ПЛАТФОРМА КОНТЕЙНЕРИЗАЦИИ БОЦМАН

Руководство администратора

Листов 57

2025

СОДЕРЖАНИЕ

АННОТАЦИЯ

Настоящее руководство администратора распространяется на программное изделие РДЦП.10501-01 «Платформа контейнеризации Боцман» (далее по тексту — Боцман), предназначенное для создания вычислительных кластеров Kubernetes и управления ими в различных видах инфраструктуры, а также для их обеспечения необходимыми для работы приложений дополнениями.

Областью применения Боцман является автоматизация деятельности системных администраторов в рамках развертывания и эксплуатации вычислительных кластеров Kubernetes.

В данном документе приведено описание технических средств для установки Боцман, порядок его установки, настройки, работы и удаления.

Документ предназначен для администраторов и разработчиков доменной инфраструктуры.

1. ОБЩИЕ СВЕДЕНИЯ

1.1. Назначение и условия применения

Боцман построен на базе открытого программного обеспечения Kubernetes. Данный программ- ный комплекс предназначен для автоматизации создания, настройки и развертывания кла- стеров Kubernetes в условиях закрытого программного контура без доступа к сети Интернет. Также Боцман предоставляет такие возможности, как сканирование образов и контейнеров на уязвимости и централизованную регистрацию событий в кластере.

Администрирование развернутого кластера Боцман и развертывание в нем приложений осуществляется с помощью программного обеспечения Kubernetes в соответствии с его руководством. Приложения представляют собой контейнеры, запущенные из образов (ар- хивы, в которые входит само приложение и необходимые для его запуска окружение и зависимости).

Кластер Боцман состоит из машин, выполняющих роли управляющих и рабочих узлов. Управ- ляющие узлы осуществляют распределение ресурсов в кластере, развертывание приложе- ний на рабочих узлах, контролируют работу приложений и обеспечивают их автоматический перенос между узлами кластера при нехватке ресурсов на узлах или при отказе узлов. Для развертывания простых приложений в кластере достаточно, чтобы он состоял из двух узлов - одного управляющего и одного рабочего. Однако, такой кластер не будет отказоустой- чивым. Для обеспечения отказоустойчивости в кластере должно быть как минимум два управляющих и два рабочих узла.

Боцман состоит из следующих компонентов, обеспечивающих работу кластера:

- `nodusd` — служба, осуществляющая подготовку узлов в процессе установки и связь между ними в процессе эксплуатации развернутого кластера;
- `nodus-agent` — служба для централизованного сбора и записи в журнал регистрируемых в кластере событий безопасности;
- `crio` — приложение для запуска контейнеров и их сканирования на уязвимости;
- прочие служебные приложения, входящие в состав программного обеспечения Kubernetes.

1.2. Функциональные возможности

Боцман предоставляет следующие возможности:

- подготовка шаблонов конфигурационных файлов для развертывания кластера;

- автоматическая установка и удаление компонентов кластера на узлах по заранее подготовленному конфигурационному файлу с использованием установочного архива;
- добавление и удаление узлов в работающем кластере;
- регистрация событий безопасности на узлах кластера с их централизованным сбором и сохранением в высокоцелостный журнал;
- сканирование образов и контейнеров на уязвимости.

1.3. Требования к техническим средствам

Машины, выполняющие роль узлов кластера, должны соответствовать следующим требованиям:

- узлы кластера должны быть доступны по сети между собой;
- на данных машинах должен быть настроен доступ по SSH;
- на данных машинах должен быть отключен мандатный контроль целостности.

Для функционирования управляющего узла необходимо следующее программное обеспечение и минимальная конфигурация оборудования:

- ОС СН с обновлением 1.7.5 или новее;
- аппаратная платформа — процессор с архитектурой x86-64 (AMD, Intel);
- центральный процессор — не менее 2 ядер;
- оперативная память — не менее 2 ГБ;
- объем свободного дискового пространства (рекомендуется использовать SSD) — не менее 20 ГБ.

Для функционирования рабочего узла необходимо следующее программное обеспечение и минимальная конфигурация оборудования:

- ОС СН с обновлением 1.7.5 или новее;
- аппаратная платформа — процессор с архитектурой x86-64 (AMD, Intel);
- центральный процессор — не менее 1 ядра;
- оперативная память — не менее 1 ГБ;
- объем свободного дискового пространства (рекомендуется использовать SSD) — не менее 20 ГБ.

1.4. Последовательность развертывания

Развертывание кластера и последующее управление им с помощью Боцман и клиентского приложения Kubernetes — `kubectl` может осуществляться с любой машины, входящей в будущий кластер, или с отдельной машины, имеющей доступ по сети к остальным машинам.

Создание кластера выполняется в следующей последовательности:

1) установка Боцман (см. 4.1);
2) добавление установочного архива (см. 4.2.1);
3) создание конфигурационного файла (см. 5.1);
4) установка службы `nodusd` (см. 5.2);
5) установка компонентов кластера (см. 5.3).

2. ДЕЙСТВИЯ ПО РЕАЛИЗАЦИИ ФУНКЦИЙ БЕЗОПАСНОСТИ СРЕДЫ ФУНКЦИОНИРОВАНИЯ

Требования к обеспечению безопасности среды функционирования, а также настройка параметров, необходимых для безопасной эксплуатации ОС СН, приведены в документе РУСБ.10015-01 97 01-1 «Операционная система специального назначения «Astra Linux Special Edition». Руководство по КСЗ. Часть 1» и выполняются администратором безопасности.

В целях обеспечения безопасности среды функционирования должны быть выполнены следующие требования:

- средства вычислительной техники (далее по тексту — СВТ), предназначенные для установки и функционирования Боцман , должно соответствовать требованиям, указанным в документе РУСБ.10015-01 31 01 «Операционная система специального назначения «Astra Linux Special Edition». Описание применения» и в документе РДЦП.10501-01 31 01 «Платформа контейнеризации «Боцман ». Описание примене- ния»;
- при развертывании Боцман в виртуальной среде для создания защищенной среды виртуализации ОС СН должна применяться совместно с аппаратным обеспечением, поддерживающим соответствующие технологии VT/SVM, в том числе при необходи- мости предоставления доступа виртуальным машинам к физическим устройствам;
- установка, управление и конфигурирование ОС СН должны проводиться в соответствии с РУСБ.10015-01 95 01-1 «Операционная система специального назначения «Astra Linux Special Edition». Руководство администратора. Часть 1»;
- должна быть обеспечена защита от осуществления действий, направленных на нарушение физической целостности СВТ, на котором функционирует ОС СН;
- должна быть обеспечена доверенная загрузка ОС СН. Доверенную загрузку ОС СН рекомендуется выполнять с помощью сертифицированных средств доверенной загрузки или аппаратно-программных модулей доверенной загрузки. При технической невозможности или нецелесообразности использования таких средств должны быть приняты организационно-технические меры, предотвращающие возможность доступа пользователя к ресурсам СВТ в обход механизмов защиты ОС СН (должна быть исключена возможность загрузки альтернативной операционной системы на СВТ, а также модификация модулей загружаемой ОС СН);
- должна быть обеспечена защита от несанкционированного изменения исполняемых файлов формата ELF и предотвращение их загрузки путем использования средств создания замкнутой программной среды ОС СН. Механизм ЗПС реализован в виде невыгружаемого модуля ядра ОС СН, выполняющего проверку электронной цифровой подписи файлов;

- необходимо обеспечить доверенный канал передачи информации между СВТ при сетевом взаимодействии, на которых установлена ОС СН (например, контроль несанкционированного подключения к ЛВС в пределах контролируемой зоны, защищенная передача сетевого трафика за пределами контролируемой зоны).

3. ПОДГОТОВКА РАБОЧЕЙ МАШИНЫ

4.1. Установка Боцман

Для установки Боцман необходимо выполнить следующие действия:

1) скопировать на машину исполняемый файл `Боцман` и установочный архив `nodus-installer-bundle.tar.gz`;
2) пометить файл `Боцман` как исполняемый, выполнив команду:

```
sudo chmod +x ./nodus
```

3) добавить каталог расположения исполняемого файла `Боцман` в переменную `PATH`, выполнив команду:

```
echo "export PATH="${PATH}:/путь/к/nodus"" >> ~/.bashrc && source ~/.bashrc
```

4.2. Управление установочными архивами

4.2.1. Добавление установочных архивов

Для развертывания кластера нужен установочный архив, содержащий все необходимые файлы и пакеты. Для использования установочного архива, его необходимо добавить в Боцман . Для добавления установочного архива из локального файла используется команда:

```
nodus bundle add
```

Синтаксис команды:

```
nodus bundle add <путь_к_архиву> [параметры]
```

Параметры команды приведены в таблице 1

Т а б л и ц а 1

Параметр	Описание
`-f, --force`	Перезаписать имеющийся архив той же версии
`-h, --help`	Вывести справку по команде

Добавляемый архив копируется в каталог:

```
/home/<имя_пользователя>/.config/nodus/bundles
```

Содержимое архива дополнительно распаковывается в каталог:

```
/home/<имя_пользователя>/.config/nodus/bundles/unpacked/<версия_архива>
```

Для вывода списка версий добавленных архивов используется команда:

```
nodus bundle ls
```

Результат выполнения команды:

```
v0.0.4
v1.1.0
v1.1.1
```

> П р и м е р
>
> Добавление установочного архива из локального файла с перезаписью уже имеющегося архива той же версии:
>
> ```
> nodus bundle add /tmp/nodus-bundle.tar.gz -f
> ```

4.2.2. Удаление добавленного архива

Для удаления из Боцман ранее добавленных архивов используется команда:

```
nodus bundle rm
```

Синтаксис команды:

```
nodus bundle rm [версия] [параметры]
```

Параметры команды приведены в таблице 2

Т а б л и ц а 2

Параметр	Описание
`-a, --all`	Удалить все добавленные архивы
`-h, --help`	Вывести справку по команде

П р и м е р ы:

1. Удаление архива версии `v0.0.4`:

```
nodus bundle rm v0.0.4
```

2. Удаление всех добавленных архивов:

```
nodus bundle rm -a
```

5. РАЗВЕРТЫВАНИЕ КЛАСТЕРА

5.1. Регистрация конфигурации кластера

5.1.1. Создание конфигурационного файла

Для создания кластера и установки необходимых программных компонентов на его узлы с помощью Боцман требуется конфигурационный файл кластера, написанный на языке разметки YAML. В данном конфигурационном файле перечисляются узлы развертываемого кластера с их сетевыми адресами, учетные данные или сертификаты для сетевого доступа по протоколу SSH. Также в нем возможно переопределить настройки компонентов кластера.

Полная структура конфигурационного файла кластера и описание основных настраиваемых компонентов кластера приведены в Приложении А.

Боцман позволяет создать шаблон конфигурационного файла кластера для его дальнейшего редактирования. Также Боцман позволяет создавать шаблоны специфичных значений helm- chart для более точной настройки параметров устанавливаемых компонентов кластера. Пути к файлам шаблонов helm-chart указываются в разделе `settings` конфигурационного файла кластера, в подразделах соответствующих компонентов. Параметры в данных файлах имеют приоритет над параметрами, указанными в конфигурационном файле кластера.

Для создания шаблона конфигурационного файла кластера используется команда:

```
nodus template cluster
```

Синтаксис команды:

```
nodus template cluster [параметры]
```

Описание параметров команды приведено в таблице 3.

Т а б л и ц а 3

Параметр	Описание
`--inventory=<"файл">`	Путь к файлу с разделом настроек `Inventory`, которые будут включены в создаваемый шаблон конфигурационного файла кластера
`-m, --masters <число>`	Количество управляющих узлов в кластере (по умолчанию 3)
`-w, --workers <число>`	Количество рабочих узлов в кластере (по умолчанию 3)
`--kubeadm-init=<"файл">`	Путь к файлу с настройками инициализации управляющего узла кластера, которые будут включены в создаваемый шаблон конфигурационного файла кластера

Окончание таблицы 3

Параметр	Описание
`--local-setup`	Добавление в шаблон настроек локального репозитория
`-o, --output=<"файл">`	Путь к сохраняемому файлу шаблона. Если параметр не указан, то содержимое шаблона выводится в консоль
`-h, --help`	Вывод справки по команде

Для создания шаблона конфигурационного файла для добавления узлов в существующий кластер (см. 6.1) используется команда:

```
nodus template join
```

Синтаксис команды:

```
nodus template join [параметры]
```

Описание параметров команды приведено в таблице 4.

Т а б л и ц а 4

Параметр	Описание
`--inventory=<"файл">`	Путь к файлу с разделом настроек `Inventory`, которые будут включены в создаваемый шаблон добавления узлов в кластер
`-m, --masters <число>`	Количество управляющих узлов, добавляемых в кластер (по умолчанию 3)
`-w, --workers <число>`	Количество рабочих узлов, добавляемых в кластер (по умолчанию 3)
`-o, --output=<"файл">`	Путь к сохраняемому файлу шаблона. Если параметр не указан, то содержимое шаблона выводится в консоль
`-h, --help`	Вывести справку по команде

Для создания шаблонов helm-chart используется команда:

```
nodus template helm
```

Синтаксис команды:

```
nodus template helm <действие> [параметры]
```

Описание действий команды приведено в таблице 5.

Таблица 5

Действие	Описание
`cilium`	Создание шаблона специфичных значений (custom values) helm-chart для компонента `cilium`
`nodelocaldns`	Создание шаблона специфичных значений (custom values) helm-chart для компонента `nodelocaldns`
`nodus-agent`	Создание шаблона специфичных значений (custom values) helm-chart для компонента `nodus-agent`

Описание параметров команды приведено в таблице 6.

Таблица 6

Параметр	Описание
`-o, --output=<"файл">`	Путь к сохраняемому файлу шаблона. Если параметр не указан, то содержимое шаблона выводится в консоль
`-h, --help`	Вывод справки по команде

Параметры, перечисленные в файлах шаблонов helm-chart, описаны в документации Kubernetes.

Примеры:

1. Создание шаблона конфигурационного файла кластера с тремя управляющими и девятью рабочими узлами и сохранение его в файл `configs/cluster_config.yml`:

```
nodus template cluster -m=3 -w=9 -o="configs/cluster_config.yml"
```

2. Создание шаблона специфичных значений helm-chart для компонента `nodus-agent` и его сохранение в файл `configs/nodus-agent.yml`

```
nodus template helm nodus-agent -o="configs/nodus-agent.yml"
```

5.1.2. Регистрация конфигурации кластера в Боцман

Для развертывания кластера на основе созданного конфигурационного файла конфигурация данного кластера должна быть зарегистрирована в Боцман . Для регистрации конфигурации кластера используется команда:

```
nodus cluster add
```

Синтаксис команды:

```
nodus cluster add -c <путь> <имя_кластера> [параметры]
```

Описание параметров команды приведено в таблице 7.

Т а б л и ц а 7

Параметр	Описание
`-c, --config <путь>`	Путь к конфигурационному файлу кластера. Обязательный параметр. Если путь содержит пробелы, то он должен быть заключен в кавычки
`-b, --bundle <версия>`	Использовать указанную версию добавленного в Боцман архива с установочными файлами для развертывания кластера. Если пара- метр не указан, то используется архив самой последней версии
`-f, --force`	Перезаписать зарегистрированную ранее конфигурацию кластера с таким же именем
`-h, --help`	Вывести справку по команде

Имя кластера не должно содержать пробелов. После регистрации конфигурации оно используется в качестве идентификатора кластера при любых операциях с ним (развертывание, удаление и т.д.), выполняемых с той машины, на которой была зарегистрирована конфигурация.

При регистрации конфигурации кластера в Боцман для устанавливаемых компонентов кла- стера создаются необходимые конфигурационные файлы в каталоге:

```
/home/<имя_пользователя>/.config/nodus/clusters/<имя_кластера>
```

Для вывода списка зарегистрированных в Боцман конфигураций кластеров используется команда:

```
nodus cluster ls
```

Для удаления из Боцман зарегистрированной конфигурации кластера используется команда:

```
nodus cluster rm <имя_кластера>
```

ВНИМАНИЕ! Удаление из Боцман зарегистрированной конфигурации кластера не затраги- вает развернутый по этой конфигурации кластер, но убирает возможность управлять им с помощью Боцман .

Удаленную из Боцман конфигурацию кластера можно повторно зарегистрировать с тем же или другим неиспользованным именем.

Примеры:

1. Регистрация конфигурации кластера под именем `test_cluster`:

```
nodus cluster add -c "/tmp/cluster config.yml" test_cluster
```

2. Перезапись зарегистрированной конфигурации кластера `work` с указанием версии установочного архива (см. 4.2):

```
nodus cluster add -c /tmp/cluster_conf_2.yml work -f -b v1.1.0
```

5.2. Служба nodusd

5.2.1. Установка и включение службы nodusd

Перед установкой компонентов кластера на его узлах должна быть установлена и включена служба `nodusd`. Данная служба создает TLS-сертификаты для связи между узлами и начальные конфигурационные файлы на узлах для последующей установки компонентов. Для установки службы `nodusd` используется команда:

```
nodus daemon install
```

Синтаксис команды:

```
nodus daemon install <имя_кластера> [параметры]
```

Описание параметров команды приведено в таблице 8.

Таблица 8

Параметр	Описание
`-j, --job <число>`	Количество параллельных процессов установки службы. Значение по умолчанию `4`
`-e, --enable`	Включить службу `nodusd` на узлах кластера после ее установки
`-h, --help`	Вывести справку по команде

Пример

Установка службы nodusd на узлы кластера test с последующим включением и 6 одновременными процессами установки:

```
nodus daemon install test -j 6 -e
```

Если команда nodus daemon install <имя_кластера> выполнялась без параметра -e, то после успешной установки службы nodusd на узлы кластера она должна быть включена на данных узлах.

Для включения службы nodusd используется команда:

```
nodus daemon enable
```

Синтаксис команды:

```
nodus daemon enable <имя_кластера> [параметры]
```

Описание параметров команды приведено в таблице 9.

Т а б л и ц а 9

Параметр	Описание
-j, --job <число>	Количество параллельных процессов включения службы. Значение по умолчанию 4
-h, --help	Вывести справку по команде

5.2.2. Отключение и удаление службы nodusd

Если служба nodusd отключена или удалена с узлов кластера, то управление кластером с помощью Боцман невозможно. При этом работоспособность развернутого кластера не затрагивается.

