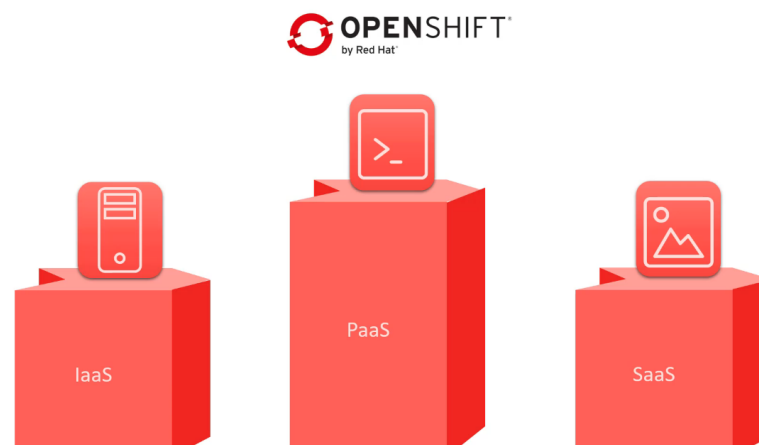


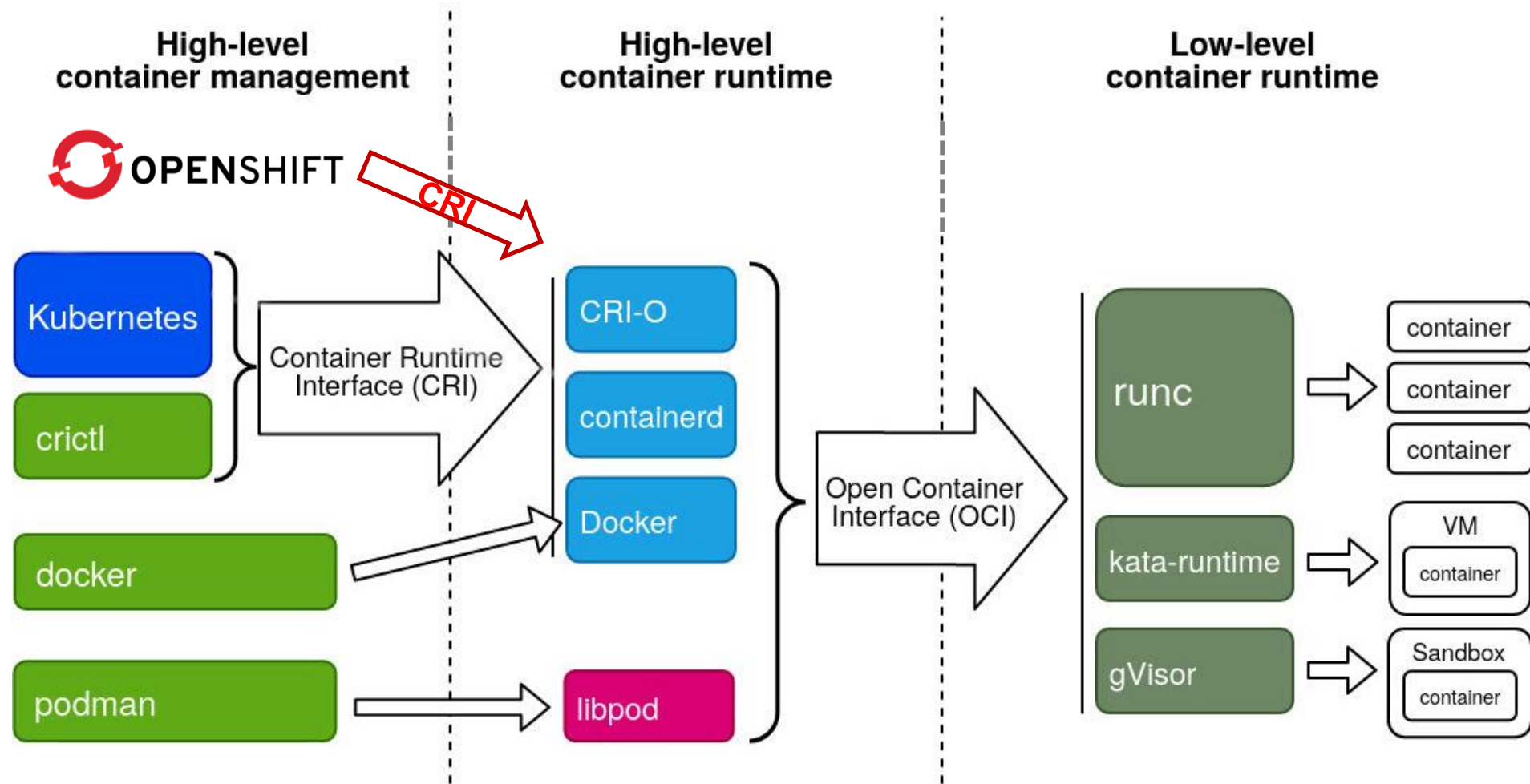
OpenShift

O que é o Openshift

É a plataforma da RedHat baseada em kubernetes para rodar e distribuir aplicações containerizadas, é necessário licença para sua utilização. O Openshift utiliza como base para sua distribuição final o OKD, este sim uma distribuição open-source do kubernetes. O OpenShift também é uma distribuição kubernetes que utiliza o CRI-O como container runtime. Possui um próprio CLI o “oc” (Openshift Console), que foi criado baseado no mais alto nível do “kubectl”.



O que é o Openshift



Projects e Users

O conceito de projetos no OpenShift tem a relação de um para um com o namespace e seu usuário. Caso deseje que mais de um usuário tenha acesso ao seu projeto, podemos adicioná-los como colaboradores. O Projeto cria uma abstração sobre o namespace do kubernetes onde o OpenShift de forma transparente faz toda a administração definindo quotas, limites, controle de acesso e gerência de usuários e grupos. Projetos que começam com “openshift-” ou “kube-” são de uso exclusivo administrativo.

