

# Installation of Minikube and Detailed

<u>Kubernetes Objects</u>	
<ul style="list-style-type: none"> <li>→ Kubernetes uses <u>Objects</u> to represent the <u>State of your Cluster</u></li> <li>→ What <u>Containerized applications</u> are running (and on which node)</li> <li>→ The policies around how those applications behave, such as restart policies, upgrades and fault tolerance.</li> <li>→ Once you <u>Create the Object</u>, the Kubernetes System will constantly work to ensure that Object exists and maintains Cluster's <u>desired State</u>.</li> <li>→ Every Kubernetes Object includes two nested fields that govern the Object Config: the <u>Object spec</u> and the <u>Object status</u></li> </ul>	<ul style="list-style-type: none"> <li>→ The <u>spec</u>, which we provide, describes your <u>desired state</u> for the object - the characteristics that you want the object to have</li> <li>→ The <u>status</u> describes the <u>actual state</u> of the object and is supplied and updated by the Kubernetes system</li> <li>→ All objects are identified by a unique name and a UID.</li> </ul> <p>The Basic Kubernetes Objects include:</p> <ol style="list-style-type: none"> <li>1) Pod</li> <li>2) Service</li> <li>3) Volume</li> <li>4) Namespace</li> <li>5) Replicaset</li> <li>6) Secrets</li> <li>7) ConfigMaps</li> <li>8) Deployment</li> <li>9) Jobs</li> <li>10) Daemonsets</li> </ol> <p>(Manifest → yml)</p>

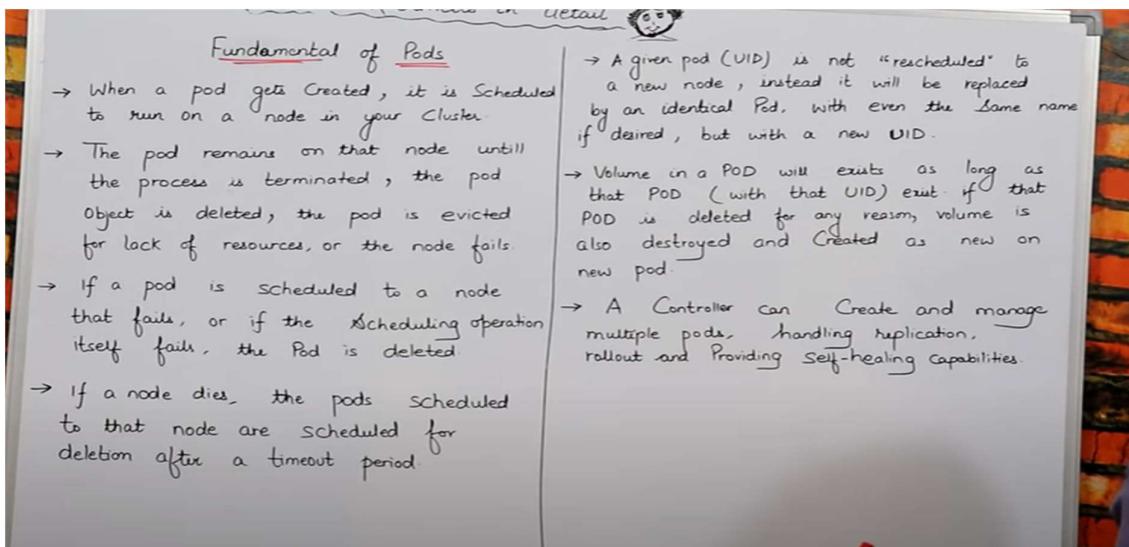
- ➔ Hum manifest file mein jo bhi likh kar denge kubernetes puri koshish karega ki us desired state ko pura kare matlab jo bhi kuch humare manifest file mein likha hai usko ye pura kar de.
- ➔ Har ek object ka alag unique name aur aur alag unique id hota hai.
- ➔ Hum manifest file ko json aur yml mein likhate hai.

Relationship b/w these Objects		Kubernetes Object Management								
<ul style="list-style-type: none"> <li>→ Pod manages Containers</li> <li>→ Replicaset manage pods</li> <li>→ Services expose pod processes to the outside world</li> <li>→ Configmaps and Secrets help you configure pods.</li> </ul>	 <p><u>Kubernetes Object Management</u></p> <p>The <u>Kubectl</u> command line tool supports several different ways to create and manage Kubernetes objects.</p> <table border="1"> <thead> <tr> <th>Management Technique</th> <th>Operates on</th> <th>Recommended Environment</th> </tr> </thead> <tbody> <tr> <td>Imperative Commands</td> <td>Live Objects</td> <td>Development projects</td> </tr> <tr> <td>Declarative Object Configuration</td> <td>Individual files (Yml/Json)</td> <td>Production</td> </tr> </tbody> </table> <p><u>Declarative</u> is about describing what you are trying to achieve, without instructing how to do it.</p> <p><u>Imperative</u>, explicitly tells "how to accomplish it".</p>	Management Technique	Operates on	Recommended Environment	Imperative Commands	Live Objects	Development projects	Declarative Object Configuration	Individual files (Yml/Json)	Production
Management Technique	Operates on	Recommended Environment								
Imperative Commands	Live Objects	Development projects								
Declarative Object Configuration	Individual files (Yml/Json)	Production								
<p><u>Kubernetes Object</u></p> <ul style="list-style-type: none"> <li>→ It represents as <u>JSON</u> or <u>YAML</u> files.</li> <li>→ You create these and then push them to the Kubernetes API with <u>kubectl</u>.</li> </ul> <p><u>State of the Object</u></p> <ul style="list-style-type: none"> <li>→ Replicas (2/2)</li> <li>→ Image (Tomcat / Ubuntu)</li> <li>→ Name</li> <li>→ Port</li> <li>→ Volume</li> </ul>	<ul style="list-style-type: none"> <li>→ startup cmd</li> <li>→ Detached (default)</li> </ul>									

- ➔ Kubectl eak command line tool hota hai, Kubectl kein through hi hum sarii command likhenge, starting mein kubectl then uske baad jo command dena chahhenge vo.

→ **Declarative** = declarative ka matlab hai ki agar hum kuch chahate hai toh usko eak file mein andar describe kar dete hai ki mujhe ye ye chahiye aur vo file jab execute hoti hai toh sab apane aap run ho jata hai.

→ **Imperative** = Imperative matlab hai ki humko jo bhi cheej chahihe har eak ko command mein baar baar likho then execute karao.



