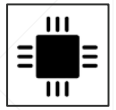
# PostgreSQL и MQTT в качестве системы обработки IoT данных

Камиль Исламов

# План



- Сравнение некоторых вариантов обработки IoT данных

- Возможности протокола MQTT

- Варианты интеграции плагина MQTT с PostgreSQL

- Реализация бизнес-логики на базе MQTT идеологии

- Итоги

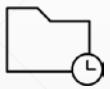
# Некоторые требования к IoT-устройствам

- Скорость передачи данных начиная от 2G

- Энергоэффективная аппаратная платформа
- Непрерывный поток данных от множества источников

- Необходимость «обратной» связи

- Функционал массового мониторинга с возможностью управления



- Возможность передачи данных после оффлайн

- Достаточная автономность/надёжность

# Некоторые способы обработки IoT Данных


- SOAP протокол (~XML)

- HTTP протокол – REST идеология

- HTTP протокол – собственная разработка

- SNMP протокол

- Собственный протокол

- Решения на базе Kafka

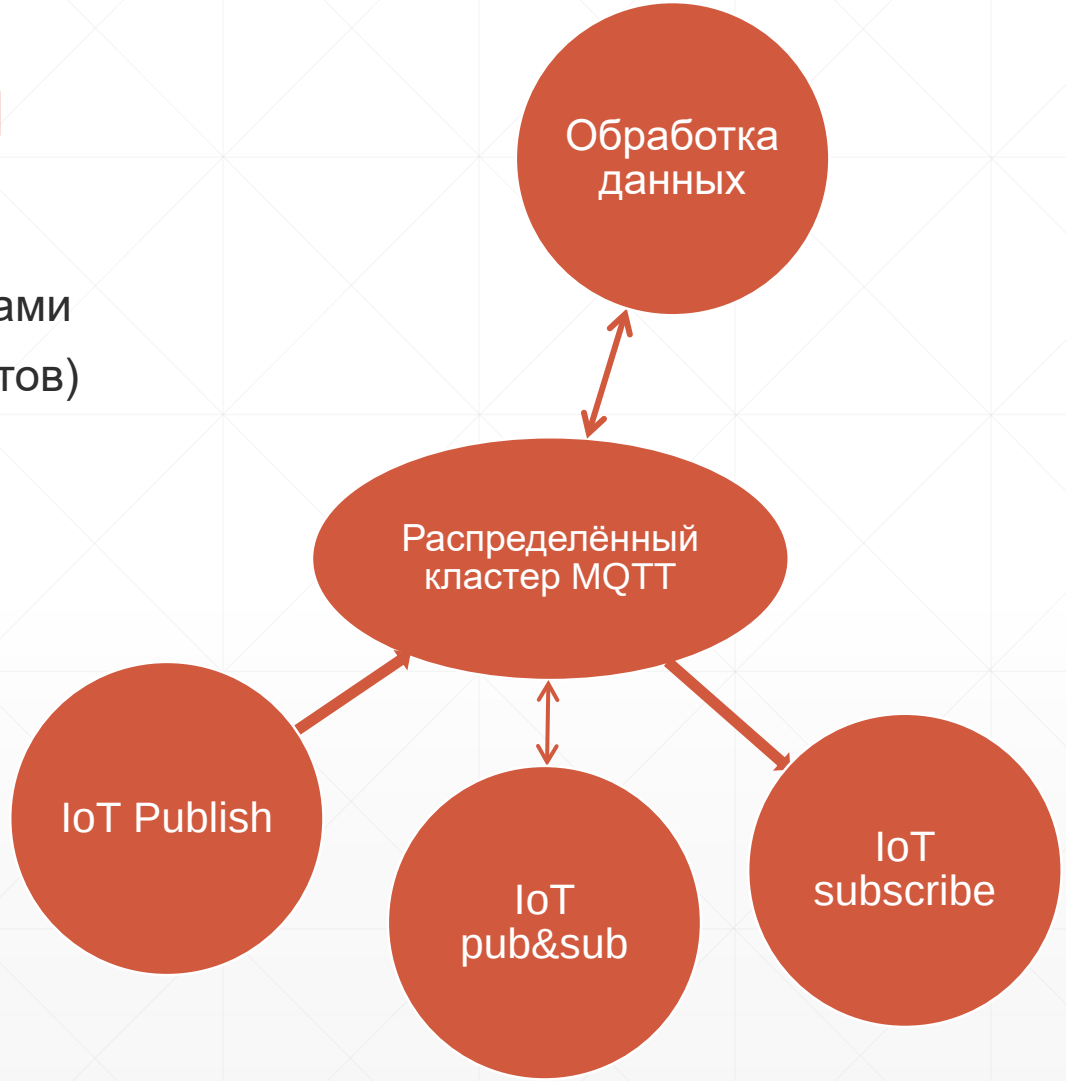
- MQTT протокол

# Некоторые преимущества не-HTTP

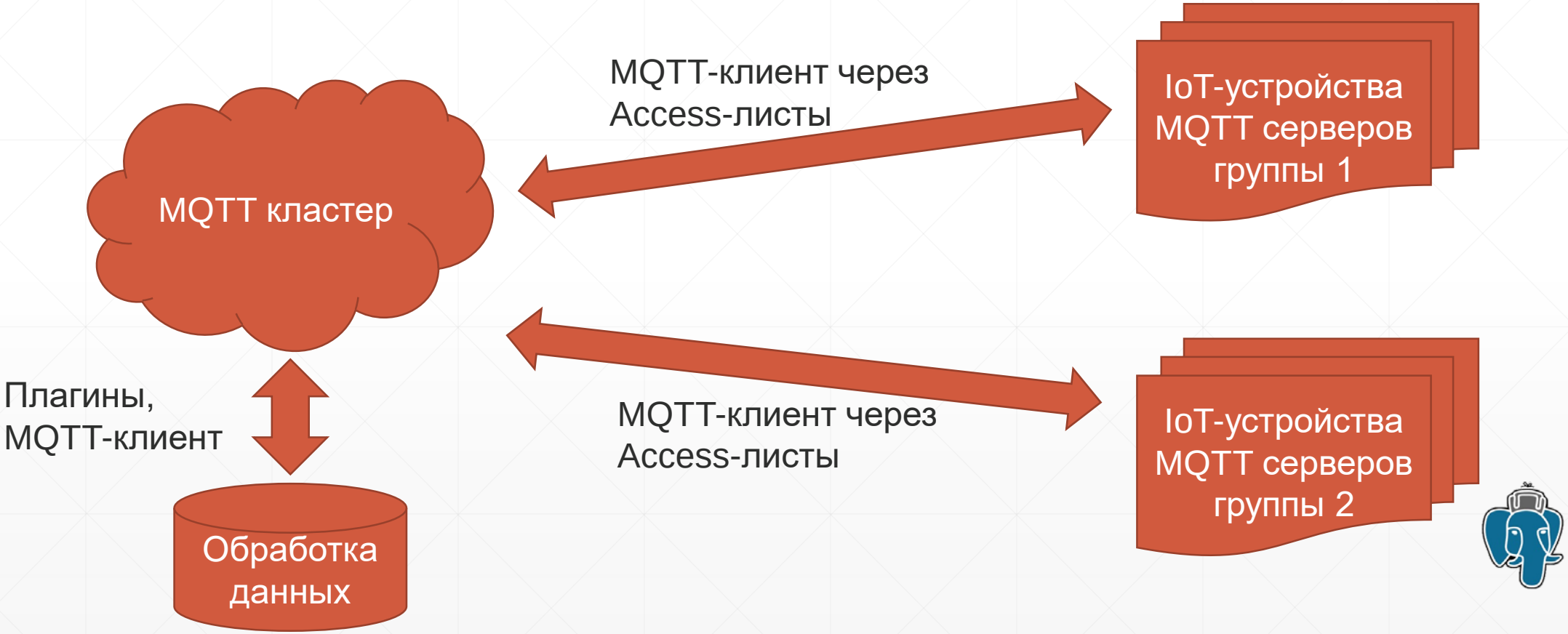
- Оптимизация накладных расходов на служебные данные
- Минимизация трафика к серверу
- Упрощение клиентской части IoT устройств
- Использование специфических преимуществ протокола/платформы
- Повышение эффективности единицы трафика
- Экономия ресурсов на масштабирование

# Описание MQTT идеологии

- Распределённый кластер
  - Поддержка постоянных соединений с клиентами
  - Доставка Сообщений согласно Topic (Адресатов)
- IoT Subscribe устройства
  - Подписываются на Topic по своему ACL
  - Получают Сообщения согласно Topic из ACL
- IoT Publish устройства
  - Определяют доступные Topic по своему ACL
  - Отправляют Сообщения на доступные Topic
- IoT pub&sub – доступ на оба действия
- Обработка данных
  - Доступ ко всем Сообщениям и их обработка



# Схема подключения IOT устройств к MQTT кластеру



# Возможности MQTT протокола

- Постоянный «онлайн» режим подключения (Socket)

- Двухсторонний обмен данными

- Минимальный объём служебных данных

- Проработанный RFC стандарт для большинства типовых задач
- Множество реализаций уровня Enterprise и Open-source

- Возможность использования встраиваемых плагинов

- В совокупности – это *мощный инструмент* обработка IoT данных

# MQTT: пользователи и акцесс-листы

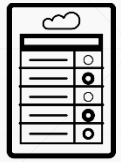
- Авторизация множеством способов
  - Логин-пароль в конфиг файле
  - СУБД (PostgreSQL, Mysql)
  - Mongo

- Гибкая система Access-List
  - Политика «Можно всё, кроме запрещённого»
  - Политика «Ничего нельзя, кроме разрешённого»
  - Для каждого топика возможность чтения и/или записи

# MQTT протокол для IoT: готовые решения

- Несколько Open-Source MQTT проектов с большим сообществом
  - Библиотеки для популярных языков программирования
  - Использования в смежных областях (Мессенджеры, Embedded, etc)
  - Обширная практика использования для IoT индустрии

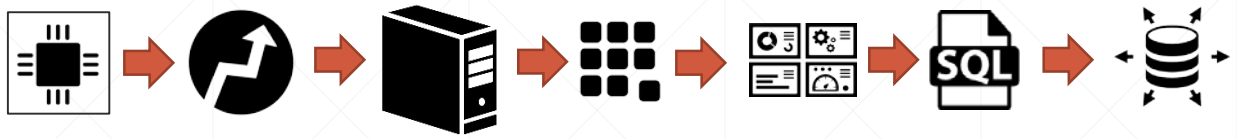
- Проприетарные Java MQTT-решения с гибкой поддержкой
  - Заявленные сверх масштабируемые решения
  - Плагины с возможностью интеграции в Enterprise системы
  - Возможность заказать разработку плагинов авторам MQTT решений



- Полноценный RFC для собственной имплементации
  - Возможность собственной реализации MQTT протокола

# PostgreSQL при обработка IoT-данных
# MQTT как альтернатива HTTP

- Архитектура при «классической» HTTP парадигме
  - IoT ⇨ HTTP клиент ⇨ HTTP сервер ⇨ Rest API ⇨ Backend сервер ⇨ Клиент psql ⇨ PG




- Архитектура на базе MQTT протокола
  - IoT ⇔ MQTT клиент ⇔ MQTT сервер ⇨ psql пул ⇨ PG

# Атрибуты данных IoT-устройств для MQTT

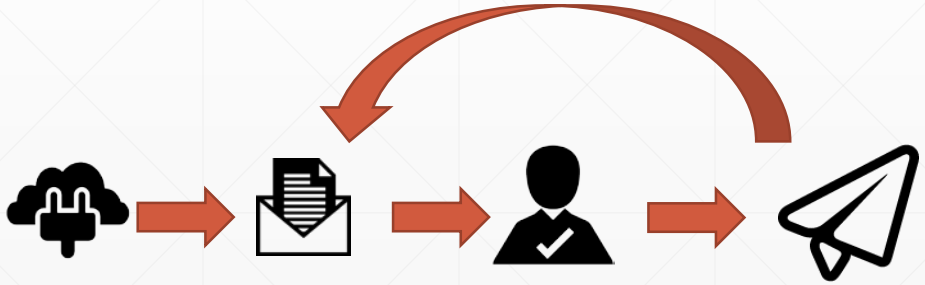
- Авторизация устройств с индивидуальным или групповым логин-паролем

