

# OpenShift

## Network and Storage

- OpenShift SDN, or OVN
- Cinder, Manila, EBS, Azure Disk, Azure File, GCE PD, vsphere volume

## Dev, Test, Deploy

Registry  
Jenkins



## Infrastructure

CoreOS  
Node mgt with Ansible  
Auto-scaling

## App Management

Web Console  
EFK  
Istio

## Security

RBAC  
IAM  
Authentication

Sept 2021

(c) Bernd Bausch 2021

1

# OpenShift Products

## OpenShift Container Platform

Red Hat's private, on-premise cloud application deployment and hosting platform.

## OpenShift Dedicated

Red Hat's managed public cloud application deployment and hosting service.

## Red Hat OpenShift Service on AWS

Red Hat OpenShift Service on AWS (ROSA) is a fully-managed OpenShift service, jointly managed and supported by Red Hat and Amazon Web Services (AWS).

## OpenShift on IBM Cloud

With Red Hat OpenShift on IBM Cloud, you can deploy apps on highly available OpenShift clusters.

## Azure Red Hat OpenShift

Azure Red Hat OpenShift provides single-tenant, high-availability Kubernetes clusters on Azure, supported by Red Hat and Microsoft.

## Red Hat Advanced Cluster Security for Kubernetes

An enterprise-ready, Kubernetes-native container security solution that enables you to securely build, deploy, and run cloud-native applications anywhere.

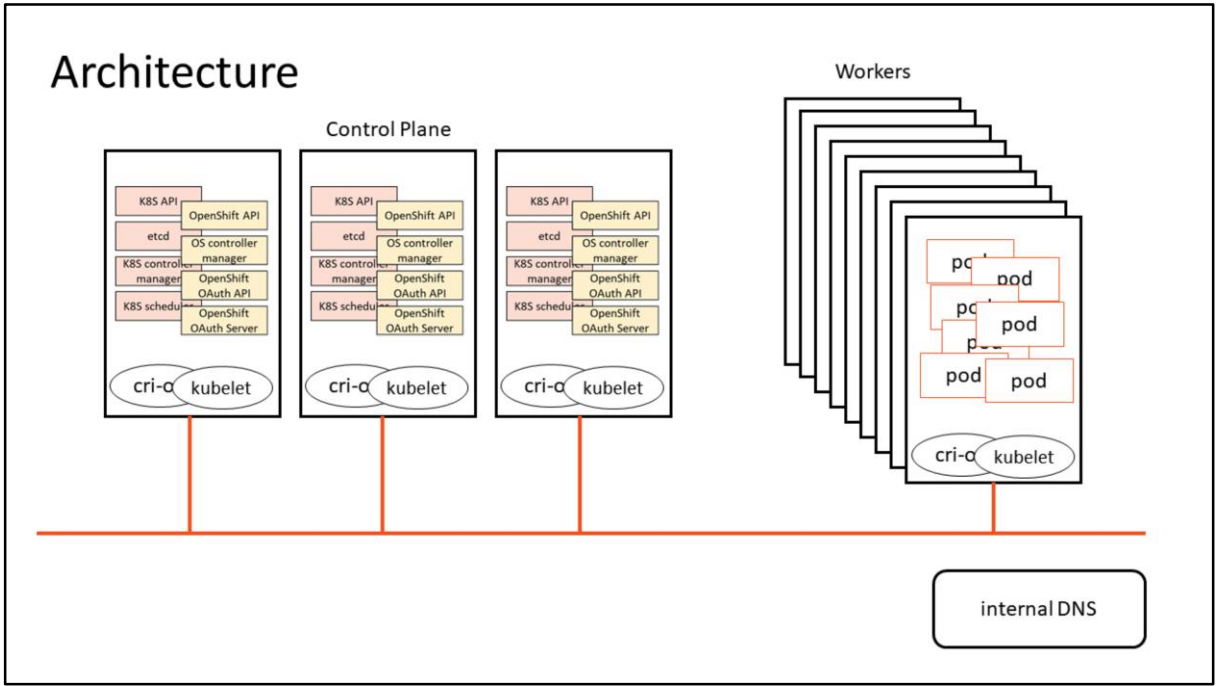
## OpenShift Online

Red Hat's public cloud application deployment and hosting platform.

