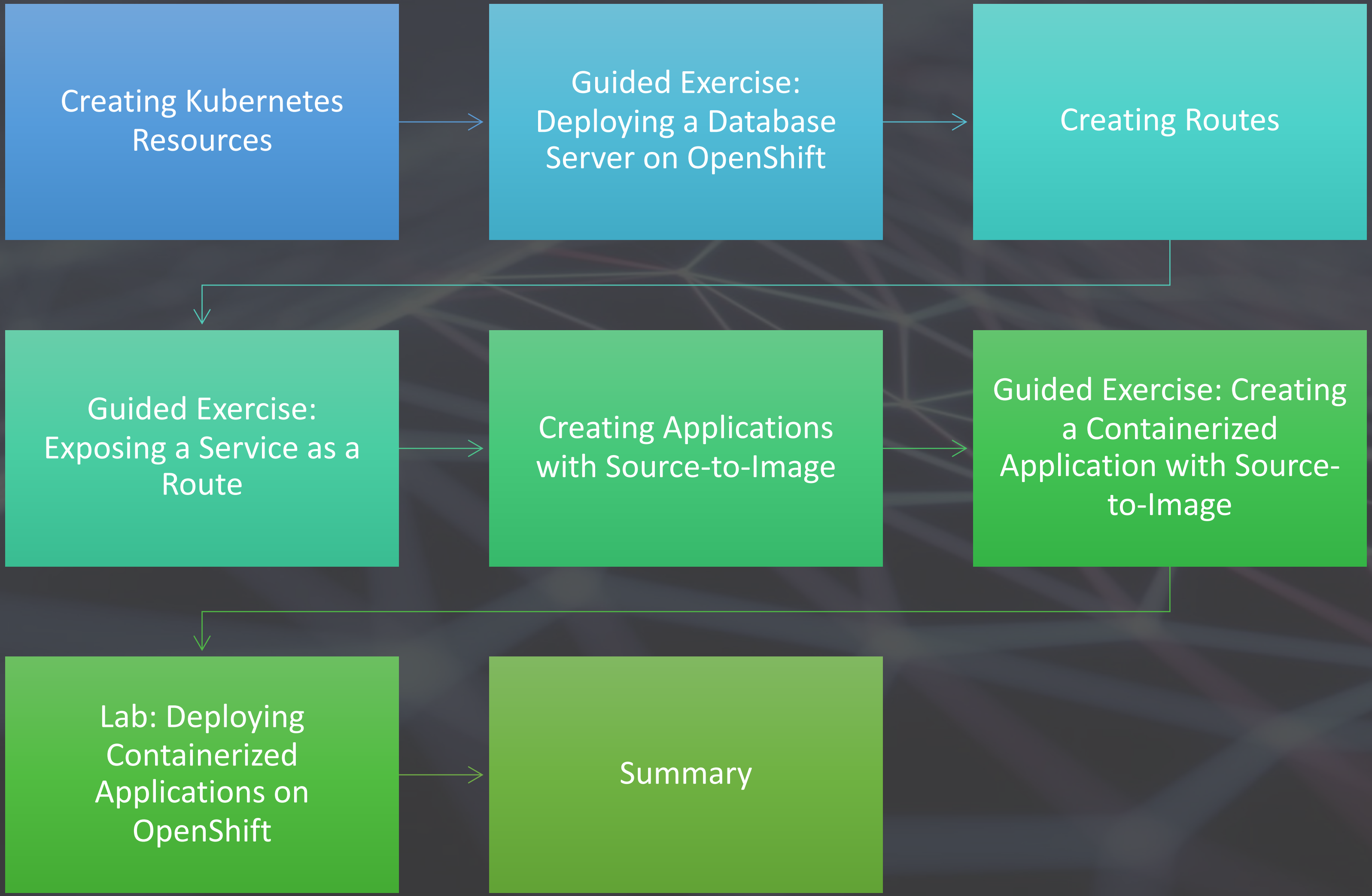
Describing K8s and OpenShift Architecture

Chapter objectives



Creating Kubernetes Resources

Guided Exercise: Deploying a Database Server on OpenShift

Creating Routes

Guided Exercise: Exposing a Service as a Route

Creating Applications with Source-to-Image

Guided Exercise: Creating a Containerized Application with Source-to-Image

Lab: Deploying Containerized Applications on OpenShift

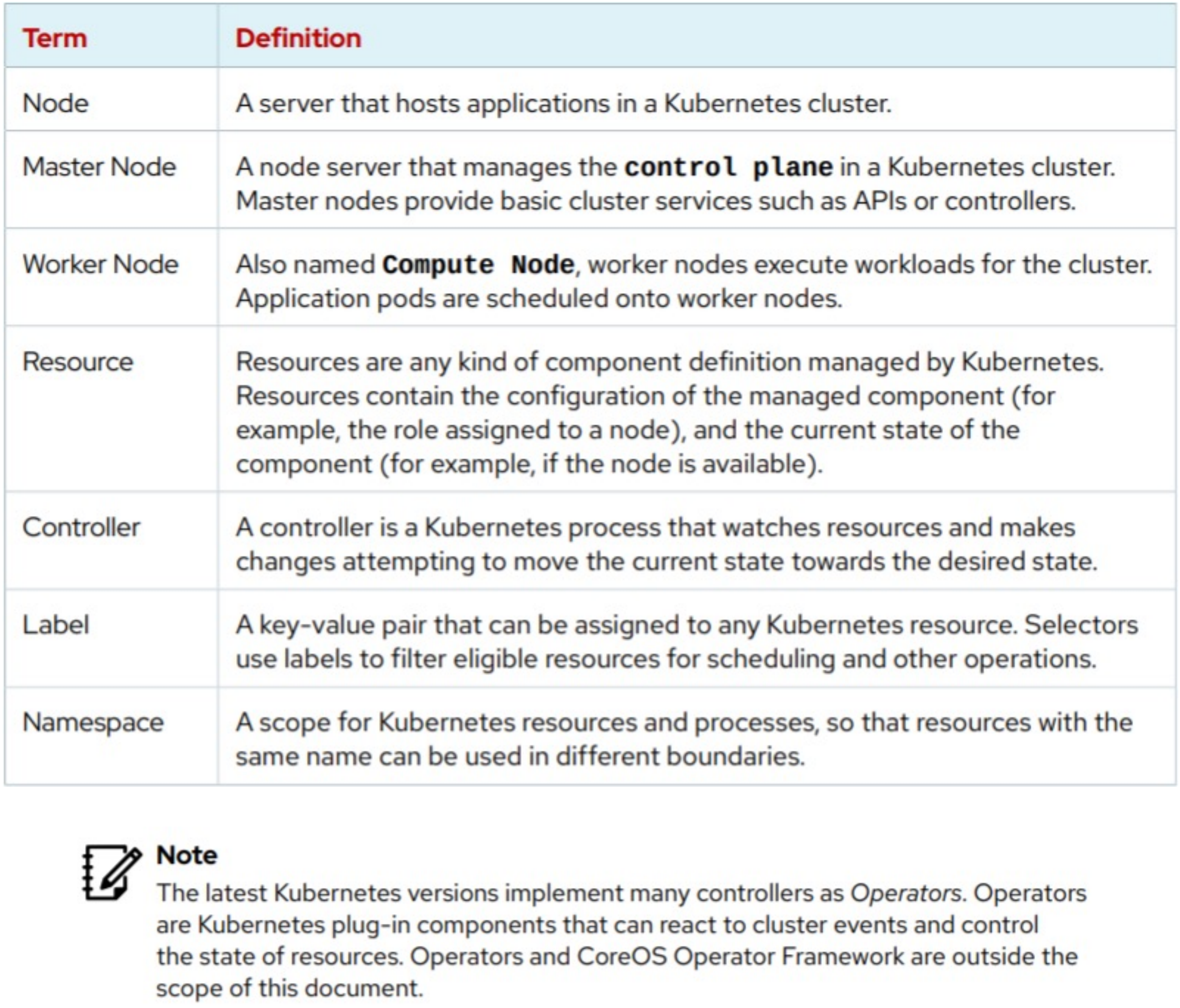
Summary

Describing Kubernetes and OpenShift Architecture

After completing this section, you will be:

- Describe the architecture of a Kubernetes cluster running on the Red Hat OpenShift Container Platform (RHOCP)
- List the main resource types provided by Kubernetes and RHOCP
- Identify the network characteristics of containers, Kubernetes, and RHOCP
- List mechanisms to make a pod externally available

