Kubernetes Concepts - Pod

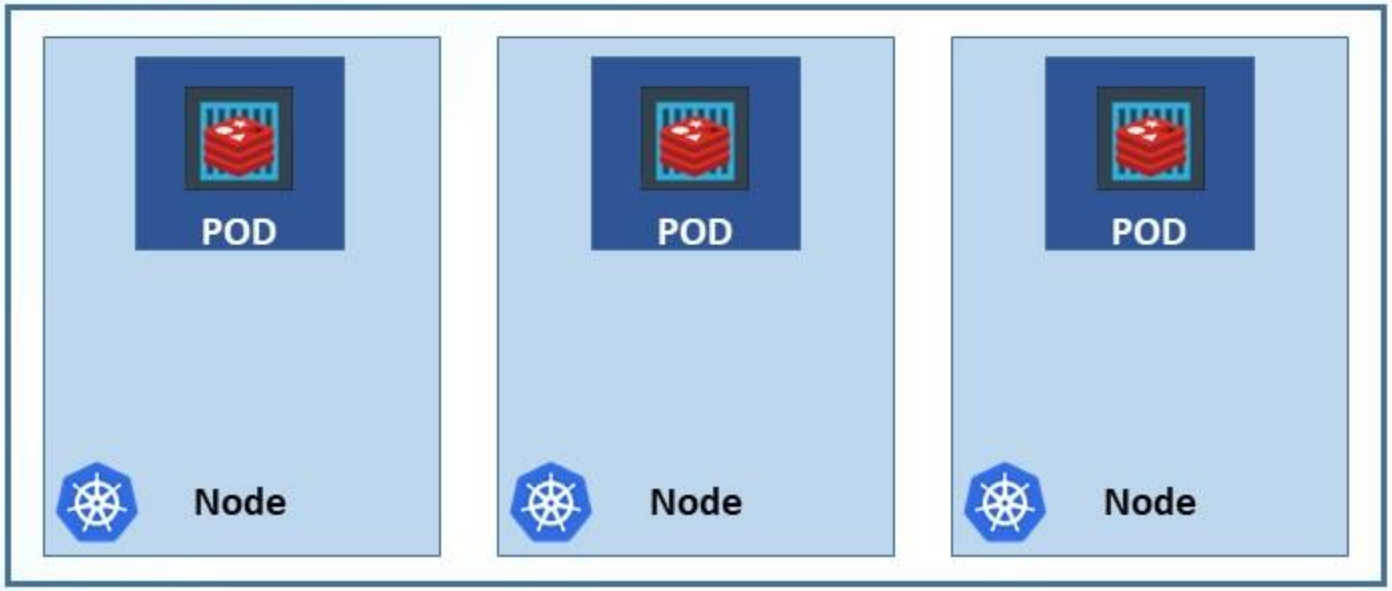
Assumptions

Docker Image

Kubernetes Cluster

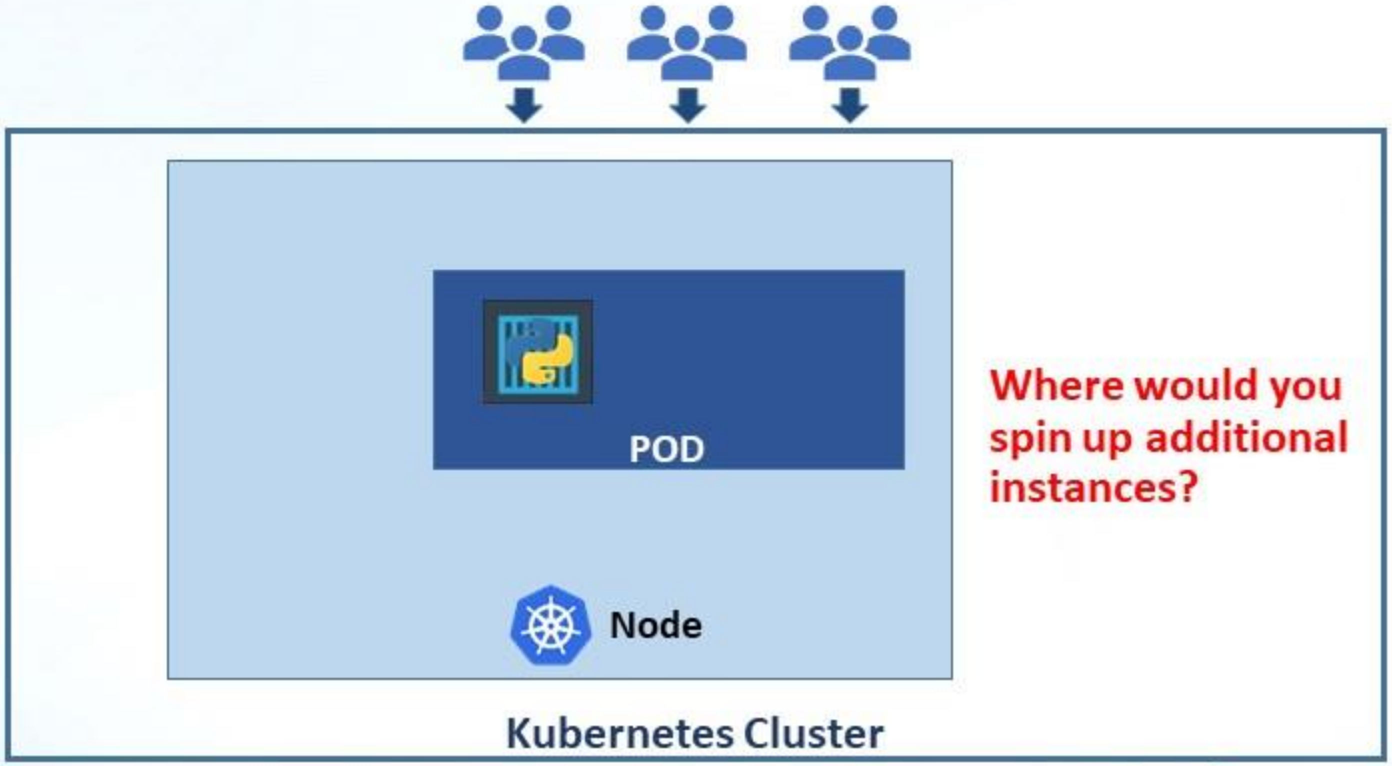
POD

- Kubernetes ultimate aim is to deploy application in the form of containers on a set of machines that are configured as worker nodes in a cluster.
- Kubernetes doses not deployed containers directly on the worker nodes
- The containers are encapsulated into a kubernetes object known as POD.
- A POD is a single instance of an application.
- It is the smallest object that you can create in Kubernetes.



