

Яндекс

**Я**ндекс Облако

# **Как устроено резервное копирование ClickHouse в Яндекс.Облаке?**

Александр Бурмак, старший разработчик

# Особенности бэкапов в облаке

**Произвольная нагрузка и схема базы данных**

**Произвольная конфигурация кластера**

- › количество шардов и реплик
- › системные ресурсы (CPU, memory, network, disk I/O)
- › тип и размер хранилища

**Обязательное шифрование данных**

# Что бэкапить?

- › Схема базы данных
- › Данные таблиц
- › Конфигурация ClickHouse'a
- › Данные ZooKeeper'a

# Что бэкапить?

- › Схема базы данных
- › Данные таблиц
- › Конфигурация ClickHouse'a
- › ~~Данные ZooKeeper'a~~

# Движки таблиц

- › Семейство MergeTree
- › Log, TinyLog, StripeLog
- › View, MaterializedView
- › Distributed
- › Kafka, MySQL, ODBC
- › Set, Join, Buffer, Null
- › ...

# MergeTree таблицы

Создание резервной копии данных

- › `ALTER TABLE ... FREEZE [PARTITION ...]`
- › Копируем данные из shadow/ в бэкап хранилище

Восстановление

- › Копируем данные из бэкап хранилища в detached/
- › `ALTER TABLE ... ATTACH PARTITION|PART ...`

# Существующие инструменты

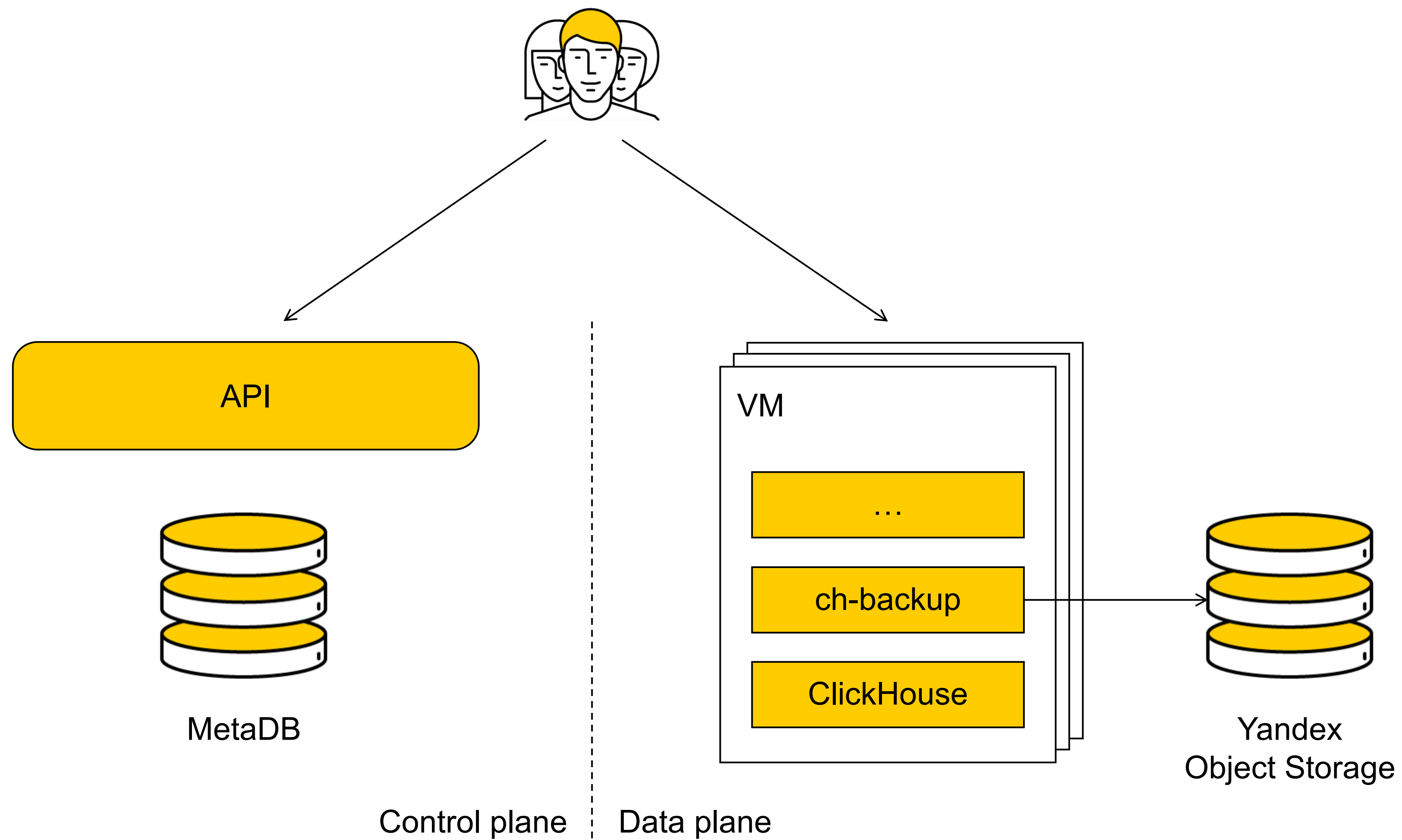
- › clickhouse-backup \*  
<https://github.com/AlexAkulov/clickhouse-backup>
- › clickhouse-copier \*
- › Снимки файловой системы

