ООО «Астра Консалтинг»

Астра ИС МД (Инфраструктурные Сервисы)

Модуль Astra ClickHouse

ИНСТРУКЦИЯ ПО УСТАНОВКЕ ЭКЗЕМПЛЯРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ПРЕДОСТАВЛЕННОГО ДЛЯ ПРОВЕДЕНИЯ ЭКСПЕРТНОЙ ПРОВЕРКИ

Москва, 2026

Содержание

1 Область применения

Настоящий документ содержит описание функциональных характеристик программного обеспечения Астра ИС МД, модуль Astra ClickHouse, развернутого в архитектуре с 3-мя серверными узлами и 3-мя координаторами с использованием Astra ClickHouse Keeper на тех же серверах.

Документ предназначен для системных администраторов и разработчиков, осуществляющих эксплуатацию и обслуживание системы управления базами данных Astra ClickHouse.

Инструкция описывает установку модуля Astra ClickHouse кластера из 3-х узлов с использованием ClickHouse Keeper в качестве координатора репликации данных и выполнения распределённых запросов.

https://clickhouse.com/docs/guides/sre/keeper/clickhouse-keeper

2 Установка системы

2.1 Технические требования к оборудованию

Для локальной инсталляции требуется виртуальный сервер (ВС) с минимальными техническими характеристиками, представленными в Таблица 1.

Таблица 1 — Минимальные требования к ВС

Вид ресурса	Ед. измерения	Требуемый объем
Частота процессора	ГГц	2.3
vCPU количество процессоров и ядер	шт.	2
Объем оперативной памяти	Гб	16
Объем жесткого диска	Гб	20
Сетевой интерфейс	шт.	1
Скорость сетевого интерфейса	Гб/с	1

2.2 Минимальные требования к сторонним компонентам

Для установки Системы на стенде используется система управления конфигурациями ansible.

Требуемые компоненты:

- ansible версии 2.9 или выше
- python 3.8 или выше

Для Ubuntu/Debian:

```
sudo apt update
sudo apt-get install ansible python -y
```

3 Архитектура системы

3.1 Характеристики системы

Таблица 2 — Характеристики демонстрационного стенда

Параметр	Значение
Astra ClickHouse версия	25.8.18.1
Координаторы	Astra ClickHouse Keeper
Кол-во узлов	3
Порты HTTPS	8443
Порты TCP	9440
Порт метрик Prometheus	9363
Имя пользователя	default
Пароль	P@ssw0rd

3.2 Настройка файла инвентаря

Создайте файл **inv** с именами хостов ваших узлов:

```
[clickhouse]
host-01.internal
host-02.internal
host-03.internal

[zookeepers]
host-01.internal
host-02.internal
host-03.internal
```

3.3 Настройка плейбука

Основной плейбук для запуска установки находится в файле `clickhouse.yml`.

Соответствующий целевой платформе дистрибутив ClickHouse или ссылка на него должны располагаться в каталоге **roles/clickhouse/files/distrib/**. Установка будет произведена стандартным для этой платформы пакетным менеджером с использованием указанного каталога.

3.4 Настройка переменных

Файл **group_vars/all/ansible.yml** содержит настройки удалённых подключений к хостам кластера. Пользователь, указанный в параметре **ansible_user**, должен иметь **sudo** права на удалённых хостах с беспарольным выполнением. Жирным шрифтом указаны поля, которые могут быть изменены.

ansible_connection: ssh

ansible_user: remote_username

ansible_become: yes

ansible_ssh_private_key_file: ~/.ssh/id_rsa

Файл **roles/clickhouse/defaults/main.yml** содержит основные переменные для настройки кластера:

ch_default_password: "P@ssw0rd" — пароль административного пользователя с именем default.

ch_dataDir: "/var/lib/clickhouse" — путь к основному каталогу данных ClickHouse.

ch_with_ssl: "yes" — если этот параметр имеет значение "yes", то на локальном хосте должны быть подготовлены файлы ssl сертификатов и ключей в каталоге roles/clickhouse/files/certs/ со следующими правилами именования и содержимым.

hostname.crt — для каждого *hostname* из инвентарного файла inv; содержит сертификат для хоста *hostname* в формате pem.

hostname.key — для каждого *hostname* из инвентарного файла inv; содержит приватный ключ для хоста *hostname* в формате pem.

truststore.pem — содержит CA сертификаты в формате pem.

