ООО «Астра Консалтинг»

Астра ИС МД (Инфраструктурные Сервисы)

Модуль Astra ArangoDB

ДОКУМЕНТАЦИЯ, СОДЕРЖАЩАЯ ИНФОРМАЦИЮ, НЕОБХОДИМУЮ ДЛЯ ЭКСПЛУАТАЦИИ ЭКЗЕМПЛЯРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ПРЕДОСТАВЛЕННОГО ДЛЯ ПРОВЕДЕНИЯ ЭКСПЕРТНОЙ ПРОВЕРКИ

Москва, 2026

Содержание

1 Область применения

Настоящий документ содержит описание функциональных характеристик программного обеспечения, модуль Astra ArangoDB, далее Astra Arango DB, развернутого в архитектуре AstraADB Cluster.

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

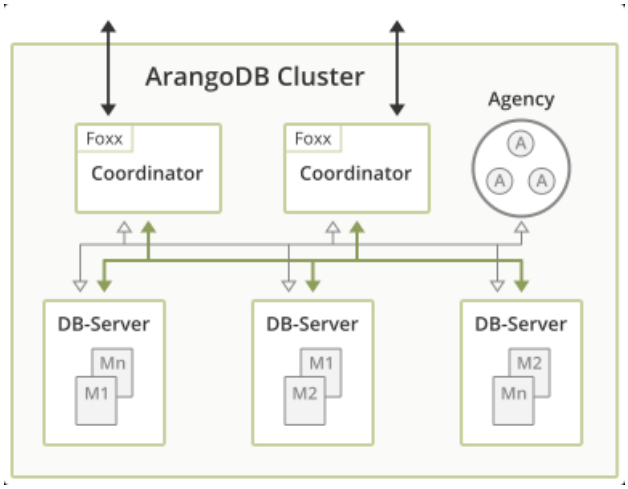
1.1 Архитектура системы

Кластер AstraADB состоит из нескольких экземпляров AstraADB, которые взаимодействуют друг с другом по сети. Они играют разные роли, описанные ниже.

Текущая конфигурация Кластера хранится в т.н. Агентстве (Agency), которое представляет из себя отказоустойчивое хранилище ключ/значение, основанное на нечётном числе экземпляров AstraADB, использующих консенсусный протокол Raft.

Существуют 3 разных типа экземпляров в Кластере AstraADB:

- Агенты (Agents)
- Координаторы (Coordinators)
- Серверы БД (DB-Servers)



1.1.1 Агенты

Один или несколько агентов формируют Агентство в Кластере AstraADB. Агентство — это центральное место хранения конфигурации Кластера. Оно выполняет выборы лидера и предоставляет остальные сервисы синхронизации для всего Кластера. Остальные компоненты не могут функционировать без Агентства.

Для Агентства в промышленных системах обязательно должна обеспечиваться отказоустойчивость. Для обеспечения этого используется консенсусный алгоритм Raft. Этот алгоритм формально гарантирует отсутствие конфликтов при управлении конфигурацией Кластера AstraADB.

1.1.2 Координаторы

Координаторы должны быть доступны извне. Работа клиентов осуществляется через них. Они координируют кластерные задачи, такие как выполнение запросов и запуск Foxx сервисов. Они знают, где хранятся данные знают, где оптимальнее запускать пользовательские запросы или их части. Координаторы не сохраняют своё состояние и легко могут быть остановлены и перезапущены при необходимости.

1.1.3 Серверы БД

Серверы БД хранят данные. Они обслуживают Разделы (shards) данных. С использованием синхронной репликации Сервер БД может быть либо Лидером, либо Ведомым (follower) для Раздела. Операции с документом сначала применяются на Лидере, а затем синхронно реплицируются на все Ведомые.

Доступ к Разделам не должен осуществляться извне, а только неявно через Координаторов. Разделы тоже могут выполнять части запросов или полностью, когда Координаторы запросили их это сделать.

1.2 Характеристики системы

Характеристики системы от команды проверки состояния кластера.

```
    $ curl -s --user "root:" --cacert /etc/security/certs/truststore.pem \
    -X GET https://haribda-01.internal:8529/_admin/cluster/health \
    | jq -r '.Health | to_entries | .[].value \
    | {"Role": .Role, "Endpoint": .Endpoint, "ShortName": .ShortName, "Status": .Status, "Version":.Version}'

{
      "Role": "Coordinator",
      "Endpoint": "ssl://10.128.0.12:8529",
      "ShortName": "Coordinator0003",
      "Status": "GOOD",
      "Version": "3.12.7-2"
    }
    {
```

```
    "Role": "Coordinator",
    "Endpoint": "ssl://10.128.0.11:8529",
    "ShortName": "Coordinator0002",
    "Status": "GOOD",
    "Version": "3.12.7-2"
  }
  {
    "Role": "DBServer",
    "Endpoint": "ssl://10.128.0.11:8530",
    "ShortName": "DBServer0002",
    "Status": "GOOD",
    "Version": "3.12.7-2"
  }
  {
    "Role": "DBServer",
    "Endpoint": "ssl://10.128.0.12:8530",
    "ShortName": "DBServer0001",
    "Status": "GOOD",
    "Version": "3.12.7-2"
  }
  {
    "Role": "Agent",
    "Endpoint": "ssl://10.128.0.12:8531",
    "ShortName": null,
    "Status": "GOOD",
    "Version": "3.12.7-2"
  }
  {
```

```
    "Role": "Agent",
    "Endpoint": "ssl://10.128.0.11:8531",
    "ShortName": null,
    "Status": "GOOD",
    "Version": "3.12.7-2"
  }
  {
    "Role": "Agent",
    "Endpoint": "ssl://10.128.0.13:8531",
    "ShortName": null,
    "Status": "GOOD",
    "Version": "3.12.7-2"
  }
```

2 Подключение к модулю Astra ArangoDB

Клиентские и серверные компоненты AstraADB могут быть загружены с сайта https://arango.ai/downloads/

Клиентский пакет имеет тип "Client Tools" и название, начинающееся с "AstraADB3e-client-". Его необходимо выбрать соответственно клиентской платформе (ОС и архитектуре).

