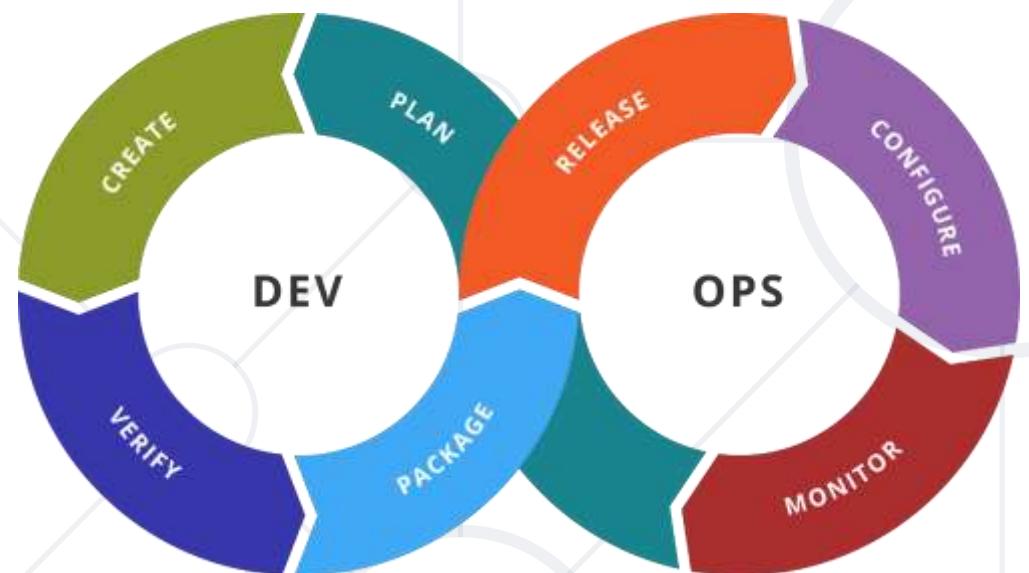


Introduction to DevOps

The Big Picture. Basic Toolkit. Basic Automation



SoftUni Team

Technical Trainers

 Software University



SoftUni



Software University
<https://softuni.bg>

You Have Questions?



sli.do

#DevOps-23

facebook.com

/groups/DevOpsContainerizationCICDMonitoringJanuary2023



This Module (M1)
Topics and Infrastructure

Table of Contents

1. The Big Picture

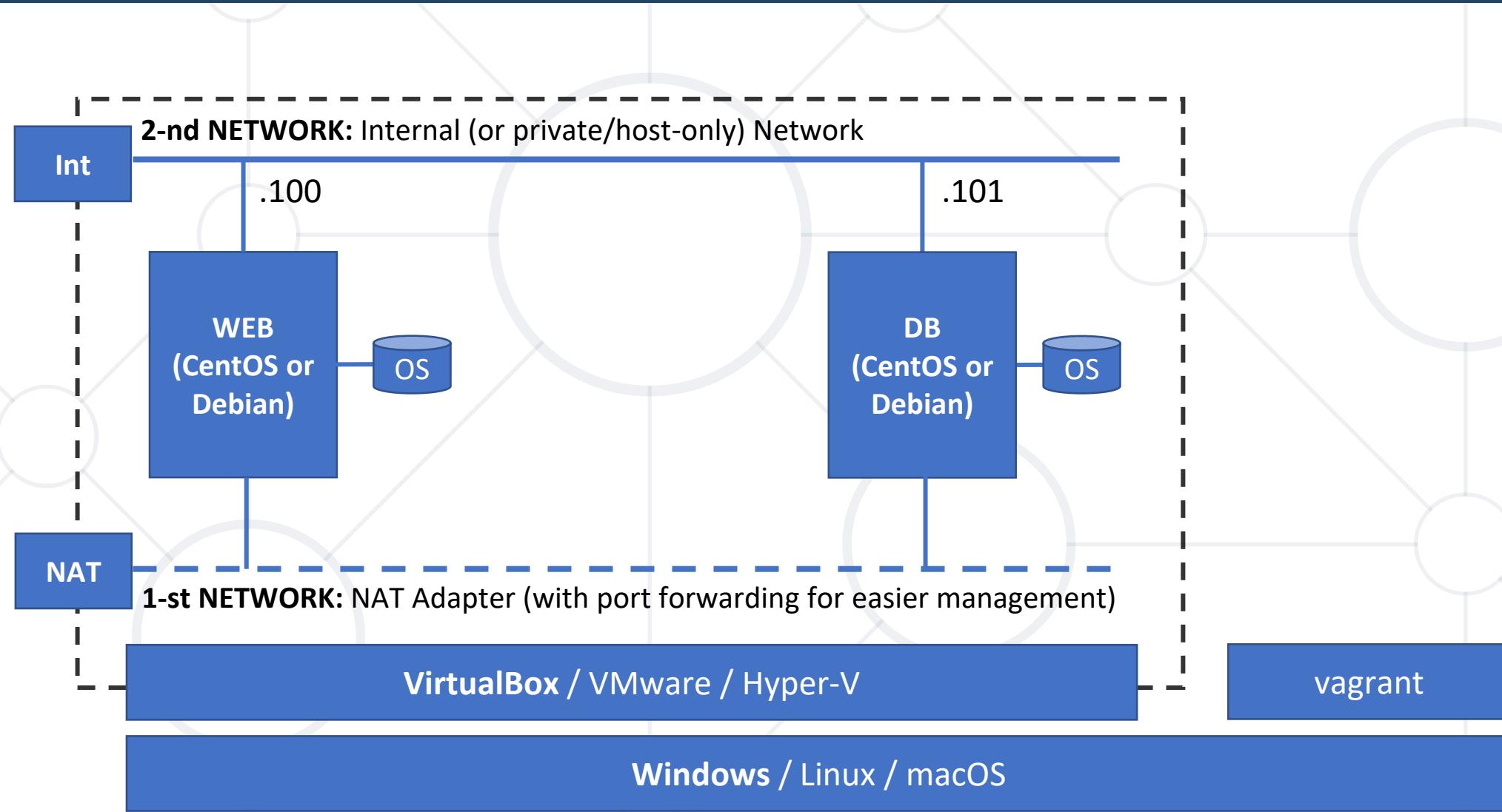
- Main Pain Points and Causes
- Goals and Benefits
- Adoption and Tools

2. Basic Toolkit

3. Basic Automation



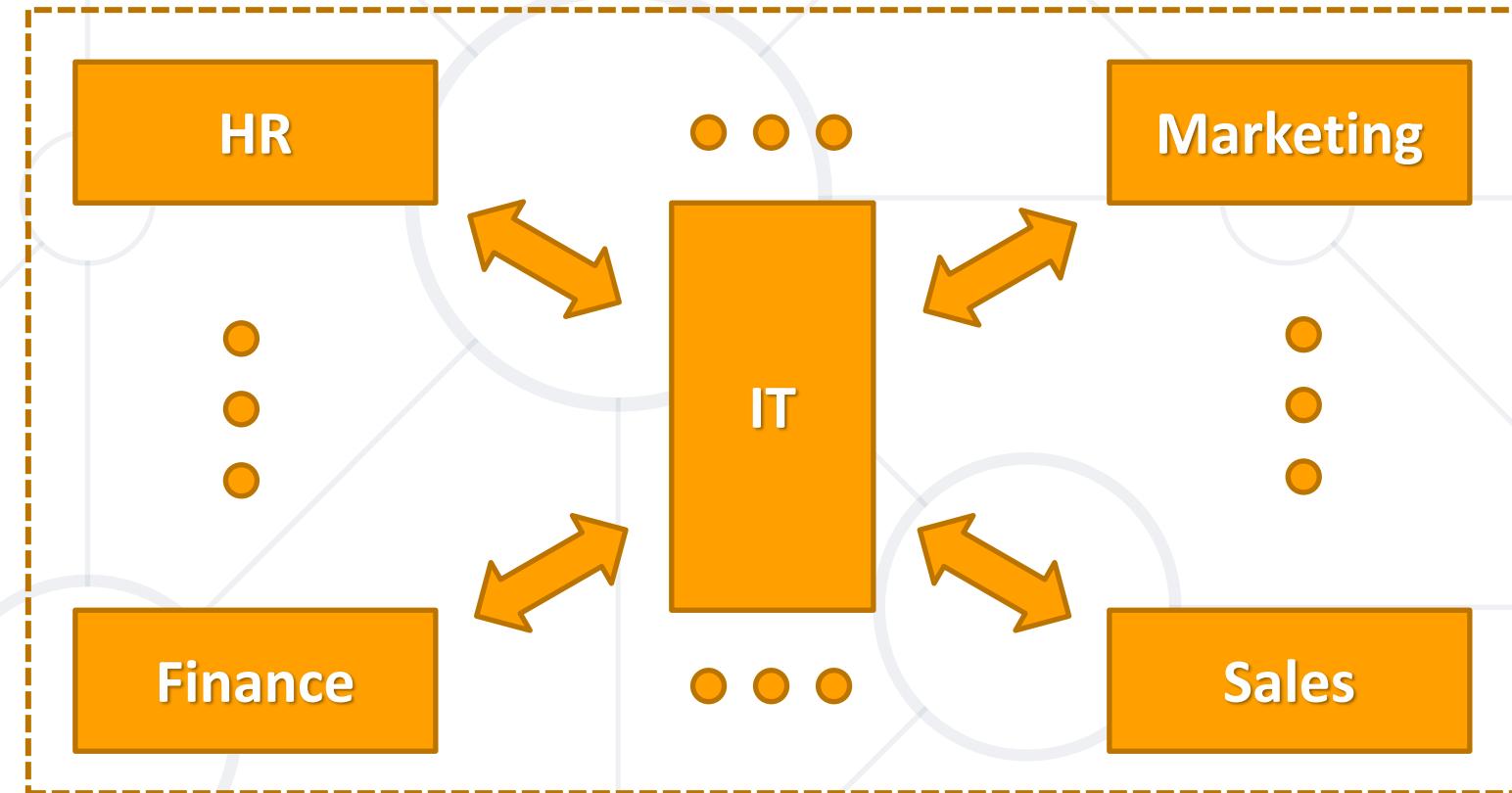
Lab Infrastructure





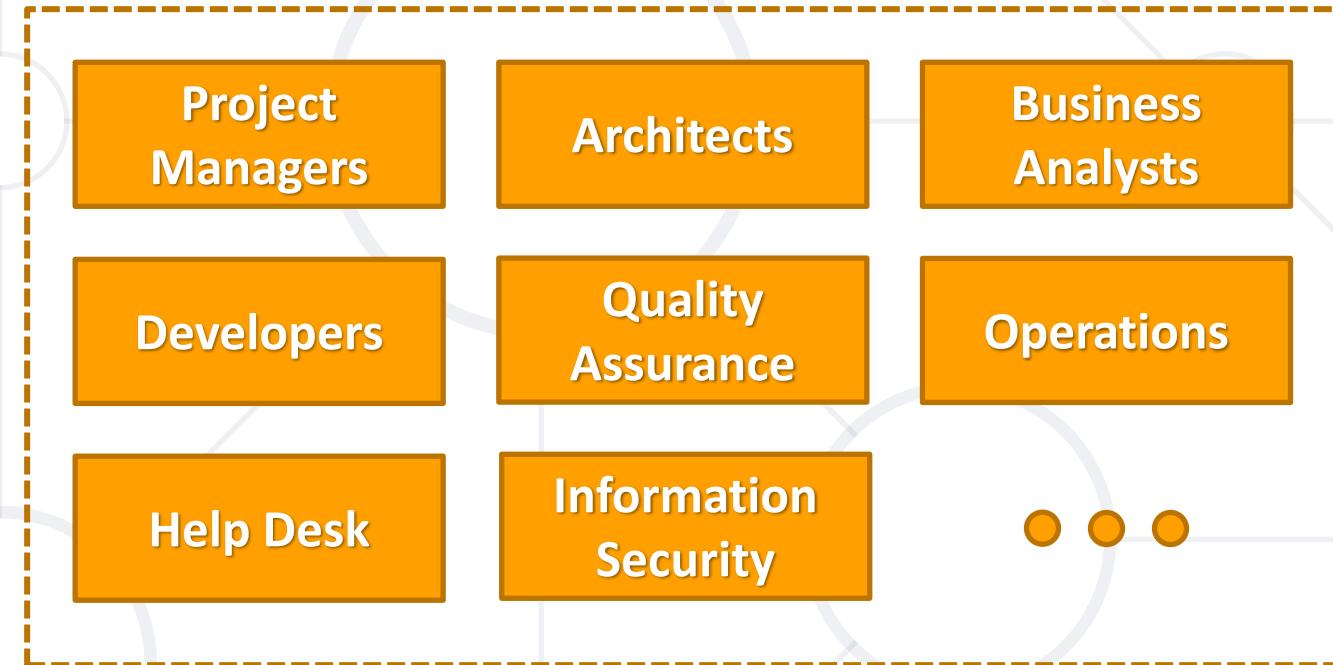
The Big Picture
(Why) Do we need a change?

Typical Company Organization*



* Many departments and all depend on IT in one way or another

Typical IT Organization*



* IT has its own units

Typical Challenges

- Complex pipeline
- Mixed environment
 - Externally customized software
 - Custom internal software
- Staff is leaving or being moved elsewhere
 - Absent know-how, outdated or missing documentation
- Operations have to maintain black-boxes



Main Pain Points
At least some of them

Outages

- Happen at the most inappropriate time
 - Typically, in high priority systems
 - Usually with long recovery time
- Lead to
 - Panic mode
 - Lost trust
- May be caused by
 - Repeated errors/issues
 - Lack of expertise

Low value

- Slow delivery
 - Long time to wait before the actual consumption
- Long implementation periods
 - Often the delivery is outdated and doesn't match the current requirements

- In fact, is more a perception than a reality
- It leads to
 - Software-as-a-Service solutions
 - Departmental solutions
- It is caused by
 - Long waiting
 - Too many restrictions
 - Poor performance

Fighting

- Involved parties
 - IT vs. Business
 - Internally in IT
- Typically caused by
 - Lost trust
 - Absence of transparency
 - Different motivators or bonus schemes



Causes

Lack of Knowledge

- **Poor communication**
 - Between the departments
 - Internally in IT, between roles
- **Missing documentation**
- Hard to **distinguish between important and unimportant**
 - Too many meetings
 - Too many and too complex reports

Slow Processes

■ Sources

- Slow provisioning of environments
- Approval takes time
- (Too many) too specialized people
- Next role is waiting for the previous one to finish

■ Affected parties

- Internal
- External

Over-something

- **Over-provision**
 - Request more resources than actually needed
- **Over-production**
 - Ask for features just to keep everyone busy
- **Over-processing**
 - For example, apply unnecessary transformations over and over
- **Over-delivery**
 - Deliver more than requested

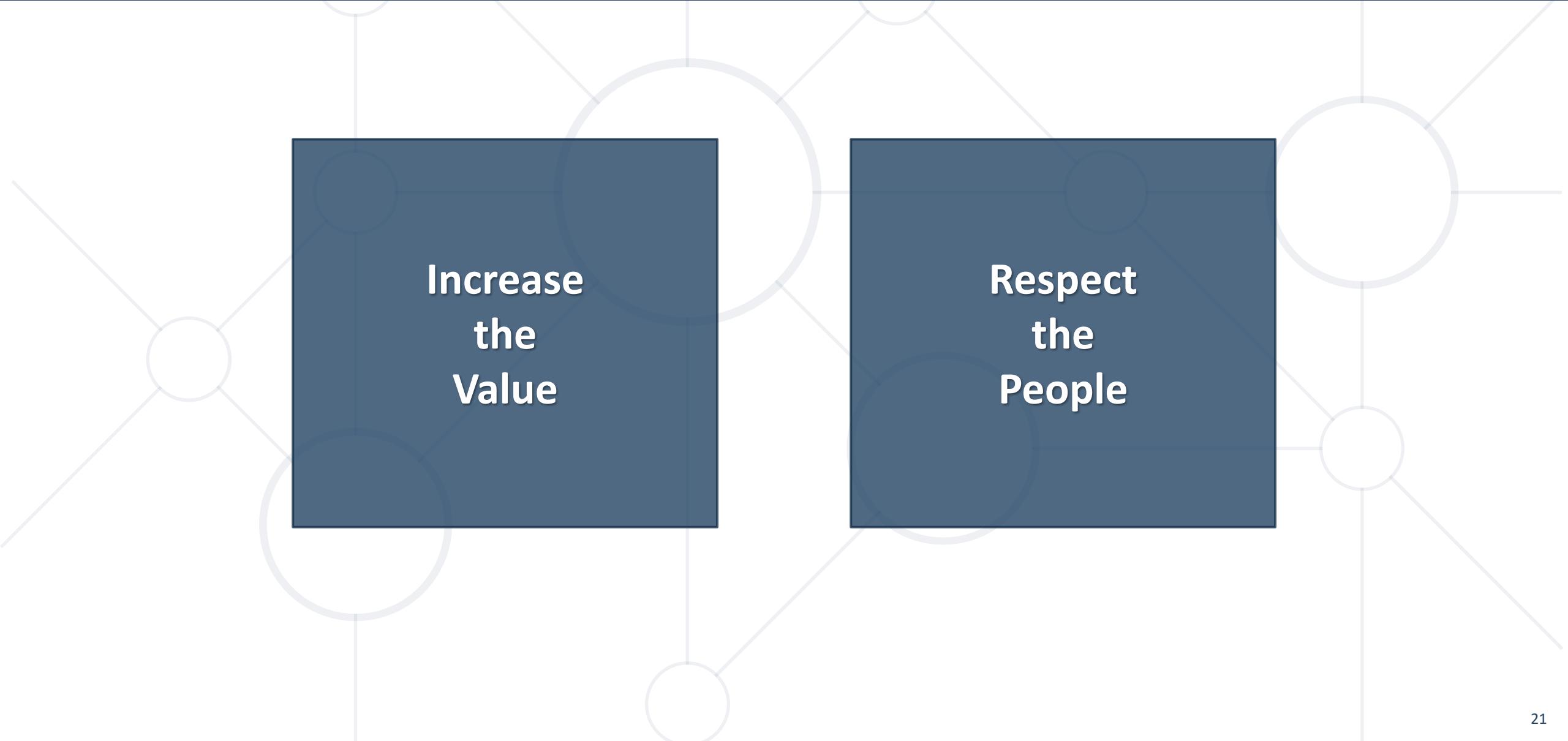
- **Slow delivery of features**
 - Long (time expensive) updates
 - Repetitive manual testing procedures
- **Unnecessary iterations**
 - From environment to environment
- **Delivery in a rush**
 - Do it on time no matter the quality
- **Postpone a delivery**
 - A ready feature is waiting something else to be shipped first



DevOps

Goals and Benefits

Main Goal



Increase
the
Value

Respect
the
People

How can DevOps help?

- Can add value and flow improvement
- Mind the prerequisites
 - Identify a **shared pain**
 - **Address the causes**
- Embrace the result
 - **Added value => financial impact**

DevOps is Lean for IT

- We should not cut costs, but free up resources
- This can be achieved by
 - Focus on customer value
 - Optimize the process
 - Reduce delivery time
 - Shared knowledge
 - Avoid batching
 - Address bottlenecks

Core Values* of DevOps Movement

- **Culture**
 - Break down barriers between teams, safe environment
- **Automation**
 - Save time, prevent defects, create consistency, self-service
- **Measurement**
 - If you can not measure it, you can not improve it
- **Sharing**
 - Sharing the tools, discoveries, and lessons



Adoption

Making the Transition

Three Main Tasks

Change
the
Culture

Change
the
Organization

Handle
the
Objections

Change Culture

- **Question** to identify the **Shared Objective**
- **Data-driven** decisions
- **Authorization** for action
- **Responsibility** for actions
- (Cross-) **Teamwork and Respect**
- **Learning and Sharing** even from **Mistakes**
- **Trust** (between parties)
- **Values and Rewards** (who, why, how)
- **Continuous improvement** mindset

Change Organization (understand & assess)

- **Understand the processes**
 - All components - systems, people, value, etc.
 - Achieve **clear vision**
- **Assess and acknowledge bottlenecks**
 - Inconsistent environments
 - Manual and custom builds
 - Poor quality
 - No communication

Change Organization (change & improve)

- **Change (or adjust) team structure (if needed)**
 - Concentrated knowledge or many tasks assigned to one person
 - Generalists vs Specialists
 - Complete (or consistent) team
 - Prepare handoff (think about the next step)
- **Assessment of people and processes**
 - Including the rewards and punishment system
 - Including the budgeting
- **Clean-up (remove extra steps, components, ...)**

Typical Objections to Handle

- **Security** (developers on production, security issues)
 - Communication, Upfront quality, Proper testing, Ship quickly
- **Compliance** (restricted access, all-or-nothing)
 - Better control - who, what, where
- **Remote teams** (internal teams or external parties)
 - Shared objectives, Technology solutions, Renegotiation
- **Impact on employees**
- **Presence of legacy systems**
- **Lack of appropriate skills** (technical and soft skills)



Mentality and Tools

~~If it isn't broke, don't fix it~~



Continuous Improvement

Tools

- Planning (transparency)
- Issue tracking (feedback)
- Source control (control code & configuration)
- Building and testing
- Security assessment (vulnerabilities, secrets, privileges, etc.)
- Continuous integration and deployment
- Configuration and infrastructure management (consistency)
- Monitoring and logging (measurement)
- Collaboration and knowledge sharing (connect & share)

Cloud Platforms could provide the whole environment or serve one or more of the listed categories



DevOps

It's Time For a Break
Back in a few minutes ☺



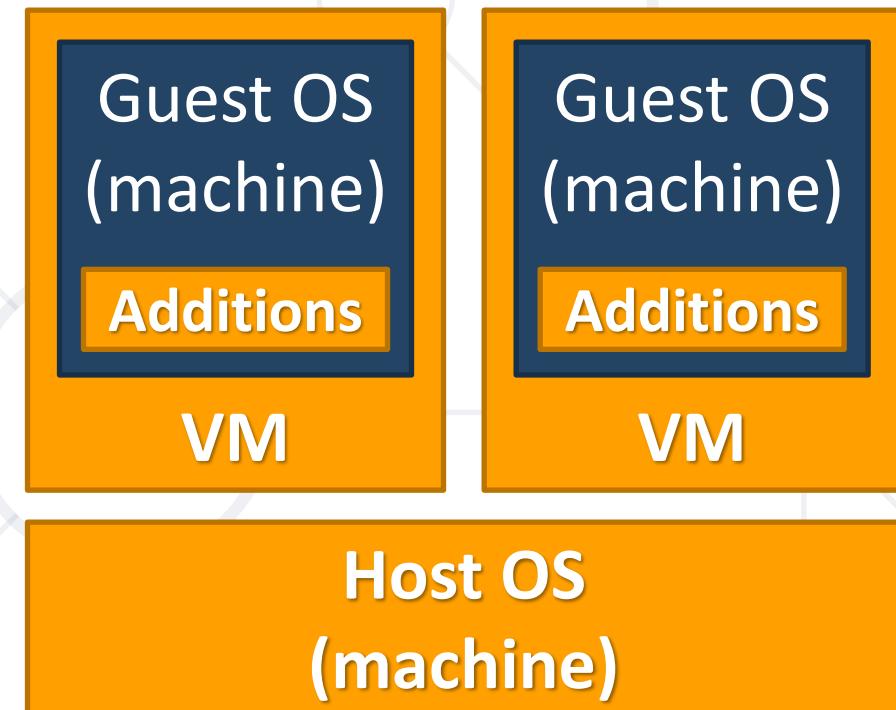
DevOps

Virtualization

Fundamental Principles and Use Cases

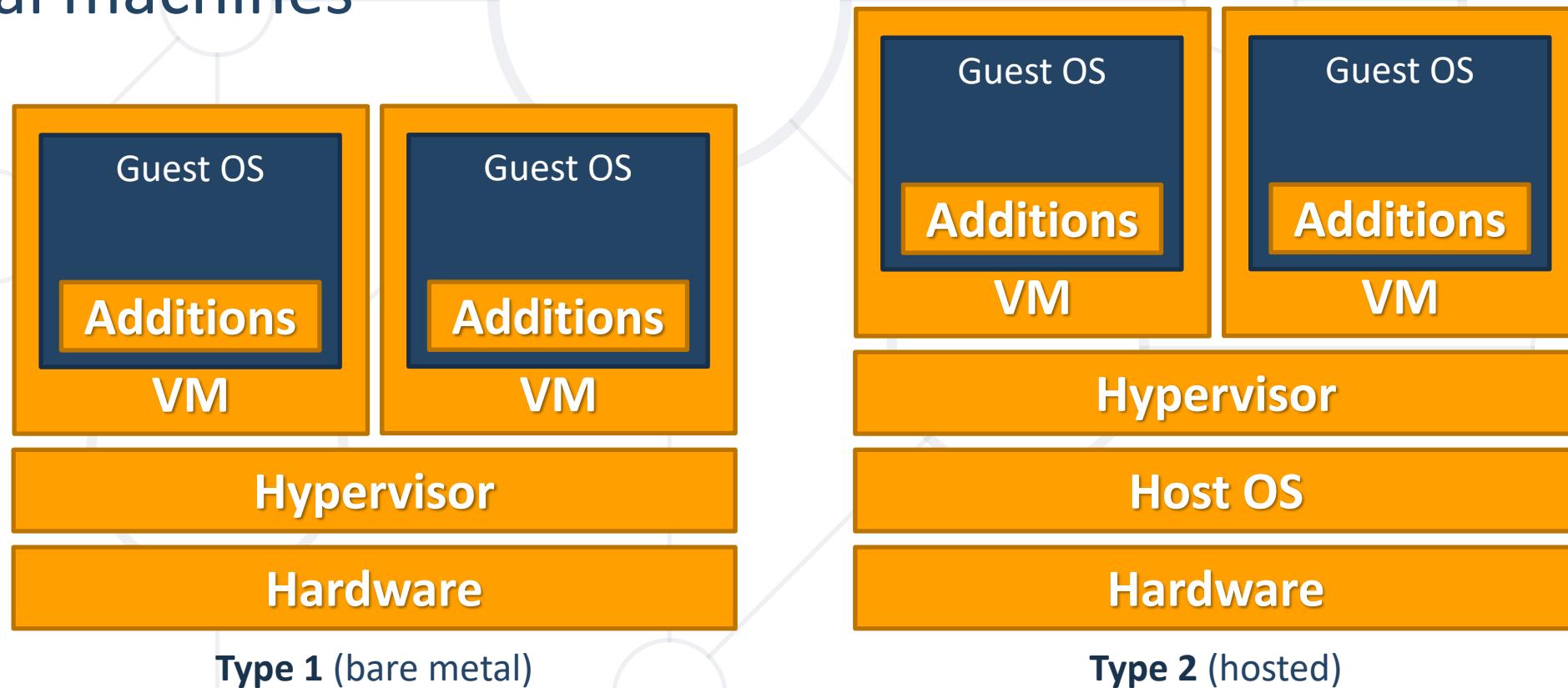
What is Virtualization?

