Программное обеспечение «Нестор»

Руководство администратора

Санкт-Петербург, 2024

Введение

Настоящий документ (далее – Описание) распространяется на программное обеспечение «Нестор» («NeStore») (далее – Платформа).

Данное Описание содержит информацию, необходимую для администрирования Платформы: область применения Платформы, назначение, условия применения, подготовка Платформы к работе, описание операций, возможные аварийные ситуации.

Содержание

1. Введение

1.1. Область применения средства автоматизации

Платформа предназначена для создания и выполнения на основе ее возможностей прикладных систем, обеспечивающих автоматизацию процессов сбора, хранения, анализа и визуализации данных, например, для диспетчерских информационно-управляющих систем реального времени.

Созданные на базе Платформы прикладные системы могут осуществлять сбор информации из разных источников, по различным протоколам, производить обработку принятых данных, их архивирование и представление пользователю информации на средствах отображения индивидуального и коллективного пользования. Также на базе Платформы можно обеспечить решение ряда расчетно-аналитических задач. Состав задач определяется потребностями автоматизируемых процессов и может меняться в процессе эксплуатации.

1.2. Краткое описание возможностей Платформы

Платформа предназначена для автоматизации технологических процессов, связанных со сбором, хранением, визуализацией и анализом данных.

1.3. Уровень подготовки администратора

Персонал Платформы должен обладать квалификацией достаточной для обеспечения:

- конфигурации, настройки или модернизации компонентов технического и программного обеспечения Платформы;
- мониторинга работы и диагностирования ошибок в работе функциональных компонентов, а также компонентов технического и программного обеспечения Платформы;
- устранения последствий, связанных со сбоями в работе Платформы;
- восстановления информации при ее потере средствами резервного копирования и восстановления.

1.4. Перечень эксплуатационной документации, с которой необходимо ознакомиться администратору

Администраторы обязаны до начала эксплуатации Платформы ознакомиться с эксплуатационной документацией, поставляемой с Платформой, включая настоящее руководство администратора.

2. Назначение и условия применения

2.1. Виды деятельности, функции, для автоматизации которых предназначена Платформа

Платформа состоит из Фреймворка и серверной инфраструктуры.

Фреймворк – набор инструментов, библиотек и правил, обеспечивающих создание и развертывание приложений на базе серверной инфраструктуры.

Фреймворк выполняет следующие функции:

- Создание и развертывание пользовательских приложений, сформированных по правилам, заданным Платформой.
- Предоставление возможности формализованного описания схем данных при помощи расширения языка SQL (VSQL).
- Описание (программирование) логики обработки данных (методов) на языке высокого уровня (GO) и формирование WASM кода.
- Создание пакета, содержащего компоненты пользовательского приложения (наборы VSQL файлов и сборки WASM).
- Развертывание пользовательских приложений на стороне серверной инфраструктуры.

Серверная инфраструктура – совокупность серверного программного обеспечения, обеспечивающего выполнение приложений, созданных на базе Фреймворка, а также предоставление прикладным компонентам сервисов, связанных со сбором, хранением, архивированием данных, конфигурированием, мониторингом и контролем за работой серверных компонент.

Серверная инфраструктура выполняет следующие функции:

- Конфигурирование инфраструктуры;
- Автоматизация развертывания и управления пользовательскими приложениями с поддержкой контейнеризации на базе Docker.
- Объединение узлов Платформы (Docker-хостов) в единый кластер и автоматическое управление запуском и масштабированием контейнеров на базе Docker Swarm.
- Выполнение пользовательских приложений, сформированных по правилам, определенными Платформой.

- Интерпретация формализованных описаний структур данных, методов работы с данными и других сведений, определенных правилами языка VSQL (расширение языка SQL).

    - Выполнение пакетов WASM, реализующих методы работы с объектами.

- Создание и модификация структур хранения данных, в соответствии с формализованными описаниями на языке VSQL.
- Распределение нагрузки между узлами, на которых развернута Платформа.
- Мониторинг событий, происходящих в компонентах Платформы.
- Уведомление и оповещение заинтересованных лиц и внешних систем о событиях, происходящих в компонентах Платформы.
- Прием из внешних систем запросов на получение данных и их исполнение.
- Прием из внешних систем команд на модификацию данных и их исполнение.
- Предоставление пользовательскому приложению механизмов надежного, распределенного и высокопроизводительного хранения данных на базе решения с открытым кодом Scylla.
- Предоставление пользовательскому приложению функций платформы с открытым исходным кодом для визуализации, мониторинга и анализа данных (Grafana):

    - Создание дашбордов с панелями, каждая из которых отображает определённые показатели в течение установленного периода времени.
    - Настройка дашбордов для конкретного приложения или с учётом любых потребностей разработки и/или бизнеса.
    - Поддержка различных источников данных (Prometheus, данные из внешних систем и т. д.).
    - Возможность использовать аннотации для отображения определённых событий на разных панелях.
    - Визуализация настраиваемой аналитики в виде круговых диаграмм, гистограмм времени и других графических элементов.

- Предоставление сервиса по сбору метрик через HTTP-вызовы к определенным конечным точкам, указанным в конфигурации (мониторингу внешних систем) на основе базы данных временных рядов Prometheus.

2.2. Условия применения Платформы в соответствии с назначением

2.2.1. Серверная инфраструктура

2.2.1.1 Комплекс технических средств

Требования к техническим средствам для размещения серверной части приведены в Табл. 1.

Табл. 1

Компонент	Минимальная конфигурация
Процессор	Не менее 4 ядер CPU
Объем оперативной памяти	Не менее 8 Гбайт
Свободное дисковое пространство	Не менее 20 Гбайт

2.2.1.2 Программное обеспечение

Для обеспечения работы серверной части Платформы используется программное обеспечение, перечень которого приведен в Табл. 2.

