МОПС

IoT приложение

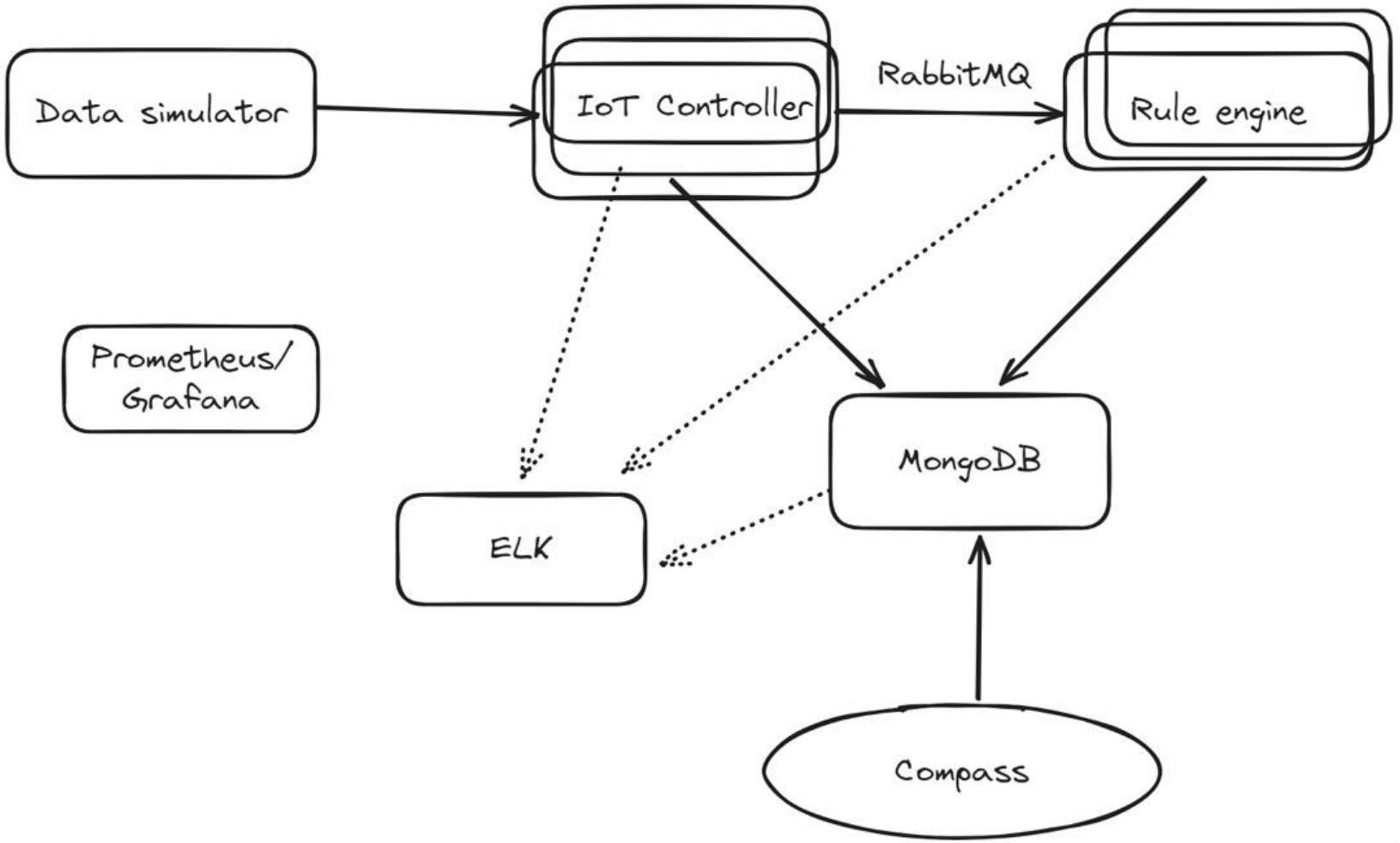
Коновалов Арсений

P4116

Воронина Дарья

P4116

Architecture



Our choice – Microservices + Event-driven IoT

Компоненты

1. Data Simulator — генерирует телеметрию от устройств и отправляет её по HTTP.
2. IoT Controller — принимает и валидирует сообщения, сохраняет сырые данные и публикует события.
3. RabbitMQ — брокер сообщений для асинхронной передачи данных между сервисами.
4. Rule Engine — обрабатывает поток сообщений, применяет мгновенные и длящиеся правила.
5. MongoDB — хранение входных (raw) сообщений устройств.
6. PostgreSQL — хранение сработавших алёртов.
7. Prometheus / Grafana, ELK — мониторинг и логирование системы.

Поток данных

1. Симулятор отправляет телеметрию устройств в IoT Controller по HTTP.
2. Controller валидирует данные, сохраняет их в MongoDB и публикует событие в RabbitMQ.
3. Rule Engine асинхронно получает сообщения из очереди и применяет правила обработки.
4. При срабатывании правила алёрт сохраняется в PostgreSQL и отображается в UI.

Our road...step by step

План из 15 этапов

1. Архитектура: формат сообщений, очередь, коллекции/таблицы, события алёртов (готово).
2. Репо-скелет: папки services/ui/infra/docs, базовый README/Makefile заготовки, .env.example.
3. Docker Compose каркас: сервисы (controller, rule-engine, simulator, mongo, rabbitmq, postgres, grafana, elastic+kibana, prometheus+exporters), сети/volumes.
4. Схемы данных: Mongo коллекция/индексы, Postgres `alerts` DDL, env образцы.
5. Flask IoT controller: `/ingest` валидация JSON, запись в Mongo, publish в RabbitMQ, healthcheck.
6. Rule engine worker: consume из очереди, мгновенные/длящиеся правила (10 пакетов), запись алёртов в Postgres.
7. Data simulator: 15 устройств, 1 msg/сек, конфигурируемая частота/число устройств.
8. Метрики: Prometheus client в Python, rabbit/mongo/postgres exporters, базовые метрики потоков.
9. Логи: структурированные JSON, Filebeat/Fluent Bit в Elasticsearch, базовые Kibana dashboards.
10. UI: React/Vite фронт для пакетов/алёртов/статусов, вызов симулятора.
11. Grafana: дашборды по метрикам потока, задержкам, алёртам.
12. Тесты: интеграционные для controller (Mongo+Rabbit) и rule engine (Rabbit+Postgres).
13. Документация запуска: инструкции docker compose, примеры curl, env.
14. Финальная проверка: прогон compose, smoke-тесты, метрики/логи/UI доступность.
15. Презентация PDF: архитектура, пайплайны, метрики/логи, выводы.

Another design principles

- Design for self-healing:
  1. Retry + DLQ
  2. Circuit breakers
  3. Transaction compensation
- Make all things redundant
  1. Несколько Rule Engine consumers
  2. Репликация БД
- Partitioning
  1. Шардирование по device_id
  2. Партиции очередей
- Use managed services
  1. Managed DB
  2. Managed message broker
  3. Managed monitoring
- Design for evolution
  1. DSL для правил
  2. Contract testing
  3. Versioned events
- Centralized identity
  1. Нет пользователей
  2. OAuth2 / Keycloak
- Design for operations
  - Distributed tracing
  - Алерты SLO
- Minimize coordination
  1. CQRS
  2. Event sourcing
  3. External state store