HCLSoftware

Software Containerization Lesson 2

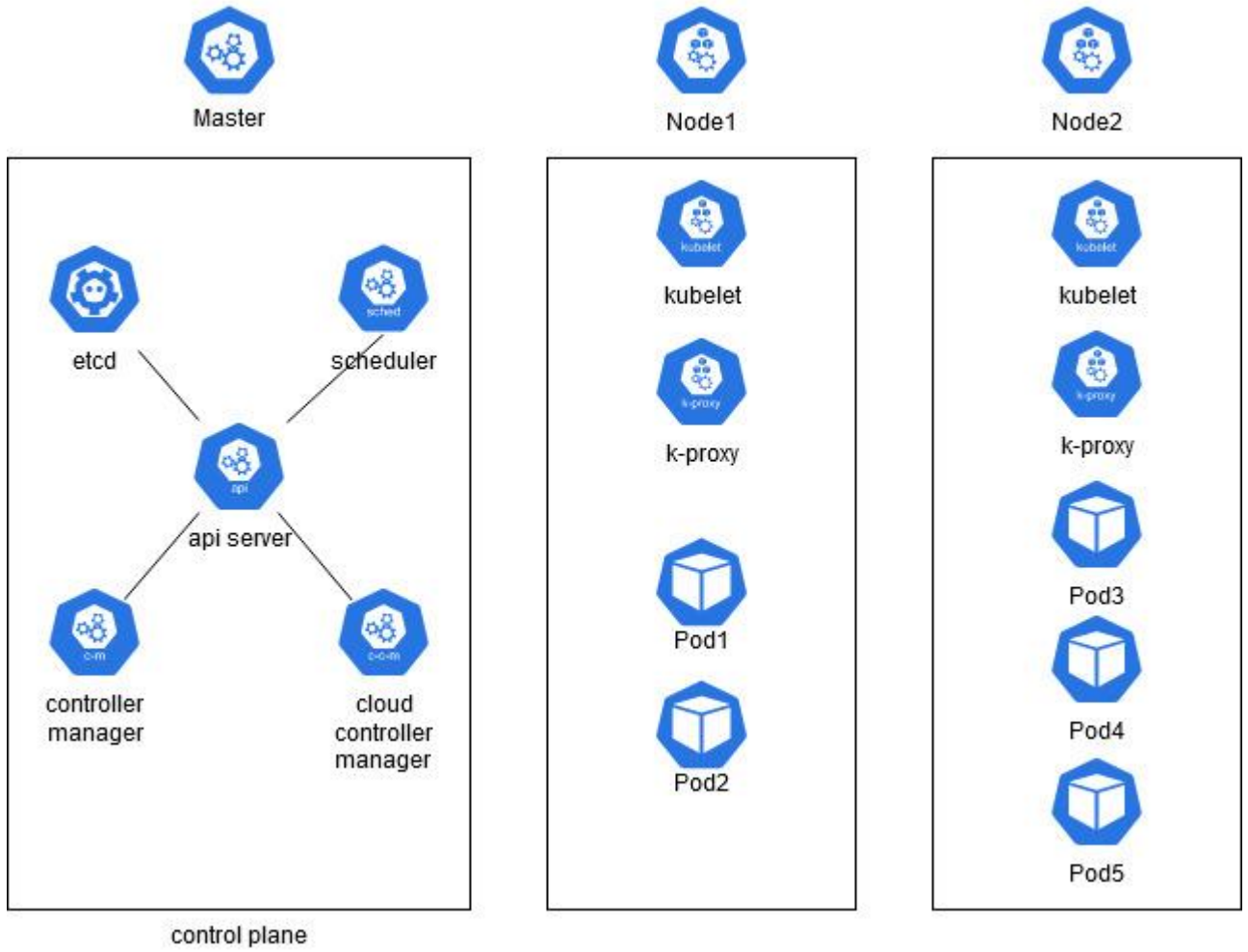
l.ziosi@vu.nl, lara.ziosi@hcl.com

Agenda

- Kubernetes architecture
  - Control plane (Master nodes)
  - Worker nodes
  - kubectl command
- Nodes
- Pods and containers
- Namespaces
- Creating pods with kubectl run
- YAML syntax
- Deployment
- ReplicaSet
- DaemonSet
- Job
- CronJob
- Garbage Collector

Kubernetes Architecture – control plane

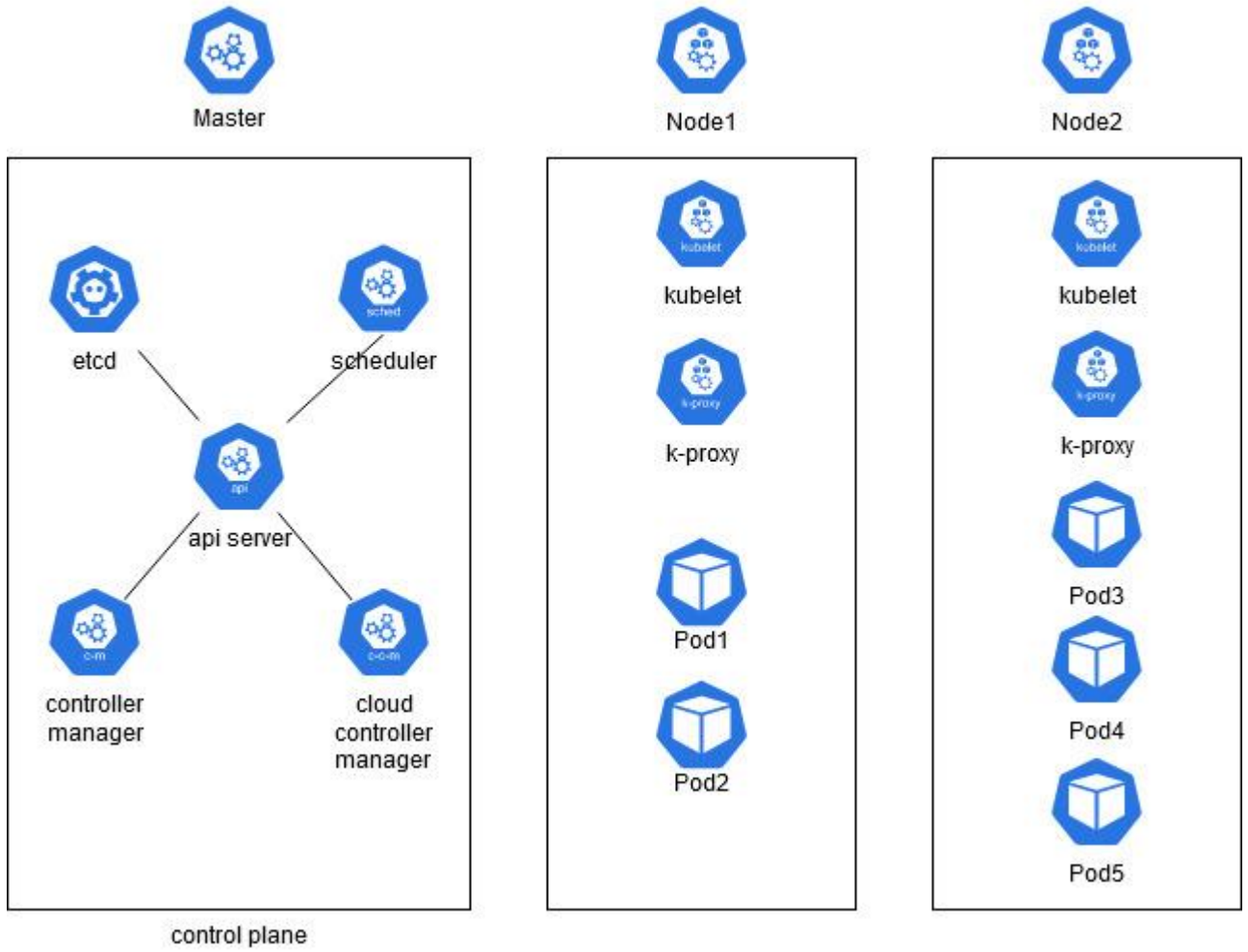
- **Control plane**: 1-3+ Master nodes
- **Master Node**: Physical or Virtual Machine that doesn't run user workloads directly but manages the cluster
- **kube-apiserver**: expose the Kubernetes api
- **etcd**: storage of key value pairs for Kubernetes configuration
- **kube-scheduler**: selects the node on which to run newly created pods
- **kube-controller-manager**: runs controller processes (Node, Replication, Endpoints, Service Account & Token controllers)
- **Cloud controller manager:** connects the cluster to the cloud provider's API, separating components that interact with the cloud platform from components that interact with the kubernetes cluster.



