Логирование, Healthcheck, Auto-Restart и Мониторинг в Docker

**📘 5.1 Логирование в Docker**

Когда приложение что-то пишет в console.log() — Docker это **сохраняет как логи контейнера**.

**📥 Просмотр логов:**

bash

КопироватьРедактировать

docker logs <container\_id>

**📡 Пример:**

bash

КопироватьРедактировать

docker logs my-node-app

**📖 Постоянный вывод:**

bash

КопироватьРедактировать

docker logs -f my-node-app

**🧰 5.2 Как Node.js пишет логи?**

js

КопироватьРедактировать

console.log('Инфо');

console.error('Ошибка');

Все эти сообщения будут доступны через docker logs.

**🧪 5.3 Перенаправление логов в файл**

**Простой способ:**

yaml

КопироватьРедактировать

services:

app:

build: .

logging:

driver: "json-file"

options:

max-size: "10m"

max-file: "3"

**Альтернативные драйверы:**

* json-file (по умолчанию)
* syslog (в систему)
* fluentd (в log-агент)
* gelf (для Graylog)
* awslogs, splunk, journald

**💡 5.4 Пример логирования в файл внутри контейнера**

js

КопироватьРедактировать

const fs = require('fs');

const logStream = fs.createWriteStream('/app/logs/app.log', { flags: 'a' });

console.log = (...args) => {

logStream.write(`[INFO] ${args.join(' ')}\n`);

process.stdout.write(`[INFO] ${args.join(' ')}\n`);

};

console.log('Приложение запущено');

⚠️ Не забудь создать logs/ в проекте и добавить volume, если хочешь сохранять лог-файл за пределами контейнера.

**🛟 5.5 Healthcheck — как Docker знает, что сервис жив?**

**🔍 Пример:**

Dockerfile

КопироватьРедактировать

HEALTHCHECK --interval=10s --timeout=3s --start-period=5s --retries=3 \

CMD curl --fail http://localhost:3000 || exit 1

➡️ Docker будет проверять каждый 10 секунд, доступен ли порт 3000.  
Если не отвечает — контейнер помечается как **unhealthy**.

**Проверить состояние:**

bash

КопироватьРедактировать

docker inspect --format='{{json .State.Health}}' <container\_id>

**🔁 5.6 Политики перезапуска**

Что делать, если контейнер **упал**?

**Варианты:**

| **Политика** | **Что делает** |
| --- | --- |
| no (по умолчанию) | Не перезапускать |
| always | Перезапускается **всегда** |
| unless-stopped | Перезапускается, пока не остановлен вручную |
| on-failure | Перезапускается только при ошибке выхода (код ≠ 0) |

**Пример:**

yaml

КопироватьРедактировать

services:

app:

build: .

restart: always

**🧪 5.7 Инструменты мониторинга контейнеров**

**1️⃣ docker stats — встроенный мониторинг**

bash

КопироватьРедактировать

docker stats

| **Имя контейнера** | **CPU %** | **Память** | **I/O** | **NET** |
| --- | --- | --- | --- | --- |

**2️⃣ Portainer — веб-интерфейс для управления Docker**

bash

КопироватьРедактировать

docker run -d -p 9000:9000 --name portainer \

-v /var/run/docker.sock:/var/run/docker.sock \

-v portainer\_data:/data \

portainer/portainer-ce

Открой в браузере: <http://localhost:9000>

**3️⃣ Prometheus + Grafana**

* Prometheus собирает метрики
* Grafana отображает красивые графики
* Требуют настройки экспортеров

Для Node.js можно использовать prom-client.

**4️⃣ Loki (от Grafana) — сбор логов**

Позволяет собирать все логи контейнеров централизованно и красиво их искать.

**🛡️ 5.8 Безопасность логирования**

* ❌ Не логируй пароли и токены
* ✅ Шифруй логи в продакшне
* ✅ Добавляй уровни логов: info, error, warn, debug

Библиотеки:

* winston
* pino
* bunyan

**📓 5.9 Полный пример Compose с логами и restart**

yaml

КопироватьРедактировать

version: "3.8"

services:

app:

build: .

ports:

- "3000:3000"

restart: always

logging:

driver: json-file

options:

max-size: "5m"

max-file: "3"

healthcheck:

test: ["CMD", "curl", "-f", "http://localhost:3000"]

interval: 10s

timeout: 5s

retries: 3

**✅ 5.10 Вывод главы**

Теперь ты умеешь:

* Сохранять и читать логи контейнеров
* Отслеживать "жив" ли сервис (HEALTHCHECK)
* Автоматически перезапускать контейнеры при сбоях
* Настраивать мониторинг и лог-пайплайн