  - MQTT логин + пароль
  - ИД клиента

- Одна «транзакция» = Публикация сообщения
  - Адресат сообщения (Topic)
  - Контент сообщения (Payload)
  - Надёжность доставки сообщения (QoS)

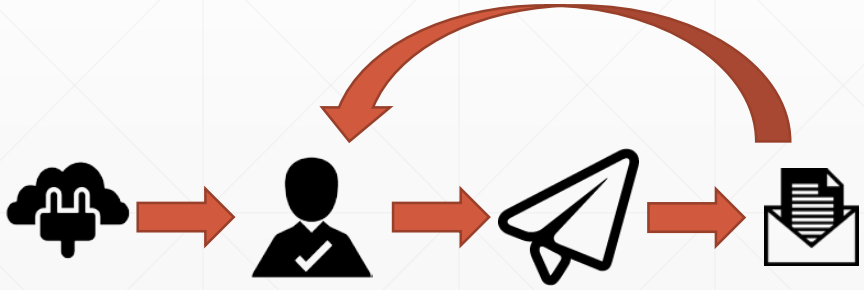
# Workflow-Out на стороне IoT устройства

1. IoT устройство инициирует и поддерживает MQTT коннект (и ACL)
2. IoT устройство формирует Payload Сообщения
3. Определяется Topic (адресат) Сообщения
4. Публикация Сообщения с Payload на Topic



# Workflow-In на стороне IoT устройства

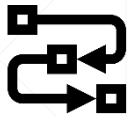
1. IoT устройство инициирует и поддерживает MQTT коннект (и ACL)
2. IoT устройство подписывается на получение
   Сообщений на определённый Topic (адресат), или несколько
3. Получает Сообщение на подписанный Topic
4. Выполняет действия в соответствии с Payload



# Варианты интеграции MQTT с Postgres

- BackEnd сервис, подписанный на публикации
  - Непрерывно работающий BackEnd процесс
  - Получает весь поток опубликованных Сообщений
  - Для подходящего сообщения подключается к БД
- Плагин к MQTT серверу
  - Выполняется в окружении облачного MQTT кластера
  - Оптимизирован на эффективное использование пула
  - Запускается непосредственно в момент Публикации

# Интеграция с MQTT через BackEnd сервис



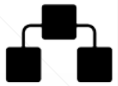
- Концептуальная архитектура реализации
  - Python, Java, NodeJS, … демон как MQTT клиент
  - Пакетная либо единичная передача Сообщений в БД

- Преимущества
  - Возможность реализации логики на BackEnd
  - Фильтрация обработки по ip, Topic, Payload

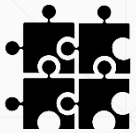
- Особенности
  - Отдельная инфраструктура, требующая ресурсов, снижающая надёжность
  - Дополнительное звено, снижающее производительность

# Интеграция с MQTT через плагин

- Концептуальная архитектура реализации
  - Плагин, запущенный в памяти MQTT кластера
  - Пул (или несколько) сессий, интегрированный в логику MQTT кластера

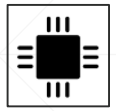


- Преимущества
  - Минимальный маршрут IoT ⇨ БД
  - Эффективное использование пула сессий



- Особенности
  - Разработка, отладка и сопровождение плагина
  - Пакетная обработка сообщений не предусмотрена в базовой архитектуре

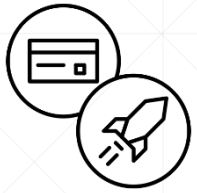
# Маршрут MQTT Сообщения IoT ⇨ СУБД



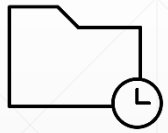
| Подсистема | Описание | Роль |
|---|---|---|
| IoT устройство | Распределённая сеть Embedded серверов | Подготовка и сбор данных для отправки |
| MQTT на клиенте | Клиент и/или Брокер, выполненный как сервис обработки локальных данных | Формирование и публикация Payload из собранных данных |
| MQTT на сервере | Облачный кластер, к которому постоянно подключены устройства | Маршрутизация опубликованных сообщений |
| MQTT плагин + PSQL пул | Интегрированный в MQTT сервис, | Формирует и отрабатывает очередь обращений к БД |

# MQTT плагин как основа бизнес-логики

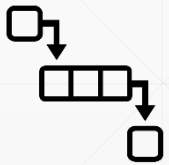

- Каждое Сообщение вызывает транзакцию в пуле соединений



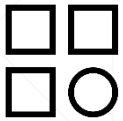
- Обработка данных в реальном времени
  - Планирование цепочки хранимых процедур
  - Разработка бизнес логики с индексированной обработкой данных
  - Ограничена задачами с минимальной стоимостью



- Отложенная обработка Сообщений
  - Возможность масштабирования и планирования
  - Оптимизация вставки данных
  - Возможность приоритезации отложенной обработки



# Примеры обработки в реальном времени



- Важные компоненты бизнес-логики приложения, управление
- Мобильные приложения
  - Быстрый обмен данными, влияющими на отображение информации
  - Показания приборов, аналитика в реальном времени
  - Элементы управления оборудования с участием бизнес-логики

- Десктопные и web-приложения
  - Контрольная панель мониторинга и управления
  - Массовая обработка команд и результатов выполнения
  - Эффективные UX/UI решения

# Вариант обработки в реальном времени

1. IoT устройство формирует и публикует Сообщение

2. MQTT сервер принимает и обрабатывает Сообщение

3. MQTT плагин формирует запускает в пуле сессий транзакцию

4. В качестве транзакции – вызов хранимой процедуры

5. Для всех Сообщений запускается единая процедура-роутер

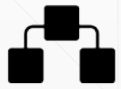
6. Процедура-роутер в зависимости от Topic запускает конечную процедуру, реализующую бизнес-логику

# Запуск единой процедуры-роутера

- Для каждого Сообщения запускается единая процедура-роутер

- В зависимости от Topic (или Payload) вызов перенаправляется на целевую процедуру бизнес-логики

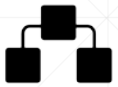
- Преимущества
  - Минимум кода и изменения базовой логики MQTT инфраструктуры
  - Возможность оперативно сменить MQTT платформу



- Варианты масштабирования
  - Вызов удалённых процедур, например PL-Proxy
  - Использование FDW для прямого доступа к разным инстансам, БД

# Варианты логики в MQTT плагине

- Прямые SQL запросы к какой-либо одной БД
  - Базовое журналирование



  - Реализация части бизнес-логики в коде плагина
  - Управление необходимостью обращения к БД

- Выбор между различными целевыми БД в зависимости от Topic
- Выбор запуска той или иной функции


  - Разгрузка БД от функций валидации и подготовки параметров
  - Оптимизация необходимости вызовов процедур

# Особенности приложений на базе MQTT плагина

- Проектирование с учётом двухстороннего обмена данными
  - Множественные активные элементы управления
  - Отображение актуальных данных в реальном времени

- Варианты эффективного сочетания MQTT / HTTP
  - Авторизация, регистрация – HTTP
  - Элементы интерфейса со статичными элементами
    - Перечень приборов – HTTP, показания – MQTT
    - Агрегированный график – HTTP, график реального времени – MQTT

- Собственные финансовые, транзакционные данные – HTTP, не критичные динамические значения – MQTT

# Преимущества концепции MQTT плагин + Postgres

- Эффективное использование возможностей асинхронной платформы

- Использование преимуществ пула сессий, интегрированного в кластер

- Двухсторонний обмен данными с применением бизнес-логики

- Использование преимуществ реализации бизнес-логики через функции

- Оптимизация ресурсов при разработке и эксплуатации

# Вопросы


- kam@kambox.ru
- ikamil
- kamil.islamov



PostgreSQL и MQTT в качестве системы обработки IoT данных