→ **Create 1 ec2 machine :- 2vcpu, t2medium**

→ ubuntu@ip-172-31-12-153:~\$ sudo su  
→ root@ip-172-31-12-153:/home/ubuntu# apt update  
→ root@ip-172-31-12-153:/home/ubuntu# apt install docker

→ root@ip-172-31-12-153:/home/ubuntu# curl -LO  
https://storage.googleapis.com/kubernetes-release/release/\$(curl -s  
https://storage.googleapis.com/kubernetes-  
release/release/stable.txt)/bin/linux/amd64/kubectl && chmod +x ./kubectl  
&& sudo mv ./kubectl /usr/local/bin/kubectl

```
Select root@ip-172-31-12-153:/home/ubuntu
root@ip-172-31-12-153:/home/ubuntu# curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/table.txt)/bin/linux/amd64/kubectl && chmod +x ./kubectl && sudo mv ./kubectl /usr/local/bin/kubectl
% Total    % Received % Xferd  Average Speed   Time   Time  Current
          Dload  Upload Total Spent   Left Speed
100 44.4M 100 44.4M    0     0  54.9M    0 --:--:--:--:--:-- 54.8M
root@ip-172-31-12-153:/home/ubuntu#
```

Iss command se hum kubectl ko download karke install karenge, aise karne se humara command line interface start ho jayega aisa isliye kyuki aage hum jo bhi kaam karenge kubectl command ki help sein karenge.

→ root@ip-172-31-12-153:/home/ubuntu# which kubectl

```
root@ip-172-31-12-153:/home/ubuntu# which kubectl
/usr/local/bin/kubectl
root@ip-172-31-12-153:/home/ubuntu#
```

(hum dekh sakte hai ki humara kubectl kaha install huva hai)

→ root@ip-172-31-12-153:/home/ubuntu# kubectl version

```
root@ip-172-31-12-153:/home/ubuntu# kubectl version
Client Version: version.Info{Major:"1", Minor:"23", GitVersion:"v1.23.0", GitCommit:"ab69524f795c42094a6630298ff53f3c3ebab7f4", GitTreeState:"clean", BuildDate:"2021-12-07T
19:16:38Z", GoVersion:"go1.17.3", Compiler:"gc", Platform:"linux/amd64"}
The connection to the server localhost:8080 was refused - did you specify the right host or port?
root@ip-172-31-12-153:/home/ubuntu#
```

(kubectl ka version check kar sakte hai hum)

→ **Example =** agar mere pass **1 master** , aur **3 worker node** hai, aur mujhe pod banana hai , toh mujhe manifest file mein andar batana padega ki **Node2** mein andar jaakar humara pod create karo , aur agar hum ye nahi bataye ki kis node par banana hai pod , toh kubernetes apne aap jis node mein andar usko samjh aayega vo apne aap us node mein andar jaakar pod create kar dega .

- Pod humara tab tak fail nahi hota jab tak koi process fail na ho jaye , koi pod delete na ho jaye , aur agar humara node hi band ho gaya toh humara pod bhi easily sein cancel ho jayega.
- Agara humane kisi node par pod create kiya hai aur vo node hi fail hogaya toh us case mein mera pod bhi fail ho jayega .
- Agar humane koi pod bana rakha hai node mein andar aur vo **node fail** hogaya toh humara pod eak **time period tak active rahega** agar uss time period tak node apne aap ko active nahi kar paya toh humara pod bhi cancel ho jayega.
- Agar humara pod fail hojata hai toh vahi same pod restart nahi hogta kubernetes eak naya **same pod bana dega** aur jo pod fail huva hogta uski sari dependencies new pod mein aajyegi aur pod naya bana dega, aur humko pata kaise chelega ki naya pod kubernetes ne create kiya hai ya vahi purana vala hi restart kar diya toh us case mein hum dekhenge pura jo pod hogta uski **UID(unique id)** alag hogi aur naye pod ki **UID(unique id)** different hogi.
- Agar humane pod ko delete kar diya toh usase juda **volume** apne aap delete ho jayega.
- Kubernetes mein handling replica, rollout, self-healing ye feature nahi hote hai isko use karne ke liye humko alag sein **API** aur kuch alag sein **controller** add karne hote hai, uske baad hum iss feature ko kubernetes mein use kar sakte hai.

## 3 types of Kubernetes configuration are:-

### 1 = All in one single node installation (use only for practice)

With all-in-one, all the master and Worker Components are installed on a Single node. This is very useful for learning, development and testing. This type should not be used in Production. Minikube is one such example, and we are going to explore it soon.

## Minikube Setup

- **Minikube** is a tool that makes it easy to run Kubernetes locally
- Minikube runs a single-node Kubernetes cluster inside a Linux VM
- It's aimed on users who want to just test it out or use it for development
- It cannot spin up a production cluster, it's a one node machine with no high availability
- It works on **Windows, Linux, and MacOS**
- You will need **Virtualization Software** installed to run minikube:
  - VirtualBox is free and can be downloaded from [www.virtualbox.org](http://www.virtualbox.org)
- You can download minikube from <https://github.com/kubernetes/minikube>
- To launch your cluster you just need to enter (in a shell / terminal / powershell):

```
$ minikube start
```

- Eak hi instance par master par bhi kaam kar raha hogा aur eak hi instance par worker bhi kaam kar raha hogा, as a practice karne kein liye ye bas badhiya hai as a production kaam nahi aayega ye
- Minikube practice karane kein liye theek hai production par kaam karane kein liye minikube better nahi hai
- Minikube mein master kewal eak ho sakta hai aur worker node hum kitane bhi bana sakte hai .

## 2 = Single – Node etcd, Single –Master and Multi-worker installation

In this setup , we have a single master node, which also runs a single-node etcd instance. Multiple worker nodes are Connected to the master node.

- isme humara single etcd hogा matlab single database, single master hogा aur worker-node hum multiple bana sakte hai.
- For example eak master ka instance bana dunga aur 3 worker node ka instance bana dunga

## 3= Single - Node etcd ,Multi –Master and multi –worker node installation

In this setup , we have multiple master nodes, which works in an HA mode , but we have a Single-node etcd instance . Multiple worker nodes are Connected to the Master node.

- ➔ Etcd single node hi rahega
- ➔ eak sein jada master rahenge iska fayada ye rahega ki agar koi eak master bhi koi fail huva toh uski jagah dusara master worker node ko sambhall laega
- ➔ Worker node multiple ho sakte hai

## Install minikube in AWS account

go to aws account → Launch instance  
→ ubuntu 18.04 → t2 medium (2 vCPU)

Now access EC2 via putty → login as "ubuntu"

→ sudo su  
→ apt update & apt -y install docker.io

Now install Kubectl (link will provide you in description)

Then install minikube ✓

→ apt install Conntrack  
→ minikube start --vm-driver=none  
→ minikube status ✓  
→ Kubectl Version ✓

Now onwards, we will use Kubectl Commands

```
→ Kubectl get nodes
O/P          Name      Status   Roles   Age    Version
ip-172-31-34-55  Ready     Master   2m     v1.20.7

→ Kubectl describe node ip-172-31-34-55

→ vi pod1.yml
Kind: Pod
apiVersion: v1
metadata:
  name: testpod
spec:
  containers:
    - name: c00
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Hello-Bhupinder; sleep 5; done"]
  restartPolicy: Never
→ :wq
```

