The show must go on: инструменты нагрузки и рецепты оптимизации онлайн конференции

Владимир Красильщик
JUG Ru Group
Joker - Autumn 2023

О себе

- Прагматичный java программист из СПб
- Java developer @ JUG Ru Group
- 20 лет разработчик
- 10 лет докладчик на конференциях JUG Ru Group
- 4 года преподаватель и ментор


НИИ


RD
Software


Continental


DAIMLERCHRYSLER


EXIGEN
services


T··Mobile·

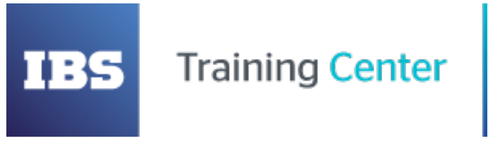

LUXOFT


Deutsche Bank


Яндекс
Маркет


DINS


CSC


IBS
Training Center


epam


SANDOZ
A Novartis
Division


КОРУС
КОНСАЛТИНГ


gm
getmentor.dev


JUG.RU
group


>h

План доклада

План доклада

1. Погружение в контекст онлайна JUG Ru Group, задачи и цели на 2023 год

План доклада

1. Погружение в контекст онлайна JUG Ru Group, задачи и цели на 2023 год

2. Знакомство с основными компонентами онлайна

План доклада

1. Погружение в контекст онлайна JUG Ru Group, задачи и цели на 2023 год

2. Знакомство с основными компонентами онлайна

3. Выбор инструмента нагрузочного тестирования онлайна (спойлер: gatling)

План доклада

1. Погружение в контекст онлайна JUG Ru Group, задачи и цели на 2023 год

2. Знакомство с основными компонентами онлайна

3. Выбор инструмента нагрузочного тестирования онлайна (спойлер: gatling)

4. Построение базового нагрузочного сценария, заведение нагрузочных ботов

План доклада

1. Погружение в контекст онлайна JUG Ru Group, задачи и цели на 2023 год

2. Знакомство с основными компонентами онлайна

3. Выбор инструмента нагрузочного тестирования онлайна (спойлер: gatling)

4. Построение базового нагрузочного сценария, заведение нагрузочных ботов

5. Выбор инструмента деплоя и запуска нагрузки (спойлер: nanocloud + 9 VM)

План доклада

1. Погружение в контекст онлайна JUG Ru Group, задачи и цели на 2023 год
2. Знакомство с основными компонентами онлайна
3. Выбор инструмента нагрузочного тестирования онлайна (спойлер: gatling)
4. Построение базового нагрузочного сценария, заведение нагрузочных ботов
5. Выбор инструмента деплоя и запуска нагрузки (спойлер: nanocloud + 9 VM)
6. Основные оптимизации для увеличения пропускной способности онлайна

План доклада

1. Погружение в контекст онлайна JUG Ru Group, задачи и цели на 2023 год
2. Знакомство с основными компонентами онлайна
3. Выбор инструмента нагрузочного тестирования онлайна (спойлер: gatling)
4. Построение базового нагрузочного сценария, заведение нагрузочных ботов
5. Выбор инструмента деплоя и запуска нагрузки (спойлер: nanocloud + 9 VM)
6. Основные оптимизации для увеличения пропускной способности онлайна
7. Выводы

Глава 1

1. Погружение в контекст онлайна JUG Ru Group, задачи и цели на 2023 год
2. Знакомство с основными компонентами онлайна
3. Выбор инструмента нагрузочного тестирования онлайна (спойлер: gatling)
4. Построение базового нагрузочного сценария, заведение нагрузочных ботов
5. Выбор инструмента деплоя и запуска нагрузки (спойлер: nanocloud + 9 VM)
6. Минимизация “шумов по данным” от нагрузочных тестов в проде
7. Выводы

Исторический контекст

Исторический контекст

1. Делаем онлайн с 2020 по 2 сезона в год

Исторический контекст

1. Делаем онлайн с 2020 по 2 сезона в год

2. Каждый сезон много внешних и внутренних изменений: эксперименты с UI/UX, игра, единый портал live, отдельные сайты конференций

Исторический контекст

1. Делаем онлайн с 2020 по 2 сезона в год

2. Каждый сезон много внешних и внутренних изменений: эксперименты с UI/UX, игра, единый портал live, отдельные сайты конференций

3. Осень 2023: 8 сезон, live4, первый сезон без драматических изменений, занимаемся продуктивизацией и подготовкой к мультиарендности

Исторический контекст

1. Делаем онлайн с 2020 по 2 сезона в год

2. Каждый сезон много внешних и внутренних изменений: эксперименты с UI/UX, игра, единый портал live, отдельные сайты конференций

3. Осень 2023: 8 сезон, live4, первый сезон без драматических изменений, занимаемся продуктивизацией и подготовкой к мультиарендности

4. Пришли к 2 основным страницам и 2 основным компонентам: расписание и доклад

Исторический контекст

1. Делаем онлайн с 2020 по 2 сезона в год

2. Каждый сезон много внешних и внутренних изменений: эксперименты с UI/UX, игра, единый портал live, отдельные сайты конференций

3. Осень 2023: 8 сезон, live4, первый сезон без драматических изменений, занимаемся продуктивизацией и подготовкой к мультиарендности

4. Пришли к 2 основным страницам и 2 основным компонентам: расписание и доклад

5. Исходные НФТ по производительности в ЛК в 2019: 10K (сейчас активно 80K+)

Исторический контекст

1. Делаем онлайн с 2020 по 2 сезона в год

2. Каждый сезон много внешних и внутренних изменений: эксперименты с UI/UX, игра, единый портал live, отдельные сайты конференций

3. Осень 2023: 8 сезон, live4, первый сезон без драматических изменений, занимаемся продуктивизацией и подготовкой к мультиарендности

4. Пришли к 2 основным страницам и 2 основным компонентам: расписание и доклад

5. Исходные НФТ по производительности в ЛК в 2019: 10K (сейчас активно 80K+)

6. Фактические внутренние НФТ по производительности для онлайн: “не лагающая трансляция в плейере для ~1K уников”

Исторический контекст

1. Делаем онлайн с 2020 по 2 сезона в год

2. Каждый сезон много внешних и внутренних изменений: эксперименты с UI/UX, игра, единый портал live, отдельные сайты конференций

3. Осень 2023: 8 сезон, live4, первый сезон без драматических изменений, занимаемся продуктивизацией и подготовкой к мультиарендности

4. Пришли к 2 основным страницам и 2 основным компонентам: расписание и доклад

5. Исходные НФТ по производительности в ЛК в 2019: 10K (сейчас активно 80K+)

6. Фактические внутренние НФТ по производительности для онлайн: “не лагающая трансляция в плейере для ~1K уников”

7. Частично нагружали HTTP и WS API: k6, Apache HttpClient

Мои задачи на 2023

Мои задачи на 2023

1. Подтвердить что держим 10K одновременных уников в онлайн трансляции

Мои задачи на 2023

1. Подтвердить что держим 10K одновременных уников в онлайн трансляции

2. Если не можем 10K, то провести необходимые оптимизации, чтобы смочь вывезти крупных внешних клиентов и мультиарендность

План действий

План действий

1. Выбрать инструмент нагрузки

План действий

1. Выбрать инструмент нагрузки

2. Создать 10K настоящих нагрузочных пользователей с билетами на настоящую конфу

План действий

1. Выбрать инструмент нагрузки
2. Создать 10K настоящих нагрузочных пользователей с билетами на настоящую конфу
3. Спроектировать и реализовать базовый нагрузочный сценарий и профиль нагрузки

План действий

1. Выбрать инструмент нагрузки
2. Создать 10K настоящих нагрузочных пользователей с билетами на настоящую конфу
3. Спроектировать и реализовать базовый нагрузочный сценарий и профиль нагрузки
4. Выбрать способ запуска нагрузочных тестов

План действий

1. Выбрать инструмент нагрузки
2. Создать 10K настоящих нагрузочных пользователей с билетами на настоящую конфу
3. Спроектировать и реализовать базовый нагрузочный сценарий и профиль нагрузки
4. Выбрать способ запуска нагрузочных тестов
5. Тестировать от 1K до 10K с шагом в 500 и оптимизировать, оптимизировать, оптим…

План действий

1. Выбрать инструмент нагрузки
2. Создать 10K настоящих нагрузочных пользователей с билетами на настоящую конфу
3. Спроектировать и реализовать базовый нагрузочный сценарий и профиль нагрузки
4. Выбрать способ запуска нагрузочных тестов
5. Тестировать от 1K до 10K с шагом в 500 и оптимизировать, оптимизировать, оптим…
6. Пропиарить себя и компанию, попросить прибавку к з/п, взорвать интернет

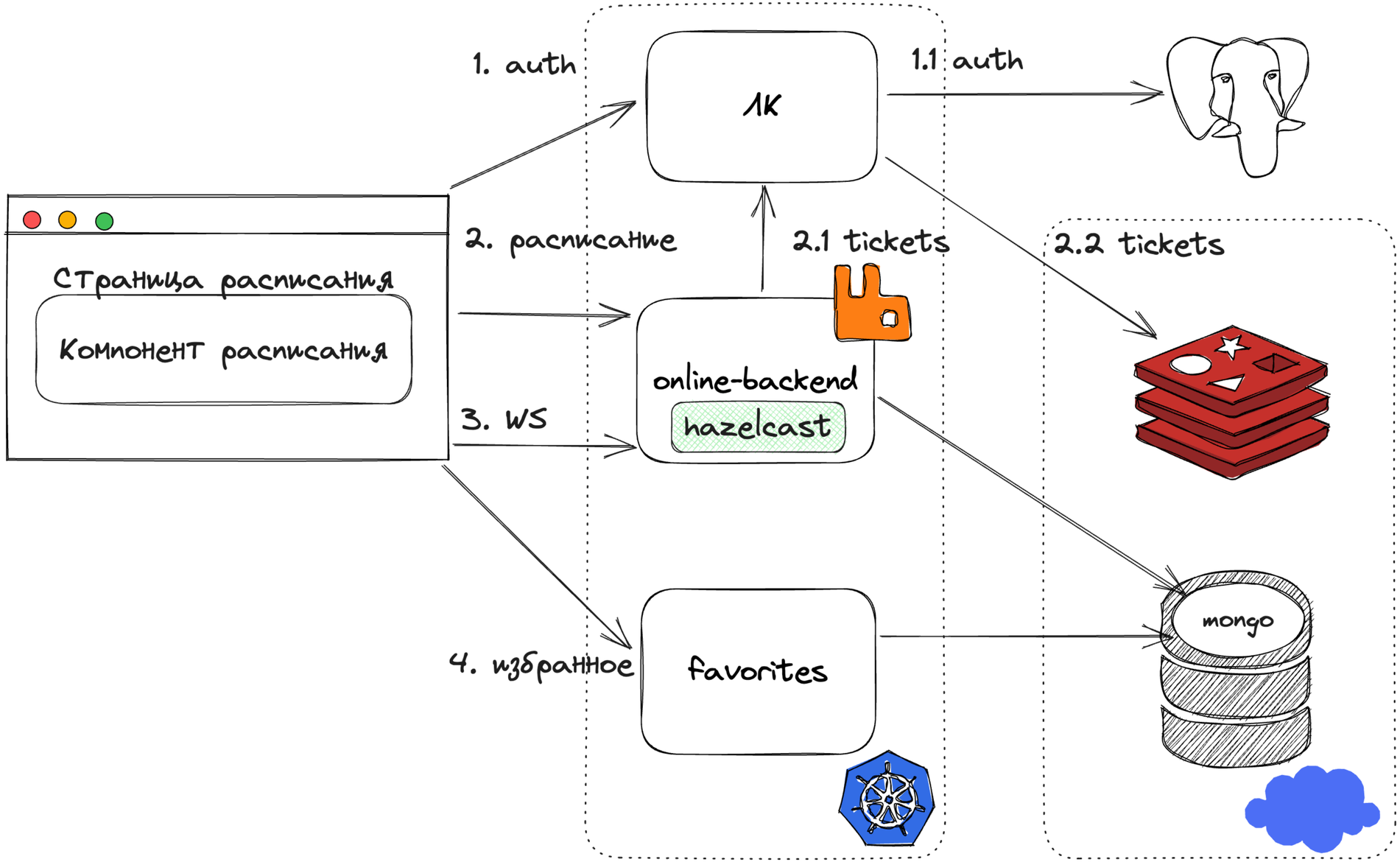
План действий

1. Выбрать инструмент нагрузки
2. Создать 10K настоящих нагрузочных пользователей с билетами на настоящую конфу
3. Спроектировать и реализовать базовый нагрузочный сценарий и профиль нагрузки
4. Выбрать способ запуска нагрузочных тестов
5. Тестировать от 1K до 10K с шагом в 500 и оптимизировать, оптимизировать, оптим…
6. ~~Пропиарить себя и компанию, попросить прибавку к з/п, взорвать интернет~~ Сделать доклад для специалистов с подобными задачами, поделиться опытом

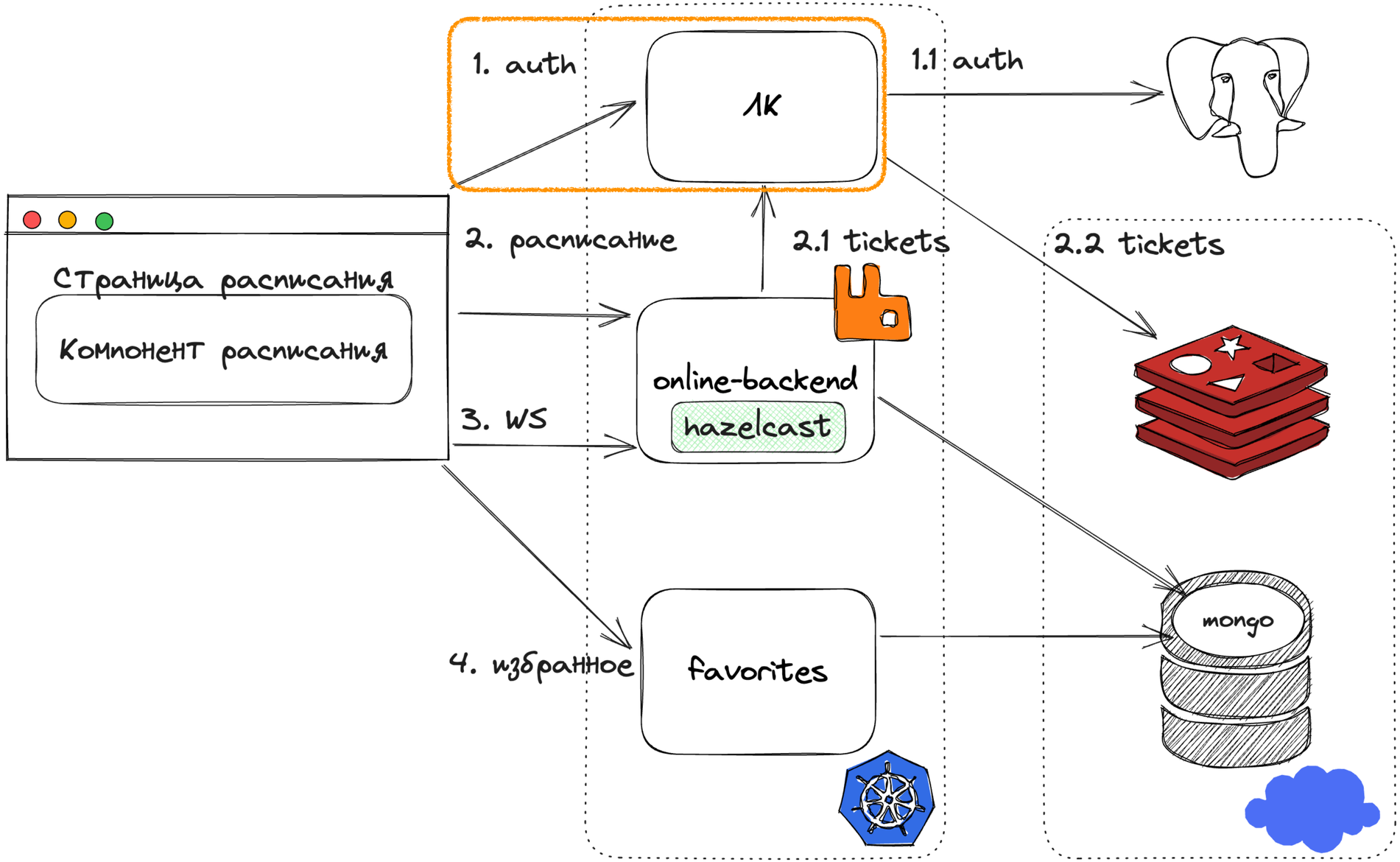
Глава 2

1. Погружение в контекст онлайна JUG Ru Group, задачи и цели на 2023 год
2. Знакомство с основными компонентами онлайна
3. Выбор инструмента нагрузочного тестирования онлайна (спойлер: gatling)
4. Построение базового нагрузочного сценария, заведение нагрузочных ботов
5. Выбор инструмента деплоя и запуска нагрузки (спойлер: nanocloud + 9 VM)
6. Основные оптимизации для увеличения пропускной способности онлайна
7. Выводы

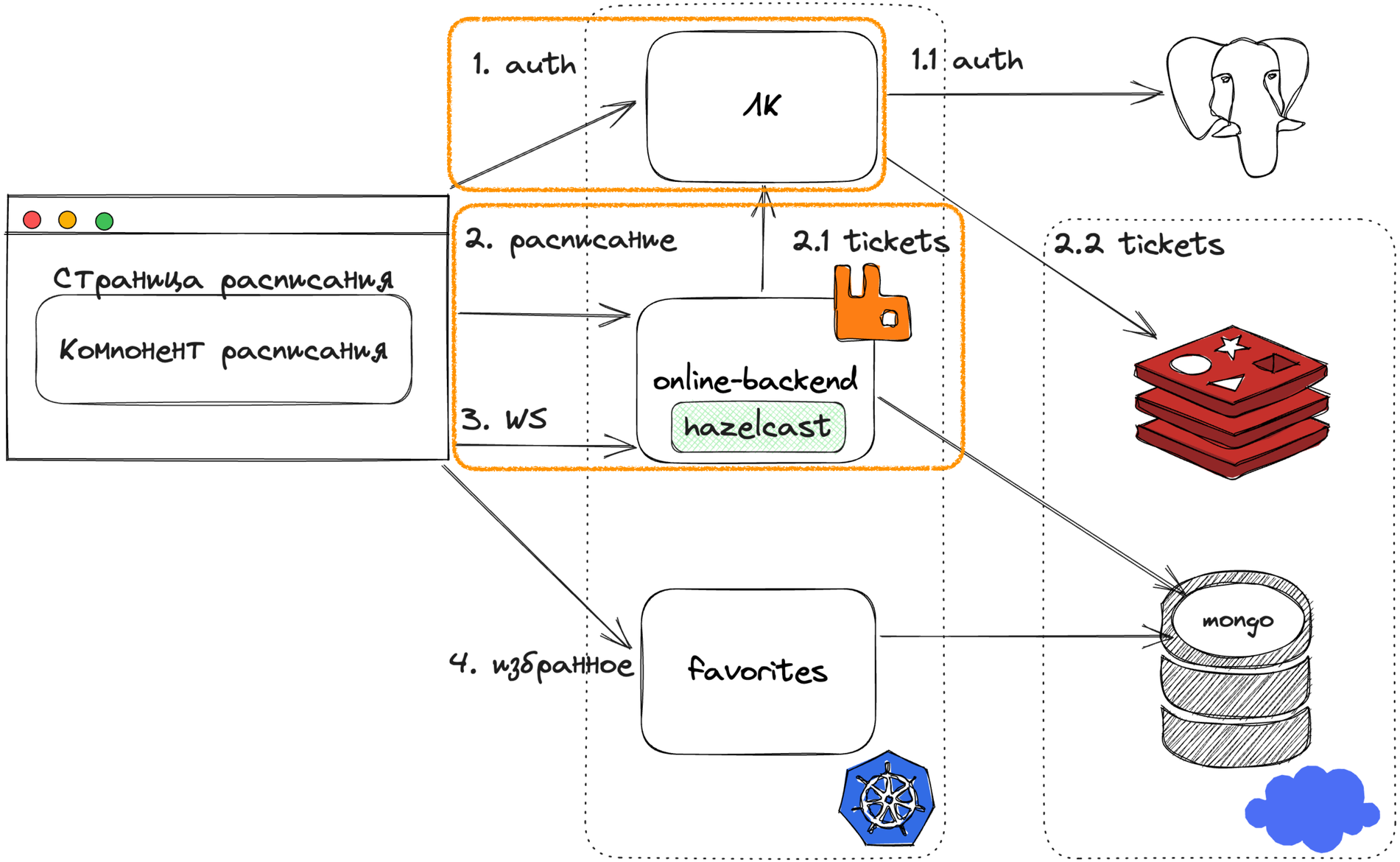
Страница и компонент расписания



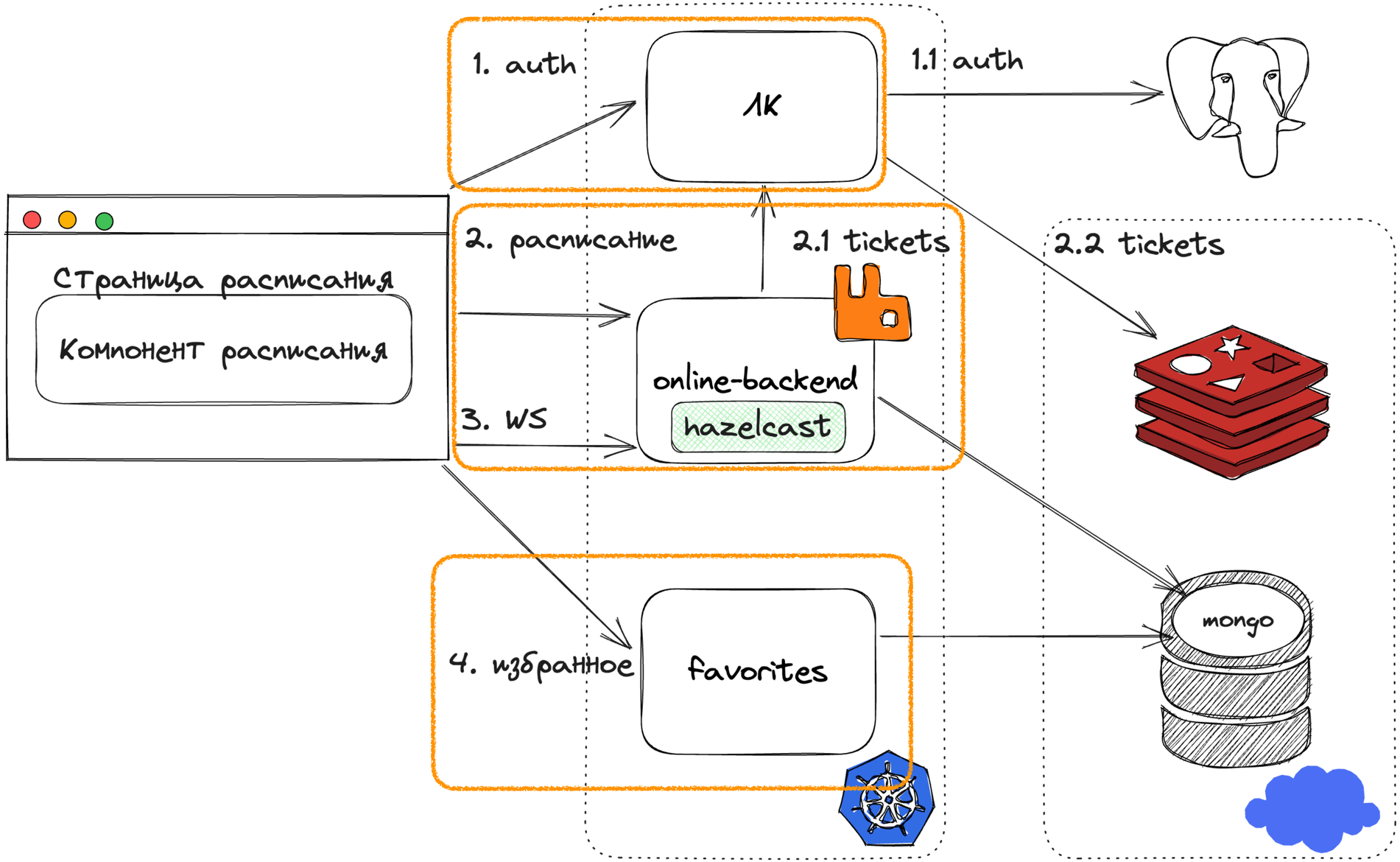
Страница и компонент расписания



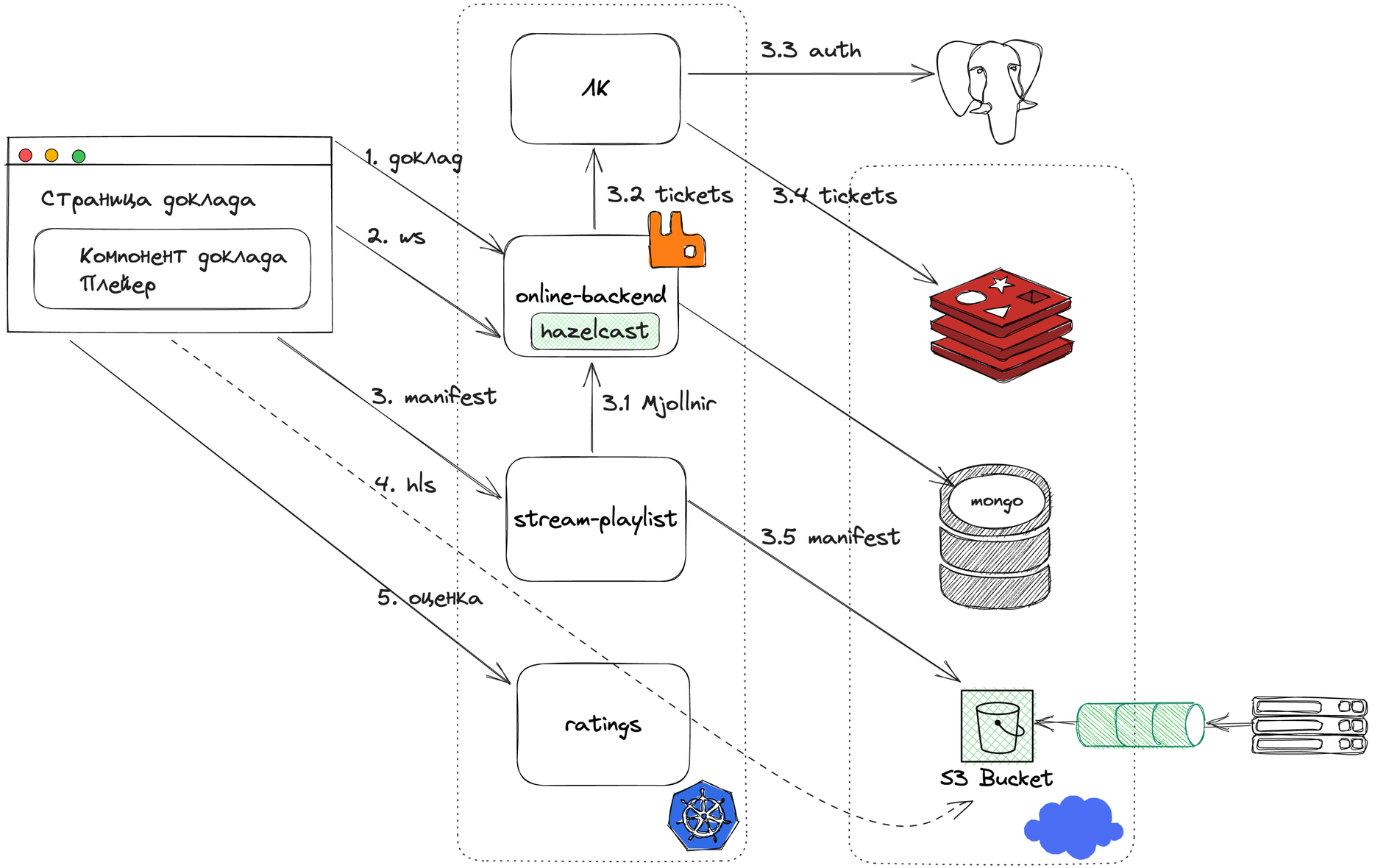
Страница и компонент расписания



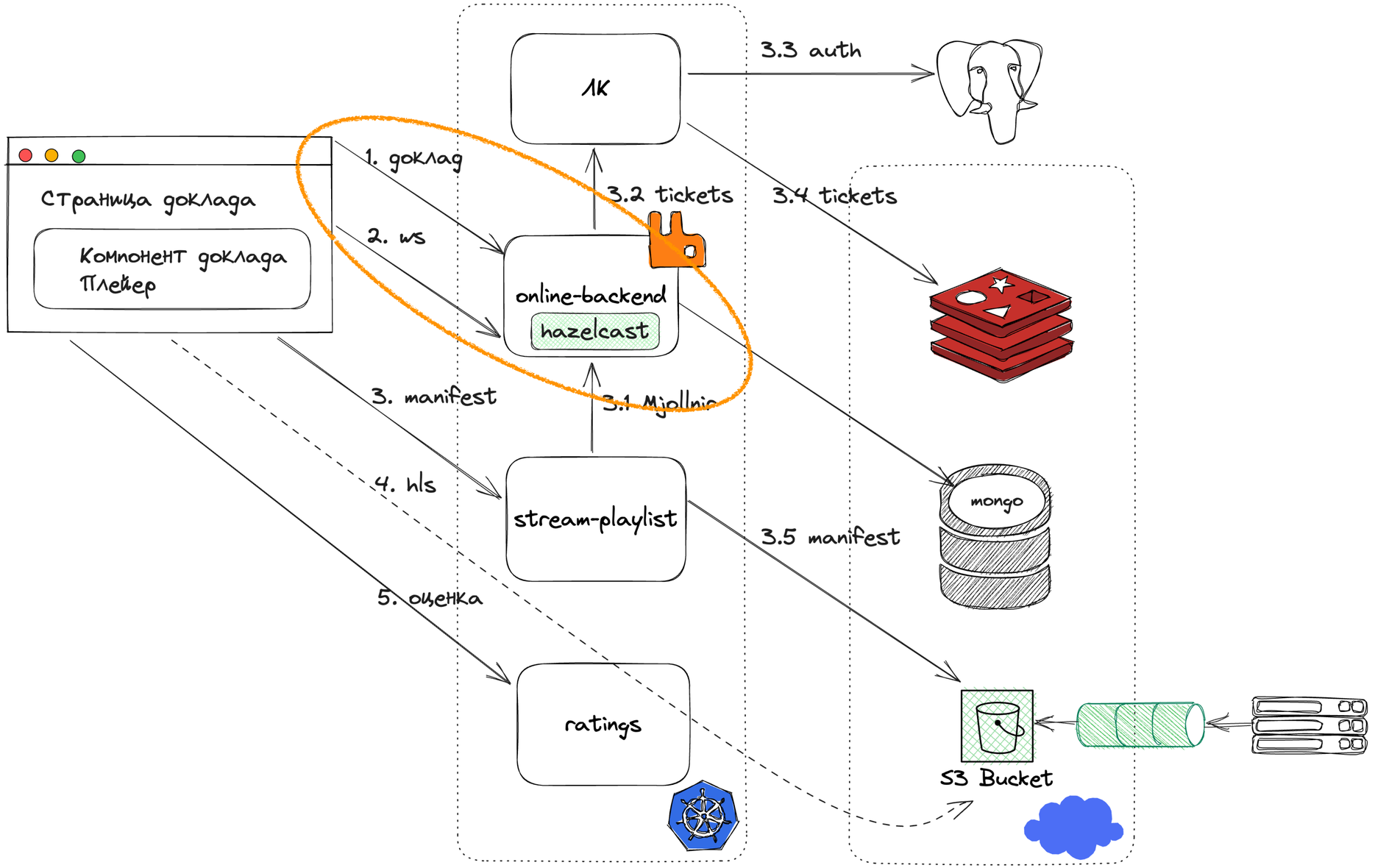
Страница и компонент расписания



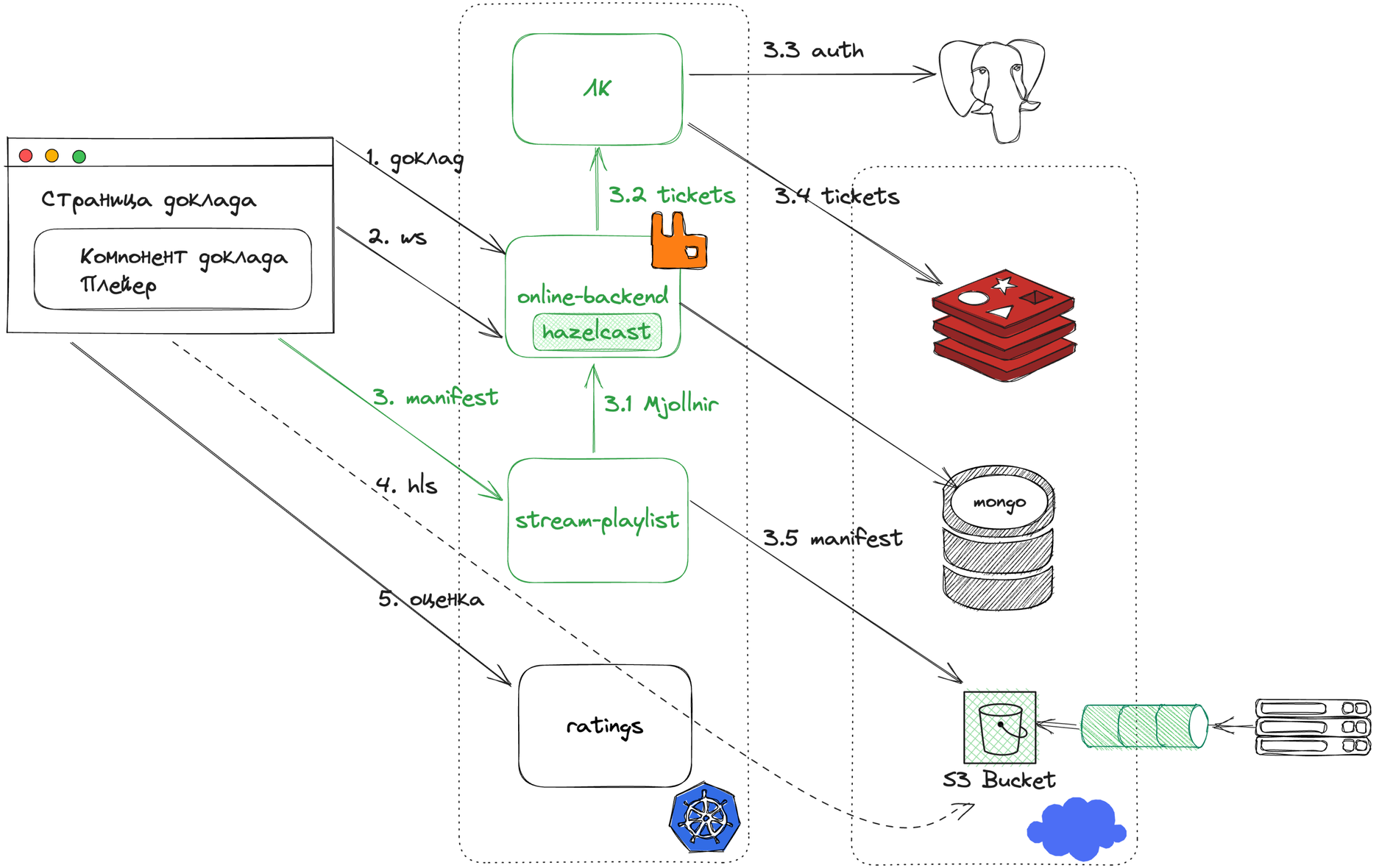
Страница и компонент доклада



Страница и компонент доклада



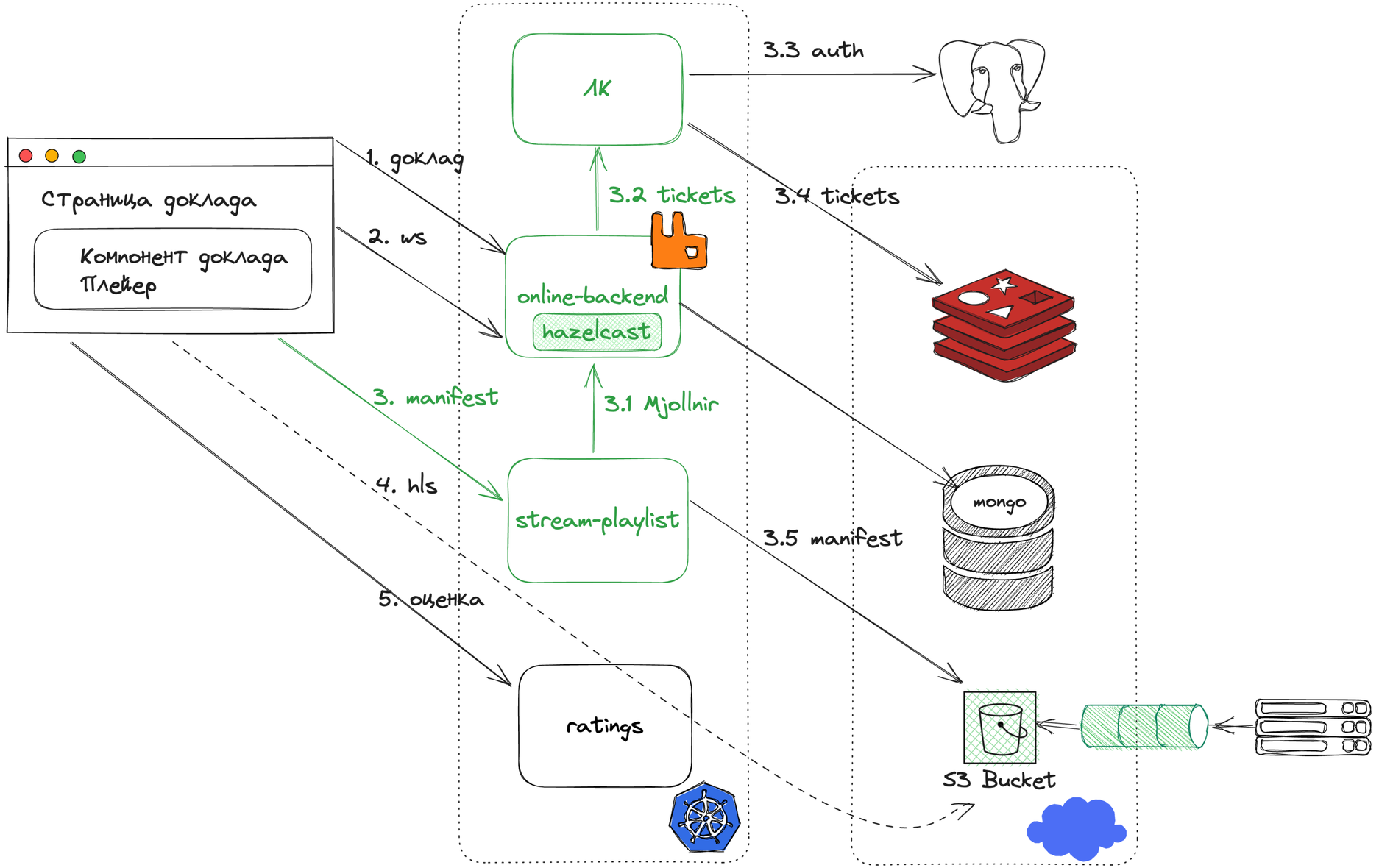
Страница и компонент доклада



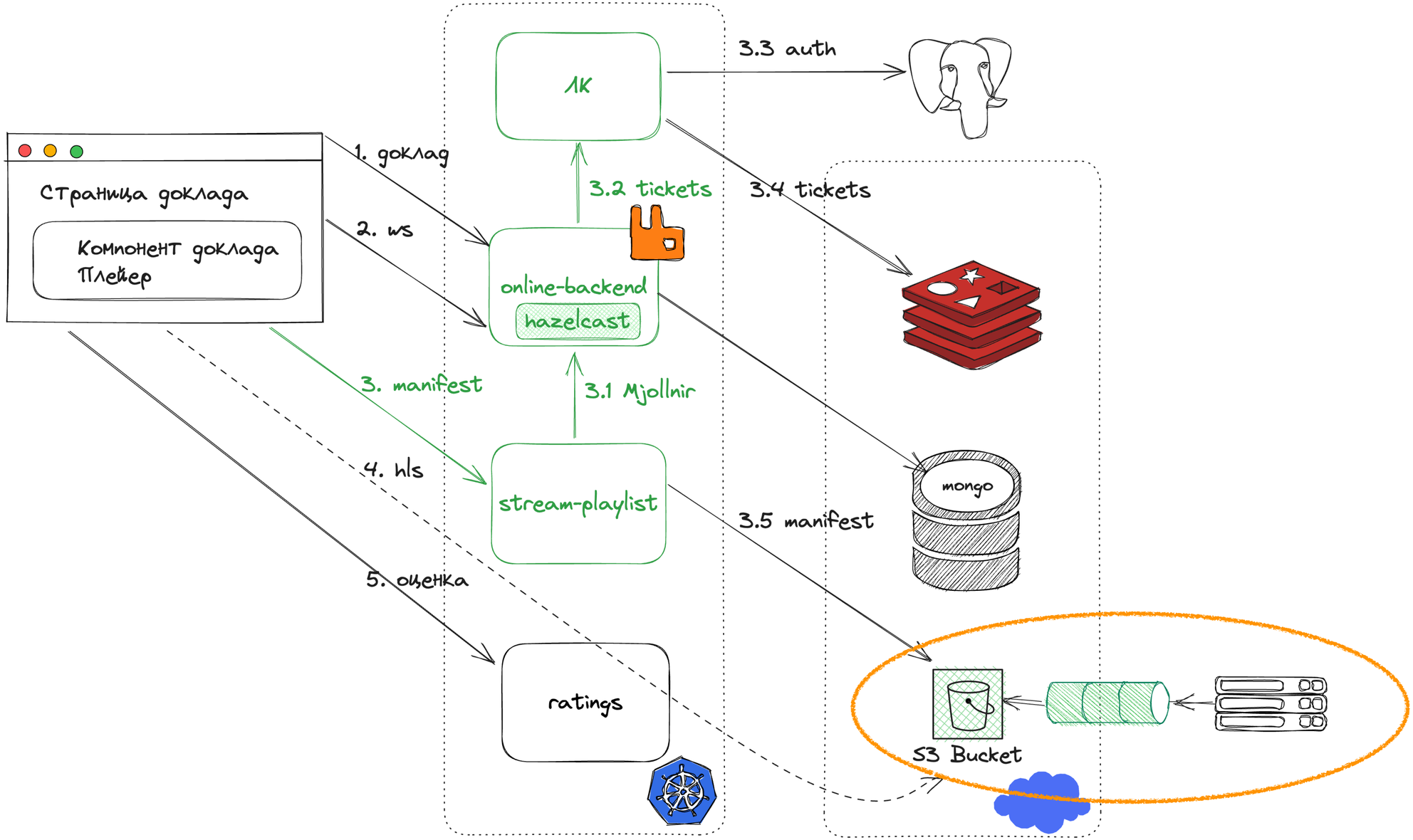
Пример манифеста: прямая трансляция

```
#EXTM3U
#EXT-X-PLAYLIST-TYPE:EVENT
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:3
#EXT-X-MEDIA-SEQUENCE:1
#EXTINF:3.00000,
https://stream.jugru.org/100171/100171-track1-day3/100171-track1-day3_1920/00000/100171-track1-
day3_1920_20230917_072929_01129.ts?
auth=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY3Rpdml0eUlkIjoyMDAwMjk0MywiZXZlbnRJZCI6MTAwMTcxLCJpYXQiOj
E2OTQ5MzYxMjcsImV4cCI6MTY5NDkzNjQyN30.hNAY3n0X7nUtedUF_X3NN3x5Rp5iiRUsBNNh8ULEIGo
#EXTINF:3.00000,
https://stream.jugru.org/100171/100171-track1-day3/100171-track1-day3_1920/00000/100171-track1-
day3_1920_20230917_072932_01130.ts?
auth=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY3Rpdml0eUlkIjoyMDAwMjk0MywiZXZlbnRJZCI6MTAwMTcxLCJpYXQiOj
E2OTQ5MzYxMjcsImV4cCI6MTY5NDkzNjQyN30.hNAY3n0X7nUtedUF_X3NN3x5Rp5iiRUsBNNh8ULEIGo
...
...
#EXTINF:3.00000,
https://stream.jugru.org/100171/100171-track1-day3/100171-track1-day3_1920/00000/100171-track1-
day3_1920_20230917_073702_01280.ts?
auth=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY3Rpdml0eUlkIjoyMDAwMjk0MywiZXZlbnRJZCI6MTAwMTcxLCJpYXQiOj
E2OTQ5MzYxMjcsImV4cCI6MTY5NDkzNjQyN30.hNAY3n0X7nUtedUF_X3NN3x5Rp5iiRUsBNNh8ULEIGo
```