\* появились после разработки собственного инструмента



# **Бэкапы в Яндекс.Облаке**

# Архитектура



# ch-backup

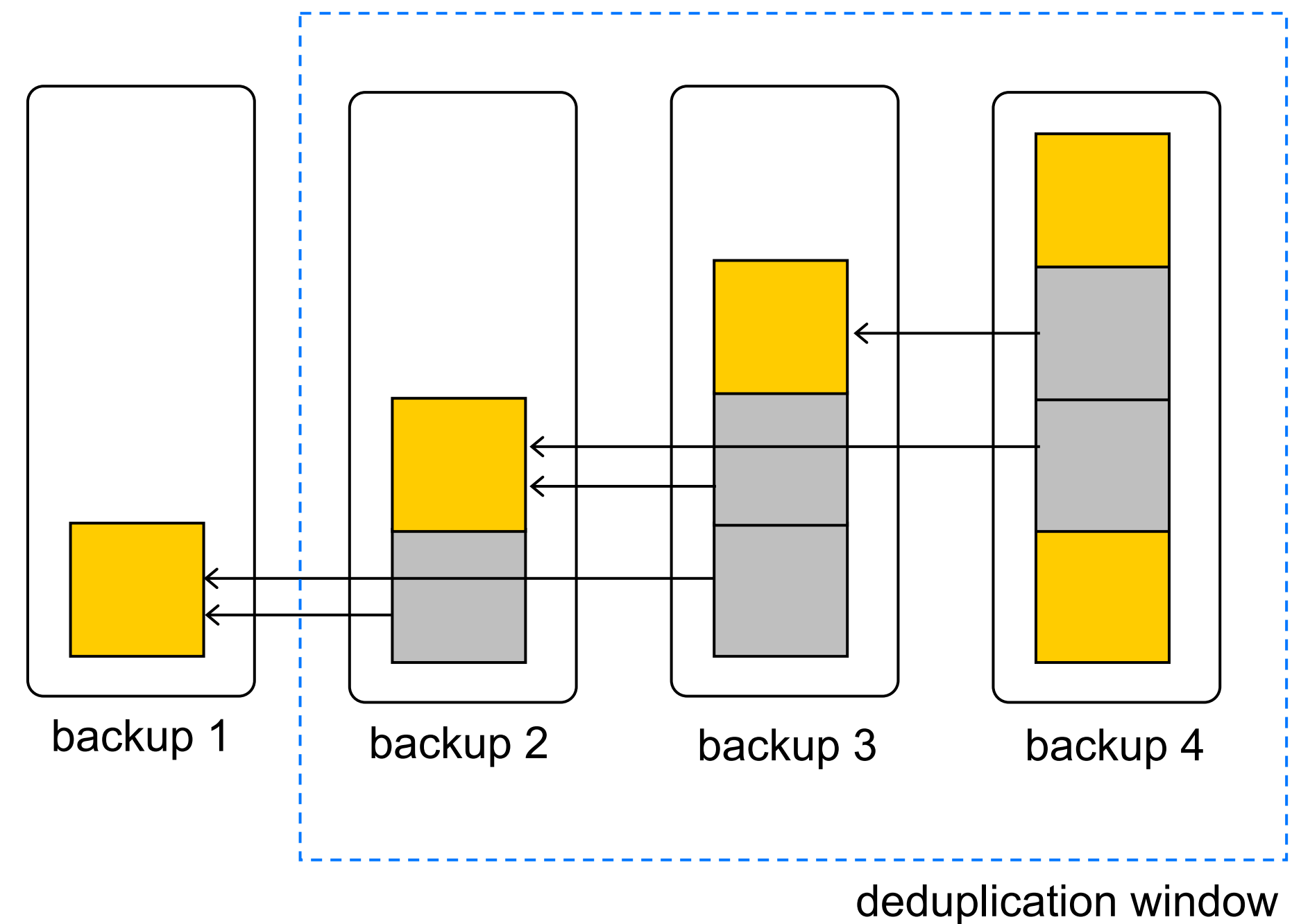
Написан на Python

Основные возможности

- › Параллелизм
- › Инкрементальные бэкапы
- › Правила хранения (retention policies)
- › Шифрование

# Инкрементальные бэкапы

- › Гранулярность данных - куски
- › Сохраняются только измененные куски
- › Наличие идентичных кусков проверяется в последних N бэкапов



# Удаление

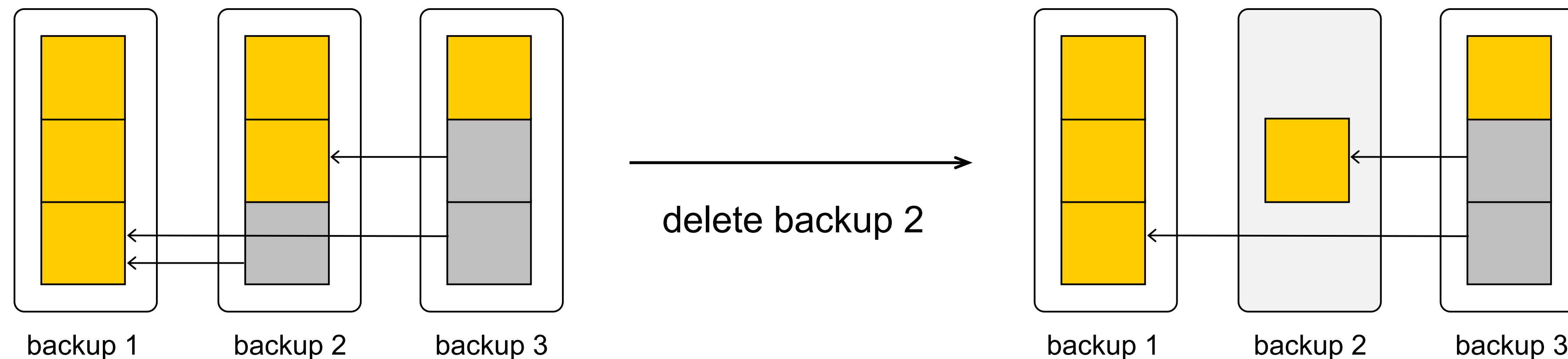
Выборочное

На основе правил

- › хранить N бэкапов
- › хранить бэкапы за N дней

# Удаление и инкрементальные бэкапы

Удаляются только куски, на которые отсутствуют ссылки



Чистка "частично удаленных" бэкапов происходит при следующем удалении

# Шифрование

- | PyNaCl (binding to the libsodium library)
  - › Симметричное шифрование
  - › API для потоковой обработки
- | Отдельный ключ шифрования для каждого кластера

# Сжатие

- Данные в ClickHouse хранятся уже в сжатом виде

- Дополнительное сжатие при создании бэкапа не используется



**Проблемы**

# Зависимости между таблицами в схеме

```
rc1c-r1tryw3j1bt9xss0.mdb.yandexcloud.net :) SELECT name, create_table_query FROM system.tables
WHERE database != 'system'
```

| name             | create_table_query  |
|------------------|---|
| events           | CREATE TABLE default.events ...<br>ENGINE = Distributed('{cluster}', default, events_part)            |
| events_agg1      | CREATE TABLE default.events_agg1 ...<br>ENGINE = Distributed('{cluster}', default, events_agg1_part)  |
| events_agg1_part | CREATE MATERIALIZED VIEW events_agg1_part ...<br>AS SELECT ... FROM events_part                       |
| events_agg2      | CREATE TABLE default.events_agg2 ...<br>ENGINE = Distributed('{cluster}', default, events_agg2_part)  |
| events_agg2_part | CREATE MATERIALIZED VIEW events_agg2_part ...<br>AS SELECT ... FROM events_part                       |
| events_part      | CREATE TABLE events_part ...<br>ENGINE = MergeTree() ...  |
| view             | CREATE VIEW default.view ...<br>AS SELECT ... FROM events_agg1 INNER JOIN events_agg2 USING (user_id) |

В какой последовательности восстанавливать?

