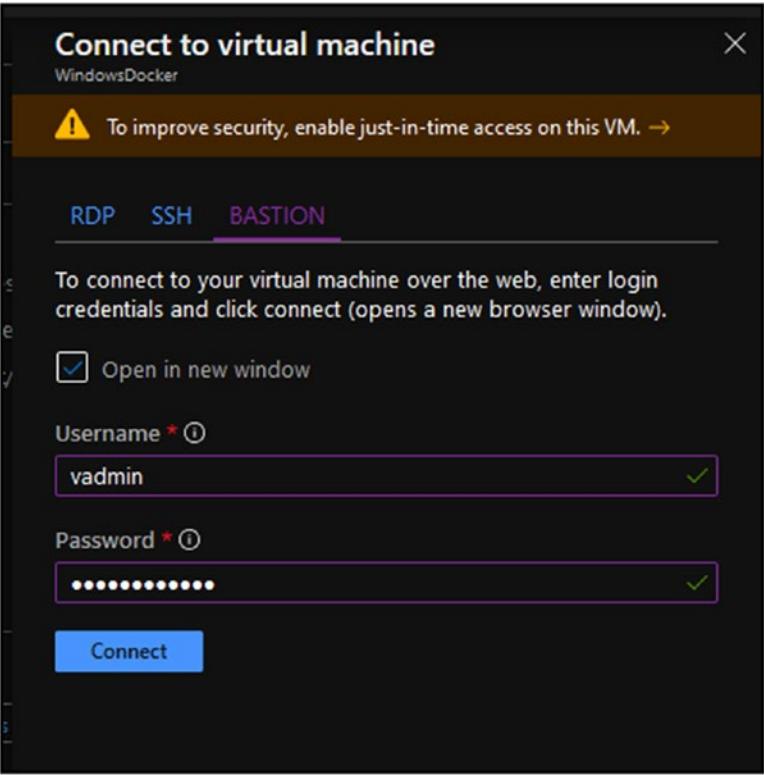


Figure 6-14. *Connect to a VM using Azure Bastion*

Note Make sure your browser popup blocker is disabled for Azure.com.

After connecting, Azure Bastion will show the desktop of my VM, the same as if I were to connect using remote desktop.

Figure 6-15 shows the desktop on my Windows Server 2019 Docker container host. You can also see in Figure 6-15 the tabs of my web browser.

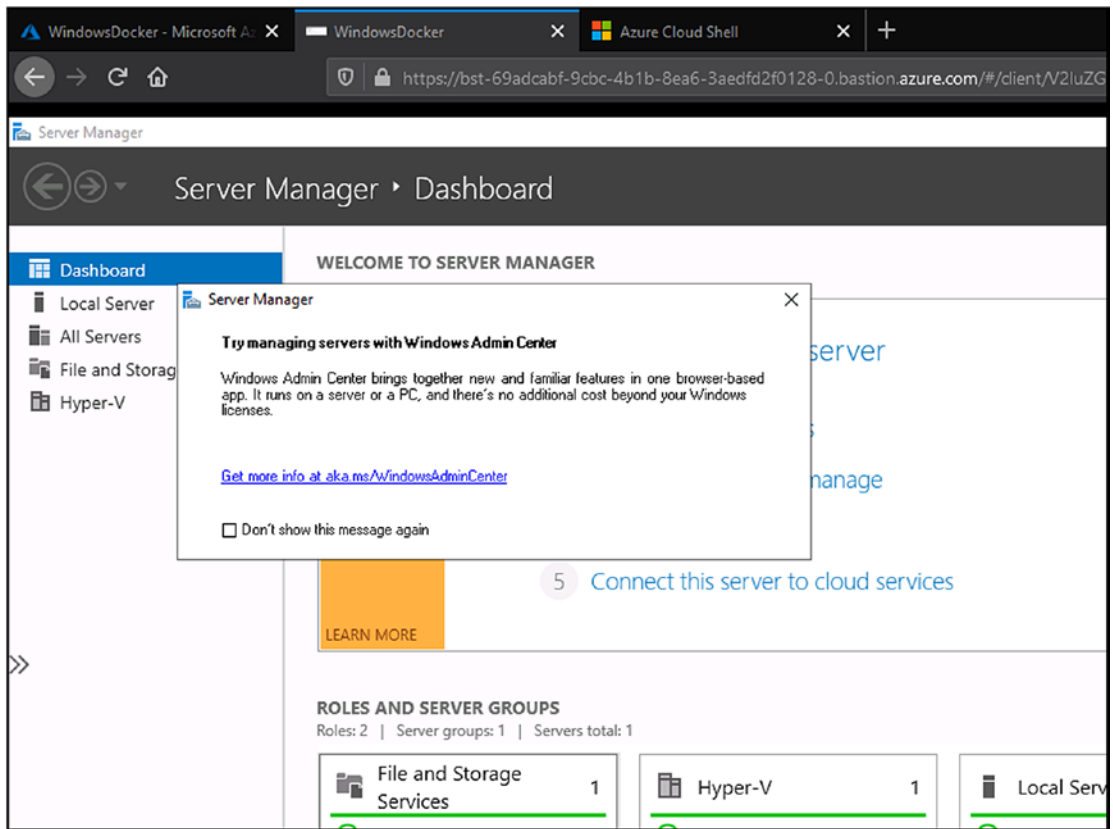


Figure 6-15. Bastion in action

Manage Bastion

The last thing I will cover in this section is the management of Bastion from the Azure portal and Azure PowerShell, starting with how to access Bastion from the portal.

To view Bastion from the Azure portal, in the search box type Bastion; the first result will come up as Bastions.

As shown in Figure 6-16, it displays as Bastions (plural) because Azure allows us to create multiple Bastions for different environments. To manage it, click on the link from the search result, as shown in Figure 6-16.

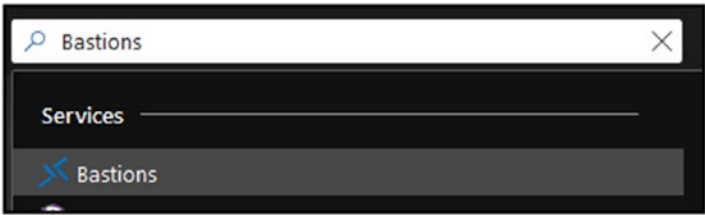


Figure 6-16. *Bastions*

After clicking on the link, you will see the configuration page with all management options available for Bastion. Figure 6-17 shows the Bastion main page.

From the main page, you can delete the Bastion via the Delete option in the overview page.

To view active sessions, click on “Session,” and you will see which sessions are active.

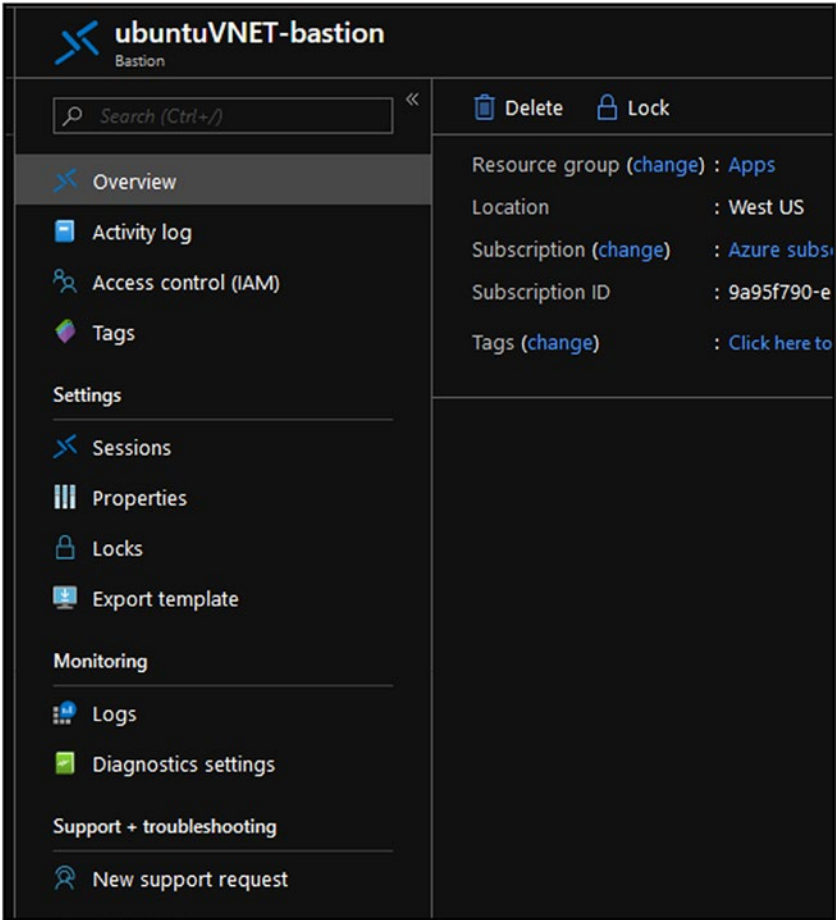


Figure 6-17. *Bastion main page*

Create Bastion Using Azure PowerShell

If you prefer to create an Azure Bastion using the command line, you can do so using Azure PowerShell.

You may have noted that you need to use Azure PowerShell and not Azure CLI.

This is because Azure Bastion is actually based on Microsoft Remote Services technology with HTML 5 enabled. Since RDS is running on Windows Server Remote Desktop Services behind the scenes, we need to use Azure PowerShell and Azure CLI.

So, if you decide to create or manage Bastion from the command line, select PowerShell from the Azure Cloud Shell console and use the following commands:

New-AzBastion — Use this command to create a new Bastion.

Get-AzBastion — Use this command to view the existing configuration.

For a full syntax of the commands, visit the following: <https://docs.microsoft.com/en-us/powershell/module/az.network/new-azbastion?view=azps-3.1.0>.

Use Azure Security Center and Secure Score to Protect Your Tenant

In this section, we will cover and learn about Azure Security services that allow us to keep our tenant secure and safe: Azure Security Center and Azure Secure Score.

These two services are very important because they constantly monitor the security health of our Azure resource and run security assessments that tell us which risks we can mitigate using best practices.

Security Center

Microsoft Azure Security Center is a cloud-based security management and threat-protection service for Azure resources.

Security Center comes with two pricing tiers:

Free

Standard

The Free tier comes with basic security assessment and recommendations for Azure virtual machines and app services.

The Standard tier comes with added security features like the following:

- Just-in-time-administration
- Compliance desktop
- Threat protection for Azure VMs and PaaS services
- SQL, MySQL, and PostgreSQL database protection
- Storage protection
- IT devices protection

The last security feature can be integrated with Azure Container Registry (ACR).

You can access the Azure Security Center from the portal by typing Security Center in the search box. When you first access Security Center it will show you all the features in the Standard tier, as shown in Figure 6-18; however, if you scroll down you will see that it will ask you to upgrade to the Standard tier.

Figure 6-19 shows the upgrade option, which in our case we don't want, because we want to start with the Free tier.

Next to the "Upgrade" button, click on "Skip," which will let you start with the Free tier option.

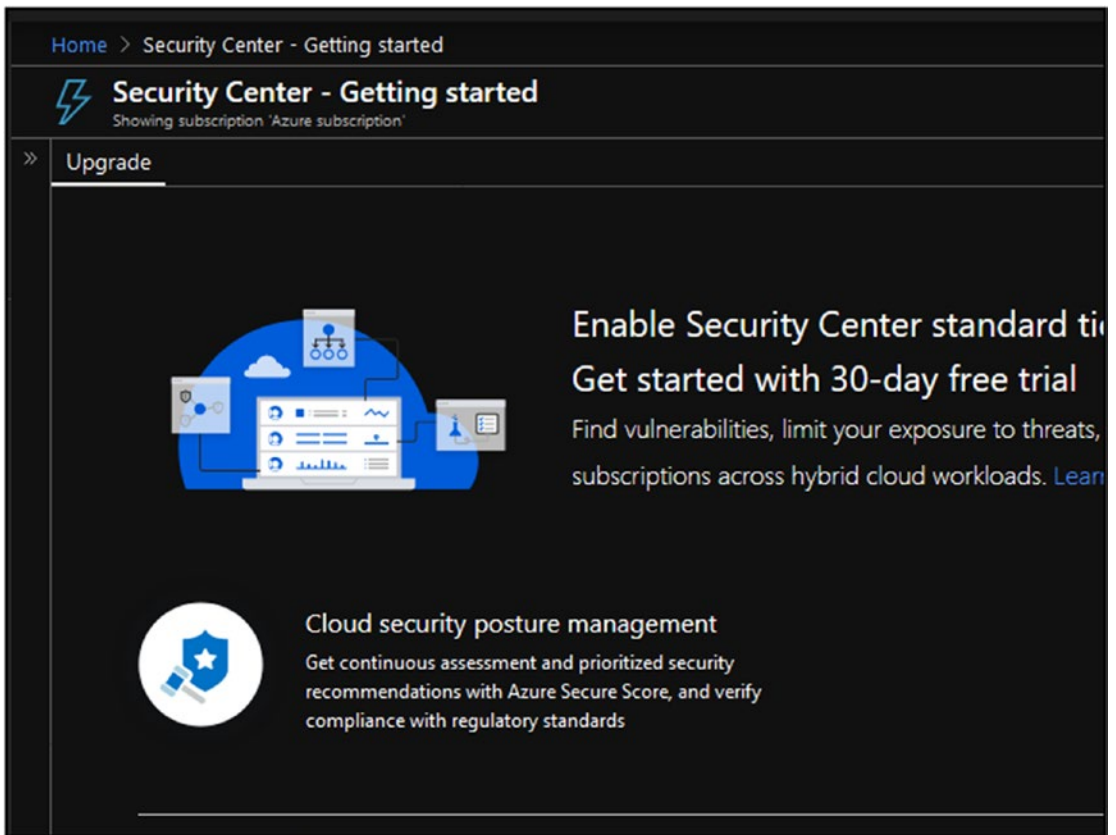


Figure 6-18. *Getting started with Security Center*

This action is really important, because if you click “Upgrade” it will start a thirty-day free trial period. Make sure you start with the Free tier by clicking “Skip,” as shown in Figure 6-19.

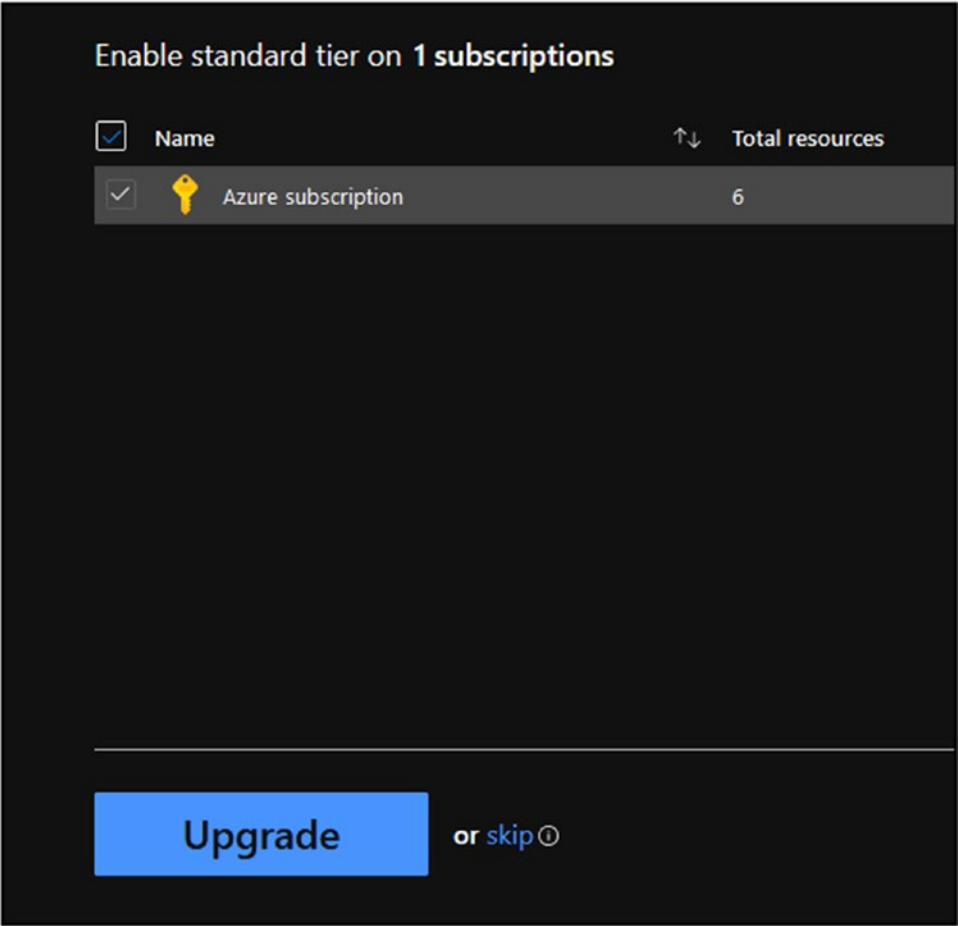


Figure 6-19. Upgrade to Standard tier option or skip

After clicking “Skip,” you will be taken to the Security Center home page, where Azure will show you all your resources and the security assessment for each service.

Figure 6-20 shows the Security Center overview page.

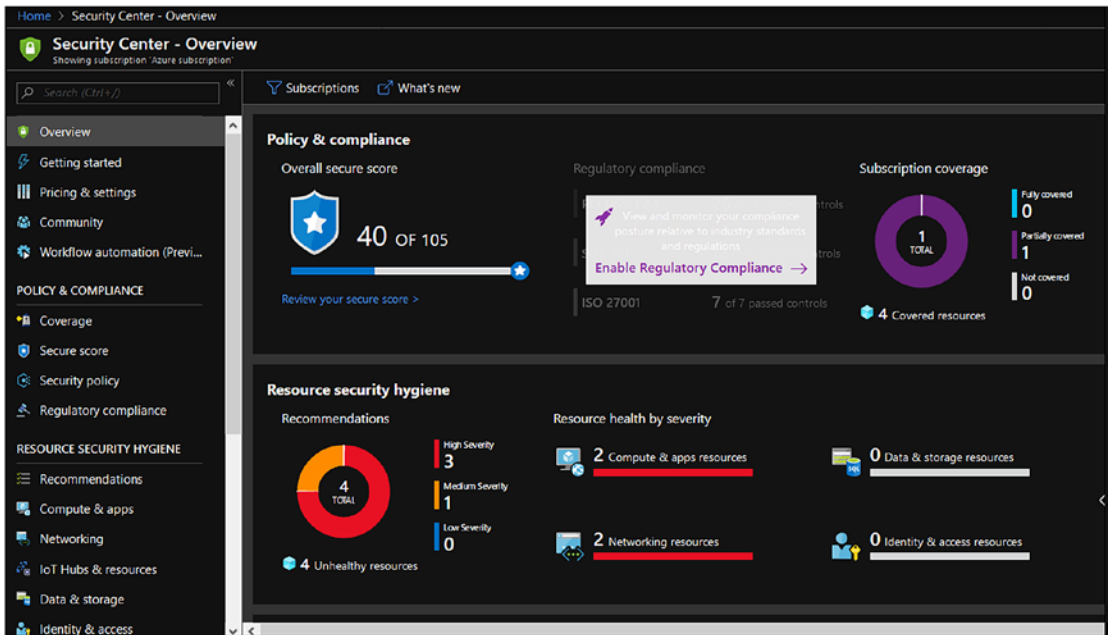


Figure 6-20. Security Center Overview page

You should review the recommendations available for your resources in the Recommendations section. For example, for my tenant, I have seven total recommendations that can significantly improve the overall security of my Azure environment.

In Figure 6-21, you can see the outstanding recommendation for my tenant.

Some of them can be fixed by a simple, quick-fix solution that takes less than a minute and has zero impact on the performance of the environment.

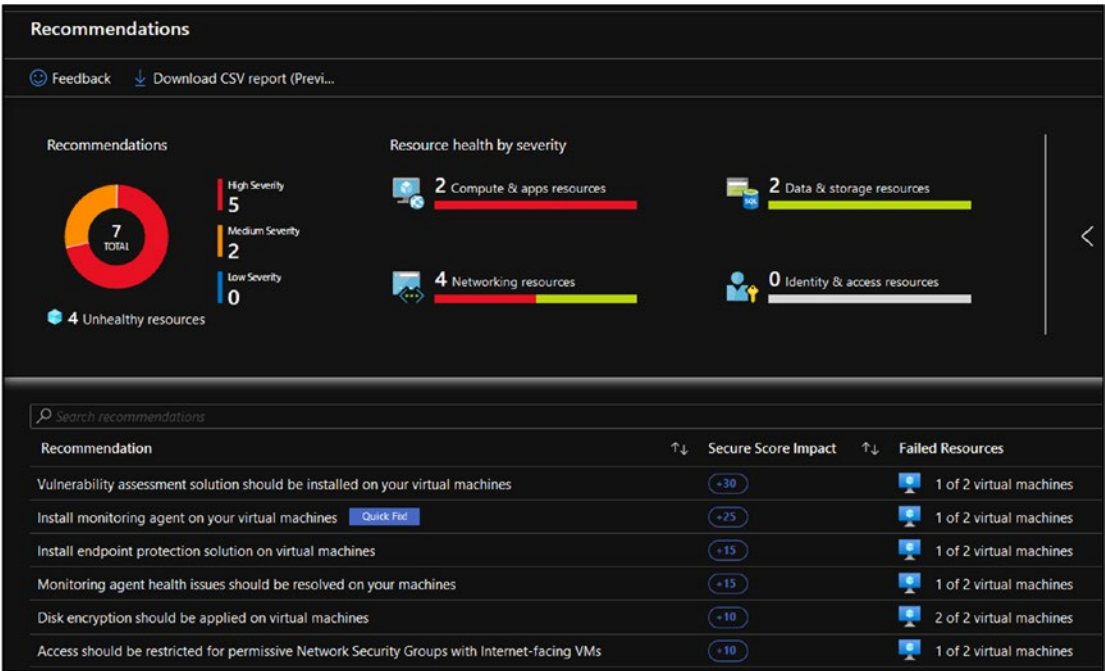


Figure 6-21. Security Center recommendations

Secure Score

Microsoft Secure Score is a security and compliance tool that helps us improve our security posture by comparing the security configuration of our Azure resource to the IT industry’s best practices.

Secure Score is available for Azure Security Center, and if you look at Figure 6-22, you can see the Secure Score of my Azure tenant. As shown, the score sits at 115 points out of 225 points, where 225 is the highest level possible for the security posture.



Figure 6-22. *Secure Score*

To review the score in great detail, click on the “Review your secure score” link.

From the Secure Score Dashboard page, shown in Figure 6-23, you can see the overall score, as well as categories attached to the subscription.

To improve the score and review the recommendations, click on the subscription name at the bottom of the page.

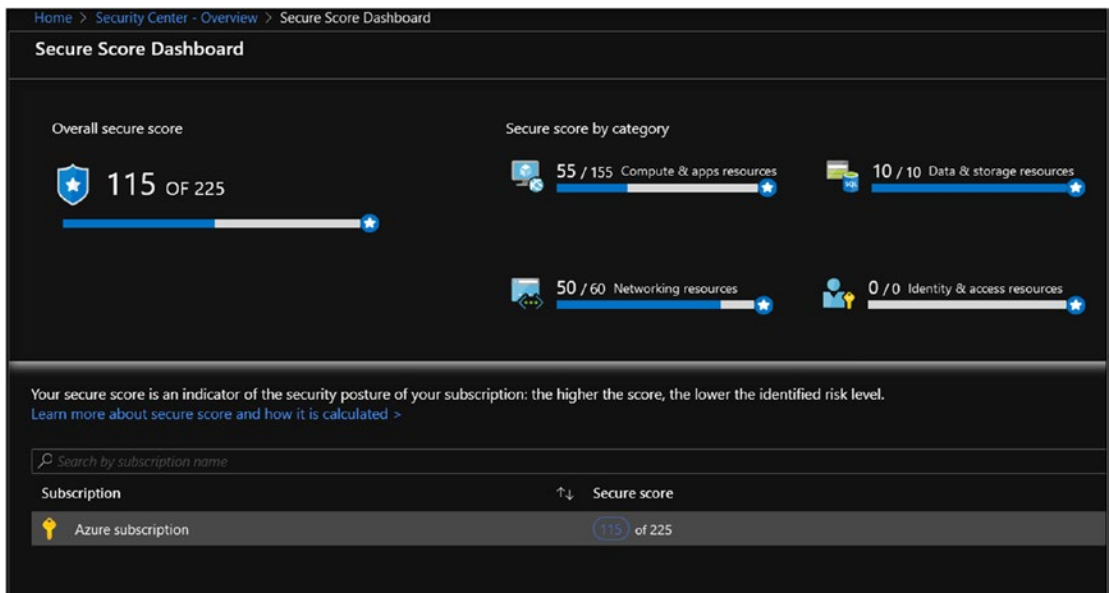


Figure 6-23. *Secure Score Dashboard*

After clicking on the subscription, you will see the recommendation. If you fix the recommendation in Azure Secure Score, it will impact the overall Secure Score posture score.

My recommendation here is to review the recommendations once a month and start by fixing any offering a quick-fix option that has zero impact on environmental performance and stability.

Secure Your Containers with Azure Firewall and Network Security Groups (NSG)

In this section, we will cover Azure Firewall and Network Security Groups (NSG), two network security features that you might need to deploy in the future.

The first feature, Azure Firewall, is an enterprise solution that is probably only needed for high-end and large-scale deployments, as it is typically used in hub-and-spoke networks.

NSGs are applicable for all deployments regardless of the size of the organization.

Azure Firewall

Azure Firewall is a cloud-based firewall service that is enterprise-ready and comes with advanced capabilities and high availability.

Azure Firewall comes with a high price tag because of the advanced feature set, and it costs \$1.25 per deployment hour and \$0.016 for every GB of data that is processed by the firewall.

I will not go into configuring Azure Firewall because of the complexity of the product; however, I feel that it is a solution you should know about in case your Azure environment is growing rapidly and you need to be protected using an enterprise-grade firewall.

Network Security Groups (NSGs)

Network Security Groups (NSGs) are much simpler than Azure Firewall and act as a basic firewall that allows us to protect our Azure virtual machines.

By default, Azure created an NSG for every deployed VM. A single NSG can be configured and applied to multiple machines by attaching it.

Create and Apply an NSG

To create an NSG, from the Azure portal search for Network Security Groups and click on the link.

Figure 6-24 shows the NSG link from the Azure search bar.

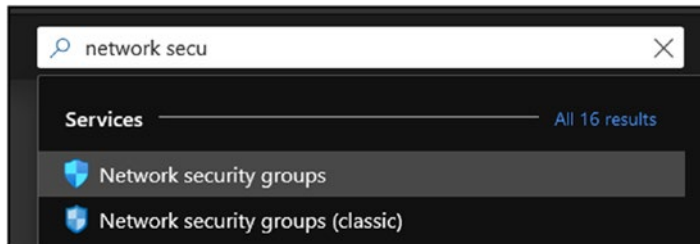


Figure 6-24. Azure NSG

In my case, I have four NSGs that I didn't create and that were created automatically when Azure created my VMs.

Figure 6-25 shows all the NSGs in my Azure tenant.

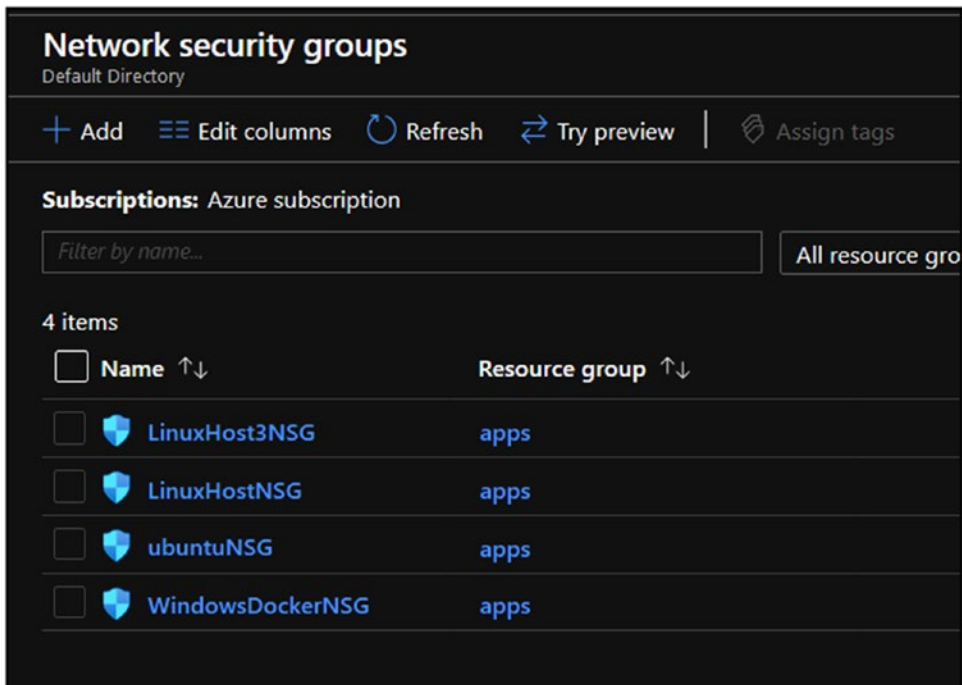


Figure 6-25. Azure NSGs

In this lab, I will create a new NSG that will allow only remote desktop and SSH protocols.

To create a new NSG, click “Add” in the top menu.

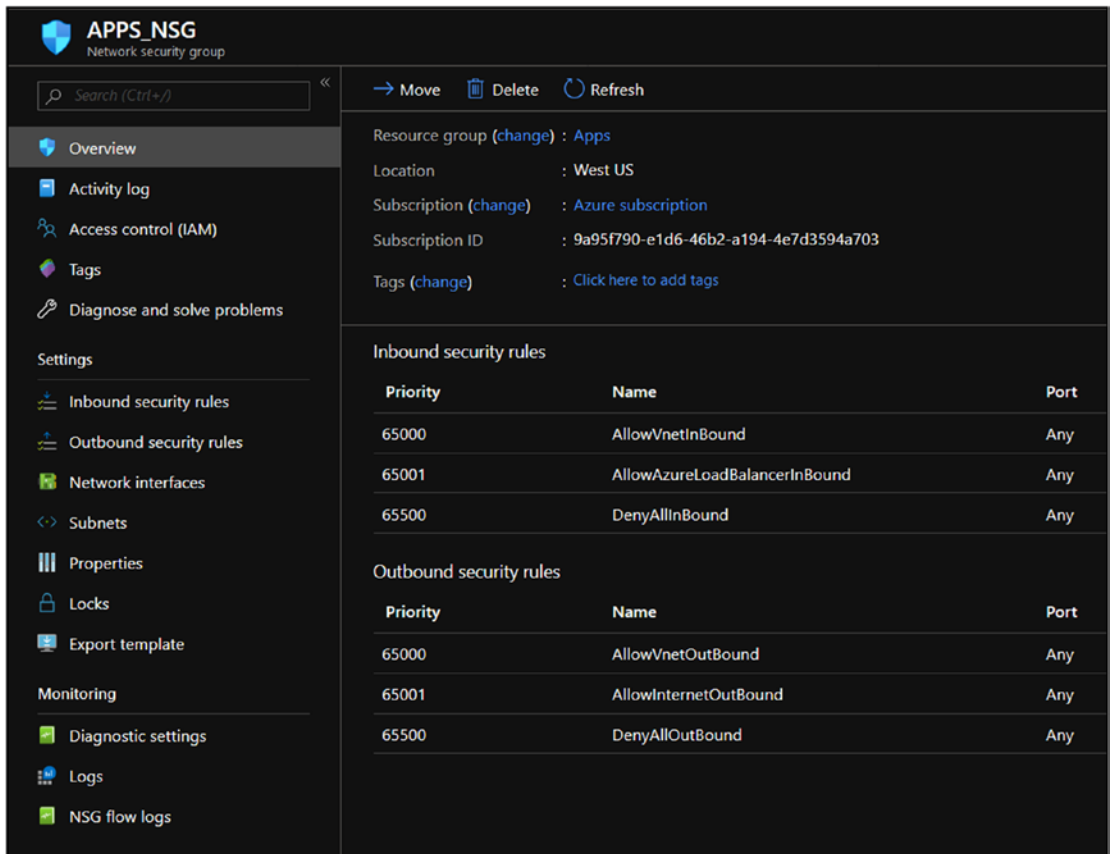
On the Create Network Security Group page, I will select my Azure subscription and resource group, type a name, and select a region, as shown as Figure 6-26.

Figure 6-26. *Create a network security group*

Once the NSG is created, I can go ahead and configure it with inbound and outbound security groups that will control which traffic is allowed.

If you look at Figure 6-27, you will see that the NSG comes with three default security rules that cannot be changed and belong to the Azure platforms.

To delete the NSG, you can use the Delete option from the overview page. To move it to another resource group, use the “Move” button.



APPS_NSG
Network security group

Search (Ctrl+/)

Move Delete Refresh

Resource group (change) : Apps

Location : West US

Subscription (change) : Azure subscription

Subscription ID : 9a95f790-e1d6-46b2-a194-4e7d3594a703

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

Inbound security rules

Priority	Name	Port
65000	AllowVnetInBound	Any
65001	AllowAzureLoadBalancerInBound	Any
65500	DenyAllInBound	Any

Outbound security rules

Priority	Name	Port
65000	AllowVnetOutBound	Any
65001	AllowInternetOutBound	Any
65500	DenyAllOutBound	Any

Figure 6-27. Network Security Group Overview page

To create an inbound security group rule for remote desktop services, follow these steps:

1. Open the NSG.
2. Click on “Inbound Security rules.”
3. Click on “Add.”
4. On the Add Inbound Security Rule page, fill in the rule details; in my case, I am using TCP port 3389.

Figure 6-28 shows my RDP inbound security rule.

