МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

МАТЕРИАЛЫ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

Технологии хранения в системах кибербезопасности

(наименование дисциплины (модуля) в соответствии с учебным планом)

Уровень	специалитет
	(бакалавриат, магистратура, специалитет)
Форма обучения	очная
	(очная, очно-заочная, заочная)
Направление(-я) подготовки	10.05.04 Информационно-аналитические системы безопасности
	(код(-ы) и наименование(-я))
Институт	Кибербезопасности и цифровых технологий (ИКБ)
	(полное и краткое наименование)
Кафедра	КБ-2 «Информационно-аналитические системы кибербезопасности»
	(полное и краткое наименование кафедры, реализующей дисциплину (модуль))
Лектор	к.т.н., Селин Андрей Александрович, Бугаев Александр Александрович
	(сокращенно – ученая степень, ученое звание; полностью – ФИО)

Используются в данной редакции с учебного года 2024/2025

(учебный год цифрами)

Проверено и согласовано «____» _____2024 г. _____ А.А. Бакаев

(подпись директора Института/Филиала с расшифровкой)

Москва 2024 г.

ПРАКТИЧЕСКАЯ РАБОТА № 7

«Знакомство с объектным хранилищем MinIO и графовой СУБД Neo4j»

Цель работы – получение практических навыков развертывания и использования объектного хранилища MinIO и графовой СУБД Neo4j.

Задание:

1. Запустите Unix-подобную систему (например, Debian 12.6.0 64-bit[1]).

2. Создайте пользователя с именем формата **fio_nn**,
где f – первая буква фамилии на латинице;
i – первая буква имени на латинице;
o – первая буква отчества на латинице (при наличии),
nn – двузначный номер по списку в группе.

Добавьте его в группу sudo. **Все дальнейшие действия необходимо выполнять от имени созданного пользователя.**

3. Запустите терминал и установите Docker и Docker Compose.

4. Создайте каталог для нового проекта и сформируйте файл **docker-compose.yml** для развертывания **MinIO** (https://hub.docker.com/r/minio/minio) в режиме Single-Node Single-Drive (**1 сервер, 1 диск**).

Требования к запускаемым сервисам:

– последние 2 цифры номера порта, на котором будет развернут сервис, должны соответствовать номеру по списку в группе (например, для 15 – 12315, 8015, 9915 и т.п.);

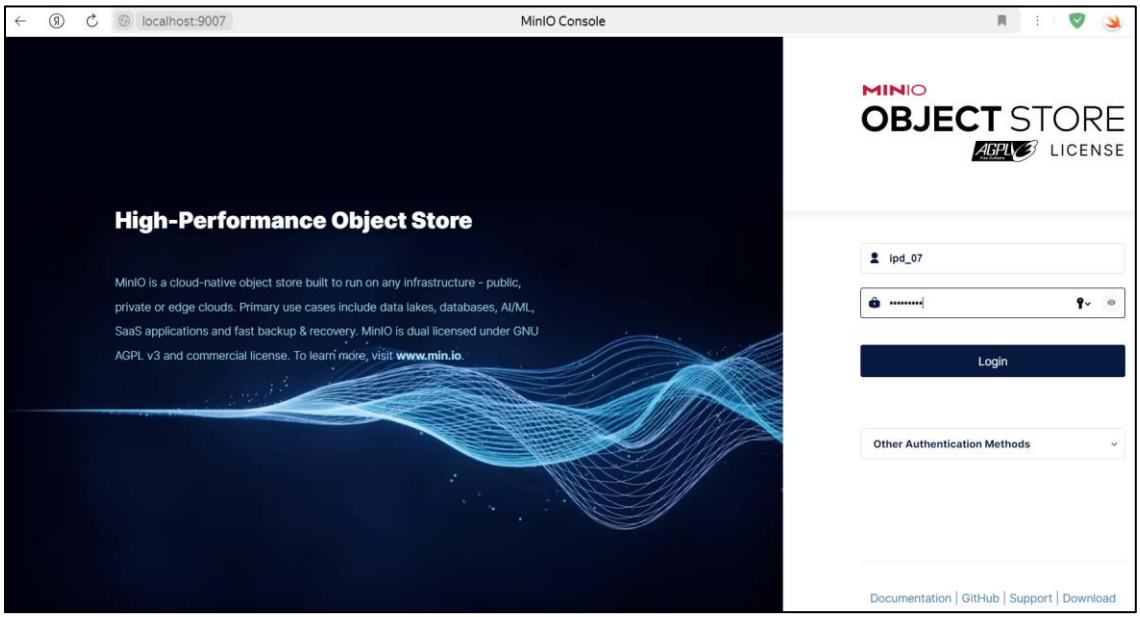
– имя контейнера должно заканчиваться на символ подчеркивания и инициалы ФИО (например, для Иванова Петра Дмитриевича – minio_ipd).

