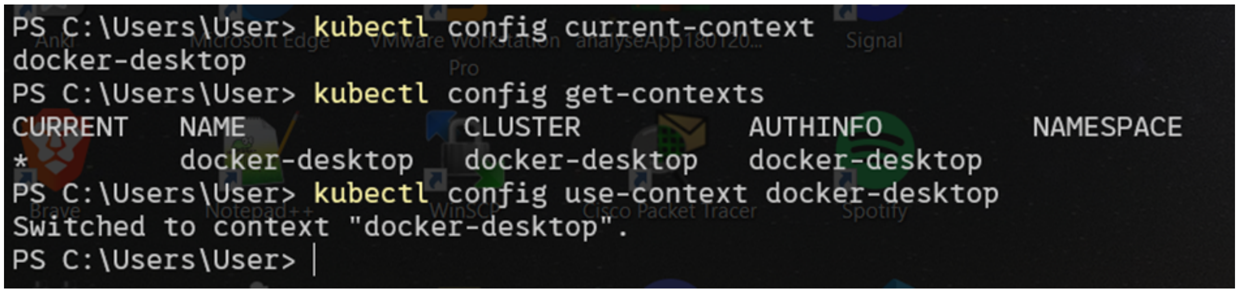
questions

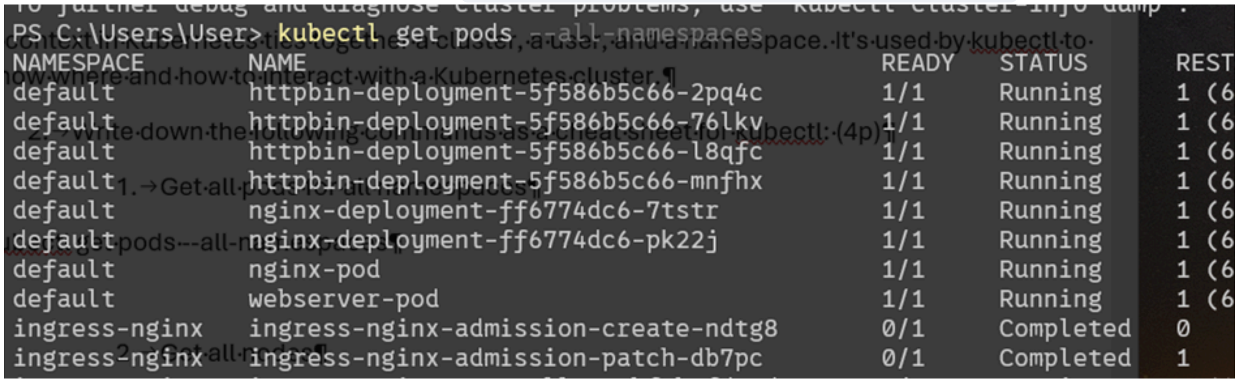
K8s Deployment Questions

1. Basic steps get familiar with kubectl and k8s (1p)
    1. How to connect to a cluster To connect to a Cluster you can use:

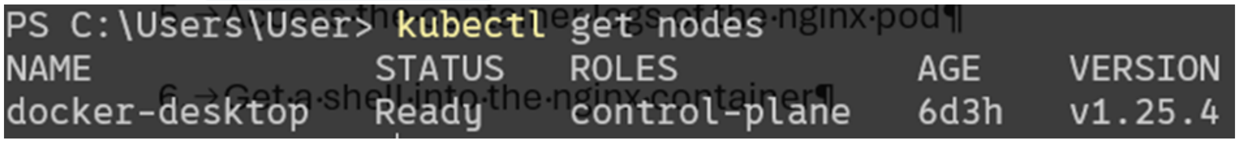
        ```
        kubectl config use-context [CLUSTER_NAME]
        ```

        in this picture below I ran `kubectl config use-context docker-desktop` to connect to my local Windows Docker-Desktop Cluster.
        