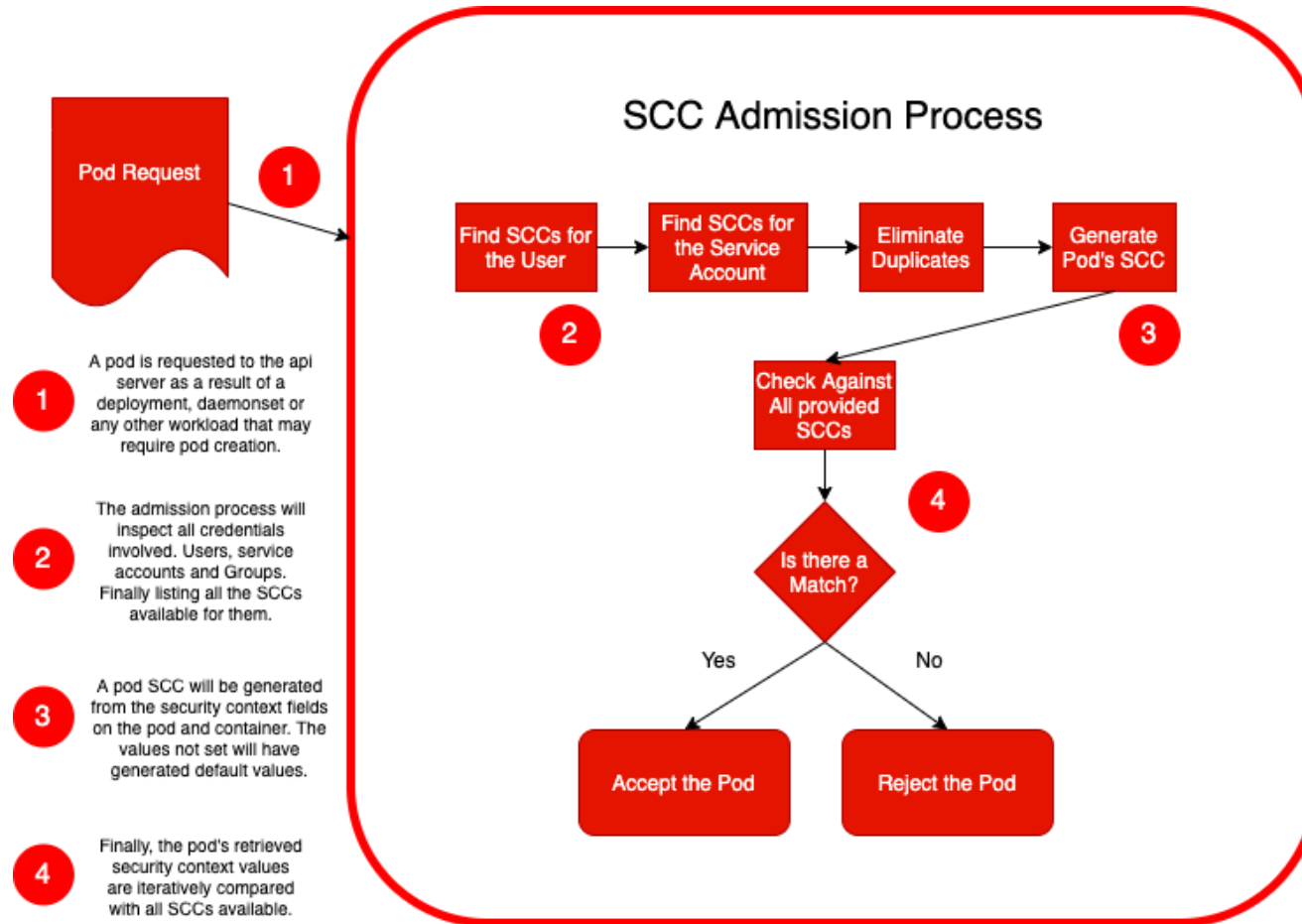
SCC

Security Context (SC) especificam as permissões que as aplicações precisam no manifesto de deploy, enquanto Security Context Constraints (SCC), especificam as permissões que o cluster permite, ou seja, limitam o acesso desta aplicação autorizando apenas o que foi liberado para o cluster. São semelhantes ao PSP do Kubernetes (descontinuado) hoje substituído pelo Pod Security Admission (PSA). O Openshift trás os seguintes SCCs por default:

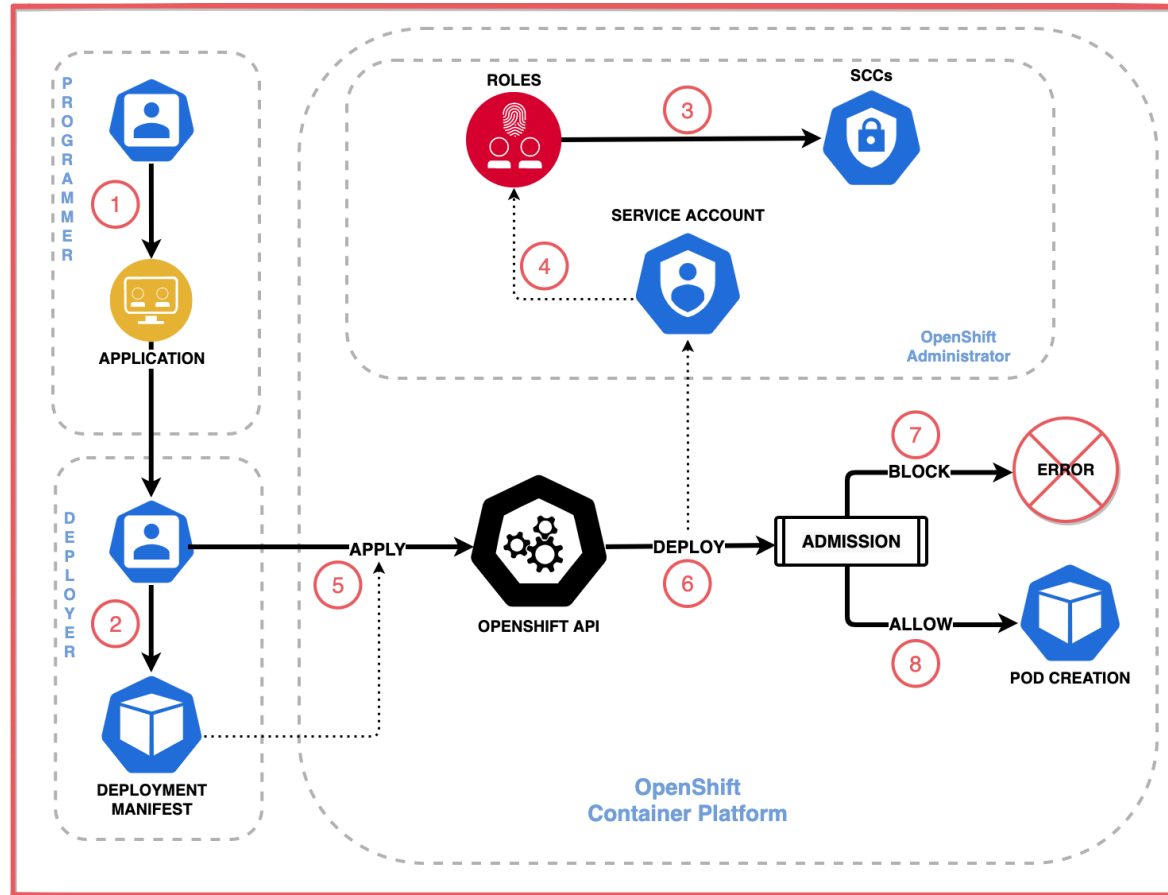
- restricted
- nonroot
- anyuid
- hostmount-anyuid
- hostnetwork
- node-exporter
- hostaccess
- Privileged

<https://cloud.redhat.com/blog/managing-sccs-in-openshift>

SCC



SCC



<https://developer.ibm.com/learningpaths/secure-context-constraints-openshift/intro/>

SCC

Project: test ▾

Pods > Pod details

P example-787f749bb-qj27n Running

Details Metrics YAML Environment Logs Eve

```
22     },
23     "default": true,
24     "dns": {}
25   }
26   openshift.io/scc: restricted
27   resourceVersion: '116722'
28   name: example-787f749bb-qj27n
29   uid: 611d0cf1-3f29-44e6-91ec-55f55278cffc
30   creationTimestamp: '2022-02-04T16:52:22Z'
```

```
[root@crc-dzk9v-master-0 /]# oc describe scc restricted
Name: restricted
Priority: <none>
Access:
  Users: <none>
  Groups: system:authenticated
Settings:
  Allow Privileged: false
  Allow Privilege Escalation: true
  Default Add Capabilities: <none>
  Required Drop Capabilities: KILL,MKNOD,SETUID,SETGID
  Allowed Capabilities: <none>
  Allowed Seccomp Profiles: <none>
  Allowed Volume Types: configMap,downwardAPI,emptyDir,persistent
  VolumeClaim,projected,secret
  Allowed Flexvolumes: <all>
  Allowed Unsafe Sysctls: <none>
  Forbidden Sysctls: <none>
  Allow Host Network: false
  Allow Host Ports: false
  Allow Host PID: false
  Allow Host IPC: false
  Read Only Root Filesystem: false
  Run As User Strategy: MustRunAsRange
    UID: <none>
    UID Range Min: <none>
    UID Range Max: <none>
  SELinux Context Strategy: MustRunAs
    User: <none>
    Role: <none>
    Type: <none>
    Level: <none>
  FSGroup Strategy: MustRunAs
    Ranges: <none>
  Supplemental Groups Strategy: RunAsAny
    Ranges: <none>
```

<https://developer.ibm.com/learningpaths/secure-context-constraints-openshift/scc-tutorial/>

Deployments vs DeploymentConfigs

DeploymentConfigs, expandem o desenvolvimento e a disponibilização dos ciclos de vida da aplicação, criando um novo Replication Controller, disponibilizando um template para as aplicações em execução, possibilitando:

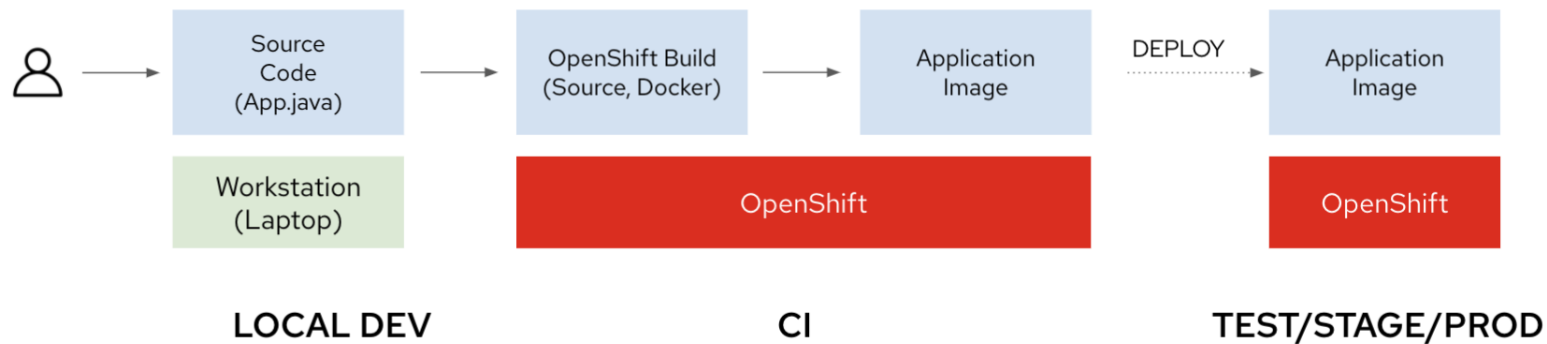
- Triggers
- Estratégias de deploy customizadas
- Lifecycle Hooks
- Versionamento Customizado
- Controle manual do autoscaling e replication

```
apiVersion: v1
kind: DeploymentConfig
metadata:
  name: frontend
spec:
  replicas: 5
  selector:
    name: frontend
  template: { ... }
  triggers:
    - type: ConfigChange ①
    - imageChangeParams:
        automatic: true
        containerNames:
          - helloworld
        from:
          kind: ImageStreamTag
          name: hello-openshift:latest
        type: ImageChange ②
  strategy:
    type: Rolling ③
```

