App Monitoring

App Monitoring Metrics and Data, Types, Prometheus Tool, Alerting, Alertmanager, Visualization, Grafana

Technical Trainers

SoftUni Team

Software University

SoftUni

https://softuni.bg

Software University

Have a Question?

Software University

sli.do

#Dev-Ops

Table of Content

App Monitoring

What is Application Monitoring?

- **Application monitoring** == the process of **tracking and measuring** the performance, availability and user experience of software application in **real time**
- Goal: ensure that **apps are running smoothly** and **identify and resolve** any **performance or availability issues**
- **Application monitoring** is not an easy job due to the dynamic nature of cloud environments

App Monitoring Metrics and Data

- **Application monitoring** involves **collecting metrics and other data** related to the performance of the app and the overall infrastructure
  - This can include information such as CPU and memory usage, network traffic, etc.
- When data is collected, it can be **analyzed** to **indicate issues** or **potential problems**
- **Alerts** can be set up to **notify** when **thresholds are exceeded** or **specific events occur**

Types of App Monitoring (1)

- **Uptime / availability monitoring**
  - Continuously poll the app to confirm that it's up and responding to the requests it receives
- **Performance monitoring**
  - Ensure your software launches fast and responds to commands in a timely manner
- **Error monitoring**
  - Keep an eye on errors and their frequency

Types of App Monitoring (2)

- **Log monitoring**
  - Collect and analyze logs from various sources
- **Database monitoring**
  - Examine the communication between the app and its database, as well as the database performance
- **Security monitoring**
  - Look out for malware signatures and flag anomalous or suspicious system activity

Monitoring Rooms

- **Software monitoring rooms** (**Network Operations Centers**) are facilities, where engineers monitor and manage software systems and networks
- Engineers typically use
  - Large **screens** to monitor the systems health in real-time
  - **Alarms** in case of emergency / downtime
  - **Notifications**: Slack / SMS / email / phone calls / etc.

App Monitoring Tools – Prometheus + Grafana

- App monitoring data is often collected using **specialized monitoring tools**
- **Prometheus** is a monitoring tool for **storing time series data** like metrics and **Grafana visualizes this data**

tower (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://tower.rhbr-labs.com:443/api/v2/metrics	UP	instance="tower.rhbr-labs.com:443" job="tower"	2.876s ago	76.07ms	

tower-01 (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://172.31.66.203:9100/metrics	UP	instance="172.31.66.203:9100" job="tower-01"	2.565s ago	8.802ms	

tower-02 (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://172.31.65.135:9100/metrics	UP	instance="172.31.65.135:9100" job="tower-02"	80ms ago	8.307ms	

tower-db-01 (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://172.31.64.218:9100/metrics	UP	instance="172.31.64.218:9100" job="tower-db-01"	762ms ago	8.726ms	

More App Monitoring Tools

- **Datadog**
  - Provides comprehensive end-to-end visibility
- **New Relic**
  - Cloud-based monitoring tool
- **Instana**
  - AI-powered observability platform
- **Nagios**
  - Highly customizable open-source monitoring tool
- And many more…

Prometheus

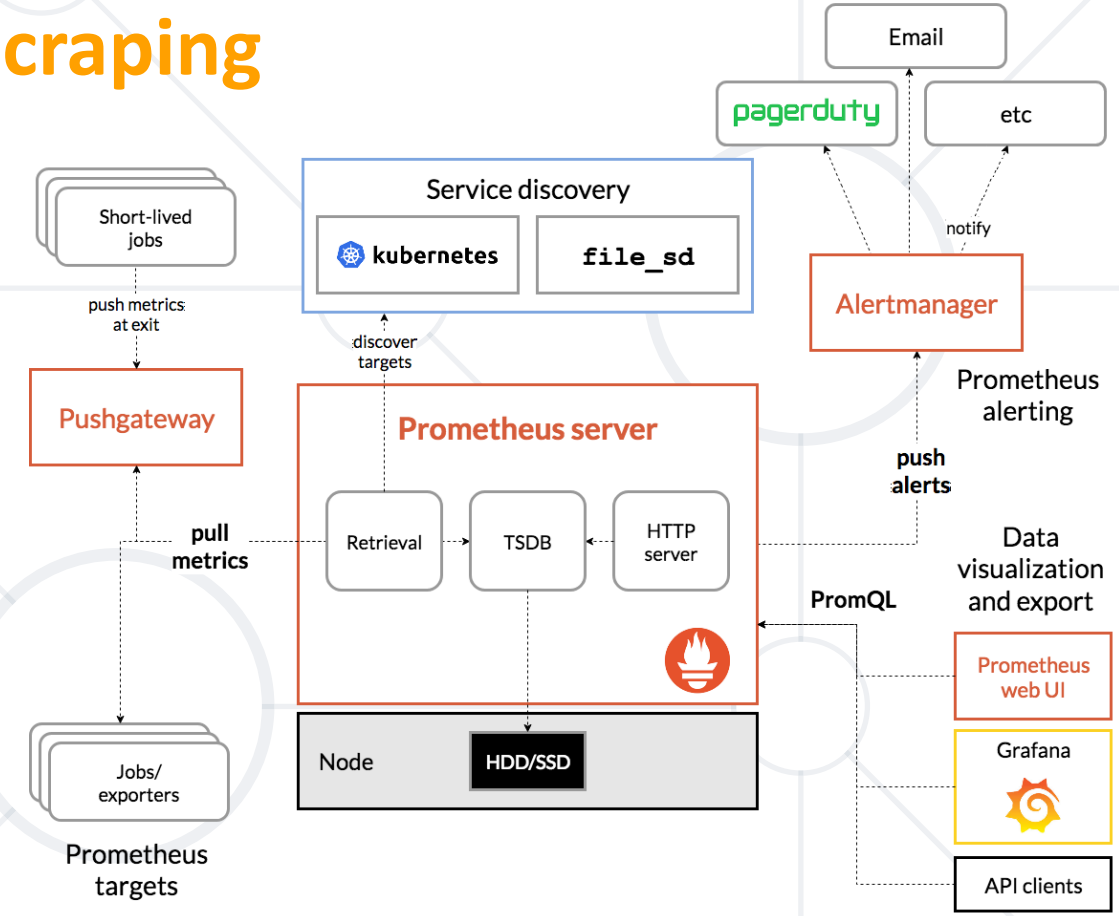
Collect and Monitor App Data and Send Alerts

What is Prometheus?

- An open-source toolkit for **monitoring** and **alerting**, designed to **collect and store metrics** from various sources
- Gains **real-time insights** into the health of an infrastructure
  - Helps quickly identify and resolve issues
- Integrates well with **cloud-native environments** and **modern software systems**
- Live demo: https://prometheus.demo.do.prometheus.io

How Does Prometheus Work?

- **Prometheus** works by **continuously scraping** (pulling) **metrics data** from target systems at regular intervals
  - Can scrape applications, servers, containers, databases, etc.
- **Data** is stored in a **time-series database** and visualized with tools like Grafana
- The database can be queried and analyzed using the **Prometheus query language** (**PromQL**)
- An **alerting system** may also be configured to send notifications



How to Extract App Metrics for Prometheus?

- Use **pre-built exporters**
  - Standalone apps or services that collect metrics from various sources and expose them to Prometheus
    - Examples: Node Exporter (Linux machines), Blackbox exporter (external services)
- **Instrument your app**
  - Use client libraries to collect and expose custom app metrics
- Using the **Pushgateway**
  - Allows you to push metrics to a centralized gateway, which Prometheus can scrape

Prometheus Data Model

- When **Prometheus collects metrics data** from a system, it organizes the **data into individual time series** (data point + timestamp)
- **Data points** are key-value pairs

  cpu_usage 14.04

  Key Value

  - The **key** is called **metric** and describes **what you are measuring** (for example CPU rate or memory usage)
  - The **value** stores the **actual measurement value**, as a number
- You can also provide more details to your metrics using **labels** (optional key-value pairs)

  cpu_usage {core="1", ip="128.0.0.1"} 14.04

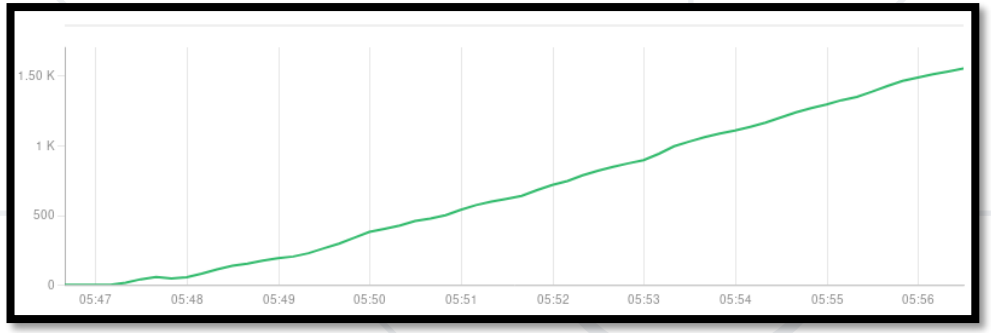
  Key Labels Value

  Metadata

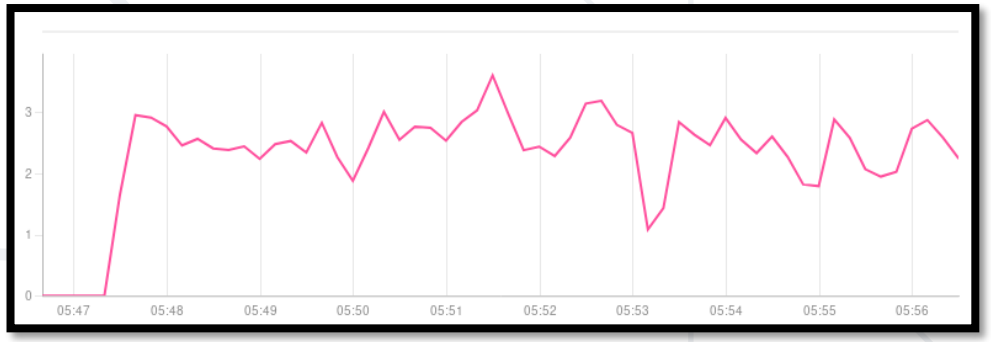
  - For example, if you want to describe CPU rate for a specific core

Metric Types (1)

- **Counter**
  - Only **goes up** or **resets on restart**, e.g., tasks completed, requests served

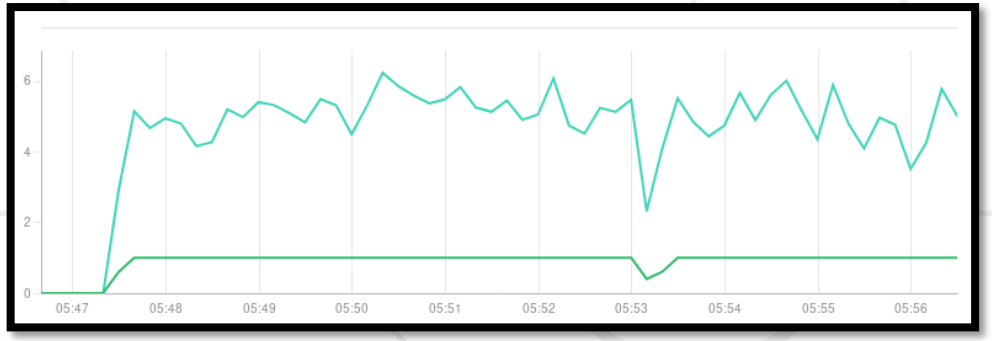


- **Gauge**
  - Can **go up and down**, e.g., temperature, requests

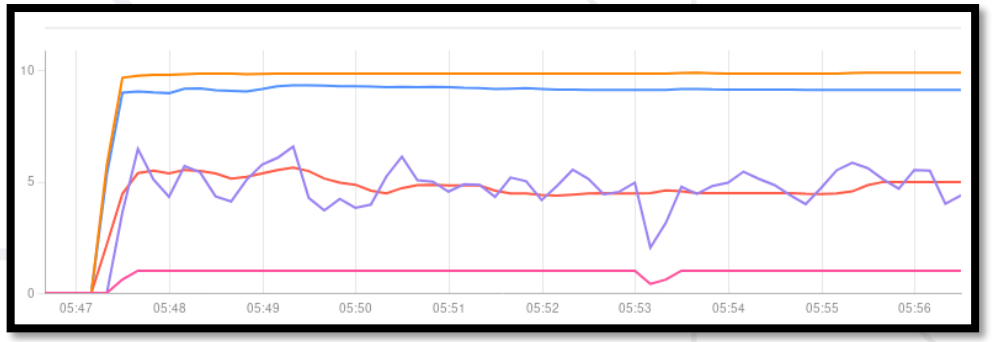


Metric Types (2)

- **Histogram**
  - **Counts observations** (e.g., request durations) in configurable buckets and provides a **sum of all values**



- **Summary**
  - Like histogram, but **calculates configurable quantiles** over a **sliding time window**



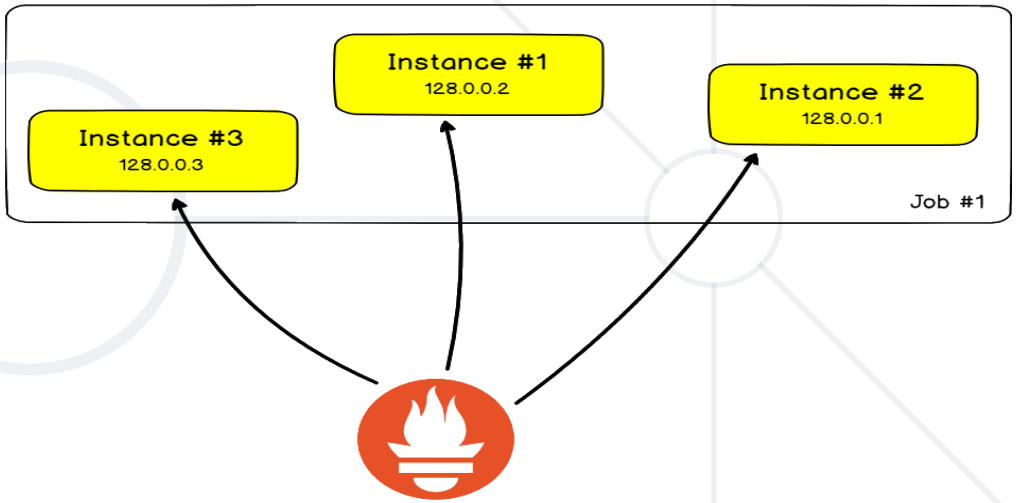
Components (1)

- **Instance** == an endpoint that can be scraped
  - Usually corresponds to a **single process**
- **Job** == a collection of instances
  - Instances should have the same purpose
- Example

```
job: api-server
   instance 1: 1.2.3.4:5670
   instance 2: 1.2.3.4:5671
   instance 3: 5.6.7.8:5670
   instance 4: 5.6.7.8:5671
```

Components (2)

- When a **target** is **scraped**, two additional **labels** are attached – **job** and **instance**
- Each instance scrape adds a **sample** to a set of system time series
  - A **sample** is a single value at a point in time in a time series
  - It consists of a **`float64` value** and a **millisecond-precision timestamp**




```
cpu_usage {job="1", instance="128.0.0.1"}   14.04
cpu_usage {job="1", instance="128.0.0.2"}   12.01
cpu_usage {job="1", instance="128.0.0.3"}   16.03
```

Prometheus Configuration

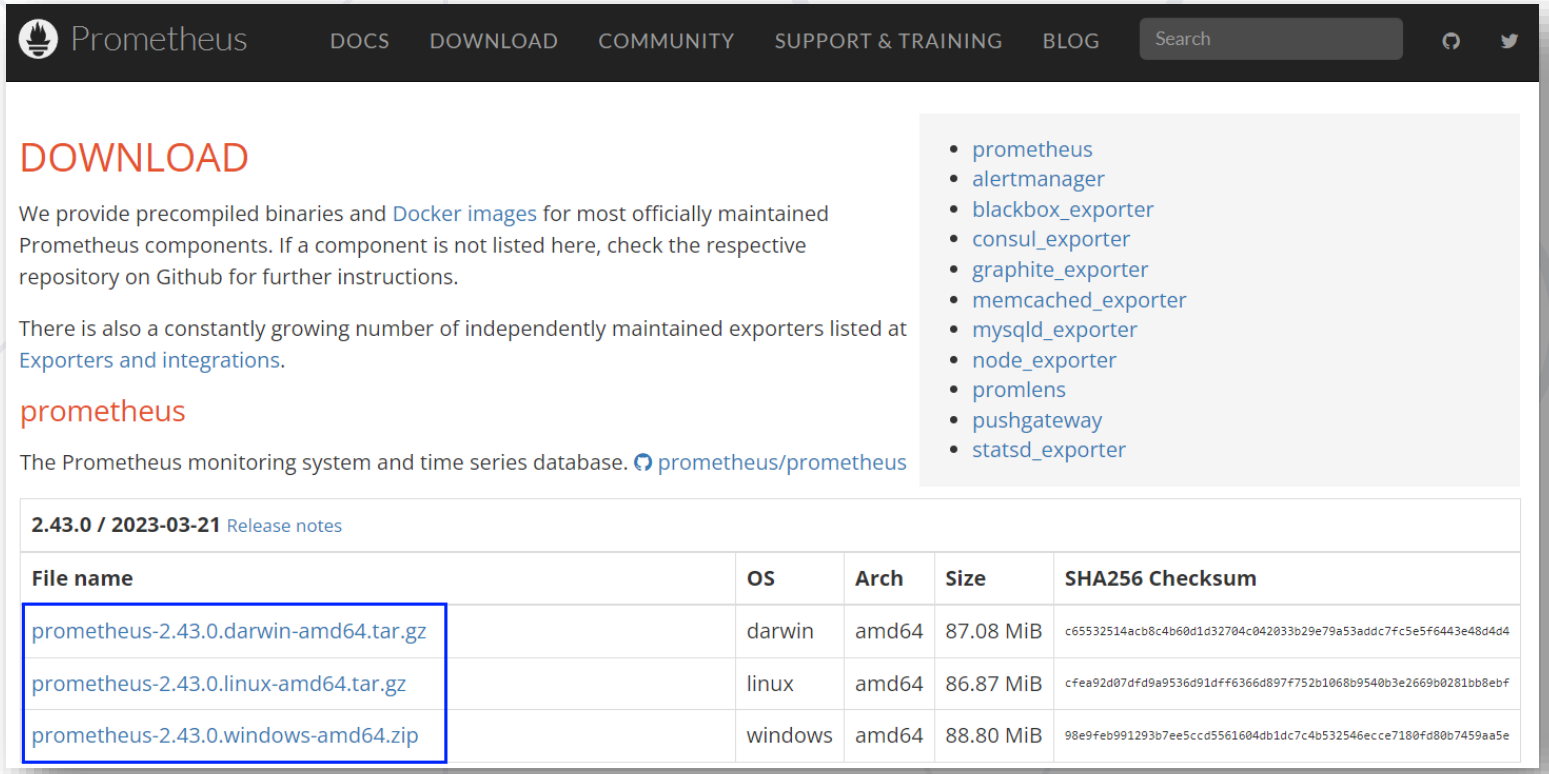
- Prometheus is configured via **command-line flags** and a **configuration file**
- **Configuration file** is used to control scraping and rules
  - It is in **YAML** format
- **Flags** are used to configure immutable system parameters like storage location, amount, etc.

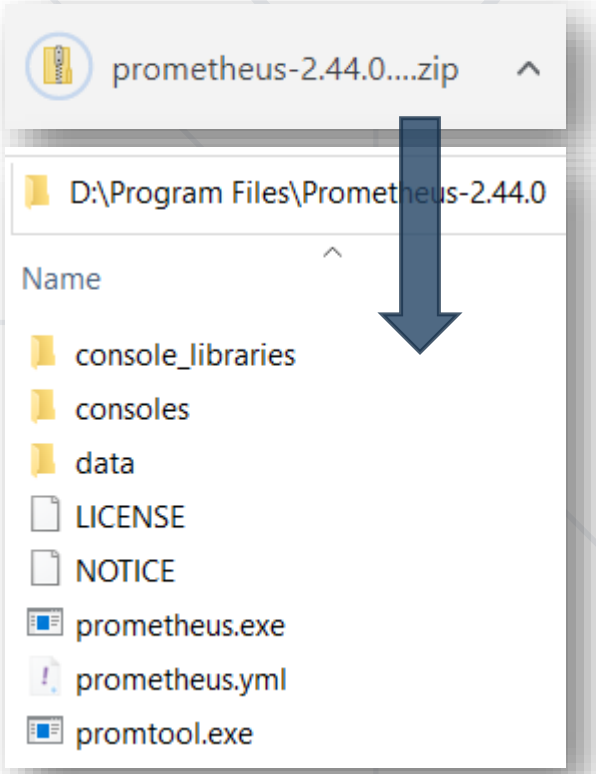
```yaml
global:
  scrape_interval: 1m
  scrape_timeout: 10s
  evaluation_interval: 1m
scrape_configs:
- job_name: prometheus
  honor_timestamps: true
  scrape_interval: 1m
  scrape_timeout: 10s
  metrics_path: /metrics
  scheme: http
  static_configs:
  - targets:
    - 65.19.71.11:9090
```

```
PS D:\Program Files\Prometheus-2.44.0> ./prometheus -h
usage: prometheus.exe [<flags>]

The Prometheus monitoring server


Flags:
  -h, --[no-]help                Show context-sensitive help (also try
                                 --help-long and --help-man).
      --[no-]version             Show application version.
      --config.file="prometheus.yml"
                                 Prometheus configuration file path.
      --web.listen-address="0.0.0.0:9090"
                                 Address to listen on for UI, API, and
                                 telemetry.
      --web.config.file=""       [EXPERIMENTAL] Path to configuration file that
                                 can enable TLS or authentication.
      --web.read-timeout=5m      Maximum duration before timing out read of the
                                 request, and closing idle connections.
      --web.max-connections=512  Maximum number of simultaneous connections.
      --web.external-url=<URL>   The URL under which Prometheus is externally
                                 reachable (for example, if Prometheus is served
                                 via a reverse proxy). Used for generating
                                 relative and absolute links back to Prometheus
                                 itself. If the URL has a path portion, it will
                                 be used to prefix all HTTP endpoints served by
                                 Prometheus. If omitted, relevant URL components
                                 will be derived automatically.
      --web.route-prefix=<path>  Prefix for the internal routes of
                                 web endpoints. Defaults to path of
                                 --web.external-url.
      --web.user-assets=<path>   Path to static asset directory, available at
                                 /user.
```

Install Prometheus

- You can **install Prometheus** easily using a **pre-compiled binary** from here: https://prometheus.io/download/
- **Extract the downloaded archive** to a directory of your choice





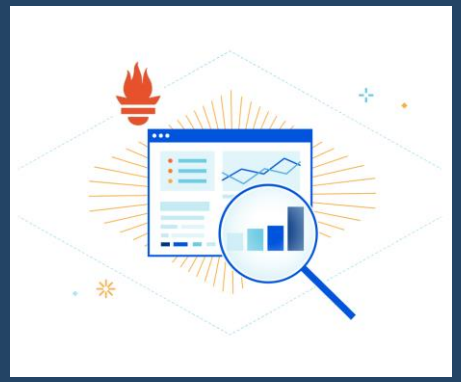
Prometheus Sample Configuration File

- **Navigate to the directory** of the extracted archive and **open the prometheus.yml file** in a text editor
  - This YAML file is the **primary configuration file** for Prometheus
  - You'll need to **modify** it to **define the targets** you want to be scraped
  - Or you can create and use a **new file**
- This YAML file has a sample configuration in which **Prometheus scrapes data from its own server**

prometheus.yml

D: > Program Files > Prometheus-2.44.0 > prometheus-main.yml

```yaml
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]
```



Demo: Prometheus and Blackbox Exporter

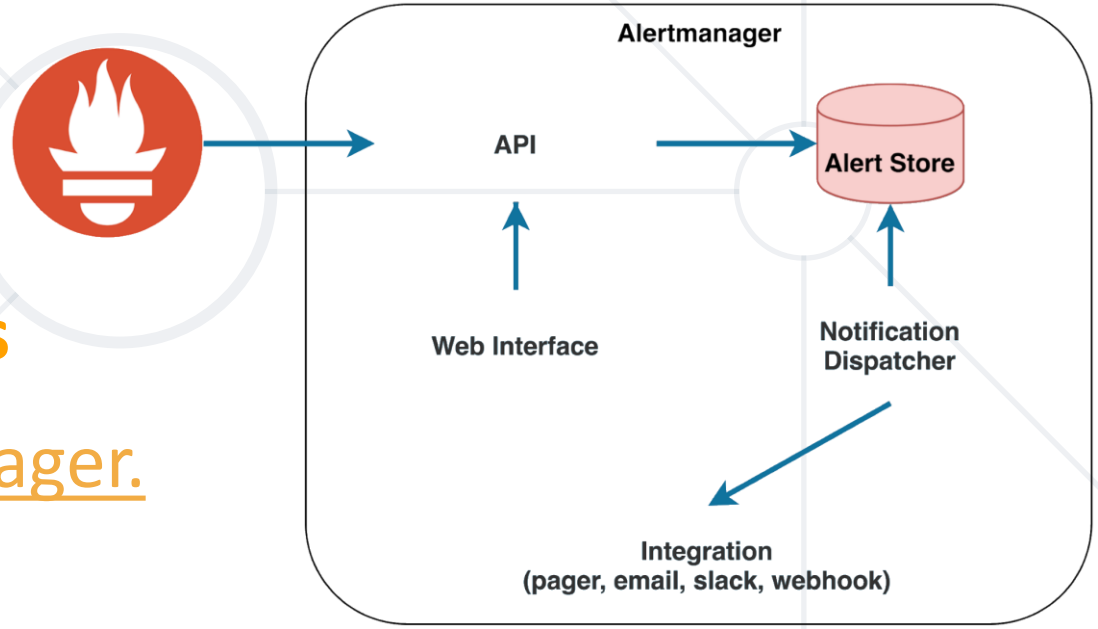
Run Prometheus Server that Monitors SoftUni.org

Alerting

Configure and Receive Notifications with Alertmanager

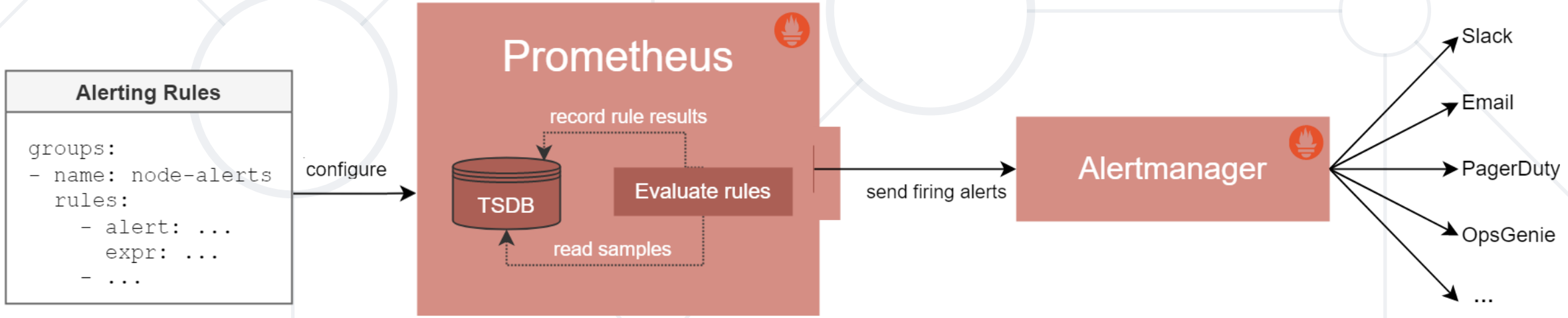
Alerting and Alertmanager

- **Alerting** is a component of a monitoring system that **performs actions** based on **changes in metric values**
- Involves setting up **rules or conditions** that **trigger alerts or notifications** when **specific events or issues occur** in the app
  - High error rates, slow response times, etc.
- **Alertmanager** handles and routs **Prometheus alerts**
- Live demo: https://alertmanager.demo.do.prometheus.io



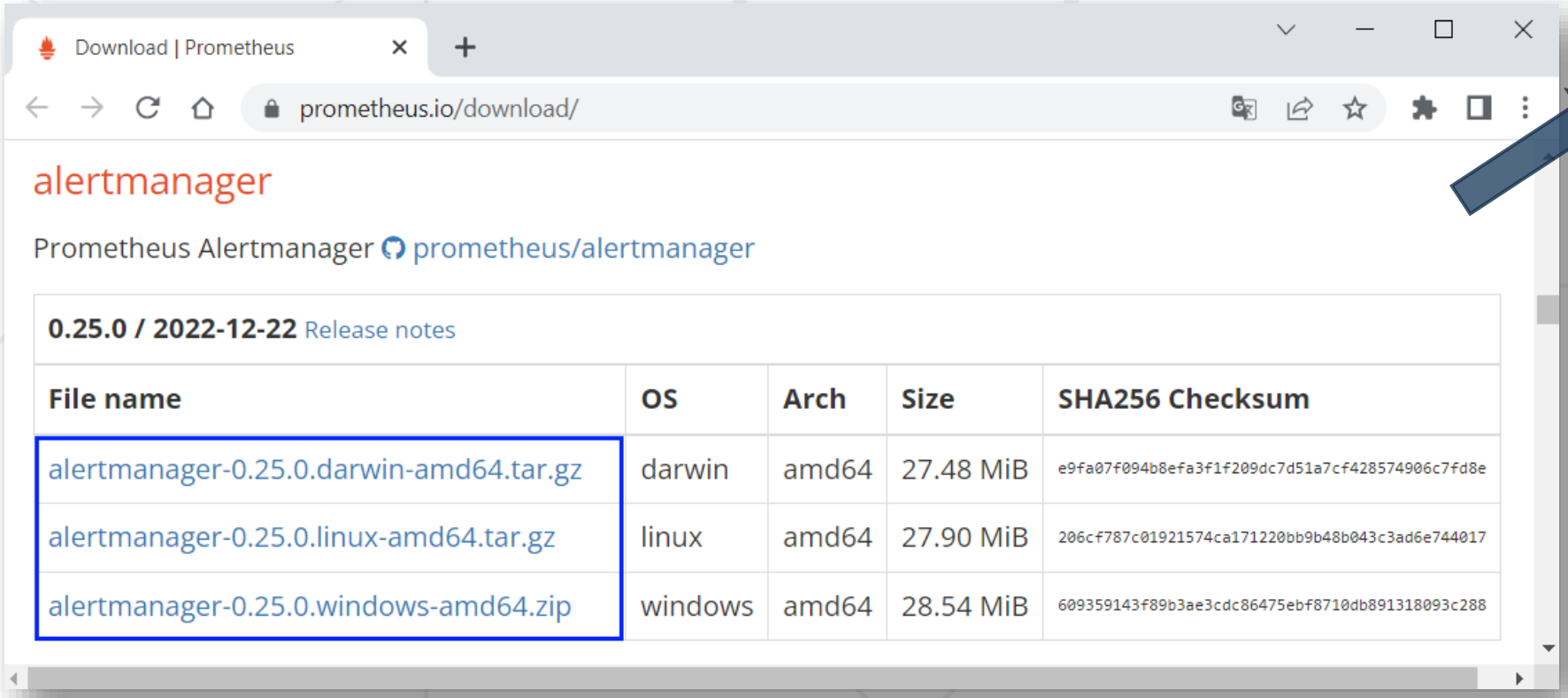
Alerting with Alertmanager

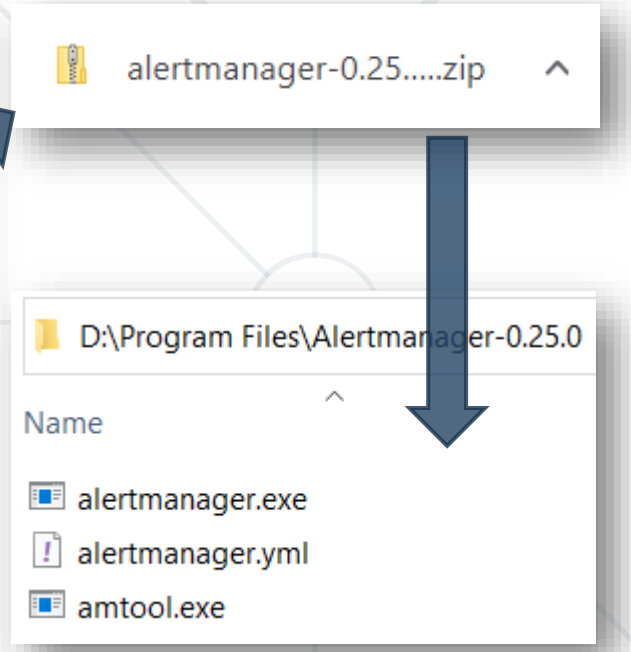
- Separated into two parts
  - **Alerting rules** in Prometheus servers send alerts to an **Alertmanager**
  - **Alertmanager manages alerts** including **silencing** (mute), **inhibition** (suppress), **aggregation** (group) and **sending out notifications**
    - **Notification methods**: email, on-call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



Install Alertmanager

- **Install Alertmanager** using a **pre-compiled binary** from the same URL as Prometheus: https://prometheus.io/download/
- **Extract the downloaded archive** to a directory of your choice





Alertmanager Configuration

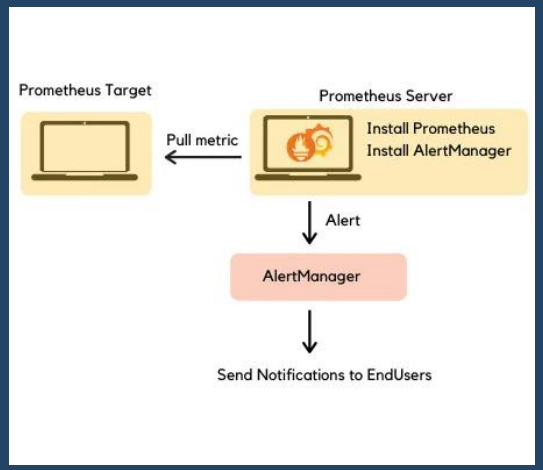
- **Alertmanager** is configured via **command-line flags** and a **configuration file**
  - The **configuration file** is in **YAML format**
- It defines **inhibition rules**, **notification routing** and **notification receivers**

alertmanager.yml

D: > Program Files > Alertmanager-0.25.0 > alertmanager.yml

```yaml
route:
  group_by: ['alertname']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 1h
  receiver: 'web.hook'
receivers:
  - name: 'web.hook'
    webhook_configs:
      - url: 'http://127.0.0.1:5001/'
inhibit_rules:
  - source_match:
      severity: 'critical'
    target_match:
      severity: 'warning'
    equal: ['alertname', 'dev', 'instance']
```

- The "`alertmanager.yaml`" file we have, defines how to route, receive and manage Prometheus alerts



Demo: Prometheus and Alertmanager

Create Alerts for Prometheus Metrics

Grafana

Open-source Analytics & Monitoring Solution

What is Grafana?

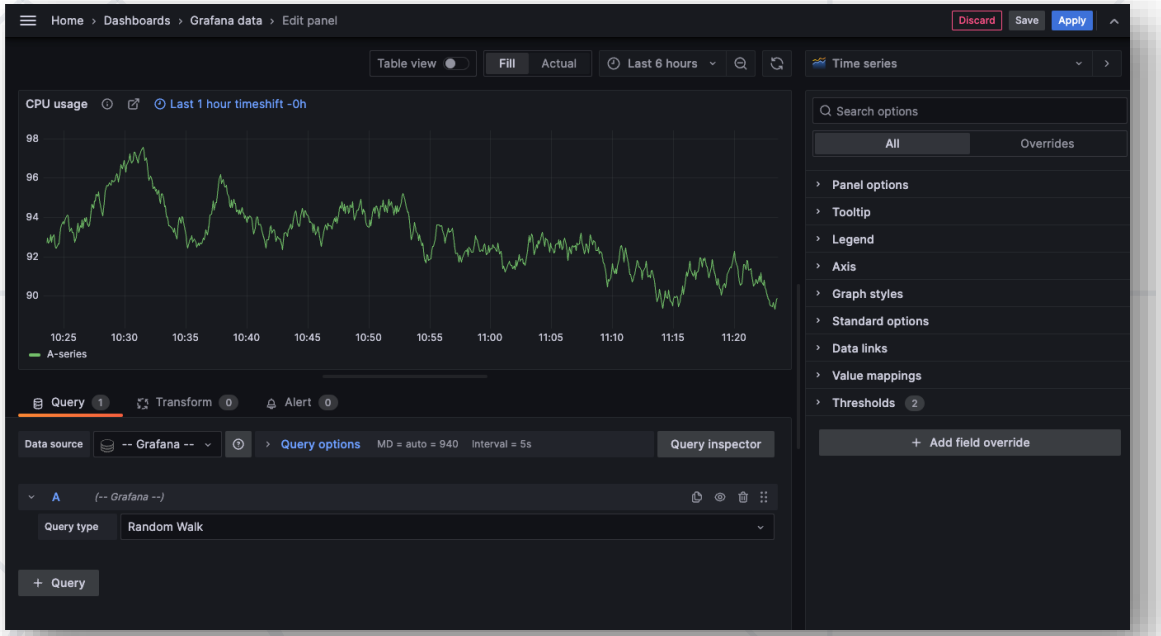
- **Grafana** is a **complete observability stack**
- Allows users to **visualize and analyze data** (metrics, logs, etc.) from **various sources** in **real-time**
- You can also create flexible **dashboards** and **alerts**
- Offered in 3 variants: open source, cloud and enterprise
- Supports **plugins** and **extensions** for additional functionalities and integration with other tools
- Live demo: https://play.grafana.org/

Dashboards

- Grafana **dashboard** == a **collection of panels** that **display data** from one or more **data sources**

Panels

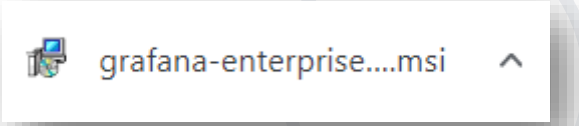
- **Panel** == a **visualization of a metric** or set of metrics
  - Types
    - Graphs
    - Tables
    - Gauges
    - Singlestat



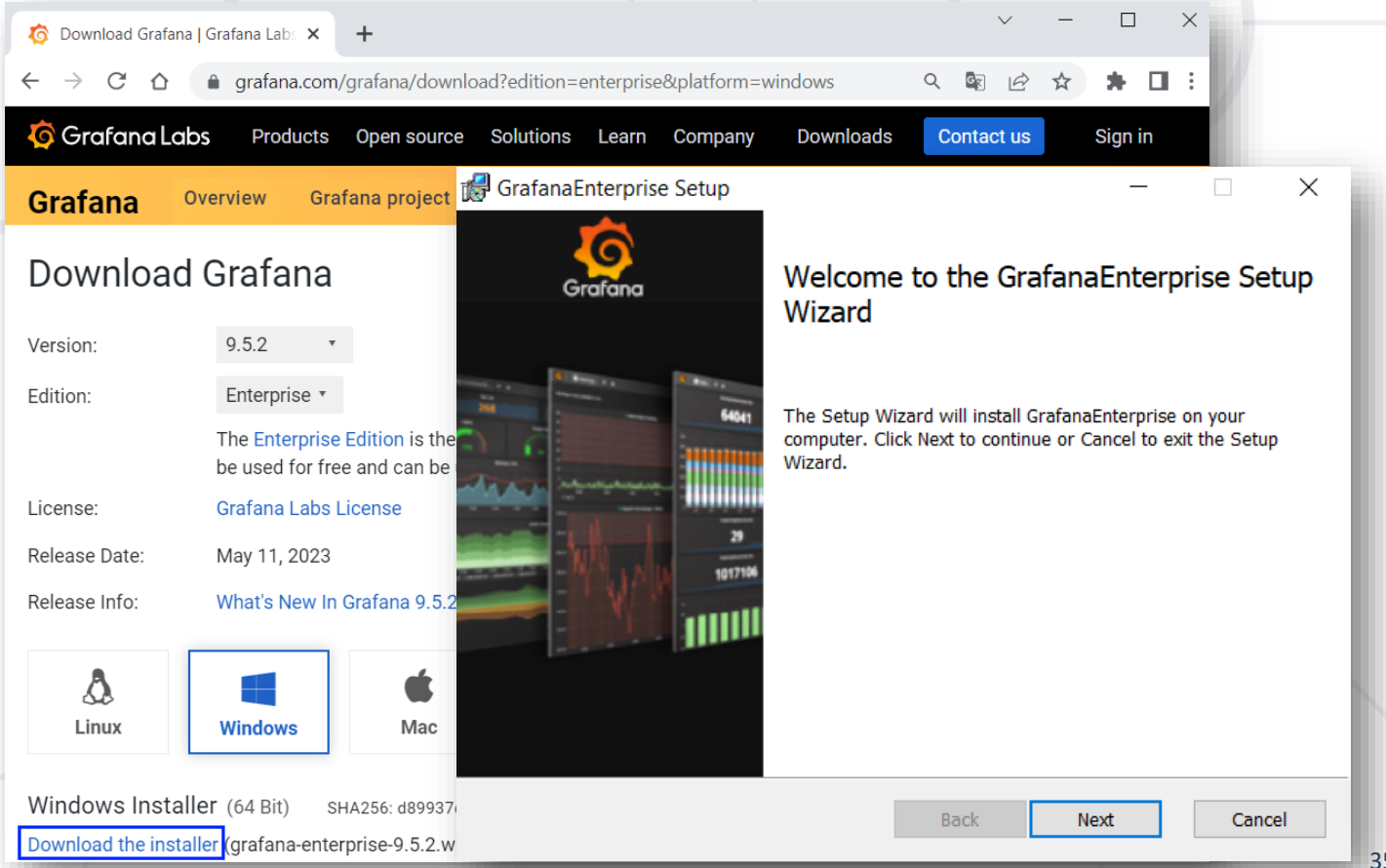
- **Panels** connect via query to a **data source** that you want to **visualize**
- Grafana provides many **options** for customizing the way **data is displayed in a panel**
- There are **different panel layout** options for customizing dashboards

Install Grafana

- To install Grafana, **download the installer file** from here https://grafana.com/grafana/download?platform=windows
  - Choose the latest version, "Enterprise" edition and your OS
- Click on the **installer file**



- Follow the **setup wizard** to install Grafana



Demo: Grafana and Prometheus

Visualize Metrics in a Dashboard

Elastic Stack

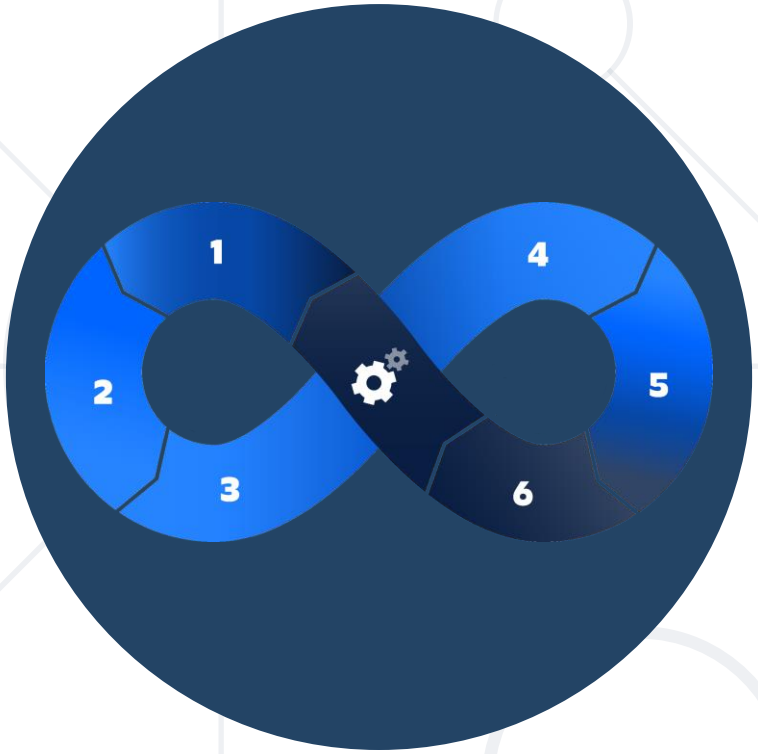
Flexible search and monitoring solution

What is Elastic Stack?

- **Elastic Stack** == set of open source products, designed to
  - Help users **take data in any kind of format from any type of source**
  - **Search**, **analyze** and **visualize** this data
- Enables **log management**, **monitoring** and **analysis**
  - Detects anomalies in app performance

Components

- **Elasticsearch**
  - Log management
- **Logstash**
  - Centralizing, transforming and storing logs
- **Kibana**
  - Visualizing and exploring data, using dashboards
  - Enables system performance monitoring
- **Beats**
  - Collect and send various types of data from servers, systems and apps to different destinations



DevSecOps Tools

Overview

DevOps vs DevSecOps

	DevOps	DevSecOps
Focus	Increasing quality and speed of software development and delivery	Secure software development processes by integrating security
Process	CI/CD	CI/CD + additional security-related processes
Activities	Continuous testing, development and monitoring QA tasks	Precommit, commit-time, build-time, test-time, deploy time checks of code

Static vs Dynamic Analysis in DevSecOps

- Static Analysis
  - Used for identifying **security vulnerabilities**
  - Analysis of the code **without executing** it
  - Catch potential security issues **early** in the development stage
- Dynamic Analysis
  - Used for identifying **security weaknesses**
  - Analysis of the code by **executing the app** in real or simulated environment
  - Detect security issues **at runtime**

DevSecOps Tools for Static Analysis

- **SonarCube**
  - Open source platform for continuous code inspection
- **Fortify**
  - Scan source code to identify security issues and provide a detailed report
- **Veracode**
  - Cloud-based tool for code analysis and generation of suggestions for prioritizing and fixing security issues
- **Chekmarx**
  - Scan for common coding practices that lead to security issues

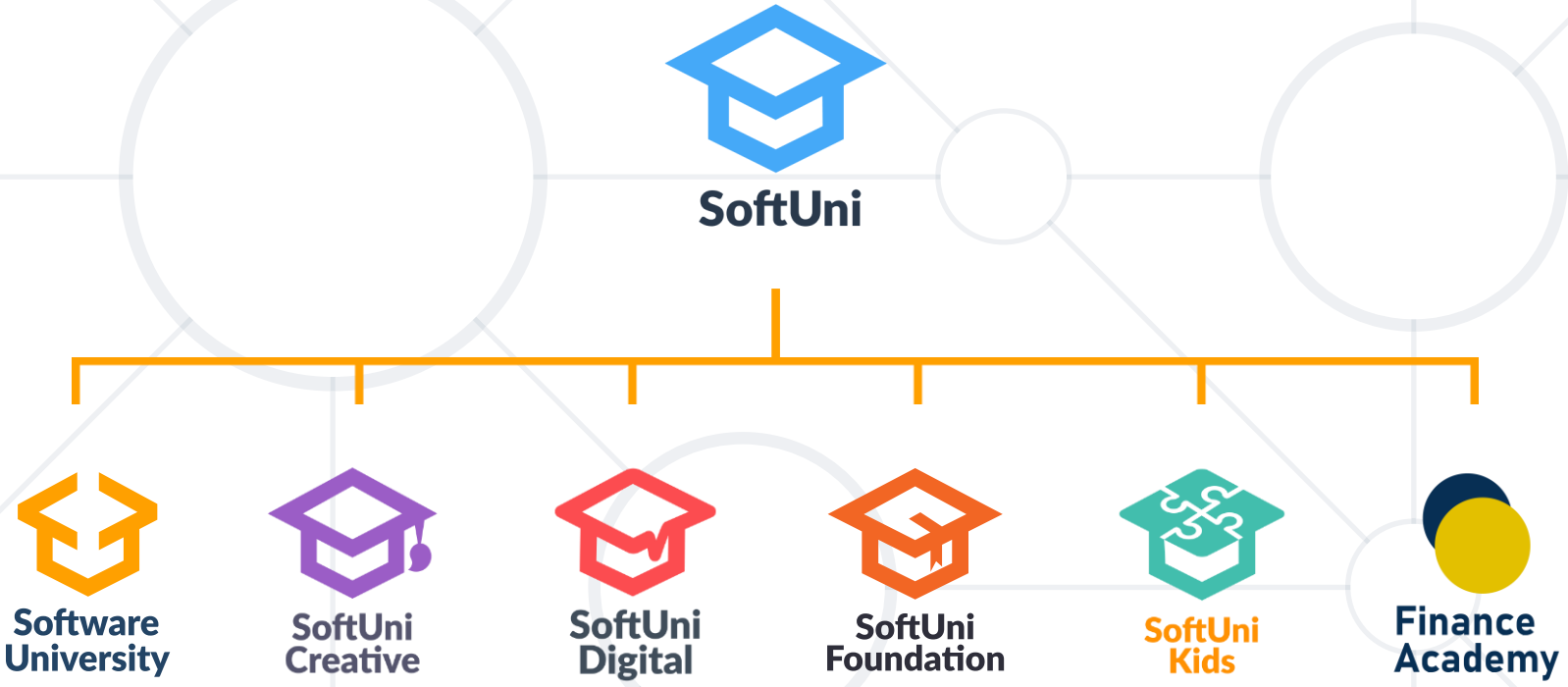
DevSecOps Tools for Dynamic Analysis

- **OWASP Zed Attack Proxy**
  - Open source tool for simulation of attacks on apps
- **Burp Suite**
  - Intercept, manipulate and analyze web app traffic
- **Acunetix**
  - Scan web apps for security vulnerabilities
- **WebInspect**
  - Identify potential weaknesses and generate suggestions on fixing them

Summary

- **Application monitoring** tracks and measures performance, availability and user experience of software apps
- The **Prometheus monitoring tool** stores metrics
- **Alerting** in a monitoring system takes action based on metric value changes
- **Alertmanager** manages and routes **Prometheus alerts**
- **Grafana** offers real-time visualization and analysis of Prometheus **data**
- **ElasticStack** is a powerful set of tools for collecting, storing and analyzing data, providing monitoring of apps and systems
- **DevSecOps tools** for static and dynamic analysis

Questions?



SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

POKERSTARS
POKER | CASINO | SPORTS
a Flutter International brand

INDEAVR
Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

createX

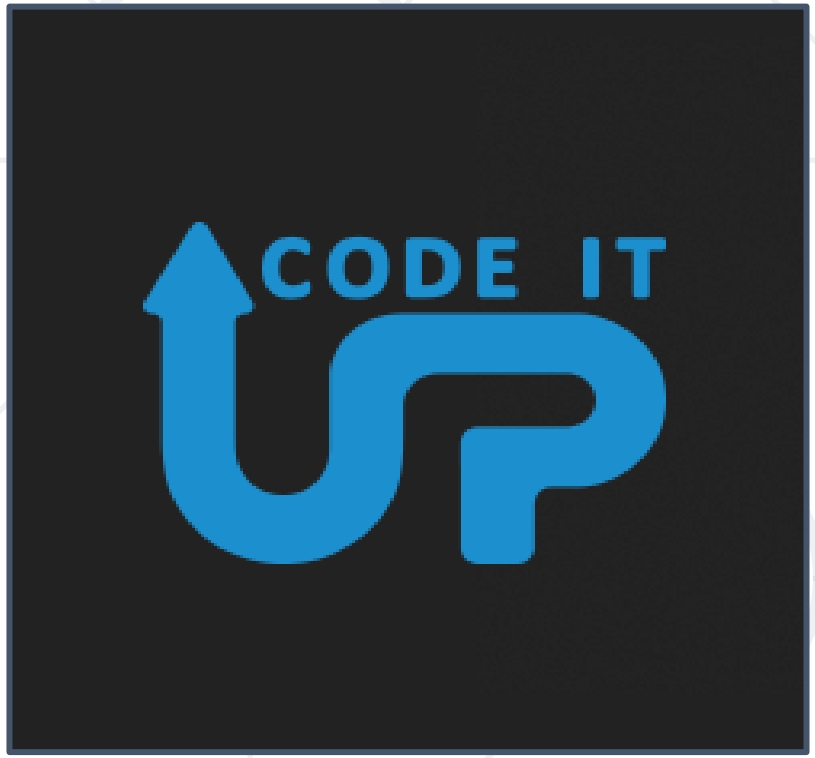
Postbank
Решения за твоето утре

BOSCH

DXC TECHNOLOGY

SmartIT

Educational Partners



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://about.softuni.bg/
- © Software University – https://softuni.bg

Trainings @ Software University (SoftUni)

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