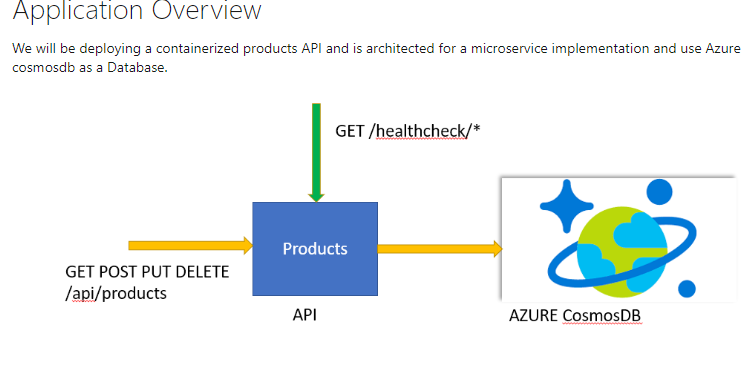
Introduction

Application Overview

We will be deploying a containerized products API and is architected for a microservice implementation and use Azure cosmosdb as a Database.



The application consists of 2 components

* A public facing Products swagger enabled API
* A Azure cosmosdb Database

Healthcheck APIs are enabled and configured to check the application health as well as database.

Application dependencies

Application used in the labs are depends on following components

* Azure CosmosDB
* Application Insights
* Key Vault

**Kubernetes** is an open-source container-orchestration system for automating application deployment, scaling, and management. It was originally designed by Google, and is now maintained by the Cloud Native Computing Foundation.

In this workshop, We can go through tasks that will help to understand the basics and some advanced topics to deploy application to kubernetes on **Azure Kubernetes Service (AKS)**. Azure Kubernetes Service (AKS) is a managed container orchestration service, based on the open source Kubernetes system, which is available on the Microsoft Azure public cloud. An organization can use AKS to deploy, scale and manage Docker containers and container-based applications across a cluster of container hosts

The Network topology of the different Azure services we have already deployed for this workshop is given below

Prerequisites

Before beginning this hands on workshp please install all the Tools given in the Prerequisites page below on your laptop.

[Prerequisites](https://dev.azure.com/cloudiq/K8S-Dev-Workshop/_git/k8s-dev-workshop?path=%2FREADME.md&version=GBmaster#path=%2Fassets%2Fdotnet-core%2Fprerequisites.md&version=GBmaster&createIfNew=true&_a=preview)

Verify Connectivity

We will be connecting to AKS using Azure CLI and **kubectl**. Kubectl is a command line interface for running commands against Kubernetes clusters.

* Enter the below command in your Windows command prompt or Mac Terminal to connect to Azure account

$ az login

*NOTE:* It is a good practice to use az account clear before az login in case you regularly use az cli on your device.

This will open a browser screen asking you to enter your Azure user id and password.

* Next we have to establish Connectivity with the AKS cluster by executing the below command

$ az aks get-credentials --resource-group <resource-group-name> --name <AKS service name>

*NOTE:* <AKS Service Name> follows the format ciq-trng-eastus-app-<team no>-aks; so if you are in team 1 the command will be:

az aks get-credentials --resource-group ciq-trng-eastus-team1-rg --name ciq-trng-eastus-app-team1-aks

This command gets the credentials for the AKS service and sets it as the current context for kubectl in the file $HOME/.kube/config Once the above command completes successfully, you can verify the Connectivity using below command:

$ kubectl cluster-info

Kubernetes master is running at https://xxx.westus2.azmk8s.io:443

healthmodel-replicaset-service is running at https://xxx.westus2.azmk8s.io:443/api/v1/namespaces/kube-system/services/healthmodel-replicaset-service/proxy

CoreDNS is running at https://xxx.westus2.azmk8s.io:443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

kubernetes-dashboard is running at https://xxx.westus2.azmk8s.io:443/api/v1/namespaces/kube-system/services/kubernetes-dashboard/proxy

Metrics-server is running at https://xxx.westus2.azmk8s.io:443/api/v1/namespaces/kube-system/services/https:metrics-server:/proxy

* kubectl commands - get nodes, get namespace

A **node** is a worker machine in Kubernetes. A node may be a VM or physical machine, depending on the cluster. To get the list of all nudes in the AKS cluster, run the command:

$ kubectl get nodes

NAME STATUS ROLES AGE VERSION

aks-nodepool1-xxxxxxxx-x Ready agent 99d v1.13.5

Kubernetes supports multiple virtual clusters backed by the same physical cluster. These virtual clusters are called **namespaces**. To get the list of namespaces available in the AKS cluster run the command:

$ kubectl get namespace

NAME STATUS AGE

default Active 99d

kube-public Active 99d

kube-system Active 99d

kv Active 8d

**Kubernetes dashboard**

Kubernetes includes a web dashboard that can be used for basic management operations. This dashboard lets you view basic health status and metrics for your applications, create and deploy services, and edit existing applications.