Табл. 2

Компонент	Конфигурация (используемое ПО)
Операционная система	Astra Linux Common Edition или Astra Linux Special Edition
Open-source веб приложение для аналитики и интерактивный визуализации.	Grafana (версия 8.3.4)*
Open-source приложение для сбора метрик с хостов и контейнеров.	Cadvisor (версия v0.47.1)*
Open-source приложение для мониторинга событий баз данных.	Prometheus (версия v2.44.0)*
Open-source приложение для оповещения о событиях в серверной инфраструктуре.	Alertmanager (версия v0.26.0)*
Open-source приложение для сбора метрик в операционной системе.	Node-exporter (версия v1.6.1)*
Open-source система управления БД	Scylla (версия 5.1.13)*
Программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой Контейнеризации и оркестратор для docker-контейнеров.	Docker, Docker Swarm*

*Компонент устанавливается утилитой «ctool»

2.2.2. Клиентская часть

2.2.2.1 Комплекс технических средств

Для работы с ПО рабочие станции пользователей должны удовлетворять требованиям к техническому обеспечению согласно Табл. 3.

Табл. 3

Компонент	Минимальная конфигурация	Рекомендуемая конфигурация
Частота процессора	1,0 ГГц	2,0 ГГц и более
Объем оперативной памяти	2 Гбайт	3 Гбайт и более
Свободное дисковое пространство	0,3 Гбайт	1 Гбайт и более

2.2.2.2 Программное обеспечение

На рабочих местах должно быть установлено программное обеспечение, перечень которого приведен в таблице Табл. 4.

Табл. 4

Компонент	Конфигурация
Операционная система	Любая, обеспечивающая работу web-браузера
Общесистемное ПО	Интернет браузер (Google Chrome, Mozilla Firefox, Microsoft Edge, Microsoft Internet Explorer, Safari, Yandex Browser, Спутник)

3. Подготовка к работе и описание операций

3.1. Состав и содержание носителя данных

Носитель данных для развертывания Платформы содержит:

- Архив с дистрибутивом Платформы;
- Руководство администратора.

До начала эксплуатации Платформы она должна быть развернута на масштабируемой отказоустойчивой платформе с применением технологий виртуализации и контейнеризации.

При подготовке к работе и применению Платформы загрузка данных с носителя не требуется.

3.2. Порядок развертывания Платформы

3.2.1. Общая информация

Утилита «ctool» позволяет развертывать и в дальнейшем изменять конфигурацию кластера Платформы двух типов:

- «Community Edition» («CE»)
- «Standart Edition» («SE»)

Кластер «CE» развертывается на одном хосте (сервере).

Кластер «SE» развертывается на пяти хостах (серверах). Два узла кластера при этом содержат приложение «nestore» и имеют роль «SENode», три остальных имеют роль «DBNode» и на них располагается СУБД «Scylla».

