Contents

Lab Guide: Module 5 - Container Orchestrators

Exercise 1: Create and Scale Azure Kubernetes Service Cluster

The goal of this exercise is to show attendees how to create an AKS cluster using the az CLI, as well as to deploy a public image in it.

Tasks

1. **Create Windows Kubernetes Cluster on Azure Kubernetes Services**

    1. Navigate to portal.azure.com
    2. On Windows server, open PowerShell.
    3. Login to Azure (the command line will give you a URL and a code, put those into your browser to login. Once you are logged in, then close the browser and go back to command line, click on the command line window, hit enter, and you will see your subscriptions).

        ```
        az login
        ```

    4. You should see a list of subscriptions once you are logged in. Set the correct subscription, put the name of the subscription in quotes.

        ```
        az account set --subscription “your azure subscrioption name goes here”
        ```

    5. Create a resource group named (replace initials with your initials so the resource group name is unique) ***k8s-win-cluster-rg-wk***

        ```
        az group create --name=k8s-aks-cluster-rg-wk --location=eastus
        ```

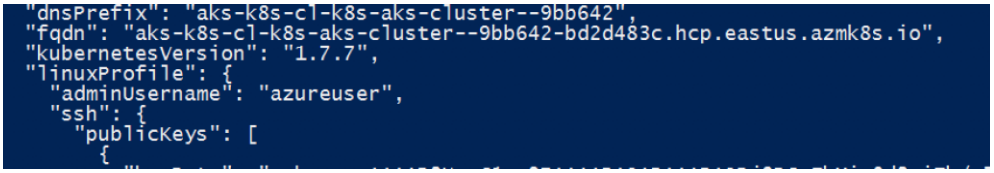
    6. Run the following command to deploy Kubernetes cluster with AKS (replace initials with your initials).

        ```
        az aks create --resource-group k8s-aks-cluster-rg-wk --name aks-k8s-cluster --disable-rbac --node-count 2 --node-vm-size "Standard_D2_v3" --generate-ssh-keys
        ```

> Note that this command will generate public/private ssh keys in c:\users\super\.ssh folder.
>
> It may take approximately 10 minutes for the cluster to provision successfully.

    7. After the cluster is provisioned successfully you will be shown JSON output describing the AKS cluster.

8. **Locate and copy the value of "fdqn" attribute from the JSON output.** Note: in the "agentPoolProfiles" there is a fqdn that is null, ignore this one, and look further down and you will see a second fqdn and this in the one you will need. Note that your fdqn value may differ from the output shown below.



```
"dnsPrefix": "aks-k8s-cl-k8s-aks-cluster--9bb642",
"fqdn": "aks-k8s-cl-k8s-aks-cluster--9bb642-bd2d483c.hcp.eastus.azmk8s.io",
"kubernetesVersion": "1.7.7",
"linuxProfile": {
  "adminUsername": "azureuser",
  "ssh": {
    "publicKeys": [
      {
```

9. **Install kubectl tool which allows you to run commands against Kubernetes cluster. Skip this step if it is already installed on your machine (run "kubectl" to see if the command is recognized).**

   Windows:

   ```
   choco install -y kubernetes-cli
   ```

   Linux:

   ```
   sudo apt-get -y update && sudo apt-get -y upgrade
   ```

   ```
   curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl
   ```

   ```
   chmod +x ./kubectl
   ```

   ```
   sudo mv ./kubectl /usr/local/bin/kubectl
   ```

   In Linux, due to current bug with azure-cli you need to revert to 2.0.23-1 version.

   ```
   sudo apt-get -y purge azure-cli
   ```

   ```
   sudo apt-get -y install azure-cli=2.0.23-1
   ```

10. **Run the following command to download the Kubernetes cluster configuration to the local config file *C:\users\super\.kube\config***

    ```
    az aks get-credentials --resource-group k8s-aks-cluster-rg-wk --name aks-k8s-cluster
    ```

> Be aware that you will need the content of *.kube/config* file while creating Kubernetes service endpoint on Azure Devops Services in the DevOps module.

