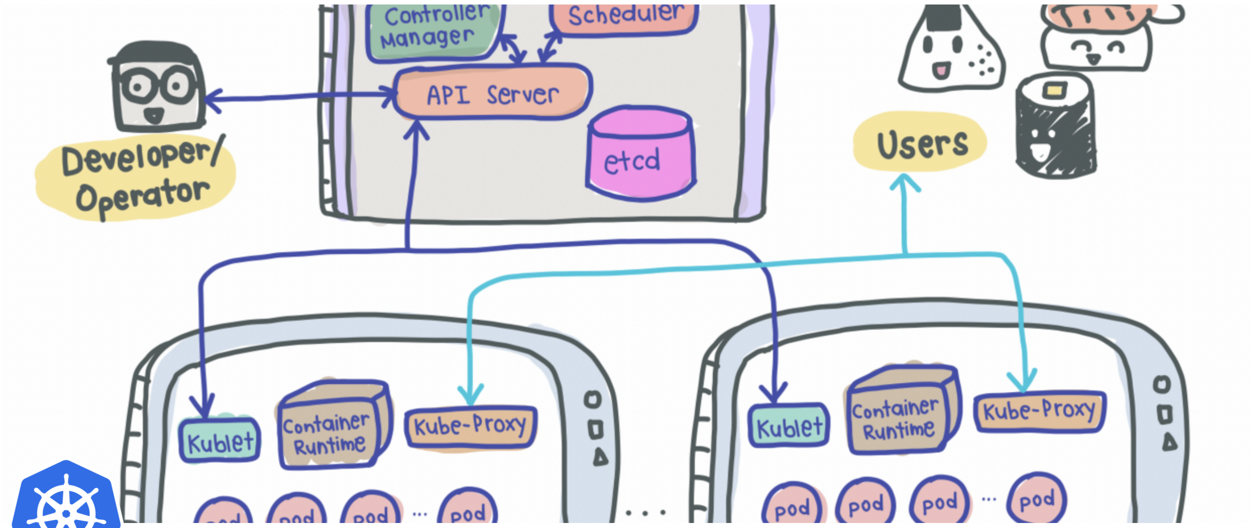
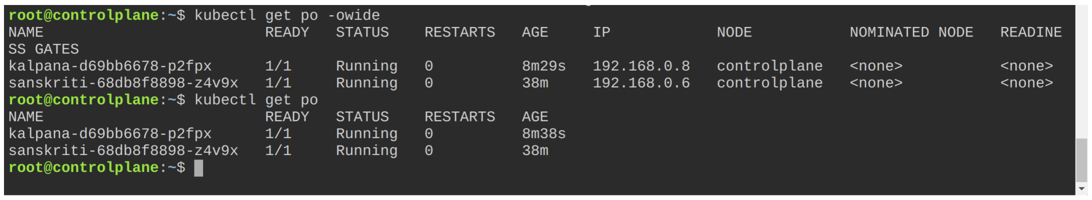


Kubernetes Notes

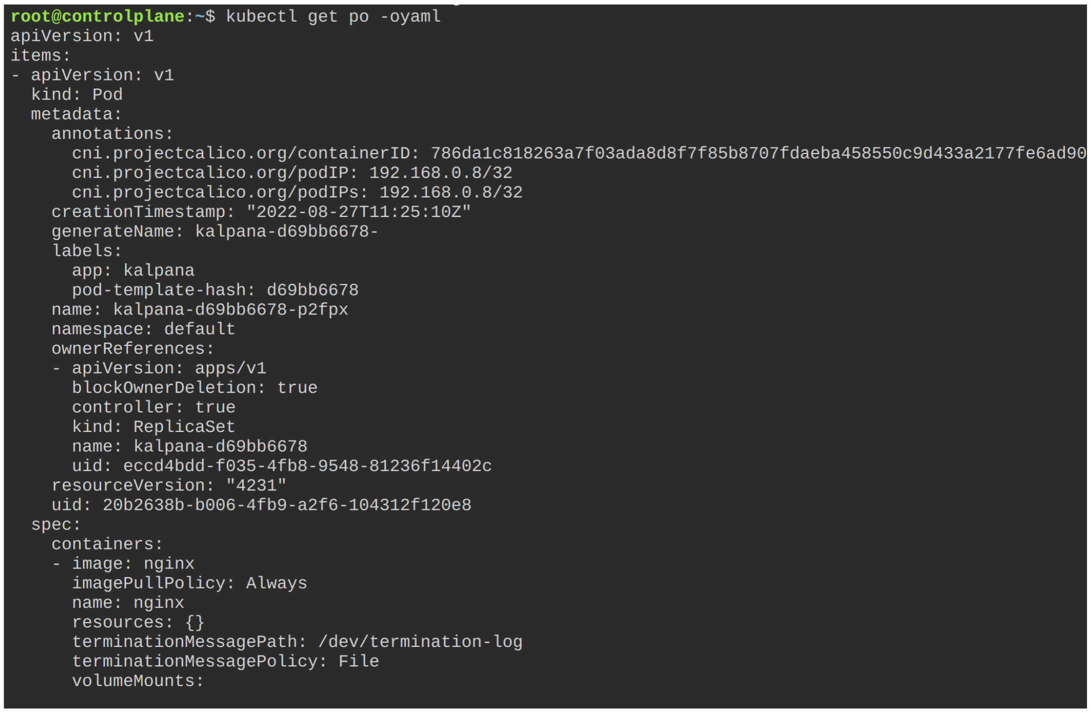
Kubernetes Tutorial

- `kubectl get nodes`
- `kubectl get pods` - instead of `pods`, you can also write `pod` or `po`
- Don't run your application databases in kube-system namespace because that is reserved for the system.
- `-owide`
  - to check which node pods are running on.



