**1st question**

**install the following microservices of the project.**

https://github.com/in28minutes/spring-boot-examples/tree/master/spring-boot-basic-microservice

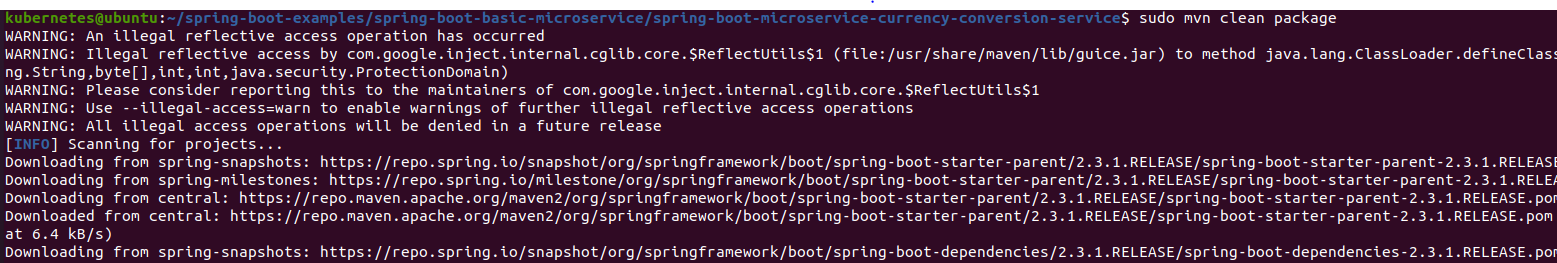
A microservice is just a computer program which runs on a server or a virtual computing instance and responds to network requests.

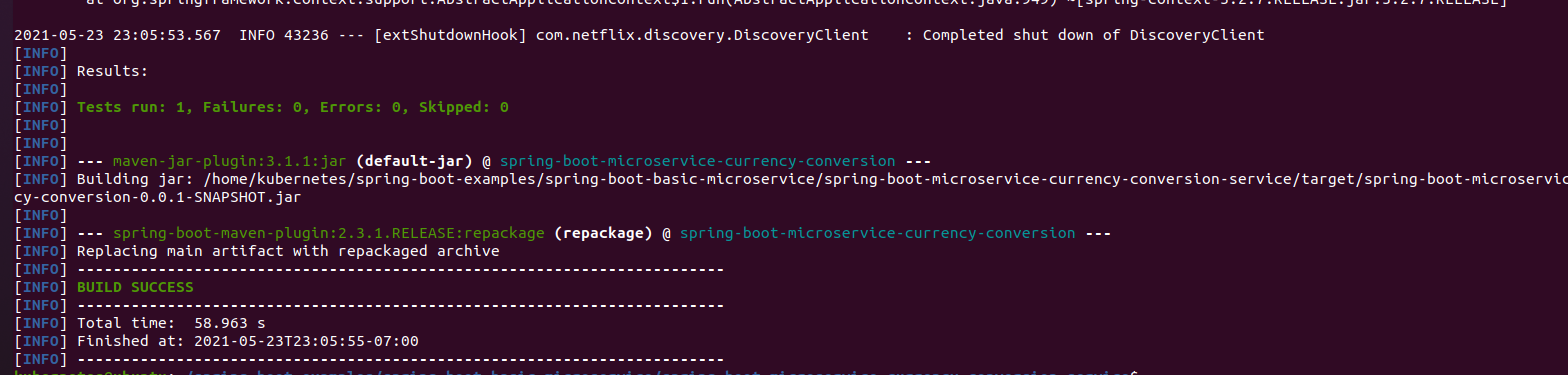
$cd spring-boot-examples/spring-boot-basic-microservice/spring-boot-microservice-currency-conversion-service

$sudo nano dockerfile

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | FROM openjdk:8-jdk-alpine | |  | VOLUME /tmp | |  | EXPOSE 8100 | |  | ADD target/\*.jar app.jar | |  | ENV JAVA\_OPTS="" | |  | ENTRYPOINT [ "sh", "-c", "java $JAVA\_OPTS -Djava.security.egd=file:/dev/./urandom -jar /app.jar" ] | |

$sudo mvn clean package





$sudo docker build -t spring-boot-microservice-currency-conversion-service:1.0-SNAPSHOT spring-boot-microservice-currency-conversion-service/.

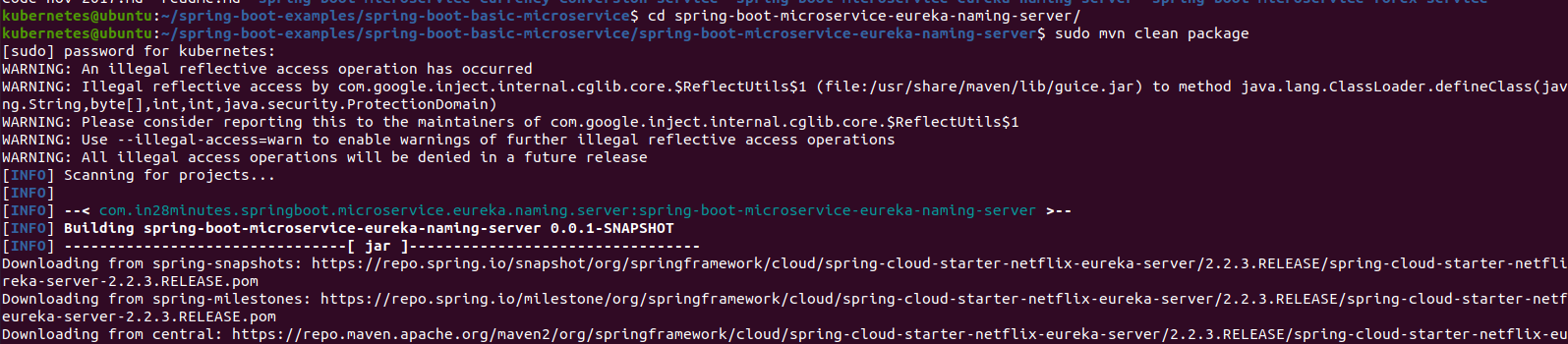


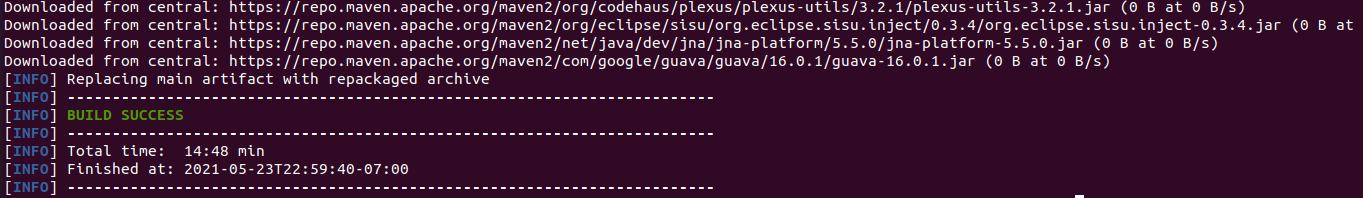
$cd spring-boot-examples/spring-boot-basic-microservice/spring-boot-microservice-eureka-naming-server

$sudo nano dockerfile

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | FROM openjdk:8-jdk-alpine | |  | VOLUME /tmp | |  | EXPOSE 8761 | |  | ADD target/\*.jar app.jar | |  | ENV JAVA\_OPTS="" | |  | ENTRYPOINT [ "sh", "-c", "java $JAVA\_OPTS -Djava.security.egd=file:/dev/./urandom -jar /app.jar" ] | |

$sudo mvn clean package





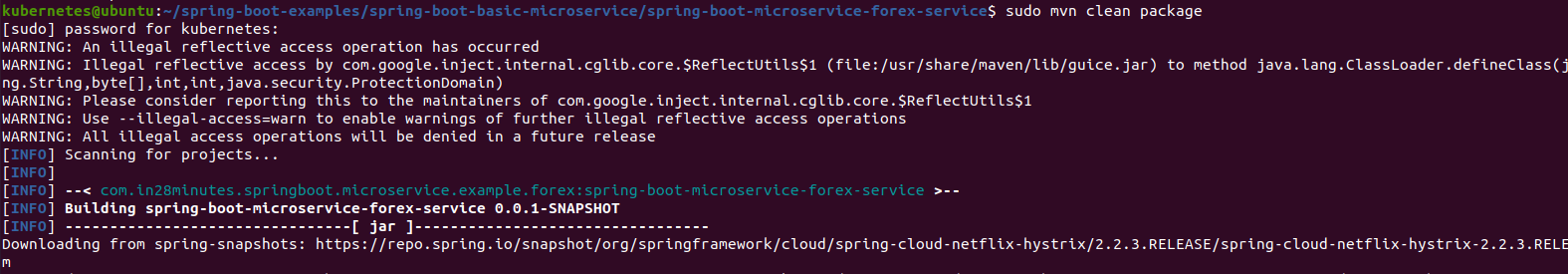
$sudo sudo docker build -t spring-boot-microservice-eureka-naming-server:1.0-SNAPSHOT spring-boot-microservice-eureka-naming-server/.