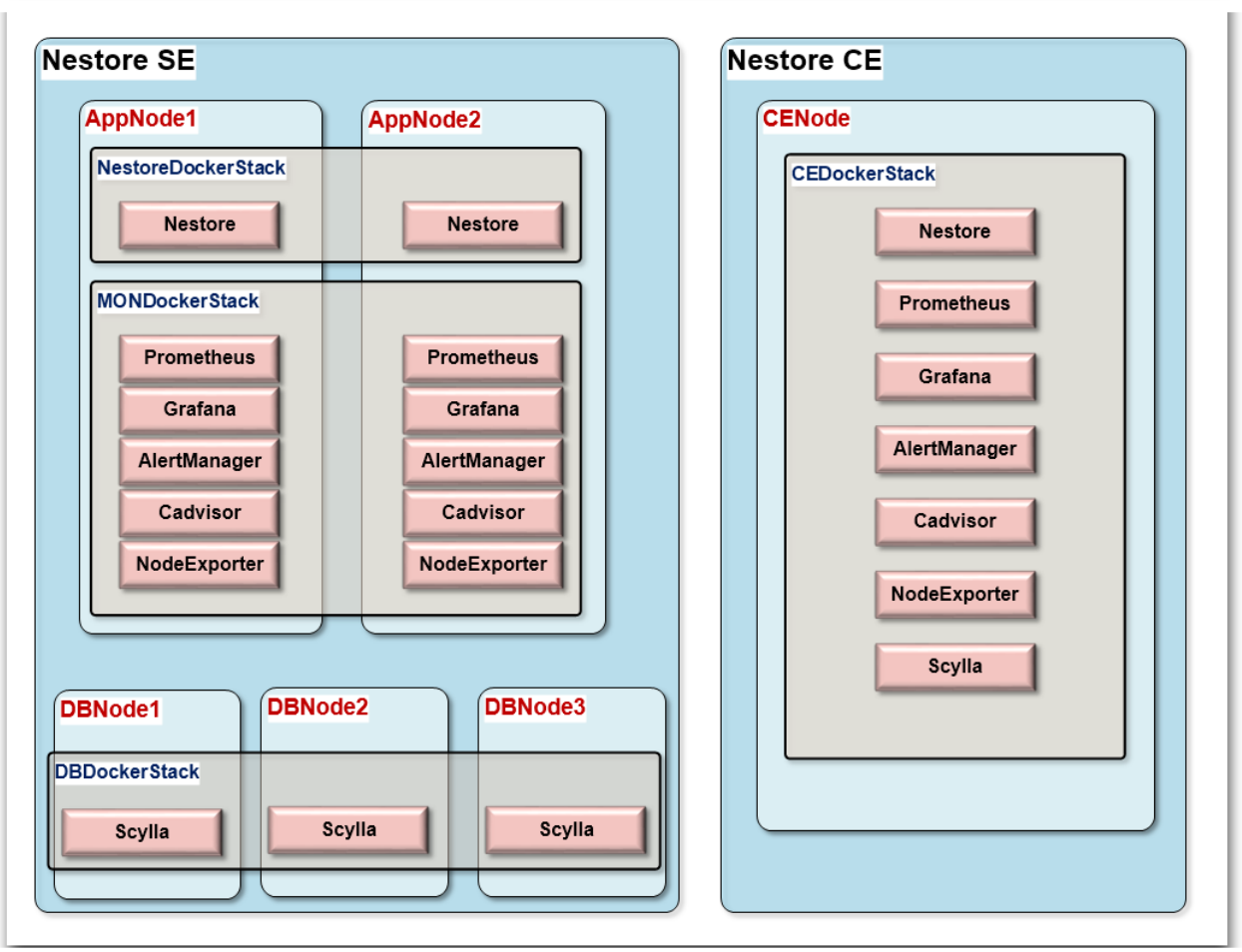

Рис. 1 Состав сервисов и их расположение в стеках

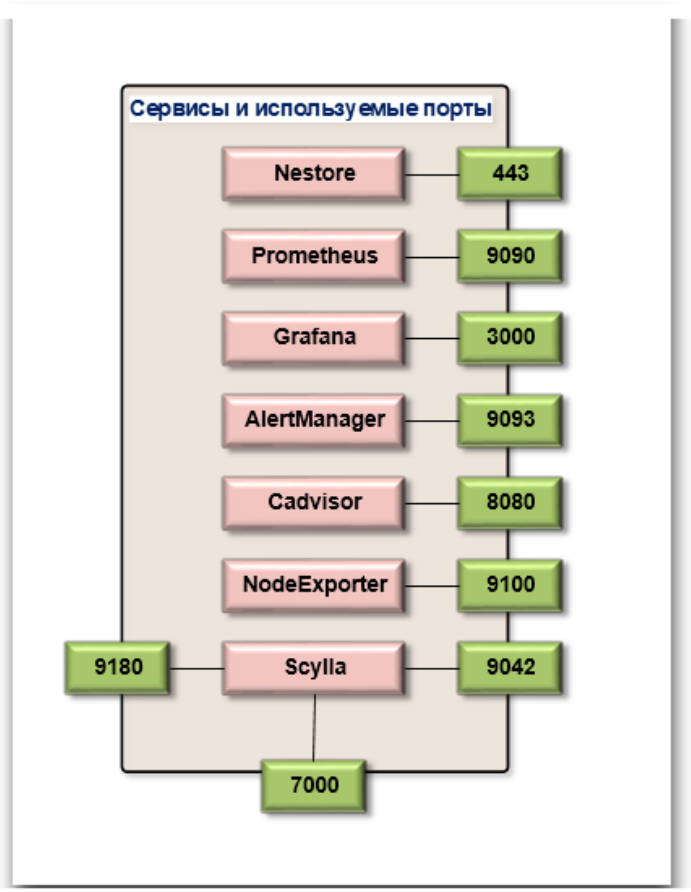

Рис. 2 Используемые порты

3.2.2. Требования к хостам (серверам) для развертывания на них Платформы

- Сервера должны быть "чистыми", содержащими ОС («Astra Linux Common Edition» или «Astra Linux Special Edition») без установленного дополнительного программного обеспечения. Соблюдение данного требования позволить использовать конфигурацию компонентов кластера с настройками "по умолчанию".
- Все компоненты кластера Платформы устанавливаются в Docker-контейнерах. Допускается наличие на хосте заранее установленного Docker'a, но только требуемой версии. Если Docker не установлен, то он будет установлен автоматически. Такое же требование предъявляется и к программному обеспечению Docker Compose.
- На всех хостах должен быть заведен одинаковый пользователь и должен использоваться одинаковый SSH-ключ для подключения.

Для развертывания Платформы используется инструмент «ctool».

3.2.3. Быстрый старт

В разделе кратко описаны основные команды для развертывания Платформы и изменения конфигурации кластера. Все IP-адреса указаны для примера. (SSH ключ пользователя находится в файле adm.key.).

3.2.3.1 Развертывание кластера «Community Edition» («CE»)

Развернуть кластер «Community Edition» («CE») на удаленном хосте 5.255.255.56:

```
$ ./ctool init CE 5.255.255.55 --ssh-key ./adm.key
```

Развернуть кластер «Community Edition» («CE») на локальном хосте:

```
$./ctool init CE 127.0.0.1
```

3.2.3.2 Развертывание кластера «Standart Edition» («SE»)

Развернуть кластер «Standart Edition» («SE») на пяти удаленных хостах (5.255.255.56, 5.255.255.57, 5.255.255.58, 5.255.255.59, 5.255.255.60):

```
$ ./ctool init SE 5.255.255.56 5.255.255.57 5.255.255.58 5.255.255.59 5.255.255.60 --ssh-key ./adm.key
```

Развернуть кластер «Standart Edition» («SE») на разных дата-центрах («dc1», «dc2», «dc3»):

```
$ ./ctool init SE 5.255.255.56 5.255.255.57 5.255.255.58 5.255.255.59 5.255.255.60 dc1 dc2 dc3 --ssh-key ./adm.key
```

Повторить выполнение, если возникла ошибка:

```
$ ./ctool repeat --ssh-key ./adm.key
```

Поднять версию кластера до версии «ctool»:

```
$ ./ctool upgrade --ssh-key ./adm.key
```

Валидация структуры файла «cluster.json»:

```
$ ./ctool validate
```

Заменить ноду кластера:

```
$ ./ctool replace 5.255.255.56 5.255.255.60 --ssh-key ./adm.key
```

3.2.3.3 Результат выполнения команд «ctool»

При выполнении команд: «init», «repeat», «replace» или «upgrade» создается файл «cluster.json», содержащий детальную информацию о конфигурации кластера и успешности его развертывания.