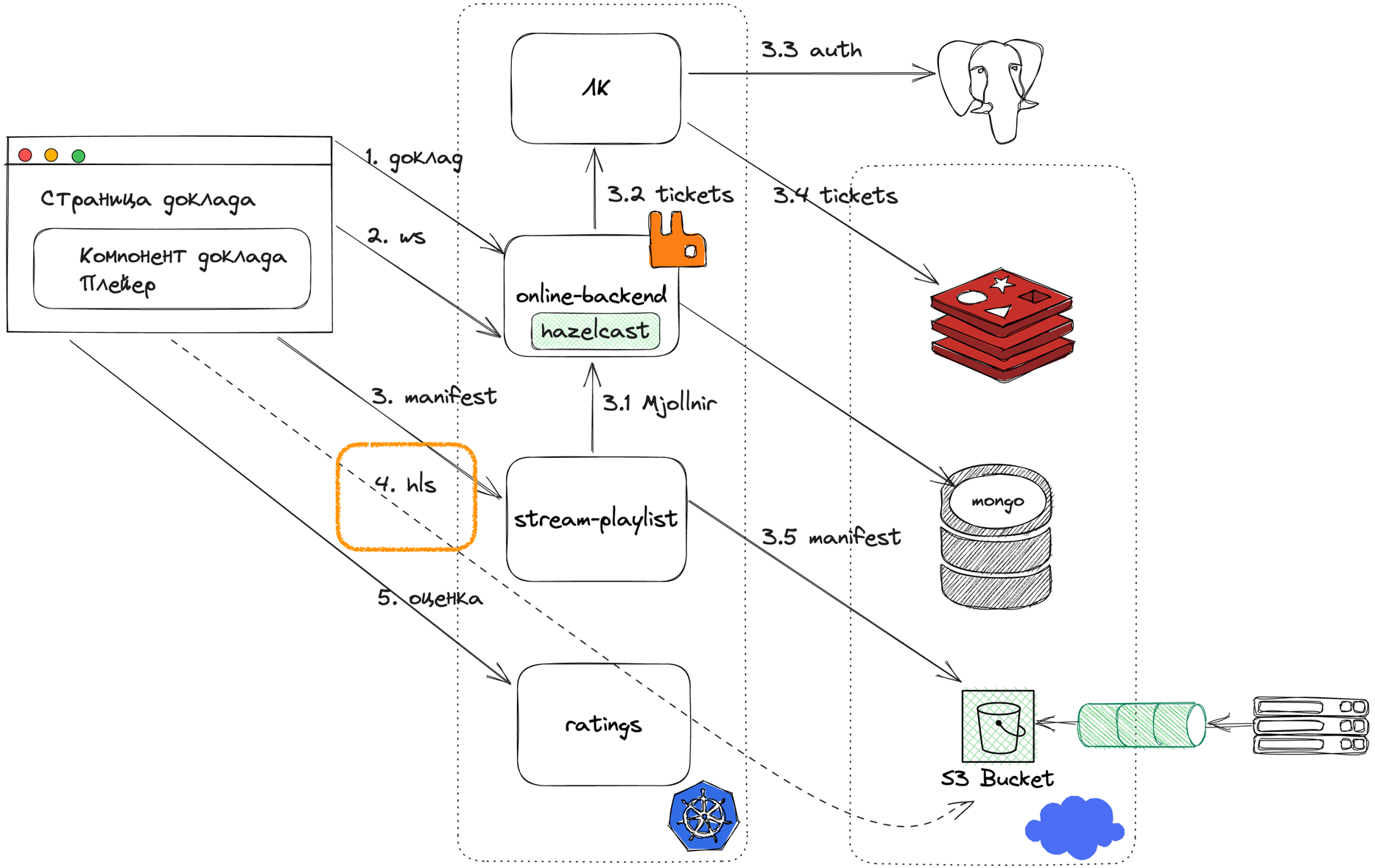
Страница и компонент доклада



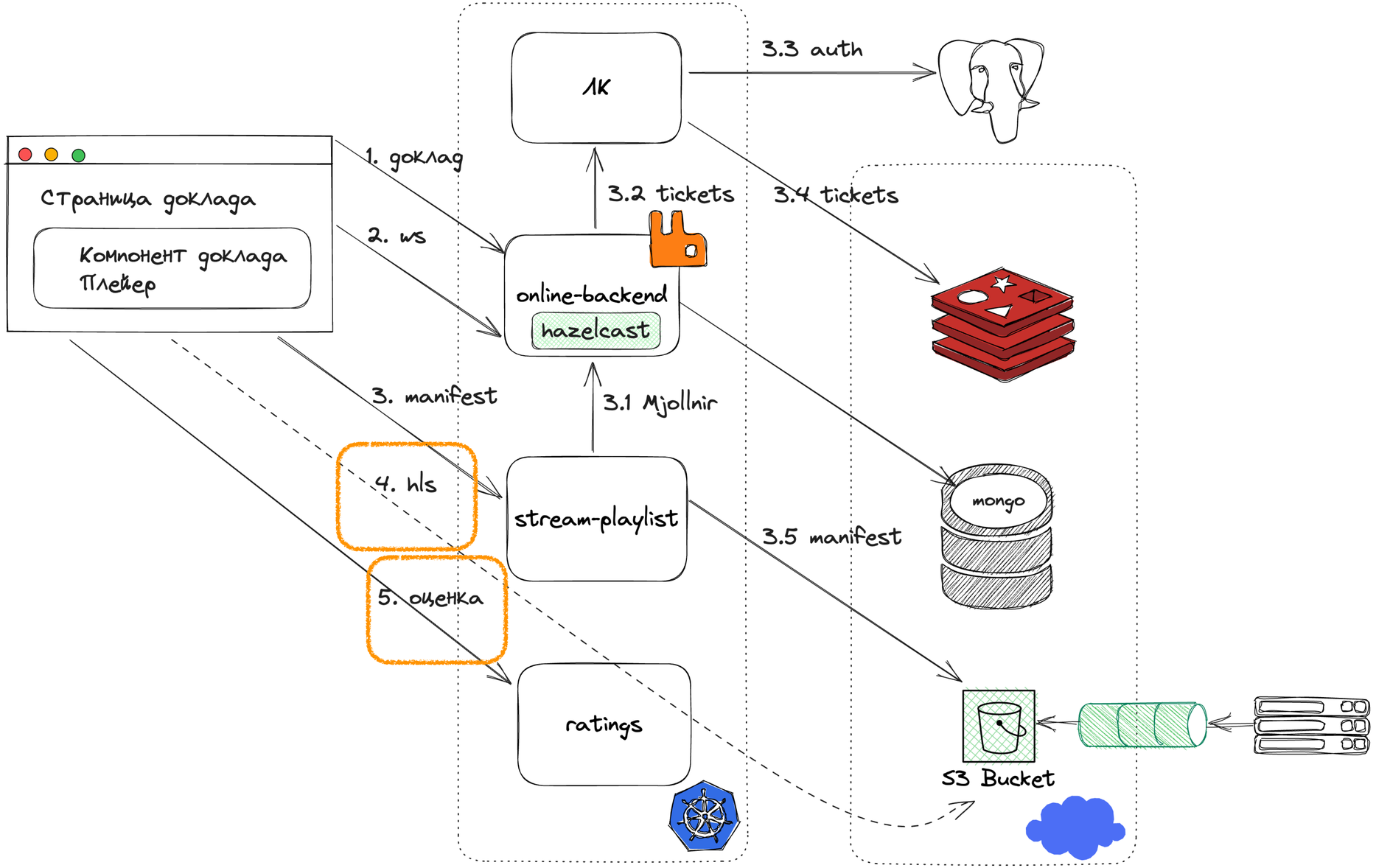
Страница и компонент доклада



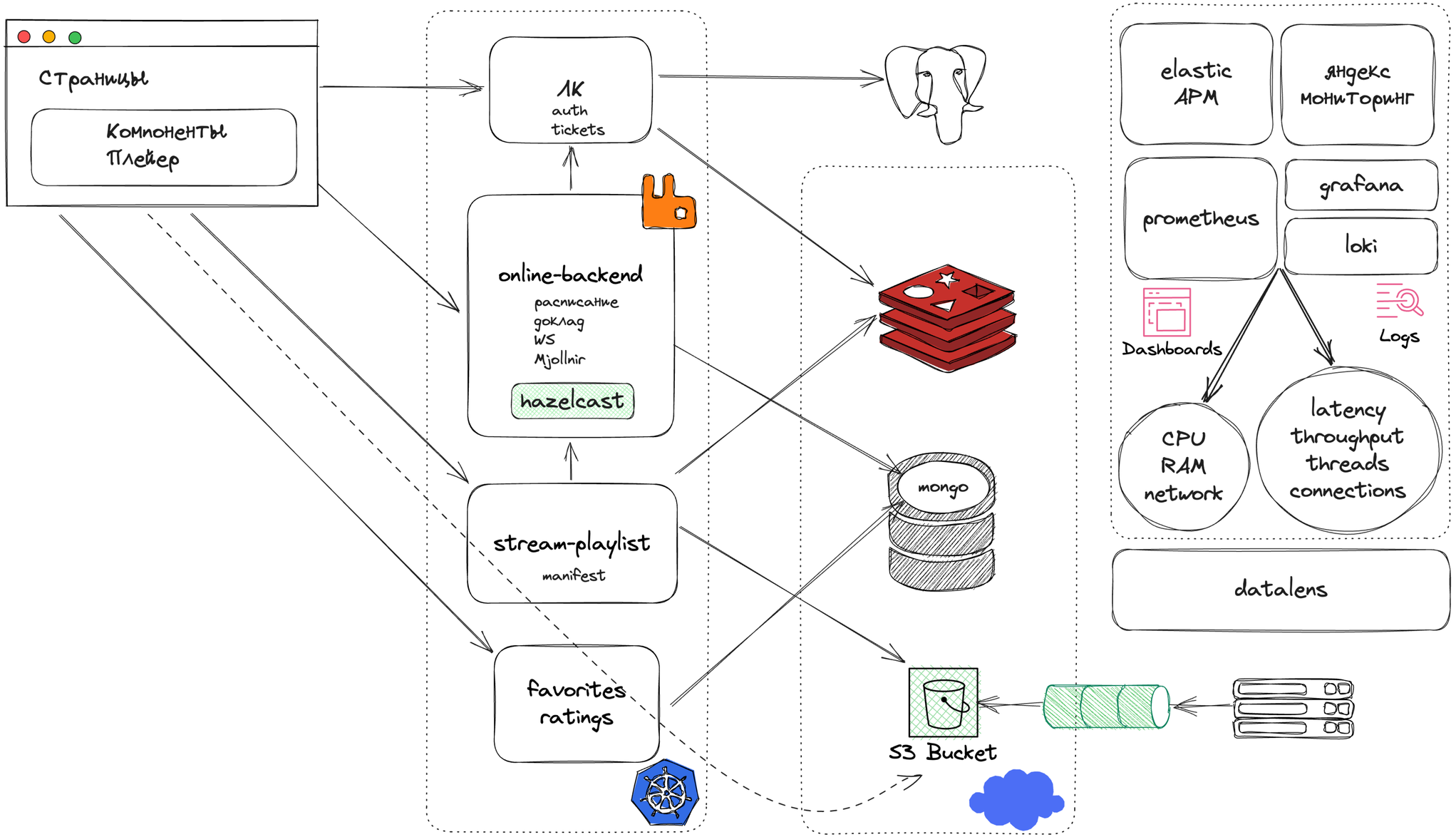
Страница и компонент доклада



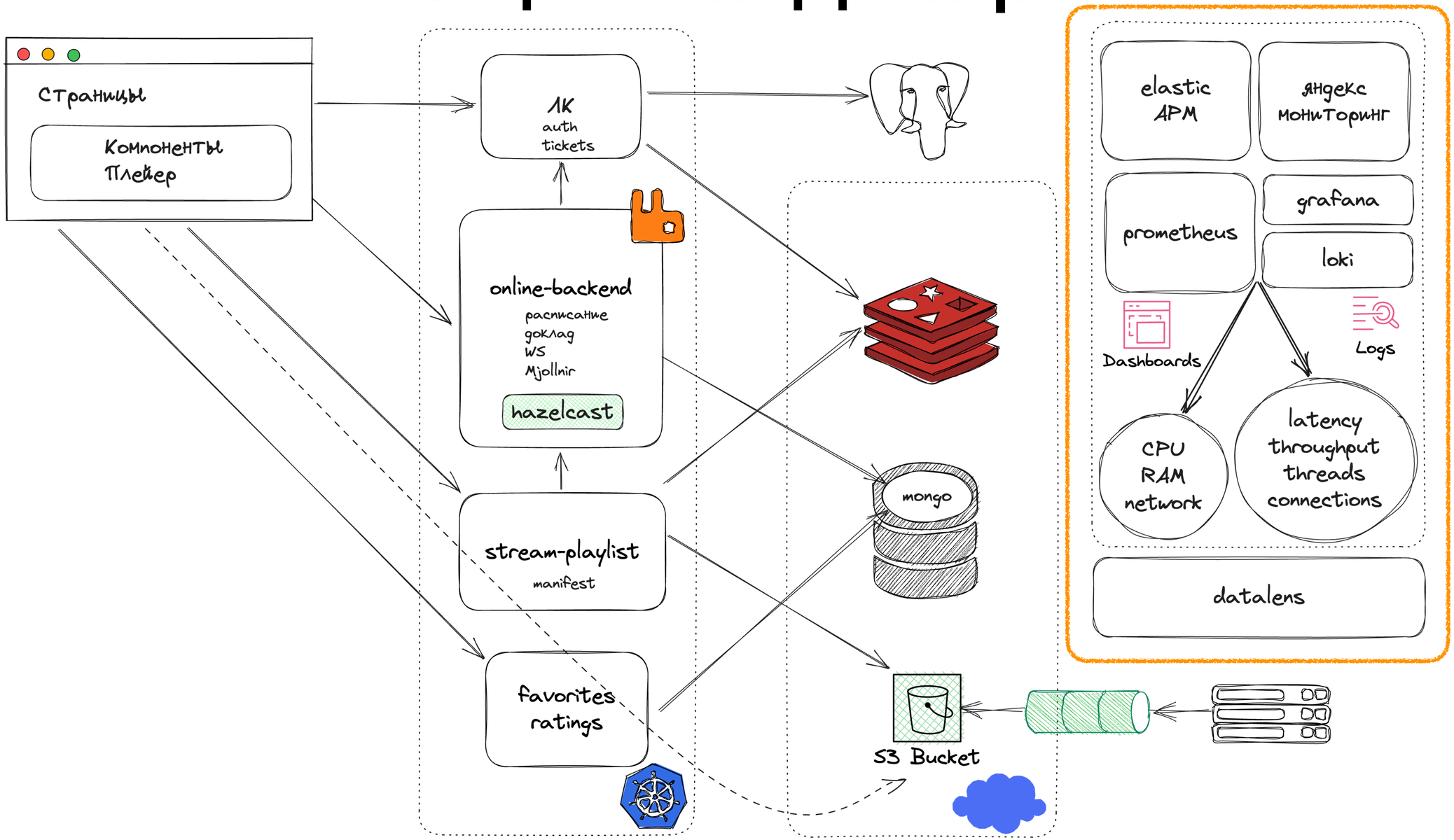
Страница и компонент доклада



Общий ландшафт



Общий ландшафт



Глава 3

1. Погружение в контекст онлайна JUG Ru Group, задачи и цели на 2023 год
2. Знакомство с основными компонентами онлайна
3. Выбор инструмента нагрузочного тестирования онлайна (спойлер: gatling)
4. Построение базового нагрузочного сценария, заведение нагрузочных ботов
5. Выбор инструмента деплоя и запуска нагрузки (спойлер: nanocloud + 9 VM)
6. Основные оптимизации для увеличения пропускной способности онлайна
7. Выводы

Выбор инструмента нагрузки

- https://cloud.yandex.ru/docs/load-testing
- https://www.loadview-testing.com
- PhantomJS
- Selenium IDE
- jmeter
- gatling
- Gatling Enterprise
- https://yandextank.readthedocs.io
- hey
- https://locust.io
- k6

Почему gatling

Почему gatling

1. Java DSL

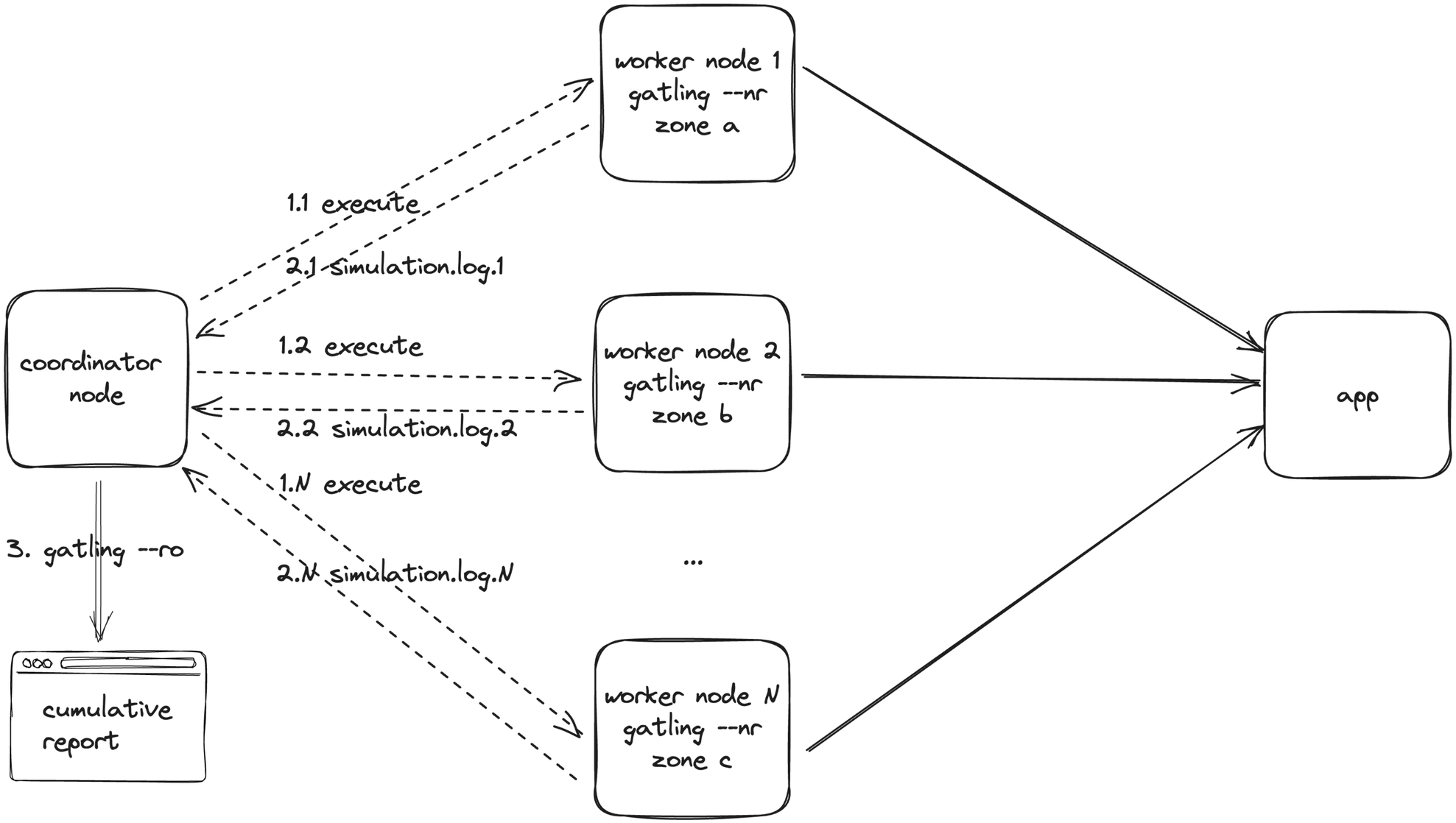
Почему gatling

1. Java DSL
2. Положительный опыт бывших коллег автоматизаторов на scala

Почему gatling

1. Java DSL
2. Положительный опыт бывших коллег автоматизаторов на scala
3. --no-reports, --reports-only

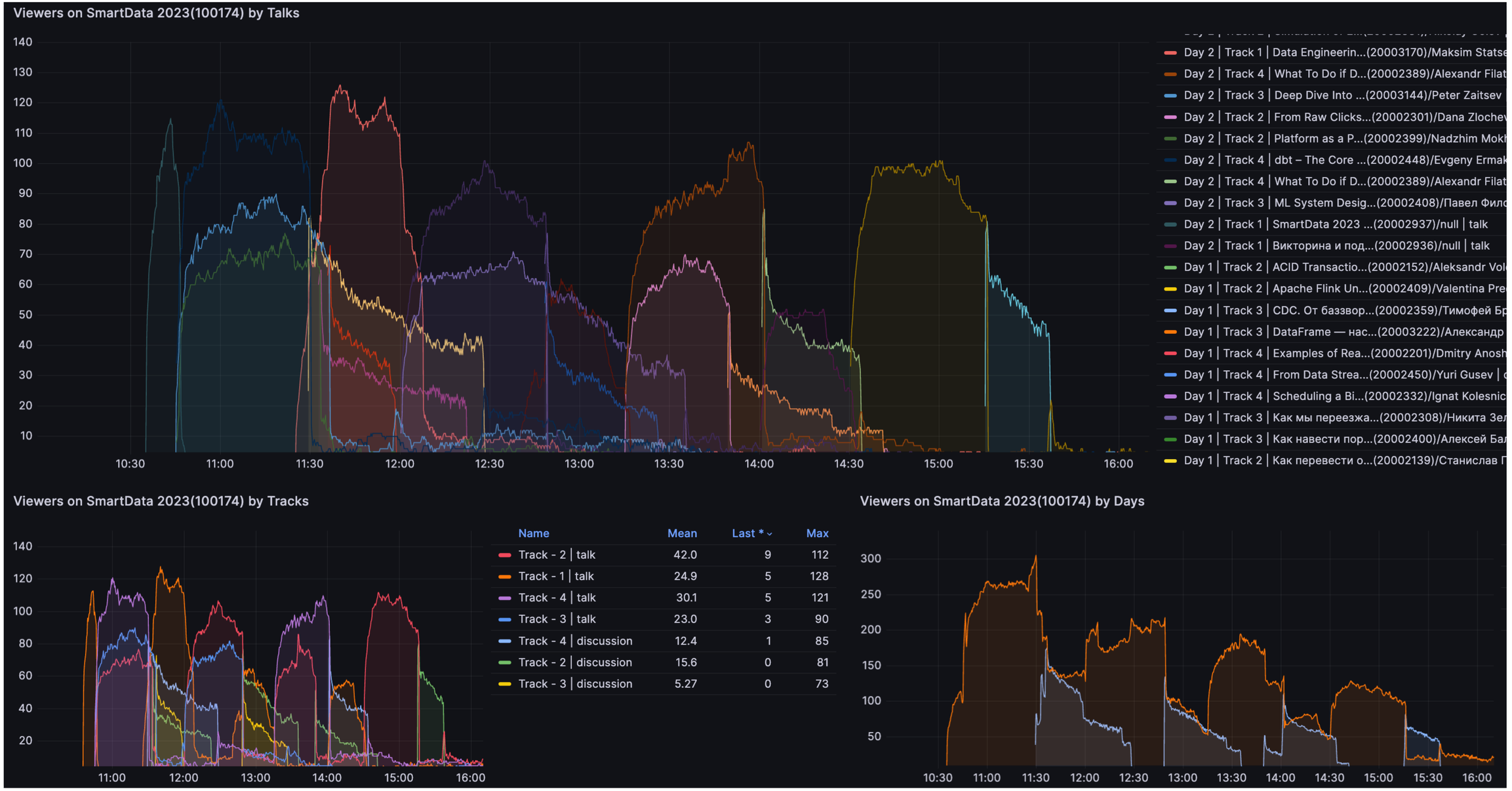
Почему gatling



Глава 4

1. Погружение в контекст онлайна JUG Ru Group, задачи и цели на 2023 год
2. Знакомство с основными компонентами онлайна
3. Выбор инструмента нагрузочного тестирования онлайна (спойлер: gatling)
4. Построение базового нагрузочного сценария, заведение нагрузочных ботов
5. Выбор инструмента деплоя и запуска нагрузки (спойлер: nanocloud + 9 VM)
6. Основные оптимизации для увеличения пропускной способности онлайна
7. Выводы

Профиль нагрузки: imw, now_watching



Профиль и сценарий нагрузки

Профиль и сценарий нагрузки

1. Резкий ramp-up: пользователи быстро набегают

Профиль и сценарий нагрузки

1. Резкий ramp-up: пользователи быстро набегают
2. Сценарий должен быть коротким - до 15 минут

Профиль и сценарий нагрузки

1. Резкий ramp-up: пользователи быстро набегают
2. Сценарий должен быть коротким - до 15 минут
3. Трапецевидный профиль нагрузки:

Профиль и сценарий нагрузки

1. Резкий ramp-up: пользователи быстро набегают
2. Сценарий должен быть коротким - до 15 минут
3. Трапецевидный профиль нагрузки:
   - 5 мин ramp-up до максимума

Профиль и сценарий нагрузки

1. Резкий ramp-up: пользователи быстро набегают
2. Сценарий должен быть коротким - до 15 минут
3. Трапецевидный профиль нагрузки:
   - 5 мин ramp-up до максимума
   - 5 минут плато на максимуме

Профиль и сценарий нагрузки

1. Резкий ramp-up: пользователи быстро набегают
2. Сценарий должен быть коротким - до 15 минут
3. Трапецевидный профиль нагрузки:
   - 5 мин ramp-up до максимума
   - 5 минут плато на максимуме
   - 5 мин спад

Профиль и сценарий нагрузки

1. Резкий ramp-up: пользователи быстро набегают
2. Сценарий должен быть коротким - до 15 минут
3. Трапецевидный профиль нагрузки:
   - 5 мин ramp-up до максимума
   - 5 минут плато на максимуме
   - 5 мин спад
   - 10 минут на пользователя

Базовый сценарий нагрузки

Базовый сценарий нагрузки

1. Расписание и выбор доклада, F5:

Базовый сценарий нагрузки

1. Расписание и выбор доклада, F5:
   - статика сайта

Базовый сценарий нагрузки

1. Расписание и выбор доклада, F5:
   - статика сайта
   - REST API

Базовый сценарий нагрузки

1. Расписание и выбор доклада, F5:
   - статика сайта
   - REST API
   - глобальные подписки по WebSocket/STOMP: изменения расписания, ограничения

Базовый сценарий нагрузки

1. Расписание и выбор доклада, F5:
   - статика сайта
   - REST API
   - глобальные подписки по WebSocket/STOMP: изменения расписания, ограничения

2. Просмотр доклада:

Базовый сценарий нагрузки

1. Расписание и выбор доклада, F5:
   - статика сайта
   - REST API
   - глобальные подписки по WebSocket/STOMP: изменения расписания, ограничения

2. Просмотр доклада:
   - статика страницы

Базовый сценарий нагрузки

1. Расписание и выбор доклада, F5:
   - статика сайта
   - REST API
   - глобальные подписки по WebSocket/STOMP: изменения расписания, ограничения

2. Просмотр доклада:
   - статика страницы
   - REST API

Базовый сценарий нагрузки

1. Расписание и выбор доклада, F5:
   - статика сайта
   - REST API
   - глобальные подписки по WebSocket/STOMP: изменения расписания, ограничения

2. Просмотр доклада:
   - статика страницы
   - REST API
   - локальные подписки по WebSocket/STOMP: изменение доклада, “now_watching”

Базовый сценарий нагрузки

1. Расписание и выбор доклада, F5:
   - статика сайта
   - REST API
   - глобальные подписки по WebSocket/STOMP: изменения расписания, ограничения

2. Просмотр доклада:
   - статика страницы
   - REST API
   - локальные подписки по WebSocket/STOMP: изменение доклада, “now_watching”
   - “i_m_watching” (далее imw) раз в 5 сек

Базовый сценарий нагрузки

1. Расписание и выбор доклада, F5:
   - статика сайта
   - REST API
   - глобальные подписки по WebSocket/STOMP: изменения расписания, ограничения

2. Просмотр доклада:
   - статика страницы
   - REST API
   - локальные подписки по WebSocket/STOMP: изменение доклада, “now_watching”
   - “i_m_watching” (далее imw) раз в 5 сек
   - манифест раз в 3 сек

Базовый сценарий нагрузки

1. Расписание и выбор доклада, F5:
   - статика сайта
   - REST API
   - глобальные подписки по WebSocket/STOMP: изменения расписания, ограничения

2. Просмотр доклада:
   - статика страницы
   - REST API
   - локальные подписки по WebSocket/STOMP: изменение доклада, “now_watching”
   - “i_m_watching” (далее imw) раз в 5 сек
   - манифест раз в 3 сек

3. Редкие прочие действия по REST API: оценка, избранное

Нагрузочные боты

Нагрузочные боты

1. Настоящая тестовая конфа в проде: https://testconf.jugru.team/

Нагрузочные боты

1. Настоящая тестовая конфа в проде: https://testconf.jugru.team/
2. Настоящие корпоративные заказы в Bitrix на 10K

Нагрузочные боты

1. Настоящая тестовая конфа в проде: https://testconf.jugru.team/
2. Настоящие корпоративные заказы в Bitrix на 10K
3. Настоящее распределение билетов из заказов через спец-API

Нагрузочные боты

1. Настоящая тестовая конфа в проде: https://testconf.jugru.team/
2. Настоящие корпоративные заказы в Bitrix на 10K
3. Настоящее распределение билетов из заказов через спец-API
4. Настоящие +10K личных кабинетов вида bot+jvxpqbhramuot5r@jugru.org

Нагрузочные боты

1. Настоящая тестовая конфа в проде: https://testconf.jugru.team/
2. Настоящие корпоративные заказы в Bitrix на 10K
3. Настоящее распределение билетов из заказов через спец-API
4. Настоящие +10K личных кабинетов вида bot+jvxpqbhramuot5r@jugru.org
5. Настоящий бан на 24 часа от Amazon SES

Реализация базового сценария на gatling java dsl

Реализация базового сценария на gatling java dsl: авторизация

Реализация базового сценария на gatling java dsl: авторизация

```
rownum,lk_user_id,userName,userPassword
1,942088,bot+xfuu6cuw092fg8bj@jugru.org,MAfbUHixnuu9pUWl
2,942089,bot+1bsmqugec3xwqbsh@jugru.org,IBiAPNWhN6WAHNRE
3,942090,bot+kdt518ch431iz5q8@jugru.org,QTT3ckEZqf7cnBNi
```

Реализация базового сценария на gatling java dsl: авторизация

```
rownum,lk_user_id,userName,userPassword
1,942088,bot+xfuu6cuw092fg8bj@jugru.org,MAfbUHixnuu9pUWl
2,942089,bot+1bsmqugec3xwqbsh@jugru.org,IBiAPNWhN6WAHNRE
3,942090,bot+kdt518ch431iz5q8@jugru.org,QTT3ckEZqf7cnBNi
```

```
static final public FeederBuilder<String> lkBots =
        CoreDsl.csv("lk-users-" + getTargetEnv() + ".csv")
                .batch(200); // default is 2000, too big
```

Реализация базового сценария на gatling java dsl: авторизация

```
rownum,lk_user_id,userName,userPassword
1,942088,bot+xfuu6cuw092fg8bj@jugru.org,MAfbUHixnuu9pUWl
2,942089,bot+1bsmqugec3xwqbsh@jugru.org,IBiAPNWhN6WAHNRE
3,942090,bot+kdt518ch431iz5q8@jugru.org,QTT3ckEZqf7cnBNi
```

```
static final public FeederBuilder<String> lkBots =
        CoreDsl.csv("lk-users-" + getTargetEnv() + ".csv")
                .batch(200); // default is 2000, too big

scenario("regular user authenticates via LK and watches video")
        .feed(lkBots)
        .exec(
                lkOAuth,
                basicCommonChain
        );
```

Реализация базового сценария на gatling java dsl: авторизация

```
static final public ChainBuilder lkOAuth =
        exec(http("authenticate in lk with password flow")
                .post(getLkOAuthEndpoint())
                .headers(
                        Map.of(
                                "Content-Type", "application/x-www-form-urlencoded",
                                "Accept", "application/json;charset=UTF-8"
                        )
                )
                .formParamMap(
                        Map.of(
                                "grant_type", "password",
                                "client_id", "online-frontend-backend",
                                "client_secret", getOauthOnlineFrontendBackendClientSecret(),
                                "username","#{userName}",
                                "password","#{userPassword}"
                        )
                )
                .check(jmesPath("access_token").saveAs("jwt"))
        );
```

Реализация базового сценария на gatling java dsl: авторизация

```
static final public ChainBuilder lkOAuth =
        exec(http("authenticate in lk with password flow")
                .post(getLkOAuthEndpoint())
                .headers(
                        Map.of(
                                "Content-Type", "application/x-www-form-urlencoded",
                                "Accept", "application/json;charset=UTF-8"
                        )
                )
                .formParamMap(
                        Map.of(
                                "grant_type", "password",
                                "client_id", "online-frontend-backend",
                                "client_secret", getOauthOnlineFrontendBackendClientSecret(),
                                "username","#{userName}",
                                "password","#{userPassword}"
                        )
                )
                .check(jmesPath("access_token").saveAs("jwt"))
        );
```

Реализация базового сценария на gatling java dsl: основной цикл

```
exec(session → session.set("refreshesCount", randomUserRefreshesCount())),
repeat("#{refreshesCount}", "refreshesCounter").on(
        exec(
                spaStaticsLoad, spaRestInteraction,
                wsConnectAndSubscribeToPersonalAndEventChannels,
                group("watching the talk").on(
                        exec(
                                talkPageStaticsLoad,
                                exec(calcRandomVideoQualityAndSaveInSession),
                                talkRestInteraction,
                                talkVideoPagePerSecondInteractionLoop,
                                rareRandomUserRestActionsPerRefresh
                        )
                ),
                wsCloseAndUnsubscribeFromPersonalAndEventChannels
        )
)
```

Реализация базового сценария на gatling java dsl: основной цикл

```
exec(session → session.set("refreshesCount", randomUserRefreshesCount())),
repeat("#{refreshesCount}", "refreshesCounter").on(
        exec(
                spaStaticsLoad, spaRestInteraction,
                wsConnectAndSubscribeToPersonalAndEventChannels,
                group("watching the talk").on(
                        exec(
                                talkPageStaticsLoad,
                                exec(calcRandomVideoQualityAndSaveInSession),
                                talkRestInteraction,
                                talkVideoPagePerSecondInteractionLoop,
                                rareRandomUserRestActionsPerRefresh
                        )
                ),
                wsCloseAndUnsubscribeFromPersonalAndEventChannels
        )
)
```

Реализация базового сценария на gatling java dsl: основной цикл

```
exec(session → session.set("refreshesCount", randomUserRefreshesCount())),
repeat("#{refreshesCount}", "refreshesCounter").on(
        exec(
                spaStaticsLoad, spaRestInteraction,
                wsConnectAndSubscribeToPersonalAndEventChannels,
                group("watching the talk").on(
                        exec(
                                talkPageStaticsLoad,
                                exec(calcRandomVideoQualityAndSaveInSession),
                                talkRestInteraction,
                                talkVideoPagePerSecondInteractionLoop,
                                rareRandomUserRestActionsPerRefresh
                        )
                ),
                wsCloseAndUnsubscribeFromPersonalAndEventChannels
        )
)
```

Реализация базового сценария на gatling java dsl: основной цикл

```
exec(session → session.set("refreshesCount", randomUserRefreshesCount())),
repeat("#{refreshesCount}", "refreshesCounter").on(
        exec(
                spaStaticsLoad, spaRestInteraction,
                wsConnectAndSubscribeToPersonalAndEventChannels,
                group("watching the talk").on(
                        exec(
                                talkPageStaticsLoad,
                                exec(calcRandomVideoQualityAndSaveInSession),
                                talkRestInteraction,
                                talkVideoPagePerSecondInteractionLoop,
                                rareRandomUserRestActionsPerRefresh
                        )
                ),
                wsCloseAndUnsubscribeFromPersonalAndEventChannels
        )
)
```

Реализация базового сценария на gatling java dsl: основной цикл

```
exec(session → session.set("refreshesCount", randomUserRefreshesCount())),
repeat("#{refreshesCount}", "refreshesCounter").on(
        exec(
                spaStaticsLoad, spaRestInteraction,
                wsConnectAndSubscribeToPersonalAndEventChannels,
                group("watching the talk").on(
                        exec(
                                talkPageStaticsLoad,
                                exec(calcRandomVideoQualityAndSaveInSession),
                                talkRestInteraction,
                                talkVideoPagePerSecondInteractionLoop,
                                rareRandomUserRestActionsPerRefresh
                        )
                ),
                wsCloseAndUnsubscribeFromPersonalAndEventChannels
        )
)
```

Реализация базового сценария на gatling java dsl: STOMP

```
public static final ChainBuilder connectToWsEndpoint =
        exec(ws("connect to ws")
                .connect("/ws/v2/#{randomInt(100,1000)}/" +
                        "#{randomAlphanumeric(8)}/websocket?access_token=#{jwt}")
        );
public static final ChainBuilder connectToStompEndpoint =
        exec(ws("connect to stomp")
                        .sendText("[\"CONNECT\\n" +
                                "accept-version:1.0,1.1,1.2\\n" +
                                "heart-beat:0,20000\\n" +
                                "\\n" +
                                "\\u0000\"]")
        );
```

Реализация базового сценария на gatling java dsl: STOMP

```
public static final ChainBuilder connectToWsEndpoint =
        exec(ws("connect to ws")
                .connect("/ws/v2/#{randomInt(100,1000)}/" +
                        "#{randomAlphanumeric(8)}/websocket?access_token=#{jwt}")
        );
public static final ChainBuilder connectToStompEndpoint =
        exec(ws("connect to stomp")
                        .sendText("[\"CONNECT\\n" +
                                "accept-version:1.0,1.1,1.2\\n" +
                                "heart-beat:0,20000\\n" +
                                "\\n" +
                                "\\u0000\"]")

        );
```

Реализация базового сценария на gatling java dsl: STOMP

```
public static final ChainBuilder connectToWsEndpoint =
        exec(ws("connect to ws")
                .connect("/ws/v2/#{randomInt(100,1000)}/" +
                        "#{randomAlphanumeric(8)}/websocket?access_token=#{jwt}")
        );
public static final ChainBuilder connectToStompEndpoint =
        exec(ws("connect to stomp")
                        .sendText("[\"CONNECT\\n" +
                                "accept-version:1.0,1.1,1.2\\n" +
                                "heart-beat:0,20000\\n" +
                                "\\n" +
                                "\\u0000\"]")
);
```

Реализация базового сценария на gatling java dsl: STOMP

```
static public ChainBuilder subscribeToChannel(String name, String subName) {
    return exec(session → session.set(name + "|subInc", next.getAndIncrement()))
            .exec(ws("subscribe to " + chName)
                    .sendText("[\"SUBSCRIBE\\n" +
                            "id:sub-#{" + chName + "|subInc}\\n" +
                            "destination:" + subName + "\\n" +"\\n" +
                            "\\u0000\"]"));
}
static public ChainBuilder sendMessage(String name, String dest, String payload) {
    return exec(ws("send message to " + name)
            .sendText("[\"SEND\\n" +
                    "destination:" + dest + "\\n" + "\\n" +
                    payload +
                    "\\u0000\"]"));
}
```

Реализация базового сценария на gatling java dsl: STOMP

```
static public ChainBuilder subscribeToChannel(String name, String subName) {
    return exec(session → session.set(name + "|subInc", next.getAndIncrement()))
            .exec(ws("subscribe to " + chName)
                    .sendText("[\"SUBSCRIBE\\n" +
                            "id:sub-#{" + chName + "|subInc}\\n" +
                            "destination:" + subName + "\\n" +"\\n" +
                            "\\u0000\"]"));
}
static public ChainBuilder sendMessage(String name, String dest, String payload) {
    return exec(ws("send message to " + name)
            .sendText("[\"SEND\\n" +
                    "destination:" + dest + "\\n" + "\\n" +
                    payload +
                    "\\u0000\"]"));
}
```

Реализация базового сценария на gatling java dsl: STOMP

```
static public ChainBuilder subscribeToChannel(String name, String subName) {
    return exec(session → session.set(name + "|subInc", next.getAndIncrement()))
            .exec(ws("subscribe to " + chName)
                    .sendText("[\"SUBSCRIBE\\n" +
                            "id:sub-#{" + chName + "|subInc}\\n" +
                            "destination:" + subName + "\\n" +"\\n" +
                            "\\u0000\"]"));
}
static public ChainBuilder sendMessage(String name, String dest, String payload) {
    return exec(ws("send message to " + name)
            .sendText("[\"SEND\\n" +
                    "destination:" + dest + "\\n" + "\\n" +
                    payload +
                    "\\u0000\"]"));
}
```

Реализация базового сценария на gatling java dsl: трансляция

```
static public ChainBuilder talkVideoPagePerSecondInteractionLoop =
        group("talk video page per second loop").on(
                exec(
                    subscribeToChannel("activity_now_watching_counter", ...),
                    subscribeToChannel("activity_speech_changed", ...),
                    perSecondInteractionLoop,
                    unsubscribeFromChannel("activity_now_watching_counter"),
                    unsubscribeFromChannel("activity_speech_changed")
                )
        );
```

Реализация базового сценария на gatling java dsl: трансляция

```
static public ChainBuilder talkVideoPagePerSecondInteractionLoop =
        group("talk video page per second loop").on(
                exec(
                    subscribeToChannel("activity_now_watching_counter", ...),
                    subscribeToChannel("activity_speech_changed", ...),
                    perSecondInteractionLoop,
                    unsubscribeFromChannel("activity_now_watching_counter"),
                    unsubscribeFromChannel("activity_speech_changed")
                )
        );
```

Реализация базового сценария на gatling java dsl: трансляция

```
during(session → {
        int refreshesCount = session.getInt("refreshesCount");
        return ofSeconds(toIntExact(divideUnsigned(
           userWatchingTimeInMinutes * 60L, refreshesCount)));
     }, "oneSecTickCounter").on(
   pace(ofSeconds(1))
     .exec(
        doIf(oneSecTickCounterPeriodCompleted(5))
           .then(sendMessage("i_m_watching", "/online/v2/i_m_watching", imw)),
        doIf(oneSecTickCounterPeriodCompleted(3))
           .then(manifest),
        pause(ofMillis(1)) //just to work in pair with outer pace
      )
),
```

