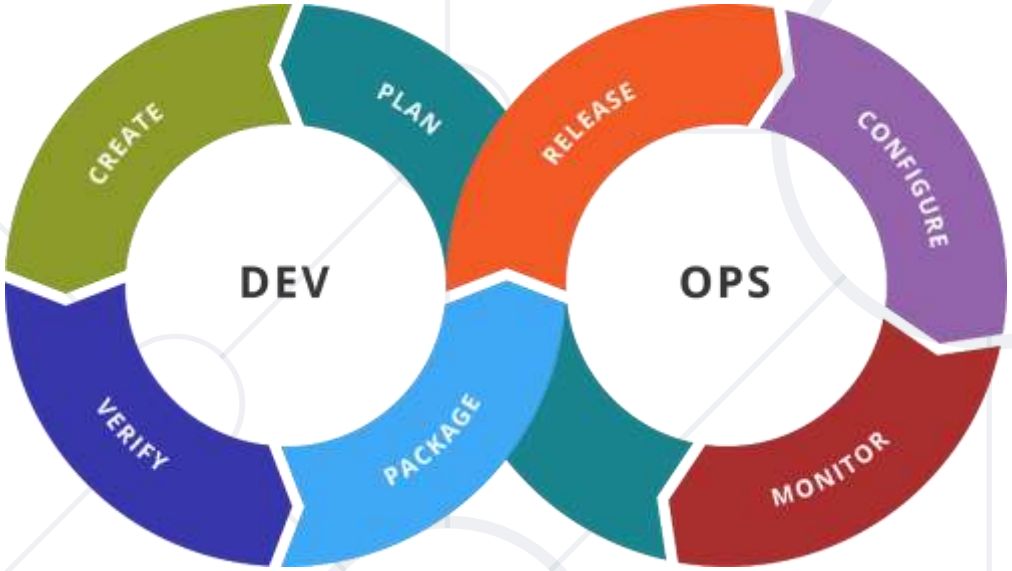
Prometheus and Grafana



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.org

You Have Questions?

Software University

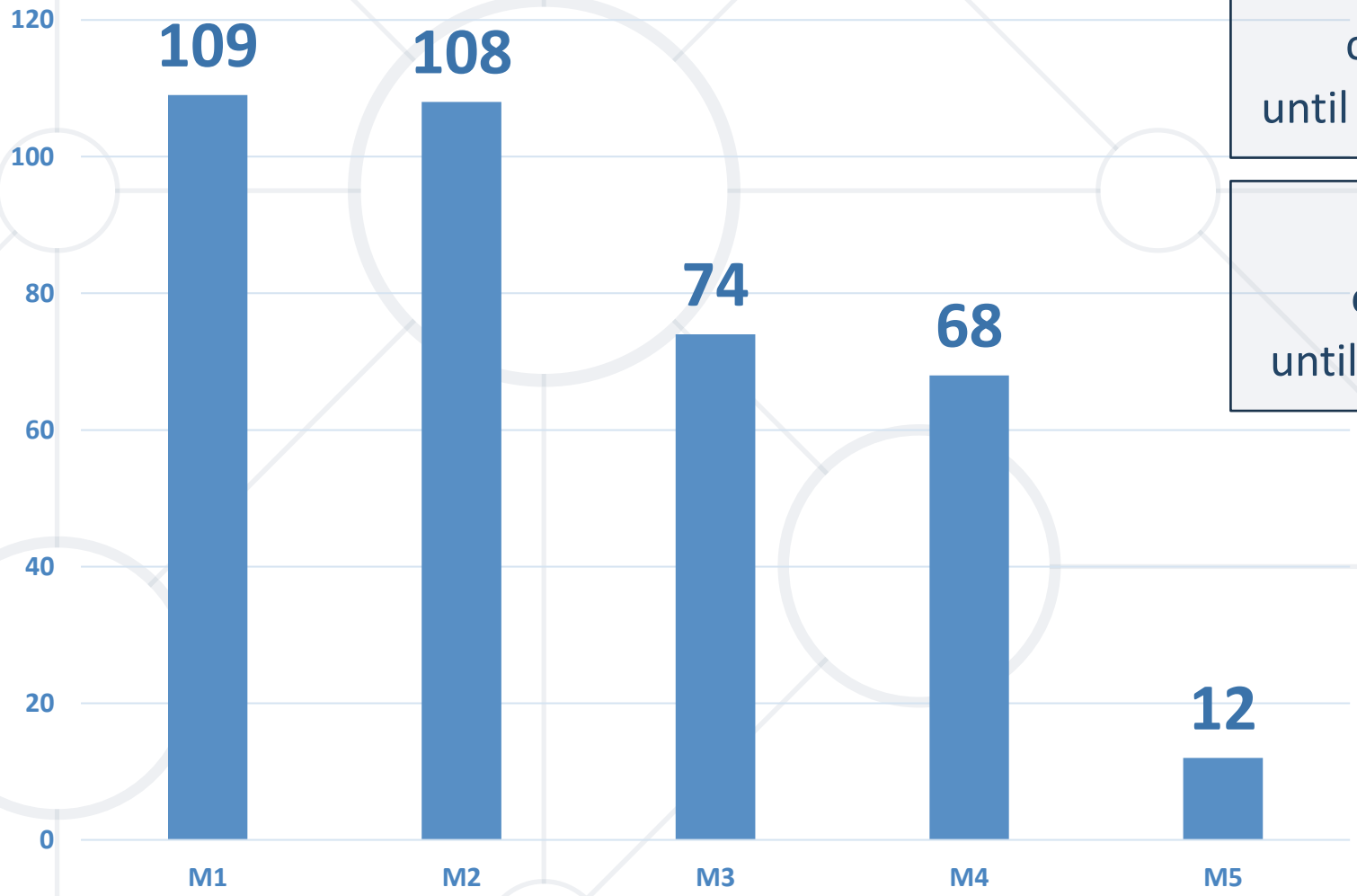
sli.do

#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

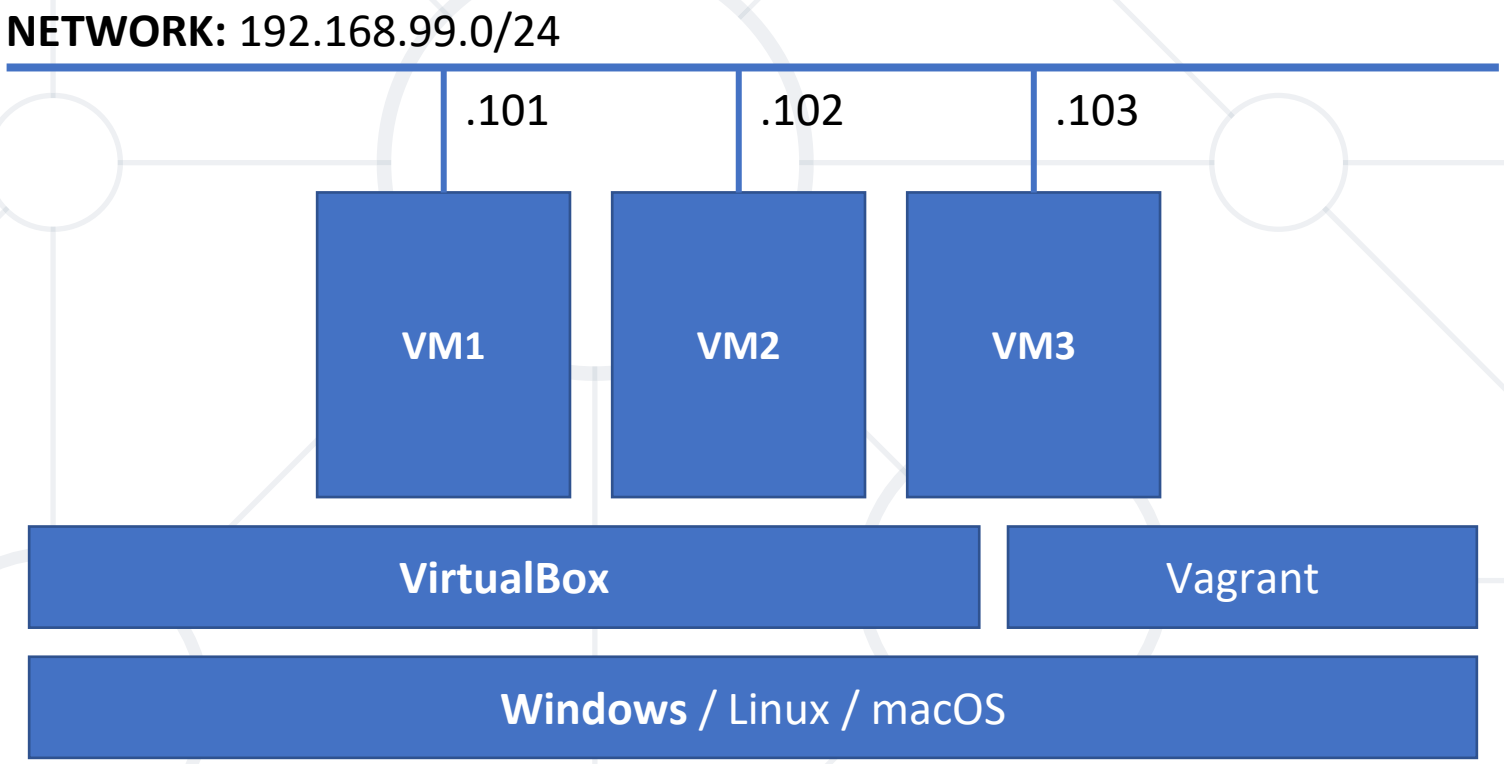
- Remote (slave) hosts
- Working with Jenkins using CLI
- Export and Import jobs
- Blue Ocean
- SCM integration

This Module (M6)

Topics and Lab Infrastructure

Table of Contents

Lab Infrastructure



Prometheus 101

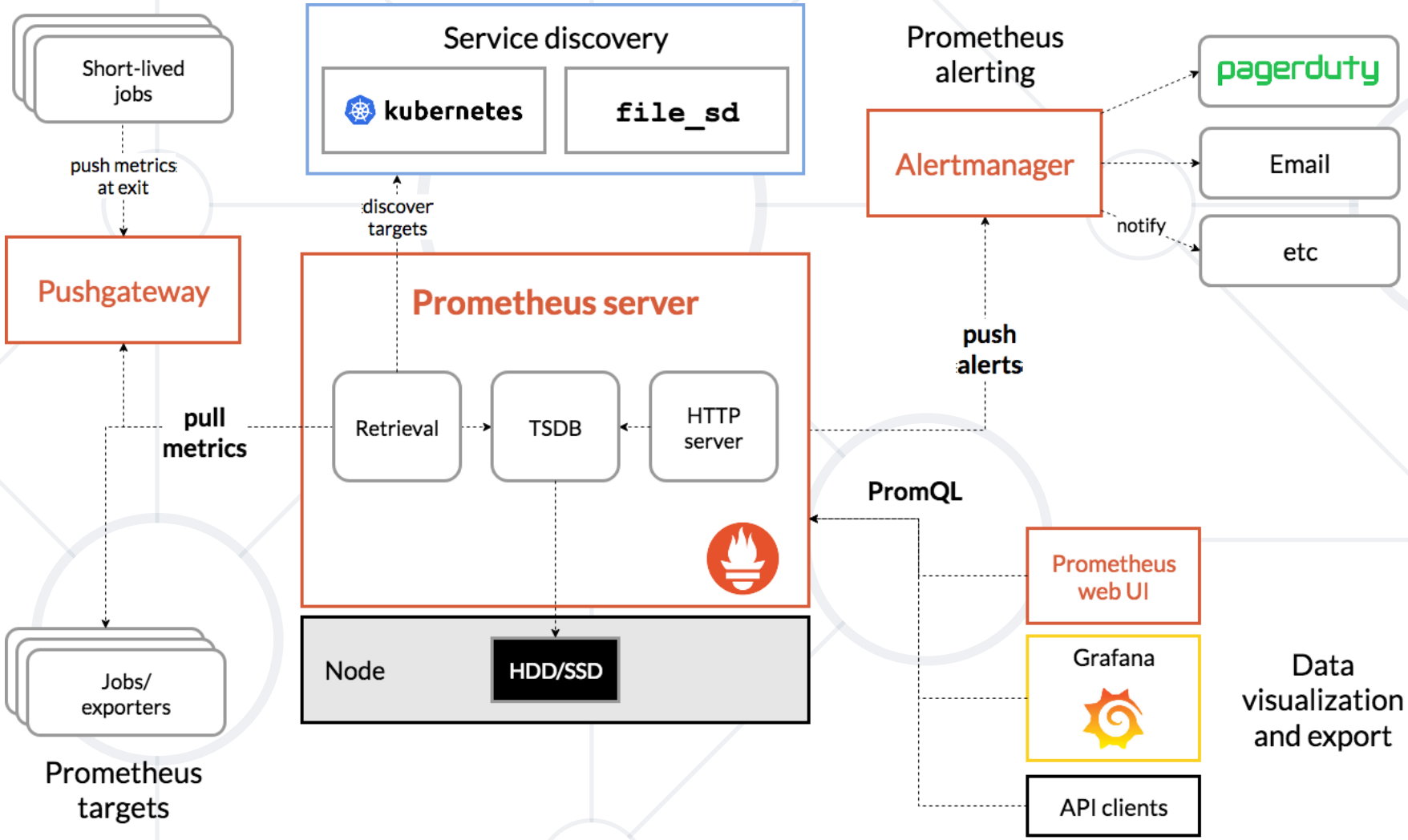
Getting to Know Prometheus

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)

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

Architecture



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

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

https://prometheus.io/docs/prometheus/latest/installation/

Prometheus Configuration

Software University

- 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)

https://prometheus.io/docs/prometheus/latest/configuration/configuration/

Data Model

- Data is sored 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"}
```

https://prometheus.io/docs/concepts/data_model/

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

https://prometheus.io/docs/concepts/metric_types/

Jobs, Instances, and Samples

Software University

- 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

https://prometheus.io/docs/concepts/jobs_instances/

Practice: See It in Action

Live Demonstration in Class

Prometheus 102

Advanced Tasks

Relabeling

- 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

https://prometheus.io/docs/prometheus/latest/configuration/configuration/#relabel_config

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

https://prometheus.io/docs/prometheus/latest/configuration/recording_rules/

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

https://prometheus.io/docs/concepts/jobs_instances/

PromQL

Introduction

PromQL

- 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

  } in Prometheus's expression browser
  - consumed by external systems via the HTTP API

https://prometheus.io/docs/concepts/jobs_instances/

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

https://prometheus.io/docs/concepts/jobs_instances/

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
```

https://prometheus.io/docs/prometheus/latest/querying/basics/#time-series-selectors

Selectors and Label Matchers

Software University

- 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.*"}
```

https://prometheus.io/docs/prometheus/latest/querying/basics/#time-series-selectors

Time Durations and Modifiers

Software University

- **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
```

https://prometheus.io/docs/prometheus/latest/querying/basics/#time-series-selectors

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.

https://prometheus.io/docs/prometheus/latest/querying/operators/

Alerting

Introduction

Alerting

- 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

Software University

- 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)

https://prometheus.io/docs/prometheus/latest/configuration/configuration/

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**

Administration

- 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/

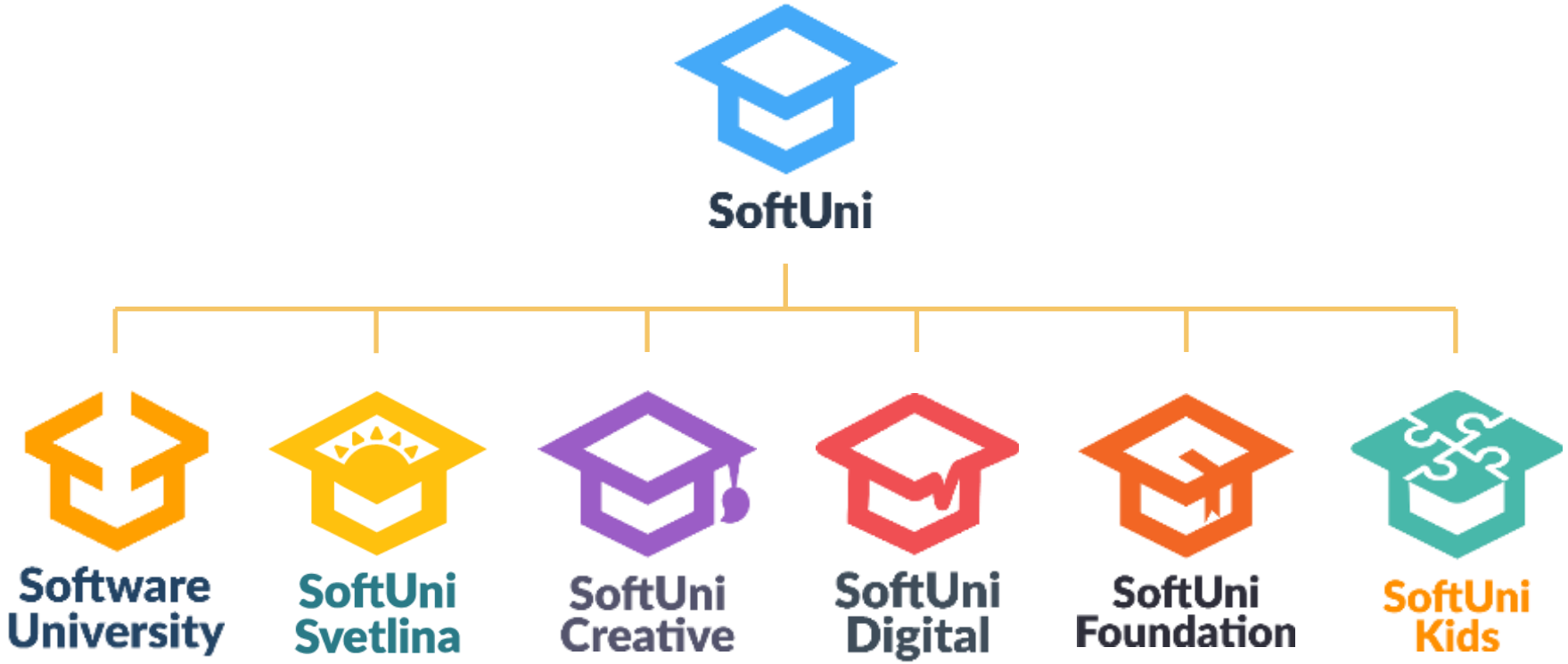
Grafana

https://grafana.com/

Grafana documentation

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

Questions?



SoftUni Diamond Partners

Software University

SCHWARZ

LIDL

Kaufland

SCHWARZ Produktion

prezero

Coca-Cola HBC Bulgaria

POKERSTARS | CAREERS

Postbank

Решения за твоето утре

AMBITIONED

DXC TECHNOLOGY

SOFTWARE GROUP

Bosch.IO

INDEAVR

Serving the high achievers

DRAFT KINGS

PHAR VISION

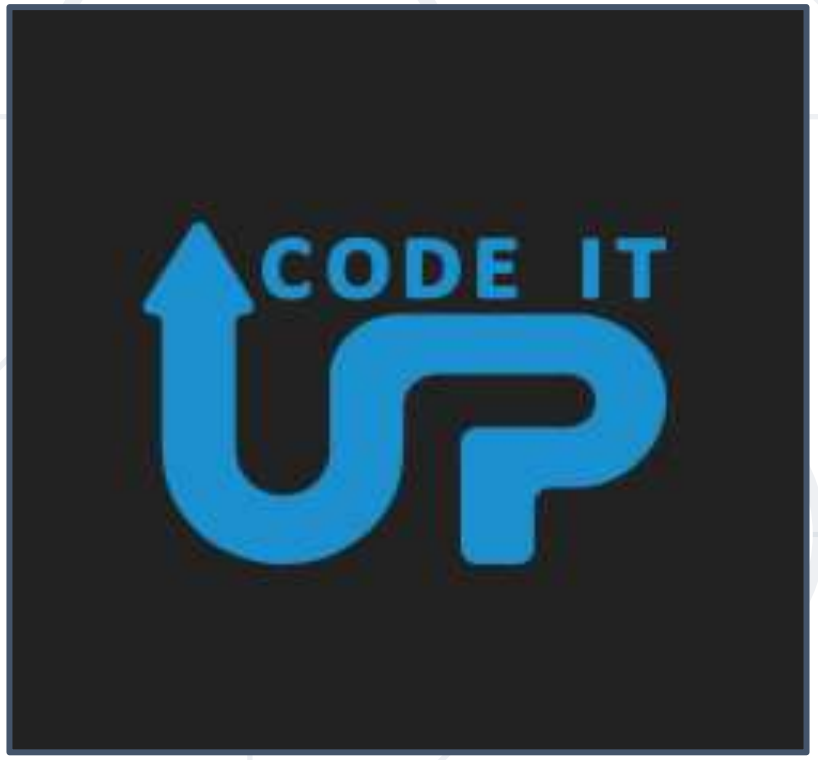
SmartIT

createX

SUPER HOSTING .BG

Educational Partners

Software University



License

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

Trainings @ Software University (SoftUni)

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