Остальные параметры обычно не меняются.

zk_group_name: "zookeepers" — имя группы хостов Zookeeper / ClickHouse Keeper в инвентарном файле.

ch_group_name: "clickhouse" — имя группы хостов серверов ClickHouse в инвентарном файле.

ch_cfgDir: "/etc/clickhouse-server" — путь к основному каталогу конфигурационных файлов ClickHouse.

ch_logDir: "/var/log/clickhouse-server" — путь к каталогу с журналами.

ch_certDir: "{{ ch_cfgDir ~ '/certs' }}" — путь к каталогу с сертификатами и ключами.

ch_zk_identity: — если переменная определена, то указанные имя и пароль используются для использования аутентификационной схемы digest для соединений с кластером Zookeeper.

ch_with_zk_ssl: "yes" — использовать ли ssl соединения с кластером Zookeeper.

ch_cert_file: "{{ ch_cfgDir ~ '/node.crt' }}" — путь к файлу ssl сертификата хоста ClickHouse.

ch_keystore_file: "{{ ch_certDir ~ '/node.key' }}" — путь к файлу ssl ключа хоста ClickHouse.

ch_truststore_file: "{{ ch_certDir ~ '/truststore.pem' }}" — путь к файлу CA сертификатов.

ch_service_name: "clickhouse-server.service" — имя файла systemd сервиса ClickHouse.

ch_use_keeper: "yes/no" — использовать ClickHouse Keeper или Zookeeper в качестве координатора.

Если используется значение "**yes**", то каждый хост из группы **zookeepers** инвентарного файла должен входить в группу серверов ClickHouse **clickhouse**. ClickHouse Keeper будет запущен на хостах группы **clickhouse** в дополнение к серверу ClickHouse на хостах группы **zookeepers**.

Если используется значение "**no**", то на хостах группы **zookeepers** должен быть сконфигурирован и запущен кластер Zookeeper отдельно.

ch_default_cluster: "s1r3" — имя кластера ClickHouse по умолчанию. Используется для удобства. Это значение получает макрос **{cluster}**, и он же используется в параметре **default_replica_path** ClickHouse. Обычно устанавливается в значение имени одного из кластеров, в которых наиболее часто создаются реплицируемые таблицы для того, чтобы можно было создавать реплицируемые таблицы без указания параметров движка ReplicatedMergeTree. Например:

```
CREATE TABLE ... ON CLUSTER '{cluster}' ... ENGINE =
ReplicatedMergeTree()
```

ch_clusters: — описание кластеров ClickHouse. В одном из **name** должно быть значение, указанное в **ch_default_cluster**. Имена хостов каждого кластера указываются с помощью их индексов в группе серверов ClickHouse инвентарного файла.

Пример 2-х кластеров:

- s3r1: 3 раздела (shards) по 1 реплике
- s1r3: 1 раздел (shards) по 3 реплики (это же имя указано в **ch_default_cluster**)

```
ch_clusters:
  - name: s3r1
    shards:
      - replicas:
        - "{{ groups[ch_group_name][0] }}"
      - replicas:
        - "{{ groups[ch_group_name][1] }}"
      - replicas:
        - "{{ groups[ch_group_name][2] }}"
  - name: "{{ ch_default_cluster }}"
    shards:
      - replicas:
        - "{{ groups[ch_group_name][0] }}"
        - "{{ groups[ch_group_name][1] }}"
        - "{{ groups[ch_group_name][2] }}"
```

4 Установка и запуск

4.1 Шаг 1: Запуск плейбука в режиме установки

Для запуска установки выполните команду:

```
ansible-playbook clickhouse.yml -e "ch_config_only=no"
```

4.2 Шаг 2: Проверка статуса

После успешного завершения плейбука убедитесь, что все узлы работают корректно, проверив статус службы Astra ClickHouse:

```
ansible -m shell -a "systemctl status clickhouse-server.service" clickhouse
```

5 Доступ к сервисам

5.1 Точки доступа

Таблица 3 — Точки доступа к сервисам

Сервис	URL	Аутентификация
arangosh	http+ssl://localhost:8529	требуется (root / "") + корневой сертификат
curl	https://localhost:8529/...	HTTP-авторизация (root / "") + корневой сертификат
Web	https://localhost:8529/	требуется (root / "") + корневой сертификат в браузере

5.1.1 Доступ через Web интерфейс

В браузер должен быть загружен корневой сертификат, указанный в переменной **ch_truststore_file** ansible роли, которая использовалась для установки.

