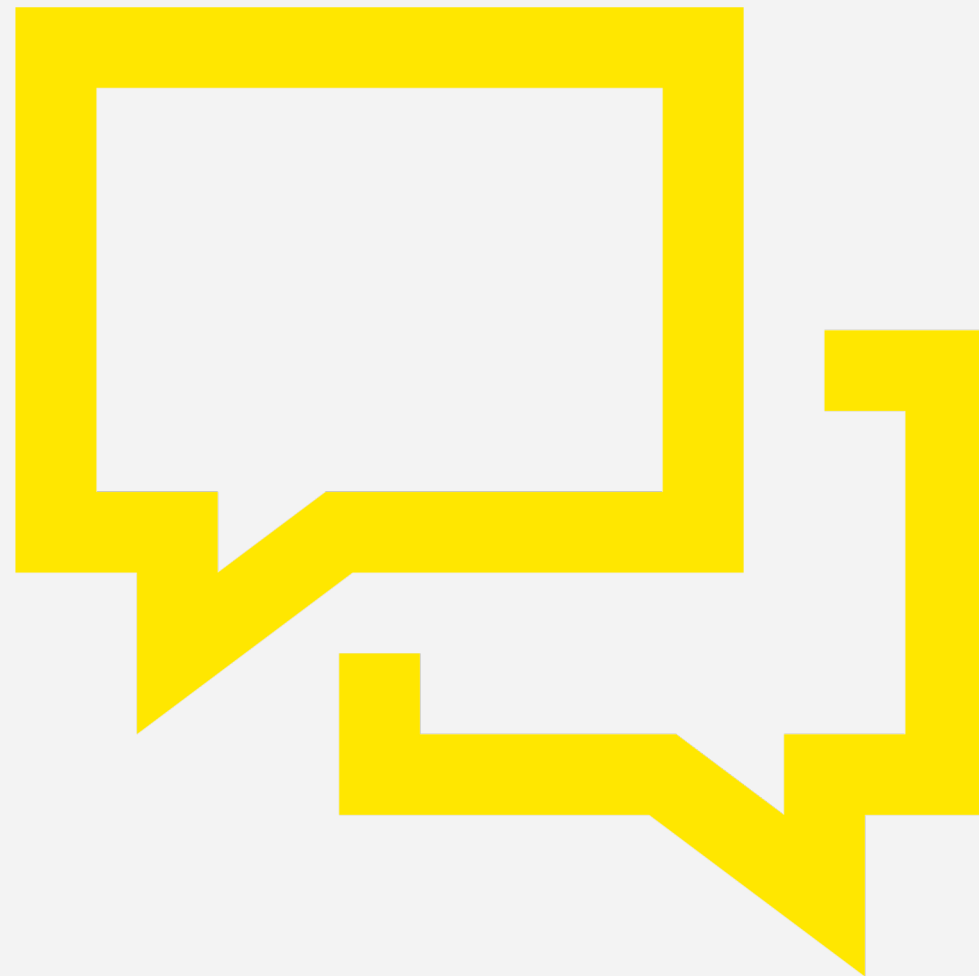


# Как и зачем мы сделали своё чат- решение?

Денис Аникин

<https://xfenix.ru>



# Денис Аникин

Кто я такой:

— работаю в Райффайзен банке



<https://xfenix.ru>

# Денис Аникин

Кто я такой:

- работаю в Райффайзен банке
- team lead в команде Chat



<https://xfenix.ru>

# Денис Аникин

Кто я такой:

- работаю в Райффайзен банке
- team lead в команде Chat
- community lead в Python Community



<https://xfenix.ru>

# Денис Аникин

Кто я такой:

- работаю в Райффайзен банке
- team lead в команде Chat
- community lead в Python Community
- fullstack: разрабатываю на back на python и front на typescript, занимаюсь devops



<https://xfenix.ru>



# Кому будет полезна презентация?

1

Тем, кто делает свои решения, связанные с чатами



# Кому будет полезна презентация?

1

Тем, кто делает свои решения, связанные с чатами

2

Тем, кто делает решения, связанные с websocket

# Кому будет полезна презентация?

1

Тем, кто делает свои решения, связанные с чатами

2

Тем, кто делает решения, связанные с websocket

3

Тем, кто пишет или хочет писать на python



# Кому будет полезна презентация?

1

Тем, кто делает свои решения, связанные с чатами

2

Тем, кто делает решения, связанные с websocket

3

Тем, кто пишет или хочет писать на python

4

Тем, кто работает с микросервисной архитектурой

# Кому будет полезна презентация?

1

Тем, кто делает свои решения, связанные с чатами

2

Тем, кто делает решения, связанные с websocket

3

Тем, кто пишет или хочет писать на python

4

Тем, кто работает с микросервисной архитектурой

5

Людям, которым интересны кейсы успешной продуктовой разработки

# Кому будет полезна презентация?

1

Тем, кто делает свои решения, связанные с чатами

2

Тем, кто делает решения, связанные с websocket

3

Тем, кто пишет или хочет писать на python

4

Тем, кто работает с микросервисной архитектурой

5

Людям, которым интересны кейсы успешной продуктовой разработки

6

Тем, кто выбирает стек для проектов

# Кому будет полезна презентация?

1

Тем, кто делает свои решения, связанные с чатами

2

Тем, кто делает решения, связанные с websocket

3

Тем, кто пишет или хочет писать на python

4

Тем, кто работает с микросервисной архитектурой

5


Людям, которым интересны кейсы успешной продуктовой разработки

6

Тем, кто выбирает стек для проектов

7

Тем, кому интересны лайфхаки из продакшена



# Что делает наша система?



# Наша система



**Что было не так с  
решением до этого?**



# Расскажу о том, что было

Поговорим о плюсах

— Это «коробка»





