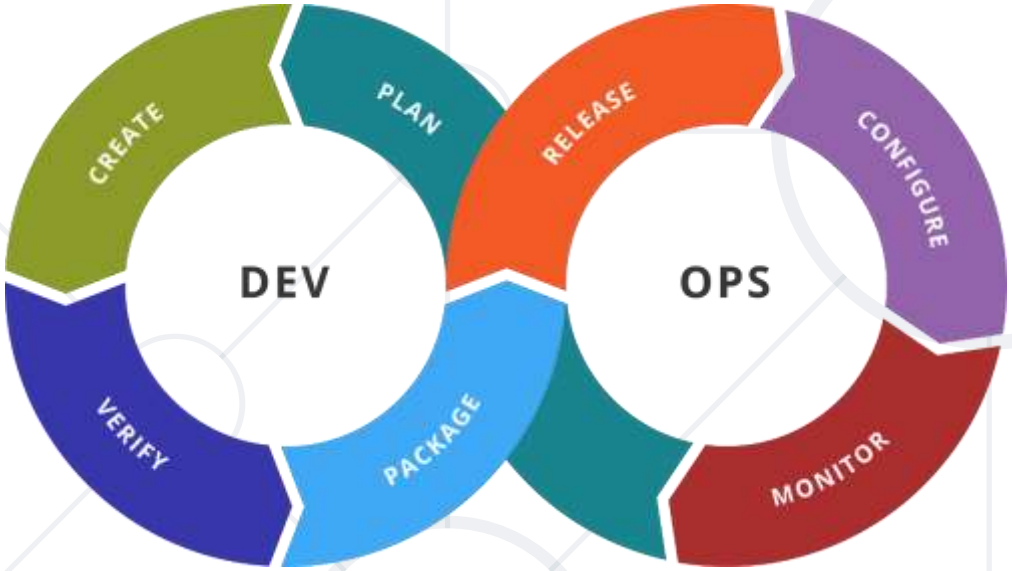
Introduction to Docker

Install Docker. Work with Images & Containers



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

You Have Questions?

Software University

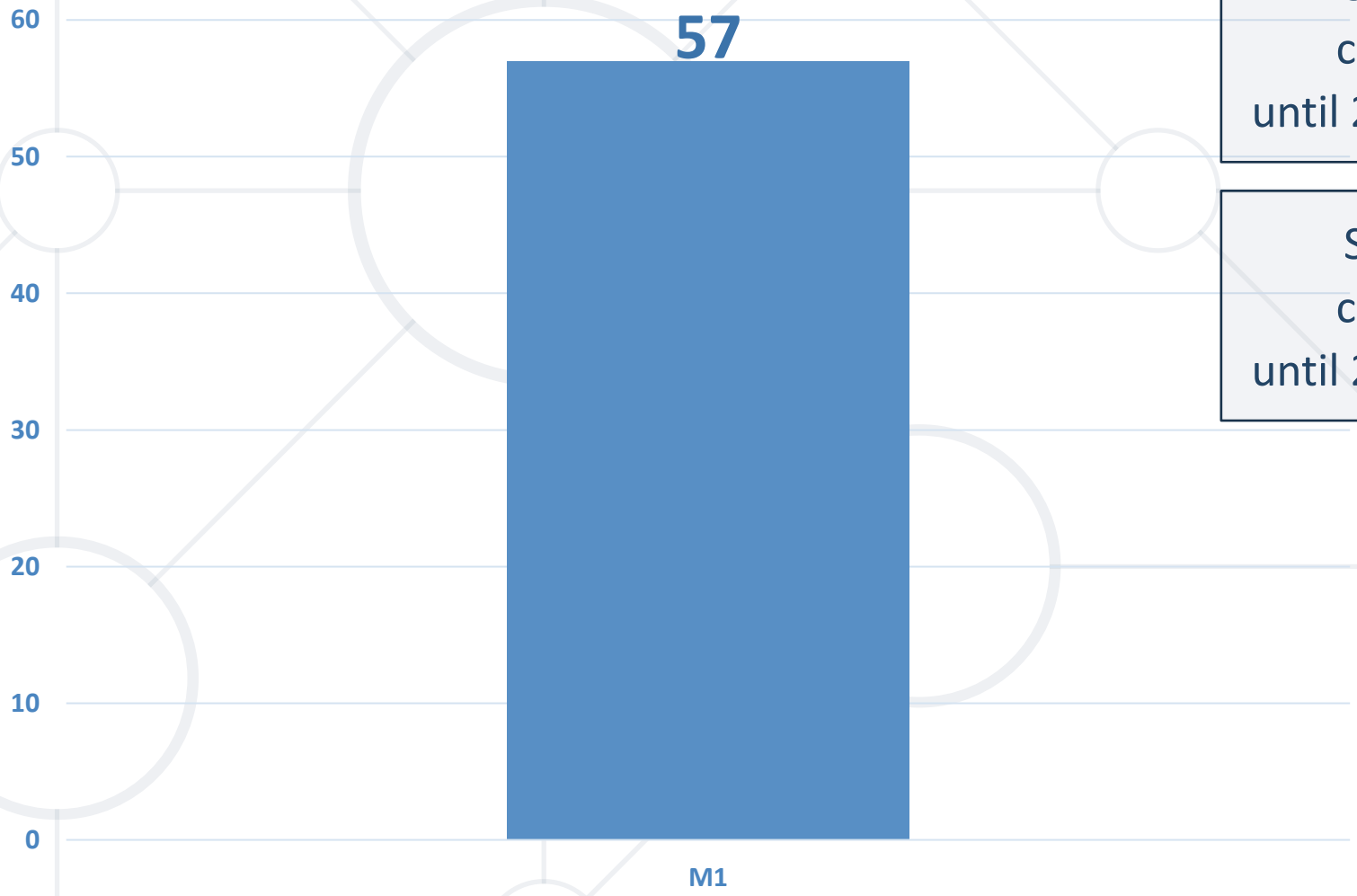
sli.do

#DevOps-23

facebook.com

/groups/DevOpsContainerizationCICDMonitoringJanuary2023

Homework Progress



Solutions for M1
can be submitted
until 23:59 on 06.02.2023

Solutions for M2
can be submitted
until 23:59 on 13.02.2023

Previous Module (M1)

Quick Overview

What We Covered

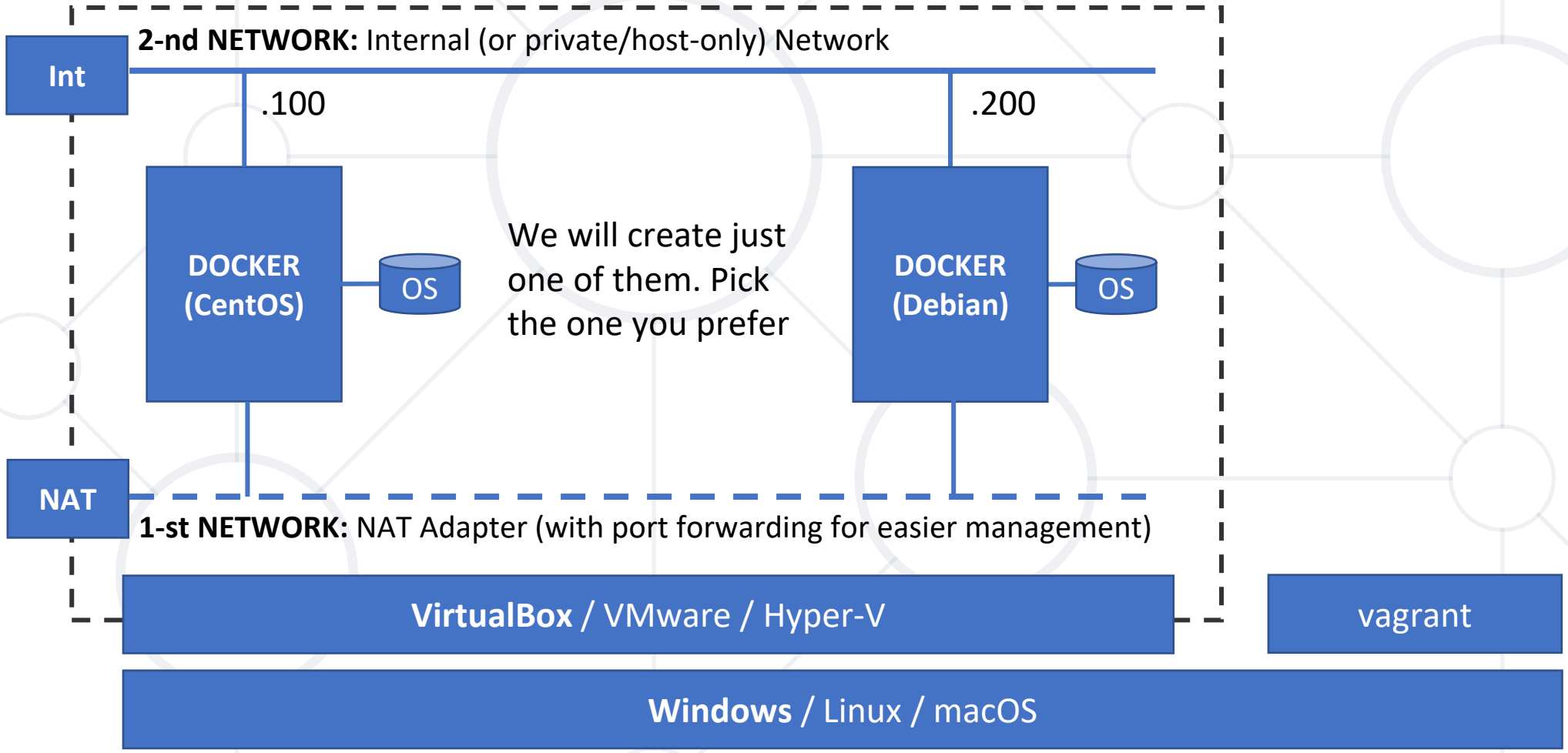
1. The Big Picture
   - Main Pain Points and Causes
   - Goals and Benefits
   - Adoption and Tools
2. Basic Toolkit
3. Basic Automation

This Module (M2)

