SMB Platform V Synapse Messaging
SMBX Брокер сообщений

ОГЛАВЛЕНИЕ

Описание функций

Термины и определения

Термин/Аббревиатура	Определение
RPC-вызов	Remote Procedure Call — вызов удаленных процедур
Pipeline	Процесс разработки по типу конвейера
АС	Автоматизированная система
DN	Distinguished Name — Уникальное имя сертификата, должно быть уникальным в пределах дерева. В DName описывается содержимое атрибутов в дереве (так называемый путь навигации), требуемое для доступа к конкретной записи ИЛИ базовой (стартовой) записи поиска. DName состоит из серии RDN (Relative Distinguished Names, относительных уникальных имен), определяемых путем перемещения вверх по дереву в направлении его корневой записи (суффикса или базовой записи), и записываемых слева направо
SSL (Secure Sockets Layer)	Secure Sockets Layer — уровень защищенных сокетов

Назначение

Цель создания

Программный продукт SMB Platform V Synapse Messaging представляет собой систему обмена сообщениями в виде брокера сообщений. В качестве транспортной системы используется платформа Apache ActiveMQ Artemis.

Программный компонент SMBX Синапс. Брокер сообщений обеспечивает возможности использования очередей сообщений в интеграционной платформе на основе open-source решений.

Назначение:

- асинхронные адресные RPC-вызовы;
- сглаживание пиков нагрузки;
- развязка по недоступности.

Основные функции (сущностные)

Название функции	Потребитель функции	Аргументы функции	Результат
Создание очереди (опционально включает создание адреса)	Pipeline Оператор	Параметры очереди и имя адреса	Появление нового адреса и новой очереди, привязанной к этому адресу. Появление новой очереди, привязанной к ранее созданному адресу.
Изменение очереди	Pipeline Оператор	Параметры очереди	Изменение параметров существующей очереди.

Название функции	Потребитель функции	Аргументы функции	Результат
Удаление очереди	Pipeline Администратор	Параметры очереди	Удаление существующей очереди.
Изменение адреса	Pipeline Оператор	Параметры адреса	Изменение параметров существующего адреса.
Удаление адреса	Pipeline Администратор	Параметры адреса	Удаление существующего адреса.
Отправка сообщения на адрес	АС-поставщик события	Сообщение	Появление нового сообщения на адресе.
Потребление сообщения из очереди	АС-потребитель события	Сообщение	Получение сообщения из очереди. Удаление сообщения из очереди после потребления.
Назначение прав доступа	Pipeline	Идентификаторы новых ролей и пользователей, DN-поля SSL-сертификатов пользователей, параметры сопоставления пользователей с ролями, параметры	В конфигурации брокера появляются новые роли и пользователи, сопоставления пользователей с ролями, сопоставления ролей с разрешенными операциями.

Название функции	Потребитель функции	Аргументы функции	Результат
		сопоставления ролей с разрешенными операциями.	
Изменение существующих прав доступа	Pipeline	Идентификаторы существующих пользователей, DN SSL-сертификатов пользователей, параметры сопоставления ролей с разрешенными операциями.	В конфигурации брокера изменяются DN SSL-сертификатов пользователей, сопоставления ролей с разрешенными операциями.
Удаление существующих прав доступа	Pipeline	Идентификаторы существующей роли и пользователя.	Удаление идентификатора пользователя из существующего сопоставления с ролью.

Варианты использования

Сценарии использования

1. Создание адресов и очередей. Администратор вносит изменения в конфигурационные файлы для создания адресов и очередей. Альтернативно, Администратор вызывает задачу Jenkins, передавая информацию о создаваемых объектах, для создания адресов и очередей.
2. Выдача прав на адреса и очереди. Администратор вносит изменения в конфигурационные файлы для выдачи прав на адреса и очереди. Альтернативно, Администратор вызывает задачу Jenkins, передавая информацию о назначаемых правах, для выдачи прав на адреса и очереди.
3. Отправка сообщений. Потребитель вызывает метод *Send* клиента Apache Artemis MQ для отправки сообщения.
4. Получение сообщений. Потребитель вызывает метод *Receive* клиента Apache Artemis MQ для получения сообщения.

Руководство по установке

Термины и определения

Термин/Аббревиатура	Определение
DNS	Domain Name System – система доменных имен
SMBX	Сервис по обмену сообщениями из состава программного продукта Platform V Synapse Messaging
inventory	Хранилище конфигурационных файлов и настроек
ОС	Операционная система
DN	Distinguished Name — Уникальное имя сертификата, должно быть уникальным в пределах дерева. В DName описывается содержимое атрибутов в дереве (так называемый путь навигации), требуемое для доступа к конкретной записи ИЛИ базовой (стартовой) записи поиска. DName состоит из серии RDN (Relative Distinguished Names, относительных уникальных имен), определяемых путем перемещения вверх по дереву в направлении его корневой записи (суффикса или базовой записи), и записываемых слева направо

Термин/Аббревиатура	Определение
Нода (Node)	Сервер, один из группы серверов, объединенных в кластер

Системные требования

Пререквизиты установки

Для установки SMBX необходимо предварительно:

- Установить Ansible 2.9.
- Узлы программного компонента SMBX проверить на возможность подключения на порт 61616 иных узлов продукта Platform V Synapse Messaging.
- Обеспечить разрешение имен хостов по IP в рамках всех хостов SMBX (DNS или записи в ./etc/hosts).