POD

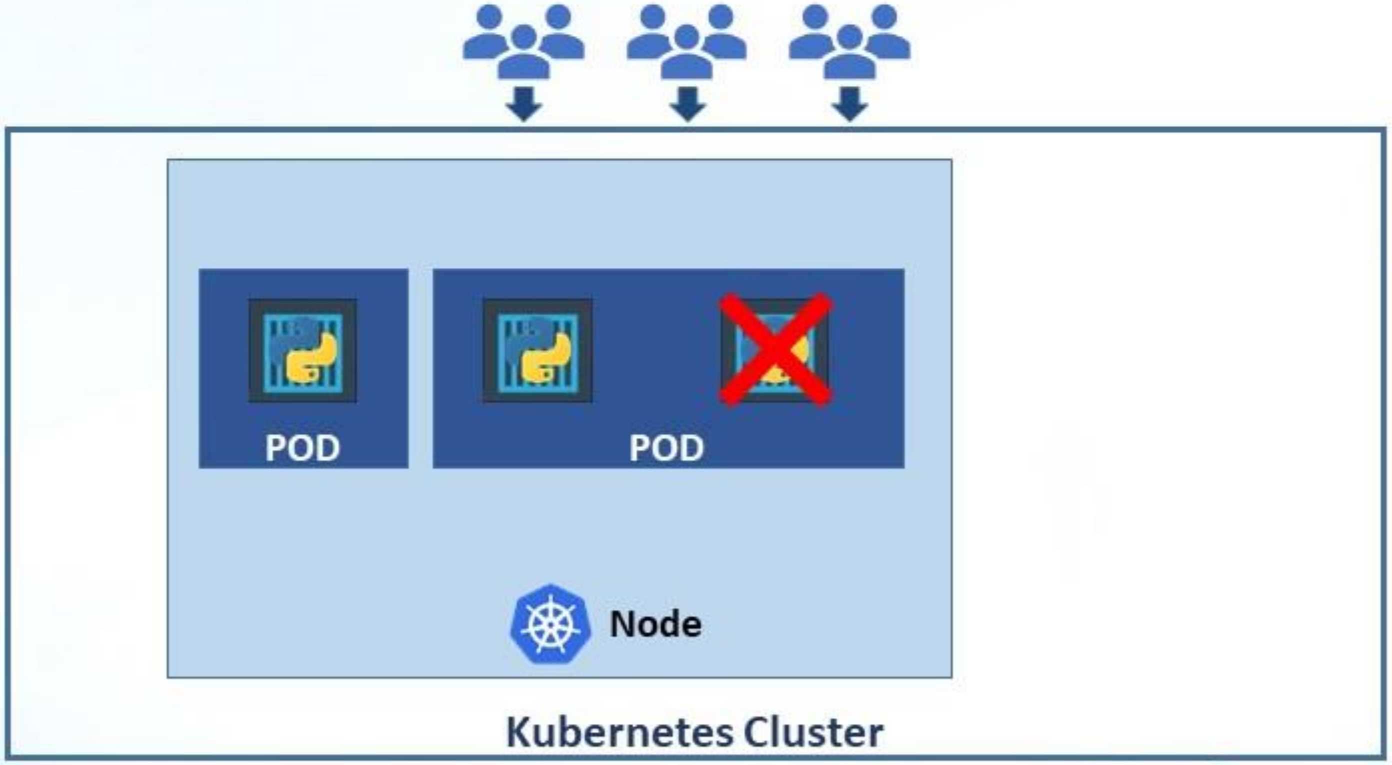
- This is the simplest of simplest cases where you have a single node Kubernetes cluster with a single instance of your application running in a single docker container encapsulated in a pod.
- When the number of users accessing our application increases, we need to scale our application
- We need to add additional instances of our web application to share the load



SkillAssure

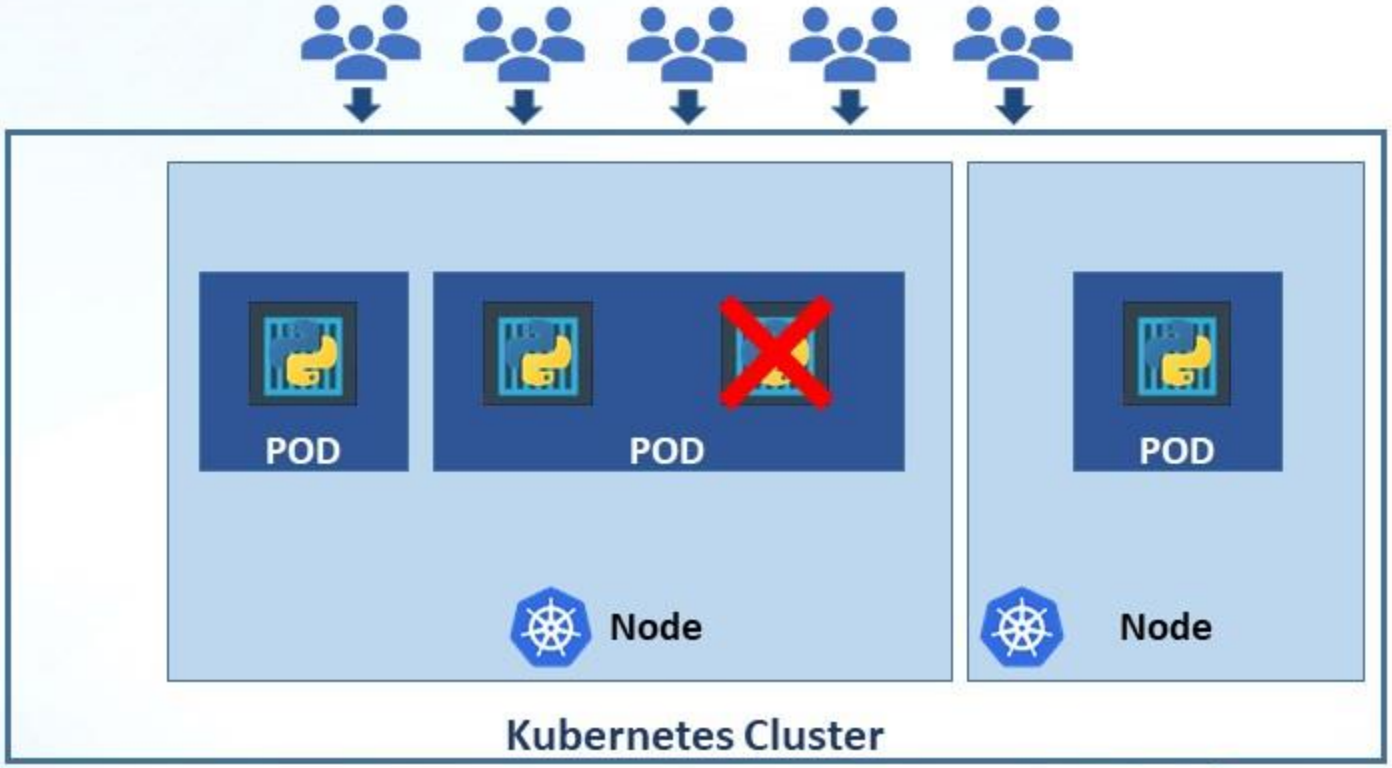
POD

- Do we bring up new container instance within the same pod?
- NO
- **We create new POD with a new instance of the same application**



POD

- What if the user base further increases and your current node has no sufficient capacity
- **Deploy additional pods on a new node in the cluster.**
- A new node is added to the cluster to expand the clusters Physical capacity.

