

# ClickHouse для инженеров и архитекторов БД



**Проверить, идет ли запись**

**Меня хорошо видно  
&& слышно?**



Тема вебинара

# Развертывание и базовая конфигурация, интерфейсы и инструменты



**Алексей Железной**

**Senior Data Engineer**

Магистратура - ФКН ВШЭ

Руководитель курсов **DWH Analyst, ClickHouse для инженеров и архитекторов БД** в OTUS

Преподаватель курсов **Data Engineer, DWH Analyst, PostgreSQL** и пр. в OTUS

[LinkedIn](#)

# Правила вебинара



Активно  
участвуем



Задаем вопрос  
в чат или голосом



Вопросы вижу в чате,  
могу ответить не сразу

## Условные обозначения



Индивидуально



Время, необходимое  
на активность



Пишем в чат



Говорим голосом



Документ



Ответьте себе или  
задайте вопрос

# Маршрут вебинара



Варианты установки ClickHouse

Базовая настройка

Интерфейсы и инструменты

Рефлексия

# Цели вебинара

1. Рассмотреть различные варианты установки ClickHouse
2. Понять принципы конфигурирования и настройки данной СУБД;
3. Изучить популярные варианты доступа к БД ClickHouse (интерфейсы, сторонние и нативные библиотеки)

# Варианты установки ClickHouse

# Варианты установки

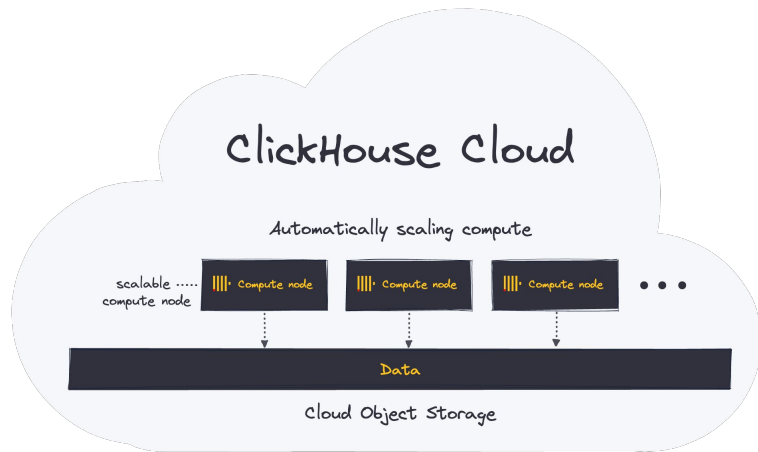
У вас есть такие варианты запуска ClickHouse в эксплуатацию:

1. [ClickHouse Cloud](#): официальный ClickHouse as a service
  - а. Managed сервисы от облачных провайдеров ([Yandex](#), [VK](#))
2. [Quick Install](#): легко загружаемый бинарник для тестирования и разработки.
3. [Production Deployments](#)
  - а. ClickHouse can run on any Linux, FreeBSD, or macOS with x86-64, ARM, or PowerPC64LE CPU architecture
4. [Образ Docker](#): официальный образ Docker в Docker Hub
5. [Homebrew](#) для macOS
6. Кластер ClickHouse на базе Kubernetes [\[1\]](#) [\[2\]](#)



# ClickHouse Cloud

The screenshot displays the ClickHouse Cloud management interface. On the left is a dark sidebar with navigation links: Services, Integrations, Members, Activity, Settings, and Learn & Support. The main content area is titled 'Services' and features a '+ New service' button. A modal window for 'Real-time dashboards v23.3' is open, showing it is 'Running' on 'aws Oregon (us-west-2)' with a 'Connect' button. Below this, other services are listed: 'Logging Service v23.3' (Running on aws Oregon (us-west-2)), 'Data Science Playground v23.3' (Running on aws Ohio (us-east-2)), 'BI Dashboards v23.3' (Running on aws Ireland (eu-west-1)), and 'Analytics Service v23.3' (Running on aws Ireland (eu-west-1)). Each service card includes a status indicator, provider/region information, and a 'Connect' button. The top right of the interface shows 'My Organization' and user avatars.



## Serverless. **Simple.** ClickHouse Cloud.

Get the performance you love from open source ClickHouse in a serverless offering that takes care of the details so you can spend more time getting insight out of the fastest database on earth.

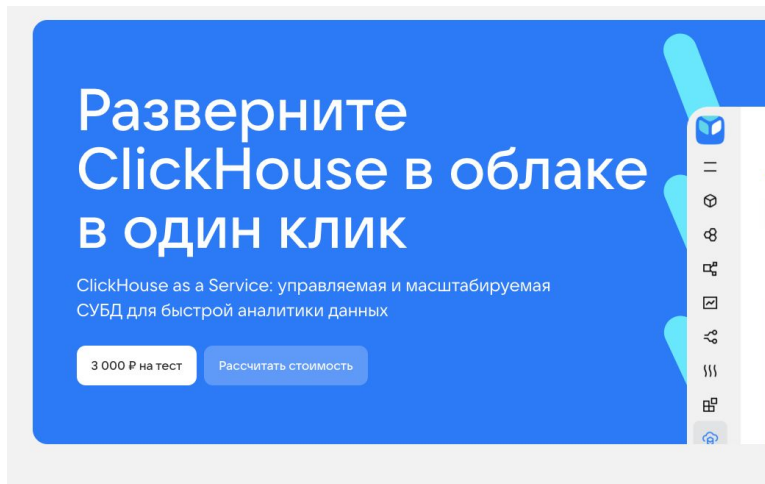
Start free trial



Soon

Interested in being notified when Azure is available? [Join the waitlist](#)

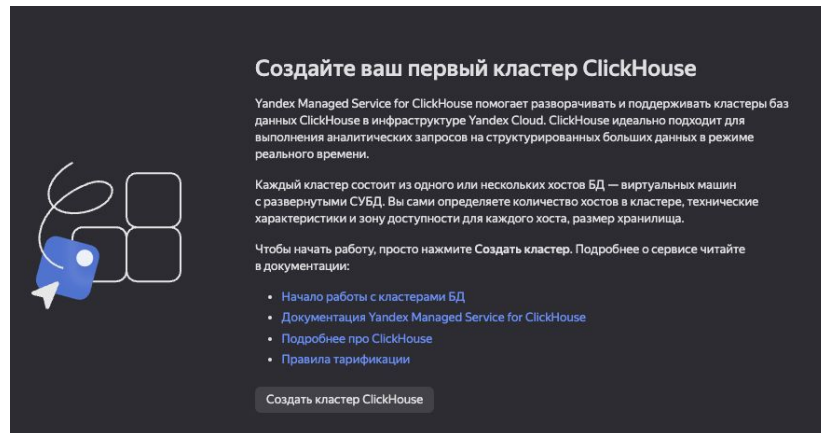
# Managed Service for ClickHouse



**Разверните ClickHouse в облаке В ОДИН КЛИК**

ClickHouse as a Service: управляемая и масштабируемая СУБД для быстрой аналитики данных

3 000 Р на тест    Рассчитать стоимость



## Создайте ваш первый кластер ClickHouse

Yandex Managed Service for ClickHouse помогает разворачивать и поддерживать кластеры баз данных ClickHouse в инфраструктуре Yandex Cloud. ClickHouse идеально подходит для выполнения аналитических запросов на структурированных больших данных в режиме реального времени.

Каждый кластер состоит из одного или нескольких хостов БД — виртуальных машин с развернутыми СУБД. Вы сами определяете количество хостов в кластере, технические характеристики и зону доступности для каждого хоста, размер хранилища.

Чтобы начать работу, просто нажмите Создать кластер. Подробнее о сервисе читайте в документации:

- Начало работы с кластерами БД
- Документация Yandex Managed Service for ClickHouse
- Подробнее про ClickHouse
- Правила тарификации

Создать кластер ClickHouse

# Quick install

1. Команда для загрузки бинарника для вашей операционной системы (Linux, macOS and FreeBSD), который может использоваться для запуска сервера ClickHouse, clickhouse-client, clickhouse-local, ClickHouse Keeper и других инструментов
  - `curl https://clickhouse.com/ | sh`
2. Запускаем ClickHouse Server
  - `./clickhouse server`
3. В новом терминале подключаемся к клиенту
  - `./clickhouse client`

# Docker image

## **Versions**

- The latest tag points to the latest release of the latest stable branch.
- Branch tags like 22.2 point to the latest release of the corresponding branch.
- Full version tags like 22.2.3.5 point to the corresponding release.
- The tag head is built from the latest commit to the default branch.
- Each tag has optional -alpine suffix to reflect that it's built on top of alpine.

# Docker image

start server instance:

- `docker run -d --name some-clickhouse-server --ulimit nofile=262144:262144 clickhouse/clickhouse-server`

connect to it:

- **from a native client**
  - `docker run -it --rm --link some-clickhouse-server:clickhouse-server --entrypoint clickhouse-client clickhouse/clickhouse-server --host clickhouse-server`
  - `docker exec -it some-clickhouse-server clickhouse-client`
- **using curl**
  - `echo "SELECT 'Hello, ClickHouse!'" | docker run -i --rm --link some-clickhouse-server:clickhouse-server curlimages/curl 'http://clickhouse-server:8123/?query=' -s --data-binary @-`

# Docker image

stopping / removing the container:

- `docker stop some-clickhouse-server`
- `docker rm some-clickhouse-server`

# Production deployments

Варианты установки:

1. DEB Packages
  - для Debian и Ubuntu
  - [Ручное скачивание](#)
2. RPM Packages
  - для CentOS, RedHat и всех других rpm-дистрибутивов Linux
3. Tgz Archives
  - для всех дистрибутивов Linux, где установка deb- или rpm-пакетов невозможна.

# Packages

- **clickhouse-common-static** — Installs ClickHouse compiled binary files.
- **clickhouse-server** — Creates a symbolic link for clickhouse-server and installs the default server configuration.
- **clickhouse-client** — Creates a symbolic link for clickhouse-client and other client-related tools. and installs client configuration files.
- **clickhouse-common-static-dbg** — Installs ClickHouse compiled binary files with debug info.
- **clickhouse-keeper** - Used to install ClickHouse Keeper on dedicated ClickHouse Keeper nodes. Installs ClickHouse Keeper and the default ClickHouse Keeper configuration files.



