ArgoCD

Installation Core Variante

stable getting started

Installation Helm Chart Variante

```
# install argocd via helm and customized values1
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+gAwIBAgIRAO4DiGR1GQx8YnDmO+SxJjYwDQYJKoZIhvcNAQELBQAw
    PTETMBEGA1UEChMKYjEtc3lzdGVtczEmMCQGA1UEAxMddHJhaW5pbmcubGFiIElu
    dGVybmFsIFJvb3QgQ0EwHhcNMjUwNTA1MDUwMTAwWhcNMzUwNTAzMDUwMTAwWjA9
    MRMwEQYDVQQKEwpiMS1zeXN0ZW1zMSYwJAYDVQQDEx10cmFpbmluZy5sYWIgSW50
    ZXJuYWwgUm9vdCBDQTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAM0p
    JoSj+Xgt6mydNUa8DiiT9puWyTm17wL4jRqZszbg/dcXtnnSeBASCX3JGVkdSu5e
    ADs4AMND6Cgxty+Uh4Yt0IHqZlB2pTKNqRvD+Xu87VRvUUPkHaZBNwif3gMyCTZ5
    bOFgQ/qH/BAJiDONmoGs/anfKs42C27JRzYBqTYzmmngZl4uae/uxEqs/UIIRlqQ
    hlxMd9AF/gllzCPO0+ORug5camt/lT84tzL3mkKs2CMrpa7ZkgpWE0xYXa9Knd1a
    NZONgThq/JjM3U19f1NsQbwIy+aM0Algql1AatJzaKEO+xsKE73mpLyDuttrSICt
    9tjL/0uT1GXDX0uKHwcCAwEAAaNCMEAwDgYDVR0PAQH/BAQDAgKkMA8GA1UdEwEB
    /wQFMAMBAf8wHQYDVR0OBBYEFMdfmTamgSMlMCiICSY4QpLkA8doMA0GCSqGSIb3
    DQEBCwUAA4IBAQCsZ+IZc/fRRO4mz9uTQCqMTBADgUA3hWLAWVuTQngdl5hZTvCz
    FZZppO/6aC9IZJ7M6fjkxDyWx/o6diM4HatLJ9l5SR94KrUzLjiYAZdLJhoKMjY1
    12ZMT2KLjHtzS+ihGiCYPvo8Ms+Dvrpr0bncrtvsSoW0JoL6Zlzoi/ffycLfEzC2
    4xXnFiggsDl3bQD/u02DIrPJpq0EzhaXnahebN3zBIar48SVa/xcjYGeBCM29BvG
    gtOn4ph/PLZQj53N8aIczYDnVyRbWaJoFif+HXWwZIgJoIrr8/jeLLOILuFZ7Vlh
    PK1fZFxmPWLJow1iT5A9lSnkquonOIAdnb0o
    -----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