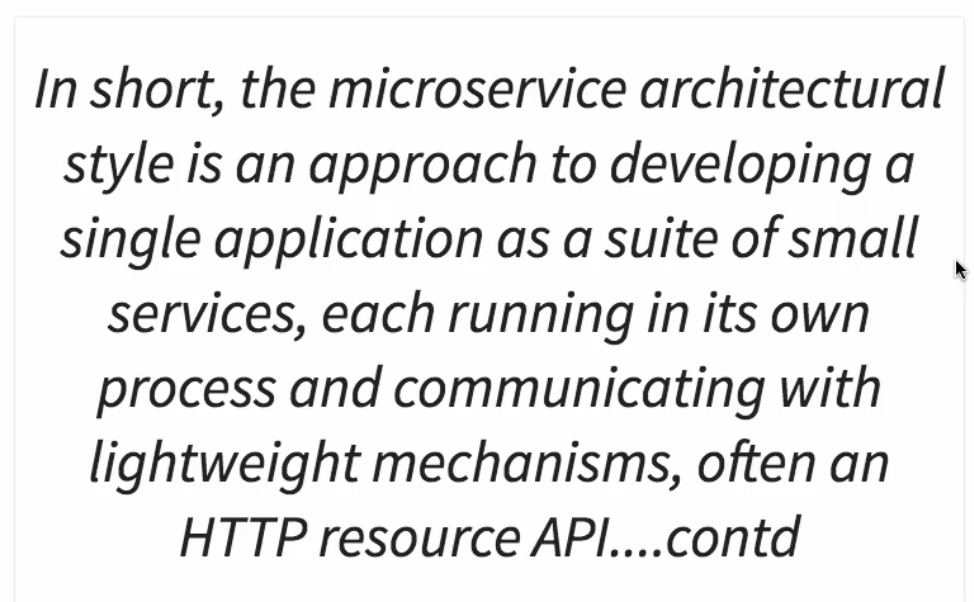
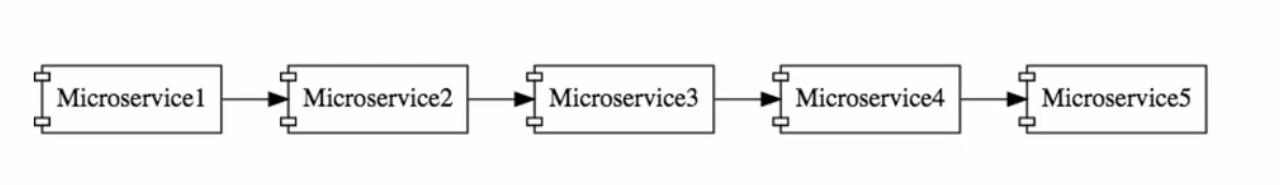
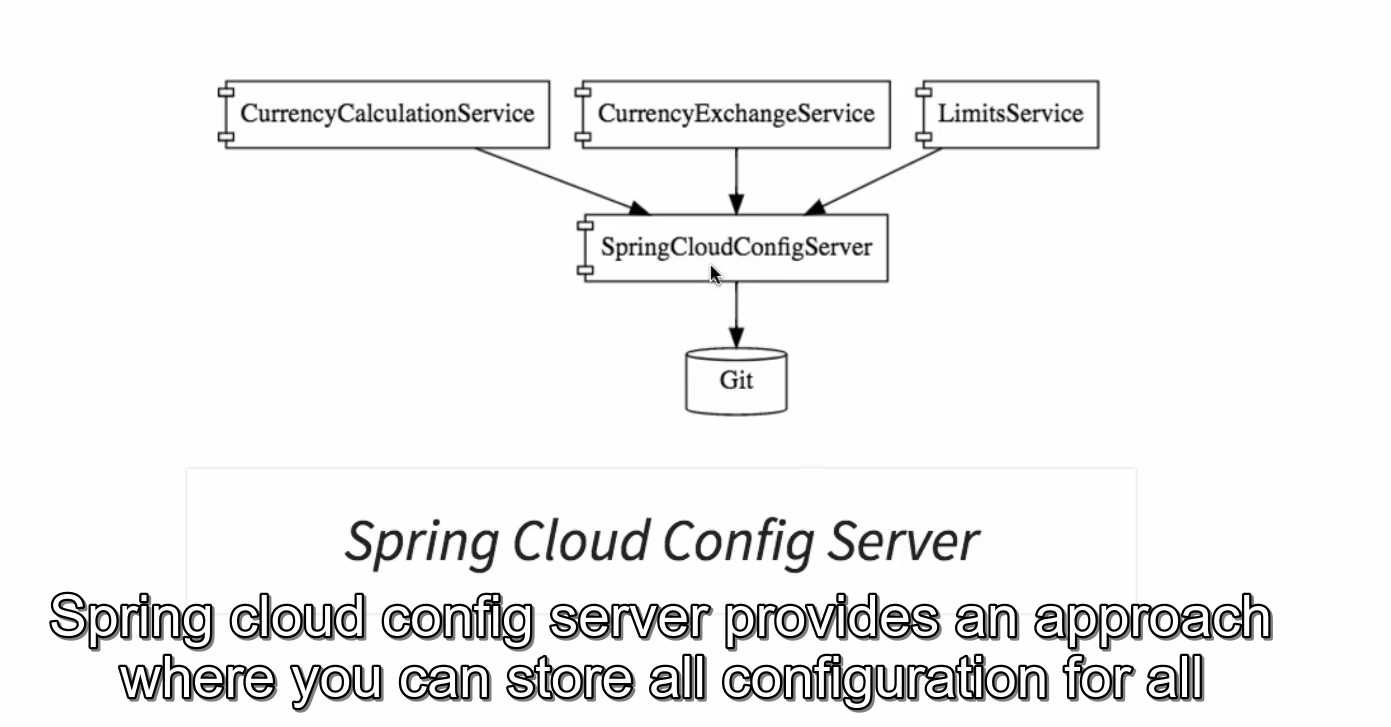
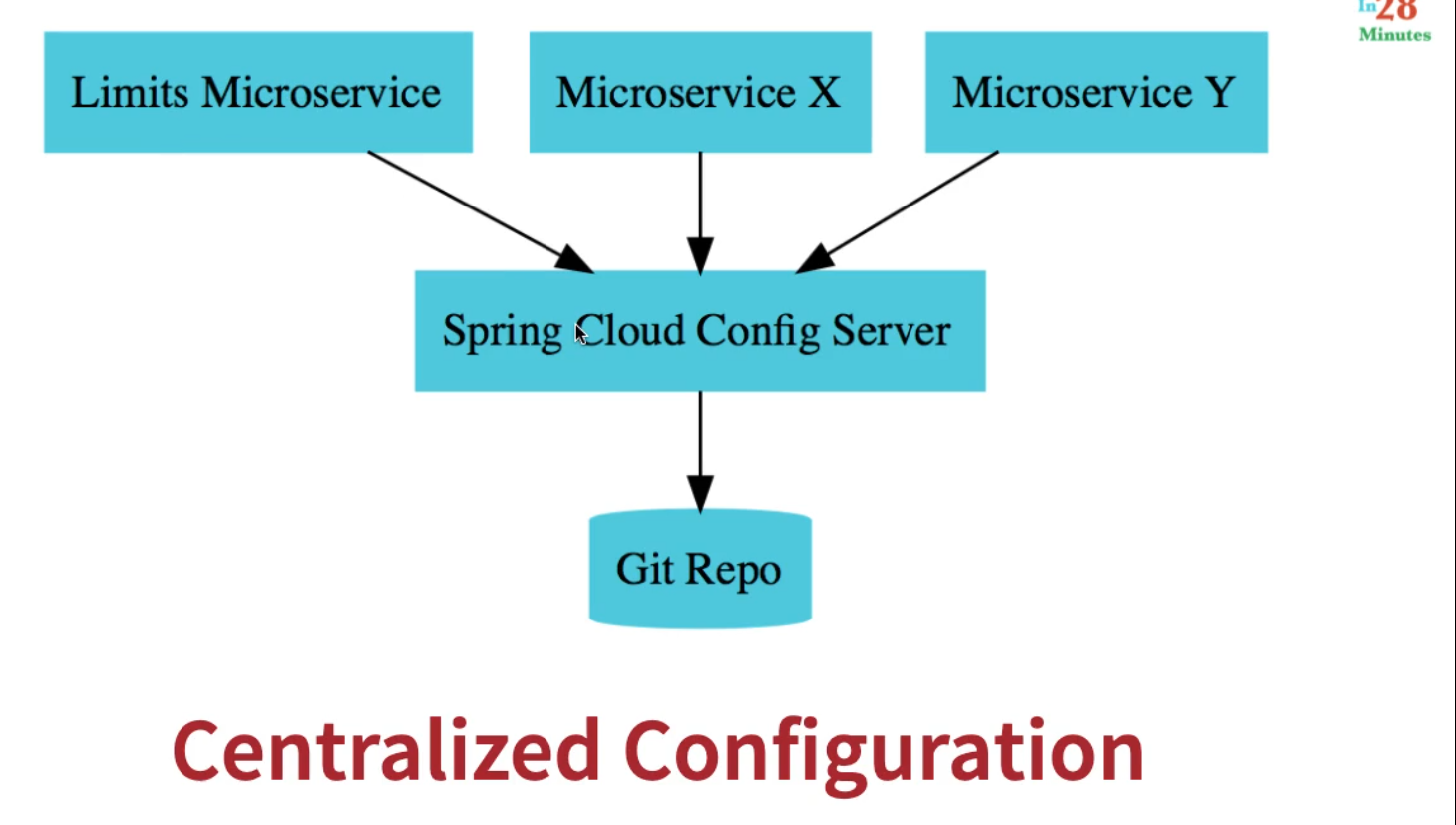
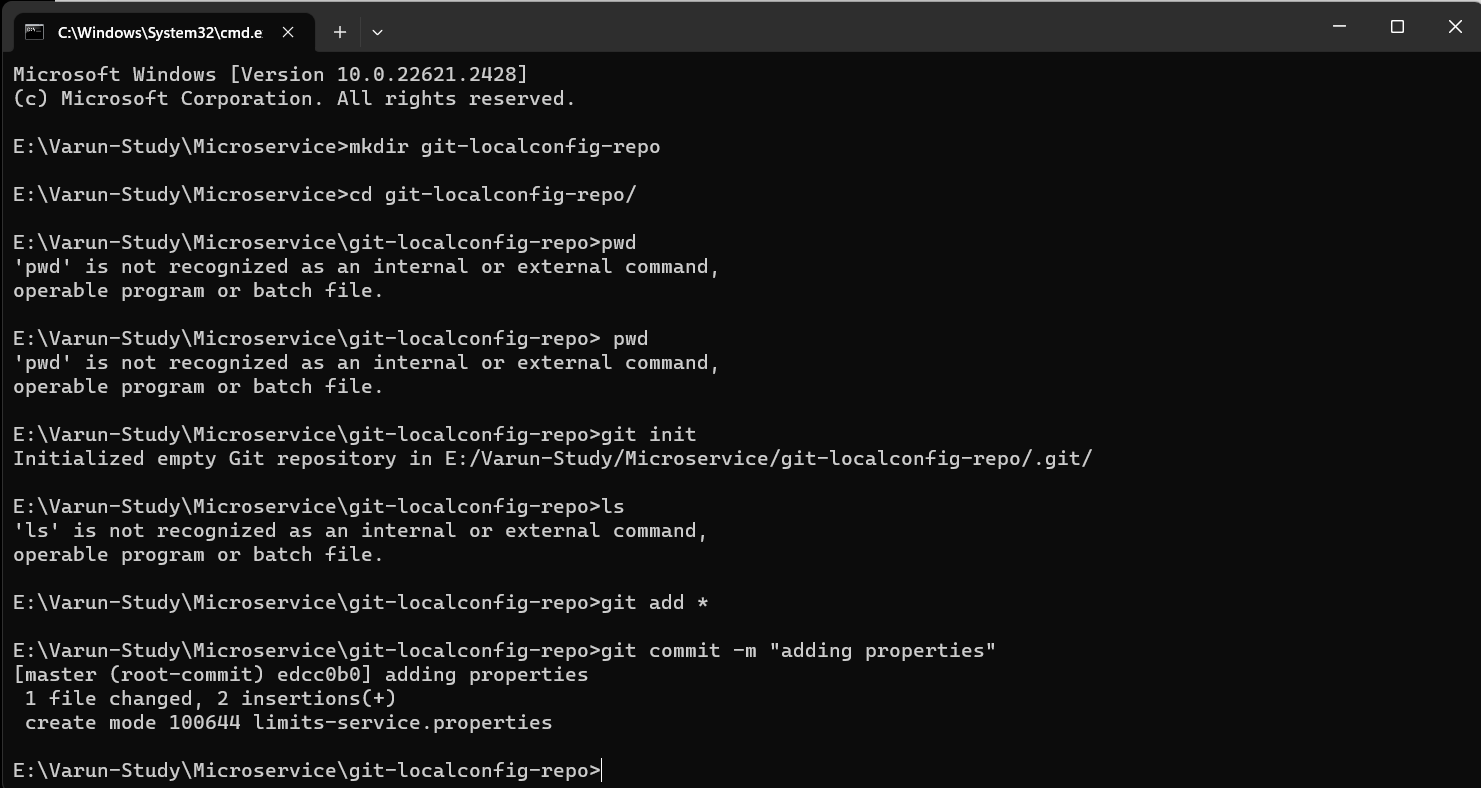
**Theory:**







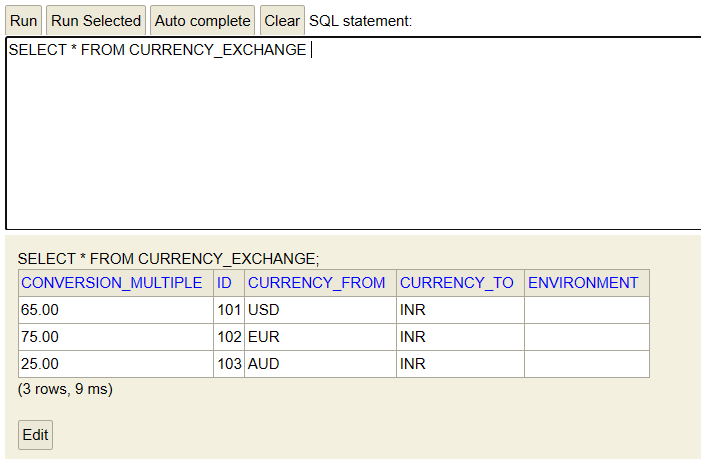




**Project:**

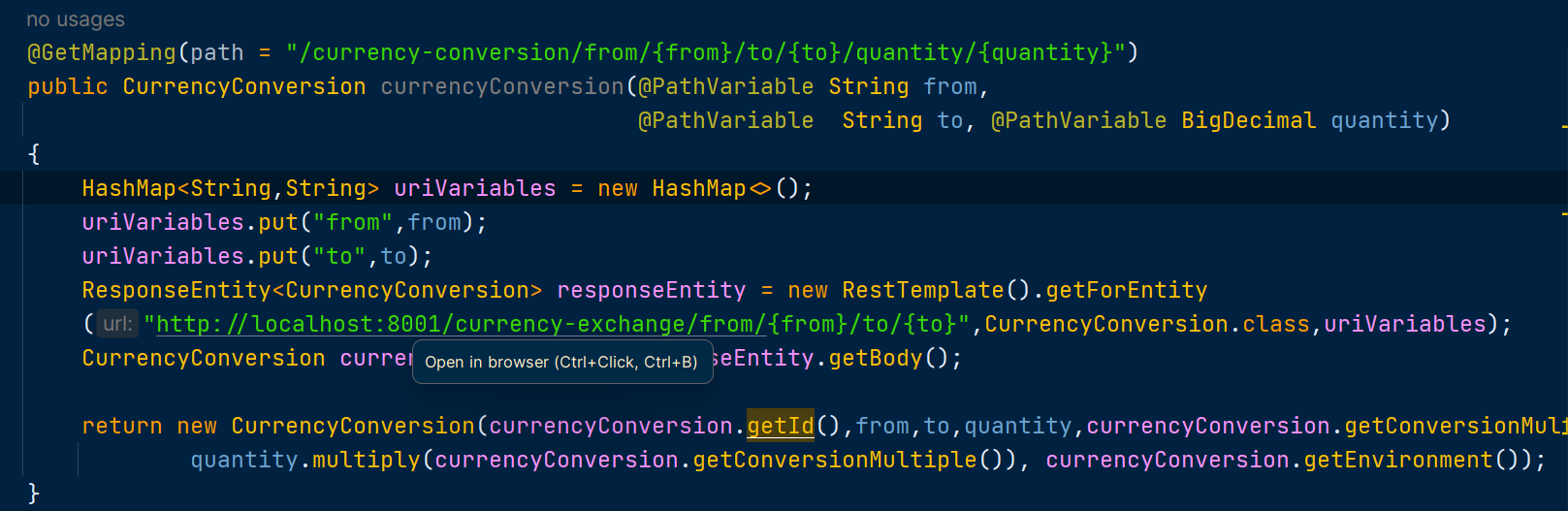
**Currency\_Exchange:**

It’s main task is to give the conversion multiple based on Country.

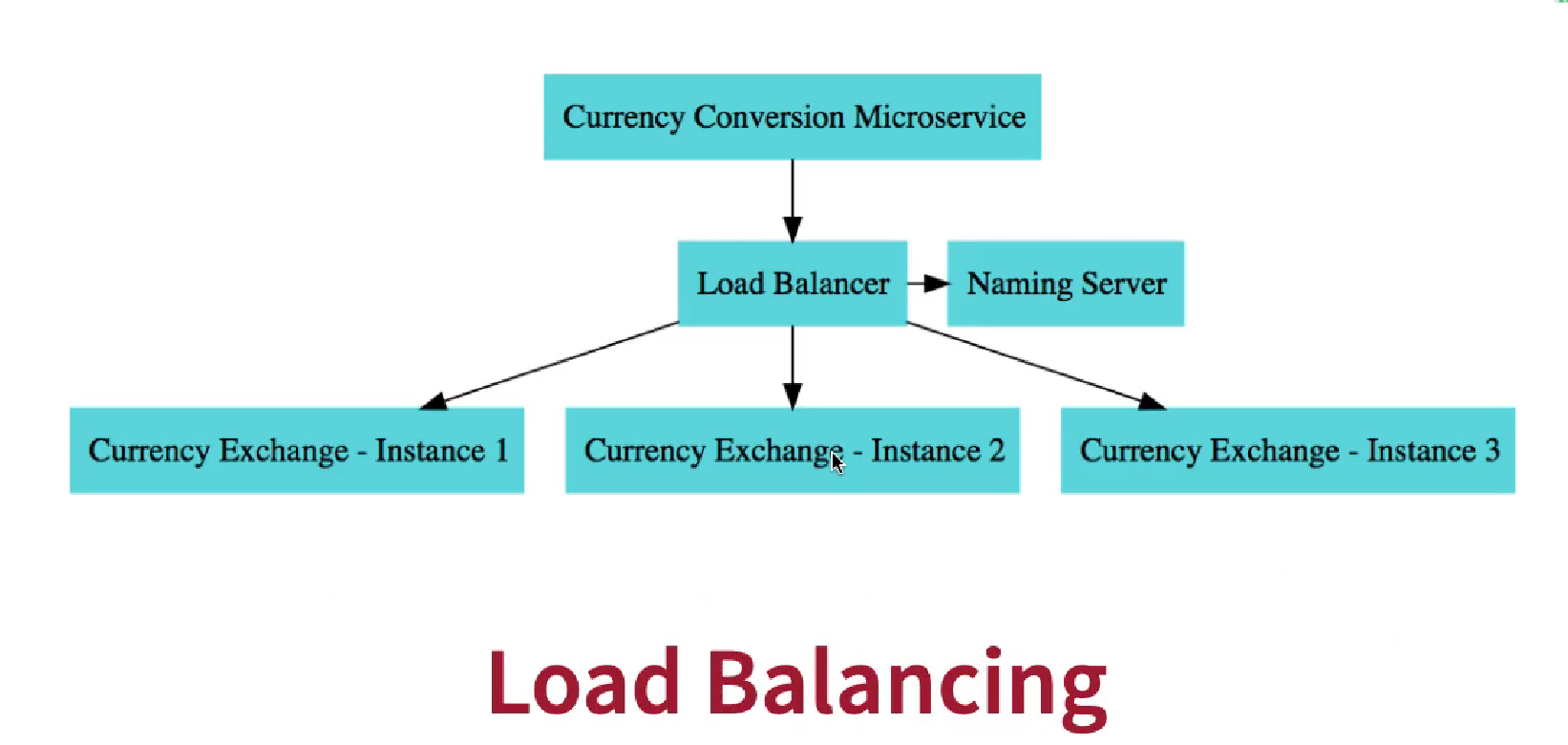
1. Created a Currency Exchange service on port 8001 which consist of a in memory Database like h2 database.
2. Entity consist of from,to, conversionMultiple and Environment.
3. Database consist of Currency\_from, Currency\_to and conversion multiple for all the currency.
4. 
5. Main work of currency exchange service is to give you conversion multiple based on the currency\_from and currency\_to parameters
6. @GetMapping("/currency-exchange/from/{from}/to/{to}")
7. 

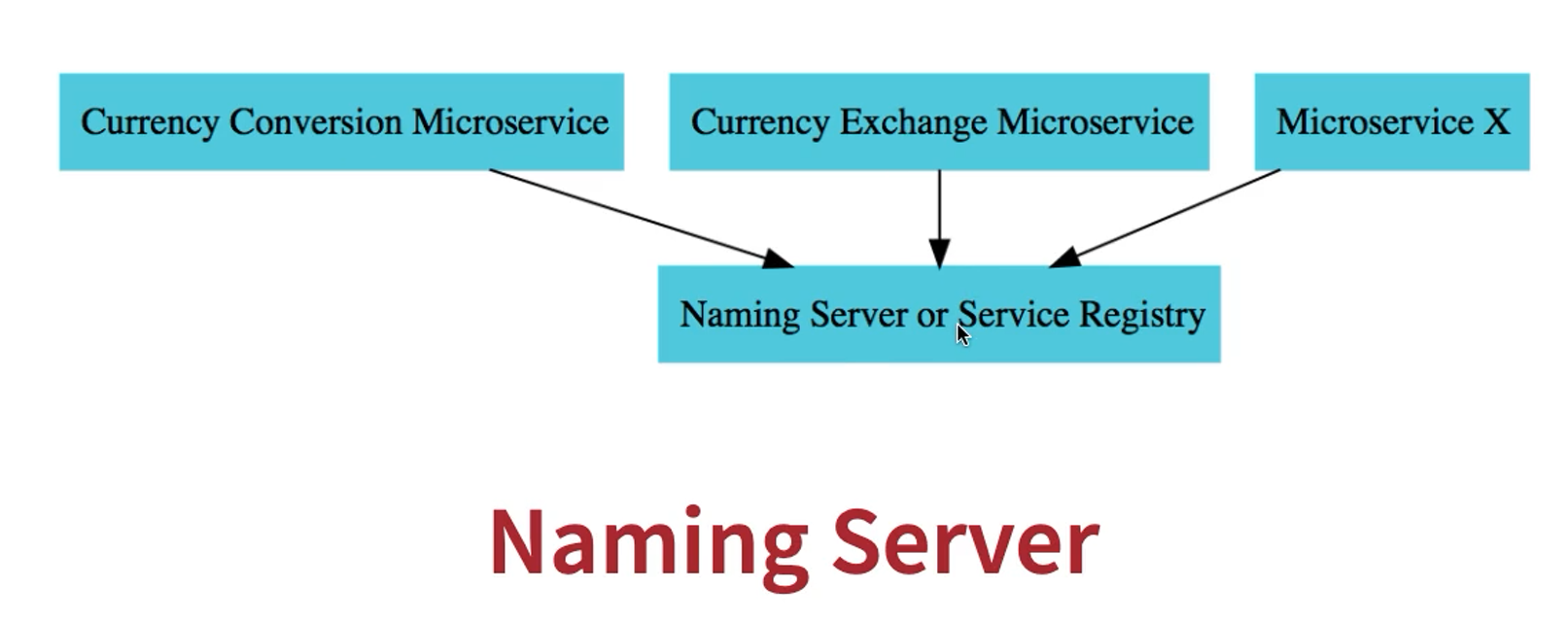
**Currency\_Conversion:**

This service takes the conversion multiple from Currency exchange service and do the currency conversion here!.

1. It uses port 8002.
2. Class consist of id, from,to, quantity, conversionMultiple, totalAmt, environment.
3. Service takes from, to and quantity as param from user.
   1. Calls currency\_exchange service to get conversionMultiple based on from and to parameters.
   2. Multiply the conversionMultiple with quantity and store it in totalAmt variable.
4. 

**Using Naming Server Eureka:**





Currency Conversion will ask Naming server about the instances or port of currency exchange service in-order to provide loan balancing for multiple instances.

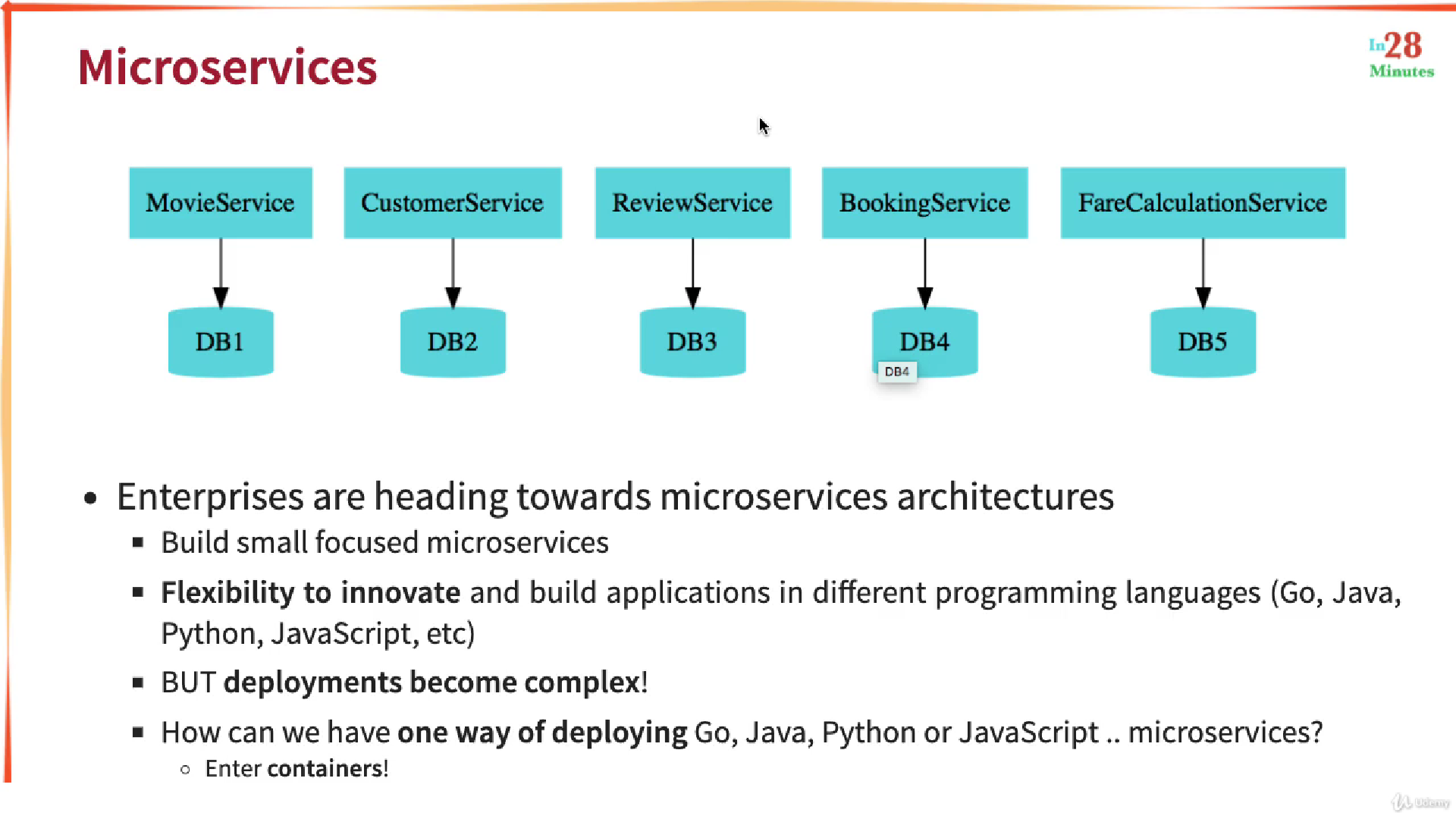
Annotation used: @EnableEurekaServer

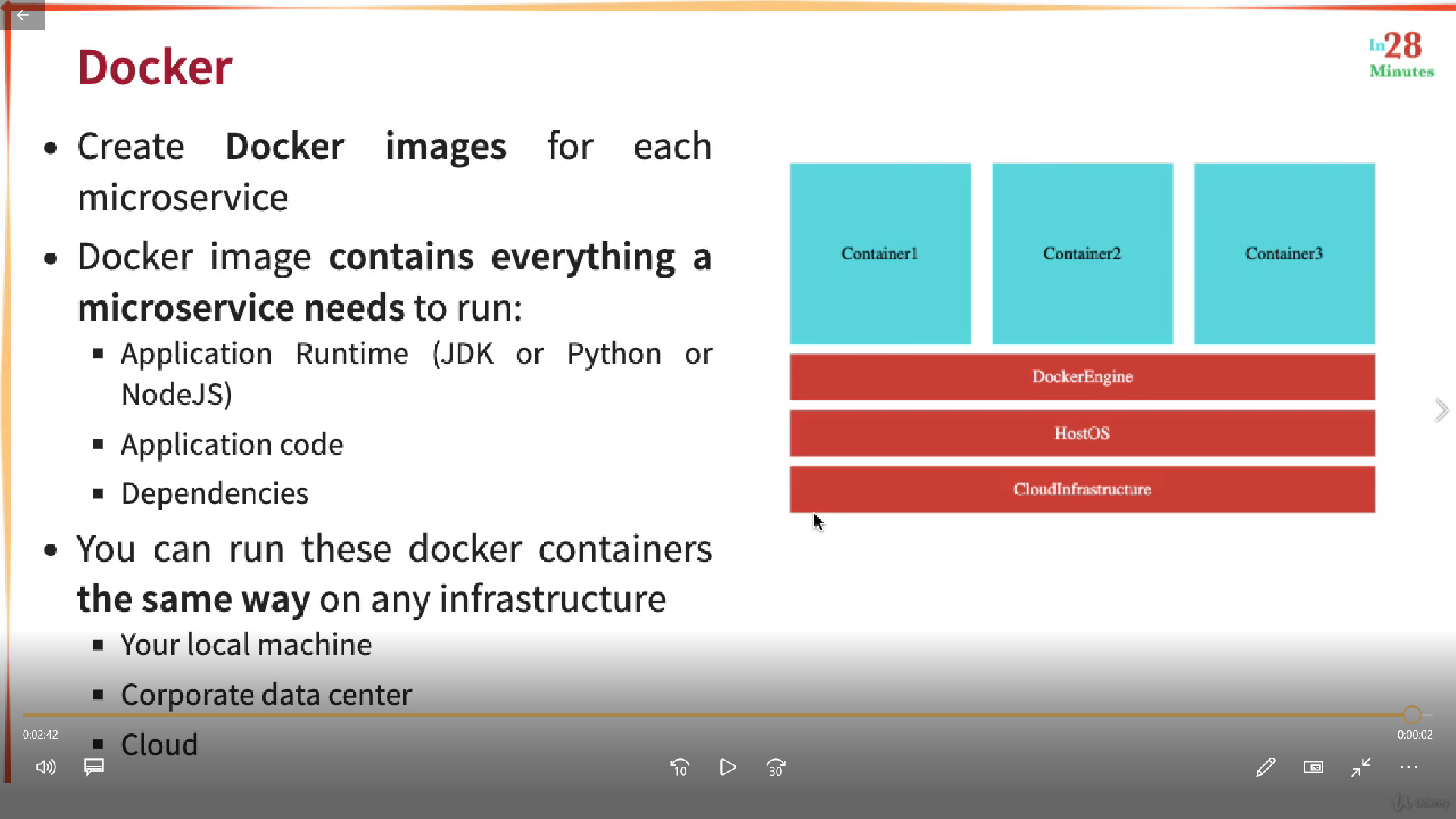
Port used: 8761

**API Gateway:**

It is used to provide security, monitoring and while routing API’s.Port used: 8765

**Docker:**





Why Docker?

Used to deploy applications very quickly



Image is downloaded from hub.docker.com (docker registry)

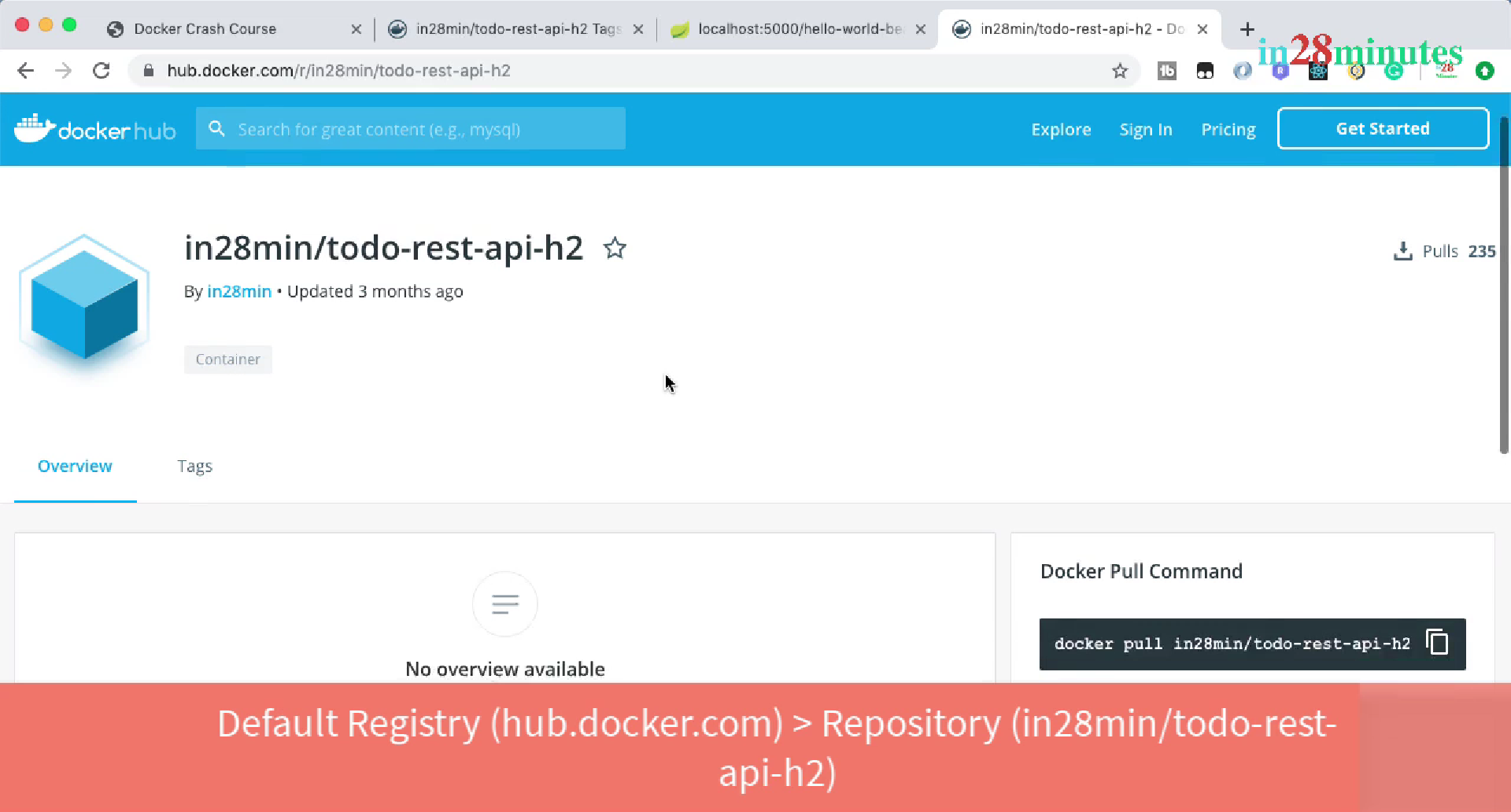


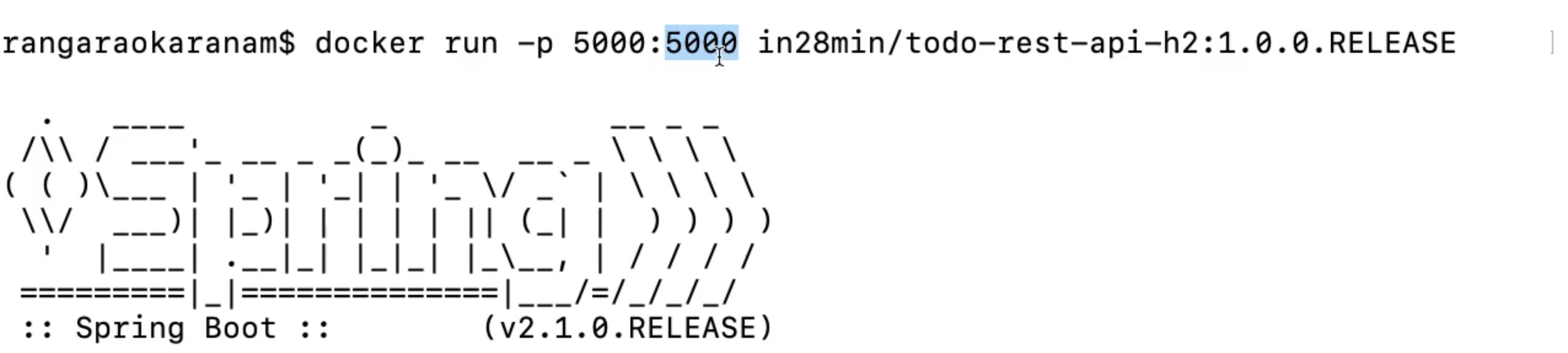
Image is the set of bytes and it contains all the things needed to run your application eg:

Java version, tomcat etc.

When image is running then it is called container.

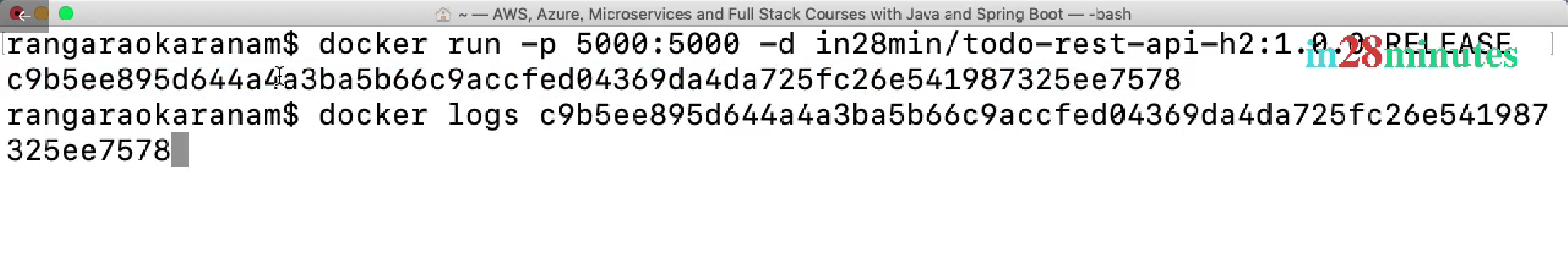
Ctrl+c = stop the running container

-p {HostPort} : {ContainerPort}



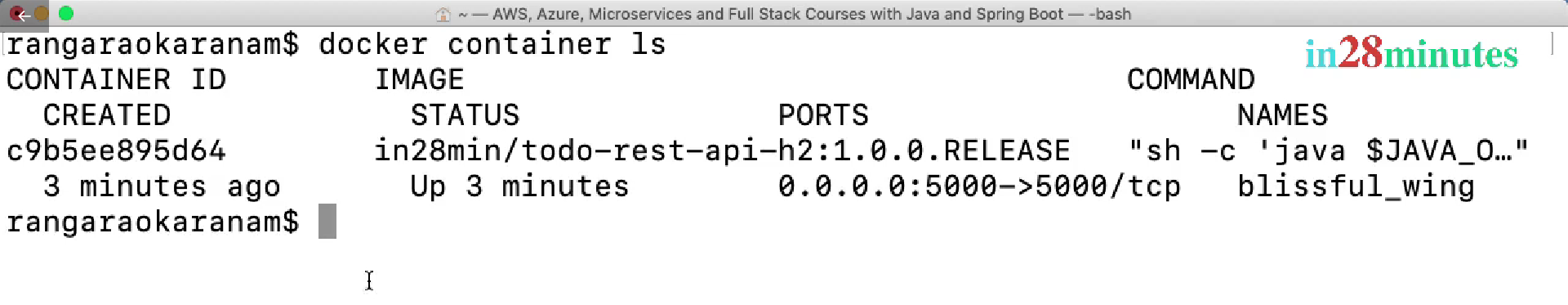
-d : detached mode (runs container in the background)

To view logs: docker logs (container image)

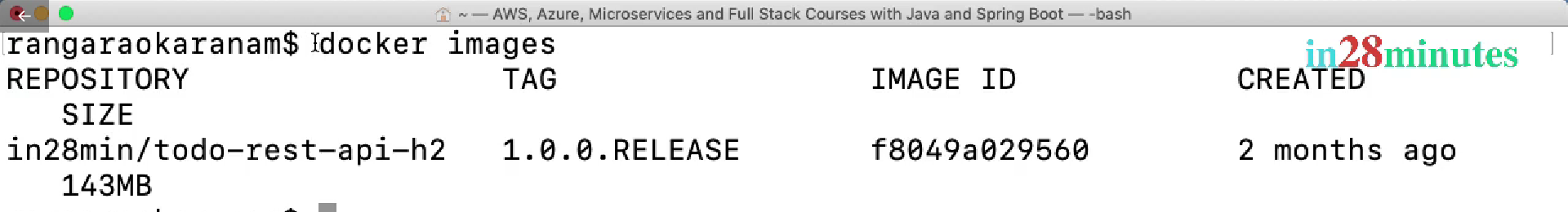


-f : Trailing logs ( will be able to see any new logs on the go)

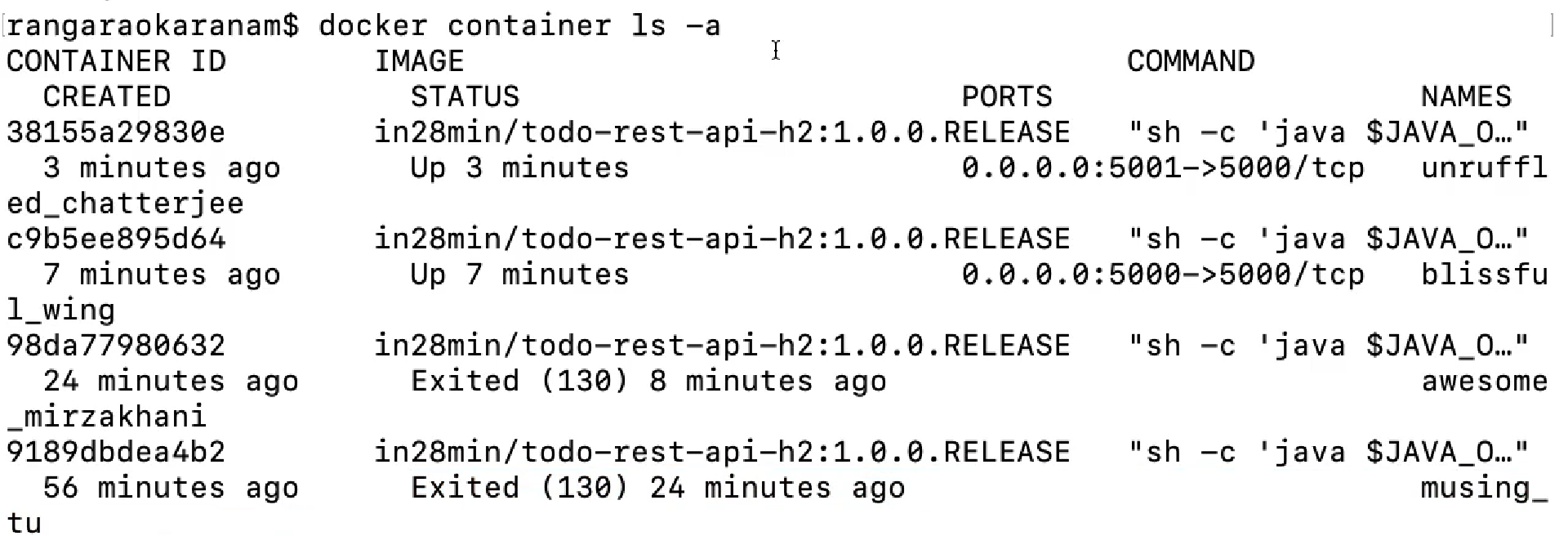
To see running containers:



Docker images:



To see all the container i.e running as well as stopped:

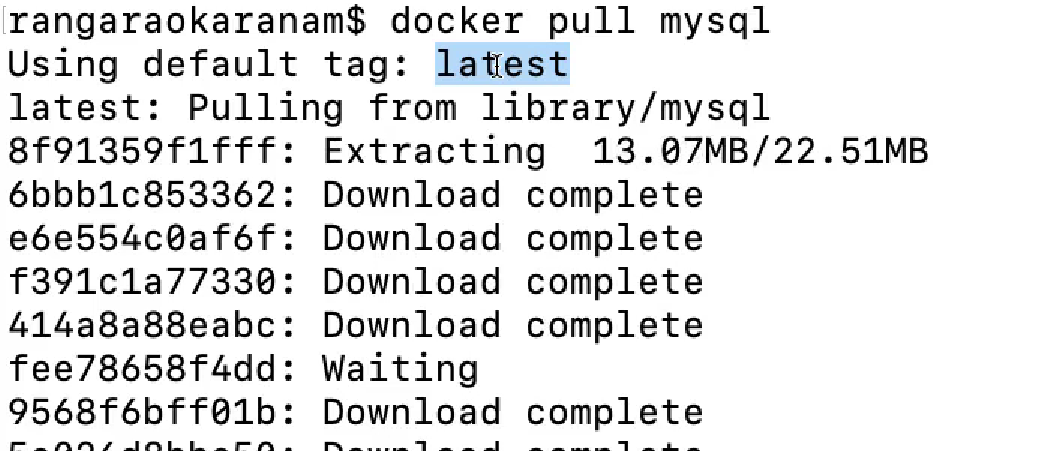


To stop a container: Docker container stop (container id)

Adding a tag:

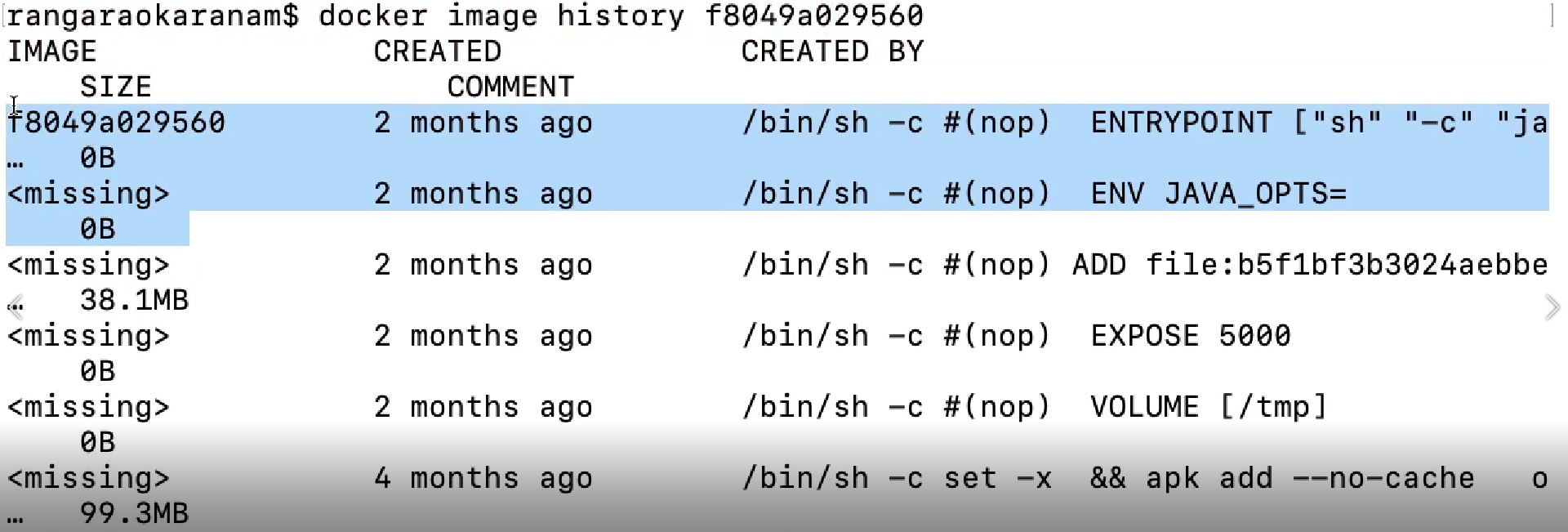


To download image from registry to local: Pull command:



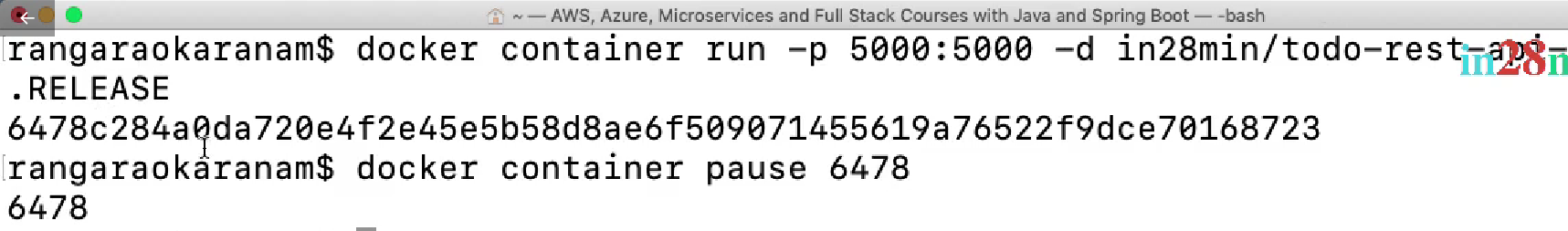
To search image : docker search mysql

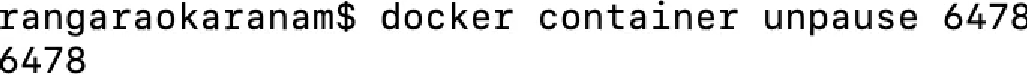
To see image components: docker image history (image id)



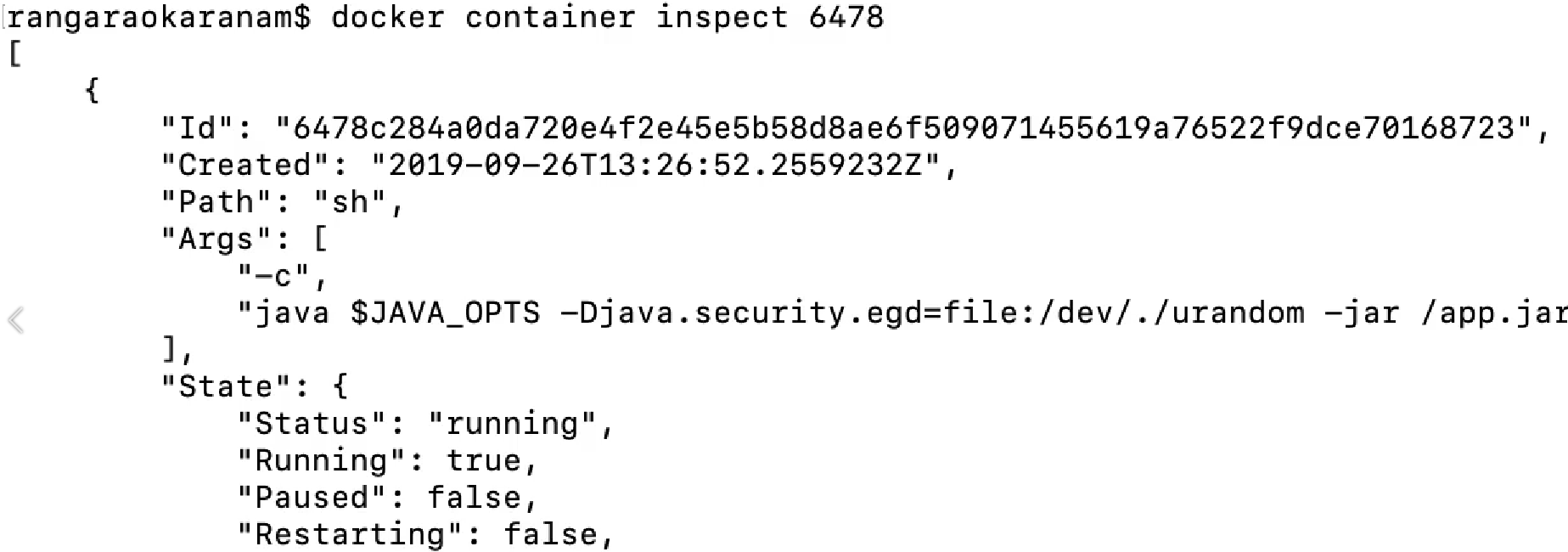
To remove image: Docker image remove (image id)

To pause/unpause a container:

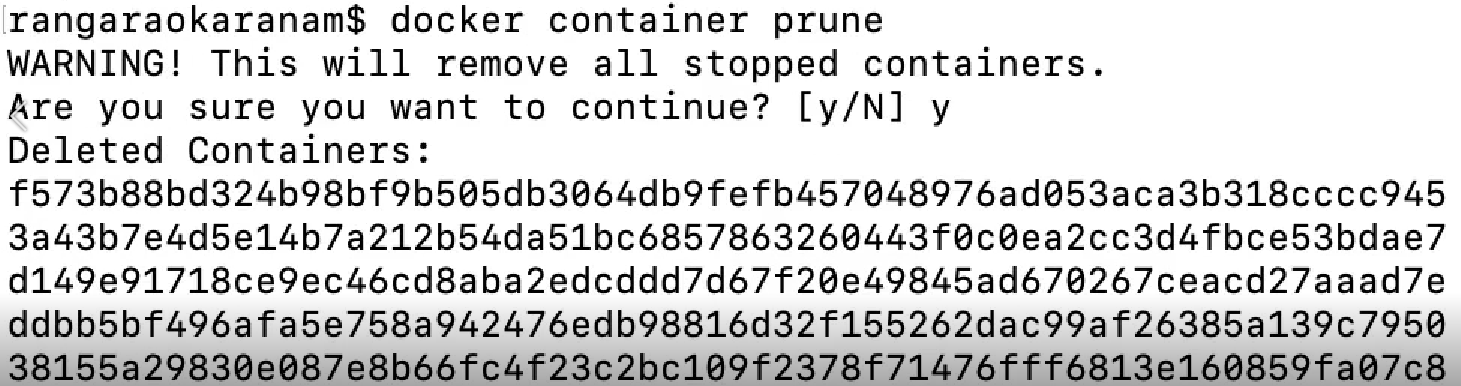




To inspect image:



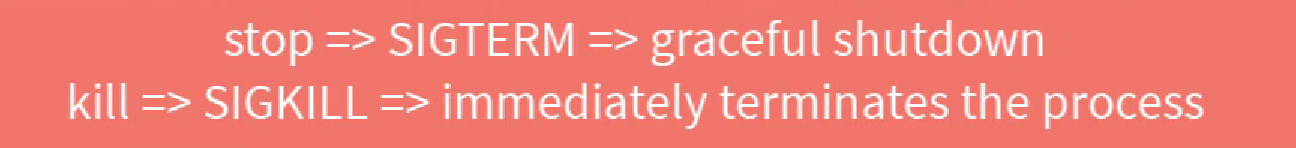
To remove all the stopped containers: prune



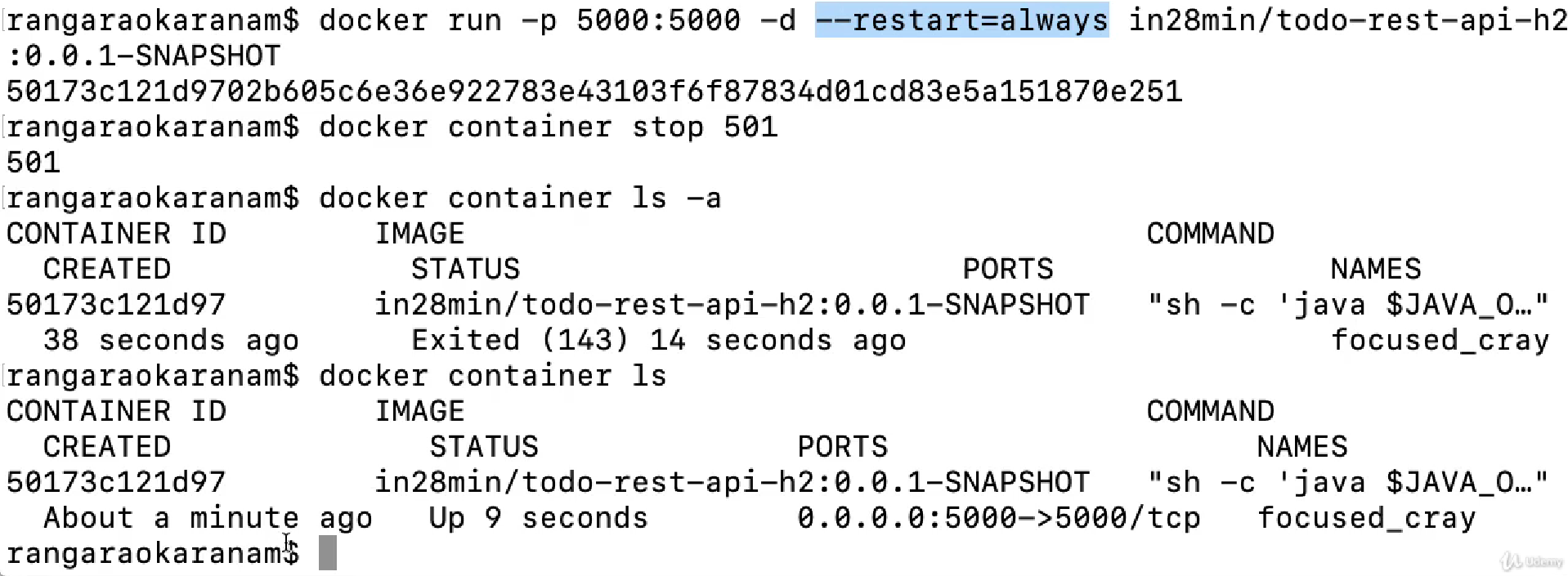
Stop vs Kill:

Container stop {id} || Container kill {id}

For graceful stop ( Closes all the process) Forced stop



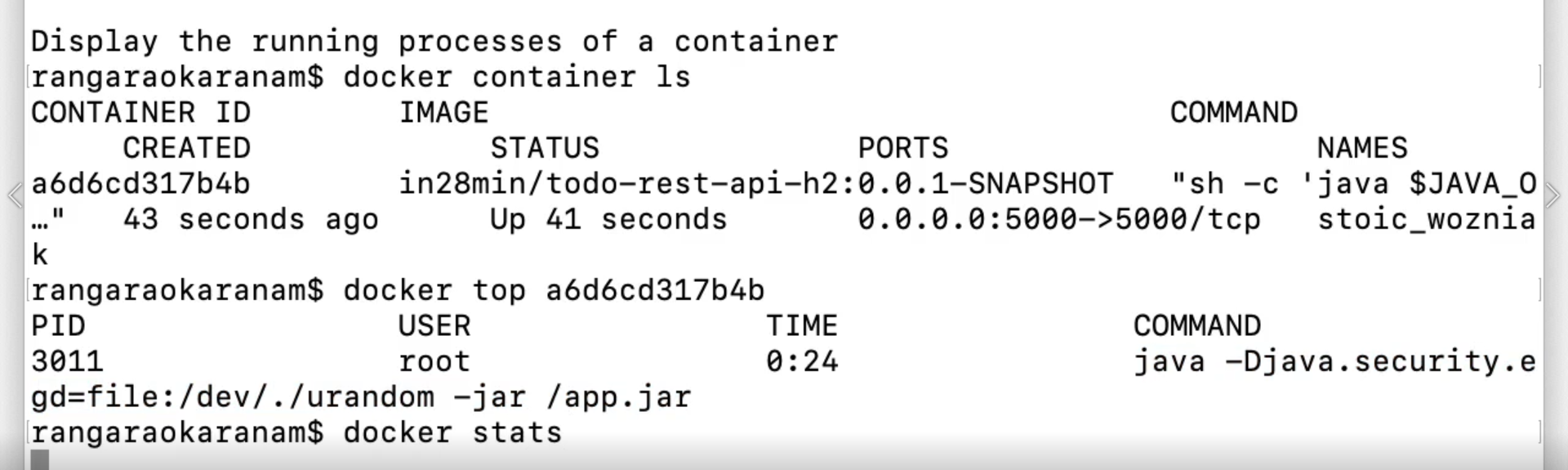
Container restart:

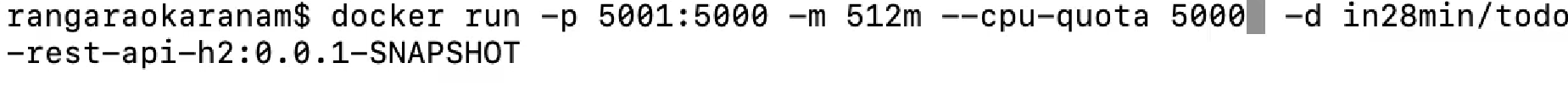


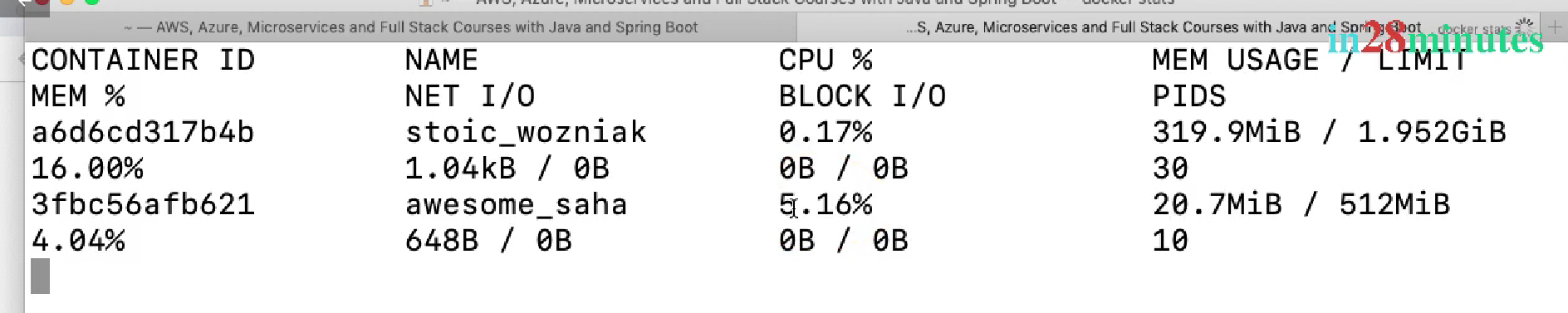
To see what’s happening with docker env. : Docker events

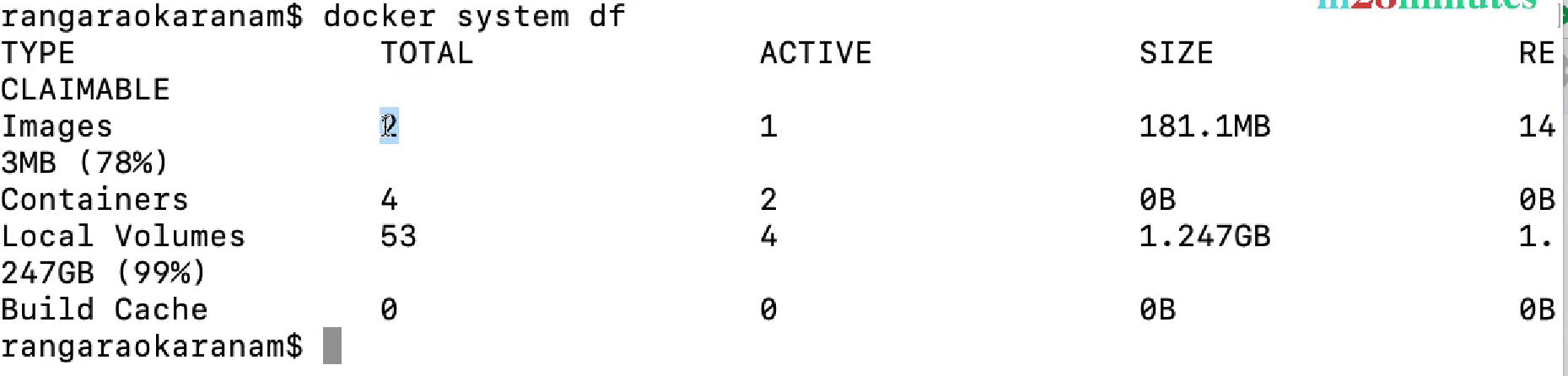
To see process running in docker container: docker top (container id)

Docker stats: CPU , memory etc.

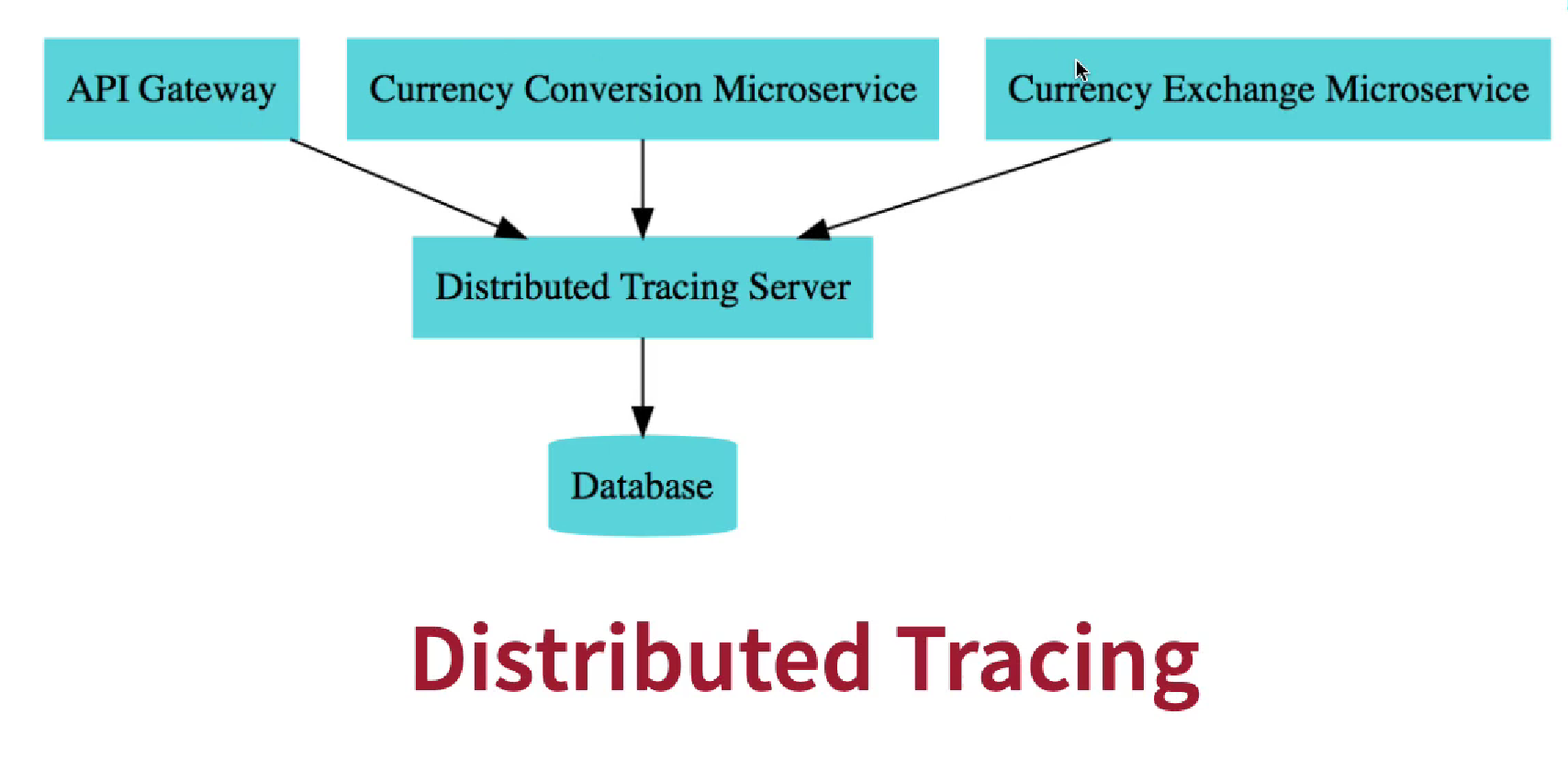








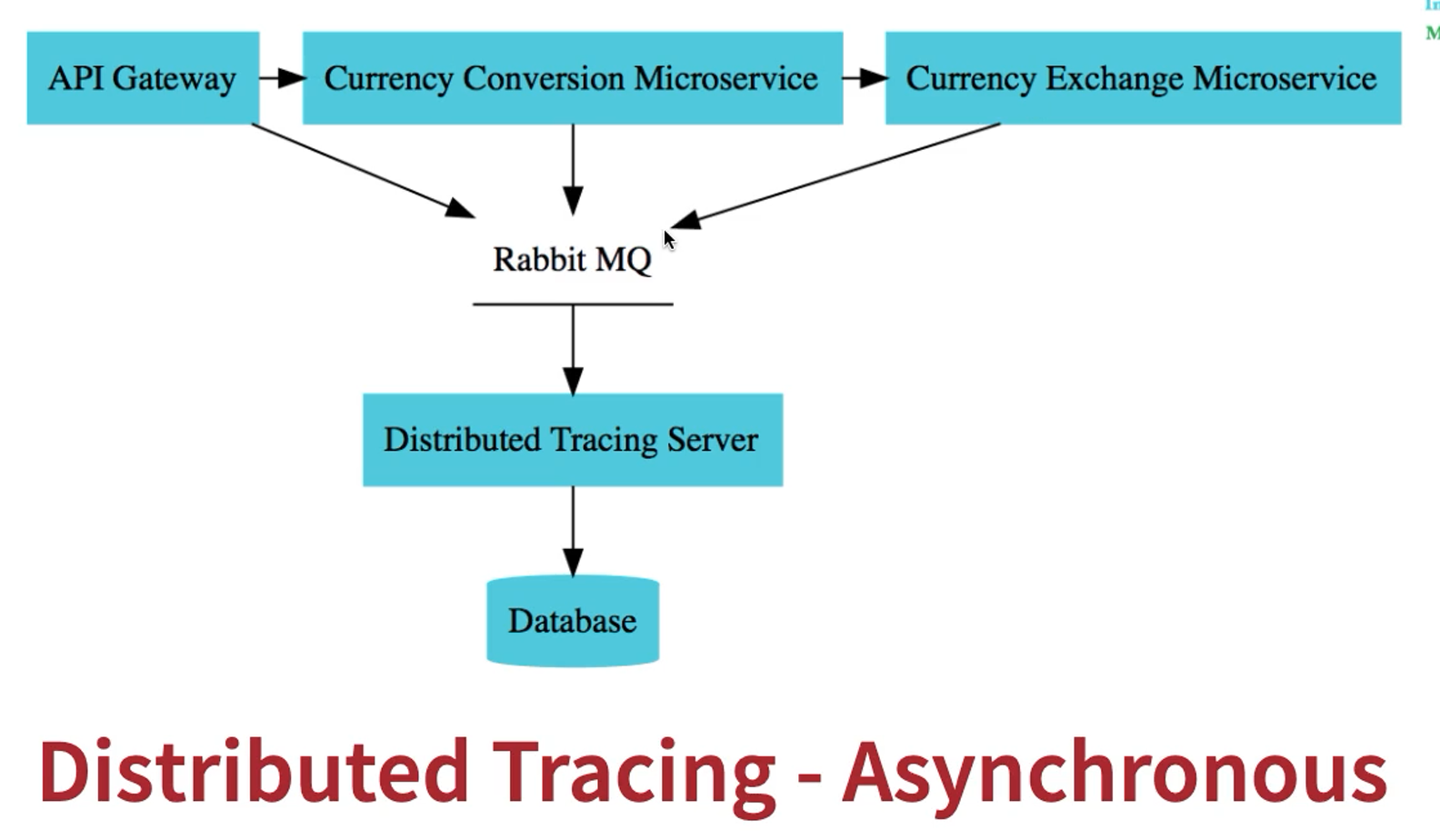
**Distributed Tracing for Microservices**:



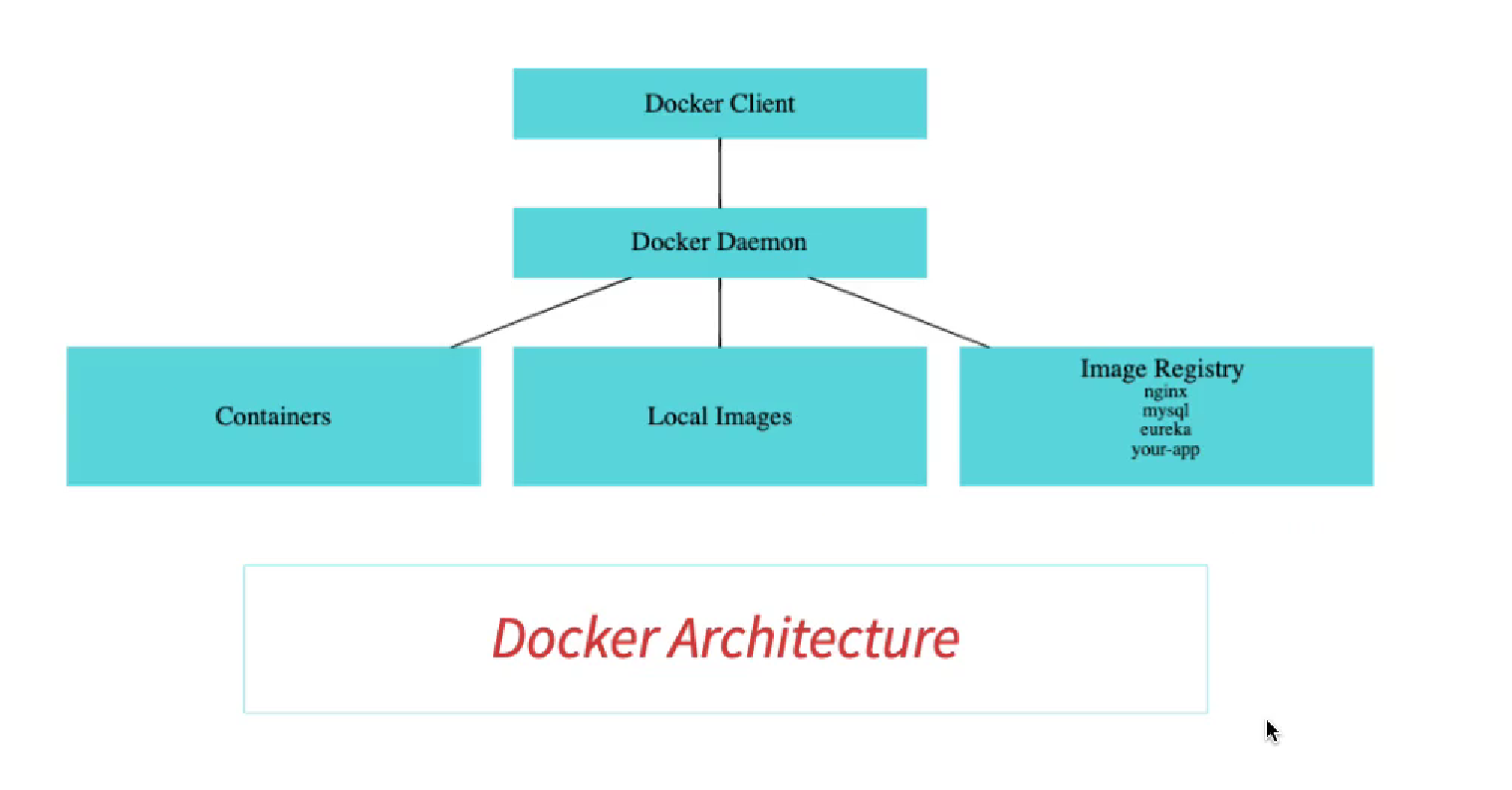
Distributed Tracing Server: Zipkin

Command: docker run -p 9411:9411 openzipkin/zipkin:2.23

Slauth: assigns unique id to each request



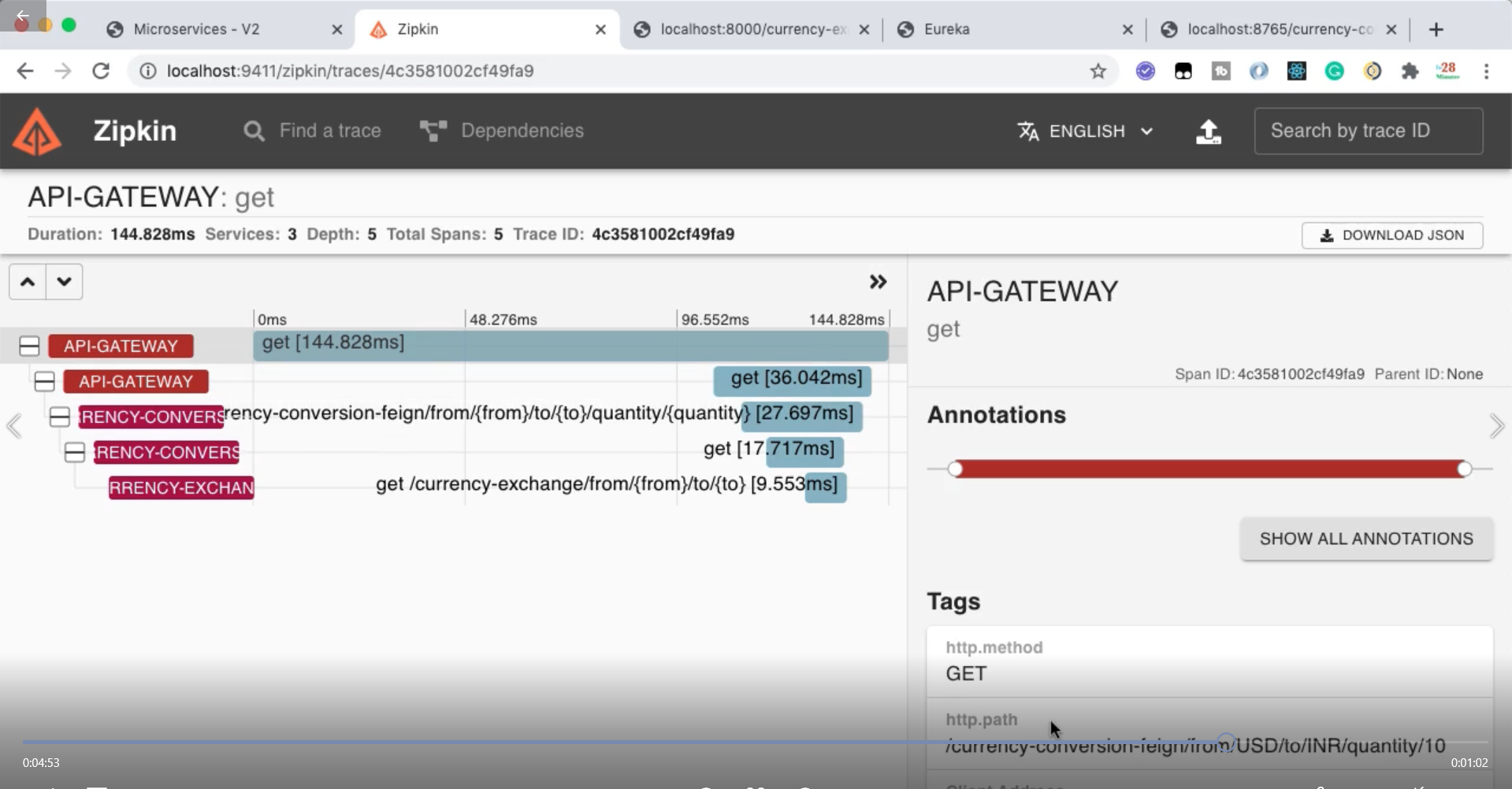
Docker Architecture: (Client-server)



Docker client: Place we are running docker commands

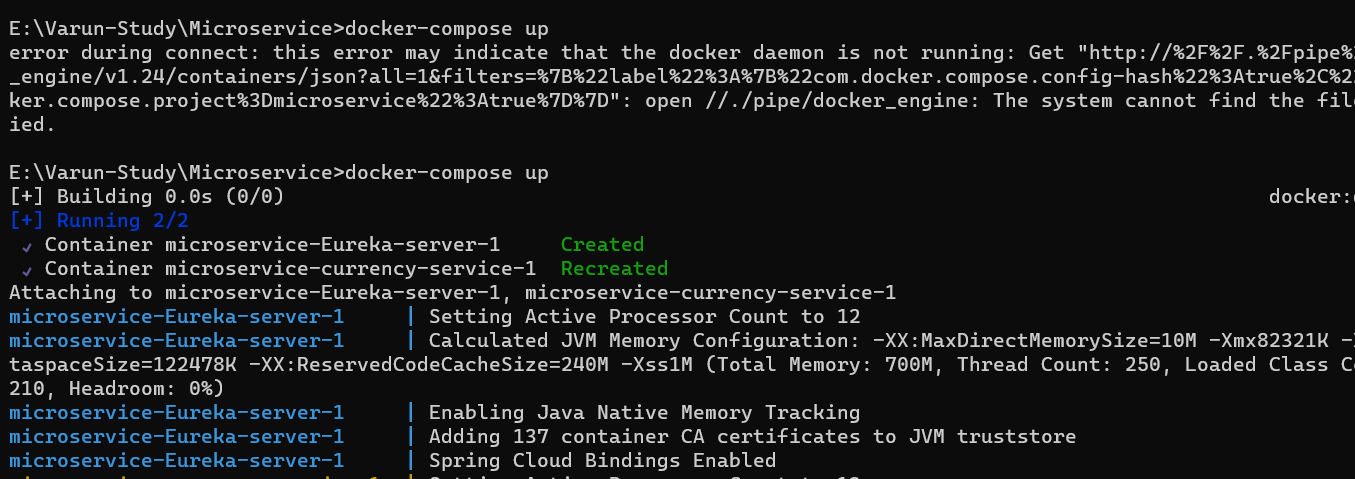
Docker Daemon ( Docker Engine): Manages the images.

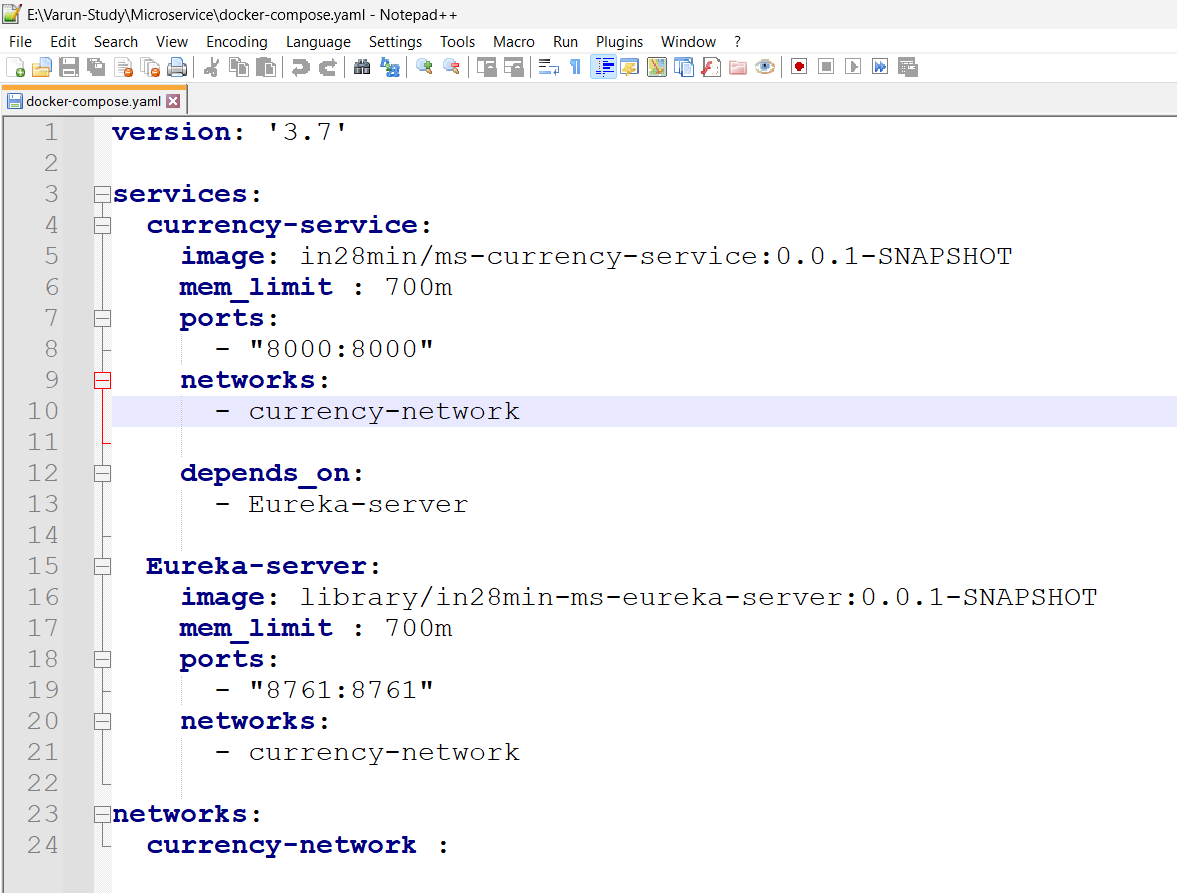
Images gets downloaded only first time



To run all the images together we can use docker compose.

It basically reads the YAML file which contains info of all the images.





To run a particular image we can use: **spring-boot:build-image**