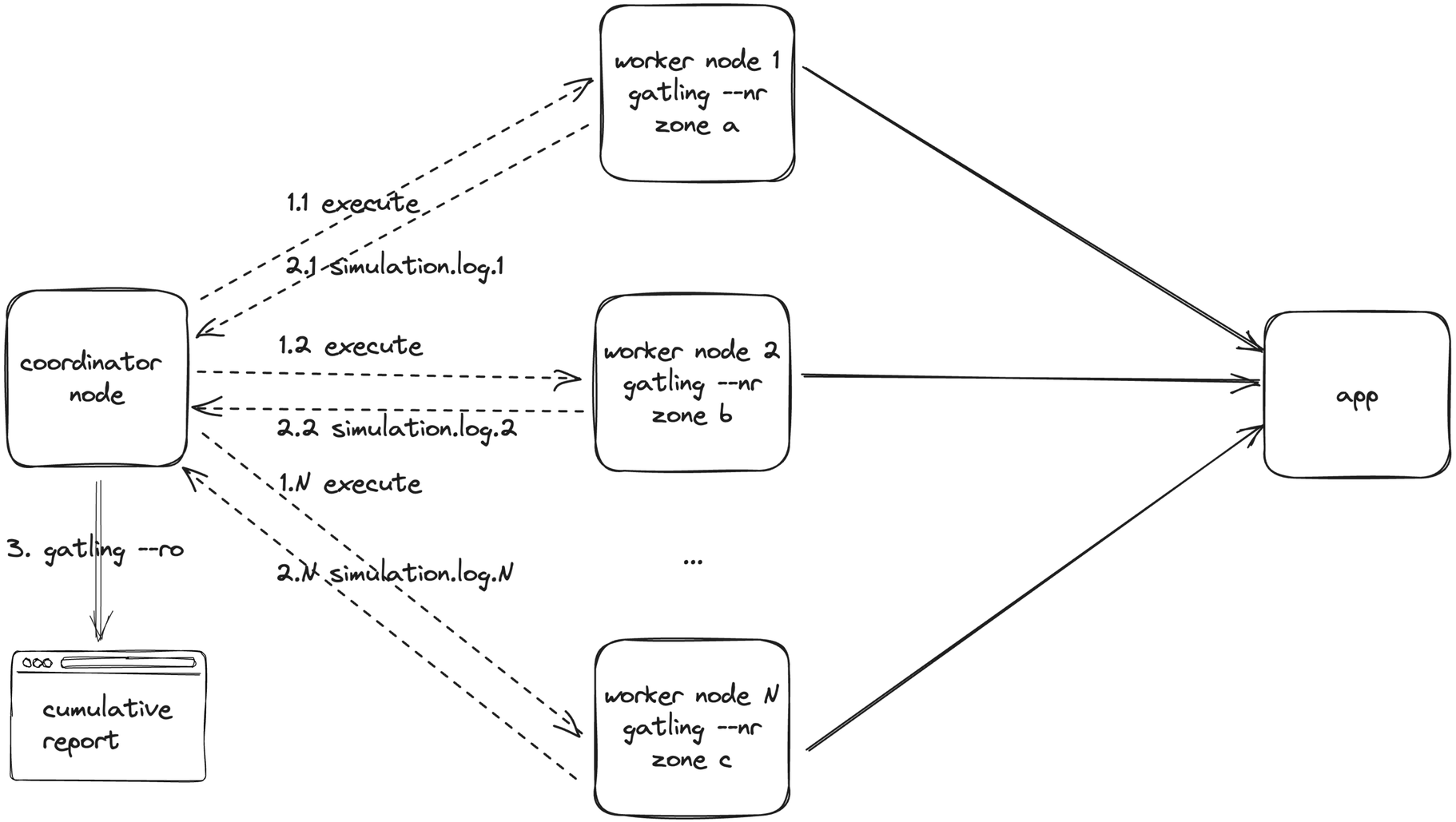
Реализация базового сценария на gatling java dsl: трансляция

```
during(session → {
        int refreshesCount = session.getInt("refreshesCount");
        return ofSeconds(toIntExact(divideUnsigned(
           userWatchingTimeInMinutes * 60L, refreshesCount)));
      }, "oneSecTickCounter").on(
    pace(ofSeconds(1))
      .exec(
        doIf(oneSecTickCounterPeriodCompleted(5))
           .then(sendMessage("i_m_watching", "/online/v2/i_m_watching", imw)),
        doIf(oneSecTickCounterPeriodCompleted(3))
           .then(manifest),
        pause(ofMillis(1)) //just to work in pair with outer pace
       )
),
```

Реализация базового сценария на gatling java dsl: трансляция

```
during(session → {
        int refreshesCount = session.getInt("refreshesCount");
        return ofSeconds(toIntExact(divideUnsigned(
           userWatchingTimeInMinutes * 60L, refreshesCount)));
      }, "oneSecTickCounter").on(
    pace(ofSeconds(1))
      .exec(
        doIf(oneSecTickCounterPeriodCompleted(5))
           .then(sendMessage("i_m_watching", "/online/v2/i_m_watching", imw)),
        doIf(oneSecTickCounterPeriodCompleted(3))
           .then(manifest),
        pause(ofMillis(1)) //just to work in pair with outer pace
       )
),
```

Подготовка базового сценария к распределенному запуску на нескольких рабочих узлах



Подготовка базового сценария к распределенному запуску на нескольких рабочих узлах

```
int loadtoolTotalInstancesNumber =
    parseInt(System.getProperty("loadtool.totalInstancesNumber", "1"));

int loadtoolThisInstanceNumber =
    parseInt(System.getProperty("loadtool.thisInstanceNumber", "0"));
```

Подготовка базового сценария к распределенному запуску на нескольких рабочих узлах

```
static final public io.gatling.javaapi.core.ScenarioBuilder basicLkScenario =
        scenario("regular user authenticates via LK and watch video")
                .feed(lkBots, loadtoolTotalInstancesNumber)
                ...
```

Подготовка базового сценария к распределенному запуску на нескольких рабочих узлах

```
static final public io.gatling.javaapi.core.ScenarioBuilder basicLkScenario =
        scenario("regular user authenticates via LK and watch video")
                .feed(lkBots, loadtoolTotalInstancesNumber)
                ...
```

Подготовка базового сценария к распределенному запуску на нескольких рабочих узлах

```
public static String indexedELProperty(String elProperty) {
    return loadtoolTotalInstancesNumber == 1
            ? "#{" + elProperty + "}"
            : "#{" + elProperty + "(" + loadtoolThisInstanceNumber + ")}";
}
```

Подготовка базового сценария к распределенному запуску на нескольких рабочих узлах

```
exec(http("manifest")
        .get(
          "/api/v2/manifest/quality/index.m3u8" +
          "?eventid=#{eventId}" +
          "&talkid=#{activity_speechId}" +
          "&quality=#{videoQuality}" +
          "&dataprovider=aws" +
          "&auth=#{jwt}" +
          "&deviceid=DevId%s"
             .formatted(indexedELProperty("lk_user_id"))
         )
        .headers(Map.of("Accept-Encoding", "gzip, deflate, br"))
        .check(status().in(200, 304))
);
```

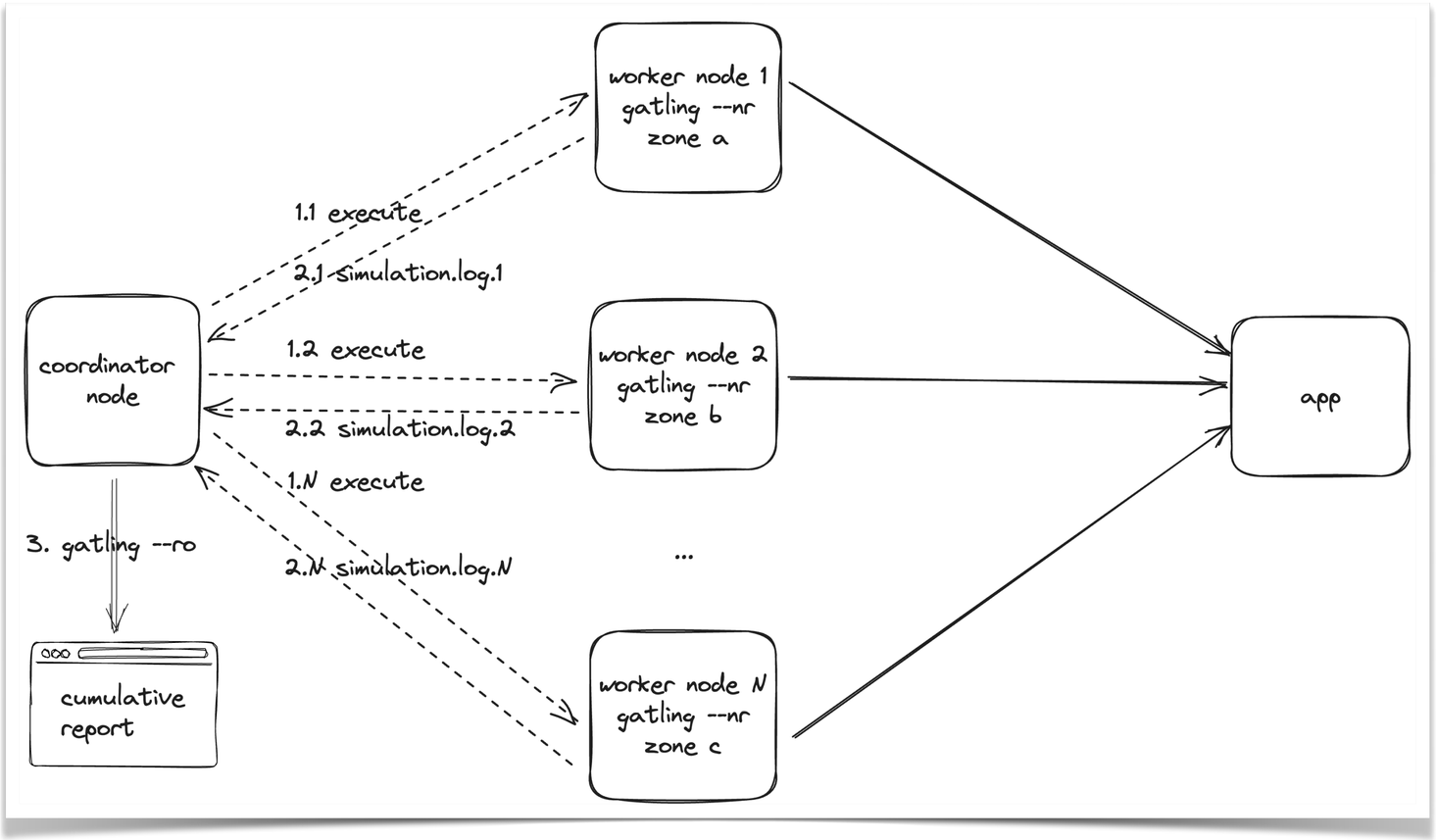
Подготовка базового сценария к распределенному запуску на нескольких рабочих узлах

```
exec(http("manifest")
        .get(
          "/api/v2/manifest/quality/index.m3u8" +
          "?eventid=#{eventId}" +
          "&talkid=#{activity_speechId}" +
          "&quality=#{videoQuality}" +
          "&dataprovider=aws" +
          "&auth=#{jwt}" +
          "&deviceid=DevId%s"
             .formatted(indexedELProperty("lk_user_id"))
         )
        .headers(Map.of("Accept-Encoding", "gzip, deflate, br"))
        .check(status().in(200, 304))
);
```

Глава 5

1. Погружение в контекст онлайна JUG Ru Group, задачи и цели на 2023 год
2. Знакомство с основными компонентами онлайна
3. Выбор инструмента нагрузочного тестирования онлайна (спойлер: gatling)
4. Построение базового нагрузочного сценария, заведение нагрузочных ботов
5. Выбор инструмента деплоя и запуска нагрузки (спойлер: nanocloud + 9 VM)
6. Основные оптимизации для увеличения пропускной способности онлайна
7. Выводы

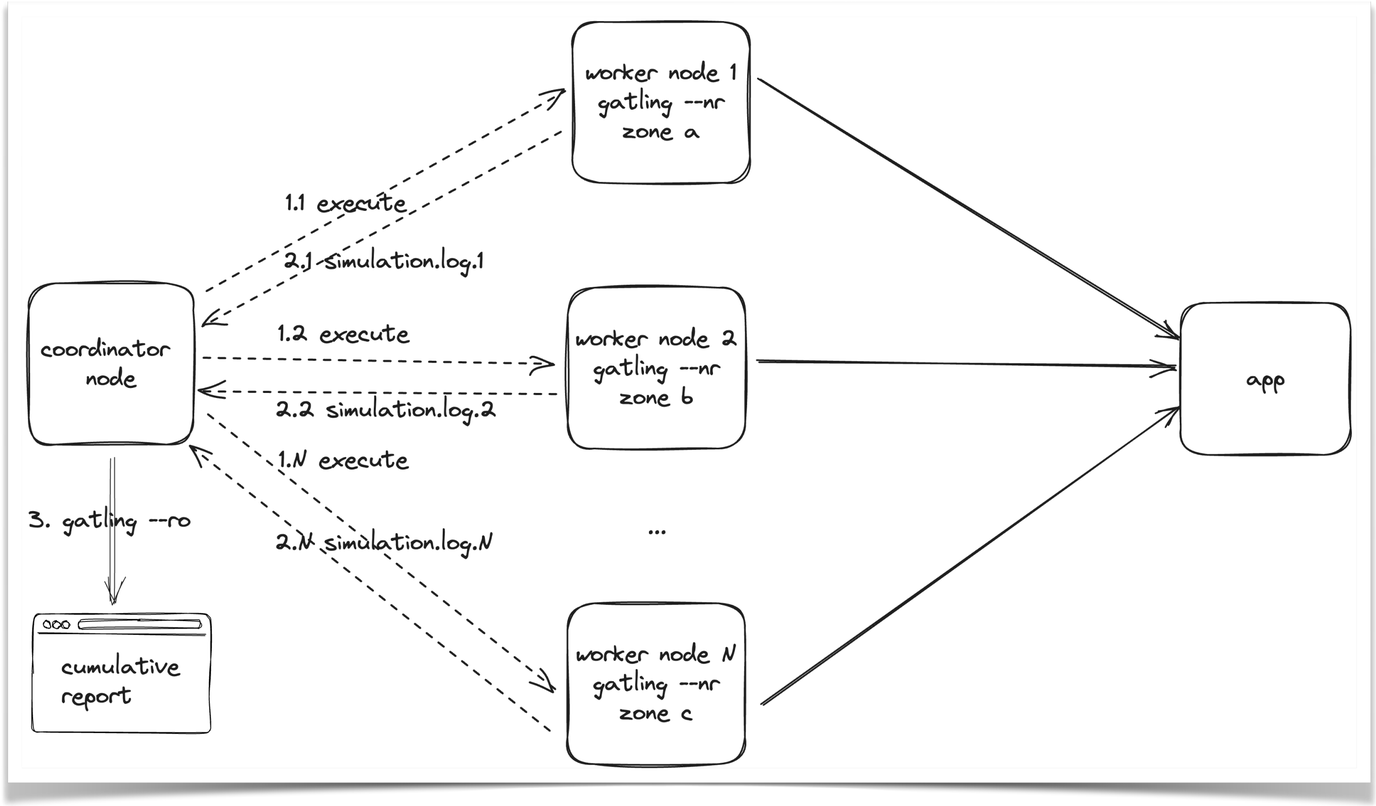
Запуск распределенной нагрузки в k8s?



1. dev -> prod, prod -> dev
2. ReplicaSet? Jobs? DaemonSet?

Nanocloud

Алексей Рагозин



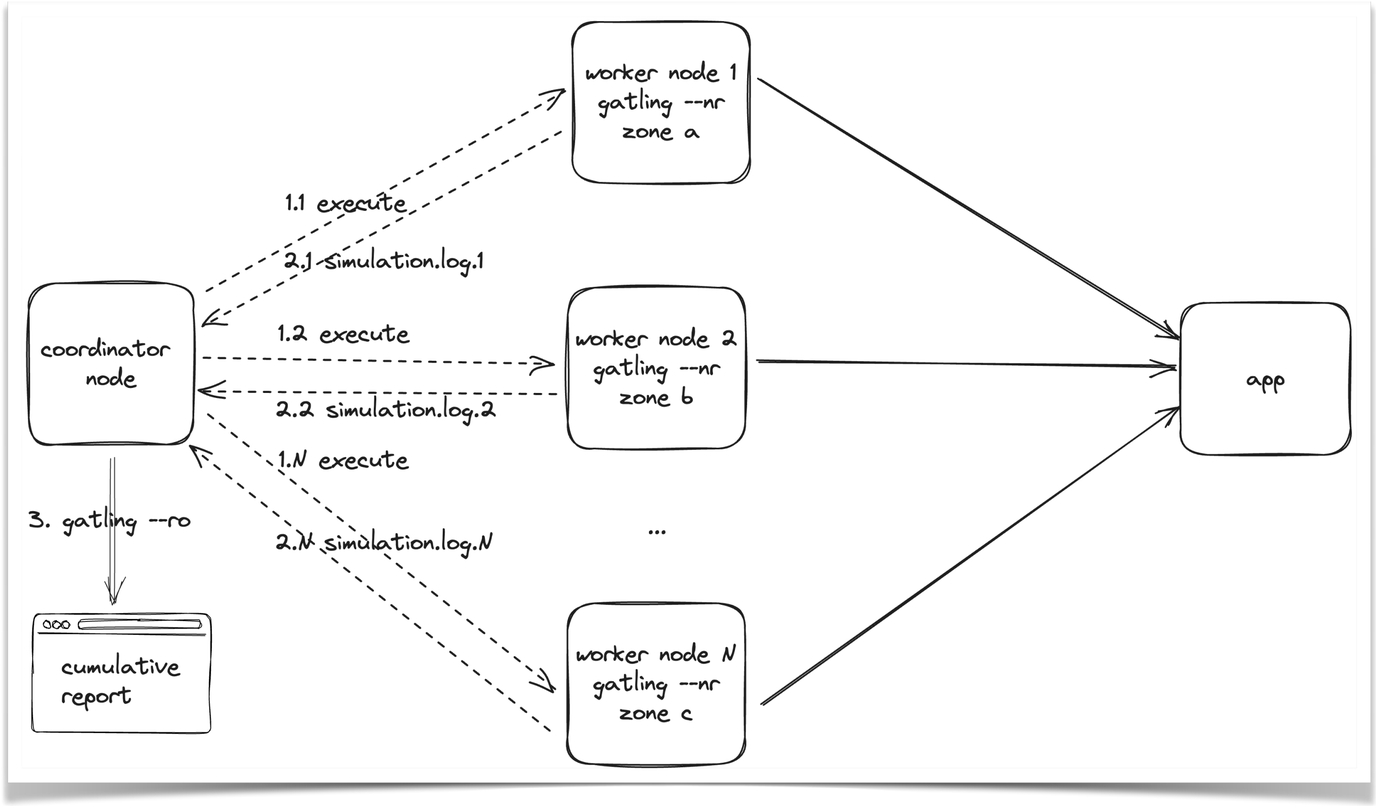
1) worker VMs

2) ssh access

3) JDK installed

Nanocloud

Алексей Рагозин



1) worker VMs

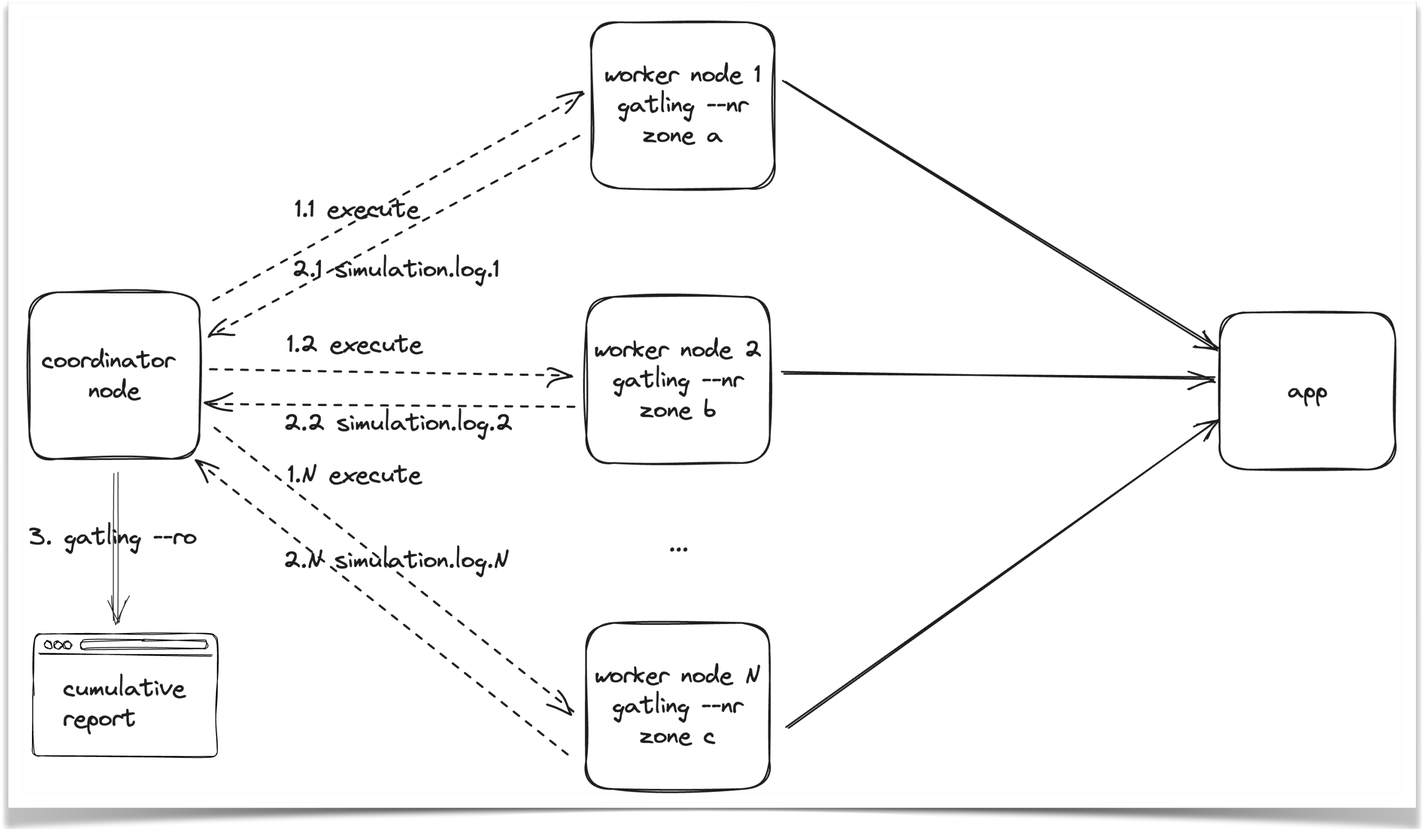
2) ssh access

3) JDK installed

https://blog.ragozin.info/2013/01/remote-code-execution-in-java-made.html
https://github.com/gridkit/nanocloud/blob/vicluster-0.8/docs/NanoCloud_Tutorial.md

Запуск распределенной нагрузки в nanocloud

1. 9 виртуалок в облаке: vCPU=10, RAM=20 ГБ, 50 ГБ, ~18р/ч каждая
2. 9 публичных статических ip
3. JDK 17
4. ssh key
5. Координатор: локальная тачка



Конфигурация и запуск nanocloud на рабочих узлах

```
Cloud cloud = CloudFactory.createCloud();
IntStream.range(0, loadtoolTotalInstancesNumber).forEach(i → cloud.node("worker" + i));

if (Boolean.parseBoolean(loadtoolDryRun)) {
    cloud.node("**").x(VX.TYPE).setLocal();
} else {
    IntStream.range(0, loadtoolTotalInstancesNumber).forEach(i →
            RemoteNode.at(cloud.node("worker" + i))
                    .setRemoteNodeType()
                    .setRemoteHost(cloudHosts[i].trim())
                    .setRemoteAccount(cloudUsers[i].trim())
                    .setSshPrivateKey(cloudUsersPrivateKeysLocations[i].trim())
    );
}
Map<String, String> commonEnvPropsProps = prepareCommonEnvProperties();
ViConf.JvmConf jvmConf = cloud.node("**").x(VX.PROCESS);
jvmConf.setProps(commonEnvPropsProps);
```

Конфигурация и запуск nanocloud на рабочих узлах

```
Cloud cloud = CloudFactory.createCloud();
IntStream.range(0, loadtoolTotalInstancesNumber).forEach(i → cloud.node("worker" + i));
```

```
if (Boolean.parseBoolean(loadtoolDryRun)) {
    cloud.node("**").x(VX.TYPE).setLocal();
} else {
    IntStream.range(0, loadtoolTotalInstancesNumber).forEach(i →
            RemoteNode.at(cloud.node("worker" + i))
                    .setRemoteNodeType()
                    .setRemoteHost(cloudHosts[i].trim())
                    .setRemoteAccount(cloudUsers[i].trim())
                    .setSshPrivateKey(cloudUsersPrivateKeysLocations[i].trim())
    );
}
Map<String, String> commonEnvPropsProps = prepareCommonEnvProperties();
ViConf.JvmConf jvmConf = cloud.node("**").x(VX.PROCESS);
jvmConf.setProps(commonEnvPropsProps);
```

Конфигурация и запуск nanocloud на рабочих узлах

```
Cloud cloud = CloudFactory.createCloud();
IntStream.range(0, loadtoolTotalInstancesNumber).forEach(i → cloud.node("worker" + i));
```

```
if (Boolean.parseBoolean(loadtoolDryRun)) {
    cloud.node("**").x(VX.TYPE).setLocal();
} else {
    IntStream.range(0, loadtoolTotalInstancesNumber).forEach(i →
            RemoteNode.at(cloud.node("worker" + i))
                    .setRemoteNodeType()
                    .setRemoteHost(cloudHosts[i].trim())
                    .setRemoteAccount(cloudUsers[i].trim())
                    .setSshPrivateKey(cloudUsersPrivateKeysLocations[i].trim())
    );
}
Map<String, String> commonEnvPropsProps = prepareCommonEnvProperties();
ViConf.JvmConf jvmConf = cloud.node("**").x(VX.PROCESS);
jvmConf.setProps(commonEnvPropsProps);
```

Конфигурация и запуск nanocloud на рабочих узлах

```
Cloud cloud = CloudFactory.createCloud();
IntStream.range(0, loadtoolTotalInstancesNumber).forEach(i → cloud.node("worker" + i));

if (Boolean.parseBoolean(loadtoolDryRun)) {
    cloud.node("**").x(VX.TYPE).setLocal();
} else {
    IntStream.range(0, loadtoolTotalInstancesNumber).forEach(i →
            RemoteNode.at(cloud.node("worker" + i))
                    .setRemoteNodeType()
                    .setRemoteHost(cloudHosts[i].trim())
                    .setRemoteAccount(cloudUsers[i].trim())
                    .setSshPrivateKey(cloudUsersPrivateKeysLocations[i].trim())
    );
}
Map<String, String> commonEnvPropsProps = prepareCommonEnvProperties();
ViConf.JvmConf jvmConf = cloud.node("**").x(VX.PROCESS);
jvmConf.setProps(commonEnvPropsProps);
```

Конфигурация и запуск nanocloud на рабочих узлах

```
Cloud cloud = CloudFactory.createCloud();
IntStream.range(0, loadtoolTotalInstancesNumber).forEach(i → cloud.node("worker" + i));
```

```
if (Boolean.parseBoolean(loadtoolDryRun)) {
    cloud.node("**").x(VX.TYPE).setLocal();
} else {
    IntStream.range(0, loadtoolTotalInstancesNumber).forEach(i →
            RemoteNode.at(cloud.node("worker" + i))
                    .setRemoteNodeType()
                    .setRemoteHost(cloudHosts[i].trim())
                    .setRemoteAccount(cloudUsers[i].trim())
                    .setSshPrivateKey(cloudUsersPrivateKeysLocations[i].trim())
    );
}
```

```
Map<String, String> commonEnvPropsProps = prepareCommonEnvProperties();
ViConf.JvmConf jvmConf = cloud.node("**").x(VX.PROCESS);
jvmConf.setProps(commonEnvPropsProps);
```

Конфигурация и запуск nanocloud на рабочих узлах

```
String nowAsIsoString =
      DateFormatUtils.format(Date.from(Instant.now()), "yyyy-MM-dd-HH-mm-ss");

IntStream.range(0, loadtoolTotalInstancesNumber).forEach(i → {
  Path resFldr = Path.of("results", nowAsIsoString + "-worker" + i + "-result");
  cloud.node("worker" + i).setProp("loadtool.thisInstanceNumber", String.valueOf(i));
  cloud.node("worker" + i).setProp("gatling.resultsDirectory", resFldr.toString());
});

cloud.node("**").touch();
Thread.sleep(1000);

List<Map<String, Object>> similationLogGzipList = cloud
        .node("**")
        .massExec(new CloudLaunchTask());
```

Конфигурация и запуск nanocloud на рабочих узлах

```
String nowAsIsoString =
      DateFormatUtils.format(Date.from(Instant.now()), "yyyy-MM-dd-HH-mm-ss");

IntStream.range(0, loadtoolTotalInstancesNumber).forEach(i → {
  Path resFldr = Path.of("results", nowAsIsoString + "-worker" + i + "-result");
  cloud.node("worker" + i).setProp("loadtool.thisInstanceNumber", String.valueOf(i));
  cloud.node("worker" + i).setProp("gatling.resultsDirectory", resFldr.toString());
});
```

```
cloud.node("**").touch();
Thread.sleep(1000);

List<Map<String, Object>> similationLogGzipList = cloud
        .node("**")
        .massExec(new CloudLaunchTask());
```

Конфигурация и запуск nanocloud на рабочих узлах

```
String nowAsIsoString =
      DateFormatUtils.format(Date.from(Instant.now()), "yyyy-MM-dd-HH-mm-ss");

IntStream.range(0, loadtoolTotalInstancesNumber).forEach(i → {
  Path resFldr = Path.of("results", nowAsIsoString + "-worker" + i + "-result");
  cloud.node("worker" + i).setProp("loadtool.thisInstanceNumber", String.valueOf(i));
  cloud.node("worker" + i).setProp("gatling.resultsDirectory", resFldr.toString());
});
```

```
cloud.node("**").touch();
Thread.sleep(1000);
```

```
List<Map<String, Object>> similationLogGzipList = cloud
        .node("**")
        .massExec(new CloudLaunchTask());
```

Конфигурация и запуск nanocloud на рабочих узлах

```
String nowAsIsoString =
      DateFormatUtils.format(Date.from(Instant.now()), "yyyy-MM-dd-HH-mm-ss");

IntStream.range(0, loadtoolTotalInstancesNumber).forEach(i → {
  Path resFldr = Path.of("results", nowAsIsoString + "-worker" + i + "-result");
  cloud.node("worker" + i).setProp("loadtool.thisInstanceNumber", String.valueOf(i));
  cloud.node("worker" + i).setProp("gatling.resultsDirectory", resFldr.toString());
});
```

```
cloud.node("**").touch();
Thread.sleep(1000);
```

```
List<Map<String, Object>> similationLogGzipList = cloud
        .node("**")
        .massExec(new CloudLaunchTask());
```

Запуск gatling на рабочем узле

```
@Slf4j
public class CloudLaunchTask implements Callable<Map<String, Object>>, Serializable {
  @Override
  public Map<String, Object> call() throws Exception {
      String jvmName = ManagementFactory.getRuntimeMXBean().getName();

      launchGatling();                                                    //1

      Path simulationLogPathGzip = locateAndGzipSimulationLog();          //2

      byte[] simulationLogGzipBytes = Files.readAllBytes(simulationLogPathGzip);//3

      return Map.of(                                                      //4
          "jvmName", jvmName,
          "simulationLogGzipBytes", simulationLogGzipBytes
      );
    }
...
```

Запуск gatling на рабочем узле

```
@Slf4j
public class CloudLaunchTask implements Callable<Map<String, Object>>, Serializable {
  @Override
  public Map<String, Object> call() throws Exception {
      String jvmName = ManagementFactory.getRuntimeMXBean().getName();

      launchGatling();                                                    //1

      Path simulationLogPathGzip = locateAndGzipSimulationLog();          //2

      byte[] simulationLogGzipBytes = Files.readAllBytes(simulationLogPathGzip);//3

      return Map.of(                                                      //4
          "jvmName", jvmName,
          "simulationLogGzipBytes", simulationLogGzipBytes
      );
    }
...
```

Запуск gatling на рабочем узле

```
public static void launchGatling() {
    GatlingPropertiesBuilder props = new GatlingPropertiesBuilder()
            .simulationClass(gatlingSimulationClass)
            .runDescription(gatlingRunDescription)
            .noReports()
            .resultsDirectory(gatlingResultsDirectory);
   Gatling.fromMap(props.build());
}
```

Запуск gatling на рабочем узле

```
public static void launchGatling() {
    GatlingPropertiesBuilder props = new GatlingPropertiesBuilder()
            .simulationClass(gatlingSimulationClass)
            .runDescription(gatlingRunDescription)
            .noReports()
            .resultsDirectory(gatlingResultsDirectory);
   Gatling.fromMap(props.build());
}
```

Построение кумулятивного отчета

```
List<Map<String, Object>> simLogGzipList = cloud
        .node("**")
        .massExec(new CloudLaunchTask());

ungzipRawSimulationsLogsFromAllWorkers(simLogGzipList, execResultsDir);//1

Gatling.fromMap(new GatlingPropertiesBuilder()                          //2
        .reportsOnly(execResultsDir)
        .build());

cloud.shutdown();                                                       //3
```

Построение кумулятивного отчета

```
List<Map<String, Object>> simLogGzipList = cloud
        .node("**")
        .massExec(new CloudLaunchTask());

ungzipRawSimulationsLogsFromAllWorkers(simLogGzipList, execResultsDir);//1

Gatling.fromMap(new GatlingPropertiesBuilder()                          //2
        .reportsOnly(execResultsDir)
        .build());

cloud.shutdown();                                                       //3
```

Построение кумулятивного отчета

```
List<Map<String, Object>> simLogGzipList = cloud
        .node("**")
        .massExec(new CloudLaunchTask());

ungzipRawSimulationsLogsFromAllWorkers(simLogGzipList, execResultsDir);//1

Gatling.fromMap(new GatlingPropertiesBuilder()                          //2
        .reportsOnly(execResultsDir)
        .build());

cloud.shutdown();                                                       //3
```

Результаты первых запусков: 700

1. 30 марта 2023 первые запуски в деве: ресурсы в деве вообще не тянут, смысла продолжать тестить в деве нет

2. 31 марта 2023 первые запуски в проде: более менее тянем, ошибок мало, расхождения количества вызовов imw и походов за манифестами с расчетными величинами менее 5%

Результаты первых запусков: 1K

Результаты первых запусков: 1K

1. Нагрузочные пользователи не доходят до обслуживания, не могут авторизоваться в ЛК, не могут подключиться к WS

Результаты первых запусков: 1K

1. Нагрузочные пользователи не доходят до обслуживания, не могут авторизоваться в ЛК, не могут подключиться к WS

2. Время отклика Mjollnir вылетает за 4 сек

Результаты первых запусков: 1K

1. Нагрузочные пользователи не доходят до обслуживания, не могут авторизоваться в ЛК, не могут подключиться к WS

2. Время отклика Mjollnir вылетает за 4 сек

3. stream-playlist умирает, манифесты не отдаются в SLA 3 сек

Результаты первых запусков: 1K

1. Нагрузочные пользователи не доходят до обслуживания, не могут авторизоваться в ЛК, не могут подключиться к WS

2. Время отклика Mjollnir вылетает за 4 сек

3. stream-playlist умирает, манифесты не отдаются в SLA 3 сек

4. Сервисы не справляются с пробками и k8s их вечно рестартует

Результаты первых запусков: 1K

1. Нагрузочные пользователи не доходят до обслуживания, не могут авторизоваться в ЛК, не могут подключиться к WS

2. Время отклика Mjollnir вылетает за 4 сек

3. stream-playlist умирает, манифесты не отдаются в SLA 3 сек

4. Сервисы не справляются с пробками и k8s их вечно рестартует

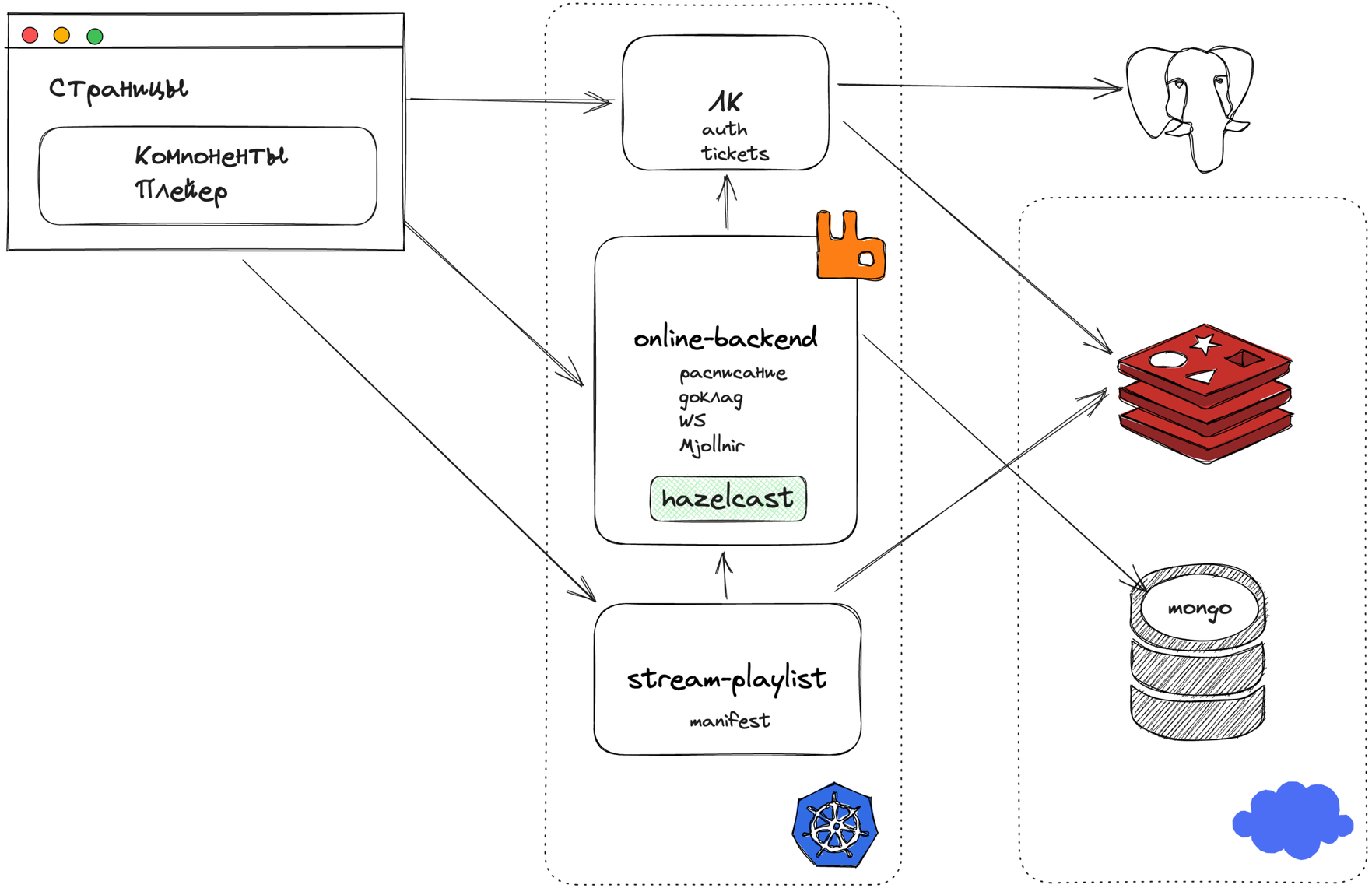
5. Ни о какой “не лагающей” трансляции речи быть не может

Глава 6

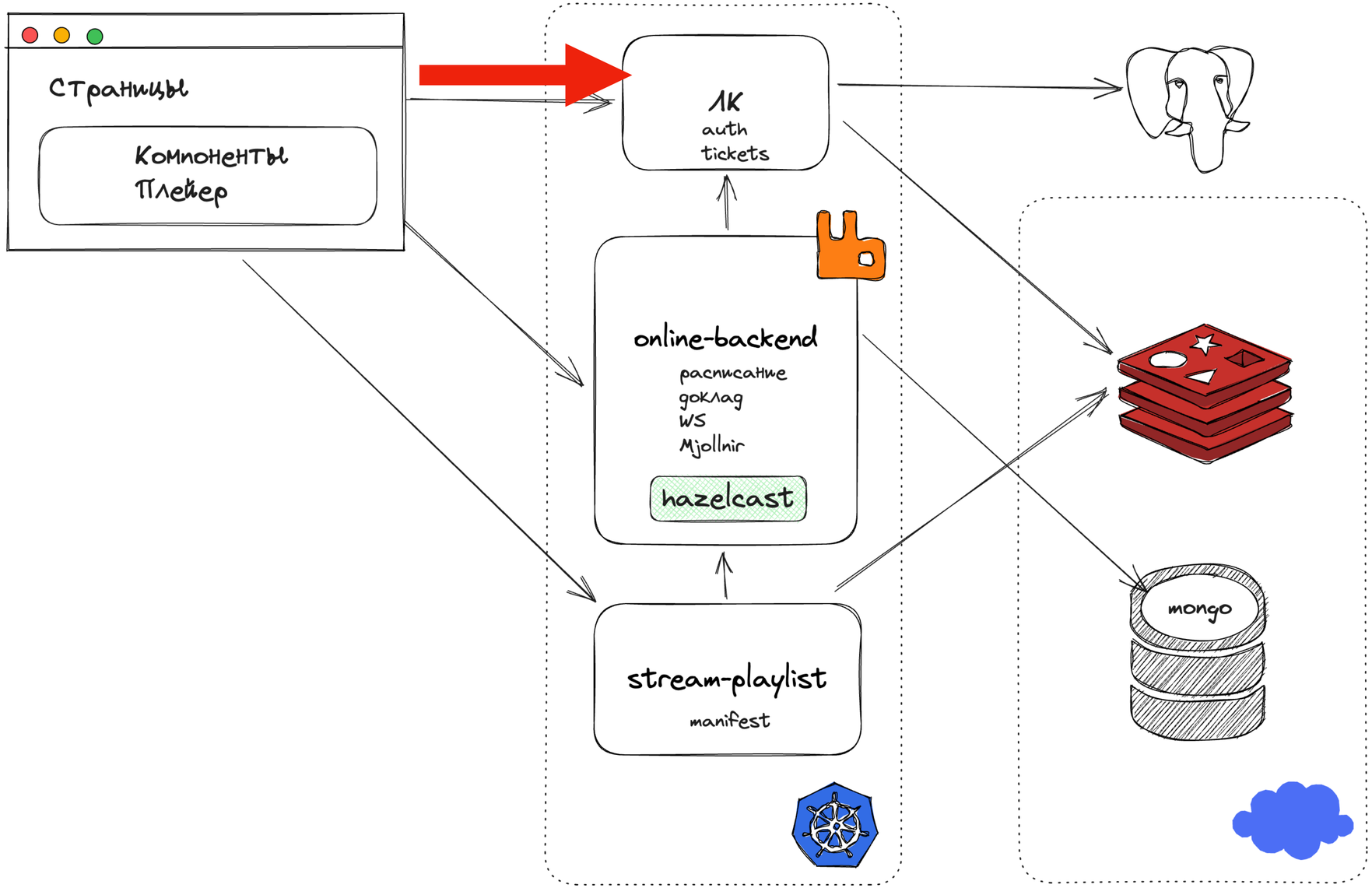
1. Погружение в контекст онлайна JUG Ru Group, задачи и цели на 2023 год
2. Знакомство с основными компонентами онлайна
3. Выбор инструмента нагрузочного тестирования онлайна (спойлер: gatling)
4. Построение базового нагрузочного сценария, заведение нагрузочных ботов
5. Выбор инструмента деплоя и запуска нагрузки (спойлер: nanocloud + 9 VM)
6. Основные оптимизации для увеличения пропускной способности онлайна
7. Выводы

Оптимизации throughput: tomcat thread pool

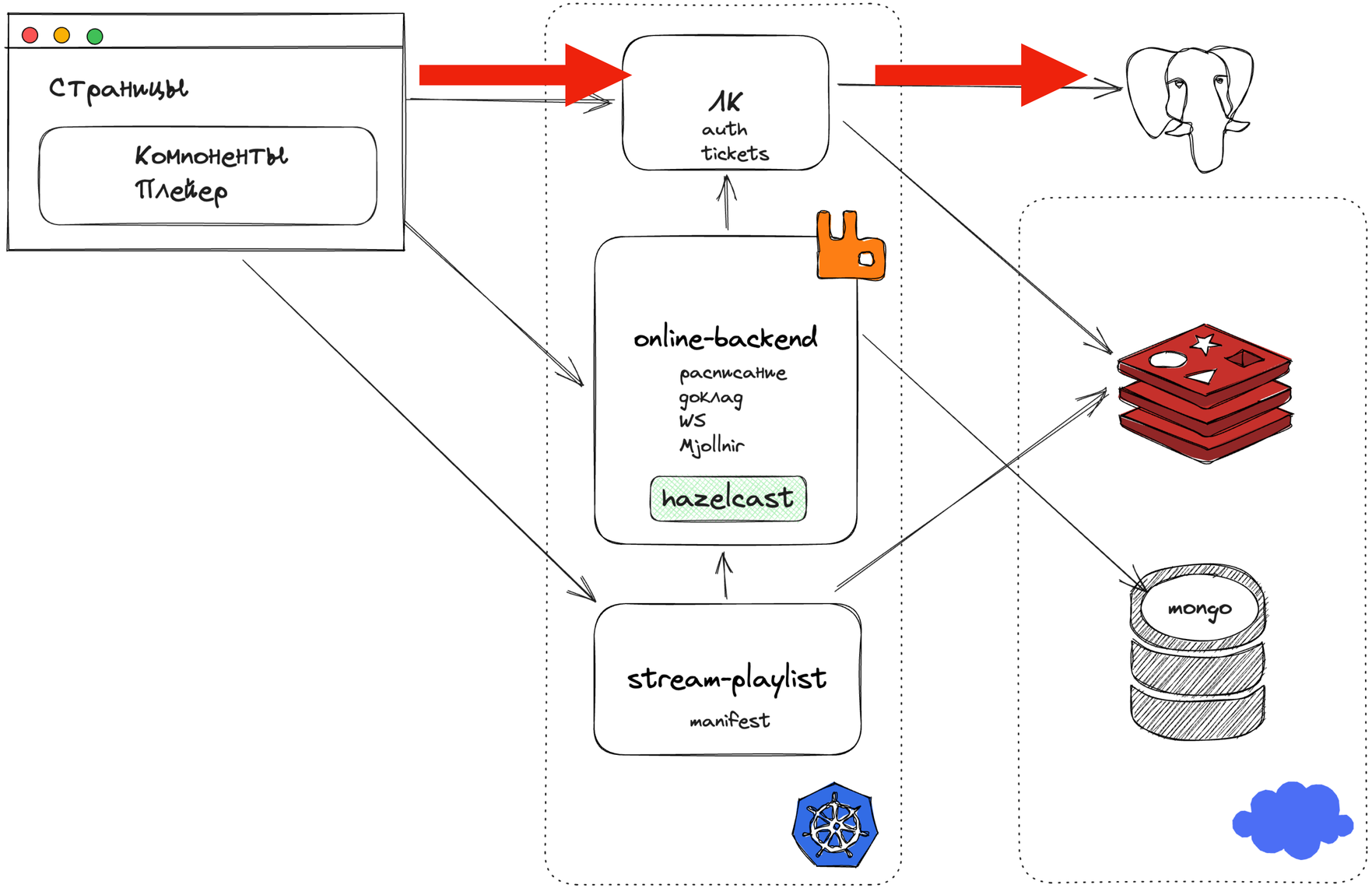
Оптимизации throughput: tomcat thread pool



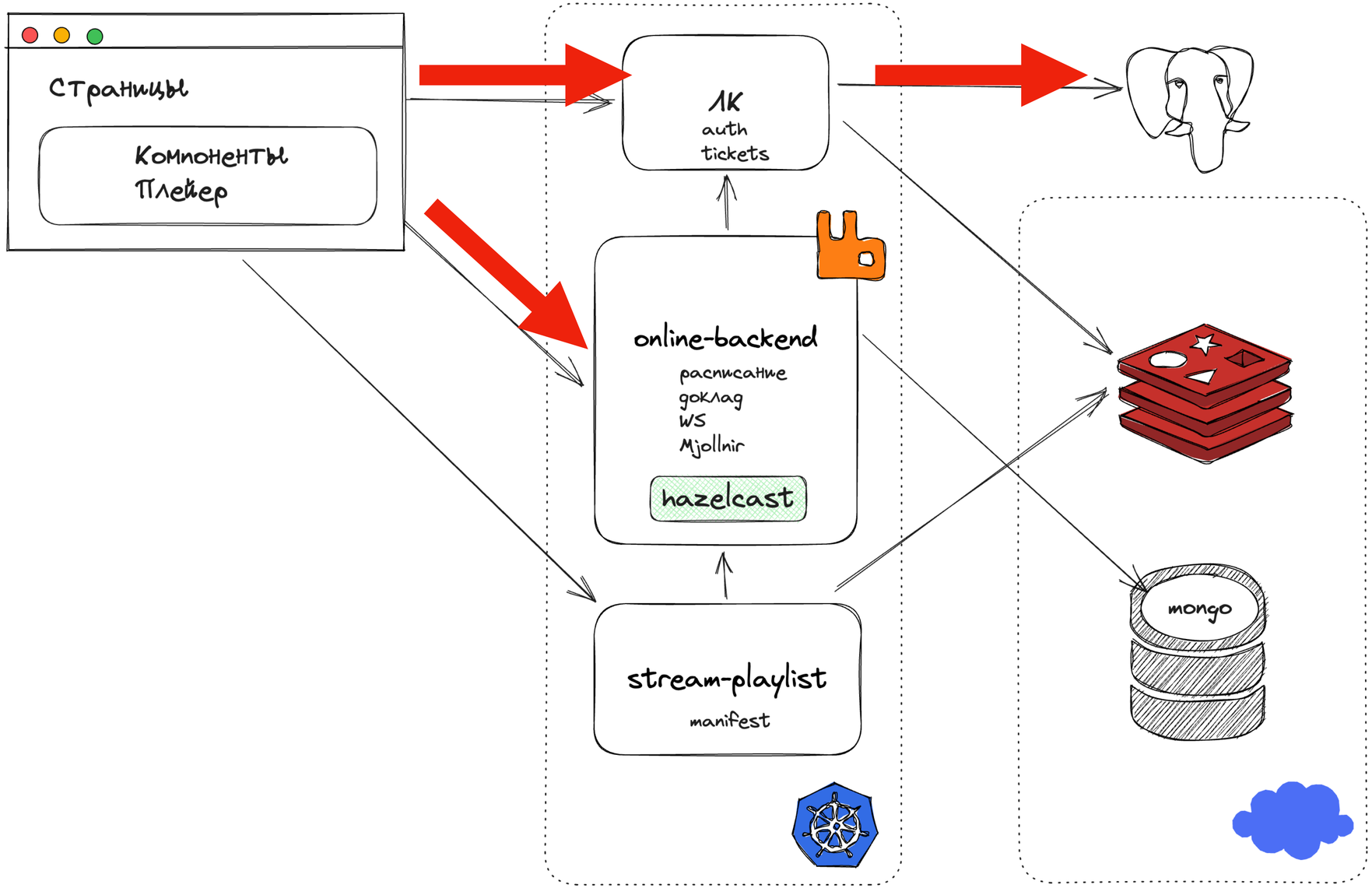
Оптимизации throughput: tomcat thread pool



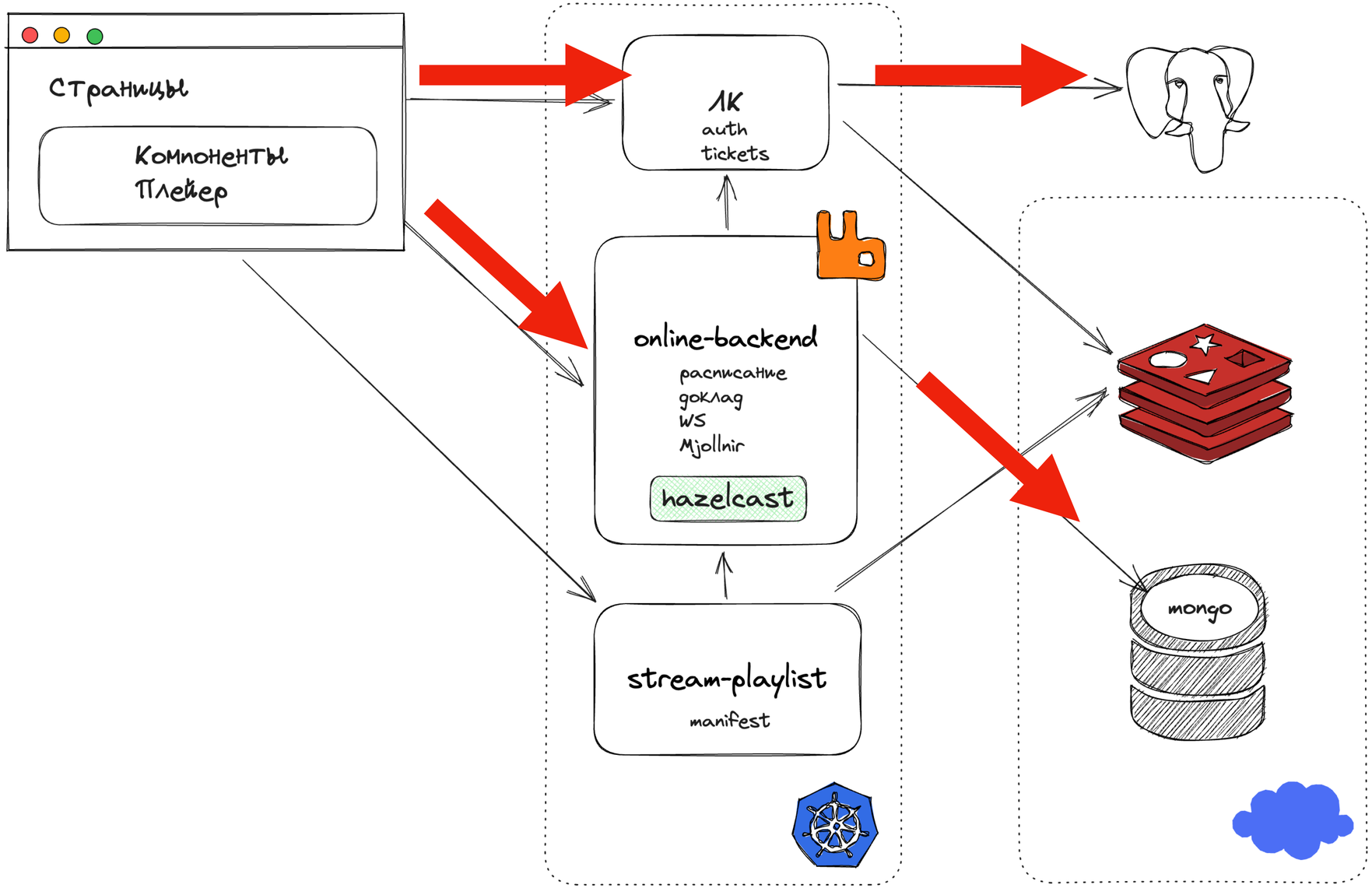
Оптимизации throughput: tomcat thread pool



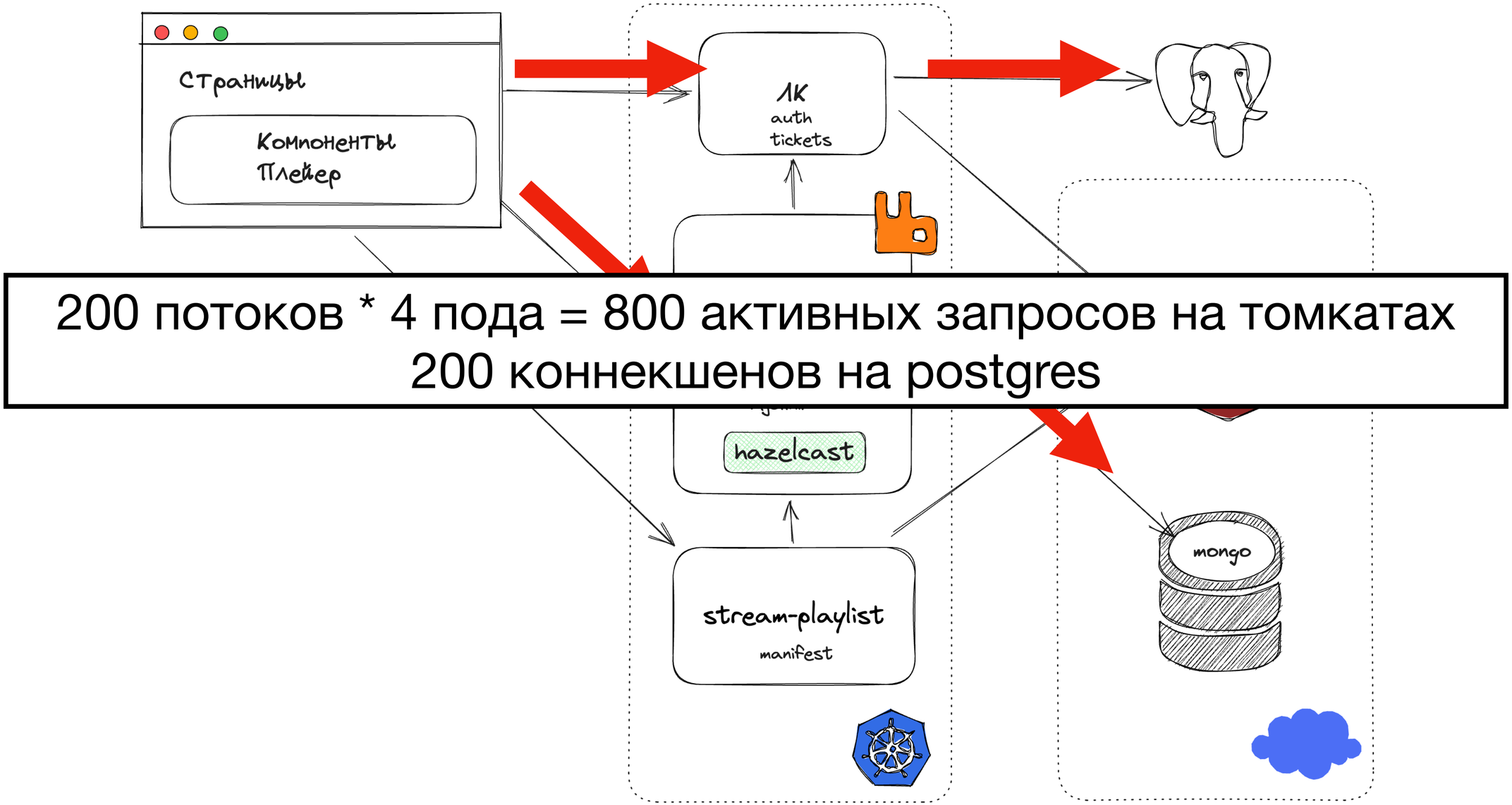
Оптимизации throughput: tomcat thread pool



Оптимизации throughput: tomcat thread pool

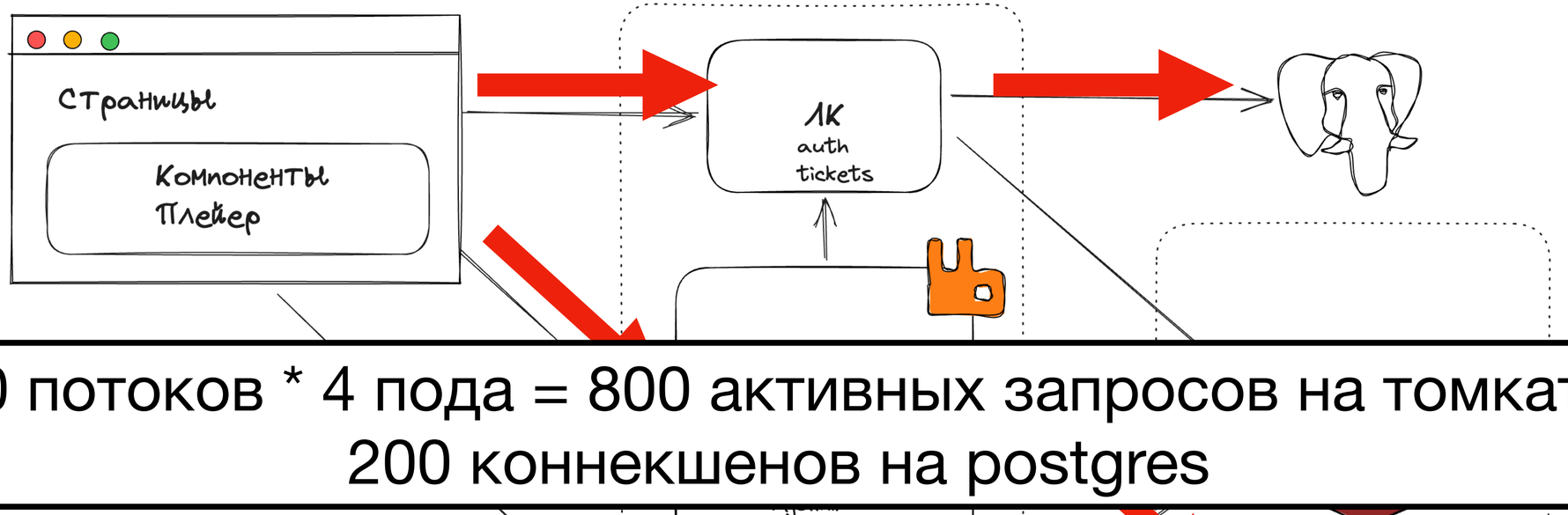


Оптимизации throughput: tomcat thread pool



200 потоков * 4 пода = 800 активных запросов на томкатах
200 коннекшенов на postgres

Оптимизации throughput: tomcat thread pool



200 потоков * 4 пода = 800 активных запросов на томкатах
200 коннекшенов на postgres

Оптимизации throughput: tomcat thread pool

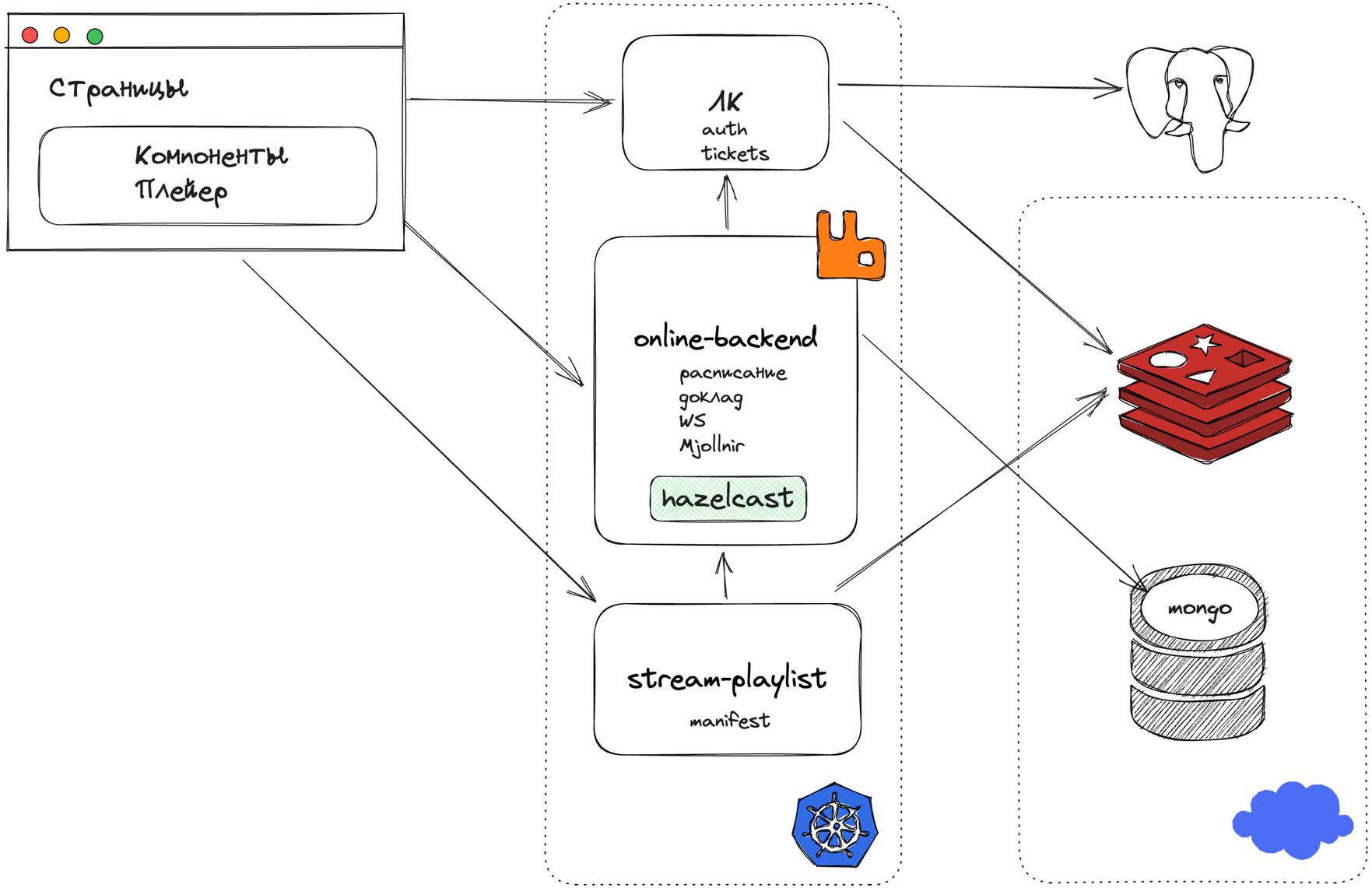
```
server.tomcat.max-threads=75 (умолчание 200)
server.tomcat.min-spare-threads=25
server.tomcat.accept-count=3000 (умолчание 100)
server.tomcat.mbeanregistry.enabled=true
```

Оптимизации throughput: tomcat thread pool

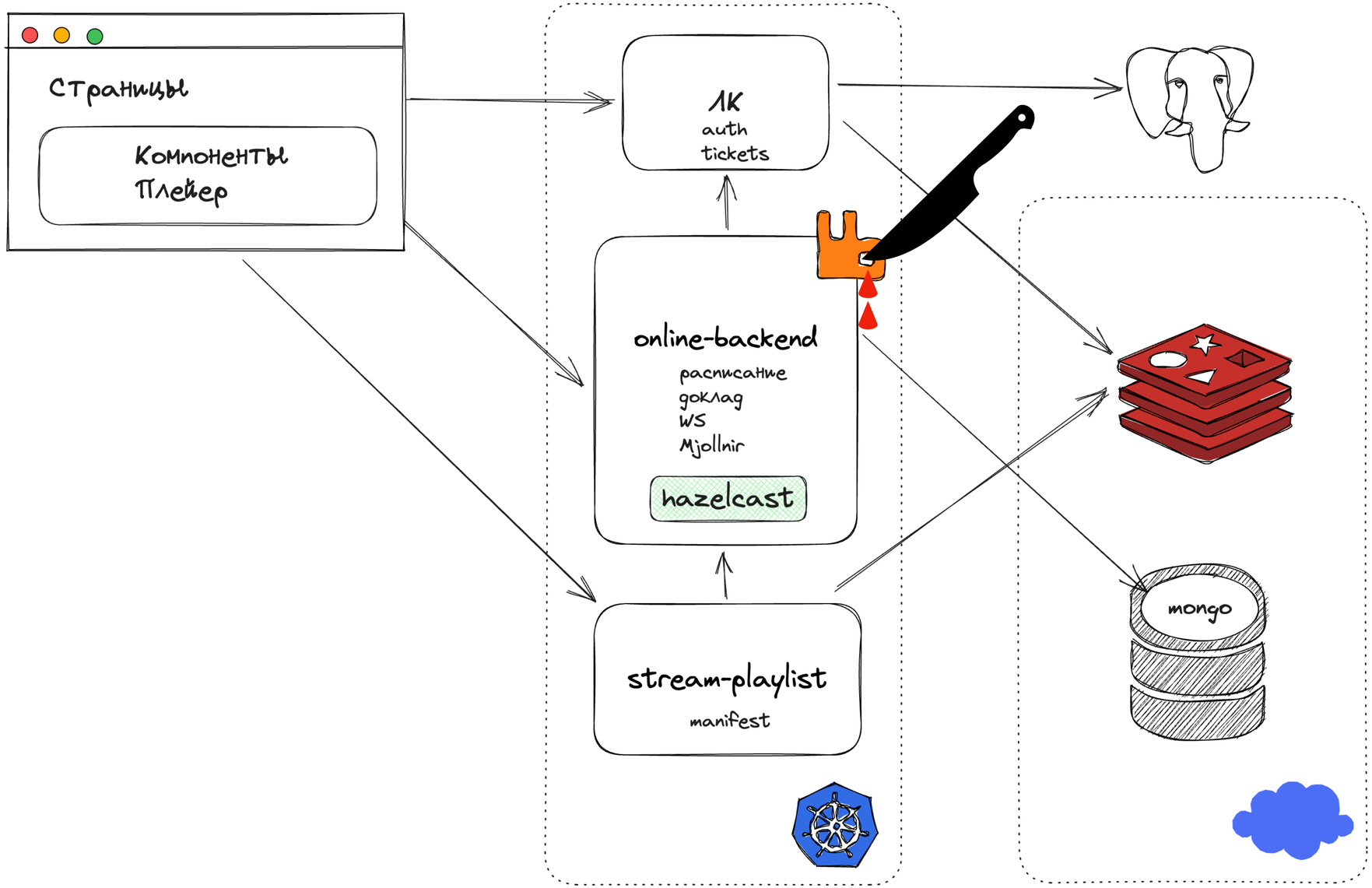
```
server.tomcat.max-threads=75 (умолчание 200)
server.tomcat.min-spare-threads=25
server.tomcat.accept-count=3000 (умолчание 100)
server.tomcat.mbeanregistry.enabled=true
```

Оптимизации throughput: rabbitmq

Оптимизации throughput: rabbitmq



Оптимизации throughput: rabbitmq



Оптимизации throughput: rabbitmq

```yaml
livenessProbe:
  exec:
    command:
      - sh
      - -ec
      - rabbitmq-diagnostics -q ping
  initialDelaySeconds: 15
  periodSeconds: 30
  timeoutSeconds: 20
  successThreshold: 1 # 1 - default
  failureThreshold: 6 # 3 - default
```

```yaml
readinessProbe:
  exec:
    command:
      - sh
      - -ec
      - rabbitmq-diagnostics -q
check_running && rabbitmq-diagnostics -q
check_local_alarms
  initialDelaySeconds: 15
  periodSeconds: 30
  timeoutSeconds: 20
  successThreshold: 1 # 1 - default
  failureThreshold: 3 # 3 - default
```

Оптимизации throughput: rabbitmq

```
livenessProbe:
  exec:
    command:
      - sh
      - -ec
      - rabbitmq-diagnostics -q ping
  initialDelaySeconds: 15
  periodSeconds: 30
  timeoutSeconds: 20
  successThreshold: 1 # 1 - default
  failureThreshold: 6 # 3 - default
```

```
readinessProbe:
  exec:
    command:
      - sh
      - -ec
      - rabbitmq-diagnostics -q
check_running && rabbitmq-diagnostics -q
check_local_alarms
  initialDelaySeconds: 15
  periodSeconds: 30
  timeoutSeconds: 20
  successThreshold: 1 # 1 - default
  failureThreshold: 3 # 3 - default
```

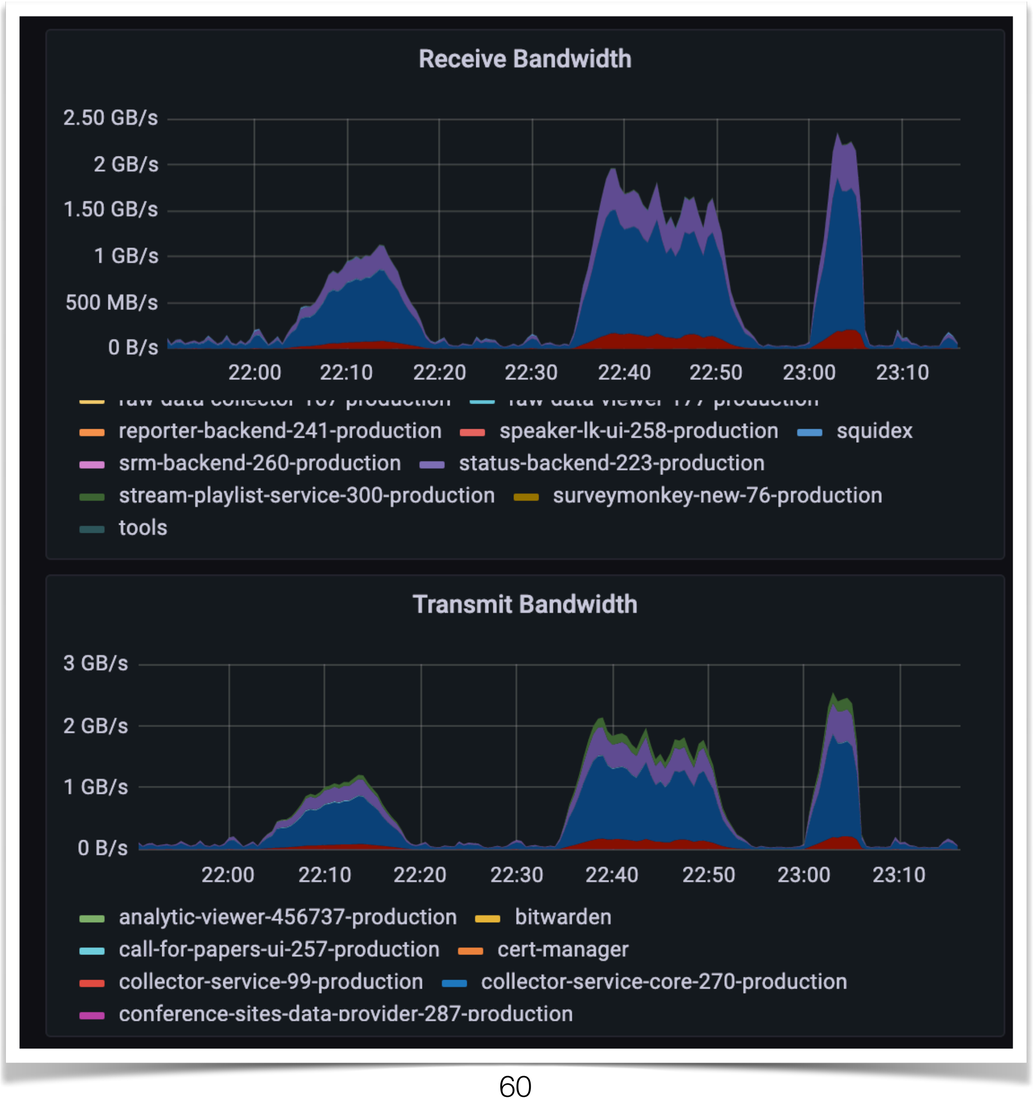
Оптимизации throughput: rabbitmq

```
livenessProbe:
  exec:
    command:
      - sh
      - -ec
      - rabbitmq-diagnostics -q ping
  initialDelaySeconds: 15
  periodSeconds: 30
  timeoutSeconds: 20
  successThreshold: 1 # 1 - default
  failureThreshold: 6 # 3 - default
```

```
readinessProbe:
  exec:
    command:
      - sh
      - -ec
      - rabbitmq-diagnostics -q
check_running && rabbitmq-diagnostics -q
check_local_alarms
  initialDelaySeconds: 15
  periodSeconds: 30
  timeoutSeconds: 20
  successThreshold: 1 # 1 - default
  failureThreshold: 3 # 3 - default
```

Оптимизации throughput: ограничения внутреннего трафика

Оптимизации throughput: ограничения внутреннего трафика



Оптимизации throughput: зипование внутренного трафика манифестов

```
#EXTM3U
#EXT-X-PLAYLIST-TYPE:EVENT
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:3
#EXT-X-MEDIA-SEQUENCE:1
#EXTINF:3.00000,
https://stream.jugru.org/100171/100171-track1-day3/100171-track1-day3_1920/00000/100171-track1-
day3_1920_20230917_072929_01129.ts?
auth=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY3Rpdml0eUlkIjoyMDAwMjk0MywiZXZlbnRJZCI6MTAwMTcxLCJpYXQ
iOjE2OTQ5MzYxMjcsImV4cCI6MTY5NDkzNjQyN30.hNAY3n0X7nUtedUF_X3NN3x5Rp5iiRUsBNNh8ULEIGo
#EXTINF:3.00000,
https://stream.jugru.org/100171/100171-track1-day3/100171-track1-day3_1920/00000/100171-track1-
day3_1920_20230917_072932_01130.ts?
auth=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY3Rpdml0eUlkIjoyMDAwMjk0MywiZXZlbnRJZCI6MTAwMTcxLCJpYXQ
iOjE2OTQ5MzYxMjcsImV4cCI6MTY5NDkzNjQyN30.hNAY3n0X7nUtedUF_X3NN3x5Rp5iiRUsBNNh8ULEIGo

...

#EXTINF:3.00000,
https://stream.jugru.org/100171/100171-track1-day3/100171-track1-day3_1920/00000/100171-track1-
day3_1920_20230917_073702_01280.ts?
auth=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY3Rpdml0eUlkIjoyMDAwMjk0MywiZXZlbnRJZCI6MTAwMTcxLCJpYXQ
iOjE2OTQ5MzYxMjcsImV4cCI6MTY5NDkzNjQyN30.hNAY3n0X7nUtedUF_X3NN3x5Rp5iiRUsBNNh8ULEIGo
```

Оптимизации throughput: зипование внутренного трафика манифестов

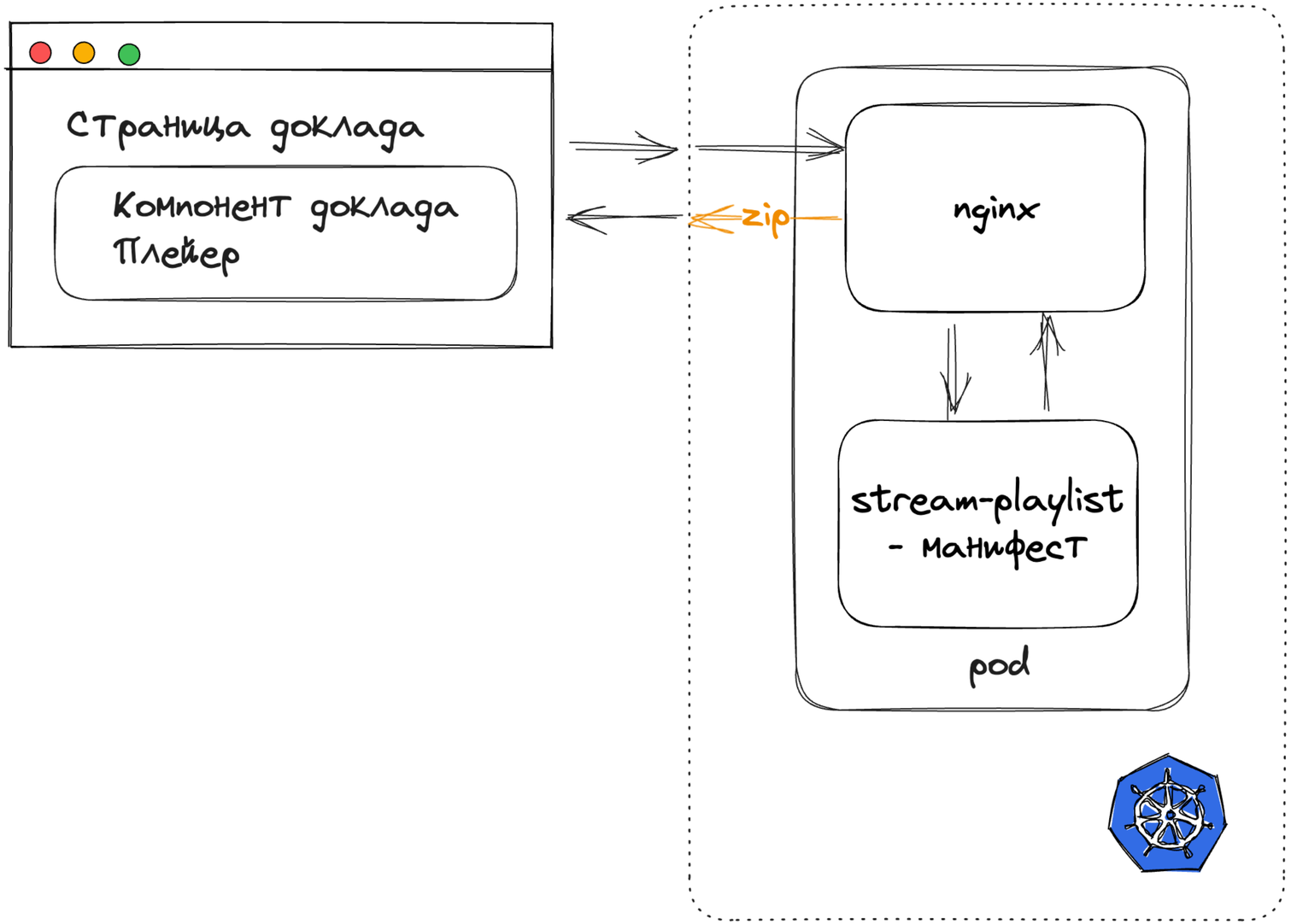
```
#EXTM3U
#EXT-X-PLAYLIST-TYPE:EVENT
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:3
#EXT-X-MEDIA-SEQUENCE:1
#EXTINF:3.00000,
https://stream.jugru.org/100171/100171-track1-day3/100171-track1-day3_1920/00000/100171-track1-
day3_1920_202309
auth=eyJhbGciOiJ                                                    MTAwMTcxLCJpYXQ
iOjE2OTQ5MzYxMjc
#EXTINF:3.00000,
https://stream.j                                                    1-track1-
day3_1920_202309
auth=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY3RpdmI0eUlkIjoyMDAwMjk0MywiZXZlbnRJZCI6MTAwMTcxLCJpYXQ
iOjE2OTQ5MzYxMjcsImV4cCI6MTY5NDkzNjQyN30.hNAY3n0X7nUtedUF_X3NN3x5Rp5iiRUsBNNh8ULEIGo

...

#EXTINF:3.00000,
https://stream.jugru.org/100171/100171-track1-day3/100171-track1-day3_1920/00000/100171-track1-
day3_1920_20230917_073702_01280.ts?
auth=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY3RpdmI0eUlkIjoyMDAwMjk0MywiZXZlbnRJZCI6MTAwMTcxLCJpYXQ
iOjE2OTQ5MzYxMjcsImV4cCI6MTY5NDkzNjQyN30.hNAY3n0X7nUtedUF_X3NN3x5Rp5iiRUsBNNh8ULEIGo
```

Часовой доклад: 60 * 60 / 3 = 1200 строк

Часовой доклад для 10K: 1200 * 10000 / 3 =

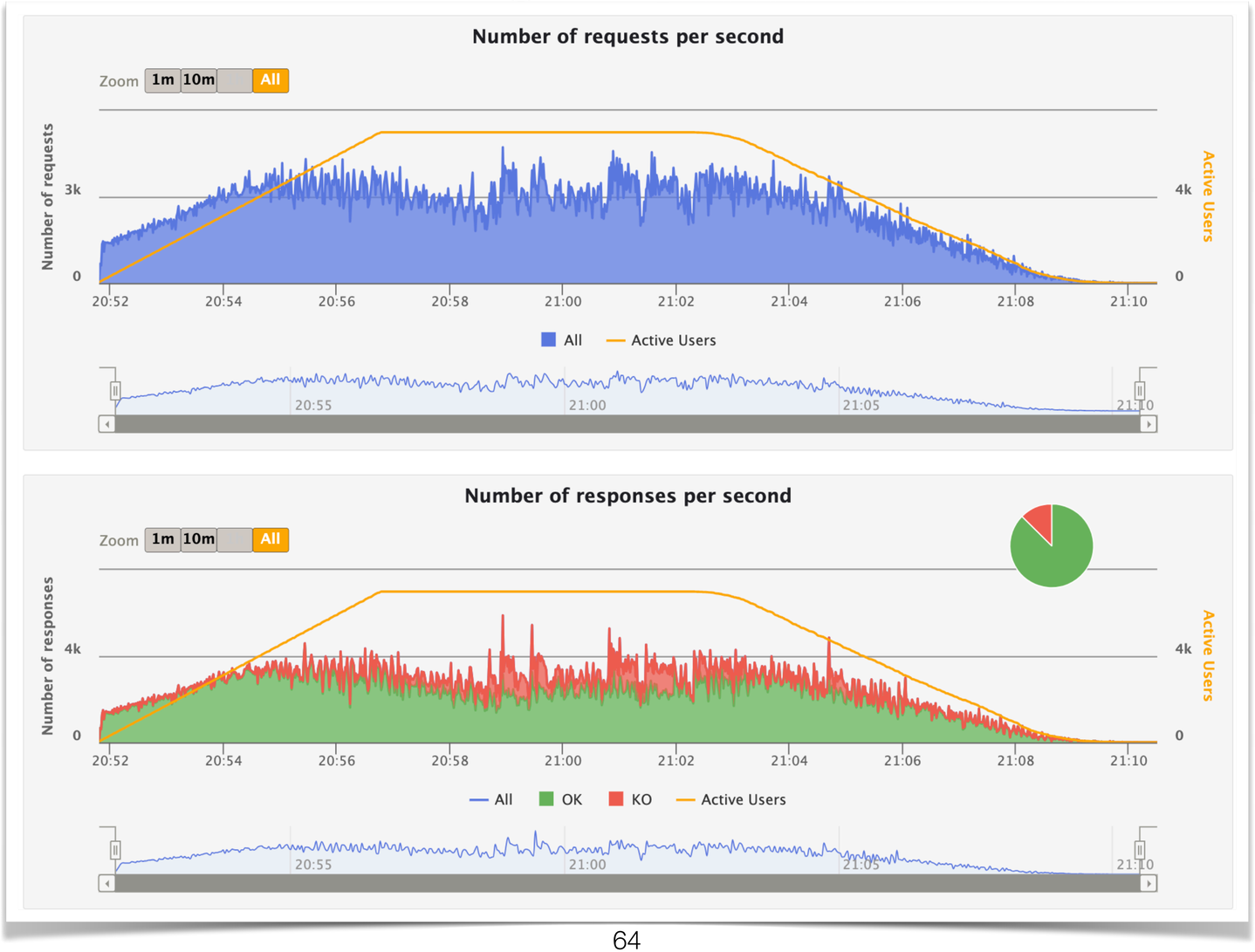
4M строк в секунду

Оптимизации throughput: зипование внутренного трафика манифестов

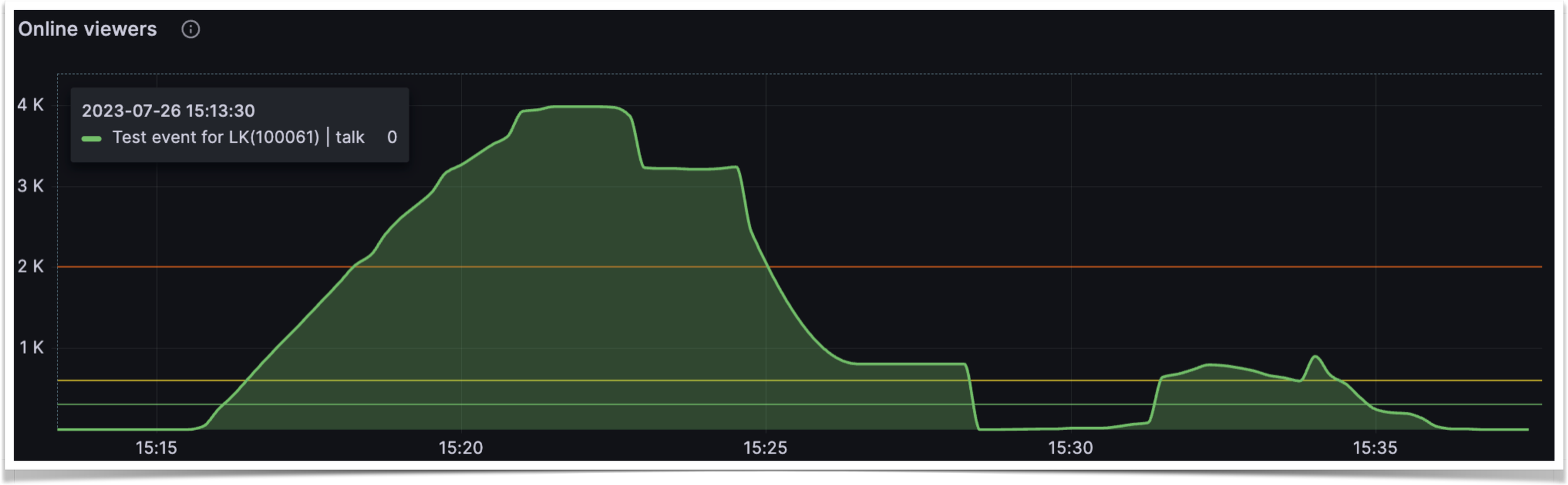


Оптимизации throughput: агрегации из hazelcast в mongo для now_watching

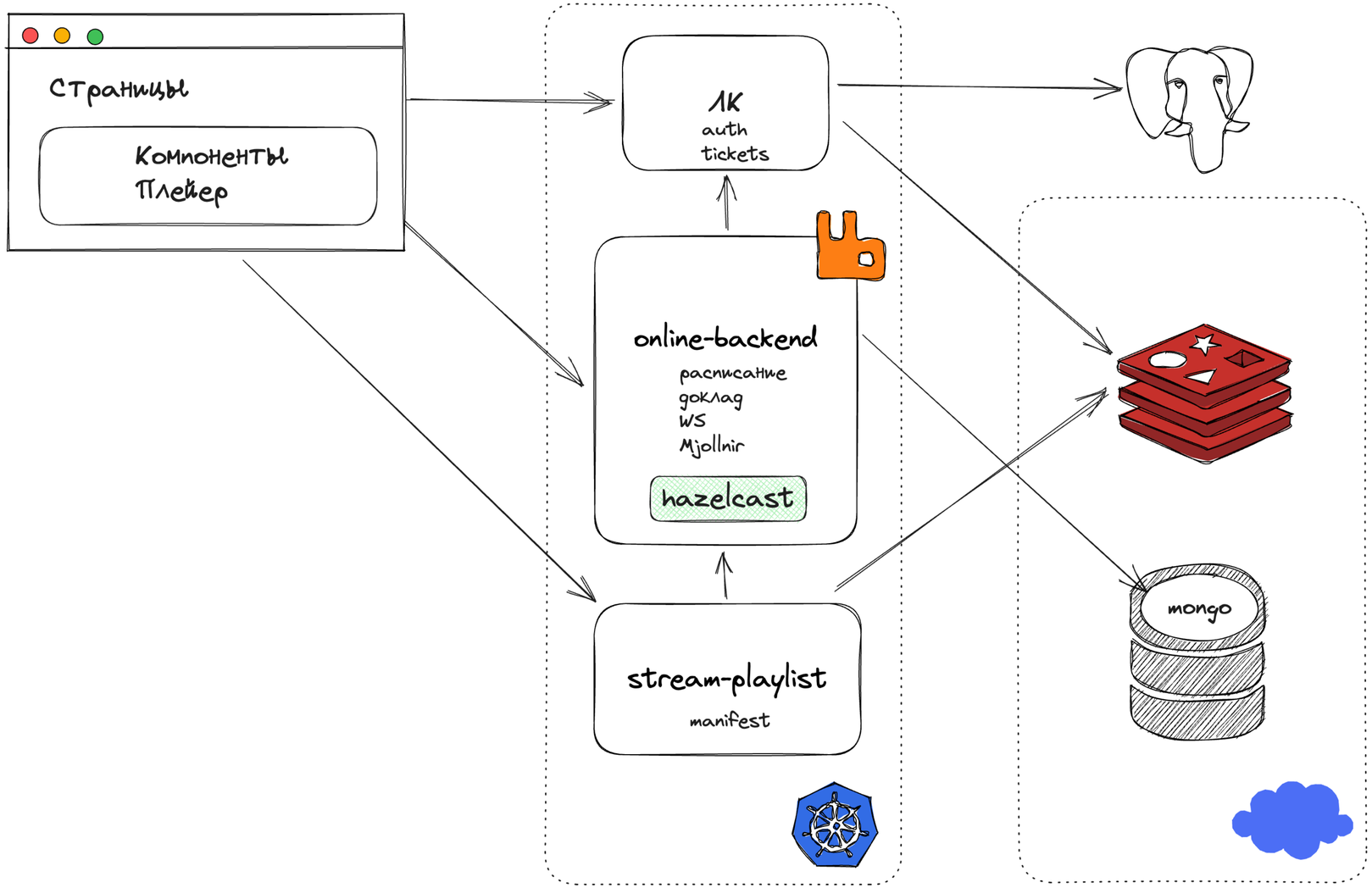
Оптимизации throughput: агрегации из hazelcast в mongo для now_watching