        ```
        PS C:\Users\User> kubectl config current-context
        docker-desktop
        PS C:\Users\User> kubectl config get-contexts
        CURRENT   NAME             CLUSTER          AUTHINFO         NAMESPACE
        *         docker-desktop   docker-desktop   docker-desktop
        PS C:\Users\User> kubectl config use-context docker-desktop
        Switched to context "docker-desktop".
        PS C:\Users\User> |
        ```

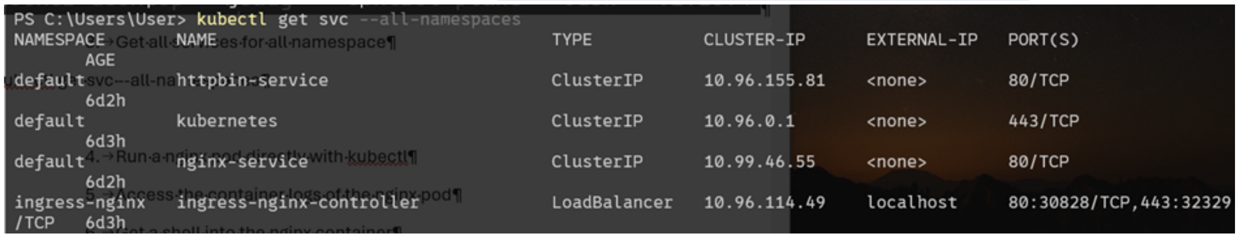
    2. What is the context used for? A context in Kubernetes ties together a cluster, a user, and a namespace. It's used by kubectl to know where and how to interact with a Kubernetes cluster.
2. Write down the following commands as a cheat sheet for kubectl: (4p)
    1. Get all pods for all namespaces `kubectl get pods --all-namespaces`
        

        ```
        To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
        PS C:\Users\User> kubectl get pods --all-namespaces
        NAMESPACE       NAME                                        READY   STATUS      REST
        default         httpbin-deployment-5f586b5c66-2pq4c         1/1     Running     1 (6
        default         httpbin-deployment-5f586b5c66-76lkv         1/1     Running     1 (6
        default         httpbin-deployment-5f586b5c66-l8qfc         1/1     Running     1 (6
        default         httpbin-deployment-5f586b5c66-mnfhx         1/1     Running     1 (6
        default         nginx-deployment-ff6774dc6-7tstr            1/1     Running     1 (6
        default         nginx-deployment-ff6774dc6-pk22j            1/1     Running     1 (6
        default         nginx-pod                                   1/1     Running     1 (6
        default         webserver-pod                               1/1     Running     1 (6
        ingress-nginx   ingress-nginx-admission-create-ndtg8        0/1     Completed   0
        ingress-nginx   ingress-nginx-admission-patch-db7pc         0/1     Completed   1
        ```

    2. Get all nodes `kubectl get nodes`
        

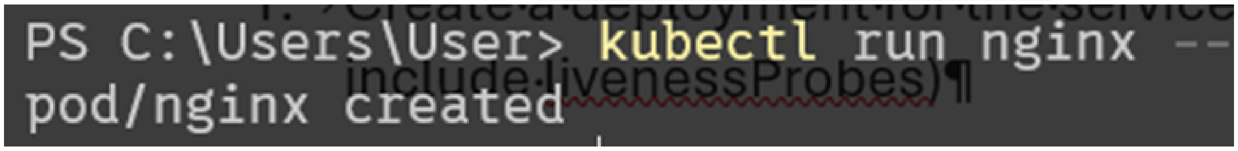
        ```
        PS C:\Users\User> kubectl get nodes
        NAME             STATUS   ROLES           AGE    VERSION
        docker-desktop   Ready    control-plane   6d3h   v1.25.4
        ```

