Contents

1 Managing Security and Identity for Azure Solutions 5

2 Lab Answer Key: Securing Secrets in Azure 5
2.1 Before we start . . . 5
2.2 Exercise 1: Deploy Key Vault resources . . . 5
2.2.0.1 Task 1: Open the Azure Portal . . . 5
2.2.0.2 Task 2: Deploy a key vault . . . 5
2.2.0.3 Task 3: Add a secret to a key vault by using the Azure portal . . . 6
2.2.0.4 Task 4: Open Cloud Shell . . . 6
2.2.0.5 Task 5: Add a secret to a key vault using the CLI . . . 6
2.2.0.6 Task 6: Add secrets to a key vault by using Azure Resource Manager templates 7
2.2.0.7 Task 7: View key vault secrets . . . 9
2.3 Exercise 2: Deploy Azure VM using Key Vault secret . . . 9
2.3.0.1 Task 1: Retrieve the value of the key vault Resource Id parameter . . . 9
2.3.0.2 Task 2: Prepare the Azure Resource Manager deployment template and parameters files . . . 10
2.3.0.3 Task 3: Configure a key vault for deployment of Azure Resource Manager templates 10
2.3.0.4 Task 4: Deploy a Linux VM with the password parameter set by using a key vault secret. . . . 10
2.3.0.5 Task 5: Verify the outcome of the deployment . . . 10
2.4 Exercise 3: Remove lab resources . . . 11
2.4.0.1 Task 1: Open Cloud Shell . . . 11
2.4.0.2 Task 2: Delete resource groups . . . 11

3 Integrating SaaS Services Available on the Azure Platform 11

4 Lab Answer Key: Deploying Service Instances as Components of Overall Azure Solutions 11
4.1 Before we start . . . 11
4.2 Exercise 1: Deploy Function App and Cognitive Service using ARM Template . . . 12
4.2.0.1 Task 1: Open the Azure portal . . . 12
4.2.0.2 Task 2: Deploy Cognitive Service using an Azure Resource Manager template . 12
4.2.0.3 Task 3: Deploy a function app . . . 13
4.2.0.4 Task 4: Test a function app using Cognitive Services . . . 14
4.3 Exercise 2: Create a Logic App that uses a Function App . . . 15
4.3.0.1 Task 1: Create a logic app . . . 15
4.3.0.2 Task 2: Configure logic app steps . . . 15
4.3.0.3 Task 2: Open Cloud Shell . . . 17
4.3.0.4 Task 3: Validate Logic App using Python . . . 17
4.4 Exercise 3: Remove lab resources . . . 18
4.4.0.1 Task 1: Open Cloud Shell . . . 18
4.4.0.2 Task 2: Delete resource groups . . . 18

5 Comparing Database Options in Azure 18

6 Lab Answer Key: Deploying Database Instances in Azure 18
6.1 Before we start . . . 18
6.2 Exercise 1: Deploy a Cosmos DB database . . . 19
6.2.0.1 Task 1: Open the Azure Portal . . . 19
6.2.0.2 Task 2: Create a Cosmos DB database and collection . . . 19
6.2.0.3 Task 3: Create and query documents in Cosmos DB . . . 20
6.3 Exercise 2: Deploy Application using Cosmos DB . . . 22
6.3.0.1 Task 1: Deploy API App code using Azure Resource Manager templates and GitHub . . . 22
6.3.0.2 Task 2: Validate API App . . . 22
6.4 Exercise 3: Connect Cosmos DB to Azure Search . . . 23
6.4.0.1 Task 1: Create Azure Cognitive Search Instance . . . 23
6.4.0.2 Task 2: Index Cosmos DB Data in Azure Search . . . 24
6.4.0.3 Task 3: Validate API App . . . 25
6.5 Exercise 4: Remove lab resources . . . 26
6.5.0.1 Task 1: Delete the resource group . . . 26

7 Monitoring and automating Azure solutions 26

8 Lab Answer Key: Deploying Configuration Management solutions to Azure 26
8.1 Before we start . . . 26
8.2 Exercise 1: Deploy compute resources . . . 26
8.2.0.1 Task 1: Open the Azure portal . . . 26
8.2.0.2 Task 2: Open Cloud Shell . . . 27
8.2.0.3 Task 3: Deploy a Linux VM . . . 27
8.2.0.4 Task 4: Deploy an Azure Automation account . . . 29
8.3 Exercise 2: Configure Azure Automation DSC . . . 30
8.3.0.1 Task 1: Import Linux PowerShell DSC modules . . . 30
8.3.0.2 Task 2: Create Linux DSC Configuration . . . 30
8.3.0.3 Task 3: Onboard Linux VM . . . 31
8.3.0.4 Task 4: Validate Linux VM onboarding . . . 31
8.4 Exercise 3: Remove lab resources . . . 32
8.4.0.1 Task 1: Open Cloud Shell . . . 32
8.4.0.2 Task 2: Delete resource groups . . . 32

9 Deploying Resources with Azure Resource Manager 32

10 Lab Answer Key: Getting Started with Azure Resource Manager Templates and Azure Building Blocks 32
10.1 Before we start . . . 32
10.2 Exercise 1: Deploy core Azure resources by using an Azure Resource Manager Template from the Azure portal . . . 33
10.2.0.1 Task 1: Open the Azure Portal . . . 33
10.2.0.2 Task 2: Deploy an Azure virtual network from the Azure portal by using an Azure Resource Manager template . . . 33
10.2.0.3 Task 3: View deployment metadata . . . 34
10.3 Exercise 2: Deploy core Azure resources by using Azure Building Blocks from the Azure Cloud Shell . . . 35
10.3.0.1 Task 1: Open Cloud Shell . . . 35
10.3.0.2 Task 2: Install the Azure Building Blocks npm package in Azure Cloud Shell . . . 35
10.3.0.3 Task 3: Deploy an Azure virtual network from Cloud Shell by using Azure Building Blocks . . . 36
10.3.0.4 Task 4: View deployment metadata . . . 36
10.4 Exercise 3: Remove lab resources . . . 37
10.4.0.1 Task 1: Open Cloud Shell . . . 37
10.4.0.2 Task 2: Delete resource groups . . . 37

11 Creating Managed Server Applications in Azure 37

12 Lab Answer Key: Deploying Managed Containerized Workloads to Azure 37
12.1 Before we start . . . 37
12.2 Exercise 1: Create Azure Kubernetes Service (AKS) cluster . . . 37
12.2.0.1 Task 1: Open the Azure Portal . . . 37
12.2.0.2 Task 2: Open Cloud Shell . . . 38
12.2.0.3 Task 3: Create an AKS cluster by using Cloud Shell . . . 38
12.2.0.4 Task 4: Connect to the AKS cluster. . . . 39
12.3 Exercise 2: Managing an AKS cluster and its containerized workloads. . . . 39
12.3.0.1 Task 1: Deploy a containerized application to an AKS cluster . . . 39
12.3.0.2 Task 2: Scaling containerized applications and AKS cluster nodes . . . 39
12.4 Exercise 3: Autoscaling pods in an AKS cluster . . . 40
12.4.0.1 Task 1: Deploy a Kubernetes pod by using a **.yaml** file. . . . 40
12.4.0.2 Task 2: Autoscale Kubernetes pods. . . . 41
12.5 Exercise 4: Implement DevOps with AKS . . . 42
12.5.0.1 Task 1: Deploy DevOps with AKS . . . 42
12.5.0.2 Task 2: Review the DevOps with AKS architecture . . . 44
12.6 Exercise 5: Remove lab resources . . . 44
12.6.0.1 Task 1: Open Cloud Shell . . . 44
12.6.0.2 Task 2: Delete resource groups . . . 44

13 Authoring Serverless Applications in Azure **44**

14 Lab Answer Key: Deploying Serverless Workloads to Azure **44**
14.1 Before we start . . . 44
14.2 Exercise 1: Create Web App . . . 45
14.2.0.1 Task 1: Open the Azure Portal . . . 45
14.2.0.2 Task 2: Open Cloud Shell . . . 45
14.2.0.3 Task 3: Create an App Service plan . . . 45
14.2.0.4 Task 4: Create a Web App instance . . . 46
14.2.0.5 Task 5: View deployment results . . . 46
14.3 Exercise 2: Deploy Web App code . . . 46
14.3.0.1 Task 1: Deploy code with a Web App Extension using an Azure Resource Manager template and GitHub . . . 46
14.3.0.2 Task 2: View deployment results . . . 48
14.3.0.3 Task 3: Deploy Code with a Docker Hub container image . . . 48
14.3.0.4 Task 4: View deployment results . . . 48
14.4 Exercise 3: Deploy a Function App . . . 48
14.4.0.1 Task 1: Deploy a Function App with code using an Azure Resource Manager template . . . 48
14.4.0.2 Task 2: View deployment results . . . 49
14.5 Exercise 4: Remove lab resources . . . 49
14.5.0.1 Task 1: Open Cloud Shell . . . 49
14.5.0.2 Task 2: Delete resource groups . . . 50

15 Building Azure IaaS-Based Server Applications. **50**

16 Lab Answer Key: Building Azure IaaS-Based Server Applications by using Azure Resource Manager Templates and Azure Building Blocks. **50**
16.1 Before we start . . . 50
16.2 Exercise 1: Deploy an Azure VM by using Azure Resource Manager templates with PowerShell Desired State Configuration (DSC) extension from the Azure portal. . . . 50
16.2.0.1 Task 1: Open the Azure Portal . . . 50
16.2.0.2 Task 2: Create an Azure VM running Windows Server 2016 Datacenter. . . . 50
16.2.0.3 Task 3: View DSC configuration . . . 52
16.2.0.4 Task 4: Create an Azure Storage account . . . 52
16.2.0.5 Task 5: Upload DSC configuration to Azure Storage . . . 53
16.2.0.6 Task 6: Deploy an Azure VM by using an Azure Resource Manager template with PowerShell DSC extension from the Azure portal. . . . 53
16.2.0.7 Task 7: Validate that the Azure VM is serving web content . . . 54
16.3 Exercise 2: Deploy an Azure Virtual Machine Scale Set (VMSS) by using Azure Resource Manager templates with PowerShell Desired State Configuration (DSC) extension from the Azure portal. . . 55
16.3.0.1 Task 1: View an Azure Resource Manager template. . . . 55
16.3.0.2 Task 2: Deploy a VMSS using ARM . . . 55
16.3.0.3 Task 3: Validate that VMSS instances are serving web content . . . 56
16.4 Exercise 3: Deploy Azure VMs running Windows Server 2016 and Linux by using Azure Building Blocks with PowerShell Desired State Configuration (DSC) extension. . . . 56
16.4.0.1 Task 1: Open Cloud Shell . . . 56
16.4.0.2 Task 2: Deploy an Azure VM running Linux Ubuntu 18.04 that will be used to perform Azure Building Blocks-based deployments. . . . 56
16.4.0.3 Task 3: Install the Azure Building Blocks npm package in the Azure VM running Linux. . . . 57
16.4.0.4 Task 4: Deploy a Windows Server 2016 Azure VM from Cloud Shell by using Azure Building Blocks . . . 58
16.4.0.5 Task 5: Validate that the Windows Server 2016 Azure VM is serving web content 59
16.4.0.6 Task 6: Deploy a Linux Azure VM from Cloud Shell by using Azure Building Blocks . . . 59
16.4.0.7 Task 7: Validate that the Linux Azure VM is serving web content . . . 60
16.5 Exercise 4: Remove lab resources . . . 60
16.5.0.1 Task 1: Open Cloud Shell . . . 60
16.5.0.2 Task 2: Delete resource groups . . . 61

17 Networking Azure Application Components **61**

18 Lab Answer Key: Deploying Network Infrastructure for Use in Azure Solutions **61**
18.1 Before we start . . . 61
18.2 Exercise 1: Configure the lab environment . . . 61
18.2.0.1 Task 1: Open the Azure Portal and Cloud Shell. . . . 61
18.2.0.2 Task 2: Deploy an Azure VM running Linux Ubuntu 18.04 that will be used to perform Azure Building Blocks-based deployments. . . . 62
18.2.0.3 Task 3: Install the Azure Building Blocks npm package in the Azure VM running Linux. . . . 63
18.2.0.4 Task 4: Prepare Building Blocks Hub and Spoke parameter files . . . 63
18.2.0.5 Task 5: Implement the hub component of the Hub and Spoke design . . . 65
18.3 Exercise 2: Review the Hub-spoke topology . . . 66
18.3.0.1 Task 1: Examine the peering configuration . . . 66
18.3.0.2 Task 2: Examine the routing configuration . . . 66
18.3.0.3 Task 3: Verify connectivity between spokes . . . 66
18.4 Exercise 3: Remove lab resources . . . 67
18.4.0.1 Task 1: Open Cloud Shell . . . 67
18.4.0.2 Task 2: Delete resource groups . . . 67

19 Integrating Azure Solution Components using Messaging Services **67**

20 Lab Answer Key: Deploying Messaging components to facilitate communication between Azure resources **67**
20.1 Before we start . . . 67
20.2 Exercise 1: Deploy a Service Bus namespace . . . 68
20.2.0.1 Task 1: Open the Azure portal . . . 68
20.2.0.2 Task 2: Create a Service Bus namespace . . . 68
20.2.0.3 Task 3: Create a Service Bus Queue . . . 68
20.2.0.4 Task 4: Get Service Bus Connection String . . . 68
20.3 Exercise 2: Create a logic app . . . 69
20.3.0.1 Task 1: Create an Azure Storage account . . . 69
20.3.0.2 Task 2: Create a logic app . . . 69
20.3.0.3 Task 3: Configure logic app steps. . . . 70
20.3.0.4 Task 4: Open Cloud Shell . . . 71
20.3.0.5 Task 5: Validate Logic App using Node.js . . . 71
20.4 Exercise 3: Remove lab resources . . . 72
20.4.0.1 Task 1: Open Cloud Shell . . . 72
20.4.0.2 Task 2: Delete resource groups . . . 72

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

1 Managing Security and Identity for Azure Solutions

2 Lab Answer Key: Securing Secrets in Azure

2.1 Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
   - Username: **Admin**
   - Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
   - Microsoft Edge
   - File Explorer
   - Visual Studio Code
   - Microsoft Azure Storage Explorer
   - Bash on Ubuntu on Windows
   - Windows PowerShell

     Note: You can also find shortcuts to these applications in the **Start Menu**.

2.2 Exercise 1: Deploy Key Vault resources

2.2.0.1 Task 1: Open the Azure Portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** (https://portal.azure.com).
3. When prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

2.2.0.2 Task 2: Deploy a key vault

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Key Vault** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Key Vault**.
4. On the **Key Vault** blade, click the **Create** button.
5. On the **Create key vault** blade, perform the following tasks:
   - Leave the **Subscription** drop-down list entry set to its default value.
   - In the **Resource group** section, select the **Create new** option and then, in the text box, type **AADesignLab0901-RG**.
   - In the **Key vault name** text box, type a globally unique value.
   - In the **Region** drop-down list, select the Azure region to which you intend to deploy resources in this lab.

- Leave all remaining settings with their default values.
- Click the **Review + Create** button and then click the **Create** button.

6. Wait for the provisioning to complete before you proceed to the next task.

2.2.0.3 Task 3: Add a secret to a key vault by using the Azure portal

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0901-RG**.
3. On the **AADesignLab0901-RG** blade, click the entry representing the newly created key vault.
4. On the key vault blade, click **Secrets**.
5. On the key vault secrets blade, click the **Generate/Import** button at the top of the pane.
6. On the **Create a secret** blade, perform the following tasks:
   - In the **Upload options** drop-down list, ensure that the **Manual** entry is selected.
   - In the **Name** text-box, type **thirdPartyKey**.
   - In the **Value** text box, enter the value **56d95961e597ed0f04b76e58**.
   - Leave all remaining settings with their default values.
   - Click the **Create** button.

2.2.0.4 Task 4: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.

   Note: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.

2. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.

   Note: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.

3. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:
   - Leave the **Subscription** drop-down list entry set to its default value.
   - In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you intend to deploy resources in this lab
   - In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab0901-RG**.
   - In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   - In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
   - Click the **Create storage** button.
4. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

2.2.0.5 Task 5: Add a secret to a key vault using the CLI

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group that contains the Azure key vault you deployed earlier in this exercise:

```
RESOURCE_GROUP='AADesignLab0901-RG'
```

2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to retrieve the name of the Azure key vault you created earlier in this exercise:

```
KEY_VAULT_NAME=$(az keyvault list --resource-group $RESOURCE_GROUP --query "[0].name" --output tsv
```

3. At the **Cloud Shell** command prompt, type in the following command, and press **Enter** to list secrets in the key vault:

```
az keyvault secret list --vault-name $KEY_VAULT_NAME --output json
```

4. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to display the value of the **thirdPartyKey** secret:

```
az keyvault secret show --vault-name $KEY_VAULT_NAME --name thirdPartyKey --query value --output t
```

5. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to add a new secret to your key vault:

```
az keyvault secret set --vault-name $KEY_VAULT_NAME --name firstPartyKey --value 56f8a55119845511c
```

6. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list secrets in the key vault:

```
az keyvault secret list --vault-name $KEY_VAULT_NAME --query "[*].{Id:id,Created:attributes.create
```

7. Close the **Cloud Shell** pane.

2.2.0.6 Task 6: Add secrets to a key vault by using Azure Resource Manager templates

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Template Deployment** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Template Deployment**.
4. On the **Template deployment** blade, click the **Create** button.
5. On the **Custom deployment** blade, click the **Build your own template in the editor** link.
6. On the **Edit template** blade, click **Load file**.
7. In the **Choose File to Upload** dialog box, navigate to the **\allfiles\AZ-301T01\Module_01\LabFiles\Starter** folder, select the **secret-template.json** file, and click **Open**. This will load the following content into the template editor pane:

```
{
    "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "vaultName": {
            "type": "string"
        }
    },
    "variables": {
        "secretName": "vmPassword"
    },
    "resources": [
        {
            "apiVersion": "2016-10-01",
            "type": "Microsoft.KeyVault/vaults/secrets",
            "name": "[concat(parameters('vaultName'), '/', variables('secretName'))]",
            "properties": {
                "contentType": "text/plain",
                "value": "StudentPa$$w.rd"
            }
        }
    ]
}
```

8. Click the **Save** button to persist the template.
9. Back on the **Custom deployment** blade, perform the following tasks:
    - Leave the **Subscription** drop-down list entry set to its default value.
    - In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab0901-RG**.
    - In the **Vault Name** text box, type the name of the key vault you created earlier in this exercise.
    - In the **Terms and Conditions** section, select the **I agree to the terms and conditions stated above** checkbox.
    - Click the **Purchase** button.
10. Do not wait for the deployment to complete but proceed to the next step.
11. In the upper left corner of the Azure portal, click **Create a resource**.
12. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Template Deployment** and press **Enter**.
13. On the **Everything** blade, in the search results, click **Template Deployment**.
14. On the **Template deployment** blade, click the **Create** button.
15. On the **Custom deployment** blade, click the **Build your own template in the editor** link.
16. On the **Edit template** blade, click **Load file**.
17. In the **Choose File to Upload** dialog box, navigate to the **\allfiles\AZ-301T01\Module_01\LabFiles\Starter** folder, select the **storage-template.json** file, and click **Open**. This will load the following content into the template editor pane:

```
{
    "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "vaultName": {
            "type": "string"
        }
    },
    "variables": {
        "secretName": "storageConnectionString",
        "storageName": "[concat('stor', uniqueString(resourceGroup().id))]"
    },
    "resources": [
        {
            "apiVersion": "2017-10-01",
            "type": "Microsoft.Storage/storageAccounts",
            "name": "[variables('storageName')]",
            "location": "[resourceGroup().location]",
            "kind": "Storage",
            "sku": {
                "name": "Standard_LRS"
            },
            "properties": {
            }
        },
        {
            "apiVersion": "2016-10-01",
            "type": "Microsoft.KeyVault/vaults/secrets",
            "name": "[concat(parameters('vaultName'), '/', variables('secretName'))]",
            "dependsOn": [
                "[resourceId('Microsoft.Storage/storageAccounts', variables('storageName'))]"
            ],
            "properties": {
```

```
                "contentType": "text/plain",
                "value": "[concat('DefaultEndpointsProtocol=https;AccountName=', variables('storag
            }
        }
    ]
}
```

18. Click the **Save** button to persist the template.
19. Back on the **Custom deployment** blade, perform the following tasks:
    - Leave the **Subscription** drop-down list entry set to its default value.
    - In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab0901-RG**.
    - In the **Vault Name** field, type the name of the key vault you created earlier in this exercise.
    - In the **Terms and Conditions** section, select the **I agree to the terms and conditions stated above** checkbox.
    - Click the **Purchase** button.
20. Wait for the deployment to complete before you proceed to the next task.

2.2.0.7 Task 7: View key vault secrets

1. In the hub menu of the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0901-RG**.
3. On the **AADesignLab0901-RG** blade, click the entry representing the key vault you created earlier in this exercise.
4. On the key vault blade, click **Secrets**.
5. On the key vault secrets blade, review the list of secrets created during this lab.
6. Click the entry representing the **vmPassword** secret.
7. On the **vmPassword** blade, click the entry representing the current version of the secret.
8. On the Secret Version blade, click the **Show secret value** button.
9. Verify that the value of the secret matches the one included in the template you deployed in the previous task.

   Review: In this exercise, you created a **Key Vault** instance and used several different methods to add secrets to the key vault.

2.3 Exercise 2: Deploy Azure VM using Key Vault secret

2.3.0.1 Task 1: Retrieve the value of the key vault Resource Id parameter

1. At the top of the portal, click the **Cloud Shell** icon to open a new Cloud Shell instance.
2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group that will contain the hub virtual network:

   ```
   RESOURCE_GROUP='AADesignLab0901-RG'
   ```

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to retrieve the resource id of the Azure key vault you created earlier in this exercise:

   ```
   KEY_VAULT_ID=$(az keyvault list --resource-group $RESOURCE_GROUP --query "[0].id" --output tsv)
   ```

4. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the Azure key vault resource id and which takes into account any special character the resource id might include:

   ```
   KEY_VAULT_ID_REGEX="$(echo $KEY_VAULT_ID | sed -e 's/\\/\\\\/g; s/\//\\\//g; s/&/\\\&/g')"
   ```

2.3.0.2 Task 2: Prepare the Azure Resource Manager deployment template and parameters files

1. In the **Cloud Shell** pane, click the **Upload/Download files** icon and, in the drop-down menu, click **Upload**.
2. In the **Open** dialog box, navigate to the **\allfiles\AZ-301T01\Module_01\LabFiles\Starter** folder, select the **vm-template.json** file, and click **Open**.
3. In the **Cloud Shell** pane, click the **Upload/Download files** icon and, in the drop-down menu, click **Upload**.
4. In the **Open** dialog box, navigate to the **\allfiles\AZ-301T01\Module_01\LabFiles\Starter** folder, select the **vm-template.parameters.json** file, and click **Open**.
5. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **$KEY_VAULT_ID** parameter in the **vm-template.parameters.json** parameters file with the value of the **$KEY_VAULT_ID** variable:

   ```
   sed -i.bak1 's/"$KEY_VAULT_ID"/"'"$KEY_VAULT_ID_REGEX"'"/' ~/vm-template.parameters.json
   ```

6. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that the placeholder was successfully replaced in the parameters file:

   ```
   cat ~/vm-template.parameters.json
   ```

2.3.0.3 Task 3: Configure a key vault for deployment of Azure Resource Manager templates

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0901-RG**.
3. On the **AADesignLab0901-RG** blade, click the entry representing the key vault you created in the previous exercise.
4. On the key vault blade, click **Access policies**.
5. On the **Access policies** blade, under the **Enable access to:** area, select the **Azure Resource Manager for template deployment** checkbox.
6. Click the **Save** button at the top of the pane.

2.3.0.4 Task 4: Deploy a Linux VM with the password parameter set by using a key vault secret.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy the Azure Resource Manager template with the specified parameters file:

   ```
   az deployment group create --resource-group $RESOURCE_GROUP --template-file ~/vm-template.json --pa
   ```

2. Wait for the deployment to complete before you proceed to the next task.

2.3.0.5 Task 5: Verify the outcome of the deployment

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group that contains the newly deployed Azure VM:

   ```
   RESOURCE_GROUP='AADesignLab0901-RG'
   ```

2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to retrieve the name of the Azure key vault containing the secret that stores the value of the password of the local Administrator account:

   ```
   KEY_VAULT_NAME=$(az keyvault list --resource-group $RESOURCE_GROUP --query "[0].name" --output tsv)
   ```

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to retrieve the value of the secret:

   ```
   az keyvault secret show --vault-name $KEY_VAULT_NAME --name vmPassword --query value --output tsv
   ```

4. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to retrieve the public IP address of the Azure VM you deployed in the previous task:

```
PUBLIC_IP=$(az network public-ip list --resource-group $RESOURCE_GROUP --query "[0].ipAddress" --o
```

5. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to connect to the Azure VM via SSH:

```
ssh Student@$PUBLIC_IP
```

6. At the **Cloud Shell** command prompt, when prompted whether you want to continue connecting, type `yes` and press **Enter**.
7. At the **Cloud Shell** command prompt, when prompted for password, type the value of the secret you retrieved earlier in this task and press **Enter**.

   Note: The cursor will not move when you type in the password.
8. Verify that you successfully authenticated.
9. At the **Cloud Shell** command prompt, type `exit` to log out from the Azure VM.

Review: In this exercise, you deployed a Linux VM using a password stored as a key vault secret.

2.4 Exercise 3: Remove lab resources

2.4.0.1 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,'AADesignLab09')]".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.

2.4.0.2 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,'AADesignLab09')]".name --output tsv | xargs -L1 bash -c
```

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

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

3 Integrating SaaS Services Available on the Azure Platform

4 Lab Answer Key: Deploying Service Instances as Components of Overall Azure Solutions

4.1 Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
   - Username: **Admin**
   - Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
   - Microsoft Edge
   - File Explorer
   - Visual Studio Code
   - Microsoft Azure Storage Explorer

- Bash on Ubuntu on Windows
- Windows PowerShell

  Note: You can also find shortcuts to these applications in the **Start Menu**.

4.2 Exercise 1: Deploy Function App and Cognitive Service using ARM Template

4.2.0.1 Task 1: Open the Azure portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** (https://portal.azure.com).
3. When prompted, authenticate with a user account account that has the owner role in the Azure subscription you will be using in this lab.

4.2.0.2 Task 2: Deploy Cognitive Service using an Azure Resource Manager template

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Template Deployment** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Template Deployment (deploy using custom templates)**.
4. On the **Template deployment (deploy using custom templates)** blade, click the **Create** button.
5. On the **Custom deployment** blade, click the **Build your own template in the editor** link.
6. On the **Edit template** blade, click **Load file**.
7. In the **Choose File to Upload** dialog box, navigate to the **\allfiles\AZ-301T01\Module_02\LabFiles\Starter** folder, select the **cognitive-template.json** file, and click **Open**. This will load the following content into the template editor pane:

```
{
    "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "variables": {
        "serviceName": "[concat('cgnt', uniqueString(resourceGroup().id))]"
    },
    "resources": [
        {
            "apiVersion": "2017-04-18",
            "type": "Microsoft.CognitiveServices/accounts",
            "name": "[variables('serviceName')]",
            "kind": "TextAnalytics",
            "location": "[resourceGroup().location]",
            "sku": {
                "name": "S1"
            },
            "properties": {}
        }
    ],
    "outputs": {
        "cognitiveEndpointUrl": {
            "type": "string",
            "value": "[reference(variables('serviceName')).endpoint]"
        },
        "cognitiveEndpointKey": {
            "type": "string",
            "value": "[listKeys(variables('serviceName'), '2017-04-18').key1]"
        }
    }
}
```

8. Click the **Save** button to persist the template.
9. Back on the **Custom deployment** blade, perform the following tasks:
    - Leave the **Subscription** drop-down list entry set to its default value.
    - In the **Resource group** section, ensure that the **Create new** option is selected and then, in the text box, type **AADesignLab1001-RG**.
    - In the **Location** drop-down list, select the Azure region to which you intend to deploy resources in this lab.
    - In the **Terms and Conditions** section, select the **I agree to the terms and conditions stated above** checkbox.
    - Click the **Purchase** button.
10. Wait for the deployment to complete before you proceed to the next step.
11. In the hub menu of the Azure portal, click **Resource groups**.
12. On the **Resource groups** blade, click **AADesignLab1001-RG**.
13. On the **AADesignLab1001-RG** blade, locate the **Deployments** header at the top of the blade and click the below the **Deployments** label, which indicates the number of successful deployments.
14. On the deployments blade, click the name of the most recent deployment.
15. On the **Microsoft.Template - Overview** blade, click **Outputs**.
16. On the **Microsoft.Template - Outputs** blade, identify the values of **cognitiveEndpointUrl** and **cognitiveEndpointKey** outputs. Record these values, since you will need them later in the lab.

4.2.0.3 Task 3: Deploy a function app

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Function App** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Function App**.
4. On the **Function App** blade, click the **Create** button.
5. On the **Basics** tab of the **Function App** blade, specify the following and click **Next: Hosting >**:
    - Subscription: the name of the Azure subscription you used in the previous task
    - Resource group: **AADesignLab1001-RG**.
    - Function App name: a globally unique name
    - Publish: **Code**
    - Runtime stack: **.NET Core**
    - Region: the Azure region to which you deployed an instance of Cognitive Service in the previous task
6. On the **Hosting** tab of the **Function App** blade, specify the following and click **Next: Monitoring >**:
    - Storage account: accept the default value of the Storage Account name.
    - Operating System: **Windows**
    - Plan type: **Consumption (Serverless)**
7. On the **Monitoring** tab of the **Function App** blade, specify the following and click **Review + create**:
    - Enable Application Insights: **No**.
8. On the **Review + create** tab of the **Function App** blade, click **Create**:
9. Wait for the provisioning of the function app to complete before you proceed to the next step.
10. In the hub menu of the Azure portal, click **Resource groups**.
11. On the **Resource groups** blade, click **AADesignLab1001-RG**.

12. On the **AADesignLab1001-RG** blade, in the list of resources, click the newly provisioned function app.
13. On the Azure function blade, under **Settings** click on **Configuration** at the left.
14. On the **Application settings** tab, click the **+ New application setting** link, perform the following tasks, and click **OK**:
    - In the **Name** text box, type **EndpointUrl**
    - In the **Value** text box, enter the value of **cognitiveEndpointUrl**
    - Leave the **Deployment slot setting** checkbox cleared.
15. In the **Application Settings** section, click the **+ New application setting** link again, perform the following tasks, and click **OK**:
    - In the **Name** text box, type **EndpointKey**.
    - In the **Value** text box, type the value of **cognitiveEndpointKey** you identified earlier.
    - Leave the **Deployment slot setting** checkbox cleared.
16. Click the **Save** button at the top of the **Application settings** tab.
17. In the **Deployment** section, select the **Deployment Center** entry.
18. On the **Deployment Center** blade, scroll down to the bottom of the blade and click **External** and then click **Continue**.
19. Click **App Service build service** and click **Continue**.
20. Once the **Code** section is displayed, perform the following tasks
    - In the **Repository** text box, type **https://github.com/polichtm/cognitive-services-function**.
    - In the **Branch** text box, type **master**.

      Note: The Branch field is case sensitive.
    - Set the value of **Repository Type** to **Git**.
    - Set the value of **Private Repository** to **No**.
    - Click the **Continue** button.
21. Click **Finish** and wait for the deployment to complete before you proceed to the next task.

    Note: You will be able to determine that the first deployment has completed by monitoring the **Deployments** tab. This tab updates automatically.

4.2.0.4 Task 4: Test a function app using Cognitive Services

1. On the function app blade, in the **Functions** section, click **Functions**.
2. Select the **DetermineLanguage** function from the list of functions.
3. On the left, under **Developer**, select the **Code + Test** option.
4. On the **DetermineLanguage | Code + Test** blade, click **Test/Run**.
5. On the **Input** tab, ensure that the **HTTP method** is set to **POST**, scroll down to the **Body** section, and set its content to the following JSON formatted text:

```
{
    "text": "I stuffed a shirt or two into my old carpet-bag, tucked it under my arm, and started
}
```

6. Click the **Run** button.
7. Switch to the **Output** tab and review its content. The output should identify the language as **en** (English).

Review: In this exercise, you created a function app that uses Azure Cognitive Services.

4.3 Exercise 2: Create a Logic App that uses a Function App

4.3.0.1 Task 1: Create a logic app

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Logic App** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Logic App**.
4. On the **Logic App** blade, click the **Create** button.
5. On the **Create logic app** blade, perform the following tasks:
   - Leave the **Subscription** drop-down list entry set to its default value.
   - In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab1001-RG**.
   - In the **Name** text box, enter the value **CognitiveWorkflow**.
   - In the **Location** drop-down list, select the same Azure region you chose in the previous exercise of this lab.
   - In the **Log Analytics** section, ensure that the **Off** button is selected.
   - Click the **Review + create** button and then click the **Create** button.
6. Wait for the provisioning to complete before you proceed to the next task.

4.3.0.2 Task 2: Configure logic app steps

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab1001-RG**.
3. On the **AADesignLab1001-RG** blade, click the entry representing the logic app you created in the previous task.
4. On the **Logic Apps Designer** blade, scroll down and click the **Blank Logic App** tile in the **Templates** section.
5. On the **Logic Apps Designer** blade, click the **Code view** button at the top of the pane.
6. On the **Logic Apps Designer** blade, review the blank Logic App JSON template:

```
{
    "definition": {
        "$schema": "https://schema.management.azure.com/providers/Microsoft.Logic/schemas/2016-06-
        "actions": {},
        "contentVersion": "1.0.0.0",
        "outputs": {},
        "triggers": {}
    },
    "parameters": {}
}
```

7. Replace the default JSON template with the following template that includes an HTTP trigger (**\\allfiles\AZ-301T01\Module_02\LabFiles\Starter\logic-app.json**) and save your changes:

```
{
    "definition": {
        "$schema": "https://schema.management.azure.com/providers/Microsoft.Logic/schemas/2016-06-
        "actions": {},
        "contentVersion": "1.0.0.0",
        "outputs": {},
        "triggers": {
            "manual": {
                "inputs": {
                    "method": "POST",
```

```
                    "schema": {
                        "properties": {
                            "text": {
                                "type": "string"
                            }
                        },
                        "type": "object"
                    }
                },
                "kind": "Http",
                "type": "Request"
            }
        }
    },
    "parameters": {}
}
```

8. On the **Logic Apps Designer** blade, click the **Designer** button.

   Note: At this point, you should see a single step in the designer. This is the ”trigger” step that begins a workflow.

9. Click the + **New Step** button in the designer.
10. In the **Choose an action** section, perform the following tasks:
    - In the search text box, type **Azure Functions**.
    - In the search results, select the action named **Choose an Azure function**.
    - In the next set of search results, select the Azure Function instance you created in the previous exercise of this lab.
    - In the final set of search results, select the **DetermineLanguage** function that will be used for the action.
11. In the **DetermineLanguage** step, perform the following tasks:
    - In the **Request Body** text box, type **@triggerBody()**.
    - In the **Add new parameter** drop-down list, select the **Method** checkbox.
    - In the **Method** drop-down list, select the **POST** option.
12. Click the + **New Step** button in the designer.
13. In the **Choose an action** dialog that displays, perform the following tasks:
    - In the search text box, type **Azure Functions**.
    - In the search results, select the action named **Choose an Azure function**.
    - In the next set of search results, select the Azure Function instance you created in the previous exercise of this lab.
    - In the final set of search results, select the **DetermineKeyPhrases** function that will be used for the action.
14. In the **DetermineKeyPhrases** step, perform the following tasks:
    - In the **Request Body** text box, enter the value **@body('DetermineLanguage')**.
    - In the **Add new parameter** drop-down list, select the **Method** checkbox.
    - In the **Method** drop-down list, select the **POST** option.
15. Click the + **New Step** button in the designer.
16. In the **Choose an action** dialog that displays, perform the following tasks:
    - In the search text box, type **Response**.
    - In the search results, select the **Action** named **Response Request**.

17. In the **Response** step, perform the following tasks:
    - In the **Status Code** text box, ensure that the value **200** is specified.
    - Leave **Headers** entries unchanged
    - In the **Body** text box, type **@body('DetermineKeyPhrases')**.
18. At the top of the **Logic Apps Designer** blade, click the **Save** button to persist your workflow.
19. Scroll to the top of the **Logic Apps Designer** area and click the **When a HTTP request is received** step.
20. Copy the value of the **HTTP POST URL** text box. This URL will be used later in this lab.

4.3.0.3 Task 2: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.

   Note: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.

2. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.

   Note: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.

3. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:
    - Leave the **Subscription** drop-down list entry set to its default value.
    - In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you deployedf resources in this lab
    - In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab1001-RG**.
    - In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
    - In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
    - Click the **Create storage** button.
4. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

4.3.0.4 Task 3: Validate Logic App using Python

1. At the **Cloud Shell** command prompt at the bottom of the portal, type the following command and press **Enter** to open the interactive **python** terminal:

   ```
   python
   ```

2. At the **Cloud Shell** command prompt at the bottom of the portal, type the following command and press **Enter** to import the **requests** library:

   ```
   import requests
   ```

3. At the **Cloud Shell** command prompt at the bottom of the portal, type the following command (replacing the placeholder `<logic app POST Url>` with the value of your url recorded earlier in this lab) and press **Enter** to create a variable containing the value of your logic app's url :

   ```
   url = "<logic app POST Url>"
   ```

4. At the **Cloud Shell** command prompt at the bottom of the portal, type the following command and press **Enter** to send an HTTP POST request to trigger your logic app workflow:

```
response = requests.post(url, json={'text': 'Circumambulate the city of a dreamy Sabbath afternoon
```

5. At the **Cloud Shell** command prompt at the bottom of the portal, type the following command and press **Enter** to display the output of the Logic App workflow:

   ```
   print(response.status_code, response.reason, response.text)
   ```

6. Close the **Cloud Shell** pane.

   Review: In this exercise, you created a logic app that leverages the function app created in the previous exercise of this lab.

4.4 Exercise 3: Remove lab resources

4.4.0.1 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,'AADesignLab10')]".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.

4.4.0.2 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,'AADesignLab10')]".name --output tsv | xargs -L1 bash -c
   ```

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

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

5 Comparing Database Options in Azure

6 Lab Answer Key: Deploying Database Instances in Azure

6.1 Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
   - Username: **Admin**
   - Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
   - Microsoft Edge
   - File Explorer
   - Visual Studio Code
   - Microsoft Azure Storage Explorer
   - Bash on Ubuntu on Windows
   - Windows PowerShell

   Note: You can also find shortcuts to these applications in the **Start Menu**.

