Лабораторная работа №2

**Тема**: Работа с Docker.

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

**Cсылки**:

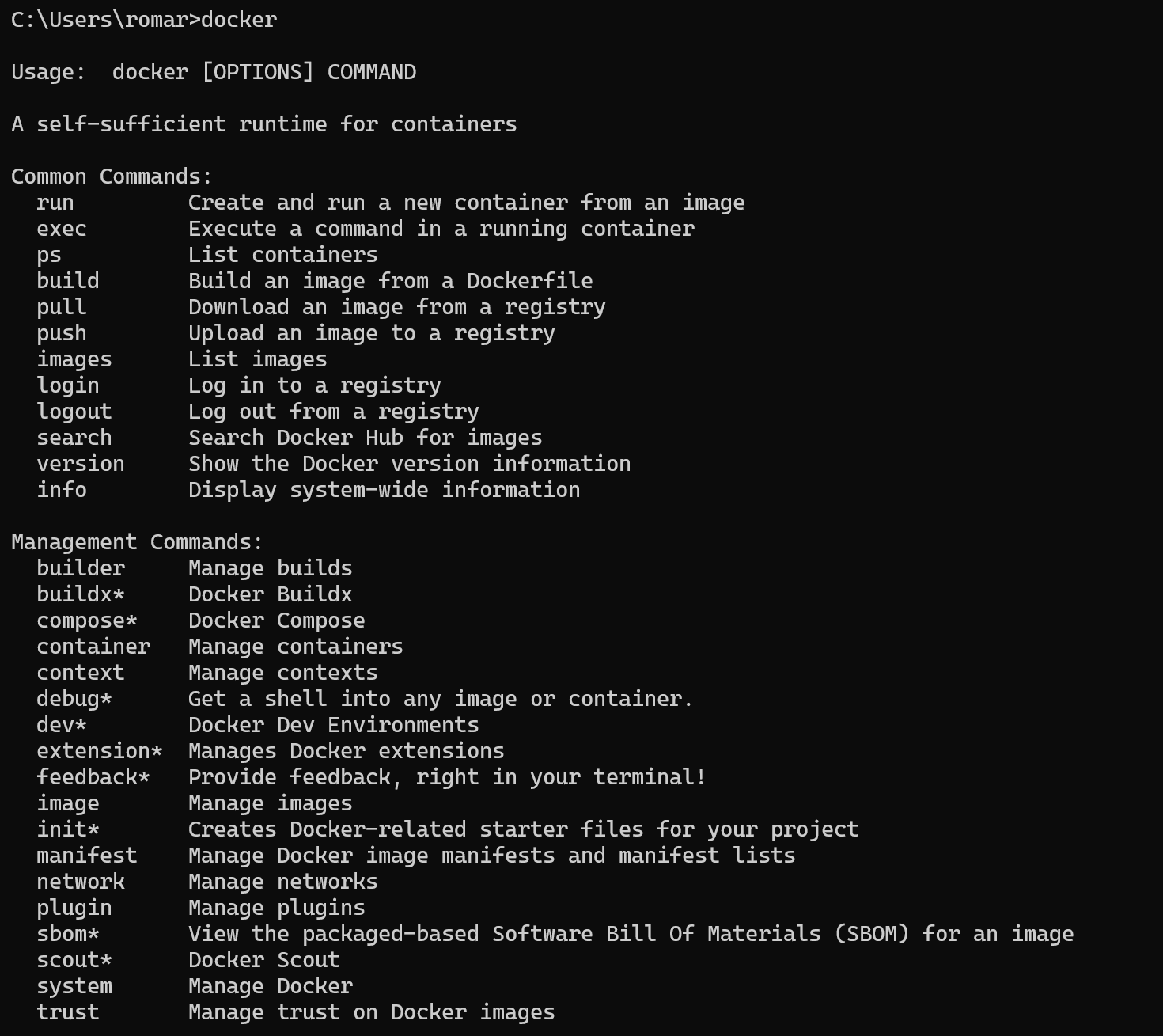
1. <https://docs.docker.com>
2. <https://hub.docker.com/>
3. <https://doka.guide/tools/dockerfile/>
4. <https://k21academy.com/docker-kubernetes/docker-storage/>
5. <https://k21academy.com/docker-kubernetes/docker-tutorial>
6. <https://k21academy.com/docker-kubernetes/docker-networking-different-types-of-networking-overview-for-beginners/>
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

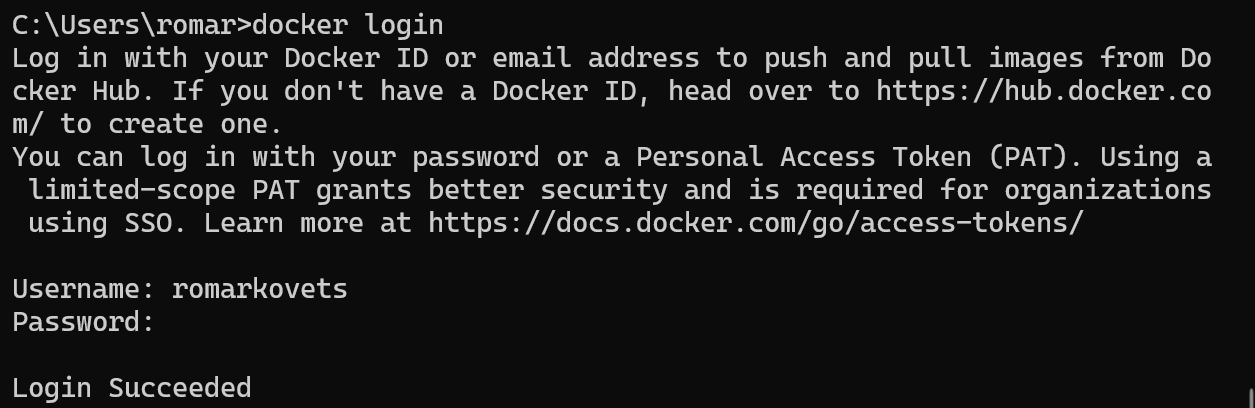
* Оформить Отчет со скринами кода в командной строке и результатами его выполнения
* Продемонстрировать работу с командами docker: build, tag, run, start, stop, pause, unpause, restart, ps, logs, ps, images, network, volumes, inspect, а также знать назначение инструкций в Dockerfile и Docker Compose