To start the Kubernetes dashboard, use the az aks browse command. The following example opens the dashboard for the cluster named *myAKSCluster* in the resource group named *myResourceGroup*:

$ az aks browse --resource-group myResourceGroup --name myAKSCluster

This command creates a proxy between your development system and the Kubernetes API, and opens a web browser to the Kubernetes dashboard. If a web browser doesn't open to the Kubernetes dashboard, copy and paste the URL address noted in the Azure CLI, typically http://127.0.0.1:8001

**Helm**

Helm is the package manager for Kubernetes. It helps us to find, share, and use software built for Kubernetes. There are two parts to Helm: The Helm client (helm) and the Helm server (Tiller).

For this workshop we have enabled a feature called Role-Based Access Control - RBAC for short. We have already installed Tiller with a service account at server end. So, *do not run helm init on your laptop*.

To see what packages have been release using Helm execute the below command

$ helm ls

Exercise 0: Kubernetes deployments basics

[Kubernetes deployments basics](https://dev.azure.com/cloudiq/K8S-Dev-Workshop/_git/k8s-dev-workshop?path=%2FREADME.md&version=GBmaster#path=%2Fassets%2Fdotnet-core%2Fk8s-deployment-basics.md&version=GBmaster&createIfNew=true&_a=preview)

Application overview

[Application overview](https://dev.azure.com/cloudiq/K8S-Dev-Workshop/_git/k8s-dev-workshop?path=%2FREADME.md&version=GBmaster#path=%2Fassets%2Fdotnet-core%2Fapp-overview.md&version=GBmaster&createIfNew=true&_a=preview)

Exercise 1: Docker application in local

* NOTE \*: Spring boot will only be Demoed, please stick to the dotnet core for lab exercises.

[Application local deployment](https://dev.azure.com/cloudiq/K8S-Dev-Workshop/_git/k8s-dev-workshop?path=%2FREADME.md&version=GBmaster#path=%2Fassets%2Fdotnet-core%2Fapp-deployment-local.md&version=GBmaster&createIfNew=true&_a=preview)

Exercise 2: Pushing Docker Images to ACR

[ACR push images](https://dev.azure.com/cloudiq/K8S-Dev-Workshop/_git/k8s-dev-workshop?path=%2FREADME.md&version=GBmaster#path=%2Fassets%2Fdotnet-core%2Facr-operations.md&version=GBmaster&createIfNew=true&_a=preview)

Exercise 3: Application deployment on AKS

[Application deployment on AKS](https://dev.azure.com/cloudiq/K8S-Dev-Workshop/_git/k8s-dev-workshop?path=%2FREADME.md&version=GBmaster#path=%2Fassets%2Fdotnet-core%2Fapp-deployment-k8s.md&version=GBmaster&createIfNew=true&_a=preview)

Exercise 4: Ingress deployment on AKS

[Ingress deployment](https://dev.azure.com/cloudiq/K8S-Dev-Workshop/_git/k8s-dev-workshop?path=%2FREADME.md&version=GBmaster#path=%2Fassets%2Fdotnet-core%2Fapp-deployment-ingress.md&version=GBmaster&createIfNew=true&_a=preview)

Exercise 5: Application deployment using Helm charts on AKS

[Helm deployment](https://dev.azure.com/cloudiq/K8S-Dev-Workshop/_git/k8s-dev-workshop?path=%2FREADME.md&version=GBmaster#path=%2Fassets%2Fdotnet-core%2Fapp-deployment-helm.md&version=GBmaster&createIfNew=true&_a=preview)

Exercise 6: Application deployment using Key vault for secrets on AKS

[Use Azure Key Vault for secrets](https://dev.azure.com/cloudiq/K8S-Dev-Workshop/_git/k8s-dev-workshop?path=%2FREADME.md&version=GBmaster#path=%2Fassets%2Fdotnet-core%2Fapp-deployment-keyvault.md&version=GBmaster&createIfNew=true&_a=preview)

Exercise 7: Monitoring

[Monitoring](https://dev.azure.com/cloudiq/K8S-Dev-Workshop/_git/k8s-dev-workshop?path=%2FREADME.md&version=GBmaster#path=%2Fassets%2Fdotnet-core%2Faks-monitoring.md&version=GBmaster&createIfNew=true&_a=preview)

Exercise 8: Horizontal Pod Autoscaler