**Approaches to create Object on k8s:**

There are two types of approaches.

* Declarative approach 2. Imperative approach

**Declarative approach:**

Writing yaml or json file for creating objects. The objects should be anything like pods, deployment, daemon set etc.

For creating these things using kubectl command we have one approach called declarative approach.

Kubectl command is the CLI tool which will give this particular API what we have write inside yaml file to kubernetes cluster. It will look into the yaml file it will do schema validation then it will pass information to “etcd” which is key-value pair storage.

**Imperative approach:** Creating objects without using yaml file is called imperative approach. If you want to do some patch work or to do some certain actions.

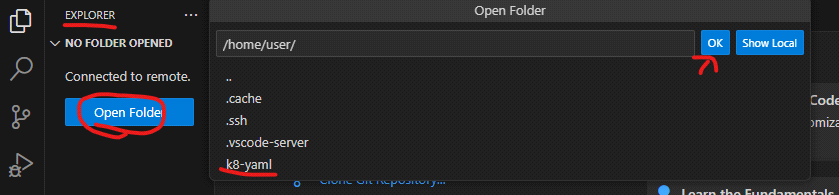
**Creating a pod using Declarative approach:**

In the VM create a directory

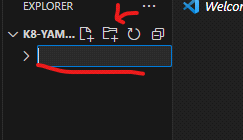
# mkdir k8-yaml



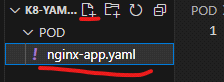
Then go to Explore – open folder then it will show the directories in the virtual machine. Select k8-yaml – then click on ok.

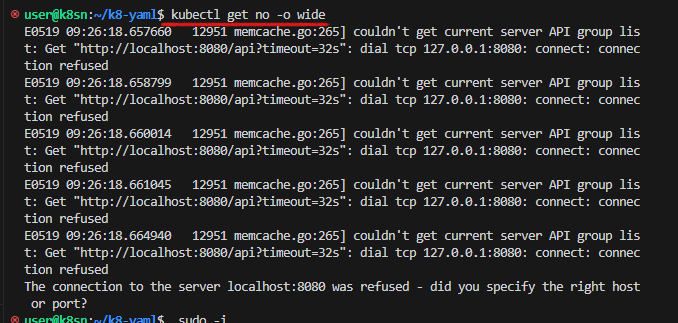


Inside of K8-yaml folder create another folder with the name POD. For that click on the icon give the name of the folder.



Now select the folder POD. In that we created a file with the name nginx-app.yaml





To solve this error, Now we are copying kube config file to the user. Then only without putting sudo we can execute kubectl commands.

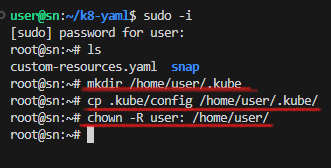
# mkdir /home/user/.kube

copy the configuration file to the path /home/user/.kube/

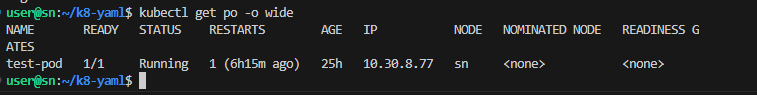
# cp .kube/config /home/user/.kube/

Give the permissions to the user to access the file.

# chown -R user: /home/user/



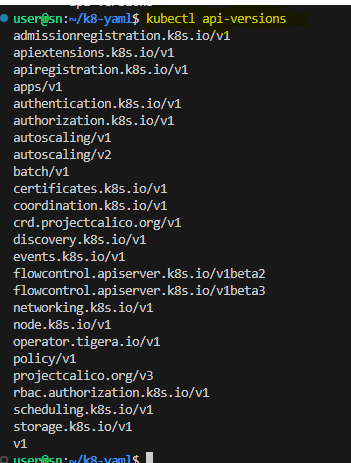
Now we are able to access exit from the root and execute the command.



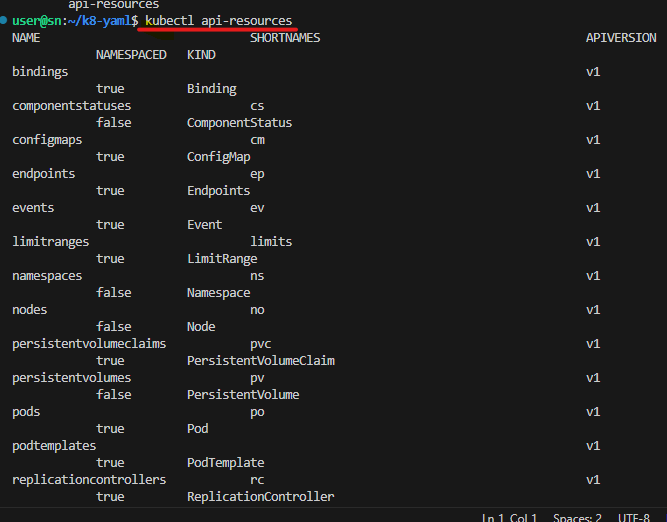
To know in kubernetes cluster which object versions needs to install use below command.

