ЦИФРОВАЯ
ИНДУСТРИАЛЬНАЯ
ПЛАТФОРМА

Руководство для системы развертывания (zifctl) (2.22.0)

Zyfra Industrial Internet of Things Platform (ZIIoT)

Изменения в документе

Версия	Дата	Автор	Описание
1.0	14.08.2024	Пеплин Ф.Н.	Создание документа
1.1	24.10.2024	Маркова А.В.	Добавление раздела 4.11 Отладочные команды zifctl debug
1.2	12.11.2024	Кот О.В.	Редактирование раздела 4.11
1.3	20.11.2024	Кот О.В.	Добавлена инструкция по переходу на Cassandra 4
1.4	03.03.2025	Хадеев И.Н.	Обновление даты в нижнем колонтитуле Добавлено: раздел 3.5
1.5	05.03.2025	Хадеев И.Н.	
1.6	12.03.2025	Хадеев И.Н.	Добавлено: Приложение 2. Список внешних URL Платформы/ZIIoT
1.7	17.03.2025	Хадеев И.Н.	Добавлено: раздел 3.6 и 3.6.1
1.8	24.03.2025	Хадеев И.Н.	Редактирование: раздел 5.3.2
1.9	24.03.2025	Хадеев И.Н.	Добавлено: раздел 4.12.3.3, Приложение 3. Список учетных записей Платформы ZIIoT
1.10	07.03.2025	Хадеев И.Н.	Добавлено: разделы 4.15, 4.15.1, 4.15.2
1.11	15.05.2025	Хадеев И.Н.	Добавлено: раздел 4.12 Раздел 4.12.3.3 перенесен в раздел 4.12
1.12	19.05.2025	Хадеев И.Н.	Добавлено: разделы 4.10, 4.10.1, 4.10.2, 4.10.3, 4.10.4, 4.10.5, 4.10.6, 4.10.7, 4.10.8, 4.10.9, 4.10.10, 4.10.11, 4.10.12, 4.10.13, 4.10.14
1.13	20.05.2025	Хадеев И.Н.	Добавлено: разделы 14, 15, 16
1.14	21.05.2025	Хадеев И.Н.	Добавлено: разделы 3.7, 3.7.1, 3.7.2, 3.7.3, 3.7.4, 3.7.5, 3.7.6
1.15	22.05.2025	Хадеев И.Н.	Добавлено: разделы 17, 4.11
1.16	26.05.2025	Хадеев И.Н.	Редактирование: раздел 4.12
1.17	30.05.2025	Хадеев И.Н.	Добавлено: разделы 18, 18.1, 18.2, 18.3, Рисунок 18.1, Таблица 18.1, Таблица 18.2, Таблица 18.3, Таблица 18.4, Таблица 18.5, Таблица 18.6, Таблица 18.7, Таблица 18.8
1.18	24.06.2025	Хадеев И.Н.	Перенос: разделов 4.12, 4.13, 4.14, 4.15 в 4; разделов 4.13.1.1, 4.13.1.2, 4.13.1.3, 4.13.1.4, 4.13.1.5 в 4.13

Содержание

1 Требования к рабочей станции, с которой будет производиться развертывание, и ее окружению

Таблица 1.1. Требования к оборудованию/рабочей станции

	Минимальные	Рекомендуемые
ЦПУ	2	4
ОЗУ	4 Гб	8Гб

Требования к окружению:

- ОС семейства **Linux** или **wsl2** (для ОС **Windows**).
- **Docker**.
- **Python** 3.10.

Для развертывания рекомендуется использовать скрипт-обертку (**wrapper script**).

Запуск системы автоматического развертывания платформы (**ZIIoT Deployment system** или **zifctl**) можно осуществить двумя способами:

- Выполнение развертывания с помощью **wrapper**, запуск **zifctl** интерактивно (рекомендуемый способ запуска).
- Запуск **zifctl** из контейнера **docker**.

2 Версионирование компонентов Платформы для релиза 2.22.0

2.1 Совместимость системы развертывания Платформы с Kubernetes

С помощью инструментов авторазвертывания **Платформа** может быть установлена только на системы оркестрации контейнеров, а именно на **Kubernetes** или производные от него дистрибутивы **ReD Hat Openshift Container Platform** (**OpenShift**/**OKD**).

Таблица 2.1 Совместимость системы развертывания ZIIoT 2.22.0 с Kubernetes

Дистрибутив	Назначение	Версия
Kubernetes	Системы оркестрации контейнеров, совместимые со стандартами CNCF.	1.21 и выше
OCP (OpenShift/OKD)		4.8 и выше
Deckhouse		1.52

Системы оркестрации контейнеров должны содержать некоторые компоненты, без которых **Платформа** не может быть установлена. Перечисленные компоненты устанавливаются, как правило, в рамках всего кластера и поэтому не могут быть включены в состав **Платформы**.

Таблица 2.2. Версии инфраструктурных компонентов в ZIIoT 2.22.0

Компонент	Обязательность	Версия
StorageClass	+	Сущность для хранения параметров подключения к системе хранения данных.
Container Network Interface (CNI)	+	Контроллер внутренней сети кластера. Входит в состав любого дистрибутива Kubernetes. Должен поддерживать работу ~150 IP сервисов, входящих в состав Платформы. Конкретная реализация (OpenShift SDN, Flannel, Canico и т.д.) не регламентируется.
external LoadBalancer	+	Общий сетевой LoadBalancer для организации сетевого взаимодействия конечных пользователей сервисов, развернутых с помощью оркестратора Kubernetes. Входит в состав любого дистрибутива Kubernetes и должен быть корректно сконфигурирован и доступен. Использование иных вариантов балансировки входящего трафика (например, установка внешнего Nginx на ноды кластера или использование NodePort) **несовместимо** с системой развертывания Платформы.
Ingress Controller	+	Контроллер входящего трафика для Kubernetes LoadBalancer. Для Kubernetes - только базовый ingress-nginx, для OCP это стандартный Route Controller, входящий в состав дистрибутива OCP. Для корректного заполнения IP адресов в сообщениях аудита, необходимо включить use-forwarded-headers.
cert-manager	-	Необязательный компонент Kubernetes, используется для генерации TLS-сертификатов: cert-manager.io. В типовом случае используется для генерации TLS-сертификатов для Ingress. Платформа может быть установлена и без него, в этом случае необходимо сгенерировать и указать в параметрах конфигурации (env-values.yaml) TLS-сертификаты для объектов Ingress вручную.

cert-manager.io/v1 Issuer или Cluster Issuer	-	Certificate Issuer для cert-manager - управляет TLS-сертификатами, в том числе Ingress Controller. Как и cert-manager, не является обязательным компонентом.
monitoring.coreos.com/v1	-	Необязательный компонент Kubernetes, используется для создания объектов ServiceMonitor, который автоматически генерирует конфигурацию Prometheus scrape на основе текущего состояния объектов на сервере API. Требуется версия prometheus-operator >0.61.0

2.2 Совместимость Платформы с инфраструктурными компонентами

Данные компоненты могут либо устанавливаться вместе с Платформой в **Kubernetes/OCP**, либо использоваться в качестве внешних зависимостей, установленных в другом месте.

В связи с этим, в таблице совместимости перечислены только требования к версиям этих компонентов, без указания конкретной реализации.

Примечание. Совместимость с иными реализациями **S3** (**Ceph**, ...) не протестирована. Точно известно, что с отличными от **MinIO** хранилищами несовместима автоматика развертывания (инициализация **bucket**, **access/secret key** и **policy**).

Таблица 2.3. Версии инфраструктурных компонентов в ZIIoT 2.22.0

Компонент	Назначение	Версия
PostgreSQL	рСУБД (Реляционная система управления базами данных). **Примечание**. Для работы сервисов Платформы необходима установка следующих extensions для PostgreSQL, не входящих в базовый пакет: ▪ tablefunc ▪ uuid-ossp ▪ pg_trgm ▪ btree_gist ▪ pgcrypto Для большинства операционных систем для установки этих extensions достаточно установить пакет postgresql-contrib, в зависимостях у этого пакета - библиотеки от Perl. postgresql-contrib входит в состав базового образа PostgreSQL при его установке в Kubernetes с помощью Stolon. в случае использования внешнего PostgreSQL в закрытых контурах необходимо убедиться, что там всё это установлено	12, 14
Apache Cassandra	БД временных рядов	3.11, 4.1
Redis	key-value БД (кэш)	6
Apache Kafka	Брокер сообщений	3.7
RabbitMQ	Брокер сообщений (используется в сервисе UDL).	3.12
Apache NiFi	Сервис управления потоками данных	1.19.1
KeyCloak	Сервис авторизации	17, 21
S3	API объектного хранилища	MiniO 2024.7.31

2.3 Версии инфраструктурных и вспомогательных сервисов в составе инструмента по авторазвертыванию (zifctl)

Релиз платформы версии 2.22.0 поставляется с инструментом по автоматическому развертыванию Платформы.

Версия инструмента (**zifctl**) аналогична версии платформы — 2.22.0. С помощью инструмента автоматического развертывания развертываются сервисы платформы, вошедшие в релиз 2.22.0, а также инфраструктурные и вспомогательные сервисы из таблицы ниже:

Таблица 2.4. Версии вспомогательных сервисов (zifctl) в ZIIoT 2.22.0

Компонент	Реализация	Назначение	Версия	Образ	Лицензия
PostgreSQL	Zyfra	PostgreSQL + утилиты для создания отказоустойчивого кластера	12.19	zif-stolon-pg12:1.2.0-v6-240801	The PostgreSQL Licence, Apache License 2.0
			14.12	zif-stolon-pg14:1.2.0-v6-240801	
Postgresql exporter	Bitnami + Zyfra	Мониторинг PostgreSQL	0.15.0	0.15.0-debian-12-r38	Apache License 2.0
pgAdmin4	pgAdmin	Графический клиент для PostgreSQL	8.10	zif-pgadmin4:8.10.0-v1-240801	The PostgreSQL Licence
Apache Cassandra	Bitnami + Zyfra	БД временных рядов	3.11.13	zif-cassandra:3.11.13-v22-240801	Apache License 2.0
			4.1.5	zif-cassandra4:4.1.5-v1-240801	
Apache Cassandra Exporter	Bitnami + Zyfra	мониторинг Cassandra	2.3.8	zif-cassandra-exporter:2.3.8-v21-240801	Apache License 2.0
Redis	Bitnami + Zyfra	key-value БД (кэш)	6.2.14	zif-redis:6.2.14-v11-240801	Apache License 2.0
Redis Exporter	Bitnami + Zyfra	Мониторинг Redis	1.62.0	zif-redis-exporter:1.62.0-v1-240801	Apache License 2.0
Apache Kafka	Confluent + Zyfra	Брокер сообщений	7.7.0	zif-kafka:7.7.0-v1-240801	Apache License 2.0
Kafka Exporter	Bitnami + Zyfra	Мониторинг Apache Kafka	1.7.0	zif-kafka-exporter:1.7.0-v18-240801	Apache License 2.0
Apache Zookeeper	Confluent + Zyfra	Распределенное хранение конфигураций Apache Kafka и Apache NiFi	7.7.0	zif-zookeeper:7.7.0-v1-240801	Apache License 2.0
Kafka Rest	Confluent + Zyfra	REST Api для Apache Kafka	7.7.0	zif-kafka-rest:7.7.0-v1-240801	Apache License 2.0

Kafka Schema Registry	Confluent + Zyfra	Схемы сообщений для топиков Apache Kafka	7.7.0	zif-schema-registry:7.7.0-v1-240801	Apache License 2.0
Kafka Connect	Confluent + Zyfra	Утилиты для интеграции различных источников данных с Apache Kafka. В Zyfra добавлена утилита jq и несколько коннекторов (например, Debezium)	7.7.0	zif-kafka-connect:7.7.0-v1-240801	Apache License 2.0
Debezium Connector	Red Hat	Сбор, сериализация и отправка в Apache Kafka логов транзакций PostgreSQL	2.5.3	см. Kafka Connect	Apache License 2.0
JDBC Connector	Confluent	Сбор, сериализация и отправка данных в Apache Kafka (как Source) и РСУБД (как Sink)	10.7.6	см. Kafka Connect	Apache License 2.0
JMX Exporter	Bitnami + Zyfra	Экспорт метрик JVM для сервисов экосистемы Kafka. В Zyfra на релиз по хэш-коммиту выставлен тег Stable	0.20.0	zif-jmx-exporter:0.20.0-v11-240802	Apache License 2.0
Kafka UI	Provectus Labs + Zyfra	UI для Apache Kafka, Kafka Connect, Kafka Schema Registry	0.7.2	zif-kafka-ui:0.7.2-v3-240801	Apache License 2.0
RabbitMQ	Bitnami + Zyfra	Брокер сообщений	3.12.14	zif-rabbitmq:3.12.14-v2-240801	Apache License 2.0
Apache NiFi	Apache Foundation + Zyfra	Сервис управления потоками данных. В Zyfra добавлен ряд процессоров для интеграции со сторонними источниками данных	1.19.1	zif-nifi:1.19.1-v35-240801	Apache License 2.0
KeyCloak	RedHat + Zyfra	Сервис авторизации. stolon В Zyfra добавлен корпоративный дизайн фронтенда и несколько плагинов	17.0.1	zif-keycloak:17.0.1-v32-240802	Apache License 2.0
			21.1.2	zif-keycloak:21.1.2-v8-240801	Apache License 2.0
KeyCloak RabbitMQ Event Listener	community	Экспорт KeyCloak Events в топики RabbitMQ	3.0.2	см. KeyCloak	Apache License 2.0
KeyCloak Metrics SPI	community	Экспорт метрик KeyCloak	2.5.3	см. KeyCloak	Apache License 2.0
OAuth2 Proxy	community	Прокси для авторизации в KeyCloak для Kafka UI, Apache NiFi	7.6.0	zif-oauth2-proxy:7.6.0-v1-240801	MIT
Alpine	Docker Inc + Zyfra	Набор утилит	3.20.2	zif-alpine-util:3.20.2-v1-240801	GPLv2

MinIO	Bitnami + Zyfra	S3-совместимое объектное хранилище	2024.7.31	zif-minio:2024.7.31-v1-240801	Apache License 2.0
Opensearch	community + Zyfra	Масштабируемая утилита полнотекстового поиска и аналитики	1.3.18	zif-opensearch:1.3.18-v1-240801	Apache License 2.0
Fluentd	Bitnami + Zyfra	Система сбора, парсинга, перенаправления и агрегации логов	1.14.6	zif-fluentd-opensearch:1.14.6-v22-240801	Apache License 2.0
Opensearch Dashboards	community + Zyfra	UI для Opensearch	1.3.18	zif-opensearch-dashboards:1.3.18-v1-240801	Apache License 2.0

3 Специальные требования к процессу обновления

3.1 Переход на Cassandra 4

Начиная с версии платформы 2.20, по умолчанию будет устанавливаться 4 версия cassandra. При обновлении предусмотрена автоматическая миграция данных. Новая версия Cassandra значительно улучшает производительность и стабильность базы данных, а также предоставляет доступ к расширенным функциональным возможностям.

ВАЖНО! После перехода на версию 4 откат на версию 3 невозможен из-за изменений в структуре данных и механизмах работы самой базы данных. Это связано с тем, что Cassandra 4 использует новые внутренние форматы и алгоритмы, которые не совместимы с предыдущей версией.

3.1.1 Рекомендации

- Перед обновлением настоятельно рекомендуем **создать резервную копию** всех данных для предотвращения потерь в случае необходимости отката.
- Проверьте и протестируйте все критически важные функции приложения на версии 4 Cassandra перед развертыванием в продуктивной среде.

3.1.2 Настройки перехода

Переход осуществляется автоматически, однако, при необходимости, доступны варианты, описанные ниже.

3.1.2.1.1 Переход с увеличением количества реплик

С переходом на Cassandra 4 вы можете увеличить количество реплик в кластере, что повышает отказоустойчивость и доступность данных.

В файл **env-values.yaml** внести следующие изменения:

```
cassandra:
  installed: True
  nodes: 3
```

3.1.2.1.2 Переход с версии Cassandra без включенного TLS на версию с активированным TLS

Для перехода с Cassandra без включенного TLS на версию с активированным TLS (Transport Layer Security) необходимо внести следующие изменения в файл **env-values.yaml**:

```
cassandra:
  installed: True
  nodes: 3
  tls:
    enabled: True
```

3.1.3 Отказ от перехода

3.1.3.1 Возврат к версии 3

Возврат к предыдущей версии возможен только через восстановление из резервной копии (см. Рекомендации)

3.1.3.2 Отказ от обновления

Для отказа от перехода на 4 версию Cassandra, в конфигурационном файле **env-values.yaml** необходимо явно указать версию

```
cassandra:
  version: 3
  installed: True
```

3.2 Плагин логирования в формате CEF в Keycloak v21

В **Keycloak v21**, поставляемый в рамках авторазвертывания, добавлен плагин для логирования в формате **CEF**. Логирование в этом формате активируется при установке параметра **logging.cef**: **True** в файле конфигурации авторазвертывания **env-values.yaml**:

```
logging:
  cef: True
```

При установке данного параметра в **Keycloak** будет отдавать логи уровня **INFO** с событиями **CEF**:

```
{"timestamp":"2024-06-
19T08:14:14.35Z","sequence":8438,"loggerClassName":"org.slf4j.impl.Slf4jLogger","logge
rName":"com.zyfra.idp.eventlistenerprovider.ZIIOTEventListenerProvider","level":"INFO"
,"message":"CEF:0|ZYFRA|ZIIoT|2.22.0|Security Admin Audit Event|CREATE
CLIENT_SCOPE|5|src=127.0.0.1 dst=<hostname> shost= suid=db4ebd54-a14d-47b0-b12d-
1e8b855def7b suser=<user>
msg=realmId:<realmId>,realmName:<realmName>,clientId:69a11568-0195-4e1d-bdf7-
810eb1a54c3a,resourcePath:client-scopes/ba782c60-4a9e-488f-b64c-c0e7c3eee80b
end=1718784854347","threadName":"executor-thread-
0","threadId":16,"mdc":{"CefMessage":"CEF:0|ZYFRA|ZIIoT|2.22.0|Security Admin Audit
Event|CREATE CLIENT_SCOPE|5|src=127.0.0.1 dst=<hostname>> shost= suid=db4ebd54-a14d-
```

```
47b0-b12d-1e8b855def7b                                                    suser=<user>
msg=<realmId>:ziiot,realmName:<realmName>,clientId:69a11568-0195-4e1d-bdf7-
810eb1a54c3a,resourcePath:client-scopes/ba782c60-4a9e-488f-b64c-c0e7c3eee80b
end=1718784854347"},"ndc":"","hostName":"zif-keycloak21-
0","processName":"QuarkusEntryPoint","processId":1}
```

Внимание! Сбор логов в формате **CEF** необходимо настроить на поле **mdc.CefMessage**.

3.3 Настройки OpenTelemetry

3.3.1 Описание

OpenTelemetry – новый стандарт сбора телеметрии приложений. Он объединил стандарты OpenTracing (используется в ZIIoT) и OpenCensus.

В рамках OpenTelemetry телеметрия приложений складывается из сигналов, их состав в будущем может быть расширен, но сейчас это:

- **Метрики** - числовые измерения ключевых характеристик, сформированные во время выполнения приложения.
- **Трейс** - распределенный ациклический граф выполняемой работы, который состоит из отдельных диапазонов - Span. Span (в .NET класс System.Diagnostics.Activity) и описывает локальную единицу работы приложения и может расширяться атрибутами, событиями (лог, детализирующий, что произошло в рамках выполняемой единицы работы) и вложенными единицами работы.
- **Лог** - это текстовая запись с меткой времени, лог являются независимым источником данных телеметрии, но в рамках концепции OpenTelemetry рекомендуется сопоставлять их с контекстом выполняемой работы, т.е. лог рекомендуется прикреплять к Span в виде событий.
- **Багаж** (baggage) - описывает контекстную информацию связанную с активностью (span), которая будет передаваться во все вложенные активности

Кроме концептуального подхода, OpenTelemetry предоставляет:

- Инструментарии для сбора телеметрии – представляет собой библиотеки для сервисов под каждый технологический стек (их развитие определяет сообщество и может отличаться по возможностям, в этом документе мы рассматриваем только .NET)
- Экспортёры – библиотеки, которые позволяют экспортировать собранную телеметрия в указанный источник в формате источника (нас интересует экспорт в консоль, Jaeger, Propmetheus (pull режим через /metrics конечную точку)
- SDK позволяет:
    - расшить перечень процессоров, обрезывающих собранную телеметрия до ее представления в экспортере.
    - расширять перечень экспортеров собственными реализациями)
    - расширить возможности интеграции для распространения контекстов трассировки (context propagation) - что позволяет передавать\получать контекст из различных источников (http headers, rabbitmq message, kafka message и т.д.) и различных форматах (b3(zipkin)\uber (jaeger) header и т.д)
- Спецификации - набор рекомендаций по формированию телеметрии
- Протокол передачи телеметрии OTLP

- OpenTelemetry Collector - сервис, который позволяет в унифицированном формате собирать телеметрию с сервисов, обогащать ее и адаптировать собранную телеметрию под потребности конечных приемников телеметрии

3.3.2 Настройки

В рамках поддержки будущего перехода на сбор телеметрии приложений с использованием **OpenTelemetry**, в модель конфигурации авторазвертывания добавлены новые параметры:

```
tracing:
  samplerRatio: 1.0
  otlpPort: 4317
```

В сервисы передаются следующие переменные окружения (при условии **jaeger.tracing_enabled** или **jaeger.enabled**):

- **OTEL_TRACING_ENABLED** - регулируется параметром **jaeger.tracing_enabled** или **jaeger.enabled**, значение по умолчанию **False**.
- **OTEL_TRACING_SAMPLER_RATIO** - регулировка сэмплирования, значение по умолчанию 1.0 (т.е. все трейсы отправляются в коллектор).
- **OTEL_OTLP_COLLECTOR_URL** - **jaeger.host:tracing.otlpPort**.

3.4 Миграция PostgreSQL (Stolon) на новую версию

3.4.1 Система миграции

Система миграции вынесена в отдельную команду **zifctl postgres** и не является частью команды **zifctl sync**, соответственно миграция не выполняется автоматически с командой **zifctl sync.**

При выполнении миграции будет создан новый **PVC** для данных нового кластера **Stolon**, куда будут скопированы данные для новой версии СУБД. В процессе работы мигратора, будут остановлены все зависимые от БД сервисы, а также **stolon-keeper**.

Для управление всеми этапами миграции используется только команда **zifctl postgres**, которая содержит список субкоманд:

```
$ zifctl postgres –help
usage: zifctl postgres [SUB-COMMAND] [OPTIONS] [--help]
Migrating PostgreSQL servers to new versions running inside Kubernetes cluster
required arguments:
  Specify this options or declare corresponding environment variables
common options:
  -v, --verbose       Enable verbose command output
  -vv, --debug        Enable verbose command output and helmfile/helm debug
```

```
                      output
  -h, --help          Show help message and exit
available subcommands (and aliases):
  Sub-command         Sub-command description
    migration-cancel   Stop running migration
    migration-clean    Clean all migration objects, include PVCs
    migration-get-logs   Save logs to disk
    migration-start   Upgrade PostgreSQL databases to a new version
```

3.4.1.1 Подготовка к миграции

Миграция состоит из нескольких этапов:

1. Проверить параметр **postgres.version** в конфигурации окружения. В нем должна быть указана текущая версия **PostgreSQL**. Также удостовериться в достаточном размере тома для данных **PostgreSQL**, т.к. новый том будет создан с тем же размером, который указан в конфигурации окружения.

```
postgres:
  storageSizeMB:        <указать        необходимое        значение        в        МБ>
  version: stolon-12
```

2. Удостовериться, что все сервисы в текущем тенанте успешно запущены и работают в штатном режиме:

```
# можно проверить командой
 $ kubecl get pods --field-selector "status.phase!=Succeeded" | awk 'match($0,"([0-9]+)/([0-9]+)|STATUS",r) {if (r[1]!=r[2] || $3 ~ /Terminating|STATUS/) print}'
```

3. Запуск миграции командой **zifctl postgres migration-start**. Также рекомендуется использовать отдельный **PVC** для сохранения логов миграции (в случае длительной миграции размер логов, выводимых в **stdout**, может превысить лимит, что приведет их ротации и потере части логов).

   Чтобы сохранять логи в персистентном хранилище (по умолчанию не используется) можно указать аргумент **--logs-pvc-size <size>,** где **size** может быть указан в мегабайтах или гигабайтах, например: **500Mi** или **1Gi**. Данные на диске хранятся в не архивированном виде.

3.4.1.2 Миграция PostgreSQL на версию 14

```
# single-tenant
```

```
$ zifctl postgres migration-start --env-path /path-to-config/env-config --postgres-version stolon-14

# multi-tenant

# первым запускаем tenant

$ zifctl postgres migration-start --env-path /path-to-tenant-config/env-config --postgres-version stolon-14

# вторым запускаем shared-infra

$ zifctl postgres migration-start --env-path /path-to-infra-config/env-config --postgres-version stolon-14
```

1. Будут остановлены все сервисы платформы, использующие БД, включая **Keycloak**, **Pgadmin** и сам **Stolon-Keeper** (**PostgreSQL**).
2. Создание PVC для данных нового экземпляра **PostgreSQL**. Создается только один PVC, даже если в конфигурации окружения указано **infraHA: true**. Создание оставшихся PVC будет выполнено в рамках деплоя нового чарта **Stolon2**.
3. Запуск **Kubernetes Job** в том же namespace, где запущен **Stolon-keeper**. К поду подключаются PVC со старыми и новыми данными, а также **PVC** для логов, если задан аргумент **--logs-pvc-size**. После запуска **job**, можно прервать выполнение команды, сама **job** продолжит выполняться в фоновом режиме. Подключиться к просмотру логов текущей **job** возможно, запустив повторно команду **zifctl postgres migration-start ...**. Остановка миграции возможна только субкомандой **migration-cancel**.

В процессе миграции будут возникать ошибки. Данные ошибки можно игнорировать:

```
```log
2023-05-22 14:39:01,697 INFO kube: ERROR: type "opaque" does not exist
2023-05-22 14:39:02,361 INFO kube: ERROR: function pg_catalog.pg_create_logical_replication_slot(name, name, boolean) does not exist
2023-05-22 14:39:03,225 INFO kube: ERROR: function pg_catalog.pg_terminate_backend(integer) does not exist
2023-05-22 14:39:03,720 INFO kube: ERROR: function pg_catalog.set_bit(bytea, integer, integer) does not exist
...
2023-05-22 14:39:04,456 INFO kube: ERROR: function pg_catalog.utf8_to_johab(integer, integer, cstring, internal, integer) does not exist
2023-05-22 14:39:04,457 INFO kube: ERROR: function pg_catalog.utf8_to_koi8r(integer, integer, cstring, internal, integer) does not exist
2023-05-22 14:39:04,458 INFO kube: ERROR: function pg_catalog.utf8_to_koi8u(integer, integer, cstring, internal, integer) does not exist
2023-05-22 14:39:04,459 INFO kube: ERROR: function pg_catalog.utf8_to_shift_jis_2004(integer, integer, cstring, internal, integer) does not exist
2023-05-22 14:39:04,460 INFO kube: ERROR: function pg_catalog.utf8_to_sjis(integer, integer, cstring, internal, integer) does not exist
2023-05-22 14:39:04,462 INFO kube: ERROR: function pg_catalog.utf8_to_uhc(integer,
```
```

```
integer,       cstring,       internal,       integer)       does       not       exist
2023-05-22 14:39:04,464 INFO    kube: ERROR:  function pg_catalog.utf8_to_win(integer,
integer,       cstring,       internal,       integer)       does       not       exist
```

4. После успешного завершения миграции, в последней строке логов будет соответствующее сообщение **postgres: Job return status: Succeeded**, необходимо в конфигурации окружения заменить параметр **postgres.version** на **stolon-14**.

5. Выполнить команду **zifctl sync**.

Внимание! В случае мультитенантности синхронизация должна быть выполнена в определенном порядке: первой выполняется синхронизация **shared-infra**, вторым **tenant**.

Во время синхронизации будут созданы недостающие PVC, если в конфигурации окружения указано **infraHA**: **true**, и будет запущена репликация данных на новых инстансах **stolon-keeper**.

3.4.1.3 Получение логов миграции

В случае, если процесс миграции завершился со статусом **Failed** и для решения проблемы потребуются логи, то можно получить логи следующими способами:

```
$ kubectl logs -l job-name=pg-upgrade  pg-upgrade.log
```

Если при старте миграции был указан аргумент **--logs-pvc-size**, тогда возможно получить файл с логами командой:

```
$ zifctl postgres migration-get-logs --env-path /path-to-config/env-config --output-
path . /var/deploy/output/pg-upgrade.log
```

3.4.1.4 Остановка миграции

Команда останавливает текущую миграцию. Повторный запуск миграции, после ее остановки, приведет к запуску миграции с самого начала. Данные нового **PostgreSQl** будут очищены.

```
$ zifctl postgres migration-cancel --env-path /path-to-config/env-config
```

3.4.1.5 Очистка данных после не успешной миграции

Удалить данные (**job**, **PVC** с логами), созданные при запуске миграции, можно командой:

```
$ zifctl postgres migration-clean --env-path /path-to-config/env-config
```

Если указать аргумент **--force**, также будет удален **PVC** с данными нового **PostgreSQL**, созданный в результате миграции.

3.5 Apache Cassandra в кластерном режиме, рекомендации и описание Replication Factor, связи кластерного развертывания, порядок снятия метрик и функциях zif-cassandra-exporter

3.5.1 Apache Cassandra в кластерном режиме

Для настройки Apache Cassandra в высоко доступном (HA) кластерном режиме необходимо в **env-values.yaml** включение режима HA - **infraHA: true.**

3.5.2 ReplicationFactor

Коэффициент репликации (Replication factor) - регулирует избыточность данных в системе. Это общее количество копий строки по всему кластеру. Коэффициент репликации:

- равный 1 - означает, что на одном узле имеется только одна копия каждой строки;
- равный 2 - означает, что имеется две копии каждой строки, но эти копии находятся на разных узлах.

Коэффициент репликации настраивается командой:

```
cassandra:
  replicationFactor: 3
```

Особенности настройки replicationFactor:

- по умолчанию у нас replicationFactor: 1 и он скрыт в env-values по дефолту (кто его не знает, тот не выставляет);
- replicationFactor > 1 мы можем выставить только на новый кластер. При увеличении количества реплик и т.д. replicationFactor изменить уже нельзя;
- cassandra.replicationFactor нельзя ставить меньше количества реплик (cassandra.nodes). Иначе он даст ошибку.

3.5.3 QUORUM

QUORUM - представляет собой уровень согласованности, при котором для успешного выполнения операции требуется подтверждение от большинства реплик.

В сервисе zif-rtdb-data CASSANDRA_READ_CONSISTENCY_LEVEL и CASSANDRA_WRITE_CONSISTENCY_LEVEL устанавливается QUORUM.

3.5.4 Cassandra: мониторинг, логирование

Для мониторинга работы кластера Cassandra, который устанавливается в среде контейнеризации, используется экспортер Cassandra Exporter отдающий метрики в формате Prometheus. Пример доступных метрик можно посмотреть по ссылке https://github.com/instaclustr/cassandra-exporter/wiki/Exported-Metrics.

Для мониторинга работы кластера Cassandra устанавливаемого в среде виртуализации используется **prometheus-jmx-exporter** отдающий метрики в формате Prometheus. Данный exporter разворачивается с помощью Ansible role устанавливающей Cassandra. Графическое отображение получаемых метрик можно настроить на dashboard Grafana.

Логирование работы Cassandra при развертывании в среде контейнеризации осуществляется в стандартный вывод контейнера, настроек не имеет. При использовании выделенной Cassandra логирование настраивается через собственный конфигурационный файл Cassandra – **logback.xml**. Также если для развертывания внешней Cassandra в среде виртуализации используются Ansible roles, то можно включить отдельно **debug.log** выставив **cassandra_debug_log_enabled: true**.

3.6 Передача строки подключения SQL_SERVERS в сервис zif-om-sqldatasource в зашифрованном виде

Примечание: при выполнении команд **zifctl secrets enc/dec** необходимо указывать обязательные опции **--env-path и --age-key**

1. При установке с нуля, после команды **zifctl init** необходимо:

a. Создать файл секретов модуля по пути **<путь до папки с конфигурацией окружения>/values/om/mod-secrets.dec.yaml** следующего содержания:

mod-secrets.dec.yaml

```
secrets:
  zif-om-sqldatasource-sql-servers:  <строка подключения>
```

Например:

mod-secrets.dec.yaml

```
secrets:
  zif-om-sqldatasource-sql-servers:  MyPostgres='Postgresql:Server=123.45.678.90;Port=5432;D
atabase=postgres;Username=postgres;Password=postgres';
```

b. Зашифровать созданный файл, используя команду **zifctl secrets enc**:

```
./zifctl secrets enc --module om
```

Результат: после выполнения команды будет создан зашифрованный файл **mod-secrets.yaml** в той же папке. После этого **mod-secrets.dec.yaml** необходимо удалить.

2. При переустановке на уже развернутый экземпляр необходимо:

a. Расшифровать файл секретов модуля, используя команду **zifctl secrets dec**:

```
./zifctl secrets dec --module om
```

b. Добавить требуемые значения (как описано в секции пункта 1.a);
c. Зашифровать файл секретов модуля, используя команду **zifctl secrets enc:**

```
./zifctl secrets enc --module om
```

3. Удалить файл **mod-secrets.dec.yam**l.

4. Создать файл конфигурации сервиса по пути **<путь до папки с конфигурацией окружения>/values/om/zif-om-sqldatasource.yaml.gotmpl** следующего содержания:

zif-om-sqldatasource.yaml.gotmpl

```
environmentVars:
  SQL_SERVERS:
```

```
valueFrom:
  secretKeyRef:
    name: zif-om-module-secrets
    key: zif-om-sqldatasource-sql-servers
```

5. Выполнить развертывание Платформы командой **zifctl sync.**
6. После развертывания обновленный файл **<путь до папки с конфигурацией окружения>/values/om/mod-secrets.yaml** необходимо сохранить в системе версионирования.

3.6.1 Описание формата строки подключения

Примечание: общая информация о строке подключения приведена в **README** сервиса **zif-om-sqldatasource.**

```
{Alias name}='{Data source type}:{Connection string}';{Alias name 1}='{Data source type}:{Connection string}';...
```

1. Валидными значениями **Data source type** являются:
   - Postgresql;
   - SqlServer;
   - MySql;
   - Oracle;
   - Cassandra;
   - Hive;
   - Phoenix.

Примечание: при наличии некорректного Data source type в логах сервиса будет отображаться ошибка ***Connection string contains unsupported data source type.***

2. **Alias name** - не должен повторяться в строке подключения.
3. **Data source type** и **Connection string -** должны разделяться двоеточием.
4. Точка с запятой в конце строки - является необязательной.

Примеры валидных строк подключения:

```
MyPostgres='Postgresql:Server=123.45.678.90;Port=5432;Database=postgres;Username=postgres;Password=postgres';MyPostgres1='Postgresql:Server=123.45.678.90;Port=5432;Database=postgres;Username=postgres;Password=postgres'
```

```
MyPostgres='Postgresql:Server=123.45.678.90;Port=5432;Database=postgres;Username=postgres;Password=postgres';MySqlServer='SqlServer:Server=123.45.678.90;Port=5432;Database=postgres;Username=postgres;Password=postgres'
```

Примеры некорректных строк подключения:

```
# Неподдерживаемый Data source type - SqlServer1
MyPostgres='Postgresql:Server=123.45.678.90;Port=5432;Database=postgres;Username=postgres;Password=postgres';MyPostgres1='SqlServer1:Server=123.45.678.90;Port=5432;Database=postgres;Username=postgres;Password=postgres'
```

```
# Не уникальный Alias name - MyPostgres
MyPostgres='Postgresql:Server=123.45.678.90;Port=5432;Database=postgres;Username=postgres;Password=postgres';MyPostgres='SqlServer:Server=123.45.678.90;Port=5432;Database=postgres;Username=postgres;Password=postgres'


# Data source type и Connection string разделены точкой с запятой вместо двоеточия
MyPostgres='Postgresql:Server=123.45.678.90;Port=5432;Database=postgres;Username=postgres;Password=postgres';MyPostgres='SqlServer;Server=123.45.678.90;Port=5432;Database=postgres;Username=postgres;Password=postgres'
```

3.7 Переход на PostgreSQL v17/Patroni

3.7.1 Создание резервных копий

Для начала рекомендуется масштабировать в 0 все сервисы Платформы, кроме сервисов **Stolon**. Это можно сделать как через UI при помощи кнопки **Lens**, так и командой:

```
kubectl scale ->0. Example
kubectl scale --replicas=<number_of_replicas> <resource_type>/<resource_name> <options>
```

Примечание: резервные копии можно выполнить любым доступным способом. Для этого рекомендуется использовать стандартную утилиту **pg_dump**. Ниже приведен скрипт для примера, который используется в качестве **cron_job** на стендах:

```
Dump script. Example
export PGHOST="${POSTGRES_HOST}"


export PGPORT="${POSTGRES_PORT}"


export PGDATABASE="postgres"


for db_name in $(psql -t -c "SELECT datname FROM pg_database
WHERE datname LIKE 'ziiot__%'"); do
    __pfx="backup-${db_name}-"
    FILENAME="${__pfx}$(date +%Y-%m-%d_%H%M%S).sql.gz"


    list=($(find ${PG_BACKUP_MOUNT} -type f -name "${__pfx}*" -exec ls -1tr "{}" +))
    if [ ${#list[@]} -gt 5 ]; then
        echo "$(date): Delete old backups"
```

```
        for backfile in "${list[@]::3}"; do
            echo ${backfile}
            rm -f "${backfile}"
        done
    fi

    echo "$(date): Start dump the database \"${db_name}\" to ${FILENAME}"
    pg_dump -Fc -b --no-owner "${db_name}" | gzip > ${PG_BACKUP_MOUNT}/${FILENAME}

    echo "$(date): Backup \"${FILENAME}\" successful"
    echo -e "$(du -h ${PG_BACKUP_MOUNT}/${FILENAME})\n"
done

echo -e "$(du -sh ${PG_BACKUP_MOUNT})\n"
```

Примечание: некоторые переменные берутся из секретов, в которых указаны данные для подключения к серверу **Postgres**, такие как: **POSTGRES_HOST**, **POSTGRES_PORT**, **PGPASSWORD**, **PGUSER**.

3.7.2 Подготовка env-values

Для подготовки **env-values** необходимо открыть на редактирование файл **env-values.yaml**. Далее перейти в блок **postgres** и заменить значение поля **version** со **stolon-14** на **patroni-17**. Для примера ниже приведен блок **postgres** из **env-values.yaml**:

```
Блок postgres из env-values с version patroni-17
postgres:
  installed: True
  configured: True
  host: 'zif-postgres.dev-guseynov'
  port: 5432
  storageSizeMB: 5120
  version: patroni-17
```

3.7.3 Изменения в секретах Платформы

Для того чтобы Patroni был развернут корректно, необходимо заменить имя пользователя в секретах:

1. Сначала расшифровать секреты:

```
Расшифрование секретов
./zifctl secrets decrypt --env-path ENV-PATH --age-key AGE_KEY
```

2. Открыть файл **env-secrets.dec.yaml** на редактирование и изменить в блоке **postgres** значение поля **adminUser** с **admin** на **postgres**. Пример блока **postgres** с измененным значением поля приведен ниже:

env-secrets.dec.yaml

```
postgres:
    adminUser: postgres
```

Примечание: другие поля не изменяются.

3. Зашифровать обратно файл с секретами командой ниже:

Шифрование секретов

```
./zifctl secrets encrypt --env-path ENV-PATH --age-key AGE_KEY
```

Примечание: файл **env-secrets.dec.yaml** можно удалить после зашифровки.

3.7.4 Выполнение установки платформы release-2.22

Sync release-2.22

```
./zifctl sync --env-path ENV-PATH --age-key AGE_KEY --version 2.22-latest --pull
```

3.7.5 Восстановление данных из резервной копии

Для начала рекомендуется масштабировать в 0 все сервисы Платформы, кроме сервисов **Patroni**. Это можно сделать как через UI при помощи кнопки **Lens**, так и командой:

kubectl scale ->0. Example

```
kubectl scale --replicas=<number_of_replicas> <resource_type>/<resource_name> <options>
```

Восстановить данные из резервной копии можно при помощи скрипта, который приведен ниже в качестве примера:

Sync release-2.22

```
#!/bin/bash


DBLIST="
ziiot__keycloak
ziiot__zif-cm-metadata
ziiot__zif-dashboard
ziiot__zif-data-emulator
ziiot__zif-datalink
ziiot__zif-datasources
ziiot__zif-events
ziiot__zif-events-integration
ziiot__zif-export-nifi-collectors
```

```
ziiot__zif-hierarchies
ziiot__zif-interface-connect
ziiot__zif-interface-manager
ziiot__zif-licensing
ziiot__zif-material-lot
ziiot__zif-mnemoschemes-storage
ziiot__zif-notifications
ziiot__zif-om-data-observer
ziiot__zif-om-datareferences
ziiot__zif-om-object
ziiot__zif-om-objectmodel2excel
ziiot__zif-om-process
ziiot__zif-om-relationship
ziiot__zif-om-sqldatasource
ziiot__zif-om-testspecification
ziiot__zif-om-uom
ziiot__zif-om-valuetypes
ziiot__zif-portal-settings
ziiot__zif-propertyset
ziiot__zif-quality-service
ziiot__zif-rdm-common
ziiot__zif-reporting
ziiot__zif-reporting-sts
ziiot__zif-rtdb-metadata
ziiot__zif-security
ziiot__zif-sm-directories
ziiot__zif-sm-operationdefinition
ziiot__zif-sm-operationperformance
ziiot__zif-sm-operationsschedule
ziiot__zif-sm-process
ziiot__zif-sm-testspecification
ziiot__zif-sm-workcalendar
ziiot__zif-sm-workdefinition
ziiot__zif-sm-workperformance
```

```
ziiot__zif-sm-workschedule
ziiot__zif-udl-dsentitywebapi
ziiot__zif-universal-datamart
ziiot__zif-workflow
"

export PGHOST="${POSTGRES_HOST}"

export PGPORT="${POSTGRES_PORT}"



for i in $DBLIST; do
                echo "-----------------${i}------------------"
                DBNAME=${i}
                DUMP=$DBNAME.sql
                psql --host $PGHOST --port $PGPORT -d postgres -c "DROP DATABASE \"${DBNAME}\";"
                psql --host $PGHOST --port $PGPORT -d postgres -c "CREATE DATABASE \"${DBNAME}\" OWNER \"${DBNAME}\";"
                pg_restore -Fc --host $PGHOST --port $PGPORT -d ${DBNAME} < ./${DUMP}
done
```

3.7.6 Повторное выполнение команды sync после восстановления

Включить Платформу возможно следующими способами:

1. Масштабирование Платформы с 0 до необходимого числа реплик.
2. Выполнение повторной команды **sync**:

```
Sync release-2.22
./zifctl sync --env-path ENV-PATH --age-key AGE_KEY --version 2.22-latest --pull
```

4 Развертывание Платформы в виде docker-образа

Внимание! Данный раздел применим только для Релиза 2.9.0 и более поздних.

Система развертывания **Платформы** (обычно называемая **zifctl** по названию скрипта запуска) до релиза 2.9.0 поставлялась в виде исходного кода, как **git**-репозиторий. Для установки требовалось скачать репозиторий и запустить из него скрипт **zifctl** с необходимыми параметрами.

Начиная с релиза 2.9.0, инструмент развертывания **Платформы** поставляется в виде готового **docker**-образа, тег которого совпадает с версией устанавливаемой версии **Платформы** (плюс, возможно, суффикс-идентификатор сборки).

Это дает следующие преимущества:

1) Нет необходимости иметь доступ к **git**-репозиторию на целевой площадке или приносить его туда в архиве. Достаточно только наличия **docker**-образа в локальном **registry** (образы для Платформы все равно туда переносить или зеркалировать).
2) Нет необходимости в скачивании и переносе **helm-chart** для развертывания Платформы. Необходимые версии чартов положены в образ на этапе сборки.

Из недостатков можно отметить более сложную схему запуска инструмента т.к. приходится пользоваться **docker**-контейнером, в который необходимо передавать пути с локальной машины. Для облегчения этой задачи разработан специальный скрипт-обертка.

Для установки Платформы необходимо иметь доступ к **docker-registry**, где размещаются образы самой Платформы, ее инфраструктурных зависимостей и контейнер с системой развертывания.

Далее в командах будет использоваться репозиторий **Цифры**: https://registry.dp.zyfra.com/repository/. Если Платформа устанавливается в закрытом контуре, следует заменить адреса **registry**, и путь к контейнеру (для переноса контейнеров в registry в закрытом контуре, можно воспользоваться новой командой **zictl image scripts**).

4.1 Логин в docker registry

Для **Qauy** необходимо использовать **encrypted password** для логина через **CLI**, который можно получить в **UI** в своем профиле (**Account Settings**) и разделе **Docker CLI Password**:

```
echo $PASSWORD | docker login registry.dp.zyfra.com -u $USERNAME --password-stdin
```

4.2 Получение образа и запуск контейнера

Пробный запуск и получение помощи по доступным командам, получение информации о версии Платформы, которую развертывает контейнер, выглядит следующим образом:

```
# Получить образ
docker pull registry.dp.zyfra.com/infra/zifctl:2.9.0

# Запуск простых команд сразу с образа
docker run --rm  registry.dp.zyfra.com/infra/zifctl:2.9.0 --help
```

```
docker run --rm  registry.dp.zyfra.com/infra/zifctl:2.9.0 version
```

4.3 Получение скрипта-обертки (wrapper script)

Для интерактивной работы в терминале рекомендуется получить из контейнера скрипт-обертку для упрощенного запуска (скрипт позволяет указывать локальные пути и передает переменные окружения **ZIF_SERVER**, **ZIF_TOKEN**, **ZIF_AGE_KEY**, **ZIF_ENV_PATH** в запускаемый контейнер).

С точки зрения запуска скрипт-обертка примерно аналогична **zifctl** в прошлых релизах. Использование этого скрипта не обязательно (можно использовать **docker run**), но он добавляет возможность записи вывода контейнера в лог-файл (сам контейнер выводит все сообщения просто на консоль).

Скрипт-обертка обновляется вместе с системой развертывания. Необходимо обновлять его при переходе на новые версии системы и Платформы (каждый новый **docker**-образ может содержать новую версию **wrapper**).

Внутри скрипта зашита версия контейнера, с которого он получен и по умолчанию он будет запускать команды именно на нем. Посмотреть какой контейнер будет запускать скрипт можно выполнив команду **zifctl version.**

Для запуска этого скрипта на машине должен быть установлен **Python 3** (достаточно только штатных библиотек).

```
# Опционально скопировать скрипт в путь доступный через PATH
docker run --rm  registry.dp.zyfra.com/infra/zifctl:2.9.0 get-wrapper > zifctl
chmod +x zifctl
sudo cp zifctl /usr/local/bin
```

Проверка работы скрипта-обертки:

```
# Помощь по командам
./zifctl --help

# Информация о версиях используемых образов в скрипте, информация о версии Платформы
./zifctl version
```

Внимание! Скрипт-обертка обновляется вместе с системой развертывания. Необходимо обновлять его при переходе на новые версии системы и Платформы (каждый новый **docker**-образ может содержать новую версию **wrapper**).

Внимание! Внутри скрипта зашита версия контейнера, с которого он получен и по умолчанию он будет запускать команды именно на нем. Посмотреть какой контейнер будет запускать скрипт можно выполнив команду **zifctl version**

Внимание! Также внутри скрипта зашита ссылка на реестр/имя контейнера. Это можно поменять при помощи аргумента **--zifctl-registry** docker.idp.yc.ziiot.ru/infra (например, для переключения на реестр ЦИП).

4.4 Создание новой конфигурации окружения (если развертывается новый экземпляр Платформы)

Если **Платформа** устанавливается с нуля, то нужно создать конфигурацию окружения (конфигурации экземпляра) Платформы.

Конфигурация окружения представляет собой каталог с файлами, в которых сохранены все необходимые данные для разворачивания с нуля экземпляра **Платформы** или его обновления.

В общем случае предполагается, что конфигурация хранится в **git** и платформа ставится и обновляется по модели **GitOps**, в которой «источником правды» служит репозиторий, и по данным из него создаются все объекты в кластере (кроме **PVC** с данными, которые сохраняются при обновлениях).

4.5 Создание ключа шифрования и задание переменной ZIF_AGE_KEY

В конфигурации окружения так же хранятся и все пароли, **connection string** и прочие секреты, необходимые для работы Платформы. Т.к. секреты хранятся вместе с конфигурацией в **git**, они должны быть зашифрованы.

Система развертывания обеспечивает прозрачное шифрование секретов с помощью **Mozilla SOPS** (https://github.com/mozilla/sops) и утилиты (бэкенда) **age** (https://github.com/FiloSottile/age).

Шифрование требует секретного ключа (**private key**) для утилиты **age**, обычно уникального для данного экземпляра.

При развертывании нового экземпляра вам необходимо создать такой ключ и сохранить его (ключ не должен сохранятся в **git**-репозитории вместе с данными, которые он шифрует). Команда для создания нового ключа и сохранения его в файл:

```
# Создание нового ключа шифрования и запись его файл
./zifctl secrets new-age-key > key.txt
```

Полученный файл будет иметь вид ниже. Строка, начинающаяся с **AGE-SECRET-KEY-…** и есть нужный закрытый ключ. Полученный файл необходимо сохранить любым надежным способом:

```
AGE-SECRET-KEY-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Полученный ключ требуется указывать практически для всех команд утилиты развертывания, которые выполняют любую работу с экземпляром платформы. Для упрощения работы можно записать этот ключ в переменную окружения **ZIF_AGE_KEY**, и не указывать параметр **--age-key** во всех командах ниже (при постоянной работе с одной площадкой можно записать создание этой переменной в профиль **shell**):

```
export                                                    ZIF_AGE_KEY="AGE-SECRET-KEY-
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
```

4.6 Создание новой конфигурации окружения

Новая конфигурация окружения создается командой **init**. Папка для создания задается параметром **--env-path** (папка должна существовать). В этой папке создаются файлы **env-values.yaml** и **env-secrets.yaml** и несколько дополнительных.

Все параметры команды **init** кроме **--env-path** *и* **--age-key** *(*или переменная **ZIF_AGE_KEY***)* опциональные и фактически задают значения для основных параметров в файлах **env-values.yaml** и **env-secrets.yaml**. Их можно задать или изменить и после создания каталога с конфигурацией (т.к. секреты в **env-secrets.yaml** зашифрованы то задать логин и пароль **registry** будет сложнее чем остальные).

```
# Помощь по параметрам команды init
./zifctl init --help


# Информация о версиях используемых образов в скрипте, информация о версии Платформы
./zifctl init --env-path /path/to/env/config --age-key $KEY \
      --namespace $NAMESPACE \
      --sa $SA \
      --hostname $ZIIOT_HOSTNAME \
      --storage $STOR_CLASS \
      --registry $REGISTRY \
      --registry-username $REG_USERNAME \
      --registry-password $REG_PASSWORD \
      --verbose
```

Основные параметры команды **init**, задающие значения в новой конфигурации:

- **--namespace** — задает **Namespace** куда будет устанавливаться **Платформы**.
- **--sa** — **Service Account** для запуска инфраструктурных контейнеров, которые требуют запуска от конкретного **UID** (т.е. повышенных привилегий). Актуально в первую очередь для **OpenShift/OKD** инсталляций. Если не задан, то все запускается от учетной записи по умолчанию.
- **--hostname** — базовое доменное имя для доступа к экземпляру Платформы (по этому имени будет доступен портал, а сервисы будут иметь пути вроде **https://<base-hostname>/<service-name>** *(указывается не ссылка* **URL***, а именно* **hostname***).*
- **--storage** — имя **Storage Class** для создания **PVC** всей инфраструктурой. Если не задано, то используется **SC** по умолчанию в кластере.

- **--registry** — имя репозитория откуда кластер **K8s/OpenShift** должен брать контейнеры Платформы и инфраструктуры (по умолчанию **registry.dp.zyfra.com**).
- **--registry-username** — имя пользователя для репозитория. Из него и **registry-password** формируется **pullSecret**, в большинстве случаев должно быть указано, если только это не полностью открытый внутренний репозиторий).
- **--registry-password** — пароль пользователя для репозитория.

При необходимости внесите изменения в созданную в команде **init** конфигурацию окружения. Для создания простого стенда Платформы с размещением всей инфраструктуры в кластере достаточно указанных выше параметров.

Все основные параметры развертывания указываются в **env-values.yaml.**

Секреты для инфраструктурных сервисов указываются в файле **env-secrets.yaml.** Данные в этом файле зашифрованы с помощью `age` и для доступа к ним можно воспользоваться группой команд **zifctl secrets** (краткая помощь **zifctl secrets --help**).

Секреты генерируются автоматически, и вам нужно их править только если используется внешние инфраструктурные сервисы, например СУБД, которые развернуты независимо от **zifct** (или если не указан пароль к **registry** при выполнении **Init**).

4.7 Доступ к кластеру Kubernetes или OpenShift для развертывания и переменные ZIF_SERVER и ZIF_TOKEN

Для развертывания Платформы в кластер **Kubernetes** или **OpenShift** вам понадобится токен сервисного аккаунта или пользователя, позволяющий выполнять изменения в указанном **Namespace** (у него должна быть роль **admin** или близкая к этому на данный **namespace**).

- Если Платформа устанавливается в **Kubernetes**, то необходимо создать **Service Account** для этой задачи и получить его токен (см. о **SA** и токенах по ссылке, токен указывается в раскодированном виде, не в **base64**).
- Если Платформа устанавливается в **OpenShift**/**OKD** то есть возможность получить токен через **UI** для своей рабочей записи, или создать **SA** для установки так же как и для **Kubernetes**.
- Этот **Service Account** не обязан быть тем же, что и указанный для запуска подов в **env-values.yaml**. Он может отличаться, т.к. обычно аккаунту для запуска подов не рекомендуется давать права для внесения изменений в кластер.

Для всех команд утилиты развертывания, которые требуют доступа в кластер **Kubernetes/OpenShift**, необходимо задавать два параметра командной строки **--server** и **--token**, в которых указывается **URL API** кластера и токен для доступа соответственно.

При работе с одной и той же площадкой эти значения можно задать в виде переменных окружения и не указывать их в каждой команде (так же как ключом шифрования и переменной **ZIF_AGE_KEY**) или прописать их в профиль **shell**:

```
export ZIF_SERVER="https://api.cluster.local"
export ZIF_TOKEN="eyJhbGcNYWNjb3VuIsImtpZCI6IkZLOE5DSUlPZ0dtMXhPYUhvTDc2eGtf..."
```

4.8 Развертывание или обновление Платформы

Команда для развертывания и для обновления платформы одна и та же. Практически все операции, выполняемые системой развертывания, идемпотентны и при повторном выполнении установки, поверх уже выполненной ранее ничего не должно измениться.

Команда установки («синхронизации» конфигурации с конфигурацией в кластере). Параметры **--age-key**, **--server**, **--token** могут быть заданы через переменные окружения **ZIF_AGE_KEY**, **ZIF_SERVER** и **ZIF_TOKEN** соответственно).

```
# Запуск развертывания с выводом информации только на консоль
./zifctl sync --env-path /path/to/env/config --age-key $KEY --server $SERVER --token $TOKEN --verbose


# Запуск развертывания с выводом информации на консоль и в лог-файл с ротацией (параметр --log-keep не обязательный, по умолчанию сохраняется 10 последних логов)
./zifctl sync --env-path /path/to/env/config --age-key $KEY --server $SERVER --token $TOKEN --verbose --log-path /var/logs/ziiot-deploy --log-keep 20
```

Если выполняется установка поверх предыдущей версии, то в общем случае должно выполниться обновление сервисов до новой версии.

Но т.к. **Платформа** представляет собой сложный комплекс из большого количества сервисов и компонентов, то полностью автоматическое обновление условно гарантируется только в рамках одного релиза (например, из 2.9.0 в 2.9.1), а обновление между релизами (например, из 2.8.3 в 2.9.0) может потребовать ручных операций.

Перед обновлением необходимо прочитать **Release Notes Платформы** и системы развертывания и убедиться, что нет обязательных ручных операций.

4.9 Удаление развернутой Платформы из кластера

Команды для удаления развернутой платформы из кластера:

```
# Удаление всех сервисов и инфраструктуры без удаления PVC
./zifctl delete --env-path /path/to/env/config --age-key $KEY --server $SERVER --token $TOKEN --verbose


# Удаление всех сервисов и инфраструктуры с удалением PVC (полная очистка). Созданные в Namespace дополнительно вручную объекты удалены не будут.
./zifctl clean --env-path /path/to/env/config --age-key $KEY --server $SERVER --token $TOKEN --verbose
```

4.10 Развертывание платформы КЦ и ЦЕХ в режиме множественной БДВР

4.10.1 Краткое описание

Развертывание для ЦЕХ-а (WORKSHOP):

1. Инфраструктурные сервисы:
    - **zif-cassandra**;
    - **zif-kafka** - если используется Kafka для каждого ЦЕХ-а отдельно;
    - **zif-nifi**;
    - **zif-postgres17**;
    - **zif-redis**.
2. Сервисы платформы:
    - **zif-quality-service**;
    - **zif-rtdb-data**;
    - **zif-rtdb-data-consumerhost**;
    - **zif-rtdb-metadata**.
3. В случае использования **process** модуля:
    - **zif-data-emulator**;
    - **zif-interface-connect**;
    - **zif-opcua-server**.

Для КЦ (КС) используется стандартный набор всех сервисов.

Для установки используется образ с необходимыми изменениями - **feature-platform01-47649**. Образ создан на основе сборки **release-2.22**. Для удобства получаем **zifctl get-wrapper**:

```
zifctl get-wrapper

    docker run --rm docker.idp.yc.ziiot.ru/infra/zifctl:feature-platform01-47649
get-wrapper > zifctl
```

4.10.2 Подготовка данных

Краткое описание действий:

1. Выполнить команду **init** для КЦ.
2. Внести изменения в **env-values** КЦ.
3. Выполнить команду **sync** для КЦ.
4. Выполнить команду **init-workshop** для ЦЕХ-а.
5. Внести изменения в **env-values** ЦЕХ-а.
6. Указать сокращенный список сервисов для ЦЕХ-а.
7. Получить зашифрованные секреты инфраструктуры КЦ и ЦЕХ-а.
8. Перенести секреты инфраструктуры части сервисов из КЦ в секреты ЦЕХ-а.

9. Зашифровать все секреты для ЦЕХ-а. Удалить расшифрованные секреты.
10. Выполнить команду **sync** для ЦЕХ-а.

4.10.3 Init КЦ

Первый этап выполняется стандартно, как обычный **init** Платформы.

Затем для КЦ (КС) выполняется команда:

Init KC

```
./zifctl init --env-path=/KC --token TOKEN_KC --age-key AGE_KEY_KC --namespace KC-NAMESPACE --sa KC_SA --hostname HOSTNAME --storage STORAGE_CLASS --registry REGISTRY_ADDR --registry-username REGISTRY_USER --registry-password REGISTRY_PASSWD --version feature-platform01-47649 --pull
```

4.10.4 Подготовка env-values для КЦ

Для подготовки **env-values** для КЦ необходимо:

1. Перейти в директорию КС или иную, если при выполнении команды **init** было указано другое название (**--env-path KC**).
2. Перейти в папку **env-config.**
3. Открыть на редактирование файл **env-values.yaml**.
4. Указать блок с настройками мульти БДВР.

Ниже приведен фрагмент файла **env-values** (только с блоками и строками, на которые необходимо обратить внимание, если значения другие):

env-values.yaml для КЦ

```
multipleTsds:
  udlDataSources:
############## Каждый блок workshop - это отдельный Цех. Необходимо заполнять все поля приведенные в блоке workshop ########################
    workshop:
      serviceUrl: "https://WORKSHOP-NAMESPACE.DOMAIN.CO/zif-rtdb-data"
      serviceExternalUrl: "https://WORKSHOP-NAMESPACE.DOMAIN.CO/zif-rtdb-data"
      additionalProperties:
############## Если используется кафка, расположенная в КЦ, то указываем её ниже ########################
        kafkaBootstrapServers: "zif-kafka-cp-kafka-headless.KC-NAMESPACE"
#########################################################################################################
############## Если используется кафка, расположенная в каждом ЦЕХе, то указываем её ниже ###############
        kafkaBootstrapServers: "zif-kafka-cp-kafka-headless.WORKSHOP-NAMESPACE"
```

```
##########################################################################################
######################
        kafkaTopic: "nifi-data-by-tag_workshop"
        kafkaOutcomingTagDataTopic: "outcoming-data-events_workshop"
        kafkaOutcomingTagDataTopicPartitions: 128
        restZifRtdbMetadataUrl:    "https://WORKSHOP-NAMESPACE.DOMAIN.CO/zif-rtdb-
metadata"
        restZifRtdbMetadataExternalUrl: "https://WORKSHOP-NAMESPACE.DOMAIN.CO/zif-
rtdb-metadata"


    workshop_2:
.................
        additionalProperties:
.................


     workshop_3:
.................
        additionalProperties:
.................
```

Примечание: описание параметров:

- **serviceUrl** и **serviceExternalUrl** - это ссылки на сервис **zif-rtdb-data**, расположенные в ЦЕХ-у. Рекомендуется использовать одинаковые ссылки на внешний ингресс;
- **kafkaBootstrapServers** - исходя из того, где расположена Kafka (общая или в каждом ЦЕХ-у своя), необходимо указывать соответствующий адрес;
- **kafkaTopic** - топик для записи значений тегов из ЦЕХ-а;
- **kafkaOutcomingTagDataTopic** - топик для записи значений **events** тегов;
- **kafkaOutcomingTagDataTopicPartitions** – число значений **partitions** для **data events** топика. По умолчанию 128. Если значение не меняется, то можно не указывать;
- **restZifRtdbMetadataUrl** и **restZifRtdbMetadataExternalUrl** - это ссылка на сервис **zif-rtdb-metadata** расположенный в ЦЕХ-у. Рекомендуется использовать одинаковые ссылки на внешний ингресс.

4.10.5 Выполнение команды sync для установки КЦ

Первый этап выполняется стандартная установка (**init**) Платформы для КЦ. Далее выполнить команду **sync** и дождаться полной установки:

```
Sync KC

./zifctl sync --env-path KC --age-key AGE_KEY_KC --version feature-platform01-47649
--pull
```

4.10.6 Init ЦЕХ-а

Для ЦЕХ-а (WORKSHOP) выполняется команда:

Init CEH

```
./zifctl init-workshop --env-path=/WORKSHOP --age-key AGE_KEY_WORKSHOP --namespace WORKSHOP-NAMESPACE --sa WORKSHOP_SA --server URL --hostname HOSTNAME --storage STORAGE_CLASS --registry REGISTRY_ADDR --registry-username REGISTRY_USER --registry-password REGISTRY_PASSWD --token TOKEN_KC --modules rtdb --modules process --corp-center-ns KC-NAMESPACE --data-source-name serverWORKSHOP --version feature-platform01-47649
```

Примечания:

- **tenantId** - должны совпадать в КЦ и ЦЕХ-е;
- **init-workshop** - новая команда для инициализации ЦЕХ-а;
- **token** - указать токен, который используется для сохранения секретов из КЦ. Данный токен используется для подключения к пространству имен КЦ;
- **modules** - данный параметр передает список модулей для загрузки секретов из КЦ. Синтаксис: **--modules rtdb --modules process**;
- **corp-center-ns** - название пространства имен, где располагается КЦ. Отсюда будут забираться секреты для ЦЕХ-а;
- **data-source-name** - параметр, который позволяет КЦ различать ЦЕХ-а между собой. Для каждого ЦЕХ-а данный параметр должен быть уникален.

Ниже приведен список доступных параметров при установке(**init**) ЦЕХ-а:

Справка по команде init-workshop

```
Usage: zifctl init-workshop [OPTIONS]


  Create new environment configuration for multi-TSDS workshop


Required options:

  --env-path PATH               Path to environment configuration directory  [env var: ZIF_ENV_PATH; required]

  --age-key KEY                          age utility private key (identity) for encryption/decryption secrets  [env var:
                                ZIF_AGE_KEY; required]

  --server URL                  OpenShift/Kubernetes API URL  [env var: ZIF_SERVER; required]

  --token SECRET                OpenShift/Kubernetes service account token for API access  [env var: ZIF_TOKEN; required]

  --corp-center-ns STRING     Corporate center kubernetes namespace for current TSDS workshop instance  [env var:
                                ZIF_CORP_CENTER_NS; required]
```

```
  --data-source-name STRING   Current TSDS workshop instance data source name  [env
var: ZIF_DATA_SOURCE_NAME;
                              required]


Registry options:

  --registry URL                Container registry for deployment to set in env-
values.yaml  [env var: ZIF_REGISTRY;
                              default: http://docker.idp.yc.ziiot.ru]

  --registry-username SECRET   Container registry login username to set in env-
secrets.yaml  [env var:
                              ZIF_REGISTRY_USERNAME]

  --registry-password SECRET  Container registry password for deployment to set in
env-values.yaml  [env var:
                              ZIF_REGISTRY_PASSWORD]


Other options:

  --hostname URL              Main hostname for URL of new platform instance  [env
var: ZIF_HOSTNAME]

  --namespace STRING             Kubernetes namespace (project for OpenShift) to
Platform deploy  [env var: ZIF_NAMESPACE]

  --sa STRING                   Name of ServiceAccount, that will be used to run
services and infra. Default:
                              '<namespace>-sa'  [env var: ZIF_SERVICE_ACCOUNT]

  --storage STRING              StorageClass for PVC of the infra (default: using
cluster default SC)  [env var:
                              ZIF_STORAGE]

  --tenant STRING               ID of tenant, if you create tenant configuration
(required if type == 'tenant')  [env
                              var: ZIF_TENANT]

  --tenant-name STRING       Name of the tenant, if not provided it will be created
from tenant ID (--tenant)  [env
                              var: ZIF_TENANT_NAME]

  --infra-ns STRING          Namespace where the shared infra located (required if
type == 'tenant')  [env var:
                              ZIF_INFRA_NS]

  --infra-ha                 Deploy infrastructure in HA configuration (for shared
infrastructure true by default)
                              [env var: ZIF_INFRA_HA]
```

```
  -f, --force                   Overwrite configuration if it already exists  [env
var: ZIF_FORCE]

  --modules MULTIPLE             List of modules to read secrets from  [env var:
ZIF_MODULES; default: rtdb]

  -v, --verbose                Enable verbose command output  [env var: ZIF_VERBOSE]

  -vv, --debug                 Enable verbose command output and helmfile/helm debug
output  [env var: ZIF_DEBUG]

  -h, --help                    Show this message and exit.
```

4.10.7 Подготовка env-values для ЦЕХ-а

Для подготовки **env-values** для ЦЕХ-а необходимо:

1. Перейти в директорию **WORKSHOP** или иную, если при **init-wokshop** было указано другое название (**--env-path WORKSHOP**).
2. Перейти в директорию **env-config** и открыть на редактирование файл **env-values.yaml**.
3. Указать блок с настройками мульти БДВР.
4. Отключить некоторые инфраструктурные сервисы, отключить некоторые задачи **init-tasks** и указать ссылки на инфраструктурные сервисы, расположенные в КЦ.

Ниже приведен фрагмент файла **env-values** (только с блоками и строками, на которые необходимо поменять, если значения другие):

env-values.yaml для ЦЕХ-а

```
init:
  disabledTasks: [keycloak, rabbitmq, s3storage]


nifi:
############ Если используется kafka, расположенная в КЦ, то включаем Zookeeper
#########################
  createZookeeper: True
##############################################################################
######################


keycloak:
  installed: False
  baseExtUri: 'https://KC-NAMESPACE.DOMAIN.CO/auth'


s3storage:
  default:
    installed: False
```

```
    fileStorageServiceEnabled: False


pgadmin:
  installed: False


kafkaui:
  installed: False


############ Если используется kafka, расположенная в КЦ, то указываем её в блоке ниже ########################
kafka:
  installed: False
  host: 'zif-kafka-cp-kafka-headless.KC-NAMESPACE'
  ## Kafka Schema Registry
  registry: 'zif-kafka-schema-cp-schema-registry.KC-NAMESPACE'
  ## Kafka Connect
  connect: 'zif-kafka-connect-cp-kafka-connect.KC-NAMESPACE'
  ## Zookeeper
  zookeeper: 'zif-kafka-zookeeper-cp-zookeeper-headless.KC-NAMESPACE'



multipleTsds:
  kafkaOutcomingTagDataTopic: "outcoming-data-events_workshop"
```

4.10.8 Набор модулей для ЦЕХ-а

Для набора модулей для ЦЕХ-а необходимо:

1. Перейти в директорию **WORKSHOP** или иную, если при **init-wokshop** было указано другое название (**--env-path WORKSHOP**).
2. Перейти в директорию **env-config.**
3. Создать там директорию **services**.
4. Внутри директории **services** создать файл **service-list-patch.yaml**.

Результат: создается файл **service-list-patch.yaml** с путем **WORKSHOP/env-config/services/**.

Содержание файла **service-list-patch.yaml**:

```
service-list-patch.yaml для ЦЕХ-а
modules:
```

```
- name: ziiot-docs
  enabled: false
- name: rtdb
  enabled: true
  services:
    - name: zui-app-rtdb
      enabled: false
- name: mnemoscheme
  enabled: false
- name: process
  enabled: false
- name: security
  enabled: false
- name: licensing
  enabled: false
- name: om
  enabled: false
- name: sm
  enabled: false
- name: udl
  enabled: false
- name: notifications
  enabled: false
- name: events
  enabled: false
- name: calc
  enabled: false
- name: bp-workers
  enabled: false
- name: documents
  enabled: false
- name: portal
  enabled: false
- name: npi
```

```
    enabled: false
```

Примечание: список модулей может меняться, так как периодически добавляются новые модули Платформы. При появлении новых модулей необходимо их отключить по такому же принципу, который был описан выше.

4.10.9 Набор модулей для ЦЕХ-а с включенными дополнительными сервисами

Набор модулей для ЦЕХ-а с включенными дополнительными сервисами:

- **zif-data-emulator**;
- **zif-interface-connect**;
- **zif-opcua-server**.

service-list-patch.yaml для ЦЕХ-а

```
modules:
  - name: ziiot-docs
    enabled: false
  - name: rtdb
    enabled: true
    services:
      - name: zui-app-rtdb
        enabled: false
  - name: mnemoscheme
    enabled: false
  - name: process
    enabled: true
    services:
      - name: zui-app-reporteditor
        enabled: false
      - name: zui-app-reporteditor-sts
        enabled: false
      - name: zui-app-datainput
        enabled: false
      - name: zui-app-dashboard-dx
        enabled: false
      - name: zui-app-datalinkeditor
        enabled: false
      - name: zui-app-workflow
```

```
      enabled: false
    - name: zif-dashboard
      enabled: false
    - name: zif-reporting
      enabled: false
    - name: zif-reporting-sts
      enabled: false
    - name: zif-workflow
      enabled: false
    - name: zif-datainput
      enabled: false
    - name: zif-universal-datamart
      enabled: false
    - name: zui-app-universal-datamart
      enabled: false
    - name: zif-datasources
      enabled: false
    - name: zui-app-datasources
      enabled: false
    - name: zui-app-interface-manager
      enabled: false
    - name: zui-app-interface-connect
      enabled: false
    - name: zif-interface-manager
      enabled: false
    - name: zif-workflow-relocator
      enabled: false
    - name: zif-datalink-xl
      enabled: false
    - name: zif-datalink
      enabled: false
    - name: zif-datalink-scripts
      enabled: false
    - name: zif-export-nifi-collectors
```

```
        enabled: false
  - name: security
    enabled: false
  - name: licensing
    enabled: false
  - name: om
    enabled: false
  - name: sm
    enabled: false
  - name: udl
    enabled: false
  - name: notifications
    enabled: false
  - name: events
    enabled: false
  - name: calc
    enabled: false
  - name: bp-workers
    enabled: false
  - name: documents
    enabled: false
  - name: portal
    enabled: false
  - name: npi
    enabled: false
```

4.10.10 Расшифровка и перенос секретов из КЦ в секреты ЦЕХ-а

Расшифровка секретов для КЦ (КС):

```
Decrypt KC

./zifctl secrets decrypt --env-path KC --age-key AGE_KEY_KC
```

Примечание: секреты для просмотра находятся в **zif-global-secrets**, в развернутом пространстве имен КЦ. И можно не расшифровывать секреты для КЦ командой выше.

Расшифровка секретов для ЦЕХ-а (СЕН):

```
decrypt CEH

./zifctl secrets decrypt --env-path WORKSHOP --age-key AGE_KEY_WORKSHOP
```

4.10.11 Перенос части секретов инфраструктуры из КЦ в секреты ЦЕХ-а

После выполнения процесса расшифровки необходимо внести изменения в расшифрованные файлы.

Выбрать данные о секретах из файла **env-secrets.dec.yaml** расположенных в каталоге **KC/env-config** и заменить на данные о секретах ЦЕХ-а, расположенных по пути **WORKSHOP/env-config/env-secrets.dec.yaml**. Секреты для переноса указаны ниже:

Секреты для переноса из КЦ в ЦЕХ

```
keycloak:
    zifRealmAdminPassword:
nifi:
    adminPassword:
    keycloakClientSecret:
```

4.10.12 Шифрование секретов и удаление расшифрованных данных

Шифрование секретов для ЦЕХ-а (WORKSHOP):

Encrypt CEH

```
./zifctl secrets encrypt --env-path /WORKSHOP --age-key AGE_KEY_WORKSHOP
```

Примечание: после завершения процедуры, файл **WORKSHOP/env-config/env-secrets.dec.yaml** может быть удален.

4.10.13 Выполнение команды sync для установки КЦ

Выполнить команду **sync** ЦЕХ-а:

Sync KC

```
./zifctl sync --env-path WORKSHOP --age-key AGE_KEY_WORKSHOP --version feature-platform01-47649 --pull
```

4.10.14 Примеры конфигураций

Ниже приведены примеры конфигурации КЦ и ЦЕХ-а, которые использовались во время тестирования:

env-values KC

```
# ======================================================================
# Файл конфигурации для платформы ZIIoT
# Тип конфигурации: instance
# Namespace:        dev-guseynov
# ======================================================================


##-------------------------------------------------------------------------
# Общие параметры для всей инсталляции
```

```
##-----------------------------------------------------------------------------

configType: instance
namespace:  'dev-guseynov'
serviceAccount: 'dev-guseynov-sa-admin-fuad-guseynov'
tenantName: 'ziiot'
tenantId: 'ziiot'
externalBaseHostname: 'dev-guseynov.kube07.yc.ziiot.ru'

infraHA: False

# Используемое хранилище
storage:
  storageClass: yc-network-ssd



# Используемое Container Registry и namespace'ы в нем (сервисы и инфраструктура)
registry:
  name: 'docker.idp.yc.ziiot.ru'
  namespace: '/digital-plant'
  infraNamespace: '/infra'

# Выделяемые по-умолчанию ресурсы (Requests/Limits) по типовой таблице
serviceResources: 'small'
infrastructureCapacity: 'sm'

## Конфигурация jaeger
jaeger:
  enabled: False
  host: ''
  port: 6831

## Конфигурация OpenTelemetry
tracing:
```

```
  enabled: False
  otlpHost: ''

## Конфигурация ABAC
abacAuthorization:
  enabled: True

## Конфигурация встроенного PKI
pki:
  enabled: True
  certManager:
    enabled: False
    issuer:
      create: False

## Конфигурация подсистемы инициализации сервисов после установки
## (Init subsystem)
init:
  tasks:
    - all

  disabledTasks: []
  ansibleLogVerbosity: 2

##------------------------------------------------------------------------------
## Параметры инфраструктурных сервисов
##------------------------------------------------------------------------------

## Конфигурация СУБД Postgre SQL
postgres:
  installed: True
  configured: True
  host: 'zif-postgres.dev-guseynov'
  port: 5432
```

```
  storageSizeMB: 5120
  version: patroni-17
  tls:
    enabled: True

## Конфигурация KeyCloak
keycloak:
  installed: True
  configured: True
  baseExtUri: 'https://dev-guseynov.kube07.yc.ziiot.ru/auth'
  version: 21

## Конфигурация СУБД Cassandra
cassandra:
  installed: True
  configured: True
  contactPoints: 'zif-cassandra-headless.dev-guseynov'
  port: 9042
  storageSizeMB: 1024
  nodes: 1
  tls:
    enabled: False

## Конфигурация RabbitMQ
rabbitmq:
  installed: True
  configured: True
  keycloakAuth:
    enabled: True
  host: 'zif-rabbitmq-headless.dev-guseynov'
  port: 5672
  hosts:
    - zif-rabbitmq-headless.dev-guseynov
```