Kubernetes Terminology

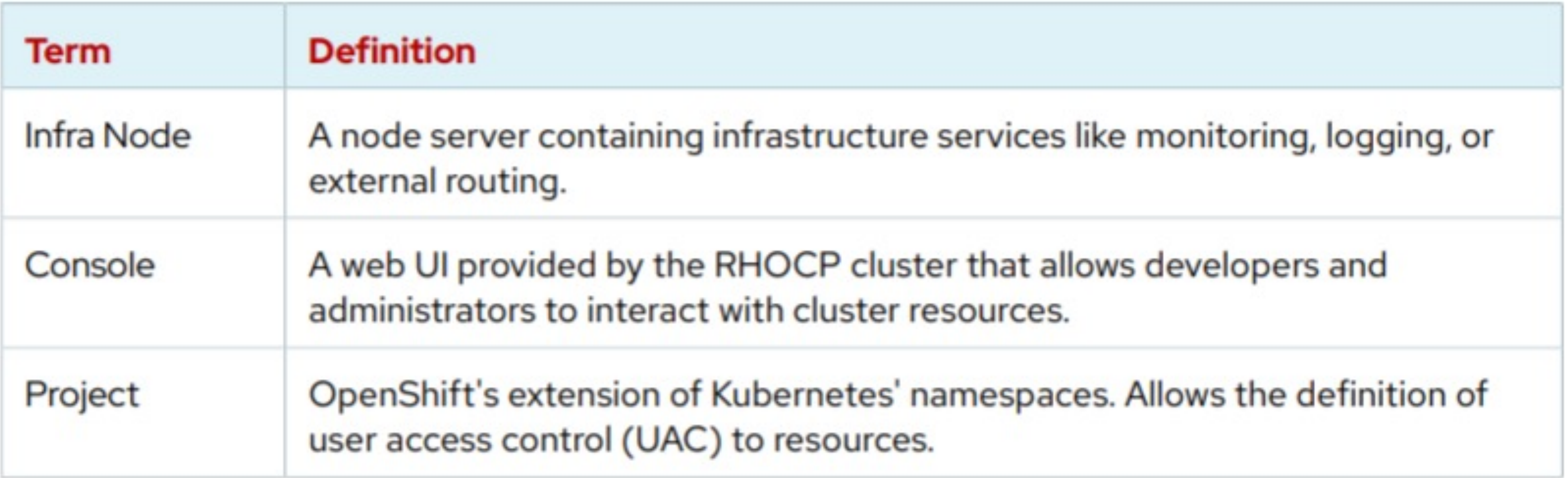


Term	Definition
Node	A server that hosts applications in a Kubernetes cluster.
Master Node	A node server that manages the **`control plane`** in a Kubernetes cluster. Master nodes provide basic cluster services such as APIs or controllers.
Worker Node	Also named **`Compute Node`**, worker nodes execute workloads for the cluster. Application pods are scheduled onto worker nodes.
Resource	Resources are any kind of component definition managed by Kubernetes. Resources contain the configuration of the managed component (for example, the role assigned to a node), and the current state of the component (for example, if the node is available).
Controller	A controller is a Kubernetes process that watches resources and makes changes attempting to move the current state towards the desired state.
Label	A key-value pair that can be assigned to any Kubernetes resource. Selectors use labels to filter eligible resources for scheduling and other operations.
Namespace	A scope for Kubernetes resources and processes, so that resources with the same name can be used in different boundaries.

Note

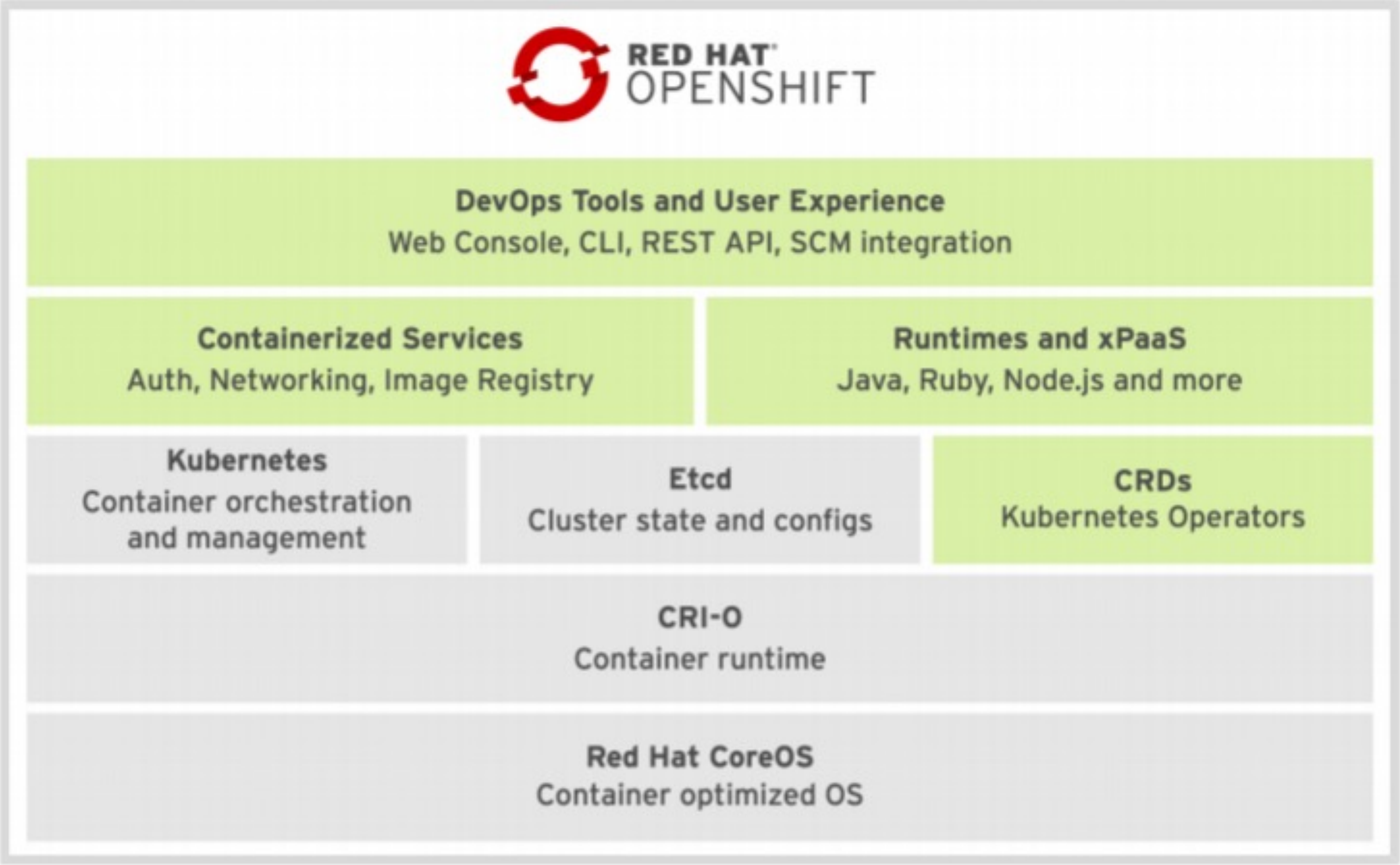
The latest Kubernetes versions implement many controllers as *Operators*. Operators are Kubernetes plug-in components that can react to cluster events and control the state of resources. Operators and CoreOS Operator Framework are outside the scope of this document.

OpenShift Terminology



Term	Definition
Infra Node	A node server containing infrastructure services like monitoring, logging, or external routing.
Console	A web UI provided by the RHOCP cluster that allows developers and administrators to interact with cluster resources.
Project	OpenShift's extension of Kubernetes' namespaces. Allows the definition of user access control (UAC) to resources.