 **Add inbound security rule**
APPS_NSG

 Basic

Source * ⓘ

Any

Source port ranges * ⓘ

*

Destination * ⓘ

Any

Destination port ranges * ⓘ

3389

Protocol *

AnyTCPUDPICMP

Action *

AllowDeny

Priority * ⓘ

100

Name *

Port_3389

Description

Add

Figure 6-28. Add an inbound security rule

Associate NSG with Azure VM

Now that my RDP inbound rule is configured, I will go ahead and attach the NSG to the network interface of my Windows Server Docker container host VM.

To associate the NSG with my VM, I will follow these rules:

1. Open the VM from the portal.
2. Click on “Networking.”
3. From the Networking page, click on the network interface name shown in brackets.

In my case, you can see in Figure 6-29 the name of my network interface highlighted in red.

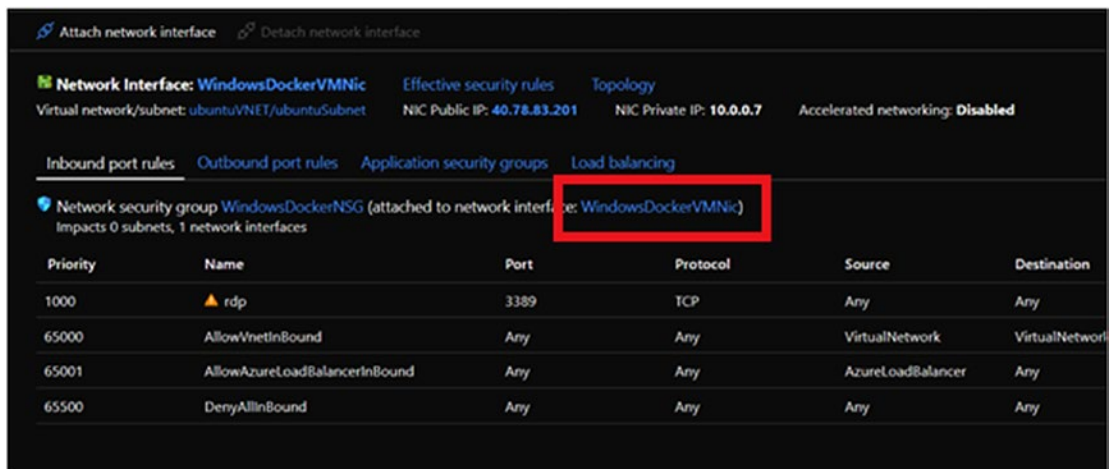


Figure 6-29. Network interface

From the Network Interface page, I will click on the “Network security group” link in the left-hand menu, as shown in Figure 6-30.

Next, I will change the NSG by clicking on the “Edit” button and clicking on the NSG name that is currently configured.

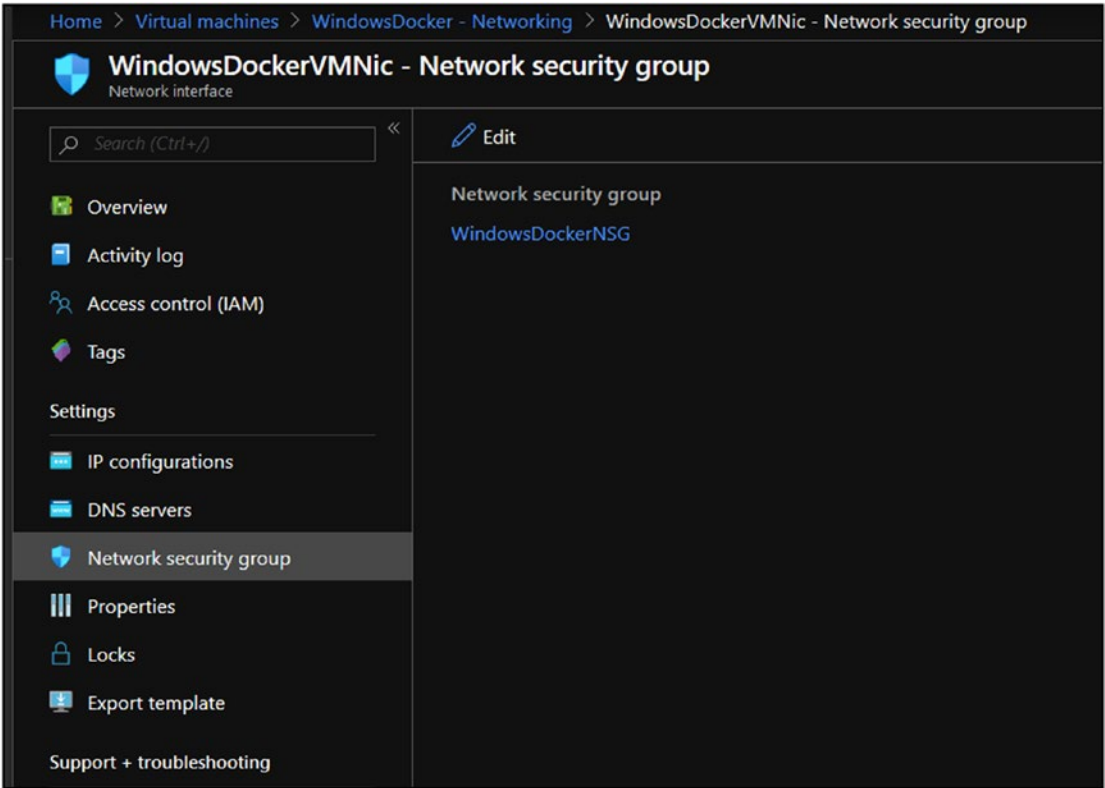


Figure 6-30. Network interface NSG

Figure 6-31 shows the edit option, and I will go ahead and click on the NSG name.

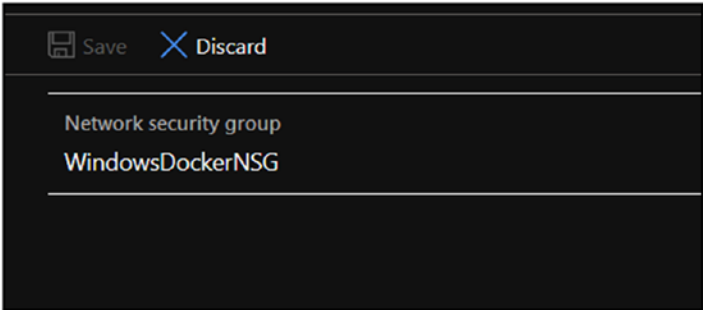


Figure 6-31. NSG name

After clicking on the currently configured NSG's name, I will be redirected to the Choose Network Security Group menu.

Figure 6-32 shows all the available NSG groups in my tenant, and as shown my NSG is available and is called APPS_NSNG.

To complete the change, I will simply click on the APPS_NSNG NSG.

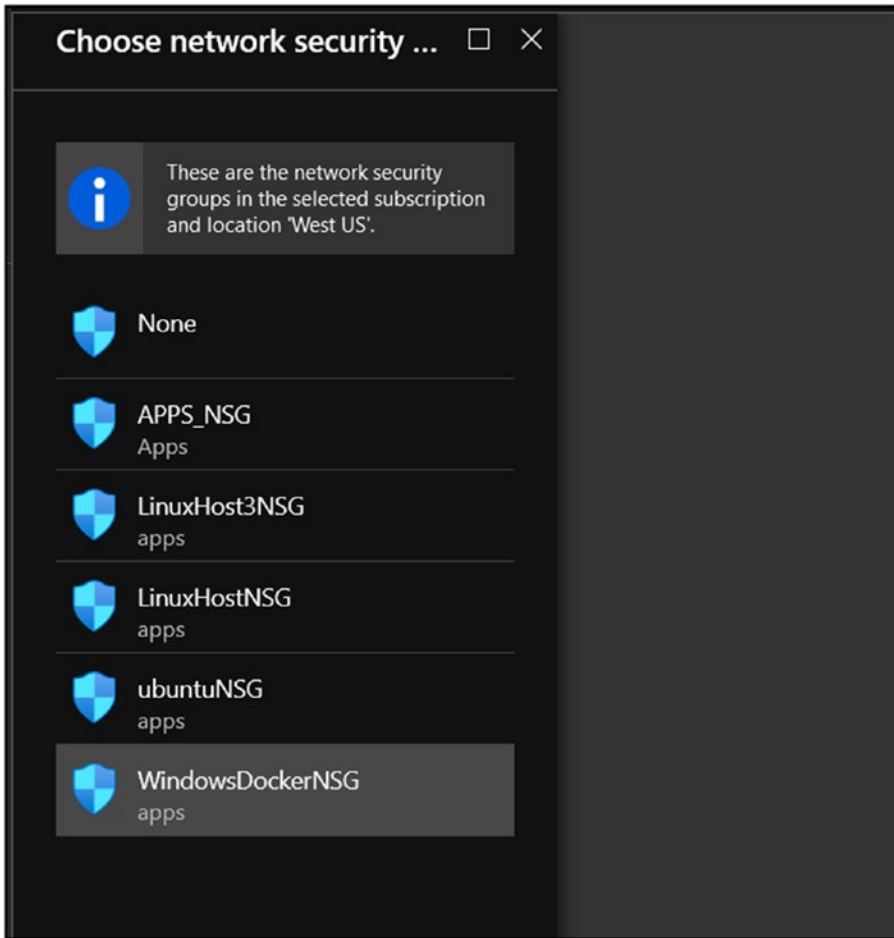


Figure 6-32. *Select NSG*

All I need to do now is confirm the change by clicking on “Save,” as shown in Figure 6-33.

Note Make sure you perform this change outside working hours.

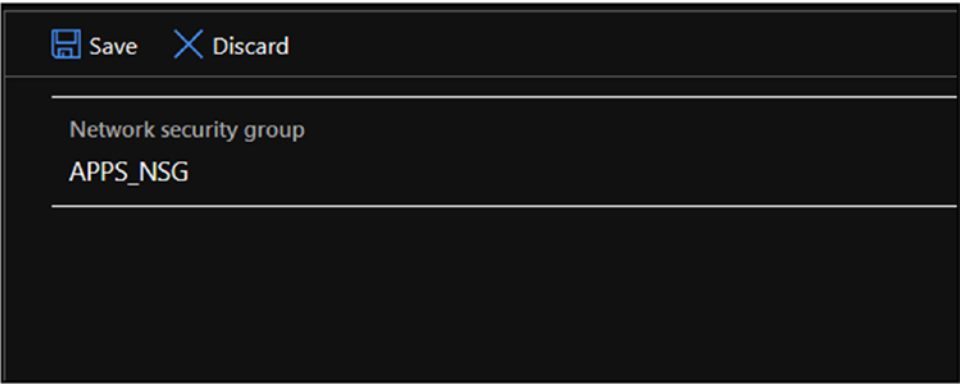


Figure 6-33. *Save configuration*

To confirm that the configuration has completed successfully, I will click on the “Network Interfaces” link from the APPS_NSG page and check if my Windows Docker VM network interface is listed.

Figure 6-34 shows the Network Interfaces page with my Windows VM network interfaces listed.

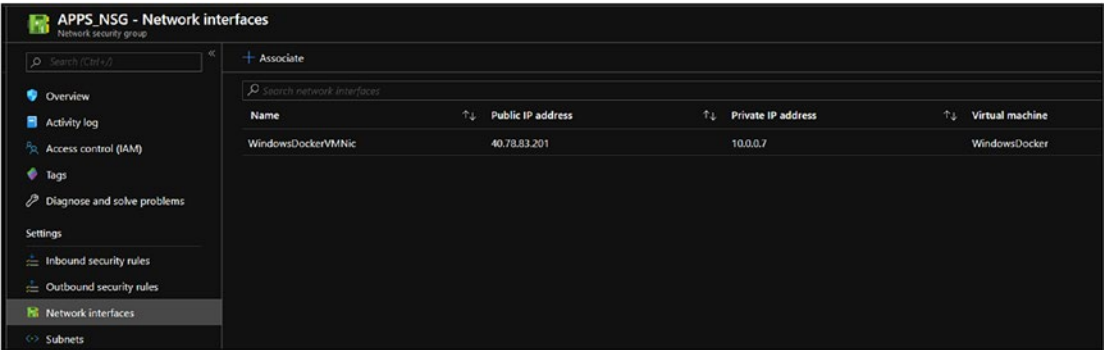


Figure 6-34. *NSG network interfaces*

At this stage, the configuration is complete.

My recommendation here is to use NSG in cases where all the VMs need to have the same security rules.

NSGs can also simplify the management of groups of machines where a single NSG controls the configuration of many VMs.

If you are not using NSGs, you will need to manage access per VM, which can get out of control very easily when access needs to be managed across many VMs.

Summary

In this chapter, we covered a few methods that can assist us in protecting our Azure resources and users.

We started with Azure AD sign-in logging and audit logs features and deployed a Bastion host for secure management of VMs using a web browser.

We also covered the Azure Security Center, and with Secure Score we learned how to apply recommendations that can improve our Secure Score posture.

And finally, we learned about Azure Firewall and how to create an NSG and associate it with existing Azure VMs.

CHAPTER 7

Scale Containers and Containerized Applications on Azure

In this chapter, we will learn how to scale all the Microsoft Azure container services we have learned about so far. In the previous chapters, we deployed a few container services, and in all of them we used the very basic instance to power them. This is because we only deployed the services for testing, not for production.

In a production environment, you will have situations where you need to scale your service because there is an increasing demand for your application. Microsoft Azure gives us the flexibility to scale almost any service in Azure and adjust it when demand increases. Using scaling and auto-scaling, we can better manage the efficiency of our Azure resources and control costs when demand increases.

In this chapter, we will cover the following topics:

- Scaling AKS clusters
- Scaling ACI
- Scaling virtual machine hosts
- Using Azure Spot virtual machines

Scale Azure Kubernetes Service (AKS)

Scaling AKS clusters is probably something you will need to manage and understand the most because of the nature of AKS.

AKS is an enterprise service that was designed to handle heavy loads and increased demand. It is also the most expensive service out of all the services we have used so far in this book, and opportunities to save are greatest when demand is low.

By default, if you remember from Chapter 3, AKS comes with three nodes, and if you are running just a test AKS cluster, that added cost is not needed. In Chapter 3, we changed the default host count to one, which is enough for testing and development.

Now, we will go over the process of creating a cluster and scaling it.

Create an AKS Cluster

To get started, I will create a new AKS cluster using Azure CLI via Azure Cloud Shell. The following Shell script will create the cluster (4.1.Create_AKS.sh):

```
#!/bin/bash

az aks create --resource-group apps \
  --name aks \
  --node-count 1 \
  --node-vm-size Standard_DS1_v2 \
  --enable-addons monitoring \
  --generate-ssh-keys $true
```

To run the script, from Azure Cloud Shell run the following command:

```
. 4.1.Create_AKS.sh
```

The script will create an AKS cluster with a single Standard_DS1_V2 node.

Scaling an AKS Cluster

Before scaling our AKS cluster, I would like to explain the concept of the AKS node pool. By default, AKS is configured to use a node pool to manage the worker nodes of our cluster. A node pool groups together worker nodes with the same configuration.

After creating my AKS cluster using Azure CLI, I will need to find out the node pool name, which can be done using the following Azure CLI command:

```
az aks show --resource-group apps --name aks --query agentPoolProfiles
```

For your convenience, you will find this script under the following name:

7.1..Show_AKS_Agentpool.sh

From the command output, I will copy the pool name, which in my case is as follows:

nodepool1

Now that I have the node pool name, I will use the following command to scale my cluster by an extra node so as to have a two-node cluster:

```
az aks scale --resource-group apps --name aks --node-count 2 --nodepool-name nodepool1
```

For your convenience, I have saved this Azure CLI script under the following name:

7.2.Scale_AKS.sh

After a few minutes, Azure will scale the cluster.

To see the number of nodes in my cluster using the Azure portal, I will open my AKS cluster from the portal and click on “Node pools” in the side menu (Figure 7-1).

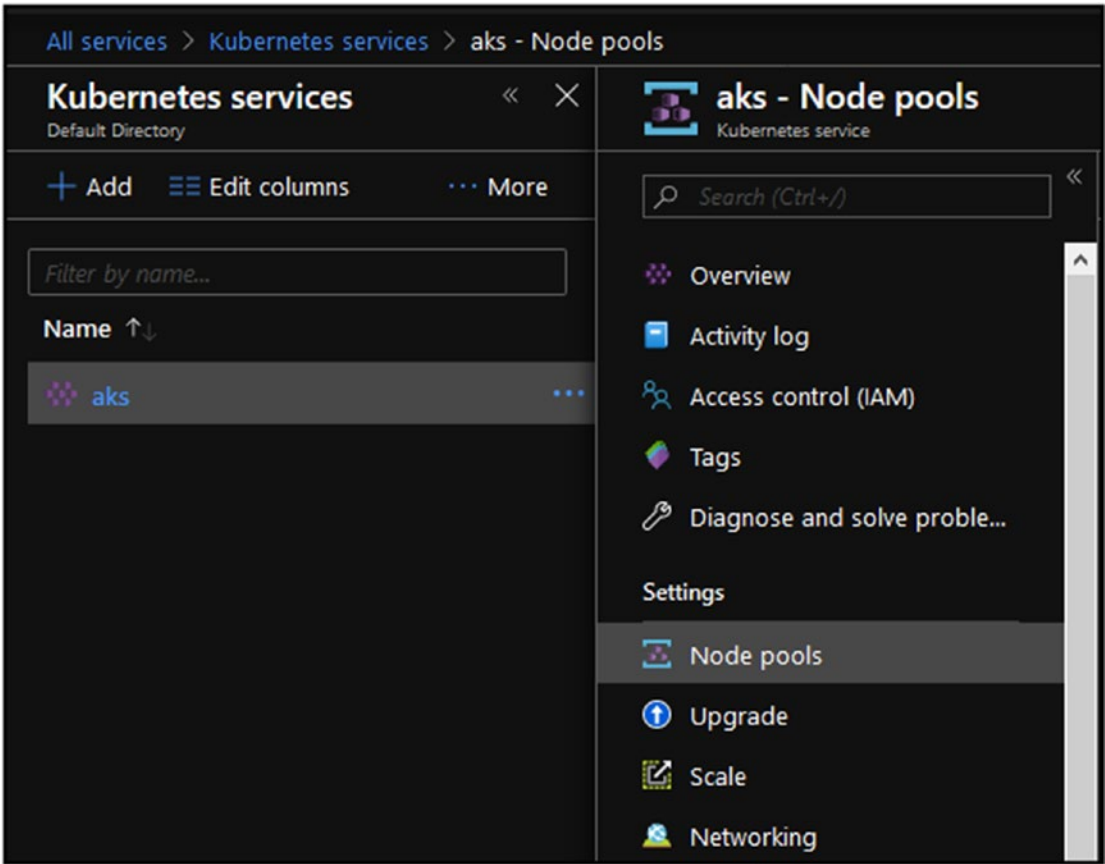


Figure 7-1. Node pools

The Node Pools page displays most of the information you need to know about your node pools, which includes the following:

- Number of node pools (you can have multiple node pools for different workloads)
- Name
- State
- Kubernetes version
- OS type (Windows or Linux)
- Node count
- Node instance size

Figure 7-2 shows the Node Pools page with my node pool.

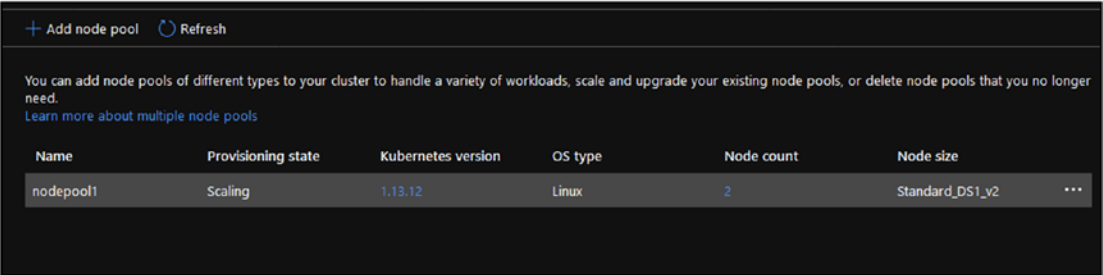


Figure 7-2. Node Pools page

Scale AKS Cluster Using the Portal

If you don't want to scale your cluster using Azure CLI and Azure Cloud Shell, you can use the Azure portal. Scaling your cluster from the portal is done from the Node Pools page.

From the Node Pools page, click on the three dots (...) menu and click on "Scale," as shown in Figure 7-3.

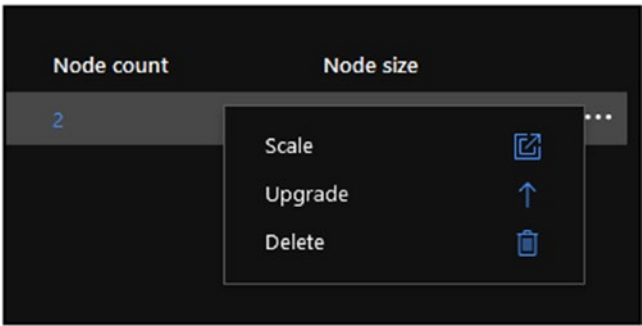


Figure 7-3. Scale AKS

The Scale menu will give you options to scale your cluster using a very intuitive interface, as shown in Figure 7-4. Just remember that you can't scale your cluster to zero nodes.

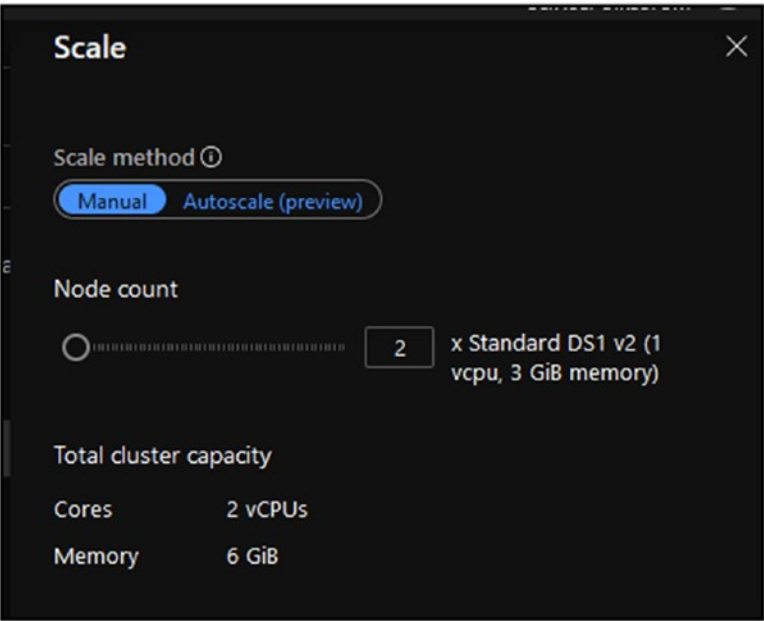


Figure 7-4. *Scale menu*

To complete the scaling process, click “OK” and wait a few minutes for the operation to complete.

Autoscaling an AKS Cluster

What if you don’t want to worry about your capacity and scaling operations? Azure has a good solution for that.

Autoscaling can do the scaling work for you without your worrying about it. The feature belongs to Kubernetes, and Azure will act on behalf of your Kubernetes master’s need for more resources.

We configure autoscaling from the same menu in which we scale our cluster, but this time we only need to select a minimum and a maximum capacity.

Figure 7-5 shows the Autoscaling feature. This feature is currently in preview mode, and you might not see it your scale menu if the feature is not available in your region.

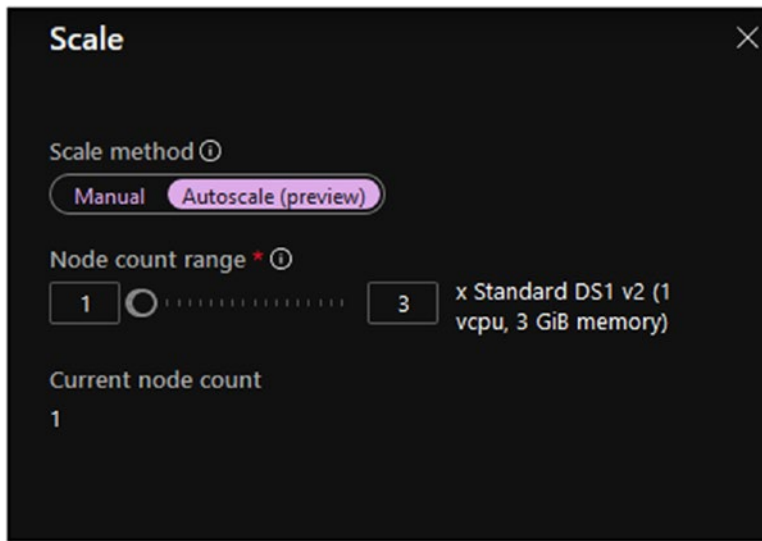


Figure 7-5. Autoscale

My recommendation is to use this feature if it is available in your region and cluster. Autoscaling can reduce the need for manual scaling and can scale your cluster up and down depending on the needs of the cluster.

Autoscale AKS Using Azure CLI

If you prefer not to use the Azure portal to autoscale your cluster, you can always use Azure CLI.

In the following code, I am creating a new AKS cluster with autoscale configured from the beginning.

You will find the script under the following name:

```
7.3.Create_AKS_Autoscale.sh
#!/bin/bash
az aks create --resource-group apps \
  --name aks \
  --node-count 1 \
  --node-vm-size Standard_DS1_v2 \
  --enable-addons monitoring \
  --generate-ssh-keys $true \
```



```
--enable-cluster-autoscaler \  
--min-count 1 \  
--max-count 3
```

Update Existing AKS Cluster to Autoscale

You can also update an existing cluster to use autoscale by using the following Azure CLI command:

```
az aks update --resource-group apps --name aks --update-cluster-  
autoscaler --min-count 1 --max-count 3
```

Disable Autoscale Using Azure CLI

To disable autoscale using Azure CLI, you can use the following code:

```
az aks update --resource-group apps --name aks --disable-cluster-autoscaler
```

Delete an AKS Cluster Using Azure CLI

And, finally, I would like to share with you the following Azure CLI command that deletes an existing AKS cluster, which I use all the time to delete my AKS lab:

```
az aks delete --name aks --resource-group apps
```

Scale Azure Container Instances (ACI)

In Chapter 3, we learned about ACI and how to deploy containers to the ACI service.

ACI is a good service with which to run batch jobs and small deployments for testing; however, it is not what you call an enterprise-ready solution for production workloads.

By design, ACI doesn't allow us to scale it. If you need to scale an ACI service, you need to delete the deployment and create a new one with more resources.

Yes, I know that this might be disappointing to hear, but it is the truth.

In the following example, I have an ACI deployment that is using a Docker image I have in ACR. By default, when deploying an ACI container, it will use 1 CPU and 1.5 GB of RAM.

In the following code, I am deploying the code and specifying CPU and memory. This time, I am using 2 CPUs and 3.5 GB of RAM.

```
az container create -g apps --name nginxwebapp02 --CPU 2 --memory  
3.5 --location westus --dns-name-label nginxwebappv --registry-username  
deployrg --registry-login-server deployrg.azurecr.io --image deployrg.  
azurecr.io/nginx:latest --registry-password passwordhere
```

If you have ACI workloads you need to scale, simply delete the deployment and create a new one with more resources. You can do this using the Azure portal instead of using Azure CLI.

To use the Azure portal to check your current ACI instance size, use the following steps:

- Open ACI from the Azure portal.

- From the ACI homepage, click on “Containers.”

- From the Containers page, click on “Properties” in the bottom menu and review the values in CPU cores and Memory.

Figure 7-6 shows the Properties menu.

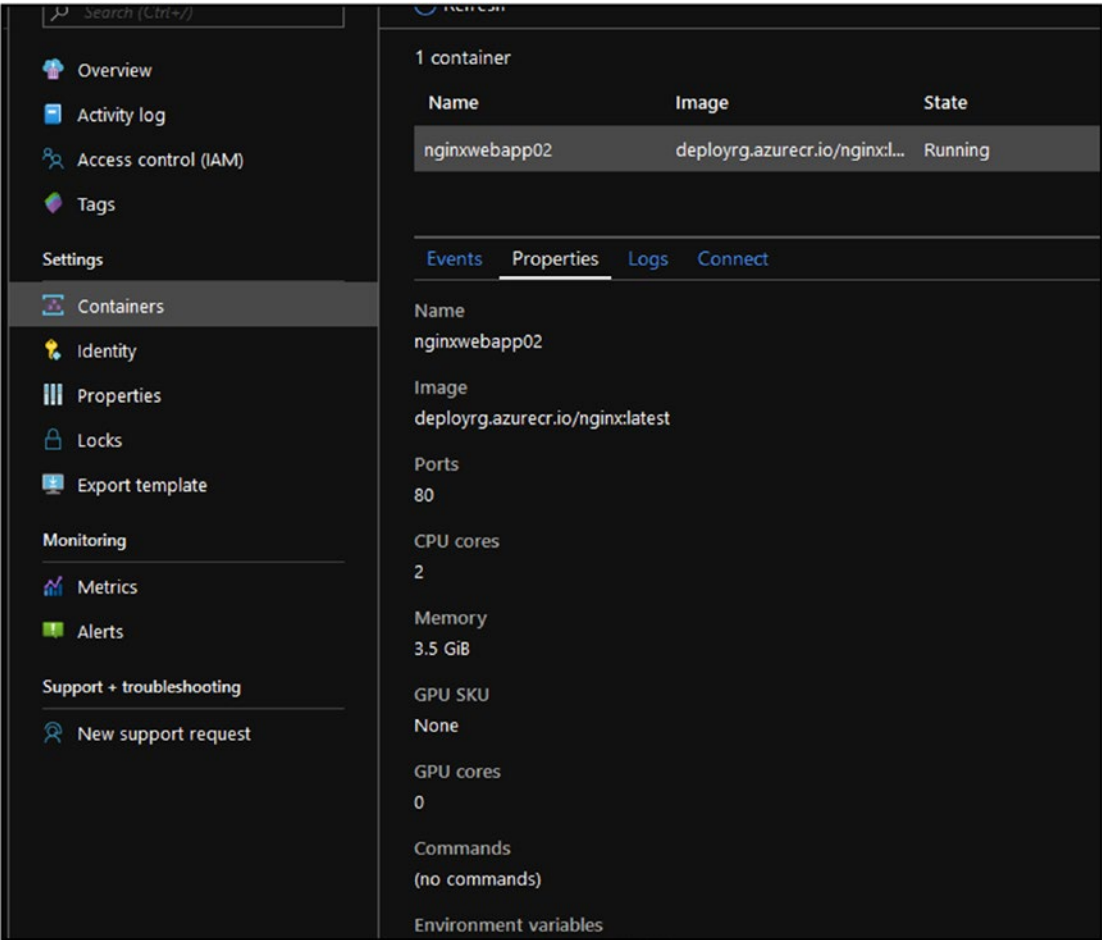


Figure 7-6. ACI Properties menu

To use Azure CLI and view all your ACI containers and their instance sizes, use the following code:

```
az container list -o=table
```

To delete an ACI deployment, use the following code:

```
az container delete --name nginxwebapp02 --resource-group apps
```

Personally, I think that when using ACI, it is much more convenient to use Azure CLI because it allows us to deploy and redeploy our code much faster without using the GUI configuration screen.

It is important to note the instance size limits of ACI. ACI offers a minimum instance size of 1 CPU core and 500 MB of RAM. The maximum capacity ACI offers is 4 CPU cores and 16 GB of RAM.

Scale Azure Container Registry (ACR)

In Chapter 2, we learned about Azure Container Registry (ACR) and how it helps store and manage Docker container images.

ACR provides a secure and safe environment that we can pull and push Docker container images from builds. Under the hood, ACR is a storage service that has many smarts integrated into it.

In a nutshell, ACR provides the following two main services:

Storage — Disk storage to save our image

Security — Security feature to secure access to our registry and maintain patched and secure images

Because of the preceding features, Azure gives us the option to scale our ACR service in the form of updating the service plan we are using.

ACR comes with three service plans that essentially allow us to scale ACR based on our needs.

In Chapter 2, we covered all the ACR plans, but in a nutshell, we have the following options:

Basic plan — 10 GB storage

Standard plan — 100 GB storage

Premium plan — 500 GB storage

Based on the ACR plans, you can adjust your ACR capabilities and capacity during peaks. Just remember that ACR pricing is based on daily usage, so you can easily change the plan if more capacity is needed for a specific period.

Scale Azure Docker Container Host VM

In this section, we will learn how to scale a Docker container host running as a virtual machine on Azure regardless of whether it is a Linux or a Windows host.

Virtual machines offer great flexibility when it comes to scaling simply because of their fewer moving parts. A VM is a complete computing unit with fewer dependencies compared to AKS or even ACI.

Scale an Azure VM Using the Portal

We will start with scaling our Azure VM using the portal, which you probably use more for ad-hoc changes.

To get started, open the Azure portal. Click on “Virtual Machines” or search for it using the search box. From the Virtual Machines menu, click on the VM you need to scale. From the VM menu, click on “Size,” as shown in Figure 7-7.

