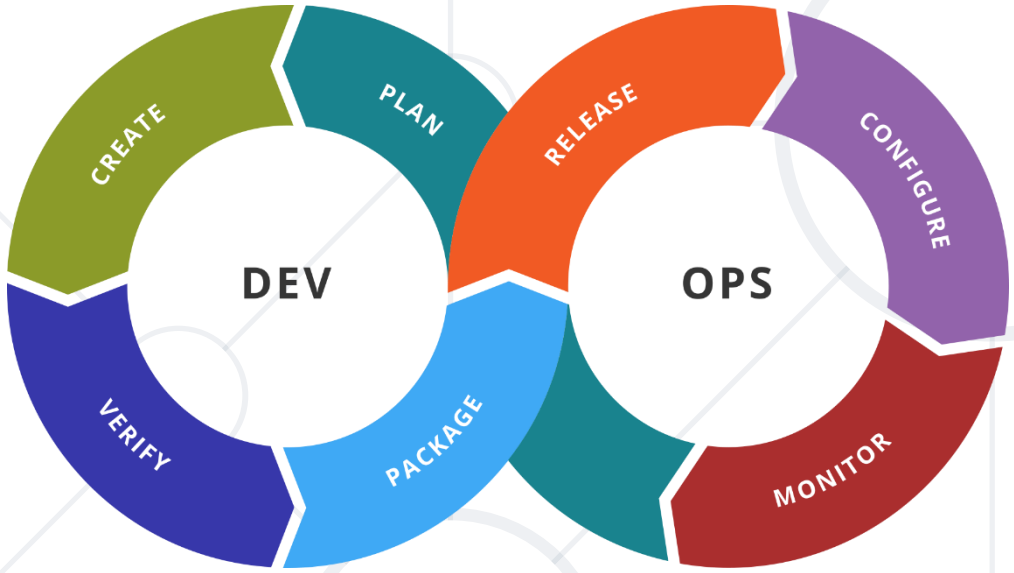
Container Orchestration

Distributed Applications. Clusters. Alternatives



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

You Have Questions?

Software University

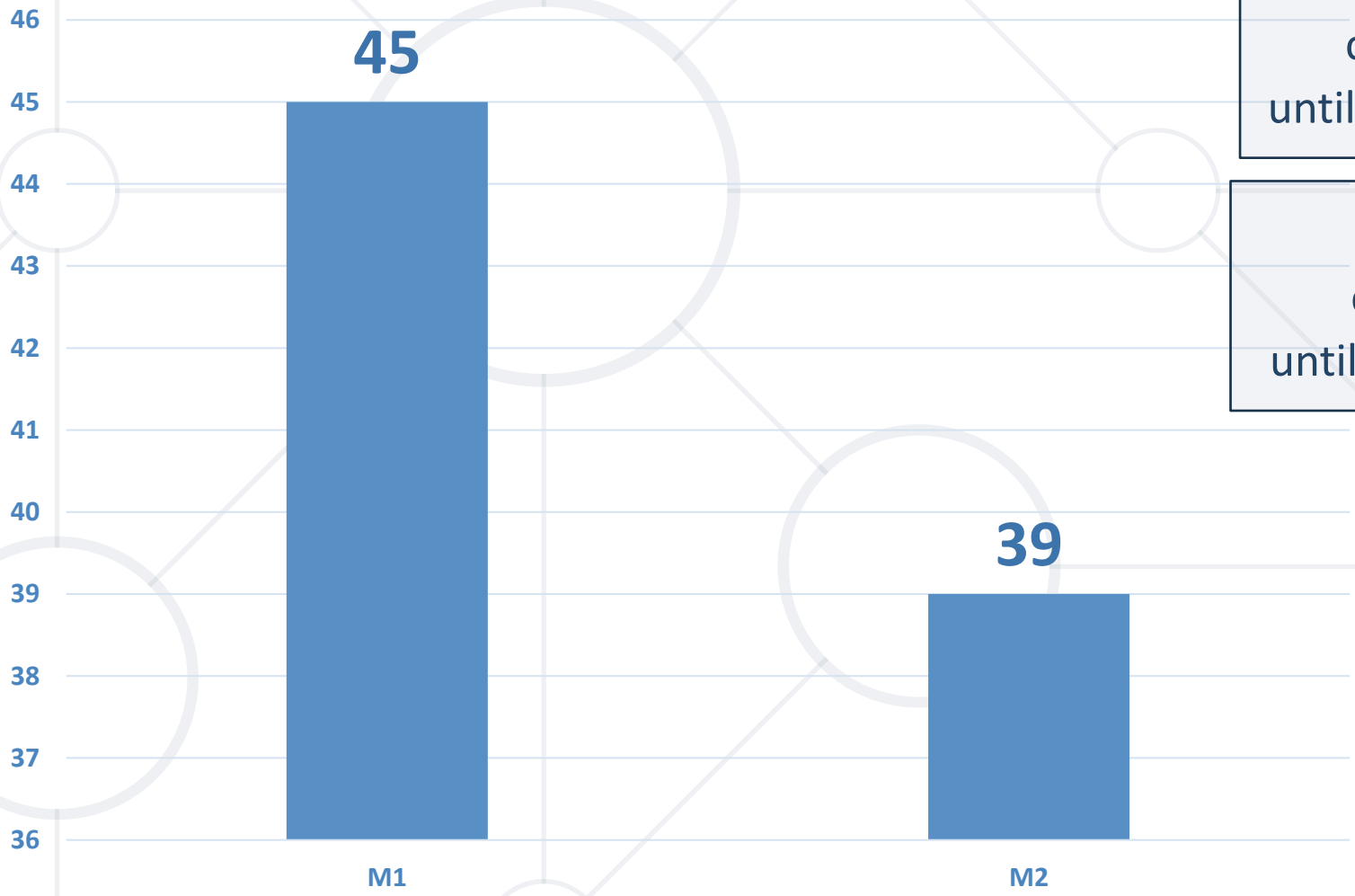
sli.do

#DevOps-CI

facebook.com/groups/

containerizationandinfrastructurejune2025

Homework Progress



Solutions for M2
can be submitted
until 23:59 on 24.06.2025

Solutions for M3
can be submitted
until 23:59 on 01.07.2025

Previous Module (M2)

