# Contents

## What are we doing?

- We are publishing the lab instructions and lab files on GitHub to allow for interaction between the course authors and MCTs. We hope this will help keep the content current as the Azure platform changes.
- There is a GitHub repository for the AZ-300, Microsoft Azure Architect Technologies, and AZ-301, Microsoft Azure Architect Design, courses.

- Within each repository there are lab guides in PDF format. If appropriate, there are accompanying zipped files with any additional files that are needed to complete the lab. Not every course has a zipped file.
- For each delivery, trainers should download the latest files from GitHub. Trainers should also check the Issues tab to see if other MCTs have reported any errors.
- Lab timing estimates are provided but trainers should check to ensure this is accurate based on the audience.
- The lab content has been placed at the end of each course for consistency and convenience. However, as the instructor, you are the best judge to determine when the lab should be offered.
- To conduct you will need an internet connection and an Azure subscription. Please read the Instructor Prep Guide for more information.
- It is recommended that you provide these materials directly to your students rather than point them to the GitHub repository.

**How are we doing?**

- If as you are teaching these courses, you identify areas for improvement, please use the Issues tab to provide feedback. We will periodically create new files to incorporate the changes.

We hope using this GitHub repository brings a sense of collaboration to the labs and improves the overall quality of the lab experience.

Regards, Azure Architect Courseware Team # Managing Azure Subscriptions and Resources

# 1 Lab: Exploring Monitoring Capabilities in Azure

#### 1.0.1 Scenario

Adatum Corporation wants to explore monitoring capabilities in Azure

#### 1.0.2 Objectives

After completing this lab, you will be able to:

- Deploy Azure VM scale sets
- Implement monitoring and alerting by using Azure Monitor

#### 1.0.3 Lab Setup

Estimated Time: 45 minutes

User Name: **Student**

Password: **Pa55w.rd**

# 2 Exercise 1: Deploy Azure VM scale sets

The main tasks for this exercise are as follows:

1. Deploy an Azure VM scale set by using an Azure QuickStart template
2. Review autoscaling settings of the Azure VM scale set

##### 2.0.0.1 Task 1: Deploy an Azure VM scale set by using an Azure QuickStart template

1. From the lab virtual machine, start Microsoft Edge and browse to the Azure portal at http://portal.azure.com and sign in by using the Microsoft account that has the Owner role in the target Azure subscription.
2. In the Azure portal, in the Microsoft Edge window, start a **PowerShell** session within the Cloud Shell.
3. If you are presented with the **You have no storage mounted** message, click **Show Advanced Settings** and then configure storage using the following settings:
    - Subscription: the name of the target Azure subscription

- Cloud Shell region: the name of the Azure region that is available in your subscription and which is closest to the lab location
- Resource group: the name of a new resource group **az3000100-LabRG**
- Storage account: a name of a new storage account (between 3 and 24 characters consisting of lower case letters and digits)
- File share: a name of a new file share: **cloudshell**

4. From the Cloud Shell pane, run the following command to identify a unique DNS domain name (substitute the placeholder `<custom-label>` with any alphanumeric string starting with a letter and no longer than 9 characters, which is likely to be unique and the placeholder `<location>` with the name of the Azure region into which you intend to deploy resources in this lab):

   ```
   Test-AzDnsAvailability -DomainNameLabel <custom-label> -Location '<location>'
   ```

5. Verify that the command returned **True**. If not, rerun the same command with a different value of the `<custom-label>` until the command returns **True**.
6. Note the value of the `<custom-label>` that resulted in the successful outcome. You will need it in the next task.
7. On the lab computer, in the Azure portal, search for and select **Template deployment (deploy using custom template)**.
8. On the **Custom deployment** blade, in the **Select a template (disclaimer)** drop-down list, type **201-vmss-bottle-autoscale** and click **Select template**.
9. In the Azure Portal, on the **Deploy VM Scale Set with Python Bottle server & AutoScale** blade, specify the following settings and initiate the deployment:
   - Subscription: the name of the target Azure subscription
   - Resource group: the name of a new resource group **az3000101-LabRG**
   - Location: the name of the Azure region that you referenced when running `Test-AzDnsAvailability` earlier in this task
   - Vm Sku: **Standard_D2s_v3**
   - Vmss Name: the custom label you identified when running `Test-AzDnsAvailability` earlier in this task
   - Instance count: **1**
   - Admin Username: **student**
   - Authentication Type: **password**
   - Admin Password: **Pa55w.rd1234**
10. Place a checkmark next to **I agree to the terms and conditions stated above**, and then click **Purchase**.
11. Wait for the deployment to complete. This will take about 5 minutes.

#### 2.0.0.2 Task 2: Review autoscaling settings of the Azure VM scale set

1. In Azure Portal, navigate to the blade representing the newly deployed **Azure VM scale set**.
2. From the VM scale set blade, navigate to the its **Scaling** blade.
3. Note that the Azure VM scale set is configured to scale dynamically based on a metric using the following criteria:
   - Scale out: increase instance count by 1 when average percentage of CPU > 40
   - Scale in: decrease instance count by 1 when average percentage of CPU < 20
   - Minimum number of instances: 1
   - Maximum number of instances: 10
4. Modify the maximum number of instances to 3 and **save** your changes.

**Result**: After you completed this exercise, you have deployed an Azure VM scale set and reviewed its autoscaling settings.

## 2.1 Exercise 2: Implementing monitoring and alerting by using Azure Monitor

The main tasks for this exercise are as follows:

1. Create Azure VM scale set metrics-based alerts
2. Configure Azure VM scale set autoscaling-based notifications
3. Test Azure VM scale set monitoring and alerting.

#### 2.1.0.1 Task 1: Create Azure VM scale set metrics-based alerts

1. In the Azure portal, navigate to the blade representing the newly deployed Azure VM scale set and, from there, switch to the **Monitoring - Metrics** blade.
2. On the **Monitoring - Metrics** blade, use the filter to display **Percentage CPU** metric with an aggregation value of **Avg** of the VM scale set resource you provisioned in the previous exercise of this lab.
3. Review the resulting chart and note the average percentage CPU within the last few minutes.
4. Navigate to the **Alerts** blade in the **Monitoring** section.
5. From the **Alerts** blade, navigate to the **Manage actions** blade.
6. In the **Manage actions** section, click **Add action group**, set the action group name to **az30001 action group**, set short name: **az30001**, select the Azure subscription you used in the previous exercise, accept the default name of the resource group, then in the **Notifications** pane, set **Name** to **az30001-email**, and set the **Notification type** to **Email/SMS message/Push/Voice**.
7. On the **Email/SMS/Push/Voice** blade, set an email address, a mobile phone number, or a phone number that you want to use to receive alerts generated by this rule and click **OK**.
8. On the **Create action group** page, click **Review + Create**, and then click **Create**.
9. Navigate to the **Alerts** blade.
10. From the **Alerts** blade, navigate to the **New alert rule** blade.
11. In the **Resource** section, select the VM scale set you provisioned in the previous exercise of this lab.
12. In the **Condition** section, click **Add condition**, select the **Percentage CPU** metric, leave the dimension settings and condition type with their default values, set the condition to **Greater than**, set the type aggregation to **Average**, set the threshold to **40**, set the Aggregation granularity (period) to **1 minute**, set the frequency to **Every 1 minute** and click **done**.
13. In the **Actions** section, click **Select action group**, select previously created action group **az30001 action group** and click **done**.
14. In the **Alert Details** section, set the alert rule name to **Percentage CPU of the VM scale set is greater than 60 percent**, its description to **Percentage CPU of the VM scale set is greater than 40 percent**, its severity to **Sev 3**, and set enable rule upon creation to **Yes**.
15. Click **Create alert rule**.

    **Note**: It can take up to 10 minutes for a metric alert rule to become active.

#### 2.1.0.2 Task 2: Configure Azure VM scale set autoscaling-based notifications

1. In the Azure portal, navigate to the blade representing the newly deployed Azure VM scale set and, from there, switch to the **Scaling** blade.
2. On the **Scaling** blade, click the **Notify** tab heading, configure the following settings, and **save** your changes:
    - Email administrators: enabled
    - Email co-administrators: disabled

- Additional administrator emails(s): add an email address that you want to use to receive notifications about autoscaling events

#### 2.1.0.3 Task 3: Test Azure VM scale set monitoring and alerting.

1. In the Azure portal, search for **Load balancers** and navigate to the Load Balancers blade. A load balancer should exist, representing the one created with the scale set you deployed from a template in a previous exercise of this lab.
2. Identify the value of the **Public IP address** assigned to the front end of the load balancer associated with the VM scale set.
3. From the lab computer, start Microsoft Edge and browse to **http://*Public IP address*:9000** (where ***Public IP address*** is the IP address you identified in the previous step)
4. On the **Worker interface** page, click the **Start work** link.
5. Use the **CPU (average)** chart on the VM scale set Overview blade to monitor changes to the CPU utilization.

   > **Note**: Alternatively, you can navigate back to the **Metrics** blade and use the filter to display **Percentage CPU** metric of the VM scale set resource, and set the time to **Last 30 minutes**.

   > **Note**: You should receive an alert regarding increased CPU utilization within a couple of minutes

6. Switch to the **Instances** blade of the VM scale set in order to identify the number of its instances.

   > **Note**: Alternatively, you can navigate back to the **Scaling** blade, in the list of resources capable of autoscaling, click the name of the VM scale set, on the **Autoscale settings** blade, click **Run history**, and then review the list of autoscale events.

   > **Note**: Autoscaling should be triggered within a couple of minutes.

7. Switch to the Microsoft Edge window displaying worker instances page and click the **Stop work** link.
8. Monitor decrease in CPU utilization and scaling in events using the same methods that you used when scaling out the VM scale set.

   **Result**: After you completed this exercise, you have implemented and tested monitoring and alerting by using Azure Monitor.

## 2.2 Exercise 3: Remove lab resources

The main tasks for this exercise are as follows:

1. Discover resource groups created in this lab
2. Delete resource groups created in this lab

### 2.2.1 Task 1: Discover resource groups created in this lab

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell panel and switch to the **PowerShell** shell if necessary.
2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all resource groups you created in this lab:

   ```
   Get-AzResourceGroup -Name 'az300010*'
   ```

3. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

#### 2.2.1.1 Task 2: Delete resource groups created in this lab

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

   ```
   Get-AzResourceGroup -Name 'az300010*' | Remove-AzResourceGroup -Force -AsJob
   ```

> **Note**: The command executes asynchronously (as determined by the -AsJob parameter), so while you will be able to run another PowerShell command immediately afterwards within the same PowerShell session, it will take a few minutes before the resource groups are actually removed.