Reference: https://kubernetes.io/docs/concepts/overview/components/

Kubernetes Architecture – Worker nodes

- **Node/Worker/Minion**: Physical or Virtual Machine that executes user workloads, interacting with the control plane
- **kubelet**: ensures that pods are running in the node as declared by their specifications
- **kube-proxy**: network proxy, maintains network rules on nodes
- **pod**: group of one or more containers that share storage/network resources, described by a specification about how to run the containers



https://kubernetes.io/docs/concepts/overview/components/

Kubectl command

- The kubectl command lets you interact with the kubernetes cluster. It is a utility that invokes the kubernetes api.
- The general syntax of kubectl is:

```
kubectl [command] [TYPE] [NAME] [flags]
```

- The value of **command** can be: create, get, describe, delete, logs, exec, run and many more
- The value of **type** can be any resource type (examples: node, pod, service, namespace, configmap, secret, job). Types are case-insensitive and you can specify the singular, plural, or abbreviated forms
- The value of **name** is case sensitive and if it is omitted, then the command will apply to all objects of that type
- The value of **flags** can be obtained using `kubectl help`
- Examples:

```
kubectl get pods
kubectl delete pod <pod-name>
kubectl describe pod <pod-name>
kubectl get pods --namespace <namespace-name>
```

- You can abbreviate types, for example write pod instead of pods: `kubectl get pod`

https://kubernetes.io/docs/reference/kubectl/overview/

Kubectl configuration

Kubectl needs to be configured so it can connect to the control-plane. The configuration is stored in the file:

```
~/.kube/config
```

If you use a microk8s single node, you don't need to set up the configuration manually.

You can view the configuration by running the command:

```
microk8s config
```

For more information, see:
https://microk8s.io/docs/working-with-kubectl

Kubectl examples

```
$ kubectl get nodes
NAME       STATUS   ROLES    AGE     VERSION
minikube   Ready    master   6m23s   v1.17.3
```

```
$ kubectl get namespace
NAME                   STATUS   AGE
default                Active   6m57s
kube-node-lease        Active   6m59s
kube-public            Active   6m59s
kube-system            Active   6m59s
kubernetes-dashboard   Active   6m50s
```

```
$ kubectl get pods --namespace kube-system
NAME                               READY   STATUS    RESTARTS   AGE
coredns-6955765f44-wr4bv           1/1     Running   0          35s
coredns-6955765f44-z8qwl           1/1     Running   0          35s
etcd-minikube                      1/1     Running   0          38s
kube-apiserver-minikube            1/1     Running   0          38s
kube-controller-manager-minikube   1/1     Running   0          38s
kube-proxy-tp2cx                   1/1     Running   0          35s
kube-scheduler-minikube            1/1     Running   0          38s
storage-provisioner                1/1     Running   0          39s
$
```

Powered by Katacoda

```
$ kubectl describe pod etcd-minikube
Error from server (NotFound): pods "etcd-minikube" not found
$ kubectl describe pod etcd-minikube -n kube-system
Name:                 etcd-minikube
Namespace:            kube-system
Priority:             2000000000
Priority Class Name:  system-cluster-critical
Node:                 minikube/172.17.0.22
Start Time:           Sat, 14 Nov 2020 13:40:01 +0000
```

Powered by Katacoda

```
$ kubectl get services -n kube-system
NAME       TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)                  AGE
kube-dns   ClusterIP   10.96.0.10   <none>        53/UDP,53/TCP,9153/TCP   3m53s
```

```
$ kubectl get secrets -n kube-system
NAME                                  TYPE
attachdetach-controller-token-mrmhv   kubernetes.io/service-account-token
bootstrap-signer-token-fknsd          kubernetes.io/service-account-token
bootstrap-token-a53kyo                bootstrap.kubernetes.io/token
certificate-controller-token-pdsv5    kubernetes.io/service-account-token
clusterrole-aggregation-controller-token-9sk2k   kubernetes.io/service-account-token
coredns-token-5w8zd                   kubernetes.io/service-account-token
cronjob-controller-token-rkk78        kubernetes.io/service-account-token
daemon-set-controller-token-8xjf7     kubernetes.io/service-account-token
default-token-ckc4h                   kubernetes.io/service-account-token
deployment-controller-token-85n6v     kubernetes.io/service-account-token
disruption-controller-token-482n8     kubernetes.io/service-account-token
endpoint-controller-token-s4kqj       kubernetes.io/service-account-token
```

Try it yourself in the browser at: https://kubernetes.io/docs/tutorials/kubernetes-basics/create-cluster/

Nodes

- A Node is a physical or virtual machine that runs the **kubelet** agent, the **kube-proxy** pod and **Network pod**.

- **Kubelet installation:**
  - If you install k8s with **kubeadm**, kubelet must be installed manually on each worker node as described in: https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/
  - If you install k8s using **microk8s**, kubelet will be installed automatically.

- **Network pod**: Kubernetes does not include any Network pod. You need to choose and install a network add-on.
  - In the following example, the Network pod is provided by **Weave Net** but there are other options: https://www.weave.works/docs/net/latest/kubernetes/kube-addon/
  - If you install **microk8s**, the **Calico** Network is installed by default.

- These components allow the cluster to manage the pods that run on the node.

- Each worker Node needs to be registered with the cluster.

Node example with WeaveNet Network

```
[root@kbnode1 ~]# kubectl describe node kbnode1
Name:               kbnode1
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=kbnode1
                    kubernetes.io/os=linux
Annotations:        kubeadm.alpha.kubernetes.io/cri-socket: /var/run/dockershim.sock
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Fri, 04 Sep 2020 19:23:04 +0530
Taints:             <none>
Unschedulable:      false
Lease:
  HolderIdentity:  kbnode1
  AcquireTime:     <unset>
  RenewTime:       Sun, 15 Nov 2020 15:31:49 +0530
Conditions:
  Type                 Status  LastHeartbeatTime                 LastTransitionTime                Reason                       Message
  ----                 ------  -----------------                 ------------------                ------                       -------
  NetworkUnavailable   False   Sun, 15 Nov 2020 14:46:25 +0530   Sun, 15 Nov 2020 14:46:25 +0530   WeaveIsUp                    Weave pod has set this
  MemoryPressure       False   Sun, 15 Nov 2020 15:28:05 +0530   Fri, 04 Sep 2020 19:23:05 +0530   KubeletHasSufficientMemory   kubelet has sufficient memory available
  DiskPressure         False   Sun, 15 Nov 2020 15:28:05 +0530   Fri, 04 Sep 2020 19:23:05 +0530   KubeletHasNoDiskPressure     kubelet has no disk pressure
  PIDPressure          False   Sun, 15 Nov 2020 15:28:05 +0530   Fri, 04 Sep 2020 19:23:05 +0530   KubeletHasSufficientPID      kubelet has sufficient PID available
  Ready                True    Sun, 15 Nov 2020 15:28:05 +0530   Fri, 04 Sep 2020 19:24:35 +0530   KubeletReady                 kubelet is posting ready status
Addresses:
  InternalIP:  10.0.2.7
  Hostname:    kbnode1
Capacity:
  cpu:                2
  ephemeral-storage:  45615348Ki
  hugepages-2Mi:      0
  memory:             8000364Ki
  pods:               110
Allocatable:
  cpu:                2
  ephemeral-storage:  42039104648
  hugepages-2Mi:      0
  memory:             7897964Ki
  pods:               110
```

