

**Lab Manual- AKS Troubleshooting with Kubernetes Dashboard and Health Check**

**Prepared for**:

**Date:** 18th Dec 2023

**Prepared by:**

Document Name: Lab Manual **Document Number** AZLabn916

**Contributor:**

Contents

[1. Objective 3](#_Toc157764830)

[2. Install and Set Up kubectl on Windows 3](#_Toc157764831)

[3. Create 1 Node Azure Kubernetes Cluster 3](#_Toc157764832)

[4. Connect the Kubernetes Cluster from azure PowerShell 4](#_Toc157764833)

[5. Create a Deploy Kubernest Dashboard from helm chart 6](#_Toc157764834)

[6. Access the Kubernetes Dashboard ( Local PowerShell) 7](#_Toc157764835)

[7. Deploy Sample NGNIX Pod 11](#_Toc157764836)

[8. Check NGNIX Pod in Kubernetes Dashboard 13](#_Toc157764837)

[9. Health Checks 14](#_Toc157764838)

[1. Liveness Probe 14](#_Toc157764839)

[2. Readiness Probe 14](#_Toc157764840)

[10. Delete the resources 15](#_Toc157764841)

# Objective

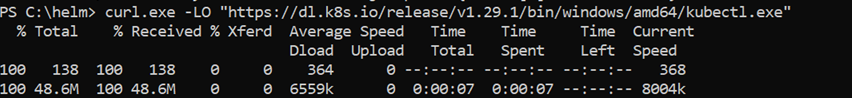
# Install and Set Up kubectl on Windows

[ Note : I already have Folder Name Helm and Hel.exe downloaded in this folder]

* Open PowerShell with Admin Mode

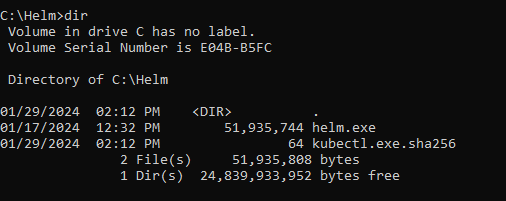
Download the latest 1.29 patch release: [kubectl 1.29.1](https://dl.k8s.io/release/v1.29.1/bin/windows/amd64/kubectl.exe). FROM <https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/>

curl.exe -LO "https://dl.k8s.io/release/v1.29.1/bin/windows/amd64/kubectl.exe"



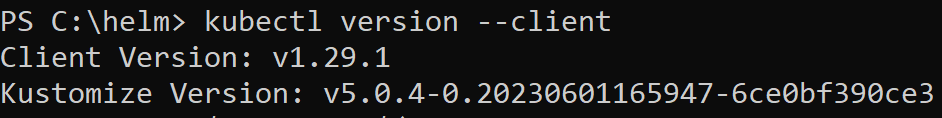
* Check the download

DIR



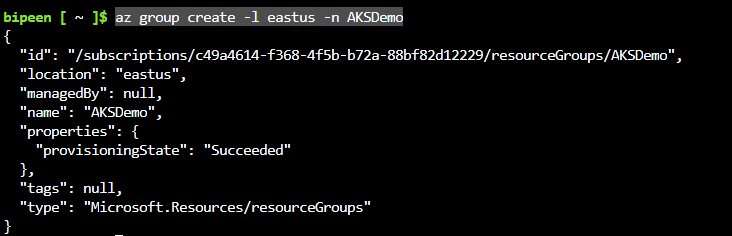
* Test to ensure the version of kubectl is the same as downloaded:

kubectl version --client



# Create 1 Node Azure Kubernetes Cluster

az group create -l eastus -n AKSDemo



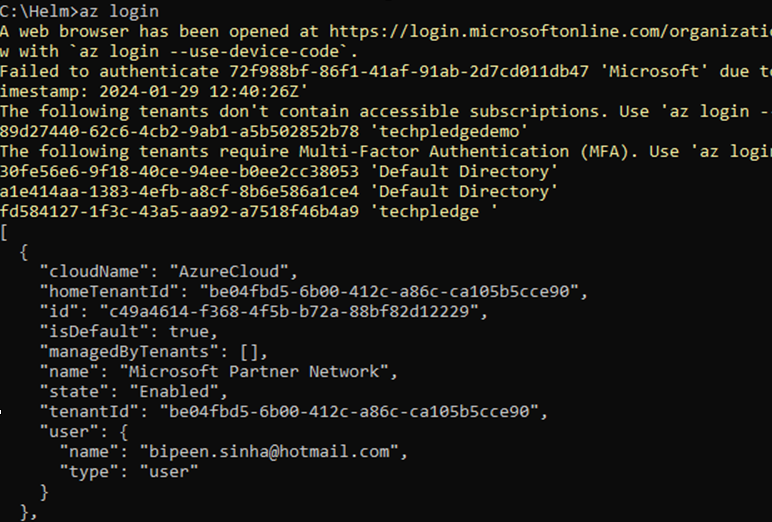
To create an AKS cluster, use the [az aks create](https://learn.microsoft.com/en-us/cli/azure/aks" \l "az-aks-create) command. The following example creates a cluster named BipeenClus01 with one node and generate SSH-key. It will automatically use 3 Nodes when you don’t specify node count

az aks create --resource-group AKSDemo --name BipeenClus01 --node-count 2 --generate-ssh-keys

