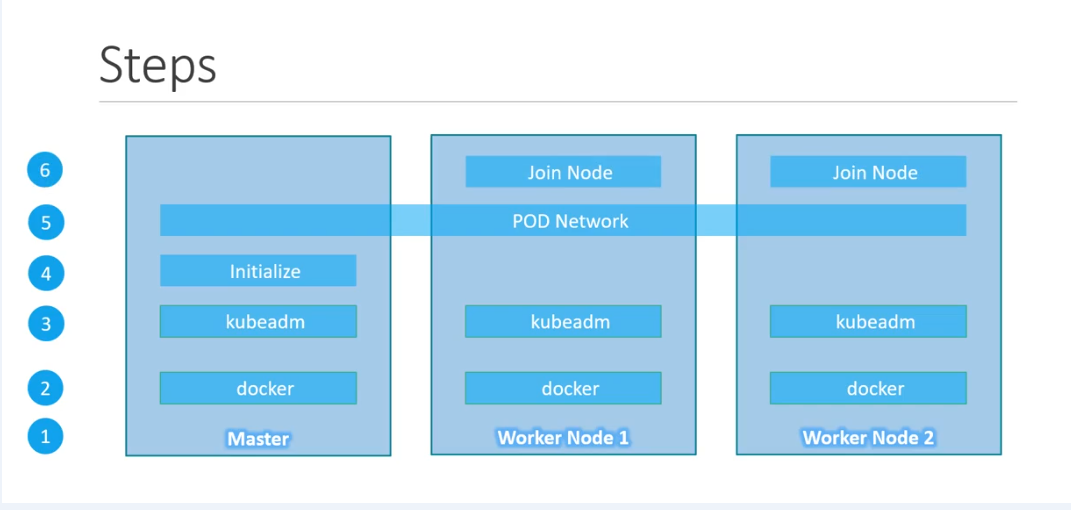
<https://docs.docker.com/engine/install/ubuntu/>

<https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/#verify-the-mac-address-and-product_uuid-are-unique-for-every-node>



To remove old docker pkg

sudo apt-get remove docker docker-engine docker.io containerd runc

sudo apt-get update –y

sudo apt-get install \

apt-transport-https \

ca-certificates \

curl \

gnupg-agent \

software-properties-common

Add Docker’s official GPG key:

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo apt-key fingerprint 0EBFCD88

sudo add-apt-repository \

"deb [arch=amd64] https://download.docker.com/linux/ubuntu \

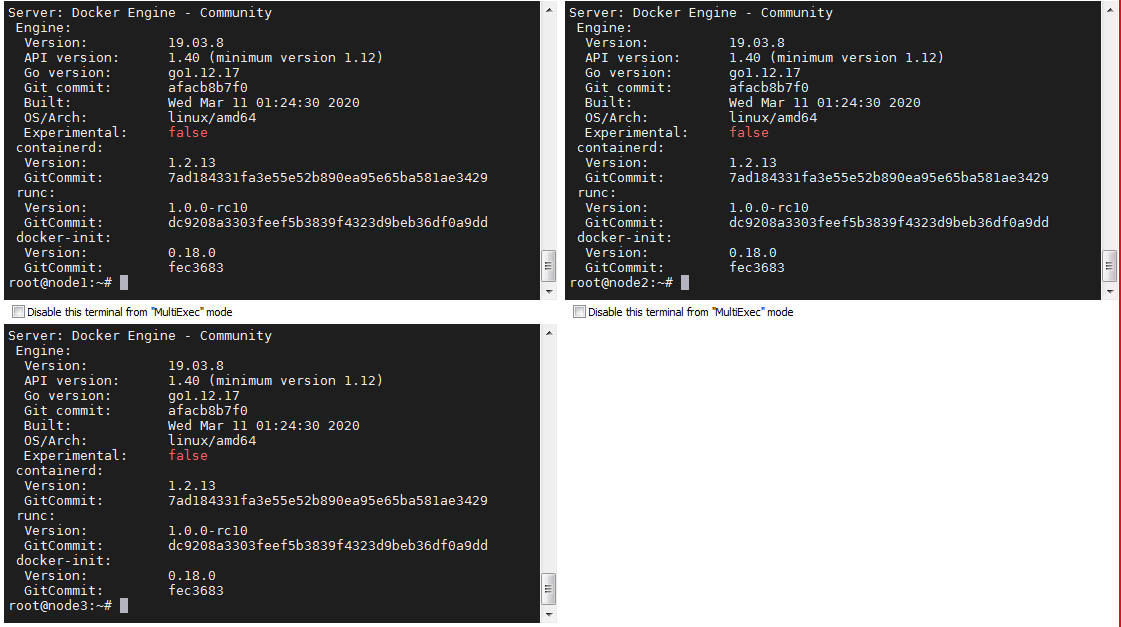
$(lsb\_release -cs) \

stable"

$ sudo apt-get update

$ sudo apt-get install docker-ce docker-ce-cli containerd.io

apt-cache madison docker-ce



<https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/#verify-the-mac-address-and-product_uuid-are-unique-for-every-node>

Kubedm installation

sudo apt-get update && sudo apt-get install -y apt-transport-https curl

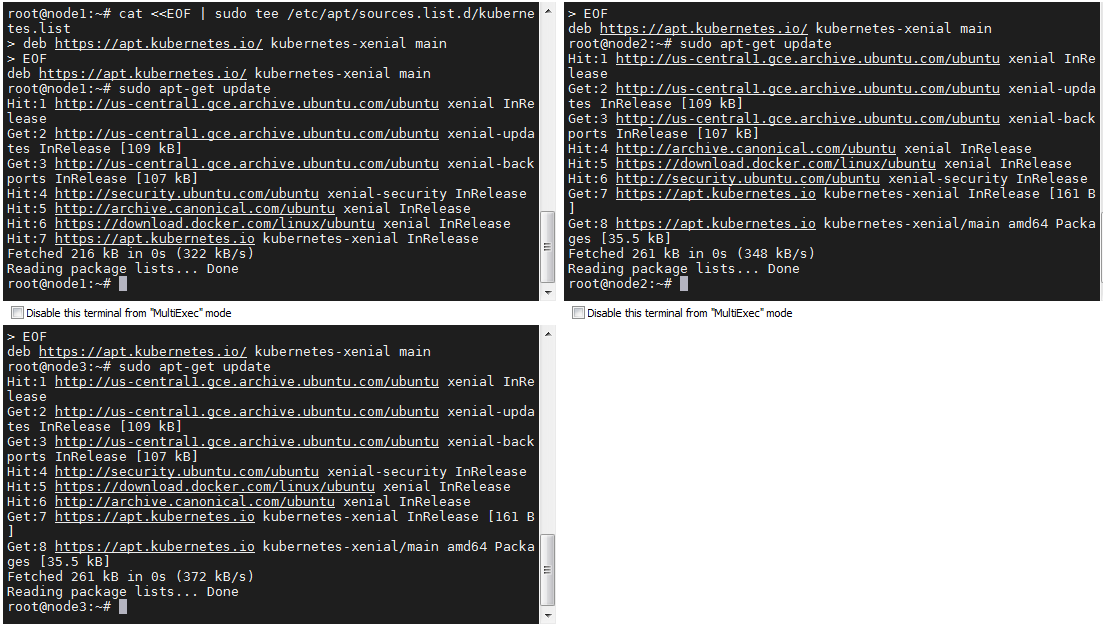
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list

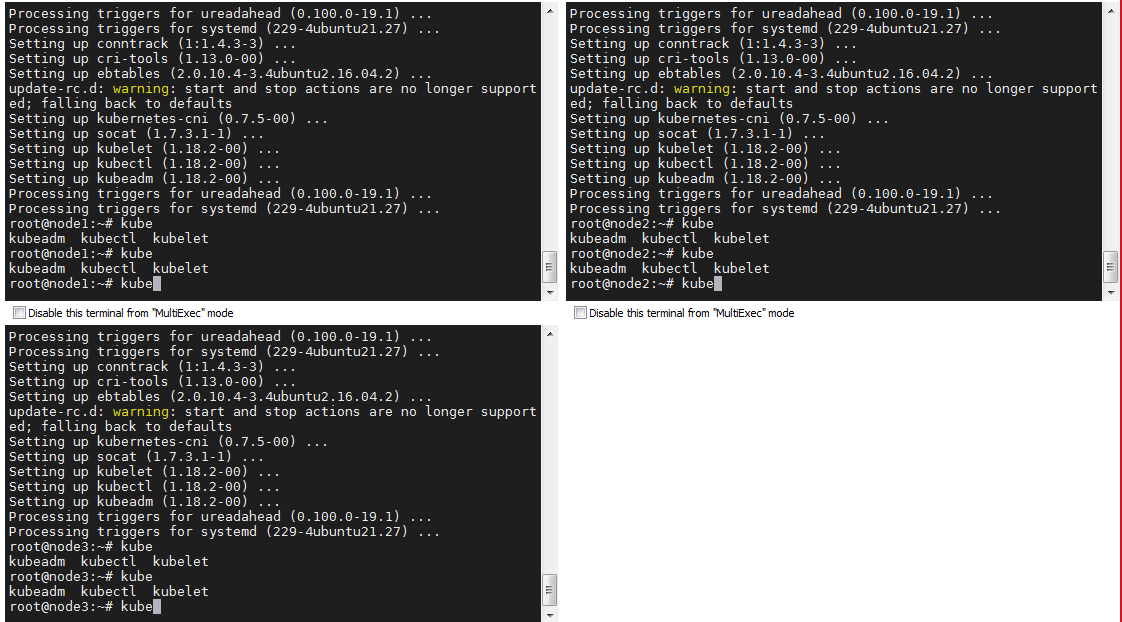
deb https://apt.kubernetes.io/ kubernetes-xenial main

EOF

sudo apt-get update



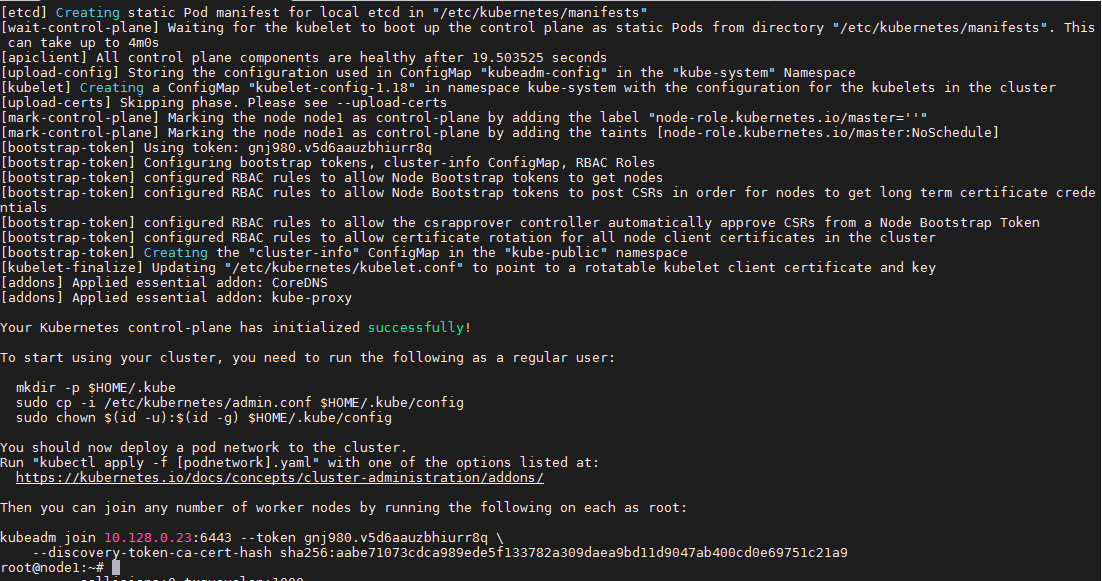
sudo apt-get install -y kubelet kubeadm kubectl



<https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/>

run below command on master node only

kubeadm init --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-address=10.128.0.23



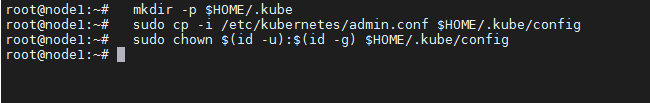
To start using your cluster, you need to run the following as a regular user:

**Run below commands on master only**

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

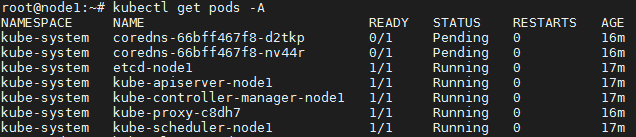
sudo chown $(id -u):$(id -g) $HOME/.kube/config



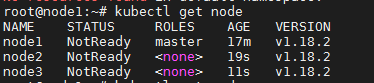
Install Flannel for pod networking

Before running the yaml file

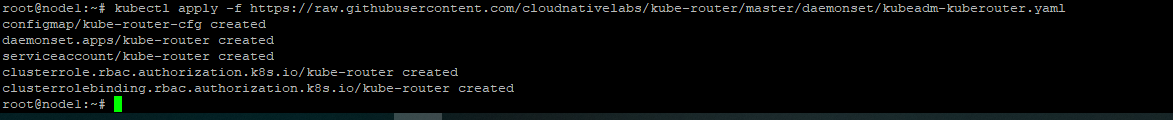
kubectl get pods –A



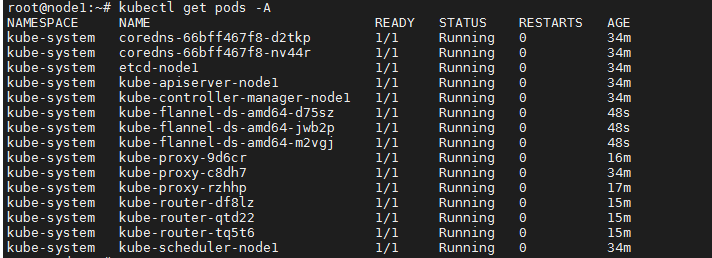
kubectl get node

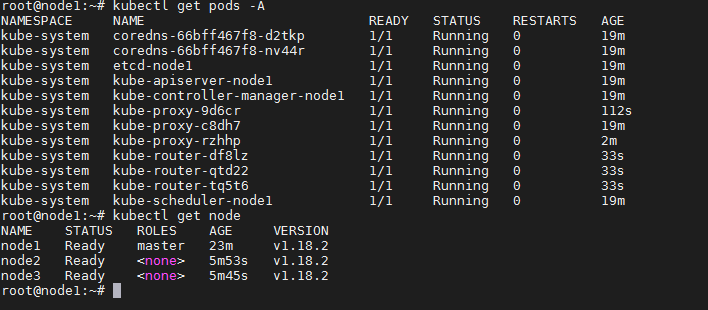


kubectl apply -f https://raw.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm-kuberouter.yaml



kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml





**Run this commands on workers node**

kubeadm join 10.128.0.23:6443 --token gnj980.v5d6aauzbhiurr8q \

--discovery-token-ca-cert-hash sha256:aabe71073cdca989ede5f133782a309daea9bd11d9047ab400cd0e69751c21a9

root@node1:~# kubectl get no

NAME STATUS ROLES AGE VERSION

node1 Ready master 38m v1.18.2

node2 Ready <none> 21m v1.18.2

node3 Ready <none> 20m v1.18.2

root@node1:~# kubectl get no -o wide

NAME STATUS ROLES AGE VERSION INTERNAL-IP EXTERNAL-IP OS-IMAGE KERNEL-VERSION CONTAINER-RUNTIME

node1 Ready master 38m v1.18.2 10.128.0.23 <none> Ubuntu 16.04.6 LTS 4.15.0-1061-gcp docker://19.3.8

node2 Ready <none> 21m v1.18.2 10.128.0.24 <none> Ubuntu 16.04.6 LTS 4.15.0-1061-gcp docker://19.3.8

node3 Ready <none> 21m v1.18.2 10.128.0.25 <none> Ubuntu 16.04.6 LTS 4.15.0-1061-gcp docker://19.3.8

root@node1:~# kubectl describe no

Name: node1

Roles: master

Labels: beta.kubernetes.io/arch=amd64

beta.kubernetes.io/os=linux

kubernetes.io/arch=amd64

kubernetes.io/hostname=node1

kubernetes.io/os=linux

node-role.kubernetes.io/master=

Annotations: flannel.alpha.coreos.com/backend-data: {"VtepMAC":"2e:1d:39:30:82:8b"}

flannel.alpha.coreos.com/backend-type: vxlan

flannel.alpha.coreos.com/kube-subnet-manager: true

flannel.alpha.coreos.com/public-ip: 10.128.0.23

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: Sun, 17 May 2020 06:46:46 +0000

Taints: node-role.kubernetes.io/master:NoSchedule

Unschedulable: false

Lease:

HolderIdentity: node1

AcquireTime: <unset>

RenewTime: Sun, 17 May 2020 07:25:42 +0000

Conditions:

Type Status LastHeartbeatTime LastTransitionTime Reason Message

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

NetworkUnavailable False Sun, 17 May 2020 07:20:45 +0000 Sun, 17 May 2020 07:20:45 +0000 FlannelIsUp Flannel is running on this node

MemoryPressure False Sun, 17 May 2020 07:20:58 +0000 Sun, 17 May 2020 06:46:43 +0000 KubeletHasSufficientMemory kubelet has sufficient memory available

DiskPressure False Sun, 17 May 2020 07:20:58 +0000 Sun, 17 May 2020 06:46:43 +0000 KubeletHasNoDiskPressure kubelet has no disk pressure

PIDPressure False Sun, 17 May 2020 07:20:58 +0000 Sun, 17 May 2020 06:46:43 +0000 KubeletHasSufficientPID kubelet has sufficient PID available

Ready True Sun, 17 May 2020 07:20:58 +0000 Sun, 17 May 2020 07:06:05 +0000 KubeletReady kubelet is posting ready status. AppArmor enabled

Addresses:

InternalIP: 10.128.0.23

Hostname: node1

Capacity:

cpu: 2

ephemeral-storage: 20145768Ki

hugepages-1Gi: 0

hugepages-2Mi: 0

memory: 2040112Ki

pods: 110

Allocatable:

cpu: 2

ephemeral-storage: 18566339759

hugepages-1Gi: 0

hugepages-2Mi: 0

memory: 1937712Ki

pods: 110

System Info:

Machine ID: e592b69248039eddf2617bef3ee1cbe7

System UUID: E592B692-4803-9EDD-F261-7BEF3EE1CBE7

Boot ID: 73bd4219-1b62-4a78-8ca9-87f6ee71389a

Kernel Version: 4.15.0-1061-gcp

OS Image: Ubuntu 16.04.6 LTS

Operating System: linux

Architecture: amd64

Container Runtime Version: docker://19.3.8

Kubelet Version: v1.18.2

Kube-Proxy Version: v1.18.2

PodCIDR: 10.244.0.0/24

PodCIDRs: 10.244.0.0/24

Non-terminated Pods: (7 in total)

Namespace Name CPU Requests CPU Limits Memory Requests Memory Limits AGE

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

kube-system etcd-node1 0 (0%) 0 (0%) 0 (0%) 0 (0%) 38m

kube-system kube-apiserver-node1 250m (12%) 0 (0%) 0 (0%) 0 (0%) 38m

kube-system kube-controller-manager-node1 200m (10%) 0 (0%) 0 (0%) 0 (0%) 38m

kube-system kube-flannel-ds-amd64-m2vgj 100m (5%) 100m (5%) 50Mi (2%) 50Mi (2%) 5m12s

kube-system kube-proxy-c8dh7 0 (0%) 0 (0%) 0 (0%) 0 (0%) 38m

kube-system kube-router-df8lz 250m (12%) 0 (0%) 250Mi (13%) 0 (0%) 20m

kube-system kube-scheduler-node1 100m (5%) 0 (0%) 0 (0%) 0 (0%) 38m

Allocated resources:

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

Resource Requests Limits

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

cpu 900m (45%) 100m (5%)

memory 300Mi (15%) 50Mi (2%)

ephemeral-storage 0 (0%) 0 (0%)

hugepages-1Gi 0 (0%) 0 (0%)

hugepages-2Mi 0 (0%) 0 (0%)

Events:

Type Reason Age From Message

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

Normal NodeHasSufficientMemory 39m (x5 over 39m) kubelet, node1 Node node1 status is now: NodeHasSufficientMemory

Normal NodeHasNoDiskPressure 39m (x5 over 39m) kubelet, node1 Node node1 status is now: NodeHasNoDiskPressure

Normal NodeHasSufficientPID 39m (x4 over 39m) kubelet, node1 Node node1 status is now: NodeHasSufficientPID

Normal Starting 39m kubelet, node1 Starting kubelet.

Warning ImageGCFailed 39m kubelet, node1 failed to get imageFs info: unable to find data in memory cache

Normal NodeHasSufficientMemory 38m kubelet, node1 Node node1 status is now: NodeHasSufficientMemory

Normal NodeHasNoDiskPressure 38m kubelet, node1 Node node1 status is now: NodeHasNoDiskPressure

Normal NodeHasSufficientPID 38m kubelet, node1 Node node1 status is now: NodeHasSufficientPID

Normal NodeAllocatableEnforced 38m kubelet, node1 Updated Node Allocatable limit across pods

Normal Starting 38m kube-proxy, node1 Starting kube-proxy.

Normal NodeReady 19m kubelet, node1 Node node1 status is now: NodeReady

Name: node2

Roles: <none>

Labels: beta.kubernetes.io/arch=amd64

beta.kubernetes.io/os=linux

kubernetes.io/arch=amd64

kubernetes.io/hostname=node2

kubernetes.io/os=linux

Annotations: flannel.alpha.coreos.com/backend-data: {"VtepMAC":"12:fe:64:7b:9a:1b"}

flannel.alpha.coreos.com/backend-type: vxlan

flannel.alpha.coreos.com/kube-subnet-manager: true

flannel.alpha.coreos.com/public-ip: 10.128.0.24

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: Sun, 17 May 2020 07:04:19 +0000

Taints: <none>

Unschedulable: false

Lease:

HolderIdentity: node2

AcquireTime: <unset>

RenewTime: Sun, 17 May 2020 07:25:49 +0000

Conditions:

Type Status LastHeartbeatTime LastTransitionTime Reason Message

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

NetworkUnavailable False Sun, 17 May 2020 07:20:44 +0000 Sun, 17 May 2020 07:20:44 +0000 FlannelIsUp Flannel is running on this node

MemoryPressure False Sun, 17 May 2020 07:20:52 +0000 Sun, 17 May 2020 07:04:19 +0000 KubeletHasSufficientMemory kubelet has sufficient memory available

DiskPressure False Sun, 17 May 2020 07:20:52 +0000 Sun, 17 May 2020 07:04:19 +0000 KubeletHasNoDiskPressure kubelet has no disk pressure

PIDPressure False Sun, 17 May 2020 07:20:52 +0000 Sun, 17 May 2020 07:04:19 +0000 KubeletHasSufficientPID kubelet has sufficient PID available

Ready True Sun, 17 May 2020 07:20:52 +0000 Sun, 17 May 2020 07:05:59 +0000 KubeletReady kubelet is posting ready status. AppArmor enabled

Addresses:

InternalIP: 10.128.0.24

Hostname: node2

Capacity:

cpu: 2

ephemeral-storage: 20145768Ki

hugepages-1Gi: 0

hugepages-2Mi: 0

memory: 2040112Ki

pods: 110

Allocatable:

cpu: 2

ephemeral-storage: 18566339759

hugepages-1Gi: 0

hugepages-2Mi: 0

memory: 1937712Ki

pods: 110

System Info:

Machine ID: f896982b06eef09e1dfab5c1102f8329

System UUID: F896982B-06EE-F09E-1DFA-B5C1102F8329

Boot ID: 494d5599-dd76-4ba1-b8fb-9d48d83d8db6

Kernel Version: 4.15.0-1061-gcp

OS Image: Ubuntu 16.04.6 LTS

Operating System: linux

Architecture: amd64

Container Runtime Version: docker://19.3.8

Kubelet Version: v1.18.2

Kube-Proxy Version: v1.18.2

PodCIDR: 10.244.2.0/24

PodCIDRs: 10.244.2.0/24

Non-terminated Pods: (4 in total)

Namespace Name CPU Requests CPU Limits Memory Requests Memory Limits AGE

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

kube-system coredns-66bff467f8-d2tkp 100m (5%) 0 (0%) 70Mi (3%) 170Mi (8%) 38m

kube-system kube-flannel-ds-amd64-jwb2p 100m (5%) 100m (5%) 50Mi (2%) 50Mi (2%) 5m12s

kube-system kube-proxy-rzhhp 0 (0%) 0 (0%) 0 (0%) 0 (0%) 21m

kube-system kube-router-tq5t6 250m (12%) 0 (0%) 250Mi (13%) 0 (0%) 20m

Allocated resources:

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

Resource Requests Limits

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

cpu 450m (22%) 100m (5%)

memory 370Mi (19%) 220Mi (11%)

ephemeral-storage 0 (0%) 0 (0%)

hugepages-1Gi 0 (0%) 0 (0%)

hugepages-2Mi 0 (0%) 0 (0%)

Events:

Type Reason Age From Message

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

Normal Starting 21m kubelet, node2 Starting kubelet.

Normal NodeHasSufficientMemory 21m kubelet, node2 Node node2 status is now: NodeHasSufficientMemory

Normal NodeHasNoDiskPressure 21m kubelet, node2 Node node2 status is now: NodeHasNoDiskPressure

Normal NodeHasSufficientPID 21m kubelet, node2 Node node2 status is now: NodeHasSufficientPID

Normal NodeAllocatableEnforced 21m kubelet, node2 Updated Node Allocatable limit across pods

Normal Starting 21m kube-proxy, node2 Starting kube-proxy.

Normal NodeReady 19m kubelet, node2 Node node2 status is now: NodeReady

Name: node3

Roles: <none>

Labels: beta.kubernetes.io/arch=amd64

beta.kubernetes.io/os=linux

kubernetes.io/arch=amd64

kubernetes.io/hostname=node3

kubernetes.io/os=linux

Annotations: flannel.alpha.coreos.com/backend-data: {"VtepMAC":"7e:46:38:23:ef:6f"}

flannel.alpha.coreos.com/backend-type: vxlan

flannel.alpha.coreos.com/kube-subnet-manager: true

flannel.alpha.coreos.com/public-ip: 10.128.0.25

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: Sun, 17 May 2020 07:04:27 +0000

Taints: <none>

Unschedulable: false

Lease:

HolderIdentity: node3

AcquireTime: <unset>

RenewTime: Sun, 17 May 2020 07:25:48 +0000

Conditions:

Type Status LastHeartbeatTime LastTransitionTime Reason Message

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

NetworkUnavailable False Sun, 17 May 2020 07:20:45 +0000 Sun, 17 May 2020 07:20:45 +0000 FlannelIsUp Flannel is running on this node

MemoryPressure False Sun, 17 May 2020 07:21:01 +0000 Sun, 17 May 2020 07:04:27 +0000 KubeletHasSufficientMemory kubelet has sufficient memory available

DiskPressure False Sun, 17 May 2020 07:21:01 +0000 Sun, 17 May 2020 07:04:27 +0000 KubeletHasNoDiskPressure kubelet has no disk pressure

PIDPressure False Sun, 17 May 2020 07:21:01 +0000 Sun, 17 May 2020 07:04:27 +0000 KubeletHasSufficientPID kubelet has sufficient PID available

Ready True Sun, 17 May 2020 07:21:01 +0000 Sun, 17 May 2020 07:05:58 +0000 KubeletReady kubelet is posting ready status. AppArmor enabled

Addresses:

InternalIP: 10.128.0.25

Hostname: node3

Capacity:

cpu: 2

ephemeral-storage: 20145768Ki

hugepages-1Gi: 0

hugepages-2Mi: 0

memory: 2040112Ki

pods: 110

Allocatable:

cpu: 2

ephemeral-storage: 18566339759

hugepages-1Gi: 0

hugepages-2Mi: 0

memory: 1937712Ki

pods: 110

System Info:

Machine ID: b779989ed57bcf7735f8b25a6d956d6a

System UUID: B779989E-D57B-CF77-35F8-B25A6D956D6A

Boot ID: a364898b-fc2d-48b0-86f1-ce43d584ab84

Kernel Version: 4.15.0-1061-gcp

OS Image: Ubuntu 16.04.6 LTS

Operating System: linux

Architecture: amd64

Container Runtime Version: docker://19.3.8

Kubelet Version: v1.18.2

Kube-Proxy Version: v1.18.2

PodCIDR: 10.244.3.0/24

PodCIDRs: 10.244.3.0/24

Non-terminated Pods: (4 in total)

Namespace Name CPU Requests CPU Limits Memory Requests Memory Limits AGE

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

kube-system coredns-66bff467f8-nv44r 100m (5%) 0 (0%) 70Mi (3%) 170Mi (8%) 38m

kube-system kube-flannel-ds-amd64-d75sz 100m (5%) 100m (5%) 50Mi (2%) 50Mi (2%) 5m12s

kube-system kube-proxy-9d6cr 0 (0%) 0 (0%) 0 (0%) 0 (0%) 21m

kube-system kube-router-qtd22 250m (12%) 0 (0%) 250Mi (13%) 0 (0%) 20m

Allocated resources:

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

Resource Requests Limits

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

cpu 450m (22%) 100m (5%)

memory 370Mi (19%) 220Mi (11%)

ephemeral-storage 0 (0%) 0 (0%)

hugepages-1Gi 0 (0%) 0 (0%)

hugepages-2Mi 0 (0%) 0 (0%)

Events:

Type Reason Age From Message

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

Normal Starting 21m kubelet, node3 Starting kubelet.

Normal NodeHasSufficientMemory 21m (x2 over 21m) kubelet, node3 Node node3 status is now: NodeHasSufficientMemory

Normal NodeHasNoDiskPressure 21m (x2 over 21m) kubelet, node3 Node node3 status is now: NodeHasNoDiskPressure

Normal NodeHasSufficientPID 21m (x2 over 21m) kubelet, node3 Node node3 status is now: NodeHasSufficientPID

Normal NodeAllocatableEnforced 21m kubelet, node3 Updated Node Allocatable limit across pods

Normal Starting 21m kube-proxy, node3 Starting kube-proxy.

Normal NodeReady 19m kubelet, node3 Node node3 status is now: NodeReady

root@node1:~#

root@node1:~# kubectl get no -o yaml

apiVersion: v1

items:

- apiVersion: v1

kind: Node

metadata:

annotations:

flannel.alpha.coreos.com/backend-data: '{"VtepMAC":"2e:1d:39:30:82:8b"}'

flannel.alpha.coreos.com/backend-type: vxlan

flannel.alpha.coreos.com/kube-subnet-manager: "true"

flannel.alpha.coreos.com/public-ip: 10.128.0.23

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: "2020-05-17T06:46:46Z"

labels:

beta.kubernetes.io/arch: amd64

beta.kubernetes.io/os: linux

kubernetes.io/arch: amd64

kubernetes.io/hostname: node1

kubernetes.io/os: linux

node-role.kubernetes.io/master: ""

managedFields:

- apiVersion: v1

fieldsType: FieldsV1

fieldsV1:

f:metadata:

f:annotations:

f:kubeadm.alpha.kubernetes.io/cri-socket: {}

f:labels:

f:node-role.kubernetes.io/master: {}

manager: kubeadm

operation: Update

time: "2020-05-17T06:46:49Z"

- apiVersion: v1

fieldsType: FieldsV1

fieldsV1:

f:metadata:

f:annotations:

f:node.alpha.kubernetes.io/ttl: {}

f:spec:

f:podCIDR: {}

f:podCIDRs:

.: {}

v:"10.244.0.0/24": {}

f:taints: {}

manager: kube-controller-manager

operation: Update

time: "2020-05-17T07:06:05Z"

- apiVersion: v1

fieldsType: FieldsV1

fieldsV1:

f:metadata:

f:annotations:

f:flannel.alpha.coreos.com/backend-data: {}

f:flannel.alpha.coreos.com/backend-type: {}

f:flannel.alpha.coreos.com/kube-subnet-manager: {}

f:flannel.alpha.coreos.com/public-ip: {}

f:status:

f:conditions:

k:{"type":"NetworkUnavailable"}:

.: {}

f:lastHeartbeatTime: {}

f:lastTransitionTime: {}

f:message: {}

f:reason: {}

f:status: {}

f:type: {}

manager: flanneld

operation: Update

time: "2020-05-17T07:20:45Z"

- apiVersion: v1

fieldsType: FieldsV1

fieldsV1:

f:metadata:

f:annotations:

.: {}

f:volumes.kubernetes.io/controller-managed-attach-detach: {}

f:labels:

.: {}

f:beta.kubernetes.io/arch: {}

f:beta.kubernetes.io/os: {}

f:kubernetes.io/arch: {}

f:kubernetes.io/hostname: {}

f:kubernetes.io/os: {}

f:status:

f:addresses:

.: {}

k:{"type":"Hostname"}:

.: {}

f:address: {}

f:type: {}

k:{"type":"InternalIP"}:

.: {}

f:address: {}

f:type: {}

f:allocatable:

.: {}

f:cpu: {}

f:ephemeral-storage: {}

f:hugepages-1Gi: {}

f:hugepages-2Mi: {}

f:memory: {}

f:pods: {}

f:capacity:

.: {}

f:cpu: {}

f:ephemeral-storage: {}

f:hugepages-1Gi: {}

f:hugepages-2Mi: {}

f:memory: {}

f:pods: {}

f:conditions:

.: {}

k:{"type":"DiskPressure"}:

.: {}

f:lastHeartbeatTime: {}

f:lastTransitionTime: {}

f:message: {}

f:reason: {}

f:status: {}

f:type: {}

k:{"type":"MemoryPressure"}:

.: {}

f:lastHeartbeatTime: {}

f:lastTransitionTime: {}

f:message: {}

f:reason: {}

f:status: {}

f:type: {}

k:{"type":"PIDPressure"}:

.: {}

f:lastHeartbeatTime: {}

f:lastTransitionTime: {}

f:message: {}

f:reason: {}

f:status: {}

f:type: {}

k:{"type":"Ready"}:

.: {}

f:lastHeartbeatTime: {}

f:lastTransitionTime: {}

f:message: {}

f:reason: {}

f:status: {}

f:type: {}

f:daemonEndpoints:

f:kubeletEndpoint:

f:Port: {}

f:images: {}

f:nodeInfo:

f:architecture: {}

f:bootID: {}

f:containerRuntimeVersion: {}

f:kernelVersion: {}

f:kubeProxyVersion: {}

f:kubeletVersion: {}

f:machineID: {}

f:operatingSystem: {}

f:osImage: {}

f:systemUUID: {}

manager: kubelet

operation: Update

time: "2020-05-17T07:25:59Z"

name: node1

resourceVersion: "6062"

selfLink: /api/v1/nodes/node1

uid: 7d2e5abe-3d2c-440f-a614-5bc1b5890efd

spec:

podCIDR: 10.244.0.0/24

podCIDRs:

- 10.244.0.0/24

taints:

- effect: NoSchedule

key: node-role.kubernetes.io/master

status:

addresses:

- address: 10.128.0.23

type: InternalIP

- address: node1

type: Hostname

allocatable:

cpu: "2"

ephemeral-storage: "18566339759"

hugepages-1Gi: "0"

hugepages-2Mi: "0"

memory: 1937712Ki

pods: "110"

capacity:

cpu: "2"

ephemeral-storage: 20145768Ki

hugepages-1Gi: "0"

hugepages-2Mi: "0"

memory: 2040112Ki

pods: "110"

conditions:

- lastHeartbeatTime: "2020-05-17T07:20:45Z"

lastTransitionTime: "2020-05-17T07:20:45Z"

message: Flannel is running on this node

reason: FlannelIsUp

status: "False"

type: NetworkUnavailable

- lastHeartbeatTime: "2020-05-17T07:25:59Z"

lastTransitionTime: "2020-05-17T06:46:43Z"

message: kubelet has sufficient memory available

reason: KubeletHasSufficientMemory

status: "False"

type: MemoryPressure

- lastHeartbeatTime: "2020-05-17T07:25:59Z"

lastTransitionTime: "2020-05-17T06:46:43Z"

message: kubelet has no disk pressure

reason: KubeletHasNoDiskPressure

status: "False"

type: DiskPressure

- lastHeartbeatTime: "2020-05-17T07:25:59Z"

lastTransitionTime: "2020-05-17T06:46:43Z"

message: kubelet has sufficient PID available

reason: KubeletHasSufficientPID

status: "False"

type: PIDPressure

- lastHeartbeatTime: "2020-05-17T07:25:59Z"

lastTransitionTime: "2020-05-17T07:06:05Z"

message: kubelet is posting ready status. AppArmor enabled

reason: KubeletReady

status: "True"

type: Ready

daemonEndpoints:

kubeletEndpoint:

Port: 10250

images:

- names:

- k8s.gcr.io/etcd@sha256:4afb99b4690b418ffc2ceb67e1a17376457e441c1f09ab55447f0aaf992fa646

- k8s.gcr.io/etcd:3.4.3-0

sizeBytes: 288426917

- names:

- k8s.gcr.io/kube-apiserver@sha256:19a8020e4aaaa8bd41f5bca223e05183cfe66157393ef7a205720c49b2405e0f

- k8s.gcr.io/kube-apiserver:v1.18.2

sizeBytes: 173001235

- names:

- k8s.gcr.io/kube-controller-manager@sha256:2d0f92f1a0e58ec8ccee866de76173d81e7d07ed874775cd3dc27d91c97ae9b3

- k8s.gcr.io/kube-controller-manager:v1.18.2

sizeBytes: 162388499

- names:

- k8s.gcr.io/kube-proxy@sha256:8bfd5a33756f4eb95a8598cbef083ad7a59f40c09dccf4ba692b6b91e24cc9c1

- k8s.gcr.io/kube-proxy:v1.18.2

sizeBytes: 116538359

- names:

- cloudnativelabs/kube-router@sha256:b81714cbaa03f8e45859e3b40c006a6822bbc73d07e65d8ef6c17b5f553658c3

- cloudnativelabs/kube-router:latest

sizeBytes: 102428036

- names:

- k8s.gcr.io/kube-scheduler@sha256:69f90a33b64c99e4c78e3cae36b0c767729b5a54203aa35524b1033708d1b482

- k8s.gcr.io/kube-scheduler:v1.18.2

sizeBytes: 95279635

- names:

- quay.io/coreos/flannel@sha256:6d451d92c921f14bfb38196aacb6e506d4593c5b3c9d40a8b8a2506010dc3e10

- quay.io/coreos/flannel:v0.12.0-amd64

sizeBytes: 52767393

- names:

- k8s.gcr.io/coredns@sha256:2c8d61c46f484d881db43b34d13ca47a269336e576c81cf007ca740fa9ec0800

- k8s.gcr.io/coredns:1.6.7

sizeBytes: 43794147

- names:

- k8s.gcr.io/pause@sha256:927d98197ec1141a368550822d18fa1c60bdae27b78b0c004f705f548c07814f

- k8s.gcr.io/pause:3.2

sizeBytes: 682696

nodeInfo:

architecture: amd64

bootID: 73bd4219-1b62-4a78-8ca9-87f6ee71389a

containerRuntimeVersion: docker://19.3.8

kernelVersion: 4.15.0-1061-gcp

kubeProxyVersion: v1.18.2

kubeletVersion: v1.18.2

machineID: e592b69248039eddf2617bef3ee1cbe7

operatingSystem: linux

osImage: Ubuntu 16.04.6 LTS

systemUUID: E592B692-4803-9EDD-F261-7BEF3EE1CBE7

- apiVersion: v1

kind: Node

metadata:

annotations:

flannel.alpha.coreos.com/backend-data: '{"VtepMAC":"12:fe:64:7b:9a:1b"}'

flannel.alpha.coreos.com/backend-type: vxlan

flannel.alpha.coreos.com/kube-subnet-manager: "true"

flannel.alpha.coreos.com/public-ip: 10.128.0.24

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: "2020-05-17T07:04:19Z"

labels:

beta.kubernetes.io/arch: amd64

beta.kubernetes.io/os: linux

kubernetes.io/arch: amd64

kubernetes.io/hostname: node2

kubernetes.io/os: linux

managedFields:

- apiVersion: v1

fieldsType: FieldsV1

fieldsV1:

f:metadata:

f:annotations:

f:node.alpha.kubernetes.io/ttl: {}

f:spec:

f:podCIDR: {}

f:podCIDRs:

.: {}

v:"10.244.2.0/24": {}

manager: kube-controller-manager

operation: Update

time: "2020-05-17T07:04:19Z"

- apiVersion: v1

fieldsType: FieldsV1

fieldsV1:

f:metadata:

f:annotations:

f:kubeadm.alpha.kubernetes.io/cri-socket: {}

manager: kubeadm

operation: Update

time: "2020-05-17T07:04:19Z"

- apiVersion: v1

fieldsType: FieldsV1

fieldsV1:

f:metadata:

f:annotations:

f:flannel.alpha.coreos.com/backend-data: {}

f:flannel.alpha.coreos.com/backend-type: {}

f:flannel.alpha.coreos.com/kube-subnet-manager: {}

f:flannel.alpha.coreos.com/public-ip: {}

f:status:

f:conditions:

k:{"type":"NetworkUnavailable"}:

.: {}

f:lastHeartbeatTime: {}

f:lastTransitionTime: {}

f:message: {}

f:reason: {}

f:status: {}

f:type: {}

manager: flanneld

operation: Update

time: "2020-05-17T07:20:44Z"

- apiVersion: v1

fieldsType: FieldsV1

fieldsV1:

f:metadata:

f:annotations:

.: {}

f:volumes.kubernetes.io/controller-managed-attach-detach: {}

f:labels:

.: {}

f:beta.kubernetes.io/arch: {}

f:beta.kubernetes.io/os: {}

f:kubernetes.io/arch: {}

f:kubernetes.io/hostname: {}

f:kubernetes.io/os: {}

f:status:

f:addresses:

.: {}

k:{"type":"Hostname"}:

.: {}

f:address: {}

f:type: {}

k:{"type":"InternalIP"}:

.: {}

f:address: {}

f:type: {}

f:allocatable:

.: {}

f:cpu: {}

f:ephemeral-storage: {}

f:hugepages-1Gi: {}

f:hugepages-2Mi: {}

f:memory: {}

f:pods: {}

f:capacity:

.: {}

f:cpu: {}

f:ephemeral-storage: {}

f:hugepages-1Gi: {}

f:hugepages-2Mi: {}

f:memory: {}

f:pods: {}

f:conditions:

.: {}

k:{"type":"DiskPressure"}:

.: {}

f:lastHeartbeatTime: {}

f:lastTransitionTime: {}

f:message: {}

f:reason: {}

f:status: {}

f:type: {}

k:{"type":"MemoryPressure"}:

.: {}

f:lastHeartbeatTime: {}

f:lastTransitionTime: {}

f:message: {}

f:reason: {}

f:status: {}

f:type: {}

k:{"type":"PIDPressure"}:

.: {}

f:lastHeartbeatTime: {}

f:lastTransitionTime: {}

f:message: {}

f:reason: {}

f:status: {}

f:type: {}

k:{"type":"Ready"}:

.: {}

f:lastHeartbeatTime: {}

f:lastTransitionTime: {}

f:message: {}

f:reason: {}

f:status: {}

f:type: {}

f:daemonEndpoints:

f:kubeletEndpoint:

f:Port: {}

f:images: {}

f:nodeInfo:

f:architecture: {}

f:bootID: {}

f:containerRuntimeVersion: {}

f:kernelVersion: {}

f:kubeProxyVersion: {}

f:kubeletVersion: {}

f:machineID: {}

f:operatingSystem: {}

f:osImage: {}

f:systemUUID: {}

manager: kubelet

operation: Update

time: "2020-05-17T07:25:53Z"

name: node2

resourceVersion: "6046"

selfLink: /api/v1/nodes/node2

uid: 9f091720-a3ff-4a0c-bce7-fe19f8a8b4f1

spec:

podCIDR: 10.244.2.0/24

podCIDRs:

- 10.244.2.0/24

status:

addresses:

- address: 10.128.0.24

type: InternalIP

- address: node2

type: Hostname

allocatable:

cpu: "2"

ephemeral-storage: "18566339759"

hugepages-1Gi: "0"

hugepages-2Mi: "0"

memory: 1937712Ki

pods: "110"

capacity:

cpu: "2"

ephemeral-storage: 20145768Ki

hugepages-1Gi: "0"

hugepages-2Mi: "0"

memory: 2040112Ki

pods: "110"

conditions:

- lastHeartbeatTime: "2020-05-17T07:20:44Z"

lastTransitionTime: "2020-05-17T07:20:44Z"

message: Flannel is running on this node

reason: FlannelIsUp

status: "False"

type: NetworkUnavailable

- lastHeartbeatTime: "2020-05-17T07:25:53Z"

lastTransitionTime: "2020-05-17T07:04:19Z"

message: kubelet has sufficient memory available

reason: KubeletHasSufficientMemory

status: "False"

type: MemoryPressure

- lastHeartbeatTime: "2020-05-17T07:25:53Z"

lastTransitionTime: "2020-05-17T07:04:19Z"

message: kubelet has no disk pressure

reason: KubeletHasNoDiskPressure

status: "False"

type: DiskPressure

- lastHeartbeatTime: "2020-05-17T07:25:53Z"

lastTransitionTime: "2020-05-17T07:04:19Z"

message: kubelet has sufficient PID available

reason: KubeletHasSufficientPID

status: "False"

type: PIDPressure

- lastHeartbeatTime: "2020-05-17T07:25:53Z"

lastTransitionTime: "2020-05-17T07:05:59Z"

message: kubelet is posting ready status. AppArmor enabled

reason: KubeletReady

status: "True"

type: Ready

daemonEndpoints:

kubeletEndpoint:

Port: 10250

images:

- names:

- k8s.gcr.io/kube-proxy@sha256:8bfd5a33756f4eb95a8598cbef083ad7a59f40c09dccf4ba692b6b91e24cc9c1

- k8s.gcr.io/kube-proxy:v1.18.2

sizeBytes: 116538359

- names:

- cloudnativelabs/kube-router@sha256:b81714cbaa03f8e45859e3b40c006a6822bbc73d07e65d8ef6c17b5f553658c3

- cloudnativelabs/kube-router:latest

sizeBytes: 102428036

- names:

- quay.io/coreos/flannel@sha256:6d451d92c921f14bfb38196aacb6e506d4593c5b3c9d40a8b8a2506010dc3e10

- quay.io/coreos/flannel:v0.12.0-amd64

sizeBytes: 52767393

- names:

- k8s.gcr.io/coredns@sha256:2c8d61c46f484d881db43b34d13ca47a269336e576c81cf007ca740fa9ec0800

- k8s.gcr.io/coredns:1.6.7

sizeBytes: 43794147

- names:

- k8s.gcr.io/pause@sha256:927d98197ec1141a368550822d18fa1c60bdae27b78b0c004f705f548c07814f

- k8s.gcr.io/pause:3.2

sizeBytes: 682696

nodeInfo:

architecture: amd64

bootID: 494d5599-dd76-4ba1-b8fb-9d48d83d8db6

containerRuntimeVersion: docker://19.3.8

kernelVersion: 4.15.0-1061-gcp

kubeProxyVersion: v1.18.2

kubeletVersion: v1.18.2

machineID: f896982b06eef09e1dfab5c1102f8329

operatingSystem: linux

osImage: Ubuntu 16.04.6 LTS

systemUUID: F896982B-06EE-F09E-1DFA-B5C1102F8329

- apiVersion: v1

kind: Node

metadata:

annotations:

flannel.alpha.coreos.com/backend-data: '{"VtepMAC":"7e:46:38:23:ef:6f"}'

flannel.alpha.coreos.com/backend-type: vxlan

flannel.alpha.coreos.com/kube-subnet-manager: "true"

flannel.alpha.coreos.com/public-ip: 10.128.0.25

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: "2020-05-17T07:04:27Z"

labels:

beta.kubernetes.io/arch: amd64

beta.kubernetes.io/os: linux

kubernetes.io/arch: amd64

kubernetes.io/hostname: node3

kubernetes.io/os: linux

managedFields:

- apiVersion: v1

fieldsType: FieldsV1

fieldsV1:

f:metadata:

f:annotations:

f:node.alpha.kubernetes.io/ttl: {}

f:spec:

f:podCIDR: {}

f:podCIDRs:

.: {}

v:"10.244.3.0/24": {}

manager: kube-controller-manager

operation: Update

time: "2020-05-17T07:04:27Z"

- apiVersion: v1

fieldsType: FieldsV1

fieldsV1:

f:metadata:

f:annotations:

f:kubeadm.alpha.kubernetes.io/cri-socket: {}

manager: kubeadm

operation: Update

time: "2020-05-17T07:04:28Z"

- apiVersion: v1

fieldsType: FieldsV1

fieldsV1:

f:metadata:

f:annotations:

f:flannel.alpha.coreos.com/backend-data: {}

f:flannel.alpha.coreos.com/backend-type: {}

f:flannel.alpha.coreos.com/kube-subnet-manager: {}

f:flannel.alpha.coreos.com/public-ip: {}

f:status:

f:conditions:

k:{"type":"NetworkUnavailable"}:

.: {}

f:lastHeartbeatTime: {}

f:lastTransitionTime: {}

f:message: {}

f:reason: {}

f:status: {}

f:type: {}

manager: flanneld

operation: Update

time: "2020-05-17T07:20:45Z"

- apiVersion: v1

fieldsType: FieldsV1

fieldsV1:

f:metadata:

f:annotations:

.: {}

f:volumes.kubernetes.io/controller-managed-attach-detach: {}

f:labels:

.: {}

f:beta.kubernetes.io/arch: {}

f:beta.kubernetes.io/os: {}

f:kubernetes.io/arch: {}

f:kubernetes.io/hostname: {}

f:kubernetes.io/os: {}

f:status:

f:addresses:

.: {}

k:{"type":"Hostname"}:

.: {}

f:address: {}

f:type: {}

k:{"type":"InternalIP"}:

.: {}

f:address: {}

f:type: {}

f:allocatable:

.: {}

f:cpu: {}

f:ephemeral-storage: {}

f:hugepages-1Gi: {}

f:hugepages-2Mi: {}

f:memory: {}

f:pods: {}

f:capacity:

.: {}

f:cpu: {}

f:ephemeral-storage: {}

f:hugepages-1Gi: {}

f:hugepages-2Mi: {}

f:memory: {}

f:pods: {}

f:conditions:

.: {}

k:{"type":"DiskPressure"}:

.: {}

f:lastHeartbeatTime: {}

f:lastTransitionTime: {}

f:message: {}

f:reason: {}

f:status: {}

f:type: {}

k:{"type":"MemoryPressure"}:

.: {}

f:lastHeartbeatTime: {}

f:lastTransitionTime: {}

f:message: {}

f:reason: {}

f:status: {}

f:type: {}

k:{"type":"PIDPressure"}:

.: {}

f:lastHeartbeatTime: {}

f:lastTransitionTime: {}

f:message: {}

f:reason: {}

f:status: {}

f:type: {}

k:{"type":"Ready"}:

.: {}

f:lastHeartbeatTime: {}

f:lastTransitionTime: {}

f:message: {}

f:reason: {}

f:status: {}

f:type: {}

f:daemonEndpoints:

f:kubeletEndpoint:

f:Port: {}

f:images: {}

f:nodeInfo:

f:architecture: {}

f:bootID: {}

f:containerRuntimeVersion: {}

f:kernelVersion: {}

f:kubeProxyVersion: {}

f:kubeletVersion: {}

f:machineID: {}

f:operatingSystem: {}

f:osImage: {}

f:systemUUID: {}

manager: kubelet

operation: Update

time: "2020-05-17T07:26:02Z"

name: node3

resourceVersion: "6069"

selfLink: /api/v1/nodes/node3

uid: 96be2707-0de8-4dbd-912b-ebba1b278bec

spec:

podCIDR: 10.244.3.0/24

podCIDRs:

- 10.244.3.0/24

status:

addresses:

- address: 10.128.0.25

type: InternalIP

- address: node3

type: Hostname

allocatable:

cpu: "2"

ephemeral-storage: "18566339759"

hugepages-1Gi: "0"

hugepages-2Mi: "0"

memory: 1937712Ki

pods: "110"

capacity:

cpu: "2"

ephemeral-storage: 20145768Ki

hugepages-1Gi: "0"

hugepages-2Mi: "0"

memory: 2040112Ki

pods: "110"

conditions:

- lastHeartbeatTime: "2020-05-17T07:20:45Z"

lastTransitionTime: "2020-05-17T07:20:45Z"

message: Flannel is running on this node

reason: FlannelIsUp

status: "False"

type: NetworkUnavailable

- lastHeartbeatTime: "2020-05-17T07:26:02Z"

lastTransitionTime: "2020-05-17T07:04:27Z"

message: kubelet has sufficient memory available

reason: KubeletHasSufficientMemory

status: "False"

type: MemoryPressure

- lastHeartbeatTime: "2020-05-17T07:26:02Z"

lastTransitionTime: "2020-05-17T07:04:27Z"

message: kubelet has no disk pressure

reason: KubeletHasNoDiskPressure

status: "False"

type: DiskPressure

- lastHeartbeatTime: "2020-05-17T07:26:02Z"

lastTransitionTime: "2020-05-17T07:04:27Z"

message: kubelet has sufficient PID available

reason: KubeletHasSufficientPID

status: "False"

type: PIDPressure

- lastHeartbeatTime: "2020-05-17T07:26:02Z"

lastTransitionTime: "2020-05-17T07:05:58Z"

message: kubelet is posting ready status. AppArmor enabled

reason: KubeletReady

status: "True"

type: Ready

daemonEndpoints:

kubeletEndpoint:

Port: 10250

images:

- names:

- k8s.gcr.io/kube-proxy@sha256:8bfd5a33756f4eb95a8598cbef083ad7a59f40c09dccf4ba692b6b91e24cc9c1

- k8s.gcr.io/kube-proxy:v1.18.2

sizeBytes: 116538359

- names:

- cloudnativelabs/kube-router@sha256:b81714cbaa03f8e45859e3b40c006a6822bbc73d07e65d8ef6c17b5f553658c3

- cloudnativelabs/kube-router:latest

sizeBytes: 102428036

- names:

- quay.io/coreos/flannel@sha256:6d451d92c921f14bfb38196aacb6e506d4593c5b3c9d40a8b8a2506010dc3e10

- quay.io/coreos/flannel:v0.12.0-amd64

sizeBytes: 52767393

- names:

- k8s.gcr.io/coredns@sha256:2c8d61c46f484d881db43b34d13ca47a269336e576c81cf007ca740fa9ec0800

- k8s.gcr.io/coredns:1.6.7

sizeBytes: 43794147

- names:

- k8s.gcr.io/pause@sha256:927d98197ec1141a368550822d18fa1c60bdae27b78b0c004f705f548c07814f

- k8s.gcr.io/pause:3.2

sizeBytes: 682696

nodeInfo:

architecture: amd64

bootID: a364898b-fc2d-48b0-86f1-ce43d584ab84

containerRuntimeVersion: docker://19.3.8

kernelVersion: 4.15.0-1061-gcp

kubeProxyVersion: v1.18.2

kubeletVersion: v1.18.2

machineID: b779989ed57bcf7735f8b25a6d956d6a

operatingSystem: linux

osImage: Ubuntu 16.04.6 LTS

systemUUID: B779989E-D57B-CF77-35F8-B25A6D956D6A

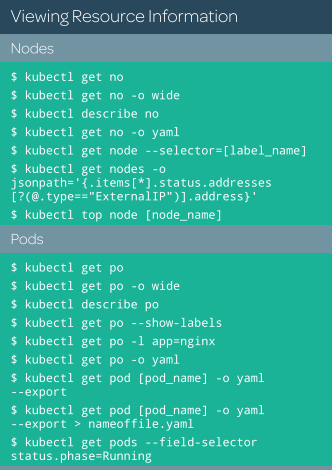
kind: List

metadata:

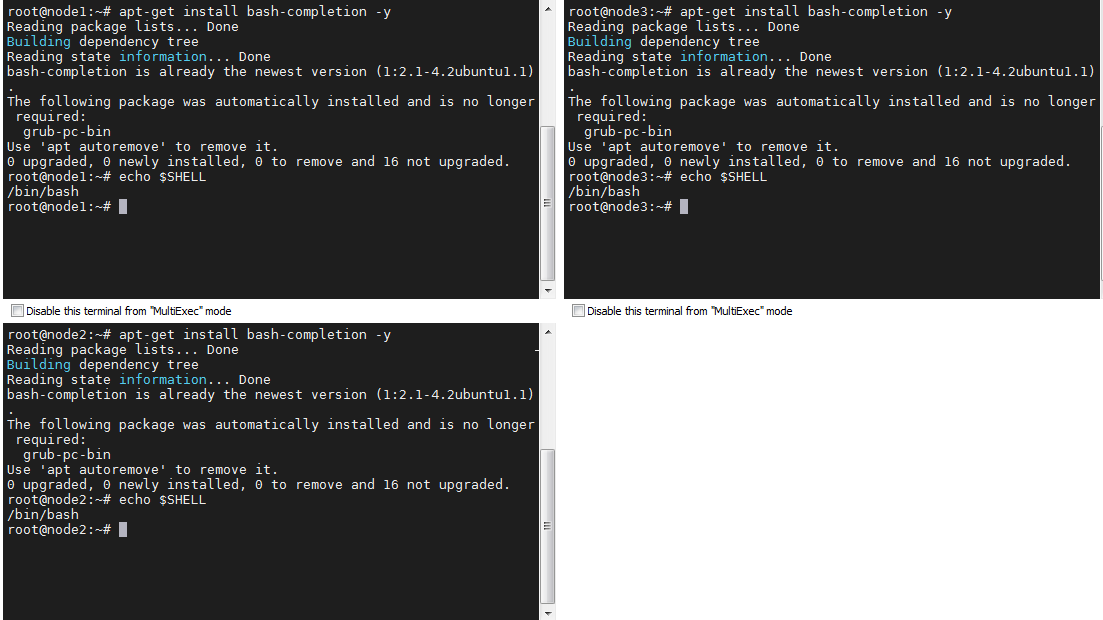
resourceVersion: ""

selfLink: ""

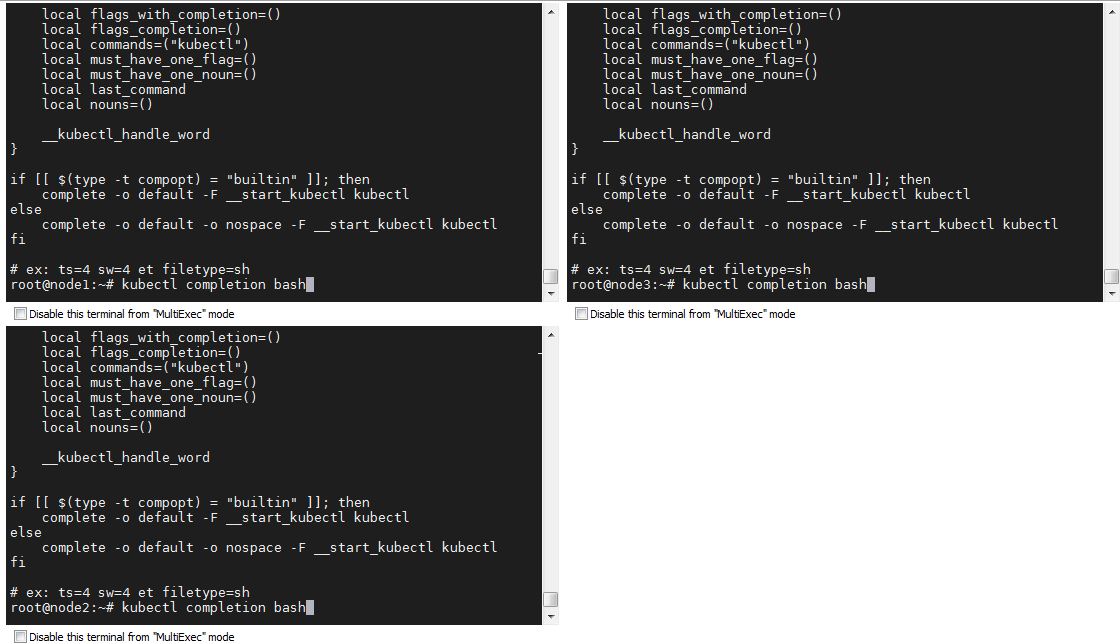
root@node1:~#

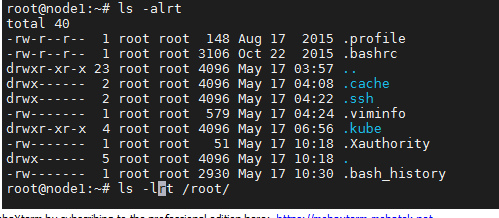


apt-get install bash-completion



kubectl completion bash





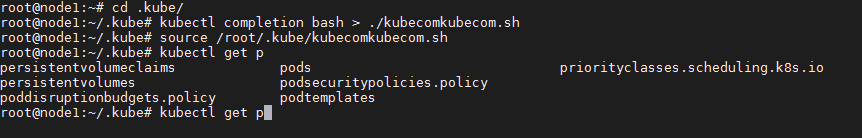
(.kube will create on master node)

root@node1:~# cd .kube/

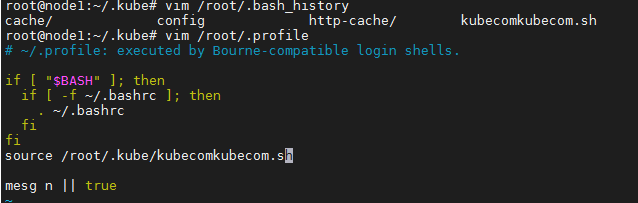
root@node1:~/.kube# kubectl completion bash > ./kubecomkubecom.sh

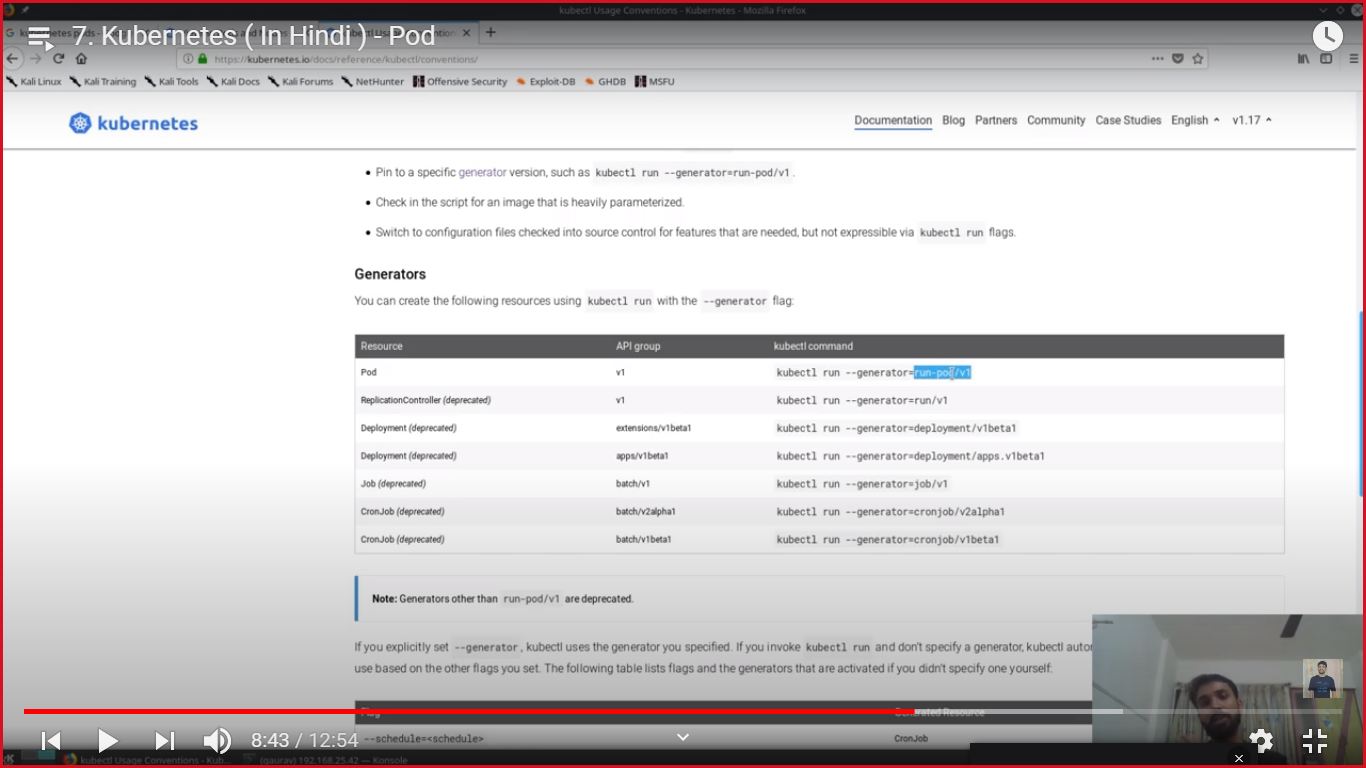
[root@node1:~/.kube#](mailto:root@node1:~/.kube)

source /root/.kube/kubecomkubecom.sh

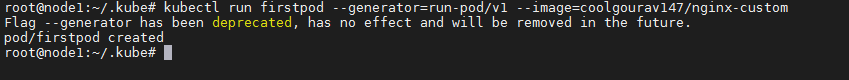


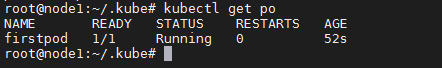
To make as permanent from reboot, we need to update in profile