**Задание**:

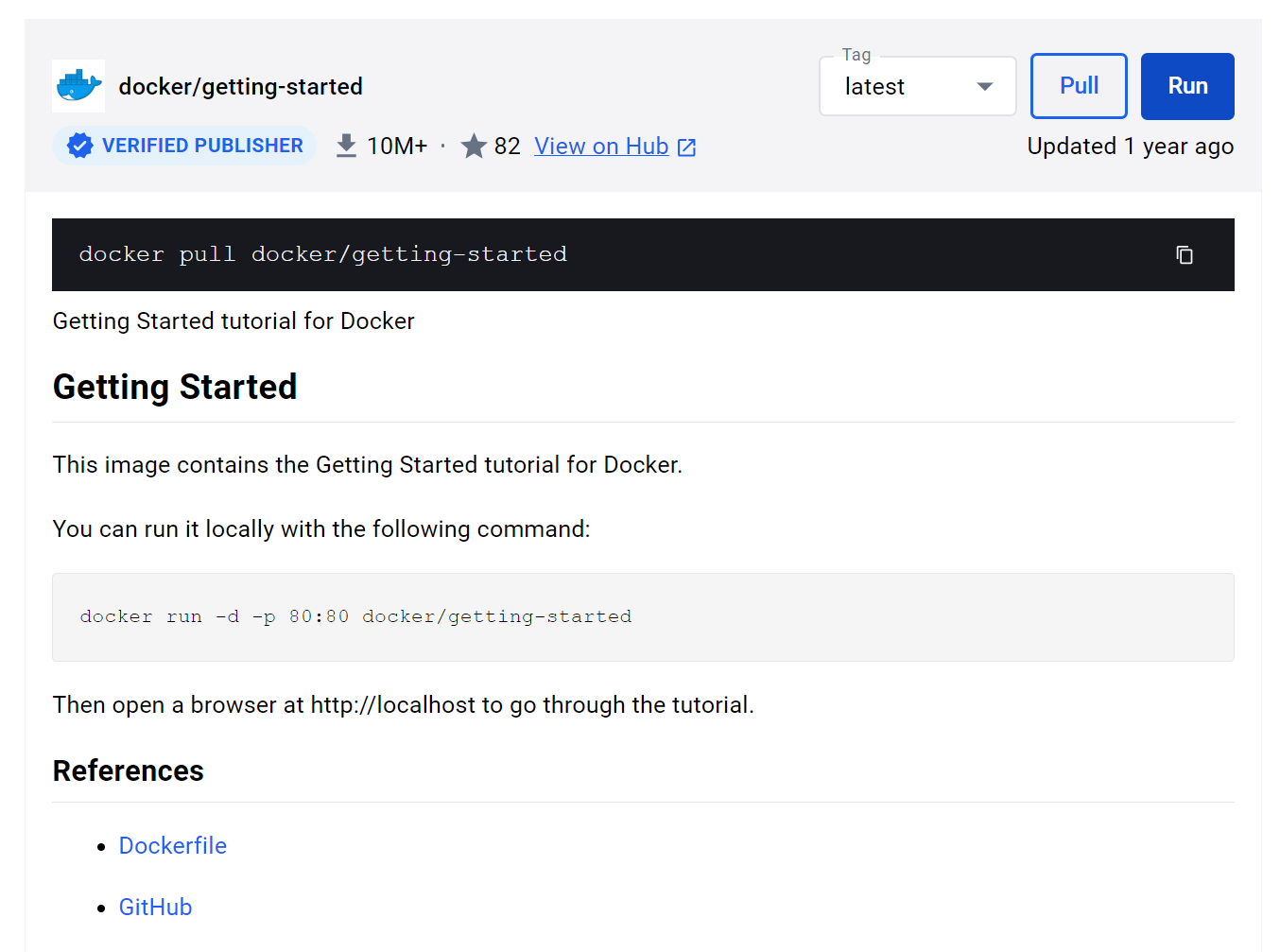
1. Подготовьте рабочее окружение в соответствии с типом вашей операционной системы
   * Установите Docker

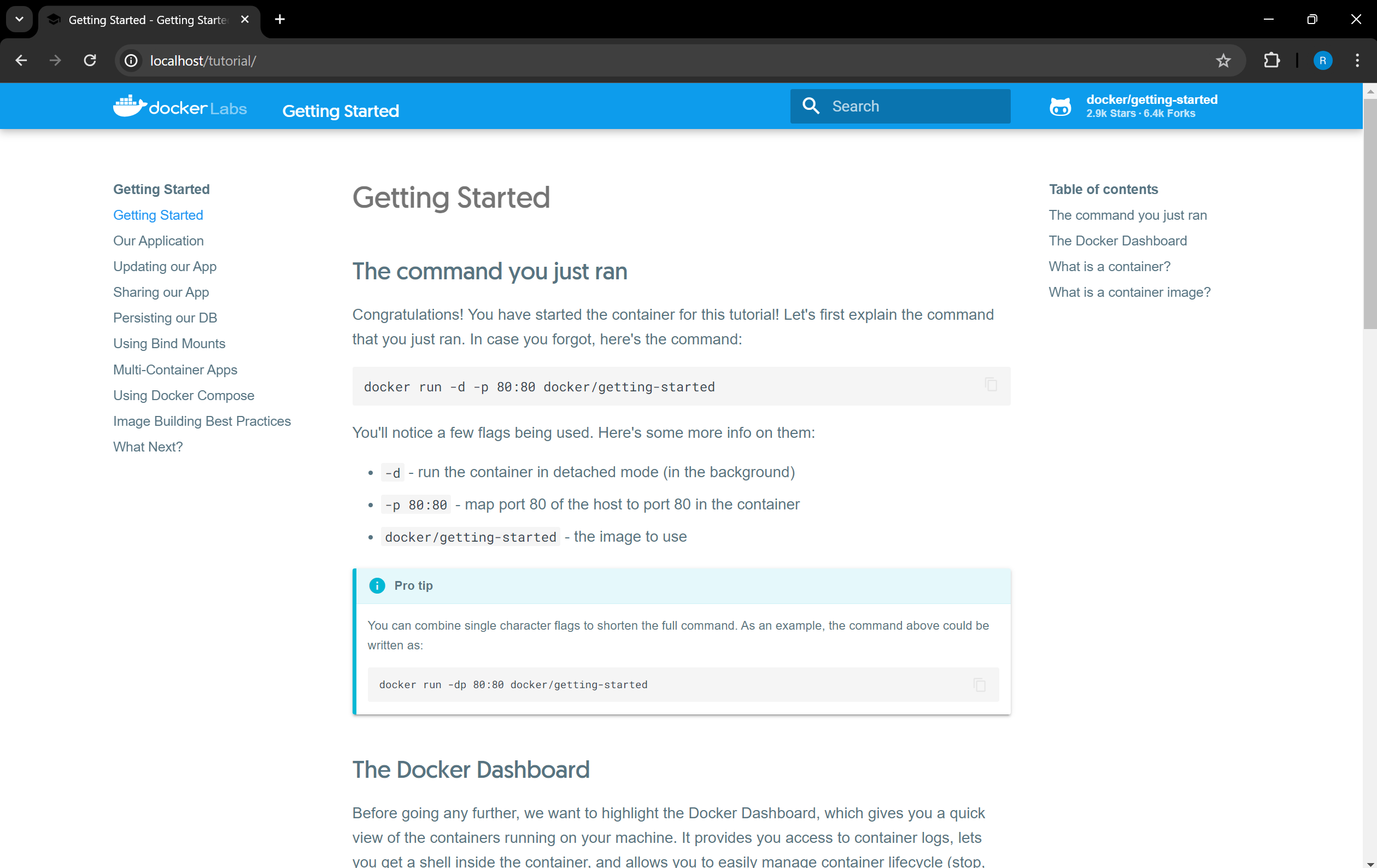


* + Выполните базовую настройку

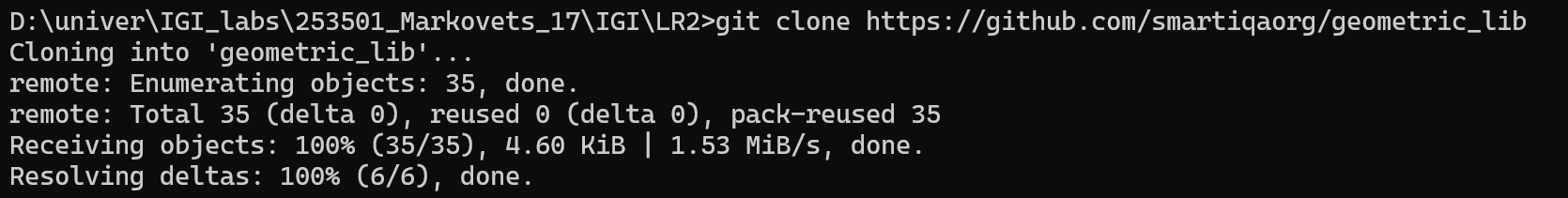


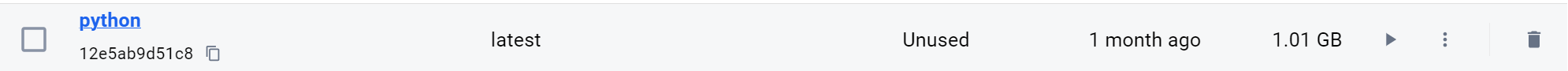
1. Изучите простейшие консольные команды и возможности Docker Desktop (см. лекцию), создать собственный контейнер docker/getting-started, открыть в браузере и изучить tutorial

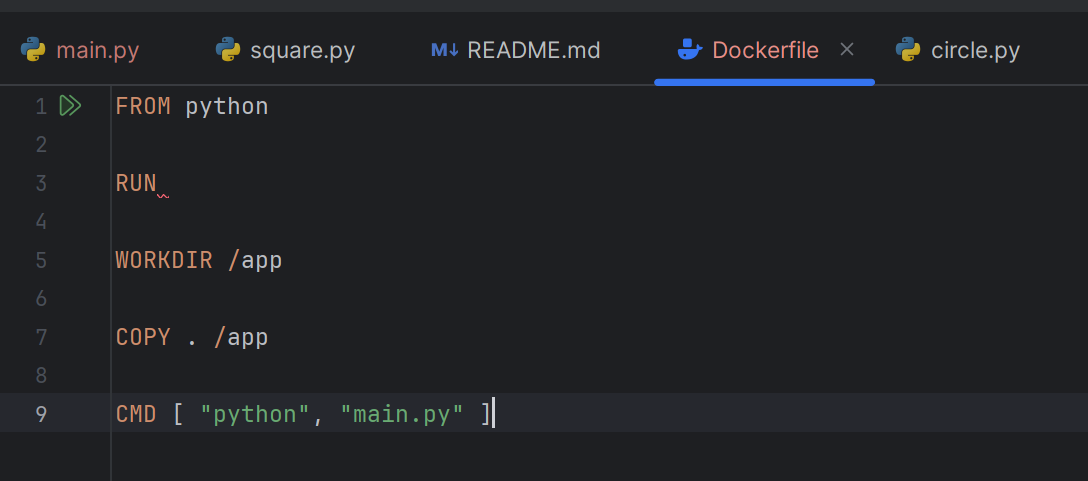
**

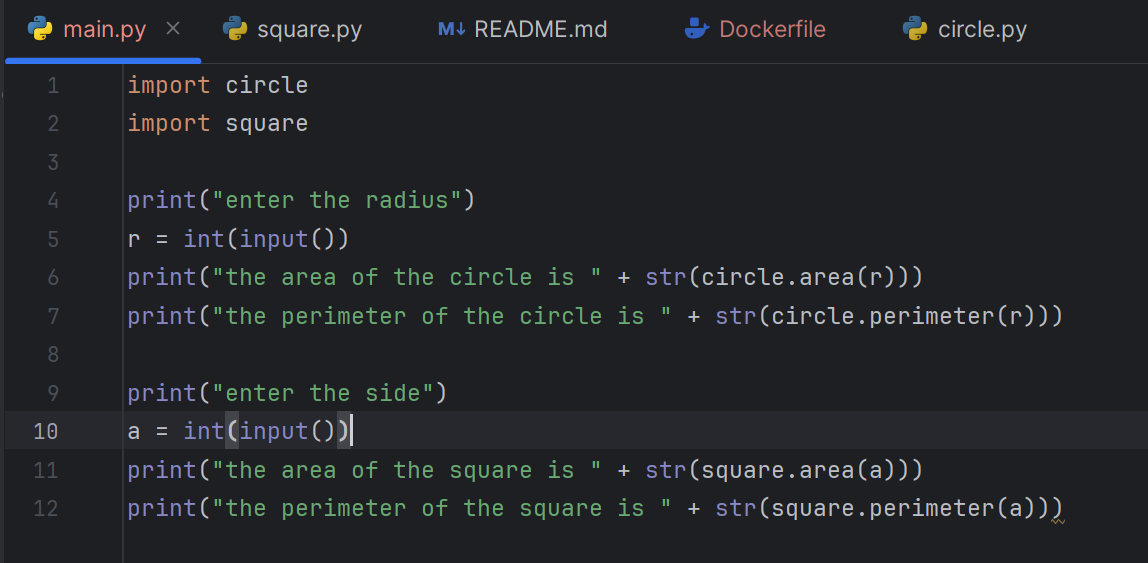
**

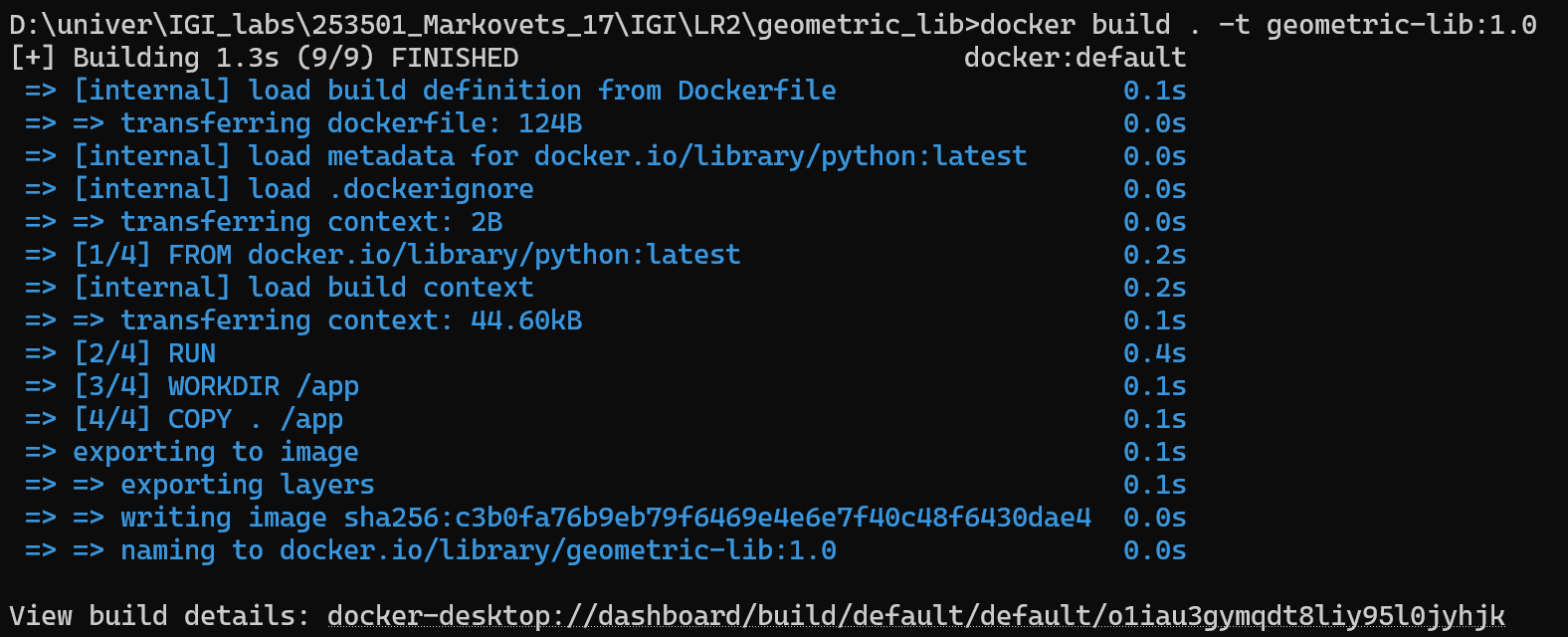
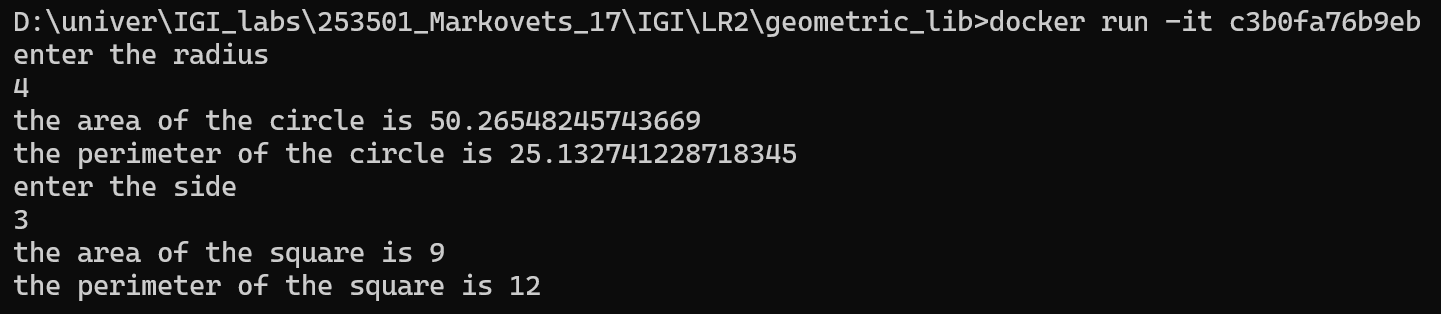
1. Создайте docker image, который запускает скрипт с использованием функций из https://github.com/smartiqaorg/geometric\_lib.



* 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  
     скачан образ python  
     
  2. Создать Dockerfile для реализации сборки собственных Docker образов

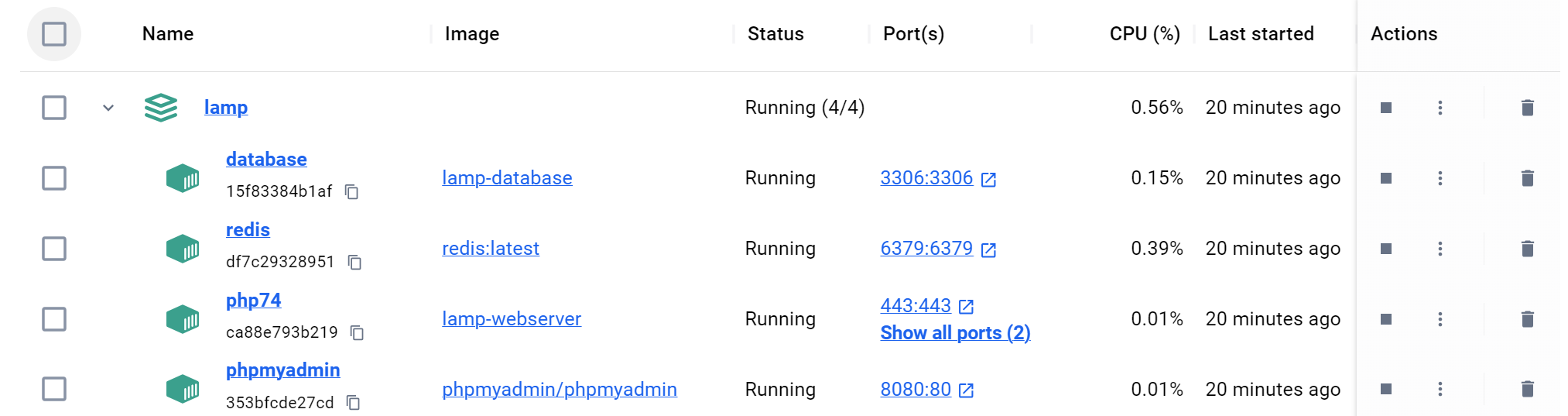
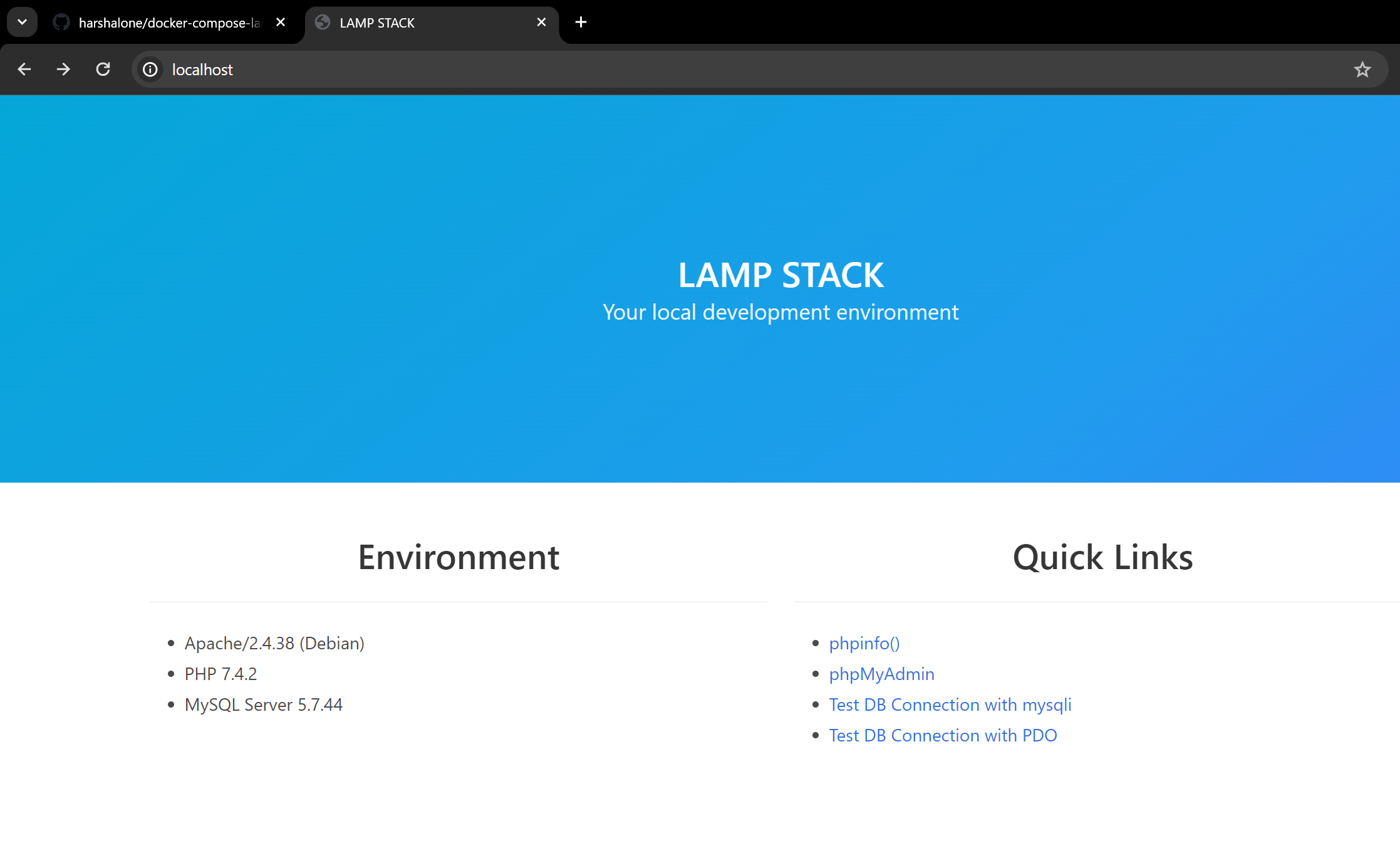


