

Kubernetes

Ryan Tolboom

New Jersey Institute of Technology



Local K8s Cluster Options



[“Kubernetes logo without workmark.svg”](#) by [Google, Inc](#) is licensed under [CC BY 4.0](#)

- [Docker Desktop](#) - Supports running a single-machine Kubernetes instance. Edit settings from the task-bar icon.
- [minikube](#) - Single-machine solution for developing on / testing Kubernetes.
- [k3s](#) - Lightweight Kubernetes that can run on embedded devices. Great for k8s on an Raspberry Pi.
- [Kind](#) - Runs Kubernetes inside a Docker container.

Kubectl

- The most common CLI to a Kubernetes cluster
- Can switch between contexts (different clusters)
- Can apply objects, read logs, get status, etc.
- Most things build off of these commands or at least reference them.



Kubectl Example

```
$ kubectl cluster-info
Kubernetes master is running at https://kubernetes.docker.internal:6443
KubeDNS is running at https://kubernetes.docker.internal:6443/api/v1/namespaces/kube-system/
services/kube-dns:dns/proxy
```

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

```
$ kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://kubernetes.docker.internal:6443
  name: docker-desktop
- cluster:
    certificate-authority: C:\Users\rxt1077\.minikube\ca.crt
    server: https://192.168.18.66:8443
  name: minikube
contexts:
- context:
<snip>
```

Pods



[“Labeled Pod”](#) by [Etienne Coutaud](#) is licensed under [CC BY 4.0](#)

- A pod is the minimum unit of compute in Kubernetes. It typically runs a single container.
- `kubectl get pod` is a good starting point when troubleshooting
- `kubectl describe <pod-name>` will give you even more info

Pod Example

```
$ kubectl get pod
```

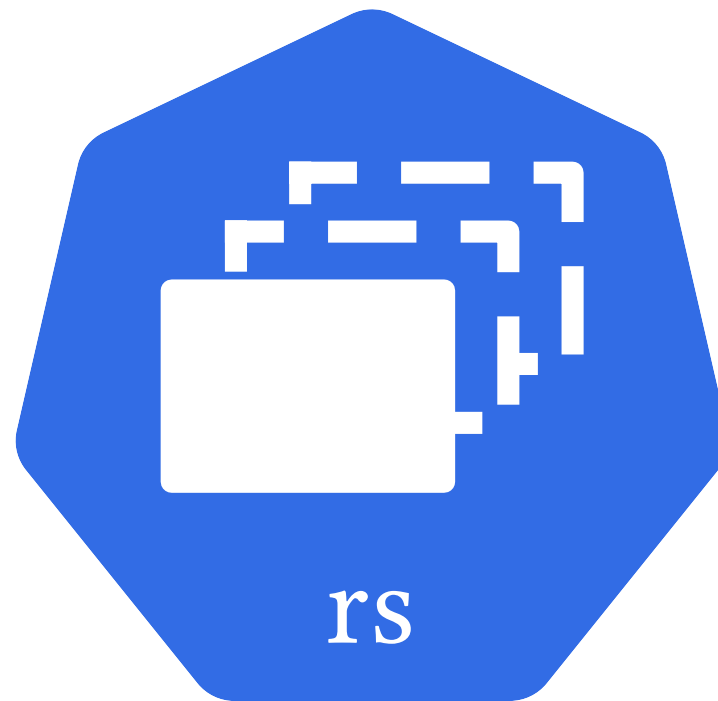
NAME	READY	STATUS	RESTARTS	AGE
hello-node-7dc7987866-xnbxz	1/1	Running	0	22s

```
$ kubectl describe pod hello-node-7dc7987866-xnbxz
```

```
Name:          hello-node-7dc7987866-xnbxz
Namespace:     default
Priority:      0
Node:          docker-desktop/192.168.65.3
Start Time:    Thu, 09 Jul 2020 16:22:31 -0400
Labels:        app=hello-node
               pod-template-hash=7dc7987866
Annotations:   <none>
Status:        Running
IP:            10.1.0.7
IPs:
  IP:          10.1.0.7
Controlled By: ReplicaSet/hello-node-7dc7987866
Containers:
  echoserver:
    <snip>
```

