Installation

- To install MicroK8s on Ubuntu, you can follow these steps
- To install the Kubernetes dashboard and set up HTTPS access, you can follow these steps:
- To automatically start the Kubernetes dashboard after booting your Ubuntu machine
- kubectl
- Change kubectl editor
- Dashboard WithOut Proxy
- Dashboard - TimeOut
- Dashboard Get Token
- To use a container registry with MicroK8s, you can follow these steps:
- Installationanleitung 2

To install MicroK8s on Ubuntu, you can follow these steps

To install MicroK8s on Ubuntu, you can follow these steps:

1. Open a terminal on your Ubuntu machine.

2. Update your system's package list by running the following command:

```
sudo apt update
```

3. Install MicroK8s using snap, which is a package management system on Ubuntu:

```
sudo snap install microk8s --classic
```

The `--classic` flag is used to enable classic snap confinement, allowing MicroK8s to access the host system.

4. After the installation is complete, add your user to the `microk8s` group so that you can run MicroK8s commands without using `sudo`:

```
sudo usermod -a -G microk8s <your-username>
```

Replace `<your-username>` with your actual username.

5. To enable required permissions for your user, run the following command:

```
sudo chown -f -R <your-username> ~/.kube
```

6. You will need to log out and log back in for the group changes to take effect. So, close the terminal and open a new one.

7. Verify that MicroK8s is up and running by checking the status:

```
microk8s status --wait-ready
```

This command will wait until all the MicroK8s services are running.

8. To enable some useful add-ons, you can use the following commands:

```
microk8s enable dns
```

```
microk8s enable hostpath-storage
```

9. install Kubernetes dashboard, see nect page

To install the Kubernetes dashboard and set up HTTPS access, you can follow these steps:

To install the Kubernetes dashboard and set up HTTPS access, you can follow these steps:

1. Install the Kubernetes dashboard by running the following command:

```
microk8s enable dashboard
```

2. Verify that the dashboard has been successfully installed by checking its status:

```
microk8s kubectl get services -n kube-system
```

Look for a service named `kubernetes-dashboard` in the `kube-system` namespace. Make a note of the port number (usually 443) associated with the service.

3. Generate a self-signed SSL certificate for HTTPS access. Run the following commands to create a directory for the certificates and generate the certificate:

```
sudo mkdir /var/snap/microk8s/current/certs
```

```
sudo microk8s kubectl create secret generic kubernetes-dashboard-certs --from-file=/var/snap/microk8s/current/certs -n kube-system
```

4. Access the dashboard with HTTPS by creating a proxy using the `kubectl` command. Replace `<port>` with the port number noted from step 2:

```
microk8s kubectl proxy --port=<port> --address='0.0.0.0' --accept-hosts='.*'
```

5. Open a web browser and navigate to

```
https://localhost:<port>/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/
```

Replace `<port>` with the same port number used in step 4.

6. You will encounter a security warning due to the self-signed certificate. Accept the warning and proceed to access the Kubernetes dashboard.

Please note that using a self-signed certificate poses security risks, and it's recommended to use a valid SSL certificate from a trusted certificate authority for production environments.

To automatically start the Kubernetes dashboard after booting your Ubuntu machine

To automatically start the Kubernetes dashboard after booting your Ubuntu machine, you can create a systemd service unit. Here's how you can do it:

1. Open a terminal on your Ubuntu machine.

2. Create a new systemd service unit file using a text editor. For example, you can use the following command to create the file with the Nano text editor:
   ```
   sudo nano /etc/systemd/system/microk8s-dashboard.service
   ```

3. Add the following content to the service unit file:
   ```
   [Unit]
   Description=MicroK8s Kubernetes Dashboard
   After=network.target

   [Service]
   ExecStart=/snap/bin/microk8s.kubectl proxy --port=8001 --address='0.0.0.0' --accept-hosts='.*'
   Restart=always
   RestartSec=10

   [Install]
   WantedBy=multi-user.target
   ```

   This configuration defines a service unit for the Kubernetes dashboard, specifying the command to start the proxy and enabling automatic restarts.

4. Save the file and exit the text editor. In Nano, you can do this by pressing Ctrl+O, then Enter to save, and Ctrl+X to exit.