В текущей папке, в каталоге с именем «YYYY-MM-DD--HH-NN-SS-<commandName>» создается файл ***<commandName>.log»***, содержащий детальную информацию о ходе выполнения команды (лог).

3.2.4. SSH key

Если через менеджера ключей (ssh-agent) добавить необходимые ssh-ключи, то тогда при запуске команд «ctool» не требуется использовать флаг --ssh-key.

```
$ eval $(ssh-agent)

$ ssh-add ./adm.key

$ ./ctool init CE 5.255.255.55
```

Если этого не сделать, то для команд: «init», «repeat», «replace» и «upgrade» должен быть использован флаг --ssh-key:

```
$ ./ctool init CE 5.255.255.55 --ssh-key ./adm.key
```

3.2.5. Резервирование и восстановление базы данных

3.2.5.1 Резервирование кластера для DBNode

Для резервирования кластера используется следующая команда:

```
$ ./ctool backup node [<node> <target folder>] [flags]
```

- <node> - адрес или имя ноды,
- <target folder> - папка на ноде для выполнения команды.

Пример:

```
$ ./ctool backup node db-node-1 ~/backups/test-backup/ --ssh-key ~/adm.key
```

3.2.5.2 Выполнение резервирования данных по расписанию.

Для выполнения резервирования данных по расписанию необходимо задать в расписании для «Cron» выполнение следующей команды:

```
$ ./ctool backup cron [<cron event>] [flags]
```

где,

<cron event> - время выполнения в формате «cron» (@monthly или "* */24 * *").

flags - флаги:

- -e, --expire – время хранения предыдущих резервных копий (например, 7d, 1m).
- --ssh-key – путь к SSH key.

Если флаг –expire задан, тогда файлы резервных копий, созданные ранее указанного значения, будут удалены.

Пример команды выполнения резервного копирования:

```
$ ./ctool backup cron "* */24 * * *" --expire 30d --ssh-key ~/adm.key
```

Резервное копирование по расписанию производится на «db-node-1», «db-node-2» и «db-node-3» нодах в каталог «/mnt/backup/nestore/». Этот каталог должен существовать, и пользователь должен иметь права для чтения и записи в него данных.

При выполнении резервного копирования данных в каталоге /mnt/backup/nestore будет создан каталог yyyymmddhhnnss-backup.

3.2.5.3 Вывод списка существующих резервных копий

Для вывода списка существующих резервных копий используется команда:

```
$ ./ctool backup list [flags]
```

где flags:

- <--json> - вывод в файл формата JSON;
- <--ssh-key> - путь к SSH key.

Пример команды

```
$ ./ctool backup list --json --ssh-key ~/adm.key
```

3.2.5.4 Восстановление базы данных

Для восстановления данных используется команда:

```
$ ./ctool restore <backup name> [flags]
```

где,

<backup name> - имя папки, где находятся данные для восстановления. Если имя папки задано не полностью, то производится поиск данных на кластере DBNodes в каталоге /mnt/backup/voneedger/. Для выполнения восстановления данных необходимо что бы папка <backup name> существовала на всех кластерах DBNodes.

flags - флаги:

- --ssh-key – путь к SSH key

Пример команды восстановления данных:

```
$ ./ctool restore 20230624120000-backup --ssh-key ~/adm.key
```

3.2.6. Развертывание кластера

Для развертывания кластера Платформы используется команда «init».

В результате выполнения команды в текущей папке создается файл «cluster.json», содержащий параметры конфигурации кластера и признак успешного его развертывания. Последующий вызов команды init анализирует наличие и контент данного файла.

«cluster.json» - содержит данные о конфигурации кластера и признак успешного развертывания кластера данной конфигурации.

Также при каждом выполнении init в текущей папке создается папка с именем, содержащим текущие дату, время и имя команды init:

YYYYMMDDHHNN-init

В папке располагается файл init.log, содержащий подробный лог:

init.log - подробный лог операций по развертыванию кластера.

3.2.6.1 Развертывание кластера «Community Edition»

Кластер развертывается на одном хосте, соответственно, команда init будет выглядеть следующим образом:

```
$ ctool init CE [имя пользователя] [файл с SSH-ключом] [IP-адрес хоста]
```

Например,

```
$ ctool init CE root ./root.key 5.255.255.55
```

Для локальной установки CE нужно выполнить команду init CE с указанием адреса 127.0.0.1:

```
$ ctool init CE 127.0.0.1
```

3.2.6.2 Развертывание кластера Standart Edition (SE)

Кластер SE развертывается на пяти хостах (серверах).

Команда init для кластера SE будет выглядеть так:

```
$ ctool init SE [имя пользователя] [файл с SSH-ключом] SENode1_addr SENode2_addr DBNode1_addr DBNode2_addr DBNode3_addr
```

Например:

```
$ ctool init SE root ./root.key 5.255.255.56 5.255.255.57 dc1:5.255.255.58 dc2:5.255.255.59 dc3:5.255.255.60
```

Для назначения роли хостам кластера «SE» важен порядок их следования в списке аргументов команды «init». Первые два из указанных хостов будут иметь роль «SENode», остальные три будут иметь роль «DBNode».

