Лабораторная работа №2

**Тема**: Работа с Docker.

**Цель**: Познакомиться с возможностями и получить практические навыки работы с Docker.

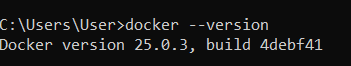
**Cсылки**:

1. <https://docs.docker.com>
2. <https://hub.docker.com/>
3. <https://doka.guide/tools/dockerfile/>
4. <https://k21academy.com/docker-kubernetes/docker-storage/>
5. <https://k21academy.com/docker-kubernetes/docker-tutorial>
6. <https://k21academy.com/docker-kubernetes/docker-networking-different-types-of-networking-overview-for-beginners/>
7. <https://ru.bitdegree.org/rukovodstvo/docker-dlja-novichkov/>
8. <https://habr.com/ru/companies/slurm/articles/528206/>
9. <https://itisgood.ru/2019/09/10/docker-setevoe-vzaimodejstvie-101/>
10. <https://itisgood.ru/2019/10/29/objasnenie-koncepcii-setej-v-docker/>
11. <https://www.tune-it.ru/web/adpashnin/blog/-/blogs/docker-network>

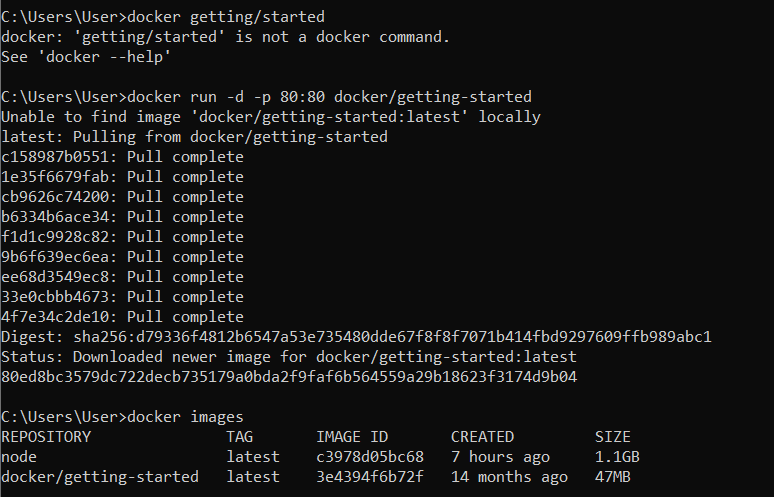
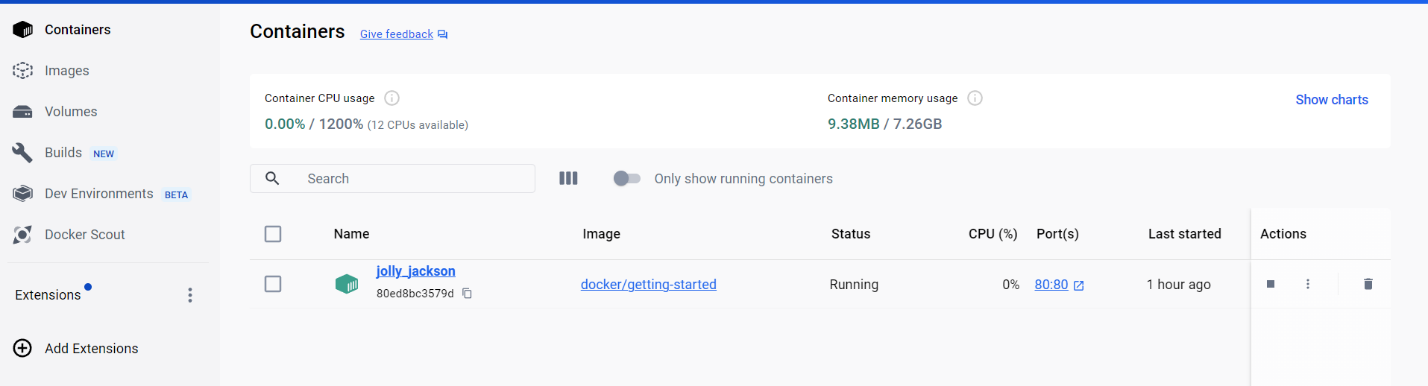
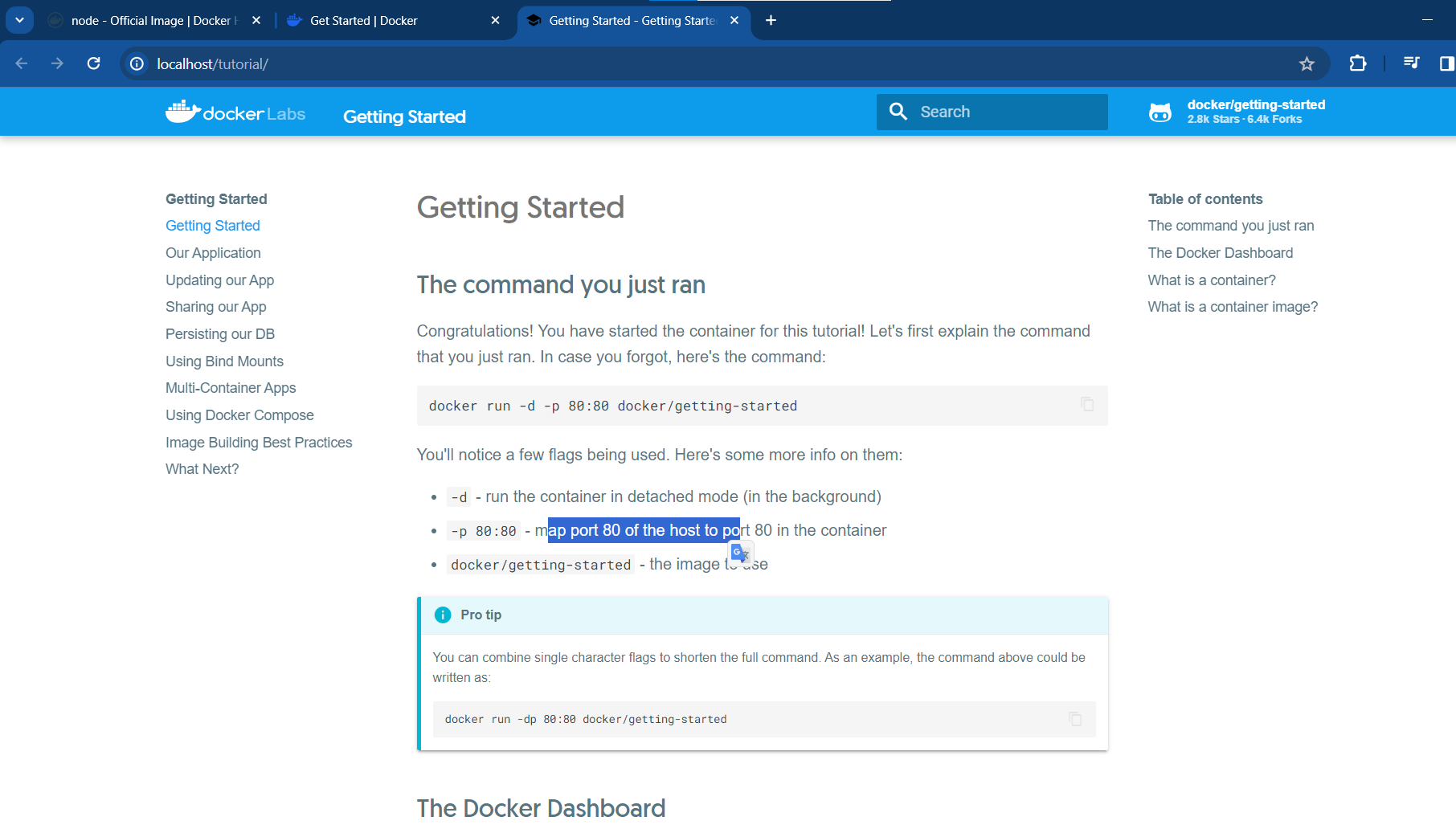
Для защиты ЛР необходимо

* Оформить Отчет со скринами кода в командной строке и результатами его выполнения
* Продемонстрировать работу с командами docker: build, tag, run, start, stop, pause, unpause, restart, ps, logs, ps, images, network, volumes, inspect, а также знать назначение инструкций в Dockerfile и Docker Compose

**Задание**:

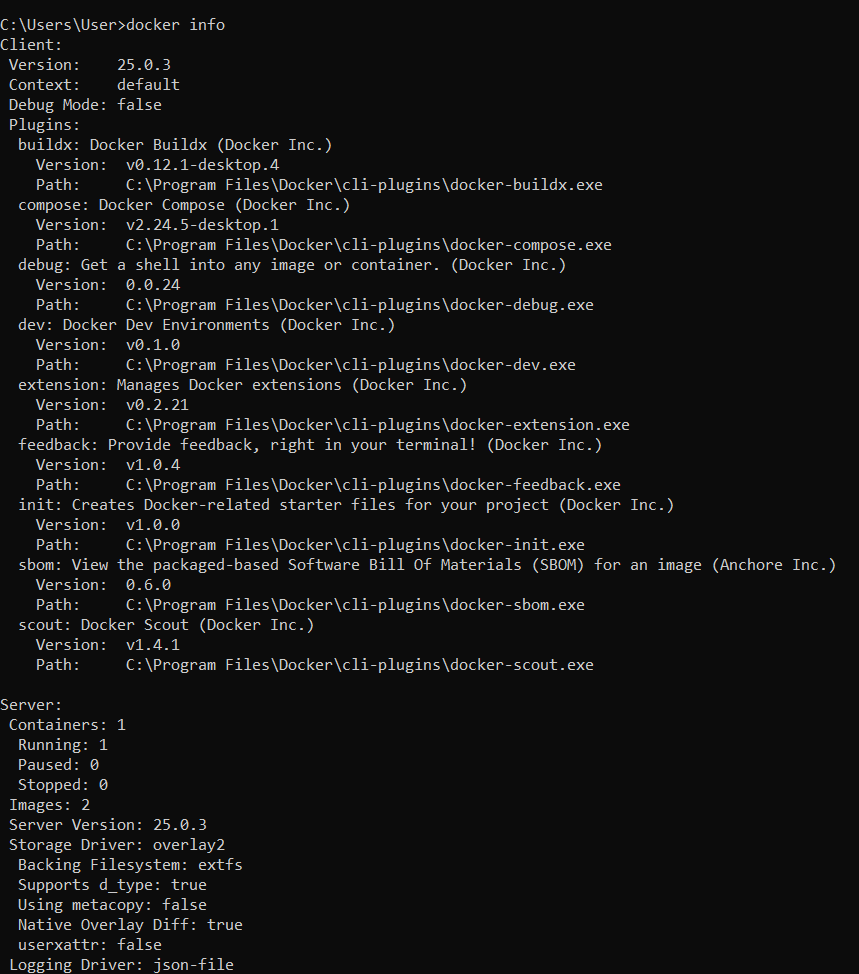
1. Подготовьте рабочее окружение в соответствии с типом вашей операционной системы
   * Установите Docker
   * Выполните базовую настройку
2. Изучите простейшие консольные команды и возможности Docker Desktop (см. лекцию), создать собственный контейнер docker/getting-started, открыть в браузере и изучить tutorial