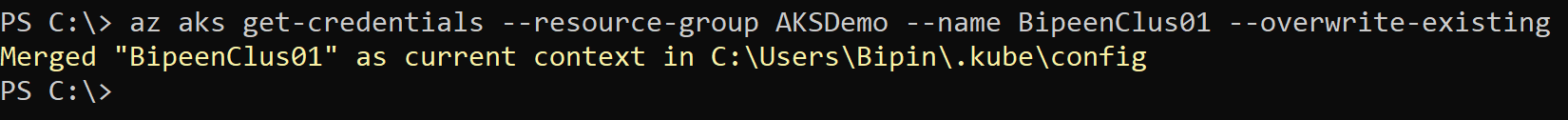


# Connect the Kubernetes Cluster from azure PowerShell

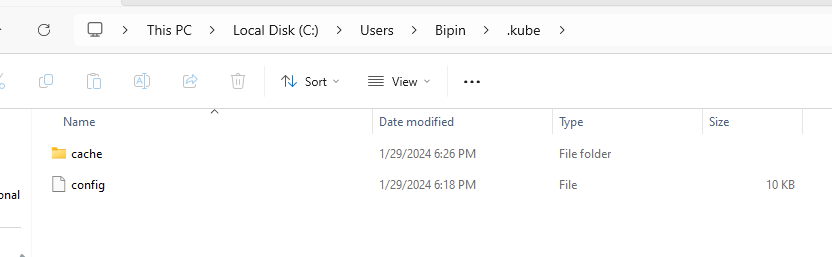
Az login



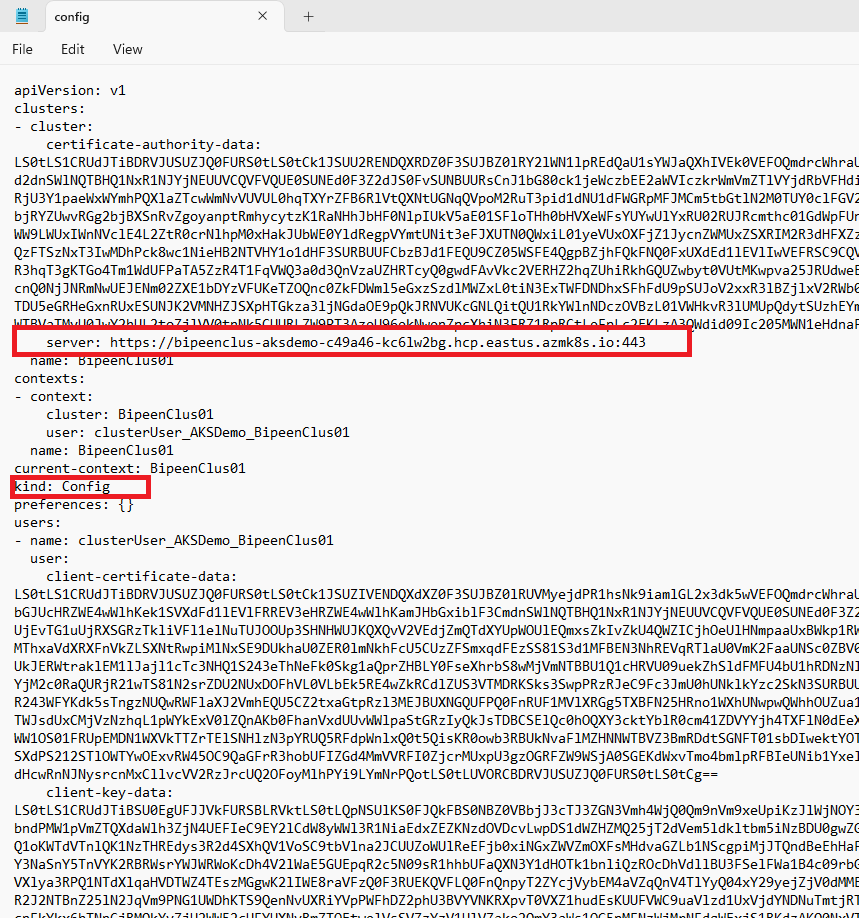
az aks get-credentials --resource-group AKSDemo --name BipeenClus01 --overwrite-existing



You can see the Config as YAML format



When you open it with notepad



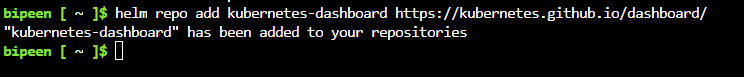
# Create a Deploy Kubernest Dashboard from helm chart

<https://artifacthub.io/packages/helm/k8s-dashboard/kubernetes-dashboard>

kubectl create ns aks-dashboard



helm repo add kubernetes-dashboard https://kubernetes.github.io/dashboard/



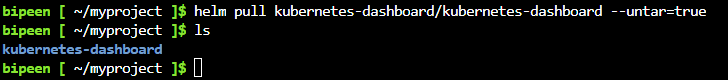
mkdir myproject

cd myproject/

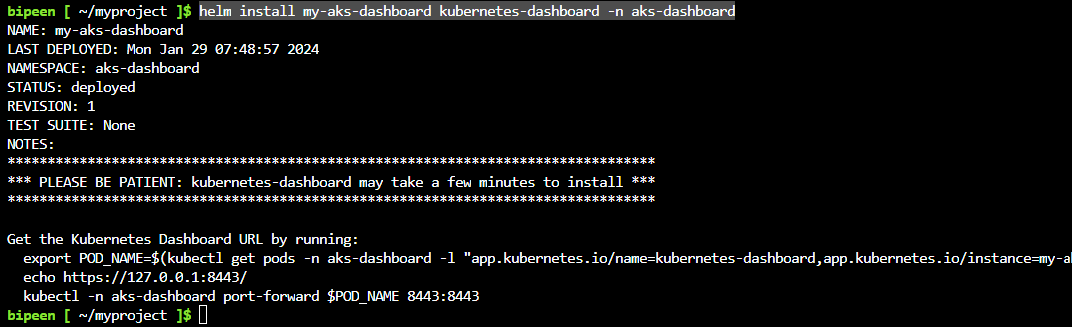


helm pull kubernetes-dashboard/kubernetes-dashboard --untar=true

ls



helm install my-aks-dashboard kubernetes-dashboard -n aks-dashboard



# Access the Kubernetes Dashboard ( Local PowerShell)

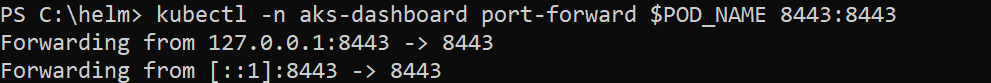
$POD\_NAME=$(kubectl get pods -n aks-dashboard -l "app.kubernetes.io/name=kubernetes-dashboard,app.kubernetes.io/instance=my-aks-dashboard" -o jsonpath="{.items[0].metadata.name}")