### → Minikube install link :-

curl -Lo minikube

<https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64> && chmod +x minikube && sudo mv minikube /usr/local/bin/

```
root@ip-172-31-12-153:/home/ubuntu# curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 && chmod +x minikube && sudo mv minikube /usr/local/bin/
% Total % Received % Xferd Average Speed Time Time Current
          Dload Upload Total Spent Left Speed
100 66.3M 100 66.3M 0 0 68.4M 0 ---:---:---:---:---:--- 68.3M
root@ip-172-31-12-153:/home/ubuntu#
```

(Is link se hum minikube install kar denge hum dekhenge ki minikube jo install huva hai uski size 68.3 mb hai )

## → apt install Conntrack

Minikube install karne kein baad contrack command chalana hoga otherwise humara minikube bhi kaam nahi karega kayade sein.

```
root@ip-172-31-12-153:/home/ubuntu# apt install conntrack
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  nftables
The following NEW packages will be installed:
  conntrack
0 upgraded, 1 newly installed, 0 to remove and 40 not upgraded.
Need to get 30.3 kB of archives.
After this operation, 104 kB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal/main amd64 conntrack amd64 1:1.4.5-2 [30.3 kB]
Fetched 30.3 kB in 4s (7688 B/s)
Selecting previously unselected package conntrack.
(Reading database ... 63764 files and directories currently installed.)
Preparing to unpack .../conntrack_1%3a1.4.5-2_amd64.deb ...
Unpacking conntrack (1:1.4.5-2) ...
Setting up conntrack (1:1.4.5-2) ...
Processing triggers for man-db (2.9.1-1) ...
root@ip-172-31-12-153:/home/ubuntu#
```

## → minikube start --vm-driver=none

iska matlab ki hum minikube ko start kar rahe hai aur abhi iske andar koi bhi vm driver nahi install kar rahe isliye hum driver ki value ko none kar rakhe hai .

agar hum apane instance ko stop karke phir sein start kar rahe toh humko ye command phir sein run karni hogi

```
root@ip-172-31-12-153:/home/ubuntu# minikube start --vm-driver=none
[+] minikube v1.24.0 on Ubuntu 20.04 (xen/amd64)
[+] Using the none driver based on user configuration
[+] Starting control plane node minikube in cluster minikube
[+] Running on localhost (CPUs=2, Memory=3928MB, Disk=7876MB) ...
[+] OS release is Ubuntu 20.04.3 LTS
[+] Preparing Kubernetes v1.22.3 on Docker 20.10.7 ...
  • kubelet.resolv-conf/run/systemd/resolve/resolv.conf
  > kubeadm.sha256: 64 B / 64 B [=====] 100.00% ? p/s 0s
  > kubectl.sha256: 64 B / 64 B [=====] 100.00% ? p/s 0s
  > kubelet.sha256: 64 B / 64 B [=====] 100.00% ? p/s 0s
  > kubectl: 44.73 MiB / 44.73 MiB [=====] 100.00% 72.74 MiB p/s 800ms
  > kubeadm: 43.71 MiB / 43.71 MiB [=====] 100.00% 69.17 MiB p/s 800ms
  > kubelet: 115.57 MiB / 115.57 MiB [=====] 100.00% 68.08 MiB p/s 1.9s
  • Generating certificates and keys ...
  • Booting up control plane ...
  • Configuring RBAC rules ...
[+] Configuring local host environment ...

[+] The 'none' driver is designed for experts who need to integrate with an existing VM
[+] Most users should use the newer 'docker' driver instead, which does not require root!
[+] For more information, see: https://minikube.sigs.k8s.io/docs/reference/drivers/none/

[+] kubectl and minikube configuration will be stored in /root
[+] To use kubectl or minikube commands as your own user, you may need to relocate them. For example, to overwrite your own settings, run:
  • sudo mv /root/.kube /root/.minikube $HOME
  • sudo chown -R $USER $HOME/.kube $HOME/.minikube

[+] This can also be done automatically by setting the env var CHANGE_MINIKUBE_NONE_USER=true
[+] Verifying Kubernetes components...
  • Using image gcr.io/k8s-minikube/storage-provisioner:v5
[+] Enabled addons: storage-provisioner, default-storageclass
[+] Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
root@ip-172-31-12-153:/home/ubuntu#
```

Mera command line bhi configure ho gaya hai aur mera minikube cluster ready hogaya hai.

## → Minikube status

Hum minikube ka status check kar sakte hai ki ye ready hai ya nahi, hum dekhenege ki humara ubectl bhi running hai humara apiserver bhi running hai .

```
root@ip-172-31-12-153:/home/ubuntu# minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubecfg: Configured
root@ip-172-31-12-153:/home/ubuntu#
```

## → Kubectl version

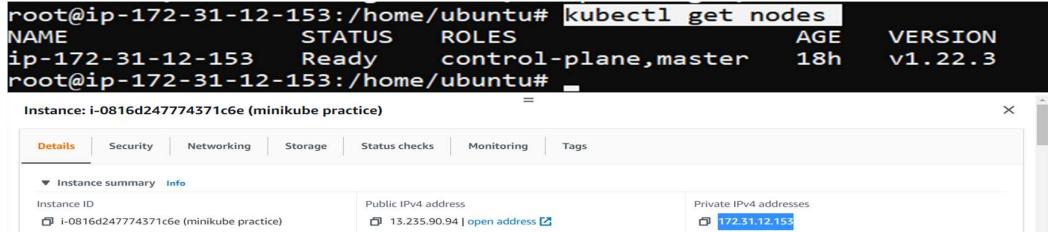
Humko kubectl check karna hoga ki kubectl humara run ho raha ya nahi kyuki aage jtani bhi command hai hum sabme kubectl use karenge.

```
root@ip-172-31-12-153:/home/ubuntu# kubectl version
Client Version: version.Info{Major:"1", Minor:"23", GitVersion:"v1.23.0", GitCommit:"ab69524f795c42094a6630298ff53f3c3ebab7f4", GitTreeState:"clean", BuildDate:"2021-12-07T18:16:20Z", GoVersion:"go1.17.3", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.0", GitCommit:"c92036820499fedefec0f847e2054d824aea6cd1", GitTreeState:"clean", BuildDate:"2021-10-27T18:35:25Z", GoVersion:"go1.18.9", Compiler:"gc", Platform:"linux/amd64"}
```

## → Kubectl get nodes

Is command sein humko ye pata chal jayega ki humare pass kitane worker nodes hai joki master mein attached hai , hum jab ye command chalayenge toh usme roles mein ye bhi dekh sakte hai ki humari jo node hai vo kisase connect hai ,

- uske andar eak ip hogi jo ki humare instance ki ip hogi.
- Status pata chalega ki vo ready hai ya nahi
- Mere instance ki private ip aur worker node ki private ip same hai kyu ki mein eak hi instance mein andar master node aur private node dono bana rakha hu.



## → kubectl describe node ip-172-31-12-153

- Agar mujhe node ki aur jankaari chahiye toh uske liye humko ye command chalani hogi mein node ki private ip daal kar us node ki full details le sakta hu.

```
root@ip-172-31-12-153:/home/ubuntu# kubectl describe node ip-172-31-12-153
Name:           ip-172-31-12-153
Roles:          control-plane,master
Labels:         beta.kubernetes.io/arch=amd64
               beta.kubernetes.io/os=linux
               kubernetes.io/arch=amd64
               kubernetes.io/hostname=ip-172-31-12-153
               kubernetes.io/os=linux
Annotations:   kubelet.alpha.kubernetes.io/ttl: 0
               volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp: Thu, 09 Dec 2021 12:28:24 +0000
Taints:          (none)
Unschedulable:  false
Leases:
  HolderIdentity: ip-172-31-12-153
  AcquiredTime:   Thu, 09 Dec 2021 06:58:21 +0000
  RenewedTime:    Fri, 10 Dec 2021 06:58:21 +0000
Conditions:
  Type        Status  LastHeartbeatTime     LastTransitionTime   Reason          Message
  ----        ----  ---------------------  ---------------------  -----          -----
  MemoryPressure  False   10 Dec 2021 06:57:45 +0000  Thu, 09 Dec 2021 12:28:22 +0000  KubeletHasSufficientMemory  kubelet has sufficient memory available
  DiskPressure   False   10 Dec 2021 06:57:45 +0000  Thu, 09 Dec 2021 12:28:22 +0000  KubeletHasNoDiskPressure  kubelet has no disk pressure
  Ready         True    Fri, 10 Dec 2021 06:57:45 +0000  Thu, 09 Dec 2021 12:28:22 +0000  KubeletHasSufficientPID  kubelet has sufficient PID
  Ready         True    Fri, 10 Dec 2021 06:57:45 +0000  Thu, 09 Dec 2021 12:28:38 +0000  KubeletReady            kubelet is posting ready status. AppArmor enabled
Addressess:
  InternalIP:  172.31.12.153
  Hostname:   ip-172-31-12-153
Capacity:
  ephemeral-storage:  2
  ephemeral-storage-2Mi:  886544Ki
  hostmemory-2Mi:    4023188Ki
  memory:        4023188Ki
  memory-1Mi:    110
Allocatable:
  (none)
Allocatable:
  cpu:          2
  ephemeral-storage:  886544Ki
  hostmemory-2Mi:    4023188Ki
  memory:        4023188Ki
  pods:          110
System Info:
  Machine ID:      9f2730954865481c940ab993923b2ad27
  OS Image:        opensuse-leap-15.2-x86_64-20211209-n02
  Boot ID:         c9b76eef-55ee-4394-9830-9074e453daed
  Kernel Version:  5.11.0-1026-aws
  OS Version:      Linux ip-172-31-12-153 5.11.0-1026-aws
  Operating System: linux
  Architecture:   x86_64
  Container Runtime Version: docker://20.10.7
  Kubelet Version: v1.22.3
  KubeProxy Version: v1.22.3
  PodCIDR:         10.244.0.0/24
  Non-terminated Pods: (7 in total)
    Namespace   Name
    kube-system coredns-78fc609978-vcdqx
    kube-system kube-dns-744141515k
    kube-system kube-apiserver-ip-172-31-12-153
    kube-system kube-controller-manager-ip-172-31-12-153
    kube-system kube-scheduler-ip-172-31-12-153
    kube-system kube-state-metrics-54445-5q7t5
    kube-system storage-provisioner
  Allocated resources:
    (Total limits may be over provisioned, i.e., overcommitted.)
  Resource   Requests   Limits
  CPU        100m (5%)  0 (0%)
  memory    760Mi (1%)  100Mi (2%)
  ephemeral-storage  0 (0%)  0 (0%)
  hostmemory-2Mi  0 (0%)  0 (0%)
  Events:    <none>
root@ip-172-31-12-153:/home/ubuntu#
```

## → Now we create a manifest file

→ vi pod1.yml

```
kind: Pod
apiVersion: v1
metadata:
  name: testpod
spec:
  containers:
    - name: c00
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Hello-Satyam; sleep 5 ; done"]
restartPolicy: Never    # Defaults to Always
```

```
root@ip-172-31-12-153: /home/ubuntu
kind: Pod
apiVersion: v1
metadata:
  name: testpod
spec:
  containers:
    - name: c00
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Hello-Satyam; sleep 5 ; done"]
      restartPolicy: Never    # Defaults to Always
```

➤ Hum yml file likhate samaye humko indentation ka bahut dhyan dena padta hai.

### ➤ Kind : pod

(iska matlab humara object pod hai hum pod create kar rahe)

### ➤ apiVersion: v1

humare pod ki api version mene de rakha hai version 1 jadatar hum likhate hai by default.

### ➤ metadata:

#### name: testpod

metadata mein agar hum koi naam dena chahahte hai apane pod ka toh hum uska naam de akte hai jaise ki mene apane pod ka naam testpod diya hai.

➤ spec:

containers:

- name: c00

image: ubuntu

command: ["/bin/bash", "-c", "while true; do echo Hello-Satyam; sleep 5 ; done"]

- spec matlab specification aur specification mein andar mene container banaya hai
- container mein andar mene ek group banaya hai jiske andar mene container mein baare mein bataya hai
- mene container ko start karane mein pehale hi – **lagaya hai**, aur agar mujhe ek pod mein andar multiple container banana hai toh hum - **laga kar ek naya group bana** kar naye container ki detail daal sakte hai .
- - **lagane** mein ye clear ho jayega ki ab dusare container ko banana ki baat ho rahi hai , aise hum kitne bhi – laga kar container bana sakte hai
- - **name: c00** mene apne container ka naam c00 diya hai
- **command: ["/bin/bash", "-c", "while true; do echo Hello-Satyam; sleep 5 ; done"]**  
agar mera container chal jata hai matlab true hota hai toh hello satyam run kardena
- uske baad hum wq likh kar file ko save kar denge.



```

→ kubectl apply -f pod1.yml
O/P → pod/testpod Created
→ kubectl get pods
O/P → testpod 1/1 Running
if you want to see, where exactly pod is running
→ kubectl get pods -o wide
→ kubectl describe pod testpod
or
kubectl describe pod/testpod
→ kubectl logs -f testpod
if you want to delete pod
→ kubectl delete pod testpod

```

## → kubectl apply -f pod1.yml

**output :- pod/testpod created**

( hum jaise hi kubectl vali command chalayenge vaise hi humari pod1.yml vali file chal jayegi aur run bhi ho jayegi, aur vo eak testpod naam sein pod create kar dega.)

```

root@ip-172-31-12-153:/home/ubuntu# kubectl apply -f pod1.yml
pod/testpod created
root@ip-172-31-12-153:/home/ubuntu#

```

## → kubectl get pods

**output :- testpod 1/1 running**

➤ ( humare pass node mein andar kitane pod hai vo dikh jayega, aur us pod mein andar kitane container hai vo bhi dikh jayega aur kitane container run ho rahe vo bhi dikh jayega.)

➤ Hum dekh sakte hai humare pass total eak hi container aur jo eak container humare pass hai vo run bhi ho raha, kyuki status running bata raha .

```

root@ip-172-31-12-153:/home/ubuntu# kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
testpod   1/1     Running   0          18m
root@ip-172-31-12-153:/home/ubuntu#

```

## → kubectl get pods -o wide

( suppose humare pass eak master hai aur us master mein 3 worker node connected hai, humare eak pod banaya aur aur hum ye check karna chahate hai ki humare pod kaun mein worker node mein andar jaakar bana hai toh uske liye hum ye command use karenge. ( wide mein humko humare pod ki exact detail pata chal jayegi. Ki vo kis instance par hai aur uss instance ki ip kya hai toh usase humko pata chal jayega ki vo kaun mein insatance par bana huva hai.)

```

root@ip-172-31-12-153:/home/ubuntu# kubectl get pods -o wide
NAME      READY   STATUS    RESTARTS   AGE   IP           NODE   NOMINATED NODE   READINESS GATES
testpod   1/1     Running   0          36m   172.17.0.3   ip-172-31-12-153   <none>        <none>
root@ip-172-31-12-153:/home/ubuntu#

```

- isme 2 ip aa rahi hai toh isme mein jo node mein neeche vali ip diya gaya hai vo mere ec2 instance ki private ip hai aur jo dusri IP hai vo meri pod ki ip hai .
- aur ye dono ip private ip hoti hai , matlab aisa ki jo humare worker node mein instance ki private ip hai vahi mere pod mein private ip sein jaake baat karega
- aur humare instance ki private ip humare pod ko bhi eak private ip dedega jisase instance ki private ip aur pod ki private ip dono aapas mein baat kar sake .
- Kabhi bhi container ki ip nahi hoti ip hardum pod ki hoti hai .

```
→ kubectl describe pod testpod  
→ kubectl describe pod/testpod
```

(agar hum apne pod mein related sab details dekhna chahate hai ki kya kya pod mein hua hai kab kya kya cheej install hua, pehle kya install hua uske baad kya install hua ye sab cheeje humko is command mein seinkar sakte hai same matlab hai.)

```
root@ip-172-31-12-153:/home/ubuntu# kubectl describe pod testpod
Name:           testpod
Namespace:      default
Priority:      0
Node:          ip-172-31-12-153/172.31.12.153
Start Time:    Fri, 10 Dec 2021 07:44:28 +0000
Labels:        <none>
Annotations:   <none>
Status:        Running
IP:            172.17.0.3
IPs:
  IP: 172.17.0.3
Containers:
  c00:
    Container ID: docker://c45cbbecedaa4c4478e152478c97db43ed46d53af4732cecea0c73ffd1ce52d5
    Image:         ubuntu
    Image ID:     docker-pullable://ubuntu@sha256:626ffe58f6e00254b638eb7e0f3b11d4da9675088f4781a50ae288f3322
    Port:          <none>
    Host Port:    <none>
    Command:
      /bin/bash
      c
      while true; do echo Hello-Satyam; sleep 5 ; done
    State:        Running
      Started:    Fri, 10 Dec 2021 07:44:36 +0000
    Ready:        True
    Restart Count: 0
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-919tn (ro)
Conditions:
  Type        Status
  Initialized  True
  Ready       True
  ContainersReady  True
  PodScheduled  True
Volumes:
  kube-api-access-919tn:
    Type:        Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName:  kube-root-ca.crt
    ConfigMapOptional: <nil>
    DownwardAPI:   true
Volumes:
  kube-api-access-919tn:
    Type:        Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName:  kube-root-ca.crt
    ConfigMapOptional: <nil>
    DownwardAPI:   true
QoS Class:      BestEffort
Node-Selectors:  <none>
Tolerations:    node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                  node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:        <none>
root@ip-172-31-12-153:/home/ubuntu#
```

→ kubectl logs -f testpod

(is command sein humko ye pata chal jayega ki humare pod mein andar jo container hai usme kya chal raha, matlab ki container mein kya kya run ho raha hai aur ye bhi bata dega ki kaun kaun sa container running hai. Aur agar maine ki mere pass multiple container hai aur mujhe kewal kisi ek container ki details chahiye ya phir uske logs dekhana ha toh hum ye command use karenge

Ctrl + z press karke hum bahar aa sakte hai

➔ **kubectl logs -f testpod -c c01**

- agar humare pass eak sein jaad container hai eak hi pod mein aur humko kisi eak particular container ki details ya log check karne ho toh uske liye hum ye command use karenge. Hum pod mein baad us container ka naam likh denge toh humko us container mein log details pata chal jayega
- mene niche pics mein container ka naam galat bataya hai container ka isliye details nahi aayi agar container ka naam c00 likhata toh **hello satyam** print kar deta kyu ki c00 naam sein jo container hai uske andar hello satyam hi details likh rakhi hai.

```
root@ip-172-31-12-153:/home/ubuntu
```

```
root@ip-172-31-12-153:/home/ubuntu# kubectl logs -f testpod -c c01
error: container c01 is not valid for pod testpod
root@ip-172-31-12-153:/home/ubuntu#
```

➔ **kubectl delete pod testpod**

( agar mujhe apane pod delete karna hai toh hum ye command use karenge, pod mein baad pod ka name)

```
root@ip-172-31-12-153:/home/ubuntu# kubectl delete pod testpod
pod "testpod" deleted
root@ip-172-31-12-153:/home/ubuntu#
```

Pod deleted

```
root@ip-172-31-12-153:/home/ubuntu# kubectl get pods
No resources found in default namespace.
root@ip-172-31-12-153:/home/ubuntu#
```

No pods available

➔ **kubelet delete -f pod1.yml**

(agar mujhe apani yml file hi delete karni ho jisase mene pod banaya tha toh uske liye hum ye command use karenge)

---

## Annotation

→ nano pod1.yml

### Annotations :-

( Annotation hum jaha metadata likhate hai apane yml file mein uske neeche annotations mein andar description mein anadar kuch bhi as a comment likh sakte hai agar kisi ko mein apani yml file de raha toh vo annotation pdh kar sakjha jayega us file mein baare mein.)

Annotations hota hai na ki annotation, s likhana jaruri hai

```
kind: Pod
apiVersion: v1
metadata:
  name: testpod
  annotations:
    description: humara pehala testpod create karne jaa raha hu
spec:
  containers:
    - name: c00
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Hello-Satyam; sleep 5 ; done"]
  restartPolicy: Never    # Defaults to Always
```

```
root@ip-172-31-12-153:/home/ubuntu# Select root@ip-172-31-12-153: /home/ubuntu
kind: Pod
apiVersion: v1
metadata:
  name: testpod
  annotations:
    description: humara pehala testpod create karne jaa raha hu
spec:
  containers:
    - name: c00
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Hello-Satyam; sleep 5 ; done"]
  restartPolicy: Never    # Defaults to Always
~
~
~
```

→ kubectl apply -f pod1.yml

( hum jab bhi apane yml file mein kuch edit karenge toh us edit ko apply karne mein liye humko ye command chalani paegi.)

```
root@ip-172-31-12-153:/home/ubuntu# kubectl apply -f pod1.yml
pod/testpod created
root@ip-172-31-12-153:/home/ubuntu#
```

## → kubectl describe pod testpod

( hum jab ye command chalayenge toh humare pod mein andar jo bhi annotation mene diya hoga vo mujhe pata chal jayega ki ye pod mene kisliye banaya tha, ya toh phir iss pod kya kaam hai ab aage is pod ka use kaha hum karenge sab pata chal jayega annotation mein jo bhi hum apane yaml file mein andr annotation mein andar likhe honge likhe honge iske liye humko yaml file kholane ki jarurat nahi padegi.)

```
Select root@ip-172-31-12-153:/home/ubuntu
root@ip-172-31-12-153:~/home/ubuntu# root@ip-172-31-12-153:~/home/ubuntu#
root@ip-172-31-12-153:~/home/ubuntu# kubectl apply -f pod.yaml
pod/testpod created
root@ip-172-31-12-153:~/home/ubuntu# kubectl describe pod testpod
Name:           testpod
Namespace:      default
Priority:       500
Node:          ip-172-31-12-153/172.31.12.153
Start Time:    Sat, 11 Dec 2021 15:17:41 +0000
Labels:         <none>
Annotations:   description: humara pehla testpod create karne jaa raha hu
               IP: 172.17.0.3
               IP: 172.17.0.3
Containers:
  Container 0:
    Container ID: docker://04dc2bf59dc3ab0a9ce54da68c6809cbbaa80034b1223967e3eb387778deaf73
    Image:         ubuntu
    Image ID:     docker://ubuntu:20.04-20211201-1455-0
    Port:          <none>
    Host Port:    <none>
    Command:
      /bin/bash
      while true; do echo Hello-Satyam; sleep 5 ; done
    State:        Started:   Sat, 11 Dec 2021 15:17:44 +0000
                  Ready:      True
                  Running:   0
                  Environment: <none>
                  Mounts:    /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-rc9tz (ro)
    Conditions:
      Type        Status
      Initialized  True
      Ready       True
      ContainersReady  True
      PodScheduled True
    Volumes:
      kube-api-access-rc9tz:
        Type:      projected (a volume that contains injected data from multiple sources)
        TokenExpirationSeconds: 3607
        ConfigMapName:   kube-root-ca.crt
        ConfigMapNamespace: default
        DownwardAPI:    true
        PodClassName:  kube-namespace
        PodAffinity:   <none>
        PodAntiAffinity: <none>
        PodTolerations: node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                        node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
    Events:
      Type  Reason  Age   From            Message
      ----  -----  --   --              --
      Normal  Scheduled  53s  default-scheduler  Successfully assigned default/testpod to ip-172-31-12-153
      Normal  Pulling  53s  kubelet        Pulling image: ubuntu
```

## MULTI CONTAINER POD ENVIRONMENT

### ➤ nano pod2.yml

```
kind: Pod
apiVersion: v1
metadata:
  name: testpod3
spec:
  containers:
    - name: c00
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Satyam - India; sleep 5 ; done"]
    - name: c01
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Hello - india; sleep 5 ; done"]
```

```
>Select root@ip-172-31-12-153:/home/ubuntu
GNU nano 4.8
kind: Pod
apiVersion: v1
metadata:
  name: testpod3
spec:
  containers:
    - name: c00
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Satyam - India; sleep 5 ; done"]
    - name: c01
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Hello - india; sleep 5 ; done"]
```

- Hum 1 pod mein multiple container banayenge aur 1<sup>st</sup> container ka name mere **c00** hai aur 2nd container ka name **c01** hai .
- Hum yml mein containers mein andar parent mein – laga kar new container ki details daal sakte hai .

### ➤ kubectl apply -f pod2.yml

```
root@ip-172-31-12-153:/home/ubuntu# kubectl apply -f pod2.yml
pod/testpod3 created
root@ip-172-31-12-153:/home/ubuntu#
```

### ➤ kubectl get pods

```
root@ip-172-31-12-153:/home/ubuntu# kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
testpod   1/1     Running   0          42h
testpod3  2/2     Running   0          70s
root@ip-172-31-12-153:/home/ubuntu#
```

mene testpod 3 name mein pod banaya aur uske andar mene 2 container banaaye thee toh hum dekh sakte hai ki humare dono container running mein hai aur dono container idhar dikh rahe hai.

- Agar mujhe apne pod mein 2 ki jagah 3 container banana ho toh mein again usi yml file mein jaunga aur eak container aur bana dunga

For example :-

```
root@ip-172-31-12-153:/home/ubuntu
  GNU nano 4.8
kind: Pod
apiVersion: v1
metadata:
  name: testpod3
spec:
  containers:
    - name: c00
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Satyam - India; sleep 5 ; done"]
    - name: c01
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Hello - india; sleep 5 ; done"]
    - name: c02
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo welcome _- india; sleep 5 ; done"]
```

- **kubectl delete pod testpod3**

```
root@ip-172-31-12-153:/home/ubuntu# kubectl delete pod testpod3
pod "testpod3" deleted
root@ip-172-31-12-153:/home/ubuntu#
```

Hum pehale testpod3 name vale pod ko delete karenge jo pehale bana tha, kyu ki running pod mein agar naya container bana rahe toh vo error dega toh naya container add karne mein liye humko pod delete karne padega uske baad apani pod2.yml vali file mein container add karenge and save karke yml phir run karenge matlab apply karadenge.

- **kubectl apply -f pod2.yml**

```
root@ip-172-31-12-153:/home/ubuntu# kubectl apply -f pod2.yml
pod/testpod3 created
```

- **kubectl get pods**

```
root@ip-172-31-12-153:/home/ubuntu# kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
testpod   1/1     Running   0          43h
testpod3  3/3     Running   0          12s
root@ip-172-31-12-153:/home/ubuntu#
```

Hum dekhenge jab mene apani yml file mein edit karke new container banaya toh mere pod mein 3 running container hogaye hai ab .