- Virtualization is the act of creating a **software-based** or virtual (rather than physical) version of something
- Main definitions
 - Host OS (machine)
 - Virtual machine
 - Guest OS (machine)
 - Guest additions



Hypervisors

- A **hypervisor** or **virtual machine monitor (VMM)** is computer software, firmware, or hardware, that creates and runs virtual machines



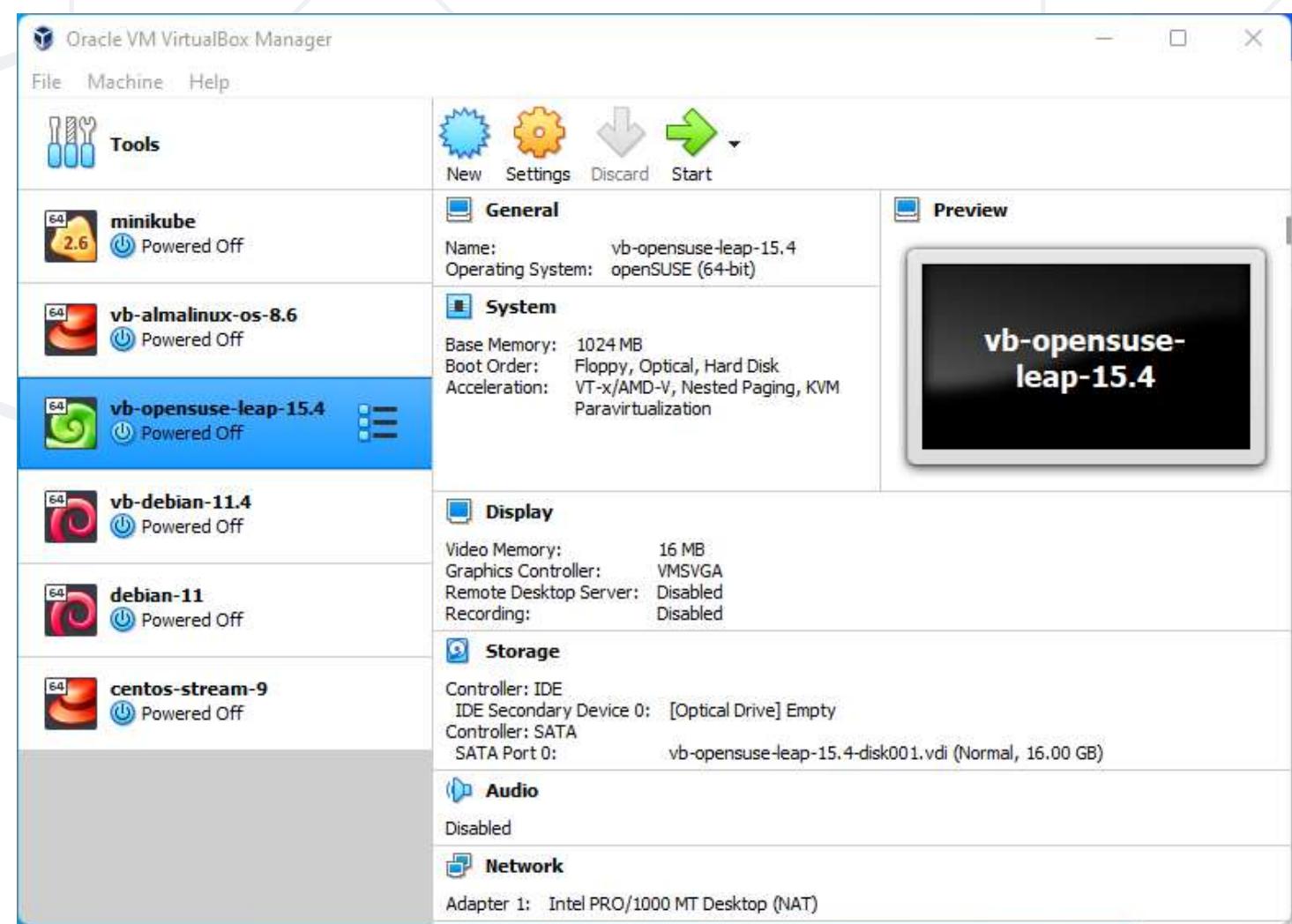
- **Infrastructure consolidation**
 - Better usage and utilization of the available hardware
- **Maintain separate environments**
 - For example – development, test, production
- **Testing and evaluation**
 - Test a newer software version or evaluate a product
- **High availability and disaster recovery**

Our Case

- We would like to
 - Install **multiple** machines on **limited** hardware resources
 - Manage their **isolation**
 - Manage their **state** – our own time-machine
 - **Move, export, and import** them
 - **Clone** them – create **multiple copies** out of one master
- The answer is **Virtualization**

VirtualBox

- Cross-platform
- Broad guest OS support
- Easy to install
- Simple GUI
- Automation options
- Free





Linux

Why Linux?

- It is a **phenomenon**
 - Went all the way from a **student's hobby** to **world domination**
- Internet **runs** on Linux
 - Operating system for **over 95%** of the top one million domains *
- It runs on **100% of the top 500** supercomputers **
- There is **huge demand** for Linux skills
- It is both **challenging** and fun

* <https://www.linuxfoundation.org/about>

** <https://www.top500.org/statistics/details/osfam/1>

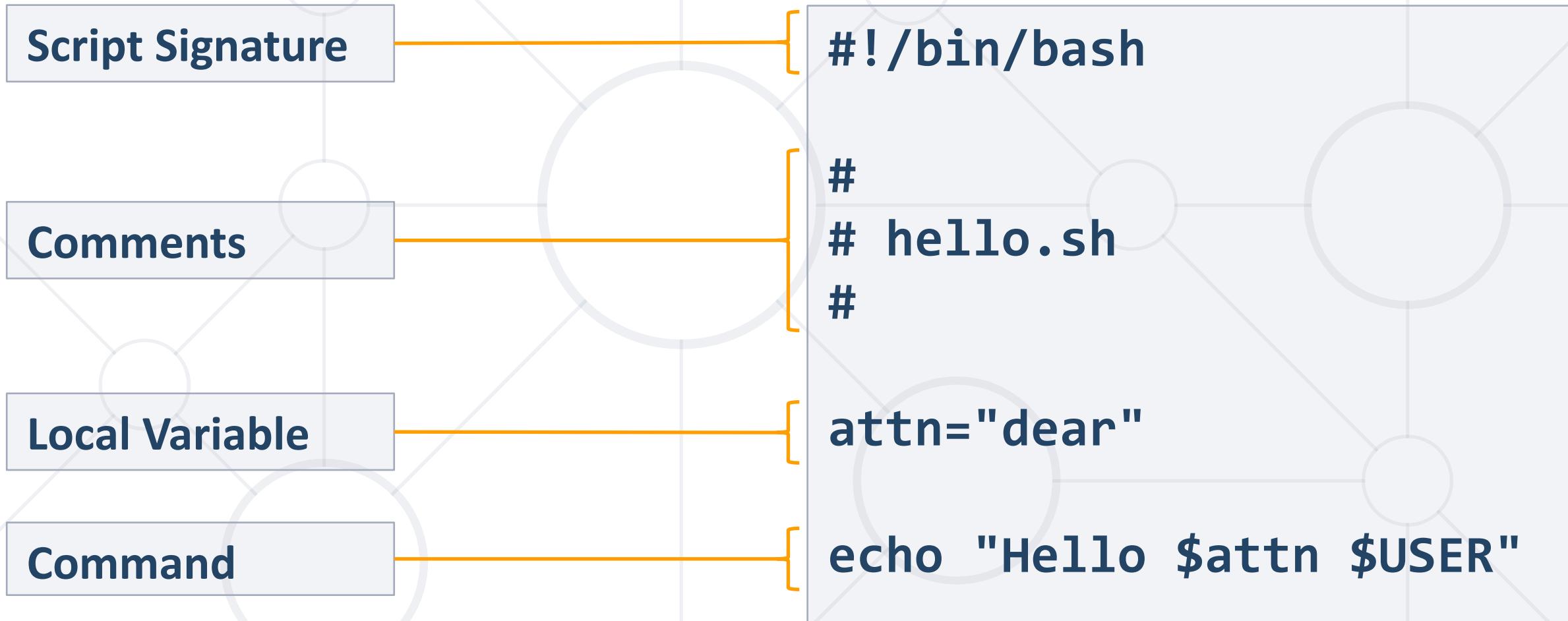
What we need to know?

- General knowledge about Linux
- Working with users, groups, and permissions
- Working with files and folders
- Handling some basic network related tasks
- Software and services management
- Basic bash scripting skills



Bash Scripting
Structure. Flow Control. Sourcing. Execution

Structure



Execution: **bash hello.sh** or **./hello.sh** or just **hello.sh**

- Description
 - Display line of text
- Example

```
[user@host ~]$ echo 'Hello world!'  
'Hello world!'
```

```
[user@host ~]$ echo -e 'Line 1\nLine2'  
Line1  
Line2
```

- Description
 - Read a line from the standard input and split it into fields
- Example

```
[user@host ~]$ read -p "Enter name:" NM_ENT  
Enter name: James
```

```
[user@host ~]$ echo $NM_ENT  
James
```

- Description
 - Execute commands based on conditional
- Example

```
count=1
if [ $count -eq 0 ]; then
    echo 'Equal to 0'
else
    echo 'Not equal to 0'
fi
```

- Description
 - Evaluate conditional expression
- Example

```
# Compare numbers: OP1 -eq|-ne|-lt|-le|-gt|-ge OP2
```

```
# Compare strings: ST1 =|!=|<|> ST2
```

```
# Compare files: FL1 -nt|-ot FL2
```

```
# File tests: -d|-e|-f|-x FILE
```

for

- Description
 - Execute command for each member in a list
- Example

```
# List all files with prefix "item:"  
  
for i in $( ls ); do  
    echo item: $i  
done
```

while

- Description
 - Execute commands as long as a test succeeds
- Example

```
# Print numbers from 1 to 5
count=1
while [ $count -le 5 ]; do
    echo $count
    count=$((count+1))
done
```

- Description
 - Execute commands as long as a test does not succeed
- Example

```
# Print numbers from 1 to 5
count=1
until [ $count -gt 5 ]; do
    echo $count
    count=$((count+1))
done
```

Sourcing vs. Execution

- Sourcing
 - No subshell is created
 - Any variables set become part of the environment
 - Methods: **. script.sh** or **source script.sh**
- Execution
 - Subshell is always created
 - No subshell if using **exec ./script.sh**

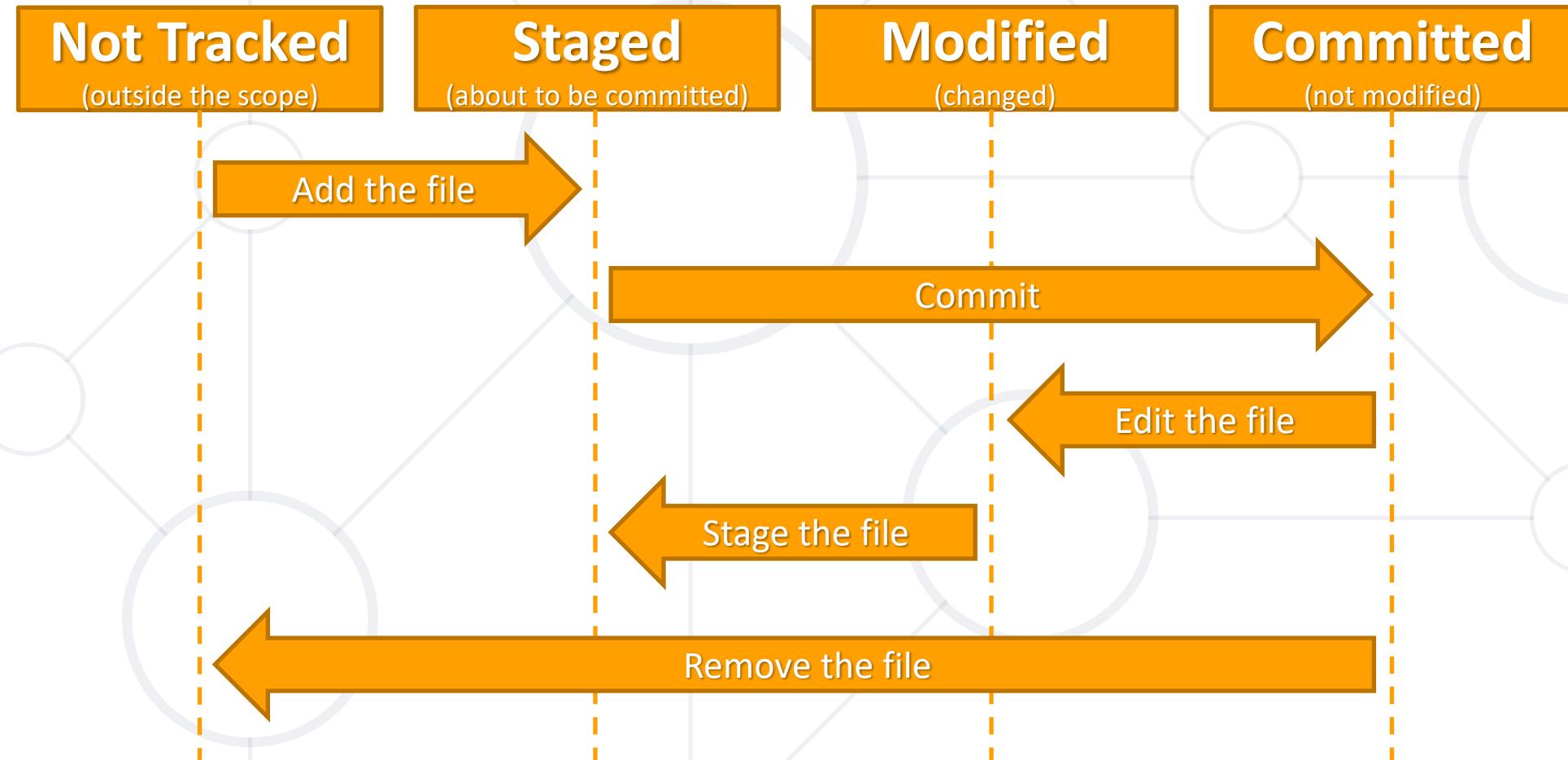


Source Control
Git. Files Lifecycle. Basic Commands

General Information

- Distributed Version Control
- Created by the Linux development community in 2005
- Can be used on-premise and in the cloud
- Snapshot based
- Three states - Committed, Modified, and Staged

Files Lifecycle



Basic Git Commands

- Create an empty Git repository

```
git init
```

- Clone an existing repository

```
git clone https://github.com/user/repo
```

- Show repository objects

```
git show
```

- Show different states of files in working directory and staging

```
git status
```

Basic Git Commands

- Add files from working directory to staging

```
git add file.txt
```

- Remove a file from staging and working directory

```
git rm old_file.txt
```

- Move or rename file

```
git mv old_file.txt new_file.txt
```

- Commit the staged changes

```
git commit
```

Basic Git Commands

- Get all not existing objects from remote repository

```
git fetch https://github.com/user/repo
```

- Get and merge changed objects from remote repository

```
git pull
```

- Push the changes to a remote repository

```
git push
```



Practice: Non-automated Way
Live Demonstration in Class



DevOps



Vagrant

Introduction. Basic Commands

Introduction

- Building and managing virtual machine environments
- Supports providers like VirtualBox, VMware, AWS, etc.
- Provisioning tools such as shell scripts, Chef, or Puppet
- Multiplatform
- Integration with source control systems
- Public boxes catalog: <https://app.vagrantup.com/boxes/search>
- Local storage for boxes: <~/.vagrant.d/boxes>

- Boxes are the package format for the Vagrant environment
- They can be used by anyone on any supported platform
- Used to bring up an identical working environment
- Box files have three different components
 - **Box File** - Compressed (tar, tar.gz, zip) file that is specific to a single provider and can contain anything
 - **Box Catalog Metadata** - JSON document that specifies the name of the box, a description, available versions, etc.
 - **Box Information** - JSON document that can provide additional information

Box Creation

- Create a tiny VM
- Install the OS with minimalistic profile (SSH included)
- Install any additional required tools and services
- Install hypervisor addons (for example, VirtualBox Add-ons)
- Make the **vagrant** user a sudoers member
- Install the insecure vagrant key
- Cleanup packages cache and align the hard drive
- Package and publish the box

Vagrantfile

- Ruby syntax
- One file per environment
- General file structure

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
...
Vagrant.configure("2") do |config|
  config.vm.box = "shekeriev/debian-11"
...
end
```

Vagrantfile

```
Vagrant.configure("2") do |config|
  config.vm.box = "shekeriev/centos-8-minimal"
  # Provider settings
  config.vm.provider "virtualbox" do |vb|
    # Display the VirtualBox GUI when booting the machine
    vb.gui = true
    # Customize the amount of memory on the VM:
    vb.memory = "1024"
  end
  # Provisioning section
  config.vm.provision "shell", inline: <>SHELL
    dnf -y upgrade
    dnf install -y httpd
  SHELL
end
```

Basic Vagrant Commands

- Initialize the environment

```
vagrant init [options] [box]
```

- Login to HashiCorp's Vagrant Cloud

```
vagrant login
```

- Connect to machine via SSH

```
vagrant ssh [options] [name|id]
```

- Check status of a vagrant machine

```
vagrant status [name|id]
```

Basic Vagrant Commands

- Start and provision vagrant environment