При использовании Jenkins (опциональный способ), дополнительно:

- Должен быть доступ в Jenkins и созданы необходимые сущности в нем.
- Все узлы сервиса SMBX должны быть доступны для вызова со стороны Jenkins.
- Должен быть доступ в BitBucket и создан в нем проект для помещения ролей и inventory.
- Должен быть доступ в Nexus и туда должен быть помещен дистрибутив Messaging.

Требования к серверам

SMBX Broker

- от 2-х серверов 4/16/ от 150Гб или 8/32/ от 150Гб;
- требуемый объем HDD и RAM зависит от нагрузки, рассчитывается предварительно;
- ОС Linux с версией kernel не ниже 3.10.0-327;
- На сервере установлены дистрибутивы: java 11, unzip.

Настройка серверов SMBX

Подготовка окружения

Установить в настройки ядра Linux следующие значения:

```
net.ipv4.tcp_syn_retries = 3
net.ipv4.tcp_synack_retries = 3
net.ipv4.tcp_keepalive_time=60
net.ipv4.tcp_keepalive_probes=5
net.ipv4.tcp_keepalive_intvl=1
net.ipv4.tcp_retries2=3
```

где:

- `tcp_syn_retries` - количество попыток передачи SYN-пакета при установлении нового соединения;
- `tcp_synack_retries` - количество попыток передачи SYN, ACK-пакета в ответ на SYN-запрос. Другими словами, максимальное число попыток установить пассивное TCP-соединение, инициированное другим хостом;
- `tcp_keepalive_time` - переменная определяет, как часто (в секундах) следует проверять соединение, если оно давно не используется;
- `tcp_keepalive_probes` - переменная определяет количество попыток проверки жизнеспособности прежде, чем будет принято решении о разрыве соединения;
- `tcp_keepalive_intvl` - переменная определяет интервал, в секундах, проверки жизнеспособность сокета. Это значение учитывается при подсчете времени, которое должно пройти перед тем как соединение будет разорвано;
- `tcp_retries2` - максимальное количество попыток повторной передачи пакетов, до того, как соединение будет считаться разорванным.

Пример:

```
echo "3"    > /proc/sys/net/ipv4/tcp_syn_retries
echo "3"    > /proc/sys/net/ipv4/tcp_synack_retries
echo "60"   > /proc/sys/net/ipv4/tcp_keepalive_time
echo "5"    > /proc/sys/net/ipv4/tcp_keepalive_probes
echo "1"    > /proc/sys/net/ipv4/tcp_keepalive_intvl
echo "3"    > /proc/sys/net/ipv4/tcp_retries2
```

Создать пользователя artemis*, выдать пользователю права sudoedit для создания сервисов и права для управления сервисами*

```
artemis ALL = NOPASSWD: /bin/systemctl start artemis, /bin/systemctl stop artemis, /bin/systemctl status artemis, /bin/systemctl restart
artemis, /bin/systemctl enable artemis, /bin/systemctl disable artemis, /bin/systemctl daemon-reload, /bin/sudoedit
/etc/systemd/system/artemis.service
```

Создать разделы на диске

- ```/opt/Apache/artemis``` - 10Гб, владелец `artemis`;
- ```/artemis-data/``` - от 100Гб, владелец `artemis` (объем зависит от нагрузки, рассчитывается предварительно).

Создать JKS хранилище с сертификатом для брокеров, подписав запрос на сертификат Удостоверяющим центром (далее - УЦ), доверенным для всех клиентов

Увеличить лимит дескрипторов для пользователя **artemis**

В файле `/etc/security/limits.conf` для пользователя *artemis* прописать:
```
nofile 128000
nproc 16384
```

Создание JKS хранилища и сертификатов

Для создания сертификата используется утилита **keytool.exe** из состава JDK. Для получения сертификата необходимо выполнить следующие действия.

1. Создать хранилище ключей и сертификатов:
   1.1. `keytool -genkey -keyalg RSA -alias Test -keystore [путь и имя файла с хранилищем ключей и сертификатов] -storepass [пароль для хранилища ключей и сертификатов] -validity 1440 -keysize 2048 -dname CN=[по правилам описанным ниже],OU=00CA,O=Org,L=Moscow,ST=Moscow,C=RU`
   Например: `keytool -genkey -keyalg RSA -alias ks -keystore D:\ks.jks -storepass 23101989 -validity 1440 -keysize 2048 -dname CN=00CA0001P.TestProducer.zzzz,OU=00CA,O=Org,L=Moscow,ST=Moscow,C=RU.`
   1.2. `keytool -certreq -alias Test -keyalg RSA -file [путь и имя файла с запросом на сертификат] -keystore [путь и имя файла с хранилищем ключей и сертификатов, созданного на шаге 1]`
2. Создать запрос на сертификат.
   Например: `keytool -certreq -alias ks -keyalg RSA -file D:\testProducer.csr -keystore D:\ks.jks`.
3. Отправить запрос на сертификат в УЦ.
4. Импортировать сертификаты в хранилище ключей.
   Полученные от УЦ файл с сертификатом и файлы с корневыми сертификатами необходимо импортировать в хранилище ключей и сертификатов при помощи утилиты **keytool.exe**.
   4.1. Первым необходимо импортировать корневой сертификат:
   `keytool -import -alias ks1 -file [путь и имя файла с корневым сертификатом, полученным от УЦ] -keystore [путь и имя файла с хранилищем ключей и сертификатов, созданного на шаге 1]`
   4.2. Вторым необходимо импортировать сертификат УЦ:
   `keytool -import -alias ks2 -file [путь и имя файла с сертификатом УЦ] -keystore [путь и имя файла с хранилищем ключей и сертификатов, созданного на шаге 1]`