5. Enable the service to start automatically on boot by running the following command:
   ```
   sudo systemctl enable microk8s-dashboard
   ```

6. Start the service manually for the current session by running:
   ```
   sudo systemctl start microk8s-dashboard
   ```

7. Verify that the service is running without any errors by checking its status:
   ```
   sudo systemctl status microk8s-dashboard
   ```

   If everything is set up correctly, the status should indicate that the service is active (running).

After following these steps, the Kubernetes dashboard should start automatically after each boot. You can access it from any machine on the network by using the appropriate URL, as mentioned in the previous responses.

kubectl

sudo snap install kubectl --classic

microk8s.kubectl config view --raw > $HOME/.kube/microk8s.config

add the minik8s cluster to kubectl

~~-kubectl config set-cluster microk8s-cluster --server=http://127.0.0.1:8080 --insecure-skip-tls-verify~~

ich habe stattdessen

```
microk8s.kubectl config view --raw > $HOME/.kube/config
```

verwendet

create the microk8s context

```
kubectl config set-context microk8s --user=admin --cluster=microk8s-cluster
```

switch to the microk8s context

```
kubectl config use-context microk8s
```

die Datei auf einem externen Rechner kopieren:

```
scp USERNAME@HOST:/home/USERNAME/.kube/config config
```

Change kubectl editor

sudo -H nano /etc/environment

insert new line: KUBE_EDITOR="nano"

Dashboard WithOut Proxy

Enable additional Add-Ons

We will need to enable a few additional Kubernetes add-ons to get this functionality up and running.

```
microk8s enable ingress # Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster.
microk8s enable dashboard # web-based Kubernetes user interface
microk8s enable dns # creates DNS records for services and pods
microk8s enable storage # provide both long-term and temporary storage to Pods in your cluster.
```

Enable Host Access

We need to alos enable one more additional add-on `host-access` to enable the access to services running on the host machine via fixed IP address.

We can enable to make use of the default address

```
microk8s enable host-access
```

Edit Kubernetes Dashboard Service

We need to edit the kubernetes-dashboard service file which provides dash-board functionality. To to edit this we need to edit dashboard service and change service **type** from **ClusterIP** to **NodePort**.

We use the following command to edit the file using vim.

```
kubectl -n kube-system edit service kubernetes-dashboard
```

This should open the contents of the file in vim and it should look something similar file below

You need to change `type` to `NodePort`

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: Service
metadata:
annotations:
kubectl.kubernetes.io/last-applied-configuration: |
{"apiVersion":"v1","kind":"Service","metadata":{"annotations":{},"labels":{"k8s-app":"kubernetes-dashboard"},"name":"kubernetes-dashboard","namespace":"kube-system"},"spec":{"ports":[{"port":443,"targetPort":8443}],"selector":{"k8s-app":"kubernetes-dashboard"}}}
creationTimestamp: "2022-03-09T14:10:50Z"
labels:
k8s-app: kubernetes-dashboard
name: kubernetes-dashboard
namespace: kube-system
resourceVersion: "1029725"
selfLink: /api/v1/namespaces/kube-system/services/kubernetes-dashboard
uid: b2608643-a712-4b44-abad-160cf140df49
spec:
clusterIP: 10.152.183.220
clusterIPs:
- 10.152.183.220
externalTrafficPolicy: Cluster
internalTrafficPolicy: Cluster
ipFamilies:
- IPv4
ipFamilyPolicy: SingleStack
ports:
- nodePort: 30536
port: 443
protocol: TCP
targetPort: 8443
selector:
k8s-app: kubernetes-dashboard
sessionAffinity: None
type: clusterIP ## Change this to NodePort
status:
loadBalancer: {}
```

Once you save and exit the file K8s will automatically restart the service.

We can then get the Port the service is running by using the following command

kubectl -n kube-system get services

Which should display something similar to the below and which we can see that the Dashboard is available on port `30536` in my case

Check Port

