Лекция 6
Специализированные сервисы и хранилища

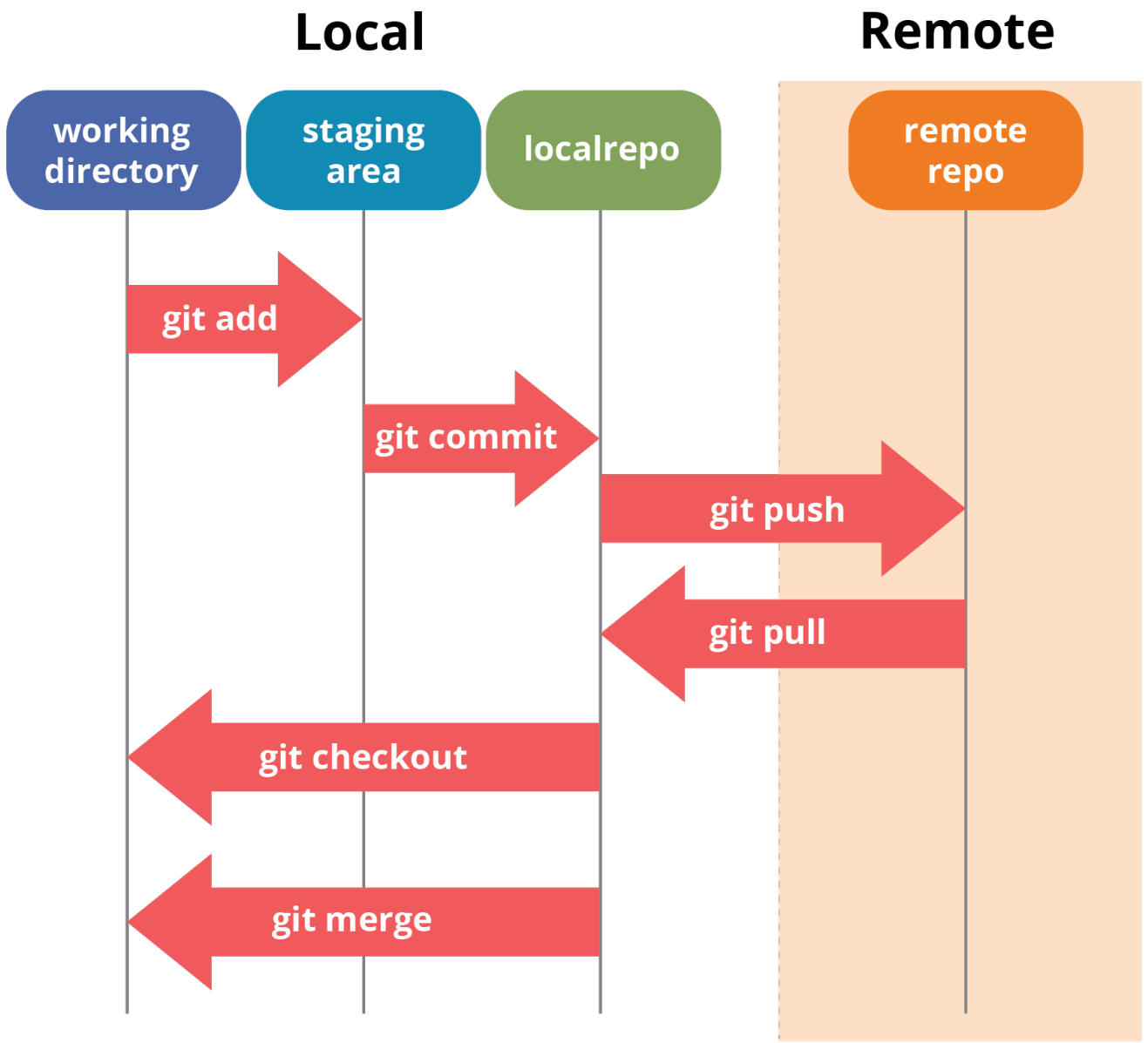
Разработка интернет приложений

Канев Антон Игоревич

Git

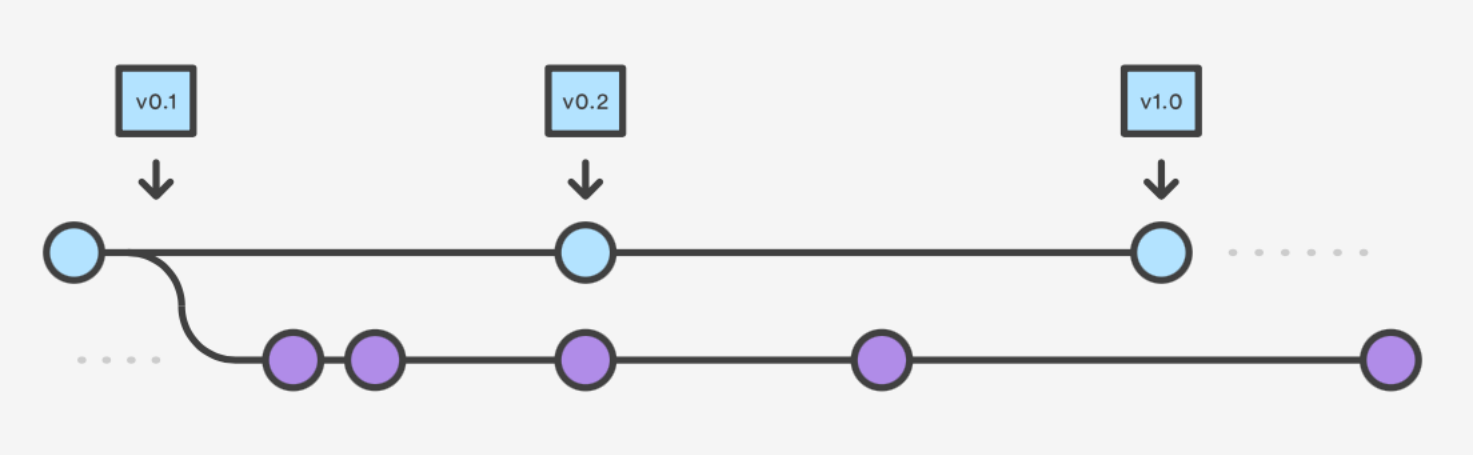
- Git – распределенная система управления версиями
- Позволяет хранить несколько версий одного и того же документа

