Contents

1 AZ-303: Microsoft Azure Architect Technologies

- **Link to LABS in HTML format**
- **Download Latest Student Handbook and AllFiles Content**
- **Are you a MCT?** - Have a look at our GitHub User Guide for MCTs
- **Need to manually build the lab instructions?** - Instructions are available in the MicrosoftLearning/Docker-Build repository

1.1 What are we doing?

- To support this course, we will need to make frequent updates to the course content to keep it current with the Azure services used in the course. We are publishing the lab instructions and lab files on GitHub to allow for open contributions between the course authors and MCTs to keep the content current with changes in the Azure platform.
- We hope that this brings a sense of collaboration to the labs like we've never had before - when Azure changes and you find it first during a live delivery, go ahead and make an enhancement right in the lab

source. Help your fellow MCTs.

1.2 How should I use these files relative to the released MOC files?

- The instructor handbook and PowerPoints are still going to be your primary source for teaching the course content.
- These files on GitHub are designed to be used in conjunction with the student handbook, but are in GitHub as a central repository so MCTs and course authors can have a shared source for the latest lab files.
- It will be recommended that for every delivery, trainers check GitHub for any changes that may have been made to support the latest Azure services, and get the latest files for their delivery.

1.3 What about changes to the student handbook?

- We will review the student handbook on a quarterly basis and update through the normal MOC release channels as needed.

1.4 How do I contribute?

- Any MCT can submit a pull request to the code or content in the GitHub repro, Microsoft and the course author will triage and include content and lab code changes as needed.
- You can submit bugs, changes, improvement and ideas. Find a new Azure feature before we have? Submit a new demo!

1.5 Notes

1.5.1 Classroom Materials

It is strongly recommended that MCTs and Partners access these materials and in turn, provide them separately to students. Pointing students directly to GitHub to access Lab steps as part of an ongoing class will require them to access yet another UI as part of the course, contributing to a confusing experience for the student. An explanation to the student regarding why they are receiving separate Lab instructions can highlight the nature of an always-changing cloud-based interface and platform. Microsoft Learning support for accessing files on GitHub and support for navigation of the GitHub site is limited to MCTs teaching this course only.

2 AZ-303 Online Lab List

Lab files list and course module order (11/20/20).

Module 05 Lab

- Lab: Implementing Highly Available Azure IaaS Compute Architecture
- Current file name: **Module_05_Lab.md**
- https://aka.ms/303_Module_05_Lab
- Previous file name: *Module_4_Lab.md*

Module 06 Lab

Lab title: Implementing and Configuring Azure Storage File and Blob Services

- Current file name: **Module_06_Lab.md**
- https://aka.ms/303_Module_06_Lab
- Previous file name: *Module_5_Lab.md*

Module 10 Lab

Lab title: Managing Azure Role-Based Access Control

- Current file name: **Module_10_Lab.md**
- https://aka.ms/303_Module_10_Lab

- Previous file name: *Module_7_Lab.md*

Module 12 Lab

Lab title: Protecting Hyper-V VMs by using Azure Site Recovery

- Current file name: **Module_12_Lab_a.md**
- https://aka.ms/303_Module_12_Lab
- Previous file name: *Module_9_Lab.md*

Module 14 lab (a)

Lab title: Implementing an Azure App Service Web App with a Staging Slot

- Current file name: **Module_14_Lab_a.md**
- https://aka.ms/303_Module_14a_Lab
- Previous file name: *Module_12_Lab_a.md*

Module 14 lab (b)

Lab title: Configuring a Message-Based Integration Architecture

- Current file name: **Module_14_Lab_b.md**
- https://aka.ms/303_Module_14b_Lab
- Previous file name: *Module_12_Lab_b.md*

2.1 title: Online Hosted Instructions permalink: index.html layout: home

3 Content Directory

Files necessary to complete the labs can be DOWNLOADED HERE

Hyperlinks to each of the lab exercises are listed below.

3.1 Labs

{% assign labs = site.pages | where_exp:"page", "page.url contains '/Instructions/Labs'" %} | Module | Lab | | --- | --- | {% for activity in labs %}| {{ activity.lab.module }} | [{{ activity.lab.title }}{% if activity.lab.type %} - {{ activity.lab.type }}{% endif %}](/home/ll/Azure_clone/Azure_new/AZ-303-Microsoft-Azure-Architect-Technologies/{{ site.github.url }}{{ activity.url }}) | {% endfor %}

4 AZ-303: Lab file names in GitHub

Source repo: https://github.com/MicrosoftLearning/AZ-303-Microsoft-Azure-Architect-Technologies

Module 4 lab

Lab title: Implementing Highly Available Azure IaaS Compute Architecture

- Current file name: **Module_4_Lab.md**
- Previous file name: LAB_01_Deploying Azure IaaS Solutions.md

Module 5 lab

Lab title: Implementing and Configuring Azure Storage File and Blob Services

- Current file name: **Module_5_Lab.md**
- Previous file name: LAB_02_Data Storage.md

Module 7 lab

Lab title: Managing Azure Role-Based Access Control

- Current file name: **Module_7_Lab.md**
- Previous file name: LAB_11_Security.md

Module 9 lab

Lab title: Protecting Hyper-V VMs by using Azure Site Recovery

- Current file name: **Module_9_Lab.md**
- Previous file name: LAB_07_Migrating to Azure.md

Module 12 lab (a)

Lab title: Implementing an Azure App Service Web App with a Staging Slot

- Current file name: **Module_12_Lab_a.md**
- Previous file name: LAB_05_Manage Compute_PaaS.md

Module 12 lab (b)

Lab title: Configuring a Message-Based Integration Architecture

- Current file name: **Module_12_Lab_b.md**
- Previous file name: LAB_09_Migrating to Azure.md

4.1 lab: title: '05: Implementing Highly Available Azure IaaS Compute Architecture' module: 'Module 05: Implement Load Balancing and Network Security'

5 Lab: Implementing Highly Available Azure IaaS Compute Architecture

6 Student lab manual

6.1 Lab scenario

Adatum Corporation has several on-premises workloads running on a mix of physical servers and virtual machines. Most of the workloads require some level of resiliency, including a range of high availability SLAs. Most of the workloads leverage either Windows Server Failover Clustering or Linux Corosync clusters and Pacemaker resource manager, with synchronous replication between cluster nodes. Adatum is trying to determine how the equivalent functionality can be implemented in Azure. In particular, the Adatum Enterprise Architecture team is exploring the Azure platform capabilities that accommodate high availability requirements within the same data center and between data centers in the same region.

In addition, the Adatum Enterprise Architecture team realizes that resiliency alone might not be sufficient to provide the level of availability expected by its business operations. Some of the workloads have highly dynamic usage patterns, which are currently addressed based on continuous monitoring and custom scripting solutions, automatically provisioning and deprovisioning additional cluster nodes. Usage patterns of others are more predictable, but also need to be occasionally adjusted to account for increase demand for disk space, memory, or processing resources.

To accomplish these objectives, the Architecture team wants to test a range of highly available IaaS compute deployments, including:

- Availability sets-based deployment of Azure VMs behind an Azure Load Balancer Basic
- Zone-redundant deployment of Azure VMs behind an Azure Load Balancer Standard
- Zone-redundant deployment of Azure VM scale sets behind an Azure Application Gateway
- Automatic horizontal scaling of Azure VM scale sets (autoscaling)
- Manual vertical scaling (compute and storage) of Azure VM scale sets

An availability set represents a logical grouping of Azure VMs which controls their physical placement within the same Azure datacenter. Azure makes sure that the VMs within the same availability set run across multiple physical servers, compute racks, storage units, and network switches. If a hardware or software failure happens, only a subset of your VMs are impacted and your overall solution stays operational. Availability Sets are essential for building reliable cloud solutions. With availability sets, Azure offers 99.95% VM uptime SLA.

Availability zones represent unique physical locations within a single Azure region. Each zone is made up of one or more datacenters equipped with independent power, cooling, and networking. The physical separation of availability zones within a region protects applications and data from datacenter failures. Zone-redundant services replicate your applications and data across availability zones to protect from single-points-of-failure. With availability zones, Azure offers 99.99% VM uptime SLA.

Azure virtual machine scale sets let you create and manage a group of identical, load balanced VMs. The number of VM instances can automatically increase or decrease in response to demand or a defined schedule. Scale sets provide high availability to your applications, and allow you to centrally manage, configure, and update many VMs. With virtual machine scale sets, you can build large-scale services for areas such as compute, big data, and container workloads.

6.2 Objectives

After completing this lab, you will be able to:

- Describe characteristics of highly available Azure VMs residing in the same availability set behind a Azure Load Balancer Basic
- Describe characteristics of highly available Azure VMs residing in different availability zones behind an Azure Load Balancer Standard
- Describe characteristics of automatic horizontal scaling of Azure VM Scale Sets
- Describe characteristics of manual vertical scaling of Azure VM Scale Sets

6.3 Lab Environment

Windows Server admin credentials

- User Name: **Admin**
- Password: **Pa55w.rd**

Estimated Time: 120 minutes

6.4 Lab Files

- \AZ303\AllFiles\Labs\05\azuredeploy30305suba.json
- \AZ303\AllFiles\Labs\05\azuredeploy30305rga.json
- \AZ303\AllFiles\Labs\05\azuredeploy30305rga.parameters.json
- \AZ303\AllFiles\Labs\05\azuredeploy30305rgb.json
- \AZ303\AllFiles\Labs\05\azuredeploy30305rgb.parameters.json
- \AZ303\AllFiles\Labs\05\azuredeploy30305rgc.json
- \AZ303\AllFiles\Labs\05\azuredeploy30305rgc.parameters.json
- \AZ303\AllFiles\Labs\05\az30305e-configure_VMSS_with_data_disk.ps1

6.5 Instructions

6.5.1 Exercise 1: Implement and analyze highly available Azure VM deployments using availability sets and Azure Load Balancer Basic

The main tasks for this exercise are as follows:

1. Deploy highly available Azure VMs into an availability set behind an Azure Load Balancer Basic by using Azure Resource Manager templates
2. Analyze highly available Azure VMs deployed into an availability set behind an Azure Load Balancer Basic
3. Remove Azure resources deployed in the exercise

6.5.1.1 Task 1: Deploy highly available Azure VMs into an availability set behind an Azure Load Balancer Basic by using Azure Resource Manager templates

1. From your lab computer, start a web browser, navigate to the Azure portal, and sign in by providing credentials of a user account with the Owner role in the subscription you will be using in this lab.
2. In the Azure portal, open the **Cloud Shell** pane by selecting on the toolbar icon directly to the right of the search textbox.
3. If prompted to select either **Bash** or **PowerShell**, select **Bash**.

   Note: If this is the first time you are starting **Cloud Shell** and you are presented with the **You have no storage mounted** message, select the subscription you are using in this lab, and select **Create storage**.

4. From the Cloud Shell pane, run the following to register the Microsoft.Insights resource provider in preparation for the upcoming exercises in this lab:

   ```
   az provider register --namespace 'Microsoft.Insights'
   ```

5. In the toolbar of the Cloud Shell pane, select the **Upload/Download files** icon, in the drop-down menu select **Upload**, and upload the file **\\AZ303\AllFiles\Labs\05\azuredeploy30305suba.json** into the Cloud Shell home directory.
6. From the Cloud Shell pane, run the following to designate the Azure region you will be using in this lab (replace the `<Azure region>` placeholder with the name of the Azure region that is available for deployment of Azure VMs in your subscription and which is closest to the location of your lab computer):

   ```
   LOCATION='<Azure region>'
   ```

   Note: To identify Azure regions where you can provision Azure VMs, refer to **https://azure.microsoft.com/en-us/regions/offers/**

   Note: To identify the names of the Azure regions to use when setting the value of the **LOCATION** variable, run `az account list-locations --query "[].{name:name}" -o table`. Make sure to use the notation which does not include a space, e.g. **eastus** rather than **US East**.

7. From the Cloud Shell pane, run the following to create an instance of Network Watcher in preparation for the upcoming exercises in this lab:

   ```
   az network watcher configure --resource-group NetworkWatcherRG --locations $LOCATION --enabled -o
   ```

   Note: If you receive an error indicating there is no "NetworkWatcherRG" resource group, create a resource group from the portal named NetworkWatcherRG and rerun the command.

8. From the Cloud Shell pane, run the following to create a resource group in the designated Azure region.

   ```
   az deployment sub create \
   --location $LOCATION \
   --template-file azuredeploy30305suba.json \
   --parameters rgName=az30305a-labRG rgLocation=$LOCATION
   ```

9. From the Cloud Shell pane, upload the Azure Resource Manager template **\\AZ303\AllFiles\Labs\05\azuredeploy**
10. From the Cloud Shell pane, upload the Azure Resource Manager parameter file **\\AZ303\AllFiles\Labs\05\azurede**
11. From the Cloud Shell pane, run the following to deploy an Azure Load Balancer Basic with its backend pool consisting of a pair of Azure VMs hosting Windows Server 2019 Datacenter Core into the same availability set:

    ```
    az deployment group create \
    --resource-group az30305a-labRG \
    --template-file azuredeploy30305rga.json \
    --parameters @azuredeploy30305rga.parameters.json
    ```

    Note: Wait for the deployment to complete before proceeding to the next task. This should take about 10 minutes.

12. In the Azure portal, close the **Cloud Shell** pane.

6.5.1.2 Task 2: Analyze highly available Azure VMs deployed into an availability set behind an Azure Load Balancer Basic

1. In the Azure portal, search for and select **Network Watcher** and, on the **Network Watcher** blade, select **Topology**.
2. On the **Network Watcher | Topology** blade, specify the following settings:

Setting	Value
Subscription	the name of the Azure subscription you are using in this lab
Resource Group	**az30305a-labRG**
Virtual Network	**az30305a-vnet**

3. Review the resulting topology diagram, noting the connections between the public IP address, load balancer, and the network adapters of Azure VMs in its backend pool.
4. On the **Network Watcher** blade, select **Effective security rules**.
5. On the **Network Watcher | Effective security rules** blade, specify the following settings:

Setting	Value
Subscription	the name of the Azure subscription you are using in this lab
Resource group	**az30305a-labRG**
Virtual machine	**az30305a-vm0**
Network interface	**az30305a-nic0**

6. Review the associated network security group and the effective security rules, including two custom rules that allow inbound connectivity via RDP and HTTP.

   Note: Alternatively, you can view **Effective security rules** from:

- the **az30305a-nic0** network interface blade.
- the **az30305a-web-nsg** network security group blade

7. On the **Network Watcher** blade, select **Connection troubleshoot**.

   Note: The intention is to verify the proximity (in the networking terms) of the two Azure VMs in the same availability set.

8. On the **Network Watcher | Connection troubleshoot** blade, specify the following settings and select **Check** :

Setting	Value
Subscription	the name of the Azure subscription you are using in this lab
Resource group	**az30305a-labRG**
Source type	**Virtual machine**
Virtual machine	**az30305a-vm0**
Destination	**Select a virtual machine**
Resource group	**az30305a-labRG**
Virtual machine	**az30305a-vm1**
Protocol	**TCP**
Destination port	**80**

   Note: You will need to wait a few minutes for the results in order for the **Azure Network Watcher Agent** VM extension to be installed on the Azure VMs.

9. Review the results and note the latency of the network connection between the Azure VMs.

   Note: The latency should be about 1 millisecond, since both VMs are in the same availability set (within the same Azure datacenter).

10. In the Azure portal, navigate to the **az30305a-labRG** resource group blade, in the list of resources, select the **az30305a-avset** availability set entry, and on the **az30305a-avset** blade, note the fault domain and

update domain values assigned the two Azure VMs.