OpenShift Component Stack



RED HAT OPENSHIFT

DevOps Tools and User Experience
Web Console, CLI, REST API, SCM integration

Containerized Services
Auth, Networking, Image Registry

Runtimes and xPaaS
Java, Ruby, Node.js and more

Kubernetes
Container orchestration and management

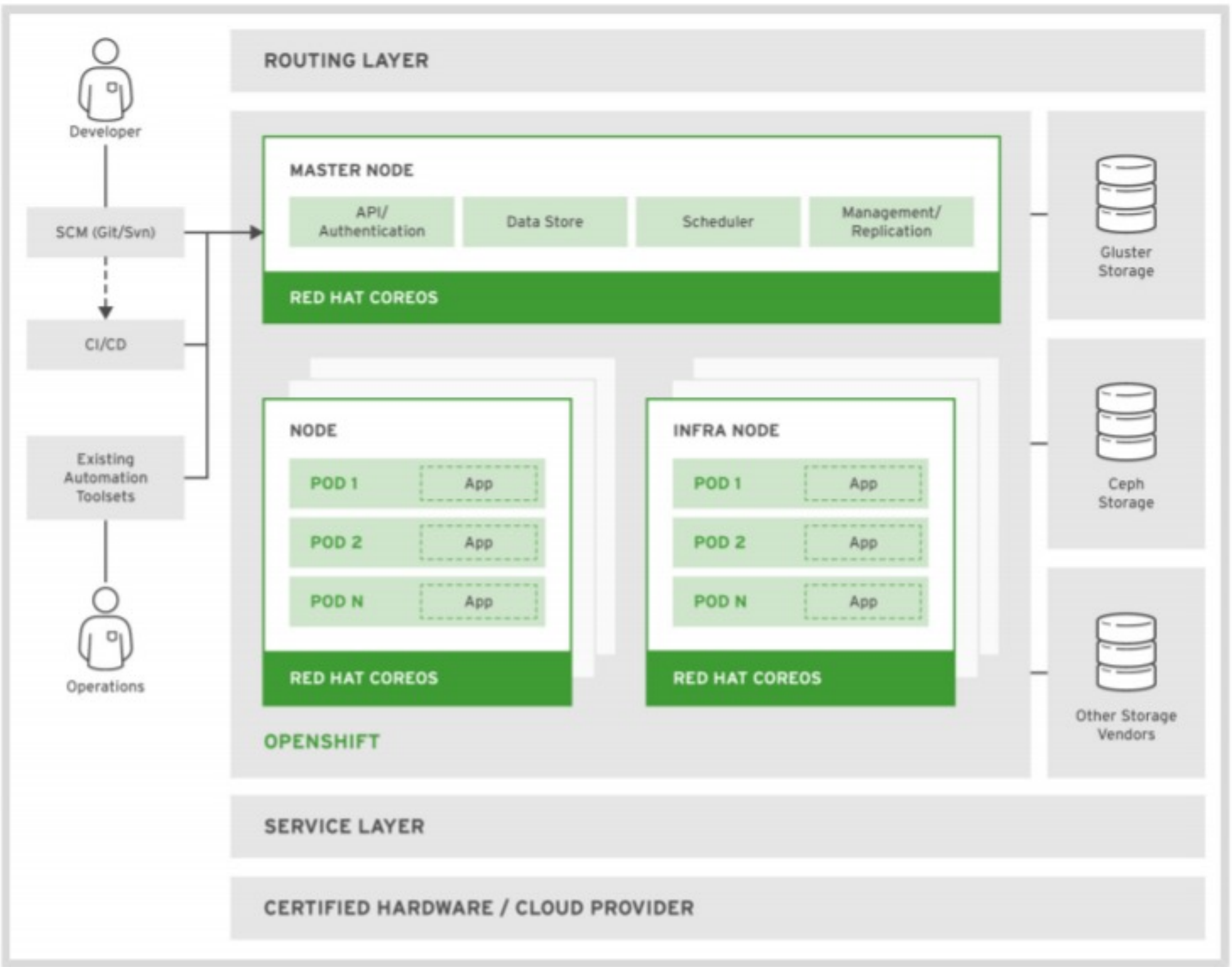
Etcd
Cluster state and configs

CRDs
Kubernetes Operators

CRI-O
Container runtime

Red Hat CoreOS
Container optimized OS

OpenShift and K8s Architecture



New Features in RHOCP v4

- CoreOS as mandatory operating system for all nodes
- Brand new cluster installer
- Self-managing platform
- Automatic cluster updates and recoveries
- Re-designed application life-cycle management
- Operator SDK: to build, test and package Operators

Describing K8s Resource Types

Pods (po)

Collection of containers that share resources. Basic unit of work for k8s

Services (svc)

Provide stable interface to all pods in deployment. By default, services connect clients to pods in round-robin fashion

Replication Controllers (rc)

A k8s resource that defines how pods are replicated and scheduled onto nodes. Basic k8s service to provide HA for pods and containers

Describing K8s Resource Types - continue

Persistent Volume (pv)

Provide permanent storage to pods

Persistent Volume Claims (pvc)

PVCs links PV to pod so that containers can mount storage to container’s file system

ConfigMaps (cm) and Secrets

Contains set of keys values pair that can be used by resources.

Provides centralized configuration values used by resources.

Secrets values are always encoded and restricted to authorized users only

Describing OpenShift Resource Types

Deployment config(dc)

Like deployment, represents set of containers included in pod. Specifies deployment strategies to be used. A dc also provides basic but extensible continuous delivery workflow

Build Config (bc)

Defines process to be executed in project.

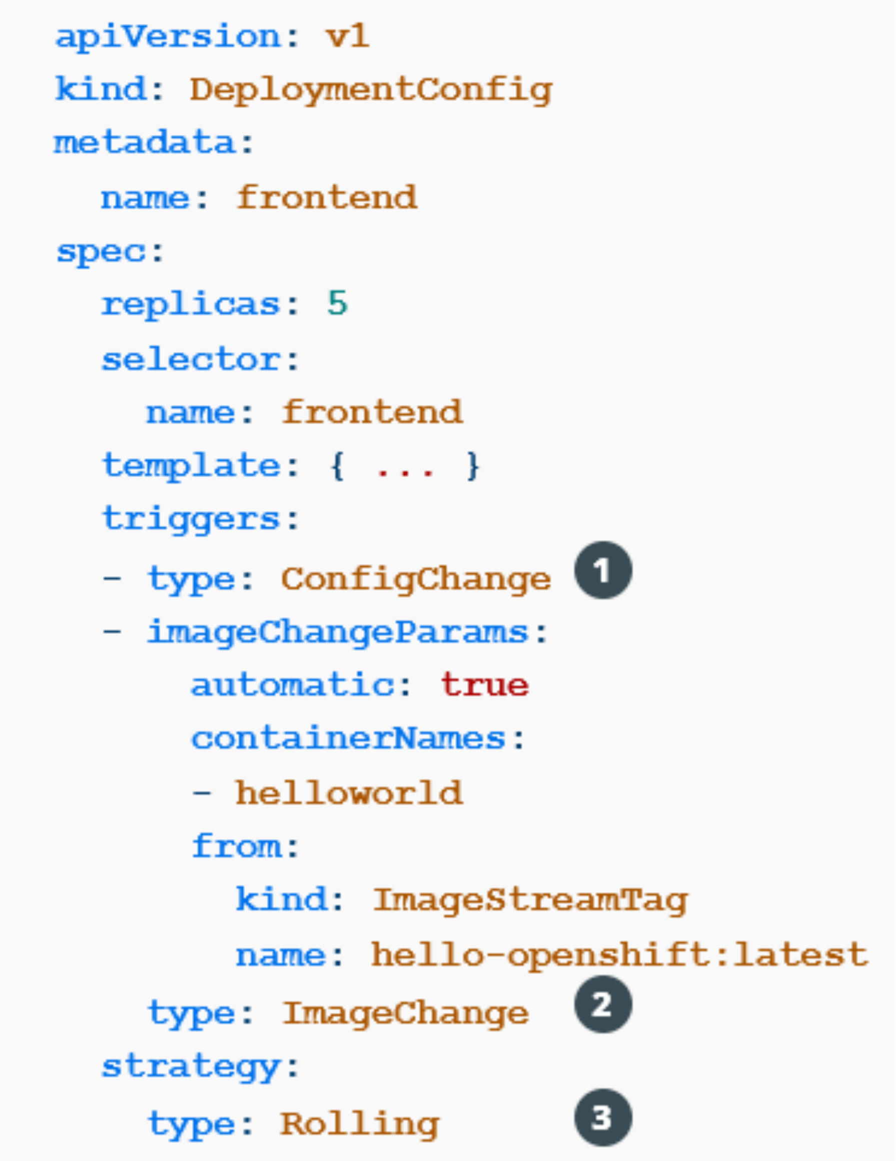
Used by S2I feature to build container image from application source code stored in Git repo.

A bc works with dc to provide basic but extensible continuous integration and continuous delivery workflow

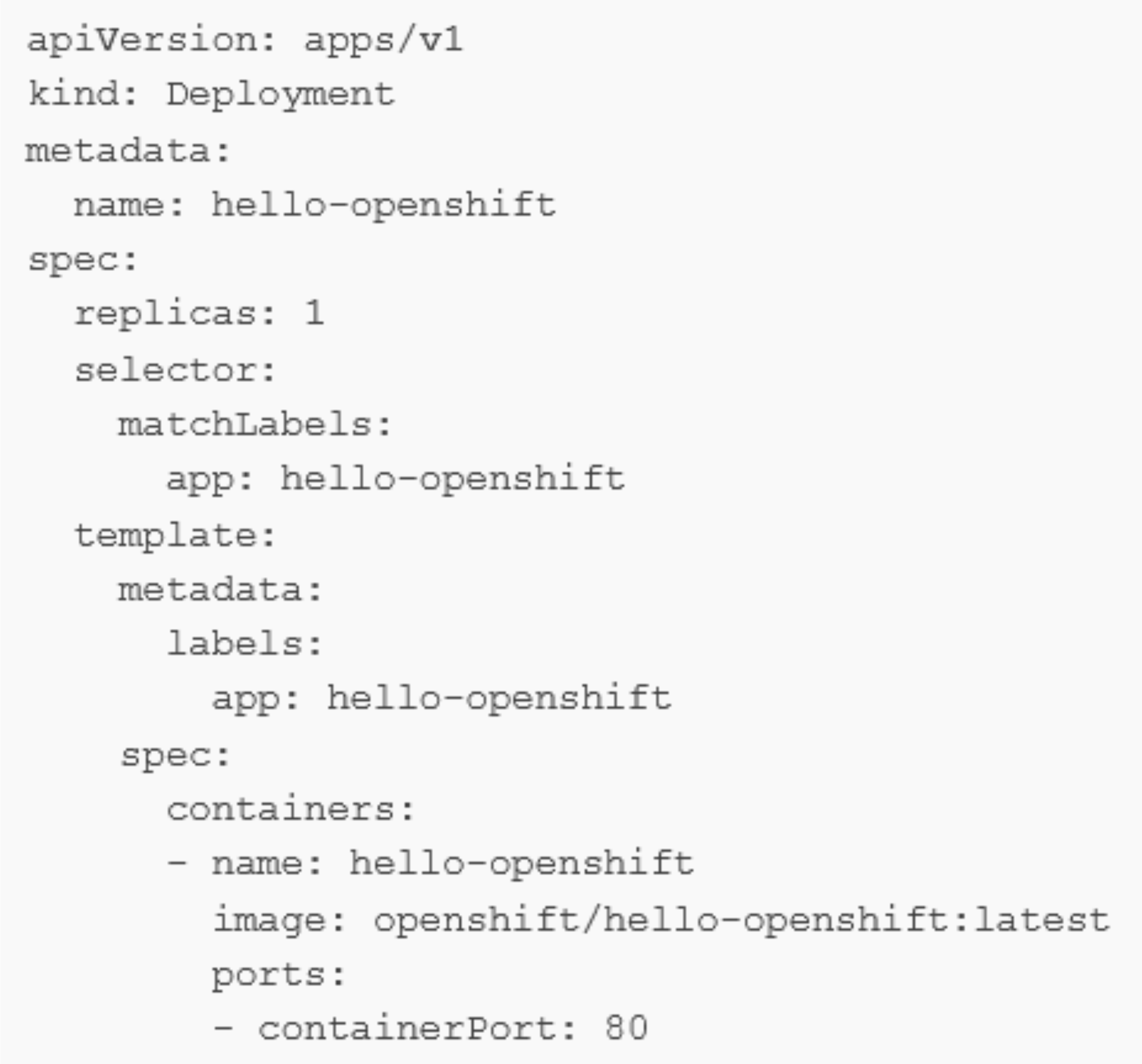
Routes

Provides fully-qualified domain name recognized by OpenShift router as an ingress point for application and microservices

