# ESTA Tekton Operator

Ziel: Anwender können  ESTA Tekton selbständig via Custom Resource in einem OpenShift Namespace installieren.

* [Erforderliche Ressourcen](#ESTATektonOperator-ErforderlicheRessour)
  + [Namespace Setup](#ESTATektonOperator-NamespaceSetup)
  + [ESTA Tekton Deployment](#ESTATektonOperator-ESTATektonDeployment)
* [Setup Prozess](#ESTATektonOperator-SetupProzess)
  + [OAuth Setup (Azure AD App Registration)](#ESTATektonOperator-OAuthSetup(AzureADAp)
  + [Credential Exchange](#ESTATektonOperator-CredentialExchange)
    - [Bitbucket User](#ESTATektonOperator-BitbucketUser)
    - [Artifactory User](#ESTATektonOperator-ArtifactoryUser)
    - [AWS S3 Access Key](#ESTATektonOperator-AWSS3AccessKey)
    - [Azure Blob Storage](#ESTATektonOperator-AzureBlobStorage)
* [Technologie](#ESTATektonOperator-Technologie)
  + [Operator Credentials](#ESTATektonOperator-OperatorCredentials)
  + [Operator Config Options](#ESTATektonOperator-OperatorConfigOption)
  + [Entwurf Custom Resource](#ESTATektonOperator-EntwurfCustomResourc)
  + [TODOs / Refactorings](#ESTATektonOperator-TODOs/Refactorings)

# Erforderliche Ressourcen

## Namespace Setup

Quellen: [ocp-argocd/namespace-configuration-operator](https://code.sbb.ch/projects/KD_PAAS/repos/ocp-argocd/browse/projects/default/applications/namespace-configuration-operator/values.yaml#49), [esta-tekton-pipeline-config](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-config/browse/charts/esta-tekton-pipeline-config/templates)

* LimitRange
  + sbb-esta-tekton-pipeline-defaults
* ResourceQuota
  + esta-tekton-pipeline-quota
* ServiceAccount
  + esta-tekton-pipeline-sa
  + esta-tekton-pipeline-root-sa :question:
  + esta-tekton-controller-sa
* RoleBinding
  + esta-tekton-sa-scc-anyuid
  + esta-tekton-pipeline-sa
  + esta-tekton-pipeline-root-sa :question:
  + esta-tekton-controller-sa
* Secret
  + azure-ad-client-credentials
  + esta-tekton-basicauth-docker
  + esta-tekton-basicauth-git
  + esta-tekton-oauth-client-ids
  + esta-tekton-sonar-api-key
  + esta-owasp-db-secret
  + esta-tekton-pipeline-env
  + argocd-env-secret
  + smtp-server-secret

Default LimitRanges (sbb-default-limits) und ResourceQuota (default-mini-project-quota) müssen vom Operator resp. vom Helm Chart via Hook gelöscht werden.

## ESTA Tekton Deployment

Quellen: [esta-tekton-pipeline-controller](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-controller/browse/charts/esta-tekton-pipeline-controller/templates), [esta-tekton-pipeline-ui](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-ui/browse/charts/esta-tekton-pipeline-ui/templates)

* Deployment
  + esta-tekton-pipeline-controller
  + esta-tekton-pipeline-ui
* Service
  + esta-tekton-pipeline-controller
  + esta-tekton-pipeline-ui
* Route
  + esta-tekton-pipeline-controller-sbb-cloud
  + esta-tekton-pipeline-ui-sbb-cloud
* ConfigMap
  + esta-tekton-pipeline-controller-application-yaml
  + esta-tekton-pipeline-controller-notification-text-yaml
  + esta-tekton-bitbucket-config
  + sbb-certificates
* PersistentVolumeClaim
  + esta-tekton-pipeline-controller-workspace

# Setup Prozess

1. Anlegen der Namespace Config Resources (LimitRange, ResourceQuota)
2. Initialisierung Umsysteme
   1. OAuth Setup
   2. Bitbucket
      1. User erstellen :question:
      2. Access Token registrieren :question:
      3. Secret esta-tekton-basicauth-git anlegen
   3. Artifactory
      1. User erstellen :question:
      2. Access Token registrieren :question:
      3. Secret esta-tekton-basicauth-docker anlegen
   4. AWS IAM User & Policy
      1. Terraform apply
      2. Secret esta-tekton-aws-cache-credentials anlegen
   5. Azure Blob Storage Setup
      1. Vorgehen noch unbekannt
3. ESTA Tekton Deployment
   1. [esta-tekton-pipeline-config](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-config/browse) Helm Chart installieren

## OAuth Setup (Azure AD App Registration)

Kann wie bisher über einen Taskrun ausgeführt werden.  
Es sollen aber pro Namespace individuelle Azure AD API Credentials generiert werden, wie dies im Konzept [Azure AD App Registration Prozess](file:///C:\display\CLEW\Azure+AD+App+Registration+Prozess) beschrieben wird.

## Credential Exchange

### Bitbucket User

Falls via API möglich: ein neuer Benutzer wird angelegt (falls nicht existent)  
Falls via API möglich: der Operator generiert via API einen Access Token für den angegebenen Benutzer und speichert diesen im Secret esta-tekton-basicauth-git.  
Falls via API möglich: Der Operator generiert ebenfalls einen SSH Key und speichert via API den Pubkey in Bitbucket. Der Private Key wird in einem Secret esta-tekton-ssh-git gespeichert.

Vault Integration

Eine allfällige Anbindung von Hashicorp Vault ist zu prüfen. Im besten Fall kann für den Tekton Namespace lediglich ein Vault Key konfiguriert werden.

### Artifactory User

Falls via API möglich: ein neuer Benutzer wird angelegt (falls nicht existent)  
Falls via API möglich: Der Operator generiert via API einen Access Token für den angegebenen Benutzer und speichert diesen im Secret esta-tekton-basicauth-docker.

Vault Integration

Eine allfällige Anbindung von Hashicorp Vault ist zu prüfen. Im besten Fall kann für den Tekton Namespace lediglich ein Vault Key konfiguriert werden.

### AWS S3 Access Key

Für den Cache Zugriff muss im gesharten AWS Account ein IAM User & Policy angelegt werden. Das kann wie bisher mittels Terraform in einem Taskrun im Operator Namespace ausgeführt werden. Als Alternative ist die direkte Ausführung von Terraform im Operator (Pod) zu prüfen.

### Azure Blob Storage

Der Prozess ist noch nicht bekannt

# Technologie

Der Operator soll primär über ein Helm Chart die oben aufgeführten Ressourcen in den Zielnamespace bringen und damit den Namespace Configuration Operator sowie das bestehende Helm Chart ([esta-tekton-pipeline-config](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-config/browse)) ablösen resp. vereinen.

Controller und UI werden als Dependencies über die individuellen Charts der einzelnen Komponenten in den konfigurierten Version herunterladen und im Namespace installieren.

Auch die App Registration und die Tasks für die Credential Exchange können als Job oder TaskRun im Helm Chart definiert und im Operator Namespace verarbeitet werden, ähnlich wie das bisher mit ArgoCD gemacht wird.

Ein [Helm-based Operator](https://sdk.operatorframework.io/docs/building-operators/helm/tutorial/) kann hier als technologische Grundlage dienen. Ein POC gemäss dem Quickstart Guide mit [diesem Helm Chart](https://code.sbb.ch/users/u233658/repos/esta-tekton-namespace/browse) wurde erfolgreich durchgeführt.

## Operator Credentials

(Als Secrets im Operator Namespace hinterlegt)

* AWS Key & Secret zum Anlegen der IAM User & Policies
* Azure AD API Client-id und Secret
* Defaults für esta-tekton-sonar-api-key, esta-owasp-db-secret und smtp-server-secret  
  (→ über [Watches/valueOverrides](https://sdk.operatorframework.io/docs/building-operators/helm/reference/watches/) injected)

## Operator Config Options

* pipelineCacheBackend: (AWS\_S3 oder AZURE\_BLOB)
* azureAdApiHost: [azure-ad.api.sbb.ch](http://azure-ad.api.sbb.ch)
* awsTerraform:
  + backendBucket: terraform-esta-tekton-infrastructure-prod
  + s3BucketName: esta-tekton-pipeline-cache-prod
* Default Values für [esta-tekton-pipeline-config](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-config/browse/charts/esta-tekton-pipeline-config/values.yaml) Helm Chart

## Entwurf Custom Resource

**esta-tekton-cr.yaml** Quelle erweitern

apiVersion: esta.sbb.ch/v1beta1

kind: EstaTektonBuildSystem

metadata:

name: my-own-tekton

namespace: esta-tekton-dev

spec:

releaseVersion: 0.14.12

appRegistration:

owners:

- userPrincipalName: replace-with-REQUESTER

allowedADGroups:

# DG\_RBT\_ESTA

- a51563af-953f-485b-8cf0-b37bc8d7c5fe

# add your groups here

   apiAccessClientIds: []

bitbucket:

host: code.sbb.ch

username: esta-build-clew

artifactory:

host: bin.sbb.ch

username: esta-build-clew

storageQuota:

default: 1Ti

overrides:

ki-tekton: 2Ti

  # values aus esta-tekton-pipeline-config Helm Chart

 esta-tekton-pipeline-controller:

appVersion: 2.15.1

pipelineTemplateVersion: 0.11.5

   defaultMavenGoals: clean install

   additionalTemplateRepos:

     - https://code.sbb.ch/scm/kd\_esta\_test/esta-tekton-pipeline-custom-templates.git?at=refs/heads/master

   splunkIndex: sbb\_esta-tekton\_internal\_prod\_events

 esta-tekton-pipeline-ui:

appVersion: 2.10.0

  splunkIndex: sbb\_esta-tekton\_internal\_prod\_events

Subchart Versioning

Bislang unklar ist, wie die Versionen der Subcharts (esta-tekton-pipeline-controller und esta-tekton-pipeline-ui) als Parameter über die Custom Resource definiert werden kann.

## TODOs / Refactorings

* "stage" Value in esta-tekton-pipeline-\* Charts müssen mit .Release.Namespace ersetzt werden.
* Die konkreten Permissions für den Service Account des Operators resp. der TaskRuns im Operator Namespace müssen definiert werden (muss Secrets im Zielnamespace anlegen/überschreiben können)
* Der Release-Prozess muss neu definiert werden. Über den (einfachen) Helm Operator können nur statische Werte für die App Versionen hinterlegt werden.  
  Eine Option wäre, ein Helm Chart analog zum [CLEW ArgoCD](https://code.sbb.ch/projects/KD_ESTA/repos/helm-argocd/browse) Chart anzubieten, welches primär die Custom Resource mit vor-definierten Values für Controller und UI schreibt.