

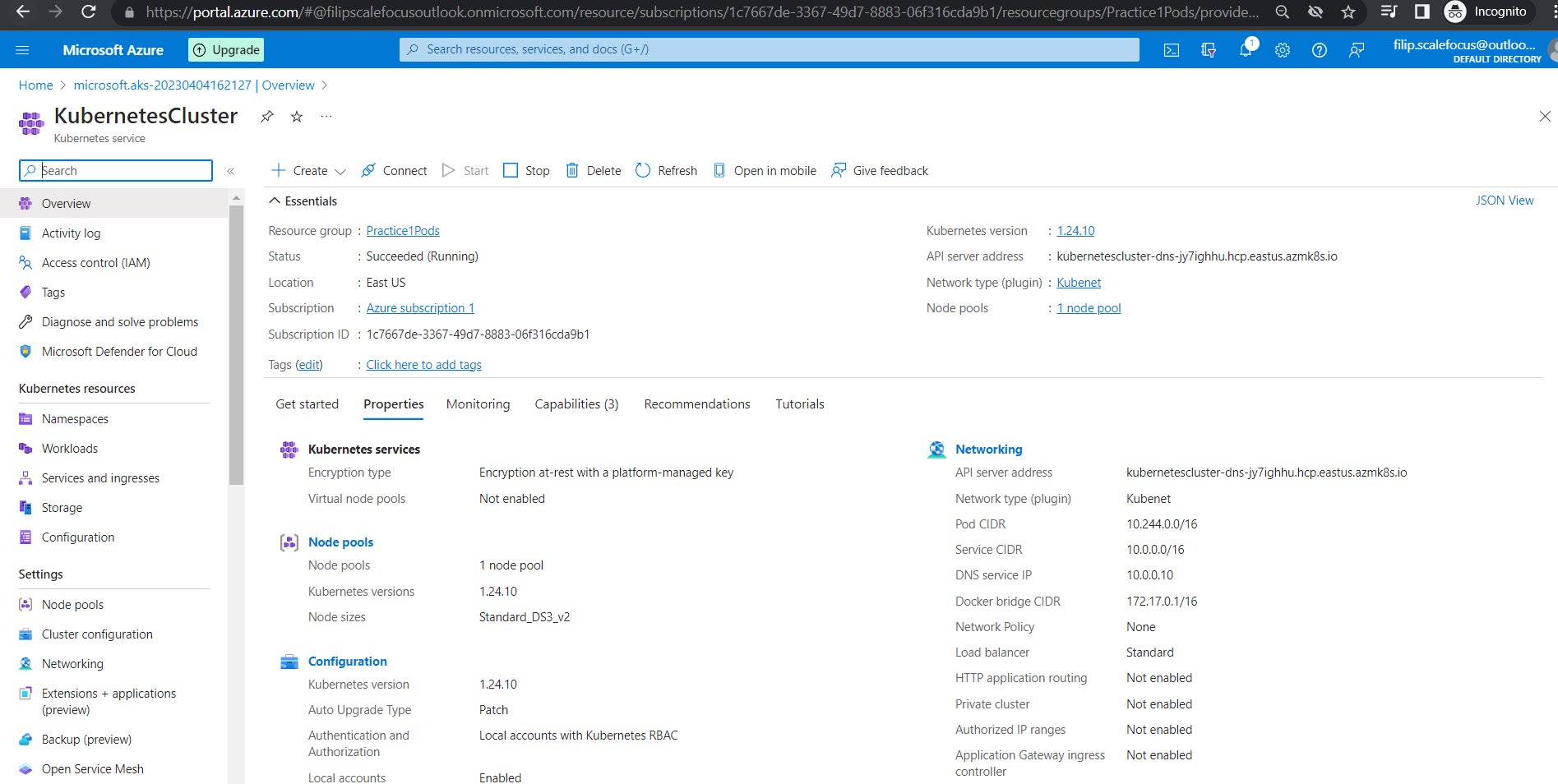
**Exercise: Pods**

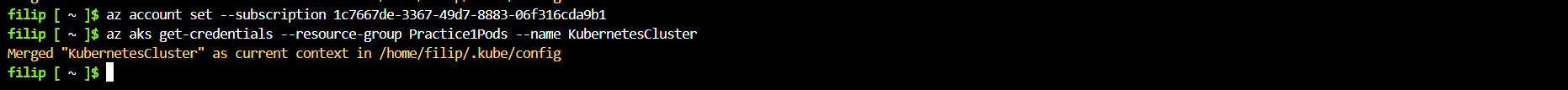
Pods are the smallest, most basic deployable objects in Kubernetes. A Pod represents a single instance of a running process in your cluster. Pods contain one or more containers, such as Docker containers. Although you want deploy pods directly (static pods), knowledge for defining pods manifest files will be used for defining more complex Kubernetes resources like Controllers.

