

ArgoCD

Installation Core Variante

[stable getting started](#)

Installation Helm Chart Variante

```
# install argocd via helm and customized values
## Optional
helm repo add argocd https://argoproj.github.io/argo-helm
helm repo update
## kann auch direkt machen
helm upgrade --install argocd argo-cd \
  --repo https://argoproj.github.io/argo-helm \
  --values values.yaml --create-namespace --namespace argocd
```

values.yaml:

```
# helm override name
nameOverride: argocd

# settings for argocd
global:
  # domain dns name for ingress
  domain: argocd.tuxnet.lab

#
configs:
  params:
    # tls termination is handled on ingress, so the server will run
    # via insecure mode
    server.insecure: true

server:
  # ingress settings
  ingress:
    enabled: true
    annotations:
      # we use the cert-manager for the tls certificates
      cert-manager.io/cluster-issuer: ca-issuer
      # ingressController traefik is used in this lab
      ingressClassName: "traefik"
      # and we use tls termination on ingress level
    tls: true
```

Add OCI Helm Repository

```
argocd repo add registry-1.docker.io/bitnamicharts --type helm --name
"Bitnami OCI" --enable-oci
```

Service Type Load Balancer

```
kubectl patch svc argocd-server -n argocd -p '{"spec": {"type":
"LoadBalancer", "allocateLoadBalancerNodePorts": false, "loadBalancerIP":
"<IP-ADRESSE>"}}'
```

Support for selfsigned rootCA

```
k -n argocd edit cm argocd-tls-certs-cm
```

Das rootCA hier einfach eintragen

```
argocd-tls-certs-cm
```

data:

git-0.training.lab: |

-----BEGIN CERTIFICATE-----

```
MIIDRzCCAi+gAwIBAgIRA04DiGR1GQx8YnDm0+SxJjYwDQYJKoZIhvcNAQELBQAw
PTETMBEGA1UEChMKYjEtY3ZldGVtczEmMCQGA1UEAxMddHJhaW5pbmcubGF1eGlu
dGVybmFsIFJvb3QgQ0EwHhcNMjUwNTA1MDUwMTAwWhcNMjUwNTA1MDUwMTAwWjA9
MRMwEQYDVQQKEwp1MS1zeXN0ZW1zMSYwJAYDVQQDEx10cmFpbmluZy5sYW50
ZXJ1eWwvUm9vdCBDQTCASiWdQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAM0p
JoSj+Xgt6mydNUa8DiiT9puWyTm17wL4jRqZszbg/dcXtnnSeBASCX3JGVkdSu5e
ADs4AMND6CgxtY+Uh4Yt0IHqZLB2pTKNqRvD+Xu87VRvUUPkHaZBNwif3gMyCTZ5
b0FgQ/qH/BAJiD0NmoGs/anfKs42C27JRzYBqTYzmmngZl4uae/uxEqs/UIIRlqQ
hlxMd9AF/gllzCP00+0Rug5camt/lT84tzL3mkKs2CMrpa7ZkgpWE0xYXa9Knd1a
NZ0NgThq/JjM3U19f1NsQbwIy+aM0Algql1AatJzaKE0+xsKE73mpLyDuttrSICt
9tjL/0uT1GXDX0uKHwcCAwEAAaNCMEAwDgYDVDR0PAQH/BAQDAgKkMA8GA1UdEwEB
/wQFMAMBAf8wHQYDVDR0BBYEFMdfmTamgSMlMCiICSY4QpLkA8doMA0GCSqGSIb3
DQEBcwUAA4IBAQCsz+IZc/fRR04mz9uTQCqMTBADgUA3hWLAwVuTQngdl5hZTVcZ
FZZpp0/6aC9IZJ7M6fjkxDyWx/o6diM4HatLJ9l5SR94KrUzLjiYAZdLJhoKMjY1
12ZMT2KLjHtzS+ihGiCYPvo8Ms+Dvrpr0bncrtvsSoW0JoL6ZlzoifffycLfEzC2
4xXnFiggsDl3bQD/u02DlRPJpq0EzhaXnaebN3zBIar48SVa/xcjYGeBCM29BvG
gt0n4ph/PLZQj53N8aIczYDnVyRbWaJoFif+HXWwZigJoIrr8/jeLL0ILuFZ7Vlh
PK1fZFxmPwLJow1iT5A9lSnkquon0IAdnb0o
```

-----END CERTIFICATE-----

Tips und Tricks zum Thema "Sync Options"

Im Applications Objekt kann man mittels

```
kind: Application
metadata:
  name: wiki
  namespace: argocd
finalizers:
  - resources-finalizer.argocd.argoproj.io
```

dafür sorgen das die Ressourcen im Kubernetes auch wirklich entfernt werden und nicht nur das ArgoCD Application Object gelöscht wird.

Wenn man sicher gehen möchte das bestimmte Ressourcen aber nicht von der ArgoCD entfernt werden sollen kann man diesen Ressourcen eine Annotation mit geben.

```
kind: PersistentVolumeClaim
metadata:
  annotations:
    argocd.argoproj.io/sync-options: Delete=false
...
```

Mehr Informationen zu diesem Thema findet man in der [Doku](#).

Application Specification

[ArgoCD Dokumentation](#)

Eine komplette Beschreibung aller Möglichkeiten des Application Objektes von ArgoCD

From:
<https://www.cooltux.net/> - TuxNet DokuWiki

Permanent link:
<https://www.cooltux.net/doku.php?id=it-wiki:kubernetes:argocd>

Last update: **2025/11/09 08:56**