# Расскажу о том, что было

Поговорим о плюсах

- Это «коробка»
- Омниканальность и все необходимые функции в наличии

# Расскажу о том, что было

Поговорим о плюсах

- Это «коробка»
- Омниканальность и все необходимые функции в наличии
- Есть админка: РМО (рабочее место оператора), РМГ/РМС (рабочее место главного специалиста/супервизора)



# А теперь о минусах

И это не для всех будет так

— Абонентская плата



# А теперь о минусах

И это не для всех будет так

- Абонентская плата
- Производительность



# А теперь о минусах

И это не для всех будет так

- Абонентская плата
- Производительность
- Безопасность



# А теперь о минусах

И это не для всех будет так

- Абонентская плата
- Производительность
- Безопасность
- Качество

# Что не так с доработками?

К сожалению, в крупной системе без них никак

- Разработка велась медленно и непредсказуемо
- Доработки нередко могли попадать в основную систему и к конкурентам

# Производительность

Самое сложное

Внутри системы  
находилось ядро на python  
2.7

Не использовалась  
нативная асинхронность

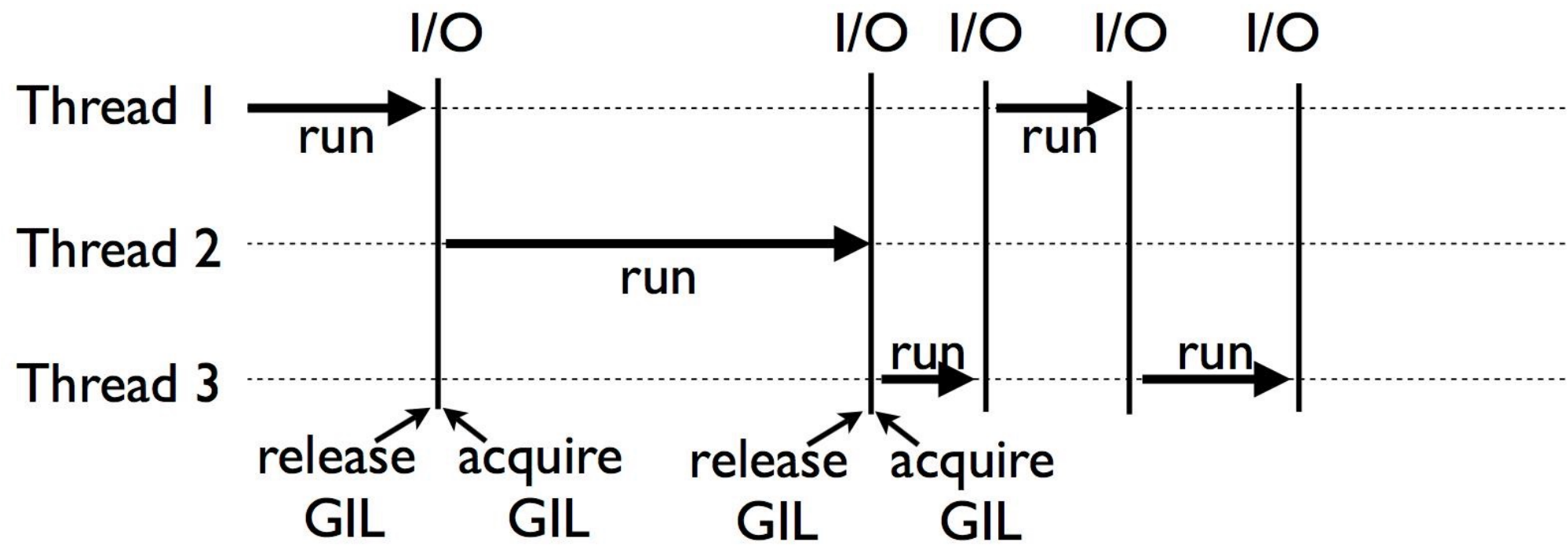
Система не поддерживала  
масштабирование вообще  
(никакое!)