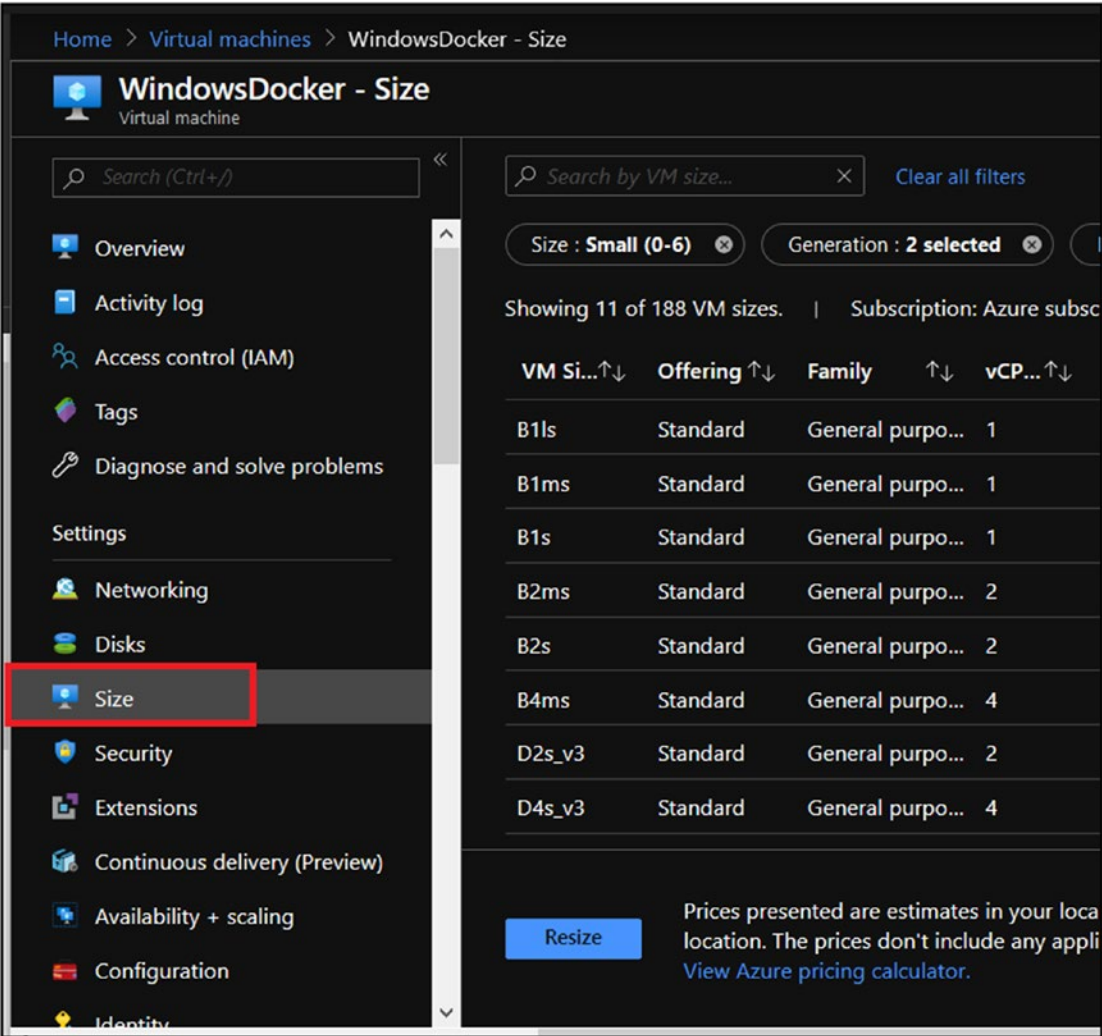


Figure 7-7. Size

All you need to do now is select the new instance size and click “Resize” to complete the scaling process.

Note During the scaling process, the VM will restart, so make sure you do this change outside business hours.

Scale an Azure VM Using Azure CLI

We can also scale our VM instance size using Azure CLI via Azure Cloud Shell.

The following command will scale my Windows Docker container host to the Standard DS3 V2 instance size:

```
az VM resize --resource-group apps --name windowsdocker --size Standard_DS3_v2
```

Using Azure CLI can be handy when you need to resize the VM as part of a process and want to automate the resize task.

Azure Spot Virtual Machines

In December 2019, Microsoft announced the preview of Azure’s spot virtual machines.

Azure spot VMs give us access to unused computing capacity in Azure infrastructure at a heavy discount price and are good for the following:

- Batch processing
- Development and test workloads
- Stateless applications that can easily scale up
- Short processing jobs
- Any workloads that can be interrupted

Azure Spot Virtual Machines Pricing

According to the Microsoft Azure VMs pricing page, spot VMs are priced at up to a 90 percent discount compared to pay-as-you-go pricing options.

For example, if you look at the table in Figure 7-8 (taken from the Microsoft Azure spot VMs pricing page) you can see the potential saving power of spot VMs.

Add to estimate	Instance	vCPU(s)	RAM	Temporary storage	Pay as you go	1 year reserved (% Savings)	3 year reserved (% Savings)	Spot (% Savings)
+	DS1 v2	1	3.5 GiB	7 GiB	\$0.126/hour	\$0.0886/hour (~30%)	\$0.0733/hour (~42%)	\$0.0504/hour (~60%)
+	DS2 v2	2	7 GiB	14 GiB	\$0.252/hour	\$0.1773/hour (~30%)	\$0.1466/hour (~42%)	\$0.1011/hour (~60%)
+	DS3 v2	4	14 GiB	28 GiB	\$0.504/hour	\$0.3555/hour (~29%)	\$0.2940/hour (~42%)	\$0.202/hour (~60%)
+	DS4 v2	8	28 GiB	56 GiB	\$1.008/hour	\$0.7101/hour (~30%)	\$0.5861/hour (~42%)	\$0.403/hour (~60%)
+	DS5 v2	16	56 GiB	112 GiB	\$2.016/hour	\$1.4201/hour (~30%)	\$1.1731/hour (~42%)	\$0.806/hour (~60%)

Figure 7-8. Spot VMs pricing

Using the same instance VM I used for my Docker hosts, I can achieve a 60 percent savings by using spot VMs.

My recommendation here is to evaluate your workloads, compute needs, and utilize spot VMs if possible.

Deploy a Spot Virtual Machine

Now that we know how much spot VMs cost and what we can do with them, I will show you how to deploy a spot VM.

To create a spot VM, we will use the same process as when we created a normal VM, but this time we will select “Spot VM” from the Create menu, as you will see.

From the Azure portal, search for virtual machines.

From the Virtual Machines main page, click on “Add” to create a new VM.

As shown in Figure 7-9, you can see the Spot VM option and how it is fully integrated into the process of creating a VM.

Instance details

Virtual machine name * ⓘ WinDockerSpot ✓

Region * ⓘ (US) West US ✓

Availability options ⓘ No infrastructure redundancy required ✓

Image * ⓘ Windows Server 2019 Datacenter with Containers ✓
[Browse all public and private images](#)

Azure Spot instance ⓘ ☒ Yes ☐ No

Eviction type ⓘ

- ☒ Capacity only: evict virtual machine when Azure needs the capacity for pay as you go workloads. Your max price is set to the pay as you go rate.
- ☐ Price or capacity: choose a max price and Azure will evict your virtual machine when the cost of the instance is greater than your max price or when Azure needs the capacity for pay as you go workloads.

Eviction policy ⓘ ☒ Stop / Deallocate ☐ Delete (currently not supported)

Size * ⓘ Standard DS2 v2
 2 vcpus, 7 GiB memory (Price unavailable)
[Change size](#)

Figure 7-9. “Spot VM” option

Eviction Policy

The lifespan of a spot instance is dependent on the eviction policy you select, and when that policy kicks in, the VM will be stopped and deallocated.

When Azure needs the unused capacity for pay-as-you-go workloads, the spot VM will be stopped/deallocated.

At this stage and during the preview mode, Azure offers only the “Stop/Deallocate” option; however, if you look closely at Figure 7-9, you will see there is a “Delete” option, which will delete the VM when Azure needs the capacity.

Currently, Azure offers two options for the eviction policy:

Capacity only — When Azure needs the capacity, the VM will stop.

Capacity and price — In this option, you can also specify a max price you are willing to pay for the spot instance, and if the price goes above your price, the VM will be stopped.

Note Azure Spot VMs have no SLA.

Create a Spot VM Using Azure CLI

You can also use Azure CLI to create a spot VM, and for this example, I will use the same script we used to create a Linux Docker host, but with an extra two lines that instruct Azure to use spot VM.

For this example, I will use the following script (7.4.Create_Spot_VM.sh):

```
az vm create --name "LinuxSpotVM" \  
    --resource-group "apps" \  
    --image "UbuntuLTS" \  
    --size Standard_DS2_v2 \  
    --location westus \  
    --admin-username vadmin \  
    --data-disk-sizes-gb 50 \  
    --admin-password enterpassswordhere \  
    --generate-ssh-keys \  
    --priority Spot \  
    --max-price -1
```

It is very important that a spot virtual machine cannot be converted to a non-spot VM.

Summary

In this chapter, we learned how to scale and increase the capacity of our Microsoft Azure container services. Scaling is an essential process that takes place in any environment. Azure has made the process of scaling services easy and risk-free; however, you should be aware of some of the limitations ACI has when it comes to scaling.

AKS has the most disruptive scaling process because of the usage of node pools, which seamlessly bring more nodes to the cluster and let Kubernetes load balance the load without affecting existing workloads. We also learned how to use autoscaling in AKS, which turns the operation of scaling into a set-and-forget task that is done once (if done correctly).

On the front of the virtual machine, we learned how to resize our VMs using the Azure portal and using Azure CLI.

In the last section, we covered the new Azure spot virtual machines service, which allows us to reduce the cost of some of our workloads by 60 percent by using unused computing power in Azure.

If you decide to use spot VMs, please make sure your workloads are suitable for spot VMs.

CHAPTER 8

Monitor Containers and Containerized Applications on Azure

In this chapter, we will learn how to monitor our container services in the Microsoft Azure infrastructure using Azure Monitor and related tools. Monitoring services' availability is a critical component in running a production environment, regardless of whether it is an enterprise environment or not. Microsoft Azure offers an array of services that can help us manage availability and monitor our services.

In this chapter, the following topics will be covered:

- Monitoring Azure Kubernetes Service (AKS)
- Monitoring Azure Container Instances (ACI)
- Monitoring Azure Container Registry (ACR)
- Monitoring Azure Docker Container host VM

Azure Monitor Overview

All of the monitoring services we are going to explore in this chapter are managed by Azure Monitor. Azure Monitor is the umbrella service that delivers monitoring solutions to all the Azure services.

The following services can be monitored with Microsoft Azure Monitor:

- Azure tenant
- Azure resources

- Azure subscriptions
- Operating systems
- Applications

Azure Monitor Products

Azure Monitor has the following products:

- Application Insights — Application-level monitoring
- Azure Monitor for Virtual Machines — VM-level monitoring
- Azure Monitor for Containers — Container-level monitoring

All the data that is being collected by Azure Monitor is stored in Azure Log Analytics, which we can tap into using log queries. Using log queries, we can join many data sources together and analyze our operations' data.

Azure Monitor also allows us to use smart alerts and to take action using Azure automation, which includes autoscaling.

Azure Monitor Data

When Azure Monitor monitors our environment, it collects data that is then stored as two types of data.

Metrics

These are numerical values like CPU usage or free disk space in an AKS host. We analyze metrics using metrics explorer; for example, viewing VM performance via the VM Overview page in the Azure portal.

Logs

Logs can be event log data collected from a container running in AKS. We analyze logs using Azure log queries from the Log Analytics portal.

Azure Monitor Activity Logs

When you create a new Azure subscription, Azure Monitor collects activity-based logs on the subscription level called Activity Logs.

Activity Logs collects logs that are related to creating, modifying, and deleting resources in Azure; they are kept for 90 days. If we need to store activity logs for more than 90 days, we can use Azure Monitor to export the data to an Azure storage account and keep it for a longer period.

Monitor Azure Kubernetes Service (AKS)

It is now time to learn how Azure Monitor is integrated with AKS and see it in action.

If you remember, in Chapter 4 when we used the Azure CLI Bash script to create our AKS cluster we also enabled Azure Monitor.

If you look at the following code (4.1.Create_AKS.sh), at line 6 we are enabling monitoring by using `enable-addons`.

```
#!/bin/bash
az aks create --resource-group apps \
  --name aks \
  --node-count 1 \
  --node-vm-size Standard_DS1_v2 \
  --enable-addons monitoring \
  --generate-ssh-keys $true
```

By default, monitoring is enabled when you create an AKS cluster using the portal. If you look at Figure 8-1, you can see the default selection.

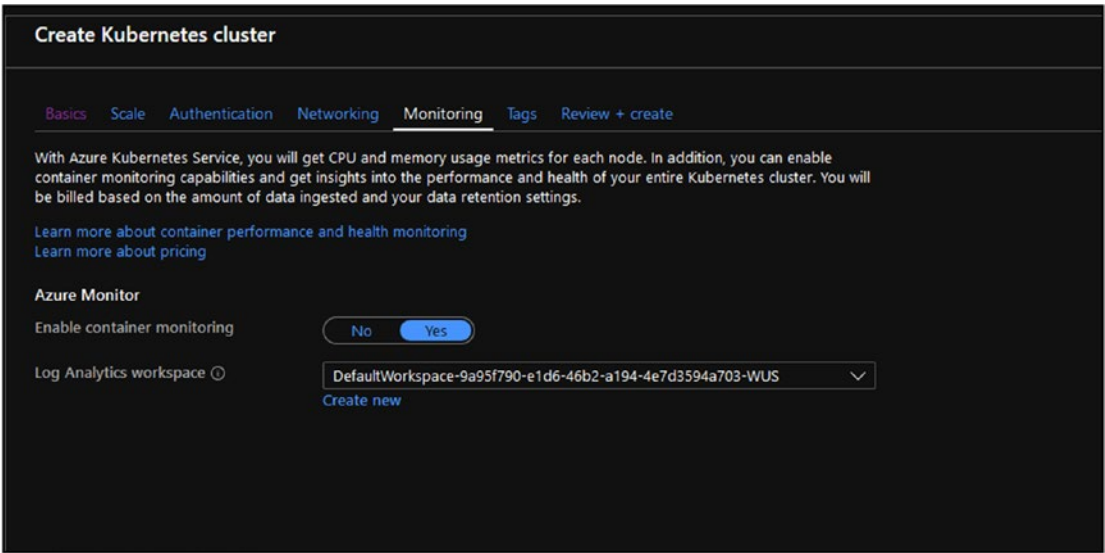


Figure 8-1. *Enable Azure Monitor*

By default, Azure will also create a Log Analytics workspace to store the logs. If you decide not to use monitoring, Azure will monitor for free the CPU and memory usage of the nodes only; if you go with the default option, Azure Monitor will also monitor the running containers inside the cluster, including all nodes and services.

Log Analytics

Azure Monitor pricing is based on the amount of data Log Analytics ingests. The price is \$2.99 per GB per month, and the first 5 GB are free every month. To check the usage and cost of your Log Analytics, perform the following steps:

- Search for Log Analytics in the search box.
- Click on the Log Analytics workspace you would like to check the cost for.
- From the left-hand pane, click on “Usage and estimated costs.”

Figure 8-2 shows the Usage and Estimated Costs screen.

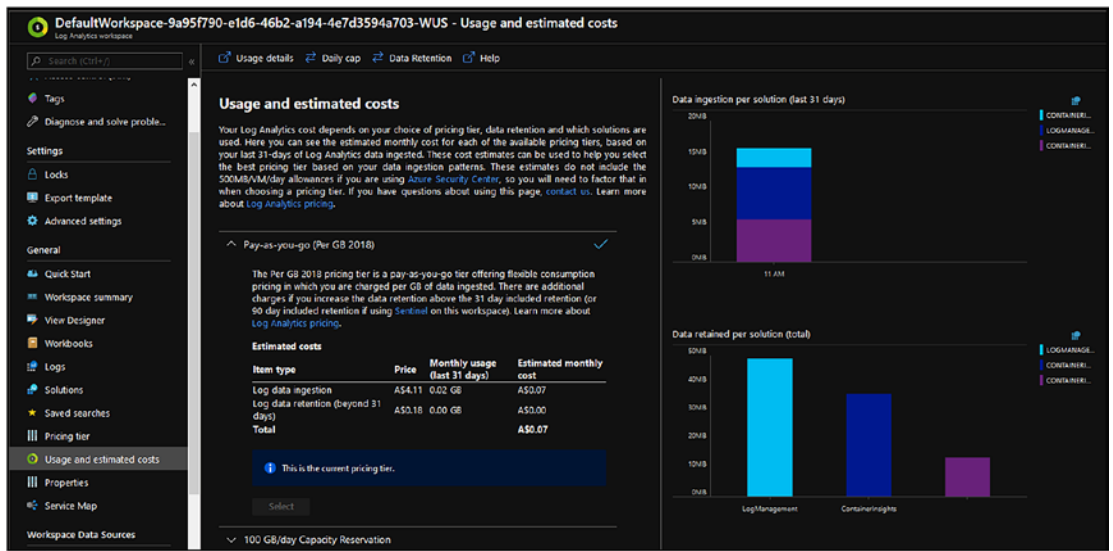


Figure 8-2. *Usage and Estimated Costs screen*

Monitoring

Out of the box, AKS comes with the following three monitoring solutions:

1. Insights
2. Metrics
3. Logs

With any Azure service, you can see which monitoring option is available under the Monitoring section in the left-hand menu pane, as shown in Figure 8-3.

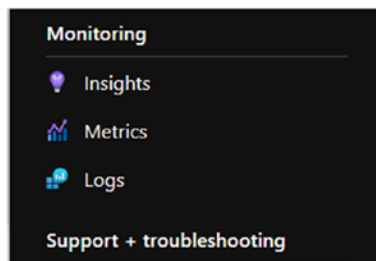


Figure 8-3. *Monitoring*

Insights

Azure Monitor Insights for AKS gives us great monitoring capabilities for each level of our AKS solutions. Using Insights, we can monitor the following:

- Cluster-level monitoring
- Kubernetes monitoring that includes API server and all related components
- Worker nodes monitoring
- Containers monitoring
- Deployments monitoring

Figure 8-4 shows the main screen of AKS Insights, and, as shown, you can see the Cluster Monitoring Overview page (default). The Cluster page gives an overview of how the AKS cluster runs, including live monitoring of all resources in real-time without any delay.

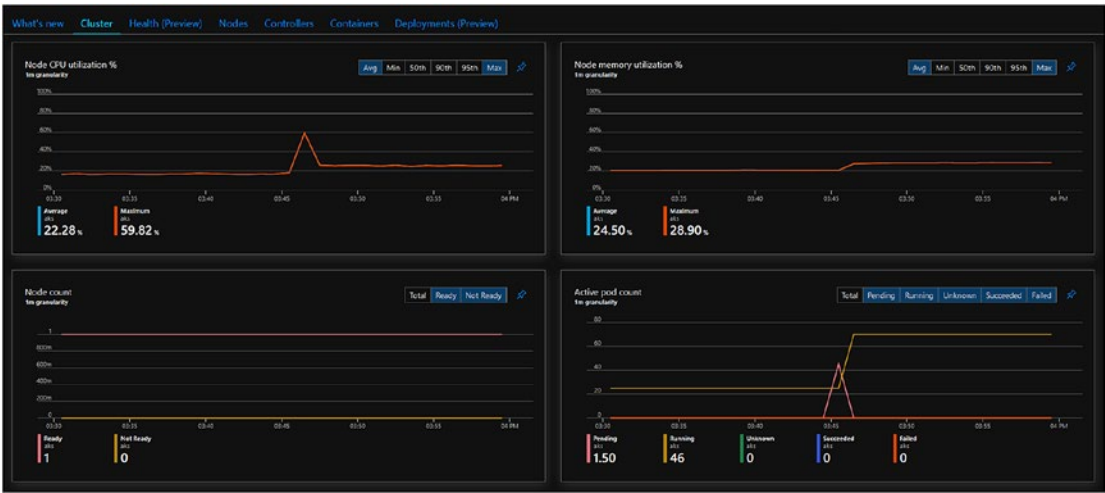


Figure 8-4. Cluster Insights page

Using the Health view, we can drill down to each system service that is running on our cluster and see its health status.

In Figure 8-5, you can see the Health status of all the Kubernetes infrastructure core services, which include API Server, CoreDNS services, and more.

If you click on each service, you can drill down further and get more information about the health of the service.

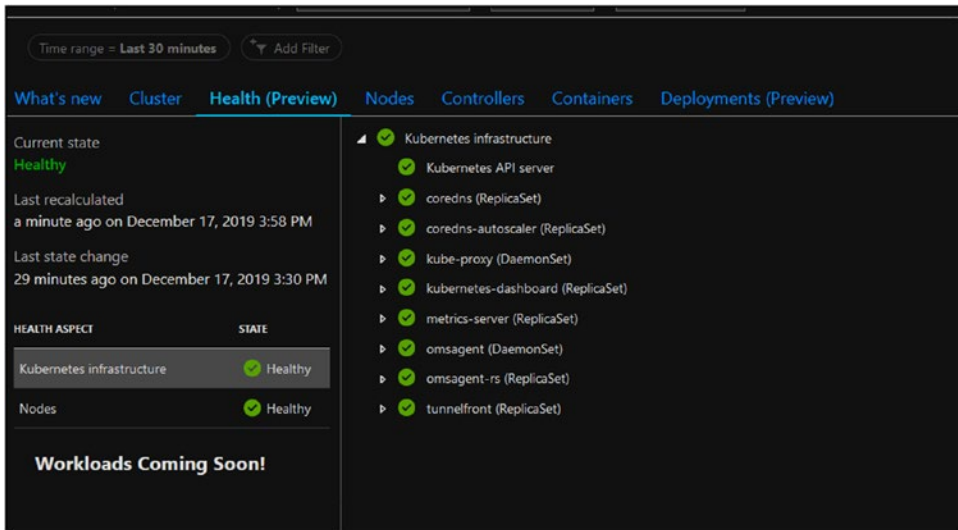


Figure 8-5. *Kubernetes infrastructure health*

We can also monitor AKS nodes and node pools using Insights and get valuable information about our nodes. If you look at Figure 8-6, you will see the Insights Nodes section.

You can see that I have 71 containers running on my node pool, which is made up of a single node. The uptime is 46 minutes, and CPU usage is just 26 percent, which mean I have room to grow without affecting performance.

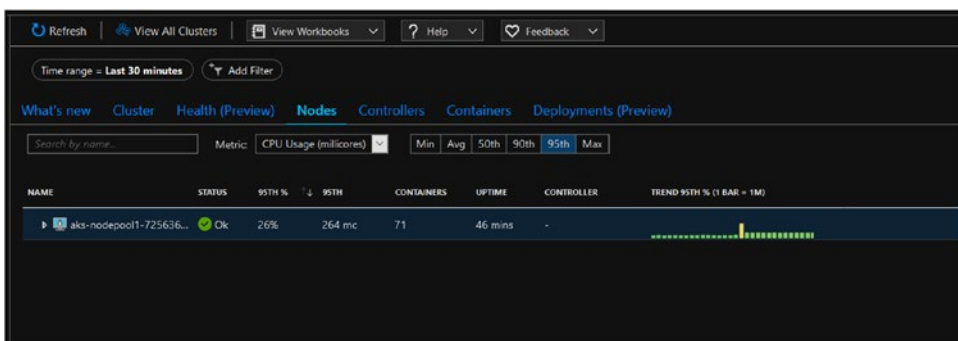


Figure 8-6. *Node performance*

We will move now to the Containers tab, where Azure Monitor gives us deep insights into our containers' and pods' performance. We can monitor the health of each pod and get average CPU usage, pod details, node pool that the pod is running on, and uptime.

Figure 8-7 shows the Containers monitoring screen.

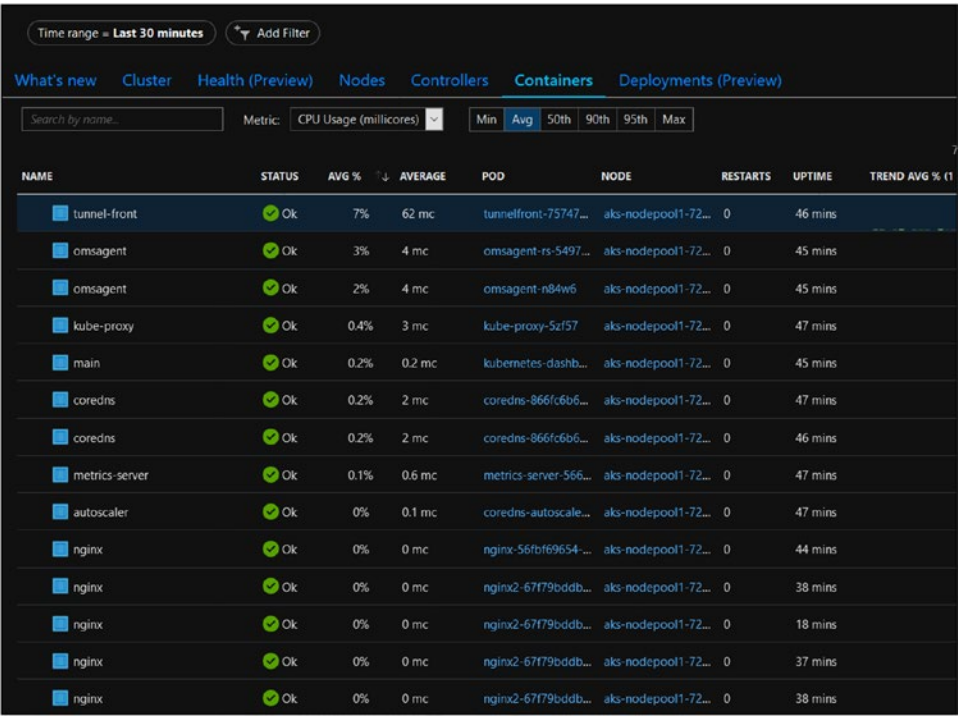


Figure 8-7. Containers

The best part about the container monitoring solution is that we can get real-time data. If you click on the container name, you will see the Container menu on the right-hand side, as shown in Figure 8-8.

The live data options give us access to advanced diagnostic solutions and give us direct access to the AKS cluster container logs without logging in to the CLI or the UI. Live data is similar to running the `kubectl get events` and `kubectl top pods`.

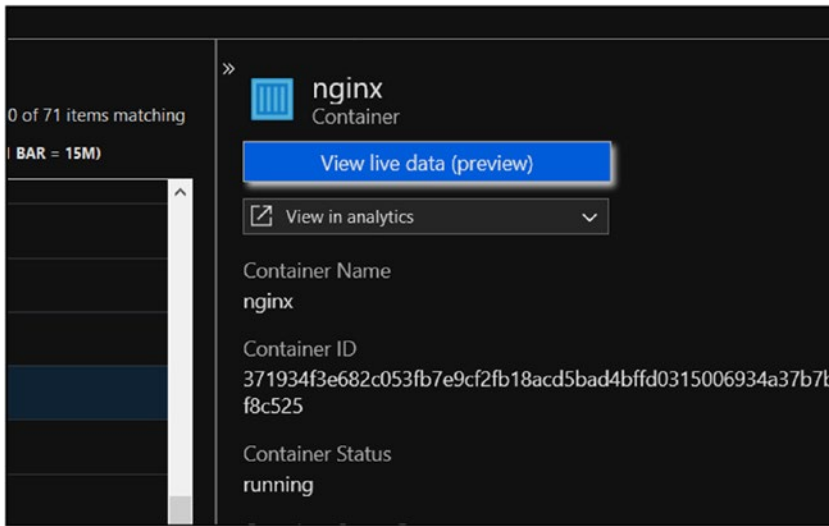


Figure 8-8. *View live data*

Metrics

If you remember from the monitoring introduction, Azure also collects metrics data about the performance of the cluster.

Azure monitoring matrices are good when we need to work with numerical values; for example, let's say we want to know how many pods we have running on our AKS infrastructure. In Figure 8-9, I am using metrics to find out how many containers I have running on my AKS cluster.

This kind of information is very useful, and currently Azure Monitor previews a live data feed that allows us to view metrics information in real-time.

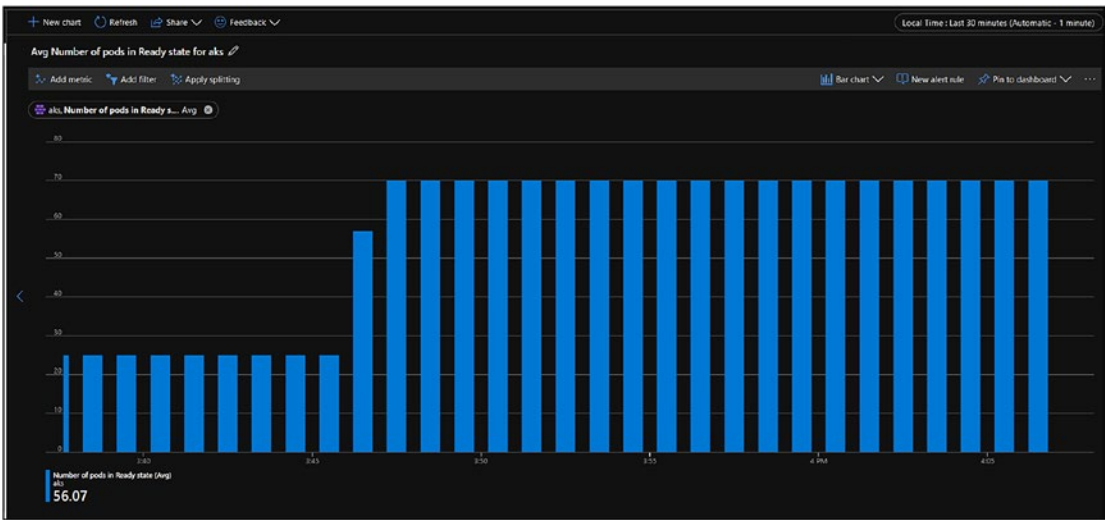


Figure 8-9. *Number of pods ready*

In Figure 8-9, I am using a bar chart to display the number of pods on my cluster; however, this is not the default view.

To change the default view, please use the view selector option on the top-right corner, as shown in Figure 8-10.

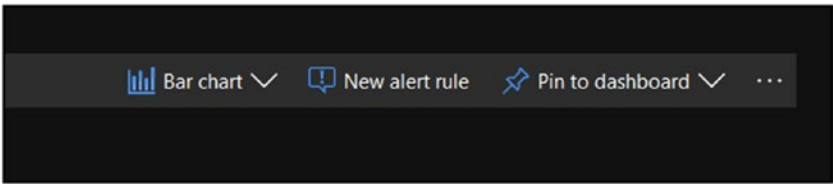


Figure 8-10. *Bar chart*

You can also pin a specific view to your Azure dashboard for quick access to that information.

You can also create email alerts or SMS alerts out of the Metrics view. If you click on the “New alert rule” button that is shown in Figure 8-10, you will be able to create an alert based on the view.

For example, I can set up an alert that sends me an email or takes action when there are more than 100 pods running on the AKS infrastructure.

In Figure 8-11, you can see the rule configuration, which is easy to use and very straightforward. Please note that alerts cost money; you can see the monthly costs associated with the alert.

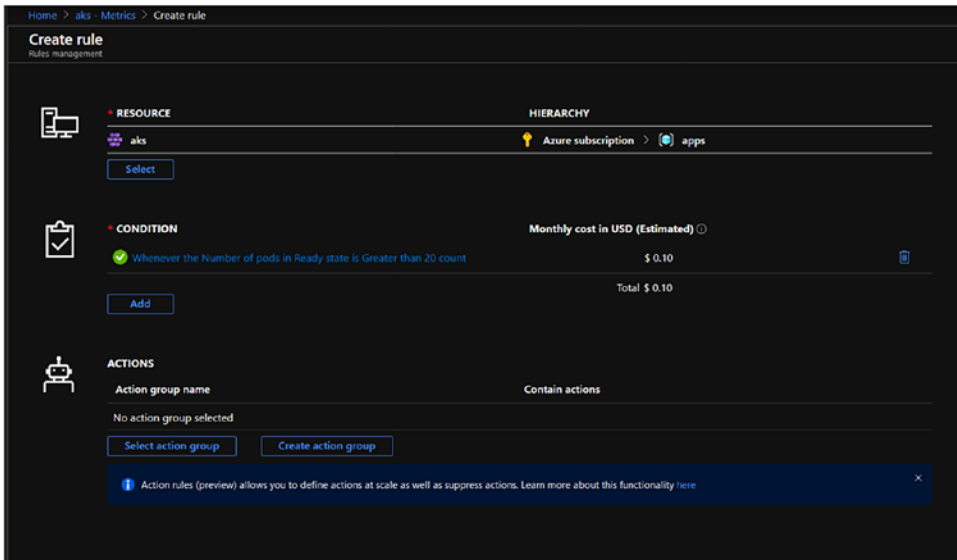


Figure 8-11. *Create alert rule*

Logs

The last thing I will show you in this section is how to access Azure Monitor logs. To use the Azure Monitor logs functionality, we need to enable a Log Analytics workspace.

By default, AKS will create one for us, so there is no need to configure one.

When you first click on Logs, you will see the Welcome to Log Analytics screen, as shown in Figure 8-12.

Click on “Get Started” to access it.