2.1 Подключение через arangosh

```
[server]

endpoint = http+ssl://haribda-01.internal:8529

database = _system

username = root

password =


[javascript]

startup-directory = /usr/share/AstraADB3/js


[ssl]

protocol = 6
```

Для подключения из командного процессора arangosh, который становится доступным после установки пакета типа "Client Tools", удобно создать конфигурационный файл arangosh.conf с содержимым выше, и использовать его в команде ниже для проверки подключения:

```
$ arangosh -c arangosh.conf --javascript.execute-string 'print(db._version())'

3.12.7-2
```

2.2 Аутентификация

К AstraADB возможно подключение с использованием:

- имени пользователя и паролем
- jwt токеном
- файла секрета (для администрирования)

Подключение с использованием имени пользователя и пароля с использованием утилит **arangosh** и **curl** было рассмотрено в разделе 2.1.

2.2.1 Подключение с использованием jwt токена

Использование jwt токена позволяет не предоставлять для каждой команды имя пользователя и пароль.

Ниже приведена команда получения jwt токена с помощью утилит ОС **curl**, имени пользователя и его пароля.

```
curl -s --cacert /etc/security/certs/truststore.pem \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d    '{"password":    "",    "username":    "root"}'    \
-X POST 'https://haribda-01.internal:8529/_open/auth' \
| jq -r '.jwt' | tee jwt.token
```

Токен сохранится в файле **jwt.token**, который потом можно использовать в различных командах подключения.

Пример для команды **curl**:

```
curl -s --cacert /etc/security/certs/truststore.pem \
  -H "Authorization: bearer $(cat jwt.token)" \
  -X GET 'https://haribda-01.internal:8529/_admin/status' \
| jq -r
```

Пример для **arangosh** с конфигурационным файлом, из которого удалены имя пользователя и пароль, а используется содержимое файла с токеном.

Содержимое **arangosh-jwt.conf**:

```
[server]
endpoint = http+ssl://haribda-01.internal:8529
database = _system

[javascript]
startup-directory = /usr/share/AstraADB3/js

[ssl]
```

```
    protocol = 6
```

Использование:

```
    arangosh -c arangosh-jwt.conf --server.jwt-token $(cat
jwt.token) \
    --javascript.execute-string 'print(db._version())'
```

2.2.2 Использование файла секрета

Файл секрета обычно используется для административных задач.

Файл **ArangoDB.secret** в нашем варианте установки лежит в клиентской файловой системе в каталоге ansible, откуда производилась установка Кластера. Он же скопирован на каждый сервер в каталог /etc/AstraADB.

Пример содержимого конфигурационного файла arangosh-secret.conf для **arangosh**.

```
    [server]
    endpoint = http+ssl://haribda-01.internal:8529
    database = _system
    jwt-secret-keyfile = files/secret/AstraADB.secret

    [javascript]
    startup-directory = /usr/share/AstraADB3/js

    [ssl]
    protocol = 6
    Использование:
    arangosh -c arangosh-secret.conf --javascript.execute-string
'print(db._version())'
```

Использование для вызова административного API.

Для этого необходимо из файла **ArangoDB.secret** получить jwt token для его использования в таких командах. Стандартная процедура получения такого токена использует утилиту **ArangoADB**, которая не входит в комплект клиентского ПО AstraADB. Поэтому, если необходимо выполнять такие команды с клиента, то можно использовать небольшой python скрипт (здесь не приводится), которая получает токен на основе payload файла определённого содержимого **jwtMessage-ArangoDB.json** и файла секрета **ArangoDB.secret**.

```
# Переменные
b64s="$(cat files/secret/AstraADB.secret | base64 -w 0)"
tf="token_hs256.txt"

# Получение токена
./python-jwt.py \
-m jwtMessage-AstraADB.json \
-s "${b64s?}" -e 36000000 \
| tee "${tf?}"

# Использование для получения конфигурации Агентства
curl -s --cacert /etc/security/certs/truststore.pem \
-H "Authorization: bearer $(cat "${tf?}")" \
https://haribda-01.internal:8531/_api/agency/config | jq -r
```

3 Основные команды модуля Astra ArangoDB

Для выполнения команд ниже с помощью утилиты curl удобно объявить алиас:

```
alias arango_curl='curl -s --user "root:" --cacert /etc/security/certs/truststore.pem'
```

Более подробная информация о ArangoDB Rest API доступна из ArangoDB Web UI по адресу (пункты меню Support, вкладка Rest API):

https://haribda-01.internal:8529

3.1 Получение информации о сервере

3.1.1 Общая информация о координаторе

arango_curl https://haribda-01.internal:8529/_admin/status | jq -r

3.1.2 Информация о работоспособности всех компонентов кластера

arango_curl https://haribda-01.internal:8529/_admin/cluster/health | jq -r

3.1.3 Режим сервера (default - read/write или readonly)

arango_curl https://haribda-01.internal:8529/_admin/server/mode | jq -r '.mode'

3.1.4 Список Координаторов

arango_curl https://haribda-01.internal:8529/_api/cluster/endpoints | jq -r '.endpoints'