```
vagrant up [options] [name|id]
```

- Stop a vagrant machine

```
vagrant halt [options] [name|id]
```

- Stop and delete vagrant machine

```
vagrant destroy [options] [name|id]
```

- Manage boxes

```
vagrant box <subcommand> [<arguments>]
```



Practice: Vagrant in Action
Live Demonstration in Class

Summary

- DevOps
 - Is for companies of any size
 - Adds value and flow improvement
- DevOps is a combination of
 - Cultural changes
 - Organizational changes
 - Tools
- We are not alone – there is a toolkit to help us
- Vagrant allows us to automate infrastructure life-cycle



Resources

Vagrant download

<https://developer.hashicorp.com/vagrant/downloads>

Vagrant documentation

<https://developer.hashicorp.com/vagrant/docs>

Vagrant boxes repository (Vagrant Cloud)

<https://app.vagrantup.com/boxes/search>

VirtualBox download

<https://www.virtualbox.org/wiki/Downloads>

VirtualBox documentation

<https://www.virtualbox.org/manual/UserManual.html>

CentOS download

<https://www.centos.org/download/>

CentOS (Red Hat) documentation

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9

Debian download

<https://www.debian.org/download>

Debian documentation

<https://www.debian.org/doc/>



SoftUni Diamond Partners



SCHWARZ



Coca-Cola HBC
Bulgaria



Postbank

Решения за твоето упре



Bosch..IO****



SmartIT



CAREERS



INDEAVR
Serving the high achievers

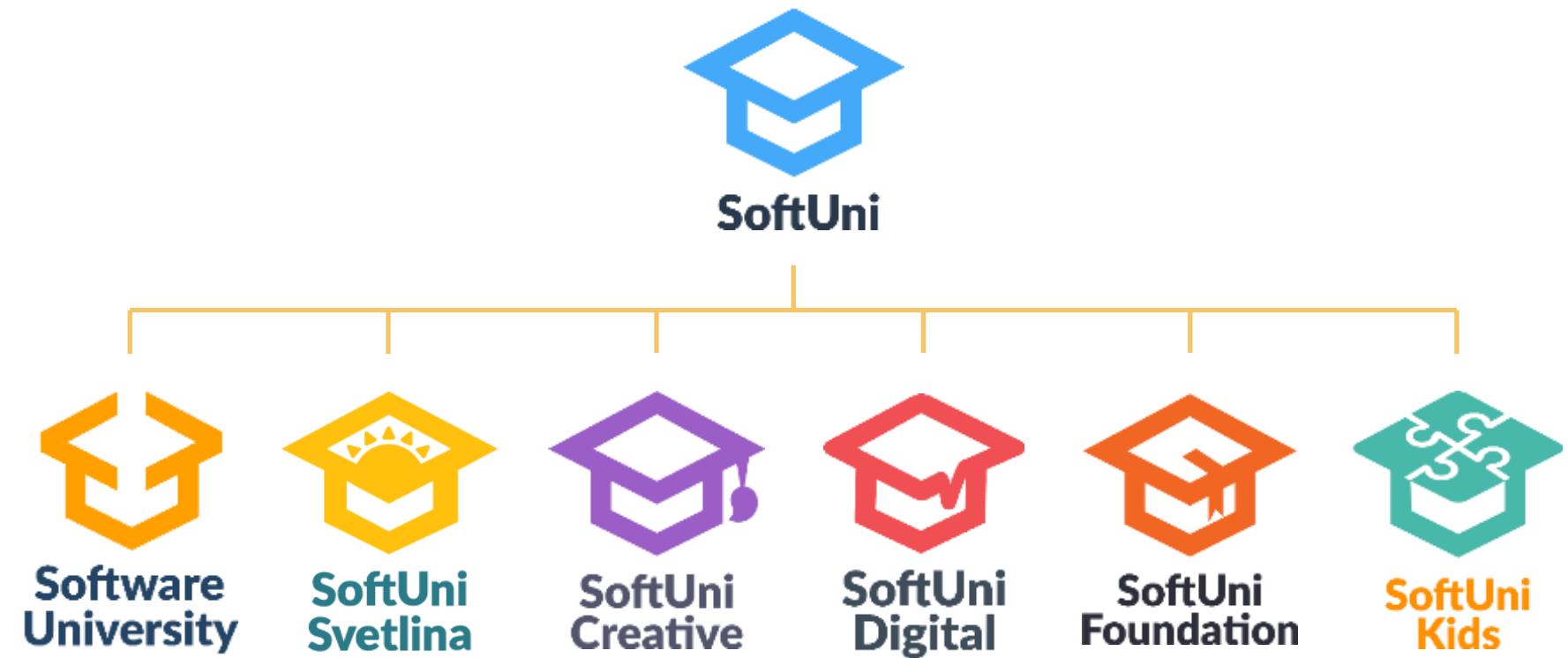


**SUPER
HOSTING
.BG**

Educational Partners



Questions?



- 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, Profession 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

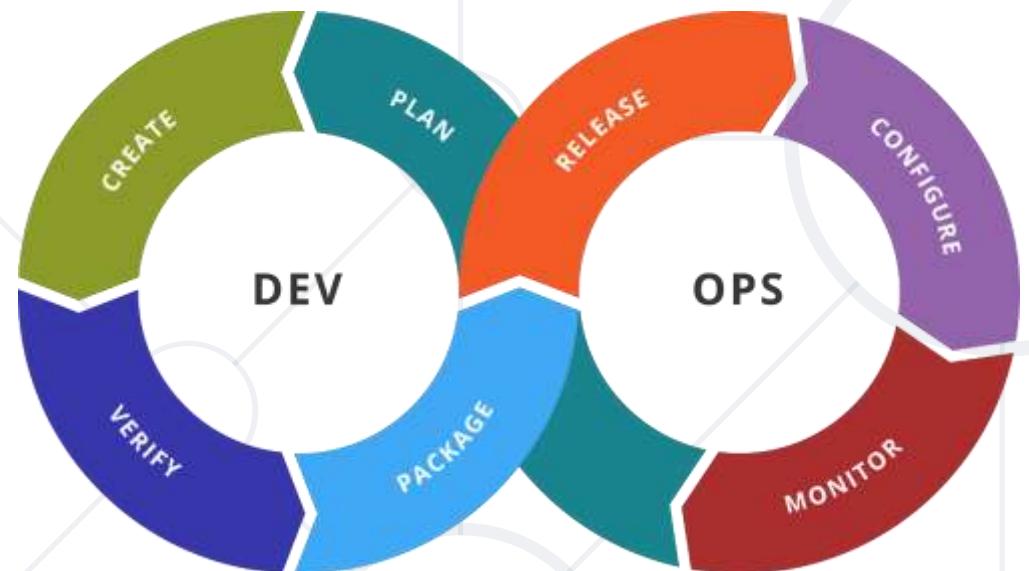


Software
University



Introduction to Docker

Install Docker. Work with Images & Containers



SoftUni Team

Technical Trainers



SoftUni



Software University
<https://softuni.bg>

You Have Questions?



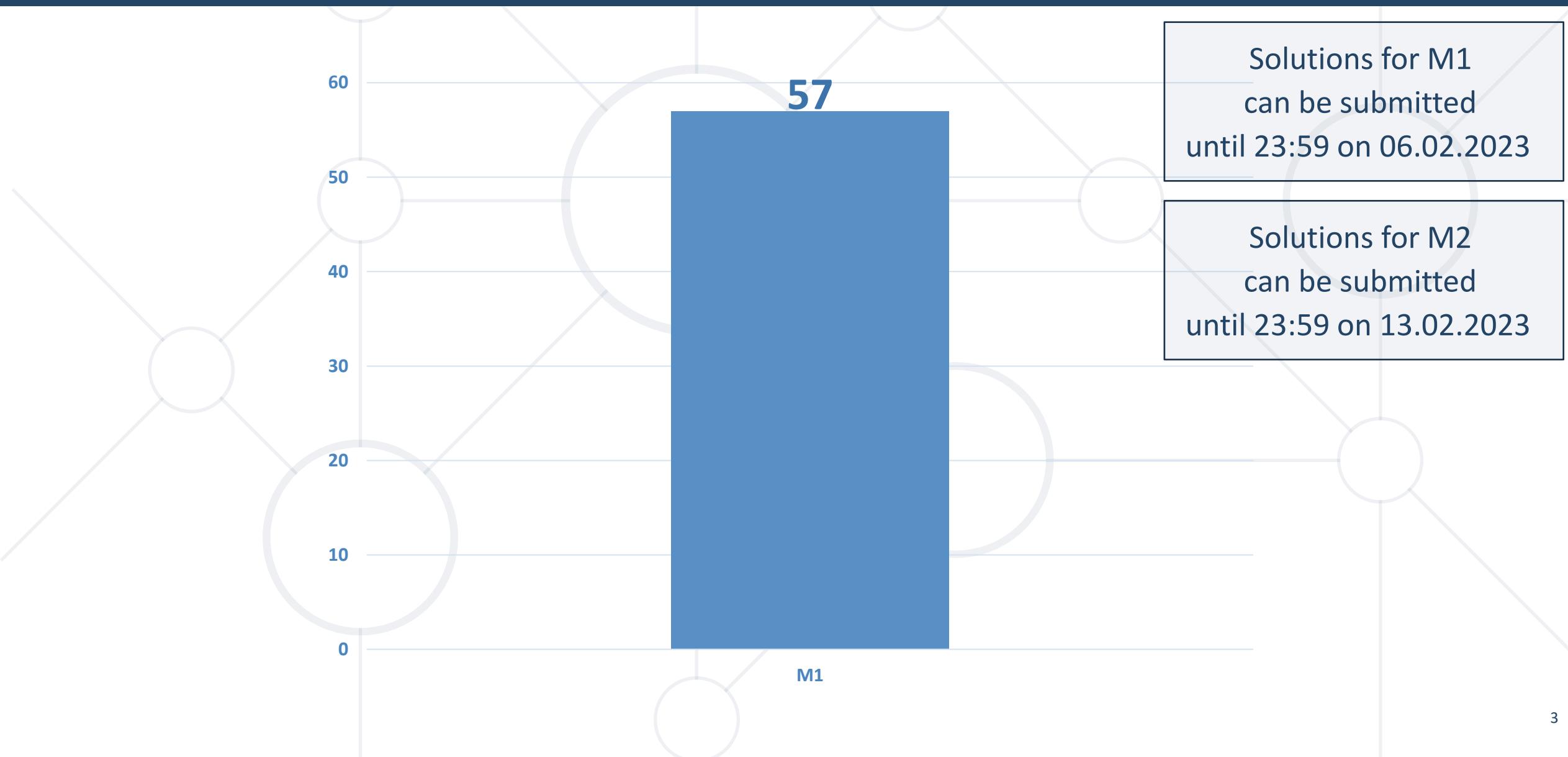
sli.do

#DevOps-23

facebook.com

/groups/DevOpsContainerizationCICDMonitoringJanuary2023

Homework Progress





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



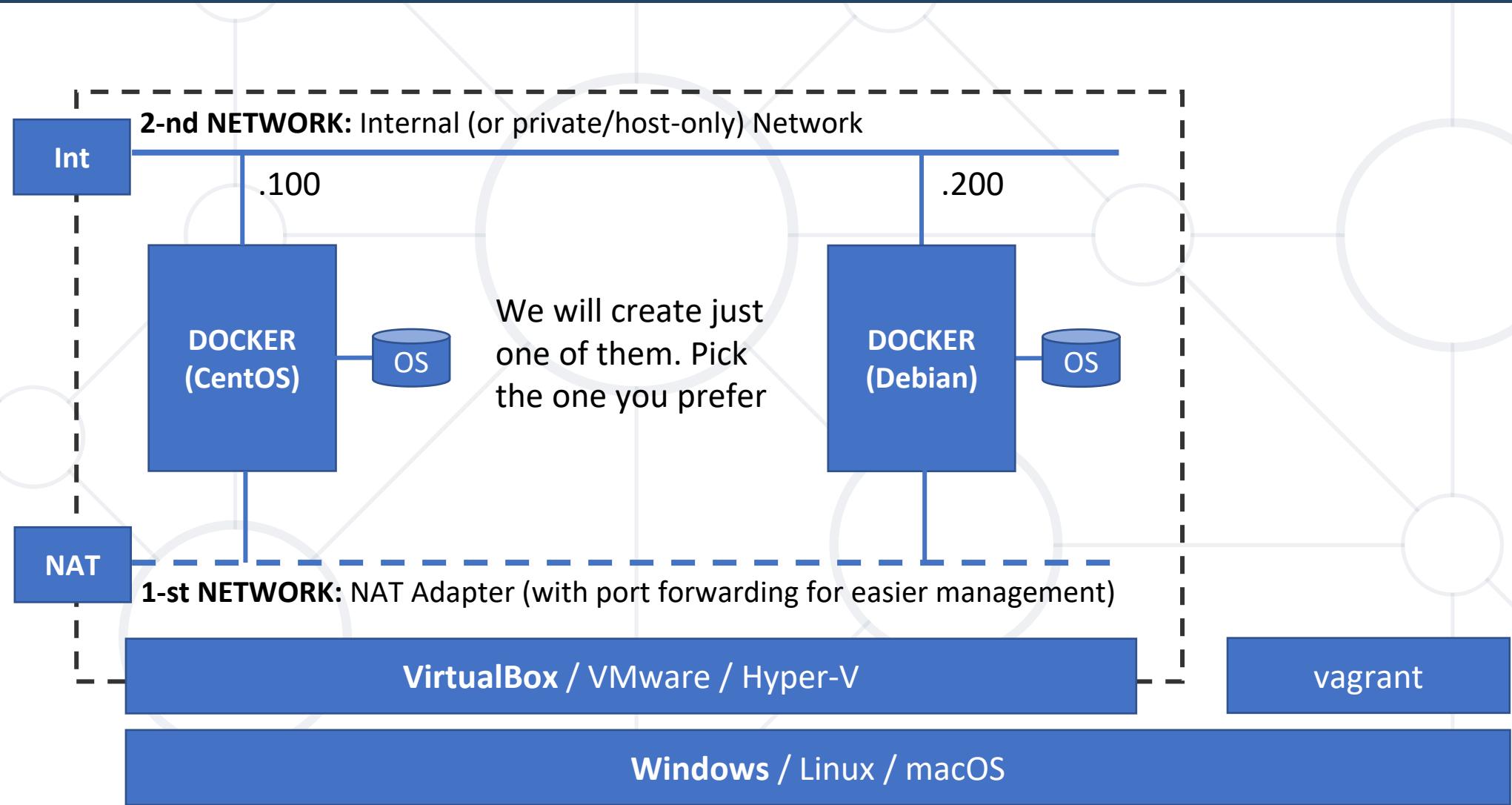
This Module (M2)
Topics and Lab Infrastructure

Table of Contents

1. Containerization
2. Introduction to Docker
3. Docker in Action
4. Create Our Own Images



Lab Infrastructure





Containers and Docker

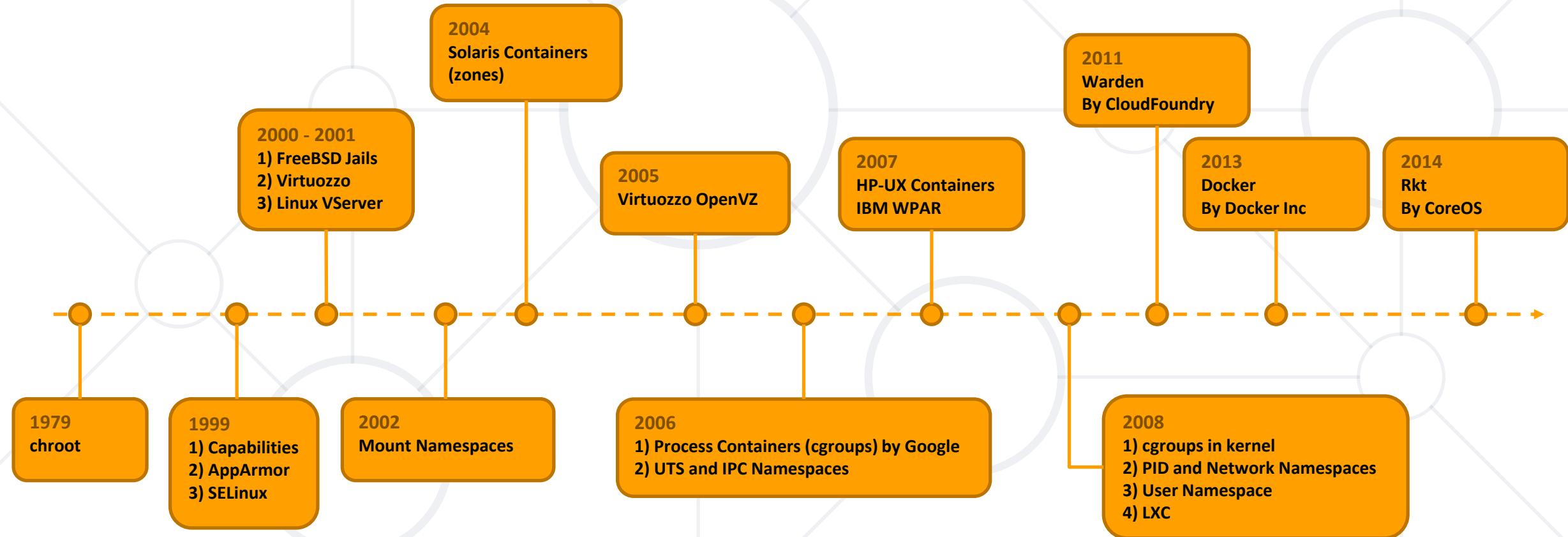
Past. Present. Future

Containerization

“ 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, ...** ”

Road to Containers

Companies and Solutions



Enablement Technologies

Container Types and Solutions

■ 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/>

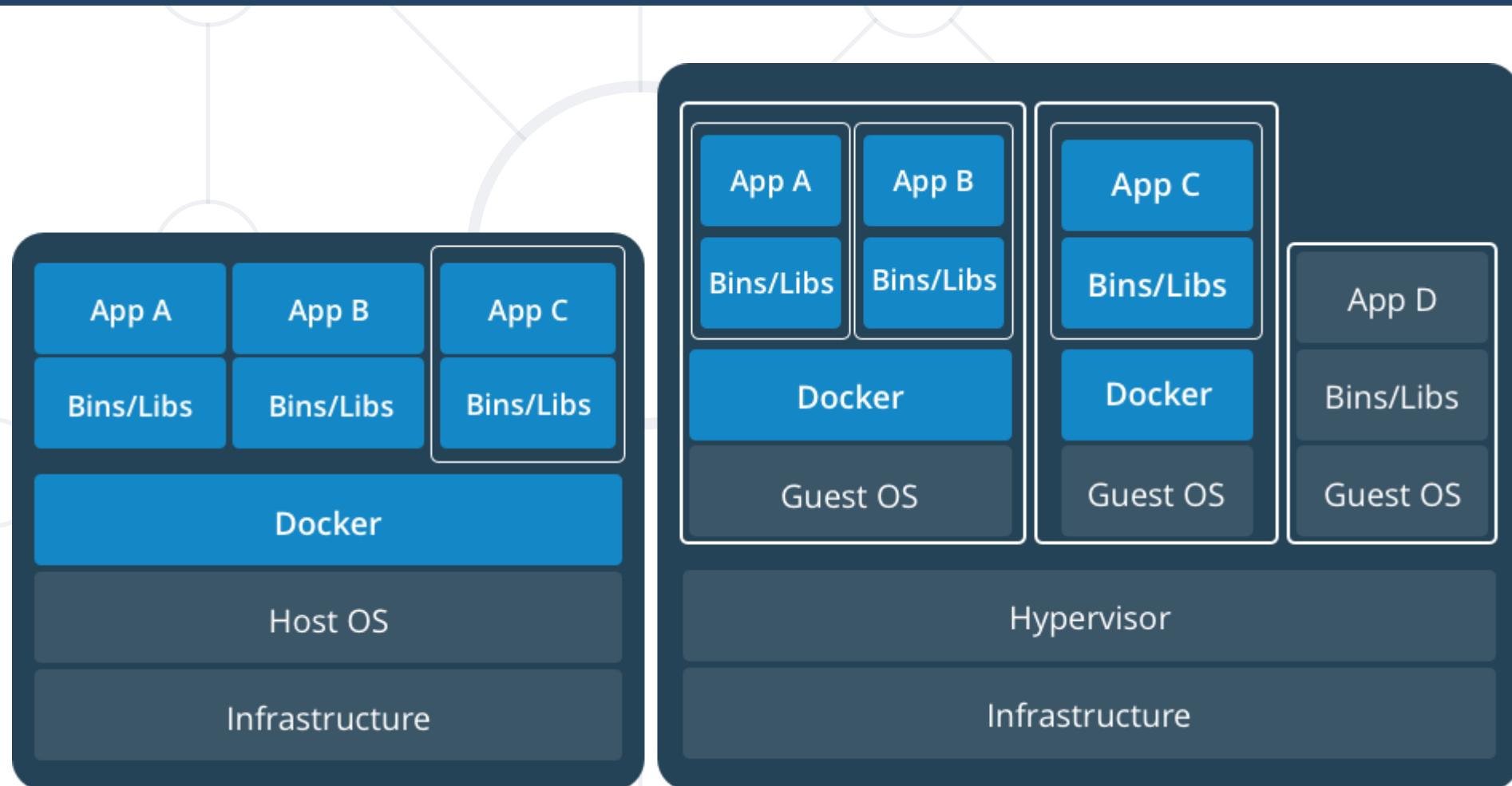
App-centric
Single process *

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)

- **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

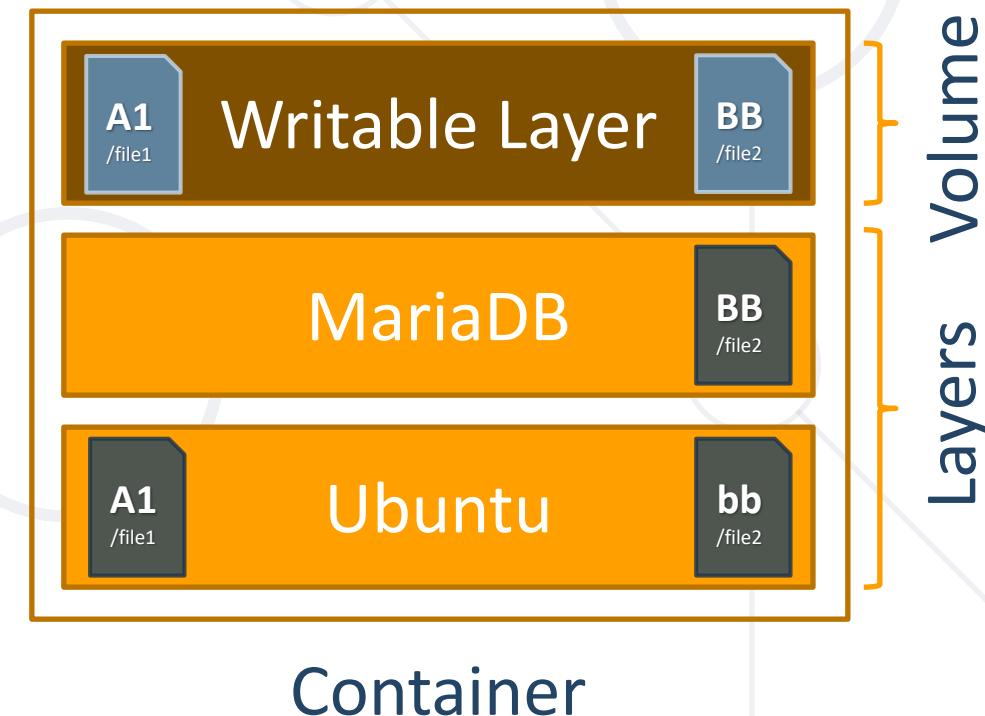
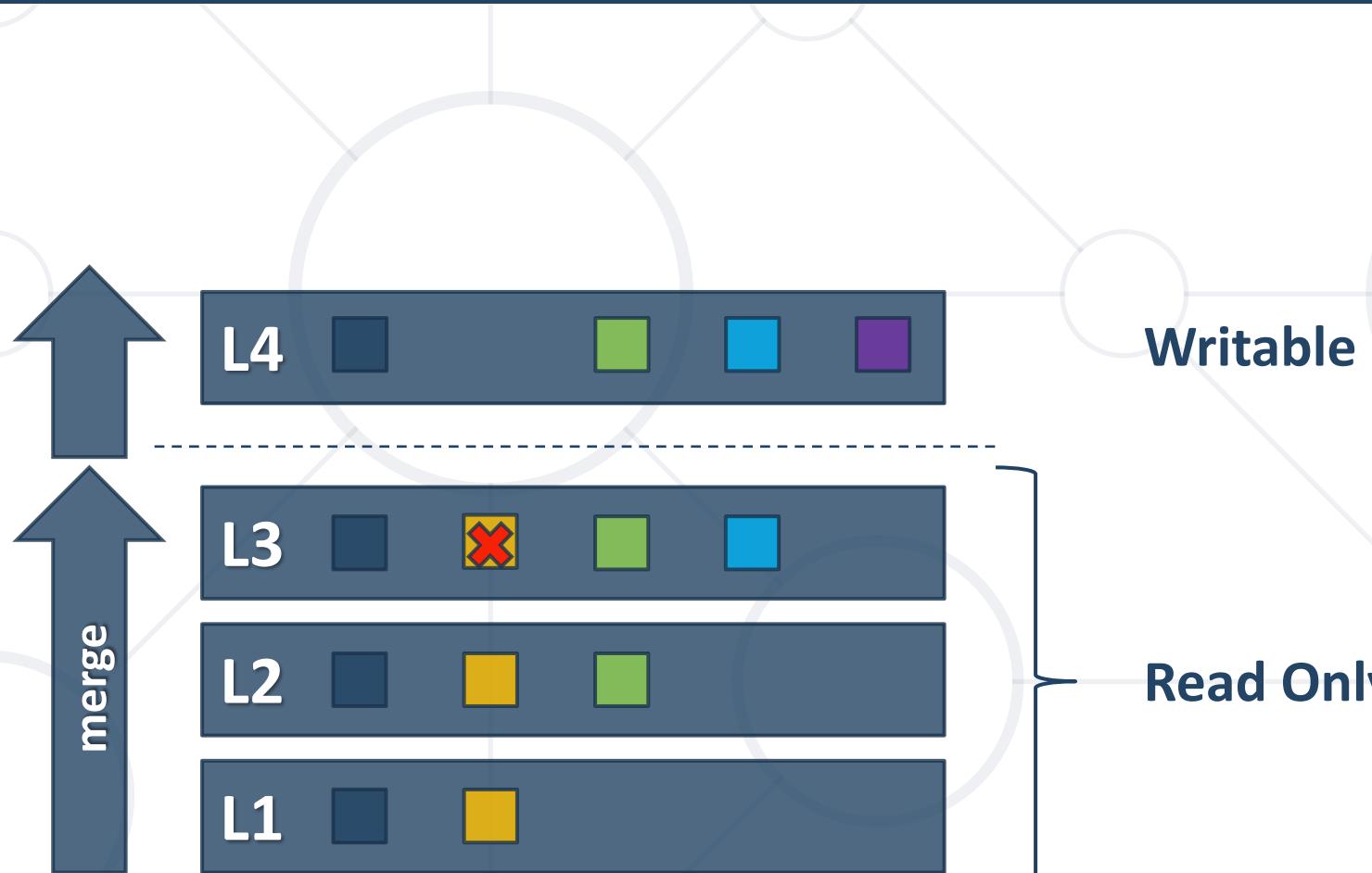
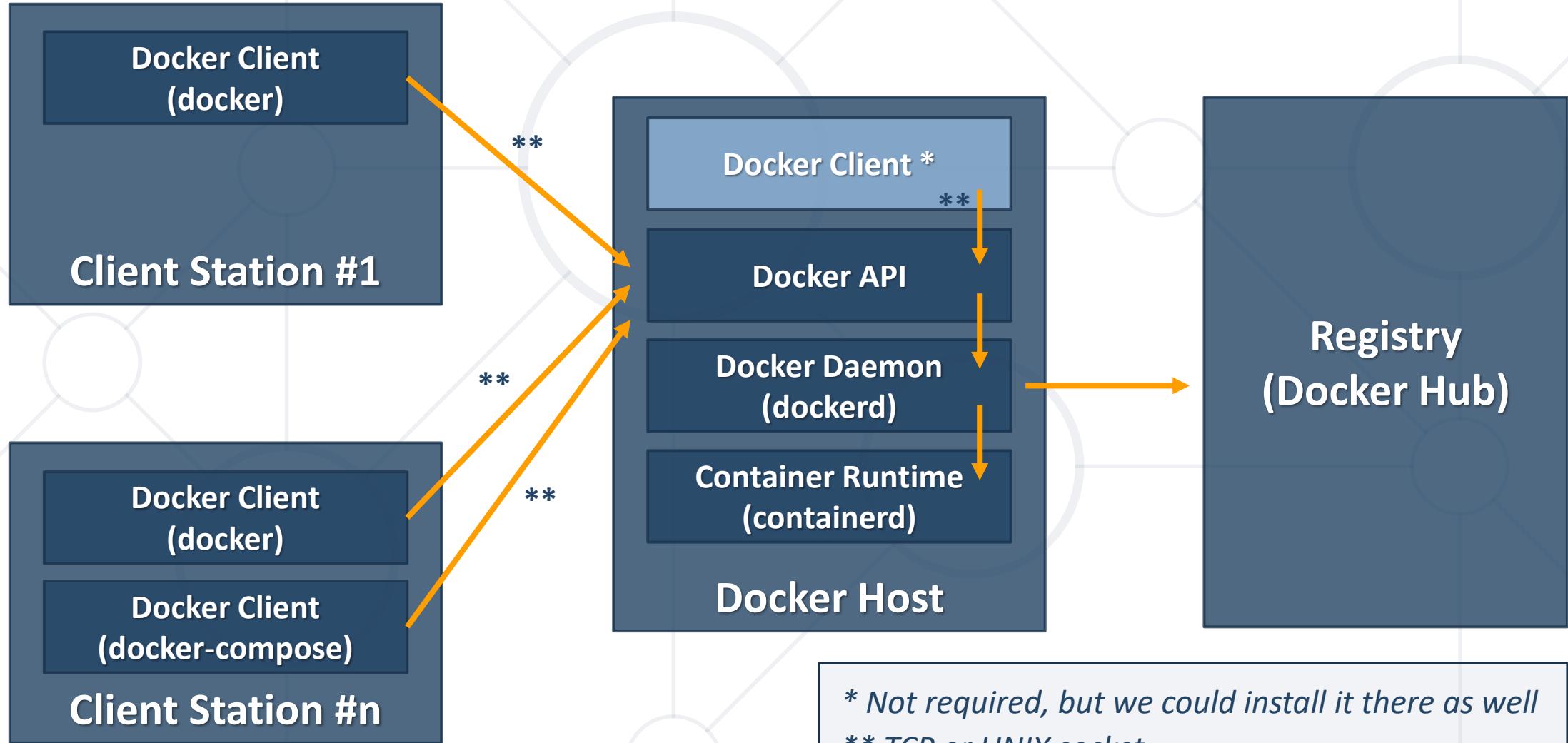


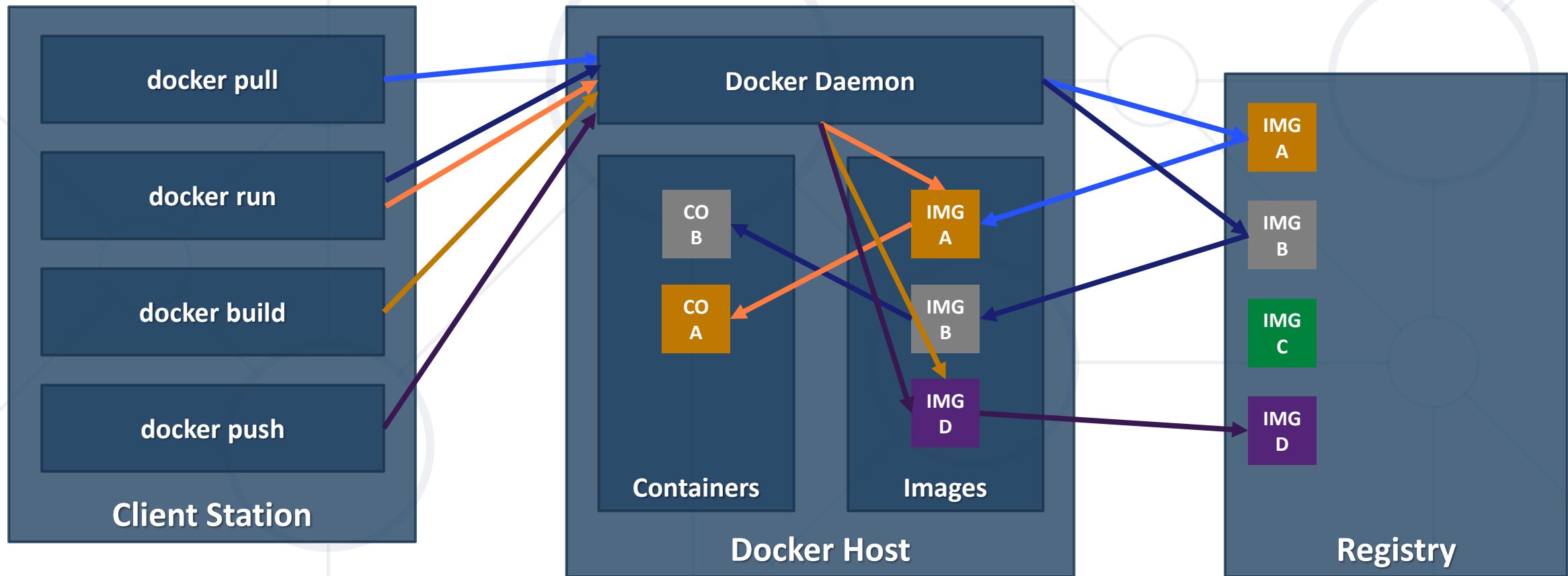
Image Layers



Docker Platform



Workflow



Installation Options

Docker Desktop

- Linux
- macOS
- Windows

Specific requirements: OS version, Hypervisor, etc.

Docker Engine

- Various Linux distributions
- Various hardware architectures

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

x86_64/amd64

arm64 (aarch64)

arm (armhf)

s390x

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

- 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
```

- 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
```

- 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
```

- 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
```

- 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
```

- 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
```

- 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
```

- 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
```

- 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
```

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

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

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

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

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

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

- 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

- 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

```
# 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
```

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"
```

- 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"
```

- 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 (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
```

- 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
```

EXPOSE

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

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

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

```
# shell form  
CMD command param1 param2
```

CMD vs ENTRYPOINT

- 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	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

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

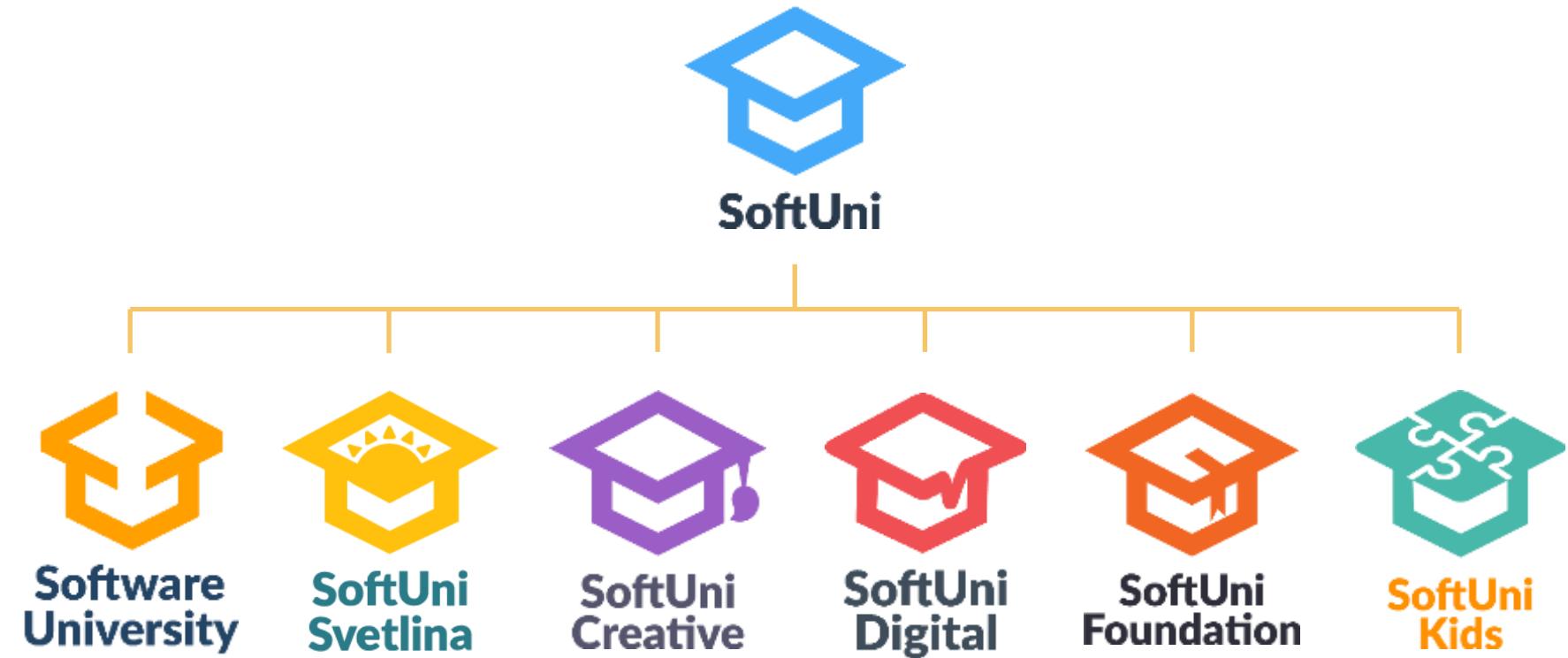
<https://docs.docker.com/docker-hub/>

Docker Registry Documentation

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



Questions?



SoftUni Diamond Partners



SCHWARZ



Coca-Cola HBC
Bulgaria



Postbank

Решения за твоето упре



Bosch..IO****



SmartIT



CAREERS



INDEAVR
Serving the high achievers



**SUPER
HOSTING
.BG**

Educational Partners



- 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, Profession 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

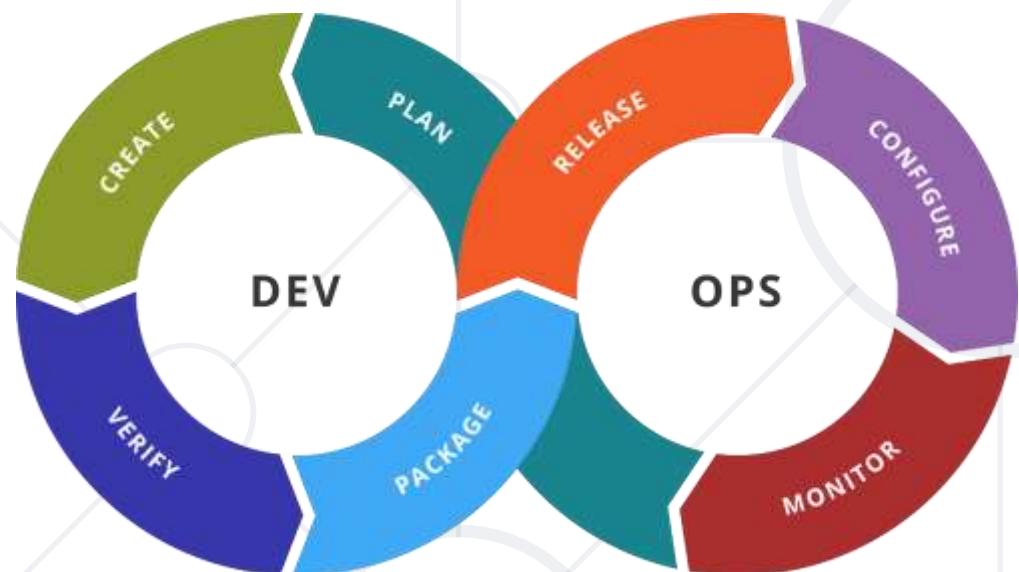


Software
University



Advanced Docker

Advanced Techniques. Distributed Applications. Clusters



SoftUni Team

Technical Trainers

 Software University



SoftUni



Software University

<https://softuni.bg>

You Have Questions?



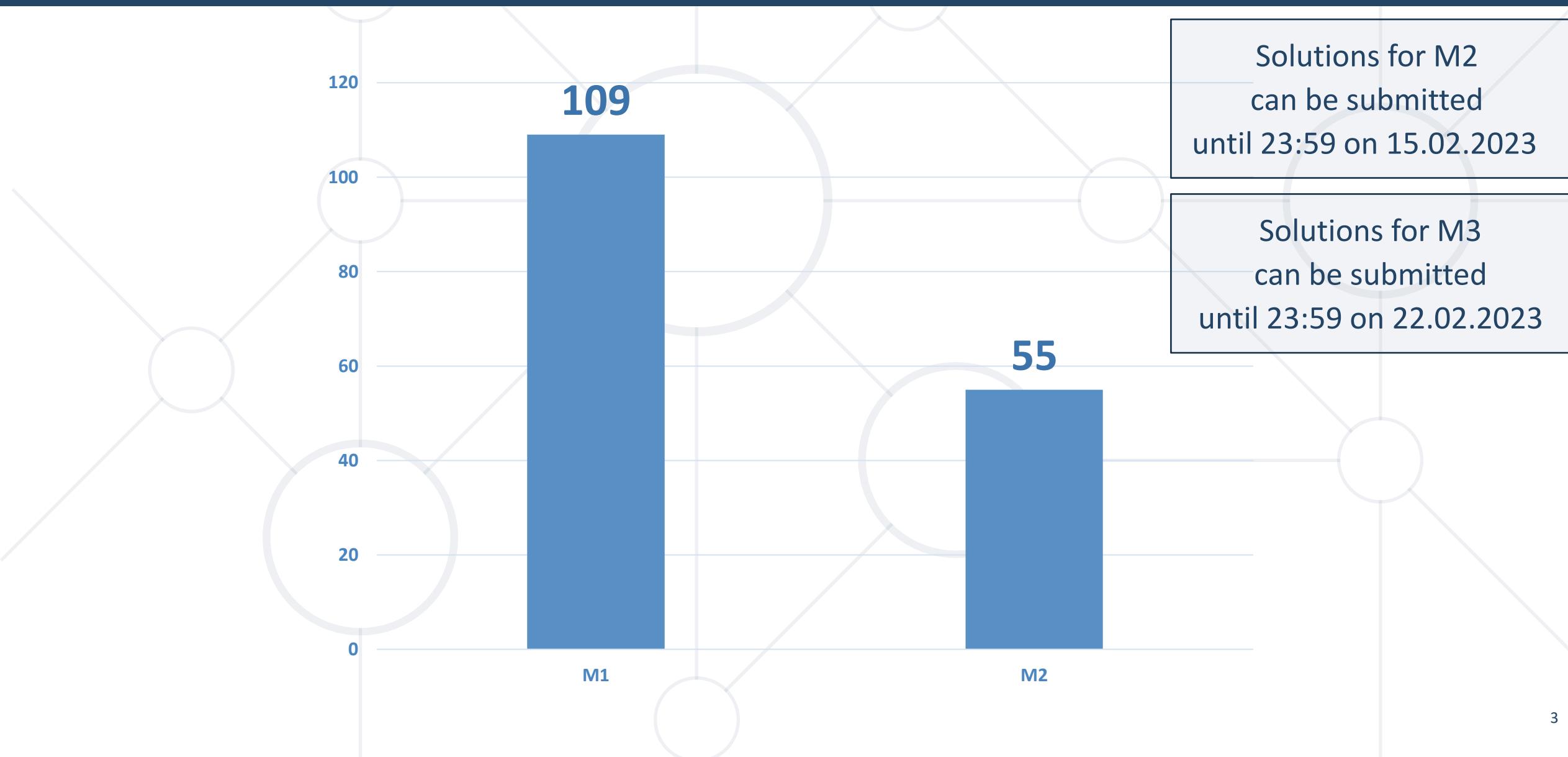
sli.do

#DevOps-23

facebook.com

/groups/DevOpsContainerizationCICDMonitoringJanuary2023

Homework Progress





Previous Module (M2)

Quick Overview

What We Covered

1. Containerization
2. Introduction to Docker
3. Docker in Action
4. Create Our Own Images



This Module (M3)
Topics and Lab Infrastructure

Table of Contents

1. Advanced techniques

- Networking
- Volumes

2. Distributed Applications

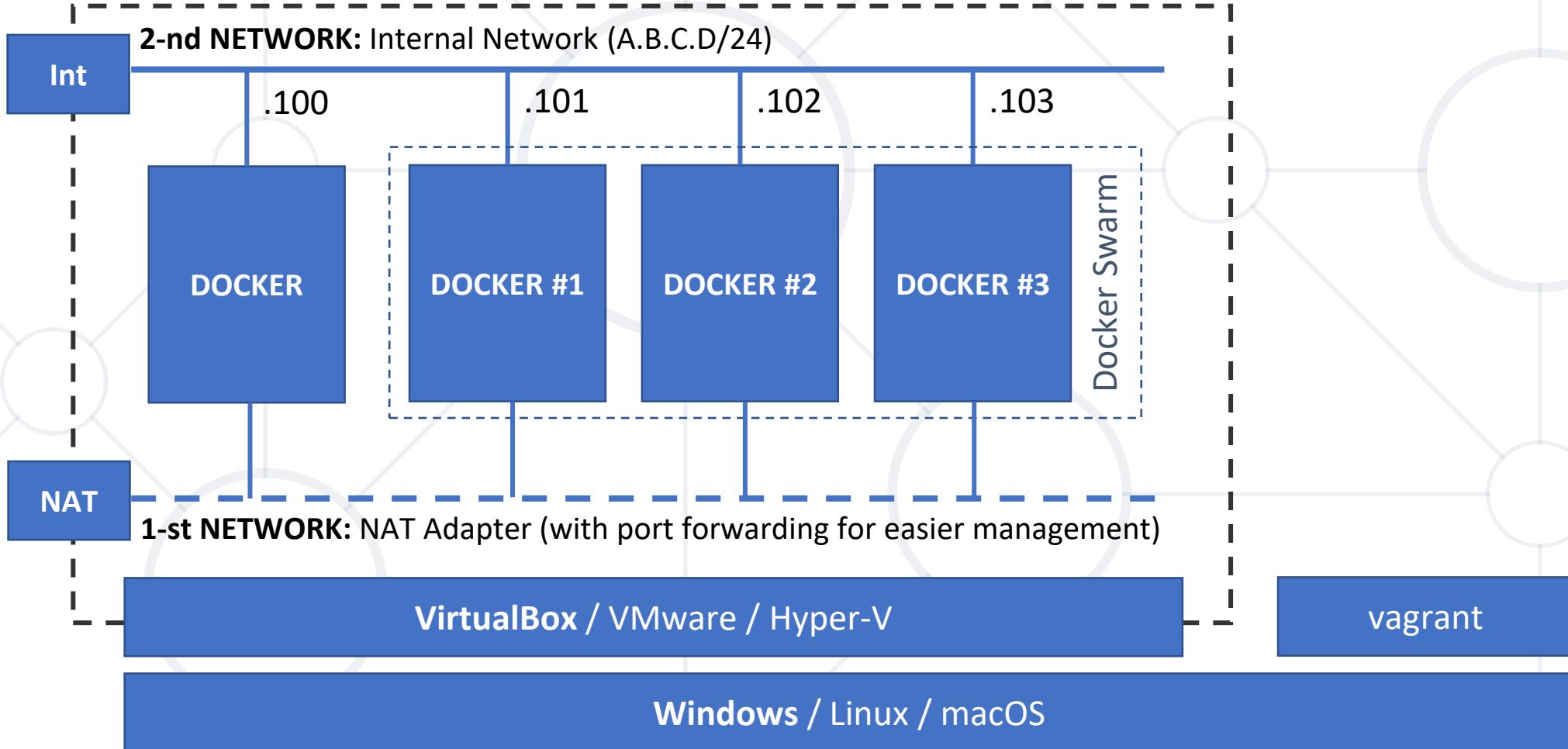
- Linking Methods
- Docker Compose

3. Docker Clusters

- Components and Principles
- Docker Swarm



Lab Infrastructure





Communication Networks: Overview and Usage

- Uses **pluggable drivers**. There is a **set of preinstalled drivers**
- **bridge** is the default driver. It allows containers connected to the same bridge to communicate while isolating them from the rest
- **host** uses the host's networking directly
- **overlay** connects multiple Docker daemons together and enables swarm services to communicate with each other
- **ipvlan** gives total control over both IPv4 and IPv6 addressing
- **macvlan** allows for assigning specific MAC addresses to containers
- **none** disables all networking for a container

Default Network

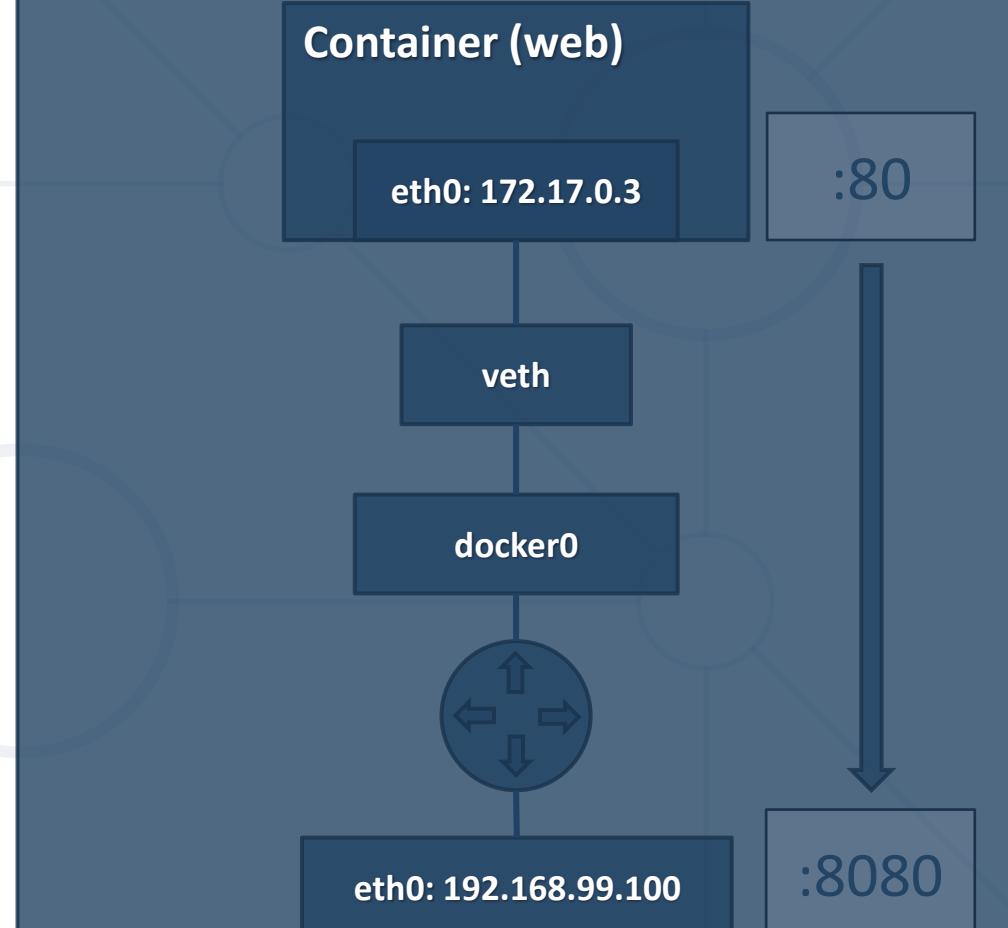
- Start a container with no explicit network settings

```
docker container run -d --name web \
  img-web
```

- We can expose a container port

```
docker container run -d --name web \
  -p 8080:80 img-web
```

Docker Host (docker)



Custom Network

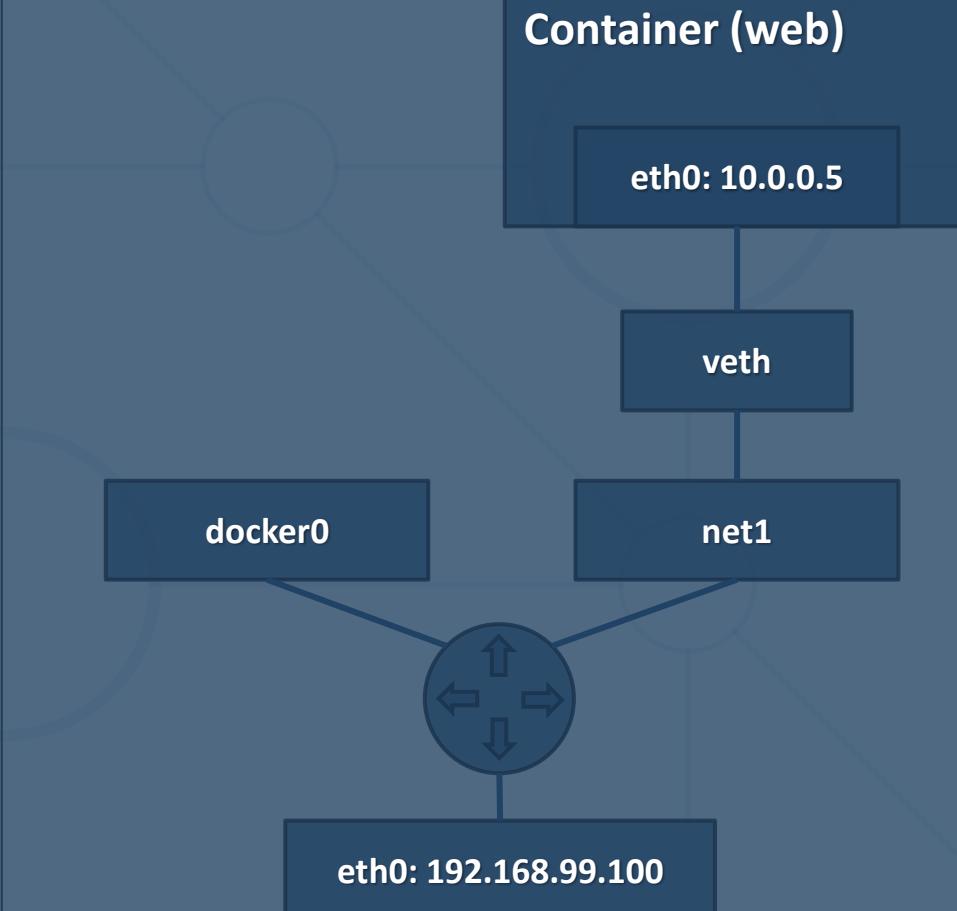
- Create a bridge network

```
docker network create -d bridge net1
```

- Start container connected to specific network

```
docker container run -d --name web \
--net net1 img-web
```

Docker Host (docker)



Two Networks

- Create a bridge network

```
docker network create -d bridge mynet
```

Start container connected to the default network

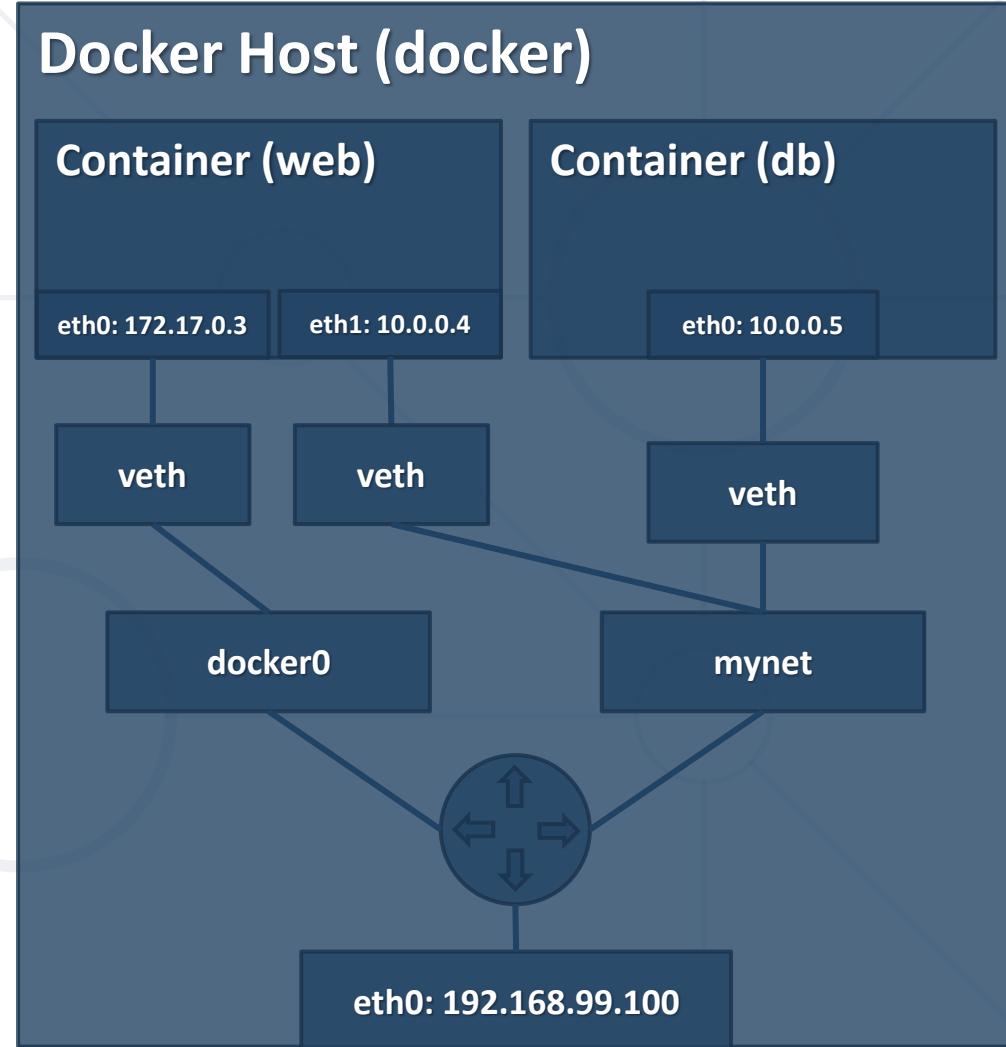
```
docker container run -d --name web \
    img-web
```

- Start container connected to specific network

```
docker container run -d --name db \
    --net mynet img-db
```

- Connect container to another network

```
docker network connect net1 web
```



- Purpose
 - List networks

- Syntax

```
docker network ls [options]
```

- Example

```
# list IDs of all networks
docker network ls -q
# list all networks that satisfy the filter
docker network ls -f driver=bridge
```

- Purpose
 - Display detailed information on one or more networks

- Syntax

```
docker network inspect [options] network [network]
```

- Example

```
# show network details
docker network inspect dob-network
```

- Purpose
 - Connect a container to a network

- Syntax

```
docker network connect [options] network container
```

- Example

```
# connect container to a network
docker network connect \
dob-bridge \
cont-001
```

- Purpose
 - Disconnect a container from a network

- Syntax

```
docker network disconnect [options] network container
```

- Example

```
# disconnect container from a network
docker network disconnect -f \
dob-bridge \
cont-001
```

network create

- Purpose
 - Create a network

- Syntax

```
docker network create [options] network
```

- Example

```
# create new bridge network
docker network create -d bridge \
  --subnet 10.0.0.1/24 \
  dob-bridge
```

- Purpose
 - Remove one or more networks

- Syntax

```
docker network rm network [network]
```

- Example

```
# remove networks net-1 and net-2
docker network rm net-1 net-2
```

- Purpose
 - Remove all unused networks

- Syntax

```
docker network prune [options]
```

- Example

```
# remove all unused networks without asking
docker network prune --force
# remove all network satisfying a filter
docker network prune --filter driver=bridge
```



Persistent Data

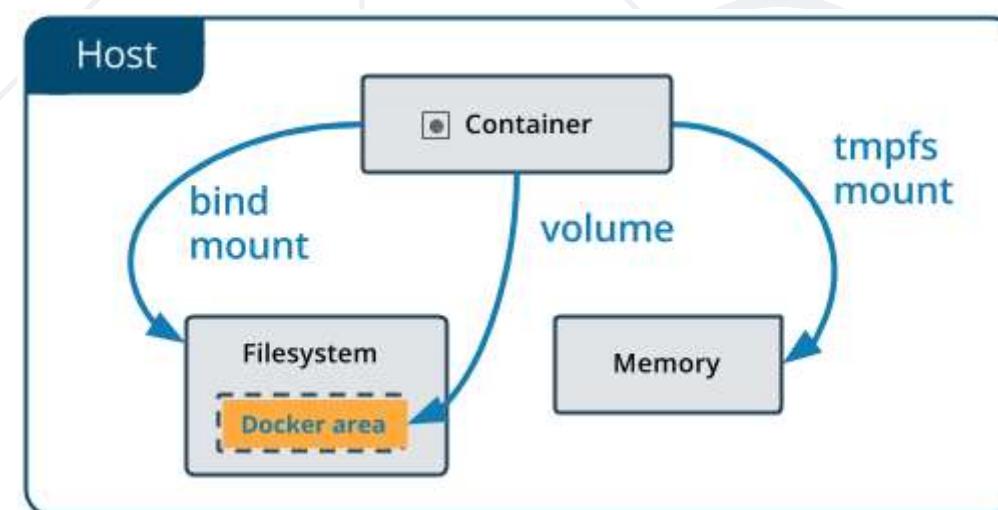
Volumes: Overview and Usage

Volume Overview

- Allow for external data in containers
- Two types
 - Data volumes
 - Data volume containers
- Created upfront, during run or build phase (`VOLUME` command)
- Data volumes can be shared
- Data volumes persist
- Data volumes are not deleted automatically

Volume Overview #2

- **Bind Mounts** are dependent on the OS and file system structure
- **Volumes** are managed by Docker
- **tmpfs mount** is for non-persistent state data
- **--volume (-v)** is simpler, and **--mount** is more explicit and verbose



volume ls

- Purpose
 - List volumes

- Syntax

```
docker volume ls [options]
```

- Example

```
# list IDs of all volumes
docker volume ls -q
# list all volumes satisfying a filter
docker volume ls --filter driver=local
```

- Purpose
 - Display detailed information on one or more volumes

- Syntax

```
docker volume inspect [options] volume [volume]
```

- Example

```
# show details about volume test-vol
docker volume inspect test-vol
```

- Purpose
 - Create a volume

- Syntax

```
docker volume create [options] [volume]
```

- Example

```
# create local volume test-vol
docker volume create test-vol
# create local volume lv-1 with label
docker volume create lv-1 --label mode=dev
```

- Purpose
 - Remove one or more volumes

- Syntax

```
docker volume rm [options] volume [volume]
```

- Example

```
# remove volume test-vol
docker volume rm test-vol
```

- Purpose
 - Remove all unused volumes

- Syntax

```
docker volume prune [options]
```

- Example

```
# remove all unused volumes without asking
docker volume prune -f
# remove all volumes satisfying a filter
docker volume prune --filter driver=local
```



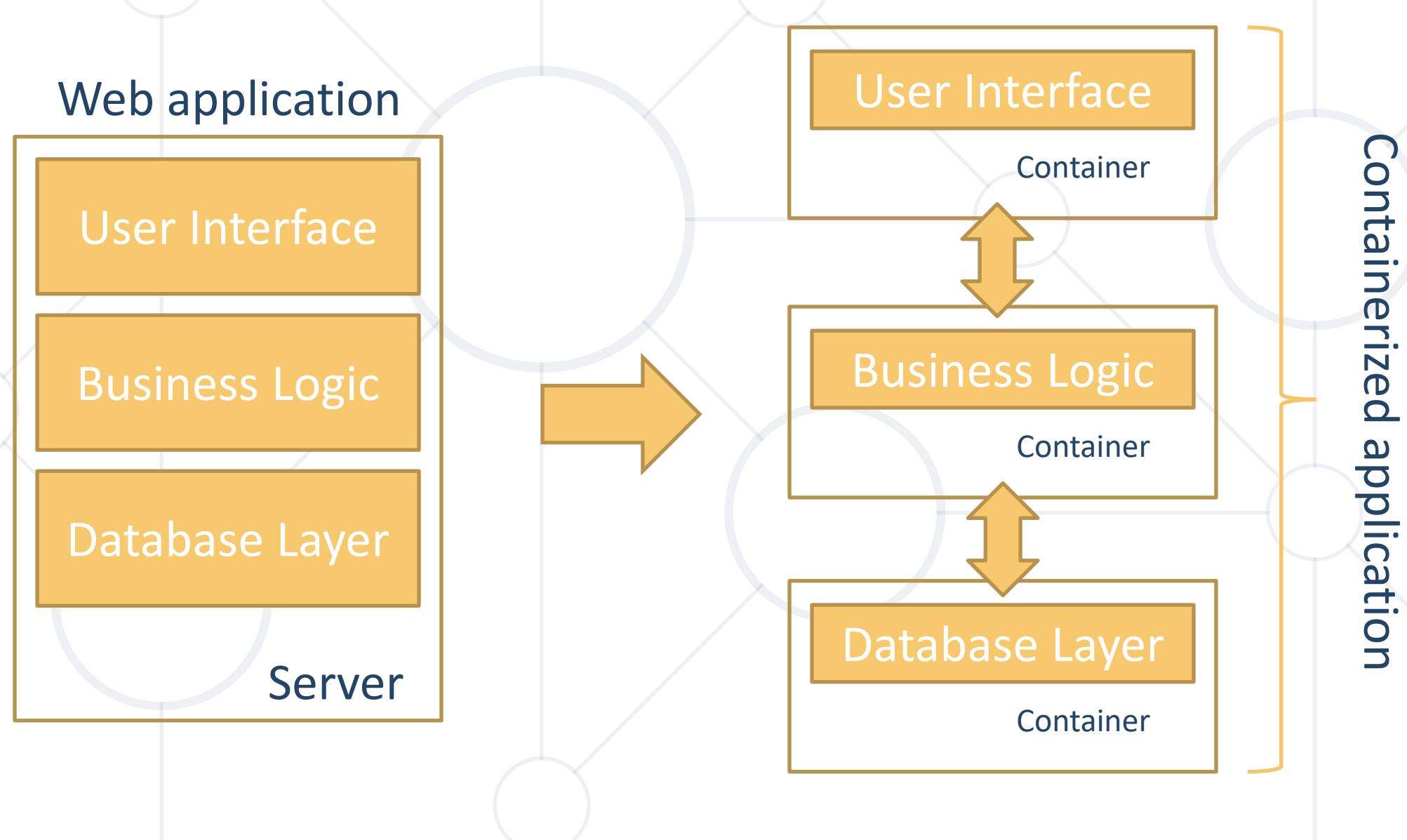
Practice: Networks & Volumes
Live Demonstration in Class



Distributed Applications

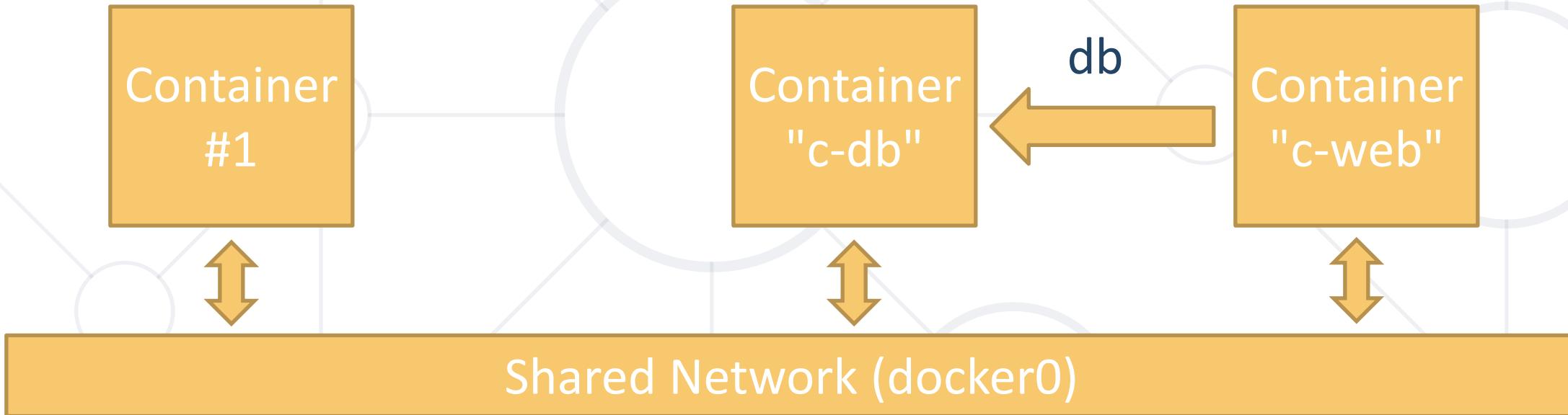
Overview and Implementation

Distributed Applications



Link Containers (Legacy) *

- By name alias



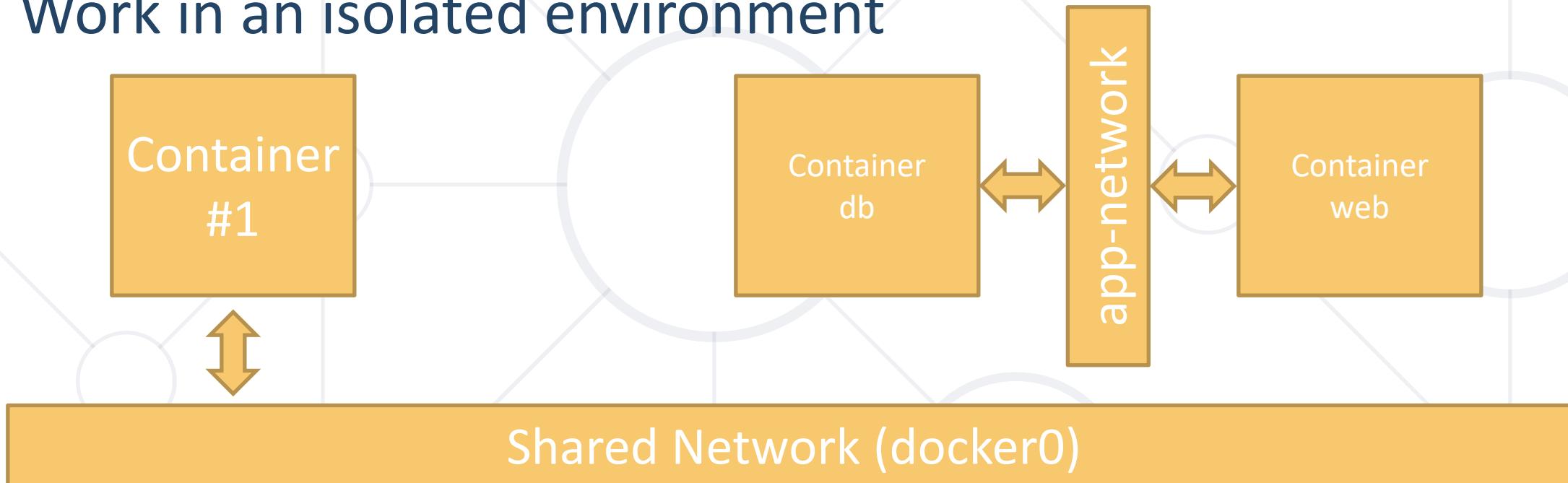
```
docker container run -d ... -p 8080:80 --link c-db:db ...
```

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

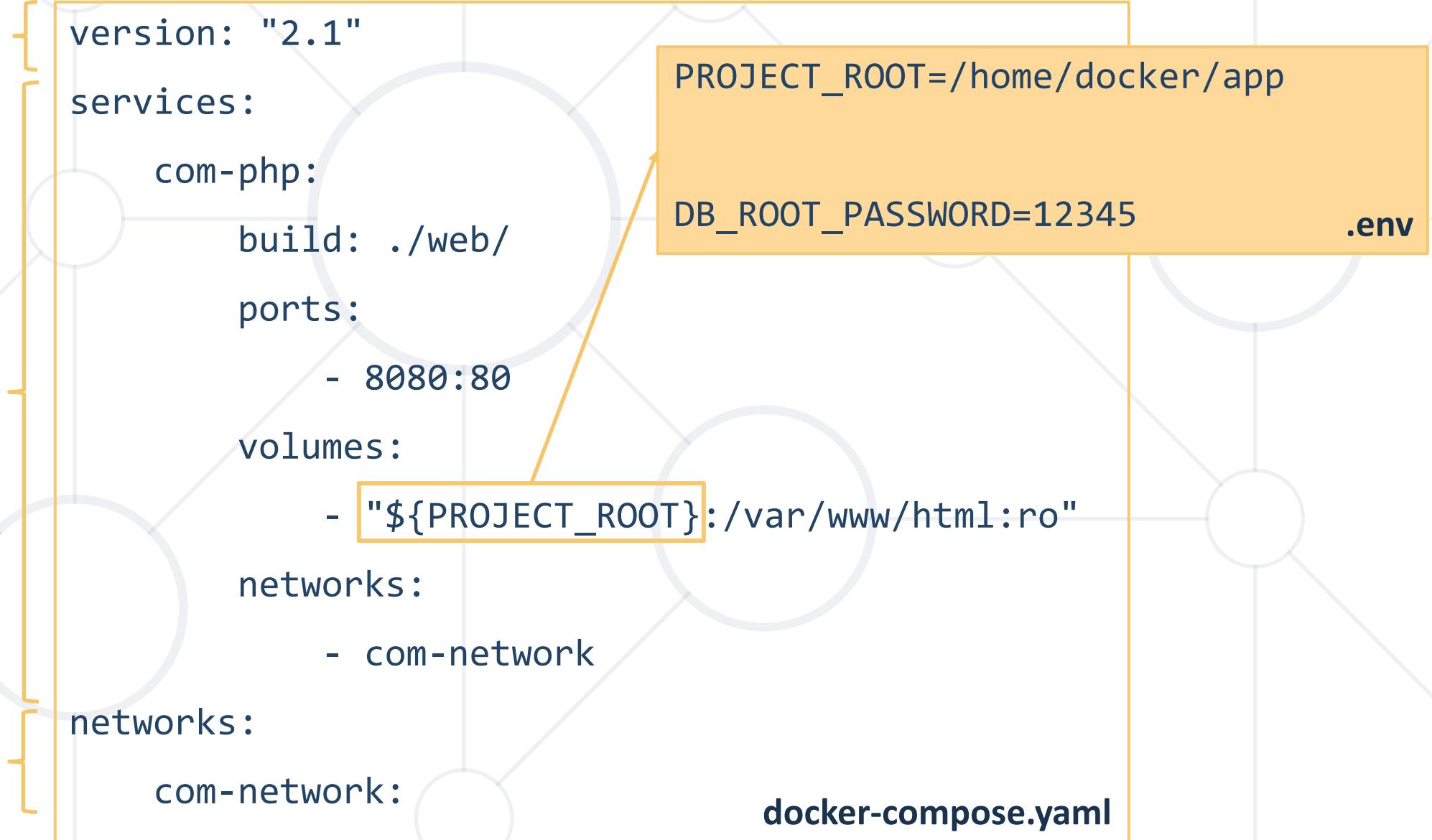
- Define and run **multi-container** Docker applications
- Multiple **isolated environments** on a single host
- **Preserve volume data** when containers are created
- Only recreate containers that **have changes**
- Supports **variables**
- Use cases
 - Development environments
 - Automated testing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

Configuration

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

Services Definition

Networks Definition



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-cache my-php
```

docker compose up

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

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

```
# start all containers and aggregate the output  
docker compose up  
# start all containers in a 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
```

- 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



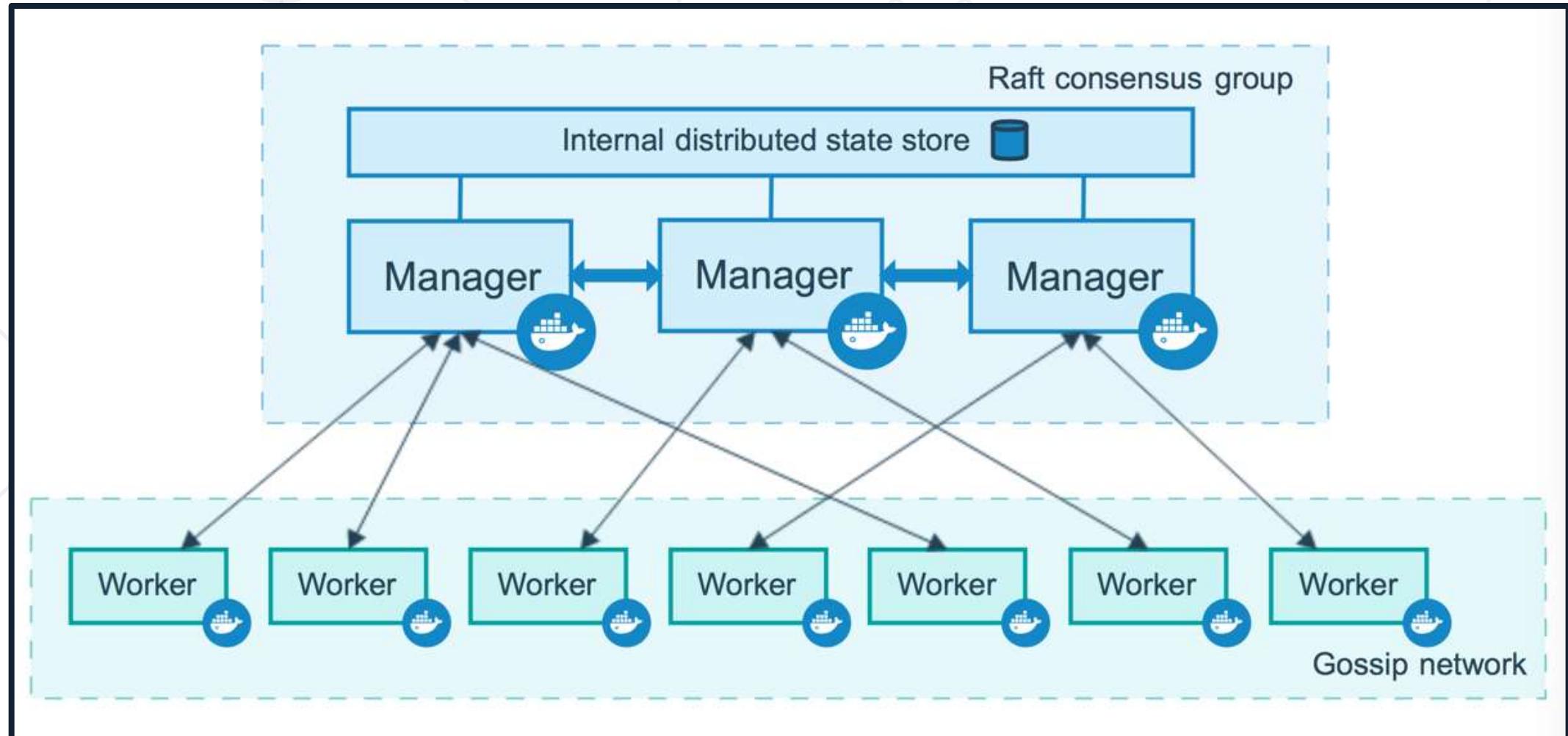
Docker Swarm

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

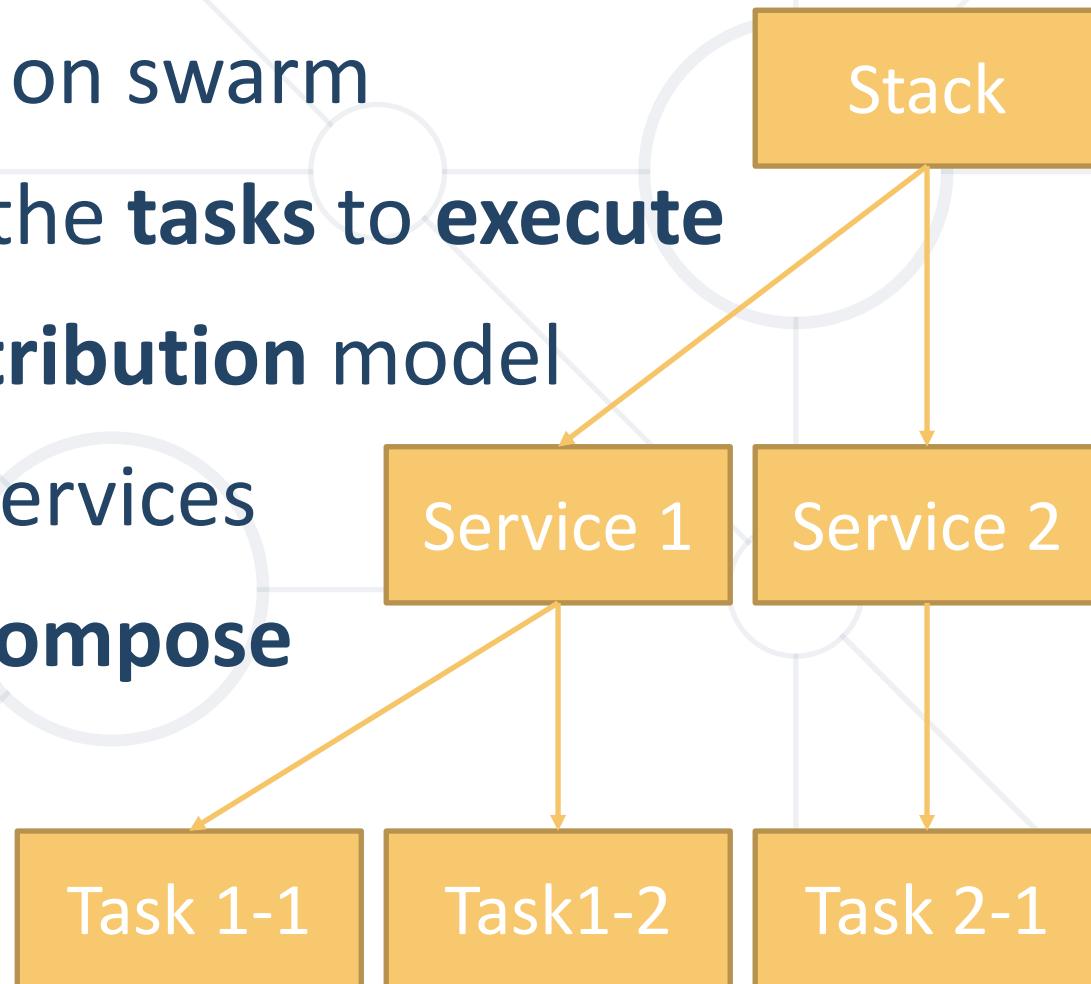
- Options
 - Cloud (Azure, AWS, ...)
 - On-premise - VM, Bare-metal
- Deployment Strategy (on-premise)
 - (Semi) Manual } Today's practice
 - Automated } Additional practice – homework ☺



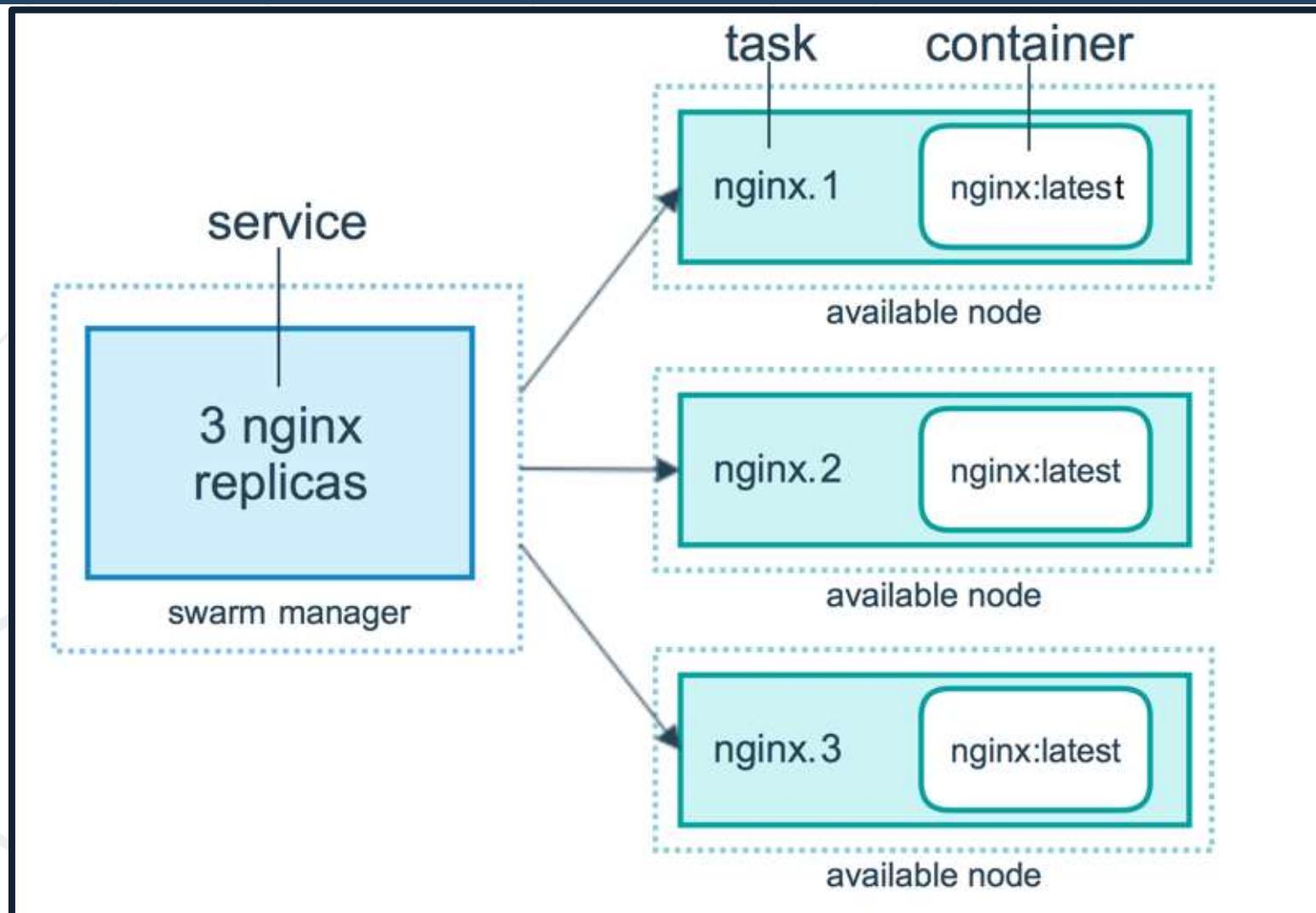
Stacks and Compose
Deployment Automation

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**

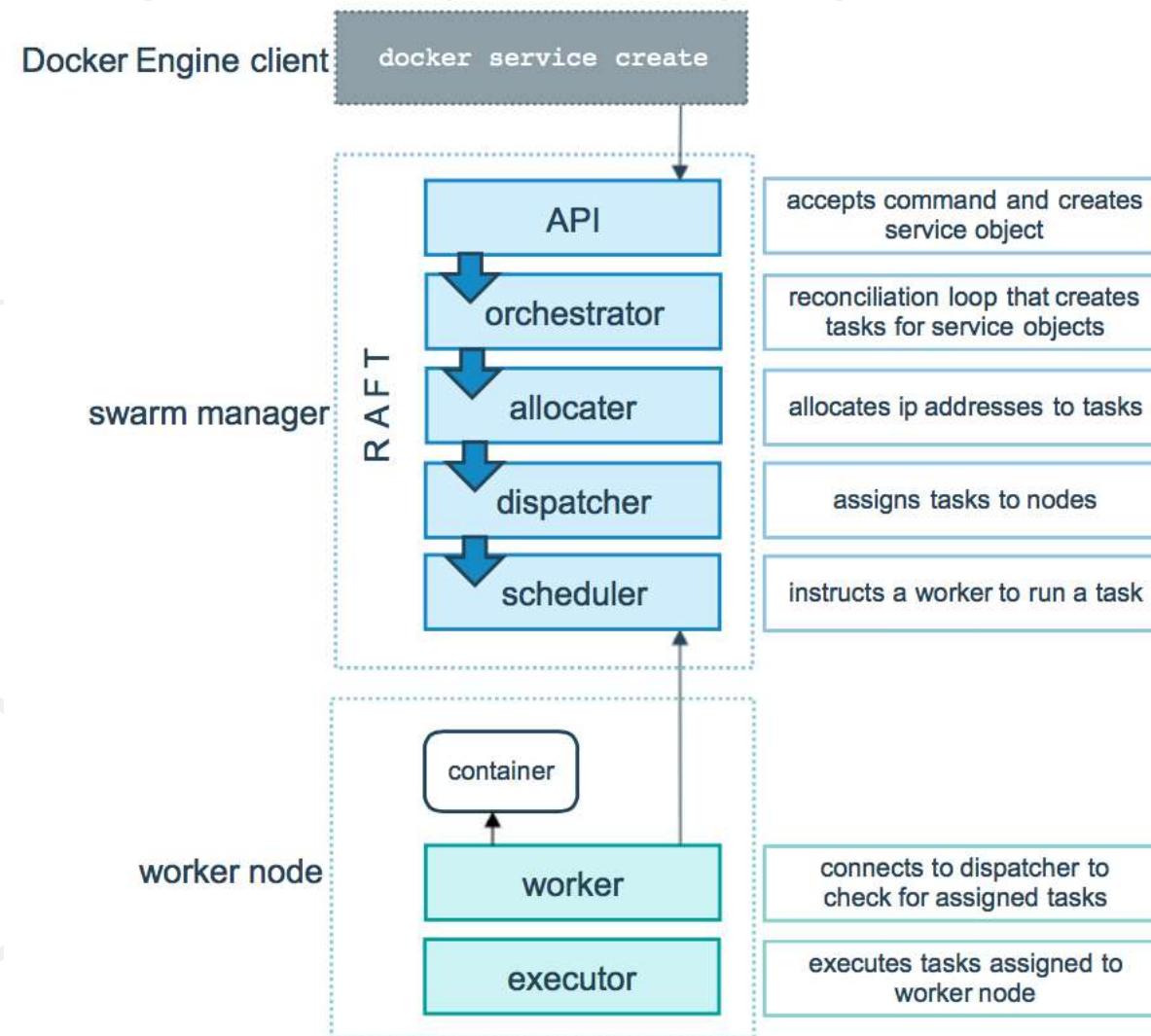


Containers, Tasks, and Services*



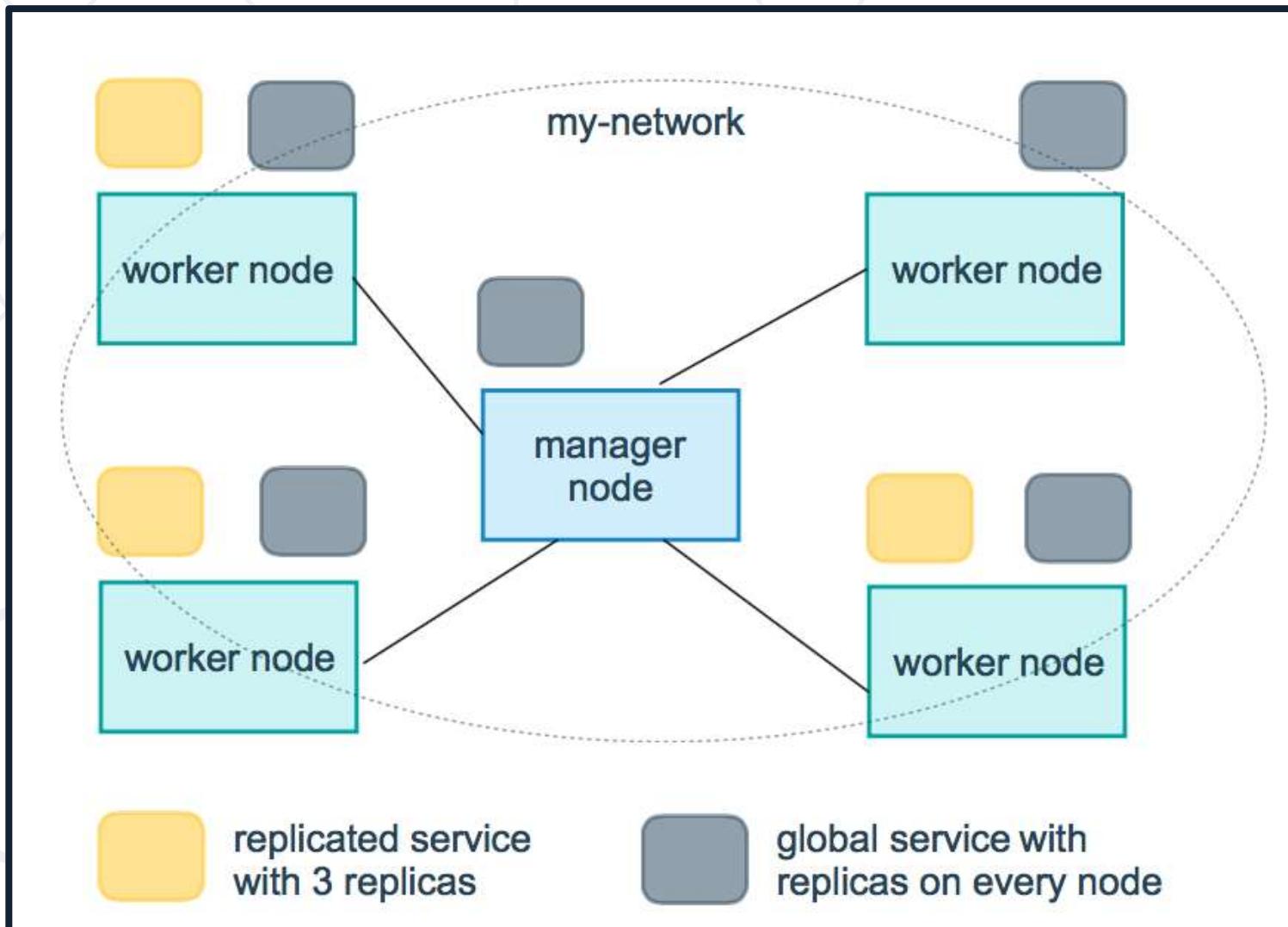
* <https://docs.docker.com/engine/swarm/how-swarm-mode-works/services/>

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

- 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

- 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**

- 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 can be added or removed at any time
 - Services can share a secret
 - Managed via separate set of commands
- docker secret ACTION [options]**
- Where ACTION is either **create**, **inspect**, **ls** or **rm**



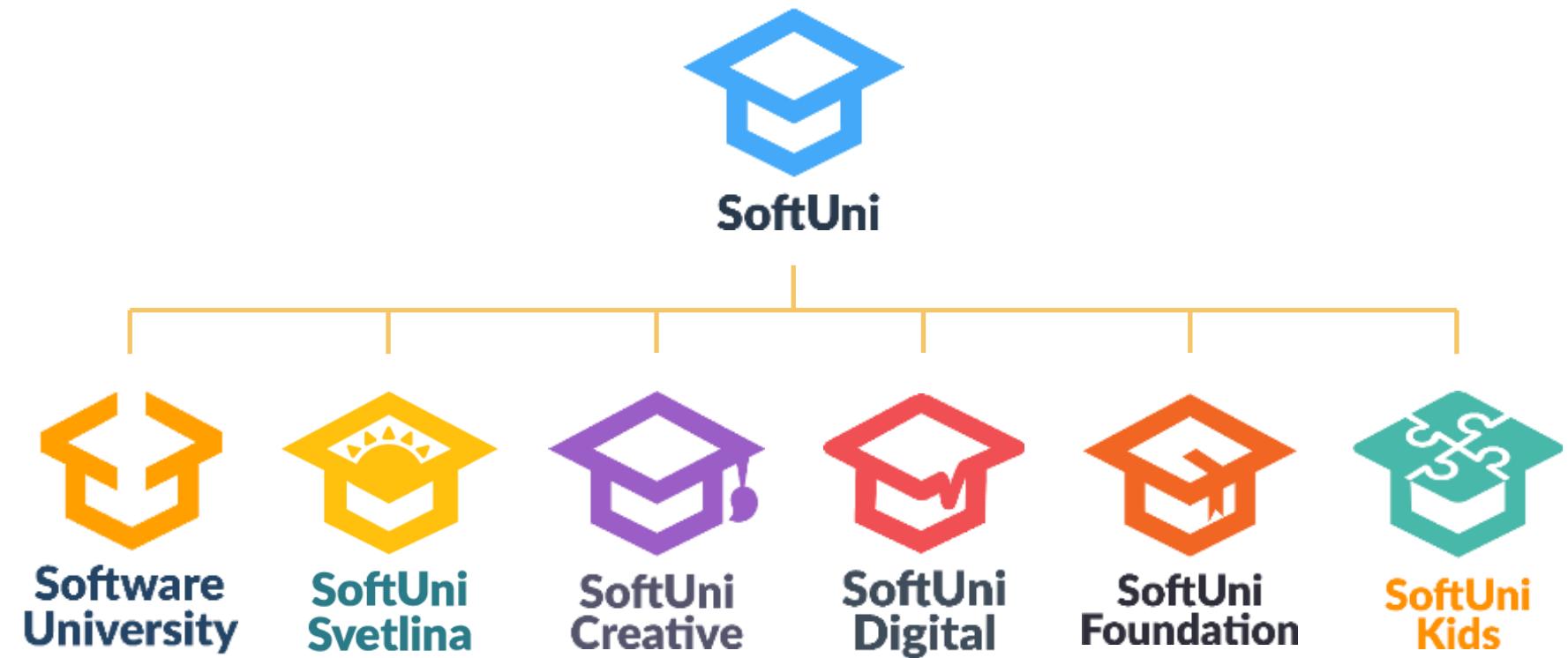
Practice: Swarm, Services and Stacks
Live Demonstration in Class

Summary

- Networking - inspect, tune, add, and remove
- Volumes - types, inspect and manage
- Distributed applications and Docker Compose
- Docker Swarm
 - How it works
 - Deployment options
 - Stacks and Compose



Questions?



SoftUni Diamond Partners



SCHWARZ



Coca-Cola HBC
Bulgaria



Postbank
Решения за твоето упре



Bosch..IO****



SmartIT



CAREERS



INDEAVR
Serving the high achievers



**SUPER
HOSTING
.BG**

Educational Partners



- 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, Profession 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



Software
University



Jenkins

Introduction and Basic Techniques



SoftUni Team

Technical Trainers



SoftUni



Software University

<https://softuni.bg>

You Have Questions?



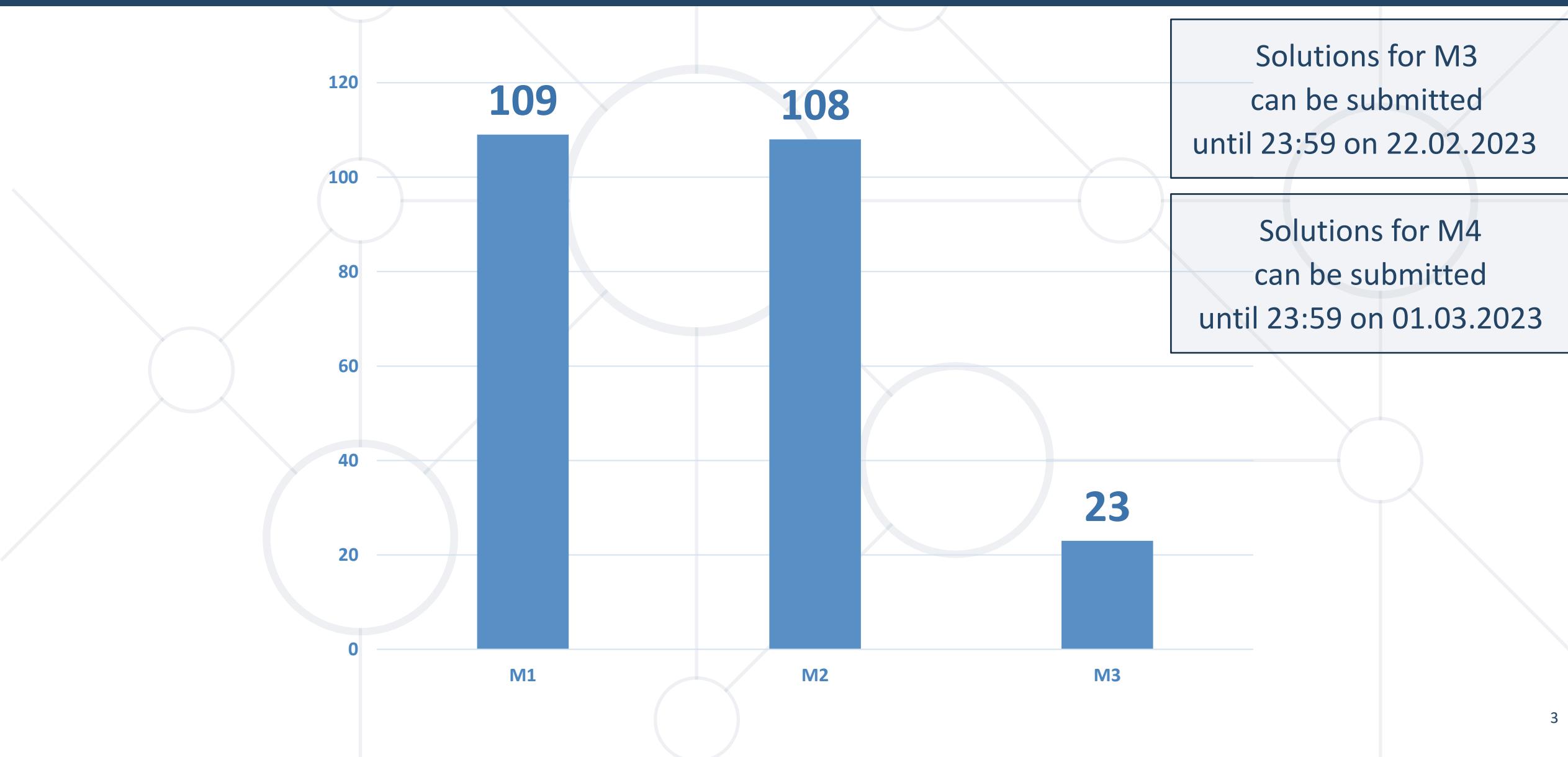
sli.do

#DevOps-23

facebook.com

/groups/DevOpsContainerizationCICDMonitoringJanuary2023

Homework Progress





Previous Module (M3) Quick Overview

What We Covered

- Advanced techniques
 - Networking
 - Volumes
- Distributed Applications
 - Linking Methods
 - Docker Compose
- Docker Clusters
 - Components and Principles
 - Docker Swarm



This Module (M4)
Topics and Lab Infrastructure

Table of Contents

1. Introduction to Jenkins

2. Working with Jenkins

- Remote Tasks
- Schedules
- Plugins

3. Advanced Jenkins

- Building with Docker
- Pipelines



Lab Infrastructure

NETWORK: 192.168.99.0/24

.100

.101

VM1
Jenkins

VM2
Docker

VirtualBox

Vagrant

Windows / Linux / macOS



**Available Solutions
For Continuous Integration/Deployment/Delivery**

- **Speed**
 - Build frequently and faster
- **Automation**
 - Build and deploy in an automated fashion
- **Predictable results**
 - Test each step and deploy only if everything is okay
- **Faster time-to-market**
 - Possibility to deliver to production at any time

Solutions (1)

- **Buildbot**
 - <http://buildbot.net/>
 - Free
- **TeamCity**
 - <https://www.jetbrains.com/teamcity/>
 - Paid and Free. Cloud and On-Premise
- **Bamboo**
 - <https://www.atlassian.com/software/bamboo>
 - Only paid

Solutions (2)

- **CircleCI**
 - <https://circleci.com/>
 - Paid (offers basic free tier)
- **Travis CI**
 - <https://travis-ci.com/>
 - Free (and Paid)
- **Jenkins (Backed by CloudBees)**
 - <https://jenkins.io/>
 - Free (and Paid)

GitHub Actions

<https://github.com/features/actions>

Paid and free tier

Main Definitions

- Continuous Delivery and Integration
 - CD is the ability to release at any time
 - CI is practice of integrating or merging code changes frequently
- Stages of CD and CI
 - Build => Deploy => Test => Release

Continuous Delivery

- Software **can be** released to production at any time
- Every change **can go** to production
- I **could be** deploying constantly

Continuous Deployment

- Software **is** released to production as part of an automated pipeline
- Every change **goes** to production
- I **am** deploying constantly





Introduction to Jenkins

Introduction. Components. Installation

What is Jenkins?

- An open source automation server
- A platform for the Software Development Life Cycle (SDLC)
- Typically used to implement CI/CD
- Easy to use and highly adaptable
- Extensible and customizable
- Works on most common operating systems
- Considered lightweight

Key Definitions

- **Job**
 - Configured task in Jenkins. It is an old term
- **Project**
 - A task configured in Jenkins. It is the current term
- **Pipeline**
 - Special type of job created by a Pipeline plugin

Requirements and Installation

- Requirements
 - Works on Unix / Linux / Mac / Windows
 - Requires Java 11 or 17
- Installation
 - Can be installed as a Native Service, Container, Java application
 - Can be installed from source or through package system

Current version
2.391 / 2.375.3





Practice: Installation. Environment Setup
Live Demonstration in Class



Basic Activities
Remote Projects. Plugins. Schedules

Project Builds

- Local and Remote
- sudo allowance and settings
- Remote credentials
 - User and Password, Key file
- Firewall and SSH settings

- Default (minimal) set of plugins installed
- Over 1800 plugins available at <https://plugins.jenkins.io/>
- Grouped in categories by platform, purpose, etc.
- Dedicated Plugin Manager
- Automated or Manual installation

Scheduled Builds

- With plugin or native
- Two options
 - Execute once at a specific point in time
 - Regular execution
- Execution interval is set with a cron like syntax

GitHub Integration

- Offered through a plugin
- Installed by default
- It works both with local git installation and GitHub
- Local git client is required
- Can receive notifications from git
- Can check periodically for changes (should be avoided)



Practice: See It in Action
Live Demonstration in Class



Advanced Jenkins #1

Docker Integration. Pipelines

Docker Integration

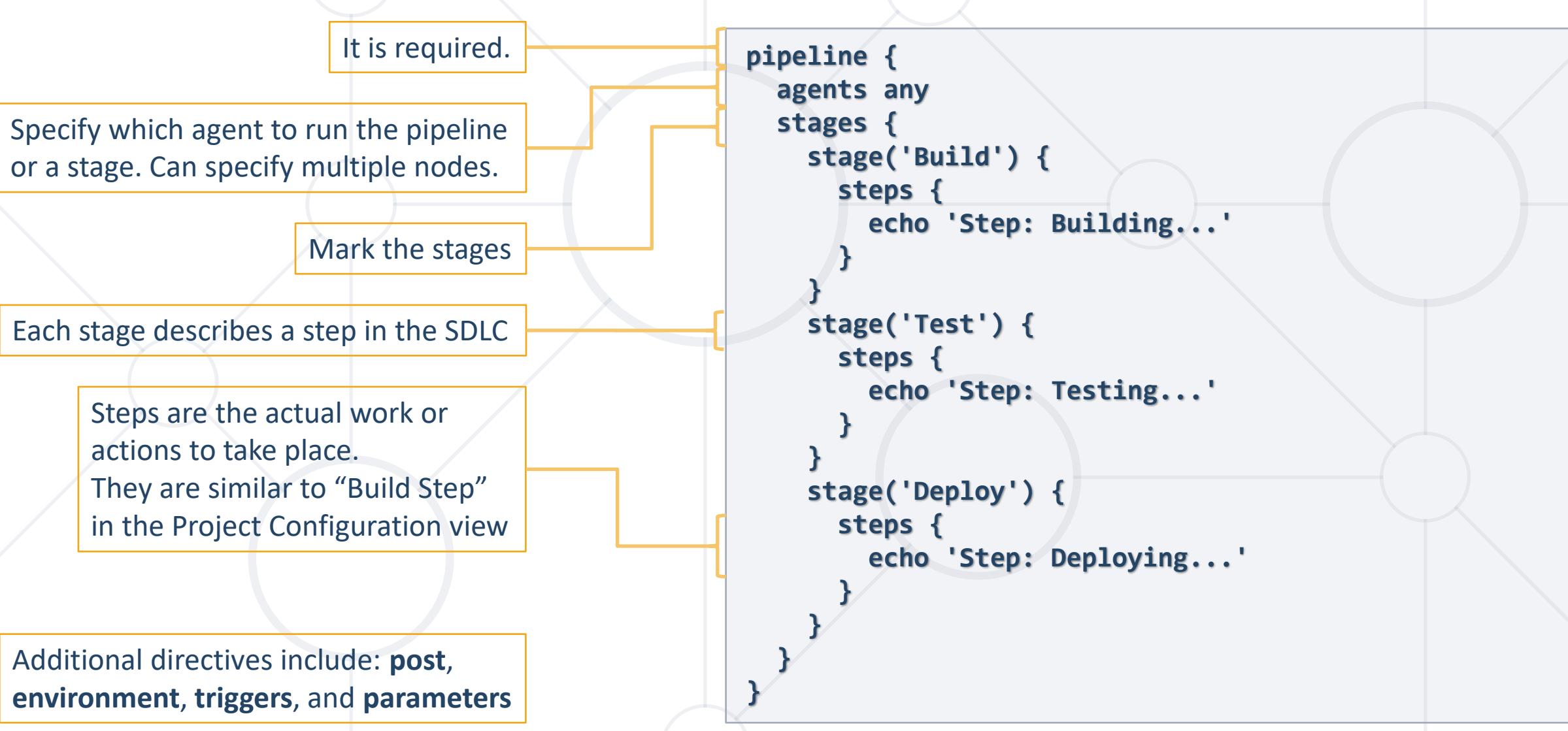
- Integration can be done through plugin or directly
- Docker client is required
- Docker client should be present on all nodes where we plan to run Docker related tasks

Pipelines

- Old name is workflows
- Used for long running activity orchestration
- Can span on multiple nodes (slaves)
- It is a suite of plugins
- “Pipeline as code” is defined with a **Jenkinsfile**

- It is used to define continuous delivery pipeline
- Stored together with our code
- Two styles are supported – declarative and scripted
- It is written in Groovy

Declarative Jenkinsfile





Practice: Slave Nodes and Docker
Live Demonstration in Class

Summary

- Jenkins is an open source automation server
- It could be installed in many ways
- It is extendable through plugins
- It is scalable – additional slave nodes could be added
- There is integration with source control systems
- Jenkins offers integration with Docker
- Jenkins could be managed with systems like Ansible, Chef, and etc.



Resources

Jenkins site

<https://jenkins.io/>

Jenkins installation guide

<https://jenkins.io/doc/book/installing/>

Jenkins Pipeline guide

<https://jenkins.io/doc/book/pipeline/>

Managing Jenkins

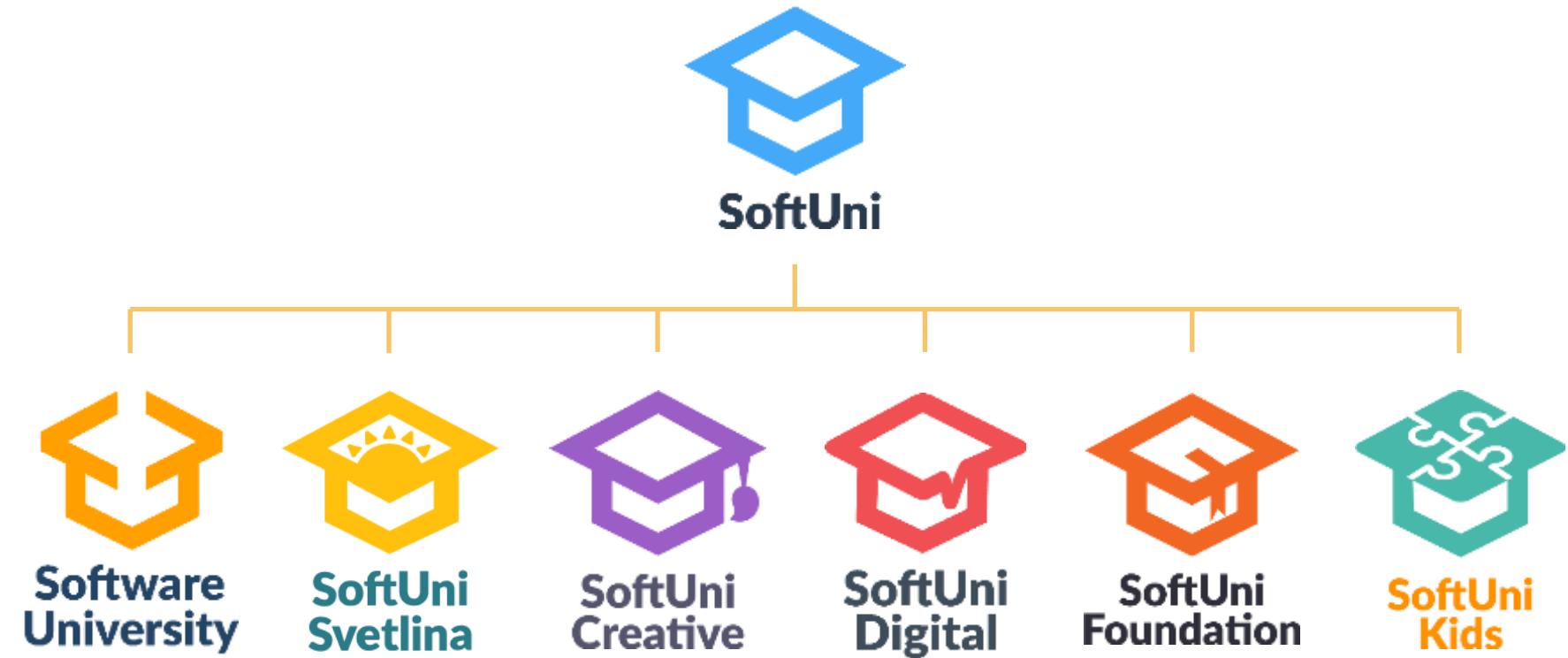
<https://jenkins.io/doc/book/managing/>

Jenkins plugins

<https://plugins.jenkins.io/>



Questions?



SoftUni Diamond Partners



SCHWARZ



Coca-Cola HBC
Bulgaria



Postbank

Решения за твоето упре



Bosch..IO****



SmartIT



CAREERS



INDEAVR
Serving the high achievers



**SUPER
HOSTING
.BG**

Educational Partners



- 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, Profession 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

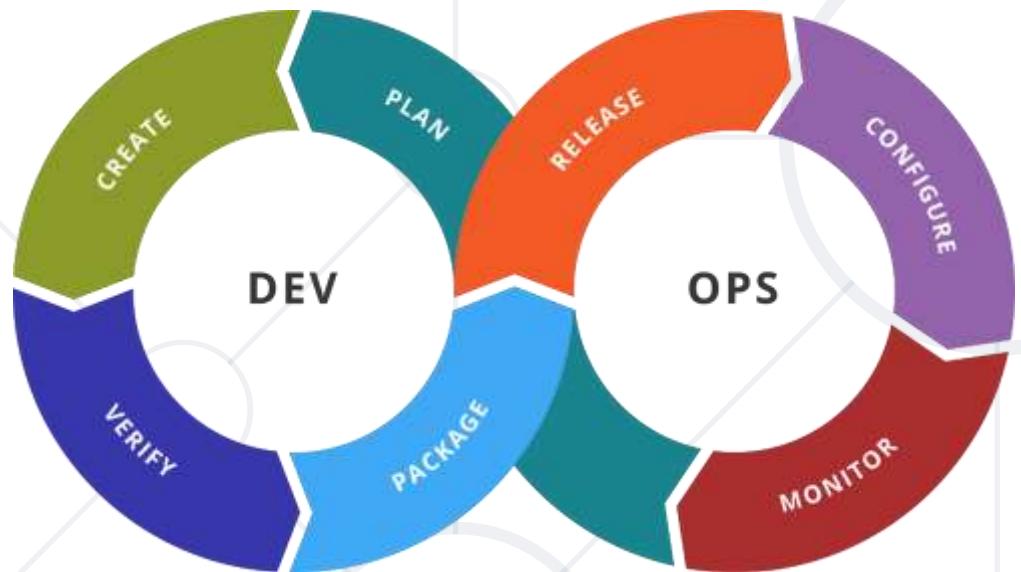


Software
University



Jenkins

Advanced Techniques



SoftUni Team

Technical Trainers

 Software University



SoftUni



Software University
<https://softuni.org>

You Have Questions?



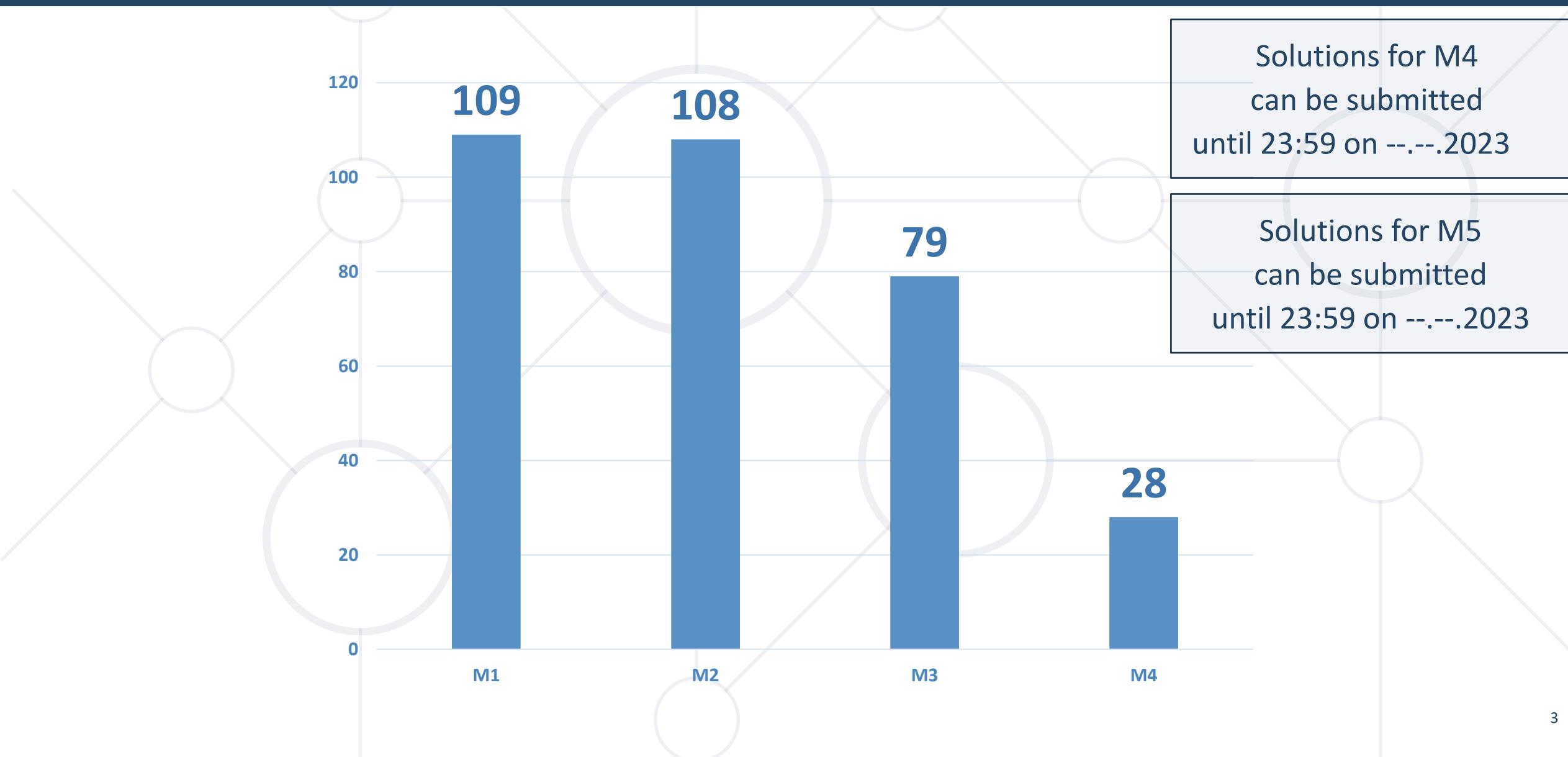
sli.do

#DevOps-23

facebook.com

/groups/DevOpsContainerizationCICDMonitoringJanuary2023

Homework Progress





Previous Module (M4)

Quick Overview

What We Covered

- Introduction to Jenkins
- Working with Jenkins
 - Remote Tasks
 - Schedules
 - Plugins
- Advanced Jenkins
 - Building with Docker
 - Pipelines



This Module (M5)
Topics and Lab Infrastructure

Table of Contents

1. Remote (slave) hosts
2. Working with Jenkins using CLI
3. Export and Import jobs
4. Blue Ocean
5. SCM integration



Lab Infrastructure

NETWORK: 192.168.99.0/24

.101

.102

VM1
Jenkins

VM2
Docker

VirtualBox

Vagrant

Windows / Linux / macOS



**Offload Jenkins Host
Using remote (slave) hosts**

Why Offloading?

- It is not considered good practice to execute jobs on the host
- Instead, we can register nodes and use them as executors
- We can mix and match platforms and environments
- Furthermore, we can have on-premises and cloud hosts
- We can even have virtual machines and physical machines
- This will increase not only the variety but also the capacity

Procedure

- Credentials and keys setup
- Register a node
- Assign label(s)
- Configure # of executors, workspace, etc.
- Deploy an agent
- Start using it



Practice: See It in Action
Live Demonstration in Class



Control Jenkins Using Command Line Interface (CLI)

- Users and administrators can use script or shell environment
- It allows access over SSH
- Or via downloadable CLI client
- We can use either HTTP, SSH or WebSocket to work with the CLI
- In addition, there is a Groovy-based script console

Partial List of Commands

- **help** can be used to get the list of all commands or information about specific command
- **list-jobs** returns a list of all available jobs
- **build** allows for job execution
- **create-job** is used to create jobs out of XML
- **get-job** returns a job represented in XML
- **copy-job** allows to duplicate a job
- **delete-job** can be used to delete jobs permanently



Jobs Management
Export / Import

Jobs Management

- We can export job from one instance and import it on another
- The same can be applied on one and the same instance
- This technique is useful for backup purposes as well
- Last but not least, it allows for automation or zero touch deployment
- Relies on the command line tools
- There are specialized plugins as well



Pipelines In Focus
Blue Ocean

- Set of plugins focused on (multibranch) pipelines
- Completely different interface and tooling
- Visual pipeline editor
- Visual pipeline representation
- GitHub & Git integration
- Instant problem diagnosis



Practice: See It in Action
Live Demonstration in Class



**SCM Integration
Pipeline Hosting and Webhooks**

Pipeline SCM Integration

- Complex pipelines are difficult to maintain in Jenkins classic UI
- Instead, we can work in IDE of our choice and use an SCM
- We store the pipeline code in **Jenkinsfile**
- During the build, Jenkins automatically checks out the file
- During job creation we point out where the **Jenkinsfile** is
- We can use the snippet generator that can be found here
 `${YOUR_JENKINS_URL}/pipeline-syntax`

- No matter where the pipeline code is, we can use webhooks to link the build process with changes in the SCM
- There are different plugins that expose endpoints in Jenkins
- Some of them are specialized and others are more generic
- They are called from the SCM when there is a push for example
- Some allow greater control like filtering for example



Practice: See It in Action
Live Demonstration in Class

Resources

Jenkins site

<https://jenkins.io/>

Jenkins installation guide

<https://jenkins.io/doc/book/installing/>

Jenkins Pipeline guide

<https://jenkins.io/doc/book/pipeline/>

Managing Jenkins

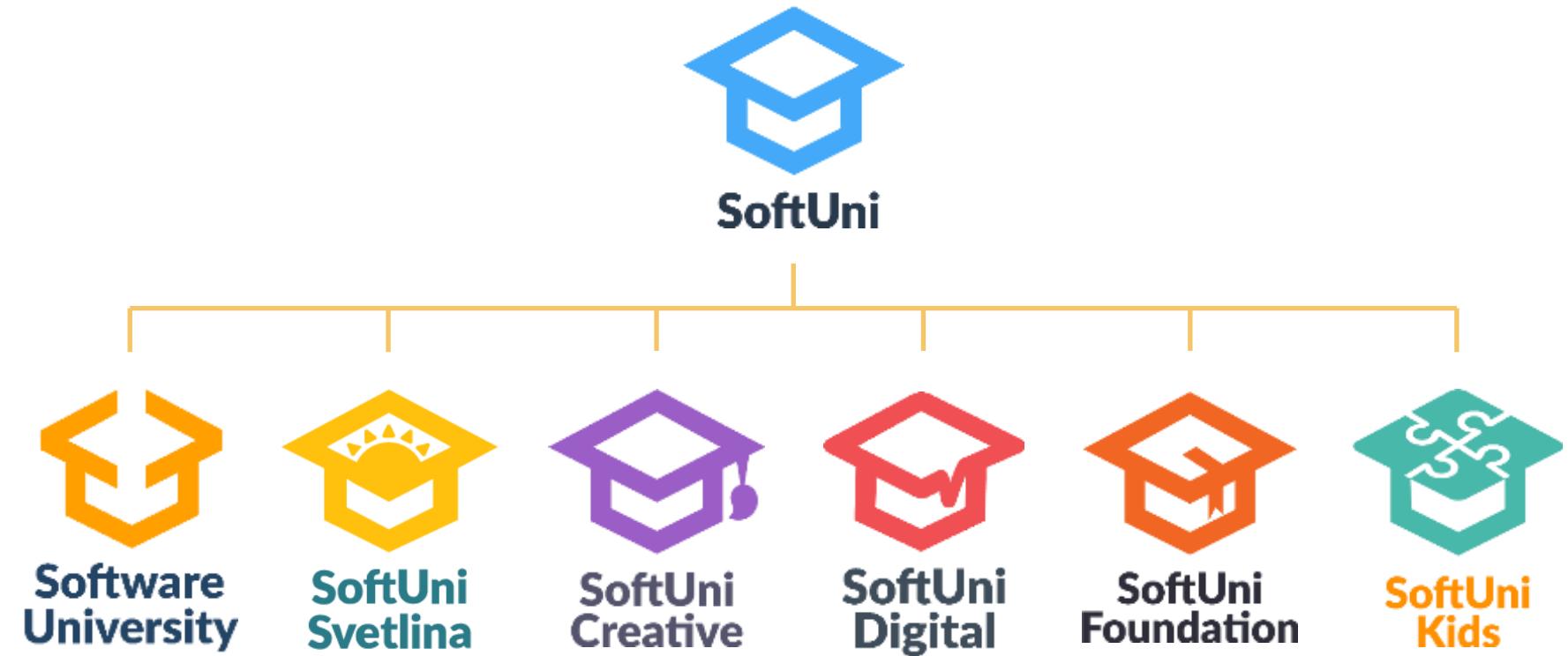
<https://jenkins.io/doc/book/managing/>

Jenkins plugins

<https://plugins.jenkins.io/>



Questions?



SoftUni Diamond Partners



SCHWARZ



Coca-Cola HBC
Bulgaria



Postbank

Решения за твоето упре



Bosch..IO****



SmartIT



POKERSTARS



CAREERS



AMBITIONED



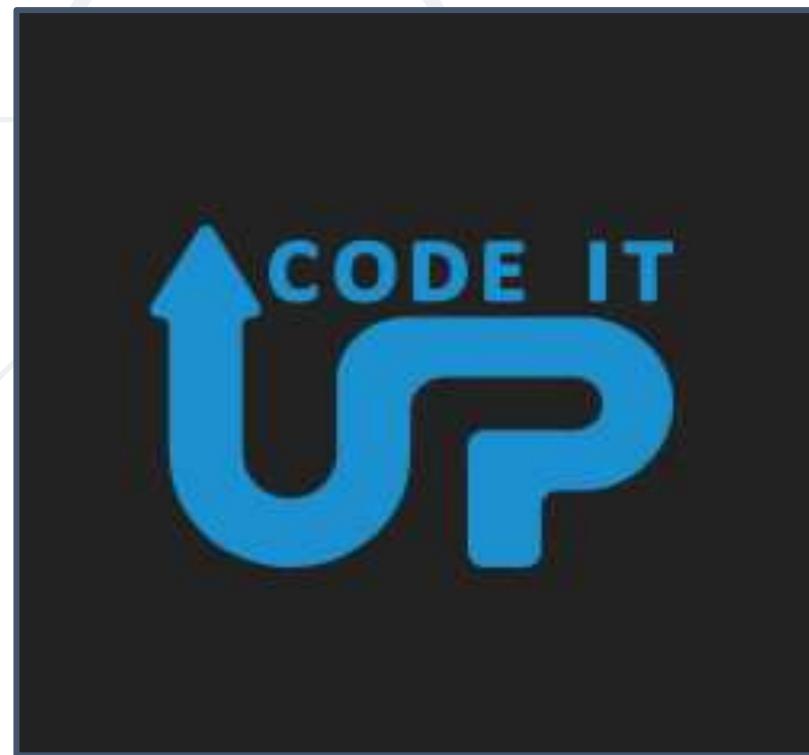
INDEAVR
Serving the high achievers



createX

**SUPER
HOSTING
.BG**

Educational Partners



- 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)



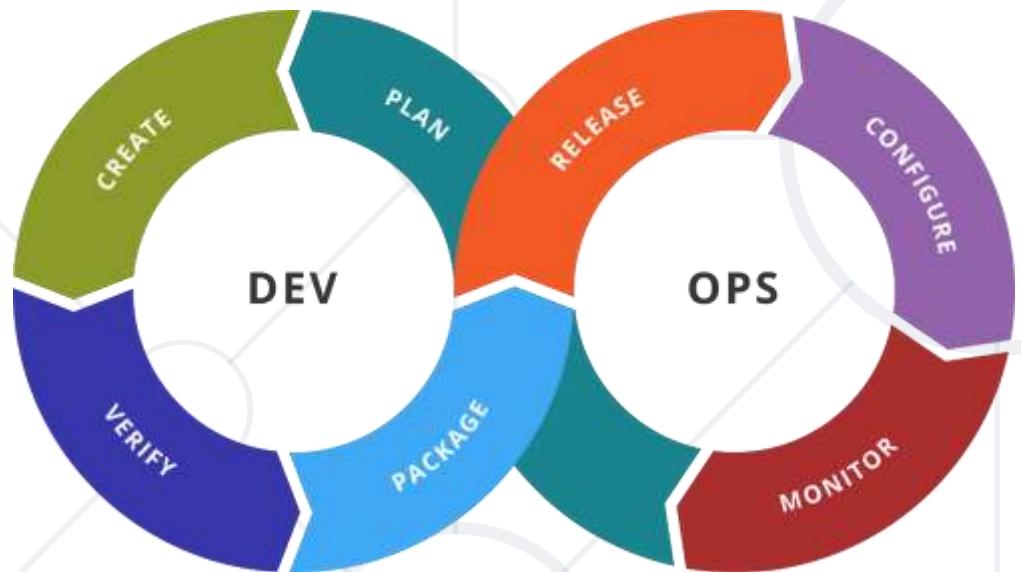
- Software University – High-Quality Education, Profession and Job for Software Developers
 - softuni.bg, softuni.org
- Software University Foundation
 - softuni.foundation
- Software University @ Facebook
 - facebook.com/SoftwareUniversity
- Software University Forums
 - forum.softuni.bg



Software
University



Prometheus and Grafana



SoftUni Team

Technical Trainers



SoftUni



Software University
<https://softuni.org>

You Have Questions?



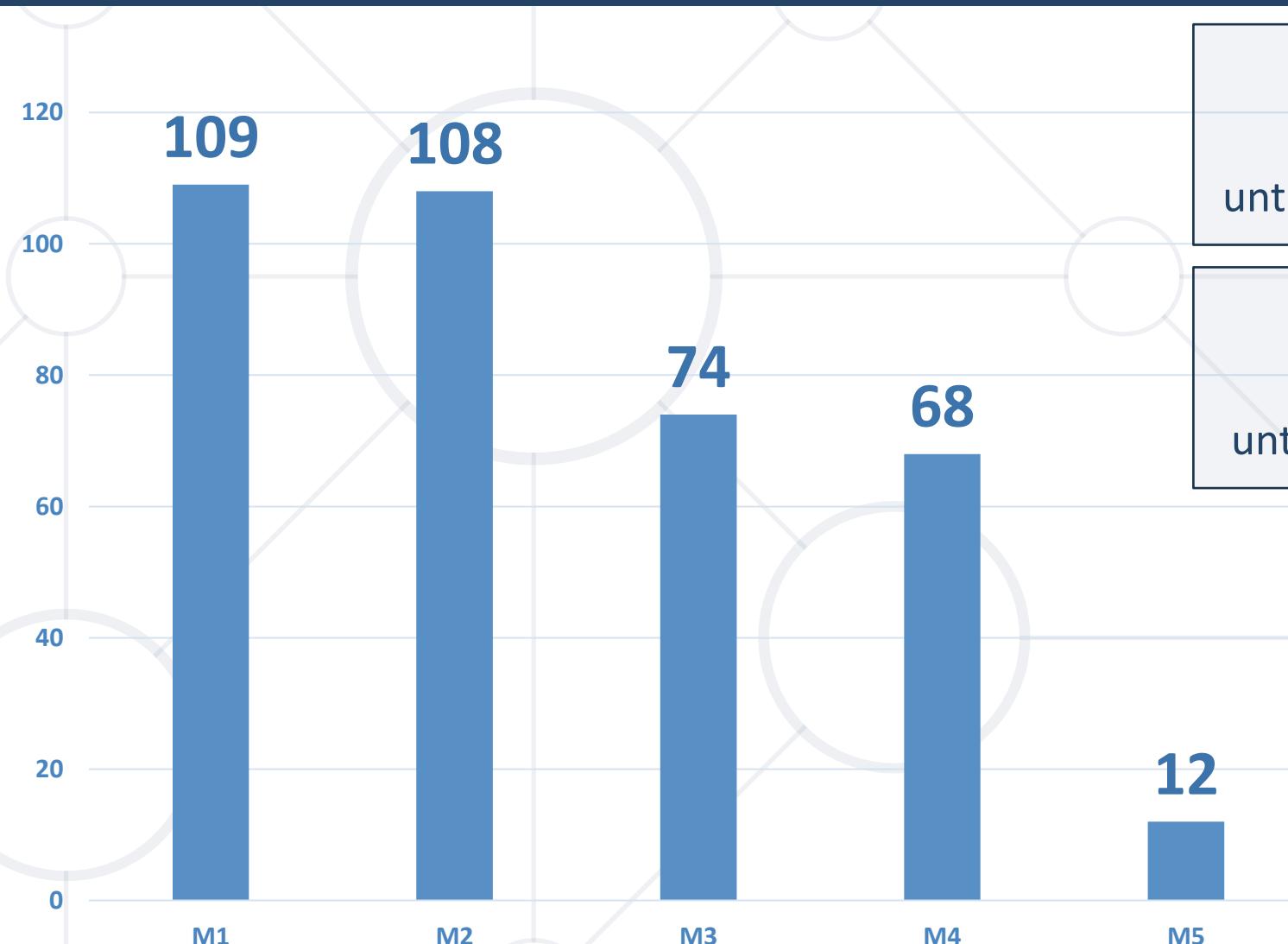
sli.do

#DevOps-23

facebook.com

/groups/DevOpsContainerizationCICDMonitoringJanuary2023

Homework Progress



Solutions for M5
can be submitted
until 23:59 on 08.03.2023

Solutions for M6
can be submitted
until 23:59 on 15.03.2023



Previous Module (M5)

Quick Overview

What We Covered

- Remote (slave) hosts
- Working with Jenkins using CLI
- Export and Import jobs
- Blue Ocean
- SCM integration



This Module (M6)
Topics and Lab Infrastructure

Table of Contents

1. Prometheus

- Architecture
- Installation and configuration
- Information processing and PromQL

2. Grafana

- Installation and configuration
- Dashboarding



Lab Infrastructure

NETWORK: 192.168.99.0/24

.101

.102

.103

VM1

VM2

VM3

VirtualBox

Vagrant

Windows / Linux / macOS



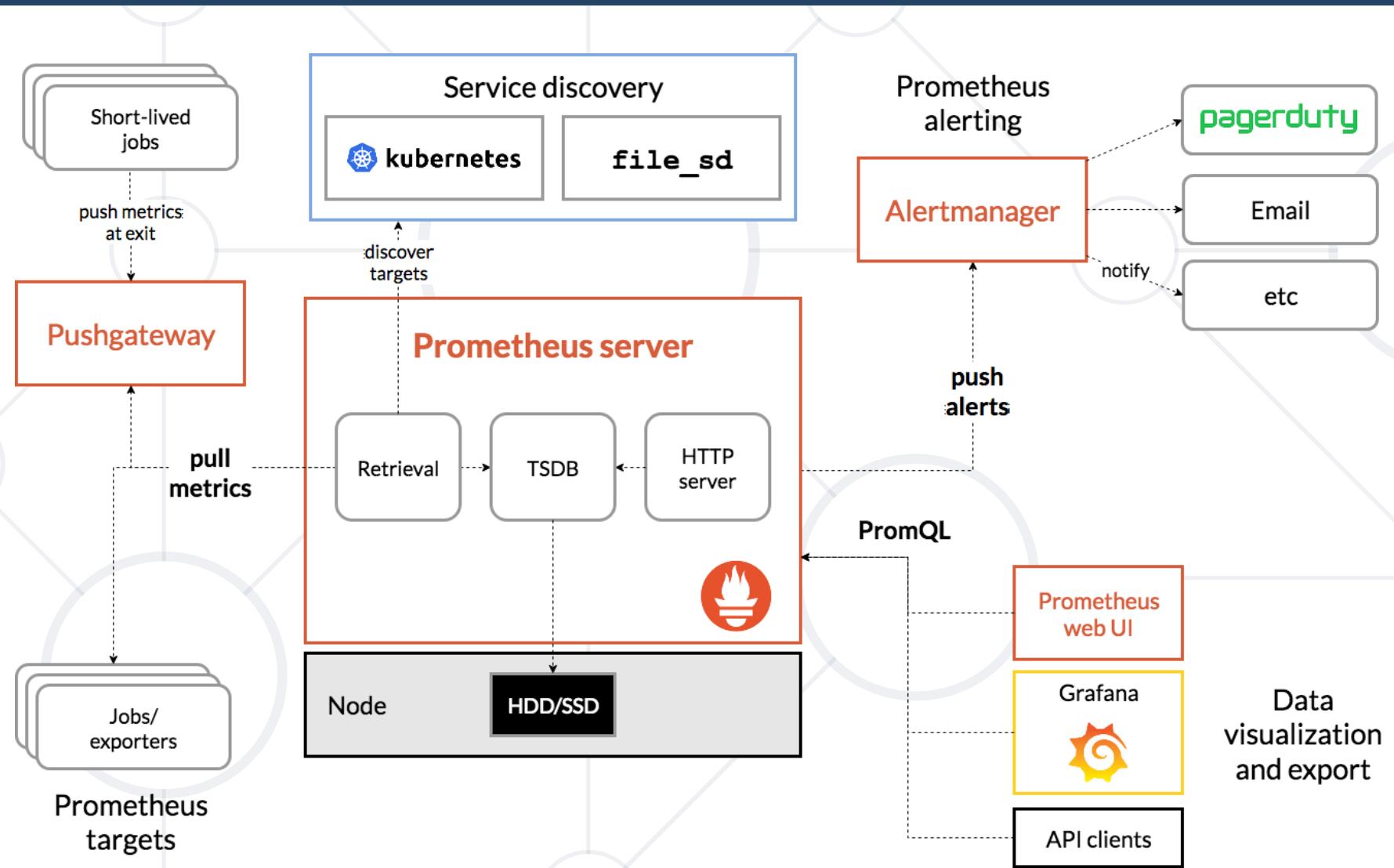
Prometheus 101

Getting to Know Prometheus

What is Prometheus?

- An open-source toolkit for **monitoring** and **alerting**
- Collects and stores **metrics** as **time series** data
- Additionally, it can store **optional labels** (key-value pairs)
- Offers flexible query language (**PromQL**)
- Data collection happens via **pull model** over HTTP
- Pushing of data is also supported via an **intermediary gateway**
- Suitable for recording pure numeric time series
- Not suitable when we need 100% accuracy (for example, for billing)

Architecture



Install Options

- Pre-compiled binaries
 - Most popular operating systems are supported
- Build from source
- Using containers
- Using configuration management systems
 - Ansible, Chef, Puppet and Salt
- Different components are available as separate artefacts

Prometheus Configuration

- Configured via **command-line flags** and a **configuration file**
- Command-line arguments are used to configure **immutable system parameters** like storage location, amount, etc.
- Configuration file is used to control **scraping** and **rules**
- The configuration file is in **YAML** format
- It can be reloaded (including any rules) by sending
 - **SIGHUP** to the Prometheus process
 - **HTTP POST** request to **/-/reload** endpoint
(*--web.enable-lifecycle* flag)

- Data is stored as **time series**
- Which are **streams of timestamped values** belonging to the same metric and the same set of labeled dimensions
- Every time series is uniquely identified by its **metric name and optional key-value pairs** called labels
- The metric name specifies the general feature of a system that is measured
- Labels enable Prometheus's dimensional data model
- Any given combination of labels for the same metric name identifies a particular dimensional instantiation of that metric

```
api_http_requests_total{method="POST", handler="/messages"}
```

Metric Types

- **Counter** is a cumulative metric that represents a single monotonically increasing counter whose value can only increase or be reset to zero on restart. For example, number of tasks completed, requests served, etc.
- **Gauge** is a metric that represents a single numerical value that can arbitrarily go up and down. For example, temperatures, number of concurrent requests, etc.
- **Histogram** samples observations (like request durations or response sizes) and counts them in configurable buckets. It also provides a sum of all observed values
- **Summary** samples observations (like request durations and response sizes). While it also provides a total count of observations and a sum of all observed values, it calculates configurable quantiles over a sliding time window

Jobs, Instances, and Samples

- An endpoint that we can scrape is called an **instance**
- Usually, it corresponds to a single process
- A collection of instances is called a **job**
- When target is scraped, two additional labels are added – **job** and **instance**
- Each instance scrape adds sample to a set of system time series
- A **sample** is a single value at a point in time in a time series
- Each sample consists of a float64 value and a millisecond-precision timestamp



Practice: See It in Action
Live Demonstration in Class



Prometheus 102

Advanced Tasks

- We can utilize two techniques to control the information flow
- **Target relabeling** is a powerful tool to dynamically rewrite the label set of a target before it gets scraped
- **Metric relabeling** is applied to samples as the last step before ingestion
- Both share the same configuration format and actions

Recording Rules

- Allow to precompute frequently needed or computationally expensive expressions and save their result as a new set of time series
- Querying the precomputed result will be much faster
- Useful for dashboards, which need to query the same expression repeatedly every time they refresh
- Stored in rule groups
- Rules within a group are run sequentially at a regular interval

Alerting Rules

- Define alert conditions and send notifications about firing alerts to an external service
- No matter if the alert condition will result in one or more vector elements at a given point in time, the alert counts as active
- They are defined the same way as recording rules



Dev
Ops

PromQL
Introduction

- PromQL = Prometheus Query Language
 - It allows to select and aggregate time series data in real time
 - The result of an expression can either be
 - shown as a graph
 - viewed as tabular data
 - consumed by external systems via the HTTP API
- } in Prometheus's expression browser

Expression Data Types

- Expressions and sub-expressions evaluate to one of the following
- **Scalar** - a simple numeric floating-point value
- **String** - a simple string value; currently unused
- **Instant vector** - a set of time series containing a single sample for each time series, all sharing the same timestamp
- **Range vector** - a set of time series containing a range of data points over time for each time series

Vector Selectors

- Instant vector selectors allow the selection of a set of time series and a single sample value for each at a given timestamp (instant)

```
http_requests_total
```

```
# all time series with the specified metric name
```

- Range vector selectors work the same way but for a range
- A time range is appended at the end of the selector
- It specifies how far back in time values should be fetched

```
http_requests_total[5m]
```

```
# values for the 5 minutes
```

Selectors and Label Matchers

- We can select just some of the time series for a metric with **exact match**

```
http_requests_total{job="web",os="win"}
```

- Other operators include **not equal** (!=)

```
http_requests_total{job="web",method!="POST"}
```

- **regex-match** (=~)

```
http_requests_total{job="web",environment=~"stage|dev|test"}
```

- and **not regex-match** (!~)

```
http_requests_total{job="web",host!~".*dev.*"}
```

Time Durations and Modifiers

- Time durations (**ms**, **s**, **m**, **h**, **d**, **w**, **y**) are specified like **[5m]** and can be concatenated

```
http_requests_total{job="prometheus"}[1h5m]    # values for the last hour and 5 minutes
```

- Offset modifier can be used to change the **start point in time**

```
http_requests_total offset 5m                  # value as of 5 minutes ago
```

- @ modifier is used to change the evaluation time for individual instant and range vectors
- Offset and @ can be combined. In addition, we can use the **start()** and **end()** functions

```
http_requests_total offset 5m @ 1646937045    # value as of 5 min before 20:30:45  
                                                # on 10.03.2022
```

Operators *

- **Arithmetic** binary operators (**+**, **-**, *****, **/**, **%**, **^**) between scalar/scalar, vector/scalar, and vector/vector value pairs
- **Comparison** binary operators (**==**, **!=**, **>**, **<**, **>=**, **<=**) between scalar/scalar, vector/scalar, and vector/vector value pairs
- **Logical/set** binary operators (**and**, **or**, **unless**) between instant vectors
- **Aggregation** operators (**sum**, **min**, **max**, **avg**, etc.) on a single instant vector

(1) Trigonometric binary operators are omitted from the list above

(2) Operations between vectors attempt to find a matching element in the right-hand side vector for each entry in the left-hand side. Two types of matching behavior - one-to-one and many-to-one/one-to-many.



Alerting Introduction

- Separated into two parts
- Alerting rules in Prometheus servers send alerts to an **Alertmanager**
- Then it manages those alerts including **silencing** (mute), **inhibition** (suppress), **aggregation** (group), and **sending out notifications**
- Notification methods include **email**, **on-call notification systems**, and **chat platforms**

Configuration

- Configured via **command-line flags** and a **configuration file**
- Command-line arguments are used to configure **immutable system parameters**
- Configuration file defines **inhibition rules**, **notification routing** and **notification receivers**
- The **visual editor** can assist in building routing trees
- The configuration file is in **YAML** format
- It can be reloaded (including any rules) by sending
 - **SIGHUP** to the Prometheus process
 - **HTTP POST** request to **/-/reload** endpoint (**--web.enable-lifecycle** flag)



Practice: See It in Action
Live Demonstration in Class



Grafana

Getting to Know Grafana

Introduction to Grafana

- Grafana is a complete observability stack
- Monitor and analyze metrics, logs and traces
- Query, visualize, alert on and understand our data
- Create, explore, and share beautiful dashboards
- Offered in three variants – **open source**, **cloud**, and **enterprise**
- Includes additional components like **Loki** and **Tempo**

Installation (on-prem)

- Native packages (for various Linux distributions)
- Windows installation either with **installer** or **binary**
- macOS via **Homebrew** or **standalone binaries**
- Docker image – **Alpine** or **Ubuntu** based
- Supported databases are **SQLite**, **MySQL**, and **PostgreSQL**
- Supported browsers are **Chrome**, **Firefox**, **Safari**, and **Edge**

- Server level settings are spread across three files **defaults.ini**, **grafana.ini**, and **custom.ini**
- We can control the instance and the server
- There is **Grafana CLI** which can be used to automate stuff
- It can be set up for high availability

Data Sources

- Most popular are supported out of the box
- Prometheus, Graphite, OpenTSDB, and InfluxDB
- Loki and Elasticsearch
- MySQL, PostgreSQL, MicrosoftSQL
- Cloud services
- Many enterprise and third-party plugin

Panels. Visualizations. Dashboards

- Panels are the building blocks of dashboards
- They connect via query to a data source
- On them we use different visualizations
- Those can be time series, charts, gauges, tables, etc.
- We then assemble panels as a dashboard



Practice: See It in Action
Live Demonstration in Class

Resources

Prometheus

<https://prometheus.io/>

Prometheus documentation

<https://prometheus.io/docs/introduction/overview/>

Grafana

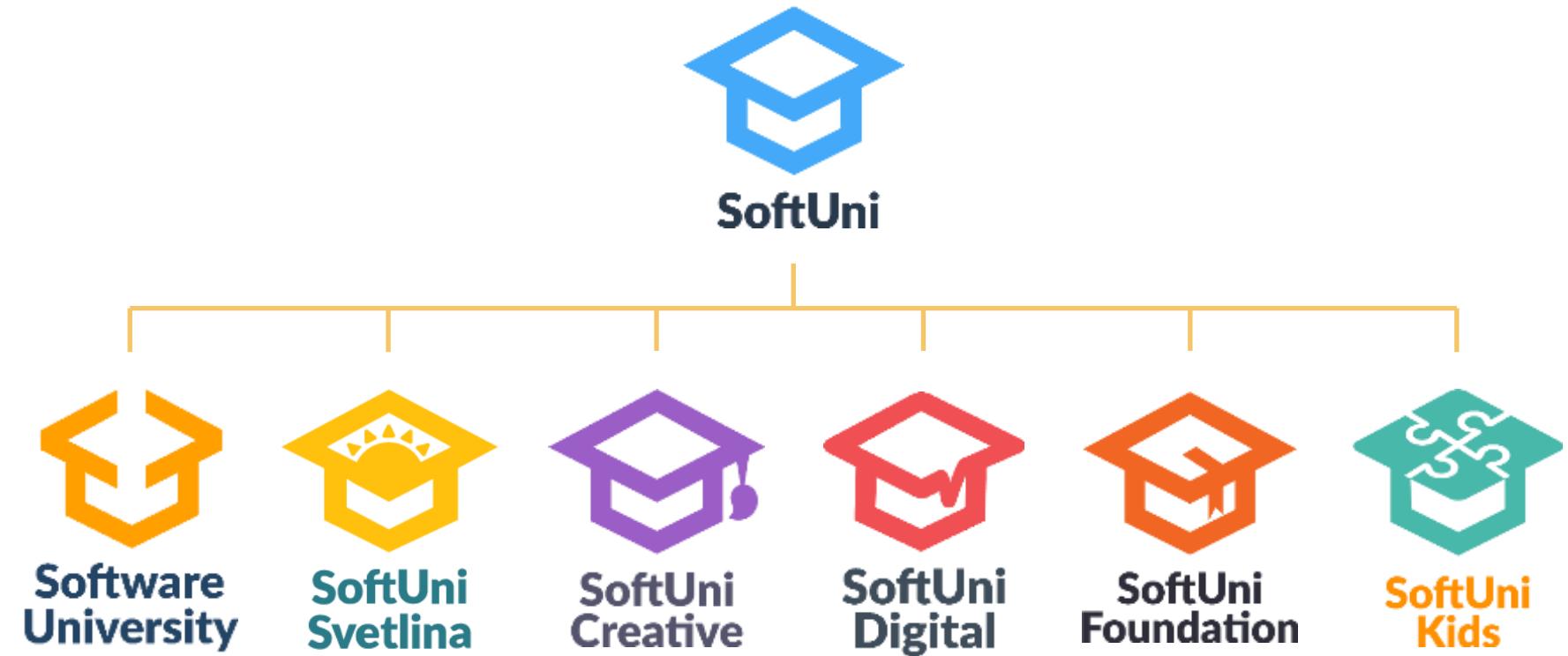
<https://grafana.com/>

Grafana documentation

<https://grafana.com/docs/grafana/latest/>



Questions?



SoftUni Diamond Partners



SCHWARZ



Coca-Cola HBC
Bulgaria



Postbank

Решения за твоето упре



Bosch..IO****



SmartIT



CAREERS

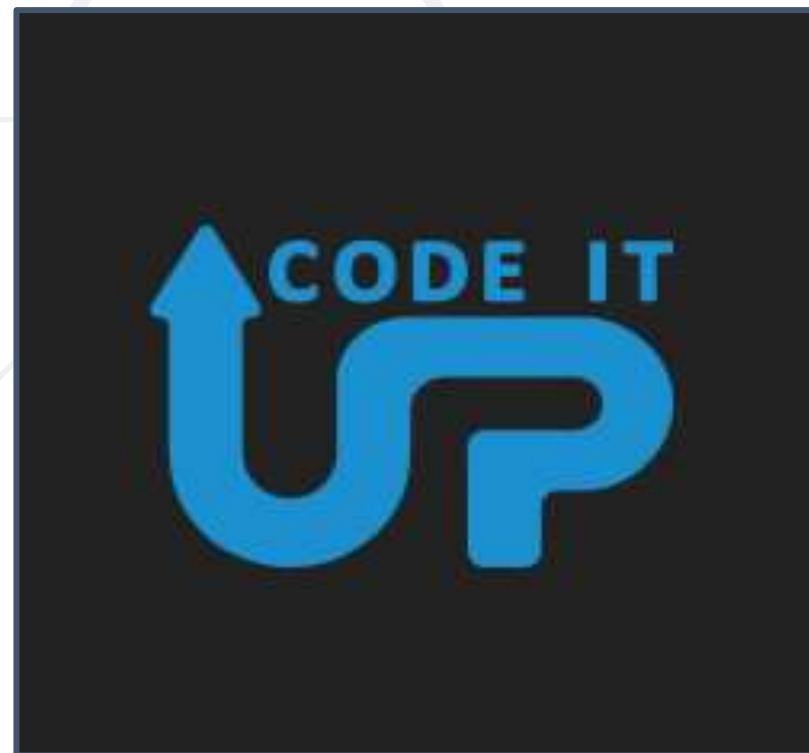


INDEAVR
Serving the high achievers



**SUPER
HOSTING
.BG**

Educational Partners



- 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)



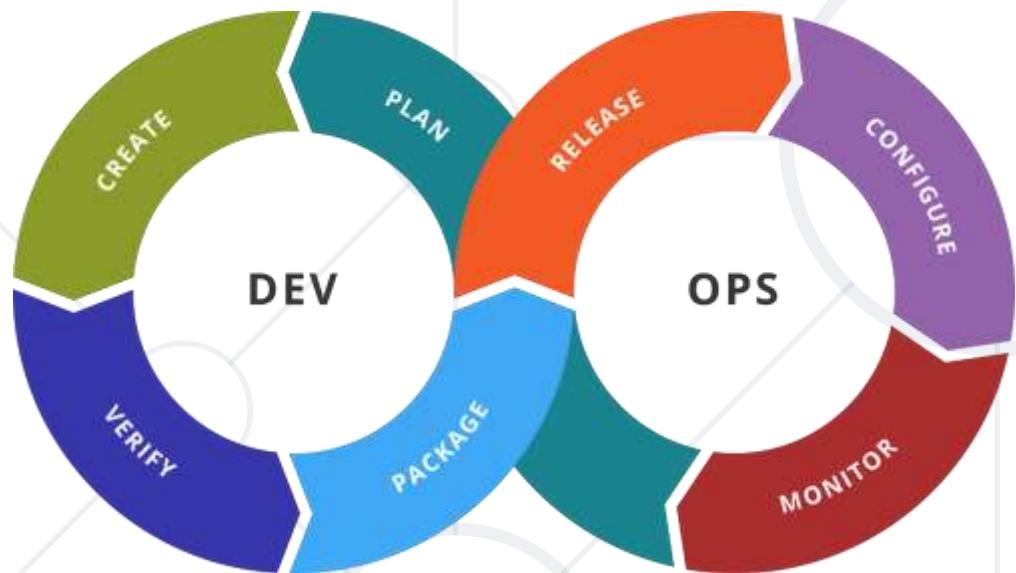
- Software University – High-Quality Education, Profession and Job for Software Developers
 - softuni.bg, softuni.org
- Software University Foundation
 - softuni.foundation
- Software University @ Facebook
 - facebook.com/SoftwareUniversity
- Software University Forums
 - forum.softuni.bg



Software
University



Elastic Stack



SoftUni Team

Technical Trainers

 Software
University



SoftUni



Software University
<https://softuni.org>

You Have Questions?



sli.do

#DevOps-23

facebook.com

/groups/DevOpsContainerizationCICDMonitoringJanuary2023

Homework Progress



Solutions for M6
can be submitted
until 23:59 on 15.03.2023

Solutions for M7
can be submitted
until 23:59 on 22.03.2023

**THIS MODULE
ONE MORE
TO GO**

AND

MORE
TO

Test Your Knowledge

<https://zahariev.pro/q/do1>



Previous Module (M6)

Quick Overview

What We Covered

- Prometheus
 - Architecture
 - Installation and configuration
 - Information processing and PromQL
- Grafana
 - Installation and configuration
 - Dashboarding



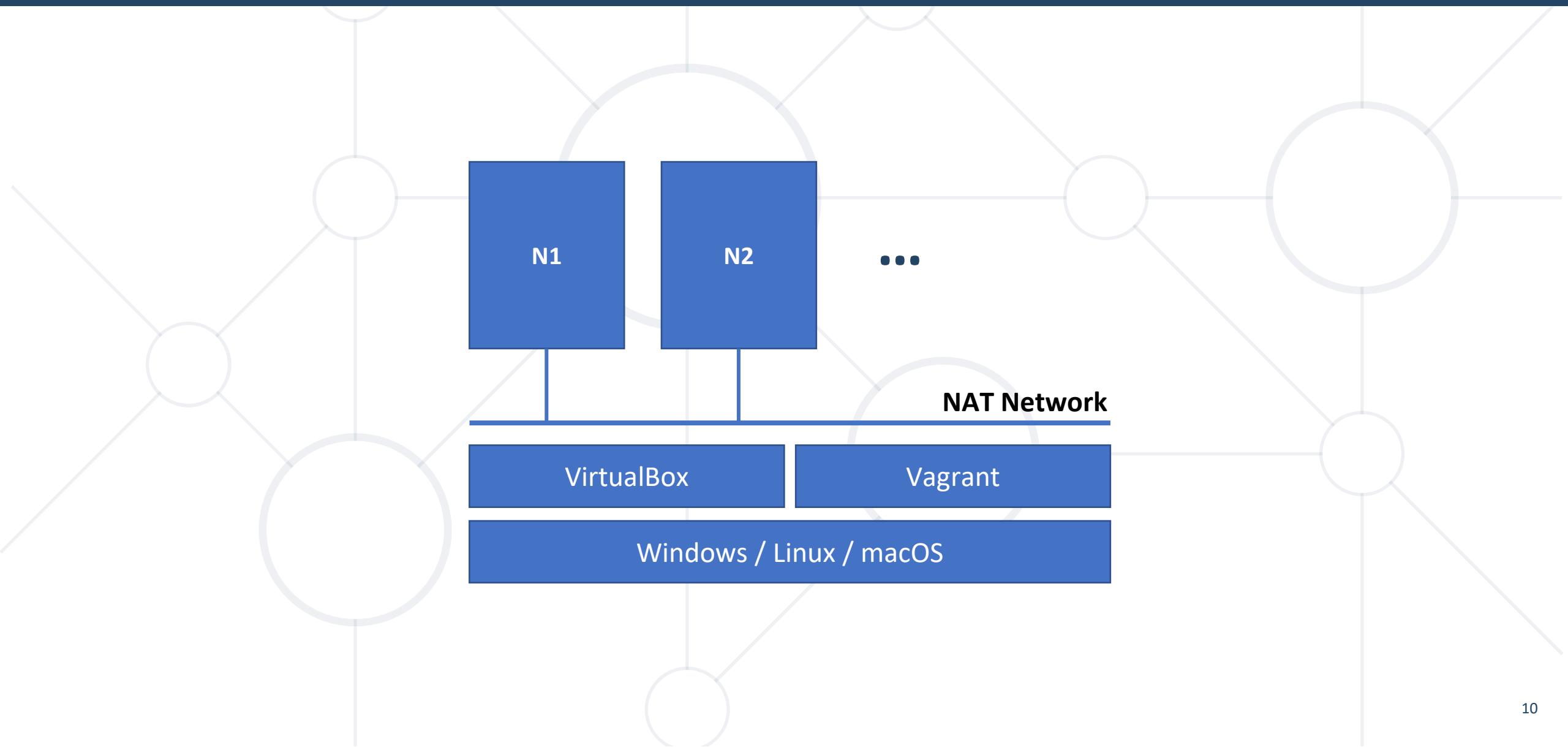
This Module (M7)
Topics and Lab Infrastructure

Table of Contents

- Elastic Stack
 - Elasticsearch
 - Logstash
 - Kibana
- Beats
- Elastic Stack and Docker



Lab Infrastructure





Elastic Stack
ex-ELK Stack

ELK (Elastic) Stack

- Started as **ELK Stack**
 - Elasticsearch
 - Logstash
 - Kibana
- Then became **Elastic Stack**
 - Beats were added
 - Then other components
 - ELK became obsolete as name

- Used for searching, analyzing, and storing of data
- The heart of the Elastic Stack
- Distributed, RESTful search and analytics engine
- Designed for horizontal scalability, maximum reliability, and easy management
- Supported on multiple platforms
- Offered through packaging systems and as standalone package

- Open source, server-side data collection and processing pipeline
- Ingests data from a multitude of sources simultaneously, transforms it, and then sends it to your favorite “stash”
- Complemented by extensible plugin ecosystem
- Strong Elasticsearch synergy
- Processing in three stages – **input**, **filter**, and **output**

- Used for both data visualization and Elastic Stack navigation
- Supports multiple visualizations
- Visualizations can be used to create Dashboards
- Offers interactive experience



Practice: See It in Action
Live Demonstration in Class



Beats
Feed the data in

- Platform for single-purpose data shippers
- Can send data from hundreds or thousands of machines and systems to **Logstash** or **Elasticsearch**
- Lightweight - they just gather and send data without processing
- Plug & Play - use modules to simplify the process
- Extensible - we can build our own

A Few Beats *

- **Heartbeat**
 - Monitor services for their availability with active probing
- **Metricbeat**
 - Collect metrics from your systems and services
- **Filebeat**
 - Offers lightweight way to forward and centralize logs and files
- **Auditbeat**
 - Audits the activities of users and processes on the systems



Practice: See It in Action
Live Demonstration in Class



DevOps

Docker and Elastic Stack

Docker and Elastic Stack

- Docker can be monitored as well as its containers
- We will explore both scenarios
- Furthermore, we will run the whole stack on Docker

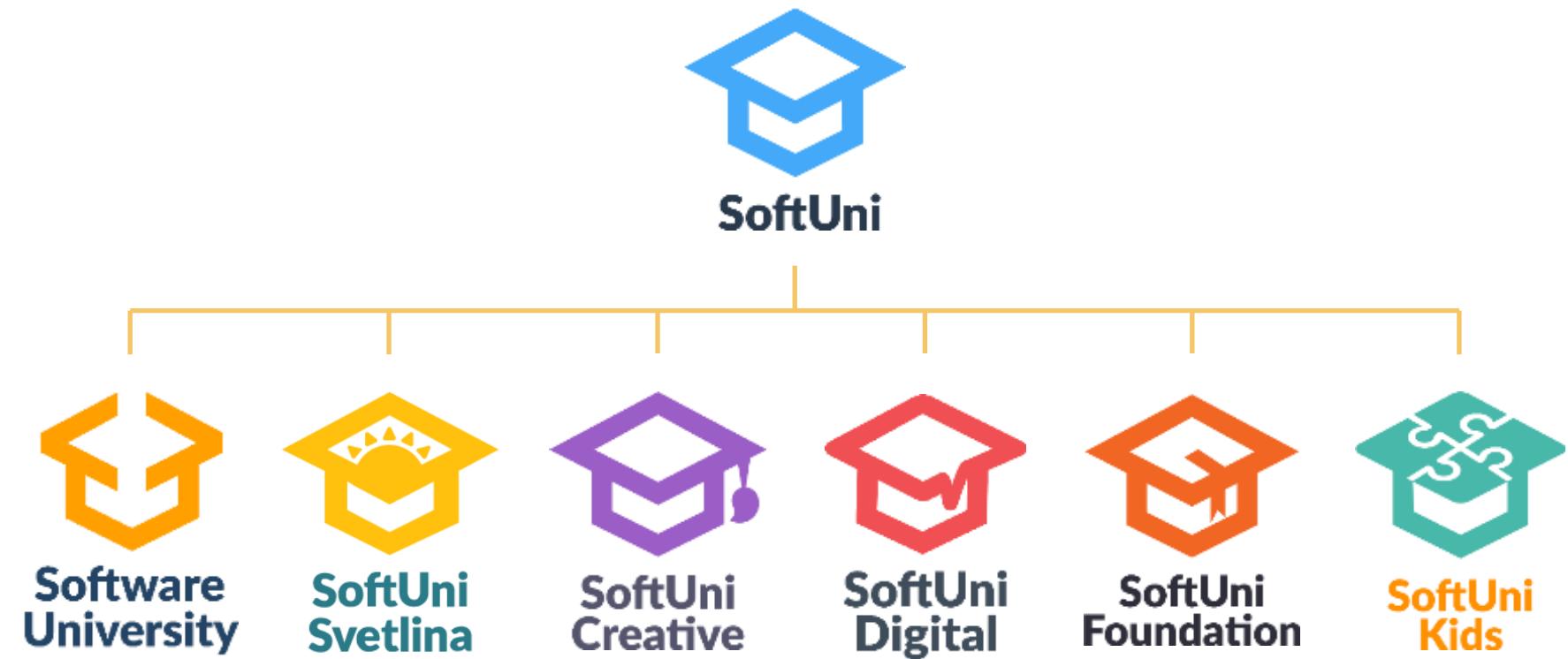


Practice: See It in Action
Live Demonstration in Class

- Get Started with Elastic Stack
 - <https://www.elastic.co/start>
- Get Started with Beats
 - <https://www.elastic.co/guide/en/beats/heartbeat/current/heartbeat-getting-started.html>
 - <https://www.elastic.co/guide/en/beats/metricbeat/current/metricbeat-getting-started.html>
 - <https://www.elastic.co/guide/en/beats/auditbeat/current/auditbeat-installation-configuration.html>
 - <https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-getting-started.html>



Questions?



SoftUni Diamond Partners



SCHWARZ



Coca-Cola HBC
Bulgaria



Postbank
Решения за твоето упре



Bosch..IO****



SmartIT



CAREERS

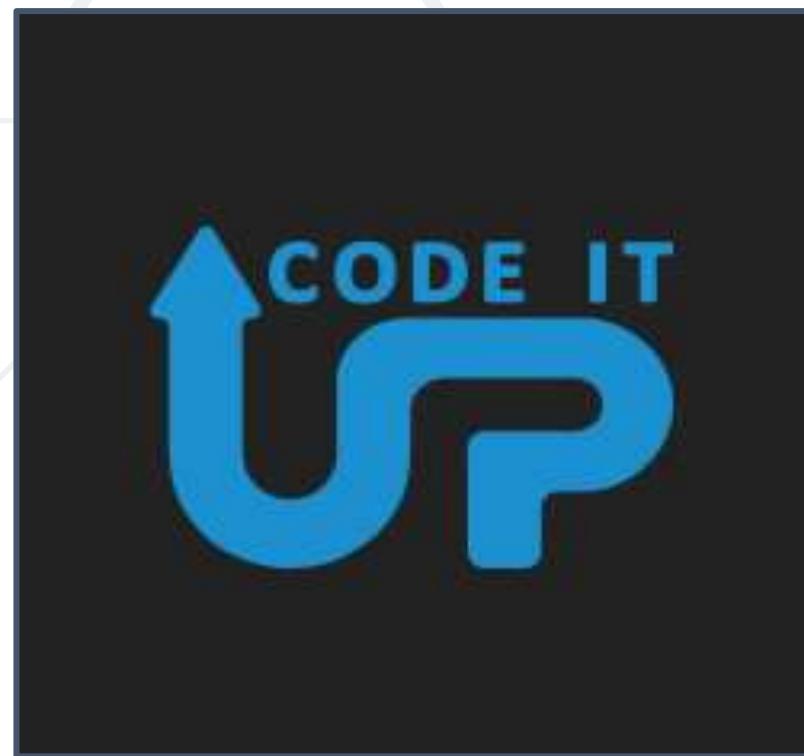


INDEAVR
Serving the high achievers



**SUPER
HOSTING
.BG**

Educational Partners



- 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, Profession 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



Software
University