echo $POD\_NAME

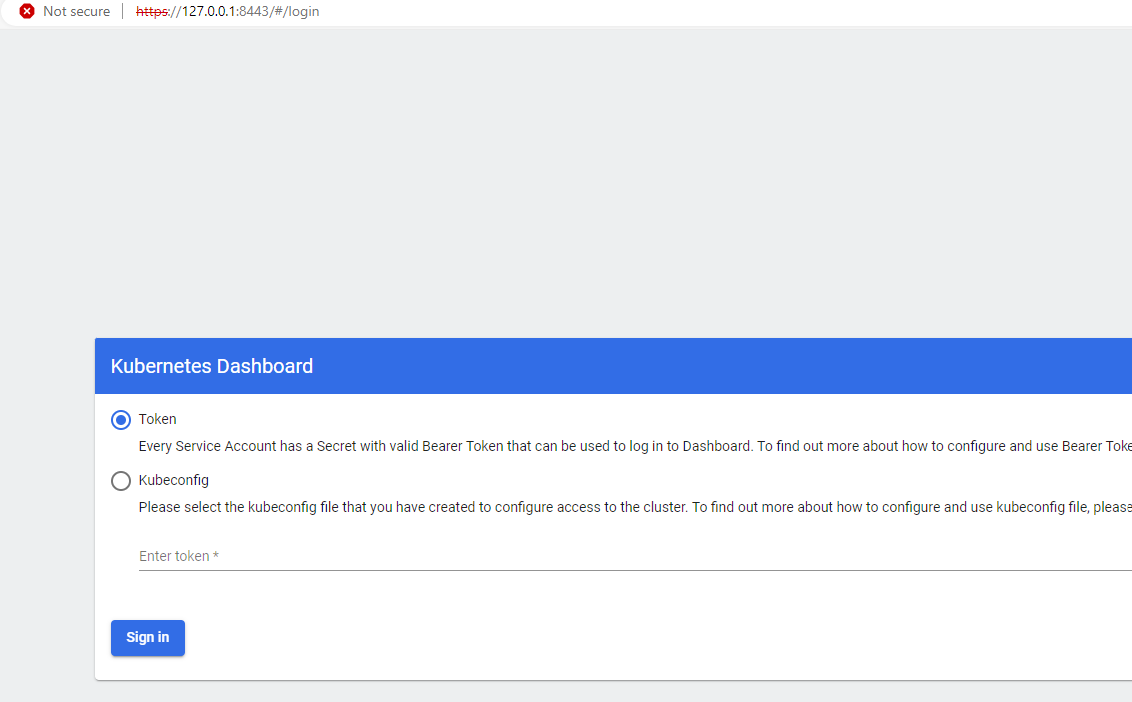


kubectl -n aks-dashboard port-forward $POD\_NAME 8443:8443

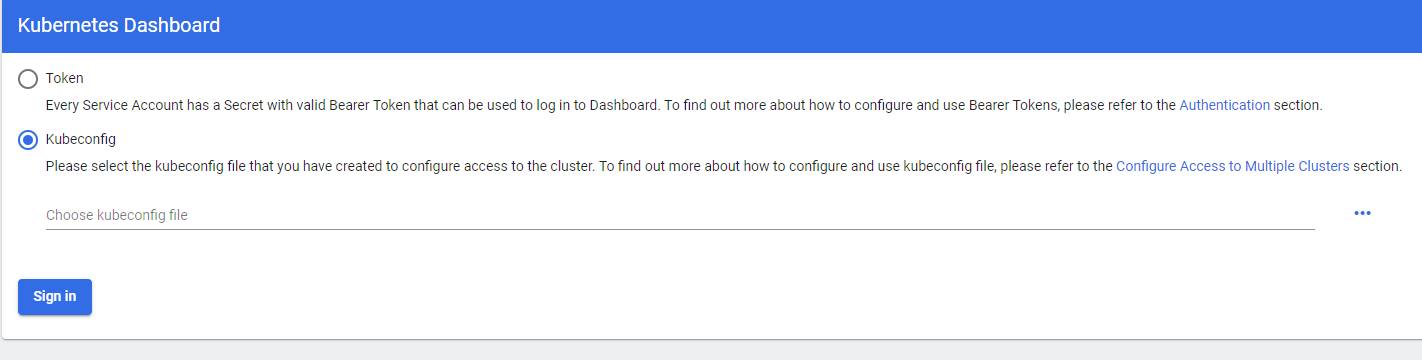


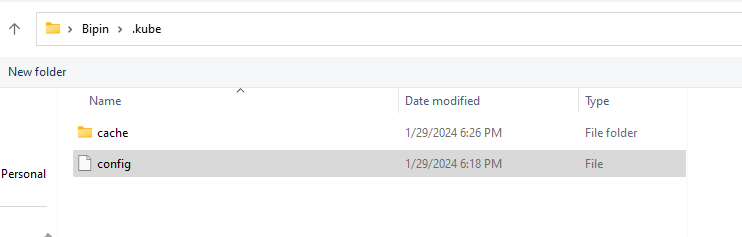
Open browser and Type

