neueda

futureproof your workforce

OPENSHIFT 4.11 CONCEPTS

NEUEDA.COM

Objectives

- OpenShift Concepts
- Kubernetes Namespaces and OpenShift Projects
- Pods
- Labels
- Services
- OpenShift Routes and Kubernetes Ingress Objects
- BuildConfigs
- Deployment Objects
- OpenShift Interfaces

OpenShift Concepts

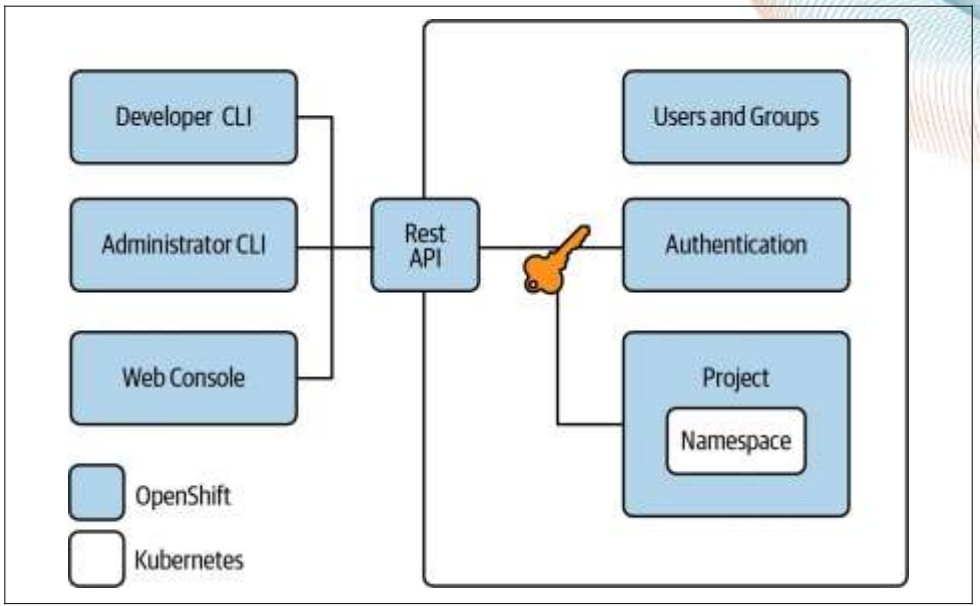
- Anything you can do in Kubernetes you can do in OpenShift
  - OpenShift is built on k8s and runs it under the hood. But there are things in OpenShift you can't do in Kubernetes
  - For example, OpenShift has some CI/CD components that k8s doesn't have by default.
- Now we'll look at some foundational OpenShift concepts.

Kubernetes Namespaces

- Kubernetes has the concept of a *namespace* to partition a cluster into virtual subclusters.
  - For multiple applications, application layers or users.
- Within a namespace, the names of resources must be unique.
- K8s has built in authorization and authentication modes to restrict access to resources
- OpenShift extends this to production-ready functionality with **Projects.**

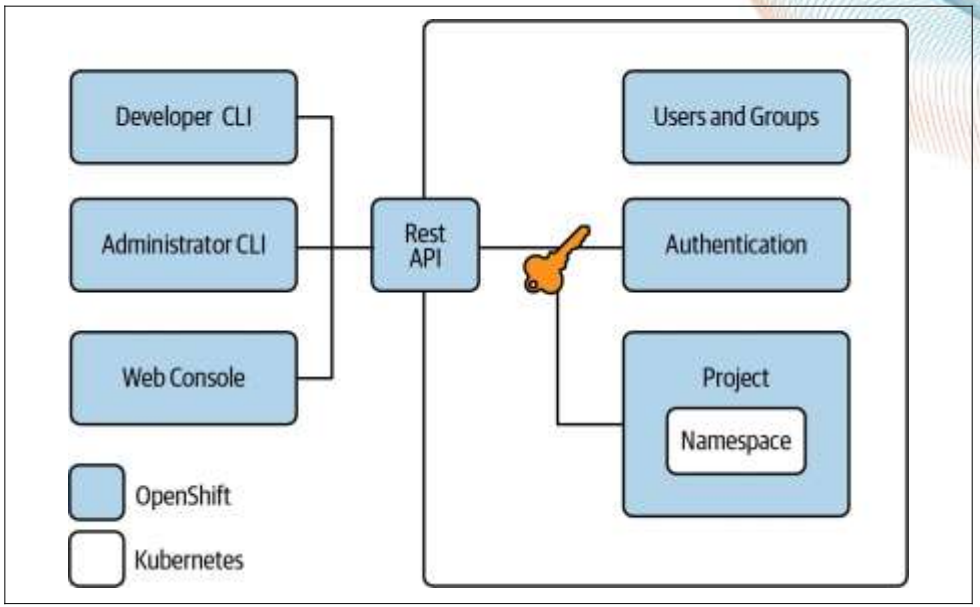
Projects

- OpenShift enforces access control to the cluster and its resources.
- Main model is Role Based Access Control (RBAC)
- RBAC rules define a user and make the user a member of at least one group.