1. Откройте в браузере: *https://localhost:8443/play/*
2. Введите учетные данные справа вверху:
   - Логин: default
   - Пароль: P@ssw0rd
1. Выполните тестовый запрос с текстом ниже, нажав кнопку **Run**:

```
select version()
```

5.1.2 Подключение с clickhouse-client

Утилита использует в качестве одного из параметров файл конфигурации, который по умолчанию имеет имя **~/.clickhouse-client/config.xml**. Удобно создать его с содержимым ниже:

```
<config>
  <user>default</user>
  <password>P@ssw0rd</password>
  <secure>true</secure>
  <openSSL>
    <client>
      <caConfig>/etc/security/certs/truststore.pem</caConfig>
      <invalidCertificateHandler>
        <name>RejectCertificateHandler</name>
          </invalidCertificateHandler>
    </client>
  </openSSL>
</config>
```

И алиас:

```
$ alias cc='clickhouse-client -h localhost -f
PrettyCompactMonoBlock'
```

Тогда команда подключения будет выглядеть так:

```
$ cc -q "select version()"
   ┌─version()─┐
1. │ 25.8.18.1 │
   └───────────┘
```

5.1.3 Подключение с curl

Удобно создать алиас для curl:

```
$ alias curl_ch="curl https://$(hostname -f):8443 \
--cacert /etc/security/certs/truststore.pem \
--user "default:P@ssw0rd" --data-binary @-"
```

Тогда команда с использованием алиаса будет выглядеть так:

```
$ echo "select version()" | curl_ch
25.8.18.1
```

6 Проверка работы

6.1 Базовая проверка

Проверка подключения с **clickhouse-server**:

```
$ cc -q "select version()"
   ┌─version()─┐
1. │ 25.8.18.1 │
   └───────────┘
```

Проверка подключения с **curl**:

```
$ echo "select version()" | curl_ch
25.8.18.1
```

6.2 Тестирование записи и чтения

Создадим тестовую таблицу trips в базе данных default.

В текущем каталоге создадим файл trips_create.sql с содержимым ниже:

```
CREATE TABLE trips on cluster '{cluster}'
(
    `trip_id` UInt32,
    `vendor_id` Enum8('1' = 1, '2' = 2, '3' = 3, '4' = 4, 'CMT' =
5, 'VTS' = 6, 'DDS' = 7, 'B02512' = 10, 'B02598' = 11, 'B02617' =
12, 'B02682' = 13, 'B02764' = 14, '' = 15),
    `pickup_date` Date,
    `pickup_datetime` DateTime,
    `dropoff_date` Date,
    `dropoff_datetime` DateTime,
    `store_and_fwd_flag` UInt8,
    `rate_code_id` UInt8,
    `pickup_longitude` Float64,
    `pickup_latitude` Float64,
    `dropoff_longitude` Float64,
    `dropoff_latitude` Float64,
    `passenger_count` UInt8,
    `trip_distance` Float64,
    `fare_amount` Float32,
    `extra` Float32,
    `mta_tax` Float32,
    `tip_amount` Float32,
    `tolls_amount` Float32,
    `ehail_fee` Float32,
    `improvement_surcharge` Float32,
    `total_amount` Float32,
    `payment_type` Enum8('UNK' = 0, 'CSH' = 1, 'CRE' = 2, 'NOC' =
3, 'DIS' = 4),
    `trip_type` UInt8,
    `pickup` FixedString(25),
```

```
    `dropoff` FixedString(25),
    `cab_type` Enum8('yellow' = 1, 'green' = 2, 'uber' = 3),
    `pickup_nyct2010_gid` Int8,
    `pickup_ctlabel` Float32,
    `pickup_borocode` Int8,
    `pickup_ct2010` String,
    `pickup_boroct2010` String,
    `pickup_cdeligibil` String,
    `pickup_ntacode` FixedString(4),
    `pickup_ntaname` String,
    `pickup_puma` UInt16,
    `dropoff_nyct2010_gid` UInt8,
    `dropoff_ctlabel` Float32,
    `dropoff_borocode` UInt8,
    `dropoff_ct2010` String,
    `dropoff_boroct2010` String,
    `dropoff_cdeligibil` String,
    `dropoff_ntacode` FixedString(4),
    `dropoff_ntaname` String,
    `dropoff_puma` UInt16
)
ENGINE = ReplicatedMergeTree
PARTITION BY toYYYYMM(pickup_date)
ORDER BY pickup_datetime;
```

