Azure DevOps Lab

MICROSOFT AZURE

Create AKS cluster and run simple python application there.

Final task

CONTENTS

1. AGENDA

Azure Kubernetes Service (AKS) is a managed Kubernetes service that lets you quickly deploy and manage clusters. In this task, we are asking you to create an AKS and deploy simple python application, let's call it "Greetings app", there.

The goal of the task is to create the AKS cluster with single user nodepool, connect to the cluster, run a multi-container Greetings application as a web front-end and a Redis instance as back-end in the cluster, and check if it works.

2. TASK

1) Create AKS cluster using AZ CLI:
   a) Create resource group under East US or Central US regions.
   b) Create Azure Kubernetes Service in that resource group.
   c) Add 2 node pools to the cluster.
2) Create Azure Container registry using AZ CLI.
3) Create container image:
   a) Build docker image with simple python application using attached source code and docker file.
   b) Push this image into your ACR (Azure Container registry).
4) Run application in AKS cluster:
   a) Connect to the cluster.
   b) Deploy Greetings application to the cluster using attached k8s manifests.
   c) Deploy Redis to the cluster.
   d) Expose the front-end service using "ClusterIP" and view it.

3. TASK DETAILS AND REQUIREMENTS

- Use the attached docker file, main.by, and requirements.txt to build the image.
- Blank deployment manifests files for k8s you find in attach.
- Update deployment for Greetings app to run 4 replicas of application.
- Greetings app can show your name in greeting message and uses "NAME" env variable to get your name. Create a config map containing your name and use it as an env variable in deployment for the Greetings app.
- Update deployment for the Greetings app with the path to your container registry and your container image.
- To make k8s able to pull an image from a private container image registry you should create Secret of kubernetes.io/dockerconfigjson type to authenticate with a container registry and update deployment to utilize this secret.
- Node pools should have by 1 node and have "frontendPool" and "backendPool" names and corresponding labels.

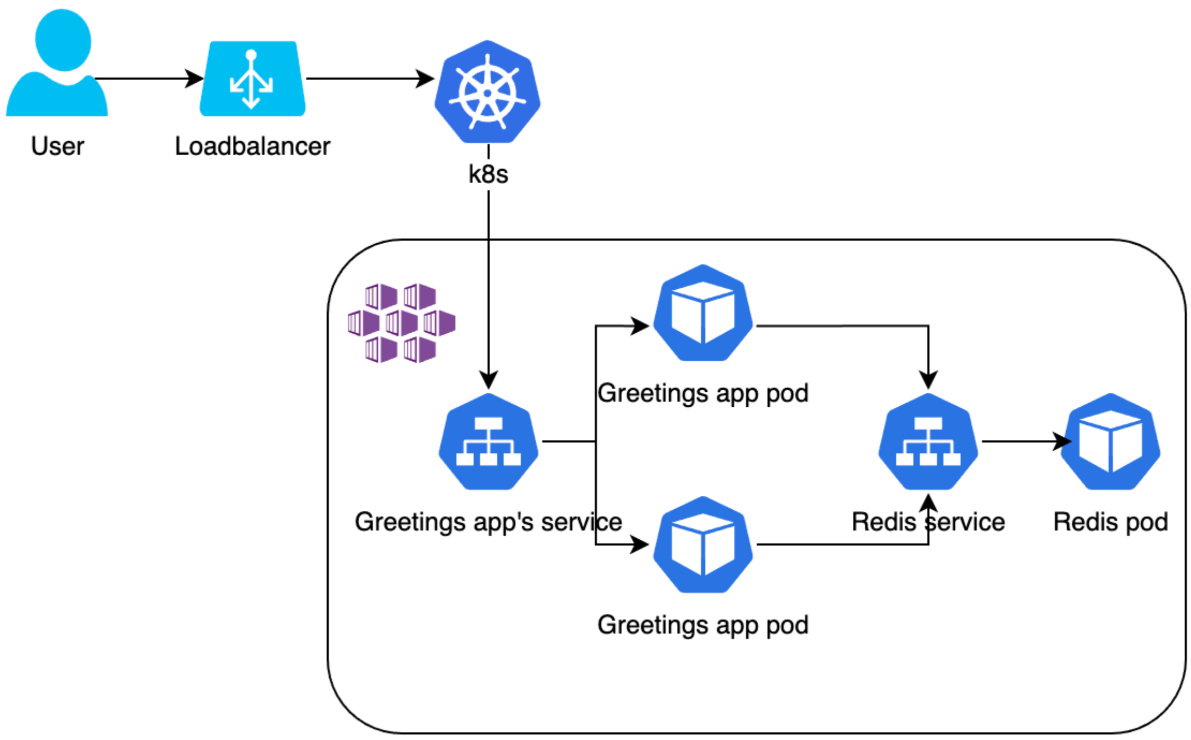
- Redis deployment should utilize “backendPool” node pool.
- Redis deployment should have configured requests:
  - cpu: 100m
  - memory: 100Mi
- K8s service CRD for Redis should has “redis” name to be correctly resolved by Greetings app.
- Greetings application should utilize “frontendPool” node pool.

Tips and tricks:

- Use nodeSelector to arrange deployments to the right node pool.
- To view the working application use you can use LoadBalancer’s external IP to view it
- Use docker documentation to tag and push image.

4. TASK RESULT

Greetings application up and running. Node pools utilized according to component designation. All requirements met.



5. USEFUL LINKS

https://docs.microsoft.com/en-us/azure/aks/kubernetes-walkthrough

https://docs.microsoft.com/en-us/cli/azure/aks?view=azure-cli-latest#az_aks_create

https://docs.microsoft.com/en-us/cli/azure/aks/nodepool?view=azure-cli-latest#az_aks_nodepool_add

https://docs.microsoft.com/en-us/azure/container-registry/container-registry-get-started-azure-cli

https://docs.docker.com/engine/reference/commandline/push/

https://kubernetes.io/docs/tasks/inject-data-application/define-environment-variable-container/

https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-private-registry/