Централизованный сбор данных о работе серверов в БД ClickHouse

- Описание
- Архитектура
    - Описание таблиц данных
        - Данные соединений (таблица ConnectionEvent)
        - Данные о событиях потоков (таблица StreamEvent)
        - Данные CDN(таблица CDNEvent)
        - Данные о метриках потоков (таблица MediaSessionEvents)
        - Данные о нарезке HLS потока (таблица HlsStreamEvents)
        - Данные о сегментах HLS потока (таблица HlsSegmenterEvents)
        - Данные о подписчиках HLS потока (таблица HlsClientEvents)
        - Данные о метриках микшера (таблица MixerEvent)
- Настройка
    - Установка и настройка ClickHouse
        - Требования к серверу
        - Установка ClickHouse на примере CentOS 7
        - Настройка ClickHouse
    - Настройка WCS
        - Остановка сбора данных без перезапуска WCS
        - Изменение адреса сервера ClickHouseбез перезапуска WCS
- Выборки информации из БД

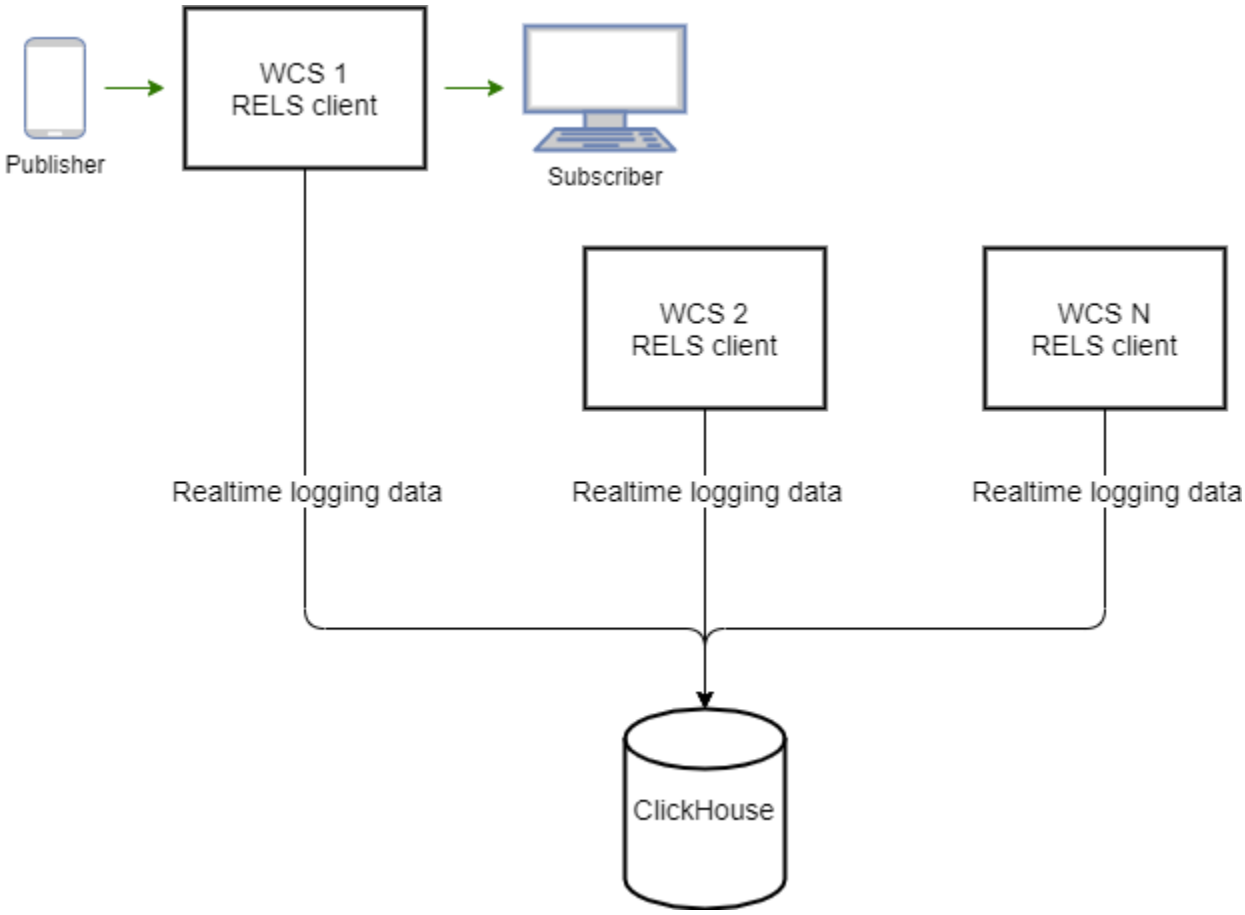
Описание

При управлении большим количеством WCS серверов, для отладки возможных проблем с вещанием потоков, возникает необходимость централизованного сбора данных о потоках, клиентских соединениях и событий CDN. Фактически, необходимо в одной точке собрать информацию, которая пишется в логи каждого сервера. При этом само по себе логирование в промышленной эксплуатации сведено к минимуму, чтобы не давать нагрузку на дисковую подсистему сервера.

Для сбора данных в реальном времени в больших объемах хорошо подходят базы данных временных рядов. На основе одной из таких БД с открытым исходным кодомClickHouse, в сборке5.2.774добавлена система удаленного сбора логов RELS (Remote Event Logging System).

Архитектура

Каждый WCS сервер отправляет данные в ClickHouse независимо, используя JDBC-драйвер и HTTP-соединение. Для оптимизации работы с БД ClickHouse, данные буферизуются и отправляются пачками по времени



Описание таблиц данных

Данные собираются в таблицы ClickHouse, перечисленные ниже. При этом, в таблицу записывается числовой идентификатор события. Для того, чтобы отобразить события в выборках в удобном для чтения виде, каждой таблице сопоставлен словарь текстовых строк, описывающих события.

Данные соединений (таблица ConnectionEvent)

Поле	Тип	Описание
timestamp	UInt64	Метка времени
ip	IPv4	Адрес сервера
sessionId	String	Идентификатор сессии
eventType	UInt64	Идентификатор типа события
eventPayload	String	Содержимое события

Данные о событиях потоков (таблица StreamEvent)

Поле	Тип	Описание
timestamp	UInt64	Метка времени
ip	IPv4	Адрес сервера
sessionId	String	Идентификатор сессии
mediaSessionId	String	Идентификатор медиасессий
streamName	String	Имя потока
eventType	UInt64	Идентификатор типа события

eventPayload	String	Содержимое события

Данные CDN(таблица CDNEvent)

Поле	Тип	Описание
timestamp	UInt64	Метка времени
ip	IPv4	Адрес сервера
nodeId	String	Идентификатор узла (строка IP адреса)
eventType	UInt64	Идентификатор типа события
eventPayload	String	Содержимое события

Данные о метриках потоков (таблица MediaSessionEvents)

В сборке 5.2.1896 добавлен сбор данных о метриках определенных потоков

Поле	Тип	Описание
timestamp	UInt64	Метка времени
ip	IPv4	Адрес сервера
mediaSessionId	String	Идентификатор медиасессий
streamName	String	Имя потока
videoProfileId	UInt32	Идентификатор профиля видео
videoWidth	UInt32	Высота картинки
videoHeight	UInt32	Ширина картинки
videoFrameRate	UInt32	Частота кадров в секунду
videoKframes	UInt64	Количество ключевых кадров (I-frame)
videoPframes	UInt64	Количество промежуточных кадров (P-frame)
videoBframes	UInt64	Количество двунаправленных кадров (B-frames)
videoRate	UInt64	Битрейт видео, бит/с
audioRate	UInt64	Битрейт аудио, бит/с
videoSyncTime	UInt64	Значение синхронизации видео
audioSyncTime	UInt64	Значение синхронизации аудио
videoTimestamp	UInt64	Метка времени из видеопакета
audioTimestamp	UInt64	Метка времени из аудиопакета
lastKeyFrameSyncTime	UInt64	Значение синхронизации видео из последнего ключевого кадра
sendNACK	UInt64	Количество отправленных NACK
recvNACK	UInt64	Количество полученных NACK
videoFramesLost	UInt64	Количество потерянных кадров видео
audioPacketsLost	UInt64	Количество потерянных пакетов аудио

Данные о нарезке HLS потока (таблица HlsStreamEvents)

