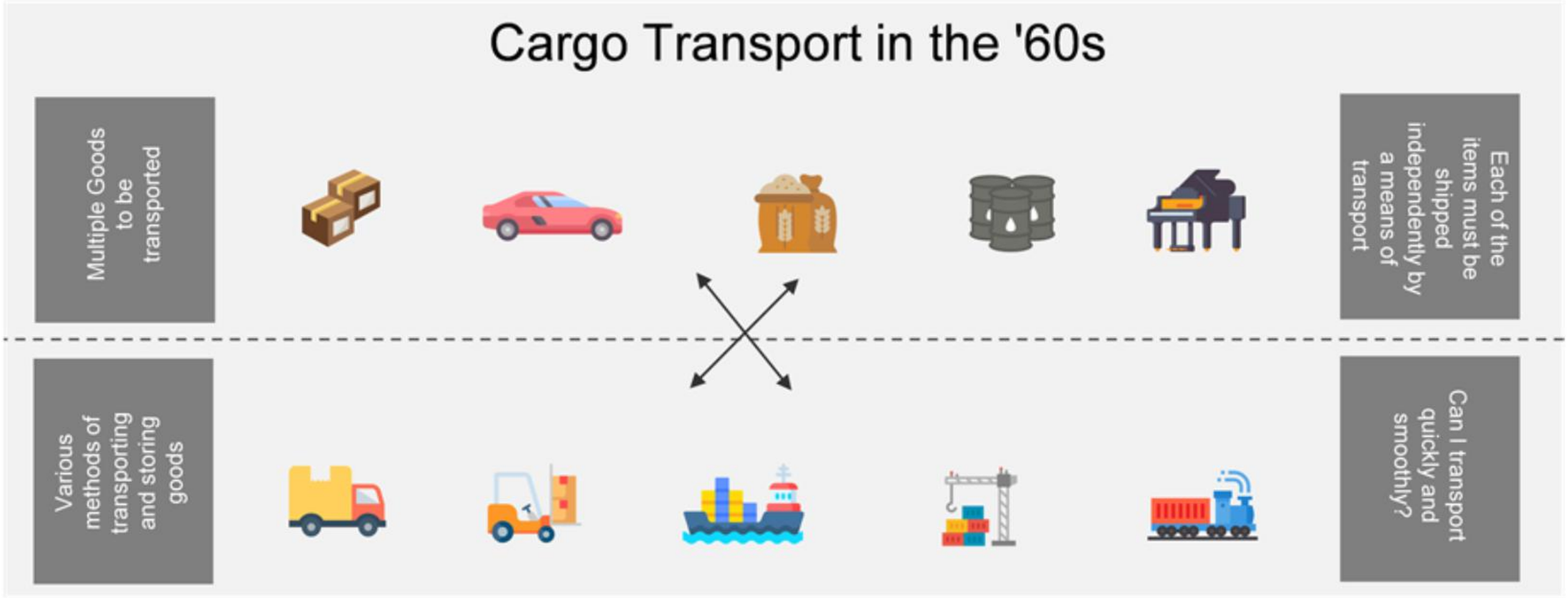
Traditional Cargo Shipping



- Can all the items be transported at the same time?
- How easy will the loading and unloading of the goods be if the items must be unloaded from a train and put into a ship?

Modern Cargo Shipping

Time to Think

- A team develops software as per a customer's requirements. How does the team deploy all the components of a software application for the customer?
- How can you ship or shift this software to the customer's system?
- How can you ensure the required versioning between development, test, and production beds?
- How can you be sure that the software runs across all the platforms?

Challenges of Shipping a Software

It works on my machine

Containerize RESTful Services and Database by Using Docker

Learning Objectives

- Explore Docker
- Understand Components of Docker
- Explore Containerize MongoDB
- Understand Networking in Docker
- Containerize MySQL
- Containerize the Spring Boot Application

Explore Docker

Slide Note

Containers allow you to package your application software and all its dependencies into one process package. Containers can be managed by different versions, allowing teams to easily replicate your same application instance across developers/testers or in your cluster.

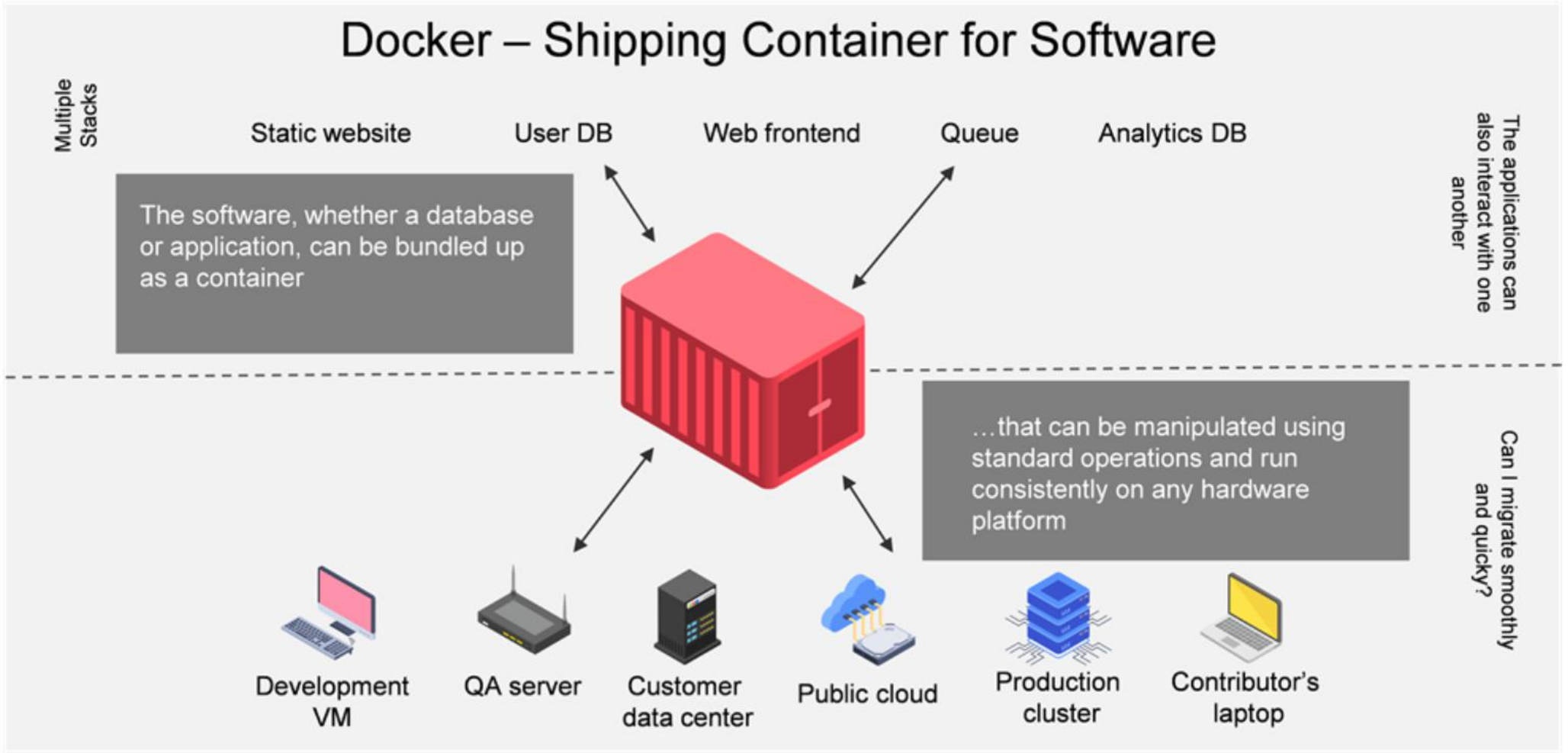
A container image includes everything needed to run it: code, runtime, system tools, system libraries, settings etc. Containers are lightweight, can work stand-alone and are available for both Linux and Windows-based apps. The best thing about container-running software is that it will always run the same, regardless of the environment. Docker containers isolate applications from one another and the underlying infrastructure. For example, containers help reduce conflicts between teams running the same application but in different environments like Development, Testing, Staging or Production.

Docker

- Docker is a container technology used to build, ship, and run an application in any device.
- Docker packages applications into standardized units called containers.
- The containers have everything the application needs to run, including libraries, system tools, code, and runtime.
- Docker can be seen as an operating system for containers.
- For example,
  - A database such as MySQL or MongoDB can be packaged as a Docker container.
  - A Spring Boot REST API can also be packaged as a Docker container.

Slide Note

Docker – Container Technology



Slide Note

Advantages of Using Docker

- There are many benefits of using Docker for building and deploying applications:
  - Flexible resource sharing
  - Scalability, multiple containers can be placed in a single host
  - The application can be run on hardware that is much cheaper than standard servers
  - Fast deployment, ease of creating new instances, and faster migrations
  - Ease of moving and maintaining your applications
  - Better security since access is restricted as the application runs inside a container

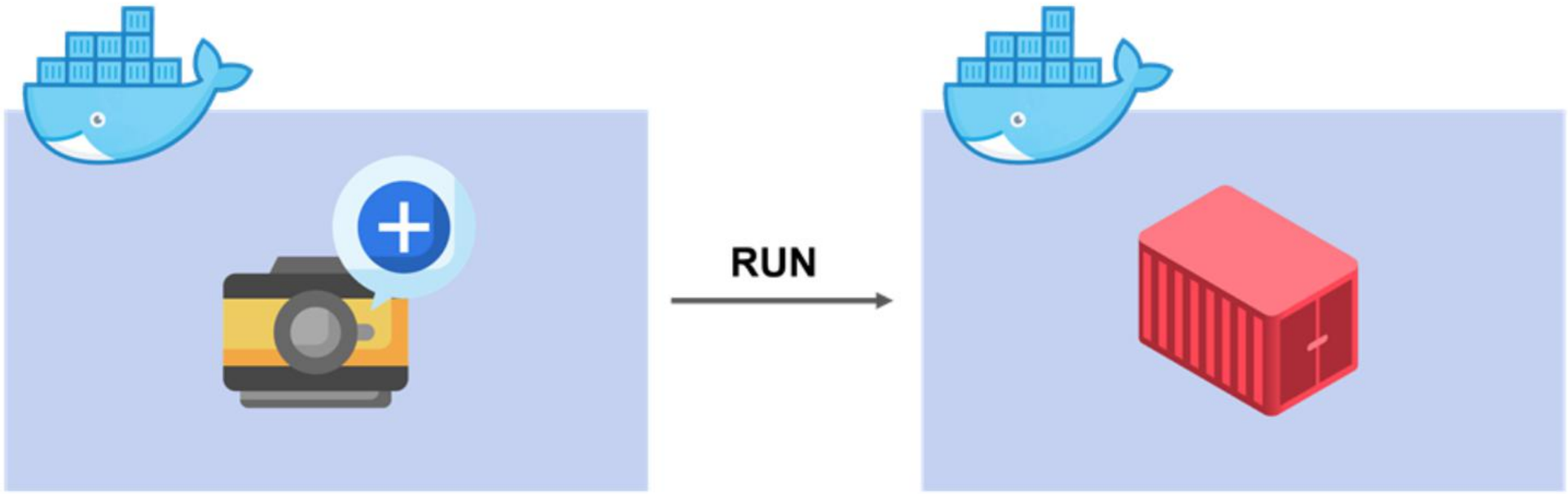
Slide Note

Understand Components of Docker

Slide Note

Docker containers are built from Docker images.

Docker Image and Container



Docker Image

Docker Container

Slide Note

Docker Image

- A Docker image is a template that includes all the dependencies, deployment, and execution configurations used by a Docker container at runtime.
- The image is immutable.
- Docker images have intermediate layers that increase reusability, decrease disk usage, and speed up Docker build by allowing each step to be cached. These intermediate layers are not shown by default.
- Multiple Docker images can be stacked in a single Docker container.
- The stacked images form the Docker container's file system.
- It can be created, started, stopped, or deleted.

Slide Note

Each Docker image references a list of read-only layers.

Layers are stacked on top of each other to form a base for a container's root filesystem, just like the pizza that is built from a base layer with other layers built on top of it.

The Docker storage driver is responsible for stacking these layers and providing a single unified view.

Each of the files that make up a Docker image is known as a **layer**. For example, a REST API needs Java and a database like Mongo or MySQL to function, thus the JDK and Mongo Docker make up layers of the Docker image of the REST API.

These layers form a series of **intermediate images**, built one on top of the other in stages, where each layer is dependent on the layer immediately below it.

The **hierarchy** of the layers is key to efficient lifecycle management of Docker images. Thus, you should organize layers that change most often as high up the stack as possible.

This is because, when you make changes to a layer in your image, Docker not only rebuilds that particular layer, but all layers built from it.

Therefore, a change to a layer at the top of a stack involves the least amount of computational work to rebuild the entire image.

Layers in Docker Images

My pizza Image

Layer 5 (Herbs)

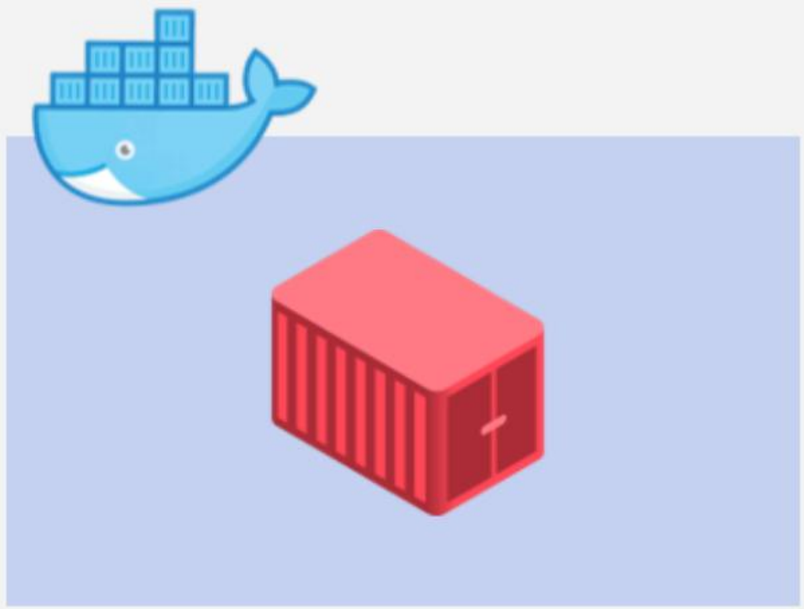
Layer 4 (Toppings)

Layer 3 (Cheese)

Layer 2 (Sauce)

Layer 1 (Base)

Slide Note



Docker Container

Docker Container

- A Docker container is created from a Docker image.
- A Docker container is a runnable instance of the Docker image.
- To build a Docker image and execute a container, multiple commands need to be executed.

Slide Note

Building a Docker Image

- The Docker image can be built in two ways:
  - From scratch
  - Using existing images from the Docker registry
- The Docker images of Java JDK, databases like MySQL, and MongoDB can be downloaded from the Docker registry.
- Docker Registry is where the Docker images are stored.
- The registry can be:
  - A local repository or folder in the local file system, or
  - A public repository like Docker Hub that allows multiple users to collaborate in building an application
- Multiple teams within the same organization can exchange or share containers by uploading them to the Docker Hub, which is a cloud repository like GitHub.

Slide Note

Installing Docker

- To install Docker on Windows:
  - Install Docker Desktop, and
  - Use the PowerShell or command prompt to run Docker commands.
- You must have administrator rights on your system to install and run Docker.
- Steps to install Docker Desktop on Windows
- Steps to install Docker on Linux

Slide Note

Explore Containerize MongoDB

Slide Note

Steps to Dockerize MongoDB

1. Pull Mongo Image from Docker Hub

2. Run the Mongo container from the image pulled

Slide Note

Pull the Mongo Image

Slide Note

Pull Mongo Image From Docker Hub

- Sign in to Docker hub,
- Search for MongoDB images.

- Use the image from the official mongo site.

Slide Note

Command to Pull the Mongo Image From Docker Hub

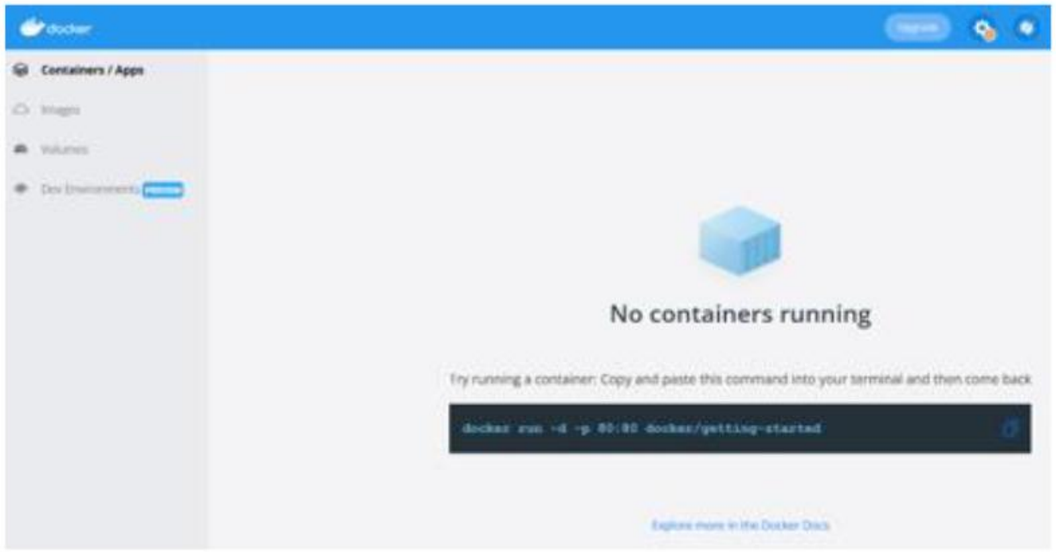
- Click on Mongo image.

- Copy the command to pull the image.