11. In the Azure portal, navigate back to the **az30305a-labRG** resource group blade, in the list of resources, select the **az30305a-lb** load balancer entry, and on the **az30305a-lb** blade, note the public IP address entry.
12. In the Azure portal, start a **Bash** session in the Cloud Shell pane.
13. From the Cloud Shell pane, run the following to test load balancing of HTTP traffic to the Azure VMs in the backend pool of the Azure load balancer (replace the `<lb_IP_address>` placeholder with the IP address of the front end of the load balancer you identified earlier):

    ```
    for i in {1..4}; do curl <lb_IP_address>; done
    ```

    > **Note**: Verify that the returned messages indicate that the requests are being delivered in the round robin manner to the backend Azure VMs

14. On the **az30305a-lb** blade, select the **Load balancing rules** entry and, on the **az30305a-lb | Load balancing rules** blade, select the **az303005a-lbruletcp80** entry representing the load balancing rule handling HTTP traffic.
15. On the **az303005a-lbruletcp80** blade, in the **Session persistence** drop-down list, select **Client IP** and then select **Save**.
16. Wait for the update to complete and, from the Cloud Shell pane, re-run the following to test load balancing of HTTP traffic to the Azure VMs in the backend pool of the Azure load balancer with session persistence (replace the `<lb_IP_address>` placeholder with the IP address of the front end of the load balancer you identified earlier):

    ```
    for i in {1..4}; do curl <lb_IP_address>; done
    ```

    > **Note**: Verify that the returned messages indicate that the requests are being delivered to the same backend Azure VMs

17. In the Azure portal, navigate back to the **az30305a-lb** blade, select the **Inbound NAT rules** entry and note the two rules that allow for connecting to the first and the second of the backend pool VMs via Remote Desktop over TCP ports 33890 and 33891, respectively.
18. From the Cloud Shell pane, run the following to test Remote Desktop connectivity via NAT to the first Azure VM in the backend pool of the Azure load balancer (replace the `<lb_IP_address>` placeholder with the IP address of the front end of the load balancer you identified earlier):

    ```
    curl -v telnet://<lb_IP_address>:33890
    ```

    > **Note**: Verify that the returned message indicates that you are successfully connected.

19. Press the **Ctrl+C** key combination to return to the Bash shell prompt and run the following to test Remote Desktop connectivity via NAT to the second Azure VM in the backend pool of the Azure load balancer (replace the `<lb_IP_address>` placeholder with the IP address of the front end of the load balancer you identified earlier):

    ```
    curl -v telnet://<lb_IP_address>:33891
    ```

    > **Note**: Verify that the returned message indicates that you are successfully connected.

20. Press the **Ctrl+C** key combination to return to the Bash shell prompt.

6.5.1.3 Task 3: Remove Azure resources deployed in the exercise

1. From the Cloud Shell pane, run the following to list the resource group you created in this exercise:

    ```
    az group list --query "[?starts_with(name,'az30305a-')]".name --output tsv
    ```

    > **Note**: Verify that the output contains only the resource group you created in this lab. This group will be deleted in this task.

2. From the Cloud Shell pane, run the following to delete the resource group you created in this lab

    ```
    az group list --query "[?starts_with(name,'az30305a-')]".name --output tsv | xargs -L1 bash -c 'az
    ```

3. Close the Cloud Shell pane.

6.5.2 Exercise 2: Implement and analyze highly available Azure VM deployments using availability zones and Azure Load Balancer Standard

The main tasks for this exercise are as follows:

1. Deploy highly available Azure VMs into availability zones behind an Azure Load Balancer Standard by using Azure Resource Manager templates
2. Analyze highly available Azure VMs deployed across availability zones behind an Azure Load Balancer Standard
3. Remove Azure resources deployed in the exercise

6.5.2.1 Task 1: Deploy highly available Azure VMs into availability zones behind an Azure Load Balancer Standard by using Azure Resource Manager templates

1. If needed, in the Azure portal, open **Cloud Shell** pane by selecting on the toolbar icon directly to the right of the search textbox.
2. If prompted to select either **Bash** or **PowerShell**, select **Bash**.
3. In the toolbar of the Cloud Shell pane, select the **Upload/Download files** icon, in the drop-down menu select **Upload**, and upload the file **\\AZ303\AllFiles\Labs\05\azuredeploy30305subb.json** into the Cloud Shell home directory.
4. From the Cloud Shell pane, run the following to create a resource groups (replace the `<Azure region>` placeholder with the name of the Azure region that is available in your subscription and which is closest to the location of your lab computer):

   ```
   LOCATION='<Azure region>'
   ```

   ```
   az deployment sub create \
   --location $LOCATION \
   --template-file azuredeploy30305subb.json \
   --parameters rgName=az30305b-labRG rgLocation=$LOCATION
   ```

5. From the Cloud Shell pane, upload the Azure Resource Manager template **\\AZ303\AllFiles\Labs\05\azuredeploy**
6. From the Cloud Shell pane, upload the Azure Resource Manager parameter file **\\AZ303\AllFiles\Labs\05\azurede**
7. From the Cloud Shell pane, run the following to deploy an Azure Load Balancer Standard with its backend pool consisting of a pair of Azure VMs hosting Windows Server 2019 Datacenter Core across two availability zones:

   ```
   az deployment group create \
   --resource-group az30305b-labRG \
   --template-file azuredeploy30305rgb.json \
   --parameters @azuredeploy30305rgb.parameters.json
   ```

   > **Note**: Wait for the deployment to complete before proceeding to the next task. This should take about 10 minutes.

8. In the Azure portal, close the Cloud Shell pane.

6.5.2.2 Task 2: Analyze highly available Azure VMs deployed across availability zones behind an Azure Load Balancer Standard

1. In the Azure portal, search for and select **Network Watcher** and, on the **Network Watcher** blade, select **Topology**.
2. On the **Network Watcher | Topology** blade, specify the following settings:

Setting	Value
Subscription	the name of the Azure subscription you are using in this lab
Resource Group	**az30305b-labRG**
Virtual Network	**az30305b-vnet**

3. Review the resulting topology diagram, noting the connections between the public IP address, load bal-

ancer, and the network adapters of Azure VMs in its backend pool.

> **Note**: This diagram is practically identical to the one you viewed in the previous exercise, since, despite being in different zones (and effectively Azure data centers), the Azure VMs reside on the same subnet.

4. On the **Network Watcher** blade, select **Effective security rules**.
5. On the **Network Watcher | Effective security rules** blade, specify the following settings:

Setting	Value
Subscription	the name of the Azure subscription you are using in this lab
Resource group	**az30305b-labRG**
Virtual machine	**az30305b-vm0**
Network interface	**az30305b-nic0**

6. Review the associated network security group and the effective security rules, including two custom rules that allow inbound connectivity via RDP and HTTP.

> **Note**: This listing is also practically identical to the one you viewed in the previous exercise, with network-level protection implemented by using a network security group associated with the subnet to which both Azure VMs are connected. Keep in mind, however, that the network security group is, in this case, required for the HTTP and RDP traffic to reach the backend pool Azure VMs, due to the usage of the Azure Load Balancer Standard SKU (NSGs are optional when using the Basic SKU).

> **Note**: Alternatively, you can view **Effective security rules** from:

- the **az30305a-nic0** network interface blade.
- the **az30305a-web-nsg** network security group blade

7. On the **Network Watcher** blade, select **Connection troubleshoot**.

> **Note**: The intention is to verify the proximity (in the networking terms) of the two Azure VMs in different zones (within different Azure datacenters).

8. On the **Network Watcher | Connection troubleshoot** blade, specify the following settings and select **Check** :

Setting	Value
Subscription	the name of the Azure subscription you are using in this lab
Resource group	**az30305b-labRG**
Source type	**Virtual machine**
Virtual machine	**az30305b-vm0**
Destination	**Select a virtual machine**
Resource group	**az30305b-labRG**
Virtual machine	**az30305b-vm1**
Protocol	**TCP**
Destination port	**80**

> **Note**: You will need to wait a few minutes for the results in order for the **Azure Network Watcher Agent** VM extension to be installed on the Azure VMs.

9. Review the results and note the latency of the network connection between the Azure VMs.

> **Note**: The latency might be slightly higher than the one you observed in the previous exercise, since the two VMs are in different zones (within different Azure datacenters).

10. In the Azure portal, navigate to the **az30305b-labRG** resource group blade, in the list of resources, select the **az30305b-vm0** virtual machine entry, and on the **az30305b-vm0** blade, note the **Location** and **Availability zone** entries.
11. In the Azure portal, navigate to the **az30305b-labRG** resource group blade, in the list of resources, select the **az30305b-vm1** virtual machine entry, and on the **az30305b-vm1** blade, note the **Location** and **Availability zone** entries.

> **Note**: The entries you reviewed confirm that each Azure VM resides in a different availability zone.

12. In the Azure portal, navigate to the **az30305b-labRG** resource group blade and, in the list of resources, select the **az30305b-lb** load balancer entry, and on the **az30305b-lb** blade, note the public IP address entry.
13. In the Azure portal, start a new **Bash** session in the Cloud Shell pane.
14. From the Cloud Shell pane, run the following to test load balancing of HTTP traffic to the Azure VMs in the backend pool of the Azure load balancer (replace the `<lb_IP_address>` placeholder with the IP address of the front end of the load balancer you identified earlier):

    ```
    for i in {1..4}; do curl <lb_IP_address>; done
    ```

    > **Note**: Verify that the returned messages indicate that the requests are being delivered in the round robin manner to the backend Azure VMs

15. On the **az30305b-lb** blade, select the **Load balancing rules** entry and, on the **az30305b-lb | Load balancing rules** blade, select the **az303005b-lbruletcp80** entry representing the load balancing rule handling HTTP traffic.
16. On the **az303005b-lbruletcp80** blade, in the **Session persistence** drop-down list, select **Client IP** and then select **Save**.
17. Wait for the update to complete and, from the Cloud Shell pane, re-run the following to test load balancing of HTTP traffic to the Azure VMs in the backend pool of the Azure load balancer with session persistence (replace the `<lb_IP_address>` placeholder with the IP address of the front end of the load balancer you identified earlier):

    ```
    for i in {1..4}; do curl <lb_IP_address>; done
    ```

    > **Note**: Verify that the returned messages indicate that the requests are being delivered to the same backend Azure VMs

18. In the Azure portal, navigate back to the **az30305b-lb** blade, select the **Inbound NAT rules** entry and note the two rules that allow for connecting to the first and the second of the backend pool VMs via Remote Desktop over TCP ports 33890 and 33891, respectively.
19. From the Cloud Shell pane, run the following to test Remote Desktop connectivity via NAT to the first Azure VM in the backend pool of the Azure load balancer (replace the `<lb_IP_address>` placeholder with the IP address of the front end of the load balancer you identified earlier):

    ```
    curl -v telnet://<lb_IP_address>:33890
    ```

    > **Note**: Verify that the returned message indicates that you are successfully connected.

20. Press the **Ctrl+C** key combination to return to the Bash shell prompt and run the following to test Remote Desktop connectivity via NAT to the second Azure VM in the backend pool of the Azure load balancer (replace the `<lb_IP_address>` placeholder with the IP address of the front end of the load balancer you identified earlier):

    ```
    curl -v telnet://<lb_IP_address>:33891
    ```

    > **Note**: Verify that the returned message indicates that you are successfully connected.

21. Press the **Ctrl+C** key combination to return to the Bash shell prompt and close the Cloud Shell pane.
22. On the **az30305b-lb** blade, select the **Load balancing rules** entry and, on the **az30305b-lb | Load balancing rules** blade, select the **az303005b-lbruletcp80** entry representing the load balancing rule handling HTTP traffic.
23. On the **az303005b-lbruletcp80** blade, in the **Outbound source network address translation (SNAT)** section, select **(Recommended) Use outbound rules to provide backend pool members access to the internet**, and then select **Save**.
24. Navigate back to the **az30305b-lb** blade, select the **Outbound rules** entry, and on the **az30305b-lb | Outbound rules** blade, select **+ Add**.
25. On the **Add outbound rule** blade, specify the following settings and select **Add** (leave all other settings with their default values):

Setting	Value
Name	**az303005b-obrule**
Frontend IP address	the name of the existing frontend IP address of the **az30305b-lb** load balancer
Backend pool	**az30305b-bepool**
Port allocation	**Manually choose number of outbound ports**
Choose by	**Maximum number of backend instances**
Maximum number of backend instances	**3**

> **Note**: Azure Load Balancer Standard allows you to designate a dedicated frontend IP address for outbound traffic (in cases where multiple frontend IP addresses are assigned).

26. In the Azure portal, navigate to the **az30305b-labRG** resource group blade, in the list of resources, select the **az30305b-vm0** virtual machine entry, and on the **az30305b-vm0** blade, in the **Operations** blade, select **Run command**.
27. On the **az30305b-vm0 | Run command** blade, select **RunPowerShellScript**.
28. On the **Run Command Script** blade, in the **PowerShell Script** text box, type the following and select **Run**.

    ```
    (Invoke-RestMethod -Uri "http://ipinfo.io").IP
    ```

    > **Note**: This command returns the public IP address from which the web request originates.

29. Review the output and verify that it matches the public IP address assigned to the frontend of the Azure Load Balancer Standard, which you assigned to the outbound load balancing rule.

6.5.2.3 Task 3: Remove Azure resources deployed in the exercise

1. In the Azure portal, start a new **Bash** session in the Cloud Shell pane.
2. From the Cloud Shell pane, run the following to list the resource group you created in this exercise:

    ```
    az group list --query "[?starts_with(name,'az30305b-')]".name --output tsv
    ```

    > **Note**: Verify that the output contains only the resource group you created in this lab. This group will be deleted in this task.

3. From the Cloud Shell pane, run the following to delete the resource group you created in this lab

    ```
    az group list --query "[?starts_with(name,'az30305b-')]".name --output tsv | xargs -L1 bash -c 'az
    ```

4. Close the Cloud Shell pane.

6.5.3 Exercise 3: Implement and analyze highly available Azure VM Scale Set deployments using availability zones and Azure Application Gateway.

The main tasks for this exercise are as follows:

1. Deploy a highly available Azure VM Scale Set into availability zones behind an Azure Application Gateway by using Azure Resource Manager templates
2. Analyze a highly available Azure VM Scale Set deployed across availability zones behind an Azure Application Gateway
3. Remove Azure resources deployed in the exercise

6.5.3.1 Task 1: Deploy a highly available Azure VM Scale Set into availability zones behind an Azure Application Gateway by using Azure Resource Manager templates

1. If needed, in the Azure portal, open **Cloud Shell** pane by selecting on the toolbar icon directly to the right of the search textbox.
2. If prompted to select either **Bash** or **PowerShell**, select **Bash**.
3. In the toolbar of the Cloud Shell pane, select the **Upload/Download files** icon, in the drop-down menu select **Upload**, and upload the file **\\AZ303\AllFiles\Labs\05\azuredeploy30305subc.json** into the Cloud Shell home directory.

4. From the Cloud Shell pane, run the following to create a resource groups (replace the `<Azure region>` placeholder with the name of the Azure region that is available in your subscription and which is closest to the location of your lab computer):

```
az deployment sub create --location '<Azure region>' --template-file azuredeploy30305subc.json --p
```

5. From the Cloud Shell pane, upload the Azure Resource Manager template \\AZ303\AllFiles\Labs\05\azuredeploy
6. From the Cloud Shell pane, upload the Azure Resource Manager parameter file \\AZ303\AllFiles\Labs\05\azurede
7. From the Cloud Shell pane, run the following to deploy an Azure Application Gateway with its backend pool consisting of a pair of Azure VMs hosting Windows Server 2019 Datacenter Core across different availability zones:

```
az deployment group create --resource-group az30305c-labRG --template-file azuredeploy30305rgc.jso
```

   Note: Wait for the deployment to complete before proceeding to the next task. This should take about 10 minutes.

