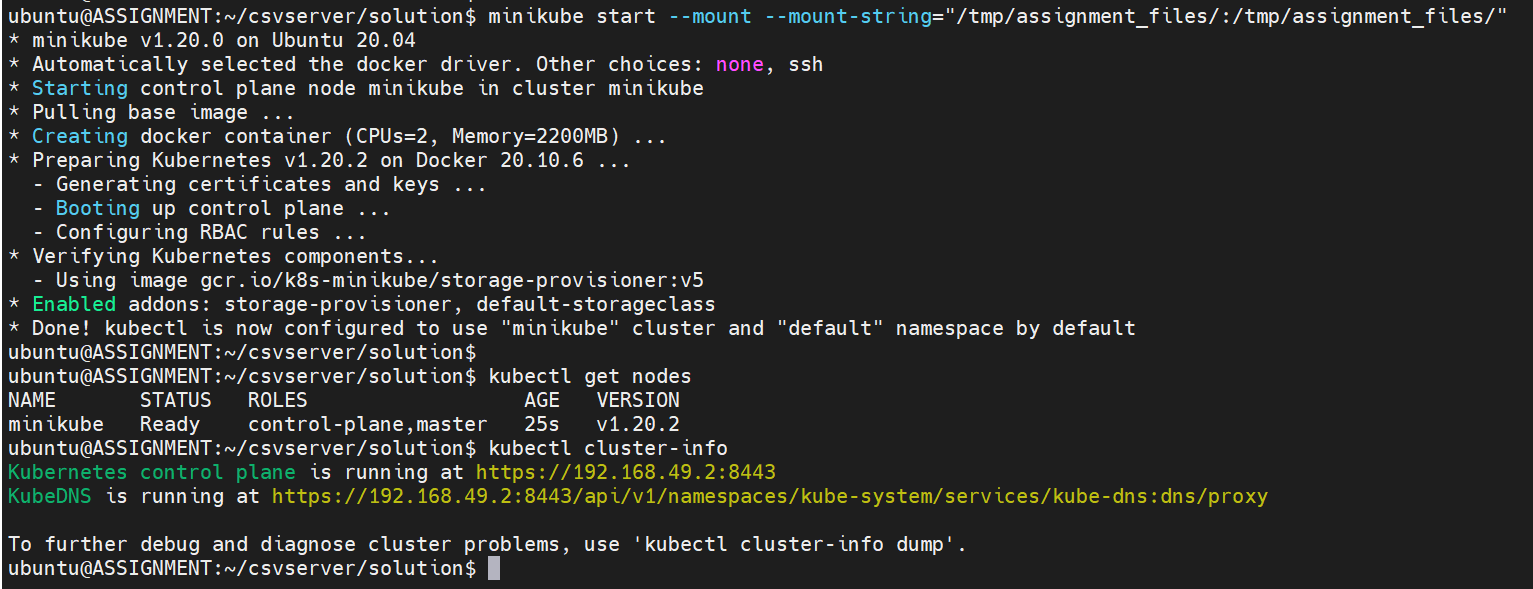
**Assignment in a Kubernetes Cluster**

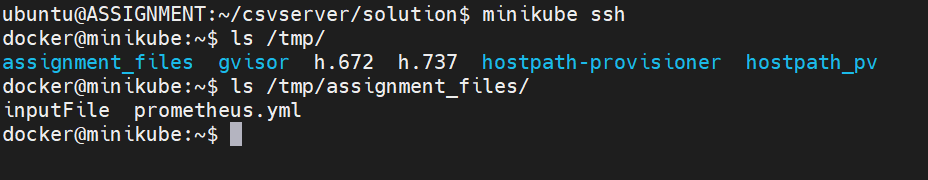
**NOTE**

* All the tasks of the assignment (Part-I, Part-II & Part-III) are performed as one single activity. Ex: In K8s environment, as the container is already getting created in YAML, there is no need to use *docker-compose*.
* This way of solving a problem is suitable only for development and testing purposes as the entire activity happens on on a single node.
* If these types of issues are observed in staging/production, then the followed approach must be modified, and relevant steps should be taken as per the situations like –
  + Multi node architecture (1 or 1+ Master & 2-3 or more Worker Nodes)
  + Using Kubernetes in Public Cloud – GKE in GCP, EKS in AWS, AKS in Azure, etc.
  + If there is a need for storage devices – storage on commodity hardware through CEPH/GLUSTER or from vendors (NetApp, EMC, etc.)

**Pre-requisites**

1. Setup the K8s environment with minikube. View the status of the cluster and ensure that the needed files are available for creating the containers.

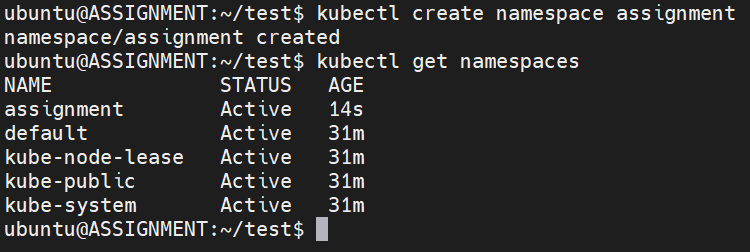




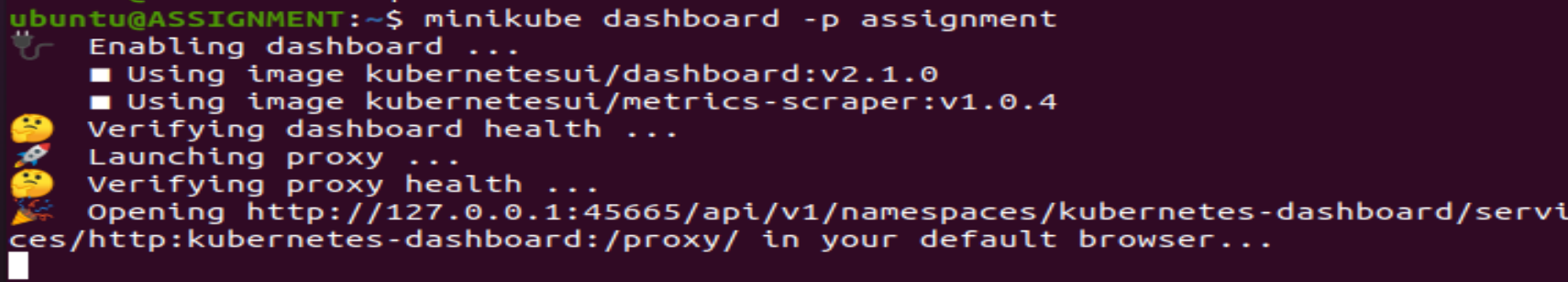
Cluster Information –

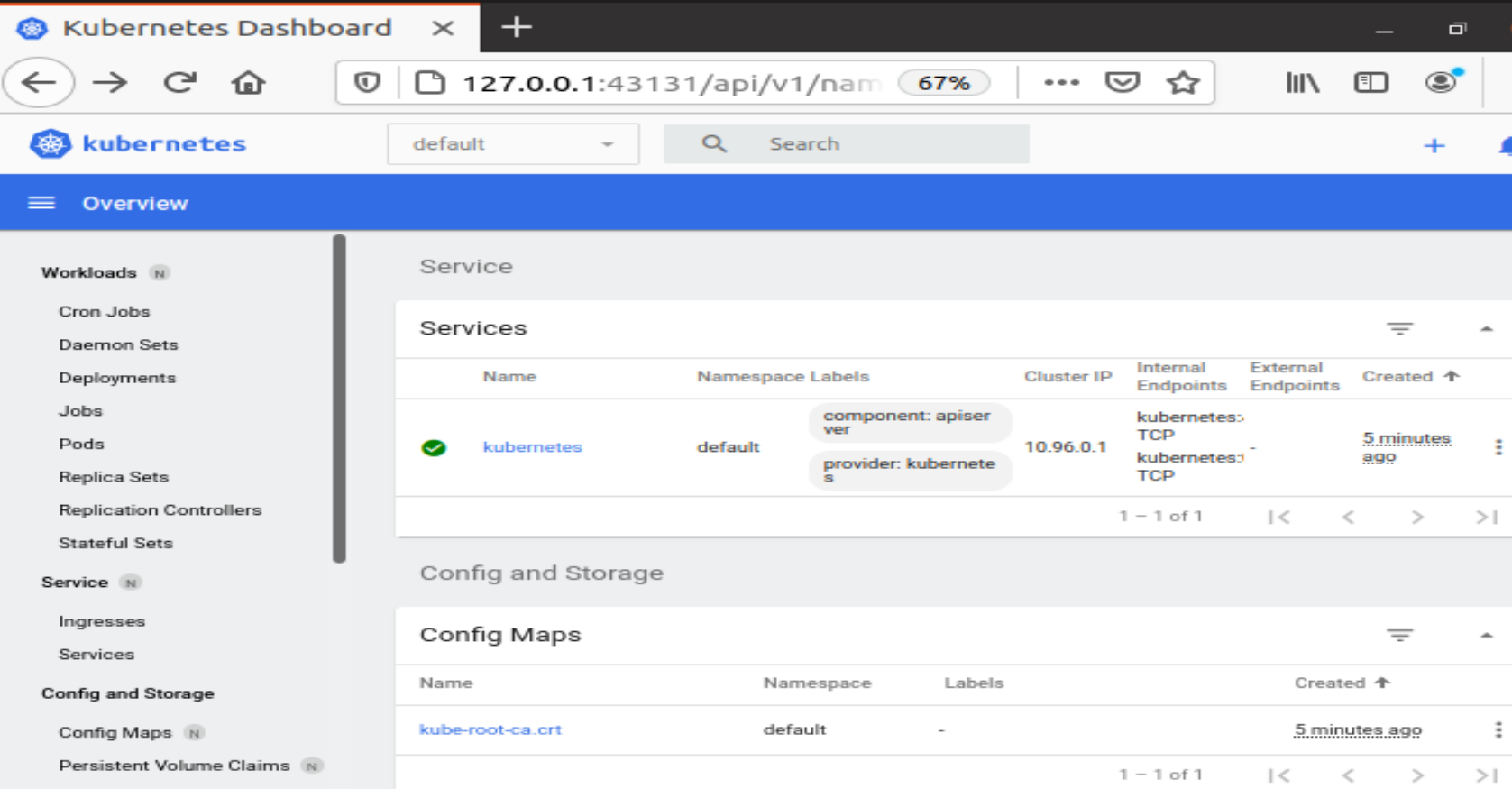


1. Create a namespace “*assignment*” in which all these tasks will be performed



Open the Kubernetes Dashboard of the created namespace – *assignment*



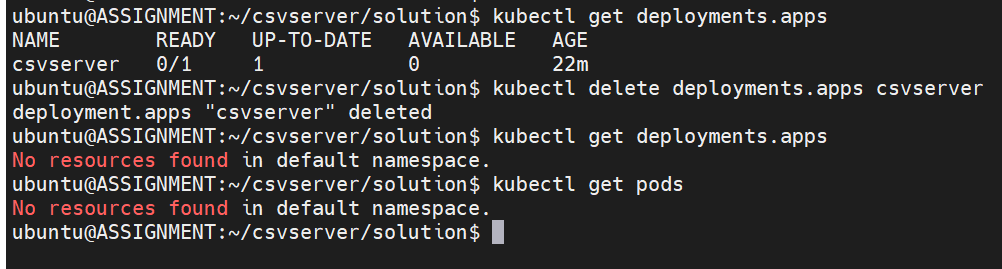


**ASSIGNMENT**

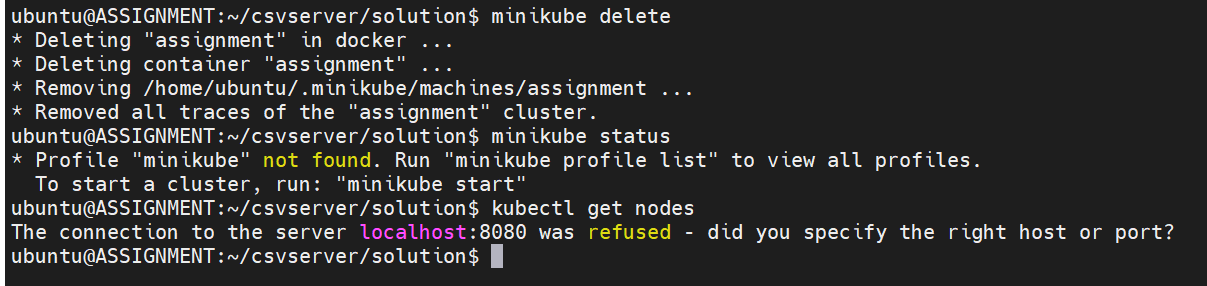
1. Run the container image *infracloudio/csvserver:latest* in background and check if it's running.`
2. If it's failing then try to find the reason,



Delete the csvserver deployment and the pod also gets deleted.



Also delete the Kubernetes setup.



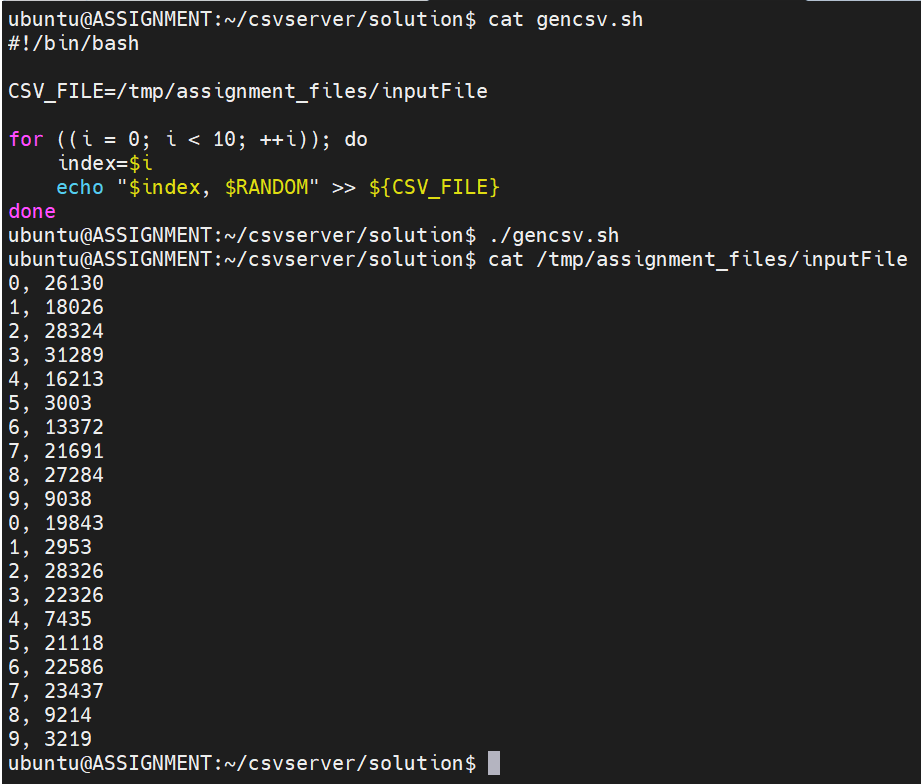
The K8s cluster created with *minikube* is for development and testing purpose only. So, it doesn’t have any impact in deleting the Kubernetes cluster as the followed method is different when deploying a K8s cluster in staging or production.

1. Write a bash script gencsv.sh to generate a file named inputFile whose content looks like

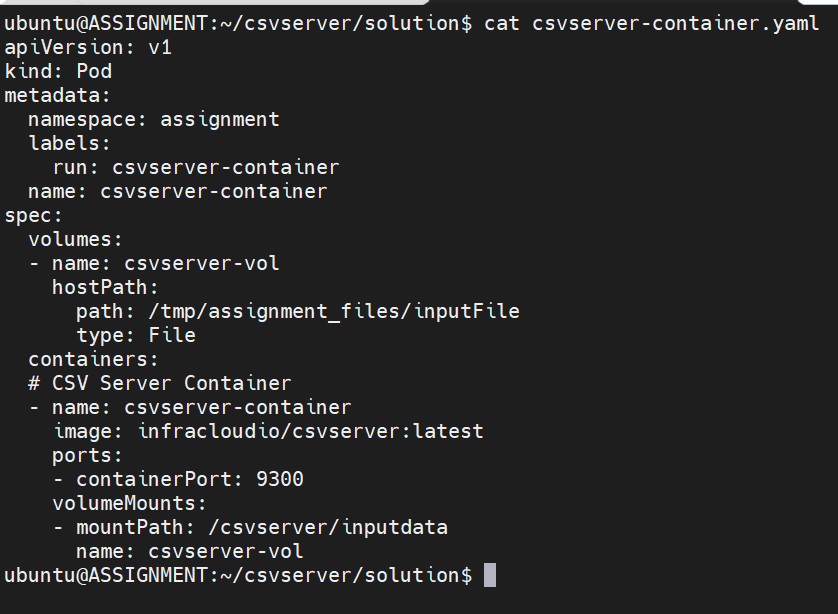
0, 234

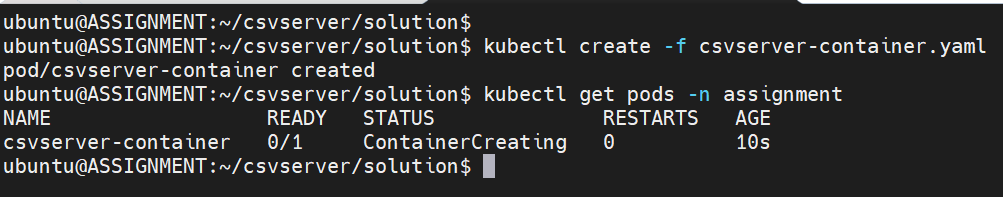
1, 98

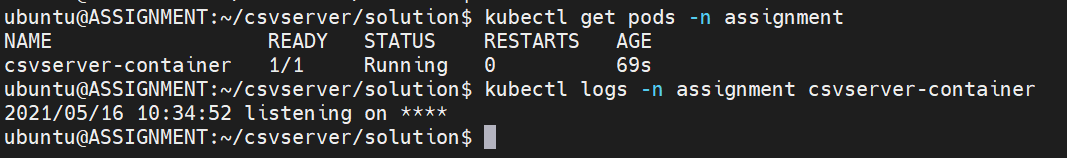
2, 34

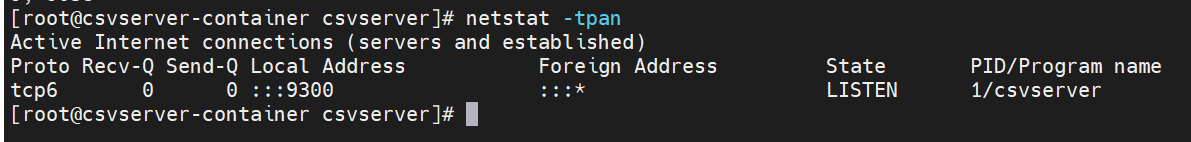
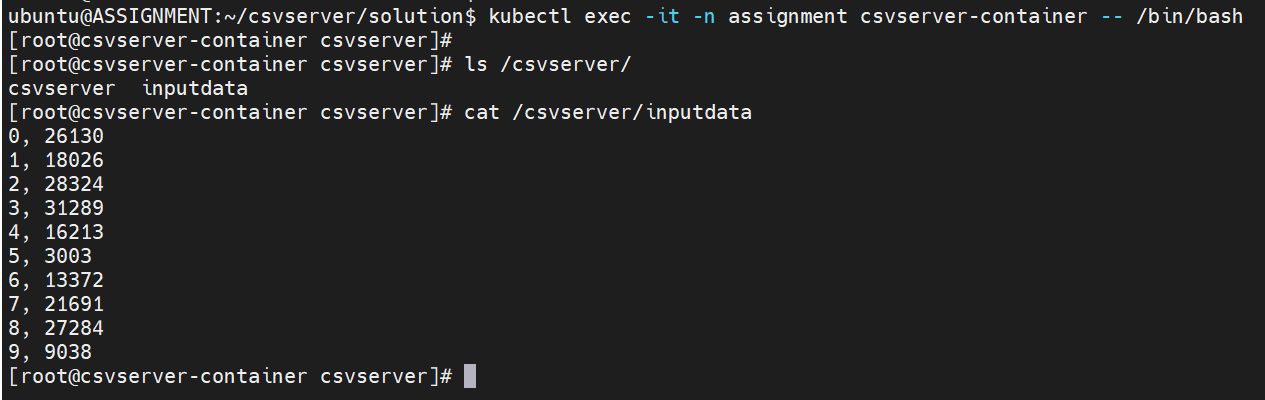


1. Run the container again in the background with file generated in (3) available inside the container (remember the reason you found in (2)).
   1. Create a YAML file for the container of csvserver.

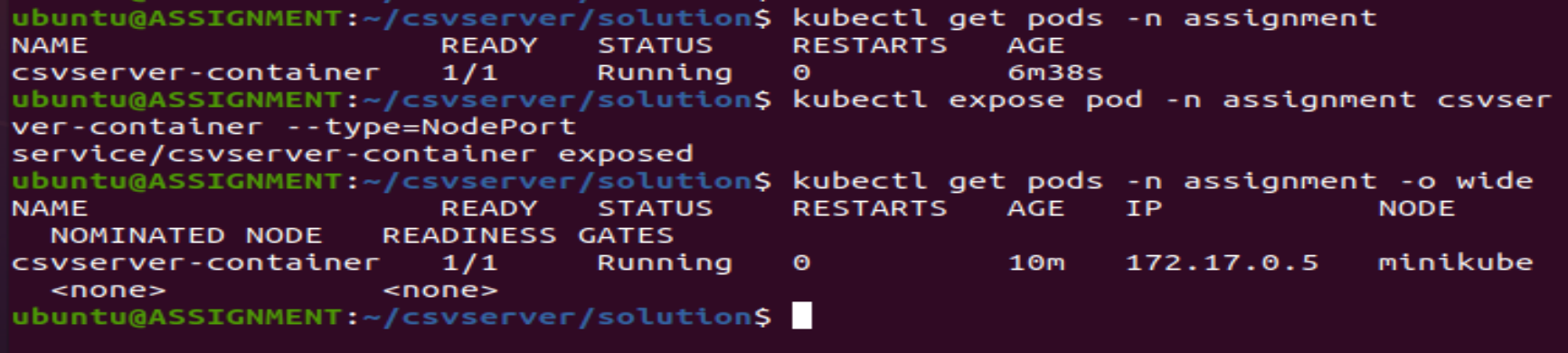


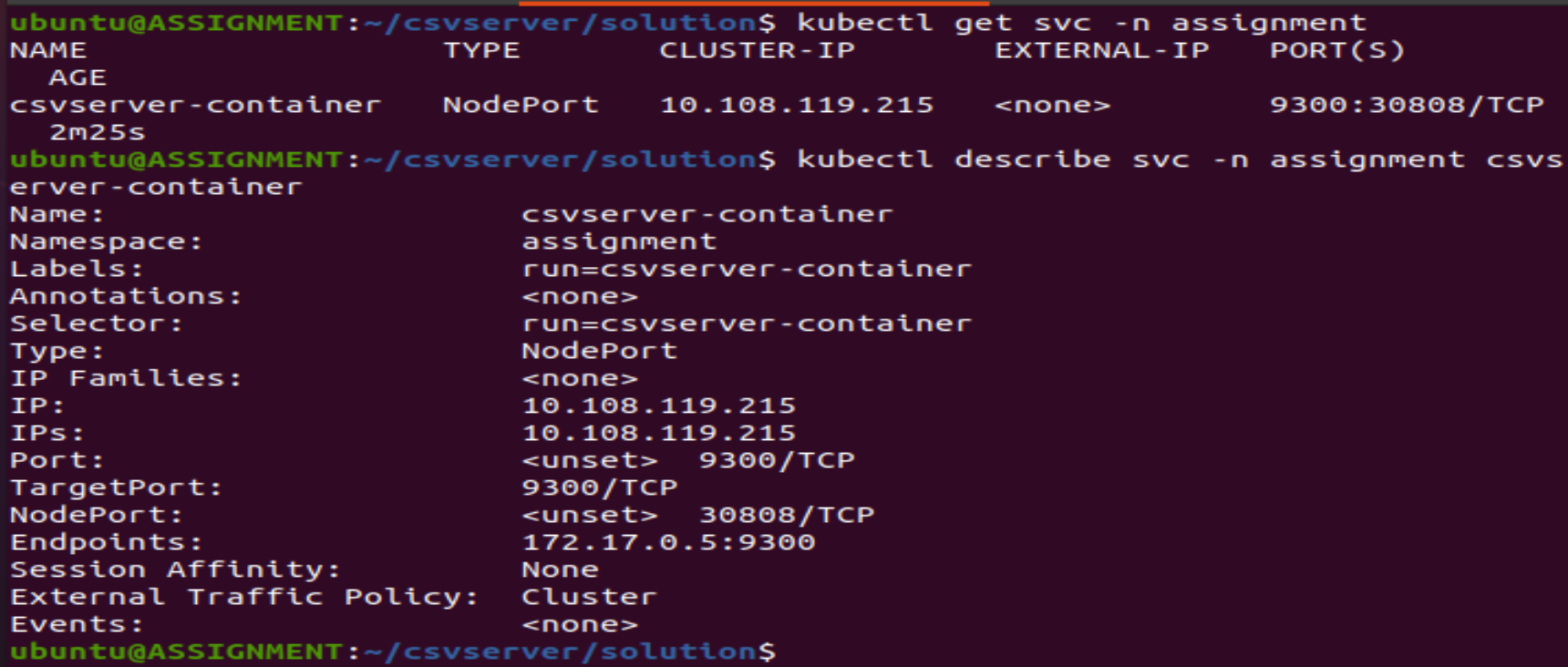
1. Now start the container. 

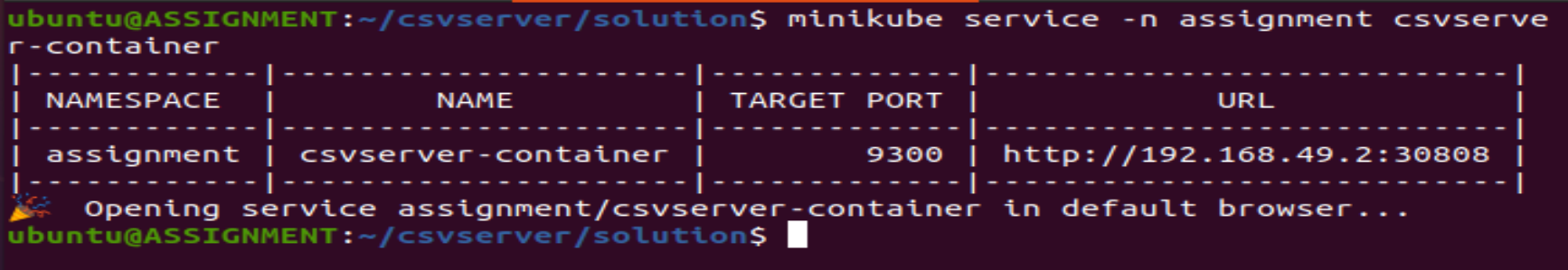


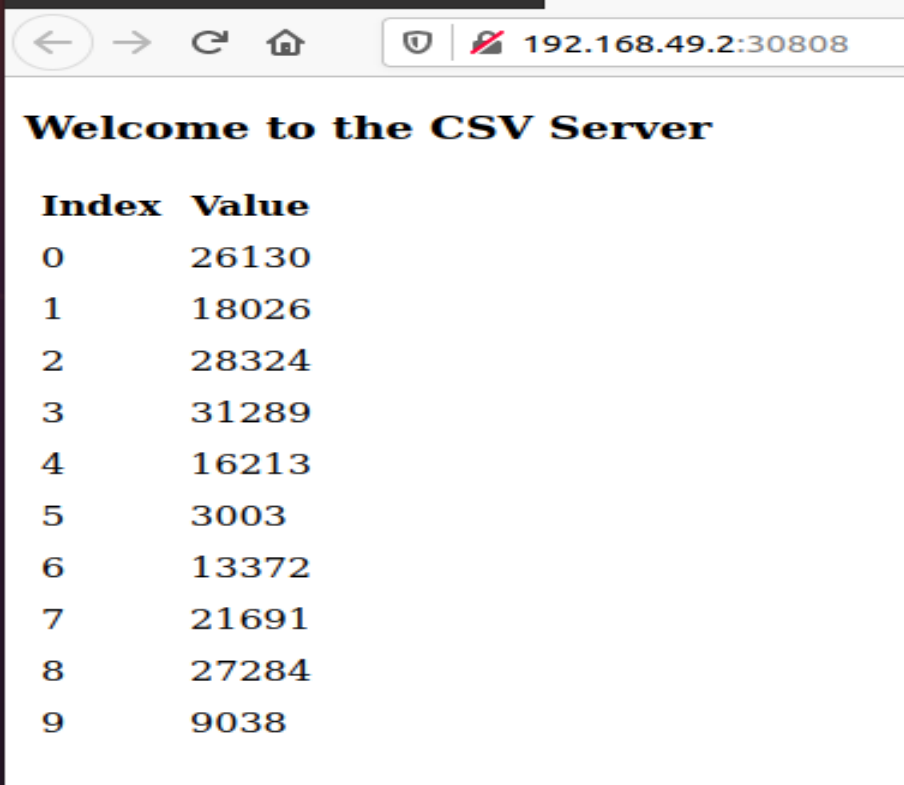
1. Get shell access to the container and find the port on which the application is listening.
   1. Access the container and check the existence of *inputdata* (generated CSV file) and the port on which the application is running
2. Run the container and make sure, the application is accessible on the host at <http://localhost:9393>

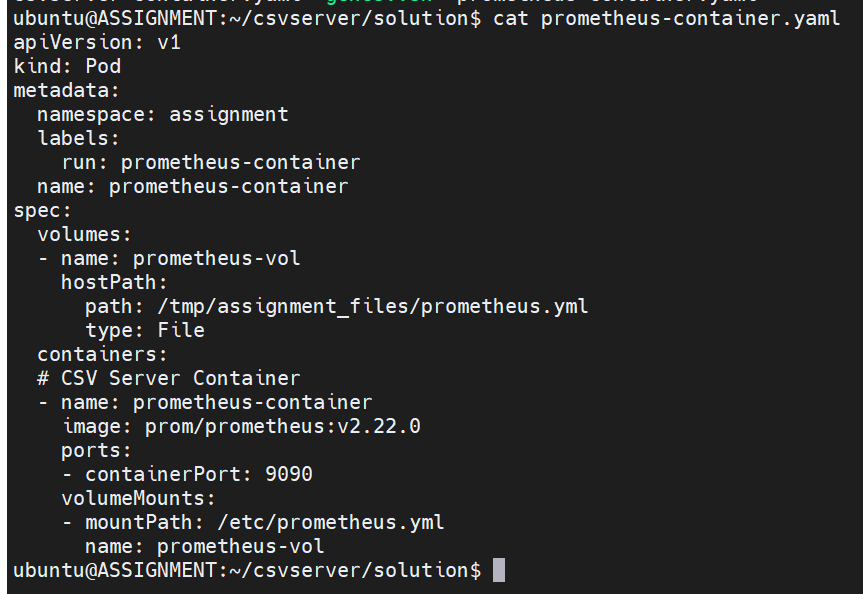
**NOTE**: The current Kubernetes environment has been setup with minikube – exposure of applications outside the cluster is possible only through **NodePort & LoadBalancer**. When using LoadBalancer, the ***minikube tunnel*** command must be running in a separate terminal to keep the LoadBalancer running.

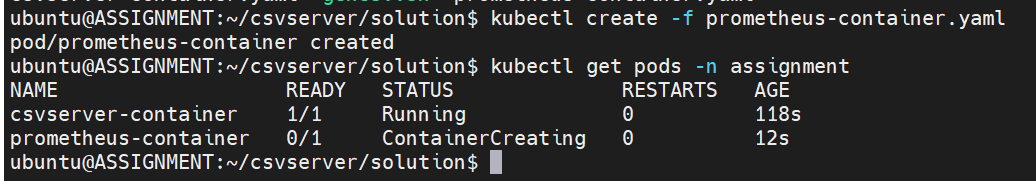
So, to avoid this NodePort has been used. Default range of the NodePort is 30000 – 32767. 

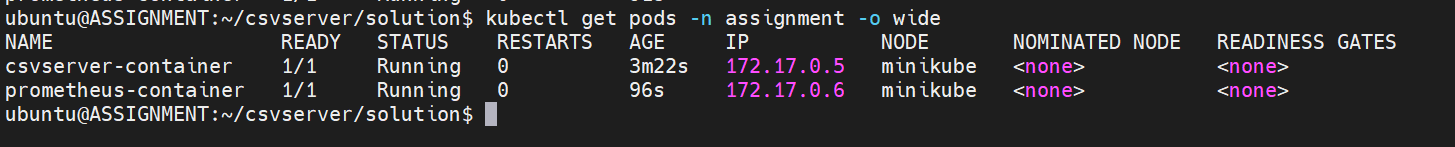


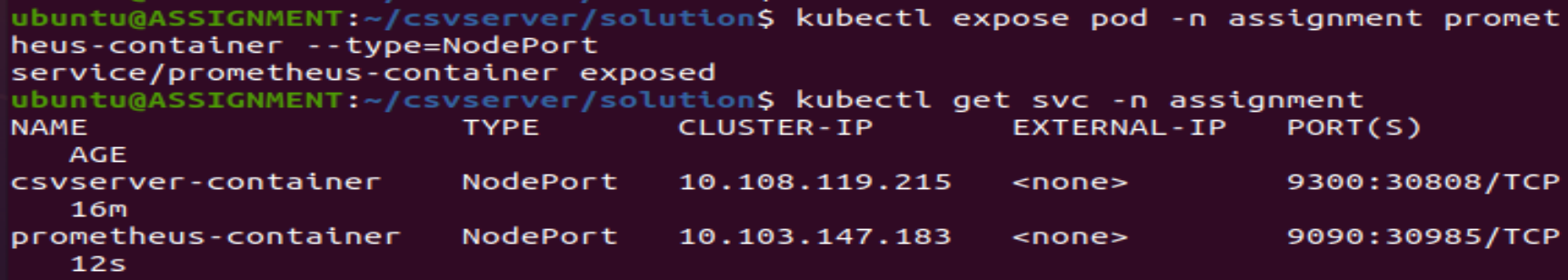
The 9300 port on which the csvserver-container is running gets exposed through NodePort and can be accessible after executing the following command – 

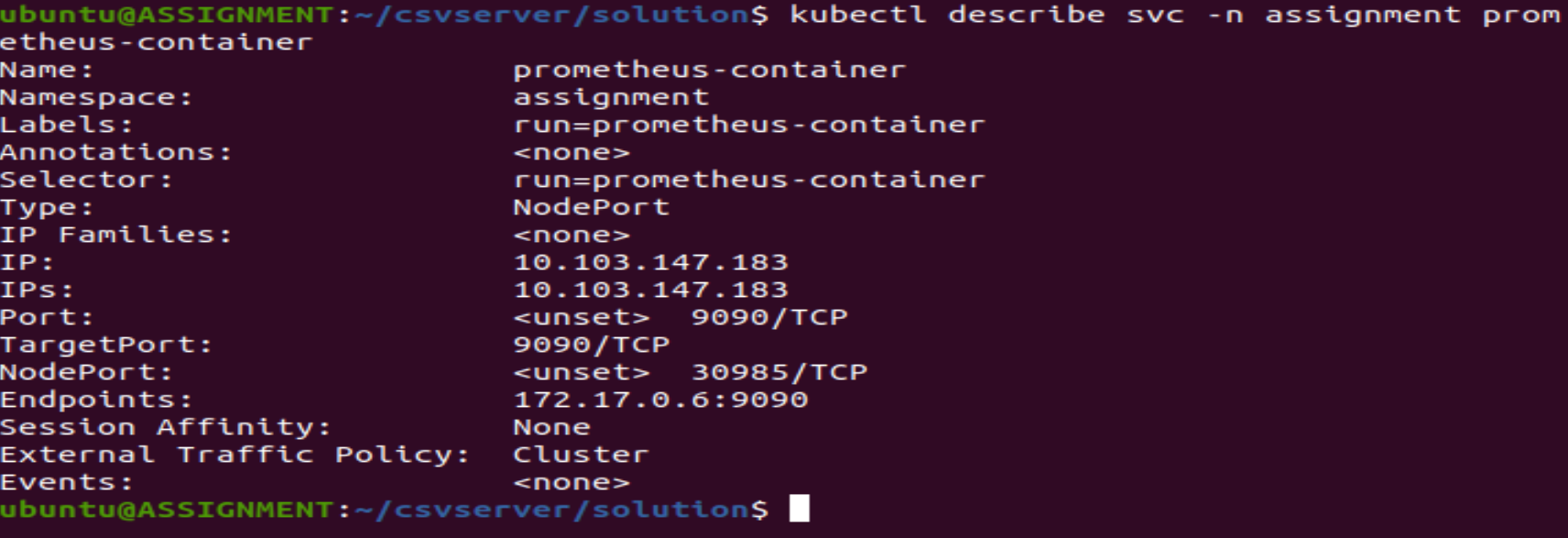


1. Add Prometheus container (prom/prometheus:v2.22.0) 

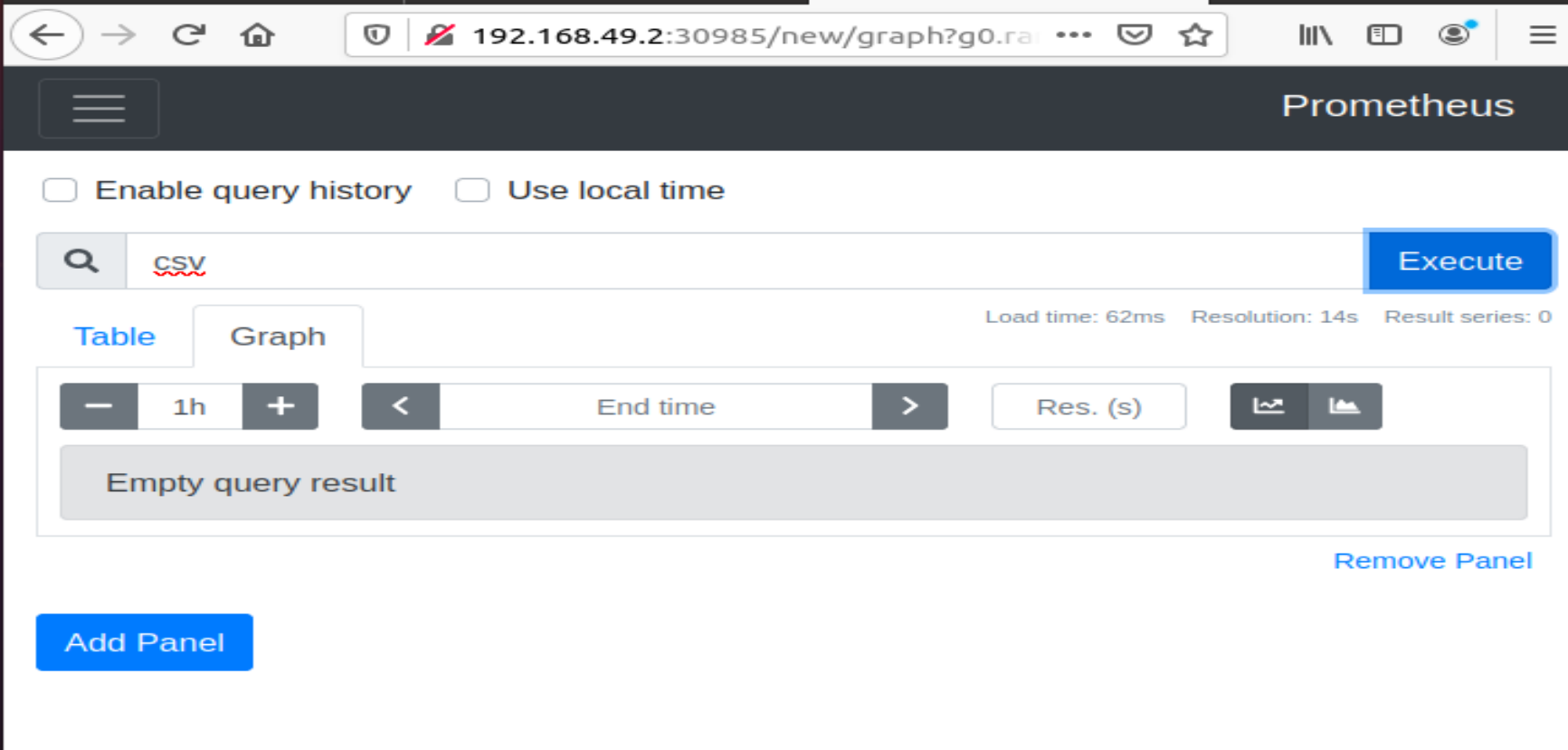








1. Now start the service for viewing the Prometheus UI



1. The Prometheus UI is accessible but the entry of csv\_records and the graph that has to appear in a straight line are not present.

**END RESULT**