Workflow


Local
Remote
working directory
staging area
localrepo
remote repo
git add
git commit
git push
git pull
git checkout
git merge

Develop

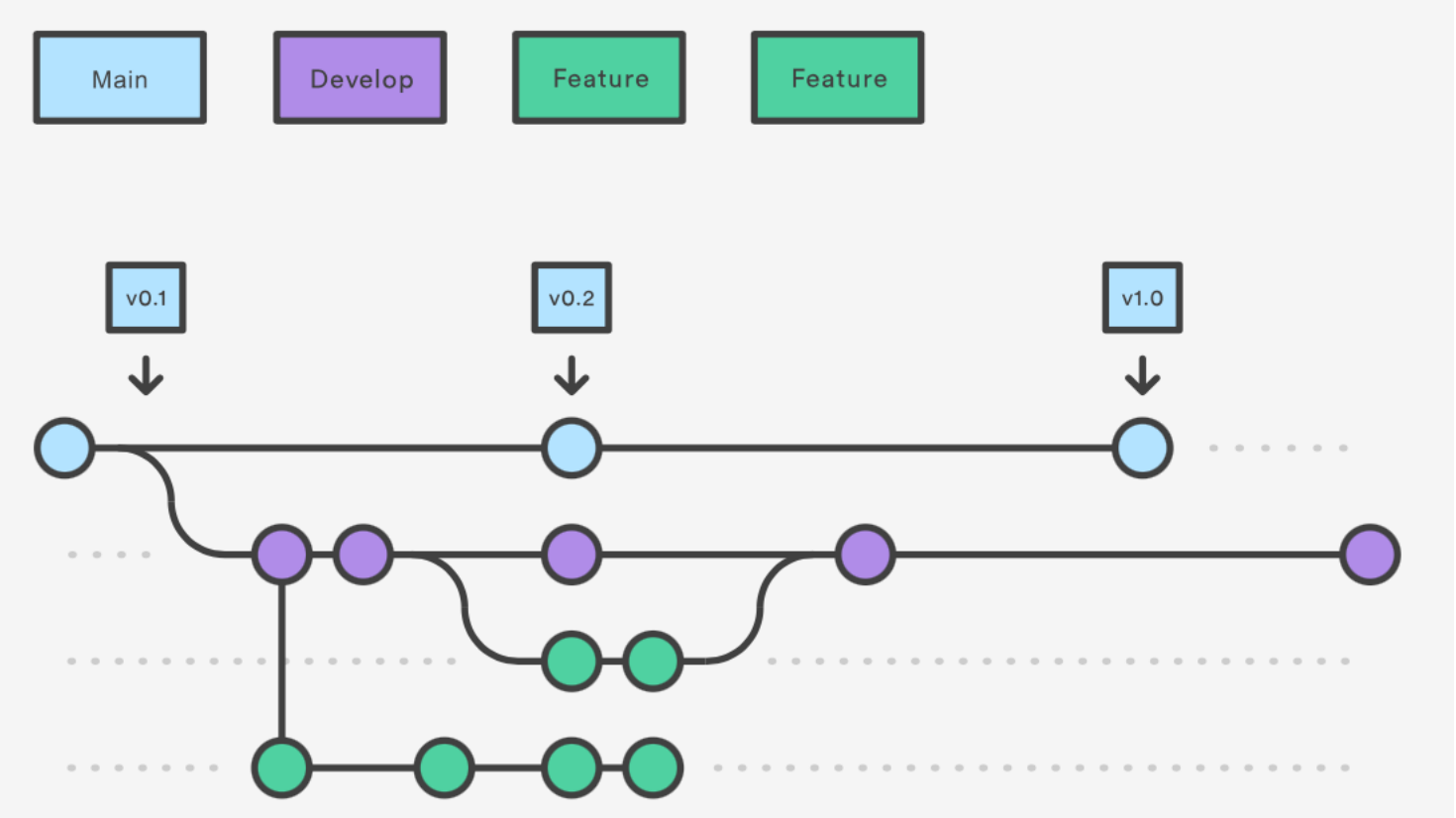
```
git branch develop
git push -u origin develop
```