**Practice1: Simple pods operations**

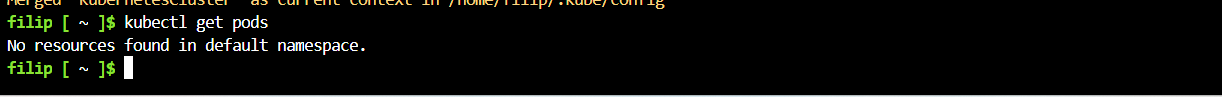
Note: Try not to do a copy/paste on commands requests unless you are instructed to do so. Copy/paste will not help you to learn Kubernetes!

1. Login to Azure and connect to your AKS cluster.

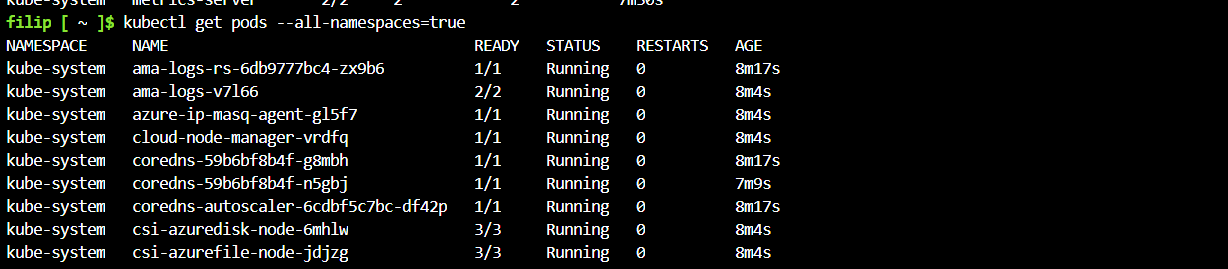




1. Check how many pods run under the default namespace. Run **kubectl get pods**.



1. You should not see any pod under the default namespace. Now check all namespaces. Run **kubectl get pods –all-namespace**.

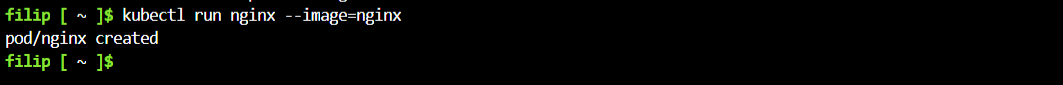


1. How many pods do you see? Who deployed these pods? Why are they deployed?

Graphical user interface, text, application

Description automatically generated

1. Now deploy you first pod using the imperative approach. Run **kubectl run nginx --image=nginx**.



1. Validate if the pods has been created. What is the status of your pod?

Graphical user interface, text

Description automatically generated with medium confidence

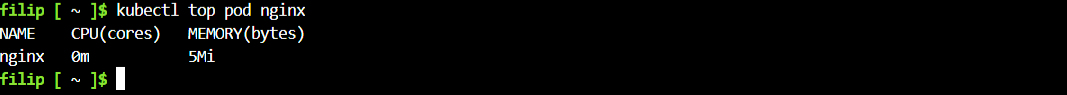
The pod was created and is Running.

1. Check the logs coming out of your pod. Run **kubectl logs nginx**.

Text

Description automatically generated

1. Run following command to check current resource consumption of your pod: **kubectl top pod nginx**.



1. Check on which Node your pods has been scheduled. Run **kubectl get pods –o wide**.

Text

Description automatically generated

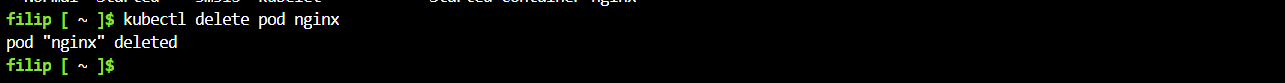
It runs on Node aks-agentpool-28366991-vmss000000/10.224.0.4.

1. Try to find the same information but this time running **kubectl describe pod nginx.**

Text

Description automatically generated

1. Delete your pod using **kubectl delete pod nginx.**



1. Let’s find the image used on one of the coredns pods under the kube-system namespace.

A screenshot of a computer

Description automatically generated with medium confidence

1. Once again list all pods under all namespaces.

Graphical user interface

Description automatically generated

1. Note one of the coredns pods. Now run **kubectl describe pod <coredns-name> -n kube-system**. Replace the <coredns-name> place holder with noted name.

Text

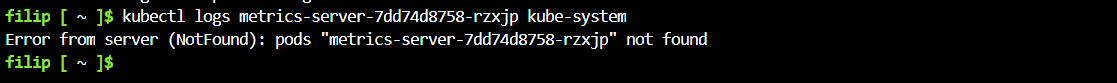
Description automatically generated

1. Inspect the output and locate the image information.

Text

Description automatically generated

1. Now let us check the logs of the metrics-server pod. Run the same command as in step 7 but don’t forget to add the namespace in which this pod is created.



**Practice2: Working with pod manifest files**

Note: Try not to do a copy/paste on commands requests unless you are instructed to do so. Copy/paste will not help you to learn Kubernetes!