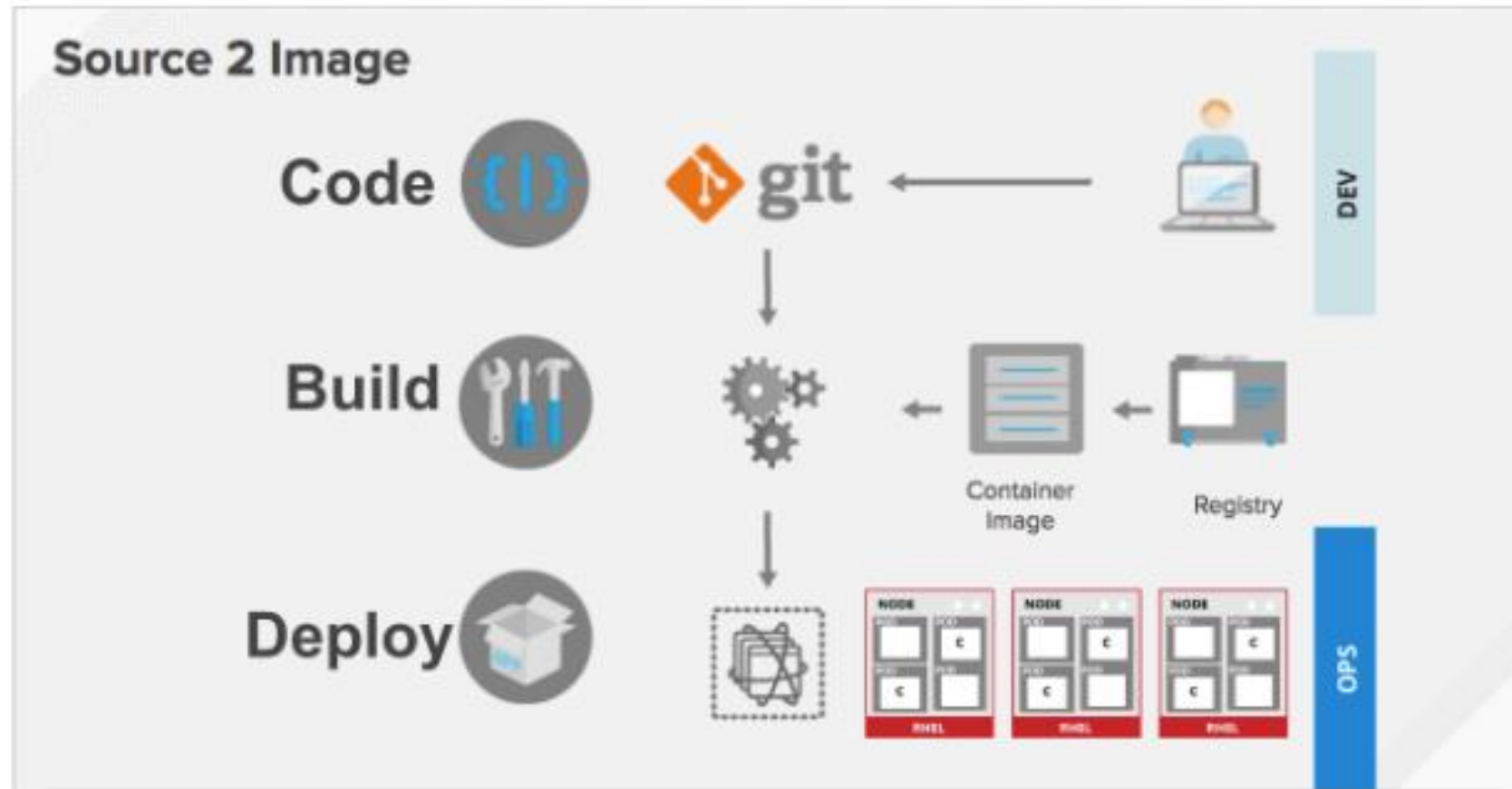
Builds e Deployments

Openshift vem com integração nativa a repositórios git como o GitLab, GitHub, BitBucket, etc. Desta maneira podemos fazer um deploy de uma aplicação direto de seu repositório de códigos. Diferente de quando colocamos uma fonte como imagem docker para deploy, ao inserir o repositório git, o Openshit lança automaticamente uma tarefa de Build, em seguida é criado um Deployment, Service e Route para efetuar a disponibilização da aplicação.

O OpenShift possui um Container Registry interno onde guarda as imagem provenientes de seus builds.

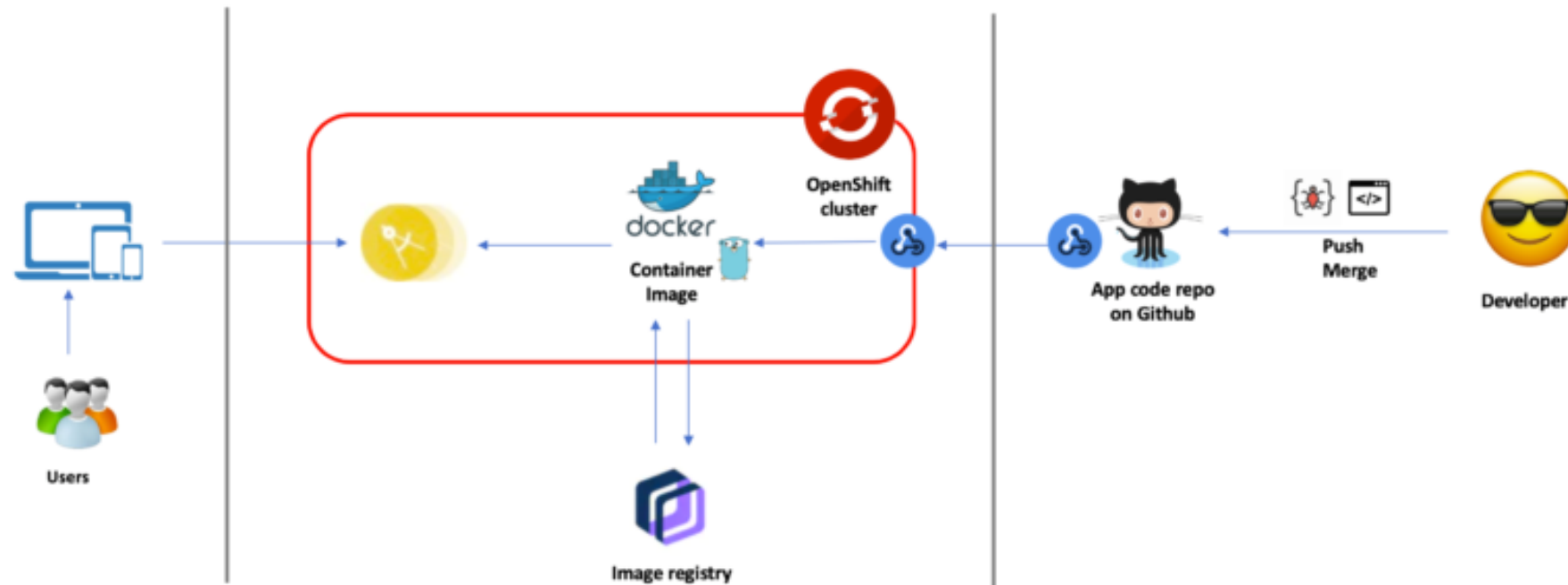


Builds e Deployments



Builds e Deployments

Webhook



Tipos de Builds

Source-to-Image

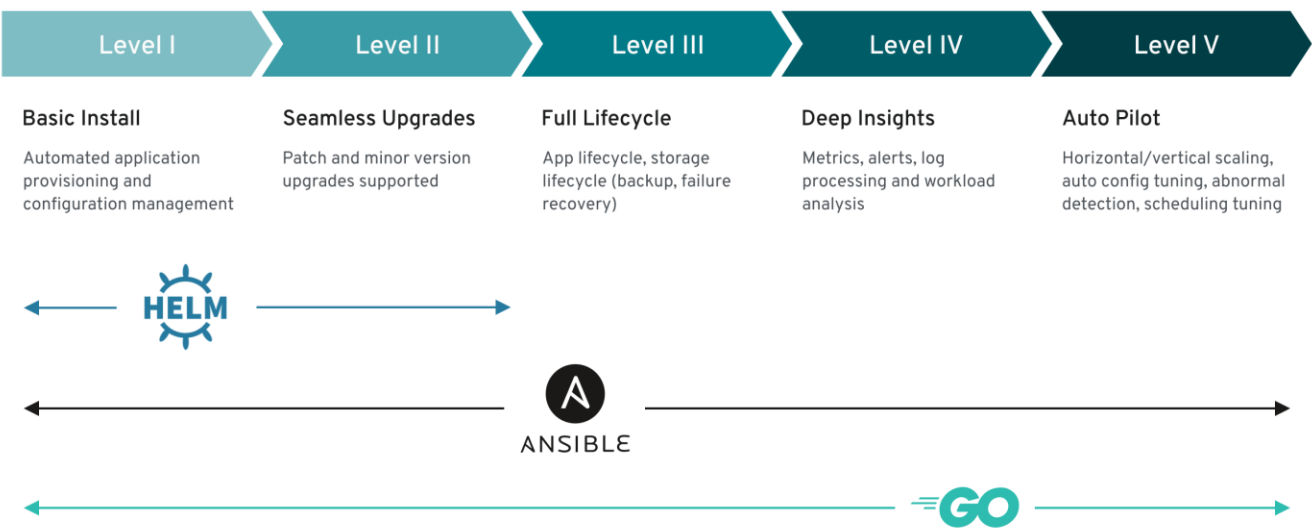
```
source:
  git:
    ref: master
    uri: 'https://github.com/openshift/ruby-ex.git'
    type: Git
  strategy:
    type: Source ←
    sourceStrategy:
      from:
        kind: ImageStreamTag
        name: 'ruby:2.4'
        namespace: openshift
      env: []
  triggers:
    - type: ImageChange
      imageChange: {}
    - type: ConfigChange
```