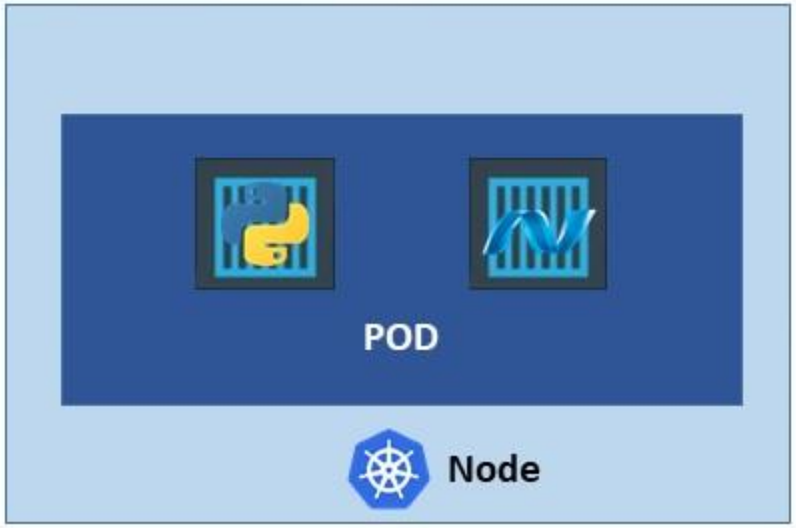


POD

- Pods usually have a one to one relationship with containers running your application.
- To scale up you create new pods.
- To scale down you delete existing pods.
- You do not add additional containers to any existing POD to scale your application.

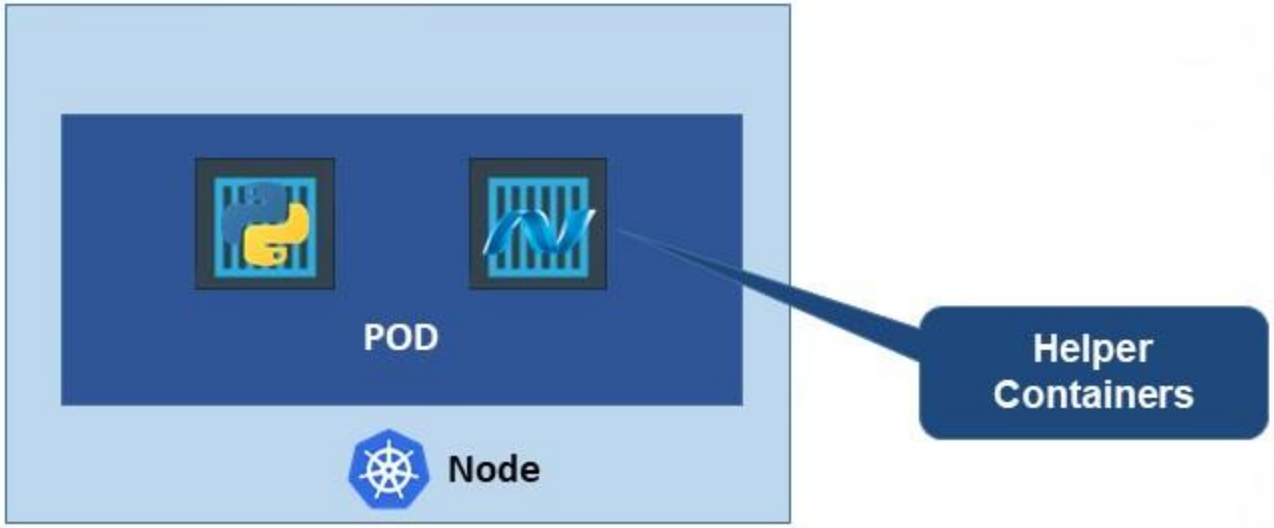
POD

- Are we restricted to have a single container in a pod?
- **No**
- Single pod can have multiple containers except for the fact that they are usually not multiple containers of the same kind.

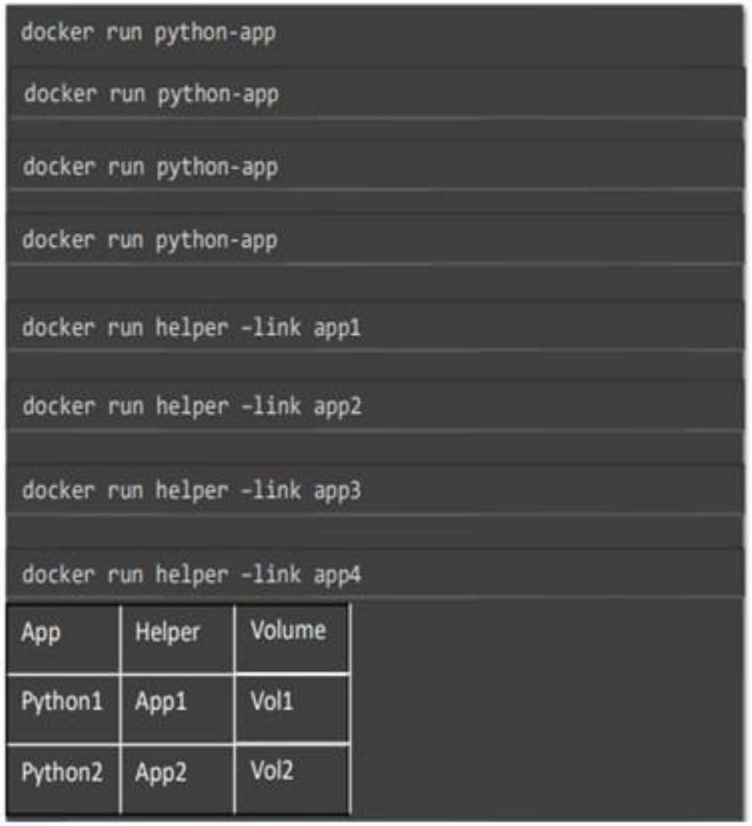


POD

- Sometimes we may have a scenario of having a helper container alongside our application container that might be doing supporting task for our web application
  - e.g. processing the user entered data, processing a file uploaded by the user, etc.
- They share same
  - Life
  - Network
  - Storage

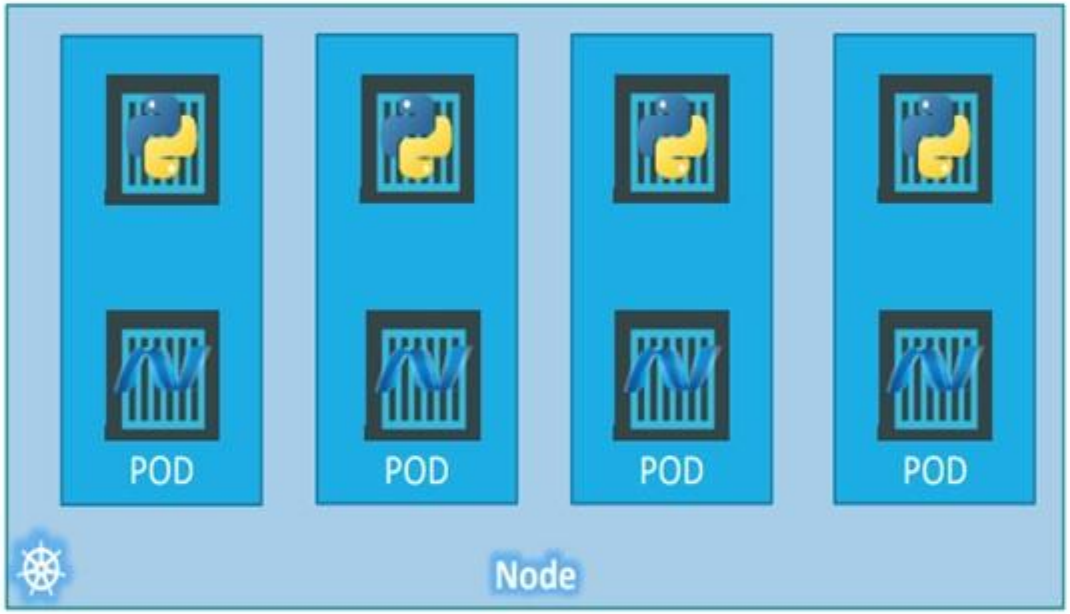


PODs Again!



```
docker run python-app
docker run python-app
docker run python-app
docker run python-app
docker run helper -link app1
docker run helper -link app2
docker run helper -link app3
docker run helper -link app4
```