Адреса хостов для узлов «DBNode» могут содержать имя дата центра в виде прификса через двоеточие. Если адрес хоста не содержит имя дата центра, то он принадлежит к последнему из указанных дата центров. Если дата центр не указан нигде, то это означает, что все узлы «DBNode» расположены в одном дата центре.

3.2.7. Изменение конфигурации кластера

Конфигурации кластеров имеют жесткие требования по количеству задействованных хостов. Для «CE» - это всегда один хост с узлом «CENode».

Для «SE» - это всегда два хоста с узлами «SENode» и три хоста с узлами «DBNode».

При таком подходе изменение конфигурации подразумевает только операцию переноса узлов кластера с одного хоста на другой.

Независимо от типа кластера и роли узла на хосте, перенос узла с текущего хоста на новый выполняется командой «relocate»:

```
$ ctool relocate [имя пользователя] [файл с SSH-ключом] [IP-адрес текущего хоста] [IP-адрес нового хоста]
```

Если хост имеет роль «DBNode», то его адрес может содержать префикс с именем дата центр через двоеточие.

Например:

```
$ ctool relocate root ./root.key dc1:5.255.255.56 dc1:5.255.255.60
```

Также при каждом выполнении «relocate» в текущей папке создается папка с именем, содержащим текущие дату, время и имя команды «relocate»:

YYYYMMDDHHNN-relocate

В папке располагается файл «relocate.log», содержащий подробный лог:

relocate.log - подробный лог операций по переносу узла кластера на новый хост.

При удачном выполнении команды «relocate» обновленная конфигурация кластера записывается в файл cluster.json.

cluster.json - содержит данные о конфигурации кластера и признак успешного развертывания кластера данной конфигурации.

3.2.8. Технический дизайн

Каждой команде утилиты «ctool» соответствует сценарий ее выполнения: «playbook».

Сценарий «playbook» содержит последовательность шагов «sequence».

Каждый шаг «step» содержит (всё или выборочно из списка):

- функцию выполнения «run»;
- список последовательностей шагов «sequenceList»;
- завершающую функцию «deferRun».

Алгоритм выполнения шага «step» «step.execute»:

- вызывается функция «run»;
- параллельно запускаются на выполнение, каждая в своем потоке, последовательность шагов «sequence» из списка «sequenceList»;
- ожидается завершение работы потока;
- вызывается функция «deferRun».

Каждый шаг «step» логически объединяет определенный набор действий, который можно считать этапом выполнения команды.

Шаг «step» имеет описание «descriptions» и индекс «idx». Таким образом, осуществляется выполнение каждой последовательности шагов.

«sequence» можно сопровождать "простым" логом с отображением прогресса её выполнения. В случае включенной опции «verbose» логирование выполнения шага может быть подробным.

Сценарии «playbook» связываются с соответствующими командами и регистрируются в поставщике «playbookProvider».

Выполнением команд занимается «ctoolEngine».

Алгоритм выполнения команд «ctoolEngine»:

- получает команду (*cobra.Command);
- запрашивает у «playbookProvider» зарегистрированный для команды сценарий выполнения «playbook»;
- запускает «playbook» на выполнение «playbook.run», передав ему необходимый набор callback функций (или других механизмов) для мониторинга и логирования процесса выполнения сценария.

Если выполнение сценария не штатно прерывается, то «ctoolEngine» может возобновить его работу с определенного шага, опираясь на статус его предыдущего выполнения. При возобновлении работы сценария надо понимать, что он может содержать шаги, обязательные к выполнению даже, если при предыдущей работе они были выполнены (например, установка ssh-соединения). Такие шаги должны быть выполнены повторно.

Структура сценариев «playbook» позволяет обеспечить следующие принципы:

- Шаги («step») в рамках одной последовательности «sequence» всегда выполняются последовательно друг за другом.
- Каждая последовательность шагов («sequence») выполняется в отдельной горутине. Соответственно последовательности «sequence» из списка «sequenceList» выполняются параллельно друг другу.
- Шаг «step», содержащий список последовательностей «sequenceList», будет завершен только после завершения работы всех последовательностей из списка.
- Ошибка при выполнении очередного шага прерывает выполнение последовательности с ошибкой.

При таком подходе сценарий может параллельно выполняться на нескольких узлах кластера, при этом «ctoolEngine», будет централизовано вести лог сценария и отслеживать его статус выполнения.

