📚 Работа с портами и томами.

Работа с портами внутренней сети Docker

Внешняя сеть хоста и внутренняя сеть docker являются независимыми и могут общаться через порты.

Выводит подробную информацию в формате JSON о контейнере. Можно посмотреть какие слушает внутренние порты (порт можно посмотреть и через docker ps) и ip

```
docker container inspect <container_id или container_name>
```

Информация о портах

```
docker port <container_id или container_name>
```

Чтобы связать внутренний порт 80/tcp nginx с нашим внешним портом 8000 необходим флаг -p при запуске

```
docker run --name container_name -d -p 8000:80 nginx
```

Теперь 80/tcp -> 0.0.0.0:8000 при вызове `docker port`

Чтобы задать точный ip адрес

```
docker run --name container_name -d -p 127.0.0.1:8000:80 nginx
```

Параллельно запустить в контейнер терминал bash

```
docker exec -it container_name bash
```

Инструкция EXPOSE в Dockerfile не открывает порт и не пробрасывает его наружу, а декларирует, что контейнер слушает указанный порт во время выполнения.

```
EXPOSE 80
EXPOSE 5000/tcp
EXPOSE 6379/udp
```

Для автоматического назаначения порта используется **-P**

```
docker run --name container_name -d -P nginx
```

Для того, чтобы при запуске процессов в контейнере через `exec` не использовать пользователя **root**, нужно добавить отдельного пользователя в **Dockerfile**

```
FROM python:3.13
RUN groupadd -r group_name && useradd -r -g group_name user_name

...

EXPOSE 4000
USER user_name
CMD ["python", "main.py"]
```

Работа с томами (volumes) Docker

Тома (**volumes**) в **Docker** — это способ постоянного хранения данных, которые нужны контейнеру. В отличие от контейнера, который временный и может быть удалён, томы хранят данные независимо от его жизненного цикла.

Зачем нужны тома:

- Сохранение данных между перезапусками контейнера.
- Общий доступ к данным между несколькими контейнерами.
- Изоляция данных от образа.

Том монтируется внутрь контейнера: данные, которые пишет контейнер в эту директорию, будут сохраняться в томе.

```
docker run -v myvolume:/app/data myimage
```

/app/data — путь в контейнере.
myvolume — имя тома на хосте.

Локальная привязка к директории на хосте (bind mount)

```
docker run -v ${PWD}/data:/app/data myimage
```

Плюсы:

- Полный контроль: видишь все файлы напрямую.
- Используется, если хочешь в реальном времени видеть или менять данные снаружи.

Минусы:

- Зависит от ОС и структуры хоста.
- Проблемы с кросс-платформенностью (особенно между Linux/Windows).

Том с именем (named volume)

```
docker volume create myvolume
docker run -v myvolume:/app/data myimage
```

или

```
docker run -v myvolume:/app/data myimage
```

Docker сам управляет хранением: обычно в /var/lib/docker/volumes/....
Используется, когда неважно, где физически лежит том, но важно — сохранить и использовать его по имени.

Автоматически созданный том (anonymous volume)

```
docker run -v /app/data myimage
```

Docker сам создаёт том с рандомным именем, и монтирует его в /app/data.

Команды для работы с томами

```
docker volume ls                  # список всех томов
docker volume inspect myvolume    # детали о томе
docker volume rm myvolume         # удалить том
docker volume prune               # удалить все неиспользуемые
```

Тип	Синтаксис	Контроль над данными	Удобство	Ко ис
Bind-монты	`/path/on/host:/path/in/container`	Полный	Среднее	Ра: liv
Именованные тома	`name:/path/in/container`	Через Docker	Высокое	Пр по заг
Анонимные тома	`:/path/in/container`	Нет	Низкое	Вр ко