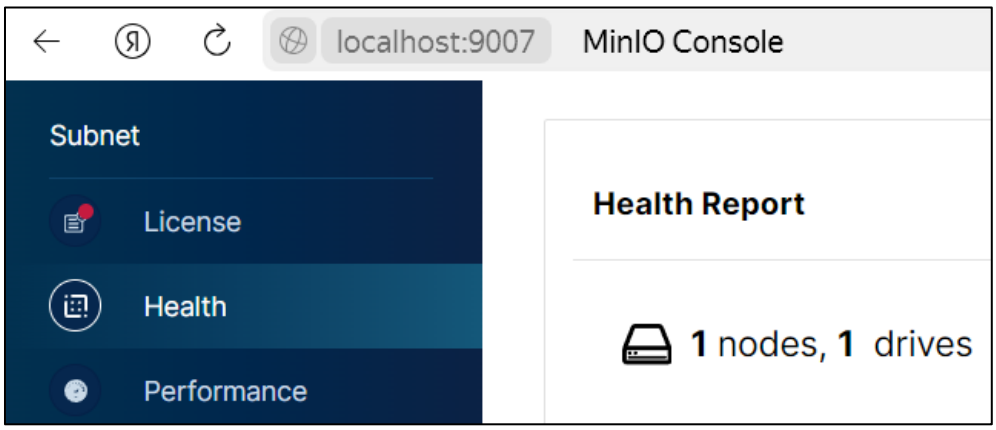
[1] Можно скачать готовый образ виртуальной машины по ссылке https://sourceforge.net/projects/osboxes/files/v/vb/14-D-b/12.6.0/64bit.7z/download

Пример файла docker-compose.yml:

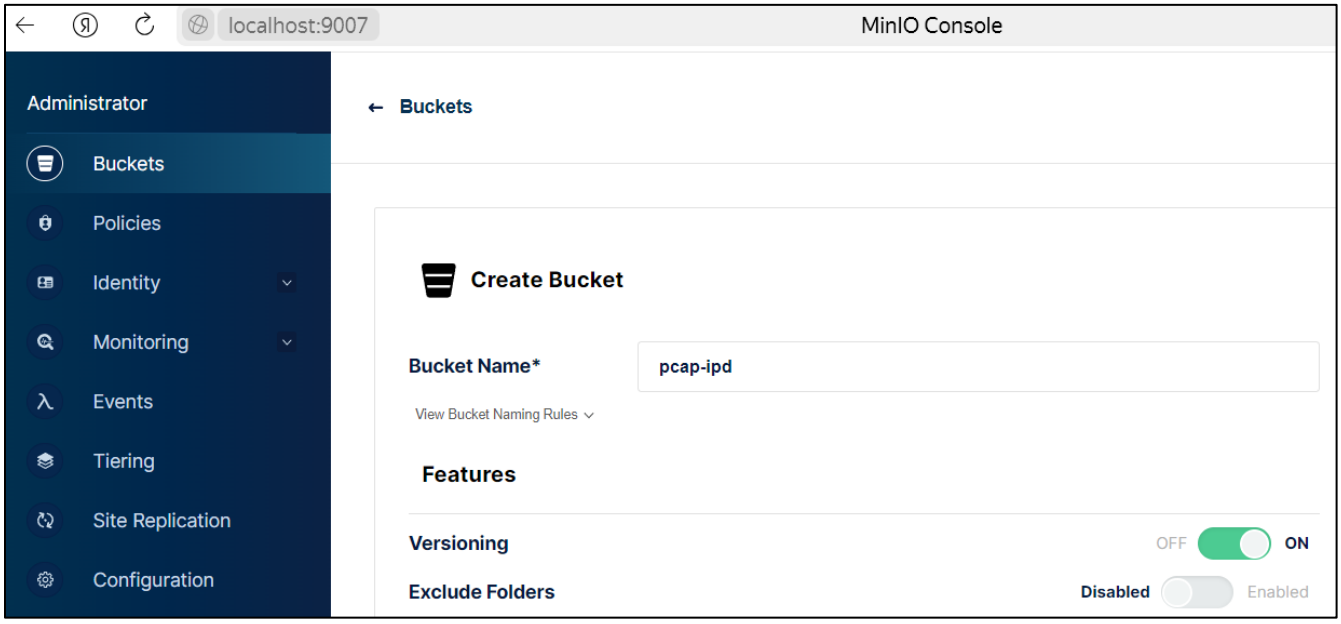
```yaml
services:
  minio:
    container_name: minio_ipd
    restart: always
    image: minio/minio:latest
    ports:
      - 7007:9000
      - 9007:9090
    environment:
      MINIO_ROOT_USER: ipd_07
      MINIO_ROOT_PASSWORD: qwerty123
    command: server /data --console-address ":9090"
    volumes:
      - ./data:/data
      - ./config:/root/.minio
```