Feature

```
git checkout develop
git checkout -b feature_branch
```

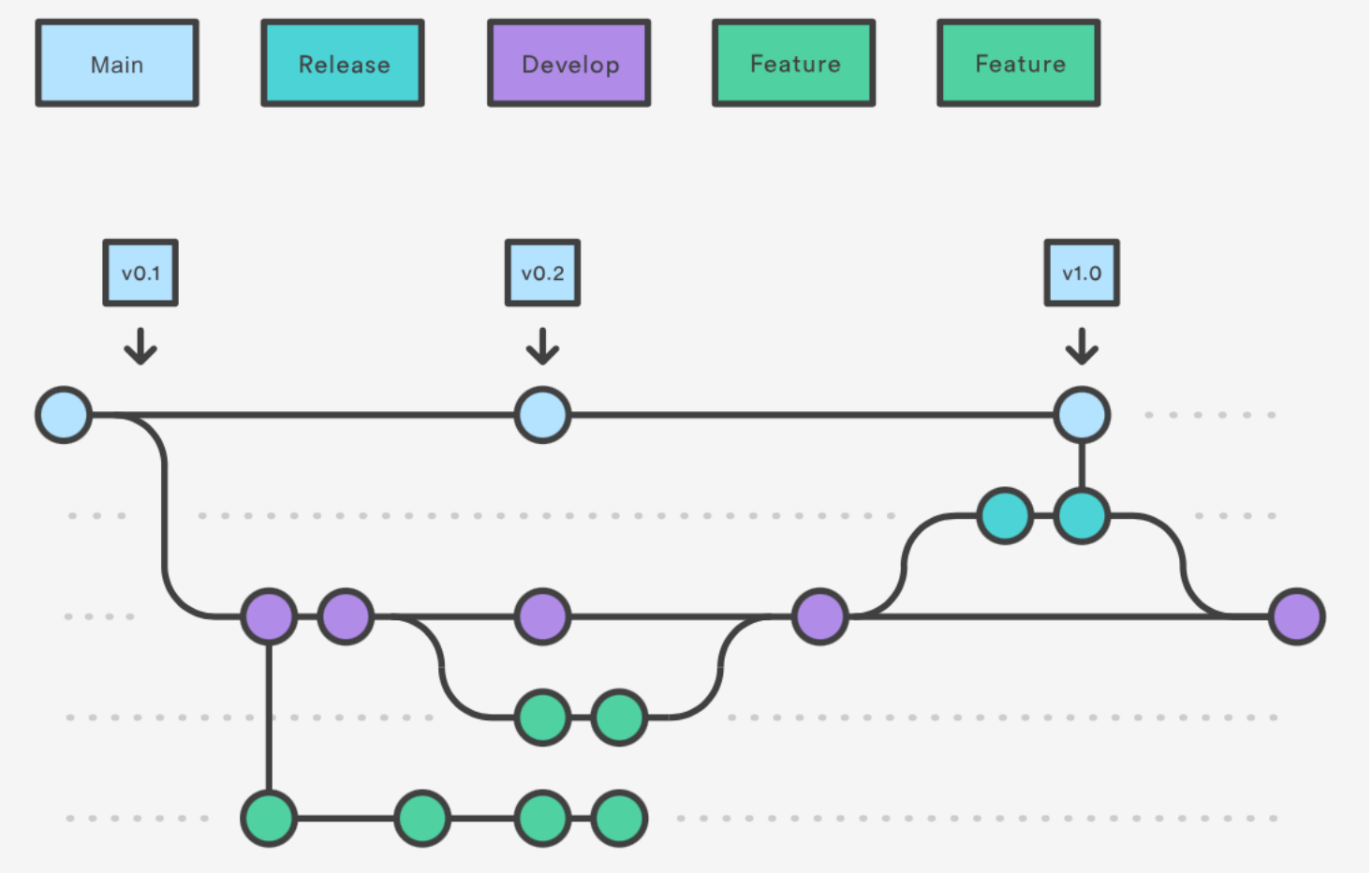
```
git checkout develop
git merge feature_branch
```



Release

```
git checkout develop
git checkout -b release/0.1.0
```

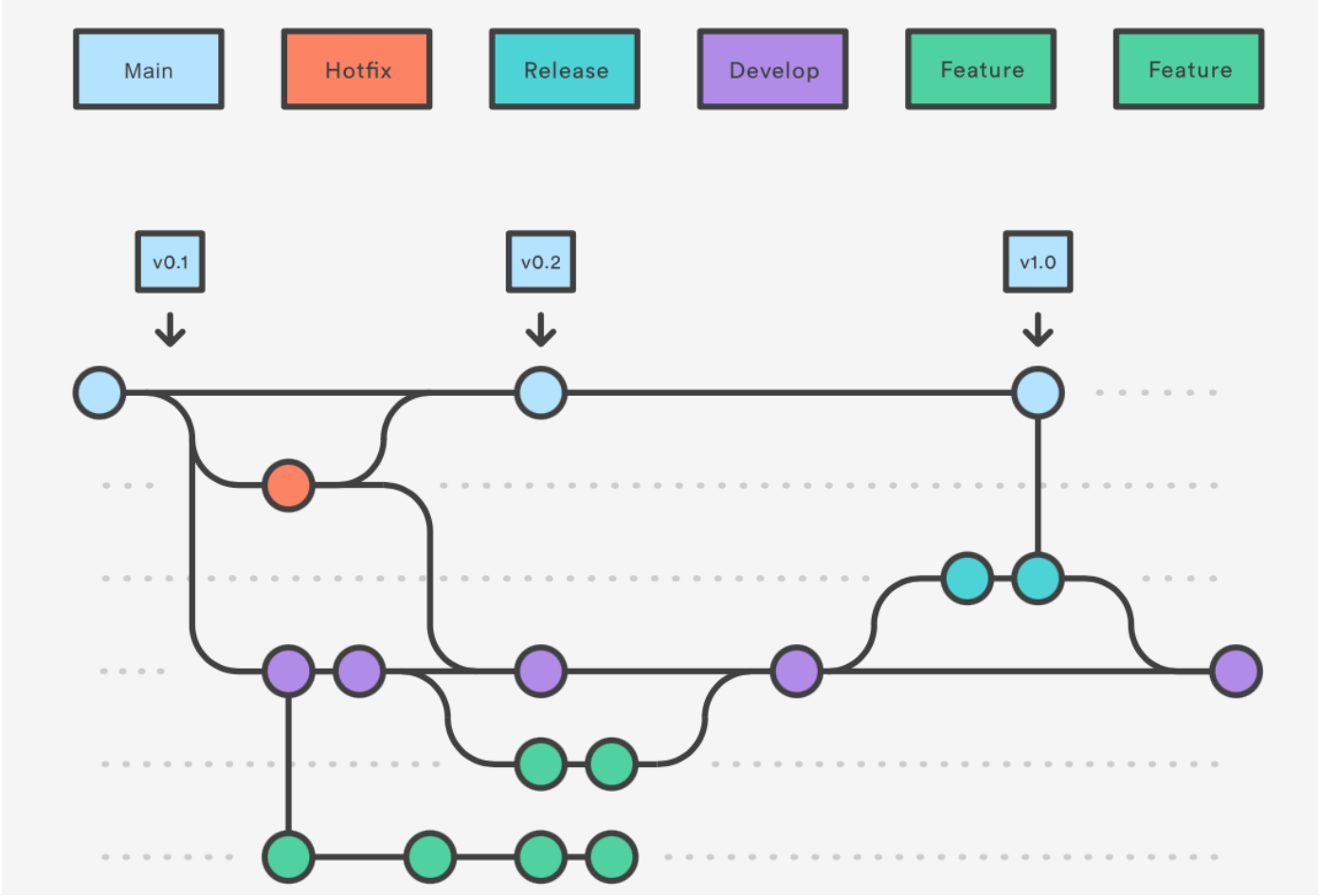
```
git checkout main
git merge release/0.1.0
```