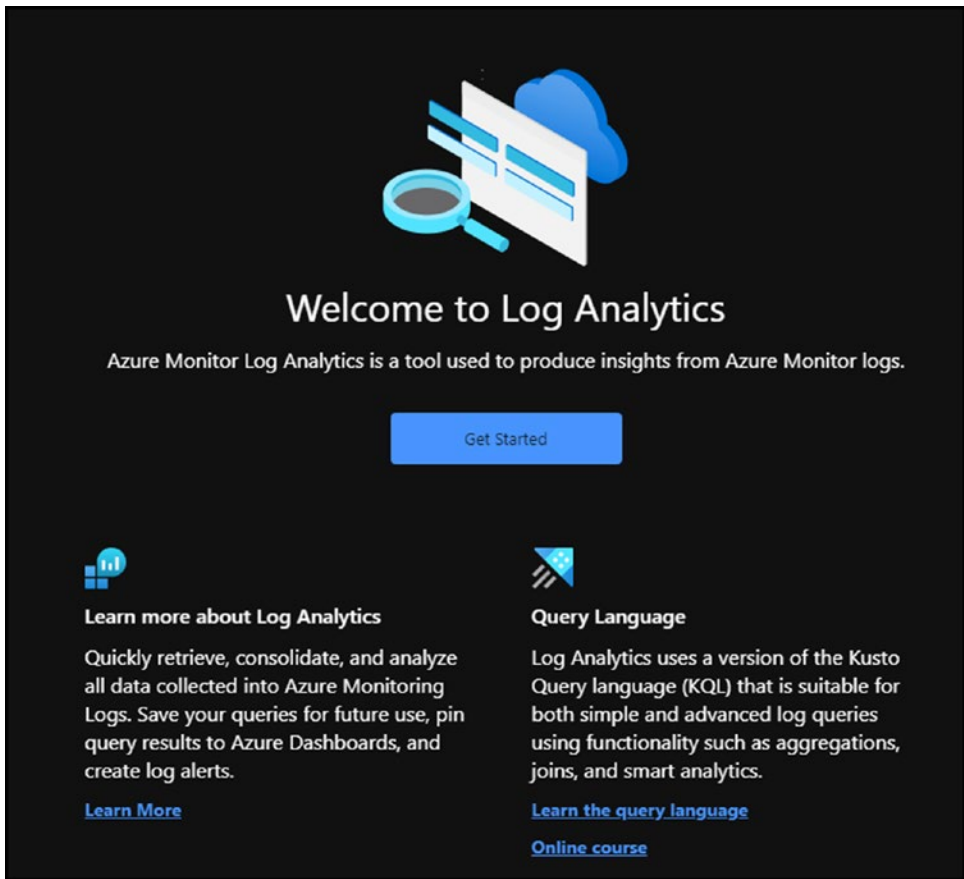


Figure 8-12. *Welcome to Log Analytics*

Log Analytics allows us to search, filter, and sort schema data (tables) that comes from our Kubernetes infrastructure and dig deep into our AKS operations.

To access our logs in Log Analytics, we use a query language to apply search terms and discover information. Under the hood, Azure Log Analytics uses KQL, which stands for Kusto Query Language. Kusto allows us to run read-only queries against data that sits inside tables and extract the information we need.

If you followed the deployment steps in Chapter 4 and created an AKS cluster, you have Log Analytics configured.

Note Kusto is a case-sensitive language.

In Figure 8-13, I have my Log Analytics screen open. On the schema pane, you can see all the AKS-related tables, which contain data that belongs to my AKS cluster and containers.

To run a simple query that looks for all my Nginx-related containers, I will type the following query in the query editor:

```
Containerlog
| where name contains "nginx"
```

In the code example, I am searching inside the `containerlog` table and running a `where` statement that looks for a container name that contains `nginx`.

The pipe sign `|` separates commands.

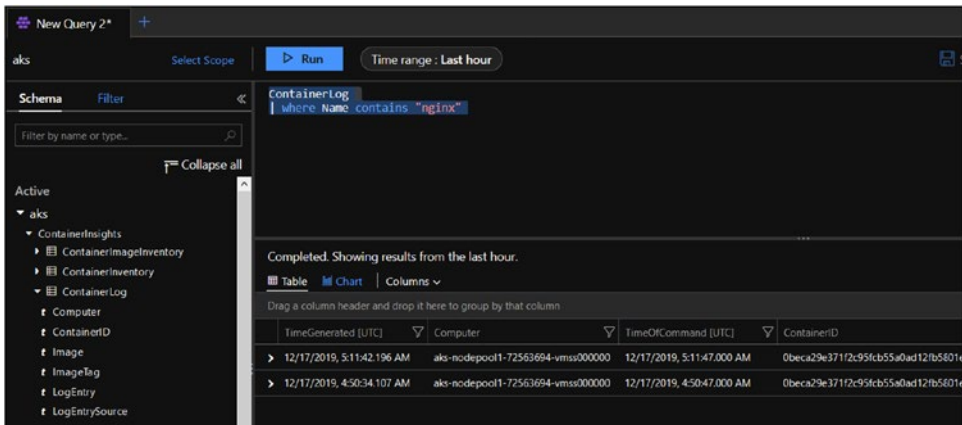


Figure 8-13. Log analytics query editor

We can also use a search query that searches for a general term, as follows:

```
search in (containerlog) "nginx"
| limit 10
```

The `limit` command returns only ten records for the results.

Disable Azure Monitor for Containers

If you would like to disable Azure Monitor for Containers and stop monitoring your cluster using the advanced monitoring solution, you can run the following Azure CLI command, which will disable the feature.

Don't forget to change the cluster name and resource group name.

```
az aks disable-addons -a monitoring -n aks -g apps
```

The next thing you need to do is to delete the Log Analytics workspace that belongs to the Azure Monitor for Containers service. In the last section of this chapter, I have included this step, which is the same for Azure Monitor for Containers and Azure Monitor for VMs. You will find the steps in the “Delete Log Analytics Workspace” section.

Monitor Azure Container Instances (ACI)

In this section, I will show you how to monitor your Azure Container Instances (ACI) using Azure Monitor.

If you recall from Chapter 3, we learned about ACI, including that it has some limitations. When it comes to monitoring, ACI comes with basic monitoring capabilities, but none of them offer deep insights like AKS does.

If you open the ACI main page, you will see on the left pane, as shown in Figure 8-14, that ACI offers two monitoring options:

1. Metrics
2. Alerts

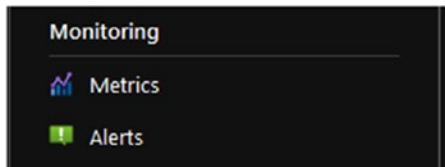


Figure 8-14. ACI monitoring

Metrics give us information about CPU, memory network usage, and more.

Because ACI is a service that offers fast deployments of containers, the monitoring solution is designed to help us get information that is suitable for this kind of workload.

Figure 8-15 shows the overview page of an ACI deployment; it shows useful performance information about the deployment.

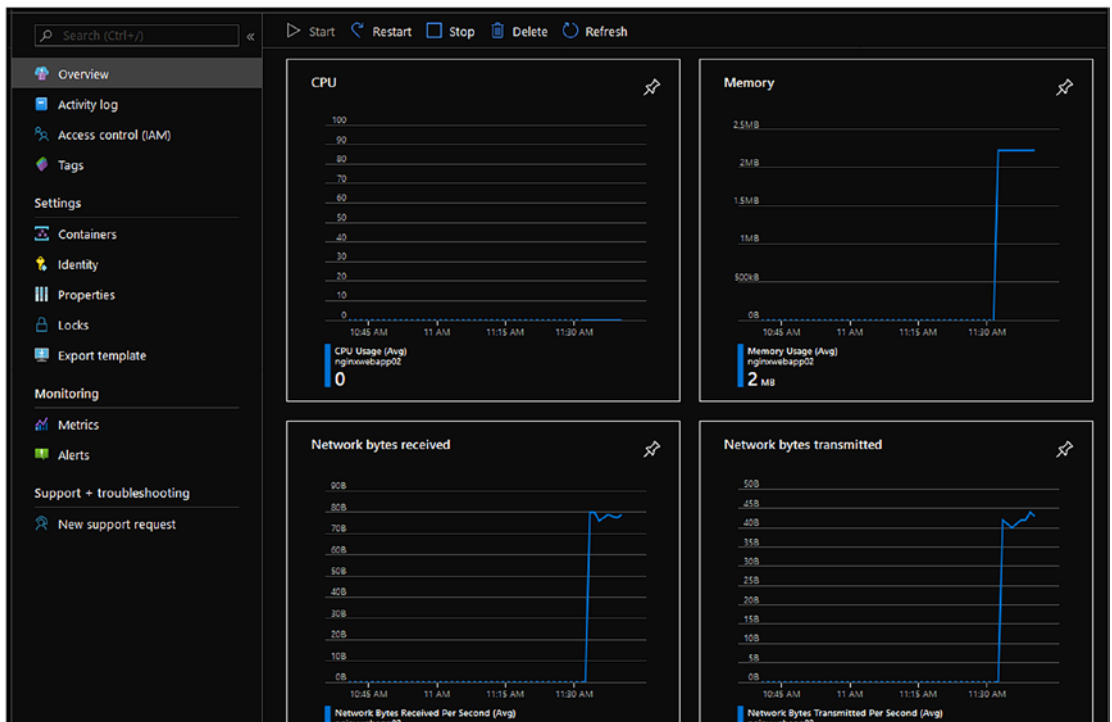


Figure 8-15. ACI overview

If you click on Metrics in the left-hand menu, you will see all the options Metrics has to offer.

With metrics, we can get deeper information from our ACI deployments, including more detailed CPU, memory, and network performance information.

Using metrics with ACI is the same as using it with AKS in that you can change the chart view, dates, and more.

In Figure 8-16, you can see the memory usage of my ACI deployment using metrics.

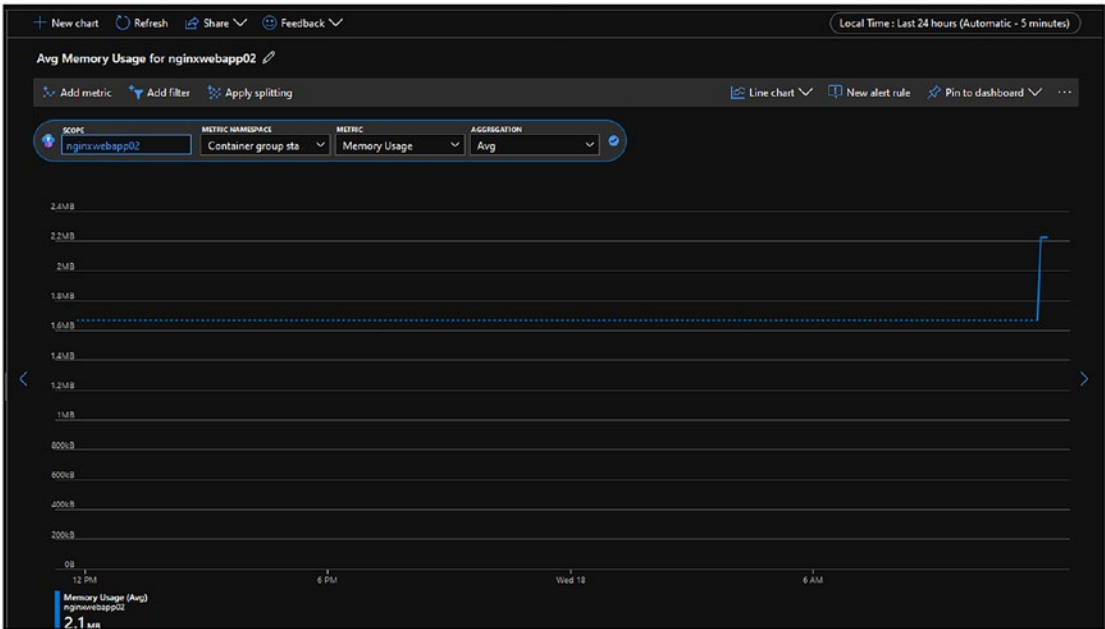


Figure 8-16. ACI metrics

Monitor ACI with Azure CLI

Up until now, I have focused on teaching you how to monitor ACI using the Azure portal, without touching Azure CLI.

With Azure CLI, ACI module, we can tap into the monitoring information we see in the portal and monitor deployments from the command line. This option is good when you have a script of some sort that takes action based on the performance of the deployment.

In the following example, I will show you how to use Azure CLI to monitor ACI.

Note This code is available under the name `8.1.Monitor_ACI.sh`.

We start by defining our ACI container group's resource group and container group names; in my case, the resource group name is `apps`, and the container group is `nginxwebapp02`.

```
CONTAINER_GROUP_NAME=$(az container show --resource-group apps --name
nginxwebapp02 --query id --output tsv)
```

To view CPU usage information, I will run the following code:

```
az monitor metrics list --resource $CONTAINER_GROUP_NAME --metric CPUUsage  
--output table
```

To view memory usage information, I will run the following code:

```
az monitor metrics list --resource $CONTAINER_GROUP_NAME --metric  
MemoryUsage --output table
```

You can view the same information with more details using the following code:

```
az monitor metrics list --resource $CONTAINER_GROUP_NAME --metric  
MemoryUsage --dimension containerName --output table
```

To view the container group logs, I will run the following command:

```
az container logs --resource-group apps --name nginxwebapp02
```

Monitor Azure Container Registry (ACR)

In this section, I will talk about monitoring Azure Container Registry (ACR). At its core, ACR is not a complicated product, because Microsoft Azure takes care of all the moving parts and gives us all the information we need to connect to the service.

When it comes to ACR, our monitoring options are limited simply because we don't need many of them. ACR offers the following monitoring options:

1. Storage usage
2. Metrics (in preview mode)
3. Logs

In Figure 8-17, you can see my ACR Overview page and the storage usage of my repositories. I am only using 0.07 GB of data out of the 10 GB repository storage available.

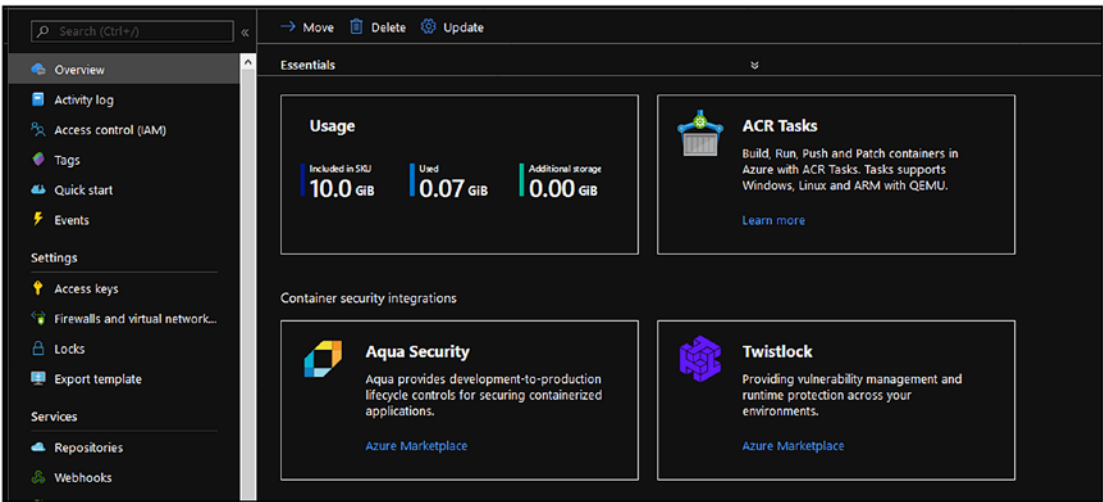


Figure 8-17. ACR usage information

ACR offers very useful metrics for monitoring and statistics about our registry. For example, in Figure 8-18, I am using metrics to find out how many pull requests my repository is receiving in one hour.

Currently, ACR metrics can provide us with the following information:

- Successful pull requests counter
- Successful push requests counter
- Total pull requests
- Total push requests

For each metric, we can get the total sum, minimum, maximum, and average values.

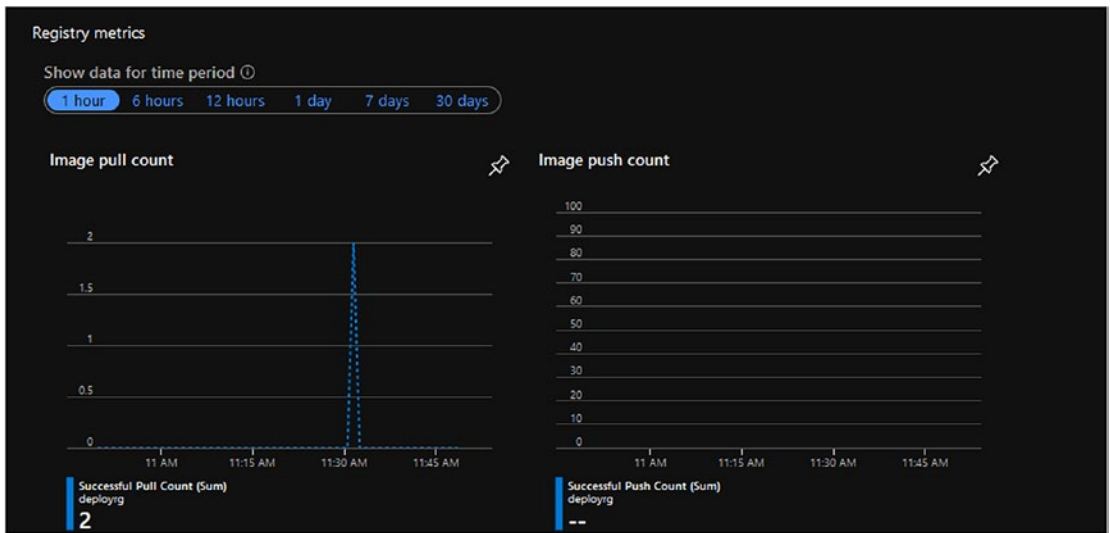


Figure 8-18. Total pull requests

In Figure 8-19, you can see the number of successful pull requests, which stands at 63 requests.

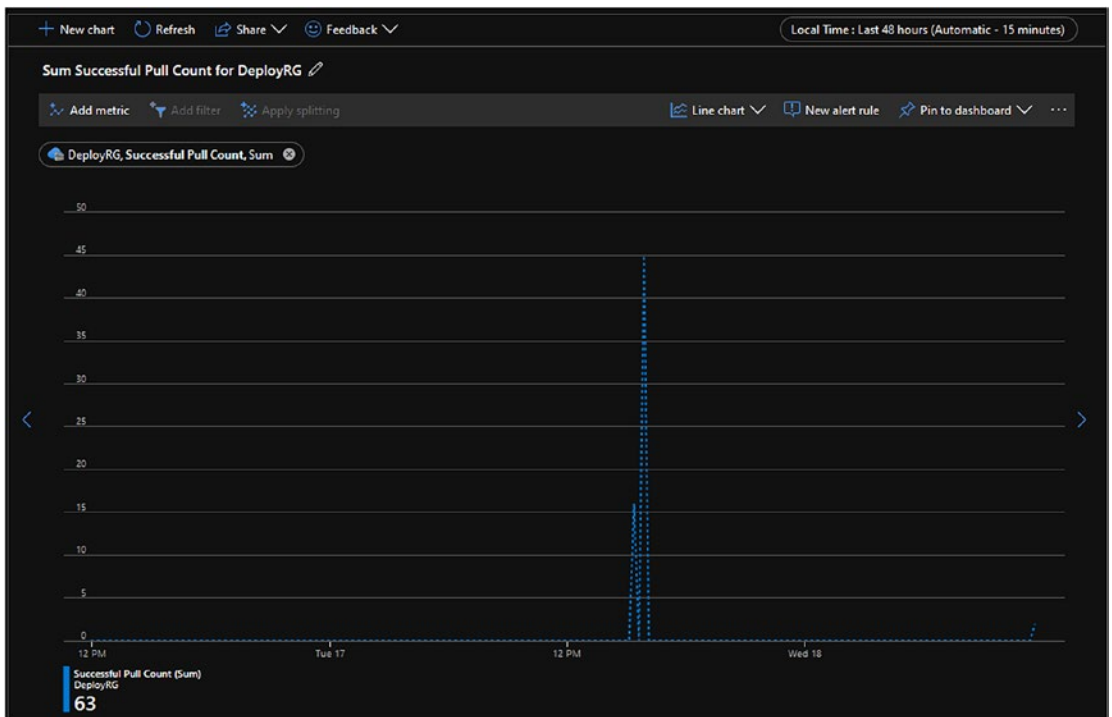


Figure 8-19. Total successful pull requests

Monitor Azure Docker Container Host VM

In this section, we will cover Azure Monitor for Virtual Machines and all the monitoring features that are available for us.

Azure Monitor for Virtual Machines offers comprehensive multi-layer monitoring tools for VMs that cover the following:

- Performance analysis of VM health

- Monitoring of processes

- Application monitoring

You can access Azure Monitor for VMs from any VM running in Azure. In Figure 8-20, you can see the Azure Monitor offerings from the Monitoring pane.

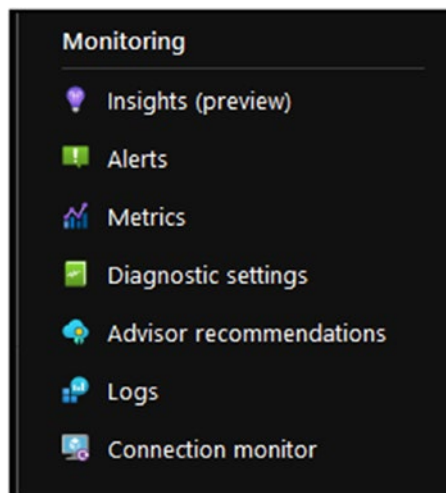


Figure 8-20. *Azure Monitor for VMs options*

By default, Azure offers impressive quick monitoring views via the VM Overview page, which gives us a snapshot of the following:

- CPU usage

- Network usage

- Disk usage

In Figure 8-21, you can see the Overview page of the VM with all the performance counters.

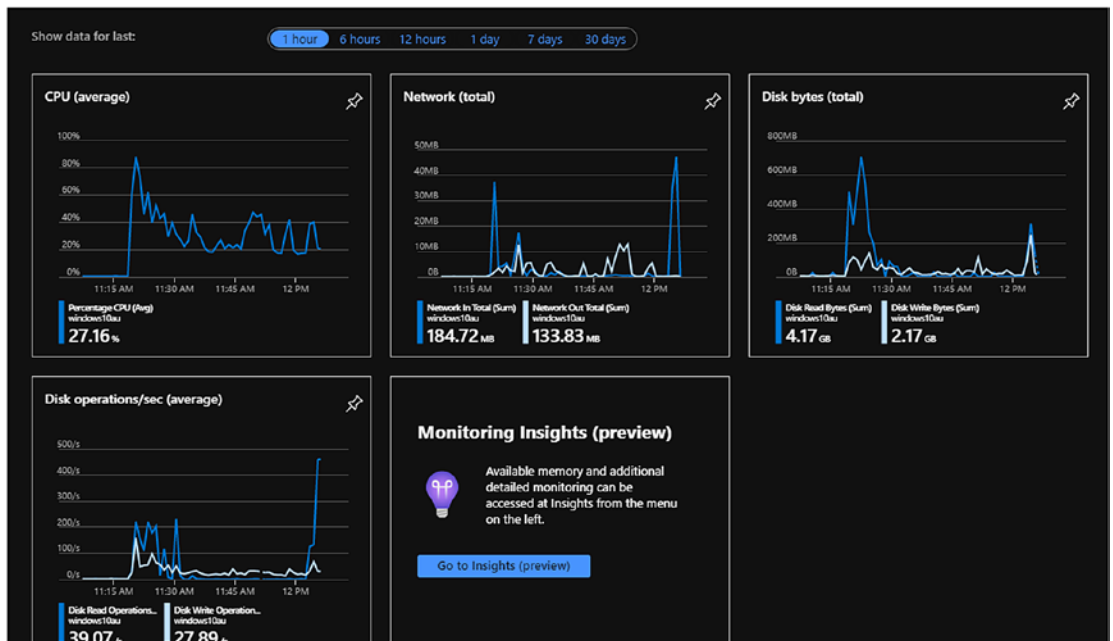


Figure 8-21. Azure Monitor for VMs

To access more performance counters, click on Metrics in the Monitoring menu in the left pane. Metrics is a free service provided by Azure.

Azure Monitor Insights for VMs

If you need deep insights into the performance of your VMs, you should use Azure Monitor Insights. To enable advanced monitoring capabilities using Insights, perform the following steps:

From the Azure VM page, click on “Insights (Preview).”

From the Enable Insights page (Figure 8-22), select the subscription and workspace to send the logs to.

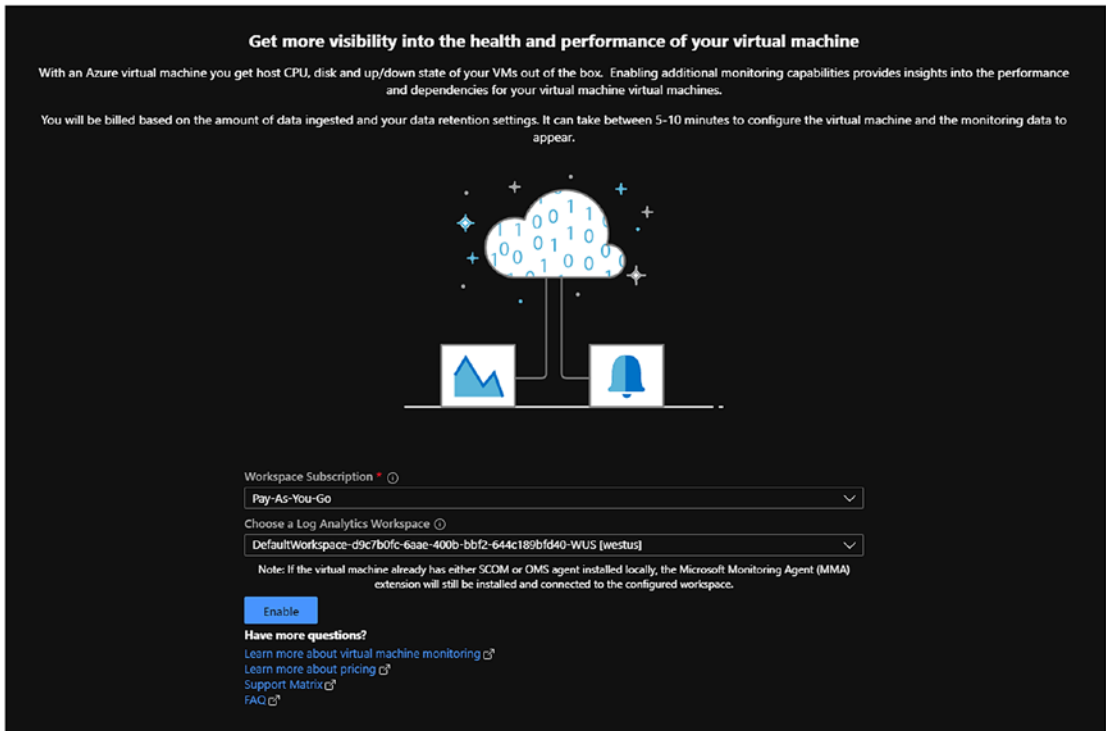


Figure 8-22. Enable Azure Insights on a VM

Logical Map

Fifteen minutes after Insights is enabled on my VM, it will display very detailed information about my VM. For this example, I have enabled Insights on a production web server.

As shown in Figure 8-23, Insights displays a logical map of my web server. In the map, you can see the following:

- A number of clients connected to the server — In my case, I have 39 users connected to the server. If I click on the clients, I will see the IP addresses they are coming from.
- The server details; in my case, it is an IIS server
- The number of processes running on the server — in my case, I have 26 processes, and if I click on the server, I can see the details.
- Open ports on the server and number of websites running; in my case, I have ports 80, 123, 443.

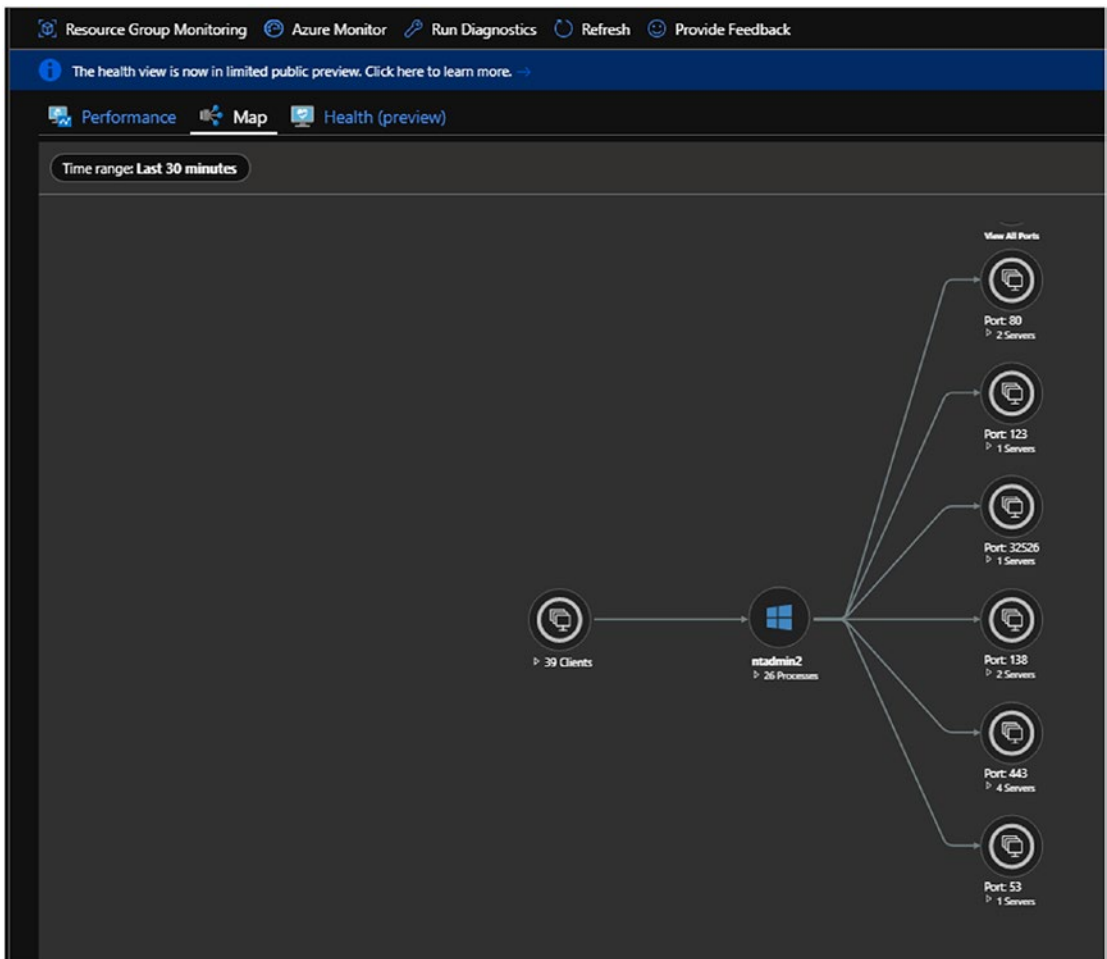


Figure 8-23. *Logical map*

Properties

On the right-hand side of the map, you will see the Properties page, as shown in Figure 8-24. The Properties page shows useful information about the VM, including IP addresses, operating system details, dependencies, health, and Azure VM details.

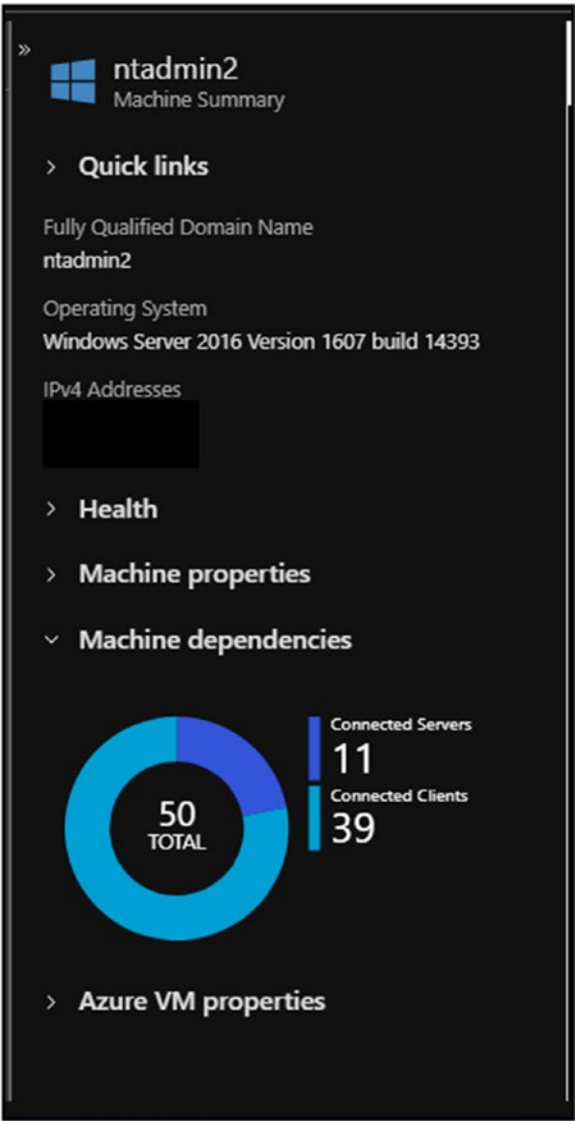


Figure 8-24. VM properties

If you click on the “Performance” button located in the top menu, you will see detailed information about the overall performance counter of the VM.

For example, in Figure 8-25, you can see the Performance page with logical disk performance. If you scroll down, you will see the following performance counters:

- CPU Utilization %
- Available Memory

- Logical Disk IOPS
- Logical Disk MB/s
- Logical Disk Latency (ms)
- Max Logical Disk Used %
- Bytes Sent Rate
- Bytes Received Rate

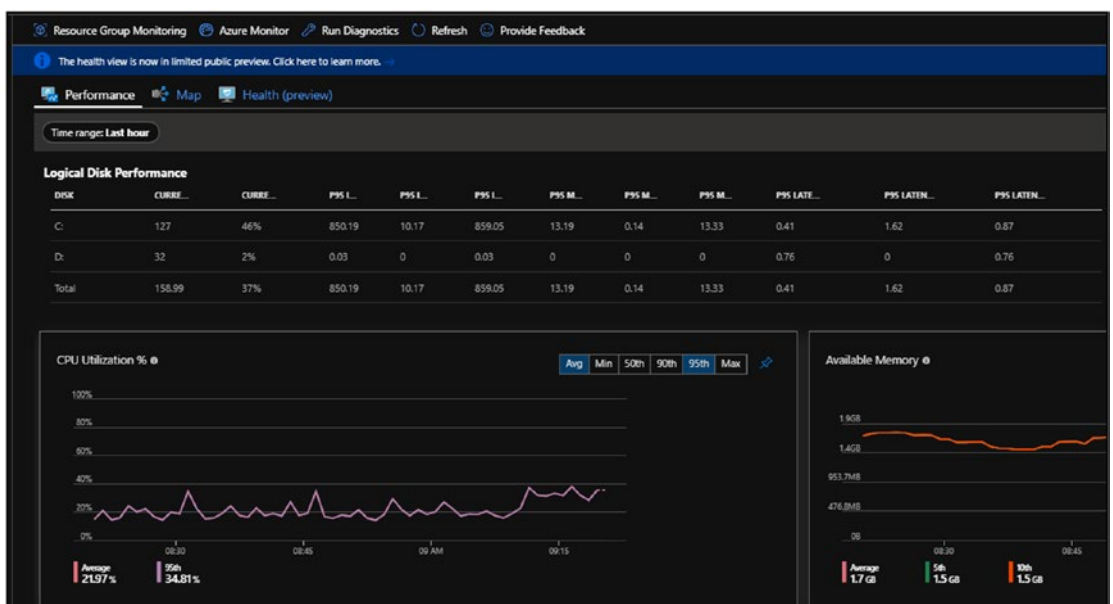


Figure 8-25. Performance counters

Log Events

Going back to the Insights screen on the left-hand menu, you can access the Log Events screen. In Figure 8-26, you can see all the events that are available for the VM.

If you click on a specific event type, you will be redirected to Log Analytics. I will go ahead and click on the Perf logs.

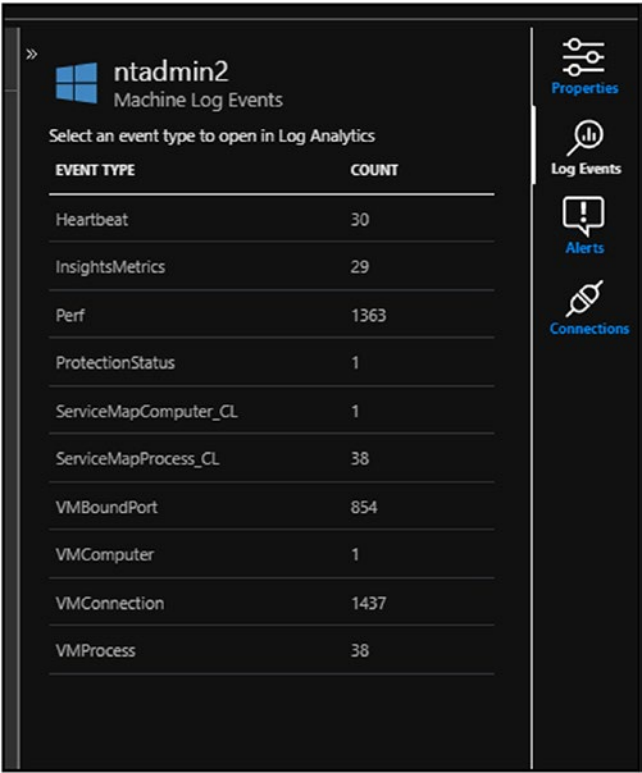
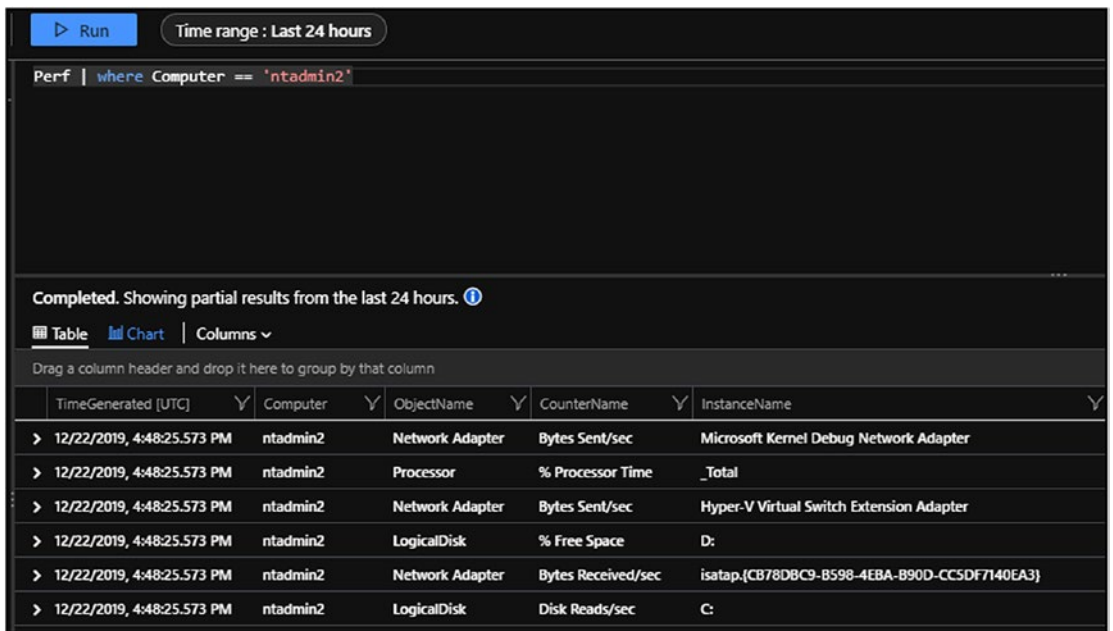


Figure 8-26. Log Events

From the Log Analytics page, I can use the Query Editor page to write queries, or use the default one that is loaded after clicking on the log (Figure 8-27).



The screenshot shows the Azure Monitor Perf query editor. At the top, there is a 'Run' button and a 'Time range : Last 24 hours' selector. Below this, the query editor shows the query: `Perf | where Computer == 'ntadmin2'`. The results are displayed in a table view, showing partial results from the last 24 hours. The table has columns: TimeGenerated [UTC], Computer, ObjectName, CounterName, and InstanceName. The results show performance data for various components on the 'ntadmin2' computer.

TimeGenerated [UTC]	Computer	ObjectName	CounterName	InstanceName
12/22/2019, 4:48:25.573 PM	ntadmin2	Network Adapter	Bytes Sent/sec	Microsoft Kernel Debug Network Adapter
12/22/2019, 4:48:25.573 PM	ntadmin2	Processor	% Processor Time	_Total
12/22/2019, 4:48:25.573 PM	ntadmin2	Network Adapter	Bytes Sent/sec	Hyper-V Virtual Switch Extension Adapter
12/22/2019, 4:48:25.573 PM	ntadmin2	LogicalDisk	% Free Space	D:
12/22/2019, 4:48:25.573 PM	ntadmin2	Network Adapter	Bytes Received/sec	isatap.{CB78DBC9-B598-4EBA-B90D-CC5DF7140EA3}
12/22/2019, 4:48:25.573 PM	ntadmin2	LogicalDisk	Disk Reads/sec	C:

Figure 8-27. Query editor for Perf logs

Alerts

I can also manage and view alerts that are related to the VM directly from the Insights screen, as shown in Figure 8-28. In my case, I have no outstanding alerts, but if I had any, they would have shown up in the Alerts screen.

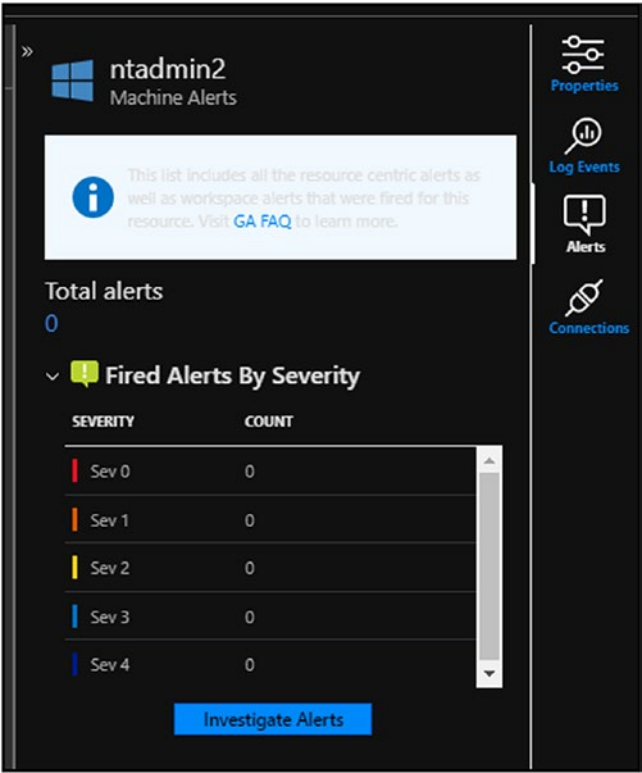


Figure 8-28. Alerts

Connections

And, lastly, Insights allows us to view all active connections to the VM using the Connections menu, as shown in Figure 8-29.

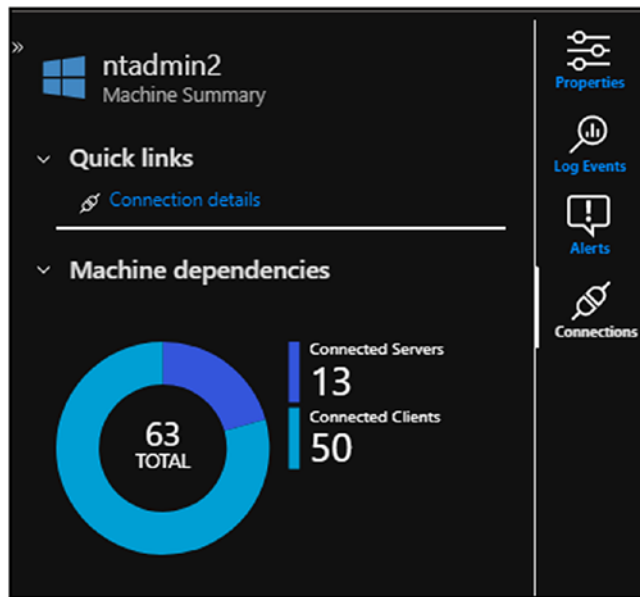


Figure 8-29. *Connections*

If you click on the “Connection details” link, you will be redirected to the Connection Overview page, which displays detailed information about each connection (shown in Figure 8-30). In my case, you can see the number of connections and whether they are malicious or not, response time, total bytes, and more details.

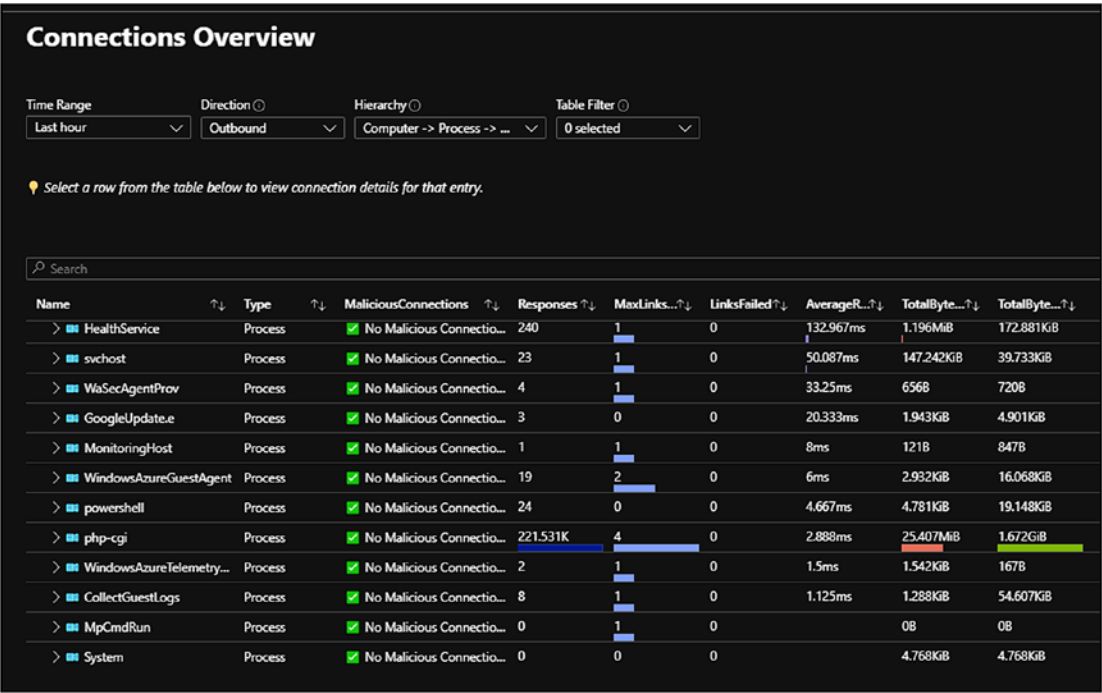


Figure 8-30. Connections Overview page

If you click on each connection, you will see the external IP details of the connected client and the port he or she is using.

On the bottom half of the page, you will see the overall information regarding all the connections and the response time, latency, network, and links (Figure 8-31).



Figure 8-31. Overall connection overview

Connection Monitor

Azure Monitor for VM comes with another cool feature called Connection Monitor, which allows the monitoring of the network connection between two virtual machines.

To deliver this feature, Azure uses a Network Watcher utility that creates the connection between the VMs and reports back using alerts if the connection is down or not stable. To create a Network Watcher for my VM, I will click on “Connection monitor” in the Monitoring menu, as shown in Figure 8-32.

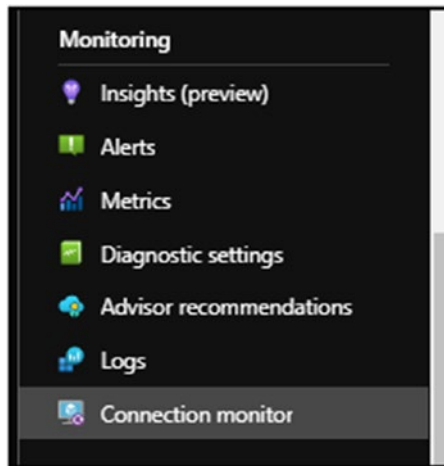


Figure 8-32. *Connection monitor*

From the Connection Monitor page, I will click on “Add.” From the Add Connection Monitor page, I will need to select source and destination virtual machines.

In Figure 8-33, you can see the menu, and you will see that the VM can reside in two different subscriptions and resource groups. I can also use an external VM and specify it manually using an IP address or host name.

In the target port, we need to use a port that is open; in my case, I know that port 3389 is open, and so that’s the port I’m going to use.

Add connection monitor

Name *

Source

Subscription * ⓘ

azuresub

Virtual machine *

Select a source virtual machine

Destination

☒ Select a virtual machine ☐ Specify manually

Virtual machine ⓘ

You must select a source before selecting a destination.

✖

Port * ⓘ

Advanced settings

Add

Figure 8-33. Add connection monitor

After enabling the Network Watcher, within a few minutes, the statistics data will show up. In Figure 8-34 you can see the results of my watcher.

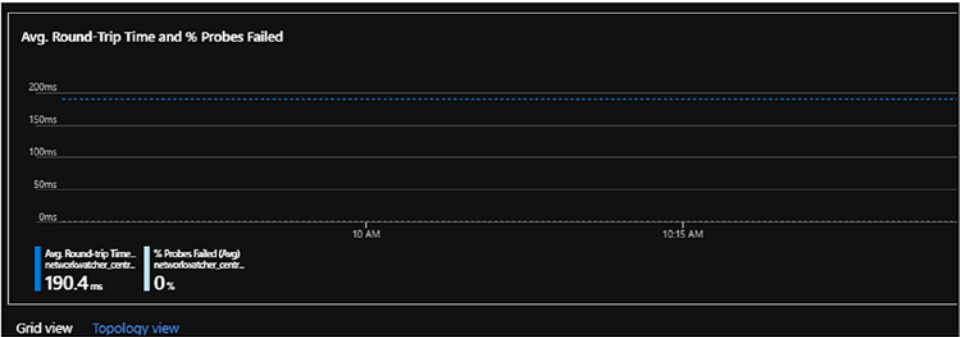


Figure 8-34. Network Watcher

Create a Network Watcher Alert

After creating the watcher, I can go ahead and create a corresponding alert that will let me know if the connection between the VMs breaks.

To create an alert, from the Monitoring menu on the left-hand pane of the VM, I will click on “Alerts.” From the Alerts page, I will click on “New alert rule.” To find the Network Watcher, I will click “Select” under the Resource option, as shown in Figure 8-35.

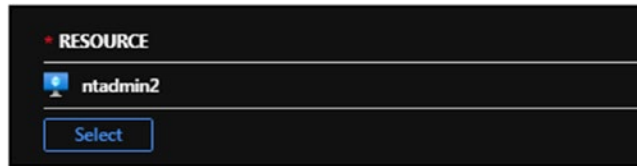


Figure 8-35. *Select resource*

From the Select a Resource page, I will select the Azure subscription the watcher was created under and select the Network Watcher from the “Filter by resource type” drop-down menu, as shown in Figure 8-36.

Once selected, I will modify the action and alert settings.

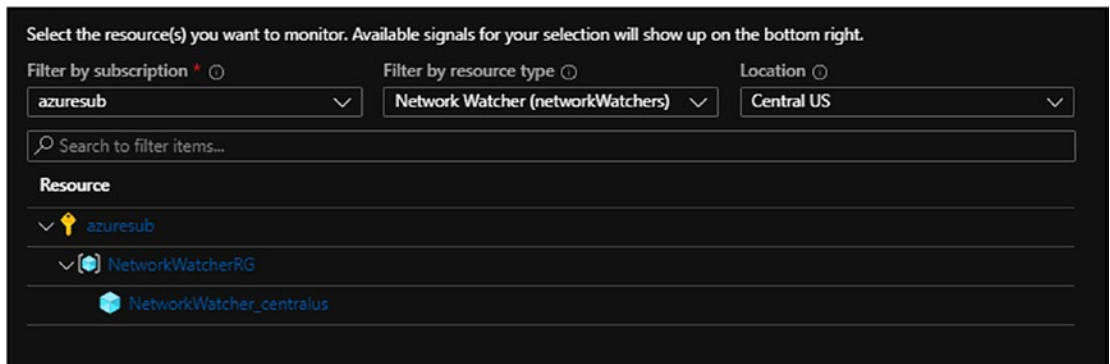


Figure 8-36. *Select a resource*

Disable Azure Monitoring for VMs

The last thing I will show you in this chapter is how to disable Azure Monitor for VMs.

If you no longer need the solution or are trying to optimize the cost of your tenant, you can disable monitoring and stop paying for it. Disabling Azure Monitor for VMs is a two-step process. First, we need to disable the monitoring extensions, and then we need to delete the Log Analytics workspace.

Disable Monitoring Extensions

To disable the monitoring extensions, in the virtual machine’s Settings menu in the left-hand pane, click on “Extensions,” as shown in Figure 8-37.

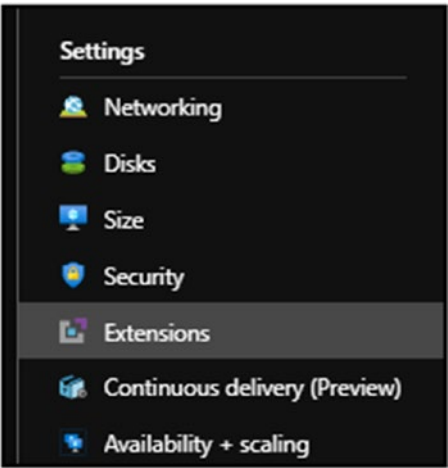


Figure 8-37. *Extensions*

From the Extensions page, click on the following extensions and uninstall them, as shown in Figures 8-38 and 8-39.

AzureNetworkWatcherExtention
DependencyAgentWindows

+ Add				
Search to filter items...				
Name	↑↓ Type	↑↓ Version	↑↓ Status	
AzureNetworkWatcherExtension	Microsoft.Azure.NetworkWatcher.NetworkWatcherAgentWindows	1.*	Provisioning succeeded	
DependencyAgentWindows	Microsoft.Azure.Monitoring.DependencyAgent.DependencyAgentWindows	9.*	Provisioning succeeded	

Figure 8-38. *VM extensions*

From the Extensions page, click “Uninstall,” as shown in Figure 8-39.

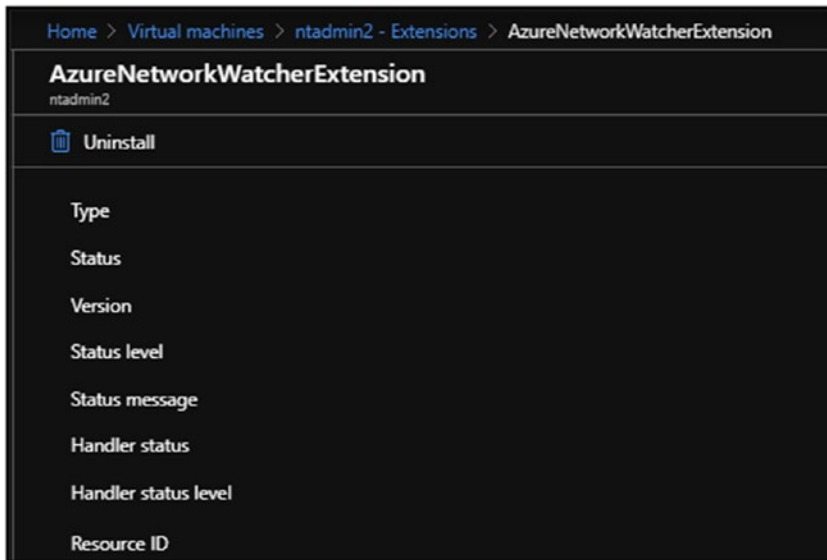


Figure 8-39. *Uninstall extension*

Delete Log Analytics Workspace

Next, I will delete the Log Analytics workspace that belonged to Azure Monitor for VMs.

From the Log Analytics page, I will select my Log Analytics workspace from the list of workspaces, as shown in Figure 8-40.

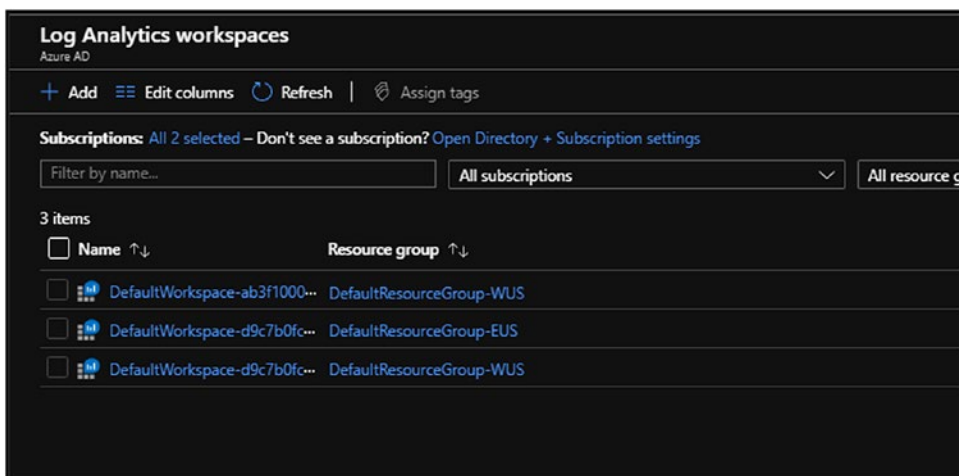


Figure 8-40. *Log Analytics workspaces*

To delete the workspace, I will click on the “Delete” button, as shown in Figure 8-41.

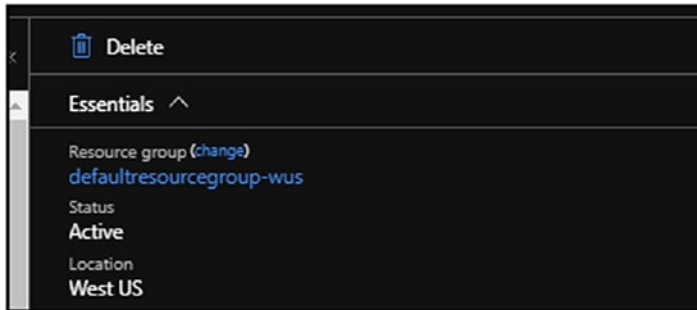


Figure 8-41. Delete workspace

Summary

In this chapter, we learned how to monitor our containers’ infrastructure using the built-in Azure monitoring tools and Azure Monitor paid services. The features I recommend you spend some time mastering are as follows:

- Azure Monitor for VMs
- Azure Monitor for Containers
- Kusto query language

Both services deliver advanced capabilities that can help you better manage and monitor the performance of your VMs and containers. For your convenience, I have included steps on how to disable monitoring if you no longer need it and would like to stop paying for the advanced monitoring solutions.

CHAPTER 9

Back Up and Restore Containers and Containerized Applications on Azure

In this chapter, we will learn about Azure backup and disaster-recovery features and how they can help us manage our containerized applications and services. One of the best practices when designing a production environment, large or small, is to have a backup solution in place. Backup and restore are two of the most critical components of every production environment, and sometimes they are forgotten until something bad has already happened. In this chapter, I will share with you the backup strategies and tools Azure has to offer that will help protect your workload in case of a disaster.

Remember, every production environment needs a backup regardless of whether you are running one container or one hundred.

The following topics will be covered in this chapter:

- Azure Backup solutions overview
- Setting up Azure recovery services vault
- Backing up AKS data
- Backing up ACI workloads
- Backing up ACR
- Backing up virtual machines

Azure Backup Solutions

As of the time of writing, Microsoft Azure offers the following two backup and disaster-recovery services:

1. Azure Backup
2. Azure Site Recovery

Both services are similar and overlap one another but are used for different scenarios.

Azure Backup

Azure Backup is your traditional backup and recovery solution that allows you to back up the following resources:

Virtual machines

Azure file shares

SQL databases

Web apps

Azure Backup is fully integrated into the Azure platform, which allows us to back up our services from the service directly without using a secondary screen; for example, Azure Backup is integrated into the virtual machine left-hand menu pane, as shown in [Figure 9-1](#).

The easy-to-access menu allows us to use the service on the resource level and reduce complexity.

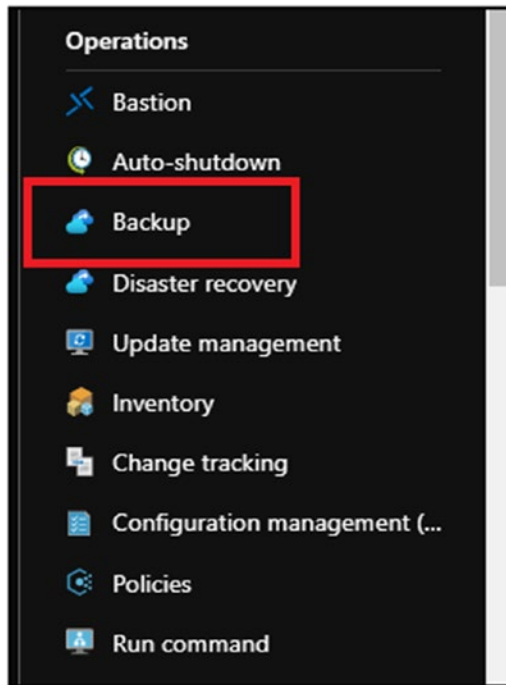


Figure 9-1. *Azure Backup*

With Azure Backup, we can create backup policies and retention policies that can keep data for weeks, months, and years. We can even use Azure Backup to back up on-premises files and VM system states using a backup agent.

All the data that Azure Backup is backing up is encrypted in transit (during data transfer) and at rest (on storage).

We can also use Azure CLI to back up our resources with Azure Backup.

Azure Site Recovery

Azure Site Recovery gives us all the features of Azure Backup plus automatic failover and replication of data between Azure regions. For example, if you have all your Azure workloads running on the WestUS region, Azure Site Recovery can copy, back up, replicate, and failover all the workloads to another Azure region in case of a disaster.

The emphasis of the service is that there is no need for manual failover or recovery of services. Azure Site Recovery can handle the entire failover process automatically in case of a planned or unplanned outage or disaster.

The service is more geared to enterprise clients because of the higher cost and configuration process.

In essence, Azure Site Recovery creates a logical disaster-recovery site in another Azure region that we choose. Once configured, Azure replicates all resources from the primary region to the disaster region in real-time. When an outage strikes, Azure can failover the primary site to the disaster site based on policies we define.

It is also important to note that Azure Site Recovery is the leader in disaster recovery as a service category, according to Gartner.

In the following list, you can see the main services Azure Site Recovery provides:

- Traditional backup and recovery solutions

- Azure virtual machine replication

- VMware and Hyper-V virtual machine on-premises replication

- Less than thirty seconds for continuous replication of VMs (Azure, Hyper-V, and VMware)

- Azure automation integration in case of a disaster

Back Up Azure Kubernetes Service (AKS)

In this section, we will learn how the backup of Azure Kubernetes Service (AKS) works and which part of the solution we need to back up.

If you remember from Chapter 4, AKS is a managed service that is managed by Microsoft. Azure looks after the master components, backend infrastructure, security, and networking. We, on the other hand, look after the containers, access, and scaling.

Because Azure is looking after the core backend components, our backup responsibility is limited to backing up our data only. When I say data, I mean the storage volumes our AKS containers are using to store data.

In Chapter 4, we created a persistent storage volume using Azure files and used it to mount a storage volume to our containers. As a refresher, I am listing the following script we used, which you can find in the code library under 4.7.Create_Volume_Claim.yaml.

