Module 5 - Deploying Azure IaaS Solutions

Lab 1: Implementing highly available Azure IaaS compute architecture

Scenario

Adatum Corporation has a number of 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 Architecutre 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 Architrecture 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 occassionally 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

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

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

Task 0: Download the lab files.

- [ ] 1. From the lab virtual machine, click **Start** and search for **PowerShell** then open **PowerShell as Administrator**.

- [ ] 2. Run the following commands to download the latest version of the lab files to the virtual machine.

> **Note**: If any of the commands fail, run them again until they are successfull.

```
Start-BitsTransfer -Source 'https://github.com/MicrosoftLearning/AZ-303-Microsoft-Azure-Architect-Technologies/archive/master.zip' -Dest
```

```
Expand-Archive -Path 'D:\master.zip' -DestinationPath 'D:\'
```

```
Move-item -Path "D:\AZ-303-Microsoft-Azure-Architect-Technologies-master\AllFiles\*" -Destination "D:\AllFiles\" -confirm:$false
```

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 VM, start a web browser, navigate to the **Azure Portal**, and sign in with the username sheikhnasirL4RT3@gdcs4.com and password Glyj4pWib48hPAcg

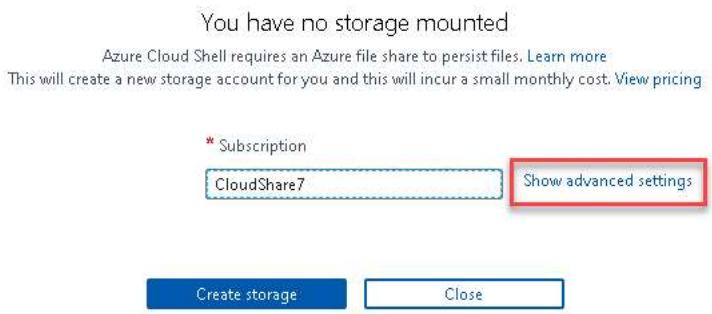
- [ ] 2. In the Azure portal, navigate to **Resource Groups** and identify the Resource Groups that have been created for you.

- [ ] 3. In the Azure portal, open the **Cloud Shell** pane by selecting on the toolbar icon directly to the right of the search textbox.

- [ ] 4. If prompted to select either **Bash** or **PowerShell**, select **Bash**.

> ⚠ **Note**: As 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 follow the instructions below.

- [ ] 5. Click **Show advanced settings**.



- [ ] 6. Select the **East US** region. Select **Use existing** Resource group and select the pre-provisioned resource group for the lab.

- [ ] 7. Enter a name for the storage account **(this must be unique)** and type "clouldshell" as the name of the File share then click **Create Storage**.

- [ ] 8. From the **Cloud Shell pane**, upload the Azure Resource Manager template **D:\AllFiles\Labs\05\azuredeploy30305rga.json.**.
- [ ] 9. From the **Cloud Shell pane**, upload the Azure Resource Manager parameter file **D:\AllFiles\Labs\05\azuredeploy30305rga.parameters.json**.
- [ ] 10. 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-VMJ9JCU1OI --template-file azuredeploy30305rga.json --parameters azuredeploy3
```

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

- [ ] 11. In the Azure portal, close the **Cloud Shell** pane.

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	**CloudShare4**
Resource Group	**az30305a-labRG-VMJ9JCU1OI**

Setting	Value
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	**CloudShare4**
Resource group	**az30305a-labRG-VMJ9JCU1OI**
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.

- [ ] 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** :

> ⚠ **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.

Setting	Value
Subscription	**CloudShare4**
Resource group	**az30305a-labRG-VMJ9JCU1OI**
Source type	**Virtual machine**
Virtual machine	**az30305a-vm0**
Destination	**Select a virtual machine**
Resource group	**az30305a-labRG-VMJ9JCU1OI**
Virtual machine	**az30305a-vm1**
Protocol	**TCP**
Destination port	80

- [ ] 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-VMJ9JCU1OI** 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-VMJ9JCU1OI** 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 update the IP address included in the **custom-allow-rdp** rule of the network security group **az30305a-web-nsg**. This is necessary to account for the new IP address associated with the new Cloud Shell session:

```
RESOURCE_GROUP_NAME=az30305a-labRG-VMJ9JCU1OI
NSG_NAME=az30305a-web-nsg
RULE_NAME=custom-allow-rdp
MY_IP=$(curl -s checkip.dyndns.org | sed -e 's/.*Current IP Address: //' -e 's/<.*$//')
az network nsg rule update --resource-group $RESOURCE_GROUP_NAME --nsg-name $NSG_NAME --name $RULE_NAME --source-address-prefixes $MY_IP
```

- [ ] 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 **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.

- [ ] 16. On the **az303005a-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 without 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 **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.

- [ ] 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.

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 "az30305a-labRG-VMJ9JCU1OI".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 "az30305a-labRG-VMJ9JCU1OI".name --output tsv | xargs -L1 bash -c 'az group delete --name $0 --no-wait --yes'
```

- [ ] 3. Close the Cloud Shell pane.

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

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

- [ ] 1. To save repetitiveness of labs, the resources for exercise have been deployed for you.

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	**CloudShare4**
Resource Group	**az30305b-labRG-VMJ9JCU1OI**
Virtual Network	**az30305b-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.

> ⚠ **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	**CloudShare4**
Resource group	**az30305b-labRG-VMJ9JCU1OI**
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).

- [ ] 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** :

> ⚠ **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.

Setting	Value
Subscription	**CloudShare4**
Resource group	**az30305b-labRG-VMJ9JCU1OI**
Source type	**Virtual machine**
Virtual machine	**az30305b-vm0**
Destination	**Select a virtual machine**
Resource group	**az30305b-labRG-VMJ9JCU1OI**
Virtual machine	**az30305b-vm1**
Protocol	**TCP**
Destination port	80

- [ ] 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 without 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 memebers 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):

> ⚠ **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).

Setting	Value
Name	az303005b-obrule
Frontend IP address	the name of the existing frontend IP address of the **az30305b-lb** load balancer

Setting	Value
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**

- [ ] 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.

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 "az30305b-labRG-VMJ9JCU1OI".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 "az30305b-labRG-VMJ9JCU1OI".name --output tsv | xargs -L1 bash -c 'az group delete --name $0 --no-wait --yes'
```

- [ ] 4. Close the Cloud Shell pane.

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

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. To save repetitiveness of labs, the resources for exercise have been deployed for you.

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	**CloudShare4**
Resource Group	**az30305c-labRG-VMJ9JCU1OI**
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 use **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 <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

- [ ] 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.

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

Task 1: Configure autoscaling of an Azure VM Scale Set

- [ ] 1. In the Azure portal, navigate to the **az30305c-labRG-VMJ9JCU1OI** 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

Setting	Value
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):

> ⚠ **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** .

Setting	Value
Time aggregation	**Maximum**
Metric namespace	**Virtual Machine Host**
Metric name	**Percentage CPU**
VMName Operator	**=**
Dimension values	**2 selected**
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

- [ ] 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**.

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** blade, review the **CPU (average)** chart and verify that the CPU utilization of the Application Gateway increased sufficiently to trigger scaling out.

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

> ⚠ **Note**: You may 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 may 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.

Exercise 5: Implementing vertical scaling of Azure VM Scale Sets

> ? The main tasks for this exercise are as follows:
>
> 1. caling compute resources of Azure virtual machine scale set instances.
> 2. Scaling storage resources of Azure virtual machine scale sets instances.

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 may need to refresh the **az30305c-vmss | Instances** blade.

- [ ] 4. Wait until the instances are updated and running.

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 disks. 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 **D:\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 may need to refresh the **az30305c-vmss | Instances** blade.

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 "az30305c-labRG-VMJ9JCU1OI".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 "az30305c-labRG-VMJ9JCU1OI".name --output tsv | xargs -L1 bash -c 'az group delete --name $0 --no-wait --yes'
```

- [ ] 3. Close the Cloud Shell pane.

> ✓ **Congratulations**. You have now completed this lab,