Hotfix

```
git checkout main
git checkout -b hotfix_branch
```

```
git checkout main
git merge hotfix_branch
git checkout develop
git merge hotfix_branch
git branch -D hotfix_branch
```



Обмен по WebSocket

Обмен статусами собеседников через WebSocket

- Необходимо реализовать механизм обмена сообщениями между двумя собеседниками по протоколу WebSocket.
- Должна быть предусмотрена гарантированная отправка и синхронизация версий, если какое-то сообщение пришло раньше-позже (вследствие задержек).
- То есть отображать нужно версии в порядке очередности.

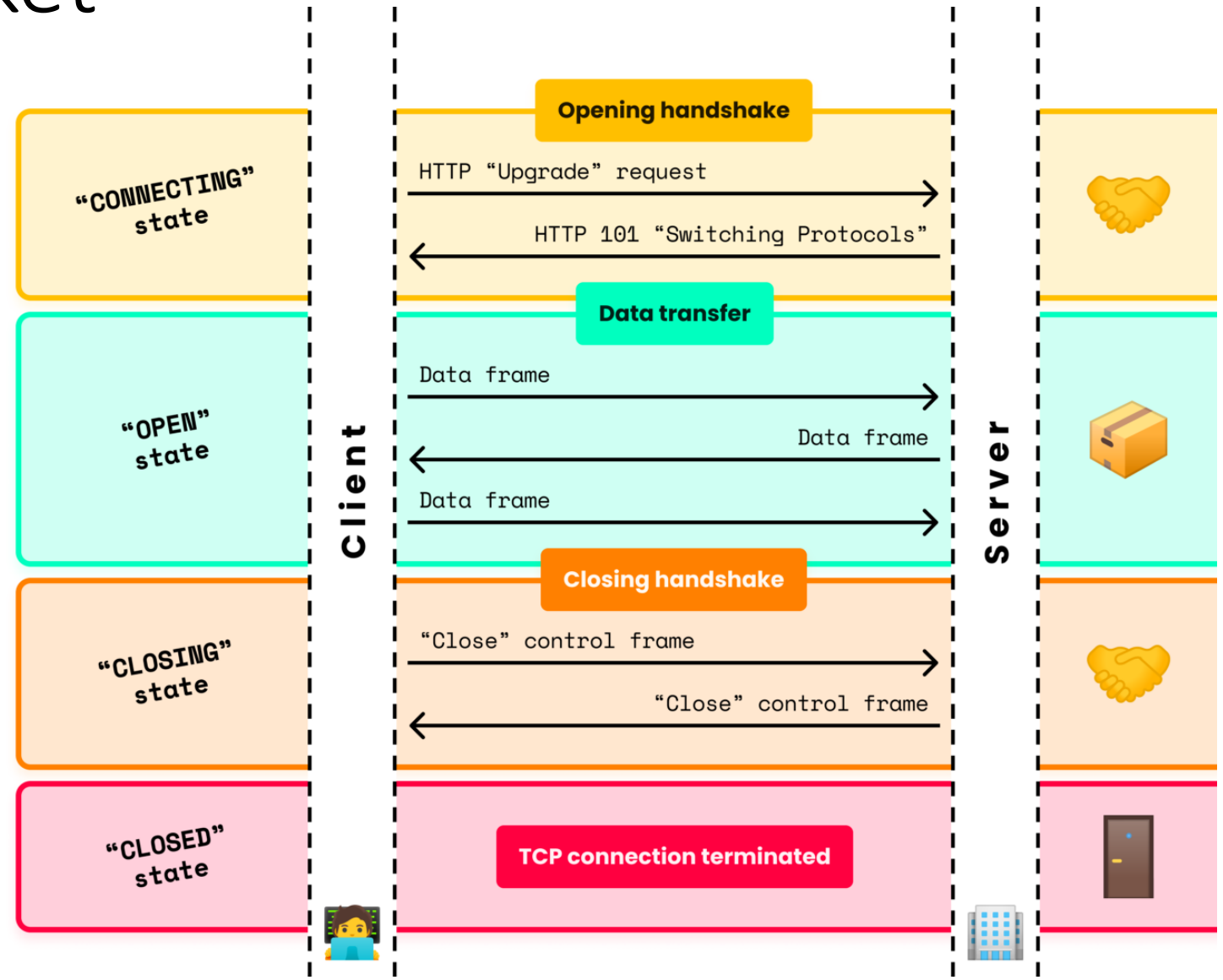
WebSocket

- Протокол связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером, используя постоянное соединение.