Node example continued

```
System Info:
  Machine ID:                 9cfc10c654e9413e8116328ba497b2a3
  System UUID:                61b81414-9ad9-400c-b804-5c83c6a90209
  Boot ID:                    c54b6a8c-8449-407e-bc28-5507f3683d98
  Kernel Version:             4.18.0-147.el8.x86_64
  OS Image:                   CentOS Linux 8 (Core)
  Operating System:           linux
  Architecture:               amd64
  Container Runtime Version:  docker://18.9.1
  Kubelet Version:            v1.19.0
  Kube-Proxy Version:         v1.19.0
Non-terminated Pods:          (19 in total)
  Namespace                   Name                                    CPU Requests  CPU Limits  Memory Requests  Memory Limits  AGE
  ---------                   ----                                    ------------  ----------  ---------------  -------------  ---
  default                     details-v1-79c697d759-vsxhw             10m (0%)      2 (100%)    40Mi (0%)        1Gi (13%)      71d
  default                     httpbin-74fb669cc6-4npjs                10m (0%)      2 (100%)    40Mi (0%)        1Gi (13%)      33d
  default                     nginx-app-d6ff45774-grm5d               10m (0%)      2 (100%)    40Mi (0%)        1Gi (13%)      42m
  default                     productpage-v1-65576bb7bf-zrk6g         10m (0%)      2 (100%)    40Mi (0%)        1Gi (13%)      71d
  default                     ratings-v1-7d99676f7f-v795f             10m (0%)      2 (100%)    40Mi (0%)        1Gi (13%)      71d
  default                     reviews-v1-987d495c-5jqxd               10m (0%)      2 (100%)    40Mi (0%)        1Gi (13%)      71d
  default                     reviews-v2-6c5bf657cf-49zfp             10m (0%)      2 (100%)    40Mi (0%)        1Gi (13%)      71d
  default                     reviews-v3-5f7b9f4f77-w9x4h             10m (0%)      2 (100%)    40Mi (0%)        1Gi (13%)      71d
  istio-system                grafana-57bb676c4c-zxpqr                0 (0%)        0 (0%)      0 (0%)           0 (0%)         37d
  istio-system                istio-egressgateway-b9d46896-qf572      10m (0%)      2 (100%)    40Mi (0%)        1Gi (13%)      71d
  istio-system                istio-ingressgateway-dc76747bf-vqqrg    10m (0%)      2 (100%)    40Mi (0%)        1Gi (13%)      71d
  istio-system                istiod-69c88fcb8-kprb6                  10m (0%)      0 (0%)      100Mi (1%)       0 (0%)         71d
  istio-system                jaeger-75948789b4-k4qh6                 10m (0%)      0 (0%)      0 (0%)           0 (0%)         37d
  istio-system                kiali-7d5cb68b45-ptd44                  0 (0%)        0 (0%)      0 (0%)           0 (0%)         37d
  istio-system                prometheus-7c8bf6df84-l7bpj             0 (0%)        0 (0%)      0 (0%)           0 (0%)         37d
  kube-system                 kube-proxy-m7glv                        0 (0%)        0 (0%)      0 (0%)           0 (0%)         71d
  kube-system                 weave-net-jm29z                         100m (5%)     0 (0%)      200Mi (2%)       0 (0%)         71d
  sftp                        sftp-768748c469-qflfr                   0 (0%)        0 (0%)      0 (0%)           0 (0%)         32d
  sftpn                       sftp-66b57db894-nqspn                   0 (0%)        0 (0%)      0 (0%)           0 (0%)         33d
Allocated resources:
```

Node example continued

```
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource           Requests    Limits
  --------           --------    ------
  cpu                220m (11%)  20 (1000%)
  memory             700Mi (9%)  10Gi (132%)
  ephemeral-storage  0 (0%)      0 (0%)
  hugepages-2Mi      0 (0%)      0 (0%)
Events:
  Type     Reason                   Age                From                Message
  ----     ------                   ----               ----                -------
  Normal   Starting                 46m                kubelet, kbnode1    Starting kubelet.
  Normal   NodeAllocatableEnforced  46m                kubelet, kbnode1    Updated Node Allocatable limit across pods
  Normal   NodeHasNoDiskPressure    46m (x7 over 46m)  kubelet, kbnode1    Node kbnode1 status is now: NodeHasNoDiskPressure
  Normal   NodeHasSufficientPID     46m (x7 over 46m)  kubelet, kbnode1    Node kbnode1 status is now: NodeHasSufficientPID
  Warning  Rebooted                 46m                kubelet, kbnode1    Node kbnode1 has been rebooted, boot id: c54b6a8c-8449-407e-bc28-5507f3683d98
  Normal   NodeHasSufficientMemory  46m (x8 over 46m)  kubelet, kbnode1    Node kbnode1 status is now: NodeHasSufficientMemory
  Normal   Starting                 46m                kube-proxy, kbnode1 Starting kube-proxy.
```

```
[root@kbnode1 ~]# ps -ef |grep kubelet
root      1498     1  9 14:44 ?        00:05:01 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kuberne
tes/kubelet.conf --config=/var/lib/kubelet/config.yaml --network-plugin=cni --pod-infra-container-image=k8s.gcr.io/pause:3.2
```

If you install microk8s, the installation paths are different:

`/snap/microk8s/1864/kubelet`

`/snap/microk8s/1864/kube-apiserver`

Communication between control plane and worker nodes

If you work in a multi-node cluster where restrictive firewall policies are enforced on each node, you need to modify the firewall rules and open all the ports required for the communication needs of the control planes and of the worker nodes.

Control-plane node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	6443*	Kubernetes API server	All
TCP	Inbound	2379-2380	etcd server client API	kube-apiserver, etcd
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	10251	kube-scheduler	Self
TCP	Inbound	10252	kube-controller-manager	Self

Worker node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	30000-32767	NodePort Services†	All

† Default port range for NodePort Services.