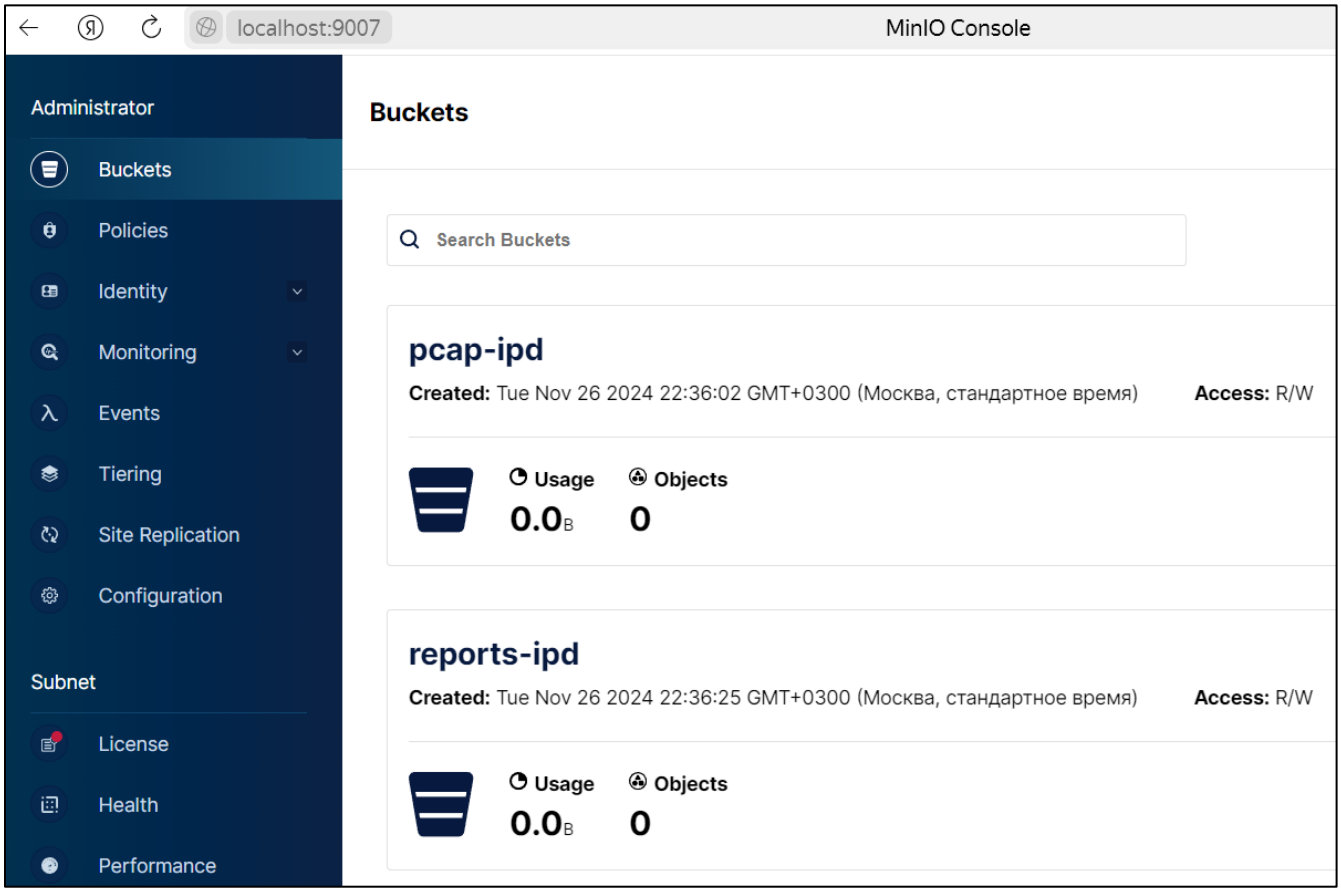
5. Разверните MinIO с помощью Docker Compose.



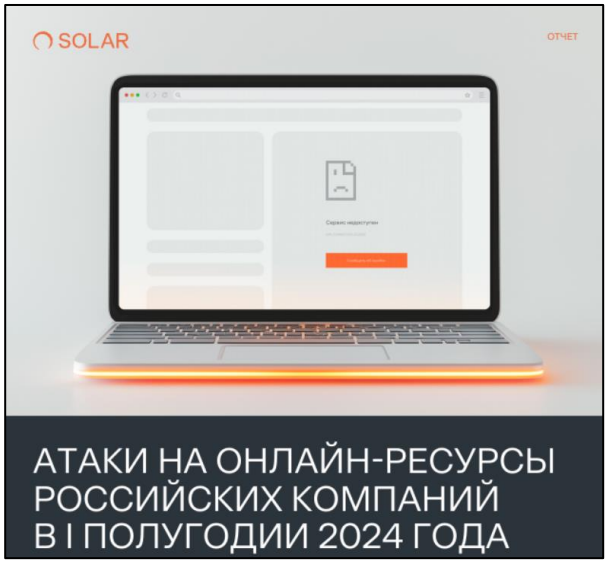


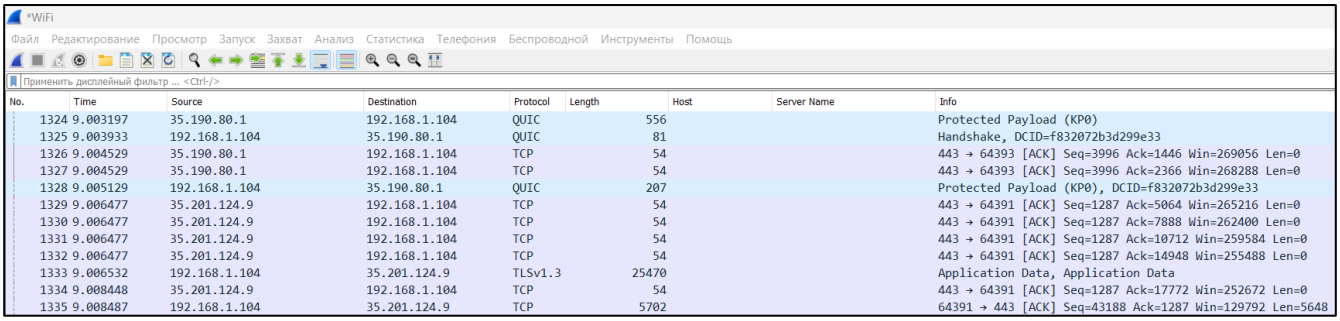
6. Изучите веб-консоль MinIO (https://min.io/docs/minio/container/index.html):
6.1. Добавьте разделы (buckets). **Названия разделов должны заканчиваться на дефис и инициалы ФИО** (например, для Иванова Петра Дмитриевича – pcap-ipd, reports-ipd, и т.п.).