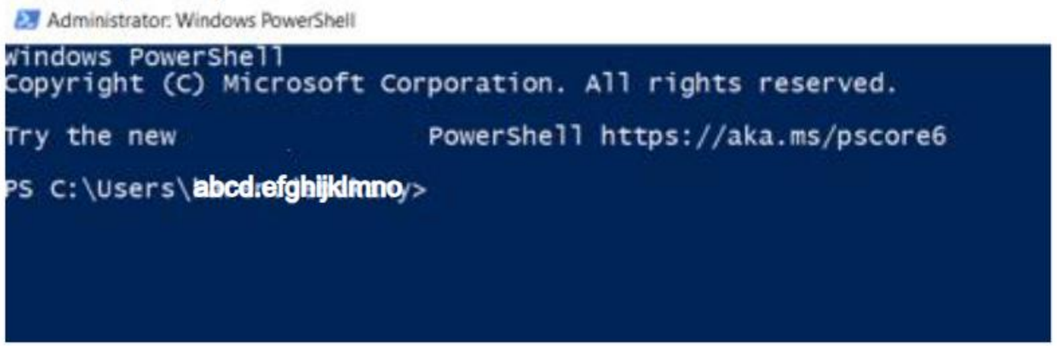
Slide Note

Start Docker Desktop to Run the Image as a Mongo Container

- Start Docker Desktop.



- Start Power shell or command prompt.



Slide Note

Pull the MongoDB Image

- Execute the Docker pull command on the PowerShell.

```
> docker pull mongo
```

Administrator: Windows PowerShell

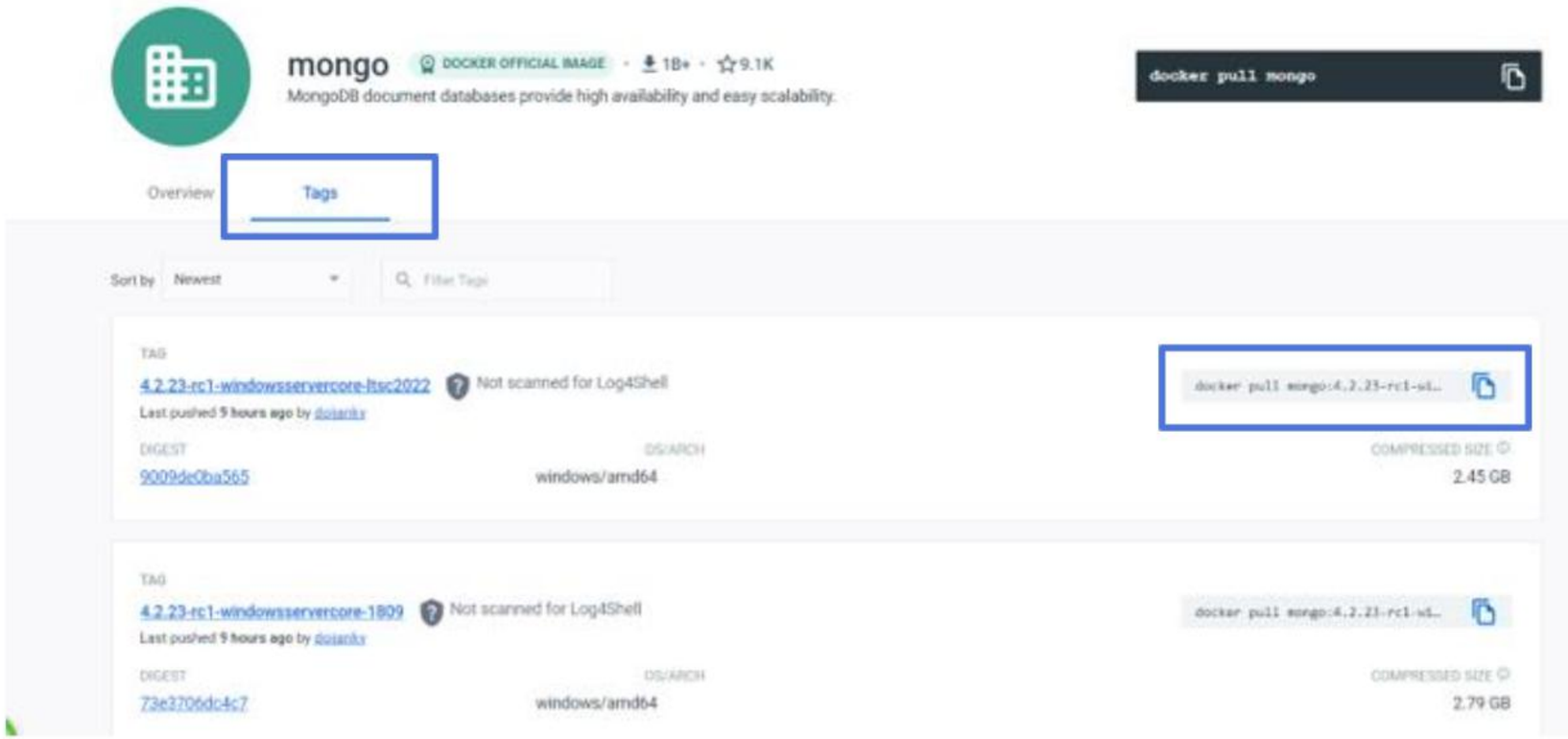
```
PS C:\Users> docker pull mongo
Using default tag: latest
latest: Pulling from library/mongo
675920708c8b: Pull complete
6f9c8c301e0f: Pull complete
73738965c4ce: Pull complete
7fd6635b9ddf: Pull complete
73a471eaa4ad: Pull complete
bcf274af89b0: Pull complete
04fc489f2a3b: Pull complete
6eff8a505292: Downloading [==============================>                    ]  122.5MB/193.2MB
a5ef4431fce7: Download complete
```

- Note that the tag used is the latest, i.e., the latest image will be downloaded since a version was not specified.
- We can also download older versions of the Mongo image by specifying the version in the tag.

Slide Note

Image Tag

- You can check for versions of Mongo from the Tag tab of the Mongo image page on the Docker hub and use the command as shown below.



Slide Note

Command to View the Images

- To view the images pulled from Docker hub, use the command

```
> docker images
```

```
REPOSITORY    TAG         IMAGE ID        CREATED          SIZE
mongo         latest      d34d21a9eb5b    3 weeks ago      693MB
```

- An Image ID is assigned by Docker to the image created.

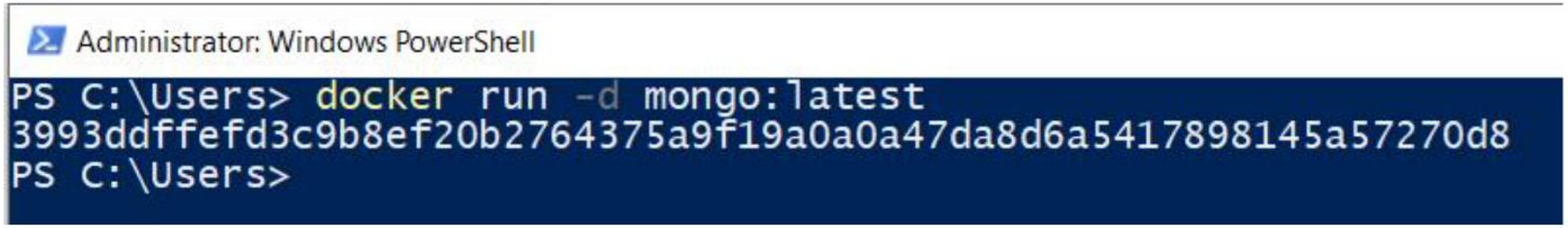
Slide Note

Run MongoDB Container

Slide Note

Create the Mongo Container

- Create the container
  - `> docker run -d mongo:latest`
  - The `-d` switch is used to run the container in detached mode and is not compulsory, which implies that the command will be run in the background.
  - `mongo: latest` is the image created earlier.



Slide Note

View Containers

- To view containers that are running currently,

```
> docker ps
```

```
PS C:\Users> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS       NAMES
3993ddffefd3   mongo:latest   "docker-entrypoint.s…"   31 minutes ago   Up 31 minutes   27017/tcp   sleepy_dhawan
```

Unique Id for container

Image that was pulled from Docker hub

Status Up indicates a container that is running

This Mongo container is listening on 27017 port, which is the default port of Mongo

- To view all containers both running and stopped, use the command below:

```
> docker ps -a
```

Slide Note

Providing Names to a Container

- If names are not given to a container when running the `docker run` command, Docker provides a default name.

```
PS C:\Users> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS       NAMES
3993ddffefd3   mongo:latest   "docker-entrypoint.s…"   31 minutes ago   Up 31 minutes   27017/tcp   sleepy_dhawan
```

- The name of the container created earlier is `sleepy_dawn;` to eliminate this we can use the command below, use the `--name` switch, and provide a meaningful name.

```
docker run -d --name mongo-container mongo:latest
```

```
PS C:\Users> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS       NAMES
1de6a6d1dc7b   mongo:latest   "docker-entrypoint.s…"   8 seconds ago    Up 7 seconds    27017/tcp   mongo-container
```

Slide Note

The -it instructs Docker to allocate a pseudo-TTY connected to the container's stdin; creating an interactive bash shell.

Execute Mongo Shell via Mongo Container

- To get into the Mongo shell of the Mongo container created, use the command below:
- `> docker exec -it mongo-container bash`

```
PS C:\Users> docker exec -it mongo-container bash
root@1de6a6d1dc7b:/# mongosh
```

The `-it` switch provides a terminal as an interactive bash shell

- This will provide a bash terminal from where mongosh can be provided to run a Mongo shell.

```
root@1de6a6d1dc7b:/# mongosh
Current Mongosh Log ID:
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTi
meoutMS=2000&appName=mongosh+1.5.4
Using MongoDB:          6.0.1
Using Mongosh:          1.5.4

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2022-09-27T11:40:26.344+00:00: Using the XFS filesystem is strongly recommended with th
e WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
   2022-09-27T11:40:26.849+00:00: Access control is not enabled for the database. Read and
 write access to data and configuration is unrestricted
   2022-09-27T11:40:26.849+00:00: /sys/kernel/mm/transparent_hugepage/enabled is 'always'.
 We suggest setting it to 'never'
   2022-09-27T11:40:26.849+00:00: vm.max_map_count is too low
------

------
   Enable MongoDB's free cloud-based monitoring service, which will then receive and displ
ay
   metrics about your deployment (disk utilization, CPU, operation statistics, etc).

   The monitoring data will be available on a MongoDB website with a unique URL accessible
 to you
   and anyone you share the URL with. MongoDB may use this information to make product
   improvements and to suggest MongoDB products and deployment options to you.

test>
   To permanently disable this reminder, run the following command: db.disableFreeMonitori
ng()
test>
```

Mongo executes on port 27017 inside the Docker container, which is the default port for MongoDB

Slide Note

Quick Check

Which is the command to list all the running containers?

1. Docker rm <container id>

2. Docker exec -it <container id> bash

3. Docker ps

4. All of the above

Slide Note

Quick Check: Solution

Which is the command to list all the running containers?

1. Docker rm <container id>

2. Docker exec -it <container id> bash

3. Docker ps

4. All of the above

Slide Note

Dockerize MongoDB

1. Dockerize a MongoDB database.
2. Pull the Mongo image from Docker Hub.
3. Run the Mongo container.
4. Execute the Mongo terminal.
5. Create a collection, users.
6. Insert username, email, and phone number into the users collection.

DEMO

Menu

00:34/00:00:57

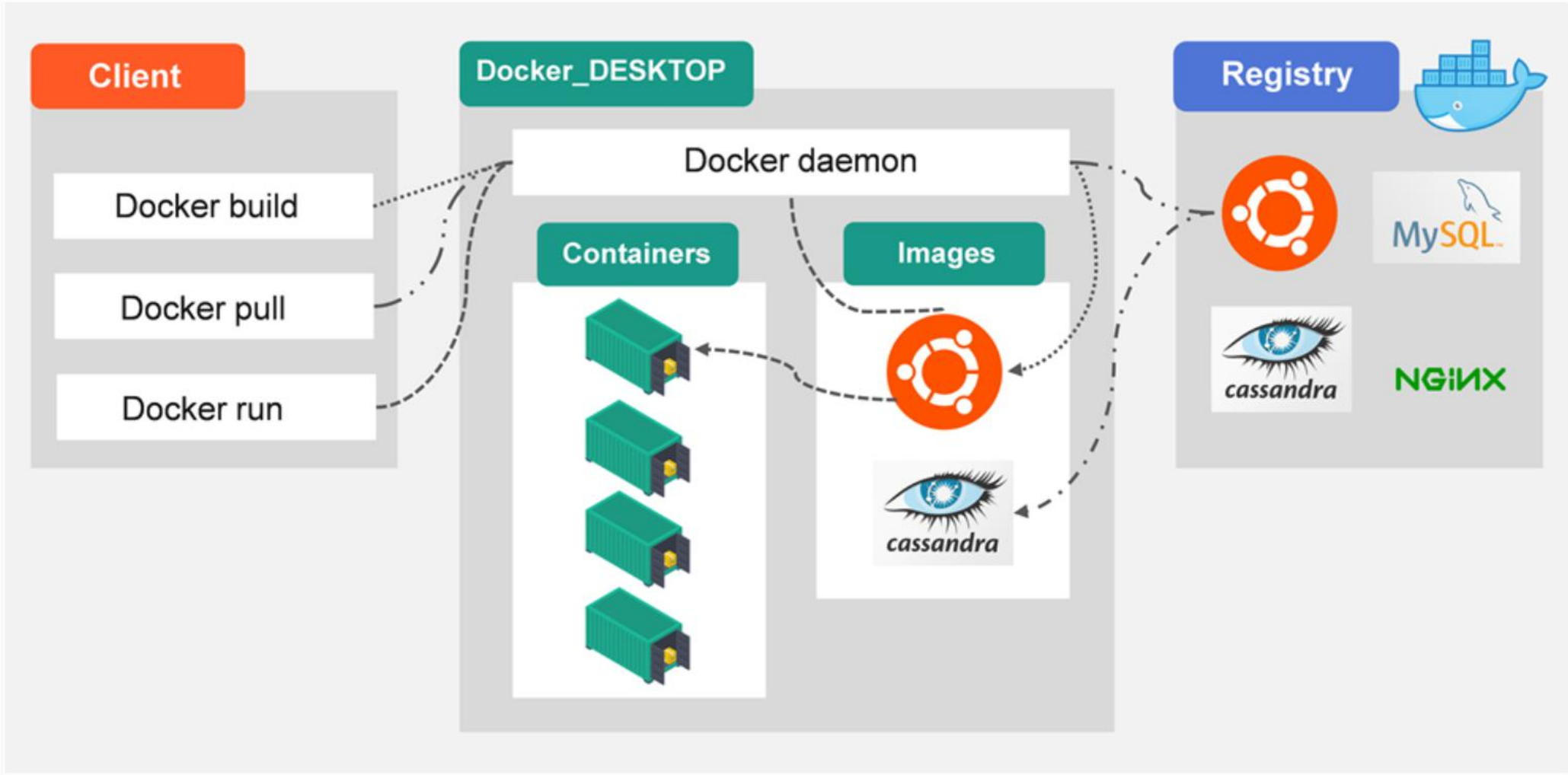
34/ 57

Slide Note

Understand Networking in Docker

Slide Note

Docker Architecture



Slide Note

Docker Architecture (contd.)

- Docker uses a client-server architecture.
- The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing the Docker containers.
- The Docker client and daemon can run on the same system or can connect to a Docker client on a remote Docker daemon.
- The Docker client and daemon communicate using a REST API over UNIX sockets.
- The Docker client is the PowerShell or command prompt where the Docker commands are run.
- The Docker Desktop is the host for running the Docker containers and building the images and contains the Docker daemon.
- The registry is the Docker hub where the predefined Docker images are present.

Slide Note

Communication Between Containers

- A Spring Boot application uses MySQL or MongoDB to store data.
- The Spring Boot application and the database can be Dockerized.
- Both containers must communicate with each other in the Docker environment.
- Since Docker communication happens through UNIX sockets, the Docker Desktop application is used.
- Docker Desktop is an easy-to-install application for Mac or Windows environments that helps to build and share containerized applications.
- Docker Desktop includes the Docker daemon `Dockerd`, the Docker client `Docker`, `Docker compose`, Docker Content Trust, Kubernetes, and Credential Helper.
- Docker expects the containers that need to communicate with each other to run on the same network.

Slide Note

Containerize MySQL

Slide Note

Create a Docker Network

- The command below is used to create a Docker network:

```
Docker network create <name of the network>
```

```
PS C:\Users> docker network create user-network
8236c75aa3e45915bfc6592f947e03f0e9be871729e346dbf8a06399d33b893c
```

- To view all the networks in Docker, use the command below:

```
Docker network ls
```

```
PS C:\Users> docker network ls
NETWORK ID     NAME           DRIVER    SCOPE
d51a4b2d1855   bridge         bridge    local
ca6c157653cd   host           host      local
8379da0c5dd0   none           null      local
8236c75aa3e4   user-network   bridge    local
```

Slide Note

Dockerize MySQL

1. Pull the MySQL image using the following command:

```
Docker pull mysql
```

2. Run the image to create the container on the network created earlier.
   - Note that in MySQL a password is required to connect to the MySQL shell.

```
Docker run -it --network user-network --name mysqlservice -e
MYSQL_ROOT_PASSWORD=root -d mysql
```

3. Execute the MySQL shell from the Docker container.

```
Docker exec -it mysqlservice bash
```

4. Enter the bash and give `mysql -u root -p` and enter the password 'root' as specified in step 2.

Slide Note

Containerize the Spring Boot Application

Slide Note

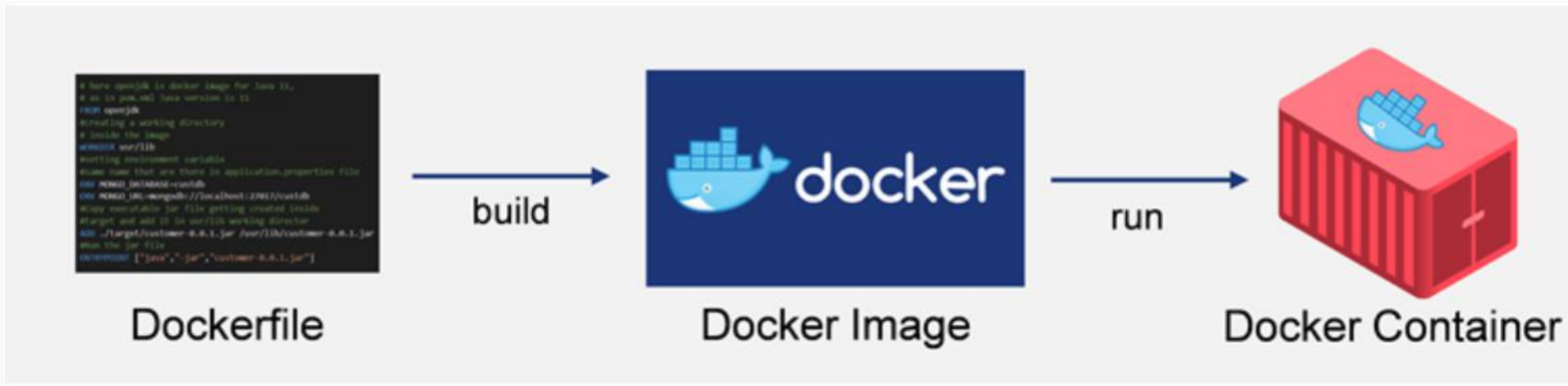
Dockerize the Spring Boot Application

- To Dockerize the Spring Boot application built earlier:
  - Create a Docker image of the Spring Boot application
  - Run the Docker image and create a container
- The Docker image of the application must be built from scratch.
- The image will not be available in Docker hub like Mongo or MySQL.

Slide Note

Steps to Create the Spring Boot Container

- Create a `Dockerfile` that contains the configurations necessary to build the image.
- Build the `Dockerfile` to create the image.
- Run the image to create the Spring Boot application container.



Slide Note

Dockerfile

- A `Dockerfile` is a text document that contains all the commands a user can call on the command line to assemble an image.
- Docker can build images automatically by reading instructions from the `Dockerfile`.
- `Dockerfile` contains step-by-step instructions for all the commands required to assemble a Docker image.

```
#openjdk is the docker image for Java JDK the application will use
FROM openjdk
#creating a working directory inside the docker container of the application
WORKDIR usr/lib
#Copy the executable jar file that is created
#in the target folder and add it to the usr/lib working directory
ADD ./target/authenticationservice-2.7.3.jar /usr/lib/authenticationservice-2.7.3.jar
#Run the jar file using the java -jar command
ENTRYPOINT ["java","-jar","authenticationservice-2.7.3.jar"]
```

Slide Note

Build the JAR File of the Application

- Execute the `mvn clean compile package` command to generate the JAR file in the target folder.
- This JAR file will be used by the Docker image to build the Spring Boot application's Docker image and is provided in the `Dockerfile`.

Slide Note

The application.properties File

- In the `application.properties` file, change the url of MySQL from `localhost` to `mysqlservice`, which is the name of the MySQL container.

```
server.port=8084
spring.datasource.url = jdbc:mysql://mysqlservice:3306/user_db_db?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
server.error.include-message=always
```

Slide Note

Build the Docker File and Create the Docker Image

- Build the Docker image using the command below at the root of the application using the Docker build command:

```
docker build -t user-app:v1 .
```

- The `-t` switch specifies that a name and version is given for the image.
- The " . " specifies that the Dockerfile is present in the root of the current folder from where the command is executed.

```
PS D:\Backend-Spring-Revamp-Solution\Course3\Demonstrations\demo-1-authenticate-the-application-using-jwt>
docker build -t user-app:v1 .
[+] Building 12.9s (9/9) FINISHED
```

Slide Note

Run the Spring Boot Container

- Run the Spring Boot container from the Docker image using the Docker run command and execute it over the same network as the MySQL container.

```
docker run -d -p 8084:8084 --name user-spring-app
--network user-network user-app:v1
```

- Verify if the containers created are up and running using the `Docker ps` command.

Slide Note

Quick Check

What is the command used to build a Spring Boot Docker image?

1. `Docker build -t <name of the image with version> .`
2. `Docker build -t <name of the image>`
3. `Docker build -t  .`
4. `Docker build .`

Slide Note

Quick Check: Solution

What is the command used to build a Spring Boot Docker image?

1. `Docker build -t <name of the image with version> .`
2. `Docker build -t <name of the image>`
3. `Docker build -t  .`
4. `Docker build .`

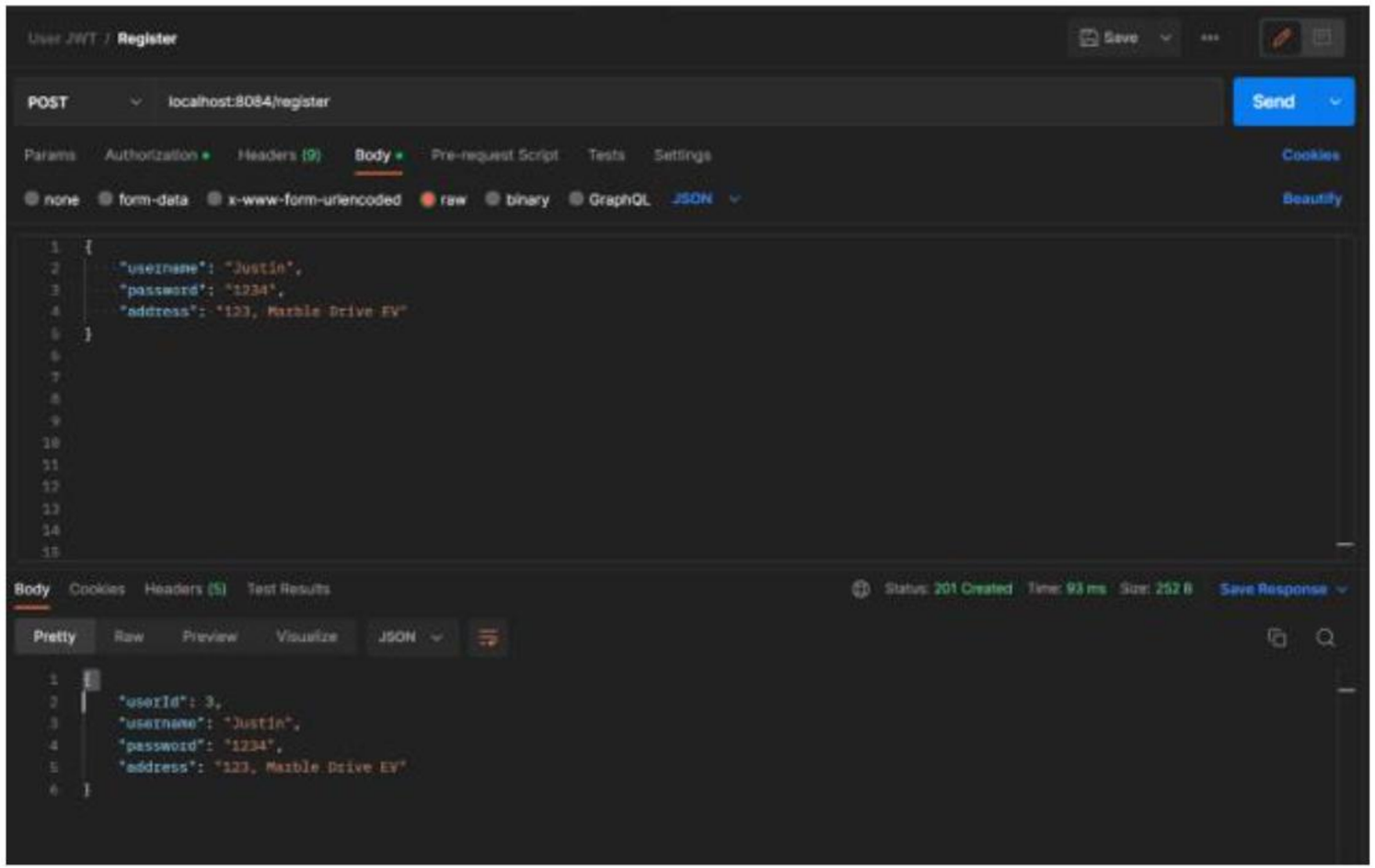
Slide Note

Postman Output

Slide Note

Register a New User

- Run the containers and verify the API calls made in Postman.



Menu

00:53:00:00:57

53/ 57

Slide Note

Log in to the Application

POST localhost:8084/login

Params Authorization • Headers (9) Body • Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "username": "Justin",
    "password": "1234"

}
```

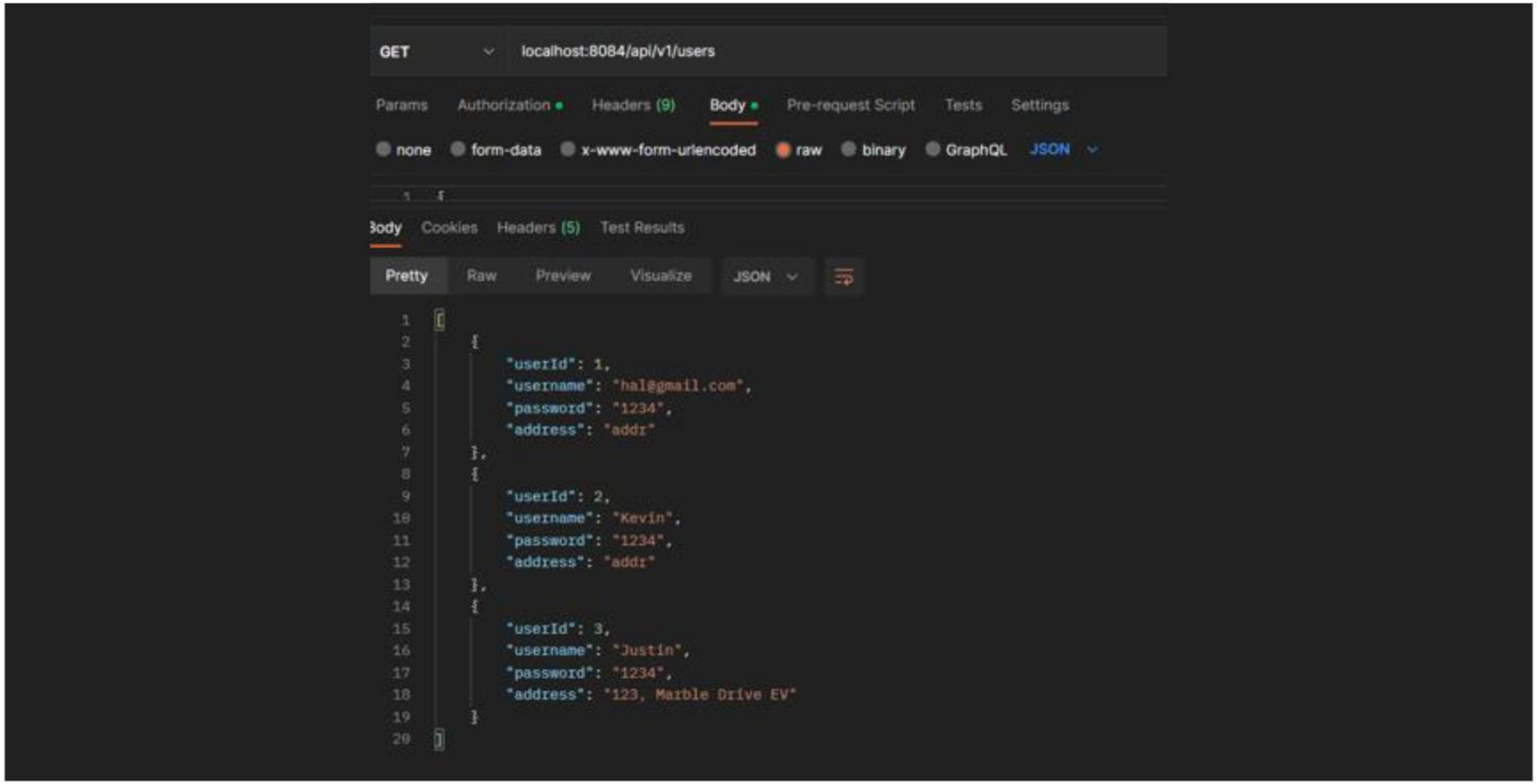
Body Cookies Headers (5) Test Results Status: 200 OK Time: 30 ms Size: 325 B

Pretty Raw Preview Visualize JSON


```
{
    "message": "User Successfully logged in",
    "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJKdXN0aW4iLCJpYXQiOjE2NjQ0MjY5ODh9.7_gnUFNg96vujT4ghE0SxueFFCAoVO4-ckjgaraQE4g"
}
```

Slide Note

Get All Users



Slide Note

Dockerize the Spring Boot Application

Create a Spring Boot REST API that stores the user data of a web application. The users must be authenticated to access certain resources. Use MySQL as the database to store user information.

Containerize the Spring Boot application and MySQL database.
Run the containers on the same network.
Test the output in Postman.

Click here for the solution.

DEMO