Table from: https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

Namespaces

- Namespaces are used to create virtual clusters inside a cluster to avoid name conflicts, restrict resource usage establish quotas, attach policies that define who can perform which actions.
- By default, K8s creates objects in the **default** namespace.
- K8s components run in separate namespace called **kube-system**.
- You can create additional namespaces to separate the work of different teams or different applications:

```
kubectl create namespace <namespace-name>
```

- Namespaces cannot be nested.
- Not all resource types belong to a namespace. For example, pods and services are in a namespace. Namespaces, persistent volumes and nodes aren't. To see resource types that are (or aren't) in a namespace use:

```
kubectl api-resources --namespaced=true
kubectl api-resources --namespaced=false
```

- To see which resources of a certain type are in a namespace, append `-n namespace-name`:

```
kubectl get pods -n <namespace-name>
kubectl describe pod <pod-name> -n <namespace-name>
```

Namespace examples

```
[root@kbnode1 ~]# kubectl create namespace sample
namespace/sample created
```

```
[root@kbnode1 ~]# kubectl describe namespace sample
Name:         sample
Labels:       <none>
Annotations:  <none>
Status:       Active

No resource quota.

No LimitRange resource.
```

```
[root@kbnode1 ~]# kubectl get pods -n sample
No resources found in sample namespace.
[root@kbnode1 ~]# kubectl get pods -n default
NAME                              READY   STATUS    RESTARTS   AGE
details-v1-79c697d759-vsxhw       2/2     Running   8          71d
httpbin-74fb669cc6-4npjs          2/2     Running   4          33d
nginx-app-d6ff45774-grm5d         2/2     Running   0          33m
productpage-v1-65576bb7bf-zrk6g   2/2     Running   8          71d
ratings-v1-7d99676f7f-v795f       2/2     Running   8          71d
reviews-v1-987d495c-5jqxd         2/2     Running   8          71d
reviews-v2-6c5bf657cf-49zfp       2/2     Running   8          71d
reviews-v3-5f7b9f4f77-w9x4h       2/2     Running   8          71d
```

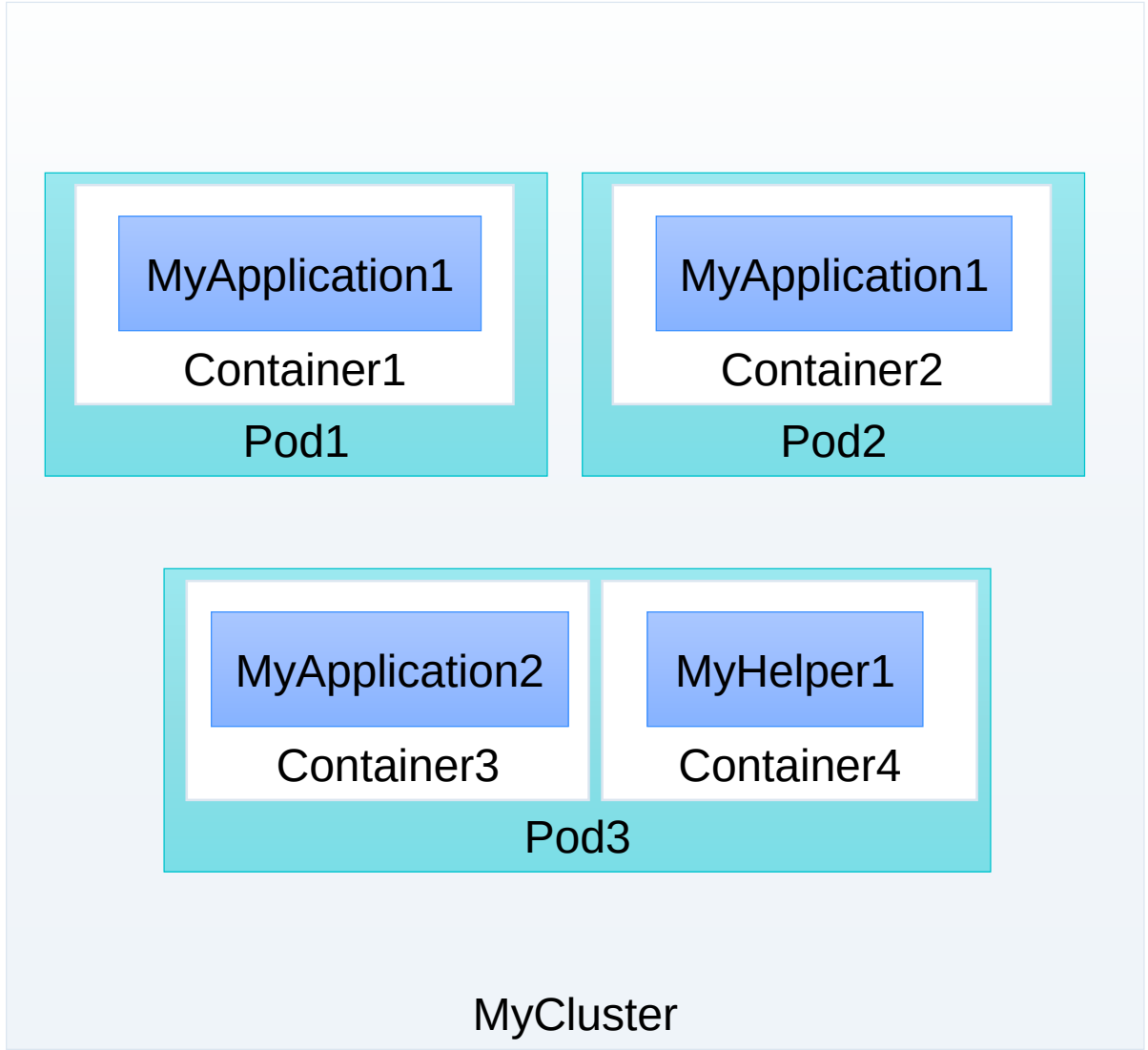
The following outputs are truncated:

```
[root@kbnode1 ~]# kubectl api-resources --namespaced=true
NAME                        SHORTNAMES   APIGROUP                       NAMESPACED   KIND
bindings                                                                true         Binding
configmaps                  cm                                          true         ConfigMap
endpoints                   ep                                          true         Endpoints
events                      ev                                          true         Event
limitranges                 limits                                      true         LimitRange
persistentvolumeclaims      pvc                                         true         PersistentVolumeClaim
pods                        po                                          true         Pod
podtemplates                                                            true         PodTemplate
replicationcontrollers      rc                                          true         ReplicationController
resourcequotas              quota                                       true         ResourceQuota
secrets                                                                 true         Secret
serviceaccounts             sa                                          true         ServiceAccount
services                    svc                                         true         Service
controllerrevisions                      apps                           true         ControllerRevision
daemonsets                  ds           apps                           true         DaemonSet
deployments                 deploy       apps                           true         Deployment
replicasets                 rs           apps                           true         ReplicaSet
statefulsets                sts          apps                           true         StatefulSet
```

```
[root@kbnode1 ~]# kubectl api-resources --namespaced=false
NAME                              SHORTNAMES   APIGROUP                       NAMESPACED   KIND
componentstatuses                 cs                                          false        ComponentStatus
namespaces                        ns                                          false        Namespace
nodes                             no                                          false        Node
persistentvolumes                 pv                                          false        PersistentVolume
```

Pods and Containers

- A **pod** is the smallest unit you can deploy to a Kubernetes Cluster
- A pod generally contains a **single container**. To **scale horizontally**, you create multiple identical pods.
- A pod may contain **multiple containers** that are strictly related. All containers in a pod share the same network and storage resources.
- A pod may contain one or more **init containers** used to initialize the application. These containers run in sequence and end as soon as they finish their job.
- You can create pods directly from the command line but more often you let the **kube-controller** create them for you.
- In this case you provide the pod specification as part of a **Deployment**, **ReplicaSet**, **StatefulSet, DaemonSet or Job.**



Creating a pod from the command line

To create and run a pod from the command line you use the `kubectl run` command.