# ClickHouse Keeper

- Сервер ClickHouse использует сервис координации ZooKeeper для репликации данных и выполнения распределенных DDL запросов. ClickHouse Keeper — это альтернативный сервис координации, совместимый с ZooKeeper.
- По умолчанию ClickHouse Keeper предоставляет те же гарантии, что и ZooKeeper. ClickHouse Keeper предоставляет совместимый клиент-серверный протокол, поэтому любой стандартный клиент ZooKeeper может использоваться для взаимодействия с ClickHouse Keeper.
- В производственной среде запускать ClickHouse Keeper на выделенных узлах.

# ClickHouse Keeper. Зачем?

- Создан осенью 2022 года. [Текущие обновления](#)
- Координатор как замена Zookeeper от Apache
- Написан на C++. Должен работать быстрее и кушать меньше памяти
- Меньше вопросов с настройкой, так как идет в комплекте

## Нюансы\*:

- Мало документации
- Не хватает нормального конфига. Часть берется из основного конфига клика
- Нет режима суперпользователя
- Все же не совместим с Zookeeper (ошибки `Coordination::Exception: Operation 101 is unknown (Unimplemented)`)

# Non-Production Deployments

- Компиляция из источников
  - Инструкция для [Linux](#)
  - Инструкция для [macOS](#)
- Install a CI-generated Binary
  - Инфраструктура непрерывной интеграции (CI) ClickHouse создает специализированные сборки для каждого коммита в [репозитории ClickHouse](#)

# Вопросы?



Ставим "+",  
если вопросы есть



Ставим "-",  
если вопросов нет

# Базовая конфигурация, запуск и тестовые датасеты

# Конфигурационные файлы

ClickHouse хранит конфигурацию в 2 различных конфигурационных файлах:

- `/etc/clickhouse-server/config.xml` содержит настройки сервера.
- `/etc/clickhouse-server/users.xml` содержит конфигурацию пользователей и настройки сервера, которые могут быть переопределены для каждого пользователя.

## ВАЖНО!

Непосредственно редактировать эти файлы не рекомендуется, вместо этого следует хранить конфигурацию в отдельных файлах в каталогах

`/etc/clickhouse-server/config.d/` и `/etc/clickhouse-server/users.d/`.

# Пользовательские конф. файлы

```
<!-- /etc/clickhouse-server/config.d/my.xml -->
<?xml version="1.0"?>
<clickhouse>
  <listen_host>:::</listen_host>
  <timezone>UTC</timezone>
</clickhouse>
```

xml

Or for user settings:

```
<!-- /etc/clickhouse-server/users.d/my.xml -->
<?xml version="1.0"?>
<clickhouse>
  <profiles>
    <default>
      <prefer_column_name_to_alias>1</prefer_column_name_to_alias>
    </default>
  </profiles>
</clickhouse>
```

xml

# Какой файл выбрать?

ClickHouse поставляется с таблицей `system.settings`, которая содержит настройки сеанса для текущего пользователя, например:

```
SELECT * FROM system.settings WHERE name = 'queue_max_wait_ms'
```

Если настройка существует в таблице `system.settings`, то ее следует поместить в каталог `users.d`. В противном случае попробуйте использовать каталог `config.d`.



# User Settings

users.xml и users.d содержат конфигурацию пользователей и настройки, которые могут быть переопределены для каждого пользователя. Чтобы переопределить настройки для пользователя по умолчанию:

```
<clickhouse>
  <users>
    <alice>
      <profile>analytics</profile>
      <networks>
        <ip>::/0</ip>
      </networks>
      <password_sha256_hex>...</password_sha256_hex>
      <quota>analytics</quota>
    </alice>
  </users>
</clickhouse>
```

# User Settings

```
SELECT value  
FROM system.settings  
WHERE name = 'prefer_column_name_to_alias'  
SETTINGS prefer_column_name_to_alias = 0;
```

```
SELECT value  
FROM system.settings  
WHERE name = 'prefer_column_name_to_alias'  
SETTINGS prefer_column_name_to_alias = 1;
```

# Какие настройки бывают и где посмотреть?

Примеры:

- `max_memory_usage`
- `max_execution_time`
- `force_primary_key`
- ...

<https://clickhouse.com/docs/en/operations/settings/settings>

<https://github.com/ClickHouse/ClickHouse/blob/master/src/Core/Settings.h>

<https://github.com/ClickHouse/ClickHouse/blob/master/docs/en/operations/server-configuration-parameters/settings.md?plain=1>

# Рекомендации после установки. ClickHouse

1. Настройте параметры памяти:
  - **max\_memory\_usage** - параметр управляет максимальным объемом памяти, который может использовать ClickHouse.
2. Настройка количества потоков:
  - Например, можно увеличить количество потоков, используемых для операций SELECT, чтобы повысить производительность чтения.
3. Настройка размера буфера:
  - Это позволит минимизировать количество операций дискового ввода-вывода.
4. Настройте merge settings
5. Настройка параметров сжатия
6. Настройка параметров реплики
7. Отслеживайте производительность

# Рекомендации после установки. Сервер

8. Увеличьте количество открытых файловых дескрипторов: Увеличьте значение параметра "ulimit -n" не менее чем до 100000.
9. Увеличьте объем разделяемой памяти: Увеличьте значение параметра "kernel.shmmax" не менее чем до 1 Гб.
10. Настроить прозрачные огромные страницы ([transparent huge pages](#)): Отключите прозрачные огромные страницы, добавив в командную строку ядра значение "transparent\_hugepage=never".
11. Настроить планировщик: Измените планировщик с используемого по умолчанию "cfq" на "noop" или "deadline".
12. Оптимизировать работу сети: Увеличьте количество сетевых очередей, включите автонастройку TCP и отключите опции разгрузки.
13. Контролируйте загрузку системы: Используйте такие инструменты, как "top" или "htop", для мониторинга загрузки системы, использования памяти и дискового ввода-вывода.
14. Мониторинг производительности ClickHouse: Используйте встроенные средства мониторинга производительности ClickHouse для выявления и устранения узких мест в работе.

# Рекомендации по выбору сервера

1. Идем поэтапно - память (RAM), процессоры, диски (SSD, NVMe)
  - Пример - 512GB, 2 проца Gold, ядер больше 16 (прод - больше 32)
2. Ограничения:
  - `max_concurrent_queries_for_user` и `max_threads` для пользователей
  - `max_result_rows` и `max_result_bytes`, чтобы избежать падения “веб-морды”
  - `max_bytes_in_set` и `max_rows_in_join`, чтобы ограничить джоины и распределенные запросы
  - `max_bytes_before_external_group_by`, в зависимости от объема оперативки
3. Еще:
  - Процессоры не армы!
  - Не забывайте про бекапы и где их в итоге нужно складывать
  - Оперативку считаем от объема словарей

# Запуск

Чтобы запустить сервер в качестве демона, выполните следующие действия:

- `$ sudo clickhouse start`

Существуют и другие способы запуска ClickHouse:

- `$ sudo service clickhouse-server start`
- `$ sudo /etc/init.d/clickhouse-server start`
- `$ sudo systemctl start clickhouse-server.service`
- Ручками из консоли `$ clickhouse-server --config-file=/etc/clickhouse-server/config.xml`

Просмотрите логи в каталоге `/var/log/clickhouse-server/`.

# Тестовые датасеты

- [Example Datasets](#)
- [Advanced Tutorial](#)



# Вопросы?



Ставим "+",  
если вопросы есть



Ставим "-",  
если вопросов нет



# Драйверы и интерфейсы

# Интерфейсы доступа к БД

## 1. Сетевые интерфейсы:

- a. HTTP
- b. TCP
- c. gRPC

## 2. Библиотеки:

- a. Command-line client
- b. JDBC driver
- c. ODBC driver
- d. C++ client library

## 3. Визуальные интерфейсы

- a. Play UI
- b. Advanced Dashboard
- c. Binary symbols viewer for ClickHouse engineers

## 4. Сторонние библиотеки

- a. Клиентские библиотеки
- b. Интеграции
- c. Визуальные интерфейсы

# CLI

# CLI (Command-Line Client)

clickhouse-client

- Клиент можно использовать в интерактивном и неинтерактивном (пакетном) режиме.
- Подключение через TCP :
  - ХОСТ и ПОРТ: обычно порт 9440 при использовании TLS или 9000 при отсутствии TLS.
  - ИМЯ БАЗЫ ДАННЫХ
  - ИМЯ ПОЛЬЗОВАТЕЛЯ И ПАРОЛЬ

```
clickhouse-client --host <HOSTNAME> \  
                  --secure \  
                  --port 9440 \  
                  --user <USERNAME> \  
                  --password <PASSWORD>
```

```
clickhouse-client --host HOSTNAME.clickhouse.cloud \  
                  --secure \  
                  --port 9440 \  
                  --user default \  
                  --password PASSWORD \  
                  --query "INSERT INTO cell_towers FORMAT CSVWithNames" \  
                  < cell_towers.csv
```

# CLI - Варианты вставки данных

```
\copy (select * from table) to '~/filename' with csv;  
cat ~/filename' | clickhouse-client -h host -u user --password  
--query="insert into table format CSV"
```

```
cat <<_EOF | clickhouse-client --database=test --query="INSERT INTO test FORMAT CSV";  
3, 'some text', '2016-08-14 00:00:00'  
4, 'some more text', '2016-08-14 00:00:01'  
_EOF
```

```
cat file.csv | clickhouse-client --database=test --query="INSERT INTO test FORMAT CSV";
```

```
echo -ne "1, 'some text', '2016-08-14 00:00:00'\n2, 'some more text', '2016-08-14 00:00:01'" | \  
clickhouse-client --database=test --query="INSERT INTO test FORMAT CSV";
```