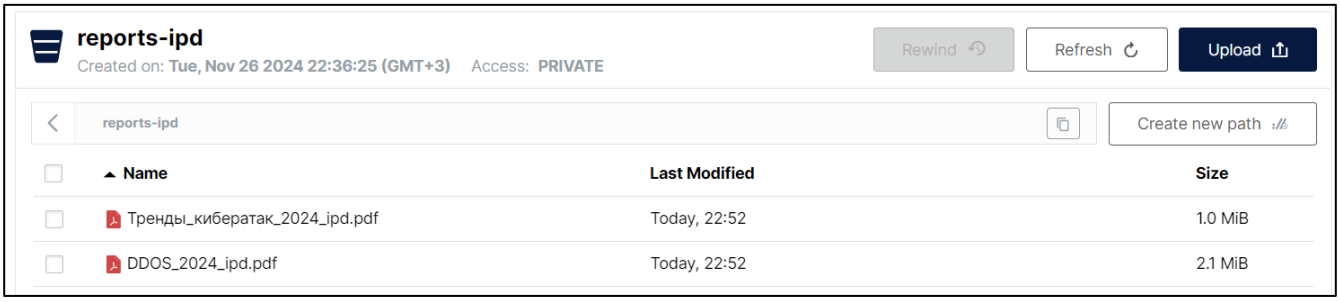


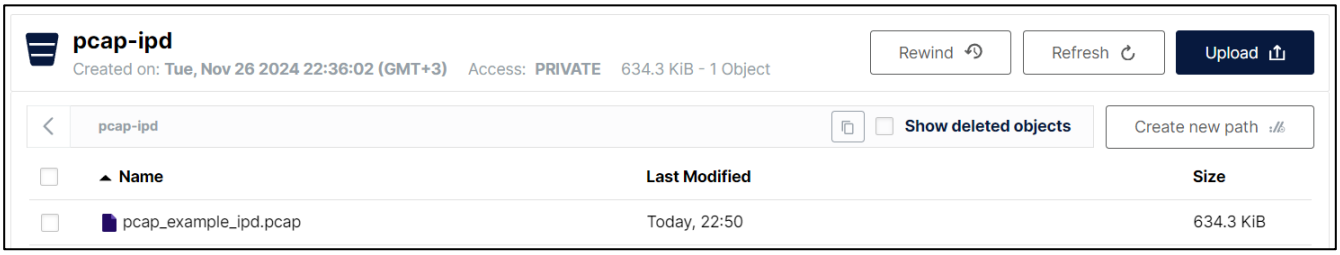


Пример загрузки файлов:

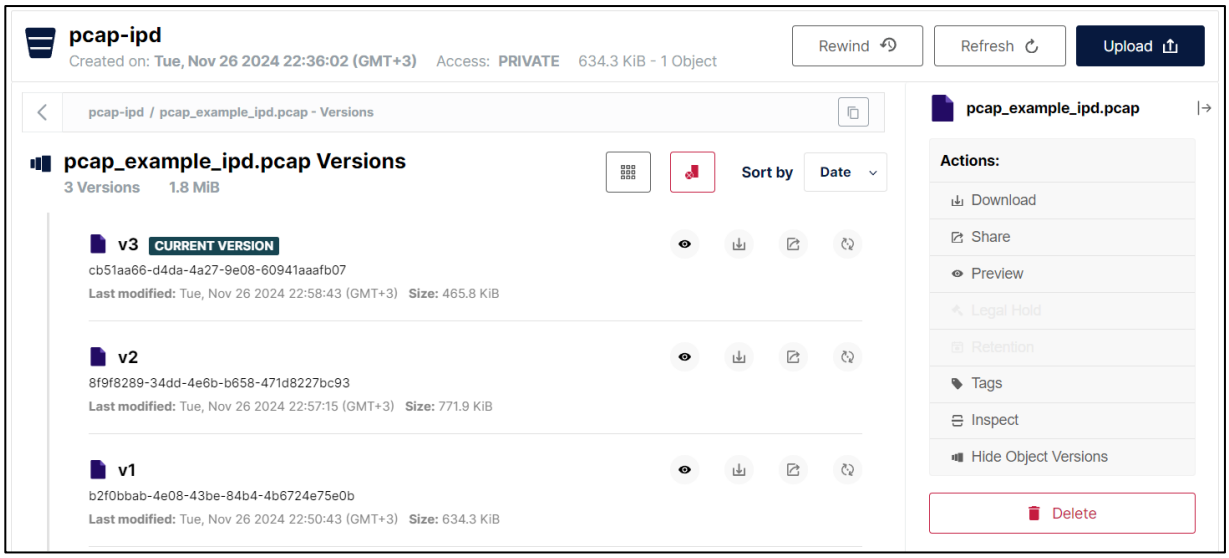








6.2. Изучите, как работает **версионирование**. Для этого загрузите несколько различных версий файла с одинаковым названием в один и тот же раздел (в разделе должно быть включено версионирование).



6.3. Добавьте пользователей с различными правами. Проанализируйте, как для них работает веб-консоль.

