Тестовое задание на позицию DevOps

Задание

Необходимо развернуть простое веб-приложение, доступное пользователю **через nginx**, запущенные в Docker-контейнерах.

1. Приложение (backend)
Используйте любой вариант:
Вариант А (рекомендуется):
простой HTTP-сервер на Python (http.server)
сервер должен отвечать на / текстом:
"Hello from Effective Mobile!"
Вариант В:
любой минимальный HTTP-сервер (Node.js / другое)

Требования:
приложение слушает порт 8080
приложение **не публикует порт наружу** (доступ только из docker-сети)

2. Nginx (reverse proxy)
Используйте официальный образ nginx, nginx должен:

1. принимать HTTP-запросы на порт 80 проксировать
2. запросы на backend-сервис использовать
3. отдельный конфигурационный файл

Конфигурация nginx должна:

1. содержать upstream или прямой proxy_pass
2. корректно проксировать путь /

3. Docker
Backend

1. отдельный Dockerfile
2. приложение запускается при старте контейнера

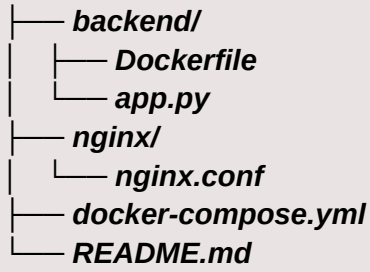
Nginx

1. можно использовать стандартный образ
2. конфигурация подключается через volume или кастомный Dockerfile

4. Docker Compose Создайте docker-compose.yml, который

1. поднимает **два сервиса**:
   backend
   nginx
2. настраивает сетевое взаимодействие между ними
3. пробрасывает **только порт 80 nginx** на хост
4. задаёт понятные имена контейнерам

5. Проверка работоспособности
После запуска должно быть возможно выполнить:
curl http://localhost
Ожидаемый ответ:
"Hello from Effective Mobile!"

6. Git
Публичный Git-репозиторий
Понятная структура проекта


7. README.md
Опишите:
как запустить проект
как проверить результат
кратко — как работает схема (nginx → backend)

Ограничения
Использовать только Docker и Docker Compose
Kubernetes использовать нельзя
Не использовать готовые reverse-proxy решения кроме nginx

Рекомендации по выполнению тестового задания

1. Docker

Рекомендуется:

- использовать **официальные образы** в качестве базовых
- минимизировать размер образа
- явно указывать WORKDIR
- не хранить временные файлы в образе

Будет плюсом:

- запуск приложения **не от root**
- использование EXPOSE для документирования порта
- многостадийная сборка (если уместно)

2. Dockerfile

Ожидается:

- один сервис — один Dockerfile
- приложение запускается через CMD или ENTRYPOINT
- отсутствие “магических” команд в docker-compose.yml

Антипаттерны:

- установка лишних пакетов
- использование latest без необходимости
- запуск нескольких процессов в одном контейнере

3. Docker Compose

Рекомендуется:

- использовать **service names** для сетевого взаимодействия
- публиковать наружу **только nginx**
- не хардкодить IP-адреса
- задавать явные имена сервисов и контейнеров

Будет плюсом:

- отдельная docker-сеть
- использование .env файла

4. Nginx

Ожидается:

- использование proxy_pass к backend по имени сервиса
- корректная базовая конфигурация

Будет плюсом:

- передача стандартных заголовков:
  - Host
  - X-Real-IP
  - X-Forwarded-For
- выключение лишних дефолтных конфигов
- аккуратная структура nginx.conf

5. Networking

Рекомендуется:

- backend-сервис **не должен быть доступен с хоста**
- взаимодействие только внутри docker-сети
- отсутствие проброса лишних портов

6. README.md

Ожидается:

- пошаговая инструкция запуска
- команда для проверки работоспособности
- краткое описание архитектуры

Будет плюсом:

- схема взаимодействия (текстом или ASCII)
- список использованных технологий

7. Git

Рекомендуется:

- осмысленные имена файлов и директорий
- отсутствие лишних файлов в репозитории

- аккуратная структура проекта

Будет плюсом:

- логичные коммиты
- .gitignore

8. Безопасность (базово)

Рекомендуется:

- не хранить секреты в репозитории
- не использовать root без необходимости
- не открывать лишние порты