1. Now it is time to deploy pod using manifest file (declarative approach). Copy the following code block on your local computer in a file called redis.yaml:

**apiVersion: v11**



**kind: pod**



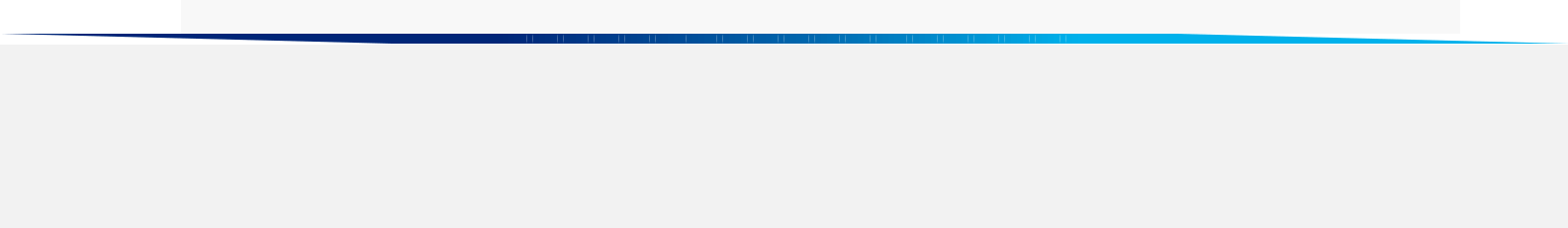
**metadata:**



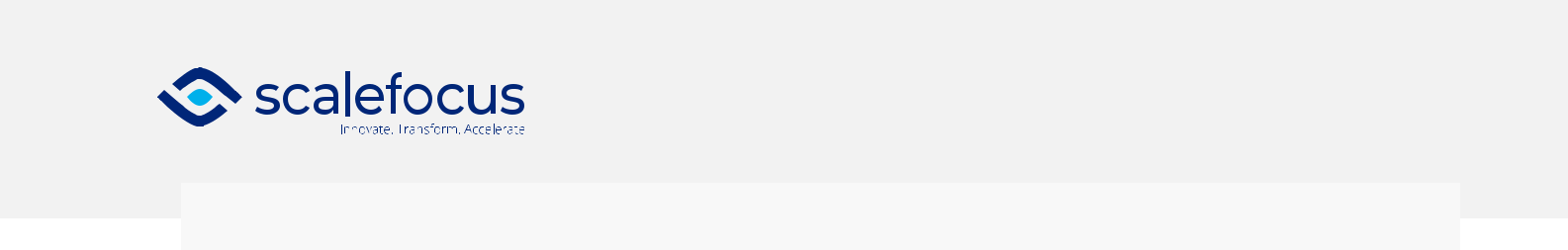
**name: static-web**



**labels:**



69 Bulgaria Blvd., Infinity Tower B, fl. 8, Triaditza district, 1404 Sofia, Bulgaria | +359 2424 6484 | sales@scalefocus.com | www.scalefocus.com

**role: myrole**

**specs:**

**containers:**



* **name: redis image: redis123**

Text

Description automatically generated

* 1. Try to deploy the pod defined in redis.yaml. Run **kubectl create –f redis.yaml**.

Shape, rectangle

Description automatically generated

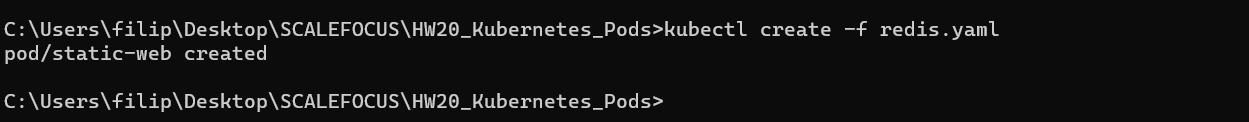
* 1. You will receive errors on your screen. Your next task will be to correct the syntax of the code you just copied. You can use the online Kubernetes documentation or you can search the internet in general.

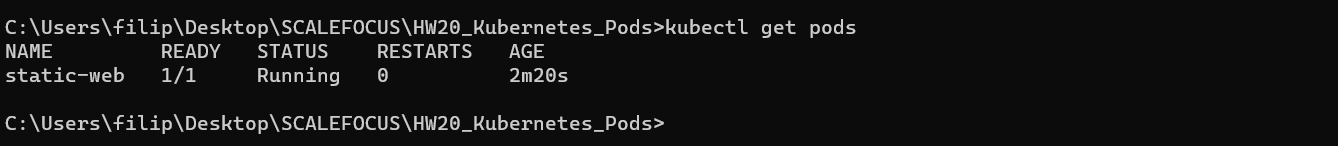
The version was changed to v1. The kind was changed to pod. The image version was added – latest.

Text

Description automatically generated

* 1. When you solve all the syntax errors your pod should be deployed but is it running? What is the status of your pod?