Deployment vs DeploymentConfig



```
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
```



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-openshift
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello-openshift
  template:
    metadata:
      labels:
        app: hello-openshift
    spec:
      containers:
      - name: hello-openshift
        image: openshift/hello-openshift:latest
        ports:
        - containerPort: 80
```

DC add enhancement to deployment.

- ***Replication Controllers***: such as ability to involve one or more replication controllers, using pod template in controlling point-in-time record state of pods
- ***Triggers***: add on triggers that drive automated deployments in response to events (Image change or Config change)
- ***Custom strategies***: user-customizeble deployment strategies to transition from previous version to new version.
- ***Lifecycle hooks***: Uses hooks or lifecycle hooks for executing custom behaviour after pods are being built
- ***Automatic rollbacks*** : versioning of app in order to support rollbacks either automatically or manually in case of failure during deployment

Networking

Container IP

Ephemeral IP address

Assigned from internal network that is accessible only from node running the container

Software-Defined network (SDN)

Provides communication between container in pods between nodes

Access to SDN only works from inside same Kubernetes cluster

Networking - continue

Services

Containers do not connect each other dynamic IP address directly

Uses services by linking more stable IP addresses from SDN to pods

Pods restarted, replicated, rescheduled to different nodes – services get updated, providing scalability and high availability

External Access / K8s Ingress

Is more complicated.

K8s uses NodePort attribute to provide external access. But it is insecured and doesn’t scale well

Networking - continue

OpenShift Routes - External Access

Simpler by defining route resources

Route defines external-facing DNS names and ports for service

A router (ingress controller) forwards HTTP(s) requests to service addresses inside K8s SDN.

OpenShift map IP addresses of RHOCP router nodes

Features	K8s Ingress	OpenShift Route
Standard Kubernetes object	X	
External access to services	X	X
Persistent (sticky) sessions	X	X
Load-balancing strategies (e.g. round robin)	X	X
Rate-limit and throttling	X	X
IP whitelisting	X	X
TLS edge termination for improved security	X	X
TLS re-encryption for improved security		X
TLS passthrough for improved security		X
Multiple weighted backends (split traffic)		X
Generated pattern-based hostnames		X
Wildcard domains		X

Quiz 1

Which two sentences are correct regarding Kubernetes architecture? (Choose two)

a) Kubernetes nodes can be managed without a master

b) Kubernetes masters schedule pods to specific nodes

c) Kubernetes tools cannot be used to manage resources in an OpenShift cluster

d) Kubernetes masters manage pod scaling

e) Containers created from Kubernetes pods cannot be managed using standalone tools such as Podman

Quiz 1

Which two sentences are correct regarding Kubernetes architecture? (Choose two)

a) Kubernetes nodes can be managed without a master

b) Kubernetes masters schedule pods to specific nodes

c) Kubernetes tools cannot be used to manage resources in an OpenShift cluster

d) Kubernetes masters manage pod scaling

e) Containers created from Kubernetes pods cannot be managed using standalone tools such as Podman

Quiz 2

Which two sentences are correct regarding Kubernetes and OpenShift resource types? (Choose two)

a) A pod is responsible for provisioning its own persistent storage

b) All pods generated from the same replication controller have to run in the same node

c) A route is responsible for providing IP addresses for external access to pods

d) A replication controller is responsible for monitoring and maintaining the number of pods for a particular application

Quiz 2

Which two sentences are correct regarding Kubernetes and OpenShift resource types? (Choose two)

a) A pod is responsible for provisioning its own persistent storage

b) All pods generated from the same replication controller have to run in the same node

c) A route is responsible for providing IP addresses for external access to pods

d) A replication controller is responsible for monitoring and maintaining the number of pods for a particular application

Quiz 3

Which two statements are true regarding Kubernetes and OpenShift networking? (Choose two)

a) A Kubernetes service can provide an IP address to access a set of pods

b) Kubernetes is responsible for providing a fully qualified domain name for a pod

c) A replication controller is responsible for routing external requests to the pods

d) A route is responsible for providing DNS names for external access

Quiz 3

Which two statements are true regarding Kubernetes and OpenShift networking? (Choose two)

a) A Kubernetes service can provide an IP address to access a set of pods

b) Kubernetes is responsible for providing a fully qualified domain name for a pod

c) A replication controller is responsible for routing external requests to the pods

d) A route is responsible for providing DNS names for external access

Quiz 4

Which statement is correct regarding persistent storage in Kubernetes and OpenShift?

a) A PVC represents a storage area that a pod can use to store data and is provisioned by the application developer

b) A PVC represents a storage area that can be requested by a pod to store data but is provisioned by the cluster administrator

c) A PVC represents the amount of memory that can be allocated to a node, so that a developer can state how much memory he requires for his application to run

d) A PVC represents the number of CPU processing units that can be allocated to pod

Quiz 4

Which statement is correct regarding persistent storage in Kubernetes and OpenShift?

a) A PVC represents a storage area that a pod can use to store data and is provisioned by the application developer

b) A PVC represents a storage area that can be requested by a pod to store data but is provisioned by the cluster administrator

c) A PVC represents the amount of memory that can be allocated to a node, so that a developer can state how much memory he requires for his application to run

d) A PVC represents the number of CPU processing units that can be allocated to pod

Quiz 5

Which statement is correct regarding OpenShift additions to Kubernetes?

a) OpenShift adds features to simplify Kubernetes configuration of many real-world use cases

b) Container images created for OpenShift cannot be used with plain Kubernetes

c) Red Hat maintains forked versions of Kubernetes internal to the RHOCP product

d) Doing continuous integration and continuous deployment with RHOCP requires external tools

Quiz 5

Which statement is correct regarding OpenShift additions to Kubernetes?

a) OpenShift adds features to simplify Kubernetes configuration of many real-world use cases

b) Container images created for OpenShift cannot be used with plain Kubernetes

c) Red Hat maintains forked versions of Kubernetes internal to the RHOCP product

d) Doing continuous integration and continuous deployment with RHOCP requires external tools

Creating Kubernetes Resources

After completing this section, you will be:

- Able to create standard K8s resources

The OpenShift Command

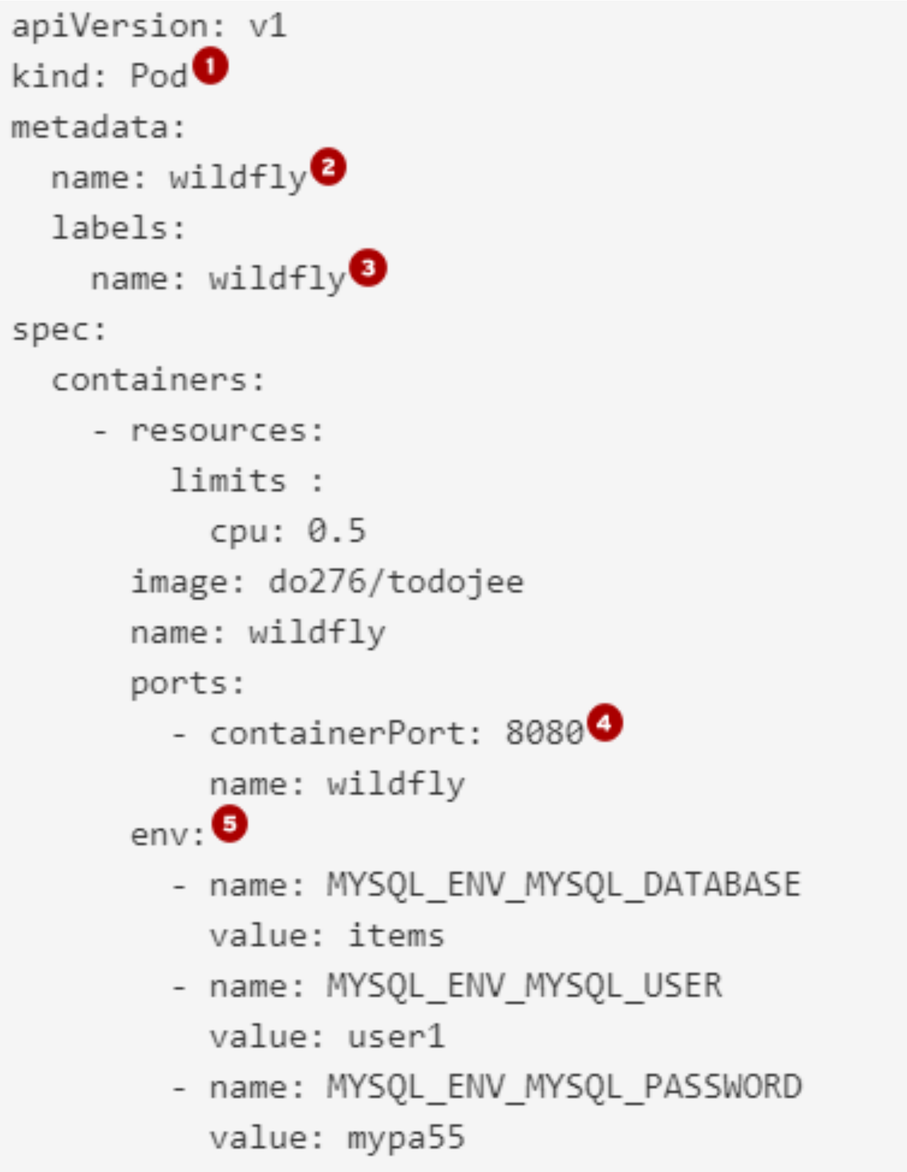
- Main method to interact with RHOCP cluster