ReplicaSets

- A ReplicaSet is a grouping of pods that Kubernetes attempts to keep running
- Kubernetes can detect failure (configurable) and restart a pod
- Kubernetes will back-off if something keeps failing



[“Labeled ReplicaSet”](#) by [Etienne Coutaud](#) is licensed under [CC BY 4.0](#)

ReplicaSet Example

```
$ kubectl get replicaset
NAME                                DESIRED    CURRENT    READY    AGE
hello-node-7dc7987866             1          1          1        4m4s

$ kubectl describe replicaset hello-node-7dc7987866
Name:                                hello-node-7dc7987866
Namespace:                          default
Selector:                           app=hello-node,pod-template-hash=7dc7987866
Labels:                             app=hello-node
                                    pod-template-hash=7dc7987866
Annotations:                         deployment.kubernetes.io/desired-replicas: 1
                                    deployment.kubernetes.io/max-replicas: 2
                                    deployment.kubernetes.io/revision: 1
Controlled By:                       Deployment/hello-node
Replicas:                            1 current / 1 desired
Pods Status:                         1 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=hello-node
           pod-template-hash=7dc7987866
  Containers:
<snip>
```


Deployments



[“Labeled Deployment”](#) by [Etienne Coutaud](#) is licensed under [CC BY 4.0](#)

- A Deployment creates ReplicaSets to support the *deployment* of an app
- Deployments also support update strategies

Deployment Example

```
$ kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
hello-node    1/1     1            1           7m18s

$ kubectl describe deployment hello-node
Name:          hello-node
Namespace:     default
CreationTimestamp: Thu, 09 Jul 2020 16:22:31 -0400
Labels:        app=hello-node
Annotations:    deployment.kubernetes.io/revision: 1
Selector:      app=hello-node
Replicas:      1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:   RollingUpdate
MinReadySeconds: 0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=hello-node
  Containers:
    echoserver:
      Image:      k8s.gcr.io/echoserver:1.4
<snip>
```

Working with YAML objects

- Define your objects in YAML files instead of just via the command line.
- `kubectl apply -f <file>` will apply your file to make the cluster provide the resources you specified.
- Similar to ansible, this is *descriptive*: if you want three replicas and you already have two, it'll just make the third.



[“Official YAML Logo”](#) by [Ingy.dôt.Net](#) is licensed under [CC BY-SA 4.0](#)

YAML Deployment Example

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
            - containerPort: 80
```

- What k8s objects will this create?
- How many pods will the cluster start?
- What container image is being used?

YAML from a running deployment

```
$ kubectl get deployment -o yaml
apiVersion: v1
items:
- apiVersion: apps/v1
  kind: Deployment
  metadata:
    annotations:
      deployment.kubernetes.io/revision: "1"
    creationTimestamp: "2020-07-09T20:22:31Z"
    generation: 1
    labels:
      app: hello-node
    name: hello-node
    namespace: default
    resourceVersion: "1816"
    selfLink: /apis/apps/v1/namespaces/default/deployments/hello-node
    uid: 31356fed-0365-49df-8327-3c3c6ff0cb32
  spec:
    progressDeadlineSeconds: 600
    replicas: 1
<snip>
```

Services

Services make ports on pods available to other parts of the cluster *or* the outside world:

```
$ kubectl expose deployment hello-node --type=LoadBalancer --port=8080
service/hello-node exposed
```

```
$ kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
hello-node	LoadBalancer	10.105.154.187	localhost	8080:30031/TCP	100s
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	32m

Addition Kubernetes Resources

- [kubectl Cheat sheet](#)
- [Deployments](#)
- [Hello Minikube](#)
- [Services](#)



[Kubernetes 3 Wishes](#) is used under fair use