Для отключения службы nodusd на узлах кластера используется команда:

```
nodus daemon disable
```

Синтаксис команды:

```
nodus daemon disable <имя_кластера> [параметры]
```

Параметры данной команды аналогичны параметрам команды `nodus daemon enable <имя_кластера>` (см. таблицу 9).

5.2.3. Удаление службы nodusd

Для полного удаления службы `nodusd` с узлов кластера (например, после его очистки (см. 5.3.2)) используется команда:

```
nodus daemon destroy
```

Синтаксис команды:

```
nodus daemon destroy <имя_кластера> [параметры]
```

Параметры данной команды аналогичны параметрам команды `nodus daemon enable <имя_кластера>` (см. таблицу 9).

5.3. Развертывание и очистка кластера

5.3.1. Установка компонентов кластера

Установка и настройка всех необходимых компонентов кластера в соответствии с параметрами зарегистрированного конфигурационного файла производится с помощью команды:

```
nodus cluster install
```

Синтаксис команды:

```
nodus cluster install <имя_кластера> [параметры]
```

Имя кластера должно соответствовать имени конфигурации кластера, зарегистрированной в Боцман (см. 5.1.2).

Описание параметров команды приведено в таблице 10.

Т а б л и ц а 10

Параметр	Описание
`-j, --job <число>`	Количество параллельных процессов установки компонентов кластера. Значение по умолчанию `4`

Окончание таблицы 10

Параметр	Описание
`--kubeconfig-overwrite=[true, false]`	Заменить на узлах конфигурационные файлы Kubernetes с настройками авторизации (`/home/<имя_пользователя>/.kube/config`), записав в них параметры Боцман . Значение по умолчанию `true`
`-k, --kubeconfigs=[true, false]`	После успешной инициализации кластера скопировать конфигурационные файлы Kubernetes с управляющего узла на машину, с которой выполняется развертывание кластера. Значение по умолчанию `true`
`-l, --log`	Включить регистрацию событий с помощью службы `nodusd` (см. 7.3.1) в процессе развертывания кластера
`-h, --help`	Вывод справки по команде

П р и м е р

Развертывание кластера `work` с запуском восьми параллельных процессов установки (одновременная установка на 8 машинах) и включенной регистрацией событий:

```
nodus cluster install work -l -j 8
```

ВНИМАНИЕ! Если с момента установки на узлы кластера службы `nodusd` прошло более 48 часов, то узлы кластера будут недоступны из-за истечения срока действия созданных службой TLS-сертификатов. В этом случае следует повторно установить службу `nodusd` на узлы кластера (см. 5.2.1).

Последующее управление ресурсами кластера и развертывание в кластере приложений осуществляется с помощью клиента Kubernetes — `kubectl` (устанавливается отдельно). На машине, с которой выполнялось развертывание кластера, Боцман создает в процессе развертывания конфигурационный файл Kubernetes:

```
/home/<имя_пользователя>/.config/nodus/clusters/<имя_кластера>/kubeconfigs/ad-
   min.\ conf
```

с параметрами, необходимыми для управления кластером.

Для использования на этой машине клиента `kubectl` после успешного развертывания кластера необходимо выполнить одно из следующих действий:

- скопировать конфигурационный файл `kubectl`, сформированный Боцман , в каталог конфигурационных файлов Kubernetes и разрешить текущему пользователю доступ к этому файлу, выполнив команды:

```
mkdir -p $HOME/.kube
sudo cp -i $HOME/.config/nodus/clusters/<имя_кластера>/kubeconfigs/admin.\
   conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

- указать каталог расположения конфигурационного файла `kubectl` в качестве значения переменной среды `KUBECONFIG`, выполнив команду:

```
echo "export KUBECONFIG="$HOME/.config/nodus/clusters/<имя_кластера>/\ kubecon-
   figs/admin.conf"" >> ~/.bashrc && source ~/.bashrc
```

Для удобства вышеуказанные команды после окончания установки компонентов кластера выводятся в консоль.

Для управления кластером с помощью Боцман или `kubectl` с другой машины см. 6.3.

5.3.2. Очистка кластера

При очистке кластера с его узлов удаляются все установленные компоненты, и отменяются выполненные на них настройки. Служба `nodusd` при очистке кластера с узлов не удаляется. Для удаления службы `nodusd` см. 5.2.3.

Очистка кластера не удаляет из Боцман зарегистрированную конфигурацию кластера. Для удаления зарегистрированной конфигурации см. 5.1.2.

Для очистки кластера используется команда:

```
nodus cluster clean
```

Синтаксис команды:

```
nodus cluster clean <имя_кластера> [параметры]
```

Параметры команды приведены в таблице 11

Т а б л и ц а 11

Параметр	Описание
`-j, --job <число>`	Количество параллельных процессов очистки кластера. Значение по умолчанию `4`
`-l, --log`	Включить регистрацию событий с помощью службы `nodusd` (см. 7.3.1) в процессе очистки кластера
`-h, --help`	Вывести справку по команде

6. УПРАВЛЕНИЕ КЛАСТЕРОМ

6.1. Добавление узлов в развернутый кластер

6.1.1. Конфигурационный файл добавления узлов

Добавление узлов в развернутый кластер возможно, если его управляющие узлы доступны по сети, и его конфигурация зарегистрирована в Боцман (см. 5.1.2).

Для добавления узлов необходимо создать конфигурационный файл на языке разметки YAML. Структура данного файла идентична структуре конфигурационного файла кластера (см. Приложение А). Конфигурационный файл добавления узлов должен содержать раздел `inventory` со следующими подразделами:

- `nodes` — список добавляемых в кластер узлов с указанием их имен, адресов и ролей в кластере;
- `settings` — настройки SSH-подключения к добавляемым узлам.

Параметры, указанные в данном конфигурационном файле, будут применены только к перечисленным в файле узлам. Параметры кластера и остальных узлов изменены не будут.

Возможно создать шаблон конфигурационного файла для его дальнейшего редактирования (см. 5.1.1).

Пример

Конфигурационный файл для добавления в кластер одного управляющего и двух рабочих узлов.

```
inventory:
  settings:
    ssh:
      pass: password
      port: 22
      user: username
  nodes:
    masters:
    - host:
        hostname: master01
        ip: 10.0.0.1
    workers:
    - host:
        hostname: worker01
        ip: 10.0.0.11
    - host:
        hostname: worker02
        ip: 10.0.0.12
```

6.1.2. Добавление узлов

Для добавления узлов в кластер используется команда:

```
nodus utility join
```

Синтаксис команды:

```
nodus utility join -c <путь> <имя_кластера> [параметры]
```

Описание параметров команды приведено в таблице 12.

Т а б л и ц а 12

Параметр	Описание
`-c, --config <путь>`	Путь к конфигурационному файлу добавления узлов. Обязательный параметр. Если путь содержит пробелы, то он должен быть заключен в кавычки
`-j, --job <число>`	Количество параллельных процессов добавления узлов. Значение по умолчанию `4`
`-l, --log`	Включить регистрацию событий с помощью службы `nodusd` (см. 7.3.1) в процессе добавления узлов
`-h, --help`	Вывести справку по команде

6.2. Удаление узлов из развернутого кластера

Для удаления узлов из кластера используется команда:

```
nodus utility remove
```

или ее псевдоним:

```
nodus utility rm
```

Синтаксис команды:

```
nodus utility remove --ip <IP-адрес_узла> <имя_кластера> [параметры]
```

Описание параметров команды приведено в таблице 13.

Таблица 13

Параметр	Описание
`--ip <IP-адрес>`	IP-адрес удаляемого из кластера узла. Если нужно удалить несколько узлов, то параметр и его значение указываются несколько раз
`-j, --job <число>`	Количество параллельных процессов удаления узлов. Значение по умолчанию `4`
`-l, --log`	Включить регистрацию событий с помощью службы `nodusd` (см. 7.3.1) в процессе удаления узлов
`-h, --help`	Вывести справку по команде

Пример

Удаление из кластера «Production» двух узлов с включенной регистрацией событий:

```
nodus utility remove --ip 10.0.0.5 --ip 10.0.0.12 Production -l
```

ВНИМАНИЕ! Удаление из кластера первого управляющего узла невозможно, так как через него выполняются все операции по добавлению или удалению узлов.

6.3. Выгрузка конфигурационных файлов из кластера

Боцман позволяет выгрузить конфигурационные файлы развернутого кластера на любую машину. Это позволяет управлять кластером с этой машины с помощью Боцман и клиента `kubectl`. Используемая машина должна иметь доступ по сети к узлам кластера.

Перед выгрузкой конфигурационных файлов на используемой машине необходимо выполнить следующие действия:

- установить на машину Боцман (см. 4.1);
- добавить в Боцман установочный архив (см. 4.2.1);
- скопировать на машину конфигурационный файл развернутого кластера;
- зарегистрировать конфигурационный файл кластера в Боцман (см. 5.1.2). Имя кластера используется только на данной машине и может не совпадать с именем, указанным при развертывании кластера.

Для выгрузки конфигурационных файлов используется команда:

```
nodus utility get-configs
```

Синтаксис команды:

```
nodus utility get-configs <имя_кластера>
```

Описание параметров команды приведено в таблице 14.

Т а б л и ц а 14

Параметр	Описание
`-j, --job <число>`	Количество параллельных процессов выгрузки конфигурационных файлов. Значение по умолчанию `4`
`-l, --log`	Включить регистрацию событий с помощью службы `nodusd` (см. 7.3.1) в процессе выгрузки конфигурационных файлов
`-h, --help`	Вывести справку по команде

После выполнения команды с машины возможно управлять кластером с помощью Боцман .

Для использования на этой машине клиента `kubectl` после выгрузки конфигурационных файлов необходимо выполнить одно из следующих действий:

- скопировать выгруженный конфигурационный файл `kubectl` в каталог конфигурационных файлов Kubernetes и разрешить текущему пользователю доступ к этому файлу, выполнив команды:

```
mkdir -p $HOME/.kube
sudo cp -i $HOME/.config/nodus/clusters/<имя_кластера>/kubeconfigs/admin.\
   conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

- указать каталог расположения конфигурационного файла `kubectl` в качестве значения переменной среды `KUBECONFIG`, выполнив команду:

```
echo "export KUBECONFIG="$HOME/.config/nodus/clusters/<имя_кластера>/\ kubecon-
   figs/admin.conf"" >> ~/.bashrc && source ~/.bashrc
```

6.4. Обновление учетных данных кластера

В развернутом кластере Боцман используются следующие учетные данные:

- TLS-сертификаты для связи между узлами;
- bootstrap-токен `kubeadm` для установки двусторонних доверительных отношений между добавляемыми в кластер узлами и управляющими узлами.