# В чём проблема с асинхронность?

Ответ запутанный — это GIL

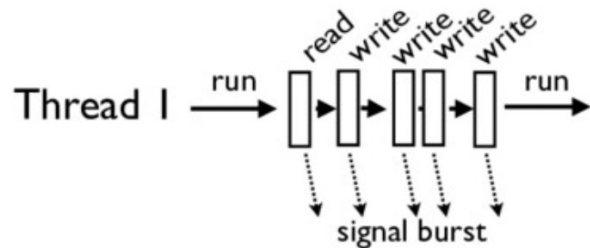


# В чём проблема с асинхронность?

Ответ запутанный — это GIL

## Behavior of I/O Handling

- I/O ops often do not block



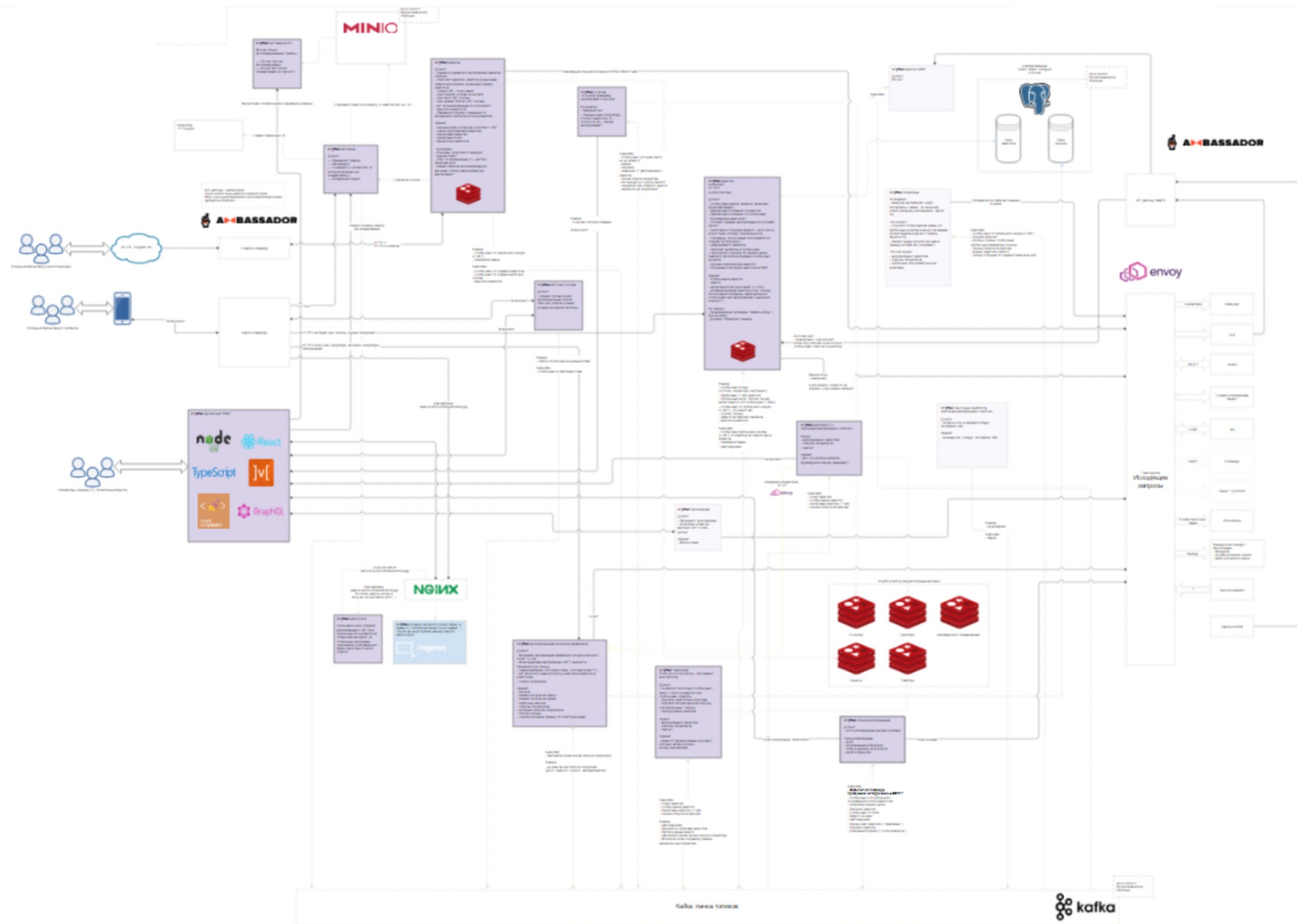
- Due to buffering, the OS is able to fulfill I/O requests immediately and keep a thread running
- However, the GIL is always released
- Results in GIL thrashing under heavy load

# А что с масштабированием?

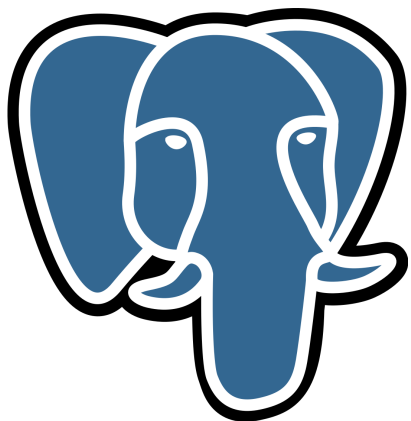
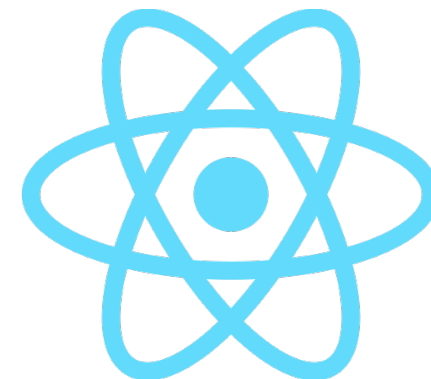
Вот это самое печальное

- Не вертикальное
- Не горизонтальное
- ...
- Никакое!