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 7
```

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

WebSocket


Opening handshake
HTTP "Upgrade" request
HTTP 101 "Switching Protocols"
"CONNECTING" state
Data transfer
Data frame
Data frame
Data frame
"OPEN" state
Client
Server
Closing handshake
"Close" control frame
"Close" control frame
"CLOSING" state
"CLOSED" state
TCP connection terminated

Клиентский скрипт WebSocket

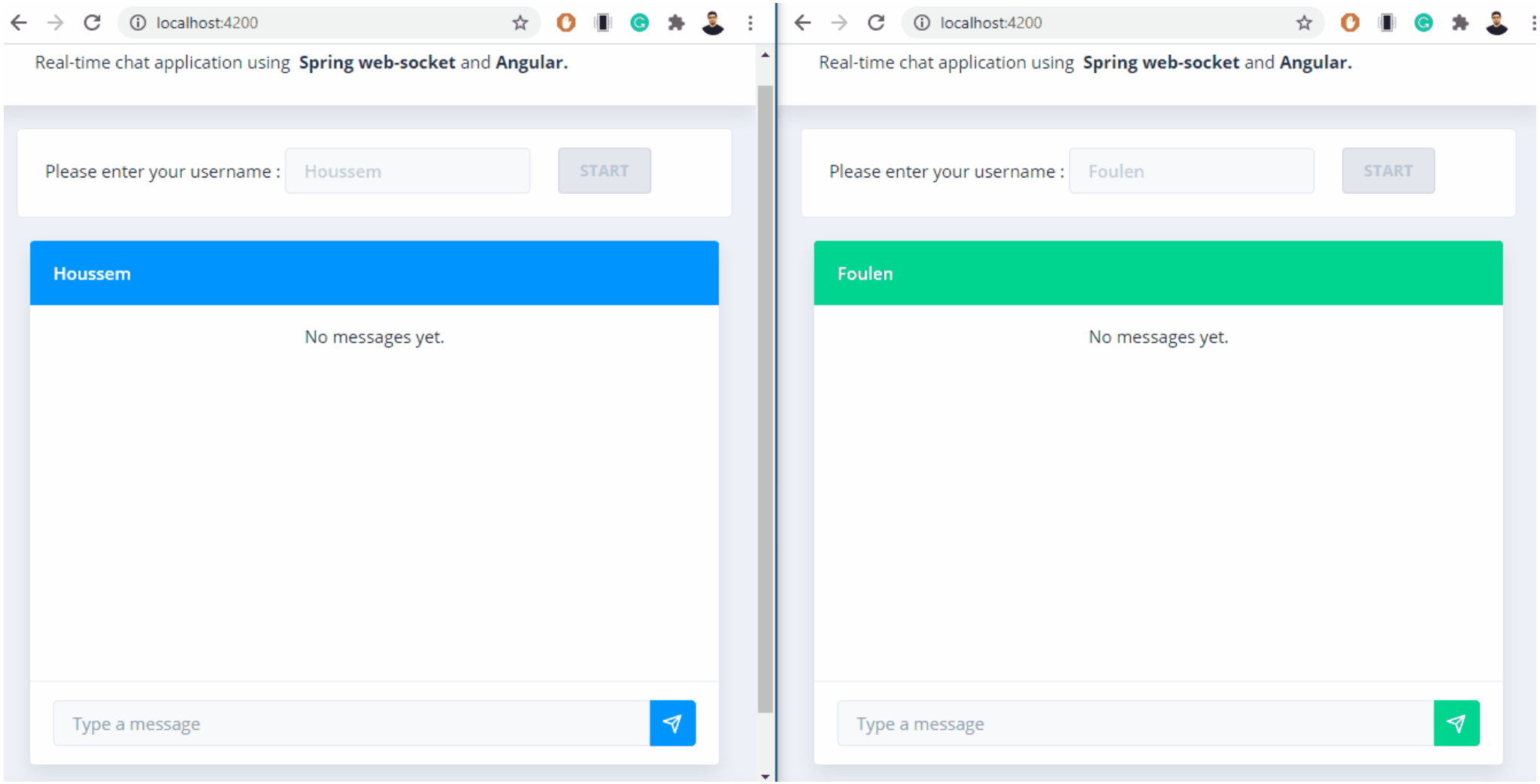
```
<html>
    <head>
        <script>
            const webSocket = new WebSocket('ws://localhost/echo');

            webSocket.onopen = event => {
                alert('onopen');
                webSocket.send("Hello Web Socket!");
            };

            webSocket.onmessage = event => {
                alert('onmessage, ' + event.data);
            };