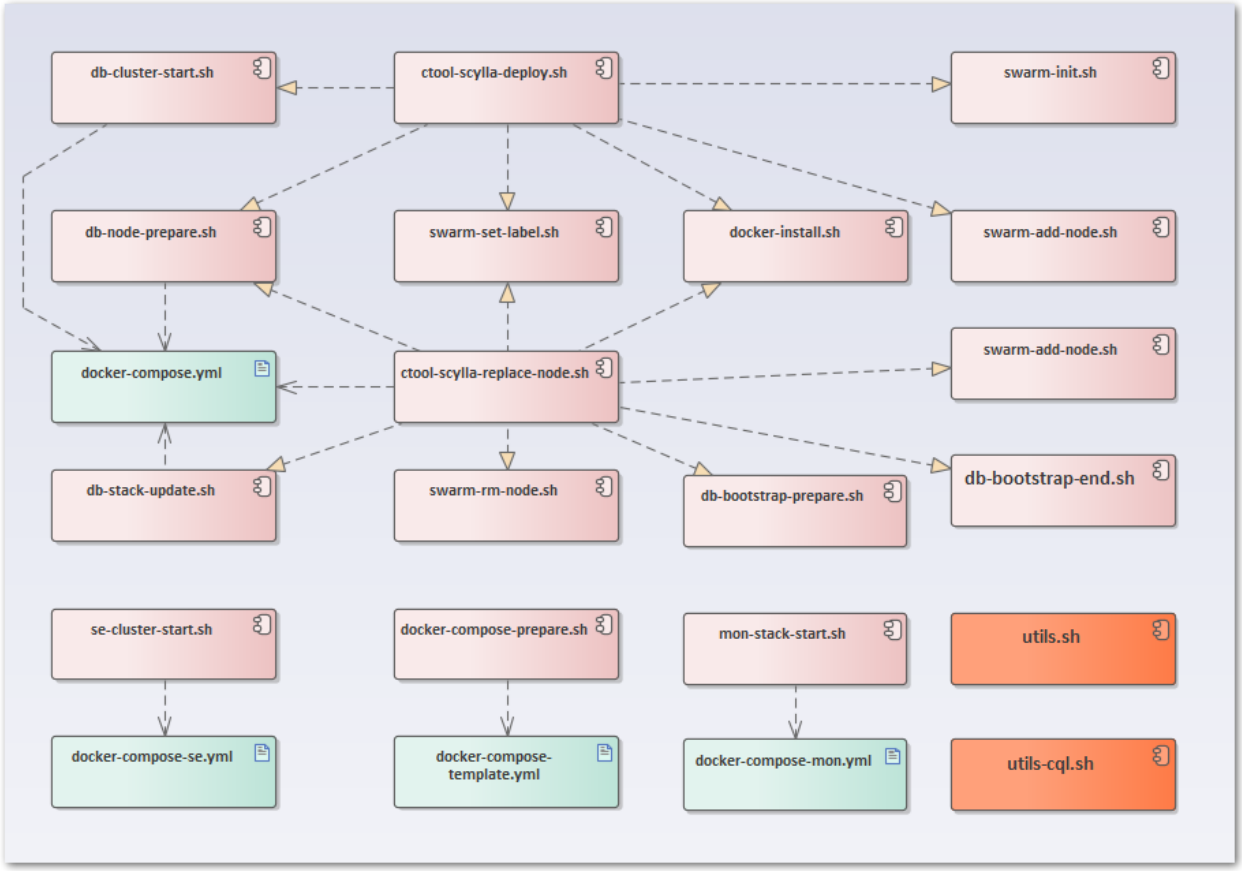

Рис. 3 Зависимости между скриптами и файлами настройки

3.2.9. Команда upgrade

- версия «ctool» связана со значением из файла «version»;
- при запуске «ctool» структура кластера зачитывается из «cluster.json» для сравнения версий «ctool», «ActualClusterVersion» и «DesiredClusterVersion»;

- если в «cluster.json» есть незавершенная команда и «DesiredClusterVersion» отличается от версии «ctool», то работа «ctool» прекращается с сообщением о необходимости установки «ctool» версии «DesiredClusterVersion» для завершения команды;
- если в «cluster.json» «ActualClusterVersion» старше чем версия «ctool», то работа «ctool» прекращается с сообщением о необходимости установки «ctool» версии «ActualClusterVersion» для продолжения работы;
- если в «cluster.json» «ActualClusterVersion» не пустая и она младше, чем версия «ctool», то выводится сообщение о необходимости выполнения команды «upgrade». Любые другие команды, изменяющие конфигурацию кластера, можно будет выполнять только после удачного завершения команды «upgrade»;
- алгоритм выполнения команды «upgrade» аналогичен алгоритму команды «init». «Upgrade» производит развертывание кластера поверх развернутого ранее. Эта операция переустанавливает, если это необходимо, docker стеки. Пользовательские данные не затрагиваются.

3.3. Структура и работа с файлом cluster.json

3.3.1. Пример файла cluster.json

```
{
// Тип установки
  "EditionType": "SE",
  // Текущая версия ctool
  "ActualClusterVersion": "0.0.2",
  // Желаемая версия CTool (Используется для команды UPGRADE)
  "DesiredClusterVersion": "0.0.3",

  // Команда, которую необходимо выполнить
  "Cmd": {
    "Kind": "Имя команды",
    "Args": "Аргументы, разделенные запятыми"
  },

  // Сообщение об ошибке при последнем выполнении (Присутствует, если команда была выполнена с ошибками)
  "LastAttemptError": "сообщение об ошибке",

  // Список компонентов ноды
  "Nodes": [
    {
      // Роль ноды
      "NodeRole": "SENode",
```

```
    // Сведения о последней попытке выполнения
     "Info": "some info",

     // Текущее состояние ноды
     "ActualNodeState": {
       "Address": "5.255.255.55",
       "NodeVersion": "0.0.2"
     },

     // Ошибка при последней попытке
     "Error": "some error",

    // Желаемое состояние ноды
     "DesiredNodeState": {
       "Address": "5.255.255.55",
       "NodeVersion": "0.0.2"
     }
    },
  ],

  // Список заменяемых адресов (заполняется при перемещении ноды)
  "ReplacedAddresses": [
    "10.0.0.28",
    "10.0.0.29"
  ]
}
```

3.3.2. Успешное развертывание кластера SE

Пример фрагмента `cluster.json` с комментариями.

В полной версии файла `cluster.json` должно быть описано 5 нод.

```
{
  // Тип кластера SE
  "EditionType": "SE",

  // Версия ctool
  "ActualClusterVersion": "0.0.2",

  "Nodes": [
    {
      // Роль ноды
      "NodeRole": "SENode",

      // Информация о последней попытке выполнения
      // Не заполняется, если сведения отсутствуют
      "Info": "информация"

      // Актуальное состояние ноды
      "ActualNodeState": {
        "Address": "5.255.255.55",
```

```
      "NodeVersion": "0.0.2"
    }

  },
 ]
}
```

3.3.3. Ошибки при развертывании SE

Пример фрагмента `cluster.json` с комментариями.