   3. Get all services for all namespace `kubectl get svc --all-namespaces`



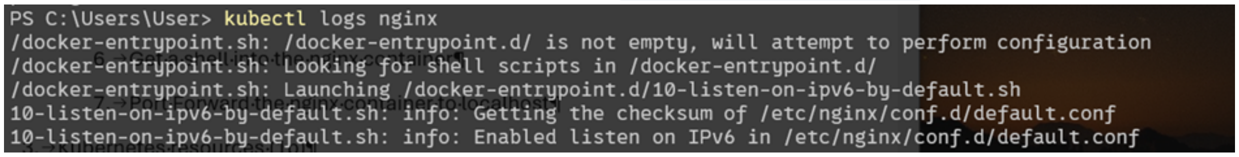
```
PS C:\Users\User> kubectl get svc --all-namespaces
NAMESPACE       NAME                                   TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)
         AGE
default         httpbin-service                        ClusterIP      10.96.155.81    <none>        80/TCP
         6d2h
default         kubernetes                             ClusterIP      10.96.0.1       <none>        443/TCP
         6d3h
default         nginx-service                          ClusterIP      10.99.46.55     <none>        80/TCP
         6d2h
ingress-nginx   ingress-nginx-controller               LoadBalancer   10.96.114.49    localhost     80:30828/TCP,443:32329
/TCP     6d3h
```

   4. Run a nginx pod directly with kubectl `kubectl run nginx --image=nginx`



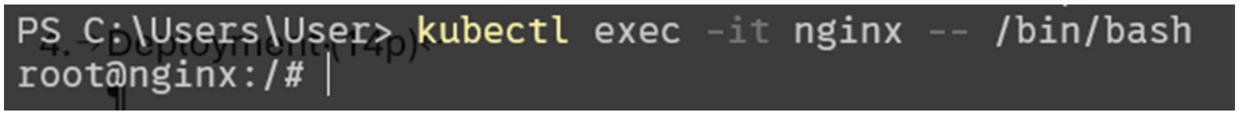
```
PS C:\Users\User> kubectl run nginx --
pod/nginx created
```

   5. Access the container logs of the nginx pod `kubectl logs nginx`



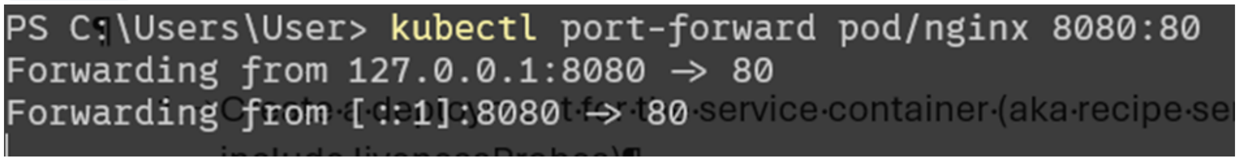
```
PS C:\Users\User> kubectl logs nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
```

   6. Get a shell into the nginx container `kubectl exec -it nginx -- /bin/bash`



```
PS C:\Users\User> kubectl exec -it nginx -- /bin/bash
root@nginx:/# |
```

   7. Port Forward the nginx container to localhost `kubectl port-forward pod/nginx 8080:80`



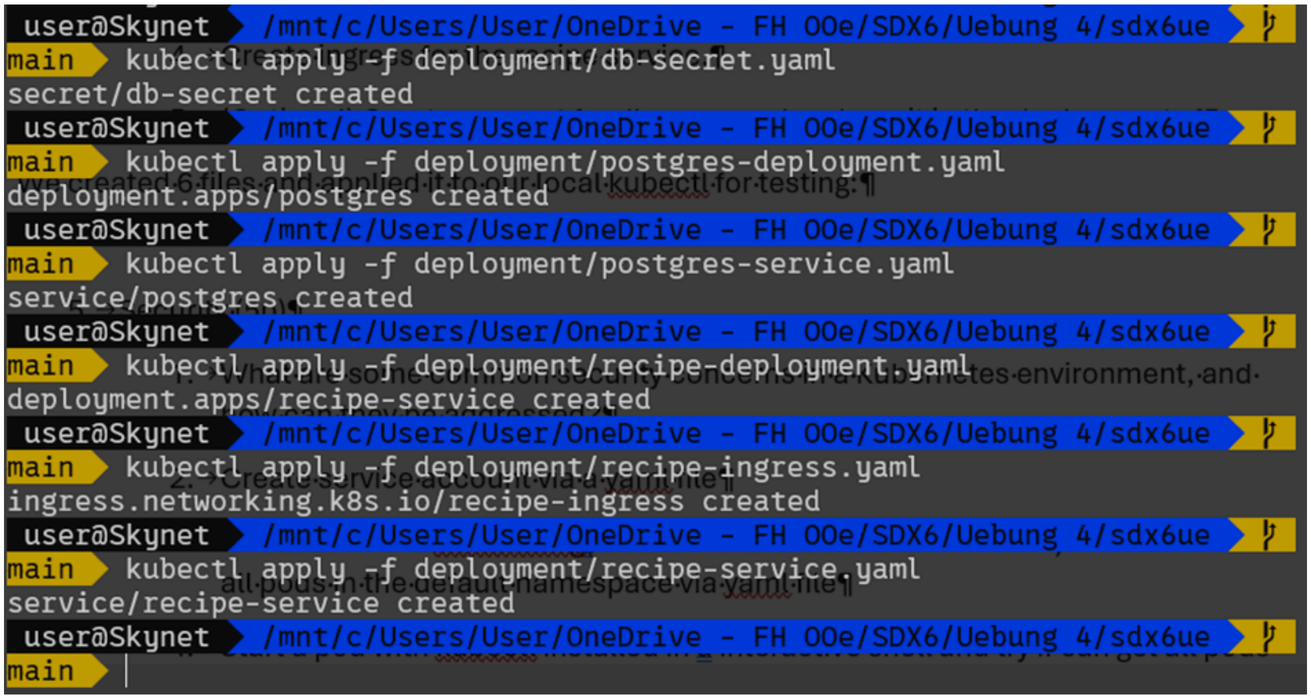
```
PS C:\Users\User> kubectl port-forward pod/nginx 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

3. Kubernetes resources (1p)
   1. What are the most common kubernetes resources/objects? Pod: Smallest deployable unit. Deployment: Manages ReplicaSets (and thus Pods). Service: Stable network endpoint for Pods. Ingress: HTTP(S) routing to Services. ConfigMap, Secret, PersistentVolume, etc.
   2. What are the different methods to make a service available outside of the k8s cluster? NodePort LoadBalancer Ingress Port-forward (local dev only)

---

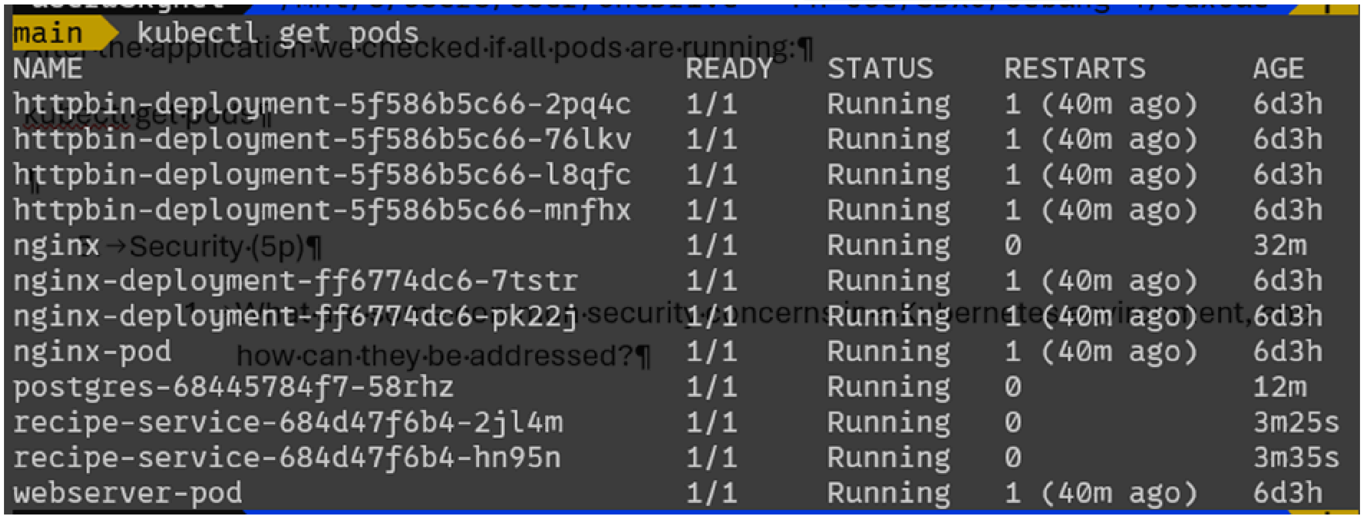
4. Deployment (14p) 1. Create a deployment for the service container (aka recipe service). (Should include livenessProbes) 2. Craate a deployment for the postgresql container. (Should include livenessProbes, no need for persistence yet) 3. Create a service for both deployments. 4. Create ingress for the recipe service. 5. (Optional) Create a secret for db password and use it in the deployments. We created 6 files and applied it to our local

kubectl for testing:



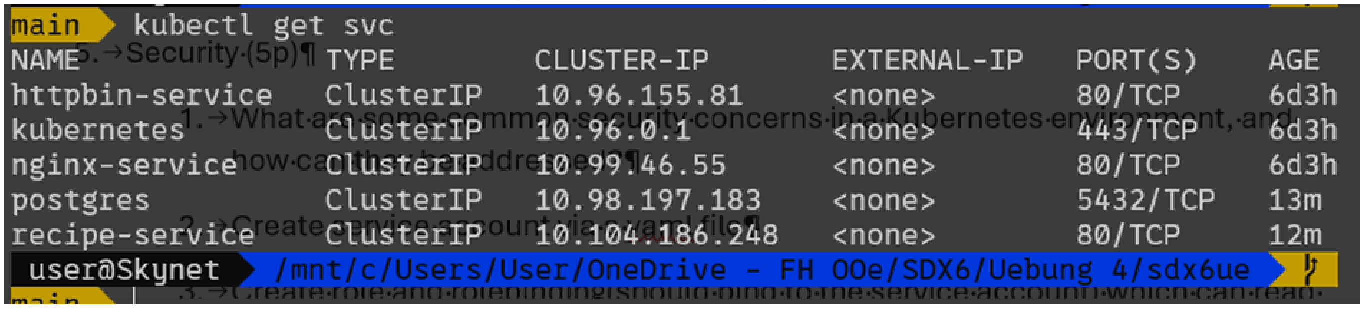
```
user@Skynet  /mnt/c/Users/User/OneDrive - FH OOe/SDX6/Uebung 4/sdx6ue
main  kubectl apply -f deployment/db-secret.yaml
secret/db-secret created
user@Skynet  /mnt/c/Users/User/OneDrive - FH OOe/SDX6/Uebung 4/sdx6ue
main  kubectl apply -f deployment/postgres-deployment.yaml
deployment.apps/postgres created
user@Skynet  /mnt/c/Users/User/OneDrive - FH OOe/SDX6/Uebung 4/sdx6ue
main  kubectl apply -f deployment/postgres-service.yaml
service/postgres created
user@Skynet  /mnt/c/Users/User/OneDrive - FH OOe/SDX6/Uebung 4/sdx6ue
main  kubectl apply -f deployment/recipe-deployment.yaml
deployment.apps/recipe-service created
user@Skynet  /mnt/c/Users/User/OneDrive - FH OOe/SDX6/Uebung 4/sdx6ue
main  kubectl apply -f deployment/recipe-ingress.yaml
ingress.networking.k8s.io/recipe-ingress created
user@Skynet  /mnt/c/Users/User/OneDrive - FH OOe/SDX6/Uebung 4/sdx6ue
main  kubectl apply -f deployment/recipe-service.yaml
service/recipe-service created
user@Skynet  /mnt/c/Users/User/OneDrive - FH OOe/SDX6/Uebung 4/sdx6ue
main  |
```

After the application we checked if all pods are running: `kubectl get pods`



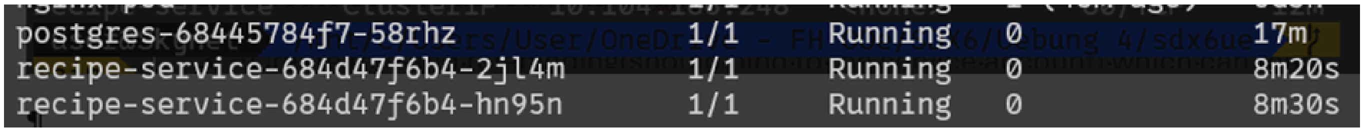
```
main  kubectl get pods
NAME                               READY   STATUS    RESTARTS      AGE
httpbin-deployment-5f586b5c66-2pq4c   1/1     Running   1 (40m ago)   6d3h
httpbin-deployment-5f586b5c66-76lkv   1/1     Running   1 (40m ago)   6d3h
httpbin-deployment-5f586b5c66-l8qfc   1/1     Running   1 (40m ago)   6d3h
httpbin-deployment-5f586b5c66-mnfhx   1/1     Running   1 (40m ago)   6d3h
nginx                                 1/1     Running   0             32m
nginx-deployment-ff6774dc6-7tstr      1/1     Running   1 (40m ago)   6d3h
nginx-deployment-ff6774dc6-pk22j      1/1     Running   1 (40m ago)   6d3h
nginx-pod                             1/1     Running   1 (40m ago)   6d3h
postgres-68445784f7-58rhz             1/1     Running   0             12m
recipe-service-684d47f6b4-2jl4m       1/1     Running   0             3m25s
recipe-service-684d47f6b4-hn95n       1/1     Running   0             3m35s
webserver-pod                         1/1     Running   1 (40m ago)   6d3h
```

After that we checked the services: `kubectl get svc`



```
main  kubectl get svc
NAME             TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
httpbin-service  ClusterIP   10.96.155.81     <none>        80/TCP     6d3h
kubernetes       ClusterIP   10.96.0.1        <none>        443/TCP    6d3h
nginx-service    ClusterIP   10.99.146.55     <none>        80/TCP     6d3h
postgres         ClusterIP   10.98.197.183    <none>        5432/TCP   13m
recipe-service   ClusterIP   10.104.186.248   <none>        80/TCP     12m
user@Skynet  /mnt/c/Users/User/OneDrive - FH OOe/SDX6/Uebung 4/sdx6ue
```

To confirm that our livenessProbes are working we let the pods run for a bit:

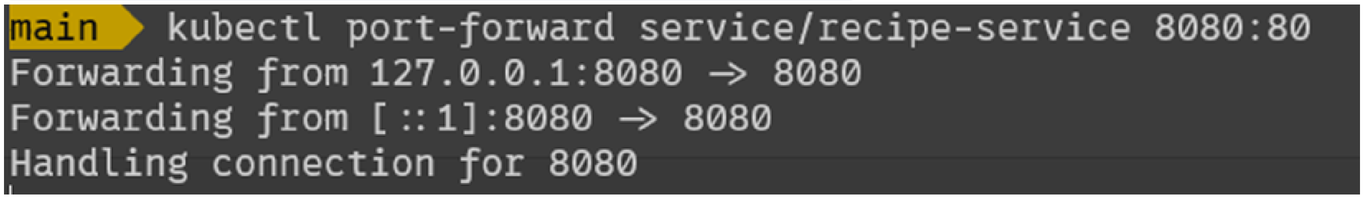


```
postgres-68445784f7-58rhz          1/1     Running   0          17m
recipe-service-684d47f6b4-2jl4m    1/1     Running   0          8m20s
recipe-service-684d47f6b4-hn95n    1/1     Running   0          8m30s
```

As we can observe after 8 minutes of running there were no restarts or crashed in to be seen through `kubectl describe pod <pod name>` so they are running fine.

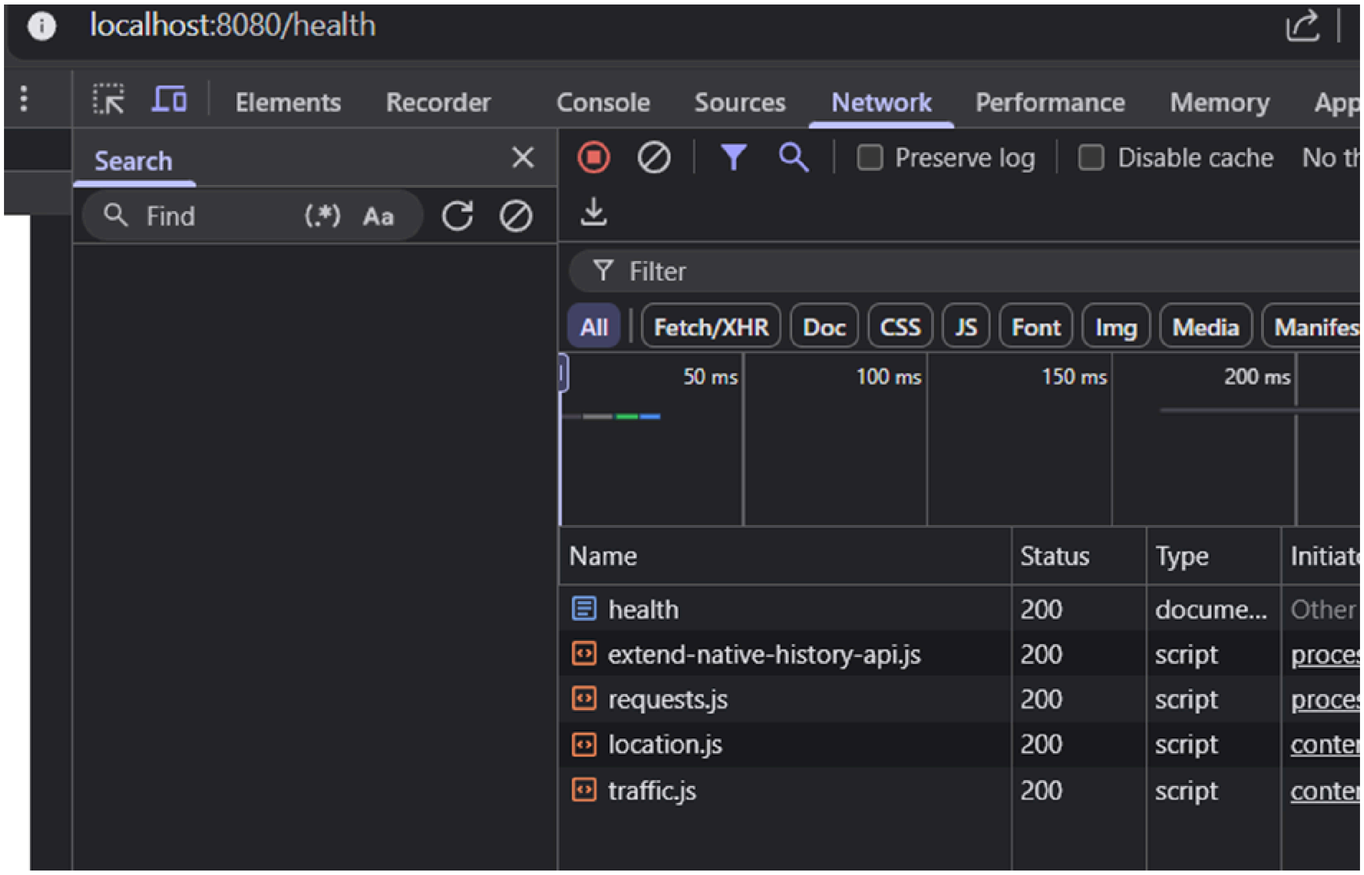
To test that the recipe-service is running correctly we forwarded the port to 8080:

`kubectl port-forward service/recipe-service 8080:80`