Docker

```
source:
  type: Git
  git:
    uri: 'https://github.com/openshift/ruby-hello-world.git'
  strategy:
    type: Docker ←
    dockerStrategy:
      from:
        kind: ImageStreamTag
        name: 'ruby:latest'
        namespace: openshift
      env:
        - name: EXAMPLE
          value: sample-app
  output:
    to:
      kind: ImageStreamTag
```

Operators

Operadores são semelhantes ao Helm, porém vão mais além, pois não se resumem a apenas facilitar a instalação e atualização de aplicações complexas, mas também podem gerir todo o lifecycle, monitoração e auto-pilot. Embora tenha sido um projeto capitaneado pela RedHat, podemos utilizar operadores em qualquer cluster kubernetes.



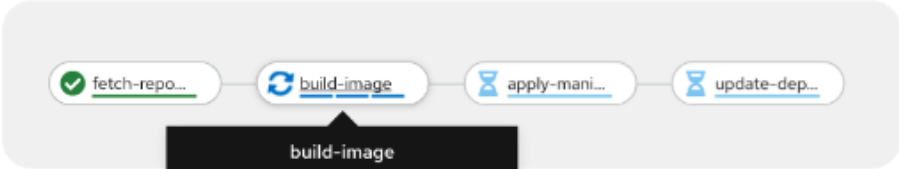
	Helm Chart	Operator
Packaging	✓	✓
App Installation	✓	✓
App Update (kubernetes manifests)	✓	✓
App Upgrade (data migration, adaption, etc)	-	✓
Backup & Recovery	-	✓
Workload & Log Analysis	-	✓
Intelligent Scaling	-	✓
Auto tuning	-	✓

Pipelines

OpenShift Pipelines é uma solução Kubernetes-native CI/CD baseada no Tekton. Ela provê uma experiência CI/CD através de integrações passo a passo com componentes e ferramentas do Openshift.

[Details](#) [YAML](#) [Task Runs](#) [Logs](#) [Events](#)

Pipeline Run details



Name
build-and-deploy-...

Namespace
NS pipelines-tutorial

Labels
tekton.dev/pipeline=build-and-deploy

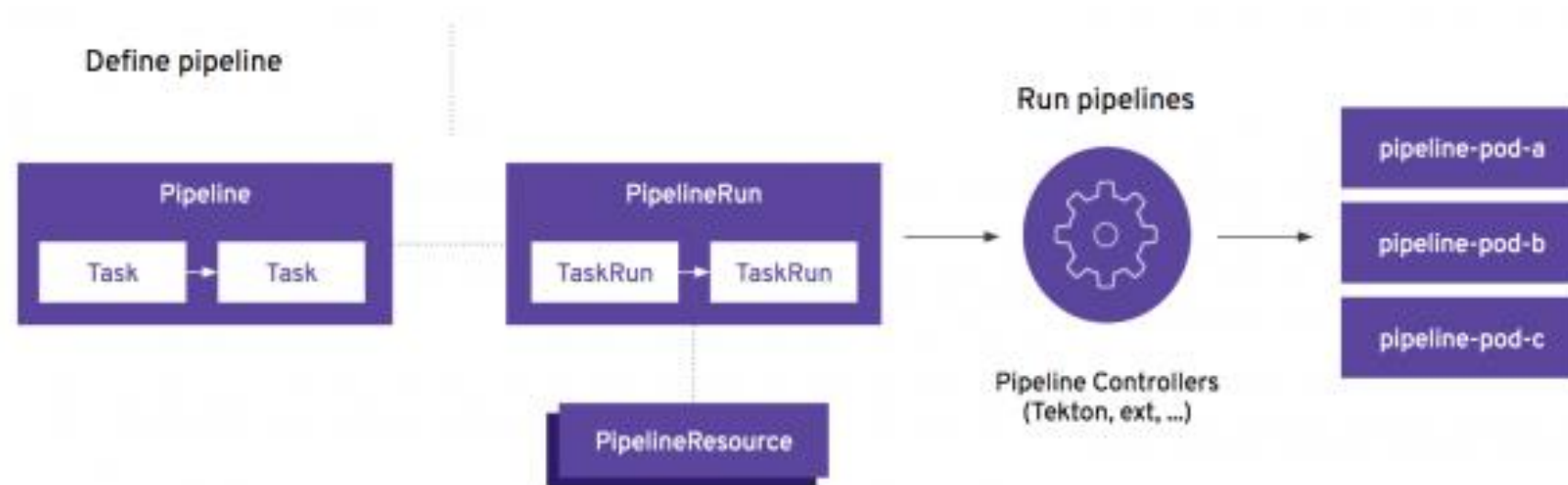
Status
Running

Pipeline
PL build-and-deploy

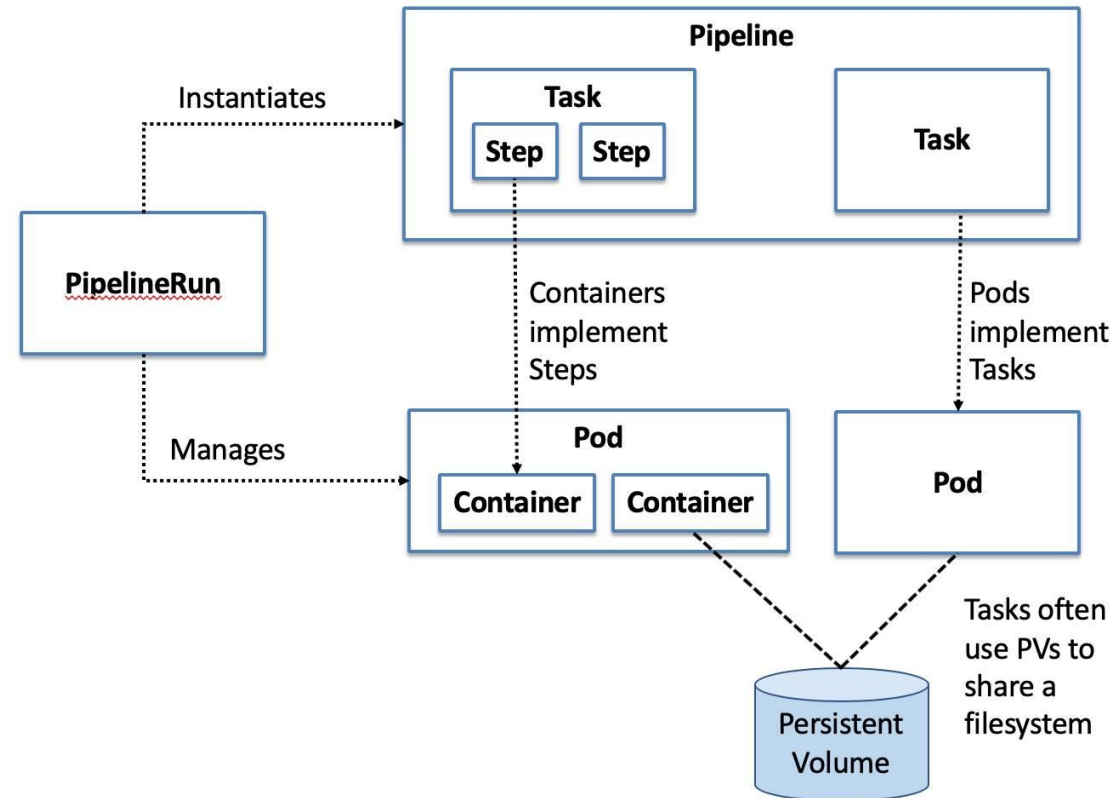
Triggered by:
kube:admin

[Edit](#)

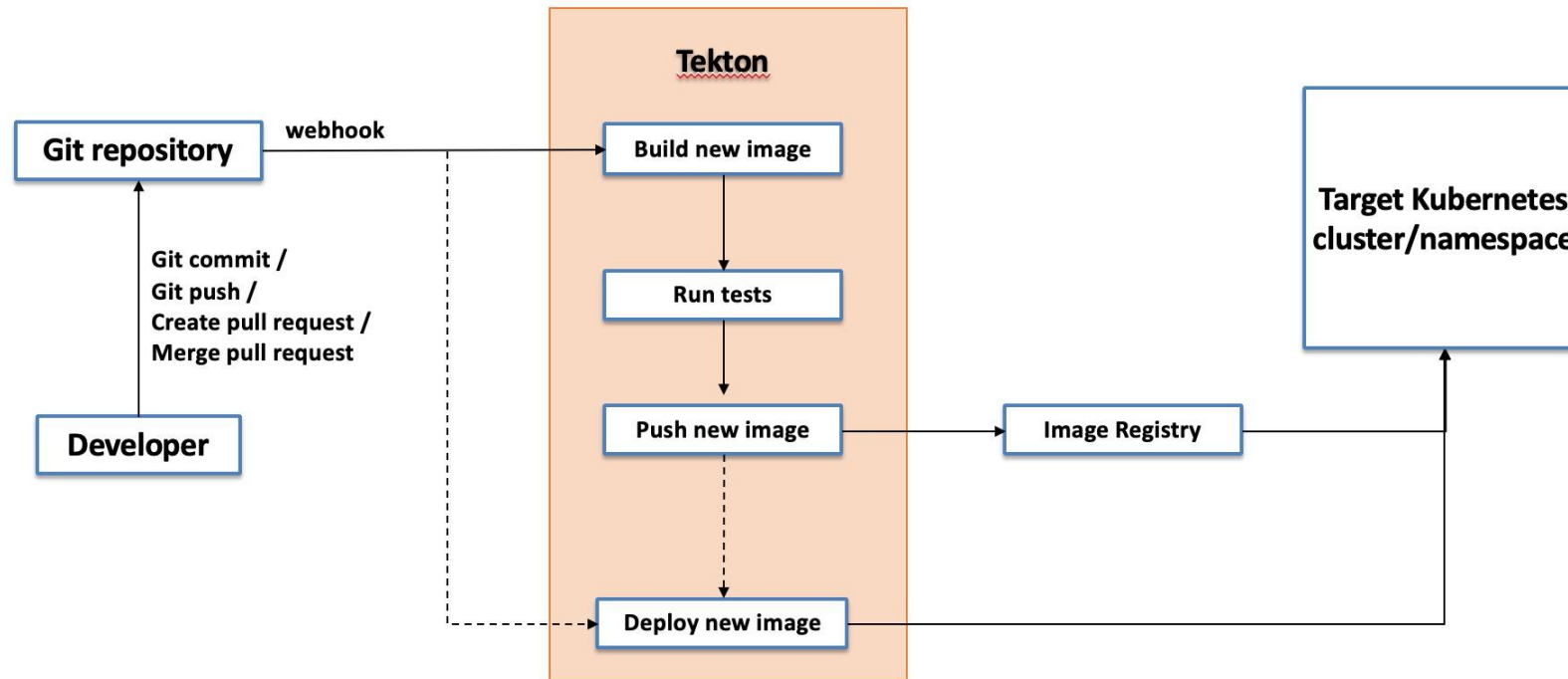
Pipelines



Pipelines



Pipelines



Pipelines

CT buildah

Actions

View shortcuts

Display name *

buildah

Parameters

Use this format when you reference variables in this form: \${ }

IMAGE *

\$(params.IMAGE)

Reference of the image buildah will produce.

Tasks

TasksTaskRunsClusterTasks

Name

Search by name...

Name	Namespace
T apply-manifests	NS goapp
T update-deployment	NS goapp



Parameters

Name *	Description	Default value
deployment-name	Description	Default value
git-url	Description	Default value
git-revision	Description	Default value
IMAGE	Description	Default value
manifest-dir	Description	Default value
docker-file	Description	Default value

Start Pipeline

Parameters

deployment-name *

git-url *

git-revision *

IMAGE *

manifest-dir *

CancelStart

PipelineRuns > PipelineRun details

PLR pipeline-pod60x Succeeded

DetailsYAMLTasksRunsLogsEvents

DownloadDownload all task logsExpand

git-clone

buildah

apply-manifests

update-deployment

buildah

Downgrading: 0 packages

Downloading packages...

Running transaction test...

Installing: libsemanage;2.9-6.el8;x86_64;ubi-8-baseos

Installing: shadow-utils;2:4.6-14.el8;x86_64;ubi-8-baseos

Complete.

STEP 18: RUN mkdir \${APP_BASEDIR}

STEP 19: WORKDIR \${APP_BASEDIR}

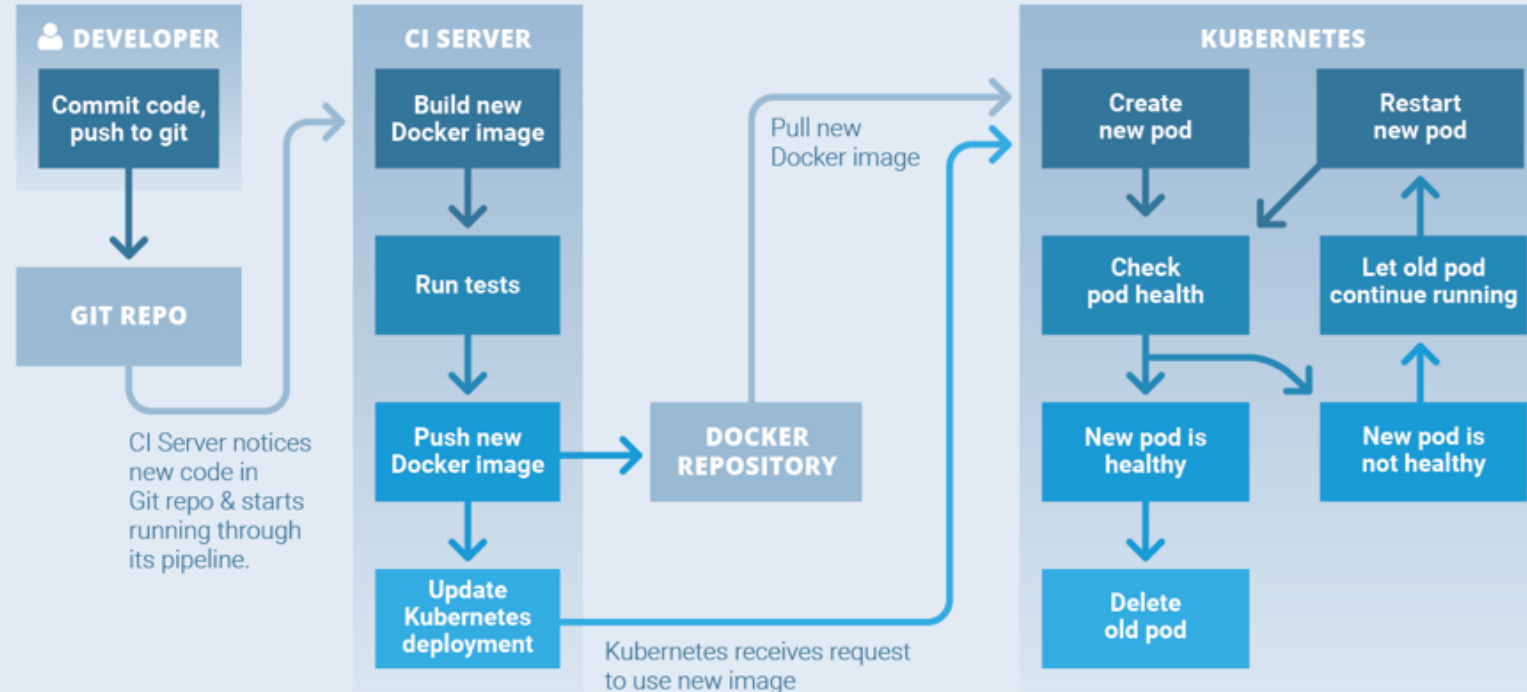
STEP 20: COPY --from=builder-goapp /tmp/goapp .

STEP 21: COPY --from=builder-migrate /tmp/migrate .

STEP 22: RUN chown -R \${APPUSER}. \${APP_BASEDIR}

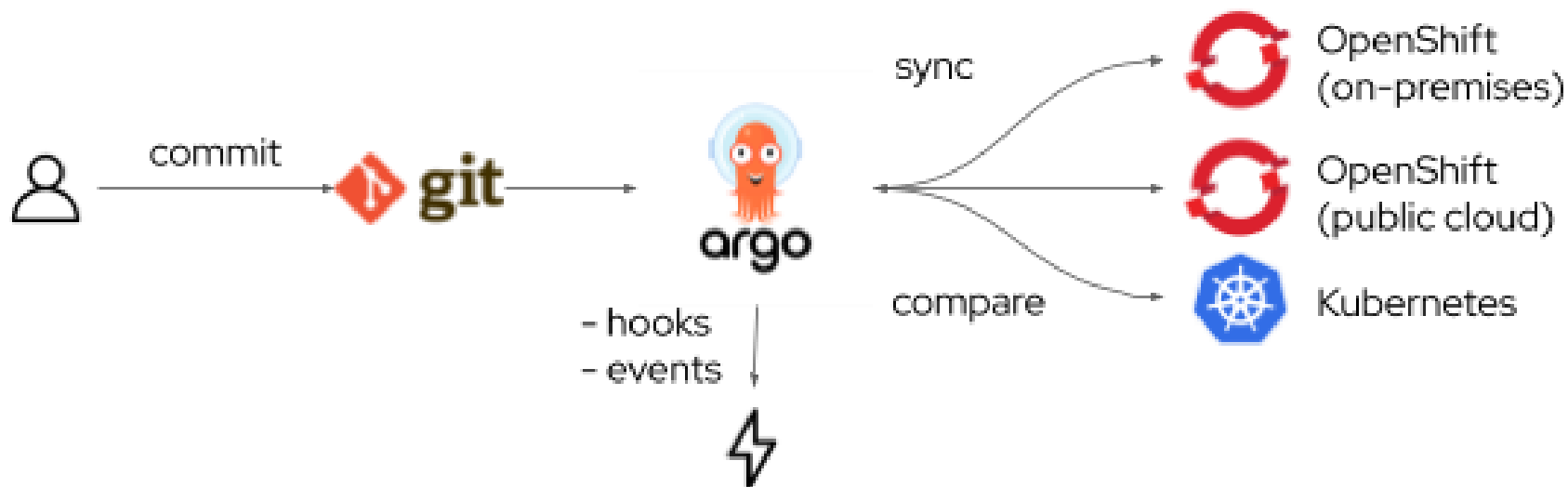
Pipelines

CI/CD Pipeline Workflow with Kubernetes



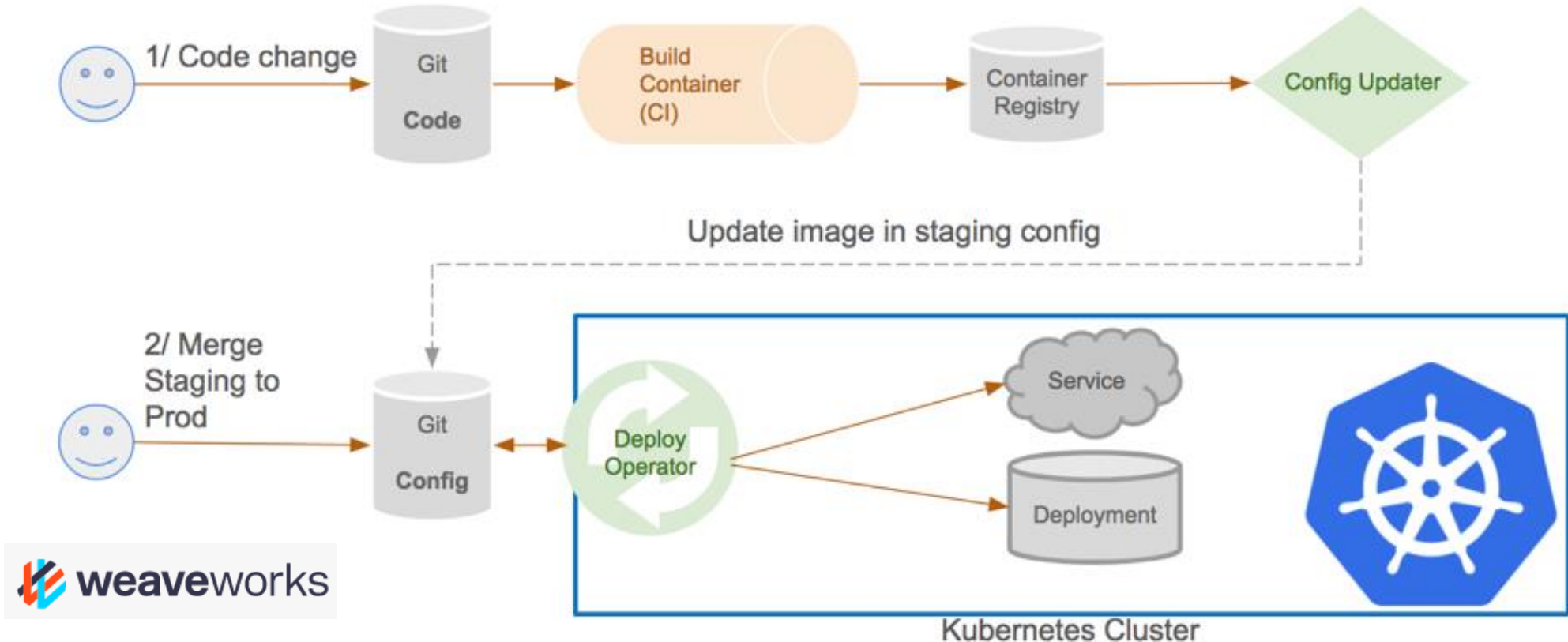
GitOps

OpenShift GitOps é um add-on que provê o ArgoCD e uma série de outras ferramentas para implementar os fluxos de trabalho do GitOps. OpenShift GitOps está disponível como Operator no OperatorHub.



Pipeline + Gitops

Example GitOps Pipeline



Tipos de Instalação do Openshift

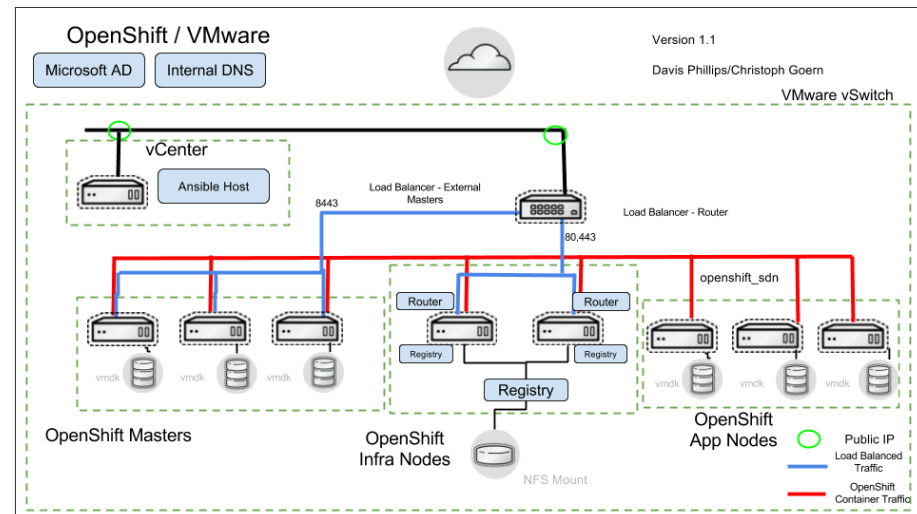
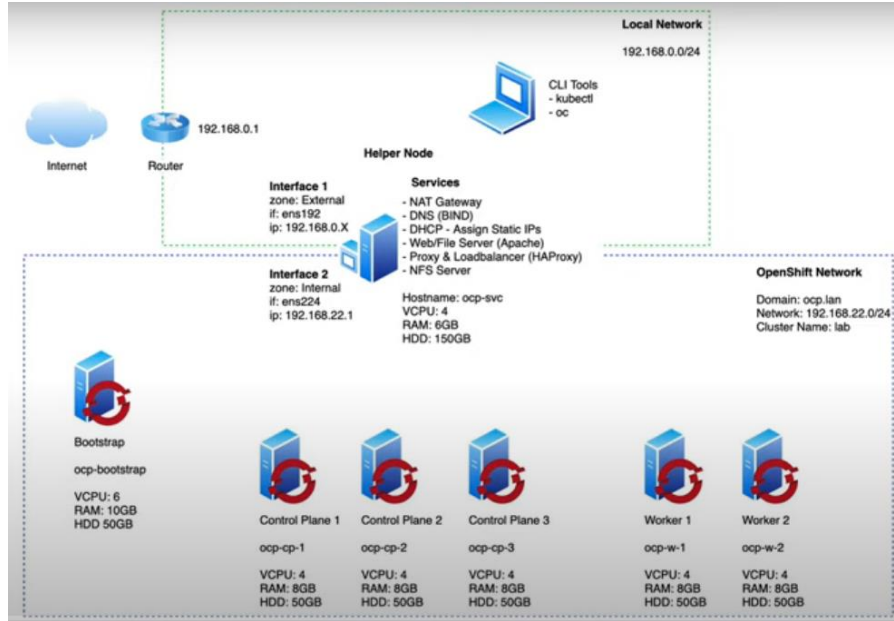
Openshift 4 tem dois tipos de instaladores, que possibilitam uma maior flexibilidade frente a versão 3 que fazia uso do Ansible.

Para instalar o Openshift é necessário escolher entre Installer-Provisioned Infrastructure (ou IPI) e User-Provisioned Infrastructure (ou UPI).

O IPI instala o Openshift de maneira automática, criando e configurando todos os componentes, é a maneira mais simples de instalar, feito para as clouds públicas e Openstack.

O UPI é uma opção válida para duas situações: quando as condições de instalação não suportam os requisitos internos, como por exemplo reutilizar uma rede já existente. Segundo, quando não possuímos todas as opções necessárias para o deploy completo e manualmente temos que criar recursos como por exemplo LBs, DNS. Este é um exemplo de Instalação em Baremetal ou VMWare.

Openshift UPI Architecture



Openshift IPI Architecture

