Скачиваем docker

snap install docker

docker run -d -p 80:80 --name d\_1 nginx



1. Запускаем докер в фоновом режиме (-d), публикуем контейнер на порт хоста (-p 80:80), и присваиваем имя (--name test1) 

http://localhost:80

1. Старт и остановка контейнера происходит следующим образом:

Запуск контейнера:

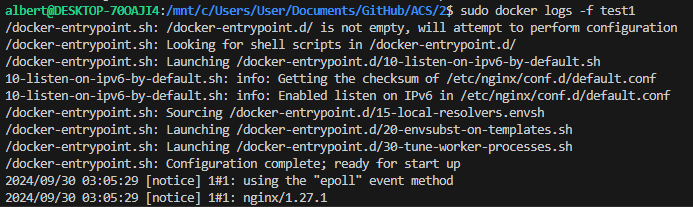
* Когда выполняется команда docker run, Docker Engine взаимодействует с ядром операционной системы для создания изолированного пространства для процесса — контейнера.
* Docker использует cgroups и namespaces для изоляции контейнера (ограничение по ресурсам и изоляция процессов).
* На основе образа nginx создается контейнер
* Когда контейнер стартует, основной процесс nginx начинает слушать запросы по порту 80

http://localhost:80

Остановка контейнера:

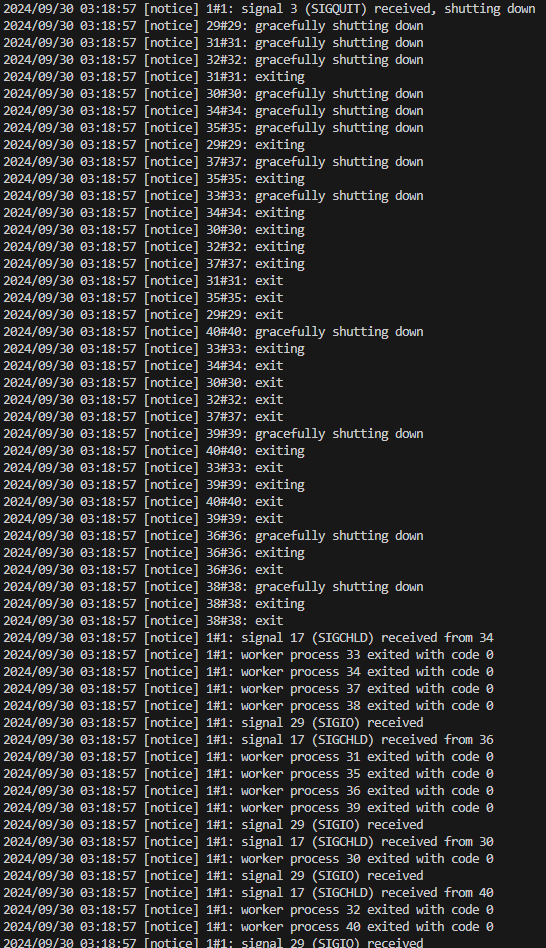
* Команда docker stop отправляет основной процессу внутри контейнера сигнал SIGTERM, давая возможность завершить работу корректно.
* Если процесс не завершится в течение заданного таймаута (по умолчанию 10 секунд), Docker отправит сигнал SIGKILL, чтобы принудительно завершить процесс.

docker logs -f d\_1

1.Подпишемся на логи контейнера, чтобы отслеживать его состояние

2. В другой вкладке терминала остановим docker и посмотрим, как это отразится на логах





После сигнала SIGQUIT, система успешно закрыла все свои процессы и остановилась

**2. Уровни изоляции контейнера**

**1. PID**

С изоляцией:

docker run -it --name pid-isolated ubuntu bash

apt update && apt install procps

exit

docker start pid-isolated

docker exec -it pid-isolated ps aux

Без изоляции:

docker run --name pid-host --pid="host" -d ubuntu

apt update && apt install procps

exit

docker start pid-isolated

docker exec -it pid-host ps aux

2. IPC (Межпроцессная коммуникация)

С изоляцией:

docker run --name ipc-isolated -d nginx

docker exec -it ipc-isolated ipcs

Без изоляции:

docker run --name ipc-host --ipc="host" -d nginx

docker exec -it ipc-host ipcs

3. Network (Сетевая изоляция)

С изоляцией:

docker run --name net-isolated -d nginx

docker exec -it net-isolated ip a

Без изоляции:

docker run --name net-host --network="host" -d nginx

docker exec -it net-host ip a

4. Users (Изоляция пользователей)

С изоляцией:

docker run --name user-isolated -d nginx

docker exec -it user-isolated id

Без изоляции:

docker run --name user-host --userns="host" -d nginx

docker exec -it user-host id

5. Mount (Файловая система)

С изоляцией:

docker run --name mount-isolated -d nginx

docker exec -it mount-isolated ls /

Без изоляции:

docker run --name mount-host -v /:/mnt -d nginx

docker exec -it mount-host ls /mnt

6. UTS (Изоляция имени хоста)

С изоляцией:

docker run --name uts-isolated -d nginx

docker exec -it uts-isolated hostname

Без изоляции:

docker run --name uts-host --uts="host" -d nginx

docker exec -it uts-host hostname

**1. Что такое cgroups?**

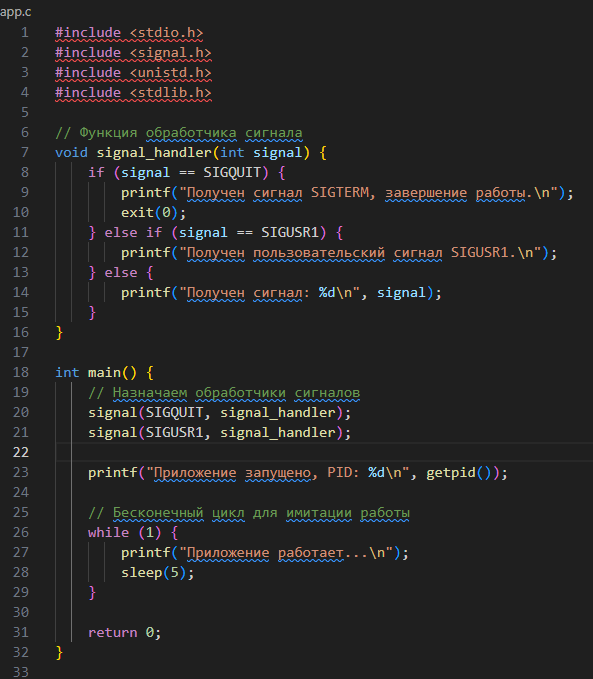
**Control Groups** (cgroups) — это механизм ядра Linux, который позволяет ограничивать, управлять и отслеживать ресурсы (CPU, память, сеть, диск и другие) для групп процессов. Cgroups используются для контроля над ресурсами, которые может использовать процесс, и широко применяются в контейнерах для изоляции и управления ресурсами каждого контейнера.

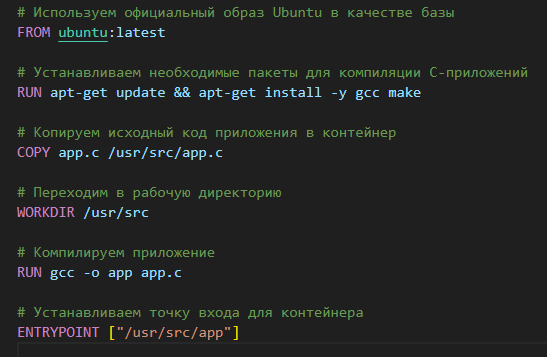
**Основные задачи cgroups:**

1. **Ограничение ресурсов**: Можно установить пределы использования памяти, процессора или сетевых ресурсов для группы процессов.
2. **Приоритеты ресурсов**: Процессам можно присвоить разные приоритеты для использования ресурсов.
3. **Учёт ресурсов**: Отслеживание использования ресурсов (например, сколько памяти или CPU использует процесс).
4. **Изоляция ресурсов**: Обеспечение того, чтобы один процесс не потреблял ресурсы, необходимые другим процессам.
5. **Ограничение количества процессов**: Можно задать ограничение на количество процессов или потоков, которые может создать группа процессов.

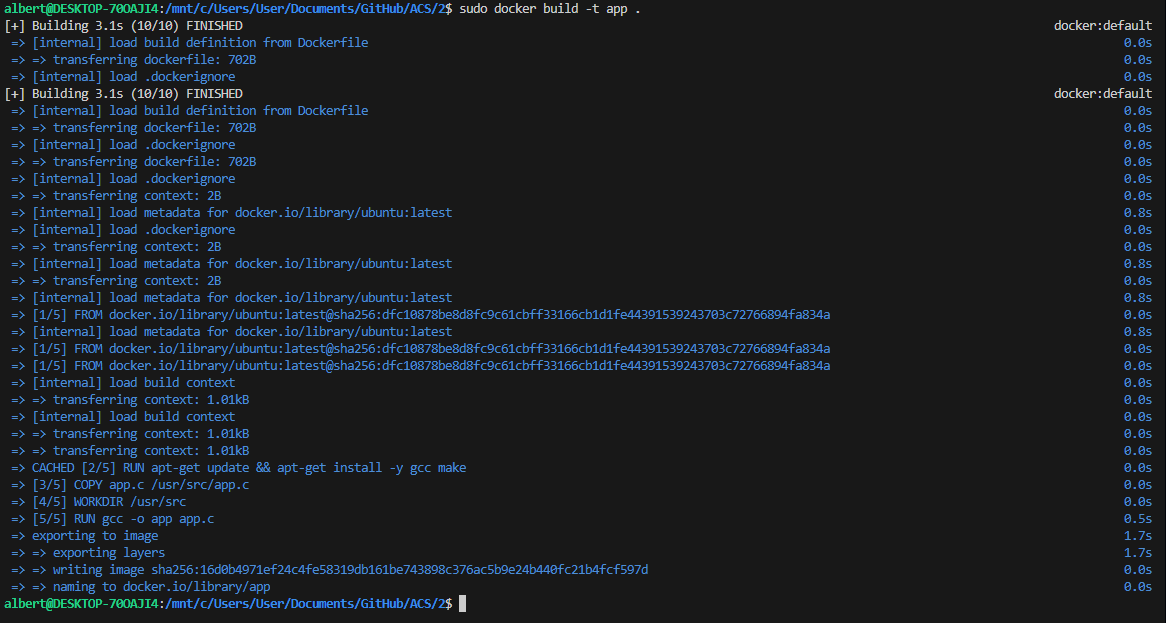
**2. Создание приложения, обрабатывающее сигналы**

Напишем код на C, обрабатывающий сигналы



Напишем DockerFile

Соберем Docker образ



Запустим образ