Example:

```
kubectl run –image <image-name> <pod-name> --port=<port_on_container_ip> --hostport=<port_on_host_ip>
```

```
lara@kube-master-gui:~$ kubectl run --image nginx nginx-app --port=80 --hostport=80
pod/nginx-app created
```

To check that the pod is running and see its IP address you can use:

```
kubectl get pods –o wide
```

```
lara@kube-master-gui:~$ kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP            NODE              NOMINATED NODE   READINESS GATES
nginx-app   1/1     Running   0          30s   10.1.96.187   kube-master-gui   <none>           <none>
```

For more options of the `kubectl run` command, type:

```
kubectl run --help
```

Pod lifecycle

The lifecycle of a pod goes through different phases, that can take the following values:

Order	Phase	Description
1	Pending	Cluster accepted the Pod, but not al containers are running yet. Possible reasons: scheduling is still ongoing or missing images are being downloaded
2	Running	The pod is bound to a node. All containers have been created. At least one container is running, starting or restarting
3.1	Succeeded	All containers terminated successfully; they will not restart.
3.2	Failed	All containers terminated. At least one container terminated with error.
any	Unknown	Status cannot be detected (controller cannot communicate with worker node)

If the status is **CrashLoopBackoff** , there's a major problem with the configuration of the pod and one of its containers exited unexpectedly.

Getting the details of the running pod

- To get the IP address of the running pod, you can use:

  `kubectl describe pod <pod-name>`

- In the example below you can see that the IP address is: 10.1.96.170. If the pod stops and restarts, the IP will change.
- You can also see the Port and Host Port values associated with a container.
- The Container State can be: `Waiting`, `Running` or `Terminating`.
- If the container image fails to download the State will be `Waiting` with reason: `ImagePullBackoff`.

```
lara@kube-master-gui:~$ kubectl describe pod nginx-app
Name:         nginx-app
Namespace:    default
Priority:     0
Node:         kube-master-gui/10.0.2.15
Start Time:   Sun, 03 Jan 2021 15:16:57 +0100
Labels:       run=nginx-app
Annotations:  cni.projectcalico.org/podIP: 10.1.96.170/32
              cni.projectcalico.org/podIPs: 10.1.96.170/32
Status:       Running
IP:           10.1.96.170
IPs:
  IP:  10.1.96.170
Containers:
  nginx-app:
    Container ID:   containerd://33eaf6dd17c0e7910b9ceed049cdbacdccaf91081c864260729f7caa1a4a0c22
    Image:          nginx
    Image ID:       docker.io/library/nginx@sha256:4cf620a5c81390ee209398ecc18e5fb9dd0f5155cd82adcbae532fec94006fb9
    Port:           80/TCP
    Host Port:      80/TCP
    State:          Running
      Started:      Sun, 03 Jan 2021 15:17:01 +0100
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-tnd79 (ro)
```

Checking that the nginx service is running

If your pod has exposed a port, you can check for the nginx service using the current IP address of the pod and the port:

```
lara@kube-master-gui:~$ curl http://10.1.96.176:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

If your pod has exposed a hostport, you can check for the nginx service using the IP address of the host and the hostport:

```
lara@kube-master-gui:~$ curl http://localhost:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

Running a command inside a container

To run a shell inside a container (that has that shell installed), you can use a command like:

`kubectl exec -it <pod-name> -- /bin/bash`

You might need to use a different shell, for example use `ash` in Alpine Linux.
The `-t` flag tells the container that the input is from a TTY and `-i` ensures that the shell input is passed to the container. To exit from shell, type exit.
You can also run other commands that are available inside the container.
In the example below, you can where the nginx image configures the server to listen to port 80.

```
lara@kube-master-gui:~$ kubectl exec -it nginx-app -- /bin/bash
root@nginx-app:/# ls
bin   docker-entrypoint.d   home   media  proc  sbin  tmp
boot  docker-entrypoint.sh  lib    mnt    root  srv   usr
dev   etc                   lib64  opt    run   sys   var
root@nginx-app:/# more /etc/nginx/conf.d/default.conf
server {
    listen       80;
    listen  [::]:80;
    server_name  localhost;
```

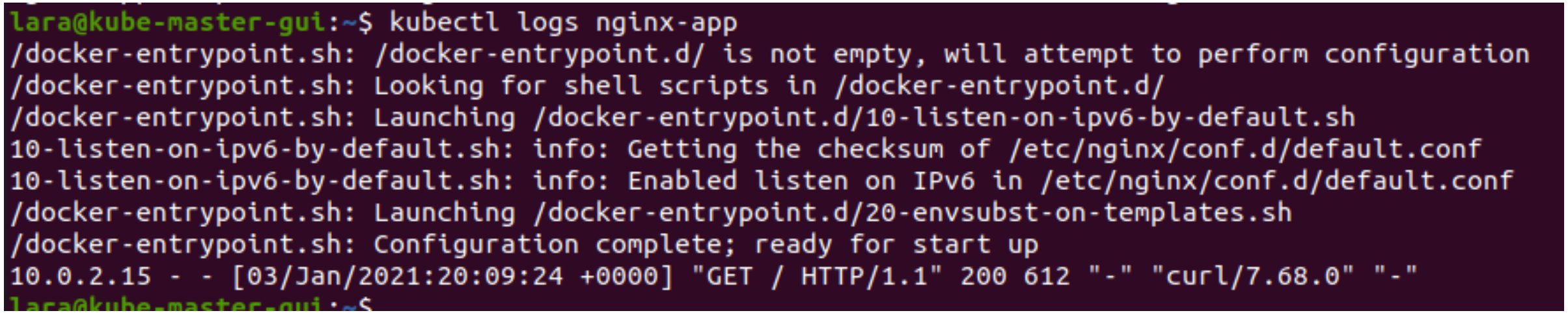
Getting the logs of a pod

While the pod runs, you can view the logs using the command:

```
kubectl logs <pod_name> -n <namespace_name> -c <container_name>
```

The log combines the stdout and stderr of the applications running in the pod.

To specify a container inside the pod, use the –c flag followed by the container name.



HCLSoftware

Kubernetes Workloads

Workload resources and controllers

Typically, instead of running individual pods, you create *workload resources* that manage a set of pods based on the *desired state* that you declare.

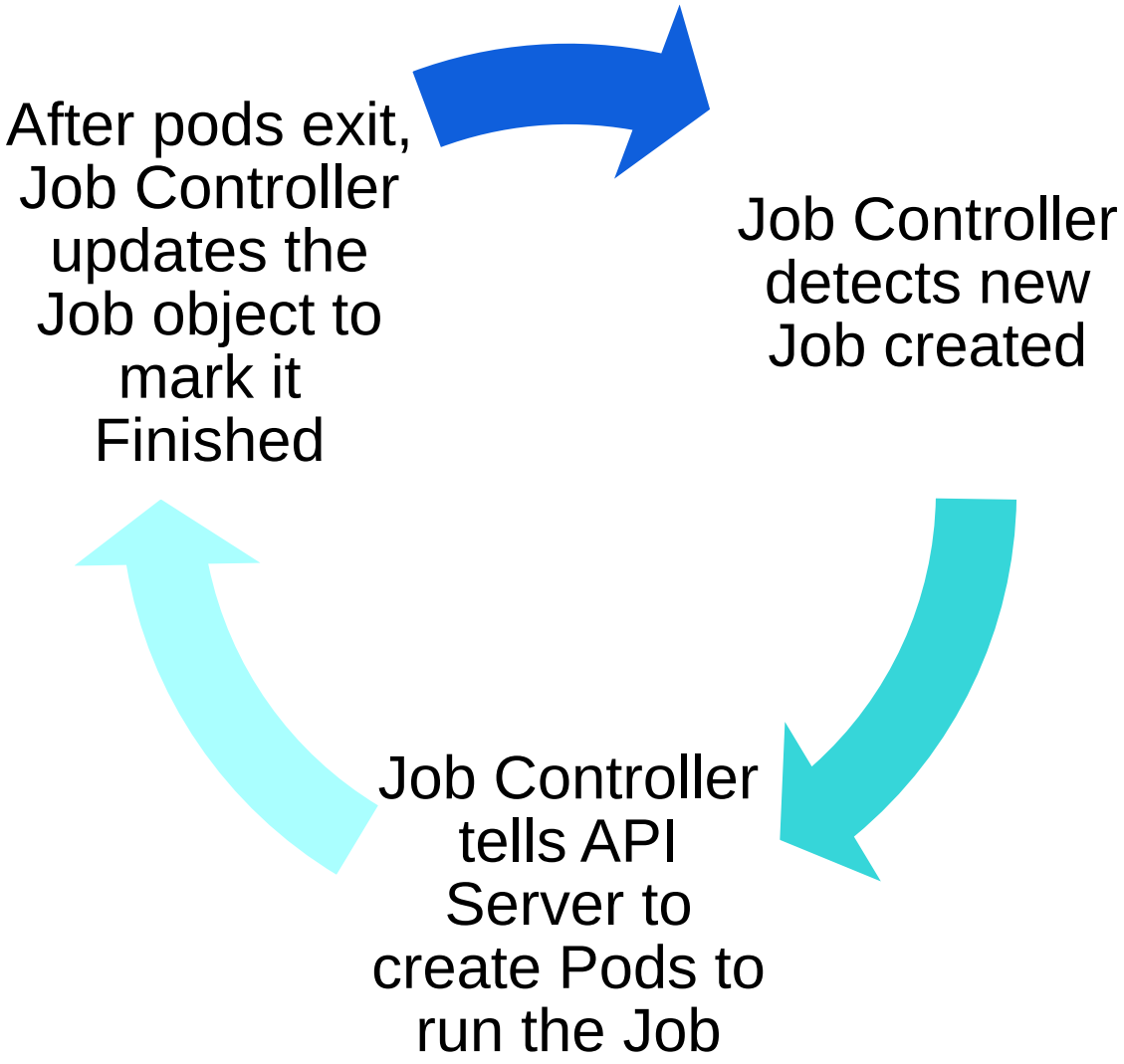
Types of workload resources:

- *Deployment*
- *ReplicaSet*
- *DaemonSet*
- *Job*
- *CronJob*
- *StatefulSet* (requires concepts about persistent storage and will be discussed in a later lesson).

When you submit a workload resource, the K8s control plane configures a corresponding *controller*.

Controllers run *control loops*: infinite loops in which the controller periodically checks if the status of the resource corresponds to the declared desired state, and if it doesn't match, it tries to correct the situation.

Control Loop (example of the Job Controller)



https://kubernetes.io/docs/reference/command-line-tools-reference/kube-controller-manager/
https://kubernetes.io/docs/concepts/architecture/controller/

Spec and status fields

Most Kubernetes objects have two fields: *spec* and *status*.

The *spec* is set when you create the object, and it describes the object **desired state**.

You create the *spec* when you write the YAML file that described the object.

The *status* describes the **current state** of the object.

When you get the object at runtime in YAML format, you see also the *status*.

The controller continually tries to make the object actual state match the desired state.

Deployment

A deployment is a type of workload that is suitable for running stateless applications, such as web servers.

You can create deployments with the CLI or with a YAML file.

In a Deployment YAML file, you describe the desired number of replicas and the specification of the containers that you want to run.

The Deployment Controller ensures that the cluster realizes the desired state, by creating or deleting pods as required to match the desired number of replicas specified by the Deployment.

Creating deployments from the command line

- To create a deployment from an image:

```
kubectl create deployment --image <image-name> <deployment-name>
```

```
lara@kube-master-gui:~$ kubectl create deployment --image nginx nginx-app
deployment.apps/nginx-app created
```

- To list the deployment:

```
kubectl get deployment <deployment-name>
```

```
lara@kube-master-gui:~$ kubectl get deployment nginx-app
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
nginx-app   0/1     1            0           27s
```

```
lara@kube-master-gui:~$ kubectl get deployment nginx-app
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
nginx-app   1/1     1            1           75s
```

- To list the pods (created as part of the deployment):

```
kubectl get pods
```

```
lara@kube-master-gui:~$ kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
nginx-app-d6ff45774-nfbl2    1/1     Running   0          6m6s
```

Describing deployments

You can describe the deployment using:

```
kubectl describe deployment <deployment-name>
```

As you can see there are no ports, hence it won't be possible to access the nginx server from a web browser. You can use `--port=80` to publish the port when creating the deployment.

```
lara@kube-master-gui:~$ kubectl describe deployment nginx-app
Name:                   nginx-app
Namespace:              default
CreationTimestamp:      Sun, 03 Jan 2021 12:43:43 +0100
Labels:                 app=nginx-app
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=nginx-app
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=nginx-app
  Containers:
   nginx:
    Image:        nginx
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   nginx-app-d6ff45774 (1/1 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  53s   deployment-controller  Scaled up replica set nginx-app-d6ff45774 to 1
```

Using YAML descriptors in Kubernetes

- To see the descriptor of the deployment in yaml format:

  `kubectl get deployment <deployment-name> -o yaml`

```
kubectl get deployment nginx-app -o yaml
apiVersion: apps/v1
kind: Deployment
metadata:
....
  labels:
    app: nginx-app
  managedFields:
....
  - apiVersion: apps/v1
...
  name: nginx-app
  namespace: default
...
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: nginx-app
....
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: nginx-app
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
status:
...
```

YAML syntax 1/3

The YAML specification can be found here: https://yaml.org/spec/1.2/spec.html

Most important rules:

- Indentation denotes structure. A nested block is indented with respect to its parent and so on, hierarchically. Indentation must be made with spaces, you cannot use tabs. The number of spaces used to indent a block does not have any meaning but must be used consistently. A typical document indents sub-blocks by 2 spaces.
- Comments are introduced by the pound sign "#" and continue for the rest of the line. They can appear anywhere.
- The elements of a sequence of scalars (strings, numbers) are introduced by a minus sign and a space:

```
- element1
- element2
- element3
```

- Maps are defined separating the key from the value with a colon followed by a space. This is to ensure that strings containing columns don't have to be quoted (e.g. https://localhost):

```
key1: value1
key2: value2
```

- Mapping scalar keys to sequence values combines the two constructs above:

```
key1:
    - value1
    - value2
```

YAML syntax 2/3

- Sequence of mappings:

```
-
  key1: value1
  key2: value2
-
  key1: value3
  key2: value4
```

- In compact form a sequence of mappings can be written as:

```
- key1: value1
  key2: value2
- key1: value3
  key2: value4
```

- Sequences can also be written in square brackets, with comma separated elements:
  `[element1, element2, element3]`
- Mappings can also be written in curly braces, with comma separated `key: value` pairs:
  `{key1: value1, key2: value2}`
- Strings may be quoted. Single quotes don't allow for escape sequences. Double quotes may contain special characters escaped with "\", such as "\n".
- Three dashes may be used to indicate the start of a document ---
- Three dots may be used to indicate the end of a document …

YAML syntax 3/3

- YAML may use tags to define the types of scalars. Tags are introduced by an exclamation mark “!”. However in Kubernetes you will mainly se untagged scalars. A YAML document can use the following data types:

```
my_integer: 10
my_float: 1.23015e+3
my_Boolean: true
my_multiline_string:
  This is the first line
  This is the second line
my_double_quoted_string: " # This is not a comment"
my_single_quoted_string: '"Nested quotes" can go inside single quotes'
my_date: 2021-01-01
```

- Long literals can be written with multiple lines that preserve newlines:

```
my_key: |
  my_value_line_1
  my_value_line_2
  my_value_line_3
```