```
sudo snap install nmap
```

```
nmap localhost -p 32061
```

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
metrics-server ClusterIP 10.152.183.108 <none> 443/TCP 7d4h
dashboard-metrics-scraper ClusterIP 10.152.183.170 <none> 8000/TCP 7d4h
kube-dns ClusterIP 10.152.183.10 <none> 53/UDP,53/TCP,9153/TCP 7d4h
kubernetes-dashboard NodePort 10.152.183.220 <none> 443:30536/TCP 7d4h

We can now open our Firefox browser on any workstation on our network and navigate to `https://{server ip}:{port number}` in my case it is https://192.168.0.35:30536

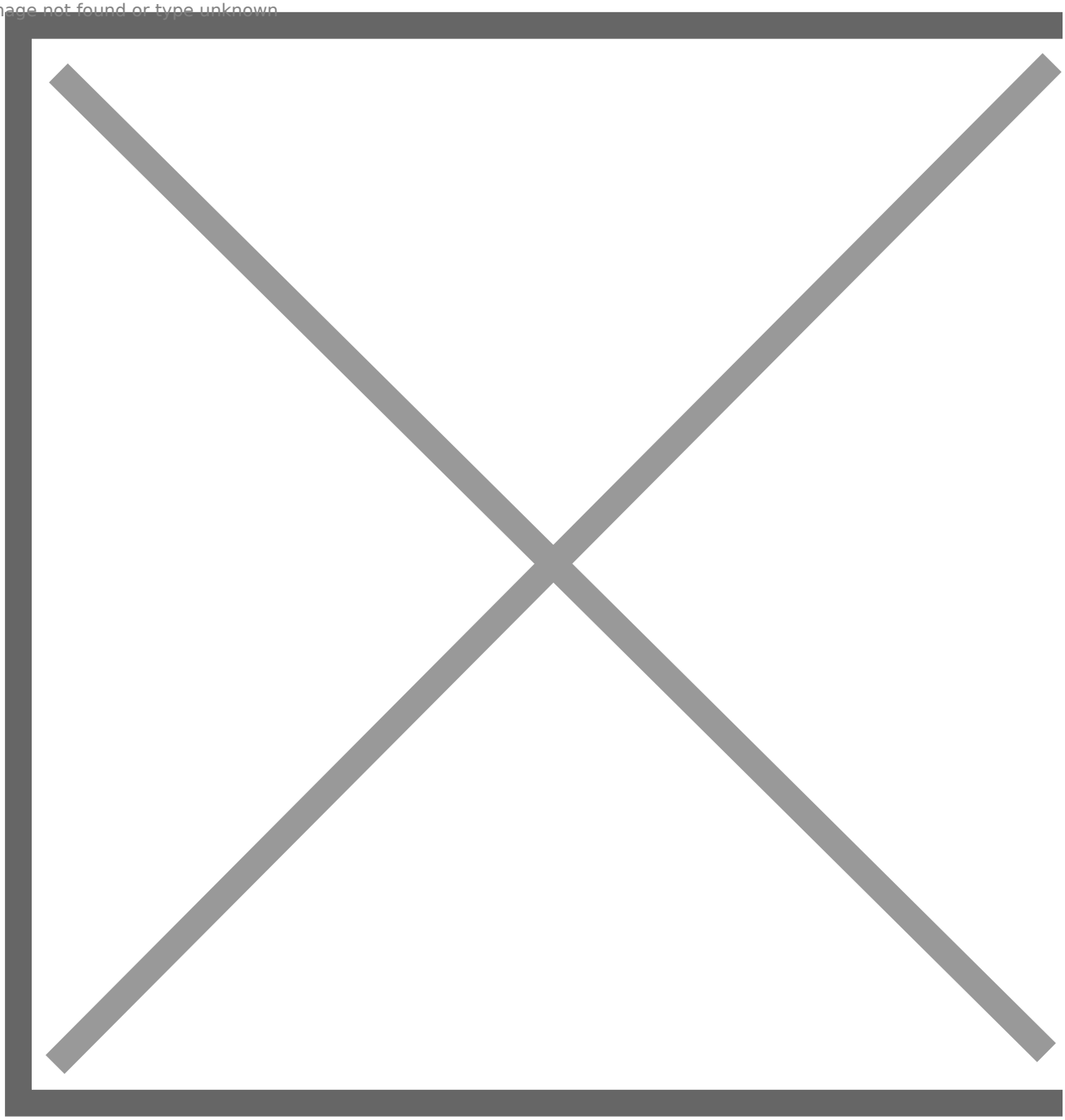
Get the token

We need to get the token from the server so we can do so using the following command in the server terminal