3.1.5 Список баз данных

arango_curl https://haribda-01.internal:8529/_db/_system/_api/database | jq -r '.result[]'

3.1.6 Создание Базы Данных (curl):

```
arango_curl --header 'accept: application/json' \
--data '{"name": "testdb", "options": { "replicationFactor": 2 } }' \
-X POST https://haribda-01.internal:8529/_api/database | jq -r
```

3.1.7 Создание Базы Данных и пользователя (arangosh)

```
arangosh -c arangosh.conf --javascript.execute-string \
```

```
    'db._createDatabase("testdb2" , { "replicationFactor": 2 },
[ {"username": "test", "passwd": "test"}])'
```

3.1.8 Выдача прав пользователю (curl)

Пользователю **test** в базе **testdb**:

```
    arango_curl --header 'accept: application/json' \
    --data '{ "grant": "rw" }' \
    -X PUT https://haribda-
01.internal:8529/_db/_system/_api/user/test/database/testdb | jq
-r
```

3.1.9 Информация о доступных пользователю Базах Данных

Имя пользователя - **test**.

```
    arango_curl https://haribda-
01.internal:8529/_db/_system/_api/user/test/database | jq -r
'.result'
```

3.1.10 Информация о пользователей

Имя пользователя - **test**.

```
    arango_curl https://haribda-
01.internal:8529/_db/_system/_api/user/test | jq -r
```

3.1.11 Информация о Базе Данных

Имя БД - **testdb**.

```
    arango_curl https://haribda-
01.internal:8529/_db/testdb/_api/database/current | jq -r
'.result'
```

3.2 Работа с коллекциями документов

3.2.1 Утилита arangoimp

Для демонстрации можно использовать наборы данных в json и csv формате, доступные на сайте ниже:

https://github.com/ArangoDB/example-datasets

Для того, чтобы коллекция создалась в базе testdb, удобно создать конфигурационный файл arangoimp.conf текущем каталоге с содержимым:

```
[server]
endpoint = http+ssl://haribda-01.internal:8529
database = testdb
username = test
password = test

[ssl]
protocol = 6
```

и конфигурационный файл arangosh-testdb.conf для arangosh:

```
[server]
endpoint = http+ssl://haribda-01.internal:8529
database = testdb
username = test
password = test

[javascript]
startup-directory = /usr/share/AstraADB3/js

[ssl]
protocol = 6
```

Создадим каталог с файлами данных:

```
mkdir data/
Аэропорты.
```

Загрузка:

```
wget -O data/airports.csv \
https://github.com/AstraADB/example-
datasets/raw/refs/heads/master/Airports/airports.csv
```

Вставка данных в коллекцию базы testdb:

```
    $ arangoimp --file data/airports.csv --collection=airports \
    --create-collection=true                              --type=csv
...
created:          43991
    warnings/errors:  0
    updated/replaced: 0
    ignored:          0
    lines read:       43993
```

3.2.2 Вывод количества документов в коллекции

```
    $ arangosh -c arangosh-testdb.conf \
    --javascript.execute-string      "print('Docs:      '      +
db.airports.count())"


    Docs: 43991
```

3.2.3 Список коллекций в Базе Данных

При подключении к нужной базе за исключением системных коллекций.

```
    arangosh -c arangosh-testdb.conf \
    --javascript.execute-string
"print(db._collections().filter(function(coll)            {return
!coll.name().startsWith('_')}))"
```

3.2.4 Вывод N первых документов коллекции

```
    $  arangosh  -c  arangosh-testdb.conf  --javascript.execute-
string "print(db.airports.all().limit(1).toArray())"
    [
      {
        "_key" : "2117231",
        "_id" : "airports/2117231",
        "_rev" : "_lJ3fPMW---",
```

```
        "id" : 28894,
        "ident" : "LFHZ",
        "type" : "small_airport",
        "name" : "Sallanches Airport",
        "latitude_deg" : 45.9538993835449,
        "longitude_deg" : 6.63917016983032,
        "elevation_ft" : 1755,
        "continent" : "EU",
        "iso_country" : "FR",
        "iso_region" : "FR-V",
        "municipality" : "Moulins",
        "scheduled_service" : "no",
        "gps_code" : "LFHZ"
      }
    ]
```

3.2.5 Документы с фильтрацией

```
    $ arangosh -c arangosh-testdb.conf --javascript.execute-string "print(db._query('FOR a IN airports FILTER a.iso_country == \"US\" LIMIT 1 RETURN a').toArray())"
    [
      {
        "_key" : "2117705",
        "_id" : "airports/2117705",
        "_rev" : "_lJ3fPMq--T",
        "id" : 6523,
        "ident" : "00A",
        "type" : "heliport",
        "name" : "Total Rf Heliport",
```

```
        "latitude_deg" : 40.07080078125,
        "longitude_deg" : -74.9336013793945,
        "elevation_ft" : 11,
        "continent" : "NA",
        "iso_country" : "US",
        "iso_region" : "US-PA",
        "municipality" : "Bensalem",
        "scheduled_service" : "no",
        "gps_code" : "00A",
        "local_code" : "00A"
      }
    ]
```

3.2.6 Количество документов с фильтрацией

```
    $ arangosh -c arangosh-testdb.conf \
    --javascript.execute-string "print(db._query('FOR a IN airports FILTER a.iso_country == \"FR\" COLLECT WITH COUNT INTO count RETURN count').toArray()[0])"
    782
```

3.2.7 Удаление коллекции

При подключении к нужной базе данных. Удаление коллекции **airports**.

```
    arangosh -c arangosh-testdb.conf \
    --javascript.execute-string "db.airports.drop()"
```

3.2.8 Удаление Базы Данных

При подключении к системной Базе Данных.

```
    arangosh -c arangosh.conf \
    --javascript.execute-string "db._dropDatabase('testdb2')"
```

4 Мониторинг и статистика

4.1 Метрики в формате Prometheus

Экземпляры AstraADB экспортируют показатели производительности в формате Prometheus.