Projects

- Groups encapsulate different levels of access teams or units need to the cluster.
- Projects divide the cluster among teams and applications, enforcing rules to keep them separate.



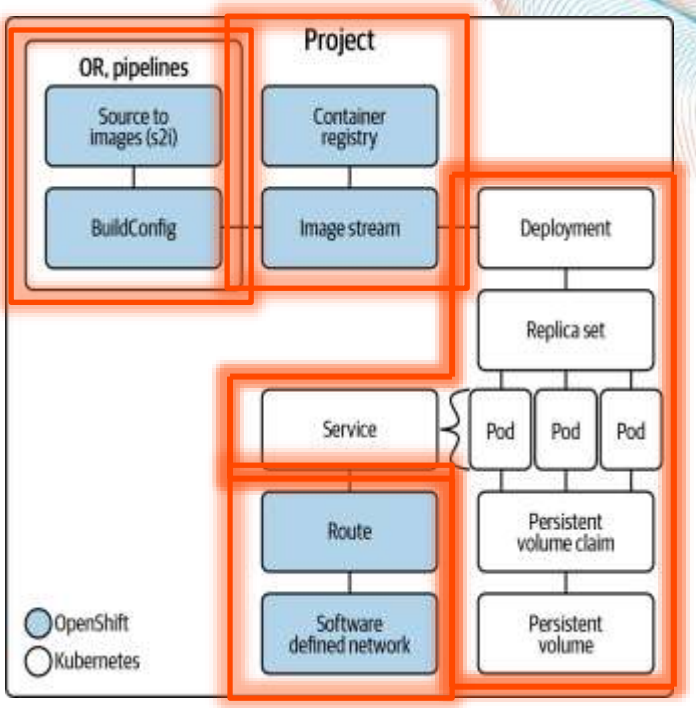
neueda
future proof your workforce

Dividing the Cluster Using Projects

- Cluster administrators are free to use Projects in whatever way suits their organization
  - Create a single Project for each application
  - Create Projects for teams, and the teams can deploy multiple applications in a Project
- OpenShift provides labels for applications, and ways to visualize and sort multi-application projects using labels.

Application Components

- In k8s an application is essentially a cluster work-load.
- In OpenShift this concept is extended to include the whole application workflow
  - Building
  - Containerizing
  - Deploying
  - Making available



Pods

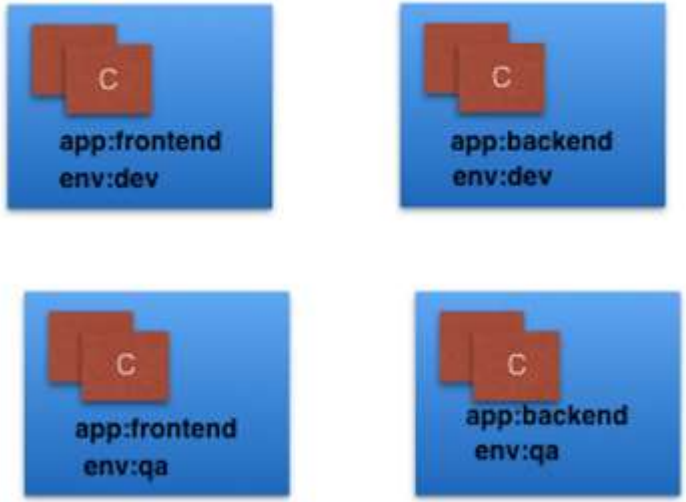
- A **Pod** is the smallest and simplest k8s object.
  - It is the basic unit of running code on the cluster.
- A Pod encapsulates one or more containers which
  - Are scheduled together on the same host with the Pod
  - Share the same network namespace (they share a single IP address originally assigned to the Pod)
  - Have access to mount the persistent storage (volumes)
  - Can communicate with each other over localhost

Pods

- Pods are ephemeral in nature and cannot self-heal
- They are the unit of horizontal scaling - created and destroyed to scale a deployment up or down.
- Copies of Pods are called **Replicas** - they have the same set of containers and config, but their own runtime state.