```
$ oc <command> --parameters …
```

- Requires logged-in user to cluster

```
$ oc login –u <username> -p <password> <cluster-api-Url>
```

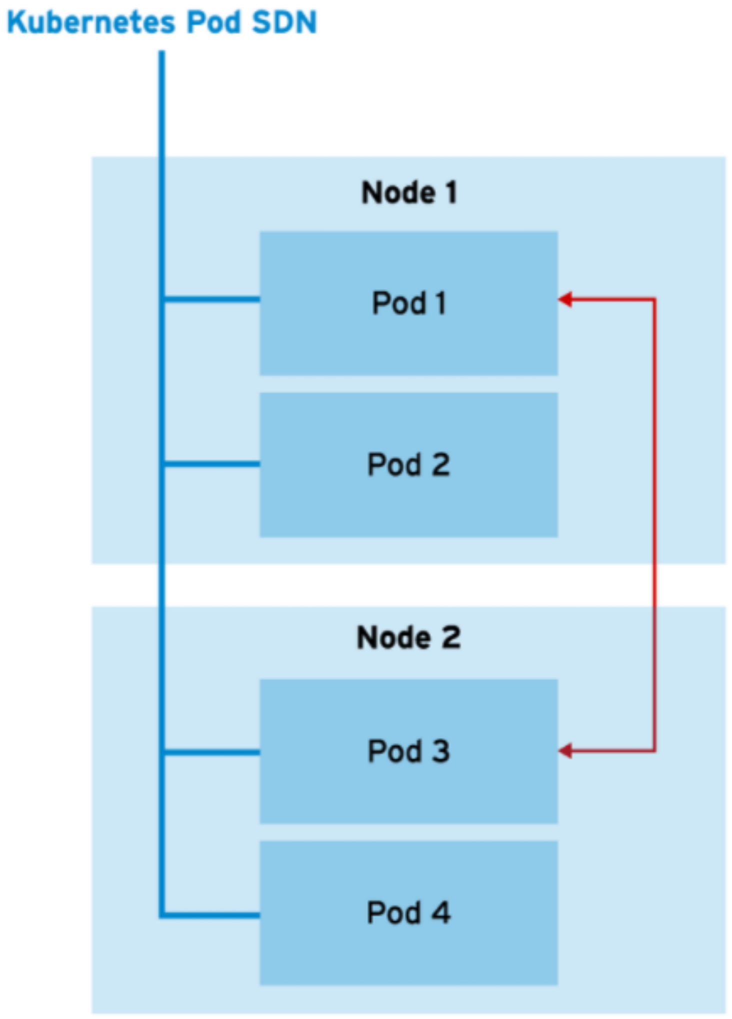
Describing Pod Resource Definition Syntax



```
apiVersion: v1
kind: Pod❶
metadata:
  name: wildfly❷
  labels:
    name: wildfly❸
spec:
  containers:
    - resources:
        limits :
          cpu: 0.5
      image: do276/todojee
      name: wildfly
      ports:
        - containerPort: 8080❹
          name: wildfly
      env:❺
        - name: MYSQL_ENV_MYSQL_DATABASE
          value: items
        - name: MYSQL_ENV_MYSQL_USER
          value: user1
        - name: MYSQL_ENV_MYSQL_PASSWORD
          value: mypa55
```

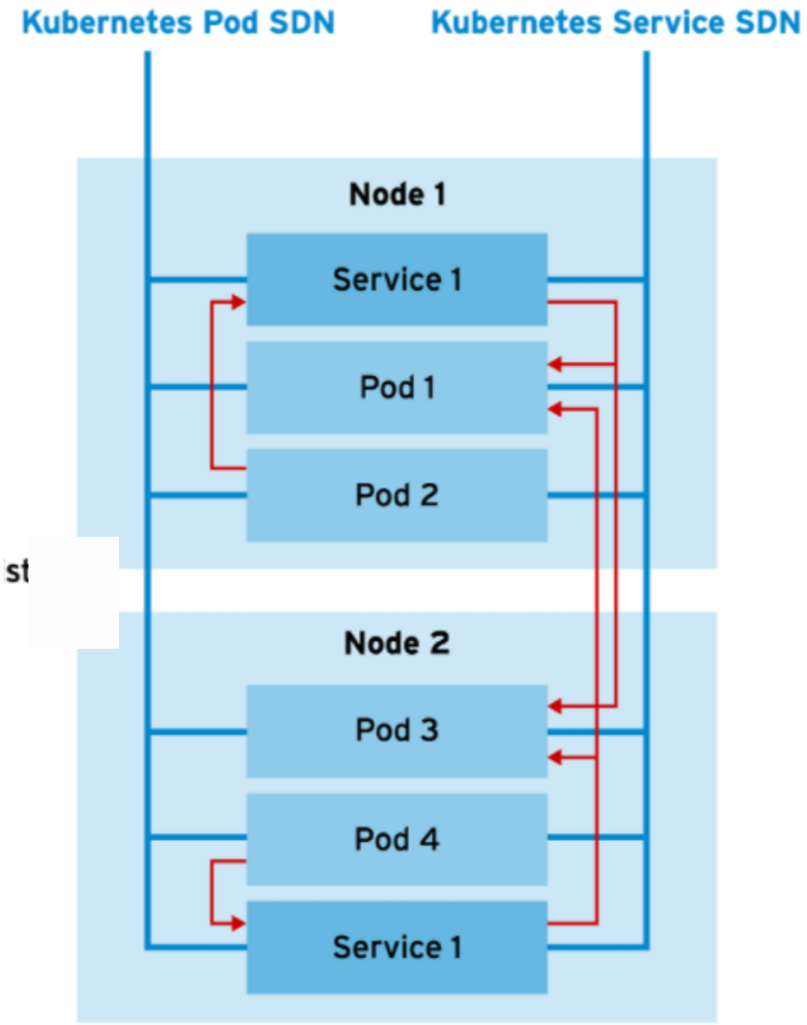
1. Declares a Kubernetes pod resource type.
2. A unique name for a pod in Kubernetes that allows administrators to run commands on it.
3. Creates a label with a key value pair
4. A container-dependent attribute identifying which port on the container is exposed.
5. Defines a collection of environment variables.

SDN exists on each node.



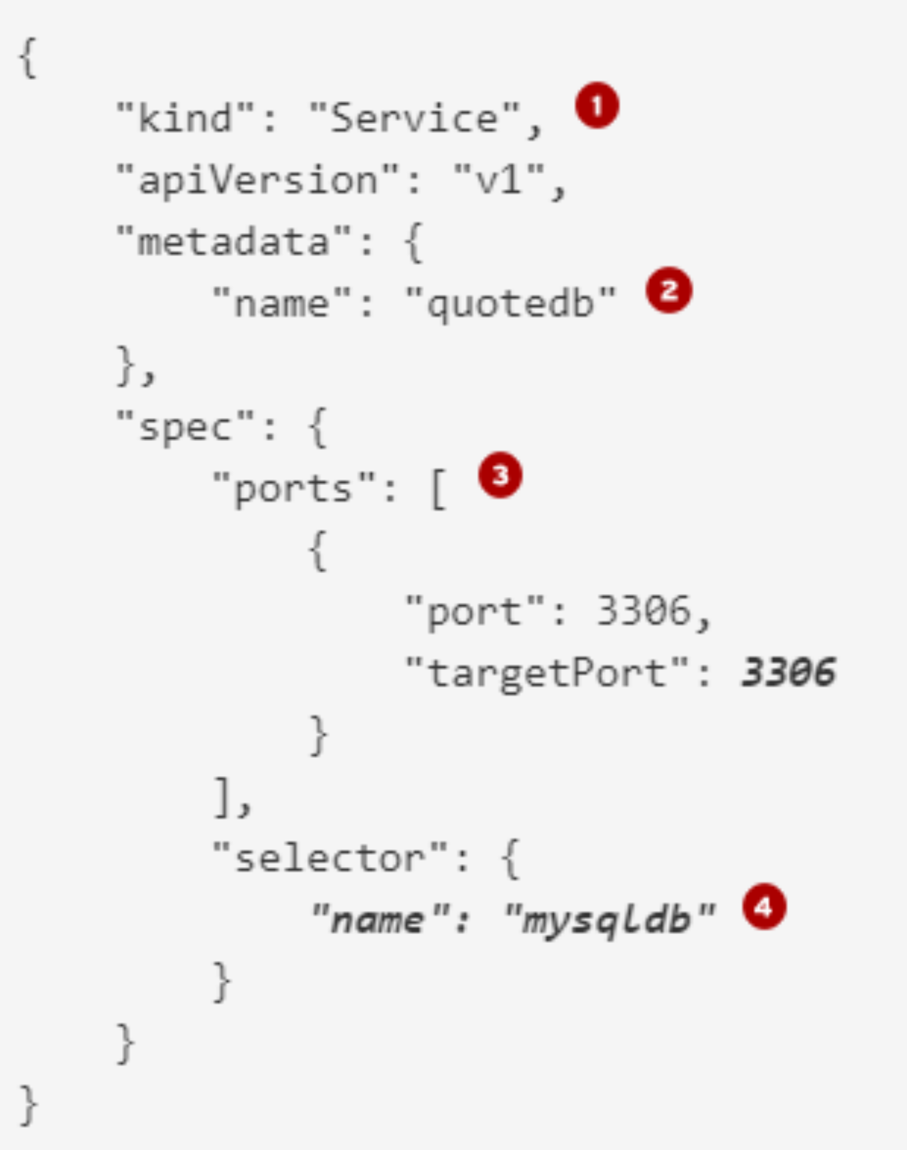
- Each node hosts own virtual network
- OS Kernel provides IP address to Nodes
- CNI provides IP address to Pods
- Pods from same node able to communicate
- Pods from different node can’t communicate
- Nodes’ subnet will always differs from Pods’ subnet

Services resource type



- Provides a stable source IP address
- Each pod matching selector is added to services as endpoint
- Can load balance incoming traffic
- Expose service to get FQDN route to Pods

Describing Service Resource Definition Syntax



```
{
    "kind": "Service",
    "apiVersion": "v1",
    "metadata": {
        "name": "quotedb"
    },
    "spec": {
        "ports": [
            {
                "port": 3306,
                "targetPort": 3306
            }
        ],
        "selector": {
            "name": "mysqldb"
        }
    }
}
```

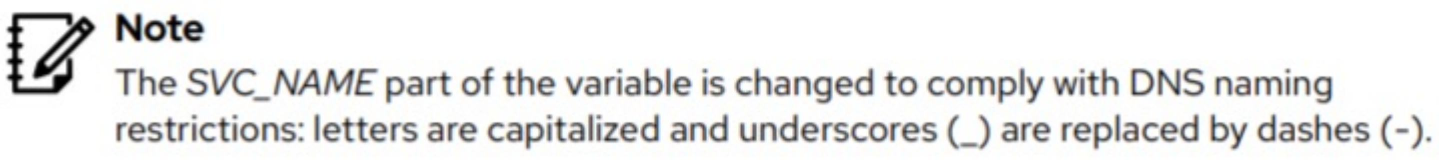
1. The kind of Kubernetes resource. In this case, a Service.
2. A unique name for the service.
3. ports is an array of objects that describes network ports exposed by the service.
4. service port and the service forwards packets to the pod targetPort.
5. selector is how the service finds pods to forward packets to.

Discovering Services

- Pods finds a service IP address and port using environment variables.

- Openshift automatically inject into containers for all pods inside same project
  - SVC_NAME_SERVICE_HOST is the service IP address.
  - SVC_NAME_SERVICE_PORT is the service TCP port.

- OpenShift internal DNS server
  - SVC_NAME.PROJECT_NAME.svc.cluster.local



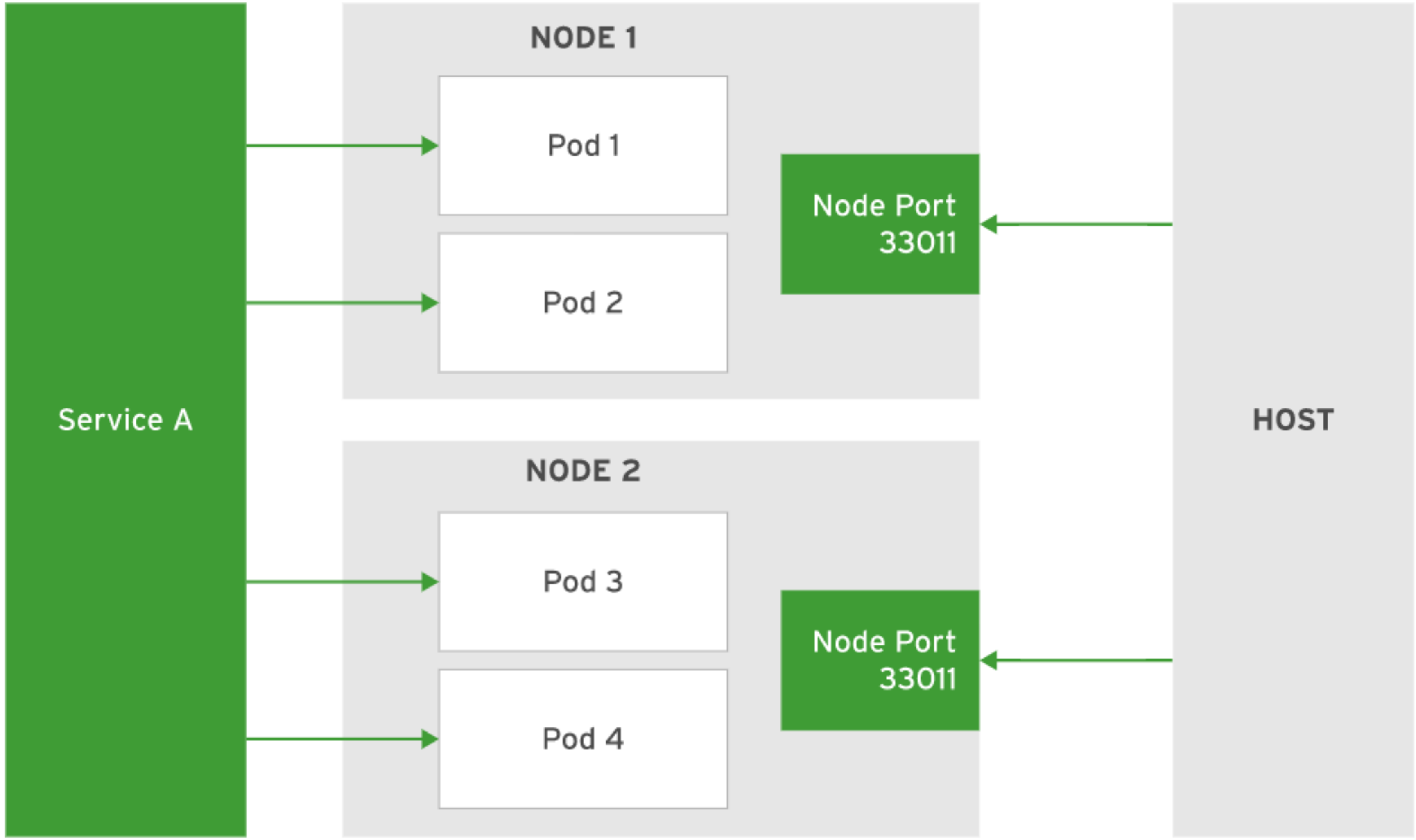
Note

The *SVC_NAME* part of the variable is changed to comply with DNS naming restrictions: letters are capitalized and underscores (_) are replaced by dashes (-).

Providing access to service from outside

- NodePort type:
  - Bind worker node port to service port
  - Use *oc edit svc* subcommand and specify NodePort as type
  - Insecured and inefficient
  - Network Port : 30000 - 37000
- OpenShift Routes:
  - Preferred approach
  - Expose services create unique URL (route)
  - Use *oc expose* subcommand or OpenShift web UI

Use NodePort type:



Use oc *port-forward* subcommand

- Mapping forwards connections to single pod.
- Different from having access via service resource
- Lack of many features offered by service resource
- Mapping exists only on workstation where oc is executed.
- For testing, troubleshooting, diagnose networking access

```
[student@workstation ~] $ oc port-forward mysql-openshift-1-glqrp 3306:3306
```

Compares to *oc expose services* subcommand

- Mappings exists for all network users
- Load-balances connections to potentially multiple pods

Creating New Applications

- Use *oc create* subcommand
  - Create resources by definition file or stdin JSON/YAML

- Use *oc apply* subcommand
  - Merge existing resources definition with stdin JSON/YAML
  - Create resources if it doesn’t exists

- Use *oc new-app* command
  - Native to OpenShift
  - More dynamic
  - Able to use different method:
    - docker-image
    - Source-code (local or remote Git repository)
    - Template

Use oc new-app subcommand - Example

- Create an application based on a image from quay.io:

```
$ oc new-app --name app1 --docker-image quay.io/jason.wong76/webserver

$ oc status
```

- Create an application using source code from github:

```
$ oc new-app --name app2 ruby~https://github.com/sclorg/ruby-ex.git --as-deployment-config

$ oc status
```

- Use openshift-template:

```
$ oc new-app --name app3 --template=mysql-ephemeral –n openshift

$ oc status
```

Use oc new-app subcommand - Example

- Create an application based on a mysql from Docker Hub with label and application’s parameters

```
$ oc new-app --name mydb --as-deployment-config \
     -e MYSQL_USER=redhat –e MYSQL_PASSWORD=redhat \
     -e MYSQL_ROOT_PASSWORD=redhat –e MYSQL_DATABASE=datadb \
     -l db=hrapp1 mysql
```

-e application_variable = data

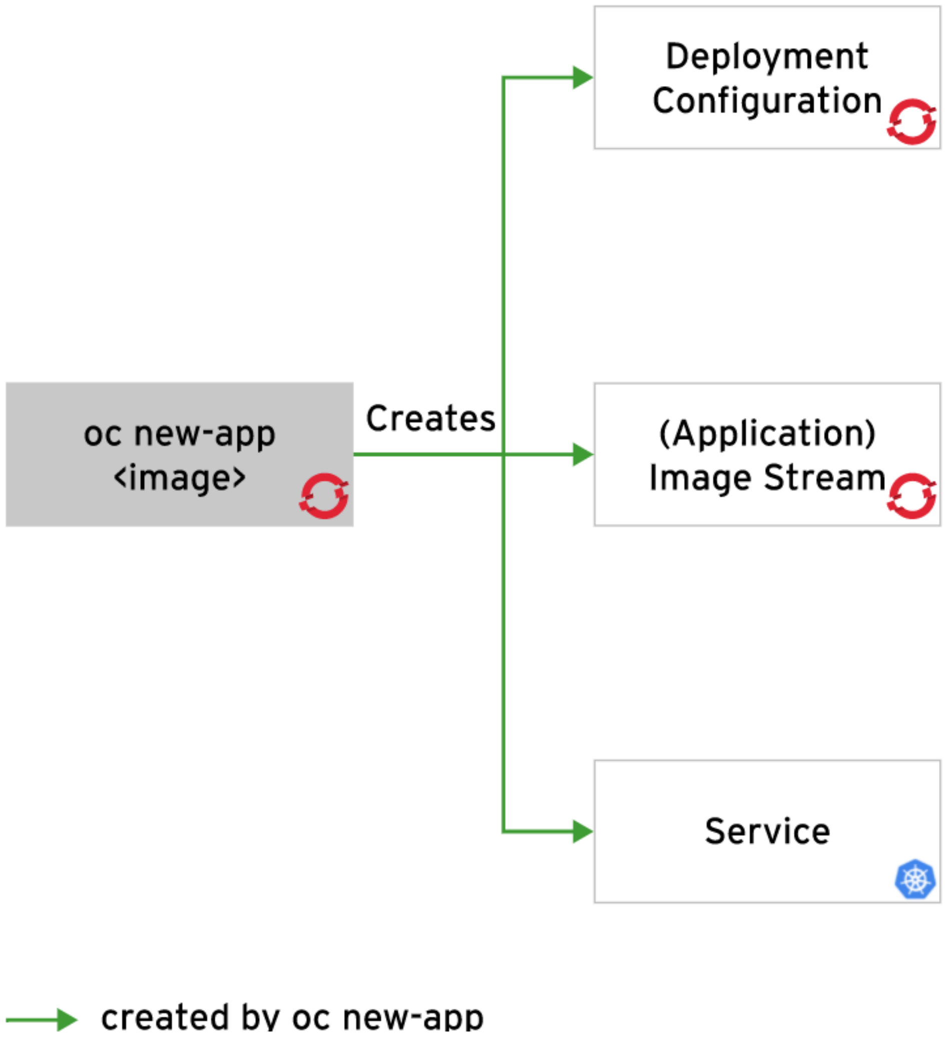
-l label = data

mysql : image stream

--name application name

--as-deployment-config : native to OCP

Resources created by oc new-app subcommand



Managing OpenShift Resources at the Command Line

- Use oc get subcommand to retrieve information about resources

- The syntax:

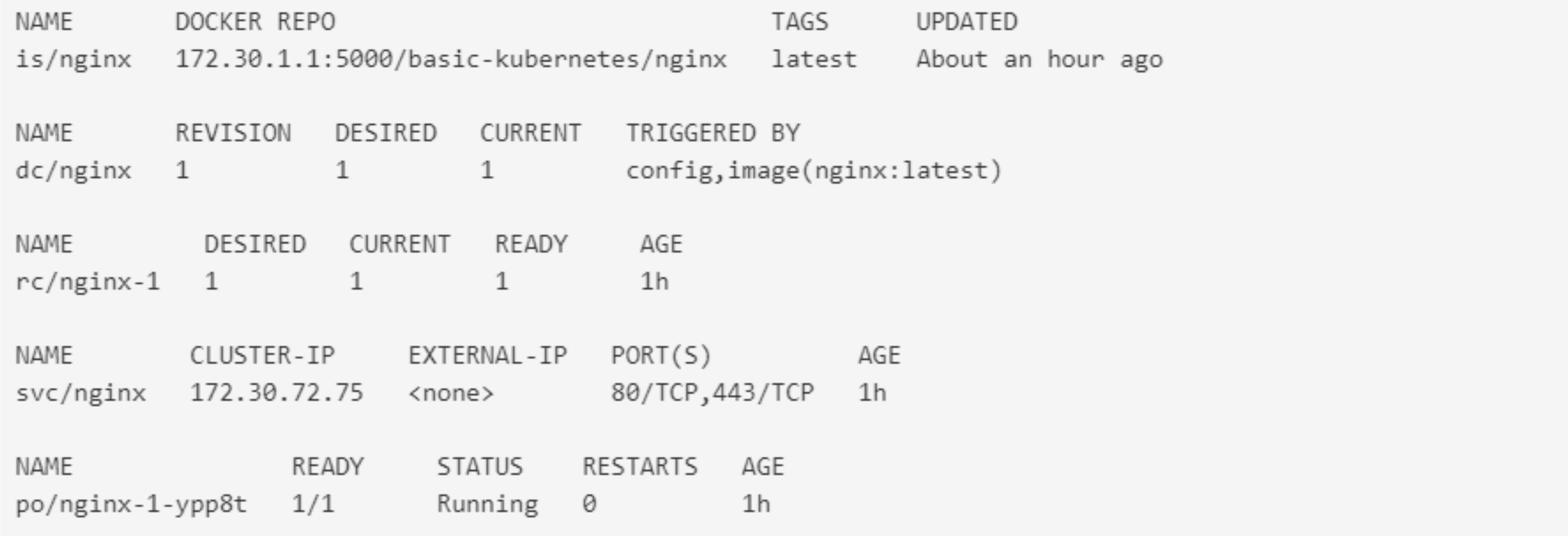
```
$ oc get RESOURCE_TYPE [-o wide] [-o yaml [ > output_file.yaml ] ]
```

- Resource_type:
  - pod
  - deployment
  - deploymentconfig or dc
  - service or svc
  - route
  - build config or bc
  - imagestream or is

Managing OpenShift Resources at the Command Line

- Use oc get all subcommand to retrieve summary info on all resources

```
$ oc get all
```



```
NAME        DOCKER REPO                              TAGS      UPDATED
is/nginx    172.30.1.1:5000/basic-kubernetes/nginx   latest    About an hour ago

NAME        REVISION   DESIRED   CURRENT   TRIGGERED BY
dc/nginx    1          1         1         config,image(nginx:latest)

NAME          DESIRED   CURRENT   READY     AGE
rc/nginx-1    1         1         1         1h

NAME         CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
svc/nginx    172.30.72.75   <none>        80/TCP,443/TCP   1h

NAME                 READY     STATUS    RESTARTS   AGE
po/nginx-1-ypp8t     1/1       Running   0          1h
```

Managing OpenShift Resources at the Command Line

- Use oc describe subcommand to retrieve details information about resources

- The syntax:

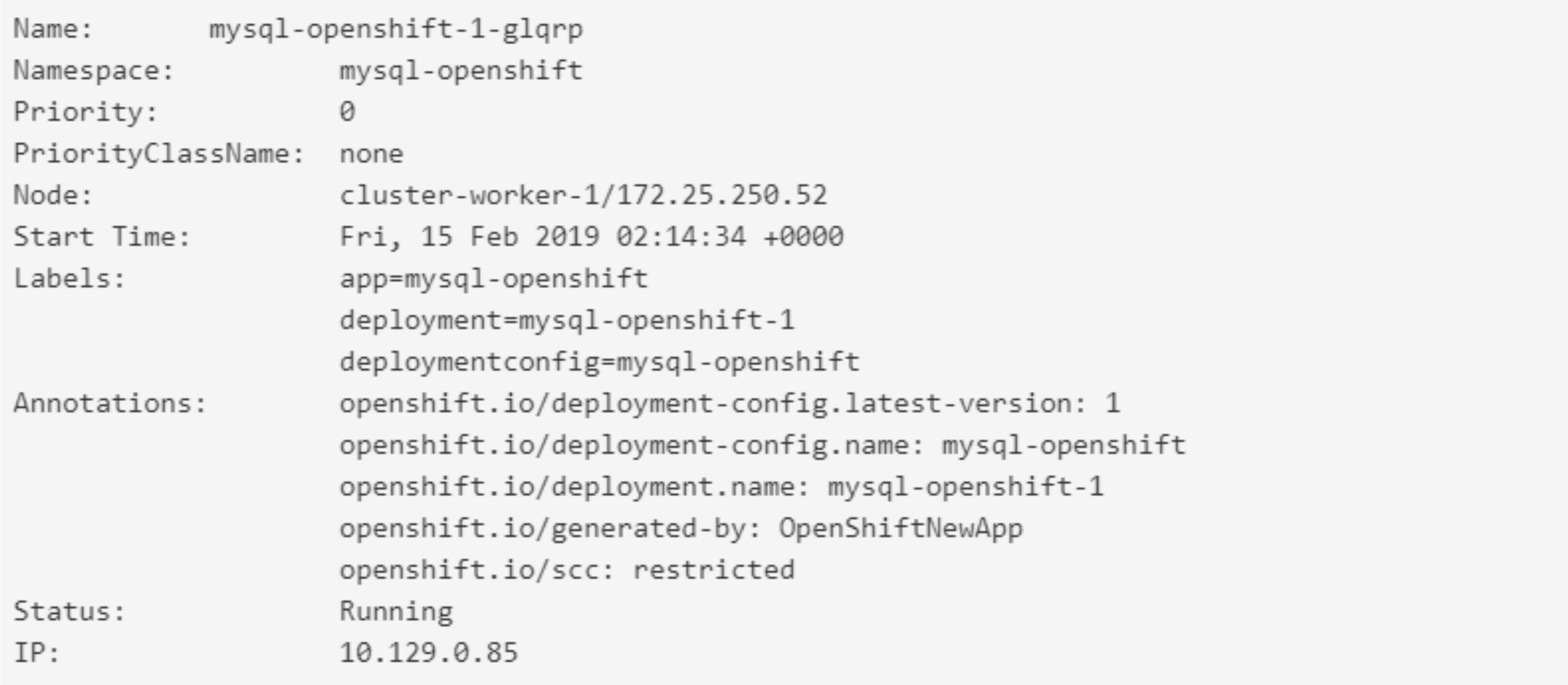
```
$ oc describe RESOURCE_TYPE
```

- Resource_type:
  - pod
  - deployment
  - deployment config or dc
  - service or svc
  - route
  - build config or bc
  - imagestream or is

Managing OpenShift Resources at the Command Line

- Retrieve detailed information on particular pod

```
$ oc describe pod mysql-openshift-1-glqrp
```



```
Name:       mysql-openshift-1-glqrp
Namespace:          mysql-openshift
Priority:           0
PriorityClassName:  none
Node:               cluster-worker-1/172.25.250.52
Start Time:         Fri, 15 Feb 2019 02:14:34 +0000
Labels:             app=mysql-openshift
                    deployment=mysql-openshift-1
                    deploymentconfig=mysql-openshift
Annotations:        openshift.io/deployment-config.latest-version: 1
                    openshift.io/deployment-config.name: mysql-openshift
                    openshift.io/deployment.name: mysql-openshift-1
                    openshift.io/generated-by: OpenShiftNewApp
                    openshift.io/scc: restricted
Status:             Running
IP:                 10.129.0.85
```

Managing OpenShift Resources at the Command Line

- Use oc edit subcommand to edit resource definition. uses vi

```
$ oc edit dc/mysql
```

- Or use oc patch subcommand to edit resource definition. Direct.

```
$ oc patch node node1 -p '{"spec":{"unschedulable":true}}'
```

- Use oc delete subcommand to delete resource.

```
$ oc delete RESOURCE_TYPE <resource_name>
```

- Execute commands inside pod named app1-n9h1t

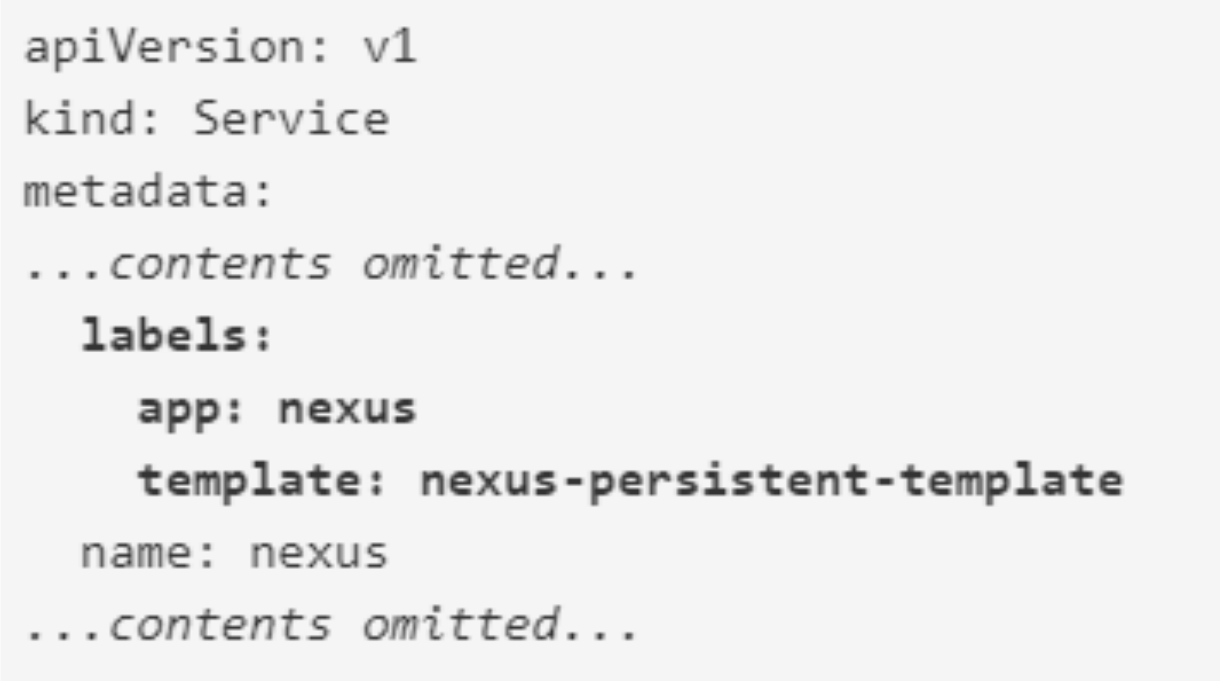
```
$ oc exec  app1-n9h1t  /bin/bash
$ oc exec  app1-n9h1t  cat /etc/*release
```

- Delete a pod named app1-n9h1t

```
$ oc delete pod app1-n9h1t
```

Labelling resources

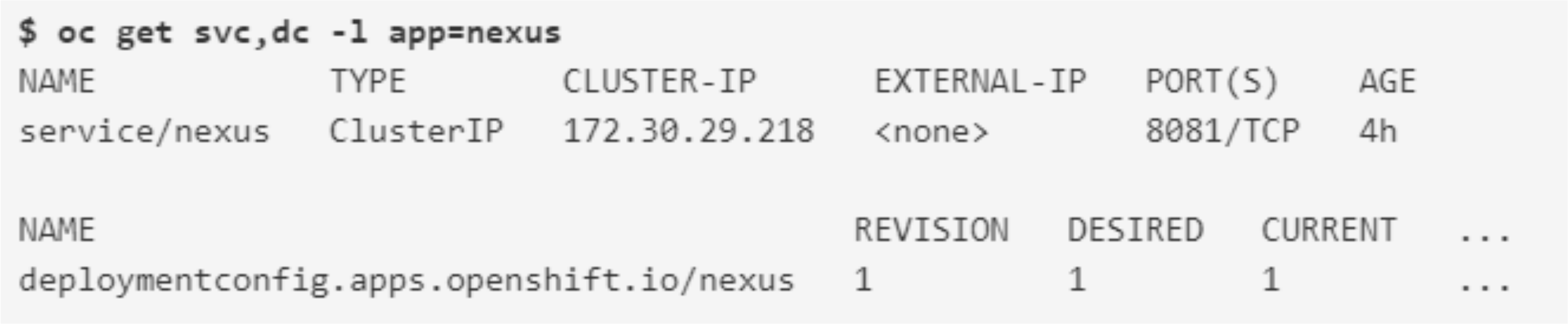
- Grouping resources by application, environment or some other criteria.
- Labels are part of metadata section of resource
- Defined as key/value pairs



```
apiVersion: v1
kind: Service
metadata:
...contents omitted...
  labels:
    app: nexus
    template: nexus-persistent-template
  name: nexus
...contents omitted...
```

Use -l option to use label

- Retrieve information on all resources belong to label app=nexus



```
$ oc get svc,dc -l app=nexus
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/nexus   ClusterIP   172.30.29.218   <none>        8081/TCP   4h

NAME                                      REVISION   DESIRED   CURRENT   ...
deploymentconfig.apps.openshift.io/nexus  1          1         1         ...
```

- Both app and template key labels are common
- The app key indicates application
- The template key labels all resources generated by the same template name

Explore it further!

- Creating Routes
- Creating Applications with Source-to-Image
- Creating Applications with OpenShift Web Console

Chapter Summary

In this chapter, you learned:

Use the OpenShift command-line client oc

Source-to-Image (S2I)

The oc new-app command can create application pods

A Route connects a public-facing IP address and DNS host name to an internal-facing service IP