

ДОМАШНЕЕ ЗАДАНИЕ №5

"МОНИТОРИНГ. НАСТРОЙКА МОНИТОРИНГА С НУЛЯ"

1. Цели и задачи

Целью данной работы является сделать 2 дашборда, Windows Exporter и Spring Boot Statistics через любую связку (Exporter, Prometheus Server, Grafana или Telegraf, InfluxDB, Grafana).

Задачи:

1. Настроить мониторинг и визуализировать метрики CPU, Ram, net, Disk с дашборда Windows Exporter

2. Настроить мониторинг и визуализировать метрики с приложения Thread, Память, GC с дашборда Actuator Spring Boot

2. Инструменты

Для выполнения задачи 1 была использована связка:

Exporter → Prometheus Server → Grafana

Для выполнения задачи 2 была использована связка:

Actuator Exporter → Prometheus Server → Grafana

3. Настройка мониторинга

Для сбора с операционной системы windows метрик, таких как: CPU, Ram, net и Disk, поднят windows exporter версии 0.23.1-386 путем запуска файла windows_exporter-0.23.1-386.msi.

Для проверки успешного запуска данной программы был вызван «диспетчер задач», и во вкладке «службы», как видно из рисунка 1, данный exporter запущен.

Так же при переходе в браузере по адресу <http://localhost:9182/> открывается GUI данного экспортера, представленный на рисунке 2.

Тут можно посмотреть метрики, который собирает данный экспортер, проверить статус и информацию о версии экспортера.

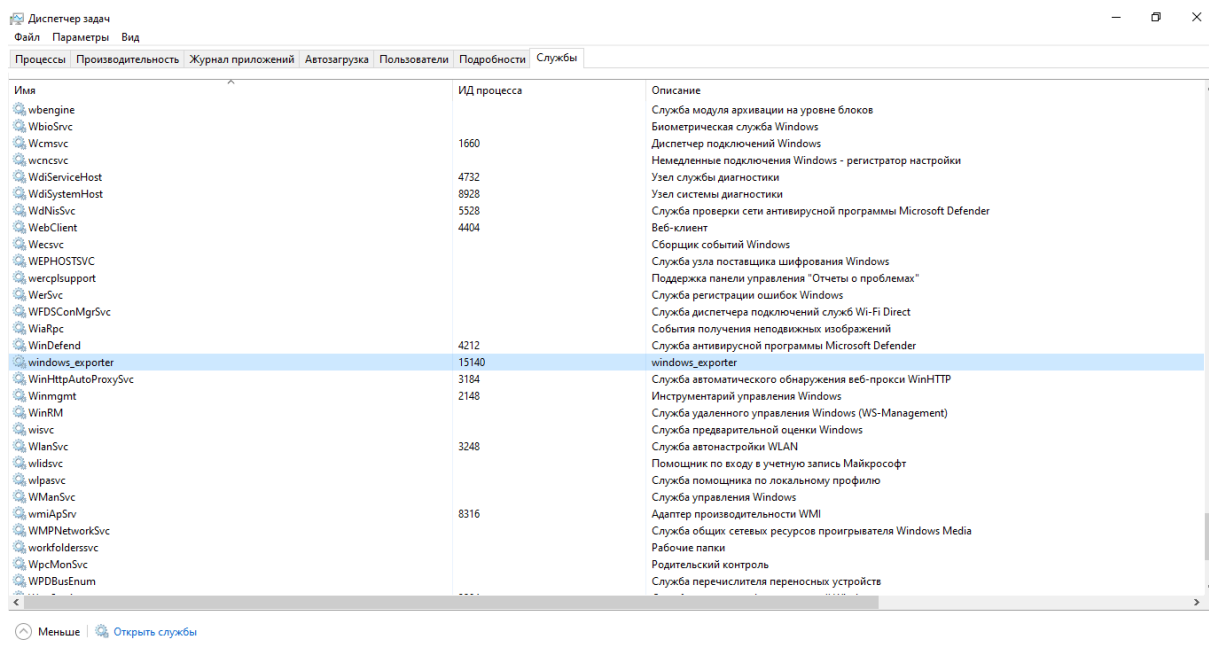


Рисунок 1 – Службы Windows



Prometheus Exporter for Windows servers

Version: (version=0.23.1, branch=heads/tags/v0.23.1, revision=05091643c64e43d84be0141c27f6db3694653668)

- [Metrics](#)
- [Health Check](#)
- [Version Info](#)

Рисунок 2 – GUI windows exporter

На следующем этапе было поднято приложение «demo-0.0.1-SNAPSHOT» со встроенным Actuator Exporter, для сбора метрик из данного приложения.

Запуск приложения производился через командную строку путем ввода команды:

«путь к программе» Java>java -jar «название приложения».jar

Результат запуска приложения через командную строку представлен на рисунке 3.

```
C:\Windows\System32\cmd.exe - java -jar demo-0.0.1-SNAPSHOT.jar
Microsoft Windows [Version 10.0.19045.4894]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

D:\Файлы Саша\Программирование\Нагрузка\тест\Программы\Мониторинг\Demo приложение Java>java -jar demo-0.0.1-SNAPSHOT.jar

  ____  _
 / ___|| | | |
| |___| |_| |
 \___ \|  _/
      |_|_|

:: Spring Boot ::
(v3.1.3)

2024-10-01T17:29:17.666+02:00 INFO 9776 --- [main] com.example.demo.DemoApplication : Starting DemoApplication v0.0.1-SNAPSHOT using Java 22.0.2 with
PID 9776 (D:\Файлы Саша\Программирование\Нагрузка\тест\Программы\Мониторинг\Demo приложение Java>java -jar demo-0.0.1-SNAPSHOT.jar started by Aser_17 in D:\Файлы Саша\Программирование\Нагрузка\тест\Программы\Мониторинг\Demo приложение Java)
2024-10-01T17:29:17.670+02:00 INFO 9776 --- [main] com.example.demo.DemoApplication : No active profile set, falling back to 1 default profile: "def
ault"
2024-10-01T17:29:20.558+02:00 INFO 9776 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8082 (http)
2024-10-01T17:29:20.573+02:00 INFO 9776 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2024-10-01T17:29:20.574+02:00 INFO 9776 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.12]
2024-10-01T17:29:20.705+02:00 INFO 9776 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2024-10-01T17:29:20.706+02:00 INFO 9776 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2928 ms
2024-10-01T17:29:21.948+02:00 INFO 9776 --- [main] o.s.b.a.e.web.EndpointLinksResolver : Exposing 2 endpoint(s) beneath base path '/actuator'
2024-10-01T17:29:22.091+02:00 INFO 9776 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8082 (http) with context path ''
2024-10-01T17:29:22.119+02:00 INFO 9776 --- [main] com.example.demo.DemoApplication : Started DemoApplication in 5.498 seconds (process running for 6.299s)
2024-10-01T17:29:25.768+02:00 INFO 9776 --- [nio-8082-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2024-10-01T17:29:25.771+02:00 INFO 9776 --- [nio-8082-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2024-10-01T17:29:25.775+02:00 INFO 9776 --- [nio-8082-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 3 ms
```

Рисунок 3 – Результат запуска приложения «demo-0.0.1-SNAPSHOT»

Для сбора и хранения метрик с ранее запущенных экспортеров необходимо развернуть базу данных временных рядов – Prometheus Server.

Перед запуском была произведена настройка Prometheus Server через конфигурационный файл prometheus.yml.

Скрипт готового конфигурационного файла prometheus.yml представлен на рисунке 4.

```
prometheus - Блокнот
Файл Правка Формат Вид Справка
# 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: 'mynode'
    static_configs:
      - targets: ['localhost:9182']
  - job_name: 'spring-actuator'
    metrics_path: '/actuator/prometheus'
    scrape_interval: 5s
    static_configs:
      - targets: ['localhost:8082']
```

Рисунок 4 – Скрипт файла prometheus.yml

После настройки Prometheus Server, был произведен его запуск через командную строку путем ввода команды:

```
«путь к программе»>prometheus --config.file=prometheus.yml
```

Результат запуска Prometheus через командную строку показан на рисунке 5.

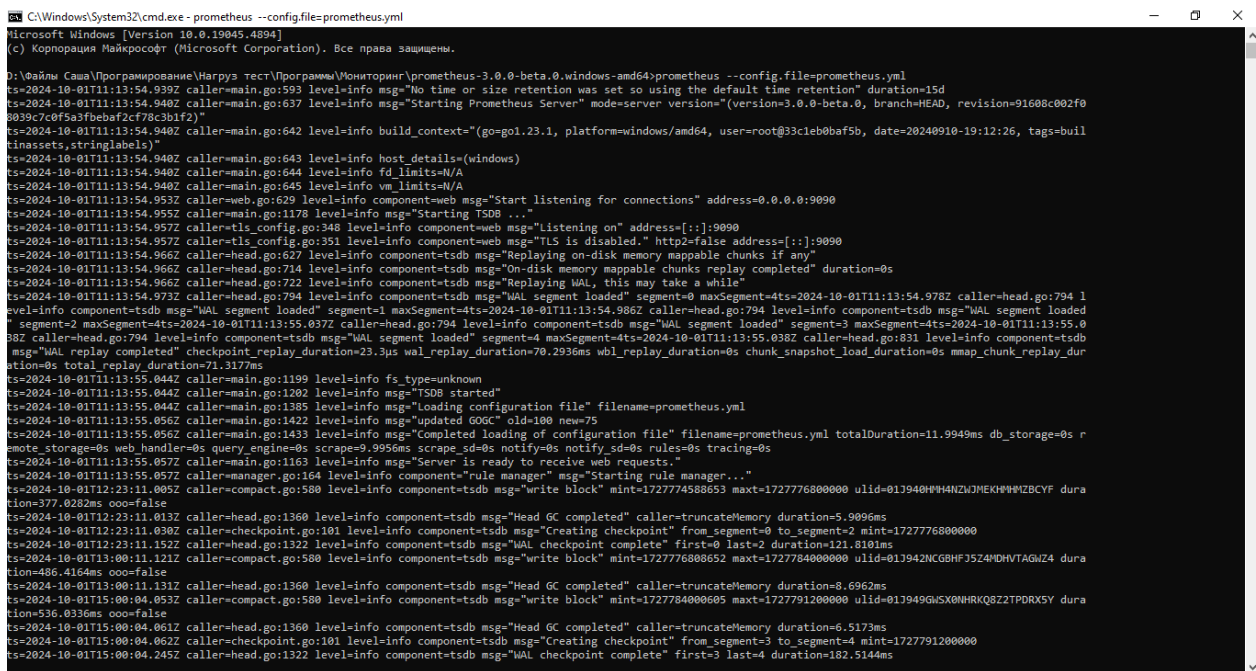


Рисунок 5 – Результат запуска Prometheus Server

Для проверки подключения Prometheus Server к экспортерам отобразим GUI программы Prometheus, для этого в браузере перейдем по адресу:

<http://localhost:9090/>

Во вкладке «Status» выберем «Target health» и проверим состояние экспортеров.

Как видно из рисунка 6, все экспортеры подняты.

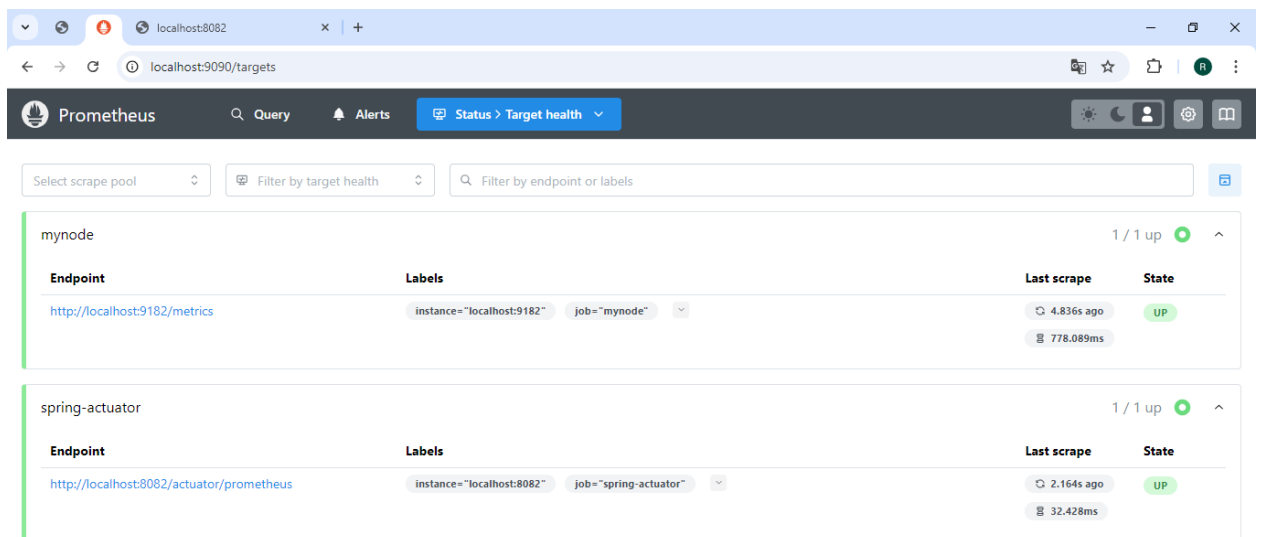


Рисунок 6 – Статус подключенных экспортеров в Prometheus Server

Для визуализации метрик, переданных из экспортеров в Prometheus Server, была установлена открытая платформа мониторинга и анализа данных – Grafana.

Далее откроем GUI приложения Grafana, для этого перейдем в браузере по адресу <http://localhost:3000>.

Во вкладке Configuration добавим Prometheus в качестве источника данных с указанием его адреса <http://localhost:9090>.

Для отображения метрик windows был скачан dashboard в формате JSON с официального сайта Grafana по адресу:

<https://grafana.com/grafana/dashboards/20763-windows-exporter-dashboard-2024/>

Для отображения метрик с приложения «demo-0.0.1-SNAPSHOT» был скачан dashboard в формате JSON с официального сайта Grafana по адресу:

<https://grafana.com/grafana/dashboards/12464-spring-boot-statistics/>

Дашборды для мониторинга операционной системы windows и приложения «demo-0.0.1-SNAPSHOT» были импортированы в Grafana и представлены на рисунках 1 и 2, соответственно.

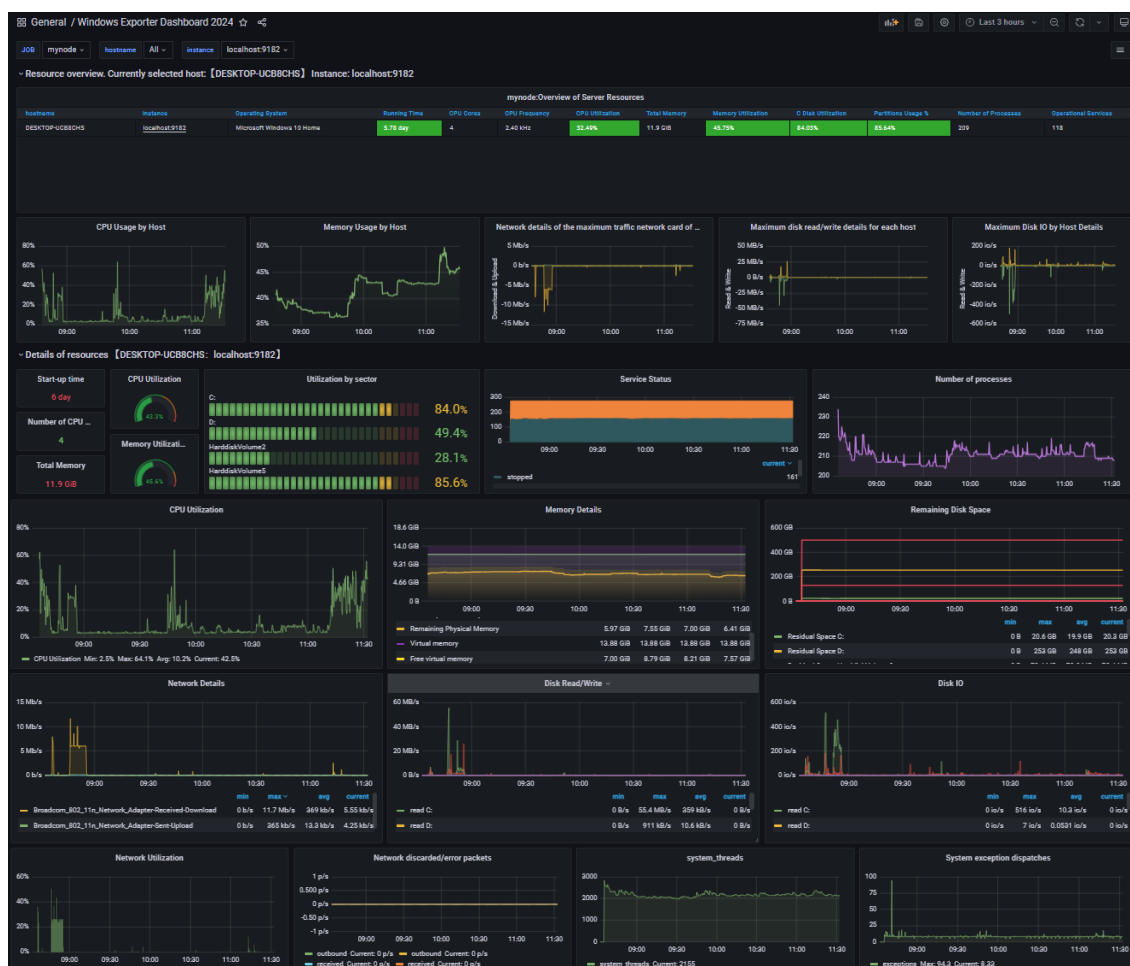


Рисунок 6 – Dashboard Windows Exporter

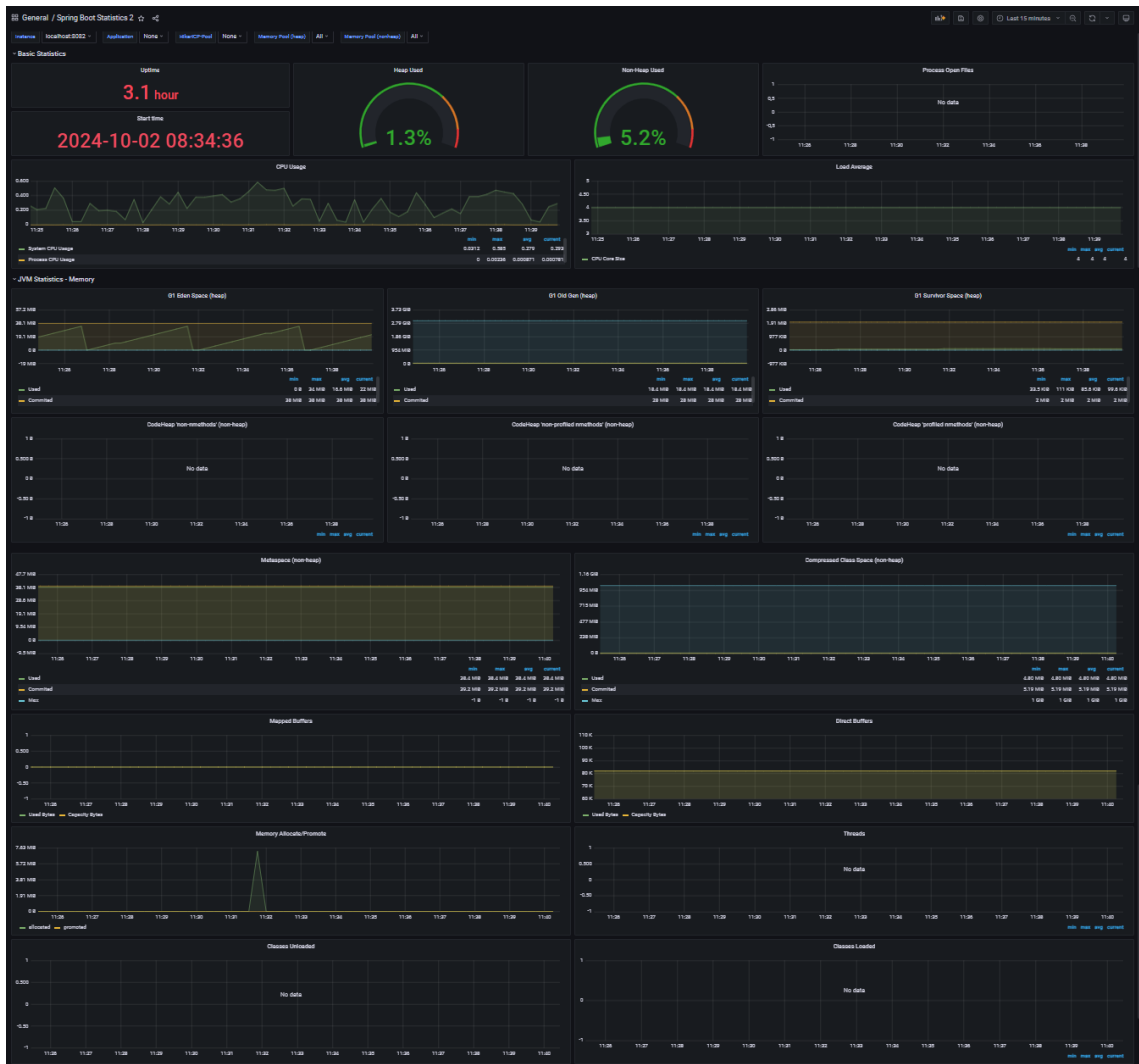


Рисунок 7 – Dashboard Spring Boot Statistics