* 1. Check the events associated with this pod. Run the **kubectl describe pod static-web** command. What are the events showing? Why your pod is not running?

Text

Description automatically generated

Text

Description automatically generated

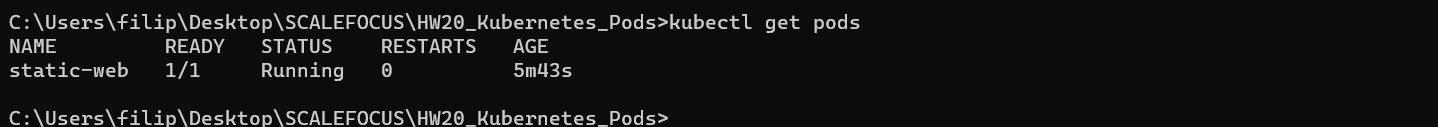
* 1. Find the correct image (check the Docker hub page) and correct it in the manifest.

My image is correct. I used the latest one. And my pod is running.

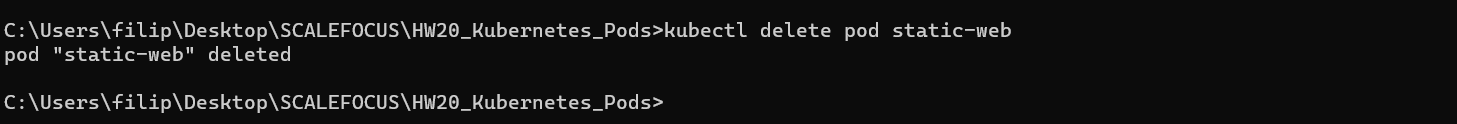
* 1. Locate the image information and put the correct image name. Redeploy the pod (fist run **kubectl delete pod static-web** to delete the pod, then run kubectl create once again).

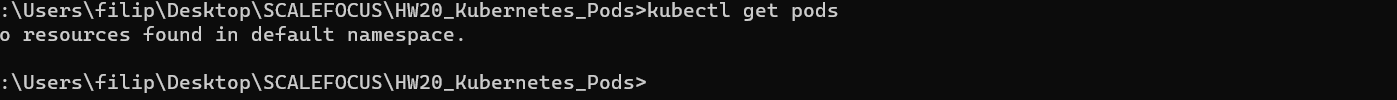
I will not delete the pod that I have created.

* 1. Check the status of your pod. It should be running now.



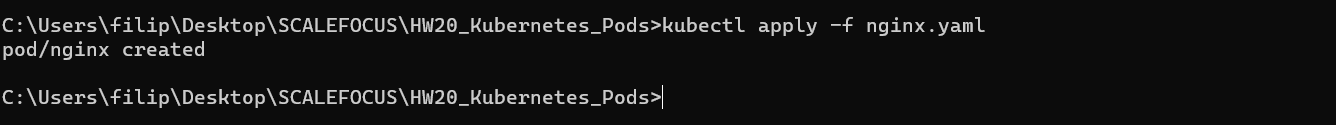
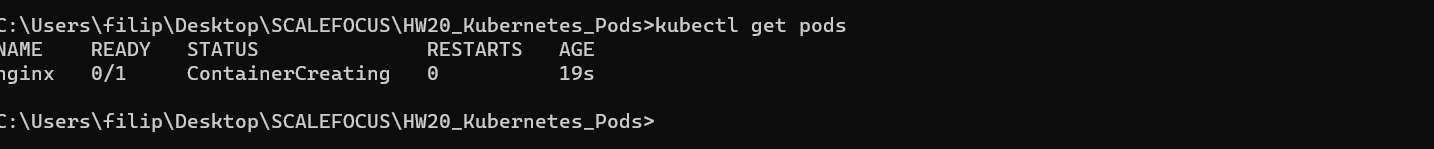
* 1. Now you can delete the pod. Try to delete it using the **kubectl delete –f redis.yaml**.





* 1. Your next task is to create and test nginx pod definition. Your definition should use the nginx official image, should use label named app with value frontend and should publish port 80. Make sure you complete this task because we will use this template in our next Labs. Your nginx pod should be running without any issues.

Text

Description automatically generated  

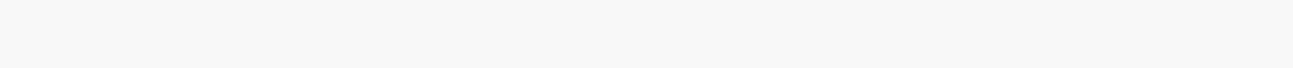
* 1. Final task of this practice will be to define pod definition with following details:
     + Image=memcached
     + Port= 11211
     + Label app=web
     + CPU request=0.35 cores
     + RAM request=0.15 GB
     + CPU limit=0.5 cores
     + Ram limit=0.25 GB
     + Restart policy=Never