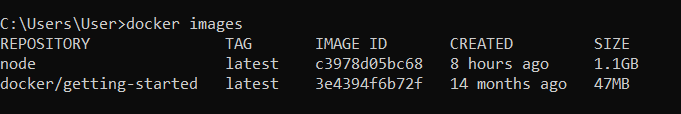
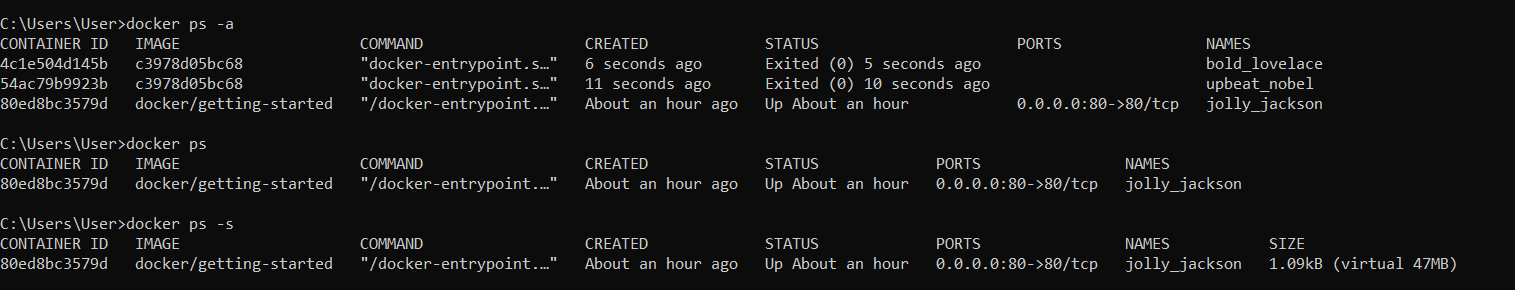
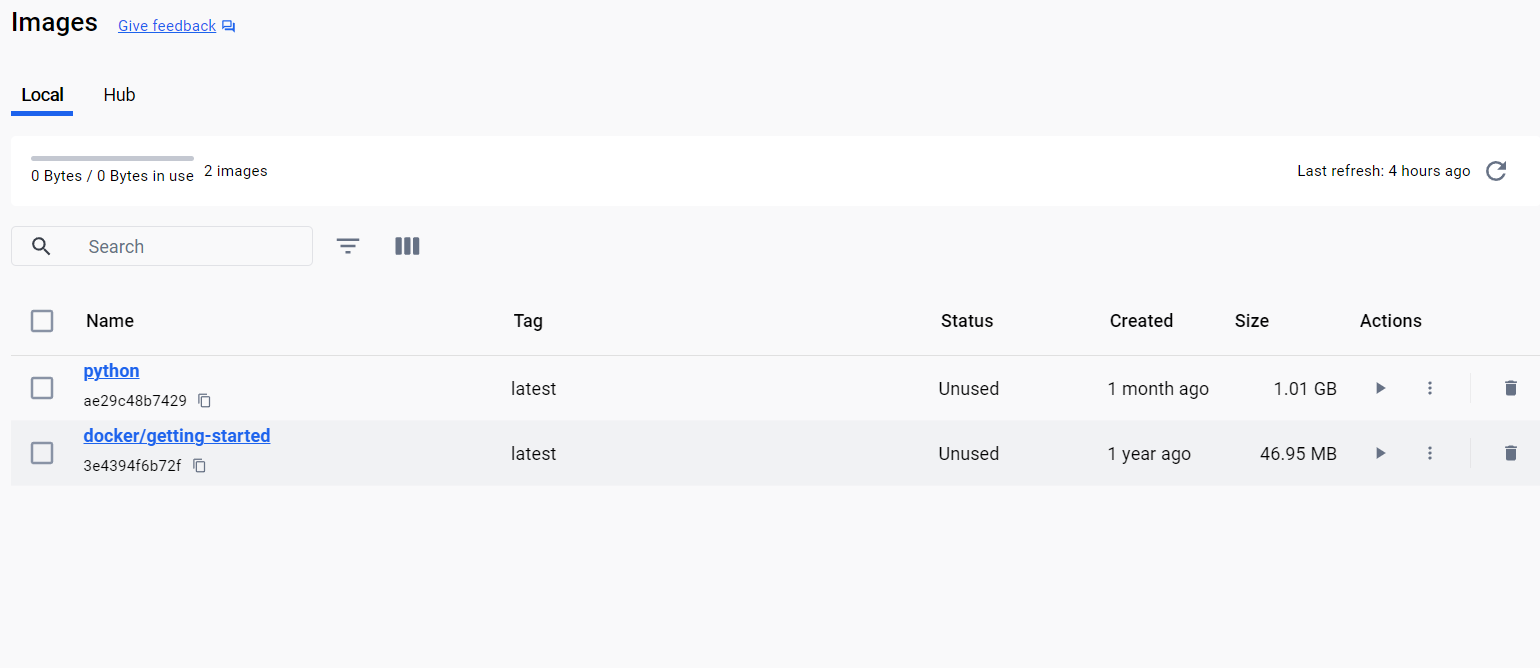
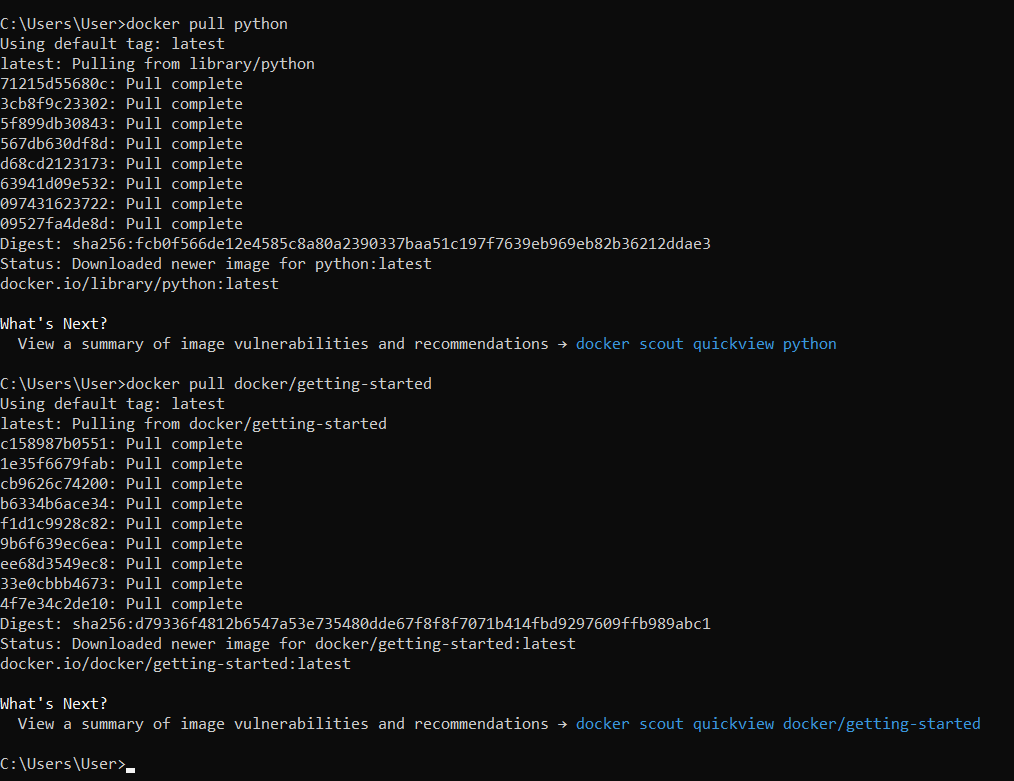
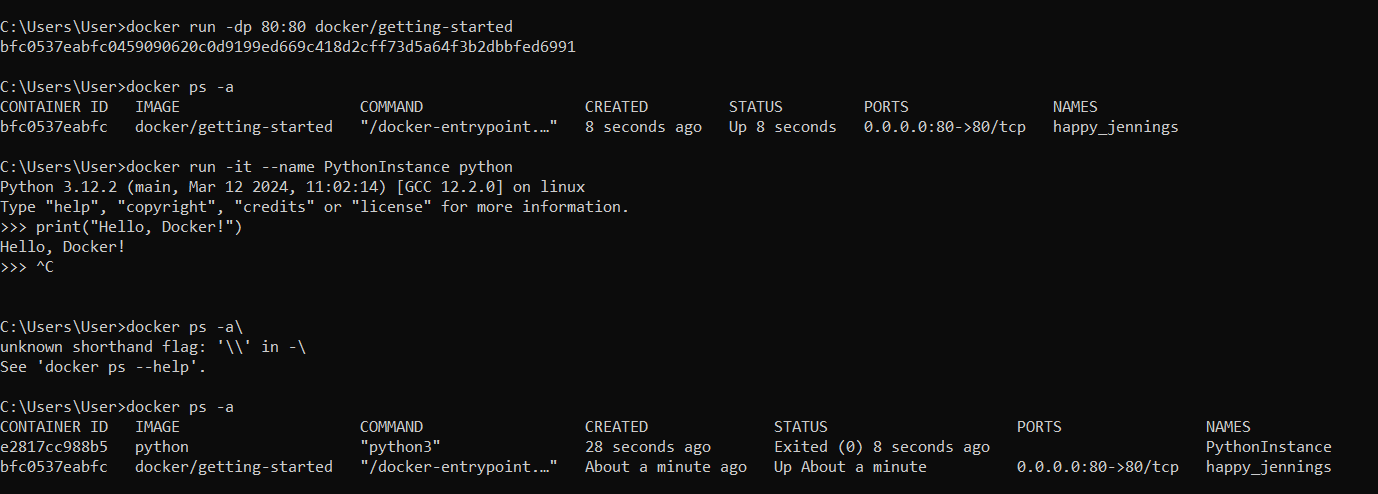
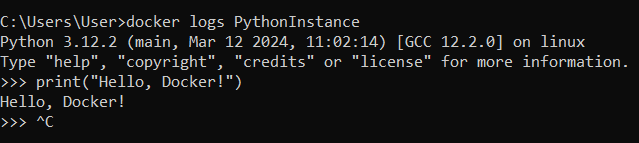
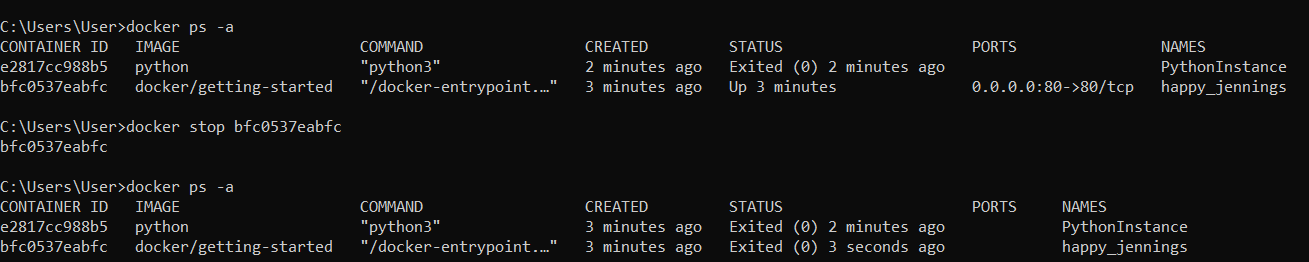
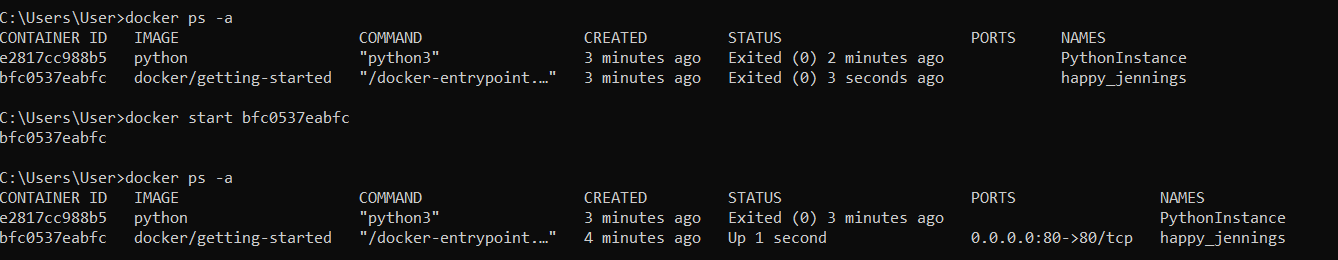
Консольный запуск docker getting/started

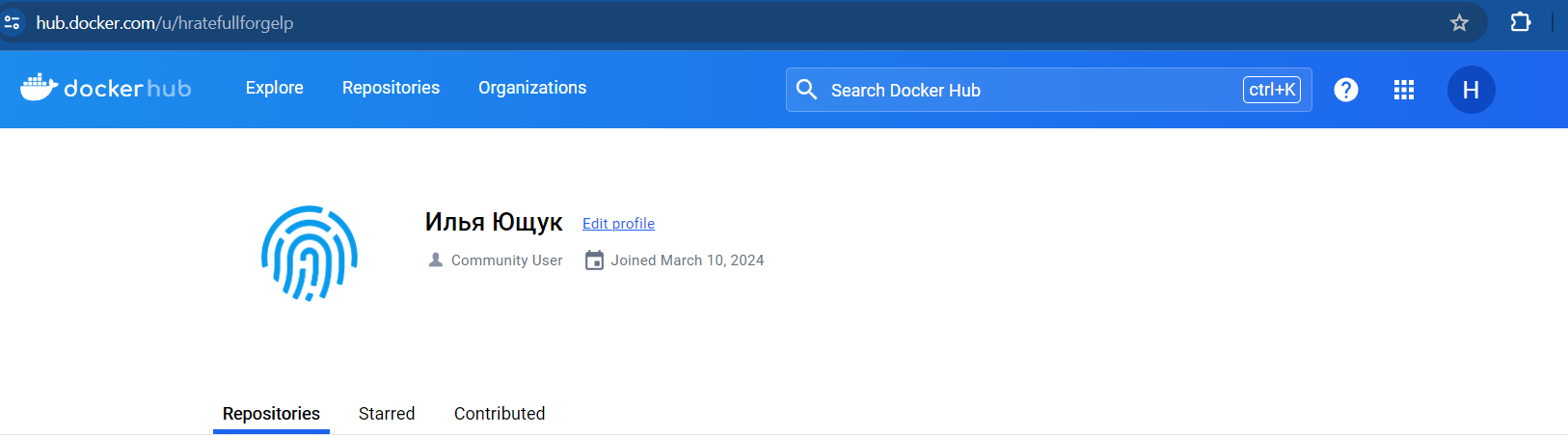
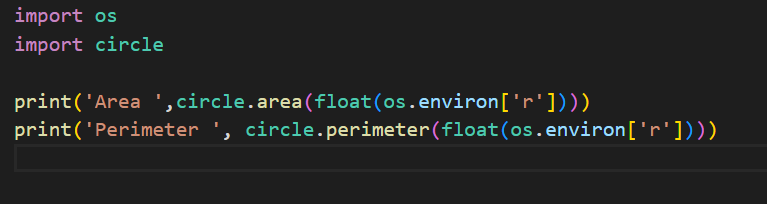
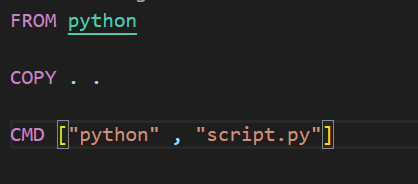
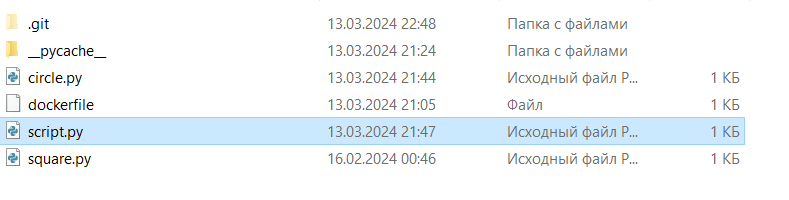
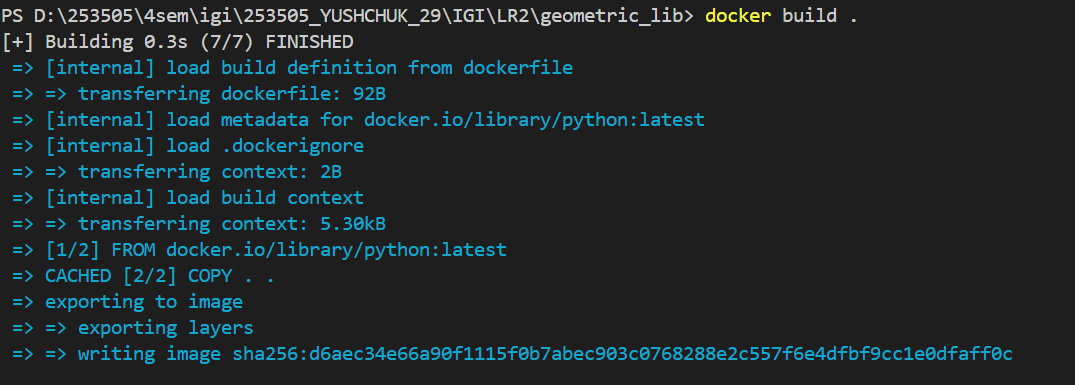
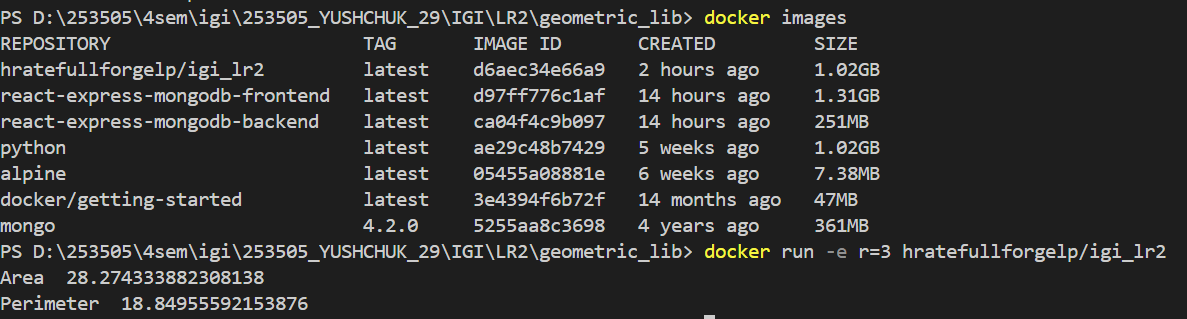
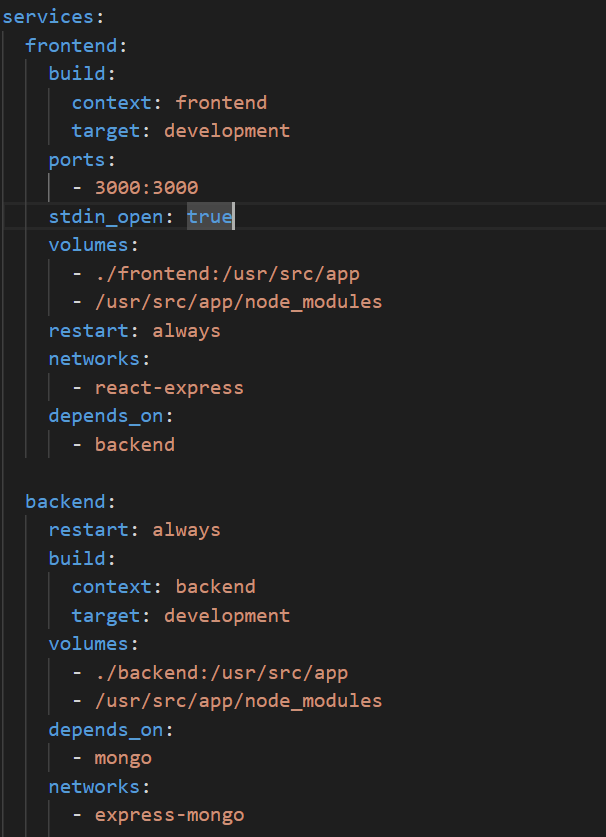
Docker Deskto

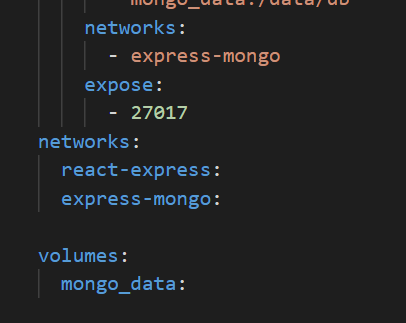
Основные команды:

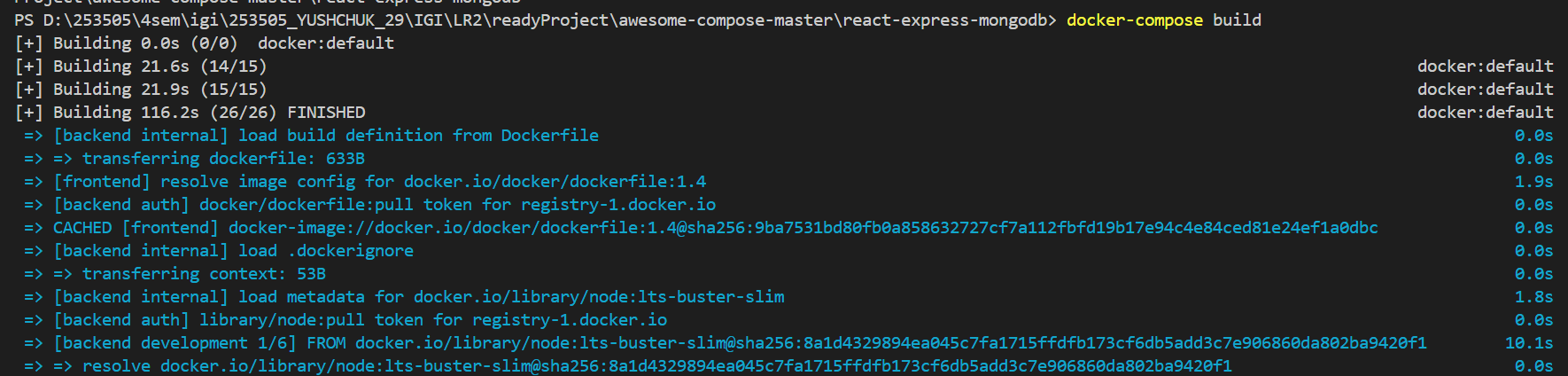
* docker info

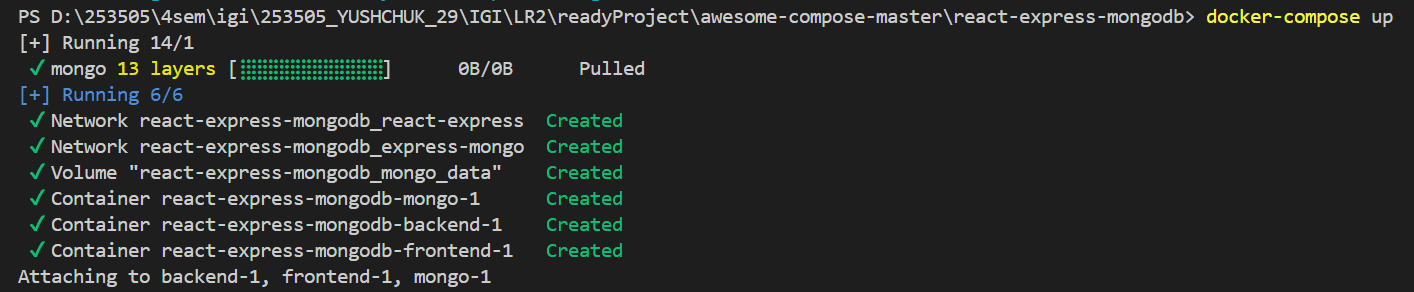
**

* *docker images*
* *docker ps*
* *docker pull*
* *docker run*
* *docker logs*
* *docker stop*
* *docker start*

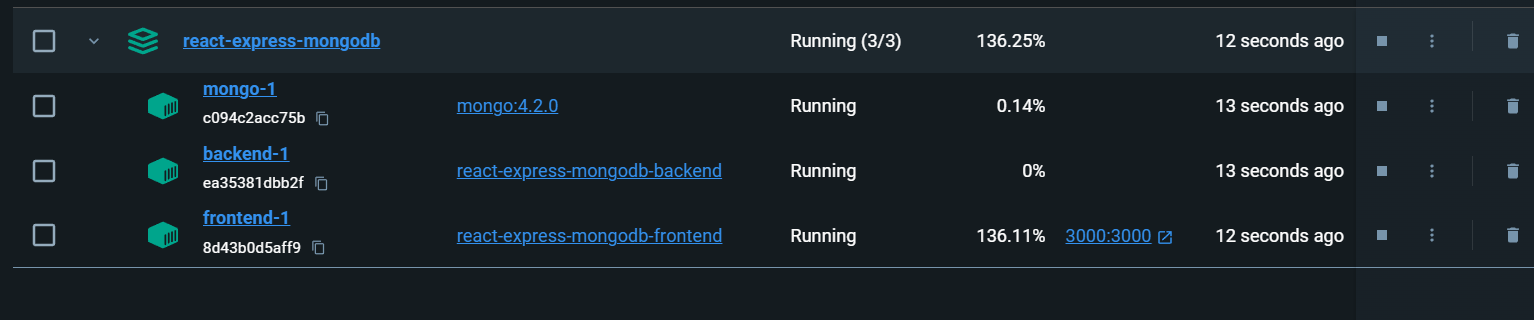
1. Создайте docker image, который запускает скрипт с использованием функций из https://github.com/smartiqaorg/geometric\_lib.
   1. Данные необходимые для работы скрипта передайте любым удобным способом (например: конфиг файл через docker volume, переменные окружения, перенаправление ввода). Изучите простейшие консольные команды для работы с docker(см. лекцию). Зарегистрируйтесь на DockerHub и выберите необходимые для проекта образы
   2. Создать Dockerfile для реализации сборки собственных Docker образов
   3. Использовать его для создания контейнера. Протестировать использование контейнера      
2. Скачать любой доступный проект с GitHub с произвольным стеком технологий или использовать свой, ранее разработанный. Создать для него необходимый контейнер, используя Docker Compose для управления многоконтейнерными приложениями. Запустить проект в контейнере.( Примеры Images: <https://hub.docker.com/_/phpmyadmin>, <https://hub.docker.com/_/mysql>, https://hub.docker.com/\_/postgres)
3. Настроить сети и тома для обеспечения связи между контейнерами и сохранения данных (исходные данные, логин, пароль и т.д.)