Quick Overview

What We Covered

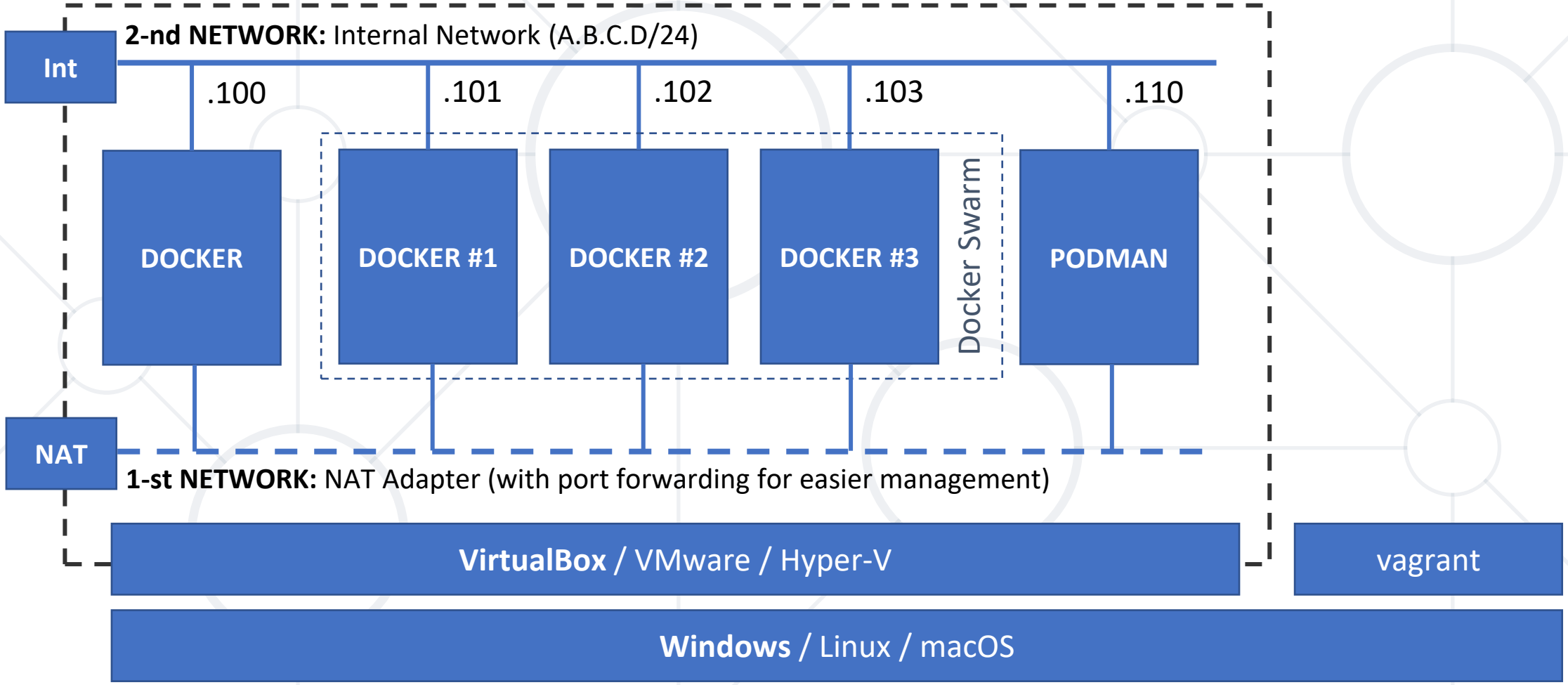
1. Networking and Volumes
2. Custom Container Images
3. Best Practices and Troubleshooting

This Module (M3)

Topics and Lab Infrastructure

Table of Contents

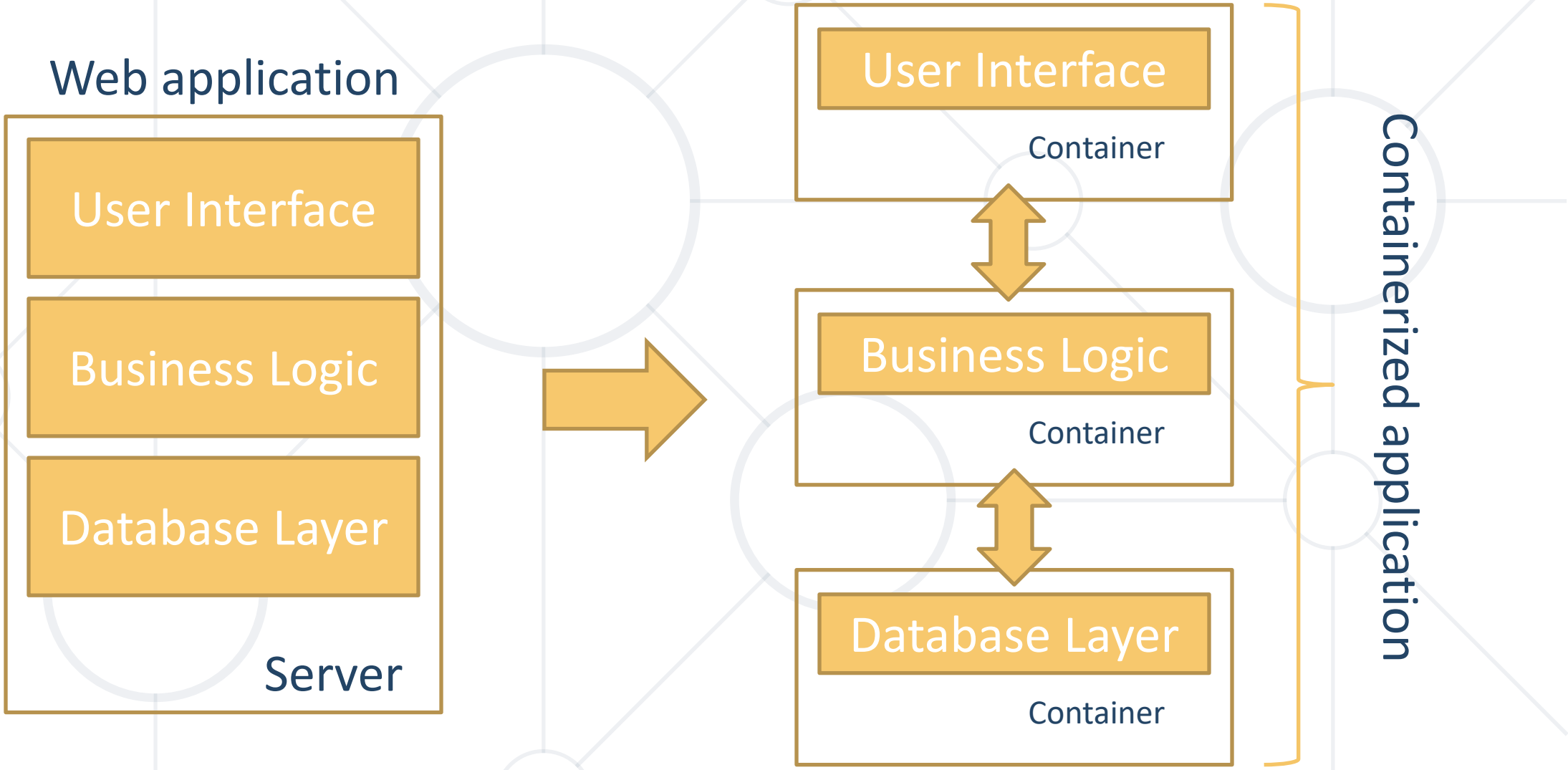
Lab Infrastructure



Distributed Applications

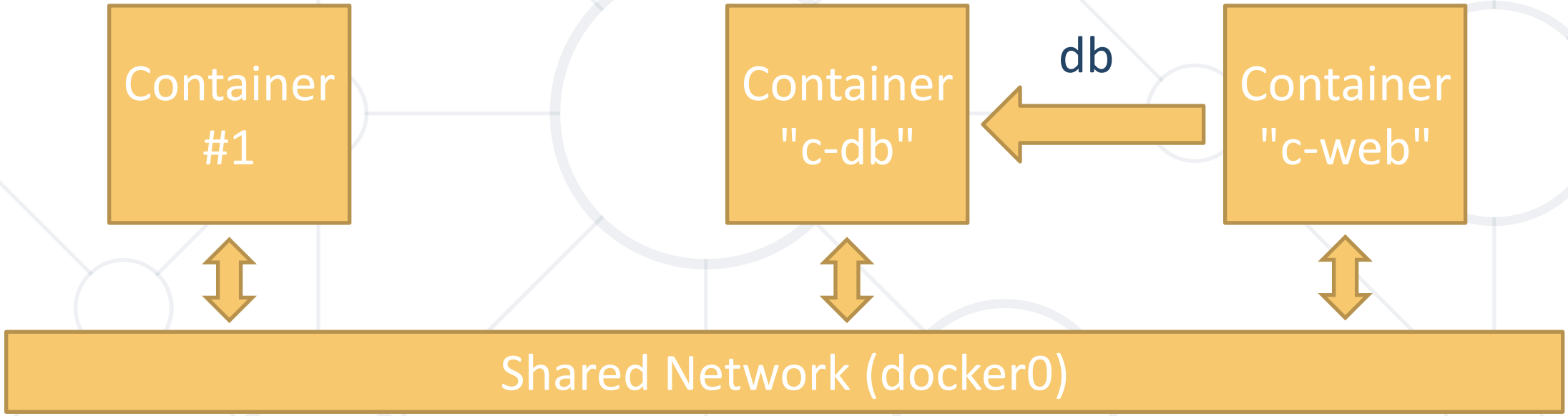
Overview and Implementation

Distributed Applications



Link Containers (Legacy) *

- By name alias



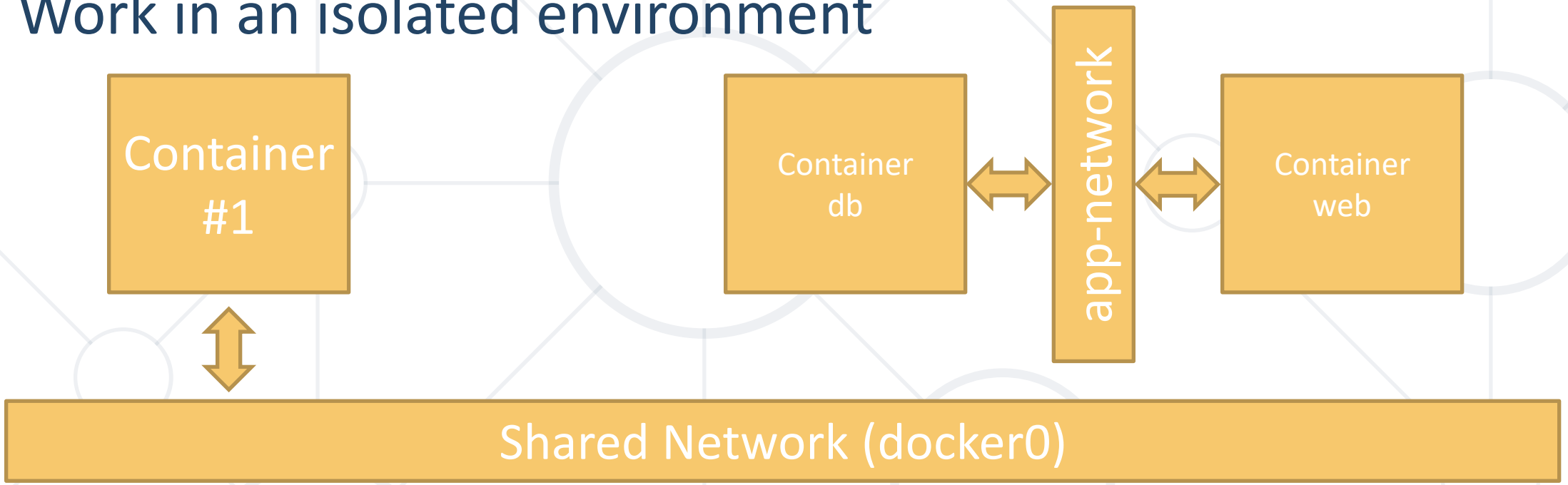
docker container run -d ... -p 8080:80 --link c-db:db ...

Linkage in the form **name:alias**

* Should be avoided as it is legacy and may be removed in future versions

Isolated Network

- Work in an isolated environment



docker container run -d ... -p 8080:80 --net app-network ...