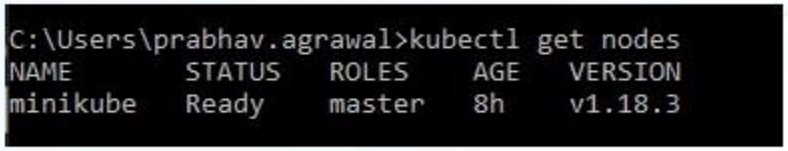
App	Helper	Volume
Python1	App1	Vol1
Python2	App2	Vol2



Note: I am avoiding networking and load balancing details to keep explanation simple.

kubectl

- List all Nodes
- **kubectl get nodes**

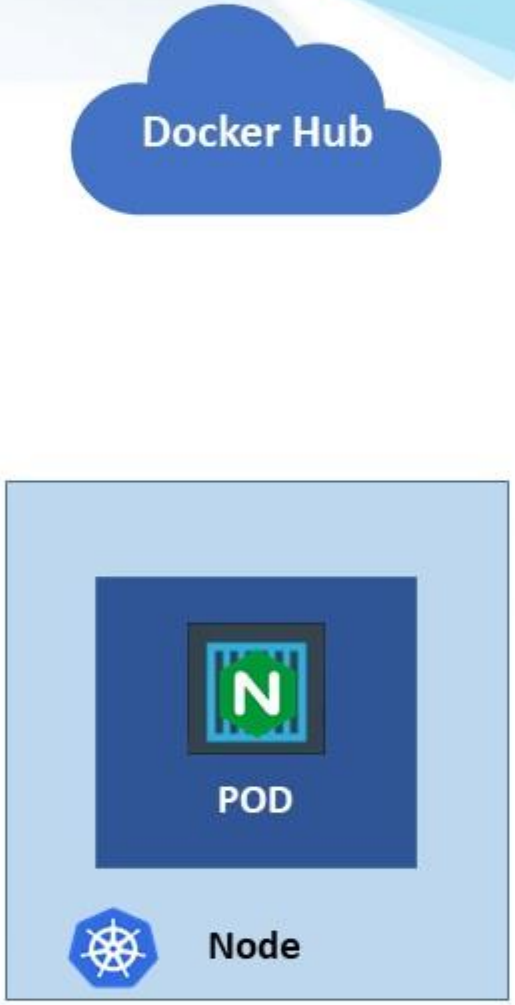


```
C:\Users\prabhav.agrawal>kubectl get nodes
NAME       STATUS   ROLES    AGE   VERSION
minikube   Ready    master   8h    v1.18.3
```

kubectl

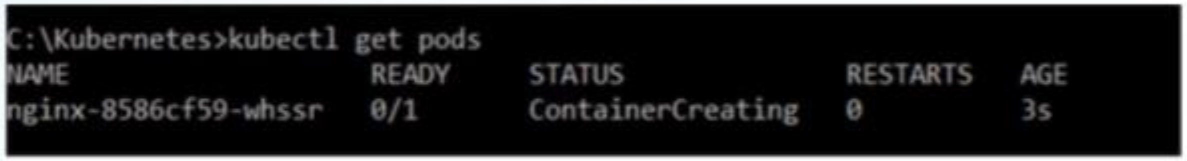
```
kubectl run nginx --image=nginx
```

- It deploys a docker container by creating a POD
- First it creates a POD automatically.
- Then deploys an instance of the nginx docker image
- Where does it get the application image from?
- The application image in this case the nginx image is downloaded from the Docker hub
- We can configure Kubernetes to pull the image from the public Docker hub or a private repository within the organization.

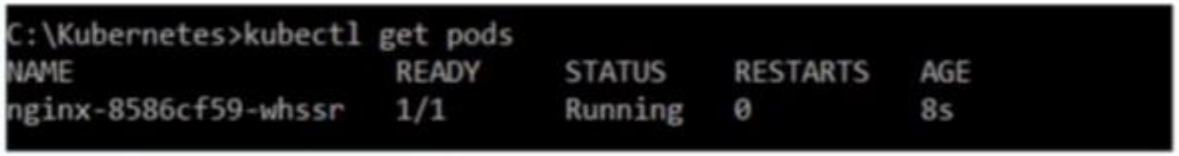


kubectl

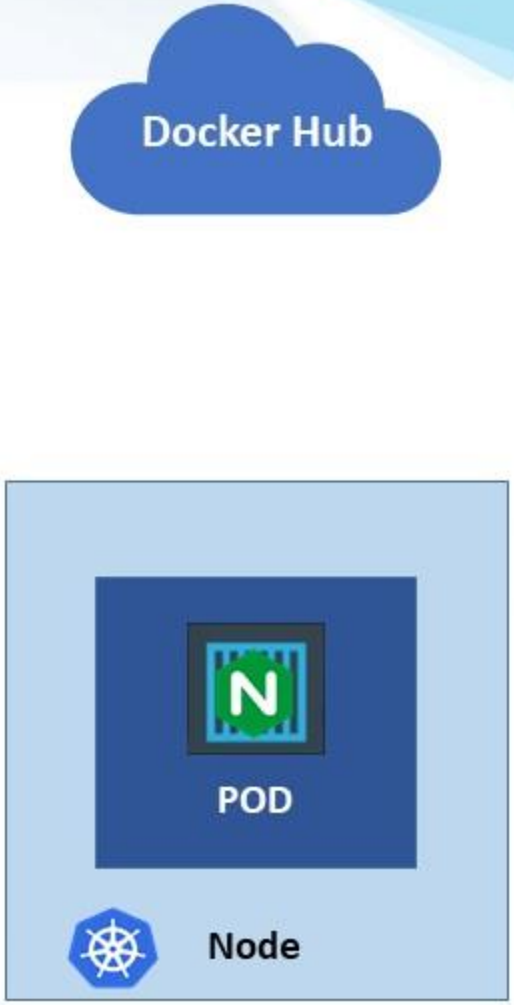
- List all PODs
- **kubectl get pods**



```
C:\Kubernetes>kubectl get pods
NAME                   READY     STATUS              RESTARTS   AGE
nginx-8586cf59-whssr   0/1       ContainerCreating   0          3s
```