2. Close the **Cloud Shell** prompt at the bottom of the portal.

   **Result**: After you completed this exercise, you removed the resources used in this lab. # Deploying and Managing Virtual Machines (VMs)

# 3 Lab: Implementing custom Azure VM images

#### 3.0.1 Scenario

Adatum Corporation wants to create custom Azure VM images

#### 3.0.2 Objectives

After completing this lab, you will be able to:

- Create a custom VM image using HashiCorp Packer
- Deploy an Azure VM based on a custom image

#### 3.0.3 Lab Setup

Estimated Time: 45 minutes

Interface: **Use Azure Cloud Shell in BASH mode**

## 3.1 Exercise 1: Creating a custom image

The main tasks for this exercise are as follows:

1. Configure a Packer template
2. Build a Packer-based image

##### 3.1.0.1 Task 1: Configure a Packer template

1. From the lab virtual machine, start Microsoft Edge and browse to the Azure portal at **http://portal.azure.com** and sign in by using the Microsoft account that has the Owner role in the target Azure subscription.
2. In the Azure portal, in the Microsoft Edge window, start a **Bash** session within the **Cloud Shell**.
3. If you are presented with the **You have no storage mounted** message, configure storage using the following settings:
   - Subsciption: the name of the target Azure subscription
   - Cloud Shell region: the name of the Azure region that is available in your subscription and which is closest to the lab location
   - Resource group: **az3000300-LabRG**
   - Storage account: a name of a new storage account
   - File share: a name of a new file share
4. From the Cloud Shell pane, run the following to create a resource group and store the JSON output in a variable (replace the `<Azure region>` placeholder with the name of the Azure region that is available in your subscription and which is closest to the lab location):

   ```
   RG=$(az group create --name az3000301-LabRG --location <Azure region>)
   ```

   > **Note**: To list Azure regions, run `az account list-locations --output table`

5. From the Cloud Shell pane, run the following to create a service principal that will be used by Packer and store the JSON output in a variable:

   ```
   AAD_SP=$(az ad sp create-for-rbac)
   ```

6. From the Cloud Shell pane, run the following to retrieve the value of the service principal appId and store it in a variable

   ```
   CLIENT_ID=$(echo $AAD_SP | jq -r .appId)
   ```

7. From the Cloud Shell pane, run the following to retrieve the value of the service principal password and store it in a variable

   ```
   CLIENT_SECRET=$(echo $AAD_SP | jq -r .password)
   ```

8. From the Cloud Shell pane, run the following to retrieve the value of the service principal tenant ID and store it in a variable

   ```
   TENANT_ID=$(echo $AAD_SP | jq -r .tenant)
   ```

9. From the Cloud Shell pane, run the following to retrieve the value of the subscription ID and store it in a variable:

   ```
   SUBSCRIPTION_ID=$(az account show --query id | tr -d '"')
   ```

10. From the Cloud Shell pane, run the following to retrive the value of the resource group location and store it in a variable:

    ```
    LOCATION=$(echo $RG | jq -r .location)
    ```

11. From the Cloud Shell pane, upload the Packer template **\allfiles\AZ-300T01\Module_03\template03.json** into the home directory. To upload a file, click the document icon that has an up and down arrow in the Cloud Shell pane.

12. From the Cloud Shell pane, run the following to replace the placeholder for the value of the **client_id** parameter with the value of the **$CLIENT_ID** variable in the Packer template:

    ```
    sed -i.bak1 's/"$CLIENT_ID"/"'"$CLIENT_ID"'"/' ~/template03.json
    ```

13. From the Cloud Shell pane, run the following to replace the placeholder for the value of the **client_secret** parameter with the value of the **$CLIENT_SECRET** variable in the Packer template:

    ```
    sed -i.bak2 's/"$CLIENT_SECRET"/"'"$CLIENT_SECRET"'"/' ~/template03.json
    ```

14. From the Cloud Shell pane, run the following to replace the placeholder for the value of the **tenant_id** parameter with the value of the **$TENANT_ID** variable in the Packer template:

    ```
    sed -i.bak3 's/"$TENANT_ID"/"'"$TENANT_ID"'"/' ~/template03.json
    ```

15. From the Cloud Shell pane, run the following to replace the placeholder for the value of the **subscription_id** parameter with the value of the **$SUBSCRIPTION_ID** variable in the Packer template:

    ```
    sed -i.bak4 's/"$SUBSCRIPTION_ID"/"'"$SUBSCRIPTION_ID"'"/' ~/template03.json
    ```

16. From the Cloud Shell pane, run the following to replace the placeholder for the value of the **location** parameter with the value of the **$LOCATION** variable in the Packer template:

    ```
    sed -i.bak5 's/"$LOCATION"/"'"$LOCATION"'"/' ~/template03.json
    ```

#### 3.1.0.2 Task 2: Build a Packer-based image

1. From the Cloud Shell pane, run the following to build the packer-based image:

   ```
   packer build template03.json
   ```

2. Monitor the built progress until it completes.

   > **Note**: The build process might take about 10 minutes.

   **Result**: After you completed this exercise, you have created a Packer template and used it to build a custom image.

## 3.2 Exercise 2: Deploying a custom image

The main tasks for this exercise are as follows:

1. Deploy an Azure VM based on a custom image
2. Validate Azure VM deployment

#### 3.2.0.1 Task 1: Deploy an Azure VM based on a custom image

1. From the Cloud Shell pane, run the following to deploy an Azure VM based on the custom image.

   ```
   az vm create --resource-group az3000301-LabRG --name az3000301-vm --image az3000301-image --admin-u
   ```

2. Wait for the deployment to complete

   > **Note**: The deployment process might take about 3 minutes.

3. Once the deployment completes, from the Cloud Shell pane, run the following to allow inbound traffic to the newly deployed VM on TCP port 80:

   ```
   az vm open-port --resource-group az3000301-LabRG --name az3000301-vm --port 80
   ```

#### 3.2.0.2 Task 2: Validate Azure VM deployment

1. From the Cloud Shell pane, run the following to identify the IP address associated with the newly deployed Azure VM.

   ```
   az network public-ip show --resource-group az3000301-LabRG --name az3000301-vmPublicIP --query ipA
   ```

2. Start Microsoft Edge and navigate to the IP address you identified in the previous step.
3. Verify that Microsoft Edge displays the **Welcome to nginx!** page.

   **Result**: After you completed this exercise, you have deployed an Azure VM based on a custom image and validated the deployment.

## 3.3 Exercise 3: Remove lab resources

#### 3.3.0.1 Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all resource groups you created in this lab:

   ```
   az group list --query "[?starts_with(name,'az30003')]".name --output tsv
   ```

3. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

#### 3.3.0.2 Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

   ```
   az group list --query "[?starts_with(name,'az30003')]".name --output tsv | xargs -L1 bash -c 'az g
   ```

2. Close the **Cloud Shell** prompt at the bottom of the portal.

   **Result**: In this exercise, you removed the resources used in this lab. # Managing Identities

# 4 Lab: Implementing user-assigned managed identities for Azure resources

### 4.0.1 Scenario

Adatum Corporation wants to use managed identities to authenticate applications running in Azure VMs

### 4.0.2 Objectives

After completing this lab, you will be able to:

- Create and configure user-assigned managed identities
- Validate functionality of user-assigned managed identities

#### 4.0.3 Lab Setup

Estimated Time: 30 minutes

User Name: **Student**

Password: **Pa55w.rd**

## 4.1 Exercise 1: Creating and configuring a user-assigned managed identity.

The main tasks for this exercise are as follows:

1. Deploy an Azure VM running Windows Server 2016 Datacenter
2. Create a user-assigned managed identity.
3. Assign the user-assigned managed identity to the Azure VM.
4. Grant RBAC-based permissions to the user-assigned managed identity.

##### 4.1.0.1 Task 1: Deploy an Azure VM running Windows Server 2016 Datacenter

1. From the lab virtual machine, start Microsoft Edge and browse to the Azure portal at **http://portal.azure.com** and sign in by using the Microsoft account that has the Owner role in the target Azure subscription.
2. In the Azure portal, in the Microsoft Edge window, start a **Bash** session within the **Cloud Shell**.
3. If you are presented with the **You have no storage mounted** message, configure storage using the following settings:
   - Subscription: the name of the target Azure subscription
   - Cloud Shell region: the name of the Azure region that is available in your subscription and which is closest to the lab location
   - Resource group: **az3000500-LabRG**
   - Storage account: a name of a new storage account
   - File share: a name of a new file share
4. From the Cloud Shell pane, create a resource group by running (replace the `<Azure region>` placeholder with the name of the Azure region that is available in your subscription and which is closest to the lab location)

   ```
   az group create --resource-group az3000501-LabRG --location <Azure region>
   ```

5. From the Cloud Shell pane, upload the Azure Resource Manager template **\allfiles\AZ-300T01\Module_05\azured** into the home directory.
6. From the Cloud Shell pane, upload the parameter file **\allfiles\AZ-300T01\Module_05\azuredeploy05.paramete** into the home directory.
7. From the Cloud Shell pane, deploy an Azure VM hosting Windows Server 2016 Datacenter into the first virtual network by running:

   ```
   az deployment group create --resource-group az3000501-LabRG --template-file azuredeploy05.json --pa
   ```

   > **Note**: Wait for the deployment to complete. This might take about 5 minutes.

##### 4.1.0.2 Task 2: Create a user-assigned managed identity and assign it to the Azure VM.

1. From the Cloud Shell pane, run the following to create a user-assigned managed identity:

   ```
   az identity create --resource-group az3000501-LabRG --name az3000501-mi
   ```

2. From the Cloud Shell pane, run the following to assign the user-assigned managed identity to the Azure VM:

   ```
   az vm identity assign --resource-group az3000501-LabRG --name az3000501-vm --identities az3000501-r
   ```

#### 4.1.0.3 Task 3: Configure RBAC referencing the user-assigned managed identity.

1. From the Cloud Shell pane, run the following to create a resource group (replace the `<Azure region>` placeholder with the name of the Azure region into which you deployed the Azure VM in this exercise):

   ```
   az group create --resource-group az3000502-LabRG --location <Azure region>
   ```

2. In the Azure portal, navigate to the **az3000502-LabRG - Access control (IAM)** blade.

3. From the **az3000502-LabRG - Access control (IAM)** blade, assign the Owner role to the newly created user-assigned managed identity.