Text

Description automatically generated

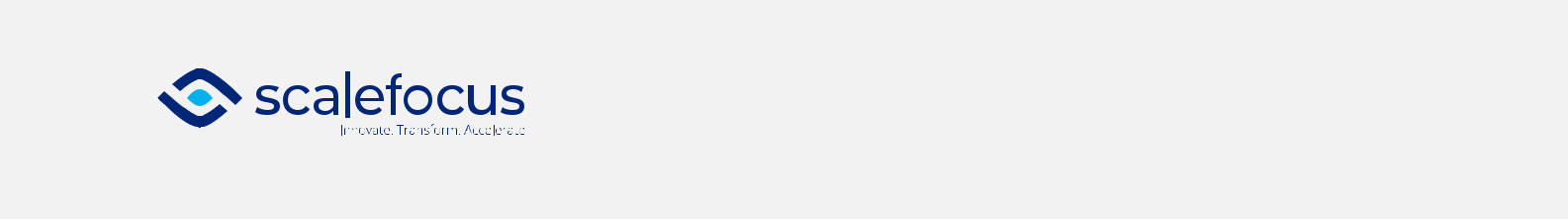
* 1. Don’t forget to try your pod definition.

Text

Description automatically generated

**Practice3: Multi-container pods**

Note: Try not to do a copy/paste on commands requests unless you are instructed to do so. Copy/paste will not help you to learn Kubernetes!

1. Once finished you can try to create multi-container pod definition. Your multi-container pod should use redis and nginx containers with port 6379 and 80 published respectively. Label name should be app with value web.

Text

Description automatically generated

1. Note that in reality there is no sense to put the redis and nginx under the same pod but it can be done for the purpose of learning.
2. Deploy your multi-container pod. It should have running status. What is written under Ready column when you kubectl get the pods? Why your pod displays different values for ready?

Text

Description automatically generated

Text

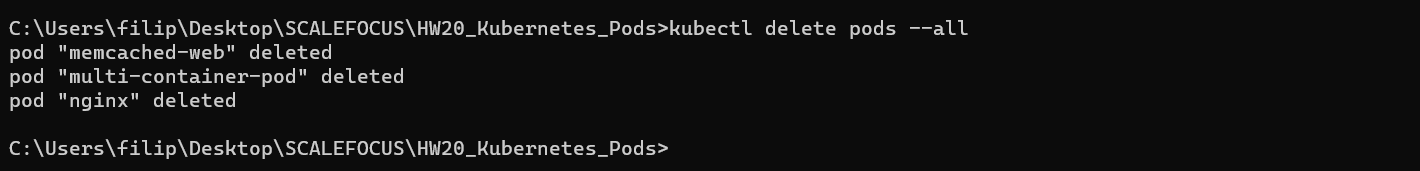
Description automatically generated2/2

1. Kubectl describe you new pod, and locate the containers section. How many containers are listed?

Text

Description automatically generated

1. Delete all the pods under the default namespace.



1. Don’t delete any of the manifest files you have created so far.

**Practice4: Probes**

Note: Try not to do a copy/paste on commands requests unless you are instructed to do so. Copy/paste will not help you to learn Kubernetes!

1. First we will create and test liveness probe with exec test. Create a file named probes\_exec.yaml with following content:

**apiVersion**: v1



**kind**: Pod



**metadata**:



**labels**:



**test**: liveness



**name**: liveness-exec



**spec**:



**containers**:



* **name**: liveness

**image**: k8s.gcr.io/busybox **args**:

* + /bin/sh
  + -c
  + touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 600 **livenessProbe**:

**exec**: **command**:

* + - cat
    - /tmp/healthy



**initialDelaySeconds**: 5



**periodSeconds**: 5



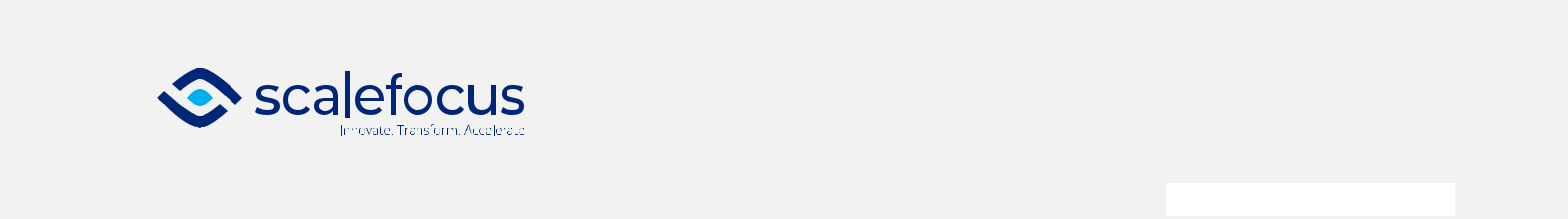
Text

Description automatically generated

1. Examine the containers args commands especially the line that start with touch. This bash pipeline will help us to test the liveness probes.
2. Run **kubectl create –f probes\_exec.yaml**.

