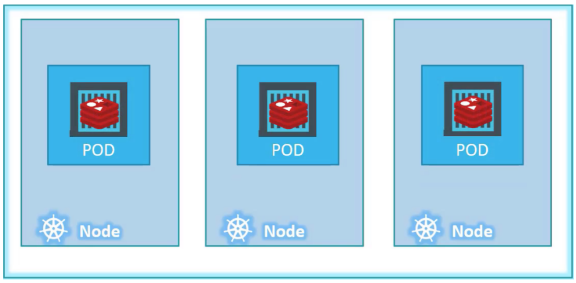
# Pods

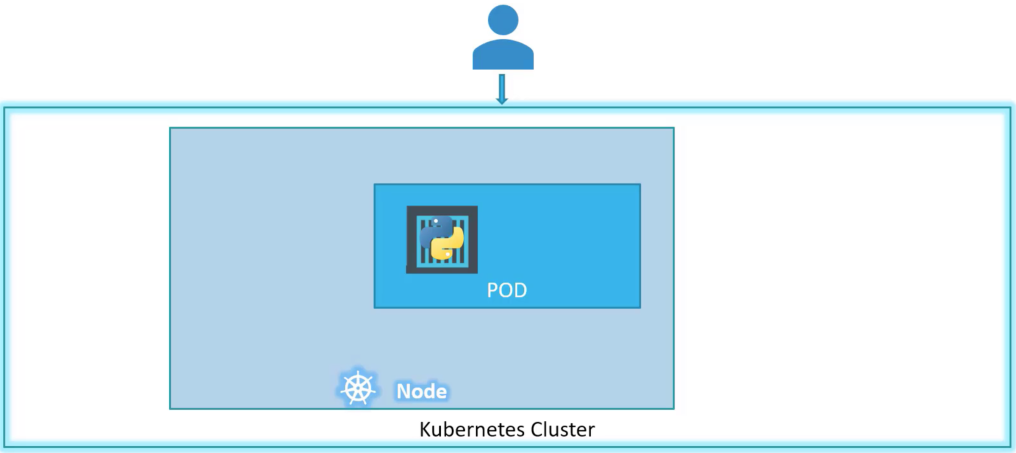
## PODS

- Before we get into PODs, lets try to assume the following have been setup already.
- At this point, lets assume the application is already developed and built into docker images and it is available in the docker repostitory (Docker hub), Hence Kubernetes can pull it down.
- Lets also assume Kubernetes cluster is already been setup and working.
- As we discussed before with help of Kubernetes our ultimate aim is to deploy applications in form of containers on a set of machines. [Which are configured as worker nodes in a cluster]
- However Kubernetes is not going to deploy containers directly in worker nodes.
- Instead, containers are going to be encapsulated into the Kubernetes object called PODs.
- PODs is the smallest object which you can create in Kubernetes.