- Long literals can be written with multiple lines that get folded (newline replaced by space)

```
my_key: >
  This is the beginning
  of a long sentence
  with just spaces between words
```

Creating a deployment using a yaml file

- A Kubernetes descriptor must have the fields:
  - apiVersion
  - kind
  - metadata
  - spec
    - The spec has the fields:
      - selector
      - template
  - To create the deployment using the file, use:
    `kubectl apply –f <file-name.yml>`

- To list deployments do:
  `kubectl get deployments`
- To describe a deployment do:
  `kubectl describe <deployment-name>`
- Describing the deployment does not provide the IP address of the pods, to see that you need to describe each individual pod.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx-app
  minReadySeconds: 5
  template:
    metadata:
      labels:
        app: nginx-app
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

Accessing nginx from a deployment, need to access specific pod ip

```
lara@kube-master-gui:~/k8s/deployments$ kubectl apply -f nginx-deployment.yml
deployment.apps/nginx-deployment created
lara@kube-master-gui:~/k8s/deployments$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6f7d8d4d55-468x9   1/1     Running   0          11s
lara@kube-master-gui:~/k8s/deployments$ kubectl describe pod
Name:         nginx-deployment-6f7d8d4d55-468x9
Namespace:    default
Priority:     0
Node:         kube-master-gui/10.0.2.15
Start Time:   Sun, 03 Jan 2021 16:49:04 +0100
Labels:       app=nginx-app
              pod-template-hash=6f7d8d4d55
Annotations:  cni.projectcalico.org/podIP: 10.1.96.179/32
              cni.projectcalico.org/podIPs: 10.1.96.179/32
Status:       Running
IP:           10.1.96.179
IPs:
  IP:           10.1.96.179
Controlled By:  ReplicaSet/nginx-deployment-6f7d8d4d55
Containers:
  nginx:
    Container ID:   containerd://bc4ea1e2d20c7a20d3bd21a9fd211eb019ae6c27e476f8bd0ba3c6819ce87e5c
    Image:          nginx
    Image ID:       docker.io/library/nginx@sha256:4cf620a5c81390ee209398ecc18e5fb9dd0f5155cd82adcbae532fec94006fb9
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sun, 03 Jan 2021 16:49:06 +0100
    Ready:          True
```

```
lara@kube-master-gui:~/k8s/deployments$ curl -k http://10.1.96.179
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

ReplicaSets

There is a specific ReplicaSet object whose purpose is to maintain the desired number of instances of a pod defined by a template.

However, in general you don't need to create an explicit ReplicaSet object and you can simply specify the desired umber of replicas in a Deployment object.

- *spec.replicas* defines the desired number of replicas
- *spec.selector.matchLabels* defines the logic for identifying which pods are part of this deployment
- *spec.metadata.labels* attaches the label *app: nginx-app* to every pod created from this deployment

```
lara@kube-master-gui:~/k8s/deployments$ kubectl apply -f nginx-3-deployment.yml
deployment.apps/nginx-deployment created
lara@kube-master-gui:~/k8s/deployments$ kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           6s
lara@kube-master-gui:~/k8s/deployments$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6f7d8d4d55-gxb7l   1/1     Running   0          12s
nginx-deployment-6f7d8d4d55-s699f   1/1     Running   0          12s
nginx-deployment-6f7d8d4d55-dggc7   1/1     Running   0          12s
```

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

Scaling the number of pods by editing a deployment

Kubectl edit allows you to interactively modify the yaml definition of an object.

```
kubectl edit deployment <deployment-name>
```

By default kubectl uses the vi editor on Linux. If you prefer the nano editor, set the following environment variable:

```
export KUBE_EDITOR=nano
```

When editing the file, change the number of replicas from 3 to 5.
When saving the edited file, nano proposes a file name that differs from the original file you used to create the deployment. This is fine, accept the default.
Then check how many pods belong to the deployment with a label selector that matches the label specified in the deployment:

```
kubectl get pods -l app=nginx-app
```

```
lara@kube-master-gui:~/k8s/deployments$ kubectl edit deployment nginx-deployment
deployment.apps/nginx-deployment edited
lara@kube-master-gui:~/k8s/deployments$ kubectl get pods -l app=nginx-app
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6f7d8d4d55-gxb7l   1/1     Running   0          5m44s
nginx-deployment-6f7d8d4d55-s699f   1/1     Running   0          5m44s
nginx-deployment-6f7d8d4d55-dggc7   1/1     Running   0          5m44s
nginx-deployment-6f7d8d4d55-ktff2   1/1     Running   0          8s
nginx-deployment-6f7d8d4d55-4dxdt   1/1     Running   0          8s
```

Scaling the number of pods from the command line

You can also scale the number of pods using an imperative command:

```
kubectl scale deployment <deployment-name> --replicas=5
```

```
lara@kube-master-gui:~/k8s/deployments$ kubectl get pods -l app=nginx-app
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6f7d8d4d55-dlfh5   1/1     Running   0          3m18s
nginx-deployment-6f7d8d4d55-kvl2r   1/1     Running   0          3m18s
nginx-deployment-6f7d8d4d55-6b9tz   1/1     Running   0          3m18s
lara@kube-master-gui:~/k8s/deployments$ kubectl scale deployment.apps/nginx-deployment --replicas=5
deployment.apps/nginx-deployment scaled
lara@kube-master-gui:~/k8s/deployments$ kubectl get pods -l app=nginx-app
NAME                                READY   STATUS              RESTARTS   AGE
nginx-deployment-6f7d8d4d55-dlfh5   1/1     Running             0          3m30s
nginx-deployment-6f7d8d4d55-kvl2r   1/1     Running             0          3m30s
nginx-deployment-6f7d8d4d55-6b9tz   1/1     Running             0          3m30s
nginx-deployment-6f7d8d4d55-9r9gf   0/1     ContainerCreating   0          3s
nginx-deployment-6f7d8d4d55-skr94   1/1     Running             0          3s
```

Scaling deployments based on conditions

You can scale deployments automatically based on certain conditions being verified. Example:

```
kubectl autoscale deployment <deployment-name> --min=7 --max=8 --cpu-percent=60
```

This command causes the creation of a HorizontalPodAutoscaler object (hpa).

If the target average CPU utilization (represented as a percent of requested CPU) over all the pods is exceeded, this autoscaler will try to increased the number of pods up to the maximum value.

You can create, delete, get, describe HorizontalPodAutoscaler (hpa) objects.

```
lara@kube-master-gui:~/k8s/deployments$ kubectl get hpa
No resources found in default namespace.
lara@kube-master-gui:~/k8s/deployments$ kubectl autoscale deployment nginx-deployment --min=7 --max=8 --cpu-percent=60
horizontalpodautoscaler.autoscaling/nginx-deployment autoscaled
lara@kube-master-gui:~/k8s/deployments$ kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   5/7     7            5           38m
lara@kube-master-gui:~/k8s/deployments$ kubectl get hpa
NAME               REFERENCE                     TARGETS         MINPODS   MAXPODS   REPLICAS   AGE
nginx-deployment   Deployment/nginx-deployment   <unknown>/60%   7         8         7          19s
lara@kube-master-gui:~/k8s/deployments$ kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   7/7     7            7           38m
```

Deleting deployments

To delete deployments, use the command:

```
kubectl delete deployment <deployment-name>
```

You can then verify that all the pods are automatically deleted.

This is an example of use of Garbage Collector: when the parent of an object no longer exists, the object itself gets deleted automatically.

It takes some times for all the pods to be deleted, and you will see them in Terminating state for some seconds.

