**Microservices**

**Microservices challenges:**

1. **How to build microservices**
2. **How to do right sizing of the microservices**
3. **How to do deployment, portability and scalability using docker**
4. **How to do configuration management in microservices**

**3. Deploy Application using docker:**

**mvn spring-boot:run - to run maven command manually by terminal or CLI**

1. Any docker command will execute if docker server is running in local system.
2. When we execute any docker command from CLI instructions goes to docker server on local server to execute that command like to build docker image or to run container.

**Command:   
Build docker image:**

**accounts> docker build . -t shubhakshi22/accounts:s4**

**Run docker container using image:**   
 **docker run -d -p 8080:8080 image name: tag name**

**Notes**: -d used for detached mode, -p for port number, many number of containers can run at the same time by using same image by exposing diff port number. Otherwise, application will give port number already in use error.

**Start any already existing container:**

**Docker start container-id**

**Stop running container:**

**Docker stop container-id**

**Check running container:   
 Docker ps**

**To generate docker image using packeto Build packs:   
 mvn spring-boot:build-image**

**Notes: size of docker image using build pack becomes less less because it follows all best standards, how to cache multiple layers and how to compress multiple components inside docker image.**

1. **Build packs prioritize caching, comprassing, consistency, security and performance over Dockerfile configuration to create docker image.**
2. **Buildpack works for NodeJS, .NET, Python, Java, GO**

**Genrate docker image using Google Jib:**

Jib used to create docker images only for java projects.

**mvn compile jib:dockerBuild**

**Notes: jib doesn’t require docker installed in local system. If docker is not there in local system then jib will create the docker image of application and push it to repository. For this case we need to add repository configuration explicitly in project’s pom.xml**

**Command to push docker images to docker hub repo:**

**docker image push docker.io/shubhakshi22/accounts:s4**

**Note: CLI will fetch the docker hub account details from docker desktop and use it to push the image to docker hub.**

**Command to check docker compose installed and version:**

**docker compose version**

**Command to run docker compose file (to run multiple containers from docker compose file):**

**Docker compose up**

**Command to stop and remove running docker container:**

**Docker compose down**

**Command to stop but not remove running docker container:**

**Docker compose stop**

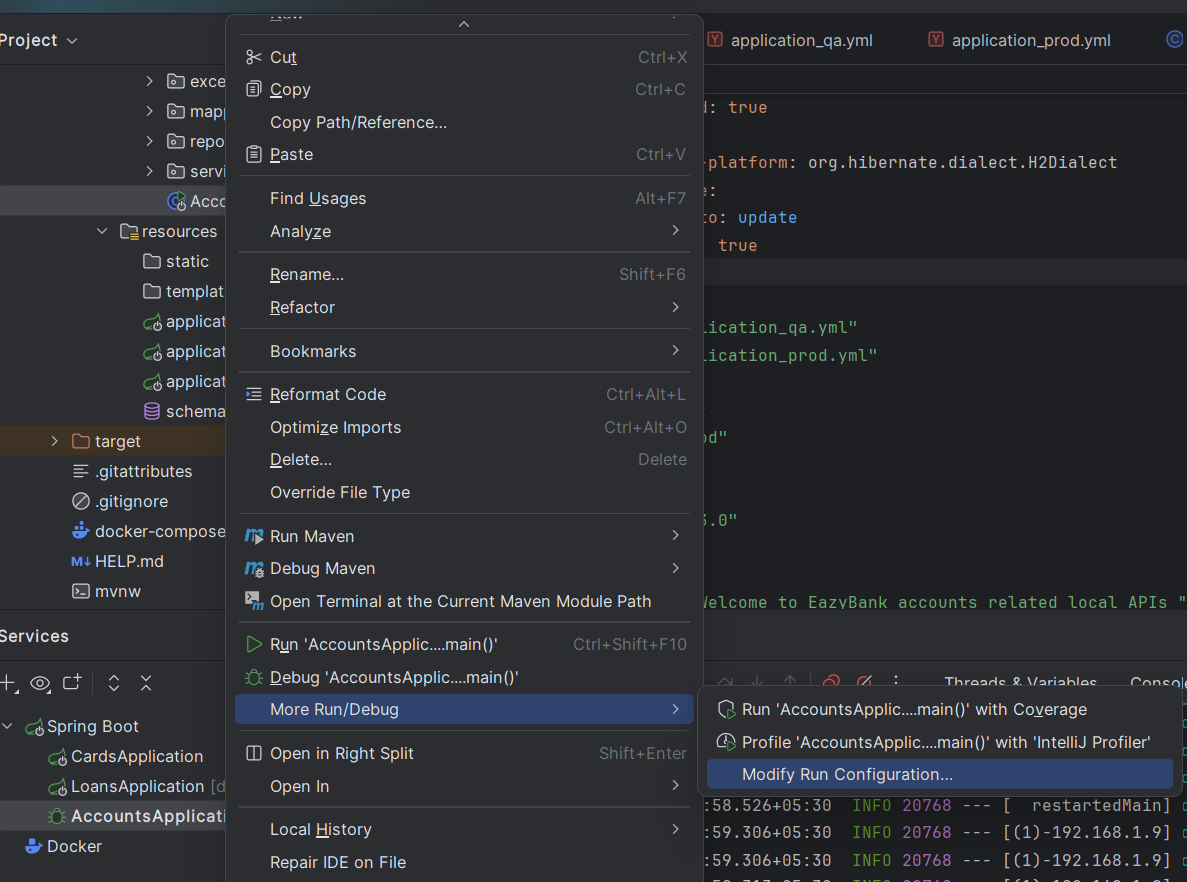
**Command to start existing stopped docker container:**

**Docker compose start**

**How to externalize configurations using CLI, JVM system properties and using ENV variables:**

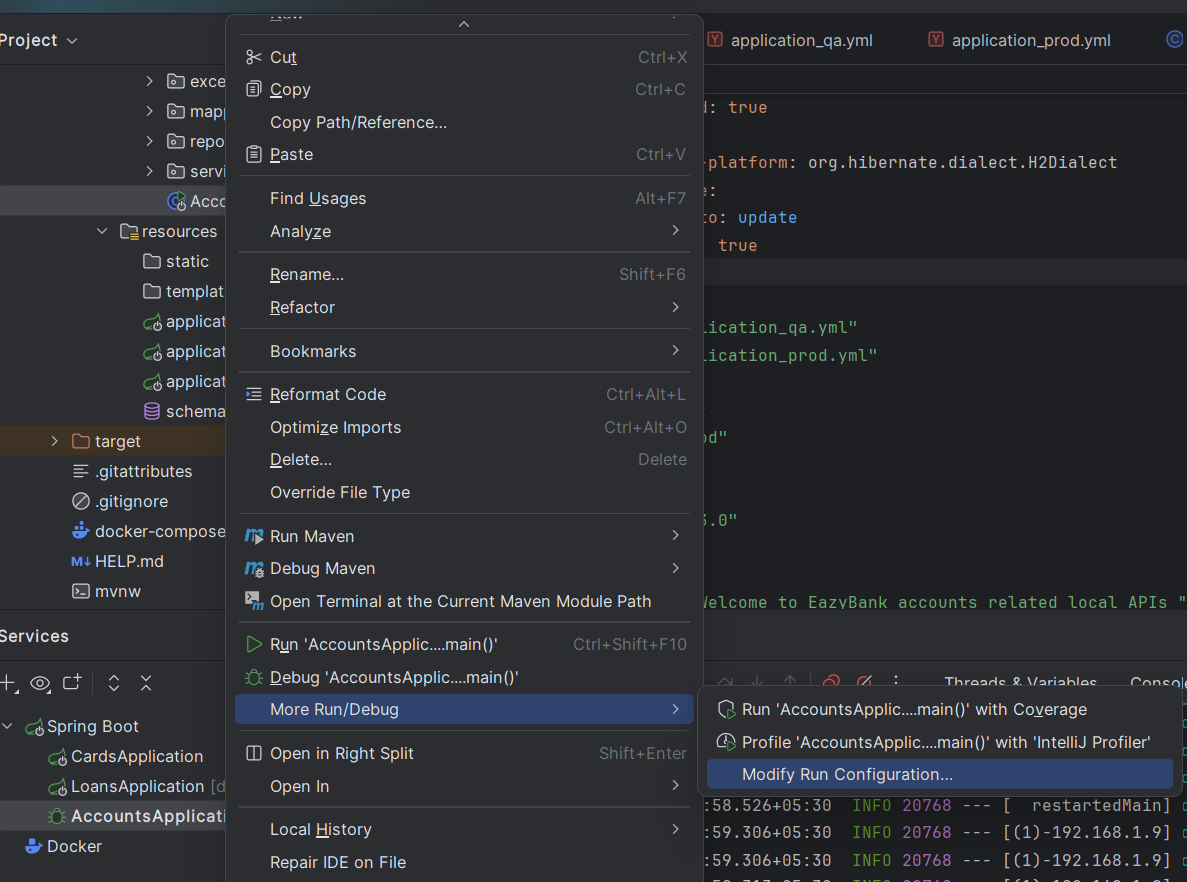
1. **Run application by using command line arguments:**

**We need to pass ‘program arguments’ in below mentioned path:**

**  
 --spring.profiles.active=prod --build.version=1.1**

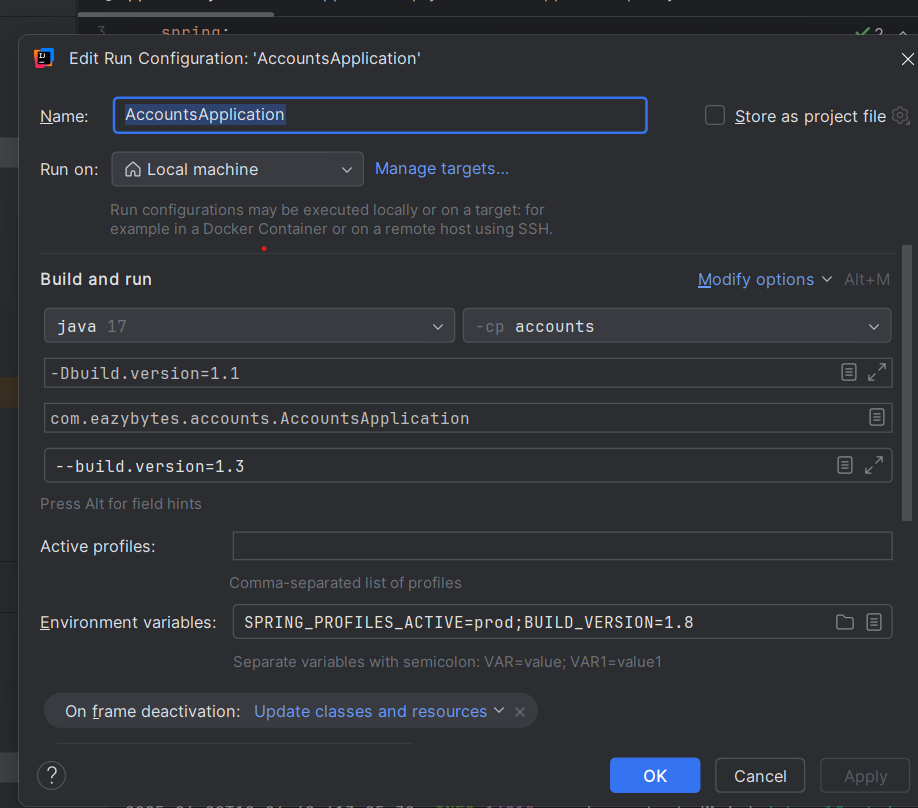
1. **Run application using JVM system properties:**

**Need to pass below mentioned command to ‘Add VM options’   
  
-Dspring.profiles.active=prod -Dbuild.version=1.3**

****

1. **Run application using ENV variables properties:**

**SPRING\_PROFILES\_ACTIVE=prod;BUILD\_VERSION=1.8**



**Precedence sequence of externalize options:**   
**1. CLI**

**2. JVM properties**

**3. ENV variable properties**