kubectl run firstpod --generator=run-pod/v1 --image=coolgourav147/nginx-custom



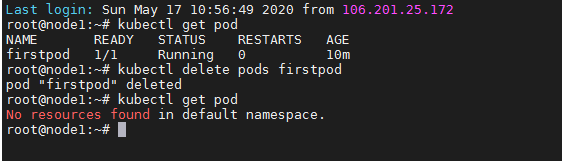


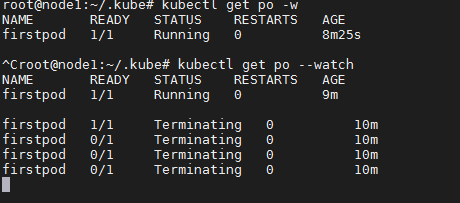


kubectl get po –w

kubectl get po –watch

To delete pods

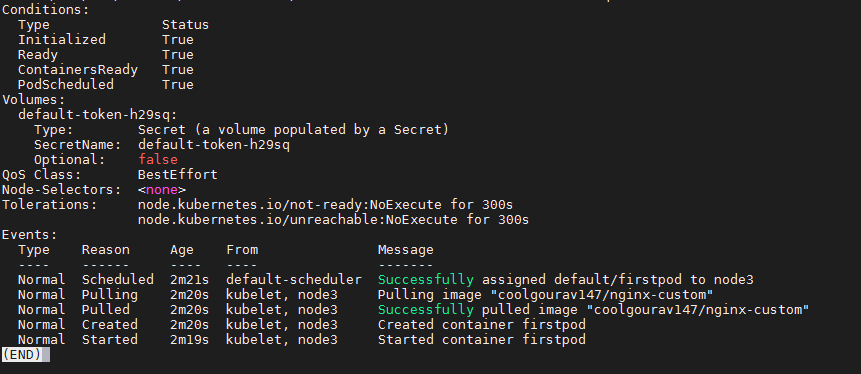




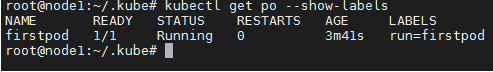
To describe the pod

kubectl describe po firstpod | less

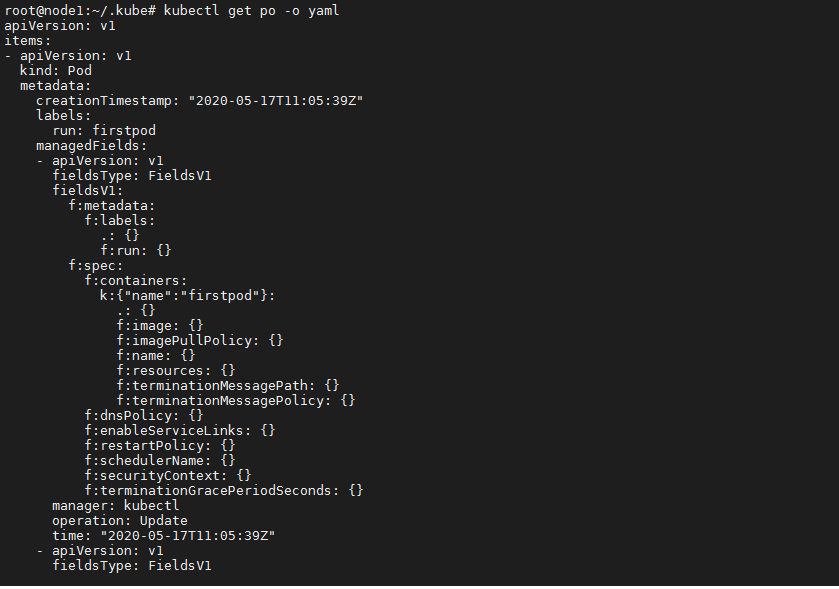


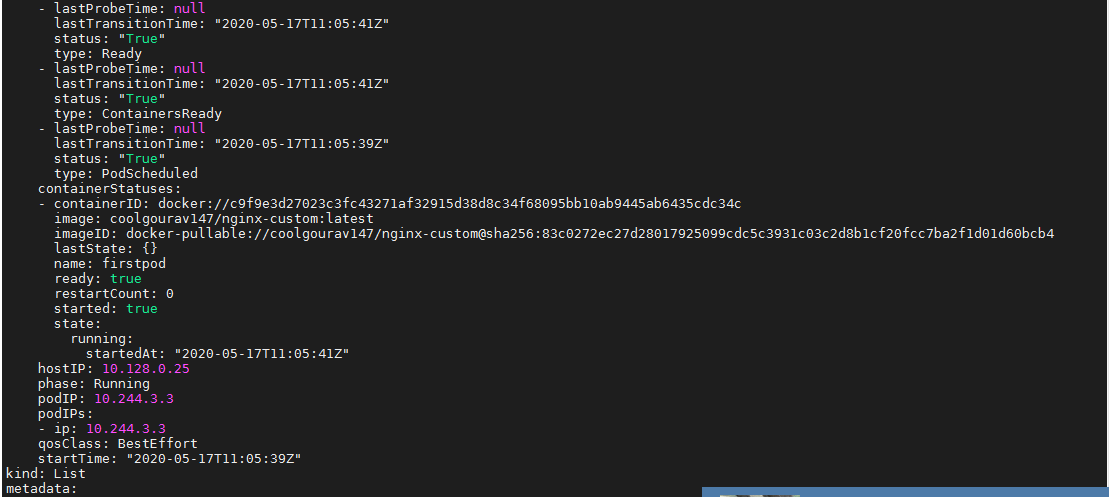


kubectl get po --show-labels



kubectl get po -o yaml



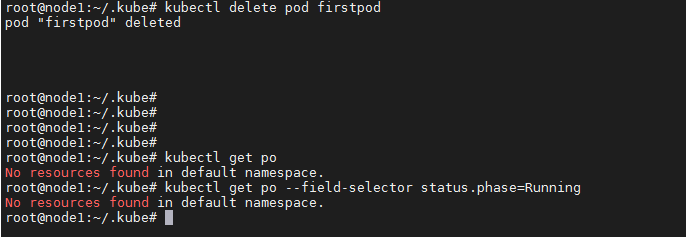


kubectl get po firstpod -o yaml --export



kubectl get po --field-selector status.phase=Running





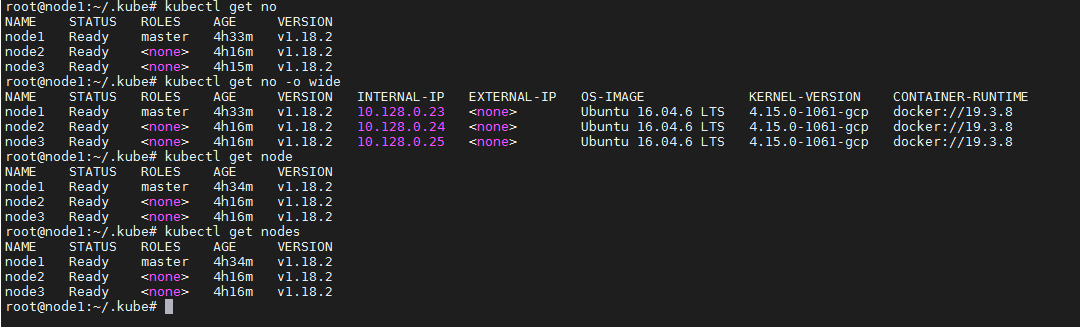
Nodes commands

122 kubectl get no

123 kubectl get no -o wide

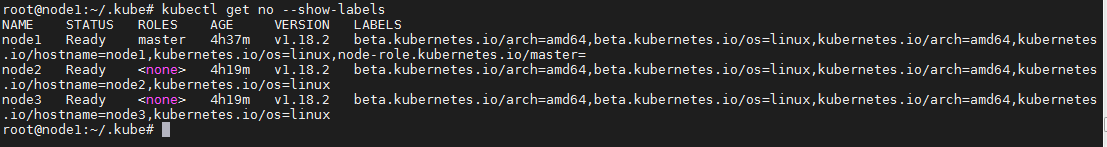
124 kubectl get node

125 kubectl get nodes

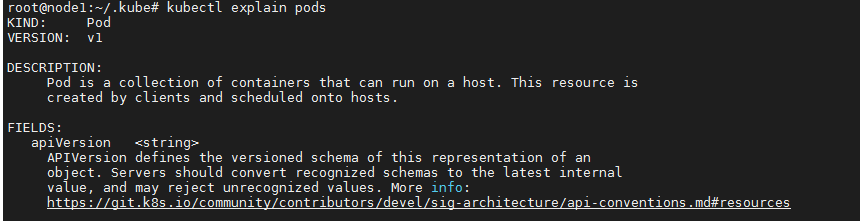


kubectl get no -o yaml

kubectl get no --show-labels



kubectl explain pods



kubectl explain replicationcontroller

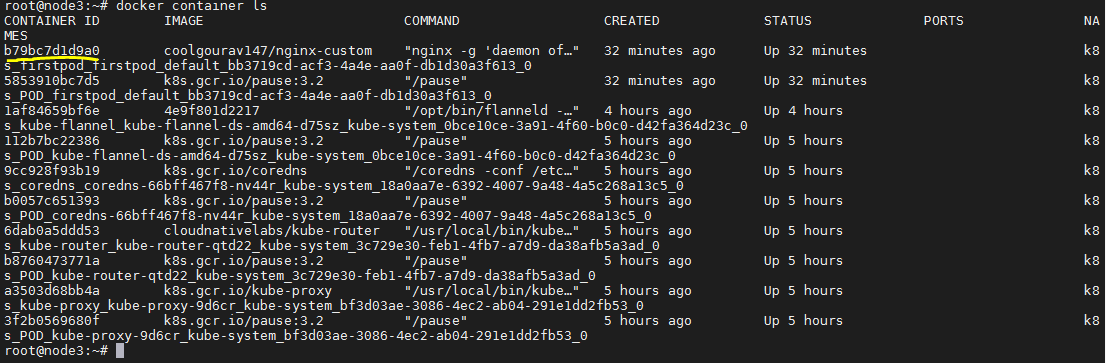
kubectl describe pod firstpod



Going to delete that container from node3 and so pod wil create container automatic

To check that container, we have to check in node3

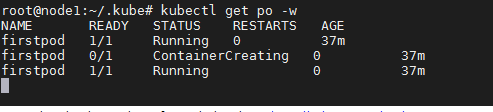
docker container ls



kubectl get po –w

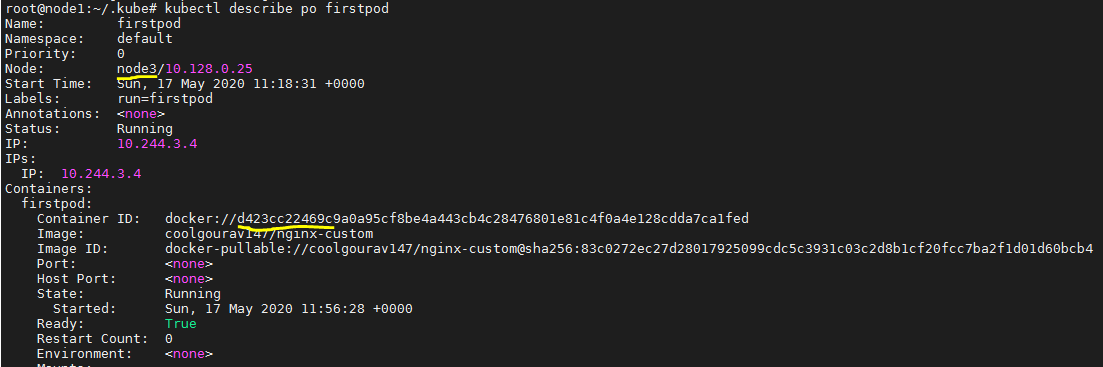
root@node3:~# docker container rm -f b79bc7d1d9a0

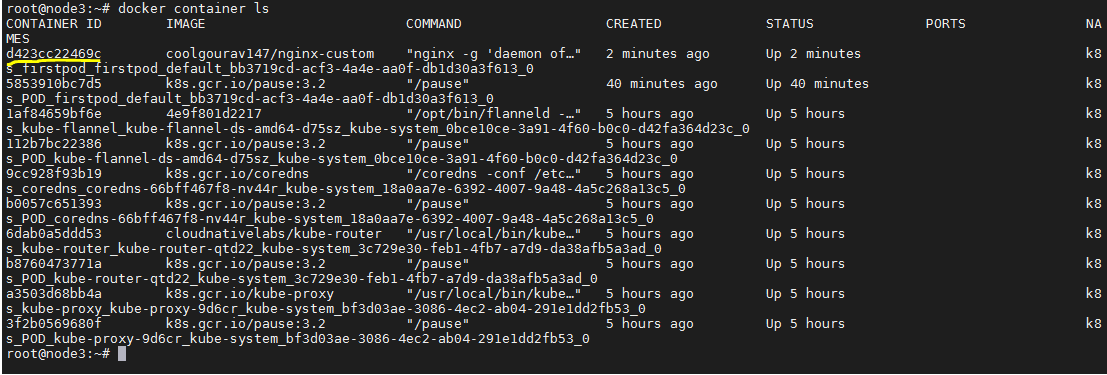
b79bc7d1d9a0



(you can see here, after deleting container , pod create another container automatic)

New container create in pod below

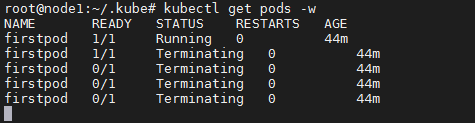


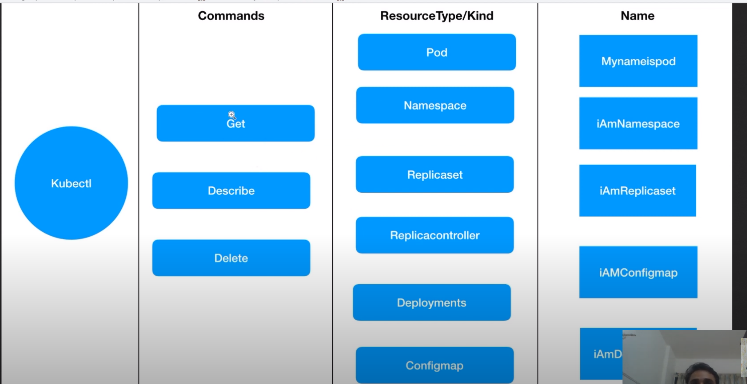


To delete pod



To watch pod real time status

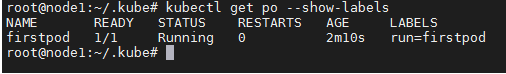


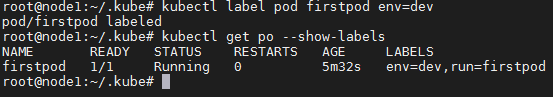




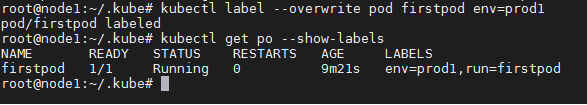
Labels

kubectl get po --show-labels



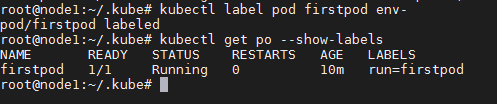


Update labels



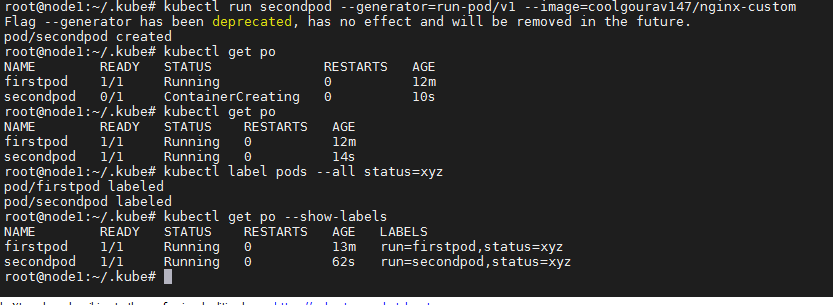
To delete label

kubectl label pod firstpod env-



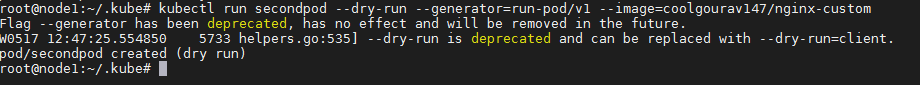
To add labels with multiple pods

# kubectl label pods --all status=xyz



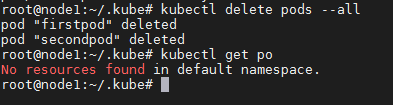
Note: To check in client side only

kubectl run secondpod --dry-run --generator=run-pod/v1 --image=coolgourav147/nginx-custom



To delete pods

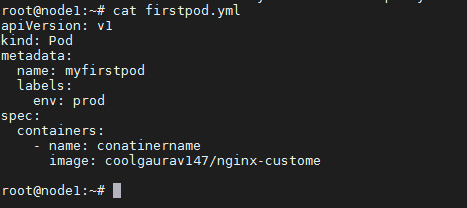
kubectl delete pods –all



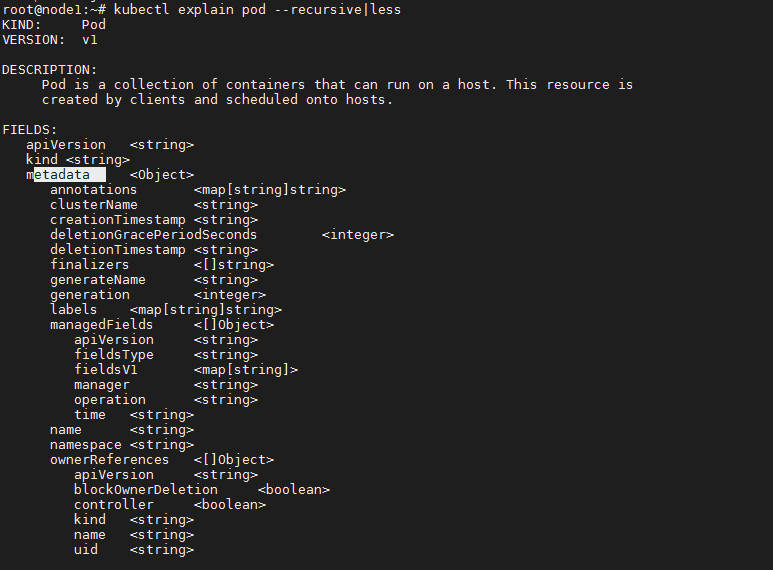
To create yam file

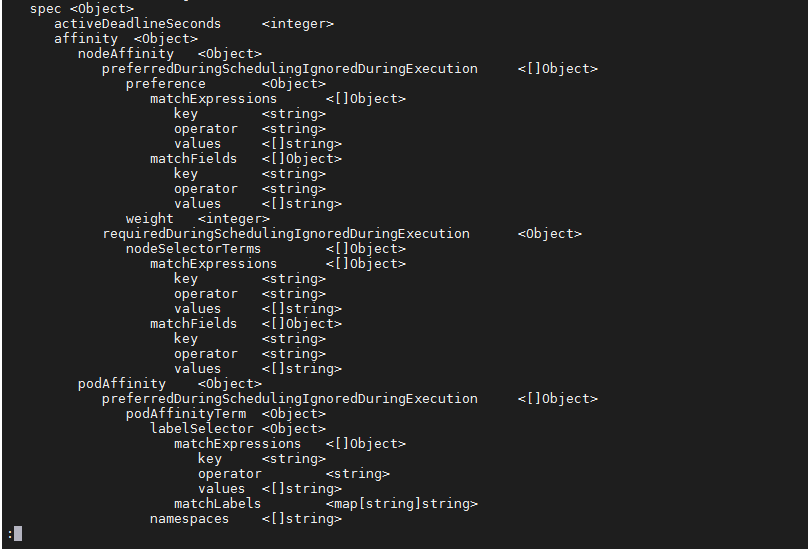
kubectl explain pods



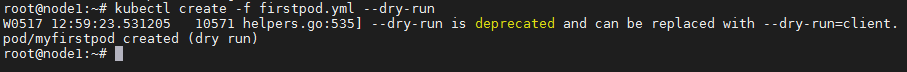


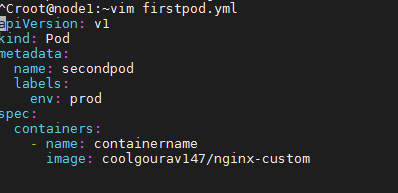
kubectl explain pod --recursive|less





kubectl create -f firstpod.yml --dry-run





^Croot@node1:~vim firstpod.yml

apiVersion: v1

kind: Pod

metadata:

name: secondpod

labels:

env: prod

spec:

containers:

- name: containername

image: coolgourav147/nginx-custom

root@node1:~# cat firstpod.yml

apiVersion: v1

kind: Pod

metadata:

name: myfirstpod

labels:

env: prod

spec:

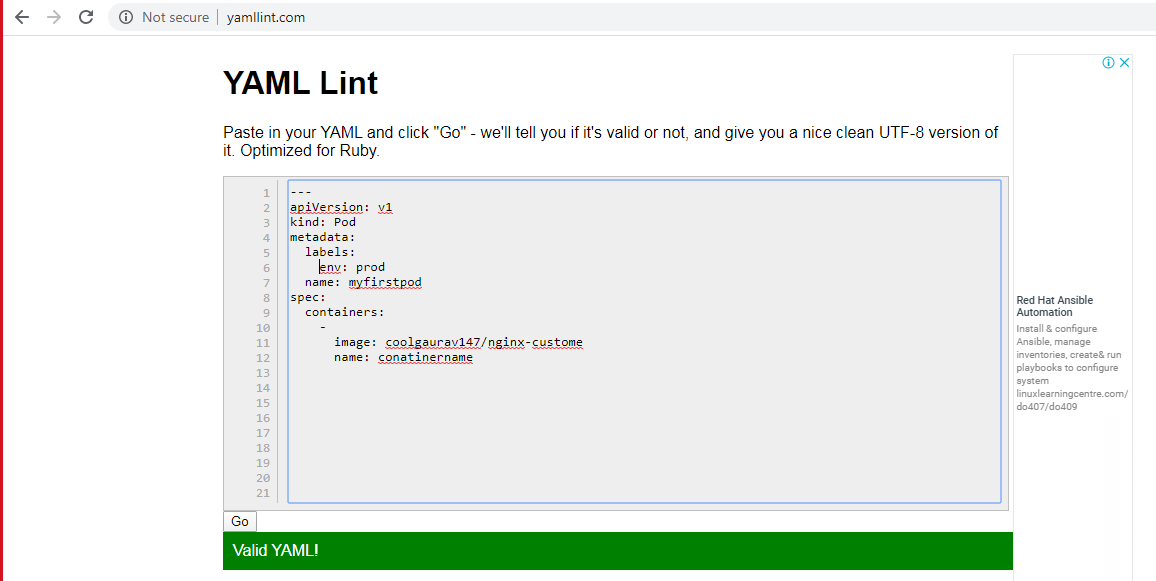
containers:

- name: conatinername

image: coolgaurav147/nginx-custome

To verify yaml file

<http://www.yamllint.com/>



To take help to create yaml file

kubectl run secondpod --dry-run --generator=run-pod/v1 --image=coolgourav147/nginx-custom -o yaml > mysecondpod.yml

root@node1:~# **kubectl run secondpod --dry-run --generator=run-pod/v1 --image=coolgourav147/nginx-custom -o yaml > mysecondpod.yml**

Flag --generator has been deprecated, has no effect and will be removed in the future.

