**Prerequisite**

1. Windows 8 or 10 OS
2. Min 8 GB RAM
3. Docker Desktop

M**inikube setup steps in windows OS :**

Minikube runs a single-node Kubernetes cluster on your machine so that you can try out Kubernetes for your daily development work.

Step 1 :

install kubectl: <https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/>

kubectl is a command line tool, using kubectl we can connect to k8s cluster from our computer .

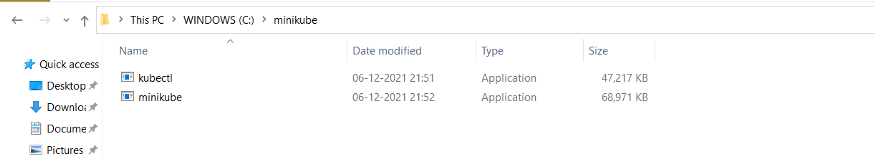
Step 2 :

install minikube : <https://v1-18.docs.kubernetes.io/docs/tasks/tools/install-minikube/>

after download minikube-windows-amd64 exe file rename it to minikube.exe

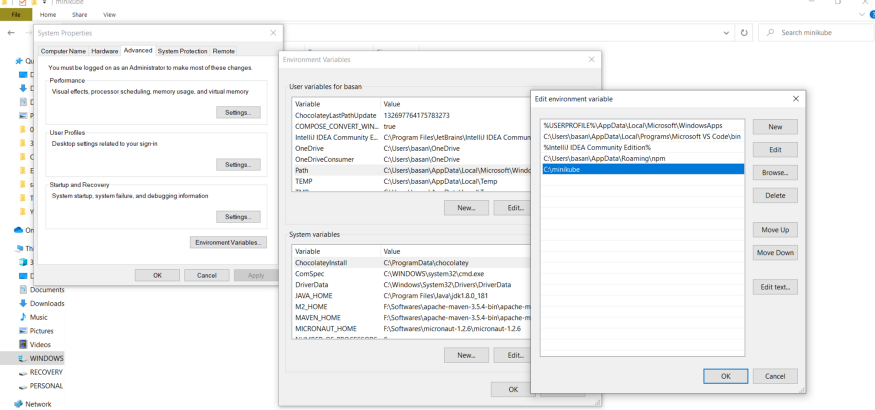
Step 3:

Once you download both exe file , just move these two file to separate directory/folder like below



Step 4:

Next set this folder path as environment variable in your windows



Step 5 :

After set environment variable , open command prompt and run command

minikube version

it should display current installed minikube version

We are good with minikube setup in windows . now you can play with k8s

Step 6 :

To create a Kubernetes cluster first we need start minikube server in our system . There is multiple driver using any of them you can start your minikube

minikube start --driver=<driver name>

1. Hyper-v

**Note**: Hyper-V can run on three versions of Windows 10: Windows 10 Enterprise, Windows 10 Professional, and Windows 10 Education.

2. VirtualBox

If you have enough memory in your system like 16 GB RAM and good processor then VirtualBox is right choice for you .

Download VirtualBox : <https://www.virtualbox.org/wiki/Downloads>

3. Docker

In this example i will use Docker as already it installed in my machine .

run below command to start minikube with docker driver

minikube start --driver=docker

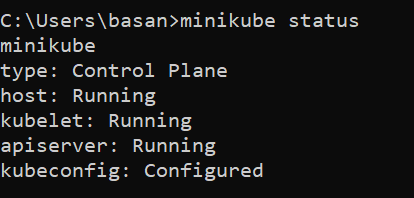
You will get below results , if your minikube starts without any error



Step 7 :

After successfully started minikube , you can verify minikube status

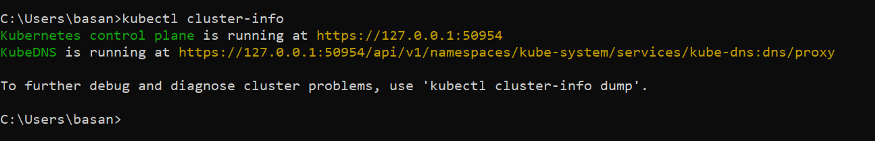
minikube status



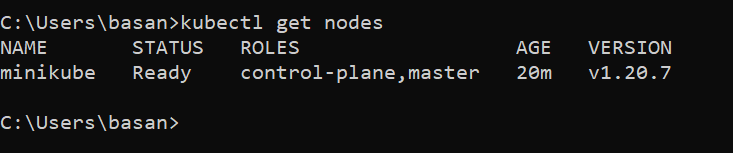
Step 8:

As we know minikube will provide you single node cluster to work with k8s locally so to verify cluster info and node status we can run below commands

kubectl cluster-info



kubectl get node



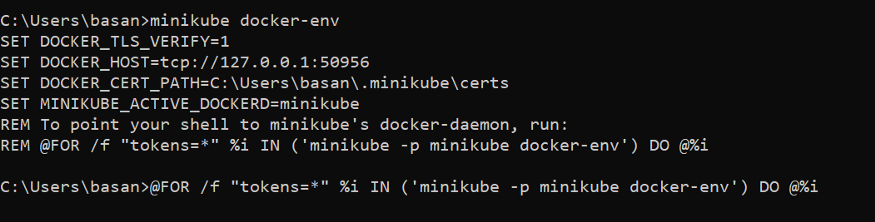
Now we have Kubernetes cluster ready with us to start our local development work

**Deploy Spring boot application to Kubernetes cluster**

Step 1:

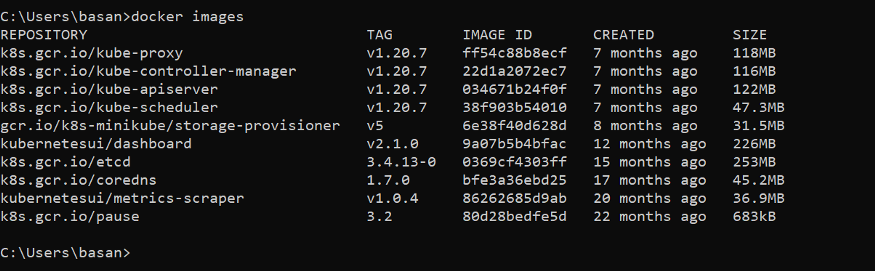
To allow Kubernetes to read your docker repository you need to run below command , so that both will be in sync

**minikube docker-env**



List down all docker images

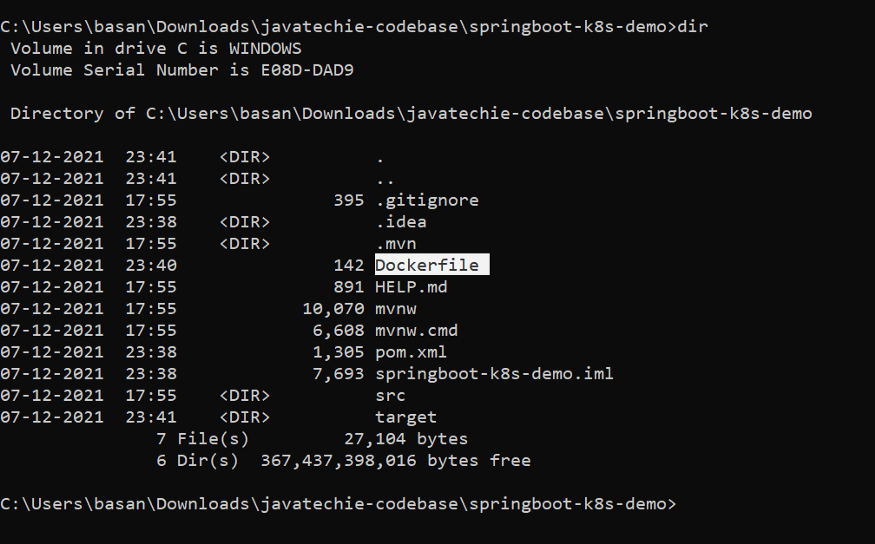
**docker images**



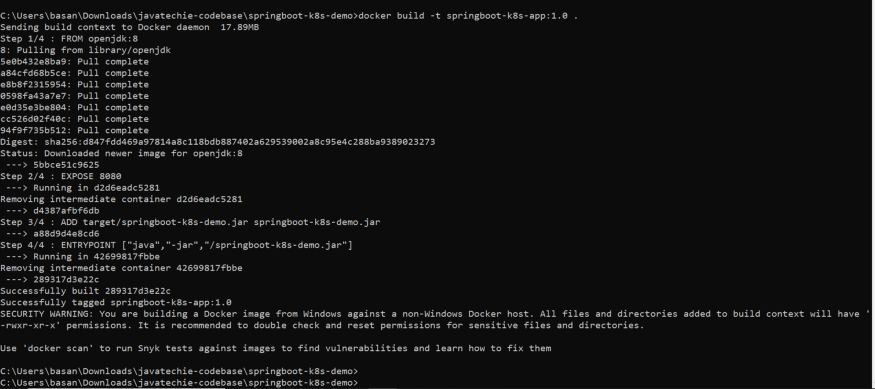
Step 2 :

Create a spring boot project then add Dockerfile and next build a docker image

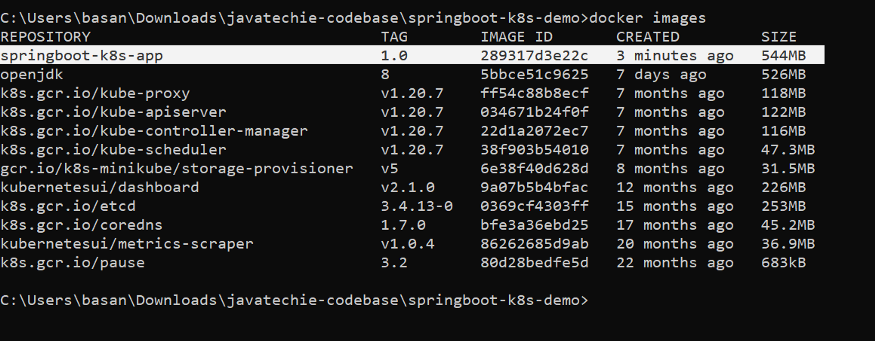
GitHub source code link : <https://github.com/Java-Techie-jt/springboot-k8s-example>



Build docker image



view docker image in k8s



Step 3 :

Create Deployment Object , as we know Deployments are Kubernetes objects that are used for managing pods. we can describe deployment object details using YML file but for this example let’s play with command .

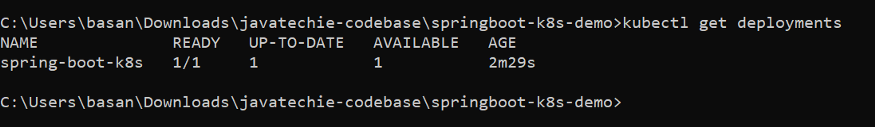
kubectl create deployment spring-boot-k8s --image=springboot-k8s-app:1.0 --port=8080



With above command we are telling to k8s , create a deployment with name spring-boot-k8s and read the docker image springboot-k8s-app:1.0 then next create a pod and run my image inside containers

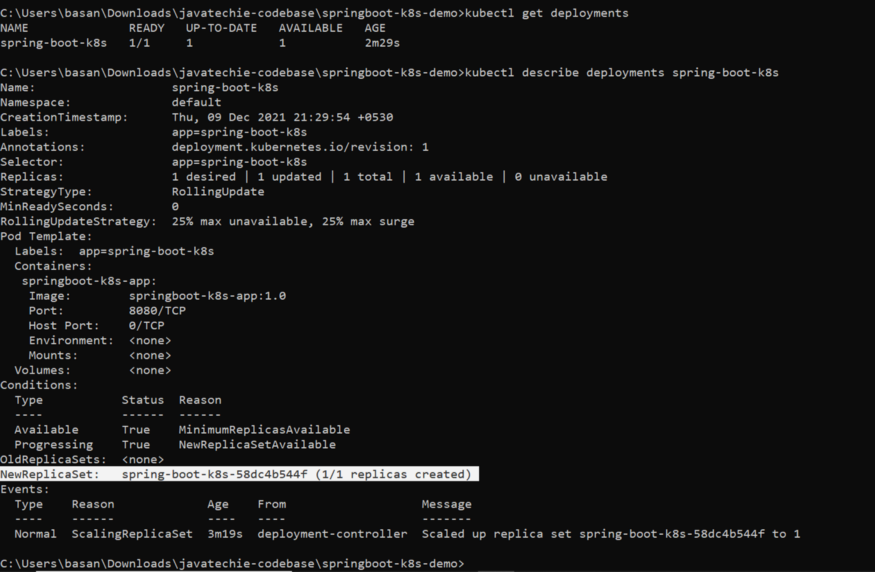
Verify deployment status

kubectl get deployments



Describe deployment object

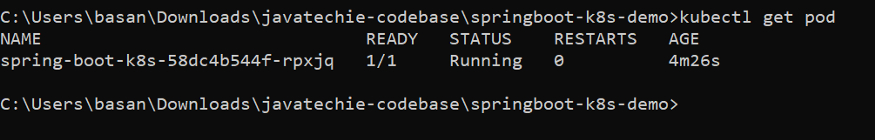
kubectl describe deployment spring-boot-k8s



Now to ensure that Kubernetes successfully pull my docker image and run it inside a pods we can execute below command

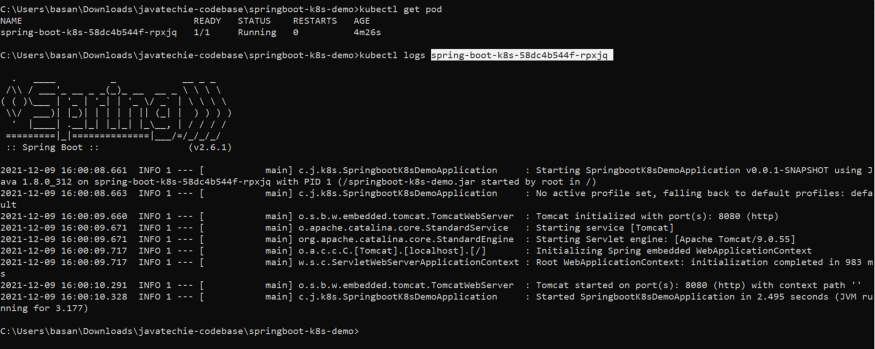
verify pod created

kubectl get pods



Validate docker image running inside pod

kubectl logs spring-boot-k8s-58dc4b544f-rpxjq



We are good now , our application is running inside k8s pods . now to expose this application to outside world we need to create service object .To create a Service object we need to exposes the deployment with specific service type

Step 4 :

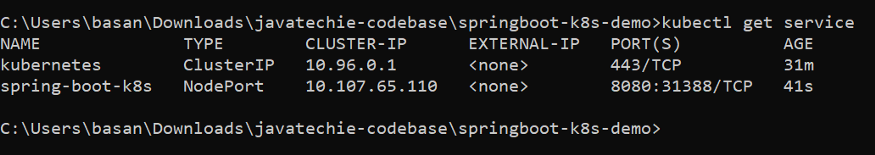
Use below command to create service obsject

kubectl expose deployment spring-boot-k8s --type=NodePort



Once service created you can verify that

kubectl get service **or** kubectl get svc

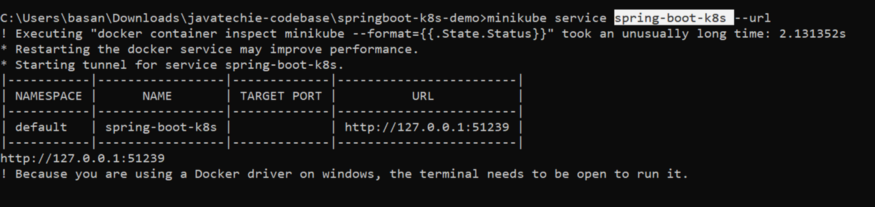


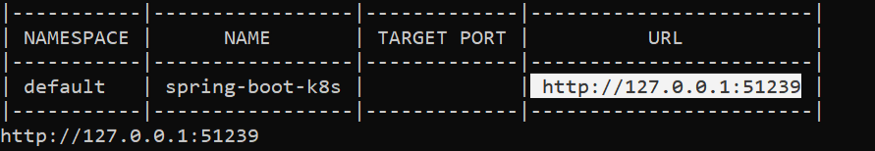
As we know all traffic will come to service and then service will redirect your request to corresponding pods based on available . since we have only one pod we can directly get the service url to access it .

Step 5 :

Start tunnel or get the proxy url of service to access it .

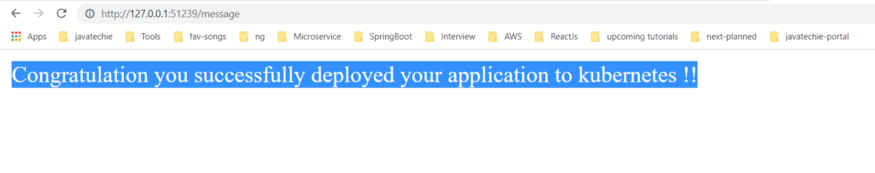
minikube service spring-boot-k8s --url





Step 6 :

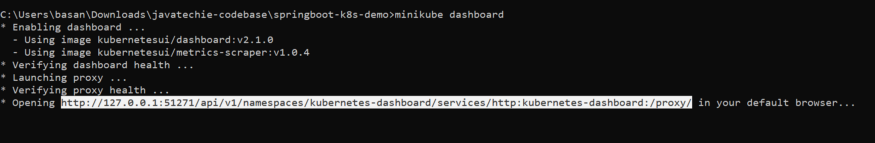
Access the url



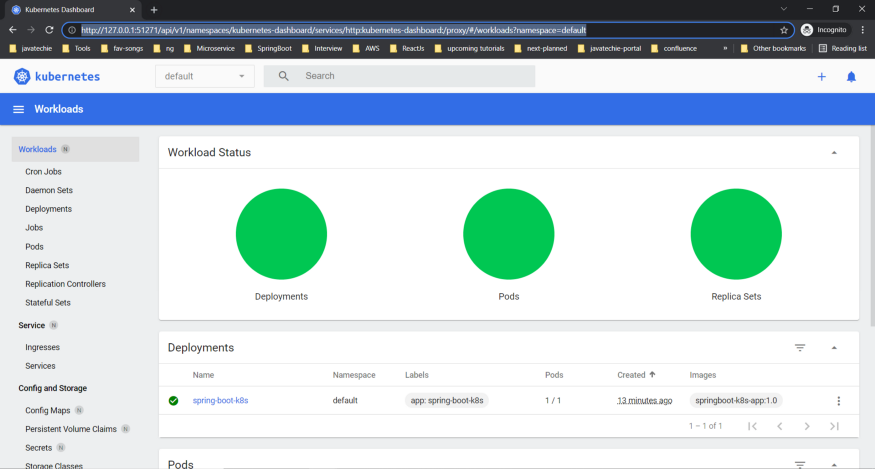
Step 7 :  
You can visualize health of your pods, service and deployment using Kubernetes dashboard

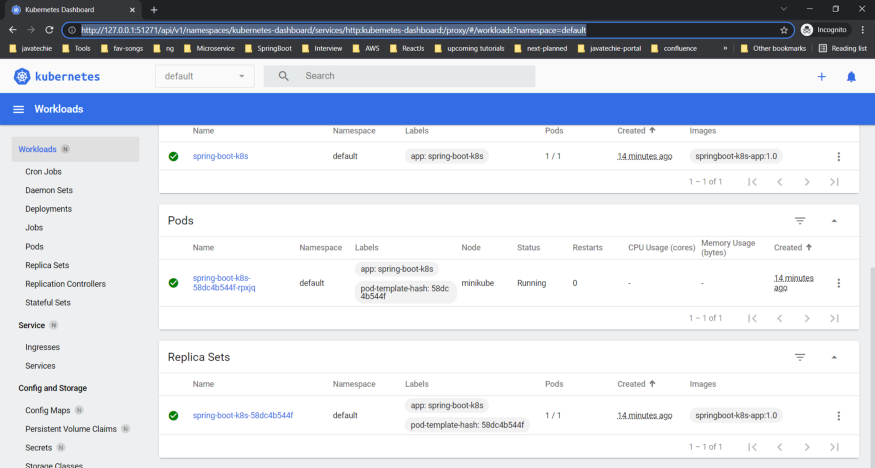
minikube dashboard

This will enable the dashboard add-on, and open the proxy in the default web browser.



You can access the above url to watch your k8s dashboard





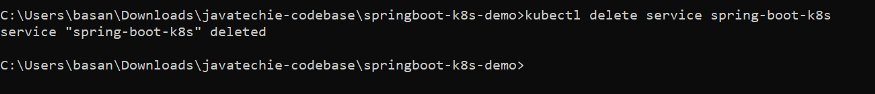
**Deployment completed ……….**

**Clean up local state**

Step 8 :

Delete Service

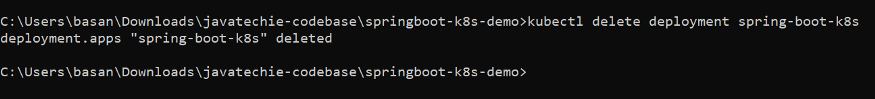
kubectl delete service spring-boot-k8s



Step 9 :

Delete Deployment

kubectl delete deployment spring-boot-k8s

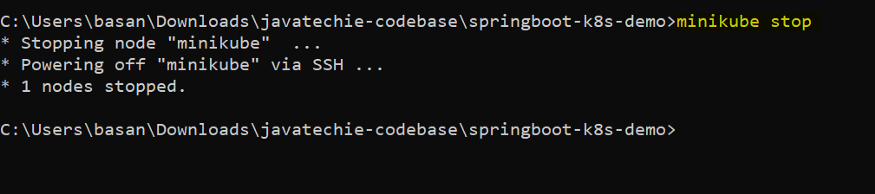


Step 10 :

Stop minikube

minikube stop

Stops a local Kubernetes cluster. This command stops the underlying VM or container, but keeps user data intact. The cluster can be started again with the “start” command.

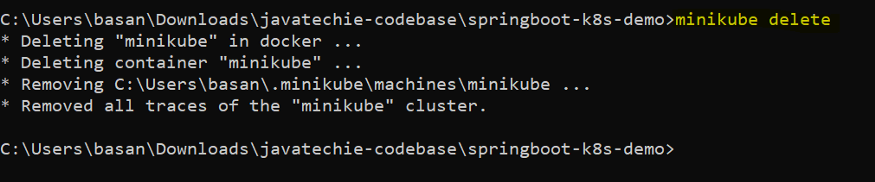


Step 11 :

Delete minikube

minikube delete

Deletes a local Kubernetes cluster. This command deletes the VM, and removes all associated files.



**What we learned ?**

1. How to install minikube in windows OS or How do we setup Kubernetes in windows
2. Kubernetes basic commands
3. How to deploy spring boot application to local Kubernetes cluster .
4. Kubernetes Dashboard and health check .

If you like this article then please do share with your colleagues ..