В полной версии файла `cluster.json` должно быть описано 5 нод.

```
{
 // Тип ноды
 "EditionType": "SE",

 // Версия ctool
 "ActualClusterVersion": "",
 "DesiredClusterVersion": "0.0.2",

 // Команда для выполнения
 "Cmd": {
  "Kind": "имя команды",
  "Args": "аргументы"
 },

 // Не заполняется, если пусто
 "LastAttemptError": “Описание ошибки",

 // Список дата-центров
 // Присутствует, если развертывание происходит на нескольких дата-центрах
 "DataCenters": [
  "dc1", // относится к Nodes[2]
  "dc2", // относится к Nodes[3]
  "dc3"  // относится к Nodes[4]
 ],

 "Nodes": [
  {
   // Роль ноды
   "NodeRole": "SENode",


   // Ошибка при последнем выполнении
   // Пусто, если не было ошибки
   "Error": "Описание ошибки"

   "ActualNodeState": {
    "Address": "",
    "NodeVersion": ""
   }
   "DesiredNodeState": {
```

```
        "Address": "5.255.255.55",
        "NodeVersion": "0.0.2"
      }

    },
  ]
}
```

3.3.4. Замена ноды

Пример фрагмента «cluster.json» с комментариями.

Замена ноды «DBNode» с адреса 10.0.0.21 на адрес 10.0.0.22.

Для успешного выполнения команды заменяемый IP-адрес добавляется в «Replacedaddresses» список и записывается в «cluster.json».

В полной версии файла «cluster.json» должно быть описано 5 нод.

```
{
  // Тип кластера
  "EditionType": "SE",

  // Версия ctool
  "ActualClusterVersion": "0.0.2",

  // Команда Replace для применения
  "Cmd": {
    "Kind": "replace",
    "Args": "10.0.0.21 10.0.0.22"
  },

  "Nodes": [
    {
      // Роль ноды
      "NodeRole": "DBNode",

      "ActualNodeState": {
        "Address": "10.0.0.21",
        "NodeVersion": "0.0.2"
      }
      "DesiredNodeState": {
        "Address": "10.0.0.22",
        "NodeVersion": "0.0.2"
      }

    },
  ],
  "ReplacedAddresses": [
    "10.0.0.28",
    "10.0.0.29"
  ]
}
```

3.3.5. NodeControllerFunction

Выполняется, если есть секция «DesiredNodeState»

- Optionally update docker;
- Assign ActualNodeState = DesiredNodeState;
- Remove DesiredNodeState;

3.3.6. ClusterCntrollerFunction

Выполняется, если есть секция «Cmd»

- DeploySwarm();
- UpdateSEDockerStack();
- UpdateDBScyllaDockerStack();
- UpdateMonDockerStack();
- Assign ActualClusterVersion = DesiredClusterVersion;
- Remove Cmd.

3.3.7. Актуализация версии

Пример фрагмента файла «cluster.json» с командой «upgrade».

В полной версии файла «cluster.json» должно быть описано 5 нод.

```
{
 // Тип кластера
 "EditionType": "SE",

 // Версия ctool
 "ActualClusterVersion": "0.0.2",
 "DesiredClusterVersion": "0.0.3",

 // Команда upgrade
 "Cmd": {
  "Kind": "upgrade",
  "Args": ""
 },

 "Nodes": [
  {
   // Node role
   "NodeRole": "SENode",

   "ActualNodeState": {
    "Address": "5.255.255.55",
    "NodeVersion": "0.0.2"
   }
   "DesiredNodeState": {
```

```
        "Address": "5.255.255.55",
        "NodeVersion": "0.0.3"
      }

    },
  ]
}
```

3.4. VSQL

VSQL основан на традиционном SQL (Structured Query Language) и используется для взаимодействия с базой данных.

Основные различия между VSQL и SQL:

- Изменяемость: VSQL устраняет необходимость в инструкции ALTER, оптимизируя модификации схемы, с автоматическими обновлениями без простоев.
- Исключение NULL: таблицы VSQL не поддерживают значения NULL, что упрощает обработку данных за счет исключения операций, связанных с NULL.
- Упрощенное использование ссылок.
- Наследование таблиц.
- Поддержка концепции рабочего пространства.
- Поддержка концепции вложенной таблицы.
- Поддержка концепции абстрактной таблицы.
- Новые типы данных: Any, raw.
- Возможность изменения.

VSQL не использует инструкцию «ALTER TABLE». Вместо этого обновления схемы вносятся непосредственно в описание «TABLE». Во время развертывания новые схемы применяются автоматически.

Рассмотрим пример эволюции схемы.

```
MyTable, версия 1:
TABLE MyTable (
   Field1 int NOT NULL,
   Field2 int,
   Field3 int
);
```

Обновлено до версии 2 путем добавления поля Field4:

```
TABLE MyTable (
   Field1 int,                  -- NOT NULL убрано
   Field2 int NOT NULL,         -- добавлено NOT NULL
   Field3 int,
```

```
    Field4 int                    -- добавлено поле Field4
);
```

Разрешены только изменения, обеспечивающие обратную совместимость. Например, нельзя изменить тип существующего поля, удалить поле или изменить порядок расположения полей. Совместимость можно проверить во время разработки с помощью команды «$ vpm compat».

Работа с NULL-значениями

Возможность столбца базы данных принимать значения NULL.

В VSQL значения столбцов не могут быть NULL, что устраняет необходимость в IS NULL, COALESCE() и других функциях, связанных с NULL.

Платформа не применяет NOT NULL атрибут при обновлении записей (и, конечно, нельзя указывать значения NULL).