Text

Description automatically generated

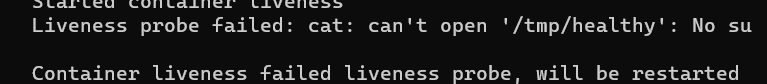
1. Run **kubectl describe pod liveness-exec** immediately after you deploy the pod. The output should indicate that no liveness probes have failed yet. Text

   Description automatically generated
2. After 35 seconds, view the Pod events again. Run **kubectl describe pod liveness-exec.**

Graphical user interface, text

Description automatically generated

1. At the bottom of the output, there should be a messages indicating that the liveness probes have failed, and the containers have been killed and recreated.

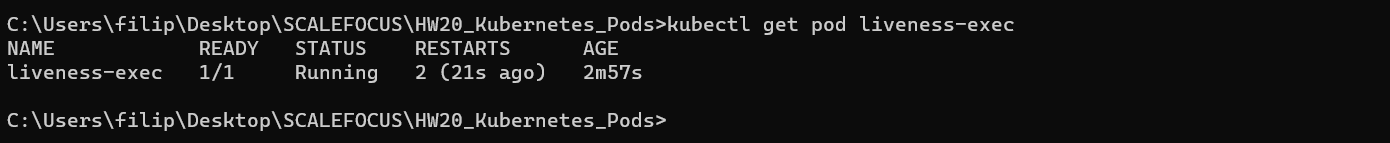


1. Wait another 30 seconds, and verify that the container has been restarted. Run **kubectl get pod liveness-exec.**

Text

Description automatically generated

1. The output should show that RESTARTS has been incremented.



1. We will continue with HTTP probe. Create file named probes\_http.yaml with following content:

**apiVersion**: v1



**kind**: Pod



**metadata**:



**labels**:



**test**: liveness



**name**: liveness-http



**spec**:

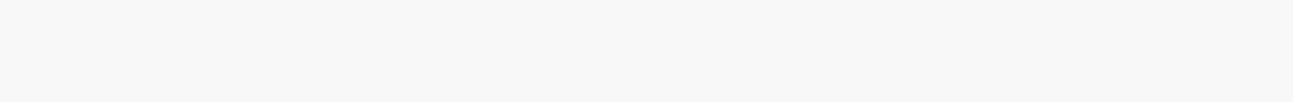


**containers**:



* **name**: liveness

**image**: k8s.gcr.io/liveness **args**:

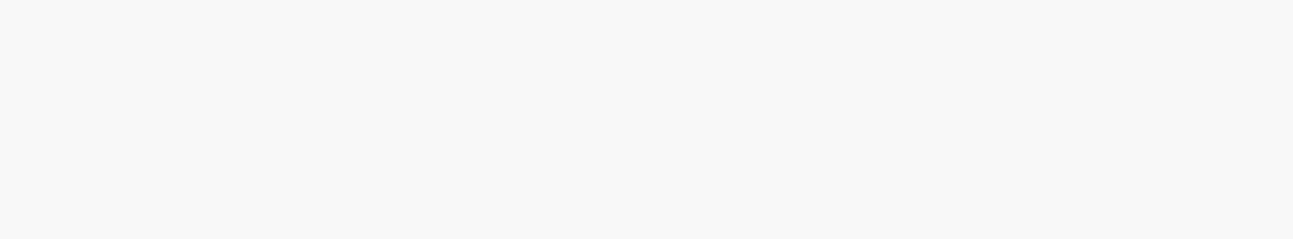


* /server **livenessProbe**: **httpGet**:

**path**: /healthz

**port**: 8080 **httpHeaders**:

* + **name**: Custom-Header



**value**: Awesome



**initialDelaySeconds**: 3



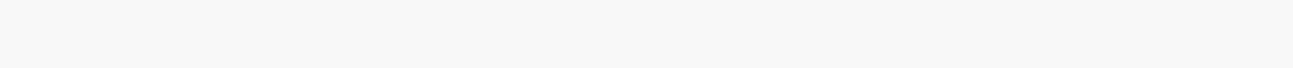
**periodSeconds**: 3

Text

Description automatically generated

10. Just for your info, /healtz handler has following function implemented:

http.HandleFunc("/healthz", **func**(w http.ResponseWriter, r \*http.Request) { duration := time.Now().Sub(started)



**if** duration.Seconds() > 10 {



w.WriteHeader(500)



w.Write([]byte(fmt.Sprintf("error: %v", duration.Seconds())))



} **else** {



w.WriteHeader(200)



w.Write([]byte("ok"))

