**Nodes**

Kubernetes nodes by example

In Kubernetes, nodes are the (virtual) machines where your workloads in shape of pods run. As a developer you typically don’t deal with nodes directly, however as an admin you might want to familiarize yourself with node operations.

To list available nodes in your cluster (note that the output will depend on the environment you’re using, I’m using Minishift):

$ kubectl get nodes

NAME STATUS AGE

192.168.99.100 Ready 14d

One interesting task, from a developer point of view, is to make Kubernetes schedule a pod on a certain node. For this, we first need to label the node we want to target:

$ kubectl label nodes 192.168.99.100 shouldrun=here

node "192.168.99.100" labeled

Now we can create a pod that gets scheduled on the node with the label shouldrun=here:

$ kubectl apply -f https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/nodes/pod.yaml

$ kubectl get pods --output=wide

NAME READY STATUS RESTARTS AGE IP NODE

onspecificnode 1/1 Running 0 8s 172.17.0.3 192.168.99.100

To learn more about a specific node, 192.168.99.100 in our case, do:

$ kubectl describe node 192.168.99.100

Name: 192.168.99.100

Labels: beta.kubernetes.io/arch=amd64

beta.kubernetes.io/os=linux

kubernetes.io/hostname=192.168.99.100

shouldrun=here

Taints: <none>

CreationTimestamp: Wed, 12 Apr 2017 17:17:13 +0100

Phase:

Conditions:

Type Status LastHeartbeatTime LastTransitionTime Reason Message

---- ------ ----------------- ------------------ ------ -------

OutOfDisk False Thu, 27 Apr 2017 14:55:49 +0100 Thu, 27 Apr 2017 09:18:13 +0100 KubeletHasSufficientDisk kubelet has sufficient disk space available

MemoryPressure False Thu, 27 Apr 2017 14:55:49 +0100 Wed, 12 Apr 2017 17:17:13 +0100 KubeletHasSufficientMemory kubelet has sufficient memory available

DiskPressure False Thu, 27 Apr 2017 14:55:49 +0100 Wed, 12 Apr 2017 17:17:13 +0100 KubeletHasNoDiskPressure kubelet has no disk pressure

Ready True Thu, 27 Apr 2017 14:55:49 +0100 Thu, 27 Apr 2017 09:18:24 +0100 KubeletReady kubelet is posting ready status

Addresses: 192.168.99.100,192.168.99.100,192.168.99.100

Capacity:

alpha.kubernetes.io/nvidia-gpu: 0

cpu: 2

memory: 2050168Ki

pods: 20

Allocatable:

alpha.kubernetes.io/nvidia-gpu: 0

cpu: 2

memory: 2050168Ki

pods: 20

System Info:

Machine ID: 896b6d970**cd**14d158be1fd1c31ff1a8a

System UUID: F7771C31-30B0-44EC-8364-B3517DBC8767

Boot ID: 1d589b36-3413-4e82-af80-b2756342eed4

Kernel Version: 4.4.27-boot2docker

OS Image: CentOS Linux 7 (Core)

Operating System: linux

Architecture: amd64

Container Runtime Version: docker://1.12.3

Kubelet Version: v1.5.2+43a9be4

Kube-Proxy Version: v1.5.2+43a9be4

ExternalID: 192.168.99.100

Non-terminated Pods: (3 **in** total)

Namespace Name CPU Requests CPU Limits Memory Requests Memory Limits

--------- ---- ------------ ---------- --------------- -------------

default docker-registry-1-hfpzp 100m (5%) 0 (0%) 256Mi (12%) 0 (0%)

default onspecificnode 0 (0%) 0 (0%) 0 (0%) 0 (0%)

default router-1-cdglk 100m (5%) 0 (0%) 256Mi (12%) 0 (0%)

Allocated resources:

(Total limits may be over 100 percent, i.e., overcommitted.

CPU Requests CPU Limits Memory Requests Memory Limits

------------ ---------- --------------- -------------

200m (10%) 0 (0%) 512Mi (25%) 0 (0%)

No events.

Note that there are more sophisticated methods than shown above, such as using affinity, to [assign pods to nodes](https://kubernetes.io/docs/concepts/configuration/assign-pod-node/) and depending on your use case, you might want to check those out as well.