Добавление нового поля с NOT NULL атрибутом в существующую таблицу означает, что запрос этого поля в существующих записях вернет значение по умолчанию: 0 для целочисленных полей, "" (пустая строка) для строковых полей, false для логических полей и т.д.

3.5. Порядок проверки работоспособности

Проверка корректности функционирования Платформы осуществляется обращением с https запросом по адресу https://host/static/sys/monitor/site/hello или http://host:8082/static/sys/monitor/site/hello .

Получение кода возврата «200» говорит об успешности функционирования Платформы.

4. Аварийные ситуации

4.1. Действия при возникновении ошибок в процессе конфигурирования и работы Платформы

В Табл. 5 указаны возможные сообщения об ошибках и действия по их устранению в процессе конфигурирования и работы Платформы.

Табл. 5

Сообщение об ошибке	Действия при возникновении ошибки
"invalid cluster edition (expected SE or CE)"	Ошибка в наименовании типа кластера. Допустимые значения «SE» или «CE». Необходимо исправить текст команды и

	повторить ее выполнение.
"invalid number of data centers"	Количество дата-центров, указанных в команде некорректно. Необходимо исправить текст команды и повторить ее выполнение.
"cluster configuration not found"	Файл, описывающий конфигурацию кластера, не найден. Проверьте наличие файла и повторите команду.
"invalid number of arguments"	Количество аргументов в команде не соответствует ожидаемому. Необходимо исправить текст команды и повторить ее выполнение.
"invalid IP-address"	Ошибка в IP-адресе. Необходимо исправить текст команды и повторить ее выполнение.
"unknown command"	Неизвестная команда. Необходимо исправить текст команды и повторить ее выполнение.
"missing command arguments"	Указана команда, но отсутствуют необходимые аргументы для ее выполнения. Необходимо исправить текст команды и повторить ее выполнение.
"no upgrade required"	Уже выполнены необходимые процедуры по подъему версии до актуального значения. Действий не требуется.
"host _____ already exists in cluster"	Указанный хост уже существует в кластере. Поправьте имя хоста и повторите команду.
"uncompleted command found"	Найдена незавершенная команда. Необходимо проанализировать лог выполнения команды и определить причину ошибочной ситуации.
"error preparing cluster nodes, command is aborted"	Ошибка подготовки ноды в кластере. Команда прервана. Необходимо проанализировать лог выполнения команды и определить причину ошибочной ситуации.
"no incomplete command was found to repeat"	Попытка повторить команду, которая была выполнена успешно. Действий не требуется.
"ctool version ____ is newer than cluster version ____	Версия программы «ctool» более новая, чем версия кластера. Необходимо использовать «ctool»

	соответствующий версии кластера.
"cluster version ___ is newer than ctool version ____ you should use the ctool version ____"	Версия программы «ctool» более новая, чем версия кластера. Необходимо использовать «ctool» версии, указанной в сообщении об ошибке.
"node version ____ do not match the cluster version ____"	Номер версии ноды не соответствует номеру версии кластера. Необходимо привести номера версий ноды и кластера в идентичное состояние.
"invalid node role"	В команде некорректно указана роль ноды. Необходимо исправить текст команды и повторить ее выполнение.
"empty IP-address"	В команде не указан IP – адрес. Необходимо исправить текст команды и повторить ее выполнение.
"command cannot be executed"	Команда не может быть выполнена. Необходимо проанализировать лог-файл выполнения команды и определить причину ошибки. Затем, повторно выполнить команду.
"host is not available"	Хост не найден. Проверьте правильность имени хоста в команде и повторите ее выполнение.
"address cannot be used"	Указанный в команде адрес не может быть использован. Исправьте команду и повторите ее выполнение.
"host not found"	Хост не найден. Ошибка в адресе хоста. Исправьте команду и повторите ее выполнение.
"host ___ not found in cluster: ___"	Хост не найден в указанном кластере. Исправьте команду и повторите ее выполнение.
"file not found"	Файл не найден. Проверьте наличие необходимых для работы файлов и повторите выполнение команды.
"ssh key ___ not found: ___"	Ключи в системе не установлены или ошибка в тексте команды. Установите ключи или исправьте команду, затем повторите ее выполнение.
"domains not found"	Указанные в команде доменные имена не найдены. Исправьте команду и повторите ее выполнение.

"domains ____ not found in cluster: ____"	Указанные в команде доменные имена не найдены в указанном кластере. Исправьте команду и повторите ее выполнение.
"backup folder error"	Ошибка в имени каталога, который должен содержать данные для восстановления. Исправьте команду и повторите ее выполнение.
"backup folder " _____ " is not prepared on host _____"	Каталог для резервной копии не создан на указанном хосте. Создайте необходимый каталог и повторите команду.
"invalid expire time"	Период времени, указанный в команде, не верен. Исправьте команду и повторите ее выполнение.
"backup does not exist"	Каталог, который должен содержать данные для восстановления, не существует. Исправьте команду и повторите ее выполнение.
"backup _____ does not exist on host ____"	Каталог, который должен содержать данные для восстановления, не существует на указанной ноде. Исправьте команду и повторите ее выполнение.
"password must be at least ____ characters long"	Длина пароля не достаточна. Увеличьте длину пароля и повторите команду.
"is not a valid URL"	Указанный URL не корректный. Поправьте значение и повторите команду.

4.2. Действия при отказах технических средств

В случае выхода из строя технических средств следует действовать в порядке, изложенном в документации производителей технических средств.

5. Рекомендации по освоению

Перед началом работы с Платформой необходимо ознакомиться с эксплуатационной документацией, поставляемой с Платформой.

В разделе 1.3 настоящего документа описаны минимальные требования к уровню подготовки администратора.