Backend Program: Course 3: Structure

STACK ROUTE

Program

Build job-ready skills

Enterprise Application Development By Using Spring Framework

Courses

Build competencies

CRS1 : Spring Framework Foundation

CRS 2 : Creating Restful Services Using SpringBoot

CRS 3 : Creating and Managing Microservices

CRS 4 : Establishing Communication Among Microservices

Learning Sprints

Perform specific tasks

Work with Virtual Machine

Enable Continuous Integration within a Backend Application to Support a Collaborative Development Environment

Containerize RESTful services and Database by Using Docker

Containerize RESTful Services and Database by using Docker Compose

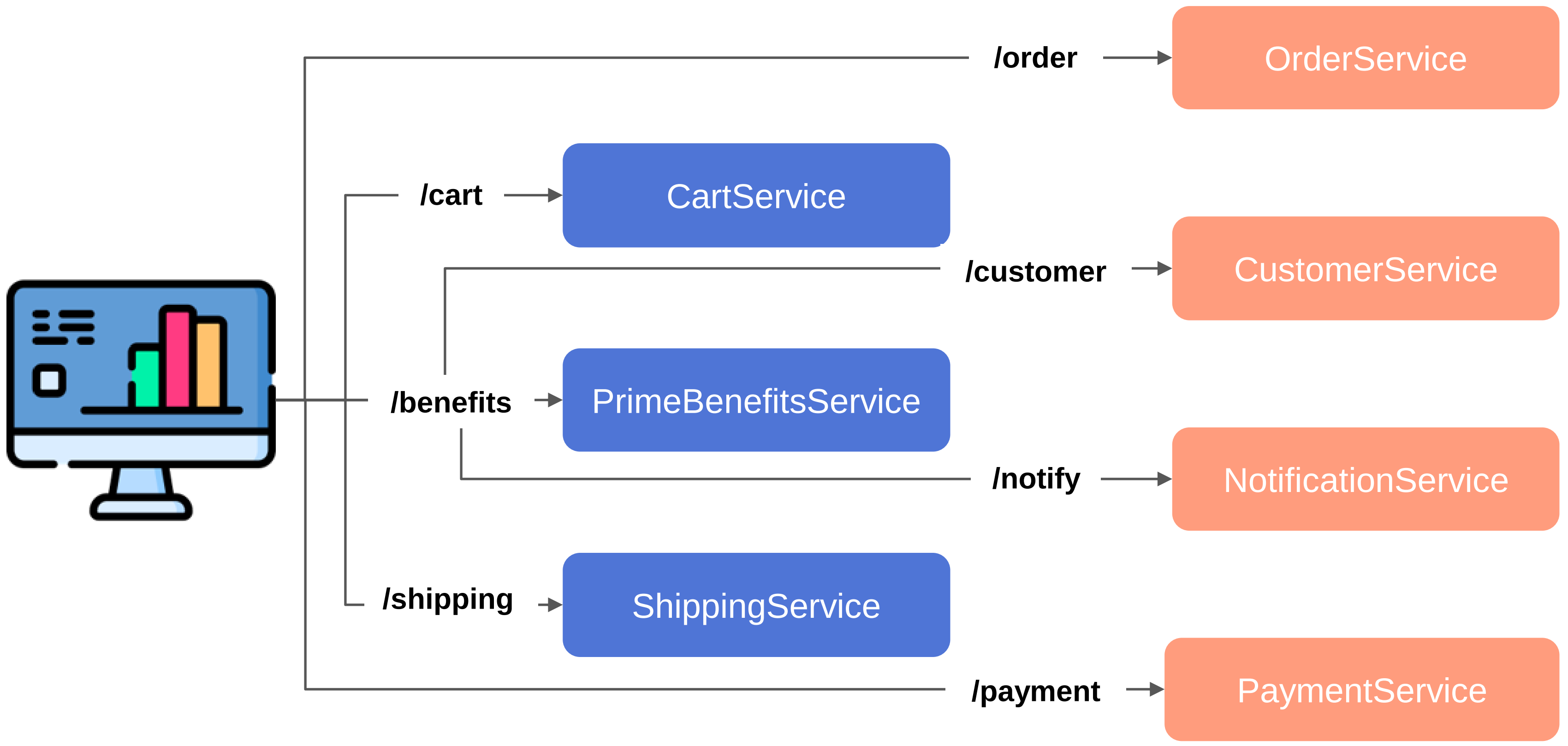
Create Microservices by Using Spring Boot

Create a Single-Entry Point to Route the Request Coming for Different Microservices Using Spring Cloud

Register Microservices on a Netflix Eureka Discovery Server

Handle the Failure of Microservices by Implementing the Circuit Breaker Pattern Using Hystrix

Application Workflow – Multiple Services



Time to Think

- When you deploy the application shown in the previous slide how many containers will you create?
- Will you create 7 containers for the application and 7 containers for the database?
- Do you think this process is simple?
- How many docker commands will you write for this?
- Do the commands have to be repeated? Will the commands be repetitive?
- Do you think the port mapping is simple or complicated for each service and database?

Containerize RESTful Services and Database by Using Docker Compose

Learning Objectives

- Describe Docker Compose
- Dockerize the Backend application with Docker Compose

Docker Compose is a “tool for used defining and running your multi-container Docker applications”

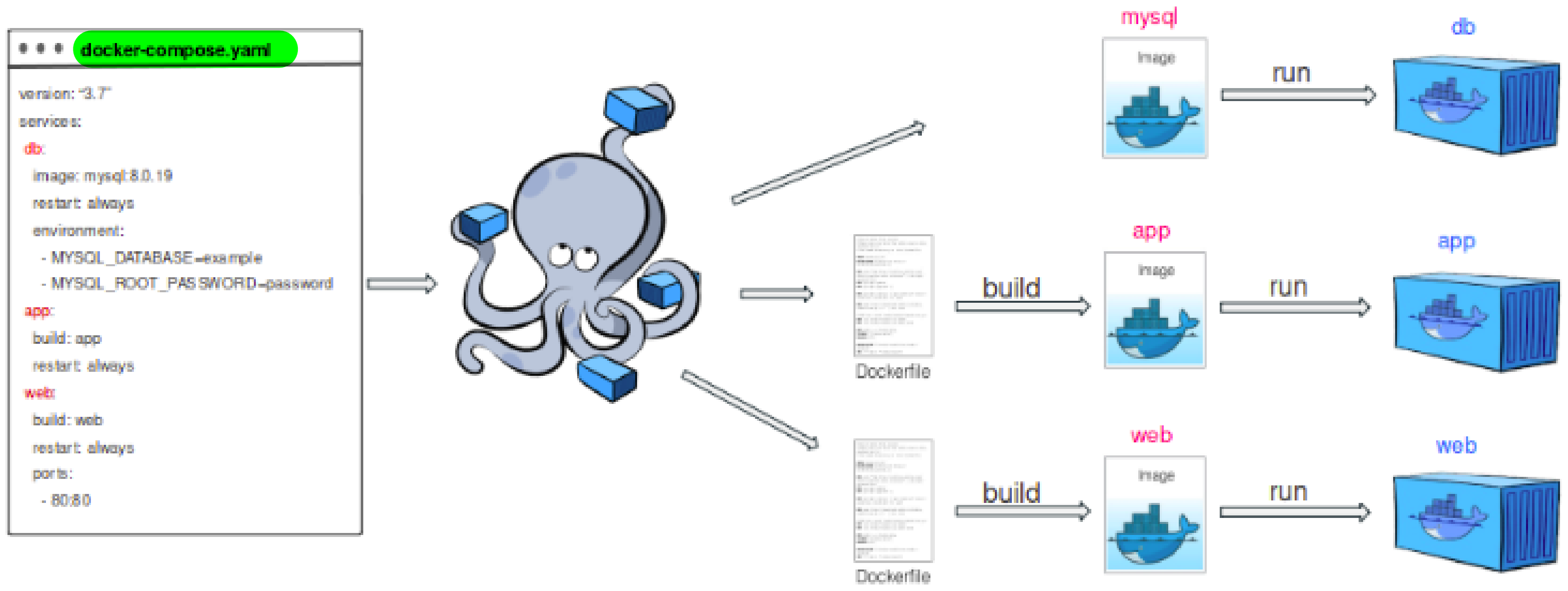
Docker Compose

- Docker Compose is used for running multiple containers as a single service.
- Each of the containers here run in isolation but they can interact with each other when required.
- Docker Compose files are very easy to write in a scripting language called YAML.
- Another great thing about Docker Compose is that users can activate all the services (containers) using a single command.

Benefits of Docker Compose

- Single host deployment - This means you can run everything on a single piece of hardware.
- Quick and easy configuration – Due to usage YAML scripts for configuration.
- High productivity - Docker Compose reduces the time it takes to perform tasks.
- Security - All the containers are isolated from each other.

How Does a Docker Compose Work?



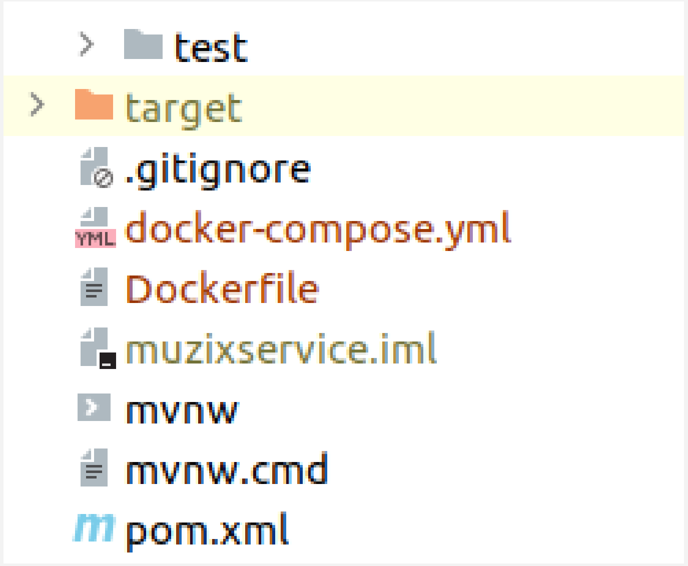
```
version: "3.7"
services:
 db:
  image: mysql:8.0.19
  restart: always
  environment:
   - MYSQL_DATABASE=example
   - MYSQL_ROOT_PASSWORD=password
 app:
  build: app
  restart: always
 web:
  build: web
  restart: always
  ports:
   - 80:80
```

Source: https://azuremarketplace.microsoft.com/cs-cz/marketplace/apps/cloud-infrastructure-services.docker-compose-ubuntu20?tab=Overview

Docker Compose Workflow

Following are steps to use Docker Compose:

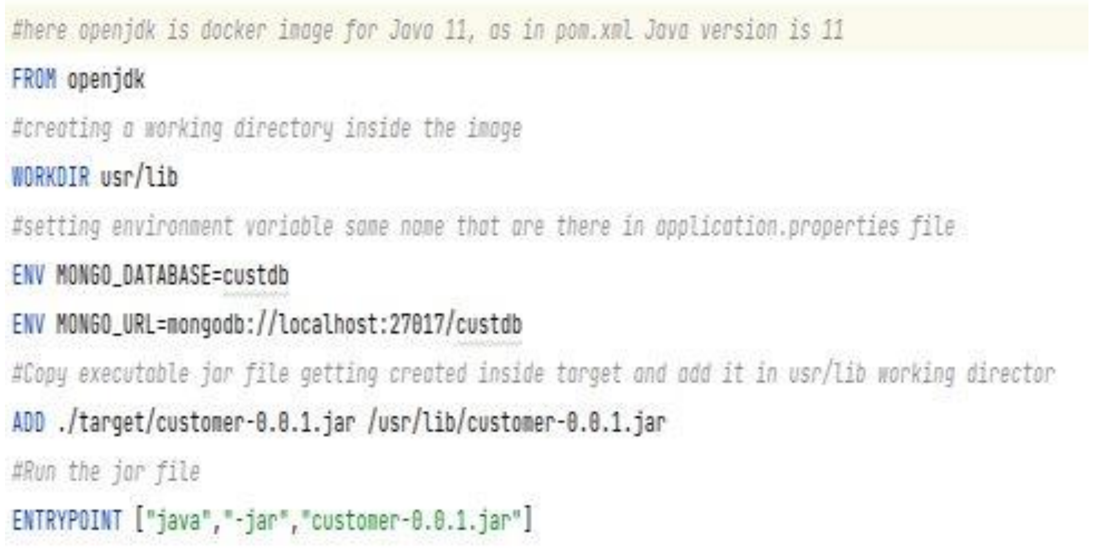
- Create Dockerfile for Spring Boot application.
- Create docker-compose.yml file in the root of the project.
- Define the services and their relation to each other in a docker-compose file.
- Start the docker-compose file.



```
> test
> target
  .gitignore
  docker-compose.yml
  Dockerfile
  muzixservice.iml
  mvnw
  mvnw.cmd
  pom.xml
```

Implement Docker Compose for the Backend Application

Create Dockerfile

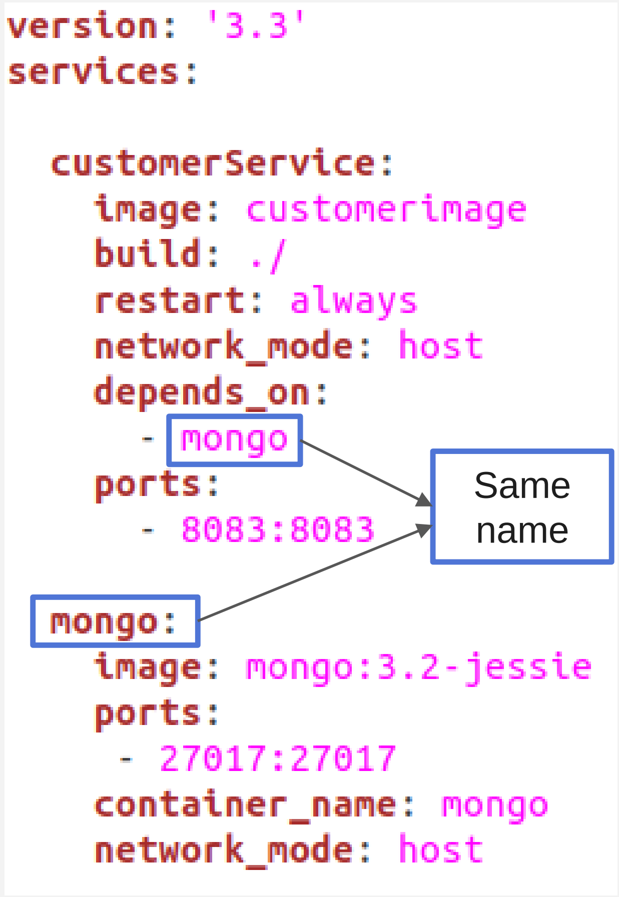


```
#here openjdk is docker image for Java 11, as in pom.xml Java version is 11
FROM openjdk
#creating a working directory inside the image
WORKDIR usr/lib
#setting environment variable same name that are there in application.properties file
ENV MONGO_DATABASE=custdb
ENV MONGO_URL=mongodb://localhost:27017/custdb
#Copy executable jar file getting created inside target and add it in usr/lib working director
ADD ./target/customer-0.0.1.jar /usr/lib/customer-0.0.1.jar
#Run the jar file
ENTRYPOINT ["java","-jar","customer-0.0.1.jar"]
```

- Docker can build images automatically by reading the instructions from a Dockerfile.
- A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.
- It describes step-by-step instructions of all the commands you need to run to assemble a Docker Image.

docker-compose.yml

- Version: Specifies the version of `docker-compose.yml` file.
- Services: Under this we mention all the services that we want to containerize.
  - `image` – This is the name of the image.
  - `build` - This is the current path of the Dockerfile.
  - `restart` – if the application is dependent on any database, there are chances that the application container may start before the database container. Therefore, until the database container is up and running, the application container must restart.
  - `depends-on` – The name of the container the application is dependent on.

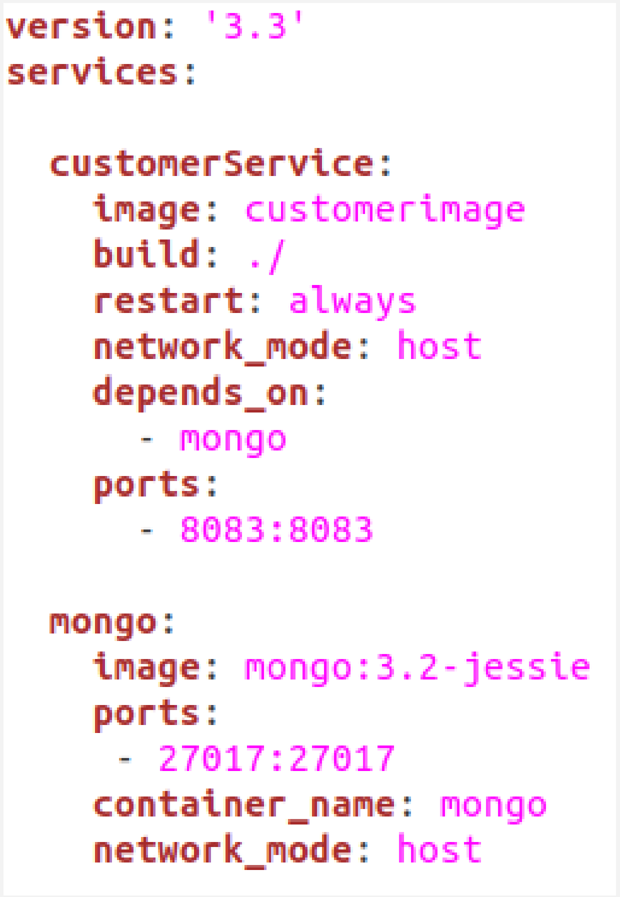


```
version: '3.3'
services:

  customerService:
    image: customerimage
    build: ./
    restart: always
    network_mode: host
    depends_on:
      - mongo
    ports:
      - 8083:8083

  mongo:
    image: mongo:3.2-jessie
    ports:
     - 27017:27017
    container_name: mongo
    network_mode: host
```

docker-compose.yml

- `network-mode` – If you use the `host` network mode for a container, then that container's network stack is not isolated from the Docker host (the container shares the host's networking namespace).
- `ports` -

  <exposed_port>:<actual_port>
- `container_name` – The name given to a container.



```
version: '3.3'
services:

  customerService:
    image: customerimage
    build: ./
    restart: always
    network_mode: host
    depends_on:
      - mongo
    ports:
      - 8083:8083

  mongo:
    image: mongo:3.2-jessie
    ports:
     - 27017:27017
    container_name: mongo
    network_mode: host
```

Commands to Run the docker-compose.yml

- `docker-compose up --build`
  - To run the docker-compose file build all the images and run the container.
- `docker-compose down`
  - Stop all the containers and remove all the containers.

Quick Check

________is a tool for defining and running a multi-container Docker applications.

1.Docker Swarm

2.Docker Cloud

3.Docker Compose

4.Docker Hub

Quick Check: Solution

_________is a tool for defining and running a multi-container Docker applications.

1.Docker Swarm

2.Docker Cloud

3.**Docker Compose**

4.Docker Hub

Dockerize Customer Application

Dockerize the entire backend application created in the previous session using a docker-compose file.

DEMO

Key Takeaways

- Workflow of Docker Compose
- Docker Compose tool
- Dockerizing a Backend application

Thank you!