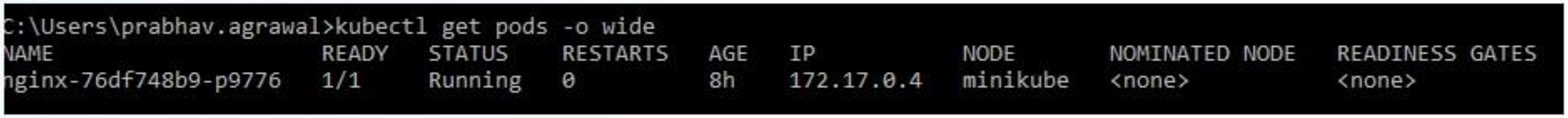


```
C:\Kubernetes>kubectl get pods
NAME                   READY     STATUS    RESTARTS   AGE
nginx-8586cf59-whssr   1/1       Running   0          8s
```



kubectl

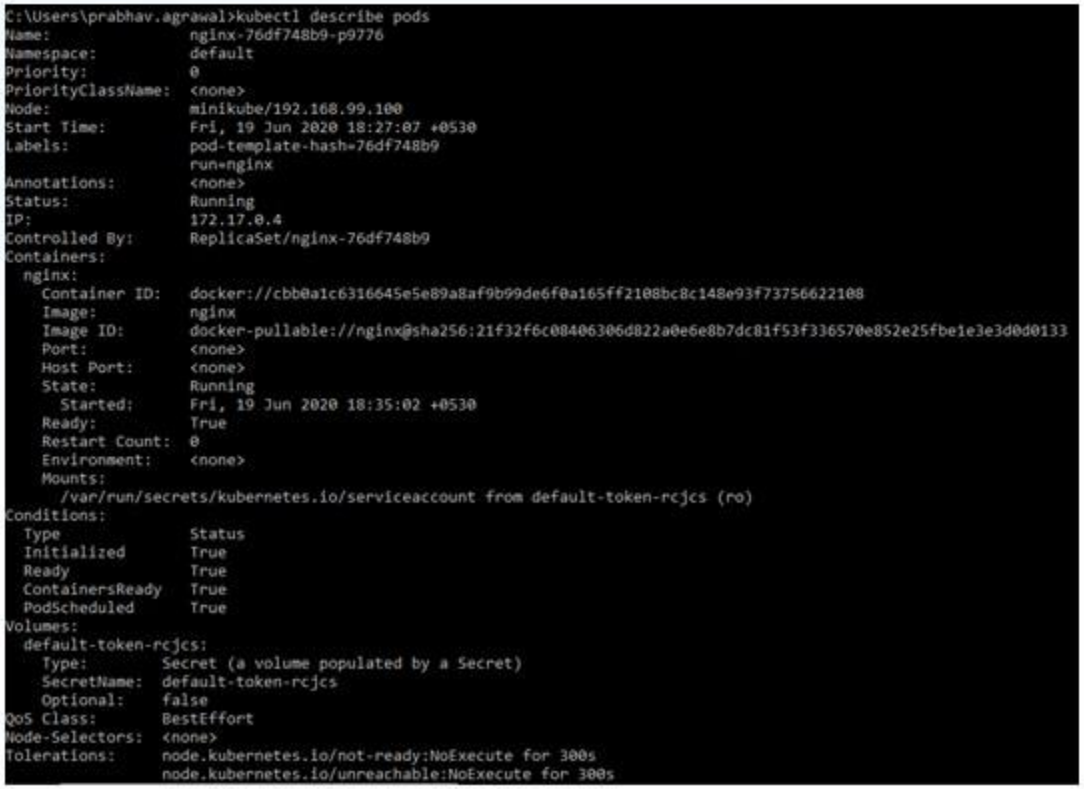
- List all PODs with IP Address
- **kubectl get pods -o wide**



```
:\Users\prabhav.agrawal>kubectl get pods -o wide
NAME                     READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
ginx-76df748b9-p9776     1/1     Running   0          8h    172.17.0.4   minikube   <none>           <none>
```

kubectl

- Describe PODs
- **kubectl describe pods**

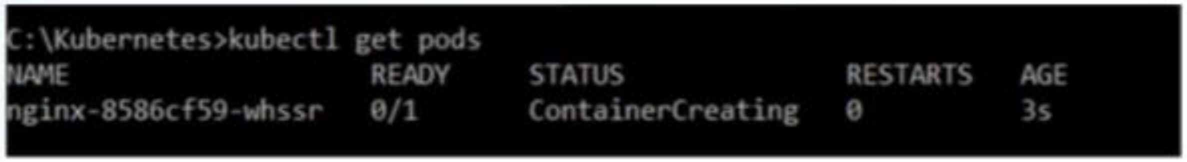


```
C:\Users\prabhav.agrawal>kubectl describe pods
Name:               nginx-76df748b9-p9776
Namespace:          default
Priority:           0
PriorityClassName:  <none>
Node:               minikube/192.168.99.100
Start Time:         Fri, 19 Jun 2020 18:27:07 +0530
Labels:             pod-template-hash=76df748b9
                    run=nginx
Annotations:        <none>
Status:             Running
IP:                 172.17.0.4
Controlled By:      ReplicaSet/nginx-76df748b9
Containers:
  nginx:
    Container ID:   docker://cbb0a1c6316645e5e89a8af9b99de6f0a165ff2108bc8c148e93f73756622108
    Image:          nginx
    Image ID:       docker-pullable://nginx@sha256:21f32f6c08406306d822a0e6e8b7dc81f53f336570e852e25fbe1e3e3d0d0133
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Fri, 19 Jun 2020 18:35:02 +0530
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-rcjcs (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-rcjcs:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-rcjcs
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
```

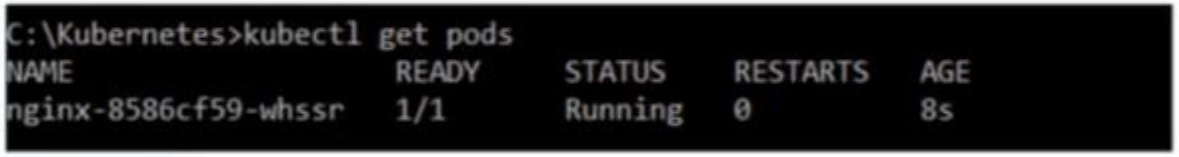
SkillAssure

kubectl

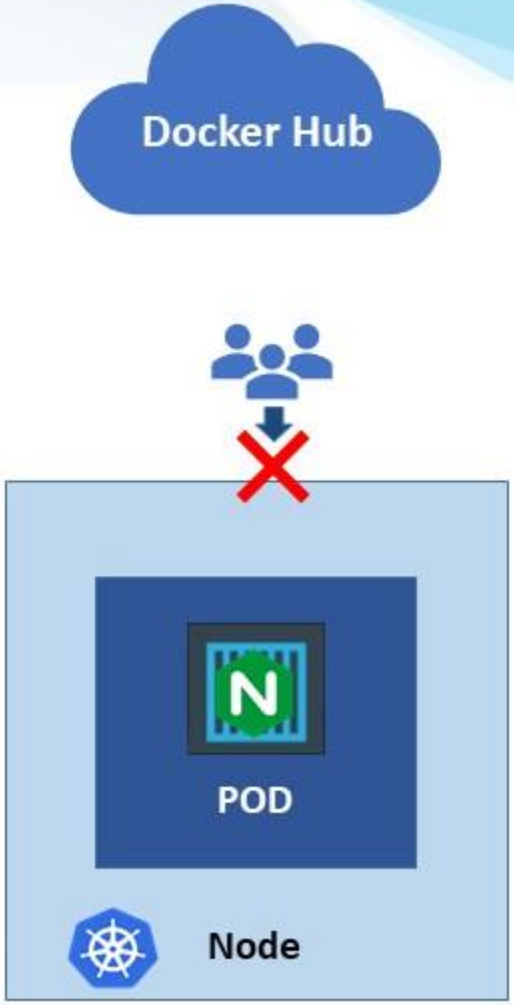
- User won't be able to access the nginx POD because we haven't made the web server accessible to external users



```
C:\Kubernetes>kubectl get pods
NAME                   READY     STATUS              RESTARTS   AGE
nginx-8586cf59-whssr   0/1       ContainerCreating   0          3s
```



```
C:\Kubernetes>kubectl get pods
NAME                   READY     STATUS    RESTARTS   AGE
nginx-8586cf59-whssr   1/1       Running   0          8s
```



Quiz

SkillAssure

Quiz

The smallest unit you can create in Kubernetes object model is:

- Service
- Application
- [✓ Pod]
- Container
- Process

SkillAssure

Quiz

A Pod can only have one container in it

- True
- [✓ False]

SkillAssure

Quiz

What is the right approach to scale an application

- Deploy additional containers in the pod
- [✓ Deploy additional pods]
- You cannot scale an application in kuberenetes. This is not a use-case of kuberenetes

POD with YAML

POD with YAML

- Kubernetes definition file always contains 4 top level fields.
  - **apiVersion**
  - **Kind**
  - **metadata**
  - **Spec**
- These are the top level or root level properties.
- These are also required fields so you must have them in your configuration file.