В сборке 5.2.1917 добавлен сбор данных о событиях определенных HLS потоков

Поле	Тип	Описание
timestamp	UInt64	Метка времени
ip	IPv4	Адрес сервера

severity	UInt8	Уровень события: INFO, WARNING, ERROR
messageType	UInt16	Тип события
streamId	String	Идентификатор HLS потока (имя)
variantName	String	Имя варианта качества
segmentId	String	Идентификатор сегмента
message	String	Описание события в том виде, как залогировано

Данные о сегментах HLS потока (таблица HlsSegmenterEvents)

В сборке 5.2.1917 добавлен сбор данных о сегментах определенных HLS потоков

Поле	Тип	Описание
timestamp	UInt64	Метка времени
ip	IPv4	Адрес сервера
streamId	String	Идентификатор HLS потока (имя)
variantName	String	Имя варианта качества
segmentId	String	Идентификатор сегмента
segmentStartPts	UInt64	Начальный PTS сегмента
videoStartPts	UInt64	Начальный PTS видео дорожки
audioStartPts	UInt64	Начальный PTS аудио дорожки
videoWidth	UInt32	Ширина картинки
videoHeight	UInt32	Высота картинки
videoFrameCount	UInt32	Количество кадров видео
audioPacketCount	UInt32	Количество пакетов аудио
segmentDuration	UInt64	Длительность сегмента, мс
independent	Bool	Независимый сегмент
gap	Bool	GAP сегмент
discontinuity	Bool	DISCONTINUTY сегмент
segmentInterval	UInt64	Интервал между сегментами, мс
partial	Bool	Частичный сегмент
playbackSpeed	Float32	Скорость проигрывания

Данные о подписчиках HLS потока (таблица HlsClientEvents)

В сборке 5.2.1929добавлен сбор данных о подписчиках определенных HLS потоков

Поле	Тип	Описание
creationTime	UInt64	Время подключения подписчика
responseTime	UInt64	Время ответа на запрос подписчика
streamId	String	Идентификатор HLS потока (имя)
variantName	String	Имя варианта качества
uri	String	URI плейлиста
localIp	IPv4	Адрес сервера
remoteIp	IPv4	Адрес клиента
remotePort	UInt32	TCP порт, с которого подключился клиента

userAgent	String	Значение заголовка User-Agent, полученное от клиента
httpStatus	UInt32	Статус ответа на запрос клиента
clientId	UInt64	Идентификатор клиентской сессии

Данные о метриках микшера (таблица MixerEvent)

В сборке 5.2.1923 добавлен сбор данных о метриках определенных микшеров

Поле	Тип	Описание
timestamp	UInt64	Метка времени
mixerMediaSessionId	String	Идентификатор медиасессии микшера
mixerStreamName	String	Имя выходного потока микшера
mediaSessionId	String	Идентификатор входящего потока микшера
streamName	String	Имя входящего потока
mixerAverageTickTimeInMs	Int64	Среднее время одного такта микширования, мс
audioMixerSync	Int64	Значение синхронизации аудио в выходном потоке микшера
videoMixerSync	Int64	Значение синхронизации видео в выходном потоке микшера
nextAudioDataTime	Int64	Время следующего аудио пакета
nextVideoDataTime	Int64	Время следующего видео пакета
audioBuffered	Int64	Количество аудио пакетов в буфере входящего потока
videoBuffered	Int64	Количество видео пакетов в буфере входящего потока
audioDropsCounter	Int64	Количество отброшенных аудио пакетов входящего потока
audioDropsSizeInBytes	Int64	Объем отброшенных данных аудио в байтах
videoDropsCounter	Int64	Количество отброшенных видео пакетов входящего потока
videoDropsSizeInBytes	Int64	Объем отброшенных данных видео в байтах
videoFps	Int64	Частота кадров выходного потока микшера в секунду
audioRate	DOUBLE	Битрейт аудио выходного потока микшера, бит/с
videoRate	DOUBLE	Битрейт видео выходного потока микшера, бит/с
eventType	UInt32	Тип события микшера
eventPayload	String	Описание события микшера

Настройка

Установка и настройка ClickHouse

Требования к серверу

- CPU не менее 4 физических ядер, частотой не менее 3 ГГц, напримерIntel(R) Xeon(R) CPU E3-1246 v3 @ 3.50GHz
- RAM не менее 32 Гб
- HDD не менее 2 Тб