8. In the Azure portal, close the Cloud Shell pane.

6.5.3.2 Task 2: Analyze a highly available Azure VM Scale Set deployed across availability zones behind an Azure Application Gateway

1. In the Azure portal, search for and select **Network Watcher** and, on the **Network Watcher** blade, select **Topology**.
2. On the **Network Watcher | Topology** blade, specify the following settings:

Setting	Value
Subscription	the name of the Azure subscription you are using in this lab
Resource Group	**az30305c-labRG**
Virtual Network	**az30305c-vnet**

3. Review the resulting topology diagram, noting the connections between the public IP address, load balancer, and the network adapters of Azure VM instances in the Azure Virtual Machine Scale Set in its backend pool.

   Note: In addition, deployment of an Azure Application Gateway requires a dedicated subnet, included in the diagram (although the gateway is not displayed).

   Note: In this configuration, it is not possible to use Network Watcher to view the effective network security rules (that is one of distinctions between Azure VMs and instances of an Azure VM Scale Set). Similarly, you cannot rely on using **Connection troubleshoot** to test network connectivity from Azure VM Scale Set instances, although it is possible to use it to test connectivity from the Azure Application Gateway.

4. In the Azure portal, navigate to the **az30305c-labRG** resource group blade, in the list of resources, and select the **az30305c-vmss** virtual machine scale set entry.
5. On the **az30305c-vmss** blade, note the **Location** and **Fault domains** entries.

   Note: Unlike Azure VMs, individual instances of Azure VM scale sets deploy into separate fault domains, including instances deployed into the same zone. In addition, they support 5 fault domains (unlike Azure VMs, which can use up to 3 fault domains).

6. On the **az30305c-vmss** blade, select **Instances**, on the **az30305c-vmss | Instances** blade, select the first instance, and identify its availability zone by reviewing the value of the **Location** property.
7. Navigate back to the **az30305c-vmss | Instances** blade, select the second instance, and identify its availability zone by reviewing the value of the **Location** property.

   Note: Verify that each instance resides in a different availability zone.

8. In the Azure portal, navigate to the **az30305c-labRG** resource group blade and, in the list of resources, select the **az30305c-appgw** load balancer entry, and on the **az30305c-appgw** blade, note the public IP address entry.
9. In the Azure portal, start a new **Bash** session in the Cloud Shell pane.

10. From the Cloud Shell pane, run the following to test load balancing of HTTP traffic to the Azure VM Scale Set instances in the backend pool of the Azure Application Gateway (replace the `<lb_IP_address>` placeholder with the IP address of the front end of the gateway you identified earlier):

    ```
    for i in {1..4}; do curl <appgw_IPaddress>; done
    ```

    > **Note**: Verify that the returned messages indicate that the requests are being delivered in the round robin manner to the backend Azure VMs

11. On the **az30305c-appgw** blade, select the **HTTP settings** entry and, on the **az30305c-appgw | HTTP settings** blade, select the **appGwBackentHttpSettings** entry representing the load balancing rule handling HTTP traffic.

12. On the **appGwBackentHttpSettings** blade, review the existing settings without making any changes and note that you can enable **Cookie-based affinity**.

    > **Note**: This feature requires that the client supports the use of cookies.

    > **Note**: You cannot use Azure Application Gateway to implement NAT for RDP connectivity to instances of an Azure VM Scale Set. Azure Application Gateway supports only HTTP/HTTPS traffic.

6.5.4 Exercise 4: Implementing autoscaling of Azure VM Scale Sets using availability zones and Azure Application Gateway.

The main tasks for this exercise are as follows:

1. Configuring autoscaling of an Azure VM Scale Set
2. Testing autoscaling of an Azure VM Scale Set

6.5.4.1 Task 1: Configure autoscaling of an Azure VM Scale Set

1. In the Azure portal, navigate to the **az30305c-labRG** resource group blade, in the list of resources, select the **az30305c-vmss** virtual machine scale set entry, and on the **az30305c-vmss** blade, select **Scaling**.
2. On the **az30305c-vmss | Scaling** blade, select the **Custom autoscale** option.
3. In the **Custom autoscale** section, specify the following settings (leave others with their default values):

Setting	Value
Scaling mode	**Scale based on a metric**
Instance limits Minimum	**1**
Instance limits Maximum	**3**
Instance limits Default	**1**

4. Select **+ Add a rule**.
5. On the **Scale rule** blade, specify the following settings and select **Add** (leave others with their default values):

Setting	Value
Time aggregation	**Maximum**
Metric namespace	**Virtual Machine Host**
Metric name	**Percentage CPU**
VMName Operator	**=**
Dimension values	**az30305c-vmss_0**
Enable metric divide by instance count	**Enabled**
Operator	**Greater than**
Metric threshold to trigger scale action	**1**
Duration (in minutes)	**1**
Time grain statistics	**Maximum**
Operation	**Increase count by**
Instance count	**1**
Cool down (minutes)	**5**

Note: These values are selected strictly for lab purposes to trigger scaling as soon as possible. For guidance regarding Azure VM Scale Set scaling, refer to Microsoft Docs .

6. Back on the **az30305c-vmss | Scaling** blade, select **+ Add a rule**.
7. On the **Scale rule** blade, specify the following settings and select **Add** (leave others with their default values):

Setting	Value
Time aggregation	**Average**
Metric namespace	**Virtual Machine Host**
Metric name	**Percentage CPU**
VMName Operator	**=**
Dimension values	**2 selected**
Enable metric divide by instance count	**Enabled**
Operator	**Less than**
Metric threshold to trigger scale action	**1**
Duration (in minutes)	**1**
Time grain statistics	**Minimum**
Operation	**Decrease count by**
Instance count	**1**
Cool down (minutes)	**5**

8. Back on the **az30305c-vmss | Scaling** blade, select **Save**.

6.5.4.2 Task 2: Test autoscaling of an Azure VM Scale Set

1. In the Azure portal, start a new **Bash** session in the Cloud Shell pane.
2. From the Cloud Shell pane, run the following to trigger autoscaling of the Azure VM Scale Set instances in the backend pool of the Azure Application Gateway (replace the `<lb_IP_address>` placeholder with the IP address of the front end of the gateway you identified earlier):

```
for (( ; ; )); do curl -s <lb_IP_address>?[1-10]; done
```

3. In the Azure portal, on the **az30305c-vmss** Overview blade, on the Monitoring tab, review the **CPU (average)** chart and verify that the CPU utilization of the Application Gateway increased sufficiently to trigger scaling out.

   Note: You might need to wait a few minutes.

4. On the **az30305c-vmss** blade, select the **Instances** entry and verify that the number of instances has increased.

   Note: You might need to refresh the **az30305c-vmss | Instances** blade.

   Note: You might see the number of instances increasing by 2 (rather than 1). This is expected as long as the final number of running instances is 3.

5. In the Azure portal, close the **Cloud Shell** pane.
6. In the Azure portal, on the **az30305c-vmss** blade, review the **CPU (average)** chart and verify that the CPU utilization of the Application Gateway decreased sufficiently to trigger scaling in.

   Note: You might need to wait a few minutes.

7. On the **az30305c-vmss** blade, select the **Instances** entry and verify that the number of instances has decreased to 2.

   Note: You might need to refresh the **az30305c-vmss | Instances** blade.

8. On the **az30305c-vmss** blade, select **Scaling**.
9. On the **az30305c-vmss | Scaling** blade, select the **Manual scale** option and select **Save**.

   Note: This will prevent any undesired autoscaling during the next exercise.

6.5.5 Exercise 5: Implementing vertical scaling of Azure VM Scale Sets

The main tasks for this exercise are as follows:

1. Scaling compute resources of Azure virtual machine scale set instances.
2. Scaling storage resources of Azure virtual machine scale sets instances.

6.5.5.1 Task 1: Scale compute resources of Azure virtual machine scale set instances.

1. In the Azure Portal, on the **az30305c-vmss** blade, select **Size**.
2. In the list of available sizes, select any available size other than currently configured and select **Resize**.
3. On the **az30305c-vmss** blade, select the **Instances** entry and, on the **az30305c-vmss | Instances** blade, observe the process of replacing existing instances with new ones of the desired size.

   Note: You might need to refresh the **az30305c-vmss | Instances** blade.
4. Wait until the instances are updated and running.

6.5.5.2 Task 2: Scale storage resources of Azure virtual machine scale sets instances.

1. On the **az30305c-vmss** blade, select **Disks**, select **+ Create and attach a new disk**, attach a new managed disk with the following settings (leave others with their default values), and select **Save**:

Setting	Value
LUN	**0**
Size	**32**
Storage account type	**Standard HDD**

2. On the **az30305c-vmss** blade, select the **Instances** entry and, on the **az30305c-vmss | Instances** blade, observe the process of updating the existing instances.

   Note: The disk attached in the previous step is a raw disk. Before it can be used, it is necessary to create a partition, format it, and mount it. To accomplish this, you will deploy a PowerShell script to Azure VM scale set instances via the Custom Script extension. First, however, you will need to remove it.
3. On the **az30305c-vmss** blade, select **Extensions**, on the **az30305c-vmss | Extensions** blade, select the **customScriptExtension** entry, and then, on the **Extensions** blade, select **Uninstall**.

   Note: Wait for uninstallation to complete.
4. In the Azure portal, navigate to the **az30305c-labRG** resource group blade, in the list of resources, select the storage account resource.
5. On the storage account blade, select **Containers** and then select **+ Container**.
6. On the **New container** blade, specify the following settings (leave others with their default values) and select **Create**:

Setting	Value
Name	**scripts**
Public access level	**Private (no anonymous access**)

7. Back on the storage account blade displaying the list of containers, select **scripts**.
8. On the **scripts** blade, select **Upload**.
9. On the **Upload blob** blade, select the folder icon, in the **Open** dialog box, navigate to the **\\AZ303\AllFiles\Labs\05** folder, select **az30305e-configure_VMSS_with_data_disk.ps1**, select **Open**, and back on the **Upload blob** blade, select **Upload**.
10. In the Azure portal, navigate back to the **az30305c-vmss** virtual machine scale set blade.
11. On the **az30305c-vmss** blade, select **Extensions**, on the **az30305c-vmss | Extensions** blade, select

+ **Add** and then, select the **customScriptExtension** entry on the **Extensions** blade.

12. On the **New resource** blade, select **Custom Script Extension** and then select **Create**.
13. From the **Install extension** blade, select **Browse**.
14. On the **Storage accounts** blade, select the name of the storage account into which you uploaded the **az30305e-configure_VMSS_with_data_disk.ps1** script, on the **Containers** blade, select **scripts**, on the **scripts** blade, select **az30305e-configure_VMSS_with_data_disk.ps1**, and then select **Select**.
15. Back on the **Install extension** blade, select **OK**.
16. On the **az30305c-vmss** blade, select the **Instances** entry and, on the **az30305c-vmss | Instances** blade, observe the process of updating existing instances.

    > **Note**: You might need to refresh the **az30305c-vmss | Instances** blade.

6.5.5.3 Task 3: Remove Azure resources deployed in the exercise

1. From the Cloud Shell pane, run the following to list the resource group you created in this exercise:

   ```
   az group list --query "[?starts_with(name,'az30305c-')]".name --output tsv
   ```

   > **Note**: Verify that the output contains only the resource group you created in this lab. This group will be deleted in this task.

2. From the Cloud Shell pane, run the following to delete the resource group you created in this lab

   ```
   az group list --query "[?starts_with(name,'az30305c-')]".name --output tsv | xargs -L1 bash -c 'az
   ```

3. Close the Cloud Shell pane.

6.6 lab: title: '6: Implementing and Configuring Azure Storage File and Blob Services' module: 'Module 6: Implement Storage Accounts'

7 Lab: Implementing and Configuring Azure Storage File and Blob Services

8 Student lab manual

8.1 Lab scenario

Adatum Corporation hosts large amounts of unstructured and semi-structured data in its on-premises storage. Its maintenance becomes increasingly complex and costly. Some of the data is preserved for extensive amount of time to address data retention requirements. The Adatum Enterprise Architecture team is looking for inexpensive alternatives that would support tiered storage, while, at the same time allow for secure access that minimizes the possibility of data exfiltration. While the team is aware of practically unlimited capacity offered by Azure Storage, it is concerned about the usage of account keys, which grant unlimited access to the entire content of the corresponding storage accounts. While keys can be rotated in an orderly manner, such operation needs to be carried out with proper planning. In addition, access keys constitute exclusively an authorization mechanism, which limits the ability to properly audit their usage.

To address these shortcomings, the Architecture team decided to explore the use of shared access signatures. A shared access signature (SAS) provides secure delegated access to resources in a storage account while minimizing the possibility of unintended data exposure. SAS offers granular control over data access, including the ability to limit access to an individual storage object, such as a blob, restricting such access to a custom time window, as well as filtering network access to a designated IP address range. In addition, the Architecture team wants to evaluate the level of integration between Azure Storage and Azure Active Directory, hoping to address its audit requirements. The Architecture team also decided to determine suitability of Azure Files as an alternative to some of its on-premises file shares.

To accomplish these objectives, Adatum Corporation will test a range of authentication and authorization mechanisms for Azure Storage resources, including:

- Using shared access signatures on the account, container, and object-level

- Configuring access level for blobs
- Implementing Azure Active Directory based authorization
- Using storage account access keys

8.2 Objectives

After completing this lab, you will be able to:

- Implement authorization of Azure Storage blobs by leveraging shared access signatures
- Implement authorization of Azure Storage blobs by leveraging Azure Active Directory
- Implement authorization of Azure Storage file shares by leveraging access keys

8.3 Lab Environment

Windows Server admin credentials

- User Name: **Student**
- Password: **Pa55w.rd1234**

Estimated Time: 90 minutes

8.4 Lab Files

- \\AZ303\AllFiles\Labs\06\azuredeploy30306suba.json
- \\AZ303\AllFiles\Labs\06\azuredeploy30306rga.json
- \\AZ303\AllFiles\Labs\06\azuredeploy30306rga.parameters.json

8.4.1 Exercise 0: Prepare the lab environment

The main tasks for this exercise are as follows:

1. Deploy an Azure VM by using an Azure Resource Manager template

8.4.1.1 Task 1: Deploy an Azure VM by using an Azure Resource Manager template

1. From your lab computer, start a web browser, navigate to the Azure portal, and sign in by providing credentials of a user account with the Owner role in the subscription you will be using in this lab.
2. In the Azure portal, open **Cloud Shell** pane by selecting on the toolbar icon directly to the right of the search textbox.
3. If prompted to select either **Bash** or **PowerShell**, select **PowerShell**.

   > **Note**: If this is the first time you are starting **Cloud Shell** and you are presented with the **You have no storage mounted** message, select the subscription you are using in this lab, and select **Create storage**.