Для доступа к ним по http(s) необходимо сгенерировать токен доступа на основе файла секрета. Ниже приведён пример скрипта get_prom_metrics.sh, который генерирует такой токен и выводит метрики со всех экземпляров кластера.

```
b64s="$(cat files/secret/AstraADB.secret | base64 -w 0)"
tf="token_hs256.txt"


# Get Token
./python-jwt.py \
-m jwtMessage-AstraADB.json \
-s "${b64s?}" -e 36000000 \
| tee "${tf?}"


# Use it in adm API:
for port in 8529 8530 8531; do
  for host_id in 1 2 3; do
    curl -s --cacert /etc/security/certs/truststore.pem \
    -H "Authorization: bearer $(cat "${tf?}")" \
    https://haribda-
0${host_id?}.internal:${port?}/_admin/metrics/v2
  done
done
```

Доступны дашборды для визуализации этих метрик в Grafana: *https://github.com/AstraADB/dashboards*

Метрики только от Координатора могут быт получены с именем пользователя и паролем (без токена):

```
arango_curl "https://haribda-
01.internal:8529/_db/_system/_admin/metrics/v2"
```

4.2 Уровни логирования

Уровни логирования компонент кластера.

```
    $ arango_curl "https://haribda-
01.internal:8529/_admin/log/level" | jq -r

    {
      "agency": "INFO",
      "agencycomm": "INFO",
      "agencystore": "WARNING",
      "backup": "DEFAULT",
      "bench": "DEFAULT",
      "cluster": "INFO",
      "communication": "INFO",
      "authentication": "WARNING",
      "config": "DEFAULT",
      "crash": "DEFAULT",
      "dump": "INFO",
      "engines": "INFO",
      "general": "INFO",
      "heartbeat": "INFO",
      "aql": "INFO",
      "graphs": "INFO",
      "maintenance": "INFO",
      "authorization": "DEFAULT",
      "queries": "INFO",
      "v8": "WARNING",
      "license": "INFO",
      "libiresearch": "INFO",
```

```
      "deprecation": "INFO",
      "rocksdb": "WARNING",
      "audit-database": "INFO",
      "validation": "INFO",
      "flush": "INFO",
      "audit-authorization": "INFO",
      "syscall": "INFO",
      "cache": "INFO",
      "security": "INFO",
      "memory": "INFO",
      "restore": "INFO",
      "httpclient": "WARNING",
      "audit-view": "INFO",
      "audit-document": "INFO",
      "audit-hotbackup": "INFO",
      "requests": "FATAL",
      "audit-service": "INFO",
      "ttl": "WARNING",
      "replication2": "WARNING",
      "ssl": "WARNING",
      "threads": "WARNING",
      "trx": "WARNING",
      "supervision": "INFO",
      "audit-collection": "INFO",
      "statistics": "INFO",
      "startup": "DEBUG",
      "audit-authentication": "INFO",
      "rep-wal": "WARNING",
```

```
      "arangosearch": "INFO",
      "views": "FATAL",
      "rep-state": "WARNING",
      "development": "FATAL",
      "replication": "INFO"
    }
```

4.3 Журнальные записи

Все записи с уровнем Warning (2) и выше, с подстрокой "heartbeat+took" в тексте сообщения, с id сообщения, начиная с 43.

```
    arango_curl "https://haribda-
01.internal:8529/_admin/log/entries?upto=2&search=heartbeat+took
&start=43" \
    | jq -r
```

5 Резервное копирование и восстановление

Команды можно запускать с клиента.

Для запуска утилит удобно создать:

- конфигурационный файл **arango-utl-sys.conf** подключения к системной базе от пользователя **root** с содержимым ниже

```
[server]

endpoint = http+ssl://haribda-01.internal:8529

database = _system

username = root

password =


[ssl]

protocol = 6
```

конфигурационный файл **my-local.json** с содержимым ниже:

```
{
  "my-local": {
    "type": "local",
    "copy-links": "false",
    "links": "false",
    "one_file_system": "false"
  }
}
```

5.1 Создание резервной копии

Утилита быстро создаёт логическую резервную копию в виде снимка на уровне ОС.

Утилита генерирует уникальный идентификатор копии, который можно будет использовать для восстановления из этой копии.

```
$ arangobackup create -c arango-utl-sys.conf --label backup-
01
```

```
    2026-03-06T08:13:58.357396Z [264682-1] INFO [11111] {general}
This executable uses the GNU C library (glibc), which is licensed
under the GNU Lesser General Public License (LGPL), see
https://www.gnu.org/copyleft/lesser.html                    and
https://www.gnu.org/licenses/gpl.html

    2026-03-06T08:13:58.395120Z [264682-1] INFO [06792] {backup}
Server version: 3.12.7-2

    2026-03-06T08:13:59.495311Z [264682-1] INFO [c4d37] {backup}
Backup     succeeded.     Generated     identifier     '2026-03-
06T08.13.58Z_backup-01'

    2026-03-06T08:13:59.495512Z [264682-1] INFO [ce423] {backup}
Total size of backup: 124004594, number of files: 57
```

5.2 Листинг резервных копий

```
    $     arangobackup     -c     arango-utl-sys.conf     list

... INFO [06792] {backup} Server version: 3.12.7-2

    ... INFO [e0356] {backup} The following backups are available:

    ... INFO [9e6b6] {backup}  - 2026-03-06T08.13.58Z_backup-01

    ... INFO [0f208] {backup}       version:   3.12.7-2

    ...  INFO  [55af7]  {backup}            date/time:  2026-03-
06T08:13:58Z

    ... INFO [43522] {backup}       size in bytes: 124004594

    ... INFO [12532] {backup}       number of files: 57

    ... INFO [43212] {backup}       number of DBServers: 2

    ... INFO [12533] {backup}       number of available pieces:
2

    ... INFO [56242] {backup}     potentiallyInconsistent: false

    ... INFO [56244] {backup}       available: true
```

5.3 Выгрузка резервной копии

На всех хостах с Серверами БД надо создать целевой каталог для архивирования.