Установка ClickHouse на примере CentOS 7

1. Создайте файл репозитория altinity_clickhouse.repo в каталоге /etc/yum.repos.d

```
sudo cat <<EOF > /etc/yum.repos.d/altinity_clickhouse.repo
[altinity_clickhouse]
name=altinity_clickhouse
baseurl=https://packagecloud.io/altinity/clickhouse/el/7/$basearch
repo_gpgcheck=1
gpgcheck=0
enabled=1
gpgkey=https://packagecloud.io/altinity/clickhouse/gpgkey
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
metadata_expire=300

[altinity_clickhouse-source]
name=altinity_clickhouse-source
baseurl=https://packagecloud.io/altinity/clickhouse/el/7/SRPMS
repo_gpgcheck=1
gpgcheck=0
enabled=1
gpgkey=https://packagecloud.io/altinity/clickhouse/gpgkey
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
metadata_expire=300
EOF
```

2. Подключите репозиторий

```
sudo yum -q makecache -y --enablerepo='altinity_clickhouse'
```

3. Установите ClickHouse

```
sudo yum install -y clickhouse-server clickhouse-client
```

4. Запустите ClickHouse

```
systemctl start clickhouse-server
```

Настройка ClickHouse

1. Для того, чтобы прослушивать входящие запросы на всех интерфесах сервера, раскомментируйте строку в файле/etc/clickhouse-server /config.xml

```
    <listen_host>::</listen_host>
```

2. Для того, чтобы создать пользователя, укажите для пользователя default в файле/etc/clickhouse-server/users.xml параметр

```
            <access_management>1</access_management>
```

3. Перезапустите ClickHouse

```
systemctl restart clickhouse-server
```

4. Создайте базу данных wcs и таблицы в ней

```
cat wcs_clickhouse.sql | clickhouse-client -mn
```

wcs_clickhouse.sql

```
CREATE DATABASE IF NOT EXISTS wcs;
```

```
DROP TABLE IF EXISTS wcs.StreamEvent;

DROP TABLE IF EXISTS wcs.ConnectionEvent;

DROP TABLE IF EXISTS wcs.CDNEvent;

DROP TABLE IF EXISTS wcs.StreamEventTypes;

DROP TABLE IF EXISTS wcs.ConnectionEventTypes;

DROP TABLE IF EXISTS wcs.CDNEventTypes;

DROP TABLE IF EXISTS wcs.MediaSessionEvents;

DROP TABLE IF EXISTS wcs.HlsStreamEvents;

DROP TABLE IF EXISTS wcs.HlsSegmenterEvents;

DROP TABLE IF EXISTS wcs.HlsStreamEventSeverity;

DROP TABLE IF EXISTS wcs.HlsStreamEventType;

DROP TABLE IF EXISTS wcs.HlsClientEvents;

DROP TABLE IF EXISTS wcs.MixerEvent;

DROP TABLE IF EXISTS wcs.MixerEventTypes;

DROP DICTIONARY IF EXISTS wcs.DictionaryStreamEvents;

DROP DICTIONARY IF EXISTS wcs.DictionaryConnectionEvents;

DROP DICTIONARY IF EXISTS wcs.DictionaryCDNEvents;

DROP DICTIONARY IF EXISTS wcs.DictionaryHlsStreamEventType;

DROP DICTIONARY IF EXISTS wcs.DictionaryHlsStreamEventSeverity;

DROP DICTIONARY IF EXISTS wcs.DictionaryMixerEvents;

CREATE TABLE wcs.ConnectionEventTypes
(
    `id` UInt32,
    `type` String
)
ENGINE = MergeTree()
ORDER BY id
SETTINGS index_granularity = 8192;

INSERT INTO wcs.ConnectionEventTypes VALUES (0, 'CONNECTED'), (1, 'DISCONNECTED');

CREATE TABLE wcs.StreamEventTypes
(
    `id` UInt32,
    `type` String
)
ENGINE = MergeTree()
ORDER BY id
SETTINGS index_granularity = 8192;

INSERT INTO wcs.StreamEventTypes VALUES (0,'CREATED'),(1,'LOCAL_SDP_CREATED'),(2,'REMOTE_SDP_RECEIVED'),
(3,'ICE_STARTED'),(4,'ICE_COMPLETE'),(5,'DTLS_STARTED'),(6,'DTLS_COMPLETE'),(7,'INITIALIZED'),(8,'DISPOSING'),
(9,'DISPOSED'),(10,'AUDIO_RECEIVED'),(11,'VIDEO_RECEIVED'),(12,'VIDEO_KFRAME_RECEIVED'),
(13,'AUDIO_RTCP_RECEIVED'),(14,'VIDEO_RTCP_RECEIVED'),(15,'RESOLUTION_RECEIVED'),(16,'VIDEO_ENCODER_CREATED'),
(17,'AUDIO_ENCODER_CREATED'),(18,'VIDEO_ENCODER_DISPOSED'),(19,'AUDIO_ENCODER_DISPOSED'),(20,'TERMINATED'),
(21,'AUDIO_SENT'),(22,'VIDEO_SENT'),(23,'VIDEO_JITTER_BUFFER_STALL'),(24,'SENT_PLI'),(25,'RECEIVED_PLI'),
(26,'SYNC_BUFFER_FULL'),(27,'SYNC_FORCE_FAILED'),(28,'SYNC_SHIFT'),(29,'SYNC_DEVIATION'),(30,'VIDEO_STATS'),
(31,'RECORD');

CREATE TABLE wcs.CDNEventTypes
```

```
(
    `id` UInt32,
    `type` String
)
ENGINE = MergeTree()
ORDER BY id
SETTINGS index_granularity = 8192;

INSERT INTO wcs.CDNEventTypes VALUES (0, 'STATE'), (1, 'CDN_STATE'), (2, 'VERSION'), (3, 'ACL_REFRESH'), (4,
'ACL_UPDATE');

CREATE DICTIONARY wcs.DictionaryStreamEvents (
    `id` UInt16,
    `type` String DEFAULT ''
)
PRIMARY KEY id
SOURCE(CLICKHOUSE(
 host 'localhost'
 port 9000
 user 'default'
 password ''
 db 'wcs'
 table 'StreamEventTypes'
))
LAYOUT(FLAT())
LIFETIME(300);

CREATE DICTIONARY wcs.DictionaryConnectionEvents (
    `id` UInt16,
    `type` String DEFAULT ''
)
PRIMARY KEY id
SOURCE(CLICKHOUSE(
 host 'localhost'
 port 9000
 user 'default'
 password ''
 db 'wcs'
 table 'ConnectionEventTypes'
))
LAYOUT(FLAT())
LIFETIME(300);

CREATE DICTIONARY wcs.DictionaryCDNEvents (
    `id` UInt16,
    `type` String DEFAULT ''
)
PRIMARY KEY id
SOURCE(CLICKHOUSE(
 host 'localhost'
 port 9000
 user 'default'
 password ''
 db 'wcs'
 table 'CDNEventTypes'
))
LAYOUT(FLAT())
LIFETIME(300);

CREATE TABLE wcs.StreamEvent
(
    `timestamp` UInt64,
    `ip` IPv4,
    `sessionId` String,
    `mediaSessionId` String,
    `streamName` String,
    `eventType` UInt64,
    `eventPayload` String
)
ENGINE = MergeTree()
ORDER BY (sessionId, mediaSessionId, streamName)
```

```
SETTINGS index_granularity = 8192;

CREATE TABLE wcs.ConnectionEvent
(
    `timestamp` UInt64,
    `ip` IPv4,
    `sessionId` String,
    `eventType` UInt64,
    `eventPayload` String
)
ENGINE = MergeTree()
ORDER BY (timestamp, sessionId)
SETTINGS index_granularity = 8192;

CREATE TABLE wcs.CDNEvent
(
    `timestamp` UInt64,
    `ip` IPv4,
    `nodeId` String,
    `eventType` UInt64,
    `eventPayload` String
)
ENGINE = MergeTree()
ORDER BY (nodeId, eventType)
SETTINGS index_granularity = 8192;

CREATE TABLE wcs.MediaSessionEvents
(
    `timestamp` UInt64,
    `ip` IPv4,
    `mediaSessionId` String,
    `streamName` String,
    `videoProfileId` UInt32,
    `videoWidth` UInt32,
    `videoHeight` UInt32,
    `videoFrameRate` UInt32,
    `videoKframes` UInt64,
    `videoPframes` UInt64,
    `videoBframes` UInt64,
    `videoRate` UInt64,
    `audioRate` UInt64,
    `videoSyncTime` UInt64,
    `audioSyncTime` UInt64,
    `videoTimestamp` UInt64,
    `audioTimestamp` UInt64,
    `lastKeyFrameSyncTime` UInt64,
    `sendNACK` UInt64,
    `recvNACK` UInt64,
    `videoFramesLost` UInt64,
    `audioPacketsLost` UInt64
)
ENGINE = MergeTree()
ORDER BY (mediaSessionId, streamName)
SETTINGS index_granularity = 8192;

CREATE TABLE wcs.HlsSegmenterEvents
(
    `timestamp` UInt64,
    `ip` IPv4,
    `streamId` String,
    `variantName` String,
    `segmentId` String,
    `segmentStartPts` UInt64,
    `videoStartPts` UInt64,
    `audioStartPts` UInt64,
    `videoWidth` UInt32,
    `videoHeight` UInt32,
    `videoFrameCount` UInt32,
    `audioPacketCount` UInt32,
    `segmentDuration` UInt64,
    `independent` Bool,
```

```
    `gap` Bool,
    `discontinuity` Bool,
    `segmentInterval` UInt64,
    `partial` Bool,
    `playbackSpeed` Float32
)
ENGINE = MergeTree()
ORDER BY (streamId)
SETTINGS index_granularity = 8192;

CREATE TABLE wcs.HlsStreamEventSeverity
(
    `id` UInt8,
    `type` String
)
ENGINE = MergeTree()
ORDER BY id
SETTINGS index_granularity = 8192;

INSERT INTO wcs.HlsStreamEventSeverity VALUES (0, 'INFO'), (1, 'WARNING'), (2, 'ERROR');

CREATE TABLE wcs.HlsStreamEventType
(
    `id` UInt16,
    `type` String
)
ENGINE = MergeTree()
ORDER BY id
SETTINGS index_granularity = 8192;

INSERT INTO wcs.HlsStreamEventType
VALUES (0, 'PLAYBACK_SPEED'), (1, 'FPS_CHANGED'), (2, 'GAP'), (3, 'RESOLUTION_CHANGED'), (4, 'DISCONTINUITY'),
(5, 'TASK_SKIPPED'), (6, 'NO_KYE_FRAME'), (7, 'NO_VIDEO'), (8, 'NO_AUDIO'), (9, 'SEGMENT_INTERVAL'), (10,
'OTHER');

CREATE DICTIONARY wcs.DictionaryHlsStreamEventSeverity
(
   `id` UInt8,
   `type` String DEFAULT ''
)
PRIMARY KEY id
SOURCE(CLICKHOUSE(
 host 'localhost'
 port 9000
 user 'default'
 password ''
 db 'wcs'
 table 'HlsStreamEventSeverity'
))
LAYOUT(FLAT())
LIFETIME(300);

CREATE DICTIONARY wcs.DictionaryHlsStreamEventType
(
   `id` UInt16,
   `type` String DEFAULT ''
)
PRIMARY KEY id
SOURCE(CLICKHOUSE(
 host 'localhost'
 port 9000
 user 'default'
 password ''
 db 'wcs'
 table 'HlsStreamEventType'
))
LAYOUT(FLAT())
LIFETIME(300);

CREATE TABLE wcs.HlsStreamEvents
(
```

```
    `timestamp` UInt64,
    `ip` IPv4,
    `severity` UInt8,
    `messageType` UInt16,
    `streamId` String,
    `variantName` String,
    `segmentId` String,
    `message` String
)
ENGINE = MergeTree()
ORDER BY (streamId)
SETTINGS index_granularity = 8192;

CREATE TABLE wcs.HlsClientEvents
(
    `creationTime` UInt64,
    `responseTime` UInt64,
    `streamId` String,
    `variantName` String,
    `uri` String,
    `localIp` IPv4,
    `remoteIp` IPv4,
    `remotePort` UInt32,
    `userAgent` String,
    `httpStatus` UInt32,
    `clientId` UInt64
)
ENGINE = MergeTree()
ORDER BY (creationTime)
SETTINGS index_granularity = 8192;

CREATE TABLE wcs.MixerEvent
(
    `timestamp` UInt64,
    `mixerMediaSessionId` String,
    `mixerStreamName` String,
    `mediaSessionId` String,
    `streamName` String,
    `mixerAverageTickTimeInMs` Int64,
    `audioMixerSync` Int64,
    `videoMixerSync` Int64,
    `nextAudioDataTime` Int64,
    `nextVideoDataTime` Int64,
    `audioBuffered` Int64,
    `videoBuffered` Int64,
    `audioDropsCounter` Int64,
    `audioDropsSizeInBytes` Int64,
    `videoDropsCounter` Int64,
    `videoDropsSizeInBytes` Int64,
    `videoFps` Int64,
    `audioRate` DOUBLE,
    `videoRate` DOUBLE,
    `eventType` UInt32,
    `eventPayload` String
)
ENGINE = MergeTree()
ORDER BY timestamp
SETTINGS index_granularity = 8192;

CREATE TABLE wcs.MixerEventTypes
(
    `id` UInt32,
    `type` String
)
ENGINE = MergeTree()
ORDER BY id
SETTINGS index_granularity = 8192;

INSERT INTO wcs.MixerEventTypes VALUES (0, 'nullEvent'), (1, 'dropBallastAudio'), (2, 'dropBallastVideo'), (3,
'audioNotBuffered'), (4, 'videoNotBuffered'), (5, 'audioBufferExhausted'), (6, 'videoBufferExhausted'), (7,
'alignStreamFailed'), (8, 'alignStreamDropAudio'), (9, 'alignStreamDropVideo'), (10, 'rateOutOfBoundsAudio'),
```

```
(11, 'rateOutOfBoundsVideo');

CREATE DICTIONARY wcs.DictionaryMixerEvents (
    `id` UInt16,
    `type` String DEFAULT ''
)
PRIMARY KEY id
SOURCE(CLICKHOUSE(
 host 'localhost'
 port 9000
 user 'default'
 password ''
 db 'wcs'
 table 'MixerEventTypes'
))
LAYOUT(FLAT())
LIFETIME(300);
```

5.Создайте пользователя wcs и дайте ему права на таблицы в базе данных wcs

```
cat wcs_clickhouse_users.sql | clickhouse-client -mn
```

wcs_clickhouse_users.sql

```
CREATE USER IF NOT EXISTS wcs IDENTIFIED BY 'wcs';
GRANT ALL ON wcs.* TO wcs WITH GRANT OPTION;
```

6.Отключите управление пользователями для пользователя default, указав в файле/etc/clickhouse-server/users.xml параметр

```
            <access_management>0</access_management>
```

7. ПерезапуститеClickHouse

```
systemctl restart clickhouse-server
```

Настройка WCS

Сбор данных в БД ClickHouse включается настройкой

```
rels_enabled=true
```

Адрес сервера ClickHouse и базы данных задается настройкой

```
rels_database_address=jdbc:clickhouse://clickhouseserver:8123/wcs?user=wcs&password=wcs
```

Остановка сбора данных без перезапуска WCS

При необходимости, передача данных с конкретного WCS сервера в ClickHouse может быть остановлена без перезапуска WCS. Для этого:

1. Отключите сбор данных в настройках сервера

```
rels_enabled=false
```

2. Перезагрузите настройки сервера из интерфейса командной строки

```
reload node-settings
```

Изменение адреса сервера ClickHouseбез перезапуска WCS

Адрес сервера ClickHouse может быть изменен без перезапуска WCS. Для этого:

1. Измените адрес в настройках сервера

```
rels_database_address=jdbc:clickhouse://newclickhouseserver:8123/wcs?user=wcs&password=wcs
```

2. Отключите сбор данных в настройках сервера

```
rels_enabled=false
```

3. Перезагрузите настройки сервера из интерфейса командной строки

```
reload node-settings
```

4. Включите сбор данных в настройках сервера

```
rels_enabled=true
```

5. Перезагрузите настройки сервера из интерфейса командной строки

```
reload node-settings
```

Выборки информации из БД

Выборки информации из БД производятся при помощи SQL запросов в клиенте ClickHouse

```
select timestamp,ip,sessionId,mediaSessionId,streamName,dictGetString('wcs.DictionaryStreamEvents','type',
eventType) as eventType from wcs.StreamEvent where streamName = 'test'
```

```
select timestamp,ip,sessionId,dictGetString('wcs.DictionaryConnectionEvents','type', eventType) as eventType
from wcs.ConnectionEvent
```

```
select timestamp,ip,nodeId,dictGetString('wcs.DictionaryCDNEvents','type', eventType) as eventType,eventPayload
from wcs.CDNEvent
```