# Архитектура

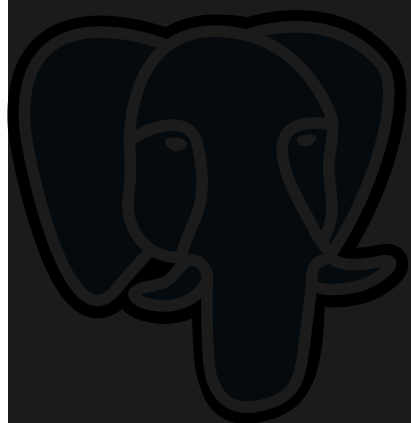
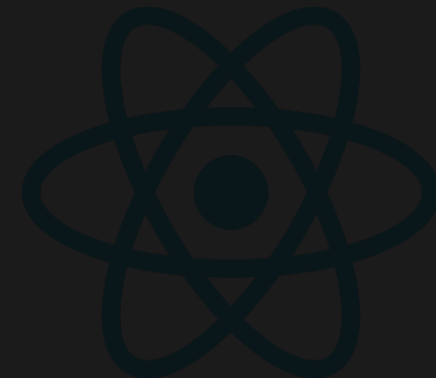


# Ключевые технологии



# Ключевые технологии

Кстати, есть ещё dragonfly и skytable





## Обращения

В очереди 4 ▾

- 
- Грошиковаан Ннильк Кказгериевич
- 
- Чат-бот: Как только специалист ответит, вам прид

Rada

Чат-бот

Подождите, пожалуйста, я приглашу специалиста, который поможет разобраться. 12:26 ✓

Как только специалист ответит, вам придет пуш. А пока можете свернуть или закрыть приложение. 12:26 ✓

19 августа 2022

#5981

Rada

Оператор 16:04

Чат-бот

Приветствую!  
Я Рэя — ваш виртуальный помощник. 16:04 ✓

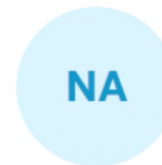
Аварийное сообщение 16:04 ✓

Подождите, пожалуйста, я приглашу специалиста, который поможет разобраться. 16:04 ✓

Как только специалист ответит, вам придет пуш. А пока можете свернуть или закрыть приложение. 16:04 ✓

Взять в работу

Назначить



Rada

### Персональная информация

Имя пользователя

Тут можно оставить важные заметки о клиенте

### Информация о диалоге

Скилл Telegram

Канал поступления Telegram

Номер диалога 5981



# Почему keydb?



# Настройка кластера

Про НА

— у нас 3 реплики

# Настройка кластера

Про HA

- у нас 3 реплики
- а так же 3 sentinel

# Настройка кластера

Про HA

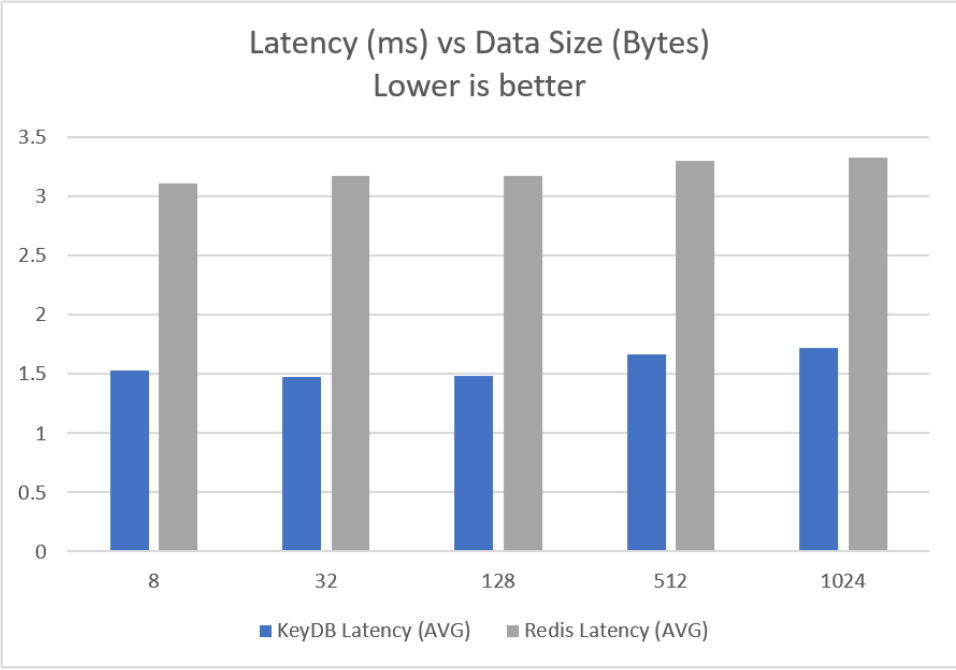
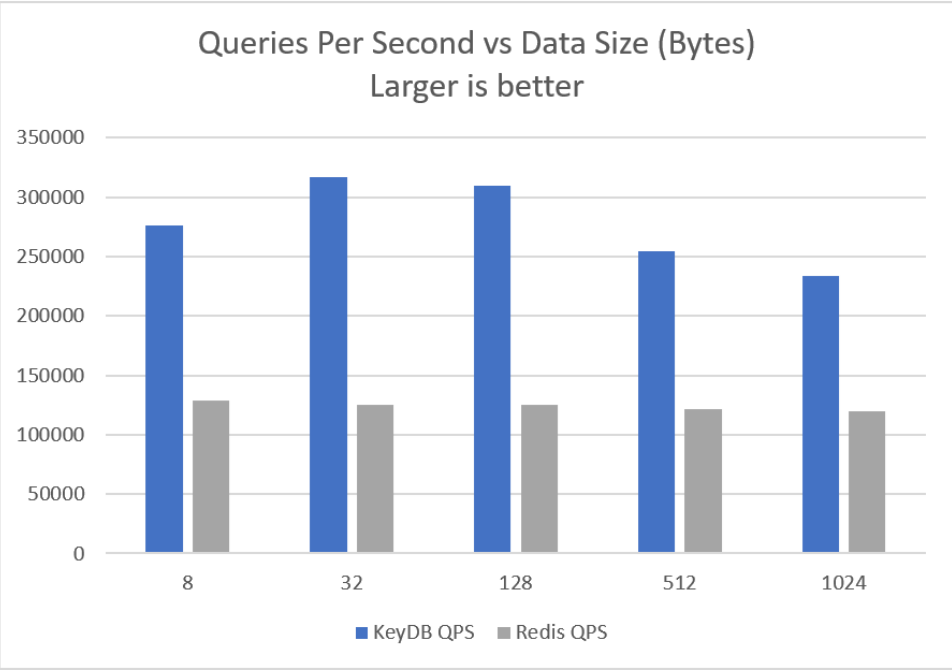
- у нас 3 реплики
- а так же 3 sentinel
- всё ради HA и быстрого failover

# Настройка кластера

Про HA

- у нас 3 реплики
- а так же 3 sentinel
- всё ради HA и быстрого failover

— мы не используем  
шардирование





# Лайфхак из продакшена #1

Супер важно

Вам необходимо мониторить количество коннекшенов

# Лайфхак из продакшена #1

Супер важно

Вам необходимо мониторить количество коннекшенов





# Лайфхак из продакшена #2

Ещё важнее

— Вам необходим пулинг соединений со стороны приложений

# Лайфхак из продакшена #2

Ещё важнее

- Вам необходим пулинг соединений со стороны приложений
- Наиболее это важно для высококонкурентных приложений: асинхронных, например, или полагающихся на green threads

# Лайфхак из продакшена #2

Ещё важнее

- Вам необходим пулинг соединений со стороны приложений
- Наиболее это важно для высококонкурентных приложений: асинхронных, например, или полагающихся на green threads
- Вам может понадобится ограничивать concurrency со стороны приложения



# Почему docker & kubernetes?

# Ну в конце концов 2022 год на дворе...





# Почему Kubernetes?

У нас > 1 машины

И мы выбираем подход «оркестрация»  
(не будем о «хореографии»)



# Почему Kubernetes?

## У нас > 1 машины

И мы выбираем подход «оркестрация»  
(не будем о «хореографии»)

## Нам нужно масштабирование

ReplicaSet, HPA и еще несколько  
концепций — и мы в дамках



# Почему Kubernetes?

## У нас > 1 машины

И мы выбираем подход «оркестрация»  
(не будем о «хореографии»)

## Нам нужно масштабирование

ReplicaSet, HPA и еще несколько концепций — и мы в дамках

## Нам нужна надежность

Здесь на помощь спешит control loop, probes, service mesh и много всего разного

# Почему Kubernetes?

## У нас > 1 машины

И мы выбираем подход «оркестрация»  
(не будем о «хореографии»)

## Мы желаем удобства разработчикам

Разработчики сами описывают то как их сервисы работают в продакшене

## Нам нужно масштабирование

ReplicaSet, HPA и еще несколько концепций — и мы в дамках

## Нам нужна надежность

Здесь на помощь спешит control loop, probes, service mesh и много всего разного

# Почему Kubernetes?

## У нас > 1 машины

И мы выбираем подход «оркестрация»  
(не будем о «хореографии»)

## Мы желаем удобства разработчикам

Разработчики сами описывают то как их сервисы работают в продакшене

## Нам нужно масштабирование

ReplicaSet, HPA и еще несколько концепций — и мы в дамках

## Мы хотим IAC

Никаких больше тонн баш скриптов, разбросанных по компьютерам и заправленных под мастрас ансибл плейбуков

## Нам нужна надежность

Здесь на помощь спешит control loop, probes, service mesh и много всего разного

# Почему Kubernetes?

## У нас > 1 машины

И мы выбираем подход «оркестрация»  
(не будем о «хореографии»)

## Мы желаем удобства разработчикам

Разработчики сами описывают то как их сервисы работают в продакшене

## Нам нужно масштабирование

ReplicaSet, HPA и еще несколько концепций — и мы в дамках

## Мы хотим IAC

Никаких больше тонн баш скриптов, разбросанных по компьютерам и запряженных под мастрас ансибл плейбуков

## Нам нужна надежность

Здесь на помощь спешит control loop, probes, service mesh и много всего разного

## Rolling update

Техника, которую довольно трудно реализовывать руками, здесь нам достается вообще бесплатно!