W0517 13:09:03.485706 14541 helpers.go:535] --dry-run is deprecated and can be replaced with --dry-run=client.

root@node1:~# cat mysecondpod.yml

apiVersion: v1

kind: Pod

metadata:

creationTimestamp: null

labels:

run: secondpod

name: secondpod

spec:

containers:

- image: coolgourav147/nginx-custom

name: secondpod

resources: {}

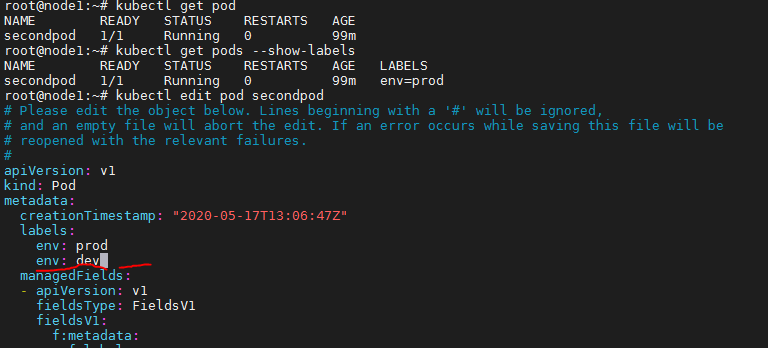
dnsPolicy: ClusterFirst

restartPolicy: Always

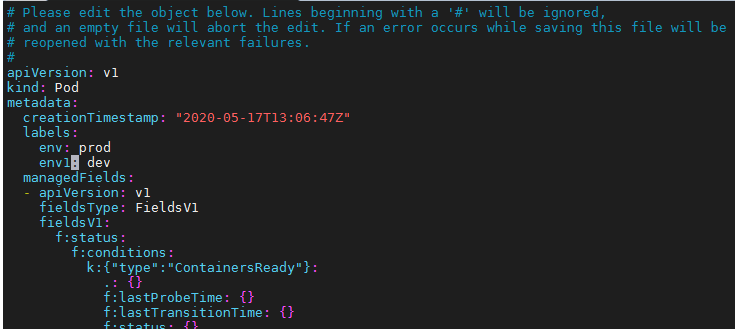
status: {}

root@node1:~#

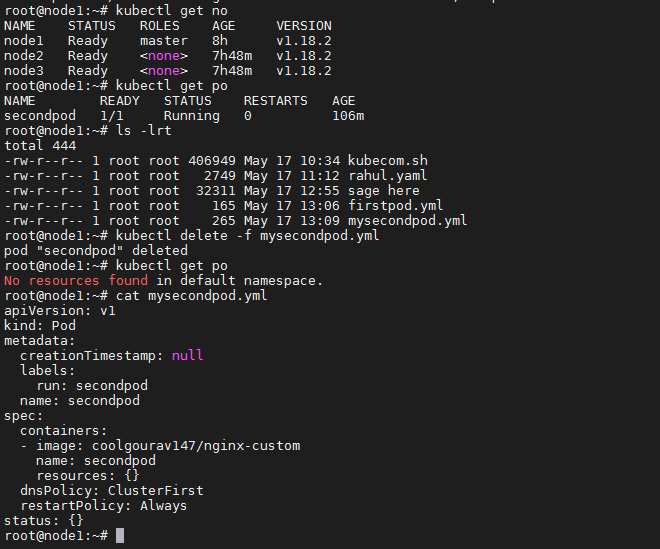
kubernetes imperative vs declarative or difference between create and apply



(here, we add another label with same pod)







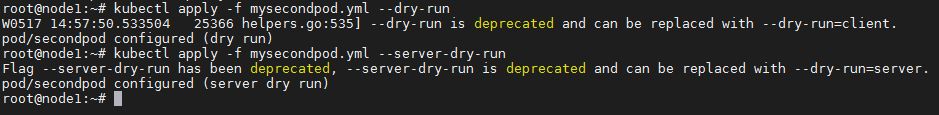


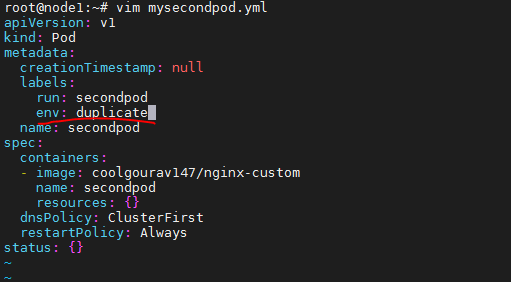


If you run ym file to create same pod, you will get below warning



To run –dry-run for client side and –dry-server-run for server side to make sure pod is exit or not







kubectl edit pod mysecondpod

kubectl get pod --show-labels

kubectl delete -f mysecondpod.yml

kubectl apply -f mysecondpod.yml

kubectl apply -f mysecondpod.yml --dry-run

kubectl apply -f mysecondpod.yml --server-dry-run

kubectl diff -f mysecondpod.yml

environment variable

^Croot@node1:~vim firstpod.yml

apiVersion: v1

kind: Pod

metadata:

name: firstpod

labels:

env: prod

spec:

containers:

- name: containername

image: coolgourav147/nginx-custom

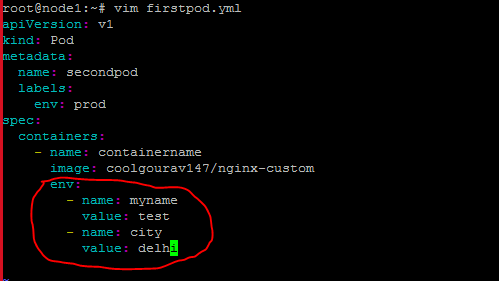
env:

- name: myname

value: test

- name: city

value: delhi

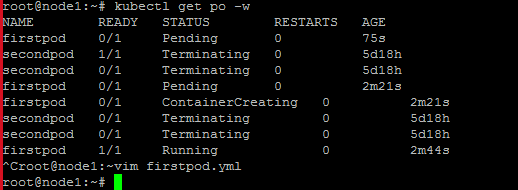
s

Now only master node running,so while applying yml file,po showing in pending status

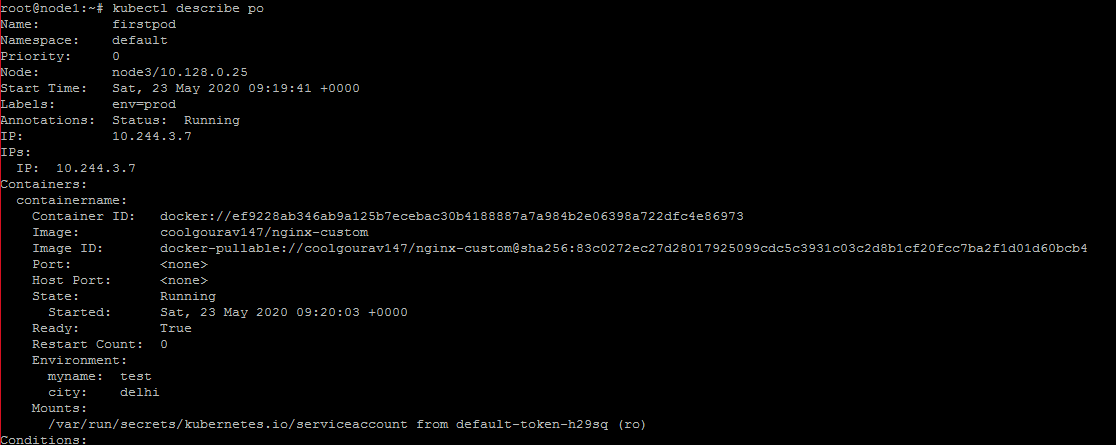
kubectl apply -f firstpod.yml



Started all the worker nodes to check, you can see after some time pod will create automatic

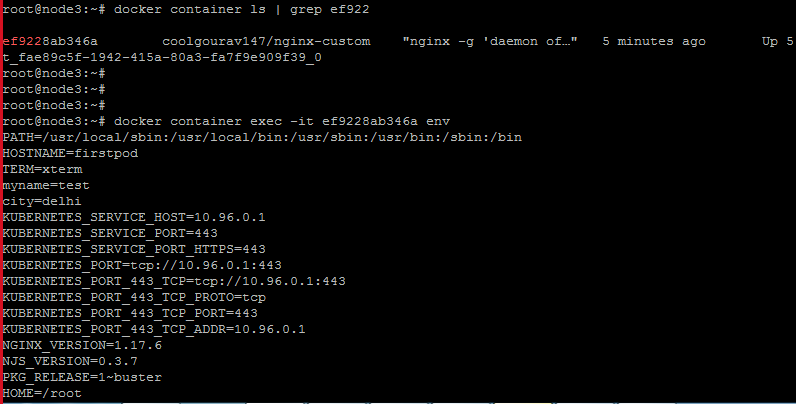


To check po deails



docker container ls

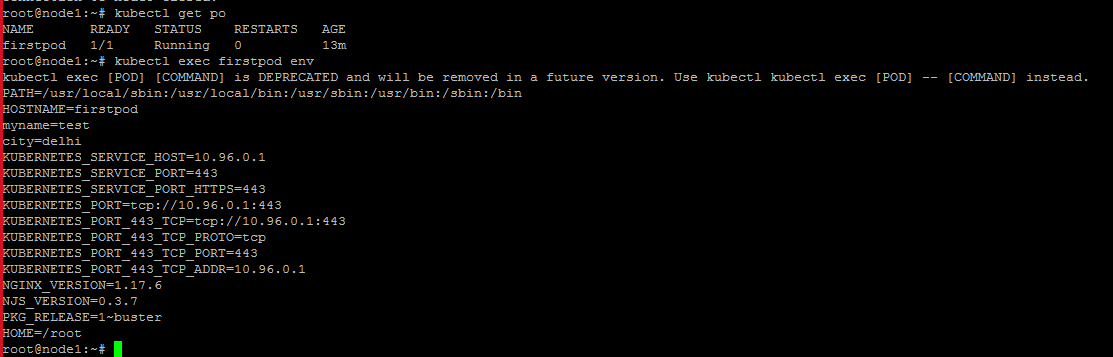
docker container exec -it ef9228ab346a env



 Run a Command in Pod's Containers

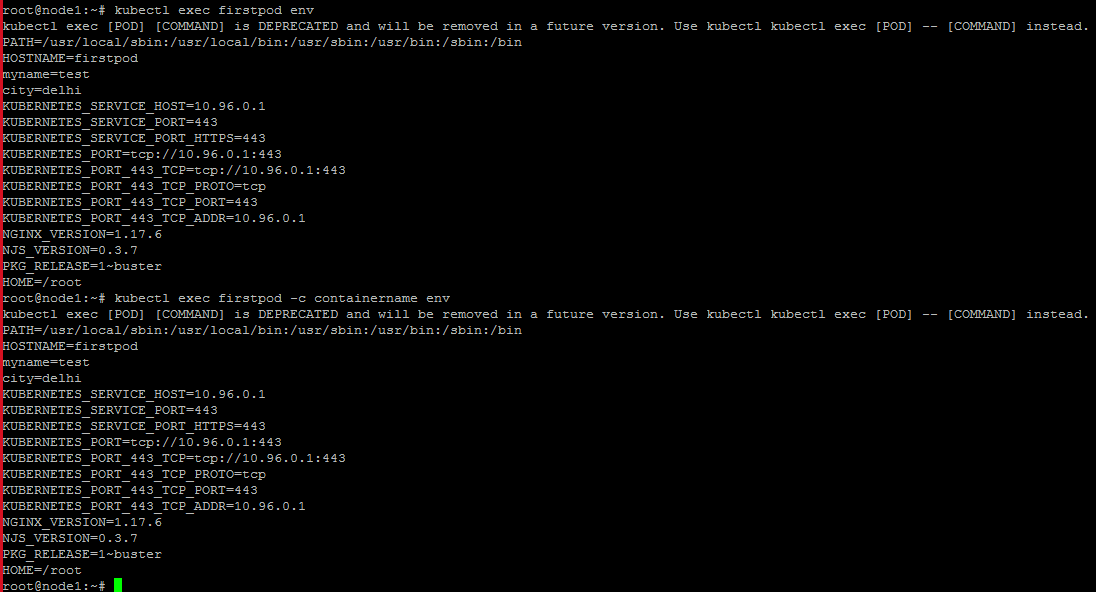
kubectl get po

kubectl exec firstpod env

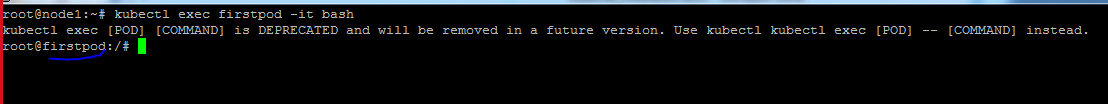


kubectl exec firstpod env

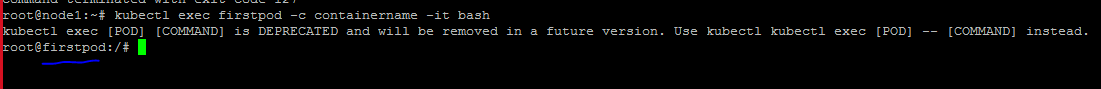
kubectl exec firstpod -c containername env



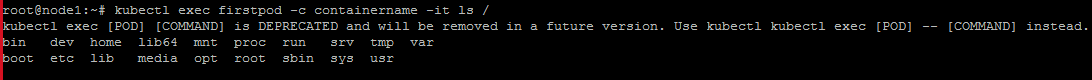
kubectl exec firstpod -it bash (to enter inside the container)

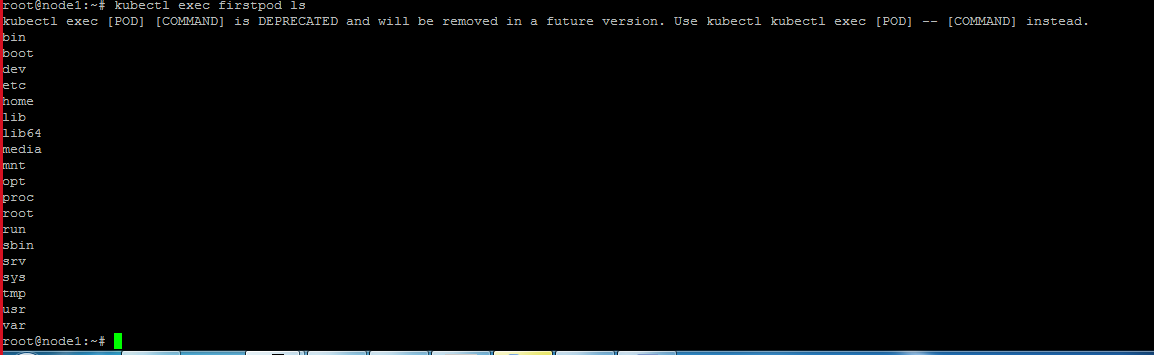


kubectl exec firstpod -c containername -it bash (it is being used if there is multiple containers)



To check ls inside container





To remove all the pods

kubectl delete po --all



15 video skipped🡪 to sleep

root@node3:~# kubectl get po -w

error: no configuration has been provided, try setting KUBERNETES\_MASTER environment variable

root@node3:~#

run below

export KUBECONFIG=/etc/kubernetes/kubelet.conf

then run commands to check

<https://www.mirantis.com/blog/multi-container-pods-and-container-communication-in-kubernetes/>

To create double container

apiVersion: v1

kind: Pod

metadata:

name: mc1

spec:

volumes:

- name: html

emptyDir: {}

containers:

- name: 1st

image: nginx

volumeMounts:

- name: html

mountPath: /usr/share/nginx/html

- name: 2nd

image: debian

volumeMounts:

- name: html

mountPath: /html

command: ["/bin/sh", "-c"]

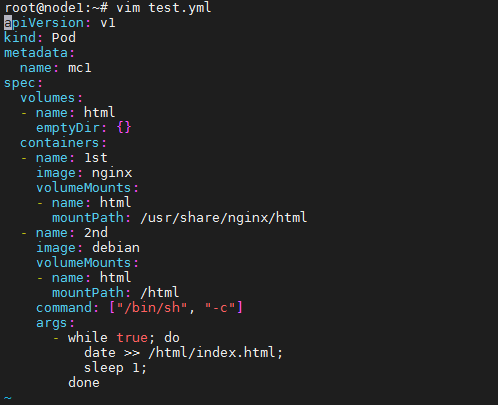
args:

- while true; do

date >> /html/index.html;

sleep 1;

done



To enter first container

kubectl exec mc1 -c 1st -it bash

apt-get

apt-install net-tools\*

ifconfig

To enter second container

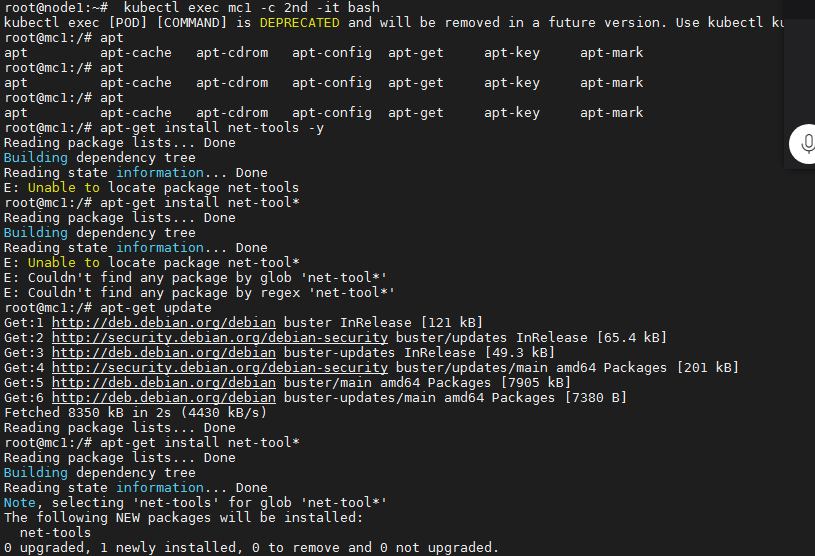
kubectl exec mc1 -c 2nd -it bash

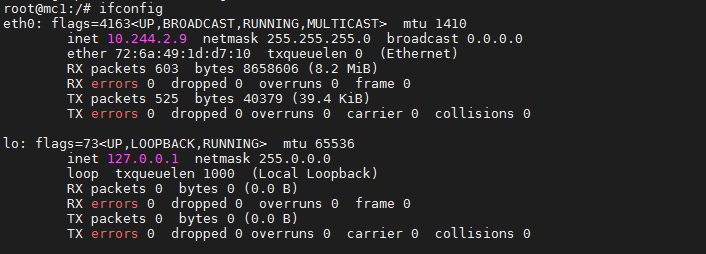
apt-get

apt-install net-tools\*

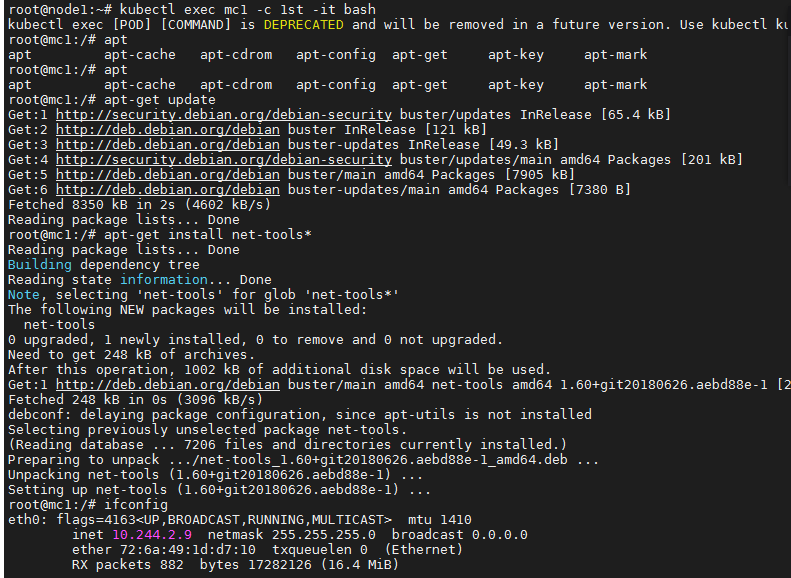
ifconfig

(We found, in both ip is same)





In 1st container



(you can in both container ip is same)

### Inter-container network communication

Containers in a Pod are accessible via “localhost”; they use the same network namespace. Also, for containers, the observable host name is a Pod’s name. Because containers share the same IP address and port space, you should use different ports in containers for incoming connections. In other words, applications in a Pod must coordinate their usage of ports.

apiVersion: v1

kind: ConfigMap

metadata:

name: mc3-nginx-conf

data:

nginx.conf: |-

user nginx;

worker\_processes 1;

error\_log /var/log/nginx/error.log warn;

pid /var/run/nginx.pid;

events {

worker\_connections 1024;

}

http {

include /etc/nginx/mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

upstream webapp {

server 127.0.0.1:5000;

}

server {

listen 80;

location / {

proxy\_pass http://webapp;

proxy\_redirect off;

}

}

}

apiVersion: v1

kind: ConfigMap

metadata:

name: mc3-nginx-conf

data:

nginx.conf: |-

user nginx;

worker\_processes 1;

error\_log /var/log/nginx/error.log warn;

pid /var/run/nginx.pid;

events {

worker\_connections 1024;

}

http {

include /etc/nginx/mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

upstream webapp {

server 127.0.0.1:5000;

}

server {

listen 80;

location / {

proxy\_pass http://webapp;

proxy\_redirect off;

}

}

}

How to create multiple container in a Pod

kubectl exec mc1 -c 1st -it bash

netstat –tnlp

netcat -l -p 8000



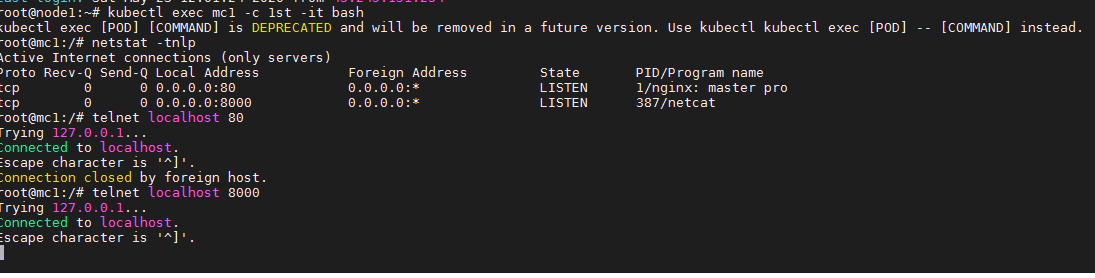
Open another termainl to check telnet

kubectl exec mc1 -c 1st -it bash

netstat –tnlp

telnet localhost 80

telnet localhost 8000



vim fourth.yml

root@node1:~# vim fourth.yml

apiVersion: v1

kind: Pod

metadata:

name: fourthpod

labels:

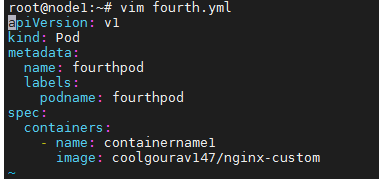
podname: fourthpod

spec:

containers:

- name: containername1

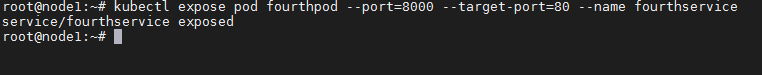
image: coolgourav147/nginx-custom



kubectl apply -f fourth.yml

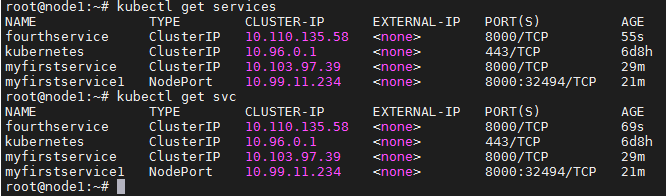


kubectl expose pod fourthpod --port=8000 --target-port=80 --name fourthservice

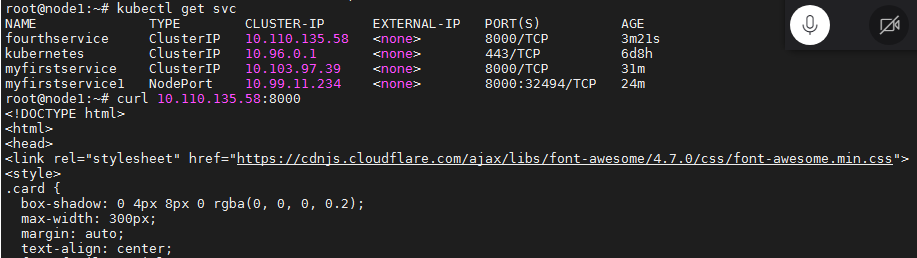


kubectl get services

kubectl get svc



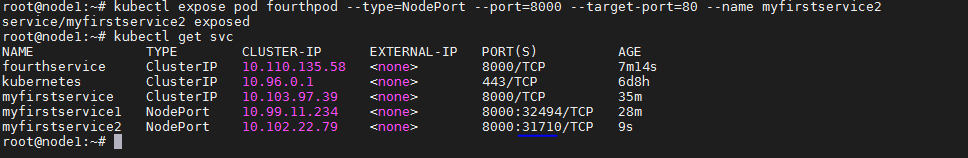
curl 10.110.135.58:8000



**Create node port**

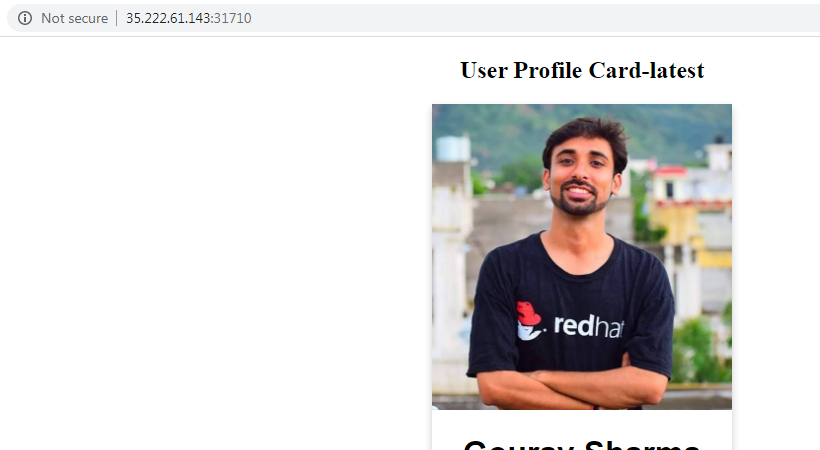
kubectl expose pod fourthpod --type=NodePort --port=8000 --target-port=80 --name myfirstservice2

kubectl get svc



Public IP of Host:port

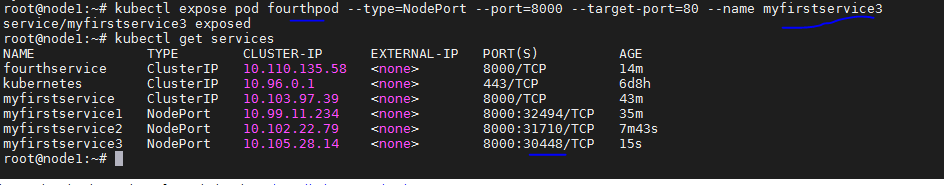
<http://35.222.61.143:31710/>



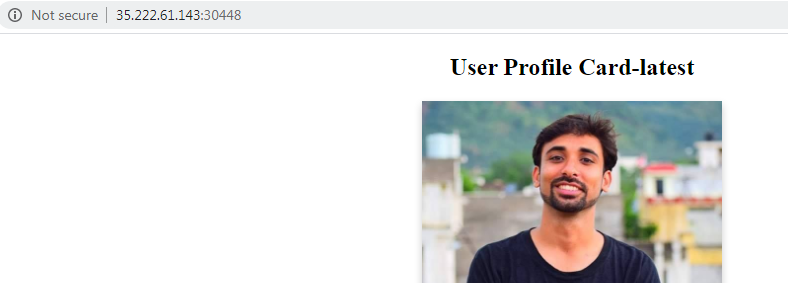
Go to each host and create nodeport to check

Tried on second worker node

kubectl expose pod fourthpod --type=NodePort --port=8000 --target-port=80 --name myfirstservice2



Host2 publicip:port



You can do on host3 as well

**How to redirect the request of multiple pods with same port**

root@node1:~# vim fifthservice.yml

apiVersion: v1

kind: Service

metadata:

name: myfirstservice

labels:

servicelabel: label1

spec:

type: NodePort

ports:

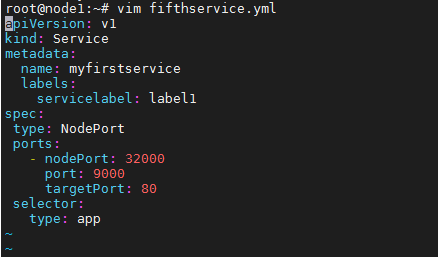
- nodePort: 32000

port: 9000

targetPort: 80

selector:

type: app



kubectl apply -f fifthservice.yml



root@node1:~# vim mysecondpod.yml

apiVersion: v1

kind: Pod

metadata:

labels:

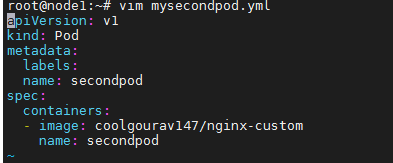
name: secondpod

spec:

containers:

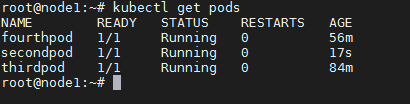
- image: coolgourav147/nginx-custom

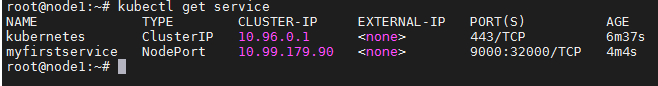
name: secondpod



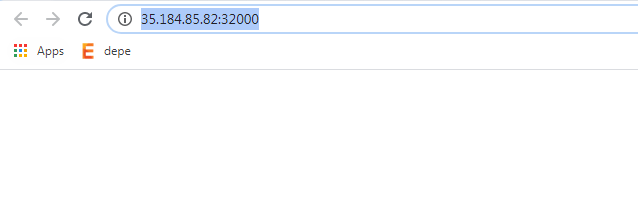
kubectl apply -f mysecondpod.yml

kubectl get pods

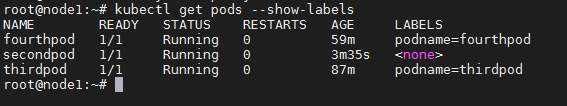




<http://35.184.85.82:32000/>



kubectl get pods --show-labels



kubectl label pod secondpod type=app



Create another yml file to create another pod

root@node1:~# vim mysecondpod.yml

apiVersion: v1

kind: Pod

metadata:

labels:

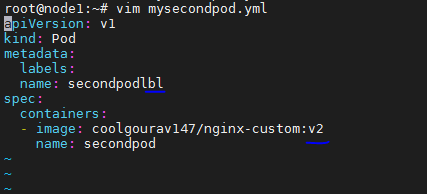
name: secondpodlbl

spec:

containers:

- image: coolgourav147/nginx-custom:v2

name: secondpod



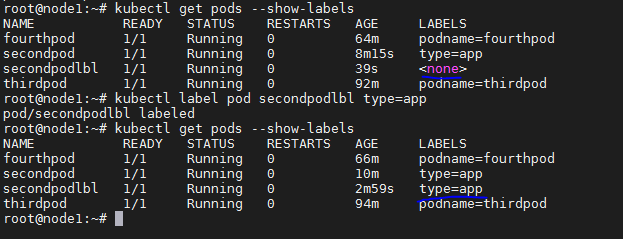
kubectl apply -f mysecondpod.yml

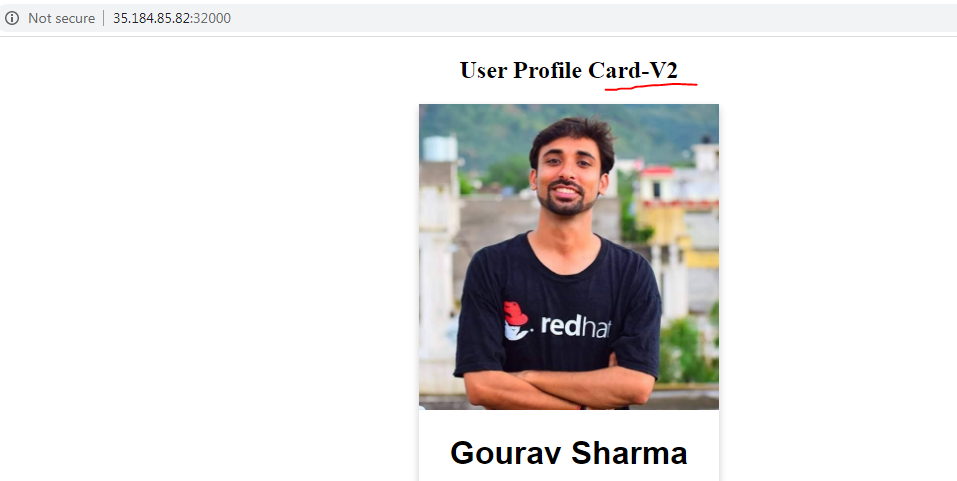


kubectl get pods --show-labels

kubectl label pod secondpodlbl type=app

kubectl get pods --show-labels





root@node1:~# vim redhat.yml

apiVersion: v1

kind: Pod

metadata:

labels:

name: redhat

spec:

containers:

- image: nginx

name: redhat

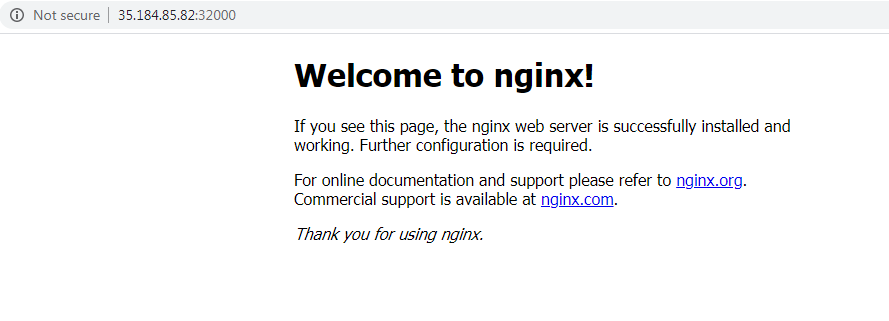
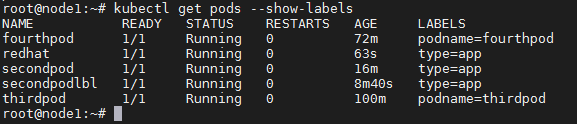
kubectl apply -f redhat.yml



kubectl get pods --show-labels

kubectl label pod redhat type=app

kubectl get pods --show-labels



Replication controller

root@node1:~# vim firstsc.yml

apiVersion: v1

kind: ReplicationController

metadata:

name: firstcontroller

labels:

app: voatingapp

spec:

replicas: 5

template:

metadata:

name: cdhpod

labels:

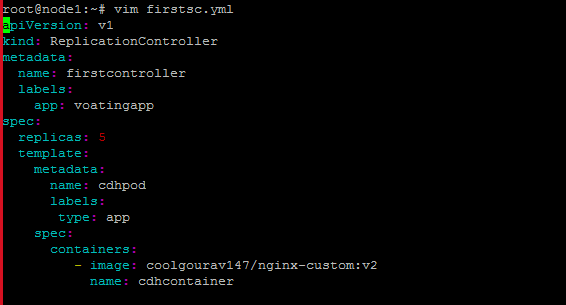
type: app

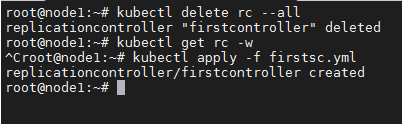
spec:

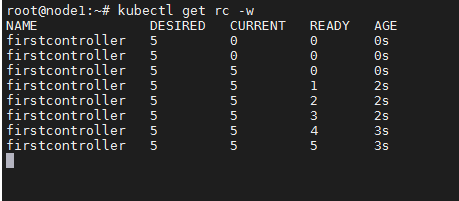
containers:

- image: coolgourav147/nginx-custom:v2

name: cdhcontainer

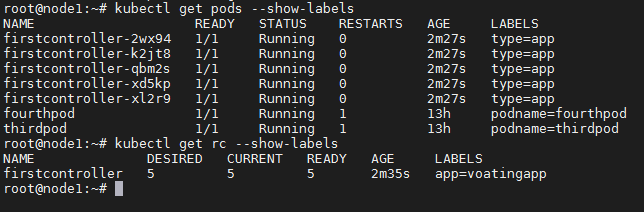






kubectl get pods --show-labels

kubectl get rc --show-labels



Lets delete one pod

kubectl delete pod firstcontroller-2wx94

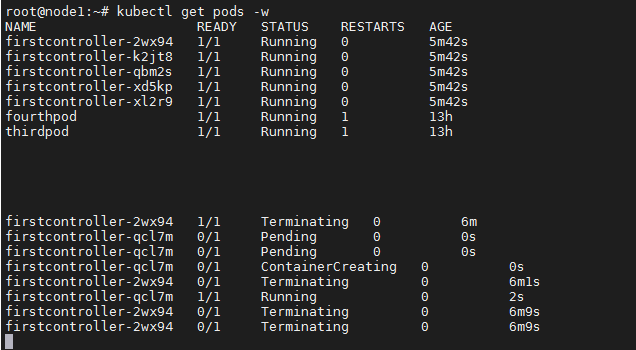
Open another terminal t check

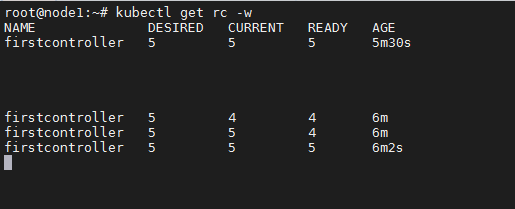
kubectl get pods –w

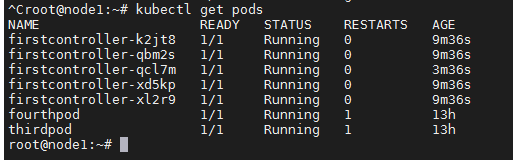
Open another second terminal to check

kubectl get rc –w

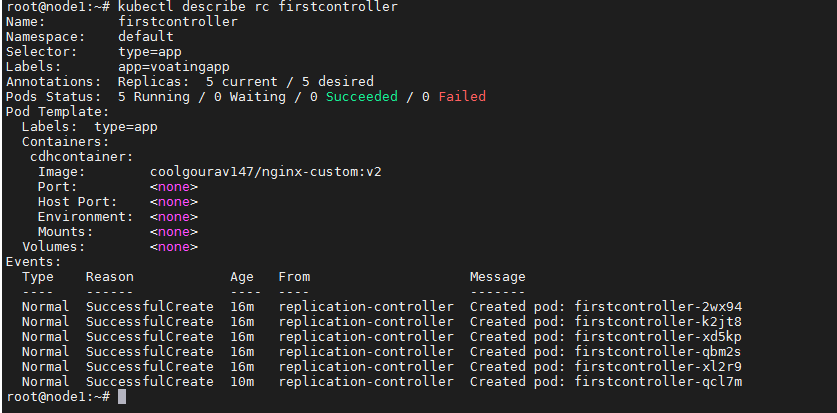




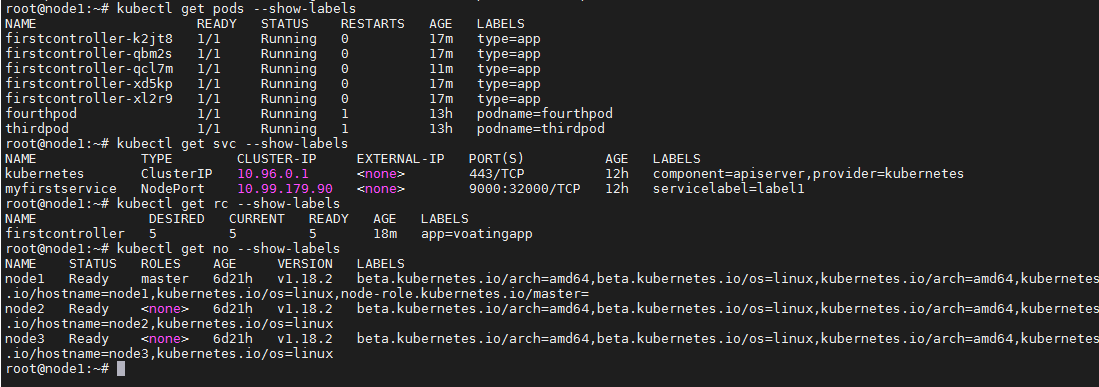




kubectl describe rc firstcontroller



To check labels of pods,rc and nodes



Scenario1

We have created 3 controller yml

1st🡪 pods labels->type=app

2nd-> pods labels->type=app

3rd🡪 pods labels->type=app34

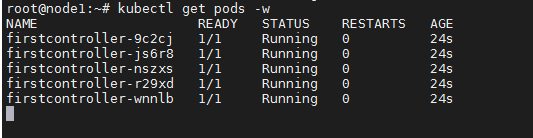
root@node1:~# kubectl apply -f firstsc.yml

replicationcontroller/firstcontroller created

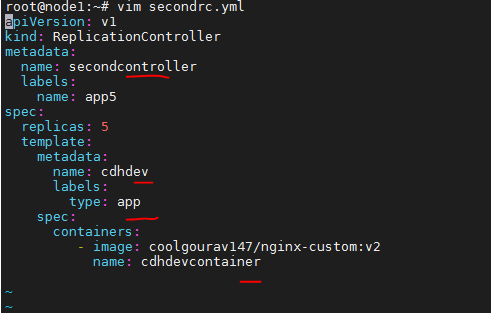




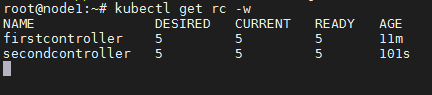


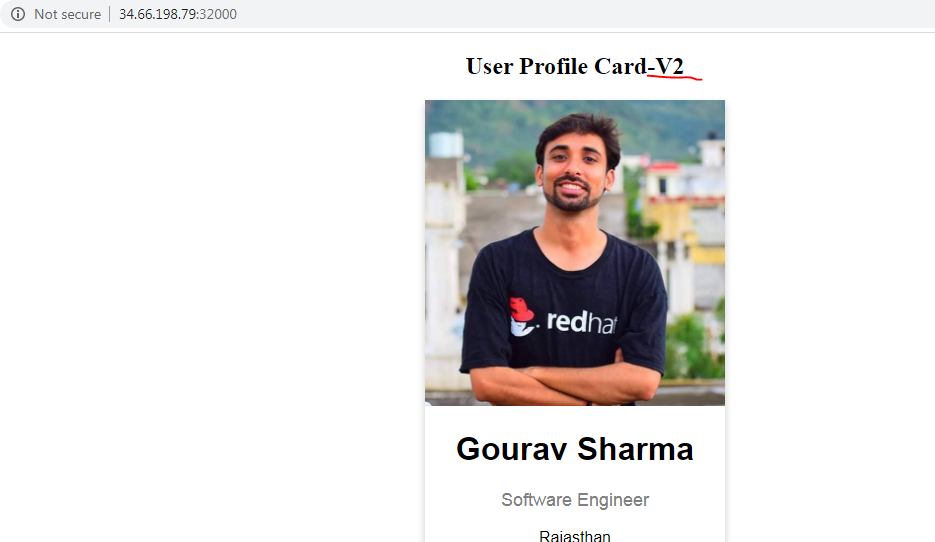


(it will open with latest on while browsing with port)

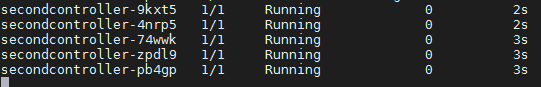




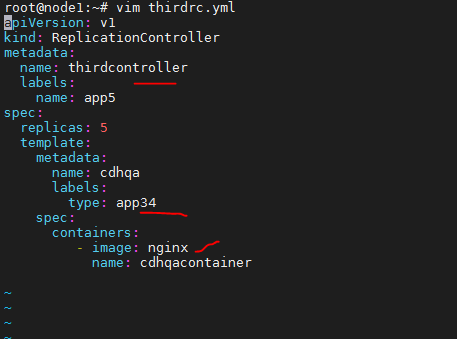




kubectl get pods -w



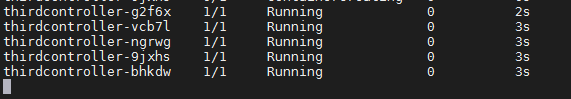
(it will open v2 pic while browsing)

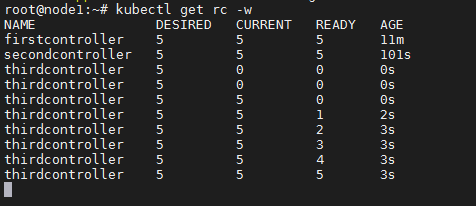


(it should open nginx webpage label type=app34 not match, it should not open)

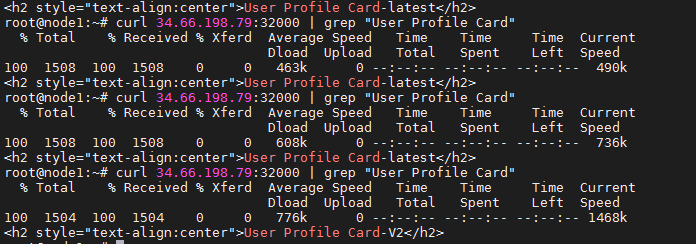


Kubectl get po –w



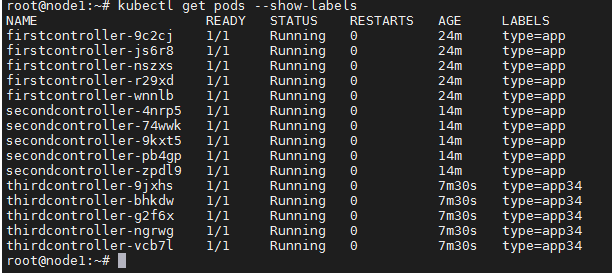


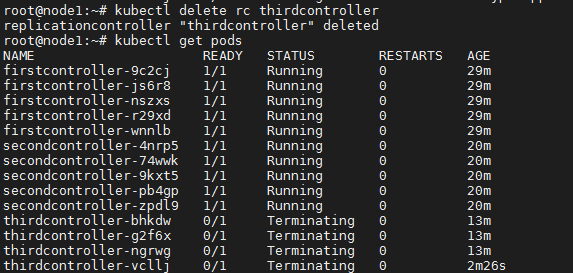
curl 34.66.198.79:32000 | grep "User Profile Card"