# Зависимости между таблицами в схеме

Способ 1. Использовать загрузчик ClickHouse'a

1. Создаем create table файлы в /var/lib/clickhouse/metadata
2. Перезагружаем ClickHouse сервер

# Зависимости между таблицами в схеме

## Способ 2. Использовать mtime

```
rc1c-r1tryw3j1bt9xss0.mdb.yandexcloud.net :) SELECT name, create_table_query, metadata_modification_time FROM system.tables WHERE database != 'system'
```

| name             | create_table_query  | metadata_modification_time |
|------------------|---|----------------------------|
| events_part      | CREATE TABLE events_part ...<br>ENGINE = MergeTree() ...  | 2019-08-28 18:14:19        |
| events           | CREATE TABLE default.events ...<br>ENGINE = Distributed('{cluster}', default, events_part)            | 2019-08-28 18:14:20        |
| events_agg1_part | CREATE MATERIALIZED VIEW events_agg1_part ...<br>AS SELECT ... FROM events_part                       | 2019-08-28 18:14:23        |
| events_agg1      | CREATE TABLE default.events_agg1 ...<br>ENGINE = Distributed('{cluster}', default, events_agg1_part)  | 2019-08-28 18:14:24        |
| events_agg2_part | CREATE MATERIALIZED VIEW events_agg2_part ...<br>AS SELECT ... FROM events_part                       | 2019-08-28 18:14:24        |
| events_agg2      | CREATE TABLE default.events_agg2 ...<br>ENGINE = Distributed('{cluster}', default, events_agg2_part)  | 2019-08-28 18:14:25        |
| view             | CREATE VIEW default.view ...<br>AS SELECT ... FROM events_agg1 INNER JOIN events_agg2 USING (user_id) | 2019-08-28 18:14:27        |

# Зависимости между таблицами в схеме

## ~~Способ 2. Использовать mtime~~

```
rc1c-r1tryw3j1bt9xss0.mdb.yandexcloud.net :) ALTER TABLE events_part ADD CONSTRAINT c1 CHECK insert_time > event_time

rc1c-r1tryw3j1bt9xss0.mdb.yandexcloud.net :) SELECT name, create_table_query, metadata_modification_time FROM system.tables
WHERE database != 'system'
```

| name             | create_table_query  | metadata_modification_time |
|------------------|---|----------------------------|
| events           | CREATE TABLE default.events ...<br>ENGINE = Distributed('{cluster}', default, events_part)                | 2019-08-28 18:14:20        |
| events_agg1_part | CREATE MATERIALIZED VIEW events_agg1_part ...<br>AS SELECT ... FROM events_part                           | 2019-08-28 18:14:23        |
| events_agg1      | CREATE TABLE default.events_agg1 ...<br>ENGINE = Distributed('{cluster}', default, events_agg1_part)      | 2019-08-28 18:14:24        |
| events_agg2_part | CREATE MATERIALIZED VIEW events_agg2_part ...<br>AS SELECT ... FROM events_part                           | 2019-08-28 18:14:24        |
| events_agg2      | CREATE TABLE default.events_agg2 ...<br>ENGINE = Distributed('{cluster}', default, events_agg2_part)      | 2019-08-28 18:14:25        |
| view             | CREATE VIEW default.view ...<br>AS SELECT ... FROM events_agg1 INNER JOIN events_agg2 USING (user_id)     | 2019-08-28 18:14:27        |
| events_part      | CREATE TABLE events_part (... , CONSTRAINT c1 CHECK insert_time > event_time)<br>ENGINE = MergeTree() ... | 2019-08-28 18:19:11        |

