Workshop – Containers & AKS

Prerequisites:

1. Azure Subscription
2. Internet Connection
3. Remote Desktop client (Built in Windows)
4. Web Browser (Any)

Section 1: Tools Installations

Setting up a VM on Azure is the easiest way to kickstart workshops and provides a clean development environment to start with. However, if you prefer to work with your own setup, feel free to do that; and in that case *skip irrelevant portions of Section 1 of this Setup*

Create Windows 10 VM

- Login into your azure portal (https://portal.azure.com)
- Click on *Create Resource* to start creating new windows VM. Choose one *Windows 10 Pro*
- On Next screen, click *Create* button to start VM Creation
- In Basics tab, select an existing resource group or use *Create new* button to create new one. Then enter name of new VM : *Windows10VM* **(** *or anything of your choice* **)** and then scroll down for more settings
- On next section, provide user credentials and port to be open for *RDP* access.
- Click Next – *Disk.* Click Next*: Networking* to skip the disk configuration (Accept defaults).
- Click Next - *Management* to skip networking (Accept defaults)
- Turn off both diagnostics and click *Review + Create*
- On final page, click *Create* after validation is passed
- Wait for VM Provisioning to finish

Install Docker CE for Windows

- Once VM is Ready, Click on *Connect* button to start RDP Session
- Choose *Use different account* in login box and then enter *username* & *password*. You may have to accept server certificate to begin session.
- Go to Start menu and click on *Windows PowerShell*
- Install VirtualBox (https://www.virtualbox.org/wiki/Downloads) Or Hyper-V (https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/quick-start/enable-hyper-v)
- (for the above selected VM, you might just need to Enable Hyper-V from *turn On/Off windows features*)
- Install *Docker For Windows* from: https://docs.docker.com/v17.09/docker-for-windows/install/
- This should install docker-compose as well. Check this by running - *docker-compose version* from PowerShell
- Test Docker Installation – *docker version*

Install Azure CLI for Windows

- Go to link: https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-windows?view=azure-cli-latest and install CLI
- Check Installation – *az –version*

Install Kubectl for Windows

- In *PowerShell* : *Install-Script -Name install-kubectl -Scope CurrentUser -Force*
- (Specify a *DownloadLocation*):
- *install-kubectl.ps1 [-DownloadLocation <path>]*
- Note: If you do not specify a *DownloadLocation*, *kubectl* will be installed in the user's temp Directory.
- The installer creates *$HOME/.kube* and instructs it to create a config file
- Test to ensure the version you installed is sufficiently up-to-date: *kubectl version*

  OR

  You can follow instructions at: https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-kubectl

Install VSCode for Windows

This is primarily for better editing on the Windows VM. If you are using your own machine/setup then you can use any other editor of your choice

https://code.visualstudio.com/download

Install Helm for Windows

K8s de-facto package manager: https://docs.helm.sh/using_helm/#from-the-binary-releases

Section 2: Environment Setup

Note: Please create a working folder in. your local directory – *ASKSChallenge*. *CD* into it.

We would run all our examples/downloads into this directory

Create an AKS cluster

- Get the latest available Kubernetes version

  region=<targeted AKS region>

  az aks get-versions -l $region -o table

  $kubernetesVersionLatest=az aks get-versions -l ${region} --query 'orchestrators[-1].orchestratorVersion' -o tsv

- Create a Resource Group

  az group create --name akschallenge --location $region

- Create AKS using the latest version and enable the monitoring addon

  az aks create --resource-group akschallenge --name <unique-aks-cluster-name> --enable-addons monitoring --kubernetes-version $kubernetesVersionLatest --generate-ssh-keys --location eastus

  Ensure you can connect to the cluster using *kubectl*

- Authenticate

  az aks get-credentials --resource-group akschallenge --name <unique-aks-cluster-name>

- List the available nodes

  kubectl get nodes

- *Deploy an instance of MongoDB to your cluster. The application expects a database called akschallenge*

  If the cluster is RBAC enabled, you have to create the appropriate ServiceAccount for Tiller (the server side Helm component) to use.