6.2 Exercise 1: Deploy a Cosmos DB database

6.2.0.1 Task 1: Open the Azure Portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** (https://portal.azure.com).
3. If prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

6.2.0.2 Task 2: Create a Cosmos DB database and collection

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Cosmos DB** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Azure Cosmos DB**.
4. On the **Azure Cosmos DB** blade, click the **Create** button.
5. On the new **Azure Cosmos DB** blade, perform the following tasks:
   - Leave the **Subscription** drop-down list entry set to its default value.
   - Resource group: ensure that the **Create new** option is selected and then, in the text box, type **AADesignLab0701-RG**.
   - In the **Account Name** text box, type a globally unique value.
   - In the **API** drop-down list, select the **Core (SQL)** option.
   - In the **Location** drop-down list, select the Azure region in which you want to deploy resources in this lab.
   - Leave all remaining settings with their default values.
   - Click the **Review + create** button and then click the **Create** button.
6. Wait for the provisioning to complete before you proceed to the next step.

   Note: The deployment could take up to 15 minutes.
7. Navigate to the blade of the newly created Cosmos DB account and click **Keys**.
8. On the Cosmos DB account Keys blade, note the value of the **PRIMARY CONNECTION STRING**. You will need it in the third exercise of this lab.
9. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.

   Note: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.
10. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.

    Note: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.
11. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:
    - Leave the **Subscription** drop-down list entry set to its default value.
    - In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location that you selected earlier in this task.
    - In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab0701-RG**.

- In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
- In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
- Click the **Create storage** button.

12. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.
13. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group that contains the Azure Cosmos DB account you deployed earlier in this task:

    ```
    RESOURCE_GROUP='AADesignLab0701-RG'
    ```

14. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the CosmosDB account you created earlier in this task:

    ```
    COSMOSDB_NAME=$(az cosmosdb list --resource-group $RESOURCE_GROUP --query "[0].name" --output tsv)
    ```

15. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the primary key of the CosmosDB account you created earlier in this task:

    ```
    PRIMARY_KEY=$(az cosmosdb keys list --resource-group $RESOURCE_GROUP --name $COSMOSDB_NAME --outpu
    ```

16. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the URI of the CosmosDB account you created earlier in this task:

    ```
    URI="https://$COSMOSDB_NAME.documents.azure.com:443/"
    ```

17. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new CosmosDB database named **FinancialClubDatabase**:

    ```
    az cosmosdb database create --url-connection $URI --key $PRIMARY_KEY --db-name 'FinancialClubDataba
    ```

18. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a fixed collection named **MemberCollection** in the newly created database:

    ```
    az cosmosdb collection create --url-connection $URI --key $PRIMARY_KEY --db-name 'FinancialClubDat
    ```

19. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to display the value of the PRIMARY_KEY variable:

    ```
    echo $PRIMARY_KEY
    ```

20. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to display the value of the URI variable:

    ```
    echo $URI
    ```

    Note: Take a note of these values - you will need them in the third exercise of this lab.

6.2.0.3 Task 3: Create and query documents in Cosmos DB

1. On the left side of the Azure Cosmos DB account blade, click **Data Explorer**.
2. In the **Data Explorer** pane, if necessary, refresh the pane and then click the **MemberCollection** child node of the **FinancialClubDatabase** node.
3. Click the **New SQL Query** button at the top of the **Data Explorer** pane.
4. In the **Query 1** tab that opened, view the default query:

    ```
    SELECT * FROM c
    ```

5. Click the **Execute Query** button at the top of the query editor and verify that the query does not return any results.
6. In the left pane of the Data Explorer, expand the **MemberCollection** node.
7. Click the **Items** child node within the **MemberCollection** node.
8. In the new **Items** tab that opened, click the **New Item** button at the top of the tab.

9. In the **Items** tab, replace the existing document with the following document:

```
{
    "firstName": "Pennington",
    "lastName": "Oneal",
    "age": 26,
    "salary": 90000.00,
    "company": "Veraq",
    "isVested": false
}
```

10. Click the **Save** button at the top of the **Items** tab (you might need to first click the ellipsis toolbar button).
11. In the **Items** tab, click the **New Item** button at the top of the tab.
12. In the **Items** tab, replace the existing document with the following document:

```
{
    "firstName": "Suzanne",
    "lastName": "Oneal",
    "company": "Veraq"
}
```

13. Click the **Save** button at the top of the **Items** tab.
14. Switch back to the **Query 1** tab, re-run the default query `SELECT * FROM c` by clicking the **Execute Query** button at the top of the query editor, and review the results.
15. In the query editor, replace the default query with the following query:

```
SELECT
    c.id,
    c.firstName,
    c.lastName,
    c.isVested,
    c.company
FROM
    c
WHERE
    IS_DEFINED(c.isVested)
```

16. Click the **Execute Query** button at the top of the query editor and review the results.
17. In the query editor, replace the existing query with the following query:

```
SELECT
    c.id,
    c.firstName,
    c.lastName,
    c.age
FROM
    c
WHERE
    c.age > 20
```

18. Click the **Execute Query** button at the top of the query editor and review the results.
19. In the query editor, replace the existing query with the following query:

```
SELECT VALUE
    c.id
FROM
    c
```

20. Click the **Execute Query** button at the top of the query editor and review the results.
21. In the query editor, replace the existing query with the following query:

```
SELECT VALUE {
    "badgeNumber": SUBSTRING(c.id, 0, 8),
    "company": c.company,
    "fullName": CONCAT(c.firstName, " ", c.lastName)
} FROM c
```

22. Click the **Execute Query** button at the top of the query editor and review the results.

    Review: In this exercise, you created a new Cosmos DB account, database, and collection, added sample items to the collection, and run sample queries targeting these items.

6.3 Exercise 2: Deploy Application using Cosmos DB

6.3.0.1 Task 1: Deploy API App code using Azure Resource Manager templates and GitHub

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Template Deployment** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Template Deployment**.
4. On the **Template deployment** blade, click the **Create** button.
5. On the **Custom deployment** blade, click the **Build your own template in the editor** link.
6. On the **Edit template** blade, click the **Load file** link.
7. In the **Open** file dialog that appears, navigate to the **\allfiles\AZ-301T02\Module_02\LabFiles\Starter** folder.
8. Select the **api.json** file.
9. Click the **Open** button.
10. Back on the **Edit template** blade, click the **Save** button to persist the template.
11. Back on the **Custom deployment** blade, perform the following tasks:
    - Leave the **Subscription** drop-down list entry set to its default value.
    - In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab0701-RG**.
    - In the **Terms and Conditions** section, click the **I agree to the terms and conditions stated above** checkbox.
    - Click the **Purchase** button.
12. Wait for the deployment to complete before you proceed to the next task.

    Note: Deployment from source control can take up to 10 minutes.

6.3.0.2 Task 2: Validate API App

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0701-RG**.
3. On the **AADesignLab0701-RG** blade, click the entry representing the newly created App Service API app.
4. On the API app blade, under **Settings**, click **Configuration**.
5. On the Application Settings blade, scroll down to the **Application settings** section and perform the following tasks:
    - Set the value of the **CosmosDB:AuthorizationKey** setting to the value of the **PRIMARY KEY** setting of the **Cosmos DB** account you created earlier in this lab.
    - Update the value of the **CosmosDB:EndpointUrl** setting to the value of the **URI** setting of the **Cosmos DB** instance you created earlier in this lab.
    - Click the **Save** button at the top of the pane (if prompted, click **Continue**).

6. On the left-side of the API app blade, click **Overview**.
7. Click the **Restart** button at the top of the blade and, when prompted to confirm, click **Yes**.
8. Click the **Browse** button at the top of the blade. This will open a new browser tab displaying the **Swagger UI** homepage.

   Note: If you click the **Browse** button before the API app has fully restarted, you may not be able to follow the remaining steps in this task. If this happens, refresh your browser until the API app is running again.
9. On the **Swagger UI** homepage, click **GET/Documents**.
10. Click the **Try it out!** button.
11. Review the results of the request (the results should include 2 items).
12. Back on the **Swagger UI** homepage, click **POST/Populate**.
13. In the **Parameters** section, in the **Value** field for the **options** parameter, paste in the following JSON content:

```
{
    "quantity": 50
}
```

14. In the **Response Messages** section, click the **Try it out!** button.
15. Review the results of the request (the results should include 50 items).
16. Back on the **Swagger UI** homepage, click **GET/Documents**.
17. Locate the **Response Messages** section. Click the **Try it out!** button.
18. Review the results of the request (the results should include 52 items).
19. Close the new browser tab and return to the browser tab displaying the Azure portal.

    Review: In this exercise, you created a new API App that uses the .NET Core DocumentDB SDK to connect to Azure Cosmos DB collection and manage its documents.

6.4 Exercise 3: Connect Cosmos DB to Azure Search

6.4.0.1 Task 1: Create Azure Cognitive Search Instance

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Search** and press **Enter**.
3. On the **Showing All Results** blade, in the search results, click **Azure Cognitive Search**.
4. On the **Azure Cognitive Search** blade, click the **Create** button.
5. On the **New Search Service** blade, perform the following tasks:
   - Leave the **Subscription** drop-down list entry set to its default value.
   - In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab0701-RG**.
   - In the **URL** text box, enter a globally unique name. Record its value. You will use it later in this lab.
   - In the **Location** drop-down list, select the Azure region matching or near the location where you deployed Cosmos DB resource earlier in this labb
   - Click **Change Pricing Tier**.
   - On the **Select Pricing Tier** blade, click **Free** and then click the **Select** button.
   - Click the **Review + create** button, review the settings then click **Create**.
6. Wait for the provisioning to complete before you proceed to the next step.
7. In the hub menu in the Azure portal, click **Resource groups**.

8. On the **Resource groups** blade, click **AADesignLab0701-RG**.
9. On the **AADesignLab0701-RG** blade, click the entry representing the newly created Azure Search instance.
10. On the Search service blade, click **Keys**.
11. In the **Keys** pane, record the value of **QUERY KEY**. You will use it later in this lab.

    Note: The query key is located below the primary and secondary keys, and does not have a name by default.

6.4.0.2 Task 2: Index Cosmos DB Data in Azure Search

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0701-RG**.
3. On the **AADesignLab0701-RG** blade, click the entry representing the Azure Search instance you created earlier in this lab.
4. On the **Overview** blade of Azure Search service, click **Import data**.
5. On the **Connect to your data** tab, perform the following tasks:
   - In the **Data Source** drop down list, select **Cosmos DB**.
   - In the **Name** text box, type **cosmosdata**.
   - In the **Cosmos DB account** text box, type the Cosmos DB account connection string you identfied earlier in this lab.
   - In the **Database** drop-down list, select the **FinancialClubDatabase** entry.
   - in the **Collection** drop-down list, select the **MemberCollection** entry.
   - In the **Query** field, enter the following SQL query:

```
SELECT
    c.id,
    c.firstName,
    c.lastName,
    c.age,
    c.salary,
    c.company,
    c.isVested,
    c._ts
FROM
    c
WHERE
    c._ts >= @HighWaterMark
ORDER BY c._ts
```

   - Ensure that the **Query results ordered by __ts** checkbox is selected.
   - Click the **Next: Add cognitive skills (optional)** button.
6. On the **Cognitive Search** blade, click the **Skip to: Customize target index** button.
7. On the **Customize target index** blade, perform the following tasks:
   - In the **Index name** text box, type **memberindex**.
   - In the **Key** drop-down list, ensure that the **id** entry is selected.
   - For the **id** field in the table, ensure that the **RETRIEVABLE**, **FILTERABLE**, and **SORTABLE** checkboxes are selected.
   - For the **firstName** field in the table, ensure that the **RETRIEVABLE**, **SORTABLE**, and **SEARCHABLE** options are selected.
   - For the **lastName** field in the table, ensure that the **RETRIEVABLE**, **SORTABLE**, and **SEARCHABLE** checkboxes are selected.

- For the **age** field in the table, ensure that the **RETRIEVABLE**, **FILTERABLE**, **SORTABLE**, and **FACETABLE** checkboxes are selected.
- For the **salary** field in the table, ensure that the **RETRIEVABLE**, **FILTERABLE**, **SORTABLE**, and **FACETABLE** checkboxes are selected.
- For the **company** field in the table, ensure that the **RETRIEVABLE**, **FACETABLE**, and **SEARCHABLE** checkboxes are selected.
- For the **isVested** field in the table, ensure that the **RETRIEVABLE**, **FILTERABLE**, **SORTABLE**, **FACETABLE** checkboxes are selected.
- Click the **Next: Create an indexer** button.

8. On the **Create an Indexer** blade, perform the following tasks:
   - In the **Name** text box, type **cosmosmemberindexer**.
   - In the **Schedule** section, select the **Custom** option.
   - In the **Interval (minutes)** text box, type **5**.
   - In the **Start time (UTC)** field, specify the current date and accept the default value of the time entry.
   - Ensure that the **Track deletions** checkbox is clear and click the **Submit** button.

6.4.0.3 Task 3: Validate API App

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0701-RG**.
3. On the **AADesignLab0701-RG** blade, click the entry representing the App Service API app you created earlier in this lab.
4. On the API app blade, click **Configuration**.
5. On the Application settings blade, scroll down to the **Application settings** section and perform the following tasks:
   - Set the value of the **Search:AccountName** setting to the name of the Azure Search instance you created earlier in this lab.
   - Set the value of the **Search:QueryKey** setting to the value of the **QUERY KEY** of the Azure Search instance you created earlier in this lab.
   - Set the value of the **Search:IndexId** setting to the value **memberindex**.
   - Click the **Save** button at the top of the blade (if prompted, click **Continue**).
6. On the API app blade, click **Overview**.
7. Click the **Restart** button at the top of the blade and, when prompted to confirm, click **Yes**.
8. Click the **Browse** button at the top of the blade. This will open a new browser tab displaying the **Swagger UI** homepage.

   Note: If you click the **Browse** button before the API app has fully restarted, you may not be able to follow the remaining steps in this task. If this happens, refresh your browser until the API app is running again.
9. On the **Swagger UI** homepage, click **Cosmos DB API v.1.0.0** at the top of the page and select the **Cosmos DB API v.2.0.0** option from the drop-down list.
10. Click **GET/Documents/search**.
11. In the **Parameters** section, in the **Value** text box of the **query** parameter, type the following text:

    ```
    Oneal
    ```
12. In the **Response Messages** section, click the **Try it out!** button.
13. Review the results of the request (the results should include 2 items).

14. In the **Parameters** section, in the **Value** text box of the **query** parameter, type the following text:

    ```
    penn*
    ```

15. In the **Response Messages** section, click the **Try it out!** button.
16. Review the results of the request (the results should include 1 item).
17. Close the new browser tab and return to the browser tab displaying the Azure portal.

    Review: In this exercise, you created an Azure Search instance that uses an indexer to index the documents in Azure Cosmos DB.

6.5 Exercise 4: Remove lab resources

6.5.0.1 Task 1: Delete the resource group

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0701-RG**.
3. On the **AADesignLab0701-RG** blade, click **Delete resource group**.
4. In the **Are you sure you want to delete "AADesignLab0701-RG"?** pane, in the **TYPE THE RESOURCE GROUP NAME** text box, type **AADesignLab0701-RG** and click **Delete**.

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

7 Monitoring and automating Azure solutions

8 Lab Answer Key: Deploying Configuration Management solutions to Azure

8.1 Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
   - Username: **Admin**
   - Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
   - Microsoft Edge
   - File Explorer
   - Visual Studio Code
   - Microsoft Azure Storage Explorer
   - Bash on Ubuntu on Windows
   - Windows PowerShell

     Note: You can also find shortcuts to these applications in the **Start Menu**.

8.2 Exercise 1: Deploy compute resources

8.2.0.1 Task 1: Open the Azure portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** (https://portal.azure.com).
3. When prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

8.2.0.2 Task 2: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.

   Note: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.

2. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.

   Note: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.

3. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:

   - Leave the **Subscription** drop-down list entry set to its default value.
   - In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you intend to deploy resources in this lab.
   - In the **Resource group** section, select the **Create New** option and then, in the text box, type **AADesignLab1201-RG**.
   - In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   - In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
   - Click the **Create storage** button.

4. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

8.2.0.3 Task 3: Deploy a Linux VM

1. In the **Cloud Shell** pane, click the **Upload/Download files** icon and, in the drop-down menu, click **Upload**.

2. In the **Open** dialog box, navigate to the **\allfiles\AZ-301T02\Module_03\LabFiles\Starter** folder, select the **linux-template.json** file, and click **Open**. The file contains the following template:

```
{
"$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
"contentVersion": "1.0.0.0",
"parameters": {
    "userName": {
        "type": "string",
        "defaultValue": "Student"
    },
    "password": {
        "type": "securestring"
    }
},
"variables": {
    "vmName": "[concat('lvm', uniqueString(resourceGroup().id))]",
    "nicName": "[concat('nic', uniqueString(resourceGroup().id))]",
    "publicIPAddressName": "[concat('pip', uniqueString(resourceGroup().id))]",
    "virtualNetworkName": "[concat('vnt', uniqueString(resourceGroup().id))]",
    "subnetName": "Linux",
    "imageReference": {
        "publisher": "OpenLogic",
        "offer": "CentOS",
        "sku": "7.5",
        "version": "latest"
```

```
    }
},
"resources": [
    {
        "apiVersion": "2017-06-01",
        "type": "Microsoft.Network/publicIPAddresses",
        "name": "[variables('publicIPAddressName')]",
        "location": "[resourceGroup().location]",
        "properties": {
            "publicIPAllocationMethod": "Dynamic"
        }
    },
    {
        "apiVersion": "2017-06-01",
        "type": "Microsoft.Network/virtualNetworks",
        "name": "[variables('virtualNetworkName')]",
        "location": "[resourceGroup().location]",
        "properties": {
            "addressSpace": {
                "addressPrefixes": [
                    "10.0.0.0/16"
                ]
            },
            "subnets": [
                {
                    "name": "[variables('subnetName')]",
                    "properties": {
                        "addressPrefix": "10.0.0.0/24"
                    }
                }
            ]
        }
    },
    {
        "apiVersion": "2017-10-01",
        "type": "Microsoft.Network/networkInterfaces",
        "name": "[variables('nicName')]",
        "location": "[resourceGroup().location]",
        "dependsOn": [
            "[resourceId('Microsoft.Network/publicIPAddresses/', variables('publicIPAddressName'))]
            "[resourceId('Microsoft.Network/virtualNetworks/', variables('virtualNetworkName'))]"
        ],
        "properties": {
            "ipConfigurations": [
                {
                    "name": "ipconfig1",
                    "properties": {
                        "privateIPAllocationMethod": "Dynamic",
                        "publicIPAddress": {
                            "id": "[resourceId('Microsoft.Network/publicIPAddresses', variables('pu
                        },
                        "subnet": {
                            "id": "[concat(resourceId('Microsoft.Network/virtualNetworks',variables
                        }
                    }
                }
            ]
        }
    },
    {
        "apiVersion": "2017-03-30",
```

```
            "type": "Microsoft.Compute/virtualMachines",
            "name": "[variables('vmName')]",
            "location": "[resourceGroup().location]",
            "dependsOn": [
                "[resourceId('Microsoft.Network/networkInterfaces/', variables('nicName'))]"
            ],
            "properties": {
                "hardwareProfile": {
                    "vmSize": "Standard_D2s_v3"
                },
                "osProfile": {
                    "computerName": "[variables('vmName')]",
                    "adminUsername": "[parameters('username')]",
                    "adminPassword": "[parameters('password')]"
                },
                "storageProfile": {
                    "imageReference": "[variables('imageReference')]",
                    "osDisk": {
                        "createOption": "FromImage"
                    }
                },
                "networkProfile": {
                    "networkInterfaces": [
                            {
                             "id": "[resourceId('Microsoft.Network/networkInterfaces', variables('nicNa
                            }
                        ]
                    }
                }
            }
        ]
    }
```

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group that will contain the virtual virtual machine:

   ```
   RESOURCE_GROUP='AADesignLab1202-RG'
   ```

4. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all region names to choose.

   ```
   az account list-locations --query "[].name" --output tsv
   ```

5. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment (replace the placeholder `<Azure region>` with the name of the Azure region from one of the list in previous **Cloud Shell** output. to which you intend to deploy resources in this lab):

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

6. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new resource group:

   ```
   az group create --name $RESOURCE_GROUP --location $LOCATION
   ```

7. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy the Azure Resource Manager template with the specified parameters file:

   ```
   az deployment group create --resource-group $RESOURCE_GROUP --template-file ~/linux-template.json
   ```

8. Do not wait for the deployment to complete before you proceed to the next task.

8.2.0.4 Task 4: Deploy an Azure Automation account

1. In the upper left corner of the Azure portal, click **Create a resource**.

2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Automation** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Automation**.
4. On the **Automation** blade, click **Create**.
5. On the **Add Automation Account** blade, perform the following tasks:
   - In the **Name** text box, type **LinuxAutomation**.
   - Leave the **Subscription** drop-down list entry set to its default value.
   - In the **Resource group** section, select the **Create new** option and then, in the text box, type **AADesignLab1203-RG**.
   - In the **Location** drop-down list, select the Azure region matching or near the location where you deployed the Azure VM in the previous task.
   - In the **Create Azure Run As account** section, ensure that **Yes** option is selected.
   - Click the **Create** button.
6. Wait for the provisioning to complete before you proceed to the next task.

   Review: In this exercise, you created a Linux VM using an Azure Resource Manager template and provisioned an Azure Automation account from the Azure portal.

8.3 Exercise 2: Configure Azure Automation DSC

8.3.0.1 Task 1: Import Linux PowerShell DSC modules

1. In the hub menu of the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab1203-RG**.
3. On the **AADesignLab1203-RG** blade, click the newly created Azure Automation account.
4. On the **LinuxAutomation** blade, in the **Shared Resources** section on the left side of the blade, click **Modules gallery**.
5. On the **LinuxAutomation | Modules gallery** blade, perform the following tasks:
   - In the **Search** text box, type **nx** and press **Enter**.
   - In the search results, click the **nx** module.
6. On the **nx** blade, click the **Import** button at the top of the blade.
7. On the **Import** blade, click the **OK** button.
8. Wait for the import process to finish before you proceed to the next task. A status message on the **nx Module** blade will indicate that the module was successfully imported.

   Note: This process should take about 2 minutes.

8.3.0.2 Task 2: Create Linux DSC Configuration

1. Navigate back to the **LinuxAutomation** blade.
2. Back on the **LinuxAutomation** blade, in the **Configuration Management** section, click **State configuration (DSC)**.
3. On the **LinuxAutomation | State configuration (DSC)** blade, click the **Configurations** tab.
4. On the **LinuxAutomation | State configuration (DSC)** blade, click the **+ Add** button at the top of the pane.
5. On the **Import** blade, perform the following tasks:
   - Next to the **Configuration file** field, click the blue button with a folder icon.
   - In the **Choose File to Upload** dialog box, navigate to the **\allfiles\AZ-301T02\Module_03\LabFiles\Star** folder.
   - Select the **lampserver.ps1** file.

- Click the **Open** button to close the dialog and return to the **Import** blade.
- In the **Name** text box, accept the default entry **lampserver**.
- In the **Description** text box, type **LAMP Server configuration using PHP and MySQL**.
- Click the **OK** button.

6. Back in the **DSC configurations** pane, click **Refresh** and then click the newly created **lampserver** configuration.
7. On the **lampserver Configuration** blade, click the **Compile** button at the top of the blade. In the confirmation dialog box, click **Yes** to proceed with compiling the configuration.
8. Wait for the compilation task to finish. To determine the status of the compilation task, review the **Status** column of the **Compilation jobs** section of the **lampserver Configuration** blade.

   Note: You may need to close and re-open the blade to see the latest compilation status. This blade does not refresh automatically.

8.3.0.3 Task 3: Onboard Linux VM

1. Navigate back to the **LinuxAutomation - State Configuration (DSC)** blade.
2. Back on the **LinuxAutomation | State Configuration (DSC)** blade, click the **Nodes** tab.
3. On the **LinuxAutomation | State configuration (DSC)** blade, click the **+ Add** button at the top of the pane.
4. On the **Virtual Machines** blade, click the entry representing the Linux virtual machine you deployed in the previous exercise.
5. On the virtual machine blade, click **+ Connect**.
6. On the **Registration** blade, perform the following tasks:
   - Leave the **Registration key** setting with its default value.
   - In the **Node configuration name** drop-down list, select the **lampserver.localhost** entry.
   - Leave all remaining settings with their default values.
   - Click the **OK** button.
7. Wait for the connection process to complete before you proceed to the next step.
8. Navigate back to the **LinuxAutomation | State Configuration (DSC)** blade.
9. On the **LinuxAutomation | State configuration (DSC)** blade, select in the **NODE** section the virtual machine you deployed in the previous exercise.

   Note: You may need to refresh the blade.
10. On the virtual machine blade, click **Assign node configuration**.
11. On the Assign Node Configuration blade, select the node configuration **lampserver.host** and click the **OK** button.
12. Back on the **LinuxAutomation | State Configuration (DSC)** blade, click the **Refresh** button.
13. In the list of DSC nodes, verify that the Linux virtual machine has the **Compliant** status.

    Note: You may need to wait for up to 30 minutes for the new status to be updated.

    Review: In this exercise, you created a PowerShell DSC configuration and applied the configuration to a Linux virtual machine.

8.3.0.4 Task 4: Validate Linux VM onboarding

1. In the Azure portal, open a Bash session in the **Cloud Shell**.
2. In the Bash session of the Cloud Shell run the following to identify installed packages (provide the password **Pa55w.rd1234** when prompted):

```
RESOURCE_GROUP='AADesignLab1202-RG'
PUBLIC_IP=$(az network public-ip list --resource-group $RESOURCE_GROUP --query "[0].ipAddress" --o
ssh Student@$PUBLIC_IP
sudo yum history
ps -aux | grep httpd
ps -aux | grep maria
exit
```

3. Close the **Cloud Shell** pane.

8.4 Exercise 3: Remove lab resources

8.4.0.1 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 at the bottom of the portal, 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,'AADesignLab12')]".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.

8.4.0.2 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,'AADesignLab12')]".name --output tsv | xargs -L1 bash -c
   ```

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

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

9 Deploying Resources with Azure Resource Manager

10 Lab Answer Key: Getting Started with Azure Resource Manager Templates and Azure Building Blocks

10.1 Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
   - Username: **Admin**
   - Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
   - Microsoft Edge
   - File Explorer
   - Visual Studio Code
   - Microsoft Azure Storage Explorer
   - Bash on Ubuntu on Windows
   - Windows PowerShell

   Note: You can also find shortcuts to these applications in the **Start Menu**.

10.2 Exercise 1: Deploy core Azure resources by using an Azure Resource Manager Template from the Azure portal

10.2.0.1 Task 1: Open the Azure Portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** (https://portal.azure.com).
3. If prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

10.2.0.2 Task 2: Deploy an Azure virtual network from the Azure portal by using an Azure Resource Manager template

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Template Deployment** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Template deployment**.
4. On the **Template deployment** blade, click the **Create** button.
5. On the **Custom deployment** blade, click the **Build your own template in the editor** link.
6. On the **Edit template** blade, click **Load file**.
7. In the **Choose File to Upload** dialog box, navigate to the **\allfiles\AZ-301T03\Module_01\Labfiles\Starter** folder, select the **vnet-simple-template.json** file, and click **Open**. This will load the following content into the template editor pane:

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "vnetNamePrefix": {
            "type": "string",
            "defaultValue": "vnet-",
            "metadata": {
                "description": "Name prefix of the vnet"
            }
        },
        "vnetIPPrefix": {
            "type": "string",
            "defaultValue": "10.2.0.0/16",
            "metadata": {
                "description": "IP address prefix of the vnet"
            }
        },
        "subnetNamePrefix": {
            "type": "string",
            "defaultValue": "subnet-",
            "metadata": {
                "description": "Name prefix of the subnets"
            }
        },
        "subnetIPPrefix": {
            "type": "string",
            "defaultValue": "10.2.0.0/24",
            "metadata": {
                "description": "IP address prefix of the first subnet"
            }
        }
    },
    "variables": {
```

```
        "vnetName": "[concat(parameters('vnetNamePrefix'), resourceGroup().name)]",
        "subnetNameSuffix": "0"
    },
    "resources": [
    {
        "apiVersion": "2018-02-01",
        "name": "[variables('vnetName')]",
        "type": "Microsoft.Network/virtualNetworks",
        "location": "[resourceGroup().location]",
        "scale": null,
        "properties": {
            "addressSpace": {
                "addressPrefixes": [
                    "[parameters('vnetIPPrefix')]"
                ]
            },
            "subnets": [
                {
                    "name": "[concat(parameters('subnetNamePrefix'), variables('subnetNameSuffix')
                    "properties": {
                        "addressPrefix": "[parameters('subnetIPPrefix')]"
                    }
                }
            ],
            "virtualNetworkPeerings": [],
            "enableDdosProtection": false,
            "enableVmProtection": false
        },
        "dependsOn": []
        }
    ]
}
```

8. Click the **Save** button to persist the template.
9. Back on the **Custom deployment** blade, perform the following tasks:
   - Leave the **Subscription** drop-down list entry set to its default value.
   - In the **Resource group** section, select the **Create new** option and, in the text box, type **AADesignLab0201-RG**.
   - In the **Location** drop-down list, select the Azure region to which you want to deploy resources in this lab.
   - Leave the **vnetNamePrefix** text box set to its default value.
   - Leave the **vnetIPPrefix** text box set to its default value.
   - Leave the **subnetNamePrefix** text box set to its default value.
   - Leave the **subnetIPPrefix** text box set to its default value.
   - In the **Terms and Conditions** section, select the **I agree to the terms and conditions stated above** checkbox.
   - Click the **Purchase** button.
10. Wait for the deployment to complete before you proceed to the next task.

10.2.0.3 Task 3: View deployment metadata

1. In the hub menu of the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click the entry representing the resource group to which you deployed the template in the previous task.
3. With the **Overview** selection active, on the resource group blade, click the **Deployments** link.

4. On the resulting blade, click the latest deployment to view its metadata in a new blade.
5. Within the deployment blade, observe the information displayed in the **Operation details** section.

Review: In this exercise, you deployed an Azure virtual network by using an Azure Resource Manager template from the Azure portal

10.3 Exercise 2: Deploy core Azure resources by using Azure Building Blocks from the Azure Cloud Shell

10.3.0.1 Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.

   Note: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.

2. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.

   Note: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.

3. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:
   - Leave the **Subscription** drop-down list entry set to its default value.
   - In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you deployed resources in this lab
   - Resource group: ensure that the **Use Existing** option is selected and select **AADesignLab0201-RG**.
   - In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   - In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
   - Click the **Create storage** button.
4. Wait for the **Cloud Shell** to finish its first-time setup procedures before you continue to the next task.

10.3.0.2 Task 2: Install the Azure Building Blocks npm package in Azure Cloud Shell

1. At the **Cloud Shell** command prompt at the bottom of the portal, type in the following command and press **Enter** to create a local directory to install the Azure Building Blocks npm package:

   ```
   mkdir ~/.npm-global
   ```

2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to update the npm configuration to include the new local directory:

   ```
   npm config set prefix '~/.npm-global'
   ```

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to open the ~./bashrc configuration file for editing:

   ```
   vi ~/.bashrc
   ```

4. At the **Cloud Shell** command prompt, in the vi editor interface, scroll down to the bottom of the file (or type **G**), scroll to the right to the right-most character on the last line (or type **$**), type **a** to enter the **INSERT** mode, press **Enter** to start a new line, and then type the following to add the newly created directory to the system path:

   ```
   export PATH="$HOME/.npm-global/bin:$PATH"
   ```

5. At the **Cloud Shell** command prompt, in the vi editor interface, to save your changes and close the file, press **Esc**, press **:**, type **wq!** and press **Enter**.
6. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to install the Azure Building Blocks npm package:

   ```
   npm install -g @mspnp/azure-building-blocks
   ```

7. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to exit the shell:

   ```
   exit
   ```

8. In the **Cloud Shell timed out** pane, click **Reconnect**.

   Note: You need to restart Cloud Shell for the installation of the Buliding Blocks npm package to take effect.

10.3.0.3 Task 3: Deploy an Azure virtual network from Cloud Shell by using Azure Building Blocks

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to download the GitHub repository containing the Azure Building Blocks templates:

   ```
   git clone https://github.com/mspnp/template-building-blocks.git
   ```

2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to view the content of the Azure Building Block parameter file you will use for this deployment:

   ```
   cat ./template-building-blocks/scenarios/vnet/vnet-simple.json
   ```

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of your Azure subscription:

   ```
   SUBSCRIPTION_ID=$(az account list --query "[0].id" | tr -d '"')
   ```

4. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group you created earlier in this exercise:

   ```
   RESOURCE_GROUP='AADesignLab0202-RG'
   ```

5. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment:

   ```
   LOCATION=$(az group list --query "[?name == 'AADesignLab0201-RG'].location" --output tsv)
   ```

6. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create the **AADesignLab0202-RG** resource group.

   ```
   az group create --location $LOCATION --name $RESOURCE_GROUP
   ```

7. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy a virtual network by using the Azure Building Blocks:

   ```
   azbb -g $RESOURCE_GROUP -s $SUBSCRIPTION_ID -l $LOCATION -p ./template-building-blocks/scenarios/vn
   ```

8. Wait for the deployment to complete before you proceed to the next task.

10.3.0.4 Task 4: View deployment metadata

1. On the left side of the portal, click the **Resource groups** link.
2. On the **Resource groups** blade, click the entry representing the resource group you created earlier in this exercise.
3. With the **Overview** selection active, on the resource group blade, click the **Deployments** link.
4. On the resulting blade, click the latest deployment to view its metadata in a new blade.
5. Within the deployment blade, observe the information displayed in the **Operation details** section.
6. Close the **Cloud Shell** pane.

   Review: In this exercise, you deployed an Azure virtual network by using Azure Building Blocks templates from the cloud shell.

10.4 Exercise 3: Remove lab resources

10.4.0.1 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 at the bottom of the portal, 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,'AADesignLab02')]".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.

10.4.0.2 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,'AADesignLab02')]".name --output tsv | xargs -L1 bash -c
   ```

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

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

11 Creating Managed Server Applications in Azure

12 Lab Answer Key: Deploying Managed Containerized Workloads to Azure

12.1 Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
   - Username: **Admin**
   - Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
   - Microsoft Edge
   - File Explorer
   - Visual Studio Code
   - Microsoft Azure Storage Explorer
   - Bash on Ubuntu on Windows
   - Windows PowerShell

   Note: You can also find shortcuts to these applications in the **Start Menu**.

12.2 Exercise 1: Create Azure Kubernetes Service (AKS) cluster

12.2.0.1 Task 1: Open the Azure Portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** (https://portal.azure.com).
3. If prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

12.2.0.2 Task 2: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.

   Note: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.

2. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.

   Note: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.

3. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:

   - Leave the **Subscription** drop-down list entry set to its default value.
   - In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you intend to deploy resources in this lab.
   - In the **Resource group** section, ensure that the **Create new** option is selected and then, in the text box, type **AADesignLab0401-RG**.
   - In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   - In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
   - Click the **Create storage** button.

4. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

12.2.0.3 Task 3: Create an AKS cluster by using Cloud Shell

1. At the **Cloud Shell** command prompt at the bottom of the portal, type in the following command and press **Enter** to create a variable which value designates the name of the resource group you will use in this task:

   ```
   RESOURCE_GROUP='AADesignLab0402-RG'
   ```

2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment (replace the placeholder `<Azure region>` with the name of the Azure region to which you intend to deploy resources in this lab. `az account list-locations` will list all available locations for your subscription.):

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

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new resource group:

   ```
   az group create --name $RESOURCE_GROUP --location $LOCATION
   ```

4. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new AKS cluster:

   ```
   az aks create --resource-group $RESOURCE_GROUP --name aad0402-akscluster --node-count 1 --node-vm-
   ```

   Note: If you receive an error message regarding availability of the VM size which value is represented by the `--node-vm-size` parameter, review the message and try other suggested VM sizes.

   Note: Alternatively, in **PowerShell** on **Cloud Shell** you can identify VM sizes available in your subscription in a given region by running the following command and reviewing the values in the **Restriction** column (make sure to replace the `region` placeholder with the name of the target region):

   ```
   Get-AzComputeResourceSku | where {$_.Locations -icontains "region"} | Where-Object {($_.ResourceTy
   ```

Note: The **Restriction** column will contain the value **NotAvailableForSubscription** for VM sizes that are not available in your subscription.

5. Wait for the deployment to complete before you proceed to the next task.

   Note: This operation can take up to 10 minutes.

12.2.0.4 Task 4: Connect to the AKS cluster.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to retrieve the credentials to access the AKS cluster:

   ```
   az aks get-credentials --resource-group $RESOURCE_GROUP --name aad0402-akscluster
   ```

2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify connectivity to the AKS cluster:

   ```
   kubectl get nodes
   ```

3. At the **Cloud Shell** command prompt, review the output and verify that the node is reporting the **Ready** status. Rerun the command until the correct status is shown.

   Result: After you complete this exercise, you should have successfully deployed a new AKS cluster.

12.3 Exercise 2: Managing an AKS cluster and its containerized workloads.

12.3.0.1 Task 1: Deploy a containerized application to an AKS cluster

1. In the Microsoft Edge window, in the Azure portal, at the **Cloud Shell** prompt, type the following command and press **Enter** in order to deploy the **nginx** image from the Docker Hub:

   ```
   kubectl create deployment aad0402-akscluster --image=nginx --replicas=1 --port=80
   ```

   Note: Make sure to use lower case letters when typing the name of the deployment. You will also receive a notification that this command is deprecated and will be removed in a future version, but successfully created the cluster.

2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that a Kubernetes pod has been created:

   ```
   kubectl get pods
   ```

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to identify the state of the deployment:

   ```
   kubectl get deployment
   ```

4. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to make the pod available from Internet:

   ```
   kubectl expose deployment aad0402-akscluster --port=80 --type=LoadBalancer
   ```

   Note: Make sure to use lower case letters when typing the name of the deployment.

5. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to identify whether the public IP address has been provisioned:

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

6. Wait until the value in the **EXTERNAL-IP** column for the **aad0402-akscluster** entry changes from **<pending>** to a public IP address, then press **Ctrl-C** key combination. Note the public IP address in the **EXTERNAL-IP** column for **aad0402-akscluster**.

7. Start Microsoft Edge and browse to the IP address you obtained in the previous step. Verify that Microsoft Edge displays a web page with the **Welcome to nginx!** message.

12.3.0.2 Task 2: Scaling containerized applications and AKS cluster nodes

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to scale the deployment:

   ```
   kubectl scale --replicas=2 deployment/aad0402-akscluster
   ```

2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify the outcome of scaling the deployment:

   ```
   kubectl get pods
   ```

   Note: Review the output of the command and verify that the number of pods increased to 2.

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to scale out the number of cluster nodes:

   ```
   az aks scale --resource-group $RESOURCE_GROUP --name aad0402-akscluster --node-count 2
   ```

4. Wait for the provisioning of the additional node to complete.

   Note: This operation can take up to 10 minutes. If it fails, rerun the `az aks scale` command.

5. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify the outcome of scaling the cluster:

   ```
   kubectl get nodes
   ```

   Note: Review the output of the command and verify that the number of nodes increased to 2.

6. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to scale the deployment:

   ```
   kubectl scale --replicas=10 deployment/aad0402-akscluster
   ```

7. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify the outcome of scaling the deployment:

   ```
   kubectl get pods
   ```

   Note: Review the output of the command and verify that the number of pods increased to 10.

8. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to review the pods distribution across cluster nodes:

   ```
   kubectl get pod -o=custom-columns=NODE:.spec.nodeName,POD:.metadata.name
   ```

   Note: Review the output of the command and verify that the pods are distributed across both nodes.

9. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the deployment:

   ```
   kubectl delete deployment aad0402-akscluster
   ```

12.4 Exercise 3: Autoscaling pods in an AKS cluster

12.4.0.1 Task 1: Deploy a Kubernetes pod by using a .yaml file.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to download a sample containerized application:

   ```
   git clone https://github.com/Azure-Samples/azure-voting-app-redis.git
   ```

2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to navigate to the location of the downloaded app:

   ```
   cd azure-voting-app-redis
   ```

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list the content of the application **.yaml** file:

   ```
   cat azure-vote-all-in-one-redis.yaml
   ```

4. Review the output of the command and verify that the pod defninition includes requests and limits in the followng format:

   ```
   resources:
     requests:
       cpu: 250m
   ```

```
    limits:
      cpu: 500m
```

5. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy the application based on the **.yaml** file:

   ```
   kubectl apply -f azure-vote-all-in-one-redis.yaml
   ```

6. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to to verify that a Kubernetes pod has been created:

   ```
   kubectl get pods
   ```

7. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to identify whether the public IP address for the containerized application has been provisioned:

   ```
   kubectl get service azure-vote-front --watch
   ```

8. Wait until the value in the **EXTERNAL-IP** column for the **azure-vote-front** entry changes from **<pending>** to a public IP address, then press **Ctrl-C** key combination. Note the public IP address in the **EXTERNAL-IP** column for **azure-vote-front**.

9. Start Microsoft Edge and browse to the IP address you obtained in the previous step. Verify that Microsoft Edge displays a web page with the **Azure Voting App** message.

12.4.0.2 Task 2: Autoscale Kubernetes pods.

1. At the **Cloud Shell** command prompt, type in the following commands and press **Enter** after each to change the current directory and download a sample containerized application:

   ```
   cd ..
   git clone https://github.com/kubernetes-incubator/metrics-server.git
   ```

2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to install **Metrics Server**:

   ```
   kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/download/v0.3.6/compon
   ```

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to configure autoscaling for the **azure-vote-front** deployment:

   ```
   kubectl autoscale deployment azure-vote-front --cpu-percent=50 --min=3 --max=10
   ```

4. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to view the status of autoscaling:

   ```
   kubectl get hpa
   ```

5. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to view the pods:

   ```
   kubectl get pods
   ```

   Note: Verify that the number of replicas increased to 3. If that is not the case, wait one minute and rerun the two previous steps.

6. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the deployment:

   ```
   kubectl delete deployment azure-vote-front
   ```

7. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the deployment:

   ```
   kubectl delete deployment azure-vote-back
   ```

8. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that the commands you ran in the previous steps completed successfully:

   ```
   kubectl get pods
   ```

9. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the AKS cluster:

   ```
   az aks delete --resource-group AADesignLab0402-RG --name aad0402-akscluster --yes --no-wait
   ```

10. Close the **Cloud Shell** pane.

    Review: In this exercise, you implemented autoscaling of pods in an AKS cluster

12.5 Exercise 4: Implement DevOps with AKS

12.5.0.1 Task 1: Deploy DevOps with AKS

Note: This solution is based on the DevOps with Containers solution described at https://docs.microsoft.com/en-us/azure/architecture/example-scenario/apps/devops-with-aks.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to generate the SSH key pair that will be used to authenticate when accessing the Linux VMs running the Jenkins instance and Grafana console:

   ```
   ssh-keygen -t rsa -b 2048
   ```

   - When prompted to enter the file in which to save the key, press **Enter** to accept the default value **(~/.ssh/id_rsa)**.
   - When prompted to enter passphrase, press **Enter** twice.

2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the public key of the newly generated key pair:

   ```
   PUBLIC_KEY=$(cat ~/.ssh/id_rsa.pub)
   ```

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the public key of the newly generated key pair and which takes into account any special character the public key might include:

   ```
   PUBLIC_KEY_REGEX="$(echo $PUBLIC_KEY | sed -e 's/\\/\\\\/g; s/\//\\\//g; s/&/\\\&/g')"
   ```

   > **Note**: This is necessary because you will use the **sed** utility to insert this string into the Azure Resource Manager template parameters file. Alternatively, you could simply open the file and enter the public key string directly into the file.

4. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group you will use for the deployment:

   ```
   RESOURCE_GROUP='AADesignLab0403-RG'
   ```

5. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment:

   ```
   LOCATION=$(az group list --query "[?name == 'AADesignLab0402-RG'].location" --output tsv)
   ```

6. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new resource group:

   ```
   az group create --name $RESOURCE_GROUP --location $LOCATION
   ```

7. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create an Azure Active Directory service principal for the authentication of services and resources within the sample solution:

   ```
   SERVICE_PRINCIPAL=$(az ad sp create-for-rbac --name http://AADesignLab0403-SP --output json)
   ```

8. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to retrieve the **appId** attribute of the newly created service principal:

   ```
   APP_ID=$(echo $SERVICE_PRINCIPAL | jq -r .appId)
   ```

9. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to retrieve the **password** attribute of the newly created service principal:

   ```
   PASSWORD=$(echo $SERVICE_PRINCIPAL | jq -r .password)
   ```

10. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create the parameters file you will use for deployment of the sample solution and open it in the vi interface:

    ```
    vi ~/parameters.json
    ```

11. At the **Cloud Shell** command prompt, in the vi editor interface, add the content of the sample parameters file (**\allfiles\AZ-301T03\Module_02\Labfiles\Starter\parameters.json**):

```
{
  "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "spClientId": {
      "value": "$APP_ID"
    },
    "spClientSecret": {
      "value": "$PASSWORD"
    },
    "linuxAdminUsername": {
      "value": "Student"
    },
    "linuxAdminPassword": {
      "value": "Pa55w.rd1234"
    },
    "linuxSSHPublicKey": {
      "value": "$PUBLIC_KEY_REGEX"
    }
  }
}
```

12. At the **Cloud Shell** command prompt, in the vi editor interface, to save your changes and close the file, press **Esc**, press **:**, type **wq!** and press **Enter**.

13. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **appId** attibute with the value of the **$APP_ID** variable in the parameters file:

```
sed -i.bak1 's/"$APP_ID"/"'"$APP_ID"'"/' ~/parameters.json
```

14. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **password** attribute with the value of the **$PASSWORD** variable in the parameters file:

```
sed -i.bak2 's/"$PASSWORD"/"'"$PASSWORD"'"/' ~/parameters.json
```

15. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **sshPublicKey** parameter with the value of the **$PUBLIC_KEY_REGEX** variable in the parameters file:

```
sed -i.bak3 's/"$PUBLIC_KEY_REGEX"/"'"$PUBLIC_KEY_REGEX"'"/' ~/parameters.json
```

16. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that the placeholders were successfully replaced in the parameters file:

```
cat ~/parameters.json
```

17. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to identify AKS versions supported in the Azure region you are using in this lab:

```
az aks get-versions --location $LOCATION --output table
```

18. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy the sample solution by using its Azure Resource Manager template residing in a GitHub repository:

```
az group deployment create --resource-group $RESOURCE_GROUP --template-uri https://raw.githubuserc
```

    Note: If prompted, provide one of the currently supported AKS versions.

    Note: If you receive an error message regarding availability of the VM size which value is represented by the `--node-vm-size` parameter, review the message and try other suggested VM sizes.

    Note: Alternatively, in **PowerShell** on **Cloud Shell** you can identify VM sizes available in your subscription in a given region by running the following command and reviewing the values

in the **Restriction** column (make sure to replace the `region` placeholder with the name of the target region):

```
Get-AzComputeResourceSku | where {$_.Locations -icontains "region"} | Where-Object {($_.ResourceTy
```

Note: The **Restriction** column will contain the value **NotAvailableForSubscription** for VM sizes that are not available in your subscription.

19. Wait for the deployment to complete before you proceed to the next task.

    Note: The deployment can take up to 15 minutes.

12.5.0.2 Task 2: Review the DevOps with AKS architecture

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click the entry representing the **AADesignLab0403-RG** resource group.
3. On the **AADesignLab0403-RG** resource group blade, review the list of resources and compare them with the information available at https://docs.microsoft.com/en-us/azure/architecture/example-scenario/apps/devops-with-aks

   Review: In this exercise, you deployed DevOps with AKS architecture.

12.6 Exercise 5: Remove lab resources

12.6.0.1 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 at the bottom of the portal, 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,'AADesignLab04')]".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.

12.6.0.2 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,'AADesignLab04')]".name --output tsv | xargs -L1 bash -c
   ```

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

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

13 Authoring Serverless Applications in Azure

14 Lab Answer Key: Deploying Serverless Workloads to Azure

14.1 Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
   - Username: **Admin**
   - Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
   - Microsoft Edge
   - File Explorer
   - Visual Studio Code

- Microsoft Azure Storage Explorer
- Bash on Ubuntu on Windows
- Windows PowerShell

  Note: You can also find shortcuts to these applications in the **Start Menu**.

14.2 Exercise 1: Create Web App

14.2.0.1 Task 1: Open the Azure Portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** (https://portal.azure.com).
3. If prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

14.2.0.2 Task 2: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.

   Note: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.

2. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.

   Note: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.

3. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:
   - Leave the **Subscription** drop-down list entry set to its default value.
   - In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you intend to deploy resources in this lab
   - In the **Resource group** section, ensure that the **Create new** option is selected and then, in the text box below, type **AADesignLab0501-RG**.
   - In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   - In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
   - Click the **Create storage** button.
4. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

14.2.0.3 Task 3: Create an App Service plan

1. At the **Cloud Shell** command prompt at the bottom of the portal, type in the following command and press **Enter** to create a variable which value designates the name of the resource group you will use in this exercise:

   ```
   RESOURCE_GROUP_APP='AADesignLab0502-RG'
   ```

2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment (Enter the name of the region when prompted):

   ```
   read -p 'Region: ' LOCATION
   ```

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create the resource group:

```
az group create --name $RESOURCE_GROUP_APP --location $LOCATION
```

4. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new App Service plan:

```
az appservice plan create --is-linux --name "AADesignLab0502-$LOCATION" --resource-group $RESOURCE
```

> **Note**: In case the command fails with the message *Linux workers are not available in resource group AADesignLab0502-RG. Use this link to learn more https://go.microsoft.com/fwlink/?linkid=831180"*, delete the resource group, set **LOCATION** to **eastus** and rerun the two previous steps.

14.2.0.4 Task 4: Create a Web App instance

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to view a list of possible runtimes for a Linux-based App Service web app instance:

```
az webapp list-runtimes --linux --output tsv
```

2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new variable which value is a randomly generated string that you will use as the name of a new web app:

```
WEBAPPNAME1=webapp05021$RANDOM$RANDOM
```

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new web app using a unique name:

```
az webapp create --name $WEBAPPNAME1 --plan AADesignLab0502-$LOCATION --resource-group $RESOURCE_G
```

> **Note**: In case the command fails due to duplicate web app name, re-run the last two steps until the command completes successfully

4. Wait for the deployment to complete before you proceed to the next task.

14.2.0.5 Task 5: View deployment results

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0502-RG**.
3. On the **AADesignLab0502-RG** blade, click the entry representing the Azure web app you created earlier in this exercise.
4. On the web app blade, click the **Browse** button at the top of the blade.
5. Review the default page generated by Azure App Service.
6. Close the new browser tab and return to the browser tab displaying the Azure portal.

Review: In this exercise, you created a Linux-based App Service Plan that contained a blank web app.

14.3 Exercise 2: Deploy Web App code

14.3.0.1 Task 1: Deploy code with a Web App Extension using an Azure Resource Manager template and GitHub

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.
2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group you will use in this exercise:

```
RESOURCE_GROUP_APP='AADesignLab0502-RG'
```

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment:

```
LOCATION=$(az group list --query "[?name == 'AADesignLab0502-RG'].location" --output tsv)
```

4. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new variable which value is a randomly generated string that you will use as the name of a new web app:

```
WEBAPPNAME2=webapp05022$RANDOM$RANDOM
```

5. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new web app using a unique name:

```
az webapp create --name $WEBAPPNAME2 --plan AADesignLab0502-$LOCATION --resource-group $RESOURCE_GI
```

> **Note**: In case the command fails due to duplicate web app name, re-run the last two steps until the command completes successfully

6. In the **Cloud Shell** pane, click the **Upload/Download files** icon and, in the drop-down menu, click **Upload**.

7. In the **Open** dialog box, navigate to the **\allfiles\AZ-301T03\Module_03\Labfiles\Starter** folder, select the **github.json** file, and click **Open**. The file contains the following Azure Resource Manager template:

```
{
    "$schema": "http://schemas.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "webAppName": {
            "type": "string"
        },
        "repositoryUrl": {
            "type": "string"
        },
        "branch": {
            "type": "string",
            "defaultValue": "master"
        }
    },
    "resources": [
        {
            "apiVersion": "2015-08-01",
            "type": "Microsoft.Web/sites",
            "name": "[parameters('webAppName')]",
            "location": "[resourceGroup().location]",
            "properties": {},
            "resources": [
                {
                    "apiVersion": "2015-08-01",
                    "name": "web",
                    "type": "sourcecontrols",
                    "dependsOn": [
                        "[resourceId('Microsoft.Web/Sites', parameters('webAppName'))]"
                    ],
                    "properties": {
                        "RepoUrl": "[parameters('repositoryUrl')]",
                        "branch": "[parameters('branch')]",
                        "IsManualIntegration": true
                    }
                }
            ]
        }
    ]
}
```

8. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy the GitHub-resident web app code by using a local Azure Resource Manager template and a local parameters file:

```
az group deployment create --resource-group $RESOURCE_GROUP_APP --template-file github.json --para
```

9. Wait for the deployment to complete before you proceed to the next task.

> **Note**: The deployment should take about a minute.

14.3.0.2 Task 2: View deployment results

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0502-RG**.
3. On the **AADesignLab0502-RG** blade, click the entry representing the Azure web app you created in the previous task.
4. On the web app blade, click the **Browse** button at the top of the blade.
5. Review the sample Node.js web application deployed from GitHub.
6. Close the new browser tab and return to the browser tab displaying the Azure portal.

14.3.0.3 Task 3: Deploy Code with a Docker Hub container image

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group you will use in this task:

   ```
   RESOURCE_GROUP_CONTAINER='AADesignLab0502-RG'
   ```

2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment:

   ```
   LOCATION=$(az group list --query "[?name == 'AADesignLab0502-RG'].location" --output tsv)
   ```

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new variable which value is a randomly generated string that you will use as the name of a new web app:

   ```
   WEBAPPNAME3=webapp05023$RANDOM$RANDOM
   ```

4. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new web app using a unique name:

   ```
   az webapp create --name $WEBAPPNAME3 --plan AADesignLab0502-$LOCATION --resource-group $RESOURCE_G
   ```

   Note: In case the command fails due to duplicate web app name, re-run the last two steps until the command completes successfully

5. Wait for the deployment to complete before you proceed to the next task.

   Note: The deployment should take less than a minute.

6. Close the **Cloud Shell** pane.

14.3.0.4 Task 4: View deployment results

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0502-RG**.
3. On the **AADesignLab0502-RG** blade, click the entry representing the Azure web app you created in the previous task.
4. On the web app blade, click the **Browse** button at the top of the blade.

   Note: If the application does not appear, switch to the web app blade, click **Restart** button at the top of the blade and then click **Browse** again.

5. Review the blog application deployed from Docker Hub.
6. Close the new browser tab and return to the browser tab displaying the Azure portal.

Review: In this exercise, you deployed code using an Azure Resource Manager template and a Docker Hub image to App Service web apps.

14.4 Exercise 3: Deploy a Function App

14.4.0.1 Task 1: Deploy a Function App with code using an Azure Resource Manager template

1. In the upper left corner of the Azure portal, click **Create a resource**.

2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Template Deployment** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Template deployment**.
4. On the **Template deployment** blade, click the **Create** button.
5. On the **Custom deployment** blade, click the **Build your own template in the editor** link.
6. On the **Edit template** blade, click the **Quickstart template** link.
7. In the **Load a quickstart template** pane, in the **Select a template** drop-down list, select the **201-function-app-dedicated-github-deploy** template.
8. Click the **Select template** button.
9. On the **Provision a function app with source deployed from GitHub** blade, perform the following tasks:
   - Leave the **Subscription** drop-down list entry set to its default value.
   - In the **Resource group** section, ensure that **Create new** option is selected and, in the text box below, type **AADesignLab0503-RG**.
   - Leave the **Location** text box set to its default value.
   - In the **App Name** text box, accept the default value.
   - In the **Sku** text box, type **B1**.
   - Leave the **Worker Size** drop-down list set to its default value.
   - Leave the **Storage Account Type** drop-down list set to its default value.
   - Leave the **Repo URL** field set to its default value.
   - Leave the **Branch** text box set to its default value.
   - Leave the **Location** text box set to its default value.
   - In the **Terms and Conditions** section, select the **I agree to the terms and conditions stated above** checkbox.
   - Click the **Purchase** button.
10. Wait for the deployment to complete before you proceed to the next task.

    Note: The deployment should take about a minute.

14.4.0.2 Task 2: View deployment results

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0503-RG**.
3. On the **AADesignLab0503-RG** blade, click the entry representing the Function App you created in the previous task.
4. On the Function App blade, locate the **Url** entry and click the hyperlink below to see the Function App landing page in a new browser tab.
5. Close the new browser tab and return to the browser tab displaying the Azure portal.

   Review: In this exercise, you deployed a Function App and code using an Azure Resource Manager template.

14.5 Exercise 4: Remove lab resources

14.5.0.1 Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. 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,'AADesignLab05')]".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.

14.5.0.2 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,'AADesignLab05')]".name --output tsv | xargs -L1 bash -c
   ```

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

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

15 Building Azure IaaS-Based Server Applications.

16 Lab Answer Key: Building Azure IaaS-Based Server Applications by using Azure Resource Manager Templates and Azure Building Blocks.

16.1 Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
   - Username: **Admin**
   - Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
   - Microsoft Edge
   - File Explorer
   - Visual Studio Code
   - Microsoft Azure Storage Explorer
   - Bash on Ubuntu on Windows
   - Windows PowerShell

   Note: You can also find shortcuts to these applications in the **Start Menu**.

16.2 Exercise 1: Deploy an Azure VM by using Azure Resource Manager templates with PowerShell Desired State Configuration (DSC) extension from the Azure portal.

16.2.0.1 Task 1: Open the Azure Portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** (https://portal.azure.com).
3. If prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

16.2.0.2 Task 2: Create an Azure VM running Windows Server 2016 Datacenter.

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Windows Server** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Windows Server**.
4. On the **Windows Server** blade, select the **[smalldisk] Windows Server 2016 Datacenter** software plan, then click the **Create** button.

5. On the **Basics** tab, perform the following tasks (leave all other settings with their default values):
   - Leave the **Subscription** drop-down list entry set to its default value.
   - In the **Resource group** section, click **Create new**, in the text box, type **AADesignLab0301-RG**, and click **OK**.
   - In the **Name** text box, enter the value **lab03vm0**.
   - In the **Region** drop-down list, select an Azure region to which you want to deploy resources in this lab.
   - In the **Availability options** drop-down list, select **Availability set**.
   - In the **Availability set** section, click **Create new**, box, enter the value **lab03avset0**, set **Fault domains** to the maximum value, leave **Update domains** with its default value, and click **OK**.
   - Leave the entry in the **Image** drop-down list set to its default value.
   - Ensure that the size is set to **Standard D2s v3**
   - In the **Username** text box, enter the value **Student**.
   - In the **Password** and **Confirm password** text boxes, enter the value **Pa55w.rd1234**.
   - In the **Public inbound ports** section, select the **Allow selected port** option and, in the **Select inbound ports** drop-down list, select **HTTP**.
   - Leave the **Already have a Windows license?** option set to **No**.
   - Click **Next: Disks >**
6. On the **Disks** tab, perform the following tasks (leave all other settings with their default values):
   - Ensure that the **OS disk type** dropdown list entry is set to **Standard HDD**
   - Click **Next: Networking >**
7. On the **Networking** tab, perform the following tasks (leave all other settings with their default values):
   o
   - In the **Virtual network** section, click **Create new**.
   - On the **Create virtual network** blade, specify the following settings and click **OK**:
     - In the **Name** text box, enter the value **lab03vnet0**.
     - In the **Address range** text box, enter the value **10.3.0.0/16**.
     - In the **Subnet name** text box, enter the value **subnet-0**.
     - In the **Subnet address range** text box, enter the value **10.3.0.0/24**, and click **OK**.
   - Leave the **Public IP** entry set to its default value.
   - Leave the **NIC network security group** option set to **Basic**.
   - Leave the **Public inbound ports** option set to **Allow selected ports**
   - Leave the **Select inbound ports** entry set to **HTTP**
   - Leave the **Accelerated networking** entry set to its default value.
   - Click **Next: Management >**
8. On the **Management** tab, perform the following tasks (leave all other settings with their default values):
   - Leave the **Boot diagnostics** option set to its default value.
   - Leave the **OS guest diagnostics** option set to its default value.
   - Leave the **Diagnostics storage account** entry set to its default value.
   - Leave the **System assigned managed identity** option set to its default value.
   - Leave the **Enable auto-shutdown** option set to its default value.
   - Leave the **Enable backup** option set to its default value.

- Click the **Review + create** button.

9. On the **Create a virtual machine** blade, review the settings of your new virtual machine and click the **Create** button.
10. Do not wait for the deployment to complete and proceed to the next task.

16.2.0.3 Task 3: View DSC configuration

1. On the Taskbar, click the **File Explorer** icon.
2. In the **File Explorer** window that appears, navigate to the **\allfiles\AZ-301T04\Module_02\LabFiles\Starter** folder.
3. Right-click the **IISWebServer.zip** file and select the **Extract All...** option.
4. In the **Extract Compressed (Zipped) Folders** dialog, perform the following tasks:
   - In the **Files will be extracted to this folder:** field, enter the name of the folder into which you want to extract the files.
   - Ensure that the **Show extracted files when complete** checkbox is selected.
   - Click the **Extract** button.
5. In the new **File Explorer** window that appears, right-click the **IISWebServer.ps1** file and select the **Open with Code** option to start the **Visual Studio Code** application.
6. In the **Visual Studio Code** window that appears, review the content of the PowerShell script.
7. At the top of the **Visual Studio Code** window, click the **File** menu and select the **Close Window** option.
8. Close both **File Explorer** windows.
9. Return to the **Microsoft Edge** window with the **Azure Portal** open.

16.2.0.4 Task 4: Create an Azure Storage account

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Storage account** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Storage account**.
4. On the **Storage account** blade, click the **Create** button.
5. On the **Create storage account** blade, perform the following tasks:
   - Leave the **Subscription** drop-down list entry set to its default value.
   - In the **Resource group** section, ensure that the **Use existing** option is selected and, in the drop-down list below, select the resource group you created earlier in this exercise.
   - In the **Name** text box, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   - Leave the **Location** entry set to the same Azure region you selected earlier in this exercise.
   - In the **Performance** section, ensure that the **Standard** option is selected.
   - In the **Account kind** drop-down list, ensure that the **Storage (general purpose v1)** option is selected.
   - In the **Replication** drop-down list, select the **Locally-redundant storage (LRS)** entry.
   - Click the **Advanced** tab at the top of the blade.
   - Select the **Enabled** radio button near the **Public blob access** line.
   - Click the **Review + Create** button, and then click **Create**.
6. Wait for the deployment to complete before you proceed to the next task.

   Note: This operation can take about 2 minutes.

16.2.0.5 Task 5: Upload DSC configuration to Azure Storage

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click the entry representing the resource group into which you deployed the storage account.
3. On the resource group blade, click the entry representing the newly created storage account.
4. With the **Overview** selection active, on the storage account blade, click **Containers**.
5. Click the **Container** button at the top of the blade.
6. In the **New container** pane that appears, specify the following settings and click **Create**:
   - In the **Name** text box, enter the value **config**.
   - In the **Public access level** list, select the **Blob (anonymous read access for blobs only)** option.
7. Back on the **Blob service** blade, click the entry representing the new **config** container.
8. On the **config** blade, click the **Upload** button at the top of the blade.
9. In the **Upload blob** pane, perform the following tasks:
   - In the **Files** field, click the blue folder button to the right of the field.
   - In the **Open file** dialog that appears, navigate to the **\allfiles\AZ-301T04\Module_02\LabFiles\Starter** folder.
   - Select the **IISWebServer.zip** file.
   - Click the **Open** button to close the dialog box and return to the **Upload blob** popup.
   - Click the **Upload** button.
10. Navigate to the **config** blade and click the entry representing the **IISWebServer.zip** blob.
11. In the **Blob properties** popup that appears, locate and record the value of the **URL** property. This URL will be used later in this lab.

16.2.0.6 Task 6: Deploy an Azure VM by using an Azure Resource Manager template with PowerShell DSC extension from the Azure portal.

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Template Deployment** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Template deployment**.
4. On the **Template deployment** blade, click the **Create** button.
5. On the **Custom deployment** blade, click the **Build your own template in the editor** link.
6. On the **Edit template** blade, click **Load file**.
7. In the **Choose File to Upload** dialog box, navigate to the **\allfiles\AZ-301T04\Module_02\LabFiles\Starter** folder, select the **dsc-extension-template.json** file, and click **Open**. This will load the following content into the template editor pane:

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "virtualMachineName": {
            "type": "string",
            "defaultValue": "lab03vm0"
        },
        "configurationModuleUrl": {
            "type": "string"
        },
        "extensionFunction": {
```

```
                "type": "string",
                "defaultValue": "IISWebServer.ps1\\IISWebServer"
            }
        },
        "resources": [
            {
                "apiVersion": "2018-06-01",
                "type": "Microsoft.Compute/virtualMachines/extensions",
                "name": "[concat(parameters('virtualMachineName'), '/dscExtension')]",
                "location": "[resourceGroup().location]",
                "properties": {
                    "publisher": "Microsoft.Powershell",
                    "type": "DSC",
                    "typeHandlerVersion": "2.75",
                    "autoUpgradeMinorVersion": true,
                    "settings": {
                        "ModulesUrl": "[parameters('configurationModuleUrl')]",
                        "ConfigurationFunction": "[parameters('extensionFunction')]",
                        "Properties": {
                            "MachineName": "[parameters('virtualMachineName')]"
                        }
                    },
                    "protectedSettings": null
                }
            }
        ]
    }
```

8. Click the **Save** button to persist the template.
9. Back on the **Custom deployment** blade, perform the following tasks:
   - Leave the **Subscription** drop-down list entry set to its default value.
   - In the **Resource group** section, select the **Use existing** option and, in the drop-down list, select the resource group you created earlier in this exercise.
   - Leave the **Location** drop-down list set to its default value.
   - Leave the **Virtual Machine Name** field set to its default value: **lab03vm0**.
   - In the **Configuration Module Url** field, enter the URL value that you recorded in the previous task.
   - Leave the **Extension Function** field set to its default value: **IISWebServer.ps1\IISWebServer**.
   - In the **Terms and Conditions** section, select the **I agree to the terms and conditions stated above** checkbox.
   - Click the **Purchase** button.
10. Wait for the deployment of the DSC configuration to complete before you proceed to the next task.

    Note: DSC configuration deployment can take up to ten minutes.

16.2.0.7 Task 7: Validate that the Azure VM is serving web content

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click the entry representing the resource group into which you deployed the virtual machine.
3. On the resource group blade, click the entry representing the **Virtual Machine** you deployed.
4. On the **Virtual machine** blade, locate the **Public IP address** entry, and identify its value.
5. Open a new Microsoft Edge tab and navigate to the IP address you identified in the previous step.
6. Verify that you are able to access the default Internet Information Services webpage.

7. Close the new browser tab.

   Review: In this exercise, you deployed an **Virtual Machine** from the Azure portal and then used the **PowerShell DSC** extension to apply changes to the virtual machine in an unattended manner.

16.3 Exercise 2: Deploy an Azure Virtual Machine Scale Set (VMSS) by using Azure Resource Manager templates with PowerShell Desired State Configuration (DSC) extension from the Azure portal.

16.3.0.1 Task 1: View an Azure Resource Manager template.

1. On the Taskbar, click the **File Explorer** icon.
2. In the **File Explorer** window that appears, navigate to the **\allfiles\AZ-301T04\Module_02\LabFiles\Starter** folder.
3. Right-click the **vmss-template.json** file and select the **Open with Code** option to start the **Visual Studio Code** application.
4. In the **Visual Studio Code** window that appears, review the content of the JSON file.
5. At the top of the **Visual Studio Code** window, click the **File** menu and select the **Close Window** option.
6. Close the **File Explorer** window.
7. Return to the **Microsoft Edge** window with the **Azure Portal** open.

16.3.0.2 Task 2: Deploy a VMSS using ARM

1. In the hub menu of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Template Deployment** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Template deployment**.
4. On the **Template deployment** blade, click the **Create** button.
5. On the **Custom deployment** blade, click **Build your own template in the editor**.
6. On the **Edit template** blade, click **Load file**.
7. In the **Open** file dialog that appears, navigate to the **\allfiles\AZ-301T04\Module_02\LabFiles\Starter*** folder.
8. Select the **vmss-template.json** file.
9. Click the **Open** button.
10. Back on the **Edit template** blade, click the **Save** button to persist the template.
11. Back on the **Custom deployment** blade, perform the following tasks:
    - Leave the **Subscription** drop-down list entry set to its default value.
    - In the **Resource group** section, select the **Create new** option and, in the text box, type **AADesignLab0302-RG**.
    - Leave the **Location** entry set to its default value.
    - In the **Admin User Name** text box, enter the value **Student**.
    - In the **Admin Password** text box, enter the value **Pa55w.rd1234**.
    - In the **Instance Count** text box, enter the value **2**.
    - Leave the **Overprovision** text box set to its default value: **true**.
    - In the **Configuration Module Url** text box, enter the URL that you recorded for the uploaded blob in the previous exercise of this lab.
    - In the **Terms and Conditions** section, select the **I agree to the terms and conditions stated above** checkbox.

- Click the **Purchase** button.

12. Wait for the deployment to complete before you proceed to the next task.

16.3.0.3 Task 3: Validate that VMSS instances are serving web content

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click the entry representing the resource group into which you deployed the virtual machine scale set.
3. On the resource group blade, click the resource of the **Public IP address** type.
4. On the Public IP address resource blade, in the **Essentials** section, identify the value of **IP address** entry.
5. Open a new Microsoft Edge tab and navigate to the IP address you identified in the previous step.
6. Verify that you are able to access the default Internet Information Services webpage.
7. Close the new browser tab and return to the browser tab with the **Azure Portal** currently active.

   Review: In this exercise, you created a Virtual Machine scale set and configured the individual instances using PowerShell DSC.

16.4 Exercise 3: Deploy Azure VMs running Windows Server 2016 and Linux by using Azure Building Blocks with PowerShell Desired State Configuration (DSC) extension.

16.4.0.1 Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.

   Note: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.

2. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.

   Note: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.

3. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:
   - Leave the **Subscription** drop-down list entry set to its default value.
   - In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you intend to deploy resources in this exercise.
   - In the **Resource group** section, ensure that the **Use existing** option is selected and then select **AADesignLab0301-RG**.
   - In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   - In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
   - Click the **Create storage** button.
4. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

16.4.0.2 Task 2: Deploy an Azure VM running Linux Ubuntu 18.04 that will be used to perform Azure Building Blocks-based deployments.

Note: This is necessary to account for breaking changes affecting running from Cloud Shell.

1. In the **Cloud Shell** pane, run the following to create a variable which value designates the name of the resource group you will use in this exercise:

   ```
   RESOURCE_GROUP='AADesignLab0303-RG'
   ```

2. In the **Cloud Shell** pane, run the following to create a variable which value designates the Azure region you will use for the deployment:

   ```
   LOCATION=$(az group list --query "[?name == 'AADesignLab0301-RG'].location" --output tsv)
   ```

3. In the **Cloud Shell** pane, run the following to create a resource group that you will use for the deployment:

   ```
   az group create --name $RESOURCE_GROUP --location $LOCATION
   ```

4. In the **Cloud Shell** pane, run the following to deploy an Azure VM running Linux Ubuntu 18.04 that you will use for deploying resources via Azure Building Blocks:

   ```
   UBUNTU_IMAGE='Canonical:UbuntuServer:18.04-LTS:latest'
   VM_NAME='lab03vm1'
   USER_NAME='student'
   az vm create \
   --name $VM_NAME \
   --resource-group $RESOURCE_GROUP \
   --location $LOCATION \
   --image $UBUNTU_IMAGE \
   --admin-username $USER_NAME \
   --generate-ssh-keys \
   --size Standard_D2s_v3
   ```

   Note: Wait until the deployment completes.

1. In the **Cloud Shell** pane, run the following to retrieve the public IP address of the newly deployed Azure VM **lab03vm1**:

   ```
   IP_ADDRESS=$(az vm show -d --resource-group $RESOURCE_GROUP --name $VM_NAME --query publicIps -o t
   ```

2. In the **Cloud Shell** pane, run the following to open an SSH session to the newly deployed Azure VM **lab03vm1**:

   ```
   ssh student@$IP_ADDRESS
   ```

3. In the **Cloud Shell** pane, when prompted whether to continue, type **yes** and press Enter.

16.4.0.3 Task 3: Install the Azure Building Blocks npm package in the Azure VM running Linux.

1. Within the SSH session to the Azure VM **lab03vm1**, run the following to update locally installed packages and their dependencies:

   ```
   sudo apt-get upgrade
   ```

2. Within the SSH session to the Azure VM **lab03vm1**, run the following to ensure that there are no remaining updates to be processed:

   ```
   sudo -i apt update
   ```

3. Within the SSH session to the Azure VM **lab03vm1**, run the following to install Azure CLI:

   ```
   curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
   ```

4. Within the SSH session to the Azure VM **lab03vm1**, run the following to install node.js v10:

   ```
   curl -sL https://deb.nodesource.com/setup_10.x | sudo bash -
   sudo apt-get install -y nodejs
   ```

5. Within the SSH session to the Azure VM **lab03vm1**, run the following to verify the versions of node.js and npm:

   ```
   node --version
   npm --version
   ```

   Note: Verify that the versions are **v10.21.0** and **6.14.4**, respectively.

6. Within the SSH session to the Azure VM **lab03vm1**, run the following to install Azure Building Blocks:

```
sudo su -
npm install -g @mspnp/azure-building-blocks
```

7. Within the SSH session to the Azure VM **lab03vm1**, run the following to exit the root mode and switch to the home directory:

```
exit
cd $HOME
```

8. Within the SSH session to the Azure VM **lab03vm1**, run the following to set the npm path:

```
npm config set prefix '~/.npm-global'
```

16.4.0.4 Task 4: Deploy a Windows Server 2016 Azure VM from Cloud Shell by using Azure Building Blocks

1. Within the SSH session to the Azure VM **lab03vm1**, run the following to download the GitHub repository containing the Azure Building Blocks reference architecture files:

```
git clone https://github.com/mspnp/reference-architectures.git
```

2. Within the SSH session to the Azure VM **lab03vm1**, run the following to view the content of the Azure Building Block parameter file you will use for this deployment:

```
cat ./reference-architectures/virtual-machines/single-vm/parameters/windows/single-vm.json
```

3. Within the SSH session to the Azure VM **lab03vm1**, run the following to authenticate to your Azure subscription:

```
az login
```

> **Note**: Follow the instructions provided in the output of the command to authenticate to your Azure subscription.

4. Once you authenticated, return to the Cloud Shell pane displaying the SSH session to the Azure VM **lab03vm1**, and run the following to create a variable which value designates the name of your Azure subscription:

```
SUBSCRIPTION_ID=$(az account list --query "[0].id" --output tsv | tr -d '"')
```

5. Within the SSH session to the Azure VM **lab03vm1**, run the following to create a variable which value designates the name of the resource group you created earlier in this exercise:

```
RESOURCE_GROUP='AADesignLab0303-RG'
```

6. Within the SSH session to the Azure VM **lab03vm1**, run the following to create a variable which value designates the Azure region you will use for the deployment:

```
LOCATION=$(az group list --query "[?name == 'AADesignLab0301-RG'].location" --output tsv)
```

7. Within the SSH session to the Azure VM **lab03vm1**, run the following to create a resource group that you will use for the deployment:

```
az group create --name $RESOURCE_GROUP --location $LOCATION
```

8. Within the SSH session to the Azure VM **lab03vm1**, run the following to replace the placeholder for the **adminUsername** parameter with the value **Student** in the Building Blocks parameter file:

```
sed -i.bak1 's/"adminUsername": ""/"adminUsername": "Student"/' ./reference-architectures/virtual-
```

9. Within the SSH session to the Azure VM **lab03vm1**, run the following to replace the placeholder for the **adminPassword** parameter with the value **Pa55w.rd1234** in the Building Blocks parameter file:

```
sed -i.bak2 's/"adminPassword": ""/"adminPassword": "Pa55w.rd1234"/' ./reference-architectures/vir
```

10. Within the SSH session to the Azure VM **lab03vm1**, run the following to set the value of the size parameter of the virtual machines to **Standard_D2s_v3** in the Building Blocks parameter file:

```
sed -i.bak3 's/"Standard_DS1_v2"/"Standard_D2s_v3"/g' ./reference-architectures/virtual-machines/s
```

11. Within the SSH session to the Azure VM **lab03vm1**, run the following to verify that the parameter values were successfully changed in the Building Blocks parameter file:

```
cat ./reference-architectures/virtual-machines/single-vm/parameters/windows/single-vm.json
```

12. Within the SSH session to the Azure VM **lab03vm1**, run the following to deploy a Windows Server 2016 Azure VM by using the Azure Building Blocks:

```
azbb -g $RESOURCE_GROUP -s $SUBSCRIPTION_ID -l $LOCATION -p ./reference-architectures/virtual-machi
```

13. Wait for the deployment to complete before you proceed to the next task.

16.4.0.5 Task 5: Validate that the Windows Server 2016 Azure VM is serving web content

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click the **AADesignLab0303-RG** entry representing the resource group into which you deployed the Windows Server 2016 Datacenter virtual machine earlier in this exercise.
3. On the resource group blade, click the **ra-single-windows-vm1** entry representing the virtual machine you deployed by using Azure Building Blocks.
4. On the **Virtual machine** blade, locate the **Public IP address** entry, and identify its value.
5. Open a new Microsoft Edge tab and navigate to the IP address you identified in the previous step.
6. Verify that you are able to access the default Internet Information Services webpage.
7. Close the new browser tab.

16.4.0.6 Task 6: Deploy a Linux Azure VM from Cloud Shell by using Azure Building Blocks

1. In the **Cloud Shell** pane, within the SSH session to the Azure VM **lab03vm1**, run the following to view the content of the Azure Building Block parameter file you will use for this deployment:

```
cat ./reference-architectures/virtual-machines/single-vm/parameters/linux/single-vm.json
```

2. Within the SSH session to the Azure VM **lab03vm1**, run the following to generate the SSH key pair that you will use to authenticate when accessing the Linux VM:

```
ssh-keygen -t rsa -b 2048
```

   - When prompted to enter the file in which to save the key, press **Enter** to accept the default value **(~/.ssh/id_rsa)**.
   - When prompted to enter passphrase, press **Enter** twice.

3. Within the SSH session to the Azure VM **lab03vm1**, run the following to create a variable which value designates the public key of the newly generated key pair:

```
PUBLIC_KEY=$(cat ~/.ssh/id_rsa.pub)
```

4. Within the SSH session to the Azure VM **lab03vm1**, run the following to create a variable which value designates the public key of the newly generated key pair and which takes into account any special character the public key might include:

```
PUBLIC_KEY_REGEX="$(echo $PUBLIC_KEY | sed -e 's/\\/\\\\/g; s/\//\\\//g; s/&/\\\&/g')"
```

   > **Note**: This is necessary because you will use the **sed** utility to insert this string into the Azure Building Blocks parameter file. Alternatively, you could simply open the file and enter the public key string directly into the file.

5. Within the SSH session to the Azure VM **lab03vm1**, run the following to create a variable which value designates the name of your Azure subscription:

```
SUBSCRIPTION_ID=$(az account list --query "[0].id" --output tsv | tr -d '"')
```

6. Within the SSH session to the Azure VM **lab03vm1**, run the following to create a variable which value designates the name of the resource group you will use for the deployment:

```
RESOURCE_GROUP='AADesignLab0304-RG'
```

7. Within the SSH session to the Azure VM **lab03vm1**, run the following to create a variable which value designates the Azure region you will use for the deployment:

```
LOCATION=$(az group list --query "[?name == 'AADesignLab0301-RG'].location" --output tsv)
```

8. Within the SSH session to the Azure VM **lab03vm1**, run the following to replace the placeholder for the **adminUsername** parameter with the value **Student** in the Building Blocks parameter file:

   ```
   sed -i.bak1 's/"adminUsername": ""/"adminUsername": "Student"/' ./reference-architectures/virtual-
   ```

9. Within the SSH session to the Azure VM **lab03vm1**, run the following to replace the placeholder for the **sshPublicKey** parameter with the value of the **$PUBLIC_KEY_REGEX** variable in the Building Blocks parameter file:

   ```
   sed -i.bak2 's/"sshPublicKey": ""/"sshPublicKey": "'"$PUBLIC_KEY_REGEX"'"/' ./reference-architectu
   ```

10. Within the SSH session to the Azure VM **lab03vm1**, run the following to set the value of the size parameter of the virtual machines to **Standard_D2s_v3** in the Building Blocks parameter file:

    ```
    sed -i.bak3 's/"Standard_DS1_v2"/"Standard_D2s_v3"/g' ./reference-architectures/virtual-machines/s
    ```

11. Within the SSH session to the Azure VM **lab03vm1**, run the following to verify that the parameter values were successfully changed in the Building Blocks parameter file:

    ```
    cat ./reference-architectures/virtual-machines/single-vm/parameters/linux/single-vm.json
    ```

12. Within the SSH session to the Azure VM **lab03vm1**, run the following to create a new resource group:

    ```
    az group create --name $RESOURCE_GROUP --location $LOCATION
    ```

13. Within the SSH session to the Azure VM **lab03vm1**, run the following to deploy a Linux Azure VM by using the Azure Building Blocks:

    ```
    azbb -g $RESOURCE_GROUP -s $SUBSCRIPTION_ID -l $LOCATION -p ./reference-architectures/virtual-machi
    ```

14. Wait for the deployment to complete before you proceed to the next task.

16.4.0.7 Task 7: Validate that the Linux Azure VM is serving web content

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click the entry representing the resource group into which you deployed the virtual machine.
3. On the resource group blade, click the entry representing the virtual machine you deployed.
4. On the **Virtual machine** blade, locate the **Public IP address** entry, and identify its value.
5. Open a new Microsoft Edge tab and navigate to the IP address you identified in the previous step.
6. Verify that you are able to access the default Apache2 Ubuntu webpage.
7. Close the new browser tab.
8. Close the **Cloud Shell** pane.

   Review: In this exercise, you deployed Azure VMs running Windows Server 2016 Datacenter and Linux from Cloud Shell by using Azure Building Blocks.

16.5 Exercise 4: Remove lab resources

16.5.0.1 Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. Within the SSH session to the Azure VM **lab03vm1**, run the following to list all resource groups you created in this lab:

   ```
   az group list --query "[?starts_with(name,'AADesignLab03')]".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.

16.5.0.2 Task 2: Delete resource groups

1. Within the SSH session to the Azure VM **lab03vm1**, run the following to delete the resource groups you created in this lab

   ```
   az group list --query "[?starts_with(name,'AADesignLab03')]".name --output tsv | xargs -L1 bash -c
   ```

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

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

17 Networking Azure Application Components

18 Lab Answer Key: Deploying Network Infrastructure for Use in Azure Solutions

18.1 Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
   - Username: **Admin**
   - Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
   - Microsoft Edge
   - File Explorer
   - Visual Studio Code
   - Microsoft Azure Storage Explorer
   - Bash on Ubuntu on Windows
   - Windows PowerShell

     Note: You can also find shortcuts to these applications in the **Start Menu**.
3. Ensure that you have 12 available Standard DSv2 Family vCPUs cores in the region that you will use in the lab. If this is not the case, follow the procedure described in Standard quota: Increase limits by VM series to increase the quotas.

18.2 Exercise 1: Configure the lab environment

18.2.0.1 Task 1: Open the Azure Portal and Cloud Shell.

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** (https://portal.azure.com).
3. When prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.
4. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.

   Note: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.
5. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.

   Note: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.
6. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:

- Leave the **Subscription** drop-down list entry set to its default value.
- In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you intend to deploy resources in this lab
- In the **Resource group** section, select the **Create new** option and then, in the text box below, type **AADesignLab0801-RG**.
- In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
- In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
- Click the **Create storage** button.

7. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

18.2.0.2 Task 2: Deploy an Azure VM running Linux Ubuntu 18.04 that will be used to perform Azure Building Blocks-based deployments.

Note: This is necessary to account for breaking changes affecting running from Cloud Shell.

1. At the **Cloud Shell** command prompt, run the following to create a variable which value designates the name of your Azure subscription:
2. At the **Cloud Shell** command prompt, run the following to create a variable which value designates the name of the resource group you will use in this exercise:

```
RESOURCE_GROUP='AADesignLab0802-RG'
```

3. At the **Cloud Shell** command prompt, run the following to create a variable which value designates the Azure region you will use for the deployment (replace the `<location>` placeholder with the name of the Azure region you want to use in this lab):

```
LOCATION='<location>'
```

4. At the **Cloud Shell** command prompt, run the following to create a resource group that you will use for the deployment:

```
az group create --name $RESOURCE_GROUP --location $LOCATION
```

5. At the **Cloud Shell** command prompt, run the following to deploy an Azure VM running Linux Ubuntu 18.04 that you will use for deploying resources via Azure Building Blocks:

```
UBUNTU_IMAGE='Canonical:UbuntuServer:18.04-LTS:latest'
VM_NAME='lab08vm1'
USER_NAME='student'
az vm create \
--name $VM_NAME \
--resource-group $RESOURCE_GROUP \
--location $LOCATION \
--image $UBUNTU_IMAGE \
--admin-username $USER_NAME \
--generate-ssh-keys \
--size Standard_DS1_v2
```

Note: Wait until the deployment completes.

1. At the **Cloud Shell** command prompt, run the following to retrieve the public IP address of the newly deployed Azure VM **lab08vm1**:

```
IP_ADDRESS=$(az vm show -d --resource-group $RESOURCE_GROUP --name $VM_NAME --query publicIps -o t
```

2. In the **Cloud Shell** pane, run the following to open an SSH session to the newly deployed Azure VM **lab08vm1**:

```
ssh student@$IP_ADDRESS
```

3. In the **Cloud Shell** pane, when prompted whether to continue, type **yes** and press Enter.

18.2.0.3 Task 3: Install the Azure Building Blocks npm package in the Azure VM running Linux.

1. Within the SSH session to the Azure VM **lab08vm1**, run the following to update the list of available packages and their versions:

   ```
   sudo -i apt update
   ```

2. Within the SSH session to the Azure VM **lab08vm1**, run the following to install updated versions of locally installed packages and their dependencies:

   ```
   sudo apt-get upgrade
   ```

3. Within the SSH session to the Azure VM **lab08vm1**, run the following to install Azure CLI:

   ```
   curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
   ```

4. Within the SSH session to the Azure VM **lab08vm1**, run the following to install node.js v10:

   ```
   curl -sL https://deb.nodesource.com/setup_10.x | sudo bash -
   sudo apt-get install -y nodejs
   ```

5. Within the SSH session to the Azure VM **lab08vm1**, run the following to verify the versions of node.js and npm:

   ```
   node --version
   npm --version
   ```

   Note: Verify that the versions are **v10.21.0** and **6.14.4**, respectively.

6. Within the SSH session to the Azure VM **lab08vm1**, run the following to install Azure Building Blocks:

   ```
   sudo su -
   npm install -g @mspnp/azure-building-blocks
   ```

7. Within the SSH session to the Azure VM **lab08vm1**, run the following to exit the root mode and switch to the home directory:

   ```
   exit
   cd $HOME
   ```

8. Within the SSH session to the Azure VM **lab08vm1**, run the following to set the npm path:

   ```
   npm config set prefix '~/.npm-global'
   ```

18.2.0.4 Task 4: Prepare Building Blocks Hub and Spoke parameter files

1. Within the SSH session to the Azure VM **lab08vm1**, run the following to download the GitHub repository containing the Azure Building Blocks reference architecture files:

   ```
   git clone https://github.com/mspnp/reference-architectures.git
   ```

2. Within the SSH session to the Azure VM **lab08vm1**, run the following to view the content of the Azure Building Block parameter file you will use for this deployment:

   ```
   cat ./reference-architectures/hybrid-networking/hub-spoke/hub-spoke.json
   ```

3. Within the SSH session to the Azure VM **lab08vm1**, run the following to change the current directory to the one hosting the **hub-spoke.json** file you will use for this deployment:

   ```
   cd ./reference-architectures/hybrid-networking/hub-spoke
   ```

4. Within the SSH session to the Azure VM **lab08vm1**, type in the following command and press **Enter** to create a new file:

   ```
   vi hub-spoke.json.patch
   ```

5. Within the SSH session to the Azure VM **lab08vm1**, in the vi editor interface, type **a** to initiate the **INSERT** mode, press **Enter** to start a new line, and then type the following:

   ```
   --- hub-spoke.json.orig 2020-06-11 16:08:04.175635600 +0200
   +++ hub-spoke.json  2020-06-11 16:12:07.494153300 +0200
   @@ -271,27 +271,6 @@
                              {
   ```

```
                                "resourceGroupName": "spoke1-vnet-rg",
                                "vmCount": 1,
-                               "namePrefix": "s1jb",
-                               "computerNamePrefix": "s1jb",
-                               "adminUsername": "[replace-with-username]",
-                               "adminPassword": "[replace-with-password]",
-                               "osType": "windows",
-                               "virtualNetwork": {
-                                   "resourceGroupName": "spoke1-vnet-rg",
-                                   "name": "spoke1-vnet"
-                               },
-                               "nics": [
-                                   {
-                                       "isPublic": false,
-                                       "subnetName": "mgmt",
-                                       "privateIPAllocationMethod": "Static",
-                                       "startingIPAddress": "10.1.0.36"
-                                   }
-                               ]
-                           },
-                           {
-                               "resourceGroupName": "spoke1-vnet-rg",
-                               "vmCount": 1,
                                "namePrefix": "s1jbl",
                                "computerNamePrefix": "s1jbl",
                                "adminUsername": "[replace-with-username]",
@@ -330,27 +309,6 @@
                                        "startingIPAddress": "10.2.0.36"
                                    }
                                ]
-                           },
-                           {
-                               "resourceGroupName": "spoke2-vnet-rg",
-                               "vmCount": 1,
-                               "namePrefix": "s2jbl",
-                               "computerNamePrefix": "s2jbl",
-                               "adminUsername": "[replace-with-username]",
-                               "adminPassword": "[replace-with-password]",
-                               "osType": "linux",
-                               "virtualNetwork": {
-                                   "resourceGroupName": "spoke2-vnet-rg",
-                                   "name": "spoke2-vnet"
-                               },
-                               "nics": [
-                                   {
-                                       "isPublic": false,
-                                       "subnetName": "mgmt",
-                                       "privateIPAllocationMethod": "Static",
-                                       "startingIPAddress": "10.2.0.37"
-                                   }
-                               ]
                            }
                        ]
                    },
```

6. Within the SSH session to the Azure VM **lab08vm1**, in the vi editor interface, press **Esc**, press **:**, type **wq!** and press **Enter** to save your changes and close the file.

7. Within the SSH session to the Azure VM **lab08vm1**, run the following to patch the **hub-spoke.json** (two virtual machines have to be removed due to a quota limitation in the Free Azure Pass subscription - 10 virtual machines in total):

```
patch < hub-spoke.json.patch
```

8. Within the SSH session to the Azure VM **lab08vm1**, run the following to replace the placeholder [**replace-with-username**] with the value **Student** in the **hub-spoke.json** Building Blocks parameter file:

```
sed -i.bak1 's/\[replace-with-username\]/Student/g' ./hub-spoke.json
```

9. Within the SSH session to the Azure VM **lab08vm1**, run the following to replace the placeholder [**replace-with-password**] with the value **Pa55w.rd1234** in the **hub-spoke.json** Building Blocks parameter file:

```
sed -i.bak2 's/\[replace-with-password\]/Pa55w.rd1234/g' ./hub-spoke.json
```

10. Within the SSH session to the Azure VM **lab08vm1**, run the following to replace the placeholder [**replace-with-shared-key**] parameter with the value **shared12345** in the **hub-spoke.json** Building Blocks parameter file:

```
sed -i.bak3 's/\[replace-with-shared-key\]/shared12345/g' ./hub-spoke.json
```

11. Within the SSH session to the Azure VM **lab08vm1**, run the following to verify that the parameter values were successfully changed in the **hub-spoke.json** Building Blocks parameter file:

```
cat ./hub-spoke.json
```

18.2.0.5 Task 5: Implement the hub component of the Hub and Spoke design

1. Within the SSH session to the Azure VM **lab08vm1**, run the following to authenticate to your Azure subscription:

```
az login
```

> **Note**: Follow the instructions provided in the output of the command to authenticate to your Azure subscription.

2. Once you authenticated, return to the Cloud Shell pane displaying the SSH session to the Azure VM **lab08vm1**, and run the following to create a variable which value designates the name of your Azure subscription:

```
SUBSCRIPTION_ID=$(az account list --query "[0].id" --output tsv | tr -d '"')
```

3. Within the SSH session to the Azure VM **lab08vm1**, run the following to create a variable which value designates the name of the resource group that will contain the hub virtual network:

```
RESOURCE_GROUP=onprem-vnet-rg
```

4. Within the SSH session to the Azure VM **lab08vm1**, run the following to create a variable which value designates the Azure region you will use for the deployment (replace the placeholder `<Azure region>` with the name of the Azure region to which you intend to deploy resources in this lab):

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

5. Within the SSH session to the Azure VM **lab08vm1**, run the following to deploy the hub component of the Hub-and-Spoke topology by using the Azure Building Blocks:

```
azbb -s  $SUBSCRIPTION_ID -g $RESOURCE_GROUP -l $LOCATION -p ./hub-spoke.json --deploy
```

6. Wait for the deployment to complete but proceed to the next task.

> **Note**: The deployment can take about 40 minutes.

7. In case the Cloud Shell session timed out, reopen it, reestablish the SSH session to the Azure VM **lab08vm1**, and run the following to authenticate to your Azure subscription:

```
az login
```

> **Note**: Follow the instructions provided in the output of the command to authenticate to your Azure subscription.

8. Within the SSH session to the Azure VM **lab08vm1**, and run the following to create a variable which value designates the name of your Azure subscription:

```
SUBSCRIPTION_ID=$(az account list --query "[0].id" --output tsv | tr -d '"')
```

9. Within the SSH session to the Azure VM **lab08vm1**, run the following to create a variable which value designates the name of the resource group that will contain the hub virtual network:

```
RESOURCE_GROUP=onprem-vnet-rg
```

10. Within the SSH session to the Azure VM **lab08vm1**, run the following to create a variable which value designates the Azure region you will use for the deployment (replace the placeholder `<Azure region>` with the name of the Azure region to which you intend to deploy resources in this lab):

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

11. Within the SSH session to the Azure VM **lab08vm1**, run the following to change the current directory to the one hosting the **hub-spoke.json** file you will use for this deployment:

    ```
    cd ./reference-architectures/hybrid-networking/hub-spoke
    ```

12. Within the SSH session to the Azure VM **lab08vm1**, run the following to deploy the hub component of the Hub-and-Spoke topology by using the Azure Building Blocks:

    ```
    azbb -s  $SUBSCRIPTION_ID -g $RESOURCE_GROUP -l $LOCATION -p ./hub-firewall.json --deploy
    ```

13. Wait for the deployment to complete but proceed to the next task.

    Note: The deployment can take about 2 minutes.

18.3 Exercise 2: Review the Hub-spoke topology

18.3.0.1 Task 1: Examine the peering configuration

1. In the hub menu in the Azure portal, click **All services**.
2. In the **All services** menu, in the **Filter** text box, type **Virtual networks** and press **Enter**.
3. In the list of results, click **Virtual networks**.
4. On the **Virtual networks** blade, click **hub-vnet**.
5. On the **hub-vnet** blade, click **Peerings**.
6. On the **hub-vnet - Peerings** blade, review the list of peerings and their status.
7. Navigate back to the **Virtual Networks** blade and click **spoke1-vnet**.
8. On the **spoke1-vnet** blade, click **Peerings**.
9. On the **spoke1-vnet - Peerings** blade, review the existing peering and its status.
10. Navigate back to the **Virtual Networks** blade and click **spoke2-vnet**.
11. On the **spoke2-vnet** blade, click **Peerings**.
12. On the **spoke2-vnet - Peerings** blade, review the existing peering and its status.

18.3.0.2 Task 2: Examine the routing configuration

1. In the **All services** menu, in the **Filter** text box, type **Route tables** and press **Enter**.
2. In the list of results, click **Route tables**.
3. On the **Route tables** blade, click **spoke1-rt**.
4. On the **spoke1-rt** blade, review the list of routes. Note the **NEXT HOP** entry for the route **toSpoke2**.
5. Navigate back to the **Route tables** blade and click **spoke2-rt**.
6. On the **spoke2-rt** blade, review the list of routes. Note the **NEXT HOP** entry for the route **toSpoke1**.

18.3.0.3 Task 3: Verify connectivity between spokes

1. In the hub menu in the Azure portal, click **All services**.
2. In the **All services** menu, in the **Filter** text box, type **Network Watcher** and press **Enter**.
3. In the list of results, click **Network Watcher**.
4. On the **Network Watcher** blade, in the **NETWORK DIAGNOSTIC TOOLS** section, click **Connection troubleshoot**.
5. On the **Network Watcher - Connection troubleshoot** blade, perform the following tasks:

- Leave the **Subscription** drop-down list entry set to its default value.
- In the **Resource group** drop-down list, select the **AADesignLab08-spoke1-vnet-rg** entry.
- In the **Virtual machine** drop-down list, leave the default entry.
- Ensure that the **Destination** option is set to **Specify manually**.
- In the **URI, FQDN, or IPv4** text box, type **10.2.0.68** entry.
- In the **Destination Port** text, type 3389.
- Click the **Check** button.

6. Wait until results of the connectivity check are returned and verify that the status is **Reachable**.

   Note: If this is the first time you are using Network Watcher, the check can take up to 5 minutes.

18.4 Exercise 3: Remove lab resources

18.4.0.1 Task 1: Open Cloud Shell

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

   ```
   az group list --query "[?starts_with(name,'AADesignLab08')]".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.

18.4.0.2 Task 2: Delete resource groups

1. In the Cloud Shell pane, run the following command and press **Enter** to delete the resource groups you created in this lab

   ```
   az group list --query "[?starts_with(name,'AADesignLab08')]".name --output tsv | xargs -L1 bash -c
   ```

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

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

19 Integrating Azure Solution Components using Messaging Services

20 Lab Answer Key: Deploying Messaging components to facilitate communication between Azure resources

20.1 Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
   - Username: **Admin**
   - Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
   - Microsoft Edge
   - File Explorer
   - Visual Studio Code
   - Microsoft Azure Storage Explorer
   - Bash on Ubuntu on Windows
   - Windows PowerShell

Note: You can also find shortcuts to these applications in the **Start Menu**.

20.2 Exercise 1: Deploy a Service Bus namespace

20.2.0.1 Task 1: Open the Azure portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** (https://portal.azure.com).
3. When prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

20.2.0.2 Task 2: Create a Service Bus namespace

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Service Bus** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Service Bus**.
4. On the **Service Bus** blade, click the **Create** button.
5. On the **Create namespace** blade, perform the following tasks:
   - In the **Name** text box, enter a globally unique name.
   - In the **Pricing tier** drop-down list, select the **Basic** option.
   - Leave the **Subscription** drop-down list entry set to its default value.
   - In the **Resource group** section, ensure that the **Create new** option is selected and then, in the text box, type **AADesignLab1101-RG**.
   - In the **Location** drop-down list, select the Azure region to which you intend to deploy resources in this lab.
   - Click the **Create** button.
6. Wait for the provisioning to complete before you proceed to the next step.

20.2.0.3 Task 3: Create a Service Bus Queue

1. In the hub menu of the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab1101-RG**.
3. On the **AADesignLab1101-RG** blade, click the newly created Service Bus namespace.
4. On the Service Bus namespace blade, in the **ENTITIES** section, click **Queues**.
5. On the Service Bus namespace blade, click the **+ Queue** button.
6. In the **Create queue** pane, perform the following tasks:
   - In the **Name** text box, type **messages**.
   - Leave all remaining settings with their default values.
   - Click the **Create** button.

20.2.0.4 Task 4: Get Service Bus Connection String

1. Back on the Service Bus namespace blade, click **Shared access policies**.
2. On the Service Bus namespace blade, click the **RootManageSharedAccessKey** policy.
3. In the **SAS Policy: RootManageSharedAccessKey** pane, locate and record the value of the **Primary Connection String** field. You will use this value later in this lab.

Review: In this exercise, you created a new Service Bus namespace and recorded a connection string to access queues in the namespace.

20.3 Exercise 2: Create a logic app

20.3.0.1 Task 1: Create an Azure Storage account

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Storage Account** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Storage Account**.
4. On the **Storage Account** blade, click the **Create** button.
5. On the **Create storage account** blade, perform the following tasks (leave all other settings with their default values):
   - On the **Basics** tab, in the **Resource group** section, in the drop-down list, select the resource group you created earlier in this exercise.
   - On the **Basics** tab, in the **Storage account name** text box, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   - On the **Basics** tab, ensure that the **Location** entry is set to the same Azure region you selected earlier in this exercise.
   - On the **Basics** tab, in the **Performance** section, ensure that the **Standard** option is selected.
   - On the **Basics** tab, in the **Account kind** drop-down list, ensure that the **Storage (general purpose v1)** option is selected.
   - In the **Replication** drop-down list, select the **Locally-redundant storage (LRS)** entry.
   - On the **Networking** tab, ensure that the **Connectivity method** option set to **Public endpoint (all networks)**.
   - On the **Advanced** tab, in the **Secure transfer required** section, select the **Disabled** option.
   - On the **Advanced** tab, in the **Blob public access** section, select the **Enabled** option.
   - Click the **Review + create** button and then click the **Create** button.
6. Wait for the provisioning to complete before you proceed to the next step.
7. In the hub menu of the Azure portal, click **Resource groups**.
8. On the **Resource groups** blade, click **AADesignLab1101-RG**.
9. On the **AADesignLab1101-RG** blade, click the newly created Azure Storage account.
10. On the Storage account blade, click the **Containers** tile.
11. On the Storage account blade, click the **+ Container** button.
12. In the **New container** pane, perform the following tasks:
    - In the **Name** text box, type **messageoutput**.
    - In the **Public access level** drop-down list, select the **Blob (anonymous read access for blobs only)** option.
    - Click the **Create** button.

20.3.0.2 Task 2: Create a logic app

1. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Logic App** and press **Enter**.
2. On the **Everything** blade, in the search results, click **Logic App**.
3. On the **Logic App** blade, click the **Create** button.
4. On the **Create logic app** blade, perform the following tasks:
   - In the **Name** text box, type **ServiceBusWorkflow**.
   - Leave the **Subscription** drop-down list entry set to its default value.

- In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab1101-RG**.
- In the **Location** drop-down list, select the same Azure region you chose in the previous task.
- In the **Log Analytics** section, ensure that the **Off** button is selected.
- Click the **Review + create** button and then click the **Create** button.

5. Wait for the provisioning to complete before you proceed to the next task.

20.3.0.3 Task 3: Configure logic app steps.

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab1101-RG**.
3. On the **AADesignLab1101-RG** blade, click the entry representing the logic app you created in the previous task.
4. On the **Logic Apps Designer** blade, scroll down and click the **Blank Logic App** tile in the **Templates** section.
5. On the **Logic Apps Designer** blade, perform the following tasks:
   - In the **Search connectors and triggers** text box, type **Service Bus**.
   - In the search results, select the trigger named **When a message is received in a queue (auto-complete) - Service Bus**.
   - In the **Connection Name** text box, type **ServiceBusConnection**.
   - In the list of **Service Bus namespaces**, select the namespace you created earlier in this lab.
   - In the list of policies, select the **RootManageSharedAccessKey** policy.
   - Click the **Create** button.
6. In the **When a message is received in a queue (auto-complete)** step, perform the following tasks:
   - In the **Queue name** drop-down list, select the **messages** entry.
   - In the **Interval** text box, type **30**.
   - In the **Frequency** drop-down list, select the **Second** entry.
7. On the **Logic Apps Designer** blade, click the **+ New Step** button.
8. On the **Logic Apps Designer** blade, perform the following tasks:
   - In the **Search connectors and actions** text box, type **Storage blob**.
   - In the search results, select the action named **Create blob - Azure Blob Storage**.
   - In the **Connection Name** text box, type **StorageConnection**.
   - In the list of *Storage accounts*, select the account you created earlier in this lab.
   - Click the **Create** button.
9. In the **Create Blob** step, perform the following tasks:
   - In the **Folder path** text box, type **/messageoutput**.
   - In the **Blob name** text box, type **@concat(triggerBody()?['MessageId'], '.txt')**.
   - In the **Blob content** text box, type **@string(decodeBase64(triggerBody()?['ContentData']))**.
10. At the top of the **Logic Apps Designer** blade, click the **Save** button to persist your workflow.

20.3.0.4 Task 4: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.

   Note: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.

2. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.

   Note: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.

3. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:

   - Leave the **Subscription** drop-down list entry set to its default value.
   - In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you deployed resources in this lab.
   - In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab1101-RG**.
   - In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   - In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
   - Click the **Create storage** button.

4. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

20.3.0.5 Task 5: Validate Logic App using Node.js

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.
2. At the **Cloud Shell** command prompt at the bottom of the portal, type in the following command and press **Enter** to install the **azure** package using NPM:

   ```
   npm install azure
   ```

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to open the interactive node terminal:

   ```
   node
   ```

4. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to import the **azure** module in Node:

   ```
   var azure = require('azure');
   ```

   Note: The output will show **undefined**. This is expected.

5. At the **Cloud Shell** command prompt, type in the following command (replacing the placeholder `<Service Bus namespace connection string>` with the value of your url you recorded earlier in this lab) and press **Enter** to create a new variable for your Service Bus namespace connection string:

   ```
   var connectionString = '<Service Bus namespace connection string>';
   ```

6. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new client to connect to the Service Bus namespace:

   ```
   var serviceBusService = azure.createServiceBusService(connectionString);
   ```

7. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to send a message to Service Bus namespace queue using the client.

   ```
   serviceBusService.sendQueueMessage('messages', { body: 'Hello World' }, function(error) { console.
   ```

8. In the hub menu of the Azure portal, click **Resource groups**.
9. On the **Resource groups** blade, click **AADesignLab1101-RG**.
10. On the **AADesignLab1101-RG** blade, click the Azure Storage account you created earlier in this lab.
11. On the Storage account blade, click the **Containers** tile.
12. On the Storage account container blade, click the **messageoutput** container.
13. Note the newly created blob in your container and select **Edit** to view its content.

    Review: In this exercise, you created a logic app that is triggered by messages from a queue in a Service Bus namespace.

20.4 Exercise 3: Remove lab resources

20.4.0.1 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,'AADesignLab11')]".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.

20.4.0.2 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,'AADesignLab11')]".name --output tsv | xargs -L1 bash -c
   ```

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

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