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: azurefile
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: azurefile
  resources:
    requests:
      storage: 10Gi

```

In the code, we are creating a 10 GB storage volume using Azure files. Based on the preceding code, you can understand why we need to back up only our persistent data when it comes to AKS.

In case of a disaster, Azure will recover the AKS cluster’s master components. The worker nodes have no data, and only act as a computing power, so there is nothing to back up. The containers’ images are stored in ACR, and the only thing that is left is our code and the data our containers need.

Azure Recovery Services

To back up services in Microsoft Azure, we first need to set up a recovery services vault that will act as a storage entity to store our data. In our case, we will need to create a recovery services vault in the same region as our AKS Azure file share.

Create Azure Recovery Services Vault

To create a recovery services vault, perform the following steps.

Open the Azure portal and search for “Recovery services”; click on the link, as shown in Figure 9-2.

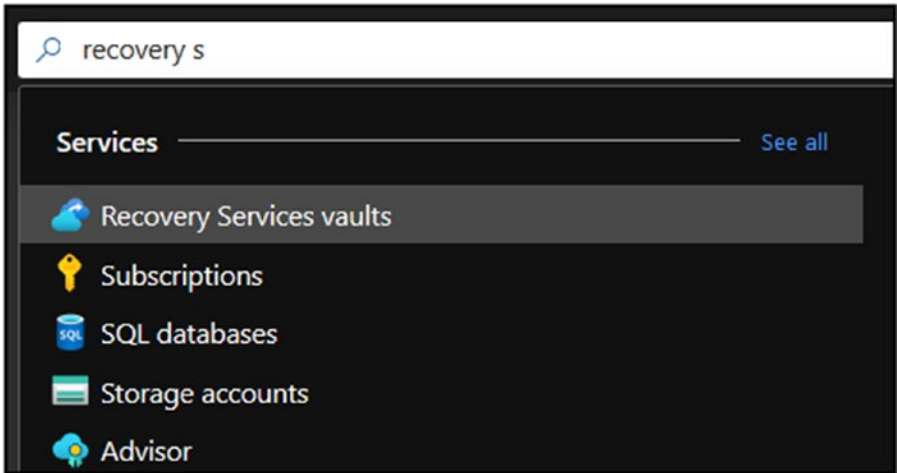


Figure 9-2. Recovery services vaults

To create a new vault, click on the “Add” button to start the creation wizard. In Figure 9-3, you can see the Create Recovery Services Vault page. Make sure you create the vault in the same region as the AKS Azure file share.

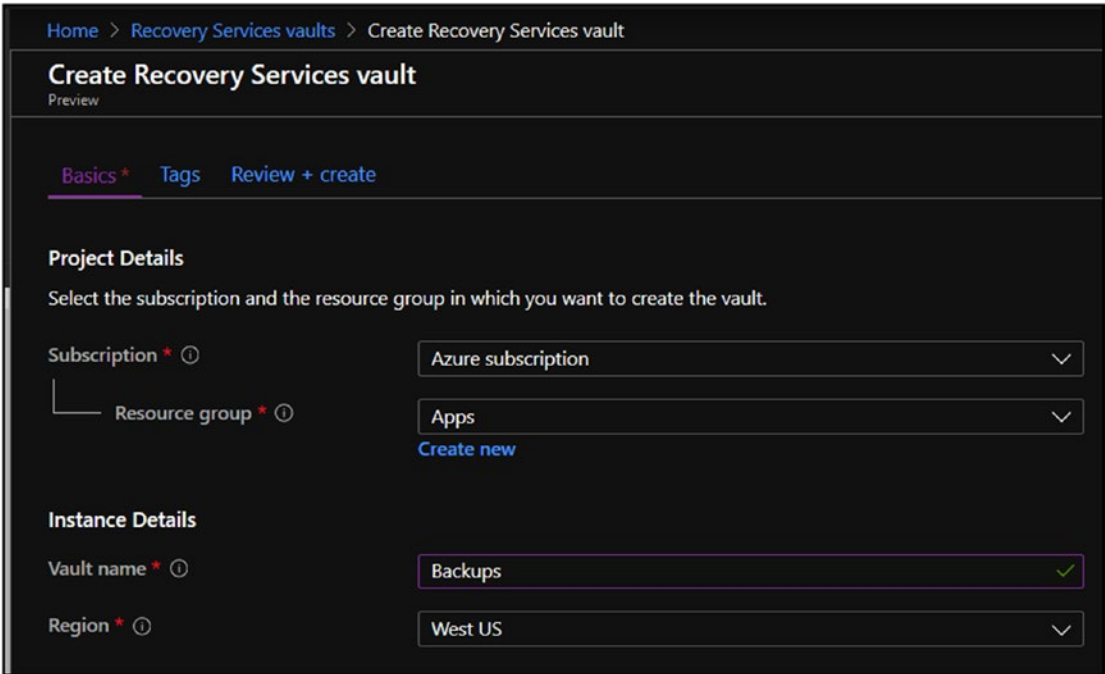


Figure 9-3. Create Azure recovery vault

It is important to note that, by default, Azure will configure the vault to replicate to another Azure location using geo-redundant storage (GRS). This is not a bad thing. However, this option will double the cost.

If you don't want to use geo-redundant storage, you must disable this option before making your first backup.

Note The second option that is configured for us by default is soft delete, which keeps deleted vaults for fourteen days before they are permanently deleted.

This option is recommended; however, if you would like to disable soft delete, you can do it from the vault's Settings ► Security Settings page.

In Figure 9-4, you can see the message about the default configuration.

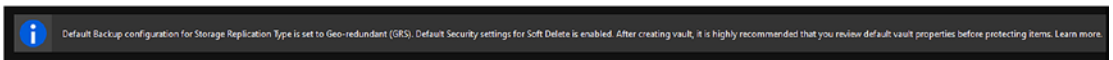


Figure 9-4. *Default replication configuration information message*

Disable Geo-redundant Storage (GRS) Replication

To disable GRS replication, wait for the vault to be created and open the Recovery Services Vaults page.

Click on the newly created vault, as shown in Figure 9-5.

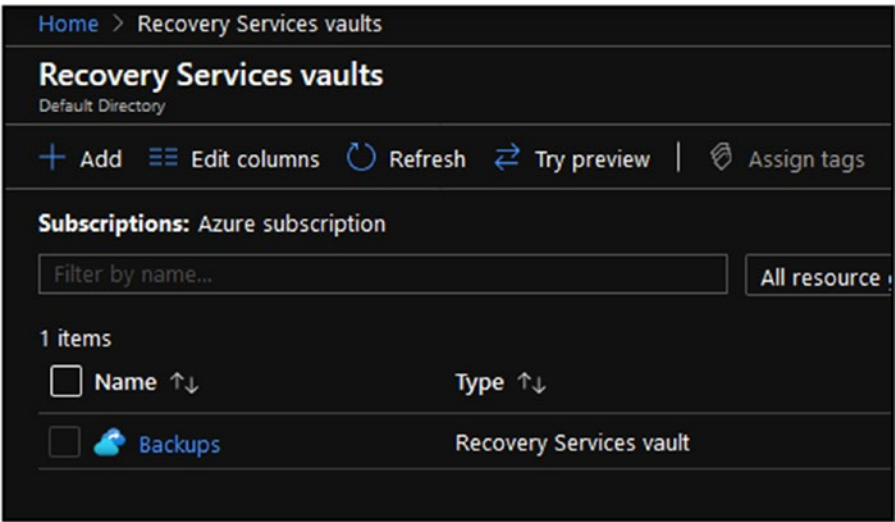


Figure 9-5. Recovery Services Vaults page

From the vault’s left-hand menu pane, click on “Settings,” as shown in Figure 9-6.

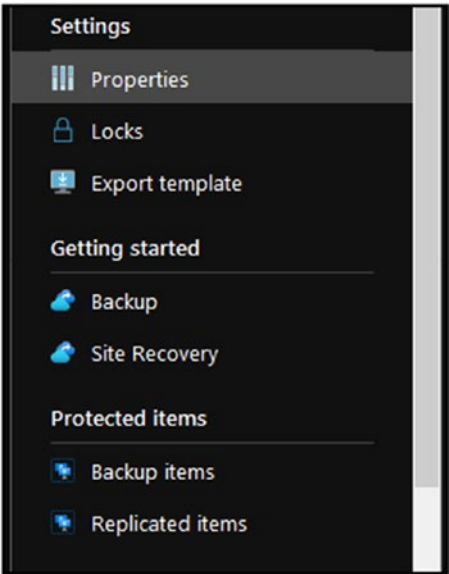


Figure 9-6. Settings

From the Settings page, click on “Update” under Backup Configuration, as shown in Figure 9-7.

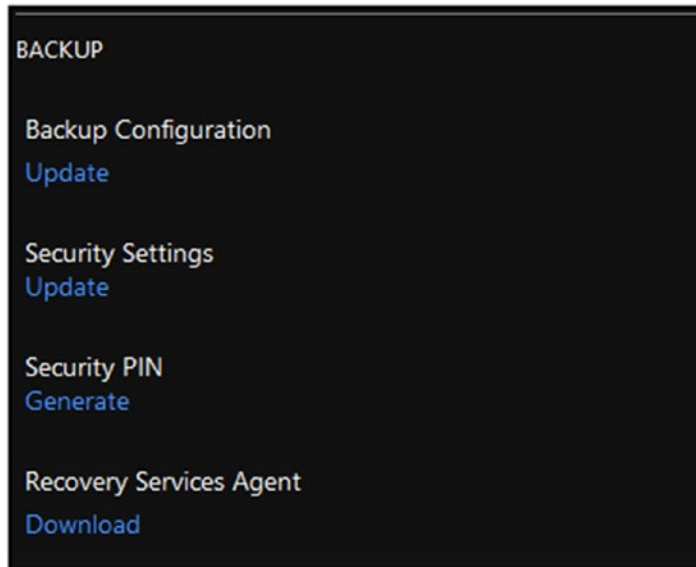


Figure 9-7. Backup configuration

On the Backup Configuration page, select “Locally-redundant” and click on “Save,” as shown in Figure 9-8.

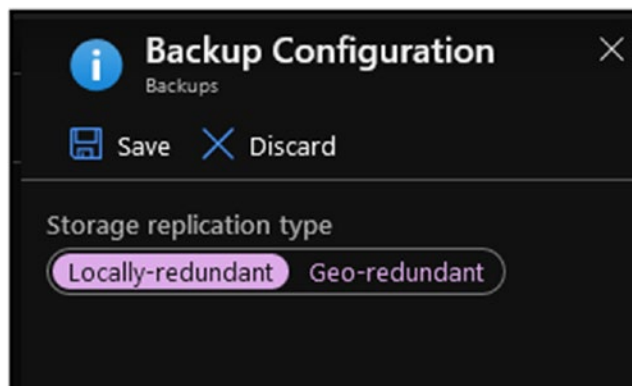


Figure 9-8. Backup configuration

Create Azure Recovery Services Vault Using Azure CLI

If you prefer to use Azure CLI to create the vault, you can use the following code:

```
az backup vault create --location westus --name Backups --resource-group apps
```

Azure CLI’s recovery-services module uses the following syntax:

```
az backup
```

To view all available commands, type the following:

```
az backup -help
```

Back Up Azure File Shares

Now that we have our recovery services vault set up and configured, we can go ahead and back up our AKS Azure file share volume. In my case, I have an Azure file share volume called `azurefile` which I created using the `4.7.Create_Volume_Claim.yaml` file located in Chapter 4.

My Azure file share volume is shown in Figure 9-9, and you can find yours if you follow all the steps in this book under Storage Accounts ➤ Select the storage account you used. On the overview page, click on “File Shares,” and you will see it.

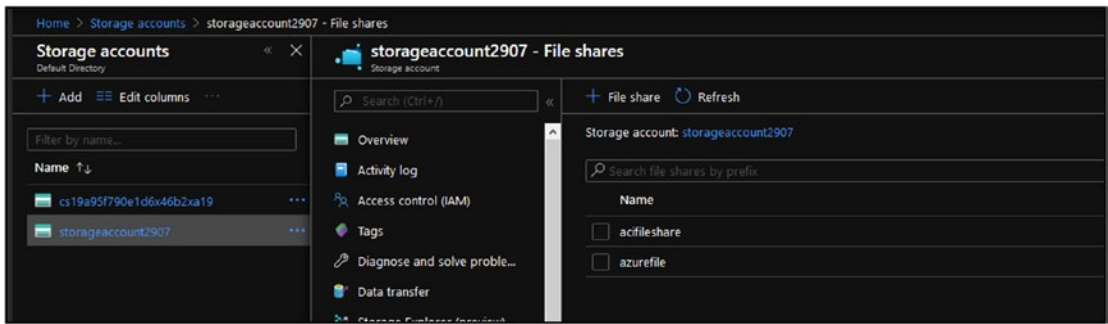


Figure 9-9. Azure file shares

For my Azure file share called azurefile, I will open the Recovery Services Vaults console. From the Console page, I will click on my vault, as shown in Figure 9-10.

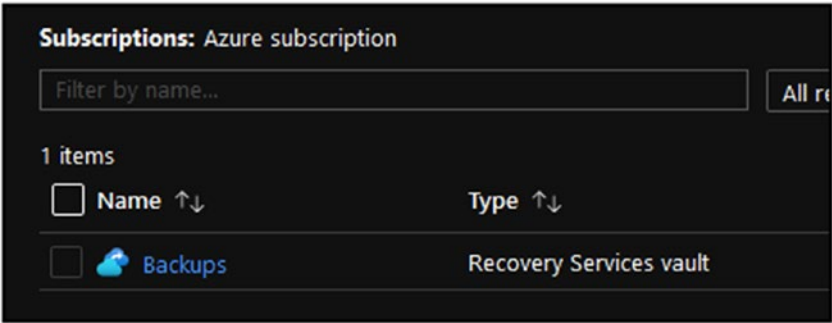


Figure 9-10. Recovery services vaults

From the vault’s overview page, click on “Backup,” as shown in Figure 9-11.

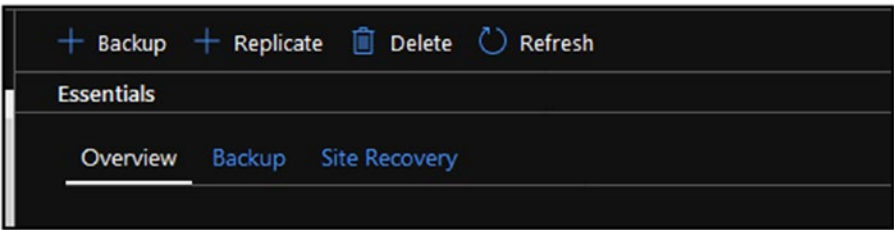


Figure 9-11. Backup

On the Backup Goal page, select “Azure” for the “Where is your workload running?” option and set the “What do you want to backup?” option to “Azure FileShare (Preview),” as shown in Figure 9-12.

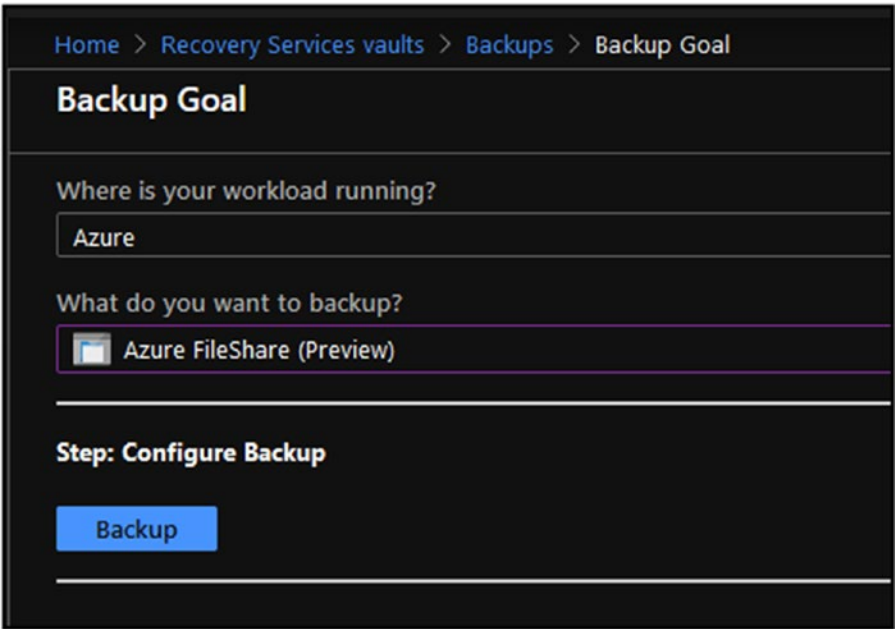


Figure 9-12. Backup Goal page

At this stage, Azure will start the process of discovering all the storage accounts that contain file shares and support backup.

Once the storage accounts are found, you will see what is shown in Figure 9-13. To continue, select the storage account that contains the share and click “OK.”

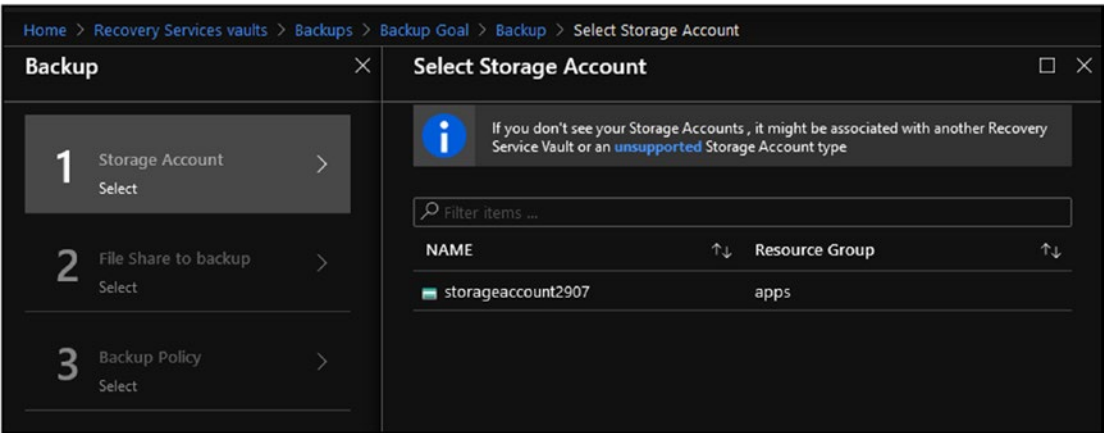


Figure 9-13. Select Storage Account page

On the Select File Shares page, I will select my AKS Azure file share volume and click “OK.”

Figure 9-14 shows the Select File Shares screen with my AKS file share selected.

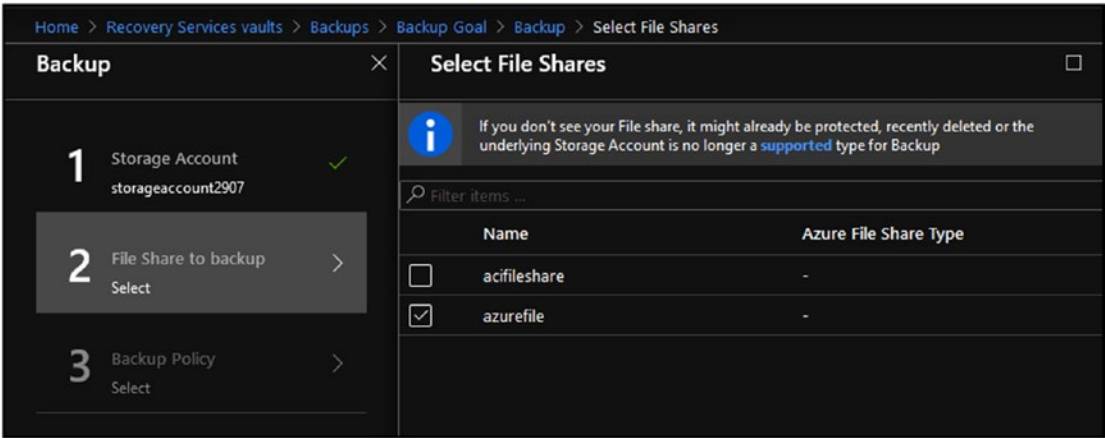


Figure 9-14. *Select File Shares screen*

The final step of the process will ask us to create a backup policy that defines the following:

Name

Backup schedule — It is recommended that you configure the schedule to run after hours and not during business hours. Running backup during business hours can impact application performance and cause network latency.

Retention period — Number of days to retain each backup

Figure 9-15 shows the Backup policy page. To complete the backup process, fill in the policy details and click “OK.”

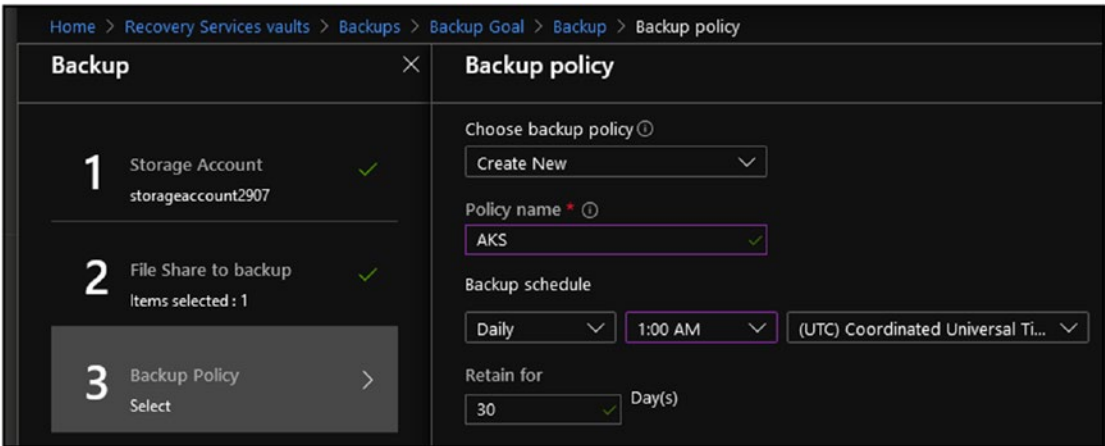


Figure 9-15. Backup policy

Note Using Azure CLI to take file share backups is not supported.

Restore Azure File Shares

In this section, we will go over the steps needed to restore an Azure file share successfully.

I will open the Recovery Services vault. From the Vault Overview page, I will click on the Backup tab, as shown in Figure 9-16.

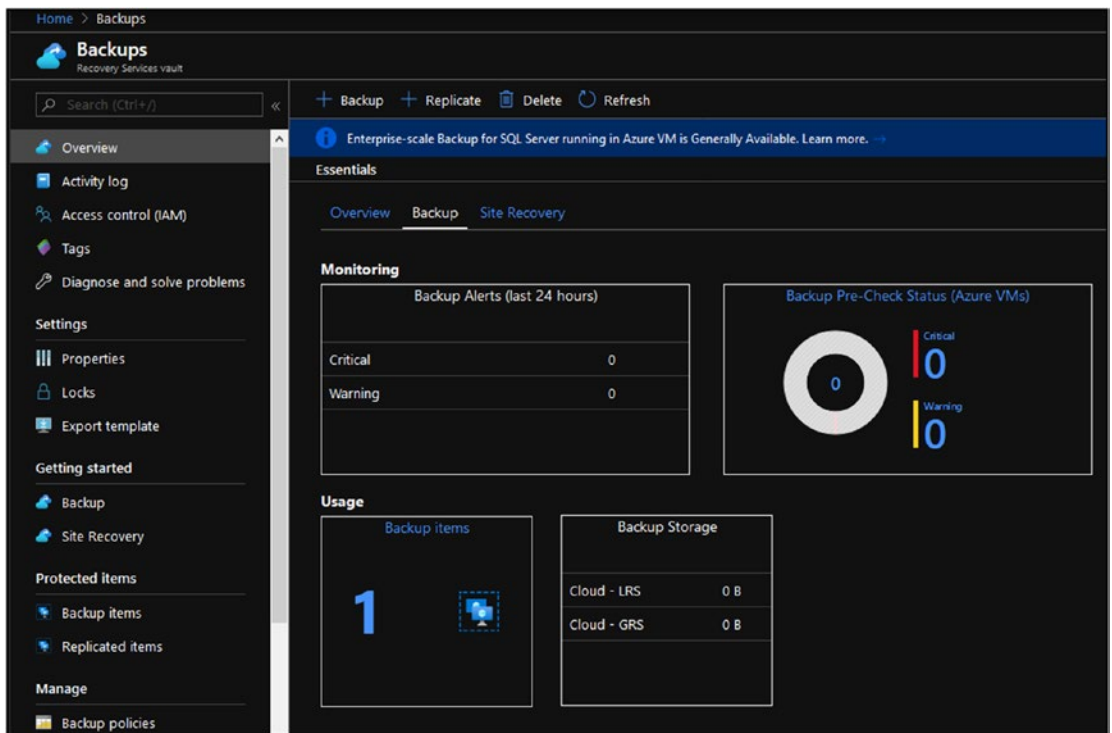


Figure 9-16. Backup tab

To restore my file share, I will click on the Backup items square, as shown in Figure 9-16.

In the Backup Management Type page, you will see all the backup and restore options, including the backup item count. To restore my file share, I will click on the “Azure Storage (Azure Files)” link, as shown in Figure 9-17.

Refresh

Primary Region

Secondary Region

BACKUP MANAGEMENT TYPE	BACKL
Azure Storage (Azure Files)	1
Azure Virtual Machine	0
Azure Backup Agent	0
Azure Backup Server	0
DPM	0
SQL in Azure VM	0

Figure 9-17. Backup management type

On the Backup Items page, I will see all the backup details and confirm that the last backup was successful, as shown in Figure 9-18. To continue, I will click on the file share I would like to restore.

Name	Storage Account	Resource Group	Last Backup Status
azurefile	storageaccount2907	apps	Success

Figure 9-18. Backup items

From the Backup Items page, I can see all the details regarding the file share and backups. To restore the file share, I will click on the “Restore Share” button, as shown in Figure 9-19.

Home > Backups - Backup items > Backup Items (Azure Storage (Azure Files)) > azurefile

azurefile

Backup Item

Backup now

Restore Share

File Recovery

Resume backup

Stop backup

Delete backup data

Essentials

Recovery services vault

Backups

Subscription name

Azure subscription

Figure 9-19. Backup items

From the Select Restore Point screen, I will select the restore point from the list, as shown in Figure 9-20.

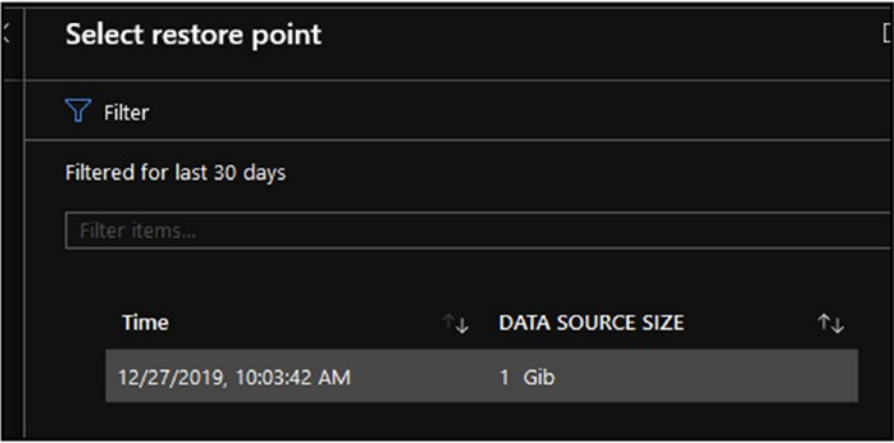


Figure 9-20. *Select restore point*

On the Select Restore Location screen, I have two options to select from.

1. Restore to the original location and overwrite the existing file or skip.
2. Restore to an alternate location.

Both options are shown in Figures 9-21 and 9-22.

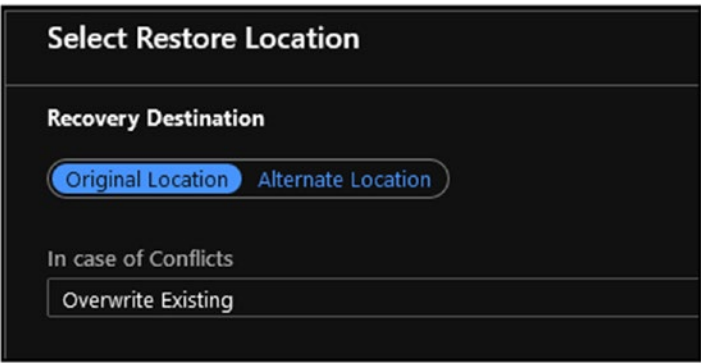


Figure 9-21. *Restore to the original location*

In Figure 9-22, you can see the “Alternate Location” option.

Recovery Destination

Original Location

Alternate Location

Storage Account *

storageaccount2907 (StandardLRS)

Select FileShare *

Filter file shares by prefix

Folder Name (created under /)

azurefile_restore

In case of Conflicts

Overwrite Existing

Figure 9-22. Restore to an alternate location

To start the restoration job, click on “Restore.” Next, I will show you how to monitor restoration jobs.

Monitor Backup Jobs

The last part of the restore job is to monitor the restore and backup job. To view the status of our backup and restore jobs, I use the following steps.

- Open Recovery Services console and select the vault I need to monitor.
- From the left-hand menu pane, under Monitoring, I will click on “Backup Jobs.”

Figure 9-23 shows the menu.

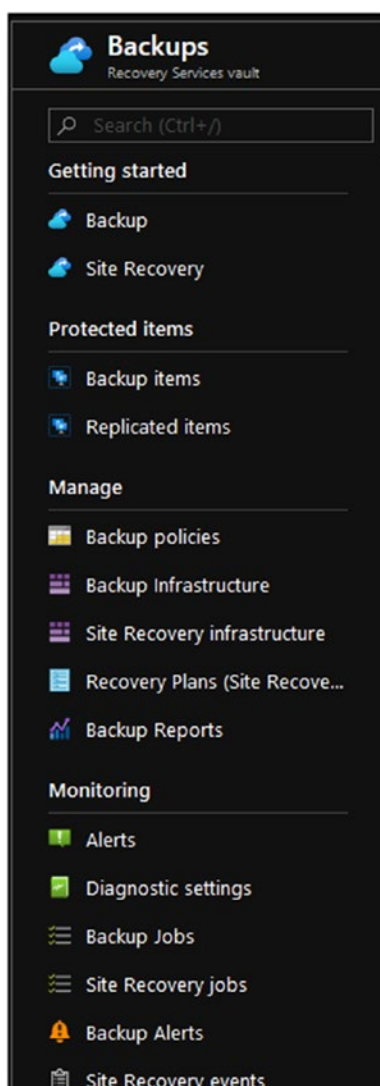


Figure 9-23. *Backup jobs*

The Backup Jobs page will show all the backup, restore, and management operation jobs and their statuses. In my case, you can see my restore job is completed, as shown in Figure 9-24.

Choose columnsFilterExport jobsRefreshView jobs in secondary region

Filtered by: Item Type - All item types, Operation - All Operations, Status - All Status, Start Time - 12/26/2019, 10:10:21 AM,

i

Completed fetching data from the service.

Filter items...

Workload name	Operation	Status	Type
azurefile(storageaccount2907)	Restore	Completed	AzureStorage
azurefile(storageaccount2907)	Backup	Completed	AzureStorage
azurefile(storageaccount2907)	Configure backup	Completed	AzureStorage
storageaccount2907(storageac...	Register	Completed	AzureStorage

Figure 9-24. Backup jobs status

Back Up Azure Container Instances (ACI)

In this section, I will show you how to back up Azure Container Instances (ACI) using Azure recovery services. Because ACI is also a managed service where Azure manages all the underlying infrastructure, our management domain is looking after the data.

If you remember from Chapter 3, we used the following three scripts to create a storage volume using Azure files:

- 1. Create_Storage.sh
- 2. Get_Storage_key.sh
- 3. Create_Container.sh

We need to follow the same process as with AKS to back up our storage volumes. If you followed Chapter 3, you should see a file share inside your storage account called acifileshare. If you look at your ACI container deployment, as shown in Figure 9-25, you will see that there is no backup option. Therefore, we need to back up the storage volume, as there is nothing to back up on the deployment level.

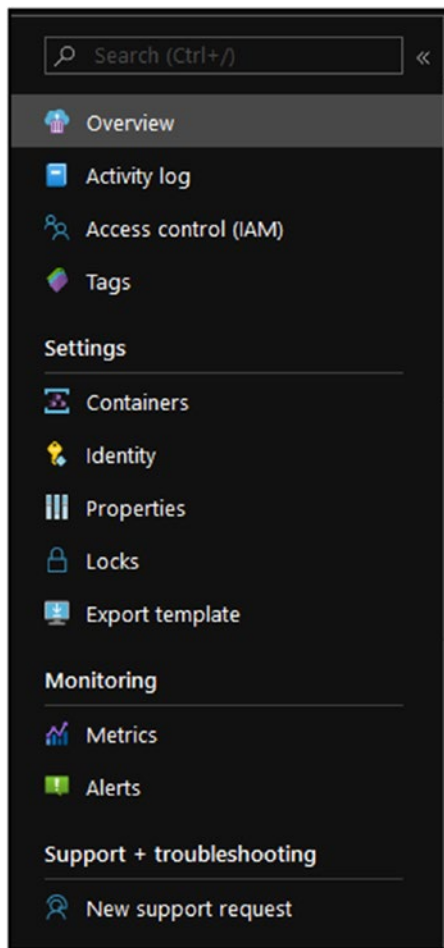


Figure 9-25. *ACI container menu*

To back up my ACI storage volume, I will use the same process as I used with AKS, but this time I will select my `acifileshare` Azure file share, which is shown in Figure 9-26.

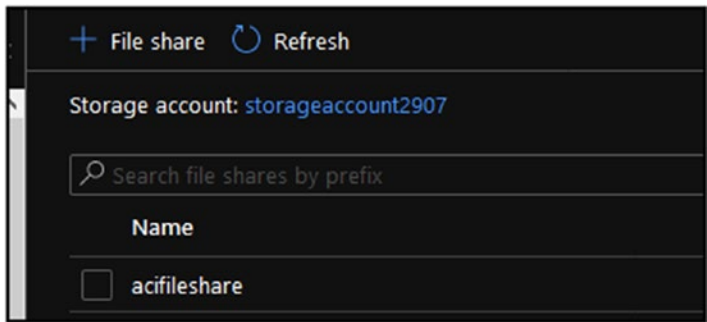


Figure 9-26. *acifileshare*

Please follow the same steps used in the AKS backup and restore sections.

Back Up Azure Container Registry (ACR)

When it comes to backing up and restoring our Azure Container Registry (ACR), Azure doesn't offer any built-in tools or methods. The only options that we have that are similar to a backup are included in the premium SKU of ACR, as follows:

Replication — Using geo-replication of repositories across multiple regions

Retention — Policies that handle retention of images stored in ACR

You can find these two options in the ACR's left-hand pane menu, as shown in Figure 9-27.

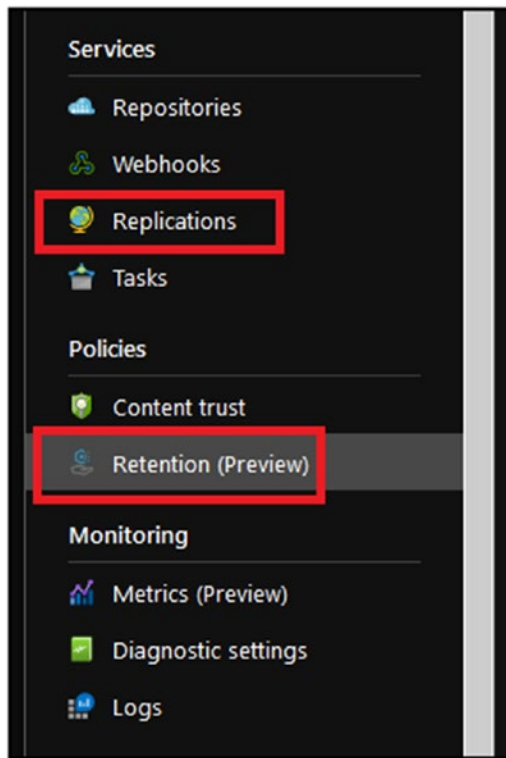


Figure 9-27. *Replication and retention policies in ACR*

Back Up Azure Docker Container Host VM

To back up my Linux container VM host, I will click on the VM, and in the left-side menu I will click on “Backup,” as shown in Figure 9-28. Because Azure Backup for VMs is a dedicated product, the backup menu is available to us directly from the VM’s left-side pane.

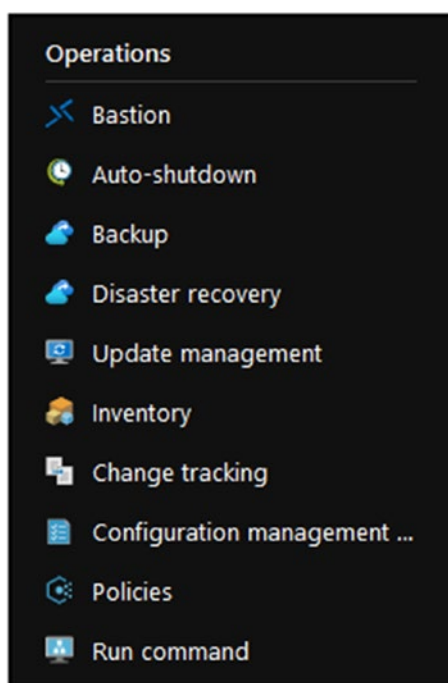
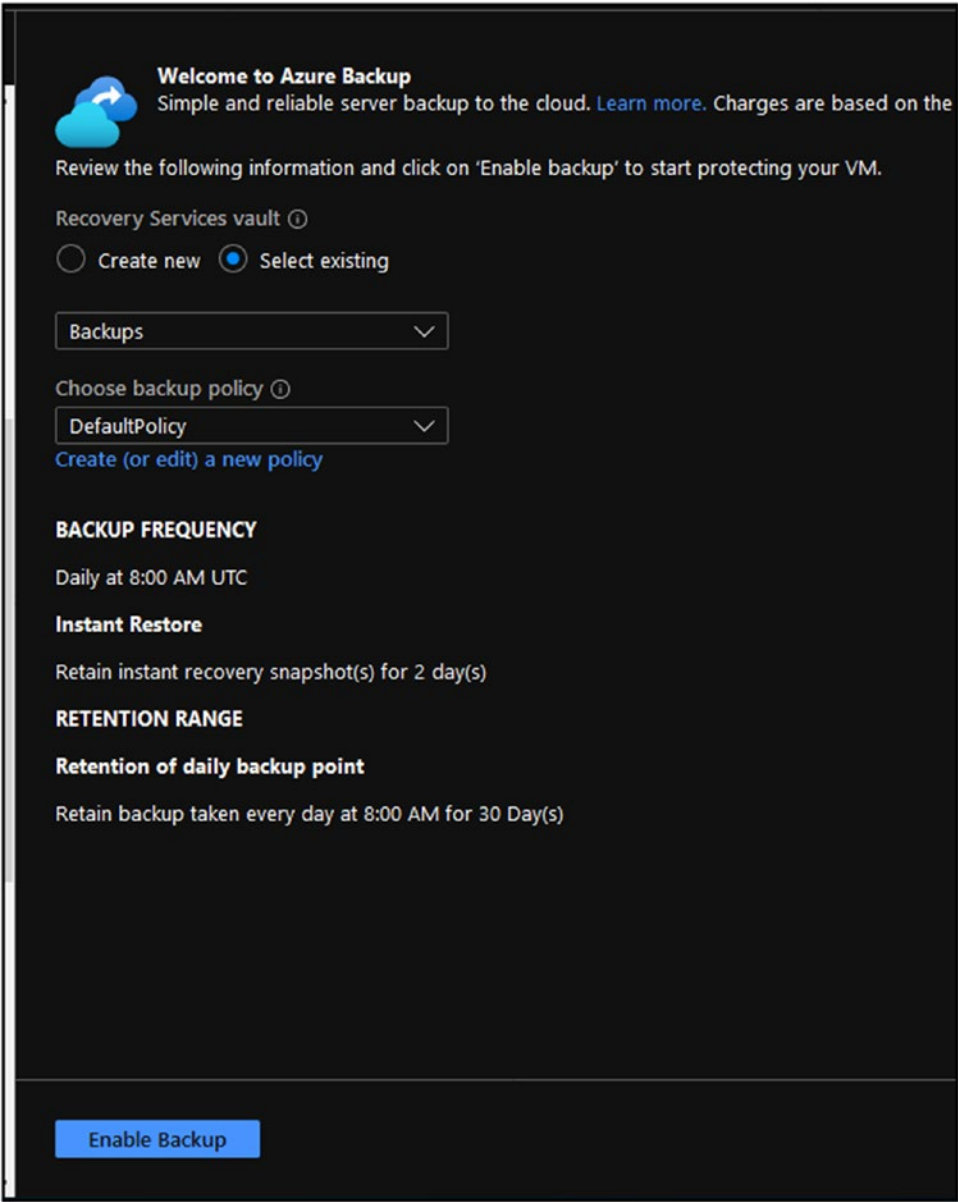


Figure 9-28. Backup

To back up my VM, I will click on “Backup,” and I will then be asked to select or create a new recovery services vault. In my case, I will use the same vault I created in the first section of this chapter.

Figure 9-29 shows the Welcome to Azure Backup screen with the Create Backup screen. I will fill in the details and click on “Enable Backup.”



Welcome to Azure Backup
Simple and reliable server backup to the cloud. [Learn more](#). Charges are based on the

Review the following information and click on 'Enable backup' to start protecting your VM.

Recovery Services vault ⓘ

☐ Create new ☒ Select existing

Backups ▾

Choose backup policy ⓘ

DefaultPolicy ▾

[Create \(or edit\) a new policy](#)

BACKUP FREQUENCY

Daily at 8:00 AM UTC

Instant Restore

Retain instant recovery snapshot(s) for 2 day(s)

RETENTION RANGE

Retention of daily backup point

Retain backup taken every day at 8:00 AM for 30 Day(s)

Enable Backup

Figure 9-29. *Create a backup*

After enabling backup, I will see the Backup page of my VM, where I have the option to back up the VM now, stop backup, and restore VM.

I will go ahead and run a backup job now and not wait for the default schedule to start, which can take some time. To run a backup job now, I will click on the “Backup now” button located in the top menu.

Figure 9-30 shows the Backup Now screen.

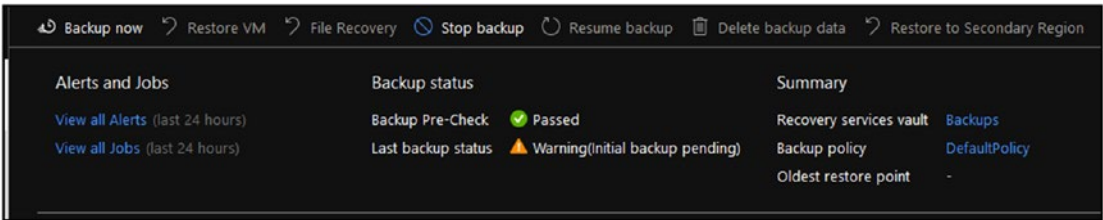


Figure 9-30. Backup now

After clicking on the “Backup now” button, I will have the option to select a custom retention period, as shown in Figure 9-31.

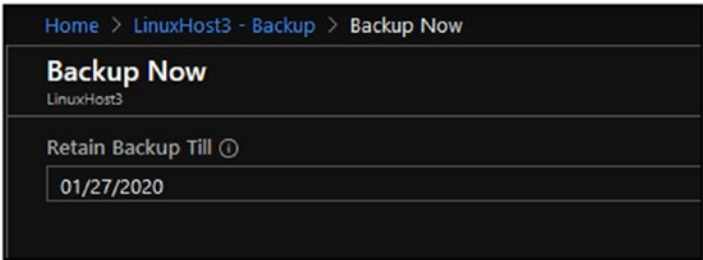


Figure 9-31. Retain backup

To check the backup status, I will click on the “View all jobs” link on the VM Backup page. I will then see the status of all the jobs and operations, as shown in Figure 9-32.

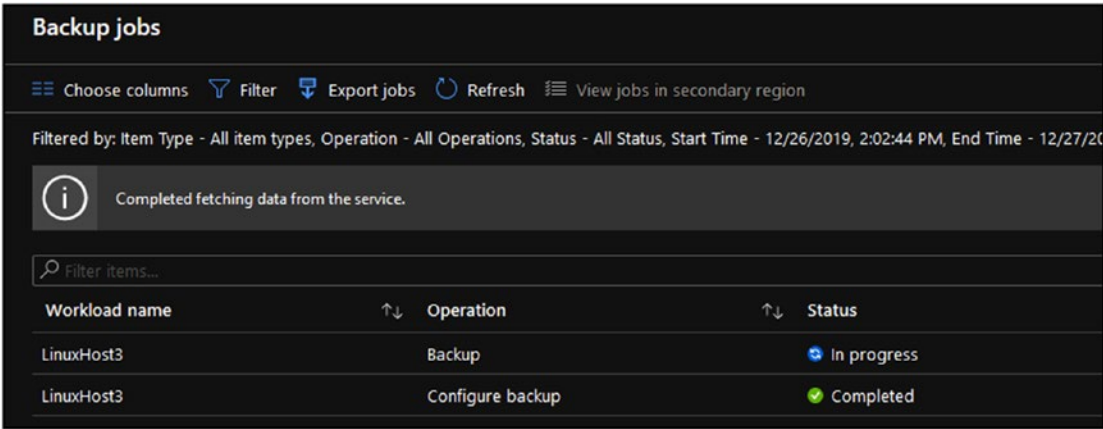


Figure 9-32. Backup jobs

Back Up Azure VM Using Azure CLI

To run the same backup job using Azure CLI, I will use the following code:

```
az backup protection backup-now --resource-group apps --backup-  
management-type AzureIaaSVM --vault-name backups --container-name  
LinuxHost3 --item-name LinuxHost3 --retain-until 30-12-2019
```

This code is available in the code library of this book under the name 9.1.Backup_VM.sh.

Note Please type the name of the VM in place of container-name and item-name.

```
az backup protection backup-now \  
--resource-group apps \  
--backup-management-type AzureIaaSVM \  
--vault-name backups \  
--container-name LinuxHost3 \  
--item-name LinuxHost3 \  
--retain-until 30-12-2019
```

Restore a Virtual Machine

To restore my VM, I will use the Restore function that is available in the VM backup menu, located in the VM's left-hand pane. In Figure 9-33, you can see the Restore VM option, which allows us to restore our VM directly on the VM page.

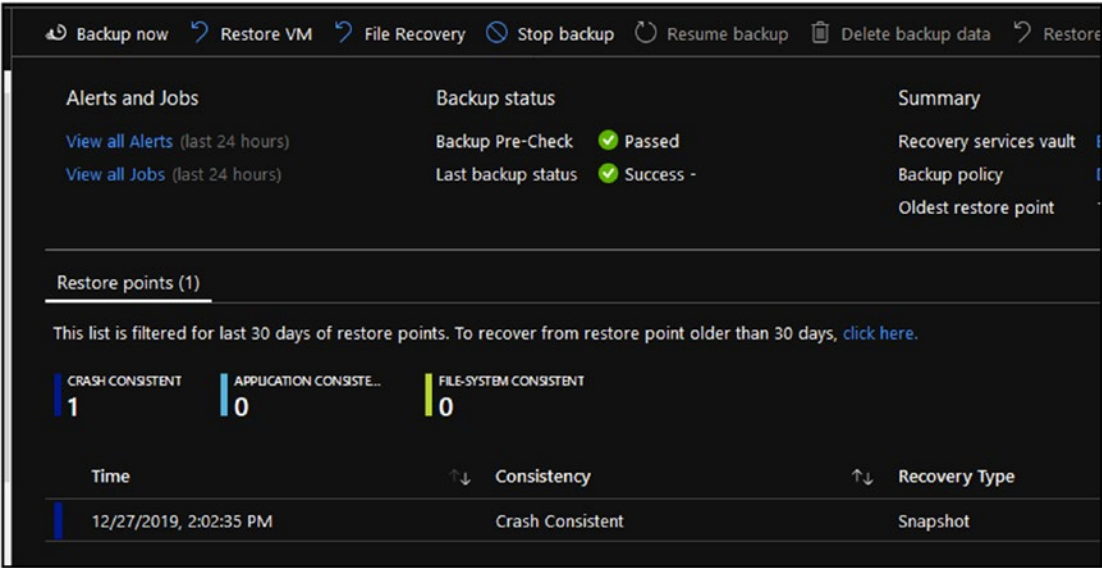


Figure 9-33. Restore VM

To restore my VM, I will click on the “Restore VM” button and then will be redirected to select a restore point, as shown in Figure 9-34.

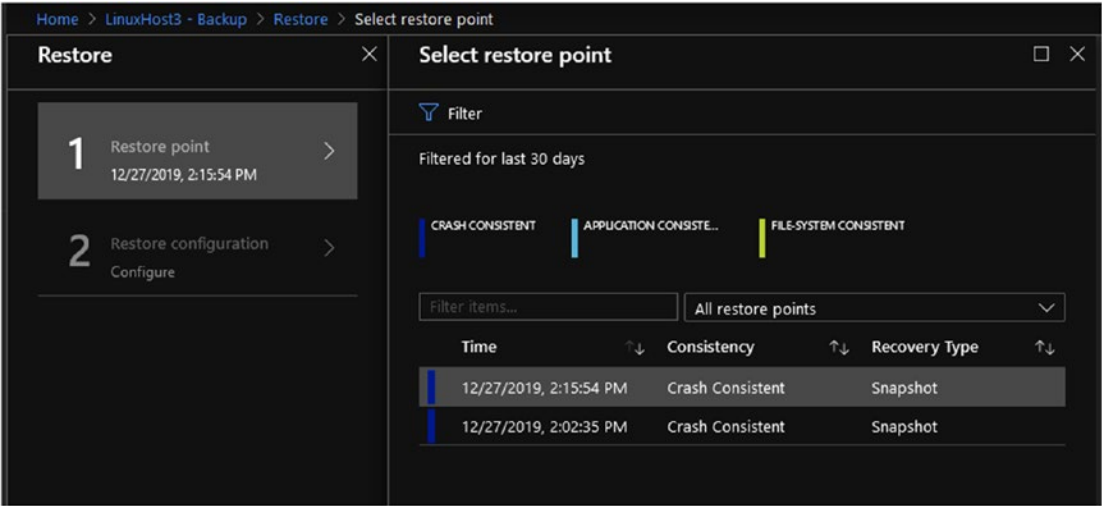


Figure 9-34. Select a restore point

After selecting the restore point, I will be presented with the following two options (shown in Figure 9-35):

1. Create a new virtual machine.
2. Replace existing virtual machine.

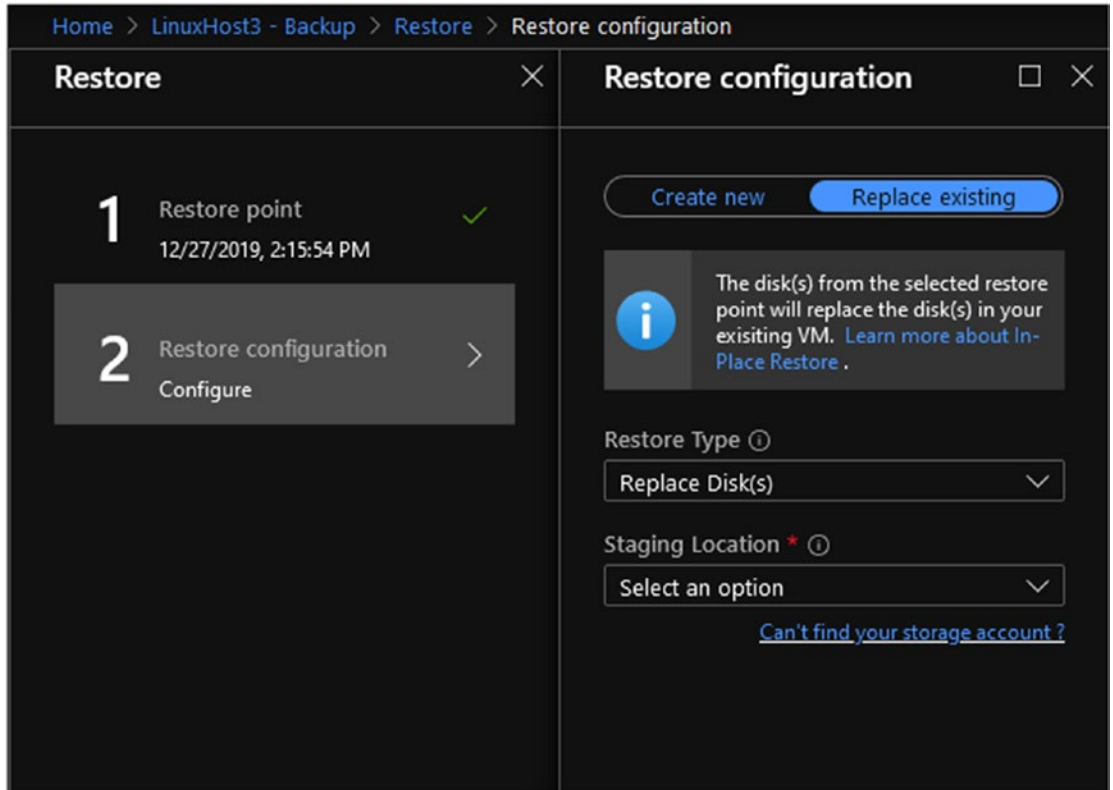


Figure 9-35. *Restore configuration*

In my case, I will select “Replace existing” and click “OK.” I will then click on the “Restore” button to start the restoration job.

To monitor the restoration job’s status, I will click on “View all jobs,” and monitor the progress as shown in Figure 9-36.

LinuxHost3	Restore	In progress
LinuxHost3	Backup	Completed

Figure 9-36. *Restore progress*

Delete Recovery Services Vault

In this final section, I will show you how to clean up your tenant from the recovery services vault after running all the preceding labs.

Keeping the vault without using it will only attract charges for the storage used. Before we can delete our recovery services vault, we need to stop all backups. Once the backup is stopped, we will be able to delete the vault after fourteen days, because by default soft delete is enabled on the vault.

To stop all backup jobs, open the recovery services vault from the Azure portal and click on the vault you would like to delete. From the vault’s left-hand menu pane, click on “Backup items” and click on all the items that show a backup item count, as shown in Figure 9-37.

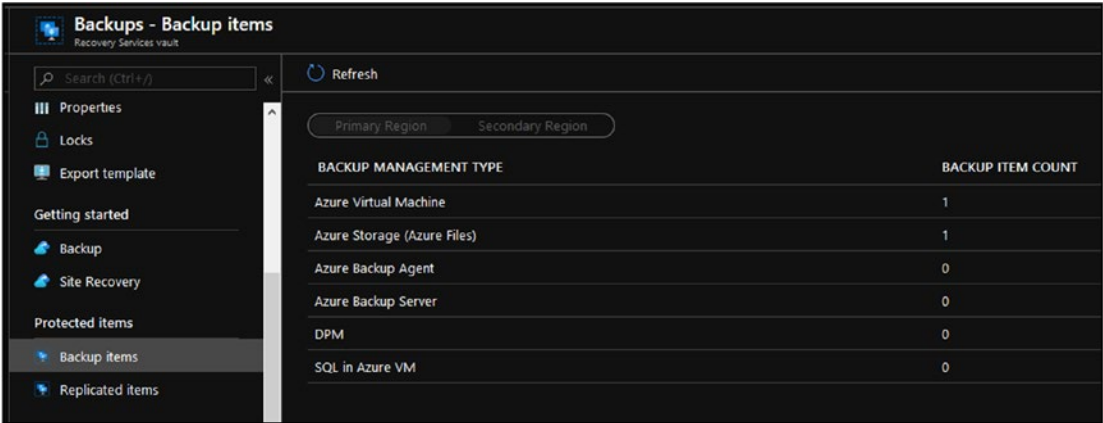


Figure 9-37. Backup items

From the action menu (...) click on “Stop backup,” as shown in Figure 9-38.

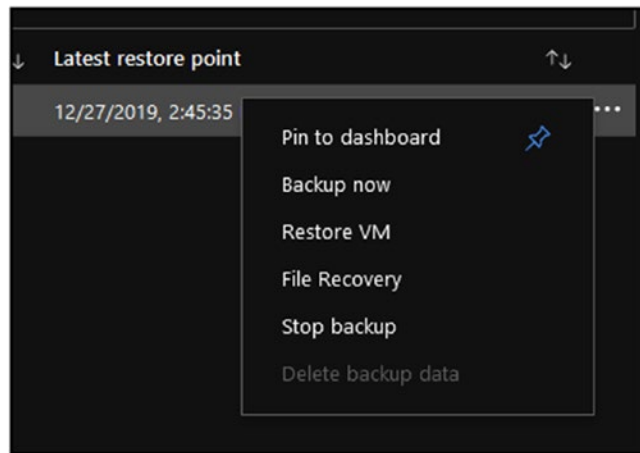


Figure 9-38. Stop backup

On the Stop Backup page, select “Delete backup data” and type the name of the item—in my case, it is the VM name—and click “OK,” as shown in Figure 9-39.

A screenshot of the 'Stop Backup' page in the Azure portal. The page title is 'Stop Backup' with the subtext 'LinuxHost3'. Below the title is a dropdown menu set to 'Delete Backup Data'. A message box with an information icon states: 'This option will stop all scheduled backup jobs and delete backup data. Learn more https://aka.ms/SoftDeleteCloudWorkloads'. Below this is a form with three fields: 'Type the name of Backup Item *' with the value 'LinuxHost3' and a green checkmark; 'Reason' with a dropdown menu showing '0 selected'; and 'Comments' with a large empty text area. The page has a dark grey background.

Figure 9-39. Stop backup

After fourteen days, you can delete the vault by using the “Delete” button in the vault’s overview page, as shown in Figure 9-40.

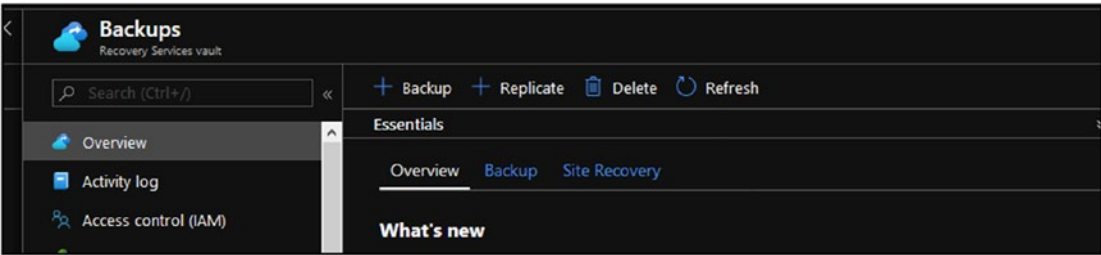


Figure 9-40. Delete vault

Summary

In this chapter, we learned how to use Microsoft Azure recovery services to back up and restore all our container services in Azure. We also learned how to use Azure CLI to back up workloads and automate our work. My recommendation for this chapter is to utilize Azure recovery services in a production environment and, more important, test the restoration procedure before going live with your environment.

CHAPTER 10

Troubleshooting Containers and Containerized Applications on Azure

In this chapter, we will learn how to troubleshoot our container services in Microsoft Azure in case things are not working as we think they should.

Not a lot of people are aware of the fact that support in Microsoft Azure is limited to the support level you purchase. Technical support in Azure doesn't cover contacting support unless you have a paid support subscription.

If you browse to the Azure Support Plans page (Figure 10-1), you will see that the basic support plan that comes with any subscription by default has no technical support, and you can't log support cases: <https://azure.microsoft.com/en-au/support/plans/>.

Compare support plans

Explore the range of Azure support options and choose the plan that best fits, whether you're a developer who's just starting your cloud journey or a large organisation that's deploying business-critical, strategic applications.

	BASIC	DEVELOPER	STANDARD	PROFESSIONAL DIRECT	PREMIER
		Purchase support	Purchase support	Purchase support	Contact Premier
Scope	Available to all Microsoft Azure accounts	Microsoft Azure: Trial and non-production environments	Microsoft Azure: Production workload environments	Microsoft Azure: Business-critical dependence	All Microsoft Products, including Azure: Substantial dependence across multiple products
Customer Service, Self-help and Communities	24/7 access to billing and subscription support, online self-help, documentation, whitepapers and support forums	24/7 access to billing and subscription support, online self-help, documentation, whitepapers and support forums	24/7 access to billing and subscription support, online self-help, documentation, whitepapers and support forums	24/7 access to billing and subscription support, online self-help, documentation, whitepapers and support forums	24/7 access to billing and subscription support, online self-help, documentation, whitepapers and support forums
Best Practices	Access to full set of Azure Advisor recommendations	Access to full set of Azure Advisor recommendations	Access to full set of Azure Advisor recommendations	Access to full set of Azure Advisor recommendations	Access to full set of Azure Advisor recommendations
Health Status and Notifications	Access to personalised Service Health Dashboard & Health API	Access to personalised Service Health Dashboard & Health API	Access to personalised Service Health Dashboard & Health API	Access to personalised Service Health Dashboard & Health API	Access to personalised Service Health Dashboard & Health API
Technical Support		Business hours access ¹ to Support Engineers via email	24x7 access to Support Engineers via email and phone	24x7 access to Support Engineers via email and phone	24x7 access to Support Engineers via email and phone
Who Can Open Cases		Unlimited contacts / unlimited cases	Unlimited contacts / unlimited cases	Unlimited contacts / unlimited cases	Unlimited contacts / unlimited cases

Figure 10-1. Azure support plans

This fact is very important in case you have a very important production environment running in Azure. However, sometimes you can get away with not contacting support by using the built-in troubleshooting tools that Microsoft has made available for us.

In this chapter, I will show you the built-in tools and teach you which things to check in case of an issue. I will cover the following topics:

- Troubleshooting AKS
- Troubleshooting ACI
- Troubleshooting ACR
- Troubleshooting VMs

Troubleshooting Azure Kubernetes Service (AKS)

AKS is a very complex product, but when you take away all the things that Azure is looking after you end up with a few items that you need to look after yourself and know how to troubleshoot.

Capacity Issues

The number one instigator of performance issues that I have seen during my fifteen-plus years in IT is lack of capacity.

Not having enough computing capacity inside your AKS cluster can cause your environment to run slowly, not perform well, and be less responsive.

My tip here is to make sure you use Azure Monitor for Containers, as we covered in Chapter 8, and increase your cluster capacity in case of performance issues.

I also recommend that when it comes to AKS worker nodes, you use the autoscaling option to increase worker nodes' capacity.

As a rule of thumb, it is recommended you scale any containerized application that uses 60 percent to 80 percent of its allocated RAM or CPU.

Storage Issues

Another issue that I see often is that applications stop working because the disk space allocated to the volume has run out of space.

If you are using Azure files for your AKS cluster storage, make sure you allocate enough disk space and that it takes into consideration growth for three years. If you plan your storage capacity this way, you will have enough time to plan and add disk space to your volumes.

In most cases, disk space runs out when people are not around or are busy with other work. Make sure you plan accordingly because once a workload is in production it is hard taking it offline to make complex changes under pressure and in a short time.

AKS Limits

By default, when you create an AKS cluster using the portal, each node is limited to running thirty pods per node. If you deploy your AKS using Azure CLI, the limit is 110.

This is very important information you should know, and if you would like to change this default behavior you can do so using Azure CLI. When you set your AKS cluster using Azure CLI, add the following switch to your command and specify a maximum number:

```
--max-pods
```

IP Address Limit

If you decide to increase the limit of your AKS pods per node, you might run into the problem of not having enough IP addresses in your cluster for all your pods.

For example, if you have three nodes with the max number of pods of 200 per node, you will need a subnet of, at minimum, 610 addresses (I'm also taking into consideration a few IP addresses that your node will use). To overcome this issue, you will need to create your virtual network and subnets before creating your AKS cluster.

After your virtual network and subnets are configured, you will need to use Azure CLI and specify the following commands with your `az aks create`:

```
--vnet-subnet-id
```

For more information, please refer to the help documentation of the following commands:

```
az aks  
az network
```

Pod Issues

If you have specific issues with pods running inside Kubernetes, you can always use Azure Cloud Shell and the Kubernetes command-line to review the configuration of your deployment.

In Azure Cloud Shell, type the following command to view all available commands:

```
kubectl
```

To view a specific command's help file, use the following:

```
Kubectl command -help
```

Troubleshoot Azure Container Instances (ACI)

In this section, we will learn how to troubleshoot the top issues you may run into with ACI containers. Because ACI is a managed service, the most important guideline you should follow is making sure you understand the requirements and limitations of the service.

Slow Start

The number one issue people see when they use ACI is a slow start of the container group in ACI. A slow start can be attributed to one of the following two reasons: image size or image location.

Image Size

A large image size can contribute to the slow start of your containers in ACI because ACI needs to pull the image down from the registry where the image is stored.

It is very important you optimize your container image and reduce any packages that are not needed.

Some Docker container images offer to build images and runtime images for the production environment. It is recommended you use a runtime image that only includes packages needed to run the code and not to build it.

Although not in the scope of this book, the documentation on Docker multi-stage builds and dockerfile best practices can be found here: https://docs.docker.com/develop/develop-images/dockerfile_best-practices/.

Image Location

If you are using ACR to host your Docker images, and your ACI deployment is in a region that is not the same as your ACR location, it will take longer for your container to start.

It is recommended you host your ACR repository in the same region in which your ACI deployment will run. Doing the opposite will result in higher network latency.

Support Versions

If you deploy an ACI container group and receive an error message saying `OsVersionNotSupported` it is because you are trying to use an image that ACI doesn't support. This error is very common with Windows containers where semi-annual releases are not supported by ACI. Please make sure you check if your image is supported by ACI before deploying it to ACI.

Bad Performance

If your ACI container group is running but experiencing bad performance like high CPU, high memory usage, and network latency, it is most likely because the hardware specifications are not enough.

Before deploying your ACI container group, please make sure you plan for at least two times the resources the application needs to prevent bad performance and downtime for redeployment.

Troubleshoot Azure Container Registry (ACR)

In this section, we will cover the main issues that might affect your ACR service and how to overcome them.

Storage Issues

One of the top issues you might come across is running out of space in your ACR repository due to the SKU limit. Remember that ACR storage is based on the SKU level you are on and is listed here:

Basic — 10 GB

Standard — 100 GB

Premium — 500 GB

Make sure you check your repository storage capacity in the ACR overview.

Performance Issues

I covered this point in the ACI section; however, if you find that your containers are taking too much time to load, please make sure your image sizes are not too large and your containers are in the same region in which your ACR repository is located. It is always recommended you place your ACR in the same region in which your containerized application is located.

Can't Log In to ACR

I think this is the number one issue you might come across in ACR.

Because ACR is a private repository that requires a username and password in the form of a server name and access keys, it is very easy to mistype the server name and access key password.

So, if you can't log in to your ACR repository, make sure you are using the correct following details:

Registry name

Login server

Password

You also need to make sure the Admin user option is enabled.

You will find all the access key and login details in the Access Key page located in your ACR left-pane menu.

Access Issues (Firewall)

If you opted to use the Premium ACR SKU and configured the advanced firewall and virtual network options and can't access your repository, it is because you might have misconfigured your firewall and blocked access to your ACR.

To overcome such firewall and virtual networking issues, please make sure you apply one rule at a time and test it.

Troubleshoot Azure Docker Container Host VM

When it comes to troubleshooting Azure VMs, Microsoft offers many tools, and personally I think many people will end up running some of their container workloads inside VMs.

VMs offer a good, isolated, and cost-effective platform to run containers for production and testing. Virtualization is also a mature technology that offers reliable computing power at lower cost.

Because of the preceding reason, you will find that Azure offers many tools to troubleshoot VMs' performance and access issues. In Figure 10-2, you can see the Support + Troubleshooting menu each VM has.

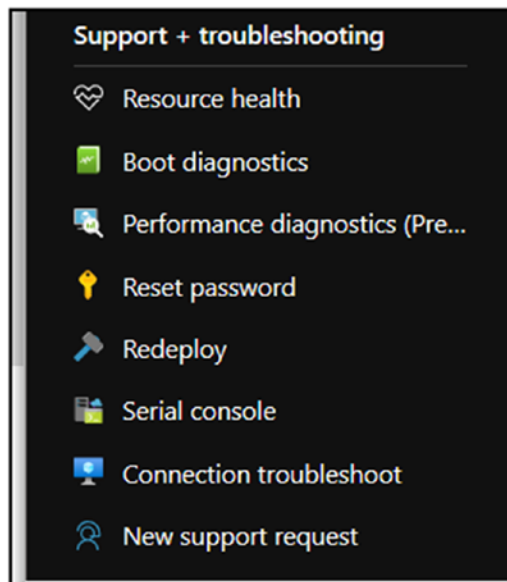


Figure 10-2. *Support + troubleshooting*

Out of the box, we can do the following:

Boot diagnostics — Check boot menu using a screenshot

Reset password — Reset local administrator password of the VM

Redeploy — If VM performance is slow, you can use redeploy to move the VM to a different host

Serial console — Connect to the VM using a serial console when you can't connect to the VM using RDP or SSH

Connection troubleshooting — Check that nothing is blocking your connection to the VM, like a misconfigured firewall.

Boot Diagnostics

Boot diagnostics allow us to troubleshoot boot issues with the VM. Before we can use it, we need to enable it and configure a storage account that will be used by the boot diagnostics.

To configure boot diagnostics, from the VM's Support + Troubleshooting menu, click on "Boot diagnostics."

From the Boot Diagnostics menu, click on "Settings," as shown in Figure 10-3.

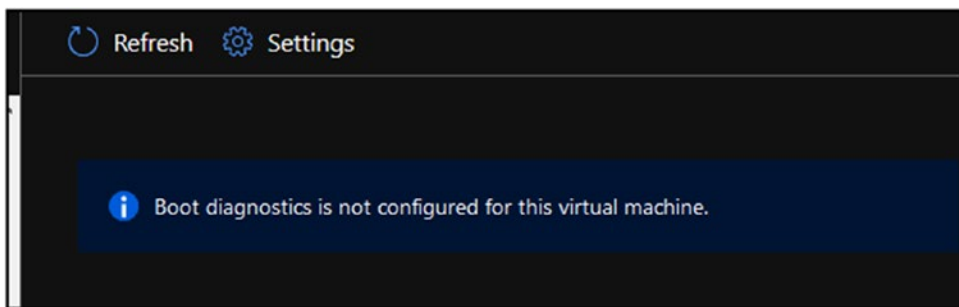


Figure 10-3. *Boot diagnostics settings*

From the Settings screen, we need to specify a storage account that will be used by the boot diagnostics agent to store logs and screenshots. Figure 10-4 shows the Settings page.

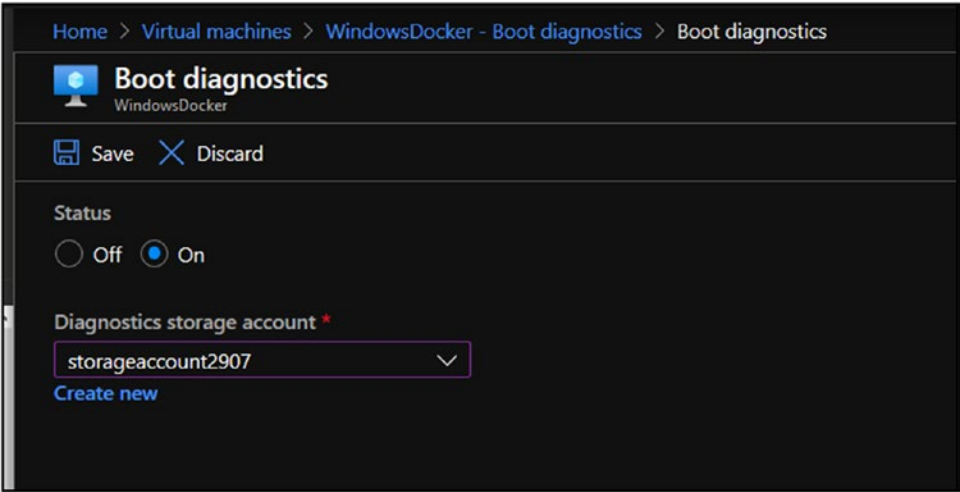


Figure 10-4. Settings page

After enabling boot diagnostics, I will go ahead and reboot the VM. As shown in Figure 10-5, I can see my boot menu and even download the boot screenshot that was taken.

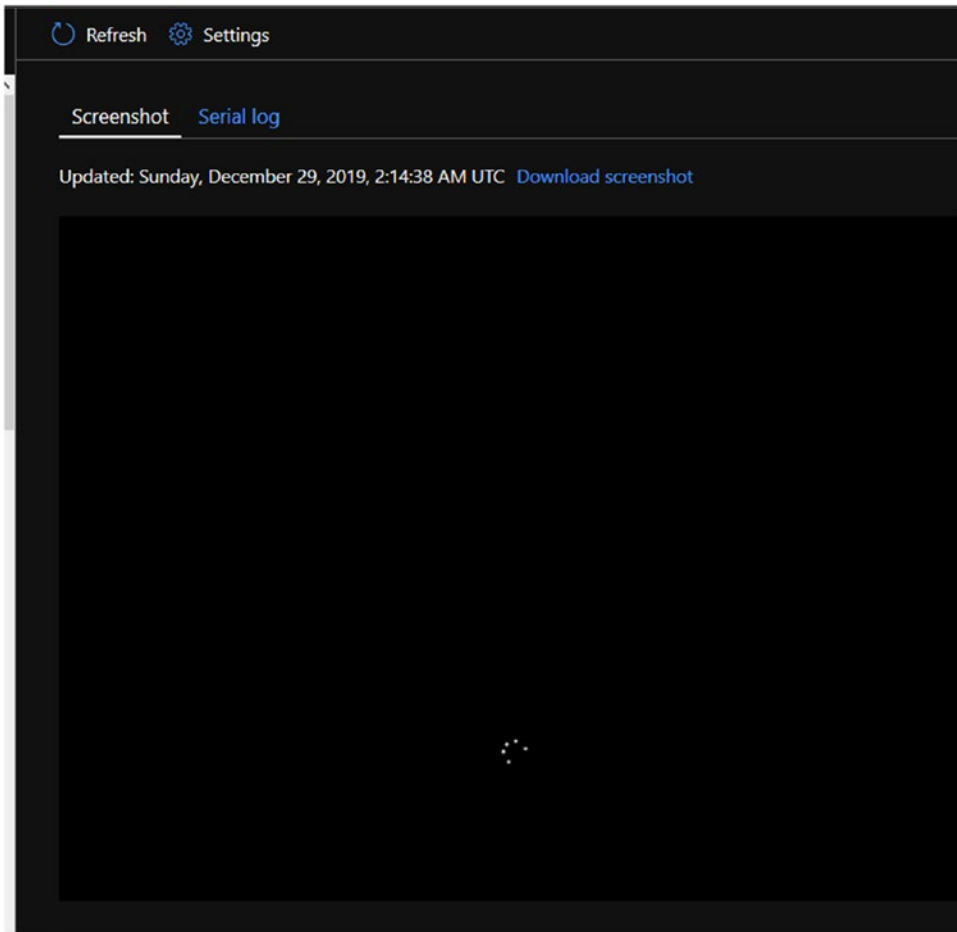


Figure 10-5. Screenshot

If I click on the Serial Log tab, I will have the option to download the serial log by clicking on “Download.” Figure 10-6 shows the serial log file.

