# Know-How

## kubectl Plugins

https://krew.sigs.k8s.io/plugins/

## Private Registry/Repository

### Abfrage Registry Katalog

```
curl -k https://registry.tuxnet.lan/v2/_catalog
```

Oder mit dem schöneren Tool **reg**

```
reg ls -k registry.tuxnet.lan
```

Docker Images in die private registry kopieren

```
skopeo copy docker://k8s.gcr.io/metrics-server/metrics-server:v0.5.0
docker://registry.tuxnet.lan/metrics-server/metrics-server:v0.5.0
```

### Befüllen der privaten Registry

```
oci-local -u -r registry.tuxnet.lan
```

### Löschen von Registry Einträgen/Images mittels crane

```
crane delete "registry.tuxnet.lan/iot/fhem@$(crane digest
registry.tuxnet.lan/iot/fhem:3.3.0)"
```

## Konfiguration der CRE (containerd) für die private Registrie

/etc/containerd/config.toml

```
  [plugins."io.containerd.grpc.v1.cri".registry]
    [plugins."io.containerd.grpc.v1.cri".registry.mirrors]
[plugins."io.containerd.grpc.v1.cri".registry.mirrors."registry.tuxnet.lan"]
        endpoint = ["https://registry.tuxnet.lan:32568"]
```

# Kubernetes Master-Node welche Ordner/Files auf weitere Master-Nodes kopieren

- Vorbereitungen zum Joinen der weiteren control Nodes
- Sicherung der wesentlichen Dateien, insbesondere der Kubernetes Cluster PKI

```
sudo tar cf kubestrap.tar /etc/kubernetes/admin.conf
/etc/kubernetes/audit.yaml /etc/kubernetes/manifests/lb.yaml
/etc/kubernetes/pki/ca.* /etc/kubernetes/pki/front-proxy-ca.*
/etc/kubernetes/pki/sa* /etc/kubernetes/pki/etcd/ca.*
```

Entpacken dann entsprechend

```
sudo tar xf kubestrap.tar -C /
```

# Ausgabe des Token Hash's

```
kubeadm token create --dry-run --print-join-command
```

# Zertifikat-Anfragen von Nodes genehmigen

Auflisten aller Anfragen

```
kubectl get csr
```

Anfragen genehmigen

```
kubectl certificate approve <cert-name>
```

# Ändern des clusterweiten Konfiguration

### Anzeige der aktuellen Cluster Konfig

```
kubectl get cm -n kube-system kubeadm-config -oyaml
```

### Ändern der Konfiguration

```
kubectl edit cm -n kube-system kubeadm-config
```

## Updating resources

```
kubectl set image deployment/frontend www=image:v2               # Rolling
update "www" containers of "frontend" deployment, updating the image
kubectl rollout history deployment/frontend                      # Check the
history of deployments including the revision
kubectl rollout undo deployment/frontend                         # Rollback
to the previous deployment
kubectl rollout undo deployment/frontend --to-revision=2         # Rollback
to a specific revision
kubectl rollout status -w deployment/frontend                    # Watch
rolling update status of "frontend" deployment until completion
kubectl rollout restart deployment/frontend                      # Rolling
restart of the "frontend" deployment
```

## new command or entry point for a container

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: golang
  name: golang
  namespace: testing-chantal
spec:
  containers:
  - name: golang
    image: golang:bookworm
    tty: true
    stdin: true
    command: ["/bin/sh"]
    resources: {}
    volumeMounts:
    - name: bin
      mountPath: /go/psp-mig
  imagePullSecrets:
  - name: regcred
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  volumes:
```

```
    - name: bin
      persistentVolumeClaim:
        claimName: pvc-golang
status: {}
```