# Esta Demo Operator - Workshop

## Content Table

* [Content Table](#EstaDemoOperatorWorkshop-ContentTable)
* [Agenda](#EstaDemoOperatorWorkshop-Agenda)
* [What is an Operator for?](#EstaDemoOperatorWorkshop-WhatisanOperat)
* [Operator SDK](#EstaDemoOperatorWorkshop-OperatorSDK)
* [Terminology CRD and CR](#EstaDemoOperatorWorkshop-TerminologyCRD)
* [How does the esta-demo-operator-quarkus setup looks like](#EstaDemoOperatorWorkshop-Howdoestheesta)
* [Breakouts to create your own Operator on base of the example Operator](#EstaDemoOperatorWorkshop-Breakoutstocre)
* [Example esta-demo-operator-quarkus](#EstaDemoOperatorWorkshop-Exampleesta-de)
* [Vorbereitung](#EstaDemoOperatorWorkshop-Vorbereitung)
* [Erkenntnisse zu Fabric 8, Operator SDK Bootproblemen und SBB Windows mit EAIO](#EstaDemoOperatorWorkshop-Erkenntnissezu)

## Agenda

In the Esta-Demo-Operator Workshop we will do following

1. What are Kubernetes Operators (15min -Jose)
2. Operator-SDK - Java Operator generieren (5 min - Manu)
3. Golang vs Java Operators (5 min)
4. Esta-Demo-Operator-Quarkus explained (15min - Manu~~)~~
5. ~~Generate your own Java Operator with the Operator-SDK and Quarkus (5min)~~
6. Use Case Ablösung estaCloudPipeline.json vs Operator → Unklarheiten? (10min)
7. Team Breakouts (Kafka, AzureAd, Apim) - Create and build Operator on base of the sample Operator (1h 15min)

## What is an Operator for?

The Operator Pattern in Openshift is a kind of Observer Pattern where an application (Container in a Pod) listens for certain resource changes in all namespaces within a cluster. If resource changes, the operator executes it's logic and updates the resource. This can be useful to setup, modify or delete resources within the Openshift Cluster or even outside. In the esta case we will have our first use cases in the deployment of AzureAd AppRegistrations, APIM Registrations or Kafka Configurations.

## Operator SDK

Documentation:

<https://javaoperatorsdk.io/docs/intro-operators>

## Terminology CRD and CR

We have 2 terms we hear often in the operator environment:

CRD: Custom Resource Definition defines how a Custom Resource should look like. What input fields and what status fields the resource can have.

**CRD**

# Generated by Fabric8 CRDGenerator, manual edits might get overwritten!

apiVersion: apiextensions.k8s.io/v1

kind: CustomResourceDefinition

metadata:

name: azureaddemoes.esta.sbb.ch

spec:

group: esta.sbb.ch

names:

kind: AzureAdDemo

plural: azureaddemoes

singular: azureaddemo

scope: Namespaced

versions:

- name: v1

schema:

openAPIV3Schema:

properties:

spec:

properties:

appRegistrationJson:

type: string

appRegSecretRef:

type: string

appRegistrationYaml:

type: string

type: object

status:

properties:

jsonResponse:

type: string

state:

enum:

- INVALID\_YAML

- APPLIED

- ERROR

- CHANGES\_NOT\_ALLOWED

type: string

message:

type: string

type: object

type: object

served: true

storage: true

subresources:

status: {}

CR: Custom Resource

**CRD**

apiVersion: esta.sbb.ch/v1

kind: AzureAdDemo

metadata:

name: montagstest

spec:

appRegSecretRef: appregistration-demo-secret

appRegistrationYaml: >-

apiVersion: azure-ad-api/v1

appRegistration:

- displayName: SBB IAM Test Application for testing Openshift Operators

id: 91b1264a-2f4c-11ed-a261-0242ac120002

megaId: 5A135B9654C78175

redirectUris:

- uri: http://localhost:8080

type: WEB

owners:

- userPrincipalName: christian.egli3@sbb.ch

- userPrincipalName: manuel.wallrapp@sbb.ch

passwords:

- displayName: pw1

endDateTime: 2022-10-24T00:00:00Z

roles:

- id: b848539c-2f4b-11ed-a261-0242ac120002

assignedToADGroups:

- 8e180531-b0b5-4cb7-8825-b1f98d9a79f5

- 6a1d58ca-06bf-4503-9825-6f2934aed896

CR after processed by the operator:

**CR after processed by the operator**

apiVersion: esta.sbb.ch/v1

kind: AzureAdDemo

metadata:

annotations:

kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"esta.sbb.ch/v1","kind":"AzureAdDemo","metadata":{"annotations":{},"name":"montagstest","namespace":"esta-operator-demo"},"spec":{"appRegSecretRef":"appregistration-demo-secret","appRegistrationYaml":"apiVersion: azure-ad-api/v1\nappRegistrations:\n - displayName: SBB IAM Test Application for testing Openshift Operators\n id: 91b1264a-2f4c-11ed-a261-0242ac120002\n megaId: 5A135B9654C78175\n redirectUris:\n - uri: http://localhost:8080\n type: WEB\n owners:\n - userPrincipalName: christian.egli3@sbb.ch\n - userPrincipalName: manuel.wallrapp@sbb.ch\n passwords:\n - displayName: pw1\n endDateTime: 2022-10-24T00:00:00Z\n roles:\n - id: b848539c-2f4b-11ed-a261-0242ac120002\n assignedToADGroups:\n - 8e180531-b0b5-4cb7-8825-b1f98d9a79f5\n - 6a1d58ca-06bf-4503-9825-6f2934aed896"}}

creationTimestamp: "2022-09-12T09:27:26Z"

generation: 4

name: montagstest

namespace: esta-operator-demo

resourceVersion: "605054260"

uid: f4c6a6f6-fa87-4aa3-830c-71ab854e8cc5

spec:

appRegSecretRef: appregistration-demo-secret

appRegistrationYaml: |-

apiVersion: azure-ad-api/v1

appRegistrations:

- displayName: SBB IAM Test Application for testing Openshift Operators

id: 91b1264a-2f4c-11ed-a261-0242ac120002

megaId: 5A135B9654C78175

redirectUris:

- uri: http://localhost:8080

type: WEB

owners:

- userPrincipalName: christian.egli3@sbb.ch

- userPrincipalName: manuel.wallrapp@sbb.ch

passwords:

- displayName: pw1

endDateTime: 2022-10-24T00:00:00Z

roles:

- id: b848539c-2f4b-11ed-a261-0242ac120002

assignedToADGroups:

- 8e180531-b0b5-4cb7-8825-b1f98d9a79f5

- 6a1d58ca-06bf-4503-9825-6f2934aed896

status:

jsonResponse: '[{"appDisplayName":"SBB IAM Test Application for testing Openshift

Operators","clientId":"3095b924-4738-4ed4-9f5e-3ea1560bd002","createdClientSecrets":[],"fileBasedRegistrationId":"91b1264a-2f4c-11ed-a261-0242ac120002","registrationErrors":{"isCreated":true,"errors":[]}}]'

message: ""

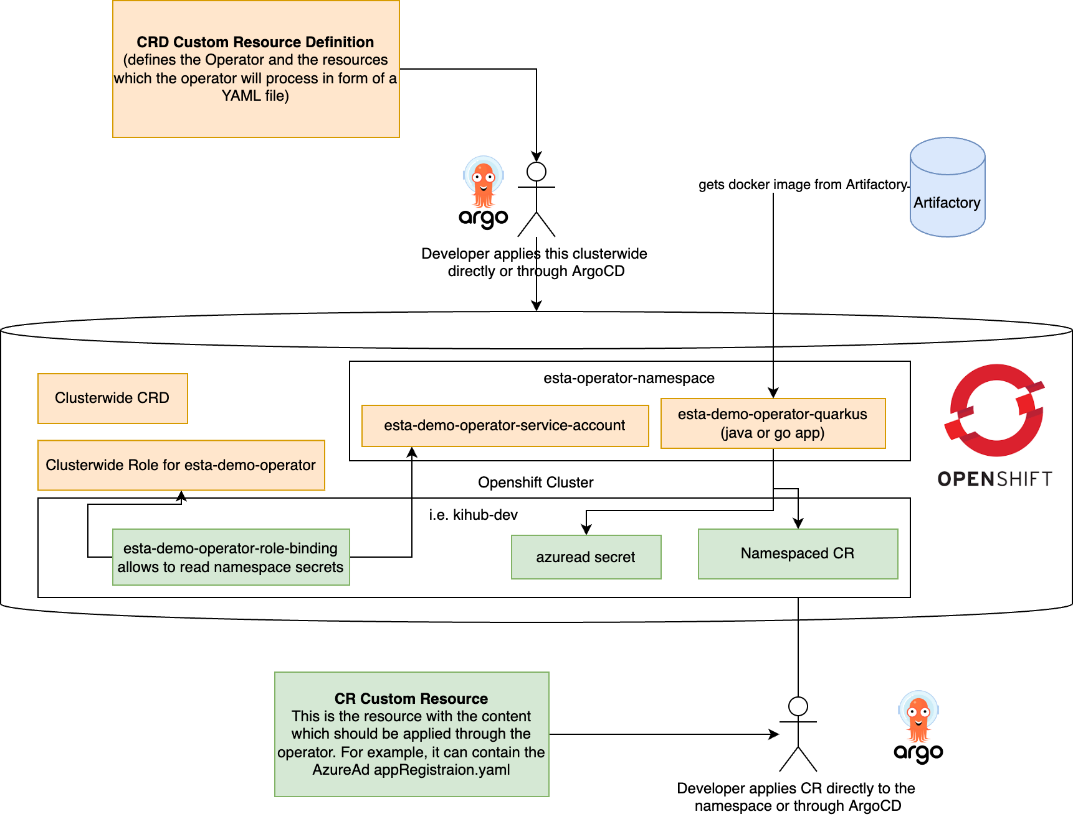
state: APPLIED

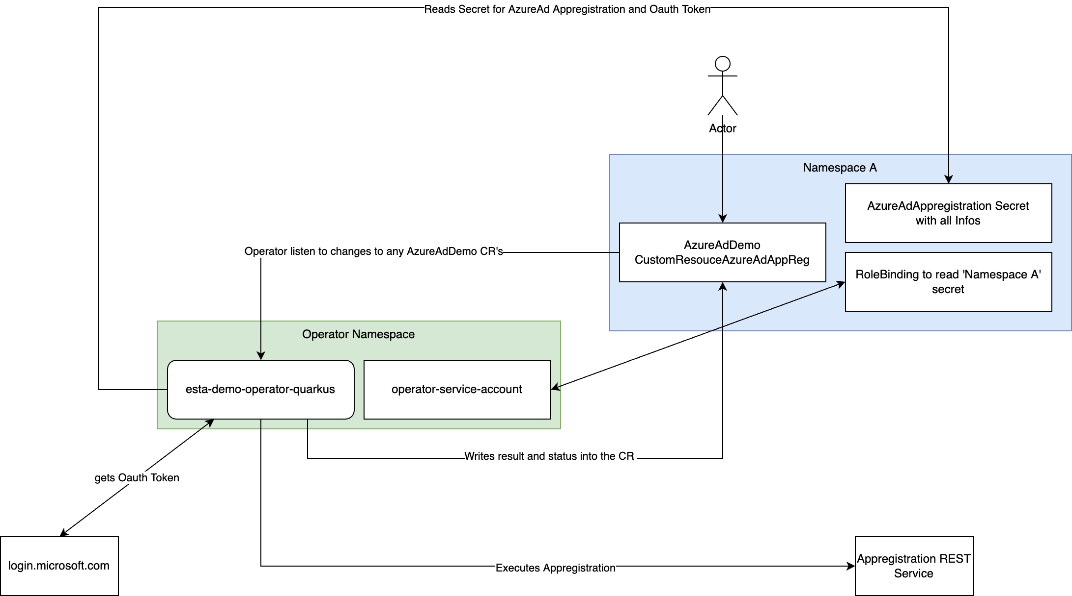
## How does the esta-demo-operator-quarkus setup looks like

An Operator consists mainly 3 core components:

1. Custom Resource Definition in form of a YAML which defines what resources the Operator is observing
2. Operator itself which is a Docker Conatiner running in a Pod somewhere in the cluster
3. Custom Resource is the resource the Operator should process. It is usually also a YAML file which get's applied to a namespace.