```
token=$(kubectl -n kube-system get secret | grep default-token | cut -d " " -f1) kubectl -n kube-system describe secret $token
```

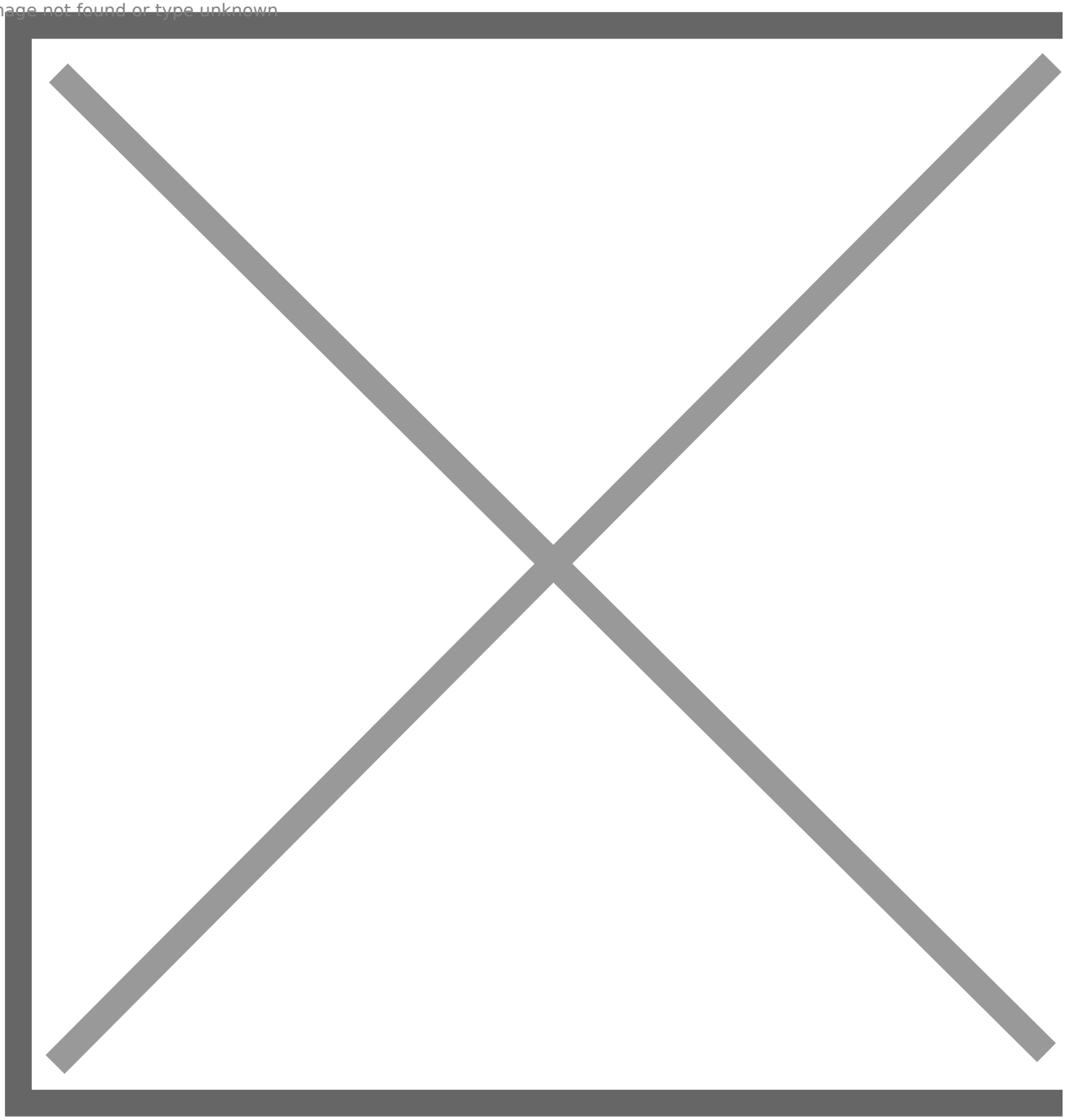
This should return something similar too

Image not found or type unknown



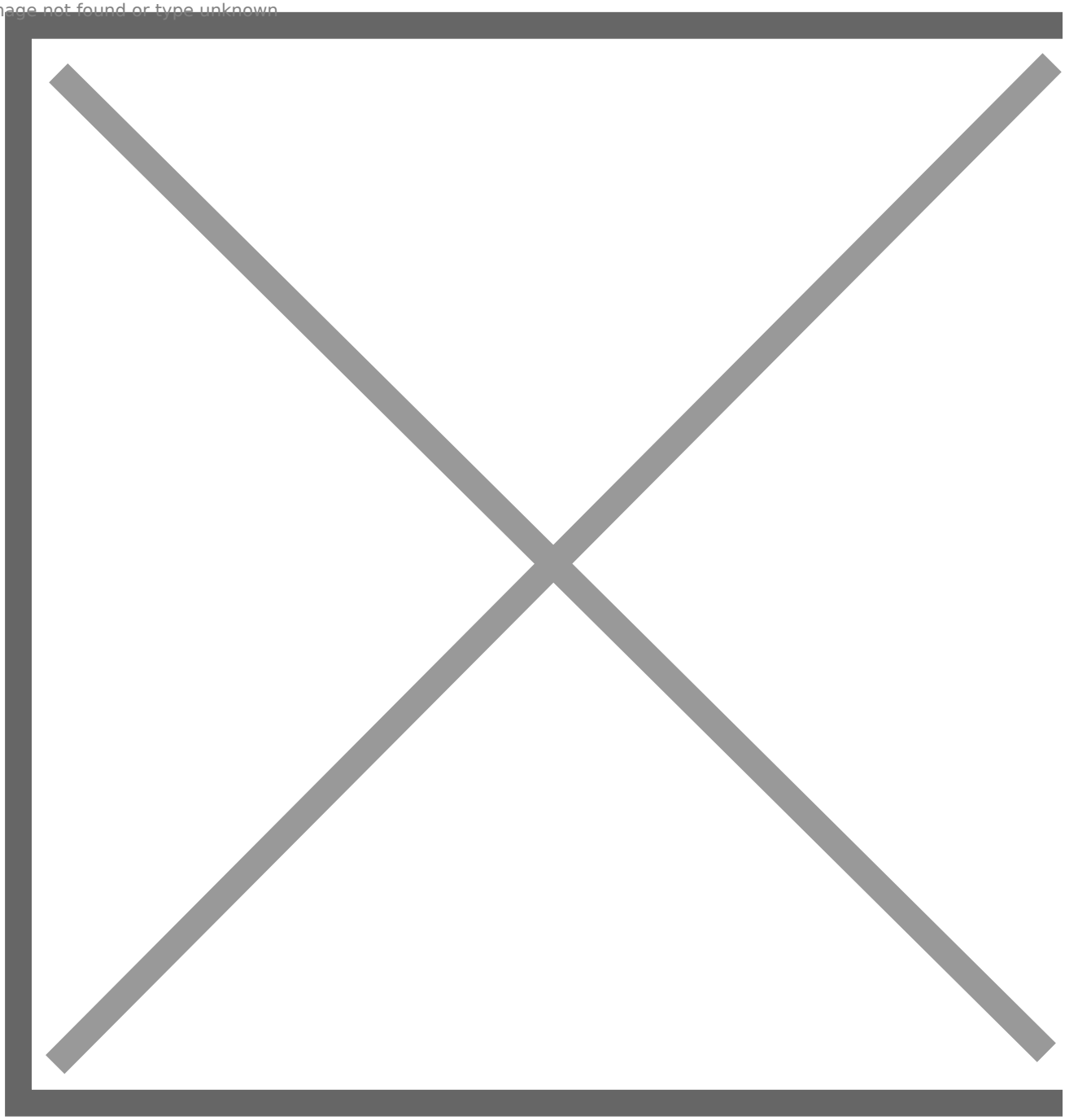
Copy the token text and paste it into the login dialog in the browser

Image not found or type unknown



Then you should be able to login with ease.

Image not found or type unknown

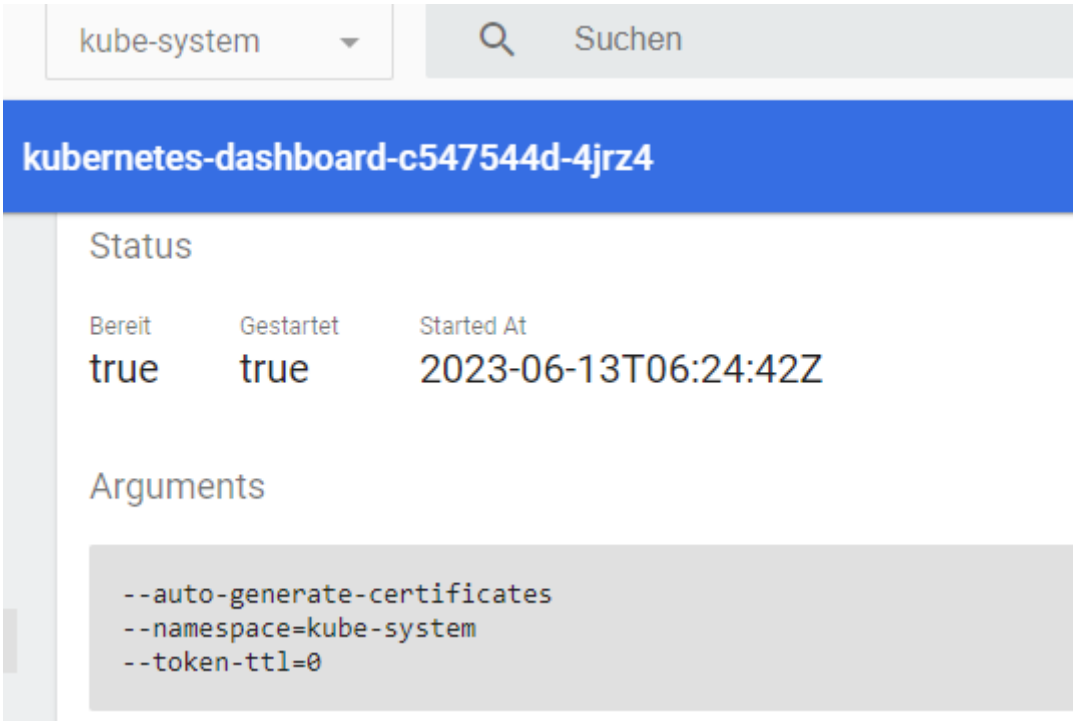


Dashboard - TimeOut

kubectl -n kube-system edit deployments kubernetes-dashboard

- --token-ttl=0

```
spec:
    containers:
    - args:
      - --auto-generate-certificates
      - --token-ttl=0
```



Dashboard Get Token

sudo cat /var/snap/microk8s/current/credentials/client.config

To use a container registry with MicroK8s, you can follow these steps:

To use a container registry with MicroK8s, you can follow these steps:

1. Enable the `registry` add-on in MicroK8s by running the following command:
   ```bash
   microk8s enable registry
   ```

   This command will start a local container registry in your MicroK8s cluster.

2. Verify that the `registry` add-on is running by checking the status:
   ```bash
   microk8s status | grep registry
   ```

   The output should show `enabled` next to the `registry` add-on.

3. Push an image to the local container registry. First, tag an existing Docker image with the address of the local registry. For example:
   ```bash
   docker tag my-image:latest localhost:32000/my-image:latest
   ```

   Replace `my-image` with the name of your image. The address `localhost:32000` corresponds to the default address of the local registry in MicroK8s.

4. Push the tagged image to the local container registry:
   ```bash
   docker push localhost:32000/my-image:latest
   ```

   This will push the image to the local registry within MicroK8s.

5. Use the image from the local registry in your Kubernetes manifests or deployments. Update your YAML files to reference the image from the local registry. For example:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: localhost:32000/my-image:latest
```