# CLI - Заметки

- В пакетном режиме по умолчанию TabSeparated
- Параметр `--multiquery` для выполнения нескольких запросов (кроме insert)
- `\G` для вертикального формата
- История командной строки записывается в `~/.clickhouse-client-history`
- Для выхода из клиента - "exit", "quit", "logout", "exit;", "quit;", "logout;", "q", "Q", ":q"
- Для отмены запроса - `ctrl+C` (`cmd+C`). Повторно для завершения работы клиента

```
$ clickhouse-client --param_parName="[1, 2]" -q "SELECT * FROM table WHERE a = {parName:Array(UInt16)}"
```

# CLI - Конфигурация и параметры

clickhouse-client использует первый существующий файл из следующих:

- Определяется в параметре --config-file.
- ./clickhouse-client.xml, .yaml, .yml
- ~/.clickhouse-client/config.xml, .yaml, .yml
- /etc/clickhouse-client/config.xml, .yaml, .yml

--host, -h

--port

--user, -u

--password

--ask-password

--query, -q

--queries-file

--multiquery,

--multiline, -m

--database, -d --format, -f

--vertical, -E

--time, -t

--stacktrace

--config-file

--secure --history\_file

--param\_<name>

--hardware-utilization

--print-profile-events

--profile-events-delay-ms



# CLI - Connection string

- Можно подключаться к нескольким хостам (URI)
- Кодировка нестандартных символов в процентах

```
clickhouse://[user[:password]@][hosts_and_ports][/database][?query_parameters]
```

```
clickhouse-client clickhouse://localhost:9000 --query "SELECT 1"
```

```
clickhouse-client clickhouse:
```

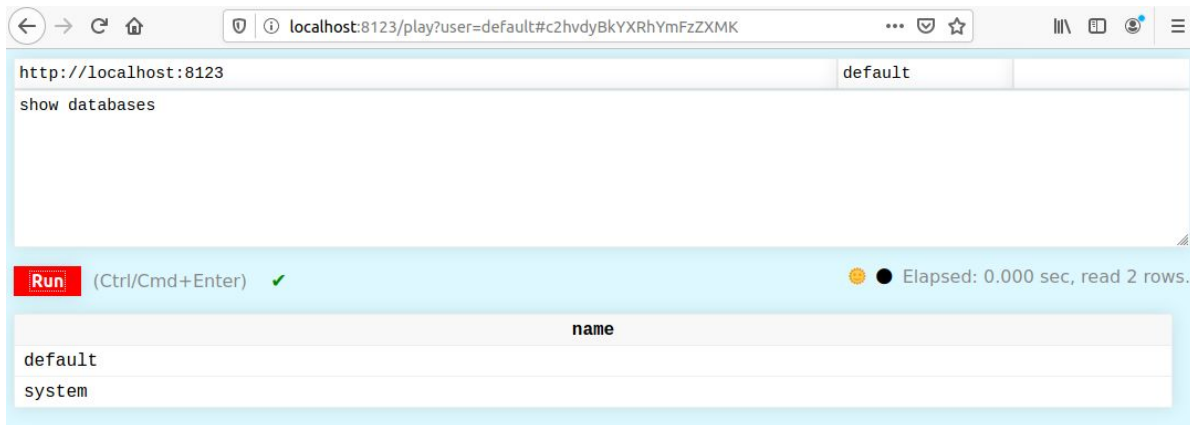
```
clickhouse-client clickhouse://some_user%40some_mail.com@localhost:9000
```

```
clickhouse-client clickhouse://192.168.1.15,192.168.1.25
```

# HTTP Interface

# HTTP Interface

- Позволяет использовать ClickHouse на любой платформе с любого языка программирования в виде REST API
- HTTP port - 8123, HTTPS - 8443
- Web UI - <http://localhost:8123/play> (/dashboard)



# HTTP Interface - полезные запросы

- `curl 'http://localhost:8123/ping'`
- `curl 'http://localhost:8123/replicas_status'`
- Для запросов, изменяющих данные - POST, GET - readonly.
  - `curl 'http://localhost:8123/?query=SELECT%201'`
  - `echo -ne 'GET /?query=SELECT%201 HTTP/1.0\r\n\r\n' | nc localhost 8123`
- `echo 'SELECT 1 FORMAT Pretty' | curl 'http://localhost:8123/?user=default&password=default'`  
`--data-binary @-`
- Метод POST для передачи данных необходим для запросов INSERT.
  - `echo 'CREATE TABLE t (a UInt8) ENGINE = Memory' | curl 'http://localhost:8123/'`  
`--data-binary @-`

# HTTP Interface - Сжатие

- Сжатие используется для уменьшения сетевого трафика при передаче большого количества данных или для создания дампов, которые сразу же сжимаются
- clickhouse-compressor для работы с внутренним форматом сжатия
- compress=1 в URL - сервер будет сжимать данные, которые он вам отправляет.  
decompress=1, сервер будет сжимать данные, которые вы передаете в методе POST.
- Вы также можете выбрать использование HTTP-сжатия
  - `echo "SELECT 1" | gzip -c | curl -sS --data-binary @- -H 'Content-Encoding: gzip' 'http://localhost:8123/'`
  - `curl -vsS "http://localhost:8123/?enable_http_compression=1" -H 'Accept-Encoding: gzip' --output result.gz -d 'SELECT number FROM system.numbers LIMIT 3'`
  - `curl -sS "http://localhost:8123/?enable_http_compression=1" -H 'Accept-Encoding: gzip' -d 'SELECT number FROM system.numbers LIMIT 3' | gunzip -`