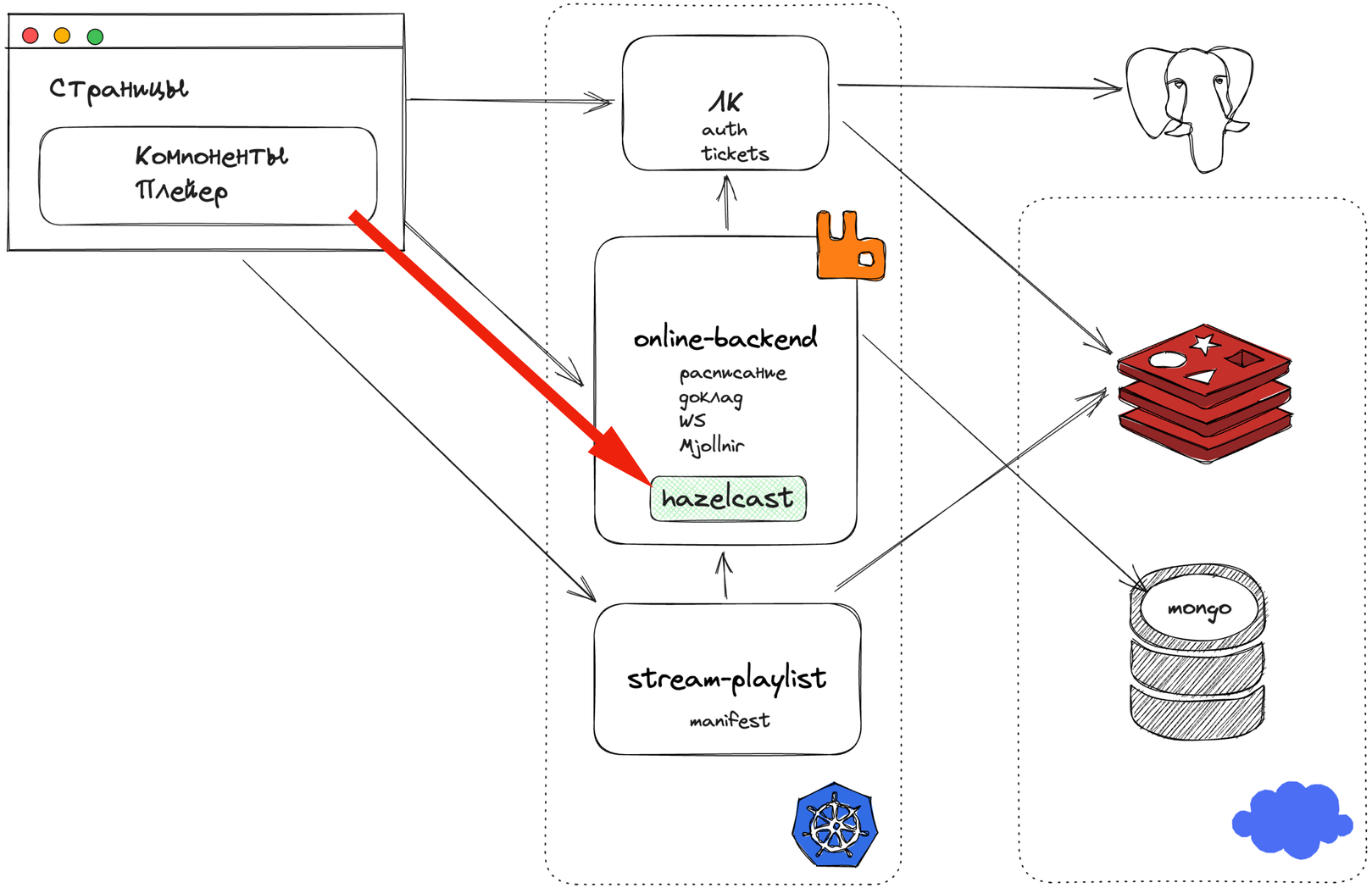
Оптимизации throughput: агрегации из hazelcast в mongo для now_watching



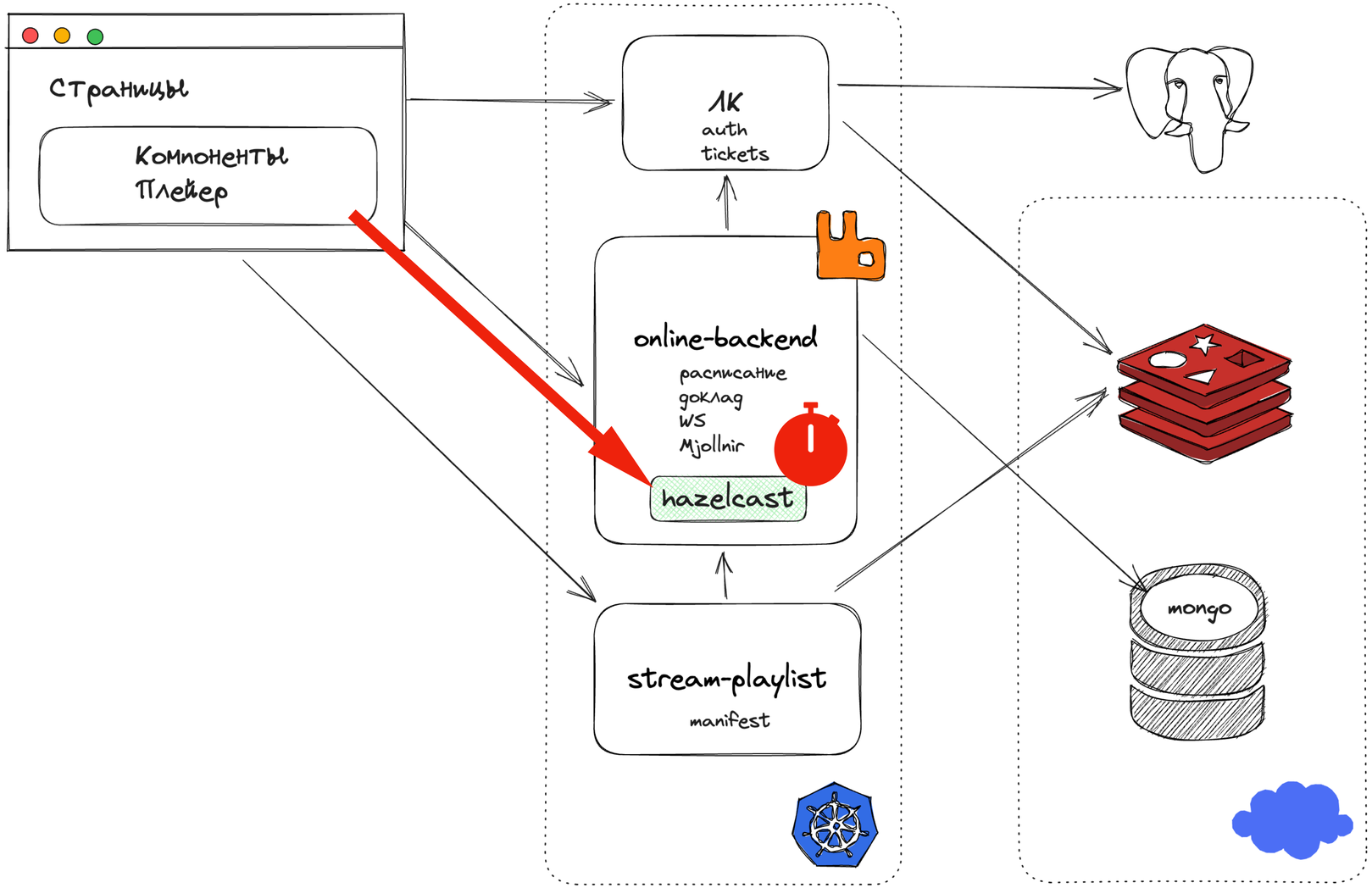
Оптимизации throughput: агрегации из hazelcast в mongo для now_watching



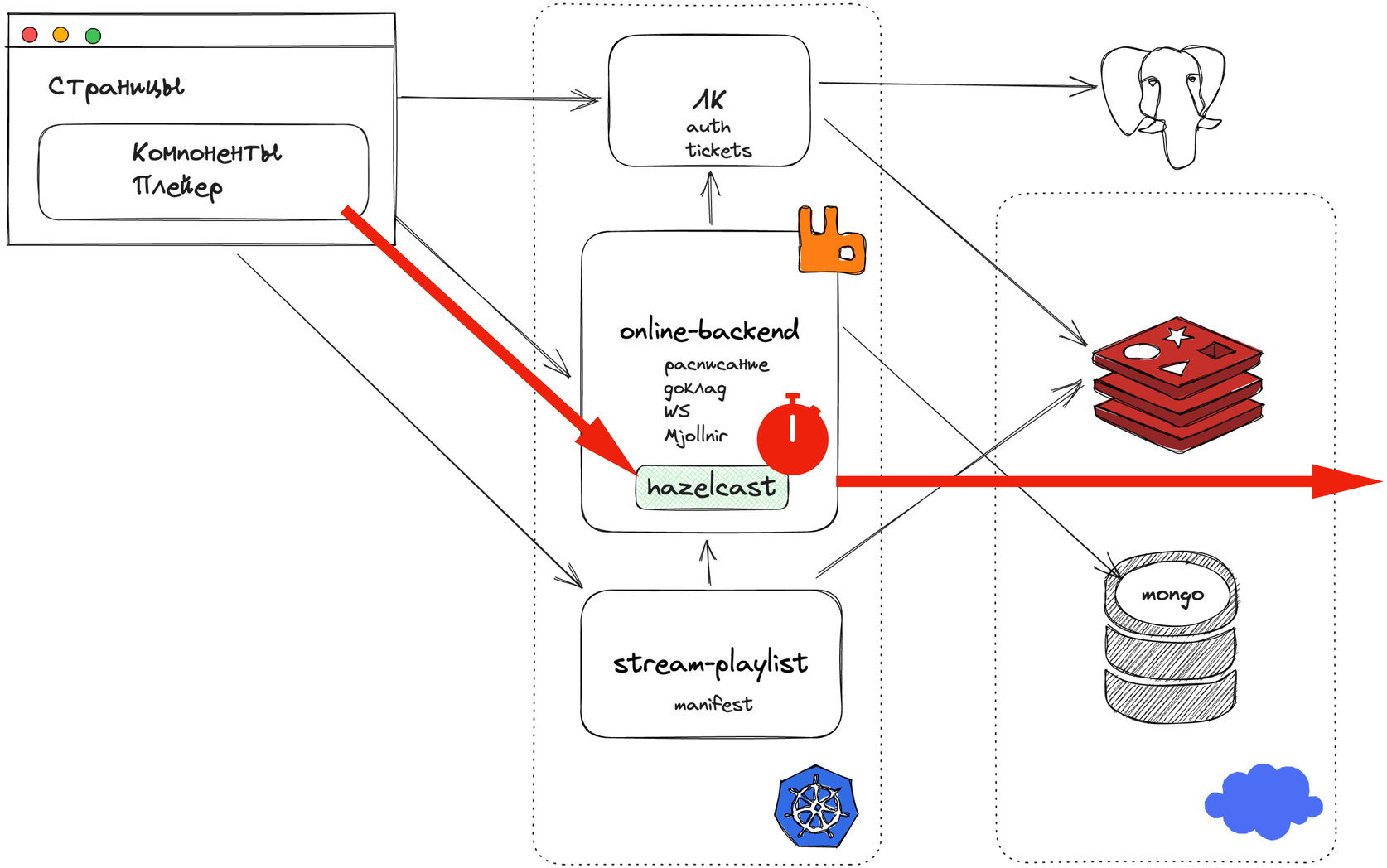
Оптимизации throughput: агрегации из hazelcast в mongo для now_watching



Оптимизации throughput: агрегации из hazelcast в mongo для now_watching



Оптимизации throughput: агрегации из hazelcast в mongo для now_watching



Оптимизации throughput: агрегации из hazelcast в mongo для now_watching

```
public class CountUniqueUsersAggregator implements Aggregator<Map.Entry<...>, Integer> {
    Set<Long> uniqueUserIds = new HashSet<>();
    @Override
    public void accumulate(Map.Entry<ImWatchingInMsg, ImWatchingInMsg> entry) {
        ImWatchingInMsg msg = entry.getValue();
        uniqueUserIds.add(msg.getLkUserId());
    }
    @Override
    public void combine(Aggregator aggregator) {
        this.uniqueUserIds.addAll(this.getClass().cast(aggregator).uniqueUserIds);
    }
    @Override
    public Integer aggregate() {
        return uniqueUserIds.size();
    }
}
```

Оптимизации throughput: агрегации из hazelcast в mongo для now_watching

```
public class CountUniqueUsersAggregator implements Aggregator<Map.Entry<...>, Integer> {
    Set<Long> uniqueUserIds = new HashSet<>();
    @Override
    public void accumulate(Map.Entry<ImWatchingInMsg, ImWatchingInMsg> entry) {
        ImWatchingInMsg msg = entry.getValue();
        uniqueUserIds.add(msg.getLkUserId());
    }
    @Override
    public void combine(Aggregator aggregator) {
        this.uniqueUserIds.addAll(this.getClass().cast(aggregator).uniqueUserIds);
    }
    @Override
    public Integer aggregate() {
        return uniqueUserIds.size();
    }
}
```

Оптимизации throughput: агрегации из hazelcast в mongo для now_watching

```
public class CountUniqueUsersAggregator implements Aggregator<Map.Entry<...>, Integer> {
    Set<Long> uniqueUserIds = new HashSet<>();
    @Override
    public void accumulate(Map.Entry<ImWatchingInMsg, ImWatchingInMsg> entry) {
        ImWatchingInMsg msg = entry.getValue();
        uniqueUserIds.add(msg.getLkUserId());
    }
    @Override
    public void combine(Aggregator aggregator) {
        this.uniqueUserIds.addAll(this.getClass().cast(aggregator).uniqueUserIds);
    }
    @Override
    public Integer aggregate() {
        return uniqueUserIds.size();
    }
}
```

Оптимизации throughput: агрегации из hazelcast в mongo для now_watching

```
public class CountUniqueUsersAggregator implements Aggregator<Map.Entry<...>, Integer> {
    Set<Long> uniqueUserIds = new HashSet<>();
    @Override
    public void accumulate(Map.Entry<ImWatchingInMsg, ImWatchingInMsg> entry) {
        ImWatchingInMsg msg = entry.getValue();
        uniqueUserIds.add(msg.getLkUserId());
    }
    @Override
    public void combine(Aggregator aggregator) {
        this.uniqueUserIds.addAll(this.getClass().cast(aggregator).uniqueUserIds);
    }
    @Override
    public Integer aggregate() {
        return uniqueUserIds.size();
    }
}
```

Оптимизации throughput: агрегации из hazelcast в mongo для now_watching

```
public class CountUniqueUsersAggregator implements Aggregator<Map.Entry<...>, Integer> {
    Set<Long> uniqueUserIds = new HashSet<>();
    @Override
    public void accumulate(Map.Entry<ImWatchingInMsg, ImWatchingInMsg> entry) {
        ImWatchingInMsg msg = entry.getValue();
        uniqueUserIds.add(msg.getLkUserId());
    }
    @Override
    public void combine(Aggregator aggregator) {
        this.uniqueUserIds.addAll(this.getClass().cast(aggregator).uniqueUserIds);
    }
    @Override
    public Integer aggregate() {
        return uniqueUserIds.size();
    }
}
```

Оптимизации throughput: агрегации из hazelcast в mongo для now_watching

```
IMap<ImWatchingInMsg, ImWatchingInMsg> iAmWatchingMap =
        hazelcast.getMap(map$fact$i_m_watching);
Integer nowWatchingNumber = iAmWatchingMap.aggregate(
        new CountUniqueUsersAggregator(),
        entry → {
              ImWatchingInMsg imw = entry.getValue();
                    return imw instanceof ImWatchingLive20InMsg
                        && imw.getCreatedAt().after(from)
                        && imw.getJugCrmId() == eventId
                        && imw.getDayNumber() == dayNumber
                        && imw.getTrackNumber() == trackNumber
                        && imw.getSlotNumber() == slotNumber
                        && imw.getTalkId() == talkId
                        && type.equals(((ImWatchingLive20InMsg) imw).getType());
        }
);
```

Оптимизации throughput: агрегации из hazelcast в mongo для now_watching

```
IMap<ImWatchingInMsg, ImWatchingInMsg> iAmWatchingMap =
        hazelcast.getMap(map$fact$i_m_watching);
Integer nowWatchingNumber = iAmWatchingMap.aggregate(
        new CountUniqueUsersAggregator(),
        entry → {
                ImWatchingInMsg imw = entry.getValue();
                        return imw instanceof ImWatchingLive20InMsg
                            && imw.getCreatedAt().after(from)
                            && imw.getJugCrmId() == eventId
                            && imw.getDayNumber() == dayNumber
                            && imw.getTrackNumber() == trackNumber
                            && imw.getSlotNumber() == slotNumber
                            && imw.getTalkId() == talkId
                            && type.equals(((ImWatchingLive20InMsg) imw).getType());
        }
);
```

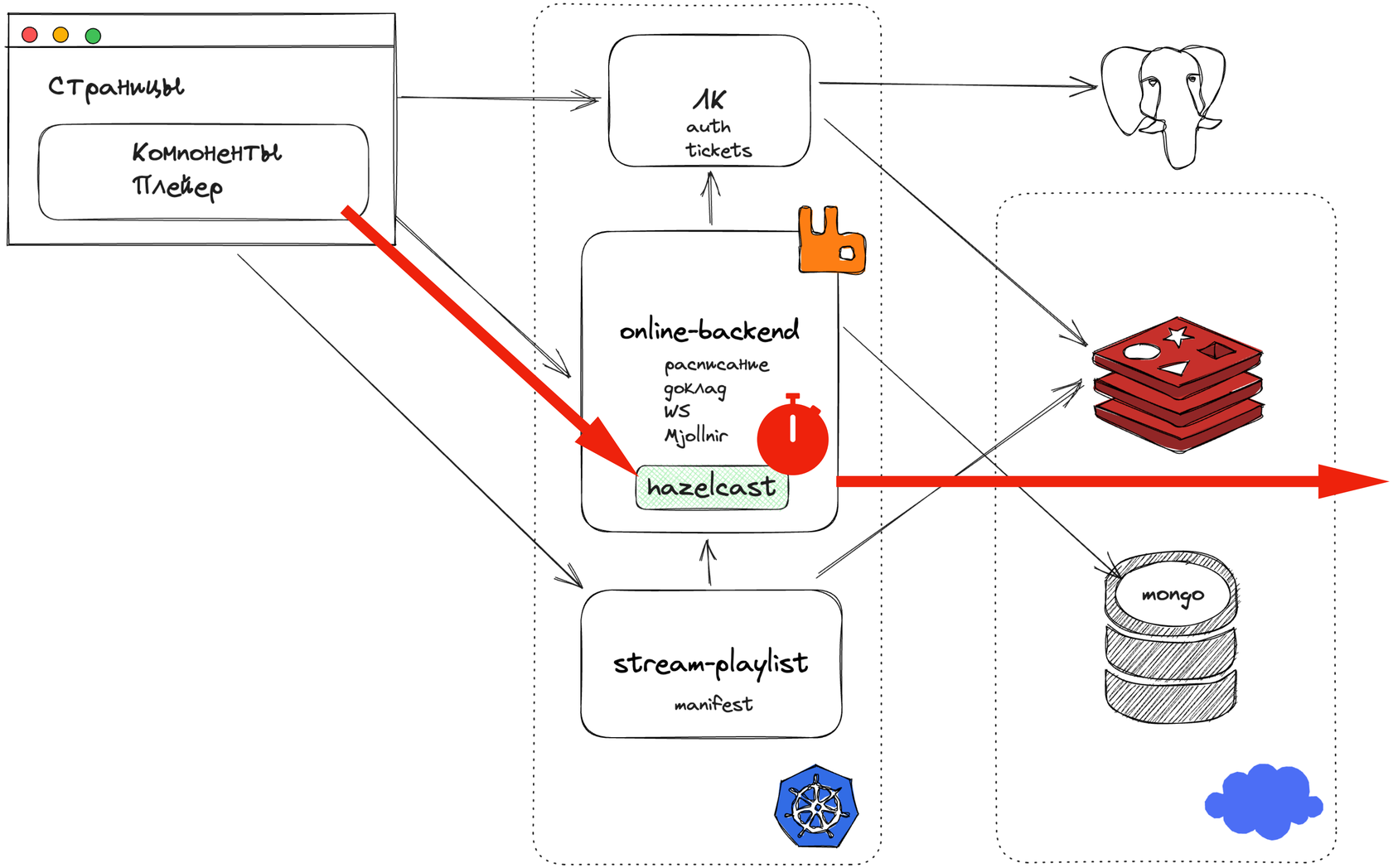
Оптимизации throughput: агрегации из hazelcast в mongo для now_watching

```
IMap<ImWatchingInMsg, ImWatchingInMsg> iAmWatchingMap =
        hazelcast.getMap(map$fact$i_m_watching);
Integer nowWatchingNumber = iAmWatchingMap.aggregate(
        new CountUniqueUsersAggregator(),
        entry → {
                ImWatchingInMsg imw = entry.getValue();
                      return imw instanceof ImWatchingLive20InMsg
                          && imw.getCreatedAt().after(from)
                          && imw.getJugCrmId() == eventId
                          && imw.getDayNumber() == dayNumber
                          && imw.getTrackNumber() == trackNumber
                          && imw.getSlotNumber() == slotNumber
                          && imw.getTalkId() == talkId
                          && type.equals(((ImWatchingLive20InMsg) imw).getType());
        }
);
```

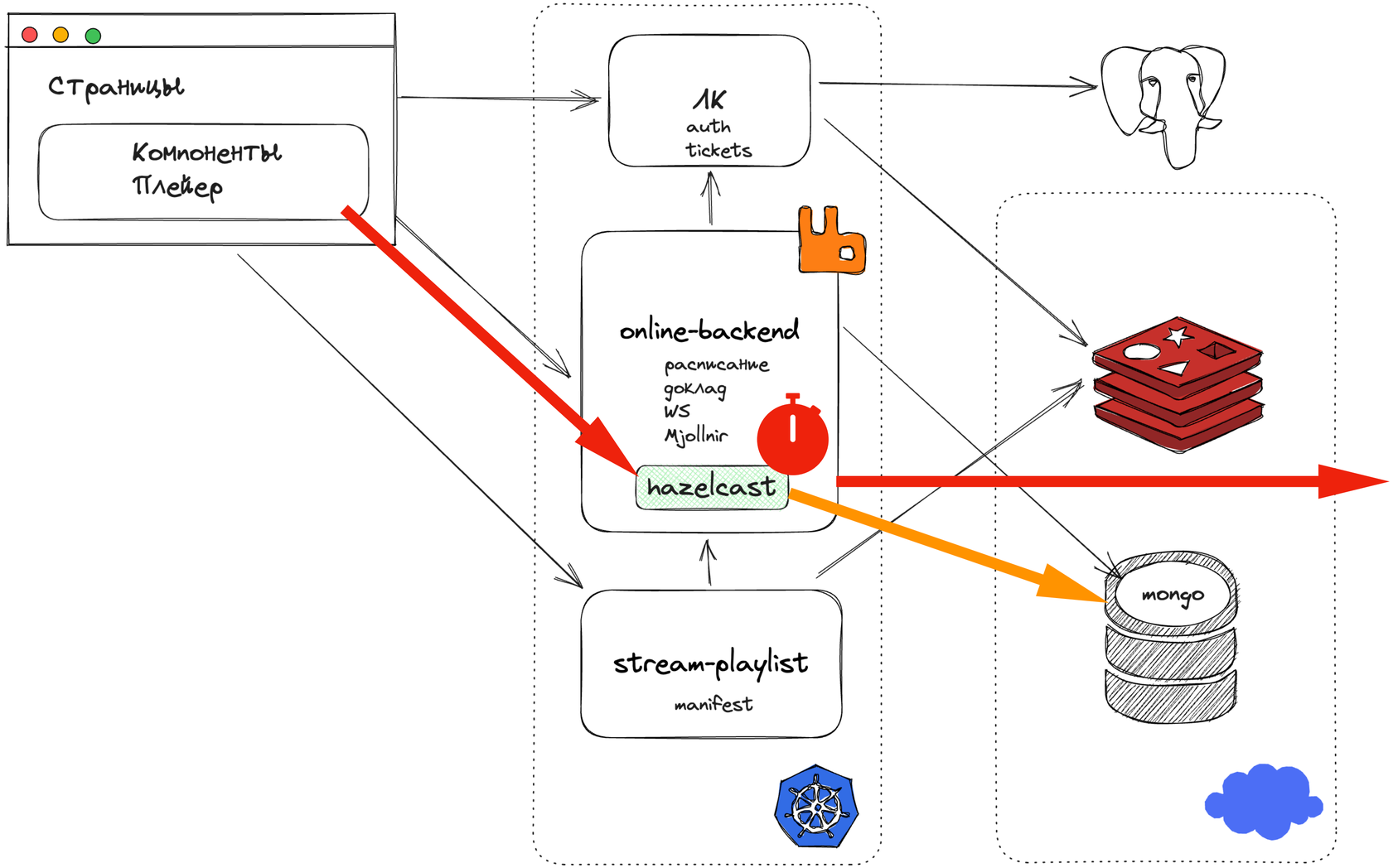
Оптимизации throughput: агрегации из hazelcast в mongo для now_watching

```
IMap<ImWatchingInMsg, ImWatchingInMsg> iAmWatchingMap =
        hazelcast.getMap(map$fact$i_m_watching);
Integer nowWatchingNumber = iAmWatchingMap.aggregate(
        new CountUniqueUsersAggregator(),
        entry → {
                ImWatchingInMsg imw = entry.getValue();
                    return imw instanceof ImWatchingLive20InMsg
                        && imw.getCreatedAt().after(from)
                        && imw.getJugCrmId() == eventId
                        && imw.getDayNumber() == dayNumber
                        && imw.getTrackNumber() == trackNumber
                        && imw.getSlotNumber() == slotNumber
                        && imw.getTalkId() == talkId
                        && type.equals(((ImWatchingLive20InMsg) imw).getType());
        }
);
```

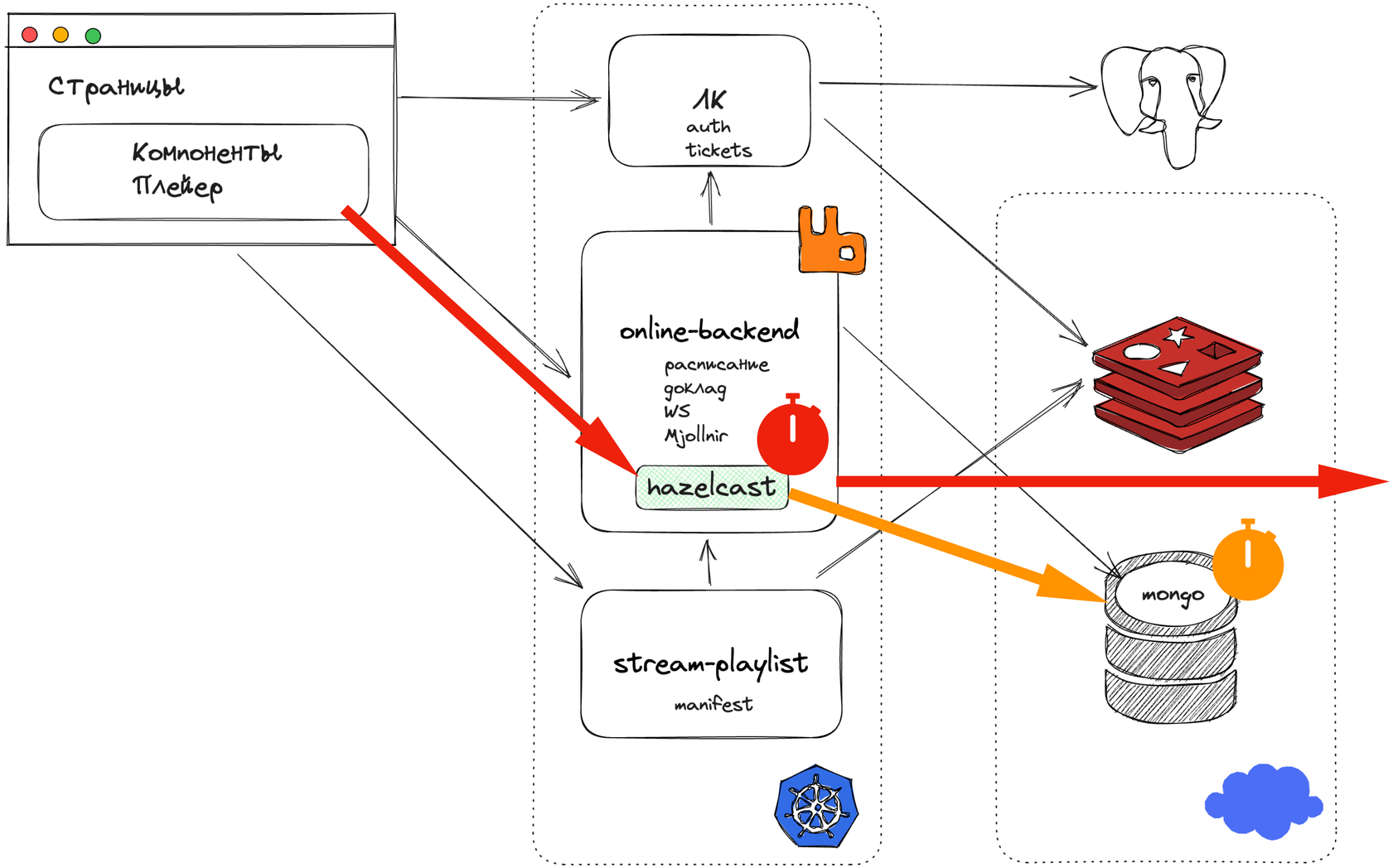
Оптимизации throughput: агрегации из hazelcast в mongo для now_watching



Оптимизации throughput: агрегации из hazelcast в mongo для now_watching



Оптимизации throughput: агрегации из hazelcast в mongo для now_watching



Оптимизации throughput: агрегации из hazelcast в mongo для now_watching

```
public List<Result> executeNowWatchingPerInternalSlotAggregation(Date from, Date to) {
    MatchOperation match = match(new Criteria(timestampAggregation).gte(from).lte(to)); //1
    GroupOperation group =                                                              //2
            group("jugCrmId", "dayNumber", "trackNumber", "slotNumber", "talkId", "type")
               .addToSet(“lkUserId").as("uniqueUsers");
    ProjectionOperation project = project()                                             //3
            .andExpression("_id.jugCrmId").as("jugCrmId")
            ...
            .and(context → new Document("$size", "$uniqueUsers")).as("uniqueUsersCount");
    Aggregation agg = newAggregation(match, group, project)
            .withOptions(newAggregationOptions().allowDiskUse(true).build());
    return secondaryMongoTemplate
            .aggregate(agg, ImWatchingLive20InMsg.class, Result.class)
            .getMappedResults();
}
```

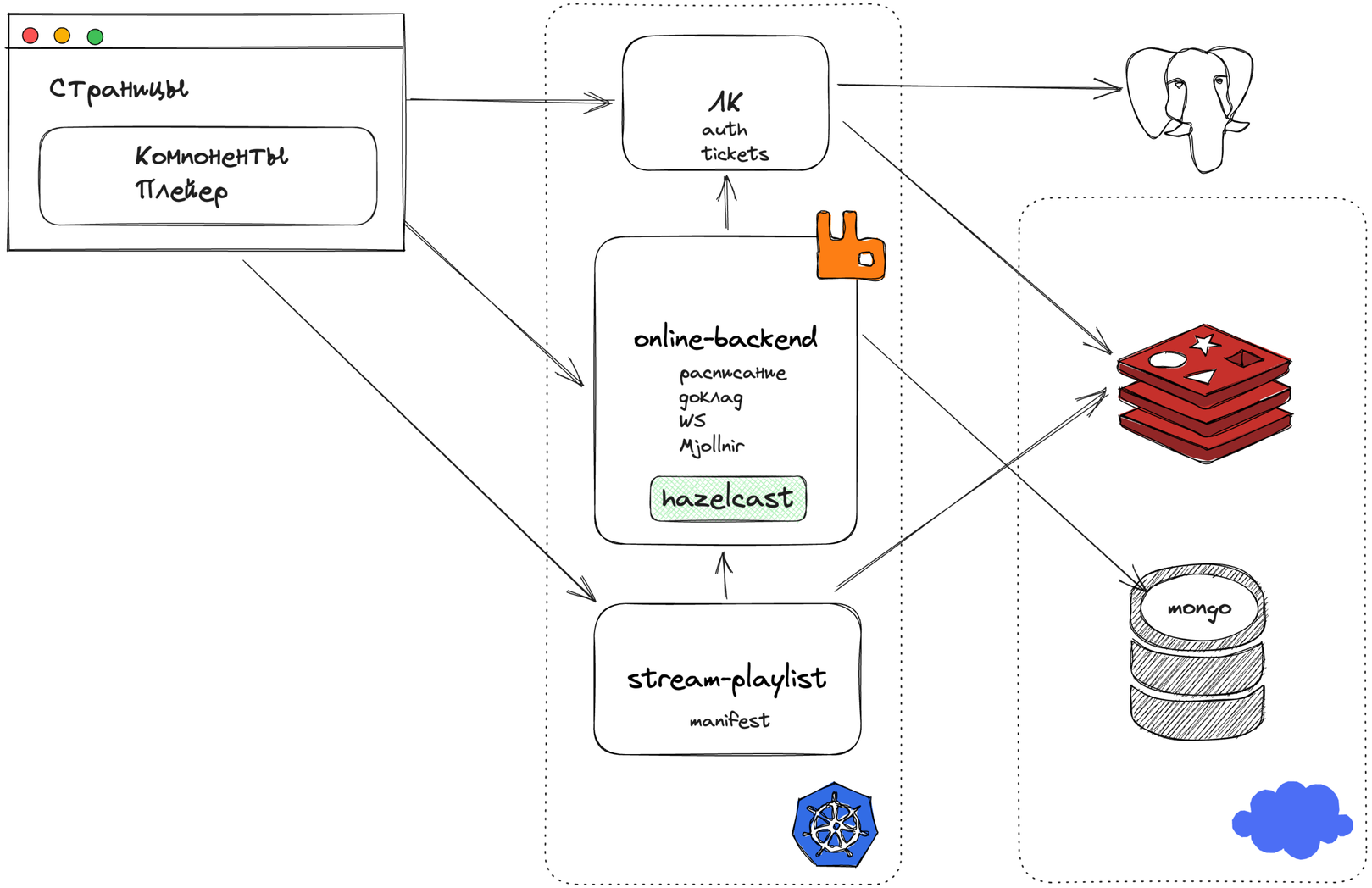
Оптимизации throughput: агрегации из hazelcast в mongo для now_watching

```
public List<Result> executeNowWatchingPerInternalSlotAggregation(Date from, Date to) {
    MatchOperation match = match(new Criteria(timestampAggregation).gte(from).lte(to)); //1
    GroupOperation group =                                                               //2
            group("jugCrmId", "dayNumber", "trackNumber", "slotNumber", "talkId", "type")
               .addToSet("lkUserId").as("uniqueUsers");
    ProjectionOperation project = project()                                              //3
            .andExpression("_id.jugCrmId").as("jugCrmId")
            ...
            .and(context -> new Document("$size", "$uniqueUsers")).as("uniqueUsersCount");
    Aggregation agg = newAggregation(match, group, project)
            .withOptions(newAggregationOptions().allowDiskUse(true).build());
    return secondaryMongoTemplate
            .aggregate(agg, ImWatchingLive20InMsg.class, Result.class)
            .getMappedResults();
}
```

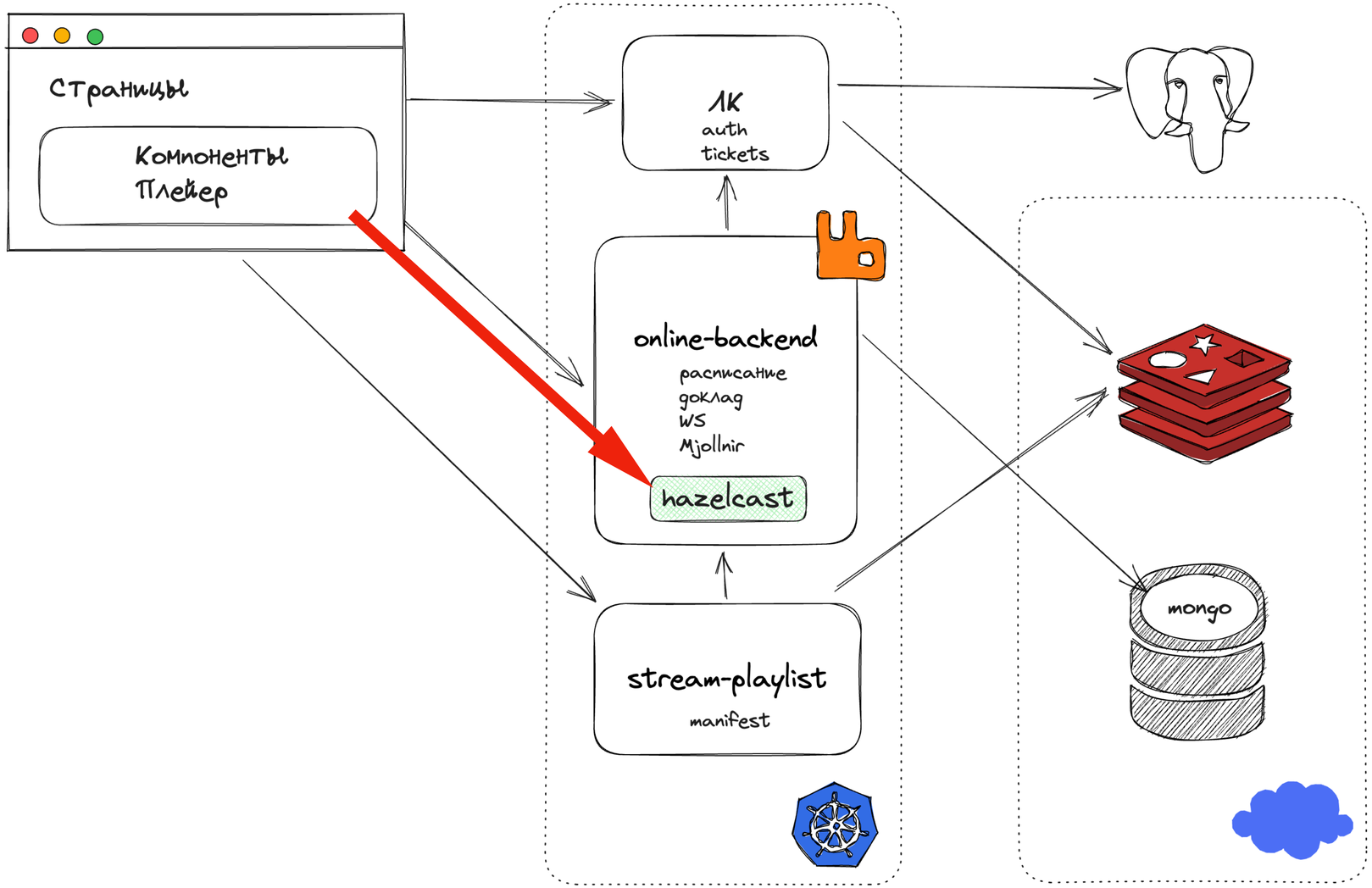
Оптимизации throughput: уменьшение нагрузки на managed mongodb

Оптимизации throughput: hazelcast IMap + MapStore tuning для imw

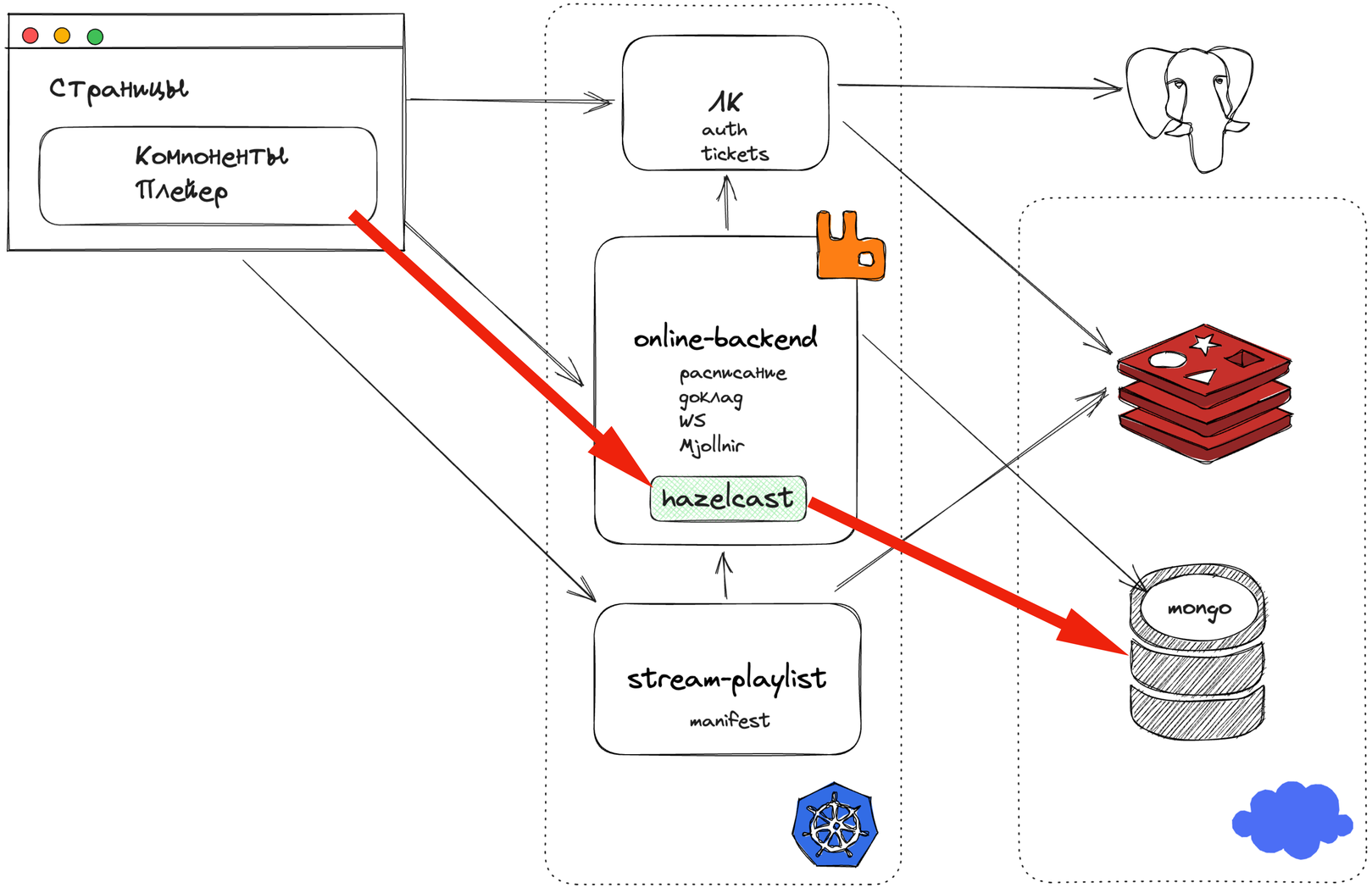
Оптимизации throughput: hazelcast IMap + MapStore tuning для imw



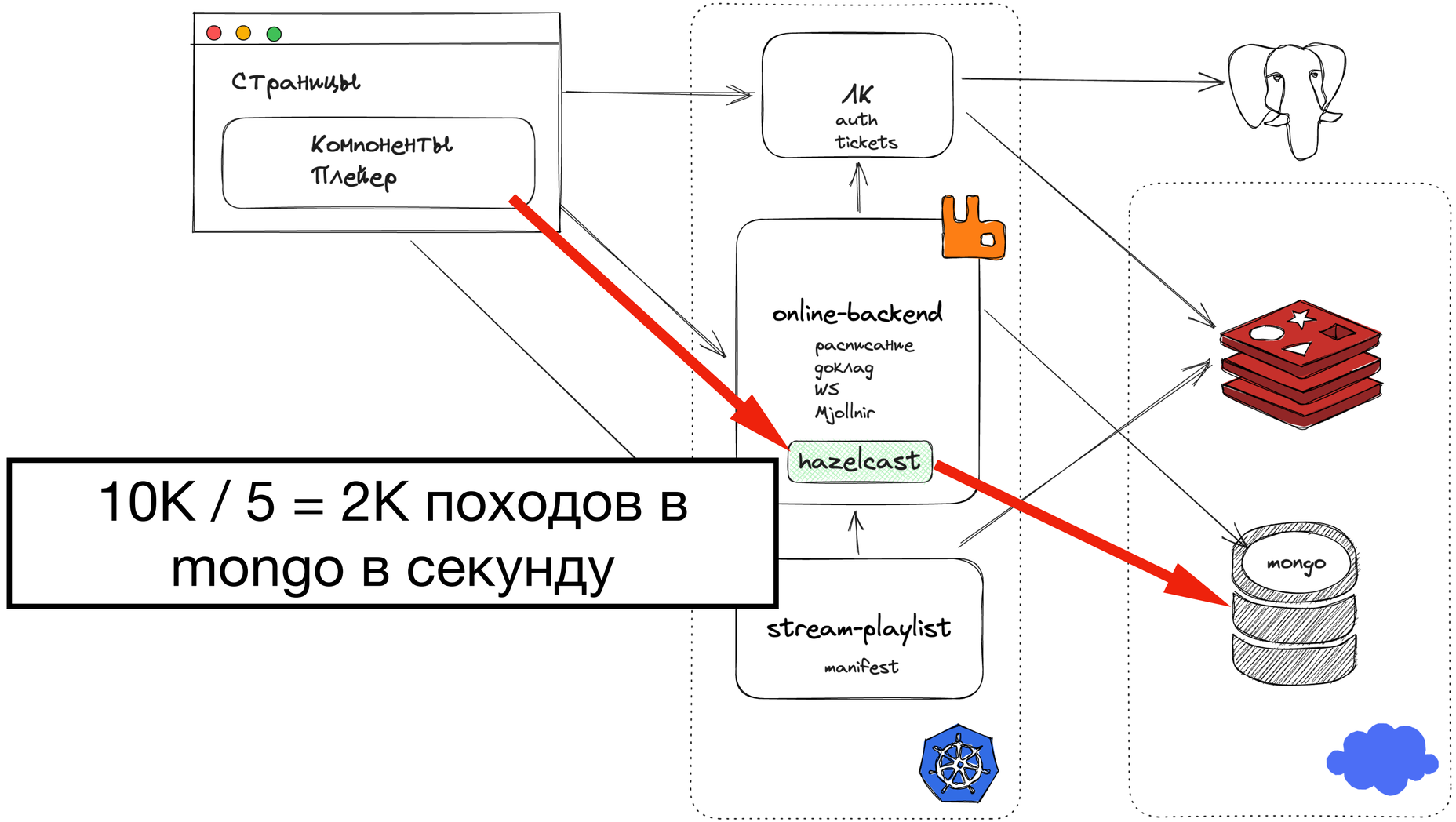
Оптимизации throughput: hazelcast IMap + MapStore tuning для imw



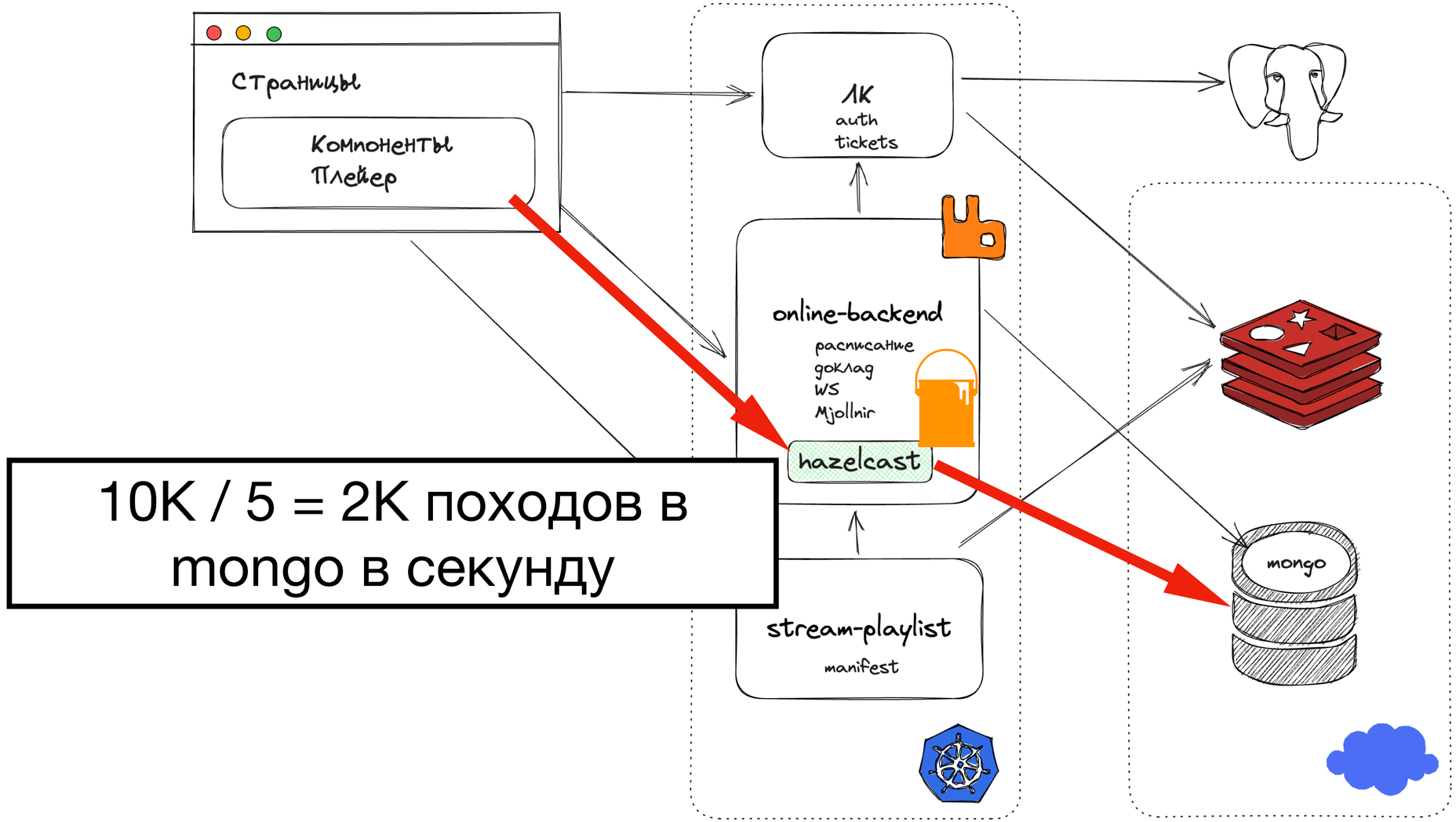
Оптимизации throughput: hazelcast IMap + MapStore tuning для imw



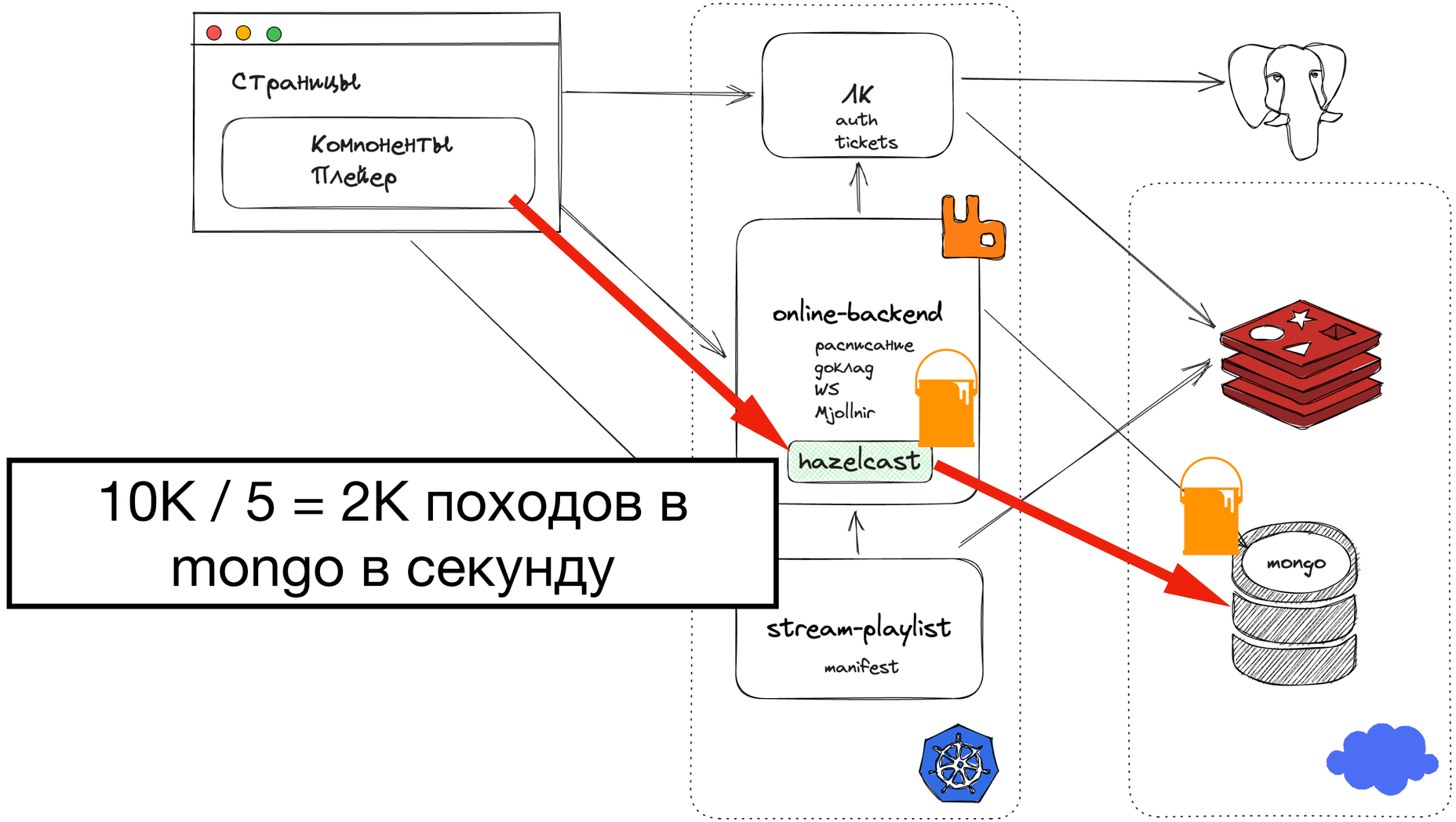
Оптимизации throughput: hazelcast IMap + MapStore tuning для imw



Оптимизации throughput: hazelcast IMap + MapStore tuning для imw



Оптимизации throughput: hazelcast IMap + MapStore tuning для imw



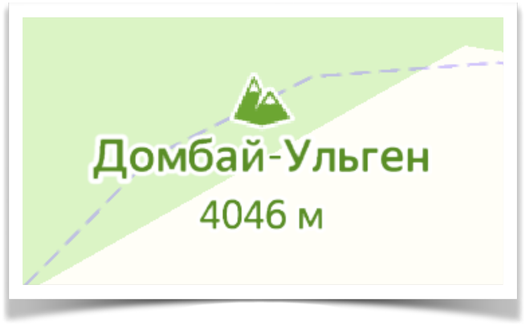
Оптимизации throughput: hazelcast IMap + MapStore tuning для imw

```
cfg.addMapConfig(new MapConfig()
        .setName(map$fact$i_m_watching$v2)
        .setMapStoreConfig(new MapStoreConfig()
                .setImplementation(imWatchingMapStore)
                .setWriteCoalescing(true)//no MAP_WRITE_BEHIND_QUEUE_CAPACITY
                //будет вызываться MapStore.storeAll()
                .setWriteDelaySeconds(hazelcastConfigMapWriteBehindDelaySeconds)
                .setWriteBatchSize(hazelcastConfigMapWriteBehindBatchSize)
        )
);

hazelcast.config.map.write.behind.delay.seconds=7
hazelcast.config.map.write.behind.batch.size=2000
```

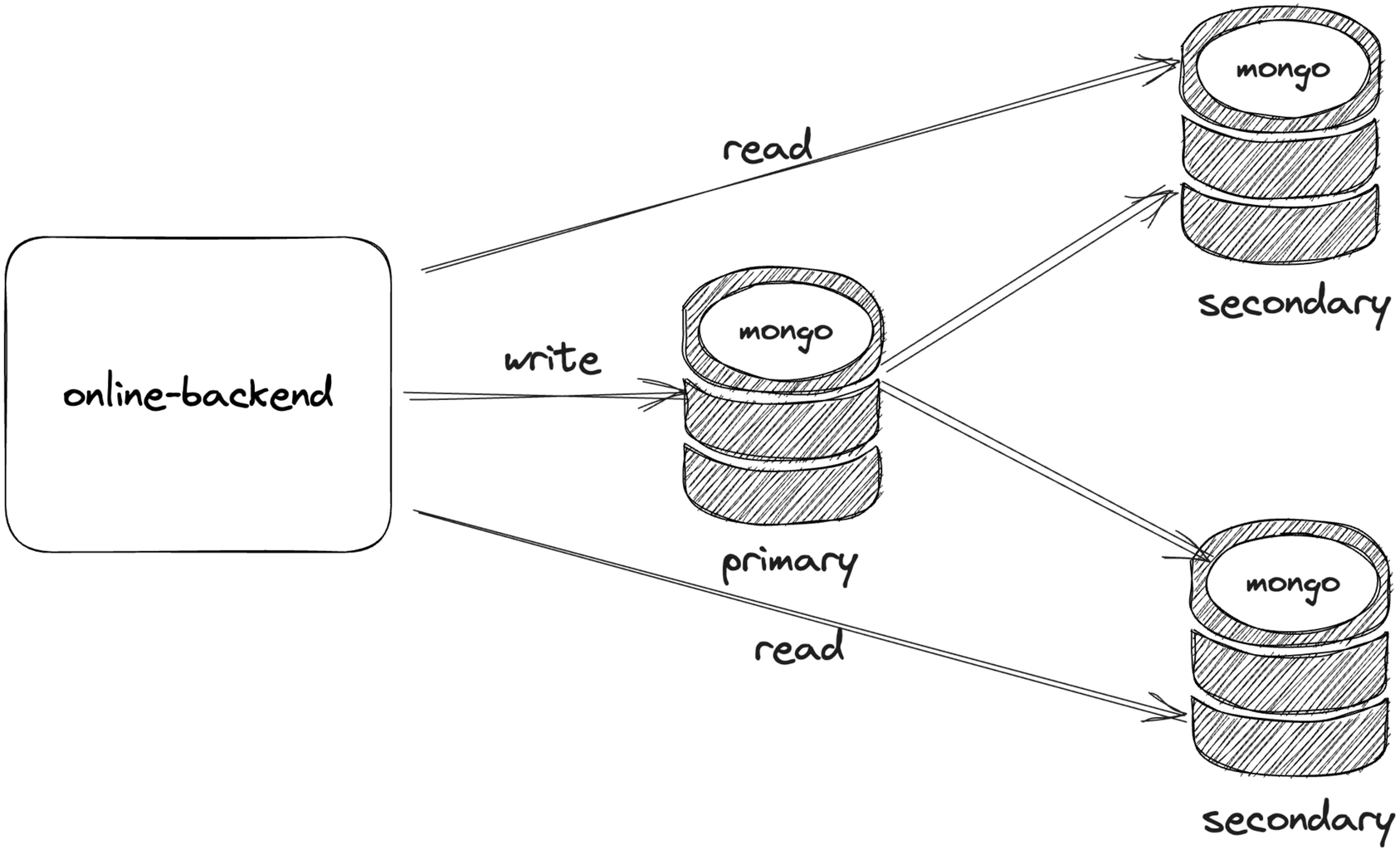
Оптимизации throughput: hazelcast IMap + MapStore tuning для imw

```
cfg.addMapConfig(new MapConfig()
        .setName(map$fact$i_m_watching$v2)
        .setMapStoreConfig(new MapStoreConfig()
                .setImplementation(imWatchingMapStore)
                .setWriteCoalescing(true)//no MAP_WRITE_BEHIND_QUEUE_CAPACITY
                //будет вызываться MapStore.storeAll()
                .setWriteDelaySeconds(hazelcastConfigMapWriteBehindDelaySeconds)
                .setWriteBatchSize(hazelcastConfigMapWriteBehindBatchSize)
        )
);

hazelcast.config.map.write.behind.delay.seconds=7
hazelcast.config.map.write.behind.batch.size=2000
```

Оптимизации throughput: глобальное чтение из реплик на монге



Оптимизации throughput: глобальное чтение из реплик на монге



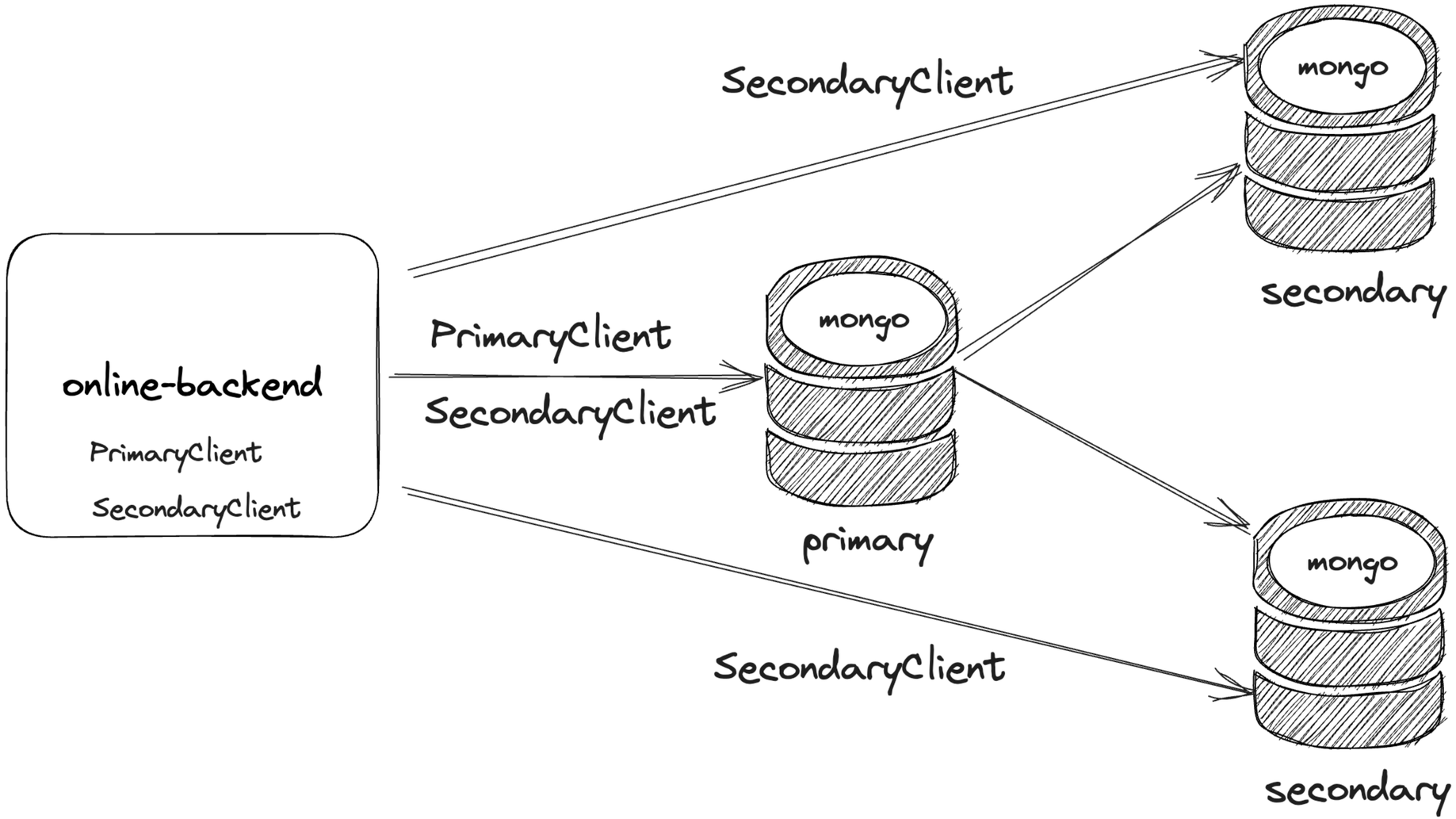
Оптимизации throughput: глобальное чтение из реплик на монге

```
spring.data.mongodb.uri=mongodb://\
  ${mongo.user}:${mongo.pass}@\
  ${mongo.replica}/\
  ${mongo.db}?authSource=admin\
  &connectTimeoutMS=${mongo.connectTimeoutMS}\
  &socketTimeoutMS=${mongo.socketTimeoutMS}\
  &readPreference=secondaryPreferred\
  &maxStalenessSeconds=180
```

Оптимизации throughput: глобальное чтение из реплик на монге

```
spring.data.mongodb.uri=mongodb://\
  ${mongo.user}:${mongo.pass}@\
  ${mongo.replica}/\
  ${mongo.db}?authSource=admin\
  &connectTimeoutMS=${mongo.connectTimeoutMS}\
  &socketTimeoutMS=${mongo.socketTimeoutMS}\
  &readPreference=secondaryPreferred\
  &maxStalenessSeconds=180
```

Оптимизации throughput: primary + secondary MongoTemplates



Оптимизации throughput: primary + secondary MongoTemplates

```
@Configuration
@EnableMongoRepositories(basePackages = "org.jugru.online.backend.repository",
                         mongoTemplateRef = "primMongoTemplate")
@EnableMongoAuditing
public class MongoPrimaryConfig {
  @Bean(name = "primMongoClient")
  public MongoClient primMongoClient(@Value("${spring.data.mongodb.uri}") String connection) {
      return MongoClients.create(connection);
  }
  @Primary
  @Bean(name = "primMongoDBFactory")
  public MongoDbFactory primMongoDatabaseFactory(@Qualifier("primMongoClient") MongoClient c) {
      return new SimpleMongoClientDbFactory(c, databaseName);
  }
  @Primary
  @Bean(name = "primMongoTemplate")
  public MongoTemplate primMongoTemplate(@Qualifier("primMongoDBFactory") MongoDbFactory f) {
      return new MongoTemplate(f);
  }
}
```

Оптимизации throughput: primary + secondary MongoTemplates

```
@Configuration
@EnableMongoRepositories(basePackages = "org.jugru.online.backend.repository",
                         mongoTemplateRef = "primMongoTemplate")
@EnableMongoAuditing
public class MongoPrimaryConfig {
  @Bean(name = "primMongoClient")
  public MongoClient primMongoClient(@Value("${spring.data.mongodb.uri}") String connection) {
      return MongoClients.create(connection);
  }
  @Primary
  @Bean(name = "primMongoDBFactory")
  public MongoDbFactory primMongoDatabaseFactory(@Qualifier("primMongoClient") MongoClient c) {
      return new SimpleMongoClientDbFactory(c, databaseName);
  }
  @Primary
  @Bean(name = "primMongoTemplate")
  public MongoTemplate primMongoTemplate(@Qualifier("primMongoDBFactory") MongoDbFactory f) {
      return new MongoTemplate(f);
  }
}
```

Оптимизации throughput: primary + secondary MongoTemplates

```
@Configuration
@EnableMongoRepositories(basePackages = "org.jugru.online.backend.repository",
                         mongoTemplateRef = "primMongoTemplate")
@EnableMongoAuditing
public class MongoPrimaryConfig {
  @Bean(name = "primMongoClient")
  public MongoClient primMongoClient(@Value("${spring.data.mongodb.uri}") String connection) {
      return MongoClients.create(connection);
  }
  @Primary
  @Bean(name = "primMongoDBFactory")
  public MongoDbFactory primMongoDatabaseFactory(@Qualifier("primMongoClient") MongoClient c) {
      return new SimpleMongoClientDbFactory(c, databaseName);
  }
  @Primary
  @Bean(name = "primMongoTemplate")
  public MongoTemplate primMongoTemplate(@Qualifier("primMongoDBFactory") MongoDbFactory f) {
      return new MongoTemplate(f);
  }
}
```

Оптимизации throughput: primary + secondary MongoTemplates

```
@Configuration
@EnableMongoRepositories(basePackages = "org.jugru.online.backend.repository",
                         mongoTemplateRef = "primMongoTemplate")
@EnableMongoAuditing
public class MongoPrimaryConfig {
  @Bean(name = "primMongoClient")
  public MongoClient primMongoClient(@Value("${spring.data.mongodb.uri}") String connection) {
      return MongoClients.create(connection);
  }
  @Primary
  @Bean(name = "primMongoDBFactory")
  public MongoDbFactory primMongoDatabaseFactory(@Qualifier("primMongoClient") MongoClient c) {
      return new SimpleMongoClientDbFactory(c, databaseName);
  }
  @Primary
  @Bean(name = "primMongoTemplate")
  public MongoTemplate primMongoTemplate(@Qualifier("primMongoDBFactory") MongoDbFactory f) {
      return new MongoTemplate(f);
  }
}
```

Оптимизации throughput: primary + secondary MongoTemplates

```
@Configuration
@EnableMongoRepositories(basePackages = "org.jugru.online.backend.repository",
                         mongoTemplateRef = "primMongoTemplate")
@EnableMongoAuditing
public class MongoPrimaryConfig {
  @Bean(name = "primMongoClient")
  public MongoClient primMongoClient(@Value("${spring.data.mongodb.uri}") String connection) {
      return MongoClients.create(connection);
  }
  @Primary
  @Bean(name = "primMongoDBFactory")
  public MongoDbFactory primMongoDatabaseFactory(@Qualifier("primMongoClient") MongoClient c) {
      return new SimpleMongoClientDbFactory(c, databaseName);
  }
  @Primary
  @Bean(name = "primMongoTemplate")
  public MongoTemplate primMongoTemplate(@Qualifier("primMongoDBFactory") MongoDbFactory f) {
      return new MongoTemplate(f);
  }
}
```

Оптимизации throughput: primary + secondary MongoTemplates

```
@Configuration
@EnableMongoRepositories(basePackages = "org.jugru.online.backend.repository.secondary",
                         mongoTemplateRef = "secMongoTemplate")
public class MongoSecondaryConfig {
  @Bean(name = "secMongoClient")
  public MongoClient secMongoClient(@Value("${spring.data.mongodb.uri}") String connection) {
      return MongoClients.create(connection);
  }
  @Bean(name = "secMongoDBFactory")
  public MongoDbFactory secMongoDatabaseFactory(@Qualifier("secMongoClient") MongoClient c) {
      return new SimpleMongoClientDbFactory(c, databaseName);
  }
  @Bean(name = "secMongoTemplate")
  public MongoTemplate secMongoTemplate(@Qualifier("secMongoDBFactory") MongoDbFactory f) {
      MongoTemplate mongoTemplate = new MongoTemplate(f);
      mongoTemplate.setReadPreference(ReadPreference.secondaryPreferred());
      return mongoTemplate;
  }
}
```

Оптимизации throughput: primary + secondary MongoTemplates

```
@Configuration
@EnableMongoRepositories(basePackages = "org.jugru.online.backend.repository.secondary",
                         mongoTemplateRef = "secMongoTemplate")
public class MongoSecondaryConfig {
  @Bean(name = "secMongoClient")
  public MongoClient secMongoClient(@Value("${spring.data.mongodb.uri}") String connection) {
      return MongoClients.create(connection);
  }
  @Bean(name = "secMongoDBFactory")
  public MongoDbFactory secMongoDatabaseFactory(@Qualifier("secMongoClient") MongoClient c) {
      return new SimpleMongoClientDbFactory(c, databaseName);
  }
  @Bean(name = "secMongoTemplate")
  public MongoTemplate secMongoTemplate(@Qualifier("secMongoDBFactory") MongoDbFactory f) {
      MongoTemplate mongoTemplate = new MongoTemplate(f);
      mongoTemplate.setReadPreference(ReadPreference.secondaryPreferred());
      return mongoTemplate;
  }
}
```

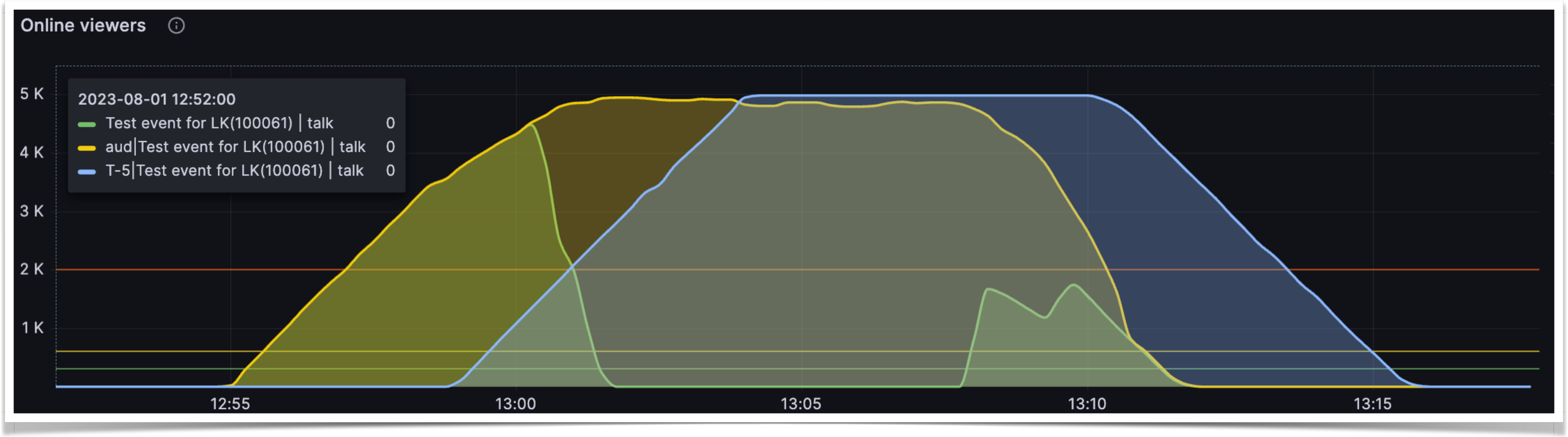
Оптимизации throughput: primary + secondary MongoTemplates

```
@Configuration
@EnableMongoRepositories(basePackages = "org.jugru.online.backend.repository.secondary",
                         mongoTemplateRef = "secMongoTemplate")
public class MongoSecondaryConfig {
  @Bean(name = "secMongoClient")
  public MongoClient secMongoClient(@Value("${spring.data.mongodb.uri}") String connection) {
      return MongoClients.create(connection);
  }
  @Bean(name = "secMongoDBFactory")
  public MongoDbFactory secMongoDatabaseFactory(@Qualifier("secMongoClient") MongoClient c) {
      return new SimpleMongoClientDbFactory(c, databaseName);
  }
  @Bean(name = "secMongoTemplate")
  public MongoTemplate secMongoTemplate(@Qualifier("secMongoDBFactory") MongoDbFactory f) {
      MongoTemplate mongoTemplate = new MongoTemplate(f);
      mongoTemplate.setReadPreference(ReadPreference.secondaryPreferred());
      return mongoTemplate;
  }
}
```

Оптимизации throughput: уменьшение нагрузки на managed mongodb

Оптимизации throughput: now_watching

Оптимизации throughput: now_watching



Оптимизации throughput: now_watching

Оптимизации throughput: now_watching

```
hazelcast.getMap(map$fact$i_m_watching$v2).setAsync(imw, imw);
```

Оптимизации throughput: now_watching

```
hazelcast.getMap(map$fact$i_m_watching$v2).setAsync(imw, imw);
```

```
com.hazelcast.map.IMap<K, V>

public abstract java.util.concurrent.CompletionStage<Void> setAsync(
    @Nonnull K key,
    @Nonnull V value
)
```

Asynchronously puts the given key and value. The entry lives forever. Similar to the put operation except that set doesn't return the old value, which is more efficient.

```
CompletionStage<Void> future = map.setAsync(key, value);
// do some other stuff, when ready wait for completion
future.toCompletableFuture().get();
```

`CompletionStage.toCompletableFuture().get()` will block until the actual map.set() operation completes. You can also register further computation stages to be invoked upon completion of the `CompletionStage` via any of `CompletionStage` methods:

```
CompletionStage<Void> future = map.setAsync("a", "b");
future.thenRunAsync(() -> System.out.println("Value is now set to b."));
```

Оптимизации throughput: now_watching

```
hazelcast.getMap(map$fact$i_m_watching$v2).setAsync(imw, imw);
```

```
com.hazelcast.map.IMap<K, V>

public abstract java.util.concurrent.CompletionStage<Void> setAsync(
    @Nonnull K key,
    @Nonnull V value
)
```

Asynchronously puts the given key and value. The entry lives forever. Similar to the put operation except that set doesn't return the old value, which is more efficient.

```
CompletionStage<Void> future = map.setAsync(key, value);
// do some other stuff, when ready wait for completion
future.toCompletableFuture().get();
```

`CompletionStage.toCompletableFuture().get()` will block until the actual map.set() operation completes. You can also register further computation stages to be invoked upon completion of the `CompletionStage` via any of `CompletionStage` methods:

```
CompletionStage<Void> future = map.setAsync("a", "b");
future.thenRunAsync(() -> System.out.println("Value is now set to b."));
```

```
hz.embedded@online-backend@production-online-mvp-backend-9f97ff7fb-5ftnd/
10.112.129.33.response-1: time in millis - 22224
```

Оптимизации throughput: now_watching

```
hazelcast.getMap(map$fact$i_m_watching$v2).setAsync(imw, imw);
```

```
com.hazelcast.map.IMap<K, V>

public abstract java.util.concurrent.CompletionStage<Void> setAsync(
    @Nonnull K key,
    @Nonnull V value
)
```

Asynchronously puts the given key and value. The entry lives forever. Similar to the put operation except that set doesn't return the old value, which is more efficient.

```
CompletionStage<Void> future = map.setAsync(key, value);
// do some other stuff, when ready wait for completion
future.toCompletableFuture().get();
```

`CompletionStage.toCompletableFuture().get()` will block until the actual map.set() operation completes. You can also register further computation stages to be invoked upon completion of the `CompletionStage` via any of `CompletionStage` methods:

```
CompletionStage<Void> future = map.setAsync("a", "b");
future.thenRunAsync(() -> System.out.println("Value is now set to b."));
```

```
hz.embedded@online-backend@production-online-mvp-backend-9f97ff7fb-5ftnd/
10.112.129.33.response-1: time in millis - 22224
```

Оптимизации throughput: now_watching

```
hazelcast.getMap(map$fact$i_m_watching$v2).setAsync(imw, imw);
```

```
com.hazelcast.map.IMap<K, V>

public abstract java.util.concurrent.CompletionStage<Void> setAsync(
    @Nonnull K key,
    @Nonnull V value
)
```

Asynchronously puts the given key and value. The entry lives forever. Similar to the put operation except that set doesn't return the old value, which is more efficient.

```
CompletionStage<Void> future = map.setAsync(key, value);
// do some other stuff, when ready wait for completion
future.toCompletableFuture().get();
```

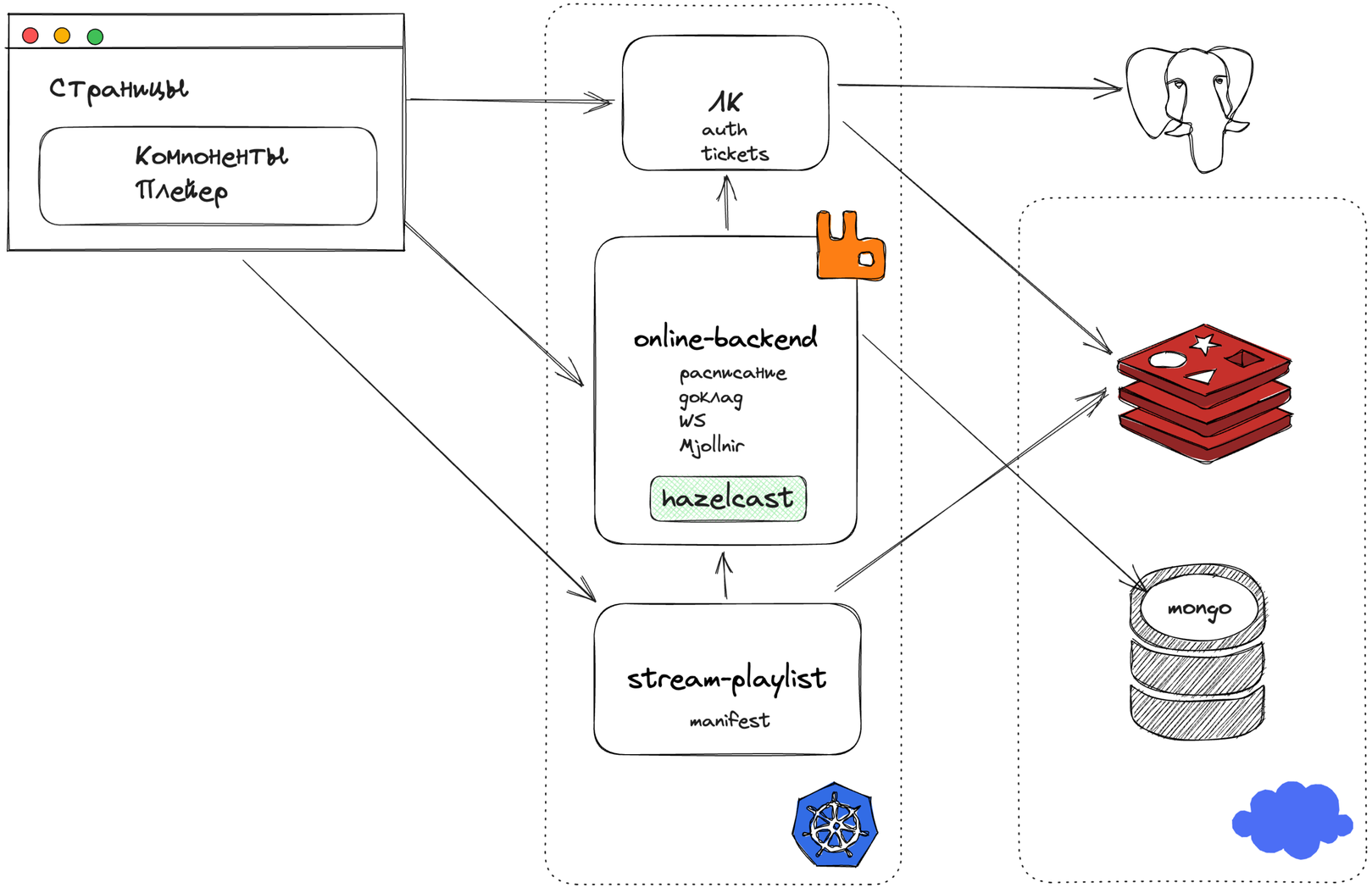
`CompletionStage.toCompletableFuture().get()` will block until the actual map.set() operation completes. You can also register further computation stages to be invoked upon completion of the `CompletionStage` via any of `CompletionStage` methods:

```
CompletionStage<Void> future = map.setAsync("a", "b");
future.thenRunAsync(() -> System.out.println("Value is now set to b."));
```

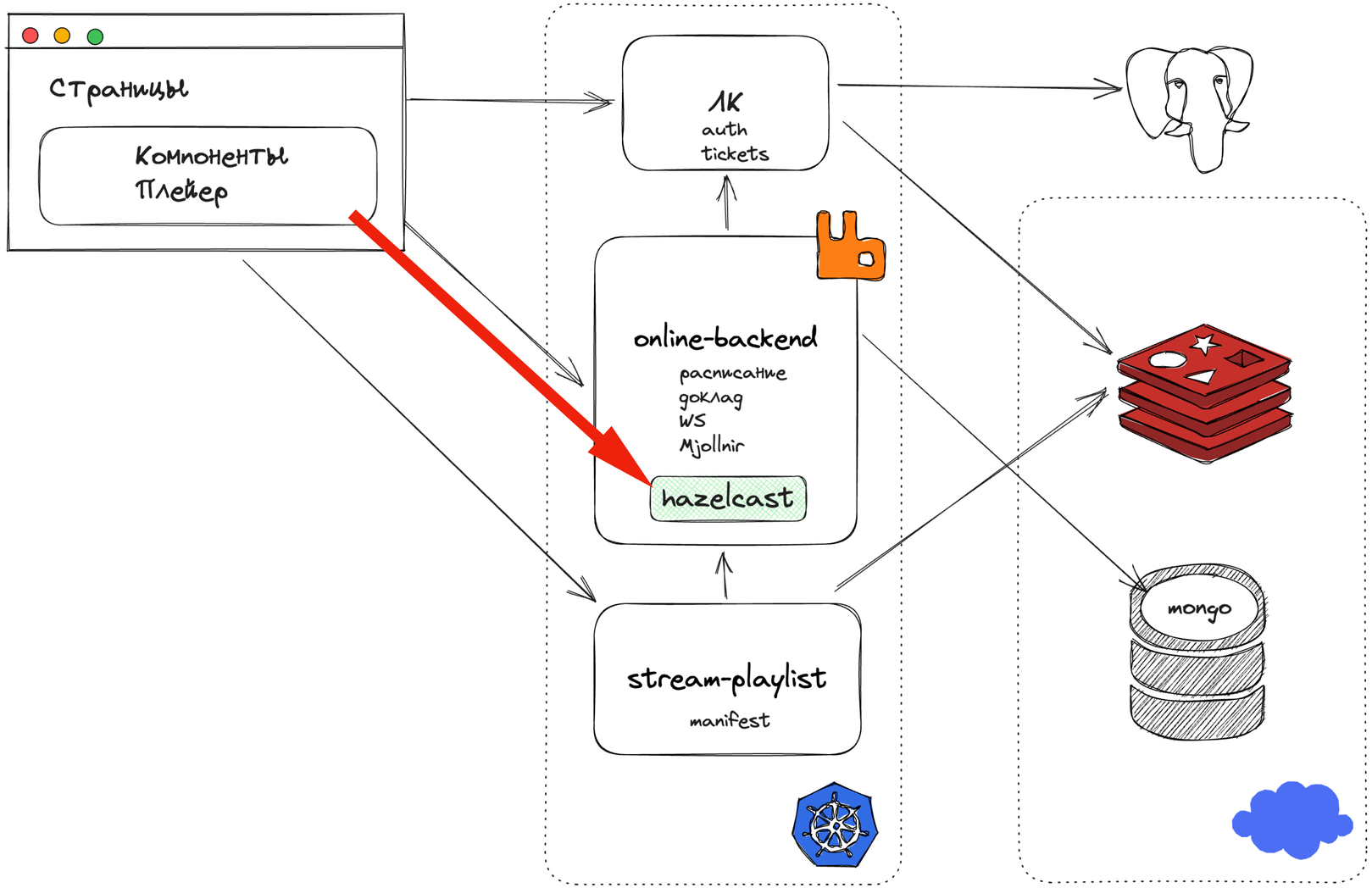
```
hz.embedded@online-backend@production-online-mvp-backend-9f97ff7fb-5ftnd/
10.112.129.33.response-1: time in millis - 22224
```

https://docs.hazelcast.com/imdg/4.1/performance/threading-model
https://docs.hazelcast.com/imdg/4.1/system-properties

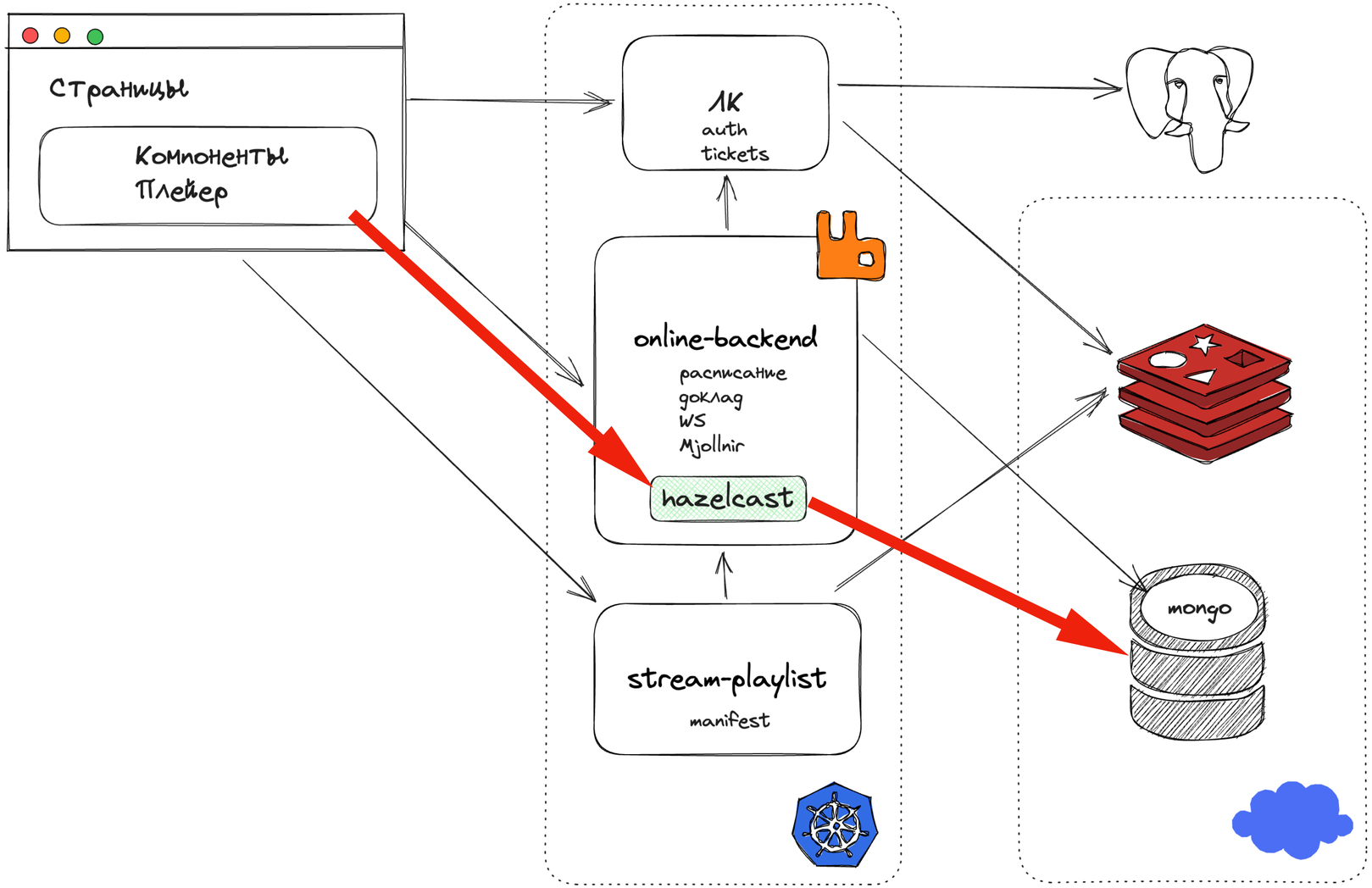
Оптимизации throughput: now_watching



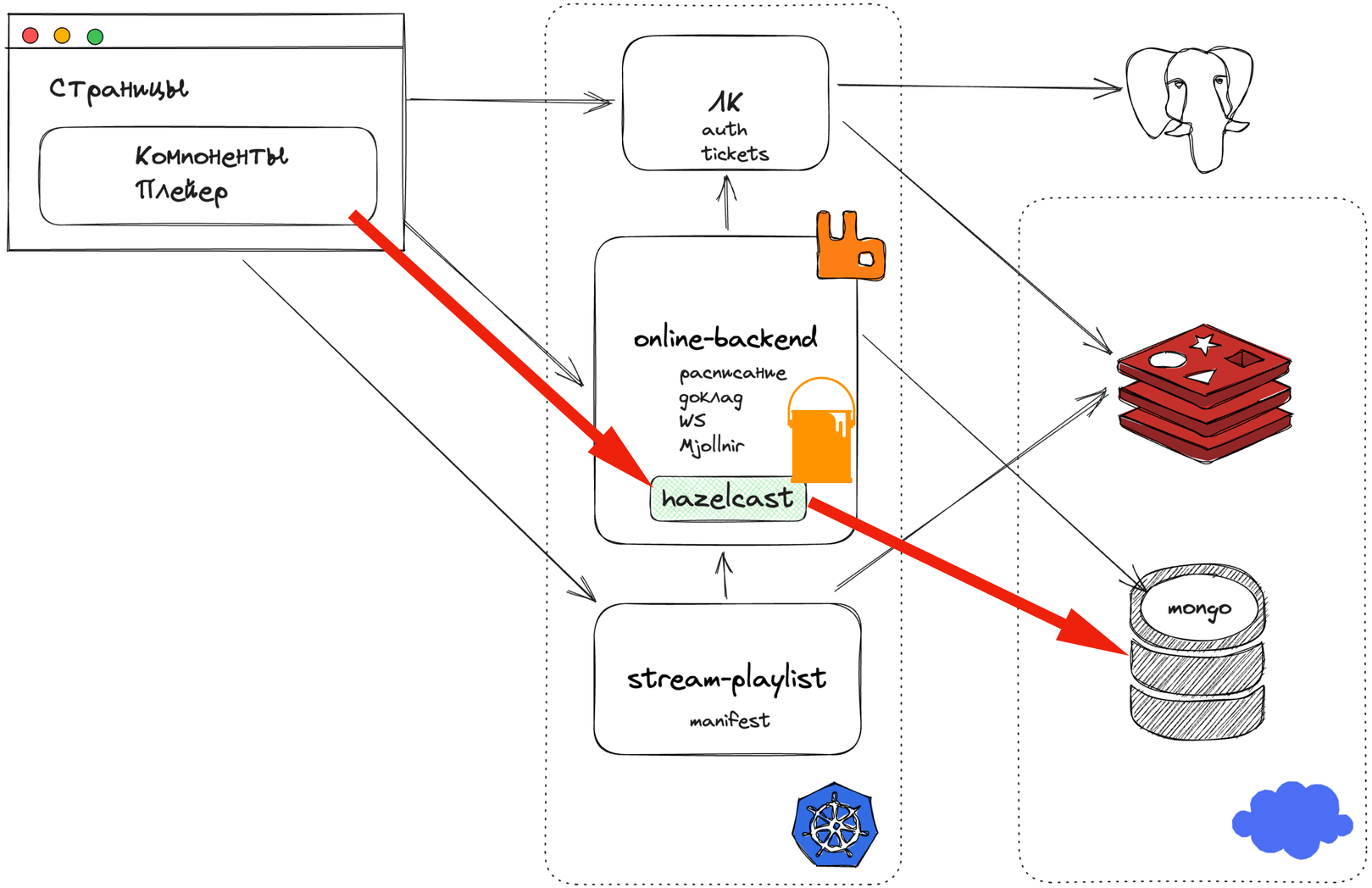
Оптимизации throughput: now_watching



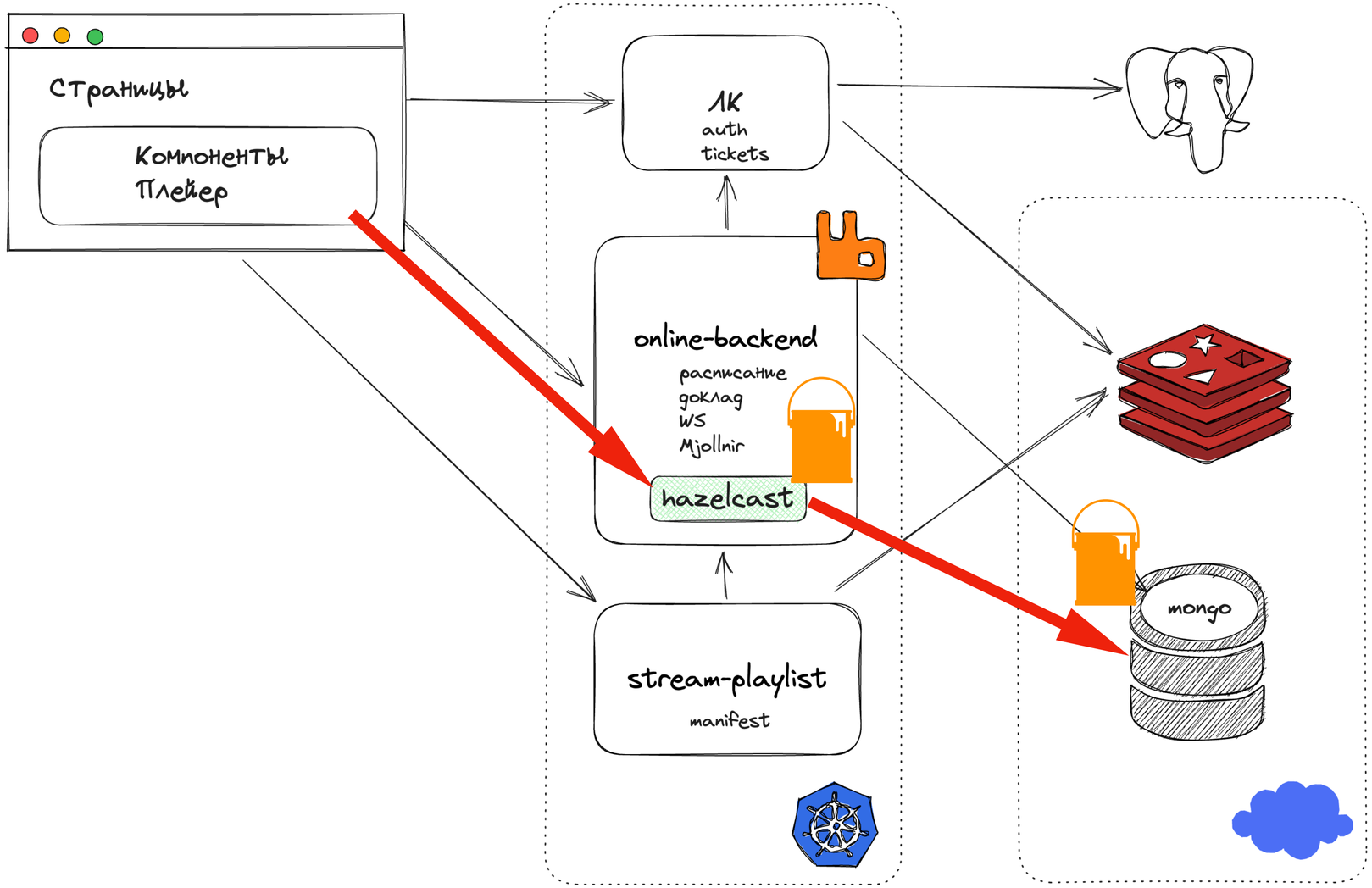
Оптимизации throughput: now_watching



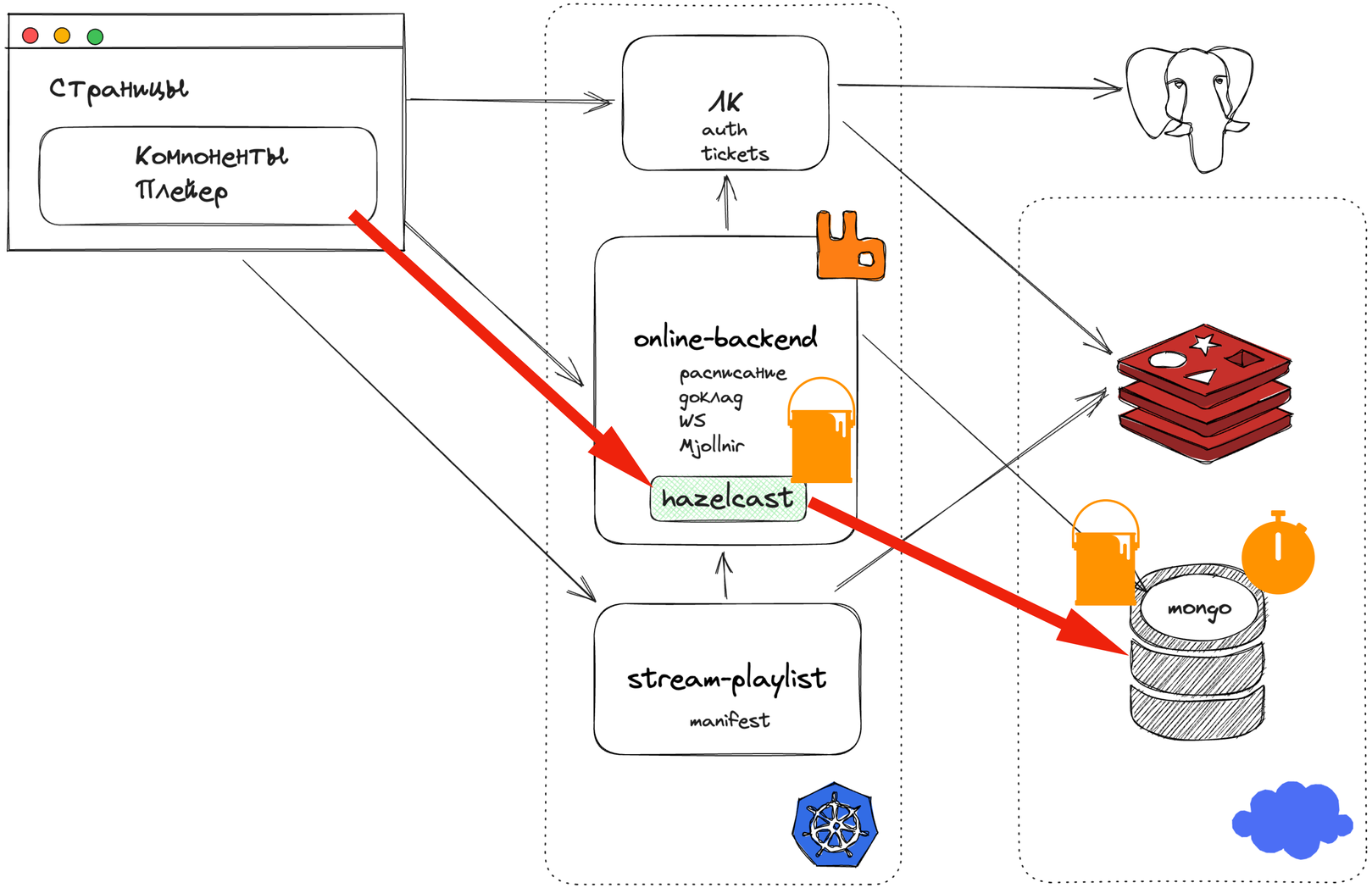
Оптимизации throughput: now_watching



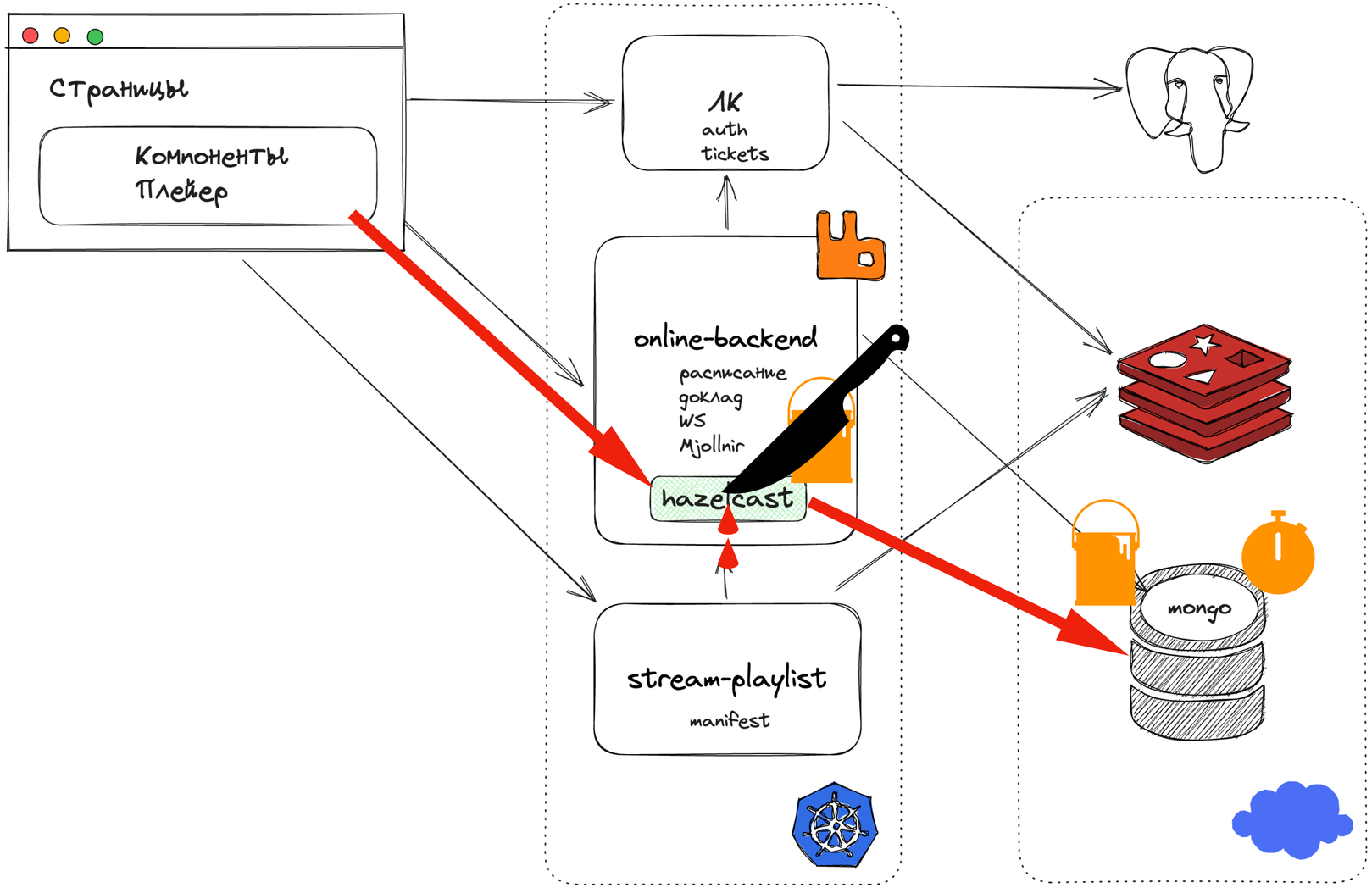
Оптимизации throughput: now_watching



Оптимизации throughput: now_watching



Оптимизации throughput: now_watching



Оптимизации throughput: now_watching

Оптимизации throughput: now_watching

```
LinkedBlockingQueue<ImWatchingLive20InMsg> unpersistedImws =
    new LinkedBlockingQueue<>(50_000);
...
```

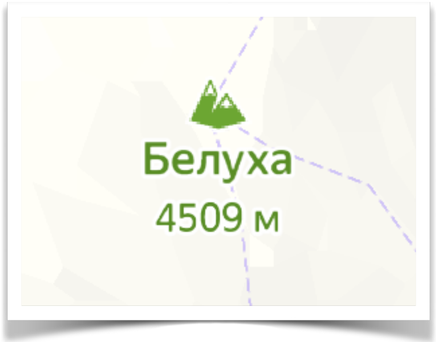
Оптимизации throughput: now_watching

```
LinkedBlockingQueue<ImWatchingLive20InMsg> unpersistedImws =
    new LinkedBlockingQueue<>(50_000);
...

//10K+ imws / 5 sec * 5 sec = 10K+
unpersistedImws.put(message);
...
```

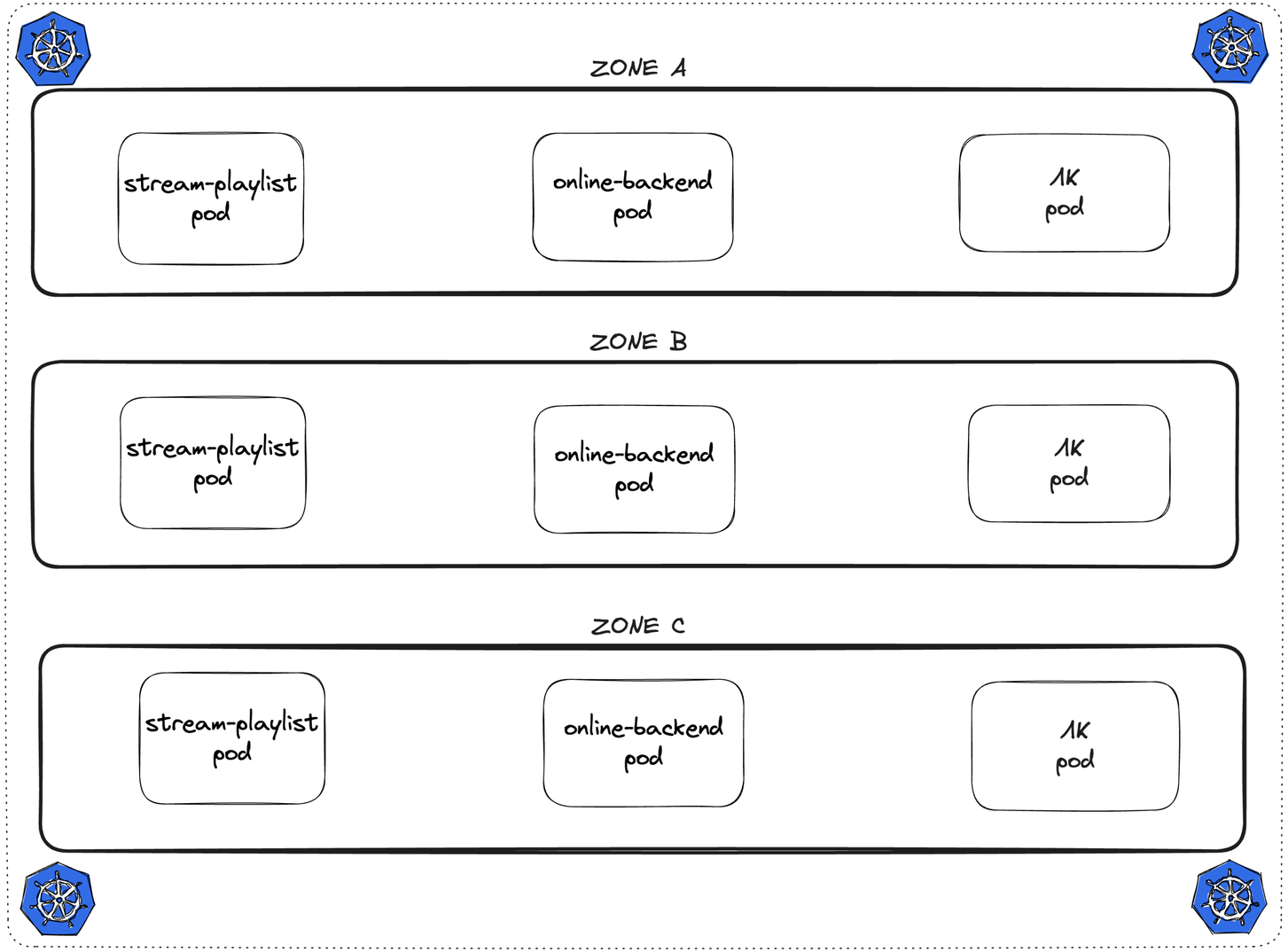
Оптимизации throughput: now_watching

```
LinkedBlockingQueue<ImWatchingLive20InMsg> unpersistedImws =
    new LinkedBlockingQueue<>(50_000);
...

//10K+ imws / 5 sec * 5 sec = 10K+
unpersistedImws.put(message);
...

@Scheduled(fixedDelayString = "${write.behind.imws.delay.millis}")
@Timed(longTask = true, value = "batch")
public void persistIamWatchingMsgs() {
    List<ImWatchingLive20InMsg> buffer = new ArrayList<>();
    int size = unpersistedImws.drainTo(buffer);
    imWatchingRepository.insert(buffer);
}
```

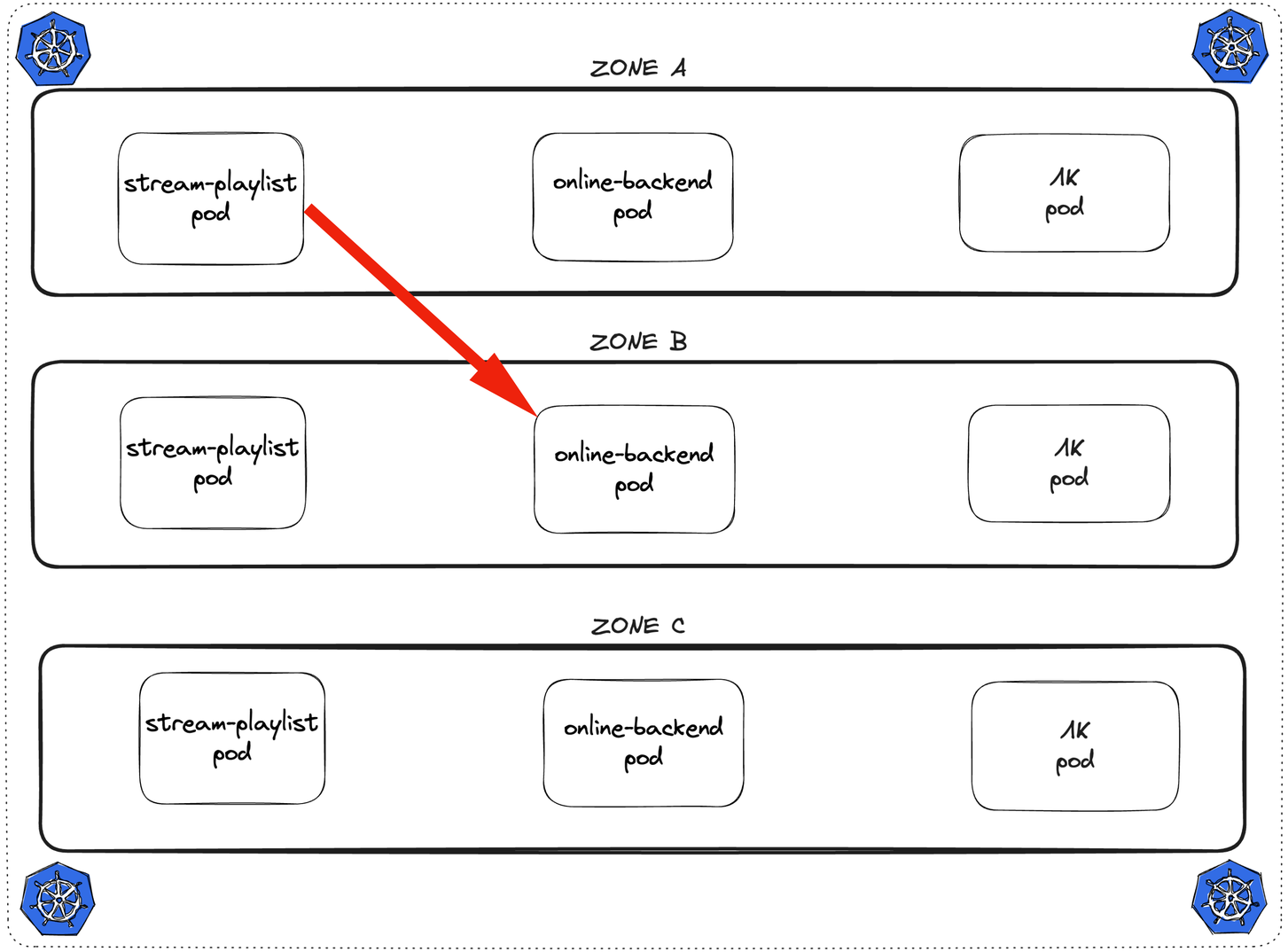
Оптимизации throughput: топологические хинты k8s



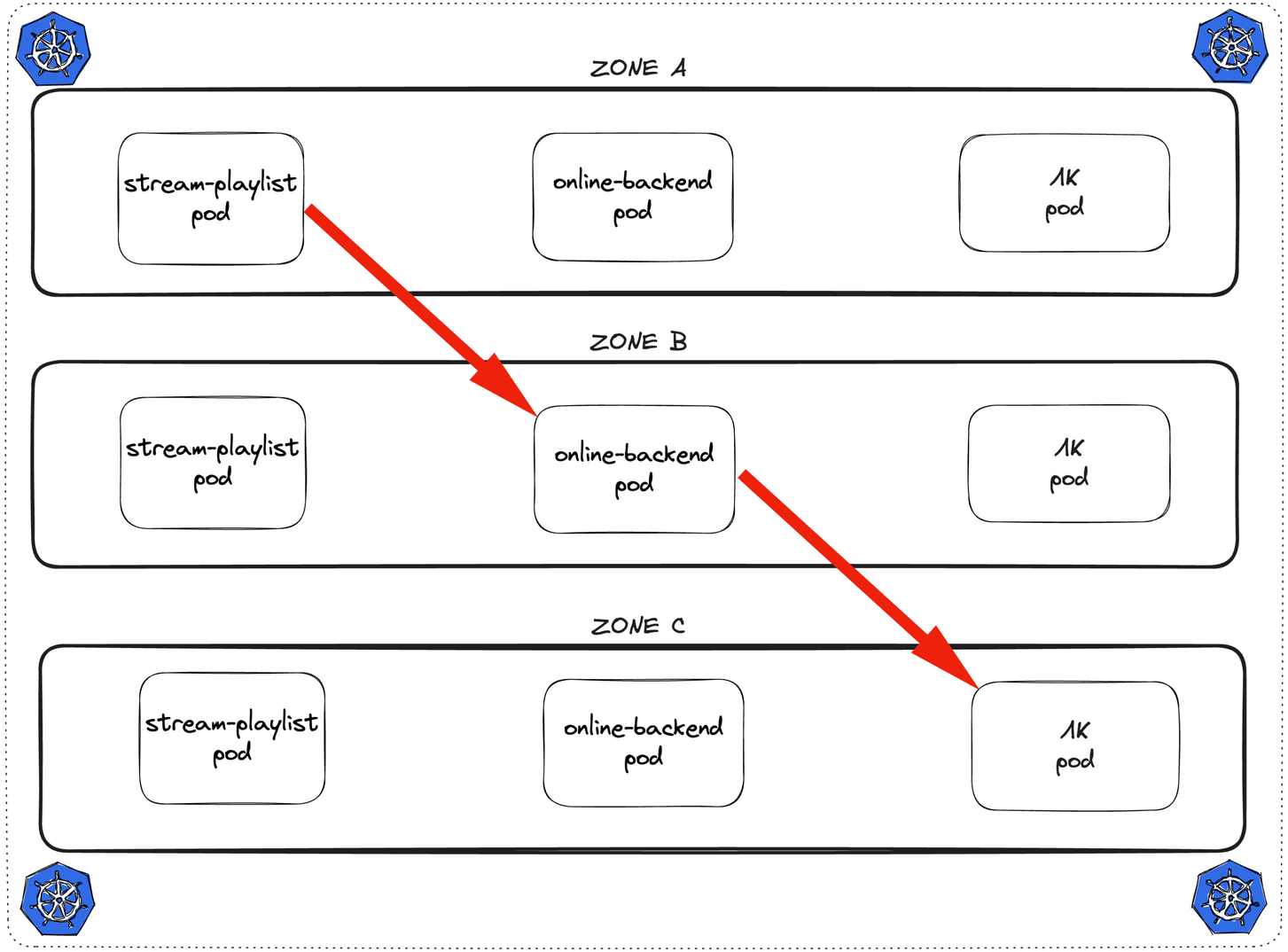
Оптимизации throughput: топологические хинты k8s



Оптимизации throughput: топологические хинты k8s



Оптимизации throughput: топологические хинты k8s



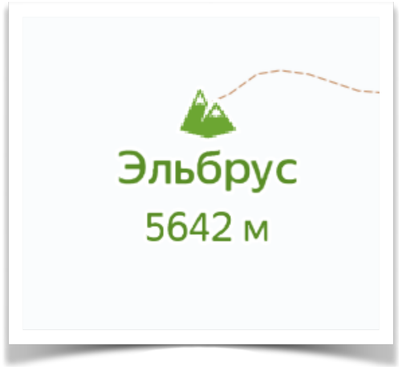
Оптимизации throughput: топологические хинты k8s

```
# до k8s 1,23
apiVersion: v1
kind: Service
metadata:
...
spec:
...
 topologyKeys:
    - "kubernetes.io/hostname"
    - "topology.kubernetes.io/zone"
    - "topology.kubernetes.io/region"
    - "*"
```

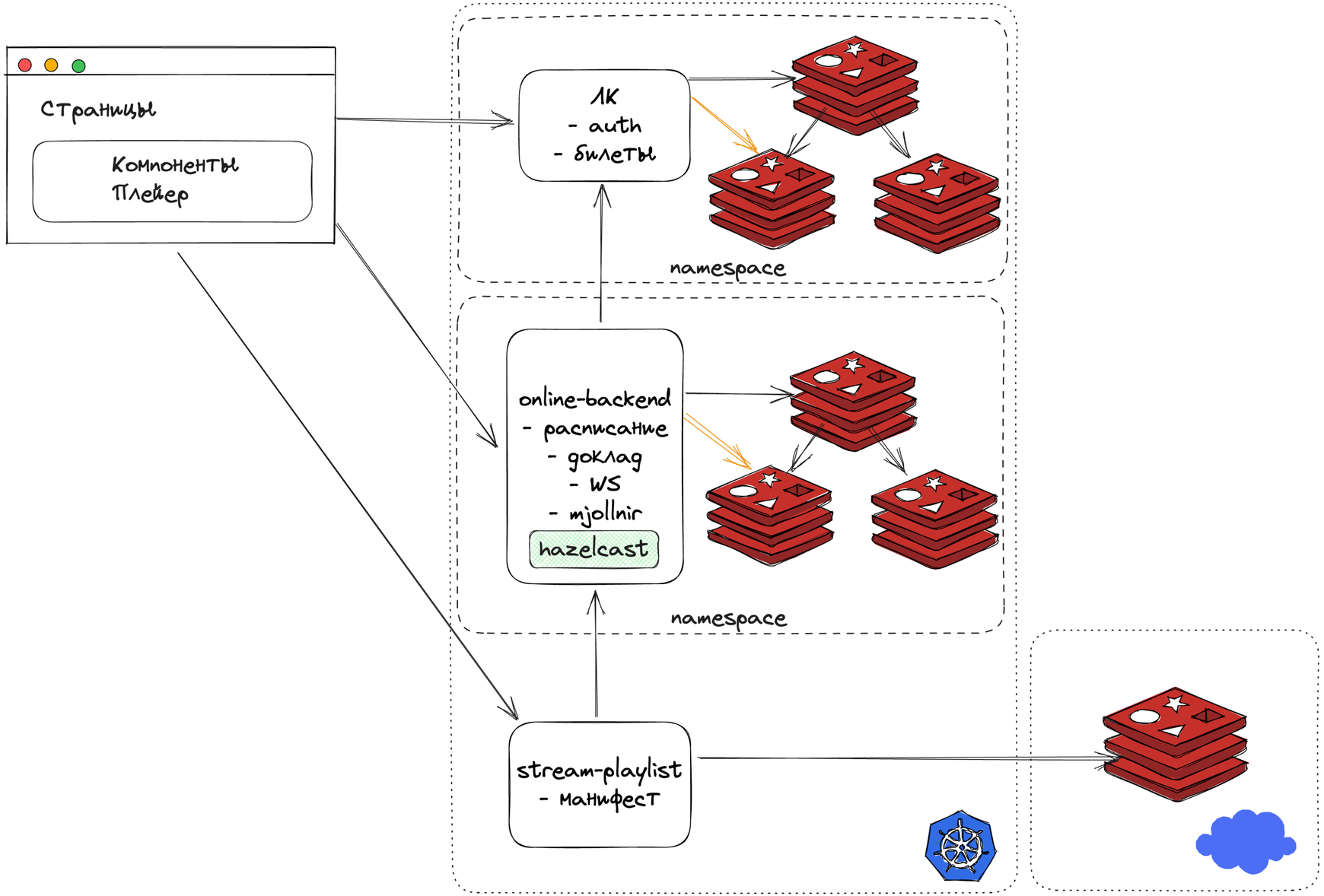
Оптимизации throughput: топологические хинты k8s

```yaml
# с k8s 1,24
apiVersion: v1
kind: Service
metadata:
  name: CI_ENVIRONMENT_SLUG-CI_PROJECT_NAME-service
  namespace: KUBE_NAMESPACE
  annotations:
    service.kubernetes.io/topology-aware-hints: auto
    ...
```

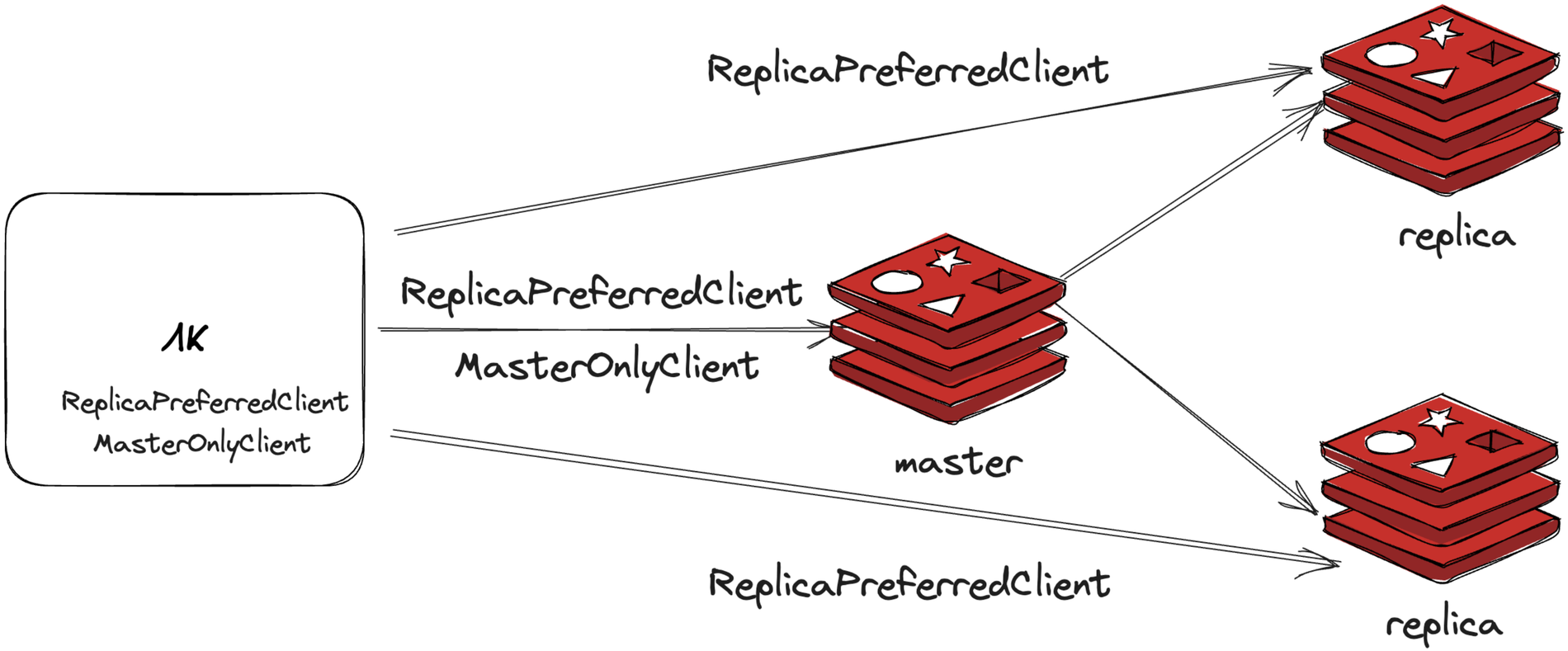
Оптимизации throughput: выделенные redis