Labels

- **Labels** are key-value pairs attached to k8s objects.
- Labels are used to organize and select a subset of objects.
- In the example:
  - The Label `env=dev` selects the top two pods.
  - The bottom left is selected by two Labels: `app=frontend AND env=qa`



Services

- We saw that each Pod has its own unique IP within the cluster, and can be reached from anywhere in the cluster.
- But what if you are scaling your application and have multiple replicas of a Pod? Or if the Pods are destroyed and replaced?
- K8s provides a **Service** object to abstract the communication to a dynamic set of replica Pods.
- A Service has an IP address and DNS name in the cluster, and connections to it are routed to the Pods in the set.

OpenShift Routes and Kubernetes Ingress

- Services mean you can access a running application from within the cluster, but the Service IP and DNS are meaningless outside the cluster.
- There are two parallel technologies available to address this problem: **Kubernetes Ingress Objects** and **OpenShift Routes**
- OpenShift Routes was first, and provides a superset of the options available with Ingress Objects
  - Essentially both do the same thing though!

OpenShift Routes

- Creating an OpenShift Route on a service causes OpenShift to configure a DNS name and IP address reachable from an external network.
  - Connections to that IP address are then routed to the service.

BuildConfig

- K8s requires deployment-ready applications (compiled, transpiled etc)
- OpenShift provisions a **BuildConfig** object describing build steps to create a new application container image from source code.
- A BuildConfig can respond to webhooks, triggering automatic builds when (for example) changes are check into Git.
- BuildConfigs are just one CI/CD option available in OpenShift, we will talk about it more later.

Deployment Objects

- Once you have a container image for your application, it needs to be deployed.
- The k8s **Deployment** object defines the template from which new Pods are created, and the rules for rollouts and rollbacks of updates.
- A Deployment usually represents a single service or application component.
- There is also a **DeploymentConfig** object, which we shall look at later.

OpenShift Permissions

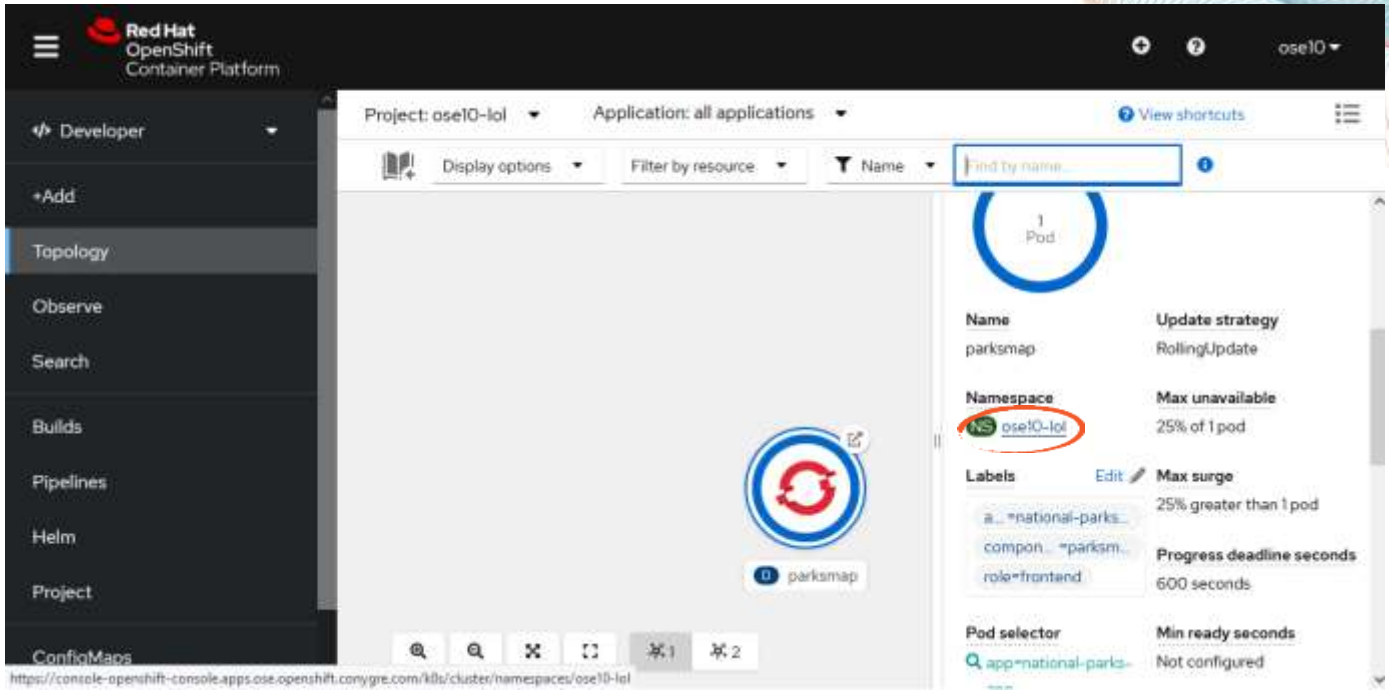
- OpenShift is a declarative platform.
  - A lot of things happen in the cluster that are not explicitly initiated by the end user.
- OpenShift also requires that appropriate authentications and authorizations accompany every action that occurs on the cluster.
- For the user this is all about having the right RoleBindings in place.
- But what about actions not initiated by the end user?