В данном примере: **/var/lib/arango/backup/**

Задание на выгрузку запускается асинхронно и сразу же возвращает управление.

```
$ arangobackup -c arango-utl-sys.conf \
--identifier '2026-03-06T08.13.58Z_backup-01' \
--rclone-config-file ./my-local.json \
--remote-path my-local://var/lib/arango/backup/ \
--compress-transfer \
upload

... INFO [06792] {backup} Server version: 3.12.7-2
... INFO [a9597] {backup} Backup initiated, use
... INFO [4c459] {backup}     arangobackup upload --status-id=18250
... INFO [5cd70] {backup}  to query progress.
```

В выводе утилите указан ID задания, который можно использовать для просмотра хода и результата выполнения.

```
$ arangobackup -c arango-utl-sys.conf upload --status-id=18250
... INFO [06792] {backup} Server version: 3.12.7-2
... INFO [24d75] {backup} PRMR-23e7ef1f-8de3-41ca-a2d6-ff583a43f33b Status: COMPLETED
... INFO [68cc8] {backup} Last progress update 2026-03-06T09:38:45Z: 29/29 files done
... INFO [24d75] {backup} PRMR-9ba7b1fb-bb9d-4615-b682-1614056df584 Status: COMPLETED
... INFO [68cc8] {backup} Last progress update 2026-03-06T09:38:45Z: 30/30 files done
```

Команда делает выгрузку на всех хостах с Серверами БД в указанный каталог и выводит статистику по выполненной работе. Выводятся идентификаторы Серверов (PRMR-* их можно получить вызовом API `/_admin/cluster/health`) и статус выполнения на них.

5.4 Удаление резервной копии

Горячие архивы занимают дополнительное дисковое пространство.

После выгрузки копии её желательно удалять.

Удаление копии:

```
$ arangobackup -c arango-utl-sys.conf \
--identifier '2026-03-06T08.13.58Z_backup-01' \
delete

... INFO [06792] {backup} Server version: 3.12.7-2
... INFO [a23cb] {backup} Successfully deleted '2026-03-06T08.13.58Z_backup-01'
```

Листинг копий больше не показывает её.

```
$ arangobackup -c arango-utl-sys.conf list

... INFO [06792] {backup} Server version: 3.12.7-2
... INFO [efc76] {backup} There are no backups available.
```

5.5 Восстановление из резервной копии

Для демонстрации удалим сначала коллекцию **airports** из базы данных **testdb**.

```
arangosh -c arangosh-testdb.conf \
--javascript.execute-string "db.airports.drop()"
```

Идентификатор копии можно найти, например, листингом каталога, в который делалась копия.

```
$ ssh haribda-01.internal -- sudo ls -1 /var/lib/arango/backup/
2026-03-06T08.13.58Z_backup-01
```

Загрузим нужную копию.

```
$ arangobackup -c arango-utl-sys.conf \
--identifier '2026-03-06T08.13.58Z_backup-01' \
--rclone-config-file ./my-local.json \
--remote-path my-local://var/lib/arango/backup/ \
```

```
    download
    ... INFO [06792] {backup} Server version: 3.12.7-2
    ... INFO [a9597] {backup} Backup initiated, use
    ... INFO [4c459] {backup}    arangobackup download --status-
id=18336
    ... INFO [5cd70] {backup}  to query progress.
```

Статус загрузки:

```
    $ arangobackup -c arango-utl-sys.conf download --status-
id=18336
    ... INFO [06792] {backup} Server version: 3.12.7-2
    ... INFO [24d75] {backup} PRMR-23e7ef1f-8de3-41ca-a2d6-
ff583a43f33b Status: COMPLETED
    ... INFO [68cc8] {backup} Last progress update 2026-03-
06T11:46:01Z: 29/29 files done
    ... INFO [24d75] {backup} PRMR-9ba7b1fb-bb9d-4615-b682-
1614056df584 Status: COMPLETED
    ... INFO [68cc8] {backup} Last progress update 2026-03-
06T11:46:01Z: 30/30 files done
```

Копия появилась в листинге копий:

```
    $     arangobackup     -c     arango-utl-sys.conf     list

... INFO [06792] {backup} Server version: 3.12.7-2
    ... INFO [e0356] {backup} The following backups are available:
    ... INFO [9e6b6] {backup}  - 2026-03-06T08.13.58Z_backup-01
    ... INFO [0f208] {backup}       version:   3.12.7-2
    ... INFO [55af7] {backup}            date/time: 2026-03-
06T08:13:58Z
    ... INFO [43522] {backup}       size in bytes: 124004594
    ... INFO [12532] {backup}       number of files: 57
    ... INFO [43212] {backup}       number of DBServers: 2
```

```
     ... INFO [12533] {backup}        number of available pieces:
2

     ... INFO [56242] {backup}      potentiallyInconsistent: false

     ... INFO [56244] {backup}        available: true
```

Восстановление из копии.