Обновление учетных данных кластера выполняется автоматически при установке службы `nodusd`, добавлении и удалении узлов из кластера.

Боцман также позволяет принудительно обновить учетные данные. Это может быть полезно при отладке кластера сторонними средствами или при ошибках доступа к узлам.

Для обновления учетных данных кластера используется команда:

```
nodus utility update-creds
```

или ее псевдоним:

```
nodus utility update
```

Синтаксис команды:

```
nodus utility update-creds <имя_кластера> [параметры]
```

Описание параметров команды приведено в таблице 15.

Т а б л и ц а 15

Параметр	Описание
`--certs`	Обновить TLS-сертификаты узлов кластера
`--token`	Обновить bootstrap-токен `kubeadm`
`-a, --all`	Обновить и TLS-сертификаты, и bootstrap-токен `kubeadm`
`-j, --job <число>`	Количество параллельных процессов обновления учетных данных. Значение по умолчанию `4`
`-l, --log`	Включить регистрацию событий с помощью службы `nodusd` (см. 7.3.1) в процессе обновления учетных данных
`-h, --help`	Вывести справку по команде

6.5. Внутренний репозиторий образов

6.5.1. Общая информация

Развертывание приложений в кластере осуществляется путем запуска контейнеров из образов приложений. Узлы, на которых запускаются контейнеры, выбираются на основе их доступности и имеющихся на них ресурсах. При отказе какого-либо узла кластера все запущенные на нем контейнеры автоматически перезапускаются на других свободных узлах. Для возможности загрузки образов приложений со всех узлов кластера необходимо создать внутренний репозиторий образов.

6.5.2. Создание и настройка репозитория

Внутренний репозиторий следует настраивать на отдельной машине, не входящей в кластер. Данная машина должна быть доступна по сети для всех узлов кластера.

6.5.2.1. Установка пакетов и создание сертификата

Перед настройкой репозитория на машине необходимо установить пакеты `docker.io` и `docker-registry`.

Аутентификация в репозитории производится с помощью сертификата и ключа. Для их формирования необходимо создать конфигурационный файл `openssl-san.cnf` следующего содержания:

```
[req]
default_bits = 2048 default_key-
file = registry.key
distinguished_name = req_distinguished_name req_exten-
sions = req_ext
x509_extensions = v3_req
prompt = no

[req_distinguished_name] country-
Name = <Страна> stateOrProvince-
Name = <Область> localityName =
<Город> organizationName =
<Организация> commonName =
<Имя_репозитория>

[req_ext]
subjectAltName = @alt_names

[v3_req]
subjectAltName = @alt_names

[alt_names]
IP.1 = <IP-адрес_репозитория>
```

Для создания сертификата и ключа следует из каталога расположения файла `openssl-san.cnf` выполнить следующую команду:

```
sudo openssl req -x509 -new -nodes -keyout /etc/docker/registry/registry.key -out /\
    etc/docker/registry/registry.crt -days 365 -config openssl-san.cnf
```

6.5.2.2. Настройка аутентификации

Для настройки аутентификации в репозитории с помощью созданного сертификата необходимо выполнить следующие действия:

1) остановить службу `docker-registry`, выполнив команду:

```
sudo systemctl stop docker-registry
```

2) отредактировать файл `/etc/docker/registry/config.yml` следующим образом:

а) закомментировать раздел `auth`;

б) добавить в раздел `http:` подраздел `tls:` следующего содержания:

```
tls:
certificate: /etc/docker/registry/registry.crt
key: /etc/docker/registry/registry.key
```

Пример отредактированного файла:

```
version: 0.1
log:
  fields:
    service: registry stor-
age:
  cache:
    blobdescriptor: inmemory
  filesystem:
    rootdirectory: /var/lib/docker-registry
  delete:
    enabled: true
http:
  addr: :5000
  headers:
    X-Content-Type-Options: [nosniff]
  tls:
    certificate: /etc/docker/registry/registry.crt
    key: /etc/docker/registry/registry.key
#auth:
# htpasswd:
# realm: basic-realm
# path: /etc/docker/registry
health:
  storagedriver:
    enabled: true
    interval: 10s
    threshold: 3
```

3) изменить владельца созданного файла ключа, выполнив команду:

```
sudo chown docker-registry:docker-registry /etc/docker/registry/registry.key
```

4) запустить службу `docker-registry`, выполнив команду:

```
sudo systemctl start docker-registry
```

5) добавить созданный сертификат в Docker, выполнив следующие команды:

```
sudo mkdir -p /etc/docker/certs.d/<IP-адрес_репозитория>\:5000
sudo cp /etc/docker/registry/registry.crt /etc/docker/certs.d/<IP-\
   адрес_репозитория>\:5000/ca.crt
```

6) перезапустить службу `docker`, выполнив команду:

```
sudo systemctl restart docker
```

Для завершения настройки необходимо скопировать созданный сертификат `registry.crt` на все узлы кластера в каталог `/usr/local/share/ca-certificates` и выполнить на каждом узле следующие команды:

```
sudo update-ca-certificates
sudo systemctl restart crio
```

6.5.3. Просмотр и удаление образов

Загруженные в репозиторий образы хранятся в распакованном виде в каталоге:

```
/var/lib/docker-registry/docker/registry/v2/repositories
```

Имена подкаталогов соответствуют именам образов.

Для удаления образа необходимо выполнить следующее:

1) удалить соответствующий подкаталог;
2) удалить связанные с образом вспомогательные файлы, выполнив команду:

```
sudo docker-registry garbage-collect -m /etc/docker/registry/config.yml
```

7. ФУНКЦИИ БЕЗОПАСНОСТИ БОЦМАН

7.1. Проверка образов на уязвимости

В Боцман реализовано сканирование образов на наличие уязвимостей. Информация об уязвимостях содержится в базе данных OpenScap, устанавливаемой Боцман на узлы кластера в процессе его развертывания. Проверка на уязвимости производится модифицированной службой запуска контейнеров `crio`, входящей в состав Боцман .

Сканирование на наличие уязвимостей может выполняться каждый раз при скачивании образа, при создании контейнера и его запуске, а также периодически с заданной частотой.

Если наличие уязвимостей в образах разрешено, и включена отправка событий безопасности, то при каждом обнаружении уязвимостей в образах и/или запущенных из них контейнерах будут создаваться события в журнале Боцман (см. 7.3).

Если наличие уязвимостей запрещено, то образы, содержащие уязвимости, не будут скачиваться, а уже скачанные будут удаляться в процессе периодического сканирования.

ВНИМАНИЕ! Если в работающем кластере включить сканирование на уязвимости с запретом их наличия, то возможен отказ запуска приложений и деградация кластера.

Параметры сканирования могут быть указаны в конфигурационном файле кластера (см. Приложение А) на этапе развертывания кластера.

Если данные параметры нужно изменить после развертывания кластера, то они должны быть изменены вручную путем редактирования на узлах кластера конфигурационного файла службы `crio`, расположенного в `/etc/crio/crio.conf`. Для применения параметров после изменения конфигурационного файла необходимо перезапустить на данных узлах службу `crio`, выполнив команду:

```
sudo systemctl restart crio
```

Параметры конфигурационного файла `/etc/crio/crio.conf`, относящиеся к Боцман , и их описание приведены в таблице 16.