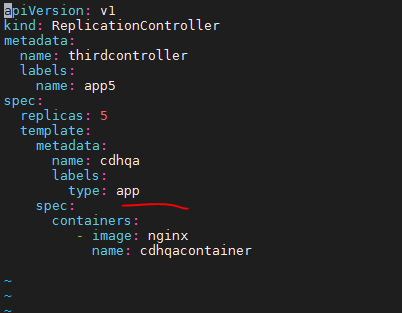


Third one is coming, as label name is different

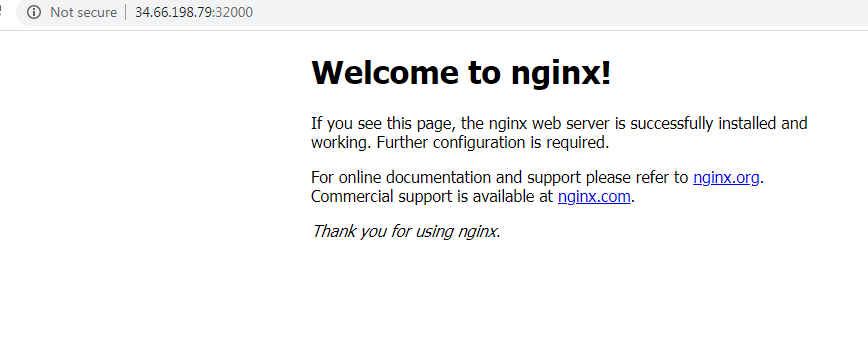
kubectl get pods --show-labels

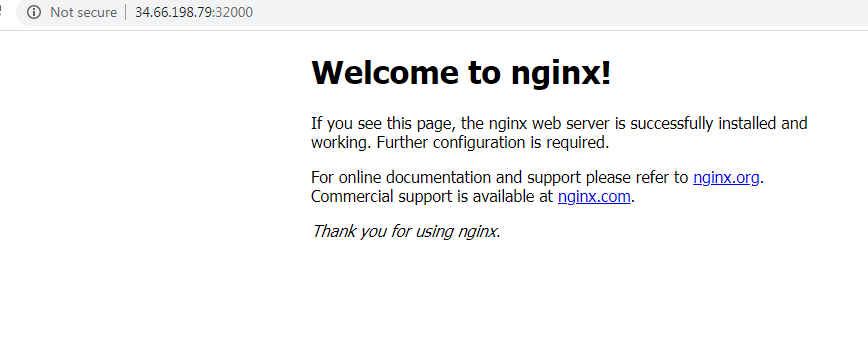






kubectl apply -f thirdrc.yml

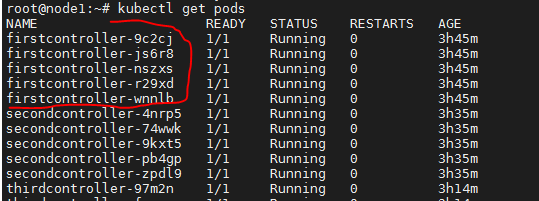




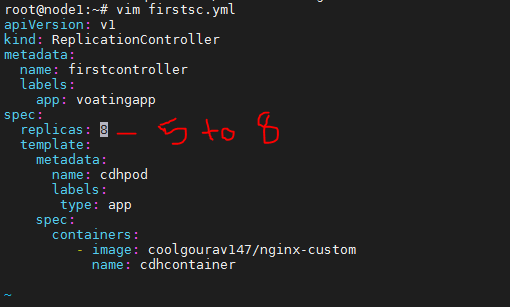
To increase the no.of replication

Now its running with 5 replica

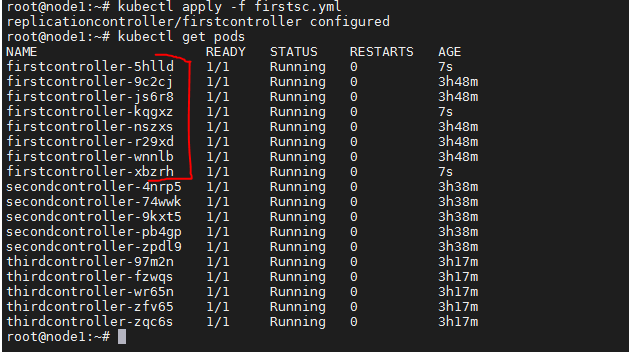
kubectl get pods

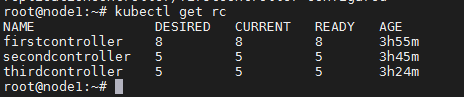


Lets increase the replica to 8

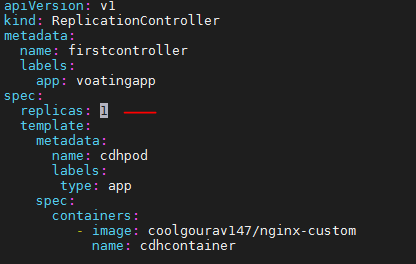


kubectl apply -f firstsc.yml





To decrease the replica

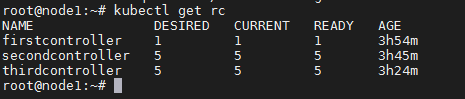


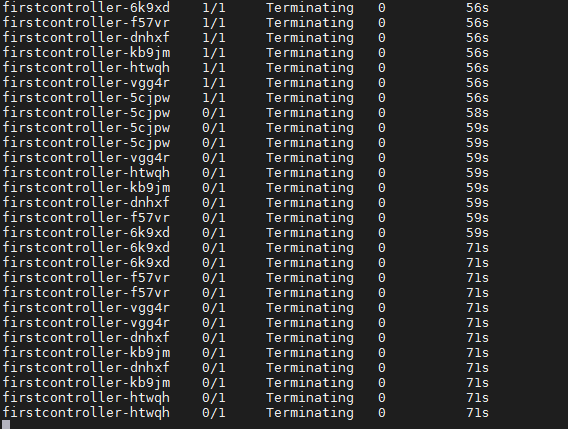
kubectl apply -f firstsc.yml

It started to terminating other 7 , will keep only one

kubectl get pods –w

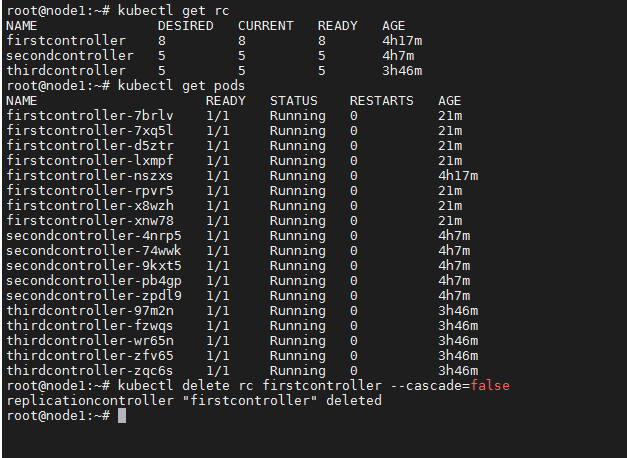


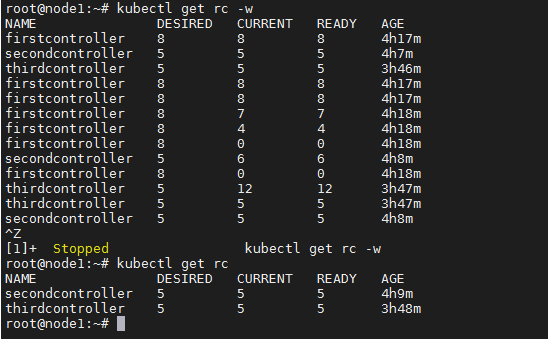




How to remove replication controller without deleting pods

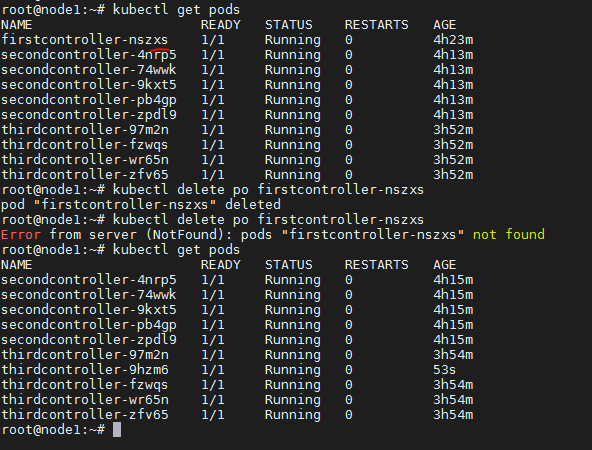
kubectl delete rc firstcontroller --cascade=false



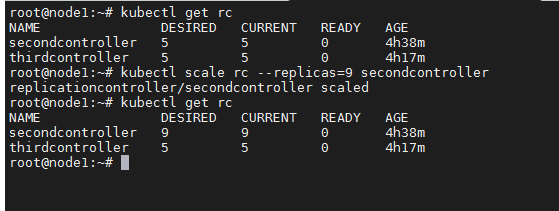


To remove pod after removing rc, now you can see pod will not re-create as we have deleted rc

kubectl delete po firstcontroller-nszxs4

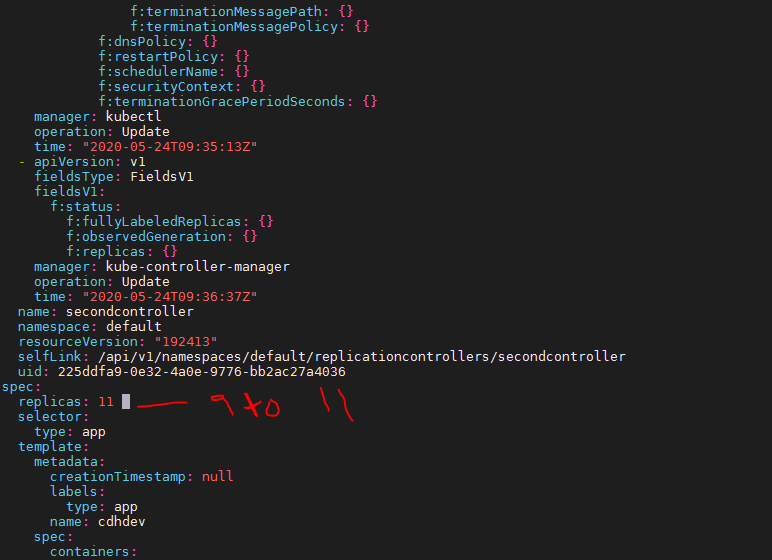


kubectl scale rc --replicas=9 secondcontroller

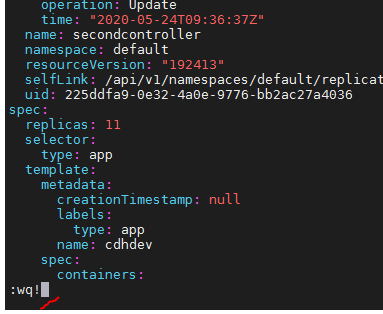


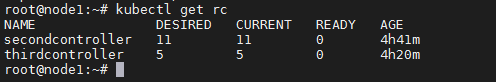
kubectl edit rc secondcontroller



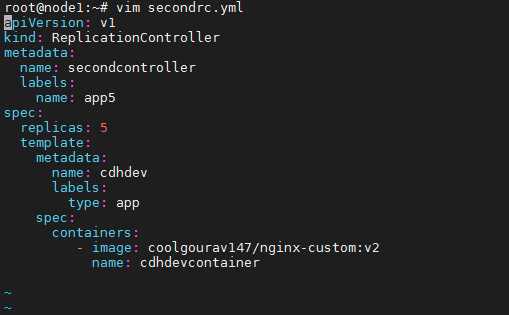


:wq to save it

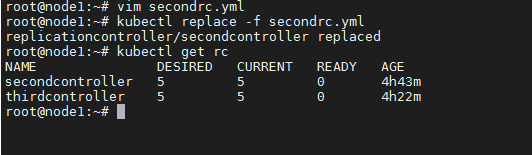




Chnaged replica to 5

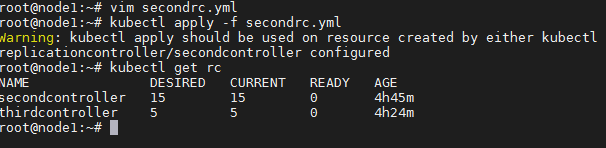


kubectl replace -f secondrc.yml



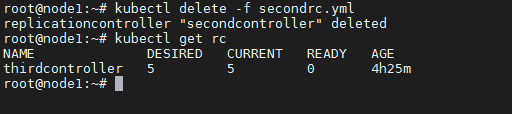
Changed replica to 15





kubectl delete -f secondrc.yml

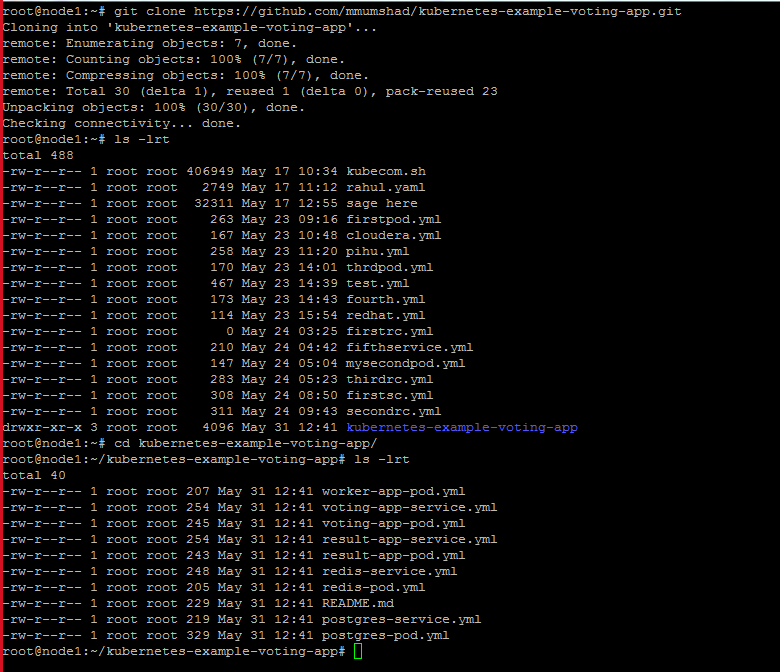
To remove controller



git clone <https://github.com/mmumshad/kubernetes-example-voting-app.git>

cd kubernetes-example-voting-app/

ls –lrt



To create voting app pod

root@node1:~/kubernetes-example-voting-app# **kubectl create -f voting-app-pod.yml**

pod/voting-app-pod created

root@node1:~/kubernetes-example-voting-app# **kubectl get pods**

NAME READY STATUS RESTARTS AGE

voting-app-pod 1/1 Running 0 39s

root@node1:~/kubernetes-example-voting-app# **kubectl create -f voting-app-service.yml**

service/voting-service created

root@node1:~/kubernetes-example-voting-app# **kubectl get svc**

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 5m22s

voting-service LoadBalancer 10.102.76.201 <pending> 80:31308/TCP 7s

root@node1:~/kubernetes-example-voting-app# **kubectl create -f worker-app-pod.yml**

pod/worker-app-pod created

root@node1:~/kubernetes-example-voting-app# **kubectl create -f result-app-pod.yml**

pod/result-app-pod created

root@node1:~/kubernetes-example-voting-app# **kubectl create -f redis-pod.yml**

pod/redis-pod created

root@node1:~/kubernetes-example-voting-app# **kubectl create -f postgres-pod.yml**

pod/postgres-pod created

root@node1:~/kubernetes-example-voting-app# **kubectl get pods -o wide**

NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

postgres-pod 1/1 Running 0 12s 10.244.3.79 node3 <none> <none>

redis-pod 1/1 Running 0 37s 10.244.2.109 node2 <none> <none>

result-app-pod 1/1 Running 0 63s 10.244.3.78 node3 <none> <none>

voting-app-pod 1/1 Running 0 7m59s 10.244.2.108 node2 <none> <none>

worker-app-pod 0/1 Error 3 87s 10.244.3.77 node3 <none> <none>

we face error while creating worker-app-pod, so we delete it and created again after creating all the services

root@node1:~/kubernetes-example-voting-app# **kubectl create -f result-app-service.yml**

service/result-service created

root@node1:~/kubernetes-example-voting-app# **kubectl create -f redis-service.yml**

service/redis created

root@node1:~/kubernetes-example-voting-app# **kubectl create -f postgres-service.yml**

root@node1:~/kubernetes-example-voting-app# **kubectl delete pod worker-app-pod**

pod "worker-app-pod" deleted

root@node1:~/kubernetes-example-voting-app# **kubectl get pods -o wide**

NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

postgres-pod 1/1 Running 0 3m2s 10.244.3.79 node3 <none> <none>

redis-pod 1/1 Running 0 3m27s 10.244.2.109 node2 <none> <none>

result-app-pod 1/1 Running 0 3m53s 10.244.3.78 node3 <none> <none>

voting-app-pod 1/1 Running 0 10m 10.244.2.108 node2 <none> <none>

root@node1:~/kubernetes-example-voting-app# **kubectl get svc**

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

db ClusterIP 10.101.65.221 <none> 5432/TCP 24m

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 39m

redis ClusterIP 10.99.221.130 <none> 6379/TCP 24m

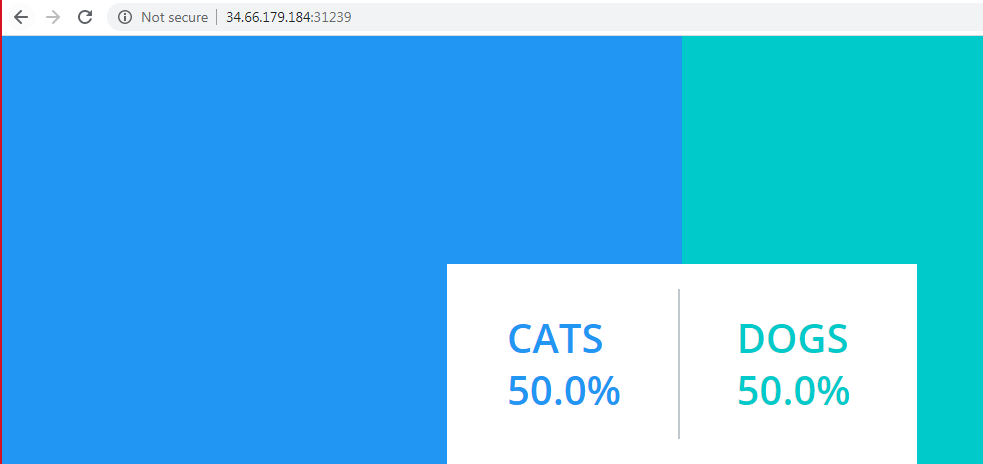
result-service LoadBalancer 10.108.34.200 <pending> 80:31239/TCP 24m

voting-service LoadBalancer 10.100.87.169 <pending> 80:31886/TCP 5m3s

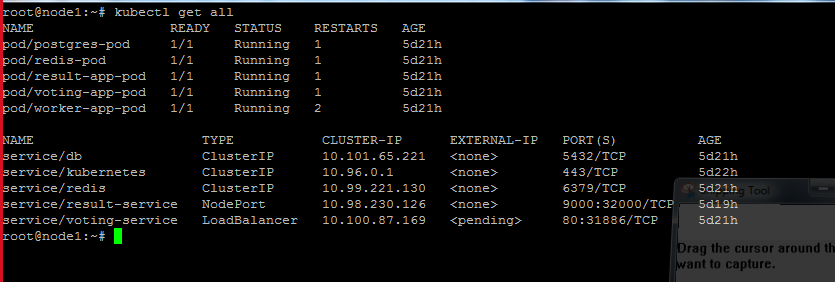
root@node1:~/kubernetes-example-voting-app#

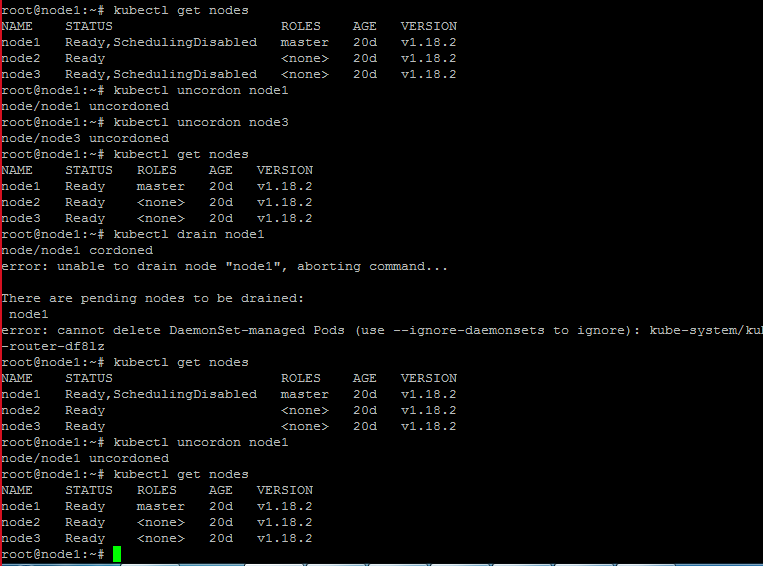
Now we have to browse each node IPwith port number

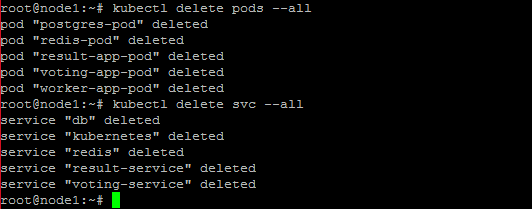
Node1externaIP: port



kubectl get all







Topic 24

To create pod

root@node1:~# cat firstpod.yml

apiVersion: v1

kind: Pod

metadata:

labels:

type: app

name: firstpod

spec:

containers:

- image: coolgourav147/nginx-custom

name: firstcontainer

**To create RC**

root@node1:~# cat firstrc.yml

apiVersion: v1

kind: ReplicationController

metadata:

name: firstcontroller

labels:

name: voatingapp

spec:

replicas: 1

template:

metadata:

name: firstpod

labels:

type: app

spec:

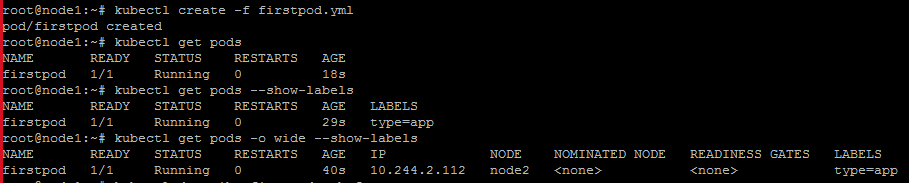
containers:

- image: coolgourav147/nginx-custom

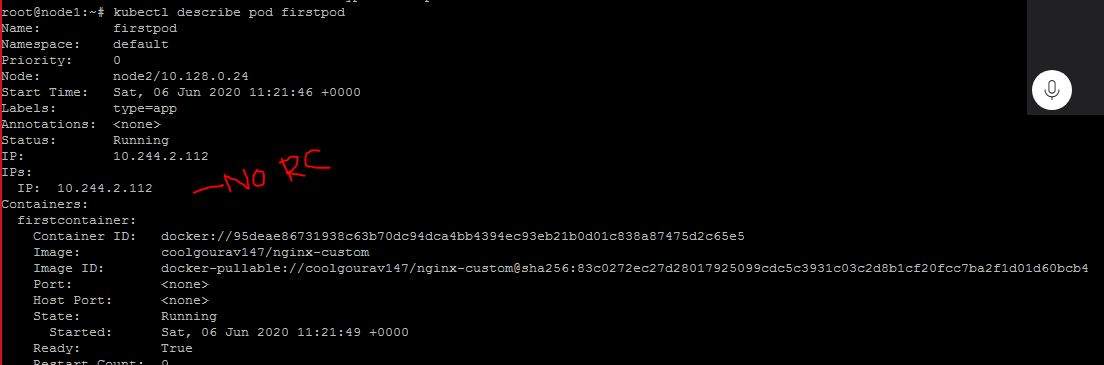
name: firstcontainer



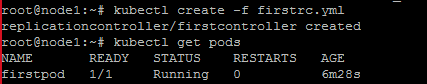


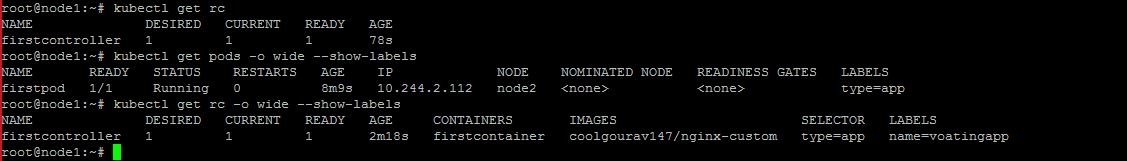


Describe before creating replication controller



As firstpod did not have any controller, but label match as per replication control,so that pod took over by first replication controller

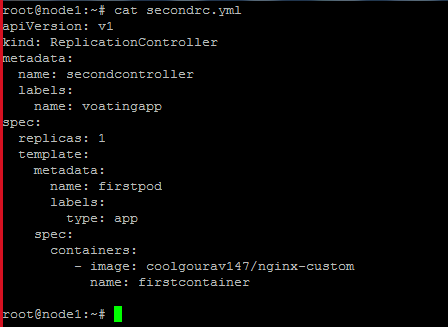




After creating controller

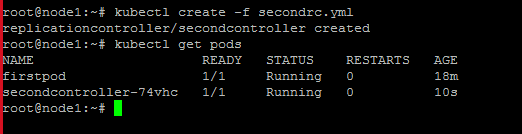


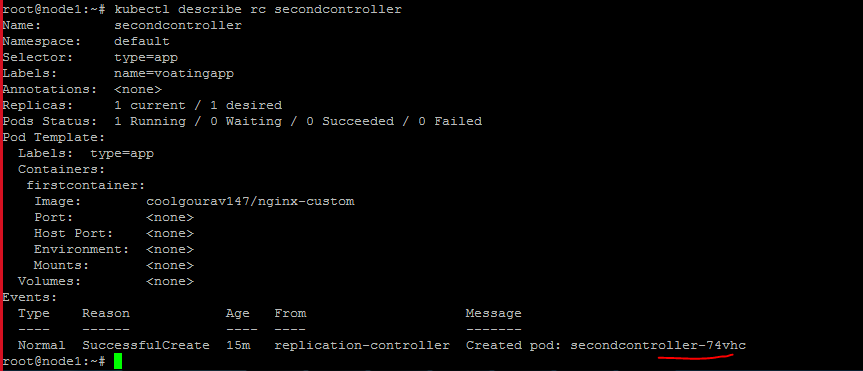
Create another yml for second replication controller



Lets create second replication controller

(here, it create scondcontroller-74vhc pod because firstpod label match with replication controller labels however that is managed by first controller)



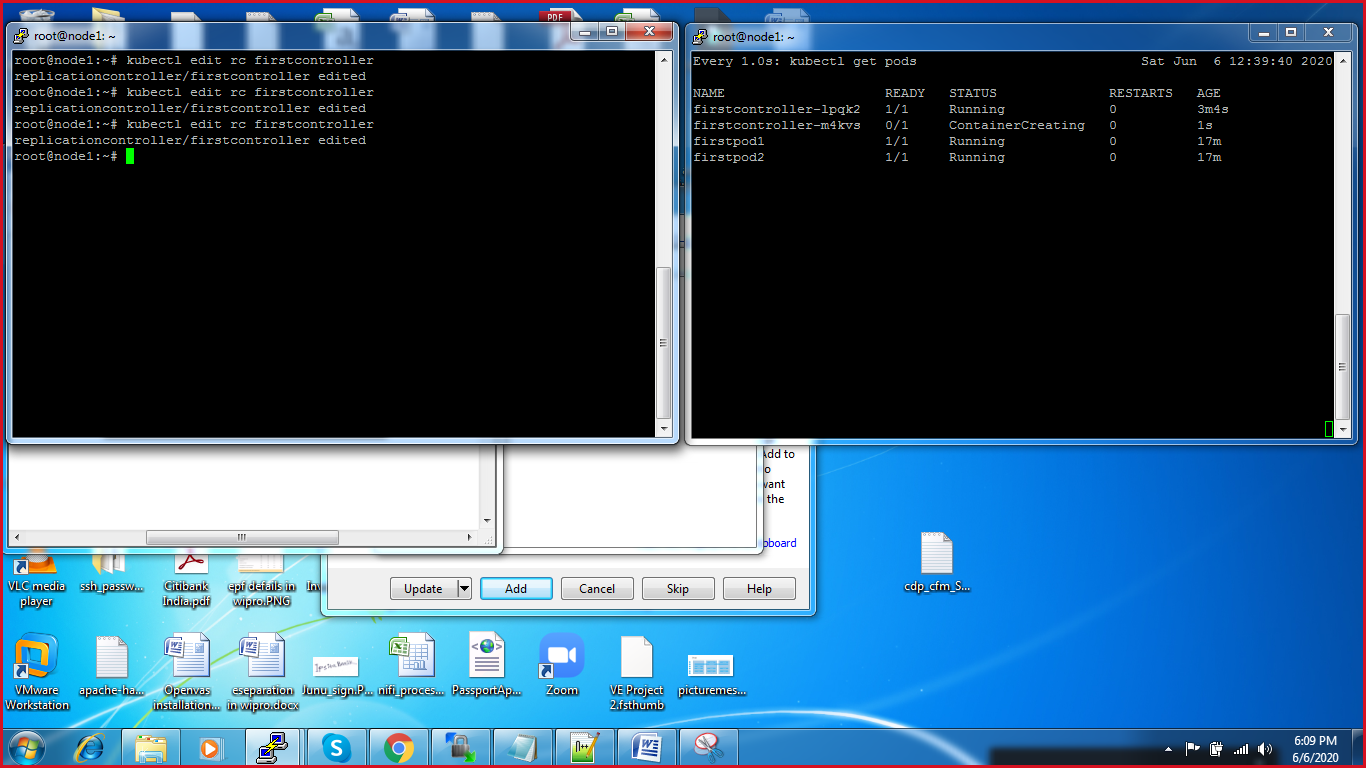


As in replication controller we have mentioned replication 2 which label type=app, so it is deleting the new pod because the new pod is trying to create with same label.



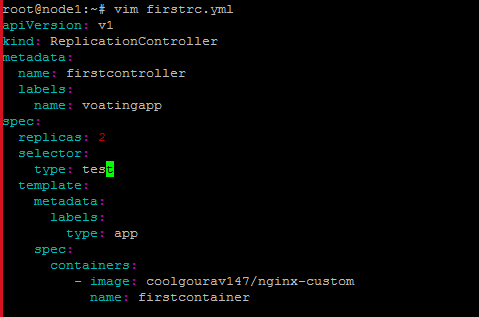
Lets edit rc -> changes replication control replica to 4

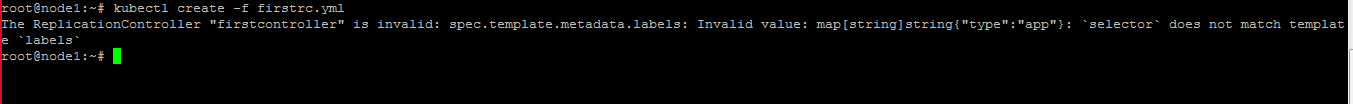
kubectl edit rc firstcontroller



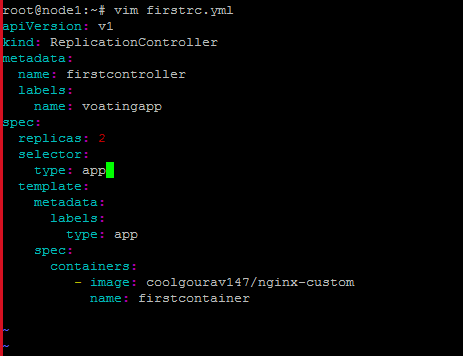
Other 2 replica create automatically

To Use selector for replication controller





Selectors and labels should be same, that’s why throwing error





**Replica Set chapter**

In pod 1 : type:app1

In pod 2: type:app

In RC: type: app

As replica 2.

So

Pod2: will be part of RC –1

Will create replicaton pod-1

root@node1:~# cat firstpod.yml

apiVersion: v1

kind: Pod

metadata:

name: pod2

labels:

app: myapp

type: frontend

spec:

containers:

- image: coolgourav147/nginx-custom

name: firstcontainer

root@node1:~# kubectl create -f firstpod.yml

pod/pod2 created

root@node1:~# cp firstpod.yml first1pod.yml

root@node1:~# vim first1pod.yml

root@node1:~# cat first1pod.yml

apiVersion: v1

kind: Pod

metadata:

name: pod1

labels:

app: myapp1

type: frontend

spec:

containers:

- image: coolgourav147/nginx-custom

name: firstcontainer

root@node1:~# kubectl create -f first1pod.yml

pod/pod1 created

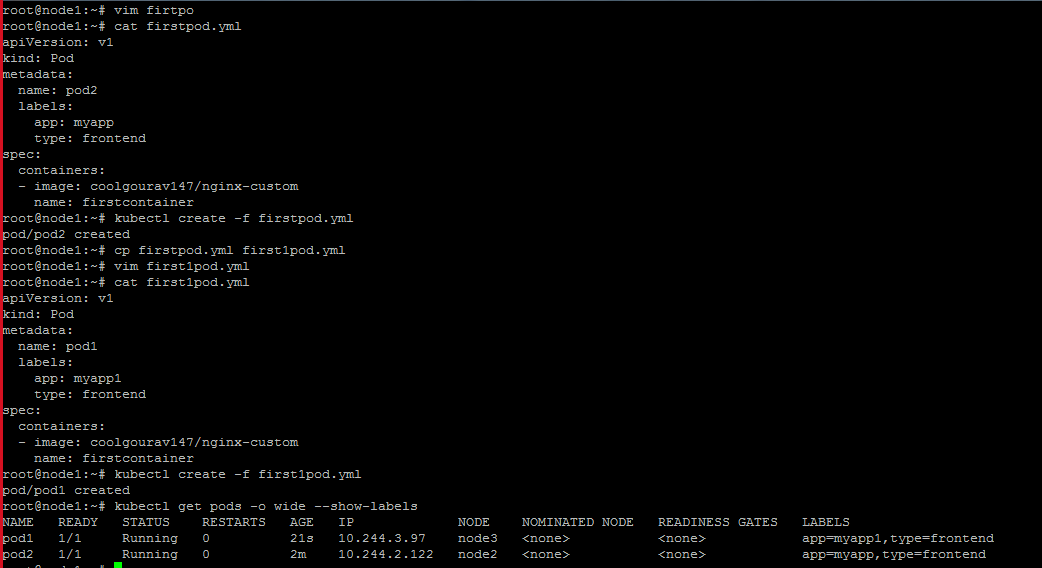
root@node1:~# kubectl get pods -o wide --show-labels

NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES LABELS

pod1 1/1 Running 0 21s 10.244.3.97 node3 <none> <none> app=myapp1,type=frontend

pod2 1/1 Running 0 2m 10.244.2.122 node2 <none> <none> app=myapp,type=frontend

root@node1:~#



root@node1:~# cat firstrc.yml

apiVersion: v1

kind: ReplicationController

metadata:

name: firstrc

labels:

name: firstrc

spec:

replicas: 2

selector:

app: myapp

template:

metadata:

name: pod

labels:

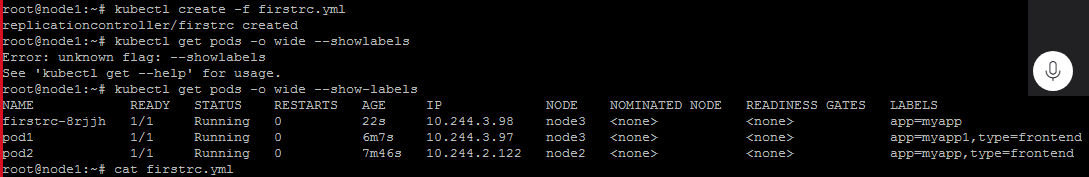
app: myapp

spec:

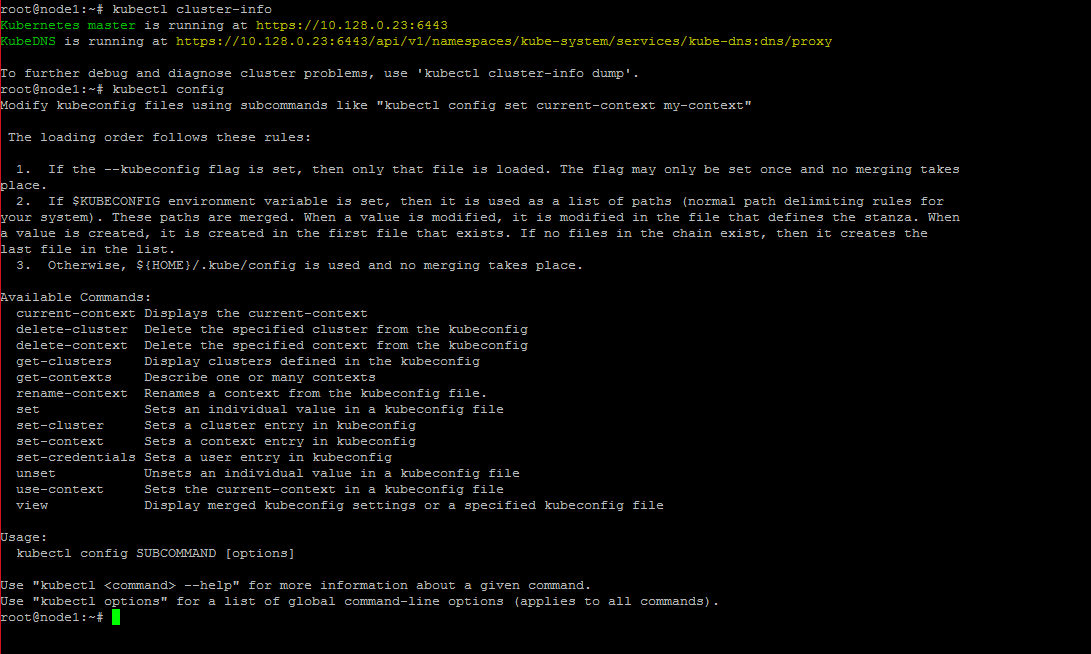
containers:

- image: coolgourav147/nginx-custom

name: firstcontainer



Cluster related commands



**Replica Set chapter**

In pod 1 : type:app

In pod 2: type:app

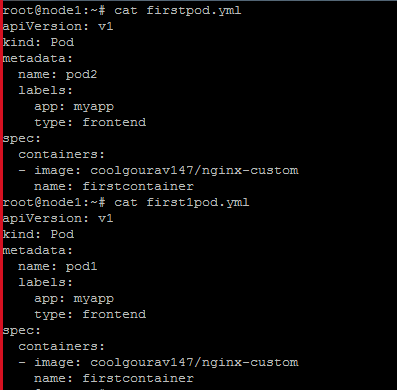
In RC: type: app

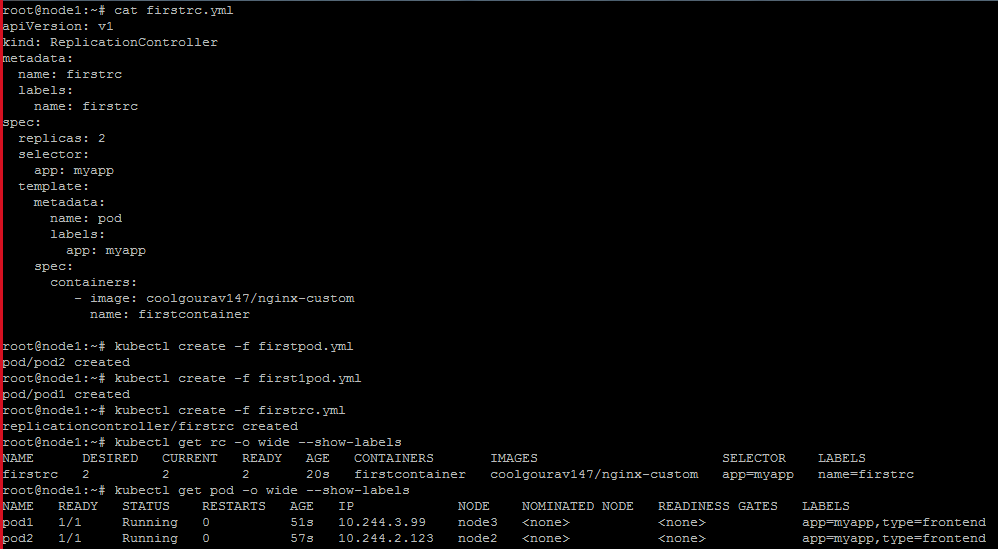
As replica 2.

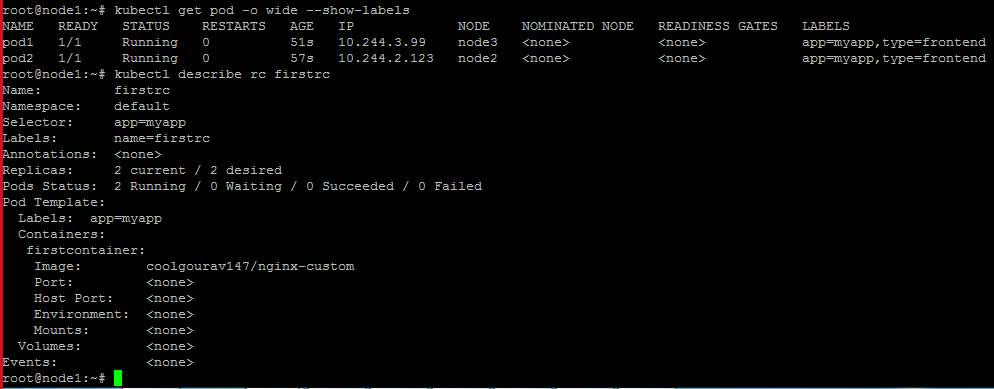
So

Pod2: will be part of RC –1

Pod1: will be part of RC 1







**Replica Set chapter**

**Scn-3**

In pod 1 : type:app1

In pod 2: type:app2

In RC: type: app

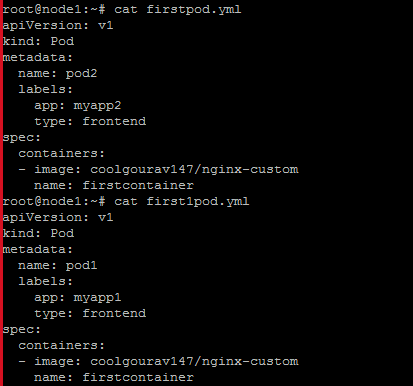
As replica 2.

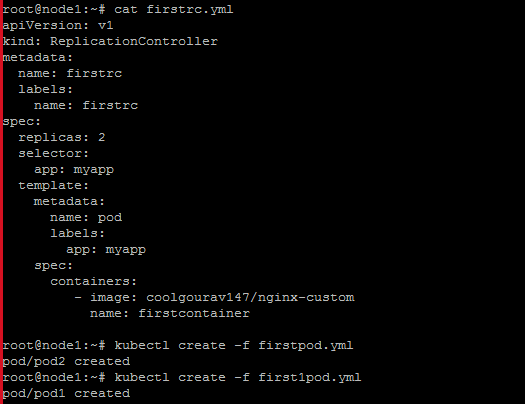
So

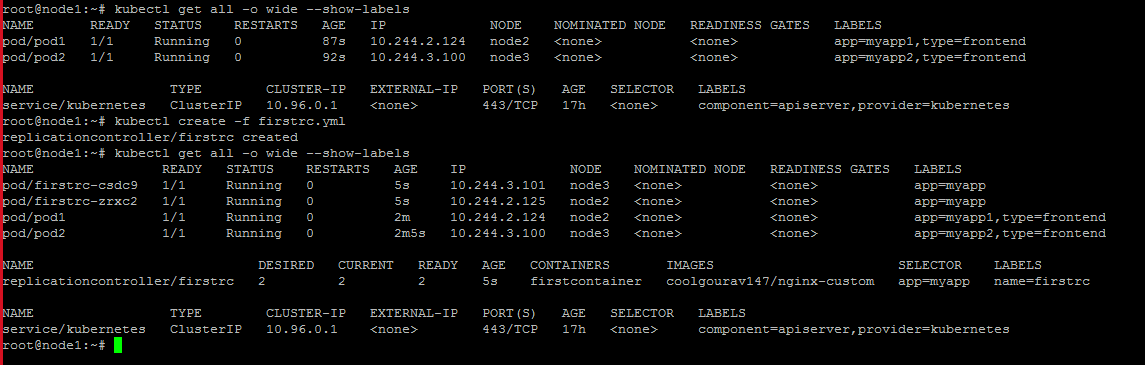
Pod2: will be not part of RC –1

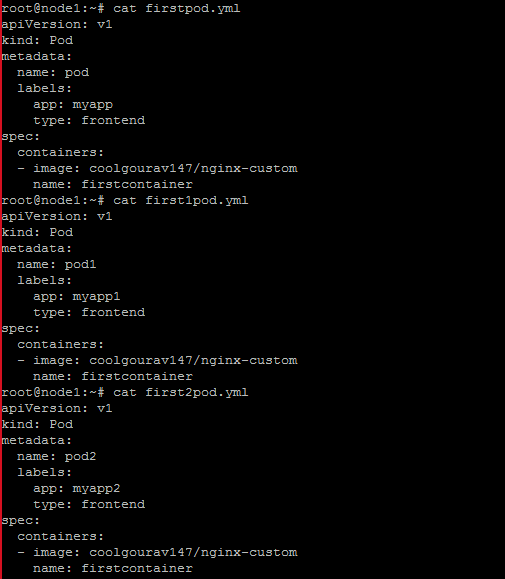
Pod1: will be not part of RC 1

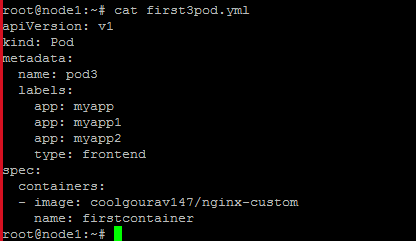
RC will create 2 replica with RCname







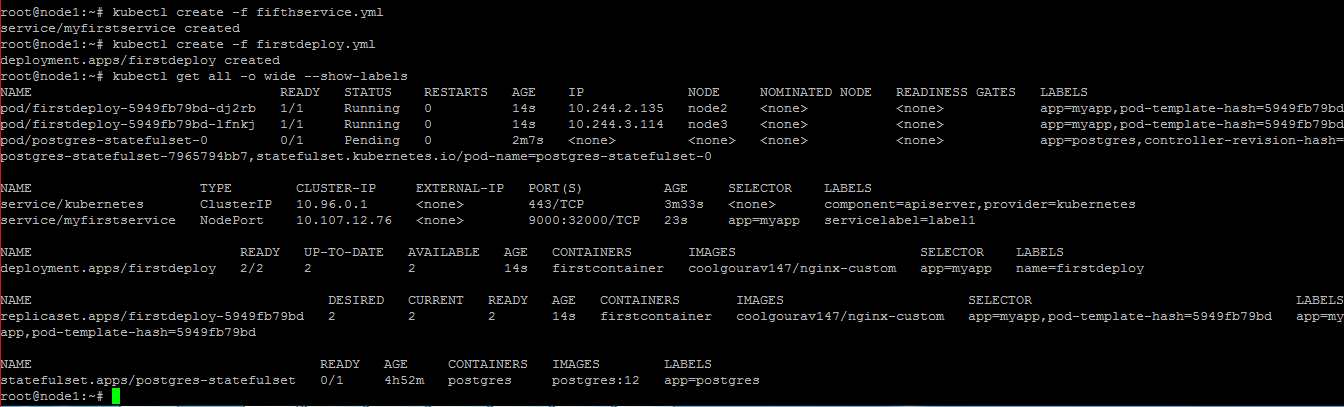




**Video no 27**

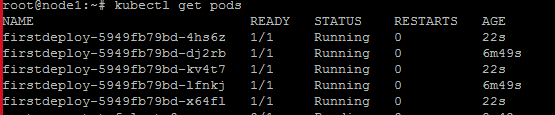
**Deployement**

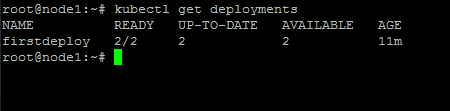
****

****

**Change replica to 5**

****

****

****

**Video no -28**