

Block 6: Openshift Plattform kennenlernen

Auftrag 6.1: OpenShift APPUIO Projekt einrichten

Container Registry Login

Als Container-Registry, verwenden wir die Github Container Registry. Um Images zu verwalten und in OpenShift abzufragen, befolgen Sie folgende Schritte:

```
minikube@minikube:~$ oc login --server=https://api.exoscale-ch-gva-2-0.appuio.cloud:6443
Logged into "https://api.exoscale-ch-gva-2-0.appuio.cloud:6443" as "zlic-mregli1" using the token provided.

You have access to the following projects and can switch between them with 'oc project <projectname>':

* kanishan
  projecttest
  silvan
  zli-bensch
  zli-jashee
  zlic-adonat1
  zlic-afischer1
  zlic-amilione1
  zlic-cpeter1
  zlic-dthoma1
  zlic-eoberle1
  zlic-fbosshard1
  zlic-jlevell1
  zlic-jschafli1
  zlic-jwetli
  zlic-jwetli1
  zlic-ldaniels1
  zlic-lperrez1
  zlic-mkos1
  zlic-mregli1
  zlic-msagaaro1
  zlic-nhofer1
  zlic-nmuller1
  zlic-nwyler1
  zlic-pdietzel1
  zlic-pkrispel1
  zlic-rbartschi1
  zlic-rharadin1
  zlic-rpuvaneswaran1
  zlic-skohler1
  zlic-smuggli1
  zlic-tklingler1

Using project "kanishan".
```

Schritt 1 - Token Erstellen

Erstellen Sie unter folgendem Link einen (Classic) Personal Access Token (PAT) mit den write:packages Berechtigungen.

<https://github.com/settings/tokens>

Schritt 2 - Docker Login

Loggen Sie sich nun lokal in die Container Registry ein. Als Passwort dient ihnen der zuvor erstellte PAT.

```
docker login ghcr.io -u <Github Username>
```

```
minikube@minikube:~$ docker login ghcr.io -u masluse
Password:
WARNING! Your password will be stored unencrypted in /home/minikube/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

Schritt 3 - OpenShift Login

Damit OpenShift ihre privaten Container-Images herunterladen kann, müssen sie für ihr Projekt die Zugangsdaten definieren. Stellen Sie sicher, dass Sie sich auf dem korrekten Projekt befinden:

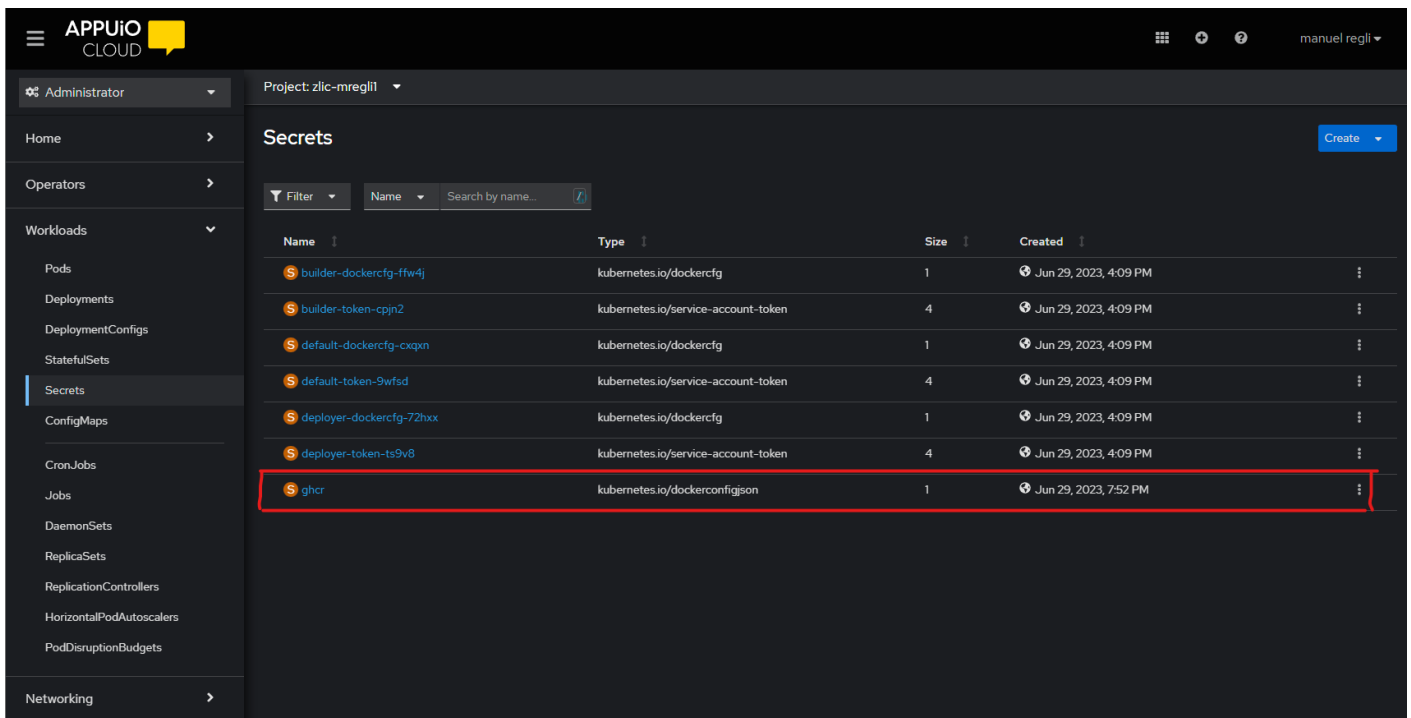
```
oc project <openshift username>
```

```
minikube@minikube:~$ oc project zlic-mregli1
Now using project "zlic-mregli1" on server "https://api.exoscale-ch-gva-2-0.appuio.cloud:6443".
```

Erstellen Sie anschliessend ein Secret mit den Credentials und konfigurieren Sie die Credentials als Default:

```
oc create secret docker-registry ghcr --docker-server=ghcr.io --docker-username=<Github Username> --docker-password=<Github PAT>
oc secrets link default ghcr --for=pull
```

```
minikube@minikube:~$ oc create secret docker-registry ghcr --docker-server=ghcr.io --docker-username=masluse --docker-password=secret/ghcr created
minikube@minikube:~$ oc secrets link default ghcr --for=pull
minikube@minikube:~$ |
```



Auftrag 6.2: HTML-Seite auf OpenShift deployen

1. Erstellen Sie einen neuen Ordner und kopieren Sie die HTML Seite aus Auftrag 1.1 und das Dockerfile aus Auftrag 3.2. Ersetzen Sie das Baseimage "nginx" mit "nginxinc/nginx-unprivileged".

```
cd
mkdir auftrag-6.2
cd auftrag-6.2
echo '<H1> Auftrag 6.2: HTML-Seite auf OpenShift deployen </H1>' > index.html
echo 'FROM nginxinc/nginx-unprivileged' > Dockerfile
echo 'WORKDIR /usr/share/nginx/html' >> Dockerfile
echo 'COPY index.html index.html' >> Dockerfile
```

2. Builden und testen Sie den Container, sodass dieser die Webseite auf Port 8080 publiziert.