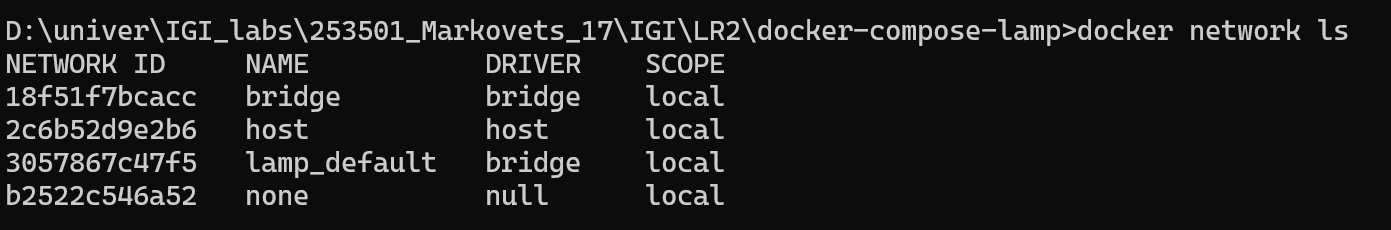
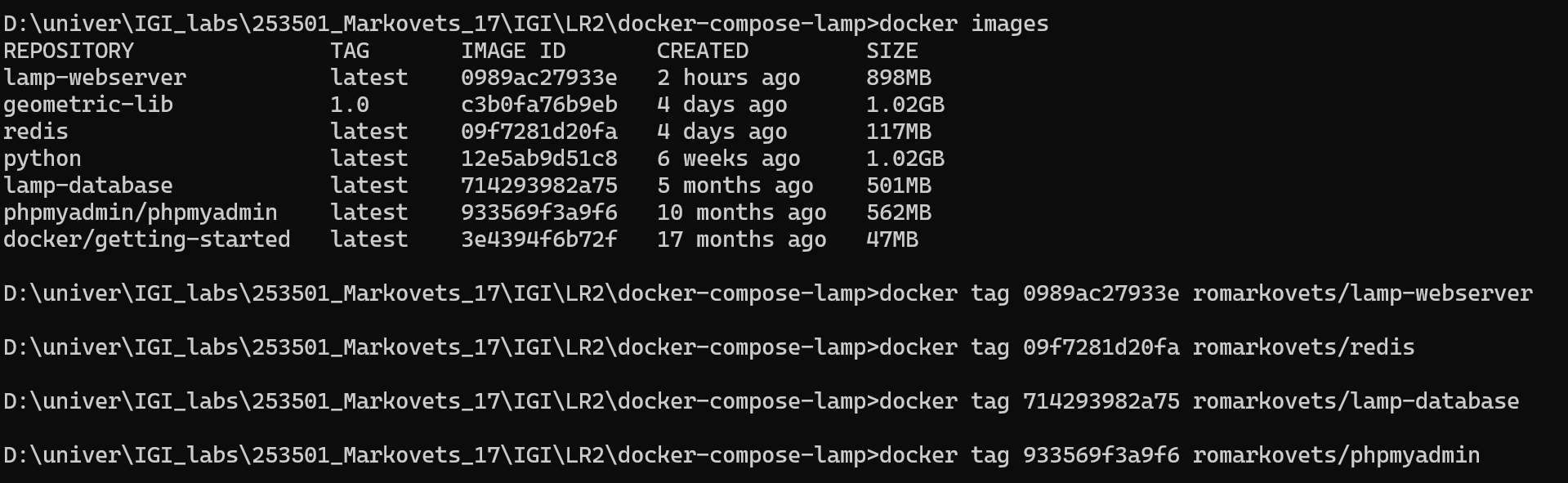
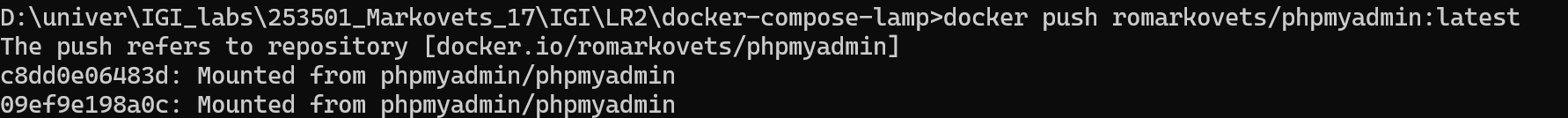
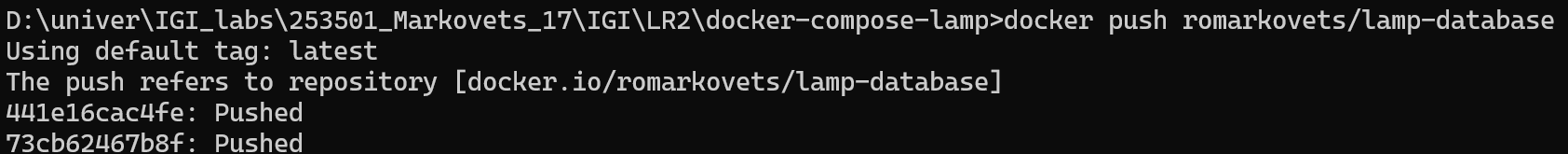
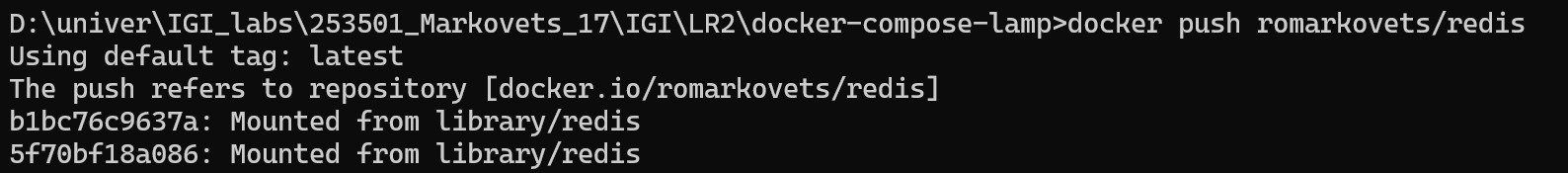
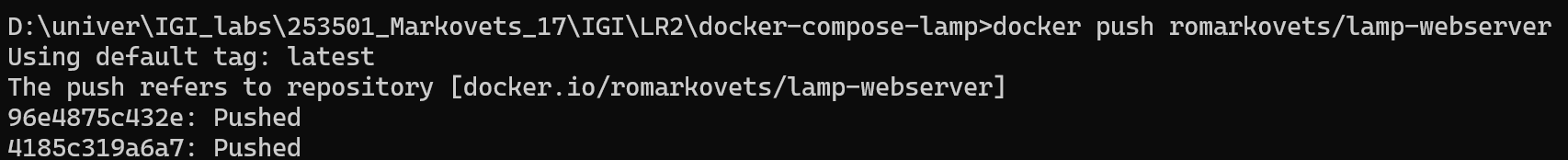