```
lara@kube-master-gui:~/k8s/deployments$ kubectl get pods -l app=nginx-app
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6f7d8d4d55-gxb7l   1/1     Running   0          5m44s
nginx-deployment-6f7d8d4d55-s699f   1/1     Running   0          5m44s
nginx-deployment-6f7d8d4d55-dggc7   1/1     Running   0          5m44s
nginx-deployment-6f7d8d4d55-ktff2   1/1     Running   0          8s
nginx-deployment-6f7d8d4d55-4dxdt   1/1     Running   0          8s
lara@kube-master-gui:~/k8s/deployments$ kubectl delete deployment nginx-deployment
deployment.apps "nginx-deployment" deleted
lara@kube-master-gui:~/k8s/deployments$ kubectl get pods -l app=nginx-app
NAME                                READY   STATUS        RESTARTS   AGE
nginx-deployment-6f7d8d4d55-gxb7l   1/1     Terminating   0          14m
nginx-deployment-6f7d8d4d55-s699f   1/1     Terminating   0          14m
nginx-deployment-6f7d8d4d55-dggc7   1/1     Terminating   0          14m
nginx-deployment-6f7d8d4d55-ktff2   1/1     Terminating   0          8m37s
nginx-deployment-6f7d8d4d55-4dxdt   1/1     Terminating   0          8m37s
```

```
lara@kube-master-gui:~/k8s/deployments$ kubectl get pods -l app=nginx-app
No resources found in default namespace.
```

DaemonSet

A DaemonSet ensures that pods run on each node of the cluster, even if nodes are added at a later time.

A typical example of use for a DaemonSet is a process that collects log on each Node of the Cluster.

Fluentd is such a tool that can be used for log collection. On microk8s you can enable the fluentd add-on.

Then you will see that there is a DaemonSet called fluentd-…

```
lara@kube-master-gui:~/k8s/deployments$ microk8s enable fluentd
Enabling Fluentd-Elasticsearch
Labeling nodes
node/kube-master-gui labeled
Addon dns is already enabled.
--allow-privileged=true
service/elasticsearch-logging created
serviceaccount/elasticsearch-logging created
clusterrole.rbac.authorization.k8s.io/elasticsearch-logging created
clusterrolebinding.rbac.authorization.k8s.io/elasticsearch-logging created
statefulset.apps/elasticsearch-logging created
configmap/fluentd-es-config-v0.2.0 created
serviceaccount/fluentd-es created
clusterrole.rbac.authorization.k8s.io/fluentd-es created
clusterrolebinding.rbac.authorization.k8s.io/fluentd-es created
daemonset.apps/fluentd-es-v3.0.2 created
deployment.apps/kibana-logging created
service/kibana-logging created
Fluentd-Elasticsearch is enabled
```

```
lara@kube-master-gui:~/k8s/deployments$ kubectl get daemonset --all-namespaces
NAMESPACE     NAME                DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
kube-system   calico-node         1         1         1       1            1           kubernetes.io/os=linux   3d6h
kube-system   fluentd-es-v3.0.2   1         1         0       1            0           <none>                   2m20s
```

Job

The purpose of the Job object is to ensure that a specific task completes even if the pod that should run the task fails. In the case of standalone pod, if the pod fails it will not be restarted, and so the task won't complete.
A Job solves this problem by ensuring that the task is completed, so if the node where the pod is running fails, the scheduler will launch the pod on another node so that the task can be completed.

Jobs can run pods sequentially or in parallel.

The metadata attribute **parallelism** indicates how many pods should run in parallel.

The metadata attribute **completions** indicates how many pods should run in total before the job is complete.

The metadata attribute **backoffLimit** indicates ho many pods should fail before the job itself is failed permanently.

Job example

```
apiVersion: batch/v1
kind: Job
metadata:
  name: job1
spec:
  template:
    spec:
      containers:
      - name: job1container
        image: alpine
        imagePullPolicy: IfNotPresent
        command: ['ash', '-c', 'echo Job Pod is Running ;
sleep 10']
      restartPolicy: Never
  backoffLimit: 4
  completions: 3
  parallelism: 2
```

```
lara@kube-master-gui:~/k8s/jobs$ kubectl apply -f job1.yml
job.batch/job1 created
lara@kube-master-gui:~/k8s/jobs$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
job1-6mnnl    1/1     Running   0          6s
job1-tl85g    1/1     Running   0          6s
lara@kube-master-gui:~/k8s/jobs$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
job1-6mnnl    1/1     Running   0          11s
job1-tl85g    1/1     Running   0          11s
lara@kube-master-gui:~/k8s/jobs$ kubectl get pods
NAME          READY   STATUS      RESTARTS   AGE
job1-6mnnl    0/1     Completed   0          24s
job1-tl85g    0/1     Completed   0          24s
job1-tkp7q    1/1     Running     0          12s
lara@kube-master-gui:~/k8s/jobs$ kubectl get jobs
NAME   COMPLETIONS   DURATION   AGE
job1   3/3           25s        37s
lara@kube-master-gui:~/k8s/jobs$ kubectl get pods
NAME          READY   STATUS      RESTARTS   AGE
job1-6mnnl    0/1     Completed   0          41s
job1-tl85g    0/1     Completed   0          41s
job1-tkp7q    0/1     Completed   0          29s
```

You can also more specifically execute: `kubectl get pods -l job-name=job1`

CronJob

A CronJob causes a pod to execute at specific times determined by the `schedule` parameter that follows the syntax of Unix cron: https://en.wikipedia.org/wiki/Cron

The cronjob in the example executes every minute.

If you look for the pods you can see that a new pod completes every minute.

Note that CronJob became stable in k8s 1.21.

```yaml
apiVersion: batch/v1
kind: CronJob
metadata:
  name: cronjob1
spec:
  schedule: "*/1 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: cronjob1container
            image: alpine
            imagePullPolicy: IfNotPresent
            command: ['ash', '-c', 'echo Job Pod is Running ; sleep 10']
          restartPolicy: OnFailure
```

```
lara@kube-master-gui:~/k8s/jobs$ kubectl get pods
NAME                          READY   STATUS      RESTARTS   AGE
cronjob1-1609713180-qlbmc     0/1     Completed   0          2m11s
cronjob1-1609713240-h2kw2     0/1     Completed   0          71s
cronjob1-1609713300-m82jl     1/1     Running     0          9s
```

Garbage Collector

- The Garbage Collector is responsible for deleting objects when their parent no longer exists.
- This is implemented by setting the field `ownerReferences` in the child object.
- In this example, a Pod created as part of a Job has `ownerReferences` set to the parent Job.
- **Cascading deletion** causes the child to be deleted when the parent is deleted, and it can happen in **foreground** or **background**.
- In **foreground**, the parent is marked for deletion. Then all objects with `blockOwnerDeletion: true` are deleted and finally, the parent is deleted.
- In **background**, first the parent is deleted and then all the children are found and deleted.

```
lara@kube-master-gui:~/k8s/jobs$ kubectl get pod job1-6mnnl -o yaml
apiVersion: v1
kind: Pod
metadata:
....
  name: job1-6mnnl
  namespace: default
  ownerReferences:
  - apiVersion: batch/v1
    blockOwnerDeletion: true
    controller: true
    kind: Job
    name: job1
    uid: d5de87ae-3f50-445a-bfbd-4b5cb956f2c4
  resourceVersion: "263565"
  selfLink: /api/v1/namespaces/default/pods/job1-6mnnl
  uid: 14d9d3ed-192d-46cc-a0f5-7bd4dc2e6cc4
spec:
  containers:
...
```

HCLSoftware

hcltechsw.com