# Почему python?

# Сложно объяснить...





# Давайте о плюсах Python

Нас интересует версия 3 конечно

1

Асинхронность из коробки



# Давайте о плюсах Python

Нас интересует версия 3 конечно

1

Асинхронность из коробки

2

Динамическая типизация == разработка  
очень быстрая





# Давайте о плюсах Python

Нас интересует версия 3 конечно

1

Асинхронность из коробки

2

Динамическая типизация == разработка  
очень быстрая

3

Аннотации типов устраняют проблемы  
предыдущего пункта



# Давайте о плюсах Python

Нас интересует версия 3 конечно

1

Асинхронность из коробки

2

Динамическая типизация == разработка  
очень быстрая

3

Аннотации типов устраняют проблемы  
предыдущего пункта

4

Язык #1 по куче рейтингов



# Давайте о плюсах Python

Нас интересует версия 3 конечно

1

Асинхронность из коробки

2

Динамическая типизация == разработка  
очень быстрая

3

Аннотации типов устраняют проблемы  
предыдущего пункта

4

Язык #1 по куче рейтингов

5

Производительности хватает на приличную  
нагрузку + язык серьезно ускоряется

# Давайте о плюсах Python

Нас интересует версия 3 конечно

1

Асинхронность из коробки

2

Динамическая типизация == разработка  
очень быстрая

3

Аннотации типов устраняют проблемы  
предыдущего пункта

4

Язык #1 по куче рейтингов

5

Производительности хватает на приличную  
нагрузку + язык серьезно ускоряется

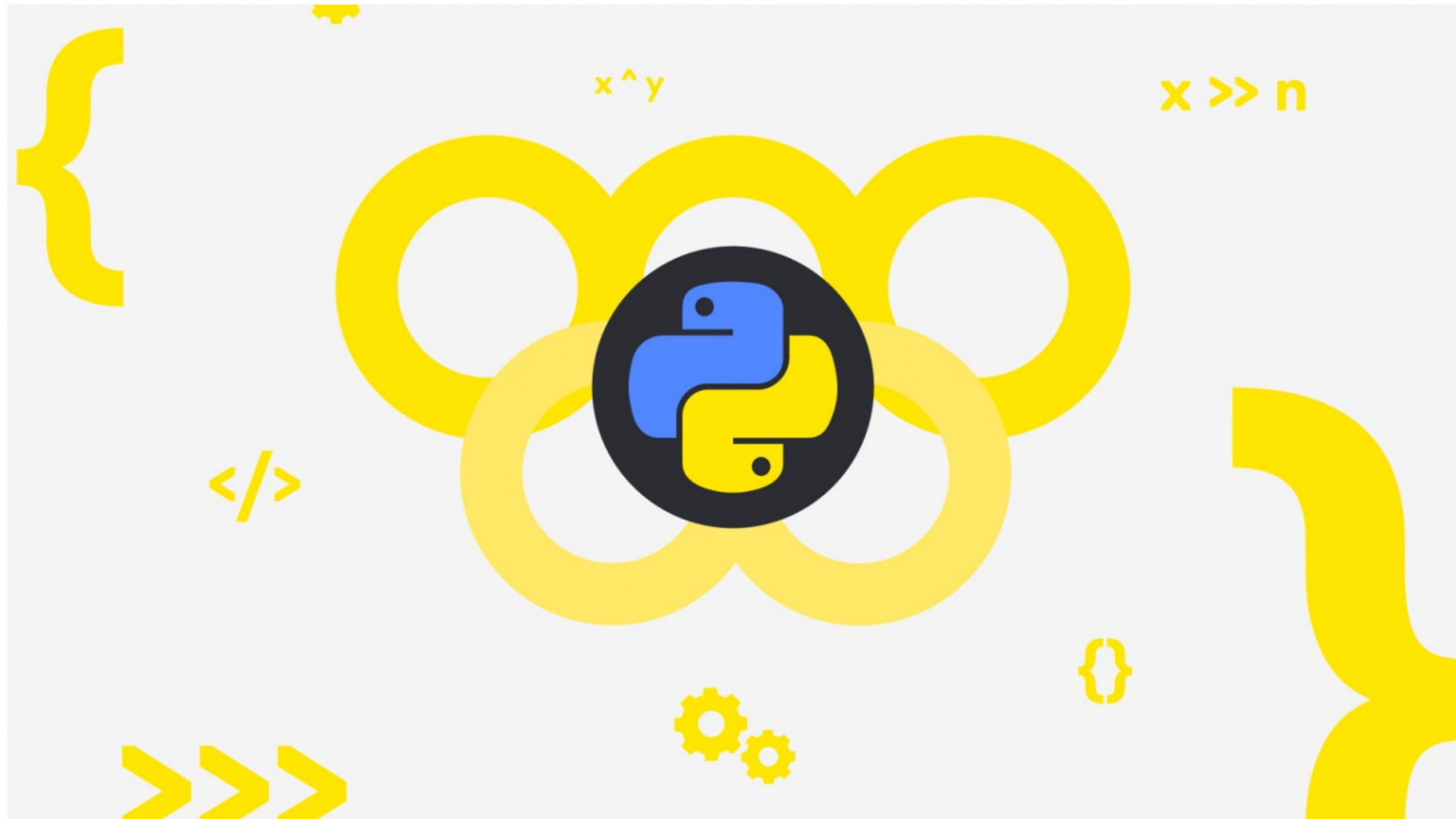
6

Есть миллион готовых библиотек



# Python — серьезный язык для разработки backend

Блог компании Райффайзен Банк, Python\*, Программирование\*, Django\*





# Python — серьезный язык для разработки backend

Блог компании Райффайзен Банк, Python\*, Программирование\*, Django\*



# Выводы по Python

Мое спорное мнение

Python — первый язык, с которого вы можете начать любой бекенд

# Выводы по Python

Мое спорное мнение

Python — первый язык, с которого вы можете начать любой бекенд

Когда вам не хватает его скорости, вы берете go, java/kotlin или rust



# Kafka



# Её плюсы

## Потрясающая скорость

Всё из-за её архитектуры

# Её плюсы

## Потрясающая скорость

Всё из-за её архитектуры

## Отменная надежность, масштабирование

Наши сообщения надежно хранятся и  
доставляются



# Её плюсы

## Потрясающая скорость

Всё из-за её архитектуры

## Отменная надежность, масштабирование

Наши сообщения надежно хранятся и доставляются

## Гарантия порядка доставки

Это одно из самых важных

# Про производительность

— Нам действительно важно сочетание масштабирования и гарантии порядка доставки

# Про производительность

- Нам действительно важно сочетание масштабирования и гарантии порядка доставки
- Нам действительно важно держать много RPS в купе с low latency



# Некоторые минусы

Почему, возможно, вы не захотите у себя иметь кафку

## Высокая сложность

Изучить кафку непросто

# Некоторые минусы

Почему, возможно, вы не захотите у себя иметь кафку

## Высокая сложность

Изучить кафку не просто

## Не очень много возможностей

По сравнению с, например, RabbitMQ у нас здесь просто распределенный лог





# Лайфхак из продакшена #1

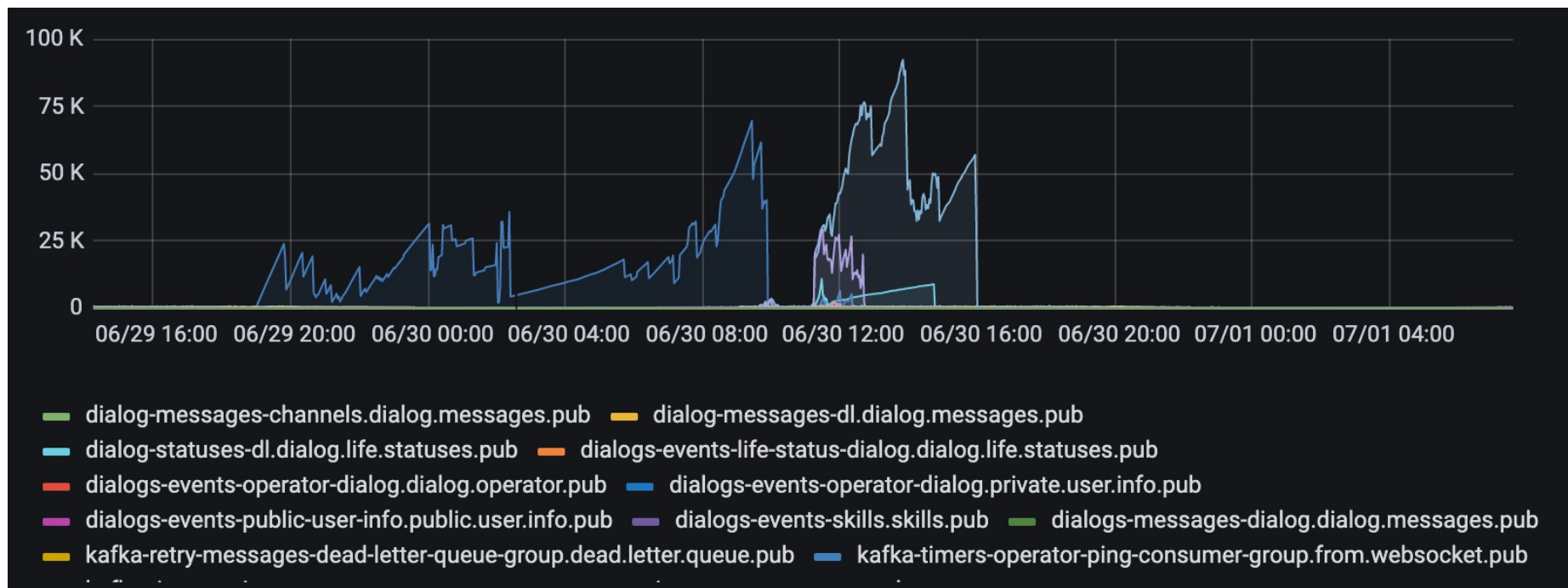
Супер важно

**Обязательно** необходимо мониторить topic lag

# Лайфхак из продакшена #1

Супер важно

**Обязательно** необходимо мониторить topic lag



# Лайфхак из продакшена #2

Тоже важно

— Вам понадобится поддерживать dead letter

# Лайфхак из продакшена #2

Тоже важно

- Вам понадобится поддерживать dead letter
- Вам понадобится поддерживать dead letter + timeout + retry

# Немного о вебсокетах

# Про сам протокол

Его плюсы

— Нативно асинхронен

# Про сам протокол

Его плюсы

- Нативно асинхронен
- Реализован под все платформы

# Про сам протокол

Его плюсы

- Нативно асинхронен
- Реализован под все платформы
- Быстр



# Про сам протокол

Минусы

— Непривычная семантика для тех, кто всю жизнь с HTTP

# Про сам протокол

## Минусы

- Непривычная семантика для тех, кто всю жизнь с HTTP
- Нет ничего, протокол очень примитивный, все приходится делать самим

# Про сам протокол

## Минусы

- Непривычная семантика для тех, кто всю жизнь с HTTP
- Нет ничего, протокол очень примитивный, все приходится делать самим
- Есть конкурент в лице SSE

# Про сам протокол

## Минусы

- Непривычная семантика для тех, кто всю жизнь с HTTP
- Нет ничего, протокол очень примитивный, все приходится делать самим
- Есть конкурент в лице SSE
- По мнению некоторых уже много лет не нужен в силу пункта выше

# Про сам протокол

## Минусы

- Непривычная семантика для тех, кто всю жизнь с HTTP
- Нет ничего, протокол очень примитивный, все приходится делать самим
- Есть конкурент в лице SSE
- По мнению некоторых уже много лет не нужен в силу пункта выше
- Много вопросов с аутентификацией

# Про сам протокол

## Минусы

- Непривычная семантика для тех, кто всю жизнь с HTTP
- Нет ничего, протокол очень примитивный, все приходится делать самим
- Есть конкурент в лице SSE
- По мнению некоторых уже много лет не нужен в силу пункта выше
- Много вопросов с аутентификацией
- Если «сессия» истекла, надо рвать соединение

# Про сам протокол

## Минусы

- Непривычная семантика для тех, кто всю жизнь с HTTP
- Нет ничего, протокол очень примитивный, все приходится делать самим
- Есть конкурент в лице SSE
- По мнению некоторых уже много лет не нужен в силу пункта выше
- Много вопросов с аутентификацией
- Если «сессия» истекла, надо рвать соединение
- Кто-то должен держать пул соединений

# Двусторонний роутинг

Мы решали эту проблему

**Когда мы «пушим» в систему, всё просто**

Мы берем сообщение снаружи и кидаем в кафку





# Двусторонний роутинг

Мы решали эту проблему

**Когда мы «пушим» в систему, всё просто**

Мы берем сообщение снаружи и кидаем в кафку

**Но как попасть «обратно»?**

Вот это совсем непросто и готовых решений в интернете нет



# Двусторонний роутинг

Мы решали эту проблему

## Когда мы «пушим» в систему, всё просто

Мы берем сообщение снаружи и кидаем в кафку

## Но как попасть «обратно»?

Вот это совсем непросто и готовых решений в интернете нет

## Ответ есть в нашем докладе

Выступление на moscow python conf ++ 2021

**И в заключение чего  
достигли**



# Подводим итоги

Мы достигли следующего:

**Дизайн в общем для  
банка стиле**



# Подводим итоги

Мы достигли следующего:

Дизайн в общем для  
банка стиле

Горизонтальное  
масштабирование

# Подводим итоги

Мы достигли следующего:

**Дизайн в общем для  
банка стиле**

**Горизонтальное  
масштабирование**

**Отказоустойчивость**

# Подводим итоги

Мы достигли следующего:

**Дизайн в общем для  
банка стиле**

**Горизонтальное  
масштабирование**

**Отказоустойчивость**

**Полный контроль над  
своей системой**

# Подводим итоги

Мы достигли следующего:

**Дизайн в общем для  
банка стиле**

**Горизонтальное  
масштабирование**

**Отказоустойчивость**

**Полный контроль над  
своей системой**

**Предсказуемая  
скорость разработки**



# Подводим итоги

Мы достигли следующего:

**Дизайн в общем для  
банка стиле**

**Горизонтальное  
масштабирование**

**Отказоустойчивость**

**Полный контроль над  
своей системой**

**Предсказуемая  
скорость разработки**

**Безопасность**

# Спасибо



Денис Аникин

<https://xfenix.ru/>