Выполним его:

```
$ cc --queries-file trips_create.sql
```

Загрузим 2 файла с данными в текущий каталог:

```
$ curl -LO https://datasets-documentation.s3.eu-west-
3.amazonaws.com/nyc-taxi/trips_1.gz
$ curl -LO https://datasets-documentation.s3.eu-west-
3.amazonaws.com/nyc-taxi/trips_2.gz
```

Создадим файл trips_insert.sql с командой вставки данных в таблицу с содержимым:

```
INSERT INTO trips
SELECT *
FROM input("
    `trip_id` UInt32,
    `vendor_id` Enum8('1' = 1, '2' = 2, '3' = 3, '4' = 4, 'CMT' =
5, 'VTS' = 6, 'DDS' = 7, 'B02512' = 10, 'B02598' = 11, 'B02617' =
12, 'B02682' = 13, 'B02764' = 14, '' = 15),
    `pickup_date` Date,
    `pickup_datetime` DateTime,
    `dropoff_date` Date,
    `dropoff_datetime` DateTime,
    `store_and_fwd_flag` UInt8,
    `rate_code_id` UInt8,
    `pickup_longitude` Float64,
    `pickup_latitude` Float64,
```

```
    `dropoff_longitude` Float64,
    `dropoff_latitude` Float64,
    `passenger_count` UInt8,
    `trip_distance` Float64,
    `fare_amount` Float32,
    `extra` Float32,
    `mta_tax` Float32,
    `tip_amount` Float32,
    `tolls_amount` Float32,
    `ehail_fee` Float32,
    `improvement_surcharge` Float32,
    `total_amount` Float32,
    `payment_type` Enum8('UNK' = 0, 'CSH' = 1, 'CRE' = 2, 'NOC' =
3, 'DIS' = 4),
    `trip_type` UInt8,
    `pickup` FixedString(25),
    `dropoff` FixedString(25),
    `cab_type` Enum8('yellow' = 1, 'green' = 2, 'uber' = 3),
    `pickup_nyct2010_gid` Int8,
    `pickup_ctlabel` Float32,
    `pickup_borocode` Int8,
    `pickup_ct2010` String,
    `pickup_boroct2010` String,
    `pickup_cdeligibil` String,
    `pickup_ntacode` FixedString(4),
    `pickup_ntaname` String,
    `pickup_puma` UInt16,
    `dropoff_nyct2010_gid` UInt8,
    `dropoff_ctlabel` Float32,
    `dropoff_borocode` UInt8,
    `dropoff_ct2010` String,
    `dropoff_boroct2010` String,
    `dropoff_cdeligibil` String,
    `dropoff_ntacode` FixedString(4),
    `dropoff_ntaname` String,
    `dropoff_puma` UInt16
")
FORMAT TabSeparatedWithNames
SETTINGS input_format_try_infer_datetimes = 0
;
```

Выполним его дважды с использованием обоих загруженных файлов:

```
$ gzip -dc trips_1.gz | cc --queries-file trips_insert.sql
$ gzip -dc trips_2.gz | cc --queries-file trips_insert.sql
```

Проверка результата загрузки данных:

```
cc -q "select count() from trips"
   ┌─count()─┐
1. │ 1999658 │ -- 2.00 million
```

```
└──────────┘
```

6.3 Проверка статистики

С помощью утилиты curl можно получить информацию о текущем состоянии всех кластеров системы. Для выполнения этой команды на клиентской машине:

```
echo -n "select hostName() as host_, host_name, cluster, shard_num,
replica_num, is_local, errors_count from
clusterAllReplicas('{cluster}', system.clusters) order by host_,
cluster, shard_num, replica_num format PrettyCompact" \
| curl_ch
```

Вывод команды выше для кластера из 3-х хостов и 2-х кластеров (поле cluster).