```

WindowsDocker.2c0a4dc3-cd31-47e5-b792-23de448f4531.serialconsole.log - Notepad
File Edit Format View Help
[2][1;1H[2][H<?xml version="1.0"?>
<machine-info>
<name>WindowsDocker</name>
<guid>2c0a4dc3-cd31-47e5-b792-23de448f4531</guid>
<processor-architecture>AMD64</processor-architecture>
<os-version>10.0</os-version>
<os-build-number>17763</os-build-number>
<os-product>Windows Server 2019 Datacenter</os-product>
<os-service-pack>None</os-service-pack>
</machine-info>
[2][2][1;1H[0m[22m[25m[27m[40m[37m

initialized. [4;1H
[5;1HUse the "ch -?" command for information about using channels.
general help. [7;1H
[8;1H

```

Figure 10-6. Serial log file

Redeploy VM

If your VM becomes unresponsive and slow, one of the options Azure gives us is redeploying the VM to another virtual host.

Redeployment will not fix performance issues that are caused by unspaced VMs or misconfigured applications. This is only to be used when you feel that Azure has underling issues.

To redeploy a VM, from the Support + Troubleshooting menu, click on “Redeploy.” From the Redeploy page, click on “Redeploy,” as shown in Figure 10-7.

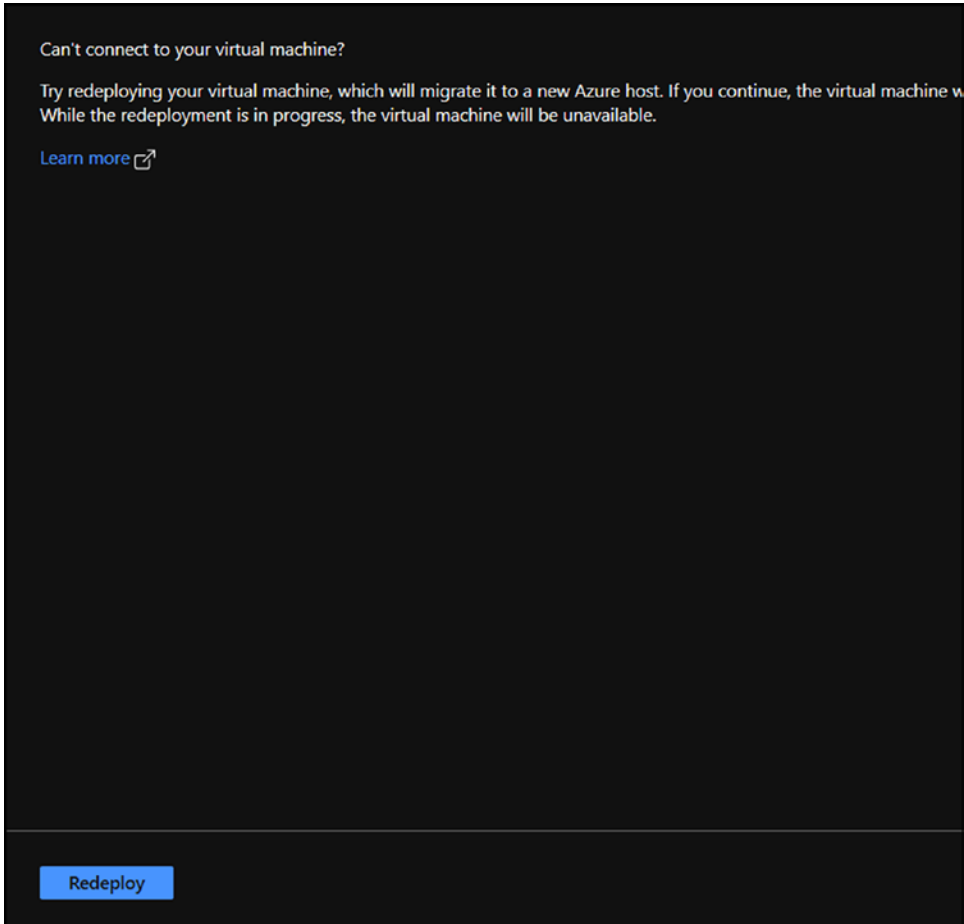


Figure 10-7. *Redeploy page*

It is important to note that once the VM is redeploying, it will be restarted, so make sure you perform this action after hours, or let the application owners know.

If you prefer to use Azure CLI and Azure Cloud Shell for this task, please use the following command:

```
az vm redeploy --resource-group apps --name linuxhost3
```

Serial Console

The serial console option allows us to connect to an Azure VM (Linux or Windows) using a COM1 port that is running on a separate network.

To access the serial console, boot diagnostics needs to be enabled, and you will need a contributor RBAC role or Global Administrator role to access the console.

If both requirements are fulfilled, click on “Serial console” in the Support + Troubleshooting menu.

Figure 10-8 shows the serial console of a Windows Server VM.

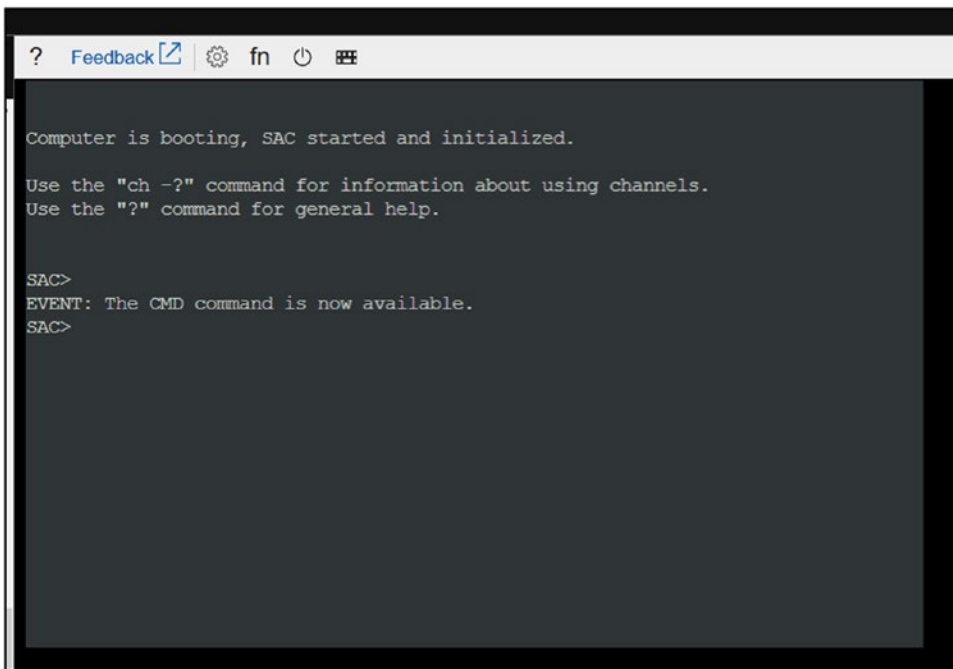
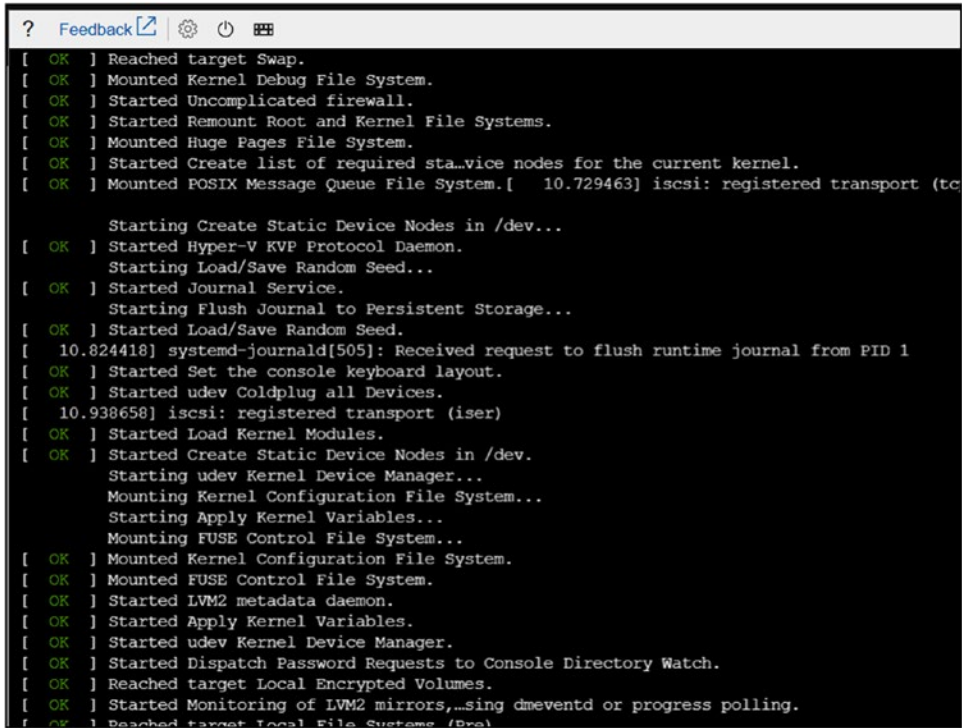


Figure 10-8. *Serial console*

To view all the available commands, I can simply type help.

A Linux virtual machine will show a more detailed console, as you can see in Figure 10-9.



```

? Feedback [link] [gear] [power] [keyboard]
[ OK ] Reached target Swap.
[ OK ] Mounted Kernel Debug File System.
[ OK ] Started Uncomplicated firewall.
[ OK ] Started Remount Root and Kernel File Systems.
[ OK ] Mounted Huge Pages File System.
[ OK ] Started Create list of required sta...vice nodes for the current kernel.
[ OK ] Mounted POSIX Message Queue File System.[ 10.729463] iscsi: registered transport (tc

Starting Create Static Device Nodes in /dev...
[ OK ] Started Hyper-V KVP Protocol Daemon.
Starting Load/Save Random Seed...
[ OK ] Started Journal Service.
Starting Flush Journal to Persistent Storage...
[ OK ] Started Load/Save Random Seed.
[ 10.824418] systemd-journald[505]: Received request to flush runtime journal from PID 1
[ OK ] Started Set the console keyboard layout.
[ OK ] Started udev Coldplug all Devices.
[ 10.938658] iscsi: registered transport (iser)
[ OK ] Started Load Kernel Modules.
[ OK ] Started Create Static Device Nodes in /dev.
Starting udev Kernel Device Manager...
Mounting Kernel Configuration File System...
Starting Apply Kernel Variables...
Mounting FUSE Control File System...
[ OK ] Mounted Kernel Configuration File System.
[ OK ] Mounted FUSE Control File System.
[ OK ] Started LVM2 metadata daemon.
[ OK ] Started Apply Kernel Variables.
[ OK ] Started udev Kernel Device Manager.
[ OK ] Started Dispatch Password Requests to Console Directory Watch.
[ OK ] Reached target Local Encrypted Volumes.
[ OK ] Started Monitoring of LVM2 mirrors,...sing dmeventd or progress polling.
[ OK ] Reached target Local File Systems (Pre)

```

Figure 10-9. Linux serial console

Connection Troubleshooting

The last option that we will learn about is connection troubleshooting, which allows us to check and test if a port we are trying to open or block is allowed or blocked.

The connection troubleshooting feature comes with a simple interface, as shown in Figure 10-10. All we need to do is type the source IP address, which can be any IP address, or the VM IP address.

Next, we select the service, port, and protocol. When ready, we click on the “Test connection” button.

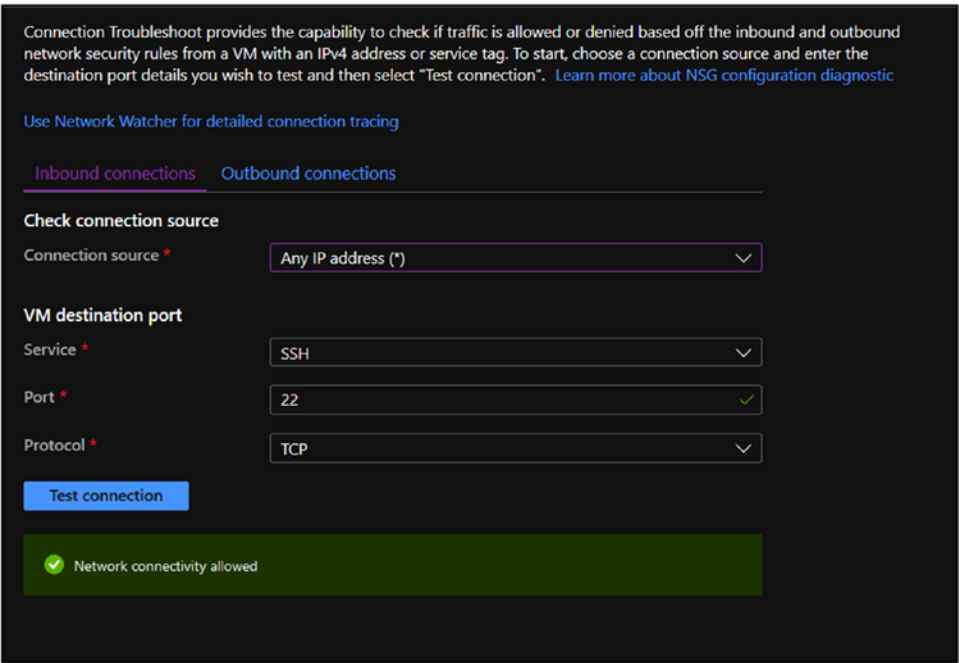


Figure 10-10. Connection troubleshooting

Summary

In this last chapter, we learned how to troubleshoot performance, networking, and other issues you might come across when you run containerized applications in Microsoft Azure.

Index

A

Active Directory (AD)

- audit logs, [165](#)
- CLI az ad, [166](#)
- login Details section, [164](#)
- premium licensing, [163](#)
- services, [162](#)
- sign-ins page, [164–167](#)

Alert rule, [221](#)

APPS_NSG, [191](#)

Autoscaling

- CLI, [201](#)
- delete, [202](#)
- disable autoscale, [202](#)
- feature, [200, 201](#)
- update, [202](#)

Auto-start and auto-shutdown host

- configuration, [142–144](#)
- features, [141](#)

Azure

- account informations, [18](#)
 - global administrator accounts, [22](#)
 - MFA (*see* Multi-factor authentication (MFA))

Cloud Shell

- cluster, [108](#)
- code editor, [15–17](#)
- created screen, [11](#)
- editor icon, [16](#)
- file share, [18](#)

icon, [9](#)

PowerShell, [14](#)

storage account setup, [10](#)

upload/download scripts, [17, 18](#)

version command, [18](#)

welcome page, [10](#)

CLI, [12](#)

Bash and PowerShell, [14](#)

confirmation screen, [14](#)

help menu, [12](#)

resource list command, [13](#)

shells, [14](#)

container services, [23–27](#)

subscription

assigning permissions, [4–9](#)

services, [1, 2](#)

sign up, [2–4](#)

AzureBastionSubnet, [169](#)

Azure CLI, [53–55, 68](#)

Azure Container Instances (ACI), [23–24, 224–227](#)

az container commands, [59](#)

backup and recovery solutions, [266–268](#)

container groups, [60](#)

containerized application, [76–83](#)

az container command line, [77](#)

deployment option, [81–83](#)

image action menu, [80](#)

memory and CPU options, [81](#)

INDEX

Azure Container Instances (ACI) (*cont.*)

- size of, [79](#)
 - source code, [78](#)
 - infrastructure, [57](#)
 - instance creation, [58](#)
 - limitation, [60](#)
 - Linux and Windows containers
 - (*see* Linux containers;
Windows containers)
 - management options
 - az container commands, [94, 95](#)
 - Bash Shell ls command, [94](#)
 - connect tab, [94](#)
 - container group page, [90](#)
 - deployments, [89](#)
 - events option, [90, 91](#)
 - logs tab, [93](#)
 - properties tab, [92](#)
 - monitoring tools, [83](#)
 - area chart options, [87](#)
 - CPU usage, [86](#)
 - drop-down menus, [85](#)
 - exported data, [88](#)
 - metrics, [84](#)
 - overview page, [89](#)
 - RAM average, [86](#)
 - share options, [88](#)
 - mount storage volume (*see* Mount
storage volumes)
 - portal, [58](#)
 - resource explorer portal, [96, 97](#)
 - scaling service, [202–205](#)
 - troubleshooting services, [283, 284](#)
- ### Azure Container Registry (ACR), [25–26](#)
- backup and recovery
 - solutions, [268, 269](#)
 - Docker Desktop download page, [29, 30](#)
 - elastic service, [28](#)

- Kubernetes service, [108, 109](#)
- monitoring services, [227–229](#)
- portal setup
 - CLI, [33](#)
 - container registries page, [33](#)
 - container registries service, [30](#)
 - details, [31](#)
 - registries page, [31](#)
- pricing details, [28](#)
- pull images
 - Docker command, [39](#)
 - downloading image, [39](#)
 - Nginx command, [38](#)
 - process of, [38](#)
- push images
 - access keys, [35](#)
 - Docker login command, [36](#)
 - login details, [34](#)
 - Nginx image, [36](#)
 - process of, [35](#)
 - repositories, [37](#)
 - tags and action menu, [38](#)
- requirements, [29](#)
- scaling service, [205](#)
- troubleshooting services, [284, 285](#)
- use of, [27](#)
- VS Code (*see* Visual Studio
Code (VS Code))

azurefile, [119, 256–257](#)

Azure Kubernetes Service (AKS), *see*
Kubernetes Service (AKS)

B

- Backup and recovery solutions
 - ACI container menu, [267–269](#)
 - acifileshare, [268](#)
 - ACR service, [268, 269](#)

- container VM host
 - backup option, [269, 270](#)
 - CLI source code, [273](#)
 - enable screen, [270, 271](#)
 - jobs and operations, [272](#)
 - restore function, [273–276](#)
 - retention period, [272](#)
- delete vault option, [276–278](#)
- easy-to-access menu, [248](#)
- features, [247](#)
- Kubernetes service
 - backup file share, [256–260](#)
 - CLI, [256](#)
 - creation wizard, [252](#)
 - GRS (*see* Geo-redundant storage (GRS))
 - replication configuration, [253](#)
 - restore file shares, [260–264](#)
 - source code, [250](#)
 - vault option, [251, 252](#)
- monitoring option, [264–266](#)
- replication and retention policies, [269](#)
- resources, [248](#)
- site recovery, [249, 250](#)
- Bastion, [167](#)
 - IP address, [167](#)
 - main page, [176](#)
 - management, [175, 176](#)
 - PowerShell, [177](#)
 - pricing, [167](#)
 - VM
 - configuration, [170](#)
 - connect option, [168](#)
 - menu option, [169](#)
 - option and page creation, [170](#)
 - overview page, [168](#)

- public IP address configuration, [172](#)
- resource group option, [173](#)
- subnet page, [171](#)
- username and password
 - option, [173, 174](#)
 - web browser, [174, 175](#)
- Windows host networking, [168](#)

C

- CLI az ad, dbadministrator, [166](#)
- Container host, [129–130](#)
- Containers
 - active directory, [162–166](#)
 - Bastion (*see* Bastion)
 - features, [161](#)
 - NSG (*see* Network Security Groups (NSG))
 - scaling service
 - CLI, [207](#)
 - portal, [206](#)
 - size, [206](#)
 - secure score, [182–184](#)
 - security center, [177–181](#)

D, E

- Docker container host
 - management
 - auto-start and auto-shutdown, [141–144](#)
 - features, [141](#)
 - start/stop host, [144](#)
 - Ubuntu VM, [130](#)
 - use of, [129](#)
 - Windows Server 2019 VM, [137–141](#)

F

File shares

- backup option
 - backup screen, [257](#)
 - console page, [257](#)
 - definition, [259](#)
 - goal page, [258](#)
 - overview page, [256](#)
 - policy page, [259](#)
 - selection, [259](#)
 - storage account page, [258](#)
 - vaults page, [257](#)
- recovery services
 - backup tab, [261](#)
 - items page, [262](#)
 - location screen, [263](#), [264](#)
 - management type, [262](#)
 - overview page, [260](#)
 - restore point screen, [263](#)

G, H

Geo-redundant storage (GRS)

- backup configuration, [255](#)
- menu pane, [254](#)
- recovery services vaults, [254](#)

I, J

Identity and access management (IAM), [4](#)

- access control, [4](#), [6](#)
- role assignment page, [7](#), [8](#)
- subscriptions page, [5](#)

Identity layer, [162](#)

Insights (AKS monitor)

- cluster page, [216](#)
- containers, [218](#)

infrastructure health, [217](#)

node performance, [217](#)

solution, [216](#)

view live data, [219](#)

K

Kubernetes Service (AKS)

add-ons features, [99](#)

ACR, [108](#), [109](#)

cluster, [100](#)

authentication screen, [104](#)

cloud shell, [108](#)

cluster page, [107](#)

monitoring screen, [106](#)

networking configuration, [105](#)

node count, [103](#)

review setup

configuration, [106](#), [107](#)

scale options, [104](#)

service portal, [100](#), [101](#)

setup page creation, [101](#), [102](#)

software versions, [102](#), [103](#)

components, [98](#)

containers

ACR image, [109](#)

Nginx web server, [110](#)

source code, [111](#), [112](#)

definition, [98](#)

management options

access control page, [126](#)

cluster CPU and memory, [124](#)

containers tab, [126](#)

controllers tab, [125](#)

insights link, [124](#)

node pools page, [122](#)

overview page, [121](#)

- role assignments, [127](#)
 - scale menu, [123](#)
 - settings section, [121](#)
 - tab view (nodes), [124](#)
 - upgrade page, [123](#)
- master subcomponents, [98](#)
- monitoring services
 - bar chart, [220](#)
 - disable option, [223](#)
 - enable option, [214](#)
 - insights, [216–219](#)
 - log analytics, [214, 215](#)
 - logs, [221–223](#)
 - matrices, [219–221](#)
 - query editor, [223](#)
 - solutions, [215](#)
 - source code, [213](#)
- mount storage volumes (*see* Mount storage volumes)
- nodes, [99](#)
- scaling service
 - autoscaling, [200–202](#)
 - cluster, [196](#)
 - node pools page, [198–201](#)
- troubleshooting services
 - capacity, [281](#)
 - IP addresses, [282](#)
 - limits, [281](#)
 - pod issues, [282](#)
 - storage, [281](#)
- Web UI (dashboard), [112](#)
 - create page, [114](#)
 - dashboard, [112](#)
 - deployment, [115](#)
 - enabling dashboard, [113](#)
 - kubectl command
 - line, [113](#)
 - Nginx web server, [116](#)

- services, [116](#)
 - workloads statuses page, [115](#)

L

- Linux containers
 - configuration page, [65](#)
 - container instance
 - page, [61, 62](#)
 - deployment tool, [66](#)
 - features, [64](#)
 - networking screen, [63, 64](#)
 - Nginx container, [67](#)
 - overview page, [67](#)
 - review and create page, [65, 66](#)
 - setup page, [62, 63](#)
 - tags configuration page, [65](#)
 - Windows (*see* Windows containers)
- Locally redundant storage (LRS), [117](#)
- Log analytics, [222](#)

M

- Monitoring services
 - ACI services
 - CLI code, [226, 227](#)
 - main page, [224](#)
 - metrics, [226](#)
 - overview page, [225](#)
 - ACR services, [227](#)
 - metrics, [228](#)
 - options, [227](#)
 - storage information, [227](#)
 - total pull requests, [229](#)
 - activity logs, [213](#)
 - AKS cluster
 - disable option, [223](#)
 - enable option, [214](#)

INDEX

Monitoring services (*cont.*)

- insights, [216–219](#)
- log analytics, [214, 215](#)
- logs, [221–223](#)
- matrices, [219–221](#)
- solutions, [215](#)
- source code, [213](#)

logs, [212](#)

metrics, [212](#)

products, [212](#)

service option, [211](#)

VM

- alerts, [237, 238](#)
- connection monitor, [241–243](#)
- connections menu, [238–241](#)
- delete workspace, [246](#)
- disable, [244–246](#)
- insights, [231–241](#)
- log events, [236–238](#)
- logical map, [232, 233](#)
- multi-layer, [230](#)
- overview page, [230, 231](#)
- pane option, [230](#)
- performance counters, [235–237](#)
- snapshot view, [230](#)

Mount storage volumes

- data services, [70](#)
- deployment, [120–122](#)
- file share volume
 - bash script, [73](#)
 - container ID, [75](#)
 - contents, [75](#)
 - folder option, [74](#)
 - Nginx image, [73](#)
 - scripts, [71](#)
 - storage access key, [72](#)
 - variables details, [71](#)
- persistent storage, [119–121](#)

storage

- class, [117](#)
- RBAC role, [118](#)
- volume claim, [119](#)
- YAML files, [116](#)

Multi-factor authentication (MFA), [19](#)

- active directory icon, [19](#)
- directory users menu, [20](#)
- enable option, [20](#)
- page, [20](#)
- quick steps menu, [21](#)

N, O

Network Security Groups (NSG)

- creation, [186](#)
- features, [184](#)
- firewall service, [184](#)
- host VM
 - edit option, [190](#)
 - interface, [189, 190](#)
 - network Interfaces page, [192](#)
 - network rules, [189](#)
 - save configuration, [192](#)
 - selection, [191](#)
- inbound security rule, [188](#)
- overview page, [187](#)
- remote desktop services, [187](#)
- search bar, [185](#)
- security group, [185](#)

P, Q

Persistent volume claim (PVC), [119](#)

R

Role-based access control (RBAC), [4](#)

S

Scaling services, [195](#)

- ACI service, [202–205](#)
- AKS cluster
 - autoscale, [201, 202](#)
 - CLI, [201](#)
 - creation, [196](#)
 - delete, [202](#)
 - menu option, [200](#)
 - node pool page, [196–199](#)
 - portal, [199, 200](#)
 - update option, [202](#)
- ACR, [205](#)
- container, [205–207](#)
- container group properties, [77](#)
- containerized application, [76–83](#)
- resources and control costs, [195](#)
- spot virtual machines
 - CLI creation, [210](#)
 - deployment, [208, 209](#)
 - eviction policy, [209, 210](#)
 - infrastructure, [207](#)
 - pricing page, [208](#)

Secure score, [182–184](#)

Security center

- free tiers, [177–181](#)
- overview page, [180, 181](#)
- recommendations, [182](#)
- skip option, [180](#)
- standard tier, [178](#)
- upgrade button, [178](#)

Start/stop container host

- add automation account page, [146](#)
- automation accounts page, [145, 147](#)
- details, [145](#)
- enable option, [149](#)
- exclude single/multiple VM

- exclude VMs, [159](#)
- External_ExcludeVMnames, [158](#)
- solution parameters, [157](#)
- variables, [157](#)

log analytics workspace, [151](#)

monitor jobs, [159, 160](#)

off-hours page, [149](#)

parameters menu, [152, 153](#)

schedules page, [155–157](#)

search bar, [145](#)

selection option, [152](#)

solution page, [150, 154](#)

task solution, [144](#)

VM page, [148](#)

workspace, [150](#)

T

Tenant secure and safe, [177](#)

Troubleshooting services

- ACI services
 - bad performance, [284](#)
 - image size, [283](#)
 - location, [283](#)
 - slow start, [283](#)
 - versions, [284](#)
- ACR services
 - access issues, [285](#)
 - log details, [285](#)
 - performance issues, [285](#)
 - storage issues, [284](#)
- built-in tools, [280](#)
- connection, [293, 294](#)
- AKS
 - capacity issues, [281](#)
 - IP addresses, [282](#)
 - limits, [281](#)

INDEX

Troubleshooting services (*cont.*)

- pod issues, [282](#)
- storage, [281](#)
- Linux serial console, [293](#)
- serial console option, [292](#), [293](#)
- support plans page, [279](#), [280](#)
- virtualization, [286-293](#)

U

Ubuntu Linux virtual machines, [130](#)

- CLI and Cloud Shell, [132](#), [133](#)
- Docker version, [136](#), [137](#)
- IP addresses script, [133](#)
- portal creation, [130-132](#)
- SSH, [135](#), [136](#)
- snap application store, [135](#)
- virtual machine wizard creation, [131](#)

V

Virtual machines (VM)

- boot diagnostics menu, [287](#)
 - screenshot, [289](#)
 - serial log file, [290](#)
 - settings page, [288](#)
- connection monitor
 - add monitor, [242](#)
 - feature, [241](#)
 - network watcher, [242](#)
 - resource page, [243](#)
- insights, [231](#), [232](#)
 - alerts, [237](#), [238](#)
 - connections, [239-242](#)
 - enable option, [232](#)
 - log events screen, [235-237](#)
 - logical map, [232](#), [233](#)

properties page, [233-235](#)

steps, [231](#)

monitoring services

- delete workspace, [246](#)
- disable, [244-246](#)
- extensions, [244](#)
- insights, [231](#), [232](#)
- log analytics workspaces, [245](#)
- overview page, [230](#), [231](#)
- uninstall extension, [245](#)

troubleshooting services

- boot diagnostics menu, [287](#)
- connection, [293](#), [294](#)
- redeployment, [290-292](#)
- serial console option, [292](#), [293](#)
- support menu, [286](#)

Virtual machine (VM)

- backup and recovery option, [269](#), [273-279](#)
- NSG network interface, [189](#)
- scaling service, [205-207](#)

Visual Studio Code (VS Code)

- CLI, [53-55](#)
- container registry
 - account extension screen, [45](#)
 - ACR registry, [47](#)
 - command palette, [49](#)
 - connect registry, [44](#), [45](#)
 - Docker extension, [48](#)
 - installation screen, [45](#)
 - login page, [47](#)
 - menu option, [46](#)
 - refresh button, [46](#)
 - right-click menu, [48](#)
 - sign out command, [49](#)
- container menu, [43](#)
- Docker extensions

- components, [42](#)
 - Microsoft, [41](#)
 - search box, [40](#), [41](#)
- extension option, [44](#)
- homepage, [40](#)
- installation, [40](#)
- managing containers, [43](#)
- secure, [49](#)
- upgrade SKU plan
 - access keys, [52](#), [53](#)
 - configuration page, [52](#)
 - container registry page, [50](#)
 - firewall and virtual networks
 - page, [51](#)
 - password option, [53](#)
 - update option, [49](#)

W, X, Y, Z

- Web apps containers
 - deployment option, [81](#), [82](#)
 - service plans, [82](#), [83](#)
- Windows containers
 - deployment, [68](#)
 - IIS homepage, [69](#)
 - images, [60](#)
 - OS type, [68](#)
- Windows Server 2019 virtual machine, [137](#)
 - CLI commands, [138](#), [139](#)
 - connection, [139](#)
 - portal window, [137](#)
 - remote desktop, [139](#)
 - update option, [140](#), [141](#)