# HTTP Interface - Аутентификация

- `echo 'SELECT 1' | curl 'http://user:password@localhost:8123/' -d @-`
- `echo 'SELECT 1' | curl 'http://localhost:8123/?user=user&password=password' -d @-`
- `echo 'SELECT 1' | curl -H 'X-ClickHouse-User: user' -H 'X-ClickHouse-Key: password' 'http://localhost:8123/' -d @-`

Примеры:

- <https://github.demo.altinity.cloud:8443/play> (user=demo/pw=demo)

# HTTP Interface - параметры

- `curl -sS "<address>?param_id=2&param_phrase=test" -d "SELECT * FROM table WHERE int_column = {id:UInt8} and string_column = {phrase:String}"`
- `curl -sS "http://localhost:8123?param_arg1=abc%5C%09123" -d "SELECT splitByChar('\t', {arg1:String})"`

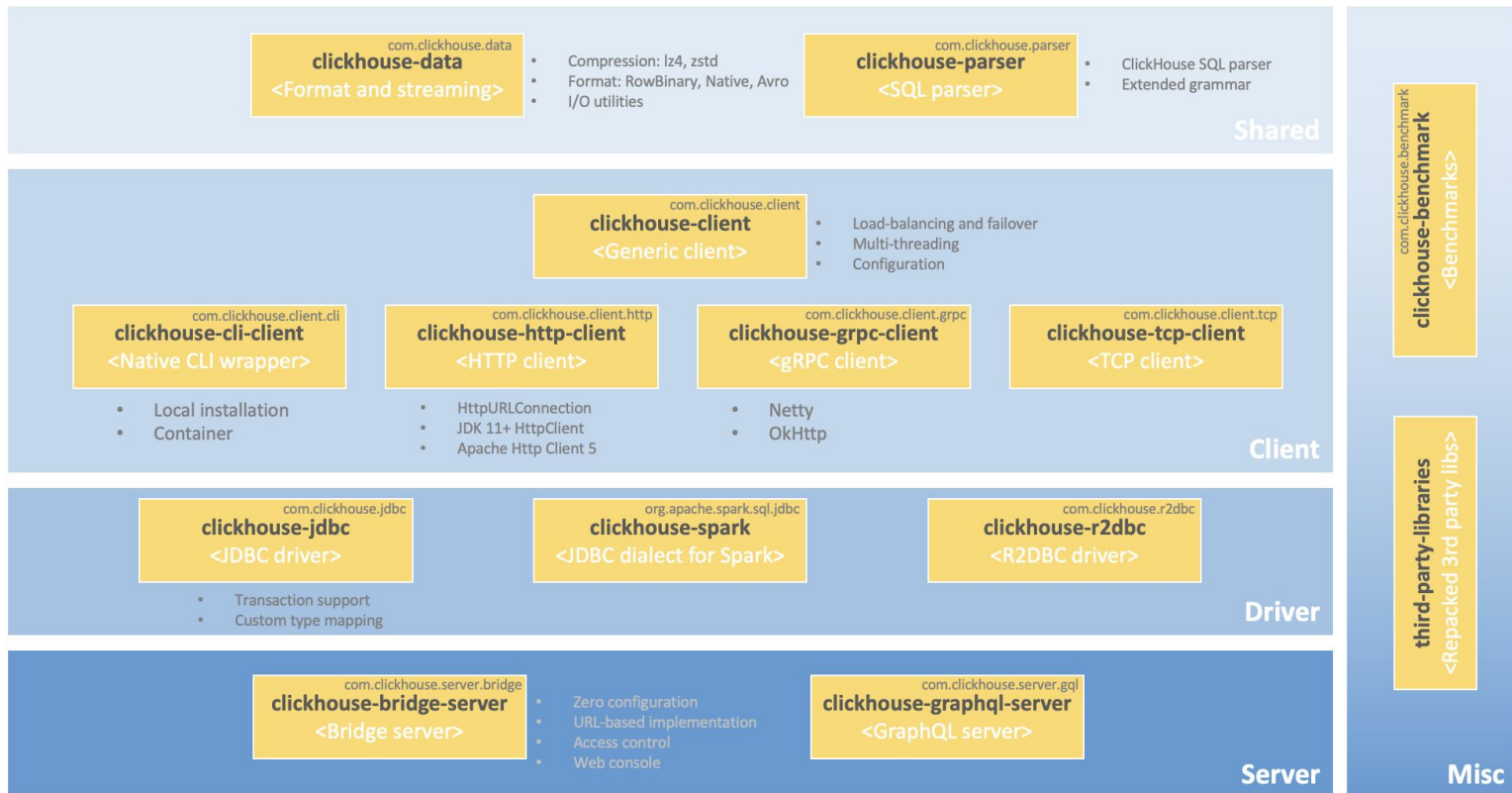
# Прочие интерфейсы и драйверы



# JDBC/ODBC Drivers

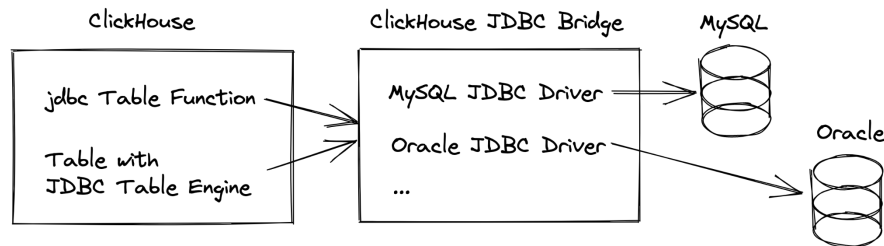
- ClickHouse Java Libraries - Java-библиотеки для подключения к ClickHouse и обработки данных в различных форматах. Java-клиент - это асинхронная, легкая и не требующая больших затрат библиотека для ClickHouse, а драйверы JDBC и R2DBC построены поверх Java-клиента и имеют больше зависимостей и возможностей.
- ODBC Driver for ClickHouse - это официальная реализация драйвера ODBC для доступа к ClickHouse в качестве источника данных.

# ClickHouse Java Libraries



# Connecting ClickHouse to external data sources with JDBC

- ClickHouse JDBC Bridge в сочетании с функцией jdbc table или JDBC table engine позволяет ClickHouse получать доступ к данным из любого внешнего источника данных, для которого доступен драйвер JDBC:



# ClickHouse JDBC

- Это удобно, когда нет встроенного механизма интеграции, табличной функции или внешнего словаря для внешнего источника данных, но существует JDBC-драйвер для источника данных.
- Можно использовать ClickHouse JDBC Bridge как для чтения, так и для записи.
- Можно использовать параллельно для нескольких внешних источников данных.

# ClickHouse ODBC

- Драйвер ClickHouse Open Database Connectivity(ODBC) позволяет пользователям подключать различные приложения к ClickHouse, например, подключать Microsoft Excel, Tableau Desktop и другие платформы.
- Официальный ODBC-драйвер для ClickHouse доступен в бинарных версиях для дистрибутивов Microsoft Windows и Linux. Его также можно установить путем компиляции исходного кода для других операционных систем, например Mac OS X.

