OZON tech

Управление УЗ в PostgreSQL, путь Ozon

Васильев Дмитрий,
Эксперт разработки информационных систем, группа PostgreSQL DBA

Дмитрий Васильев

* В IT около 20 лет, занимался разработкой, инфраструктурой и базами
* Люблю котиков

1

С чего все начиналось

Проблемы которые мы будем решать:

* Много микросервисов и у каждого своя база (несколько тысяч баз)
* Разработчики знали пароль базы данных и пользовались им для решения инцидентов.
* Некоторые сервисы использовали одну базу данных с одной УЗ (!!!).
* Поменять пароль в случае утечки было тяжело.
* Блокирование УЗ, которые являлись owner'ами объектов в базе — часто приводит к недопониманию и ошибкам, как на стороне заказчиков так и администраторов.

Как создавались УЗ

DBA в оттянутых свитерах делали по хардкору, почти руками:

* Это было долго для разработчиков (заказчиков).
* Это было неудобно для администраторов (генерировали руками секреты и запускали SCM).
* Часто происходили ошибки (человеческий фактор).

2

Как в итоге можно было сделать

Упростить доступ личных УЗ в базу

OZON tech

В первую очередь хотели взять под контроль личные УЗ:

* Заложить ролевое управление в базу.
* Синхронизировать группы/пользователей из AD (pg-ldap-sync?).
* Ввести в домен (?!).