# Зависимости между таблицами в схеме

## Способ 3. Использовать информацию о зависимостях

```
rc1c-r1tryw3j1bt9xss0.mdb.yandexcloud.net :) SELECT name, create_table_query, dependencies_table FROM system.tables
WHERE database != 'system'
```

| name             | create_table_query  | dependencies_table                      |
|------------------|---|---|
| events           | CREATE TABLE default.events ...<br>ENGINE = Distributed('{cluster}', default, events_part)            | []                                      |
| events_agg1      | CREATE TABLE default.events_agg1 ...<br>ENGINE = Distributed('{cluster}', default, events_agg1_part)  | []                                      |
| events_agg1_part | CREATE MATERIALIZED VIEW events_agg1_part ...<br>AS SELECT ... FROM events_part                       | []                                      |
| events_agg2      | CREATE TABLE default.events_agg2 ...<br>ENGINE = Distributed('{cluster}', default, events_agg2_part)  | []                                      |
| events_agg2_part | CREATE MATERIALIZED VIEW events_agg2_part ...<br>AS SELECT ... FROM events_part                       | []                                      |
| events_part      | CREATE TABLE events_part ...<br>ENGINE = MergeTree() ...  | ['events_agg1_part','events_agg2_part'] |
| view             | CREATE VIEW default.view ...<br>AS SELECT ... FROM events_agg1 INNER JOIN events_agg2 USING (user_id) | []                                      |

# Зависимости между таблицами в схеме

## ~~Способ 3. Использовать информацию о зависимостях~~

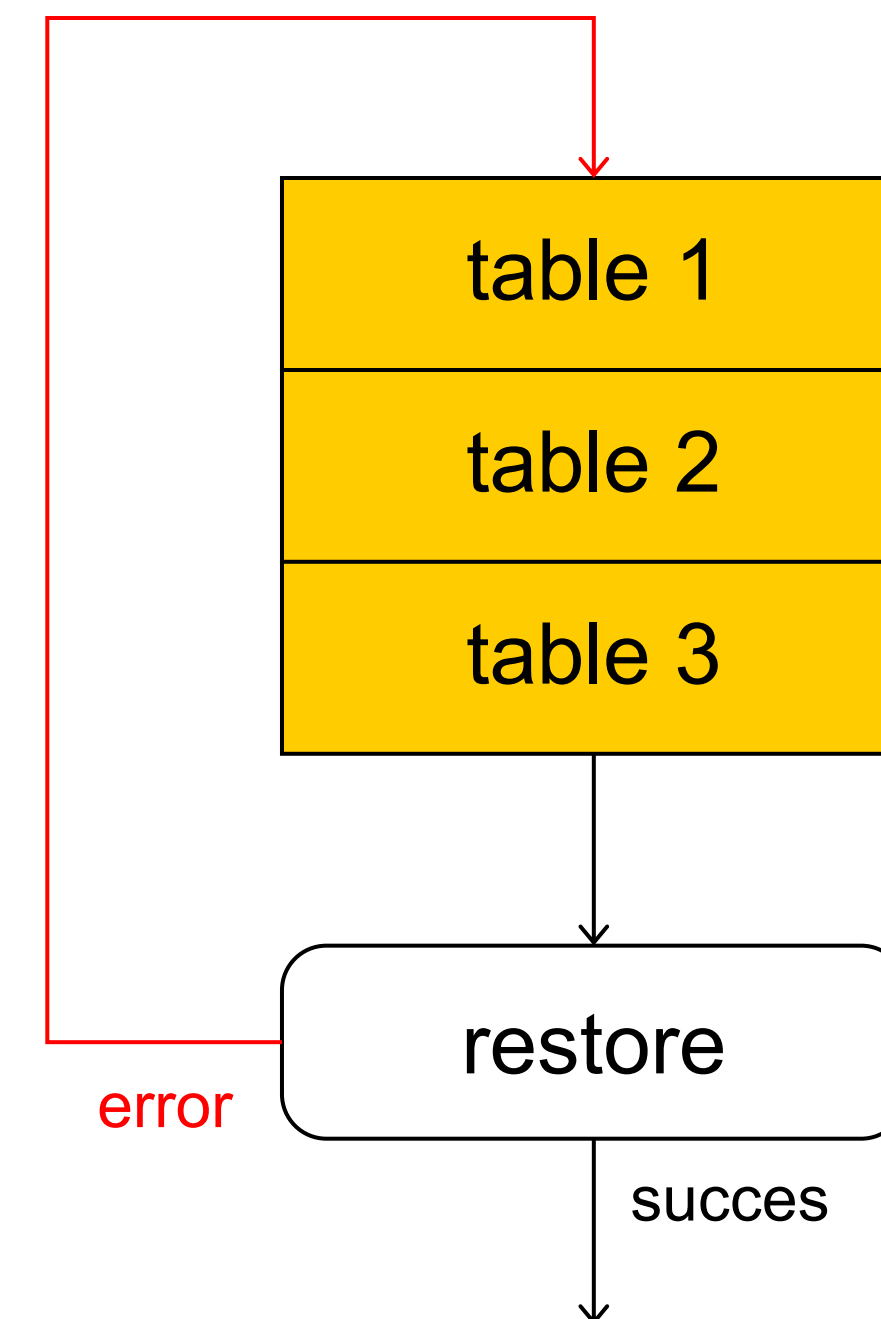
```
rc1c-r1tryw3j1bt9xss0.mdb.yandexcloud.net :) SELECT name, create_table_query, dependencies_table FROM system.tables
WHERE database != 'system'
```

| name             | create_table_query  | dependencies_table                      |
|------------------|---|---|
| events           | CREATE TABLE default.events ...<br>ENGINE = Distributed('{cluster}', default, events_part)            | []                                      |
| events_agg1      | CREATE TABLE default.events_agg1 ...<br>ENGINE = Distributed('{cluster}', default, events_agg1_part)  | []                                      |
| events_agg1_part | CREATE MATERIALIZED VIEW events_agg1_part ...<br>AS SELECT ... FROM events_part                       | []                                      |
| events_agg2      | CREATE TABLE default.events_agg2 ...<br>ENGINE = Distributed('{cluster}', default, events_agg2_part)  | []                                      |
| events_agg2_part | CREATE MATERIALIZED VIEW events_agg2_part ...<br>AS SELECT ... FROM events_part                       | []                                      |
| events_part      | CREATE TABLE events_part ...<br>ENGINE = MergeTree() ...  | ['events_agg1_part','events_agg2_part'] |
| view             | CREATE VIEW default.view ...<br>AS SELECT ... FROM events_agg1 INNER JOIN events_agg2 USING (user_id) | []                                      |

# Зависимости между таблицами в схеме

## Способ 4. Оптимистичный метод

- › Последовательно восстанавливаем таблицы по списку
- › В случае ошибки, повторно добавляем таблицу в начало списка
- › Таблицы отсортированы по типу движка для уменьшения количества повторных попыток MergeTree, Log, ... → Distributed, MaterializedView, View, ...





# Изменение схемы во время бэкапа

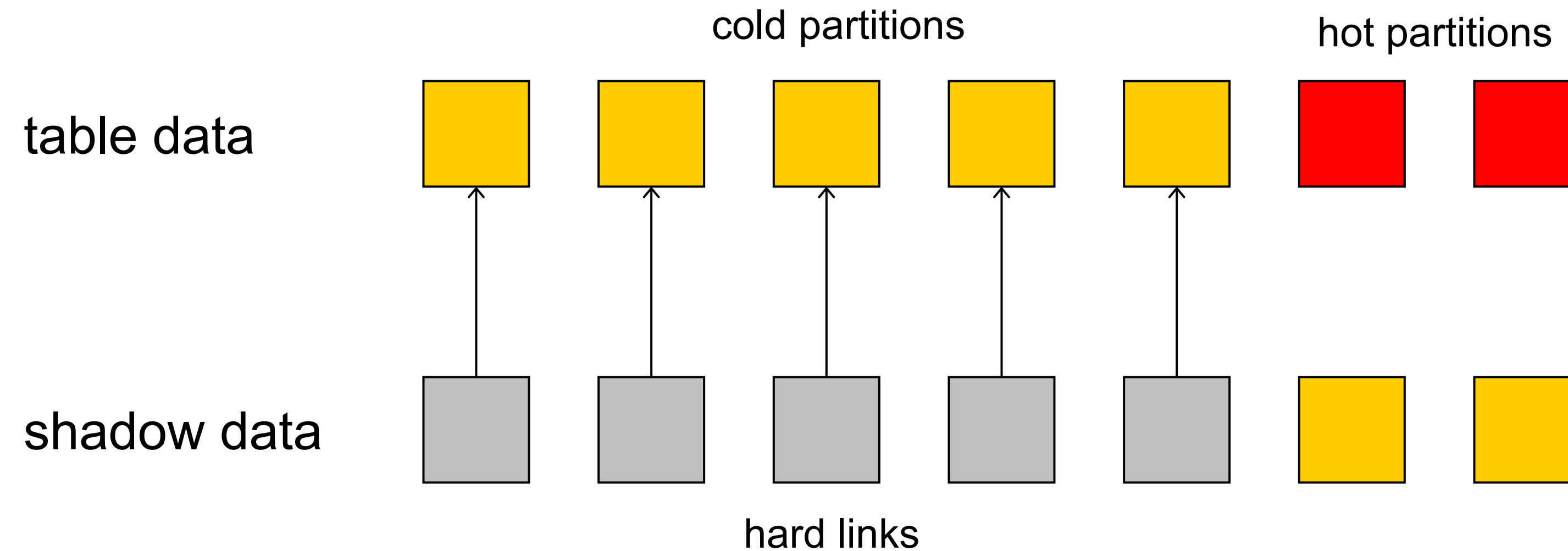
```
# less /var/log/clickhouse-server/clickhouse-server.err.log
2019.08.28 02:59:19.609220 [ 96 ] {00689f2d-adc7-4bfb-b2c2-65c9365a8986} <Error> executeQuery: Code: 60, e.displayText() =
DB::Exception: Table db1.__materi_f2t9stnxxftmd_1566338406_7e0a16 doesn't exist. (from
[2a02:6b8:c02:605:0:1519:d16d:3d0f]:49646) (in query: ALTER TABLE `db1`.`__materi_f2t9stnxxftmd_1566338406_7e0a16` FREEZE),
Stack trace:
```

```
rc1c-jerv0zm7qwavmbzv.mdb.yandexcloud.net :) SELECT query_start_time, substring(query, 1, 60) FROM system.query_log
WHERE (type != 1) AND (query_start_time <= toDateTime('2019.08.28 02:59:19')) and match(query, '^(CREATE|DROP)')
ORDER BY query_start_time DESC LIMIT 20
```

| query_start_time    | substring(query, 1, 60)                                      |
|---------------------|--|
| 2019-08-28 02:59:13 | DROP TABLE IF EXISTS `db1`.`__materi_f2t9stnxxftmd_156633840 |
| 2019-08-28 02:01:48 | DROP TABLE IF EXISTS `db1`.`__materi_tk4ozga4q0dmr_156686040 |
| 2019-08-28 02:00:07 | CREATE TABLE `db1`.`__materi_tk4ozga4q0dmr_1566946805_75ed78 |
| 2019-08-28 01:00:08 | CREATE TABLE `db1`.`__materi_f2t9stnxxftmd_1566943207_211dba |
| 2019-08-28 00:08:12 | DROP TABLE IF EXISTS `db1`.`__materi_68yguezlbmaw4_156685320 |
| 2019-08-28 00:04:55 | DROP TABLE IF EXISTS `db1`.`__materi_8b9434esl4za5_156676680 |
| 2019-08-28 00:04:25 | DROP TABLE IF EXISTS `db1`.`__materi_twdid6r0wvw0q_156676680 |
| 2019-08-28 00:03:58 | DROP TABLE IF EXISTS `db1`.`__materi_k3eof0cw22agi_156685320 |
| 2019-08-28 00:02:41 | DROP TABLE IF EXISTS `db1`.`__materi_jr2all5669lwg_156692797 |
| 2019-08-28 00:00:11 | CREATE TABLE `db1`.`__materi_769uk9crqt4o4_1566939606_0ab263 |

# Потребление места во время бэкапа

Чем больше % горячих данных, тем больше места занимает shadow/



# Потребление места во время бэкапа

Как уменьшить?

- › Удалять файлы из shadow/ по мере загрузки в бэкап хранилище
- › Делать freeze по партициям

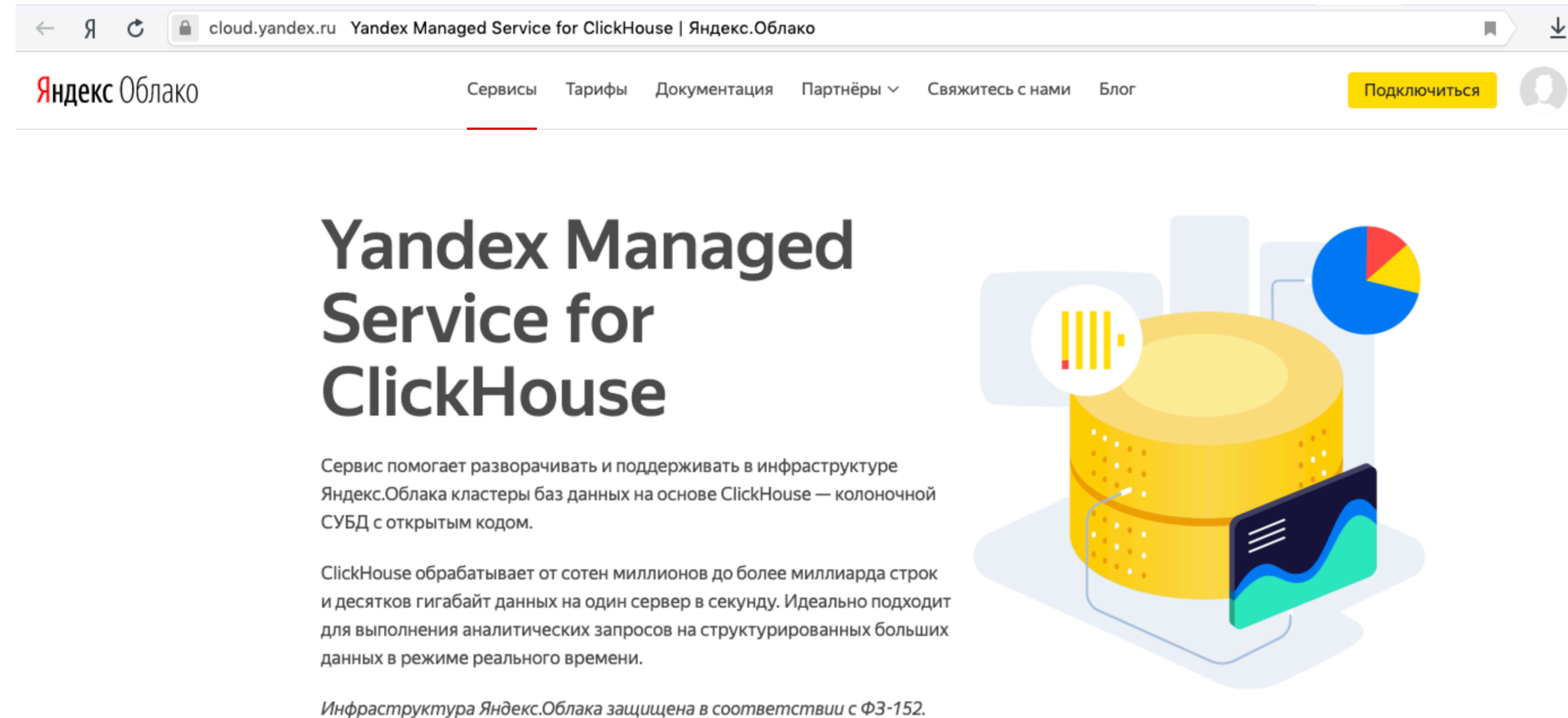
A large yellow arrow pointing from left to right, spanning the entire width of the image. The arrow is composed of a yellow rectangular area on the left and a yellow triangular area on the right, which tapers to a point.

# Планы

# Планы

- › Расширенные правила хранения (retention rules)  
*например, хранить 7 ежедневных + 12 ежемесячных бэкапов*
- › Частичное восстановление
- › Восстановление с преобразованием таблиц  
MergeTree ↔ ReplicatedMergeTree

# ClickHouse в Яндекс Облаке



The screenshot shows the Yandex Cloud website for the Managed Service for ClickHouse. The browser address bar displays "cloud.yandex.ru Yandex Managed Service for ClickHouse | Яндекс.Облако". The navigation bar includes links for "Сервисы", "Тарифы", "Документация", "Партнёры", "Свяжитесь с нами", and "Блог", along with a "Подключиться" button and a user profile icon. The main heading is "Yandex Managed Service for ClickHouse". The text describes the service as helping to deploy and maintain ClickHouse clusters in the Yandex Cloud infrastructure, highlighting its performance and real-time capabilities. An illustration of a database cylinder with charts is positioned to the right. A footer note states that the infrastructure is compliant with F3-152.

Yandex Managed Service for ClickHouse

Сервис помогает разворачивать и поддерживать в инфраструктуре Яндекс.Облака кластеры баз данных на основе ClickHouse — колоночной СУБД с открытым кодом.

ClickHouse обрабатывает от сотен миллионов до более миллиарда строк и десятков гигабайт данных на один сервер в секунду. Идеально подходит для выполнения аналитических запросов на структурированных больших данных в режиме реального времени.

*Инфраструктура Яндекс.Облака защищена в соответствии с ФЗ-152.*

<https://cloud.yandex.ru/services/managed-clickhouse>

Яндекс Облако

**Спасибо!**

Александр Бурмак  
старший разработчик

 alex-burmak@yandex-team.ru

 @AlexanderBurmak