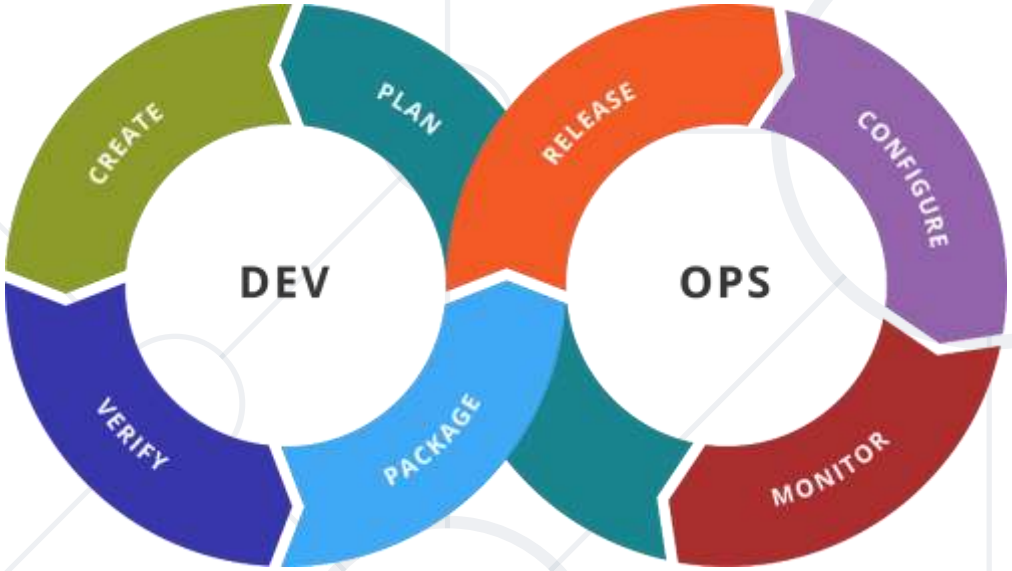
Introduction to DevOps

The Big Picture. Basic Toolkit. Basic Automation



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

You Have Questions?

Software University

sli.do

#DevOps-23

facebook.com

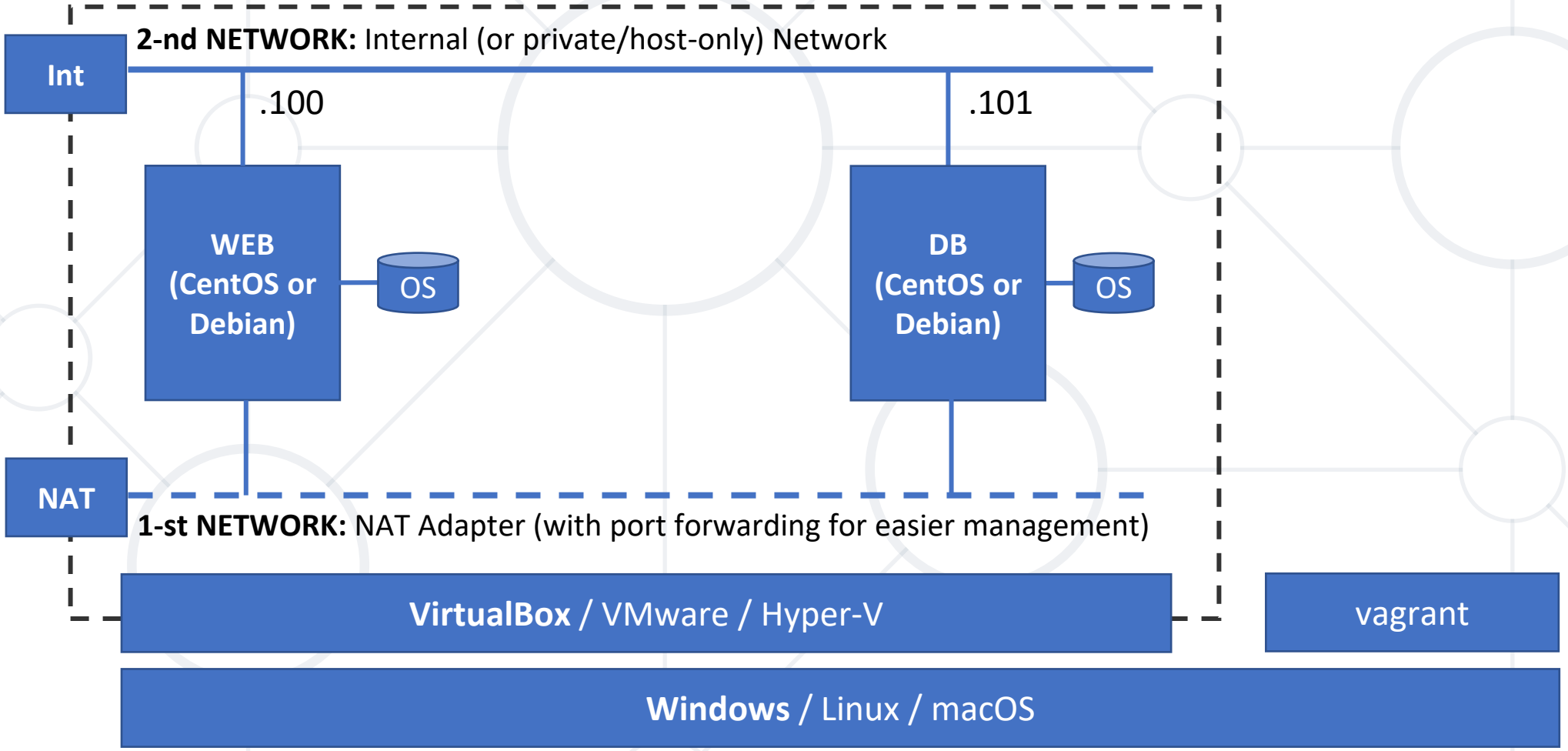
/groups/DevOpsContainerizationCICDMonitoringJanuary2023

This Module (M1)

Topics and Infrastructure

Table of Contents

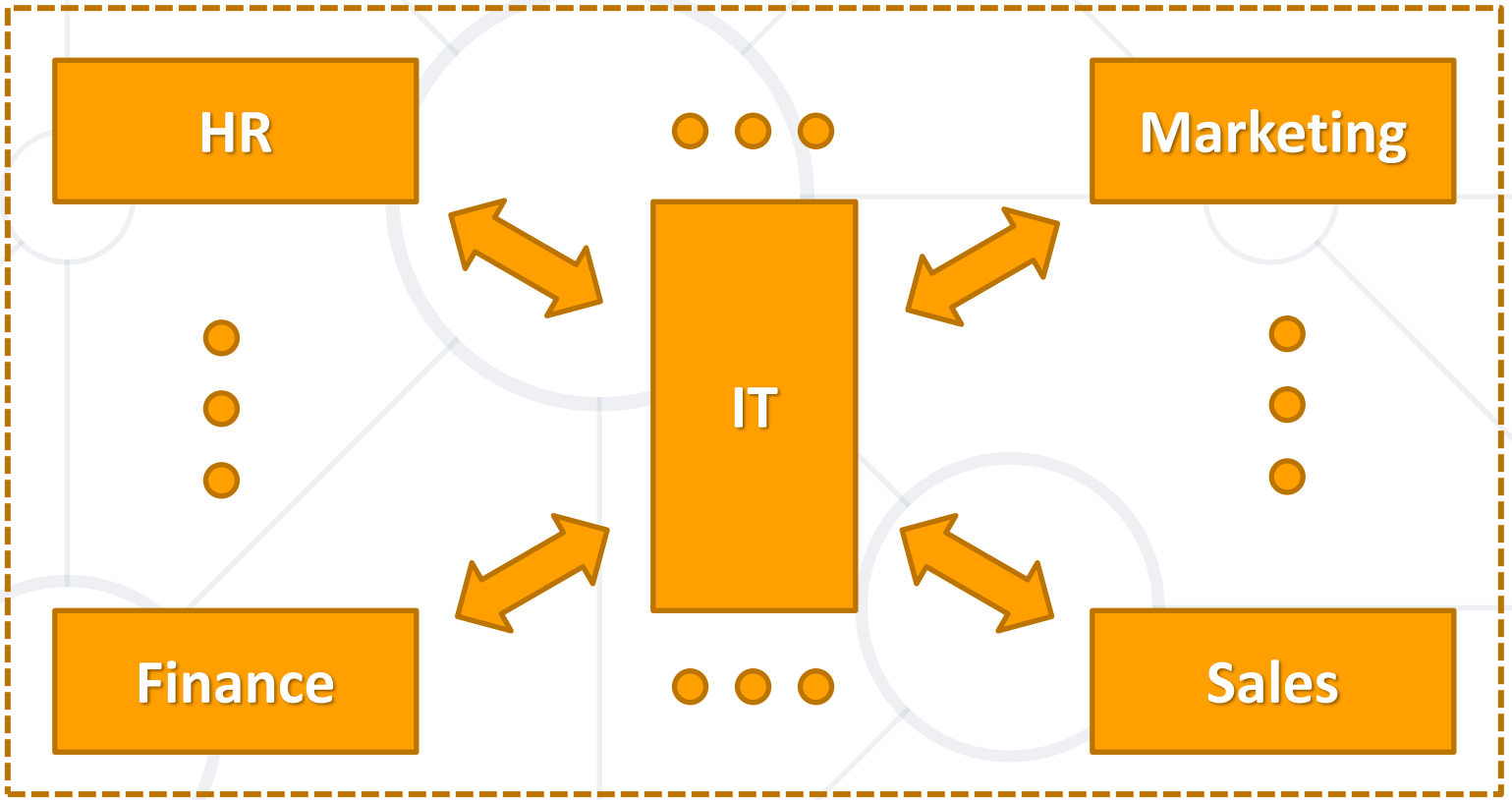
Lab Infrastructure



The Big Picture

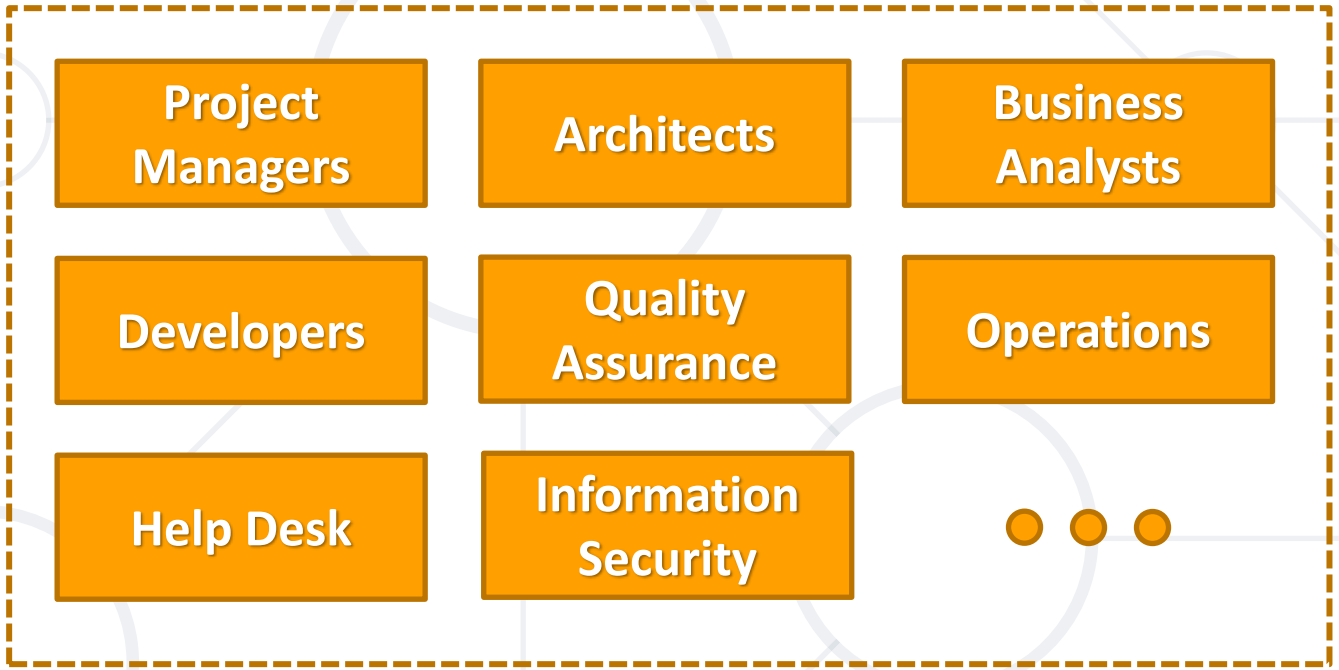
(Why) Do we need a change?

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  - Absent know-how, outdated or missing documentation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 leas 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

Slow IT

- 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

Logistics

- **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

Goals and Benefits

Main Goal

Software University

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

* Damon Edwards and John Willis

Adoption

Making the Transition

Three Main Tasks

Software University

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)

Software University

- **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

Mentality

Software University

~~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

It’s Time For a Break

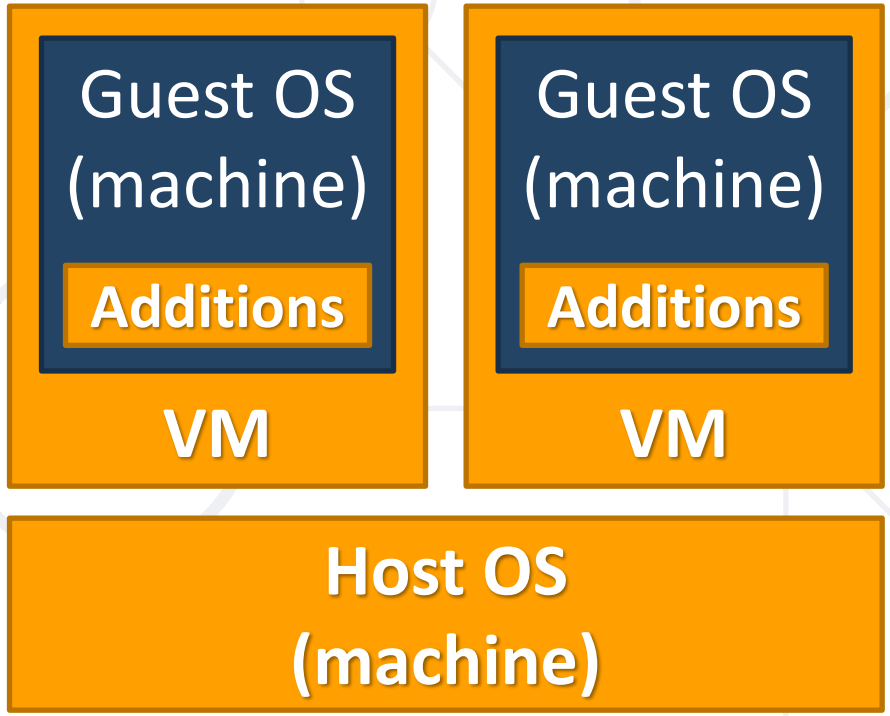
Back in a few minutes ☺

Virtualization

Fundamental Principles and Use Cases

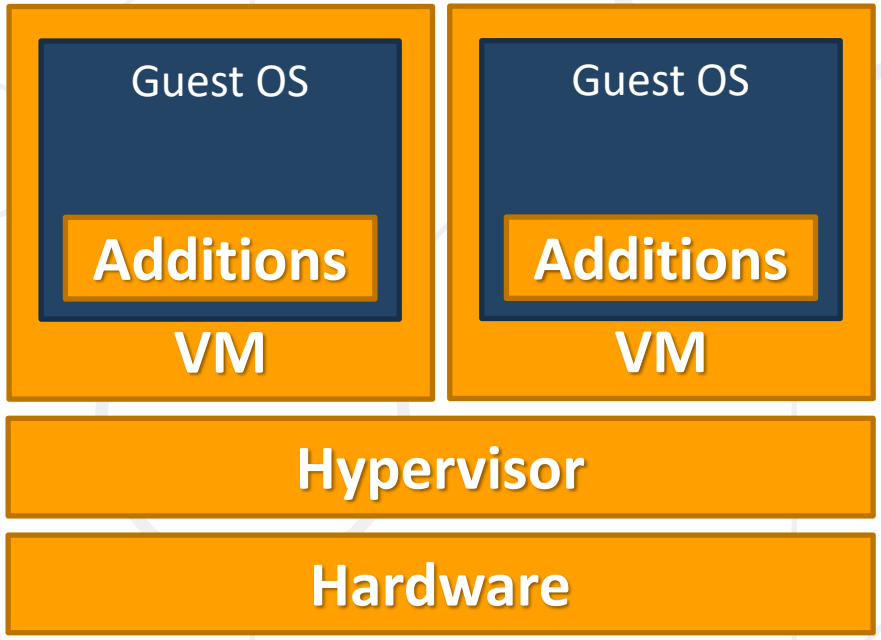
What is Virtualization?

- 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

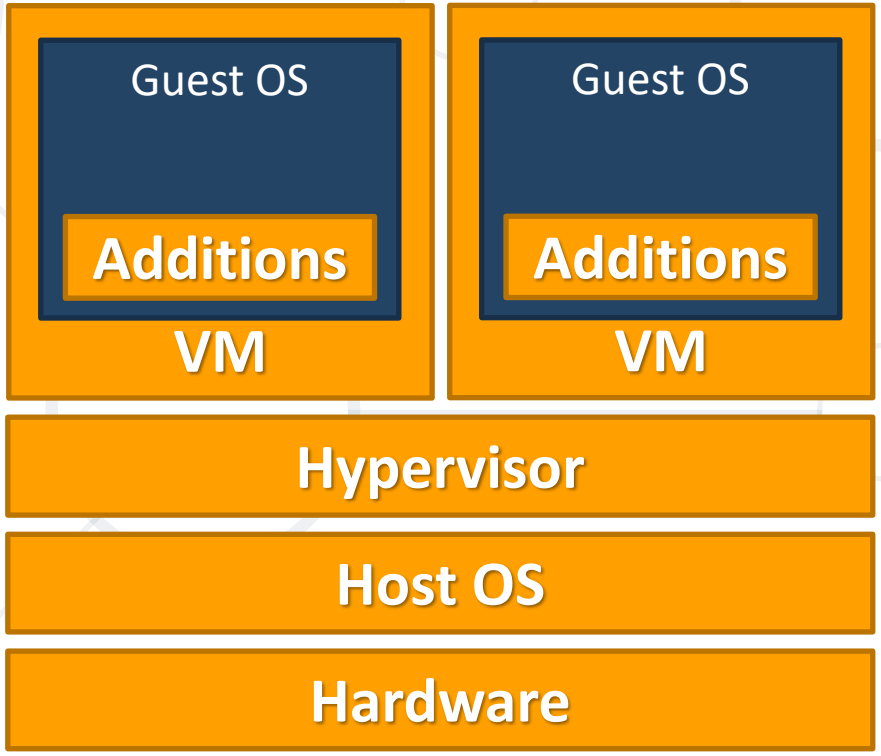


Hypervisors

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



Type 1 (bare metal)



Type 2 (hosted)

Use Cases

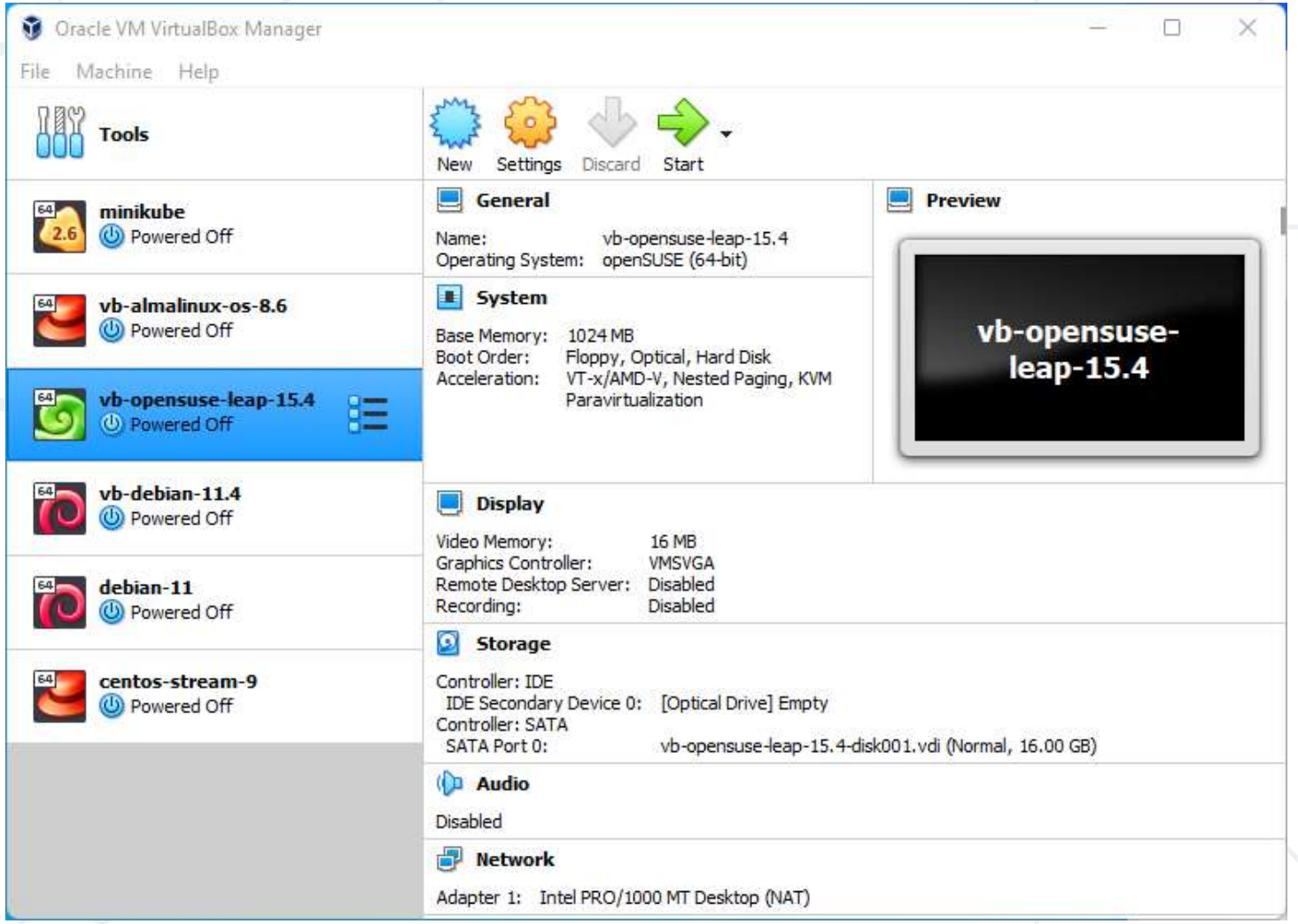
- Infrastructure **consolidation**
  - Better usage and utilization of the available hardware
- Maintain **separate environments**
  - For example – development, test, production
- **Testing** and **evaluation**
  - Test a newer software version or evaluate a product
- **High availability** and **disaster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

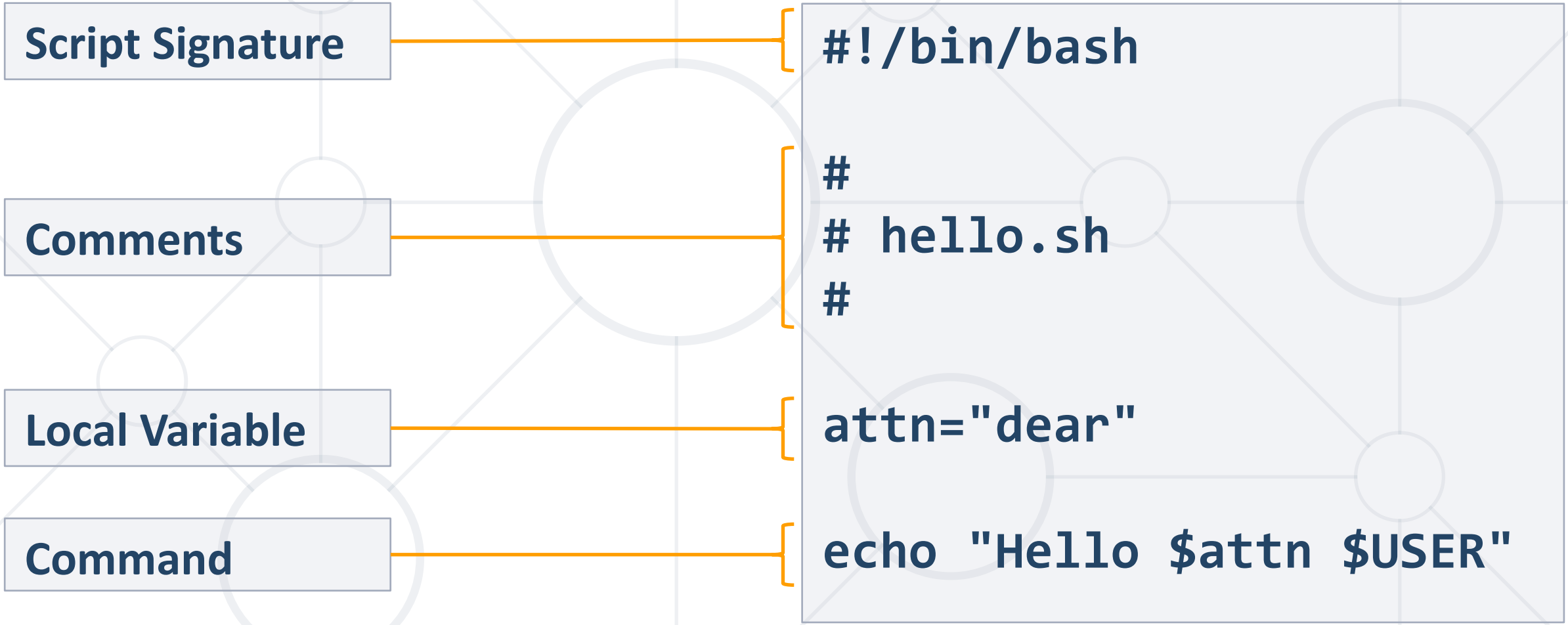
What we need to know?

- 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`**

echo

- Description
  - Display line of text
- Example

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

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

read

- 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
```

if

- 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
```

test

- 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
```

until

- 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`

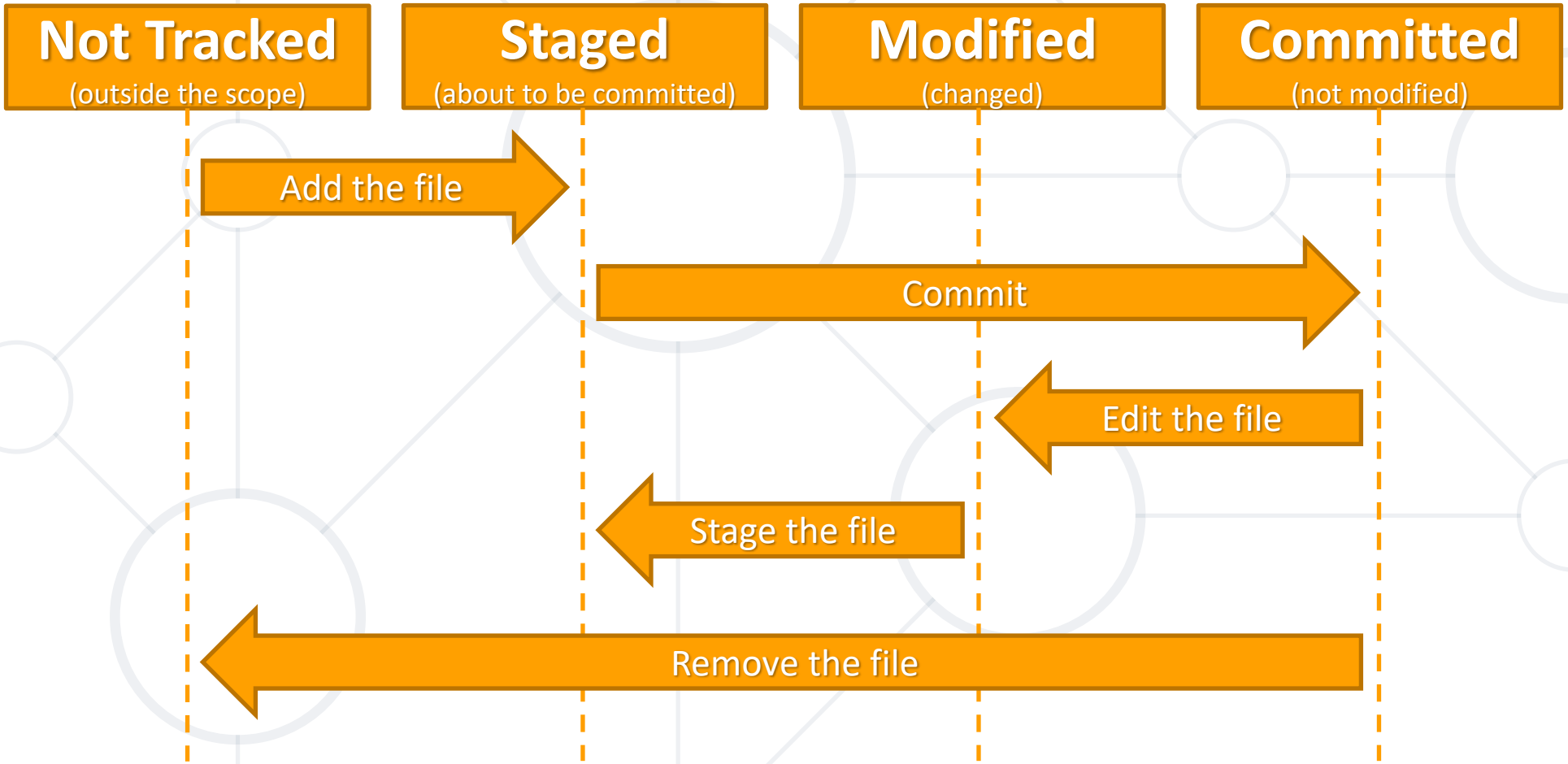
Source Control

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

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

- 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

https://www.vagrantup.com/docs/boxes/base.html

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 grant 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

Software University

- 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

Software University

SCHWARZ

LIDL

Kaufland

SCHWARZ Produktion

prezero

Coca-Cola HBC Bulgaria

POKERSTARS | CAREERS

Postbank

Решения за твоето утре

AMBITIONED

DXC TECHNOLOGY

SOFTWARE GROUP

Bosch.IO

INDEAVR

Serving the high achievers

DRAFT KINGS

PHAR VISION

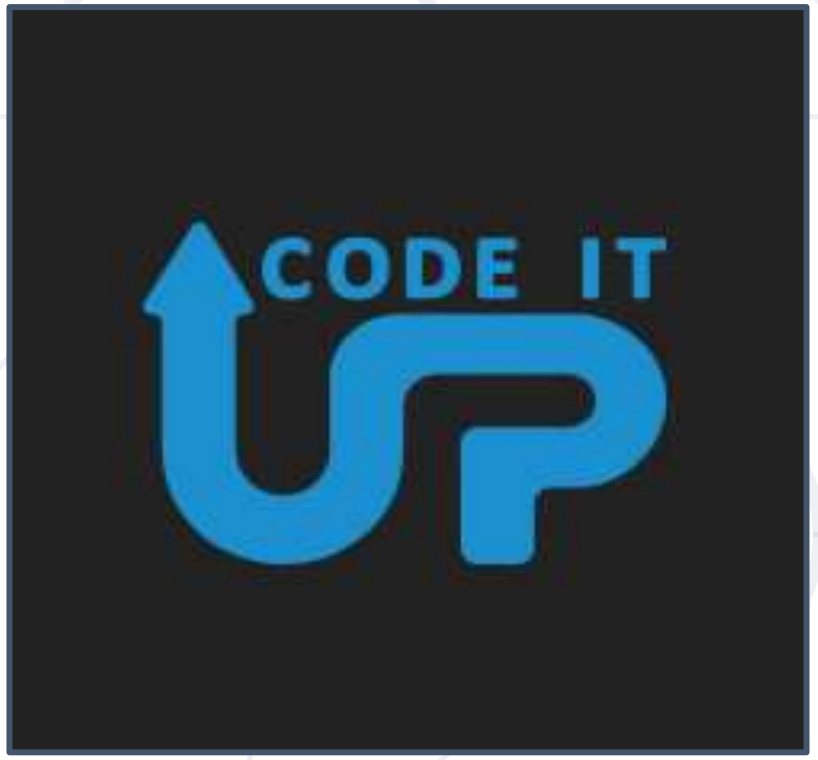
SmartIT

createX

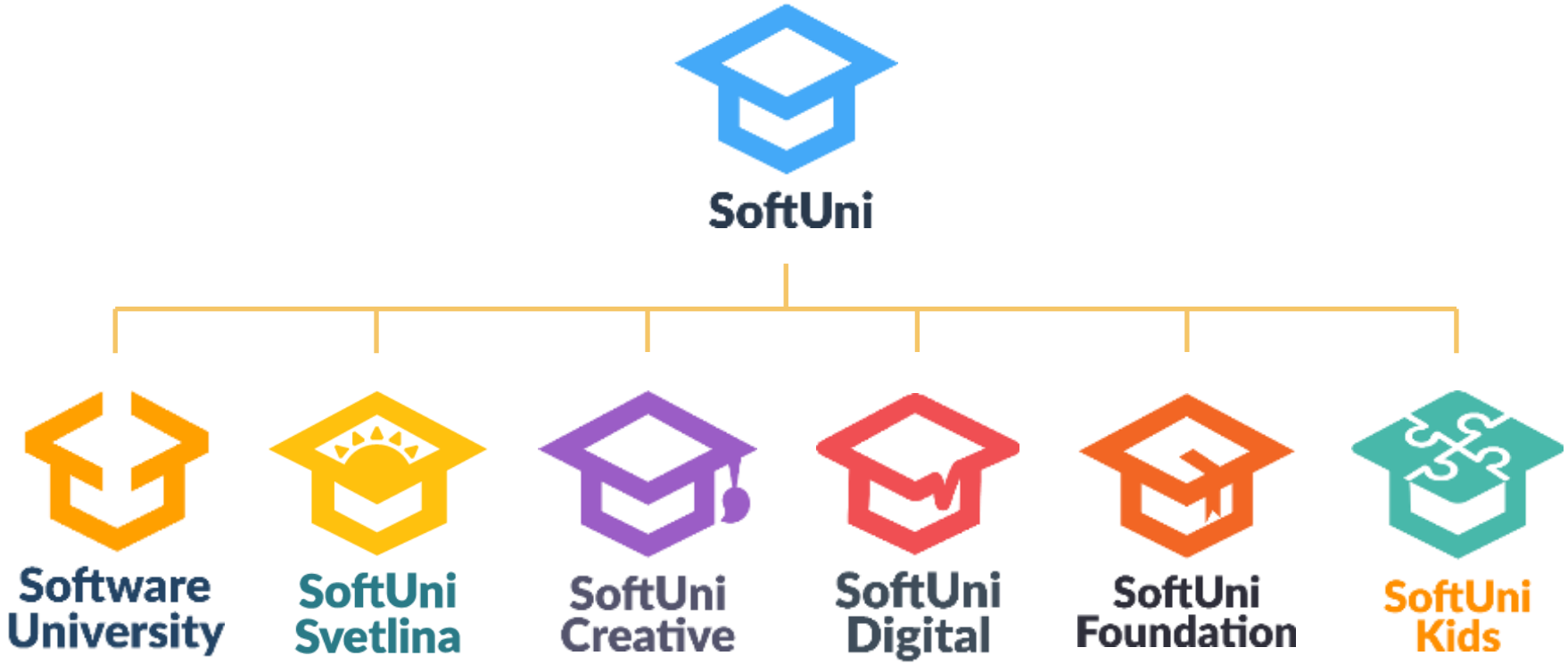
SUPER HOSTING .BG

Educational Partners

Software University



Questions?



License

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

Trainings @ Software University (SoftUni)

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