```
docker build -t ghcr.io/<Github Username>/html-page:v1 .
docker run -p 8080:8080 ghcr.io/<Github Username>/html-page:v1
```



Auftrag 6.2: HTML-Seite auf OpenShift deployen

3. Pushen Sie den Container in die Github Container Registry:

```
docker push ghcr.io/<Github Username>/html-page:v1
```

```
minikube@minikube:~/auftrag-6.2$ docker push ghcr.io/masluse/html-page:v1
The push refers to repository [ghcr.io/masluse/html-page]
088024538e55: Pushed
5f70bf18a086: Pushed
aefa4ac94b0d: Pushed
2957d90a4b00: Pushed
27e29e7dc201: Pushed
ef67c084d4c1: Pushed
b5cd47658bf8: Pushed
5f3343bb4bed: Pushed
86dc269365df: Pushed
ac4d164fef90: Pushed
v1: digest: sha256:4443a088a44e4116ef91a6348951eaa209d070f0f97d9c81c16ec6b785273d53 size: 2
```

4. Erstellen Sie ein Deployment für den Container, wie Sie es in Auftrag 4.2 gelernt haben.
Und applizieren Sie dieses:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: m210-page
  labels:
    app: m210-page
spec:
  replicas: 3
  selector:
    matchLabels:
      app: m210-page
  template:
    metadata:
      labels:
        app: m210-page
    spec:
      containers:
        - name: m210-page
          image: ghcr.io/masluse/html-page:v1
```

ports:

- containerPort: 8080

```
oc apply -f deployment.yaml
```

```
minikube@minikube:~/auftrag-6.2$ oc apply -f deployment.yaml
deployment.apps/nginx-deployment created
```

The screenshot shows the AppUIO Cloud dashboard interface. On the left is a navigation sidebar with categories like Administrator, Home, Operators, Workloads, Networking, Storage, Builds, User Management, and Administration. The 'Workloads' section is expanded, showing 'Pods' and 'Deployments'. The 'Deployments' sub-section is selected, displaying the details for a deployment named 'm210-page' in the 'zlic-mregli' namespace. The deployment status is 'Up to date' with 3 pods. The update strategy is 'RollingUpdate'. Other details include 'Max unavailable' (25% of 3 pods), 'Max surge' (25% greater than 3 pods), 'Progress deadline seconds' (600 seconds), 'Min ready seconds' (Not configured), and 'PodDisruptionBudgets' (No PodDisruptionBudgets). The deployment was created at 'Jun 29, 2023, 9:15 PM' and has no owner.

apiVersion: apps/v1

kind: Deployment

metadata:

name: m210-page

labels:

app: m210-page

spec:

replicas: 3

selector:

matchLabels:

app: m210-page

template:

```
metadata:
labels:
  app: m210-page
spec:
containers:
- name: m210-page
  image: ghcr.io/masluse/html-page:v1
  ports:
    - containerPort: 8080
```

```
apiVersion: v1
kind: Service
metadata:
labels:
  app: m210-page
name: m210-page
spec:
ports:
- name: 8080-tcp
  port: 8080
  protocol: TCP
  targetPort: 8080
selector:
  app: m210-page
sessionAffinity: None
type: ClusterIP
```

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
labels:
  app: m210-page
name: m210-page
spec:
port:
  targetPort: 8080-tcp
to:
  kind: Service
  name: m210-page
tls:
```

```
termination: edge
```

```
insecureEdgeTerminationPolicy: Redirect
```

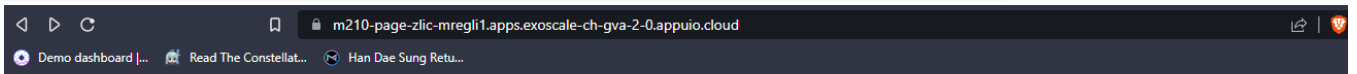
```
oc apply -f deployment.yaml
```

```
minikube@minikube:~/auftrag-6.2$ oc apply -f deployment.yaml
deployment.apps/nginx-deployment unchanged
service/m210-page created
route.route.openshift.io/m210-page created
```

5. Rufen Sie die erstellte Route ab und testen Sie die Webseite:

```
oc get routes
```

<https://m210-page-zlic-mregli1.apps.exoscale-ch-gva-2-0.appuio.cloud/>



Auftrag 6.2: HTML-Seite auf OpenShift deployen

Revision #1

Created 15 December 2023 14:13:02 by Manuel Regli

Updated 15 December 2023 14:19:01 by Manuel Regli