   **Result**: After you completed this exercise, you have created and configured a user-assigned managed identity.

## 4.2 Exercise 2: Validating functionality of user-assigned managed identities

The main tasks for this exercise are as follows:

1. Configure an Azure VM for authenticating via user-assigned managed identity.
2. Validate functionality of user-assigned managed identity from the Azure VM.

#### 4.2.0.1 Task 1: Configure an Azure VM for authenticating via user-assigned managed identity.

1. In the Azure portal, navigate to the **az3000501-vm** blade.

2. Connect to the Azure VM by using Remote Desktop and authenticate by providing the following credentials:

   - Username: **Student**
   - Password: **Pa55w.rd1234**

3. Once you establish a Remote Desktop session, you will be presented with an **Administrator: C:\Windows\system32\cmd.exe** window. To start a PowerShell session, at the command prompt, type `PowerShell` and press Enter.

4. From the PowerShell prompt, run the following to install the latest version of the PowerShellGet module (press Enter if prompted for confirmation):

   ```
   [Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
   Install-Module -Name PowerShellGet -Force -SkipPublisherCheck
   ```

5. From the PowerShell prompt, run the following to install the latest version of the Az module (type **Y** and press Enter when prompted for confirmation):

   ```
   Install-Module -Name Az -AllowClobber -SkipPublisherCheck
   ```

6. Exit the current PowerShell session by typing `exit` and pressing Enter and then start it again by typing at the command prompt `PowerShell` and pressing Enter.

7. From the PowerShell prompt, run the following to install the AzureRM.ManagedServiceIdentity module:

   ```
   [Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
   ```

8. From the PowerShell prompt, run the following to install the AzureRM.ManagedServiceIdentity module (type **Y** and press Enter when prompted for confirmation):

   ```
   Install-Module -Name Az.ManagedServiceIdentity
   ```

#### 4.2.0.2 Task 2: Validate functionality of user-assigned managed identity from the Azure VM.

1. From the PowerShell prompt, run the following to sign-in as the user-assigned managed identity:

   ```
   Add-AzAccount -Identity
   ```

2. From the PowerShell prompt, run the following to attempt to retrieve the currently used managed identity:

   ```
   (Get-AzVM -ResourceGroupName az3000501-LabRG -Name az3000501-vm).Identity
   ```

3. Note the error message. As the message states, the current security context does not grant sufficent authorization to the target resource. To resolve this issue, switch to the Azure portal, navigate to the **az3000501-LabRG - Access control (IAM)** blade.
4. From the **az3000501-LabRG - Access control (IAM)** blade, assign the Contributor role to the user-assigned managed identity **az3000501-mi**.
5. Switch back to the Remote Desktop session, and, from the PowerShell prompt, run the following to attempt to retrieve the currently used managed identity:

   ```
   (Get-AzVM -ResourceGroupName az3000501-LabRG -Name az3000501-vm).Identity
   ```

   > **Note**: If you receive an error message indicating insufficient privileges, from the PowerShell prompt, run

   ```
   Remove-AzAccount
   ```

   followed by:

   ```
   Add-AzAccount -Identity
   ```

6. From the PowerShell prompt, run the following to store location in a variable:

   ```
   $location = (Get-AzResourceGroup -Name az3000502-LabRG).Location
   ```

7. From the PowerShell prompt, run the following to create a public IP address resource:

   ```
   New-AzPublicIpAddress -Name az3000502-pip -ResourceGroupName az3000502-LabRG -AllocationMethod Dyn
   ```

8. Verify that the command completed successfully.

   **Result**: After you completed this exercise, you have validated the functionality of the user-defined managed identity.

## 4.3 Exercise 3: Remove lab resources

#### 4.3.0.1 Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all resource groups you created in this lab:

   ```
   az group list --query "[?starts_with(name,'az30005')].name" --output tsv
   ```

3. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

#### 4.3.0.2 Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

   ```
   az group list --query "[?starts_with(name,'az30005')].name" --output tsv | xargs -L1 bash -c 'az g
   ```

2. Close the **Cloud Shell** prompt at the bottom of the portal.

   **Result**: In this exercise, you removed the resources used in this lab. # Evaluating and Performing Server Migration to Azure

# 5 Lab: Implementing Azure to Azure migration

### 5.0.1 Scenario

Adatum Corporation wants to migrate their existing Azure VMs to another region

### 5.0.2 Objectives

After completing this lab, you will be able to:

- Implement Azure Site Recovery Vault
- Migrate Azure VMs between Azure regions

### 5.0.3 Lab Setup

Estimated Time: 45 minutes

User Name: **Student**

Password: **Pa55w.rd**

## 5.1 Exercise 1: Implement prerequisites for migration of Azure VMs by using Azure Site Recovery

The main tasks for this exercise are as follows:

1. Deploy an Azure VM to be migrated
2. Create an Azure Recovery Services vault

##### 5.1.0.1 Task 1: Deploy an Azure VM to be migrated

1. From the lab virtual machine, start Microsoft Edge and browse to the Azure portal at **http://portal.azure.com** and sign in by using the Microsoft account that has the Owner role in the target Azure subscription.
2. In the Azure portal, in the Microsoft Edge window, start a **PowerShell** session within the **Cloud Shell**.
3. If you are presented with the **You have no storage mounted** message, configure storage using the following settings:
    - Subsciption: the name of the target Azure subscription
    - Cloud Shell region: the name of the Azure region that is available in your subscription and which is closest to the lab location
    - Resource group: the name of a new resource group **az3000600-LabRG**
    - Storage account: a name of a new storage account
    - File share: a name of a new file share
4. From the Cloud Shell pane, create a resource group by running (replace the `<Azure region>` placeholder with the name of the Azure region that is available in your subscription and which is closest to the lab location)

    ```
    New-AzResourceGroup -Name az3000601-LabRG -Location <Azure region>
    ```

5. From the Cloud Shell pane, upload the Azure Resource Manager template **\allfiles\AZ-300T02\Module_01\azurec** into the home directory.
6. From the Cloud Shell pane, upload the parameter file **\allfiles\AZ-300T02\Module_01\azuredeploy06.paramete** into the home directory.
7. From the Cloud Shell pane, switch to your home directory :

    ```
    cd $home
    ```

8. From the Cloud Shell pane, deploy an Azure VM hosting Windows Server 2016 Datacenter by running:

    ```
    New-AzResourceGroupDeployment -ResourceGroupName az3000601-LabRG -TemplateFile azuredeploy06.json
    ```

    > **Note**: Do not wait for the deployment to complete but instead proceed to the next task.

#### 5.1.0.2 Task 2: Implement an Azure Recovery Service vault

1. From Azure Portal, create an instance of **Recovery Services vault** with the following settings:
   - Name: **vaultaz3000602**
   - Subscription: the name of the target Azure subscription
   - Resource group: the name of a new resource group **az3000602-LabRG**
   - Location: the name of an Azure region that is available in your subscription and which is **different** from the region you deployed an Azure VM in the previous task
2. Wait until the vault is provisioned. This will take about a minute.

   **Result**: After you completed this exercise, you have created an Azure VM to be migrated and an Azure Site Recovery vault that will host the migrated disk files of the Azure VM.

## 5.2 Exercise 2: Migrate an Azure VM between Azure regions by using Azure Site Recovery

The main tasks for this exercise are as follows:

1. Configure Azure VM replication
2. Review Azure VM replication settings
3. Disable replication of an Azure VM and delete the Azure Recovery Services vault

#### 5.2.0.1 Task 1: Configure Azure VM replication

1. In the the Azure portal, navigate to the blade of the newly provisioned Azure Recovery Services vault.
2. On the Recovery Services vault blade, click the **+ Replicate** button.
3. Enable replication by specifying the following settings:
   - Source: **Azure**
   - Source location: the same Azure region into which you deployed the Azure VM in the previous exercise of this lab
   - Azure virtual machine deployment model: **Resource Manager**
   - Source resource group: **az3000601-LabRG**
   - Virtual machines: **az3000601-vm**
   - Target location: the name of an Azure region that is available in your subscription and which is different from the region you deployed an Azure VM in the previous task
   - Target resource group: **(new) az3000601-LabRG-asr**
   - Target virtual network: **(new) az3000601-vnet-asr**
   - Cache storage account: accept the default setting
   - Replica managed disks: **(new) 1 premium disk(s), 0 standard disk(s)**
   - Target availability sets: **Not Applicable**
   - Replication policy: **Create new**
   - Name: **12-hour-retention-policy**
   - Recovery point retention: **12 Hours**
   - App consistent snapshot frequency: **6 Hours**
   - Multi-VM consistency: **No**
4. Initiate creation of target resources.
5. Enable the replication.

   > **Note**: Wait for the operation of enabling the replication to complete. Then proceed to the next task.

#### 5.2.0.2 Task 2: Review Azure VM replication settings

1. In the Azure portal, from the Azure Site Recovery vault blade, navigate to the replicated item blade representing the Azure VM **az3000601-vm**.
2. On the replicated item blade, review the **Health and status**, **Latest available recovery points**, and **Failover readiness** sections. Note the **Failover** and **Test Failover** entries in the toolbar. Scroll down to the **Infrastructure view**.
3. If time permits, wait until the status of the Azure VM changes to **Protected**. This might take additional 15-20 minutes. At that point, examine the values **Crash-consistent** and **App-consistent** recovery points. In order to view **RPO**, you should perform a test failover.

#### 5.2.0.3 Task 3: Disable replication of an Azure VM and delete the Azure Recovery Services vault

1. In the Azure portal, disable replication of the Azure VM **az3000601-vm**.
2. Wait until the replication is disabled.
3. From the Azure portal, delete the Recovery Services vault.

    > **Note**: You must ensure that the replicated item is removed first before you can delete the vault.