OpenShift Permissions: Service Account user

- Internal cluster activities are carried out by a special user called a **Service Account**.
- OpenShift automatically creates a few special service accounts in every project.
- The **default** service account is the one taking responsibility for running the pods.
- OpenShift injects this service account into every pod that is launched.

Service Account Permissions

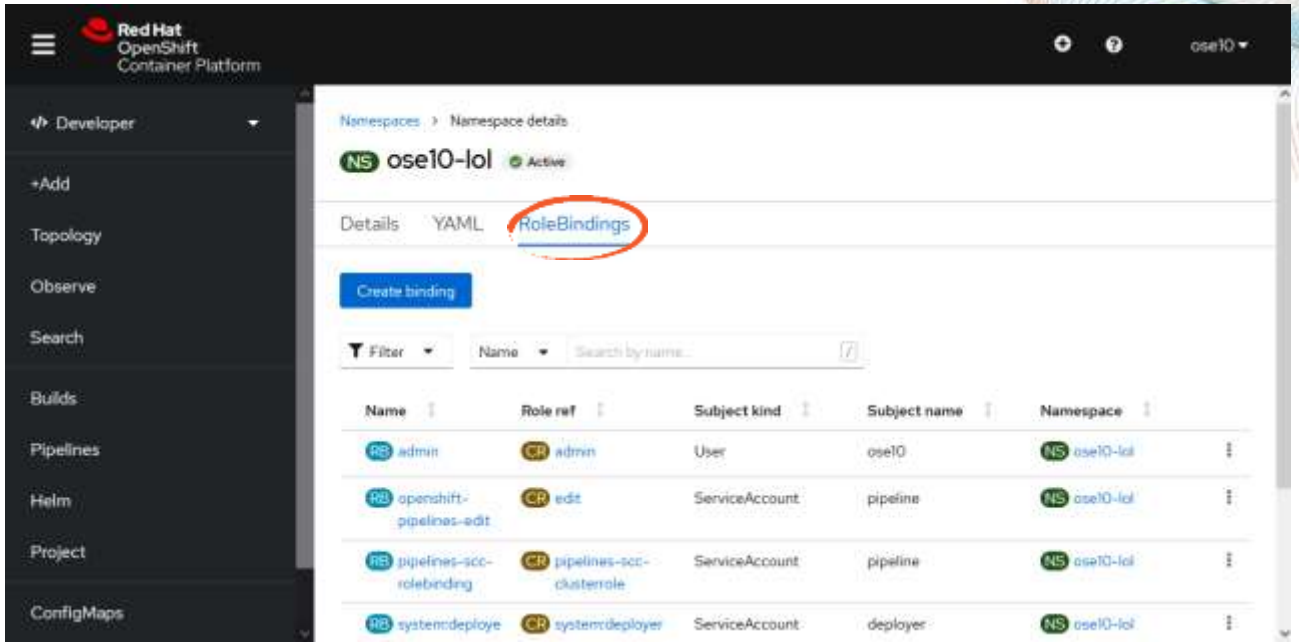
- Click on the Namespace link in the details panel of your app.



neueda
future proof your workforce

Service Account Permissions

- Click on the RoleBindings tab to see the current permissions.



neueda
futureproof your workforce

When things Go Wrong

- Service Account permissions mean you can do interesting things with your deployments.
- However they can also be the source of interesting bugs!
- For example, in a deployment where you wish apps to discover each other automatically, the default service account needs to have the view RoleBinding set.
  - This is <u>not</u> set by default!

Main OpenShift Interfaces

oc Command Line

- Strict superset of `kubectl`
- Also understands all the OpenShift specific functionality

Web Console

- Fully functioned GUI for deploying, managing and monitoring your applications

API

- If you want to integrate OpenShift with external systems (e.g. container building process)

Summary

- OpenShift Concepts
- Kubernetes Namespaces and OpenShift Projects
- Pods
- Labels
- Services
- OpenShift Routes and Kubernetes Ingress Objects
- BuildConfigs
- Deployment Objects
- OpenShift Interfaces

Questions and Comments?

neueda
futureproof your workforce