4. In the toolbar of the Cloud Shell pane, select the **Upload/Download files** icon, in the drop-down menu select **Upload**, and upload the file **\\AZ303\AllFiles\Labs\06\azuredeploy30306suba.json** into the Cloud Shell home directory.
5. From the Cloud Shell pane, run the following to create a resource groups (replace the `<Azure region>` placeholder with the name of the Azure region that is available for deployment of Azure VMs in your subscription and which is closest to the location of your lab computer):

   ```
   $location = '<Azure region>'
   ```

   ```
   New-AzSubscriptionDeployment `
     -Location $location `
     -Name az30306subaDeployment `
     -TemplateFile $HOME/azuredeploy30306suba.json `
     -rgLocation $location `
     -rgName 'az30306a-labRG'
   ```

Note: To identify Azure regions where you can provision Azure VMs, refer to https://azure.microsoft.com/en-us/regions/offers/

6. From the Cloud Shell pane, upload the Azure Resource Manager template \\AZ303\AllFiles\Labs\06\azuredeploy
7. From the Cloud Shell pane, upload the Azure Resource Manager parameter file \\AZ303\AllFilesLabs\06\azuredep
8. From the Cloud Shell pane, run the following to deploy a Azure VM running Windows Server 2019 that you will be using in this lab:

```
New-AzResourceGroupDeployment `
  -Name az30306rgaDeployment `
  -ResourceGroupName 'az30306a-labRG' `
  -TemplateFile $HOME/azuredeploy30306rga.json `
  -TemplateParameterFile $HOME/azuredeploy30306rga.parameters.json `
  -AsJob
```

Note: Do not wait for the deployment to complete but instead proceed to the next exercise. The deployment should take less than 5 minutes.

9. In the Azure portal, close the **Cloud Shell** pane.

8.4.2 Exercise 1: Configure Azure Storage account authorization by using shared access signature.

The main tasks for this exercise are as follows:

1. Create an Azure Storage account
2. Install Storage Explorer
3. Generate an account-level shared access signature
4. Create a blob container by using Azure Storage Explorer
5. Upload a file to a blob container by using AzCopy
6. Access a blob by using a blob-level shared access signature

8.4.2.1 Task 1: Create an Azure Storage account

1. In the Azure portal, search for and select **Storage accounts** and, on the **Storage accounts** blade, select **+ Create**.
2. On the **Basics** tab of the **Create a storage account** blade, specify the following settings (leave others with their default values) and select **Next: Advanced >**.

Setting	Value
Subscription	the name of the Azure subscription you are using in this lab
Resource group	the name of the new resource group **az30306a-labRG**
Storage account name	any globally unique name between 3 and 24 in length consisting of letters and digits
Location	the name of an Azure region where you can create an Azure Storage account
Performance	**Standard: Recommended for most scenarios (general-purpose v2 account)**
Redundancy	**Locally redundant storage (LRS)**

3. On the **Advanced** tab of the **Create a storage account** blade, review the available options, accept the defaults and Select **Next: Networking >**.
4. On the **Networking** tab of the **Create a storage account** blade, review the available options, accept the default option **Public endpoint (all networks)** and select **Next: Data protection >**.
5. On the **Data protection** tab of the **Create storage account** blade, review the available options, accept the defaults, and select **Next: Tags >**.
6. Select **Review + Create**, wait for the validation process to complete and select **Create**.

Note: Wait for the Storage account to be created. This should take about 2 minutes.

8.4.2.2 Task 2: Install Storage Explorer

Note: Ensure that the deployment of the Azure VM you initiated at the beginning of this lab has completed before you proceed.

1. In the Azure portal, search for and select **Virtual machines**, and, on the **Virtual machines** blade, in the list of virtual machines, select **az30306a-vm0**.
2. On the **az30306a-vm0** blade, select **Connect**, in the drop-down menu, select **RDP**, and then select **Download RDP File**, then select **open file** and select **Connect**.
3. When prompted, sign in with the following credentials:

Setting	Value
User Name	**Student**
Password	**Pa55w.rd1234**

4. Within the Remote Desktop session to **az30306a-vm0**, in the Server Manager window, select **Local Server**, select the **On** link next to the **IE Enhanced Security Configuration** label, and, in the **IE Enhanced Security Configuration** dialog box, select both **Off** options.
5. Within the Remote Desktop session to **az30306a-vm0**, start Internet Explorer, navigate to the download page of Microsoft Edge, download Microsoft Edge installer and perform the installation.
6. Within the Remote Desktop session to **az30306a-vm0**, in Microsoft Edge, navigate to the download page of Azure Storage Explorer
7. Within the Remote Desktop session to **az30306a-vm0**, download and install Azure Storage Explorer with the default settings.

8.4.2.3 Task 3: Generate an account-level shared access signature

1. Within the Remote Desktop session to **az30306a-vm0**, start Microsoft Edge, navigate to the Azure portal, and sign-in by providing credentials of the user account with the Owner role in the subscription you are using in this lab.
2. Navigate to the blade of the newly created storage account, select **Access keys** and review the settings of the target blade.

   Note: Each storage account has two keys which you can independently regenerate. Knowledge of the storage account name and either of the two keys provides full access to the entire storage account.

3. On the storage account blade, select **Shared access signature** and review the settings of the target blade.
4. On the resulting blade, specify the following settings (leave others with their default values):

Setting	Value
Allowed services	**Blob**
Allowed resource types	**Service** and **Container**
Allowed permissions	**Read**, **List** and **Create**
Blob versioning permissions	disabled
Start	24 hours before the current time in your current time zone
End	24 hours after the current time in your current time zone
Allowed protocols	**HTTPS only**
Signing key	**key1**

5. Select **Generate SAS and connection string**.
6. Copy the value of **Blob service SAS URL** into Clipboard.

8.4.2.4 Task 4: Create a blob container by using Azure Storage Explorer

1. Within the Remote Desktop session to **az30306a-vm0**, start Azure Storage Explorer.
2. In the Azure Storage Explorer window, on the **Select Resource** tab of the **Connect to Azure Storage** window, select **Storage account or service**.
3. In the Azure Storage Explorer window, on the **Select Connection Method** tab of the **Connect to Azure Storage** window, select **Shared access signature URL (SAS)** and select **Next**.
4. In the Azure Storage Explorer window, on the **Enter Connection Info** tab of the **Connect to Azure Storage** window, in the **Display name** text box, type **az30306a-blobs**, in the **Service URL** text box, paste the value you copied into Clipboard, and select **Next**.

   Note: If Ctrl-V paste doesn't seem to work within the RDP session, try copying the Service URL into a Notepad on the SEA-Dev VM and then copying the value back into the RDP session.

5. In the Azure Storage Explorer window, on the **Summary** tab of the **Connect to Azure Storage** window, select **Connect**.
6. In the Azure Storage Explorer window, in the **EXPLORER** pane, navigate to the **az30306a-blobs** entry, expand it and note that you have access to **Blob Container** endpoint only.
7. Right select the **Blob Containers** entry (nested in the **az30306a-blobs** entry), in the right-click menu, select **Create Blob Container**, and use the empty text box to set the container name to **container1**.
8. Select **container1** to open a new tab in the main window pane of the Storage Explorer window, on the **container1** tab, select **Upload**, and in the drop-down list, select **Upload Files**.
9. In the **Upload Files** window, select the ellipsis button next to the **Selected files** label, in the **Choose files to upload** window, select **C:\Windows\system.ini**, and select **Open**.
10. Back in the **Upload Files** window, select **Upload** and note the error message displayed in the **Activities** list.

    Note: This is expected, since the shared access signature does not provide object-level permissions.

11. Leave the Azure Storage Explorer window open.

8.4.2.5 Task 5: Upload a file to a blob container by using AzCopy

1. Within the Remote Desktop session to **az30306a-vm0**, in the browser window, on the **Shared access signature** blade, specify the following settings (leave others with their default values):

Setting	Value
Allowed services	**Blob**
Allowed resource types	**Object**
Allowed permissions	**Read**, **Create**
Blob versioning permissions	disabled
Start	24 hours before the current time in your current time zone
End	24 hours after the current time in your current time zone
Allowed protocols	**HTTPS only**
Signing key	**key1**

2. Select **Generate SAS and connection string**.
3. Copy the value of **SAS token** into Clipboard.
4. In the Azure portal, open **Cloud Shell** pane by selecting on the toolbar icon directly to the right of the search textbox.
5. If prompted to select either **Bash** or **PowerShell**, select **PowerShell**.
6. From the Cloud Shell pane, run the following to create a file and add a line of text into it:

```
New-Item -Path './az30306ablob.html'

Set-Content './az30306ablob.html' '<h3>Hello from az30306ablob via SAS</h3>'
```

7. From the Cloud Shell pane, run the following to upload the newly created file as a blob into container1 of the Azure Storage account you created earlier in this exercise (replace the `<sas_token>` placeholder with the value of the shared access signature you copied to Clipboard earlier in this task):

```
$storageAccountName = (Get-AzStorageAccount -ResourceGroupName 'az30306a-labRG')[0].StorageAccount

azcopy cp './az30306ablob.html' "https://$storageAccountName.blob.core.windows.net/container1/az30
```

8. Review the output generated by azcopy and verify that the job completed successfully.
9. Close the Cloud Shell pane.
10. Within the Remote Desktop session to **az30306a-vm0**, in the browser window, on the storage account blade, in the **Data storage** section, select **Containers**.
11. In the list of containers, select **container1**.
12. On the **container1** blade, verify that **az30306ablob.html** appears in the list of blobs.

8.4.2.6 Task 6: Access a blob by using a blob-level shared access signature

1. Within the Remote Desktop session to **az30306a-vm0**, in the browser window, on the **container1** blade, select **Change access level**, verify that is set to **Private (no anonymous access)**, and select **Cancel**.

   > **Note**: If you want to allow anonymous access, you can set the public access level to **Blob (anonymous read access for blobs only)** or **Container (anonymous read access for containers and blobs)**.

2. On the **container1** blade, select **az30306ablob.html**.
3. On the **az30306ablob.html** blade, select **Generate SAS**, review the available options without modifying them, and then select **Generate SAS token and URL**.
4. Copy the value of the **Blob SAS URL** into Clipboard.
5. Open a new tab in the browser window and navigate to the URL you copied into Clipboard in the previous step.
6. Verify that the message **Hello from az30306ablob via SAS** appears in the browser window.

8.4.3 Exercise 2: Configure Azure Storage blob service authorization by using Azure Active Directory

The main tasks for this exercise are as follows:

1. Create an Azure AD user
2. Enable Azure Active Directory authorization for Azure Storage blob service
3. Upload a file to a blob container by using AzCopy

8.4.3.1 Task 1: Create an Azure AD user

1. Within the Remote Desktop session to **az30306a-vm0**, in the browser window, open **PowerShell** session within a **Cloud Shell** pane.
2. From the Cloud Shell pane, run the following to explicitly authenticate to your Azure AD tenant:

```
Connect-AzureAD
```

3. From the Cloud Shell pane, run the following to identify the Azure AD DNS domain name:

```
$domainName = ((Get-AzureAdTenantDetail).VerifiedDomains)[0].Name
```

4. From the Cloud Shell pane, run the following to create a new Azure AD user:

```
$passwordProfile = New-Object -TypeName Microsoft.Open.AzureAD.Model.PasswordProfile
$passwordProfile.Password = 'Pa55w.rd1234'
$passwordProfile.ForceChangePasswordNextLogin = $false
New-AzureADUser -AccountEnabled $true -DisplayName 'az30306auser1' -PasswordProfile $passwordProfi
```

5. From the Cloud Shell pane, run the following to identify the user principal name of the newly created Azure AD user:

```
(Get-AzureADUser -Filter "MailNickName eq 'az30306auser1'").UserPrincipalName
```

6. Note the user principal name. You will need it later in this exercise.
7. Close the Cloud Shell pane.

8.4.3.2 Task 2: Enable Azure Active Directory authorization for Azure Storage blob service

1. Within the Remote Desktop session to **az30306a-vm0**, in the browser window displaying the Azure portal, navigate back to the **container1** blade.
2. On the **container1** blade, select **Switch to Azure AD User Account**.
3. Note the error message indicating that you no longer have permissions to list data in the blob container. This is expected.

   > **Note**: Despite having the **Owner** role in the subscription, you also need to be assigned either built-in or a custom role that provides access to the blob content of the storage account, such as **Storage Blob Data Owner**, **Storage Blob Data Contributor**, or **Storage Blob Data Reader**.

4. In the Azure portal, navigate back to the blade of the storage account hosting **container1**, select **Access control (IAM)**, select **+ Add**, and, in the drop-down list, select **Add role assignment**.

   > **Note**: Write down the name of the storage account. You will need it in the next task.

5. On the **Add role assignment** blade, in the **Role** drop-down list, select **Storage Blob Data Owner**, ensure that the **Assign access to** drop-down list entry is set to **User, group, or service principal**, select both your user account and the user account you created in the previous task from the list displayed below the **Select** text box, and select **Save**.
6. Navigate back to the **container1** blade and verify that you can see the content of the container.

8.4.3.3 Task 3: Upload a file to a blob container by using AzCopy

1. Within the Remote Desktop session to **az30306a-vm0**, start Windows PowerShell.
2. From the Windows PowerShell prompt, run the following to download the **azcopy.zip** archive, extract its content, and switch to the location containing **azcopy.exe**:

   ```
   $url = 'https://aka.ms/downloadazcopy-v10-windows'
   $zipFile = '.\azcopy.zip'

   Invoke-WebRequest -Uri $Url -OutFile $zipFile

   Expand-Archive -Path $zipFile -DestinationPath '.\'

   Set-Location -Path 'azcopy*'
   ```

3. From the Windows PowerShell prompt, run the following to authenticate AzCopy by using the Azure AD user account you created in the first task of this exercise.

   ```
   .\azcopy.exe login
   ```

   > **Note**: You cannot use for this purpose a Microsoft account, which is the reason that Azure AD user account had to be created first.

4. Follow instructions provided in the message generated by the command you run in the previous step to authenticate as the **az30306auser1** user account. When prompted for credentials, provide the user principal name of the account you noted in the first task of this exercise and its password **Pa55w.rd1234**.
5. Once you successfully authenticated, from the Windows PowerShell prompt, run the following to create a file you will upload to **container1**:

   ```
   New-Item -Path './az30306bblob.html'

   Set-Content './az30306bblob.html' '<h3>Hello from az30306bblob via Azure AD</h3>'
   ```

6. From the the Windows PowerShell prompt, run the following to upload the newly created file as a blob into **container1** of the Azure Storage account you created in the previous exercise (replace the

<storage_account_name> placeholder with the value of the storage account you noted in the previous task):

```
.\azcopy cp './az30306bblob.html' 'https://<storage_account_name>.blob.core.windows.net/container1
```

7. Review the output generated by azcopy and verify that the job completed successfully.
8. From the Windows PowerShell prompt and run the following to verify that you do not have access to the uploaded blob outside of the security context provided by the AzCopy utility (replace the <storage_account_name> placeholder with the value of the storage account you noted in the previous task):

```
Invoke-WebRequest -Uri 'https://<storage_account_name>.blob.core.windows.net/container1/az30306bbl
```

9. Within the Remote Desktop session to **az30306a-vm0**, in the browser window, navigate back to **container1**.
10. On the **container1** blade, verify that **az30306bblob.html** appears in the list of blobs.
11. On the **container1** blade, select **Change access level**, set the public access level to **Blob (anonymous read access for blobs only)** and select **OK**.
12. Switch back to the Windows PowerShell prompt and re-run the following command to verify that now you can access the uploaded blob anonymously (replace the <storage_account_name> placeholder with the value of the storage account you noted in the previous task):

```
Invoke-WebRequest -Uri 'https://<storage_account_name>.blob.core.windows.net/container1/az30306bbl
```

8.4.4 Exercise 3: Implement Azure Files.

The main tasks for this exercise are as follows:

1. Create an Azure Storage file share
2. Map a drive to an Azure Storage file share from Windows
3. Remove Azure resources deployed in the lab

8.4.4.1 Task 1: Create an Azure Storage file share

1. Within the Remote Desktop session to **az30306a-vm0**, in the browser window displaying the Azure portal, navigate back to the blade of the storage account you created in the first exercise of this lab and, in the **Data storage** section, select **File shares**.
2. Select **+ File share** and create a file share with the following settings:

Setting	Value
Name	**az30306a-share**
Quota	**1024**

8.4.4.2 Task 2: Map a drive to an Azure Storage file share from Windows

1. Select the newly created file share and select **Connect**.
2. On the **Connect** blade, ensure that the **Windows** tab is selected, and select **Copy to clipboard**.

   Note: Azure Storage file share mapping uses the storage account name and one of two storage account keys as the equivalents of user name and password, respectively in order to gain access to the target share.

3. Within the Remote Desktop session to **az30306a-vm0**, open a PowerShell session and at the PowerShell prompt, paste and execute the script you copied.
4. Verify that the script completed successfully.
5. Start File Explorer, navigate to **Z:** drive and verify that the mapping was successful.
6. In File Explorer, create a folder named **Folder1** and a text file inside the folder named **File1.txt**.

7. Switch back to the browser window displaying the Azure portal, on the **az30306a-share** blade, select **Refresh**, and verify that **Folder1** appears in the list of folders.
8. Select **Folder1** and verify that **File1.txt** appears in the list of files.

8.4.4.3 Task 3: Remove Azure resources deployed in the lab

1. Within the Remote Desktop session to **az30306a-vm0**, in the browser window displaying the Azure portal, start a PowerShell session within the Cloud Shell pane.
2. From the Cloud Shell pane, run the following to list the resource group you created in this exercise:

   ```
   Get-AzResourceGroup -Name 'az30306*'
   ```

   > **Note**: Verify that the output contains only the resource group you created in this lab. This group will be deleted in this task.

3. From the Cloud Shell pane, run the following to delete the resource group you created in this lab

   ```
   Get-AzResourceGroup -Name 'az30306*' | Remove-AzResourceGroup -Force -AsJob
   ```

4. Close the Cloud Shell pane.
5. In the Azure portal, navigate to the **Users** blade of the Azure Active Directory tenant associated with your Azure subscription.
6. In the list of user accounts, select the entry representing the **az30306auser1** user account, select the ellipsis icon in the toolbar, select **Delete user** and select **Yes** when prompted to confirm.

8.5 lab: title: '10: Managing Azure Role-Based Access Control' module: 'Module 10: Implement and Manage Azure Governance'

9 Lab: Managing Azure Role-Based Access Control

10 Student lab manual

10.1 Lab scenario

With Azure Active Directory (Azure AD) becoming integral part of its identity management environment, the Adatum Enterprise Architecture team must also determine the optimal authorization approach. In the context of controlling access to Azure resources, such approach must involve the use of Azure Role-Based Access Control (RBAC). Azure RBAC is an authorization system built on Azure Resource Manager that provides fine-grained access management of Azure resources.

The key concept of Azure RBAC is role assignment. A role assignment consists of three elements: security principal, role definition, and scope. A security principal is an object that represents a user, group, service principal, or managed identity that is requesting access to Azure resources. A role definition is a collection of the operations that the role assignments will grant, such as read, write, or delete. Roles can be generic or resource specific. Azure includes four built-in generic roles (Owner, Contributor, Reader, and User Access Administrator) and a fairly large number of built-in resource-specific roles (such as, for example, Virtual Machine Contributor, which includes permissions to create and manage Azure virtual machines). It is also possible to define custom roles. A scope is the set of resources that the access applies to. A scope can be set at multiple levels: management group, subscription, resource group, or resource. Scopes are structured in a parent-child relationship.

The Adatum Enterprise Architecture team wants to test delegation of Azure management by using custom Role-Based Access Control roles. To start its evaluation, the team intends to create a custom role that provides restricted access to Azure virtual machines.

10.2 Objectives

After completing this lab, you will be able to:

- Define a custom RBAC role
- Assign a custom RBAC role

10.3 Lab Environment

Windows Server admin credentials

- User Name: **Student**
- Password: **Pa55w.rd1234**

Estimated Time: 60 minutes

10.4 Lab Files

- \\AZ303\AllFiles\Labs\10\azuredeploy30310suba.json
- \\AZ303\AllFiles\Labs\10\azuredeploy30310rga.json
- \\AZ303\AllFiles\Labs\10\azuredeploy30310rga.parameters.json
- \\AZ303\AllFiles\Labs\10\roledefinition30310.json

10.5 Instructions

10.5.1 Exercise 0: Prepare the lab environment

The main tasks for this exercise are as follows:

1. Deploy an Azure VM by using an Azure Resource Manager template
2. Create an Azure Active Directory user

10.5.1.1 Task 1: Deploy an Azure VM by using an Azure Resource Manager template

1. From your lab computer, start a web browser, navigate to the Azure portal, and sign in by providing credentials of a user account with the Owner role in the subscription you will be using in this lab.
2. In the Azure portal, open **Cloud Shell** pane by selecting on the toolbar icon directly to the right of the search textbox.
3. If prompted to select either **Bash** or **PowerShell**, select **PowerShell**.

   > **Note**: If this is the first time you are starting **Cloud Shell** and you are presented with the **You have no storage mounted** message, select the subscription you are using in this lab, and select **Create storage**.

4. In the toolbar of the Cloud Shell pane, select the **Upload/Download files** icon, in the drop-down menu select **Upload**, and upload the file **\\AZ303\AllFiles\Labs\10\azuredeploy30310suba.json** into the Cloud Shell home directory.
5. From the Cloud Shell pane, run the following to create a resource groups (replace the `<Azure region>` placeholder with the name of the Azure region that is available for deployment of Azure VMs in your subscription and which is closest to the location of your lab computer):

   ```
   $location = '<Azure region>'
   New-AzSubscriptionDeployment `
     -Location $location `
     -Name az30310subaDeployment `
     -TemplateFile $HOME/azuredeploy30310suba.json `
     -rgLocation $location `
     -rgName 'az30310a-labRG'
   ```

   > **Note**: To identify Azure regions where you can provision Azure VMs, refer to **https://azure.microsoft.com/en-us/regions/offers/**

6. From the Cloud Shell pane, upload the Azure Resource Manager template **\\AZ303\AllFiles\Labs\10\azuredeploy**
7. From the Cloud Shell pane, upload the Azure Resource Manager parameter file **\\AZ303\AllFilesLabs\10\azuredep**
8. From the Cloud Shell pane, run the following to deploy a Azure VM running Windows Server 2019 that you will be using in this lab:

```
New-AzResourceGroupDeployment `
  -Name az30310rgaDeployment `
  -ResourceGroupName 'az30310a-labRG' `
  -TemplateFile $HOME/azuredeploy30310rga.json `
  -TemplateParameterFile $HOME/azuredeploy30310rga.parameters.json `
  -AsJob
```

> **Note**: Do not wait for the deployment to complete but instead proceed to the next task. The deployment should take less than 5 minutes.

10.5.1.2 Task 2: Create an Azure Active Directory user

1. In the Azure portal, from the PowerShell session in the Cloud Shell pane, run the following to authenticate to the Azure AD tenant associated with your Azure subscription:

   ```
   Connect-AzureAD
   ```

2. From the Cloud Shell pane, run the following to identify the Azure AD DNS domain name:

   ```
   $domainName = ((Get-AzureAdTenantDetail).VerifiedDomains)[0].Name
   ```

3. From the Cloud Shell pane, run the following to create a new Azure AD user:

   ```
   $passwordProfile = New-Object -TypeName Microsoft.Open.AzureAD.Model.PasswordProfile
   $passwordProfile.Password = 'Pa55w.rd1234'
   $passwordProfile.ForceChangePasswordNextLogin = $false
   New-AzureADUser -AccountEnabled $true -DisplayName 'az30310aaduser1' -PasswordProfile $passwordPro
   ```

4. From the Cloud Shell pane, run the following to identify the user principal name of the newly created Azure AD user:

   ```
   (Get-AzureADUser -Filter "MailNickName eq 'az30310aaduser1'").UserPrincipalName
   ```

   > **Note**: Record the user principal name of the newly created Azure AD user. You will need it later in this lab.

5. Close the Cloud Shell pane.

10.5.2 Exercise 1: Define a custom RBAC role

The main tasks for this exercise are as follows:

1. Identify actions to delegate via RBAC
2. Create a custom RBAC role in an Azure AD tenant

10.5.2.1 Task 1: Identify actions to delegate via RBAC

1. In the Azure portal, navigate to the **az30310a-labRG** blade.
2. On the **az30310a-labRG** blade, select **Access Control (IAM)**.
3. On the **az30310a-labRG - Access Control (IAM)** blade, select **Roles (Classic)**.
4. On the **Roles** blade, select **Owner**.
5. On the **Owner** blade, select **Permissions**.
6. On the **Permissions (preview)** blade, select **Microsoft Compute**.
7. On the **Microsoft Compute** blade, select **Virtual machines**.
8. On the **Virtual Machines** blade, review the list of management actions that can be delegated through RBAC. Note that they include the **Deallocate Virtual Machine** and **Start Virtual Machine** actions.

10.5.2.2 Task 2: Create a custom RBAC role in an Azure AD tenant

1. On the lab computer, open the file **\\AZ303\AllFiles\Labs\10\roledefinition30310.json** and review its content:

```
{
   "Name": "Virtual Machine Operator (Custom)",
   "Id": null,
   "IsCustom": true,
   "Description": "Allows to start/restart Azure VMs",
   "Actions": [
       "Microsoft.Compute/*/read",
       "Microsoft.Compute/virtualMachines/restart/action",
       "Microsoft.Compute/virtualMachines/start/action"
   ],
   "NotActions": [
   ],
   "AssignableScopes": [
       "/subscriptions/SUBSCRIPTION_ID"
   ]
}
```

2. On the lab computer, in the browser window displaying the Azure portal, start a **PowerShell** session within the **Cloud Shell**.
3. From the Cloud Shell pane, upload the Azure Resource Manager template \\AZ303\AllFiles\Labs\10\roledefinitic into the home directory.
4. From the Cloud Shell pane, run the following to replace the `SUBSCRIPTION_ID` placeholder with the ID value of the Azure subscription:

```
$subscription_id = (Get-AzContext).Subscription.id
(Get-Content -Path $HOME/roledefinition30310.json) -Replace 'SUBSCRIPTION_ID', "$subscription_id"
```

5. From the Cloud Shell pane, run the following to verify that the `SUBSCRIPTION_ID` placeholder was replaced with the ID value of the Azure subscription:

```
Get-Content -Path $HOME/roledefinition30310.json
```

6. From the Cloud Shell pane, run the following to create the custom role definition:

```
New-AzRoleDefinition -InputFile $HOME/roledefinition30310.json
```

7. From the Cloud Shell pane, run the following to verify that the role was created successfully:

```
Get-AzRoleDefinition -Name 'Virtual Machine Operator (Custom)'
```

8. Close the Cloud Shell pane.

10.5.3 Exercise 2: Assign and test a custom RBAC role

The main tasks for this exercise are as follows:

1. Create an RBAC role assignment
2. Test the RBAC role assignment

10.5.3.1 Task 1: Create an RBAC role assignment

1. In the Azure portal, navigate to the **az30310a-labRG** blade.
2. On the **az30310a-labRG** blade, select **Access Control (IAM)**.
3. On the **az30310a-labRG - Access Control (IAM)** blade, select **+ Add** and select the **Add role assignment** option.
4. On the **Add role assignment** blade, specify the following settings (leave others with their existing values) and select **Save**:

Setting	Value
Role	**Virtual Machine Operator (Custom)**
Assign access to	**User, group, or service principal**
Select	**az30310aaduser1**

10.5.3.2 Task 2: Test the RBAC role assignment

1. From the lab computer, start a new in-private web browser session, navigate to the Azure portal, and sign in by using the **az30310aaduser1** user account with the **Pa55w.rd1234** password.

   > **Note**: Make sure to use the user principal name of the **az30310aaduser1** user account, which you recorded earlier in this lab.

   > **Note**: If you want to skip the Microsoft Security Default for the account during the login process, use the link *"Skip for now (14 days until this is required)"* option.

2. In the Azure portal, navigate to the **Resource groups** blade. Note that you are not able to see any resource groups.
3. In the Azure portal, navigate to the **All resources** blade. Note that you are able to see only the **az30310a-vm0** and its managed disk.
4. In the Azure portal, navigate to the **az30310a-vm0** blade. Try stopping the virtual machine. Review the error message in the notification area and note that this action failed because the current user is not authorized to carry it out.
5. Restart the virtual machine and verify that the action completed successfully.
6. Close the in-private web browser session.

10.5.3.3 Task 3: Remove Azure resources deployed in the lab

1. From the lab computer, in the existing browser window displaying the Azure portal, start a PowerShell session within the Cloud Shell pane.
2. From the Cloud Shell pane, run the following to list the resource group you created in this exercise:

   ```
   Get-AzResourceGroup -Name 'az30310*'
   ```

   > **Note**: Verify that the output contains only the resource group you created in this lab. This group will be deleted in this task.

3. From the Cloud Shell pane, run the following to delete the resource group you created in this lab:

   ```
   Get-AzResourceGroup -Name 'az30310*' | Remove-AzResourceGroup -Force -AsJob
   ```

4. From the Cloud Shell pane, run the following to delete the lab files you uploaded earlier in this lab:

   ```
   Get-ChildItem -Path . -Filter 'azuredeploy30310*.json' | Remove-Item -Force
   Get-ChildItem -Path . -Filter 'roledefinition30310.json' | Remove-Item -Force
   ```

5. Close the Cloud Shell pane.
6. In the Azure portal, navigate to the **Users** blade of the Azure Active Directory tenant associated with your Azure subscription.
7. In the list of user accounts, select the entry representing the **az30310aaduser1** user account, select the ellipsis icon in the toolbar, select **Delete user** and select **Yes** when prompted to confirm.
8. In the Azure portal, navigate to the blade displaying properties of your Azure subscriptions, select the **Access control (IAM)** entry, and then select **Roles**.
9. In the list of roles, select the **Virtual Machine Operator (Custom)** entry, select **Remove** and, when prompted to confirm, select **Yes**.

10.6 lab: title: ’12: Protecting Hyper-V VMs by using Azure Site Recovery’ module: ’Module 12: Manage Workloads in Azure’

11 Lab: Protecting Hyper-V VMs by using Azure Site Recovery

12 Student lab manual

12.1 Lab scenario

While Adatum Corporation has, over the years, implemented a number of high availability provisions for their on-premises workloads, its disaster recovery capabilities are still insufficient to address the Recovery Point Objectives (RPOs) and Recovery Time Objectives (RTOs) demanded by its business. Maintaining the existing secondary on-premises site requires an extensive effort and incurs significant costs. The failover and failback procedures are, for the most part, manual and are poorly documented.

To address these shortcomings, the Adatum Enterprise Architecture team decided to explore capabilities of Azure Site Recovery, with Azure taking on the role of the hoster of the secondary site. Azure Site Recovery automatically and continuously replicates workloads running on physical and virtual machines from the primary to the secondary site. Site Recovery uses storage-based replication mechanism, without intercepting application data. With Azure as the secondary site, data is stored in Azure Storage, with built-in resilience and low cost. The target Azure VMs are hydrated following a failover by using the replicated data. The Recovery Time Objectives (RTO) and Recovery Point objectives are minimized since Site Recovery provides continuous replication for VMware VMs and replication frequency as low as 30 seconds for Hyper-V VMs. In addition, Azure Site Recovery also handles orchestration of failover and failback processes, which, to large extent, can be automated. It is also possible to use Azure Site Recovery for migrations to Azure, although the recommended approach relies on Azure Migrate instead.