Показывается список кластеров на каждом хосте (поле host_) так, как этот хост видит эти кластеры.

Порядок следования полей в записи совпадает с порядком следования полей в команде.

```
    ┌─host_───┬─host_name─┬─cluster─┬─shard_num─┬─replica_num─┬─is_local─┬─errors_count─┐
 1. │ host-01 │ host-01   │ s1r3    │         1 │           1 │        1 │            0 │
 2. │ host-01 │ host-02   │ s1r3    │         1 │           2 │        0 │            0 │
 3. │ host-01 │ host-03   │ s1r3    │         1 │           3 │        0 │            0 │
 4. │ host-01 │ host-01   │ s3r1    │         1 │           1 │        1 │            0 │
 5. │ host-01 │ host-02   │ s3r1    │         2 │           1 │        0 │            0 │
 6. │ host-01 │ host-03   │ s3r1    │         3 │           1 │        0 │            0 │
 7. │ host-02 │ host-01   │ s1r3    │         1 │           1 │        0 │            0 │
 8. │ host-02 │ host-02   │ s1r3    │         1 │           2 │        1 │            0 │
 9. │ host-02 │ host-03   │ s1r3    │         1 │           3 │        0 │            0 │
10. │ host-02 │ host-01   │ s3r1    │         1 │           1 │        0 │            0 │
11. │ host-02 │ host-02   │ s3r1    │         2 │           1 │        1 │            0 │
12. │ host-02 │ host-03   │ s3r1    │         3 │           1 │        0 │            0 │
13. │ host-03 │ host-01   │ s1r3    │         1 │           1 │        0 │            0 │
14. │ host-03 │ host-02   │ s1r3    │         1 │           2 │        0 │            0 │
15. │ host-03 │ host-03   │ s1r3    │         1 │           3 │        1 │            0 │
16. │ host-03 │ host-01   │ s3r1    │         1 │           1 │        0 │            0 │
17. │ host-03 │ host-02   │ s3r1    │         2 │           1 │        0 │            0 │
18. │ host-03 │ host-03   │ s3r1    │         3 │           1 │        1 │            0 │
    └─────────┴───────────┴─────────┴───────────┴─────────────┴──────────┴──────────────┘
```

Экземпляры Astra ClickHouse экспортируют показатели производительности в формате Prometheus.

Для включения экспорта метрик необходимо на сервере поместить в каталог /etc/clickhouse-server/config.d/ файл с содержимым ниже.

```
<clickhouse>
    <prometheus>
        <endpoint>/metrics</endpoint>
        <port>9363</port>
        <metrics>true</metrics>
        <events>true</events>
        <asynchronous_metrics>true</asynchronous_metrics>
    </prometheus>
```

```
</clickhouse>
```

После этого метрики на соответствующем сервере, например, можно получить с помощью команды ниже:

```
$ curl -s localhost:9363/metrics
```

Метрики запросами.

Для анализа метрик с помощью SQL система Astra ClickHouse предоставляет таблицы с большим количеством:

- текущих метрик в таблицах:
  - system.metrics
  - system.asynchronous_metrics
- исторических метрик в таблицах:
  - system.metric_log
  - system.asynchronous_metric_log

Для включения истории метрик необходимо на сервере поместить в каталог /etc/clickhouse-server/config.d/ файл с примерным содержимым ниже.

