Online Lab - Creating Managed Server Applications in Azure

Topic: Deploying Managed Containerized Workloads to Azure

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**.

Exercise 1: Create Azure Kubernetes Service (AKS) cluster

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.

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.

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):

   ```
   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-size Standard_D1_v2 --generate-ssh-
keys
```

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.

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

Note: This operation can take up to 10 minutes.

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.

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

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 run aad0402-akscluster --image=nginx --replicas=1 --port=80
```

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

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 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.

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
```

Exercise 3: Autoscaling pods in an AKS cluster

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 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.
10. At the Cloud Shell command prompt, type in the following command and press **Enter** to return to the home directory:

    cd ~

Task 2: Autoscale Kubernetes pods.

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/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 create -f ~/metrics-server/deploy/1.8+/
```

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 incrased 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

Exercise 4: Implement DevOps with AKS

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 AADesignLab0403-SP)
```

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 .appId | tr -d '"')
```

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 .password | tr -d '"')
```

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 ~/github.json
```

11. At the **Cloud Shell** command prompt, in the vi editor interface, add the content of the sample parameters file (**F:\Labfiles\Mod04\Starter\parameters.json**):

```
{ "$schema": "http://schema.management.azure.com/schemas/2015-01-
01/deploymentParameters.json#", "contentVersion": "1.0.0.0",
"parameters": { "webAppName": { "value": "$" }, "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 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.githubusercontent.com/mspnp/solution-
    architectures/master/apps/devops-with-aks/azuredeploy.json --parameters
    @parameters.json
    ```

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

Note: The deployment can take up to 15 minutes.

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 Azure VMs running Windows Server 2016 Datacenter and Linux from Cloud Shell by using Azure Building Blocks.

Exercise 5: Remove lab resources

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.

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 'az group delete --name $0 --no-wait --
yes'
```

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

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