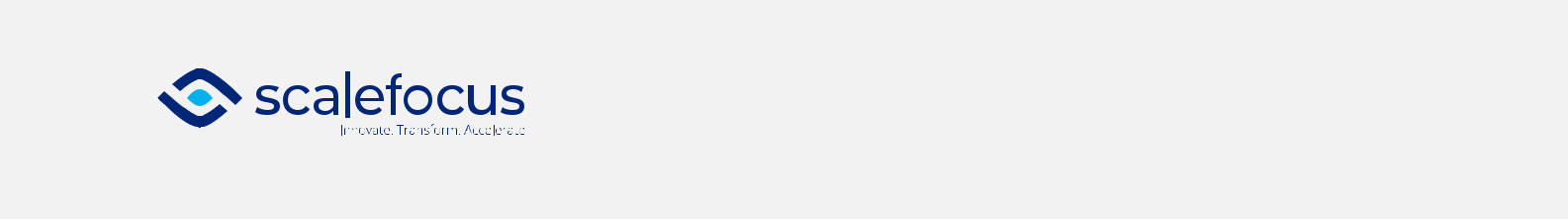


}



})



1. For the first 10 seconds that the container is alive, the /healthz handler returns a status of 200. After that, the handler returns a status of 500.
2. Run **kubectl create –f probes\_http.yaml.**

Text

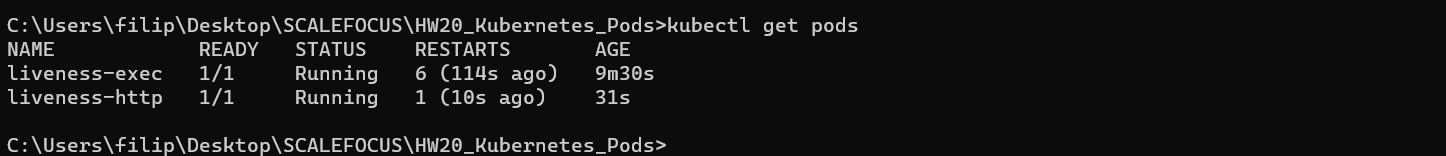
Description automatically generated

1. Immediately run (you only have 10 secs to run this command) **kubectl describe pod liveness-http.**

Text

Description automatically generated

1. Your pod should be live and running.



1. After 10 seconds, view Pod events to verify that liveness probes have failed and the container has been restarted. Run again **kubectl describe pod liveness-http.**

Text

Description automatically generated

1. You should see the same output as in step 7. Kubelet will reboot he container.

Text

Description automatically generated with medium confidence

1. We continue with TCP probes. Create file named probes\_tcp.yaml with following content:

**apiVersion**: v1



**kind**: Pod



**metadata**:



**name**: liveness-tcp



**labels**:



**app**: goproxy



**spec**:



**containers**:



* **name**: goproxy

**image**: k8s.gcr.io/goproxy:0.1 **ports**:

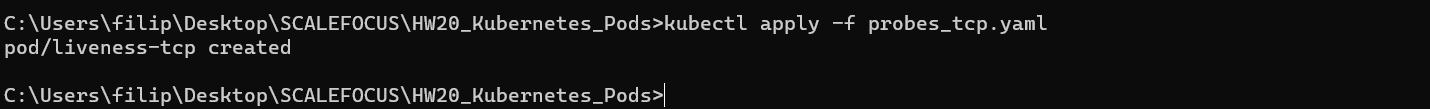
* + **containerPort**: 8080 **livenessProbe**: **tcpSocket**:

**port**: 9999 #8080 is valid port **initialDelaySeconds**: 15 **periodSeconds**: 20

Text

Description automatically generated

* + 1. Run **kubectl create –f probes\_tcp.yaml.**



* + 1. Immediately run (you only have 10 secs to run this command) **kubectl describe pod liveness-tcp.**

Text

Description automatically generated

* + 1. Your pod should be live and running.
    2. After 10 seconds, view Pod events to verify that liveness probes have failed and the container has been restarted. Run again **kubectl describe pod liveness-tcp.**

Text

Description automatically generated

* + 1. You should see the same output as in step 7 and 16. Kubelet will reboot he container.
    2. Our last job will be to define one readiness probe using HTTP test.
    3. Create file named readiness\_http.yaml with following content:



**apiVersion**: v1



**kind**: Pod



**metadata**:



**name**: readiness-http



**labels**:



**app**: test



**spec:**

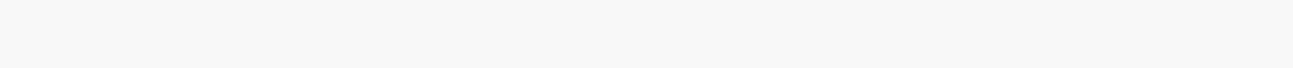


**containers:**



* **name: nginx**

**image: nginx**



**ports:**

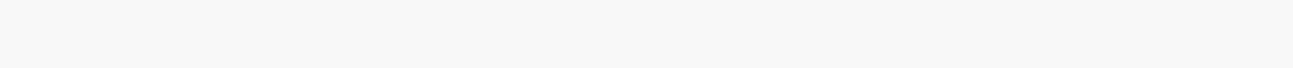
* **containerPort: 80 readinessProbe: initialDelaySeconds: 1 periodSeconds: 2 timeoutSeconds: 1 successThreshold: 1 failureThreshold: 1 httpGet:**

