

# Containerization & Documentation



**SoftUni Team**  
**Technical Trainers**



**SoftUni**



**Software University**

<https://softuni.bg>

# Table of Contents

1. Containerization
2. Introduction to Docker
3. Docker in Action
4. Create Our Own Images
5. Documentation
6. Swagger 3



sli.do

**#java-web**

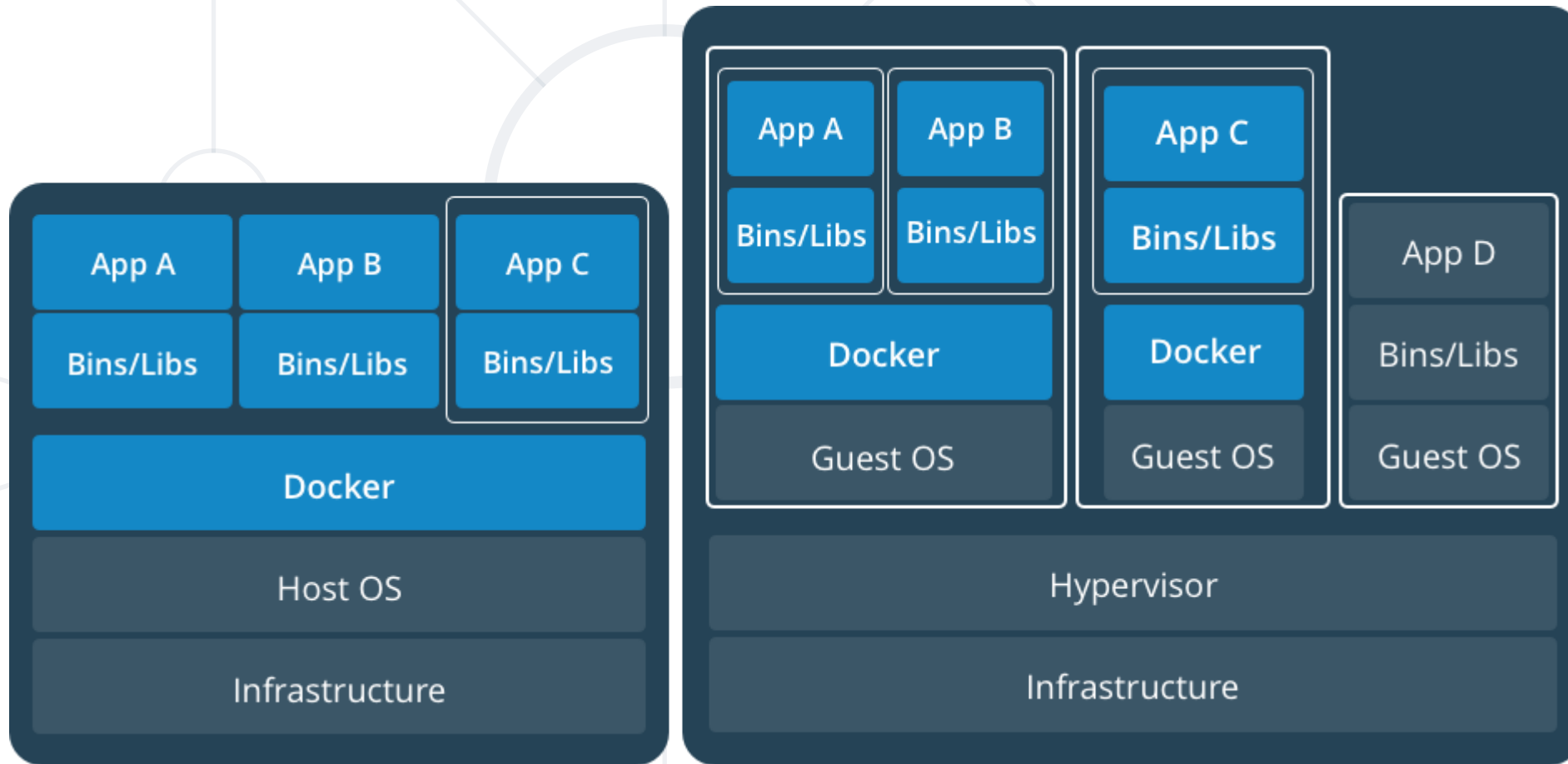


# Containers and Docker

Past. Present. Future

!! OS-level virtualization refers to an operating system paradigm in which the kernel allows the existence of **multiple isolated user space instances** known as **containers, zones, jails, ...** !!

