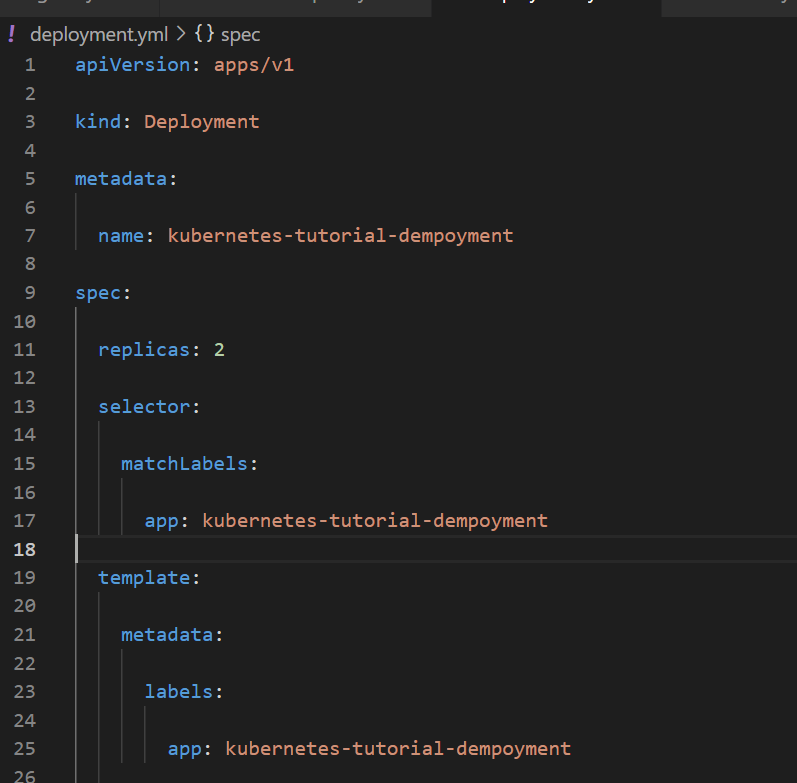
**Kubernetes:**

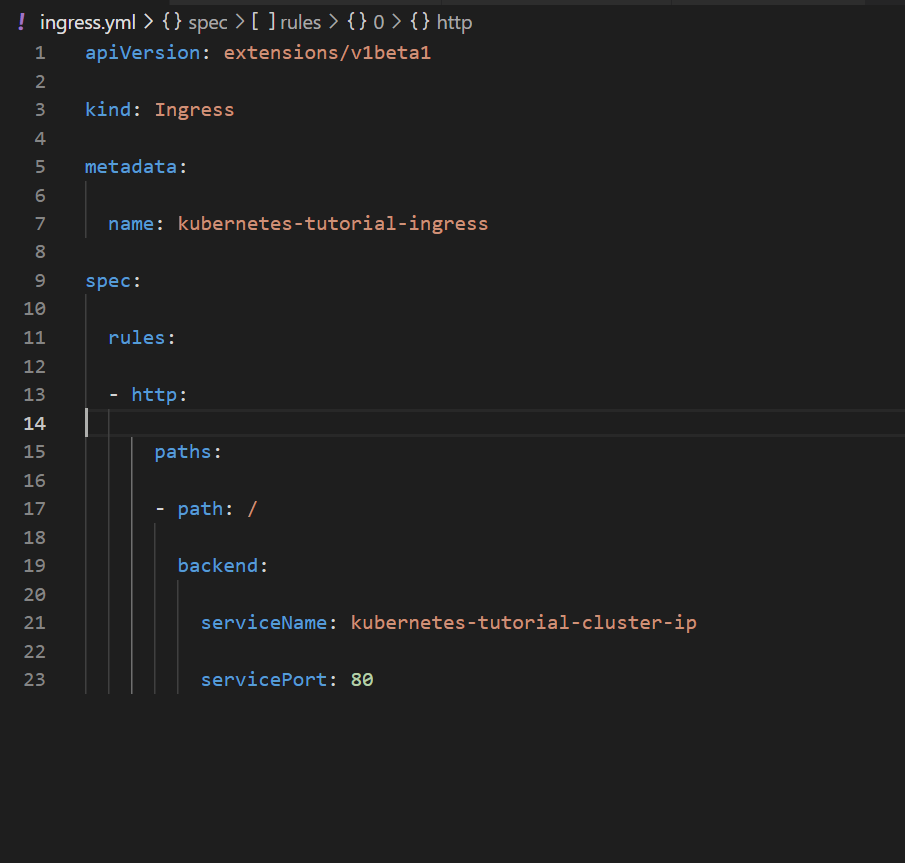
is an [open-source](https://en.wikipedia.org/wiki/Open-source_software) [container](https://en.wikipedia.org/wiki/Operating-system-level_virtualization)-[orchestration](https://en.wikipedia.org/wiki/Orchestration_(computing)) system for automating computer [application](https://en.wikipedia.org/wiki/Application_software) deployment, scaling, and management.[[5]](https://en.wikipedia.org/wiki/Kubernetes#cite_note-5) It was originally designed by [Google](https://en.wikipedia.org/wiki/Google) and is now maintained by the [Cloud Native Computing Foundation](https://en.wikipedia.org/wiki/Cloud_Native_Computing_Foundation). It aims to provide a "platform for automating deployment, scaling, and operations of container workloads".[[6]](https://en.wikipedia.org/wiki/Kubernetes#cite_note-:2-6) It works with a range of container tools and runs containers in a cluster, often with images built using [Docker](https://en.wikipedia.org/wiki/Docker_(software)). Kubernetes originally interfaced with the Docker runtime[[7]](https://en.wikipedia.org/wiki/Kubernetes#cite_note-7) through a "Dockershim"; however, the [shim](https://en.wikipedia.org/wiki/Shim_(computing)) has since been deprecated in favor of directly interfacing with the container through [containerd](https://en.wikipedia.org/wiki/Containerd" \o "Containerd), or replacing Docker with a runtime that is compliant with the Container Runtime Interface (CRI) introduced by Kubernetes in 2016.

1)**write configuration file for deployment**

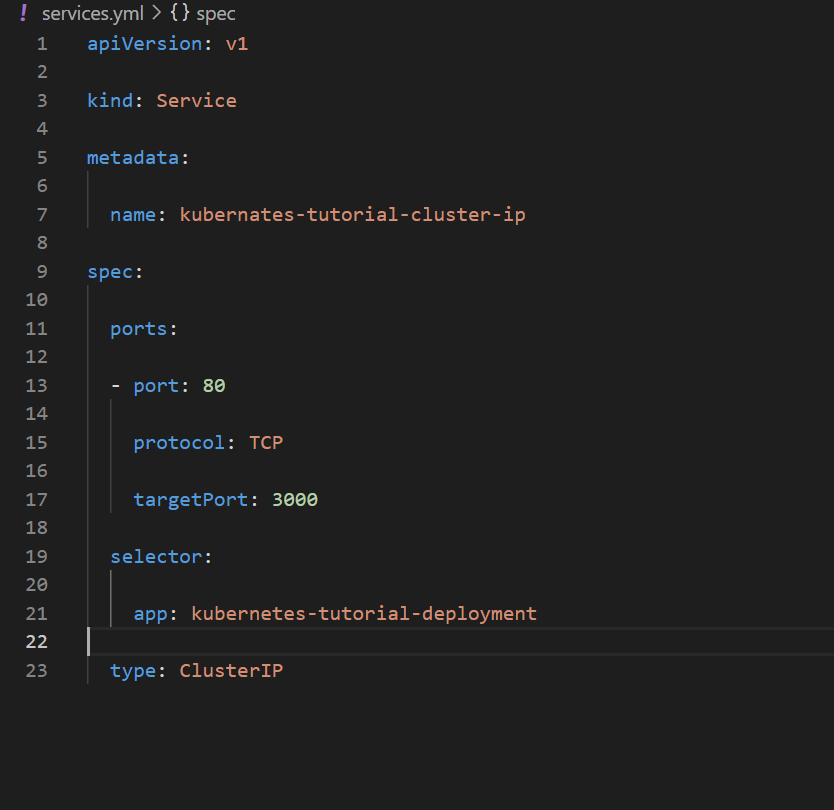




**2)then write manifest for ingress**



**3)write configuration file(manifest) for services**



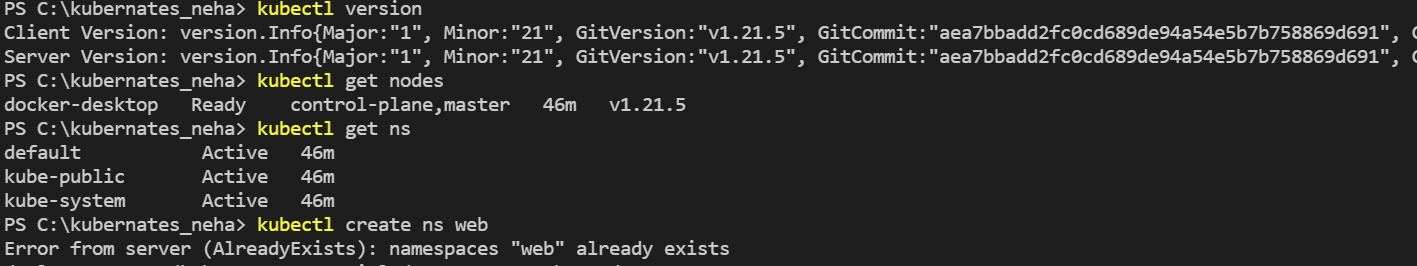
**4)then save above files and run following commands:**

**1.kubectl version: for checking version for kubernetes**

**2.kubectl get nodes: we get all nodes**

**3.kubectl get ns: we will get all namespaces available**

**4. kubectl create ns name : this command used to create namespaces of our own**



**5. kubectl apply -f .\deployment.yml :** You can create a Deployment using the kubectl apply , or kubectl create commands. Once created,

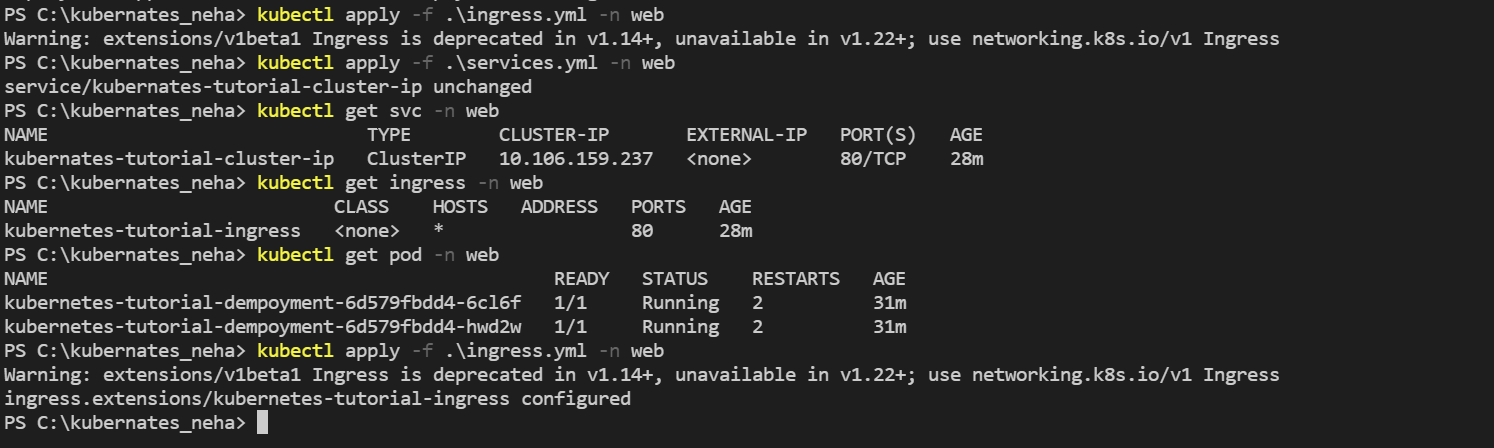
the Deployment ensures that the desired number of Pods are running and available at all times. The Deployment automatically replaces Pods that fail or are evicted from their nodes.

**6. kubectl apply -f .\ingress.yml :** You can create a ingress using the kubectl apply , or kubectl create commands.

**7. kubectl apply -f .\services.yml :** You can create a services using the kubectl apply , or kubectl create commands.

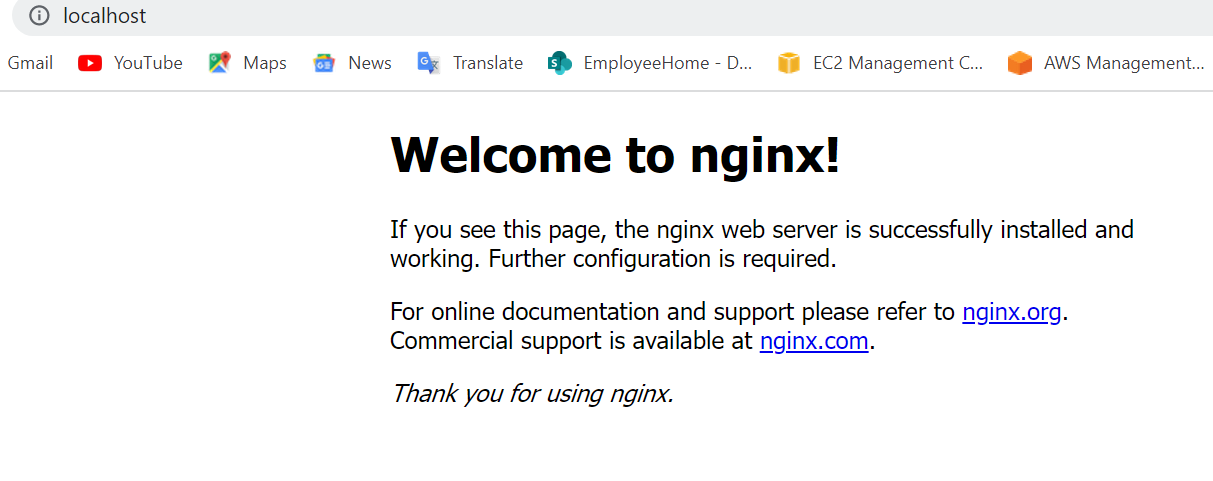
**8.kubectl get svc :** Issue the following command to get the external IP address

**9.** **kubectl get pods :**Issue the following command to get the pod name

****

**docker run -d -p 80:80 nginx:** : To run or manage Docker containers, open a PowerShell console. Start a Nginx web server container in the background (--detach) on port 80 using the docker container run parameter: docker container **run** --detach -port 80:80 nginx. It will return a container ID that uniquely identifies the container.

Output:

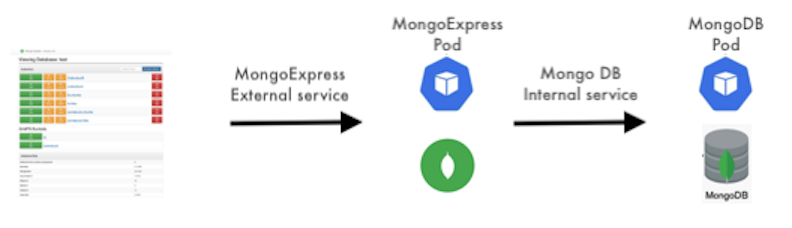


\*\*\*\*\*\*\*\*\* **To run mongodb deployment and mongodb express \*\*\*\*\*\*\*\*\*:**

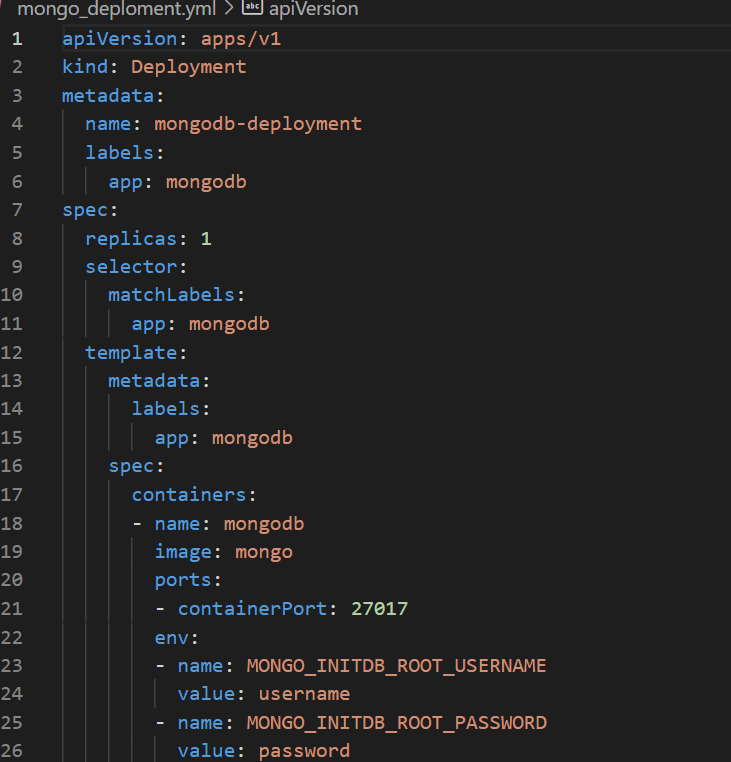
**MongoDB:**

**mongo-express** is a web-based MongoDB admin interface written with Node.js and Express.

Here is a digram that we want to implement with Kubernetes:

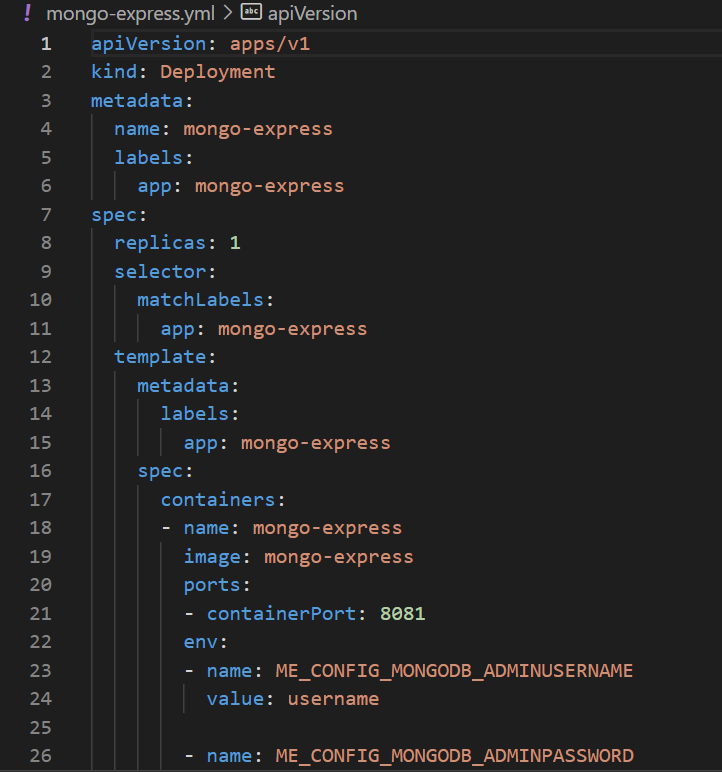


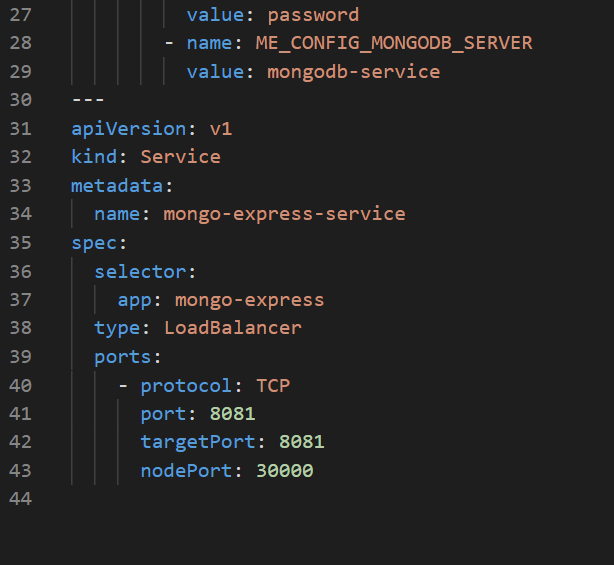
**1)write configuration file for mongodb deployment**



****

**2) write manifest for mongodb -express**





**3)open powershell and run following commands to download minikube for above Kubernetes file:**

**1.** **To install the latest minikube stable release on x86-64 Windows using .exe download:**

New-Item -Path 'c:\' -Name 'minikube' -ItemType Directory -Force

Invoke-WebRequest -OutFile 'c:\minikube\minikube.exe' -Uri 'https://github.com/kubernetes/minikube/releases/latest/download/minikube-windows-amd64.exe' -UseBasicParsing

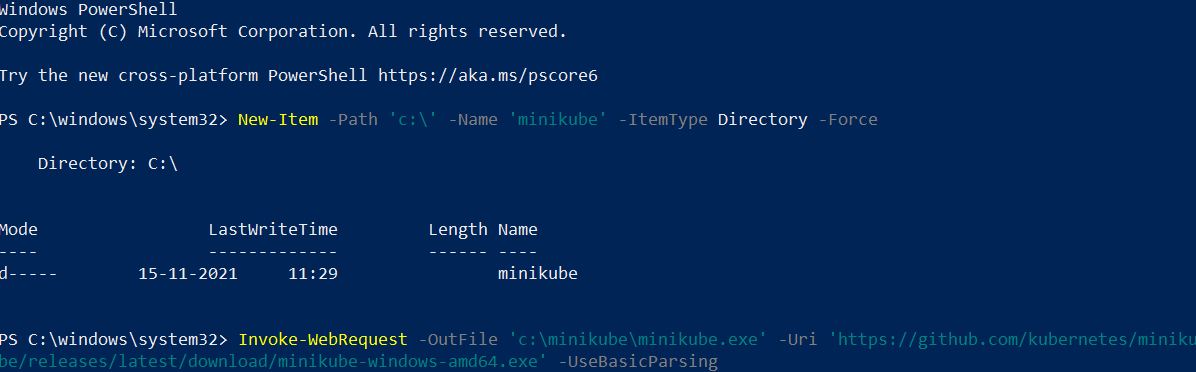
**2.** **Add the binary in to your PATH.  
  
*Make sure to run PowerShell as Administrator.***

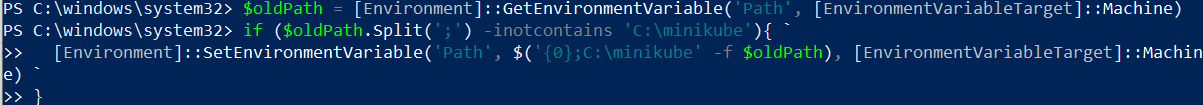
$oldPath = [Environment]::GetEnvironmentVariable('Path', [EnvironmentVariableTarget]::Machine)

if ($oldPath.Split(';') -inotcontains 'C:\minikube'){ `

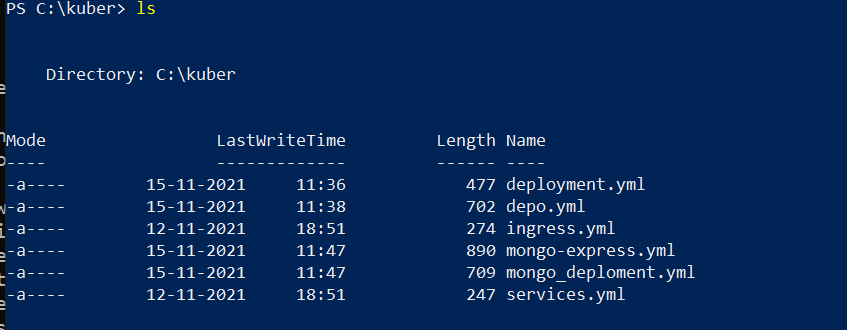
[Environment]::SetEnvironmentVariable('Path', $('{0};C:\minikube' -f $oldPath), [EnvironmentVariableTarget]::Machine) `

}

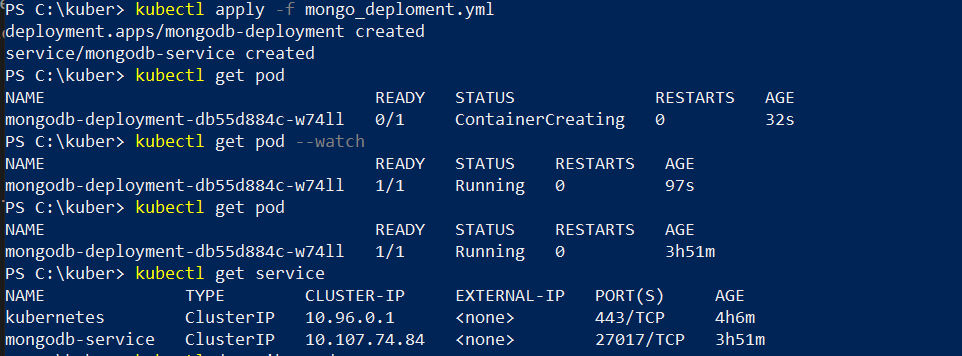


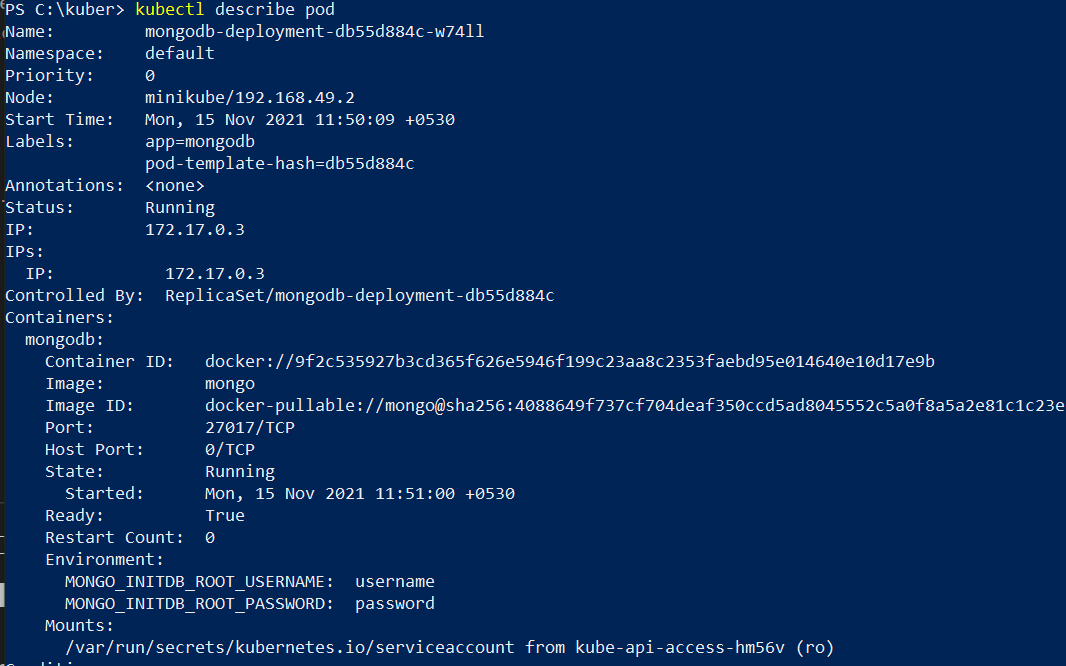


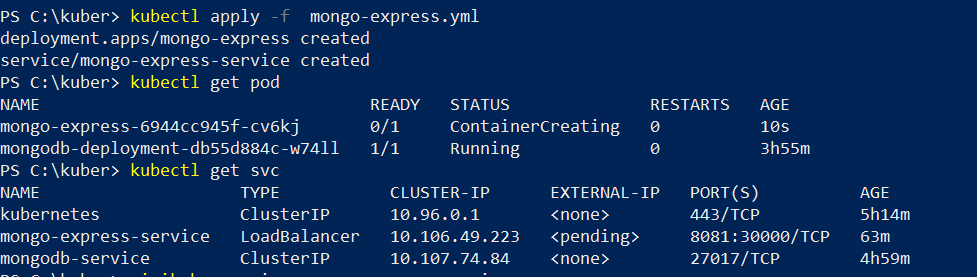
**4)check files using ls command**



**5)then run following commands to run manifest**

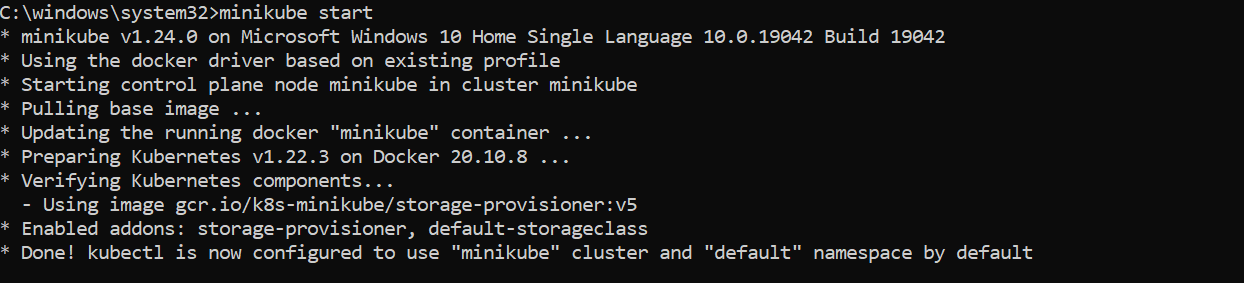




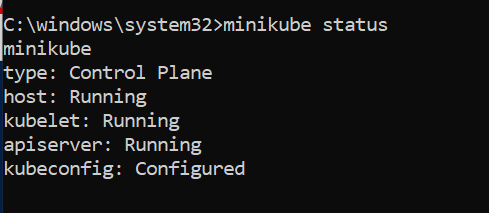


**6)then in cmd run**

**1.minikube start :to start the downloaded minikube**

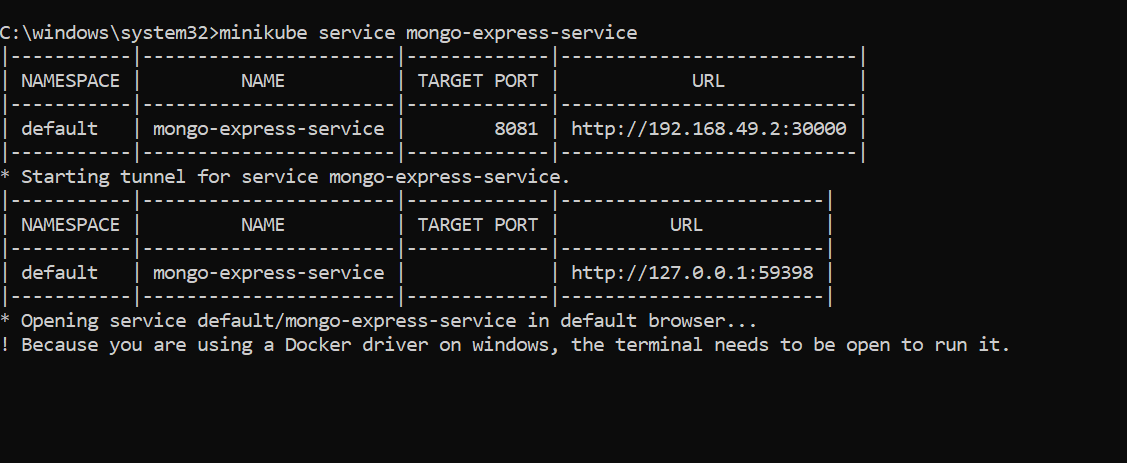


**7)check status using minikube status**



**8)run minikube service mongo-express-service:** Unlike the internal service which has a Cluster-IP (internal) by default, the the **LoadBalancer** service type has additional IP (external). As show in the output, it's currently in **<pending>**, meaning not assigned yet. This is specific to Minikube, otherwise, it should have one. For example, in environments such as AWS or GCP etc.

So, in Minikube, we can assign external service an IP using minikube service command. With this command, we can access a Service exposed via a node port:



**9)** **The run opens up the following:**