Оптимизации throughput: выделенные redis



Оптимизации throughput: выделенные redis



Оптимизации throughput: выделенные redis

```
@Primary
@Bean("redisConnectionFactory")
public LettuceConnectionFactory masterReplicasRedisConnectionFactory(...) {
    LettuceClientConfiguration clientConfig =
        buildLettuceClientConfiguration(timeout, REPLICA_PREFERRED);
    RedisStaticMasterReplicaConfiguration serverConfig =
        new RedisStaticMasterReplicaConfiguration(masterHost, masterPort);
    serverConfig.setDatabase(database);
    serverConfig.setPassword(password);
    if (isNotBlank(replicasHost)) {
        serverConfig.addNode(replicasHost, replicasPort);
    }
    return new LettuceConnectionFactory(serverConfig, clientConfig);
}
private LettuceClientConfiguration buildLettuceClientConfiguration(long timeout, ReadFrom rf) {
    return LettuceClientConfiguration.builder()
            .clientName(format("lk-%s-redis-client", environment))
            .commandTimeout(Duration.of(timeout, MILLIS))
            .readFrom(rf)
            .build();
}
```

Оптимизации throughput: выделенные redis

```
@Primary
@Bean("redisConnectionFactory")
public LettuceConnectionFactory masterReplicasRedisConnectionFactory(...) {
    LettuceClientConfiguration clientConfig =
        buildLettuceClientConfiguration(timeout, REPLICA_PREFERRED);
    RedisStaticMasterReplicaConfiguration serverConfig =
        new RedisStaticMasterReplicaConfiguration(masterHost, masterPort);
    serverConfig.setDatabase(database);
    serverConfig.setPassword(password);
    if (isNotBlank(replicasHost)) {
        serverConfig.addNode(replicasHost, replicasPort);
    }
    return new LettuceConnectionFactory(serverConfig, clientConfig);
}
private LettuceClientConfiguration buildLettuceClientConfiguration(long timeout, ReadFrom rf) {
    return LettuceClientConfiguration.builder()
            .clientName(format("lk-%s-redis-client", environment))
            .commandTimeout(Duration.of(timeout, MILLIS))
            .readFrom(rf)
            .build();
}
```

Оптимизации throughput: выделенные redis

```
@Primary
@Bean("redisConnectionFactory")
public LettuceConnectionFactory masterReplicasRedisConnectionFactory(...) {
    LettuceClientConfiguration clientConfig =
        buildLettuceClientConfiguration(timeout, REPLICA_PREFERRED);
    RedisStaticMasterReplicaConfiguration serverConfig =
        new RedisStaticMasterReplicaConfiguration(masterHost, masterPort);
    serverConfig.setDatabase(database);
    serverConfig.setPassword(password);
    if (isNotBlank(replicasHost)) {
        serverConfig.addNode(replicasHost, replicasPort);
    }
    return new LettuceConnectionFactory(serverConfig, clientConfig);
}
private LettuceClientConfiguration buildLettuceClientConfiguration(long timeout, ReadFrom rf) {
    return LettuceClientConfiguration.builder()
            .clientName(format("lk-%s-redis-client", environment))
            .commandTimeout(Duration.of(timeout, MILLIS))
            .readFrom(rf)
            .build();
}
```

Оптимизации throughput: выделенные redis

```
@Primary
@Bean("redisConnectionFactory")
public LettuceConnectionFactory masterReplicasRedisConnectionFactory(...) {
    LettuceClientConfiguration clientConfig =
        buildLettuceClientConfiguration(timeout, REPLICA_PREFERRED);
    RedisStaticMasterReplicaConfiguration serverConfig =
        new RedisStaticMasterReplicaConfiguration(masterHost, masterPort);
    serverConfig.setDatabase(database);
    serverConfig.setPassword(password);
    if (isNotBlank(replicasHost)) {
        serverConfig.addNode(replicasHost, replicasPort);
    }
    return new LettuceConnectionFactory(serverConfig, clientConfig);
}
private LettuceClientConfiguration buildLettuceClientConfiguration(long timeout, ReadFrom rf) {
    return LettuceClientConfiguration.builder()
            .clientName(format("lk-%s-redis-client", environment))
            .commandTimeout(Duration.of(timeout, MILLIS))
            .readFrom(rf)
            .build();
}
```

Оптимизации throughput: выделенные redis

```
@Primary
@Bean("redisConnectionFactory")
public LettuceConnectionFactory masterReplicasRedisConnectionFactory(...) {
    LettuceClientConfiguration clientConfig =
        buildLettuceClientConfiguration(timeout, REPLICA_PREFERRED);
    RedisStaticMasterReplicaConfiguration serverConfig =
        new RedisStaticMasterReplicaConfiguration(masterHost, masterPort);
    serverConfig.setDatabase(database);
    serverConfig.setPassword(password);
    if (isNotBlank(replicasHost)) {
        serverConfig.addNode(replicasHost, replicasPort);
    }
    return new LettuceConnectionFactory(serverConfig, clientConfig);
}
private LettuceClientConfiguration buildLettuceClientConfiguration(long timeout, ReadFrom rf) {
    return LettuceClientConfiguration.builder()
            .clientName(format("lk-%s-redis-client", environment))
            .commandTimeout(Duration.of(timeout, MILLIS))
            .readFrom(rf)
            .build();
}
```

Оптимизации throughput: выделенные redis

```
@Bean("masterOnlyRedisConnectionFactory")
public LettuceConnectionFactory masterOnlyRedisConnectionFactory(...) {
    LettuceClientConfiguration clientConfig = buildLettuceClientConfiguration(timeout, MASTER);
    RedisStandaloneConfiguration serverConfig =
         new RedisStandaloneConfiguration(masterHost, masterPort);
    serverConfig.setDatabase(database);
    serverConfig.setPassword(password);
    return new LettuceConnectionFactory(serverConfig, clientConfig);
}
@Bean("masterOnlyRedisTemplate")
public RedisTemplate<String, Object> masterOnlyRedisTemplate(
            RedisConnectionFactory masterOnlyRedisConnectionFactory) {
    return buildRedisTemplate(masterOnlyRedisConnectionFactory);
}
@Bean
public BoundHashOperations<String, String, OAuth2Authentication> oauthCodeRedisStore(
            RedisTemplate<String, Object> masterOnlyRedisTemplate) {
    return masterOnlyRedisTemplate.boundHashOps(
             format(REDIS_STORE_KEY_PATTERN_OAUTH_AUTHORIZATION_CODE, environment));
}
```

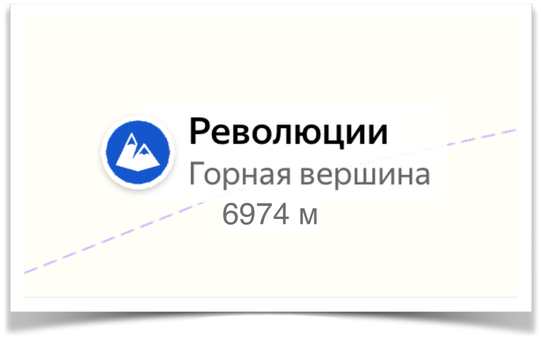
Оптимизации throughput: выделенные redis

```
@Bean("masterOnlyRedisConnectionFactory")
public LettuceConnectionFactory masterOnlyRedisConnectionFactory(...) {
    LettuceClientConfiguration clientConfig = buildLettuceClientConfiguration(timeout, MASTER);
    RedisStandaloneConfiguration serverConfig =
          new RedisStandaloneConfiguration(masterHost, masterPort);
    serverConfig.setDatabase(database);
    serverConfig.setPassword(password);
    return new LettuceConnectionFactory(serverConfig, clientConfig);
}
@Bean("masterOnlyRedisTemplate")
public RedisTemplate<String, Object> masterOnlyRedisTemplate(
            RedisConnectionFactory masterOnlyRedisConnectionFactory) {
    return buildRedisTemplate(masterOnlyRedisConnectionFactory);
}
@Bean
public BoundHashOperations<String, String, OAuth2Authentication> oauthCodeRedisStore(
            RedisTemplate<String, Object> masterOnlyRedisTemplate) {
    return masterOnlyRedisTemplate.boundHashOps(
             format(REDIS_STORE_KEY_PATTERN_OAUTH_AUTHORIZATION_CODE, environment));
}
```

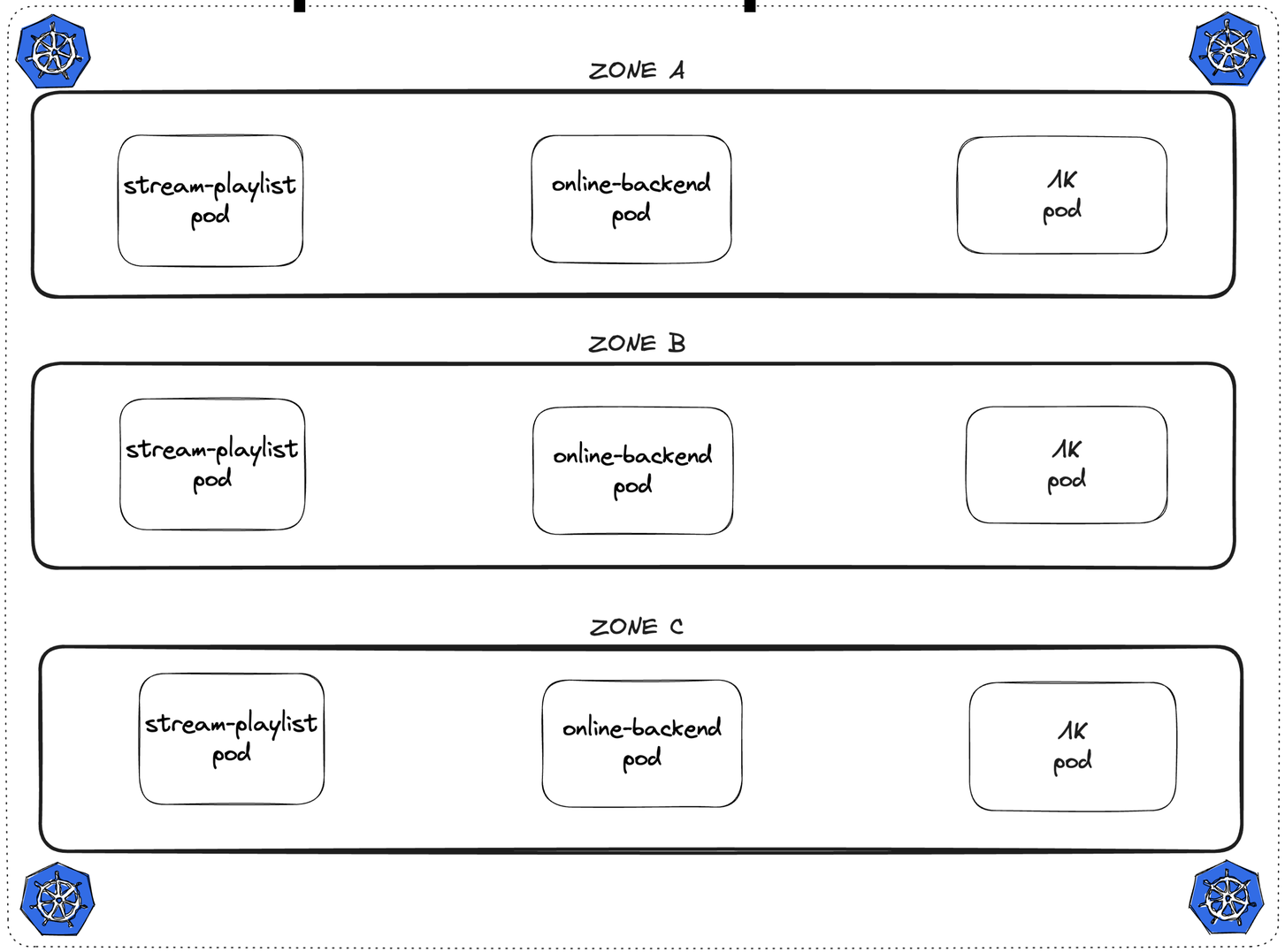
Оптимизации throughput: выделенные redis

```
@Bean("masterOnlyRedisConnectionFactory")
public LettuceConnectionFactory masterOnlyRedisConnectionFactory(...) {
    LettuceClientConfiguration clientConfig = buildLettuceClientConfiguration(timeout, MASTER);
    RedisStandaloneConfiguration serverConfig =
          new RedisStandaloneConfiguration(masterHost, masterPort);
    serverConfig.setDatabase(database);
    serverConfig.setPassword(password);
    return new LettuceConnectionFactory(serverConfig, clientConfig);
}
@Bean("masterOnlyRedisTemplate")
public RedisTemplate<String, Object> masterOnlyRedisTemplate(
            RedisConnectionFactory masterOnlyRedisConnectionFactory) {
    return buildRedisTemplate(masterOnlyRedisConnectionFactory);
}
@Bean
public BoundHashOperations<String, String, OAuth2Authentication> oauthCodeRedisStore(
            RedisTemplate<String, Object> masterOnlyRedisTemplate) {
    return masterOnlyRedisTemplate.boundHashOps(
            format(REDIS_STORE_KEY_PATTERN_OAUTH_AUTHORIZATION_CODE, environment));
}
```

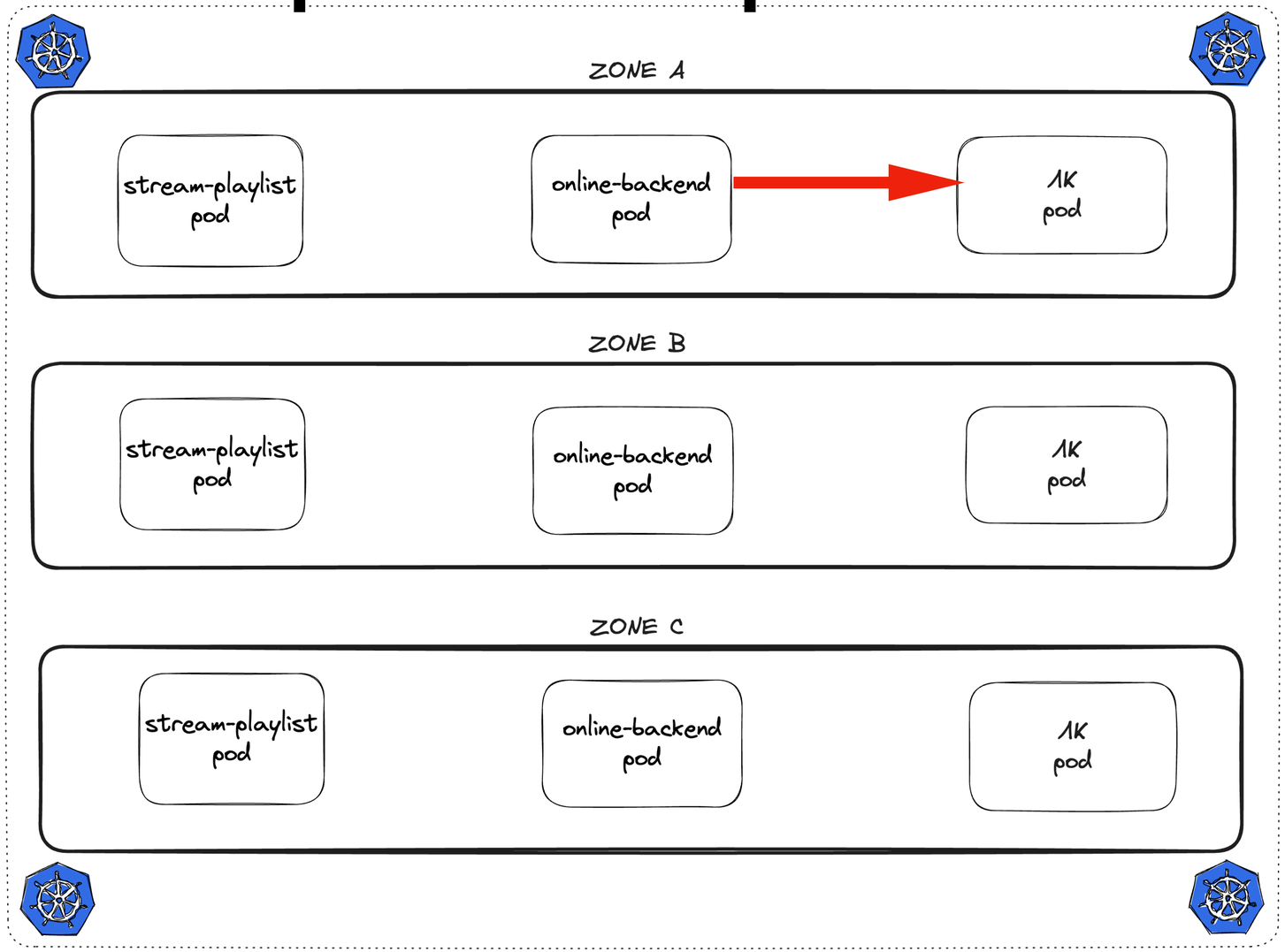
Оптимизации throughput: RestTemplate / Apache HttpClient



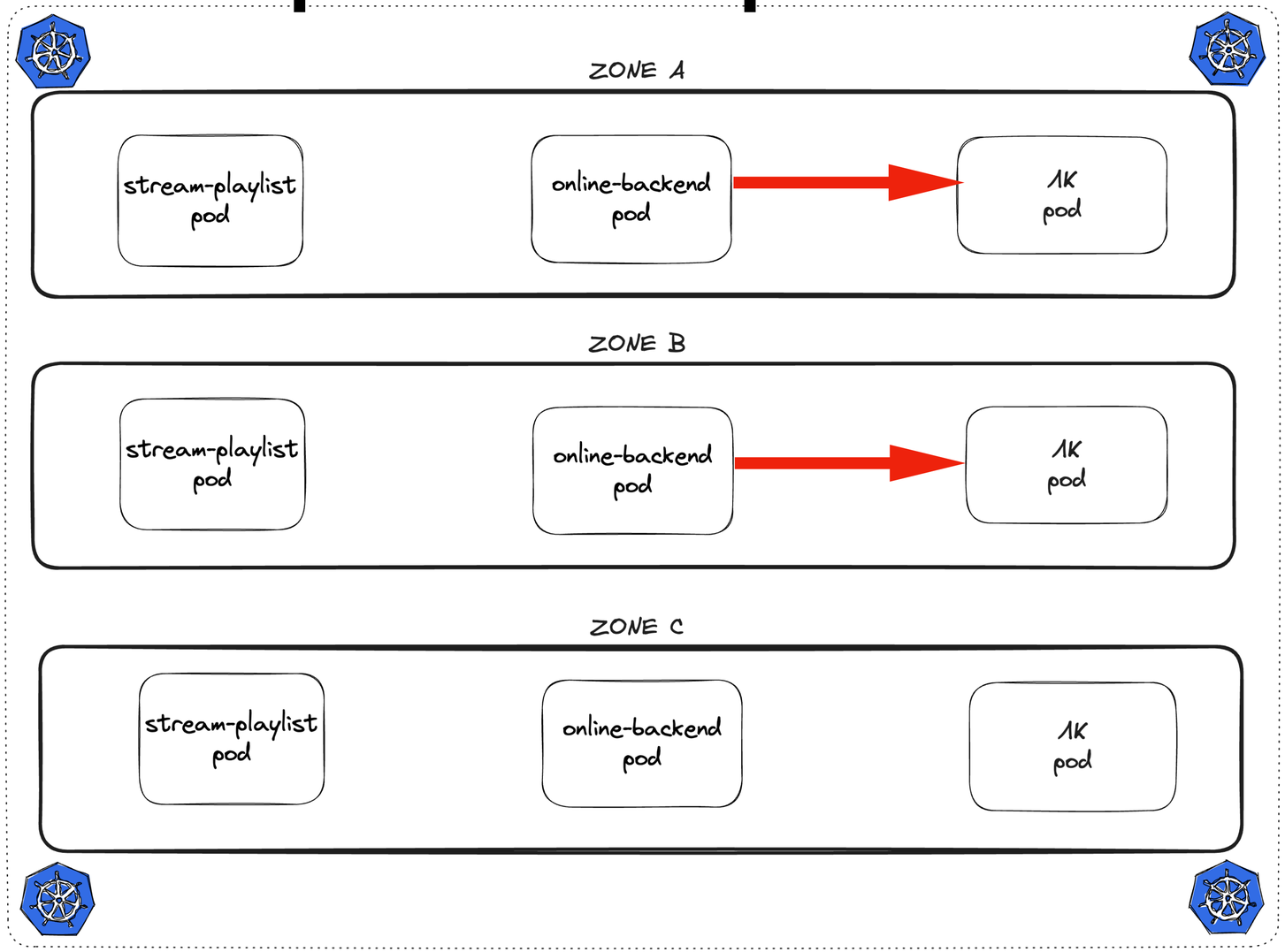
Оптимизации throughput: RestTemplate / Apache HttpClient



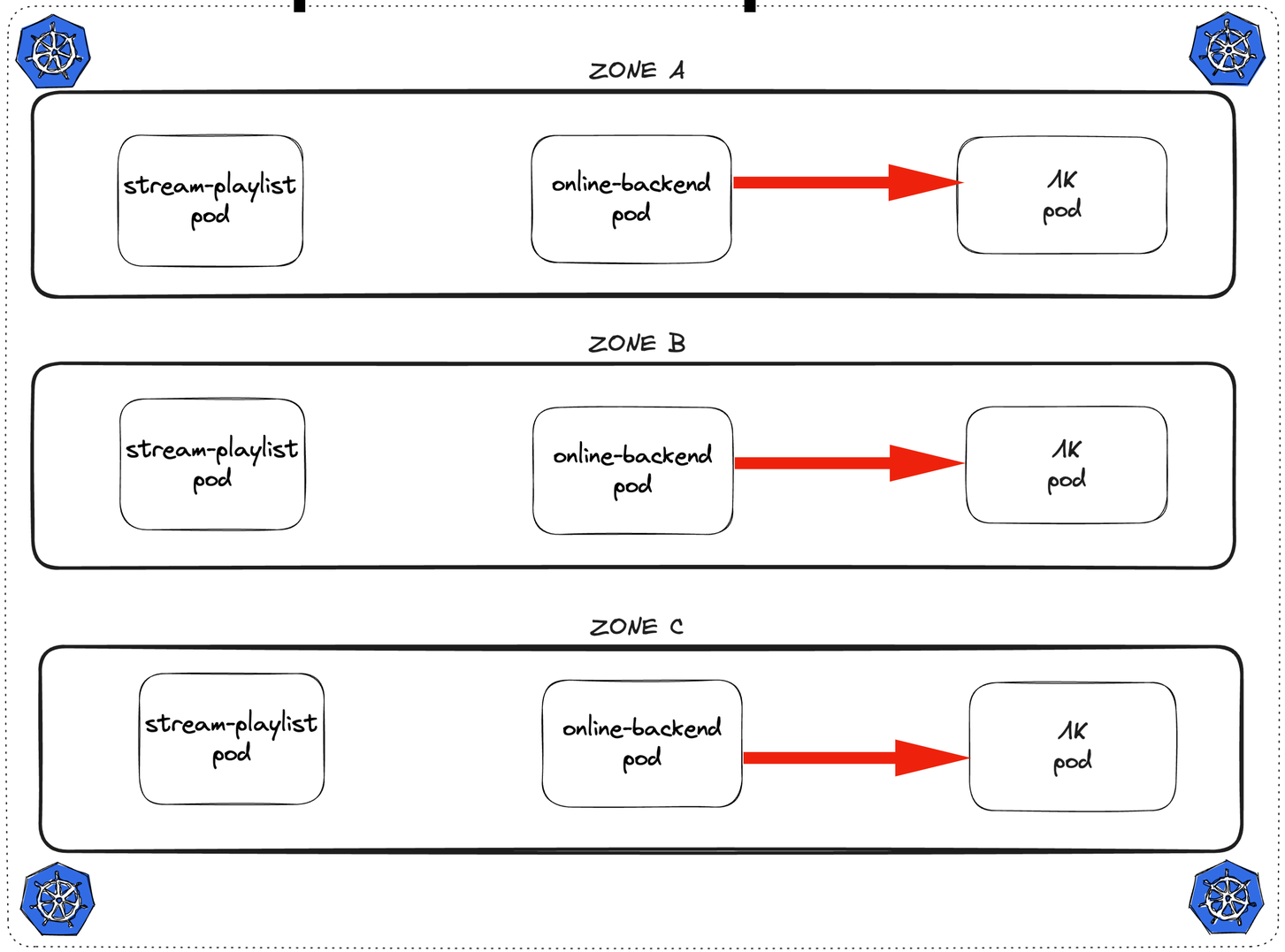
Оптимизации throughput: RestTemplate / Apache HttpClient



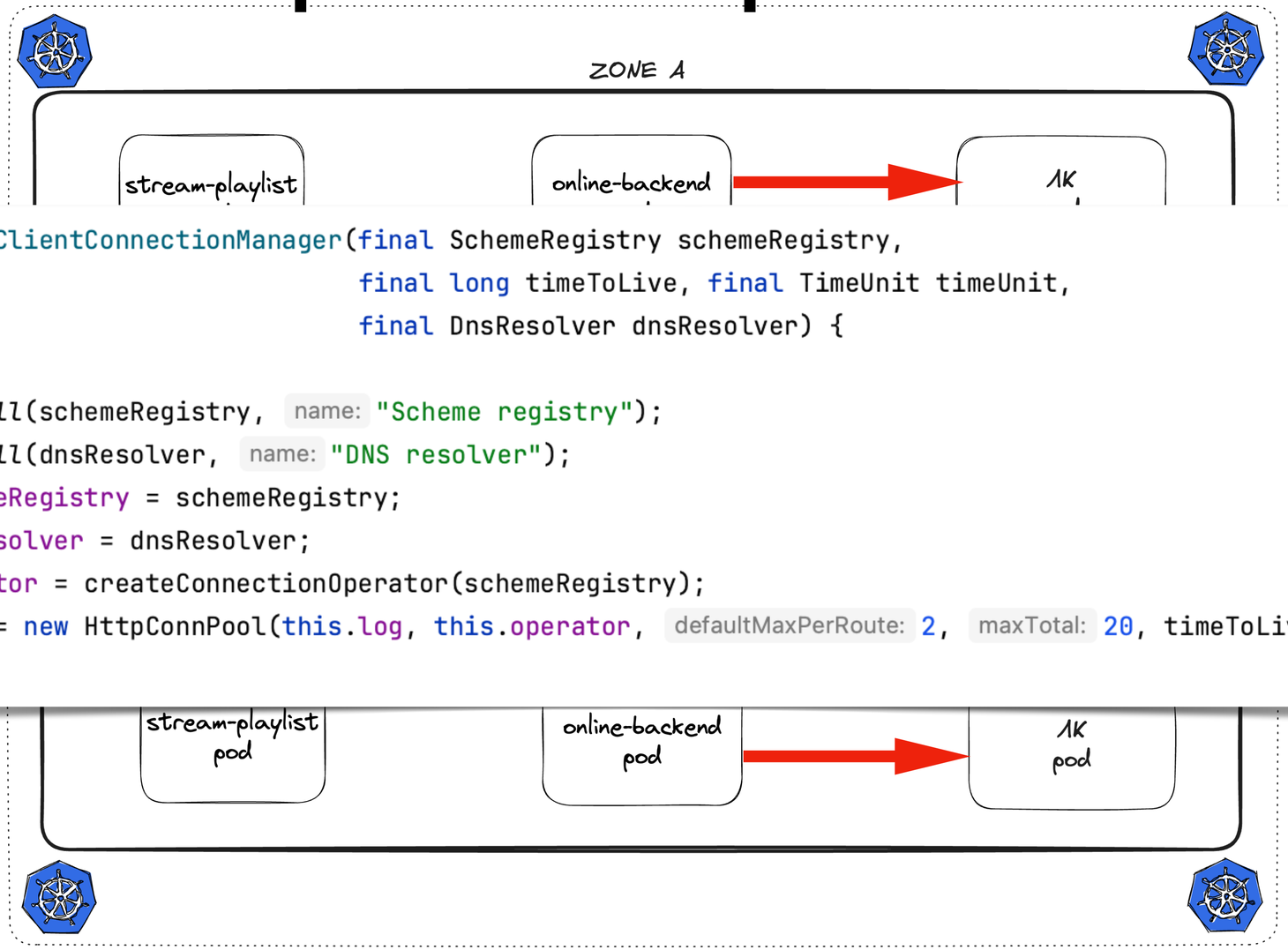
Оптимизации throughput: RestTemplate / Apache HttpClient



Оптимизации throughput: RestTemplate / Apache HttpClient

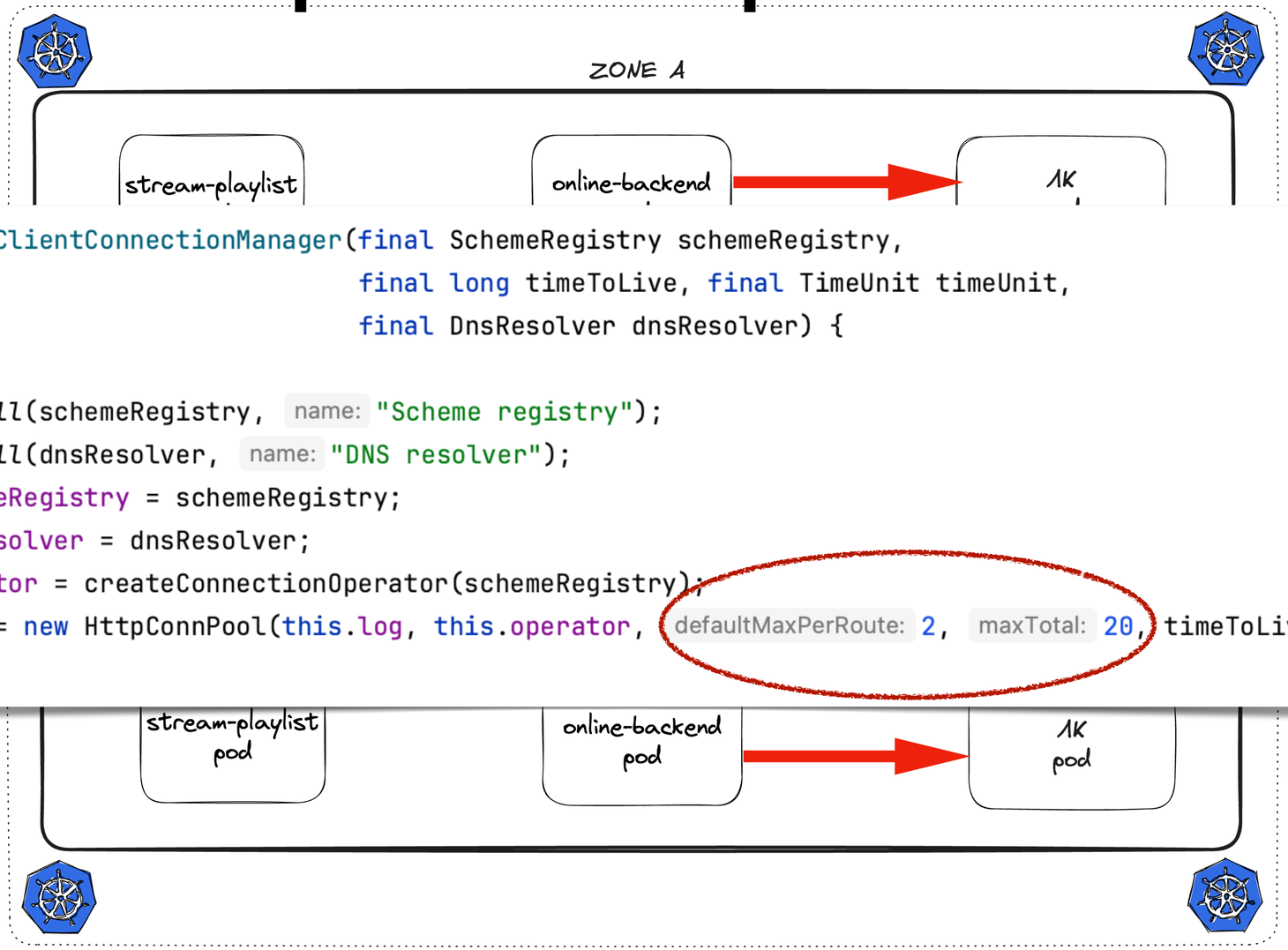


Оптимизации throughput: RestTemplate / Apache HttpClient




```
public PoolingClientConnectionManager(final SchemeRegistry schemeRegistry,
                                      final long timeToLive, final TimeUnit timeUnit,
                                      final DnsResolver dnsResolver) {
    super();
    Args.notNull(schemeRegistry, name: "Scheme registry");
    Args.notNull(dnsResolver, name: "DNS resolver");
    this.schemeRegistry = schemeRegistry;
    this.dnsResolver = dnsResolver;
    this.operator = createConnectionOperator(schemeRegistry);
    this.pool = new HttpConnPool(this.log, this.operator, defaultMaxPerRoute: 2, maxTotal: 20, timeToLive, timeUnit);
}
```

Оптимизации throughput: RestTemplate / Apache HttpClient




```
public PoolingClientConnectionManager(final SchemeRegistry schemeRegistry,
                                      final long timeToLive, final TimeUnit timeUnit,
                                      final DnsResolver dnsResolver) {
    super();
    Args.notNull(schemeRegistry, name: "Scheme registry");
    Args.notNull(dnsResolver, name: "DNS resolver");
    this.schemeRegistry = schemeRegistry;
    this.dnsResolver = dnsResolver;
    this.operator = createConnectionOperator(schemeRegistry);
    this.pool = new HttpConnPool(this.log, this.operator, defaultMaxPerRoute: 2, maxTotal: 20, timeToLive, timeUnit);
}
```

Оптимизации throughput: RestTemplate / Apache HttpClient

```
@Bean
public CloseableHttpClient lkApiHttpClient() {
    RequestConfig requestConfig = RequestConfig.custom()
            .setConnectTimeout(connectTimeout)
            .setConnectionRequestTimeout(connectionRequestTimeout)
            .setSocketTimeout(readTimeout).build();
    return HttpClients.custom()
            .setDefaultRequestConfig(requestConfig)
            .setConnectionManager(lkApiPoolingConnectionManager())
            .setKeepAliveStrategy(lkApiConnectionKeepAliveStrategy())
            .build();
}
@Bean
public PoolingHttpClientConnectionManager lkApiPoolingConnectionManager() {
    PoolingHttpClientConnectionManager poolingConnectionManager =
             new PoolingHttpClientConnectionManager();
    poolingConnectionManager.setMaxTotal(maxConnectionsTotal);
    poolingConnectionManager.setDefaultMaxPerRoute(maxConnectionsPerRoute);//default is 2!
    return poolingConnectionManager;
}
```

Оптимизации throughput: RestTemplate / Apache HttpClient

```
@Bean
public CloseableHttpClient lkApiHttpClient() {
    RequestConfig requestConfig = RequestConfig.custom()
            .setConnectTimeout(connectTimeout)
            .setConnectionRequestTimeout(connectionRequestTimeout)
            .setSocketTimeout(readTimeout).build();
    return HttpClients.custom()
            .setDefaultRequestConfig(requestConfig)
            .setConnectionManager(lkApiPoolingConnectionManager())
            .setKeepAliveStrategy(lkApiConnectionKeepAliveStrategy())
            .build();
}
@Bean
public PoolingHttpClientConnectionManager lkApiPoolingConnectionManager() {
    PoolingHttpClientConnectionManager poolingConnectionManager =
             new PoolingHttpClientConnectionManager();
    poolingConnectionManager.setMaxTotal(maxConnectionsTotal);
    poolingConnectionManager.setDefaultMaxPerRoute(maxConnectionsPerRoute);//default is 2!
    return poolingConnectionManager;
}
```

Оптимизации throughput: RestTemplate / Apache HttpClient

```
@Bean
public CloseableHttpClient lkApiHttpClient() {
    RequestConfig requestConfig = RequestConfig.custom()
            .setConnectTimeout(connectTimeout)
            .setConnectionRequestTimeout(connectionRequestTimeout)
            .setSocketTimeout(readTimeout).build();
    return HttpClients.custom()
            .setDefaultRequestConfig(requestConfig)
            .setConnectionManager(lkApiPoolingConnectionManager())
            .setKeepAliveStrategy(lkApiConnectionKeepAliveStrategy())
            .build();
}
@Bean
public PoolingHttpClientConnectionManager lkApiPoolingConnectionManager() {
    PoolingHttpClientConnectionManager poolingConnectionManager =
             new PoolingHttpClientConnectionManager();
    poolingConnectionManager.setMaxTotal(maxConnectionsTotal);
    poolingConnectionManager.setDefaultMaxPerRoute(maxConnectionsPerRoute);//default is 2!
    return poolingConnectionManager;
}
```

Оптимизации throughput: RestTemplate / Apache HttpClient

```
@Bean
public CloseableHttpClient lkApiHttpClient() {
    RequestConfig requestConfig = RequestConfig.custom()
            .setConnectTimeout(connectTimeout)
            .setConnectionRequestTimeout(connectionRequestTimeout)
            .setSocketTimeout(readTimeout).build();
    return HttpClients.custom()
            .setDefaultRequestConfig(requestConfig)
            .setConnectionManager(lkApiPoolingConnectionManager())
            .setKeepAliveStrategy(lkApiConnectionKeepAliveStrategy())
            .build();
}
@Bean
public PoolingHttpClientConnectionManager lkApiPoolingConnectionManager() {
    PoolingHttpClientConnectionManager poolingConnectionManager =
             new PoolingHttpClientConnectionManager();
    poolingConnectionManager.setMaxTotal(maxConnectionsTotal);
    poolingConnectionManager.setDefaultMaxPerRoute(maxConnectionsPerRoute);//default is 2!
    return poolingConnectionManager;
}
```

Оптимизации throughput: RestTemplate / Apache HttpClient

```
@Bean
public HttpComponentsClientHttpRequestFactory lkApiClientHttpRequestFactory(
            CloseableHttpClient lkApiHttpClient) {
    HttpComponentsClientHttpRequestFactory clientHttpRequestFactory
            = new HttpComponentsClientHttpRequestFactory();
    clientHttpRequestFactory.setHttpClient(lkApiHttpClient);
    return clientHttpRequestFactory;
}
@Bean
public RestTemplate lkApiRestTemplate(RestTemplateBuilder builder,
    HttpComponentsClientHttpRequestFactory lkApiClientHttpRequestFactory) {
    return builder
            ...
            .requestFactory(() -> lkApiClientHttpRequestFactory)
            .build();
}
```

Оптимизации throughput: RestTemplate / Apache HttpClient

```
@Bean
public HttpComponentsClientHttpRequestFactory lkApiClientHttpRequestFactory(
            CloseableHttpClient lkApiHttpClient) {
    HttpComponentsClientHttpRequestFactory clientHttpRequestFactory
            = new HttpComponentsClientHttpRequestFactory();
    clientHttpRequestFactory.setHttpClient(lkApiHttpClient);
    return clientHttpRequestFactory;
}
@Bean
public RestTemplate lkApiRestTemplate(RestTemplateBuilder builder,
    HttpComponentsClientHttpRequestFactory lkApiClientHttpRequestFactory) {
    return builder
            ...
            .requestFactory(() -> lkApiClientHttpRequestFactory)
            .build();
}
```

Оптимизации throughput: RestTemplate / Apache HttpClient

```
@Bean
public HttpComponentsClientHttpRequestFactory lkApiClientHttpRequestFactory(
            CloseableHttpClient lkApiHttpClient) {
    HttpComponentsClientHttpRequestFactory clientHttpRequestFactory
            = new HttpComponentsClientHttpRequestFactory();
    clientHttpRequestFactory.setHttpClient(lkApiHttpClient);
    return clientHttpRequestFactory;
}
@Bean
public RestTemplate lkApiRestTemplate(RestTemplateBuilder builder,
    HttpComponentsClientHttpRequestFactory lkApiClientHttpRequestFactory) {
    return builder
            ...
            .requestFactory(() -> lkApiClientHttpRequestFactory)
            .build();
}
```

Оптимизации throughput: RestTemplate / Apache HttpClient

```
@Bean
public HttpComponentsClientHttpRequestFactory lkApiClientHttpRequestFactory(
            CloseableHttpClient lkApiHttpClient) {
    HttpComponentsClientHttpRequestFactory clientHttpRequestFactory
            = new HttpComponentsClientHttpRequestFactory();
    clientHttpRequestFactory.setHttpClient(lkApiHttpClient);
    return clientHttpRequestFactory;
}
@Bean
public RestTemplate lkApiRestTemplate(RestTemplateBuilder builder,
    HttpComponentsClientHttpRequestFactory lkApiClientHttpRequestFactory) {
    return builder
            ...
            .requestFactory(() -> lkApiClientHttpRequestFactory)
            .build();
}
```

Оптимизации throughput: RestTemplate / Apache HttpClient

```java
@Bean
public HttpComponentsClientHttpRequestFactory lkApiClientHttpRequestFactory(
            CloseableHttpClient lkApiHttpClient) {
    HttpComponentsClientHttpRequestFactory clientHttpRequestFactory
            = new HttpComponentsClientHttpRequestFactory();
    clientHttpRequestFactory.setHttpClient(lkApiHttpClient);
    return clientHttpRequestFactory;
}
@Bean
public RestTemplate lkApiRestTemplate(RestTemplateBuilder builder,
    HttpComponentsClientHttpRequestFactory lkApiClientHttpRequestFactory) {
    return builder
            ...
            .requestFactory(() -> lkApiClientHttpRequestFactory)
            .build();
}
```

Глава 7

1. Погружение в контекст онлайна JUG Ru Group, задачи и цели на 2023 год
2. Знакомство с основными компонентами онлайна
3. Выбор инструмента нагрузочного тестирования онлайна (спойлер: gatling)
4. Построение базового нагрузочного сценария, заведение нагрузочных ботов
5. Выбор инструмента деплоя и запуска нагрузки (спойлер: nanocloud + 9 VM)
6. Основные оптимизации для увеличения пропускной способности онлайна
7. Выводы

Выводы

Выводы

1. Прямо сейчас работы снова ведутся на высоте Эльбруса, хотя еще недавно почти добрались до высоты пика Революции

Выводы

1. Прямо сейчас работы снова ведутся на высоте Эльбруса, хотя еще недавно почти добрались до высоты пика Революции

2. Gatling Java DSL - теперь мой выбор! Киллер фичи gatling: feed пачками + indexedELProperty, noReport + reportOnly

Выводы

1. Прямо сейчас работы снова ведутся на высоте Эльбруса, хотя еще недавно почти добрались до высоты пика Революции

2. Gatling Java DSL - теперь мой выбор! Киллер фичи gatling: feed пачками + indexedELProperty, noReport + reportOnly

3. KISS! Не кубером единым! Gatling, nanocloud, 9 виртуалок и самодельный “Gatling Enterprise” готов к использованию

Поваренная книга: оптимизации которые помогли приблизится к 10K

1. Настройка дефолтных потоков очереди в tomcat в соответствии с коннекшенами бд
2. Переход от агрегаций в hazelcast на агрегации в mongodb
3. Уменьшение нагрузки на mongodb: 2 темплейта и прицельное чтение из secondary
4. Уменьшение нагрузки на redis: 2 темплейта и прицельное чтение из master
5. Увеличение defaultMaxPerRoute в Apache HttpClient для RestTemplate
6. Переход с hazelcast IMap + MapStore на LinkedBlockingQueue + @Scheduled
7. Настройка probes для rabbit в k8s
8. Зипование манифестов на отдельном nginx для уменьшения внутреннего трафика
9. Уменьшение походов между зонами доступности в k8s через топологические хинты
10. Выделенные redis в k8s вместо общего managed redis

Спасибо за внимание
и до встречи на дискуссии!

@vlakra
vladimir.krasilschik@gmail.com