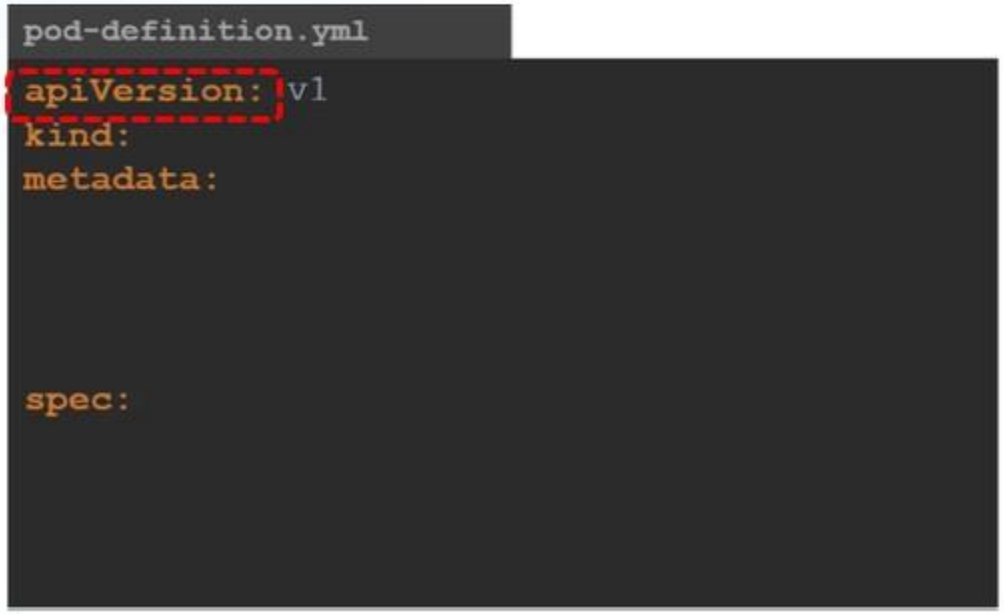
pod-definition.yml

```yaml
apiVersion:
kind:
metadata:




spec:
```

SkillAssure

POD with YAML (apiVersion)

- This is the version of the Kubernetes API you're using to create the objects.
- Depending on what we are trying to create we must use the right API version.
- Since we are working on POD, we will set the API version as v1
- Few other possible values for this field are
  - apps/v1.
  - extensions/v1Beta
  - etc.

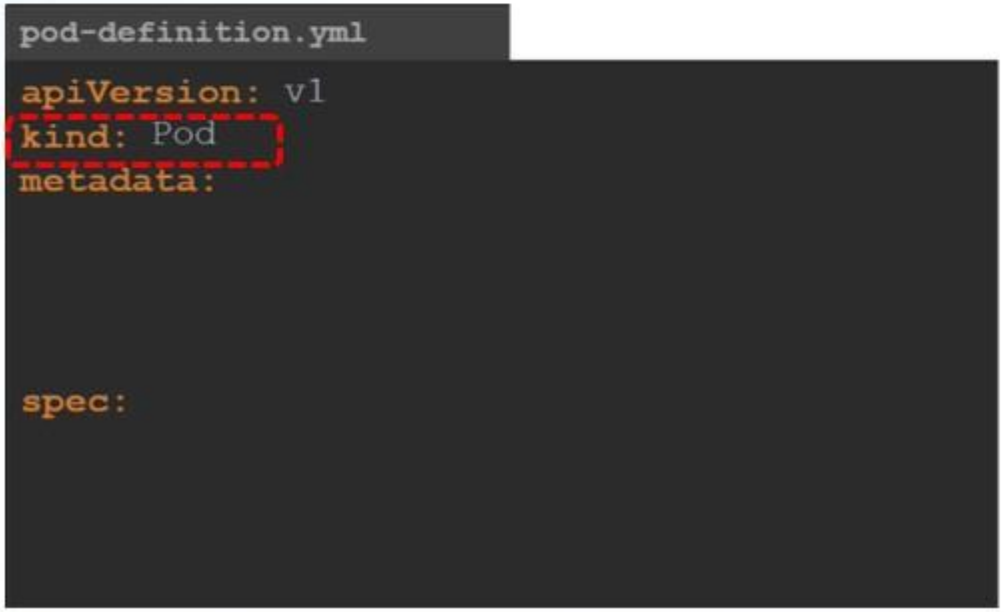


```
pod-definition.yml
apiVersion: v1
kind:
metadata:




spec:
```

SkillAssure

POD with YAML (kind)

- The kind refers to the type of object we are trying to create
- Since we are working on POD, we will set the kind as v1
- Few other possible values for this field are
  - Service
  - ReplicaSet
  - Deployment



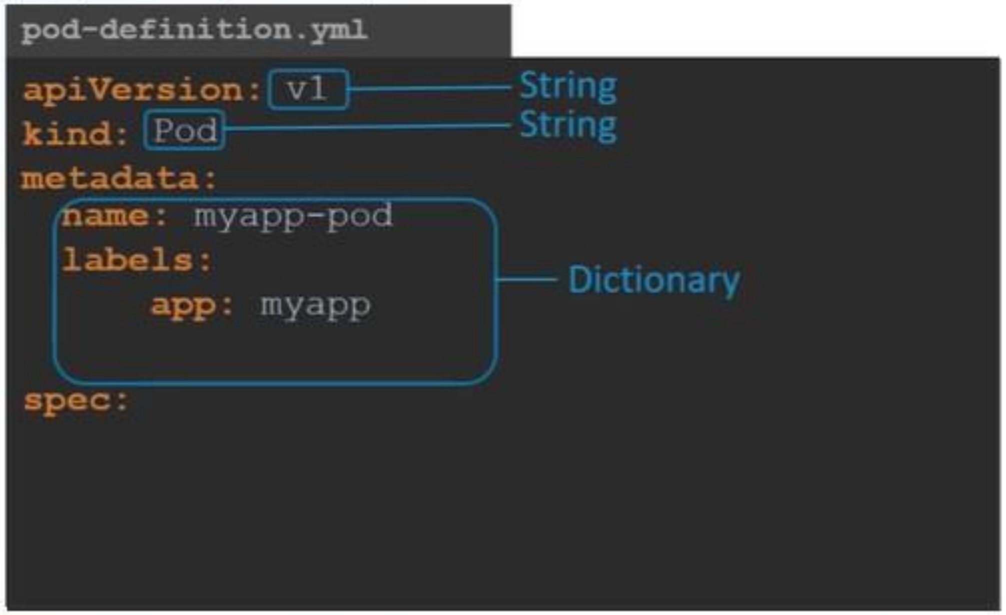
pod-definition.yml

```yaml
apiVersion: v1
kind: Pod
metadata:




spec:
```

SkillAssure

POD with YAML (metadata)

- The metadata is data about the object like its name labels etc.
- Unlike the first two where we have specified a string value, metadata is in the form of a dictionary.
- Everything under metadata is intended to the right a little bit and so names and labels are children of metadata.
  - name (String)
  - labels (Dictionary within the metadata dictionary)
  - labels can have any key value pairs as you wish.

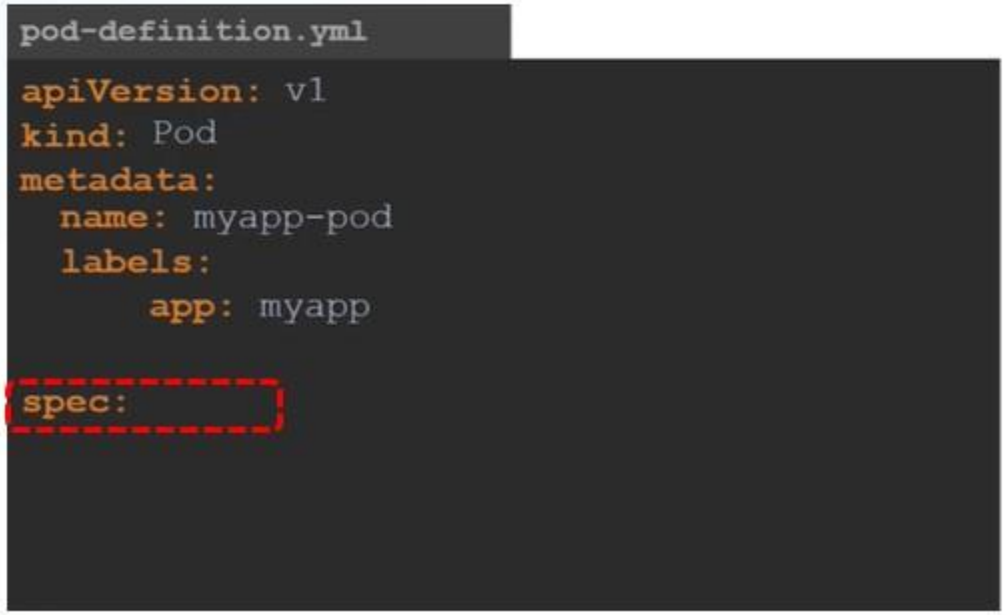


pod-definition.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
      app: myapp

spec:
```

POD with YAML (spec)

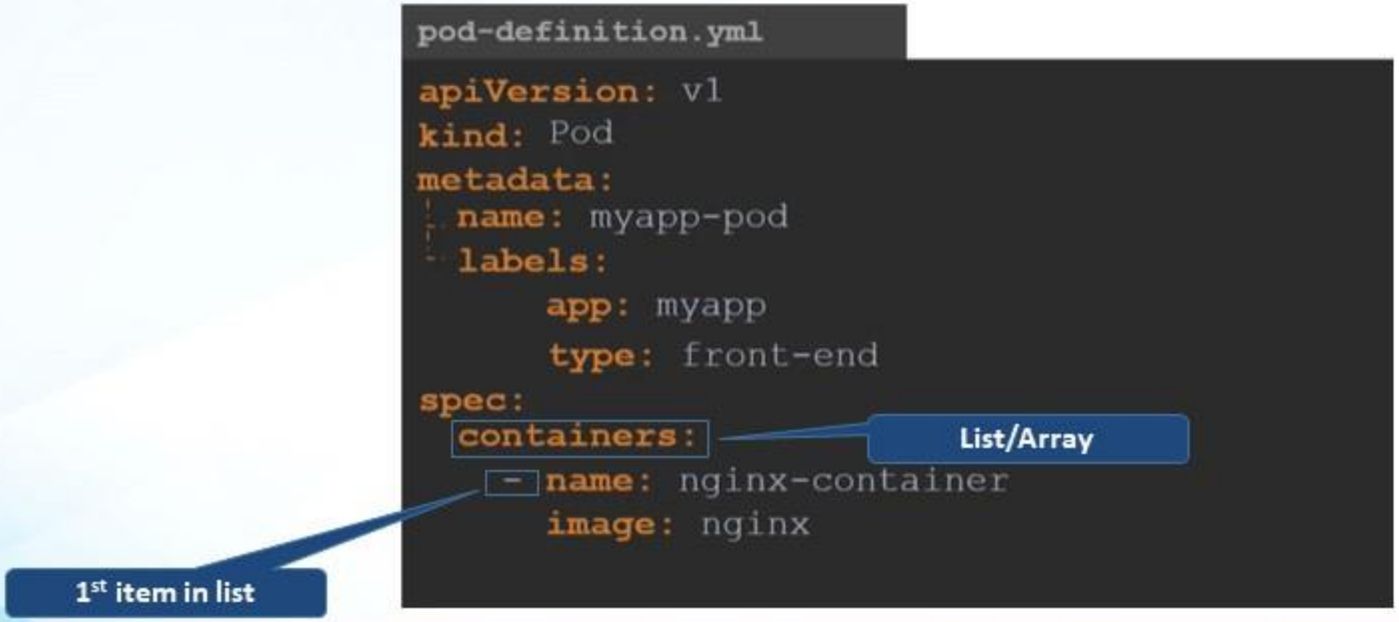
- Spec is a dictionary
- Depending on the object we are going to create, this is where we would provide additional information to Kubernetes
- Spec is going to be different for different objects



```
pod-definition.yml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
      app: myapp

spec:
```

POD with YAML (spec)

- Let's look at spec for a single container pod using nginix image
- There is a property under it called containers which is a list or an array.
  - Because the PODs can have multiple containers within them
- The - right before the name indicates that this is the first item in the list
- The item in the list is a dictionary, so we add a name and image property

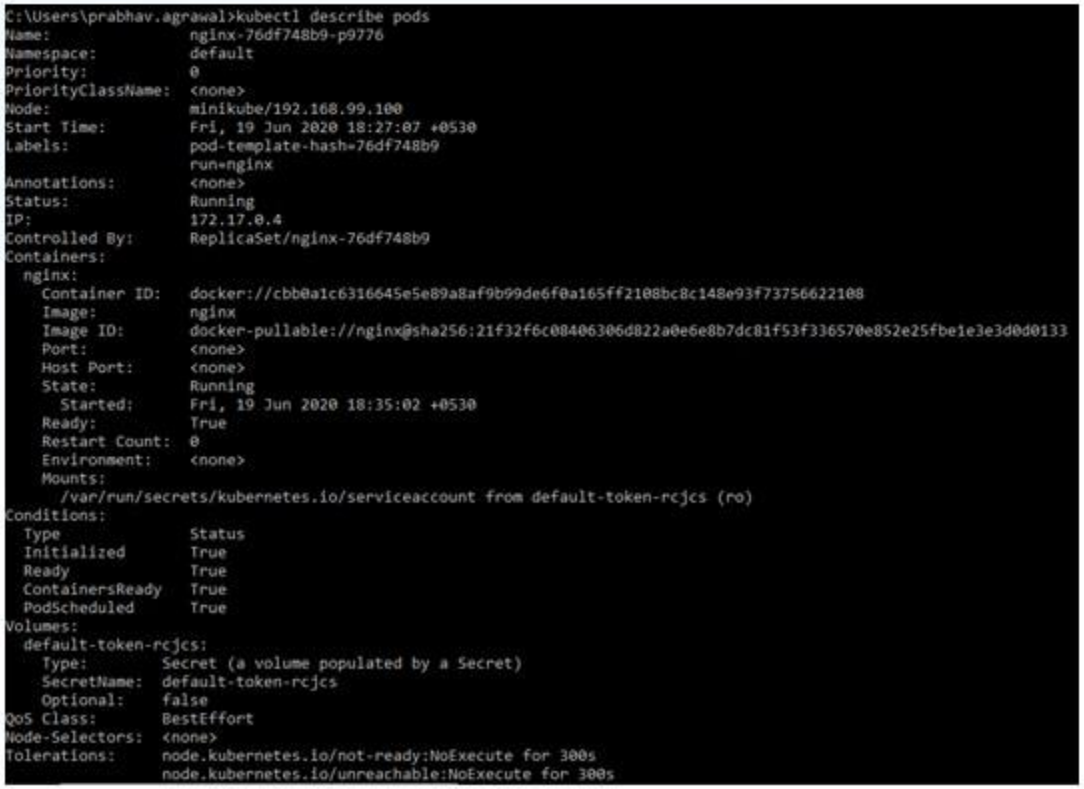


```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
      app: myapp
      type: front-end
spec:
  containers:
    - name: nginx-container
      image: nginx
```

kubectl

- Create a POD using YAML file

kubectl create -f pod-definition.yml



```
C:\Users\prabhav.agrawal>kubectl describe pods
Name:               nginx-76df748b9-p9776
Namespace:          default
Priority:           0
PriorityClassName:  <none>
Node:               minikube/192.168.99.100
Start Time:         Fri, 19 Jun 2020 18:27:07 +0530
Labels:             pod-template-hash=76df748b9
                    run=nginx
Annotations:        <none>
Status:             Running
IP:                 172.17.0.4
Controlled By:      ReplicaSet/nginx-76df748b9
Containers:
  nginx:
    Container ID:   docker://cbb0a1c6316645e5e89a8af9b99de6f0a165ff2108bc8c148e93f73756622108
    Image:          nginx
    Image ID:       docker-pullable://nginx@sha256:21f32f6c08406306d822a0e6e8b7dc81f53f336570e852e25fbe1e3e3d0d0133
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Fri, 19 Jun 2020 18:35:02 +0530
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-rcjcs (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-rcjcs:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-rcjcs
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
```

Exercise 1

<u>**Introduction**</u>: Let's start simple! Given a pod-definition.yml file. We are only getting started with it. I have added two root level properties apiVersion and kind

<u>**Instruction**</u>: Add the missing two properties

pod-definition.yml

```yaml
apiVersion:
kind:
```

Exercise 1 - Solution

Introduction: Let's start simple! Given a pod-definition.yml file. We are only getting started with it. I have added two root level properties apiVersion and kind

Instruction: Add the missing two properties

pod-definition.yml

```
apiVersion:
kind:
metadata:
spec:
```

SkillAssure

Exercise 2

Introduction: Let’s now populate values for each property. Start with **apiVersion**

Instruction: Update value of **apiVersion** to **v1**

pod-definition.yml

```
apiVersion:
kind:
metadata:
spec:
```

Exercise 2 - Solution

Introduction: Let's now populate values for each property. Start with **apiVersion**

Instruction: Update value of **apiVersion** to **v1**

pod-definition.yml

```yaml
apiVersion: v1
kind:
metadata:
spec:
```

Exercise 3

Introduction: Let’s now populate values for each property. Start with **kind**

Instruction: Update value of **kind** to **POD**

pod-definition.yml

```yaml
apiVersion: v1
kind:
metadata:
spec:
```

Exercise 3 - Solution

Introduction: Let’s now populate values for each property. Start with **kind**

Instruction: Update value of **kind** to **POD**

pod-definition.yml

```
apiVersion: v1
kind: Pod
metadata:
spec:
```

Exercise 4

Introduction: Let’s now get to the metadata section

Instruction: Add a property **name** under metadata with value **myapp-pod**

pod-definition.yml

```
apiVersion: v1
kind: Pod
metadata:
spec:
```

SkillAssure

Exercise 4 - Solution

Introduction: Let's now get to the metadata section

Instruction: Add a property **name** under metadata with value **myapp-pod**

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
spec:
```

Exercise 5

Introduction: Let's add some label to our Pod

Instruction: Add a property **labels** under metadata with a child property **app** with a value **myapp**

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
spec:
```

SkillAssure

Exercise 5 - Solution

Introduction: Let’s add some label to our Pod

Instruction: Add a property **labels** under metadata with a child property **app** with a value **myapp**

pod-definition.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
```

Exercise 6

Introduction: Let's provide information regarding docker image

Instruction: Add a property **containers** under **spec** section. Do not add anything else under it.

pod-definition.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
```

Exercise 6 - Solution

Introduction: Let's provide information regarding docker image

Instruction: Add a property **containers** under **spec** section. Do not add anything else under it.

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
```

Exercise 7

Introduction: Let's provide information regarding docker image

Instruction: Containers is an array/list. Create the **first element/item** in the array/list and add the following properties to it: **name – nginx** and **image – nginx**

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
```

Exercise 7 - Solution

Introduction: Let's provide information regarding docker image

Instruction: Containers is an array/list. Create the **first element/item** in the array/list and add the following properties to it: **name – nginx** and **image – nginx**

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
    - name: nginx
      image: nginx
```

Exercise 7 - Solution

Introduction: Let's provide information regarding docker image

Instruction: Containers is an array/list. Create the **first element/item** in the array/list and add the following properties to it: **name – nginx** and **image – nginx**

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
    - name: nginx
      image: nginx
```

Exercise 8

Introduction: Let's try creating 1 more file. Now all on your own.

Instruction: Create a Kubernetes Pod definition file using below values:

- **Name:** postgres
- **Labels:** tier => db-tier
- **Container Name:** postgres
- **Image:** postgres

Exercise 8 - Solution

Introduction: Let's try creating 1 more file. Now all on your own.

Instruction: Create a Kubernetes Pod definition file using below values:

- **Name:** postgres
- **Labels:** tier => db-tier
- **Container Name:** postgres
- **Image:** postgres

pod-definition.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: postgres
  labels:
    tier: db-tier
spec:
  containers:
    - name: postgres
      image: postgres
```

Exercise 9

Introduction: Postgres Docker Image requires an environment variable to be set of r password.

Instruction: Set an environment variable for the docker container. **POSTGRES_PASSWORD** with a value **mysecretpassword**.

Hint: To pass an environment variable add a new property **env** to the container object. It is a sibling of image and name. **env** is an array/list. So add a new liner under it. The item will have properties **name** and **value**. **Name** should be the name of the environment variable - **POSTGRES_PASSWORD** and **value** should be the password - **mysecretpassword**

Exercise 9 - Solution

Introduction: Postgres Docker Image requires an environment variable to be set of r password.

Instruction: Set an environment variable for the docker container. **POSTGRES_PASSWORD** with a value **mysecretpassword**.

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: postgres
  labels:
    tier: db-tier
spec:
  containers:
    - name: postgres
      image: postgres
      env:
        -
          name: POSTGRES_PASSWORD
          value:  mysecretpassword
```

Exercise

Thank You