Здесь задаются имена соответствующих таблиц и правила хранения данных в них. В примере данные в таблицах хранятся 14 дней, после чего автоматически удаляются.

```
<clickhouse>
  <asynchronous_metric_log replace="1">
    <database>system</database>
    <table>asynchronous_metric_log</table>
    <engine>ENGINE = MergeTree
            PARTITION BY (event_date)
            ORDER BY (event_time)
            TTL event_date + INTERVAL 14 DAY DELETE
    </engine>
    <flush_interval_milliseconds>7500</flush_interval_milliseconds>
  </asynchronous_metric_log>

  <metric_log replace="1">
    <database>system</database>
    <table>metric_log</table>
    <engine>ENGINE = MergeTree
            PARTITION BY (event_date)
            ORDER BY (event_time)
            TTL event_date + INTERVAL 14 DAY DELETE
    </engine>
    <flush_interval_milliseconds>7500</flush_interval_milliseconds>
  </metric_log>
</clickhouse>
```

7 Управление с помощью ansible

7.1 Реконфигурация с перезапуском

При изменениях параметров кластера, требующих перезапуск служб ClickHouse.

```
ansible-playbook clickhouse.yml
```

Реконфигурация без перезапуска

При изменениях параметров кластера, не требующих перезапуск служб ClickHouse.

```
ansible-playbook clickhouse.yml -e "ch_reconfig_restart=no"
```

7.2 Останов кластера

```
ansible-playbook clickhouse.yml -e "play_action=stop"
```

7.3 Старт кластера

```
ansible-playbook clickhouse.yml -e "play_action=start"
```

8 Просмотр логов работы системы

Журналы дают возможность просматривать сообщения, выводимые программным обеспечением, работающим в ВС.

8.1 Уровни логирования

Для изменения уровня и других параметров журналирования необходимо на сервере поместить в каталог /etc/clickhouse-server/config.d/ файл с примерным содержимым ниже.

```
<clickhouse>
  <logger>
    <level>information</level>
    <log>/var/log/clickhouse-server/clickhouse-server.log</log>
    <errorlog>/var/log/clickhouse-server/clickhouse-
server.err.log</errorlog>
    <size>50M</size>
    <count>2</count>
  </logger>
</clickhouse>
```

8.2 Журнальные записи

Есть возможность получать и анализировать журнальные записи с помощью SQL. Для этого необходимо на сервере поместить в каталог /etc/clickhouse-server/config.d/ файл с примерным содержимым ниже.

```
<clickhouse>
  <text_log replace="1">
    <level>notice</level>
    <database>system</database>
    <table>text_log</table>
    <flush_interval_milliseconds>7500</flush_interval_milliseconds>
    <max_size_rows>1048576</max_size_rows>
    <reserved_size_rows>8192</reserved_size_rows>

<buffer_size_rows_flush_threshold>524288</buffer_size_rows_flush_th
reshold>
    <flush_on_crash>false</flush_on_crash>
    <!-- <partition_by>event_date</partition_by> -->
    <engine>Engine = MergeTree
            PARTITION BY event_date
            ORDER BY event_time
            TTL event_date + INTERVAL 30 day
    </engine>
  </text_log>
</clickhouse>
```

В примере конфигурируется таблица system.text_log и параметры хранения в ней данных. Данные будут храниться 30 дней, после чего будут удалены автоматически.

Пример запроса на вывод 10 последних сообщений за сегодня с уровнем Warning или Error.

```
$ cc -q "select *
from system.text_log
where event_date = today() and level in ('Warning', 'Error')
order by event_time desc
limit 10"
```

8.3 Другие _log таблицы

В дополнение к определяемым пользователем таблицам логирования в системе по умолчанию доступны некоторые другие. Их список можно получить так:

```
$ cc -q "select name
from system.tables where database = 'system' and name like '%_log'
order by name"

    ┌─name──────────────────────┐
1.  │ asynchronous_metric_log   │
2.  │ error_log                 │
3.  │ metric_log                │
4.  │ part_log                  │
5.  │ processors_profile_log    │
6.  │ query_log                 │
7.  │ query_metric_log          │
8.  │ text_log                  │
9.  │ trace_log                 │
    └───────────────────────────┘
```

Среди них, например, таблицы с историей:

- сообщений об ошибках сервере (error_log)
- выполнения запросов (query_log)
- метрик выполненных запросов (query_metric_log)
- операций с кусками таблиц (part_log)