            webSocket.onclose = event => {
                alert('onclose');
            };
        </script>
    </head>
    <body>
    </body>
</html>
```

WebSocket


localhost:4200
Real-time chat application using Spring web-socket and Angular.
Please enter your username :
Houssem
START
Houssem
No messages yet.
Type a message
localhost:4200
Real-time chat application using Spring web-socket and Angular.
Please enter your username :
Foulen
START
Foulen
No messages yet.
Type a message

Файловое хранилище

Отправка, получение файлов в файловом хранилище

- Необходимо настроить файловое хранилище S3 и разработать методы доступа к нему.
- Хранилище необходимо для синхронизации двух узлов (условные клиент и сервер облака).
- Должна быть предусмотрена синхронизация, если одна из версий устарела (на клиенте не было интернета)

S3

- S3 - Simple Storage Service

Создание озера данных

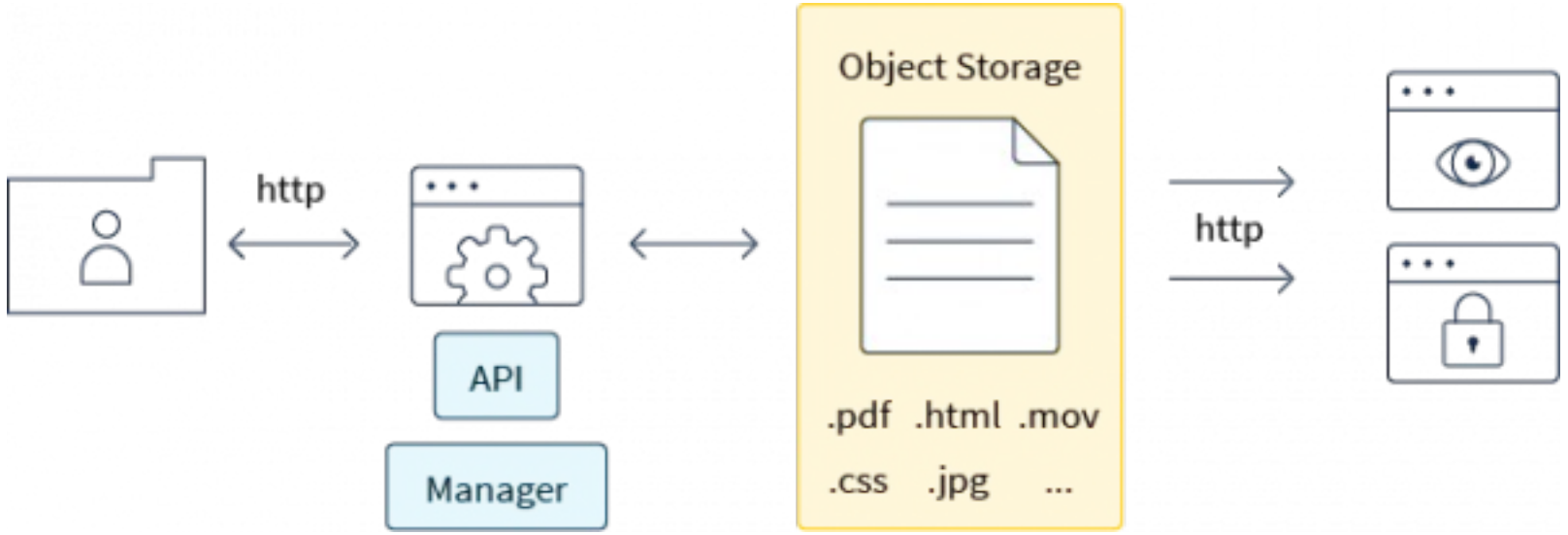
- Приложения для аналитики больших данных, искусственного интеллекта (ИИ), машинного обучения (ML) и высокопроизводительных вычислений (HPC).

- Объектное хранилище S3 — масштабируемый и гибкий сервис, позволяющий хранить и обрабатывать данные в исходном формате
- Данные размещаются в виде объектов в плоском адресном пространстве, что дает возможность для доступа к хранилищу по API из любого места
- Данные помещаются во множество контейнеров (папок)
- Содержимое любого контейнера можно просматривать, перемещать или удалять
- У каждого контейнера и объекта есть адрес в виде уникального идентификатора

Ceph

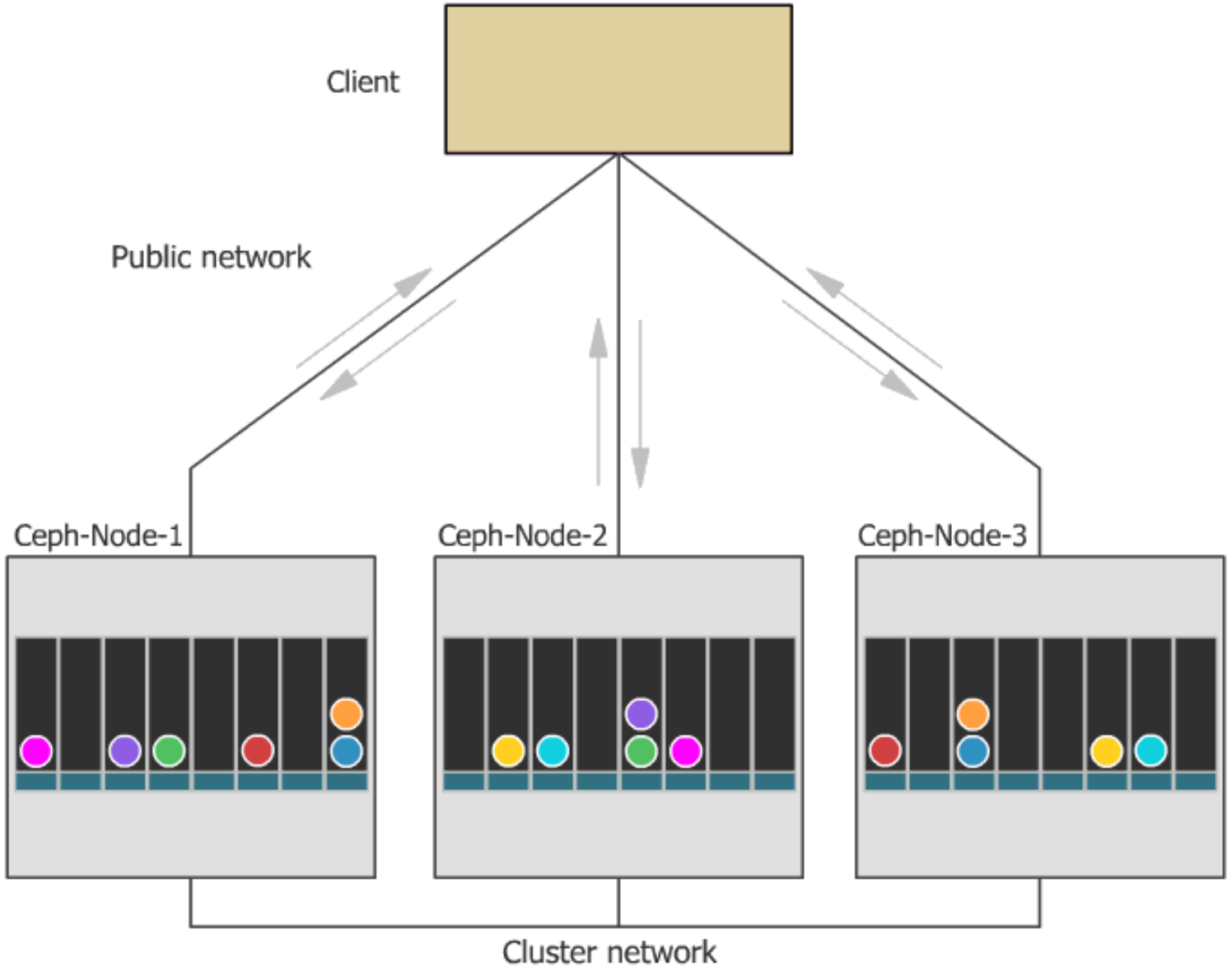
- Свободная программная объектная сеть хранения
- Обеспечивает файловый и блочный интерфейсы доступа

- Объектное хранилище



Ceph

- При выходе любого диска, узла или группы узлов из строя Ceph обеспечит сохранность данных