<https://127.0.0.1:8443/>

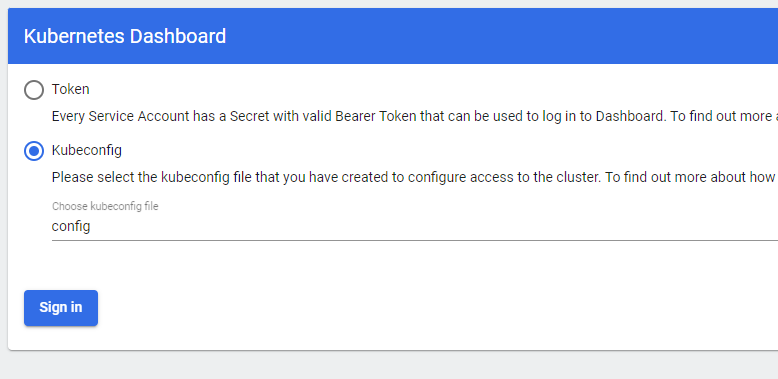


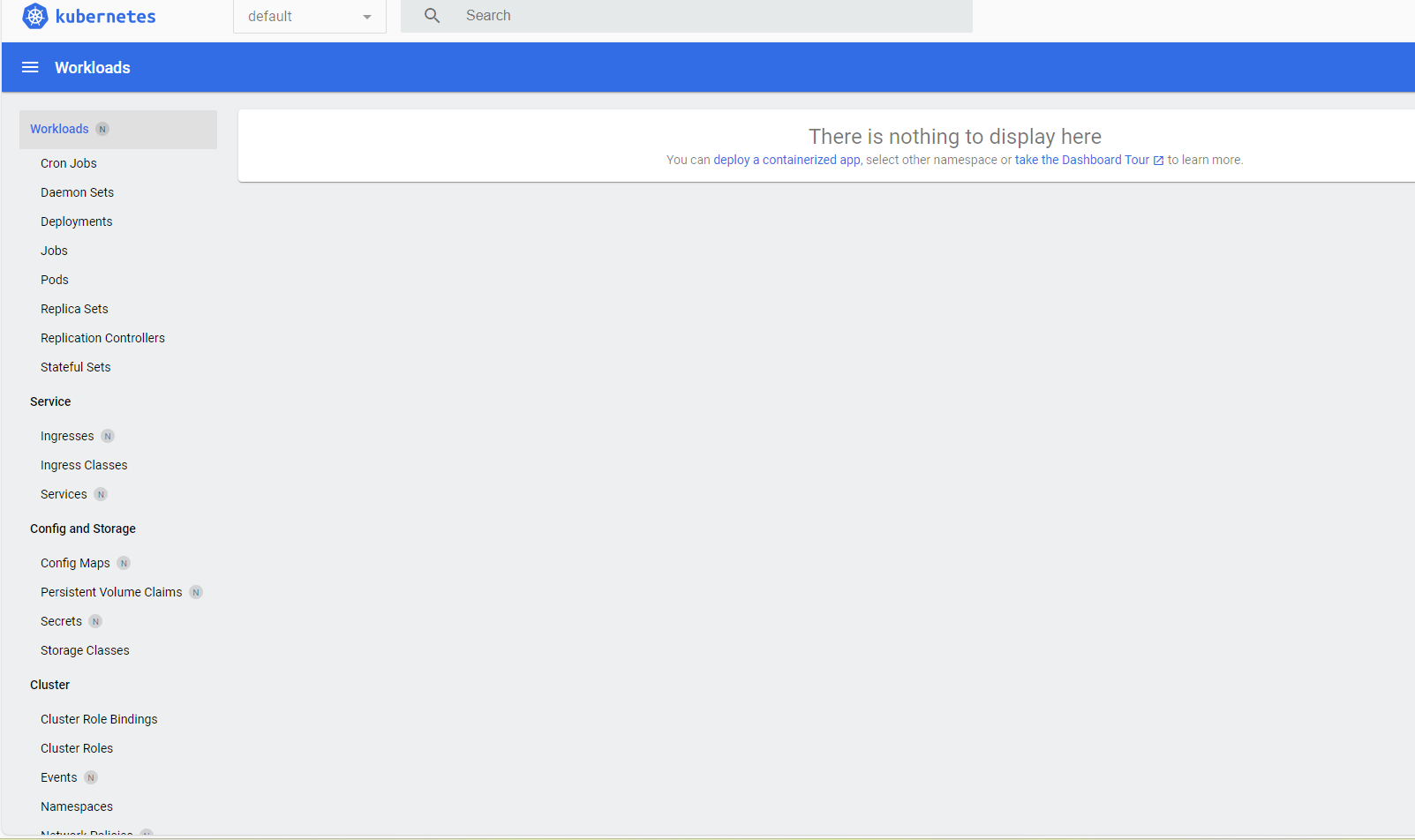
Select kubeconfig and click …



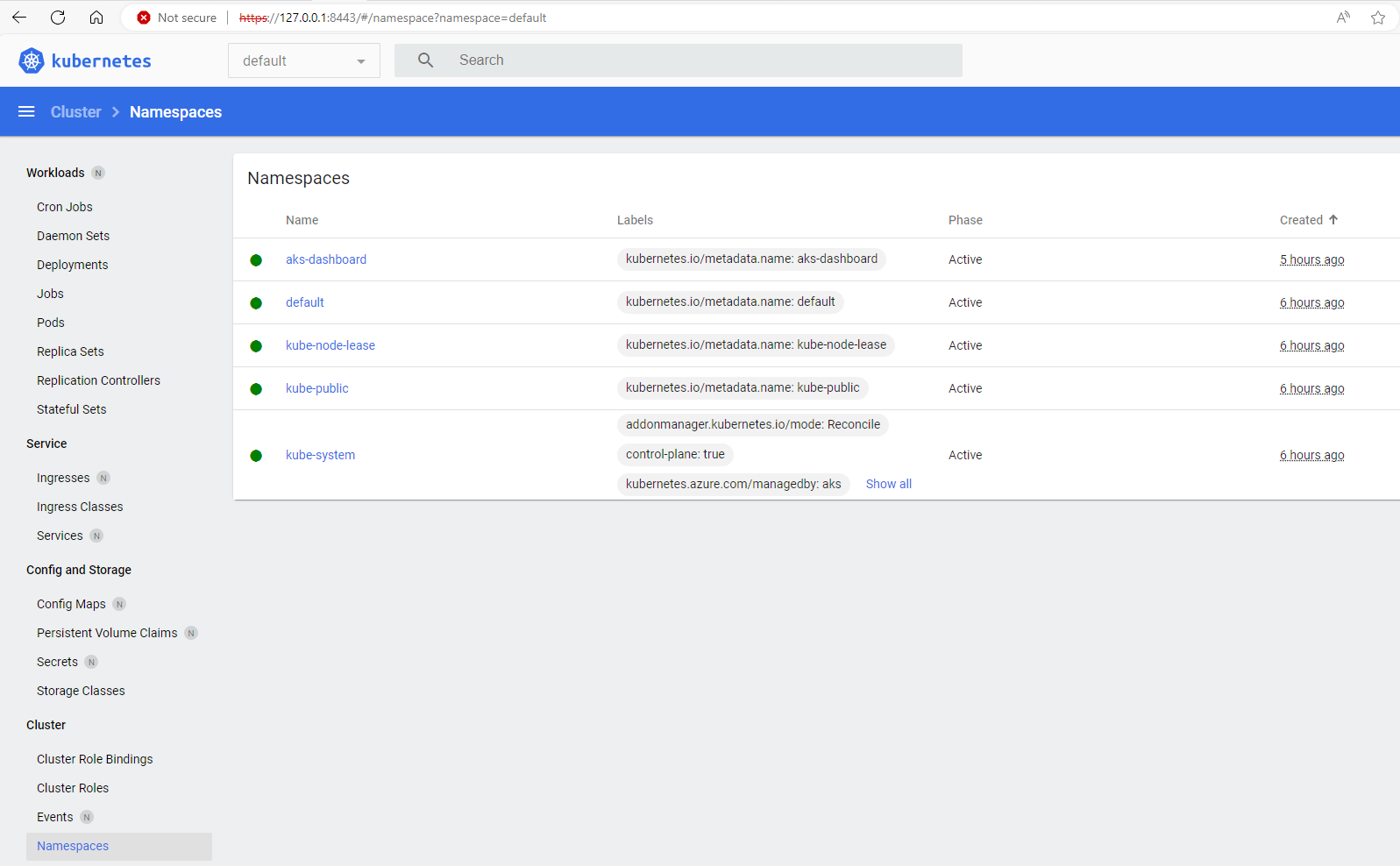


Click sign-in

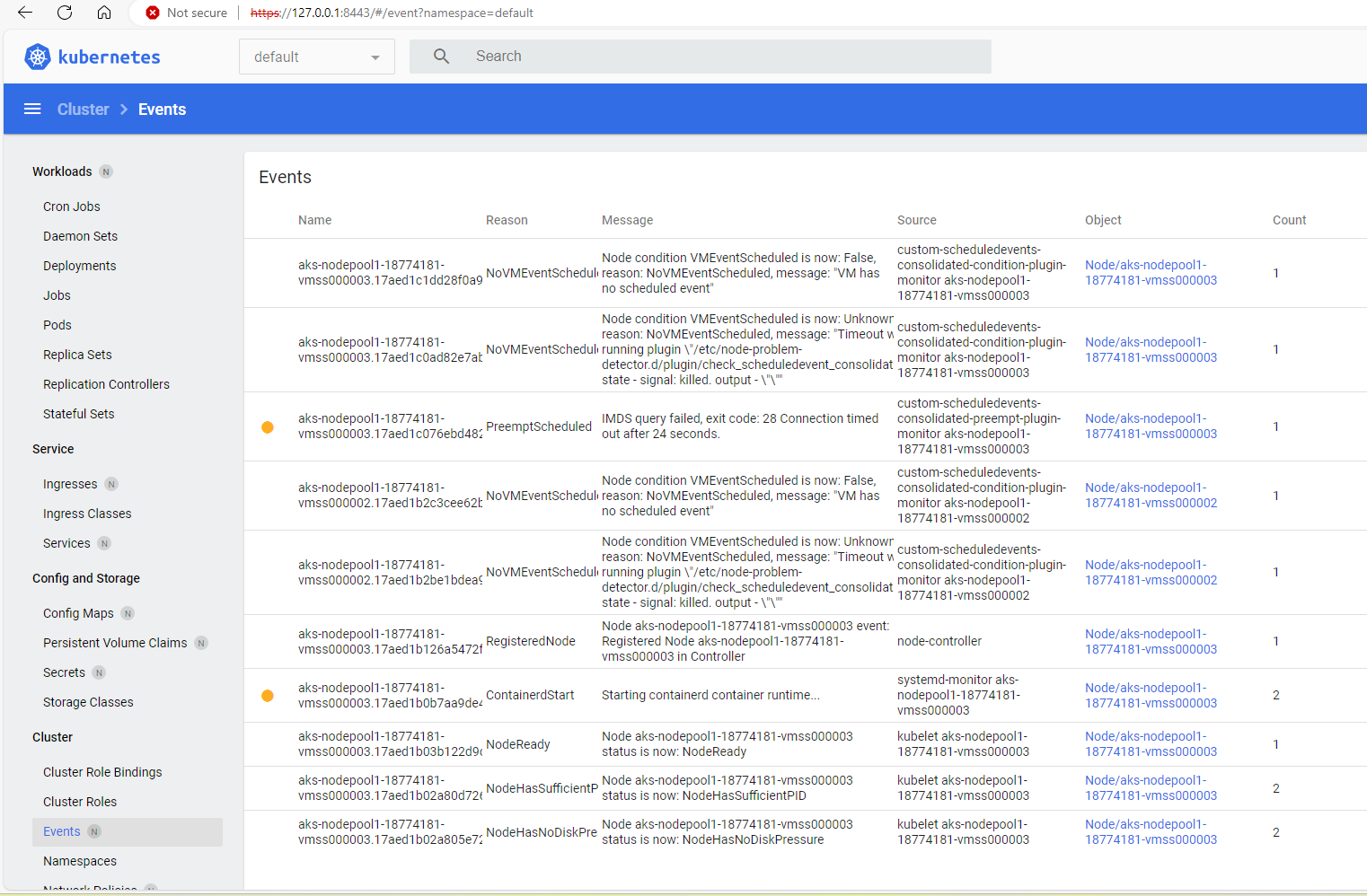