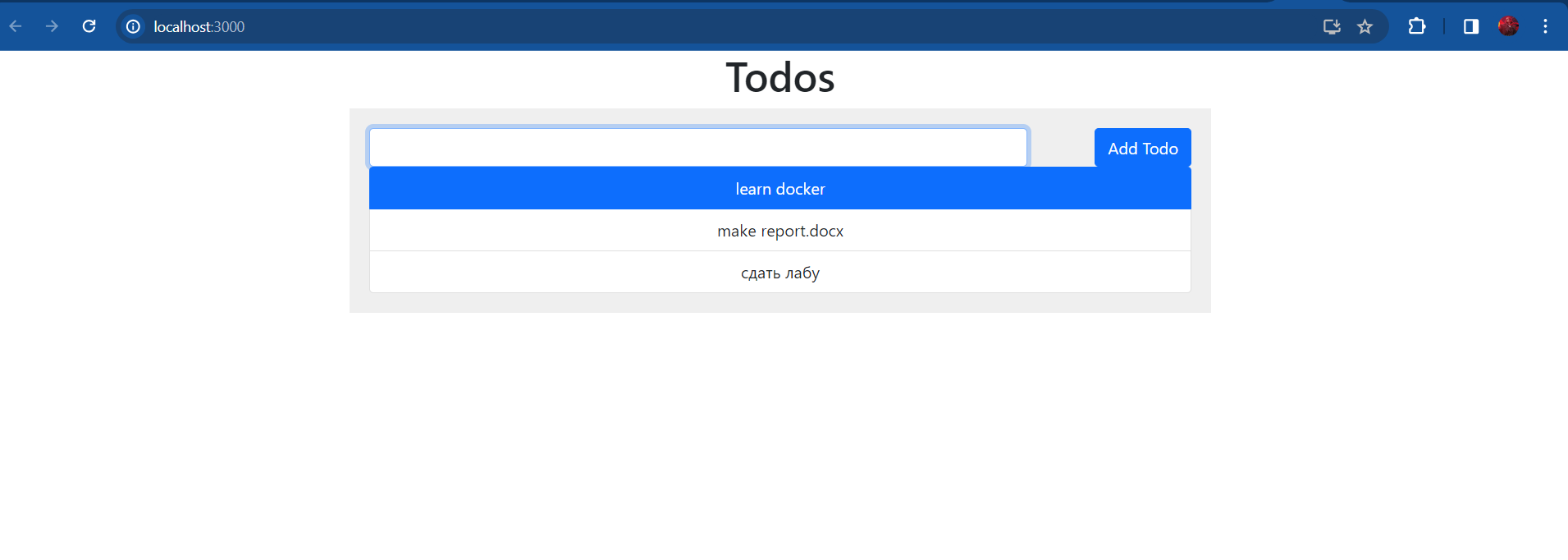


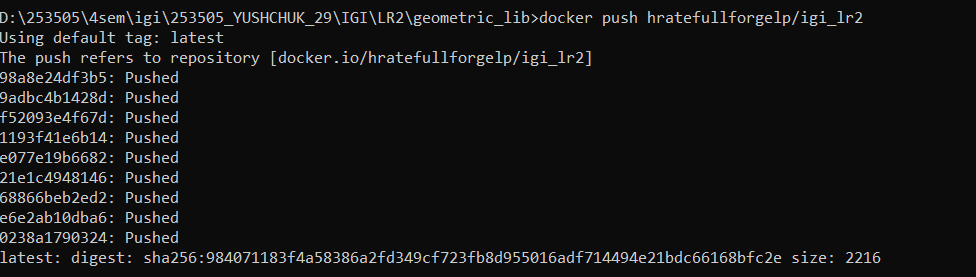
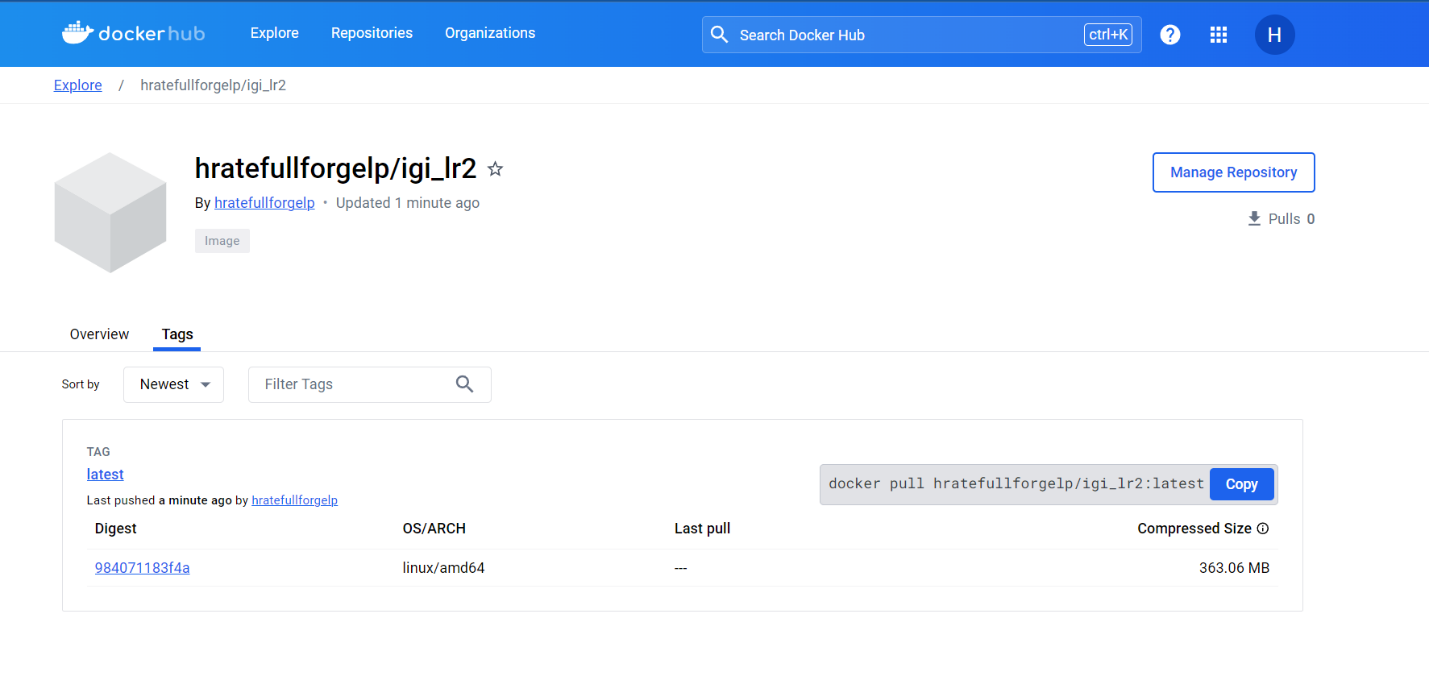
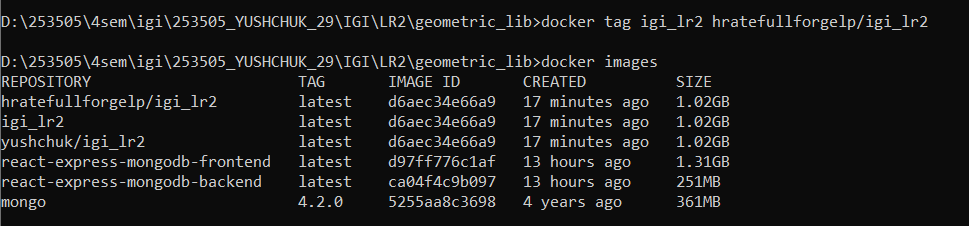
Билдим докер компос 

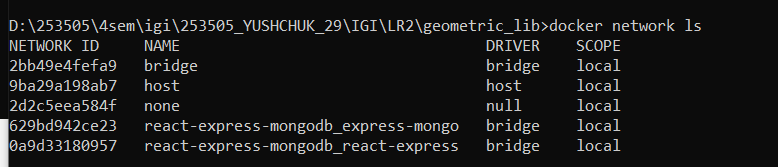
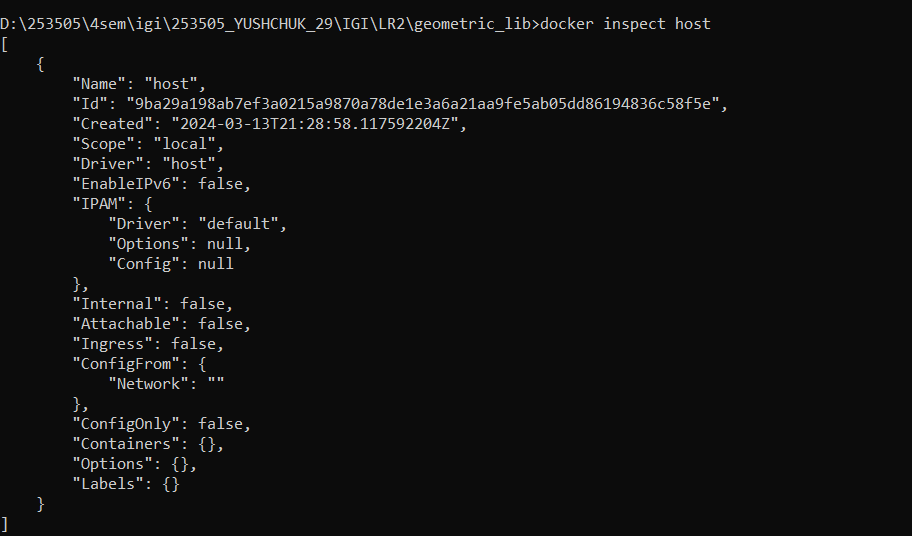
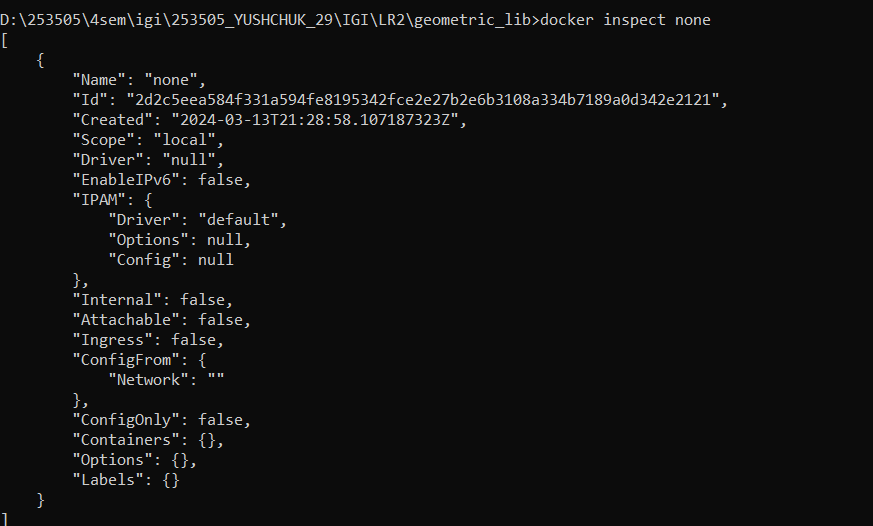
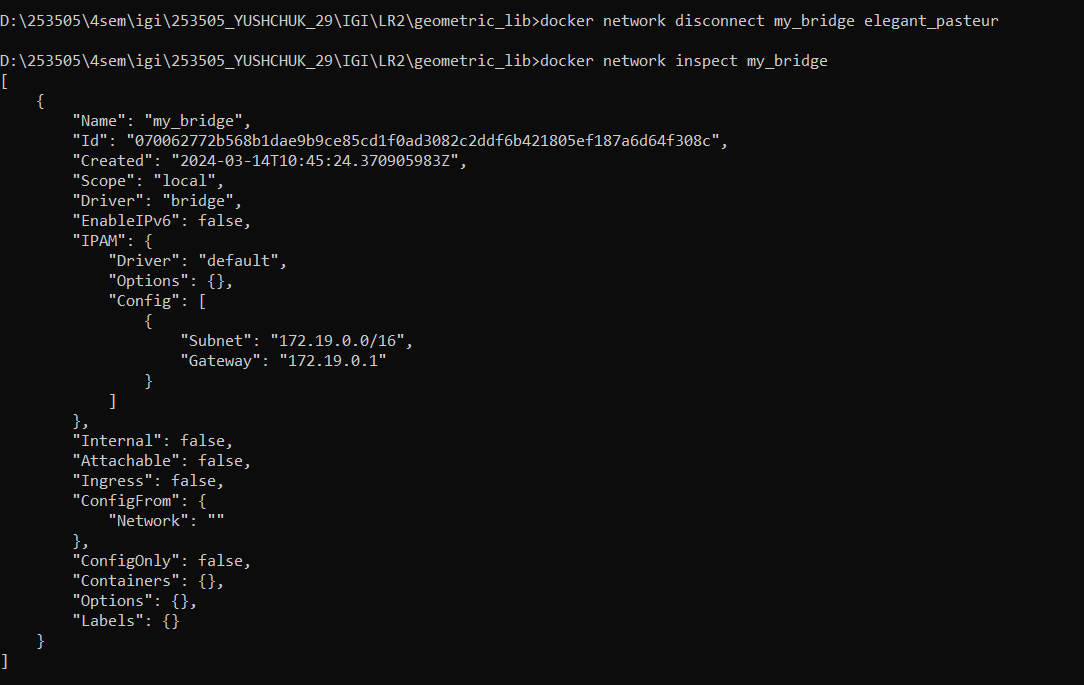
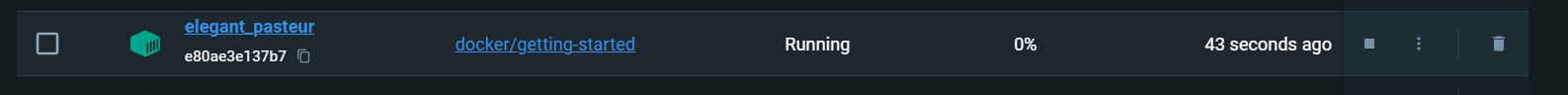
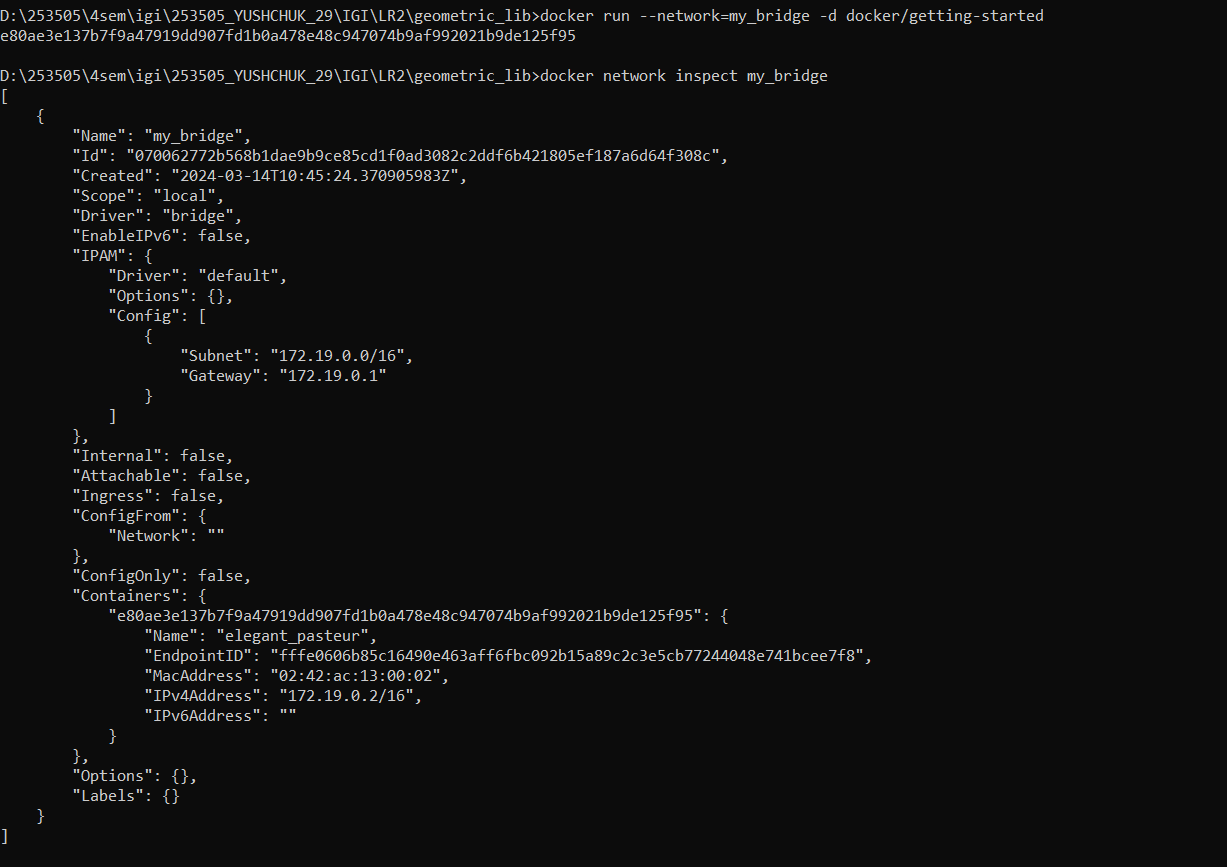
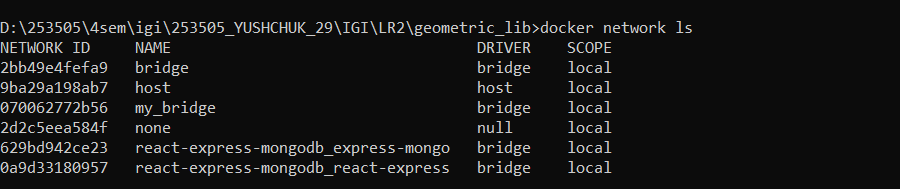
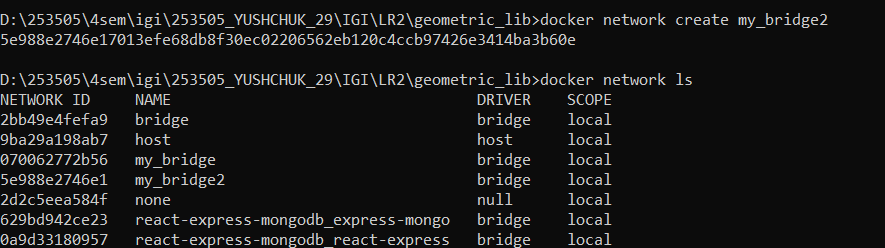
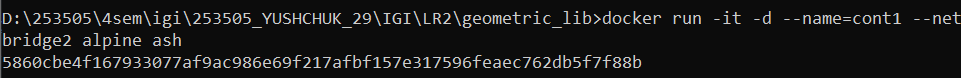
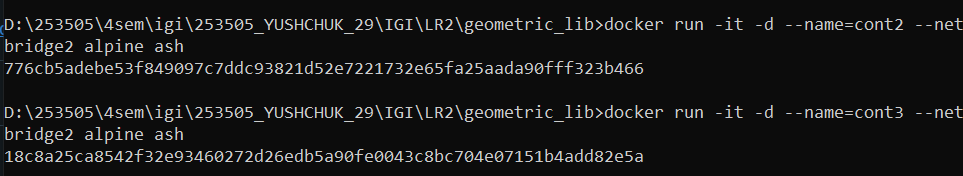
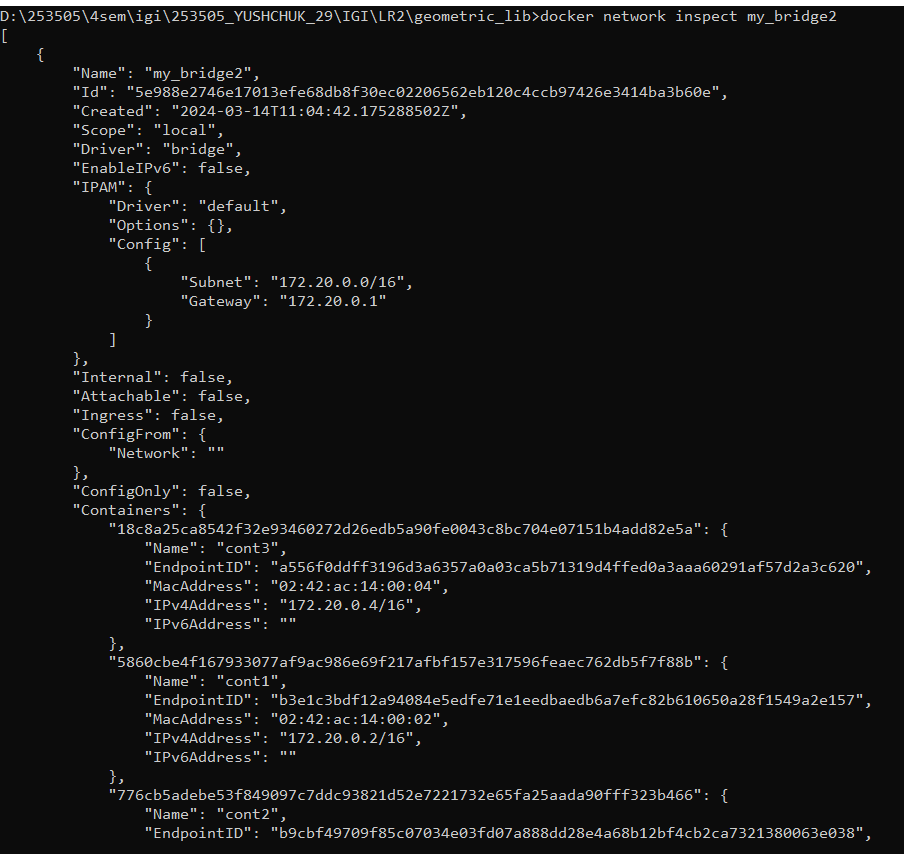
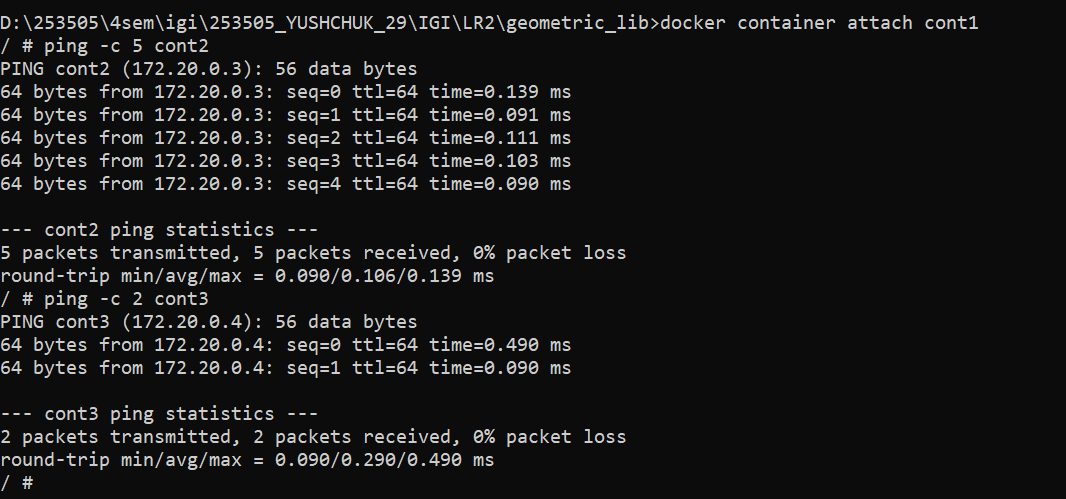
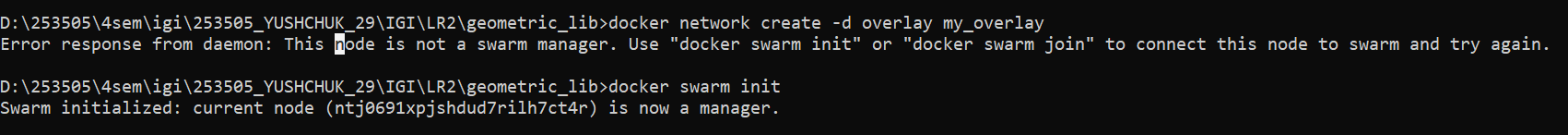
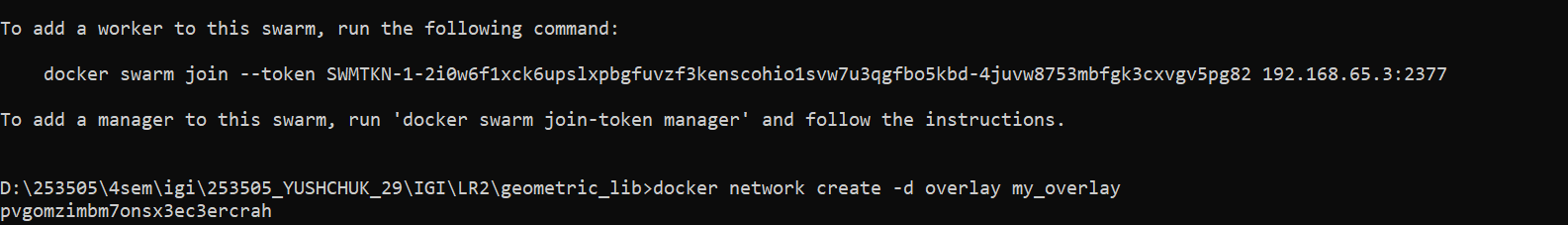
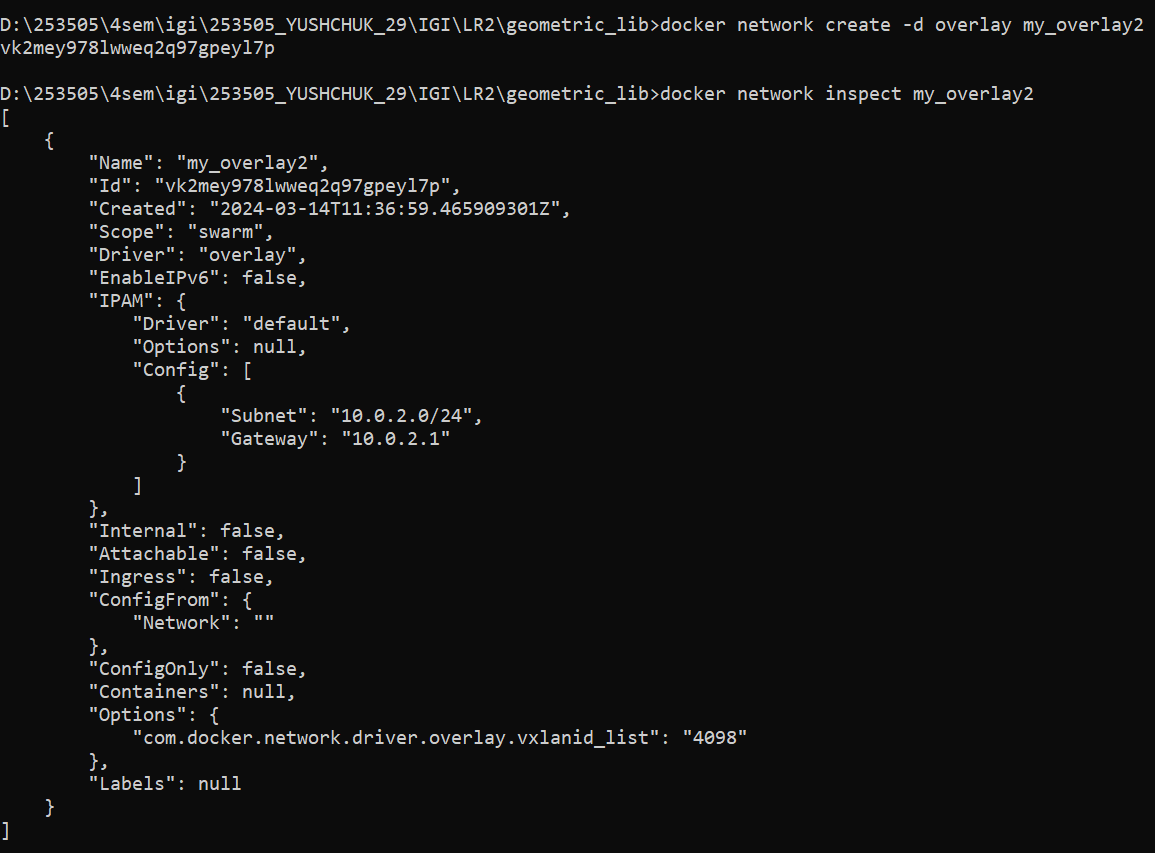
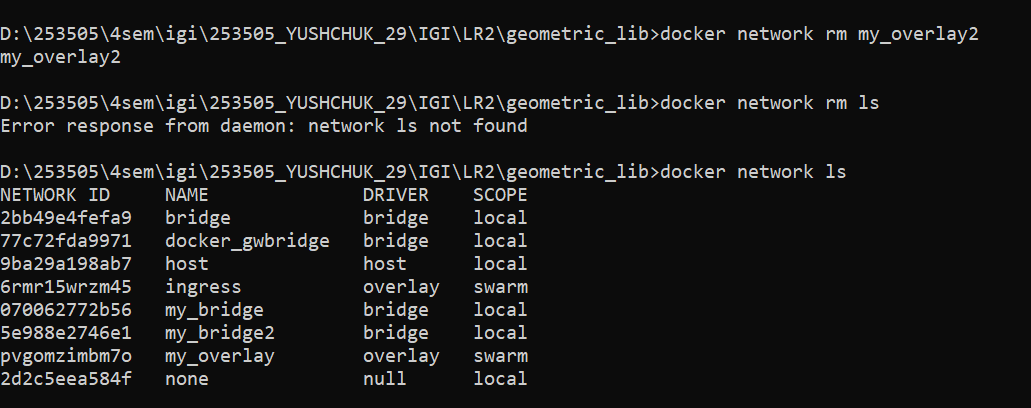
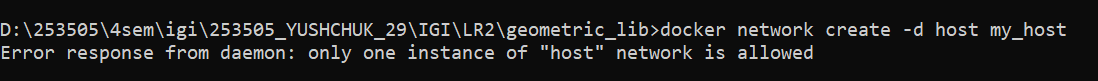
Запускаем проект 

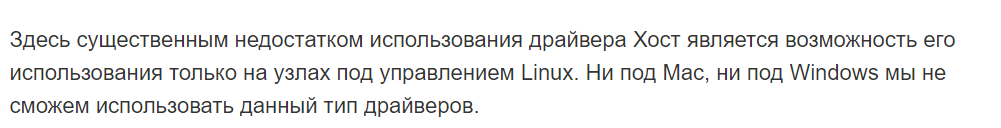
Docker Destop





1. Разместите результат в созданный репозиторий в DockerHub
2. Выполните следующие действия с целью изучить особенности сетевого взаимодействия:

* Получить информацию о всех сетях, работающих на текущем хосте и подробности о каждом типе сети
* Создать свою собственную сеть bridge, проверить, создана ли она, запустить Docker-контейнер в созданной сети, вывести о ней всю информацию(включая IP-адрес контейнера), отключить сеть от контейнера
* Создать еще одну сеть bridge, вывести о ней всю информацию, запустить в ней три контейнера, подключиться к любому из контейнеров и пропинговать два других из оболочки контейнера, убедиться, что между контейнерами происходит общение по IP-адресу    
* Создать свою собственную сеть overlay, проверить, создана ли она, вывести о ней всю информацию   
* Создать еще одну сеть overlay, проверить, создана ли она, вывести о ней всю информацию, удалить сеть  
* Попробовать создать сеть host, сохранить результат  в отчет. 



**Контрольные вопросы**

1. Что представляет собой контейнеризация?

**Контейнеры — это фактически форма виртуализации, позволяющая изолировать приложения. Все возможности виртуализации ОС недоступны, да и не нужны. Следовательно, это приводит к более быстрому времени загрузки! Контейнеризация** (*виртуализация на уровне операционной системы*, *контейнерная виртуализация*, *зонная виртуализация*[[1]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%B9%D0%BD%D0%B5%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F#cite_note-1)) — метод [виртуализации](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), при котором [ядро операционной системы](https://ru.wikipedia.org/wiki/%D0%AF%D0%B4%D1%80%D0%BE_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B) поддерживает несколько изолированных экземпляров [пространства пользователя](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D1%81%D1%82%D0%B2%D0%BE_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) вместо одного.

1. Что такое Docker?

**Docker** — это набор продуктов « [платформа как услуга](https://en.wikipedia.org/wiki/Platform_as_a_service) » (PaaS), которые используют [виртуализацию на уровне ОС](https://en.wikipedia.org/wiki/OS-level_virtualization) для доставки программного обеспечения в пакетах, называемых [*контейнерами*](https://en.wikipedia.org/wiki/Container_(virtualization)) .

1. Что является реестром Docker?

Реестр Docker — это хранилище образов Docker. Клиенты Docker подключаются к реестрам, чтобы загружать («вытягивать») образы для использования или выгружать («проталкивать») созданные ими образы. Реестры могут быть публичными или частными. Получается DockerHub

1. Что такое Docker Engine?

Docker Engine: часть Docker, которая создает и запускает контейнеры Docker

Движок Docker ([Docker Engine](https://www.docker.com/products/docker-engine)) — это клиент-серверное приложение. Компания Docker разделила движок Docker на два продукта.

В Docker Engine есть  **три**  компонента:

**а)** **Сервер** : это демон docker, называемый  **dockerd** . Он может создавать и управлять образами докеров, т.е. контейнерами, сетями.

**b)** **Rest API** : он используется для указания демону докера, что делать.

**c)** **Интерфейс командной строки (CLI)** : это клиент, который используется для ввода команд Docker.

1. Как происходит связь между клиентом Docker и демоном Docker?

 Клиент и демон общаются через REST API поверх UNIX-сокетов или сетевого интерфейса.

1. Для чего необходим образ контейнера?

**Образ** Docker — это шаблон только для чтения, используемый для создания контейнеров. Используются для хранения и доставки приложений.  Может быть несколько и за счет них вы наполняете общий контейнер (проект). Контейнер состоит из образов (готовых решений) и вашего кода.

1. Что представляет собой контейнер Docker?

**Контейнер** Docker — это стандартизированная инкапсулированная среда, в которой выполняются приложения, программный пакет со всем, что необходимо приложению для запуска: **операционная система, код приложения, среда выполнения, системные инструменты, системные библиотеки и т. д**. Управляется с помощью Docker API или [CLI](https://en.wikipedia.org/wiki/Command-line_interface) .

1. Каков метод создания Docker-контейнера?

Docker create

1. Каковы этапы жизненного цикла контейнера Docker?

Существуют разные этапы создания контейнера, который известен как жизненный цикл контейнера, т. е. **создание**, **запуск**, **приостановка**, **удаление** и **остановка**. **create**, **run,** **pause**, **delete** & **stopped**

Первая фаза – это созданное состояние. Далее контейнер переходит в рабочее состояние, пока мы используем команду **Docker run** .

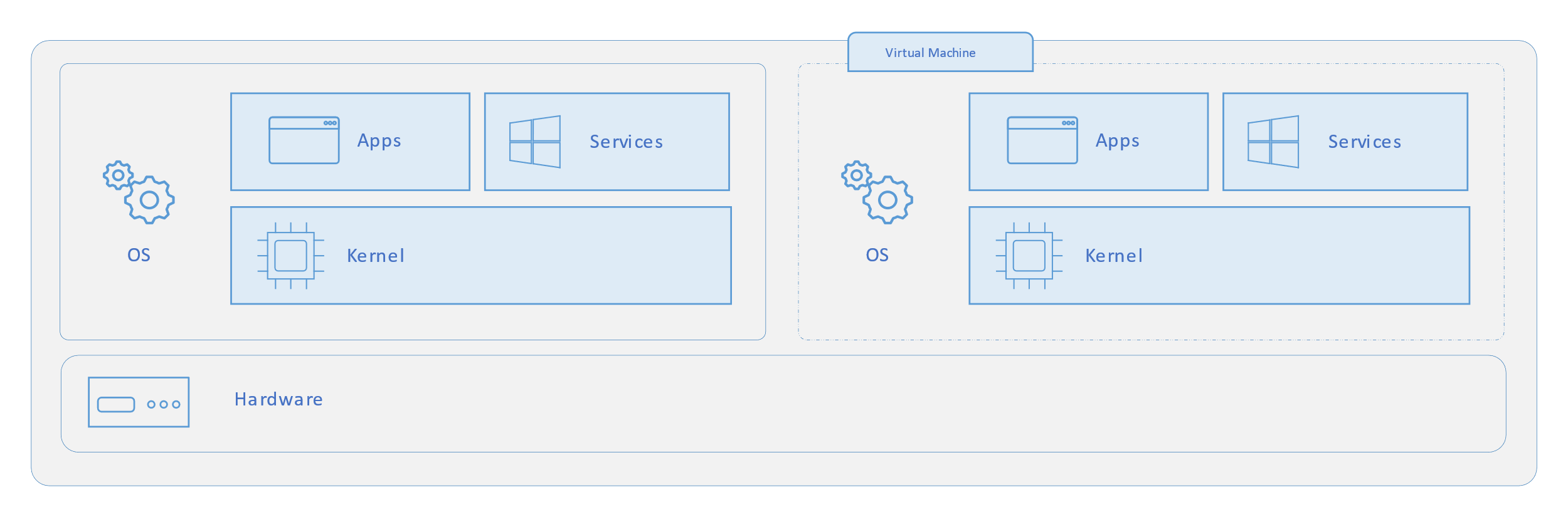
Мы можем остановить или приостановить работу контейнера с помощью команды **Docker stop/pause** . А чтобы вернуть контейнер из остановленного состояния в работающее, мы используем команду **запуска Docker** .

Мы можем удалить запущенный или остановленный контейнер с помощью команды **Docker rm** .

1. В каких средах можно использовать контейнерные приложения? Поидее в любых. Windows, Linux Mac
2. Что такое виртуальная машина?

это виртуальный компьютер, который использует выделенные ресурсы реального компьютера (процессор, диск, адаптер).

1. Является ли Docker VM-технологией?

Нет. В отличие от контейнера, виртуальная машина (ВМ) работает под управлением полноценной операционной системы, включая ее собственное ядро, как показано на этой схеме.

1. Какие различия между виртуальными машинами и контейнерами Docker?

**Виртуальная машина (VM)** — это виртуальный компьютер со всеми виртуальными устройствами и виртуальным жёстким диском, на который и устанавливается новая независимая ОС (гостевая ОС) вместе с виртуальными драйверами устройств, управлением памятью и другими компонентами. Т. е. мы получаем абстракцию физического оборудования, позволяющую запускать на одном компьютере множество виртуальных компьютеров. Виртуальное оборудование отображается в свойствах системы, а установленные приложения взаимодействуют с ним как с настоящим. При этом сама виртуальная машина полностью изолирована от реального компьютера, хотя и может иметь доступ к его диску и периферийным устройствам.

**Docker** — это ПО для создания приложений на основе контейнеров. Контейнеры и виртуальные машины решают одну задачу, но делают это по-разному. Контейнеры занимают меньше места, т.к. переиспользуют большее количество общих ресурсов хост-системы чем VM, т.к. в отличие от VM, обеспечивает виртуализацию на уровне ОС, а не аппаратного обеспечение. Такой подход обеспечивает меньший объем занимаемого места на жёстком диске, быстрое развертывание и более простое масштабирование.

Docker-контейнер даёт более эффективный механизм инкапсуляции приложений, обеспечивая необходимые интерфейсы хост-системы. Данная возможность позволяет контейнерам разделить ядро системы, где каждый из контейнеров работает как отдельный процесс основной ОС, у которого есть своё собственное виртуальное адресное пространство, таким образом данные, принадлежащие разным областям памяти, не могут быть изменены.

1. Зачем использовать контейнеры?

***Контейнер*** — это абстракция на уровне приложения, объединяющая код и зависимости. Контейнеры всегда создаются из образов, добавляя доступный для записи верхний слой и инициализирует различные параметры. Т. к. контейнер имеет свой собственный слой для записи и все изменения сохраняются в этом слое, несколько контейнеров могут совместно использовать доступ к одному и тому же образу. Каждый контейнер можно настроить через файл в проекте docker-compose.yml, задавая различные параметры, такие как имя контейнера, порты, идентификаторы, зависимости между другими контейнерами. Если в настройках не задавать имя контейнера, то Docker каждый раз будет создавать новый контейнер, присваивая ему имя случайным образом.

Когда контейнер запускается из образа, Docker монтирует файловую систему для чтения и записи поверх любых слоев ниже. Именно здесь будут выполняться все процессы. При первом запуске контейнера начальный слой чтения-записи пуст. Когда происходят изменения, они применяются к этому слою; например, если вы хотите изменить файл, этот файл будет скопирован из нижнего слоя только для чтения в слой для чтения и записи. Версия файла, доступная только для чтения, все еще будет существовать, но теперь она скрыта под копией.

1. Что выдаст команда $ docker ps ?

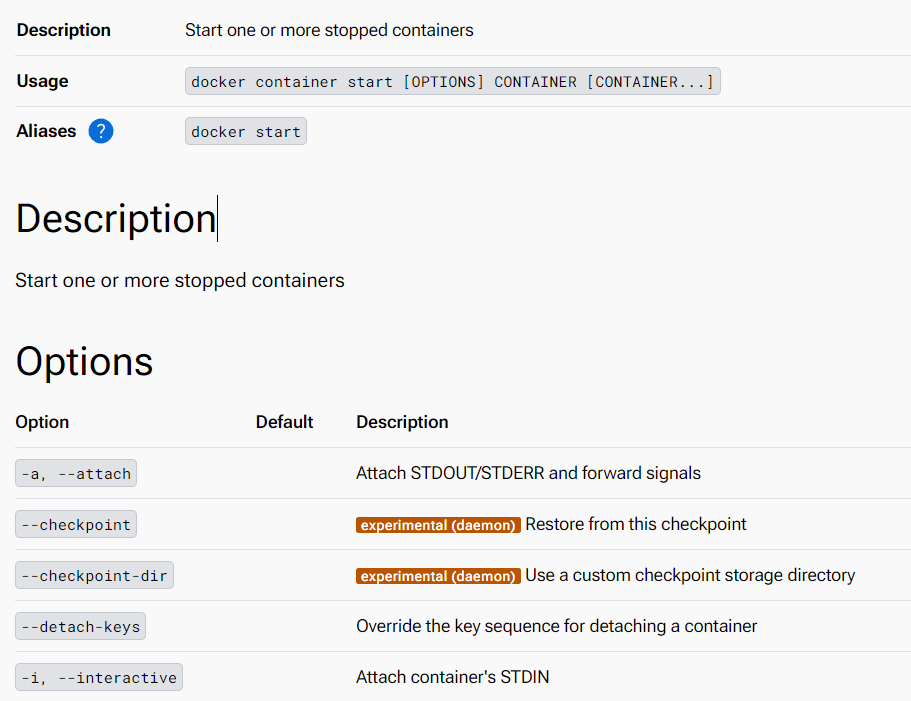
Список всех активных контейнеров

1. Сколько идентификаторов по умолчанию имеет каждый контейнер?
2. Как задается имя контейнера?

docker run --name= имя \_контейнера

1. Для чего нужна команда container start?

Start one or more stopped containers



1. Как создать контейнер без запуска?

Docker create

1. Что такое запуск контейнеров в интерактивном режиме?

Запуск контейнера в интерактивном режиме.Это значит, что вы сможете выполнять команды bash внутри контейнера с оболочкой. Для старта интерактивного режима используются опции -i и -t.

1. Что представляет собой образ?

Образ Docker – это снимок или схема библиотек и зависимостей, необходимых в контейнере для запуска приложения.

1. Что такое Dockerfile?
2. Для чего необходим Docker Compose?
3. Какая структура файла Compose?
4. Как запустить несколько контейнеров, используя один сервис?
5. Что такое предоставление порта связанным службам?
6. Для чего используются тома?
7. Для чего применяют переменные среды?
8. Как определяют правила связи между контейнерами? Написать файл Docker для создания и копирования каталога и построить его с использованием модулей Python?
9. Какие сети доступны по умолчанию в Docker?
10. Приведите необходимые шаги для развертывания докеризированного приложения, сохраненного в репозитории Git