    **Result**: After you completed this exercise, you have implemented automatic replication of an Azure VM.

## 5.3 Exercise 3: Remove lab resources

#### 5.3.0.1 Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. If needed, switch to the Bash shell session by using the drop down list in the upper left corner of the Cloud Shell pane.
3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all resource groups you created in this lab:

    ```
    az group list --query "[?starts_with(name,'az30006')].name" --output tsv
    ```

4. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

#### 5.3.0.2 Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

    ```
    az group list --query "[?starts_with(name,'az30006')].name" --output tsv | xargs -L1 bash -c 'az g
    ```

2. Close the **Cloud Shell** prompt at the bottom of the portal.

    **Result**: In this exercise, you removed the resources used in this lab. # Configuring and Managing Virtual Networks

# 6 Lab: Configuring VNet peering and service chaining

### 6.0.1 Scenario

ADatum Corporation wants to implement service chaining between Azure virtual networks in its Azure subscription.

### 6.0.2 Objectives

After completing this lab, you will be able to:

- Deploy Azure VMs by using Azure Resource Manager templates.
- Configure VNet peering.

- Implement routing
- Validate service chaining

#### 6.0.3 Lab Setup

Estimated Time: 45 minutes

User Name: **Student**

Password: **Pa55w.rd**

## 6.1 Exercise 1: Creating an Azure lab environment by using deployment templates

The main tasks for this exercise are as follows:

1. Create the first Azure virtual network environment by using an Azure Resource Manager template
2. Create the second Azure virtual network environment by using an Azure Resource Manager template

##### 6.1.0.1 Task 1: Create the first Azure virtual network environment by using an Azure Resource Manager template

1. From the lab virtual machine, start Microsoft Edge and browse to the Azure portal at **http://portal.azure.com** and sign in by using the Microsoft account that has the Owner role in the target Azure subscription.
2. In the Azure portal, in the Microsoft Edge window, start a **Bash** session within the **Cloud Shell**.
3. If you are presented with the **You have no storage mounted** message, configure storage using the following settings:
   - Subsciption: the name of the target Azure subscription
   - Cloud Shell region: the name of the Azure region that is available in your subscription and which is closest to the lab location
   - Resource group: the name of a new resource group **az3000400-LabRG**
   - Storage account: a name of a new storage account
   - File share: a name of a new file share
4. From the Cloud Shell pane, create two resource groups by running (replace the `<Azure region>` placeholder with the name of the Azure region that is available in your subscription and which is closest to the lab location)

   ```
   az group create --resource-group az3000401-LabRG --location <Azure region>
   ```

   ```
   az group create --resource-group az3000402-LabRG --location <Azure region>
   ```

5. From the Cloud Shell pane, upload the first Azure Resource Manager template **\allfiles\AZ-300T02\Module_03\azuredeploy0401.json** into the home directory.
6. From the Cloud Shell pane, upload the parameter file **\allfiles\AZ-300T02\Module_03\azuredeploy04.paramete** into the home directory.
7. From the Cloud Shell pane, deploy the two Azure VMs hosting Windows Server 2016 Datacenter into the first virtual network by running:

   ```
   az deployment group create --resource-group az3000401-LabRG --template-file azuredeploy0401.json -
   ```

   > **Note**: Do not wait for the deployment to complete but proceed to the next task.

##### 6.1.0.2 Task 2: Create the second Azure virtual network environment by using an Azure Resource Manager template

1. From the Cloud Shell pane, upload the second Azure Resource Manager template **\allfiles\AZ-300T02\Module_03\azuredeploy0402.json** into the home directory.
2. From the Cloud Shell pane, deploy an Azure VM hosting Windows Server 2016 Datacenter into the second virtual network by running:

```
az deployment group create --resource-group az3000402-LabRG --template-file azuredeploy0402.json -
```

**Note**: The second template uses the same parameter file.

**Note**: Do not wait for the deployment to complete but proceed to the next exercise.

**Result**: After completing this exercise, you should have created two Azure virtual networks hosting Azure VMs running Windows Server 2016 Datacenter.

## 6.2 Exercise 2: Configuring VNet peering

The task for this exercise is as follows:

1. Configure VNet peering for both virtual networks

#### 6.2.0.1 Task 1: Configure VNet peering for both virtual networks

1. In the Microsoft Edge window displaying the Azure portal, navigate to the **az3000401-vnet** virtual network blade.
2. From the **az3000401-vnet** blade, create a VNet peering with the following settings:
   - Name of the peering from the first virtual network to the second virtual network: **az3000401-vnet-to-az3000402-vnet**
   - Virtual network deployment model: **Resource manager**
   - Subscription: the name of the Azure subscription you are using for this lab
   - Virtual network: **az3000402-vnet**
   - Name of the peering from the second virtual network to the first virtual network: **az3000402-vnet-to-az3000401-vnet**
   - Allow virtual network access from the first virtual network to the second virtual nework: **Enabled**
   - Allow virtual network access from the second virtual network to the first virtual nework: **Enabled**
   - Allow forwarded traffic from the first virtual network to the second virtual network: **Disabled**
   - Allow forwarded traffic from the second virtual network to the first virtual network: **Disabled**
   - Allow gateway transit: disabled

## 6.3 Exercise 3: Implementing routing

The main tasks for this exercise are as follows:

1. Enable IP forwarding
2. Configure user defined routing
3. Configure routing on an Azure VM running Windows Server 2016

#### 6.3.0.1 Task 1: Enable IP forwarding

1. In Microsoft Edge, navigate to the **az3000401-nic2** blade (the NIC of **az3000401-vm2**)
2. On the **az3000401-nic2** blade, modify the **IP configurations** by setting **IP forwarding** to **Enabled**.

#### 6.3.0.2 Task 2: Configure user defined routing

1. In the Azure portal, create a new route table with the following settings:
   - Name: **az3000402-rt1**
   - Subscription: the name of the Azure subscription you use for this lab
   - Resource group: **az3000402-LabRG**
   - Location: the same Azure region in which you created the virtual networks
   - Virtual network gateway route propagation: **Disabled**

Once the creation of the route table has finished, click on **Go to resource**

2. In the Azure portal, on the route table az3000402-rt1 that was created on the previous step, click on **Routes** under **Settings** and add a route with the following settings:
   - Route name: **custom-route-to-az3000401-vnet**
   - Address prefix: **10.0.0.0/22**
   - Next hop type: **Virtual appliance**
   - Next hop address: **10.0.1.4**
3. In the Azure portal, associate the route table with the **subnet-1** of the **az3000402-vnet**.

#### 6.3.0.3 Task 3: Configure routing on an Azure VM running Windows Server 2016

1. On the lab computer, from the Azure portal, start a Remote Desktop session to **az3000401-vm2** Azure VM.
2. When prompted to authenticate, specify the following credentials:
   - User name: **Student**
   - Password: **Pa55w.rd1234**
3. Once you are connected to az3000401-vm2 via the Remote Desktop session, install the **Remote Access** role. In Server Manager, click **Manage**, and then click **Add roles and features**.
4. On the Before you Begin screen, click **Next**.
5. On the Installation Type screen, leave the default **Role-based or feature-based installation** selected and click **Next**.
6. On the Server Selection screen, leave the default server selected and click **Next**.
7. On the Server Roles screen, place a checkmark next to **Remote Access** and then click **Next**.
8. On the Features screen, leave the defaults selected and then click **Next**.
9. On the Remote Acces screen, click **Next**.
10. On the Services screen, place a checkmark next to **Routing** and then click **Next**.
11. On the Confirmation screen, click **Next**.
12. On the Results screen, click **Install**.
13. In the Remote Desktop session to az3000401-vm2, from Server Manager, click **Tools** and then click **Routing and Remote Access** to open the RRAS console.
14. In the **Routing and Remote Access** console, right click under the name of the server az3000401-vm2 and select **Configure and Enable Routing and Remote Access** to run the **Routing and Remote Access Server Setup Wizard**.
15. In the **Routing and Remote Access Server Setup Wizard**, select **Custom configuration** under **Configuration** and enable **LAN routing**.

    **Note**: If you receive a warning pop-up, click **OK**.

1. Start **Routing and Remote Access** service.
2. In the Remote Desktop session to az3000401-vm2, click start and then click **Windows Administrative Tools**.
3. From Administrative Tools, launch the **Windows Firewall with Advanced Security** console.
4. In the console, click **Inbound Rules**.
5. Locate the **File and Printer Sharing (Echo Request - ICMPv4-In)** inbound rule, right-click the rule and click **Enable Rule**.

   **Result**: After completing this exercise, you have configured custom routing within the second virtual network.

## 6.4 Exercise 4: Validating service chaining

The main tasks for this exercise are as follows:

1. Configure Windows Firewall with Advanced Security on an Azure VM
2. Test service chaining between peered virtual networks

#### 6.4.0.1 Task 1: Configure Windows Firewall with Advanced Security on the target Azure VM

1. On the lab computer, from the Azure portal, start a Remote Desktop session to **az3000401-vm1** Azure VM.
2. When prompted to authenticate, specify the following credentials:
   - User name: **Student**
   - Password: **Pa55w.rd1234**
3. In the Remote Desktop session to az3000401-vm1, start the **Windows Firewall with Advanced Security** console and enable **File and Printer Sharing (Echo Request - ICMPv4-In)** inbound rule for all profiles.

#### 6.4.0.2 Task 2: Test service chaining between peered virtual networks

1. On the lab computer, from the Azure portal, start a Remote Desktop session to **az3000402-vm1** Azure VM.
2. When prompted to authenticate, specify the following credentials:
   - User name: **Student**
   - Password: **Pa55w.rd1234**
3. Once you are connected to az3000402-vm1 via the Remote Desktop session, start **Windows PowerShell**.
4. In the **Windows PowerShell** window, run the following:

   ```
   Test-NetConnection -ComputerName 10.0.0.4 -TraceRoute
   ```
5. Verify that test is successful and note that the connection was routed over 10.0.1.4

   **Result**: After completing this exercise, you should have validated service chaining between peered virtual networks.

## 6.5 Exercise 5: Remove lab resources

#### 6.5.0.1 Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. If needed, switch to the Bash shell session by using the drop down list in the upper left corner of the Cloud Shell pane.
3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all resource groups you created in this lab:

   ```
   az group list --query "[?starts_with(name,'az30004')]".name --output tsv
   ```
4. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

#### 6.5.0.2 Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

   ```
   az group list --query "[?starts_with(name,'az30004')]".name --output tsv | xargs -L1 bash -c 'az g
   ```
2. Close the **Cloud Shell** prompt at the bottom of the portal.