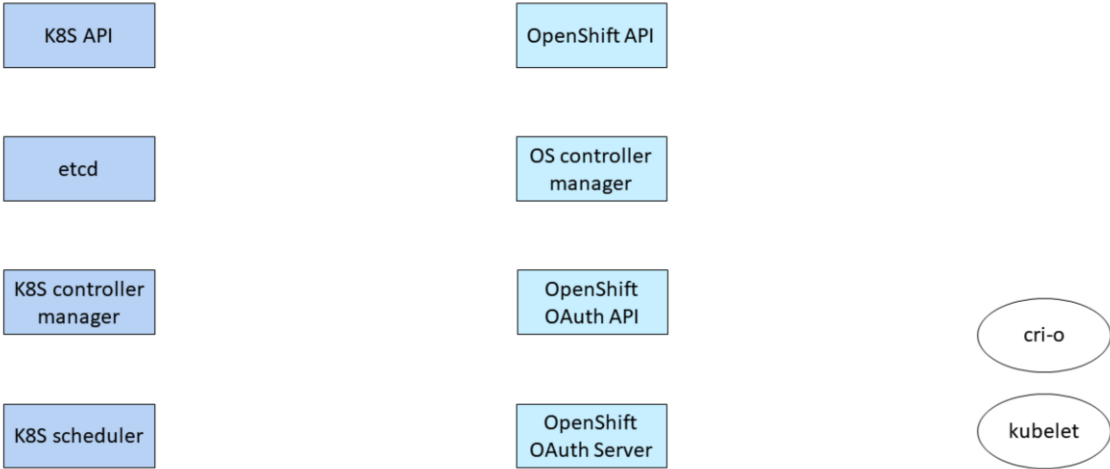
## OpenShift Kubernetes Engine

The OpenShift Kubernetes Engine is the core of the OpenShift Container Platform. Use OpenShift Container Platform docs links for OpenShift Kubernetes Engine documentation.

Screenshot from <https://docs.openshift.com>



# Control Plane Services

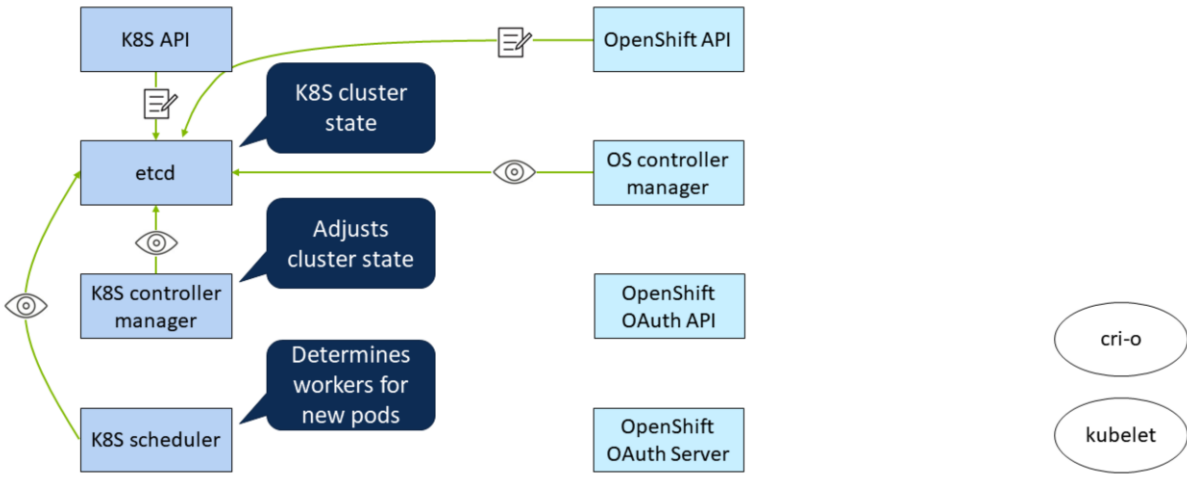


Sept 2021

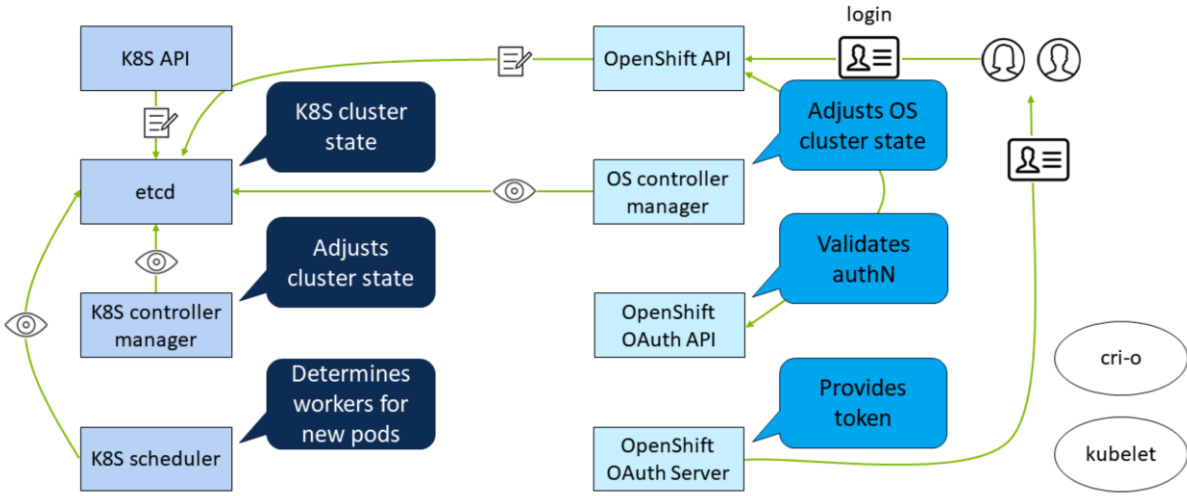
(c) Bernd Bausch 2021

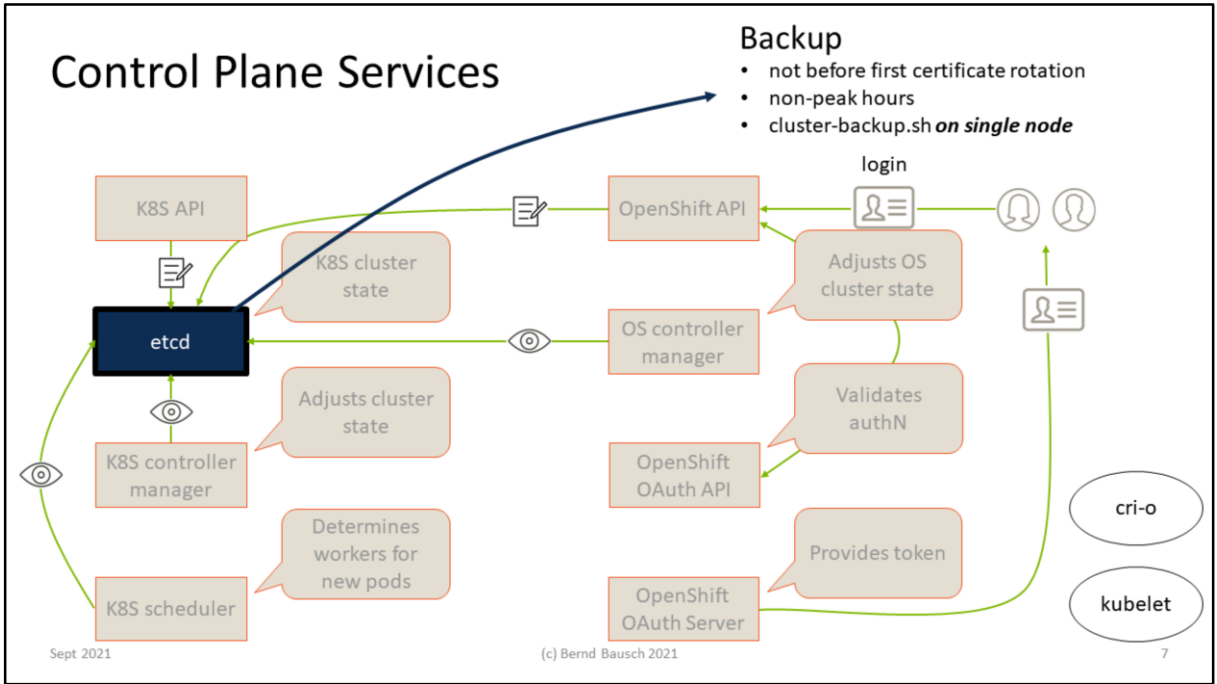
4

Control Plane Services



Control Plane Services





# The command line

oc	verb	object type	object
	get	pod	
	describe	replicaSet, rs	
	create	deployment, deploy	
	edit	daemonSet, ds	
	delete	configMap	
	apply	secret	
	explain	buildconfig	
	login	project	
	...	...	

The OpenShift command line client is named oc. It is a superset of kubectl; this means that you can use all kubectl commands with oc.



kubeconfig, normally at \$HOME/.kube/config

## Client Configuration

```
apiVersion: v1
clusters:
- cluster:
  certificate-authority-data: LS0...
  server: https://api.crc.testing:6443
  name: api-crc-testing:6443
- cluster:
  certificate-authority: <FILE>
  name: minikube
contexts:
- context:
  cluster: api-crc-testing:6443
  namespace: default
  user: kubeadmin
  name: crc-admin
```

```
- context:
  cluster: minikube
  namespace: default
  user: minikube
  name: minikube
current-context: minikube
...
users:
- name: developer
  user:
    token: sha256~jKB-QMX5P5dvaDF...
- name: kubeadmin
  user:
    token: sha256~EuCfSDowf1lW...
- name: minikube
  user:
    client-certificate: <FILE>
    client-key: <FILE>
```

Sept 2021

(c) Bernd Bausch 2021

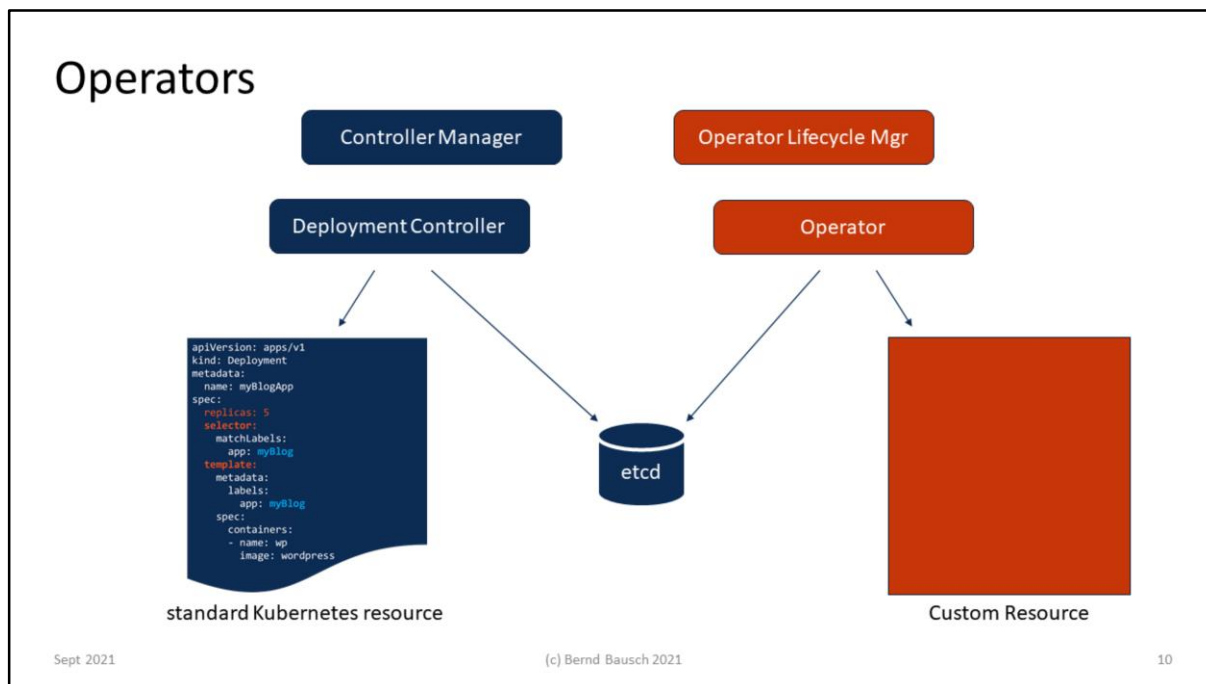
9

Like `kubectl`, `oc` also uses a `kubeconfig` file (`$HOME/.kube/config` by default) to learn about clusters and identities.

A **cluster** is described by its API endpoint (a URL) and an X.509 certificate authority that is used for authentication. In the example, the CRC cluster (OpenShift) includes the entire CA in the config file, whereas the Minikube cluster's CA is stored in a separate crt file.

The **users** section stores users' credentials. The example uses so-called *bearer tokens* to authenticate with the OpenShift cluster and an X.509 certificate for Minikube.

A **context** is essentially a combination of cluster and user account, plus the default Kubernetes *namespace* that will be used for the login session.



## An operator

- is an app-specific controller
- extends the Kubernetes API
- to create/configure/manage complex apps on behalf of a K8s user

An controller runs in a loop, comparing current with desired state. When current state deviates from the desired state, it takes action.

While a normal controller watches over standard Kubernetes resources like pods or deployments, an operator watches over Custom Resources (CR), declared by a Custom Resource Definition (CRD). A CRD lists all possible configurations of a CR.

More precisely, a CR is an **API extension**, or a new API endpoint, and a CRD is a schema for a CR.

An operator may perform tasks like monitoring its application, backing up/recovering data, or updating the application.

"It encodes in software the skills of an expert administrator" (from the O'Reilly book).

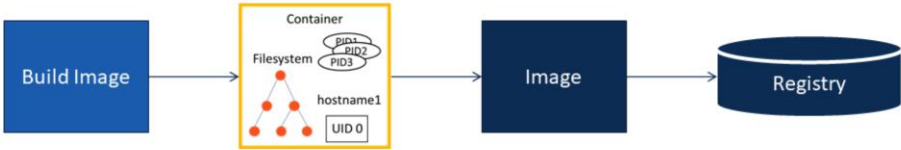
**Example etcd cluster.** When an etcd pod crashes, Kubernetes can launch a new one, but it doesn't know how to synchronize it with the remaining etcd nodes. One could write an etcd-specific operator that has that knowledge.

# Building and Deploying

---

BuildConfig, DeploymentConfig, ImageStream

# Build



- Docker  
Dockerfile, buildah
- Source-to-Image  
S2i image, source code, buildah
- Custom  
Custom build image, privileged users only

# BuildConfig

git push → rebuild  
Must create webhook in Github

Docker, S2i, Custom

ImageStream: Local Registry  
DockerImage: Docker-type Registry

```
kind: BuildConfig
apiVersion: build.openshift.io/v1
metadata:
  name: "ruby-sample-build"
spec:
  runPolicy: "Serial"
  triggers:
    - type: "GitHub"
      github: { secret: "name-of-secret" }
    - type: "Generic"
      generic: { secret: "name-of-secret" }
    - type: "ImageChange"
  source:
    git:
      uri: "https://github.com/openshift/ruby-hello-world"
  strategy:
    sourceStrategy:
      from:
        kind: "ImageStreamTag"
        name: "ruby-20-centos7:latest"
    output:
      to:
        kind: "ImageStreamTag"
        name: "origin-ruby-sample:latest"
  postCommit:
    script: "bundle exec rake test"
```

Sept 2021

(c) Bernd Bausch 2021

13

A BuildConfig is an OpenShift object usually described by a manifest. It describes how a build is carried out. In particular, it defines the build strategy - Docker, S2i, Custom - and strategy-specific parameters such as the build image for S2i. It also determines what happens with the build result and it can define conditions that trigger an automatic rebuild, such as the availability of a new build image version.

Build hooks are tasks that are run at certain points during the build process. The example runs a rake script after committing the result image.

The run policy determines whether several builds can take place in parallel or must be performed serially (the default).