Почему мы так не сделали

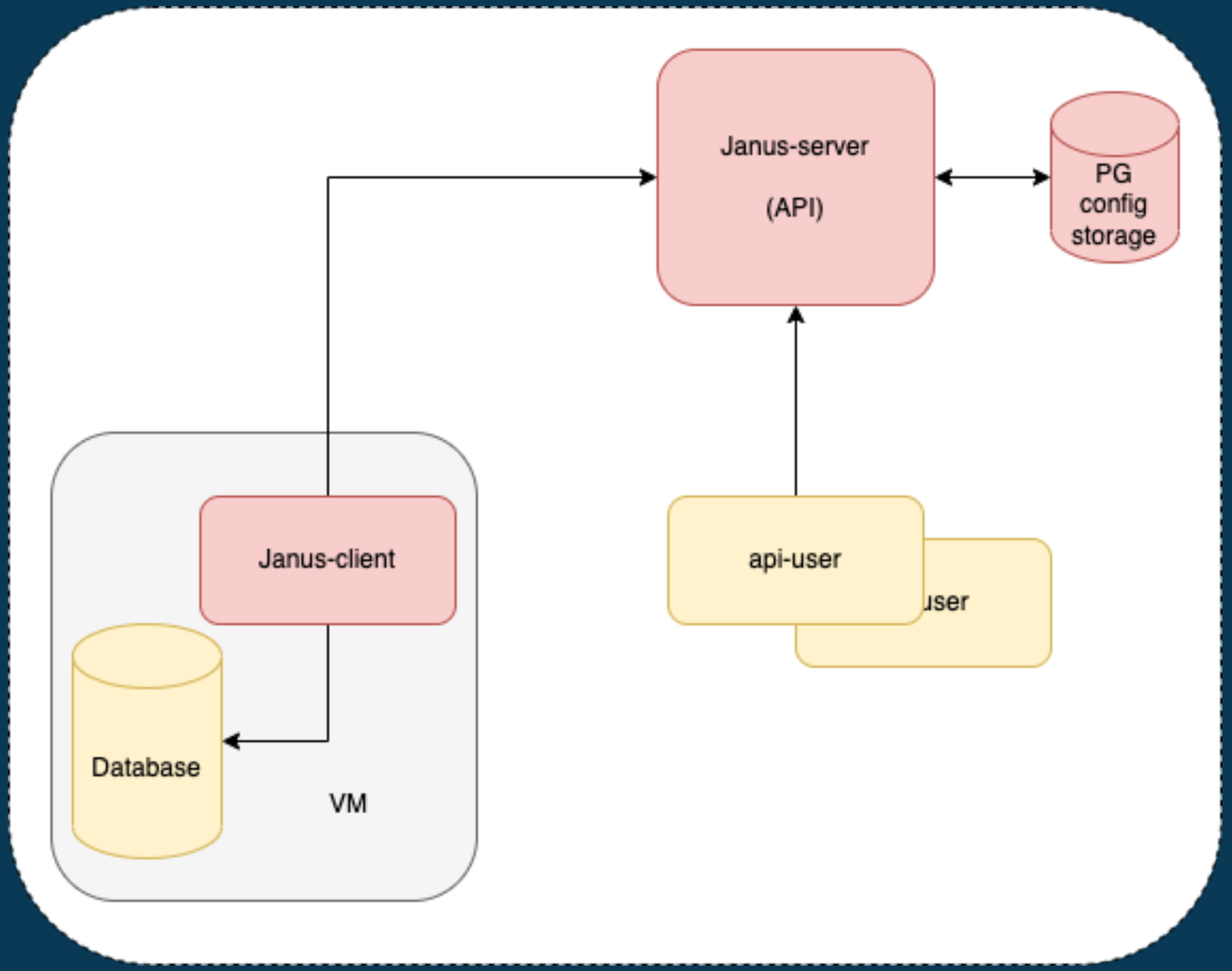
OZON tech

* В первую очередь, мы DBA, не хотели менять настройки сервера PostgreSQL.
* Лишняя зависимость доступа сервисных УЗ от сторонних сервисов (связность систем).

3

Как у нас получилось

Janus-server и janus-client



Janus-server

Хранилище конфигураций пользователей для всех баз.

* Хранит конфигурацию доступа всех баз в своей db.
* Может сгенерировать конфигурацию для конкретной db, которую обслуживает janus-client.
* Изменение конфигурации — обновление счетчика configuration-id.
* Может выписать «токен» — пароль, подписанный приватным ключом, который используется клиентами для аутентификации.

* ~~Много микросервисов и у каждого своя база~~

Janus-client

OZON tech

* Создаёт и поддерживает роли для доступа
* Пулит изменения с janus-server и сообщает номер configuration-id
* Создание аккаунта полусинхронное:
  * Мы достаточно часто (раз в 3 секунды) пулим configuration-id с janus-server.
  * Если janus-client понимает, что его текущее configuration-id ниже, чем серверное — идём за новой конфигурацией и сообщаем janus-server о применении.
  * Таким образом при обращении в api janus-server ручка создания/обновления аккаунта является полусинхронной — дожидается применения на стороне janus-client.
  * Компенсации при таймауте нет, так как api в janus-server идемпотентное

4

Предопределённые роли доступа

Роли

В микросервисах одна база — одно приложение, поэтому можем себе позволить.

* read_only: может читать все таблицы.
* read_write: read_only + может писать и использовать функции.
* owner: read_write + ddl.

* ~~Блокирование УЗ которые являлись owner'ами объектов в базе - часто приводит к недопониманию и ошибкам как на стороне заказчиков так и администраторов.~~

Эти роли разделяют как сервисные УЗ, так и пользовательские

* Добавление прав — просто добавление в role.
* Если у аккаунта есть роль owner, то alter user <account> set role owner.

* ~~Блокирование УЗ которые являлись owner’ами объектов в базе - часто приводит к недопониманию и ошибкам как на стороне заказчиков так и администраторов.~~

Роли поддерживает в актуальном состоянии janus-client

Owner'ом таблицы становится тот, кто её создал. Бывают разные кейсы с миграциями и инцидентами, поэтому роли надо поддерживать в актуальном состоянии.

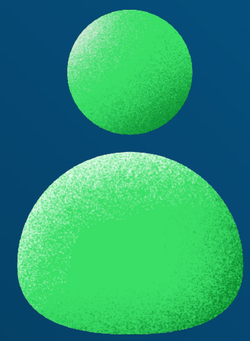
Alter default privileges — уже не работает.

pg_{read,write}_all_data — нет в старых версиях pg.

* Проверяем, не появилось ли новых объектов по oid и, если появились расхождения, исправляем.
* Эпизодический force-resync

5

Пользовательские УЗ



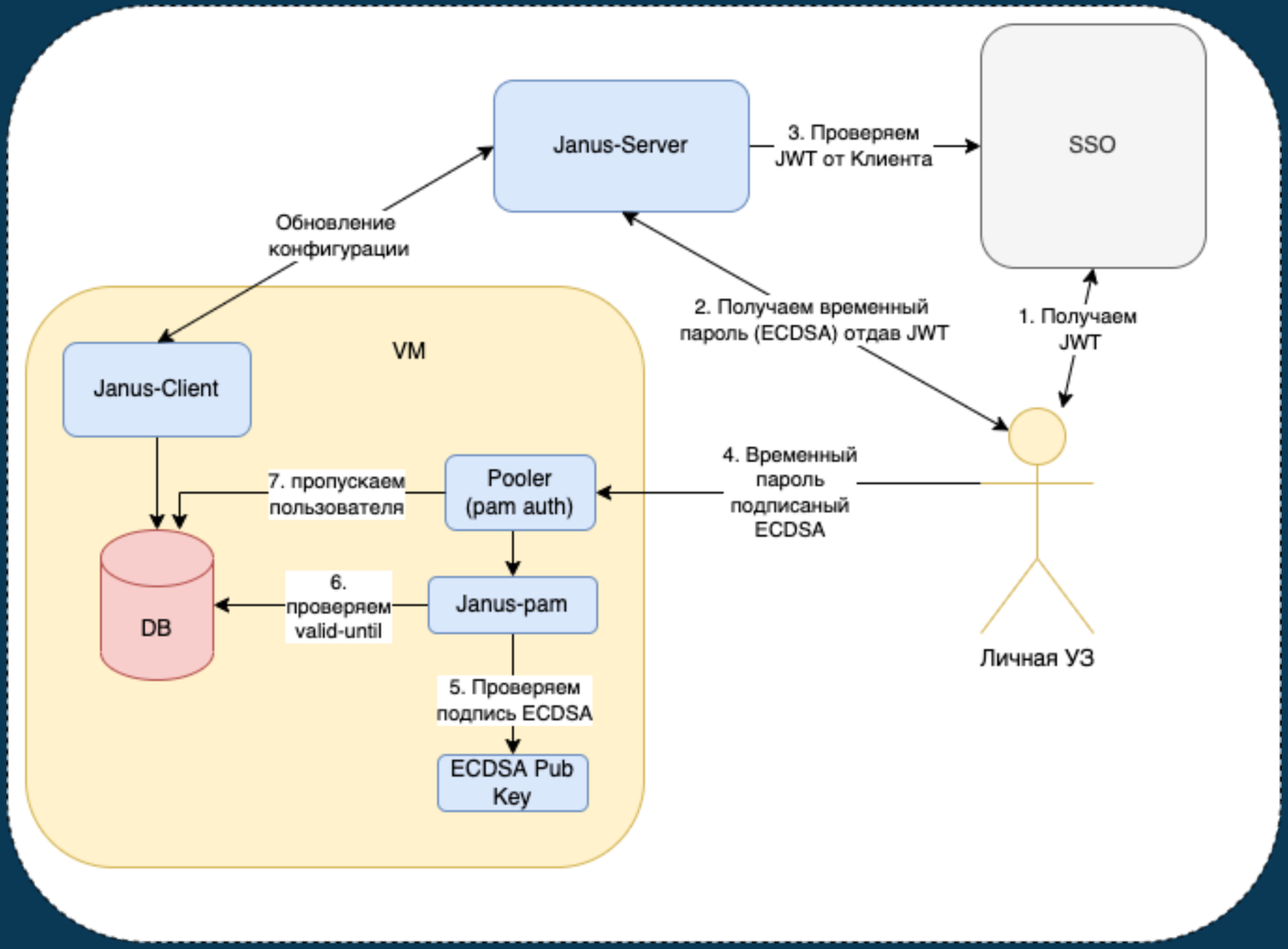
Продолжаем разделять сервисы и пользователей

OZON tech

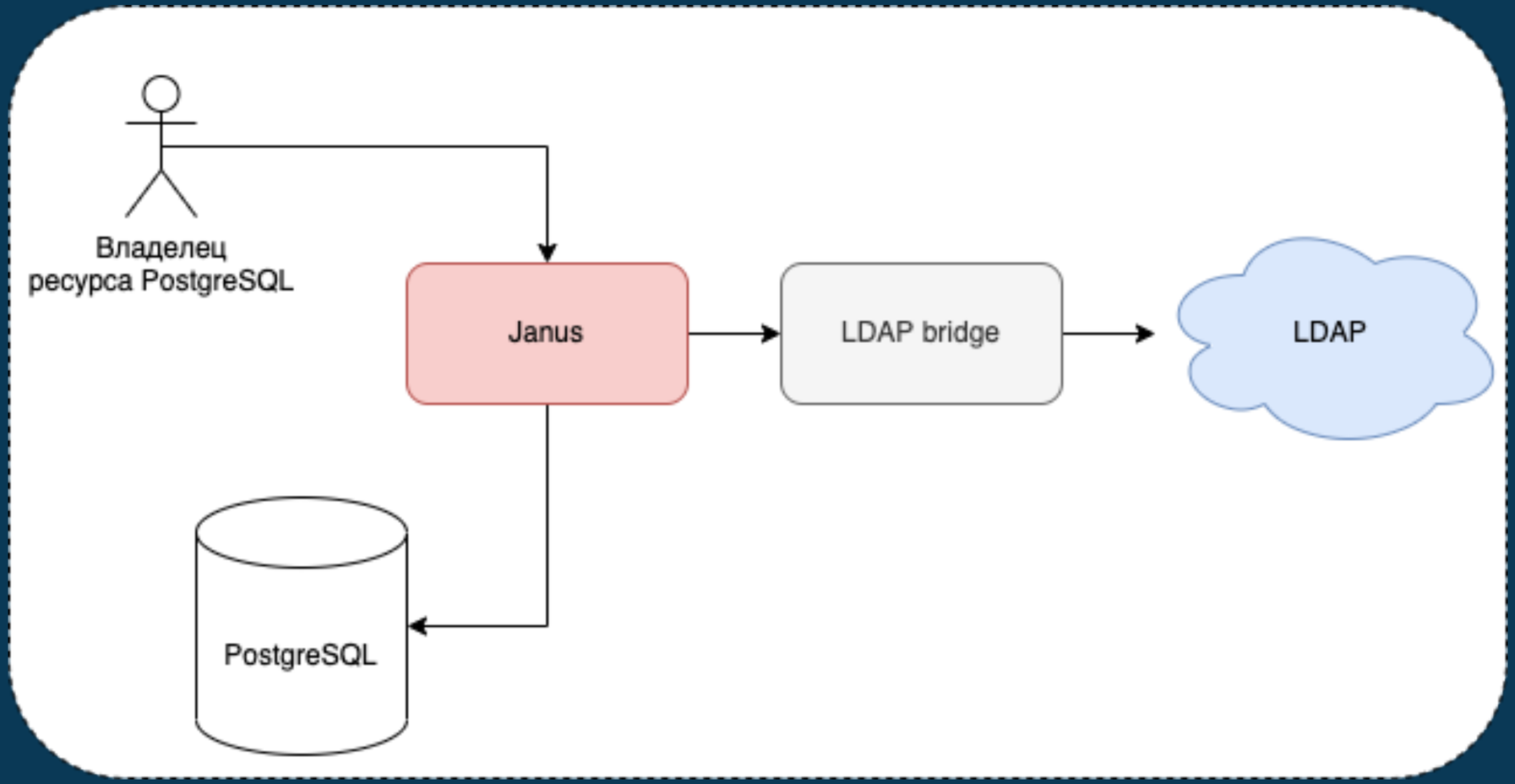
* db pooler, который используют сервисы и аутентификация через scram:
    * Войти сюда пользовательской УЗ нельзя (HBA).
* db pooler, который использует pam для аутентификации пользовательских УЗ:
    * Войти сюда сервисной УЗ нельзя (тип пароля).

* ~~Разработчики знали пароль базы данных и пользовались им для решения инцидентов.~~

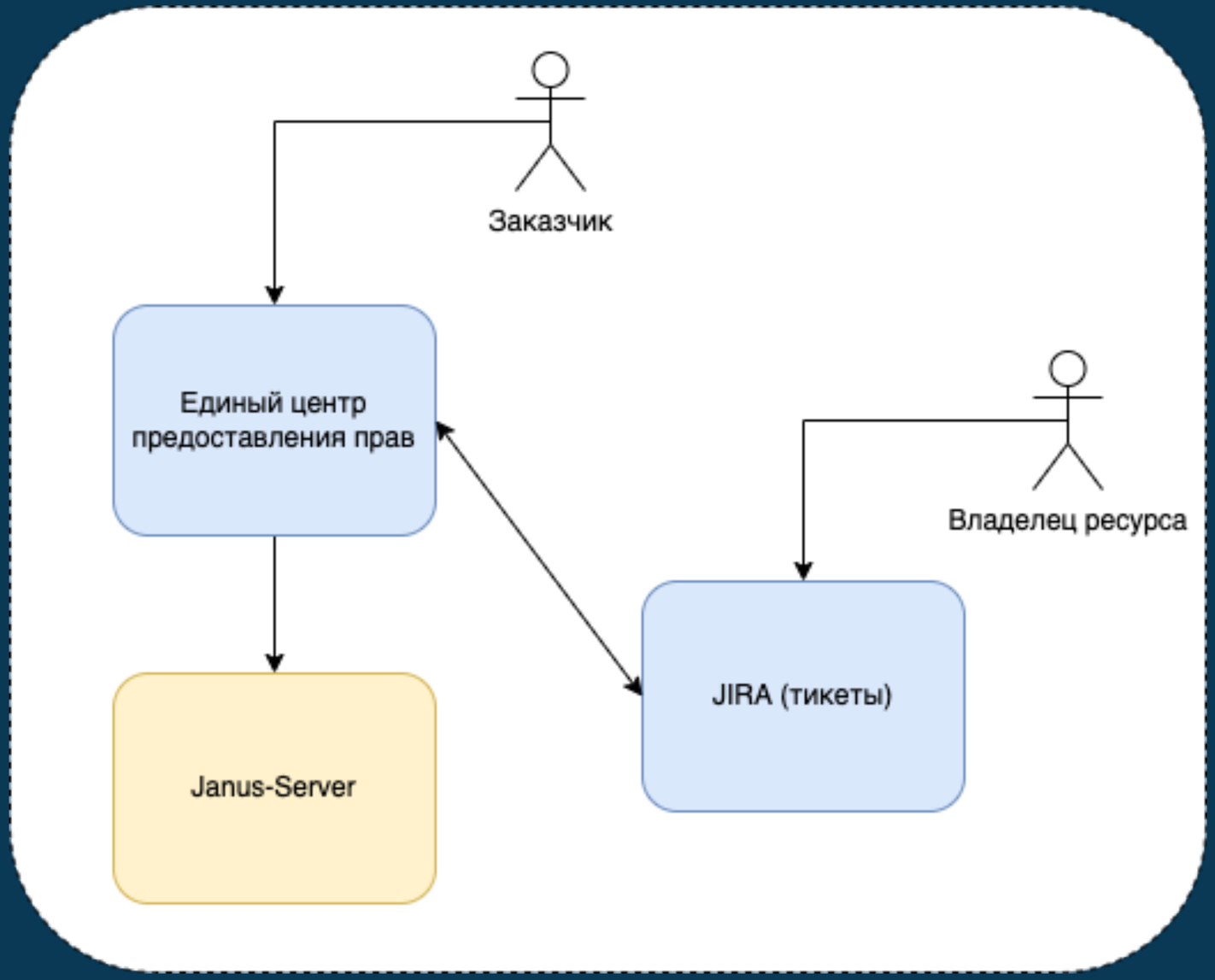
PAM — пользовательские УЗ и временные токены



Создание личный УЗ: первый этап



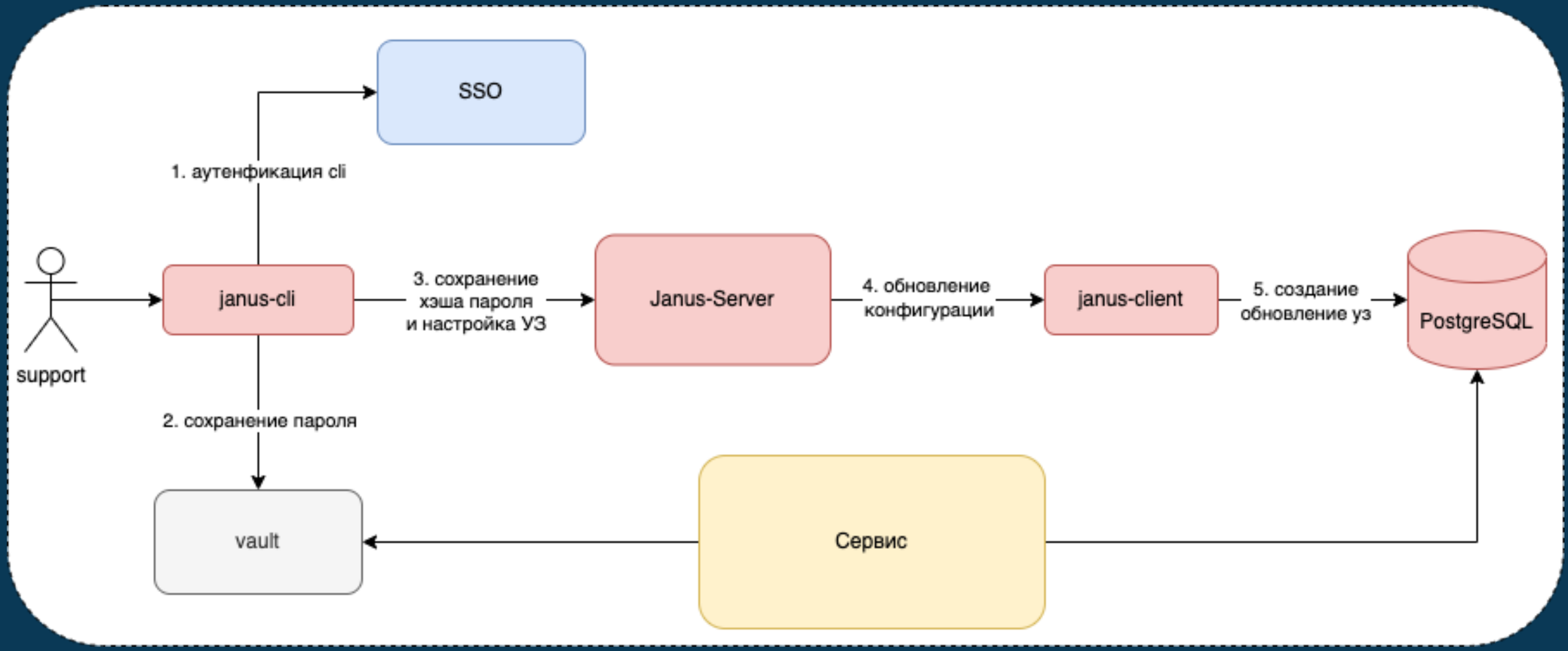
Текущее состояние: единый центр предоставления прав



5

Сервисные УЗ

Сервисные УЗ, текущее состояние



Смена пароля сервисной УЗ

* Создаем новую сервисную УЗ с правами owner.
* Блокируем доступ старой сервисной УЗ.

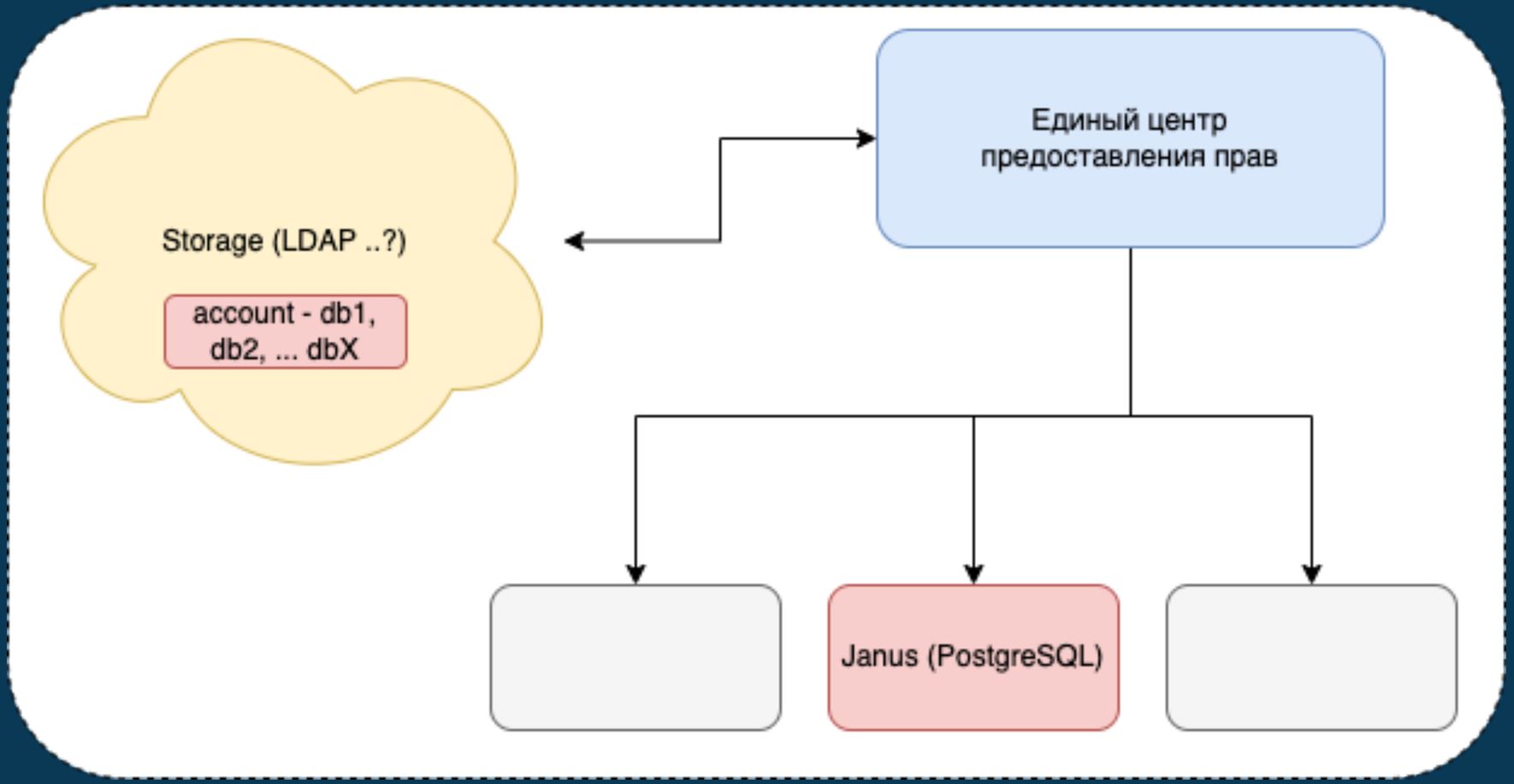
* ~~Поменять пароль в случае утечки было тяжело~~

Будущее (сервисные УЗ): нерешенные проблемы

OZON tech

* Некоторые сервисы использовали одну базу данных с одной УЗ.

Будущее (сервисные УЗ)



OZON tech

Спасибо за внимание

Васильев Дмитрий, Эксперт разработки информационных систем, группа PostgreSQL DBA

dmitrivasilyev@ozon.ru

Вопросы?