The Adatum Enterprise Architecture team wants to evaluate the use of Azure Site Recovery for protecting on-premises Hyper-V virtual machines to Azure VM.

12.2 Objectives

After completing this lab, you will be able to:

- Configure Azure Site Recovery
- Perform test failover
- Perform planned failover
- Perform unplanned failover

12.3 Lab Environment

Windows Server admin credentials

- User Name: **Student**
- Password: **Pa55w.rd1234**

Estimated Time: 120 minutes

12.4 Lab Files

- \\AZ303\AllFiles\Labs\12\azuredeploy30312suba.json

12.4.1 Exercise 0: Prepare the lab environment

The main tasks for this exercise are as follows:

1. Deploy an Azure VM by using an Azure Resource Manager QuickStart template
2. Configure nested virtualization in the Azure VM

12.4.1.1 Task 1: Deploy an Azure VM by using an Azure Resource Manager QuickStart template

1. From your lab computer, start a web browser, navigate to the Azure portal, and sign in by providing credentials of a user account with the Owner role in the subscription you will be using in this lab.
2. In the Azure portal, open **Cloud Shell** pane by selecting on the toolbar icon directly to the right of the search textbox.
3. If prompted to select either **Bash** or **PowerShell**, select **PowerShell**.

   Note: If this is the first time you are starting **Cloud Shell** and you are presented with the **You have no storage mounted** message, select the subscription you are using in this lab, and select **Create storage**.

4. In the toolbar of the Cloud Shell pane, select the **Upload/Download files** icon, in the drop-down menu select **Upload**, and upload the file **\\AZ303\AllFiles\Labs\12\azuredeploy30312suba.json** into the Cloud Shell home directory.
5. From the Cloud Shell pane, run the following to create a resource groups (replace the `<Azure region>` placeholder with the name of the Azure region that is available for deployment of Azure VMs in your subscription and which is closest to the location of your lab computer):

```
$location = '<Azure region>'
New-AzSubscriptionDeployment `
  -Location $location `
  -Name az30312subaDeployment `
  -TemplateFile $HOME/azuredeploy30312suba.json `
  -rgLocation $location `
  -rgName 'az30312a-labRG'
```

   Note: To identify Azure regions where you can provision Azure VMs, refer to **https://azure.microsoft.com/en-us/regions/offers/**

6. In the Azure portal, close the **Cloud Shell** pane.
7. From your lab computer, open another browser tab, navigate to the nested-vms-in-virtual-network Azure QuickStart template and select **Deploy to Azure**. This will automatically redirect the browser to the **Hyper-V Host Virtual Machine with nested VMs** blade in the Azure portal.
8. On the **Hyper-V Host Virtual Machine with nested VMs** blade in the Azure portal, specify the following settings (leave others with their default values):

Setting	Value
Subscription	the name of the Azure subscription you are using in this lab
Resource group	**az30312a-labRG**
Host Public IP Address Name	**az30312a-hv-vm-pip**
Virtual Network Name	**az30312a-hv-vnet**
Host Network Interface1Name	**az30312a-hv-vm-nic1**
Host Network Interface2Name	**az30312a-hv-vm-nic2**
Host Virtual Machine Name	**az30312a-hv-vm**
Host Admin Username	**Student**
Host Admin Password	**Pa55w.rd1234**

9. On the **Hyper-V Host Virtual Machine with nested VMs** blade, select **Review + create** and then select **Create**.

   Note: Wait for the deployment to complete. The deployment might take about 10 minutes.

12.4.1.2 Task 2: Configure nested virtualization in the Azure VM

1. In the Azure portal, search for and select **Virtual machines** and, on the **Virtual machines** blade, select **az30312a-hv-vm**.
2. On the **az30312a-hv-vm** blade, select **Networking**.

3. On the **az30312a-hv-vm | Networking** blade, select **az30312a-hv-vm-nic1** and then select **Add inbound port rule**.

   Note: Make sure that you modify the settings of **az30312a-hv-vm-nic1**, which has the public IP address assigned to it.

4. On the **Add inbound security rule** blade, specify the following settings (leave others with their default values) and select **Add**:

Setting	Value
Destination port range	**3389**
Protocol	**Any**
Name	**AllowRDPInBound**

5. On the **az30312a-hv-vm** blade, select **Overview**.
6. On the **az30312a-hv-vm** blade, select **Connect**, in the drop-down menu, select **RDP**, and then click **Download RDP File**, after the RDP file downloads select **Open file** and select **Connect**.
7. When prompted, sign in with the following credentials:

- User Name: **Student**
- Password: **Pa55w.rd1234**

1. Within the Remote Desktop session to **az30312a-hv-vm**, in the Server Manager window, click **Local Server**, click the **On** link next to the **IE Enhanced Security Configuration** label, and, in the **IE Enhanced Security Configuration** dialog box, select both **Off** options.
2. Within the Remote Desktop session to **az30312a-hv-vm**, start Internet Explorer, navigate to the download page of Microsoft Edge, download Microsoft Edge installer and perform the installation.
3. Within the Remote Desktop session to **az30312a-hv-vm**, use Microsoft Edge to browse to Windows Server Evaluations, and download the Windows Server 2019 **VHD** file to the **F:\VHDs** folder (you will need to create it first).
4. Within the Remote Desktop session to **az30312a-hv-vm**, start **Hyper-V Manager**.
5. In the **Hyper-V Manager** console, select the **az30312a-hv-vm** node, select **New** and, in the cascading menu, select **Virtual Machine**. This will start the **New Virtual Machine Wizard**.
6. On the **Before You Begin** page of the **New Virtual Machine Wizard**, select **Next >**.
7. On the **Specify Name and Location** page of the **New Virtual Machine Wizard**, specify the following settings and select **Next >**:

Setting	Value
Name	**az30312a-vm1**
Store the virtual machine in a different location	selected
Location	**F:\VMs**

   Note: Make sure to create the **F:\VMs** folder.

8. On the **Specify Generation** page of the **New Virtual Machine Wizard**, ensure that the **Generation 1** option is selected and select **Next >**:
9. On the **Assign Memory** page of the **New Virtual Machine Wizard**, set **Startup memory** to **2048** and select **Next >**.
10. On the **Configure Networking** page of the **New Virtual Machine Wizard**, in the **Connection** drop-down list select **NestedSwitch** and select **Next >**.
11. On the **Connect Virtual Hard Disk** page of the **New Virtual Machine Wizard**, select the option **Use an existing virtual hard disk**, set location to the VHD file you downloaded to the **F:\VHDs** folder, and select **Next >**.
12. On the **Summary** page of the **New Virtual Machine Wizard**, select **Finish**.

13. In the **Hyper-V Manager** console, select the newly created virtual machine and select **Start**.
14. In the **Hyper-V Manager** console, verify that the virtual machine is running and select **Connect**.
15. In the Virtual Machine Connection window to **az30312a-vm1**, on the **Hi there** page, select **Next**.
16. In the Virtual Machine Connection window to **az30312a-vm1**, on the **License terms** page, select **Accept**.
17. In the Virtual Machine Connection window to **az30312a-vm1**, on the **Customize settings** page, set the password of the built-in Administrator account to **Pa55w.rd1234** and select **Finish**.
18. In the Virtual Machine Connection window to **az30312a-vm1**, sign in by using the newly set password.
19. In the Virtual Machine Connection window to **az30312a-vm1**, start Windows PowerShell and, in the **Administrator: Windows PowerShell** window run the following to set the computer name.

    ```
    Rename-Computer -NewName 'az30312a-vm1' -Restart
    ```

12.4.2 Exercise 1: Create and configure an Azure Site Recovery vault

The main tasks for this exercise are as follows:

1. Create an Azure Site Recovery vault
2. Configure the Azure Site Recovery vault

12.4.2.1 Task 1: Create an Azure Site Recovery vault

1. Within the Remote Desktop session to **az30312a-hv-vm**, start Microsoft Edge, navigate to the Azure portal, and sign in by providing credentials of a user account with the Owner role in the subscription you will be using in this lab.
2. In the Azure portal, search for and select **Recovery Services vaults** and, on the **Recovery Services vaults** blade, select **+ New**.
3. On the **Basics** tab of the **Create Recovery Services vault** blade, specify the following settings (leave others with their default values) and select **Review + create**:

Setting	Value
Subscription	the name of the Azure subscription you are using in this lab
Resource group	the name of a new resource group **az30312b-labRG**
Vault name	**az30312b-rsvault**
Location	the name of the Azure region different from the one into which you deployed the virtual machine earlier i

4. On the **Review + create** tab of the **Create Recovery Services vault** blade, select **Create**:

    > **Note**: By default, the default configuration for Storage Replication type is set to Geo-redundant (GRS) and Soft Delete is enabled. You will change these settings in the lab to simplify deprovisioning, but you should use them in your production environments.

12.4.2.2 Task 2: Configure the Azure Site Recovery vault

1. In the Azure portal, search for and select **Recovery Services vaults** and, on the **Recovery Services vaults** blade, select **az30312b-rsvault**.
2. On the **az30312b-rsvault** blade, select **Properties**.
3. On the **az30312b-rsvault | Properties** blade, select the **Update** link under the **Backup Configuration** label.
4. On the **Backup Configuration** blade, set **Storage replication type** to **Locally-redundant**, select **Save** and close the **Backup Configuration** blade.

    > **Note**: Storage replication type cannot be changed once you start protecting items.

5. On the **az30312b-rsvault | Properties** blade, select the **Update** link under the **Security Settings** label.

6. On the **Security Settings** blade, set **Soft Delete** to **Disable**, select **Save** and close the **Security Settings** blade.

12.4.3 Exercise 2: Implement Hyper-V protection by using Azure Site Recovery vault

The main tasks for this exercise are as follows:

1. Implement the target Azure environment
2. Implement protection of a Hyper-V virtual machine
3. Perform a failover of the Hyper-V virtual machine
4. Remove Azure resources deployed in the lab

12.4.3.1 Task 1: Implement the target Azure environment

1. In the Azure portal, search for and select **Virtual networks** and, on the **Virtual networks** blade, select **+ New**.
2. On the **Basics** tab of the **Create virtual network** blade, specify the following settings (leave others with their default values) and select **Next: IP Addresses**:

Setting	Value
Subscription	the name of the Azure subscription you are using in this lab
Resource group	the name of a new resource group **az30312c-labRG**
Name	**az30312c-dr-vnet**
Region	the name of the Azure region into which you deployed the Recovery Services vault in the previous exercis

3. On the **IP addresses** tab of the **Create virtual network** blade, in the **IPv4 address space** text box, type **10.7.0.0/16** and select **+ Add subnet**.
4. On the **Add subnet** blade, specify the following settings (leave others with their default values) and select **Add**:

Setting	Value
Subnet name	**subnet0**
Subnet address range	**10.7.0.0/24**

5. Back on the **IP addresses** tab of the **Create virtual network** blade, select **Review + create**.
6. On the **Review + create** tab of the **Create virtual network** blade, select **Create**.
7. In the Azure portal, search for and select **Virtual networks** and, on the **Virtual networks** blade, select **+ New**.
8. On the **Basics** tab of the **Create virtual network** blade, specify the following settings (leave others with their default values) and select **Next: IP Addresses**:

Setting	Value
Subscription	the name of the Azure subscription you are using in this lab
Resource group	**az30312c-labRG**
Name	**az30312c-test-vnet**
Region	the name of the Azure region into which you deployed the Recovery Services vault in the previous exercis

9. On the **IP addresses** tab of the **Create virtual network** blade, in the **IPv4 address space** text box, type **10.7.0.0/16** and select **+ Add subnet**.
10. On the **Add subnet** blade, specify the following settings (leave others with their default values) and select **Add**:

Setting	Value
Subnet name	**subnet0**
Subnet address range	**10.7.0.0/24**

11. Back on the **IP addresses** tab of the **Create virtual network** blade, select **Review + create**.
12. On the **Review + create** tab of the **Create virtual network** blade, select **Create**.
13. In the Azure portal, search for and select **Storage accounts** and, on the **Storage accounts** blade, select **+ New**.
14. On the **Basics** tab of the **Create storage account** blade, specify the following settings (leave others with their default values):

Setting	Value
Subscription	the name of the Azure subscription you are using in this lab
Resource group	**az30312c-labRG**
Storage account name	any globally unique name between 3 and 24 in length consisting of letters and digits
Location	the name of the Azure region in which you created the virtual network earlier in this task
Performance	**Standard: Recommended for most scenarios (general-purpose v2)**
Redundancy	**Locally redundant storage (LRS)**

15. On the **Basics** tab of the **Create storage account** blade, select **Review + create**.
16. On the **Review + create** tab of the **Create storage account** blade, select **Create**.

12.4.3.2 Task 2: Implement protection of a Hyper-V virtual machine

1. Within the Remote Desktop session to **az30312a-hv-vm**, in the Azure portal, on the **az30312b-rsvault** blade, in the **Site Recovery**section, select **Getting started**.
2. On the **az30312b-rsvault | Site Recovery** blade, in the **Hyper-V machines to Azure** section, select **1. Prepare infrastructure**.
3. On the **Deployment planning** tab of the **Prepare infrastructure** blade, in the **Deployment planning completed?** drop-down list, select **Yes, I have done it** and select **Next**.
4. On the **Source settings** tab of the **Prepare infrastructure** blade, next to the **Are you Using System Center VMM to manage Hyper-V hosts** label, select the **No** option.
5. On the **Source settings** tab of the **Prepare infrastructure** blade, select the **Add Hyper-V site** link, on the **Create Hyper-V Site** blade, in the **Name** text box, type **az30312b Hyper-V site** and select **OK**:
6. On the **Source settings** tab of the **Prepare infrastructure** blade, select the **Add Hyper-V server** link.
7. On the **Add Server** blade, select the **Download** link in step 3 of the procedure for adding on-premises Hyper-V hosts in order to download the Microsoft Azure Site Recovery Provider.
8. When the download completes, select **Open file** then select **Run** to launch **AzureSiteRecoveryProvider.exe**. This will start the **Azure Site Recovery Provider Setup (Hyper-V server)** wizard.
9. On the **Microsoft Update** page, select **Off** and select **Next**.
10. On the **Provider installation** page, select **Install**.
11. Switch to the Azure portal and, on the **Add Server** blade, select the **Download** button in step 4 of the procedure for registering on-premises Hyper-V hosts in order to download the vault registration key. If prompted, select **Save** to save the vault credentials file in the **Downloads** folder.
12. Switch to the **Provider installation** wizard window and select **Register**. This will start the **Microsoft Azure Site Recovery Registration Wizard**.