  - YAML link - *https://aksworkshop.io/yaml-solutions/01.%20challenge-02/helm-rbac.yaml*

  - Deploy it using

    kubectl apply -f helm-rbac.yaml
    Initialize Tiller (ommit the --service-account flag if your cluster is not RBAC enabled)
    helm init –upgrade --service-account tiller

  - Install MongoDB using Helm chart

    helm install stable/mongodb --name orders-mongo --set mongodbUsername=orders-user,mongodbPassword=orders-password,mongodbDatabase=akschallenge

Create Azure Container Registry

- Go to your resource group in Azure Portal
- Select *Add* and then *Container Registry*
- Follow instructions to Create the ACR. This will take few minutes.
- Once created, Open *Access Keys* section in the portal and note down the details

Alternate using PowerShell:

az acr create --resource-group akschallenge --name <unique-acr-name> --sku Standard --location <location>

Deploy the Order Capture API

- Source Link: https://hub.docker.com/r/azch/captureorder/

- YAML link: https://aksworkshop.io/yaml-solutions/01.%20challenge-02/captureorder-deployment.yaml

  kubectl apply -f captureorder-deployment.yaml

- Verify that the pods are up and running

  kubectl get pods -l app=captureorder

- Retrieve the External-IP of the Service

  kubectl get service captureorder (note down the IP address)

Deploy the frontend using Ingress

- Source Link: https://github.com/Azure/azch-frontend

- YAML link: https://aksworkshop.io/yaml-solutions/01.%20challenge-02/frontend-deployment.yaml

  kubectl apply -f frontend-deployment.yaml

- Verify that the pods are up and running

  kubectl get pods -l app=frontend

Expose the frontend on a hostname

- Enable the HTTP routing add-on on your cluster

  az aks enable-addons --resource-group akschallenge --name <unique-aks-cluster-name> --addons http_application_routing

- YAML link: https://aksworkshop.io/yaml-solutions/01.%20challenge-02/frontend-service.yaml

  kubectl apply -f frontend-service.yaml

- Ingress*:*

  az aks show --resource-group akschallenge --name <unique-aks-cluster-name> --query addonProfiles.httpApplicationRouting.config.HTTPApplicationRoutingZoneName -o table

  https://aksworkshop.io/yaml-solutions/01.%20challenge-02/frontend-ingress.yaml

  kubectl apply -f frontend-ingress.yaml

- *Display App:*

  http://frontend.9f9c1fe7-21a1-416d-99cd-3543bb92e4c3.eastus.aksapp.io

Monitoring

- Primarily done by Insights and Log Analytics in the portal
- Follow discussion on this during workshop and try out various options

Scaling

- Run Load Test:

az container create -g akschallenge -n loadtest --image azch/loadtest --restart-policy Never -e SERVICE_IP=<public ip of order capture service>

- Check Container Logs:

az container logs -g akschallenge -n loadtest

Or in Portal

- Create Horizontal Pod Autoscaler:

  YAML link: https://aksworkshop.io/yaml-solutions/01.%20challenge-04/captureorder-hpa.yaml

  az container delete -g akschallenge -n loadtest

  az container create -g akschallenge -n loadtest --image azch/loadtest --restart-policy Never -e SERVICE_IP=<public ip of order capture service>

  kubectl get pods -l

  az container delete -g akschallenge -n loadtest

DEVOPS

- Login to the registry

  az acr login --name <unique-acr-name>

- Clone the application code on Azure Cloud Shell

  git clone https://github.com/Azure/azch-captureorder.git

- cd *azch-captureorder*
- Use Azure Container Registry Build to build and push the container images

  az acr build -t "captureorder:{{.Run.ID}}" -r <unique-acr-name> .

- Create Kubernetes secret

  kubectl create secret docker-registry acr-auth --docker-server <acr-login-server> --docker-username <service-principal-ID> --docker-password <service-principal-password> --docker-email <email-address>

- Update your deployment with a reference to the created secret

```
spec:

  imagePullSecrets:

  - name: acr-auth

  containers:
```

- Edit deployment

  kubectl edit deploy

Follow instructions in the session to complete the DevOps exercise