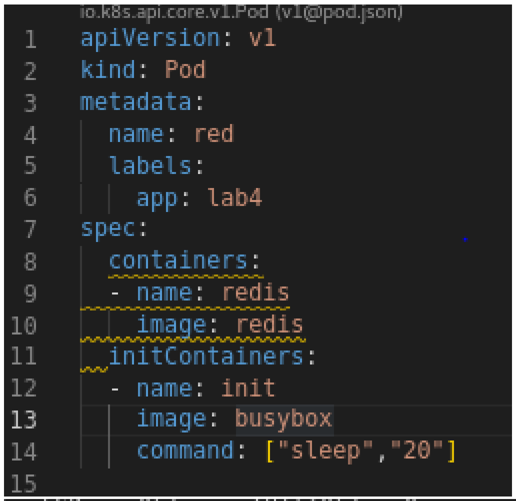
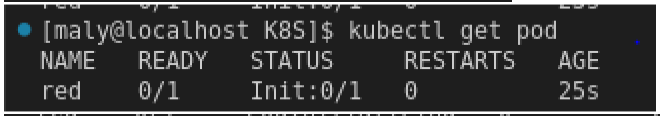
LAB4

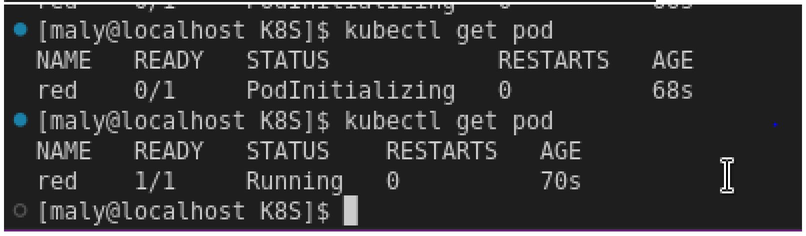
1- Create a pod red with redis image and use an initContainer that uses the busybox image and sleeps for 20 seconds



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: red
  labels:
    app: lab4
spec:
  containers:
  - name: redis
    image: redis
  initContainers:
  - name: init
    image: busybox
    command: ["sleep","20"]
```



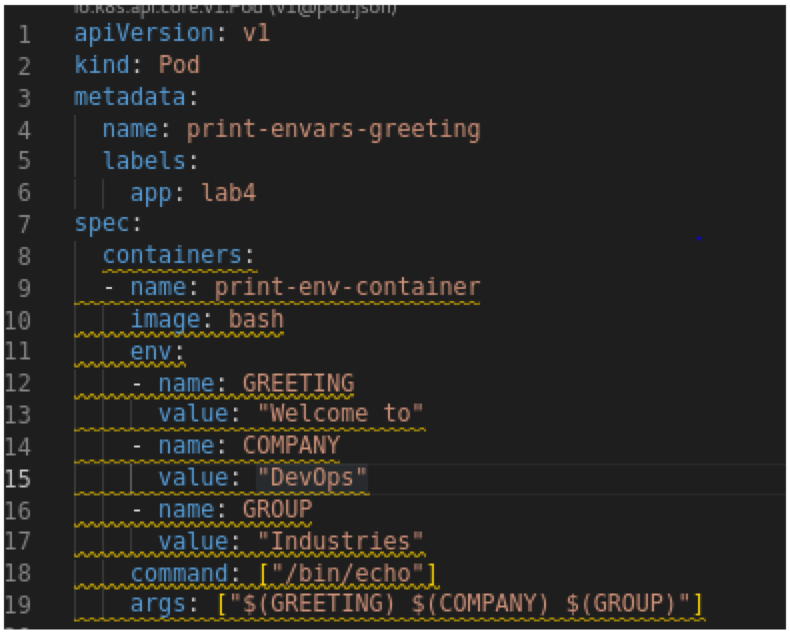
```
[maly@localhost K8S]$ kubectl get pod
NAME   READY   STATUS     RESTARTS   AGE
red    0/1     Init:0/1   0          25s
```



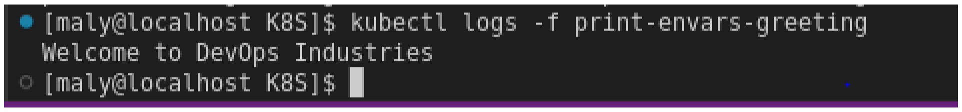
```
[maly@localhost K8S]$ kubectl get pod
NAME   READY   STATUS            RESTARTS   AGE
red    0/1     PodInitializing   0          68s
[maly@localhost K8S]$ kubectl get pod
NAME   READY   STATUS    RESTARTS   AGE
red    1/1     Running   0          70s
[maly@localhost K8S]$
```

2- Create a pod named print-envars-greeting.

1. Configure spec as, the container name should be print-env-container and use bash image.
2. Create three environment variables:
    a. GREETING and its value should be “Welcome to”
    b. COMPANY and its value should be “DevOps”
    c. GROUP and its value should be “Industries”
3. Use command to echo ["$(GREETING) $(COMPANY) $(GROUP)"] message.
4. You can check the output using <kubctl logs -f [pod-name]> command

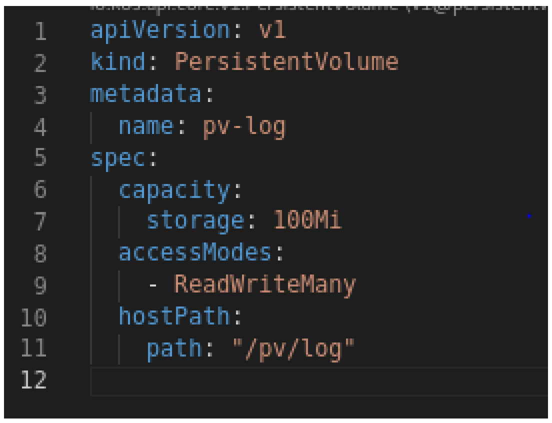


```yaml
apiVersion: v1
kind: Pod
metadata:
  name: print-envars-greeting
  labels:
    app: lab4
spec:
  containers:
  - name: print-env-container
    image: bash
    env:
    - name: GREETING
      value: "Welcome to"
    - name: COMPANY
      value: "DevOps"
    - name: GROUP
      value: "Industries"
    command: ["/bin/echo"]
    args: ["$(GREETING) $(COMPANY) $(GROUP)"]
```

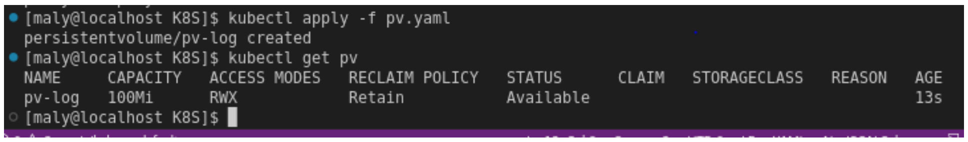


```
[maly@localhost K8S]$ kubectl logs -f print-envars-greeting
Welcome to DevOps Industries
[maly@localhost K8S]$
```

3- Create a Persistent Volume with the given specification.
Volume Name: pv-log
Storage: 100Mi
Access Modes: ReadWriteMany
Host Path: /pv/log

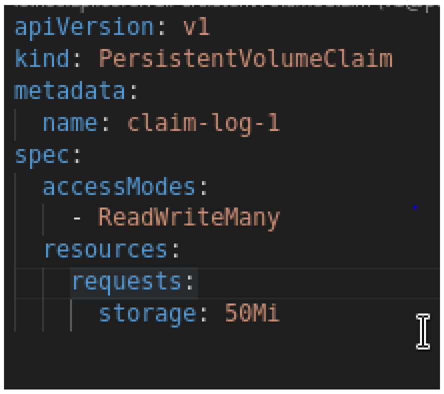


```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-log
spec:
  capacity:
    storage: 100Mi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/pv/log"
```

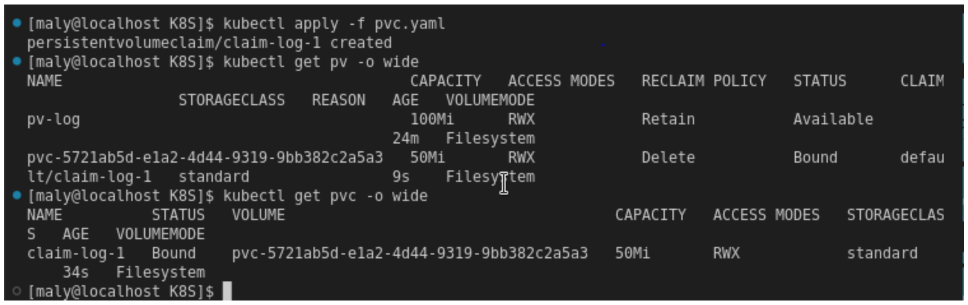


```
[maly@localhost K8S]$ kubectl apply -f pv.yaml
persistentvolume/pv-log created
[maly@localhost K8S]$ kubectl get pv
NAME     CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
pv-log   100Mi      RWX            Retain           Available                                   13s
[maly@localhost K8S]$
```

4- Create a Persistent Volume Claim with the given specification.
Volume Name: claim-log-1
Storage Request: 50Mi
Access Modes: ReadWriteMany

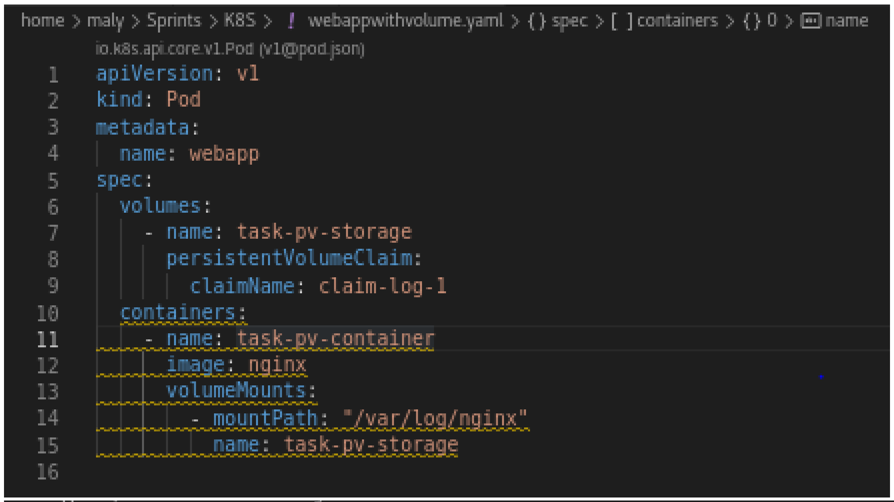


```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: claim-log-1
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 50Mi
```



```
[maly@localhost K8S]$ kubectl apply -f pvc.yaml
persistentvolumeclaim/claim-log-1 created
[maly@localhost K8S]$ kubectl get pv -o wide
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM
                 STORAGECLASS   REASON   AGE   VOLUMEMODE
pv-log                                     100Mi      RWX            Retain           Available
                                         24m   Filesystem
pvc-5721ab5d-e1a2-4d44-9319-9bb382c2a5a3   50Mi       RWX            Delete           Bound       defau
lt/claim-log-1   standard                9s    Filesystem
[maly@localhost K8S]$ kubectl get pvc -o wide
NAME          STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLAS
S   AGE   VOLUMEMODE
claim-log-1   Bound    pvc-5721ab5d-e1a2-4d44-9319-9bb382c2a5a3   50Mi       RWX            standard
    34s   Filesystem
[maly@localhost K8S]$
```

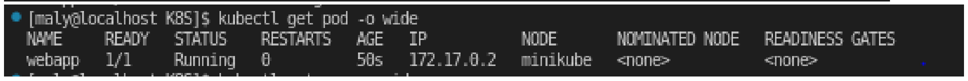
5- Create a webapp pod to use the persistent volume claim as its storage.
Name: webapp
Image Name: nginx
Volume: PersistentVolumeClaim=claim-log-1
Volume Mount: /var/log/nginx



home > maly > Sprints > K8S > ! webappwithvolume.yaml > {} spec > [ ] containers > {} 0 > name

io.k8s.api.core.v1.Pod (v1@pod.json)

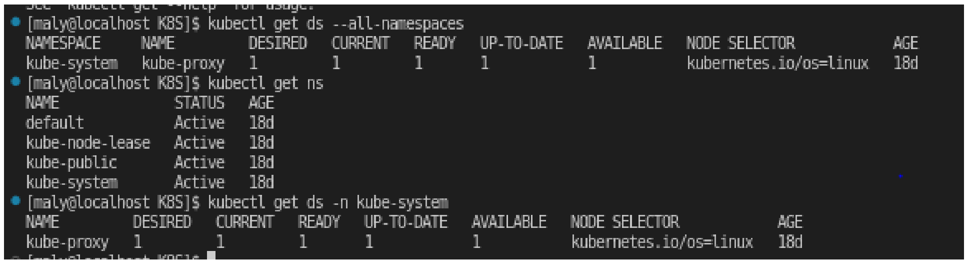
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: webapp
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
        claimName: claim-log-1
  containers:
    - name: task-pv-container
      image: nginx
      volumeMounts:
        - mountPath: "/var/log/nginx"
          name: task-pv-storage

```



```
[maly@localhost K8S]$ kubectl get pod -o wide
NAME     READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
webapp   1/1     Running   0          50s   172.17.0.2   minikube   <none>           <none>
```

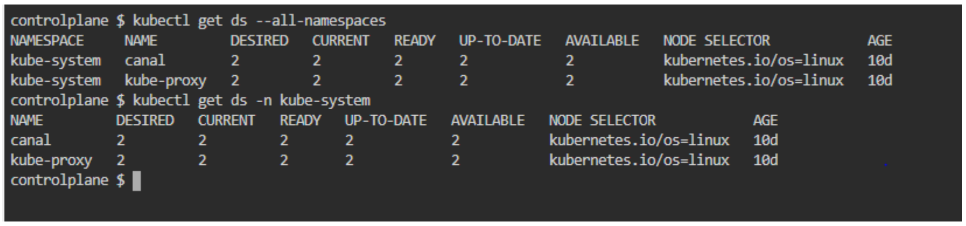
6- How many DaemonSets are created in the cluster in all namespaces?
7- what DaemonSets exist on the kube-system namespace?

VM:



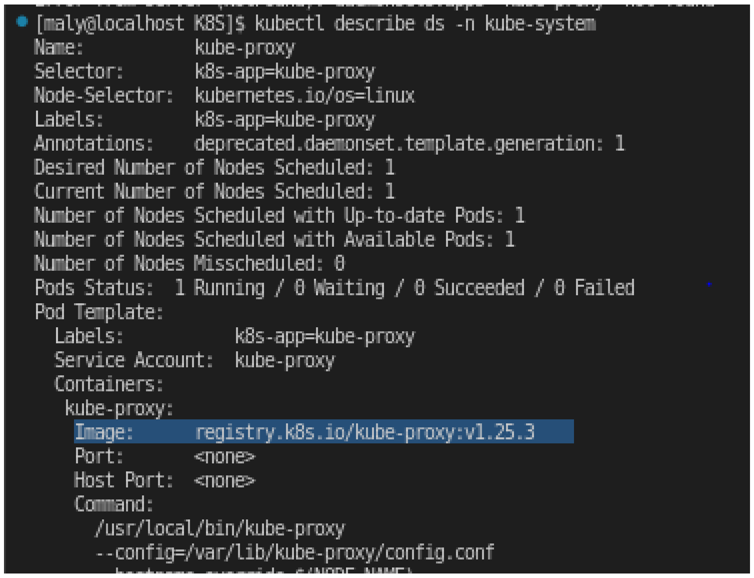
```
[maly@localhost K8S]$ kubectl get ds --all-namespaces
NAMESPACE     NAME         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
kube-system   kube-proxy   1         1         1       1            1           kubernetes.io/os=linux   18d
[maly@localhost K8S]$ kubectl get ns
NAME              STATUS   AGE
default           Active   18d
kube-node-lease   Active   18d
kube-public       Active   18d
kube-system       Active   18d
[maly@localhost K8S]$ kubectl get ds -n kube-system
NAME         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
kube-proxy   1         1         1       1            1           kubernetes.io/os=linux   18d
```

Killer-Coda:



```
controlplane $ kubectl get ds --all-namespaces
NAMESPACE     NAME         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
kube-system   canal        2         2         2       2            2           kubernetes.io/os=linux   10d
kube-system   kube-proxy   2         2         2       2            2           kubernetes.io/os=linux   10d
controlplane $ kubectl get ds -n kube-system
NAME         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
canal        2         2         2       2            2           kubernetes.io/os=linux   10d
kube-proxy   2         2         2       2            2           kubernetes.io/os=linux   10d
controlplane $ 
```

8- What is the image used by the POD deployed by the kube-proxy DaemonSet



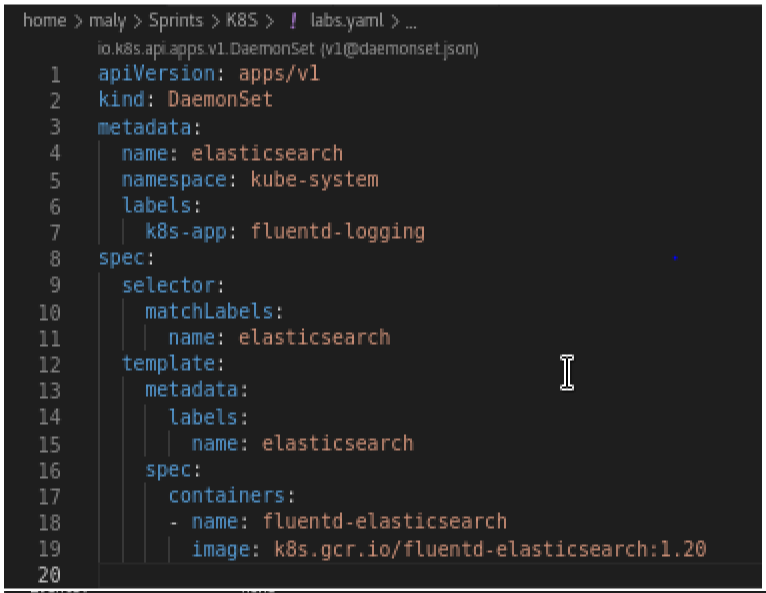
```
[maly@localhost K8S]$ kubectl describe ds -n kube-system
Name:           kube-proxy
Selector:       k8s-app=kube-proxy
Node-Selector:  kubernetes.io/os=linux
Labels:         k8s-app=kube-proxy
Annotations:    deprecated.daemonset.template.generation: 1
Desired Number of Nodes Scheduled: 1
Current Number of Nodes Scheduled: 1
Number of Nodes Scheduled with Up-to-date Pods: 1
Number of Nodes Scheduled with Available Pods: 1
Number of Nodes Misscheduled: 0
Pods Status:  1 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:           k8s-app=kube-proxy
  Service Account:  kube-proxy
  Containers:
   kube-proxy:
    Image:      registry.k8s.io/kube-proxy:v1.25.3
    Port:       <none>
    Host Port:  <none>
    Command:
      /usr/local/bin/kube-proxy
      --config=/var/lib/kube-proxy/config.conf
```

9- Deploy a DaemonSet for FluentD Logging, Use the given specifications.

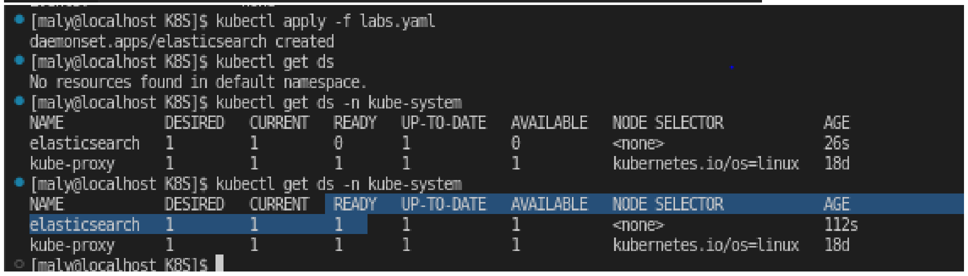
Name: elasticsearch

Namespace: kube-system

Image: k8s.gcr.io/fluentd-elasticsearch:1.20

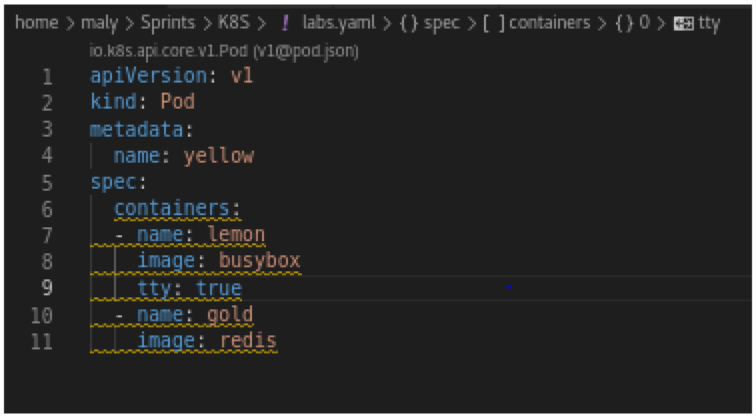


```
home > maly > Sprints > K8S > ! labs.yaml > ...
io.k8s.api.apps.v1.DaemonSet (v1@daemonset.json)
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: elasticsearch
  namespace: kube-system
  labels:
    k8s-app: fluentd-logging
spec:
  selector:
    matchLabels:
      name: elasticsearch
  template:
    metadata:
      labels:
        name: elasticsearch
    spec:
      containers:
      - name: fluentd-elasticsearch
        image: k8s.gcr.io/fluentd-elasticsearch:1.20

```

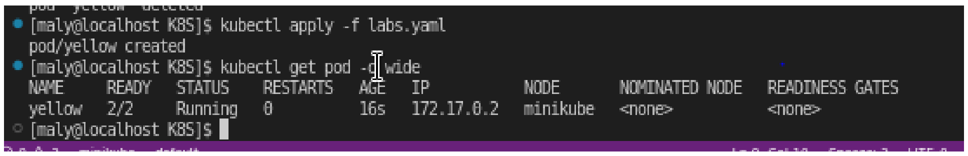


```
[maly@localhost K8S]$ kubectl apply -f labs.yaml
daemonset.apps/elasticsearch created
[maly@localhost K8S]$ kubectl get ds
No resources found in default namespace.
[maly@localhost K8S]$ kubectl get ds -n kube-system
NAME            DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
elasticsearch   1         1         0       1            0           <none>                   26s
kube-proxy      1         1         1       1            1           kubernetes.io/os=linux   18d
[maly@localhost K8S]$ kubectl get ds -n kube-system
NAME            DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
elasticsearch   1         1         1       1            1           <none>                   112s
kube-proxy      1         1         1       1            1           kubernetes.io/os=linux   18d
[maly@localhost K8S]$
```

10- Create a multi-container pod with 2 containers.
Name: yellow
Container 1 Name: lemon
Container 1 Image: busybox
Container 2 Name: gold
Container 2 Image: redis



```
home > maly > Sprints > K8S > ! labs.yaml > {} spec > [ ] containers > {} 0 > tty
io.k8s.api.core.v1.Pod (v1@pod.json)
apiVersion: v1
kind: Pod
metadata:
  name: yellow
spec:
  containers:
  - name: lemon
    image: busybox
    tty: true
  - name: gold
    image: redis
```



```
pod "yellow" deleted
[maly@localhost K8S]$ kubectl apply -f labs.yaml
pod/yellow created
[maly@localhost K8S]$ kubectl get pod -o wide
NAME     READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
yellow   2/2     Running   0          16s   172.17.0.2   minikube   <none>           <none>
[maly@localhost K8S]$
```