13. On the **Vault Settings** page of the **Microsoft Azure Site Recovery Registration Wizard**, select **Browse**, in the **Open** window, navigate to the **Downloads** folder, select the vault credentials file, and select **Open**.
14. Back on the **Vault Settings** page of the **Microsoft Azure Site Recovery Registration Wizard**, select **Next**.
15. On the **Proxy Settings** page of the **Microsoft Azure Site Recovery Registration Wizard**, accept the default settings and select **Next**.
16. On the **Registration** page of the **Microsoft Azure Site Recovery Registration Wizard**, select **Finish**.
17. Switch back to the browser window displaying the Azure portal and refresh the page. When prompted, select **Leave the page** or **Reload** depending on the version of the browser.
18. Back on the **az30312b-rsvault | Site Recovery** blade, in the **Hyper-V machines to Azure** section, select **1. Prepare infrastructure**.
19. On the **Deployment planning** tab of the **Prepare infrastructure** blade, in the **Deployment planning completed?** drop-down list, select **Yes, I have done it** and select **Next**.
20. On the **Source settings** tab of the **Prepare infrastructure** blade, next to the **Are you Using System Center VMM to manage Hyper-V hosts** label, select the **No** option.
21. Verify that the **Hyper-V site** and **Hyper-V servers** settings are set correctly and select **Next**.
22. On the **Target settings** tab of the **Prepare infrastructure** blade, accept the default settings and select **Next**.
23. On the **Replication policy** tab of the **Prepare infrastructure** blade, select **Create new policy and associate**.
24. On the **Create and associate policy** blade, specify the following settings (leave others with their default values) and select **OK**:

Setting	Value
Name	**az30312b replication policy**
Copy frequency	**30 seconds**

25. Back on the **Replication policy** tab of the **Prepare infrastructure** blade, wait until the site has been associated with the policy and select **Next**.
26. On the **Review** tab of the **Prepare infrastructure** blade, select **Prepare**.
27. On the **az30312b-rsvault | Site Recovery** blade, in the **Hyper-V machines to Azure** section, select **2. Enable replication**.
28. On the **Source environment** tab of the **Enable replication** blade, in the **Source location** drop-down list, select **az30312b Hyper-V site** and select **Next**.
29. On the **Target environment** tab of the **Enable replication** blade, specify the following settings (leave others with their default values) and select **Next**:

Setting	Value
Subscription	the name of the Azure subscription you are using in this lab
Post-failover resource group	**az30312c-labRG**
Post-failover deployment model	**Resource Manager**
Storage account	the name of the storage account you created in the first task of this exercise
Network	Configure now for selected machines
Virtual network	**az30312c-dr-vnet**
Subnet	**subnet0 (10.7.0.0/24)**

30. On the **Virtual machine selection** tab of the **Enable replication** blade, select the **az30312a-vm1** checkbox and select **Next**.

31. On the **Replication settings** tab of the **Enable replication** blade, in the **Defaults** row and **OS type** column, select **Windows** from the drop-down list and select **Next**.
32. On the **Replication policy** tab of the **Enable replication** blade, accept the default settings and select **Next**.
33. On the **Review** tab of the **Enable replication** blade, select **Enable replication**.

12.4.3.3 Task 3: Review Azure VM replication settings

1. In the Azure portal, back on the **az30312b-rsvault | Site Recovery** blade, in the vertical menu, select **Replicated items**.
2. On the **az30312b-rsvault | Replicated items** blade, ensure that there is an entry representing the **az30312a-vm1** virtual machine and verify that its **Replication Health** is listed as **Healthy** and that its **Status** is listed as either **Enabling protection**, **Waiting for first recovery point**, or displaying a current percentage of synchronization progress.

   > **Note**: You might need to wait a few minutes until the **az30312a-vm1** entry appears on the **az30312b-rsvault | Replicated items** blade.

3. On the **az30312b-rsvault | Replicated items** blade, select the **az30312a-vm1** entry.
4. On the **az30312a-vm1** replicated items blade, review the **Health and status**, **Failover readiness**, **Latest recovery points**, and **Infrastructure view** sections. Note the **Planned Failover**, **Failover** and **Test Failover** toolbar icons.

   > **Note**: Wait until the status changes to **Protected**. This might take additional 15 minutes. You will need to refresh the browser page for the status to be updated.

5. On the **az30312a-vm1** replicated items blade, select **Latest recovery points** and review **Latest crash-consistent** and **Latest app-consistent** recovery points.

12.4.3.4 Task 4: Perform a failover of the Hyper-V virtual machine

1. Within the Remote Desktop session to **az30312a-hv-vm**, in the browser window displaying the Azure portal, on the **az30312a-vm1** replicated items blade, select **Test failover**.
2. On the **Test failover** blade, specify the following settings (leave others with their default values) and select **OK**:

Setting	Value
Choose a recovery point	the default option
Azure virtual network	**az30312c-test-vnet**

3. In the Azure portal, navigate back to the **az30312b-rsvault** blade and select **Site Recovery jobs**. Wait until the status of the **Test failover** job is listed as **Successful**.
4. In the Azure portal, search for and select **Virtual machines** and, on the **Virtual machines** blade, note the entry representing the newly provisioned virtual machine **az30312a-vm1-test**.
5. In the Azure portal, navigate back to the on the **az30312a-vm1** replicated items blade and select **Cleanup test failover**.
6. On the **Test failover cleanup** blade, select the checkbox **Testing is complete. Delete test failover virtual machine(s)** and select **OK**.
7. Once the test failover cleanup job completes, refresh the browser page displaying the **az30312a-vm1** replicated items blade and note that you have the option to perform planned and unplanned failover.
8. On the **az30312a-vm1** replicated items blade, select **Planned failover**.
9. On the **Planned failover** blade, note that the failover direction settings are already set and not modifiable.
10. Close the **Planned failover** blade and, on the **az30312a-vm1** replicated items blade, select **Failover**.
11. On the **Failover** blade, note the available options geared towards minimizing potential data loss.
12. Close the **Failover** blade.

12.4.3.5 Task 5: Remove Azure resources deployed in the lab

1. Within the Remote Desktop session to **az30312a-hv-vm**, in the browser window displaying the Azure portal, start a PowerShell session within the Cloud Shell pane.
2. From the Cloud Shell pane, run the following to list the resource group you created in this exercise:

   ```
   Get-AzResourceGroup -Name 'az30312*'
   ```

   > **Note**: Verify that the output contains only the resource group you created in this lab. This group will be deleted in this task.

3. From the Cloud Shell pane, run the following to delete the resource group you created in this lab

   ```
   Get-AzResourceGroup -Name 'az30312*' | Remove-AzResourceGroup -Force -AsJob
   ```

4. Close the Cloud Shell pane.

12.5 lab: title: '14A: Implementing an Azure App Service Web App with a Staging Slot' module: 'Module 14: Implement an Application Infrastructure'

13 Lab: Implementing an Azure App Service Web App with a Staging Slot

14 Student lab manual

14.1 Lab scenario

Adatum Corporation has a number of web apps that are updated on relatively frequent basis. While Adatum has not yet fully embraced DevOps principles, it relies on Git as its version control and is exploring the options to streamline the app updates. As Adatum is transitioning some of its workloads to Azure, the Adatum Enterprise Architecture team decided to evaluate the use of Azure App Service and its deployment slots to accomplish this objective.

Deployment slots are live apps with their own host names. App content and configurations elements can be swapped between two deployment slots, including the production slot. Deploying apps to a non-production slot has the following benefits:

- It is possible to validate app changes in a staging deployment slot before swapping it with the production slot.
- Deploying an app to a slot first and swapping it into production makes sure that all instances of the slot are warmed up before being swapped into production. This eliminates downtime when during app deployment. The traffic redirection is seamless, and no requests are dropped because of swap operations. This workflow can be automated by configuring auto swap when pre-swap validation is not needed.
- After a swap, the slot with previously staged app has the previous production app. If the changes swapped into the production slot need to be reversed, this simply involves another swap immediately to return to the last known good state.

Deployment slots facilitate two common deployment patterns: blue/green and A/B testing. Blue-green deployment involves deploying an update into a production environment that is separate from the live application. After the deployment is validated, traffic routing is switched to the updated version. A/B testing involves gradually routing some of the traffic to a staging site in order to test a new version of an app.

The Adatum Architecture team wants to use Azure App Service web apps with deployment slots in order to test these two deployment patterns:

- Blue/Green deployments
- A/B testing

14.2 Objectives

After completing this lab, you will be able to:

- Implement Blue/Green deployment pattern by using deployment slots of Azure App Service web apps
- Perform A/B testing by using deployment slots of Azure App Service web apps

14.3 Lab Environment

Estimated Time: 60 minutes

14.4 Lab Files

None

14.5 Instructions

14.5.1 Exercise 1: Implement an Azure App Service web app

1. Deploy an Azure App Service web app
2. Create an App Service web app deployment slot

14.5.1.1 Task 1: Deploy an Azure App Service web app

1. From your lab computer, start a web browser, navigate to the Azure portal, and sign in by providing credentials of a user account with the Owner role in the subscription you will be using in this lab.
2. In the Azure portal, open **Cloud Shell** pane by selecting on the toolbar icon directly to the right of the search textbox.
3. If prompted to select either **Bash** or **PowerShell**, select **Bash**.

   > **Note**: If this is the first time you are starting **Cloud Shell** and you are presented with the **You have no storage mounted** message, select the subscription you are using in this lab, and select **Create storage**.

4. From the Cloud Shell pane, run the following to create a new directory named **az30314a1** and set it as your current directory:

   ```
   mkdir az30314a1
   cd ~/az30314a1/
   ```

5. From the Cloud Shell pane, run the following to clone a sample app repository to the **az30314a1** directory:

   ```
   REPO=https://github.com/Azure-Samples/html-docs-hello-world.git
   git clone $REPO
   cd html-docs-hello-world
   ```

6. From the Cloud Shell pane, run the following to configure a deployment user:

   ```
   USERNAME=az30314user$RANDOM
   PASSWORD=az30314pass$RANDOM
   az webapp deployment user set --user-name $USERNAME --password $PASSWORD
   echo $USERNAME
   echo $PASSWORD
   ```

7. Verify that the deployment user was created successfully. If you receive an error message indicating a conflict, repeat the previous step.

   > **Note**: Make sure to record the value of the username and the corresponding password.

8. From the Cloud Shell pane, run the following to create the resource group which will host the App Service web app (replace the `<location>` placeholder with the name of the Azure region that is available in your subscription and which is closest to the location of your lab computer):

   ```
   LOCATION='<location>'
   RGNAME='az30314a-labRG'
   az group create --location $LOCATION --resource-group $RGNAME
   ```

9. From the Cloud Shell pane, run the following to create a new App Service plan:

```
SPNAME=az30314asp$LOCATION$RANDOM
az appservice plan create --name $SPNAME --resource-group $RGNAME --location $LOCATION --sku S1
```

10. From the Cloud Shell pane, run the following to create a new, Git-enabled App Service web app:

```
WEBAPPNAME=az30314$RANDOM$RANDOM
az webapp create --name $WEBAPPNAME --resource-group $RGNAME --plan $SPNAME --deployment-local-git
```

    Note: Wait for the deployment to complete.

11. From the Cloud Shell pane, run the following to retrieve the publishing URL of the newly created App Service web app:

```
URL=$(az webapp deployment list-publishing-credentials --name $WEBAPPNAME --resource-group $RGNAME
```

12. From the Cloud Shell pane, run the following to set the git remote alias representing the Git-enabled Azure App Service web app:

```
git remote add azure $URL
```

13. From the Cloud Shell pane, run the following to push to the Azure remote with git push azure master:

```
git push azure master
```

    Note: Wait for the deployment to complete.

14. From the Cloud Shell pane, run the following to identify the FQDN of the newly deployed App Service web app.

```
az webapp show --name $WEBAPPNAME --resource-group $RGNAME --query defaultHostName --output tsv
```

15. Close the Cloud Shell pane.

14.5.1.2 Task 2: Create an App Service web app deployment slot

1. In the Azure portal, search for and select **App Services** and, on the **App Services** blade, select the newly created App Service web app.
2. In the Azure portal, navigate to the blade displaying the newly deployed App Service web app, select the **URL** link, and verify that it displays the **Azure App Service - Sample Static HTML Site**. Leave the browser tab open.
3. On the App Service web app blade, in the **Deployment** section, select **Deployment slots** and then select **+ Add Slot**.
4. On the **Add a slot** blade, specify the following settings, select **Add**, and then select **Close**.

Setting	Value
Name	**staging**
Clone settings from	the name of the web app

14.5.2 Exercise 2: Manage App Service web app deployment slots

The main tasks for this exercise are as follows:

1. Deploy web content to an App Service web app staging slot
2. Swap App Service web app staging slots
3. Configure A/B testing
4. Remove Azure resources deployed in the lab

14.5.2.1 Task 1: Deploy web content to an App Service web app staging slot

1. In the Azure portal, open **Cloud Shell** pane by selecting on the toolbar icon directly to the right of the search textbox.
2. From the Cloud Shell pane, run the following to ensure that the current set **az30314a1/html-docs-hello-world** as the current directory:

```
cd ~/az30314a1/html-docs-hello-world
```

3. In the Cloud Shell pane, run the following to start the built-in editor:

   ```
   code index.html
   ```

4. In the Cloud Shell pane, in the code editor, replace the line:

   ```
   <h1>Azure App Service - Sample Static HTML Site</h1>
   ```

   with the following line:

   ```
   <h1>Azure App Service - Sample Static HTML Site v1.0.1</h1>
   ```

5. Save the changes and close the editor window.
6. From the Cloud Shell pane, run the following to specify the required global git configuration settings:

   ```
   git config --global user.email "user@az30314.com"
   git config --global user.name "user az30314"
   ```

7. From the Cloud Shell pane, run the following to commit the change you applied locally to the master branch:

   ```
   git add index.html
   git commit -m 'v1.0.1'
   ```

8. From the Cloud Shell pane, run the following to retrieve the publishing URL of the newly created staging slot of the App Service web app:

   ```
   RGNAME='az30314a-labRG'
   WEBAPPNAME=$(az webapp list --resource-group $RGNAME --query "[?starts_with(name,'az30314')]".name
   SLOTNAME='staging'
   URLSTAGING=$(az webapp deployment list-publishing-credentials --name $WEBAPPNAME --slot $SLOTNAME
   ```

9. From the Cloud Shell pane, run the following to set the git remote alias representing the staging slot of the Git-enabled Azure App Service web app:

   ```
   git remote add azure-staging $URLSTAGING
   ```

10. From the Cloud Shell pane, run the following to push to the Azure remote with git push azure master:

    ```
    git push azure-staging master
    ```

    > **Note**: Wait for the deployment to complete.

11. Close the Cloud Shell pane.
12. In the Azure portal, navigate to the blade displaying the deployment slots of the App Service web app and select the staging slot.
13. On the blade displaying the staging slot overview, select the **URL** link.

14.5.2.2 Task 2: Swap App Service web app staging slots

1. In the Azure portal, navigate back to the blade displaying the App Service web app and select **Deployment slots**.
2. On the deployment slots blade, select **Swap**.
3. On the **Swap** blade, select **Swap** and then select **Close**.
4. Switch to the browser tab showing the App Service web app and refresh the browser window. Verify that it displays the changes you deployed to the staging slot.
5. Switch to the browser tab showing the staging slot of the App Service web app and refresh the browser window. Verify that it displays the original web page included in the original deployment.

14.5.2.3 Task 3: Configure A/B testing

1. In the Azure portal, navigate back to the blade displaying the deployment slots of the App Service web app.

2. In the Azure portal, on the blade displaying the App Service web app deployment slots, in the row displaying the staging slot, set the value in the **TRAFFIC %** column to 50. This will automatically set the value of **TRAFFIC %** in the row representing the production slot to 50.
3. On the blade displaying the App Service web app deployment slots, select **Save**.
4. In the Azure portal, open **Cloud Shell** pane by selecting on the toolbar icon directly to the right of the search textbox.
5. From the Cloud Shell pane, run the following to verify set the variables representing the name of the target web app and its distribution group:

   ```
   RGNAME='az30314a-labRG'
   WEBAPPNAME=$(az webapp list --resource-group $RGNAME --query "[?starts_with(name,'az30314')]".name
   ```

6. From the Cloud Shell pane, run the following several times to identify the traffic distribution between the two slots.