- **kubectl describe pod testpod3**

```
root@ip-172-31-12-153:/home/ubuntu# kubectl describe pod testpod3
Name:           testpod3
Namespace:      default
Priority:       0 (x1 high)
Node:          ip-172-31-12-153/172.31.12.153
Labels:         <none>
Annotations:    <none>
Status:        Running
IP:            172.31.0.4
IPv6:          <none>
Containers:
  c00:
    Container ID: docker://73db20bb0a525c298e4d6b1abc1345fea5d18fc037b9200a7bd1f39f8cd2b7fc
    Image:          docker-pullable://ubuntu@sha256:626fffe58f6ce7560ee00254bc3Reb2e0f3b11d4da9675088f4781a50ae288f3322
    ImageID:       sha256:626fffe58f6ce7560ee00254bc3Reb2e0f3b11d4da9675088f4781a50ae288f3322
    Port:          <none>
    Host Port:    <none>
    Command:
      /bin/bash
    Args:
      while true; do echo Satyam - India; sleep 5 ; done
    State:          Running
      Started:      Tue, 13 Dec 2021 10:40:05 +0000
    Ready:          True
    Reasons:       <none>
    Environment:   <none>
    Ports:         <none>
    Links:         <none>
    Container ID: docker://a9e0dab4db97d50fcaaa7ab72512b5df562b7311a08ef8735eac7706ea27ca6
    Image ID:      docker-pullable://ubuntu@sha256:626fffe58f6ce7560ee00254bc3Reb2e0f3b11d4da9675088f4781a50ae288f3322
    Port:          <none>
    Host Port:    <none>
    Command:
      /bin/bash
    Args:
      while true; do echo Hello - india; sleep 5 ; done
    State:          Running
      Started:      Tue, 13 Dec 2021 10:40:07 +0000
    Ready:          True
    Reasons:       <none>
    Environment:   <none>
    Ports:         <none>
    Links:         <none>
    Container ID: docker://5c2b8fb2a18712620dd8d2dfr7853dd87c45d6bb3b22a13d12909falte8114068
    Image ID:      docker-pullable://ubuntu@sha256:626fffe58f6ce7560ee00254bc3Reb2e0f3b11d4da9675088f4781a50ae288f3322
    Port:          <none>
    Host Port:    <none>
```

```
Select root@ip-172-31-12-153:/home/ubuntu
Container ID: docker://6c2b8f62a18712020d8cd2df87853dd87c45d6bb3b22413d12909fa8e81140b8
Image ID: docker-pullable://ubuntu@sha256:626ffe58f6e7566e00254b638eb7e0f3b11d4da9675088f4781a50ae288f3322
Port: <none>
Host Port: <none>
Command:
/bin/bash
-c
while true; do echo welcome - india; sleep 5 ; done
State: Running
Started: Mon, 13 Dec 2021 10:40:10 +0000
Ready: True
Restart Count: 0
Environment: <none>
Mounts:
/var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-s49j5 (ro)
Conditions:
Type Status
Initialized True
Ready True
ContainersReady True
PodScheduled True
Volumes:
kube-api-access-s49j5:
  Type:      PersistentVolumeClaim   Status:      Projected (a volume that contains injected data from multiple sources)
  TokenExpirationSeconds: 3607
  ConfigMapName:    kube-root-ca.crt
  ConfigMapOptional: <nil>
  DownwardAPI:     true
QoS Class: BestEffort
Node-Selectors: <none>
Tolerations: node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
          node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
Type Reason Age From Message
---- ---- -- -- -----
Normal Scheduled 11m default-scheduler Successfully assigned default/testpod3 to ip-172-31-12-153
Normal Pulling 11m kubelet Pulling image "ubuntu"
Normal Pulled 11m kubelet Successfully pulled image "ubuntu" in 2.30772449s
Normal Created 11m kubelet Created container c00
Normal Started 11m kubelet Started container c00
Normal Pulling 11m kubelet Pulling image "ubuntu"
Normal Pulled 11m kubelet Successfully pulled image "ubuntu" in 2.322410142s
Normal Created 11m kubelet Created container c01
Normal Started 11m kubelet Started container c01
Normal Pulling 11m kubelet Pulling image "ubuntu"
Normal Pulled 11m kubelet Successfully pulled image "ubuntu" in 2.294017277s
Normal Created 11m kubelet Created container c02
Normal Started 11m kubelet Started container c02
root@ip-172-31-12-153:/home/ubuntu#
```

Hum events mein dekh sakte hai ki pehale kaise –kaise kya kya isane banaya hai matlab pehale kaun sa container phir uske baad kaun sa container.

➤ **kubectl logs -f testpod3 -c c00**

Agar mujhe apane kisi containers mein log check karne hai toh uske liye hum ye command use karenge -c mein baad container name likh denge toh mujhe us container mein andar jo bhi logs honge vo sab dijk jayenge

## ➤ Container 2 logs

## ➤ Container3 logs



#### ➤ **kubectl exec testpod3 -c c00 -- hostname -i**

agar mujhe ye pata karna ho ki mera container jis pod mein andar rakha hai us pod ki IP kya hai toh hum ye command likh kar pata kar sakte hai, aur dusari baat ye hai ki kabhi bhi humare **container ki IP nahi hoti hardum pod ki ip hoti hai** isliye mene neeche 3 command chalai hai sab mein container name change hai but **sab mein same ip hai** aisa isliye kyuki jitane bhi container hai ye sab eak hi pod mein rakhe hai .

```
root@ip-172-31-12-153:/home/ubuntu# kubectl exec testpod3 -c c00 -- hostname -i  
172.17.0.4
```

```
root@ip-172-31-12-153:/home/ubuntu# kubectl exec testpod3 -c c01 -- hostname -i  
172.17.0.4
```

```
root@ip-172-31-12-153:/home/ubuntu# kubectl exec testpod3 -c c02 -- hostname -i  
172.17.0.4
```

#### ➤ **kubectl exec testpod3 -it -c c00 -- /bin/bash**

- > agar mujhe pod mein andar jaake kisi container mein andar kaam karna ho toh uske liye hum ye command use karenge.
- > (-it matlab interactive mode) kehate hai, aur phir uske baad mene -c matlab container vale directory mein jao aur c00 naam ka jo container hai uske andar mujhe entry kara do

root@testpod3: /

```
root@ip-172-31-12-153:/home/ubuntu# kubectl exec testpod3 -it -c c00 -- /bin/bash  
root@testpod3:#
```

#### ➤ **ps -ef**

- > Ab mujhe container mein andar ki details dekhni hai ki us container mein andar kaun sein logs chal rahe toh hum is command mein dekh sakte hai.
- > running container mein bahar ane mein liye hum exit command chalayenge .

```
root@testpod3:# ps -ef  
UID      PID  PPID   C STIME TTY          TIME CMD  
root        1      0  0 Dec13 ?        00:00:05 /bin/bash -c while true; do echo Satyam - India; sleep 5 ; done  
root    14410      0  0 06:39 pts/0    00:00:00 /bin/bash  
root    14819      1  0 07:13 ?        00:00:00 sleep 5  
root    14820  14410  0 07:13 pts/0    00:00:00 ps -ef  
root@testpod3:#
```

#### ➤ **kubectl delete -f pod2.yml**

- > hum dekh sakte hai mene pod2.yml chalaya hai but humara testpod3 delete huva aisa isliye kyuki humne yml file mein pod aka naam testpod3 de rakha hai.

```
root@ip-172-31-12-153:/home/ubuntu# kubectl delete -f pod2.yml  
pod "testpod3" deleted  
root@ip-172-31-12-153:/home/ubuntu#
```



## Environmental variable in pod

### ➔ nano pod3.yml

```
kind: Pod
apiVersion: v1
metadata:
  name: giit
spec:
  containers:
    - name: c00
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Hello-kubernetes; sleep 5 ; done"]
      env:           # List of environment variables to be used inside the pod
        - name: MYNAME
          value: Satyam
```

```
root@ip-172-31-12-153: /home/ubuntu
  GNU nano 4.8
kind: Pod
apiVersion: v1
metadata:
  name: giit
spec:
  containers:
    - name: c00
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Hello-kubernetes; sleep 5 ; done"]
      env:           # List of environment variables to be used inside the pod
        - name: MYNAME
          value: Satyam
```

- Agar apne pod mein andar kuch predefined **key value** pair define karna chahate hai toh usko hum environment variable mein andar key pair value mein bata sakte hai.
- For example :- mene ek pod ka naam satyam de rakha hai key pair value mein aur jab-jab mein satyam kahi call kar raha hunga toh vo pod hi automatically call hoga , jis pod ki value mene satyam de rakhi hai.
- Agar env karke kuch bhi hum apana yml file mein likhnge toh iska matlab ye hai ki vo environment declaration ki baat kar raha kisi cheej ka.
- Meri env **key=MYNAME** hai aur iski **value=Satyam** hai .
- Env variable ko search karne mein liye humko **\$** ka sign lagana padta hai.