Topics and Lab Infrastructure

Table of Contents

Lab Infrastructure



Containers and Docker

Past. Present. Future

Containerization

Software University

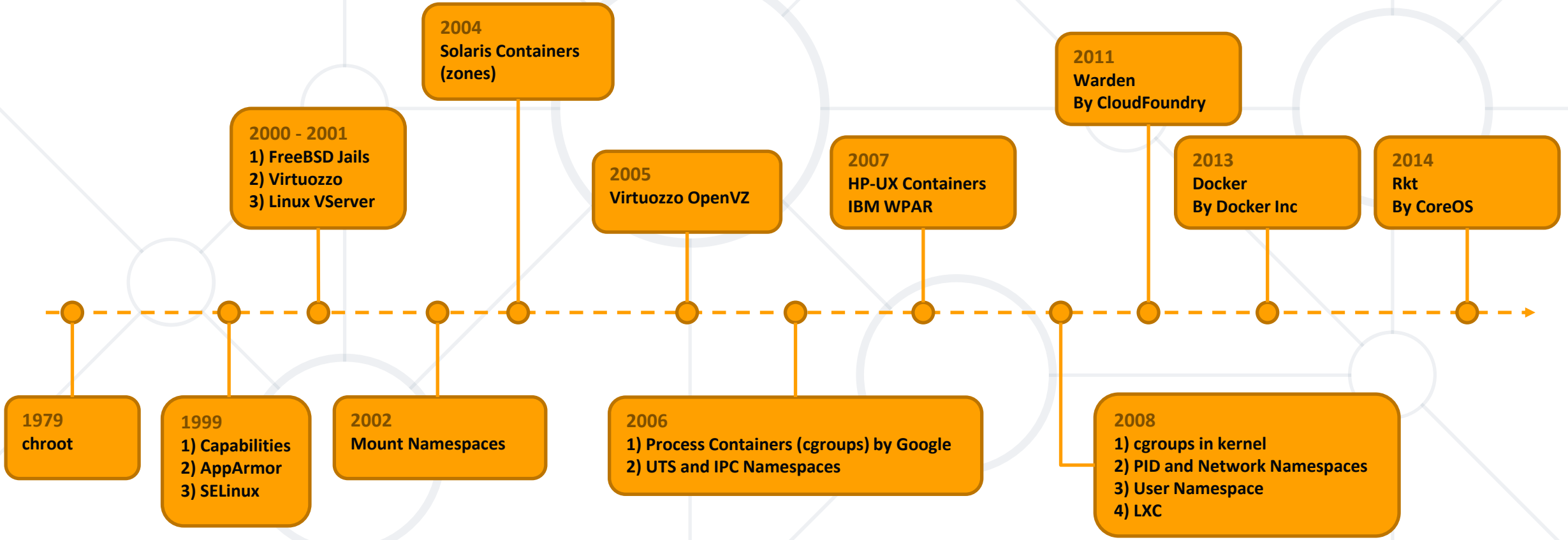
> OS-level virtualization refers to an operating system paradigm in which the kernel allows the existence of **multiple isolated user space instances** known as **containers, zones, jails, …**

https://en.wikipedia.org/wiki/OS-level_virtualization

Road to Containers

Software University

Companies and Solutions



Enablement Technologies

Container Types and Solutions

Software University

- **System Containers (BSD Jails, Solaris Zones, …)**
  - **LXC + LXD + LXCF** by **Canonical**
  - Container hypervisor (system containers)
  - https://linuxcontainers.org/

OS-centric
Multiple processes

- **Application Containers (containerd, CRI-O, …)**
  - **Docker** by **Docker Inc**
  - Tools and application container engine
  - https://www.docker.com/

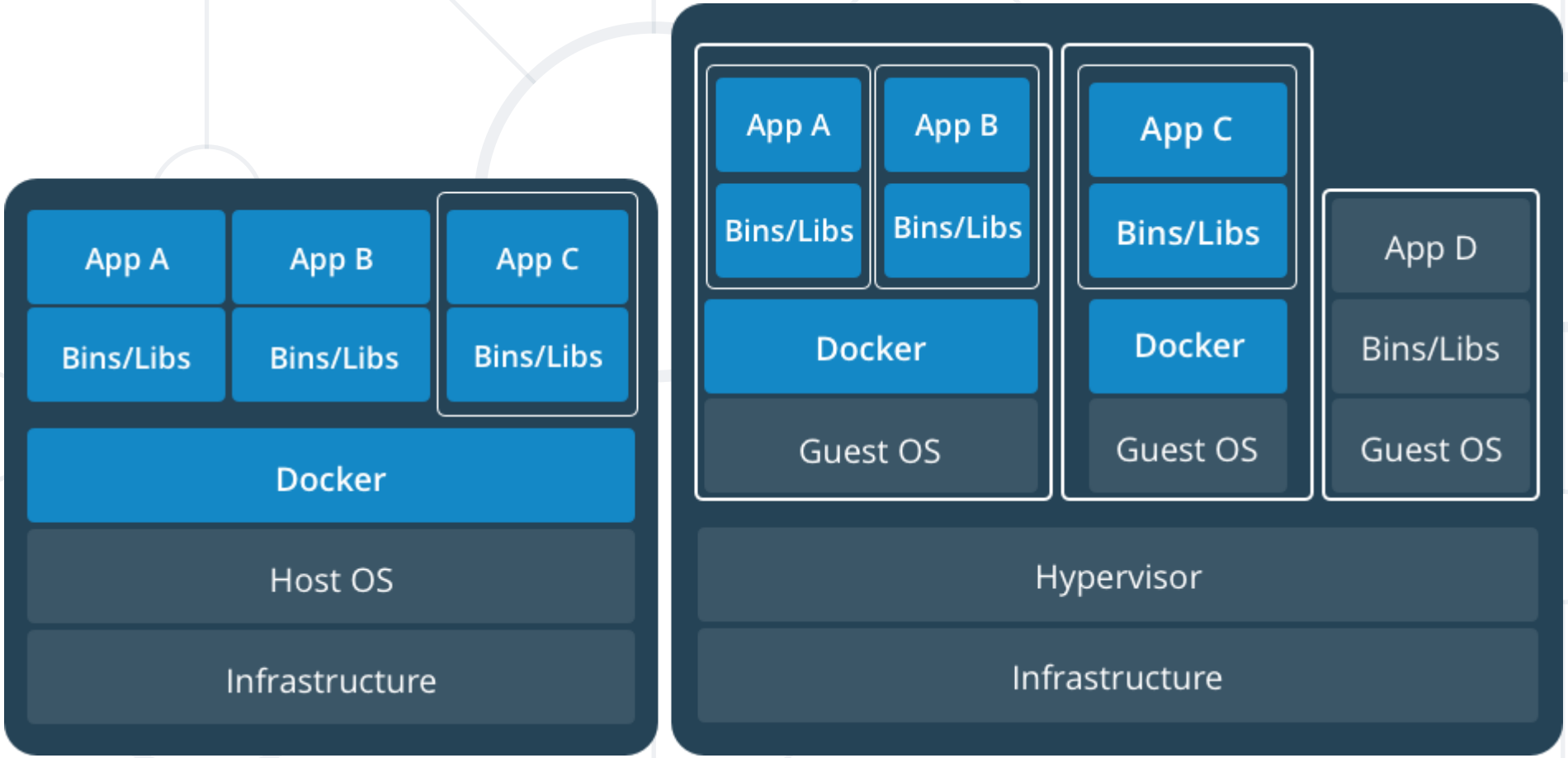
App-centric
Single process *

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

Together: VMs and Containers



https://www.docker.com/what-container

Docker

Whole New World

Containers Concepts (Docker View)