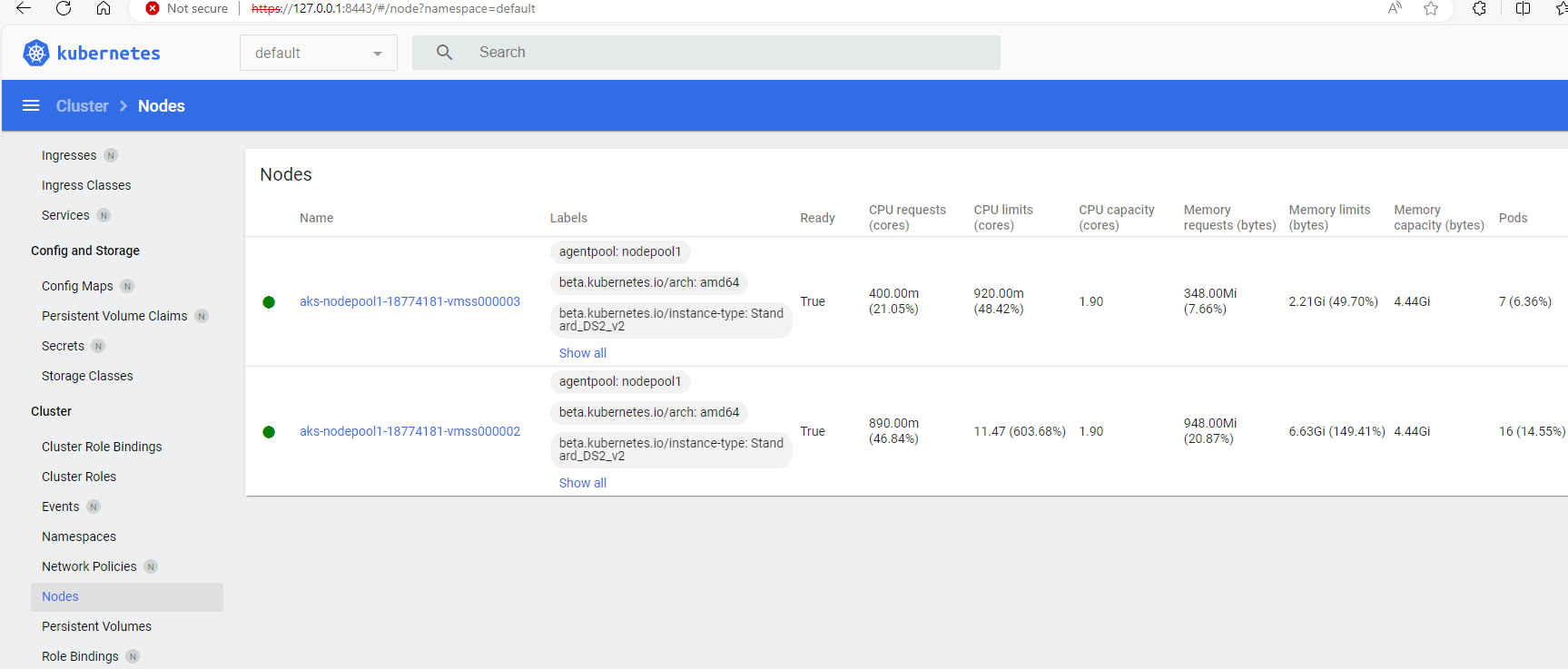
Click Namespace



Click event

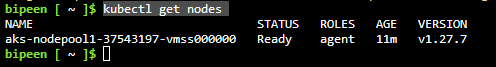


Click Nodes



# Deploy Sample NGNIX Pod

kubectl get nodes



We will first create the deployment using the following command.

kubectl create deployment nginx-project --image=nginx

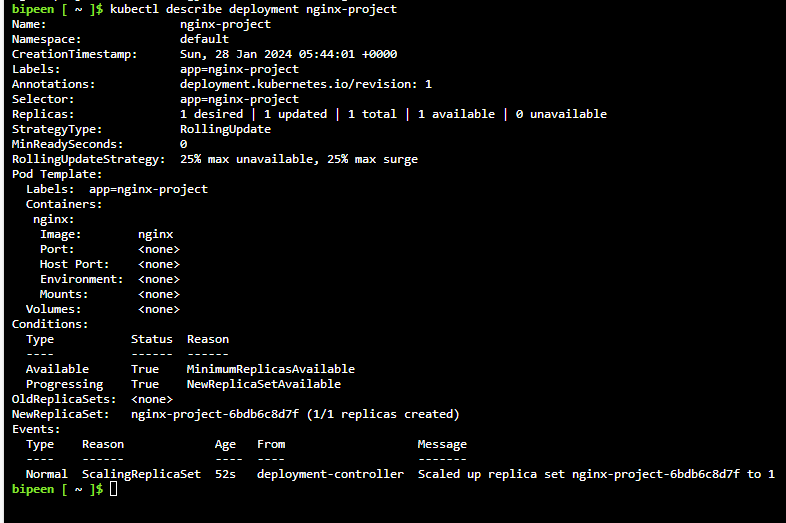


kubectl get po



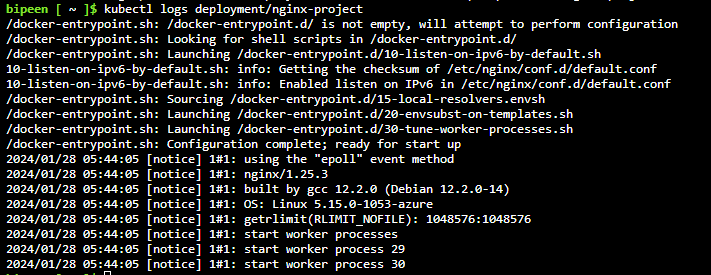
Use the following command to get the details of the deployment.

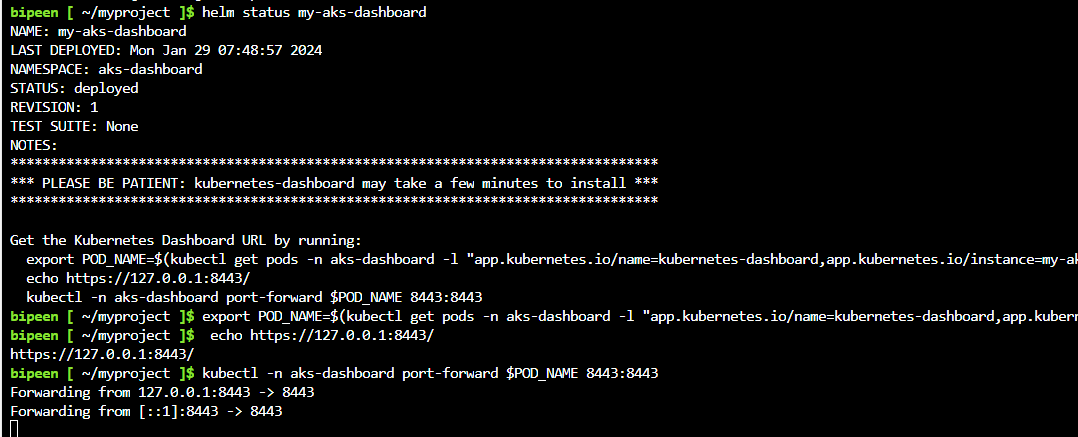
kubectl describe deployment nginx-project



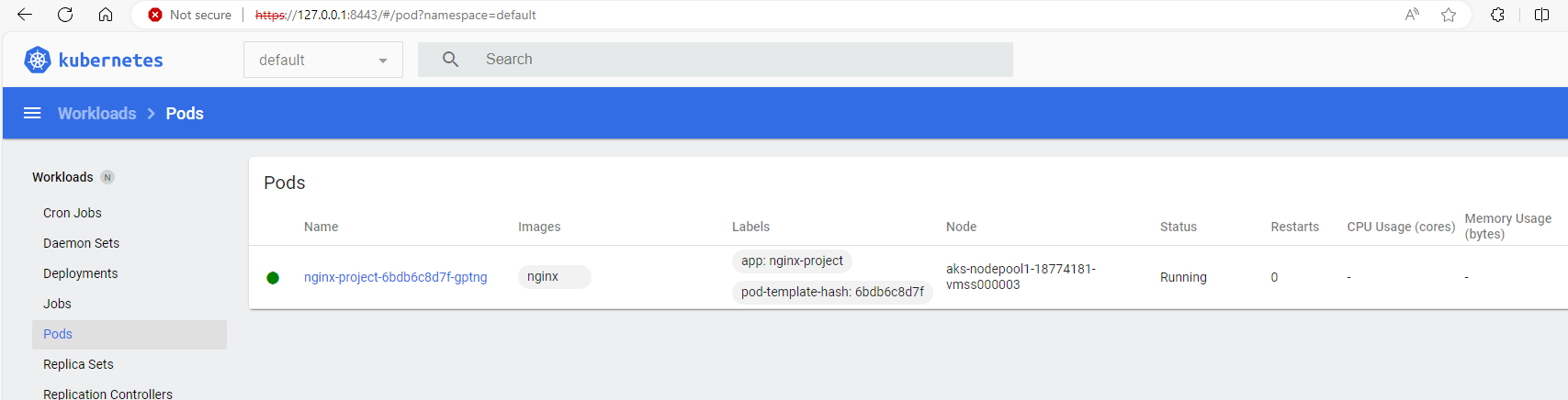
To get the logs, we can run the following command.

kubectl logs deployment/nginx-project

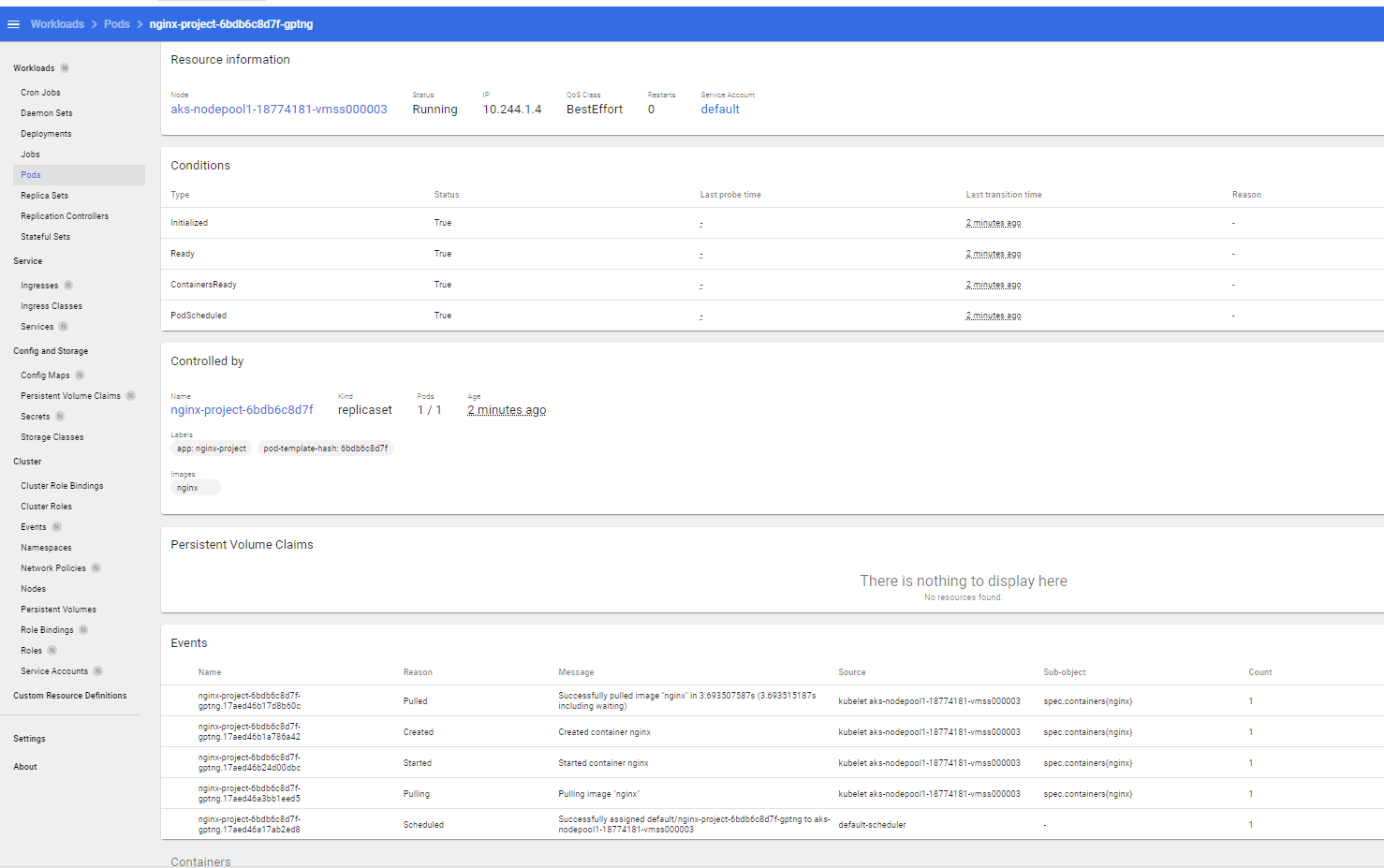




# Check NGNIX Pod in Kubernetes Dashboard



Click the pod name to see the details including **events**



# Health Checks

We want to check if our pod services are healthy. Because processes can get into weird states, we want to make sure all of our containers in a pod are actually behaving normally. You get to define 'normally' in your pod manifest, and then that check will run against every container in your pod:

## Liveness Probe

"Is this container alive?"

## Readiness Probe

"Is this container ready to receive traffic?" Good for containers that have some bootstrapping/initialization to do before they're ready to receive traffic or respond to liveness checks.

apiVersion: v1

kind: Pod

metadata:

  name: kuard

spec:

  containers:

  - name: kuard

    image: gcr.io/kuar-demo/kuard-amd64:1

    ports:

    - containerPort: 8080

    livenessProbe:

      httpGet:

        path: /healthy

        port: 8080

      initialDelaySeconds: 5

      timeoutSeconds: 1

      periodSeconds: 10

      failureThreshold: 3

    readinessProbe:

      httpGet:

        path: /ready

        port: 8080

    protocol: TCP

kubectl apply -f kuard-pod-health.yaml

kubectl port-forward kuard 8080:8080

Point your browser to [http://localhost:8080](http://localhost:8080/). Click the “Liveness Probe” tab. You should see a table that lists all of the probes that this instance of kuard has received. If you click the “fail” link on that page, kuard will start to fail health checks.

# Delete the resources

az group delete --name AKSDemo --yes --no-wait