Replace `my-image` with the name of your image, and `localhost:32000` with the address of the local registry.

MicroK8s will now use the local container registry for pulling images within your cluster.

Please note that the local container registry in MicroK8s uses port `32000` by default. If you encounter any issues or conflicts with this port, you can customize the port by modifying the `registry` add-on configuration.

By following these steps, you can enable and use a local container registry with MicroK8s to push and pull container images within your cluster.

To resolve this issue, you can configure Docker to allow insecure connections to the local registry by following these steps:

1. Open the Docker configuration file (`daemon.json`) for editing. The location of this file depends on your operating system:
   - Linux: `/etc/docker/daemon.json`
   - Windows: `C:\ProgramData\Docker\config\daemon.json`
   - macOS: `~/Library/Group Containers/group.com.docker/daemon.json`
2. If the file doesn't exist, create it.
3. Add the following content to the `daemon.json` file:
   json Copy code
   ```
   {
    "insecure-registries": ["192.168.178.100:32000"]
   }
   ```
   Replace `192.168.178.100:32000` with the IP address and port of your local container registry.
4. Save the changes to the `daemon.json` file.
5. Restart the Docker daemon for the changes to take effect. The procedure for restarting the Docker daemon varies depending on your operating system. You can usually do this by restarting the Docker service or using the Docker desktop application.
6. ```
   sudo snap restart docker.dockerd
   ```
   ```

Once Docker is configured to allow insecure connections, you should be able to push images to the local container registry without encountering the HTTPS error.

Installationanleitung 2

1. The first thing to do is open a command line on your Ubuntu system and execute the following `snap` command to install MicroK8s:

   ```
   sudo snap install microk8s --classic
   ```

2. Then, execute the following commands to configure Ubuntu's ufw firewall to allow communication between the Kubernetes pods and traffic to and from the internet to pods.

   ```
   sudo ufw allow in on cni0
   sudo ufw allow out on cni0
   sudo ufw default allow routed
   ```

3. Next, add your current user to the MicroK8s user group so you have access to all of the necessary commands:

   ```
   sudo usermod -a -G microk8s $USER
   sudo mkdir ~/.kube && sudo chown -R $USER ~/.kube
   sudo newgrp microk8s
   ```

4. The MicroK8s deployment only has the essential elements enabled out of the box. You can see a full list with the following command:

   ```
   microk8s status
   ```

5. You can enable any number of needed services with the following commands. Enable all of the services that you will need:

   ```
   $ microk8s enable dns
   $ microk8s enable dashboard
   $ microk8s enable storage
   $ microk8s enable ingress
   $ microk8s enable gpu
   $ microk8s enable istio
   $ microk8s enable registry
   ```

   To disable any of these services later, just issue the same command but replace `enable` with `disable`:

   ```
   $ microk8s disable dns
   ```

---

6. You can see the status of your services with the following command. They are ready to go once they switch to the ‘Running’ status:

```
$ microk8s kubectl get all --all-namespaces
```

Image not found or type unknown
Checking the status of services in MicroK8s on Ubuntu Linux

7. As we can see in the screenshot above, our Cluster IP Kubernetes dashboard is `10.152.183.38` but yours might be different, so be sure to check. We can now navigate to this address in our browser to access the web based dashboard.

Image not found or type unknown
Accessing the MicroK8s web based dashboard in Firefox

8. To log in to the web based dashboard, generate a token with the following commands, and then paste it into login prompt in Firefox.

```
$ token=$(microk8s kubectl -n kube-system get secret | grep default-token | cut -d " " -f1)
$ microk8s kubectl -n kube-system describe secret $token
```

9. You can now get started with deploying your containerized application, or follow along with the three commands below to deploy, and expose a microbot service as an example and proof of concept:

```
$ microk8s kubectl create deployment microbot --image=dontrebootme/microbot:v1
$ microk8s kubectl scale deployment microbot --replicas=2
$ microk8s kubectl expose deployment microbot --type=NodePort --port=80 --name=microbot-service
```

10. You will see your new deployment in the output when executing:

```
$ microk8s kubectl get all --all-namespaces
```

11. To see all of the other MicroK8s commands available, you can use the `-h` option and view a full list:

```
$ microk8s -h
```

12. If, after your testing, you need to get rid of MicroK8s, you can execute:

```
$ microk8s stop
$ sudo snap remove microk8s
```