

eh1act06 - System Monitoring

1- Objectives of activity eh1act6

- a) Deployment of Prometheus and Grafana tools using Docker containers
- b) Basic system monitoring: CPU, RAM and Disk.

2- Concepts to be developed in this activity

a) Some ideas about monitoring:

- Monitoring a system consists of continuously tracking the state of a machine's resources and its system.
- The purpose of monitoring is to check whether the system is working correctly, whether it is at its limit and needs to be upgraded, to check for anomalies causing malfunctions in some of its parts, to improve the configuration of the system/applications/services/hardware, etc...
- In general, any computing device or any program can be monitored. Typically monitored items include:
 - Servers and client machines (RAM, CPU, Disk, Network...)
 - Services (Apache, SSH, DHCP, DNS, etc...)
 - Mobile phones/Tablets
 - Printers
 - Hubs/Switches/Routers
- Typically, at a minimum, CPU usage and the amount of used and free RAM memory must be monitored. It is also common to monitor the status of storage devices and the status of network connections.

b) **Metrics** are used for monitoring. A metric is data about the usage of a particular resource. For example, the % of free RAM memory, the % of CPU time used, the number of users connected to a system, the load time of a web page, the number of times a file in a shared folder has been accessed, ...

With the right monitoring software, you can:

- **Collect** metrics from your systems, services and applications.
- **Store** metrics as **time-series** data.
- **Visualize** metrics thorough graphs, tables and indicators.
- Run **queries** on collected metrics.

c) **Prometheus:**

- It is a tool for collecting, stores metrics as time-series visualizing and answering queries about collected metrics of the system such as CPU usage, RAM memory, disk space, network access..
- Components of Prometheus:
 - **Exporter** → it collects and exposes metrics to server.
 - **Server** → It retrieves metrics exposed by the exporter. Also, it stores metrics as time-series, visualizes metrics and it can run queries about collected metrics.
- It allows exporting and sharing data with other specialized applications that have better visualization or data analysis tools such as **Grafana**.

d) Grafana:

- It is a tool that allows visualizing data collected by **Prometheus** (or other similar monitoring tools) in a way that is easy for the administrator to interpret.
- It allows the creation and customization of Dashboards (control panels) with multiple charts in various formats, all types of indicators, and custom tables that allow the administrator to clearly view all the data needed to monitor a system or network.
- It also allows generating and configuring system or network alerts.

e) Prometheus + Grafana:

- **Prometheus** to collect metric and store them as time-series.
- **Grafana** to visualize metrics.
- **Prometheus** and **Grafana** are commonly deployed together using **Docker**, where each component runs as an isolated container; with a simple **docker-compose.yml** file, you can spin up the entire monitoring stack in minutes.

3- System monitoring with Grafana and Prometheus

3.1- Deployment of the Prometheus and Grafana via Docker containers

a) Create a folder called **eh1act06**. Access to **eh1act06**.

b) Download a file called **master.zip** found at:

<https://github.com/stefanprodan/dockprom/archive/refs/heads/master.zip>

c) Uncompress **master.zip**.

d) Verify that a folder named **dockprom-master** has been created.

e) Access to **dockprom-master**.

f) Check that a file **docker-compose.yml** exists in the folder.

g) Deploy and start the **Grafana** and **Prometheus** monitoring tools using containers. Run:

docker compose up -d

It takes a few minutes to complete this task. *Take it easy!!!*

h) Verify that the **Prometheus** and **Grafana** containers are running. Execute:

- **docker compose ps -a | grep prometheus** → Check that the **Prometheus** (server part) docker is running
- **docker compose ps -a | grep nodeexporter** → Check that the **exporter** docker is running.
- **docker compose ps -a | grep grafana** → Check that the **grafana** docker is running

3.2- Accessing metrics

- a) Access **Grafana** using a browser by connecting to the IP address of your virtual machine on port **3000**, which is where **Grafana** listens.
- b) Log in to the application using the default username **admin** and the default password **admin**.
- c) Create (and confirm) a new password for the application (suggested: **fjeclot**).
- d) Go to:
 - Select **Dashboards** (Left bar) → click button **New** (right upper side) → select **Import**.
 - On the next page, in **Load** → Indicate that you want to import the dashboard with ID **1860** and then click **LOAD**.
 - On the next page, click **IMPORT**.
 - Verify that a **Dashboard** loads that allows you to view metrics from the local system.
 - Configure the **Dashboard** to display metrics from the last **5 min.** and refresh the page every **5s**.
- d) Verify that you can see the basic local metrics (CPU, RAM, SWAP, disk usage, network traffic, etc.....) of your virtual machine.
- e) Configure the **Dashboard** to display metrics from the last **5 minutes** and refresh the page every **5 seconds**.

4- Stress tests and system monitoring

- a) Install a program called **stress**. Run: **sudo apt-get install stress**
- b) Stress the **CPUs** for **30 seconds** with **4 processes** performing complex mathematical calculations. Run the command:

sudo stress -c 4 -t 30
- c) Check with **Grafana** how the value of **CPU Busy** increases and the **CPU Basic** graph changes.
- d) Wait until **CPU Busy** and **CPU Basic** graph go back to their previous states.
- e) Stress the **Memory** usage for **30 seconds** with **4 processes** making intensive use of **memory**. Run:

sudo stress -m 4 -t 30
- f) Check with **Grafana** how the value of **CPU Busy** and **RAM Used** increase.
- g) Wait until **CPU Busy** and **RAM Used** go back to its previous state.
- h) Create a **10GiB** file running: **dd if=/dev/zero of=bigFile.img bs=10M count=1000**
- i) Check with **Grafana** how the value of **Root FS Used** increases and **CPUDisk Space Used Basic** graph changes.
- j) Remove **bigFile.img** and check how **Root FS Used** and **CPUDisk Space Used Basic** graph go back to their previous states.

5- Checking your system

- 1- Show the properly configured **Dashboard** for monitoring your system.
- 2- Stress the **CPUs** for **20** seconds with **2** processes and check with **Grafana** how the value of **CPU Busy** increases and the **CPU Basic** graph changes.
- 3- Stress the **Memory** usage for **15 seconds** with **2 processes** and check with **Grafana** how the value of **CPU Busy** and **RAM Used** increase.
- 4- Create a **5GiB** file and check with **Grafana** how the value of **Root FS Used** increases and **CPUDisk Space Used Basic** graph changes.