4.3. Последним импортируется TLS-сертификат:

```
keytool -import -alias cmks -file [путь и имя файла с TLS-сертификатом, полученным от УЦ] -keystore [путь и имя файла с хранилищем ключей и сертификатов, созданного на шаге 1]
```

Формирование DN сертификата

CN = 00ZZ0001M.monitoringsystem.segment.contour.AS (пример)

Рекомендуется заполнять следующим образом:

- код ЦА - «00CA»;
- порядковый номер ключа (4 цифры) – до появления централизованной системы управления сертификатами не заполняется;
- тип ключа:
  - P - Producer (Продьюсер)
  - C - Consumer (Консьюмер)
  - S - Support monitoring (Инженер мониторинга)
  - M - Monitoring System (Система мониторинга)
  - A - Access manager (Администратор доступа)
  - E - Maintenance Engineer (Инженер сопровождения)
  - B - Broker (Брокер)
  - I - InfoSec Admin (Администратор безопасности)

  Для одного бизнес-сервиса допускается сертификат с несколькими ролями. Например сертификат системы, которая является одновременно поставщиком и потребителем событий, должен иметь тип ключа «PC» в DN сертификата.
- логин учетной записи пользователя;
- сетевой сегмент;
- тип (контур) кластера (только для брокера и администрирования, **роли producer и consumer не должны включать данный раздел**);

OU = OrganizationalUnitName

O = Organization

L = LocalityName

ST = StateOrProvinceName

C = CountryName

Установка

Ручной способ установки

1. Проверить, что на сервер, с которого будет производиться установка, установлен Ansible и с него доступны все узлы SMBX.
2. Распаковать дистрибутив и поместить содержимое директории *modules* в директорию *scripts/Ansible*.
3. Все дальнейшие операции производить из директории *scripts/Ansible*.
4. Создать свой inventory (например, *ID*). Для этого создать директорию *ID* в папке *inventories*.
5. Создать структуру файлов по примеру, указанному в таблице.

Файл конфигурации	Секция	Параметры	Описание значения
`group_vars/all/vars.yml`		`ansible_user:` `ansible_port:`	Пользователь и порт для ssh-соединения ansible
	`artemis:`		
		`distr:`	Наименование дистрибутива
		`installdir:`	Абсолютный путь на конечном сервере до приложения (брокер располагается в ./broker/)
		`datadir:`	Абсолютный путь до данных брокера
		`logdir:`	Абсолютный путь до логов брокера
		`cleanLog:`	Очистка logdir при установке
		`cleanData:`	Очистка datadir при установке (чистая установка брокера)
		`xms:`	Обычно выставляется в 90% доступного RAM сервера

Файл конфигурации	Секция	Параметры	Описание значения
		`xmx`:	Обычно выставляется в 90% доступного RAM сервера
		`global_max_size`:	Максимальный размер всех очередей брокера. Обычно выставляется в 50% xmx
		`user`	Логин пользователя под которым работает кластер
		`password`:	Пароль пользователя под которым работает кластер
		`console_port`:	Порт для веб консоли
		`data_port`:	Порт для подключения
		`jmx_port`:	Порт для получения JMX метрик
		`jmx_user`:	Пользователь для получения JMX метрик
		`jmx_password`:	Пароль пользователя для получения JMX метрик
		`persistence`:	Хранение сообщений на диске (если true, то используется режим replication, настройка через is_master/is_slave в inventory)
		`keyStorePath`:	Путь от inventories/*стенд*/ до файла с keyStore
		`keyStorePassword`:	Пароль от keyStore
		`keyPassword`:	Пароль от ключа в хранилище

Файл конфигурации	Секция	Параметры	Описание значения
		`trustStorePath`:	Путь от inventories/*стенд*/ до файла с trustStore
		`trustStorePassword`:	Пароль от trustStore
		`enabled_protocols`:	Доступные для подключения протоколы
		`ui_mtls`:	Закрытие ui на вход по сертификату (а не логин/пароль)
		`admins`:	Список DN администраторов сопровождения

`inventory.yml`	`[artemis]`		Перечень хостов брокеров в виде записей:
			`<fqdn сервера>`
			`advertised_host=<fqdn для подключения извне>`
			Пример:
			`[kafka]`
			`my-host.org`
			`advertised_host=my-host.org`
			`my-host.org`
			`advertised_host=my-host.org`
			`my-host.org`
			`advertised_host=my-host.org`

Папка `ssl`			Поместить jks-хранилище сертификата узла SMBX

Важно: Все параметры inventory аннотированы, приведенные выше параметры являются теми, на которые требуется обратить внимание (стендозависимые). В случае необходимости кастомизации рекомендуется читать аннотации параметров.

Использование vault для шифрования паролей

При хранении сенситивной информации в Git ее рекомендуется шифровать. Для этого можно использовать утилиту ansible-vault.