# kubectl api-versions

So, these are the versions of objects we need to use in this cluster.

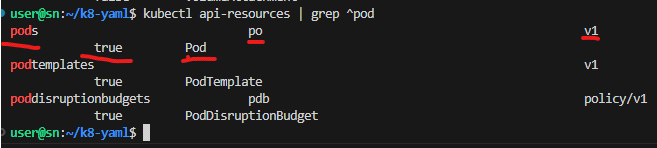


These are the resources and versions we need to use.



In this result to filter out exactly about pod

# kubectl api-resources | grep ^pod



# apiVersion: here we provide the version of object supported by the current cluster.

# kind: Pod ### the object which you want to create mention it here

# metadata: In side of metadata only we give the name for the pod and labels

# labels: are just the more information about the pod to filter it.

# spec: means specifications. The specifications which we want to provide for the pod mention here. Under specifications only we provide container details.

#Containers: under this we use the parameters

# name: name we want to assign to the container

# image: the base image of the container

# ports:

# containerPort: on which port container will listen. It doesn’t mean that the port which we provide on that only the application will listen.

# volumeMount:

# name: name of the volume mount

# mountPath: the path of the container where data stored

# volumes: this parameter we need to write parallel to the container

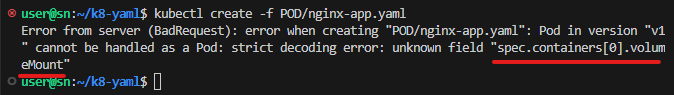
# name: here we need to provide the name which we provide in the volume mount. Otherwise it doesn’t mount with the container path.

# emptyDir: {}

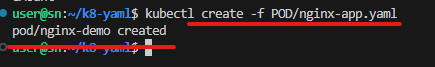
Now we need to give this declarative approach to the api-server. For that we need to use either kubectl create/apply commands.

Run below command to deploy pod.

# kubectl create -f POD/nginx-app.yaml



In the above page it is showing the error in the underlined line because there is a spelling mistake.



The code in the nginx-app.yaml file is

apiVersion: v1

kind: Pod

metadata:

  name: nginx-demo

  labels:

    app: nginx

    env: prod

spec:

  containers:

    - name: nginx

      image: nginx:1.23.0

      ports:

        - containerPort: 80

      volumeMounts:

        - name: nginx-data

          mountPath: /var/www/html

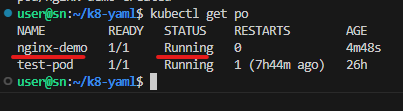
  volumes:

    - name: nginx-data

      emptyDir: {}

To know the status of the we can run any of the command kubectl get pods/po/pod.

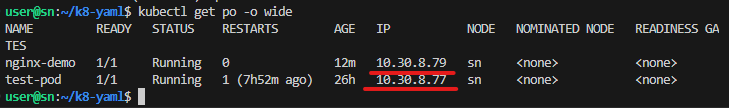
# kubectl get po



In the Ready 1/1 actual/desired. If the actual value mismatched with the desired one means, there is an error.

To know more information about the pod use the following command.

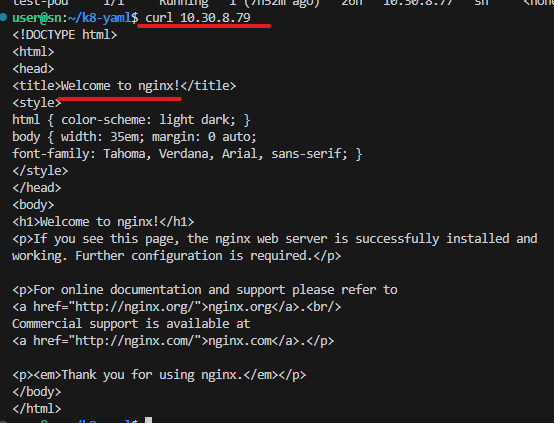
# kubectl get po -o wide



In the output the pod is allotted with IP address with the CIDR which we mention.

# curl 10.30.8.79

We are getting the welcome to nginx page means the application inside the pod is successfully running.



Creating a pod using namespace in declarative approach

Create a file nginx-app.yaml – place the following code in that.

apiVersion: v1

kind: Pod

metadata:

  name: nginx-demo

  namespace: my-namespace

  labels:

    app: nginx

    env: prod

spec:

  containers:

    - name: nginx

      image: nginx:1.23.0

      command:

        - sleep

        - "3600"

      ports:

        - containerPort: 80

      volumeMounts:

        - name: nginx-data

          mountPath: /var/www/html

  volumes:

    - name: nginx-data

      emptyDir: {}

Now command to create pod

# kubectl create -f POD/nginx-app.yaml

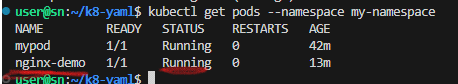


Now to check the status of the pod I executed the below command.

# kubectl get po -o wide

But after executing this command I didn’t get the result of my pod. My pod name is nginx-demo. But in the output of pods it is not there. To know the status of the pod run the below command.

# kubectl get pods --namespace my-namespace



Now we got the result because we created pour POD is under a specified namespace. So we need to mention the namespace.

* To check the events associated with the pod run the below command. This will help you to identify any errors or issues during the creation process:

# kubectl describe pod mypod --namespace my-namespace

* The below command will display the logs generated by the container running inside the pod. By reviewing the logs, you can check if your application is running as expected or if there are any errors or issues reported.

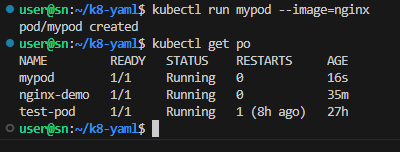
# kubectl logs nginx-demo --namespace my-namespace

**Creating a POD using imperative approach:**

To create a pod using the imperative approach in kubernetes, you can use the Kubectl command-line tool. Here’s an example of how to create a pod using the imperative approach:

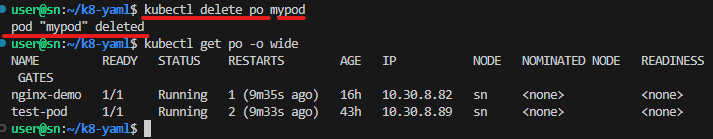
* Open the terminal or command prompt.
* Run the following command to create a pod. Replace the name mypod with the name which you want. –image is the parameter where we want to give the image name.

# kubectl run mypod --image=nginx



Now delete the mypod using the below command

# kubectl delete po mypod

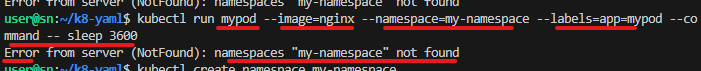


**Namespaces:**

Namespaces are used to create virtual clusters within the physical cluster. They provide a way to divide cluster resources among multiple users, teams, or applications, effectively isolating them from each.

Create a pod with the ‘namespace’ parameter using ‘kubectl’ command.

# kubectl run mypod --image=image-name --namespace=my-namespace --labels=app=mypod --command -- sleep 3600

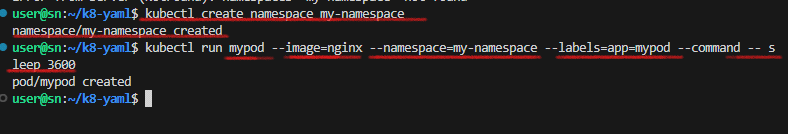


After executing the command, we got the error namespaces “my-namespace” not found. Means, there is no any namespace with the name my-namespace. To solve this error, we need to create a namespace first. For creating namespace use the below command

# kubectl create namespace my-namespace

Now run the command for creating pod with the parameters namespaces and labels

# kubectl run mypod --image=nginx --namespace=my-namespace --labels=app=mypod --command -- sleep 3600



**References:**

* For imperative approach: [https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Imperative-vs-declarative-Kubernetes-commands-Whats-the-difference#:~:text=Imperative%20K8s%20pod%20configuration HYPERLINK "https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Imperative-vs-declarative-Kubernetes-commands-Whats-the-difference" HYPERLINK "https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Imperative-vs-declarative-Kubernetes-commands-Whats-the-difference" HYPERLINK "https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Imperative-vs-declarative-Kubernetes-commands-Whats-the-difference"& HYPERLINK "https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Imperative-vs-declarative-Kubernetes-commands-Whats-the-difference" HYPERLINK "https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Imperative-vs-declarative-Kubernetes-commands-Whats-the-difference" HYPERLINK "https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Imperative-vs-declarative-Kubernetes-commands-Whats-the-difference"text=To%20create%20a%20pod%20declaratively,file%20in%20YAML%2C%20named%20nicepod](https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Imperative-vs-declarative-Kubernetes-commands-Whats-the-difference).