**Access pod application outside cluster using hostport:**

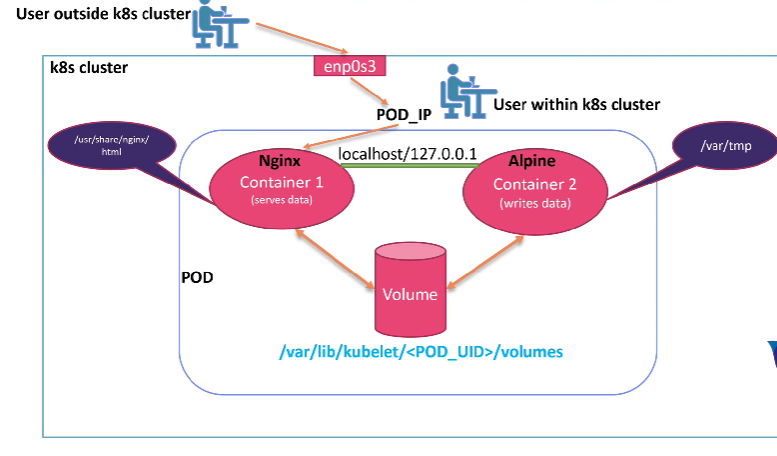
What happens if an application running in a pod needs to be accessed from outside of the kubernetes cluster?

Here comes the concept called HostPort for providing a solution.

Hostport is the phenomenon of binding the host port number to the container port number to access the application running in a pod to the outside of the kubernetes cluster.

For suppose, any user is trying to access nginx application within the kubernetes cluster, it will go to any node then access the application using the pod\_ipaddress along with the port number on which that application is running(by default nginx runs on 80 port).

But the concern is when we need to access the same application from outside of the cluster, we use hostport. Host port means any port number (in this case 8090 is taken) is binded to the container port 80, then we can expose our applications to outside of the kubernetes cluster by the ip address of the machine where kubernetes is installed along with the hostport-8090.



These are some of the fields with the description we can implement in the container ports section.

| **Field** | **Description** |
| --- | --- |
| Container Port (integer) | Port number to expose on the pods ip address  This must be a valid port number, 0 < x< 65536 |
| hostIP (string) | What host IP to bind the external port to. |
| hostPort (integer) | Port number to expose on the host.  This must be a valid port number, 0 < x< 65536 |
| Name (string) | This must be an IANA\_SVC\_NAME and unique within the pod. Name of the port can be referred to by the services. |
| Protocol (string) | Protocols for the port must be UDP, TCP or SCTP. The Default is TCP. |

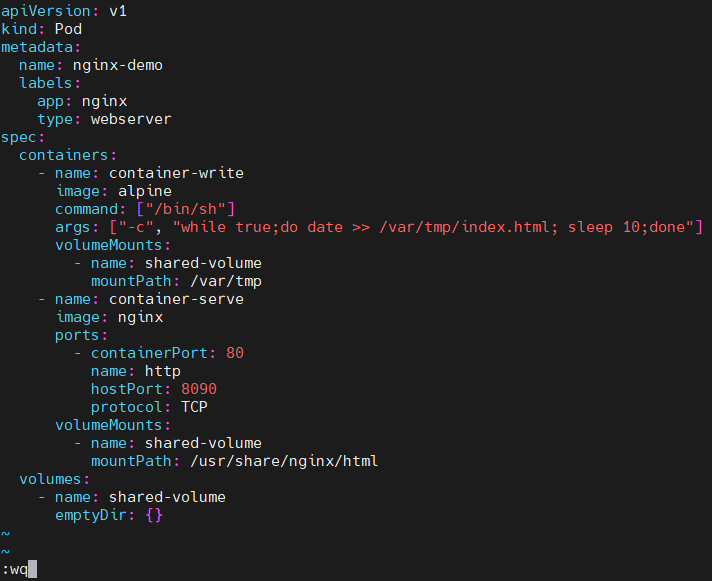
Create a yaml file using vim file editor to create multiple containers inside a pod which will have different purpose.



In this file, apiversion for pod is v1, kind is pod as we are trying to create a pod object and then name the pod under metadata, also label it(optional) with app and type.

Under specifications create the container by providing the name(container-write) using alpine image to write the data and run the bash shell. Now under args,we run a while loop to execute the date command for every 10 seconds, since date command output will be stored in /var/tmp/index.html. Volume mounts section is to mount the shared volume to the above container at the mount path /var/tmp.

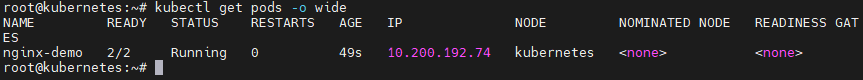
Let us now create one more container with the name as container-serve which is created with the nginx image meant to serve the data. Now in the ports section container port is 80, the name of the port is http binded to the hostport 8090 with the TCP protocol. While mounting the path /usr/share/nginx/html directory to the shared volumes which is defined under volume section along with emptyDir. Finally save the file.



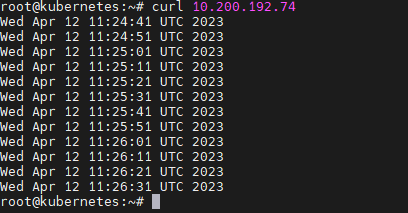
Kubectl create -f file\_name command will create the pod according to the specified file. Hence the pod is created.



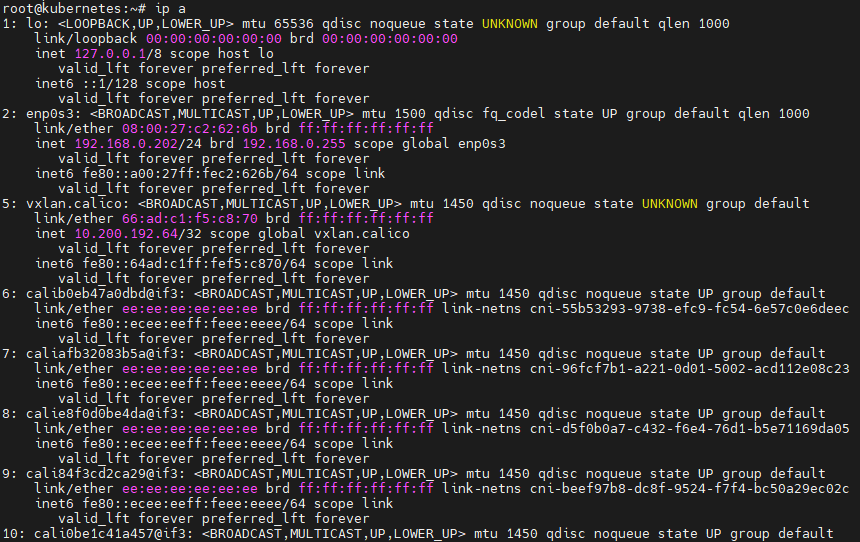
Kubectl get pods -o wide command will give the pod details like status, age, pod ip address and on which node that particular pod is created.



Curl the pod ip address to check whether the while loop passed in arguments of the pod file(i.e date command output) is executed for every 10 seconds.



Ip a command will show the ip address of the host machine.



On any browser, paste the ip address of the machine where kubernetes is installed along with the hostport-8090(<hostmachine\_ip:hostPort>). With this we access the date command output generated for every 10 seconds and get stored in the index.html file under volume.