Т а б л и ц а 16

Параметр в `crio.conf`	Описание
`nodus_agent_endpoint`	IP-адрес и порт узла с работающей службой `nodus-agent`, куда отправляются события безопасно- сти

Окончание таблицы 16

Параметр в `crio.conf`	Описание
`nodus_agent_tls_ca_cert`	Путь к CA-сертификату службы `nodus-agent`. Если значение не задано, то устанавливается значение `/etc/ssl/certs/nodus-agent-ca.crt`. Применяется только вместе с параметром `nodus_agent_use_tls` с установленным значением `true`
`nodus_agent_tls_client_cert`	Путь к клиентскому сертификату службы `nodus-agent`. Если значение не задано, то устанавливается значение `/etc/ssl/certs/nodus-agent-client.crt`. Применяется только вместе с параметром `nodus_agent_use_tls` с установленным значением `true`
`nodus_agent_tls_client_key`	Путь к ключу клиентского сертификата службы `nodus-agent`. Если значение не задано, то устанавливается значение `/etc/ssl/certs/nodus-agent-client.key`. Применяется только вместе с параметром `nodus_agent_use_tls` с установленным значением `true`
`nodus_agent_use_tls`	Включение/выключение защищенного соединения между `crio` и службой `nodus-agent`
`nodus_allow_vulnerables`	Разрешение на наличие уязвимостей в образах. При установленном значении `false` наличие уязвимостей запрещено. Образы с уязвимостями не будут скачаны, если уязвимости нашлись в момент загрузки, или будут удалены, если уязвимости нашлись при периодическом сканировании
`nodus_events_enabled`	Включение отправки событий безопасности. Для приема событий безопасности требуется настроенный сервер со службой `nodus-agent`
`nodus_scanner_database`	Путь к XML-файлу с базой уязвимостей. Если значение не задано, то устанавливается значение `/usr/local/share/oval/db.xml`
`nodus_scanner_enabled`	Включение сканирования на уязвимости при скачивании образа, создании контейнера и запуске контейнера
`nodus_scanner_reports`	Путь для сохранения отчетов сканирования OpenScap. Если значение не задано, то устанавливается значение `/usr/local/share/crio/oscap-reports`
`nodus_scanner_service_enabled`	Включение/выключение периодического сканирования. Частота сканирования задается параметром `nodus_scanner_service_period`
`nodus_scanner_service_period`	Частота сканирования образов и контейнеров на уязвимости. Доступные значения: `hourly` (раз в час), `weekly` (раз в неделю), `monthly` (раз в месяц). Значение по умолчанию `monthly`

7.2. Ролевая модель управления доступом

7.2.1. Доступ к кластеру на основе назначенной роли

Доступ к развернутому кластеру Боцман и его ресурсам осуществляется с помощью ролевой модели управления доступом (RBAC — Role-Based Access Control).

В ролевую модель управления доступом входят следующие объекты:

- Роль (Role) — описывает набор прав, которые могут быть применены к ресурсам в пределах определенного пространства имен;
- Кластерная роль (ClusterRole) — описывает набор прав, которые могут быть применены ко всем ресурсам в кластере;
- Привязка роли (RoleBinding) — связывает роль (или кластерную роль) с конкретным пользователем или группой пользователей в определенном пространстве имен;
- Привязка кластерной роли (ClusterRoleBinding) — связывает кластерную роль с пользователем или группой пользователей на уровне всего кластера.

Для получения доступа к кластеру и выполнения действий в отношении различных ресурсов кластера субъекту (пользователю, группе или сервисной учетной записи) должна быть назначена определенная роль или ролевая связь.

Создание ролей и управление ими осуществляется с помощью инструмента `kubectl` (см. документацию Kubernetes).

В определении роли указываются пространства имен, в которых может осуществляться доступ к ресурсам, сами ресурсы и действия, которые могут над ними выполняться.

Пример

Определение роли, дающей права на чтение ресурса «Поды». Комментарии в примере начинаются с символа `#`.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default # Пространство имен, в котором задана роль
  name: pod-reader # Имя роли
rules:
- apiGroups: [""]
  resources: ["pods"] # Ресурс, к которому предоставляется доступ
  verbs: ["get", "watch", "list"] # Действия с ресурсом, выполнение \
     которых разрешено ролью
```

Перед применением роли пользователь должен пройти аутентификацию в Боцман с помощью сертификата X.509 или токена `kubeadm`. Далее пользователю необходимо назначить одну или несколько ролей, из имеющихся в кластере.

Назначение пользователю роли осуществляется с помощью файла привязки в формате YAML. В данном файле указываются следующие данные:

- пользователь, которому назначается роль;
- назначаемая роль;
- пространство имен, в котором действует данная роль, если оно отличается от указанного в определении роли. Для кластерных ролей пространство имен не указывается.

Пример

Содержимое файла привязки кластерной роли `container-image-developer` к пользователю `jane`.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: jane-binding
subjects:
- kind: User
  name: jane
  apiGroup: rbac.authorization.k8s.io rol-
eRef:
  kind: ClusterRole
  name: container-image-developer
  apiGroup: rbac.authorization.k8s.io
```

Для привязки роли следует ввести команду:

```
kubectl apply -f <файл_привязки>
```

Для отображения прав определенной роли следует ввести команду:

```
kubectl describe role <имя_роли>
```

Для отображения прав определенной кластерной роли следует ввести команду:

```
kubectl describe clusterrole <имя_роли>
```

Для просмотра прав текущего пользователя в рамках назначенной ему роли следует ввести команду:

```
kubectl auth can-i --list
```

Встроенные в Kubernetes роли с их описанием приведены в документации Kubernetes.

7.2.2. Роли Боцман

В дополнение к имеющимся встроенным ролям Kubernetes в Боцман реализованы три роли:

- разработчик образов контейнеров (`container-image-developer`);
- администратор информационной (автоматизированной) системы (`information-system-security-admin`);
- администратор безопасности средства контейнеризации (`containerization-tool-security-admin`).

При инициализации кластера Боцман автоматически создает пользователей с привязкой к своим ролям, формирует и подписывает сертификаты безопасности, а также создает конфигурационные файлы пользователей для работы в рамках вышеуказанных ролей.

Права ролей Боцман приведены в таблице 17. Доступные ролям ресурсы и действия указаны в той форме, в которой они указываются в конфигурационных файлах Kubernetes.

Т а б л и ц а 17

Права в кластере	Права ролей Боцман и доступные им действия		
	Разработчик образов контейнеров	Администратор информационной (автоматизированной) системы	Администратор безопасности средства контейнеризации
Создание и просмотр атрибутов субъекта после аутентификации в кластере	+	+	+
Доступ к подам (ресурс `pods`)	Чтение/Запись	Чтение/Запись	Чтение/Запись
Присоединение контейнера к поду (ресурсы `pods/exec`, `pods/attach`)	–	`"create"`, `"get"`	`"create"`, `"get"`
Назначение пода определенному узлу или миграция пода с узла (ресурсы `pods/binding`, `pods/eviction`)	–	`"create"`, `"get"`	`"create"`, `"get"`, `"delete"`

Окончание таблицы 17

Права в кластере	Права ролей Боцман и доступные им действия		
	Разработчи к образов контейнеро в	Администратор информационной (автоматизированной) системы	Администратор безопасности средства контейнеризации
Доступ к журналам и статусу пода и контейнеров в нем (ресурсы `pods/log`, `pods/status`)	–	–	Чтение
Доступ к событиям кластера (ресурс `events`)	–	–	`"get"`, `"list"`, `"watch"`
Доступ к узлам (ресурс `nodes`)	–	Чтение	Чтение
Доступ к пространству имен (ресурс `namespaces`)	–	Чтение	Чтение
Право на шаблоны подов (ресурс `podtemplates`)	–	Чтение/Запись	Чтение/Запись
Право на репликацию подов в кластере (ресурсы `deployments`, `replicasets`, `daemonsets`)	–	Чтение/Запись	Чтение/Запись
Право на роли и назначение субъектам ролей в определенных пространствах имен (ресурсы `roles`, `rolebindings`)	–	–	Чтение/Запись
Право на ресурс `serviceaccounts`	–	–	`"get"`, `"list"`, `"create"`, `"delete"`
Право на ресурс `serviceaccounts/token`	–	–	`"get"`, `"list"`, `"create"`, `"delete"`

7.2.3. Назначение ролей Боцман пользователям

7.2.3.1. Передача конфигурационного файла

Назначение роли пользователю может осуществляться путем передачи ему конфигурационного файла данной роли. Конфигурационный файл роли содержит имя пользователя, наименование назначаемой роли и необходимые учетные данные для аутентификации пользователя с данной ролью в кластере.

Конфигурационные файлы для всех ролей создаются автоматически на этапе развертывания кластера. Они хранятся на машине, с которой осуществлялось развертывание, в каталоге:

```
/home/<имя_пользователя>/.config/nodus/clusters/<имя_кластера>/kubeconfigs/
```

Имя файла соответствует роли, предоставляемой с его помощью.

Для назначения пользователю роли необходимо:

1) скопировать конфигурационный файл в домашний каталог пользователя под именем:

```
/home/<имя_пользователя>/.kube/config
```

2) изменить владельца файла и права доступа к нему, выполнив на машине пользователя команды:

```
sudo chown <имя_пользователя> $HOME/.kube/config
sudo chmod u+rw $HOME/.kube/config
```

7.2.3.2. Привязка роли к пользователю

Назначение роли также возможно выполнить путем создания пользователя и привязки роли к нему.

Для создания привязки роли необходимо на управляющем узле выполнить следующее:

1) создать закрытый ключ пользователя, выполнив команду:

```
openssl genrsa -out <имя_пользователя>.key 2048
```

2) создать файл запроса для выпуска сертификата пользователя, выполнив команду:

```
openssl req -new -key <имя_пользователя>.key -out <имя_пользователя>.csr -\
    subj "/CN=<имя_пользователя>"
```

3) создать на основе файла запроса сертификат и подписать его с помощью корневого сертификата кластера, выполнив команду:

```
sudo openssl x509 -req -in <имя_пользователя>.csr -CA /etc/kubernetes/pki/ca.\
    crt -CAkey /etc/kubernetes/pki/ca.key -CAcreateserial -out <\
    имя_пользователя>.crt -days 365
```

4) создать файл привязки `rolebinding.yaml` следующего содержания для назначения роли пользователю (в примере выполняется привязка пользователя к кластерной роли `container-image-developer`):

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
```

```
metadata:
  name: <имя_пользователя>-binding
  namespace: <пространство_имен>
subjects:
  - kind: User
    name: <имя_пользователя>
    apiGroup: rbac.authorization.k8s.io rol-
eRef:
  kind: ClusterRole
  name: container-image-developer api-
  Group: rbac.authorization.k8s.io
```

Для отображения доступных пространств имен следует выполнить команду:

```
kubectl get namespaces
```

5) применить настройки, указанные в файле привязки роли, выполнив от имени пользователей `cluster-admin` или `containerization-tool-security-admin` следующую команду:

```
kubectl apply -f rolebinding.yaml
```

Для создания пользователя и назначения ему прав в соответствии с ролью необходимо на управляющем узле выполнить следующее:

1) задать имя и учетные данные кластера, выполнив команду:

```
kubectl config set-cluster <служебное_имя_кластера> --server=https://<\
   адрес_сервера> --certificate-authority=/etc/kubernetes/pki/ca.crt --embed-\
   certs=true --kubeconfig=config
```

Для отображения служебного имени кластера и адреса сервера следует выполнить следующие команды:

```
kubectl config get-clusters
kubectl get endpoints
```

2) задать имя и учетные данные пользователя, выполнив команду:

```
kubectl config set-credentials <имя_пользователя> --client-certificate=<\
   имя_пользователя>.crt --client-key=<имя_пользователя>.key --embed-certs=\
   true --kubeconfig=config
```

3) создать контекст — группу настроек, объединяющую в себе кластер, пространство имен и пользователя, выполнив команду:

```
kubectl config set-context <имя_пользователя>-context --cluster=<\
   служебное_имя_кластера> --namespace=<пространство_имен> --user=<\
   имя_пользователя> --kubeconfig=config
```

4) переключить пользователя на созданный контекст, выполнив команду:

```
kubectl config use-context <имя_пользователя>-context --kubeconfig=config
```

5) скопировать сформированный конфигурационный файл `config` на машину пользователя в каталог:

```
/home/<имя_пользователя>/.kube
```

6) изменить владельца файла и права доступа к нему, выполнив на машине пользователя команды:

```
sudo chown <имя_пользователя> $HOME/.kube/config
sudo chmod u+rw $HOME/.kube/config
```

7) установить на машине пользователя пакет `kubernetes-client`.

Для проверки возможности создания пользователем подов (действие, разрешенное используемой в примере ролью) следует на машине пользователя выполнить команду:

```
kubectl auth can-i create pods
```

В случае успешной привязки команда выведет:

```
yes
```

7.2.4. Пример работы назначенных ролей

Назначенная пользователю роль определяет доступные ему ресурсы кластера и действия, которые он может выполнять с данными ресурсами (см. таблицу 17).

П р и м е р ы:

1. просмотр событий кластера пользователем с ролью «Разработчик образов контейнеров» (права на ресурс `events` отсутствуют):

```
kubectl get events
```

вывод команды:

```
Error from server (Forbidden): events is forbidden: User "container-image-\ de-
    veloper" cannot list resource "events" in API group "" in the namespace \ "de-
    fault"
```

2. просмотр событий кластера пользователем с ролью «Администратор безопасности средства контейнеризации» (есть права на ресурс `events`):

```
LAST SEEN TYPE  REASON      OBJECT   MESSAGE
. . .
39m   Normal  Starting     node/worker01 Starting kubelet.
```

```
39m   Normal  NodeHasSufficientMemory  node/worker01 Node worker01 status is \
   now: NodeHasSufficientMemory
39m   Normal  NodeHasNoDiskPressure  node/worker01 Node worker01 status is \
   now: NodeHasNoDiskPressure
39m   Normal  NodeHasSufficientPID  node/worker01 Node worker01 status is now\
   : NodeHasSufficientPID
. . .
```

3. выполнение команды внутри запущенного контейнера пользователем с ролью «Разработчик образов контейнеров» (права на ресурс `pods/exec` отсутствуют):

```
kubectl exec nginx-56fcf95486-fvrws -- ls
```

вывод команды:

```
Error from server (Forbidden): pods "nginx-56fcf95486-fvrws" is forbidden: \
   User "container-image-developer" cannot create resource "pods/exec" in API\
    group "" in the namespace "default"
```

4. выполнение команды внутри запущенного контейнера пользователем с ролью «Администратор информационной системы» (есть права на ресурс `pods/exec`):

```
kubectl exec nginx-56fcf95486-fvrws -- ls
```

вывод команды:

```
bin
boot
dev
. . .
```

7.3. Регистрация событий

7.3.1. Регистрация событий при операциях с кластером

При развертывании, очистке и изменении состава кластера с включенной регистрацией событий служба `nodusd` регистрирует все события и направляет их на машину, с которой выполняются операции с кластером.

Файлы журналов с узлов кластера сохраняются в каталоге:

```
/home/<имя_пользователя>/.config/nodus/clusters/<имя_кластера>/logs/remote
```

Файлы журнала машины, с которой выполняется развертывание или очистка кластера, сохраняются в каталоге:

```
/home/<имя_пользователя>/.config/nodus/clusters/<имя_кластера>/logs/local
```

7.3.2. Регистрация событий безопасности

Регистрация событий безопасности Боцман внутри кластера производится службой `crio`. Регистрируемые события безопасности делятся на следующие группы:

- события безопасности, связанные с образами контейнеров;
- события безопасности, связанные с контейнерами;
- события безопасности, связанные с подами.

События получения списка образов, контейнеров или подов не регистрируются, так как службы кластера постоянно опрашивают их состояние 1-2 раза в секунду, что приводит к переполнению журнала.

Структура события включает в себя следующие элементы:

- `name` — название события;
- `level` — уровень события;
- `payload` — содержание события;
- `initiator` — информация об инициаторе события;
- `error` — ошибка, полученная во время события.

Сбор событий безопасности с узлов кластера осуществляется службой `nodus-agent`, устанавливаемой автоматически при развертывании кластера и запущенной на первом управляющем узле кластера. Собранные события служба `nodus-agent` направляет в системную утилиту `syslog-ng`, которая сохраняет их в файл журнала `/parsec/log/astra/nodus`, также расположенный на первом управляющем узле.

Регистрация событий безопасности может быть включена в конфигурационном файле кластера (см. Приложение А) на этапе развертывания кластера.

Для включения регистрации событий безопасности на узлах кластера после его развертывания необходимо выполнить на данных узлах следующее:

1) в конфигурационном файле `/etc/crio/crio.conf` раскомментировать следующие параметры с установкой для них значения `true`:

```
nodus_events_enabled nodus_scan-
ner_enabled
```

```
nodus_scanner_service_enabled
nodus_agent_use_tls
```

2) для предотвращения деградации кластера вследствие удаления запущенных образов с обнаруженными уязвимостями в конфигурационном файле `/etc/crio/crio.conf` раскомментировать следующий параметр с установкой для него значения `true`:

```
nodus_allow_vulnerables
```

3) перезапустить службу `crio`, выполнив команду:

```
sudo systemctl restart crio
```

7.3.3. Виды событий безопасности

События безопасности, создаваемые службой `crio`, перечислены в таблице 18

Т а б л и ц а 18

Подгруппа события	Имя события в журнале	Уровень события	Описание события
События безопасности образов			
Скачивание образа	image.pulling	Информационное	Попытка загрузки образа. Регистрируется всегда
	image.pulled	Информационное	Успешная загрузка образа. Не регистрируется при ошибках загрузки, или если образ существует в локальном хранилище
Удаление образа	image.removing	Информационное	Попытка удаления образа. Регистрируется всегда
	image.removed	Информационное	Успешное удаление образа. Не регистрируется при ошибках удаления, например, если образ не существует

Подгруппа события	Имя события в журнале	Уровень события	Описание события
Уязвимости в образе	image.vulnerabilities-found	Экстренное	Обнаружение уязвимостей в образе. Регистрируется при включенном сканировании на уязвимости
	image.removed- vulnerable	Экстренное	Удаление образа с уязвимостями. Регистрируется при включенном сканировании на уязвимости и запрете использования уязвимых образов
События безопасности контейнеров			
Подключение контейнера	container.attach	Информационное	Присоединение ввода/вывода к запущенному контейнеру. Не регистрируется, если контейнер отсутствует или не запущен
Сохранение состояния контейнера	container.checkpointing	Информационное	Попытка сохранения текущего состояния контейнера. Регистрируется всегда
	container.checkpointed	Информационное	Успешное сохранение текущего состояния контейнера. Не регистрируется в случае ошибки при выполнении операции
Создание контейнера	container.creating	Информационное	Попытка создания нового контейнера. Регистрируется всегда
	container.created	Информационное	Успешное создание нового контейнера. Не регистрируется в случае ошибки при выполнении операции
Выполнение команд в контейнере	container.exec	Информационное	Попытка выполнения команды внутри контейнера. Регистрируется всегда
Синхронизация выполнения команд	container.exec-sync	Информационное	Попытка синхронного выполнения команды внутри контейнера. Регистрируется всегда
Проброс портов контейнера	container.port-forward	Информационное	Попытка проброса портов с хоста в контейнер. Регистрируется всегда
Удаление контейнера	container.removing	Информационное	Попытка удаления контейнера. Регистрируется всегда
	container.removed	Информационное	Успешное удаление контейнера. Не регистрируется в случае ошибки при выполнении операции
Пересоздание журнала контейнера	container.reopening-log	Информационное	Попытка пересоздания файла журнала контейнера. Регистрируется всегда
	container.reopened-log	Информационное	Успешное пересоздание файла журнала контейнера. Не регистрируется в случае ошибки при выполнении операции

Подгруппа события	Имя события в журнале	Уровень события	Описание события
Восстановление контрольной точки контейнера	container.restoring-checkpoint	Информационное	Попытка восстановления контрольной точки контейнера. Регистрируется всегда
	container.restored-checkpoint	Информационное	Успешное восстановление контрольной точки контейнера. Не регистрируется в случае ошибки при выполнении опера- ции
Запуск контейнера	container.starting	Информационное	Попытка запуска контейнера. Регистрируется всегда
	container.started	Информационное	Успешный запуск контейнера. Не регистрируется в случае ошибки при выполнении операции
Статистика контейнера	container.stats	Информационное	Попытка получения статистики состояния контейнера. Не регистрируется, если контейнер не существует
Остановка контейнера	container.stopping	Информационное	Попытка остановки контейнера. Регистрируется всегда
	container.stopped	Информационное	Успешная остановка контейнера. Не регистрируется в случае ошибки при выполнении операции
Изменение конфигурации контейнера	container.updating-resources	Информационное	Попытка изменения файла конфигурации контейнера. Регистрируется всегда
	container.updated-resources	Информационное	Успешное изменение конфигурации контейнера. Не регистрируется в случае ошибки при выполнении операции
Уязвимости в контейнере	container.reject- vulnerable	Экстренное	Отказ запуска контейнера из образа с уязвимостями. Регистрируется при включенном событийном сканировании на уязвимости и запрете использования уязвимых контейнеров
	container.removed-vulnerable	Экстренное	Удаление контейнера, при обнаружении уязвимостей в его образе. Регистрируется при включенном периодическом сканировании на уязвимости и запрете использования уязвимых контейнеров или образов
События безопасности подов			
Сохранение состояния пода	pod.checkpointing	Информационное	Попытка сохранения текущего состояния пода. Регистрируется всегда
	pod.checkpointed	Информационное	Успешное сохранение текущего состояния пода. Не регистрируется в случае ошибки при выполнении операции

Окончание таблицы 18

Подгруппа события	Имя события в журнале	Уровень события	Описание события
Удаление пода	pod.removing	Информационное	Попытка удаления пода. Регистрируется всегда
	pod.removed	Информационное	Успешное удаление пода. В полезную нагрузку события будут добавлены входящие в под запущенные контейнеры со сквозной нумерацией, начиная с «0». Не регистрируется в случае ошибки при выполнении операции
Запуск пода	pod.running	Информационное	Попытка создания и запуска пода. Регистрируется всегда
	pod.ran	Информационное	Успешное создание и запуск пода. Не регистрируется в случае ошибки при выполнении операции
Статистика пода	pod.stats	Информационное	Попытка получения статистики состояния пода. Не регистрируется, если контейнер не существует
Остановка пода	pod.stopping	Информационное	Попытка остановки пода. Регистрируется всегда
	pod.stopped	Информационное	Успешная остановка пода. В полезную нагрузку события будут добавлены входящие в под запущенные контейнеры со сквозной нумерацией, начиная с «0». Не регистрируется в случае ошибки при выполнении операции
	pod.stopped-already	Информационное	Попытка остановки уже остановленного пода. Регистрируется всегда

Пример

Событие безопасности, создаваемое при попытке загрузки образа с уязвимостями при включенном сканировании на уязвимости и запрете использования уязвимых образов:


```
{"PROGRAM":"nodus-agent","PRIORITY":"emerg","PID":"1","MSG":{"astra-nodus-
agent":{"unique_id":"1727875668.461:49","type_ru":"События Nodus-agent",
"type_en":"Nodus-agent events","payload":"[{\"name\":\"name\",\"value\":\
"registry.astralinux.ru/library/alse:1.7.1\"},{\"name\":\"vulnerabilities-
amount\",\"value\":\"26\"}]","name_ru":"Событие безопасности Nodus-agent",
"name_en":"Security event from Nodus-agent","message_id":"nodus-agent_ se-
curity_event","initiator":"[{\"name\":\"user.id\",\"value\":\"0\"},{\"name
\":\"user.name\",\"value\":\"root\"},{\"name\":\"machine.id\",\"value\":\
"ab83c9a290c449259d7e04a907dd05a3\"},{\"name\":\"timestamp.create-at.date-
time
\",\"value\":\"October 02 2024 16:27:48\"},{\"name\":\"timestamp.create-at.
timezone\",\"value\":\"UTC+03\"}]","action":"image.removed-vulnerable"}},
"ISODATE":"2024-10-02T13:27:48+00:00","HOST":"127.0.0.1","FACILITY":"dae-
mon"}
```

ПРИЛОЖЕНИЕ А

(обязательное)

КОНФИГУРАЦИОННЫЙ ФАЙЛ Боцман

А.1. СТРУКТУРА КОНФИГУРАЦИОННОГО ФАЙЛА

Конфигурационный файл для развертывания кластера создается на языке разметки YAML и состоит из следующих разделов:

1) `inventory` — настройки подключения узлов в кластере. Наличие данного раздела обязательно;

2) `kubeadm` — настройки для инициализации кластера и конфигурирования его основных компонентов с помощью инструмента `kubeadm`;

3) `crio` — настройки службы запуска контейнеров `crio`;

4) `settings` — дополнительные настройки кластера:

- расширенные настройки компонентов кластера, в том числе указываемые в отдельных файлах специфичных значений (custom values) helm-chart;
- настройки локального репозитория с образами;
- настройки регистрации событий, сканирования на уязвимости, компонентов кластера и пр.

ВНИМАНИЕ! Раздел `settings` должен обязательно присутствовать в конфигурационном файле и содержать следующий параметр:

```
localRegistry:
enable: true
```

A.2. РАЗДЕЛ INVENTORY

В разделе `inventory` указываются общие сетевые настройки узлов кластера и их роли. Комментарии в файле начинаются с символа `#`.

```
inventory:
  # Общие сетевые настройки для всех узлов
  settings:
    # DNS-сервера, которые будут прописаны на узлах
    dns:
      - 10.0.1.1
      - 10.0.1.2
    # Настройки SSH-подключения к узлам
    ssh:
      # Путь к файлу закрытого ключа SSH (ключ не должен быть защищен паролем) key-
      File: /путь/к/файлу/ключа/ssh
      # Пароль пользователя (при аутентификации без использования ключей)
      pass: password1
      # Порт, настроенный для подключения
      port: 22
      # Имя пользователя для подключения
      user: user1
  # Список входящих в кластер узлов с их ролями
  nodes:
    # Список мастер-узлов
    masters:
    - host:
        # Имя, присваиваемое узлу при развертывании кластера
        hostname: master01
        # реальный IP-адрес, присвоенный узлу
        ip: 10.0.0.1
      # Настройки SSH, переопределяющие общие настройки раздела settings
      ssh:
        pass: password2
        port: 2201
        user: user2
      # Настройки DNS, переопределяющие общие настройки раздела settings
      dns:
        - 10.0.2.1
        - 10.0.2.2
    - host:
        hostname: master02
        ip: 10.0.0.2
    # Список рабочих узлов work-
    ers:
    - host:
        hostname: worker01
        ip: 10.0.0.11
    - host:
        hostname: worker02
```

```
    ip: 10.0.0.12
  - host:
    hostname: worker03
    ip: 10.0.0.13
# Узел, с которого начинается инициализация кластера (первый управляющий узел). \
  Если раздел отсутствует, то выбирается первый узел из списка masters
firstMaster: host-
  name: master01 ip:
  10.0.0.1
```

A.3. РАЗДЕЛ KUBEADM

В разделе `kubeadm` возможно переопределить настройки конфигурационного файла пакета `kubeadm` в следующих подразделах:

- `InitConfiguration`;
- `ClusterConfiguration`;
- `KubeletConfiguration`;
- `KubeProxyConfiguration`.

Данный раздел заполняется в соответствии с официальной документацией Kubernetes.

ВНИМАНИЕ! Перед каждым подразделом конфигурации `kubeadm` необходимо указывать значение параметра `kind:` со строчной буквы и с двоеточием в конце. Строки с символами «`---`», отделяющие друг от друга разделы стандартного конфигурационного файла `kubeadm`, должны быть удалены.

Пример

```
kubeadm:
initConfiguration: # Добавлено имя подраздела из параметра kind: apiVer-
  sion: kubeadm.k8s.io/v1beta3
  kind: InitConfiguration nodeReg-
  istration:
    criSocket: unix:///var/run/crio/crio.sock
    imagePullPolicy: IfNotPresent
    name: master01
    taints:
    - effect: NoSchedule
      key: node-role.kubernetes.io/control-plane clusterConfigura-
tion: # Вместо символов "---" указано имя подраздела
  apiVersion: kubeadm.k8s.io/v1beta3
  imageRepository: localhost:5000
  kind: ClusterConfiguration
  kubernetesVersion: v1.29.2-nodus1.0.0
kubeProxyConfiguration: # Вместо символов "---" указано имя подраздела apiVer-
  sion: kubeproxy.config.k8s.io/v1alpha1
  kind: KubeProxyConfiguration
kubeletConfiguration: # Вместо символов "---" указано имя подраздела apiVer-
  sion: kubelet.config.k8s.io/v1beta1
  kind: KubeletConfiguration clus-
  terDNS:
  - 169.254.20.11
  clusterDomain: cluster.local
```

А.4. РАЗДЕЛ CRIO

В разделе `crio` возможно переопределить настройки конфигурационного файла службы `crio` в следующих подразделах:

- `root`;
- `api`;
- `runtime`;
- `image`;
- `metrics`.

Данный раздел заполняется в соответствии с официальной документацией Kubernetes.

Также в данном разделе возможно указать настройки безопасности контейнеров в подразделе `nodus`.

