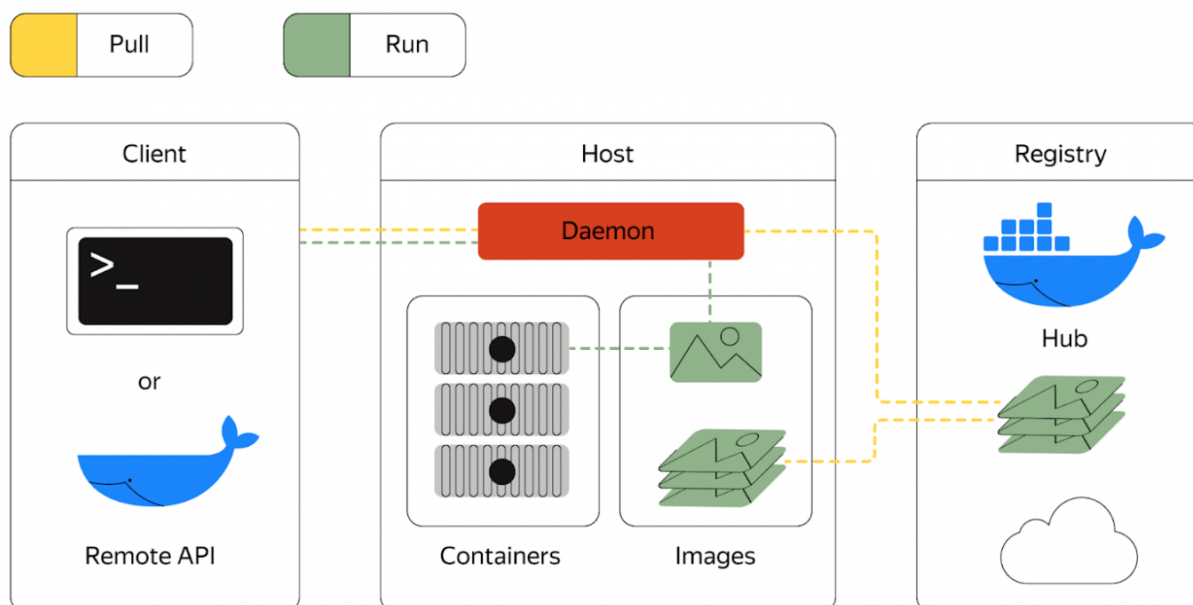


Docker Architecture



Основным элементом являются Docker-**образы (Images)** — это заранее подготовленные шаблоны, которые включают в себя все необходимое для корректной и быстрой работы приложения, включая библиотеки, сам код и все зависимости, а также произведенные настройки. Эти образы используются для создания контейнеров — изолированных исполняемых единиц, в которых запускается приложение.

Реестр Docker (Docker **Registry**) представляет собой удалённую платформу, используемую для хранения образов Docker. В ходе работы с Docker образы отправляют в реестр и загружают из него. Подобный реестр может быть организован тем, кто пользуется Docker. Кроме того, поставщики облачных услуг могут поддерживать и собственные реестры. Например, это касается AWS и Google Cloud.

Хаб Docker (Docker **Hub**) — это самый крупный реестр образов Docker. Кроме того, именно этот реестр используется при работе с Docker по умолчанию. Пользоваться хабом Docker можно бесплатно.

Контейнеры Docker по своей природе неизменяемы. Это означает, что при перезапуске контейнера все данные, сохраненные в нем, **сотрутся**. Но сохранить эти данные можно. Для этого Docker предоставляет возможность подключения к контейнеру локальных директорий.

b4d699d2189c -----> id контейнера

`docker exec -it b4d699d2189c bash` -----> команда для входа в контейнер

`docker start b4d699d2189c` -----> Запуск суц-ся контейнера

`docker ps` -----> Посмотреть список запущенных контейнеров

`docker ps -a` -----> Посмотреть список всех контейнеров

.

.

```
root@nashimudes:~# docker cp /media/nashimudes/New\ Volume/Downloads/demo-small/demo-small-20170815.sql test_postgres1:/
```

-----> Копируем файл demo-small-20170815.sql в контейнер test_postgres1

.

.

```
root@nashimudes:~# docker run --name small_postgres -p 5411:5432 -e POSTGRES_PASSWORD=1234 -d postgres
```

-----> Создание контейнера с именем small_postgres с внешним портом 5411 и внутренним 5432 с паролем 1234 образ которого является postgres

.

.

```
root@nashimudes:~# docker cp small_postgres: "/backup/mydb.sql" "/home/nashimudes/Desktop"
```

-----> Копируем файл mydb.sql из контейнера в локальную машину

.

.

```
root@283018d372df: /# pg_dump -U postgres demo > /backup/mydb.sql
```

-----> Создание backup-а базы demo в папку /backup под именем postgres.

283018d372df----> контейнер внутри которого мы находимся

.

.

```
root@283018d372df: /test# pg_dump -d demo -U postgres > /test/demo_back.sql
```

-----> -d demo указываем, что нужно скопировать бд demo

.

.

```
version: '2'
```

```
services:
```

```
  db:
```

```
image: postgres
container_name: test_cont1
ports:
  - 5467:5432
environment:
  POSTGRES_PASSWORD: 1234
  POSTGRES_USER: postgres
  POSTGRES_DB: postgres
volumes:
  - /home/nashimudes/Desktop/test_cont1:/var/lib/postgresql/data
```

----> Create compose.yml

.

.

```
version: '2'
```

```
services:
  postgresql_01:
    image: postgres
    container_name: postgres_master
    restart: always
    networks:
      - app-network
    ports:
      - 5411:5432
    environment:
      POSTGRES_PASSWORD: 1234
      POSTGRES_USER: postgres
      POSTGRES_DB: postgres
    volumes:
      - /home/nashimudes/Desktop/test_cont1:/var/lib/postgresql/data

  postgresql_02:
    image: postgres
    container_name: postgres_slave
    restart: always
    networks:
      - app-network
    ports:
      - 5422:5432
    environment:
      POSTGRES_PASSWORD: 1234
      POSTGRES_USER: postgres
```

```
POSTGRES_DB: postgres
volumes:
  - /home/nashimudes/Desktop/test_cont2:/var/lib/postgresql/data

networks:
  app-network:
    driver: bridge
```

```
sudo docker run --name test_postgres1 -p 5455:5432 -v
/home/nashimudes/Desktop/test:/var/lib/postgresql/data -e POSTGRES_PASSWORD=1234 postgres
```

```
root@283018d372df:/# psql -U postgres < demo-small-20170815.sql
```

-----> Распаковываем файл demo-small-20179815.sql внутрь бд под пользователем postgres

```
pg_config --version ---> проверка версии postgresql
```

```
linux
root
abdu0102A
```

```
sudo -u postgres psql template1
```

-----> Эта команда подключится к PostgreSQL базе данных template1 как пользователь postgres.

```
test_db1=# ALTER GROUP readonly_users ADD USER test_user1, test_user2;
```

----> Добавить поль-лей test_user1, test_user2 в группу readonly_users.

.
.
.
.
.

```
test_db1=# \du
```

----> Посмотреть список пользователей

.
.

`CTRL + SHIFT + L` ----> clear psql terminal

.
.

```
test_db1=# SELECT username FROM pg_catalog.pg_user;
```

----> Посмотреть список пользователей

.
.

```
test_db1=# ALTER GROUP readonly_users ADD USER test_user1, test_user2;
```

----> Добавляем в группу readonly_users польз-лей test_user1, test_user2

.
.

```
SELECT r.rolname as username, r1.rolname as "role"
FROM pg_catalog.pg_roles r JOIN pg_catalog.pg_auth_members m
ON (m.member = r.oid)
JOIN pg_roles r1 ON (m.roleid=r1.oid)
WHERE r.rolcanlogin
ORDER BY 1;
```

.
.

```
select * from current_user, session_user;
```

.
.

```
SELECT current_user, session_user;
```

----> Посмотреть текущего поль-ля и кто создал эту сессию

.

```
.  
  
SELECT * FROM information_schema.role_table_grants WHERE grantee =  
'readonly_user';
```

-----> Посмотреть список привелегий пользователя
'readonly_user'

```
.  
.  
  
REVOKE SELECT ON bookings.boarding_passes, bookings.flights,  
bookings.airports_data FROM readonly_user;
```

-----> убрать привилегии у readonly_user

```
.  
.  
  
GRANT INSERT, DELETE ON bookings.tickets TO test_user1;
```

-----> Даём привилегий на добавление
и удаление в таблице tickets для поль-ля
test_user1.

```
.  
.  
  
demo=# CREATE USER nassim WITH PASSWORD '1234';
```

-----> Создание поль-ля с именем nassim и паролем 1234
demo-->название базы в которой мы находимся

```
.  
.  
  
psql -U test_user1 demo;
```

-----> подключится под пользователем test_user1 к бд demo

To clear the screen using the Ctrl + L keyboard shortcut, simply press Ctrl + L while you are in psql.

Sequences в PostgreSQL - генераторы уникальных последовательностей чисел, использовать можно аналогичным auto_increment полям образом в MySQL. Т.е. создаете sequence для таблицы, при вставке в нее новой записи в поле, которое должно быть уникальным (primary key) пишете значение, которое выдает sequence, созданный для этой таблицы.

Кластеризация, если говорить простыми словами, - это разбиение на группы, по определенным критериям. Кластер это группа объектов.

Попытаюсь объяснить на пальцах

В большинстве случаев основная нагрузка идет на чтение БД, часто бывает, что одна машина не справляется с существующей нагрузкой, тогда поднимают кластер — запускают СУБД на нескольких машинах, одна из них объявляется мастером, остальные репликами. Мастер занимается только записью и распространением готовых изменений по репликам. А читаем мы только из реплик, балансируя нагрузку между ними, тем самым снижая нагрузку на каждую из них и уменьшая время отклика.

схема в POSTGRES

Схема в PostgreSQL - это логическая структура, которая представляет собой набор таблиц, индексов, представлений, функций и других объектов базы данных, организованных в единую группу. Схема помогает организовать данные в базе данных, делая их более удобными для управления и обеспечивая логическую разделяемость объектов базы данных.

С помощью схемы можно создавать отдельные пространства имен для объектов базы данных, что позволяет избежать конфликтов имен и обеспечивает более удобную организацию базы данных. Каждая таблица, индекс или другой объект базы данных может принадлежать к определенной схеме, что позволяет структурировать базу данных и упрощает ее обслуживание.

Схемы также играют важную роль при управлении правами доступа к данным, так как позволяют назначать различные права на доступ к объектам базы данных в зависимости от их принадлежности к определенной схеме.

Таким образом, схема в PostgreSQL является важным средством организации данных и обеспечивает более эффективное управление базой данных.

База данных (БД) — это набор данных, который как-то структурирован. Например, можно взять сто картинок с котами и отсортировать их по цвету или по позе.

Система управления базами данных (СУБД) — это набор инструментов, которые позволяют удобно управлять базами данных: удалять, добавлять, фильтровать и находить элементы, менять их структуру и создавать резервные копии.

Популярные **СУБД** — PostgreSQL, MySQL, Microsoft SQL Server, SQLite, MongoDB, Redis, Oracle Database.

База данных — это набор элементов, которые сгруппированы по определённым правилам. Они бывают реляционными, графовыми и иерархическими.

СУБД — это инструменты, которые помогают управлять базами данных. Например, с их помощью можно удалять, изменять и находить элементы.

Базы данных отличаются от **СУБД** тем, что сами по себе представляют лишь файл на компьютере. Базы данных не умеют ничего делать с этими данными — только хранить. А вот СУБД уже предоставляют возможности по манипуляции ими.

Репликация базы данных — это процесс создания и обслуживания нескольких копий базы данных в разных местах или на разных серверах. Основная цель репликации — повысить доступность данных, распределить рабочую нагрузку и повысить общую производительность системы базы данных. Он также обеспечивает избыточность и отказоустойчивость, гарантируя, что данные реплицируются на несколько узлов синхронно или асинхронно.

1
 2
 3
 4
 5
 6
 7
 8
 9
 10
 11
 12

```
$ ср опции /путь/к/файлу/источнику /путь/к/директории/назначения
```

•

To get out of insert mode: [esc]

■

```
mv appendix apndx.a
```

•

```
root@nashimudes:/home/nashimudes# chown -Rv nashimudes:nashimudes ./Desktop/
```


chmod 777 foldername will give read, write, and execute permissions for everyone.



Для удаления папки и ее содержимого (рекурсивное удаление) воспользуйтесь командой rm с опцией -R:

```
rm -R название-папки
```

```
sudo apt-get update  
sudo apt-get install vim
```

-----> install vim for editing files