Команда синхронная, восстанавливаются все базы, по окончании все Серверы БД перезапускаются.

```
     $ arangobackup -c arango-utl-sys.conf \

     --identifier '2026-03-06T08.13.58Z_backup-01' \

     restore


...  INFO  [b6d4c]  {backup}  Successfully  restored  '2026-03-
06T08.13.58Z_backup-01'
```

Листинг коллекций в базе **testdb**, которую бы до этого удалили.

```
     $ arangosh -c arangosh-testdb.conf \

     --javascript.execute-string
"print(db._collections().filter(function(coll)            {return
!coll.name().startsWith('_')}))"


[

       [ArangoCollection  13020032,  "airports"  (type  document,
status loaded)]

     ]
```

Приложение А Часто задаваемые вопросы

А.1 Как подключиться к Astra ArangoDB извне?

Для подключения к AstraADB могут быть использованы следующие интерфейсы:

А.1.1.1 Базовые API

- **HTTP API (RESTful):** основной интерфейс взаимодействия с сервером AstraADB с использованием стандартных методов HTTP (GET, POST, PUT, DELETE, PATCH) и данными в формате JSON. Этот API документирован с использованием спецификации OpenAPI и может быть использован в Web интерфейсе с помощью Swagger UI. Есть примеры для работы с коллекциями (/_api/collection), базами данных (/_api/database), документами (/_api/document), графами (/_api/gharial).
- **JavaScript API (@ArangoDB модуль):** этот API предназначен для написания серверной логики, такой как микросервисы Foxx, или для использования с интерфейсом командной строки arangosh. Он предоставляет объекты и методы для прямого взаимодействия с базами, коллекциями, AQL запросами.

API для языков программирования AstraADB предоставляет официальные клиентские драйверы, которые используют HTTP API в функциях различных языков программирования, включая:

- Java driver
- Go driver
- Python driver
- Node.js driver (arangojs)
- PHP client

А.1.2 Как изменить пароль пользователя

При соединении с базой _system с использованием шаблона:

```
    require("@AstraADB/users").update("username", "newPassword", true)
```

Пример:

```
arangosh -c arangosh.conf \--javascript.execute-string 'require("@AstraADB/users").update("test", "password", true)'
```

A.2 Установка Astra ArangoDB с использованием инсталлятора

A.2.1 Установка Astra ArangoDB с использованием инсталлятора

Для использования инсталлятора требуется предварительная подготовка виртуальных машин и серверов, включая настройку операционной системы, установку необходимых пакетов и инструментов, а также настройку сетевых подключений.

A.2.1.1 Установка Ansible:

– Убедитесь, что Ansible версии 2.9 и выше установлен.

Рекомендуется использовать Python 3.8 и выше.

```
sudo apt update
sudo apt install -y ansible python3-pip
```

A.2.1.2 Установка необходимых Python-модулей

```
pip3 install --user ansible
```

A.2.2 Инструкция по установке AstraADB с использованием плейбуков Ansible

Инструкция описывает установку в режиме AstraADB Cluster.

https://docs.arango.ai/ArangoDB/stable/deploy/cluster/#structure-of-an-ArangoDB-cluster

Пример ниже показан для кластера из 3-х узлов, где 2 узла обслуживают по 3 компоненты - Coordinator, DB-Server, Agent, а 1 узел - только Agent.

A.2.2.1 Шаг 1: Настройка файла инвентаря

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

```
[AstraADB:children]
dbservers
agents
```

```
[dbservers]
host-01.internal
host-02.internal

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

A.2.2.2 Шаг 2: Настройка плейбука

Основной плейбук для запуска установки находится в файле `**ArangoDB.yml**`.

Соответствующий целевой платформе дистрибутив AstraADB или ссылка на него должны располагаться в каталоге **roles/ArangoDB/files/distrib/**. Установка будет произведена стандартным для этой платформы пакетным менеджером из указанного файла.

A.2.2.3 Шаг 3: 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/ArangoDB/defaults/main.yml`*** содержит основные переменные для настройки кластера:

adb_baseDir: "/var/lib/arango" – Путь к основному каталогу данных AstraADB.

adb_with_ssl: "yes" - Если этот параметр имеет значение "yes", то на локальном хосте должны быть подготовлены файлы ssh сертификатов и ключей в каталоге **roles/AstraADB/files/certs/** со следующими правилами именования и содержимым

- hostname.pem - для каждого hostname из инвентарного файла inv; содержит сертификат и приватный ключ для хоста hostname в формате pem
- truststore.pem - содержит CA сертификаты в формате pem

Остальные параметры обычно не меняются.

adb_cfgDir: "/etc/ArangoDB" – Путь к основному каталогу конфигурационных файлов AstraADB

adb_certDir: "{{ adb_cfgDir ~ '/certs' }}" – Путь к каталогу с сертификатами и ключами

adb_env_file: "{{ adb_cfgDir ~ '/ArangoDB.env' }}" – Путь к файлу переменных окружения AstraADB

adb_keystore_file: "{{ adb_certDir ~ '/node.pem' }}" – Путь к файлу сертификата и ключа хоста AstraADB

adb_truststore_file: "{{ adb_certDir ~ '/truststore.pem' }}" – Путь к файлу CA сертификатов

adb_service_name: " ArangoDB -starter.service" – Имя файла systemd сервиса AstraADB

adb_unit_filename: "{{ '/etc/systemd/system/' ~ adb_service_name }}" – Полный путь к файлу systemd сервиса AstraADB

adb_auth_jwt_secret: "{{ adb_cfgDir ~ '/ ArangoDB.secret' }}" – Файл секрета AstraADB

adb_ArangoDB_options: – Список параметров сервиса (длинный, здесь не приводится)

A.2.2.4 Шаг 4: Запуск плейбука в режиме установки

Для запуска установки выполните команду:

```
ansible-playbook ArangoDB.yml -e "adb_config_only=no"
```

A.2.2.5 Дополнительные шаги

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

```
ansible -m shell -a "systemctl status arangodb-starter.service" all
```

A.2.3 Проверка состояния кластера

После завершения установки и настройки кластера AstraADB, вы можете проверить состояние всех узлов с помощью утилиты ОС curl с использованием HTTP API. Для удобного форматирования результатов вызова curl может использоваться утилита jq.

Если кластер сконфигурирован с использованием SSL/TLS, то в параметре --cacert необходимо указывать тот же файл с CA сертификатами, который использовался для установки.

По умолчанию в кластере создается пользователь root без пароля с административными правами, который и используется везде ниже.

A.2.3.1 Получение состояния всех компонент кластера

С помощью использования пути /_admin/cluster/health HTTP API можно получить информацию о текущем состоянии всех узлов в кластере. Для выполнения этой команды на клиентской машине:

```
curl -s --user "root:" \
--cacert /etc/security/certs/truststore.pem \
-X GET https://host-01:8529/ admin/cluster/health \
| jq -r '.Health | to_entries \
| .[].value | [.Role, .Endpoint, .ShortName, .Status] | @csv'
```

A.2.3.2 Пример вывода команды получения состояния всех компонент кластера

Вывод команды выше для кластера из 2-х Coordinators и DB-Servers и 3-х агентов.

Формируется по одной записи на каждую компоненту кластера.

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

```
"Coordinator","ssl://10.128.0.12:8529","Coordinator0001","GOOD"
"Coordinator","ssl://10.128.0.11:8529","Coordinator0002","GOOD"
"DBServer","ssl://10.128.0.11:8530","DBServer0002","GOOD"
"DBServer","ssl://10.128.0.12:8530","DBServer0001","GOOD"
"Agent","ssl://10.128.0.12:8531",,"GOOD"
"Agent","ssl://10.128.0.11:8531",,"GOOD"
"Agent","ssl://10.128.0.13:8531",,"GOOD"
```

Пример результата команды с ненормальными состояниями компонент (один из серверов недоступен):

```
"Coordinator","","Coordinator0001","FAILED"
"Coordinator","ssl://10.128.0.11:8529","Coordinator0002","GOOD"
"DBServer","ssl://10.128.0.11:8530","DBServer0002","GOOD"
"DBServer","","DBServer0001","BAD"
"Agent","ssl://10.128.0.12:8531",,"BAD"
"Agent","ssl://10.128.0.11:8531",,"GOOD"
"Agent","ssl://10.128.0.13:8531",,"GOOD"
```

A.2.4 Операции с кластером

A.2.4.1 Реконфигурация опций

При измененииях опций кластера запустить команду ниже.

```
ansible-playbook ArangoDB.yml
```

A.2.4.2 Реконфигурация ssl <-> non-ssl

При изменении параметра **adb_with_ssl**.

```
ansible-playbook ArangoDB.yml -e "adb_ssl_reconfig=yes"
```

A.2.4.3 Останов кластера

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

A.2.4.4 Старт кластера

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

A.2.4.5 Обновление кластера онлайн

Убедитесь в том, что кластер запущен и все его компоненты работоспособны.

Скопируйте новый дистрибутив или создайте ссылку не него в каталоге **roles/AstraADB/files/distrib** и удалите старый дистрибутив/ссылку.

Запустите команду:

```
ansible-playbook ArangoDB.yml -e "play_action=upgrade-online"
```

A.2.5 Заключение

После развертывания и настройки кластера с помощью Ansible и запуска AstraADB, регулярная проверка состояния узлов с помощью утилиты curl является ключевым шагом в мониторинге здоровья и производительности кластера. Этот инструмент позволяет вам оперативно обнаружить проблемы и принять необходимые меры для их устранения.

Если статус всех компонент кластера отображаются как GOOD, то это означает, что ваш кластер функционирует корректно.

А.3 Инструкция по подключению к стенду

Доступ к стенду предоставляется с помощью ssh подключения:

```
ssh 4Check@80.85.253.160
```

На этом сервере:

- установлены утилиты для работы с AstraADB, такие как arangosh и arangobench
- создан файл ~/ArangoDB/arangosh.conf для подключения к кластеру для примера

Пример получения соединения с кластером и выполнения команды:

```
$ arangosh -c ~/ArangoDB/arangosh.conf \
--javascript.execute-string 'print(db._version())'

3.12.7-2
```

Пример использования утилиты curl с запросом на проверку жизнеспособности всех компонентов кластера:

```
$ curl -s --user "root:" --cacert /etc/security/certs/truststore.pem \
-X GET https://haribda-01.internal:8529/_admin/cluster/health | jq -r

{
  "error": false,
  "code": 200,
  "Health": {
    "CRDN-5d366411-f4f5-4ea2-a0e7-e03875e1e711": {
      "Endpoint": "ssl://10.128.0.12:8529",
      "Engine": "rocksdb",
      "Host": "cfe86e3c7a3a482fb1105d34438413e6",
      "LastAckedTime": "2026-03-02T12:20:43Z",
      "ShortName": "Coordinator0003",
      "Status": "GOOD",
      "SyncStatus": "SERVING",
      "SyncTime": "2026-03-02T12:20:43Z",
      "Timestamp": "2026-03-02T12:20:43Z",
      "Version": "3.12.7-2",
      "Role": "Coordinator",
      "CanBeDeleted": false
    },
    "CRDN-bdab5531-47a5-493b-9f82-3a5d4a39e367": {
      "Endpoint": "ssl://10.128.0.11:8529",
      "Engine": "rocksdb",
      "Host": "528209b2888a43d8a0f9ab55af99d762",
      "LastAckedTime": "2026-02-27T09:51:15Z",
      "ShortName": "Coordinator0002",
      "Status": "GOOD",
      "SyncStatus": "SERVING",
      "SyncTime": "2026-02-27T09:51:14Z",
      "Timestamp": "2026-02-27T09:51:15Z",
      "Version": "3.12.7-2",
      "Role": "Coordinator",
      "CanBeDeleted": false
    },
    "PRMR-23e7ef1f-8de3-41ca-a2d6-ff583a43f33b": {
      "Endpoint": "ssl://10.128.0.11:8530",
      "Engine": "rocksdb",
      "Host": "528209b2888a43d8a0f9ab55af99d762",
      "LastAckedTime": "2026-02-27T09:51:15Z",
```

```
      "ShortName": "DBServer0002",
      "Status": "GOOD",
      "SyncStatus": "SERVING",
      "SyncTime": "2026-02-27T09:51:15Z",
      "Timestamp": "2026-02-27T09:51:15Z",
      "Version": "3.12.7-2",
      "Role": "DBServer",
      "CanBeDeleted": false
    },
    "PRMR-9ba7b1fb-bb9d-4615-b682-1614056df584": {
      "Endpoint": "ssl://10.128.0.12:8530",
      "Engine": "rocksdb",
      "Host": "cfe86e3c7a3a482fb1105d34438413e6",
      "LastAckedTime": "2026-03-02T12:20:42Z",
      "ShortName": "DBServer0001",
      "Status": "GOOD",
      "SyncStatus": "SERVING",
      "SyncTime": "2026-03-02T12:20:42Z",
      "Timestamp": "2026-03-02T12:20:42Z",
      "Version": "3.12.7-2",
      "Role": "DBServer",
      "CanBeDeleted": false
    },
    "AGNT-1f9de0af-be86-4a1e-91cf-5b7212461aa2": {
      "Role": "Agent",
      "Endpoint": "ssl://10.128.0.12:8531",
      "CanBeDeleted": false,
      "Leading": false,
      "LastAckedTime": 1.104,
      "Engine": "rocksdb",
      "Version": "3.12.7-2",
      "Leader": "AGNT-f853e283-350f-455e-a713-c2ae1d83709d",
      "Status": "GOOD"
    },
    "AGNT-66b86ad8-0170-4372-a243-d49ceb80a7db": {
      "Role": "Agent",
      "Endpoint": "ssl://10.128.0.11:8531",
      "CanBeDeleted": false,
      "Leading": false,
      "LastAckedTime": 1.104,
      "Engine": "rocksdb",
      "Version": "3.12.7-2",
      "Leader": "AGNT-f853e283-350f-455e-a713-c2ae1d83709d",
      "Status": "GOOD"
    },
    "AGNT-f853e283-350f-455e-a713-c2ae1d83709d": {
      "Role": "Agent",
      "Endpoint": "ssl://10.128.0.13:8531",
      "CanBeDeleted": false,
      "Leading": true,
      "LastAckedTime": 0,
      "Engine": "rocksdb",
      "Version": "3.12.7-2",
      "Leader": "AGNT-f853e283-350f-455e-a713-c2ae1d83709d",
      "Status": "GOOD"
    }
  },
  "ClusterId": "26d01042-c686-48a5-bfa1-27e7bb9b502e"
}
```


По запросу может быть представлен пример вызова утилиты нагрузочного тестирования arangobench.

А.4 Инструкция по проверке экземпляра программного обеспечения модуль Astra ArangoDB

А.4.1 Выполнение проверки

С помощью использования пути /_admin/cluster/health HTTP API можно получить информацию о текущем состоянии всех узлов в кластере. Для выполнения этой команды на клиентской машине:

```
curl -s --user "root:" \
--cacert /etc/security/certs/truststore.pem \
-X GET https://haribda-01.internal:8529/_admin/cluster/health \
| jq -r '.Health | to_entries \
| .[].value | [.Role, .Endpoint, .ShortName, .Status] | @csv'
```

Пример вывода команды получения состояния всех компонент кластера.

Вывод команды выше для кластера из 2-х Coordinators и DB-Servers и 3-х агентов.

Формируется по одной записи на каждую компоненту кластера.

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

А.4.2 Ожидаемый результат проверки

Ожидаемым результатом является выполнение команды без ошибок со словом "GOOD" в каждой строке.

```
"Coordinator","ssl://10.128.0.12:8529","Coordinator0001","GOOD"
"Coordinator","ssl://10.128.0.11:8529","Coordinator0002","GOOD"
"DBServer","ssl://10.128.0.11:8530","DBServer0002","GOOD"
"DBServer","ssl://10.128.0.12:8530","DBServer0001","GOOD"
"Agent","ssl://10.128.0.12:8531",,"GOOD"
"Agent","ssl://10.128.0.11:8531",,"GOOD"
"Agent","ssl://10.128.0.13:8531",,"GOOD"
```

Пример результата команды с ненормальными состояниями компонент (один из серверов недоступен):

```
"Coordinator","","Coordinator0001","FAILED"
"Coordinator","ssl://10.128.0.11:8529","Coordinator0002","GOOD"
"DBServer","ssl://10.128.0.11:8530","DBServer0002","GOOD"
"DBServer","","DBServer0001","BAD"
```

```
"Agent","ssl://10.128.0.12:8531",,"BAD"
"Agent","ssl://10.128.0.11:8531",,"GOOD"
"Agent","ssl://10.128.0.13:8531",,"GOOD"
```