The whole picture is explained a little better in this diagram.





In the end, an Operator is a simple app which listens to Custom Resource changes in one or all namespaces, processes this changes, writes a status into the Custom Resource.

## Breakouts to create your own Operator on base of the example Operator

1. Fork the Esta-Demo-Operator to esta-azurad-operator, esta-kafka-operator, esta-apim-operator
2. Adapt the sample Operator incl. Tests to your needs
3. Build the operator locally so we have the CRD's and Deploymentconfigs
4. Push your changes, build your first Operator Release
5. Add your yaml files (Deployment, Service, Route, CRD) to the corresponding ocp-argocd  project (Instructions from us)
6. We deploy our Operator through ArgoCd
7. Create a CustomResource and see how the Operator reacts

## Example esta-demo-operator-quarkus

Sourcecode: <https://code.sbb.ch/projects/KD_ESTA_TEST/repos/esta-demo-operator-quarkus/browse>

Deployment: <https://code.sbb.ch/projects/OM_PAAS/repos/ocp-argocd/browse/projects/esta/applications/esta-demo-operator-quarkus?at=refs%2Fheads%2Ffeature%2FESTA-5242>

## Vorbereitung

* Präsentation
  + Operator Slides (Jose) :tick:
  + Quarkus Example Explained (Manu)
* personalized namespaces (Jose legt an mit Berechtigungen für User)
  + uxyz-operator-workshop (u...) :tick:
    - alle NS wurden erstellt und die entsprechenden Users berechtigt  
      <https://console-openshift-console.apps.clew01d.sbb-aws-dev.net/k8s/cluster/projects?project-name=operator-workshop>
  + runtime-operator-workshop :tick:
* Deployment, Service, SeviceAccount, ClusterRole → ocp-argocd (wird vom Operator mvn build generiert)
  + azureAd
  + kafka
  + apim
* Charts vorbereiten für alle 3 Operators (Jose) :tick:
* Demo Operator forken und für alle 3 Teams vorbereiten.
  + Build muss laufen
  + Docker Image vorhanden

## Erkenntnisse zu Fabric 8, Operator SDK Bootproblemen und SBB Windows mit EAIO

[Egli Christian (IT-PTR-CEN2-SL2)](file:///C:\display\~u210150) [Benutzer-9af42](file:///C:\display\~2c948083803fc438018040ffd3a80000) [Jeanneret Julien (IT-PTR-CEN1-BDE3)](file:///C:\display\~u231154) [Grüner Michael (IT-PTR-CEN2-BDE16)](file:///C:\display\~u234142)  und [Wallrapp Manuel (IT-PTR-EXT-EXT2 - Extern)](file:///C:\display\~ue64007)

Wir haben gestern ziemlich viel Zeit investiert Fabric8 bzw. den Java Demo Operator auf Windows zum laufen zu bringen. Schlussendlich gab es 3 Ursachen für die Verbindungs- / Bootstrap-Probleme von Fabric8 bzw. dem Operator SDK.

1. Intellij wird bei der EAIO Installation über **ein bat File** gestartet, welches eine NO\_PROXY Variable setzt. In dieser Variable war der Host **sbb-aws-dev.net** nicht gelistet, welches dann die Verbindung aus dem internen SBB Netz nicht ermöglichte.
2. Als wir aber auch keine Verbindung über SBB Free aufbauen konnten, haben wir in der Library von Fabric8 festegestellt, dass fabric8 das .kube/config file nicht nur im $HOME sucht sondern auch im $HOME\_DIR welches unter Windows auf das Laufwerk H: zeigt. Das heisst auch dort muss die .kube/config liegen. Wurde beides richtig gesetzt, startet Fabric8 und der Demo Operator einwandfrei.
3. Das automatische generieren und applien muss im application.yaml deaktiviert werden (untersten 2 Zeilen):
4. quarkus:
5. container-image:
6. build: false
7. name: azuread-operator-operator
8. operator-sdk:
9. bundle:
10. enabled: false
11. crd:
12. apply: false

generate: false