Software University

- **Container host** is a physical or virtual computer system configured with **a container engine**
- **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

Definitions

- **Container**
  - A runnable instance of an image. Containers are processes with much more isolation
- **Image**
  - A read-only template of a container built from layers. Images provide a way for simpler software distribution
- **Repository**
  - A collection of different versions of an image identified by tags
- **Registry**
  - A collection of repositories

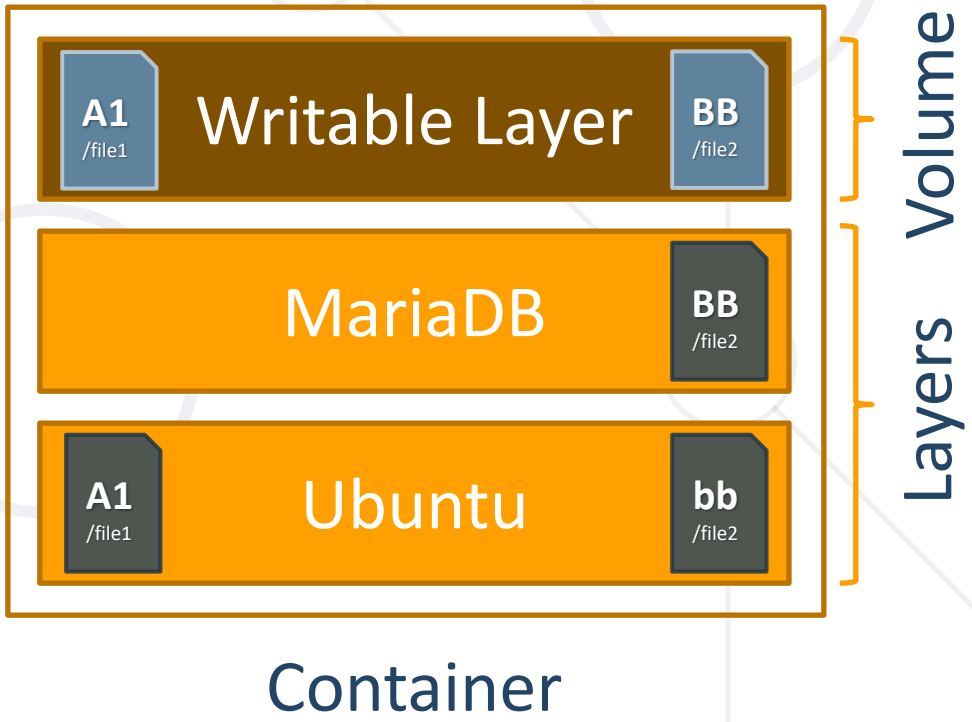
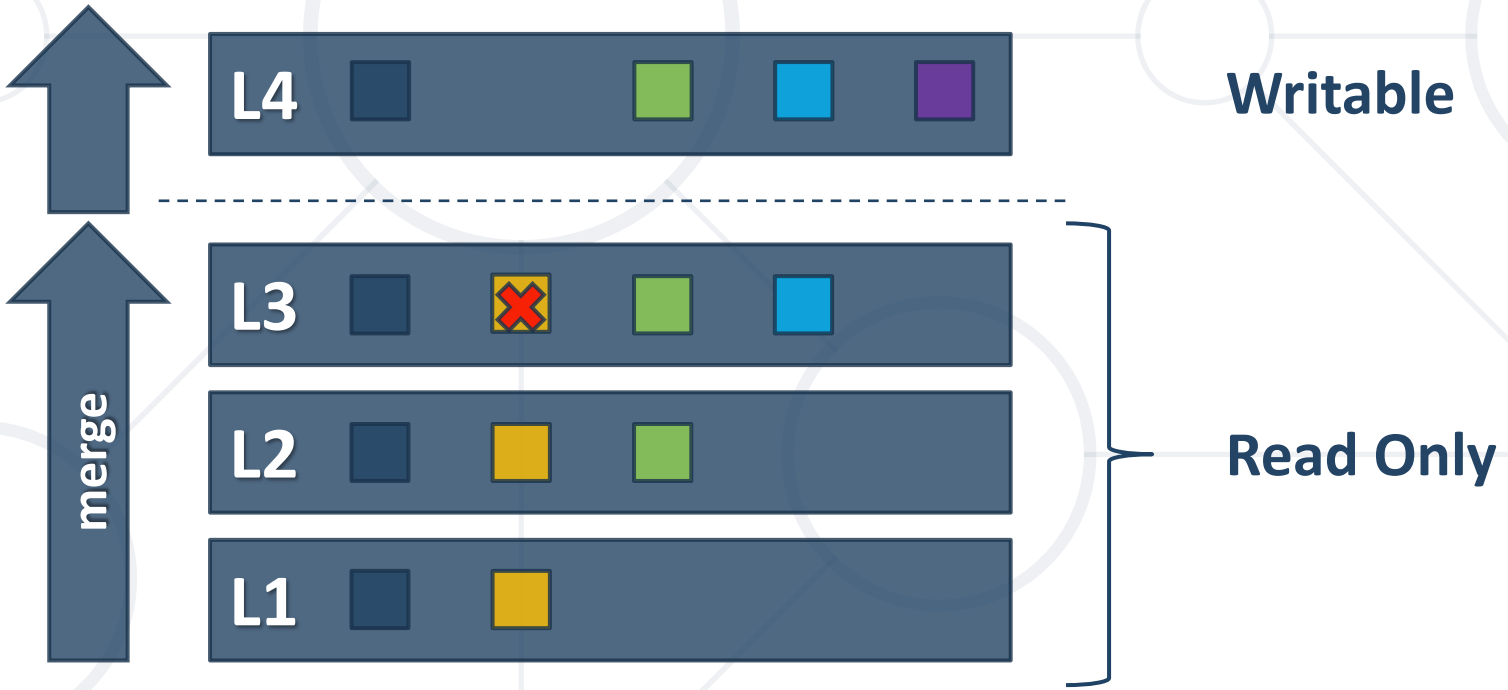
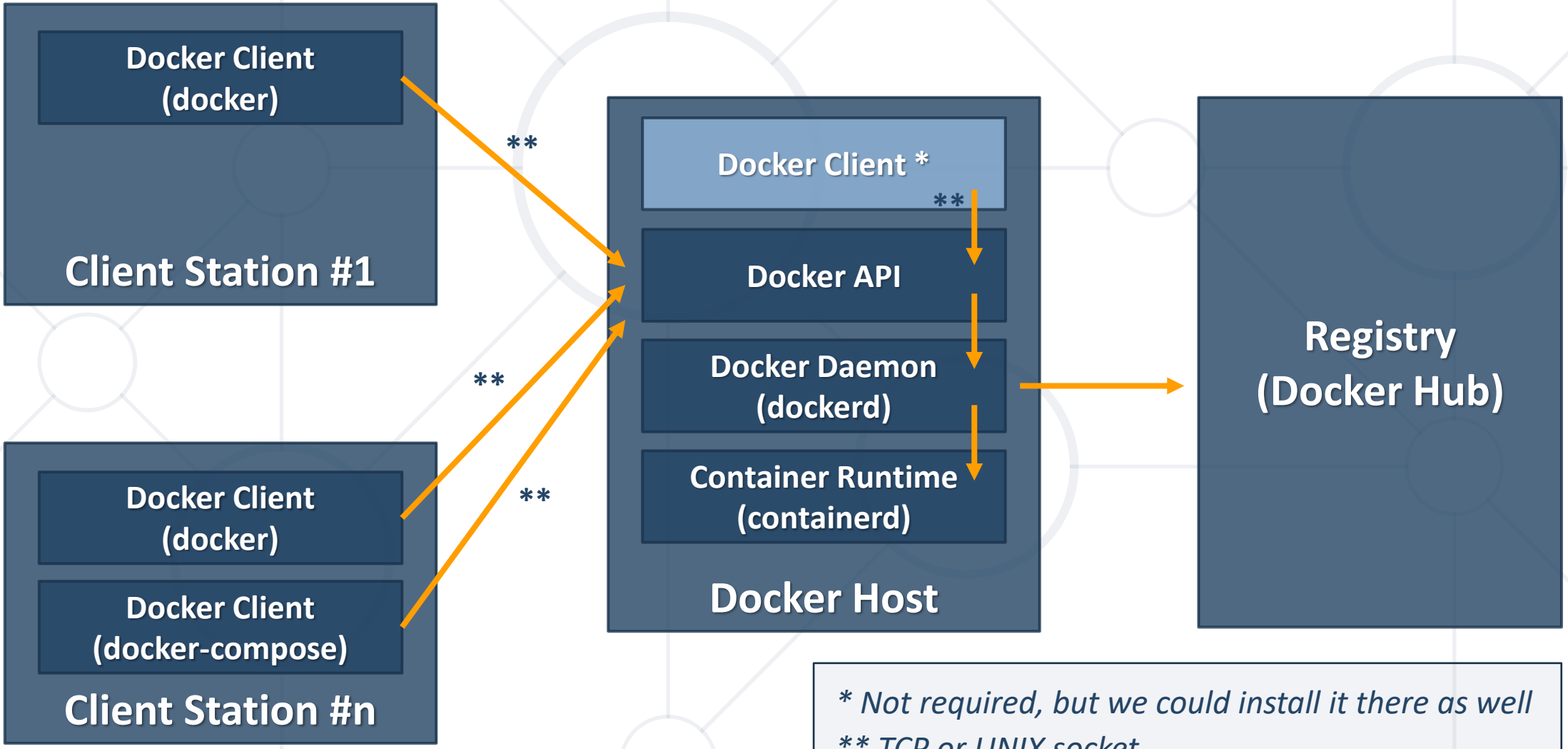


Image Layers



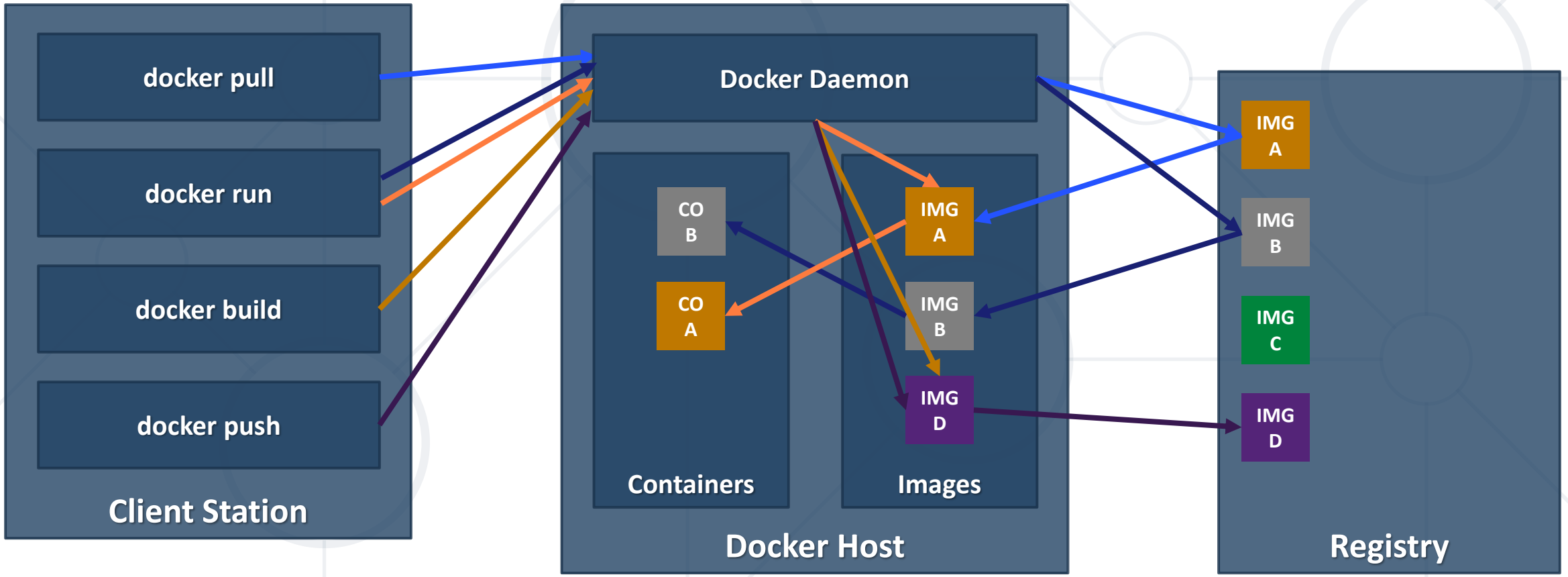
Docker Platform



** Not required, but we could install it there as well*

*** TCP or UNIX socket*

Workflow


docker pull
docker run
docker build
docker push
Client Station
Docker Daemon
CO B
CO A
Containers
IMG A
IMG B
IMG D
Images
Docker Host
IMG A
IMG B
IMG C
IMG D
Registry

Installation Options

- **Docker Desktop**
  - Linux
  - macOS
  - Windows

Specific requirements: OS version, Hypervisor, etc.

- **Docker Engine**
  - Various Linux distributions → Deployment via **package system** (two channels – **stable**, and **test**), **script**, or **archive**
  - Various hardware architectures

x86_64/amd64	arm64 (aarch64)	arm (armhf)	s390x

https://docs.docker.com/engine/install/

Registries

- Provided by Docker
  - Cloud
    - Docker Hub (https://hub.docker.com/explore/)
  - On-premise
    - Standalone
    - Containerized
- Provided by 3rd parties
  - Quay.io, Artifactory, Google Container Registry, etc.

Registries can be **private** or **public**

Repositories can also be **private** or **public**

Practice: Installation & Hello World

Live Demonstration in Class

Working with Docker

Commands

Command Specifics

- Syntax varies amongst versions
  - Old (short) style – still available
  - New style – preferred
- Grouped by target
  - Management Commands
  - General Commands

search

- Purpose
  - Search the Docker Hub for images
- Old (short) syntax

```
docker search [OPTIONS] TERM
```

- New syntax

```
# same
```

- Example *(search for image that have ubuntu in their name)*

```
docker search ubuntu
```

pull / image pull

- Purpose
  - Pull an image or a repository from a registry
- Old (short) syntax

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

- New syntax

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

- Example *(download the ubuntu:latest image locally)*

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

run / container run

- Purpose
  - Run a command in a new container
- Old (short) syntax

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

- New syntax

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

- Example *(run bash in ubuntu based container interactively)*

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

images / image ls

- Purpose
  - List locally available images
- Old (short) syntax

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

- New syntax

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

- Example *(list all tags for the fedora image)*

```
docker image ls fedora
```

ps / container ls

- Purpose
  - List containers
- Old (short) syntax

```
docker ps [OPTIONS]
```

- New syntax

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

- Example *(return running and stopped container IDs)*

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

rm / container rm

- Purpose
  - Remove one or more containers
- Old (short) syntax

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

- New syntax

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

- Example *(remove container by its name)*

```
docker container rm weezy_snake
```

rmi / image rm

- Purpose
  - Remove one or more images
- Old (short) syntax

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

- New syntax

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

- Example *(remove the ubuntu and fedora images)*

```
docker image rm ubuntu fedora
```

create / container create

- Purpose
  - Create a new container
- Old (short) syntax

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

- New syntax

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

- Example *(create a container without starting it)*

```
docker container create -t -i fedora bash
```

rename / container rename

- Purpose
  - Rename a container
- Old (short) syntax

```
docker rename CONTAINER NEW_NAME
```

- New syntax

```
docker container rename CONTAINER NEW_NAME
```

- Example *(change container name from cont1 to newcont1)*

```
docker container rename cont1 newcont1
```

kill / container kill

- Purpose
  - Kill one or more running containers
- Old (short) syntax

```
docker kill [OPTIONS] CONTAINER [CONTAINER]
```

- New syntax

```
docker container kill [OPTIONS] CONTAINER [CONTAINER]
```

- Example *(stop a container by its ID)*

```
docker container kill 0cbf27183
```

start / container start

- Purpose
  - Start one or more stopped containers
- Old (short) syntax

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

- New syntax

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

- Example *(start a container by its ID and attach to it)*

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

restart / container restart

- Purpose
  - Restart a one or more containers
- Old (short) syntax

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

- New syntax

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

- Example *(restart a container by its ID)*

```
docker container restart 0cbf27183
```

stop / container stop

- Purpose
  - Stop one or more running containers
- Old (short) syntax

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

- New syntax

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

- Example *(stop a container by its ID)*

```
docker container stop 0cbf27183
```

pause / container pause

- Purpose
  - Pause all processes within one or more containers
- Old (short) syntax

```
docker pause CONTAINER [CONTAINER]
```

- New syntax

```
docker container pause CONTAINER [CONTAINER]
```

- Example *(pause a container by its ID)*

```
docker container pause 0cbf27183
```

unpause / container unpause

- Purpose
  - Resume all processes within one or more containers
- Old (short) syntax

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

- New syntax

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

- Example *(resume a container by its ID)*

```
docker container unpause 0cbf27183
```

attach / container attach

- Purpose
  - Attach to a running container
- Old (short) syntax

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

- New syntax

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

- Example *(attach to the process in a container by its ID)*

```
docker container attach 0cbf27183
```

tag / image tag

- Purpose
  - Tag an image into a repository
- Old (short) syntax

```
docker tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]
```

- New syntax

```
docker image tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]
```

- Example *(create a new tag for an existing image)*

```
docker image tag test:1.0 repo-name/test:latest
```

push / image push

- Purpose
  - Push an image or repository to a registry
- Old (short) syntax

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

- New syntax

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

- Example *(publish a local image to a remote registry)*

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

login

- Purpose
  - Log into a Docker registry
- Old (short) syntax

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

- New syntax

```
# same
```

- Example *(authenticate against a custom registry)*

```
docker login localrepo:5000
```

logout

- Purpose
  - Log out from a Docker registry
- Old (short) syntax

```
docker logout [OPTIONS] [SERVER]
```

- New syntax

```
# same
```

- Example *(log out from a custom registry)*

```
docker logout localrepo:5000
```

export / container export

- Purpose
  - Export a container's filesystem as a tar archive
- Old (short) syntax

```
docker export [OPTIONS] CONTAINER
```

- New syntax

```
docker container export [OPTIONS] CONTAINER
```

- Example *(export container's filesystem as a file)*

```
docker container export -o file.tar test
```

import / image import

- Purpose
  - Import the contents from a tar to create a filesystem image
- Old (short) syntax

```
docker import [OPTIONS] file|URL|- [REPOSITORY[:TAG]]
```

- New syntax

```
docker image import [OPTIONS] file|URL|- [REPOSITORY[:TAG]]
```

- Example *(import the file.tar as a new-test container image)*

```
docker image import file.tar new-test
```

save / image save

- Purpose
  - Save one or more images to a tar archive or STDOUT
- Old (short) syntax

```
docker save [OPTIONS] IMAGE [IMAGE]
```

- New syntax

```
docker image save [OPTIONS] IMAGE [IMAGE]
```

- Example *(export the busybox image as a file)*

```
docker image save -o busybox.tar busybox
```

load / image load

- Purpose
  - Load an image from a tar archive or STDIN
- Old (short) syntax

```
docker load [OPTIONS]
```

- New syntax

```
docker image load [OPTIONS]
```

- Example *(import an image from a tar archive file)*

```
docker image load -i busybox.tar
```

Practice: Working with Containers

Live Demonstration in Class

Image from Container

Introduction

- **Commit** a container's file changes to a new image
- Use **running** or **stopped** container
- If running, all processes are **paused**
- Mounted volumes' data is **not included**
- It is advisable to use **Dockerfile** instead

commit / container commit

- Purpose
  - Create a new image from a container's changes
- Old (short) syntax

```
docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]
```

- New syntax

```
docker container commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]
```

- Example *(commit a container by its ID and create an image)*

```
docker container commit 0cf23a31 new-cont
```

Image from File

General Structure and Common Fields

General Structure (Dockerfile)

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

Comment

Command (Instruction)

```
# Set the base image
FROM nginx

# Set the maintainer
MAINTAINER John Smith

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

https://docs.docker.com/engine/reference/builder/

FROM

- 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
```

https://docs.docker.com/engine/reference/builder/#from

MAINTAINER

- 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"
```

https://docs.docker.com/engine/reference/builder/#maintainer-deprecated

LABEL

- Purpose
  - Adds metadata to the image
- Syntax

```
LABEL <key>=<value> <key>=<value> <key>=<value> ...
```

- Example

```
# single key-value pair
LABEL maintainer="John Smith“
# multiple key-value pairs
LABEL maintainer="John Smith" version="1.0"
```

https://docs.docker.com/engine/reference/builder/#label

RUN

- 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
```

https://docs.docker.com/engine/reference/builder/#run

COPY

- Purpose
  - Copy files between the host and the container (adds a layer)
- 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
```

https://docs.docker.com/engine/reference/builder/#copy

ADD

- Purpose
  - Copy files to the image (adds a layer)
- 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
```

https://docs.docker.com/engine/reference/builder/#add

EXPOSE

- 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
```

https://docs.docker.com/engine/reference/builder/#expose

ENTRYPOINT

- 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
```

https://docs.docker.com/engine/reference/builder/#entrypoint

CMD

- 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
```

https://docs.docker.com/engine/reference/builder/#cmd

CMD vs ENTRYPOINT

Software University

- 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

CMD \ ENTRYPOINT	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	**exec_entry p1_entry exec_cmd p1_cmd**
["p1_cmd", "p2_cmd"]	p1_cmd p2_cmd	/bin/sh -c exec_entry p1_entry	**exec_entry p1_entry p1_cmd p2_cmd**
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**~~

Ignores any **CMD** or **docker run** command line arguments

https://docs.docker.com/engine/reference/builder/#understand-how-cmd-and-entrypoint-interact

build / image build

- 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 .
```

Recommendations

Just a Few

Recommendations

- 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/

Heredoc Container

Why Not?!

Heredoc Container

- Purpose
  - Create a new image on the fly ☺
- Example

```
devops@sulab:~$ docker build -t htop - << EOF
FROM alpine
RUN apk --no-cache add htop
EOF
```

Practice: Creating Images

Live Demonstration in Class

Summary

- Containerization is a hot topic, but it isn't something new
- Docker is de-facto a standard
- Docker is offered in two versions – CE and EE
- There are several installation options
- Images can be published to private or public registries
- Two sets of commands are still available
- There are multiple ways to create an image

Resources

Docker Documentation

https://docs.docker.com/

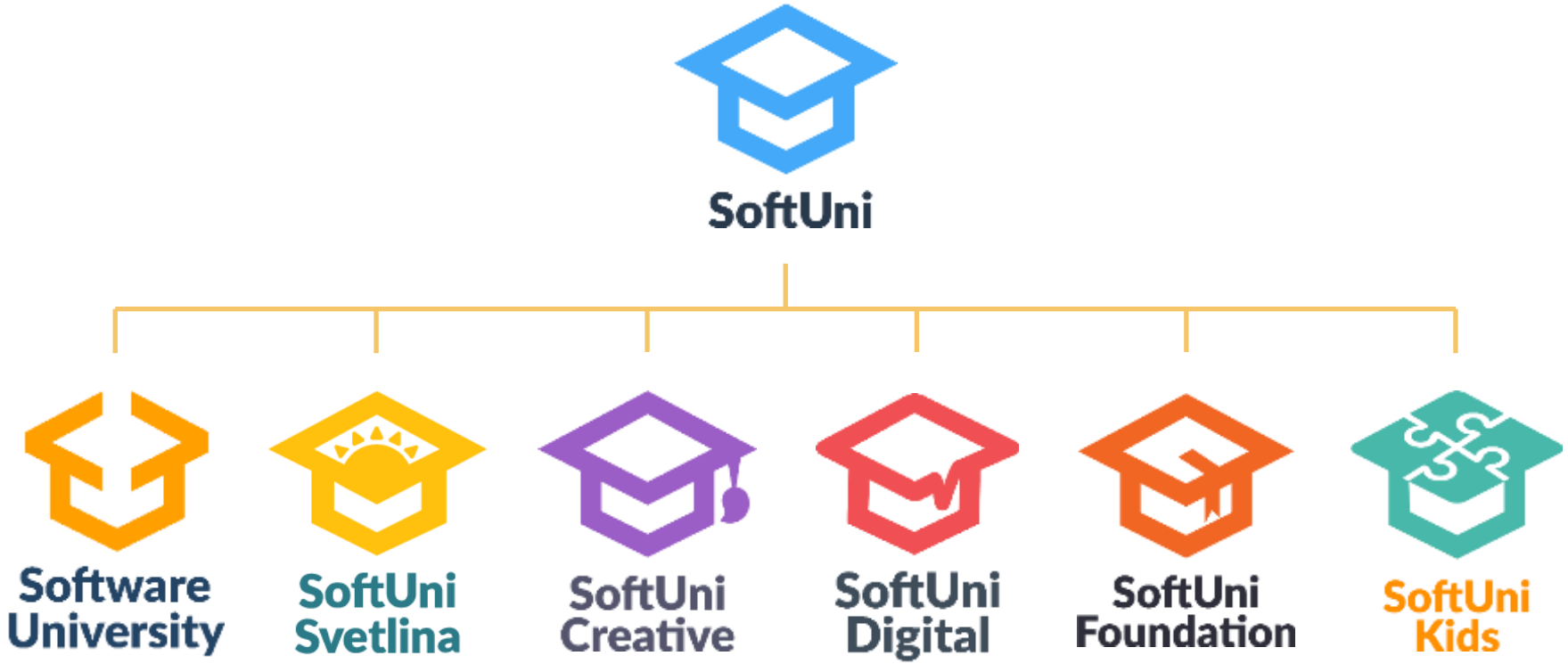
Docker Hub Documentation

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

Docker Registry Documentation

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

Questions?



SoftUni Diamond Partners

Software University

SCHWARZ

LIDL

Kaufland

SCHWARZ Produktion

prezero

Coca-Cola HBC Bulgaria

POKERSTARS | CAREERS

Postbank

Решения за твоето утре

AMBITIONED

DXC TECHNOLOGY

SOFTWARE GROUP

Bosch.IO

INDEAVR

Serving the high achievers

DRAFT KINGS

PHAR VISION

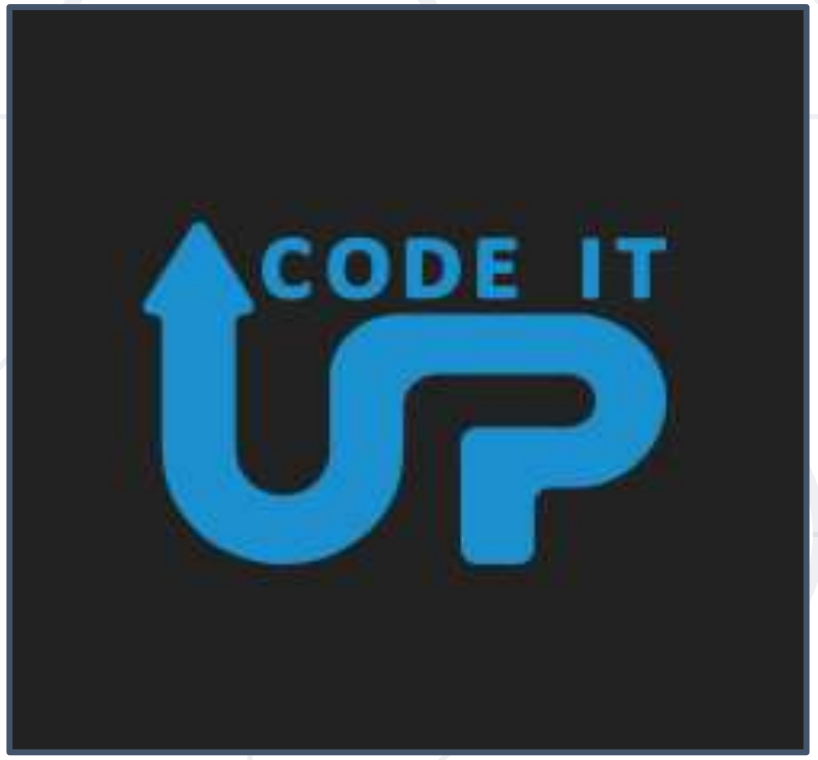
SmartIT

createX

SUPER HOSTING .BG

Educational Partners

Software University



License

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

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Pr ofession and Job for Software Developers
  - softuni.bg, softuni.org
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg