inCountry

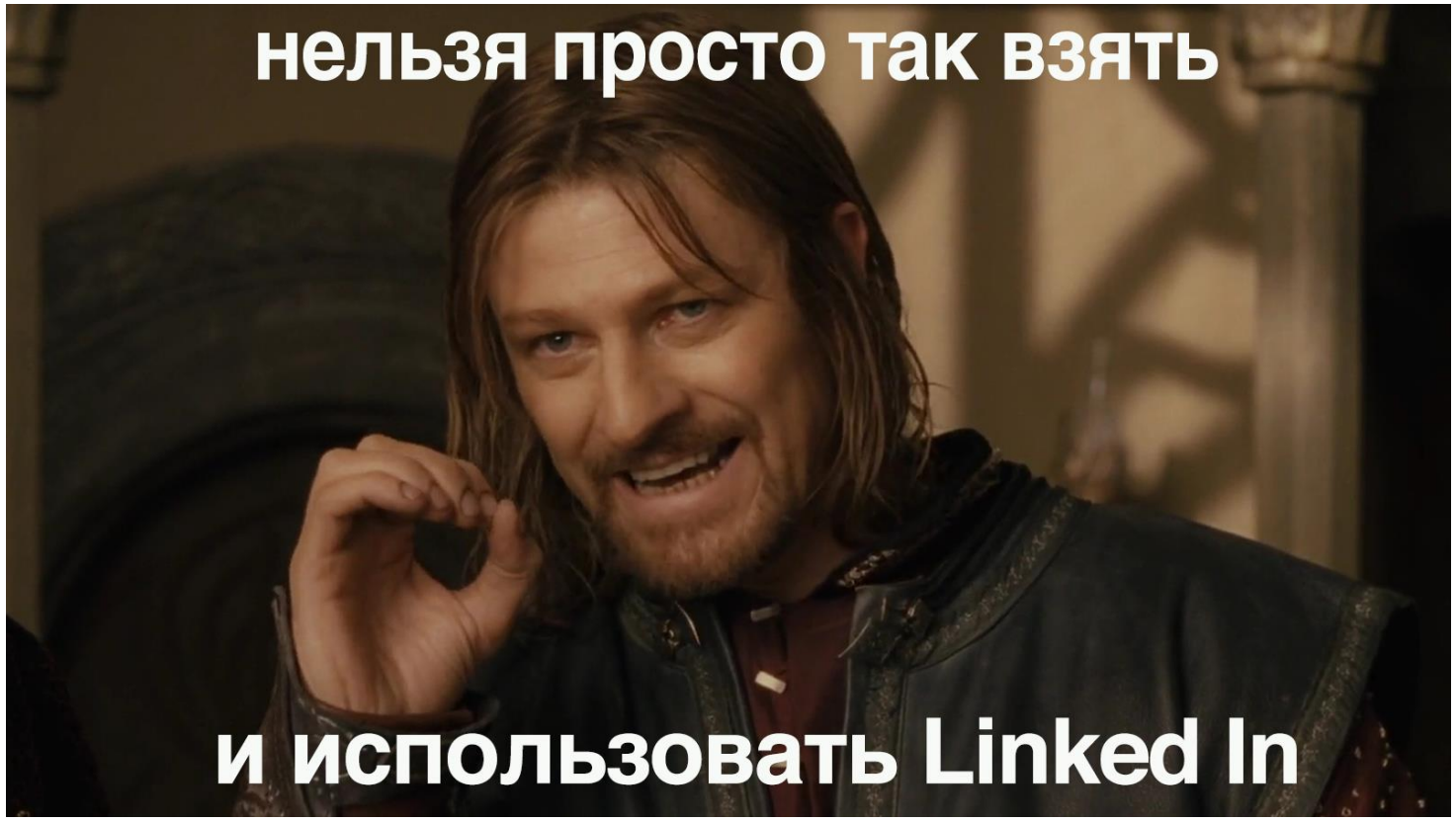
хранение регулируемых данных в 62 странах мира

Павел Конотопов - kakoka@gmail.com
Леонид Альбрехт - lalbrekht@gmail.com

DBA team

Идея

- Давайте создадим и предоставим глобальному бизнесу инфраструктуру для локального хранения регулируемых данных



- Инфраструктура должна соответствовать определенным требованиям

Данные

- Регулируемые законодательством данные

Персональные данные: возможность однозначно идентифицировать человека

«… любая информация, относящаяся к прямо или косвенно определенному или определяемому физическому лицу — субъекту персональных данных…»

- Обработка персональных данных

совокупность действий, совершаемых с персональными данными, включая сбор, запись, систематизацию, накопление, **хранение**, уточнение, изменение, извлечение, использование, передачу, обезличивание, блокирование, удаление, уничтожение

Требования

- Хранить данные в странах их происхождения;
- Геораспределенная инфраструктура;
- Но при этом данные не могут пересекать границ страны происхождения;
- Соответствие промышленным стандартам безопасности: SOC, PCI DSS, HIPAA, ISO…;
- Соответствие локальным законодательствам: ФЗ-152, GDPR,CCPA…
- Уметь интегрироваться в уже готовые системы
- Поддержка шифрования и
- Приемлемая скорость работы
- Дешево (мы стартап)

Продукты

- REST API

- не требует дополнительных усилий

- модифицирует данные внутри уже существующего сервиса

- SDK

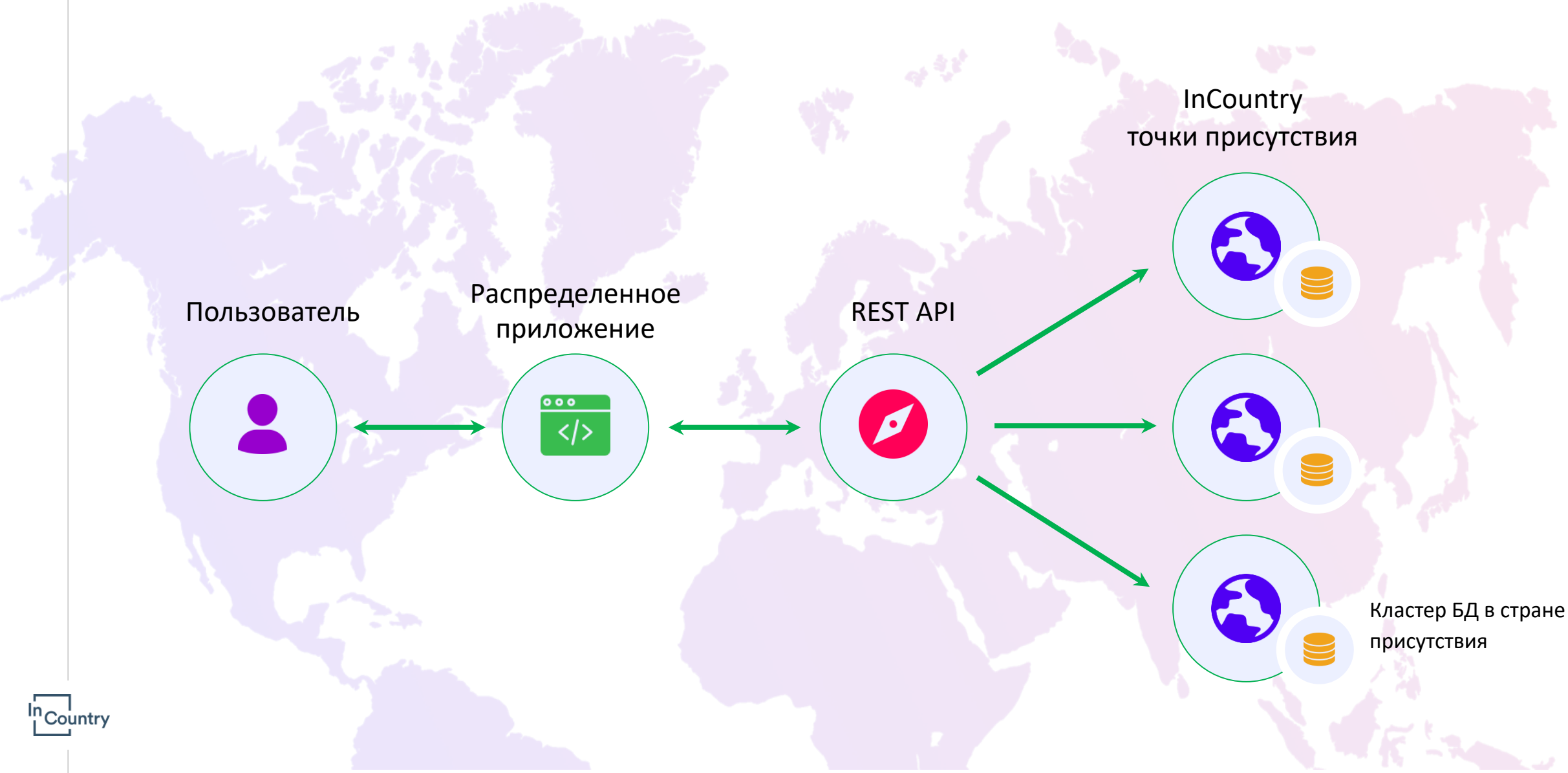
Java, PHP, Python, Node.js SDKs

Self-service

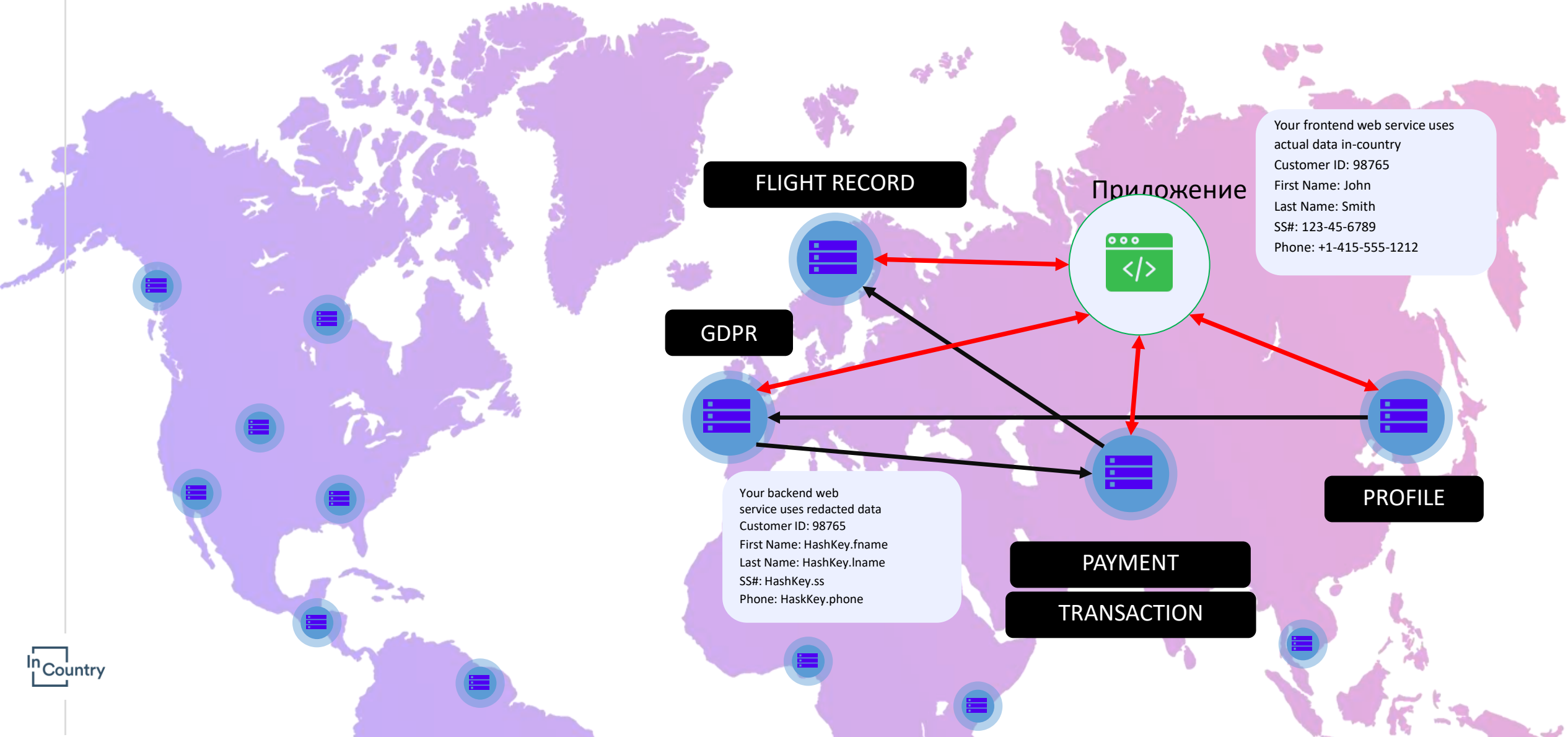
Client-side encryption

- Border proxy

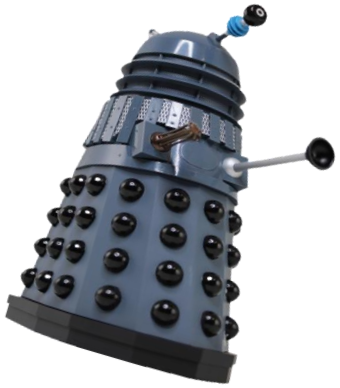
Архитектура сервиса



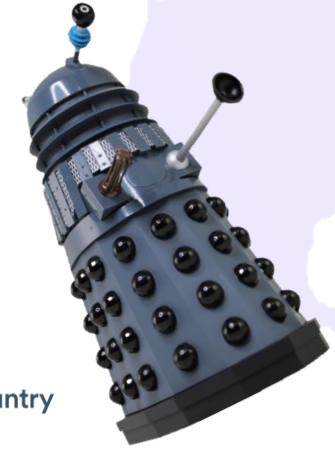
Точки присутствия



Инфраструктура

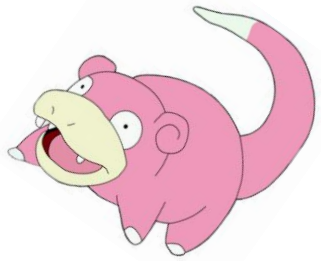


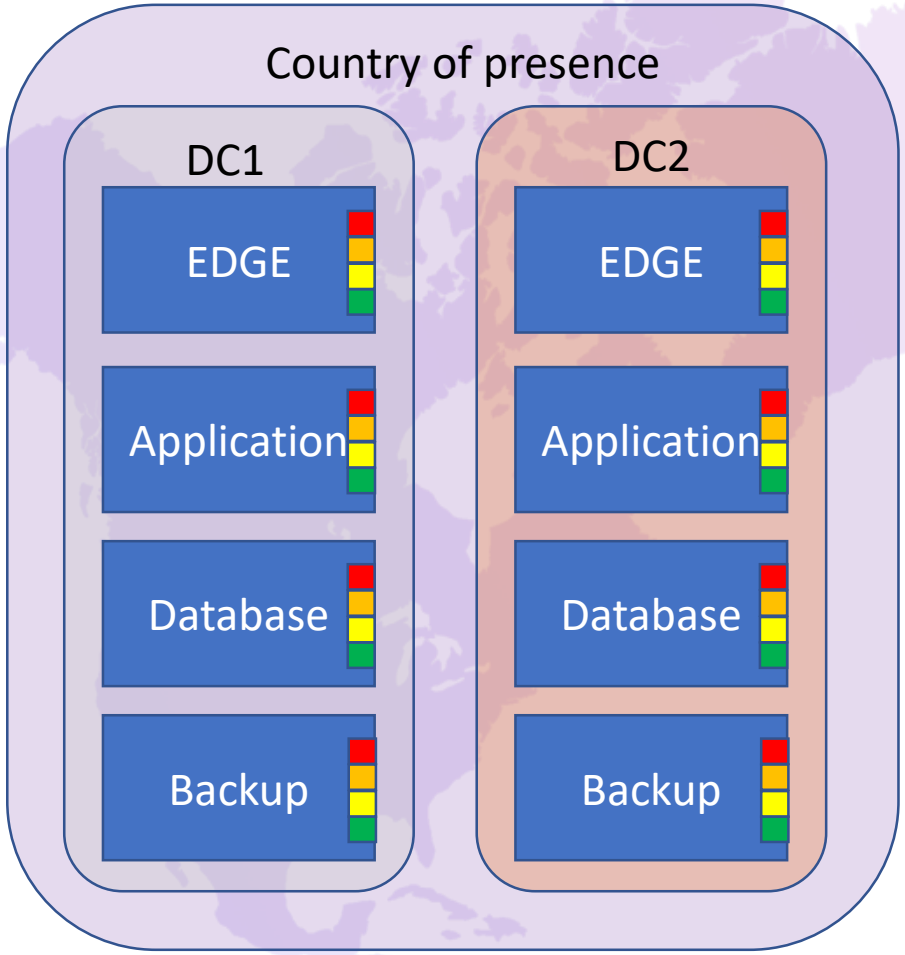
- Точки присутствия – 62 страны
  - 30% Cloud
  - 70% BARE-METAL
    - Сразу забыть про K8s
  - 2 дата центра в стране
  - От 4 до 8 сервера на страну

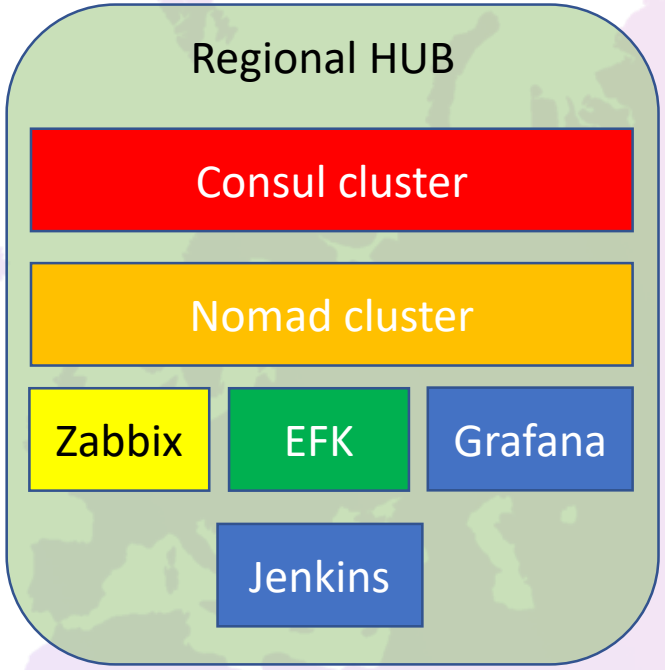


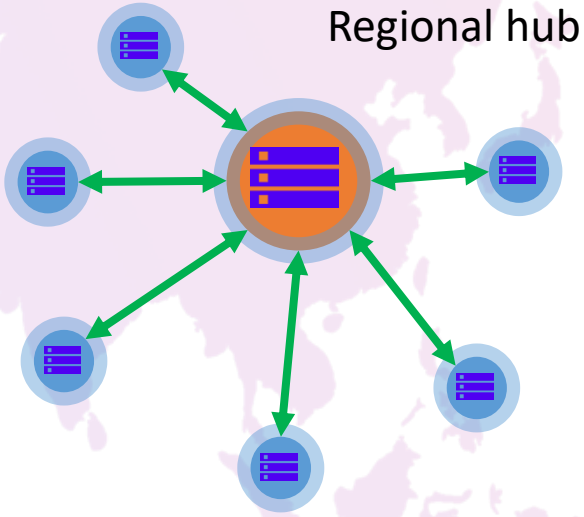
- Централизованная инфраструктура развертывания
- Централизованная инфраструктура управления
- Безопасность
- Мониторинг

Инфраструктура



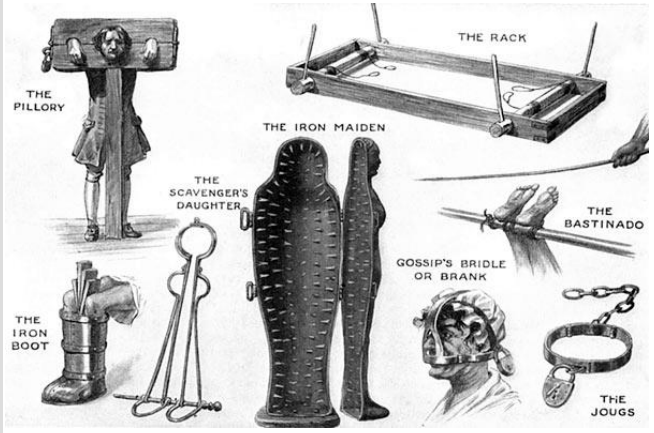






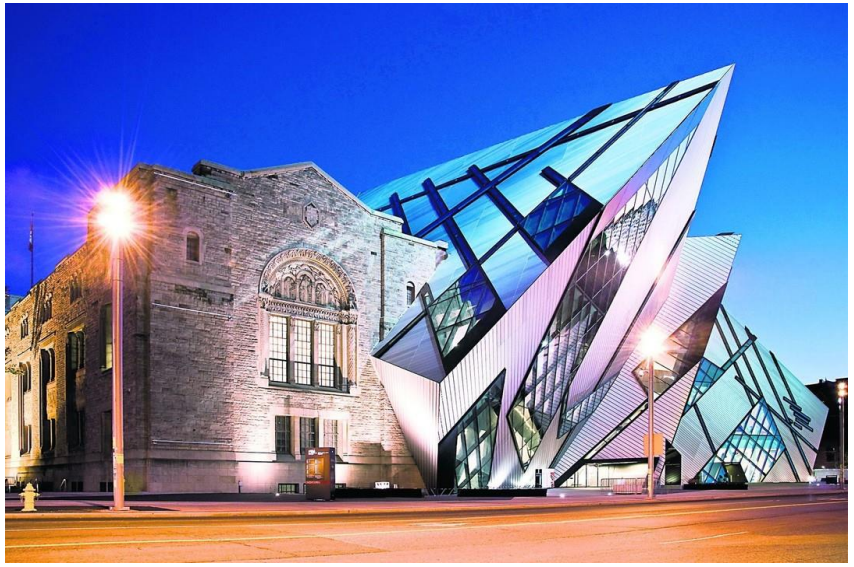
Дешево-сердито vs Дорого-богато vs Голос разума

- А давайте возьмем Open Source и сделаем Enterprise
- Но есть проблема



+

=



OpenSource + DevOps = T x $$$
Limited OpenSource + Devops = $ + T x $
Non-OpenSource + DevOps = $$$ + T x $

Технологический стек

OpenSource

- Terraform/Packer/Ansible/AWX

- Consul – discovery service
- Nomad – планировщик
- Vault – хранилище секретов
- Docker – контейнеризация
- Jenkins – CI/CD
- Nginx
- PostgreSQL/Pgbouncer/Patroni
- Zabbix/ElasticSearch/Fluentd/Grafana/Kibana

Не OpenSource

- JumpCloud – LDAP
- Cisco VPN
- OpsGenie – тревоги и предупреждение
- JFROG artifactory – хранение docker образов

Задачи для PostgreSQL

- Георассредоточенность
- 2 дата центра в одной стране
- Высокая доступность
- Синхронная реплика
- Автоматический failover
- Защита от второго мастера в кластере. Fencing

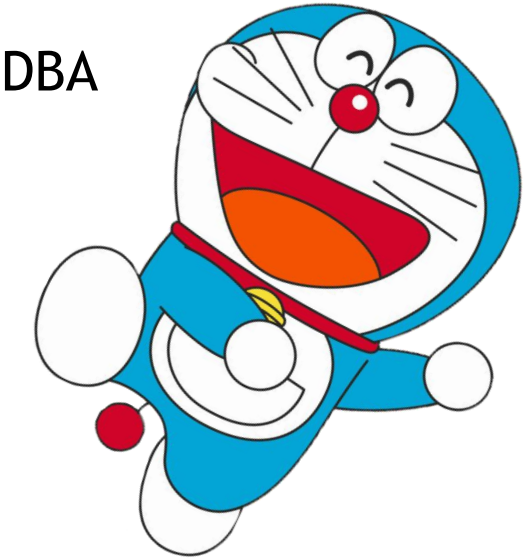
Шестеренки

- PostgreSQL
- pgbouncer - пулер соединений
- Patroni - как механизм управления кластером
- HashiCorp Consul
  - Хранит информацию о состоянии кластера
  - Хранит текущую конфигурацию кластера
  - Consul Templates
- HashiCorp Nomad
  - Автоматически поднимает контейнеры при их падении
- HashiCorp Vault - хранение секретов: статические и динамические роли

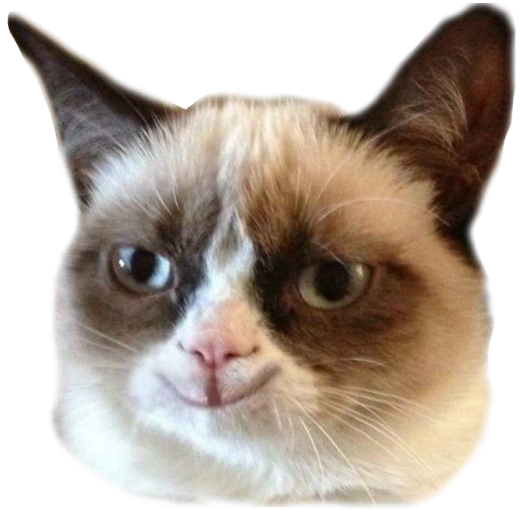
PostgreSQL: Деплой

PostgreSQL: Patroni

- PostgreSQL из коробки не имеет решения для автоматического фейловера
- Демон запущенный рядом с PostgreSQL
- Он взаимодействует с DCS
- Демон Patroni принимает решение о promotion/demotion
- Конфиг в виде YAML
- patronictl - удобный инструмент не только для DBA



PostgreSQL: Nomad



- Плюсы
  - Единое место для управления контейнерами
  - Декларативный подход
  - Удобное выделение ресурсов для контейнера

- Минусы
  - Нельзя так просто взять и остановить контейнер
  - В некоторых “особенных” случаях контейнер срубается в неожиданном месте

Nomad.TemplateFile

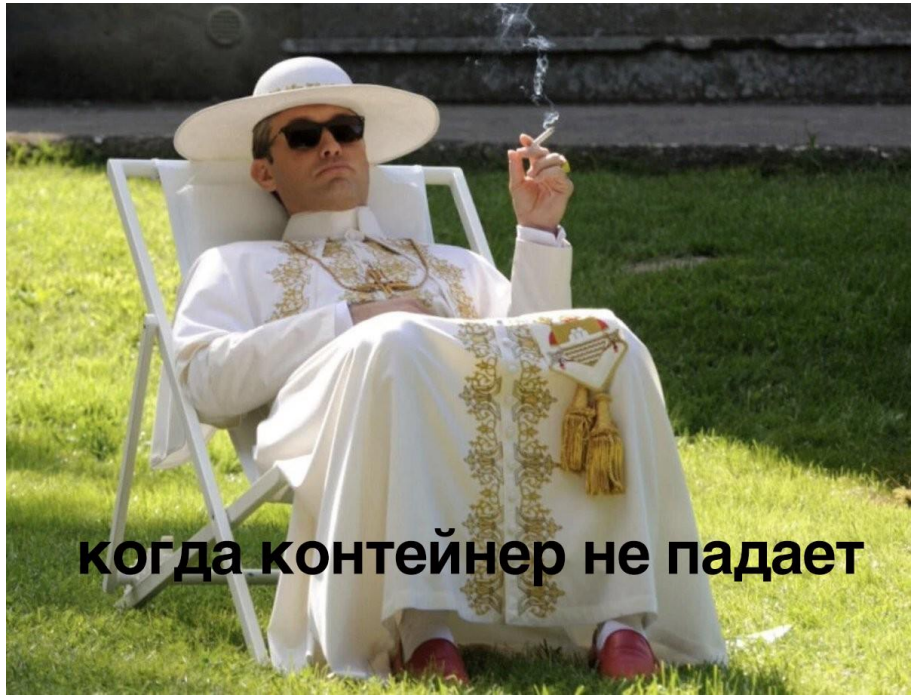
```
update {
    max_parallel = 1
    min_healthy_time = "10s"
    healthy_deadline = "3m"
    progress_deadline = "10m"
    auto_revert = false
    canary = 0
  }

…
resources {
        cpu    = ${NOMAD_RESOURCES_CPU}
        memory = ${NOMAD_RESOURCES_MEM}
        network {
          mbits  = 1
          port "${NOMAD_POSTGRES_SERVICE_NAME}" {
            static = “${NOMAD_POSTGRES_PORT}"
…
```

```
port_map {
          ${NOMAD_PATRONI_SERVICE_NAME} = “$
{NOMAD_PATRONI_PORT}"
        }
        port_map {
          ${NOMAD_POSTGRES_SERVICE_NAME} = “$
{NOMAD_POSTGRES_PORT}”
        }
mounts = [
          {
            type = "bind"
            source = "/etc/patroni/patroni.yml"
            target =
"/home/postgres/.config/patroni/patroni.yml"
            readonly = true
          },
```

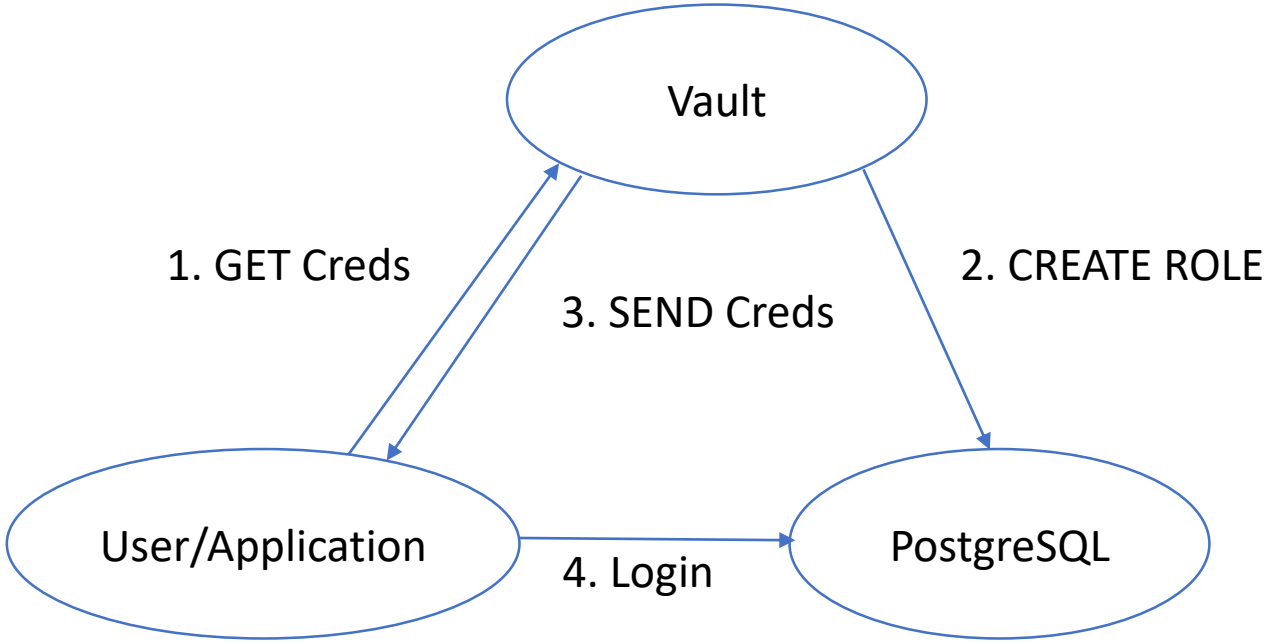
PostgreSQL: Docker

- Persistent Storage для файлов базы
- Облегчение образа с помощью мульти-стейдж сборки
  - ~70-90 мб образ
  - Как следствие ускорение доставки до региона
- Прекрасно живется, если не падает контейнер
- Практически никакого оверхеда



PostgreSQL: HashiCorp Vault

- Хранит статические секреты
- Позволяет обслуживать статические роли
  - Соблюдает политику обновления паролей
- Может создавать динамические роли
  - Такие роли создаются на заданное время при запросе к Vault
  - Обладают необходимыми правами
  - Удаляются после истечения срока жизни



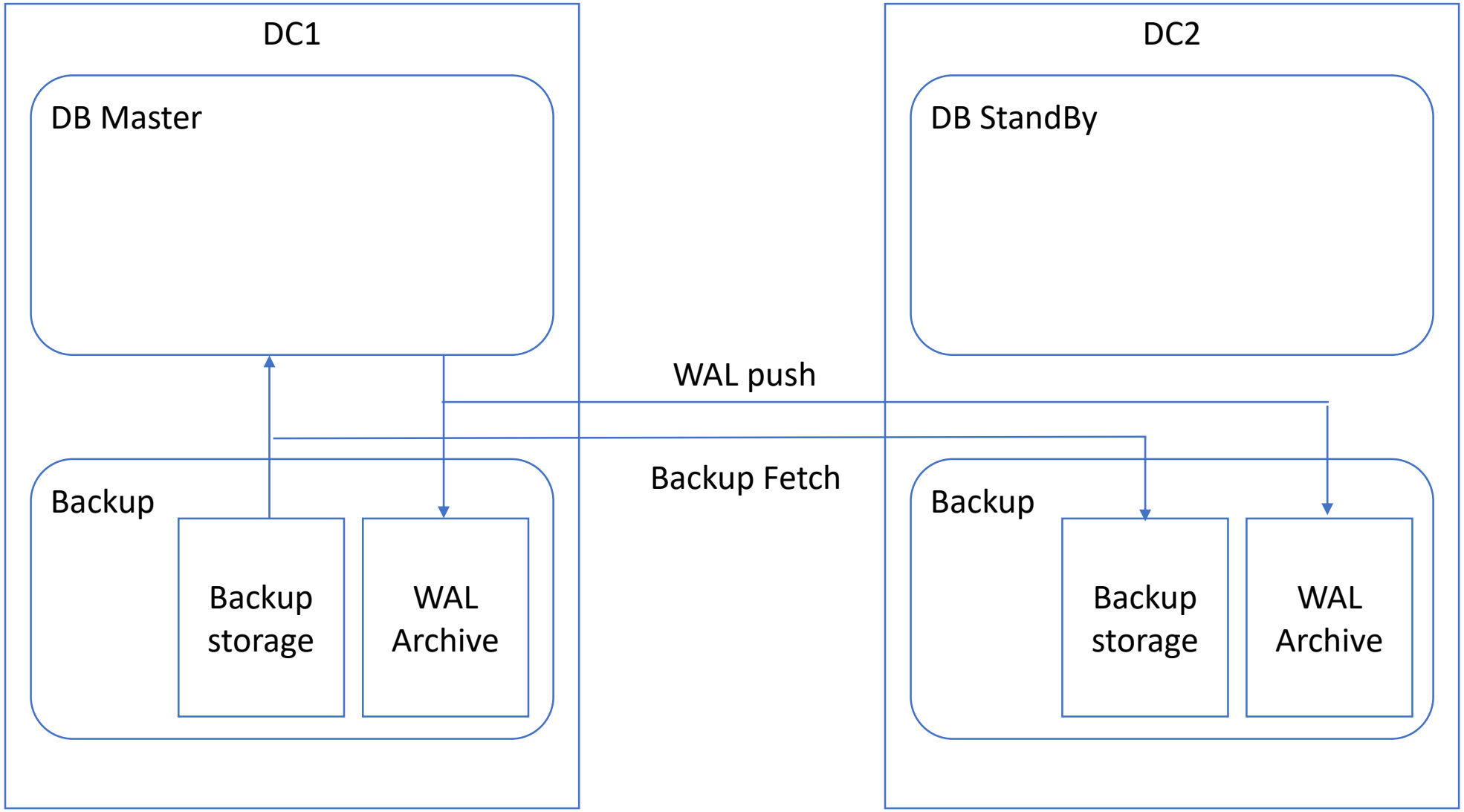
PostgreSQL: Репликация

- Слоты включены по умолчанию. Но не нужно забывать про wal_keep_segments
- Особенности синхронной репликации:
  - Узкое место - сеть
  - synchronous mode
- maximum_lag_on_failover: 1048576 (1M)

PostgreSQL: Резервное копирование

- pg_probackup
- Валидация
- Логирование и мониторинг
- Индивидуальные политики хранения для клиентов
- Особенности двух DC:
  - Двойной WAL архив
  - Двойной бекап архив

PostgreSQL: Резервное копирование



PostgreSQL: Резервное копирование

- По умолчанию реплика создается с помощью утилиты pg_basebackup
- Это поведение можно переопределить параметром ‘create_replica_methods’

```
postgresql:
  create_replica_methods:
    - probackup
    - basebackup
  probackup:
    command: "ssh dbbackup@192.168.0.250 ‘bash /var/backup/pg_restore.sh'"
    no_params: True
  basebackup:
    max-rate: '100M'
```

PostgreSQL: восстановление кластера

- Возможность восстановиться из бекапа на любую точку по:
  - Времени
  - Id транзакции (xid)
  - LSN транзакционной записи в журнале
- Редкое явление при синхронной реплике, но подготовиться надо:

```
bootstrap:
  method: probackup
  probackup:
    command: ssh dbbackup@backup 'bash /var/backup/pg_restore.sh'
    keep_existing_recovery_conf: false
  recovery_conf:
    recovery_target_timeline: latest
    restore_command: pg_probackup archive-get -B /var/backup --instance db-mt --remote-user=dbbackup --wal-file-path %p --
wal-file-name %f —remote-host=255.255.255.255
```

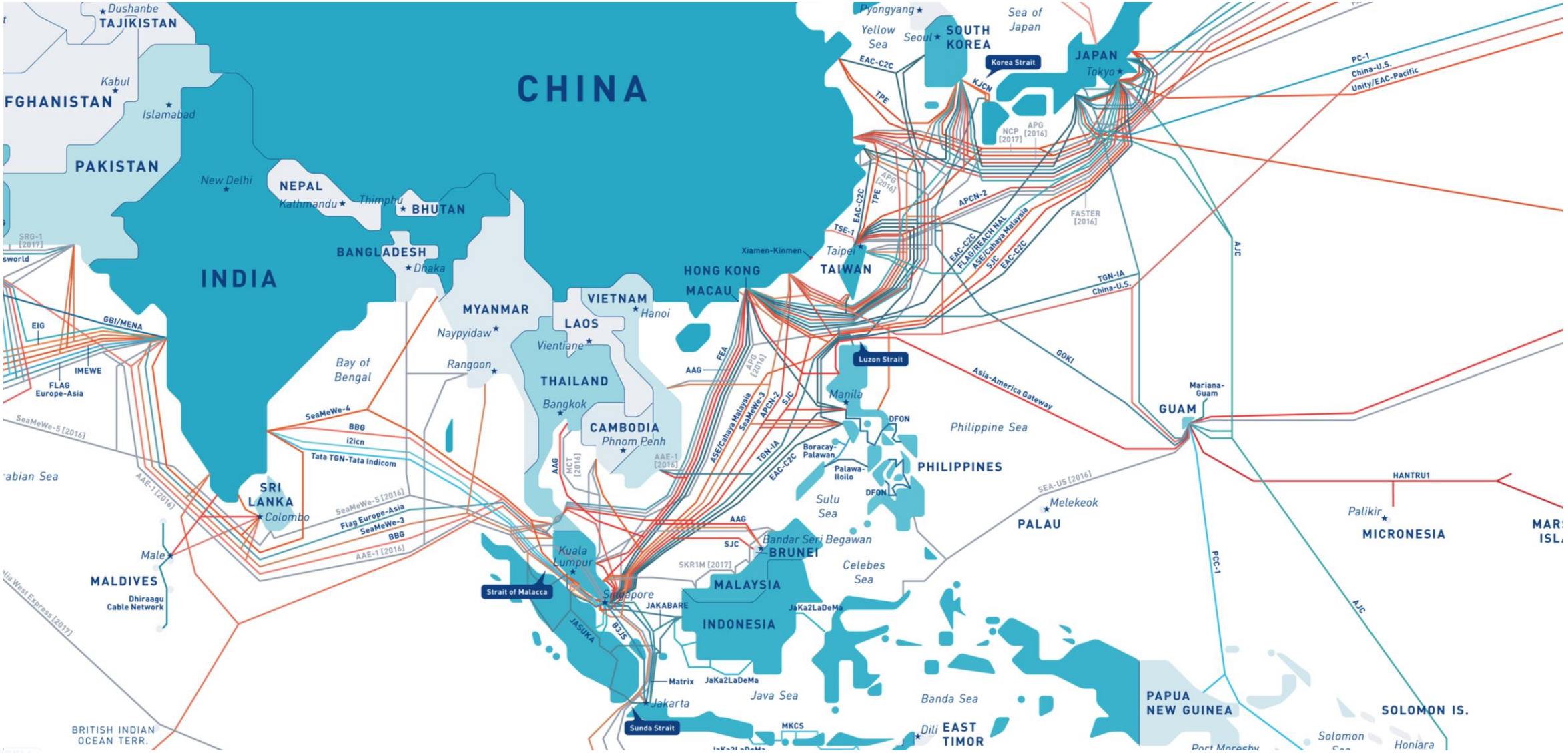
PostgreSQL: Валидация

- Docker образ для минимального запуска
- Скрипт восстановления из бекапа
- Минимальный postgresql.conf для старта
- Проверяем успешность запуска
- docker exec pgvalid pg_dump -h localhost -U postgres > /dev/null
- Служебная таблица для сравнения данных до и после бекапа
- amcheck
  - CREATE EXTENSION amcheck
  - pg_probackup checkdb —amcheck —heapallindexed
- Логируем и мониторим каждое действие и его (не)успешность

Troubleshooting

- Все что связано с Consul-ом:
  - Сетевые проблемы в одном из DC
  - Недоступен из обоих DC
- Проблемы с репликацией и HA. Частые переключения при перебоях в сети
- Неожиданный рестарт контейнера

Troubleshooting: Расстояния



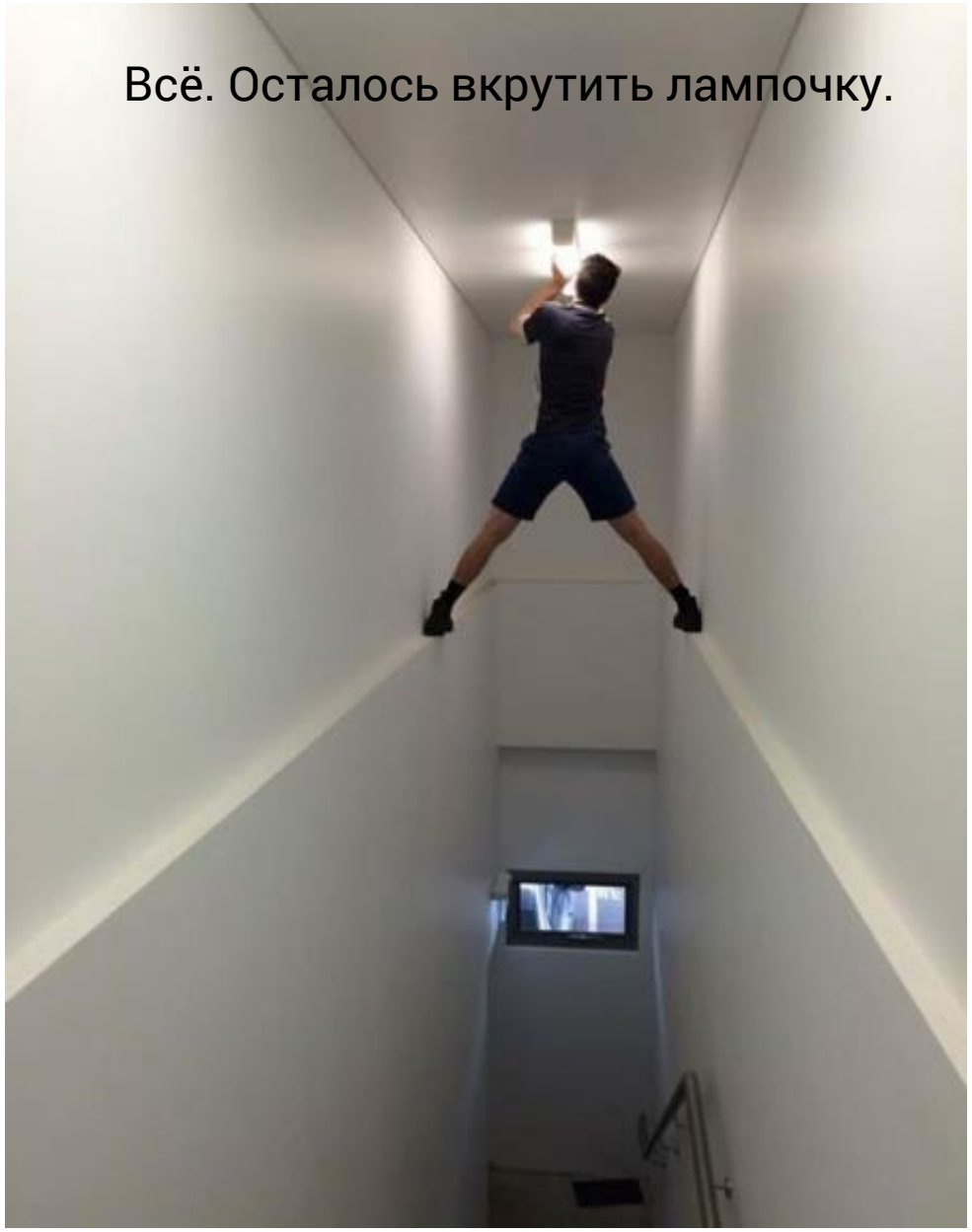
Troubleshooting: Аудит

- Personally identifiable information в логах
  - Pgaudit - не хватает
  - Собственный API – хранимые процедуры

- Ограничение доступа
  - Даем права только для тех, кому это необходимо
  - Выбираем датацентры которые сертифицированы (PCI DSS)

Команда проекта

- Architecture team
- Developer team
- DevOps
  - CloudOps/DevOps
  - SRE
- Monitoring Team
- DBA Team
- Security Team
- Application Security team
- QA and Performance testing team
- NOC, SOC



Роль DBA

- Роль DBA
  - Немного Архитектор
  - Немного DevOps
  - Немного SecOps
  - Немного Monitoring
  - Немного QA&Perf
  - Немного Developer
  - Linux Administrator


inCountry team


I told you, it's pure perfection


Data Residency-As-A-Service


competitor
inCountry
Customer