**host: scheme: HTTP path: / httpHeaders:**



* **name: Host**

**value: myapplication1.com**

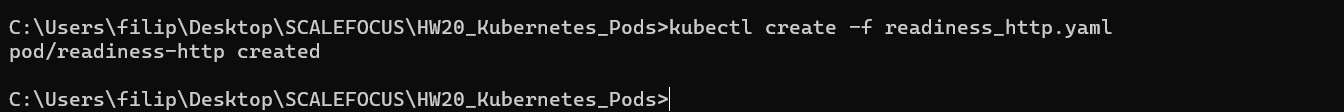


**port: 80**

Text

Description automatically generatedRun **kubectl create –f readiness\_http.yaml**.

1. Run **kubectl get pods –A** to see the status of your pod.

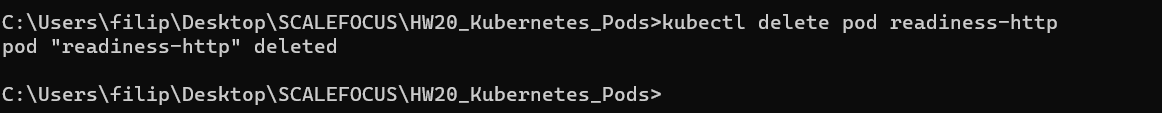


1. Pods and their status and ready states will be displayed; our pod should be in running state.
2. Run **kubectl describe pod readiness-http**. Examine the events for this pod. Everything should be OK.

Text

Description automatically generated

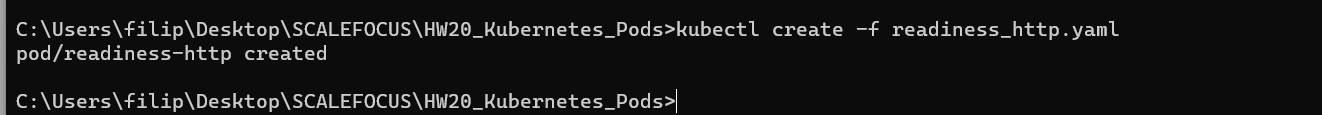
1. Now delete the pod and edit the readiness\_http.yaml so that the port parameter has 81 value.



Text

Description automatically generated

1. Run again **kubectl create –f readiness\_http.yaml.**



1. Run **kubectl get pods –A** to see the status of your pod. You should see that the pod is running but it is not in ready state.

Text

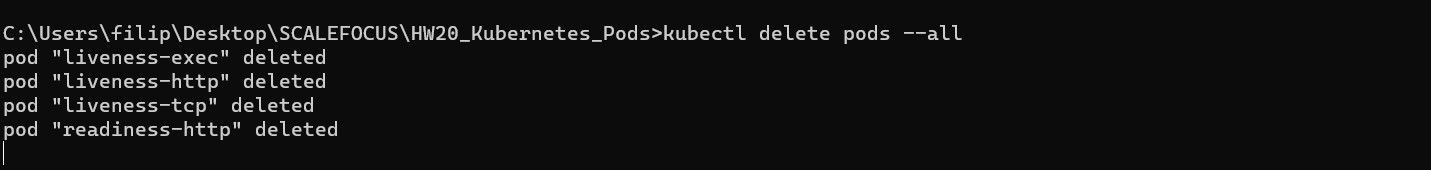
Description automatically generated

1. Describe the pod. Run **kubectl describe pod readiness-http.**

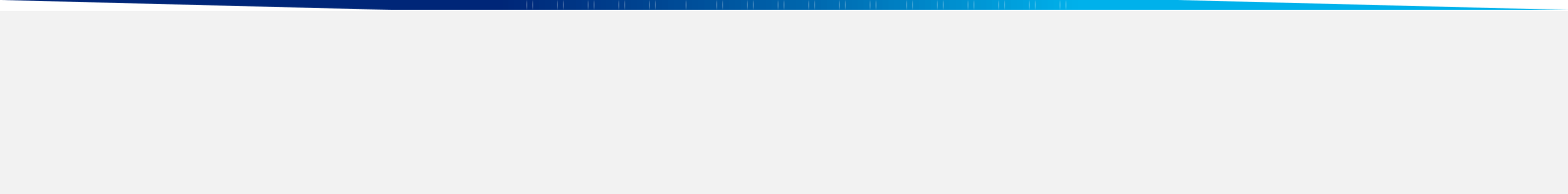
Text

Description automatically generated

1. From the events we can see that readiness probe failed due to the connection being refused therefore pod will not receive any traffic.
2. Delete all pods under the default namespace.



1. Don’t delete any manifest files created so far.



69 Bulgaria Blvd., Infinity Tower B, fl. 8, Triaditza district, 1404 Sofia, Bulgaria | +359 2424 6484 | sales@scalefocus.com | www.scalefocus.com