```
crio:
  root:
    root:
    runRoot:
  api:
    streamAddress:
    streamPort:
    streamEnableTLS:
    streamTLSCert:
    streamTLSKey:
    streamTLSCa:
  runtime:
    conmon:
    cgroupManager:
    defaultCapabilities:
    readOnly:
    logLevel:
  image:
    pauseImage:
    pauseImageAuthFile:
    insecureRegistries:
  metrics:
    enableMetrics:
    metricsPort:
  # подраздел настроек безопасности и регистрации событий, приведены значения \
     параметров по умолчанию
  nodus:
    # Включение/выключение сканера уязвимостей scanner-
    Enabled: false
    # Путь к базе данных с уязвимостями (oval db) scan-
    nerDatabase: /usr/local/share/oval/db.xml
    # Путь сохранения отчета о результатах сканирования на уязвимости
```

```
scannerReports: /usr/local/share/crio/oscap-reports
# Включение/выключение периодического сканирования
scannerServiceEnabled: false
# Частота периодического сканирования (раз в час (hourly), раз в неделю (weekly)\
  , раз в месяц (monthly)) scannerSer-
vicePeriod: monthly
# Разрешение/запрет наличия уязвимостей на узле al-
lowVulnerables: true
# Включение/выключение создания и отправки событий безопасности в nodus-agent ->\
   syslog-ng
eventsEnabled: false
# IP-адрес и порт узла с работающей службой nodus-agent, куда отправляются \
  события безопасности
agentEndpoint: localhost:30007
# Включение/выключение защищенного соединения между службами crio и nodus-agent \
  (сертификаты автоматически создаются Боцман)
agentUseTLS: false
# Путь к CA-сертификату службы nodus-agent agent-
TLSCACert: /etc/ssl/certs/nodus-agent-ca.crt
# Путь к клиентскому сертификату службы nodus-agent
agentTLSClientCert: /etc/ssl/certs/nodus-agent-client.crt
# Путь к ключу клиентского сертификата службы nodus-agent
agentTLSClientKey: /etc/ssl/certs/nodus-agent-client.key
```

A.5. РАЗДЕЛ SETTINGS

В разделе `settings` возможно переопределить следующие настройки установки и работы дополнительных компонентов кластера:

- запуск локального репозитория на узлах кластера;
- политика регистрации событий;
- настройки службы `node-local-dns` (поддержка кеша DNS на каждом узле для снижения сетевой нагрузки и ускорения преобразования доменных имен);
- настройки службы `nodus-agent` (сбор со всех узлов регистрируемых событий безопасности и их запись в единый журнал);
- настройки службы `cilium` (обеспечение безопасного сетевого взаимодействия узлов кластера друг с другом);
- настройки службы `nginx-proxy` (обеспечение доступа к запущенным в кластере приложениям снаружи кластера).

```
settings:
  # Настройки, связанные с запуском на узлах локального репозитория с образами localReg-
  istry:
    # Включить/выключить запуск локального репозитория на узлах
    enable: false
    # Настройка портов репозитория
    ports:
      # Порт репозитория на локальном хосте
      repository: 5000
      # Порт для приема отладочной информации на локальном хосте
      debug: 5001
  # Настройки политики регистрации событий kubernetes
  auditPolicy:
    # Максимальное количество дней для хранения старых файлов журнала log-
    MaxAge: 30
    # Определяет максимальное количество сохраняемых старых файлов журнала log-
    MaxBackup: 10
    # Определяет максимальный размер файла журнала в мегабайтах до его ротации log-
    MaxSize: 100
    # Путь к файлу журнала
    logPath: /var/log/kubernetes/audit/audit.log
    # Путь к файлу настройки политик регистрации событий
    policyFile: /etc/kubernetes/audit/policy.yaml
  # Порт, который будет назначен для API серверов и балансировщика нод (nginx-proxy)
  kubeApiServerBindPort: 6443
  # Время ожидания готовности кластера после его успешной инициализации clusterWaiting-
  Timeout: 300s
  # Настройки дополнительных компонентов кластера
  Боцман
  components:
    nodusAgent:
      # Настройка пользовательских значений в соответствии с документацией kubernetes
```

```
  customValues:
    # Путь к существующему файлу helm-chart с пользовательскими значениями. \
      Значения в файле имеют приоритет над перечисленными здесь значениями
    path:
    # Значения, которые будут изменены
    Боцман values:
      # Настройки образа
      image:
        # Путь к репозиторию с образом
        repository:
        # Метка образа (версия)
        tag: v0.0.44
      vector:
        image:
          repository:
          tag: 0.39.0-distroless-libc
      # Ресурсы, выделяемые на поды
      resources:
        requests:
          cpu: "100m"
          memory: "128Mi"
        limits:
          cpu: "100m"
          memory: "128Mi"
      # Настройки оповещений администратора при появлении уязвимости/угрозы в \
        кластере
      alerter:
        config:
          # Список почтовых адресов
          mails: ""
          # Стписок уровней угроз/уязвимостей, которые будут задействованы в \
            оповещениях
          severities: ""
          mailSubject: ""
        creds:
          # IP-адрес SMTP-сервера
          server:
          # порт SMTP-сервера
          port:
          # имя пользователя для подключения к SMTP серверу
          username:
          # пароль для подключения к SMTP серверу
          password:
nodeLocalDns:
  # Включить/выключить установку агента на узлах en-
  able: true
  # Настройка пользовательских значений в соответствии с документацией Node-Local\
    -DNS
  customValues:
```

```
    # Путь к существующему файлу helm-chart с пользовательскими значениями. \
      Значения в файле имеют приоритет над перечисленными здесь значениями
    path:
    values:
      image:
        repository:
        tag: 1.23.0
      # Список секретов для загрузки образов im-
      agePullSecrets: []
      # Настройки конфигурации node-local-dns con-
      fig:
        # Переопределение IP-агента node-local-dns
        localDnsIp:
cilium:
  # Настройка пользовательских значений в соответствии с документацией Cilium custom-
  Values:
    # Путь к существующему файлу helm-chart с пользовательскими значениями. \
      Значения в файле имеют приоритет над перечисленными здесь значениями
    path:
    values:
      image:
        repository:
        tag: v1.15.5
      operator:
        image:
          repository:
          tag: v1.15.5 im-
      agePullSecrets: []
      # Управление IP-адресами
      ipam:
        clusterPoolIPv4PodCIDRList: [] cluster-
        PoolIPv4MaskSize:
nginxProxy:
  # Включить/выключить установку балансировщика нагрузки на рабочих узлах (если в\
      kubeadm -> initConfiguration не указан endpoint балансировщика, то \
    nginxProxy переходит в состояние true принудительно) en-
  able: true
  settings: keepalive-
    Timeout: 5m
    healthcheckPort: 8081
    # Настройка пода
    pod:
      # Имя пода
      name: nginx-proxy
      image:
        repository:
        tag: 1.25.2-alpine
        # Политика загрузки образа
        pullPolicy: IfNotPresent
      resources:
```

```
requests:
  cpu: "25m"
  memory: "32M"
```

A.6. МИНИМАЛЬНЫЙ КОНФИГУРАЦИОННЫЙ ФАЙЛ

Минимальный конфигурационный файл, достаточный для создания кластера Боцман с тремя управляющими и тремя рабочими узлами:

```
inventory:
settings:
ssh:
pass: password
port: 22
user: username
nodes: mas-
ters:
- host:
hostname: master01
ip: 10.0.0.1
- host:
hostname: master02
ip: 10.0.0.2
- host:
hostname: master03
ip: 10.0.0.3
workers:
- host:
hostname: worker01
ip: 10.0.0.11
- host:
hostname: worker02
ip: 10.0.0.12
- host:
hostname: worker03
ip: 10.0.0.13
settings: local-
Registry: enable:
true
```

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

ЛВС — локальные вычислительные сети
ОС — операционная система
ОС СН — операционная система специального назначения «Astra Linux Special Edition»
ПО — программное обеспечение
СВТ — средства вычислительной техники

CA — Certification Authority (удостоверяющий центр)
DNS — Domain Name System (система доменных имен)
SSH — Secure Shell (протокол удаленного управления с помощью командной строки)
TLS — Transport Layer Security (протокол защиты транспортного уровня)
VT/SVM — Virtualization Technology/Secure Virtual Machines (технологии виртуализации)