СИСТЕМА УСЛОВНОГО ДОСТУПА «CAS RSCC»

Руководство по эксплуатации

2025

Эксплуатация системы условного доступа «CAS RSCC»

Настоящая инструкция описывает процесс настройки системы «CAS RSCC» после успешной установки всех компонентов. Она предназначена для технических специалистов, ответственных за развертывание и конфигурирование системы.

1. Общие принципы настройки

1.1. Предварительные Условия

Перед началом настройки убедитесь, что выполнены следующие условия:

- Все необходимые компоненты системы установлены согласно документу «Руководство по установке».
- Kubernetes кластер функционирует (рекомендуемая конфигурация: 3 worker, 2 master).
- Docker Registry содержит все необходимые образы из установочного архива.
- Базы данных (PostgreSQL, MongoDB, Redis, ClickHouse) развернуты и доступны.
- Брокер сообщений Kafka настроен и функционирует.

1.2. Структура Конфигураций

Конфигурации для микросервисов системы «CAS RSCC» определяются на основе стандартных принципов развертывания в контейнерных средах, таких как Kubernetes. Основные параметры, которые необходимо задать, включают:

- **Версия API и тип объекта**: определяют, как система управления контейнерами будет интерпретировать и применять конфигурацию (например, для развертывания приложения или создания сервиса).
- **Метаданные**: включают имя объекта и метки для его идентификации и группировки.
- **Спецификация развертывания**: определяет количество экземпляров (реплик) приложения, правила выбора подов и шаблон для создания контейнеров.
- **Переменные окружения**: используются для передачи параметров подключения к базам данных, брокерам сообщений и другим внешним сервисам.

1.3. Процедура Настройки

Для настройки каждого микросервиса выполните следующие шаги:

1. Подготовьте конфигурационный файл для каждого микросервиса, используя соответствующий формат (например, YAML для Kubernetes).
2. Заполните учетные данные и адреса серверов в соответствующих параметрах конфигурации.
3. Выполните проверку конфигурации без применения (dry-run) для валидации:
   kubectl apply -f <filename>.yml --dry-run=client
4. При успешной валидации примените конфигурацию:
   kubectl apply -f <filename>.yml

2. Настройка компонентов

2.1. Компонент “CAS” (Подсистема условного доступа)

2.1.1. Настройка cas-mdb

Назначение: Основной сервер API с модулями hardware, cam-modul, client- hardware, client, monitoring, procedures, udp_streams, channels.

Необходимые параметры конфигурации:

- **Образ Docker**: Укажите ссылку на Docker образ и тег сервиса cas-mdb.
- **Переменные окружения**: Задайте параметры подключения к базам данных и брокеру сообщений:
  - POSTGRES_URL: Адрес и учетные данные для PostgreSQL (например, postgresql+asyncpg://ЛОГИН:ПАРОЛЬ@АДРЕС_POSTGRESQL/cas_mdb).
  - REDIS_URL: Адрес Redis (например, redis://@АДРЕС_REDIS).
  - MONGO_URL: Адрес и учетные данные для MongoDB (например, mongodb://ПОЛЬЗОВАТЕЛЬ:ПАРОЛЬ@АДРЕС_MONGODB/).
  - KAFKA_BROKERS: Адрес брокера Kafka (например, АДРЕС_KAFKA:9092).
- **Ресурсы**: определите лимиты и запросы по памяти (например, 1Gi лимит, 512Mi запрос).
- **Порты**: Укажите внутренний порт контейнера (например, 21300) и внешний порт NodePort (например, 31300).

2.1.2. Настройка cas-ssm

Назначение: Сервер управления правами с модулями users, roles, self_user, tokens, user_notification, integration_access_token.

Необходимые параметры конфигурации:

- **Образ Docker**: Укажите ссылку на Docker образ и тег сервиса cas-ssm.
- **Переменные окружения**: Задайте параметры подключения к базам данных:
  - REDIS_URL: Адрес Redis (например, redis://@АДРЕС_REDIS).
  - MONGO_URL: Адрес и учетные данные для MongoDB (например, mongodb://ПОЛЬЗОВАТЕЛЬ:ПАРОЛЬ@АДРЕС_MONGODB/).
- **Ресурсы**: определите лимиты и запросы по памяти (например, 512Mi лимит).
- **Порты**: Укажите внутренний порт контейнера (например, 21600) и внешний порт NodePort (например, 31600).

2.1.3. Настройка cas-stream-analyzer

Назначение: Анализатор потоков с модулями stream-stats, channels.

Необходимые параметры конфигурации:

- **Образ Docker**: Укажите ссылку на Docker образ и тег сервиса cas-stream-analyzer.
- **Переменные окружения**: Задайте параметры подключения к базам данных и брокеру сообщений:
  - MONGO_URL: Адрес и учетные данные для MongoDB (например, mongodb://ПОЛЬЗОВАТЕЛЬ:ПАРОЛЬ@АДРЕС_MONGODB/).
  - KAFKA_BROKERS: Адрес брокера Kafka (например, АДРЕС_KAFKA:9092).
- **Ресурсы**: определите лимиты и запросы по памяти (например, 512Mi лимит).
- **Порты**: Укажите внутренний порт контейнера (например, 21800) и внешний порт NodePort (например, 31800).

2.1.4. Настройка cas-snmp-worker

Назначение: SNMP-мониторинг оборудования.

Необходимые параметры конфигурации:

- **Образ Docker**: Укажите ссылку на Docker образ и тег сервиса cas-snmp-worker.
- **Переменные окружения**: Задайте параметры подключения к базам данных:
  - MONGO_URL: Адрес и учетные данные для MongoDB (например, mongodb://ПОЛЬЗОВАТЕЛЬ:ПАРОЛЬ@АДРЕС_MONGODB/).
  - REDIS_URL: Адрес Redis (например, redis://@АДРЕС_REDIS).
- **Ресурсы**: Определите лимиты и запросы по памяти (например, 256Mi лимит).
- **Порты**: Укажите внутренний порт контейнера (например, 21900) и внешний порт NodePort (например, 31900).

2.1.5.

2.2. Компонент “Billing”

Модули биллинга интегрированы в cas-mdb (см. раздел 2.1.1) и включают дополнительный микросервис cas-crm.

Необходимые параметры конфигурации для cas-crm:

- **Образ Docker**: Укажите ссылку на Docker образ и тег сервиса cas-crm.
- **Переменные окружения**: Задайте параметры подключения к базам данных и брокеру сообщений:
  - POSTGRES_URL: Адрес и учетные данные для PostgreSQL (например, postgresql+asyncpg://ЛОГИН:ПАРОЛЬ@АДРЕС_POSTGRESQL/cas_crm).
  - KAFKA_BROKERS: Адрес брокера Kafka (например, АДРЕС_KAFKA:9092).
- **Ресурсы**: определите лимиты и запросы по памяти (например, 512Mi лимит).
- **Порты**: Укажите внутренний порт контейнера (например, 21750) и внешний порт NodePort (например, 31750).

2.3. Компонент “EMM” (Модуль генерации EMM)

Необходимые параметры конфигурации:

- **Образ Docker**: Укажите ссылку на Docker образ и тег сервиса cas-emm.
- **Переменные окружения**: Задайте параметры подключения к базам данных и брокеру сообщений:
  - MONGO_URL: Адрес и учетные данные для MongoDB (например, mongodb://ПОЛЬЗОВАТЕЛЬ:ПАРОЛЬ@АДРЕС_MONGODB/).
  - KAFKA_BROKERS: Адрес брокера Kafka (например, АДРЕС_KAFKA:9092).
- **Ресурсы**: определите лимиты и запросы по памяти (например, 512Mi лимит).
- **Порты**: Укажите внутренний порт контейнера (например, 21400) и внешний порт NodePort (например, 31400).

2.4. Компонент “Info” (Информационный модуль)

2.4.1. Настройка cas-reports

Необходимые параметры конфигурации:

- **Образ Docker**: Укажите ссылку на Docker образ и тег сервиса cas-reports.
- **Переменные окружения**: Задайте параметры подключения к базам данных:
  - POSTGRES_URL: Адрес и учетные данные для PostgreSQL (например, postgresql+asyncpg://ЛОГИН:ПАРОЛЬ@АДРЕС_POSTGRESQL/cas_reports).
  - REDIS_URL: Адрес Redis (например, redis://@АДРЕС_REDIS).
  - MONGO_URL: Адрес и учетные данные для MongoDB (например, mongodb://ПОЛЬЗОВАТЕЛЬ:ПАРОЛЬ@АДРЕС_MONGODB/).
- **Ресурсы**: определите лимиты и запросы по памяти (например, 512Mi лимит).
- **Порты**: Укажите внутренний порт контейнера (например, 21200) и внешний порт NodePort (например, 31200).

2.4.2. Настройка cas-email

Необходимые параметры конфигурации:

- **Образ Docker**: Укажите ссылку на Docker образ и тег сервиса cas-email.
- **Переменные окружения**: Задайте параметры подключения к Redis и SMTP-серверу:
  - REDIS_URL: Адрес Redis (например, redis://@АДРЕС_REDIS).
  - SMTP_SERVER: Адрес SMTP-сервера.
  - SMTP_PORT: Порт SMTP-сервера (например, 587).
  - SMTP_USERNAME: Логин для SMTP-аутентификации.
  - SMTP_PASSWORD: Пароль для SMTP-аутентификации.

- **Ресурсы**: определите лимиты и запросы по памяти (например, 256Mi лимит).
- **Порты**: Укажите внутренний порт контейнера (например, 21450) и внешний порт NodePort (например, 31450).

2.5. Компоненты “SMS” и “DRM”

Примечание: Компоненты SMS и DRM реализуются через модули микросервиса cas-mdb (см. раздел 2.1.1), поэтому отдельная настройка конфигурационных параметров для них не требуется.

2.6. Компонент “BD” (База данных)

Примечание: Компонент BD представляет собой архитектурный слой, реализованный через независимые СУБД каждого микросервиса. Настройка осуществляется через переменные окружения в конфигурационных параметрах соответствующих компонентов.

2.7. Компонент Encryptor

Основные настройки программного компонента Encryptor производятся с помощью конфигурационного файла encryptor.conf. Этот файл отвечает за все аспекты передачи транспортного потока, включая выбор типа потока (MPEGTS, T2MI), определение битрейта, частичное кодирование, а также источники приема и отдачи потока.

Для выборочного кодирования определенных PID каналов, на сервере кодирования необходимо создать отдельный JSON-файл, в который будут вноситься PID каналы, которые необходимо передавать в открытом виде. Для изменения списков открытых каналов в режиме реального времени можно использовать команду curl или панель администратора в веб-интерфейсе

Необходимые параметры конфигурации в encryptor.conf:

- src-addr: Адрес источника потока.
- dest-addr: Адрес назначения потока.
- emm-addr: Адрес для EMM.
- side-data-addr: Адрес для дополнительных данных.
- operation-mode: Режим работы (например, partial-mpegts).
- enable-ts-filter: Включение фильтрации транспортного потока.
- receive-pre-buffer: Размер буфера приема.
- ts-packet-size: Размер пакета транспортного потока.
- send-buffer-size: Размер буфера отправки.
- output-bitrate: Выходной битрейт.
- payload-encryption: Тип шифрования полезной нагрузки (например, gost-256-ctr).
- private-keys: Массив объектов, содержащих target, file (путь к файлу приватного ключа) и password (пароль к приватному ключу).
- pcr-mode: Режим работы PCR (например, replace).
- hash-block-size: Размер блока хеширования.
- rtp-output: Включение RTP-выхода.
- http-addr: HTTP-адрес для управления.
- unenc-pids: Путь к файлу JSON со списком незашифрованных PID.
- enabled: Включение компонента.

2.8. Компонент Decryptor

Основные настройки программного компонента Decryptor производятся с помощью конфигурационного файла decryptor.conf. Этот файл отвечает за все аспекты приема скремблируемого транспортного потока, включая выбор типа потока (MPEGTS, T2MI), определение битрейта и ключей, необходимых для дескремблирования транспортного потока.

Необходимые параметры конфигурации в decryptor.conf:

- src-addr: Адрес источника потока.
- dest-addr: Адрес назначения потока.
- operation-mode: Режим работы (например, partial-mpegts).
- receive-pre-buffer: Размер буфера приема.
- send-buffer-size: Размер буфера отправки.
- output-bitrate: Выходной битрейт.
- pcr-mode: Режим работы PCR (например, replace).
- public-key-file: Путь к файлу публичного ключа.
- rtp-input: Включение RTP-входа.
- decrypt-all: Включение полного декодирования.
- device-id: Идентификатор устройства.
- http-addr: HTTP-адрес для управления.
- enabled: Включение компонента.

2.9. Компонент прошивка CAM-модуля

Для корректной работы CAM-модуля используется адаптированный компонент Decryptor. Версия прошивки и ее настройка зависит от используемой аппаратной части самого CAM-модуля. Настройка и разрешение на дескремблирование транспортного потока, а также на открытие каналов в тарифном плане производится с помощью панели администратора, добавив соответствующее ID CAM-модуля в систему, и его дальнейшей привязки к конкретному пользователю, тарифу или зоне применения.

2.10. Компонент PCI-e modulator

Для корректной настройки и использования PCI-e modulator необходимо установить соответствующие драйвера для используемой pci платы. После проверки корректности установки драйверов платы необходимо настроить соответствующую программное обеспечение для приема дескремблируемого потока от Decryptor. В конфигурационном файле используется настройка источника и модулятора:

- sources: исходящий от Decryptor мультикаст адрес (возможно использовать 4 адреса)
- modulators: настройка физического интерфейса для подключения ТВ оборудования (возможно использования 4 частот). Пример:
    - {
    - "device": 0,
    - "channel": 0,
    - "modulation": "QAM256",
    - "symbol-rate": 7200000,
    - "frequency": 474000000,
    - "input-bitrate": 60,
    - "gain": 60

- }

3. Проверка и Валидация Настроек

3.1. Валидация Конфигурационных файлов

Для каждого конфигурационного файла, созданного для микросервисов, выполните dry-run проверку для валидации синтаксиса и структуры (если применимо для формата):

kubectl apply -f <filename>.yml --dry-run=client # Пример для Kubernetes YAML

Ожидаемый результат: deployment.apps/<service-name> configured (dry run) (для Kubernetes YAML)

3.2. Применение Конфигураций

После успешной валидации примените конфигурации и проверьте статус развернутых подов:

- Применение конфигурации:
  kubectl apply -f <filename>.yml # Пример для Kubernetes YAML
- Проверка статуса подов:
  kubectl get pods -l app=<service-name>
- Проверка логов для диагностики возможных проблем:
  kubectl logs deployment/<service-name>

3.3. Финальная Проверка

После развертывания всех компонентов выполните финальную проверку системы:

- Убедитесь, что все поды находятся в состоянии Running.
- Проверьте доступность сервисов через NodePort.