Для шифрования пароля следует выполнить команду:

```
ansible-vault encrypt_string -n jks_password 'ENCRYPT_STRING',
```
где `ENCRYPT_STRING` - строка, которую необходимо зашифровать, а `jks_password` - имя переменной.

При запросе вести пароль для шифрования, а на выходе получится то, что нужно занести в inventory:

```
jks_password: !vault | $ANSIBLE_VAULT;1.1;AES256
303236323463316162663632343033386639653665393435353531336261653165646332376265 36
393233383135373935613537646332336332613333333838340a336338623837303937393538313939
376265313834323666623034663637616165663936383065646236613231333561336138633313032
396665353164363166 0a666136623361613863643137396663653363316139316566393366653838
3039
```

Аналогично, возможно шифрование файлов:

```
ansible-vault encrypt <имя файла>
```

Запуск установки

Запустить установку командой:

```
ansible-playbook -i inventories/<ID>/inventory artemis.yml --ask-vault-pass
```
где `ID` - имя недавно созданного inventory.

Установка будет производиться на все хосты из inventory. Для ограничения списка узлов используем команду:

```
ansible-playbook -i inventories/<ID>/inventory artemis.yml --ask-vault-pass -l <узлы через запятую без пробелов>
```

Установка с помощью Jenkins (опциональный способ)

1. Распаковать дистрибутив и поместить содержимое папки *scripts* в BitBucket.
2. Создать и настроить inventory (см. раздел *Ручной способ установки* выше) и поместить изменения в BitBucket.
3. В Jenkins создать Jenkins Pipeline с получением скриптов развертывания из BitBucket.
    - Pipepline script from SCM

        - SCM - GIT
        - repository url - ссылка на репозиторий, куда поместили скрипты.
        - Выбираем или добавляем учетные данные для доступа к BitBucket
    - Script_path - относительный путь *SYN_custom.groovy*. Убедиться, что не стоит галочка *Lightweight checkout*.
4. Сохранить получившийся Jenkins Pipeline и запустить его.
5. Проверить, что после запуска подгрузились дополнительные параметры.
6. При необходимости, поменять для параметров значения по умолчанию. Например, изменить имя используемых *credentials*.

Запуск установки

При запуске задания Jenkins по установке в параметрах выбирать нужный inventory, playbook `artemis.yml`, а в поле `customURL` указать ссылку на дистрибутив.

Настройка сервисов

Создать файл сервиса **/etc/systemd/system/artemis.service** (при необходимости отредактировать):

```
[Unit]
Description=ActiveMQ Artemis service
After=local-fs.target network.serive

[Service]
WorkingDirectory=/opt/Apache/artemis/broker
Type=simple
User=artemis
Group=artemis
ExecStart=/opt/Apache/artemis/broker/bin/artemis run
ExecStop=/opt/Apache/artemis/broker/bin/artemis stop
LimitNOFILE=512000
LimitNPROC=512000
Restart=on-failure
RestartSec=30

[Install]
WantedBy=default.target
```

Обновить список сервисов командой:

```
sudo systemctl daemon-reload
```

Обновление

Общий подход

Поузловое обновление кластера через установку новой версии дистрибутива:

1. Проверить, что в *inventory* в **vars.yml** установлено cleanData = false.
2. Произвести установку.

   2.1. При ручной установке: при помощи команды:

   ```
   ansible-playbook -i inventories/<ID>/inventory artemis.yml --ask-vault-pass -l <узел для обновления>
   ```

   2.2. При использовании Jenkins: через запуск с параметром *only_on_host*, в котором указать хост из *inventory*, на котором будет производиться обновление.

3. Убедиться, что произошел успешный запуск брокера через консоль администрирования по адресу `https://_ip_:8161/console` (где `_ip_` - ip адрес брокера, `8161` - порт из artemis.console_port).
4. Перейти к следующему узлу.

Проверка работоспособности

По завершению установки скрипты автоматически проверяют корректность и успешность произведенных действий. При возникновении ошибки при ручной установке обработка скриптом остановится, а в консоль будет выведен текст ошибки. При возникновении ошибки при автоматической установке Jenkins Build завершиться с ошибкой, а в Console Output будет содержаться сообщение об ошибке.

Откат

Понодное обновление кластера через установку старой версии дистрибутива.

Часто встречающиеся проблемы и пути их устранения

Не выявлено.

Чек-лист валидации установки

Проверить, что:

- установка через Ansible прошла корректно, без ошибок;
- статус Cluster info в консоли администрирования соответствует ожиданиям.