7. Установите MinIO Client (https://min.io/docs/minio/linux/reference/minio-mc.html). Пример установки и проверки работоспособности:

**curl https://dl.min.io/client/mc/release/linux-amd64/mc **
** --create-dirs **
** -o $HOME/minio-binaries/mc**

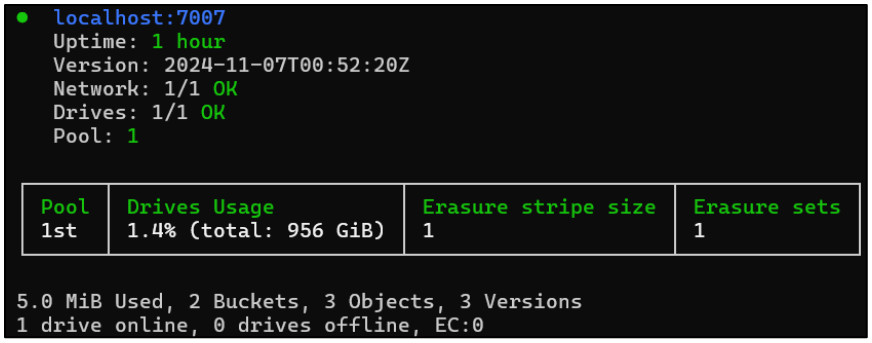
chmod +x $HOME/minio-binaries/mc

export PATH=$PATH:$HOME/minio-binaries/

mc --help

8. Создайте алиас (alias) в mc (авторизуйтесь). Пример:

mc alias set minio http://localhost:7007 ipd_07 qwerty123
mc admin info minio



9. Изучите команды MinIO Client (https://hub.docker.com/r/minio/mc/). Попрактикуйтесь в выполнении основных:

- создание, удаление разделов;
- добавление, обновление, удаление файлов;
- поиск объектов;
- отображение содержимого файлов;
- другие команды (не менее 2).

10. Напишите скрипт/программу для работы с MinIO. Минимальный функционал:

- создание и удаление разделов;
- загрузка файлов в раздел;
- скачивание файлов;
- удаление файлов;
- просмотр списка файлов в разделе.

Например, можно использовать библиотеку **minio** для Python.
Продемонстрируйте работу всех функций. **Названия разделов должны заканчиваться на инициалы ФИО** (например, files_ipd), а **названия файлов – начинаться на них** (ipd_01, ipd_02, …).

11. Разверните MinIO в режиме Multi-Node Multi-Drive (не менее **2 серверов и 4 дисков**) и проверьте его работоспособность.
Пример файла docker-compose.yml:

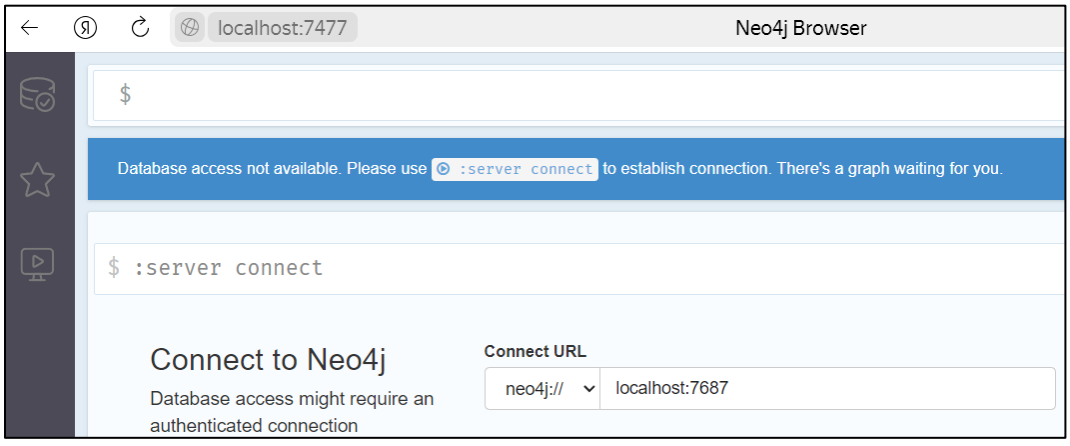
```yaml
# Общие настройки для контейнеров MinIO
x-minio-common: &minio-common
  image: minio/minio:latest
  command: server --console-address ":9090" http://minio{1...2}/data{1...2}
  expose:
    - "9000"
    - "9090"
  environment:
    MINIO_ROOT_USER: ipd_07
    MINIO_ROOT_PASSWORD: qwerty123
  healthcheck:
    test: ["CMD", "curl", "-f", "http://localhost:9000/minio/health/live"]
    interval: 30s
    timeout: 20s
    retries: 5

services:
  minio1:
    <<: *minio-common
    ports:
      - "9107:9000"
      - "9197:9090"
    volumes:
      - data1-1:/data1
      - data1-2:/data2

  minio2:
    <<: *minio-common
    ports:
      - "9207:9000"
      - "9297:9090"
    volumes:
      - data2-1:/data1
      - data2-2:/data2

volumes:
  data1-1:
  data1-2:
  data2-1:
  data2-2:
```

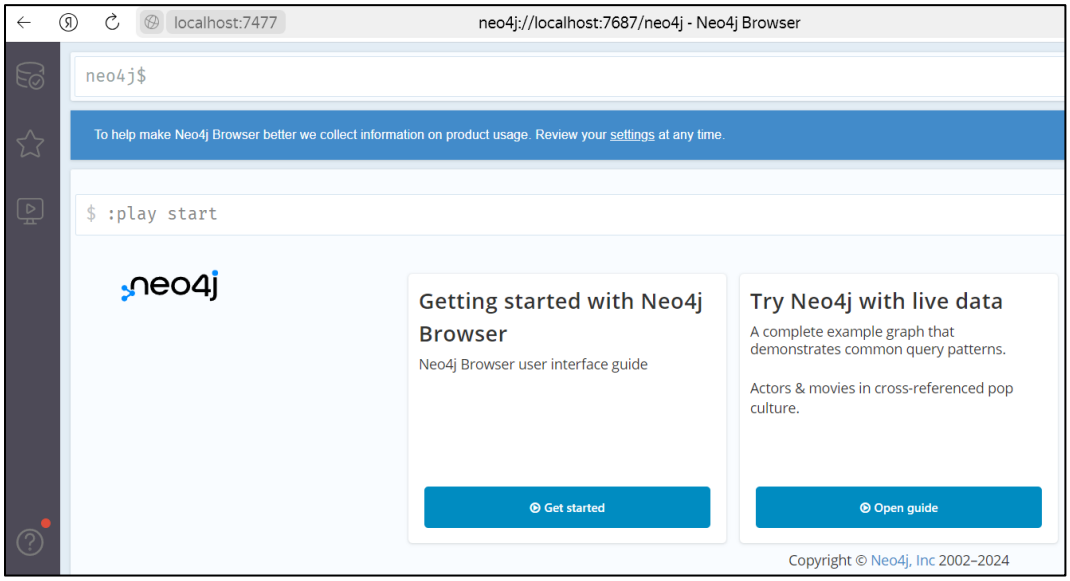
Дополнительно: разверните MinIO в режиме Multi-Node Multi-Drive (не менее 2 серверов и 4 дисков) с использованием прокси-сервера Nginx для балансировки нагрузки на серверы MinIO и проверьте его работоспособность.

12. Создайте каталог для проекта и сформируйте файл **docker-compose.yml** для развертывания **Neo4j** (https://hub.docker.com/_/neo4j).
Пример файла docker-compose.yml:

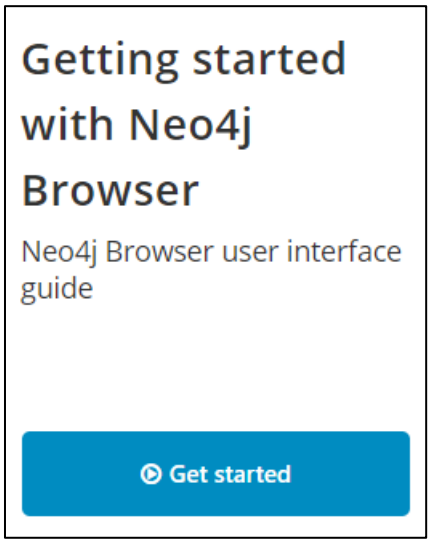
```yaml
services:
  neo4j:
    image: neo4j:latest
    container_name: neo4j_ipd
    restart: always
    ports:
      - 7477:7474
      - 7687:7687
    environment:
      NEO4J_AUTH: none
      NEO4J_dbms_directories_import: /import
    volumes:
      - ./data:/data
      - ./import:/import
```

13. Разверните Neo4j с помощью Docker Compose.

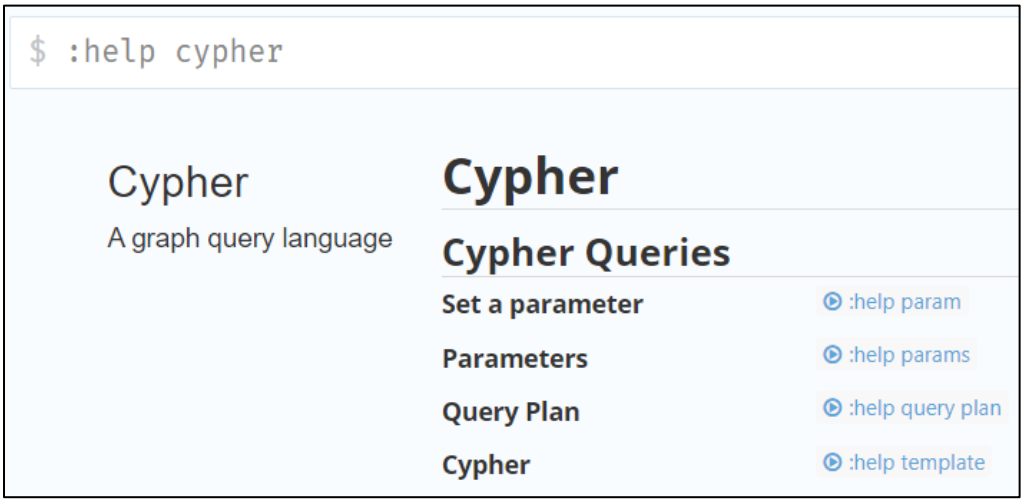




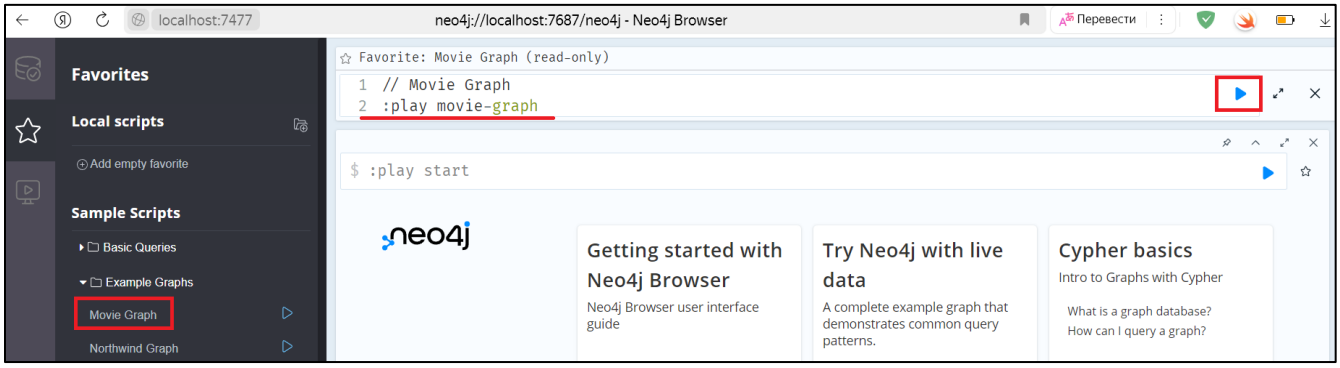
14. Изучите функционал пользовательского интерфейса.

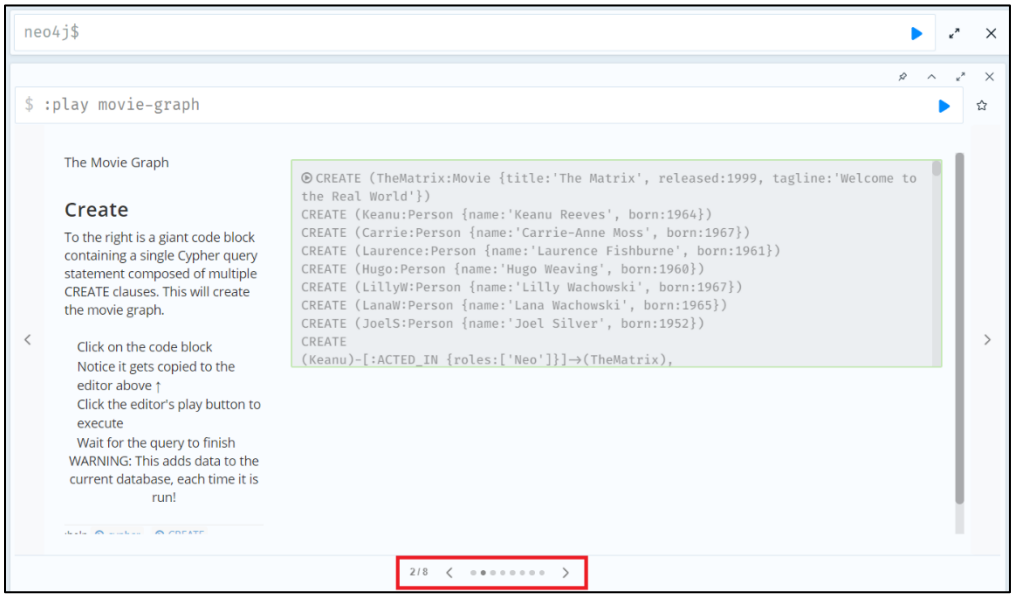


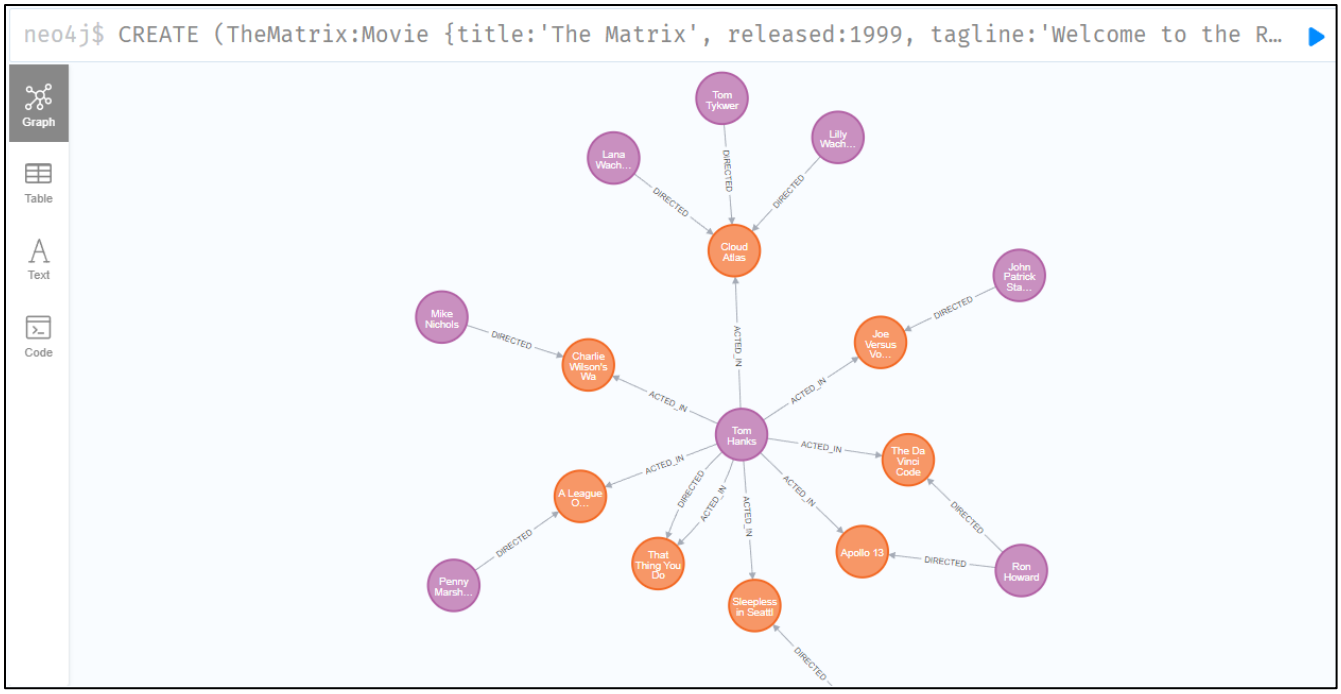
15. Изучите синтаксис написания запросов Cypher Query Language с использованием команды «**:help cypher**».



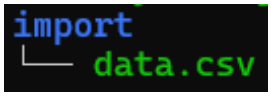
16. Изучите основные операции и возможности в Neo4j на примере графа «Movie Graph», выполняя последовательные действия в интерактивном режиме.







17. Сформируйте тестовый набор данных в формате CSV для импорта и создания графа в Neo4j (например, на сайте https://sqldatagenerator.com/generator). Поместите файл в каталог **import**.




```
"id","firstName","lastName","phone","email","companyName"
"1","Sammy","Auer","688-937","Rhea.Wehner@gmail.com","Company_1"
"2","Jayden","Padberg","708-760","Kraig_Ruecker@hotmail.com"," Company_2"
"3","Eryn","Rau","503-796","Tyreek83@hotmail.com"," Company_3"
"4","Jett","Gutmann","682-250","Sonia_Cummings@yahoo.com","Company_1"
"5","Dorthy","Von","479-969","Nathaniel12@yahoo.com"," Company_2"
"6","Nola","Aufderhar","130-851","Pierce.Dickinson71@gmail.com"," Company_3"
"7","Greyson","Thiel","120-983","Avery_Heaney@gmail.com","Company_1"
```

18. Импортируйте данные в Neo4j. Пример:

```
1 LOAD CSV WITH HEADERS FROM "file:///data.csv" as row
2 MERGE (n:Person_ipd {id: row.id, firstName: row.firstName, lastName: row.lastName,
  companyName: row.companyName, phone: row.phone, email: row.email} )
```

Table

Added 499 labels, created 499 nodes, set 2994 properties, completed after 216 ms.

```
1 LOAD CSV WITH HEADERS FROM "file:///data.csv" as row
2 MERGE (c:Company_ipd {name: row.companyName} )
```

Table

Added 3 labels, created 3 nodes, set 3 properties, completed after 50 ms.

19. Создайте связи между двумя сущностями.

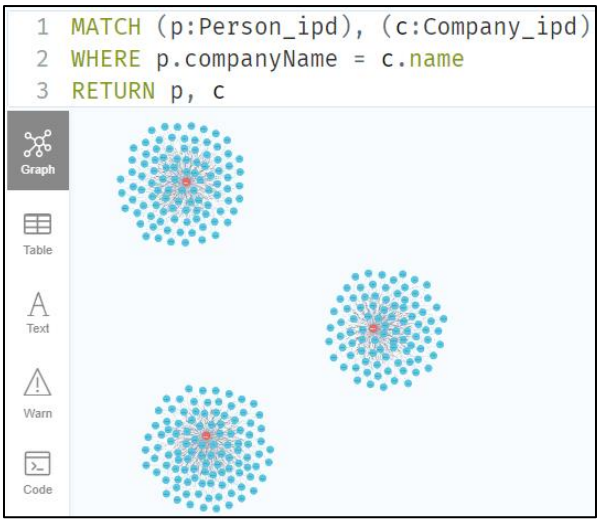
```
1 MATCH (p:Person_ipd), (c:Company_ipd)
2 WHERE p.companyName = c.name
3 CREATE (p)-[:WORK_IN]->(c)
```

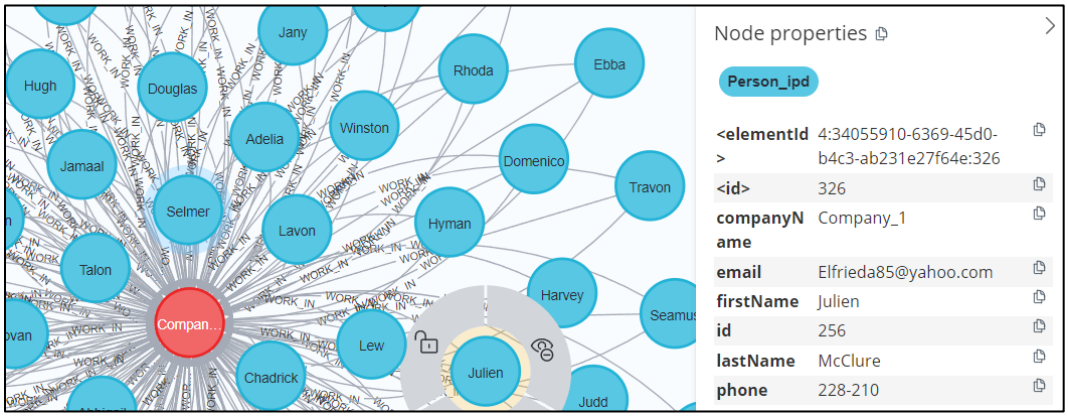
Table

Created 499 relationships, completed after 27 ms.

20. Проанализируйте полученный граф.

```
1 MATCH (p:Person_ipd), (c:Company_ipd)
2 WHERE p.companyName = c.name
3 RETURN p, c
```





21. Выполните в созданном графе операции:
   - поиска, добавления, удаления узлов;
   - добавления, удаления свойств.

22. Добавьте новые сущности и атрибуты в ваш граф. Сформируйте новые связи. Проанализируйте полученные группы объектов.