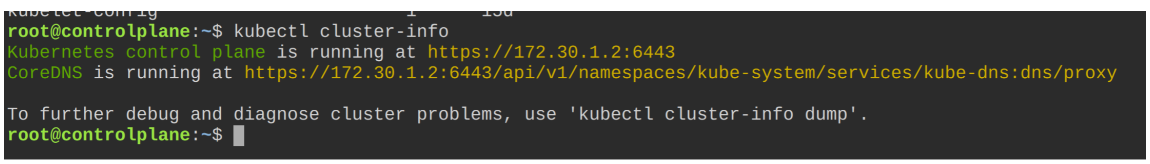
```
root@controlplane:~$ kubectl get po -owide
NAME                            READY   STATUS    RESTARTS   AGE     IP            NODE           NOMINATED NODE   READINE
SS GATES
kalpana-d69bb6678-p2fpx         1/1     Running   0          8m29s   192.168.0.8   controlplane   <none>           <none>
sanskriti-68db8f8898-z4v9x      1/1     Running   0          38m     192.168.0.6   controlplane   <none>           <none>
root@controlplane:~$ kubectl get po
NAME                            READY   STATUS    RESTARTS   AGE
kalpana-d69bb6678-p2fpx         1/1     Running   0          8m38s
sanskriti-68db8f8898-z4v9x      1/1     Running   0          38m
root@controlplane:~$ █
```

- `-oyaml`
  - To get the details in the yaml format. Other option `-ojson`



```
root@controlplane:~$ kubectl get po -oyaml
apiVersion: v1
items:
- apiVersion: v1
  kind: Pod
  metadata:
    annotations:
      cni.projectcalico.org/containerID: 786da1c818263a7f03ada8d8f7f85b8707fdaeba458550c9d433a2177fe6ad90
      cni.projectcalico.org/podIP: 192.168.0.8/32
      cni.projectcalico.org/podIPs: 192.168.0.8/32
    creationTimestamp: "2022-08-27T11:25:10Z"
    generateName: kalpana-d69bb6678-
    labels:
      app: kalpana
      pod-template-hash: d69bb6678
    name: kalpana-d69bb6678-p2fpx
    namespace: default
    ownerReferences:
    - apiVersion: apps/v1
      blockOwnerDeletion: true
      controller: true
      kind: ReplicaSet
      name: kalpana-d69bb6678
      uid: eccd4bdd-f035-4fb8-9548-81236f14402c
    resourceVersion: "4231"
    uid: 20b2638b-b006-4fb9-a2f6-104312f120e8
  spec:
    containers:
    - image: nginx
      imagePullPolicy: Always
      name: nginx
      resources: {}
      terminationMessagePath: /dev/termination-log
      terminationMessagePolicy: File
      volumeMounts:
```

- `kubectl get cm -n kube-system`
- `kubectl cluster-info`



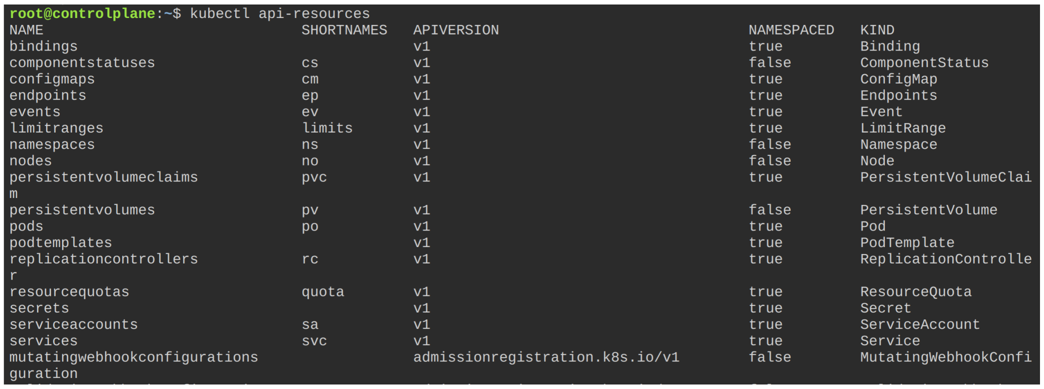
```
kubelet-config                        1      15d
root@controlplane:~$ kubectl cluster-info
Kubernetes control plane is running at https://172.30.1.2:6443
CoreDNS is running at https://172.30.1.2:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
root@controlplane:~$ 
```

Namespaces

- quota, control, policies.
- `kubectl get ns`
- `kubectl get pods -n default` → display the pods belonging to *default* namespace.
- Each namespace belong to separate team for separate purposes. Eg, dev team and test team.
- Makes it easy to aggregate and isolate various components belonging to a particuar ns, let's say service mesh, application, database, monitoring.

- Ideally, let’s say you create a Monitoring ns and add monitoring components over there. Easy to monitor, aggregate and isolate.
- Don’t blindly deploy everything into default ns.
- **Namespaced or not:**
    - `kubectl api-resources`



```
root@controlplane:~$ kubectl api-resources
NAME                              SHORTNAMES   APIVERSION                        NAMESPACED   KIND
bindings                                       v1                                true         Binding
componentstatuses                 cs           v1                                false        ComponentStatus
configmaps                        cm           v1                                true         ConfigMap
endpoints                         ep           v1                                true         Endpoints
events                            ev           v1                                true         Event
limitranges                       limits       v1                                true         LimitRange
namespaces                        ns           v1                                false        Namespace
nodes                             no           v1                                false        Node
persistentvolumeclaims            pvc          v1                                true         PersistentVolumeClai
m
persistentvolumes                 pv           v1                                false        PersistentVolume
pods                              po           v1                                true         Pod
podtemplates                                   v1                                true         PodTemplate
replicationcontrollers            rc           v1                                true         ReplicationControlle
r
resourcequotas                    quota        v1                                true         ResourceQuota
secrets                                        v1                                true         Secret
serviceaccounts                   sa           v1                                true         ServiceAccount
services                          svc          v1                                true         Service
mutatingwebhookconfigurations                  admissionregistration.k8s.io/v1   false        MutatingWebhookConfi
guration
```

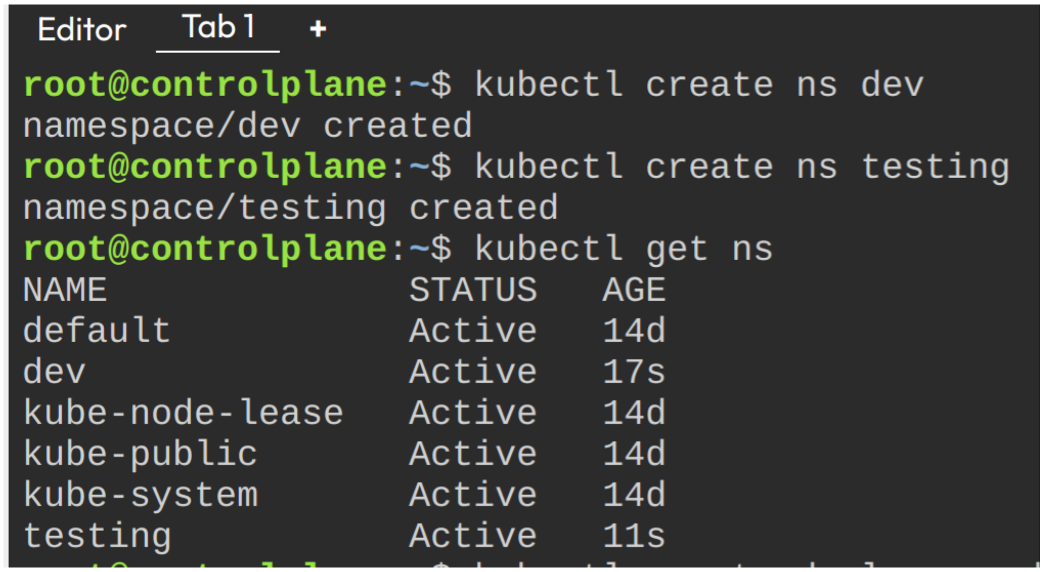
Observe the NAMESPCAED column. Some have a value of ‘true’ and some has ‘false’ as it’s value.

`kubectl api-resources --namespaced=false`

- You can create same name of deployment but in different namespaces.

▼ How to create a namespace (Imperative way)

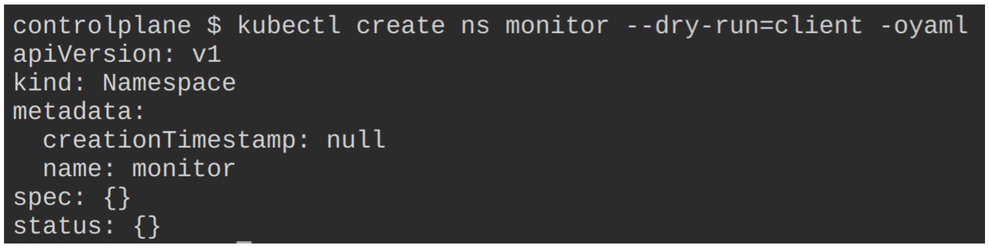
- `kubectl create ns dev` : will create a ns named *dev*



```
root@controlplane:~$ kubectl create ns dev
namespace/dev created
root@controlplane:~$ kubectl create ns testing
namespace/testing created
root@controlplane:~$ kubectl get ns
NAME              STATUS   AGE
default           Active   14d
dev               Active   17s
kube-node-lease   Active   14d
kube-public       Active   14d
kube-system       Active   14d
testing           Active   11s
```

▼ How to create a namespace (Declarative way)

- Let's say we want to create a ns named *monitor* using declarative way.
- `kubectl create ns monitor --dry-run=client -oyaml`
  - `--dry-run=client` flag to **preview the object that would be sent to your cluster, without really submitting it**.
  - `-oyaml` flag displays the output in the *yaml* format. You can replace it with `-ojson` to view the result in *json* format.

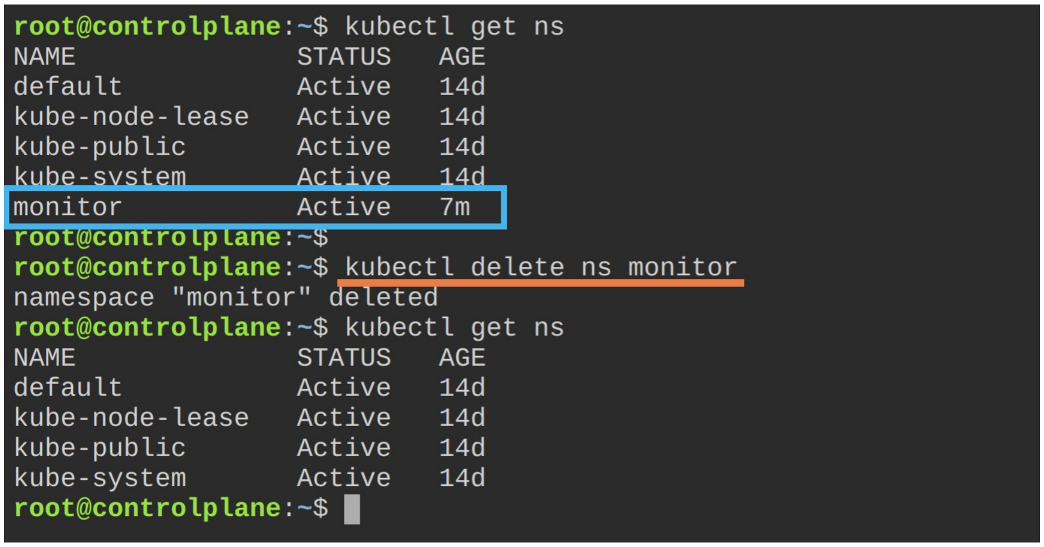


```
controlplane $ kubectl create ns monitor --dry-run=client -oyaml
apiVersion: v1
kind: Namespace
metadata:
  creationTimestamp: null
  name: monitor
spec: {}
status: {}
```

This ocmmand is not creating a ns but is rather previewing how the object will be sent (in this case as a .yaml file) to your cluster.

▼ How to delete a ns?

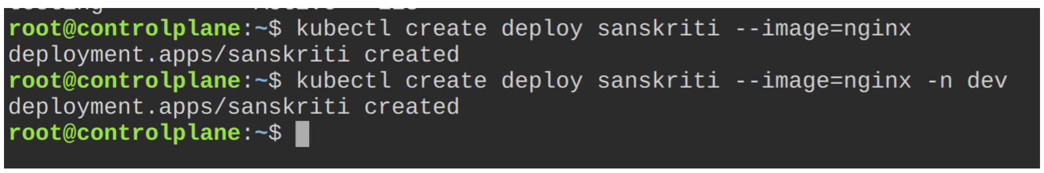
- `kubectl delete ns <nameSpace>`



```
root@controlplane:~$ kubectl get ns
NAME              STATUS   AGE
default           Active   14d
kube-node-lease   Active   14d
kube-public       Active   14d
kube-system       Active   14d
monitor           Active   7m
root@controlplane:~$
root@controlplane:~$ kubectl delete ns monitor
namespace "monitor" deleted
root@controlplane:~$ kubectl get ns
NAME              STATUS   AGE
default           Active   14d
kube-node-lease   Active   14d
kube-public       Active   14d
kube-system       Active   14d
root@controlplane:~$
```

▼ How to create a deployment

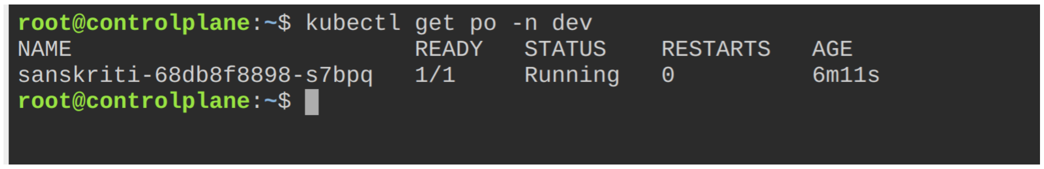
- `kubectl create deploy sanskriti --image=nginx`
- `kubectl create deploy sanskriti --image=nginx -n dev` : specifies the namespace to which *sanskriti* belong.



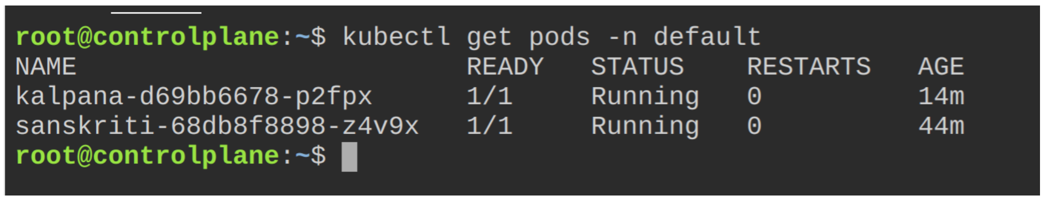
```
root@controlplane:~$ kubectl create deploy sanskriti --image=nginx
deployment.apps/sanskriti created
root@controlplane:~$ kubectl create deploy sanskriti --image=nginx -n dev
deployment.apps/sanskriti created
root@controlplane:~$
```

▼ How to see which pods belong to a particular ns?

- `kubectl get pods -n <namespace>`

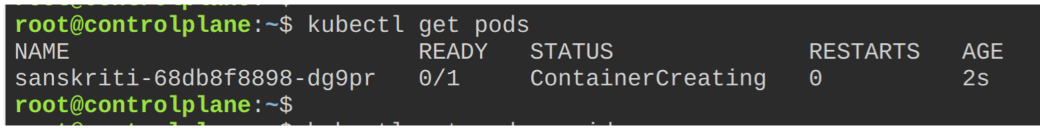


```
root@controlplane:~$ kubectl get po -n dev
NAME                         READY   STATUS    RESTARTS   AGE
sanskriti-68db8f8898-s7bpq   1/1     Running   0          6m11s
root@controlplane:~$
```



```
root@controlplane:~$ kubectl get pods -n default
NAME                         READY   STATUS    RESTARTS   AGE
kalpana-d69bb6678-p2fpx      1/1     Running   0          14m
sanskriti-68db8f8898-z4v9x   1/1     Running   0          44m
root@controlplane:~$
```

- In case you don't specify the ns then the pods belonging to the *default* ns will be displayed.

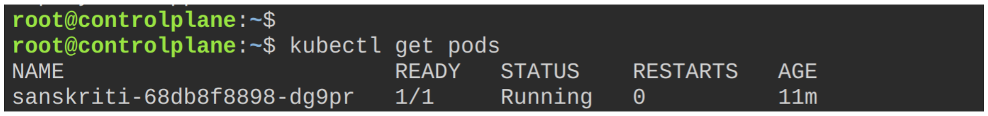


```
root@controlplane:~$ kubectl get pods
NAME                         READY   STATUS              RESTARTS   AGE
sanskriti-68db8f8898-dg9pr   0/1     ContainerCreating   0          2s
root@controlplane:~$
```

- But what if I don't want to display pods belongs to *default* ns whenever I run the `kubectl get pods` command ? What if I want this command to display pods belonging to the *dev* ns?
  - **Ans:** Change the namespace context from current to the desired one.

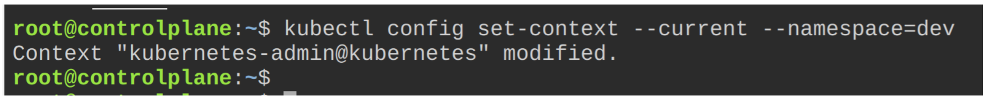
▼ How to change context in namespaces?

- `kubectl config set-context --current --namespace=dev` : This command will change your current ns context to *dev* ns.
  - `--namespace=dev` → specifies that the "namespace" context has to be changed from *current* ns to *dev* ns.
- Eg, Let's say currently we are in *default* ns and there is only one pod named *sanskriti* belonging to this ns. Now, I want the ns context to switch to *dev* ns to which pods *kalpana* and *sunita* belongs to.
  - As you can see that the pods belonging to *defalut* ns are being displayed.



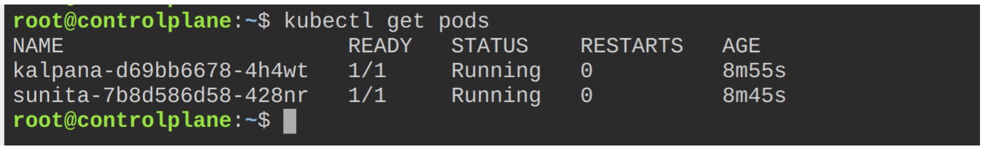
```
root@controlplane:~$
root@controlplane:~$ kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
sanskriti-68db8f8898-dg9pr   1/1     Running   0          11m
```

  - Now, switch the context by running the above command.



```
root@controlplane:~$ kubectl config set-context --current --namespace=dev
Context "kubernetes-admin@kubernetes" modified.
root@controlplane:~$
```

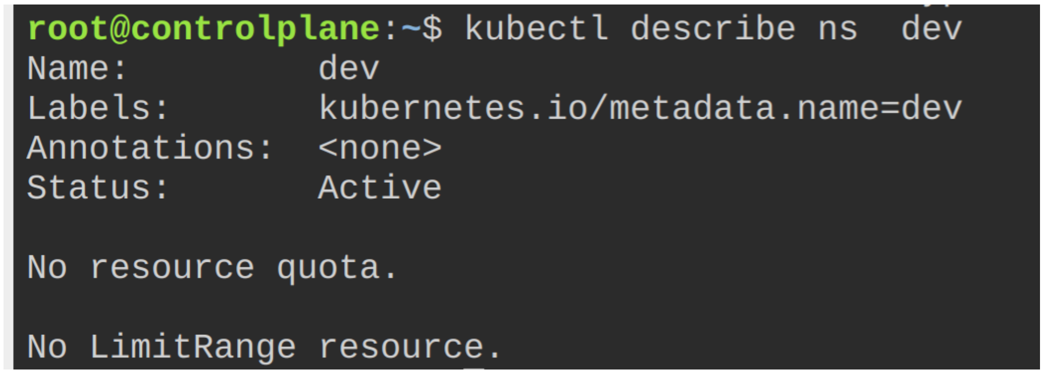
- Again, run `kubectl get pods` ; now you can see that the *default* ns context has been switched to *dev* ns as it is displaying pods belonging to *dev* ns.



```
root@controlplane:~$ kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
kalpana-d69bb6678-4h4wt      1/1     Running   0          8m55s
sunita-7b8d586d58-428nr      1/1     Running   0          8m45s
root@controlplane:~$
```

▼ Description of a namespace

- `kubectl describe ns <nameSpace>`



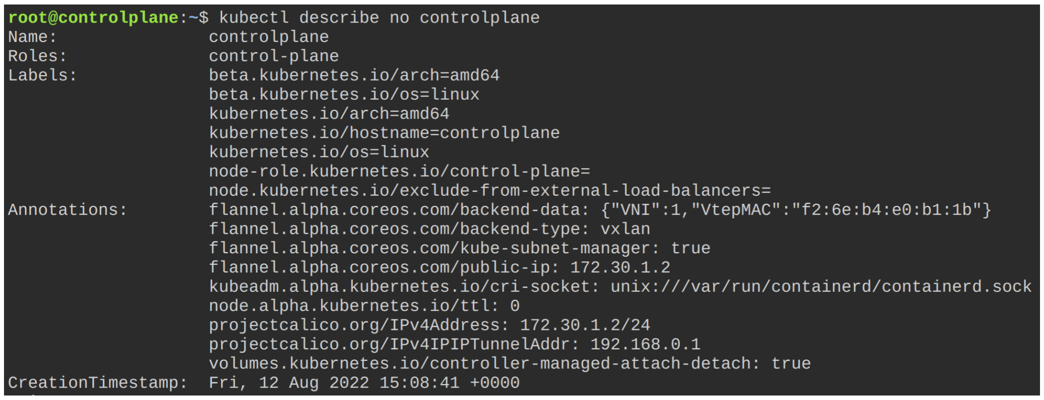
```
root@controlplane:~$ kubectl describe ns  dev
Name:         dev
Labels:       kubernetes.io/metadata.name=dev
Annotations:  <none>
Status:       Active

No resource quota.

No LimitRange resource.
```

▼ Description of a node

- `kubectl describe no`
- `kubectl describe no controlplane` → specifying that *controlplane* node should be descibed.
- `kubectl describe no <nodeName>`

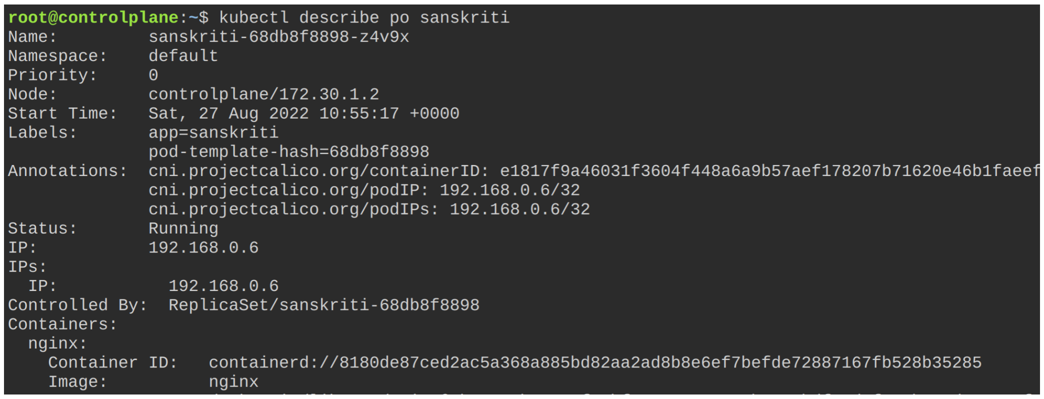


```
root@controlplane:~$ kubectl describe no controlplane
Name:               controlplane
Roles:              control-plane
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=controlplane
                    kubernetes.io/os=linux
                    node-role.kubernetes.io/control-plane=
                    node.kubernetes.io/exclude-from-external-load-balancers=
Annotations:        flannel.alpha.coreos.com/backend-data: {"VNI":1,"VtepMAC":"f2:6e:b4:e0:b1:1b"}
                    flannel.alpha.coreos.com/backend-type: vxlan
                    flannel.alpha.coreos.com/kube-subnet-manager: true
                    flannel.alpha.coreos.com/public-ip: 172.30.1.2
                    kubeadm.alpha.kubernetes.io/cri-socket: unix:///var/run/containerd/containerd.sock
                    node.alpha.kubernetes.io/ttl: 0
                    projectcalico.org/IPv4Address: 172.30.1.2/24
                    projectcalico.org/IPv4IPIPTunnelAddr: 192.168.0.1
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Fri, 12 Aug 2022 15:08:41 +0000
```

▼ Description of a pod

I create two deployments with the same name i.e *sanskriti.* One belongs to the default ns and the other one belons to the *dev* ns that I have created earlier.

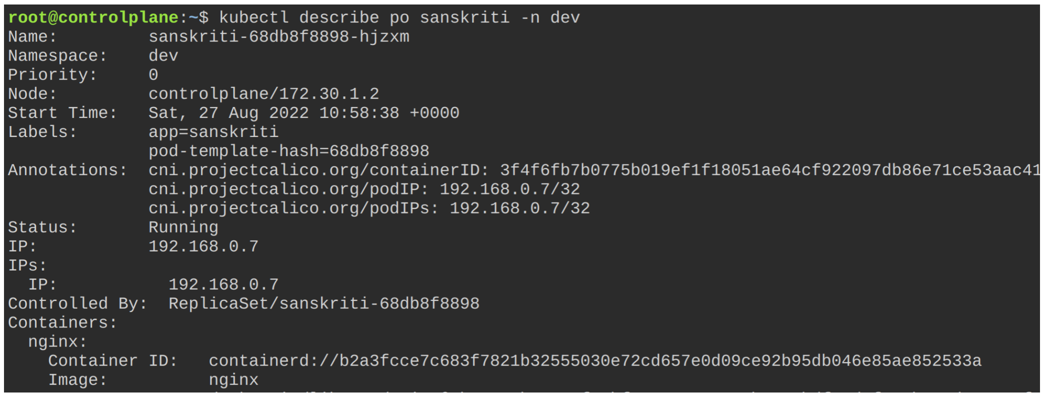
- `kubectl describe po` → describe all the pods belonging to the *default* ns.
- `kubectl describe po <podName>` → describes pod named *podName* belonging to the *default* ns.



```
root@controlplane:~$ kubectl describe po sanskriti
Name:         sanskriti-68db8f8898-z4v9x
Namespace:    default
Priority:     0
Node:         controlplane/172.30.1.2
Start Time:   Sat, 27 Aug 2022 10:55:17 +0000
Labels:       app=sanskriti
              pod-template-hash=68db8f8898
Annotations:  cni.projectcalico.org/containerID: e1817f9a46031f3604f448a6a9b57aef178207b71620e46b1faeef
              cni.projectcalico.org/podIP: 192.168.0.6/32
              cni.projectcalico.org/podIPs: 192.168.0.6/32
Status:       Running
IP:           192.168.0.6
IPs:
  IP:           192.168.0.6
Controlled By:  ReplicaSet/sanskriti-68db8f8898
Containers:
  nginx:
    Container ID:   containerd://8180de87ced2ac5a368a885bd82aa2ad8b8e6ef7befde72887167fb528b35285
    Image:          nginx
```

Observe that the Namespace is *default.*

- `kubectl describe po <podName> -n <nameSpace>` → describes specified pod belonging to the specified ns.



```
root@controlplane:~$ kubectl describe po sanskriti -n dev
Name:         sanskriti-68db8f8898-hjzxm
Namespace:    dev
Priority:     0
Node:         controlplane/172.30.1.2
Start Time:   Sat, 27 Aug 2022 10:58:38 +0000
Labels:       app=sanskriti
              pod-template-hash=68db8f8898
Annotations:  cni.projectcalico.org/containerID: 3f4f6fb7b0775b019ef1f18051ae64cf922097db86e71ce53aac41
              cni.projectcalico.org/podIP: 192.168.0.7/32
              cni.projectcalico.org/podIPs: 192.168.0.7/32
Status:       Running
IP:           192.168.0.7
IPs:
  IP:           192.168.0.7
Controlled By:  ReplicaSet/sanskriti-68db8f8898
Containers:
  nginx:
    Container ID:   containerd://b2a3fcce7c683f7821b32555030e72cd657e0d09ce92b95db046e85ae852533a
    Image:          nginx
```

Observe that the Namespace is *dev* , as specified.

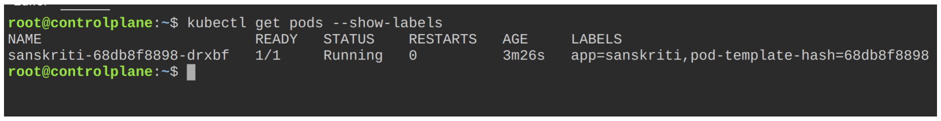
▼ Labels

- Labels → key-value pairs ; add meaning to your Kubernets object.
- Labels are defined in the metadata section.

▼ How to view label on a pod?

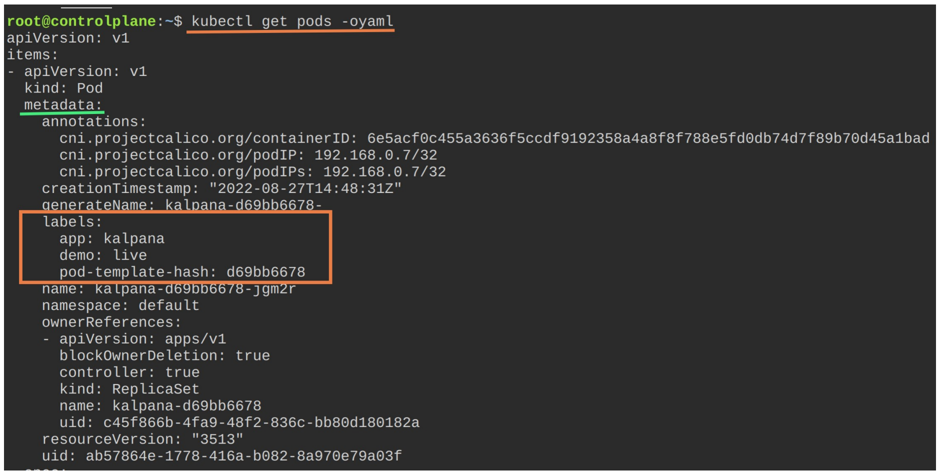
Two methods:

1. `kubectl get pod --show-labels`
    - Let's say, I want to view label on pod named *sanskriti-68db8f8898-drxbf* : `kubectl get pods --show-labels` . As you can see below, it has two labels.



```
root@controlplane:~$ kubectl get pods --show-labels
NAME                         READY   STATUS    RESTARTS   AGE     LABELS
sanskriti-68db8f8898-drxbf   1/1     Running   0          3m26s   app=sanskriti,pod-template-hash=68db8f8898
root@controlplane:~$ 
```

2. `kubectl get pod -oyaml`
    a. It displays all the pods belonging to current ns in the yaml format.
    b. As you can see below that the pod named *kalpana* has three labels assigned to it.

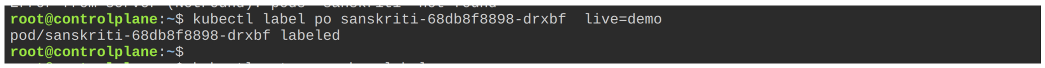


```
root@controlplane:~$ kubectl get pods -oyaml
apiVersion: v1
items:
- apiVersion: v1
  kind: Pod
  metadata:
    annotations:
      cni.projectcalico.org/containerID: 6e5acf0c455a3636f5ccdf9192358a4a8f8f788e5fd0db74d7f89b70d45a1bad
      cni.projectcalico.org/podIP: 192.168.0.7/32
      cni.projectcalico.org/podIPs: 192.168.0.7/32
    creationTimestamp: "2022-08-27T14:48:31Z"
    generateName: kalpana-d69bb6678-
    labels:
      app: kalpana
      demo: live
      pod-template-hash: d69bb6678
    name: kalpana-d69bb6678-jgm2r
    namespace: default
    ownerReferences:
    - apiVersion: apps/v1
      blockOwnerDeletion: true
      controller: true
      kind: ReplicaSet
      name: kalpana-d69bb6678
      uid: c45f866b-4fa9-48f2-836c-bb80d180182a
    resourceVersion: "3513"
    uid: ab57864e-1778-416a-b082-8a970e79a03f
```

▼ How to add label to a pod?

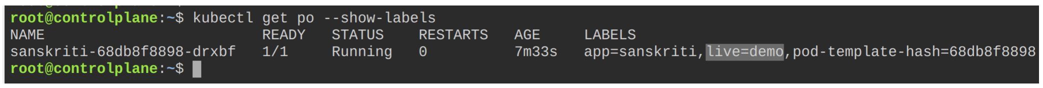
- `kubectl label pod <podName> key=value -n <nameSpace>` :
    - You need not add the `-n <nameSpace>` in the command in case the pod belongs to the current ns.

- Eg, I want to add label `live=demo` to pod *sanskriti-68db8f8898-drxbf.* For that I'll run `kubectl label pod sanskriti-68db8f8898-drxbf live=demo`



```
root@controlplane:~$ kubectl label po sanskriti-68db8f8898-drxbf  live=demo
pod/sanskriti-68db8f8898-drxbf labeled
root@controlplane:~$
```

- Now. check the labels to verify:



```
root@controlplane:~$ kubectl get po --show-labels
NAME                         READY   STATUS    RESTARTS   AGE     LABELS
sanskriti-68db8f8898-drxbf   1/1     Running   0          7m33s   app=sanskriti,live=demo,pod-template-hash=68db8f8898
root@controlplane:~$
```

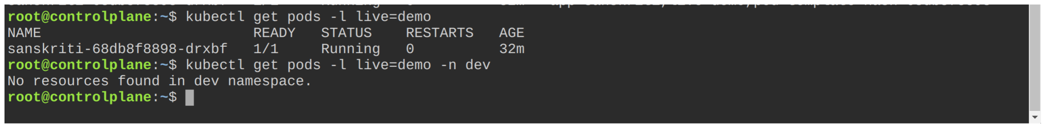
Label `live=demo` added.

Selectors

- Selecting on the basis of labels.
  - **Use case**: You can use selectors to divert your traffic on pods having same labels.

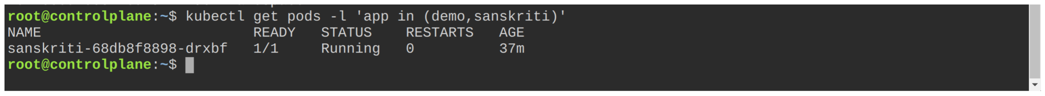
How to display pods having specific labels?

1. `kubectl get pods -l live=demo` : It will display all the pods in the current ns having label `live=demo` .



```
root@controlplane:~$ kubectl get pods -l live=demo
NAME                         READY   STATUS    RESTARTS   AGE
sanskriti-68db8f8898-drxbf   1/1     Running   0          32m
root@controlplane:~$ kubectl get pods -l live=demo -n dev
No resources found in dev namespace.
root@controlplane:~$
```

2. `kubectl get pods -l 'app in (demo,sanskriti)'` → display pod named *sanskriti* having label value *demo* present in the current ns.



```
root@controlplane:~$ kubectl get pods -l 'app in (demo,sanskriti)'
NAME                         READY   STATUS    RESTARTS   AGE
sanskriti-68db8f8898-drxbf   1/1     Running   0          37m
root@controlplane:~$
```