2. Demonstrate how to deploy resources through the kubectl CLI

1. Run following command to ensure context is set to the correct cluster:

```
kubectl config set-context aks-k8s-cluster
```

2. You will now test the cluster by running a nginx container. First create a new deployment using the following command:

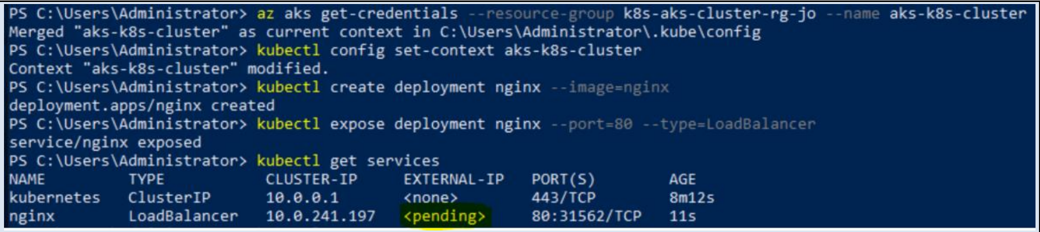
```
kubectl create deployment nginx --image=nginx
```

3. Next, make sure that you can access the container from the external (public IP). To do that use the expose command to expose port 80 and enable the external IP (type=LoadBalancer).

```
kubectl expose deployment nginx --port=80 --
type=LoadBalancer
```

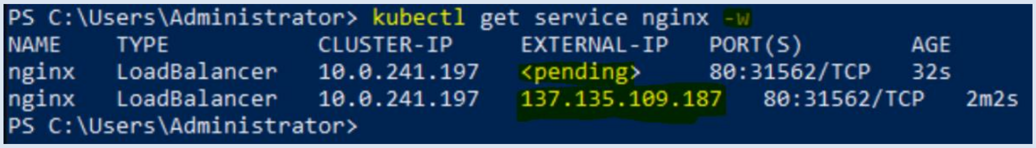
4. The above command essentially creates a service with the name nginx. You can view the service by running following command

```
kubectl get service nginx
```



```
PS C:\Users\Administrator> az aks get-credentials --resource-group k8s-aks-cluster-rg-jo --name aks-k8s-cluster
Merged "aks-k8s-cluster" as current context in C:\Users\Administrator\.kube\config
PS C:\Users\Administrator> kubectl config set-context aks-k8s-cluster
Context "aks-k8s-cluster" modified.
PS C:\Users\Administrator> kubectl create deployment nginx --image=nginx
deployment.apps/nginx created
PS C:\Users\Administrator> kubectl expose deployment nginx --port=80 --type=LoadBalancer
service/nginx exposed
PS C:\Users\Administrator> kubectl get services
NAME         TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP      10.0.0.1       <none>        443/TCP        8m12s
nginx        LoadBalancer   10.0.241.197   <pending>     80:31562/TCP   11s
```

Please wait until EXTERNAL-IP for nginx service change from <pending> to a valid IP address. It may take few minutes and you **may have to run the command *kubectl get service nginx* few times to probe the status of external IP assignment.** Another useful parameter is **-w** that can be added to the command to keep watching the service as it changes. When it is done, it will look like below:



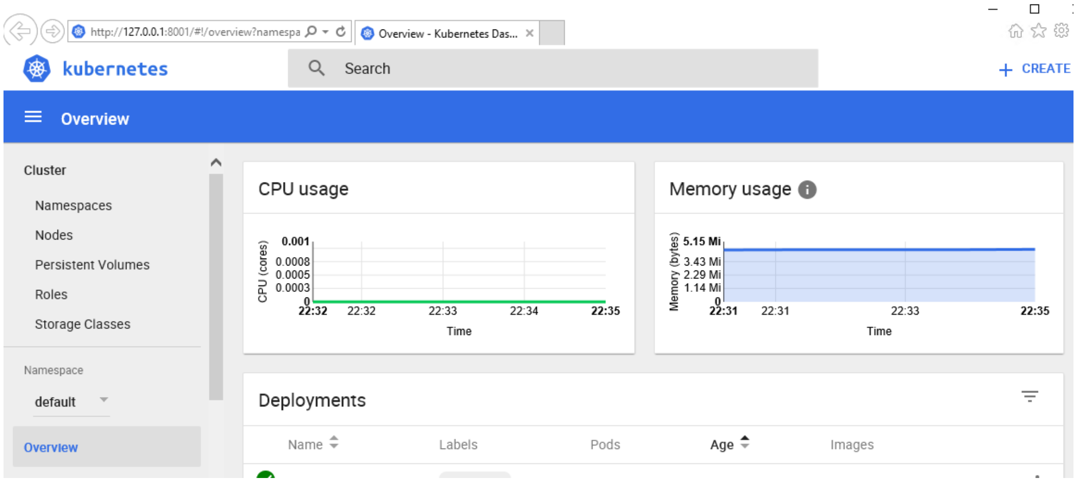
```
PS C:\Users\Administrator> kubectl get service nginx -w
NAME    TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
nginx   LoadBalancer   10.0.241.197   <pending>     80:31562/TCP   32s
nginx   LoadBalancer   10.0.241.197   137.135.109.187   80:31562/TCP   2m2s
PS C:\Users\Administrator>
```

5. You can now simply query the content hosted by nginx by using the curl command:

```
curl http://$( kubectl get service nginx -
o=jsonpath='{.status.loadBalancer.ingress[*].ip}')
```

6. Kubernetes also provide UI in the form of dashboard. To access the dashboard run the following command. Please **refresh the opened browser if needed after a couple of seconds.**

```
az aks browse --resource-group k8s-aks-cluster-rg-wk --name
aks-k8s-cluster
```



You can always access the dashboard by browsing to the URL: http://127.0.01:8001.

7. **Stop running by hitting "Ctrl + C", then "Y" to terminate the batch job**.
8. Let's check total number of pods running now. You should see one pod ready with a status of running.

   ```
   kubectl get pods
   ```

9. You will now scale the number of pods using replicaset. You will get more details about the replica set using the following command:

   ```
   kubectl get replicaset
   ```

10. To scale, run the command and pass the name of deployment that was created earlier:

    ```
    kubectl scale --replicas=3 deployment/nginx
    ```

11. Now if you run the command to get the number of pods running after scaling it should return three pods (earlier it was one). Note the name of the nodes running the pods as well as pods' IP.

    ```
    kubectl get pods -o wide
    ```

12. In pervious step you have successfully increased the number of pods by increasing the replica set. However, all the pods are still running on two nodes node since the cluster was initially created with two nodes. You can also check that by running the following command:

    ```
    kubectl get nodes
    ```

In addition to the number of pods, we might want to adjust the compute capacity of the cluster itself. For instance, to add a worker node to the cluster, we can run the following command:

NOTE: you might want to only show the following few commands as it takes a long time to complete.

```
az aks scale --node-count=3 --resource-group k8s-aks-
cluster-rg-wk --name aks-k8s-cluster
```

13. If you check the number of nodes again, you should see three worker nodes instead of two running.

Next time when you scale the pods using replica set, third worker node will also participate. You can also look at the maximum pod capacity of a node by running the command:

kubectl get node NODE-NAME -o=jsonpath='{.status.capacity.pods}'

14. To scale back down to a node count of 2, run the following command:

```
az aks scale --node-count=2 --resource-group k8s-aks-
cluster-rg-wk --name aks-k8s-cluster
```

15. Finally, remove the deployment and service using the commands below:

```
kubectl delete deployment nginx
```

```
kubectl delete service nginx
```