[Как подружить ClickHouse и Power BI](#)

# Databases Interfaces

## PostgreSQL Interface

- Протокол PostgreSQL wire
- *В некотором смысле ClickHouse может притвориться экземпляром PostgreSQL, что позволит вам подключить к ClickHouse клиентское приложение PostgreSQL, которое еще не поддерживается ClickHouse напрямую (например, Amazon Redshift).*
- Новый файл в config.d - определяем порт
- `psql -p [port] -h [hostname] -U [username] [database_name]`

```
<clickhouse>  
  <postgresql_port>9005</postgresql_port>  
</clickhouse>
```

# Databases Interfaces

## MySQL Interface

- Протокол MySQL wire
- Новый файл в config.d - определяем порт
- Подключение - `mysql --protocol tcp -h [hostname] -u [username] -P [port_number] [database_name]`
- Указывать в конфигурационном файле пароль пользователя с двойным SHA1, иначе некоторые клиенты не смогут пройти аутентификацию

```
<clickhouse>
  <mysql_port>9004</mysql_port>
</clickhouse>
```

# Автоматическое определение схемы на основе входных данных

- ClickHouse может автоматически определять структуру входных данных почти во всех поддерживаемых форматах ввода (Schema Inference).
- Вывод схемы используется, когда ClickHouse необходимо прочитать данные в определенном формате, а их структура неизвестна.

```
{"id" : 1, "age" : 25, "name" : "Josh", "hobbies" : ["football", "cooking", "music"]}
{"id" : 2, "age" : 19, "name" : "Alan", "hobbies" : ["tennis", "art"]}
{"id" : 3, "age" : 32, "name" : "Lana", "hobbies" : ["fitness", "reading", "shopping"]}
{"id" : 4, "age" : 47, "name" : "Brayan", "hobbies" : ["movies", "skydiving"]}
```

```
SELECT * FROM file('hobbies.jsonl')
```

```
DESCRIBE file('hobbies.jsonl')
```



# Schema Inference - Примеры

- `CREATE TABLE hobbies ENGINE=File(JSONEachRow, 'hobbies.jsonl')`
- `clickhouse-local --file='hobbies.jsonl' --table='hobbies' --query='DESCRIBE TABLE hobbies'`
- `use_structure_from_insertion_table_in_table_functions`
  - 0 - табличная функция будет извлекать структуру из данных.
  - 1 - функция таблицы будет использовать структуру из таблицы вставки.
  - 2 - ClickHouse автоматически определит, можно ли использовать структуру из таблицы вставки или использовать вывод схемы. Значение по умолчанию.

