CI/CD как драйвер разработки

Кирилл Сюзев
Технический лидер,
SourceCraft

infra.conf

Кирилл Сюзев

Tech Lead

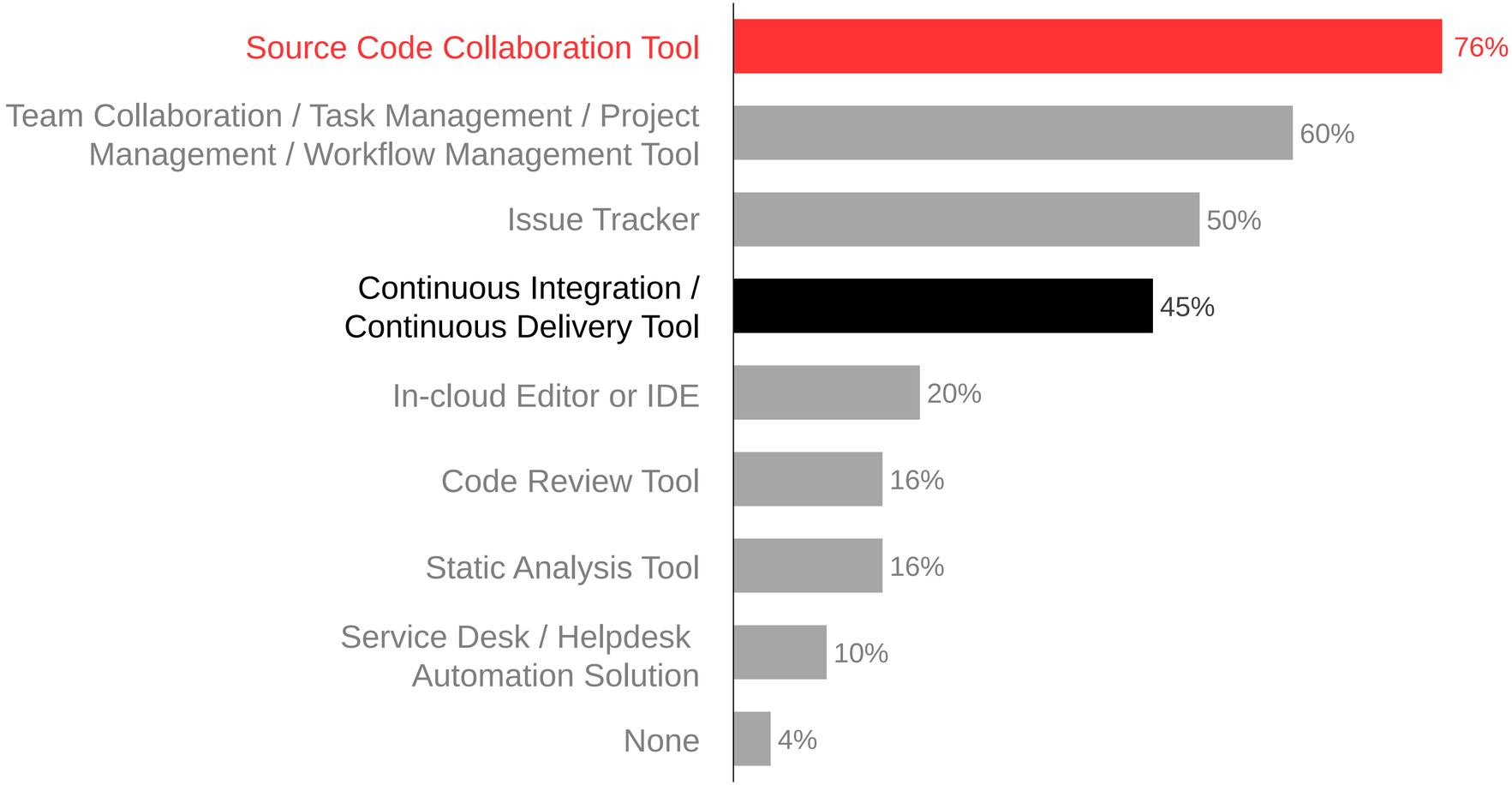
Yandex Infrastructure

План доклада

1. Немного истории
2. Идея создания SourceCraft CI/CD
3. Существующие решения
4. Архитектура SourceCraft CI/CD
5. Углубляемся в инфраструктуру
6. Планы на будущее

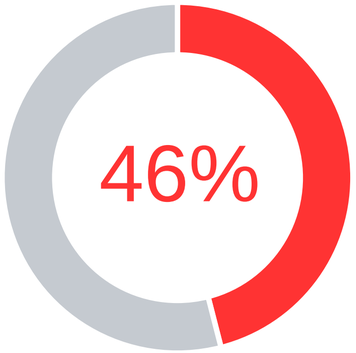
Инструменты разработчика

JB: The State of Developer Ecosystem 2023 Survey

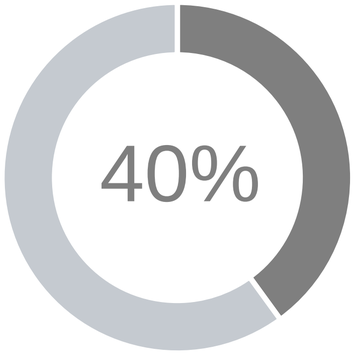


Инструменты разработчика в облаке

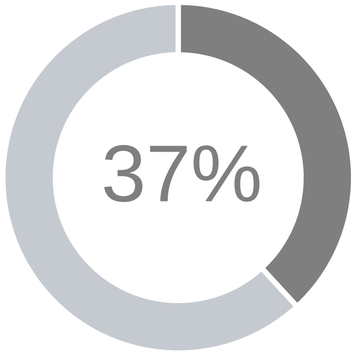
Do you use any of the following types of tools in the cloud?



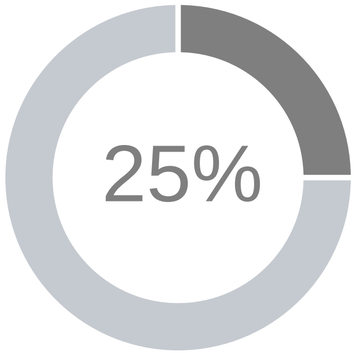
Continuous Integration Tool



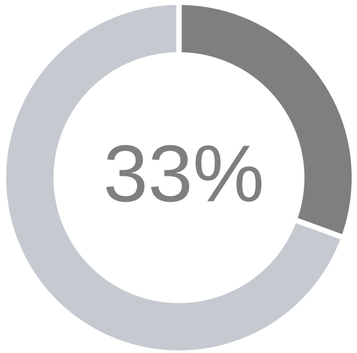
Issue Tracker



Continuous Delivery Tool



Automated Code Review / Static Analysis Tool

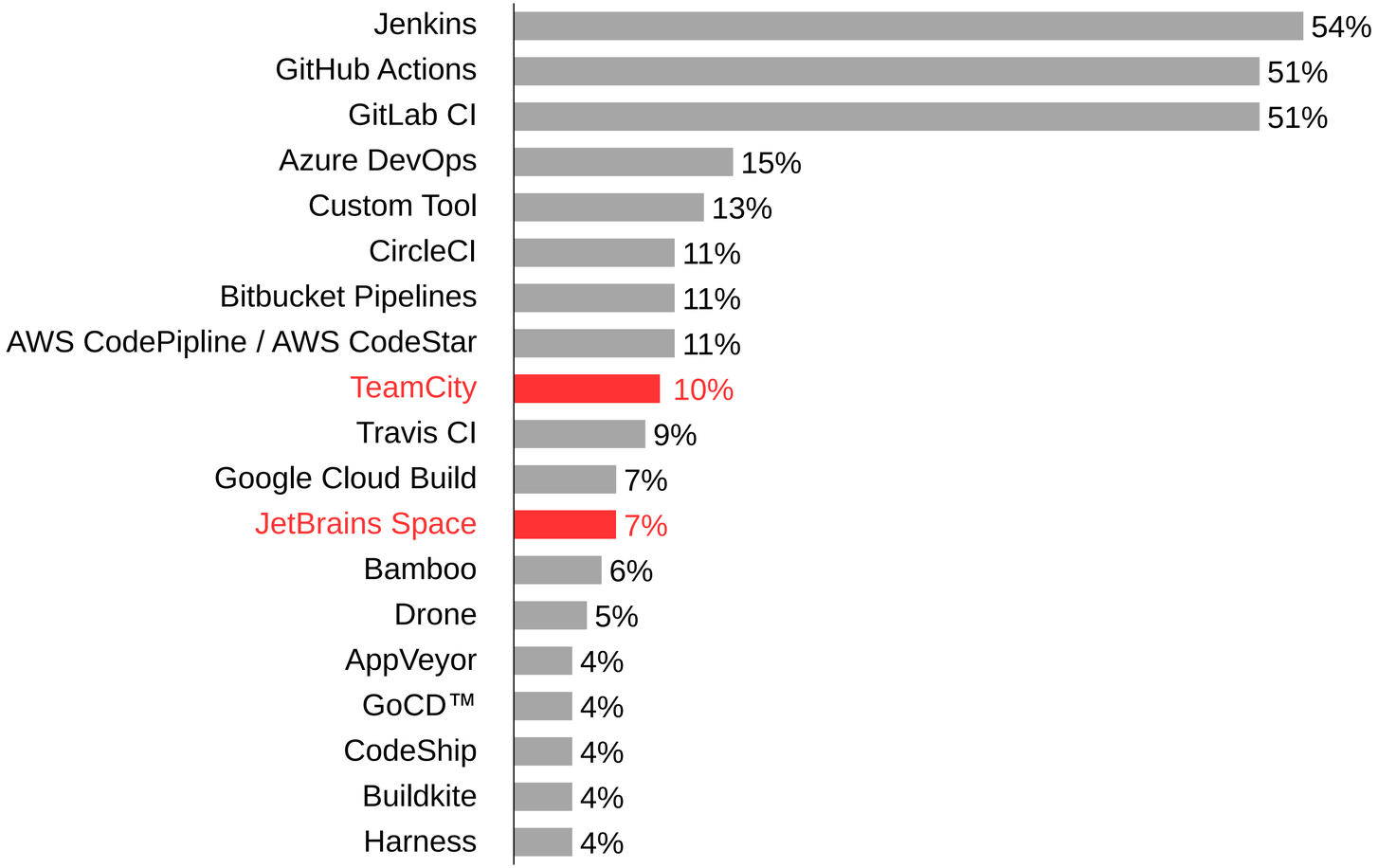


I don’t use any of these tools

Ваш опыт работы с CI

Что на самом деле?

Which Continuous Integration (CI) Systems do you regularly use?



Идея появления SourceCraft CI/CD

Наши цели:

1. Простой и декларативный язык описания
2. Self-hosted
3. Бесшовная интеграция с облаком
4. Масштабируемость

История почти успеха: Arcadia CI

История почти успеха: взять Arcadia CI

Плюсы Arcadia CI

Относительно молодой
проект без Legacy

Большой накопленный опыт
решения пользовательских задач

Активно развивается

Космический корабль

Минусы Arcadia CI

Оптимизирован под монорепы

Завязан на внутреннюю
инфраструктуру
большого Яндекса

Космический корабль

GitLab
vs GitHub

GitLab vs GitHub: DSL

GitLab

```
coverage:
  script:
    - pytest --cov=./ --cov-report=xml:c.xml
  artifacts:
    reports:
      cobertura: c.xml
```

GitHub

```
- name: Run tests with coverage
  run: |
    pytest --cov=./ --cov-report=xml:c.xml

- name: Upload to Codecov (optional)
  uses: codecov/codecov-action@v3
  with:
    token: ${{ secrets.CODECOV_TOKEN }}
    file: ./c.xml
```

GitLab vs GitHub: DSL

Декларативность

GitLab побеждает
с небольшим перевесом

Интегриро-
ванность

GitLab побеждает
нокаутом

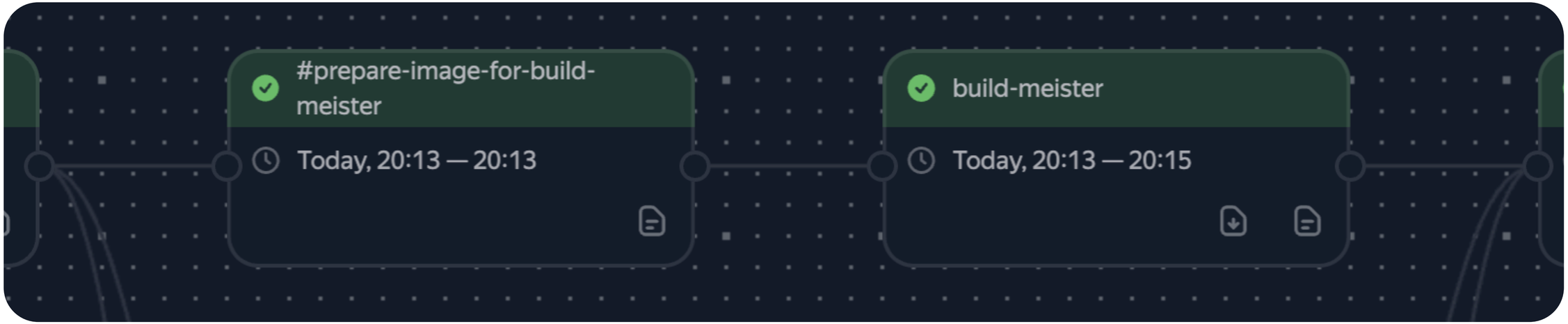
Гибкость

GitHub выходит
победителем

Наш подход к DSL

1. Берём себе декларативность
2. Заодно берём интеграцию
3. Добавляем щепотку Public API
4. Получаем гибкость

```
- name: build-meister
  image: golang:1.22-alpine3.20
  needs: [ "vendor" ]
  script:
    - go build -C ci/cmd/meister
```



Архитектура SourceCraft CI/CD

Проектируем кластер исполнения

Требования к безопасности

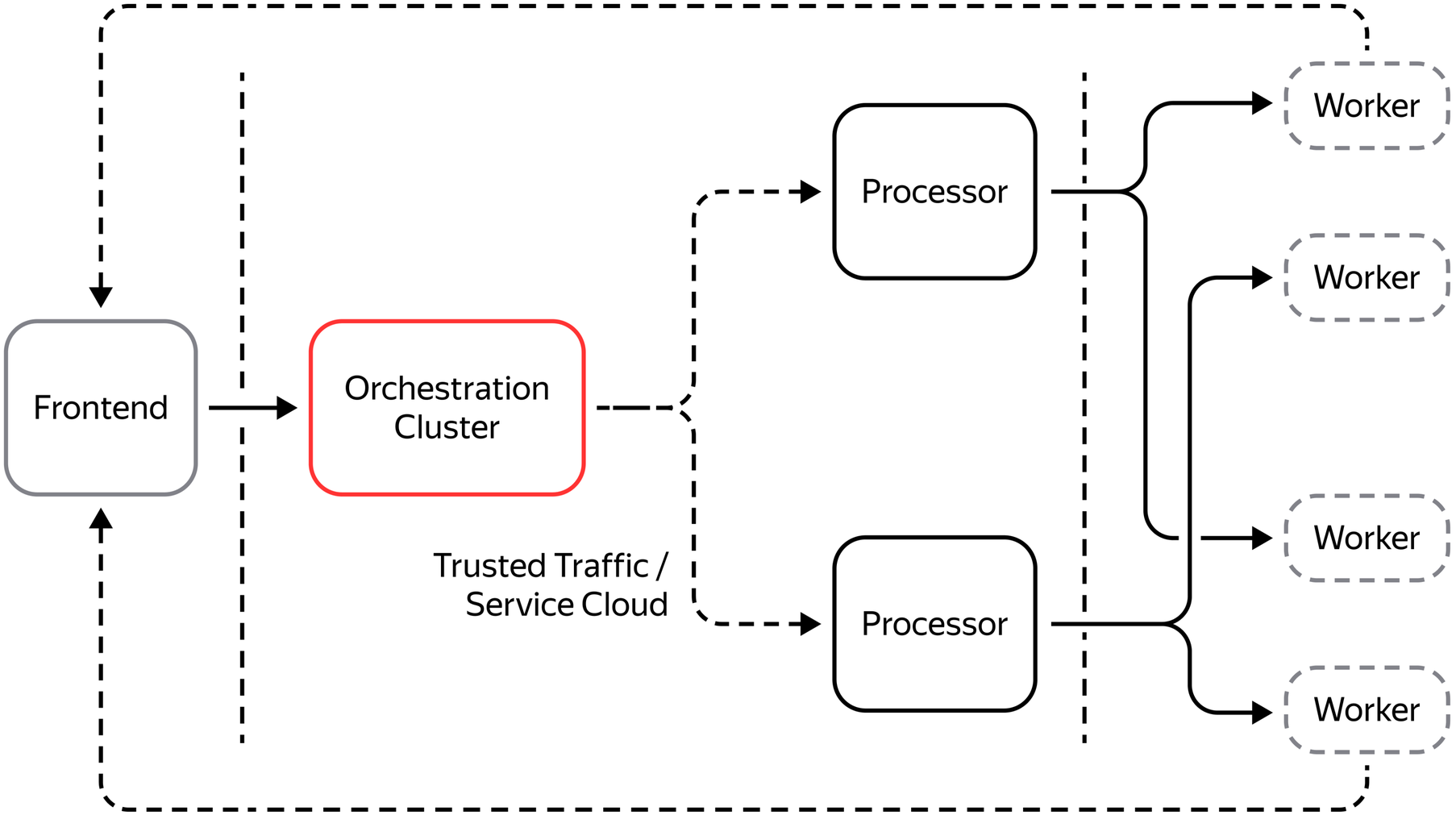
Пользователь
может делать
что хочет
в кубике

Пользователь
видит только
свои данные
и секреты

Пользователь
не может
попасть в чужой
репозиторий

Пользователь
не может
попасть
в Control Plane

Архитектура SourceCraft CI/CD



Подход #1: K8s®

Идея:

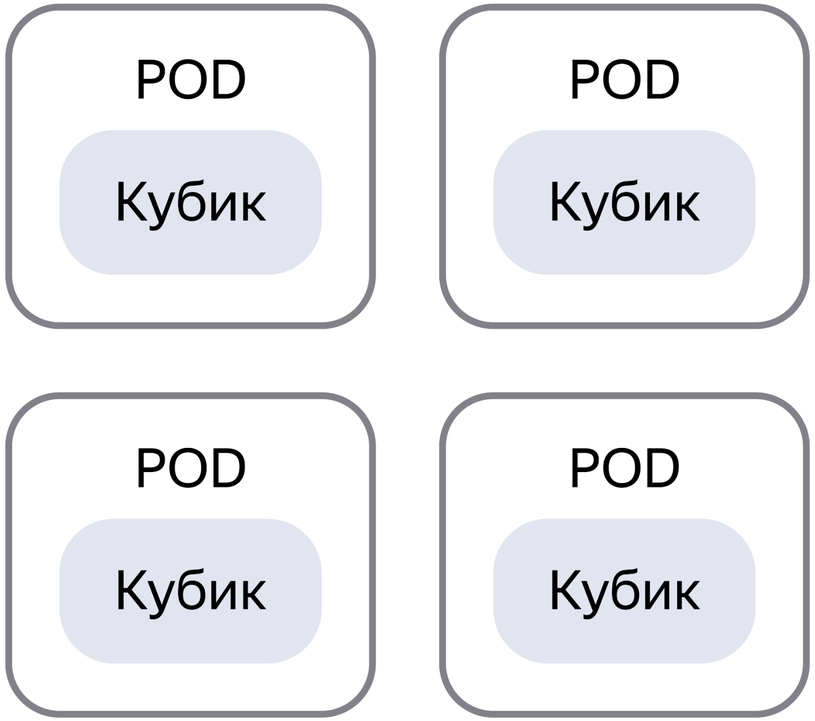
1. Создаём отдельный K8s®-кластер
2. Запускаем каждую задачу в отдельном Pod’е
3. По завершении задачи удаляем Pod

Плюсы:

- Pod — быстро и дёшево создать и удалить
- Быстро масштабируется
- K8s® — стандартный способ исполнения

Минусы:

- K8s® Pods — изоляция
- Безопасность
- Передача данных
- Docker® in Docker® — кошмарный кошмар



Подход #2: Docker® внутри EC2

Идея:

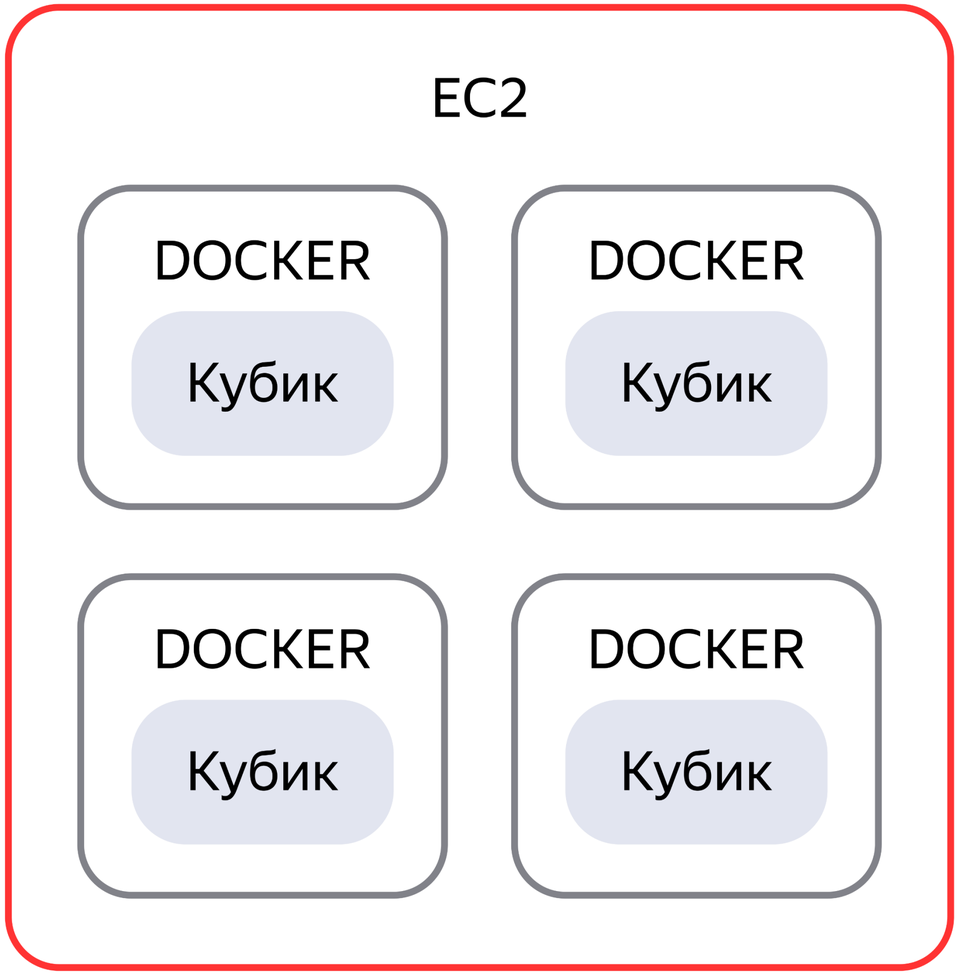
1. Task == EC2
2. Кубик == Docker® Image
3. Автоудаление

Плюсы:

- Ключи от VM <> доступ
- Изоляция

Минусы:

- Передача данных
- Docker® in Docker® —
  всё ещё кошмарный кошмар



Подход #3: честный EC2

Идея:

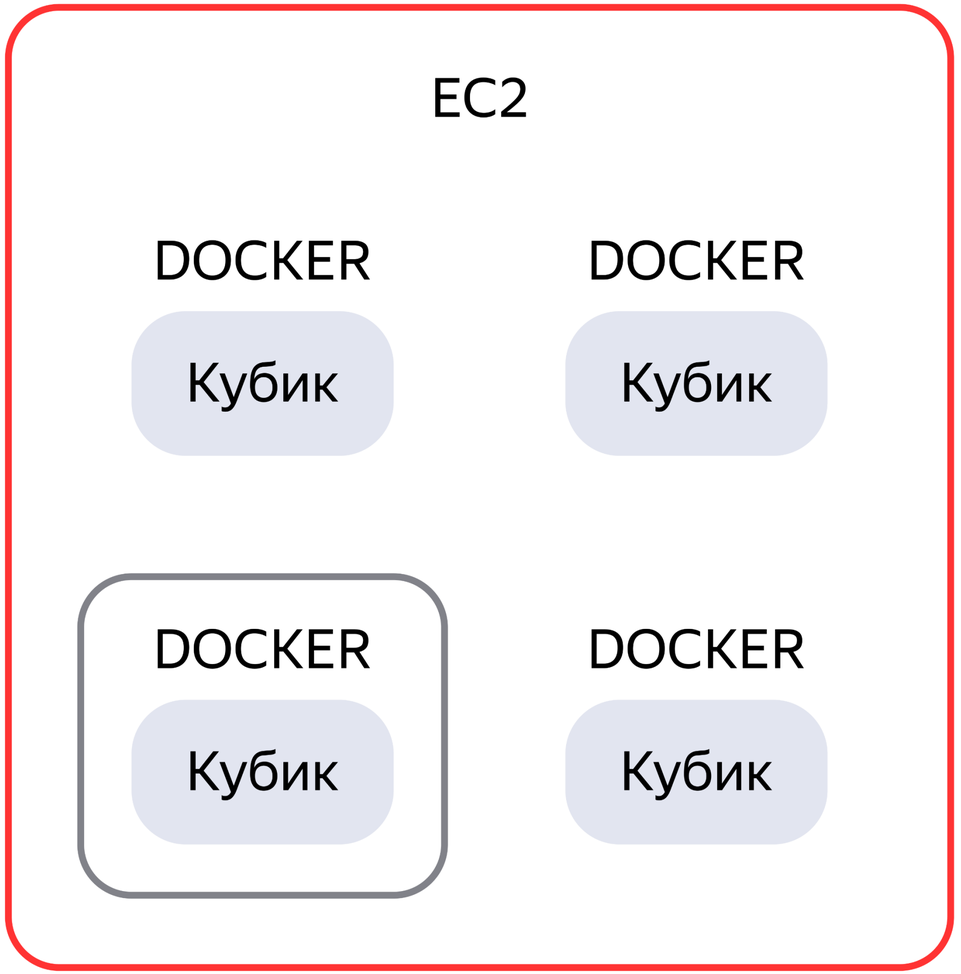
1. Task == VM
2. Кубик — Docker® или нет

Плюсы:

- Все плюсы предыдущего решения
- Нет Docker® in Docker®

Минусы:

Медленно



Подход #4: пул EC2

Идея:

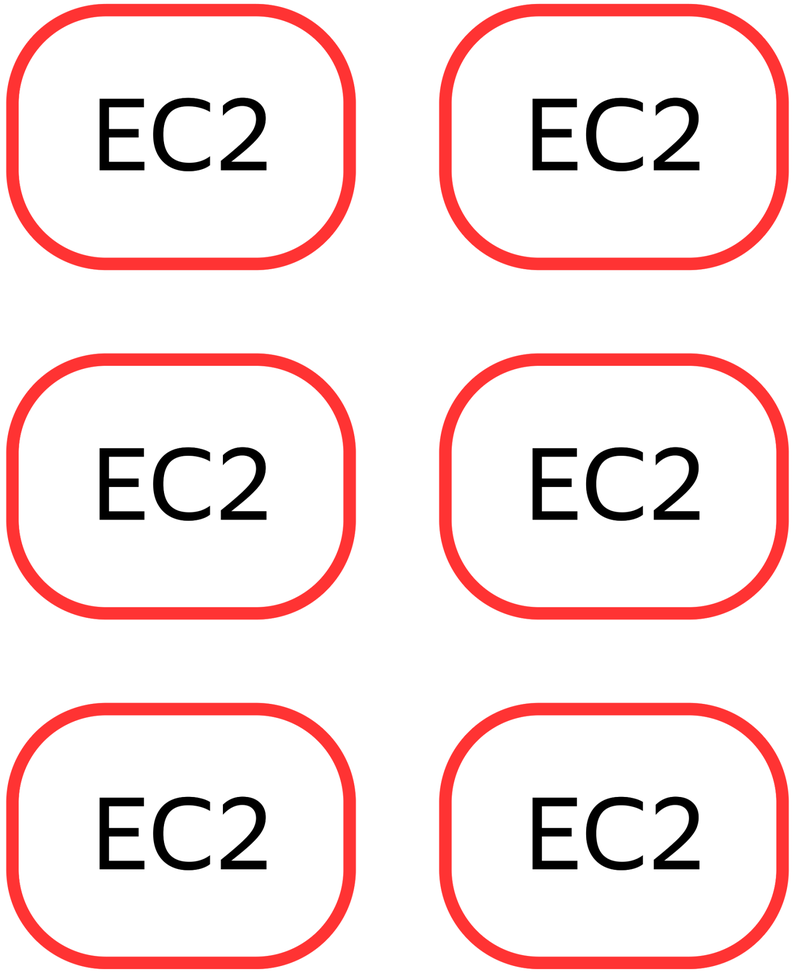
Добавляем пул прогретых EC2

Плюсы:

- Все плюсы предыдущего решения
- Очень быстро

Минусы:

Дорого



Подход #5: Serverless

Идея:

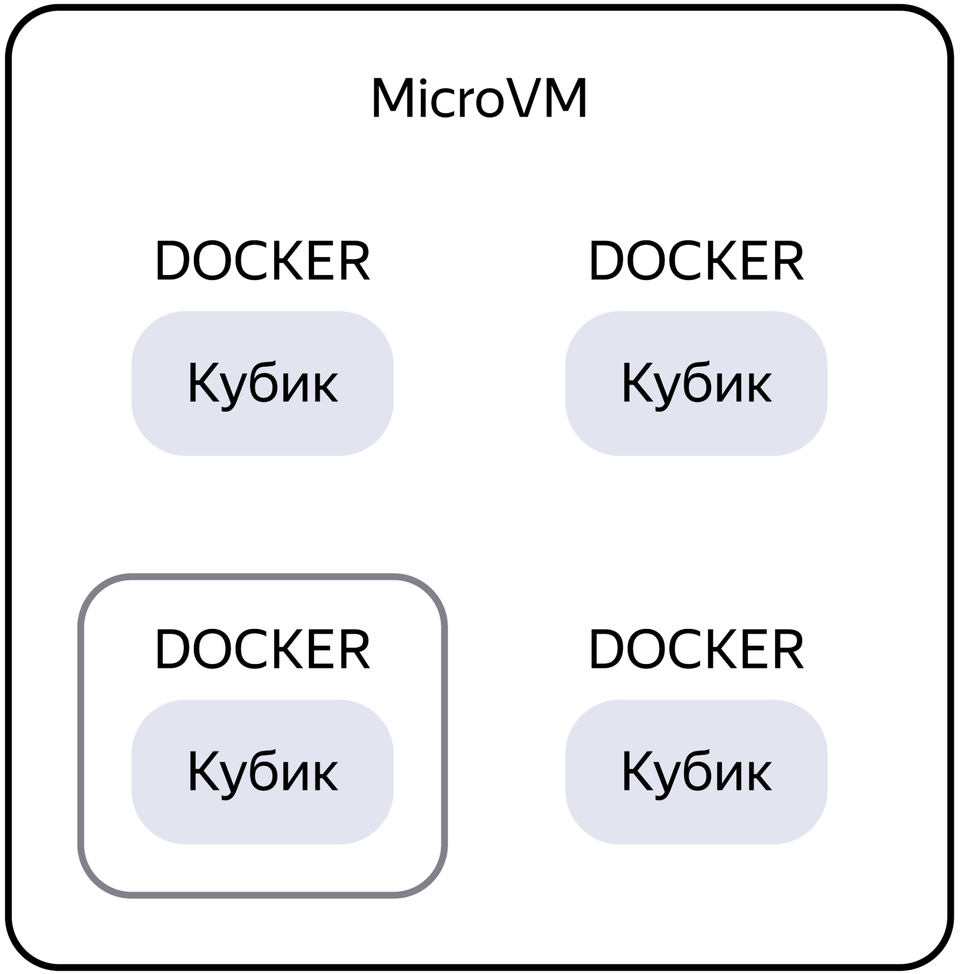
Используем MicroVM вместо EC2

Плюсы:

- Почти мгновенный старт
- Не нужен пул
- Изолированность
- Безопасность

Минусы:

- Передача данных
- Docker® in Docker®
- Время на прогрев



Подход #6: Personal VMs (в проработке)

Идея:

1. Не зависит от типа VM
2. Прикапываем VM для текущего пользователя
3. Отдаём только её для следующих запусков

Плюсы:

- Почти мгновенный старт
- Отличная изолированность

Минусы:

- Более сложная схема, вопросы по безопасности
- Ошибки при кешировании

Поищем RCE

1. RCE — Remote Code Execution
2. Выполнить код на своей VM — OK
3. Вылезти из Data Plane в Control Plane — не OK
4. Попасть на чужую VM — не OK

Простой RCE

1. Выполняем пользовательский скрипт
2. В скрипте может быть что угодно
3. Here Doc для многострочных скриптов
4. SSH вызывается в Control Plane

```
func wrong(hackerScript string) string {
    return fmt.Sprintf(
        `ssh -T user@remote.host <<'EOF'
        %s
        EOF`, hackerScript)
}
```

Вот и решение?

```
import (
    "fmt"
    "math/rand/v2"
)

func wrong(hackerScript string) string {
    eof := fmt.Sprintf("EOF_%v", rand.Uint64())
    return fmt.Sprintf(
        `ssh -T user@remote.host <<'%v'
        %s
        %v`, eof, hackerScript, eof)
}
```

Теперь точно исправили?

```
import (
    "math/rand/v2"
    "time"
)

func getRandomUint64() uint64 {
    now := time.Now().UnixNano()
    r := rand.NewPCG(uint64(now), uint64(now>>32))
    return r.Uint64()
}
```

Финальная версия

```
import (
    "crypto/rand"
    "encoding/binary"
)

func getRandomUint64() uint64 {
    buf := make([]byte, 8)
    _, _ = rand.Read(buf)
    return binary.BigEndian.Uint64(buf)
}
```

Проектируем Real Time Logging

Основная проблема — скорость и стоимость решения

Нужно доставлять логи до пользователя в режиме реального времени

Подход #1

Файл на S3

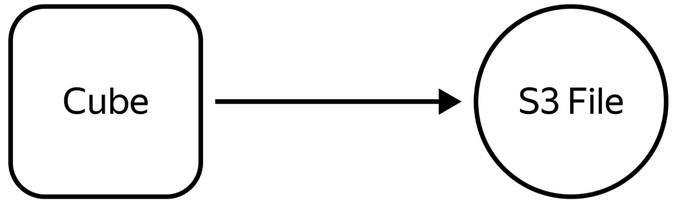
Лог пишем в файл на S3

Плюсы:

- Очень дёшево
- Быстрое чтение — запись

Минусы:

- Нет Append
- Нет Real Time



Подход #2

#1 + много файлов

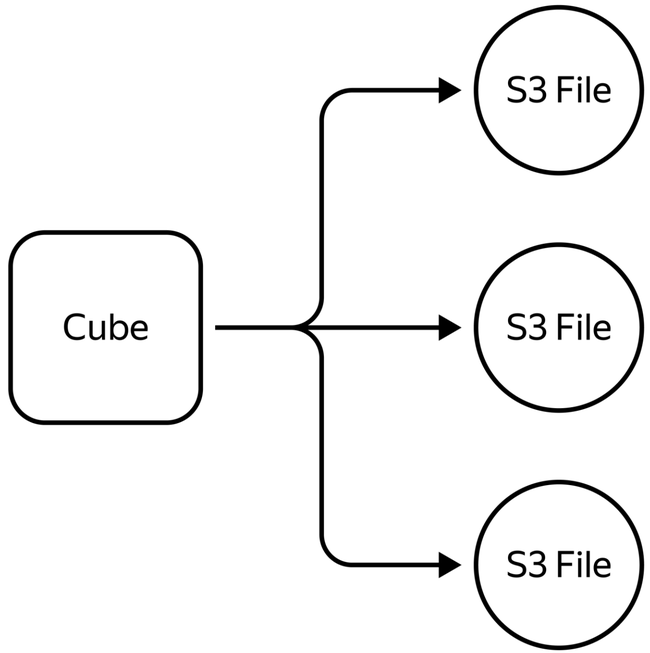
Пишем в новый файл с каждым блоком логов

Плюсы:

- Очень дёшево
- Быстрое чтение — запись
- Почти Real Time

Минусы:

Иногда долго ждать



Подход #3

#2 + Batch

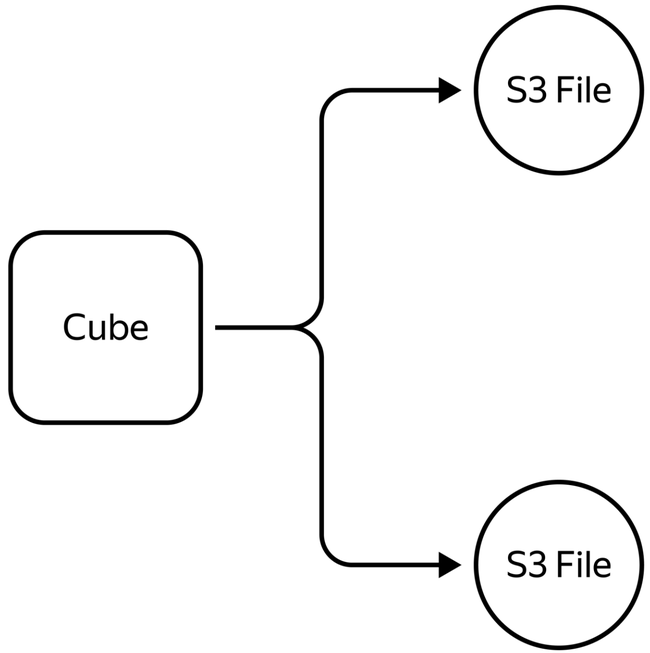
Собираем логи в батч, пишем батч в новый файл

Плюсы:

- Очень дёшево
- Быстрое чтение — запись
- Почти Real Time

Минусы:

Иногда долго ждать



Подход #4

#3 + Parallel

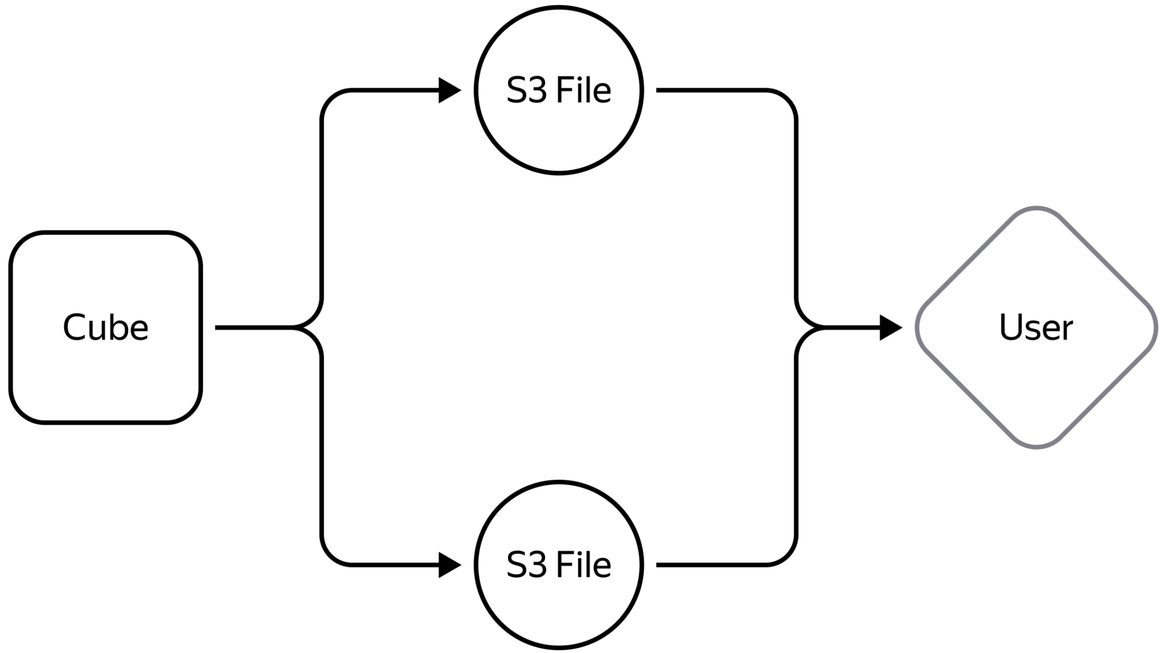
При показе читаем файлы из S3 в несколько потоков

Плюсы:

- Очень дёшево
- Быстрое чтение — запись
- Почти Real Time

Минусы:

Иногда долго ждать



Подход #4

YDB

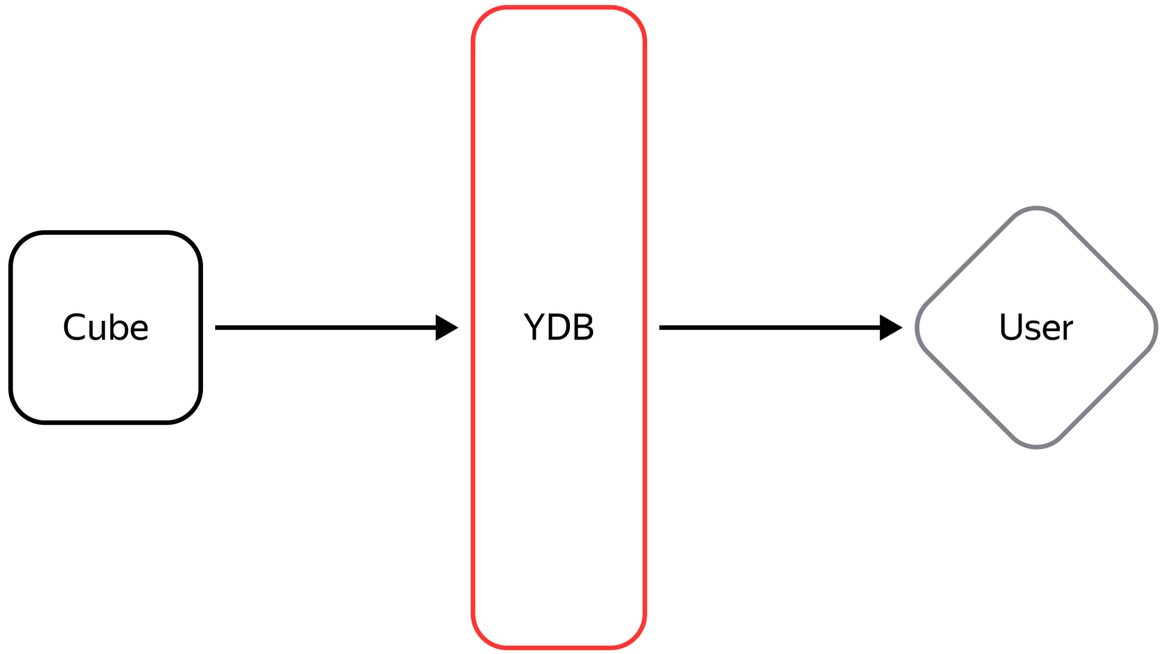
Кладём логи в таблицу.
Один батч — одна запись

Плюсы:

- По стоимости близко к S3
- Быстрое чтение — запись
- Real Time

Минусы:

- Нет полнотекстового поиска
- Ждём выполнения Select



Подход #5

YDB + Streaming

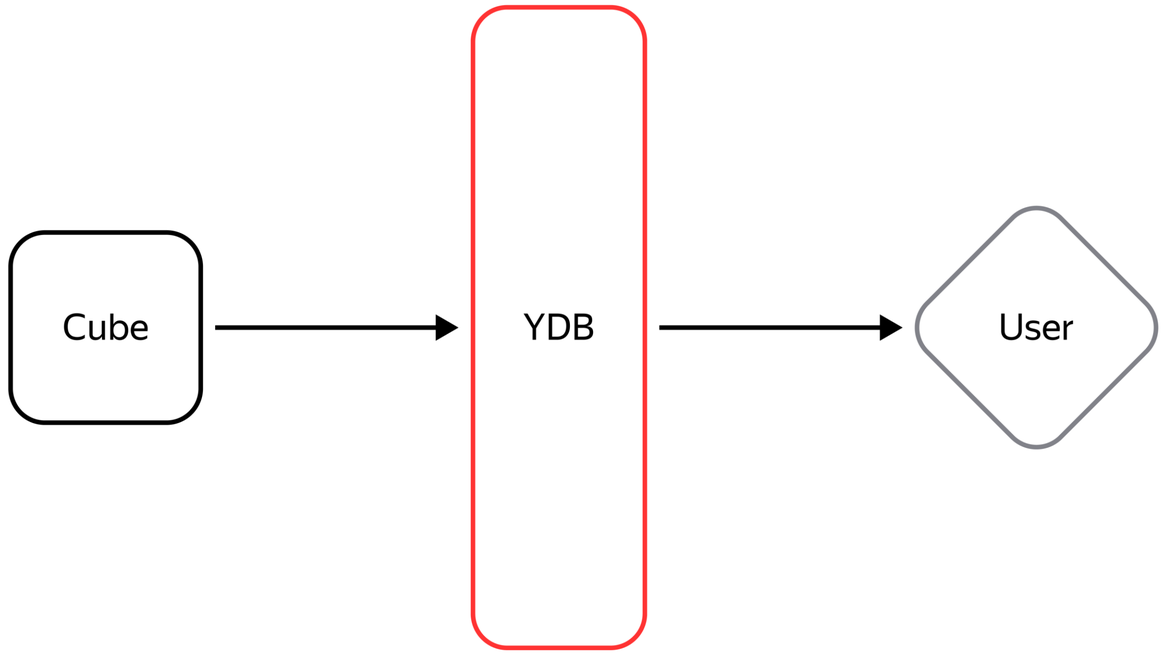
Отдаём записи пользователю в реальном времени

Плюсы:

- По стоимости близко к S3
- Быстрое чтение — запись
- Real Time

Минусы:

Нет полнотекстового поиска



Планы на будущее

1

Персональные машинки

2

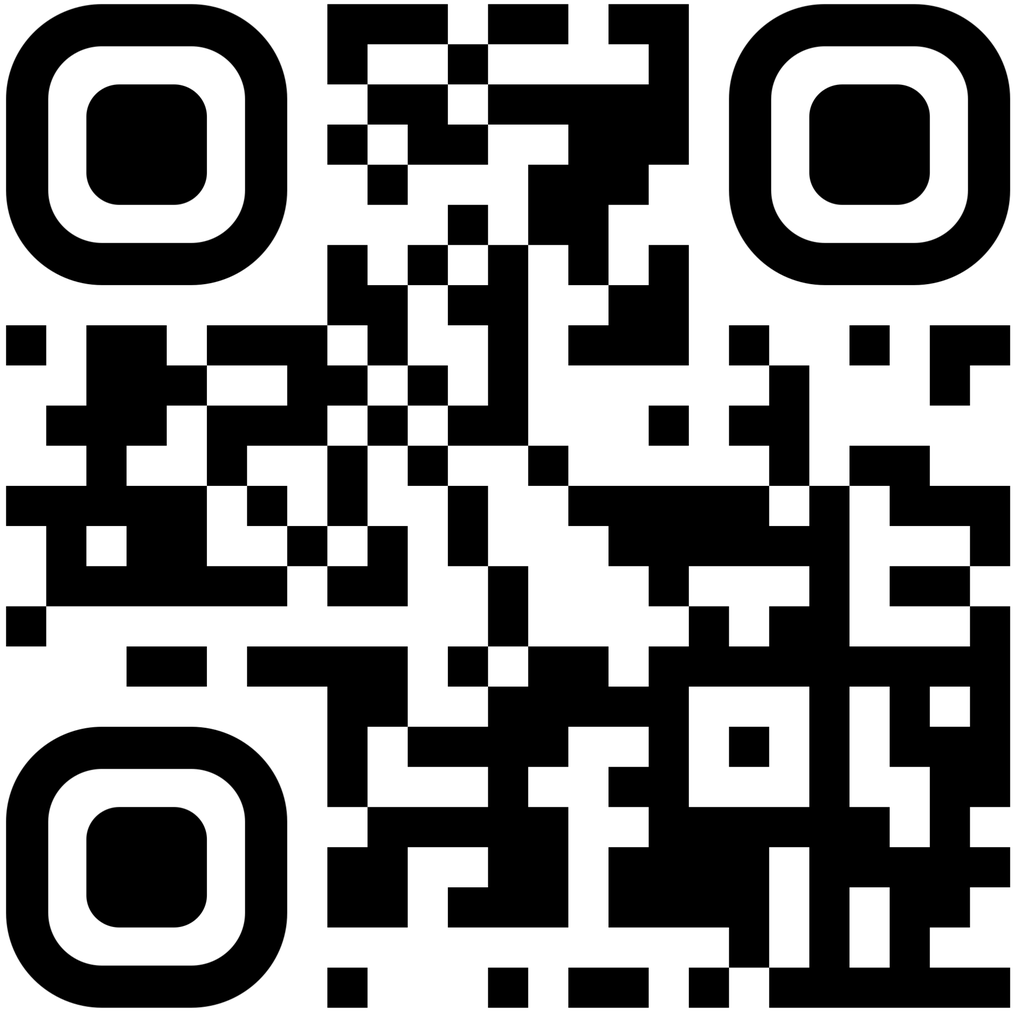
Marketplace

3

Продвинутая оркестрация

Мы открыты!

- Теперь без инвайтов
- Пробуйте и оставляйте отзывы



sourcecraft.dev

Спасибо за внимание!

Кирилл Сюзев
Технический лидер,
SourceCraft

Задайте вопросы спикеру
в чате мероприятия: