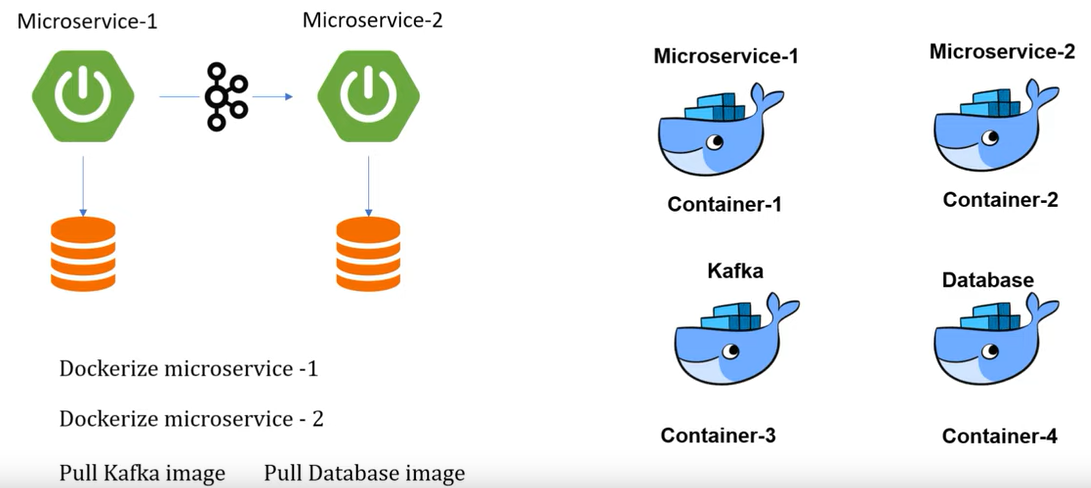
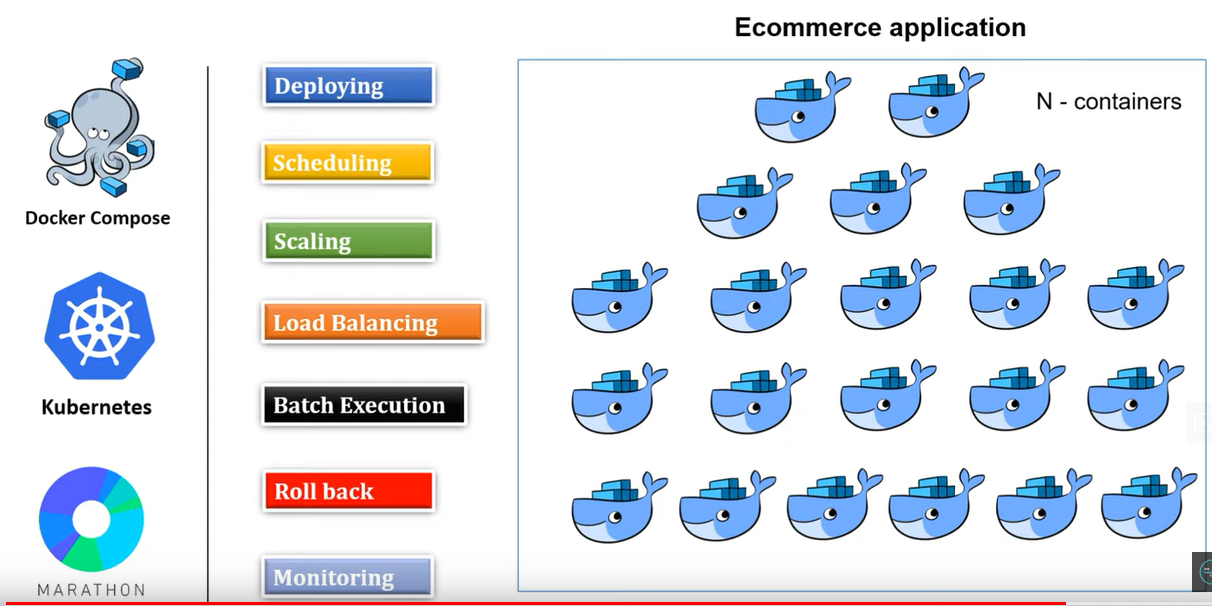
## What and Why Kubernetes?

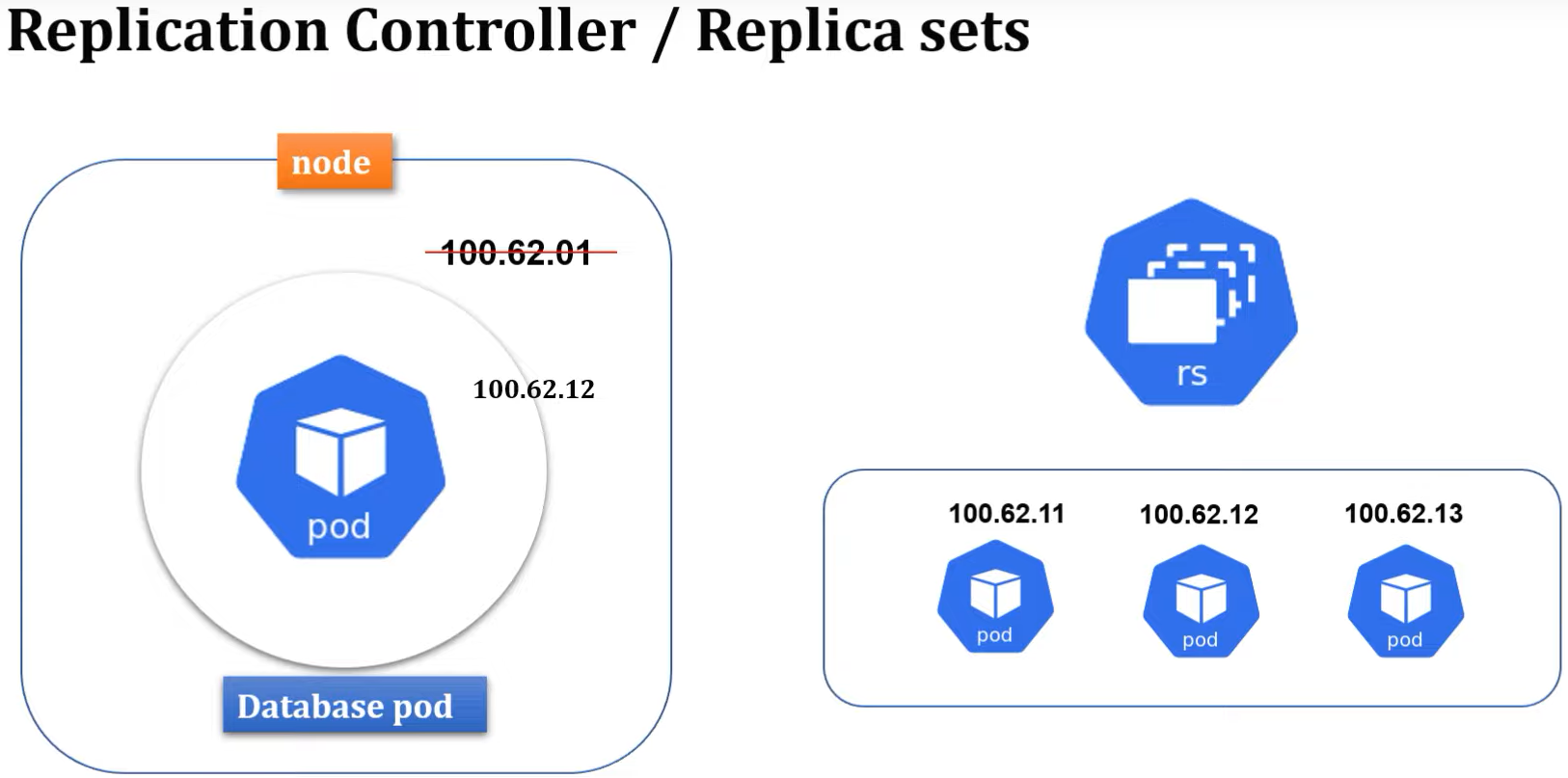
Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications. Container management tool

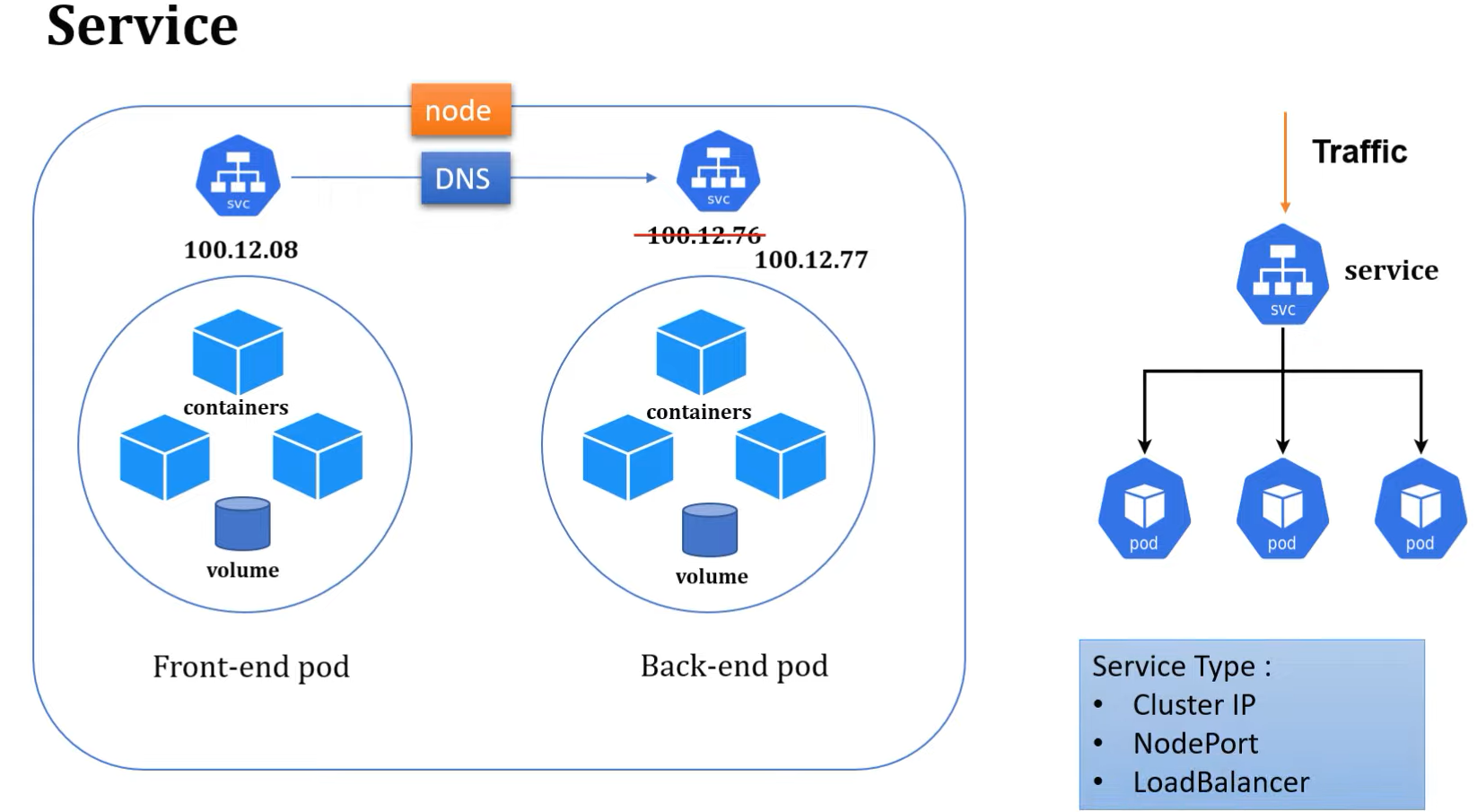




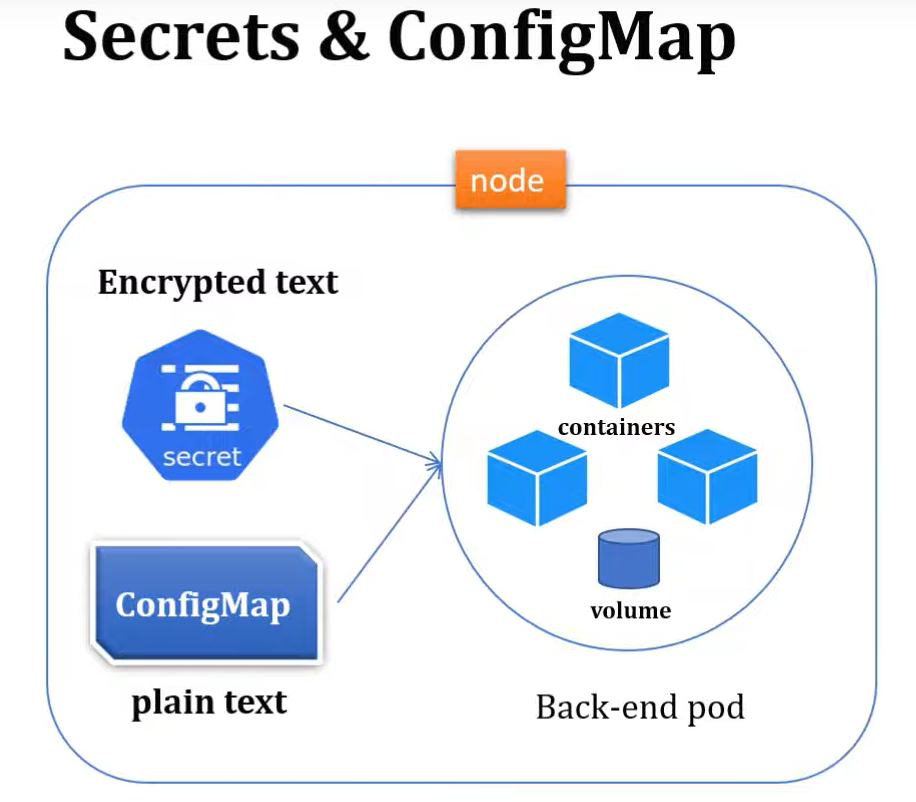
## Kubernetes components?

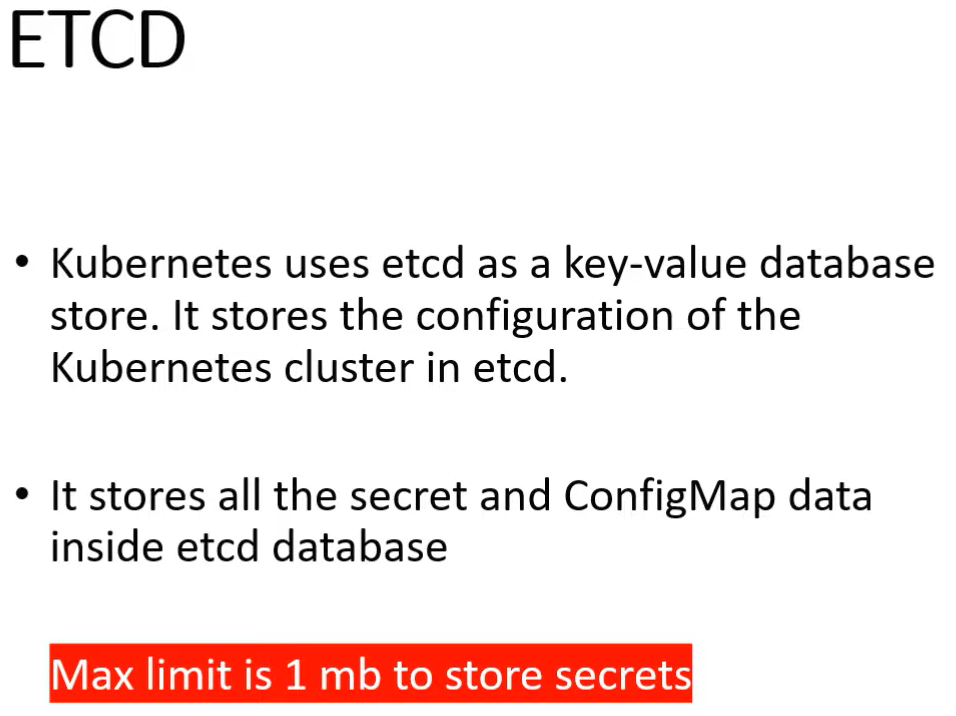












## Install docker desktop and enabled Kubernetes

1. Follow this <https://www.youtube.com/watch?v=xhxmExC9N1U>
2. Check the version: *minikube version*
3. Start minikube: *minikube start --driver=doker*
4. Verify minikube well started: *minikube status*
5. Verify cluster: *kubectl cluster-info*
6. Verify node: *kubectl get node*
7. Allow kubernetes to read our repository: *eval $(minikube docker-env)*

## Run & Deploy Spring boot application on k8s cluster

Resource: <https://www.youtube.com/watch?v=qof9A8k64rA&list=PLVz2XdJiJQxybsyOxK7WFtteH42ayn5i9>

1. Deploy first spring boot application to Kubernetes
2. Create a spring boot application
3. Dockerize your spring boot application by adding a Dockerfile

|  |  |
| --- | --- |
| Build image | docker build -t *spring-boot-k8s:1.0* . |
| Display images | docker images |
| Deploy your images | kubectl create deployment spring-boot-k8s --image=spring-boot-k8s-demo:1.0 --port=8080 |
| Verify your deployment | kubectl get deployment |
| Describe your deployment | kubectl describe deployment *spring-boot-k8s* |
| Check the deployment status pods | kubectl get pods |
| Check the logs of the particular pods | kubectl logs *spring-boot-k8s-694d58c9b4-6gnq5* |
| Expose your deployment | kubectl expose deployment *spring-boot-k8s* –type=NodePort |
| Verify your service | kubectl get service |
| Get url | minikube service *spring-boot-k8s* --url |
| Get a dashboard. Copy the link and paste it on the browser | minikube dashboard |
| Delete all the component | kubectl delete service *spring-boot-k8s*  kubectl delete deployment *spring-boot-k8s* |
| Stop minikube | minikube stop |
| Delete minikube | minikube delete |

Resources:

<https://www.youtube.com/watch?v=xhxmExC9N1U>

## Run & Deploy Spring Boot Application in K8s Cluster using yaml configuration

1. Create a spring boot application
2. Dockerize your spring boot application by adding a Dockerfile
3. Build the image: docker build -t *spring-boot-k8s:1.0* .
4. Add the deployment file
5. Create deployment object: kubectl apply -f *k8s-deployment.yaml*
6. Verify deployment: kubectl get deployments
7. Delete deployment: kubectl delete deployment *spring-boot-k8s*
8. Verify pods: kubectl get pods
9. Logs a pods: kubectl logs *spring-boot-k8s-79844c64cd-5sr6k*
10. Addservice yaml
11. Create the service yaml: kubectl apply -f *k8s-service.yaml*
12. Verify service: kubectl get service
13. Delete deployment: kubectl delete service *spring-boot-k8s*
14. Verify the node IP : kubectl get node -o wide
15. Get the internal or minikube Ip: minikube ip
16. Get service url : minikube service *spring-boot-k8s* --url
17. Display the dashboard: minikube dashboard

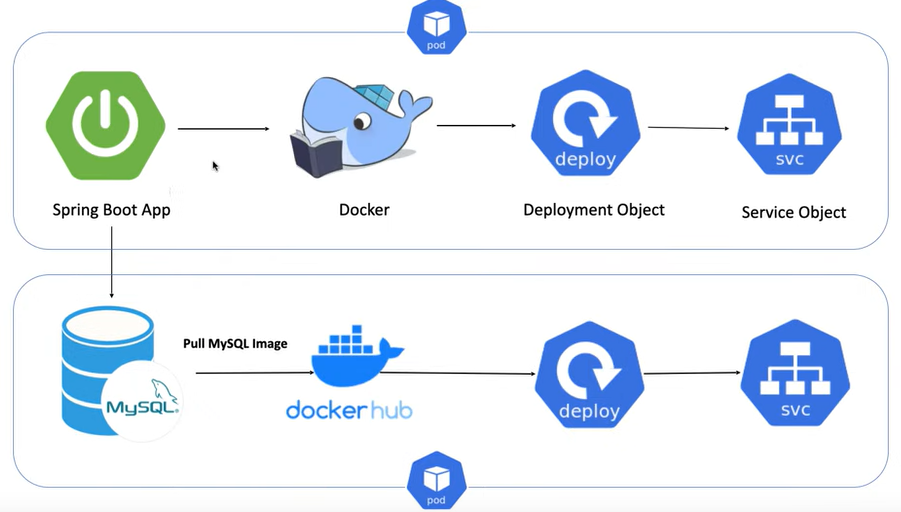
Resources:

<https://www.youtube.com/watch?v=7o7e8OAAWyg>

## Run & Deploy Spring Boot CRUD Application With MySQL on K8S

Resource: <https://www.youtube.com/watch?v=pIPji3_rYPY&list=PLVz2XdJiJQxybsyOxK7WFtteH42ayn5i9&index=7>

### Application Structured on the K8s



### Install minikube and kubectl

* <https://minikube.sigs.k8s.io/docs/start/>
* <https://kubernetes.io/docs/tasks/tool/>

1. Check the version: *minikube version*
2. Start minikube: *minikube start --driver=docker (installed docker desktop before)*

Or Start minikube: *minikube start --driver=virtualbox (installed virtualbox before, resource* [*https://www.youtube.com/watch?v=M64aA5Icuk4*](https://www.youtube.com/watch?v=M64aA5Icuk4) *)*

1. Verify minikube well started: *minikube status*
2. Verify cluster: *kubectl cluster-info*
3. Verify node: *kubectl get node*
4. Allow kubernetes to read our repository: *eval $(minikube docker-env)*

### Implementation based on that structure

1. Create your spring boot application and make sure it worked locally
2. Dockerize you application by adding a docker file
3. Add deployment yaml file (db-deployment.yml). this fille will help to pull mysql image from docker-hub and deployed it.
4. Create the deployment object by executing db-deployment.yml file :

kubectl apply -f db-deployment.yaml

kubectl get deployments

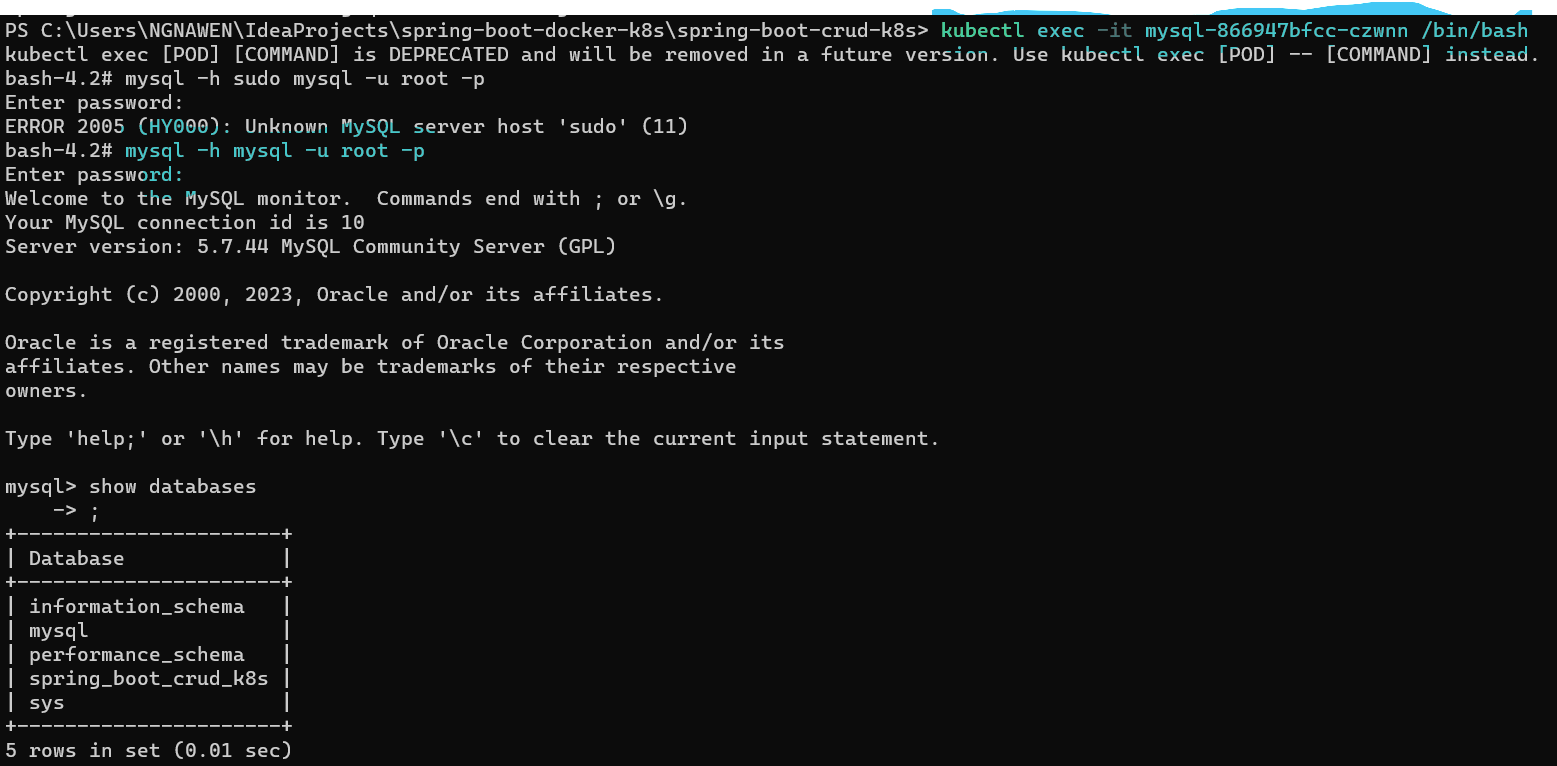
kubectl get pods

kubectl logs mysql-866947bfcc-czwnn

kubectl exec -it mysql-866947bfcc-czwnn /bin/bash

mysql -h mysql -u root -p

enter your password (root in our case)



In this step, mysql running perfectly on my pod.

1. Update the application.yml file. Make it dynamically
2. Build the docker image:

docker build -t spring-boot-crud-k8s:1.0 .

docker images

1. add deployment yaml file to your application (app-deployment.yaml)
2. Create the deployment object by executing app-deployment.yaml

kubectl apply -f app-deployment.yaml

kubectl get deployments

kubectl get services spring-boot-crud-k8s-svc

minikube service spring-boot-crud-k8s-svc

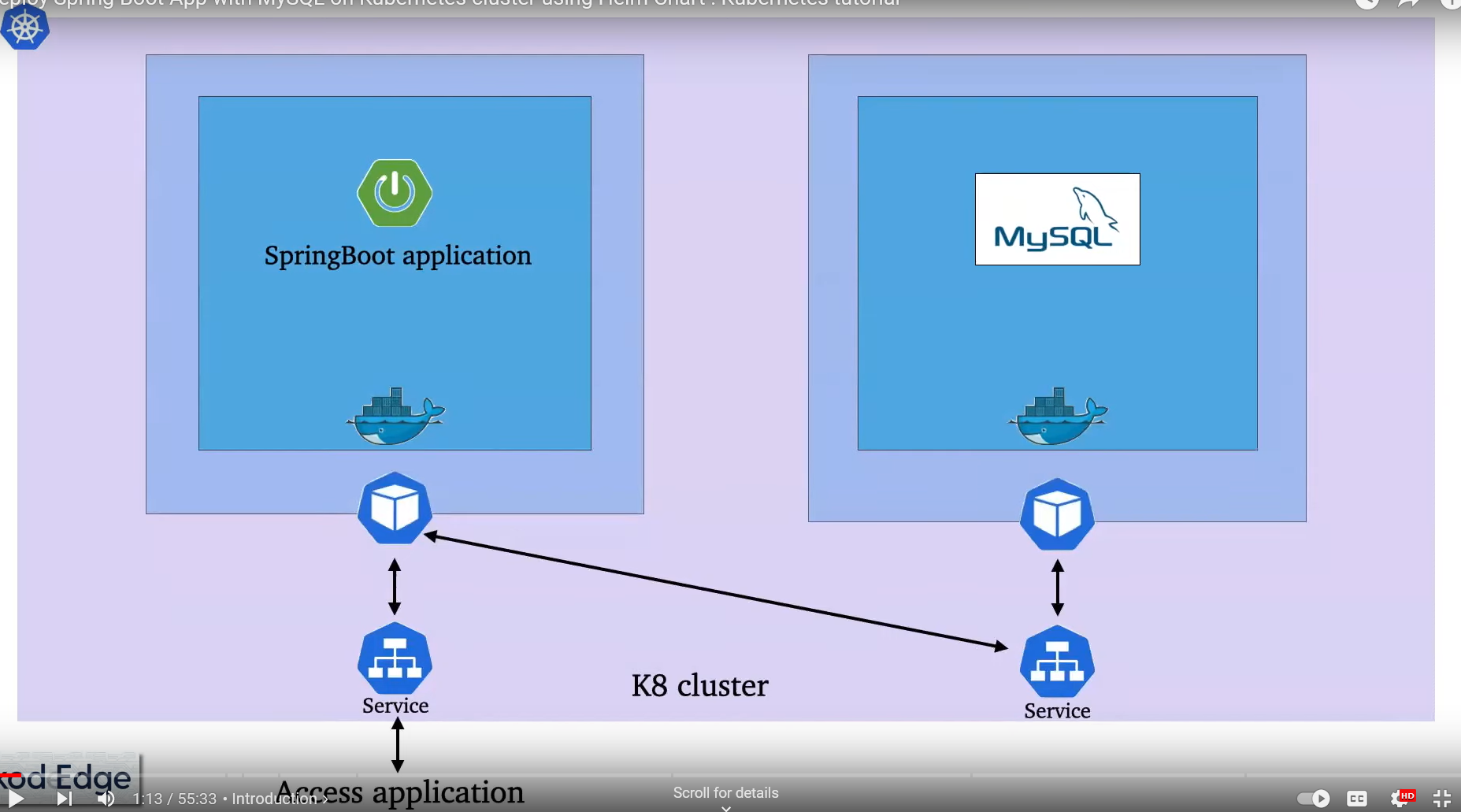
eval $(minikube docker-env)

## Deploy Spring Boot App with MySQL on Kubernetes cluster using Helm Chart: Kubernetes tutorial

### Resources

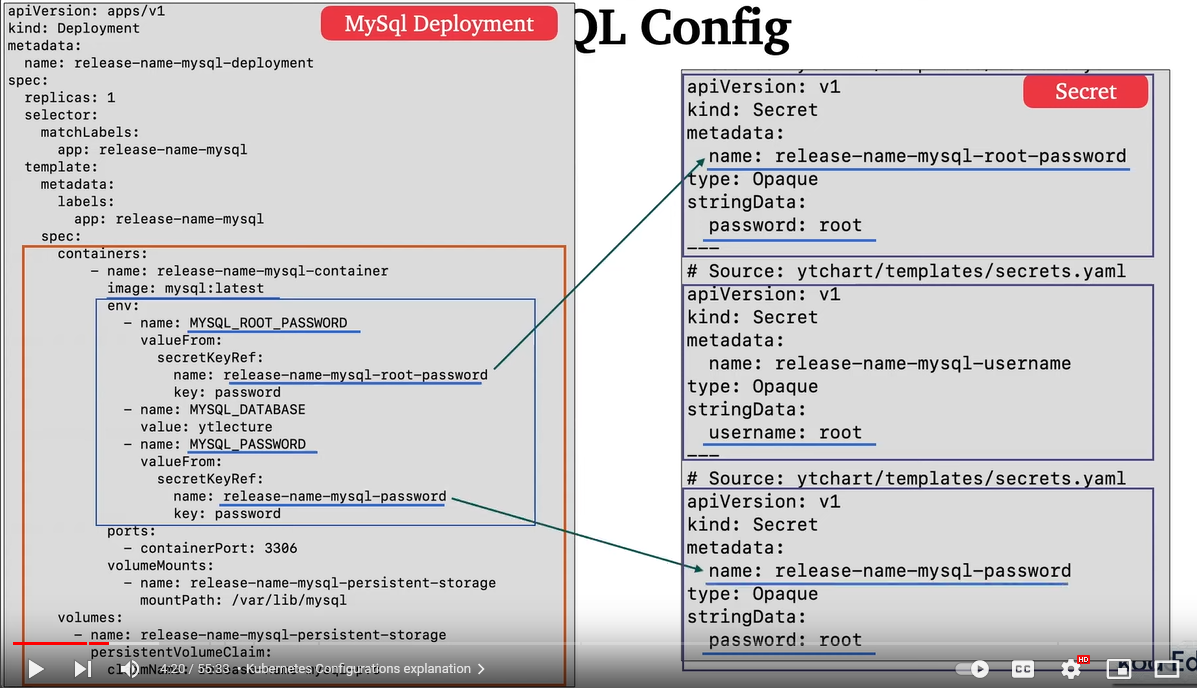
* <https://www.youtube.com/watch?v=yet0-x8Ab6c> ;
* <https://github.com/kodedge-swapneel/spring-mysql-demo>

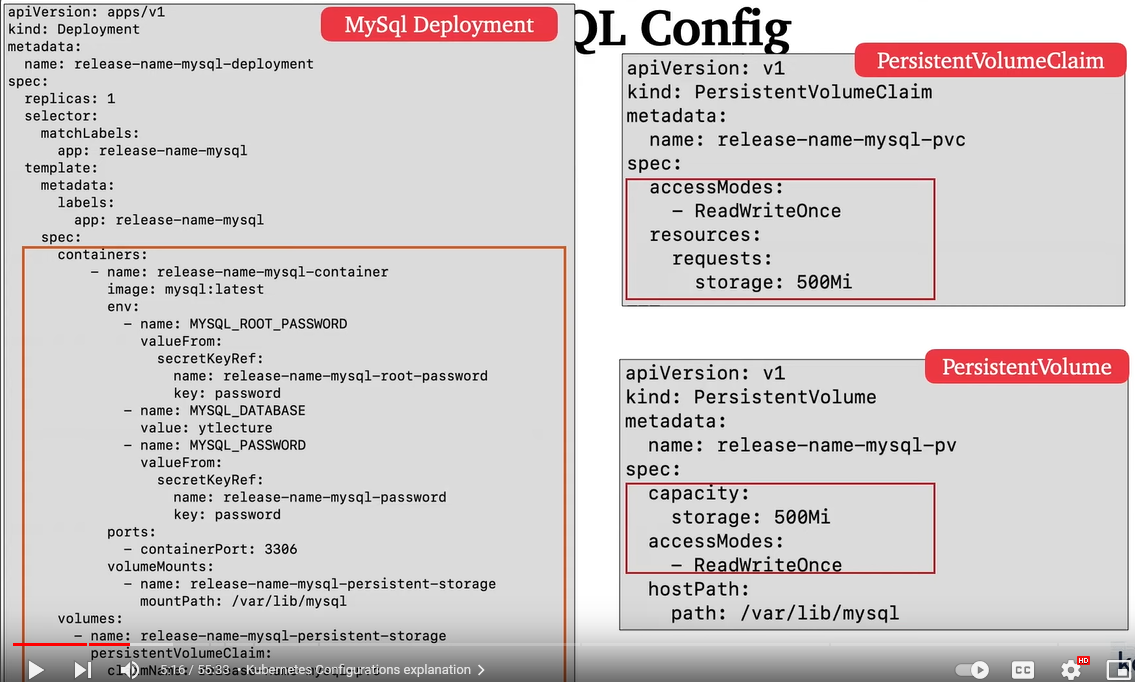
### K8s cluster architecture

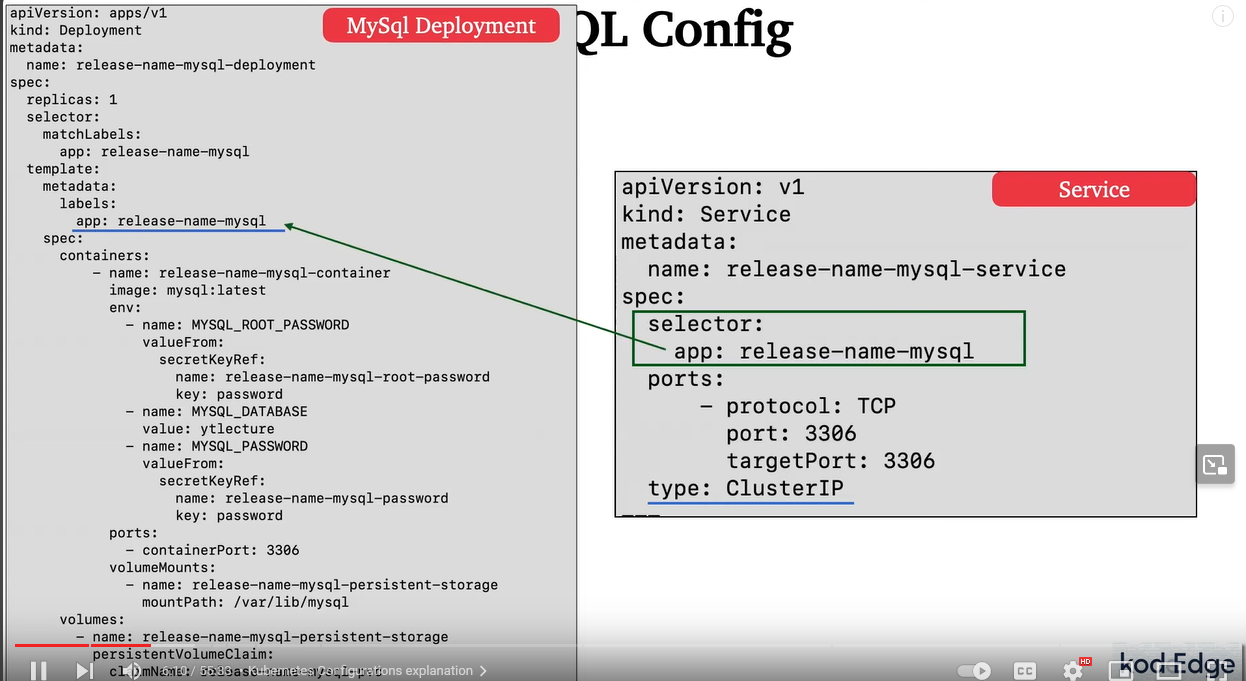




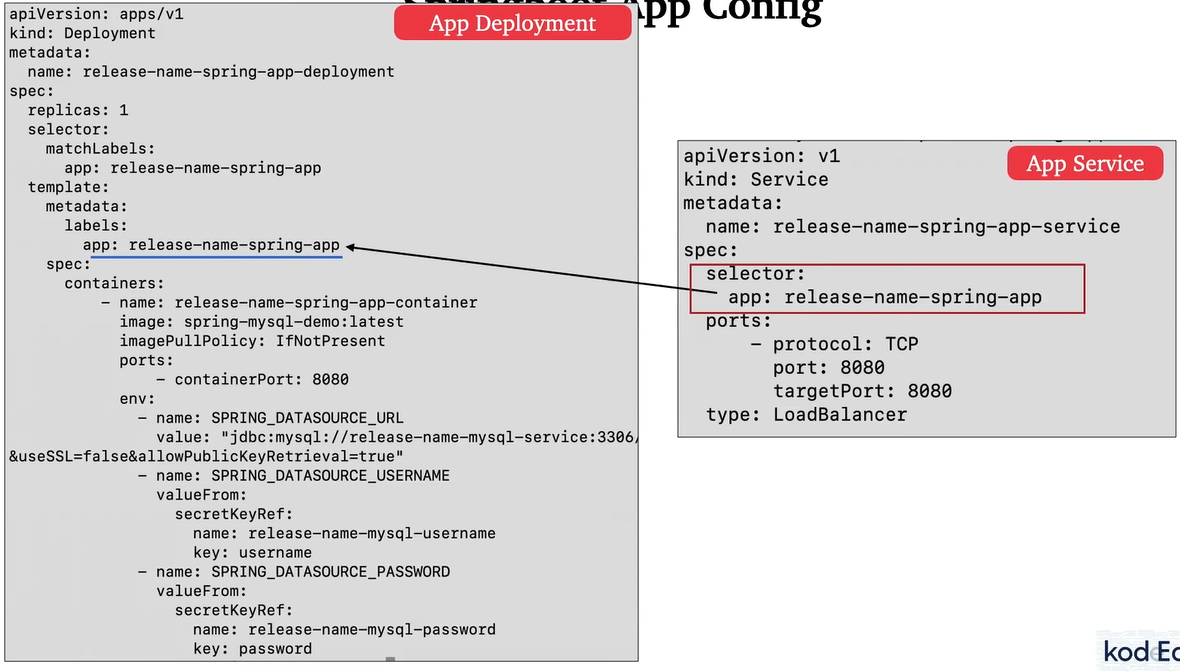
### Mysql config



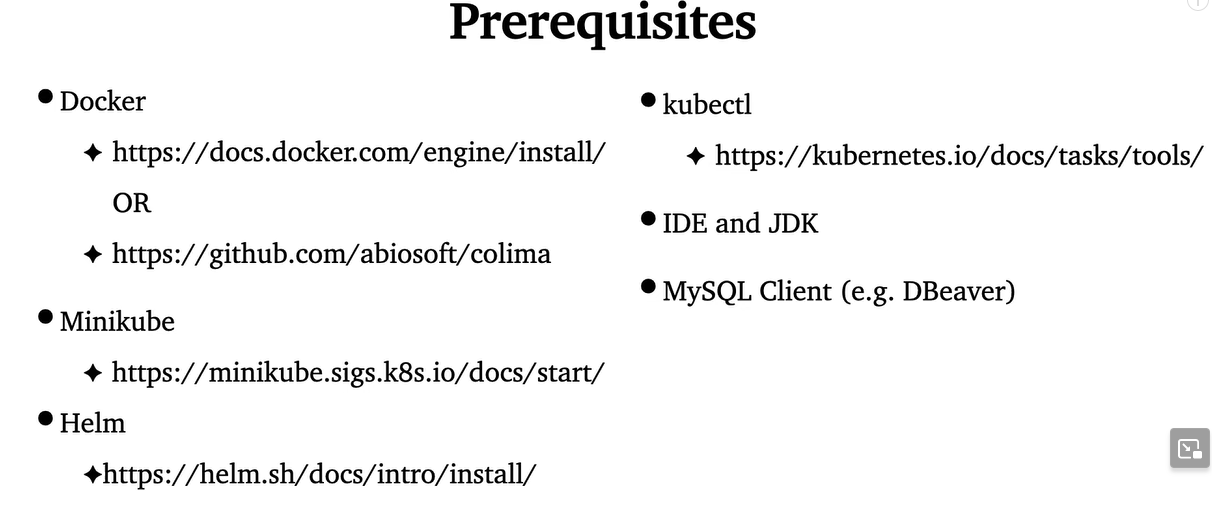




### Spring boot app config



### Prerequisites



### Start the docker desktop on your machine

### Download/pull the mysql image and run the mysql container

=>docker pull mysql:latest

=>docker run -d --name mysql-container -e MYSQL\_ROOT\_PASSWORD=root -p 3306:3306 mysql:latest

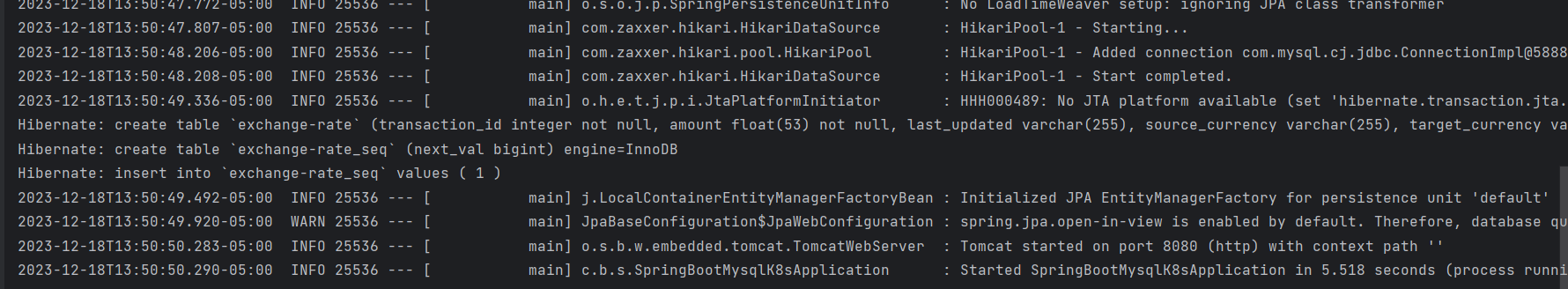
=>docker ps

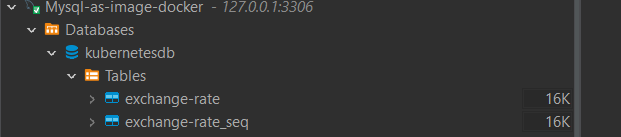
### Create your spring boot application and make sure it works locally

Make sure your application.yaml/application.properties is correctly configure to connect to MySQL container running.

### Run your application

If all things work successfully, you will get those responses





### Dockerize your application by adding a Dockerfile

Dockerfile is a text document with a series of commands used to build a Docker image. It describes how to build Docker images.

mvn clean install (build application to genere the jar file)

### Create a docker compose file

Docker-compose is a tool for defining and running multi-container applications. docker-compose is used to run Docker containers.

### Create a .env file

This file is used to keep the information’s to connect to database.

### Deploy the application to the docker

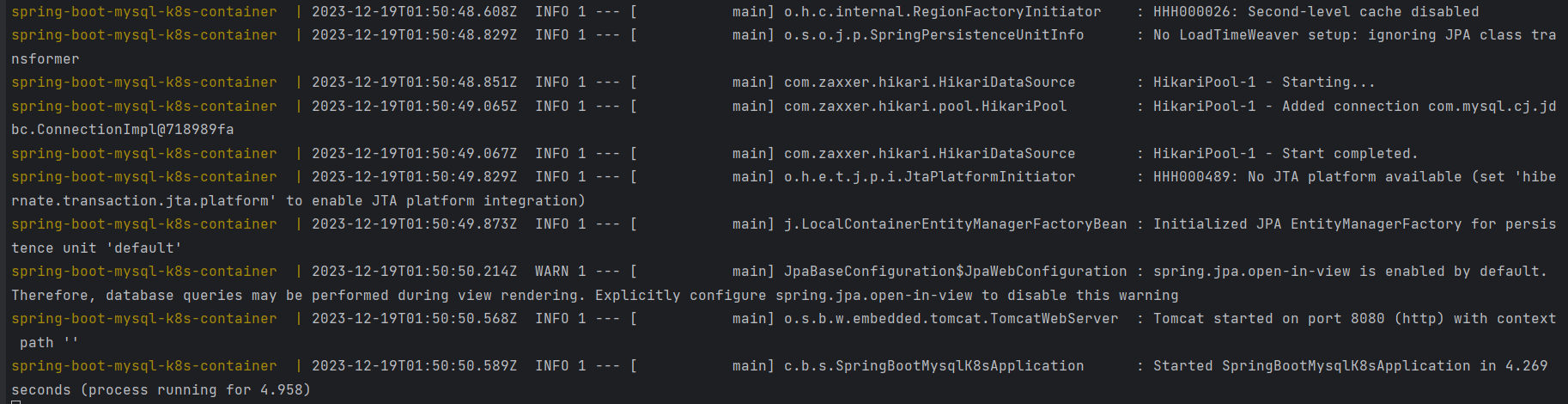
Stop any component listening to MySQL port (3306 usually): docker stop *mysql-container* and any app listening to port 8080

docker build -t spring-boot-mysql-k8s:latest .

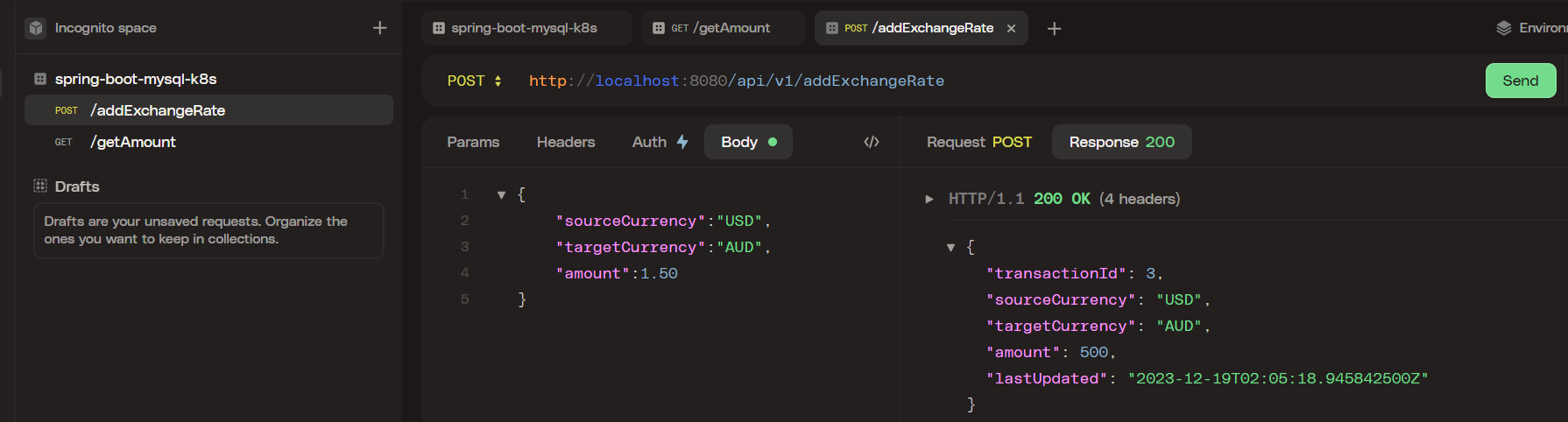
docker images

docker-compose up (<https://stackoverflow.com/questions/29289785/how-to-install-docker-compose-on-windows> , <https://stackoverflow.com/questions/64307077/docker-compose-only-one-usage-of-each-socket-address-protocol-network-address> ) is using to run your application

If all things work successfully, you will get those responses:



### Test your application using postman or HTTPie or …



### Kubernetes configuration using Helm

* helm create *ytchart* (command to create k8s config)
* Once the ytchat is created,delete all the file inside the templates folder,
* update the value.yaml,
* create the secrets.yaml,
* create the mysql-deployment.yaml,
* create the mysql-pv.yaml(mysql persistence volume),
* create the mysql-pvc.yaml(mysql persistence volume claim),
* create the spring-app-deployment.yaml,
* create the mysql-service.yaml,
* create the \_helpers.tpl

### Deploy application into the Kubernetes cluster/Minikube

=>minikube start --driver=docker (start minikube for creating the cluster)

=>helm template *ytchart/* (display all the k8s config)

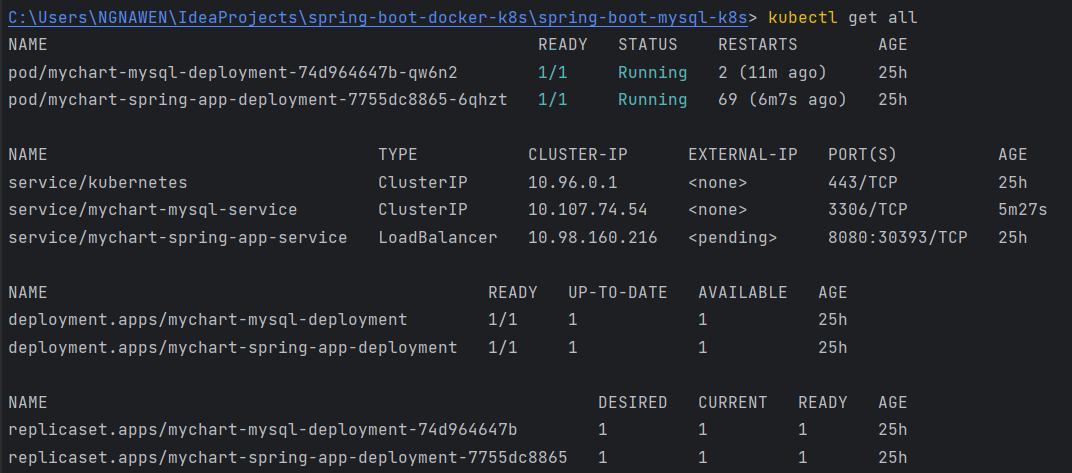
=>eval '$(minikube docker-env)' (or on windows run: minikube docker-env and minikube -p minikube docker-env --shell powershell | Invoke-Expression)

=>docker build -t spring-boot-mysql-k8s:latest . (build docker image)

=>minikube image ls (display images)

=>helm install mychart ytchart (deploy k8s config)

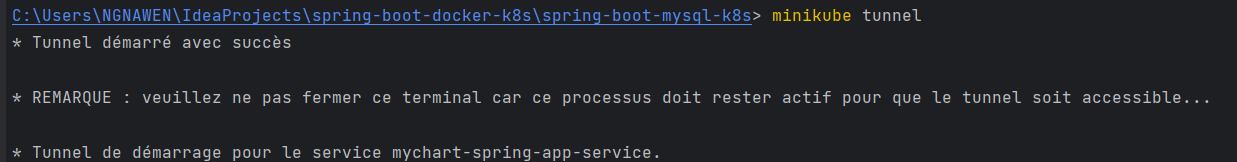
=>kubectl get all (display all the configuration)



=>kubectl get services

=>minikube service *mychart-mysql-service* --url

=>minikube tunnel (enable the app service to be accessible outside of the cluster)



=>kubectl get pods

=>kubectl logs -f *pod\_name (*display the logs of your application)