Руководство по системному администрированию

Термины и определения

Термин/Аббревиатура	Определение
SMBX	Сервис по обмену сообщениями из состава программного продукта Platform V Synapse Messaging
ОС	Операционная система
DN	Distinguished Name — Уникальное имя сертификата, должно быть уникальным в пределах дерева. В DName описывается содержимое атрибутов в дереве (так называемый путь навигации), требуемое

Термин/Аббревиатура	Определение
	для доступа к конкретной записи ИЛИ базовой (стартовой) записи поиска. DName состоит из серии RDN (Relative Distinguished Names, относительных уникальных имен), определяемых путем перемещения вверх по дереву в направлении его корневой записи (суффикса или базовой записи), и записываемых слева направо
label	Наименование, метка
Jenkins slave	Сервер, на котором исполняются задачи Jenkins
Нода (Node)	Сервер, один из группы серверов, объединенных в кластер
YAML	Yet Another Markup Language - формат сериализации данных, близкий к языкам разметки, но ориентированный на удобство ввода-вывода типичных структур данных многих языков программирования
credentials	Реквизиты для подключения куда-либо
SSH	Сетевой протокол прикладного уровня, позволяющий производить удаленное управление операционной системой и туннелирование TCP-соединений
APM	Автоматизированное рабочее место

Термин/Аббревиатура	Определение
JMS	Java Message Service — стандарт промежуточного ПО для рассылки сообщений, позволяющий приложениям, выполненным на платформе Java, создавать, посылать, получать и читать сообщения
JMX	Управленческие расширения Java (англ. Java Management Extensions) — технология Java, предназначенная для контроля и управления приложениями, системными объектами, устройствами (например, принтерами) и компьютерными сетями
AMQ Broker	ActiveMQ Broker — брокер сообщений с открытым исходным кодом (распространяется под лицензией Apache 2.0), реализующий спецификацию JMS 1.1 с возможностями кластеризации, возможностью использовать для хранения сообщений различные СУБД, кэшированием, ведением журналов
MBEAN	Java-объекты, которые реализуют определенный интерфейс

Сценарии администрирования

Работа с кластером

Ручной способ

На сервере, с которого производилась установка, выполнить команду:

`ansible-playbook -i inventories/<ID>/inventory artemis.yml --ask-vault-pass` где *ID* — имя созданного inventory.

С помощью Jenkins (опциональный способ)

Наименование задания Jenkins: **artemis_custom**

Скрытые параметры настройки:

- nexus_repository_url - ссылка на репозиторий Nexus для генерации списка nexus_version
- nexus_group_id - группа для генерации списка nexus_version
- nexus_artifact_id - артефакт для генерации списка nexus_version
- nexus_user_cred - credential типа Username with password для выкачивания дистрибутива из Nexus
- vault_cred - credential типа Secret file для расшифровывания Ansible паролей
- server_ssh_cred - credential типа SSH Username with private key для подключения к серверам установки

Параметры запуска:

- job_config_renew - технический параметр для обновления конфигурации джобы Jenkins (параметры, credential, плейбуки, теги)
- inventory - имя inventory для установки
- nexus_version - используемая версия дистрибутива из Nexus (при включенном параметре downloadFromNexus и пустом customURL)
- jenkins_slave - label slave Jenkins для установки
- emailto - список адресов почт для отправки технических писем о начале/окончании установки с логами
- playbook - запускаемый плейбук
    - artemis.yml - работа с Artemis
    - artemis_queues.yml - работа с адресами и очередями Artemis (через конфигурацию в inventory)
    - artemis_queues_roles.yml - работа с адресами, очередями и пользователями Artemis (через конфигурацию в inventory)
    - artemis_roles.yml - работа с ролями пользователей Artemis (через конфигурацию в inventory)
- tags – используемые теги
    - admin_roles - назначение прав администратора сопровождения
    - backup - выполнение бекапа artemis.installdir в директорию artemis.backup_installdir. Входит в тег install
    - backup_remove - удаление бекапа из artemis.backup_installdir
    - backup_restore - очистка artemis.installdir и восстановление из бекапа из artemis.backup_installdir
    - distribute - загрузка дистрибутива, скачанного из Nexus, во временную директорию на сервера установки tmp_dir
    - install - распаковка дистрибутива из tmp_dir в artemis.installdir, создание брокера, настройка конфигурационных файлов
    - start - запуск брокера
    - status - проверка статуса (если порт брокера открыт, считаем, что брокер запущен). Входит в тег start
    - stop - остановка брокера
    - template - подготовка конфигурации логирования из темплейта, модифицирует logback.xml. Входит в тег install
    - xml_change - модификация конфигурационных xml файлов, указанных в блоке artemis.xml_change. Входит в тег install
- downloadFromNexus - выкачивание дистрибутива из Nexus (требуется только для тэга distribute в artemis.yml)
- only_on_host - выполнять плейбук на определенных хостах из inventory
- customURL - полная ссылка на используемый дистрибутив (параметр nexus_version игнорируется)
- m36_check - проверка метрики m36 (для ПРОМ установок)

Работа с конфигурационными дистрибутивами

Создание дистрибутива

Ручной способ

Необходимо в соответствующем inventory заполнить файл `group_vars/all/vars.yml`:

```
artemis_queues:
  list:
    - name: <имя очереди>
      address: <имя адреса>
    - name: <имя очереди2>
      address: <имя адреса2>
      max_size_bytes: <максимальный размер сообщений в адресе>

artemis_roles:
  - user: <DN пользователя>
    queue: <имя очереди>
    address: <имя адреса>
    type: <тип операции, доступны send/consume/browse>
```

Весь список настраиваемых параметров можно посмотреть в **roles/artemis_queues/defaults/main.yml** и **roles/artemis_roles/defaults/main.yml**.

С помощью Jenkins (опциональный способ)

Наименование задания Jenkins: **artemis_config_create**

Параметры настройки:

Параметры запуска:

```
- majorVersion - мажорная версия дистрибутива. Пример полный версии дистрибутива - ${majorVersion}.001.00_suffix-00
- nexusUrl- ссылка на репозиторий Nexus
- nexusCred - credential типа Username with password для выкачивания дистрибутива из Nexus
- suffix - суффикс для дистрибутива в формате <ServiceName>_<C|P>_<System>, где
         - ServiceName - наименование сервиса;
         - C|P - является ли дистрибутив дистрибутивом потребителя сервиса (C) или поставщика сервиса (P);
         - System - система-владелец дистрибутива;
- rebuildVersion - версия дистрибутива для пересборки или пусто, если выпускается новый релиз
- jenkinsSlaveNode - label slave Jenkins для установки
```

- emailTo - список адресов почт для отправки технических писем о начале/окончании установки с логами
- artemisConfig - описание конфигурационного дистрибутива для создания адресов, очередей и прав для пользователями в YAML формате

Развертывание дистрибутива

Ручной способ

На сервере, с которого производилась установка, выполнить команду:

```
ansible-playbook -i inventories/<ID>/inventory artemis_queues_roles.yml --ask-vault-pass
```

где *ID* - имя созданного inventory

С помощью Jenkins (опциональный способ)

Наименование задания Jenkins: **artemis_config_deploy**

Скрытые параметры настройки:

- nexus_repository_url - ссылка на репозиторий Nexus для генерации списка nexus_version
- nexus_group_id - группа для генерации списка nexus_version
- nexus_artifact_id - артефакт для генерации списка nexus_version
- nexus_user_cred - credential типа Username with password для выкачивания дистрибутива из Nexus
- vault_cred - credential типа Secret file для расшифровывания Ansible паролей
- server_ssh_cred - credential типа SSH Username with private key для подключения к серверам установки

Параметры запуска:

- job_config_renew - технический параметр для обновления конфигурации задания Jenkins (параметры, credential, плейбуки, теги)
- inventory - имя inventory для установки
- nexus_version - используемая версия дистрибутива из Nexus (при пустом customURL)
- custom_version - список версий дистрибутива из Nexus (при пустом customURL, nexus_version игнорируется)
- customURL - полная ссылка на используемый дистрибутив (параметр nexus_version игнорируется)
- tags
• - erase_queue - удаление очереди
• - erase_role - удаление роли
- emailto - список адресов почт для отправки технических писем о начале/окончании установки с логами

Администрирование через АРМ администратора

Вход в АРМ

Для аутентификации пользователя используются сертификаты, администратору необходимо выпустить сертификат, сконвертировать его в формат *p12*:

```
keytool -importkeystore -srckeystore <mycert>.jks -destkeystore <mycert>.p12 -deststoretype PKCS12
```

После чего импортировать его в браузер (обычно, Настройки→Безопасность→Управление сертификатами).

АРМ доступен на каждом брокере по адресу https://*ip*:8161/console (где *ip* - ip адрес брокера, 8161 - порт из порт из artemis.console_port в настройках развертывания).

Просмотр топологии брокера

- Вкладка `Broker diagram`, показывает список объектов на конкретном брокере и подключения к брокерам кластера. Нажатие на элемент выводит его свойства.
- Вкладка `Addresses`, показывает список адресов с возможностью фильтрации. Нажатие на ссылку в виде количества очередей, перенаправляет к списку очередей данного адреса.
- Древовидный список слева, объекты под `addresses`.

Основные атрибуты адреса:

- `Address size` - размер хранимых сообщений во всех очередях адреса в байтах;
- `Number of messages` - количество сообщений во всех очередях адреса;
- `Roles as json` - права на адрес.

Основные атрибуты очереди:

- `Consumer count` - количество подключенных читателей;
- `Message count` - количество сообщений в очереди.

Просмотр подключений к адресу/очереди

Вкладка *Consumers* - перечень подключений на чтение.

Вкладка *Producers* - перечень подключений на запись.

События системного журнала

Общая настройка логирования

По умолчанию SMBX использует JBoss Logging framework для логирования и настраивается с помощью файла **logging.properties**, находящегося в каталоге *etc* рассматриваемого брокера.

Для использования logback файла достаточно добавить "`-Dorg.jboss.logging.provider=slf4j -Dlogback.configurationFile=/path/to/logback.xml`" в *artemis.profile*, находящийся в каталоге *etc* рассматриваемого брокера и добавить библиотеки `logback-classic-*.jar` и `logback-core-*.jar` в каталог lib.

Есть несколько доступных логеров:

- `org.apache.activemq.artemis.core.server` - логирует основной сервер;
- `org.apache.activemq.artemis.integration.bootstrap` - логирует вызовы начальной загрузки;
- `org.apache.activemq.artemis.jms` - логирует вызовы JMS;
- `org.apache.activemq.artemis.journal` - логирует вызовы журнала;
- `org.apache.activemq.artemis.utils` - логирует вызовы утилит;
- `org.jboss.logging` - регистрирует все вызовы, не обработанные основными логерами.

Пример **logging.properties**:

```
# Root logger option
loggers=org.jboss.logging,org.apache.activemq.artemis.core.server,org.apache.activemq.artemis.utils,org.apache.activemq.artemis.journal,org.apache.activemq.artemis.jms,org.apache.activemq.artemis.ra

# Root logger level
logger.level=INFO
# Apache ActiveMQ Artemis logger levels
logger.org.apache.activemq.artemis.core.server.level=INFO
logger.org.apache.activemq.artemis.utils.level=INFO
logger.org.apache.activemq.artemis.jms.level=DEBUG

# Root logger handlers
logger.handlers=FILE,CONSOLE
```

```
# Console handler configuration
handler.CONSOLE=org.jboss.logmanager.handlers.ConsoleHandler
handler.CONSOLE.properties=autoFlush
handler.CONSOLE.level=FINE
handler.CONSOLE.autoFlush=true
handler.CONSOLE.formatter=PATTERN

# File handler configuration
handler.FILE=org.jboss.logmanager.handlers.FileHandler
handler.FILE.level=FINE
handler.FILE.properties=autoFlush,fileName
handler.FILE.autoFlush=true
handler.FILE.fileName=activemq.log
handler.FILE.formatter=PATTERN

# Formatter pattern configuration
formatter.PATTERN=org.jboss.logmanager.formatters.PatternFormatter
formatter.PATTERN.properties=pattern
formatter.PATTERN.pattern=%d{HH:mm:ss,SSS} %-5p [%c] %s%E%n
```

Пример artemis.profile для брокера, использующего logback.xml:

```
ARTEMIS_HOME='/path/to/apache-artemis'
ARTEMIS_INSTANCE='/path/to/apache-artemis/mybroker'
ARTEMIS_DATA_DIR='/path/to/apache-artemis/mybroker/data'
ARTEMIS_ETC_DIR='/path/to/apache-artemis/mybroker/etc'
ARTEMIS_OOME_DUMP='/path/to/apache-artemis/mybroker/log/oom_dump.hprof'

ARTEMIS_INSTANCE_URI='file:/path/to/apache-artemis/mybroker/'
ARTEMIS_INSTANCE_ETC_URI='file:/path/to/apache-artemis/mybroker/etc/'

HAWTIO_ROLE='amq'

# Java Opts добавлен путь до logback.xml
if [ -z "$JAVA_ARGS" ]; then
    JAVA_ARGS=" -XX:+PrintClassHistogram -XX:+UseG1GC -XX:+UseStringDeduplication -Xms512M -Xmx2G -Dhawtio.disableProxy=true -
Dhawtio.realm=activemq -Dhawtio.offline=true -Dhawtio.rolePrincipalClasses=org.apache.activemq.artemis.spi.core.security.jaas.RolePrincipal -
Djolokia.policyLocation=${ARTEMIS_INSTANCE_ETC_URI}jolokia-access.xml -Dorg.jboss.logging.provider=slf4j -
Dlogback.configurationFile=/path/to/logback.xml"
fi
```

Логирование событий аудита

При использовании стандартного логирования логирование событий аудита выключено. Для включения логирования событий аудита необходимо в файле logging.properties заменить уровень логирования ERROR на INFO в следующих строках (по умолчанию, логирование будет производится в файл log/audit.log):

- `logger.org.apache.activemq.audit.base.level=INFO` - регистрация доступа ко всем методам объекта JMX;
- `logger.org.apache.activemq.audit.message.level=INFO` - регистрация операций с сообщениями, таких как отправка, потребление и просмотр сообщений;
- `logger.org.apache.activemq.audit.resource.level=INFO` - регистрация событий аутентификации, создания или удаления ресурсов брокера из консоли управления JMX или AMQ Broker, а также просмотр сообщений в консоли управления.

Отправка логов в Kafka

Для отправки логов в Kafka помимо настройки использования **logback.xml**, необходимо добавить в каталог lib следующие библиотеки: `logback-kafka-appender-*.jar`, `kafka-clients-*.jar` (для работы кафка аппендера), `grok-*.jar`, `gson-*.jar` (для форматирования сообщений)

Пример файла **logback.xml**, отправляющего сообщения в формате { "message": "AMQ101000: Starting ActiveMQ Artemis Server", "timestamp": "2021-03-15T18:22:31.541" }

```
<configuration debug="false">
    <appender name="STDOUT"
        class="ch.qos.logback.core.ConsoleAppender">
        <layout class="ch.qos.logback.classic.PatternLayout">
            <Pattern>
                %d{yyyy-MM-dd HH:mm:ss} [%thread] %-5level %logger{36} - %msg%n
            </Pattern>
        </layout>
    </appender>

    <appender name="kafka_all" class="com.github.danielwegener.logback.kafka.KafkaAppender">
         <encoder class="com.github.danielwegener.logback.kafka.encoding.LayoutKafkaMessageEncoder">
            <layout class="ru.sbt.logback.layout.Grok">
                <message>message</message>
                <timestamp>timestamp</timestamp>
            </layout>
        </encoder>
        <deliveryStrategy class="com.github.danielwegener.logback.kafka.delivery.AsynchronousDeliveryStrategy"/>
        <topic>artemis</topic>
```

```
    <producerConfig>bootstrap.servers=localhost:9092</producerConfig>
</appender>
<appender name="kafka_audit_base" class="com.github.danielwegener.logback.kafka.KafkaAppender">
    <encoder class="com.github.danielwegener.logback.kafka.encoding.LayoutKafkaMessageEncoder">
        <layout class="ru.sbt.logback.layout.Grok">
            <message>message</message>
            <timestamp>timestamp</timestamp>
        </layout>
    </encoder>
    <deliveryStrategy class="com.github.danielwegener.logback.kafka.delivery.AsynchronousDeliveryStrategy"/>
    <topic>artemis_audit_base</topic>
    <producerConfig>bootstrap.servers=localhost:9092</producerConfig>
</appender>
<appender name="kafka_audit_resource" class="com.github.danielwegener.logback.kafka.KafkaAppender">
     <encoder class="com.github.danielwegener.logback.kafka.encoding.LayoutKafkaMessageEncoder">
        <layout class="ru.sbt.logback.layout.Grok">
            <message>message</message>
            <timestamp>timestamp</timestamp>
        </layout>
    </encoder>
    <deliveryStrategy class="com.github.danielwegener.logback.kafka.delivery.AsynchronousDeliveryStrategy"/>
    <topic>artemis_audit_resource</topic>
    <producerConfig>bootstrap.servers=localhost:9092</producerConfig>
</appender>
    <appender name="kafka_audit_message" class="com.github.danielwegener.logback.kafka.KafkaAppender">
     <encoder class="com.github.danielwegener.logback.kafka.encoding.LayoutKafkaMessageEncoder">
        <layout class="ru.sbt.logback.layout.Grok">
            <message>message</message>
            <timestamp>timestamp</timestamp>
        </layout>
    </encoder>
    <deliveryStrategy class="com.github.danielwegener.logback.kafka.delivery.AsynchronousDeliveryStrategy"/>
    <topic>artemis_audit_message</topic>
    <producerConfig>bootstrap.servers=localhost:9092</producerConfig>
</appender>


<logger name="org.apache.activemq.audit.base" level="INFO" additivity="false">
    <appender-ref ref="kafka_audit_base" />
</logger>
<logger name="org.apache.activemq.audit.resource" level="INFO" additivity="false">
    <appender-ref ref="kafka_audit_resource" />
</logger>
<logger name="org.apache.activemq.audit.message" level="INFO" additivity="false">
    <appender-ref ref="kafka_audit_message" />
```

```
    </logger>
    <root level="INFO">
        <appender-ref ref="STDOUT" />
        <appender-ref ref="kafka_all" />
    </root>

</configuration>
```

События мониторинга

Мониторинг производится самостоятельно через преднастроенный JMX-порт.

Отслеживаются следующие метрики:

Метрика	Описание	Триггер	MBEAN Name
`org.apache.activemq.artemis.[broker].AddressMemoryUsagePercentage`	процент использования памяти брокером относительно global-max-size	0.7	`org.apache.activemq.artemis:broker="<broker>"/AddressMemoryUsagePercentage`
`org.apache.activemq.artemis.[broker].DiskStoreUsage`	процент использования дискового пространства	0.7	`org.apache.activemq.artemis:broker="<broker>"/DiskStoreUsage`
`org.apache.activemq.artemis.[broker].Uptime`	время жизни брокера	-	`org.apache.activemq.artemis:broker="<broker>"/Uptime`

Метрика	Описание	Триггер	MBEAN Name
org.apache.activemq.artemis.[broker].TotalMessagesAdded	получено сообщений с момента запуска брокера	-	org.apache.activemq.artemis:broker="<broker>"/TotalMessagesAdded
org.apache.activemq.artemis.[broker].TotalMessagesAcknowledged	забрано сообщений с момента запуска брокера	-	org.apache.activemq.artemis:broker="<broker>"/TotalMessagesAcknowledged
org.apache.activemq.artemis.[broker].addresses.DLQ.MessageCount	количество недоставленных сообщений в очередях адреса	растет	org.apache.activemq.artemis:broker="<broker>",component=addresses,address="DLQ"/MessageCount
org.apache.activemq.artemis.[broker].addresses.ExpiryQueue.MessageCount	количество просроченных сообщений в очередях адреса	растет	org.apache.activemq.artemis:broker="<broker>",component=addresses,address="ExpiryQueue"/MessageCount
org.apache.activemq.artemis.[broker].addresses.[queues].anycast.[queueName].MessageCount	количество сообщений в очереди	-	org.apache.activemq.artemis:broker="<broker>",component=addresses,address="<queue>",subcomponent=queues,routing-type="anycast",queue="<queue>"/MessageCount
org.apache.activemq.artemis.[broker].addresses.[queues].anycast.[queueName].MessagesAdded	количество сообщений отправленных в очередь с	-	org.apache.activemq.artemis:broker="<broker>",component=addresses,address="<queue>",subcomponent=queues,routing-type="anycast",queue="<queue>"/MessagesAdded

Метрика	Описание	Триггер	MBEAN Name
	момента ее создания		
org.apache.activemq.artemis.[broker].addresses.[queues].anycast.[queueName].MessagesAcknowledged	количество сообщений забранных из очереди с момента ее создания	-	org.apache.activemq.artemis:broker="<broker>",component=addresses,address="<queue>",subcomponent=queues,routing-type="anycast",queue="<queue>"/MessagesAcknowledged
org.apache.activemq.artemis.[broker].addresses.[queues].anycast.[queueName].ConsumerCount	количество подключенных потребителей	-	org.apache.activemq.artemis:broker="<broker>",component=addresses,address="<queue>",subcomponent=queues,routing-type="anycast",queue="<queue>"/ConsumerCount

Часто встречающиеся проблемы и пути их устранения.

Не выявлено.