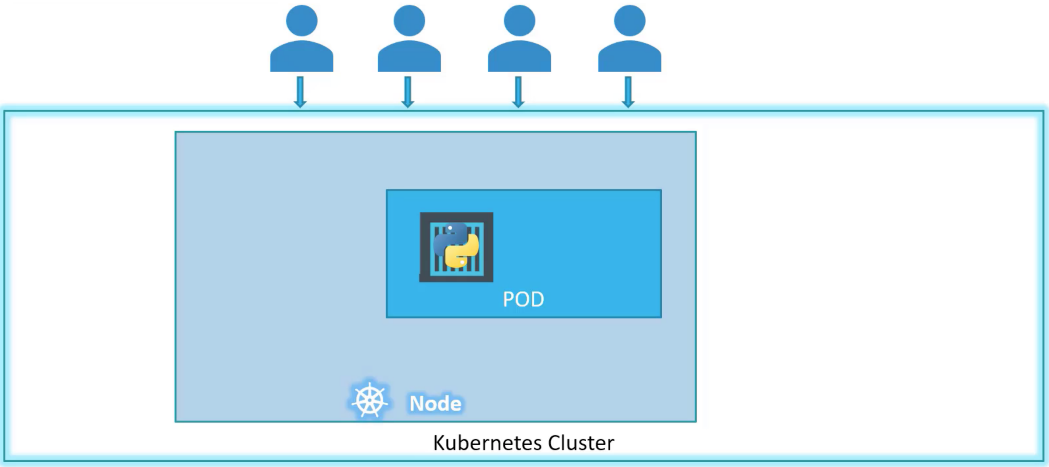


## Lets try to understand this more in details

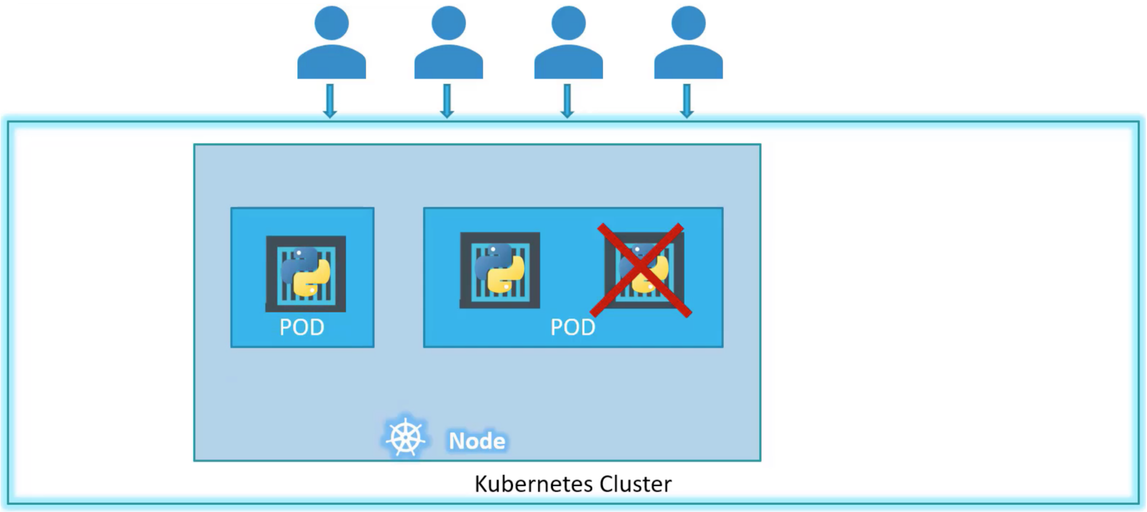
- Here we see, we have single node Kubernetes cluster with single container contains your application which is running inside Pod.



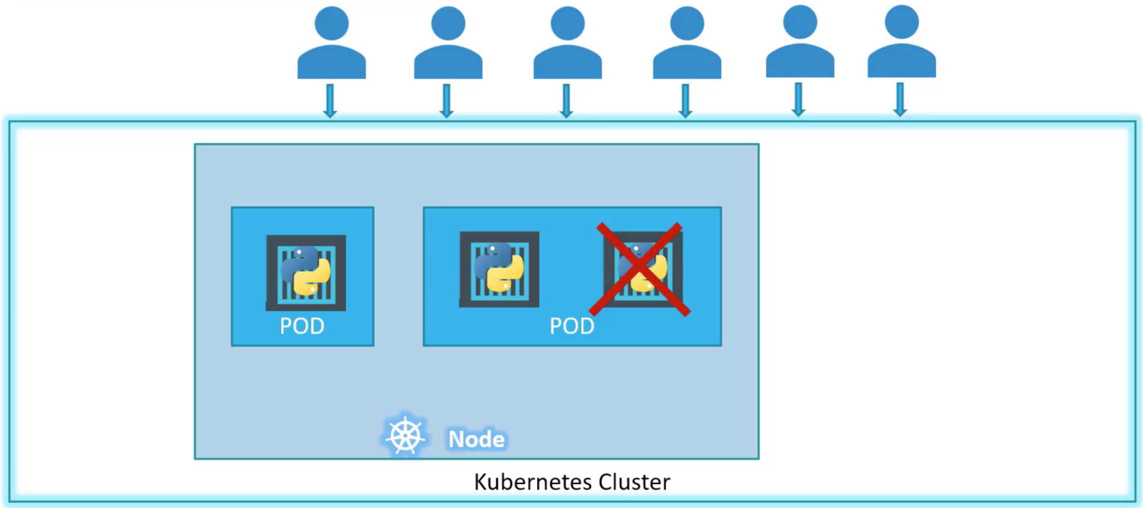
- Right now the number users accessing your application is one, What if the number of user accessing your application is getting increased ?



- In this case you may need to scale your application to server the demand.
- You may need to create additional instances to share the workload.
- Now where would your spin up additional instances (containers).
- Do we bring new instances within the same PODs ?
- No we are going to create new pod altogether which is going to contain fresh container with same application.

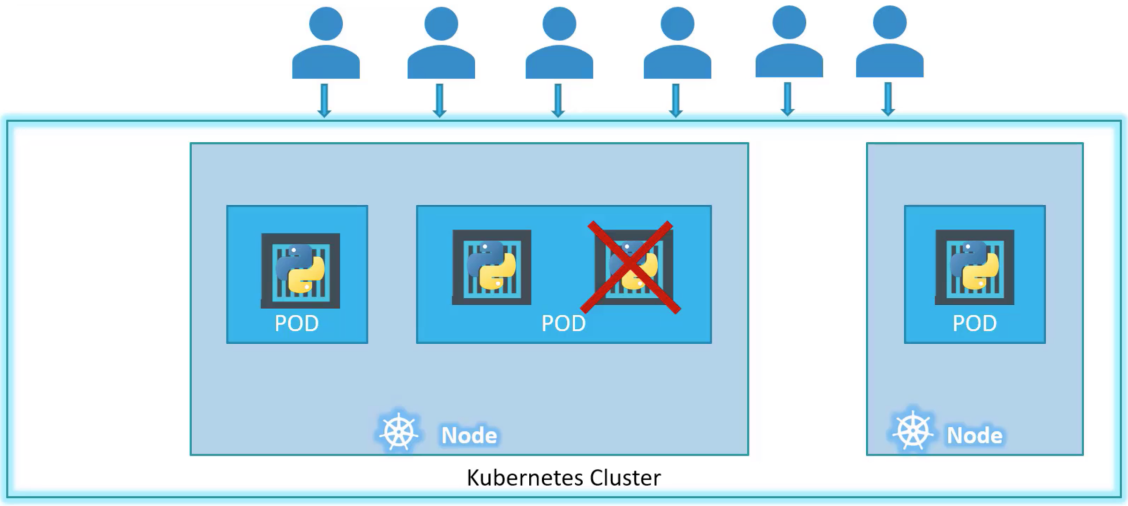


- As you can see, we now have two instances of our web application running on two separate PODs on same Kubernetes cluster.
- But what if the user base further increases and your current node has sufficient capacity ?



- Well in that case you can create new set of nodes and add it to cluster.

- Further PODs will be deployed in on a new node, This will expand the clusters physical capacity.



**To summarise :**

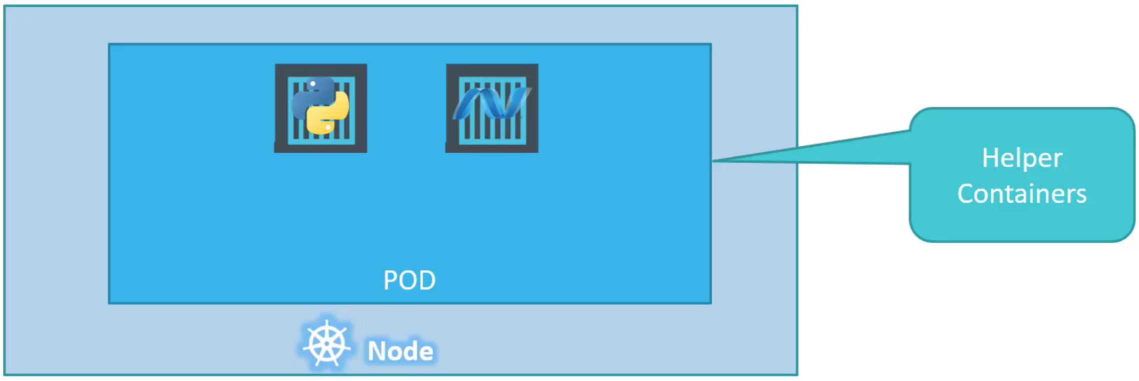
- POD is an Kubernetes object which is going to help you to encapsulate and run the container.
- PODs usually have 1-1 relationship with containers which is running your application.
- To scale up you can create new POD and to scale down you can delete the existing POD.
- Also, you are not going to add any containers into existing POD to scale your application.

## Multi-container PODs

**I just said that PODs usually maintain 1-1 relationship with containers but are we restricted to having single container in an single POD ? and the Answer would be No.**

- A single POD can have multiple containers, but the criteria to have many container in POD is every container should server different purpose.
- As we discussed earlier, if our intention is to scale up the existing container we should always go with new POD.
- Lets assume this scenario, we have an web application and it may require some helper container which is going to serve required data main container.  [Basically supporting machines for main application container]

- **In this case you may need your helper containers to live alongside with your main container.**

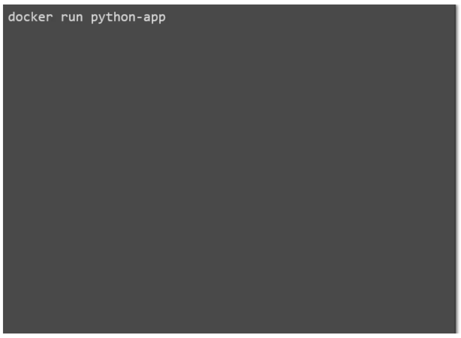


- As an exceptional Kubernetes allows you to put this kind of containers into same POD.
- Now, when an application container is created the helper containers also get created and when it dies helper also dies since they are part of same POD.
- These containers can communicate each other directly by referring to each other as localhost since they share the same network space.
- Also they can easily share the same storage between each other.

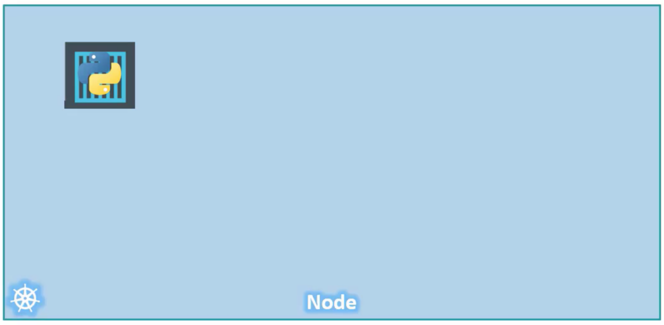
## PODs vs Containers

**To understand better about PODs - Lets try to understand the below scenario.**

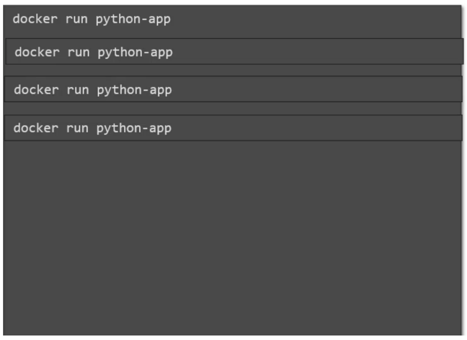
- Let's not think about Kubernetes now.
- Imagine you want to host your application in an container.
- You are going to simpy trigger docker run command and create container which will accessed by your customers.



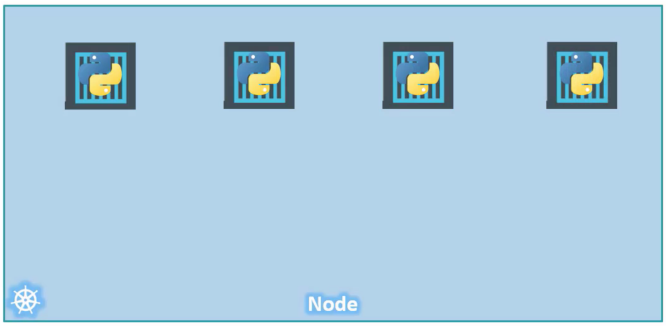
```
docker run python-app
```



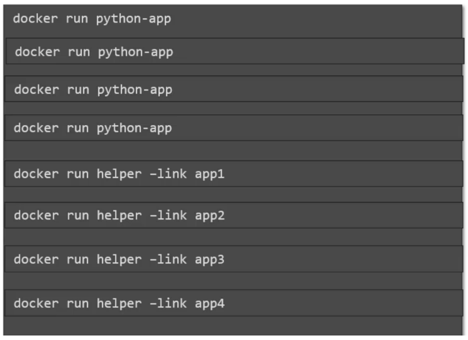
- **Eventually over the period of time when load increases you main create more number of containers more similar to that.**



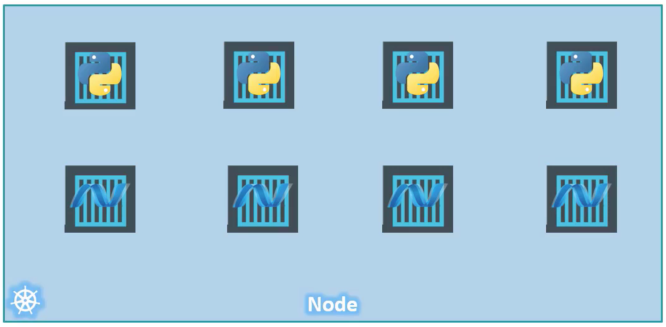
```
docker run python-app
docker run python-app
docker run python-app
docker run python-app
```



- **In future, our application is further developed undergoes architectural changes. In this case number of machine grows and gets complex.**



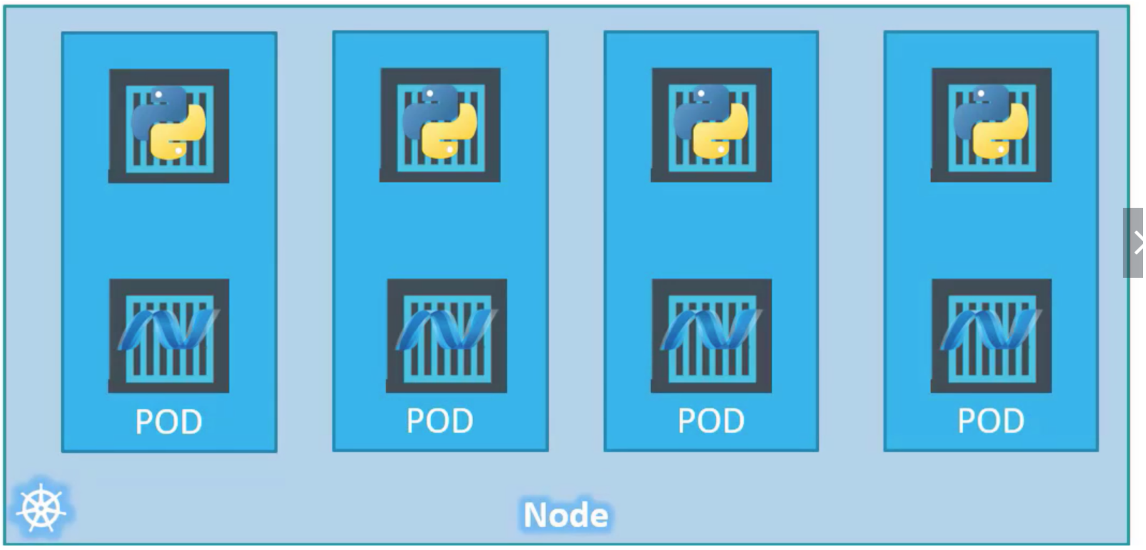
```
docker run python-app
docker run python-app
docker run python-app
docker run python-app
docker run helper -link app1
docker run helper -link app2
docker run helper -link app3
docker run helper -link app4
```



- In this case you may need to care of enabling communication between containers.
- Ensure machines are servers with proper shared storage.
- Also keep a track of which machine is created for which purpose.
- Also need to ensure removing/creating helper machine when relevant master machines is removed.
- Hence while your infrastructure grows it may become very difficult to manage these things.

**Now with help of Kubernetes "PODs" we are going to eradicate all this complications because PODs will take care of**

- Creating & removing containers.
- Maintaining communication between other machines in POD.
- Ensuring Network & Storage is properly shared among machines.



- If you think in longer run this way of implementation is good by foreseeing the upcoming architectural changes and scalability.
- Also note that multi-pod container is a rare use case. Our focus would be more on single container per PODS in upcoming sessions.