   **Result**: In this exercise, you removed the resources used in this lab. # Implementing and Managing Storage

# 7 Lab: Implementing Azure Storage access controls

#### 7.0.1 Scenario

Adatum Corporation wants to protect content residing in Azure Storage

#### 7.0.2 Objectives

After completing this lab, you will be able to:

- Create an Azure Storage account.
- Upload data to Azure Storage.
- Implement Azure Storage access controls

#### 7.0.3 Lab Setup

Estimated Time: 30 minutes

User Name: **Student**

Password: **Pa55w.rd**

## 7.1 Exercise 1: Creating and configuring an Azure Storage account

The main tasks for this exercise are as follows:

1. Create a storage account in Azure
2. View the properties of the storage account

##### 7.1.0.1 Task 1: Create a storage account in Azure

1. From the lab virtual machine, start Microsoft Edge and browse to the Azure portal at **http://portal.azure.com** and sign in by using the Microsoft account that has the Owner role in the target Azure subscription.
2. From Azure Portal, create a new storage account with the following settings:
   - Subscription: the name of the target Azure subscription
   - Resource group: a new resource group named **az3000201-LabRG**
   - Storage account name: any valid, unique name between 3 and 24 characters consisting of lowercase letters and digits
   - Location: the name of the Azure region that is available in your subscription and which is closest to the lab location
   - Performance: **Standard**
   - Account kind: **Storage (general purpose v1)**
   - Replication: **Locally-redundant storage (LRS)**
   - Secure transfer required: **Disabled**
   - Network connectivity: **Public endpoint (All networks)**
   - Blob soft delete: **Disabled**
   - Data Lake Storage Gen2: **Disabled**
3. Wait for the storage account to be provisioned. This will take about a minute.

##### 7.1.0.2 Task 2: View the properties of the storage account

1. In Azure Portal, with your storage account blade open, review the **Overview** section, including the location, replication, and performance settings.
2. Display the **Access keys** blade. On the access keys blade, note that you have the option of copying the values of storage account names including key1 and key2. You also have the ability to regenerate both keys.

3. Display the **Configuration** blade.
4. On the **Configuration** blade, notice that you have the option of performing an upgrade to **General Purpose v2** account and changing the replication settings. However, you cannot change the performance setting (this can only be assigned when the storage account is created).

   **Result**: After you completed this exercise, you have created your Azure Storage and examined its properties.

## 7.2 Exercise 2: Creating and managing blobs

The main tasks for this exercise are as follows:

1. Create a container
2. Upload data to the container by using the Azure portal
3. Access content of Azure Storage account by using a SAS token

#### 7.2.0.1 Task 1: Create a container

1. In the Azure portal, navigate to the blade displaying the properties of the storage account you created in the previous task.
2. From the storage account blade, create a new blob container with the following settings:
   - Name: **labcontainer**
   - Access type: **Private**

#### 7.2.0.2 Task 2: Upload data to the container by using the Azure portal

1. In the Azure portal, navigate to the **labcontainer** blade.
2. From the **labcontainer** blade, upload the file: **C:\Windows\ImmersiveControlPanel\images\splashscreen.con white_scale-400.png**.

#### 7.2.0.3 Task 3: Access content of Azure Storage account by using a SAS token

1. From the **labcontainer** blade, identify the URL of the newly uploaded blob.
2. Start Microsoft Edge and navigate to that URL.
3. Note the **ResourceNotFound** error message. This is expected since the blob is residing in a private container, which requires authenticated access.
4. Switch to the Microsoft Edge window displaying the Azure portal and, on the **splashscreen.contrast-white_scale-400.png** blade, switch to the **Generate SAS** tab.
5. On the **Generate SAS** tab, enable the **HTTP** option and generate blob SAS token and the corresponding URL.
6. Open a new Microsoft Edge window and, in the navigate to the URL generated in the previous step.
7. Note that you can view the image. This is expected since this time you are authorized to access the blob based on the SAS token included in the URL.
8. Close the Microsoft Edge window displaying the image.

#### 7.2.0.4 Task 4: Access content of Azure Storage account by using a SAS token and a stored access policy.

1. In the Azure portal, navigate to the **labcontainer** blade.
2. From the **labcontainer** blade, navigate to the **labcontainer - Access policy** blade.
3. Add a new policy with the following settings:
   - Identifier: **labcontainer-read**
   - Permissions: **Read**
   - Start time: current date and time

- Expiry time: current date and time + 24 hours

4. Click **OK** to create the Policy, and then click **Save**.
5. In the Azure portal, in the Microsoft Edge window, start a **PowerShell** session within the **Cloud Shell**.
6. If you are presented with the **You have no storage mounted** message, configure storage using the following settings:
   - Subsciption: the name of the target Azure subscription
   - Cloud Shell region: the name of the Azure region that is available in your subscription and which is closest to the lab location
   - Resource group: **az3000201-LabRG**
   - Storage account: a name of a new storage account
   - File share: a name of a new file share
7. From the Cloud Shell pane, run the following to identify the storage account resource you created in the first exercise of this lab and store it in a variable:

   ```
   $storageAccount = (Get-AzStorageAccount -ResourceGroupName az3000201-LabRG)[0]
   ```
8. From the Cloud Shell pane, run the following to establish security context granting full control to the storage account:

   ```
   $keyContext = $storageAccount.Context
   ```
9. From the Cloud Shell pane, run the following to create a blob-specific SAS token based on the access policy you created in the previous task:

   ```
   $sasToken = New-AzStorageBlobSASToken -Container 'labcontainer' -Blob 'splashscreen.contrast-white
   ```
10. From the Cloud Shell pane, run the following to establish security context based on the newly created SAS token:

    ```
    $sasContext = New-AzStorageContext $storageAccount.StorageAccountName -SasToken $sasToken
    ```
11. From the Cloud Shell pane, run the following to retrieve properties of the blob:

    ```
    Get-AzStorageBlob -Container 'labcontainer' -Blob 'splashscreen.contrast-white_scale-400.png' -Con
    ```
12. Verify that you successfully accessed the blob.
13. Minimize the Cloud Shell pane.

#### 7.2.0.5 Task 5: Invalidate a SAS token by modifying its access policy.

1. In the Azure portal, navigate to the **labcontainer - Access policy** blade.
2. Edit the existing policy **labcontainer-read** by setting its start and expiry time to yesterday's date.
3. Reopen the Cloud Shell pane.
4. From the Cloud Shell pane, re-run the following to attempt retrieving properties of the blob:

   ```
   Get-AzStorageBlob -Container 'labcontainer' -Blob 'splashscreen.contrast-white_scale-400.png' -Con
   ```
5. Verify that you no longer can access the blob.

   **Result**: After you completed this exercise, you have created a blob container, uploaded a file into it, and tested access control by using a SAS token and a stored access policy.

## 7.3 Exercise 3: Remove lab resources

#### 7.3.0.1 Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. If needed, switch to the Bash shell session by using the drop down list in the upper left corner of the Cloud Shell pane.
3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all resource groups you created in this lab:

```
az group list --query "[?starts_with(name,'az30002')]".name --output tsv
```

4. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

#### 7.3.0.2 Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

```
az group list --query "[?starts_with(name,'az30002')]".name --output tsv | xargs -L1 bash -c 'az g
```

2. Close the **Cloud Shell** prompt at the bottom of the portal.

   **Result**: In this exercise, you removed the resources used in this lab. # Implementing Advanced Virtual Networking

# 8 Lab: Implementing Azure Load Balancer Standard

### 8.0.1 Scenario

Adatum Corporation wants to implement Azure Load Balancer Standard to direct inbound and outbound traffic of Azure VMs.

### 8.0.2 Objectives

After completing this lab, you will be able to:

- Implement inbound load balancing by using Azure Load Balancer Standard
- Configure outbound SNAT traffic by using Azure Load Balancer Standard

### 8.0.3 Lab Setup

Estimated Time: 45 minutes

User Name: **Student**

Password: **Pa55w.rd**

## 8.1 Exercise 1: Implement inbound load balancing and NAT by using Azure Load Balancer Standard

The main tasks for this exercise are as follows:

1. Deploy Azure VMs in an availability set by using an Azure Resource Manager template
2. Create an instance of Azure Load Balancer Standard
3. Create a load balancing rule of Azure Load Balancer Standard
4. Create a NAT rule of Azure Load Balancer Standard
5. Test functionality of Azure Load Balancer Standard

#### 8.1.0.1 Task 1: Deploy Azure VMs in an availability set by using an Azure Resource Manager template

1. From the lab virtual machine, start Microsoft Edge and browse to the Azure portal at **http://portal.azure.com** and sign in by using the Microsoft account that has the Owner role in the target Azure subscription.
2. In the Azure portal, in the Microsoft Edge window, start a **Bash** session within the **Cloud Shell**.
3. If you are presented with the **You have no storage mounted** message, configure storage using the following settings:
   - Subsciption: the name of the target Azure subscription
   - Cloud Shell region: the name of the Azure region that is available in your subscription and which is closest to the lab location

- Resource group: the name of a new resource group **az3000800-LabRG**
- Storage account: a name of a new storage account
- File share: a name of a new file share

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

   ```
   az group create --name az3000801-LabRG --location <Azure region>
   ```

5. From the Cloud Shell pane, upload the Azure Resource Manager template **\allfiles\AZ-300T03\Module_03\azurec** into the home directory.
6. From the Cloud Shell pane, upload the parameter file **\allfiles\AZ-300T03\Module_03\azuredeploy0801.param** into the home directory.
7. From the Cloud Shell pane, deploy a pair of Azure VMs hosting Windows Server 2016 Datacenter by running:

   ```
   az deployment group create --resource-group az3000801-LabRG --template-file azuredeploy0801.json --
   ```

   > **Note**: Wait for the deployment before you proceed to the next task. This might take about 10 minutes.

