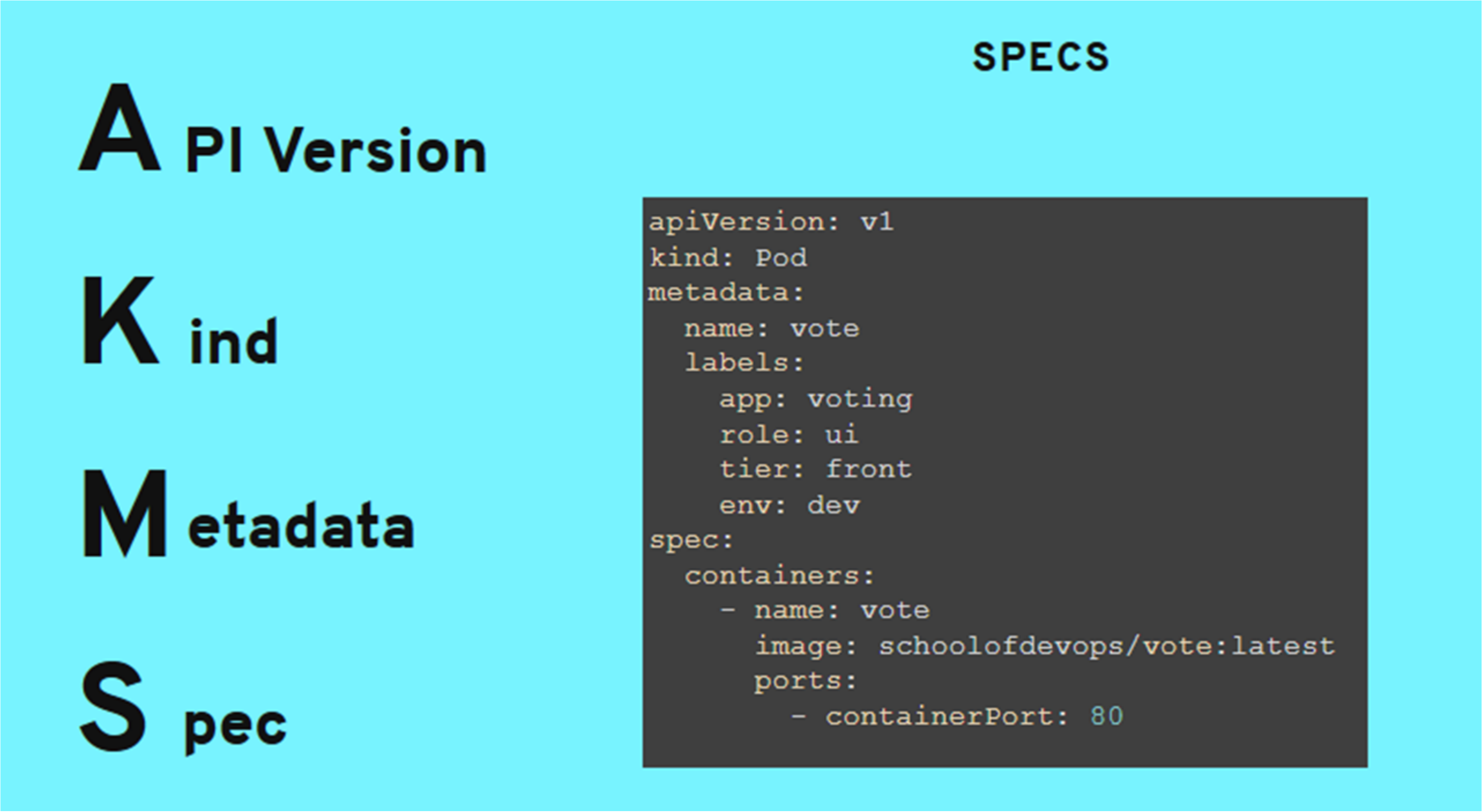
Pod

AKMS



SPECS

A PI Version

K ind

M etadata

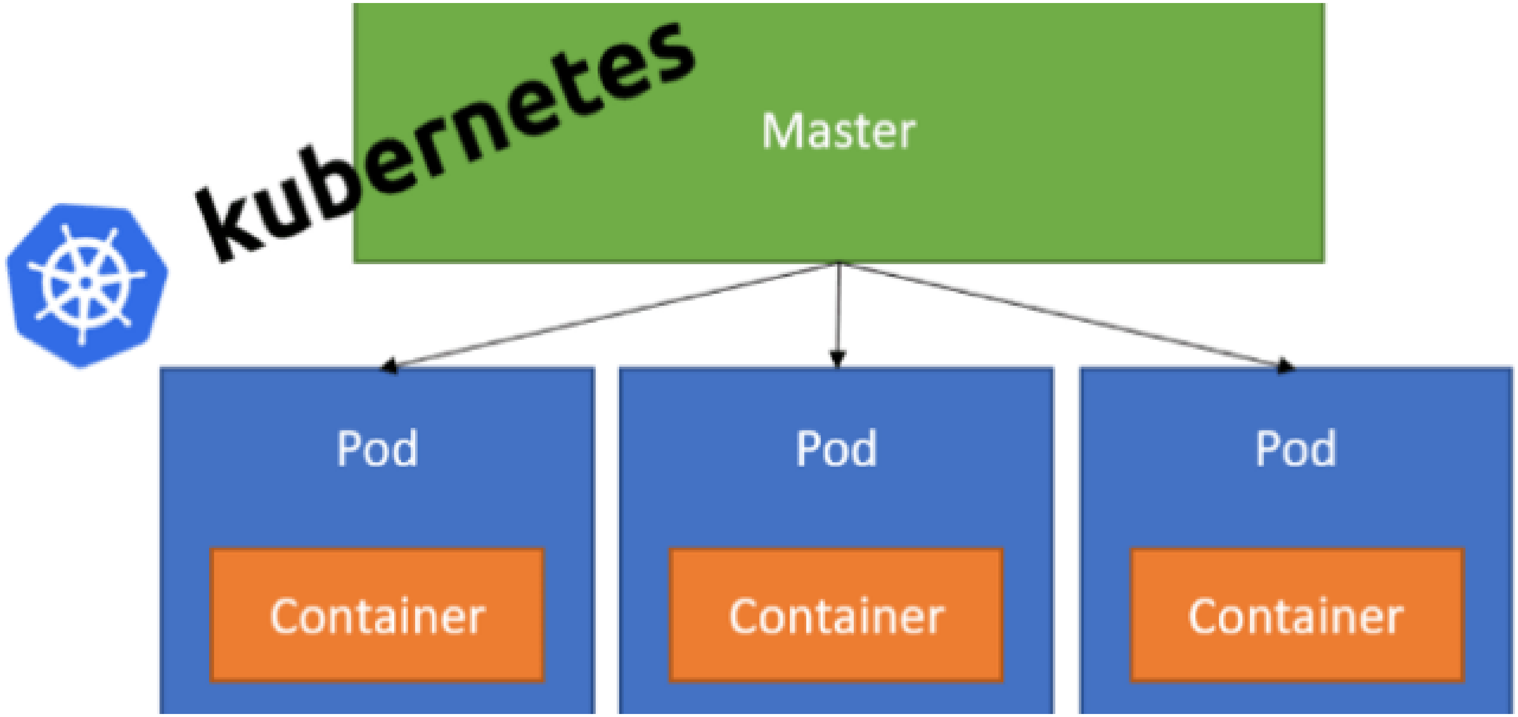
S pec

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: vote
  labels:
    app: voting
    role: ui
    tier: front
    env: dev
spec:
  containers:
    - name: vote
      image: schoolofdevops/vote:latest
      ports:
        - containerPort: 80
```

..

- apiVersion — This defines the Kubernetes API version that we want to use in this YAML file. You can read more about API versioning in Kubernetes here.
- kind — This defines what kind of Kubernetes object we want to create.
- metadata — This is data that helps us uniquely identify the object that we want to create. Here we can provide a name for our app, as well as apply labels to our object.
- spec — This defines the state that we want or our object. The format that we use for spec.

POD


kubernetes
Master
Pod
Container
Pod
Container
Pod
Container

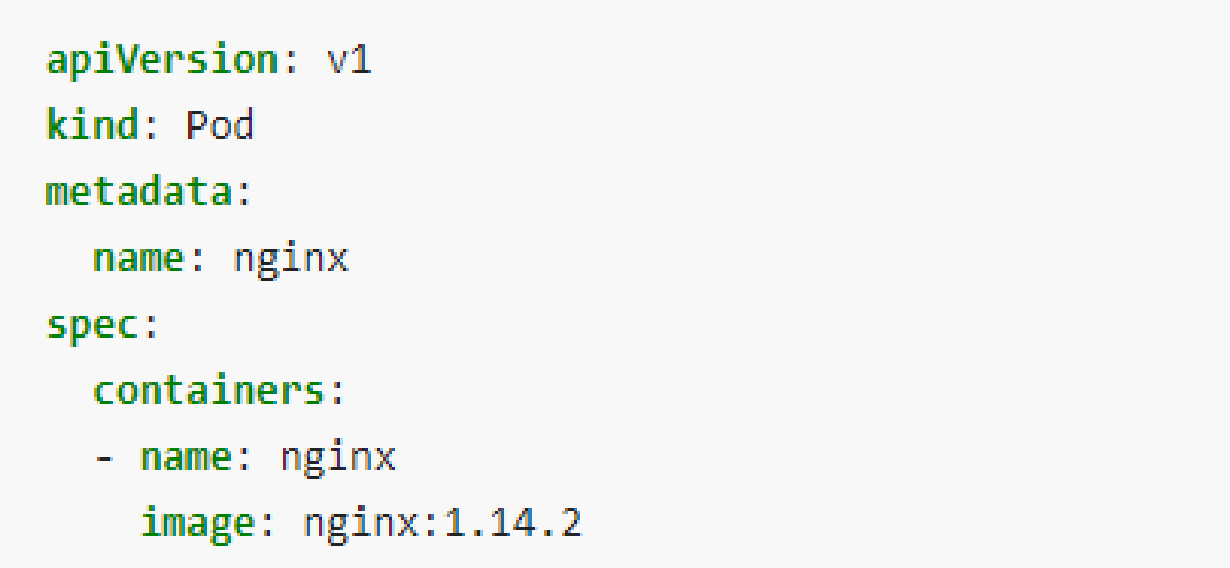
Pods

- Pods are the smallest deployable units of computing that you can create and manage in Kubernetes.
- A Pod (is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers.
- While Kubernetes supports more container runtimes than just Docker, Docker is the most commonly known runtime
  - containerd
  - CRI-O
  - Docker Engine

Pods type

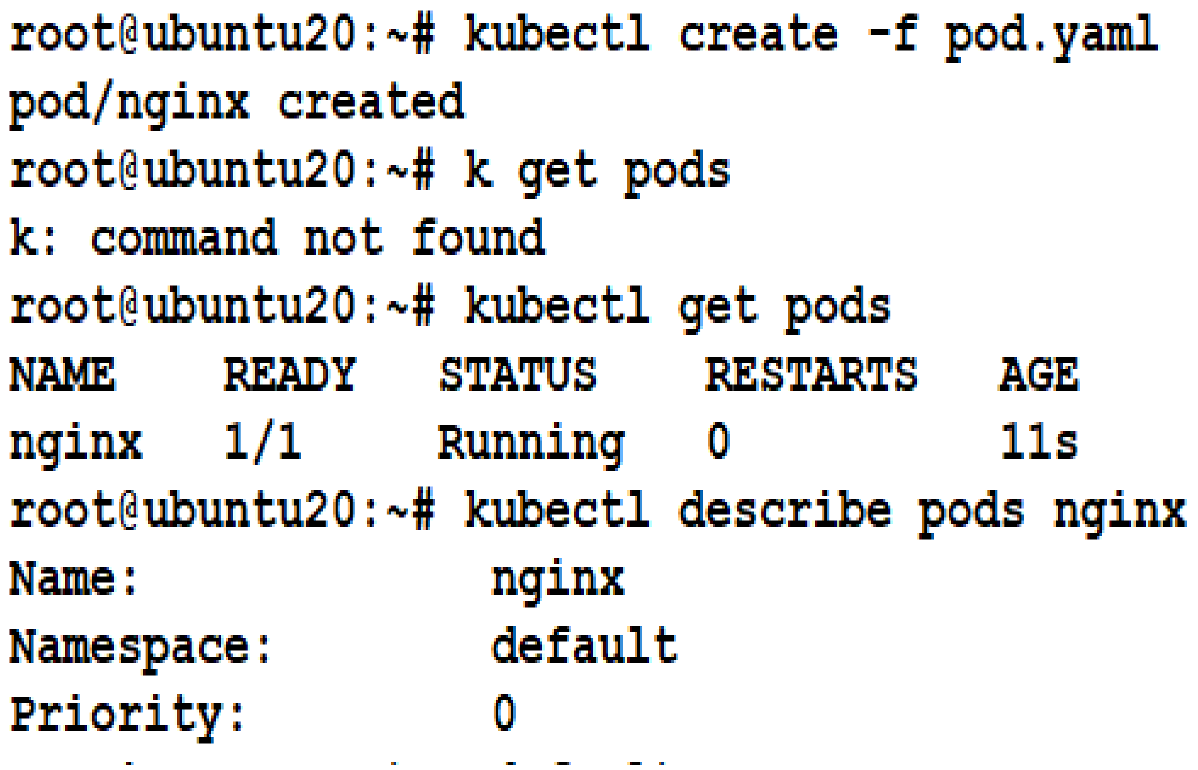
- Pods in a Kubernetes cluster are used in two main ways:
  - **Pods that run a single container**. The "one-container-per-Pod" model is the most common Kubernetes use case; in this case, you can think of a Pod as a wrapper around a single container; Kubernetes manages Pods rather than managing the containers directly.
  - **Pods that run multiple containers that need to work together**. A Pod can encapsulate an application composed of multiple co-located containers that are tightly coupled and need to share resources. The Pod wraps these containers, storage resources, and an ephemeral network identity together as a single unit.

Pod yaml file



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
```

..

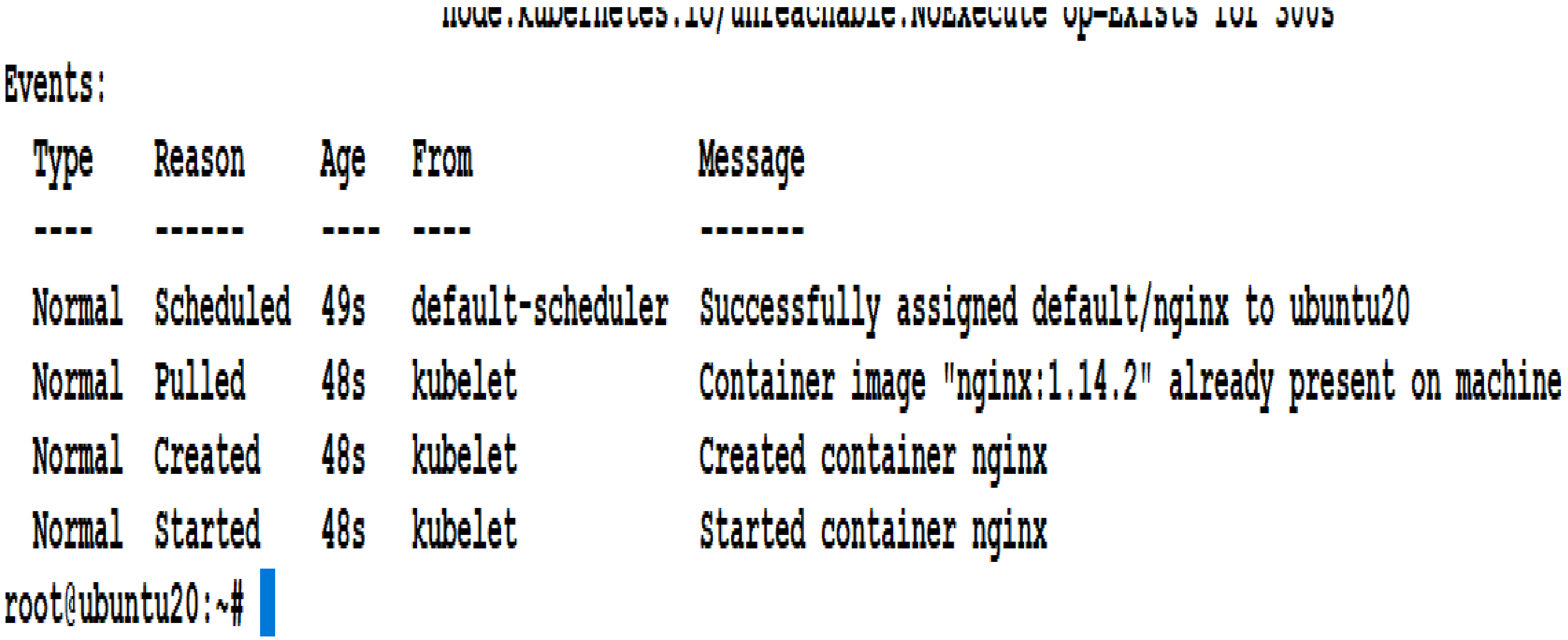


```
root@ubuntu20:~# kubectl create -f pod.yaml
pod/nginx created
root@ubuntu20:~# k get pods
k: command not found
root@ubuntu20:~# kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          11s
root@ubuntu20:~# kubectl describe pods nginx
Name:              nginx
Namespace:         default
Priority:          0
```

kubectl cmds

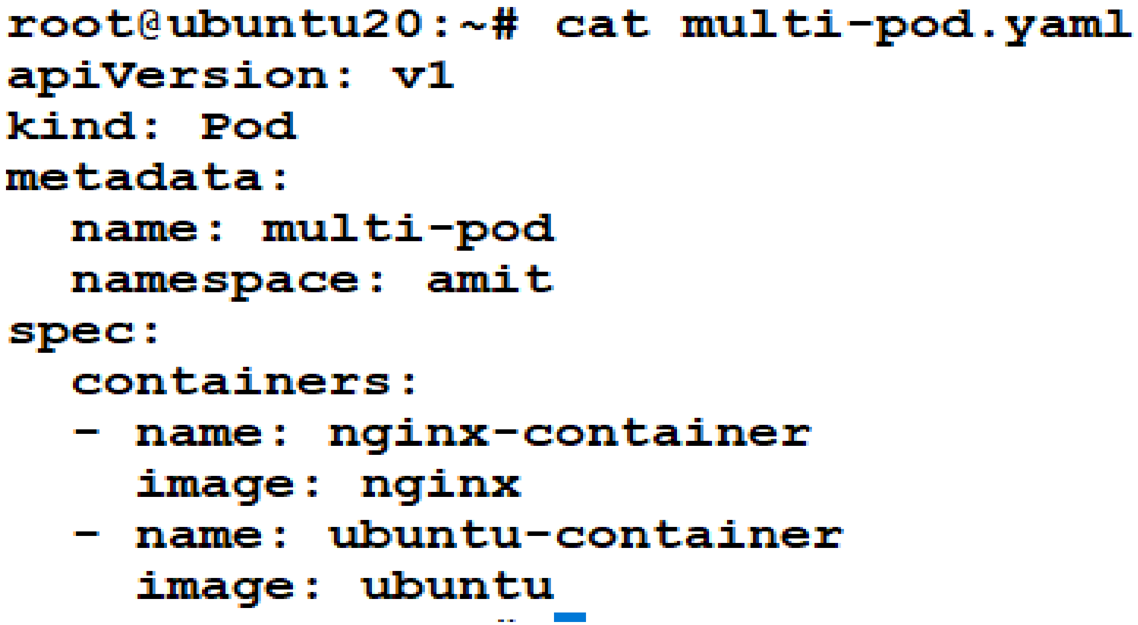
- kubectl create –f <pod yaml>
- kubectl get pods
- kubectl describe pods nginx
- kubectl get pod nginx -o wide
- kubectl delete pod nginx

Events during pod creating



```
                    node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  49s   default-scheduler  Successfully assigned default/nginx to ubuntu20
  Normal  Pulled     48s   kubelet            Container image "nginx:1.14.2" already present on machine
  Normal  Created    48s   kubelet            Created container nginx
  Normal  Started    48s   kubelet            Started container nginx
root@ubuntu20:~#
```

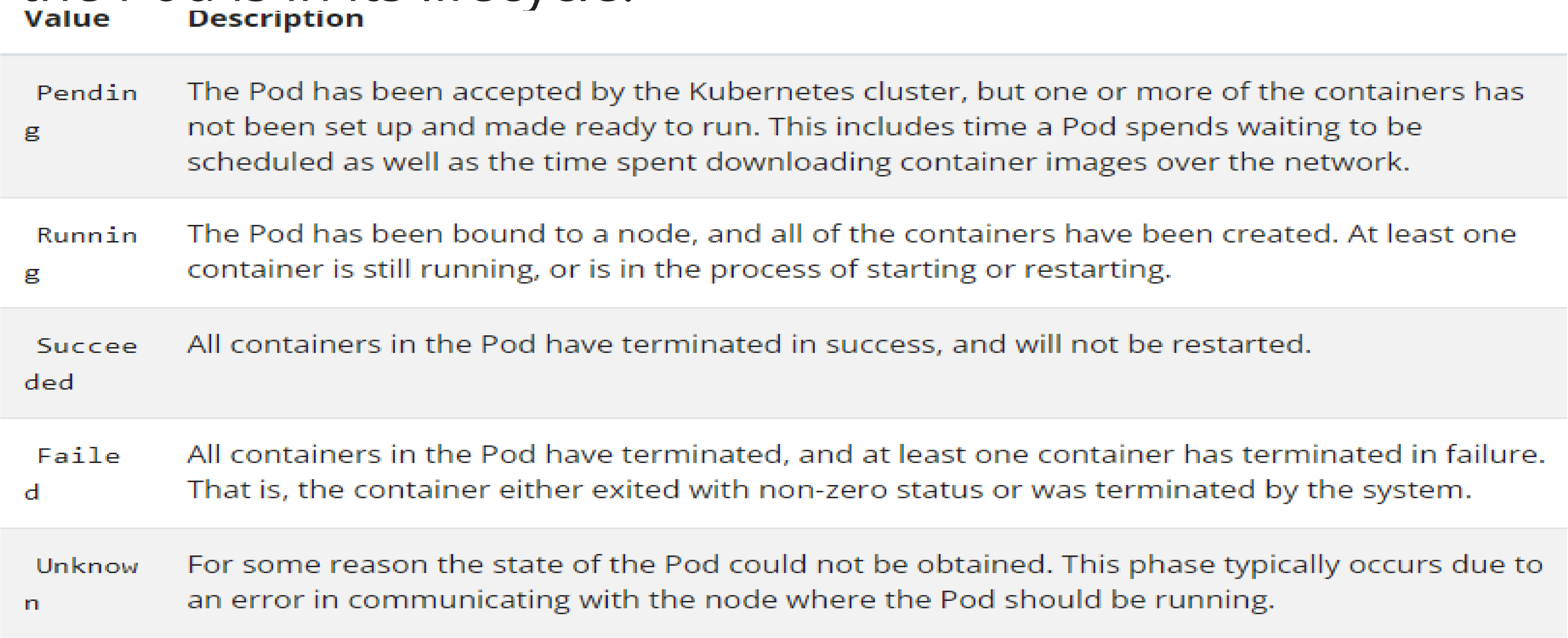
Multiple container pod



```
root@ubuntu20:~# cat multi-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: multi-pod
  namespace: amit
spec:
  containers:
  - name: nginx-container
    image: nginx
  - name: ubuntu-container
    image: ubuntu
```

Pod lifecycle

- The phase of a Pod is a simple, high-level summary of where the Pod is in its lifecycle.



Value	Description
`Pending`	The Pod has been accepted by the Kubernetes cluster, but one or more of the containers has not been set up and made ready to run. This includes time a Pod spends waiting to be scheduled as well as the time spent downloading container images over the network.
`Running`	The Pod has been bound to a node, and all of the containers have been created. At least one container is still running, or is in the process of starting or restarting.
`Succeeded`	All containers in the Pod have terminated in success, and will not be restarted.
`Failed`	All containers in the Pod have terminated, and at least one container has terminated in failure. That is, the container either exited with non-zero status or was terminated by the system.
`Unknown`	For some reason the state of the Pod could not be obtained. This phase typically occurs due to an error in communicating with the node where the Pod should be running.