```
## Конфигурация Kafka и Schema Registry
kafka:
  installed: True
  configured: True
  tls:
    internal: 'none'
  externalAccess: False

  storageSizeMB: 1024

  ## Kafka
  host: 'zif-kafka-cp-kafka-headless.dev-guseynov'
  port: 9092

  ## Kafka Schema Registry
  registry: 'zif-kafka-schema-cp-schema-registry.dev-guseynov'
  registryPort: 8081

  ## Kafka Connect
  connect: 'zif-kafka-connect-cp-kafka-connect.dev-guseynov'
  connectPort: 8083

  ## Zookeeper
  zookeeper: 'zif-kafka-zookeeper-cp-zookeeper-headless.dev-guseynov'
  zookeeperPort: 2181

## Конфигурация Redis
redis:
  installed: True
  configured: True
  host: 'zif-redis.dev-guseynov'
  port: 6379
  storageSizeMB: 1024
```

```
## Конфигурация Redis используемого только инфраструктурными сервисами
redisInfra:
  installed: False
  configured: False
  host: 'zif-redis-infra.dev-guseynov'
  port: 6379
  storageSizeMB: 1024

## Конфигурация NiFi
nifi:
  installed: True
  configured: True
  extHostname: 'dev-guseynov.kube07.yc.ziiot.ru'
  createZookeeper: false
  externalZookeeperHost: 'zif-kafka-zookeeper-cp-zookeeper-headless.dev-guseynov'
  externalZookeeperPort: 2181
  auth: 'oauth-proxy'
  extensionsStorage: False
  processors: []
  nodes: 1

## Конфигурация MinIO/S3
s3storage:
  default:
    installed: True
    configured: True
    uri: 'https://dev-guseynov-s3.kube07.yc.ziiot.ru'
    internalUri: 'http://zif-minio.dev-guseynov:9000'
    ui: True
    provisioning: True
    fileStorageServiceEnabled: True
    driveSizeMB: 1024
    drivesPerNode: 1
```

```
    nodes: 1
    parity: 2


## Конфигурация pgadmin
pgadmin:
    installed: True
    configured: True
    keycloakAuth:
      enabled: True
    storage:
      enabled: False


## Конфигурация KafkaUI
kafkaui:
  installed: True
  configured: True


multipleTsds:
  udlDataSources:
    ServerBDVR:
      serviceUrl: "https://dev-guseynov-bdvr.kube07.yc.ziiot.ru/zif-rtdb-data"
      serviceExternalUrl:  "https://dev-guseynov-bdvr.kube07.yc.ziiot.ru/zif-rtdb-
data"
      additionalProperties:
        kafkaBootstrapServers: "zif-kafka-cp-kafka-headless.dev-guseynov"
        kafkaTopic: "nifi-data-by-tag_ServerBDVR"
        kafkaOutcomingTagDataTopic: "outcoming-data-events_ServerBDVR"
        restZifRtdbMetadataUrl: "https://dev-guseynov-bdvr.kube07.yc.ziiot.ru/zif-
rtdb-metadata"
        restZifRtdbMetadataExternalUrl:                    "https://dev-guseynov-
bdvr.kube07.yc.ziiot.ru/zif-rtdb-metadata"
```

env-values ЦЕХ-а

```
# ====================================================================
# Файл конфигурации для платформы ZIIoT
```

```
# Тип конфигурации: instance
# Namespace:        dev-guseynov-bdvr
# ======================================================================


##-------------------------------------------------------------------------
# Общие параметры для всей инсталляции
##-------------------------------------------------------------------------

configType: instance
namespace:  'dev-guseynov-bdvr'
serviceAccount: 'dev-guseynov-bdvr-sa-ns-admin'
tenantName: 'ziiot'
tenantId: 'ziiot'
externalBaseHostname: 'dev-guseynov-bdvr.kube07.yc.ziiot.ru'
infraHA: False



# Используемое Container Registry и namespace'ы в нем (сервисы и инфраструктура)
registry:
  name: 'docker.idp.yc.ziiot.ru'
  namespace: '/digital-plant'
  infraNamespace: '/infra'

# Выделяемые по-умолчанию ресурсы (Requests/Limits) по типовой таблице
serviceResources: 'small'
infrastructureCapacity: 'sm'

## Конфигурация jaeger
jaeger:
  enabled: False
  host: ''
  port: 6831

## Конфигурация OpenTelemetry
```

```
tracing:
  enabled: False
  otlpHost: ''

## Конфигурация ABAC
abacAuthorization:
  enabled: True

## Конфигурация встроенного PKI
pki:
  enabled: True
  certManager:
    enabled: False
    issuer:
      create: False

## Конфигурация подсистемы инициализации сервисов после установки
## (Init subsystem)
init:
  tasks:
    - all

  disabledTasks: [keycloak, rabbitmq, s3storage]
  ansibleLogVerbosity: 4

##-------------------------------------------------------------------------
## Параметры инфраструктурных сервисов
##-------------------------------------------------------------------------

## Конфигурация СУБД Postgre SQL
postgres:
  installed: True
  configured: True
  host: 'zif-postgres.dev-guseynov-bdvr'
```

```
  port: 5432
  storageSizeMB: 5120
  version: patroni-17
  tls:
    enabled: False

## Конфигурация KeyCloak
keycloak:
  installed: False
  configured: True
  baseExtUri: 'https://dev-guseynov.kube07.yc.ziiot.ru/auth'
  version: 21

## Конфигурация СУБД Cassandra
cassandra:
  installed: True
  configured: True
  contactPoints: 'zif-cassandra-headless.dev-guseynov-bdvr'
  port: 9042
  storageSizeMB: 1024
  nodes: 1
  tls:
    enabled: False

## Конфигурация RabbitMQ
rabbitmq:
  installed: False
  configured: True
  keycloakAuth:
    enabled: True
  host: 'zif-rabbitmq-headless.dev-guseynov-bdvr'
  port: 5672
  hosts:
    - zif-rabbitmq-headless.dev-guseynov-bdvr
```

```
## Конфигурация Kafka и Schema Registry
kafka:
  installed: False
  configured: True
  tls:
    internal: 'none'
  externalAccess: False

  storageSizeMB: 1024

  ## Kafka
  host: 'zif-kafka-cp-kafka-headless.dev-guseynov'
  port: 9092

  ## Kafka Schema Registry
  registry: 'zif-kafka-schema-cp-schema-registry.dev-guseynov'
  registryPort: 8081

  ## Kafka Connect
  connect: 'zif-kafka-connect-cp-kafka-connect.dev-guseynov'
  connectPort: 8083

  ## Zookeeper
  zookeeper: 'zif-kafka-zookeeper-cp-zookeeper-headless.dev-guseynov'
  zookeeperPort: 2181

## Конфигурация Redis
redis:
  installed: True
  configured: True
  host: 'zif-redis.dev-guseynov-bdvr'
  port: 6379
```

```
  storageSizeMB: 1024
  tls:
    enabled: true
  auth: true
  acl: true

## Конфигурация Redis используемого только инфраструктурными сервисами
redisInfra:
  installed: False
  configured: False
  host: 'zif-redis-infra.dev-guseynov'
  port: 6379
  storageSizeMB: 1024

## Конфигурация NiFi
nifi:
  installed: True
  configured: True
  extHostname: 'dev-guseynov-bdvr.kube07.yc.ziiot.ru'
  createZookeeper: true
  externalZookeeperHost:  'zif-kafka-zookeeper-cp-zookeeper-headless.dev-guseynov-
bdvr'
  externalZookeeperPort: 2181
  auth: 'native'
  extensionsStorage: False
  processors: []
  nodes: 1

## Конфигурация MinIO/S3
s3storage:
  default:
    installed: False
    configured: True
    uri: 'https://dev-guseynov-bdvr-s3.kube07.yc.ziiot.ru'
    internalUri: 'http://zif-minio.dev-guseynov:9000'
```

```
    ui: True
    provisioning: True
    fileStorageServiceEnabled: False
    driveSizeMB: 1024
    drivesPerNode: 1
    nodes: 1
    parity: 2


## Конфигурация pgadmin
pgadmin:
    installed: false
    configured: True
    keycloakAuth:
      enabled: True
    storage:
      enabled: False


## Конфигурация KafkaUI
kafkaui:
  installed: False
  configured: True


## Конфигурация множественной БДВР
multipleTsds:
  dataSourceName: serverBDVR
  corpCenterNs: dev-guseynov
  kafkaOutcomingTagDataTopic: outcoming-data-events_ServerBDVR
```

4.11 Создание нескольких client для Keycloak через service-list

Для того, чтобы сохранить обратную совместимость и обеспечить создание дополнительных **client** добавлен дополнительный параметр **additionalKeycloakClient** в **service-list.yaml**.

Создание **client** происходит через **init-job** модуля, для новых **client** автоматически генерируются секреты.

Последовательность действий:

1. В **service-list** прописать список **client**, например:

```
  - name: process
```

```
  chart: zif-process-module
  description: Process
  services:
    .....
    - name: zif-datalink-xl
      gitLabProjectID: 1860
      image: zif-datalink-xl
      .....
      keycloakClient:
        redirectUris: ["*"]
        webOrigins: ["*"]
      .....
      additionalKeycloakClient:
        - name: "test_user_process_1"
          redirectUris: ["*"]
          webOrigins: ["*"]
        - name: "test_user_process_2"
          redirectUris: ["*"]
          webOrigins: ["*"]
          directAccessGrantsEnabled: False
```

Примечание: можно изменить параметры:

- **name** (по умолчанию **""**);
- **redirectUris** (по умолчанию **"/*");**
- **webOrigins (**по умолчанию **external_base_url**);
- **directAccessGrantsEnabled** (по умолчанию **True**).

2. Для назначения **ServiceAccountRoles** к дополнительным **client** необходимо вписать следующее:

```
- name: process
  chart: zif-process-module
  description: Process
  services:
    .....
    - name: zif-datalink-xl
      gitLabProjectID: 1860
      image: zif-datalink-xl
```

```
.....
keycloakClient:
  redirectUris: ["*"]
  webOrigins: ["*"]
.....
additionalKeycloakClient:
  - name: "test_user_process_1"
    redirectUris: ["*"]
    webOrigins: ["*"]
    serviceAccountRoles:
      roleName:
        - role-action-1
  - name: "test_user_process_2"
    redirectUris: ["*"]
    webOrigins: ["*"]
    directAccessGrantsEnabled: False
    serviceAccountRoles:
      roleName:
        - role-action-2
```

3. Чтобы назначить **ServiceAccountRoles**, нужно вписать следующее к дополнительным **client.**

4.12 Инструкция по использованию команды zifctl secrets recreate

4.12.1 Общие параметры

Общие параметры:

- **--scope -** опциональный флаг, который определяет область действия команды. Возможные значения:
  - **infra**;
  - **module**;
  - **all** (по умолчанию);
- **--exclude** - опциональный флаг для исключения определённых секретов из повторной генерации. Пример: **--exclude=keycloak.adminPassword --exclude=redis.** Не доступен при **--scope=module**.

4.12.2 Секреты, которые никогда не меняются

Базовые секреты:

- **registry.username**;
- **registry.password**;
- **logging.aggregator**;
- **logging.storage**;
- **logging.dashboards.**

Дополнительные секреты для типа развертывания `tenant`:

- **kafka.keystorePassword**;
- **nifi.keystorePassword**;
- **nifi.sensitivePropsKeyEncrypted.**

Дополнительные секреты для остальных типов развертывания:

- **cassandra.keystorePassword**;
- **kafka.keystorePassword**;
- **nifi.keystorePassword**;
- **nifi.sensitivePropsKeyEncrypted**;
- **rabbitmq.erlangCookie.**

Сервисы, которые в текущем варианте развертывания не поддерживают смену паролей, но будут проработаны позже:

- **logging.aggregator**;
- **logging.storage**;
- **logging.dashboards.**

4.12.3 Действия в зависимости от параметров

4.12.3.1 Примеры команд и их действия

4.12.3.1.1 Общие правила

Общие правила для всех вариантов:

- секреты, добавленные вручную и не относящиеся к инфраструктурным сервисам, не изменяются;
- секреты, относящиеся к инфраструктурным сервисам измененные вручную, должны быть исключены из перегенерации флагами **–exclude**;
- генерация новых секретов происходит для всех сервисов, кроме тех, которые указаны в списке неизменяемых (см. п. 4.12.2);
- команда не меняет секреты, переданные для исключения;

- команда не меняет пароли инфраструктурных сервисов, которые имеют параметр **installed: False**.

Примечания:

- тип развертывания определяется из параметра **configType** в **env-values.yaml**;
- параметр **--exclude** приведен для примера.

4.12.3.1.2 Применение для типа развертывания instance (по умолчанию)

Пример команды:

```
    zifctl secrets recreate --env-path=<path to dir env-configs> --exclude=keycloak.adminPassword --exclude=redis
```

Действия:

- копирует файл **env-config/env-secrets.yaml** в **env-config/env-secrets-backup.yaml**;
- генерирует новый файл **env-config/env-secrets.yaml**;
- генерирует новые файлы **env-config/values/<module-name>/mod-secrets.yaml**.

Пример команды:

```
    zifctl secrets recreate --env-path=<path to dir env-configs> --scope=infra --exclude=keycloak.adminPassword --exclude=redis
```

Действия:

- копирует файл **env-config/env-secrets.yaml** в **env-config/env-seccrets-backup.yaml**;
- генерирует новый файл **env-config/env-secrets.yaml**.

Пример команды:

```
    zifctl secrets regenerate --scope=module
```

Действия:

- генерирует новые файлы **env-config/values/<module-name>/mod-secrets.yaml**;
- флаг **--exclude** не доступен и вызовет ошибку, если будет указан.

4.12.3.1.3 Применение для типа развертывания tenant

Пример команды:

```
    zifctl secrets recreate --env-path=<path to dir env-configs> --exclude=redis
```

Действия:

- копирует файл **env-config/env-secrets.yaml** в **env-config/env-seccrets-backup.yaml**;
- получает пароли из развернутой **shared-infra**;
- генерирует новые файлы **env-config/values/<module-name>/mod-secrets.yaml**;
- не меняет пароли инфраструктурных сервисов, которые имеют параметр **installed: False** и устанавливаются в тенант.

4.12.3.1.4 Применение для типа развертывания shared-infra

Пример команды:

```
zifctl secrets regenerate
```

Действия:

- копирует файл **env-config/env-secrets.yaml** в **env-config/env-secrets-backup.yaml**;
- генерирует новый файл **env-config/env-secrets.yaml**.

4.12.3.1.5 Применение для типа развертывания app

Пример команды:

```
zifctl secrets recreate --env-path=<path to dir env-configs>
```

Действия: получает пароли из развернутой **shared-infra**.

4.12.3.2 Перезагрузка сервисов

При изменениях в **env-values.yaml** и **env-secrets.yaml** для конкретных сервисов, формируется хеш-сумма параметров и секретов и записывается в **config-signature.yaml** для последующего использования в качестве контрольных сумм в аннотациях манифестов Kubernetes.

Перезагрузка обеспечивается изменением аннотации в **deployment/sts** с последующим пересозданием подов.

Контрольные суммы файлов **mod-secrets.yaml** и **env-secrets.yaml** добавляются в аннотации сервисов для обеспечения перезагрузки при изменении секретов. Контрольные суммы добавляются только для тех сервисов, которые зависят от соответствующих инфраструктурных сервисов и вычисляются по параметру **<infra-service-name>: True** в **dservice-list.yaml** каждого сервиса Платформы, обеспечивая тем самым перезагрузку только тех сервисов, которые зависят от конкретных инфраструктурных сервисов.

Пример файла **config-signature.yaml**:

```
configSchemaVersion: 100
files:
  defaults: e7f04c55dafe6b2a1bb4a136bc83c760f96dee7fd930c827f152a2e595f1f6e7
  env-secrets.yaml:
5b2f59c5f9929d5d2a0b1aec3d635666054ca60a6047100d51a7da7d53ab6008
  env-values.yaml:
f18f16debd5ce82161f222c51072577571c6ad25411c60c952521b1187428bcd
  service-list-patch.yaml: ''
  service-list.yaml:
019f19492991972c338bdcee6851b14da9766fd1f3bafee02cfe1d3e4780e97c
infra:
  cassandra: eba529634436ea32e75e2aef61ae51808e7bc57bc438487f8829b3e513bbe566
  kafka: 5137f2dc9d485d3f22ac8f3e80e501ab5443350f52c37d294bb772222f5bb4a4
  kafkaui: f048f7787334ed48e33e849c6ffbb874d90764134497570a94833e843b72d333
```

```
      keycloak: eea63363be69ddc576bb91d7a7dd94a8dd57dfebbbabfec04e01bd58c7e8f35b
      logging:
        aggregator:
b0ec0398856958340a0aa14831aebccf053109ef0869a8188d7954b82b781b5d
        dashboards:
3f0c032f22eda87fed7cf1bdd4ec37105c90aeffcd8c9070ef4a518f827bf73e
        storage: 4ce1d6e3694018aa3955dfcb5b078afd1e08947bbc1073b303387f4b0f174d40
      nifi: 0492dd930895594c2872d208cbb23ce41d9a75c73677f6f02801d7ca1c504cb4
      pgadmin: 9ef509014dfcea7dee87b6a01889cc7b4bcff8252061da3e9a62366758f69157
      postgres: 410cbd5054e4e4ea8feef792550e4b391836acd0d651c56e571bbd2cb416151e
      rabbitmq: d0df46389c6dddc31afedee8c5f58ea5c480ef0147035c144d99c0dacb641b92
      redis: 9d548a12dda3876b1ca6038073d054617d1ca84ea5bb7407e62f52a1f1666d46
      redis_infra:
31377e023dd9d3ae69bd21684820396cc46950aee76ca2dde8367f88a373d41b
      s3storage:
        default: ae1804a12ffd4395b6b18d71c6ff04fb95bca6b2511c00656c20b571c01a16cb
    modules:
      bp-workers: ed7ae9b50be6530a9fe82af783d2f37e0b7d06a7dde72cf004c7ecb7260c3bb3
      calc: 0cef36f599debd41c34161fb8856802cb7b999d1134426e56bd84bced0ea72aa
      documents: cf79ad30b32233fedb8abf922b5a19a9bba0c01521a684e8d3634e9738509679
      events: f166b157a391a8f4fc017e261f14019b4180f1e847a867d4d5613bfe441d0224
      licensing: 02ebb28eb1c51831ddc640381e0a001c34e8e2f815fb63307d87d4690f1ad77c
      mnemoscheme:
53c756f13b0de4f6f3e39034b4ce4efeab60162199d303838043add81fcff9ac
      notifications:
997ee58487eaaf066b2995185ef3bd2c2870224964fc62482bd33ea17d04ef13
      om: 1aa3b4f9e07a0f5923feace4aa6628784920049e1a8cf5fa6570341f9f6dadf7
      portal: 7d6d532fdc98c77cae7df2fe80cfc422b7a85187765fc23043e2301f6eb1d208
      process: 73356d1805f7b8b1e6852a02ecdf5c084d885aa152068531db0f7d3eb972db7a
      rtdb: cb34a90e6ae79c14d9b42e5af2ece4a75965f08fb8bd6504bc7f57db471df917
      security: a0cc982a556535d0b2808e73f14170bf7d4b73bc4a62f2abae0fcaf63078e3e5
      sm: 7ab230ca2e76f81127eac68ec2fb4e60c031a288869bed3071cf5917b1de7c33
      udl: 96b4dc134849af0e9eef39fd3766b01f0822242a4a5243d6cf81aa868b46af83
      ziiot-docs: 27ffaea54629540ed56c22e5ad40750562fff792378258ce18a968121530b309
      ziiot-tests:
2884da08ab4e562f2bd44cba1eeb95a1bcbd0c7ad5819821841e7fe8fcf623bf
```

```
    platformVersion: 2.21.0

    trustedRootCa:
dd3cf76680e2a20552e951cf762228358b44cee950140fcb47703cdc09fa1300
```

4.12.3.3 Обновление паролей с помощью zifctl secrets recreate

Команда **zifctl secrets recreate** используется для массовой замены паролей и секретов инфраструктурных компонентов и модулей **Платформы ZIIoT** в различных вариантах развертывания. В зависимости от параметров, команда может выполнять различные действия.

Новые пароли применяются для инсталляции **Платформы ZIIoT** (синхронизируются с ней) по команде **zifctl sync**.

Для обеспечения смены паролей, включая пароли технических учетных записей, с определенной периодичностью для обеспечения задач ИБ необходимо:

1. Запустить команду инсталлятора **zifctl (zifctl secrets recreate)** для генерации новых паролей.

Примечание: новые пароли генерируются (файл с перечнем УЗ и зашифрованных паролей, см. Приложение 3. Список учетных записей Платформы ZIIoT) в файле параметров **env-secrets.yaml.**

2. Запустить команду **zifctl sync**, при выполнении которой пароли применяются для инсталляции **Платформы.** При первом запуске **zifctl sync** после смены паролей происходит проверка наличия файла **env-secrets-backup.yaml** ,а при его наличии запускается **hook,** начинающий смену паролей в платформе. При этом выполняются следующие действия

- создаётся секрет **zif-old-global-secrets**, содержащий старые и новые секреты. Удаляется после успешного завершения смены паролей;
- перед **helmfile** создаётся **job change-passwords**, который удаляется после успешного завершения. Данное задание выполняет действия(python/cli/http) с инфраструктурными сервисами, не поддерживающими автоматическую смену паролей из переменных окружения (cassandra, rabbitmq, keycloak, redis). При выполнении проверяет наличие установленного нового пароля, при ошибке авторизации подключается со старым паролем и меняет его на новый для указанной учетной записи.

Примечание: измененные пароли сохраняются в **configmaps/secretmaps**. Если выполнение команды завершилось неудачей из-за недоступности сервиса или ошибок авторизации с использованием нового и старого пароля, требуется перезапуск **zifctl sync** или ручное вмешательство.

Ручное вмешательство заключается в необходимости перезагрузки подов, обусловленной техническими особенностей Kubernetes. После такой перезагрузки изменения **configmaps/secretmaps** применяются для пода соответствующего сервиса

Результат: после завершения выполнения команды **zifctl sync** - сервисы **Платформы ZIIoT** используют новые пароли.

Для того, чтобы ограничить перечень сервисов, для которых требуется смена паролей в технических УЗ, платформенными или инфраструктурными сервисами, или вручную задаваемым перечнем, необходимо: запустить команду инсталлятора **zifctl** (**zifctl secrets recreate** с параметрами **--mod infra** или **--mod module** для платформенных или инфраструктурных сервисов).

Примечание: файл **--mod services.yaml** необходим для генерации паролей для списка из **services.yaml.**

Результат: генерируется список с новыми паролями только для платформенных или инфраструктурных (образы /infra) сервисов.

Общее примечание: Платформа ZIIoT продолжает работать в обычном режиме во время процедуры смены паролей.

4.13 Отладочные команды

При выявлении каких-то проблем с развертыванием Платформы на любом этапе работы системы, можно воспользоваться командами **write-values** и **template,** а также ключом выдачи отладочной информации **--verbose** и **–debug.**

Система развертывания **Платформы** в целом работает по следующей схеме (схема упрощенная, в ней не указаны различные вспомогательные скрипты на разных этапах и не указана система инициализации сервисов, запускаемая уже в кластере после загрузки манифестов):

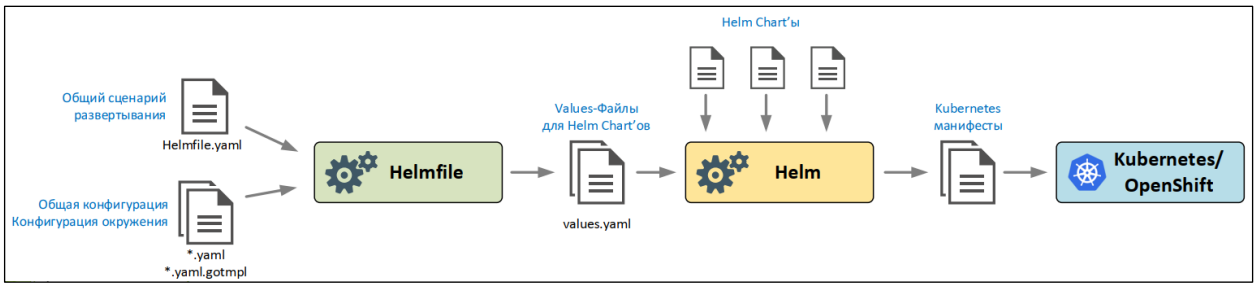

Рисунок 4.1 Отладочные команды

1) На основании файлов сценариев **Helmfile***, шаблонов конфигурации (**values***) и при участии доп. скриптов создаются параметры для **Helm Chart** (эту функцию выполняет утилита **Helmfile**).
2) Полученные параметры (**values**) передаются в **Helm Chart**, из которых генерируются манифесты для объектов **K8s** (эту функцию выполняет **Helm**).
3) Созданные манифесты загружаются в кластер **Kubernetes**, при необходимости с обновлением уже существующих (эту задачу тоже выполняет **Helm**).
4) Запускаются объекты заданий (**Job**) внутри кластера, которые выполняют начальную инициализацию развернутой инфраструктуры и сервисов (этот этап на схеме не указан).

В системе развертывания предусмотрены команды, позволяющие посмотреть генерируемые значения для **chart** и генерируемые манифесты на основании этих значений (аналогично командам **Helm**):

```
# Запись значений, полученных на первом этапе, которые затем должны быть переданы в Helm Chart. Значения записываются по пути, указанном в параметре --output-path

./zifctl write-values --env-path /path/to/env/config --age-key $KEY --output-path $OUTPUT_PATH


# Запись манифестов, сгенерированных Helm на основании переданных значений (результат второго этапа). Манифесты записываются по пути, указанном в параметре --output-path. Можно сгенерировать манифесты для всей установки или для модуля

./zifctl template --env-path /path/to/env/config --age-key $KEY --output-path $OUTPUT_PATH [--module rtdb ]
```

Количество выдаваемой на консоль (и в лог при использовании параметра **--log-path**) информации регулируется двумя опциональными параметрами **zifctl**:

- **--verbose —** включает отладочные сообщения из скриптов самой системы развертывания **zifctl** и вызываемых им.

- **--debug —** включает помимо отладочных сообщений системы развертывания еще и отладочные сообщения в **Helmfile** и **Helm** (а также в скрипте обертке).

4.13.1.1 restart-init-task

Перезапуск задачи **init-job** в случае возникновения проблем или зависания.

Важно:

Перезапуск возможен только для модулей из файла **service-list.yaml** и в статусе **enabled = True.** Перезапустить **init-job** инфраструктуры или отключенные модули данной командой нельзя!

Команда:

./**zifctl debug restart-init-task**

Обязательные ключи:

--env_path

--module

--server

--token

Рассмотрим для примера перезапуск задачи **init-job** для модуля **rtdb.**

Результат выполнения команды можно увидеть в логе zifctl. В частности, лог содержит информацию о том, что предыдущая init-job удалена.

```
~$ ./zifctl debug restart-init-task \
    --env-path /home/user/env_config/ \
    --module rtdb \
    --server https://12.34.56.78 \
    --token $MY_TOKEN

Status: Image is up to date for docker.idp.yc.ziiot.ru/infra/zifctl:feature-
platform01-44787
 2024-09-04 21:14:43,590 INFO    debug: Environment path: /var/deploy/env
 2024-09-04 21:4:43,590 INFO    debug: Working namespace: dev-develop
 2024-09-04 21:14:43,590 INFO    debug: Module name: rtdb
 2024-09-04 21:14:43,590 INFO    debug: Job name: zif-rtdb-module-init
 2024-09-04 21:14:43,590 INFO    debug: Ansible log verbosity: 2
 2024-09-04 21:14:43,673 INFO    debug: Found existed job: zif-rtdb-module-
init in namespace dev-develop
 2024-09-04 21:14:43,734 INFO    debug: Wait for job deletion...
 2024-09-04 21:14:45,761 INFO    debug: Job was deleted: zif-rtdb-module-init
 2024-09-04 21:14:45,772 INFO    debug: Render template init-job.yaml
 2024-09-04 21:14:45,816 INFO    debug: Create new job: zif-rtdb-module-init
```

4.13.1.2 show-defaults

Вывод значений по умолчанию для параметров развертывания. Сортировка списка в алфавитном порядке.

Для переопределения значений параметров, необходимо внести их в файл **env-values.yaml**.

Команда:

./zifctl debug show-defaults

У команды нет обязательных ключей.

Пример результата работы флаги приведен ниже. Он сокращен, так как является очень объемным.

```
~$ ./zifctl debug show-defaults

abacAuthorization:
  enabled: false
  importPolicies: true
  pip:
    keycloak:
      enabled: false
      rabbitmq:
        enabled: true
        loginEvents: false
    om:
      enabled: false
      kafka_enabled: false
appName: test_app
cassandra:
  auth: true
  commitLogSegmentSizeMB: 32
  commitLogSizeMB: 256
  contactPoints: ''
  defaultServiceName: zif-cassandra-headless
  installed: false
  metricsEnabled: false
  nodes: 1
  port: 9042
  replicationFactor: 1
  storageSizeMB: 1024
  tls:
    enabled: false
    force: false
    hostname: ''
    internode: all
    keystoreFile: /bitnami/cassandra/certs/stores/keystore.jks
    truststoreFile: /bitnami/cassandra/certs/stores/truststore.jks
  version: 3

------------Часть вывода удалена-----------------

fileStorage:
  anonymousUsername: filestorage_anonymous_user
  basePath: /files
healthcheck:
  livenessFailureThreshold: 3
  livenessInitialDelaySeconds: 60
  livenessPath: /health/liveness
  livenessPeriodSeconds: 10
  livenessSuccessThreshold: 1
  livenessTimeoutSeconds: 10
  readinessFailureThreshold: 3
  readinessInitialDelaySeconds: 60
  readinessPath: /health/readiness
  readinessPeriodSeconds: 10
```

```
  readinessSuccessThreshold: 1
  readinessTimeoutSeconds: 10

------------Часть вывода удалена------------------

init:
  ansibleCallbackPlugin: default
  ansibleHideKeycloakSecrets: true
  ansibleLogVerbosity: 2
  disabledTasks: []
  fullInitTaskList:
  - postgres
  - postgresGrant
  - postgresOwnership
  - keycloak
  - cassandra
  - redis
  - rabbitmq
  - debezium
  - portalSettings
  - rest
  - objectModel
  - custom
  - logging
  - s3storage
  - authorization
  - kafka
  - nifiMetrics
  longRetries: 60
  longRetriesDelay: 10
  shortRetries: 3
  shortRetriesDelay: 5
  tasks:
  - all

------------Часть вывода удалена------------------
```

4.13.1.3 show-config

Вывод текущей конфигурации Платформы (содержимое файла env-values.yaml). Сортировка по алфавиту.

Команда*:*

./zifctl debug show-config

Обязательные ключи:

--env-path

Пример:

```
~$ ./zifctl debug show-config --env-path /home/path_to/env_config/
```

Вывод команды похож на предыдущий и также является очень объемным.

4.13.1.4 show-tenant-id

Вывод ID тенанта из текущей конфигурации окружения и из развернутого экземпляра платформы.

Команда:

./zifctl debug show-tenant-id

Ключи:

--env-path - обязательный

--server

--token

В случае использования единственного ключа env_path, результатом выполнения команды будет вывод значения параметра tenantId из файла env-values.yaml

Пример выполнения команды с ключом env_path:

```
~$ ./zifctl debug show-tenant-id  --env-path ~/dev-develop-2.19-default/

Show ID of the tenant for configuration:
  env-values.yaml:       ziiot
```

При использовании ключей server и token (или установке значений переменных окружения ZIF_SERVER и ZIF_TOKEN), выводится значение параметра TENANT_ID из конфигурации zif-global-config. При этом идет обращение к значению параметра пространство имен и используются аргументы ключей server и token.

Пример выполнения команды с ключами server и token

```
~$ ./zifctl debug show-tenant-id  --env-path /home/path_to/env_config/   --
server https://12.34.56.78 --token $MY_TOKEN

Show ID of the tenant for configuration:
  env-values.yaml:       ziiot
  zif-global-config:     ziiot             from installation in k8s
```

4.13.1.5 scale-down-platform

Отключение стенда без удаления каких-либо сущностей: обнуляет количество подов во всех развертываниях (deployment) и наборах состояний (statefulset).

Примечание. Стенд включается командой **zifctl sync**.

Команда*:*

./zifctl debug scale-down-platform

Обязательные ключи:

--env-path

--server

--token

Пример результата работы команды. Сокращен.

```
~$ ./zifctl debug scale-down-platform  --env-path /home/path_to/env_config/ --
server https://12.34.56.78 --token $MY_TOKEN

 2024-09-04 21:41:20,928 INFO    kube: deployment.apps/zif-cm-context-functions
scaled
 2024-09-04 21:41:21,039 INFO    kube: deployment.apps/zif-cm-engine-mvel
scaled
 2024-09-04 21:41:21,173 INFO    kube: deployment.apps/zif-cm-metadata scaled
 2024-09-04 21:41:21,301 INFO    kube: deployment.apps/zif-dashboard scaled
 2024-09-04 21:41:21,415 INFO    kube: deployment.apps/zif-data-emulator scaled
 2024-09-04 21:41:21,545 INFO    kube: deployment.apps/zif-datainput scaled
 2024-09-04 21:41:21,672 INFO    kube: deployment.apps/zif-datalink scaled
 2024-09-04 21:41:21,804 INFO    kube: deployment.apps/zif-datalink-scripts
scaled
 2024-09-04 21:41:21,909 INFO    kube: deployment.apps/zif-datalink-xl scaled
 2024-09-04 21:41:22,053 INFO    kube: deployment.apps/zif-docs scaled
 2024-09-04 21:41:22,167 INFO    kube: deployment.apps/zif-document-archive
scaled
 2024-09-04 21:41:22,330 INFO    kube: deployment.apps/zif-events scaled
 2024-09-04 21:41:22,438 INFO    kube: deployment.apps/zif-events-integration
scaled
 2024-09-04 21:41:22,564 INFO    kube: deployment.apps/zif-export-nifi-
collectors scaled
 2024-09-04 21:41:22,721 INFO    kube: deployment.apps/zif-file-storage-default
scaled
 2024-09-04 21:41:22,835 INFO    kube: deployment.apps/zif-hierarchies scaled
 2024-09-04 21:41:22,932 INFO    kube: deployment.apps/zif-interface-connect
scaled
 2024-09-04 21:41:23,059 INFO    kube: deployment.apps/zif-interface-manager
scaled

------------Часть вывода удалена-------------------
```

4.14 Получение списка требуемых docker-образов для развертывания и export/import образов

Для запуска Платформы в кластере **Kubernetes** требуются **docker**-образы сервисов Платформы, образы всей устанавливаемой инфраструктуры (**Postgres**, **Cassandra**, **Kafka** и т.д.) и образ самой системы развертывания (**zifctl**).

В отличие от предыдущих релизов, образы с **Helm Chart** и копия **git**-репозитория не требуются (это все включено в образ **zifctl**).

Все необходимые **docker**-образы размещаются в репозитории компании **Zyfra**, который система развертывания использует по умолчанию: https://registry.dp.zyfra.com.

Платформу часто приходится развертывать в закрытых контурах, куда **docker**-образы приходится переносить вспомогательными средствами, поэтому в систему развертывания включены команды, позволяющие получить список требуемых образов и сгенерировать заготовки скриптов для их выгрузки и загрузки в другой репозиторий.

```
# Получение списка требуемых docker-образов. Формат вывода указывается в параметре
--output и может быть: table, list, json, yaml, csv  (для удобства интеграции с
инструментами для экспорта-импорта образов)
```

```
./zifctl image list --output table


# Получение заготовок скриптов для экспорта (pull-save) и импорта в другой репозиторий (load-tag-save). Скрипты сохраняются по пути --output-path
./zifctl image scripts --output-path /path/to/save/scripts


# Получение заготовок скриптов для экспорта (pull-save) и импорта в другой репозиторий (load-tag-save). Скрипты сохраняются по пути --output-path
./zifctl image scripts --output-path /path/to/save/scripts


# Вызывается команда zifctl image all для контейнера Zifctl
./zifctl image all --output-path
```

Результат:

- выводится объединенный список всех docker и не-docker артефактов используя данные из **service-list.yaml**, **chart-list.yaml** и **artifact-list.yaml**;
- формат вывода и остальные параметры **zifctl images** остаются неизменными;
- формат вывода - в консоль в формате yaml;
- есть возможность перенаправить вывод в файл **<name>.yaml** для получения валидного yaml файла.

Формат вызова

```
usage: zifctl images all [OPTIONS] [--help]
Generate list all artifacts
required arguments:
  Specify this options or declare corresponding environment variables
optional arguments:
  -r REGISTRY, --registry REGISTRY
                        Custom registry address. http format
```

Примечание: дополнительный флаг **-r** задает адреса репозитория **nexus**.

По умолчанию https://nexus.idp.yc.ziiot.ru/repository.

Команда **Zifctl image all** доступна также для Инсталлятора Приложений.

4.15 Поддержка политик ABAC для Приложений Платформы

4.15.1 Размещение политик и ролей внутри образа Приложения

Политики размещаются в виде ABAC файлов в директории **init/module/data/abac/policies** в Zifctl. Роли размещаются в виде ABAC файлов в init/module/data/abac/roles.

Примечание: политика или роль для каждого модуля представлена в виде каталога с названием сервиса внутри которой, находится файл в форматах **json** или **jina template**.

Результат: при установке/синхронизации Приложения посредством Zifctl базовые политики и роли применяются.

Дополнительные действия с базовыми политиками и ролями при установке/синхронизации Приложения посредством Zifctl:

- добавление файлов в форматах json или jinja непосредственно при сборке образа того или иного приложения;

Примечание: таким образом папка с политиками (ролями аналогно, только в формате json) будет собираться и для определенной версии образа и закреплена за определенным тегом сборки.

- удаление устаревших политик через файл **init/module/data/abac/policies/deprecated-policies.json**;

Примечание: для ролей такой механизм не предусмотрен.

- в директории с ролями также можно размещать файл с конфигурацией, который используется для формирования композитных ролей **clients_configuration.json**.

Для правильной работы ABAC авторизации в контейнере Zifctl необходимо настроить несколько уровней конфигурации. Роль будет отрабатываться при выполнении условий:

1. Конфигурация **env-values.yaml** (целиком для Приложения):

```
abacAuthorization:
  enabled: True  # default False
  importPolicies: True  # default True
```

Уровень **env-values.yaml**:

- проверяется **abacAuthorization.enabled**;
- если False, ABAC не активируется;
- importPolicies - должна быть True;

2. Конфигурация **service-list-patch.yaml** (для конкретного модуля/сервиса):

```
modules:
  - <module_name>:
      importRoles: true # default true
      services:
        - <service_name>:
          abac:
            enabled: True # Default: True
```

Уровень **service-list-patch.yaml**:

- проверяется importRoles для модуля;
- проверяется abac.enabled для сервиса;
- проверяется importRoles для сервиса;

3. Конфигурация **service-list.yaml** (для конкретного модуля/сервиса):

```
modules:
  - <module_name>:
      services:
        - <service_name>:
            abac:
              enabled: True
              importRoles: true
```

Уровень **service-list.yaml**:

- проверяется abac.enabled;
- проверяется importRoles.

4.15.2 Размещение политик и ролей в файле env-config директории при установке Приложения

Политики и роли размещаются в файле **env-config** в каталогах (папках) **env-config/abac-policy**, **env-config/abac-roles**.

Результат: при установке/синхронизации Приложения посредством Zifctl применяются дополнительные политики и роли.

Примечание: при импорте политик с одинаковыми идентификаторами (уже применённых в п. 4.15.1) - новые политики заменяют первоначальные.

5 Справочник команд zifctl

Список команд инсталлятора **zifctl** представлен в Таблица 5.1.

Таблица 5.1. Справочник команд zifctl

Название команды	Описание
Основные команды	
init	Создание (подготовка) новой конфигурации развертывания
sync (deploy)	Развертывание экземпляра Платформы ZIIoT (выполнение синхронизации платформы ZIIoT)
clean (clear)	Полное удаление экземпляра Платформы из пространства имен (включая PVC, jobs, etc)
delete (remove)	Удаление экземпляра Платформы из пространства имен (с сохранением PVC)
Служебные и информационные команды	
get-wrapper (wrapper)	Команда используется для получения и управления оберткой, которая обеспечивает дополнительный функционал и расширенные возможности работы в zifctl
env	Выводит все переменные среды, используемые zifctl, со значениями по умолчанию/определенными пользователем

images (image)	Просмотр версий образов образов и чартов. Генерация скриптом для импорта и экспорта образов
secrets (secret)	Работа с секретами окружения
version	Получение версии zifctl
Отладочные команды	
debug	Сборник полезных подкоманд
template (templates)	Генерация манифестов Kubernetes для отладки
write-values (values)	Создание шаблонов Helm для отладки
build-helmfile	Создание сборочного файла helmfile
Команды переноса	
postgres (pg)	Миграция postgresql поставляемого экземпляра Платформы ZIIoT
Команды документирования	
doc (document)	Генерация документации

Примечание: при работе с командами используется набор подкоманд и обязательных аргументов. Если аргумент не задается явно, то система пытается взять значение из стандартных переменных окружения в соответствии с Таблица 5.1. В случае, если система не находит значения ни в аргументе, ни в переменной окружения, выдается ошибка.

5.1 Основные команды zifctl

5.1.1 zifctl init и zifctl init-workshop

Команда **zifctl init** используется для инициализации или подготовки окружения для дальнейшей работы с **zifctl**. Список флагов **zifctl init** представлено в Таблица 5.2.

Таблица 5.2. Флаги zifctl init

Флаг	Переменная окружения	Описание
Обязательные параметры		
--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией развертывания
--age-key	$ZIF_AGE_KEY	Ключ, который используется для шифрования секретов (генерируется при помощи утилиты age)
Прочие параметры		
--registry URL	$ZIF_REGISTRY	Имя репозитория с образами для развертывания (по умолчанию: http://docker.idp.yc.ziiot.ru)
--registry-username SECRET	$ZIF_REGISTRY_USERNAME	Имя пользователя для входа в репозиторий с образами. В зашифрованном виде хранится в файле env-secrets.yaml
--registry-password SECRET	$ZIF_REGISTRY_PASSWORD	Пароль для входа в репозиторий с образами. В зашифрованном виде хранится в файле env-secrets.yaml

--server URL	$ZIF_SERVER	Адрес API кластера Kubernetes/OpenShift
--token SECRET	$ZIF_TOKEN	Токен учетной записи сервиса OpenShift/Kubernetes для доступа к API
--type [instance|tenant|shared-infra|app]	$ZIF_CONFIG_TYPE	Тип развертывания: 'instance' (по умолчанию, ziiot+infra), 'tenant' (только ziiot), 'shared-infra' (только infra), 'app' (приложение) (по умолчанию: instance)
--app-name STRING	$ZIF_APP_NAME	Имя приложения (при типе развертывания 'приложения')
--hostname URL	$ZIF_HOSTNAME	Базовое доменное имя для доступа к экземпляру платформы (по этому имени будет доступен портал, а сервисы будут иметь пути вроде https://<base-hostname>/<service-name> (ВАЖНО: указывается не ссылка URL, а именно hostname)
--namespace STRING	$ZIF_NAMESPACE	Задает пространство имен в кластере Kubernetes, которое будет использоваться при установке Платформы
--sa STRING	$ZIF_SERVICE_ACCOUNT	Сервисный аккаунт, который будет использоваться для установки Платформы (по умолчанию: '<namespace>-sa'). Для запуска инфраструктурных контейнеров, требуется аккаунт с повышенными привилегиями
--storage STRING	$ZIF_STORAGE	Класс, определяющий параметры Persistent Volume Claims (PVC) используемых в сервисах инфраструктуры (по умолчанию: используется дефолтный SC кластера)
--tenant STRING	$ZIF_TENANT	Идентификатор тенанта (обязателен при типе развертывания тип== 'tenant'))
--tenant-name STRING	$ZIF_TENANT_NAME	Имя тенанта, если оно не указано, будет создано на основе идентификатора тенанта (--tenant)

--infra-ns STRING	$ZIF_INFRA_NS	Пространство имен, в котором находится общая инфраструктура (требуется, если тип == 'tenant')
--infra-ha	$ZIF_INFRA_HA	Развертывание инфраструктуры в конфигурации высокой доступности (High Available: поднимается по несколько подов каждого инфраструктурного сервиса) (для типа развертывания 'shared-infra' по умолчанию: `true`)
-f, --force	$ZIF_FORCE	Перезаписывает конфигурацию окружения, если она уже в директории
-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
--help		Используется для получения справочной информации о командах и параметрах zifctl

Команда **zifctl init-workshop** предназначена для подготовки рабочей среды, включая настройку необходимых компонентов, конфигурацию окружения и инициализацию базовых сервисов. Список флагов **zifctl init-workshop** представлено в Таблица 5.3.

Таблица 5.3. Флаги zifctl init-workshop

Флаг	Переменная окружения	Описание
Обязательные параметры		
--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией развертывания
--age-key	$ZIF_AGE_KEY	Ключ, который используется для шифрования секретов (генерируется при помощи утилиты age)
--server URL	$ZIF_SERVER	Адрес API кластера Kubernetes/OpenShift
--token SECRET	$ZIF_TOKEN	Токен учетной записи сервиса OpenShift/Kubernetes для доступа к API
--corp-center-ns STRING	$ZIF_CORP_CENTER_NS	Пространство имен корпоративного центра kubernetes для текущего экземпляра TSDS workshop
--data-source-name STRING	$ZIF_DATA_SOURCE_NAME	Имя источника данных для текущего экземпляра TSDS workshop
Прочие параметры		

--registry URL	$ZIF_REGISTRY	Имя репозитория с образами для развертывания (по умолчанию: http://docker.idp.yc.ziiot.ru)
--registry-username SECRET	$ZIF_REGISTRY_USERNAME	Имя пользователя для входа в репозиторий контейнеров, которое нужно указать в файле env-secrets.yaml
--registry-password SECRET	$ZIF_REGISTRY_PASSWORD	Пароль для входа в репозиторий контейнеров (для развертывания) задается в файле env-values.yaml
--hostname URL	$ZIF_HOSTNAME	Базовое доменное имя для доступа к экземпляру платформы (по этому имени будет доступен портал, а сервисы будут иметь пути вроде https://<base-hostname>/<service-name> (ВАЖНО: указывается не ссылка URL, а именно hostname)
--namespace STRING	$ZIF_NAMESPACE	Задает пространство имен в кластере Kubernetes, которое будет использоваться при установке Платформы
--sa STRING	$ZIF_SERVICE_ACCOUNT	Сервисный аккаунт, который будет использоваться для установки Платформы (по умолчанию: '<namespace>-sa'). Для запуска инфраструктурных контейнеров, требуется аккаунт с повышенными привилегиями
--storage STRING	$ZIF_STORAGE	Класс, определяющий параметры Persistent Volume Claims (PVC) используемых в сервисах инфраструктуры (по умолчанию: используется дефолтный SC кластера)
--tenant STRING	$ZIF_TENANT	Идентификатор тенанта (обязателен при типе развертывания тип== 'tenant'))
--tenant-name STRING	$ZIF_TENANT_NAME	Имя тенанта, если оно не указано, будет создано на основе идентификатора тенанта (--tenant)
--infra-ns STRING	$ZIF_INFRA_NS	Пространство имен, в котором находится общая

		инфраструктура (требуется, если тип == 'tenant')
--infra-ha	$ZIF_INFRA_HA	Развертывание инфраструктуры в конфигурации высокой доступности (High Available: поднимается по несколько подов каждого инфраструктурного сервиса) (для типа развертывания 'shared-infra' по умолчанию: `true`)
-f, --force	$ZIF_FORCE	Перезаписывает конфигурацию, если она уже существует
--modules MULTIPLE	$ZIF_MODULES	Список модулей, из которых можно прочитать секреты (по умолчанию: rtdb)
-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
--help		Используется для получения справочной информации о командах и параметрах zifctl

5.1.2 zifctl sync (deploy)

Команда **zifctl sync** используется для синхронизации состояния локальной конфигурации или данных с удаленным источником (например, кластером, репозиторием, облачной платформой). Список флагов **zifctl sync** представлено в Таблица 5.4.

Таблица 5.4. Флаги zifctl sync (deploy)

Флаг	Переменная окружения	Описание
Обязательные параметры		
--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией окружения
--age-key	$ZIF_AGE_KEY	Ключ, который используется для шифрования секретов (генерируется при помощи утилиты age)
--server URL	$ZIF_SERVER	Адрес API кластера Kubernetes/OpenShift
--token SECRET	$ZIF_TOKEN	Токен учетной записи сервиса OpenShift/Kubernetes для доступа к API
Прочие параметры		

--init-log-verbosity	$ZIF_INIT_LOG_VERBOSITY	Уровень логирования для ANSIBLE заданий, которые выполняются при инициализации контейнеров (по умолчанию: 1; 1<=x<=5)
--show-init-log-keycloak-secrets	$ZIF_SHOW_INIT_LOG_KEYCLOAK_SECRETS	Показывает пароли пользователей keycloak и клиентские секреты в логах заданиях файлов инициализации
--module STRING	$ZIF_MODULE	Команда выполняется только для указанного модуля
--hide-results-page	$ZIF_HIDE_RESULTS_PAGE	Не выводить в консоль и в лог имена и пароли от инфраструктурных сервисов после успешной синхронизации
-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
--help		Используется для получения справочной информации о командах и параметрах zifctl

5.1.3 zifctl clean (clear)

Команда **zifctl clean** (**clear)** используется для очистки или удаления временных файлов, кэшей, устаревших ресурсов или других нежелательных артефактов, связанных с работой **zifctl**. Список флагов **zifctl clean** (**clear)** представлено в Таблица 5.5.

Таблица 5.5. Флаги zifctl clean (clear)

Флаг	Переменная окружения	Описание
Обязательные параметры		
--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией окружения
--age-key	$ZIF_AGE_KEY	Ключ, который используется для шифрования секретов (генерируется при помощи утилиты age)
--server URL	$ZIF_SERVER	Адрес API кластера Kubernetes/OpenShift
--token SECRET	$ZIF_TOKEN	Токен учетной записи сервиса OpenShift/Kubernetes для доступа к API
Прочие параметры		
--hide-results-page	$ZIF_HIDE_RESULTS_PAGE	Не выводить в консоль и в лог имена и пароли от инфраструктурных сервисов после успешной синхронизации

-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
--help		Используется для получения справочной информации о командах и параметрах zifctl

5.1.4 zifctl delete

Команда **zifctl delete** является общей командой, которая, как правило, используется для удаления каких-либо ресурсов или объектов, управляемых **zifctl**. Тип ресурса, который будет удален, определяется аргументами и опциями, переданными команде. Список флагов **zifctl delete** представлено в Таблица 5.6.

Таблица 5.6. Флаги zifctl delete

Флаг	Переменная окружения	Описание
Обязательные параметры		
--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией окружения
--age-key	$ZIF_AGE_KEY	Ключ, который используется для шифрования секретов (генерируется при помощи утилиты age)
--server URL	$ZIF_SERVER	Адрес API кластера Kubernetes/OpenShift
--token SECRET	$ZIF_TOKEN	Токен учетной записи сервиса OpenShift/Kubernetes для доступа к API
Прочие параметры		
-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
--help		Используется для получения справочной информации о командах и параметрах zifctl

5.2 Служебные и информационные команды zifctl

5.2.1 zifctl get-wrapper (wrapper)

Команда **zifctl get-wrapper** используется для получения информации о wrapper-конфигурации в **zifctl**. Wrapper-конфигурация определяет параметры запуска и окружения для компонентов платформы. Список флагов **zifctl get-wrapper** представлено в Таблица 5.7.

Таблица 5.7. Флаги zifctl get-wrapper (wrapper)

Флаг	Переменная окружения	Описание
Параметры		

-s, --source	$ZIF_SOURCE	Возвращает 'zifctl wrapper' в виде python-скрипта вместо бинарного файла
-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
--help		Используется для получения справочной информации о командах и параметрах zifctl

5.2.2 zifctl env

Команда **zifctl env** используется для управления окружением платформы **zifctl**. Она позволяет просматривать, создавать и управлять переменными окружения, которые используются при развертывании и работе компонентов платформы. Список флагов **zifctl env** представлено в Таблица 5.8.

Таблица 5.8. Флаги zifctl env

Флаг	Переменная окружения	Описание
Параметры		
-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
--help		Используется для получения справочной информации о командах и параметрах zifctl

5.2.3 zifctl images (image)

Команда **zifctl images** используется для управления Docker-образами в **zifctl**. Она позволяет просматривать, загружать и управлять образами, которые используются при развертывании компонентов платформы. Команда **zifctl images** состоит из следующих подкоманд:

- **zifctl images list** - показывает список доступных образов;
- **zifctl images scripts** - управляет скриптами, связанными с Docker-образами;
- **zifctl images charts** - работает с Helm-чартами для Docker-образов;
- **zifctl images all** – команда для управления всеми образами одновременно.

Список команд **zifctl images** с их флагами представлено в Таблица 5.9.

Таблица 5.9. Команды zifctl images с флагами

Команда	Флаг	Переменная окружения	Описание
zifctl images list	Параметры		
	-o, --output	$ZIF_OUTPUT	Форматы вывода: таблица (table), список (list), csv, yaml, json (по умолчанию: таблица (table))

	-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl
zifctl images charts	Параметры		
	-o, --output	$ZIF_OUTPUT	Форматы вывода: таблица (table), список (list), csv, yaml, json (по умолчанию: таблица (table))
	-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl
zifctl images scripts	Обязательные параметры		
	--output-path PATH	$ZIF_OUTPUT_PATH	Директория, в котором сохраняются скрипты для импорта (import-push.sh) \экспорта образов (pull-export-images.sh)
	Прочие параметры		
	-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl
zifctl images all	Параметры		
	-r, --registry REGISTRY	$ZIF_REGISTRY	Адрес, по которому хранятся вспомогательные утилиты Zif.Interface.Agent.exe, zif.nifi.processors.tar и т.д. (по умолчанию: https://nexus.idp.yc.ziiot.ru/repository)
	-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl

5.2.4 zifctl secrets (secret)

Команда **zifctl secrets (secret)** используется для управления секретами в **zifctl**. Она позволяет создавать, просматривать и управлять секретами, которые используются для хранения конфиденциальной информации, такой как ключи API, пароли и сертификаты. Команда **zifctl secrets (secret)** ссостоит из следующих подкоманд:

- **zifctl secrets enc**- команда для шифрования секретов;

- **zifctl secrets dec**- команда для расшифровки секретов;
- **zifctl secrets show** - команда для просмотра секретов;
- **zifctl secrets set**– команда для установки нового значения отдельного секрета;
- **zifctl secrets delete** – команда удаляет секрет;
- **zifctl secrets new-age-key** - команда для генерации нового ключа для шифрования секретов;
- **zifctl secrets recreate** - команда для массового пересоздания секретов.

Список команд **zifctl secrets (secret)** с их флагами представлено в Таблица 5.10.

Таблица 5.10. Команды zifctl secrets (secret) с флагами

Команда	Флаг	Переменная окружения	Описание
zifctl secrets enc	Обязательные параметры		
	--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией окружения
	--age-key	$ZIF_AGE_KEY	Ключ, который используется для шифрования секретов (генерируется при помощи утилиты age)
	Прочие параметры		
	--module STRING	$ZIF_MODULE	Команда выполняется только для указанного модуля
	-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl
zifctl secrets dec	Обязательные параметры		
	--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией окружения
	--age-key	$ZIF_AGE_KEY	Ключ, который используется для шифрования секретов (генерируется при помощи утилиты age)
	Прочие параметры		
	--module STRING	$ZIF_MODULE	Команда выполняется только для указанного модуля
	-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки

	--help		Используется для получения справочной информации о командах и параметрах zifctl
zifctl secrets show	Обязательные параметры		
	--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией окружения
	--age-key	$ZIF_AGE_KEY	Ключ, который используется для шифрования секретов (генерируется при помощи утилиты age)
	Прочие параметры		
	--module STRING	$ZIF_MODULE	Команда выполняется только для указанного модуля
	-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl
zifctl secrets set	Позиционные аргументы		
	SECRET-NAME		Имя секрета
	SECRET-VALUE		Новое секретное значение
	Обязательные параметры		
	--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией окружения
	--age-key	$ZIF_AGE_KEY	Ключ, который используется для шифрования секретов (генерируется при помощи утилиты age)
	Прочие параметры		
	-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl
zifctl secrets delete	Позиционные аргументы		
	SECRET-NAME		Имя секрета
	Обязательные параметры		
	--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией окружения

	--age-key	$ZIF_AGE_KEY	Ключ, который используется для шифрования секретов (генерируется при помощи утилиты age)
	Прочие параметры		
	-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl
zifctl secrets new-age-key	Параметры		
	-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl
zifctl secrets recreate	Обязательные параметры		
	--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией окружения
	--age-key	$ZIF_AGE_KEY	Ключ, который используется для шифрования секретов (генерируется при помощи утилиты age)
	Прочие параметры		
	--server URL	$ZIF_SERVER	Адрес API кластера Kubernetes/OpenShift
	--token SECRET	$ZIF_TOKEN	Токен учетной записи сервиса OpenShift/Kubernetes для доступа к API
	--scope [infra|module|all]	$ZIF_SECRETS_RECREATE_SCOPE	Указывает, к каким секретам применяется команда (по умолчанию: all)
	--exclude MULTIPLE	$ZIF_SECRETS_EXCLUDE_LIST	Список секретов, которые нельзя раскрывать
	-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl

5.2.5 zifctl version

Команда **zifctl version** используется для получения информации о версии **zifctl** и его компонентов. Список флагов **zifctl version** представлено в Таблица 5.11.

Таблица 5.11. Флаги zifctl version

Флаг	Переменная окружения	Описание
Параметры		
-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
--help		Используется для получения справочной информации о командах и параметрах zifctl

5.3 Отладочные команды

5.3.1 zifctl debug

Команда **zifctl debug** предназначена для получения информации, полезной для отладки проблем с **zifctl.**

При выполнении команды **zifctl debug** доступен следующий набор подкоманд:

- **zifctl restart-init-task** - команда перезапускает процесс инициализации задач в системе;
- **zifctl show-defaults** - отображает все значения по умолчанию в **zifctl**;
- **zifctl show-config** - отображает текущую конфигурацию **zifctl**;
- **zifctl show-tenant-id** - отображает идентификатор текущего тенента (tenant) в **zifctl**;
- **zifctl scale-down-platform** - команда для масштабирования платформы.

Список команд **zifctl debug** с их флагами представлено в Таблица 5.12.

Таблица 5.12. Команды zifctl debug с флагами

Команда	Флаг	Переменная окружения	Описание
zifctl restart-init-task	Обязательные параметры		
	--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией окружения
	--server URL	$ZIF_SERVER	Адрес API кластера Kubernetes/OpenShift
	--token SECRET	$ZIF_TOKEN	Токен учетной записи сервиса OpenShift/Kubernetes для доступа к API
	--module STRING	$ZIF_MODULE	Команда выполняется только для указанного модуля
	Прочие параметры		

	--init-log-verbosity	$ZIF_INIT_LOG_VERBOSITY	Уровень логирования для ANSIBLE заданий, которые выполняются при инициализации контейнеров (по умолчанию: 1; 1<=x<=5)
	--show-init-log-keycloak-secrets	$ZIF_SHOW_INIT_LOG_KEYCLOAK_SECRETS	Показывает пароли пользователей keycloak и клиентские секреты в логах заданиях файлов инициализации
	--job-image JOB_IMAGE	$ZIF_JOB_IMAGE	Версия образа Zifctl для запуска созданных заданий (по умолчанию == текущий zifctl)
	-v, --verbose	$ZIF_VERBOSE	Включить подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включите подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl
zifctl debug show-defaults	Параметры		
	-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl
zifctl debug show-config	Обязательные параметры		
	--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией окружения
	Прочие параметры		
	-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl
zifctl debug show-tenant-id	Обязательные параметры		
	--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией окружения
	Прочие параметры		
	--server URL	$ZIF_SERVER	Адрес API кластера Kubernetes/OpenShift

	--token SECRET	$ZIF_TOKEN	Токен учетной записи сервиса OpenShift/Kubernetes для доступа к API
	-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl
zifctl debug scale-down-platform	Обязательные параметры		
	--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией окружения
	--server URL	$ZIF_SERVER	Адрес API кластера Kubernetes/OpenShift
	--token SECRET	$ZIF_TOKEN	Токен учетной записи сервиса OpenShift/Kubernetes для доступа к API
	Прочие параметры		
	-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl

Примечание: флаги имеют наборы обязательных ключей. Аргументы ключей в первую очередь задаются явно. Если ключ (или его аргумент) не заданы явно, то система берет значения аргументов из переменных окружения (файл env-values.yaml). В случае, если система не находит обязательный ключ, выдается ошибка.

5.3.2 zifctl template (templates)

Команда **zifctl template** представляет собой инструмент для работы с шаблонами в **zifctl**. Список флагов **zifctl template** представлено в Таблица 5.13.

Таблица 5.13. Флаги zifctl template

Флаг	Переменная окружения	Описание
Обязательные параметры		
--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией окружения
--age-key	$ZIF_AGE_KEY	Ключ, который используется для шифрования секретов (генерируется при помощи утилиты age)

--output-path PATH	$ZIF_OUTPUT_PATH	Директория, в котором сохраняются скрипты для импорта (import-push.sh) \экспорта образов (pull-export-images.sh)
Прочие параметры		
--init-log-verbosity	$ZIF_INIT_LOG_VERBOSITY	Уровень логирования для ANSIBLE заданий, которые выполняются при инициализации контейнеров (по умолчанию: 1; 1<=x<=5)
--show-init-log-keycloak-secrets	$ZIF_SHOW_INIT_LOG_KEYCLOAK_SECRETS	Показывает пароли пользователей keycloak и клиентские секреты в логах заданиях файлов инициализации
--server URL	$ZIF_SERVER	Адрес API кластера Kubernetes/OpenShift
--token SECRET	$ZIF_TOKEN	Токен учетной записи сервиса OpenShift/Kubernetes для доступа к API
--module STRING	$ZIF_MODULE	Команда выполняется только для указанного модуля
-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
--help		Используется для получения справочной информации о командах и параметрах zifctl

5.3.3 zifctl write-values

Команда **zifctl write-values** является важным инструментом для записи значений в **zifctl**. Список флагов **zifctl write-values** представлено в Таблица 5.14.

Таблица 5.14. Флаги zifctl write-values

Флаг	Переменная окружения	Описание
Обязательные параметры		
--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией окружения
--age-key	$ZIF_AGE_KEY	Ключ, который используется для шифрования секретов (генерируется при помощи утилиты age)
--output-path PATH	$ZIF_OUTPUT_PATH	Директория, в котором сохраняются скрипты для

		импорта (import-push.sh) \экспорта образов (pull-export-images.sh)
Прочие параметры		
--init-log-verbosity	$ZIF_INIT_LOG_VERBOSITY	Уровень логирования для ANSIBLE заданий, которые выполняются при инициализации контейнеров (по умолчанию: 1; 1<=x<=5)
--show-init-log-keycloak-secrets	$ZIF_SHOW_INIT_LOG_KEYCLOAK_SECRETS	Показывает пароли пользователей keycloak и клиентские секреты в логах заданиях файлов инициализации
--server URL	$ZIF_SERVER	Адрес API кластера Kubernetes/OpenShift
--token SECRET	$ZIF_TOKEN	Токен учетной записи сервиса OpenShift/Kubernetes для доступа к API
--module STRING	$ZIF_MODULE	Команда выполняется только для указанного модуля
-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
--help		Используется для получения справочной информации о командах и параметрах zifctl

5.3.4 zifctl build-helmfile

Команда **zifctl build-helmfile** представляет собой инструмент для сборки Helmfile конфигураций в **zifctl**. Список флагов **zifctl build-helmfile** представлено в Таблица 5.15.

Таблица 5.15. Флаги zifctl build-helmfile

Флаг	Переменная окружения	Описание
Обязательные параметры		
--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией окружения
--age-key	$ZIF_AGE_KEY	Ключ, который используется для шифрования секретов (генерируется при помощи утилиты age)
--output-path PATH	$ZIF_OUTPUT_PATH	Директория, в котором сохраняются скрипты для импорта (import-push.sh) \экспорта образов (pull-export-images.sh)

Прочие параметры		
--server URL	$ZIF_SERVER	Адрес API кластера Kubernetes/OpenShift
--token SECRET	$ZIF_TOKEN	Токен учетной записи сервиса OpenShift/Kubernetes для доступа к API
-v, --verbose	$ZIF_VERBOSE	Включает подробный вывод команд
-vv, --debug	$ZIF_DEBUG	Включает подробный вывод команд и вывод helmfile/helm debug для отладки
--help		Используется для получения справочной информации о командах и параметрах zifctl

5.4 Команды переноса

Команда **zifctl postgres (pg)** представляет собой инструмент для управления PostgreSQL в **zifctl.**

При выполнении команды **zifctl postgres (pg)** доступен следующий набор подкоманд:

- **zifctl postgres migration-start** - обновляет базы данных PostgreSQL до новой версии;
- **zifctl postgres migration-cancel** - останавливает запуск миграции;
- **zifctl postgres migration-clean** - очищает все объекты миграции, включая PVC-файлы;
- **zifctl postgres migration-get-logs** - сохранение журналов на диск.

Список команд **zifctl postgres (pg)** с их флагами представлено в Таблица 5.16.

Таблица 5.16. Команды zifctl postgres (pg) с флагами

Команда	Флаг	Переменная окружения	Описание
zifctl postgres migration-start	Обязательные параметры		
	--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией окружения
	--server URL	$ZIF_SERVER	Адрес API кластера Kubernetes/OpenShift
	--token SECRET	$ZIF_TOKEN	Токен учетной записи сервиса OpenShift/Kubernetes для доступа к API
	-s, --logs-pvc-size SIZE	$ZIF_LOGS_PVC_SIZE	Устанавливает размер файла logs в формате PVC (разрешает использовать единицы измерения "Mi" или "Gi")

	-m, --mem-size MEM_SIZE	$ZIF_MEM_SIZE	Устанавливает предельный объем памяти в гигабайтах
	Прочие параметры		
	-p, --postgres-version	$ZIF_POSTGRES_VERSION	Устанавливает версию PostgreSQL для миграции (по умолчанию: stolon-14)
	-v, --verbose	$ZIF_VERBOSE	Включить подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включите подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl
zifctl postgres migration-cancel	Обязательные параметры		
	--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией окружения
	--server URL	$ZIF_SERVER	Адрес API кластера Kubernetes/OpenShift
	--token SECRET	$ZIF_TOKEN	Токен учетной записи сервиса OpenShift/Kubernetes для доступа к API
	Прочие параметры		
	-v, --verbose	$ZIF_VERBOSE	Включить подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включите подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl
zifctl postgres migration-clean	Обязательные параметры		
	--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией окружения
	--server URL	$ZIF_SERVER	Адрес API кластера Kubernetes/OpenShift
	--token SECRET	$ZIF_TOKEN	Токен учетной записи сервиса OpenShift/Kubernetes для доступа к API
	-f, --force	$ZIF_FORCE	Перезаписывает конфигурацию, если она уже существует

	Прочие параметры		
	-v, --verbose	$ZIF_VERBOSE	Включить подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включите подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl
zifctl postgres migration-get-logs	Обязательные параметры		
	--env-path	$ZIF_ENV_PATH	Абсолютный или относительный путь к директории с конфигурацией окружения
	--server URL	$ZIF_SERVER	Адрес API кластера Kubernetes/OpenShift
	--token SECRET	$ZIF_TOKEN	Токен учетной записи сервиса OpenShift/Kubernetes для доступа к API
	--out-path FILE_PATH	$ZIF_OUT_PATH	Путь к локальному файлу для сохранения журнала
	Прочие параметры		
	-v, --verbose	$ZIF_VERBOSE	Включить подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включите подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl

5.5 Команды документирования

Команды **zifctl doc** предоставляют инструменты для работы с документацией и переменными окружения в **zifctl.**

При выполнении команды **zifctl doc** доступен следующий набор подкоманд:

- **zifctl doc env-values** - команда для просмотра и управления переменными окружения;
- **zifctl doc help** - команда для получения справочной информации о других командах **zfctl**.

Список команд **zifctl doc** с их флагами представлено в Таблица 5.16.

Таблица 5.17. Команды zifctl doc с флагами

Команда	Флаг	Переменная окружения	Описание
	Обязательные параметры		

zifctl doc env-values	--out-path FILE_PATH	$ZIF_OUT_PATH	Путь к локальному файлу для сохранения журнала
	Прочие параметры		
	-f, --format	$ZIF_OUTPUT_FORMAT	Выходной формат (по умолчанию: md)
	-v, --verbose	$ZIF_VERBOSE	Включить подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включите подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl
zifctl doc help	Параметры		
	-v, --verbose	$ZIF_VERBOSE	Включить подробный вывод команд
	-vv, --debug	$ZIF_DEBUG	Включите подробный вывод команд и вывод helmfile/helm debug для отладки
	--help		Используется для получения справочной информации о командах и параметрах zifctl

6 Руководство по обновлению

Специальные требования к процессу обновления:

1) Для некоторых топиков `debezium` заданы параметры в `services/kafka-topics-list.yaml`.

   При обновлении с предыдущего релиза и нестандартной конфигурации статичных топиков `debezium`, необходимо их отразить в `services/service-list-patch.yaml`. Количество партиций в топиках нельзя уменьшать автоматически. Управление доступно только для топиков начинающих с `<tenantid>__`.

2) Удален параметр `keylocak.baseUri`, при обновлении с предыдущего релиза его необходимо удалить из `env-values.yaml`.

7 Генерация документа (zifctl doc) env-values - конфигурирования переменных

Добавлена команда `zifctl doc`, вместе с подкомандой (пока единственной) `env-values`, которая и генерирует документацию по env-values.yaml.

Обязательный параметр: `--ouput=path` - папка в которую сохранять документацию. Необязательный: `--format` - принимает значения `md/html`, по умолчанию md.

Пример команды:

1) Запуск из `docker`-образа:

```
docker run --rm -it \
-v "<путь для сгенерированных файлов документации>":/output-path \
docker-group.idp.yc.ziiot.ru/infra/zifctl:2.16.0 \
doc env-values --output-path=/output-path
```

По указанному пути будет сгенерирован файл `env-values.md` в формате `markdown`.

По указанному пути будет сгенерирован файл `env-values.html` в формате `html`:

```
docker run --rm -it \
-v "<путь для сгенерированных файлов документации>":/output-path \
docker-group.idp.yc.ziiot.ru/infra/zifctl:2.16.0 \
doc env-values --output-path=/output-path --format html
```

2) Запуск из враппера (wrapper) zifctl:

```
docker pull docker.idp.yc.ziiot.ru/infra/zifctl:2.16.0
docker run --rm  docker.idp.yc.ziiot.ru/infra/zifctl:2.16.0 get-wrapper > zifctl
```

```
./zifctl doc env-values --output-path .
```

По указанному пути будет сгенерирован файл `env-values.md` в формате `markdown`:

```
docker pull docker.idp.yc.ziiot.ru/infra/zifctl:2.16.0
docker run --rm  docker.idp.yc.ziiot.ru/infra/zifctl:2.16.0 get-wrapper > zifctl
./zifctl doc env-values --output-path . --format html
```

По указанному пути будет сгенерирован файл `env-values.html` в формате `html`.

8 Изменения Apache Keycloak, начиная с релиза Платформы 2.17.0

Обновление текущей версии 17, входящей в состав Платформы на версию 21.

Обновление не должно повлиять или исказить/повредить данные как самой Платформы, так и информацию о текущих пользователях.

Изменения:

1) `baseUri` больше не будет использоваться, поэтому был удалён из модели конфига `keycloak`.
2) `KEYCLOAK_URL` не используется и была удалена.
3) `Helm chart keycloak` в будущем будет удален так как будет использоваться `helm`-чарт от `Bitnami`.
4) Добавлен новый класс `version` для указания версии `keycloak`, которую необходимо развернуть.
5) Изменена логика установки значения `default_servicename` — для версии 17 имя сервиса было с добавлением `-http`.
6) Добавлен новый файл с переменными `zif-keycloak21.yaml.gotmpl` для развертывания нового чарта `keycloak`.
7) Внесены изменения в чарте `zif-infra-ingress` чтобы при изменении версии ссылка `/auth` ссылалась на верный `keycloak` сервис (`zif-keycloak-http` или `zif-keycloak21`).
8) Внесены изменения в главный `helmfile` чтобы при установке выбиралась версия, указанная в конфигурационном файле `env-values.yaml`.
9) В базовый образ добавлен `keycloak-config-cli-21.0.1` для работы с серверной версией 21 `keycloak`.
10) Образ `zif-keycloak21` содержит темы `zif`. Темы необходимо было переработать для работы с новой версией.
11) Образ `zif-keycloak21` подготовлен для работы в `ОКД` без использования `security context`.

Необходимые действия для обновления:

Внимание! Необходимо удалить `baseUri` из блока `keycloak` в конфигурационном файле `env-values.yaml`:

1) Откройте файл `env-values.yaml` на редактирование и удалите из блока `keycloak` строку:

```
baseUri: '[http://zif-keycloak-http.dev-guseynov/auth'](#описание-задачи)**
```

2) Укажите версию 21, если хотите обновить текущий `keycloak`:

```
version: 21
```

Если версия не указана, то по умолчанию будет взята версия 21.

Файл до обновления:

```
env-config.yaml
## Конфигурация встроенного keycloak
keycloak:
  installed: True
  baseUri: 'http://zif-keycloak-http.dev-guseynov/auth'
  baseExtUri: 'https://dev-guseynov.kube07.yc.ziiot.ru/auth'
```

Файл после обновления:

```
env-config.yaml
## Конфигурация встроенного keycloak
keycloak:
  installed: True
  baseExtUri: 'https://dev-guseynov.kube07.yc.ziiot.ru/auth'
  version: 21
```

9 Поддержка встроенной (native) OpenID Connect аутентификации для Apache NiFi

9.1 Два типа аутентификации для NiFI

Касательно Apache Nifi, поставляемом в составе Платформы, начиная с релиза 2.13.0 произошли следующие существенные изменения:

- Повышение поставляемой версии Apache Nifi до 1.19.1.
- Добавлена поддержка native-аутентификации пользователей через KeyCloak, с использованием протокола OpenId Connect.
- Добавлена возможность генерации TLS-сертификатов для узлов NiFi при развертывании, что необходимо для включения native-аутентификации.

В результате этих доработок, Платформы теперь поддерживает два варианта аутентификации и авторизации пользователей для NiFi:

1. Аутентификация с использованием OAuth Proxy (oauth-proxy), этот режим существовал и в предыдущих версиях.
   - Плюсы: проще в настройке: не требует сертификатов на узлах NiFi, не требует TLS на узлах NiFi, не требует создания учетных записей в NiFi.
   - Минусы: нет полноценной авторизации (разделения прав доступа) - любой пользователь, который сможет аутентифицироваться через OAuth Proxy в NiFi будет являться полным администратором и фактически на уровне NiFi нет разделения учетных записей пользователей (а сама возможность аутентификации на proxy ограничивается одной общей группой в KeyCLoak).
2. Аутентификация NiFi напрямую в OpenID Connect Provider (в KeyCloak), она же native-аутентификация
   - Плюсы: с точки зрения NiFi пользователи обладают отдельными полноценными учетными записями, которым можно предоставлять и ограничивать права средствами самого NiFi, все действия в NiFi реализуются от имени конкретной (а не общей) учетной записи.
   - Минусы: требуется включение TLS на узлах NiFi, что в свою очередь требует выдачи и обновления TLS-сертификатов узлов. Так же требуется создание и управление учетными записями в NiFi (фактически надо поддерживать два согласованных набора учетных записей для пользователей - в KeyCloak и отдельно в NiFi).

9.2 Генерация сертификатов для узлов NiFi

Для поддержки аутентификации пользователей средствами самого NiFi (а не внешними вроде oauth-proxy), узлы Apache NiFi должны быть сконфигурированы в т.н. защищенный кластер и на них должен быть включен TLS.

Для работы TLS необходимо обеспечить каждый узел кластера NiFi собственным уникальным закрытым ключом и TLS-сертификатом.

Для решения этой задачи в систему установки Платформы начиная с версии 2.13.0 добавлен инструментарий PKI, позволяющий генерировать сертификаты на этапе развертывания для узлов

некоторых сервисов инфраструктуры, включая NiFi. Инструментарий поддерживает два режима работы:

- Генерация сертификатов силами решения cert-manager, который должен быть предварительно установлен в кластере. В этом случае zifctl создает Namespace-level Issuer (если нужно) и создает объекты certificate с необходимыми параметрами. Это рекомендованный метод создания и управления сертификатами.
- Генерация сертификатов силами самого zifctl: в этом случае на этапе создания конфигурации создается RootCA-сертификат, уникальный для данной инсталляции, и на этапе развертывания zifctl выпускает необходимые TLS-сертификаты и самостоятельно загружает их в объекты secret кластера Kubernetes. Обновление сертификатов выполняется при очередных обновлениях платформы (zifctl sync).

9.3 Управление пользователями в NiFI

Если используется oauth-proxy аутентификация, то управление пользователями в NiFI невозможно: все пользователи с точки зрения Apache NiFi являются анонимными с полными административными правами (фактически в самом NiFi аутентификация и авторизация отключены). Администратор платформы может только ограничить список тех, кто может зайти в NiFi с помощью специальной роли в KeyCloak: apache-nifi-client:nifi-admin (client role).

Если используется native аутентификация, то пользователи будут представлены в NiFi полноценными индивидуальными учетными записями, которым можно назначить права и ограничения средствами самого NiFI. Но при это Apache NiFi не предоставляет штатных средств синхронизации этих учетных записей с провайдером аутентификации. Для логина пользователя с использованием KeyCloak необходимо:

- Создать учетную запись пользователя в KeyCloak (или синхронизировать ее из ActiveDirectory).
- Создать точно такую же учетную запись в NiFi (она должна совпадать с полем email УЗ в KeyCloak).
- Назначить права доступа учетной записи в NiFi (и/или включить ее в соответствующие группы в NiFi).
- При удалении учетной записи из KeyCloak, необходимо удалить и соответствующую учетную запись из NiFi.

После начального развертывания в NiFi будет создана только одна учетная запись (initial admin identity), соответствующая создаваемой zifctl учетной записи в KeyCloak nifi-admin.

9.4 Возможность развертывания стороннего приложения на кластеры kubernetes при помощи zifctl

9.4.1 Основные принципы развертывания приложения при помощи zifctl

- приложение инициализируется (создает конфигурационные файлы) используя образ zifctl в режиме установки приложения (--type app)
- приложение развертывается в отдельный неймспейс k8s, отличный от неймспейса платформы
- приложение "читает" конфигурацию платформы, из неймспейса платформы для получения необходимых учетных записей и информации об окружении на этапе инициализации;

- приложение при помощи образа `zifctl` производит установку (`sync`) конфигурации на кластер `k8s`.

9.4.2 Функционал установки приложения при помощи zifctl

- создание необходимых баз в `postgres`;
- создание клиентов, пользователей, ролей в `keycloak`;
- публикация необходимых сведений о платформе в неймспейсе приложения;
- настройка политики ABAC на платформе для приложения.

9.4.3 Этапы развертывания приложения инсталлятором

Установка стороннего приложения средствами `zifctl` во многом повторяет логику развертывания платформы (`init` → `sync`).

9.4.3.1 Подготовка

Этапы подготовки окружения к развертыванию приложения практически идентичны этапам, которые необходимо пройти в случае установки платформы:

- создание `AGE`-ключа шифрования;
- создание `namespace` на целевом кластере;
- создание сервисной учетной записи (`Service Account`) с правами на запись в namespace приложения и с правами на чтение объектов `configmaps`/`secrets` из `namespace` платформы.

9.4.3.2 Инициализация конфигурации (`init`)

Обязательным условием для инициализации конфигурации приложения является указание параметра `--type app`.

Пример команды:

```
# Определяем переменные
export ZIF_REGISTRY=docker-group.idp.yc.ziiot.ru
export ZIF_AGE_KEY=************
export ZIF_SERVER=https://51.250.9.223
export ZIF_TOKEN=************
export SA=sa-name-for-deployment
export ZIFCTL_REG_USERNAME=************@idp.zyfra.com
export ZIFCTL_REG_PASSWORD=************
# Помощь по параметрам команды init
zifctl init -help
zifctl init --env-path /path/to/env/config \
```

```
            --type app \
            --app-name sampleapp \
            --sa $SA \
            --infra-ns $INFRA_NS \
            --age-key $ZIF_AGE_KEY \
            --namespace $PROJECT_NAME \
            --hostname $KUBE-API \
            --registry-username $ZIFCTL_REG_USERNAME \
            --registry-password $ZIFCTL_REG_PASSWORD \
            --registry $ZIF_REGISTRY
            --token $ZIF_TOKEN
```

Основные параметры команды `init`, задающие значения в новой конфигурации:

- `--namespace` - задает `Namespace` куда будет устанавливаться приложение;
- `--type app` - режим установки приложения;
- `--app-name` - опциональный параметр, задает имя развертываемого приложения
- `--sa` - `Service Account` для запуска инфраструктурных контейнеров, которые требуют запуска от конкретного `UID` (т.е. повышенных привилегий). Актуально в первую очередь для `OpenShift/OKD` инсталляций. Если не задан, то все запускается от учетной записи по умолчанию
- `--infra-ns` - `Namespace` в котором установлена инфраструктура платформы (учетная запись, используемая в ключе "--sa" должна иметь `rolebindings` с правами на чтение конфигмэпов и секретов в неймсейсе, указанном в `--infra-ns`);
- `--hostname` - базовое доменное имя для доступа к экземпляру платформы (по этому имени будет доступен портал, а сервисы будут иметь пути вроде `https://<base-hostname>/<service-name>` (ВАЖНО: указывается не ссылка URL, а именно `hostname`);
- `--age-key` - (или переменная `ZIF_AGE_KEY`) опциональные и фактически задают значения для основных параметров в файлах `env-values.yaml` и `env-secrets.yaml`;
- `--registry` - имя репозитория откуда кластер `K8s/OpenShift` должен брать контейнеры Платформы и инфраструктуры (по умолчанию `docker.idp.yc.ziiot.ru`);
- `--registry-username` - имя пользователя для репозитория. Из него и `registry-password` формируется `pullSecret`, в большинстве случаев должно быть указано, если только это не полностью открытый внутренний репозиторий);
- `--registry-password` - пароль пользователя для репозитория `--token - kubeconfig` сервисной учетной записи кластера `k8s`.

При успешной инициации должно появиться следующее сообщение:

```
2024-10-12 11:32:10,242 DEBUG   cmd: Finish processing command: init, execution time: 0.6128787994384766 s
```

После инициализации в каталоге, указанном в ключе `--env-path` появятся следующие файлы:

```
├── deployment-id.yaml
├── env-secrets.yaml
├── env-values.yaml
├── trusted-root-ca
│   └── README.md
└── values
    └── global
        └── global-config.yaml.gotmpl
```

- `deployment-id.yaml` - `id` развертывания;
- `env-secrets.yaml` - секреты в зашифрованном виде;
- `env-values.yaml` - общая конфигурация развертываемого приложения;
- `trusted-root-ca` - директория с корневыми сертификатами, используемые сервисами приложения;
- `values` - директория с конфигурациями конкретного сервиса.

Файл "`global-config.yaml.gotmpl`" содержит конфигурацию, которая будет применена ко всему приложению. В следующих версиях `zifctl` данный файл не будет создаваться при инициации конфигурации.

После добавления файлов необходимых для дальнейшего развертывания, директория будет выглядеть следующим образом:

```
├── services
│   └── service-list.yaml
├── deployment-id.yaml
├── env-secrets.yaml
├── env-values.yaml
├── trusted-root-ca
│   └── README.md
└── values
    ├── global
    │   └── global-config.yaml.gotmpl
    └── app-folder
        ├── service1.yaml.gotmpl
```

```
        ├── service2.yaml.gotmpl
        └── serviceN.yaml.gotmpl
```

- `service-list.yaml` - `service-list.yaml` содержит список сервисов, являющихся частью приложения
- `env-values.yaml` - основной конфигурационный файл развертываемого приложения
- `*.gotmpl` файлы конфигурации под каждый конкретный сервис

В файле `env-values.yaml` необходимо указать имя приложения, которое планируется к развертыванию:

```
appName: sample-app
```

Примечание. Указанное имя приложения должно быть идентичным следующему значению, указанному в `service-list.yaml`:

```
modules:
  - name: sample-app
```

9.4.3.3 Модификация service-list.yaml

`service-list.yaml` - один из ключевых конфигурационных файлов приложения. Данный файл описывает сервисы, которые необходимо развернуть, настраивая, следующие параметры:

- имя сервиса;
- путь к образу;
- тэг;
- пробы;
- сетевой доступ;
- инициализации;
- политики доступа;
- лимиты/реквесты;
- и т.д.

Пример конфигурации `service-list.yaml`:

```
platform:
  version: 1.1.1
  configSchemaVersion: 100
```

```
modules:
  - name: sample-app
    chart: zif-sample-module
    description: Sample app to be deployed via zifctl
    importRoles: true
    services:
      - name: app-number-1
        enabled: true
        gitLabProjectID: 2190
        image: app-gatewayrouting
        imageNamespace: idp-pmc
        tag: 1.1.1
        auth: confidential
        ingressPath: ""
        type: netcore
        postgres: true
        abac:
          enabled: true
      - name: app-number-2
        enabled: true
        gitLabProjectID: 2191
        image: app-number-2
        imageNamespace: idp-pmc
        tag: 1.1.1
        auth: confidential
        ingress: true
        ingressPath: /
        type: netcore
        postgres: no
```

9.4.3.4 Модификация "*.gotmpl" файлов

`Helmfile` может наполняться значениями как с помощью `yaml` файлов, так и посредством `Go` шаблонов. Например, шаблон конфигурации сервиса `service1` будет "читаться" из файла `values/app-folder/service1.yaml.gotmpl`. В файле `values/app-`

`folder/defaults.yaml.gotmpl` задаются параметры, переменные, которые будут переданы всем сервисам приложения.

Пример параметров, которые возможно определить в `Go` шаблонах:

```
# Переменные окружения для сервиса
environmentVars:
  TZ: Europe/Moscow
  JAEGER_ENABLED: "false"
# Порт, который будет использоваться для маппинга сервиса
containerPorts:
  - name: http
    container: 7991
    service: 80
# Переопределение реквестов и лимитов
resources:
  requests:
    cpu: 100m
    memory: 128Mi
  limits:
    cpu: 500m
    memory: 512Mi
# Определение конфигурационных файлов для сервиса
extraConfigMaps:
  json-sample-config:
    sample.json: |-
      {
        "wsUrl": "wss://{{ .Values.externalBaseHostname }}/ws",
        "restUrl": https://{{ .Values.externalBaseHostname }}",
        "projectName": "SampleProject"
      }

  yaml-sample-config:
    sample.yaml: |-
      sampleconf:
        - enabled: true
```

```
        - type: yaml
```

9.4.3.5 Включение ABAC

Для включения\импорта `ABAC` политик приложения, данные политики необходимо определить в `Go` темплейтах, после чего `zifctl` монтирует политики и импортирует их в `zif-security`.

Примечание. За формирование и предоставление корректных шаблонов `ABAC` политик отвечает команда приложения.

Процедура:

1. Включить политики `ABAC` в `env-values.yaml` платформы:

```
## Конфигурация ABAC
abacAuthorization:
  enabled: True
  importPolicies: True
```

2. Аналогичным образом включить `ABAC` политики в Приложении:

```
## Конфигурация ABAC
abacAuthorization:
  enabled: True
  importPolicies: True
```

Так как политики `ABAC` "включены" для всех сервисов по умолчанию, то в случае отсутствия необходимости использования `ABAC`, отключение политики необходимо указать в `service-list.yaml` явным образом:

```
services:
     - name: app-number-1
       abac:
         enabled: false
```

3. Синхронизировать версию `keycloak` приложения и платформы

Версия `keycloak` в `env-values.yaml` платформы должна быть идентичной версии `keycloak` указанной в `env-values.yaml` приложения. В данный момент наличествует совместимость приложения с `keycloak` версий 17 и 21.

```
## Конфигурация KeyCloak
keycloak:
  installed: False
  baseExtUri: 'https://sample-platform-path.kube07.yc.ziiot.ru/auth'
  version: 21
```

4. Внести изменения в `Go` темплейты:

```
extraConfigMaps:
{{/* configMap for abac policies */}}
{{- if (index .Values.ZifService "abac") }}
  {{- if (.Values.ZifService.abac | get "enabled" true) }}

  {{ .Values.ZifService.name }}-abac-policy:
    {{ .Values.ZifService.name }}.json: |-
      {
          "id": "sample-policy",
          "description": "Политика sample",
          "actions": [
              {
                  "id": "update",
                  "name": "Обновить"
              }
          ],
          "policy": {
              "defaults": {
                  "advice": "Обратитесь к руководителю АСУТП"
              },
              "allowRules": [
                  {
                      "name": "Доступ на чтение и изменение",
                      "description": "Чтение и запись для роли: zif-datalink.folders-
editor",
                      "subj": {
```

```
                        "contains": [
                            {
                                "attribute": "Roles",
                                "comparison": "contains",
                                "value": "zif-datalink.folders-editor"
                            }
                        ]
                    },
                    "obj": {
                        "match": [
                            {

                                "attribute": "Id",
                                "comparison": "match",
                                "value": "*"
                            }
                        ]
                    },
                    "act": ["write", "update", "create", "read"]
                }
            ]
        }
    }
  {{ end }}
{{ end }}
```

5. Настроить `init job` модуля приложения в файле `module-config.yaml.gotmpl`.

```
initjobs:
  {{ .Release.Name }}-init:
    mountConfigMaps:
      {{ .Values.ZifService.name }}-abac-policy:
        configMap: {{ .Values.ZifService.name }}-abac-policy
        mountPath: /platform/init/module/data/abac/policies/{{ .Values.ZifService.name
}}/{{ .Values.ZifService.name }}-abac-policy.json
```

```
        subPath: {{ .Values.ZifService.name }}.json
    environmentVars:
      REST_ZIF_SECURITY_URL:  http://zif-security.sample-platform-ns
      LARGE_CLIENT_HEADER_BUFFERS: "4 32k"
```

9.4.3.6 Модификация env-values.yaml

Файл `env-values.yaml` содержит верхнеуровневые настройки приложения, которые редко изменяются. В `env-values.yaml` указываются имя приложения, неймспейс платформы, неймспейс приложения, `storage class`, и.т.д. Так же в этом конфигурационном файле можно настроить установку сторонних сервисов, таких как `redis`, `kafka`, `cassandra`, `rabbitmq`, etc.

Эталонный `env-values.yaml` приложения:

```
configType: app
namespace:  'app-sample-ns'
serviceAccount: 'app-sample-ns-sa'
tenantName: 'ziiot'
tenantId: 'ziiot'
externalBaseHostname: 'app-sample-ns.kube02.yc.ziiot.ru'
infraHA: False
appName: sample-app
sharedInfraNamespace: 'ziiot-sample-ns'
storage:
  storageClass: sample-sc-ssd
## Конфигурация Redis
redis:
  installed: True
  host: 'zif-redis-master.app-sample-ns'
  port: 6379
  storageSizeMB: 1024
```

Примечание. На текущий момент работоспособность приложения протестирована с платформой, развернутой в режиме монолитного инстанса. Значения полей `'tenantName'` и `'tenantId'` должны совпадать с соответствующими полями развернутой платформы. Значение полей `'tenantName'` и `'tenantId'` должно быть следующим:

```
tenantName: 'ziiot'
```

```
tenantId: 'ziiot'
```

Примечание. При формировании имен баз данных и учетных записей БД, будут учитываться имя приложения, неймспейс, имя сервиса, но результирующая строка будет ограничена 63-мя символами.

9.4.3.7 Применение (sync) конфигурации приложения

Команда для развертывания и обновления приложения одна и та же (`'zifctl sync'`). Практически все операции, выполняемые системой развертывания, идемпотентны и при повторном выполнении «установки», поверх уже выполненной ранее, конфигурация приложения не изменится. Команда установки ("синхронизации" конфигурации с конфигурацией в кластере). Параметры `--age-key`, `--server`, `--token` могут быть заданы через переменные окружения `ZIF_AGE_KEY`, `ZIF_SERVER` и `ZIF_TOKEN` соответственно. Подробнее о процедуре генерации пары ключей средствами `age-keygen` описано в документации `zifctl`.

Синхронизация стороннего приложение на целевой ландшафт:

```
zifctl sync --env-path /path/to/env/config \
                    --age-key $ZIF_AGE_KEY \
                    --server $ZIF_SERVER \
                    --token $ZIF_TOKEN
```

10 Профили postgres (postgres profiles) или настройки postgres для мультикластерных PG-инсталяций

Малые инсталяции платформы или инсталяции с небольшим количеством данных позволяют обойтись скейлингом Платформы в `kubernetes`.

В редких случаях требуется не только выносить `PG` на ВМ (для удовлетворения требований `ИБ`), но и раскидывать `БД` инсталяции по разным кластерам `PG` для разгрузки ВМ и удовлетворения нужного количествава "9-ок" по доступности.

Начиная с версии `zifctl 2.16.X` появилась возможность в конфигурации учитывать такое разделение. Как его настроить описано ниже:

Внутренняя структура работы с `PG` у Платформы:

К PG ходят 2 сущности платформы:

- `init job`, которые получают `user/pass` как `admin user`, `dbname` генерируют на лету на основе `tenantID` и соответствующего сервиса, `host/port/connectdb` из переменных `ENV (POSTGRES_HOST / POSTGRES_PORT / POSTGRES_CONNECT_DB)`.
- сервисы Платформы, который получают `user/pass/dbname/host/port` из переменных окружения.

Начиная с версии `zifctl 2.16.X` можно полноценно переместить сервис на другие `postgres.host / postgres.port`, а `init job` заставить создать БД/поправить права на неё и т.д. на нужном вам кластере с помощью `postgres profiles` и параметра `connectDB`.

Внимание! Для сервиса настройка `connectDB` не нужна. Эта настройка нужно только для `init job` для выполнения задач по созданию БД сервиса, пользователя и создания/исправления прав на БД. Т.е. чтобы выполнить запрос `CREATE DATABASE` и подобные `init job` должна куда-то подключится (не указывая БД при запуске командочке в `psql` вы на самом деле подключаетесь к `БД postgres`).

10.1 Переключение сервиса Keycloak на другой PG

В рамках `Keycloak` теперь возможно полноценно с текущей версии начинается работать на двухкластерной инсталяции Платформы, когда всё кроме `keycloak` работало на `default.cluster`, а сам `keycloak` на `new.cluster`:

- создайте новый профиль `PG` и переопределить в нём `host/port/connectDB` сущности (одну из них как минимум):

```
$ grep -A 11 -F 'postgres:' ./env-config/env-values.yaml
postgres:
  installed: True
  host: 'default.cluster'
  port: 5432
  connectDB: postgres
  storageSizeMB: 5120
```

```
  profiles:
    fakeprofilename:
      host: 'new.cluster'
      port: 5432
      connectDB: postgres
```

- переключить сервис `keycloak` на нужный профиль в `env-values.yaml`:

```
$ grep -A 4 -F 'keycloak:' ./env-config/env-values.yaml
keycloak:
  installed: True
  postgresProfile: fakeprofilename
  baseUri: 'http://zif-keycloak-http.dev-dyukov/auth'
  baseExtUri: 'https://dev-dyukov.kube07.yc.ziiot.ru/auth'
```

10.2 Переключение сервиса X (не keycloak) на другой PG

В рамках `Keycloak` теперь возможно полноценно с текущей версии начинается работать на двухкластерной инсталяции Платформы, когда всё кроме `keycloak` работало на `default.cluster`, а сам `keycloak` на `new.cluster`:

В текущей реализации Платформы сервис `X` можно полноценно подключить к работе на двухкластерной инсталяции Платформы, когда всё кроме `X` работало на `default.cluster`, а сам `X` на `new.cluster`:

- создать новый профиль PG и переопределить в нём `host/port/connectDB` сущности (одну из них как минимум):

```
$ grep -B 9 -A 1 myfakeprofilename  ./env-config/env-values.yaml
postgres:
  installed: True
  host: 'default.cluster'
  port: 5432
  connectDB: postgres
  storageSizeMB: 5120
  profiles:
    fakeprofilename:
      host: 'new.cluster'
```

```
      port: 5432
      connectDB: postgres
```

- переключить сервис X на нужный профиль через `service-list-patch.yaml`:

```
$ cat ./services/service-list-patch.yaml
modules:
  - name: rtdb
    services:
      - name: zif-rtdb-metadata
        postgresProfile: fakeprofilename
```

- **Внимание**! Если Вы забыли (или намеренно) не указали параметры в профиле они будут взяты из `postgres.host / postgres.port / postgres.connectDB`, т.е.:

```
postgres:
  host: 'default.cluster'
  port: 5555
  connectDB: postgres1
  profiles:
    fakeprofilename:
      connectDB: postgres
```

Обращение выше можно заменить аналогичным более подробным:

```
postgres:
  host: 'default.cluster'
  port: 5555
  connectDB: postgres1
  profiles:
    fakeprofilename:
      host: 'default.cluster'
      port: 5555
      connectDB: postgres
```

- пустой профиль является некорректной конфигурацией, потому как нет смысла создавать пустой профиль в виду пункта выше:

```
# некорректная конфигурация
postgres:
  host: 'default.cluster'
  port: 5555
  connectDB: postgres1
  profiles:
    myprofile1:
    myprofile2:
```

- после создания двух дополнительных кластеров PG для двух баз и не используете никакие балансировщики, то типичная конфигурация будет выглядеть так:

```
postgres:
  host: 'default.cluster'
  port: 5432
  connectDB: postgres1
  profiles:
    cluster1:
      host: 'custom1.cluster'
    cluster2:
      host: 'custom2.cluster'
# port не нужно указывать, если он совпадает с postgres.port
# В данном случае у меня 3 кластера на разных машинах, мастера которых доступны по hostname:
# 1) custom1.cluster
# 2) custom2.cluster
# 3) default.cluster
```

- если установлен балансировщик и несколько кластеров PG, то порядок настройки:
  1) На каждом из кластеров создать маркировочные БД "пустышки" для создания маршрута через балансировщик для `init job`, например `cluster1db` , `cluster2db`, `cluster3db`.
  2) Настройте профили:

```
postgres:
  host: 'pgbouncer.manualinfra'
  port: 5432
  connectDB: cluster1db
  profiles:
    secondcluster:
      connectDB: cluster2db
    thirdcluster:
      connectDB: cluster3db
# host и port в каждом из профилей не требуется дублировать, т.к. он совпадает с postgres.host и postgres.port
# В данном случае у меня 3 кластера на разных машинах, ко всем мы ходим через балансировщик на pgbouncer.manualinfra:5432
```

3) Переключите сервисы на нужный профиль:

```
$ cat ./services/service-list-patch.yaml
modules:
  - name: rtdb
    services:
      - name: zif-rtdb-metadata
        postgresProfile: thirdcluster
  - name: om
    services:
      - name: zif-om-object
        postgresProfile: secondcluster
```

4) Произвести команду `zifctl sync`.

11 Автоматическое создание и конфигурация топиков Kafka в процессе развертывания

Система развертывания должна автоматически создавать топики в `Kafka` с указанными параметрами (`replicas`, `partitions`) и обновлять параметры топиков, если они изменились.

В рамках данной задачи решается вопрос только управления "статическими" топиками, состав и названия которых известны заранее на этапе конфигурации системы. Управление "динамическими" топиками, которые создаются сервисами в процессе работы "по запросу" вне этой задачи.

В рамки этой задачи так же входит управление топиками `Debezium` (`zifctl` должен взять на себя создание и изменение параметров топиков). Конфигурация `Debezium` не меняется относительно существующей, просто добавляется их предварительное создание вместе со всеми остальными.

Изначально заложенные платформой параметры топиков определены в файле `/services/kafka-topics-list.yaml`, по формату структуре аналогичен `/services/service-list-patch.yaml`. Так же файл является статичным в образе `zifctl` и изменению не подлежит.

Начиная с 2.16.0 появилась возможность управления `kafka` топиками.

Для того, чтобы внести изменения в `kafka` топики для конкретного экземпляра Платформы, вам необходимо создать патч-файл `/services/service-list-patch.yaml` в каталоге конфигурации окружения, в котором будут перечислены те модули и сервисы, в параметры которых вы хотите внести корректировки:

- для включения роли по управлению топиков необходимо убедиться что в `env.values.yaml` включена `tasks -kafka` или `-all`. Пример файла `service-list-patch.yaml``:

```
init:
  tasks:
    - all
```

- для управления топика необходимо в `/services/service-list-patch.yaml` на уровне сервиса добавить блок `kafka`: вложенным словарем топиков и их параметрами. Все указанные параметры указаны в примере ниже:

```
modules:
  - name: <<module>>
    services:
      - name: <<service>>
        enabled: true
        kafka:                                          #required.
           enabled: true                                #required.
           topics:                                      #required.
              topic-1:                                        #required. Должно
совпадать с названием топика без префикса "tenantid__"
```

```
                name: "topic-1"                                #required. Наименование топика без префикса "tenantid__". Пр. <<tenantid>>__zif-om-object__dbz.public.properties_s, должен быть указан как zif-om-object__dbz.public.properties_s
                partitions: "1"                                #required. default = "1". ВАЖНО!!! Данный параметр не может быть ниже чем текущее количество партиций в указанном топике
                cleanup_policy: "delete,compact"               #optional. default = "delete" ["compact","delete","delete,compact"]
                segment_ms: "604800000"                        #optional. default = "604800000" (7 days) [1,...]
                flush_messages: "9223372036854775807"          #optional. default = "9223372036854775807" [1,...]
                max_message_bytes: "1048588"                   #optional. default = "1048588" [0,...]
                min_insync_replicas: "1"                       #optional. default = count pods cp-kafka
                retention_bytes: "-1"                          #optional. default = "-1" [-1, 0,...]
                retention_ms: "86400000"                       #optional. default = "86400000" (1 day) [-1, 0,...]
                flush_ms: "9223372036854775807"                #optional. default = "9223372036854775807" [0,...]
                segment_bytes: "1073741824"                    #optional. default = "1073741824" (1 gibibyte) [14,...]
                segment_jitter_ms: "0"                         #optional. default = "0" [0,...]
                unclean_leader_election_enable : "false"       #optional. default = "false" ["false","true"]
              topic-2:
                name: "topic-2"
                partitions: "2"
                retention_ms: "100000"
                flush_ms: "200000"
                cleanup_policy: "compact"
                segment_ms: "2000000000"
              topic-N:
                name: "topic-N"
                partitions: "2"
```

```
  - name: <<module>>
    services:
      - name: <<service>>
        enabled: true
        kafka:
          enabled: true
          topics:
            topic-N42:
              name: "topic-N42"
              partitions: "16"
              retention_ms: "14400200"
```

- для 2.16.0 `/services/kafka-topics-list.yaml` имеет следующий вид:

```
modules:
  - name: om
    services:
      - name: zif-om-object
        kafka:
          enabled: true
          topics:
            zif-om-object__dbz.public.propertyconfigurations_s:
              name: "zif-om-object__dbz.public.propertyconfigurations_s"
              partitions: "1"
            zif-om-object__dbz.public.properties_s:
              name: "zif-om-object__dbz.public.properties_s"
              partitions: "1"
            zif-om-object__dbz.public.objects_s:
              name: "zif-om-object__dbz.public.objects_s"
              partitions: "1"
```

Технические детали решения:

1) В репозитории системы развертывания состав релиза (список топиков и их параметры) хранится в файле `/services/kafka-topics-list.yaml`.

2) При выполнении установки утилита `zifctl` с помощью вспомогательного скрипта `/scripts/hooks/prepare/15-merge-services-list.py` выполняет полный merge `/services/kafka-topics-list.yaml` с `/services/service-list-patch.yaml` (если этот файл существует).

При конфликте значение берется из `/services/service-list-patch.yaml`. Далее полученный файл объединяется с `/services/*service-list.yaml` согласно Правилам объединения базового и патч-файлов.

3) Добавлены кастомные роли (`roles`) (`initrolessrv_kafkatasksmain.yaml`), `module_utils` (`scriptslibansiblemodule_utils`), `modules` (`scriptslibansiblemoduleskafka_topic.py`) для управления `kafka topics`.
4) Вызов роли происходит с помощью существующих `Job` формирующих `initmodule00-init.yaml`.
5) Добавлена новая `init tasks ("kafka")` отвечающая за вызов роли.
6) Вынесена конфигурирование топиков по умолчанию в отдельный файл `services/kafka-topics-list.yaml`. Все кастомные изменения вносятся в `services/service-list-patch.yaml`.
7) В базовый образ `zifctl-base` добавлены компоненты `images/zifctl-base/requirements.txt`:

```
kafka-python>=2.0.0,<2.1.0
kazoo==2.6.1
pure-sasl==0.5.1
jsonmerge==1.9.2
```

8) Управление топиков основано на https://github.com/StephenSorriaux/ansible-kafka-admin.

12 Включение TLS, аутентификации и авторизации в Redis

Для обеспечения безопасности данных и шифрование соединения необходимо реализовать защищенное `TLS` соединение с `Redis` а так же добавить возможность включение аутентификации и авторизации.

В данной документации будет рассмотрено, как включить различные режимы безопасности и аутентификации для `Redis`, а также какие параметры буду добавлены в деплойменты в каждом режиме.

12.1 Включение режима TLS (Transport Layer Security)

`TLS` обеспечивает шифрование данных между клиентами и сервером `Redis` для обеспечения безопасности передачи данных.

Для включения режима `TLS`, установите следующий параметр в файле конфигурации `env-values.yaml`:

```
pki:
  enabled: true
redis:
  tls:
    enabled: true
```

Создаваемые параметры:

При включении режима `TLS` будут сгенерированы следующие переменные окружения:

- `REDIS_HOST=\<адрес подключения к redis>`;
- `REDIS_PORT=\<порт подключения, по умолчанию 6380>`;
- `REDIS_SSL=true`: Устаревшая переменная для указания поддержки `TLS`. Оставлена для совместимости;
- `REDIS_TLS_ENABLED=true`: Эта переменная указывает на поддержку `TLS`;
- `REDIS_TLS_CA_CRT_PATH=\<путь к корневому сертификату>`: Путь к корневому сертификату для `TLS`.

Подключены и смонтированы следующие `volumes`:

- `zif-generic-tls`;
- `zif-redis-tls`.

12.2 Включение режима AUTH (аутентификация)

Режим `AUTH` предоставляет аутентификацию клиентов при подключении к серверу `Redis`.

Для включения режима `AUTH`, установите следующий параметр в файле конфигурации `env-values.yaml`:

```
redis:
  auth: true
```

Создаваемые переменные окружения:

При включении режима `AUTH` будет сгенерирована следующая переменная окружения:

- `REDIS_HOST=\<адрес подключения к redis>;`
- `REDIS_PORT=\<порт подключения, по умолчанию 6379>;`
- `REDIS_PASSWORD=\<ваш пароль>`: Эта переменная указывает на установленный пароль для аутентификации.

12.3 Включение режима AUTH и ACL

Режим `AUTH` и `ACL` объединяет в себе аутентификацию клиентов и управление списками контроля доступа для максимальной безопасности и управления доступом.

Для включения режима `AUTH` и `ACL`, установите следующие параметры в файле конфигурации `env-values.yaml`:

```
redis:
  auth: true
  acl: true
```

Создаваемые переменные окружения при включении режима `AUTH` и `ACL` будут сгенерированы следующие переменные окружения:

- `REDIS_HOST=\<адрес подключения к redis>;`
- `REDIS_PORT=\<порт подключения, по умолчанию 6379>;`
- `REDIS_PASSWORD=\<ваш пароль>`: Эта переменная указывает на установленный пароль для аутентификации;
- `REDIS_USER=\<имя пользователя, по умолчанию tenant_modulename>`.

12.4 Включение режима TLS, AUTH и ACL

Режим `TLS` и `ACL` объединяет в себе шифрование данных, обязательной аутентификации и управление списками контроля доступа для максимальной безопасности.

Для включения режима `TLS` и `ACL`, установите следующие параметры в файле конфигурации `env-values.yaml`:

```
pki:
  enabled: true
redis:
```

```
  tls:
    enabled: true
  auth: true
  acl: true
```

При включении режима `TLS` и `ACL` должны создаваться все переменные окружения и монтироваться `volumes` как и в режимах `TLS` и `ACL`.

12.5 Режим публикации внешнего Ingress/Route соединения для сервера Redis

Этот режим позволяет обеспечить внешний доступ к серверу `Redis` через `Ingress` или `Route` с использованием `SSL passthrough`.

Параметры, необходимые для включения:

```
pki:
  enabled: true
redis:
  tls:
    enabled: true
  externalAccess: True
```

Для корректного создания внешнего доступа к `redis` в кластере с `ingress nginx` должен быть включен параметр `--enable-ssl-passthrough` при запуске `ingress controller`.

Создаваемые сущности:

При включении внешнего доступа будет создан `ingress/route` с адресом формата `\<namespace>-redis.\<domain.example>`.

В ingress присутствует аннотация `[nginx.ingress.kubernetes.io/ssl-passthrough](http://nginx.ingress.kubernetes.io/ssl-passthrough): "true"\`, указывающая на использовании `nginx` режима `ssl-passthrough` для этого адреса.

Подключение:

При подключении через любую утилиту или программу нужно указать:

- адрес указанный в `ingress/route`;
- порт подключения 443;
- указать путь до `CA` сертификата или сертификат в текстовом виде (зависит от способа);
- указать `SNI` который равен адресу;
- при использовании авторизации указать логин и пароль.

13 Включение отдельного Redis для инфраструктуры и настройка Apache Nifi

В ходе тестирования работы сервисов `UDL` обнаружилась избыточная сетевая нагрузка на сервис `zif-redis-master`, вызванная логикой работы платформенного процессора `Apache Nifi\_ProcessTSDSData`.

Для решения этой проблемы в авто-развертывание добавлена возможность установки отдельного экземпляра `Redis`, который будет использоваться только инфраструктурными компонентами платформы.

В документе описаны действия, необходимые для установки инфраструктурного `Redis` и конфигурации процессоров `Nifi`, использующих `Redis`.

13.1 Установка Redis для инфраструктуры

При переустановке на уже развернутую Платформу:

1) Добавьте в файл конфигурации `*\<путь до папки конфигурации zifctl>/env-config/env-values.yaml*` следующие параметры (включение `Redis` для инфраструктурных сервисов):

```
env-values.yaml | Включение Redis для инфраструктурных сервисов
## Конфигурация Redis используемого только инфраструктурными сервисами
redisInfra:
  installed: True
  host: 'zif-redis-infra-master.<namespace>'
  auth: True
  port: 6379 # по умолчанию, можно не указывать
  storageSizeMB: 1024 # по умолчанию, можно не указывать
```

2) Расшифровуйте файл в секретами `*\<путь до папки конфигурации zifctl>/env-config/env-secrets.yaml*`, используя команду `zifctl secrets dec` бинарного установщика `zifctl` (Дешифрование):

```
env-secrets.yaml | Дешифрование
./zifctl secrets dec --env-path=$ENV_PATH --age-key=$AGE_KEY
 2023-09-05 16:15:25,083 INFO    secrets: Write decrypted file: /var/deploy/env/env-secrets.dec.yaml, don't forget to delete it!
```

В результате будет создан файл `*\<путь до папки конфигурации zifctl>/env-config/env-secrets.dec.yaml*`, в который необходимо добавить параметры для `redisInfra` (Добавление параметров `Redis` для инфраструктуры):

```
env-secrets.yaml | Добавление параметров Redis для инфраструктуры
...
redisInfra:
    adminPassword: <password>
    adminUser: admin
...
```

3) Зашифруйте файл при помощи команды `zifctl secrets enc`:

```
env-secrets.yaml | Шифрование
./zifctl secrets enc --env-path=$ENV_PATH --age-key=$AGE_KEY
 2023-09-05 16:24:55,527 INFO   secrets: Encrypt secrets file: /var/deploy/env/env-secrets.dec.yaml => /var/deploy/env/env-secrets.yaml
 2023-09-05 16:24:55,539 INFO   secret_store_base: Save secrets to file: /var/deploy/env/env-secrets.yaml
```

В результате в `\<путь до папки конфигурации zifctl>/env-config/env-secrets.yaml`` будут добавлены зашифрованные параметры redisInfra`.

После удаления файла `*\<путь до папки конфигурации zifctl>/env-config/env-secrets.dec.yaml*` можно начать установку платформы командой `zifctl sync`.

При установке с нуля:

В `env-values.yaml` и `env-secrets.yaml` уже будут записаны параметры для `redisInfra`, перед выполнением команды `zifctl sync` необходимо только включить установку `redisInfra` (с аутентификацией):

```
env-values.yaml | Включение Redis для инфраструктурных сервисов
...
## Конфигурация Redis используемого только инфраструктурными сервисами
redisInfra:
  installed: True
  auth: True
...
```

13.2 Конфигурация Apache Nifi

Для тестирования корректной работы `Nifi` с `Redis` необходимо воспользоваться руководством из данного раздела.

Добавление сервисов контроллера `RedisConnectionPoolService` и `RedisDistributedMapCacheClientService`:

1) Войти в UI `Nifi` под административной УЗ.
2) Нажмите ПКМ и выберите `Configure`:

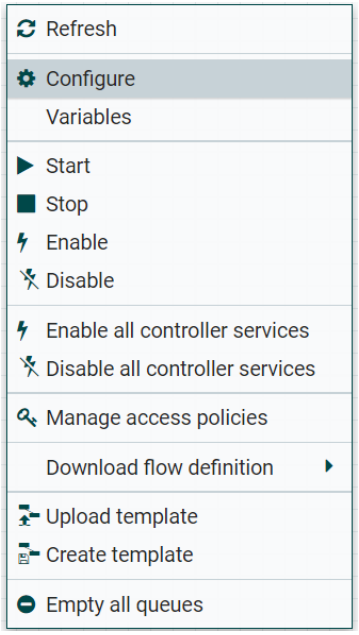

Рисунок 13.1 Configure

3) Перейдите на вкладку `Controller Services` → кнопка `"+"` в правом верхнем углу.
4) В открывшемся диалоговом окне `Add Controller Service` в текстовом поле `Filter` введите `Redis`:

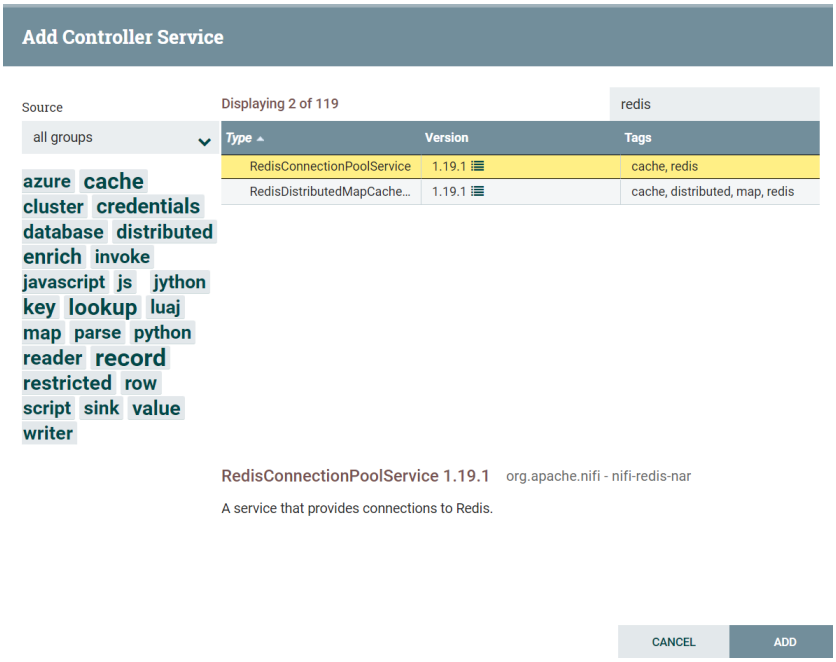

Рисунок 13.2 Redis

Последовательно добавляем сервисы контроллера `RedisConnectionPoolService`и `RedisDistributedMapCacheClientService`.

13.3 Конфигурация сервисов контроллера RedisConnectionPoolService и RedisDistributedMapCacheClientService

Конфигурация сервиса контроллера `RedisConnectionPoolService`:

1) Выберите кнопку конфигурации `RedisConnectionPoolService`:

Рисунок 13.3 RedisConnectionPoolService

2) В открывшемся диалоговом окне `Configure Controller \| ServiceRedisConnectionPoolService 1.19.1`, вкладка `Properties`, введите значения полей:

   - `Connection String`: `zif-redis-infra-headless.<namespace>:6379`;
   - `Password`: пароль администратора `redisInfra`; (`redisInfra.adminPassword` из `env-secrets.yaml` либо значение переменной `REDIS_INFRA_ADMIN_PASSWORD` из секрета `zif-global-secrets` в кластере);
   - `Apply`.

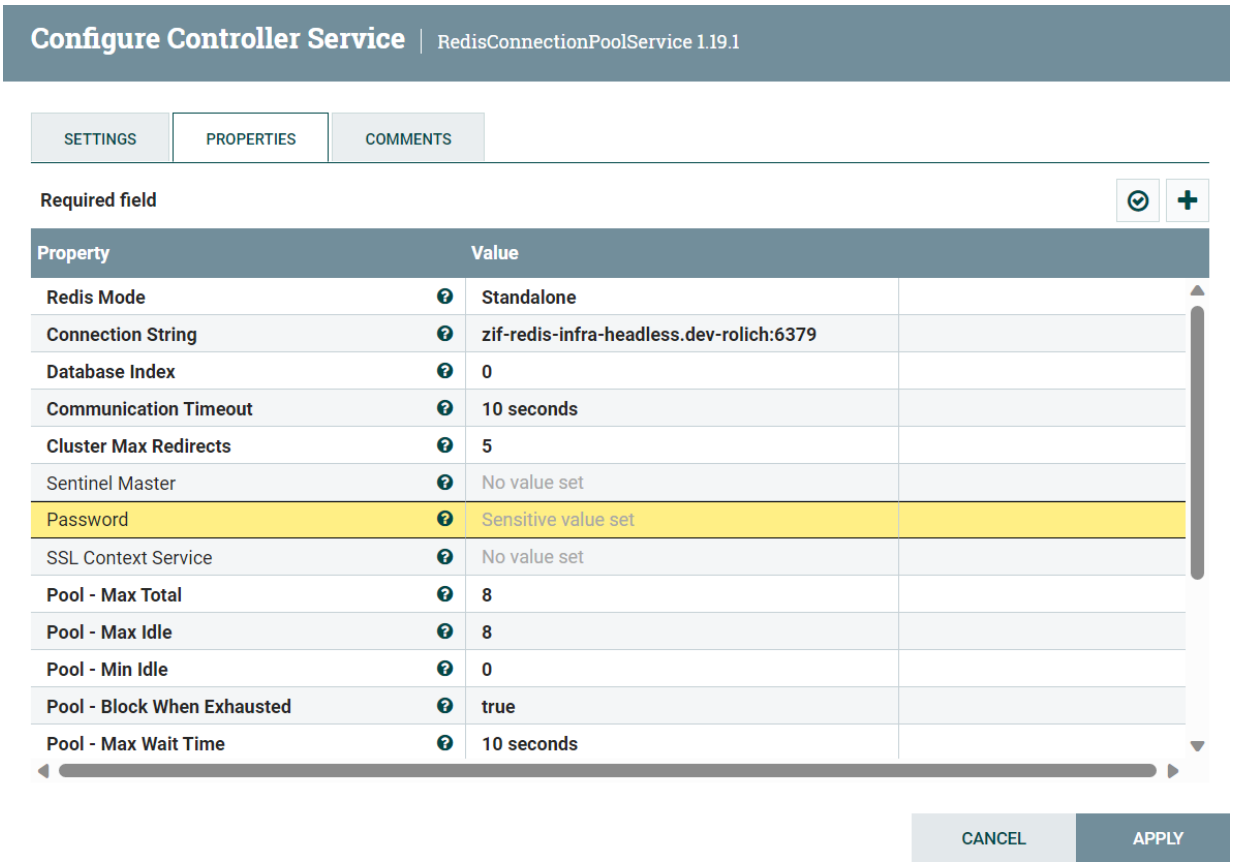

Рисунок 13.4 Вкладка Properties

Конфигурация сервиса контроллера `RedisDistributedMapCacheClientService`:

1) Нажмите кнопку конфигурации сервиса `RedisDistributedMapCacheClientService`.
2) В открывшемся диалоговом окне `Configure Controller \| ServiceRedisDistributedMapCacheClientService 1.19.1` выберите вкладку `Properties`.
3) Поле `Redis Connection Pool``, нажмите` Value, `в выпадающем списке выберите` RedisConnectionPoolService → OK → Apply`.

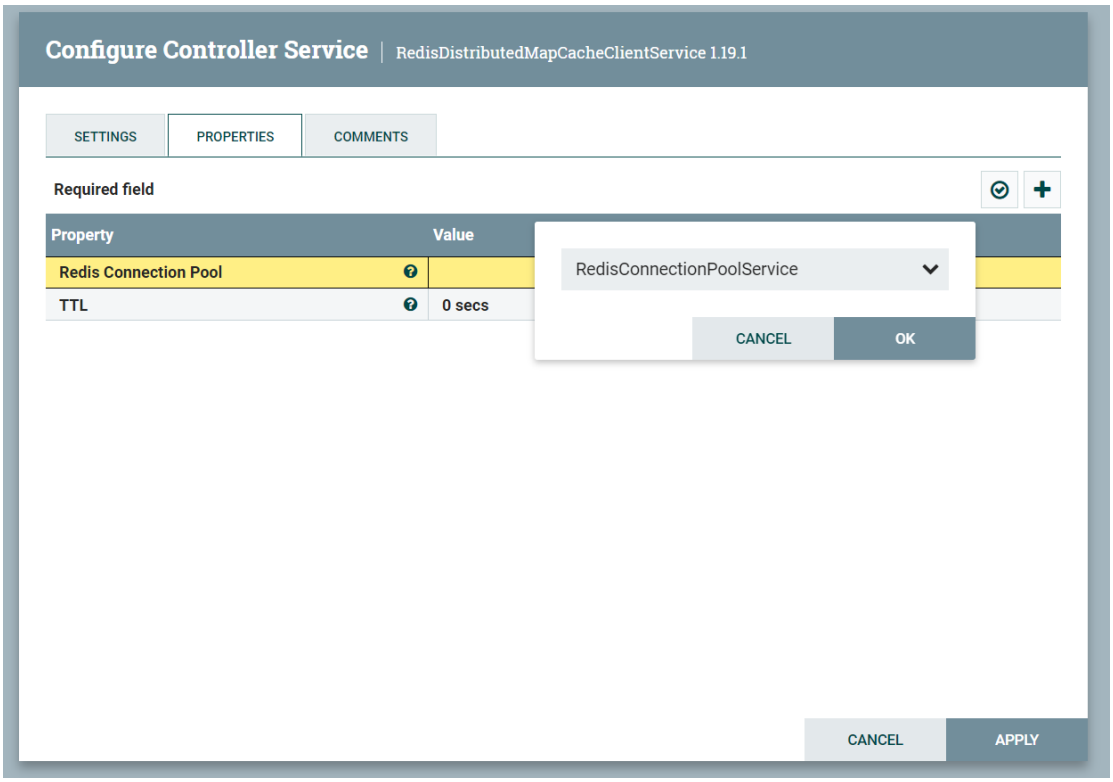

Рисунок 13.5 Configure Controller

13.4 Включение сервисов контроллера RedisConnectionPoolService и RedisDistributedMapCacheClientService

1) Включите сервис `RedisConnectionPoolService`:

Рисунок 13.6 RedisConnectionPoolService

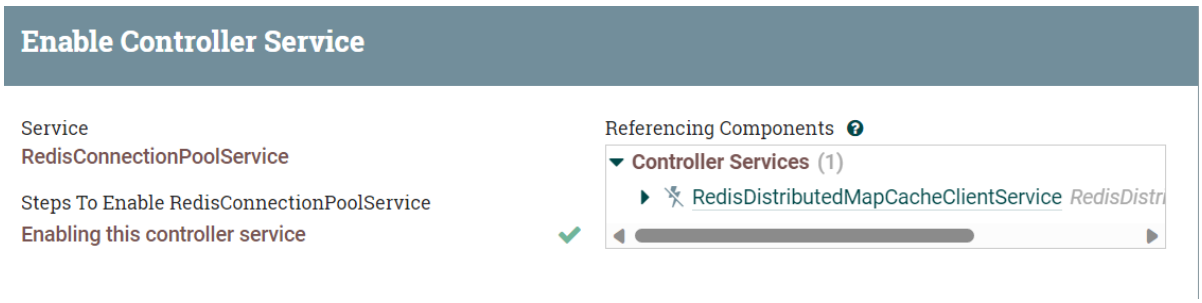

Рисунок 13.7 RedisConnectionPoolService

2) Включите сервис `RedisDistributedMapCacheClientService`:

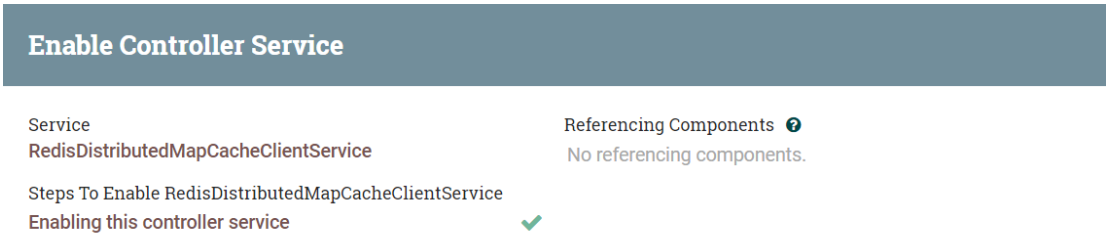

Рисунок 13.8 RedisDistributedMapCacheClientService

Добавление процессоров `GenerateFlowFile` и `PutDistributedMapCache`:

1) Для добавления процессоров наведите курсор на `Processors` и перетащите на поле:

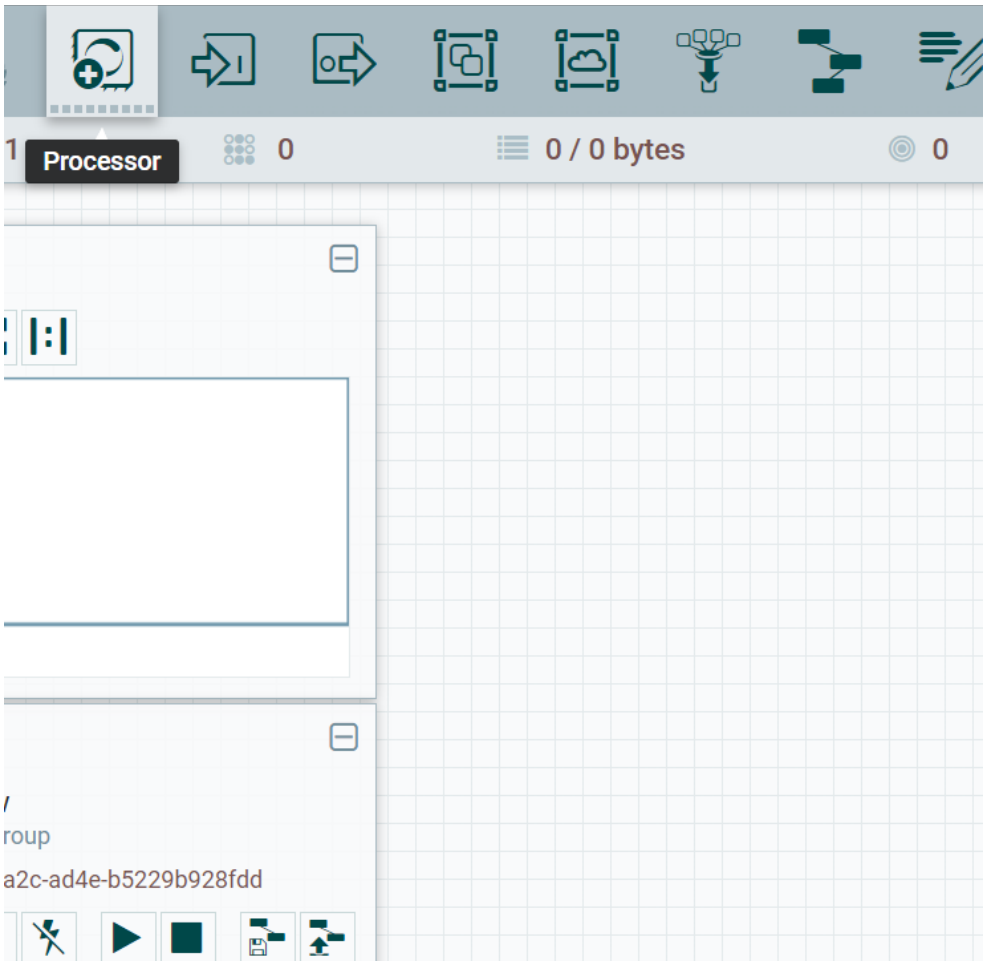

Рисунок 13.9 Processors

2) В открывшемся диалоговом окне `Add Processor` в поле `Filter` введите `GenerateFlowFile` → `Add`:

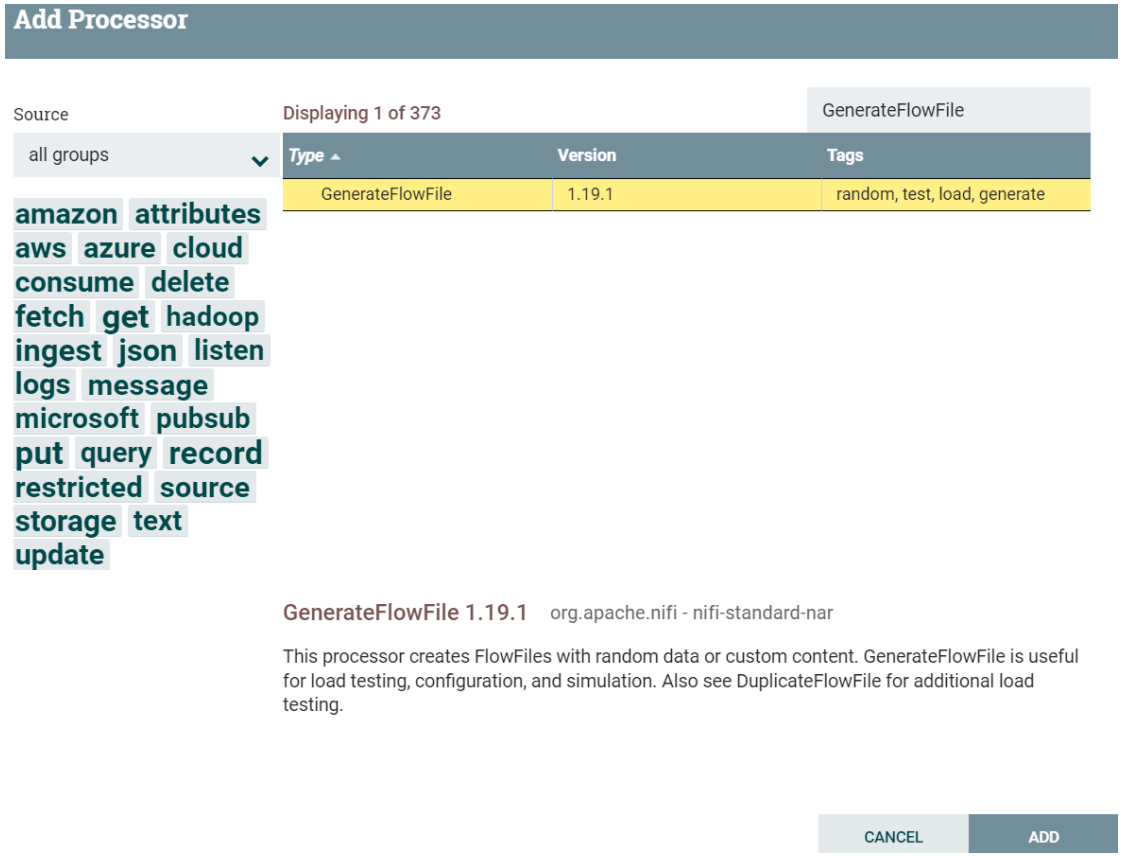

Рисунок 13.10 GenerateFlowFile → Add

3) Аналогичным образом добавьте процессор `PutDistributedMapCache`:

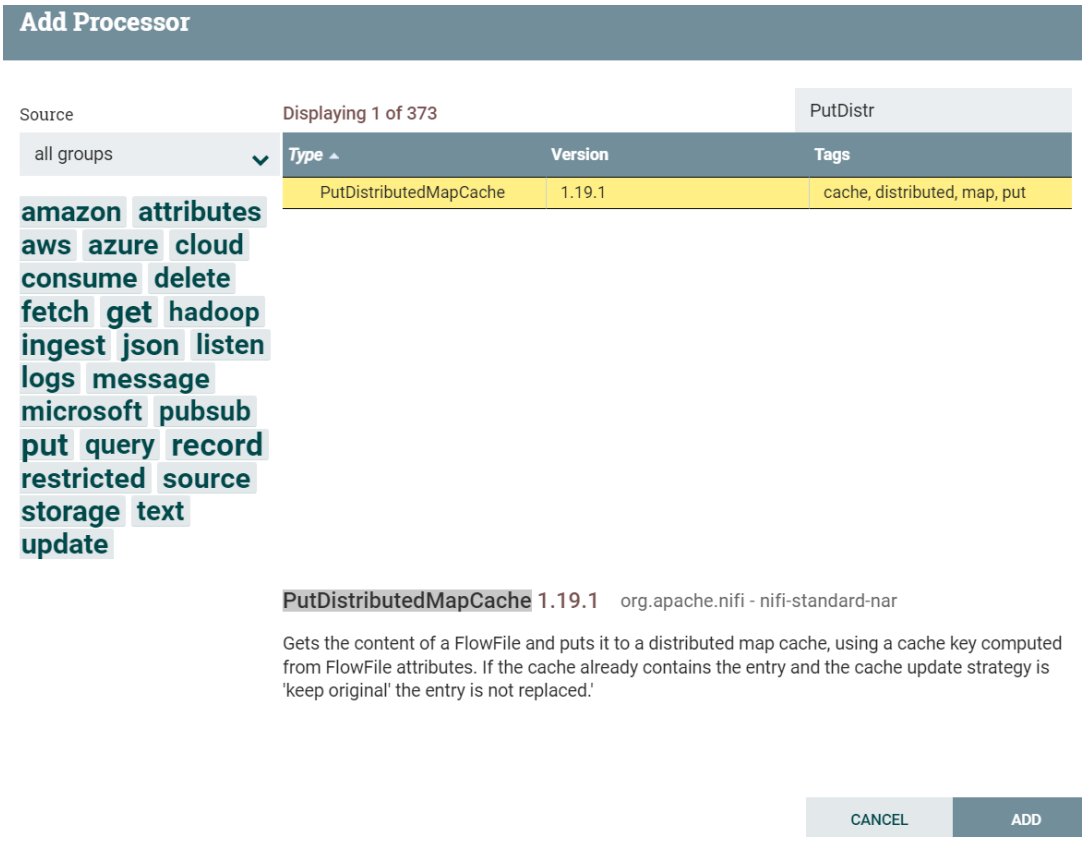

Рисунок 13.11 PutDistributedMapCache

4) Нажмите на стрелочку процессора `GenerateFlowFile` и тянем до `PutDistributedMapCache`:

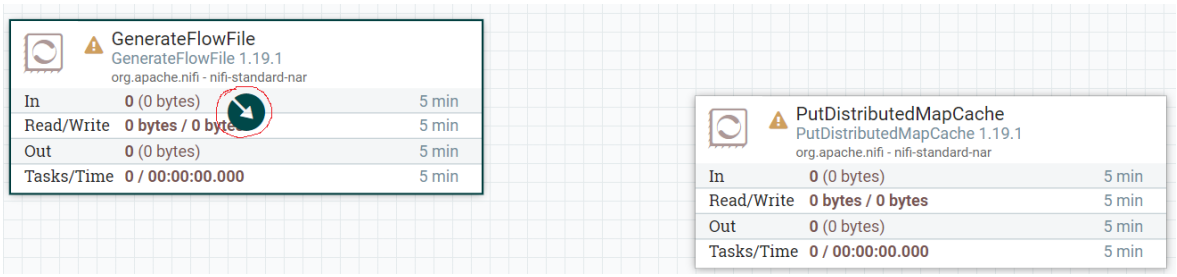

Рисунок 13.12 PutDistributedMapCache

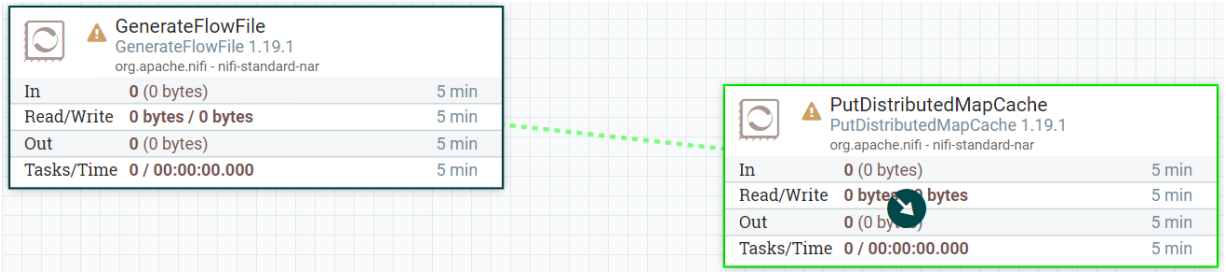

Рисунок 13.13 PutDistributedMapCache

5) В открывшемся диалоговом окне выберите `Create Connection → Add`.

13.5 Конфигурация процессоров GenerateFlowFile и PutDistributedMapCache

Конфигурация процессора `GenerateFlowFile`:

1) Двойной клик (либо ПКМ), затем `Configure` на процессоре `GenerateFlowFile`.
2) Вкладка `Properties`, в поле `Custom Text` введите `hello from nifi`, нажмите `Apply`:

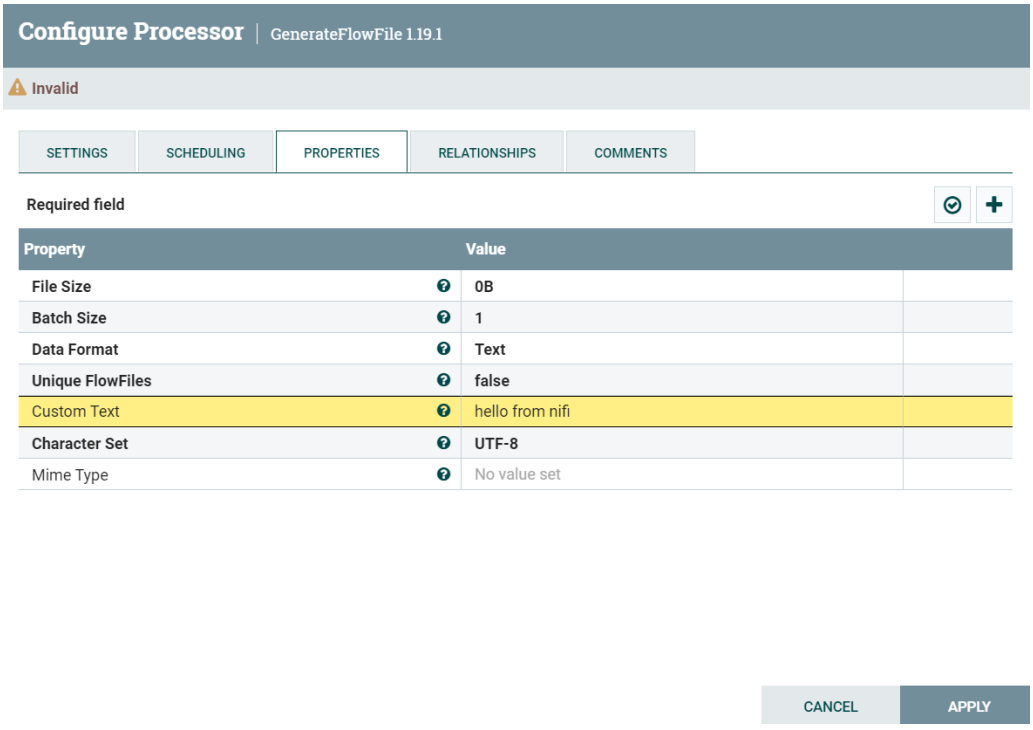

Рисунок 13.14 Properties

Конфигурация процессора `PutDistributedMapCache`:

1) Двойной клик (либо правый клик → `Configure`) на процессоре `PutDistributedMapCache`.
2) Вкладка `Properties`, в значения в следующие поля:
   - **Cache Entry Identifier**: nifi-key;
   - **Distributed Cache Service**: из выпадающего списка выбираем **RedisDistributedMapCacheClientService.

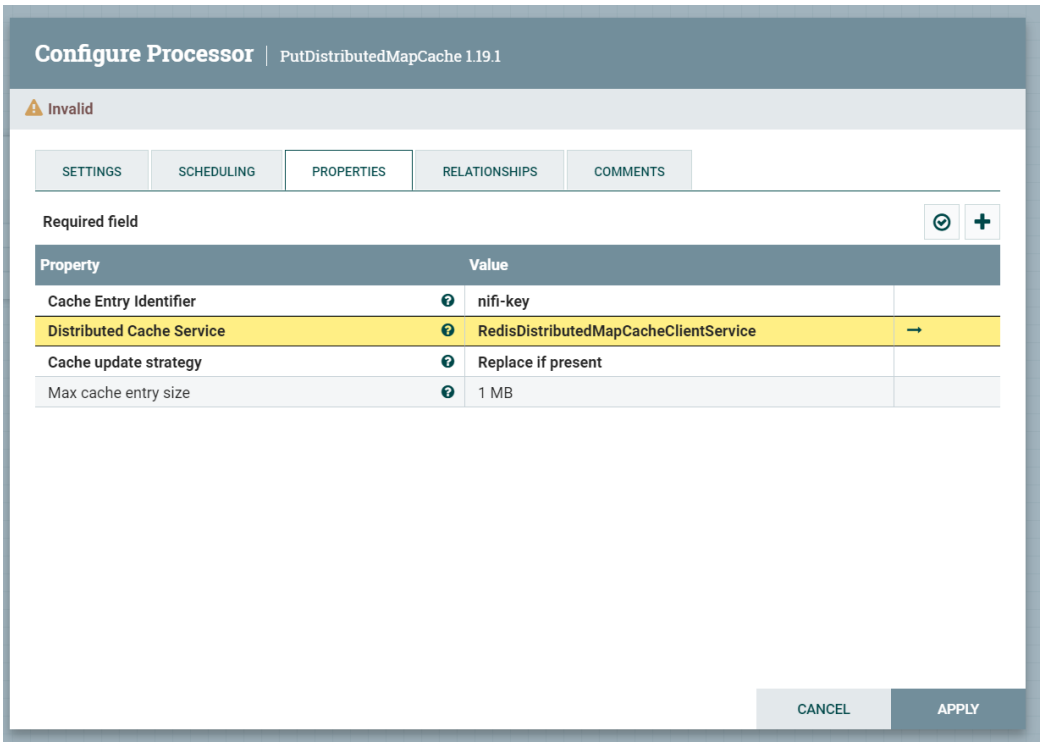

Рисунок 13.15 Properties

3) Вкладка `Relationships`, выберите галочки `terminate` в секциях `failure` и `success`:

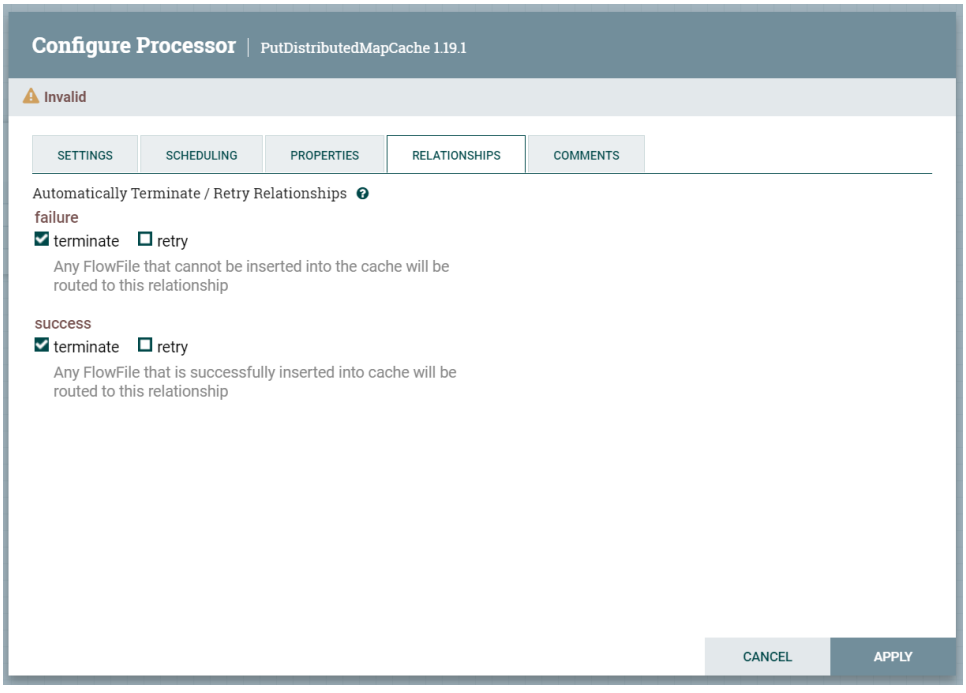

Рисунок 13.16 Relationships

13.6 Запуск процессоров GenerateFlowFile и PutDistributedMapCache

1) Нажмите ПКМ на процессоре `PutDistributedMapCache`, нажмите `Start`.
2) Нажмите ПКМ на процессоре `GenerateFlowFile`, затем `Run Once`.
3) Нажмите ПКМ на самом поле (вне процессоров), затем `Refresh`.
4) Убедитесь про отсутствие ошибок в правом верхнем угле процессора `PutDistributedMapCache`:

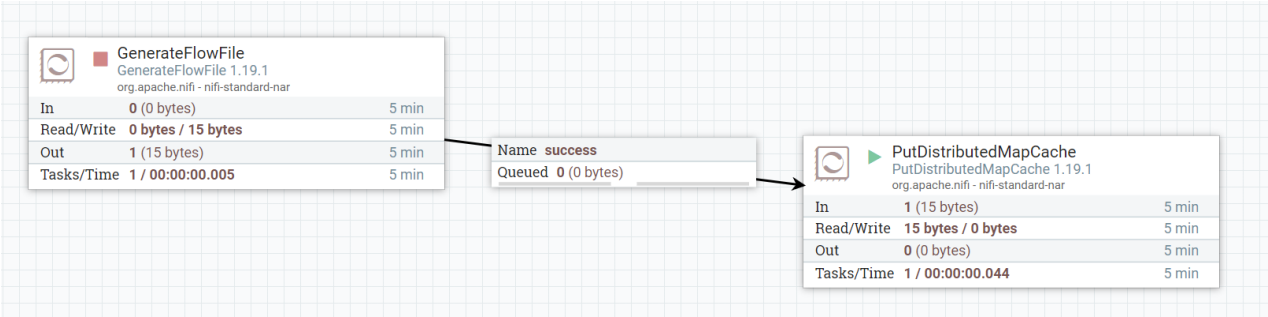

Рисунок 13.17 Relationships

13.7 Тестирование Redis

1) Используйте `Lens` заходим в оболочку (`shell`) пода `zif-redis-infra-master-0.
2) Введите команды:

```
**redis-cli
AUTH "\<пароль администратора Redis>"
KEYS \* →** в результате должен быть выведен ключ **nifi-key
GET nifi-key →** значение ключа должно быть **"hello from nifi"
```

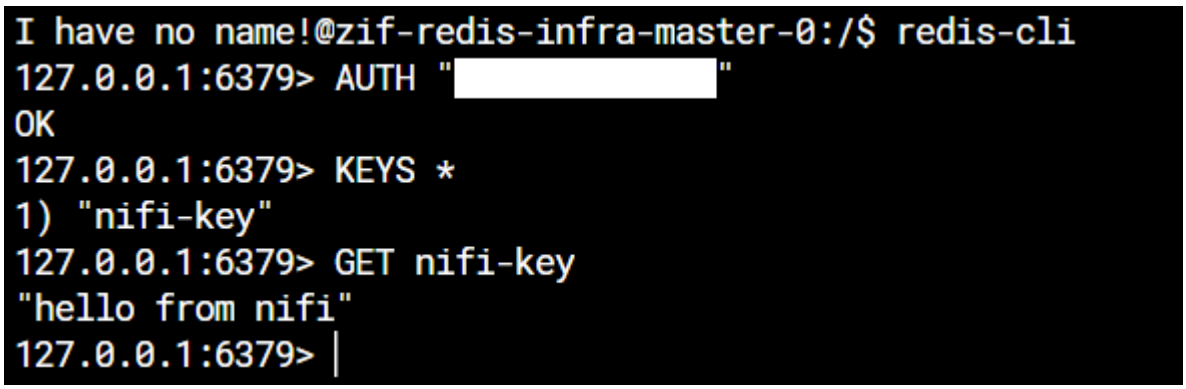

Рисунок 13.18 Тестирование Redis

14 Импорт политик посредством JSON файлов

Начиная с версии корневого образа **zifctl-core:2.20.0**, как альтернатива импорту при помощи Go шаблонов, существует возможность импортировать политики безопасности в форме **JSON** файлов.

Для включения этой функциональности необходимо включить работу с политиками ABAC (см. п. 4.15) на уровне конфигурации стенда:

1. Добавить файлы конфигураций политик в конфигурацию стенда, соблюдая следующие условия:
    - файлы с конфигурацией политик должны находиться в папке **policie**s в конфигурации стенда (**{конфигурация стенда}/values/abac/policies/{имя_сервиса}**);
    - каталог с конфигурацией политик должна называться идентично имени сервиса в **service-list.yaml**, к которому будут применяться политики;
    - текстовые файлы с конфигурацией политик должны иметь расширение ***.json.**

ABAC policies folder

```
└── values
   └── abac
      └── policies
         ├── service1
         ├── service2
         └── service3
            ├── abac-policy-file-1.json
            ├── abac-policy-file-2.json
            └── abac-policy-file-N.json
```

Примечание: версия **zifctl-core:2.20.0** поддерживает только ABAC политики версии v3.

2. Сформировать список "клиентов":

Сервисы с включенным ABAC должны быть указаны в файле **{конфигурация стенда}/values/abac/roles/clients_configuration.json.**

Clients configuration location

```
└── values
   └── abac
      └── roles
        └── clients_configuration.json
```

Пример формирования **clients_configuration.json**:

clietns_configuration.json

```
{
    "clients": {
        "energycontrol-configurator": {
        },
        "energycontrol-monitoring": {
```

```
        },
        "energycontrol-deviationcards": {
        },
        "energycontrol-dynamicrationing": {
        },
        "energycontrol-equipment": {
        },
        "energycontrol-balance": {
        },
        "energycontrol-ramnter": {
        },
        "energycontrol-aspes-adapter": {
        }
    }
}
```

3. Сформировать список "ролей":

Настройки ролей безопасности должны быть описаны в файле **zif-имя-модуля.json** в директории **roles**, где "имя модуля" соответствует имени модуля приложения из **service-list.yaml**:

```
Module name
modules:
  - name: some-app-name
```

Путь к файлу конфигурации ролей безопасности:

```
Roles config
└── values
   └── abac
      └── roles
        └── zif-some-app-name.json
```

15 Импорт политик посредством JSON шаблонов jinja (*.json.j2)

Для импорта ABAC политики в виде **jinja** шаблонов - импортируемый файл должен иметь расширение ***.json.j2** и размещен в каталоге сервиса, для которого включен ABAC (**env-config/values/abac/policies/zif-datasources/zif-jinja-policy.json.j2**). Такую политику можно параметризировать, используя **jinja** код и переменные окружения:

```
jinja policy
```

```
{
  "id": "jinja-policy-{{ env.tenant_name }}",
  "description": "Sample jinja policy for {{ env.tenant_name }}",
  "actions": [
    {
      "id": "read",
      "name": "Чтение"
    },
    {
      "id": "write",
      "name": "Запись"
    },
    {
      "id": "create",
      "name": "Создание"
    },
    {
      "id": "delete",
      "name": "Удаление"
    }
  ],
...
```

16 Интеграция политик в виде JSON шаблонов jinja (*.json.j2) и JSON файлов в образ приложения

В случае, если политики ABAC уместно добавить в образ приложения, то в **Dockerfile** сборки образа приложения должна присутствовать соответствующая инструкция

```
COPY values/abac/policies /platform/init/module/data/abac/policies/
```

при условии, что политики в репозитории структурированы следующим образом:

```
Policies in repo
└── values
    └── abac
        └── policies
```

```
        ├── service1
        ├── service2
        └── service3
            ├── abac-policy-file-1.json
            ├── abac-policy-file-2.json.j2
            ├── abac-policy-file-N.json
            └── abac-policy-file-N.json.j2
```

17 Флаг `configured` в конфигурации инфраструктурных сервисов

Для разрешения или запрета установки сервиса в **env-values.yaml** используется флаг **installed: True/False**, при этом не отменяется генерация переменных окружения и **initJob**, которые относятся к данному сервису. Поэтому добавлен новый флаг **configured** для того, чтобы при необходимости отменить процесс конфигурирования инфраструктурных компонентов Платформы. По умолчанию для всех сервисов установлен параметр **configured: True**.

В **env-values.yaml** это выглядит таким образом:

```
....
postgres:
  installed: true
  configured: true
....
```

Флаг добавлен в **Pydantic** для класса **InfraBaseModel** и его применение, возможно как для текущих сервисов, так и для тех, которые будут добавлены в будущем.

При включении **configured: False** происходит следующее:

1. Происходит стандартная установка сервиса в режиме **single-tenant** или **multi-tenant**.
2. Отключается:
   - генерация переменных окружения из **values/global/global-config.yaml.gotmpl**, **values/global/infra-config.yaml.gotmpl**, **values/global/shared-infra-secrets.yaml.gotmpl**, **values/global/shared-infra-variables.yaml.gotmpl**, **values/defaults.yaml.gotmpl, values/module-defaults.yaml.gotmpl**;
   - монтирование файлов конфигурации (для Keycloak) из **values/global/global-config.yaml.gotmpl**;
   - **initJob** для отключенных сервисов добавляются **в disabledTasksList** и не выполняются.

При использовании флага предусмотрено несколько сценариев использования:

1. **installed: True, configured: True** - вариант по умолчанию, инфраструктурные сервисы Платформы устанавливаются и конфигурируются.
2. **installed: False, configured: True** - вариант внешнего сервиса: Заказчик ставит один или несколько сервисов на своем отдельном сервере или серверах и предоставляет их параметры и учетные данные для выполнения инициализации, создание баз, пользователей

и т.д. В варианте **multi** в тенанте для сервисов **shared-infra** уже проставляется **installed: False, configured: True.**

3. **installed: False, configured: False** - вариант для приложений **appctl**. Необходим для того, чтобы при установке инсталлятором приложения в пространстве имен не конфигурировались неактивные сервисы: не прописывались переменные, не выполнялись задачи **init-job**, не проверялось наличие внешнего сервиса в **startup-guardian** контейнеров сервисов приложения.

18 Ролевая модель Инфраструктурных сервисов (Postgresql)

В состав **Модуля хранения реляционных данных** входит Сервис хранения данных – Postgresql.

Сервис для задач ролевой модели (аутентификации/авторизации) использует логические ресурсы:

- **Сервер Баз Данных** (Server);
- **Пользователь** (Login);
- **База Данных** (DataBase).

Упрощенная концепция ролевой модели Postgresql:

- На **Сервере Баз данных** создаются пользователи и базы данных;
- **Пользователь** может иметь расширенные права (Priveleges) на уровне **Базы Данных** (Superuser, CreateDatabase, CreateRole). Пользователь по умолчанию имеет права на подключение к **Серверу** (Can Login);
- Для **База Данных** есть пользователь-владелец (Owner), имеющий полные права для **Базы Данных;**
- Для каждой **Базы Данных** и для **Пользователя** могут быть заданы отдельные права (Priveleges), помимо **Owner** (например, право на подключение Connect, или на создание объектов Create, Temporary);
- Каждая база **Базы Данных** содержит одну или несколько схем (Schemes), содержащих объекты, такие как таблицы, процедуры, для которых есть пользователь-владелец. Для пользователя могут быть заданы отдельные права (ALL, CREATE, USAGE);
- Схемы включают таблицы, процедуры, для которых есть владелец (Owner), на которые могут быть установлены отдельные права (SELECT, INSERT).

Использование ролевой модели Postgresql для Платформы ZIIoT представлено на Рисунок 18.1 и состоит из следующих возможностей:

- Платформа при развертывании создает **Сервер Баз Данных**. При развертывании **Сервер Баз Данных** создаются несколько служебных пользователей;
- Для каждого сервиса Платформы при развёртывании может быть задан конфигурационный параметр **postgres: true**;
- В случае если указан **postgres:true** для сервиса создается пользователь вида **тенант_имясервиса**, например **ziiot_zif-om-uom**, с автоматически генерируемым паролем. Дополнительные привилегии на уровне сервера не предоставляются;
- Для сервиса создается база данных вида **тенант_имясервиса**, например **ziiot_zif-om-uom** с владельцем базы - пользователем предыдущего этапа;

- Для создаваемых объектов в **Базе Данных** (таких как таблицы) при развертывании, данный пользователь становится владельцем;
- Параметры подключения передаются в сервис в виде полностью сформированной строки подключения в переменной окружения **POSTGRES_CONNECTION_STRING**, так и в виде переменных окружения **POSTGRES_HOST, POSTGRES_DATABASE, POSTGRES_USER, POSTGRES_PASSWORD;**
- Для отслеживания изменений в **Базы Данных** с использованием лога репликации Postgresql для Debezium создается ряд пользователей с предоставлением прав на доступ к ряду баз данных, участвующих в логической репликации.

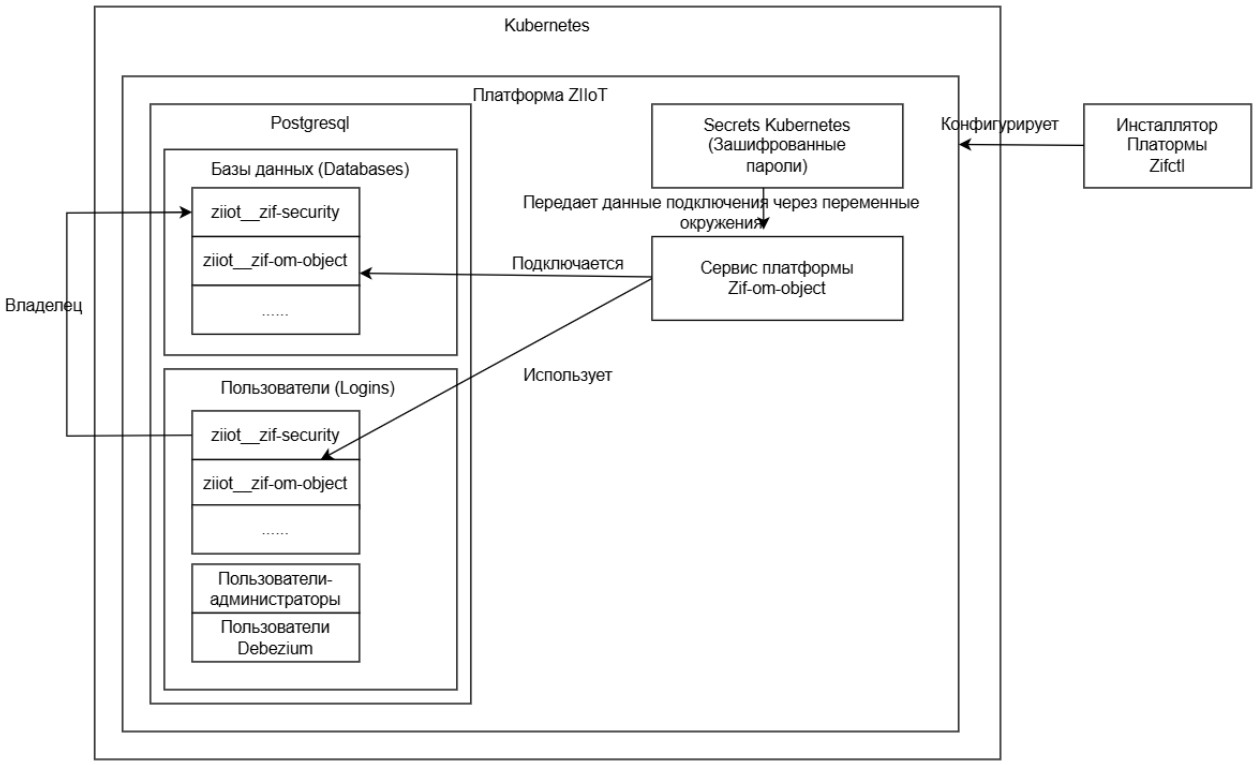

Рисунок 18.1 Использование ролевой модели Postgresql для Платформы ZIIoT

Примечание: для сервиса, в случае необходимости, при развертывании создается пользователь и база данных с полным доступом для этого пользователя. Также создается ряд служебных пользователей, для административных задач.

18.1 Ролевая модель на уровне Сервера Баз Данных

Матрица доступа по умолчанию содержит составные роли, которые представлены в Таблица 18.1

Таблица 18.1. Матрица доступа на уровне Сервера Баз Данных

Роль/Действие	Назначение	Подключение к серверу	Управление репликацией	Все действия на сервере
rolsuper	все права	Да	Да	Да
rolreplication	управление репликацией	Да	Да	Нет
rolcanlogin		Да	Нет	Нет

Роль **rolsuper** создается через команду **SELECT * FROM pg_roles where rolsuper=true** и включает пользователей:

- admin;
- stolon.

Роль **rolreplication** создается через команду **SELECT * FROM pg_roles where rolreplication=true** и включает пользователей:

- repluser;
- ziiot__debezium;
- ziiot__debezium_zif-sm-directories;
- ziiot__debezium_zif-sm-operationdefinition;
- ziiot__debezium_zif-om-objectmodel2excel;
- ziiot__debezium_zif-sm-process;
- ziiot__debezium_zif-om-object.

Примечание: ziiot - название тенанта по умолчанию.

Роль **rolcanlogin** создается через команду **SELECT * FROM pg_roles where rolecanlogin=true** и включает пользователя - тенант_имя сервиса (ziiot_zif-om-object).

18.2 Ролевая модель на уровне Базы Данных

Матрица доступа по умолчанию содержит составные роли и и представлены в Таблица 18.2.

Таблица 18.2. Матрица доступа на уровне Баз Данных

Роль/Действие	Назначение	Подключение Connect	Временные Temporary	Создание Create	Все действия на уровне БД
Поддержка Debezium*	Отслеживание изменений в данных	Да	Да	Да	Нет
Прочие пользователи		Нет	Нет	Нет	Нет
Сервис Платформы (Владелец БД)	Хранение и обработка данных сервиса	Да	Да	Да	Да
Суперпользователь Сервера БД	Задачи администрирования	Да	Да	Да	Да

Примечание:

- *для баз, в которых включена поддержка Debezium, включая следующие базы:
  - ziiot__zif-sm-directories;
  - ziiot__zif-sm-operationdefinition;
  - ziiot__zif-om-objectmodel2excel;
  - ziiot___zif-sm-process;
  - ziiot__zif-om-object.

Проверить доступность действия для пользователя можно с помощью функции запросом вида **SELECT has_database_privilege (usename, databasename, 'CONNECT')**.

Ролевая модель для **Схемы Базы Данных** (**dbo** и для баз с поддержкой Debezium - **dbc**) представлена в **Таблица 18.3**.

Таблица 18.3. Ролевая модель для Схемы Базы Данных

Роль/Действие	Назначение	Создание Create	Использование Usage	Все действия в схеме
Сервис Платформы (владелец схемы)	Хранение и обработка данных сервиса	Да	Да	Да
Поддержка Debezium	Отслеживание изменений в данных	Да	Да	Нет
Суперпользователь Сервера БД	Задачи администрирования	Да	Да	Да
Прочие пользователи		Нет	Нет	Нет

Примечание: проверить доступность действия для пользователя можно с помощью запроса вида **SELECT has_schema_privilege(usename, nspname, 'CREATE')**.

Ролевая модель на **Объекты Базы Данных** представлена в Таблица 18.4.

Таблица 18.4. Ролевая модель для Объектов Базы Данных

Роль/Действие	Назначение	Чтение данных Select	Добавление данных Insert	Изменение данных Update	Удаление данных Delete	Все действия с объектом
Сервис Платформы (владелец объекта)	Хранение и обработка данных сервиса	Да	Да	Да	Да	Да
Поддержка Debezium	Отслеживание изменений в данных	Да	Да	Да	Да	Нет
Суперпользователь Сервера БД	Задачи администрирования	Да	Да	Да	Да	Да
Прочие пользователи		Нет	Нет	Нет	Нет	Нет

Роли на уровне Базы Данных включают пользователей, представленных в таблице Таблица 18.5.

Таблица 18.5. Роли на уровне Базы Данных

Роль БД	Пользователь БД
Сервис Платформы	тенант_имясервиса
Поддержка Debezium	тенант__debezium тенант__debezium_имясервиса
Суперпользователь Сервера БД	Все пользователи сервера БД с привилегией rolsuper
Прочие пользователи	не входящие в приведенные группы

18.3 Перечень БД сервисов

Перечень БД сервисов представлен в Таблица 18.6.

Таблица 18.6. Перечень БД сервисов

Сервис Платформы	База Данных	Пользователь-Владелец БД	Пользователи - поддержка Debezium
zif-om-objectmodel2excel	ziiot__zif-om-objectmodel2excel	ziiot__zif-om-objectmodel2excel	ziiot__debezium_zif-om-objectmodel2excel ziiot__debezium
zif-om-object	ziiot__zif-om-object	ziiot__zif-om-object	ziiot__debezium_zif-om-object ziiot__debezium
zif-sm-directories	ziiot__zif-sm-directories	ziiot__zif-sm-directories	ziiot__debezium_zif-sm-directories ziiot__debezium
zif-sm-operationdefinition	ziiot__zif-sm-operationdefinition	ziiot__zif-sm-operationdefinition	ziiot__debezium_zif-sm-operationdefinition ziiot__debezium
zif-sm-process	ziiot__zif-sm-process	ziiot__zif-sm-process	ziiot__debezium_zif-sm-process ziiot__debezium
keycloak	ziiot__keycloak	ziiot__keycloak	
zif-cm-metadata	ziiot__zif-cm-metadata	ziiot__zif-cm-metadata	
zif-dashboard	ziiot__zif-dashboard	ziiot__zif-dashboard	
zif-data-emulator	ziiot__zif-data-emulator	ziiot__zif-data-emulator	
zif-datalink	ziiot__zif-datalink	ziiot__zif-datalink	
zif-datasources	ziiot__zif-datasources	ziiot__zif-datasources	
zif-events	ziiot__zif-events	ziiot__zif-events	
zif-events-integration	ziiot__zif-events-integration	ziiot__zif-events-integration	
zif-export-nifi-collectors	ziiot__zif-export-nifi-collectors	ziiot__zif-export-nifi-collectors	
zif-hierarchies	ziiot__zif-hierarchies	ziiot__zif-hierarchies	
zif-interface-connect	ziiot__zif-interface-connect	ziiot__zif-interface-connect	
zif-interface-manager	ziiot__zif-interface-manager	ziiot__zif-interface-manager	
zif-licensing	ziiot__zif-licensing	ziiot__zif-licensing	
zif-material-lot	ziiot__zif-material-lot	ziiot__zif-material-lot	
zif-mnemoschemes-storage	ziiot__zif-mnemoschemes-storage	ziiot__zif-mnemoschemes-storage	
zif-notifications	ziiot__zif-notifications	ziiot__zif-notifications	
zif-om-data-observer	ziiot__zif-om-data-observer	ziiot__zif-om-data-observer	
zif-om-datareferences	ziiot__zif-om-datareferences	ziiot__zif-om-datareferences	
zif-om-relationship	ziiot__zif-om-relationship	ziiot__zif-om-relationship	
zif-om-uom	ziiot__zif-om-uom	ziiot__zif-om-uom	
zif-portal-settings	ziiot__zif-portal-settings	ziiot__zif-portal-settings	
zif-propertyset	ziiot__zif-propertyset	ziiot__zif-propertyset	

zif-quality-service	ziiot__zif-quality-service	ziiot__zif-quality-service	
zif-rdm-common	ziiot__zif-rdm-common	ziiot__zif-rdm-common	
zif-reporting	ziiot__zif-reporting	ziiot__zif-reporting	
zif-reporting-sts	ziiot__zif-reporting-sts	ziiot__zif-reporting-sts	
zif-rtdb-metadata	ziiot__zif-rtdb-metadata	ziiot__zif-rtdb-metadata	
zif-security	ziiot__zif-security	ziiot__zif-security	
zif-sm-operationperformance	ziiot__zif-sm-operationperformance	ziiot__zif-sm-operationperformance	
zif-sm-operationsschedule	ziiot__zif-sm-operationsschedule	ziiot__zif-sm-operationsschedule	
zif-sm-testspecification	ziiot__zif-sm-testspecification	ziiot__zif-sm-testspecification	
zif-sm-workcalendar	ziiot__zif-sm-workcalendar	ziiot__zif-sm-workcalendar	
zif-sm-workdefinition	ziiot__zif-sm-workdefinition	ziiot__zif-sm-workdefinition	
zif-sm-workperformance	ziiot__zif-sm-workperformance	ziiot__zif-sm-workperformance	
zif-sm-workschedule	ziiot__zif-sm-workschedule	ziiot__zif-sm-workschedule	
zif-tests	ziiot__zif-tests	ziiot__zif-tests	
zif-universal-datamart	ziiot__zif-universal-datamart	ziiot__zif-universal-datamart	
zif-workflow	ziiot__zif-workflow	ziiot__zif-workflow	

Примечание: при установке с тенантом по умолчанию (**tenantId**) - **ziiot**, иначе вместо **ziiot** в наименовании БД и пользователя применить - **tenantId__имя сервиса**.

Служебные базы данных, созданные по умолчанию в Postgres представлены в Таблица 18.7.

Таблица 18.7. Служебные базы данных

Назначение	База Данных	Пользователь - Владелец БД
БД по умолчанию	postgres	stolon
Шаблон БД	template0	stolon
Шаблон БД	template1	stolon

Служебные Пользователи представлены в Таблица 18.8.

Таблица 18.8. Служебные Пользователи

Назначение	База Данных	Администратор сервера (superuser)
Администратор сервера	admin	Да
УЗ для управления кластером stolon	stolon	Да
Поддержка механизма репликации	repluser	

Приложение 1. Сетевое взаимодействие Платформы ZIIoT

Общее описание принципов сетевого взаимодействия сервисов Платформы ZIIoT:

Группы сервисов на Платформе ZIIoT:

- **Фронтенд - сервисы** - предназначены для формирования визуального пользовательского интерфейса в браузере, для обычных пользователей. Как правило, предоставляют только визуальную разметку и скрипты.
- **Бекенд - сервисы (публичные)** - предназначены для предоставления данных, как правило по REST API. Данные предоставляются как для пользователя, так и возможно для других сервисов.
- **Бекенд - сервисы (непубличные)** - предназначены для предоставления данных, но только для других сервисов.
- **Инфраструктурные сервисы** - сервисы сторонней разработки, предназначенные для хранения/обработки данных. Как правило, не предполагают публичного доступа.
- **Административные сервисы** - предназначены для выполнения административных функций, таких как управления данными/сервисами/платформой в целом, являются вспомогательными. Не предполагается доступ обычных пользователей. Могут использоваться для разграничения административного и пользовательского внешних сегментов, а также для их совместного использования.

Сегменты сети:

- **Сегмент Платформы** (в Kubernetes организовано соответствующее пространство namespace).
- **Сегмент Shared-Infra** (в Kubernetes организовано соответствующее пространство namespace)- в котором может быть размещена часть инфраструктурных сервисов.
- **Внешний сегмент** - куда может быть вынесена часть инфраструктурных сервисов.
- **Пользовательский сегмент** - пользователи и внешние системы.
- **Опциональные Сегменты приложений Платформы** (отдельные пространства namespace в том же кластере Kuberentes, в которых устанавливаются Приложения, использующие сервисы Плафомы).

Схема сетевых сегментов при типовой установке Платформы

Общие сетевые правила

1. Ко всем сервисам, по всем портам напрямую доступ снаружи - запрещен.
2. Всем сервисам по всем портам, напрямую вне кластера - запрещен.
3. Для доступа извне (из Пользовательского сегмента) для фронтенд - сервисов и бекенд - сервисов (публичных) и Административных сервисов создаются **Ingress - контроллеры**. Их список приведен в Таблице 1.
4. Для части Инфраструктурных сервисов в целях доступа снаружи также могут быть созданы **Ingress - контролеры**.
5. В сегменте Платформы оступ между сервисами открыт по портам, объявленным в Service/Endpoints (в направлении In/Out). Список Ingres\Routing контроллеров приведен в Таблице 2.
6. В сегменте **Shared Infra** (если включен) - доступ между сервисами открыт по портам, объявленным в Service/Endpoints (в направлении In/Out). Список приведен в Таблице 2.

7. Между сегментами Платформы и **Shared Infra** (если включен) - доступ открыт по направлению из сегмента Платформы в сегмент **Shared Infra** по портам, приведенным в Таблице 4.

8. В случае развертывания части Инфраструктурных сервисов во Внешнем сегменте, что является проектным решением, доступ настраивается отдельно, по портам, необходимым для конкретного сервиса.

9. Доступ к портам, помеченным меткой **Мониторинг** (и также ко всем HTTP API портам бекенд и фронтенд - сервисов) должен быть предоставлен со стороны опциональной инфраструктуры сборы метрик (например, на базе Prometheus).

10. Для сегмента Приложений по умолчанию доступны сервисы Платформы через **Ingress - контролеры** (см п. 3) и возможен доступ к сегменту **Shared Infra** (если включен) по портам из п. 7.

Таблица 0.1. Общие сетевые правила

Сегмент/Сервис источника	Порты/Протокол источника	Сегмент/Сервис приемник	Порты/Протокол приемника	Правило
Пользовательский сегмент	Все	Сегмент Платформы	Все	Запрещено
Пользовательский сегмент	443/HTTPS	Сегмент Платформы	443/HTTPS	Разрешено через Ingress/Route контроллер, для сервисов, приведенных в Таблице 2
Сегмент Платформы	Все	Пользовательский сегмент	Все	Запрещено
Сегмент Платформы	Указанные в Services/Endpoints (приведены в Таблице 3)	Сегмент Платформы	Указанные в Services/Endpoints (приведены в Таблице 3)	Разрешено
Shared Infra (если включен)	Все	Shared Infra (если включен)	Все	Разрешено
Сегмент Платформы	Все	Shared Infra (если включен)	Порты, протоколы приведены в Таблице 4	Разрешено
Shared Infra (если включен)	Все	Сегмент Платформы	Все	Запрещено

Таблица 0.2. Список Ingres\Routing контроллеров

Сервис	Ingress контроллер	URL (правило)	Тип
zif-cm-engine-mvel	zif-cm-engine-mvel	/zif-cm-engine-mvel(/|$)(.*)	ImplementationSpecific
zif-cm-metadata	zif-cm-metadata	/zif-cm-metadata	Prefix
zif-dashboard	zif-dashboard	/zif-dashboard(/|$)(.*)	ImplementationSpecific
zif-data-emulator	zif-data-emulator	/zif-data-emulator	Prefix
zif-datainput	zif-datainput	/zif-datainput	Prefix
zif-datalink	zif-datalink	/zif-datalink	Prefix
zif-datalink-scripts	zif-datalink-scripts	/zif-datalink-scripts	Prefix
zif-datalink-xl	zif-datalink-xl	/zif-datalink-xl	Prefix
zif-docs	zif-docs	/docs(/|$)(.*)	ImplementationSpecific
zif-document-archive	zif-document-archive	/zif-document-archive	Prefix
zif-events	zif-events	/zif-events	Prefix

zif-events-integration	zif-events-integration	/zif-events-integration	Prefix
zif-export-nifi-collectors	zif-export-nifi-collectors	/zif-export-nifi-collectors	Prefix
zif-file-storage-default	zif-file-storage-default	/files/default	Prefix
zif-hierarchies	zif-hierarchies	/zif-hierarchies(/|$)(.*)	ImplementationSpecific
zif-infra-ingress-zif-kafka-ui	zif-kafka-ui	/kafka-ui	Prefix
zif-infra-ingress-zif-keycloak	zif-keycloak21	/auth	Prefix
zif-infra-ingress-zif-minio-api	zif-minio	/	Prefix
zif-infra-ingress-zif-minio-ui	zif-minio	/	Prefix
zif-infra-ingress-zif-nifi-metrics	zif-nifi-metrics	/nifi-metrics	Prefix
zif-infra-ingress-zif-pgadmin4	zif-pgadmin4	/pgadmin	Prefix
zif-infra-ingress-zif-rabbitmq	zif-rabbitmq	/rabbitmq	Prefix
zif-interface-connect	zif-interface-connect	/zif-interface-connect	Prefix
zif-interface-manager	zif-interface-manager	/zif-interface-manager	Prefix
zif-licensing	zif-licensing	/zif-licensing	Prefix
zif-material-lot	zif-material-lot	/zif-material-lot	Prefix
zif-mnemoschemes-storage	zif-mnemoschemes-storage	/zif-mnemoschemes-storage	Prefix
zif-nifi-ingress	zif-nifi-oauth	/nifi-api	Prefix
zif-nifi-ingress	zif-nifi-oauth	/nifi-content	Prefix
zif-nifi-ingress	zif-nifi-oauth	/nifi-content-viewer	Prefix
zif-nifi-ingress	zif-nifi-oauth	/nifi-docs	Prefix
zif-nifi-ingress	zif-nifi-oauth	/nifi	Prefix
zif-notifications	zif-notifications	/zif-notifications	Prefix
zif-om-data-observer	zif-om-data-observer	/zif-om-data-observer	Prefix
zif-om-datareferences	zif-om-datareferences	/zif-om-datareferences	Prefix
zif-om-graphql	zif-om-graphql	/zif-om-graphql	Prefix
zif-om-hashtags-to-sm-directories	zif-sm-directories	/zif-om-hashtags(/|$)(.*)	ImplementationSpecific
zif-om-object	zif-om-object	/zif-om-object	Prefix
zif-om-objectmodel2excel	zif-om-objectmodel2excel	/zif-om-objectmodel2excel	Prefix
zif-om-relationship	zif-om-relationship	/zif-om-relationship	Prefix
zif-om-sqldatasource	zif-om-sqldatasource	/zif-om-sqldatasource	Prefix
zif-om-testspecification	zif-om-testspecification	/zif-om-testspecification	Prefix
zif-om-uom	zif-om-uom	/zif-om-uom	Prefix
zif-portal-settings	zif-portal-settings	/zif-portal-settings(/|$)(.*)	ImplementationSpecific
zif-propertyset	zif-propertyset	/zif-propertyset	Prefix
zif-rdm-common	zif-rdm-common	/zif-rdm-common	Prefix
zif-reporting	zif-reporting	/zif-reporting(/|$)(.*)	ImplementationSpecific
zif-security	zif-security	/zif-security(/|$)(.*)	ImplementationSpecific
zif-sm-directories	zif-sm-directories	/zif-sm-directories	Prefix
zif-sm-domain-api	zif-sm-domain-api	/zif-sm-domain-api	Prefix
zif-sm-operationdefinition	zif-sm-operationdefinition	/zif-sm-operationdefinition	Prefix

zif-sm-operationperformance	zif-sm-operationperformance	/zif-sm-operationperformance	Prefix
zif-sm-operationsschedule	zif-sm-operationsschedule	/zif-sm-operationsschedule	Prefix
zif-sm-process	zif-sm-process	/zif-sm-process	Prefix
zif-sm-testspecification	zif-sm-testspecification	/zif-sm-testspecification	Prefix
zif-sm-workcalendar	zif-sm-workcalendar	/zif-sm-workcalendar	Prefix
zif-sm-workdefinition	zif-sm-workdefinition	/zif-sm-workdefinition	Prefix
zif-sm-workperformance	zif-sm-workperformance	/zif-sm-workperformance	Prefix
zif-sm-workschedule	zif-sm-workschedule	/zif-sm-workschedule	Prefix
zif-udl-dfawebapi	zif-udl-dfawebapi	/zif-udl-dfawebapi	Prefix
zif-udl-mdswebapi	zif-udl-mdswebapi	/zif-udl-mdswebapi	Prefix
zif-udl-rtd-core	zif-udl-rtd-core	/zif-udl-rtd-core	Prefix
zif-udl-rtdwebapi	zif-udl-rtdwebapi	/zif-udl-rtdwebapi	Prefix
zif-universal-datamart	zif-universal-datamart	/zif-universal-datamart	Prefix
zif-workflow	zif-workflow	/zif-workflow	Prefix
zif-workflow-relocator	zif-workflow-relocator	/zif-workflow-relocator	Prefix
ziiot-tests	ziiot-tests	/ziiot-tests(/|$)(.*)	ImplementationSpecific
zui-app-calculation-service	zui-app-calculation-service	/hostCalculationService(/|$)(.*)	ImplementationSpecific
zui-app-cm-components	zui-app-cm-components	/hostCmComponents(/|$)(.*)	ImplementationSpecific
zui-app-cm-specifications	zui-app-cm-specifications	/hostSpecifications(/|$)(.*)	ImplementationSpecific
zui-app-dashboard-dx	zui-app-dashboard-dx	/hostDashboardDx(/|$)(.*)	ImplementationSpecific
zui-app-datainput	zui-app-datainput	/hostDi(/|$)(.*)	ImplementationSpecific
zui-app-datalinkeditor	zui-app-datalinkeditor	/hostDle(/|$)(.*)	ImplementationSpecific
zui-app-documentexplorer	zui-app-documentexplorer	/hostDx(/|$)(.*)	ImplementationSpecific
zui-app-ds-prop-sql	zui-app-ds-prop-sql	/hostSqlSelect(/|$)(.*)	ImplementationSpecific
zui-app-ds-prop-tag-rtdb	zui-app-ds-prop-tag-rtdb	/hostTagRtdbModal(/|$)(.*)	ImplementationSpecific
zui-app-events-registry	zui-app-events-registry	/hostEventsRegistry(/|$)(.*)	ImplementationSpecific
zui-app-event-types	zui-app-event-types	/hostEventTypes(/|$)(.*)	ImplementationSpecific
zui-app-interface-manager	zui-app-interface-manager	/hostIm(/|$)(.*)	ImplementationSpecific
zui-app-marketplace	zui-app-marketplace	/hostMarket(/|$)(.*)	ImplementationSpecific
zui-app-mnemoeditor	zui-app-mnemoeditor	/hostMnemo(/|$)(.*)	ImplementationSpecific
zui-app-mnemoeditor-res	zui-app-mnemoeditor	/mnemo	Prefix
zui-app-nds-configurator	zui-app-nds-configurator	/hostNdsConfigurator(/|$)(.*)	ImplementationSpecific
zui-app-nifi	zui-app-nifi	/hostDataCollectors(/|$)(.*)	ImplementationSpecific
zui-app-om	zui-app-om	/hostOM(/|$)(.*)	ImplementationSpecific
zui-app-om-sqldatasource	zui-app-om-sqldatasource	/hostSqldatasource(/|$)(.*)	ImplementationSpecific
zui-app-rdm-common	zui-app-rdm-common	/hostRdmCommon(/|$)(.*)	ImplementationSpecific
zui-app-reporteditor	zui-app-reporteditor	/hostReporteditor(/|$)(.*)	ImplementationSpecific
zui-app-security-settings	zui-app-security-settings	/hostSecuritySettings(/|$)(.*)	ImplementationSpecific
zui-app-shell	zui-app-shell	/	Prefix

zui-app-sm-directories	zui-app-sm-directories	/hostOpi(/|$)(.*)	ImplementationSpecific
zui-app-sm-operations	zui-app-sm-operations	/hostOperations(/|$)(.*)	ImplementationSpecific
zui-app-sm-processes	zui-app-sm-processes	/hostProcesses(/|$)(.*)	ImplementationSpecific
zui-app-sm-workdefinitions	zui-app-sm-workdefinitions	/hostWorks(/|$)(.*)	ImplementationSpecific
zui-app-universal-datamart	zui-app-universal-datamart	/hostUd(/|$)(.*)	ImplementationSpecific
zui-app-workflow	zui-app-workflow	/hostWorkflow(/|$)(.*)	ImplementationSpecific

Таблица 0.3. Список Services

Service Name	Port_Protocol	Port	Protocol	Name
rabbitmq-exporter-prometheus-rabbitmq-exporter	9419/TCP	9419	TCP	rabbitmq-exporter
zif-cassandra	9042/TCP	9042	TCP	cql
zif-cassandra	9160/TCP	9160	TCP	thrift
zif-cassandra	8080/TCP	8080	TCP	metrics
zif-cassandra-headless	7000/TCP	7000	TCP	intra
zif-cassandra-headless	7001/TCP	7001	TCP	tls
zif-cassandra-headless	7199/TCP	7199	TCP	jmx
zif-cassandra-headless	9042/TCP	9042	TCP	cql
zif-cassandra-headless	9160/TCP	9160	TCP	thrift
zif-cm-context-functions	80/TCP	80	TCP	http
zif-cm-context-functions	6565/TCP	6565	TCP	grpc
zif-cm-context-functions-headless	6565/TCP	6565	TCP	grpc
zif-cm-engine-mvel	80/TCP	80	TCP	http
zif-cm-engine-mvel	6565/TCP	6565	TCP	grpc
zif-cm-engine-mvel-headless	6565/TCP	6565	TCP	grpc
zif-cm-metadata	80/TCP	80	TCP	http
zif-dashboard	80/TCP	80	TCP	http
zif-data-emulator	80/TCP	80	TCP	http
zif-datainput	80/TCP	80	TCP	http
zif-datalink	80/TCP	80	TCP	http
zif-datalink-scripts	80/TCP	80	TCP	http
zif-datalink-xl	80/TCP	80	TCP	http
zif-docs	80/TCP	80	TCP	http
zif-document-archive	80/TCP	80	TCP	http
zif-events	80/TCP	80	TCP	http
zif-events-integration	80/TCP	80	TCP	http
zif-export-nifi-collectors	80/TCP	80	TCP	http
zif-file-storage-default	80/TCP	80	TCP	http
zif-hierarchies	80/TCP	80	TCP	http
zif-interface-connect	80/TCP	80	TCP	http
zif-interface-manager	80/TCP	80	TCP	http
zif-kafka-connect-cp-kafka-connect	8083/TCP	8083	TCP	kafka-connect
zif-kafka-connect-cp-kafka-connect	5556/TCP	5556	TCP	metrics
zif-kafka-cp-kafka	9092/TCP	9092	TCP	internal
zif-kafka-cp-kafka	5556/TCP	5556	TCP	metrics
zif-kafka-cp-kafka	9308/TCP	9308	TCP	exporter
zif-kafka-cp-kafka-headless	9093/TCP	9093	TCP	broker
zif-kafka-cp-kafka-headless	9092/TCP	9092	TCP	internal

zif-kafka-rest-cp-kafka-rest	8082/TCP	8082	TCP	rest-proxy
zif-kafka-rest-cp-kafka-rest	5556/TCP	5556	TCP	metrics
zif-kafka-schema-cp-schema-registry	8081/TCP	8081	TCP	schema-registry
zif-kafka-schema-cp-schema-registry	5556/TCP	5556	TCP	metrics
zif-kafka-ui	80/TCP	80	TCP	http
zif-kafka-zookeeper-cp-zookeeper	2181/TCP	2181	TCP	client
zif-kafka-zookeeper-cp-zookeeper	5556/TCP	5556	TCP	metrics
zif-kafka-zookeeper-cp-zookeeper-headless	2181/TCP	2181	TCP	client
zif-kafka-zookeeper-cp-zookeeper-headless	2888/TCP	2888	TCP	server
zif-kafka-zookeeper-cp-zookeeper-headless	3888/TCP	3888	TCP	leader-election
zif-keycloak21	80/TCP	80	TCP	http
zif-keycloak21-headless	80/TCP	80	TCP	http
zif-keycloak21-metrics	8080/TCP	8080	TCP	http
zif-licensing	443/TCP	443	TCP	https
zif-licensing	80/TCP	80	TCP	http
zif-material-lot	80/TCP	80	TCP	http
zif-minio	9000/TCP	9000	TCP	minio-api
zif-minio	9001/TCP	9001	TCP	minio-console
zif-mnemoschemes-storage	80/TCP	80	TCP	http
zif-nifi	8080/TCP	8080	TCP	http
zif-nifi-headless	8080/TCP	8080	TCP	http
zif-nifi-headless	6007/TCP	6007	TCP	cluster
zif-nifi-metrics	9092/TCP	9092	TCP	metrics
zif-nifi-oauth	80/TCP	80	TCP	http
zif-notifications	80/TCP	80	TCP	http
zif-om-data-observer	80/TCP	80	TCP	http
zif-om-datareferences	80/TCP	80	TCP	http
zif-om-graphql	80/TCP	80	TCP	http
zif-om-object	80/TCP	80	TCP	http
zif-om-objectmodel2excel	80/TCP	80	TCP	http
zif-om-properties	80/TCP	80	TCP	http
zif-om-relationship	80/TCP	80	TCP	http
zif-om-sqldatasource	80/TCP	80	TCP	http
zif-om-testspecification	80/TCP	80	TCP	http
zif-om-uom	80/TCP	80	TCP	http
zif-opcua-server	4840/TCP	4840	TCP	opcua
zif-opcua-server	80/TCP	80	TCP	http
zif-opcua-server-headless	4840/TCP	4840	TCP	opcua
zif-pgadmin4	80/TCP	80	TCP	http
zif-portal-settings	80/TCP	80	TCP	http
zif-postgres	5432/TCP	5432	TCP	postgres
zif-postgres	8080/TCP	8080	TCP	metrics
zif-postgres14-keeper	5432/TCP	5432	TCP	postgres
zif-postgres14-keeper	8080/TCP	8080	TCP	metrics
zif-postgres14-sentinel	8080/TCP	8080	TCP	metrics
zif-propertyset	80/TCP	80	TCP	http
zif-rabbitmq	5672/TCP	5672	TCP	amqp
zif-rabbitmq	4369/TCP	4369	TCP	epmd
zif-rabbitmq	25672/TCP	25672	TCP	dist
zif-rabbitmq	15672/TCP	15672	TCP	http-stats
zif-rabbitmq	9419/TCP	9419	TCP	metrics

zif-rabbitmq-headless	4369/TCP	4369	TCP	epmd
zif-rabbitmq-headless	5672/TCP	5672	TCP	amqp
zif-rabbitmq-headless	25672/TCP	25672	TCP	dist
zif-rabbitmq-headless	15672/TCP	15672	TCP	http-stats
zif-rdm-common	80/TCP	80	TCP	http
zif-redis-headless	6379/TCP	6379	TCP	tcp-redis
zif-redis-master	6379/TCP	6379	TCP	tcp-redis
zif-redis-master	6380/TCP	6380	TCP	tcp-redis-tls
zif-redis-metrics	9121/TCP	9121	TCP	http-metrics
zif-redis-replicas	6379/TCP	6379	TCP	tcp-redis
zif-redis-replicas	6380/TCP	6380	TCP	tcp-redis-tls
zif-reporting	80/TCP	80	TCP	http
zif-security	80/TCP	80	TCP	http
zif-sm-directories	80/TCP	80	TCP	http
zif-sm-domain-api	80/TCP	80	TCP	http
zif-sm-operationdefinition	80/TCP	80	TCP	http
zif-sm-operationperformance	80/TCP	80	TCP	http
zif-sm-operationsschedule	80/TCP	80	TCP	http
zif-sm-process	80/TCP	80	TCP	http
zif-sm-testspecification	80/TCP	80	TCP	http
zif-sm-workcalendar	80/TCP	80	TCP	http
zif-sm-workdefinition	80/TCP	80	TCP	http
zif-sm-workperformance	80/TCP	80	TCP	http
zif-sm-workschedule	80/TCP	80	TCP	http
zif-udl-dfawebapi	80/TCP	80	TCP	http
zif-udl-mdswebapi	80/TCP	80	TCP	http
zif-udl-propertychangelistener	80/TCP	80	TCP	http
zif-udl-rtd-core	80/TCP	80	TCP	http
zif-udl-rtdwebapi	80/TCP	80	TCP	http
zif-universal-datamart	80/TCP	80	TCP	http
zif-workflow	80/TCP	80	TCP	http
zif-workflow-relocator	80/TCP	80	TCP	http
ziiot-tests	80/TCP	80	TCP	http
zui-app-calculation-service	80/TCP	80	TCP	http
zui-app-cm-components	80/TCP	80	TCP	http
zui-app-cm-specifications	80/TCP	80	TCP	http
zui-app-dashboard-dx	80/TCP	80	TCP	http
zui-app-datainput	80/TCP	80	TCP	http
zui-app-datalinkeditor	80/TCP	80	TCP	http
zui-app-documentexplorer	80/TCP	80	TCP	http
zui-app-ds-prop-sql	80/TCP	80	TCP	http
zui-app-ds-prop-tag-rtdb	80/TCP	80	TCP	http
zui-app-events-registry	80/TCP	80	TCP	http
zui-app-event-types	80/TCP	80	TCP	http
zui-app-interface-manager	80/TCP	80	TCP	http
zui-app-marketplace	80/TCP	80	TCP	http
zui-app-mnemoeditor	80/TCP	80	TCP	http
zui-app-nds-configurator	80/TCP	80	TCP	http
zui-app-nifi	80/TCP	80	TCP	http
zui-app-om	80/TCP	80	TCP	http
zui-app-om-sqldatasource	80/TCP	80	TCP	http

zui-app-rdm-common	80/TCP	80	TCP	http
zui-app-reporteditor	80/TCP	80	TCP	http
zui-app-security-settings	80/TCP	80	TCP	http
zui-app-shell	80/TCP	80	TCP	http
zui-app-sm-directories	80/TCP	80	TCP	http
zui-app-sm-operations	80/TCP	80	TCP	http
zui-app-sm-processes	80/TCP	80	TCP	http
zui-app-sm-workdefinitions	80/TCP	80	TCP	http
zui-app-universal-datamart	80/TCP	80	TCP	http
zui-app-workflow	80/TCP	80	TCP	http

Таблица 0.4. Список открытых портов для доступа между сегментами Shared Infra (направление In) и остальными сегментами

сервис	порт	протокол
Keycloak	80, 443	TCP
cassandra	9042	TCP
postgresql	5432	TCP
rabbitmq	5672,15672	TCP
miniio	9000, 9001	TCP

Приложение 2. Список внешних URL Платформы/ZIIoT

Описание

В процессе развертывания, сервис **zifctl** генерирует следующие типы URL (доступных снаружи Платформы):

- **Административные UI** - для выполнения служебных задач администратором Платформы;
- **Пользовательские UI** - для формирования пользовательского интерфейса Пользователя;
- **Интеграционные** – для API сервисов.

Внешние URL

Административные

Наименование	Сервис	URL
zif-infra-ingress-zif-kafka-ui	zif-kafka-ui	/zif-kafka-ui
zif-infra-ingress-zif-minio-ui	zif-minio	/zif-minio
zif-infra-ingress-zif-pgadmin4	zif-pgadmin4	/zif-pgadmin4

Интеграционные

Наименование	Сервис	URL
zif-cm-engine-mvel	zif-cm-engine-mvel	/zif-cm-engine-mvel
zif-cm-metadata	zif-cm-metadata	/zif-cm-metadata
zif-dashboard	zif-dashboard	/zif-dashboard
zif-datainput	zif-datainput	/zif-datainput
zif-datalink	zif-datalink	/zif-datalink
zif-datalink-scripts	zif-datalink-scripts	/zif-datalink-scripts
zif-datalink-xl	zif-datalink-xl	/zif-datalink-xl
zif-docs	zif-docs	/zif-docs
zif-document-archive	zif-document-archive	/zif-document-archive
zif-events	zif-events	/zif-events
zif-events-integration	zif-events-integration	/zif-events-integration
zif-export-nifi-collectors	zif-export-nifi-collectors	/zif-export-nifi-collectors
zif-infra-ingress-zif-keycloak	zif-keycloak-http	/auth
zif-infra-ingress-zif-minio-api	zif-minio	/zif-minio
zif-infra-ingress-zif-nifi-api	nifi	/nifi-api
zif-infra-ingress-zif-nifi-content	nifi	/nifi-content
zif-infra-ingress-zif-nifi-content-viewer	nifi	/nifi-content-viewer
zif-infra-ingress-zif-nifi-docs	nifi	/nifi-docs
zif-infra-ingress-zif-nifi-metrics	zif-nifi-metrics	/zif-nifi-metrics
zif-infra-ingress-zif-nifi-ui	nifi	/nifi
zif-infra-ingress-zif-rabbitmq	zif-rabbitmq	/zif-rabbitmq
zif-interface-manager	zif-interface-manager	/zif-interface-manager
zif-licensing	zif-licensing	/zif-licensing

zif-material-lot	zif-material-lot	/zif-material-lot
zif-mnemoschemes-storage	zif-mnemoschemes-storage	/zif-mnemoschemes-storage
zif-notifications	zif-notifications	/zif-notifications
zif-om-b2mml	zif-om-b2mml	/zif-om-b2mml
zif-om-datareferences	zif-om-datareferences	/zif-om-datareferences
zif-om-graphql	zif-om-graphql	/zif-om-graphql
zif-om-hashtags	zif-om-hashtags	/zif-om-hashtags
zif-om-object	zif-om-object	/zif-om-object
zif-om-objectmodel2excel	zif-om-objectmodel2excel	/zif-om-objectmodel2excel
zif-om-process	zif-om-process	/zif-om-process
zif-om-properties	zif-om-properties	/zif-om-properties
zif-om-properties-view	zif-om-properties-view	/zif-om-properties-view
zif-om-relationship	zif-om-relationship	/zif-om-relationship
zif-om-sqldatasource	zif-om-sqldatasource	/zif-om-sqldatasource
zif-om-testspecification	zif-om-testspecification	/zif-om-testspecification
zif-om-uom	zif-om-uom	/zif-om-uom
zif-om-valuetypes	zif-om-valuetypes	/zif-om-valuetypes
zif-portal-settings	zif-portal-settings	/zif-portal-settings
zif-propertyset	zif-propertyset	/zif-propertyset
zif-quality-service	zif-quality-service	/zif-quality-service
zif-rdm-common	zif-rdm-common	/zif-rdm-common
zif-reporting	zif-reporting	/zif-reporting
zif-rtdb-data	zif-rtdb-data	/zif-rtdb-data
zif-rtdb-data-consumerhost	zif-rtdb-data-consumerhost	/zif-rtdb-data-consumerhost
zif-rtdb-metadata	zif-rtdb-metadata	/zif-rtdb-metadata
zif-security	zif-security	/zif-security
zif-sm-directories	zif-sm-directories	/zif-sm-directories
zif-sm-domain-api	zif-sm-domain-api	/zif-sm-domain-api
zif-sm-operationdefinition	zif-sm-operationdefinition	/zif-sm-operationdefinition
zif-sm-operationperformance	zif-sm-operationperformance	/zif-sm-operationperformance
zif-sm-operationsschedule	zif-sm-operationsschedule	/zif-sm-operationsschedule
zif-sm-process	zif-sm-process	/zif-sm-process
zif-sm-testspecification	zif-sm-testspecification	/zif-sm-testspecification
zif-sm-workcalendar	zif-sm-workcalendar	/zif-sm-workcalendar
zif-sm-workdefinition	zif-sm-workdefinition	/zif-sm-workdefinition
zif-sm-workperformance	zif-sm-workperformance	/zif-sm-workperformance
zif-sm-workschedule	zif-sm-workschedule	/zif-sm-workschedule
zif-udl-dfawebapi	zif-udl-dfawebapi	/zif-udl-dfawebapi
zif-udl-dsentitywebapi	zif-udl-dsentitywebapi	/zif-udl-dsentitywebapi
zif-udl-mdswebapi	zif-udl-mdswebapi	/zif-udl-mdswebapi
zif-udl-rtdwebapi	zif-udl-rtdwebapi	/zif-udl-rtdwebapi
zif-universal-datamart	zif-universal-datamart	/zif-universal-datamart
zif-workflow	zif-workflow	/zif-workflow
zif-workflow-relocator	zif-workflow-relocator	/zif-workflow-relocator

Пользовательские

Наименование	Сервис	URL
zui-app-shell	zui-app-shell	/
zui-app-calculation-service	zui-app-calculation-service	/hostCalculationService
zui-app-dashboard-dx	zui-app-dashboard-dx	/hostDashboardDx

zui-app-datainput	zui-app-datainput	/hostDi
zui-app-documentexplorer	zui-app-documentexplorer	/hostDx
zui-app-events-registry	zui-app-events-registry	/hostEventRegistry
zui-app-event-types	zui-app-event-types	/hostEventTypes
zui-app-interface-manager	zui-app-interface-manager	/hostIM
zui-app-mnemoeditor	zui-app-mnemoeditor	/hostMnemo
zui-app-nds-configurator	zui-app-nds-configurator	/hostNdsConfigurator
zui-app-nifi	zui-app-nifi	/hostOM
zui-app-rdm-common	zui-app-rdm-common	/hostrdmcommon
zui-app-reporteditor	zui-app-reporteditor	/hostReporteditor
zui-app-security-settings	zui-app-security-settings	/hostSecuritySettings
zui-app-rtdb	zui-app-rtdb	/hostTsds
zui-app-universal-datamart	zui-app-universal-datamart	/hostUd
zui-app-workflow	zui-app-workflow	/hostWorkflow
zui-app-mnemoeditor-res	zui-app-mnemoeditor	/mnemo
zui-app-om	zui-app-om	/zui-app-om
zui-app-quality-service	zui-app-quality-service	hostqualityservice

Приложение 3. Список учетных записей Платформы ZIIoT

Служебные УЗ в рамках realm ziiot Таблица 0.1

№	Учётная запись	Описание (назначение / требования)	Полномочия (чтение \ запись)
1	admin	Сервисная интерактивная УЗ суперпользователя сервиса keycloak Платформы ZIIoT	Полный административный доступ к управлению всеми realm, включая master сервиса keycloak
2	realm-admin	Сервисная интерактивная УЗ суперпользователя с ограниченными привилегиями сервиса keycloak, входящего в состав Платформы ZIIoT	Полный административный доступ к управлению realm ziiot сервиса keycloak
3	kafka-admin	Сервисная интерактивная УЗ суперпользователя сервиса kafka Платформы ZIIoT	Полный административный доступ к сервису kafka в рамках realm ziiot
4	nifi-admin	Сервисная интерактивная УЗ суперпользователя сервиса nifi Платформы ZIIoT	Полный административный доступ к сервису nifi в рамках realm ziiot
5	apache-nifi-client	Клиентская УЗ сервиса nifi	Обеспечение межсервисного взаимодействия сервиса nifi в рамках realm ziiot
6	kafka-ui	Клиентская УЗ сервиса kafka	Обеспечение межсервисного взаимодействия сервиса kafka в рамках realm ziiot
7	test-client	Клиентская УЗ для доступа к restAPI прочих сервисов	Обеспечение межсервисного взаимодействия swagger ui с restAPI интерфейсами прочих сервисов Платформы ZIIoT в рамках realm ziiot
8	bp-workers-init-job	Клиентская УЗ init job сервиса bp-workers	Обеспечение межсервисного взаимодействия сервиса bp-workers в процессе его развертывания в рамках realm ziiot
9	calc-init-job	Клиентская УЗ init job сервиса calc	Обеспечение межсервисного взаимодействия сервиса calc в процессе его развертывания в рамках realm ziiot
10	documents-init-job	Клиентская УЗ init job сервиса documents	Обеспечение межсервисного взаимодействия сервиса documents в процессе его развертывания в рамках realm ziiot
11	licensing-init-job	Клиентская УЗ init job сервиса licensing	Обеспечение межсервисного взаимодействия сервиса licensing в процессе его развертывания в рамках realm ziiot
12	events-init-job	Клиентская УЗ init job сервиса events	Обеспечение межсервисного взаимодействия сервиса events в процессе его развертывания в рамках realm ziiot

13	mnemoscheme-init-job	Клиентская УЗ init job сервиса mnemoscheme	Обеспечение межсервисного взаимодействия сервиса mnemoscheme в процессе его развертывания в рамках realm ziiot
14	notifications-init-job	Клиентская УЗ init job сервиса notifications	Обеспечение межсервисного взаимодействия сервиса notifications в процессе его развертывания в рамках realm ziiot
15	om-init-job	Клиентская УЗ init job сервиса om	Обеспечение межсервисного взаимодействия сервиса om в процессе его развертывания в рамках realm ziiot
16	portal-init-job	Клиентская УЗ init job сервиса portal	Обеспечение межсервисного взаимодействия сервиса portal в процессе его развертывания в рамках realm ziiot
17	process-init-job	Клиентская УЗ init job сервиса process	Обеспечение межсервисного взаимодействия сервиса process в процессе его развертывания в рамках realm ziiot
18	rtdb-init-job	Клиентская УЗ init job сервиса rtdb	Обеспечение межсервисного взаимодействия сервиса rtdb в процессе его развертывания в рамках realm ziiot
19	security-init-job	Клиентская УЗ init job сервиса security	Обеспечение межсервисного взаимодействия сервиса security в процессе его развертывания в рамках realm ziiot
20	sm-init-job	Клиентская УЗ init job сервиса sm	Обеспечение межсервисного взаимодействия сервиса sm в процессе его развертывания в рамках realm ziiot
21	udl-init-job	Клиентская УЗ init job сервиса udl	Обеспечение межсервисного взаимодействия сервиса udl в процессе его развертывания в рамках realm ziiot
22	zif-bp-calculate-specification-worker	Клиентская УЗ для процессоров Camunda сервиса zif-bp-calculate-specification	Обеспечение межсервисного взаимодействия сервиса zif-bp-calculate-specification в ходе обработки Camunda в рамках realm ziiot
23	zif-bp-dataworker	Клиентская УЗ для процессоров Camunda сервиса zif-bp-data	Обеспечение межсервисного взаимодействия сервиса zif-bp-data в ходе обработки Camunda в рамках realm ziiot
24	zif-bp-events-worker	Клиентская УЗ для процессоров Camunda сервиса zif-bp-events	Обеспечение межсервисного взаимодействия сервиса zif-bp-events в ходе обработки Camunda в рамках realm ziiot
25	zif-bp-material-worker	Клиентская УЗ для процессоров Camunda сервиса zif-bp-material	Обеспечение межсервисного взаимодействия сервиса zif-bp-material в ходе обработки Camunda в рамках realm ziiot

26	zif-bp-message-bus-worker	Клиентская УЗ для процессоров Camunda сервиса zif-bp-message-bus	Обеспечение межсервисного взаимодействия сервиса zif-bp-message-bus в ходе обработки Camunda в рамках realm ziiot
27	zif-bp-notifications-worker	Клиентская УЗ для процессоров Camunda сервиса zif-bp-notifications	Обеспечение межсервисного взаимодействия сервиса zif-bp-notifications в ходе обработки Camunda в рамках realm ziiot
28	zif-bp-objects-worker	Клиентская УЗ для процессоров Camunda сервиса zif-bp-objects	Обеспечение межсервисного взаимодействия сервиса zif-bp-objects в ходе обработки Camunda в рамках realm ziiot
29	zif-bp-properties-worker	Клиентская УЗ для процессоров Camunda сервиса zif-bp-properties	Обеспечение межсервисного взаимодействия сервиса zif-bp-properties в ходе обработки Camunda в рамках realm ziiot
30	zif-bp-reference-worker	Клиентская УЗ для процессоров Camunda сервиса zif-bp-reference	Обеспечение межсервисного взаимодействия сервиса zif-bp-reference в ходе обработки Camunda в рамках realm ziiot
31	zif-bp-sm-directories-worker	Клиентская УЗ для процессоров Camunda сервиса zif-bp-sm-directories	Обеспечение межсервисного взаимодействия сервиса zif-bp-sm-directories в ходе обработки Camunda в рамках realm ziiot
32	zif-bp-sm-operation-definition-worker	Клиентская УЗ для процессоров Camunda сервиса zif-bp-sm-operation-definition	Обеспечение межсервисного взаимодействия сервиса zif-bp-sm-operation-definition в ходе обработки Camunda в рамках realm ziiot
33	zif-bp-sm-operation-performance-worker	Клиентская УЗ для процессоров Camunda сервиса zif-bp-sm-operation-performance	Обеспечение межсервисного взаимодействия сервиса zif-bp-sm-operation-performance в ходе обработки Camunda в рамках realm ziiot
34	zif-bp-sm-operations-schedule-worker	Клиентская УЗ для процессоров Camunda сервиса zif-bp-sm-operations-schedule	Обеспечение межсервисного взаимодействия сервиса zif-bp-sm-operations-schedule в ходе обработки Camunda в рамках realm ziiot
35	zif-bp-sm-work-definition-worker	Клиентская УЗ для процессоров Camunda сервиса zif-bp-sm-work-definition	Обеспечение межсервисного взаимодействия сервиса zif-bp-sm-work-definition в ходе обработки Camunda в рамках realm ziiot
36	zif-bp-sm-work-performance-worker	Клиентская УЗ для процессоров Camunda сервиса zif-bp-sm-work-performance	Обеспечение межсервисного взаимодействия сервиса zif-bp-sm-work-performance в ходе обработки Camunda в рамках realm ziiot
37	zif-bp-sm-work-schedule-worker	Клиентская УЗ для процессоров Camunda сервиса zif-bp-sm-work-schedule	Обеспечение межсервисного взаимодействия сервиса zif-bp-sm-work-schedule в ходе обработки Camunda в рамках realm ziiot
52	zif-attributes-directory	Клиентская УЗ сервиса zif-attributes-directory	Обеспечение межсервисного взаимодействия сервиса zif-attributes-directory в рамках realm ziiot

53	zif-cm-engine-mvel	Клиентская УЗ сервиса zif-cm-engine-mvel	Обеспечение межсервисного взаимодействия сервиса zif-cm-engine-mvel в рамках realm ziiot
54	zif-cm-metadata	Клиентская УЗ сервиса zif-cm-metadata	Обеспечение межсервисного взаимодействия сервиса zif-cm-metadata в рамках realm ziiot
55	zif-cm-provider-udl-service	Клиентская УЗ сервиса zif-cm-provider-udl-service	Обеспечение межсервисного взаимодействия сервиса zif-cm-provider-udl-service в рамках realm ziiot
56	zif-dashboard	Клиентская УЗ сервиса zif-dashboard	Обеспечение межсервисного взаимодействия сервиса zif-dashboard в рамках realm ziiot
57	zif-datainput	Клиентская УЗ сервиса zif-datainput	Обеспечение межсервисного взаимодействия сервиса zif-datainput в рамках realm ziiot
58	zif-datalink	Клиентская УЗ сервиса zif-datalink	Обеспечение межсервисного взаимодействия сервиса zif-datalink в рамках realm ziiot
59	zif-datalink-xl	Клиентская УЗ сервиса zif-datalink-xl	Обеспечение межсервисного взаимодействия сервиса zif-datalink-xl в рамках realm ziiot
60	zif-document-archive	Клиентская УЗ сервиса zif-document-archive	Обеспечение межсервисного взаимодействия сервиса zif-document-archive в рамках realm ziiot
61	zif-document-storage	Клиентская УЗ сервиса zif-document-storage	Обеспечение межсервисного взаимодействия сервиса zif-document-storage в рамках realm ziiot
62	zif-events-integration	Клиентская УЗ сервиса zif-events-integration	Обеспечение межсервисного взаимодействия сервиса zif-events-integration в рамках realm ziiot
63	zif-events	Клиентская УЗ сервиса zif-events	Обеспечение межсервисного взаимодействия сервиса zif-events в рамках realm ziiot
64	zif-licensing	Клиентская УЗ сервиса zif-licensing	Обеспечение межсервисного взаимодействия сервиса zif-licensing в рамках realm ziiot
65	zif-material-lot	Клиентская УЗ сервиса zif-material-lot	Обеспечение межсервисного взаимодействия сервиса zif-material-lot в рамках realm ziiot
66	zif-mnemoschemes-storage	Клиентская УЗ сервиса zif-mnemoschemes-storage	Обеспечение межсервисного взаимодействия сервиса zif-mnemoschemes-storage в рамках realm ziiot
67	zif-notifications	Клиентская УЗ сервиса zif-notifications	Обеспечение межсервисного взаимодействия сервиса zif-notifications в рамках realm ziiot
68	zif-om-b2mml	Клиентская УЗ сервиса zif-om-b2mml	Обеспечение межсервисного взаимодействия сервиса zif-om-b2mml в рамках realm ziiot
69	zif-om-datareferences	Клиентская УЗ сервиса zif-om-datareferences	Обеспечение межсервисного взаимодействия сервиса zif-om-datareferences в рамках realm ziiot

70	zif-om-graphql	Клиентская УЗ сервиса zif-om-graphql	Обеспечение межсервисного взаимодействия сервиса zif-om-graphql в рамках realm ziiot
71	zif-om-hashtags	Клиентская УЗ сервиса zif-om-hashtags	Обеспечение межсервисного взаимодействия сервиса zif-om-hashtags в рамках realm ziiot
72	zif-om-objectmodel2excel	Клиентская УЗ сервиса zif-om-objectmodel2excel	Обеспечение межсервисного взаимодействия сервиса zif-om-objectmodel2excel в рамках realm ziiot
73	zif-om-object	Клиентская УЗ сервиса zif-om-object	Обеспечение межсервисного взаимодействия сервиса zif-om-object в рамках realm ziiot
74	zif-om-process	Клиентская УЗ сервиса zif-om-process	Обеспечение межсервисного взаимодействия сервиса zif-om-process в рамках realm ziiot
75	zif-om-properties-view	Клиентская УЗ сервиса zif-om-properties-view	Обеспечение межсервисного взаимодействия сервиса zif-om-properties-view в рамках realm ziiot
76	zif-om-propertydata	Клиентская УЗ сервиса zif-om-propertydata	Обеспечение межсервисного взаимодействия сервиса zif-om-propertydata в рамках realm ziiot
77	zif-om-relationship	Клиентская УЗ сервиса zif-om-relationship	Обеспечение межсервисного взаимодействия сервиса zif-om-relationship в рамках realm ziiot
78	zif-om-sqldatasource	Клиентская УЗ сервиса zif-om-sqldatasource	Обеспечение межсервисного взаимодействия сервиса zif-om-sqldatasource в рамках realm ziiot
79	zif-om-testspecification	Клиентская УЗ сервиса zif-om-testspecification	Обеспечение межсервисного взаимодействия сервиса zif-om-testspecification в рамках realm ziiot
80	zif-om-uom	Клиентская УЗ сервиса zif-om-uom	Обеспечение межсервисного взаимодействия сервиса zif-om-uom в рамках realm ziiot
81	zif-om-valuetypes	Клиентская УЗ сервиса zif-om-valuetypes	Обеспечение межсервисного взаимодействия сервиса zif-om-valuetypes в рамках realm ziiot
82	zif-portal-settings	Клиентская УЗ сервиса zif-portal-settings	Обеспечение межсервисного взаимодействия сервиса zif-portal-settings в рамках realm ziiot
83	zif-propertyset	Клиентская УЗ сервиса zif-propertyset	Обеспечение межсервисного взаимодействия сервиса zif-propertyset в рамках realm ziiot
84	zif-quality-service	Клиентская УЗ сервиса zif-quality-service	Обеспечение межсервисного взаимодействия сервиса zif-quality-service в рамках realm ziiot
85	zif-rdm-common	Клиентская УЗ сервиса zif-rdm-common	Обеспечение межсервисного взаимодействия сервиса zif-rdm-common в рамках realm ziiot
86	zif-reporting	Клиентская УЗ сервиса zif-reporting	Обеспечение межсервисного взаимодействия сервиса zif-reporting в рамках realm ziiot

87	zif-rtdb-data-consumerhost	Клиентская УЗ сервиса zif-rtdb-data-consumerhost	Обеспечение межсервисного взаимодействия сервиса zif-rtdb-data-consumerhost в рамках realm ziiot
88	zif-rtdb-data	Клиентская УЗ сервиса zif-rtdb-data	Обеспечение межсервисного взаимодействия сервиса zif-rtdb-data в рамках realm ziiot
89	zif-rtdb-metadata	Клиентская УЗ сервиса zif-rtdb-metadata	Обеспечение межсервисного взаимодействия сервиса zif-rtdb-metadata в рамках realm ziiot
90	zif-security	Клиентская УЗ сервиса zif-security	Обеспечение межсервисного взаимодействия сервиса zif-security в рамках realm ziiot
91	zif-sm-directories	Клиентская УЗ сервиса zif-sm-directories	Обеспечение межсервисного взаимодействия сервиса zif-sm-directories в рамках realm ziiot
92	zif-sm-operationdefinition	Клиентская УЗ сервиса zif-sm-operationdefinition	Обеспечение межсервисного взаимодействия сервиса zif-sm-operationdefinition в рамках realm ziiot
93	zif-sm-operationperformance	Клиентская УЗ сервиса zif-sm-operationperformance	Обеспечение межсервисного взаимодействия сервиса zif-sm-operationperformance в рамках realm ziiot
94	zif-sm-operationsschedule	Клиентская УЗ сервиса zif-sm-operationsschedule	Обеспечение межсервисного взаимодействия сервиса zif-sm-operationsschedule в рамках realm ziiot
95	zif-sm-process	Клиентская УЗ сервиса zif-sm-process	Обеспечение межсервисного взаимодействия сервиса zif-sm-process в рамках realm ziiot
96	zif-sm-testspecification	Клиентская УЗ сервиса zif-sm-testspecification	Обеспечение межсервисного взаимодействия сервиса zif-sm-testspecification в рамках realm ziiot
97	zif-sm-workcalendar	Клиентская УЗ сервиса zif-sm-workcalendar	Обеспечение межсервисного взаимодействия сервиса zif-sm-workcalendar в рамках realm ziiot
98	zif-sm-workdefinition	Клиентская УЗ сервиса zif-sm-workdefinition	Обеспечение межсервисного взаимодействия сервиса zif-sm-workdefinition в рамках realm ziiot
99	zif-sm-workperformance	Клиентская УЗ сервиса zif-sm-workperformance	Обеспечение межсервисного взаимодействия сервиса zif-sm-workperformance в рамках realm ziiot
100	zif-sm-workschedule	Клиентская УЗ сервиса zif-sm-workschedule	Обеспечение межсервисного взаимодействия сервиса zif-sm-workschedule в рамках realm ziiot
101	zif-udl-dsentitywebapi	Клиентская УЗ сервиса zif-udl-dsentitywebapi	Обеспечение межсервисного взаимодействия сервиса zif-udl-dsentitywebapi в рамках realm ziiot
102	zif-udl-omconfigloader	Клиентская УЗ сервиса zif-udl-omconfigloader	Обеспечение межсервисного взаимодействия сервиса zif-udl-omconfigloader в рамках realm ziiot
103	zif-udl-omconfigupdater	Клиентская УЗ сервиса zif-udl-omconfigupdater	Обеспечение межсервисного взаимодействия сервиса zif-udl-omconfigupdater в рамках realm ziiot

104	zif-workflow	Клиентская УЗ сервиса zif-workflow	Обеспечение межсервисного взаимодействия сервиса zif-workflow в рамках realm ziiot
145	zui-app-acs	Клиентская УЗ сервиса zui-app-acs	Обеспечение межсервисного взаимодействия сервиса zui-app-acs в рамках realm ziiot
146	zui-app-calculation	Клиентская УЗ сервиса zui-app-calculation	Обеспечение межсервисного взаимодействия сервиса zui-app-calculation в рамках realm ziiot
147	zui-app-calculation-service	Клиентская УЗ сервиса zui-app-calculation-service	Обеспечение межсервисного взаимодействия сервиса zui-app-calculation-service в рамках realm ziiot
150	zui-app-dashboard-dx	Клиентская УЗ сервиса zui-app-dashboard-dx	Обеспечение межсервисного взаимодействия сервиса zui-app-dashboard-dx в рамках realm ziiot
151	zui-app-datainput	Клиентская УЗ сервиса zui-app-datainput	Обеспечение межсервисного взаимодействия сервиса zui-app-datainput в рамках realm ziiot
152	zui-app-documentexplorer	Клиентская УЗ сервиса zui-app-documentexplorer	Обеспечение межсервисного взаимодействия сервиса zui-app-documentexplorer в рамках realm ziiot
154	zui-app-events-registry	Клиентская УЗ сервиса zui-app-events-registry	Обеспечение межсервисного взаимодействия сервиса zui-app-events-registry в рамках realm ziiot
155	zui-app-event-types	Клиентская УЗ сервиса zui-app-event-types	Обеспечение межсервисного взаимодействия сервиса zui-app-event-types в рамках realm ziiot
158	zui-app-interface-manager	Клиентская УЗ сервиса zui-app-interface-manager	Обеспечение межсервисного взаимодействия сервиса zui-app-interface-manager в рамках realm ziiot
161	zui-app-mnemoeditor	Клиентская УЗ сервиса zui-app-mnemoeditor	Обеспечение межсервисного взаимодействия сервиса zui-app-mnemoeditor в рамках realm ziiot
162	zui-app-nds-configurator	Клиентская УЗ сервиса zui-app-nds-configurator	Обеспечение межсервисного взаимодействия сервиса zui-app-nds-configurator в рамках realm ziiot
163	zui-app-nifi	Клиентская УЗ сервиса zui-app-nifi	Обеспечение межсервисного взаимодействия сервиса zui-app-nifi в рамках realm ziiot
165	zui-app-om	Клиентская УЗ сервиса zui-app-om	Обеспечение межсервисного взаимодействия сервиса zui-app-om в рамках realm ziiot
173	zui-app-quality-service	Клиентская УЗ сервиса zui-app-quality-service	Обеспечение межсервисного взаимодействия сервиса zui-app-quality-service в рамках realm ziiot
174	zui-app-rdm-common	Клиентская УЗ сервиса zui-app-rdm-common	Обеспечение межсервисного взаимодействия сервиса zui-app-rdm-common в рамках realm ziiot
175	zui-app-reporteditor	Клиентская УЗ сервиса zui-app-reporteditor	Обеспечение межсервисного взаимодействия сервиса zui-app-reporteditor в рамках realm ziiot

176	zui-app-rtdb	Клиентская УЗ сервиса zui-app-rtdb	Обеспечение межсервисного взаимодействия сервиса zui-app-rtdb в рамках realm ziiot
179	zui-app-security-settings	Клиентская УЗ сервиса zui-app-security-settings	Обеспечение межсервисного взаимодействия сервиса zui-app-security-settings в рамках realm ziiot
180	zui-app-shell	Клиентская УЗ сервиса zui-app-shell	Обеспечение межсервисного взаимодействия сервиса zui-app-shell в рамках realm ziiot
182	zui-app-testspecification	Клиентская УЗ сервиса zui-app-testspecification	Обеспечение межсервисного взаимодействия сервиса zui-app-testspecification в рамках realm ziiot
183	zui-app-workflow	Клиентская УЗ сервиса zui-app-workflow	Обеспечение межсервисного взаимодействия сервиса zui-app-workflow в рамках realm ziiot
186	zif-universal-datamart	Клиентская УЗ сервиса zif-universal-datamart	Обеспечение межсервисного взаимодействия сервиса zif-universal-datamart в рамках realm ziiot
187	zui-app-universal-datamart	Клиентская УЗ сервиса zui-app-universal-datamart	Обеспечение межсервисного взаимодействия сервиса zui-app-universal-datamart в рамках realm ziiot

Таблица 0.2 Служебные УЗ пользователей БД PostgreSQL

№	Учётная запись	Описание (назначение / требования)	Полномочия (чтение \ запись)
1	stolon	Интерактивная УЗ суперпользователя доступная по умолчанию и имеющая все права на любые объекты/отношения в СУБД (в.	Полный доступ к любым объектам/отношениям на сервере Postgresql
2	admin	Интерактивная УЗ суперпользователя, которая используется для автоматизированного создания УЗ прочих сервисов инструментом zifctl на этапе развертывания платформы.	Полный доступ к любым объектам/отношениям на сервере Postgresql
3	repluser	Встроенная УЗ для поддердки репликации	
4	ziiot__debezium	УЗ инфраструктурного сервиса debezium **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД debezium
5	ziiot__keycloak	УЗ инфраструктурного сервиса keycloak **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД keycloak
6	ziiot__zif-attributes-directory	УЗ сервиса zif-attributes-directory **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-attributes-directory
7	ziiot__zif-cm-metadata	УЗ сервиса zif-cm-metadata **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-cm-metadata
8	ziiot__zif-dashboard	УЗ сервиса zif-dashboard **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-dashboard

9	ziiot__zif-datalink	УЗ сервиса zif-datalink **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-datalink
10	ziiot__zif-document-storage	УЗ сервиса zif-document-storage **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-document-storage
11	ziiot__zif-events	УЗ сервиса zif-events **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-events
12	ziiot__zif-events-integration	УЗ сервиса zif-events-integration **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-events-integration
13	ziiot__zif-interface-manager	УЗ сервиса zif-interface-manager **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-interface-manager
14	ziiot__zif-licensing	УЗ сервиса zif-licensing **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-licensing
15	ziiot__zif-material-lot	УЗ сервиса zif-material-lot **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-material-lot
16	ziiot__zif-mnemoschemes-storage	УЗ сервиса zif-mnemoschemes-storage **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-mnemoschemes-storage
17	ziiot__zif-notifications	УЗ сервиса zif-notifications **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-notifications
18	ziiot__zif-om-datareferences	УЗ сервиса zif-om-datareferences **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-om-datareferences
19	ziiot__zif-om-hashtags	УЗ сервиса zif-om-hashtags **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-om-hashtags
20	ziiot__zif-om-object	УЗ сервиса zif-om-object **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-om-object
21	ziiot__zif-om-objectmodel2excel	УЗ сервиса zif-om-objectmodel2excel **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-om-objectmodel2excel
22	ziiot__zif-om-process	УЗ сервиса zif-om-process **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-om-process
23	ziiot__zif-om-properties-view	УЗ сервиса zif-om-properties-view **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-om-properties-view
24	ziiot__zif-om-relationship	УЗ сервиса zif-om-relationship **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-om-relationship
25	ziiot__zif-om-sqldatasource	УЗ сервиса zif-om-sqldatasource **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-om-sqldatasource
26	ziiot__zif-om-testspecification	УЗ сервиса zif-om-testspecification **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-om-testspecification

27	ziiot__zif-om-uom	УЗ сервиса zif-om-uom **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-om-uom
28	ziiot__zif-om-valuetypes	УЗ сервиса zif-om-valuetypes **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-om-valuetypes
29	ziiot__zif-portal-settings	УЗ сервиса zif-portal-settings **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-portal-settings
30	ziiot__zif-propertyset	УЗ сервиса zif-propertyset **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-propertyset
31	ziiot__zif-quality-service	УЗ сервиса zif-quality-service **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-quality-service
32	ziiot__zif-rdm-common	УЗ сервиса zif-rdm-common **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-rdm-common
33	ziiot__zif-reporting	УЗ сервиса zif-reporting **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-reporting
34	ziiot__zif-rtdb-metadata	УЗ сервиса zif-rtdb-metadata **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-rtdb-metadata
35	ziiot__zif-security	УЗ сервиса zif-security **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-security
36	ziiot__zif-sm-directories	УЗ сервиса zif-sm-directories **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-sm-directories
37	ziiot__zif-sm-operationdefinition	УЗ сервиса zif-sm-operationdefinition **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-sm-operationdefinition
38	ziiot__zif-sm-operationperformance	УЗ сервиса zif-sm-operationperformance **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-sm-operationperformance
39	ziiot__zif-sm-operationsschedule	УЗ сервиса zif-sm-operationsschedule **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-sm-operationsschedule
40	ziiot__zif-sm-process	УЗ сервиса zif-sm-process **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-sm-process
41	ziiot__zif-sm-testspecification	УЗ сервиса zif-sm-testspecification **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-sm-testspecification
42	ziiot__zif-sm-workcalendar	УЗ сервиса zif-sm-workcalendar **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-sm-workcalendar
43	ziiot__zif-sm-workdefinition	УЗ сервиса zif-sm-workdefinition **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-sm-workdefinition
44	ziiot__zif-sm-workperformance	УЗ сервиса zif-sm-workperformance **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-sm-workperformance

45	ziiot__zif-sm-workschedule	УЗ сервиса zif-sm-workschedule **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-sm-workschedule
46	ziiot__zif-udl-dsentitywebapi	УЗ сервиса zif-udl-dsentitywebapi **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-udl-dsentitywebapi
47	ziiot__zif-workflow	УЗ сервиса zif-workflow **Платформы ZIIoT**	Полный доступ к любым объектам/отношениям БД zif-workflow

Таблица 0.3 Служебные УЗ БДВР Cassandra

№	Учётная запись	Описание (назначение / требования)	Полномочия (чтение \ запись)
1	cassandra	Интерактивная УЗ суперпользователя, доступная по умолчанию и имеющая ВСЕ права на любые keyspace и объекты keyspace	Полный доступ к любым keyspace
2	admin	Интерактивная УЗ суперпользователя, по правам приравненная к cassandra, которая используется для запуска сервисов создающих keyspace для прочих сервисов	Полный доступ к любым keyspace
3	ziiot__zif-om	УЗ сервиса zif-om **Платформы ZIIoT**	Полный доступ к любым keyspace zif-om
4	ziiot__zif-rtdb	УЗ сервиса zif-rtdb **Платформы ZIIoT**	Полный доступ к любым keyspace zif-rtdb
5	ziiot__zif-sm	УЗ сервиса zif-sm **Платформы ZIIoT**	Полный доступ к любым keyspace zif-sm

Таблица 0.4 Прочие Служебные УЗ

№	Учётная запись	Описание (назначение / требования)	Полномочия (чтение \ запись)
1	RabbitMQ admin	УЗ администратора сервиса RabbitMQ	Доступ к данным RabbitMQ
2	Redis admin	УЗ администратора сервиса Redis	Доступ к данным Redis
3	RedisInfra admin	УЗ администратора дополнительного сервиса Redis (если сервис устанавливается)	Доступ к данным RedisInfra
4	S3 Strorage AccessKey admin	rootAccessKey для сервиса Minio	Доступ к данным Minio
5	Pgadmin user	Пользователь UI Pgadmin	Доступ к UI Pgadmin