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

#### 8.1.0.2 Task 2: Create an instance of Azure Load Balancer Standard

1. In the Azure portal, create a new Azure Load Balancer with the following settings:
   - Subsciption: the name of the target Azure subscription
   - Resource group: **az3000801-LabRG**
   - Name: **az3000801-lb**
   - Region: the name of the Azure region in which you deployed Azure VMs in the previous task of this exercise
   - Type: **Public**
   - SKU: **Standard**
   - Public IP address: **Create new** named **az3000801-lb-pip01**
   - Availability zone: **Zone-redundant**

#### 8.1.0.3 Task 3: Create a load balancing rule of Azure Load Balancer Standard

1. In the Azure portal, navigate to the blade displaying the properties of the newly deployed Azure Load Balancer **az3000801-lb**.
2. On the **az3000801-lb** blade, click **Backend pools**.
3. On the **az3000801-lb - Backend pools** blade, click **+ Add**.
4. On the **Add backend pool** blade, specify the following settings and click **Add**:
   - Name: **az3000801-bepool**
   - Virtual network: **az3000801-vnet (2 VM)**
   - VIRTUAL MACHINE: **az3000801-vm0** IP ADDRESS: **ipconfig1 (10.0.0.4)** or **ipconfig1 (10.0.0.5)**
   - VIRTUAL MACHINE: **az3000801-vm1** IP ADDRESS: **ipconfig1 (10.0.0.5)** or **ipconfig1 (10.0.0.4)**

   > **Note**: It is possible that the IP addresses of virtual machines are asssigned in the reversed order.

   > **Note**: Wait for the operation to complete. This should not take more than 1 minute.

5. Back on the **az3000801-lb - Backend pools** blade, click **Health probes**.
6. On the **az3000801-lb - Health probes** blade, click **+ Add**.
7. On the **Add health probe** blade, specify the following settings and click **OK**:
   - Name: **az3000801-healthprobe**
   - Protocol: **TCP**
   - Port: **80**
   - Interval: **5**
   - Unhealthy threshold: **2**

   **Note**: Wait for the operation to complete. This should not take more than 1 minute.
8. Back on the **az3000801-lb - Health probes** blade, click **Load balancing rules**.
9. On the **az3000801-lb - Load balancing rules** blade, click **+ Add**.
10. On the **Add load balancing rule** blade, specify the following settings and click **OK**:
    - Name: **az3000801-lbrule01**
    - IP Version: **IPv4**
    - Frontend IP address: select the public IP address assigned to the **LoadBalancedFrontEnd** from the drop-down list
    - Protocol: **TCP**
    - Port: **80**
    - Backend port: **80**
    - Backend pool: **az3000801-bepool (2 virtual machines)**
    - Health probe: **az3000801-healthprobe (TCP:80)**
    - Session persistence: **None**
    - Idle timeout (minutes): **4**
    - Floating IP (direct server return): **Disabled**

    **Note**: Wait for the operation to complete. This should not take more than 1 minute.

#### 8.1.0.4 Task 4: Create a NAT rule of Azure Load Balancer Standard

1. In the Azure portal, on the **az3000801-lb** blade, click **Inbound NAT rules**.
2. On the **az3000801-lb - Inbound NAT rules** blade, click **+ Add**.
3. On the **Add inbound NAT rule** blade, specify the following settings and click **OK**:
   - Name: **az3000801-vm0-RDP**
   - Frontend IP address: select the public IP address assigned to the **LoadBalancedFrontEnd** from the drop-down list
   - IP Version: **IPv4**
   - Service: **RDP**
   - Protocol: **TCP**
   - Port: **33890**
   - Target virtual machine: **az3000801-vm0**
   - Network IP configuration: **ipconfig1 (10.0.0.4)** or **ipconfig1 (10.0.0.5)**
   - Port mapping: **Custom**
   - Floating IP (direct server return): **Disabled**
   - Target port: **3389**

   **Note**: Wait for the operation to complete. This should not take more than 1 minute.

4. Back on the **az3000801-lb - Inbound NAT rules** blade, click **+ Add**.
5. On the **Add inbound NAT rule** blade, specify the following settings and click **OK**:
   - Name: **az3000801-vm1-RDP**
   - Frontend IP address: select the public IP address assigned to the **LoadBalancedFrontEnd** from the drop-down list
   - IP Version: **IPv4**
   - Service: **RDP**
   - Protocol: **TCP**
   - Port: **33891**
   - Target virtual machine: **az3000801-vm1**
   - Network IP configuration: **ipconfig1 (10.0.0.5)** or **ipconfig1 (10.0.0.4)**
   - Port mapping: **Custom**
   - Floating IP (direct server return): **Disabled**
   - Target port: **3389**

   **Note**: Wait for the operation to complete. This should not take more than 1 minute.

#### 8.1.0.5 Task 5: Test functionality of Azure Load Balancer Standard

1. In the Azure portal, navigate to the **az3000801-lb** blade and note the value of the **Public IP address** entry.
2. On the lab computer, start Microsoft Edge and navigate to the IP address you identified in the previous step.
3. Verify that you are presented with the default **Internet Information Services Welcome** page.
4. On the lab computer, right-click **Start**, click **Run**, and, from the **Open** text box, run the following (replace the `<IP address>` placeholder with the public IP address you identified earlier in this task):

   ```
   mstsc /v:<IP address>:33890
   ```

5. When prompted, authenticate by specifying the following values:
   - User name: **Student**
   - Password: **Pa55w.rd1234**
6. Within the Remote Desktop session, switch to the **Local Server** view in the Server Manager window and verify that you are connected to **az3000801-vm0** Azure VM.
7. Switch to the lab computer, right-click **Start**, click **Run**, and, from the **Open** text box, run the following (replace the `<IP address>` placeholder with the IP address you identified earlier in this task):

   ```
   mstsc /v:<IP address>:33891
   ```

8. When prompted, authenticate by specifying the following values:
   - User name: **Student**
   - Password: **Pa55w.rd1234**
9. Within the Remote Desktop session, switch to the **Local Server** view in the Server Manager window and verify that you are connected to **az3000801-vm1** Azure VM.
10. Within the Remote Desktop session, start a Windows PowerShell session and run the following to determine your current public IP address:

    ```
    Invoke-RestMethod http://ipinfo.io/json
    ```

11. Review the output of the cmdlet and verify that the IP address entry matches the public IP address you identified earlier in this task.

12. Leave the Remote Desktop sessions open. You will use them in the next exercise.

    **Result**: After you completed this exercise, you have implemented and tested Azure Load Balancer Standard inbound load balancing and NAT rules

## 8.2 Exercise 2: Configure outbound SNAT traffic by using Azure Load Balancer Standard

The main tasks for this exercise are as follows:

1. Deploy Azure VMs into an existing virtual network by using an Azure Resource Manager template
2. Create an Azure Standard Load Balancer and configure outbound SNAT rules
3. Test outbound rules of Azure Standard Load Balancer

#### 8.2.0.1 Task 1: Deploy Azure VMs into an existing virtual network by using an Azure Resource Manager template

1. From the lab virtual machine, start Microsoft Edge and browse to the Azure portal at **http://portal.azure.com** and sign in by using the Microsoft account that has the Owner role in the target Azure subscription.
2. In the Azure portal, in the Microsoft Edge window, start a **Bash** session within the **Cloud Shell**.
3. From the Cloud Shell pane, upload the Azure Resource Manager template **\allfiles\AZ-300T03\Module_03\azurec** into the home directory.
4. From the Cloud Shell pane, upload the parameter file **\allfiles\AZ-300T03\Module_03\azuredeploy0802.param** into the home directory.
5. From the Cloud Shell pane, deploy a pair of Azure VMs hosting Windows Server 2016 Datacenter by running:

   ```
   az deployment group create --resource-group az3000801-LabRG --template-file azuredeploy0802.json -
   ```

   > **Note**: Wait for the deployment before you proceed to the next task. This might take about 5 minutes.

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

#### 8.2.0.2 Task 2: Create an Azure Standard Load Balancer and configure outbound SNAT rules

1. In the Azure portal, in the Microsoft Edge window, start a **Bash** session within the **Cloud Shell**.
2. In the Azure portal, from the Cloud Shell pane, run the following to create an outbound public IP address of the load balancer:

   ```
   az network public-ip create --resource-group az3000801-LabRG --name az3000802-lb-pip01 --sku standa
   ```

3. In the Azure portal, from the Cloud Shell pane, run the following to create an Azure Load Balancer Standard:

   ```
   LOCATION=$(az group show --name az3000801-LabRG --query location --out tsv)
   az network lb create --resource-group az3000801-LabRG --name az3000802-lb --sku standard --backend-
   ```

4. From the Cloud Shell pane, run the following to create an outbound rule:

   ```
   az network lb outbound-rule create --resource-group az3000801-LabRG --lb-name az3000802-lb --name 
   ```

   > **Note**: Wait for the operation to complete. This should not take more than 1 minute.

5. Close the Cloud Shell pane.
6. In the Azure portal, navigate to the blade displaying the properties of the Azure Load Balancer **az3000802-lb**.
7. On the **az3000802-lb** blade, click **Backend pools**.
8. On the **az3000802-lb - Backend pools** blade, click **az3000802-bepool**.
9. On the **az3000802-bepool** blade, specify the following settings and click **Save**:

- Virtual network: **az3000801-vnet (4 VM)**
- VIRTUAL MACHINE: **az3000802-vm0** IP ADDRESS: **ipconfig1 (10.0.1.4)** or **ipconfig1 (10.0.1.5)**
- VIRTUAL MACHINE: **az3000802-vm1** IP ADDRESS: **ipconfig1 (10.0.1.5)** or **ipconfig1 (10.0.1.4)**

  **Note**: Wait for the operation to complete. This should not take more than 1 minute.

#### 8.2.0.3 Task 3: Verify that the outbound rule took effect

1. In the Azure portal, navigate to the **az3000802-lb** blade and note the value of the **Public IP address** entry.
2. On the lab computer, from the Remote Desktop session to **az3000801-vm0**, run the following to start a Remote Desktop session to **az3000802-vm0**.

   ```
   mstsc /v:az3000802-vm0
   ```
3. When prompted, authenticate by specifying the following values:
   - User name: **Student**
   - Password: **Pa55w.rd1234**
4. Within the Remote Desktop session to **az3000802-vm0**, start a Windows PowerShell session and run the following to determine your current public IP address:

   ```
   Invoke-RestMethod http://ipinfo.io/json
   ```
5. Review the output of the cmdlet and verify that the IP address entry matches the public IP address you identified earlier in this task.

   **Result**: After you completed this exercise, you have configured and tested Azure Load Balancer Standard outbound rules

## 8.3 Exercise 3: Remove lab resources

#### 8.3.0.1 Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. If needed, switch to the Bash shell session by using the drop down list in the upper left corner of the Cloud Shell pane.
3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all resource groups you created in this lab:

   ```
   az group list --query "[?starts_with(name,'az30008')]".name --output tsv
   ```
4. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

#### 8.3.0.2 Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

   ```
   az group list --query "[?starts_with(name,'az30008')]".name --output tsv | xargs -L1 bash -c 'az g
   ```
2. Close the **Cloud Shell** prompt at the bottom of the portal.

   **Result**: In this exercise, you removed the resources used in this lab. # Securing Identities

# 9 Lab: Implementing custom Role Based Access Control (RBAC) roles

### 9.0.1 Scenario

Adatum Corporation wants to implement custom RBAC roles to delegate permissions to start and stop (deallocate) Azure VMs.

### 9.0.2 Objectives

After completing this lab, you will be able to:

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

### 9.0.3 Lab Setup

Estimated Time: 30 minutes

User Name: **Student**

Password: **Pa55w.rd**

## 9.1 Exercise 1: Define a custom RBAC role

The main tasks for this exercise are as follows:

1. Deploy an Azure VM by using an Azure Resource Manager template
2. Identify actions to delegate via RBAC
3. Create a custom RBAC role in an Azure AD tenant

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

1. From the lab virtual machine, start Microsoft Edge and browse to the Azure portal at **http://portal.azure.com** and sign in by using the Microsoft account that has the Owner role in the target Azure subscription.
2. In the Azure portal, in the Microsoft Edge window, start a **PowerShell** session within the **Cloud Shell**.
3. If you are presented with the **You have no storage mounted** message, configure storage using the following settings:
   - Subsciption: the name of the target Azure subscription
   - Cloud Shell region: the name of the Azure region that is available in your subscription and which is closest to the lab location
   - Resource group: the name of a new resource group **az3000900-LabRG**
   - Storage account: a name of a new storage account
   - File share: a name of a new file share
4. From the Cloud Shell pane, create a resource groups by running (replace the `<Azure region>` placeholder with the name of the Azure region that is available in your subscription and which is closest to the lab location)

   ```
   New-AzResourceGroup -Name az3000901-LabRG -Location <Azure region>
   ```
5. From the Cloud Shell pane, upload the Azure Resource Manager template **\allfiles\AZ-300T03\Module_04\azurec** into the home directory.
6. From the Cloud Shell pane, upload the parameter file **\allfiles\AZ-300T03\Module_04\azuredeploy09.paramete** into the home directory.
7. From the Cloud Shell pane, deploy an Azure VM hosting Ubuntu by running:

   ```
   New-AzResourceGroupDeployment -ResourceGroupName az3000901-LabRG -TemplateFile $home/azuredeploy09
   ```

   **Note**: Do not wait for the deployment to complete but instead proceed to the next task.

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

#### 9.1.0.2 Task 2: Identify actions to delegate via RBAC

1. In the Azure portal, navigate to the **az3000901-LabRG** blade.
2. On the **az3000901-LabRG** blade, click **Access Control (IAM)**.
3. On the **az3000901-LabRG - Access Control (IAM)** blade, click **Roles**.
4. On the **Roles** blade, click **Owner**.
5. On the **Owner** blade, click **Permissions**.
6. On the **Permissions (preview)** blade, click **Microsoft Compute**.
7. On the **Microsoft Compute** blade, click **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.

#### 9.1.0.3 Task 3: Create a custom RBAC role in an Azure AD tenant

1. On the lab computer, open the file **\allfiles\AZ-300T03\Module_04\customRoleDefinition09.json** and review its content:

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

2. In the Azure portal, in the Microsoft Edge window, start a **PowerShell** session within the **Cloud Shell**.
3. From the Cloud Shell pane, upload the Azure Resource Manager template **\allfiles\AZ-300T03\Module_04\custor** 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 $HOME/customRoleDefinition09.json) -Replace 'SUBSCRIPTION_ID', "$subscription_i
   ```

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

   ```
   New-AzRoleDefinition -InputFile $HOME/customRoleDefinition09.json
   ```

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

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

7. Close the Cloud Shell pane.

   **Result**: After you completed this exercise, you have defined a custom RBAC role

## 9.2 Exercise 2: Assign and test a custom RBAC role

The main tasks for this exercise are as follows:

1. Create an Azure AD user

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

#### 9.2.0.1 Task 1: Create an Azure AD user

1. In the Azure portal, in the Microsoft Edge window, start a **PowerShell** session within the **Cloud Shell**.
2. From the Cloud Shell pane, run the following to explicitly authenticate to the target 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 'lab user0901' -PasswordProfile $passwordProfil
   ```

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 'labuser0901'").UserPrincipalName
   ```

6. Close the Cloud Shell pane.

#### 9.2.0.2 Task 2: Create an RBAC role assignment

1. In the Azure portal, navigate to the **az3000901-LabRG** blade.
2. On the **az3000901-LabRG** blade, click **Access Control (IAM)**.
3. On the **az3000901-LabRG - Access Control (IAM)** blade, click **+ Add** and select the **Add role assignment** option.
4. On the **Add role assignment** blade, specify the following settings and click **Save**:
   - Role: **Virtual Machine Operator (Custom)**
   - Assign access to: **Azure AD user, group, or service principal**
   - Select: **lab user0901**

#### 9.2.0.3 Task 3: Test the RBAC role assignment

1. Start a new in-private Microsoft Edge window, browse to the Azure portal at **http://portal.azure.com** and sign in by using the newly created user account:
   - Username: the user principal name you identified in the first task of this exercise
   - Password: **Pa55w.rd1234**
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 **az3000901-vm** and its managed disk.
4. In the Azure portal, navigate to the **az3000901-vm** blade. Try restarting 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. Stop the virtual machine and verify that the action completed successfully.

   **Result**: After you completed this exercise, you have assigned and tested a custom RBAC role

## 9.3 Exercise 3: Remove lab resources

#### 9.3.0.1 Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. If needed, switch to the Bash shell session by using the drop down list in the upper left corner of the Cloud Shell pane.
3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all resource groups you created in this lab:

   ```
   az group list --query "[?starts_with(name,'az30009')]".name --output tsv
   ```

4. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

#### 9.3.0.2 Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

   ```
   az group list --query "[?starts_with(name,'az30009')]".name --output tsv | xargs -L1 bash -c 'az g
   ```

2. Close the **Cloud Shell** prompt at the bottom of the portal.

   **Result**: In this exercise, you removed the resources used in this lab. # Implementing and Managing Application Services

# 10 Lab: Implementing Azure Logic Apps

### 10.0.1 Scenario

Adatum Corporation wants to implement custom monitoring of changes to a resource group.

### 10.0.2 Objectives

After completing this lab, you will be able to:

- Create an Azure logic app
- Configure integration of an Azure logic app and an Azure event grid

### 10.0.3 Lab Setup

Estimated Time: 45 minutes

User Name: **Student**

Password: **Pa55w.rd**

## 10.1 Exercise 1: Set up the lab environment that consists of an Azure storage account and an Azure logic app

The main tasks for this exercise are as follows:

1. Create an Azure storage account
2. Create an Azure logic app
3. Create an Azure AD service principal
4. Assign the Reader role to the Azure AD service principal
5. Register the Microsoft.EventGrid resource provider

#### 10.1.0.1 Task 1: Create a storage account in Azure

1. From the lab virtual machine, start Microsoft Edge and browse to the Azure portal at **http://portal.azure.com** and sign in by using the Microsoft account that has the Owner role in the target Azure subscription.
2. From Azure Portal, create a new storage account with the following settings:
   - Subscription: the name of the target Azure subscription
   - Resource group: a new resource group named **az3000701-LabRG**
   - Storage account name: any valid, unique name between 3 and 24 characters consisting of lowercase letters and digits
   - Location: the name of the Azure region that is available in your subscription and which is closest to the lab location
   - Performance: **Standard**
   - Account kind: **Storage (general purpose v1)**
   - Replication: **Locally-redundant storage (LRS)**
   - Network connectivity: **Public endpoint(All networks)**
   - Secure transfer required: **Enabled**
   - Data Lake Storage Gen2: **Disabled**

     **Note**: Do not wait for the deployment to complete but instead proceed to the next task.

#### 10.1.0.2 Task 2: Create an Azure logic app

1. From Azure Portal, create an instance of **Logic App** with the following settings:
   - Name: **logicapp3000701**
   - Subscription: the name of the target Azure subscription
   - Resource group: the name of a new resource group **az3000702-LabRG**
   - Location: the same Azure region into which you deployed the storage account in the previous task
   - Log Analytics: **Off**
2. Wait until the app is provisioned. This will take about a minute.

#### 10.1.0.3 Task 3: Create an Azure AD service principal

1. In the Azure portal, in the Microsoft Edge window, start a **PowerShell** session within the **Cloud Shell**.
2. If you are presented with the **You have no storage mounted** message, configure storage using the following settings:
   - Subsciption: the name of the target Azure subscription
   - Cloud Shell region: the name of the Azure region that is available in your subscription and which is closest to the lab location
   - Resource group: the name of a new resource group **az3000700-LabRG**
   - Storage account: a name of a new storage account
   - File share: a name of a new file share
3. From the Cloud Shell pane, run the following to create a new Azure AD application that you will associate with the service principal you create in the subsequent steps of this task:

```
$password = 'Pa55w.rd1234'
$securePassword = ConvertTo-SecureString -Force -AsPlainText -String $password
$aadApp30007 = New-AzADApplication -DisplayName 'aadApp30007' -HomePage 'http://aadApp30007' -Iden
```

4. From the Cloud Shell pane, run the following to create a new Azure AD service principal associated with the application you created in the previous step:

```
New-AzADServicePrincipal -ApplicationId $aadApp30007.ApplicationId.Guid
```

5. In the output of the **New-AzADServicePrincipal** command, note the value of the **ApplicationId** property. You will need this in the next exercise of this lab.
6. From the Cloud Shell pane, run the following to identify the value of the **Id** property of the current Azure subscription and the value of the **TenantId** property of the Azure AD tenant associated with that subscription (you will also need them in the next exercise of this lab):

   ```
   Get-AzSubscription
   ```
7. Close the Cloud Shell pane.

#### 10.1.0.4 Task 4: Assign the Reader role to the Azure AD service principal

1. In the Azure portal, navigate to the blade displaying properties of your Azure subscription.
2. On the Azure subscription blade, click **Access control (IAM)**.
3. Assign the **Reader** role within the scope of the Azure subscription to the **aadApp30007** service principal.

#### 10.1.0.5 Task 5: Register the Microsoft.EventGrid resource provider

1. In the Azure portal, in the Microsoft Edge window, reopen the **PowerShell** session within the **Cloud Shell**.
2. From the Cloud Shell pane, run the following to register the Microsoft.EventGrid resource provider:

   ```
   Register-AzResourceProvider -ProviderNamespace Microsoft.EventGrid
   ```
3. Close the Cloud Shell pane.

   **Result**: After you completed this exercise, you have created a storage account, a logic app that you will configure in the next exercise of this lab, and an Azure AD service principal that you will reference during that configuration.

## 10.2 Exercise 2: Configure Azure logic app to monitor changes to a resource group.

The main tasks for this exercise are as follows:

1. Add a trigger to the Azure logic app
2. Add an action to the Azure logic app
3. Identify the callback URL of the Azure logic app
4. Configure an event subscription
5. Test the logic app

#### 10.2.0.1 Task 1: Add a trigger to the Azure logic app

1. In the the Azure portal, navigate to the **Logic App Designer** blade of the newly provisioned Azure logic app.
2. Click **Blank Logic App**. This will create a blank designer workspace and display a list of connectors and triggers to add to the workspace.
3. Search for **Event Grid** triggers and, in the list of results, click the **When a resource event occurs** Azure Event Grid trigger to add it to the designer workspace.
4. In the **Azure Event Grid** tile, click the **Connect with Service Principal** link, specify the following values, and click **Create**:
   - Connection Name: **egc30007**
   - Client ID: the **ApplicationId** property you identified in the previous exercise
   - Client Secret: **Pa55w.rd1234**
   - Tenant: the **TenantId** property you identified in the previous exercise
5. In the **When a resource event occurs** tile, specify the following values:

  - Subscription: the subscription **Id** property you identified in the previous exercise
  - Resource Type: **Microsoft.Resources.resourceGroups**
  - Resource Name: **az3000701-LabRG**
  - Event Type Item - 1: **Microsoft.Resources.ResourceWriteSuccess**
  - Event Type Item - 2: **Microsoft.Resources.ResourceDeleteSuccess**

6. Click **Add new parameter** and select **Subscription Name**
7. In the **Subscription Name** text box, type **event-subscription-az3000701** and click **Save**.

#### 10.2.0.2 Task 2: Add an action to the Azure logic app

1. In the the Azure portal, on the Logic App Designer blade of the newly provisioned Azure logic app, click **+ New step**.
2. In the **Choose an action** pane, in the **Search connectors and actions** text box, type **Outlook**.
3. In the list of results, click **Outlook.com**.
4. In the list of actions for **Outlook.com**, click **Send an email**.
5. In the **Outlook.com** pane, click **Sign in**.
6. When prompted, authenticate by using the Microsoft Account you are using in this lab.
7. When prompted for the consent to grant Azure Logic App permissions to access Outlook resources, click **Yes**.
8. In the **Send an email** pane, specify the following settings and click **Save**:
   - To: the primary e-mail address of your Microsoft Account
   - Subject: type **Resource updated:** and, in the **Dynamic Content** column to the right of the **Send an email** pane, click **Subject**.
   - Body: type **Resource group:**, in the **Dynamic Content** column to the right of the **Send an email** pane, click **Topic**, type **Event type:**, in the **Dynamic Content** column to the right of the **Send an email** pane, click **Event Type**, type **Event ID:**, in the **Dynamic Content** column to the right of the **Send an email** pane, click **ID**, type **Event Time:**, and in the **Dynamic Content** column to the right of the **Send an email** pane, click **Event Time**.

#### 10.2.0.3 Task 3: Identify the callback URL of the Azure logic app

1. In the Azure portal, navigate to the **logicapp3000701** blade and, in the **Summary** section, click **See trigger history**. Ignore any **Forbidden** error messages.
2. On the **When_a_resource_event_occurs** blade, copy the value of the **Callback url [POST]** text box.

#### 10.2.0.4 Task 4: Configure an event subscription

1. In the Azure portal, navigate to the **az3000701-LabRG** resource group and, in the vertical menu, click **Events**.
2. On the **az3000701-LabRG - Events** blade, select **Get Started** and click **Web Hook**.
3. On the **Create Event Subscription** blade, in the **Filter to Event Types** drop down list, ensure that only the checkboxes next to the **Resource Write Success** and **Resource Delete Success** are selected.
4. In the **Endpoint Type** drop down list, ensure that **Web Hook** is selected and click the **Select an endpoint** link.
5. On the **Select Web Hook** blade, in the **Subscriber Endpoint**, paste the value of the **Callback url [POST]** of the Azure logic app you copied in the previous task and click **Confirm Selection**.
6. In the **Name** text box within the **EVENT SUBSCRIPTION DETAILS** section, type **event-subscription-az3000701**.
7. Click **Create**.

#### 10.2.0.5 Task 5: Test the logic app

1. In the Azure portal, navigate to the **az3000701-LabRG** resource group and, in the vertical menu, click **Overview**.
2. In the list of resources, click the Azure storage account you created in the first exercise.
3. On the storage account blade, in the vertical menu, click **Configuration**.
4. On the configuration blade, set the **Secure transfer required** setting to **Disabled** and click **Save**
5. Navigate to the **logicapp3000701** blade, click **Refresh**, and note that the **Runs history** includes the entry corresponding to configuration change of the Azure storage account.
6. Navigate to the inbox of the email account you configured in this exercise and verify that includes an email generated by the logic app.

   **Result**: After you completed this exercise, you have configured an Azure logic app to monitor changes to a resource group.

## 10.3 Exercise 3: Remove lab resources

#### 10.3.0.1 Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. If needed, switch to the Bash shell session by using the drop down list in the upper left corner of the Cloud Shell pane.
3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all resource groups you created in this lab:

   ```
   az group list --query "[?starts_with(name,'az30007')]".name --output tsv
   ```
4. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

#### 10.3.0.2 Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

   ```
   az group list --query "[?starts_with(name,'az30007')]".name --output tsv | xargs -L1 bash -c 'az g
   ```
2. Close the **Cloud Shell** prompt at the bottom of the portal.

   **Result**: In this exercise, you removed the resources used in this lab. # Developing for the cloud

# 11 Lab: Configuring a Message-Based Integration Architecture

### 11.0.1 Scenario

Adatum 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 eventing 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 of time (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.

#### 11.0.2 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

#### 11.0.3 Lab

Estimated Time: 60 minutes

## 11.1 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

##### 11.1.0.1 Task 1: Configure an Azure Function App Storage Blob trigger

1. From the lab virtual machine, start Microsoft Edge, browse to the Azure portal at **http://portal.azure.com**, and sign in by using the Microsoft account that has the Owner role in the target Azure subscription.
2. In the Azure portal, in the Microsoft Edge window, start a **Bash** session within the **Cloud Shell**.
3. If you are presented with the **You have no storage mounted** message, configure storage using the following settings:
    - Subsciption: the name of the target Azure subscription
    - Cloud Shell region: the name of the Azure region that is available in your subscription and which is closest to the lab location

- Resource group: **az300T0601-LabRG**
- Storage account: a name of a new storage account
- File share: a name of a new file share

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 (make sure to replace the placeholder `<Azure region>` with the name of the target Azure region and remove any spaces in the region name):

```
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='az300T0602-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 and a container that will host blobs to be processed by the Azure function:

```
export STORAGE_ACCOUNT_NAME="az300t06st2${PREFIX}"

export CONTAINER_NAME="workitems"

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

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

   **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 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="az300t06ai${PREFIX}"

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

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

10. 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="az300t06f${PREFIX}"

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

11. 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
```

12. Switch to the Azure portal and navigate to the blade of the Azure Function app you created earlier in this task.
13. On the Azure Function app blade, click **Functions** and then, click **+ New function**.

14. On the **Choose a template below or go to the quickstart** blade, click **Azure Blob Storage trigger** template.
15. On the **Extensions not Installed** blade, click **Install**. Wait until the installation completes and click **Continue**.

    > **Note**: Azure Functions V2 runtime implement bindings in the form of Nuget packages, referred to as "binding extensions" (in particular, the Azure Storage Blob binding uses the Microsoft.Azure.WebJobs.Extensions.Storage package).

16. On the **Azure Blob Storage trigger** blade, specify the following and click **Create** to create a new function within the Azure function:
    - Name: **BlobTrigger**
    - Path: **workitems/{name}**
    - Storage account connection: **STORAGE_CONNECTION_STRING**
17. On the Azure Function app **BlobTrigger** function blade, 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.

#### 11.1.0.2 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='az300T0602-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. On the Azure Function app blade, click **Monitor** entry in the **Functions** section.
6. Note a single event entry representing uploading of the blob. Click the entry to view the **Invocation Details** blade.

    > **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.

7. Go back to the **Monitor** blade, and click the link **Run in Application Insights**.
8. In the Application Insights portal, review the autogenerated Kusto query and its results.

## 11.2 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

#### 11.2.0.1 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='az300T0602-LabRG'

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

   export RESOURCE_GROUP_NAME='az300T0603-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 and its container that will be used by the Event Grid subscription that you will configure in this task:

   ```
   export STORAGE_ACCOUNT_NAME="az300t06st3${PREFIX}"

   export CONTAINER_NAME="workitems"

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

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

6. 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" -
   ```

7. 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="az300t06q3${PREFIX}"

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

8. 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="az300t06qsub3${PREFIX}"

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

#### 11.2.0.2 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, click **Queues** to display the list of its queues.
4. Click the entry representing the queue you created in the previous task of this exercise.
5. Note that the queue contains a single message. Click 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.

## 11.3 Exercise 3: Remove lab resources

#### 11.3.0.1 Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. If needed, switch to the Bash shell session by using the drop down list in the upper left corner of the Cloud Shell pane.
3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all resource groups you created in this lab:

   ```
   az group list --query "[?starts_with(name,'az300T06')]".name --output tsv
   ```

4. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

#### 11.3.0.2 Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

   ```
   az group list --query "[?starts_with(name,'az300T06')]".name --output tsv | xargs -L1 bash -c 'az 
   ```

2. Close the **Cloud Shell** prompt at the bottom of the portal.

   **Result**: In this exercise, you removed the resources used in this lab.