- Ceph восстановит утраченные копии на других узлах до тех пор, пока вышедшие из строя узлы или диски не заменят на рабочие.



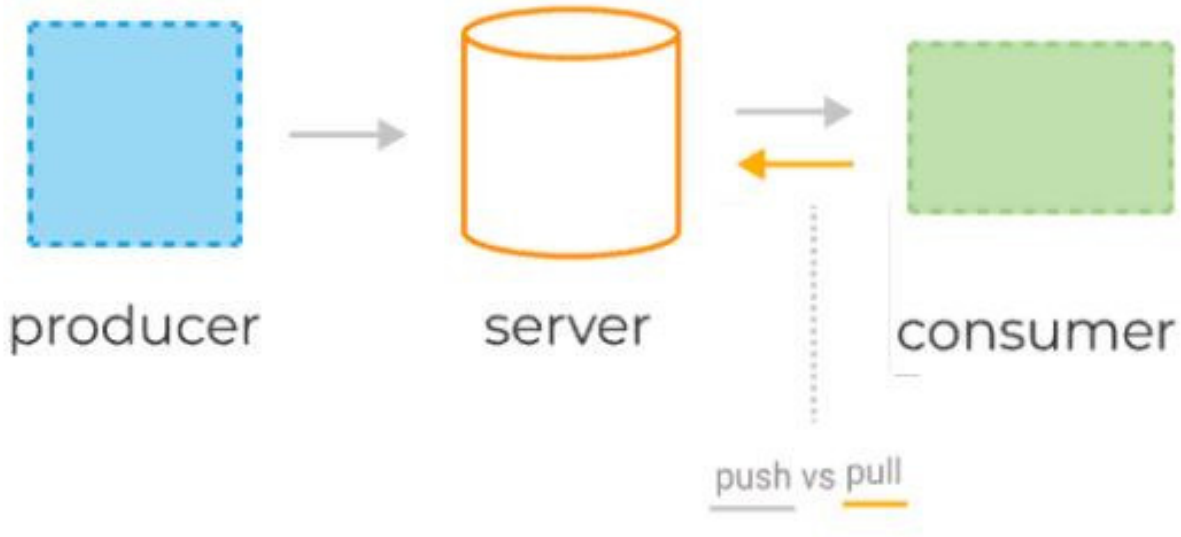
Шлюз сообщений

Отправка email, телеграм, vk через очередь сообщений

- Необходимо разработать два сервиса для отправки сообщений: брокер с очередью сообщений и сервис отправки.
- У каждого студента один из источников (email, телеграм, vk) по варианту.
- За счет очереди должна быть предусмотрена гарантированная отправка в случае недоставки сообщения.
- Если сообщение не было доставлено, оно повторно отправляется из очереди.

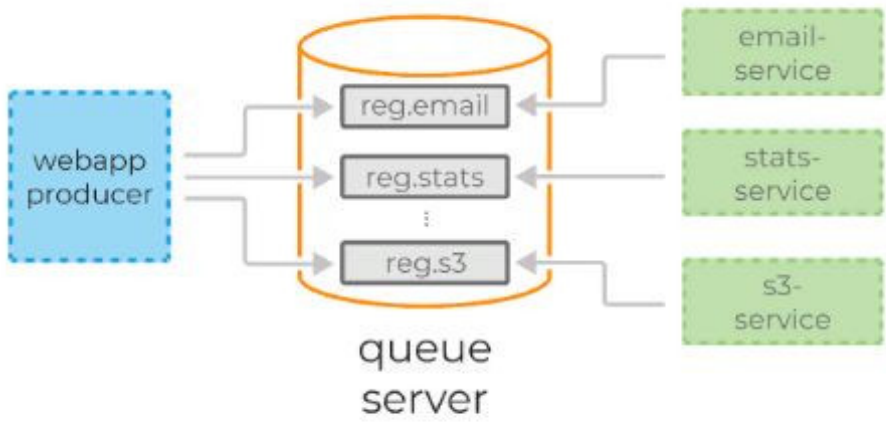
Варианты хранения в брокере

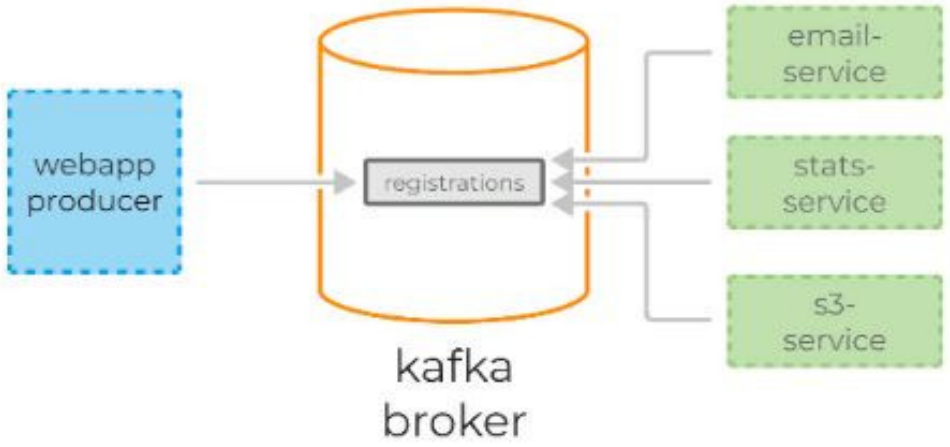
- Redis – резидентная база данных
- Apache Kafka – брокер сообщений
- RabbitMQ – брокер сообщений
- В веб-приложениях очереди часто используются для отложенной обработки событий или в качестве временного буфера между другими сервисами, тем самым защищая их от всплесков нагрузки.



Apache Kafka

- Представим, что есть некий сайт, на котором происходит регистрация пользователя. Для каждой регистрации мы должны:
  1) отправить письмо пользователю,
  2) пересчитать дневную статистику регистраций.
- Kafka упрощает задачу - достаточно послать сообщения всего один раз, а консьюмеры сервиса отправки сообщений и консьюмеры статистики сами считают его по мере необходимости





Redis

- **RE**mote **DI**ctionary **S**erver, «удалённый серверный словарь»
- Резидентная система управления базами данных
- Данные размещаются в оперативной памяти
- Механизмы снимков на дисках для постоянного хранения
- Структура данных ключ-значение, словаря
- Максимальная производительность на атомарных операциях
- Механизм подписок не гарантирует, что сообщение будет доставлено