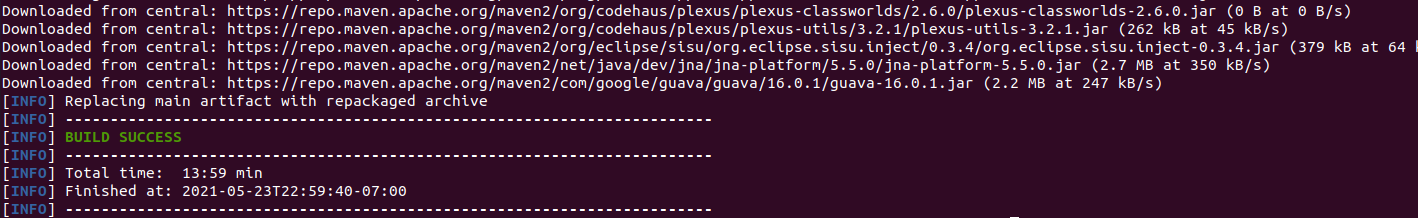


$cd spring-boot-examples/spring-boot-basic-microservice/spring-boot-microservice-forex-service

$Sudo nano dockerfile

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | FROM openjdk:8-jdk-alpine | |  | VOLUME /tmp | |  | EXPOSE 8000 | |  | ADD target/\*.jar app.jar | |  | ENV JAVA\_OPTS="" | |  | ENTRYPOINT [ "sh", "-c", "java $JAVA\_OPTS -Djava.security.egd=file:/dev/./urandom -jar /app.jar" ] | |

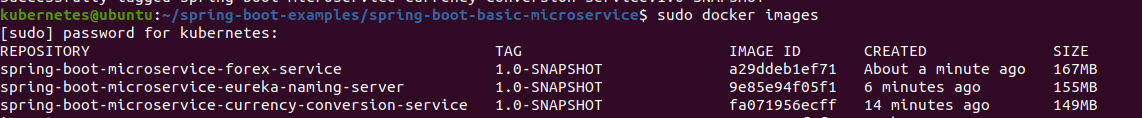
$sudo sudo mvn clean package 

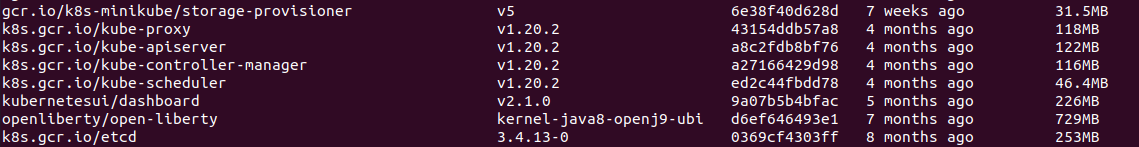


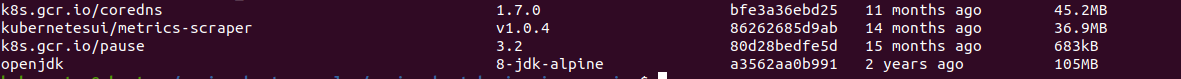
$sudo docker build -t spring-boot-microservice-forex-service:1.0-SNAPSHOT spring-boot-microservice-forex-service/.



$sudo docker images



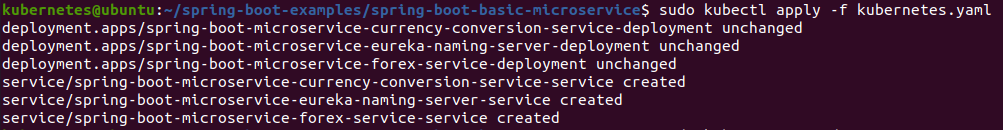




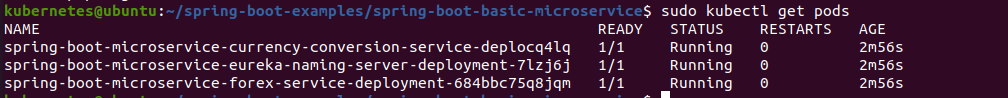
$sudo nano Kubernetes.yaml

|  |
| --- |
| apiVersion: apps/v1  kind: Deployment  metadata:  name: spring-boot-microservice-currency-conversion-service-deployment  labels:  app: spring-boot-microservice-currency-conversion-service  spec:  selector:  matchLabels:  app: spring-boot-microservice-currency-conversion-service  template:  metadata:  labels:  app: spring-boot-microservice-currency-conversion-service  spec:  containers:  - name: spring-boot-microservice-currency-conversion-service-container  image: spring-boot-microservice-currency-conversion-service:1.0-SNAPSHOT  ports:  - containerPort: 8000  ---  apiVersion: apps/v1  kind: Deployment  metadata:  name: spring-boot-microservice-eureka-naming-server-deployment  labels:  app: spring-boot-microservice-eureka-naming-server  spec:  selector:  matchLabels:  app: spring-boot-microservice-eureka-naming-server  template:  metadata:  labels:  app: spring-boot-microservice-eureka-naming-server  spec:  containers:  - name: spring-boot-microservice-eureka-naming-server-container  image: spring-boot-microservice-eureka-naming-server:1.0-SNAPSHOT  ports:  - containerPort: 8010  ---  apiVersion: apps/v1  kind: Deployment  metadata:  name: spring-boot-microservice-forex-service-deployment  labels:  app: spring-boot-microservice-forex-service  spec:  selector:  matchLabels:  app: spring-boot-microservice-forex-service  template:  metadata:  labels:  app: spring-boot-microservice-forex-service  spec:  containers:  - name: spring-boot-microservice-forex-service-container  image: spring-boot-microservice-forex-service:1.0-SNAPSHOT  ports:  - containerPort: 8020  ---  apiVersion: v1  kind: Service  metadata:  name: spring-boot-microservice-currency-conversion-service-service  spec:  type: NodePort  selector:  app: spring-boot-microservice-currency-conversion-service  ports:  - protocol: TCP  port: 8000  targetPort: 8000  nodePort: 30000  ---  apiVersion: v1  kind: Service  metadata:  name: spring-boot-microservice-eureka-naming-server-service  spec:  type: NodePort  selector:  app: spring-boot-microservice-eureka-naming-server  ports:  - protocol: TCP  port: 8010  targetPort: 8010  nodePort: 30100  ---  apiVersion: v1  kind: Service  metadata:  name: spring-boot-microservice-forex-service-service  spec:  type: NodePort  selector:  app: spring-boot-microservice-forex-service  ports:  - protocol: TCP  port: 8020  targetPort: 8020  nodePort: 30200 |

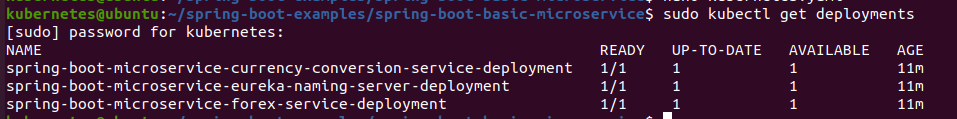
$sudo kubectl apply -f kubernetes.yaml



$sudo kubectl get pods



$sudo kubectl get deployments



**2nd Question**

# **Deploying WordPress with Nginx and MySQL with Caching mechanism**

### **Prerequisites**

Kubernetes cluster with nginx ingress installed

Kubernetes <= 0.10

### Installing

A step-by-step series of examples that tell you how to get a setup WordPress env running

**Build custom WordPress docker**

**Dockerfile**

|  |
| --- |
| FROM wordpress:5-php7.2  COPY ./ /usr/src/wordpress/  EXPOSE 80  EXPOSE 9000 |

### **If WordPress taking Db connection values from environment variable**

**Edit & apply wordpress-configmap.yaml**

$kubectl apply -f wordpress-configmap.yaml

### **If want to implement fast-cgi with nginx ingress**

**Edit & apply fast-cgi-configmap.yaml & ingress.yaml**

$kubectl apply -f fast-cgi-configmap.yaml

$kubectl apply -f ingress.yaml

### **SSL/TLS certificate**

**If want to use SSL/TLS certificate setup cert-manager and apply clusterissuer**

**Edit & apply cluster-issuer.yaml**

$sudo kubectl apply -f cluster-issuer.yaml

### **WordPress service deployment**

**WordPress deployment will create One with Two container inside it (Nginx + Wordpress php-fpm)**

**Edit & apply wordpress-deployment.yaml & wordpress-service.yaml**

$sudo kubectl apply -f wordpress-service.yaml

$sudo kubectl apply -f wordpress-deployment.yaml

## 