# Schema Inference Cache

- Чтобы предотвратить вывод одной и той же схемы каждый раз, когда ClickHouse считывает данные из одного и того же файла, выведенная схема кэшируется, и при повторном обращении к тому же файлу ClickHouse будет использовать схему из кэша.
  - DESCRIBE TABLE  
`s3('https://datasets-documentation.s3.eu-west-3.amazonaws.com/github/github-2022.ndjson.gz')`
  - SETTINGS allow\_experimental\_object\_type = 1
  - DESCRIBE TABLE  
`s3('https://datasets-documentation.s3.eu-west-3.amazonaws.com/github/github-2022.ndjson.gz')`
  - SETTINGS input\_format\_json\_read\_objects\_as\_strings = 1
  - SELECT schema, format, source FROM system.schema\_inference\_cache WHERE storage='S3'
  - SYSTEM DROP SCHEMA CACHE FOR S3

# Schema Inference - Text formats

- Для текстовых форматов ClickHouse считывает данные строка за строкой, извлекает значения столбцов в соответствии с форматом, а затем использует некоторые рекурсивные парсеры и эвристики для определения типа каждого значения.
  - DESC format(JSONEachRow, '{"int" : 42, "float" : 42.42, "string" : "Hello, World!"}');
  - DESC format(CSV, '42,42.42,true,"Hello,World!"')
  - DESC format(Values, \$\$ (42, 42.42, true, 'Hello,World!') \$\$)

# Сторонние интерфейсы

# Другие библиотеки

- [Client libraries](#)
- [Integrations](#)
- [GUI](#)
- [Proxies](#)

# Другие библиотеки

- [C++ Client Library](#)
- [Native Interface \(TCP\)](#) - используется в клиенте командной строки, для межсерверного взаимодействия при распределенной обработке запросов, а также в других программах на C++.
- [gRPC Interface](#). gRPC - это современный высокопроизводительный фреймворк для удаленного вызова процедур (RPC) с открытым исходным кодом.

# Вопросы?



Ставим "+",  
если вопросы есть



Ставим "-",  
если вопросов нет

# Домашнее задание



# Домашнее задание

1. Установить ClickHouse.
2. Подгрузить датасет для примера и сделать селект из таблицы.
3. Для проверки отправить скрины работающего инстанса ClickHouse, созданной виртуальной машины и результата запроса select.
4. Провести тестирование производительности и сохранить результаты;
  - а. `echo "SELECT * FROM default.cell_towers LIMIT 10000000 OFFSET 10000000" | clickhouse-benchmark -i 10`
5. Изучить конфигурационные файлы БД;
6. Произвести наиболее оптимальную настройку системы на основании характеристик вашей ОС и провести повторное тестирование;
7. Подготовить отчет касательно прироста/изменения производительности системы на основе проведенных настроек.

Результатом выполнения работы является подготовленный отчет в формате pdf, в котором указано описание всех выполненных пунктов. Поощряется работа в отдельных гит-репозиториях

# Рефлексия

# Рекомендации по Self-Managed ClickHouse

- ClickHouse использует все доступные аппаратные ресурсы для обработки данных.
- ClickHouse работает эффективнее с большим количеством ядер с низкой тактовой частотой, чем с меньшим количеством ядер с высокой тактовой частотой.
- Для выполнения нетривиальных запросов мы рекомендуем использовать не менее 4 Гбайт оперативной памяти. Сервер ClickHouse может работать и с гораздо меньшим объемом оперативной памяти, но в этом случае запросы будут часто прерываться.
- Необходимый объем оперативной памяти в общем случае зависит от:
  - Сложности запросов.
  - Объема данных, обрабатываемых в запросах.
- Для расчета необходимого объема оперативной памяти можно оценить размер временных данных для GROUP BY, DISTINCT, JOIN и других используемых операций.
- Для уменьшения потребления памяти ClickHouse может осуществлять обмен временными данными на внешнюю память.

# Рекомендации по Self-Managed ClickHouse

- В производственных средах рекомендуется отключать файл подкачки операционной системы (swap file).
- Объем памяти, необходимый для хранения ваших данных, может быть рассчитан отдельно на основе оценки объема данных и коэффициента сжатия данных.
- Для расчета конечного объема данных, подлежащих хранению, примените коэффициент сжатия к предполагаемому объему данных. Если планируется хранить данные в нескольких репликах, то умножьте расчетный объем на количество реплик.
- При распределенном развертывании ClickHouse (кластеризации) рекомендуется использовать сетевое подключение класса не ниже 10G.
- Пропускная способность сети критична для обработки распределенных запросов с большим объемом промежуточных данных. Кроме того, скорость сети влияет на процессы репликации.

# Рефлексия



С какими впечатлениями уходите с вебинара?



Как будете применять на практике то, что узнали на вебинаре?

# Список материалов для изучения

1. [Установка и использование ClickHouse на Linux](#)
2. [How to manage server config files in Clickhouse](#)
3. [Available Installation Options](#)
4. [Clickhouse init tutorial](#)
5. [How to Install ClickHouse on Ubuntu 22.04 LTS Linux](#)
6. [Installation and Management of clickhouse-operator for Kubernetes](#)

**Заполните, пожалуйста,  
опрос о занятии  
по ссылке в чате**

Спасибо за внимание!

# Приходите на следующие вебинары



**Алексей Железной**

**Senior Data Engineer**

Магистратура - ФКН ВШЭ

Руководитель курсов **DWH Analyst, ClickHouse** для инженеров и архитекторов  
**БД** в OTUS

Преподаватель курсов **Data Engineer, DWH Analyst, PostgreSQL** и пр. в OTUS

[LinkedIn](#)