PROJECT

Multi-tenant cybersecurity operations centre designed
as an open cloud service with machine learning elements

Linked to work package KPB1 – Design of the virtualization solution

Linked to output
V3 – Description of the selected solution meeting the specified criteria

type – documentation

Analysis I.

Analysis of network monitoring tools for the needs of the NOC

CONTENTS

1 Introduction 4
1.1 Methodology 4
1.2 Benefits of the project 4
2 Theoretical analysis 5
2.1 Network operation center (NOC) / SOC 5
2.2 Operational processes in the NOC 5
2.2.1 Importance and benefits of the NOC 6
2.3 Monitoring 6
2.3.1 Types of monitoring 6
2.3.2 Methods of data acquisition 7
2.3.3 Automation and integration 7
2.4 Zabbix 8
2.4.1 Key features of Zabbix 8
2.4.2 Zabbix architecture 8
2.4.3 Specific functions of Zabbix 9
2.4.4 Licensing and community 9
2.5 LibreNMS 9
2.5.1 Key features of LibreNMS 9
2.5.2 LibreNMS architecture 10
2.5.3 Differences compared to Zabbix 10
2.5.4 Licensing and community 10
2.6 Prometheus 10
2.6.1 Key features of Prometheus 10
2.6.2 Prometheus architecture 11
2.6.3 Specific functions of Prometheus 11
2.6.4 Integration with other systems 11
2.6.5 Licensing and community 11
3 Basic installation and configuration of the tools 12
3.1 Zabbix 12
3.1.1 Notification settings 13
3.2 LibreNMS 14
3.3 Prometheus 15
4 Comparative analysis 19
4.1.1 Scoring 20
5 Conclusion 23
6 Sources 24

1 INTRODUCTION

Modern IT infrastructures are complex networks of devices, applications and services that must operate with maximum reliability and efficiency. To ensure their continuity and the rapid resolution of problems, it is essential to introduce systems for monitoring their operation. Monitoring enables organizations not only to detect potential problems in real time, but also to analyse historical data to improve performance and operations.

In this project, we focus on the creation and implementation of a Security Operations Center (SOC) or Network Operations Center (NOC). This project responds to the need to monitor and secure the department's IT infrastructure, with the prospect of extending it to the entire faculty.

The aim is to build a fully functional operations centre that will monitor key network components and provide regular reports on the state of the infrastructure. In addition, the centre will be capable of detecting security incidents and providing recommendations for their resolution.

1.1 Methodology

The implementation of the project comprises the following steps:

- Requirements analysis: At the beginning of the project, requirements for the NOC were defined, including specifications of the monitored components, security functions and scalability of the solution.
- Drawing up the criteria: To compare the monitoring tools (Zabbix, LibreNMS, Prometheus), criteria with assigned weights were created.
- Testing of the tools: Each tool was first installed locally using Docker for testing purposes. Subsequently, these tools were deployed in the OpenStack environment on the university cloud.
- Comparison and evaluation: The tools were compared on the basis of the defined criteria, to which weights were assigned, and their scores were used to select the most suitable solution.

1.2 Benefits of the project

The implementation of the NOC at KIS/FRI will bring improvements in the following areas:

- Monitoring of the network and devices,
- Predictive analysis for the detection of potential problems,
- Provision of regular reports on the state of the network.

In the following chapters, we focus in detail on the theoretical basis of monitoring, the characteristics of the individual monitoring tools and an analysis of their use in this project.

2 THEORETICAL ANALYSIS

2.1 Network operation center (NOC) / SOC

A Network Operations Center (NOC) is a key element in ensuring the efficient management of IT infrastructure. It is a central point of monitoring that provides oversight of networks, servers, applications and other IT systems [1]. The role of the NOC is to minimize outages, increase availability and ensure the smooth operation of the organization's entire infrastructure [3].

The basic idea of the NOC is the consolidation of data from all monitored components into a single operations centre. This makes it possible to monitor the state of the infrastructure in real time, identify potential problems and respond to incidents before they affect users [1][4].

The NOC is designed not only to respond to current problems, but also to analyse long-term trends that may indicate the need to optimize or expand the infrastructure [6].

The operation of a NOC requires a combination of functional systems and qualified personnel. These two aspects complement each other to ensure maximum efficiency and reliability [3].

1. **Technologies in the NOC:** The technological background of the NOC includes:
    - Monitoring tools: Software such as Zabbix, LibreNMS and Prometheus enables the monitoring of system performance, the analysis of metrics and the generation of alerts [5][6].
    - Databases and analytical systems: These solutions collect and analyse data to identify trends and predict problems [4].
    - Communication tools: Integrated systems enable the rapid exchange of information between team members and automated notifications [7].
    - Backup infrastructure: Servers, UPS devices and redundant network components minimize the risk of outages [1].

2. **Team of experts:** The NOC is usually staffed by a team that includes:
    - Operators: The first point of contact in monitoring and basic problem analysis [6].
    - Analysts: Specialists in detailed diagnostics and the design of solutions [5].
    - Technical engineers: People who implement corrective measures and optimizations [7].

2.2 Operational processes in the NOC

The operation of the NOC comprises a wide range of activities that work together to ensure the reliability of IT systems :

1. Real-time monitoring: Continuous monitoring of system and network performance. Automated tools generate alerts when various deviations are detected. [3].
2. Analysis of historical data: Collection and processing of long-term data to identify trends and predict possible problems. This analysis helps to prevent outages and improve infrastructure performance [4][6].
3. Incident handling: When a problem is detected, an incident management process is triggered, which includes diagnostics, remediation and documentation. This process is supported by ticketing systems that track the status of problem resolution [5].
4. Capacity planning: On the basis of data analysis, the NOC plans the expansion of resources to meet the growing needs of the organization [6].
5. SLA monitoring: Ensuring compliance with the agreed service levels (Service Level Agreements) and preparing regular reports for management and clients [7].

2.2.1 Importance and benefits of the NOC

Deploying a NOC brings organizations a number of advantages, including:

- Increased availability: The NOC minimizes outages and ensures the continuous operation of IT systems [5].
- Greater security: Thanks to advanced monitoring and rapid response to incidents, the NOC helps protect the infrastructure against cyber threats [4].
- Efficient resource management: Process automation enables the optimization of costs and resources [7].
- Decision support: Analysis of data from the NOC provides management with important information for strategic planning [3][6].

2.3 Monitoring

Monitoring is the systematic observation and evaluation of IT infrastructure with the aim of ensuring its optimal performance, availability and security. It is a key element in preventing problems and increasing the operational efficiency of IT systems [1].

The main task of monitoring is to collect and analyse data on the various components of the infrastructure, including networks, servers, applications and cloud environments. This information helps to identify potential problems, uncover weak points and make data-driven decisions [3][4].

2.3.1 Types of monitoring

Monitoring can focus on various aspects of the IT environment, which allows flexible adaptation to the needs of the organization:

1. **Network monitoring:** Network monitoring includes tracking throughput, latency, packet loss and the availability of network devices such as switches, routers and firewalls. It uses protocols such as SNMP and tools such as Paessler PRTG, which enable detailed monitoring of network traffic and the identification of bottlenecks [2][5].

2. **Server monitoring:** This type of monitoring tracks the state and load of servers, including CPU utilization, memory, disk space and hardware temperature. Tools such as Zabbix and Nagios provide detailed information on server performance and help to prevent server overload [3][6].
3. **Application monitoring:** Application performance monitoring covers response time, error messages, availability and the resource usage of applications. Tools such as Dynatrace provide a detailed view of application performance and help to identify problems at code level, which improves the user experience [4][7].
4. **Cloud monitoring:** Cloud monitoring focuses on tracking the performance and availability of cloud services, including virtual servers, containers and databases. Tools such as Prometheus and Grafana enable detailed monitoring of cloud systems with an emphasis on scalability and availability [5][8].
5. **Security monitoring:** Security monitoring identifies and tracks threats, unauthorized access and suspicious activities. It provides key information for the prevention of cyber attacks and the protection of sensitive data [6][9].

2.3.2 Methods of data acquisition

- SNMP (Simple Network Management Protocol): SNMP is one of the most widely used protocols for collecting data on network devices. It enables the monitoring of metrics such as bandwidth, port status and device status, and provides them in real time [1][7].
- API-based monitoring (API interfaces): Many modern applications and systems provide APIs that enable direct access to metrics and specific data. This method is often used in cloud environments and modern applications [8].
- NetFlow/sFlow/jFlow: These protocols enable the monitoring of network traffic. They provide detailed data on the sources, destinations and volume of traffic in the network. They are ideal for the analysis of network flows [5][9].
- Syslog: System messages and events can be sent to a centralized logging server, where they are analysed for diagnostic and security purposes [3].
- WMI (Windows Management Instrumentation): Used to monitor devices running Windows. It makes it possible to obtain information on processes, performance metrics and system configurations [4].
- Logs: Logs provide detailed information on activities in the system, including errors, warnings and events. Their analysis is key to diagnosing problems and improving security [4][9].
- Agents: Software agents installed on monitored devices collect data on system performance and status. They are capable of providing detailed metrics that improve the accuracy of monitoring [3].
- Packet capture: Tools such as Wireshark enable the monitoring of detailed information on network traffic. This method is useful in diagnosing and analysing security problems [6].

2.3.3 Automation and integration

Modern monitoring systems place emphasis on automation and easy integration with other tools, which increases efficiency and reliability:

- Automatic alerts: Systems automatically notify when threshold values are exceeded or anomalies occur, which enables a rapid response [5][6].
- Self-healing: Some advanced tools can automatically perform corrective measures, such as restarting services or applying patches.
- Integration with other platforms: Tools such as Prometheus integrate with Grafana for data visualization or with ticketing systems for incident management [7][8].

2.4 Zabbix

Zabbix is an open-source tool designed for monitoring IT infrastructure, including networks, servers, virtual machines, cloud platforms and applications. With its ability to collect, store and analyse huge volumes of data, Zabbix is suitable for organizations that need a robust solution for monitoring performance and availability [11].

2.4.1 Key features of Zabbix

Real-time monitoring: Zabbix makes it possible to monitor millions of metrics in real time from servers, network devices and applications. This process is supported by a high frequency of data collection and immediate data processing [12].

Flexible notification mechanism: Notifications can be sent via e-mail, SMS, webhooks or integration with external applications such as Slack or PagerDuty. Zabbix makes it possible to set conditions for generating notifications for almost any scenario [13].

Automatic discovery: Automatic detection of network devices and services minimizes the time needed to configure monitoring. This process includes the detection of new devices on the basis of rules, which simplifies the expansion of the monitoring environment [11].

Distributed monitoring: Zabbix supports centralized administration via a web interface, which makes it possible to monitor remote sites without the need for physical presence [14].

Support for various databases: The diversity of database support, including MySQL, PostgreSQL, Oracle, IBM DB2 and SQLite, provides flexibility in the choice of infrastructure and its integration [12][15].

2.4.2 Zabbix architecture

Zabbix is designed as a modular platform that contains the following main components:

1. **Zabbix Server:** The main component responsible for processing data, performing calculations, generating notifications and distributing them [12].
2. **Zabbix Agents:** Client programs installed on monitored devices collect data on device performance and status. The agent provides support for operating systems such as Linux, Windows and UNIX [14].

3. **Proxy:** Proxy servers mediate communication between agents and the main server, which increases efficiency when monitoring large and geographically distributed environments [13].
4. **Database:** Central storage for configuration data, historical metrics and logs [15].
5. **Web interface:** Front-end application for configuration, data access and visualization [11].

2.4.3 Specific functions of Zabbix

- **Support for multiple protocols:** In addition to its own agents, it supports monitoring via SNMP, IPMI, JMX, SSH and Telnet, which increases versatility and adaptability [12][13].
- **Scalability:** The ability to handle millions of metrics per second makes Zabbix suitable for monitoring large and dynamic environments [14].
- **Security:** It supports encrypted communication between components and advanced authentication, which minimizes the risk of compromise [15].
- **Data visualization:** Dashboards, maps and graphs provide clear and rapid analysis of key metrics [11].

2.4.4 Licensing and community

Zabbix is distributed under the GNU Affero General Public License (AGPL) version 3. The source code is publicly available, which allows the community to contribute to its development. Extensive documentation and an active forum provide a high level of support[12][13].

2.5 LibreNMS

LibreNMS is an open-source network monitoring system designed for the automatic detection and monitoring of network devices and services. It was created as a fork of the Observium project and has been actively developed since 2013 with an emphasis on community development and flexibility [16][17].

2.5.1 Key features of LibreNMS

- **Automatic device discovery:** LibreNMS supports protocols such as CDP, FDP, LLDP, OSPF, BGP, SNMP and ARP for device detection, which reduces manual work when deploying and expanding the monitoring system [17].
- **Flexible alerting system:** It provides a customizable notification mechanism via e-mail, IRC, Slack and other communication channels. This ensures that administrators are immediately informed of problems [18].
- **API:** LibreNMS provides an API that enables automation, integration with other tools and access to monitoring data [16].
- **Billing system:** The tool offers the option of generating bandwidth bills based on usage, which is useful for internet service providers [17].
- **Automatic updates:** LibreNMS is updated regularly, which ensures access to new features and security vulnerability fixes without the need for manual intervention [18].

2.5.2 LibreNMS architecture

LibreNMS consists of the following components:

1. **Main server:** Processes the data collected from devices and ensures that it is stored in the database [16].
2. **Web interface:** Provides access to data visualization, configuration and performance analysis [17].
3. **Database:** Stores configuration data and collected metrics. LibreNMS is compatible with MySQL and MariaDB databases [18].

2.5.3 Differences compared to Zabbix

- LibreNMS has native support for bandwidth billing, which Zabbix does not have as a native feature [17].
- LibreNMS is more focused on network monitoring, whereas Zabbix covers a wider scope, including applications and servers [16].

2.5.4 Licensing and community

LibreNMS is available under the GPLv3 licence and its development is driven by the community. However, the community is not as active as that of Zabbix.

2.6 Prometheus

Prometheus is an open-source monitoring and alerting tool specifically designed for modern cloud and containerized environments. Its architecture and features distinguish it from traditional solutions, with a focus on flexibility, high performance and easy integration with other systems [19][20].

2.6.1 Key features of Prometheus

- **Pull-based model:** Prometheus collects metrics via HTTP endpoints. This model eliminates the need for agents and increases control over the data. Monitored services must be able to provide metrics in a format that Prometheus can process [20].
- **Multidimensional data:** Metrics are stored in the form of time series identified by unique labels, which enables advanced data analysis and a high level of customization [21].
- **PromQL query language:** Prometheus uses its own query language, PromQL, which enables powerful analyses, aggregations and calculations directly on time series. This language is the basis for creating alerts and visualizations [22].
- **Autonomy:** Prometheus operates as a standalone unit without dependence on external systems, which increases its reliability and resilience to outages [23].
- **Alertmanager:** Part of Prometheus is Alertmanager, which processes alerts and enables their distribution via e-mail, Slack, PagerDuty and other platforms. [24]

2.6.2 Prometheus architecture

Main components of Prometheus:

1. **Prometheus Server:** This component is responsible for collecting metrics, storing time series and processing PromQL queries [20].
2. **Exporter**: Exporters are special modules that convert metrics from various systems and applications into a format readable by Prometheus. Examples include Node Exporter (OS monitoring) and Blackbox Exporter (availability testing) [21].
3. **Alertmanager**: Manages all generated alerts and offers options such as grouping and silencing notifications [24].
4. **Pushgateway**: Enables short-lived applications that do not have their own endpoints to send metrics to Prometheus [23].
5. **Web interface**: Provides basic data querying and visualization capabilities, although most users choose Grafana for this purpose [19].

2.6.3 Specific functions of Prometheus

- **Optimization for containers and microservices**: Prometheus is ideal for dynamic environments. Its integration with Kubernetes enables automatic service discovery and monitoring of service status [22].
- **Data history and retention**: Data is stored locally on disk, which enables high query speed. For long-term retention, tools such as Thanos or Cortex can be integrated [23].
- **Flexible alerting**: The combination of PromQL and Alertmanager makes it possible to create complex rules for generating alerts based on time series [24].

2.6.4 Integration with other systems

- **Grafana**: Prometheus is fully compatible with Grafana, which is used for data visualization and the creation of advanced dashboards. [19].
- **Kubernetes**: Prometheus supports native integration with Kubernetes, including automatic discovery of pods, services and other objects. This feature is key to monitoring modern cloud environments [22].
- **Thanos**: Thanos extends the capabilities of Prometheus with long-term data retention, horizontal scaling and data availability in distributed systems [23].
- **CI/CD tools**: Prometheus can be integrated into a CI/CD pipeline for monitoring and alerting during application deployment, thereby ensuring greater stability of production environments [24].

2.6.5 Licensing and community

Prometheus is distributed under the Apache 2.0 licence, which provides great flexibility in its deployment and modification. An active community and membership of the CNCF (Cloud Native Computing Foundation) ensure its continuous development [19].

3 BASIC INSTALLATION AND CONFIGURATION OF THE TOOLS

Docker Compose was used to install the individual tools, which makes it possible to run all the necessary containers simultaneously.

3.1 Zabbix

The Zabbix Server container contains the main logic of Zabbix. Environment variables define the database connection („DB_SERVER_HOST“) and the login credentials („MYSQL_DATABASE“, „MYSQL_USER“, „MYSQL_PASSWORD“). The „depends_on“ component ensures that Zabbix Server starts only after MySQL Server has been launched.

The Zabbix Web Interface container provides a web interface for managing and configuring Zabbix. The environment variables share the same database settings as Zabbix Server. Port 8080 is mapped to the host system, which enables access to the interface via a browser.

MySQL Server is the database container that stores all Zabbix data, including the configuration and historical metrics. The volume („mysql-data“) ensures that the data persists even after the container is restarted or removed.

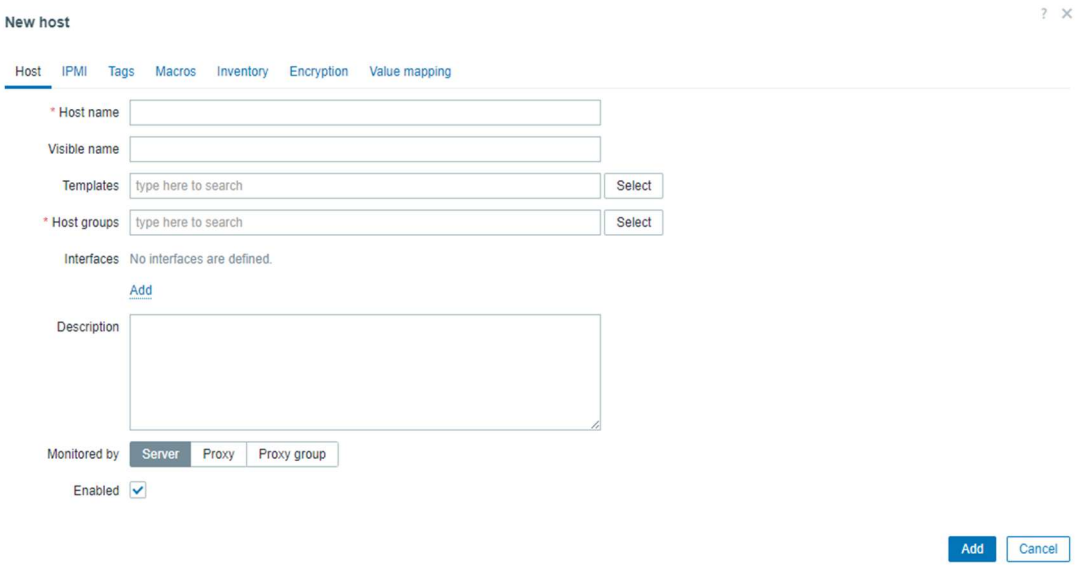


Figure 1: Adding a device for monitoring using Zabbix

```yaml
default_version: '3.8'
services:
  # Zabbix Server
  zabbix-server:
    image: zabbix/zabbix-server-mysql:latest
    environment:
      DB_SERVER_HOST: mysql-server
      MYSQL_DATABASE: zabbix
      MYSQL_USER: zabbix
      MYSQL_PASSWORD: zabbix_pass
      MYSQL_ROOT_PASSWORD: root_pass
    depends_on:
      - mysql-server

  # Zabbix Web Interface
  zabbix-web:
    image: zabbix/zabbix-web-nginx-mysql:latest
    environment:
      DB_SERVER_HOST: mysql-server
      MYSQL_DATABASE: zabbix
      MYSQL_USER: zabbix
      MYSQL_PASSWORD: zabbix_pass
      MYSQL_ROOT_PASSWORD: root_pass
    ports:
      - "8080:8080"
    depends_on:
      - zabbix-server

  # MySQL Server
  mysql-server:
    image: mysql:latest
    environment:
      MYSQL_DATABASE: zabbix
      MYSQL_USER: zabbix
      MYSQL_PASSWORD: zabbix_pass
      MYSQL_ROOT_PASSWORD: root_pass
    volumes:
      - mysql-data:/var/lib/mysql
volumes:
  mysql-data:
```

Figure 2: Example of the zabbix .yml file

3.1.1 Notification settings

It is necessary to go to the „Administration -> Media types“ section and add the required communication channel, for example e-mail. After adding the media, it is necessary to configure the specific settings, such as login credentials or the webhook URL.

Next, it is necessary to go to the „Configuration -> Actions“ section, where a new action for sending notifications is created. This action must contain conditions that determine when a notification is sent (for example, when a specific problem occurs or a certain metric threshold is exceeded).

The action also contains steps specifying to which user or user group the notifications will be sent and via which communication channel. In this way, the notification system can be precisely tailored to the requirements of the monitored infrastructure.

3.2 LibreNMS

```yaml
version: "3.8"
services:
  db:
    image: mariadb:10.5
    container_name: librenms-db
    environment:
      MYSQL_ROOT_PASSWORD: root_password
      MYSQL_DATABASE: librenms
      MYSQL_USER: librenms
      MYSQL_PASSWORD: librenms_password
    volumes:
      - db-data:/var/lib/mysql
    restart: unless-stopped

  librenms:
    image: librenms/librenms:latest
    container_name: librenms-app
    depends_on:
      - db
    environment:
      DB_HOST: db
      DB_NAME: librenms
      DB_USER: librenms
      DB_PASSWORD: librenms_password
      TZ: UTC
      BASE_URL: http://localhost
    ports:
      - "8000:8000"
    volumes:
      - librenms-data:/data
    restart: unless-stopped

  memcached:
    image: memcached:latest
    container_name: librenms-memcached
    restart: unless-stopped

  redis:
    image: redis:latest
    container_name: librenms-redis
    restart: unless-stopped

volumes:
  db-data:
  librenms-data:
```

Figure 3: Example of the librenms .yml file

The configuration contains a database container, an application container and supporting services such as Memcached and Redis.

The database container uses the mariadb:10.5 image. The environment variables MYSQL_ROOT_PASSWORD, MYSQL_DATABASE, MYSQL_USER and MYSQL_PASSWORD set the password for root access, the database name and the login credentials for LibreNMS. The database storage is persistent and mapped to db-data:/var/lib/mysql.

The application container uses the librenms/librenms:latest image and depends on the database container. The variables DB_HOST, DB_NAME, DB_USER and DB_PASSWORD define the database connection. The TZ variable

sets the time zone and BASE_URL defines the address for accessing the application. The application data is stored in librenms-data:/data, while port 8000 is mapped to the host system.

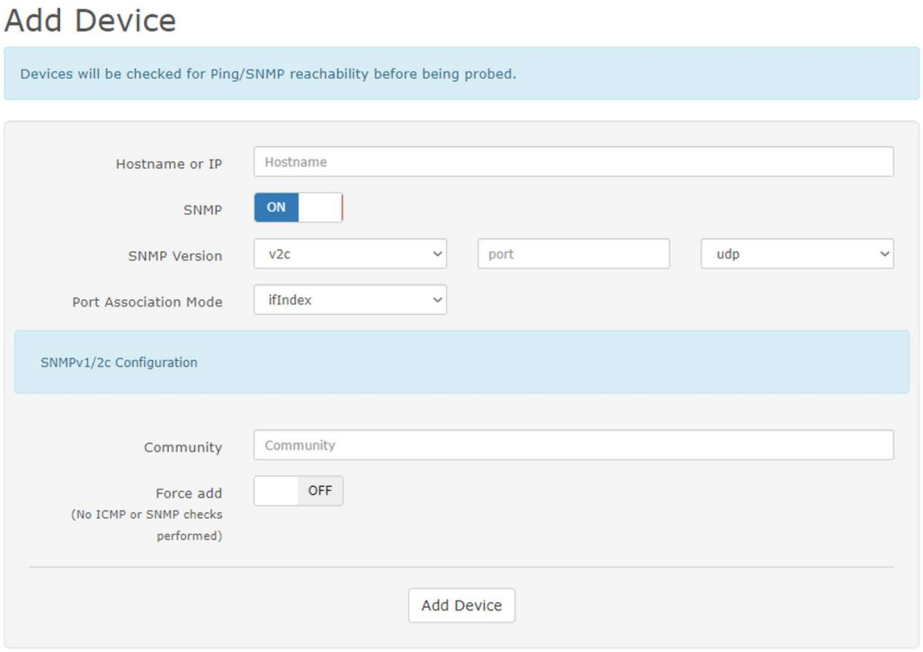


Figure 4: Adding a device for monitoring using LibreNMS

3.3 Prometheus

This section describes the setup of Prometheus and Grafana using Docker Compose. The configuration includes the main components needed for monitoring, including the Prometheus server, SNMP Exporter for collecting SNMP metrics and Grafana for data visualization.

```yaml
version: "3.8"
services:
  prometheus:
    image: prom/prometheus:latest
    container_name: prometheus
    ports:
      - "9090:9090"
    volumes:
      - prometheus-config:/etc/prometheus
    command:
      - --config.file=/etc/prometheus/prometheus.yml
    restart: unless-stopped

  snmp-exporter:
    image: prom/snmp-exporter:latest
    container_name: snmp-exporter
    ports:
      - "9116:9116"
    volumes:
      - snmp-exporter-config:/etc/snmp_exporter
    restart: unless-stopped

  grafana:
    image: grafana/grafana:latest
    container_name: grafana
    ports:
      - "3000:3000"
    environment:
      GF_SECURITY_ADMIN_USER: admin
      GF_SECURITY_ADMIN_PASSWORD: admin
    volumes:
      - grafana-data:/var/lib/grafana
    restart: unless-stopped

volumes:
  prometheus-config:
  snmp-exporter-config:
  grafana-data:
```

Figure 5: Example of the prometheus, snmp exporter and grafana configuration

This configuration includes three main components: Prometheus, SNMP Exporter and Grafana, which work together to collect, process and visualize data.

The Prometheus container uses the prom/prometheus:latest image and is named prometheus. This container runs on port 9090, which is mapped to the host system. The storage for configuration files is mapped to prometheus-config:/etc/prometheus. The unless-stopped restart policy guarantees that the container will be started automatically if it fails.

SNMP Exporter is another important component of this configuration. It uses the prom/snmp-exporter:lates image. This container is responsible for collecting data from devices via the SNMP protocol and runs on port 9116.

The Grafana container runs on port 3000 and provides a user interface for creating and managing dashboards. The administrator login credentials are defined by the variables GF_SECURITY_ADMIN_USER and GF_SECURITY_ADMIN_PASSWORD. Like the other containers, this one also uses the unless-stopped restart policy.

The shared volumes in this configuration ensure persistent storage of data and configurations. The Prometheus configuration files are stored in prometheus-co

nfig, the SNMP Exporter settings in snmp-exporter-config and the Grafana data in grafana-data.

Prometheus is available at http://<IP_adresa>:9090, SNMP Exporter on port 9116 and Grafana at http://<IP_adresa>:3000. To manage monitoring, it is necessary to add the IP addresses or hostnames of the devices to the prometheus.yml configuration file, which allows SNMP Exporter to collect data from these devices. Once correctly set up, the collected data can be displayed in Grafana, where a clear and customizable dashboard for infrastructure monitoring can be created.

```yaml
# prometheus.yml
global:
  scrape_interval: 15s  # Interval zberu dát

scrape_configs:
  - job_name: "prometheus"
    static_configs:
      - targets: ["localhost:9090"]

  - job_name: "snmp_router"
    static_configs:
      - targets: ["158.193.154.59"]  # IP adresa SNMP zariadenia
    metrics_path: /snmp
    params:
      module: [if_mib]  # SNMP modul pre rozhrania
    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target
      - source_labels: [__param_target]
        target_label: instance
      - target_label: __address__
        replacement: snmp-exporter:9116  # Endpoint SNMP Exportera

# snmp.yml
modules:
  if_mib:
    walk:
      - 1.3.6.1.2.1.1  # System group (uptime, názov systému)
      - 1.3.6.1.2.1.2  # Interfaces (stav rozhraní, traffic)
    version: 2
    auth:
      community: public
```

Figure 6: Example of prometheus .yml and snmp.yml

The prometheus.yml configuration file contains the basic settings for data collection by Prometheus. The scrape_interval parameter is set to 15 seconds, which determines the interval between individual data collection cycles. The scrape_configs section contains the definition of the monitoring targets. The first target, labelled prometheus, monitors the Prometheus server itself at localhost:9090. The second target, named snmp_router, is aimed at monitoring the device with the IP address 158.193.154.59 via SNMP. Data collection is specified using the /snmp path and uses the parameters of the if_mib module, which is designed for monitoring network interfaces.

In the params section, the SNMP module if_mib is defined, which includes monitoring of system information and interface states. The relabel_configs section allows metric names to be customized and remapped to individual Prometheus metrics. The SNMP Exporter target is defined at snmp-exporter:9116, which enables integration between Prometheus and SNMP Exporter.

The snmp.yml configuration file contains the settings for SNMP Exporter. The if_mib module contains a list of OIDs that determine which data will be

collected. OID 1.3.6.1.2.1.1 is used to monitor system information such as uptime and system name, while OID 1.3.6.1.2.1.2 monitors the status of network interfaces and the data transferred. The SNMP version is set to version 2 and authentication uses the public community.

4 COMPARATIVE ANALYSIS

To begin with, it was necessary to create a list of criteria for comparing the examined tools with one another. Each of the criteria has an assigned weight on a scale of 1-10, which indicates the importance of the particular criterion.

In the analysis, each tool is assigned a number of points in the range 1-10 for a given criterion. The number of points is then multiplied by the weight of the given criterion, which ensures that more important criteria have a greater influence on the final result. The points for all criteria are then summed, which gives the resulting number of points. For better clarity of the results, the final points for all tools are multiplied by x0.01.

Criterion	Weight	Zabbix	LibreNMS	Prometheus
Hardware requirements	8	9	8	7
Performance	6	8	7	9
Security features	5	6	7	8
Support for SNMP protocol versions	7	10	10	10
Automation and scripting	5	8	8	9
Monitoring configuration options	9	8	7	9
Notifications	9	8	7	9
History of monitored data and data storage	6	7	6	8
Scalability	7	8	7	9
API support	6	8	7	9
Integration with other tools	7	7	6	9
Redundancy support and high availability	5	6	5	7
Visualization and reporting	9	8	7	9
User interface	7	7	8	8
Compatibility	8	9	7	8
Support and community	8	9	7	9

Ease of configuration	7	8	8	8
Costs	7	6	7	8
Ease of installation	4	7	7	9
System upgrades	7	8	9	9
Licence	10	10	10	10
Documentation	7	8	7	9
Currency of the solution	10	8	7	9
AAA against LDAP/AD	5	5	4	6
Installation in VM/Kubernetes	2	5	4	6
Total		15,28	14,16	16,88

4.1.1 Scoring

- Hardware requirements: Zabbix achieved a high score in this category because its hardware requirements are optimized for deployment in larger IT infrastructures, while offering robustness and flexibility even on smaller systems. LibreNMS and Prometheus have higher demands when scaling in large environments.
- Performance: Prometheus processes large volumes of data with low latency, which makes it well suited to modern environments. Zabbix and LibreNMS are less efficient in intensive monitoring.
- Security features: Prometheus excels with its TLS support and flexible access management. Zabbix and LibreNMS offer basic security options.
- Support for SNMP protocol versions: All three tools support SNMPv1, SNMPv2c and SNMPv3, enabling encrypted communication and authentication.
- Automation and scripting: Prometheus excels in this area thanks to its support for modern tools such as Grafana and easy integration with automation tools. Zabbix and LibreNMS provide basic scripting capabilities but are not as extensible.
- Monitoring configuration options: Prometheus and Zabbix allow extensive customization of monitoring, whereas LibreNMS has more limited configuration options.
- Notifications: All three tools offer good options for configuring notifications, but Prometheus is slightly better thanks to its flexible notification rules and integrations with other tools.

- History of monitored data and data storage: Prometheus stores time-series data efficiently. Zabbix and LibreNMS retain historical data, but with higher resource requirements.
- Scalability: Prometheus is optimized for easy scaling in modern environments. Zabbix and LibreNMS require more complex configuration.
- API support: Prometheus offers a modern API with easy integration. Zabbix and LibreNMS have less advanced but still functional APIs.
- Integration with other tools: Prometheus supports a wide range of integrations, especially with tools such as Grafana. Zabbix and LibreNMS are less flexible.
- Redundancy support and high availability: Prometheus offers basic redundancy support, but both Zabbix and LibreNMS lag behind owing to the more complex implementation of these features.
- Visualization and reporting: Prometheus with Grafana excels in visualizations. Zabbix offers high-quality reports, while LibreNMS lags slightly behind.
- User interface: LibreNMS has a simple and intuitive interface, which gives it an advantage over Zabbix and Prometheus, which can be more demanding to use.
- Compatibility: Zabbix is compatible with Linux, Windows and Unix, which makes it ideal for various environments. Prometheus is optimized for containerized and cloud technologies such as Kubernetes and Docker. LibreNMS focuses primarily on Linux, with limited support for other operating systems.
- Support and community: Zabbix and Prometheus have a strong community and extensive documentation. LibreNMS provides smaller but useful support.
- Ease of configuration: Prometheus allows simple customization via YAML configuration files, which makes it easier to set up metrics and alerts. LibreNMS has an intuitive user interface that facilitates the configuration of basic functions. Zabbix offers more robust configuration options but requires more manual steps and more complex management.
- Costs: LibreNMS is a free tool and has lower deployment and operating costs. Prometheus and Zabbix have higher maintenance costs in larger environments.
- Ease of installation: Both LibreNMS and Zabbix offer fast and simple deployment, mainly via containers. Prometheus is optimized for containerized environments but requires basic knowledge.
- System upgrades: Prometheus and LibreNMS support simple and often automated upgrades. Zabbix offers reliable updates every 6 months but requires more manual work in complex environments.
- Licence: All three tools have a full open-source licence.
- Documentation: Prometheus offers the most up-to-date and most extensive documentation, which makes it easier to use. Zabbix and LibreNMS also provide high-quality resources, but less detailed.

- Currency of the solution: Prometheus is the most up to date thanks to modern development and frequent updates. Zabbix and LibreNMS are stable but less innovative.
- AAA against LDAP/AD: Prometheus offers basic support through integration with LDAP for access verification, which is simple to configure but less extensive for advanced scenarios. Zabbix allows integration with LDAP and Active Directory with more detailed configuration of access rights; however, the configuration may be more complex. LibreNMS supports LDAP/AD for basic authentication functions.
- Installation in VM/Kubernetes: Prometheus is designed for simple deployment in modern containerized environments such as Kubernetes and Docker. Zabbix supports deployment in VMs and containers but requires more manual steps and configuration. LibreNMS can also be deployed in VMs and containers; however, its support is limited and less well documented.

CONCLUSION

Based on the analysis and the point-based evaluation, a combination of the Zabbix and Prometheus tools can be recommended. This choice is supported by a detailed analysis of the tools' properties and their ability to cover a broad spectrum of monitoring needs.

Zabbix excels in environments where detailed control over traditional network devices is required. Support for protocols such as SNMP, IPMI and JMX, robust visualizations and alerts enable rapid detection of problems. Thanks to its strong community and documentation it is easy to implement, and its support for redundancy and scalability guarantees stability when the infrastructure is expanded.

Prometheus is ideal for dynamic and cloud environments. Its ability to process large numbers of metrics with low latency, integration with Grafana for advanced visualizations and a flexible API make it a modern tool for rapid adaptation to changes in the infrastructure. For a department planning to deploy cloud technologies, Prometheus is indispensable.

Zabbix will be optimal for monitoring traditional devices and applications, while Prometheus will be used to monitor the dynamic and cloud elements of the infrastructure. This combination will provide a comprehensive monitoring solution covering all the needs of the department, from stability and reliability to flexibility and scalability. At the same time, the proposal ensures that the department will be prepared for future expansion of the infrastructure and the integration of further technologies.

SOURCES

1. "The Comprehensive Guide to SNMP" – LogicMonitor. Available at: https://www.logicmonitor.com/resource/the-comprehensive-guide-to-snmp
2. Dirk Schrader - " Understanding the 4 Types of Network Monitoring Tools and Comparing Available Solutions. Available at: https://blog.netwrix.com/2023/12/27/network-monitoring-tools/
3. "NOC Tools and Software" – INOC. Available at: https://www.inoc.com/blog/noc-tools-and-software
4. "Important Roles in the NOC" – Extnoc. Available at: https://www.extnoc.com/network-operations-center/what-are-the-important-roles-in-the-noc/
5. "Enterprise Network Management Guide" – INOC. Available at: https://www.inoc.com/enterprise-network-management-guide
6. "Network Operations Center Overview" – Extnoc. Available at: https://www.extnoc.com/network-operations-center/
7. "Best Practices for Effective NOC Operations" – BMC. Available at: https://www.bmc.com/blogs/noc-network-operations-center/
8. " What is network monitoring?. – IBM. Available at: https://www.ibm.com/think/topics/network-monitoring
9. Grafana and Prometheus Integration" – Grafana Labs. Available at: https://grafana.com/docs/grafana/latest/getting-started/prometheus/
10. "Zabbix Architecture and Key Features" – Hawatel. Available at: https://hawatel.com/en/blog/introduction-to-zabbix-architecture-and-key-features/
11. "Zabbix Overview" – Initmax. Available at: https://www.initmax.cz/wp-content/uploads/2022/03/zabbix-overview-en.pdf
12. "Zabbix Documentation" – Zabbix.com. Available at: https://www.zabbix.com/documentation
13. "Comprehensive Guide to Zabbix" – Zabbix Solutions. Available at: https://www.zabbix.com/solutions
14. "Zabbix Features and Advantages" – Tech Resources. Available at: https://www.techresources.com/zabbix-features
15. "LibreNMS Overview" – LibreNMS Documentation. Available at: https://docs.librenms.org/
16. "Getting Started with LibreNMS" – LibreNMS Community. Available at: https://community.librenms.org/
17. "LibreNMS Features and Comparison" – Network Monitoring Tools. Available at: https://www.networkmonitoringtools.org/librenms
18. "Introduction to Prometheus" – Prometheus Documentation. Available at: https://prometheus.io/docs/introduction/overview/
19. "PromQL Guide" – Prometheus Documentation. Available at: https://prometheus.io/docs/prometheus/latest/querying/basics/
20. "Monitoring with Prometheus and Grafana" – Grafana Labs. Available at: https://grafana.com/docs/grafana/latest/getting-started/prometheus/
21. "Kubernetes Integration" – Prometheus Documentation. Available at: https://prometheus.io/docs/prometheus/latest/getting_started/
22. "Thanos and Long-term Storage" – Thanos Documentation. Available at: https://thanos.io/

23. "Alertmanager" – Prometheus Documentation. Available at: https://prometheus.io/docs/alerting/latest/alertmanager/