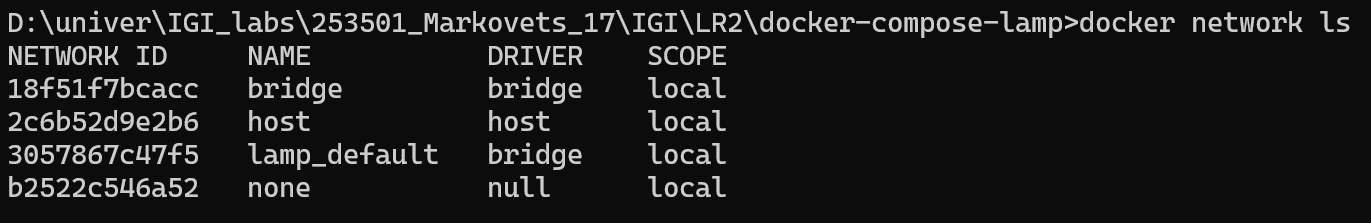
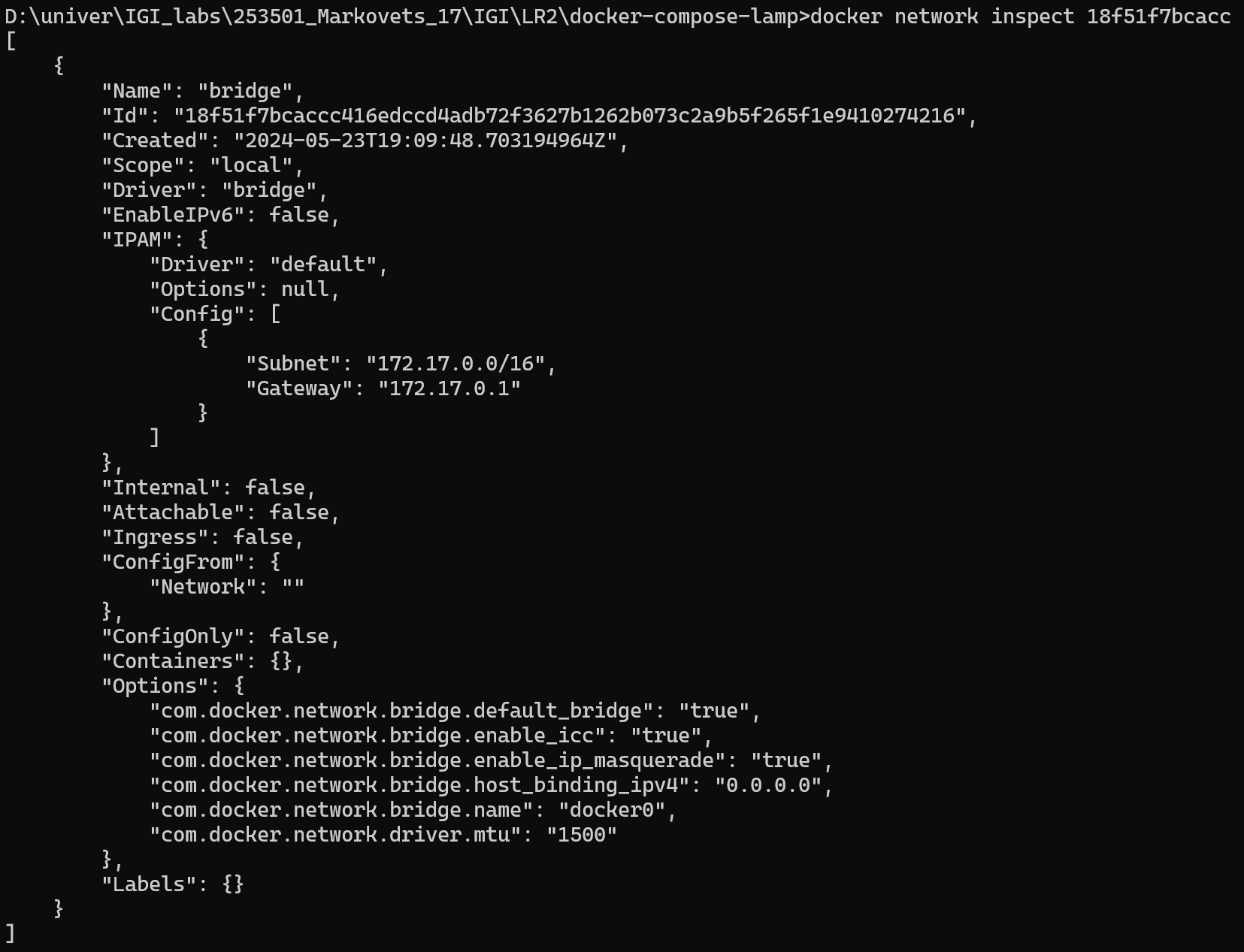
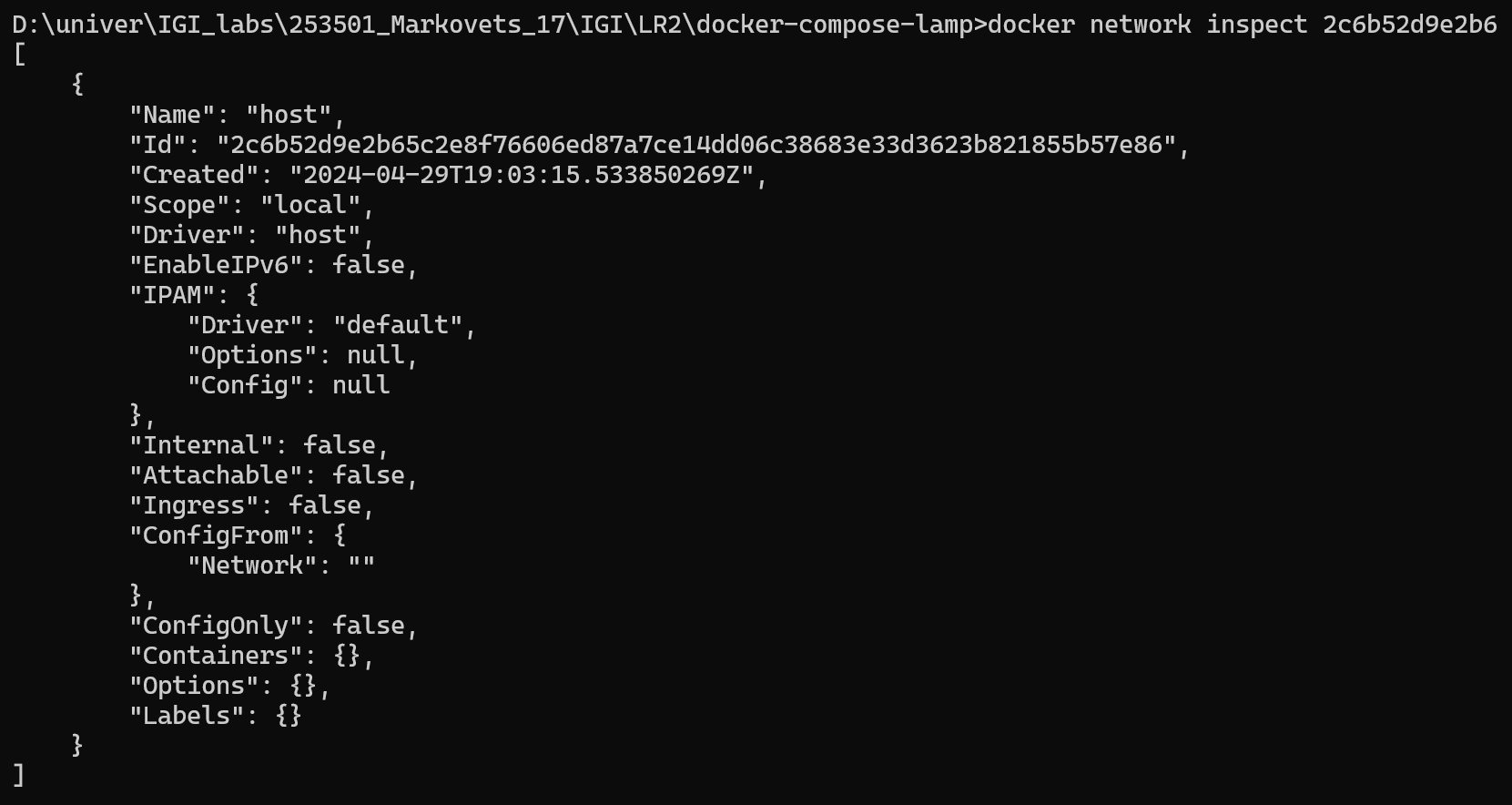
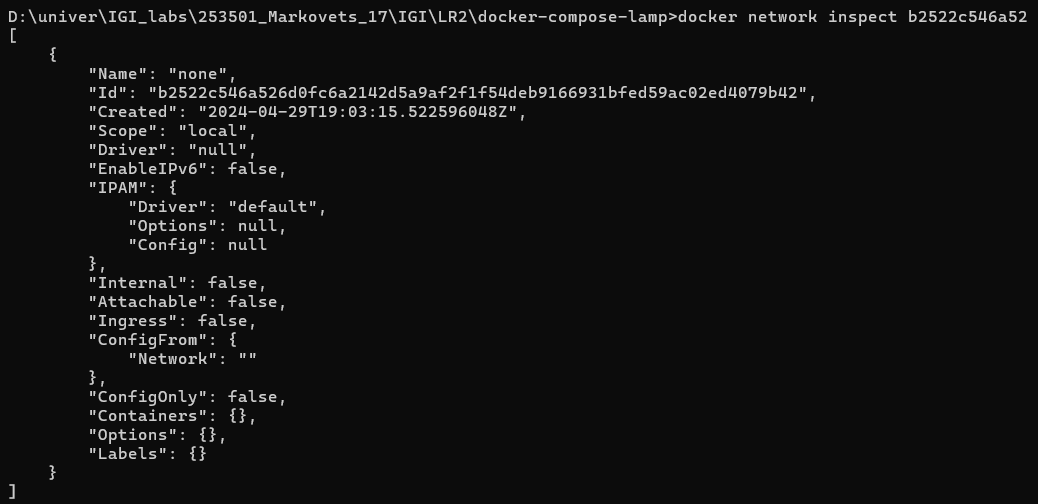
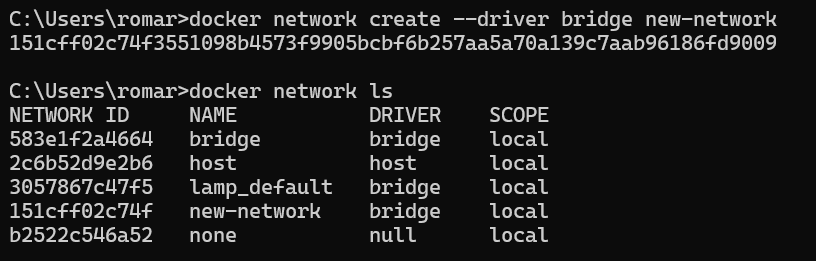
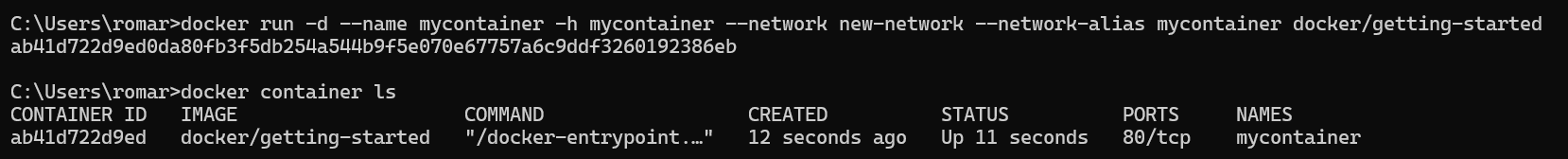
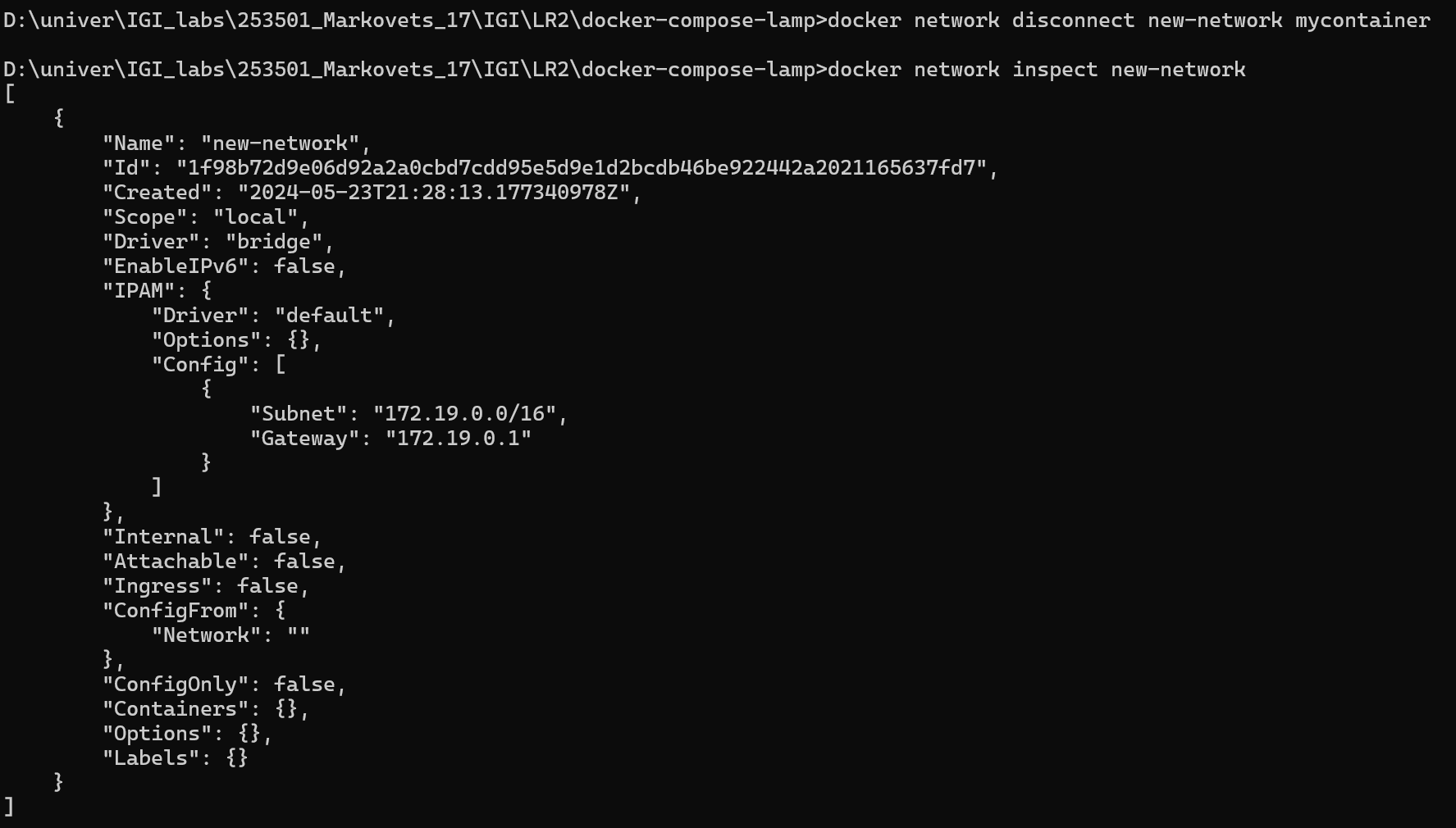
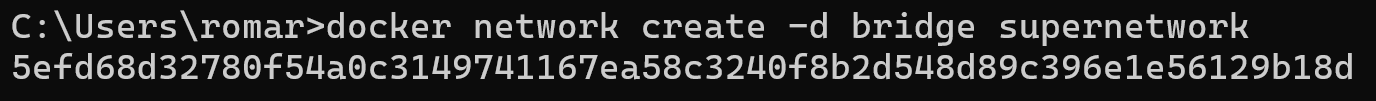
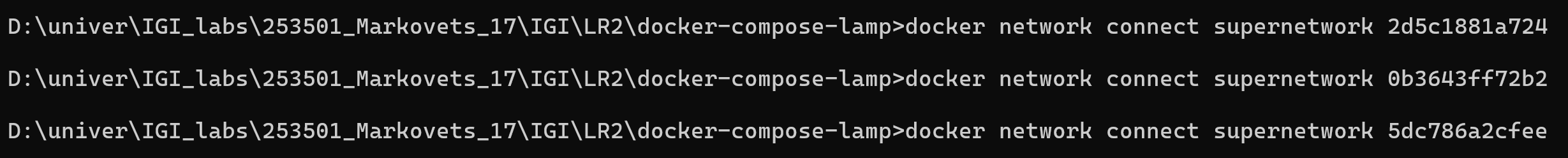
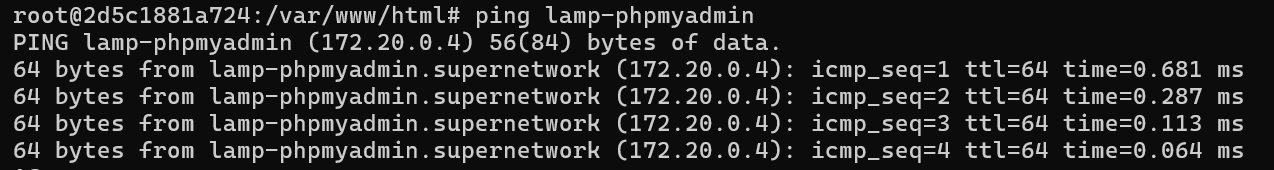
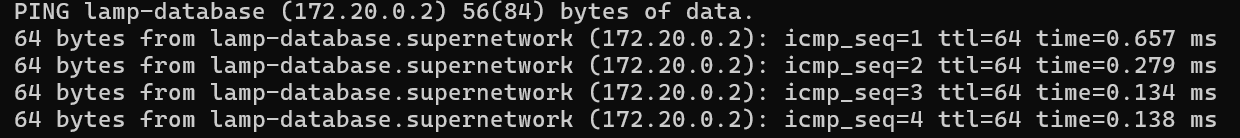
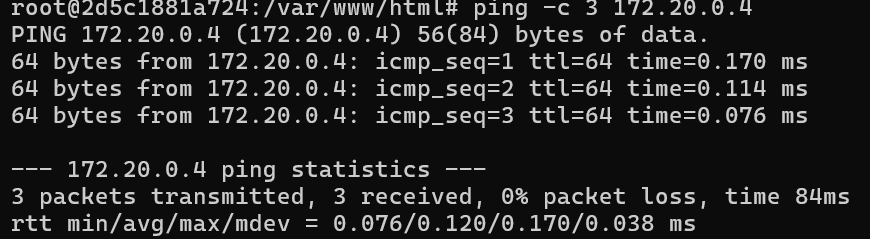
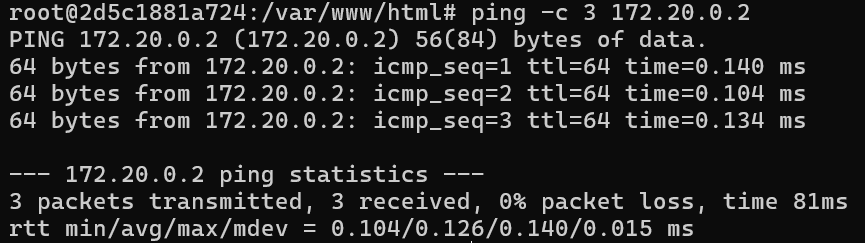
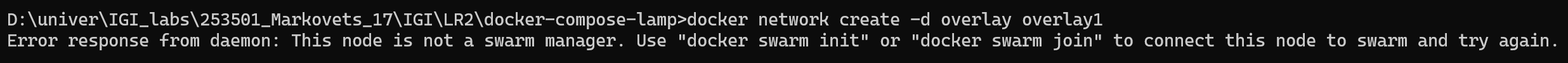
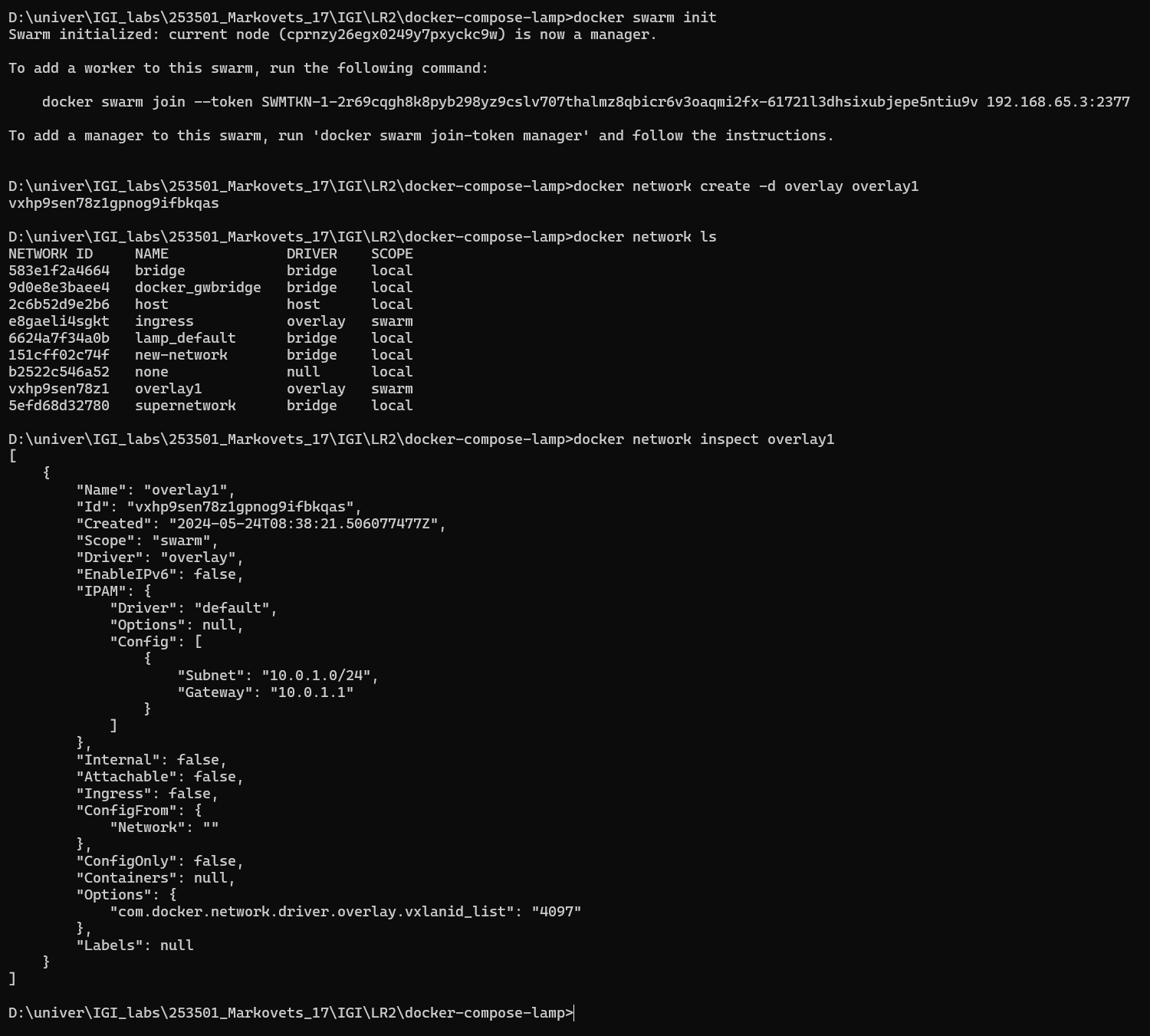
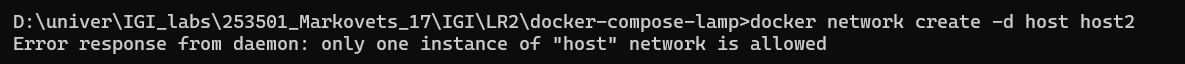
* 1. Использовать его для создания контейнера. Протестировать использование контейнера  
     создан образ geometric-lib  
       
       
     

1. Скачать любой доступный проект с GitHub с произвольным стеком технологий (пример – см. индивидуальное задание) 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)   
   



1. Настроить сети и тома для обеспечения связи между контейнерами и сохранения данных (исходные данные, логин, пароль и т.д.)  
     
   
2. Разместите результат в созданный репозиторий в DockerHub   
     
     
     
     
   
3. 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  
    
    
    
    
  подключимся к контейнеру lamp-php74  
  пропингуем остальные контейнеры  
    
    
  пинг по ip  
    
  
* Создать свою собственную сеть overlay, проверить, создана ли она, вывести о ней всю информацию  
  попытка создать сеть overlay  
    
  
* Создать еще одну сеть overlay, проверить, создана ли она, вывести о ней всю информацию, удалить сеть  
  
* Попробовать создать сеть host, сохранить результат  в отчет.  
    
  (сеть host может быть только одна)

**Индивидуальное задание**

|  |  |
| --- | --- |
| **Вар-т** | **Условие – стэк технологий** |
|  | ReactJs-Django-SQLite |
|  | ReactJS-Django-MySQL |
|  | PHP, Django REST framework, MySQL and React |
|  | Django 5, React, Bootstrap 5 with Python 3, PostgreSQL |
|  | React, Node , PostgreSQL |
|  | React.js - Redux - Node.js - Mongodb - Webpack |
|  | Mongo DB, Express, Angular, Node.js |
|  | Linux, Apache, MySQL, PHP |
|  | MongoDB, Express.js, AngularJS, Node.js |
|  | Ruby, Rails, Rack, Passenger |
|  | .NET, MS SQL Server |
|  | Java, MySQL |
|  | ASP.NET MVC, IIS, Microsoft Azure, SQL Server |
|  | React.js - Redux - Node.js - Mongodb |
|  | ReactJs-Django-SQLite |
|  | ReactJS-Django-MySQL |
|  | PHP, Django REST framework, MySQL and React |
|  | Django 5, React, Bootstrap 5 with Python 3, PostgreSQL |
|  | React, Node , PostgreSQL |
|  | React.js - Redux - Node.js - Mongodb - Webpack |
|  | Mongo DB, Express, Angular, Node.js |
|  | Linux, Apache, MySQL, PHP |
|  | MongoDB, Express.js, AngularJS, Node.js |
|  | Ruby, Rails, Rack, Passenger |
|  | .NET, MS SQL Server |
|  | Java, MySQL |
|  | ASP.NET MVC, IIS, Microsoft Azure, SQL Server |
|  | React.js - Redux - Node.js - Mongodb |
|  | Linux, Apache, MySQL, PHP |
|  | MongoDB, Express.js, AngularJS, Node.js |

**Контрольные вопросы**

1. Что представляет собой контейнеризация?
2. Что такое Docker?
3. Что является реестром Docker?
4. Что такое Docker Engine?
5. Как происходит связь между клиентом Docker и демоном Docker?
6. Для чего необходим образ контейнера?
7. Что представляет собой контейнер Docker?
8. Каков метод создания Docker-контейнера?
9. Каковы этапы жизненного цикла контейнера Docker?
10. В каких средах можно использовать контейнерные приложения? Что такое виртуальная машина?
11. Является ли Docker VM-технологией?
12. Какие различия между виртуальными машинами и контейнерами Docker?
13. Зачем использовать контейнеры?
14. Что выдаст команда $ docker ps ?
15. Сколько идентификаторов по умолчанию имеет каждый контейнер?
16. Как задается имя контейнера?
17. Для чего нужна команда container start?
18. Как создать контейнер без запуска?
19. Что такое запуск контейнеров в интерактивном режиме?
20. Что представляет собой образ?
21. Что такое Dockerfile?
22. Для чего необходим Docker Compose?
23. Какая структура файла Compose?
24. Как запустить несколько контейнеров, используя один сервис?
25. Что такое предоставление порта связанным службам?
26. Для чего используются тома?
27. Для чего применяют переменные среды?
28. Как определяют правила связи между контейнерами? Написать файл Docker для создания и копирования каталога и построить его с использованием модулей Python?
29. Какие сети доступны по умолчанию в Docker?
30. Приведите необходимые шаги для развертывания докеризированного приложения, сохраненного в репозитории Git