   ```
   curl -H 'Cache-Control: no-cache' https://$WEBAPPNAME.azurewebsites.net --stderr - | grep '<h1>Azu
   ```

   > **Note**: Traffic distribution is not entirely deterministic, but you should see several responses from each target site.

14.5.2.4 Task 4: Remove Azure resources deployed in the lab

1. From the Cloud Shell pane, run the following to list the resource group you created in this exercise:

   ```
   az group list --query "[?starts_with(name,'az30314')]".name --output tsv
   ```

   > **Note**: Verify that the output contains only the resource group you created in this lab. This group will be deleted in this task.

2. From the Cloud Shell pane, run the following to delete the resource group you created in this lab

   ```
   az group list --query "[?starts_with(name,'az30314')]".name --output tsv | xargs -L1 bash -c 'az g
   ```

3. From the Cloud Shell pane, run the following to remove the **az30314a1** directory:

   ```
   rm -r -f ~/az30314a1
   ```

4. Close the Cloud Shell pane.

14.6 lab: title: '14B: Configuring a Message-Based Integration Architecture' module: 'Module 14: Implement an Application Infrastructure'

15 Lab: Configuring a Message-Based Integration Architecture

16 Student lab manual

16.1 Lab scenario

Adatum Corporation has several web applications that process files uploaded in regular intervals to their on-premises file servers. Files sizes vary, but they can reach up to 100 MB. Adatum is considering migrating the applications to Azure App Service or Azure Functions-based apps and using Azure Storage to host uploaded files. You plan to test two scenarios:

- using Azure Functions to automatically process new blobs uploaded to an Azure Storage container.
  - using Event Grid to generate Azure Storage queue messages that will reference new blobs uploaded to an Azure Storage container.

These scenarios are intended to address a challenge common to a messaging-based architecture, when sending, receiving, and manipulating large messages. Sending large messages to a message queue directly is not recommended as they would require more resources to be used, result in more bandwidth to be consumed, and negatively impact the processing speed, since messaging platforms are usually fine-tuned to handle high volumes of small messages. In addition, messaging platforms usually limit the maximum message size they can process.

One potential solution is to store the message payload into an external service, like Azure Blob Store, Azure SQL or Azure Cosmos DB, get the reference to the stored payload and then send to the message bus only that reference. This architectural pattern is known as ”claim check”. The clients interested in processing that specific message can use the obtained reference to retrieve the payload, if needed.

On Azure this pattern can be implemented in several ways and with different technologies, but it typically it relies on events to either automate the claim check generation and to push it into the message bus to be used by clients or to trigger payload processing directly. One of the common components included in such implementations is Event Grid, which is an event routing service responsible for delivery of events within a configurable period (up to 24 hours). After that, events are either discarded or dead lettered. If archival of event contents or replayability of event stream are needed, it is possible to facilitate this requirement by setting up an Event Grid subscription to the Event Hub or a queue in Azure Storage where messages can be retained for longer periods and archival of messages is supported.

In this lab, you will use Azure Storage Blob service to store files to be processed. A client just needs to drop the files to be shared into a designated Azure Blob container. In the first exercise, the files will be consumed directly by an Azure Function, leveraging its serverless nature. You will also take advantage of the Application Insights to provide instrumentation, facilitating monitoring and analyzing file processing. In the second exercise, you will use Event Grid to automatically generate a claim check message and send it to an Azure Storage queue. This allows a client application to poll the queue, get the message and then use the stored reference data to download the payload directly from Azure Blob Storage.

It is worth noting that the Azure Function in the first exercise relies on the Blob Storage trigger. You should opt for Event Grid trigger instead of the Blob storage trigger when dealing with the following requirements:

- blob-only storage accounts: blob storage accounts are supported for blob input and output bindings but not for blob triggers. Blob storage triggers require a general-purpose storage account.
  - high scale: high scale can be loosely defined as containers that have more than 100,000 blobs in them or storage accounts that have more than 100 blob updates per second.
  - reduced latency: if your function app is on the Consumption plan, there can be up to a 10-minute delay in processing new blobs if a function app has gone idle. To avoid this latency, you can use an Event Grid trigger or switch to an App Service plan with the Always On setting enabled.
  - processing of blob delete events: blob delete events are not supported by blob storage triggers.

16.2 Objectives

After completing this lab, you will be able to:

- Configure and validate an Azure Function App Storage Blob trigger
- Configure and validate an Azure Event Grid subscription-based queue messaging

16.3 Lab Environment

Estimated Time: 60 minutes

16.4 Lab Files

None

16.5 Instructions

16.5.1 Exercise 1: Configure and validate an Azure Function App Storage Blob trigger

The main tasks for this exercise are as follows:

1. Configure an Azure Function App Storage Blob trigger
2. Validate an Azure Function App Storage Blob trigger

16.5.1.1 Task 1: Configure an Azure Function App Storage Blob trigger

1. From your lab computer, start a web browser, navigate to the Azure portal, and sign in by providing credentials of a user account with the Owner role in the subscription you will be using in this lab.

2. In the Azure portal, open **Cloud Shell** pane by selecting on the toolbar icon directly to the right of the search textbox.
3. If prompted to select either **Bash** or **PowerShell**, select **Bash**.

   > **Note**: If this is the first time you are starting **Cloud Shell** and you are presented with the **You have no storage mounted** message, select the subscription you are using in this lab, and select **Create storage**.

4. From the Cloud Shell pane, run the following to generate a pseudo-random string of characters that will be used as a prefix for names of resources you will provision in this exercise:

```
export PREFIX=$(echo `openssl rand -base64 5 | cut -c1-7 | tr '[:upper:]' '[:lower:]' | tr -cd '[[
```

5. From the Cloud Shell pane, run the following to designate the Azure region into which you want to provision resources in this lab (replace the `<Azure region>` placeholder with the name of the Azure region that is available in your subscription and which is closest to the location of your lab computer):

```
export LOCATION='<Azure region>'
```

6. From the Cloud Shell pane, run the following to create a resource group that will host all resources that you will provision in this lab:

```
export RESOURCE_GROUP_NAME='az30314b-labRG'

az group create --name "${RESOURCE_GROUP_NAME}" --location "$LOCATION"
```

7. From the Cloud Shell pane, run the following to create an Azure Storage account that will host container with blobs to be processed by the Azure function:

```
export STORAGE_ACCOUNT_NAME="az30314b${PREFIX}"

export CONTAINER_NAME="workitems"

export STORAGE_ACCOUNT=$(az storage account create --name "${STORAGE_ACCOUNT_NAME}" --kind "Storag
```

   > **Note**: The same storage account will be also used by the Azure function to facilitate its own processing requirements. In real-world scenarios, you might want to consider creating a separate storage account for this purpose.

8. From the Cloud Shell pane, run the following to create a variable storing the value of the connection string property of the Azure Storage account:

```
export STORAGE_CONNECTION_STRING=$(az storage account show-connection-string --name "${STORAGE_ACC
```

9. From the Cloud Shell pane, run the following to create a container that will host blobs to be processed by the Azure function:

```
az storage container create --name "${CONTAINER_NAME}" --account-name "${STORAGE_ACCOUNT_NAME}" --
```

10. From the Cloud Shell pane, run the following to create an Application Insights resource that will provide monitoring of the Azure Function processing blobs and store its key in a variable:

```
export APPLICATION_INSIGHTS_NAME="az30314bi${PREFIX}"

az resource create --name "${APPLICATION_INSIGHTS_NAME}" --location "${LOCATION}" --properties '{"

export APPINSIGHTS_KEY=$(az resource show --name "${APPLICATION_INSIGHTS_NAME}" --query "propertie
```

11. From the Cloud Shell pane, run the following to create the Azure Function that will process events corresponding to creation of Azure Storage blobs:

```
export FUNCTION_NAME="az30314f${PREFIX}"

az functionapp create --name "${FUNCTION_NAME}" --resource-group "${RESOURCE_GROUP_NAME}" --app-in
```

12. From the Cloud Shell pane, run the following to configure Application Settings of the newly created function, linking it to the Application Insights and Azure Storage account:

```
az functionapp config appsettings set --name "${FUNCTION_NAME}" --resource-group "${RESOURCE_GROUP

az functionapp config appsettings set --name "${FUNCTION_NAME}" --resource-group "${RESOURCE_GROUP
```

13. Switch to the Azure portal and navigate to the blade of the Azure Function app you created earlier in this task.
14. On the Azure Function app blade, select **Functions** and then, select + **Add**.
15. On the **Add Function** blade, select **Azure Blob Storage trigger** template.
16. On the **Add Function** blade, specify the following and select **Add** to create a new function within the Azure function:

Setting	Value
Name	**BlobTrigger**
Path	**workitems/{name}**
Storage account connection	**STORAGE_CONNECTION_STRING**

17. On the Azure Function app **BlobTrigger** function blade, select **Code + Test** and review the content of the run.csx file.

```
public static void Run(Stream myBlob, string name, ILogger log)
{
    log.LogInformation($"C# Blob trigger function Processed blob\n Name:{name} \n Size: {myBlob.Le
}
```

> **Note**: By default, the function is configured to simply log the event corresponding to creation of a new blob. In order to carry out blob processing tasks, you would modify the content of this file.

16.5.1.2 Task 2: Validate an Azure Function App Storage Blob trigger

1. If necessary, restart the Bash session in the Cloud Shell.
2. From the Cloud Shell pane, run the following to repopulate variables that you used in the previous task:

```
export RESOURCE_GROUP_NAME='az30314b-labRG'

export STORAGE_ACCOUNT_NAME="$(az storage account list --resource-group "${RESOURCE_GROUP_NAME}" -

export CONTAINER_NAME="workitems"
```

3. From the Cloud Shell pane, run the following to upload a test blob to the Azure Storage account you created earlier in this task:

```
export STORAGE_ACCESS_KEY="$(az storage account keys list --account-name "${STORAGE_ACCOUNT_NAME}"

export WORKITEM='workitem1.txt'

touch "${WORKITEM}"

az storage blob upload --file "${WORKITEM}" --container-name "${CONTAINER_NAME}" --name "${WORKITE
```

4. In the Azure portal, navigate back to the blade displaying the Azure Function app you created in the previous task.
5. From the Function app, click **Functions**.
6. In the list of functions, click the name of your function.
7. On the Azure Function app blade, select **Monitor** entry.
8. Note a single event entry representing uploading of the blob. Select the entry to view the **Invocation Traces** details.

> **Note**: Since the Azure function app in this exercise runs in the Consumption plan, there may be a delay of up to several minutes between uploading a blob and the function being triggered.

It is possible to minimize the latency by implementing the Function app by using an App Service (rather than Consumption) plan.

9. Under **Invocation Traces**, click **Run query in Application Insights**.
10. In the Application Insights portal, review the autogenerated Kusto query and its results.

16.5.2 Exercise 2: Configure and validate an Azure Event Grid subscription-based queue messaging

The main tasks for this exercise are as follows:

1. Configure an Azure Event Grid subscription-based queue messaging
2. Validate an Azure Event Grid subscription-based queue messaging
3. Remove Azure resources deployed in the lab

16.5.2.1 Task 1: Configure an Azure Event Grid subscription-based queue messaging

1. If necessary, restart the Bash session in the Cloud Shell.
2. From the Cloud Shell pane, run the following to register the eventgrid resource provider in your subscription:

   ```
   az provider register --namespace microsoft.eventgrid
   ```

3. From the Cloud Shell pane, run the following to generate a pseudo-random string of characters that will be used as a prefix for names of resources you will provision in this exercise:

   ```
   export PREFIX=$(echo `openssl rand -base64 5 | cut -c1-7 | tr '[:upper:]' '[:lower:]' | tr -cd '[[
   ```

4. From the Cloud Shell pane, run the following to identify the Azure region hosting the target resource group and its existing resources:

   ```
   export RESOURCE_GROUP_NAME_EXISTING='az30314b-labRG'

   export LOCATION=$(az group list --query "[?name == '${RESOURCE_GROUP_NAME_EXISTING}'].location" --

   export RESOURCE_GROUP_NAME='az30314c-labRG'

   az group create --name "${RESOURCE_GROUP_NAME}" --location $LOCATION
   ```

5. From the Cloud Shell pane, run the following to create an Azure Storage account that will host a container to be used by the Event Grid subscription that you will configure in this task:

   ```
   export STORAGE_ACCOUNT_NAME="az30314cst${PREFIX}"

   export CONTAINER_NAME="workitems"

   export STORAGE_ACCOUNT=$(az storage account create --name "${STORAGE_ACCOUNT_NAME}" --kind "Storag
   ```

6. From the Cloud Shell pane, run the following to create a variable storing the value of the connection string property of the Azure Storage account:

   ```
   export STORAGE_CONNECTION_STRING=$(az storage account show-connection-string --name "${STORAGE_ACC
   ```

7. From the Cloud Shell pane, run the following to create a container that will to be used by the Event Grid subscription:

   ```
   az storage container create --name "${CONTAINER_NAME}" --account-name "${STORAGE_ACCOUNT_NAME}" --
   ```

8. From the Cloud Shell pane, run the following to create a variable storing the value of the Resource Id property of the Azure Storage account:

   ```
   export STORAGE_ACCOUNT_ID=$(az storage account show --name "${STORAGE_ACCOUNT_NAME}" --query "id"
   ```

9. From the Cloud Shell pane, run the following to create the Storage Account queue that will store messages generated by the Event Grid subscription that you will configure in this task:

```
export QUEUE_NAME="az30314cq${PREFIX}"

az storage queue create --name "${QUEUE_NAME}" --account-name "${STORAGE_ACCOUNT_NAME}" --connecti
```

10. From the Cloud Shell pane, run the following to create the Event Grid subscription that will facilitate generation of messages in Azure Storage queue in response to blob uploads to the designated container in the Azure Storage account:

```
export QUEUE_SUBSCRIPTION_NAME="az30314cqsub${PREFIX}"

az eventgrid event-subscription create --name "${QUEUE_SUBSCRIPTION_NAME}" --included-event-types
```

16.5.2.2 Task 2: Validate an Azure Event Grid subscription-based queue messaging

1. From the Cloud Shell pane, run the following to upload a test blob to the Azure Storage account you created earlier in this task:

```
export AZURE_STORAGE_ACCESS_KEY="$(az storage account keys list --account-name "${STORAGE_ACCOUNT_

export WORKITEM='workitem2.txt'

touch "${WORKITEM}"

az storage blob upload --file "${WORKITEM}" --container-name "${CONTAINER_NAME}" --name "${WORKITE
```

2. In the Azure portal, navigate to the blade displaying the Azure Storage account you created in the previous task of this exercise.
3. On the blade of the Azure Storage account, select **Queues** to display the list of its queues.
4. Select the entry representing the queue you created in the previous task of this exercise.
5. Note that the queue contains a single message. Select its entry to display the **Message properties** blade.
6. In the **MESSAGE BODY**, note the value of the **url** property, representing the URL of the Azure Storage blob you uploaded in the previous task.

16.5.2.3 Task 3: Remove Azure resources deployed in the lab

1. From the Cloud Shell pane, run the following to list the resource group you created in this exercise:

```
az group list --query "[?starts_with(name,'az30314')]".name --output tsv
```

   Note: Verify that the output contains only the resource group you created in this lab. This group will be deleted in this task.

2. From the Cloud Shell pane, run the following to delete the resource group you created in this lab

```
az group list --query "[?starts_with(name,'az30314')]".name --output tsv | xargs -L1 bash -c 'az g
```

3. Close the Cloud Shell pane.