Attached to the isolated network

Docker Compose

Docker Compose

- Define and run **multi-container** Docker applications
- Multiple **isolated environments** on a single host
- **Preserve volume data** when containers are created
- Only recreate containers that **have changes**
- Supports **variables**
- Use cases
  - Development environments
  - Automated testing
  - Single host deployments

Configuration

Software University

Version (up to 3.9)
(optional since v1.27.0)

Services Definition

Networks Definition

```
version: "2.1"
services:
    com-php:
        build: ./web/
        ports:
            - 8080:80
        volumes:
            - "${PROJECT_ROOT}:/var/www/html:ro"
        networks:
            - com-network
networks:
    com-network:
```

docker-compose.yaml

```
PROJECT_ROOT=/home/docker/app

DB_ROOT_PASSWORD=12345
```

.env

docker compose build

- Purpose
  - Build or rebuild services
- Syntax

```
… build [options] [--build-arg key=val...] [SERVICE...]
```

- Example

```
# rebuild all services
docker compose build
# rebuild particular service with no-cache
docker compose build --no-chache my-php
```

docker compose up

- Purpose
  - Build, (re)create, start, and attach to containers for a service
- Syntax

```
… up [options] [--scale SERVICE=NUM...] [SERVICE...]
```

- Example

```
# start all containers and aggregate the output
docker compose up
# start all containers in daemon mode
docker compose up -d
```

docker compose down

- Purpose
  - Stop containers and remove everything created by up
- Syntax

```
docker compose down [options]
```

- Example

```
# remove everything including all images
docker compose down --rmi all
# remove declared named volumes and anonymous volumes
docker compose down --volumes
```

docker compose ps

- Purpose
  - List containers
- Syntax

```
docker compose ps [options] [SERVICE...]
```

- Example

```
# list running containers
docker compose ps
# display ID for a particular container
docker compose ps -q com-php
```

docker compose logs

- Purpose
  - View output from containers
- Syntax

```
docker compose logs [options] [SERVICE...]
```

- Example

```
# view logs for all containers
docker compose logs
# follow the log for com-php service
docker compose logs -f com-php
```

docker compose start

- Purpose
  - Start existing containers
- Syntax

```
docker compose start [SERVICE...]
```

- Example

```
# start all containers
docker compose start
# start particular container / service
docker compose start com-php
```

docker compose stop

- Purpose
  - Stop running containers without removing them
- Syntax

```
docker compose stop [options] [SERVICE...]
```

- Example

```
# stop all containers
docker compose stop
# stop particular container / service with timeout
docker compose stop -t 20 com-php
```

docker compose rm

- Purpose
  - Remove stopped service containers
- Syntax

```
docker compose rm [options] [SERVICE...]
```

- Example

```
# remove all stopped containers
docker compose rm
# stop all containers and remove them without asking
docker compose rm -s -f
```

Practice: Docker Compose

Live Demonstration in Class

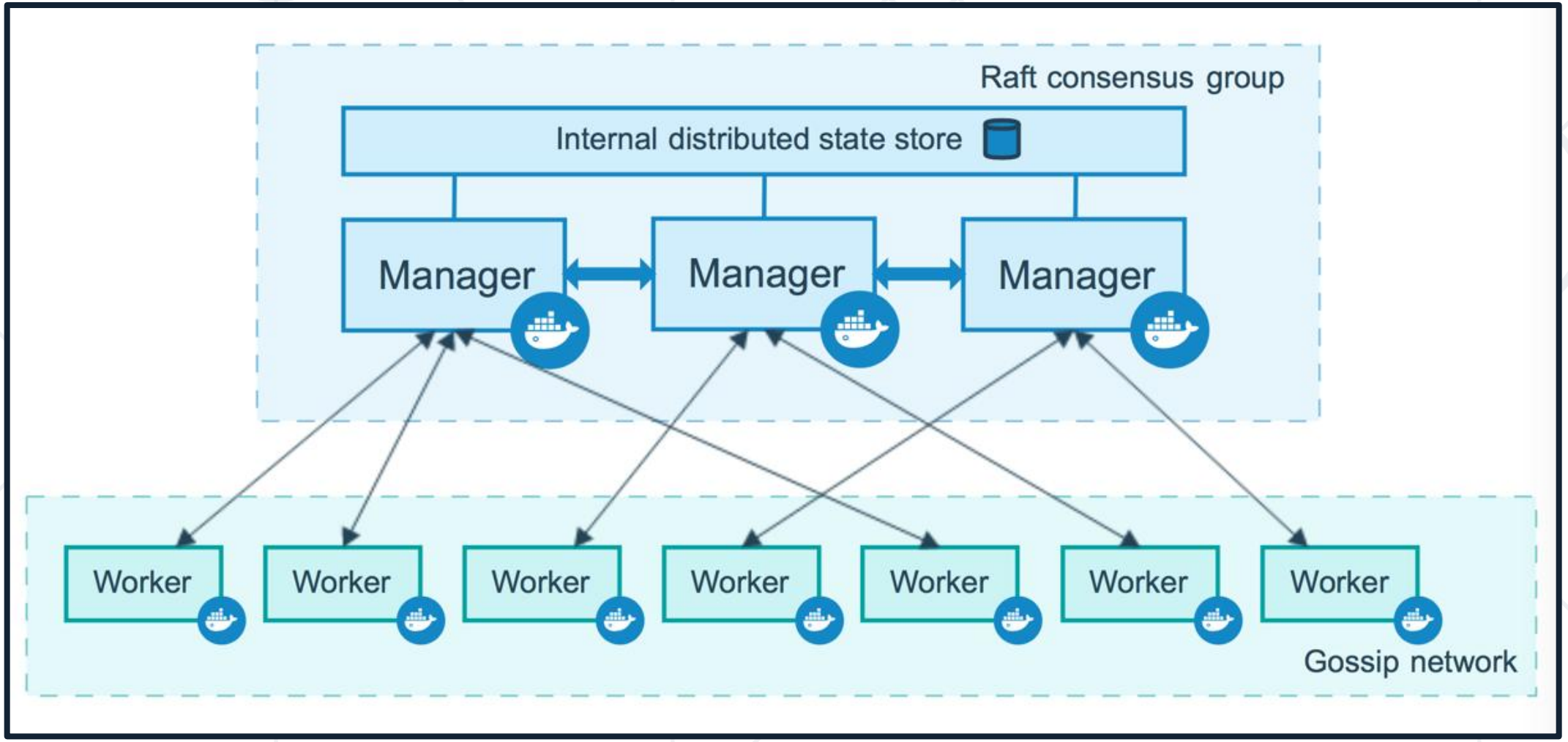
Docker Swarm

What is it? How it works?

What is it?

- Docker **engines joined** in a cluster
- Commands are executed by the **swarm manager**
- There could be more than one manager, but only one is **Leader**
- Nodes that are not managers are called **workers**
- **Both** managers and workers are **running containers**
- There are different **strategies** to run containers
- Nodes can be **physical** or **virtual**

The Big Picture*



* https://docs.docker.com/engine/swarm/how-swarm-mode-works/nodes/

Three Simple Actions

- Initialize cluster
  - `docker swarm init`
- Join to a cluster
  - `docker swarm join`
- Leave a cluster
  - `docker swarm leave`

Deployment Options and Alternatives

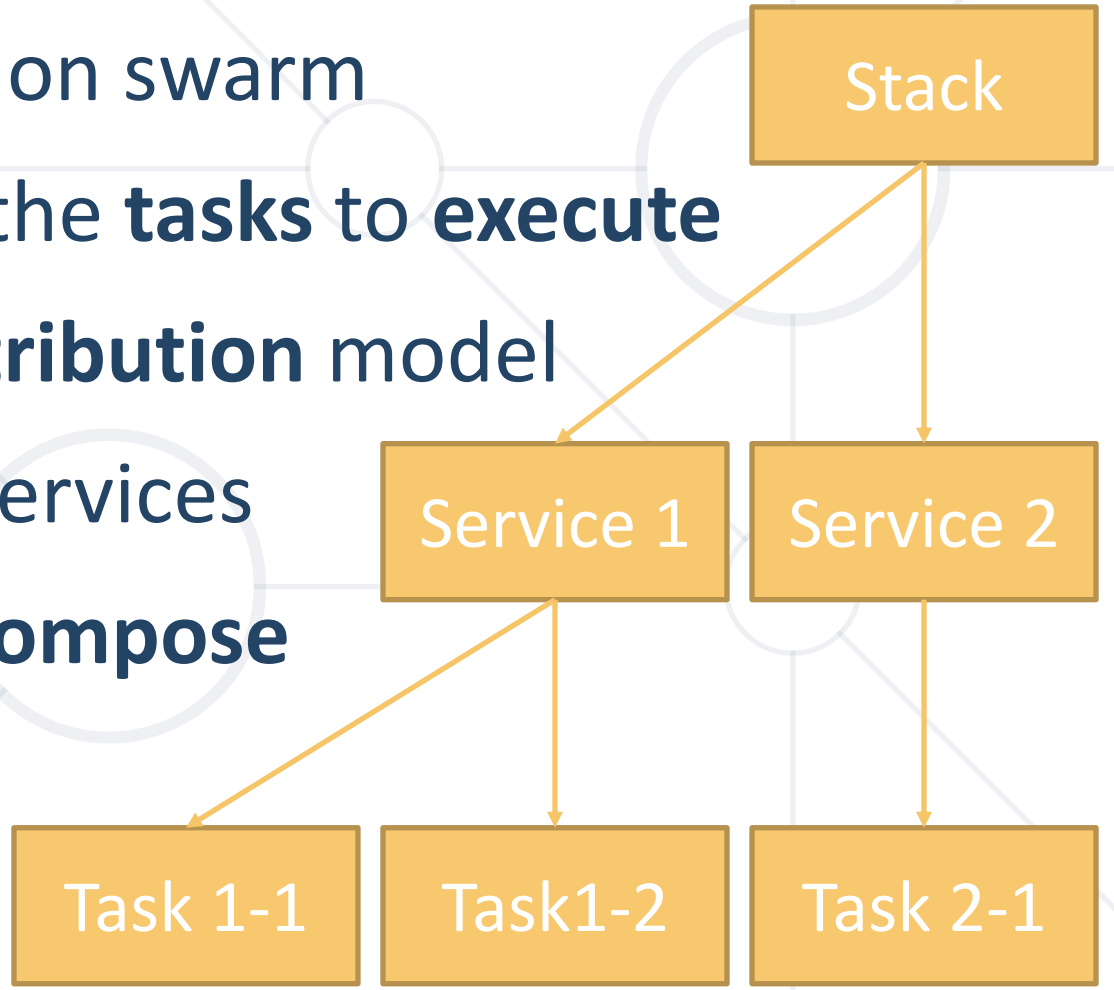
- Deployment Options
  - Cloud (Azure, AWS, …)
  - On-premise - VM, Bare-metal
- Deployment Strategy (on-premise)
  - (Semi) Manual } Today's practice
  - Automated } Additional practice – homework ☺
- Alternative Orchestrators
  - Nomad
  - Kubernetes

Stacks and Compose

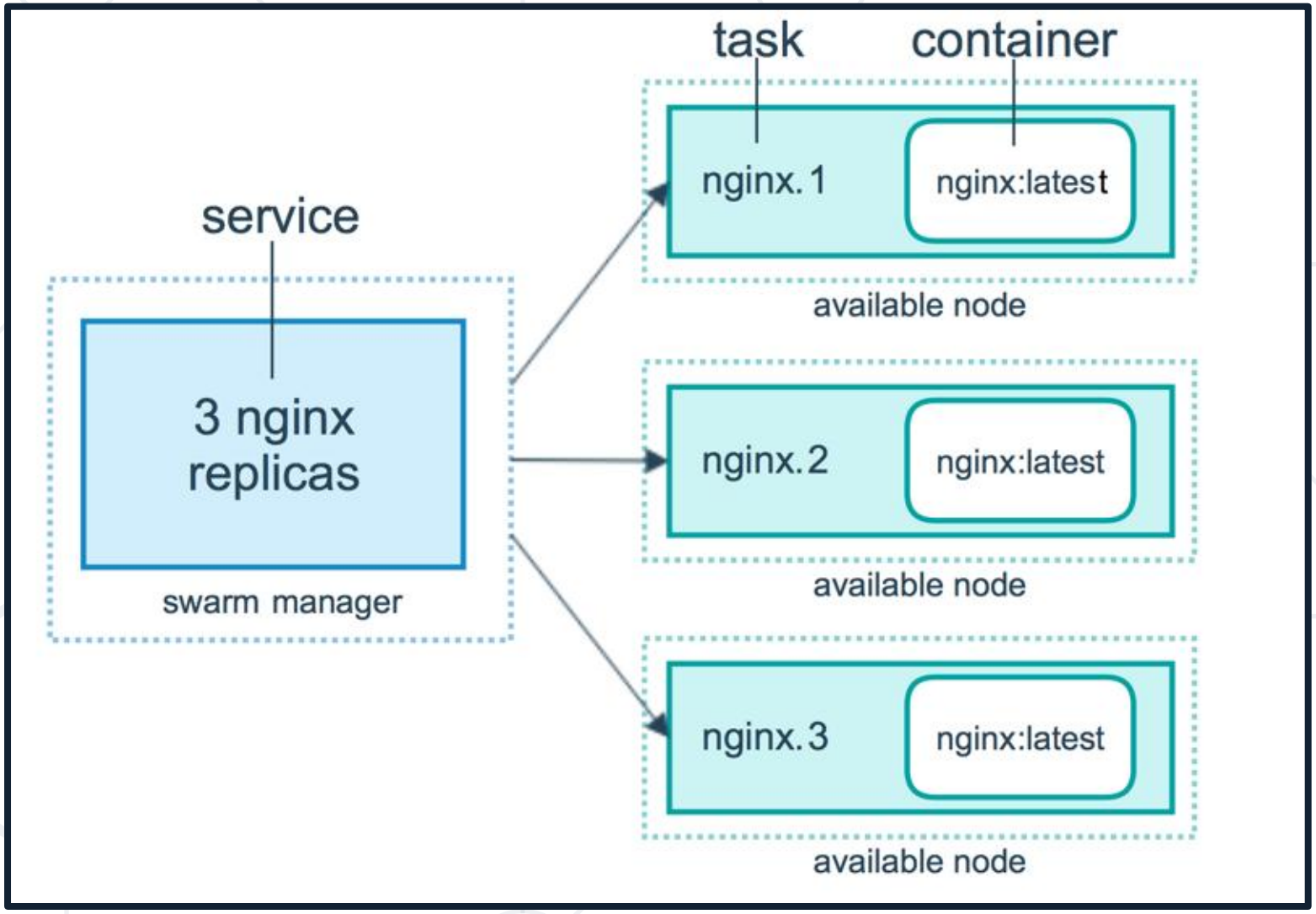
Deployment Automation

Tasks, Services, and Stacks

- Tasks are **units** of **work** distributed to nodes
- **Service** is an **application** deployed on swarm
- In fact, service is the **definition** of the **tasks** to **execute**
- **Replicated** and **global** services **distribution** model
- **Stacks** are **groups** of **interrelated** services
- Stacks are **deployed** with **docker-compose**

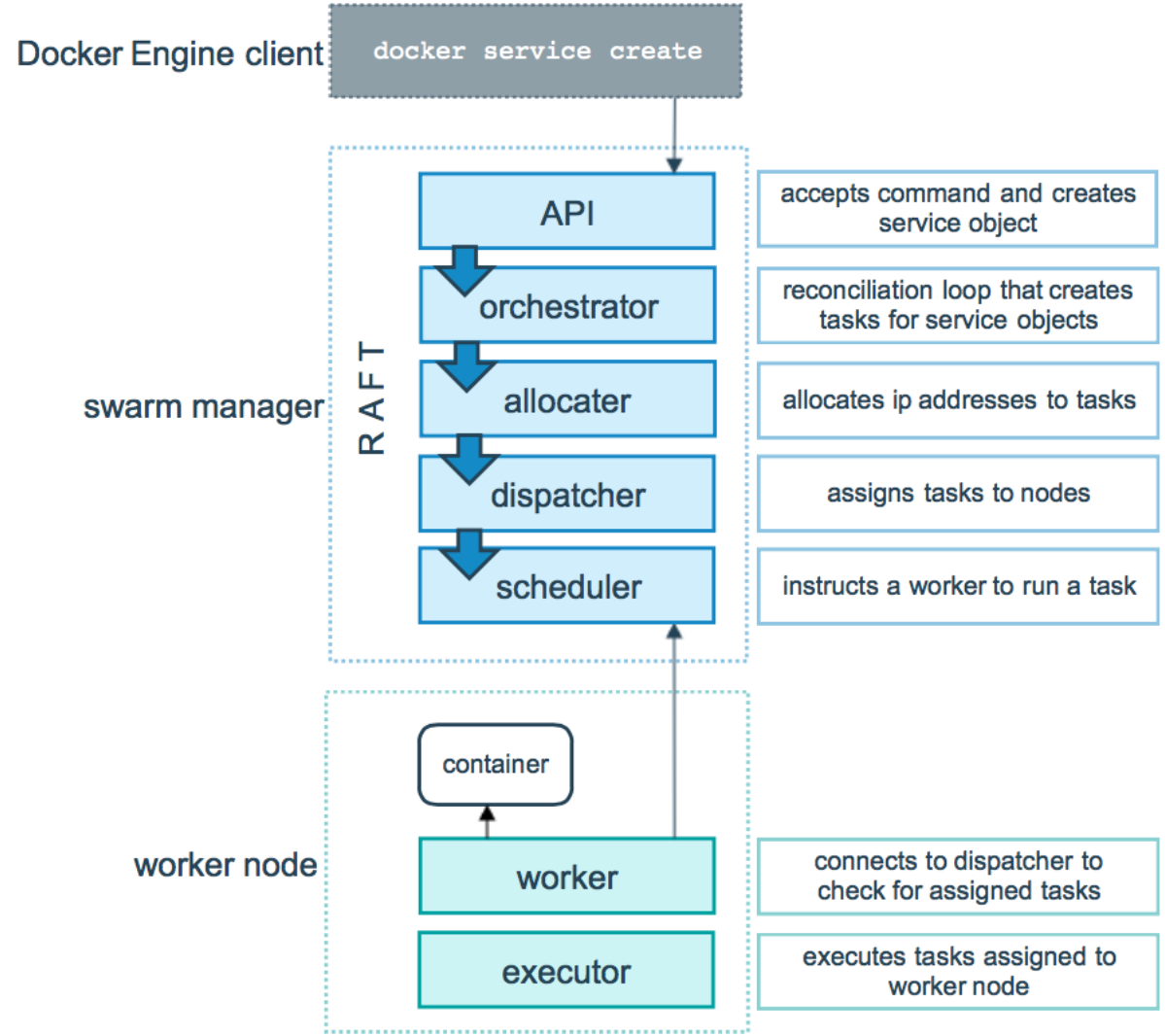


Containers, Tasks, and Services*



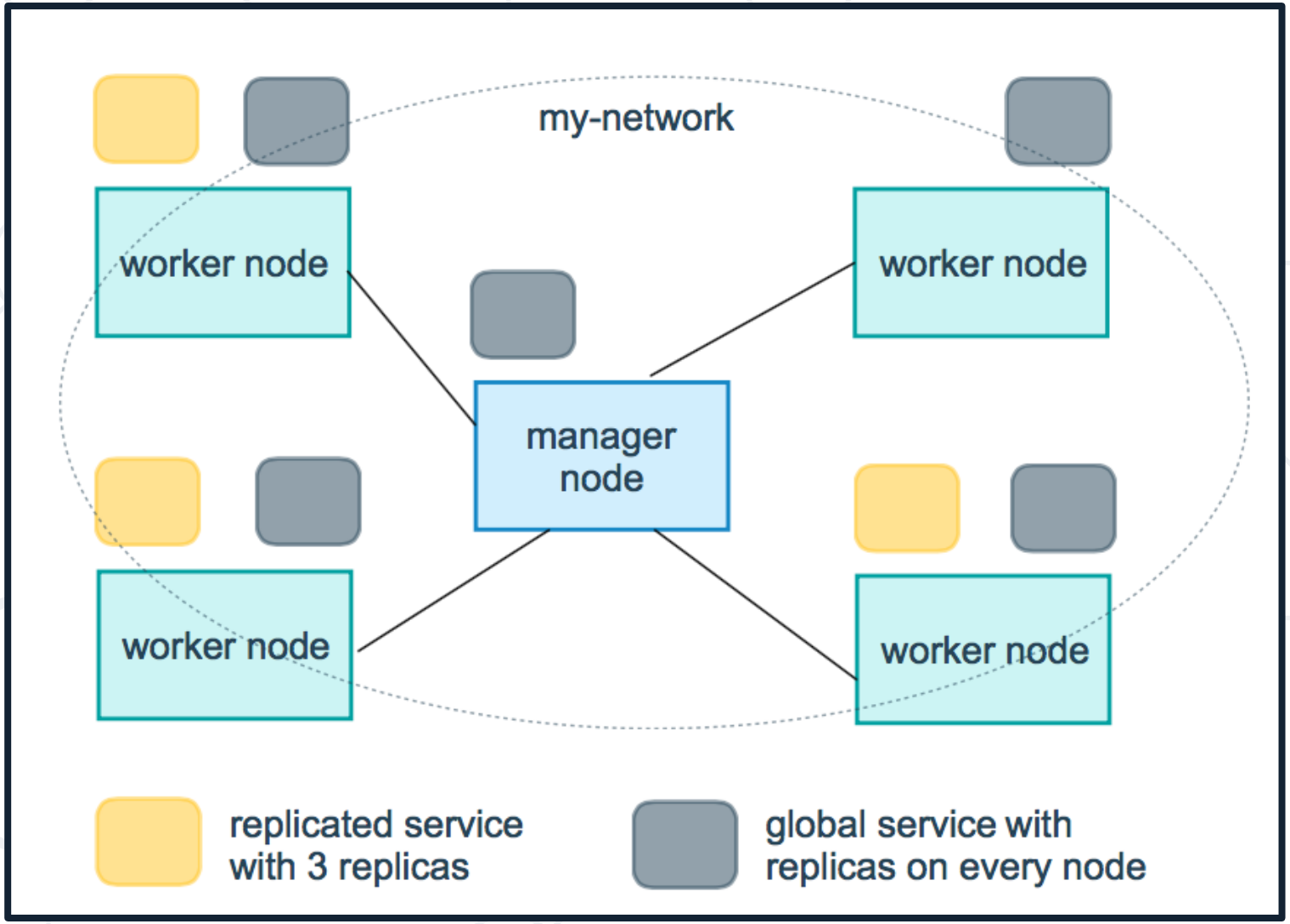
* https://docs.docker.com/engine/swarm/how-swarm-mode-works/services/

Tasks and Scheduling*



* https://docs.docker.com/engine/swarm/how-swarm-mode-works/services/

Replicated and Global Services*



* https://docs.docker.com/engine/swarm/how-swarm-mode-works/services/

Sharing Data

Configuration and sensitive data

Share Data

- Parts of a service can be scheduled on different nodes
- They may be driven by external information
- We can store the data on every node and mount it from there
- While this is working, it is not the best solution
- Especially for **configuration data** and **sensitive information**
- For these we can use one of the two special object types
  - **Configs**
  - **Secrets**

Docker Configs

Software University

- Configs are available only in Swarm mode
- Can be generic strings or binary data (up to 500 KB in size)
- Mounted directly in the container's filesystem
- Can be added or removed at any time
- Multiple services can share a config
- Managed via separate set of commands

Not encrypted

```
docker config ACTION [options]
```

- Where ACTION is either **create**, **inspect**, **ls** or **rm**

https://docs.docker.com/engine/swarm/configs/

Docker Secrets

Software University

- Secrets are available only in Swarm mode
- Can be usernames, passwords, SSH keys, certificates, generic strings or binary data (up to 500 KB in size)
- Mounted via RAM disk to the containers
- Access to secrets and be added or removed at any time
- Services can share a secret
- Managed via separate set of commands

Encrypted

```
docker secret ACTION [options]
```

- Where ACTION is either **create**, **inspect**, **ls** or **rm**

https://docs.docker.com/engine/swarm/secrets/

Practice: Swarm, Services and Stacks

Live Demonstration in Class

Podman

Docker Alternative

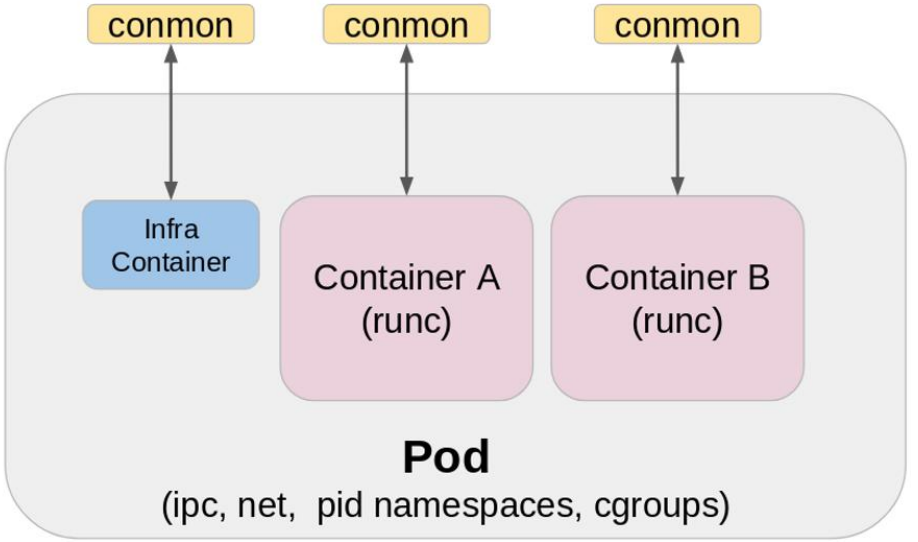
What is Podman?

Software University

- A **daemonless**, open source, Linux native tool
- Makes it easy to **find**, **run**, **build**, **share** and **deploy** applications using **Open Containers Initiative** (**OCI**) **Containers** and **Container Images**
- Provides CLI familiar to Docker users
- Relies on an OCI compliant **Container Runtime** (**runc**, **crun**, **runv**, etc.) to interface with the operating system
- Containers can either be run by root or by a non-privileged user
- Supports REST API to manage containers and remote client

Podman Objects

- Containers
- Container Images
- Container Volumes
- Pods
  - A **group of containers** that share the same network, storage, and process namespace
  - Allow us to run multiple containers that need to work closely together



https://developers.redhat.com/blog/2019/01/15/podman-managing-containers-pods

Docker Compose and Podman

- Basic support introduced with v3.0.0
- Support for Docker Compose v2.2 and higher added with v4.1.0
- Two ways to get there
  - Using **docker-compose** – depends on the REST API endpoint enablement and the installation of ***podman-docker*** and ***docker-compose*** packages
  - Using **podman-compose** – depends just on the installation of podman-compose (not developed by the project)

Practice: Podman

Live Demonstration in Class

Summary

- Distributed applications and Docker Compose
- Docker Swarm
  - How it works
  - Deployment options
  - Stacks and Compose
- Podman as Docker alternative

SoftUni Diamond Partners

Software University

Coca-Cola HBC Bulgaria

Postbank

top EMPLOYER България Bulgaria 2025

FOR A BETTER WORLD OF WORK

SUPER HOSTING .BG

createX

DRAFT KINGS

THE CROWN IS YOURS

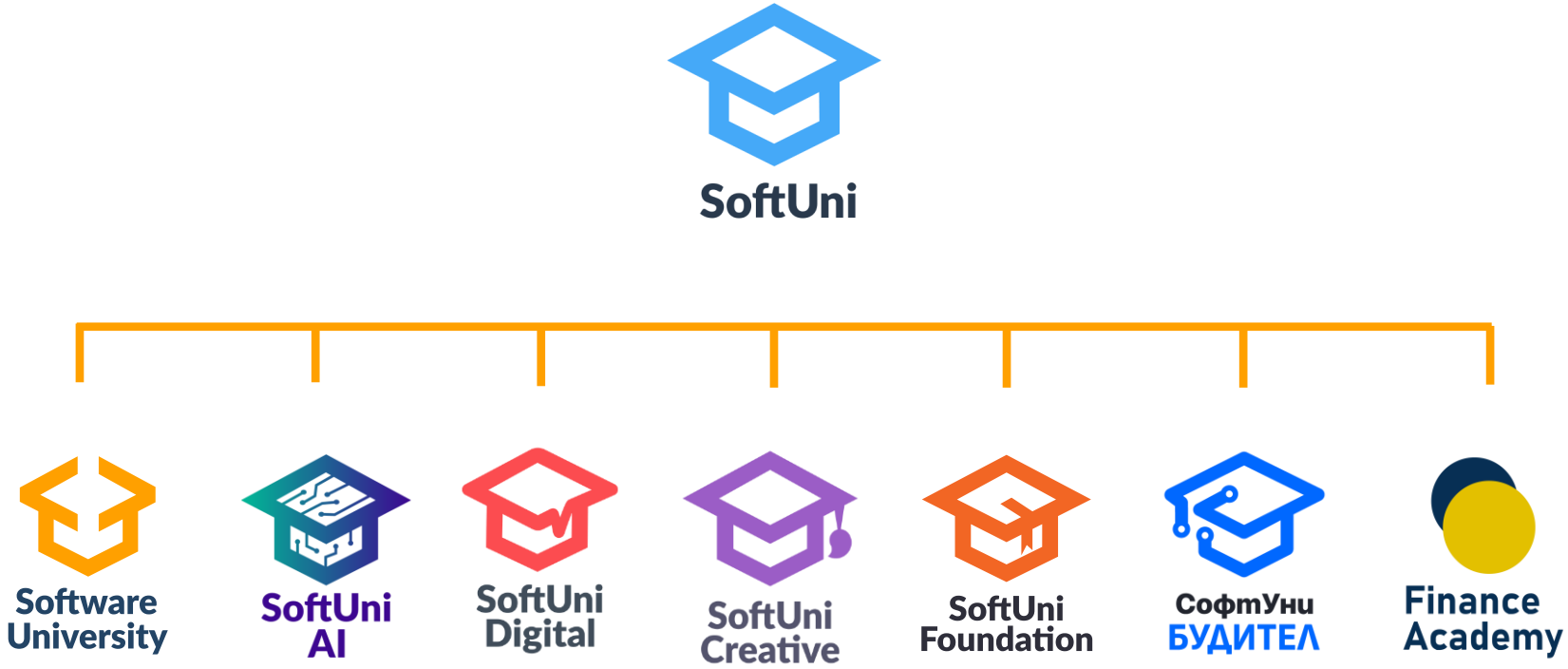
INDEAVR

Serving the high achievers

encorp.io

VIVACOM

Questions?



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

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