# VMs and Containers



- **rkt** by CoreOS
  - Application container engine
  - <https://coreos.com/rkt>
- **Docker** by Docker Inc
  - Application container engine
  - <https://www.docker.com/>

# VMs vs Containers

- VMs virtualize the hardware
  - Complete isolation
  - Complete OS installation. Requires more resources
  - Runs almost any OS
- Containers virtualize the OS
  - Lightweight isolation
  - Shared kernel. Requires fewer resources
  - Runs on the same OS

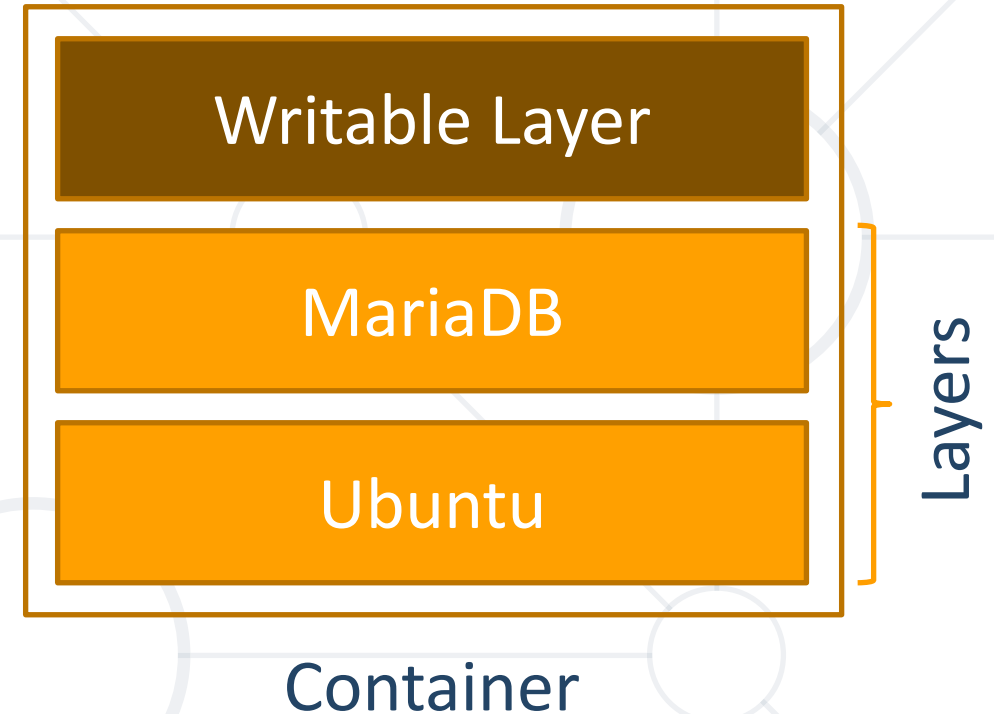




# Containers Concepts (Docker View)

- **Container image** shows the state of a container, including registry or file system changes
- **Container OS image** is the first layer of potentially many image layers that make up a container
- **Container repository** stores container images and their dependencies

- Container
  - Containers are processes with much more isolation
- Image
  - Images provide a way for simpler software distribution

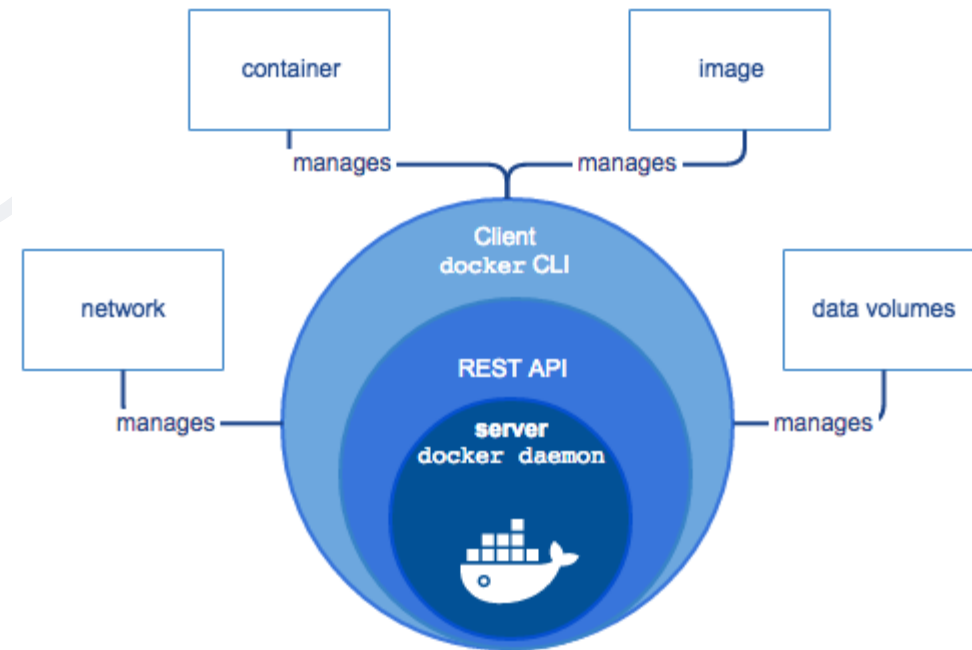




**Docker**

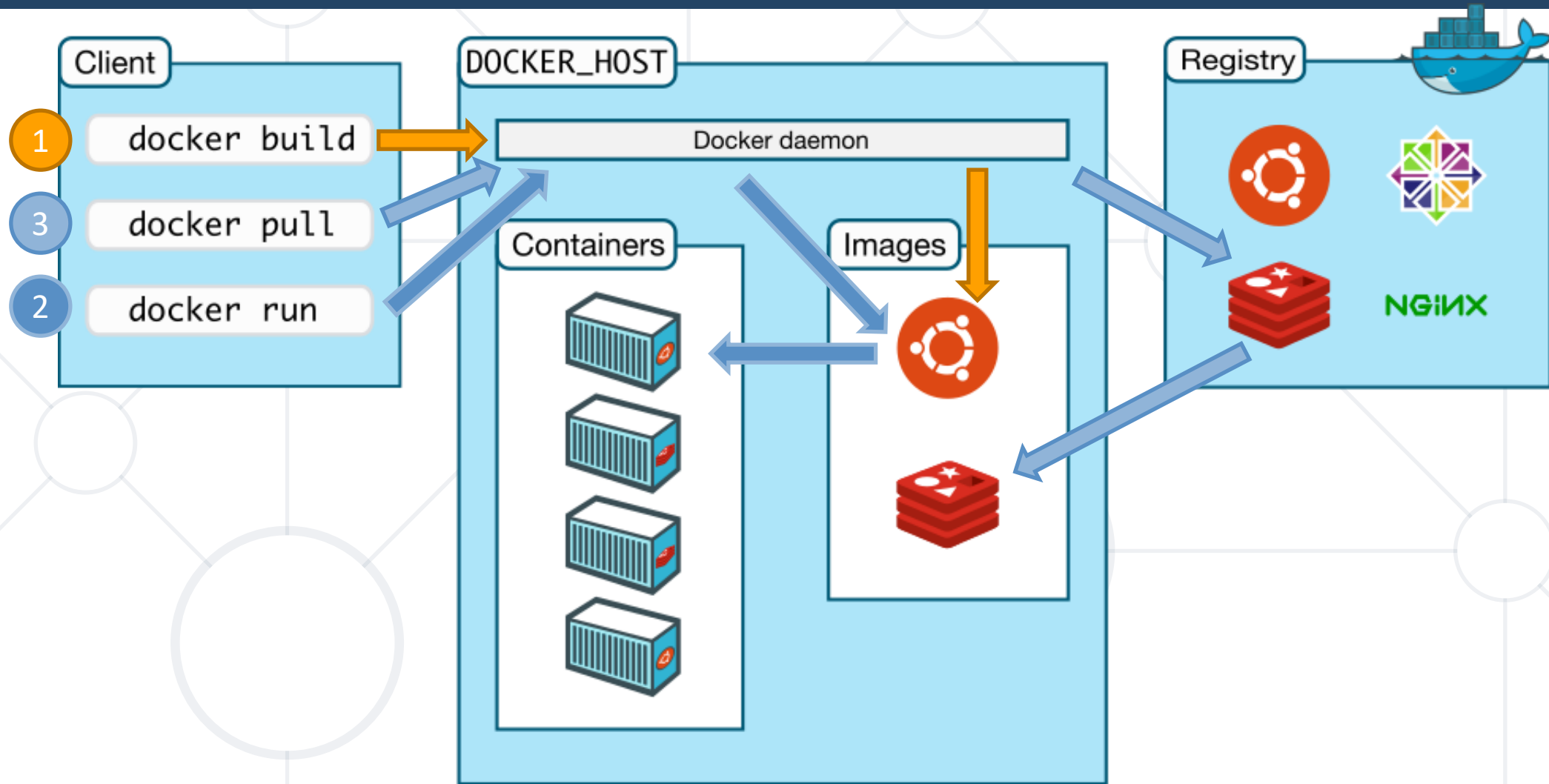
Whole New World

- Docker Mission – **Build, Ship, Deploy**
- Client-server application
- Components
  - dockerd daemon
  - REST API
  - docker CLI



- Provided by Docker
  - Cloud
    - Docker Hub (<https://hub.docker.com/explore/>)
    - Docker Store (<https://store.docker.com/>)
  - On-premise
- Provided by 3<sup>rd</sup> parties
  - Quay.io, Artifactory, Google Container Registry

# Workflow





# Docker Installation

Options

# What We Need to Know?

- **Two Editions** (Community and Enterprise)
- **Native Options**
  - Docker for Linux
  - Docker for MAC
  - Docker for Windows
- **Docker Toolbox** (deprecated) - All-in-one solution
  - For Mac and Windows



Deployment via package system (three channels – **stable**, **nightly**, and **test**), script, or archive

Specific requirements: OS version, Hypervisor, etc.





# Working with Docker

Commands

- Purpose
  - Pull an image or a repository from a registry

- Old syntax

```
docker pull [OPTIONS] NAME[:TAG|@DIGEST]
```

- New syntax

```
docker image pull [OPTIONS] NAME[:TAG|@DIGEST]
```

- Example

```
docker image pull ubuntu:latest
```

- Purpose
  - Run a command in a new container

- Old syntax

```
docker run [OPTIONS] IMAGE [COMMAND] [ARG]
```

- New syntax

```
docker container run [OPTIONS] IMAGE [COMMAND] [ARG]
```

- Example

```
docker container run -it ubuntu
```

- Purpose
  - List locally available images

- Old syntax

```
docker images [OPTIONS] [REPOSITORY[:TAG]]
```

- New syntax

```
docker image ls [OPTIONS] [REPOSITORY[:TAG]]
```

- Example

```
docker image ls fedora
```

- Purpose
  - List containers
- Old syntax

```
docker ps [OPTIONS]
```

- New syntax

```
docker container ls [OPTIONS]
```

- Example

```
docker container ls -a -q
```

- Purpose
  - Remove one or more containers

- Old syntax

```
docker rm [OPTIONS] CONTAINER [CONTAINER]
```

- New syntax

```
docker container rm [OPTIONS] CONTAINER [CONTAINER]
```

- Example

```
docker container rm weezy_snake
```

- Purpose
  - Remove one or more images

- Old syntax

```
docker rmi [OPTIONS] IMAGE [IMAGE]
```

- New syntax

```
docker image rm [OPTIONS] IMAGE [IMAGE]
```

- Example

```
docker image rm ubuntu fedora
```

- Purpose

- Start one or more stopped containers

- Old syntax

```
docker start [OPTIONS] CONTAINER [CONTAINER]
```

- New syntax

```
docker container start [OPTIONS] CONTAINER [CONTAINER]
```

- Example

```
docker container start -a -i 0cbf27183
```



# Restart / Container Restart

- Purpose
  - Restart a one or more containers

- Old syntax

```
docker restart [OPTIONS] CONTAINER [CONTAINER]
```

- New syntax

```
docker container restart [OPTIONS] CONTAINER [CONTAINER]
```

- Example

```
docker container restart 0cbf27183
```

# Stop / Container Stop

- Purpose
  - Stop one or more running containers

- Old syntax

```
docker stop [OPTIONS] CONTAINER [CONTAINER]
```

- New syntax

```
docker container stop [OPTIONS] CONTAINER [CONTAINER]
```

- Example

```
docker container stop 0cbf27183
```

# Unpause / Container Unpause

- Purpose
  - Unpause all processes within one or more containers

- Old syntax

```
docker unpause CONTAINER [CONTAINER]
```

- New syntax

```
docker container unpause CONTAINER [CONTAINER]
```

- Example

```
docker container unpause 0cbf27183
```

- Purpose
  - Attach to a running container

- Old syntax

```
docker attach [OPTIONS] CONTAINER
```

- New syntax

```
docker container attach [OPTIONS] CONTAINER
```

- Example

```
docker container attach 0cbf27183
```

- Purpose
  - Push an image or repository to a registry

- Old syntax

```
docker push [OPTIONS] NAME[:TAG]
```

- New syntax

```
docker image push [OPTIONS] NAME[:TAG]
```

- Example

```
docker image push repo-name/test:latest
```

- Purpose
  - Log into a Docker registry

- Old syntax

```
docker login [OPTIONS] [SERVER]
```

- New syntax

```
# same
```

- Example

```
docker login
```

- Purpose
  - Log out from a Docker registry

- Old syntax

```
docker logout [SERVER]
```

- New syntax

```
# same
```

- Example

```
docker logout
```



# **Image from File**

General Structure and Common Fields



# General Structure (Dockerfile)

- Script, composed of commands and arguments
- Always begins with FROM instruction

Comment

```
# Set the base image  
FROM nginx
```

Command  
(Instruction)

```
# Set the maintainer  
MAINTAINER John Smith
```

```
# Copy files  
COPY index.html /usr/share/nginx/html/
```

- Purpose
  - Defines the base image to use to start the build process

- Syntax

```
FROM <image>[:<tag>] [AS <name>]
```

- Example

```
# it is a good practice to state a version (tag)  
FROM ubuntu:18.04  
# for the latest version the tag could be skipped  
FROM ubuntu
```

- Purpose
  - Sets the author field of the image. It is deprecated

- Syntax

```
MAINTAINER <name>
```

- Example

```
# deprecated  
MAINTAINER John Smith  
# newer variant is this:  
LABEL maintainer="John Smith"
```

- Purpose
  - Used during build process to add software (forms another layer)

- Syntax

```
RUN <command>
```

- Example

```
# single command
```

```
RUN apt-get -y update
```

```
# more than one command
```

```
RUN apt-get -y update && apt-get -y upgrade
```

- Purpose
  - Copy files between the host and the container

- Syntax

```
COPY [--chown=<user>:<group>] <src>... <dest>
```

- Example

```
# Copy single file  
COPY readme.txt /root  
# Copy multiple files  
COPY *.html /var/www/html/my-web-app
```

- Purpose
  - Copy files to the image
- Syntax

```
ADD [--chown=<user>:<group>] <src>... <dest>
```

- Example

```
# Add single file from URL  
ADD https://softuni.bg/favicon.ico /www/favicon.ico  
# Add tar file content  
ADD web-app.tar /var/www/html/my-web-app
```

- Purpose
  - Informs Docker that the container listens on the specified ports

- Syntax

```
EXPOSE <port> [<port>/<protocol>...]
```

- Example

```
# single port  
EXPOSE 80  
# multiple ports  
EXPOSE 80 8080
```

- Purpose
  - Allows configuration of container that will run as an executable
- Syntax

*# exec form, this is the preferred form*

```
ENTRYPOINT ["executable", "param1", "param2"]
```

*# shell form*

```
ENTRYPOINT command param1 param2
```



- Purpose
  - Main purpose is to provide defaults for an executing container
- Syntax

*# exec form, this is the preferred form*

```
CMD ["executable", "param1", "param2"]
```

*# as default parameters to ENTRYPOINT*

```
CMD ["param1", "param2"]
```

*# shell form*

```
CMD command param1 param2
```

- Both **define** what **command** gets **executed** when **running** a container
- **Dockerfile** should specify at **least one** of them
- **ENTRYPOINT** should be defined when using the **container** as an **executable**
- **CMD** should be used as a way of defining **default arguments** for an **ENTRYPOINT** command or for **executing** an **ad-hoc** command in a container
- **CMD** will be overridden when **running** the container with **alternative** arguments

# CMD vs ENTRYPOINT

- Both have **exec** and shell **form**
- When used **together always** use their **exec** form

		ENTRYPOINT		
CMD		N/A	exec_entry p1_entry	["exec_entry", "p1_entry"]
	N/A	Error	/bin/sh -c exec_entry p1_entry	exec_entry p1_entry
	["exec_cmd", "p1_cmd"]	exec_cmd p1_cmd	/bin/sh -c exec_entry p1_entry	<b>exec_entry p1_entry exec_cmd p1_cmd</b>
	["p1_cmd", "p2_cmd"]	p1_cmd p2_cmd	/bin/sh -c exec_entry p1_entry	<b>exec_entry p1_entry p1_cmd p2_cmd</b>
	exec_cmd p1_cmd	/bin/sh -c exec_cmd p1_cmd	/bin/sh -c exec_entry p1_entry	exec_entry p1_entry /bin/sh -c exec_cmd p1_cmd

- Purpose
  - Build an image from a Dockerfile

- Old syntax

```
docker build [OPTIONS] PATH | URL | -
```

- New syntax

```
docker image build [OPTIONS] PATH | URL | -
```

- Example

```
docker image build -t new-image .
```

- Don't create large images
- Don't use only the "latest" tag
- Don't run more than one process in a single container
- Don't rely on IP addresses
- Put information about the author

<https://developers.redhat.com/blog/2016/02/24/10-things-to-avoid-in-docker-containers/>

<http://www.projectatomic.io/docs/docker-image-author-guidance/>



**Swagger 3**

# Swagger

- With the Swagger we can **simplifies** API development for users, teams, and enterprises
- Why we need documentations:
  - front-end and back-end components often **separate** a web application
  - usually, we expose APIs as a back-end component for the front-end component
- Reference documentation should **simultaneously describe** every change in the API



- Spring Boot, using the **SpringDoc** implementation of the **Swagger 3 specification**.
- It's enough to add a single **springdoc-openapi-ui** dependency:

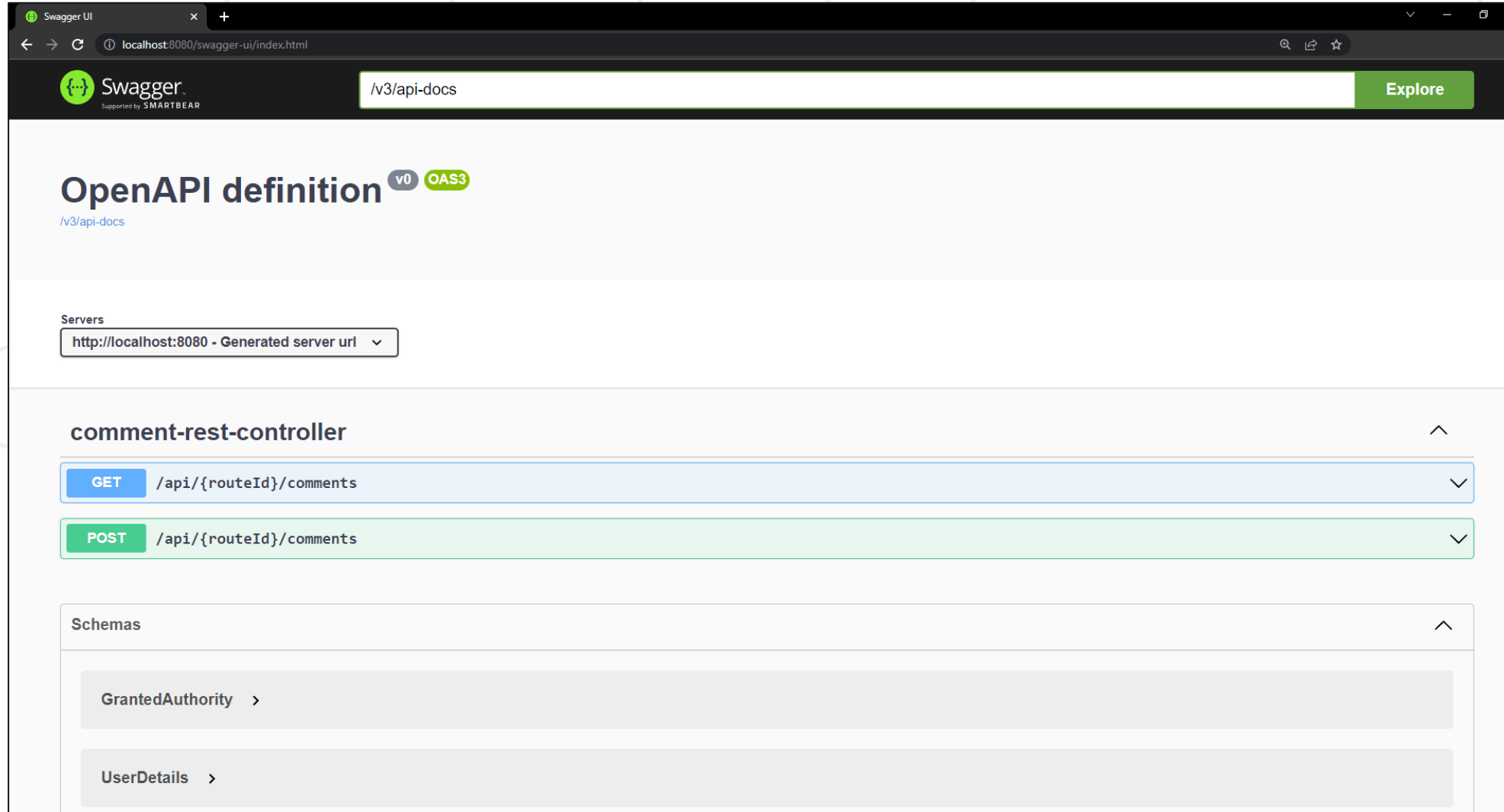
```
<dependency>  
  <groupId>org.springdoc</groupId>  
  <artifactId>springdoc-openapi-ui</artifactId>  
  <version>1.6.8</version>  
</dependency>
```

```
dependencies {  
    implementation 'org.springdoc:springdoc-openapi-ui'  
}
```

- Go to <http://localhost:8080/swagger-ui/index.html> to test it.



# Using Swagger UI example



The screenshot displays the Swagger UI interface in a web browser. The address bar shows the URL `localhost:8080/swagger-ui/index.html`. The Swagger logo and "Supported by SMARTBEAR" are in the top left. A search bar contains `/v3/api-docs` and an "Explore" button is on the right. The main heading is "OpenAPI definition" with version tags `v0` and `OAS3`. Below this, the "Servers" section shows a dropdown menu with the selected value `http://localhost:8080 - Generated server url`. The "comment-rest-controller" section is expanded, showing two endpoints: a GET endpoint `/api/{routeId}/comments` and a POST endpoint `/api/{routeId}/comments`. The "Schemas" section is also expanded, showing two schema entries: `GrantedAuthority` and `UserDetails`, each with a right-pointing arrow.

- **Swagger UI** allows anyone to **visualize** and **interact** with the API's resources without having any of the implementation logic in place.
- It's **automatically generated** from your OpenAPI (formerly known as Swagger) Specification, with the visual documentation making it easy for back-end implementation and client-side consumption.

- Containerization is a hot topic, but it isn't something new
- Docker is de-facto a standard
- Images can be published to private or public registries
- Using Swagger 3 for easy documenting our application



Docker Documentation

<https://docs.docker.com/>

Docker Hub Documentation

<https://docs.docker.com/docker-hub/>

Docker Registry Documentation

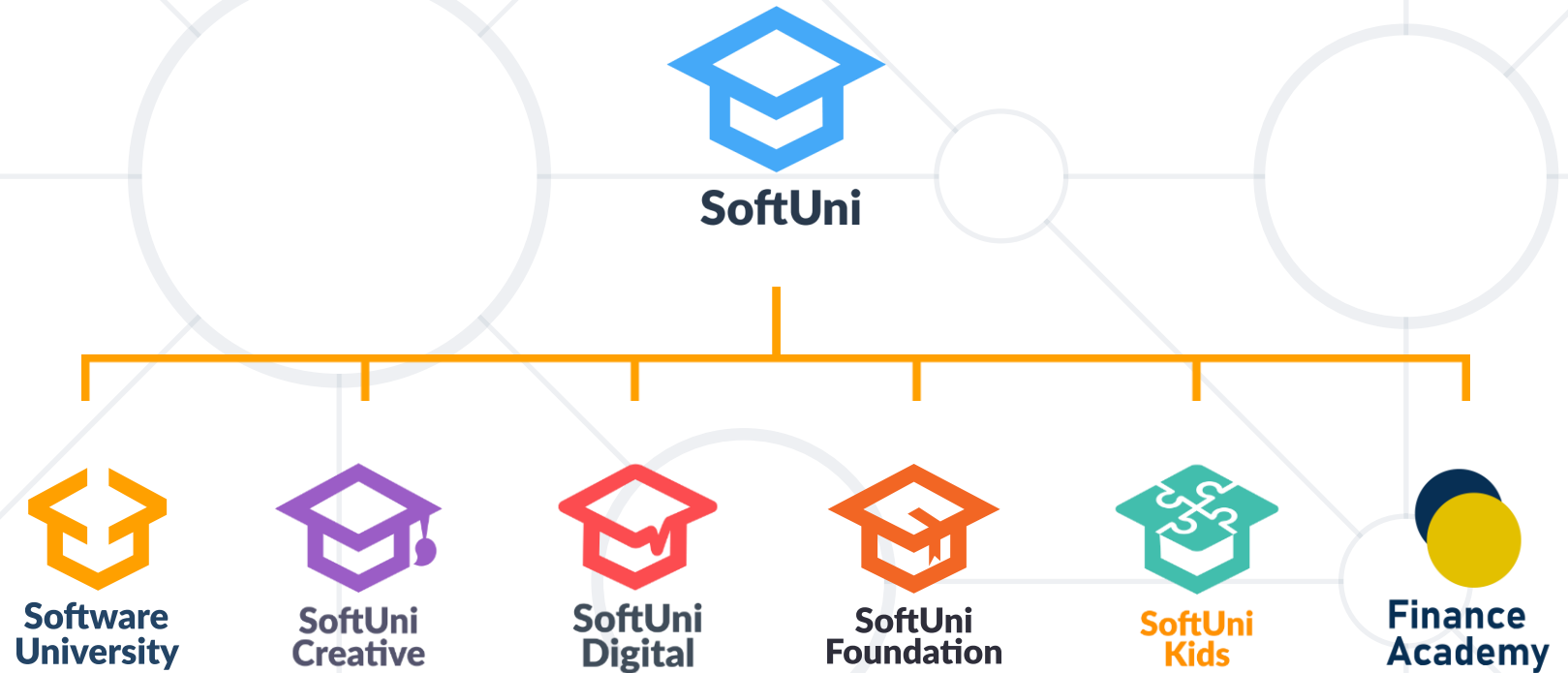
<https://docs.docker.com/registry/>

Swagger Documentation

<https://swagger.io/>



# Questions?



# SoftUni Diamond Partners

**SUPER  
HOSTING  
.BG**



**Coca-Cola HBC  
Bulgaria**

 **Flutter**<sup>TM</sup>  
International

**INDEAVR**  
Serving the high achievers



**AMBITIONED**

 **DRAFT  
KINGS**

 **SOFTWARE  
GROUP**



**BOSCH**

 **Postbank**  
*Решения за твоето утре*

 **PHAR  
VISION**



**SmartIT**

**DXC**  
TECHNOLOGY

**createX**

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>



- Software University – High-Quality Education, Profession and Job for Software Developers

- [softuni.bg](http://softuni.bg), [about.softuni.bg](http://about.softuni.bg)

- Software University Foundation

- [softuni.foundation](http://softuni.foundation)

- Software University @ Facebook

- [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)

- Software University Forums

- [forum.softuni.bg](http://forum.softuni.bg)

