# Мониторинг: Прост наръчник

## Какво е мониторинг и защо ни трябва?

Мониторингът е като наблюдаване на здравето на нашите компютърни системи. Също както лекарят следи жизнените показатели на пациента, така и ние следим различни показатели на нашите сървъри и приложения. Това ни позволява да разберем кога нещо не е наред, преди да стане голям проблем.

В нашия проект ние създадохме система за мониторинг, защото: - Искахме да знаем кога сървърите са претоварени - Трябваше да следим колко памет използват приложенията ни - Искахме да сме сигурни, че дисковото пространство не се запълва - Трябваше ни начин да виждаме всичко това на едно място

## Какви инструменти използвахме?

Решихме да използваме няколко безплатни инструмента, които работят добре заедно:

### Docker

Docker ни позволява да опаковаме програмите в “контейнери”, които могат да работят навсякъде по един и същи начин. Използвахме Docker, защото: - Лесен е за инсталиране - Не трябва да се притесняваме за различни операционни системи - Можем бързо да стартираме и спираме нашите инструменти

### Prometheus

Prometheus е програма, която събира информация за това как работят нашите системи. Той: - Записва данни като използвана памет, CPU и дисково пространство - Съхранява тези данни във времето, така че можем да видим как се променят - Периодично проверява сървърите и записва техните показатели - Може да ни предупреди, когато нещо не е наред

### Node Exporter

Node Exporter е малка програма, която помага на Prometheus да събира данни. Той: - Работи на всеки сървър, който искаме да наблюдаваме - Събира информация за хардуера и операционната система - Изпраща тази информация на Prometheus, когато той я поиска

### Grafana

Grafana е програма, която показва красиви графики и табла с данните, събрани от Prometheus. Тя: - Позволява ни да правим табла за управление с много графики - Може да показва данните по различни начини (линейни графики, измервателни уреди и др.) - Има цветово кодиране за лесно разпознаване на проблеми (зелено = добре, жълто = внимание, червено = проблем) - Може да се настрои за изпращане на известия, когато има проблем

## Как конфигурирахме всичко?

### Стъпка 1: Създадохме Docker Compose файл

Първо, създадохме файл с име docker-compose.yml, който казва на Docker какви програми искаме да стартираме и как да бъдат свързани помежду си. Този файл изглежда така:

version: '3.8'  
  
services:  
 prometheus:  
 image: prom/prometheus:latest  
 container\_name: prometheus  
 restart: unless-stopped  
 volumes:  
 - ./prometheus/prometheus.yml:/etc/prometheus/prometheus.yml  
 - prometheus\_data:/prometheus  
 ports:  
 - "9090:9090"  
 networks:  
 - monitoring  
  
 node\_exporter:  
 image: prom/node-exporter:latest  
 container\_name: node\_exporter  
 restart: unless-stopped  
 volumes:  
 - /proc:/host/proc:ro  
 - /sys:/host/sys:ro  
 - /:/rootfs:ro  
 command:  
 - '--path.procfs=/host/proc'  
 - '--path.rootfs=/rootfs'  
 - '--path.sysfs=/host/sys'  
 ports:  
 - "9100:9100"  
 networks:  
 - monitoring  
  
 grafana:  
 image: grafana/grafana:latest  
 container\_name: grafana  
 restart: unless-stopped  
 volumes:  
 - grafana\_data:/var/lib/grafana  
 environment:  
 - GF\_SECURITY\_ADMIN\_USER=admin  
 - GF\_SECURITY\_ADMIN\_PASSWORD=admin  
 ports:  
 - "3000:3000"  
 networks:  
 - monitoring  
  
networks:  
 monitoring:  
 driver: bridge  
  
volumes:  
 prometheus\_data:  
 grafana\_data:

Този файл казва на Docker да стартира три програми: - Prometheus за събиране на данни - Node Exporter за предоставяне на данни - Grafana за визуализация на данните

### Стъпка 2: Конфигурирахме Prometheus

След това създадохме файл prometheus.yml, който казва на Prometheus какво да наблюдава:

global:  
 scrape\_interval: 15s  
 evaluation\_interval: 15s  
  
scrape\_configs:  
 - job\_name: 'prometheus'  
 static\_configs:  
 - targets: ['localhost:9090']  
  
 - job\_name: 'node\_exporter'  
 static\_configs:  
 - targets: ['node\_exporter:9100']

Този файл казва на Prometheus: - Да проверява наблюдаваните системи на всеки 15 секунди - Да наблюдава себе си (на порт 9090) - Да наблюдава Node Exporter (на порт 9100)

### Стъпка 3: Стартирахме контейнерите

След като подготвихме файловете, стартирахме всичко с проста команда:

docker-compose up -d

Тази команда казва на Docker да стартира всички програми, описани в docker-compose.yml файла в заден план.

### Стъпка 4: Настроихме Grafana

След като всичко беше стартирано, трябваше да настроим Grafana за визуализация на данните:

1. Отворихме Grafana в браузъра на адрес http://localhost:3000
2. Влязохме с потребителско име admin и парола admin
3. Добавихме Prometheus като източник на данни:
   * Отидохме на “Configuration” > “Data Sources”
   * Избрахме “Add data source”
   * Избрахме “Prometheus”
   * За URL въведохме http://prometheus:9090
   * Натиснахме “Save & Test”

### Стъпка 5: Създадохме табло за управление

Накрая създадохме табло, което показва важната информация:

1. Отидохме на “Create” > “Dashboard”
2. Добавихме нови панели за:
   * Използване на CPU
   * Използване на памет
   * Използване на дисково пространство

За всеки панел трябваше да укажем какви данни да показва:

* За CPU използване: 100 - (avg by (instance) (irate(node\_cpu\_seconds\_total{mode="idle"}[5m])) \* 100)
* За използване на памет: 100 \* (1 - ((node\_memory\_MemFree\_bytes + node\_memory\_Buffers\_bytes + node\_memory\_Cached\_bytes) / node\_memory\_MemTotal\_bytes))
* За дисково пространство: 100 - ((node\_filesystem\_avail\_bytes{mountpoint="/"} \* 100) / node\_filesystem\_size\_bytes{mountpoint="/"})

## Какво измерваме и защо?

### CPU използване

Измерваме колко процента от процесора се използва. Това е важно, защото: - Ако CPU използването е постоянно високо (над 80%), системата може да стане бавна - Ако CPU използването е много ниско за дълго време, може би имаме повече сървъри, отколкото ни трябват - Внезапни скокове в CPU използването могат да сигнализират за проблем

### Използване на памет

Следим колко оперативна памет (RAM) се използва. Това е важно, защото: - Ако паметта е почти пълна, системата започва да използва swap (диска), което я прави много по-бавна - Ако използването на паметта постоянно расте, може да имаме “memory leak” (теч на памет) - Ако използването на паметта е ниско, може би сме заделили повече памет, отколкото е необходимо

### Дисково пространство

Наблюдаваме колко място на диска е използвано. Това е важно, защото: - Ако дискът се напълни, приложенията няма да могат да запазват данни - Ако използването на диска расте бързо, може скоро да имаме проблем - Ако използваме малко дисково пространство, може би сме заделили твърде много

## Как изглежда нашето табло за управление?

Нашето табло за управление показва три типа информация:

1. **Графики за промяна във времето**
   * Линейни графики, показващи как се променят показателите с времето
   * Помагат да видим модели и тенденции
   * Позволяват да разберем кога системата е най-натоварена
2. **Текущо състояние**
   * Измервателни уреди, показващи моментното използване на ресурсите
   * Цветово кодирани (зелено = добре, жълто = внимание, червено = опасно)
   * Дават бърз поглед върху състоянието на системата
3. **Статистика**
   * Показва средни, минимални и максимални стойности
   * Помага да разберем дали текущата ситуация е нормална или необичайна

## Предимства на нашето решение

Нашата система за мониторинг има няколко предимства:

### Лесна за инсталиране

* Работи с проста команда (docker-compose up -d)
* Не изисква сложни конфигурации
* Може да се стартира на почти всеки компютър с Linux, Windows или Mac

### Безплатна

* Всички използвани компоненти са с отворен код
* Няма лицензионни такси
* Може да се използва за колкото време искаме

### Гъвкава

* Лесно можем да добавяме нови сървъри за наблюдение
* Можем да променяме таблото според нуждите ни
* Лесно се интегрира с други системи

### Лека

* Използва минимални ресурси
* Може да работи на малък сървър
* Не забавя наблюдаваните системи

## Как да използваме нашата система?

### Достъп до таблото

1. Отваряме уеб браузър
2. Въвеждаме адрес http://localhost:3000 (или IP адреса на сървъра вместо localhost)
3. Влизаме с потребителско име admin и парола admin (препоръчително е да промените паролата)
4. Отиваме на “Dashboards” и избираме нашето табло

### Интерпретиране на данните

* **Зелени зони** (0-70%): Всичко е наред, системата работи нормално
* **Жълти зони** (70-85%): Внимание, системата е натоварена, но все още работи добре
* **Червени зони** (над 85%): Опасност, системата е претоварена и може да има проблеми

### Какво да правим при проблеми

Ако видим проблем (червена зона), можем да: 1. Проверим кой ресурс е претоварен (CPU, памет или диск) 2. Идентифицираме кои процеси използват много ресурси (с команди като top, ps, или df) 3. Предприемем действия: - Рестартираме проблемни услуги - Освобождаваме дисково пространство - Добавяме повече ресурси, ако е необходимо

## Заключение

Създадохме проста, но ефективна система за мониторинг, използвайки: - Docker за лесно инсталиране - Prometheus за събиране на данни - Node Exporter за предоставяне на системни метрики - Grafana за визуализация

Тази система ни позволява да следим: - Използването на CPU - Използването на памет - Използването на дисково пространство

С нейна помощ можем да открием проблеми, преди да станат сериозни, и да осигурим по-добра работа на нашите системи. Най-хубавото е, че създаването й отне само няколко прости стъпки и използва напълно безплатни инструменти с отворен код.