Kubernetes native development workflows

Belastingdienst

```
[peter@bld:~/openshift]$ oc new-app https://my-git-repo.nl/CollectTaxRepo
```

Motivation

SKAFFOLD

QUARKUS

Kustomize.io

OPENSHIFT

OPEN CONTAINER INITIATIVE

JKube

Skupper

argo

TEKTON

skopeo

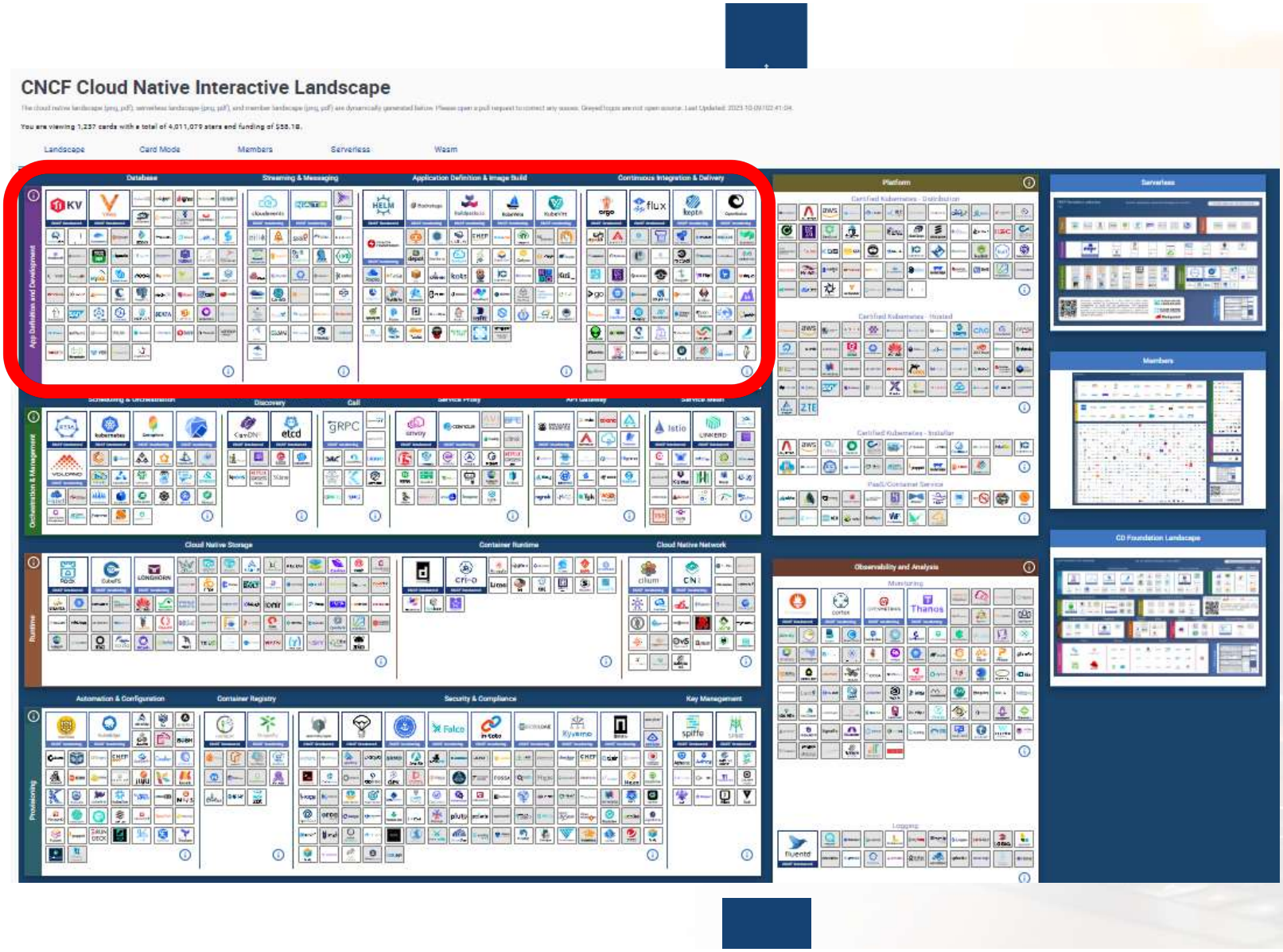
Jib
Containerize your Java application.

docker

buildah

odo

dekorate


CNCF Cloud Native Interactive Landscape
You are viewing 1,237 cards with a total of 4,011,079 stars and funding of $58.1B.
Landscape
Card Mode
Members
Serverless
Wasm
Database
Streaming & Messaging
Application Definition & Image Build
Continuous Integration & Delivery
App Definition and Development
Orchestration & Management
Runtime
Provisioning
Platform
Serverless
Members
CD Foundation Landscape
Observability and Analysis
Cloud Native Storage
Container Runtime
Cloud Native Network
Automation & Configuration
Container Registry
Security & Compliance
Key Management


MacBook Pro

Belastingdienst

Peter Keereweer

Java Developer at Tax Department

pj.keereweer@belastingdienst.nl

Belastingdienst

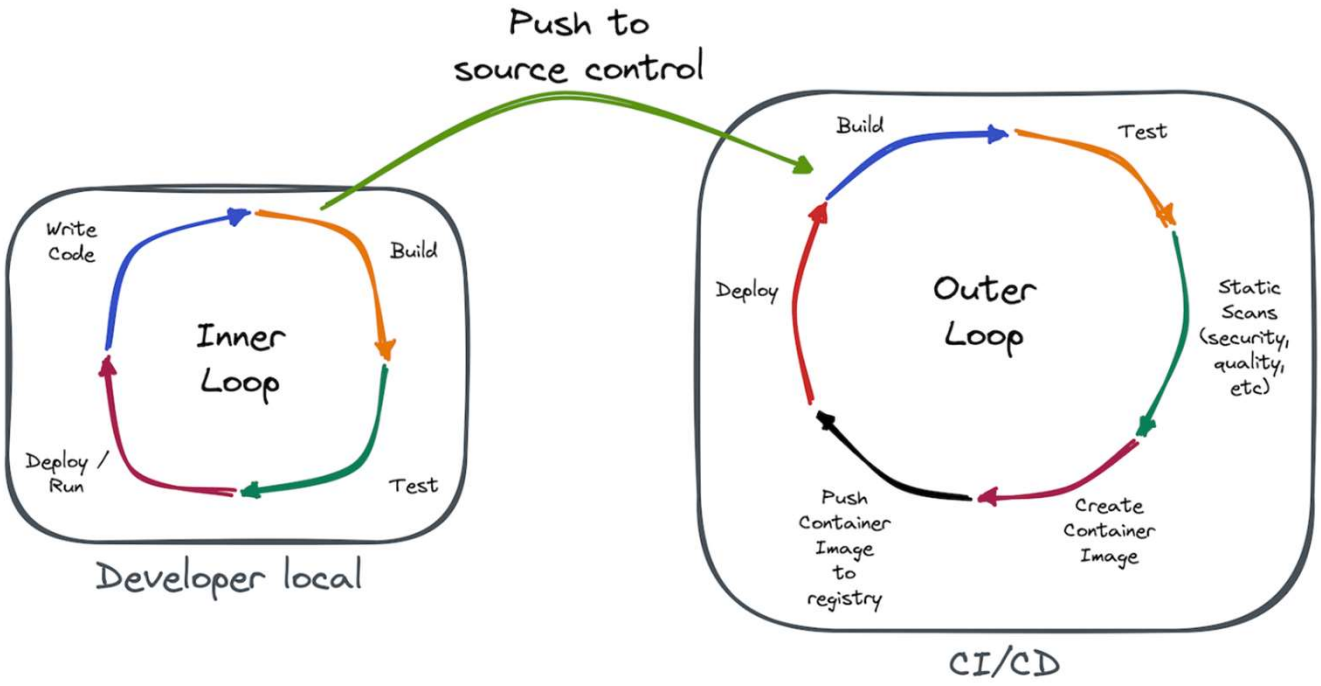
- Development workflow & Challenges
- Container creation & Orchestration
- OpenShift workflow
- Remote workflow
- Remocal workflow
- Quarkus

Belastingdienst

Development workflow
&
Challenges

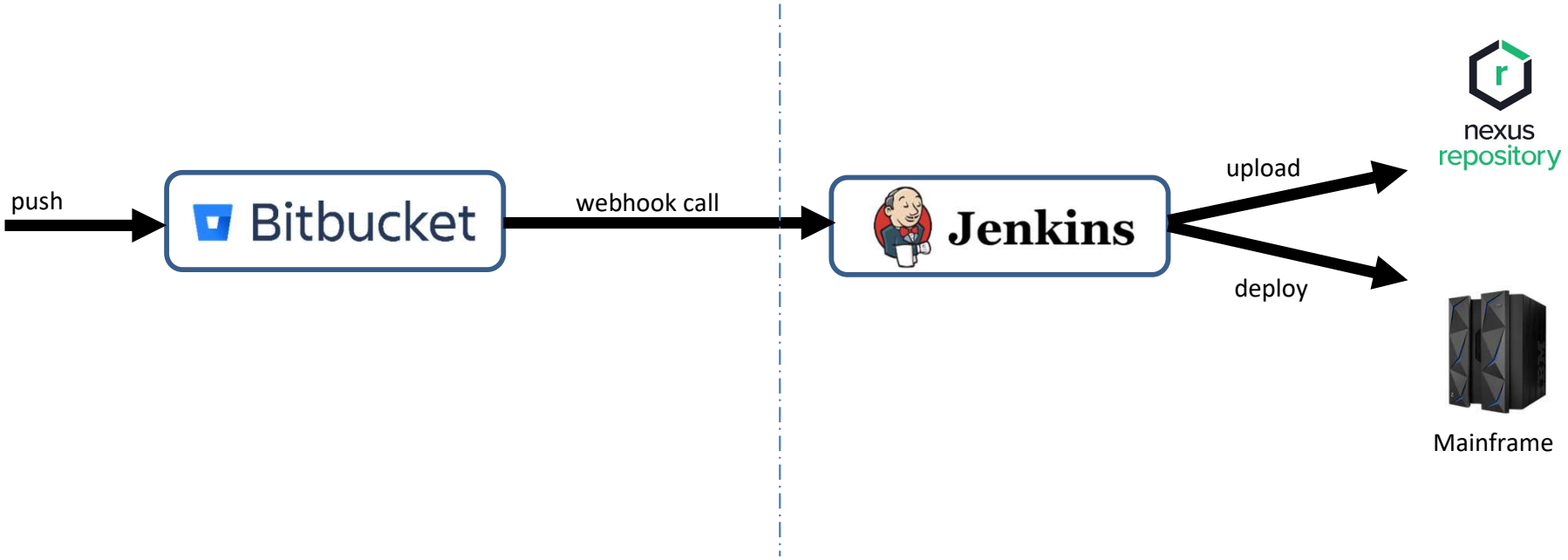
Development workflow

Belastingdienst

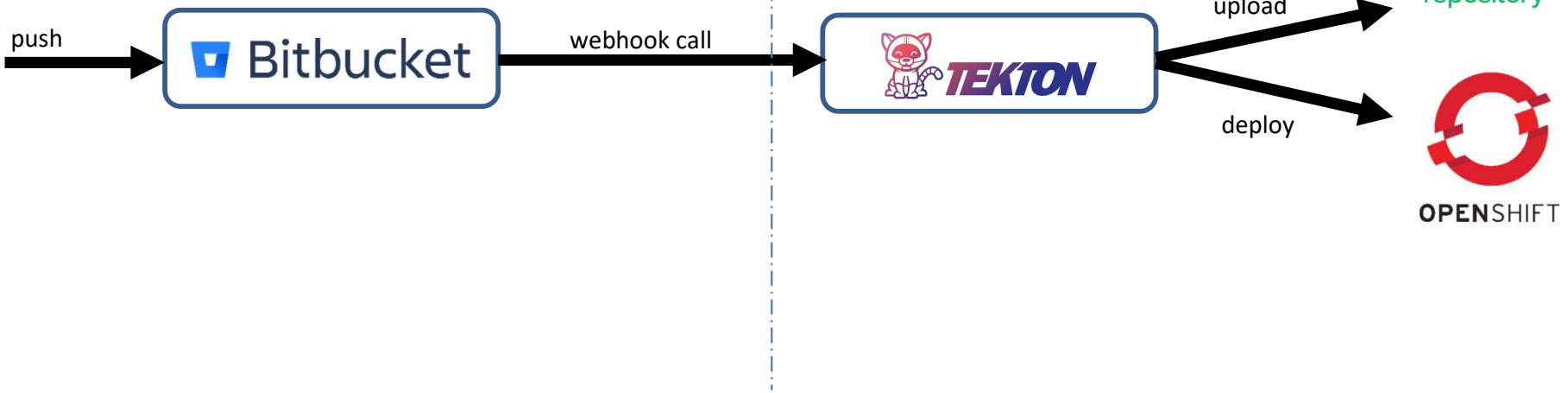


Source

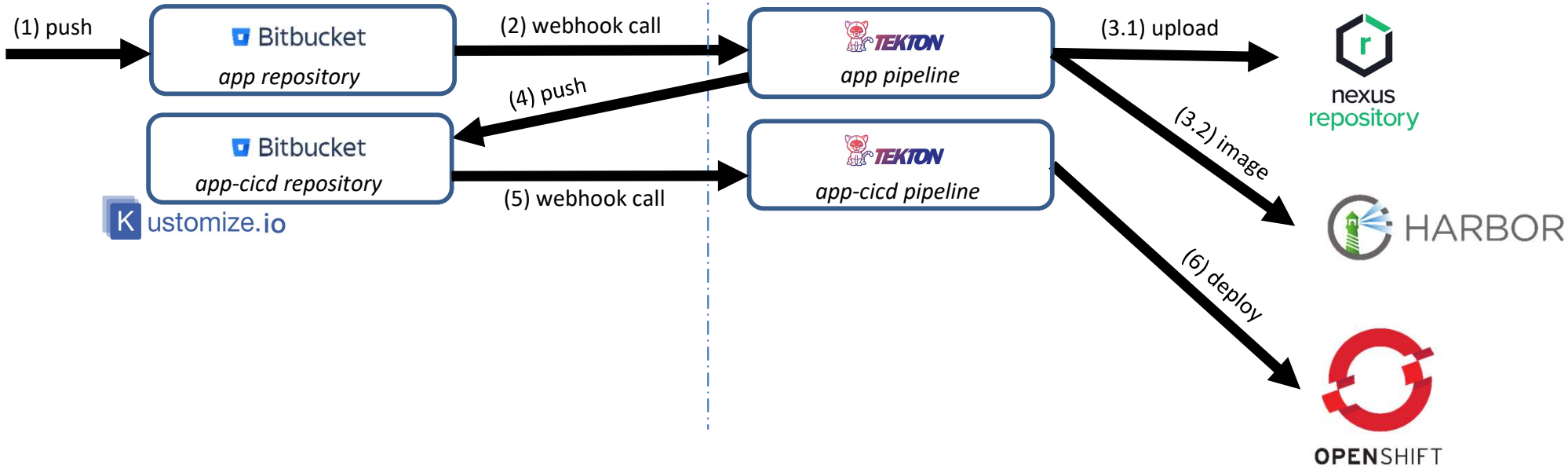
Development workflow



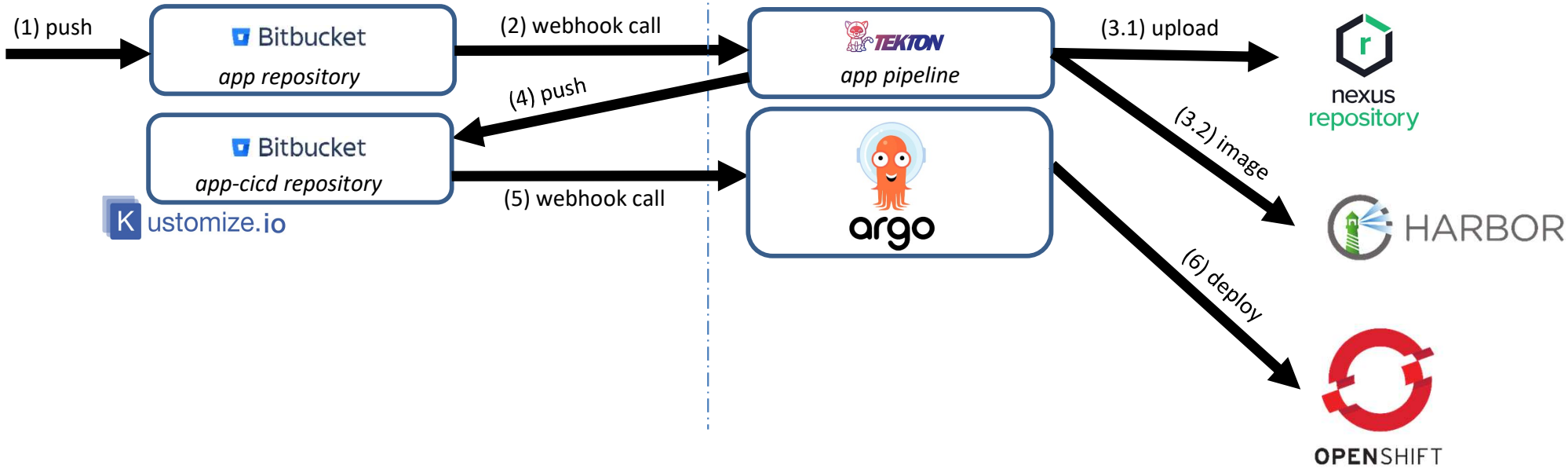
Development workflow



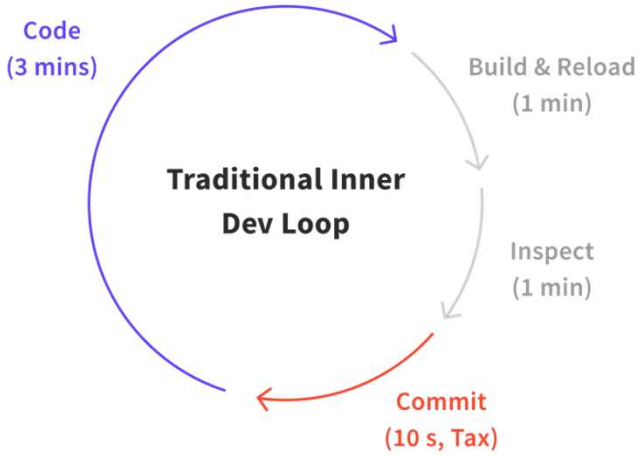
Development workflow

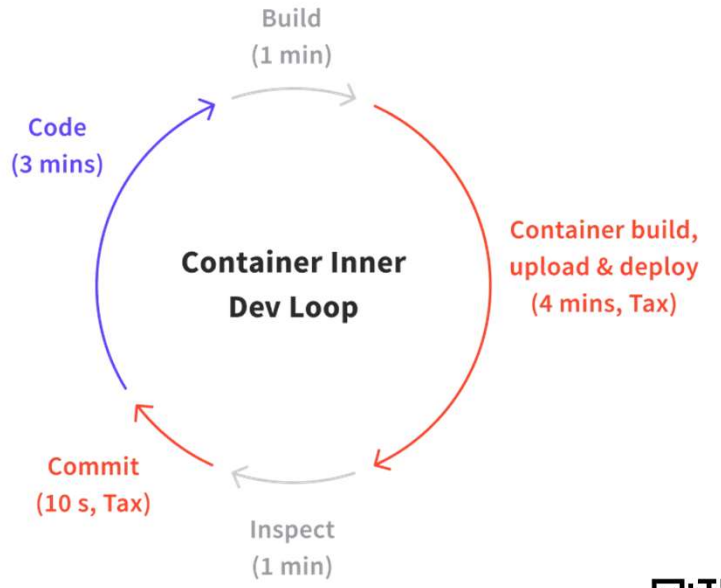


Development workflow



Challenges





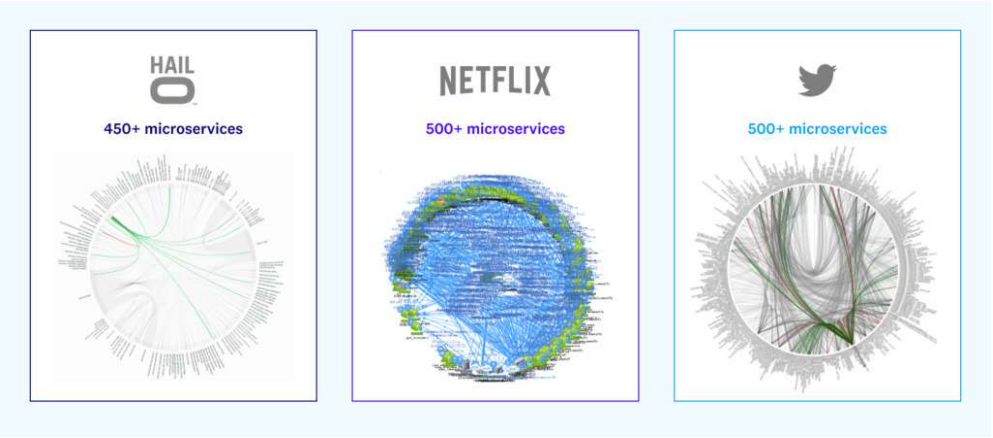
Source

Challenges

Belastingdienst

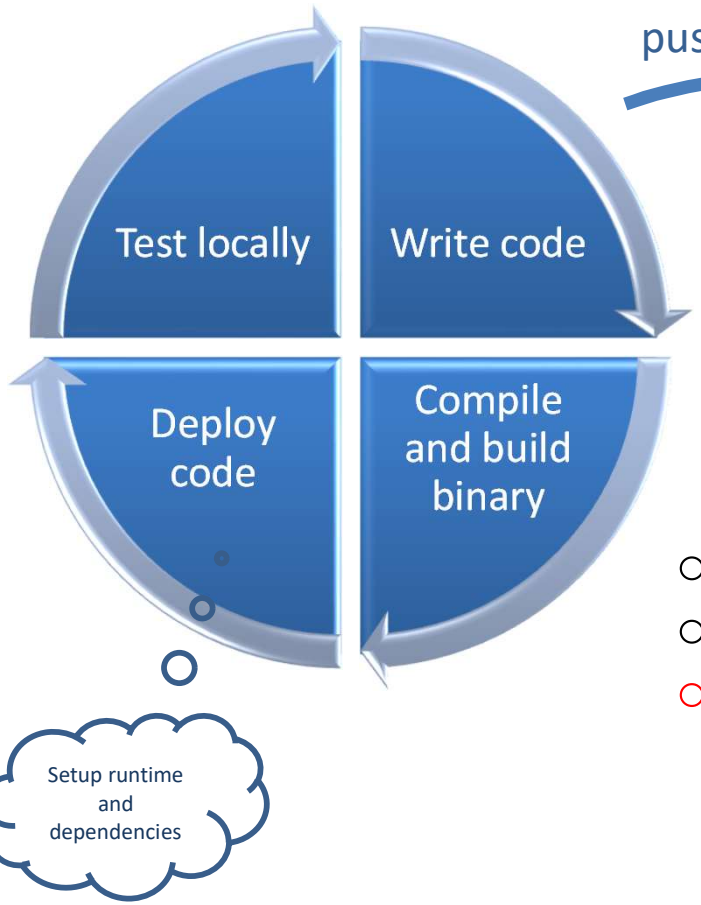
Challenges

- How to run a microservice locally that communicates with other microservices?
- Run all microservices locally?



Source

Belastingdienst

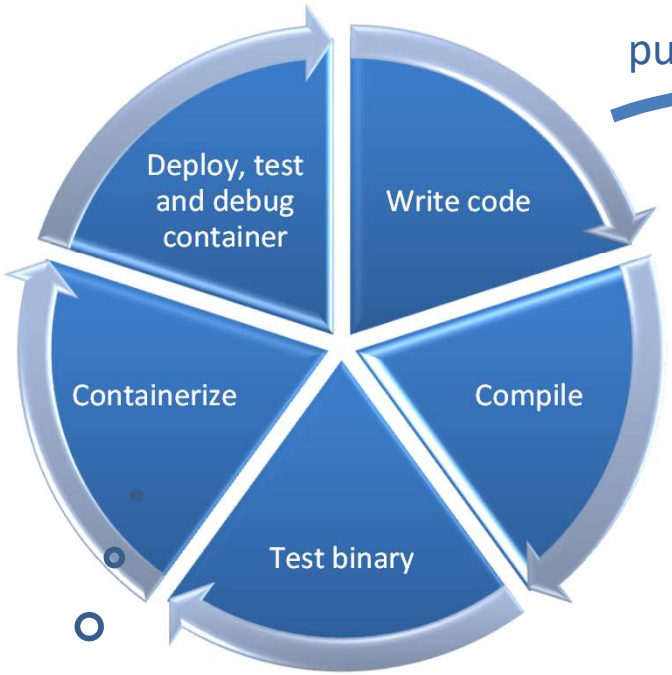


push to source control

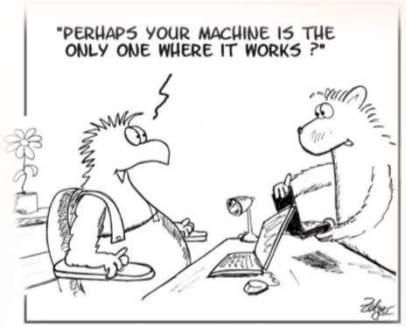
```
[user@jfall:~/demo]$ git commit –m ‘works on my machine’
```

- No containers in development
- Containers created in outer loop
- Development doesn’t match production

Belastingdienst



push to source control

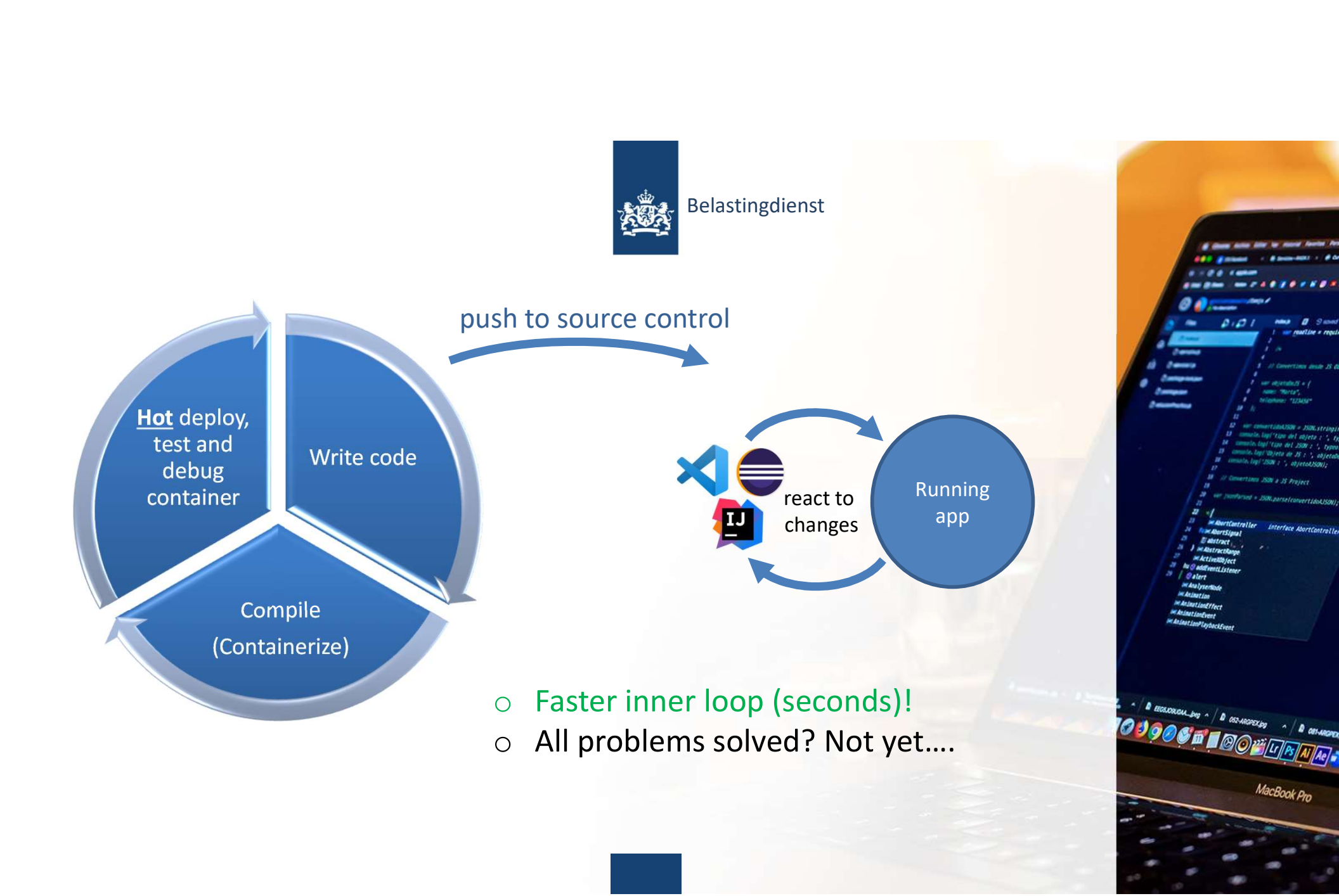


It works on my machine

- Use containers in development
- Slow inner loop (minutes)!
- Still doesn't match production

Create /edit Dockerfile

Source


Belastingdienst
push to source control
Hot deploy, test and debug container
Write code
Compile (Containerize)
react to changes
Running app
Faster inner loop (seconds)!
All problems solved? Not yet….

Belastingdienst

Container creation
&
Orchestration

Container creation

podman

Buildpacks.io

Jib
Containerize your Java application.

docker

buildah

source-to-image (s2i)

Container creation

- OCI compliant containers
  - Open Container Initiative (OCI)

- Build a container locally
  - Dockerfile / Containerfile

- Run container locally

```
[user@jfall:~/demo]$ podman run ubi8/ubi:8.3 echo 'Hello world!'
Hello world!
```

Container creation

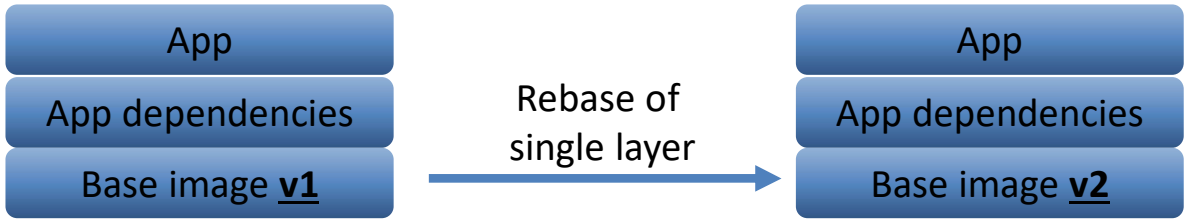
- Buildpacks
  - Build a container with *pack*
  - Uses *builder container*
  - No JDK, Maven or something else needed

Buildpacks.io

```
[user@jfall:~/demo]$ pack build myapp --builder paketobuildpacks/builder:base
```

Container creation

- *Rebase* with Buildpacks



Source

Container creation

- Source-to-image (s2i)
  - Uses *builder image*
- Container creation process runs on cluster
- *BuildConfig*
  - OpenShift resource

```yaml
apiVersion: build.openshift.io/v1
kind: BuildConfig
metadata:
  name: maven-webapp-app
spec:
  source:   # input parameters of source
  strategy: # build strategy to use
  output:   # registry details
  triggers: # list of triggers
```

Container creation

- BuildConfig vs Tekton

  - BuildConfig ***specific*** for OpenShift
  - ***Quickly*** build container on OpenShift with BuildConfig
  - Build more ***complex*** pipelines with Tekton
  - ***Everything*** what can be done with BuildConfig can be accomplished as well ***with Tekton***

Container orchestration

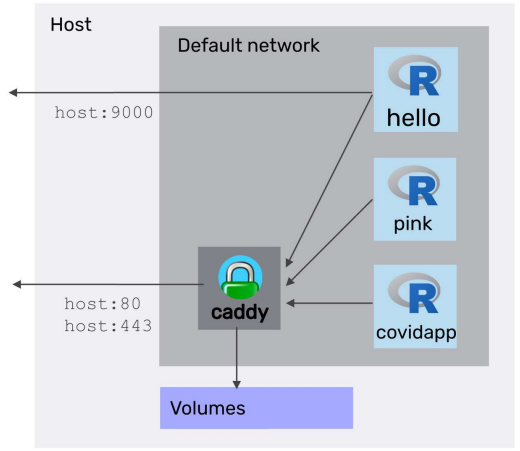
- Run multiple containers
  - docker-compose (feature-rich)
  - podman-compose
- *podman play kube*
  - Better integration with k8s
  - Run YAML locally and inside k8s
- Minishift (of Minikube)

```
version: "3.7"

services:
  hello:
    image: hello
  pink:
    build: ./pink
  covidapp:
    image: covidapp
  caddy:
    image: caddy

volumes:
```

docker-compose.yml



Minishift

Source

Belastingdienst

OpenShift workflow

OpenShift workflow

- *oc new-app* command
  - Recognizes programming language
  - Detects Dockerfile

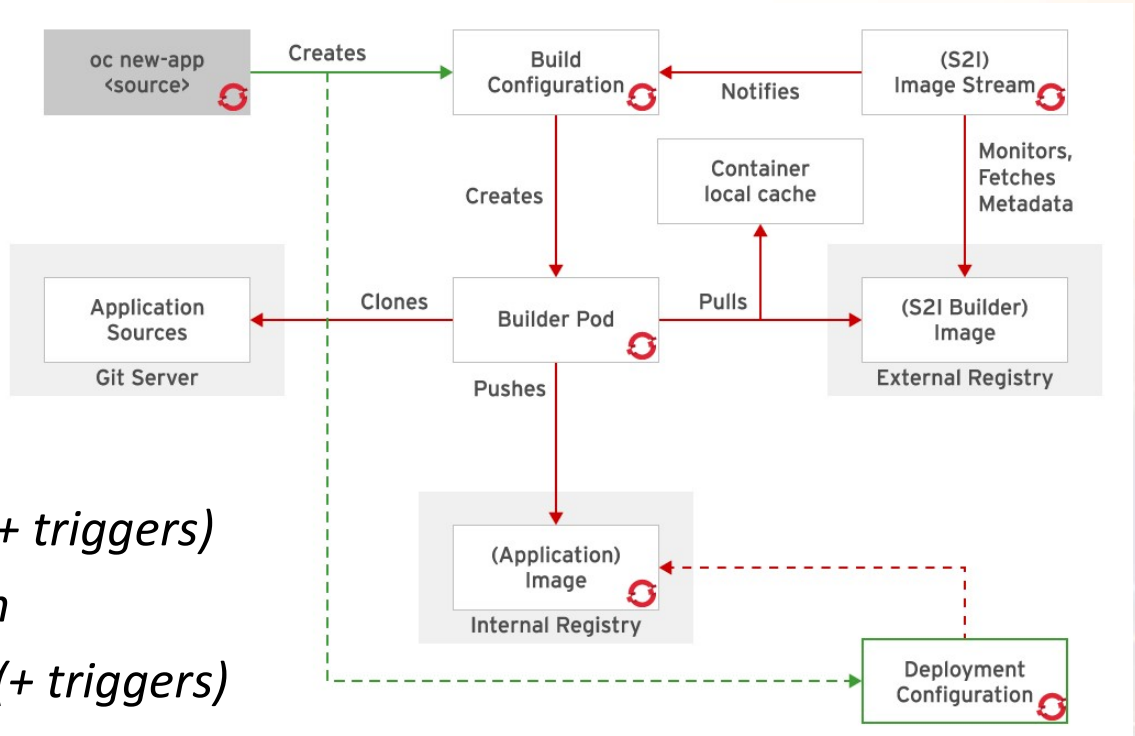
```
[user@jfall:~/demo]$ oc new-app https://my-git-server.nl/MyGitRepo
```

- BitBucket clone via SSH
  - Create SSH key pair
  - Create a (sealed) secret
  - Link to *builder* service account

```
[user@jfall:~/demo]$ oc secrets link builder bitbucket-secret
```

```
[peter@bld:~/openshift]$ oc new-app https://my-git-repo.nl/CollectTaxRepo
```

Belastingdienst



- Resources:
  - *BuildConfig (+ triggers)*
  - *ImageStream*
  - *Deployment (+ triggers)*
  - *Service*
  - *Route*

Source

OpenShift workflow


```
# Specify s2i image
[user@jfall:~/demo]$ oc new-app -i php https://my-git-server.nl/MyGitRepo

# Specify context-dir
[user@jfall:~/demo]$ oc new-app --context-dir=/test https://my-git-server.nl/MyGitRepo

# Specify a branch
[user@jfall:~/demo]$ oc new-app https://my-git-server.nl/MyGitRepo#beta-branch

# Build based on Dockerfile
[user@jfall:~/demo]$ oc new-app --strategy docker https://my-git-server.nl/MyGitRepo
```

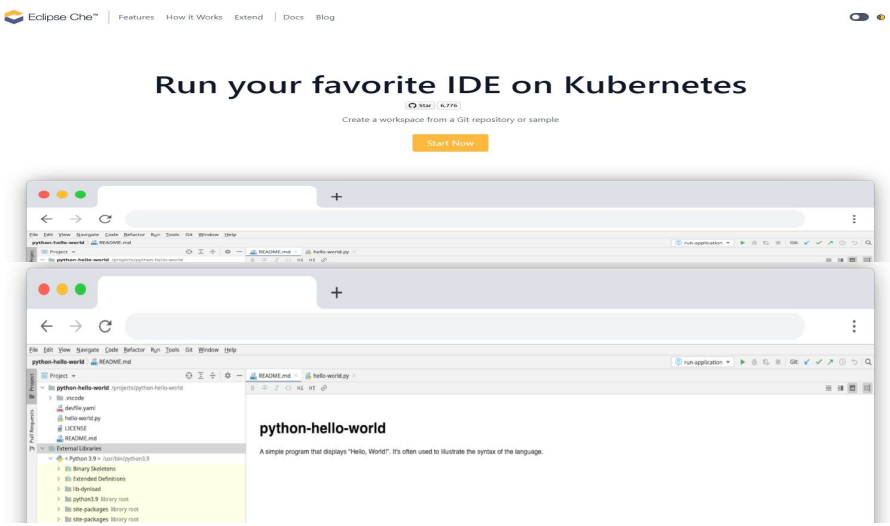
Belastingdienst

Remote workflow

Remote workflow

Gitpod

Coder



Remote workflow

OpenShift Dev Spaces

- OpenShift Dev spaces
  - Based on *Eclipse Che*
  - e.g. license management
- Extensions (CRD) to the Kubernetes API
  - DevSpace operator
  - DevWorkspace operator
- Universal Developer Image (UDI)

Summary

Belastingdienst

- Traditional workflows
- Containers
- OpenShift workflows
- Remote workflows

Summary

Belastingdienst

Local workflows → ? → Remote workflows

Local + Remote = Remocal

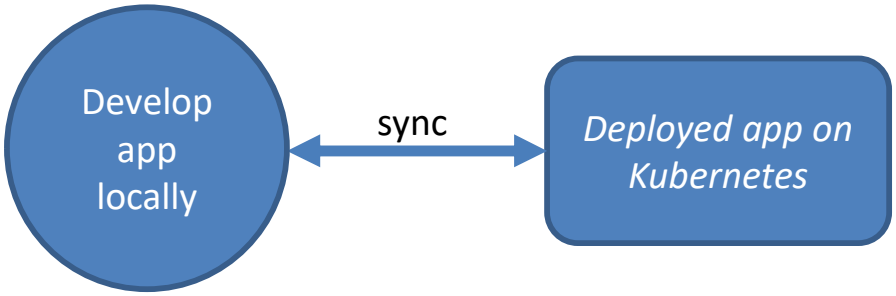
Belastingdienst

Remocal workflow

Remocal workflow

- ***Approach 1:***
  - develop locally + run remotely
  - *sync* with remote app

- ***Approach 2***:
  - develop + run locally
  - *route* remote traffic to local app

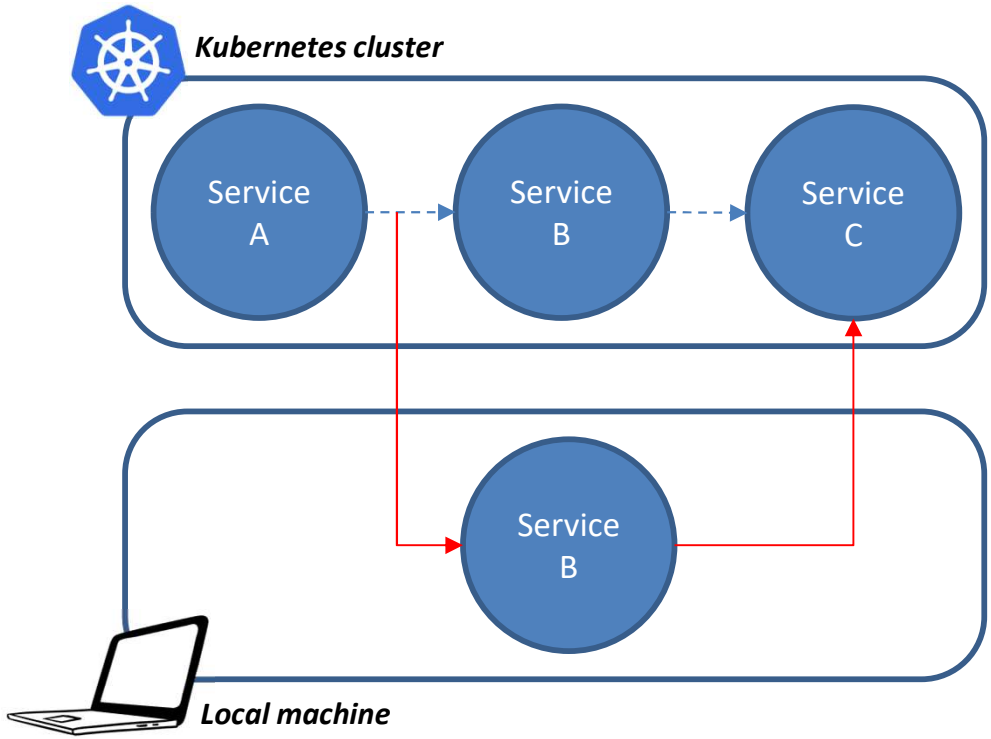


Remocal workflow

Belastingdienst

- ***Approach 1:***
  - develop locally + run remotely
  - *sync* with remote app

- ***Approach 2***:
  - develop + run locally
  - *route* remote traffic to local app



Remocal workflow

- *Remocal* development tools
  - OpenShift Do (odo)
  - OpenShift Dev Spaces
  - Skaffold
  - Telepresence
  - Draft
  - …
  - Quarkus

odo

Remocal workflow

Belastingdienst

- *OpenShift do (odo)*

- *odo init / odo dev*
- Devfile registry
- Independent of runtime
  - Quarkus
  - Liberty
  - …

```
[user@jfall2023:~/demo]$ odo dev
  __
 /  \__     Developing using the "demoservice" Devfile
 \__/  \    Namespace: jfall_namespace
 /  \__/    odo version: v3.1.0
 \__/

↳ Deploying to the cluster in developer mode
 •  Waiting for Kubernetes resources  ...
 ✓  Added storage m2 to component
 ⚠  Pod is Pending
 ✓  Pod is Running
 ✓  Syncing files into the container [542ms]
 ✓  Executing init-compile command  [23s]
 •  Executing the application (command: dev-run)  ...

 -  Forwarding from 127.0.0.1:40001 -> 8080

 -  Forwarding from 127.0.0.1:40002 -> 5858


↳ Dev mode
 Status:
 Watching for changes in the current directory

 Keyboard Commands:
[Ctrl+c] - Exit and delete resources from the cluster
     [p] - Manually apply local changes to the application on the cluster
```

Remocal workflow

Belastingdienst

- *OpenShift do (odo)*

- No need to create YAML files!

```
[user@jfall2023:~/demo]$ odo dev
  __
 /  \__     Developing using the "demoservice" Devfile
 \__/  \    Namespace: jfall_namespace
 /  \__/    odo version: v3.1.0
 \__/

↳ Deploying to the cluster in developer mode
 •  Waiting for Kubernetes resources  ...
 ✓  Added storage m2 to component
 ⚠  Pod is Pending
 ✓  Pod is Running
 ✓  Syncing files into the container [542ms]
 ✓  Executing init-compile command  [23s]
 •  Executing the application (command: dev-run)  ...

 -  Forwarding from 127.0.0.1:40001 -> 8080

 -  Forwarding from 127.0.0.1:40002 -> 5858


↳ Dev mode
 Status:
 Watching for changes in the current directory

 Keyboard Commands:
[Ctrl+c] - Exit and delete resources from the cluster
     [p] - Manually apply local changes to the application on the cluster
```

Automatic port forwarding

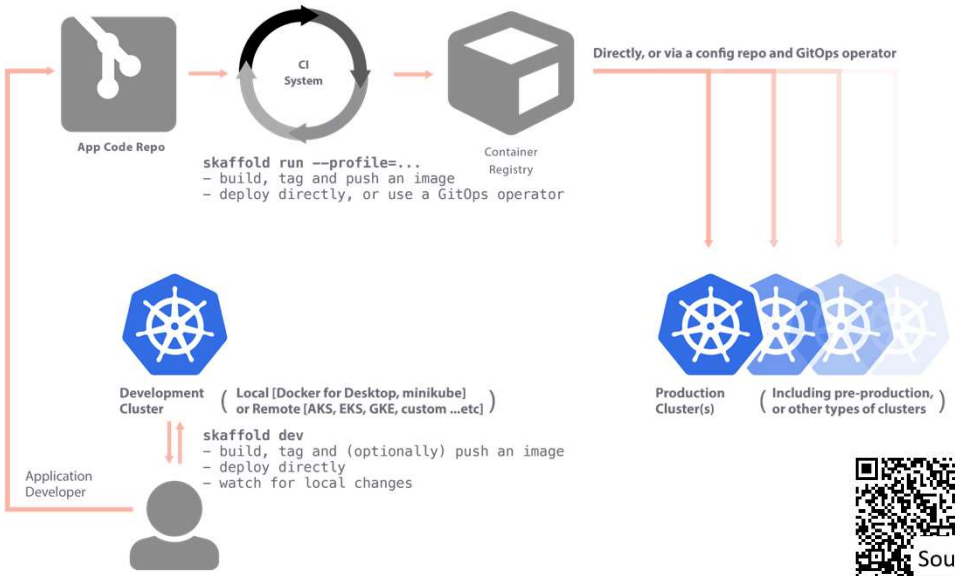
dev mode!

Remocal workflow

Belastingdienst

- *Skaffold* is a *client-side* CI/CD tool
- Automatically build, test and deploy
  - local or remote
- Create Kubernetes manifests (YAMLs) with *Dekorate*

```
[user@jfall2023:~/demo]$ skaffold init
[user@jfall2023:~/demo]$ skaffold dev
```



Source

Remocal workflow

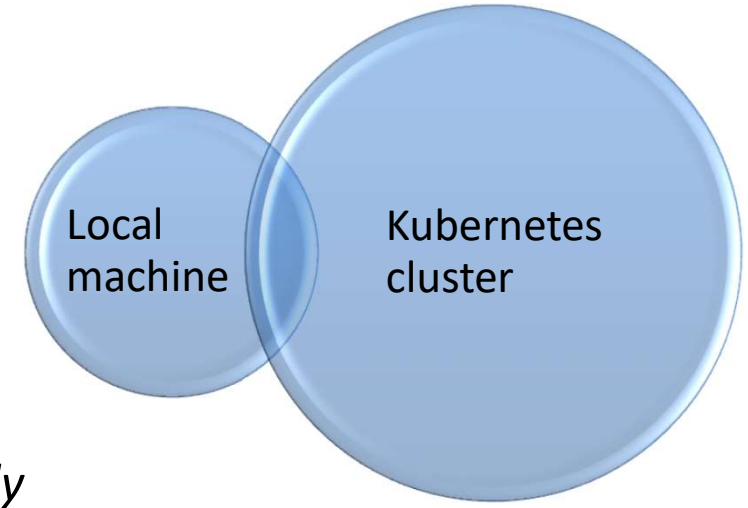
Belastingdienst

A1

- *Telepresence*

- Let your local developer machine act as part of the remote cluster

- Often used together with skaffold (or any other '*dev mode*' tool)
  - Skaffold : *build + deploy locally*
  - Telepresence : *test*



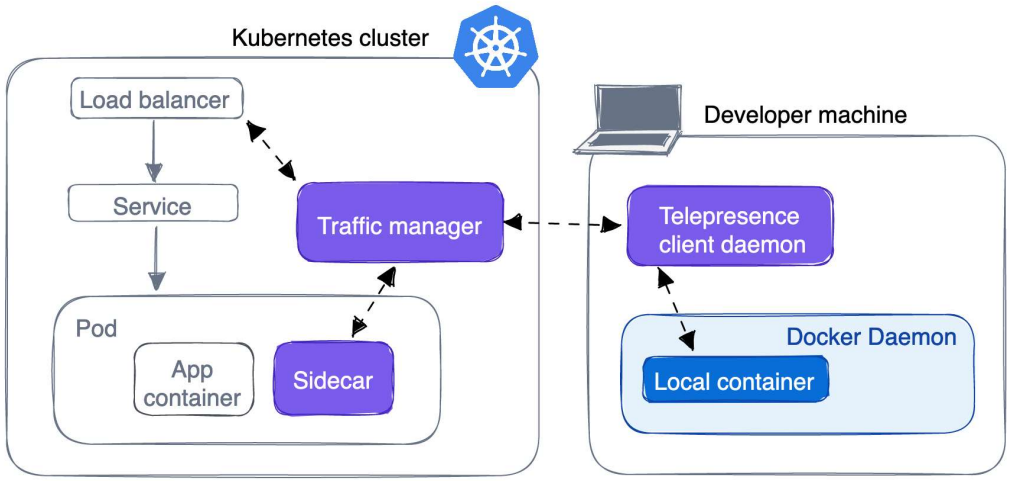
Dia 43

A1 Auteur; 23-10-2023

Remocal workflow

Belastingdienst

- *Telepresence* provides two components:
  - *Cluster-side* traffic manager
  - *Local* traffic agent
- Call remote microservices from your local machine
- Route remote traffic to your local application



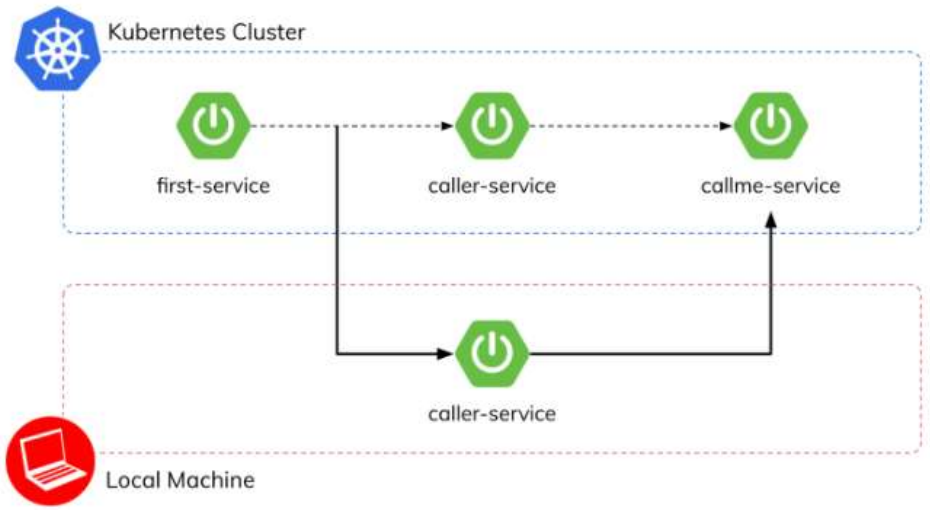

```
[user@jfall2023:~/demo]$ telepresence connect
```

```
[user@jfall2023:~/demo]$ telepresence intercept demo-service --port 8080:8080
```

Source

Remocal workflow

- *Telepresence* intercepts a call to the remote running microservice
- Call will be routed to the microservice running locally
- Local microservice is able to call remote running microservices



Source

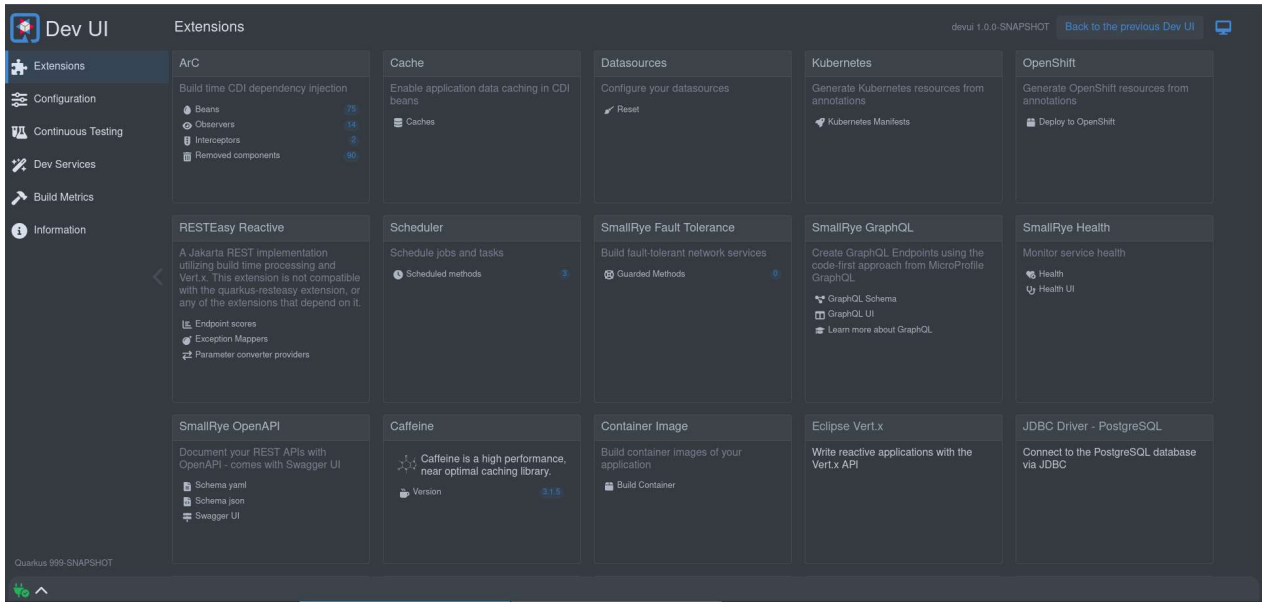

Belastingdienst
Quarkus
MacBook Pro

Quarkus

- *Kubernetes-native* Java development stack
- *Build on standards*
  - *MicroProfile*
  - *Jakarta*

Quarkus

- Quarkus *dev* mode
- Continuous testing
- Dev UI
- Quarkus *remote-dev* mode



Source

Quarkus

- Dev services
  - Based on *testcontainers*
- Fully automatic!
  - No need to add testcontainer libraries manually
  - No manual integration
  - No configuration
  - No code

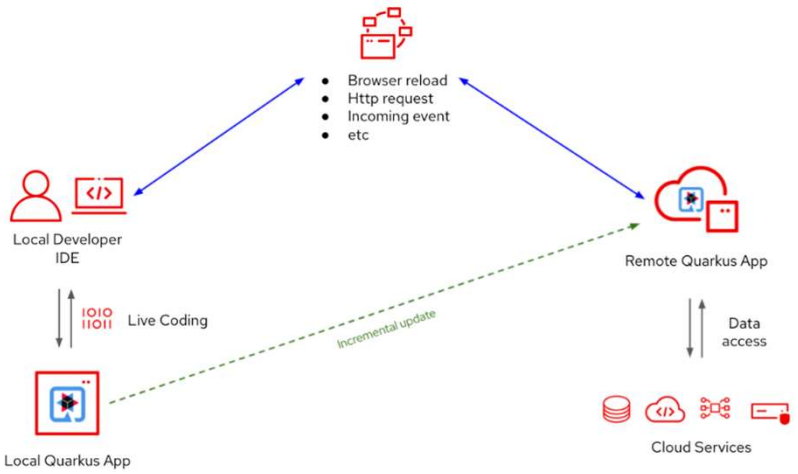
Quarkus

- *Quarkus Remote Development*

```
[user@jfall2023:~/demo]$ ./mvnw quarkus:remote-dev
```

- Create *mutable* JAR

```
quarkus.package.type=mutable-jar 1
quarkus.live-reload.password=changeit 2
quarkus.live-reload.url=http://my.cluster.host.com:8080 3
```



Source

Quarkus

- *Quarkus* : *build + deploy locally*
- *Jkube* : *test*

QUARKUS + JKube

Quarkus

Belastingdienst

- *Eclipse Jkube Remote Development*

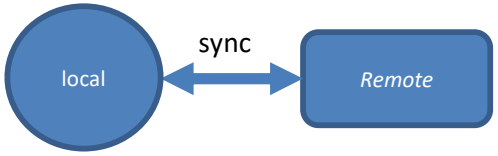
Definition local microservices

Definition remote microservices

```
<plugin>
  <groupId>org.eclipse.jkube</groupId>
  <artifactId>openshift-maven-plugin</artifactId>
  <version>1.14.0</version>
  <configuration>
    <remoteDevelopment>
      <localServices>
        <localService>
          <serviceName>service-B</serviceName>
          <port>8082</port>
        </localService>
      </localServices>
      <remoteServices>
        <remoteService>
          <hostname>service-A</hostname>
          <port>80</port>
          <localPort>8083</localPort>
        </remoteService>
        <remoteService>
          <hostname>service-C</hostname>
          <port>80</port>
          <localPort>8084</localPort>
        </remoteService>
      </remoteServices>
    </remoteDevelopment>
  </configuration>
</plugin>
```

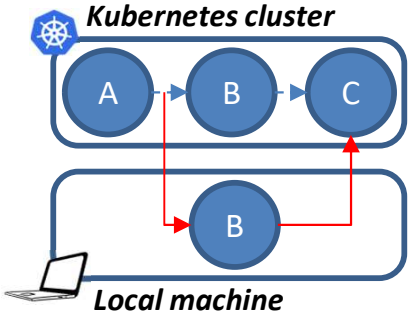
```
[user@jfall2023:~/demo]$ ./mvnw oc:remote-dev
```

Summary



Sync approach

- Odo
- Skaffold
- Quarkus remote development



Route approach

- Skaffold + Telepresence
- Quarkus + JKube