Redis. Отличия от реляционных

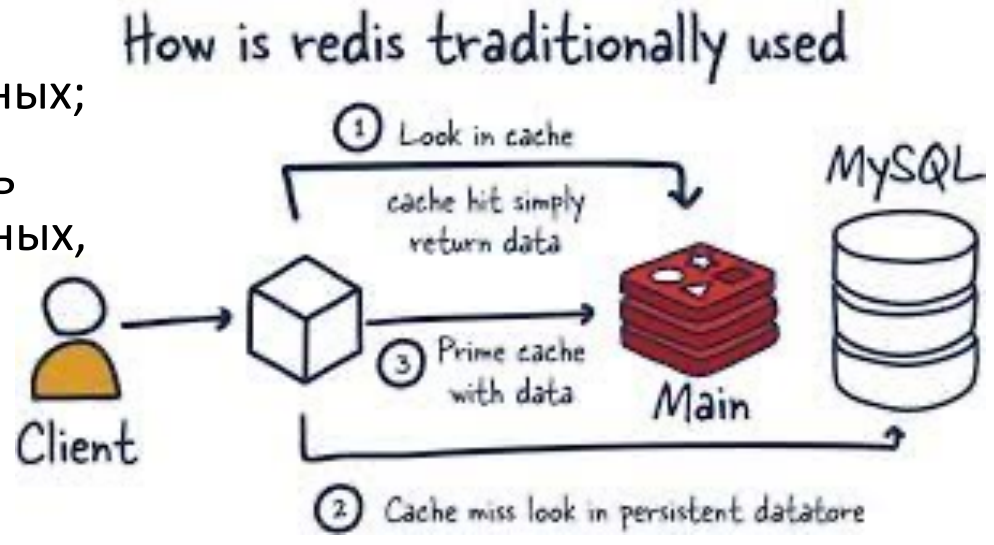
От реляционных баз Redis отличается:

- **более высокой производительностью** (благодаря хранению данных в оперативной памяти сервера, значительно увеличивается число выполняемых операций);
- **отсутствием языка SQL** (Lua-скрипты как альтернатива);
- **гибкостью** (данные находятся не в жёстких структурах (таблицах), а в более удобных (строки, списки, хеши, множества, сортированные множества), что облегчает работу программисту;
- **лучшей масштабируемостью**.

Однако Redis редко используется как основное хранилище в крупных системах, так как не удовлетворяет требованиям ACID, то есть не обеспечивает 100%-ной целостности данных.

Redis. Применение

- для хранения пользовательских сессий (HTML-фрагменты веб-страниц или товары корзины интернет-магазина);
- для хранения промежуточных данных (поток сообщений на стене, голосовалки, таблицы результатов);
- как брокер сообщений (стратегия «издатель-подписчик» позволяет создавать новостные ленты, групповые чаты);
- как СУБД для небольших приложений, блогов;
- для кэширования данных из основного хранилища, что значительно снижает нагрузку на реляционную базу данных;
- для хранения «быстрых» данных — когда важны скорость и критичны задержки передачи (аналитика и анализ данных, финансовые и торговые сервисы).



Redis. Пример

- HSET — сохраняет значение по ключу
- создали объект person1 с двумя полями (name и age) и соответствующими значениями.

```
127.0.0.1:6379> HSET person1 name "Aleksey"
(integer) 1
127.0.0.1:6379> HSET person1 age 25
(integer) 1
```

Redis. Пример

- HGET — получение значения по ключу (для определённого поля)
- Получили значение поля name у ключа person1

```
127.0.0.1:6379> HGET person1 name
"Aleksey"
```