### ➔ Kubectl apply -f pod3.yml

- Isko chalane mein mera pod create ho jayega mere container create ho jayega jo jo cheej ehumane yml mein likhi hogi sab start ho jayegi.

```
root@ip-172-31-12-153:/home/ubuntu# kubectl apply -f pod3.yml
pod/giit created
root@ip-172-31-12-153:/home/ubuntu#
```

### → Kubectl get pods

```
root@ip-172-31-12-153:/home/ubuntu# kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
giit      1/1     Running   0          39s
testpod   1/1     Running   0          2d19h
root@ip-172-31-12-153:/home/ubuntu#
```

### → kubectl exec giit -it -- /bin/bash

```
root@ip-172-31-12-153:/home/ubuntu# kubectl exec giit -it -- /bin/bash
root@giit:/#
```

### → root@giit:/# env

mujhe iss command sein dikh jayega ki mera kubernetes ka aport kaun sa hai jaise ki mera port hai 443 aur mujhe private ip bhi dikh rahi , baki environment sein related bahut cheeje dikh rahi hai .

```
root@giit:/# env
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_SERVICE_PORT=443
MYNAME=giit
HOSTNAME=giit
PWD=/
HOME=/root
KUBERNETES_PORT_442_TCP=tcp://10.96.0.1:443
$COLORS=rx-dH-01;24;ln-01;36;wh-00;pi-00;22;co-01;35;do-01;25;bd-00;33;01;cd-00;40;22;01;or-00;31;01;mi-00;su-01;37;41;sg-00;42;ca-00;41;ts-00;42;ov-01;32;*;tar-01;31;*;tgr-01;31;*;arc-01;31;*l-01;31;*;txo-01;31;*;lha-01;31;*;lzd-01;31;*;l2h-01;31;*;lzo-01;31;*;txz-01;31;*;tzo-01;31;*;tx-01;31;*;z-01;31;*;dz-01;31;*;gx-01;31;*;lxr-01;31;*;lx-01;31;*;xz-01;31;*;cpio-01;31;*;tzt-01;31;*;bz-01;31;*;bz2-01;31;*;bz-01;31;*;deb-01;31;*;rpm-01;31;*;jar-01;31;*;war-01;31;*;ear-01;31;*;sar-01;31;*;ar-01;31;*;ace-01;31;*;zoo-01;31;*;cipio-01;31;*;7z-01;31;*;rz-01;31;*;cab-01;31;*;win-01;31;*;sum-01;31;*;shm-01;31;*;esd-01;31;*;jpg-01;35;*;jpeg-01;35;*;mpj-01;35;*;mjpeg-01;35;*;gif-01;35;*;bmp-01;35;*;pbm-01;35;*;pgm-01;35;*;ppm-01;35;*;pnm-01;35;*;tga-01;35;*;mp4v-01;35;*;vob-01;35;*;svf-01;35;*;png-01;35;*;webp-01;35;*;open-01;35;*;amf-01;35;*;mp4-01;35;*;emf-01;35;*;vob-01;35;*;nra-01;35;*;swv-01;35;*;wmv-01;35;*;avi-01;35;*;flv-01;35;*;l1v-01;35;*;gl-01;35;*;x11-01;35;*;xci-01;35;*;xad-01;35;*;yuv-01;35;*;psm-01;35;*;xpm-01;35;*;xsp-01;36;*;xspf-00;36;
```

TERM=xterm

SHELL=/bin/sh

KUBERNETES\_PORT\_443\_TCP\_PROTO=tcp

KUBERNETES\_PORT\_443\_TCP\_ADDR=10.96.0.1

KUBERNETES\_PORT=tcp://10.96.0.1:443

KUBERNETES\_PORT\_443\_TCP\_PORT=443

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

/usr/bin/env

root@giit:/#

### → echo \$MYNAME

Mene jaise hi apane key dalli key dalne mein baad mene jo us key ki ko value dali thi vo mujhe mil gayi.

Key hardum dalte samaye pehale \$ lagate hai .

```
root@giit:/# echo $MYNAME
```

```
Satyam
```

```
root@giit:/#
```

### → kubectl delete -f pod3.yml

```
root@ip-172-31-12-153:/home/ubuntu# kubectl delete -f pod3.yml
pod "giit" deleted
root@ip-172-31-12-153:/home/ubuntu#
```

=====

## POD WITH PORTS

### ➔ Nano pod4.yml

Mene iss baar container image httpd choose kiya hai yml file mein, aur **httpd port 80** par chalta hai ., isliye mene container **port 80** bhi expose kar rakha hai .

```
root@ip-172-31-12-153:/home/ubuntu
```

```
GNU nano 4.8
kind: Pod
apiVersion: v1
metadata:
  name: testpod4
spec:
  containers:
    - name: c00
      image: httpd
      ports:
        - containerPort: 80
```

```
kind: Pod
apiVersion: v1
metadata:
  name: testpod4
spec:
  containers:
    - name: c00
      image: httpd
      ports:
        - containerPort: 80
```

### ➔ kubectl apply -f pod4.yml

```
root@ip-172-31-12-153:/home/ubuntu# kubectl apply -f pod4.yml
pod/testpod4 created
root@ip-172-31-12-153:/home/ubuntu#
```

### ➔ kubectl get pods -o wide

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
testpod	1/1	Running	0	2d20h	172.17.0.3	ip-172-31-12-153	<none>	<none>
testpod4	1/1	Running	0	46m	172.17.0.4	ip-172-31-12-153	<none>	<none>

Iss command sein humko pod ki aur instance ki dono ki ip dikh jayegi

### ➔ curl 172.17.0.4:80

- Mene apane pod ki ip kein dalane mein baad :80 likha hu dekhana chahata hu ki mera pod jo hai vo port 80 sein connect hai ya nahi
- Aur hum dekh sakte hai image mein output mein likh kar aa raha **IT works !** iska matlab humara pod port 80 sein connect hai .
- Hum yml mein andar jis bhi port ko allow karenge vahi port jaakar connect hogा
- Curl command sein hum check karte hai ki humara pod connected hai ya nahi port sein .

```
root@ip-172-31-12-153:/home/ubuntu# curl 172.17.0.4:80
<html><body><h1>It works!</h1></body></html>
root@ip-172-31-12-153:/home/ubuntu#
```