```
main  kubectl port-forward service/recipe-service 8080:80
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
Handling connection for 8080
```

Then opened our browser with the url “localhost:8080/health” and got a HTTP Status Code 200 confirming the service is running correctly.



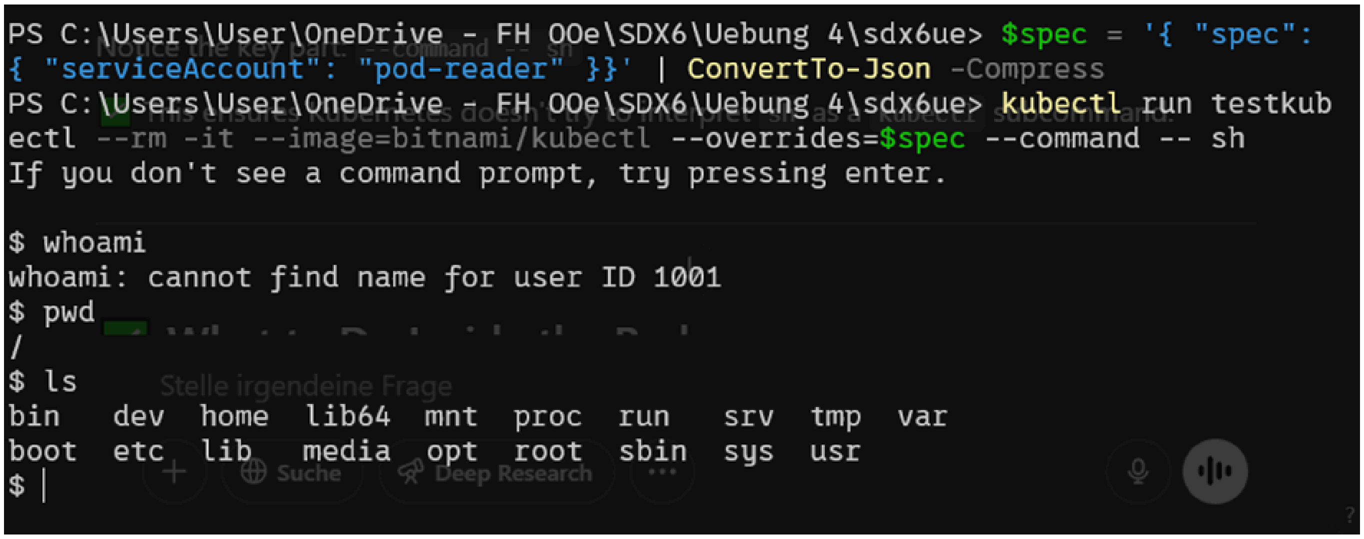
Name	Status	Type
health	200	docume...
extend-native-history-api.js	200	script
requests.js	200	script
location.js	200	script
traffic.js	200	script

5. Security (5p)
    1. What are some common security concerns in a Kubernetes environment, and how can they be addressed?

Concern	Description	Solution
Over-permissive RBAC	Users/pods might have more privileges than needed	Use least privilege via roles and role bindings
Secret leakage	Secrets may be exposed in plaintext or logs	Store secrets in Kubernetes Secrets, use volumeMounts, and enable encryption at rest
Running containers as root	Increases risk of container breakout	Use securityContext to drop root privileges
Unrestricted service exposure	Services (e.g. databases) may be exposed to the public	Use Ingress + firewall rules; avoid LoadBalancer on internal services
Image vulnerabilities	Pulling untrusted images can introduce exploits	Use signed, scanned images from trusted sources (e.g., Trivy, GHCR)
Pod-to-pod traffic unrestricted	Any pod can talk to any other pod	Apply NetworkPolicies

```
2. Create service account via a yaml file
3. Create role and rolebinding(should bind to the service account) which can
read all pods in the default namespace via yaml file
4. Start a pod with kubectl installed in a interactive shell and try if can
get all pods in this namespace. (Tip: Add Service Account to run command `--
overrides='{ "spec": { "serviceAccount": "<name>" }  }'`
```

Its all working:



```
PS C:\Users\User\OneDrive - FH OOe\SDX6\Uebung 4\sdx6ue> $spec = '{ "spec":
{ "serviceAccount": "pod-reader" }}' | ConvertTo-Json -Compress
PS C:\Users\User\OneDrive - FH OOe\SDX6\Uebung 4\sdx6ue> kubectl run testkub
ectl --rm -it --image=bitnami/kubectl --overrides=$spec --command -- sh
If you don't see a command prompt, try pressing enter.

$ whoami
whoami: cannot find name for user ID 1001
$ pwd
/
$ ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
$ |
```