Contents

[Running the code locally to validate its working. 1](#_Toc179142952)

[Step1 to set up application on Azure AKS. 1](#_Toc179142953)

[Updated the code to support health probe and expose metrics endpoint 1](#_Toc179142954)

[Docker image: 2](#_Toc179142955)

[Step2 to deploy python flask application with mongo db 2](#_Toc179142956)

[Step3 to set up Prometheus and Grafana for visualization 2](#_Toc179142957)

[Steps to expose metrics endpoint from the python application and configure to scrape metrics of AKS. 3](#_Toc179142958)

[Prometheus: Validation: 5](#_Toc179142959)

[Grafana Dashboard: 6](#_Toc179142960)

[Step 4: Integrate with ELK Stack 6](#_Toc179142961)

[Validation Screen shots: 7](#_Toc179142962)

[Logs on kibana: 7](#_Toc179142963)

Implementation

## Running the code locally to validate its working.

* 1. Using pycharm and mongodb

Tested locally to verify the working.

## Step1 to set up application on Azure AKS.

1. Create Kubernetes cluster on Azure with RBAC.

Note: AKS has inbuild extension to enable Prometheus and Grafana. However, i have decided to go with approach of spinning them as pods on the AKS.

1. Create ACR – Azure container registry to store the docker images.
2. Provided AKS Managed identity access “Image Pull access” over the ACR to be able to fetch the images during the deployments.
3. Create Docker file for the python application and pushed into the container registry.

### Updated the code to support health probe and expose metrics endpoint

1. To enable health probe.
2. To publish metric end points. Prometheus uses /metrics to pull the application logs.

1. Create Kubernetes cluster on Azure with RBAC.

Note: AKS has inbuild extension to enable Prometheus and Grafana. However, i have decided to go with approach of spinning them as pods on the AKS.

1. Create ACR – Azure container registry to store the docker images.
2. Provided AKS Managed identity access “Image Pull access” over the ACR to be able to fetch the images during the deployments.
3. Create Docker file for the python application and pushed into the container registry.

### Docker image:

1. “Docker build -t flask-mongo-app:v3 .”
2. “Docker tag flask-mongo-app:v3 acrcommonwesteu.azurecr.io/flask-mongo-app:v3”
3. “Docker push acrcommonwesteu.azurecr.io/flask-mongo-app:v3

A computer screen with white text and red text

Description automatically generated

## Step2 to deploy python flask application with mongo db

1. Create load balancer service for mongodb and python application.
2. Create manifest file to deploy the python application with mongodb as deployment on the Kubernetes.
3. Enable horizontal Pod Scaling (HPA).

## Step3 to set up Prometheus and Grafana for visualization

1. Create new name space called Prometheus
2. Create manifest file for Prometheus and expose it as service. (Note: This can also be set up with help of helm charts)
3. Create manifest file for Grafana and expose it as service.
4. Configure Prometheus connection in the targets of Grafana.

Prometheus url : http://prometheus-service.prometheus.svc.cluster.local:80

A screenshot of a computer

Description automatically generated

Note: You will be able to view Prometheus metrics alone. Application logs and AKS metrics will be missing.

### Steps to expose metrics endpoint from the python application and configure to scrape metrics of AKS.

1. Modify the python code to expose metrics.
2. Use helm package to scrape the AKS metrics.

[helm-charts/charts/kube-state-metrics/README.md at main · prometheus-community/helm-charts (github.com)](https://github.com/prometheus-community/helm-charts/blob/main/charts/kube-state-metrics/README.md)

* helm repo add prometheus-community <https://prometheus-community.github.io/helm-charts>
* helm repo update
* helm install prom-metrics prometheus-community/kube-state-metrics

1. Update the config map of Prometheus
2. Kubectl edit configmap -m prometheus-server-conf -n prometheus

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

### Prometheus: Validation:

A screen shot of a graph

Description automatically generatedA screenshot of a computer

Description automatically generated

### Grafana Dashboard:

A screenshot of a computer

Description automatically generated

## Step 4: Integrate with ELK Stack

Successfully integrated with ELK , with help of Filebeat , Elastic search , logstash and kibana.

Following verified versions of helms published by at artifact hub are used for this assessment

https://artifacthub.io/packages/helm/elastic/elasticsearch/7.15.0

https://artifacthub.io/packages/helm/elastic/filebeat/7.15.0

https://artifacthub.io/packages/helm/elastic/logstash - version 8.5.1

[kibana 7.15.0 · elastic/elastic (artifacthub.io)](https://artifacthub.io/packages/helm/elastic/kibana/7.15.0)

### Validation Screen shots:

Following screen shot showing list of successful running services.

A computer screen shot of text

Description automatically generated

### Logs on kibana:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated