Чтобы прокачать тестирование производительности нужно всего лишь...

Егор
Романов

QA @Supabase

eromanov

egor@supabase.io

HEISENBUG

supabase

Егор Романов

eromanov

supabase

Пару слов обо мне

В детстве был благовоспитанным мальчиком, но связался с айти и покатился… Характер скверный. Женат, двое котов.

→ Работаю SDET 5 лет

→ Сервисы развертывания инфраструктуры

→ Отвечаю за качество в Supabase

Разрабатывайте быстрее,
фокусируясь на своем продукте

Database

В каждом проекте полноценная Postgres DB, реляционная база, которой доверяют больше всего.

Auth

Добавьте регистрацию и авторизацию, обезопасив данные с Row Level Security.

Realtime

Создавайте коллаборативные приложения, делясь, рассылая и подписываясь на обновления других пользователей или базы.

Storage

Храните, организуйте, и раздавайте большие объекты. Любое медиа, включая видео и изображения.

Functions

Пишите кастомный код без нужды создавать и масштабировать сервера.

Serverless APIs

Все данные из базы моментально доступны с помощью автоматически сгенерированных GraphQL и REST APIs.

supabase

Немного чисел

Инстансов Postgres DB запущено	Более 300 тысяч
Сообщений через Realtime	Порядка ½ миллиарда в неделю
Запросов к StorageAPI (не считая cache)	> 100 миллионов в неделю
Запросов к PostgREST	До 2 миллиардов в неделю

Данные на март 2023

supabase

О чем мы сегодня поговорим

- → Почему начали тестировать производительность
- → Какие этапы прошли
- → Небольшой инструмент запуска нагрузочных тестов
- → Масштабируем процесс горизонтально и вертикально
- → Демо: запускаем у себя

supabase

Почему начали тестировать производительность?

Зачем вообще это всё?

→ Сравнить несколько решений

→ Определить скорость, производительность, стабильность

→ Понять как масштабируется

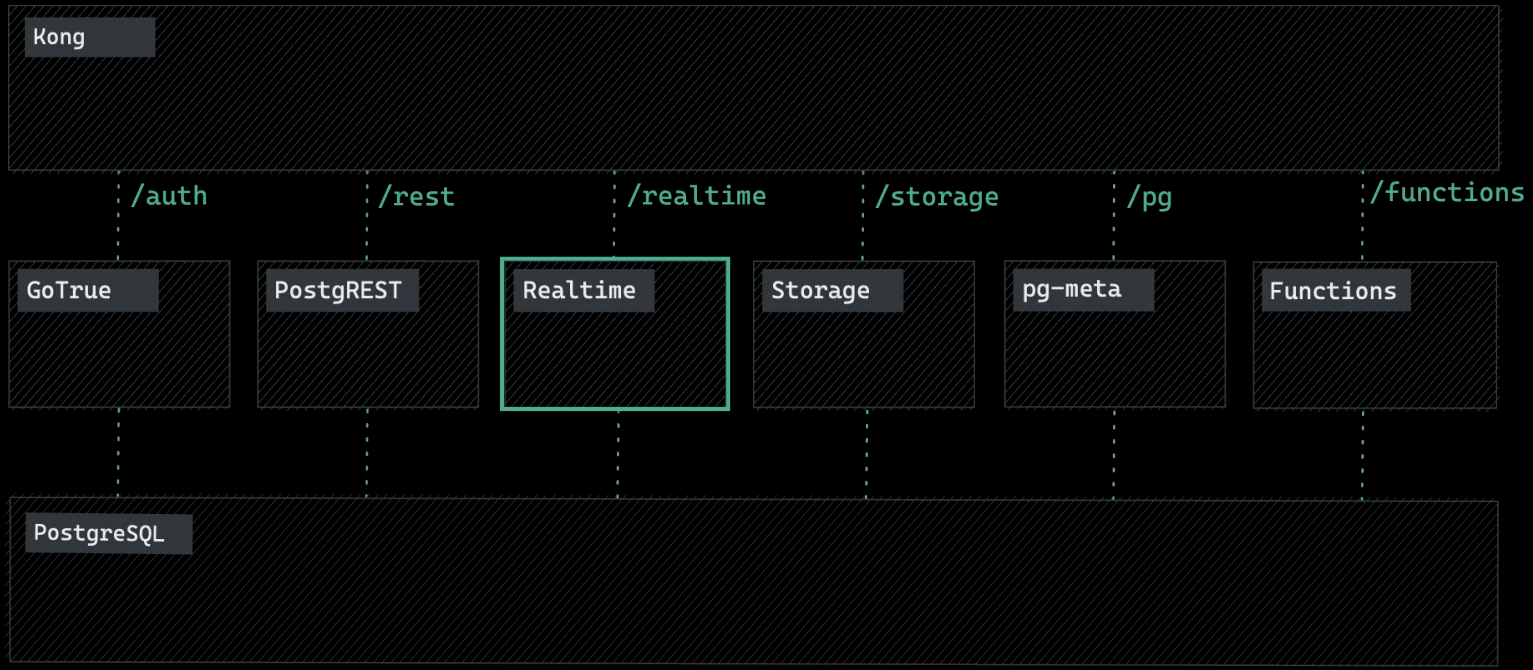
→ Найти возможные проблемы

→ Установить стандарты для приложения

→ Проверять на удовлетворение стандарту потом

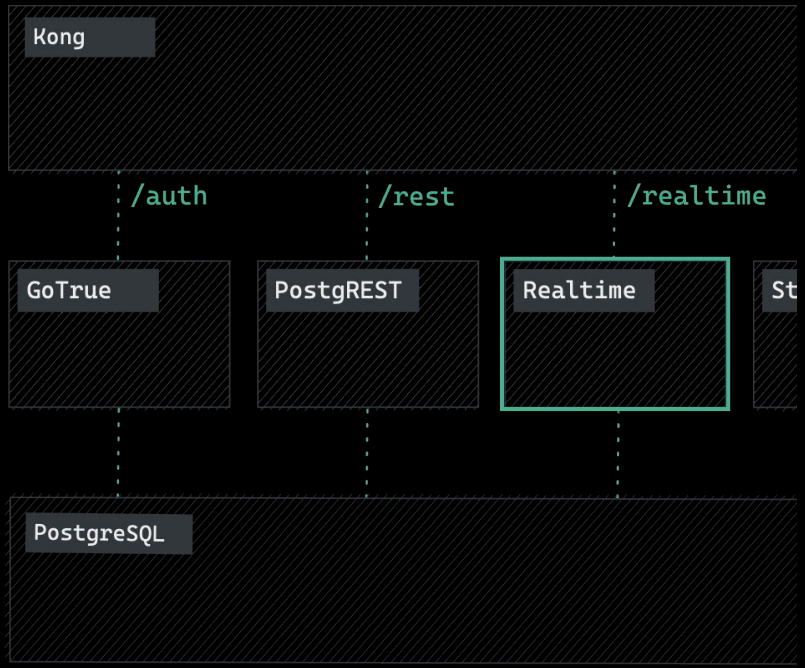
supabase

Архитектура



Сервис Realtime

WebSocket сервис для обмена сообщениями



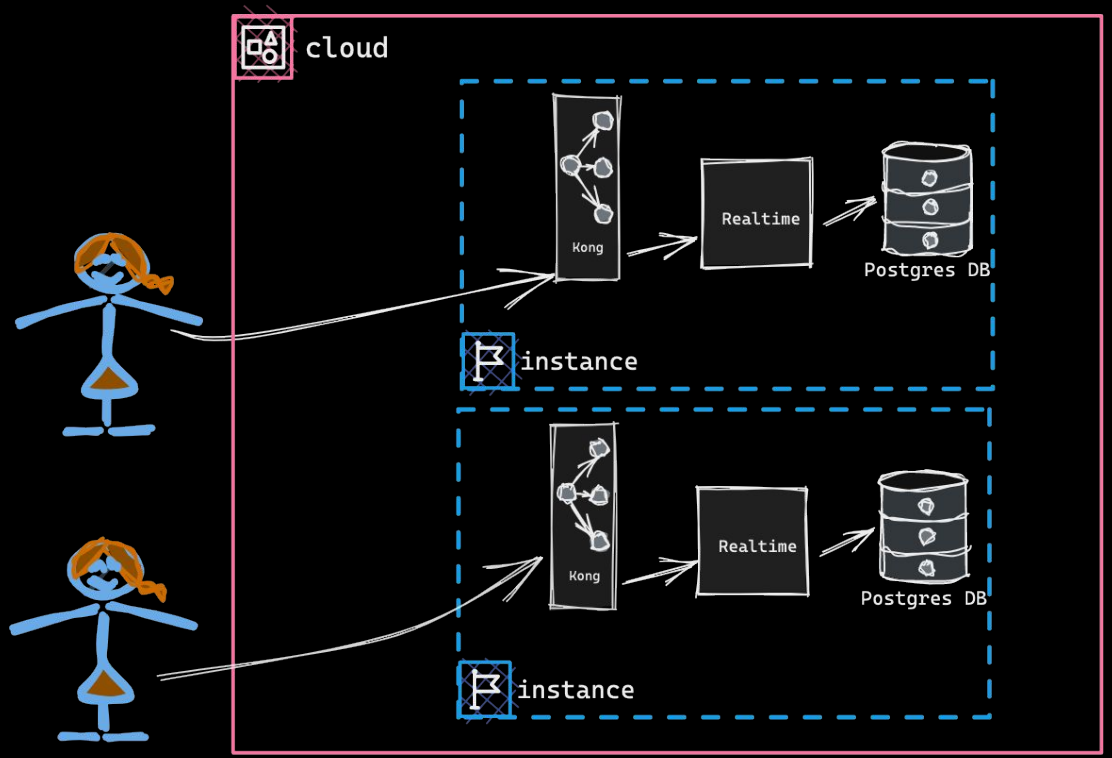
Что из себя представляет

- → написан на Elixir
- → стримит обновления прямо из Postgres DB
- → v2 - начал поддерживать RLS
- → v3 - добавили обмен сообщениями между пользователями через Broadcast и Presence
- → v3 - выносим из каждого проекта в один общий кластер

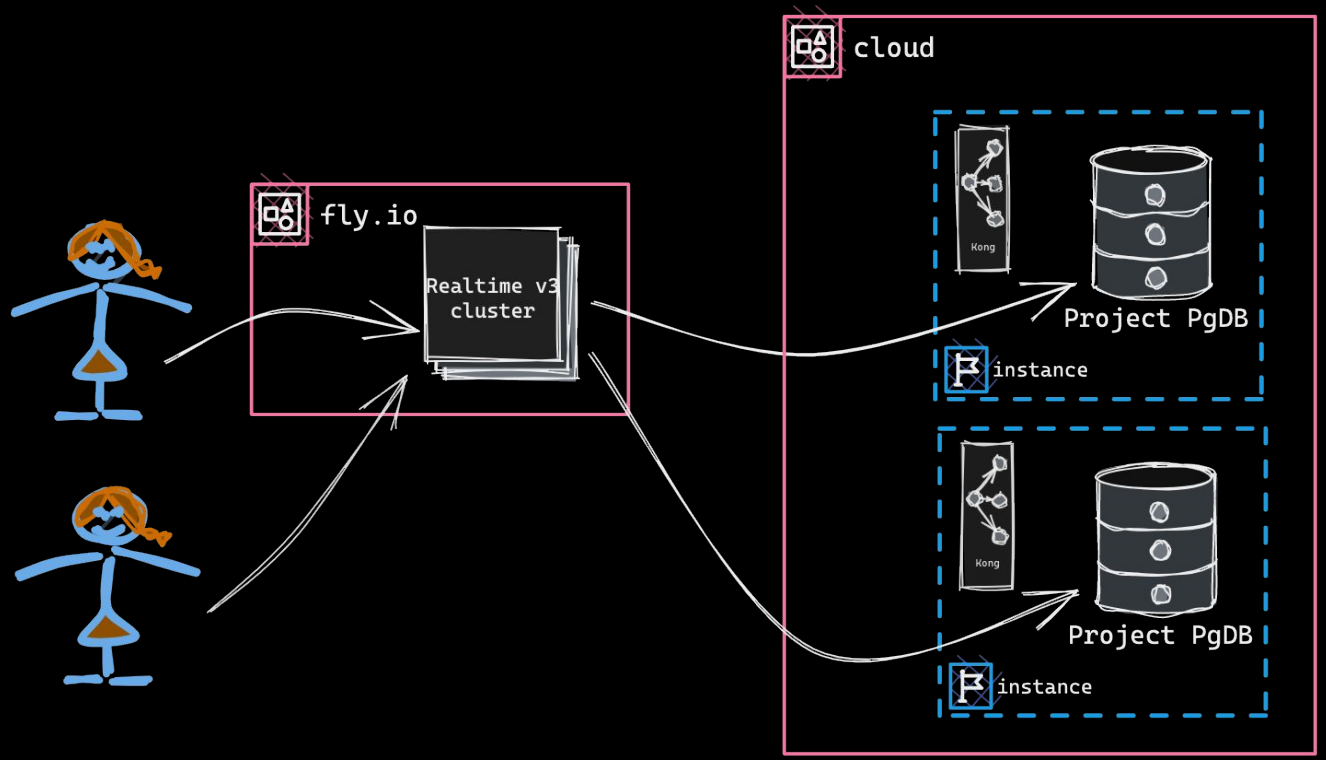
RLS (row level security) - в PostgreSQL это концепция, которая позволяет определять политики (policies), чтобы контролировать, как записи (rows) будут отображаться или управляться ролями или конечными пользователями. Это дополнительные фильтр-условия, применяемые к таблице во время исполнения запроса.

Broadcast - позволяет отправлять эфемерные сообщения между клиентами с низкой задержкой.
Presence - позволяет следить и синхронизировать shared state между клиентами (например онлайн статус).

Realtime v1 и v2



Realtime v3


cloud
fly.io
Realtime v3 cluster
Kong
Project PgDB
instance
Kong
Project PgDB
instance

HTTP vs WebSocket

HTTP	WebSocket
Модель запрос-ответ	Полный дуплекс
Короткоживущее соединение	Постоянное соединение
Запросы всегда от клиента	Оба могут посылать сообщения

supabase

Особенности тестирования WS и Realtime

→ Много постоянных коннектов к серверу
→ Чуть больше интересна задержка в доставке сообщений
→ Мало инструментов поддерживают WebSocket
→ Может потреблять больше ресурсов нагрузчиков
→ Разные скрипты для генерации сообщений и работы подписчиков
→ Несколько сценариев работы (database changes, broadcast, presence)

supabase

Первые шаги

k6

Плюсы

- → Порекомендовали профессионалы
- → JS для скриптов и golang для плагинов
- → Хорошая документация
- → OSS и большое комьюнити
- → WS из коробки, плагин для БД

Минусы

- → Нет GUI
- → Все еще молодой проект
- → Уже куплен grafana и возможно закручивание гаек

supabase

Нагружаем руками

Задача:

Сравнить 3 версии Realtime:

- Replication
- RLS
- Cluster

Архитектура:

Realtime v1 и v2

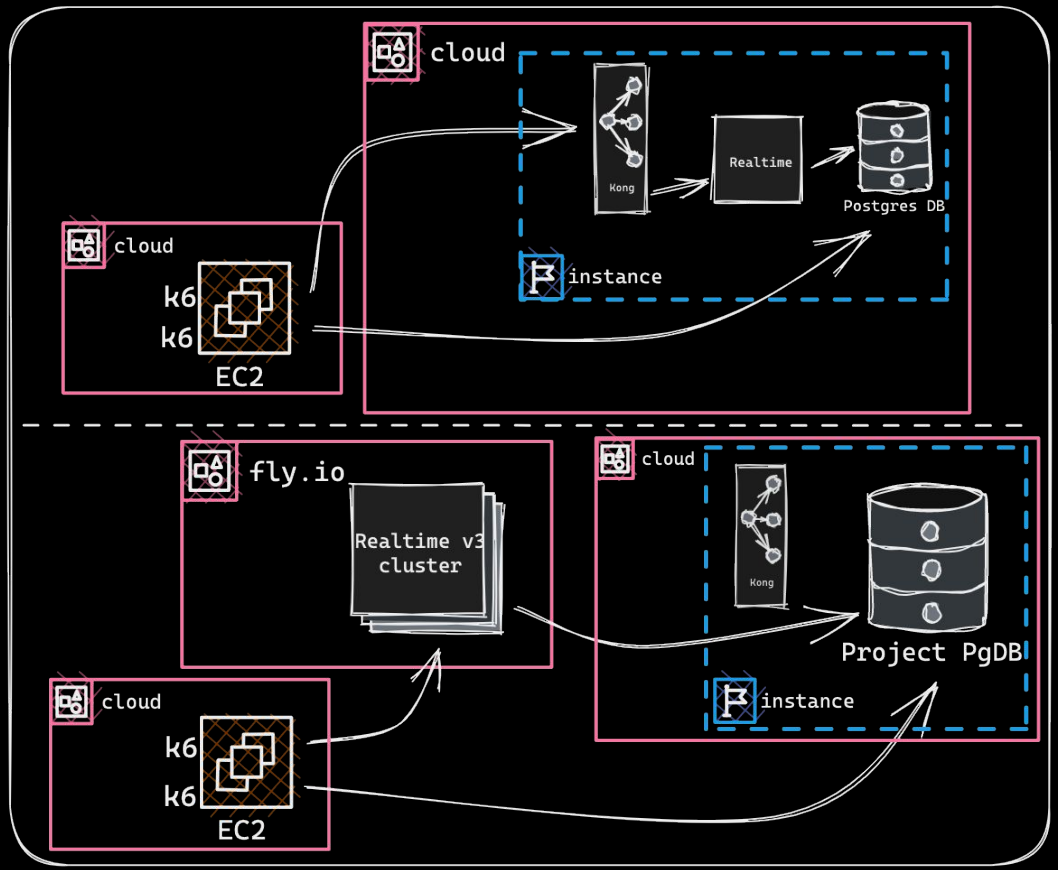
- Деплоятся рядом с базой данных

Realtime v3

- Деплоится как отдельный кластер в другом облаке

Нагрузчики:

- Просто несколько виртуалок в облаке



k6 output

```
✓ subscribed to realtime
✓ got realtime notification
✓ status is 101

checks.............: 100.00%  ✓ 3074821      ✗ 0
data_received......: 1.6 GB  5.1 MB/s
data_sent..........: 3.0 MB  9.4 kB/s
latency_trend......: avg=4577.483972 med=3275    p(99)=13696    p(95)=12436   p(0.1)=42
count=3073605
received_updates...: 3073603 9757.349166/s
vus................: 0       min=0         max=1000
vus_max............: 1000    min=1000      max=1000
ws_connecting......: avg=23.72ms     med=15.91ms p(99)=213.22ms p(95)=44.96ms p(0.1)=11.99ms
count=1220
ws_msgs_received...: 3086009 9796.732806/s
ws_msgs_sent.......: 11159   35.424959/s
ws_sessions........: 1220    3.872968/s
```

supabase

k6 output

```
✓ subscribed to realtime
✓ got realtime notification
✓ status is 101

checks.............: 100.00%  ✓ 3074821      ✗ 0
data_received......: 1.6 GB  5.1 MB/s
data_sent..........: 3.0 MB  9.4 kB/s
latency_trend......: avg=4577.483972 med=3275    p(99)=13696    p(95)=12436    p(0.1)=42
count=3073605
received_updates...: 3073603 9757.349166/s
vus................: 0       min=0         max=1000
vus_max............: 1000    min=1000      max=1000
ws_connecting......: avg=23.72ms     med=15.91ms p(99)=213.22ms p(95)=44.96ms p(0.1)=11.99ms
count=1220
ws_msgs_received...: 3086009 9796.732806/s
ws_msgs_sent.......: 11159   35.424959/s
ws_sessions........: 1220    3.872968/s
```

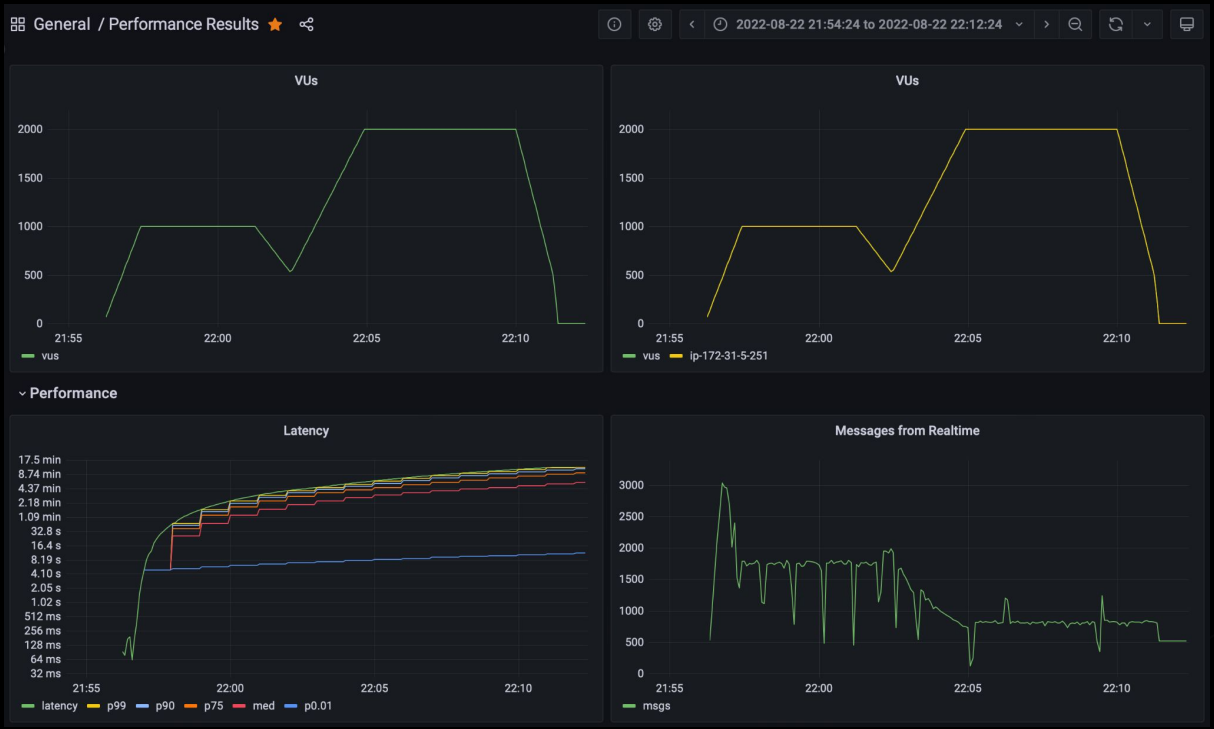
supabase

k6 output

```
✓ subscribed to realtime
✓ got realtime notification
✓ status is 101

checks.............: 100.00%  ✓ 3074821      ✗ 0
data_received......: 1.6 GB   5.1 MB/s
data_sent..........: 3.0 MB   9.4 kB/s
latency_trend......: avg=4577.483972 med=3275     p(99)=13696    p(95)=12436    p(0.1)=42
count=3073605
received_updates...: 3073603 9757.349166/s
vus................: 0        min=0         max=1000
vus_max............: 1000     min=1000      max=1000
ws_connecting......: avg=23.72ms     med=15.91ms p(99)=213.22ms p(95)=44.96ms p(0.1)=11.99ms
count=1220
ws_msgs_received...: 3086009 9796.732806/s
ws_msgs_sent.......: 11159   35.424959/s
ws_sessions........: 1220    3.872968/s
```

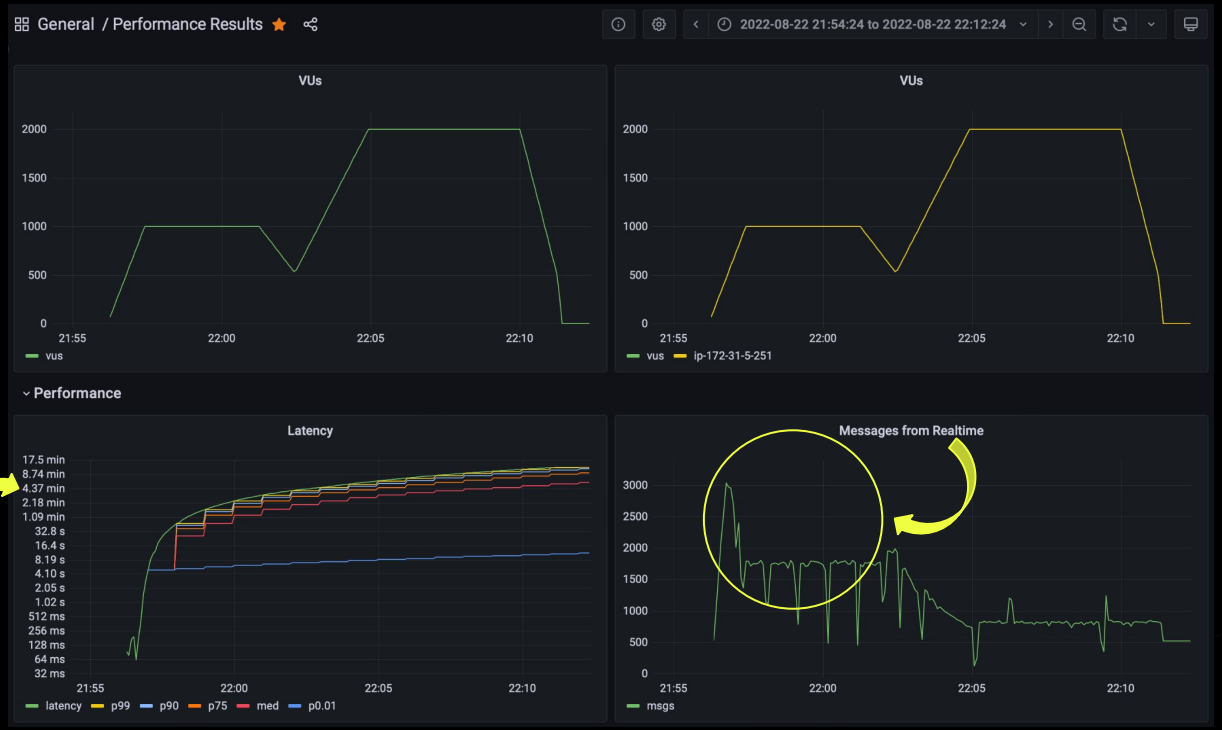
supabase

Grafana



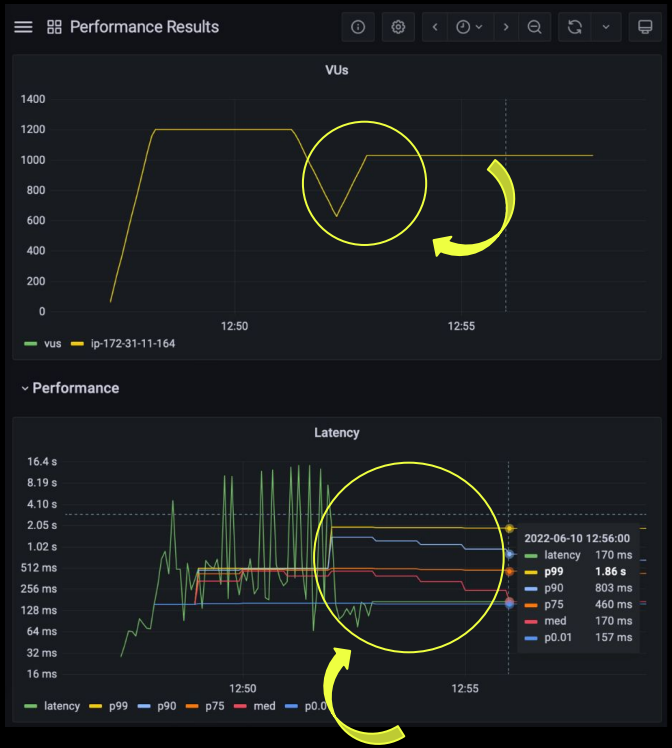
supabase

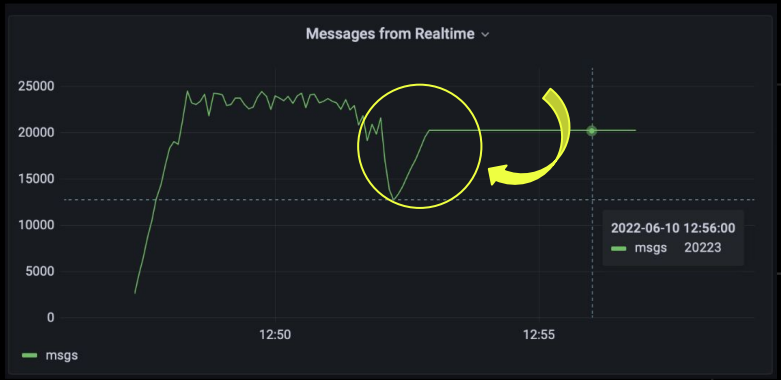
Слишком слабый нагрузчик



supabase

Исчезновение метрик





supabase

Telegraf спешит на помощь

→ Собирает метрики из разных источников: k6 скрипты, host (cpu, ram, network, etc.)

→ Мониторинг здоровья самого нагрузчика

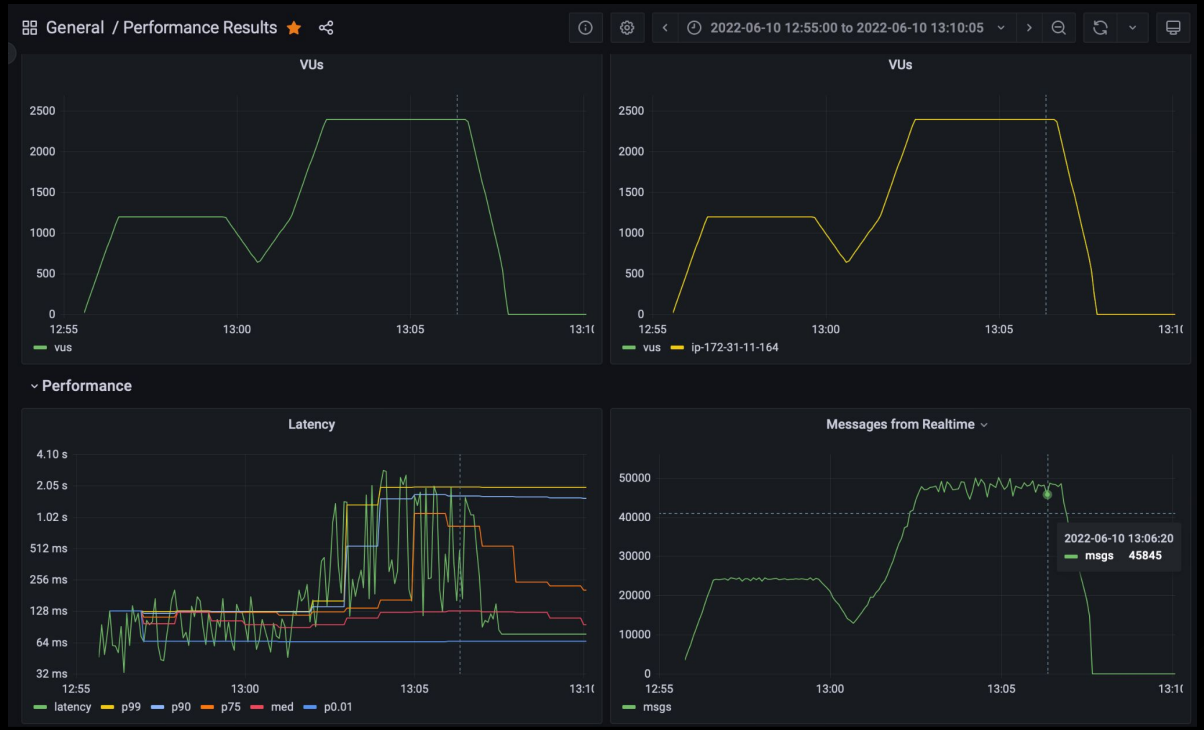
→ Интегрируется со множеством time series DB (Influx, VictoriaMetrics, Prometheus, etc.)

→ In-memory metric buffer

→ Pull & push чтобы собирать и отдавать метрики

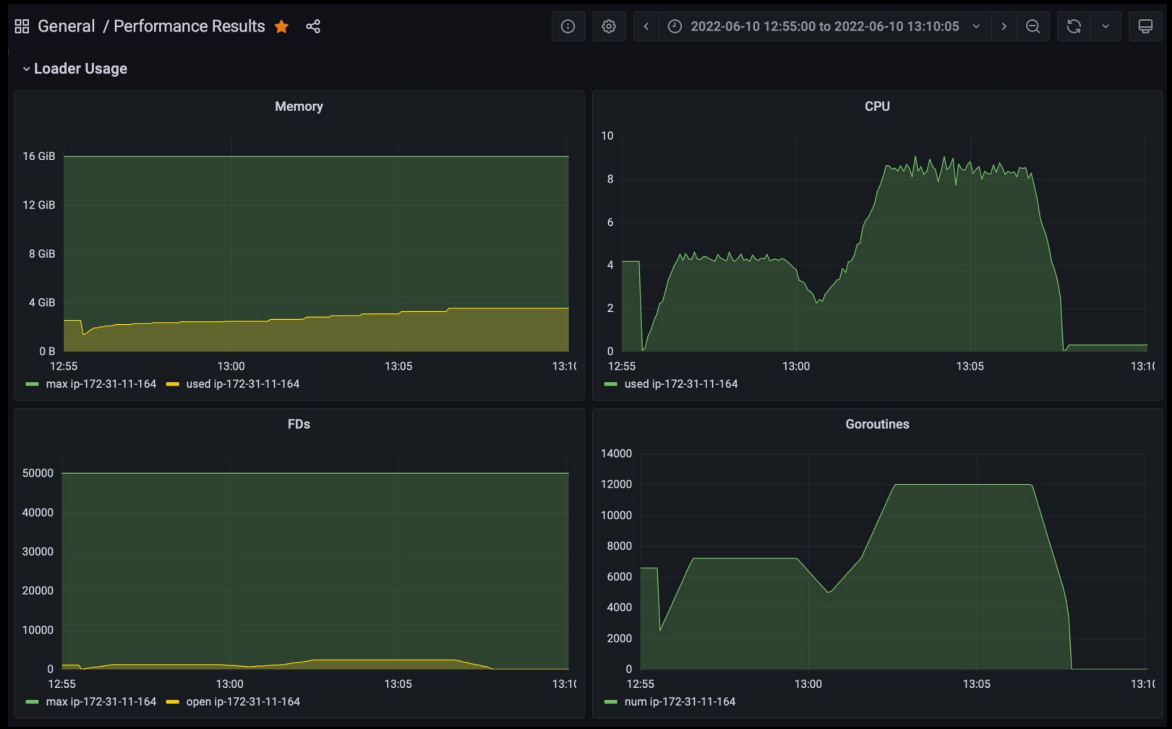
supabase

И наконец



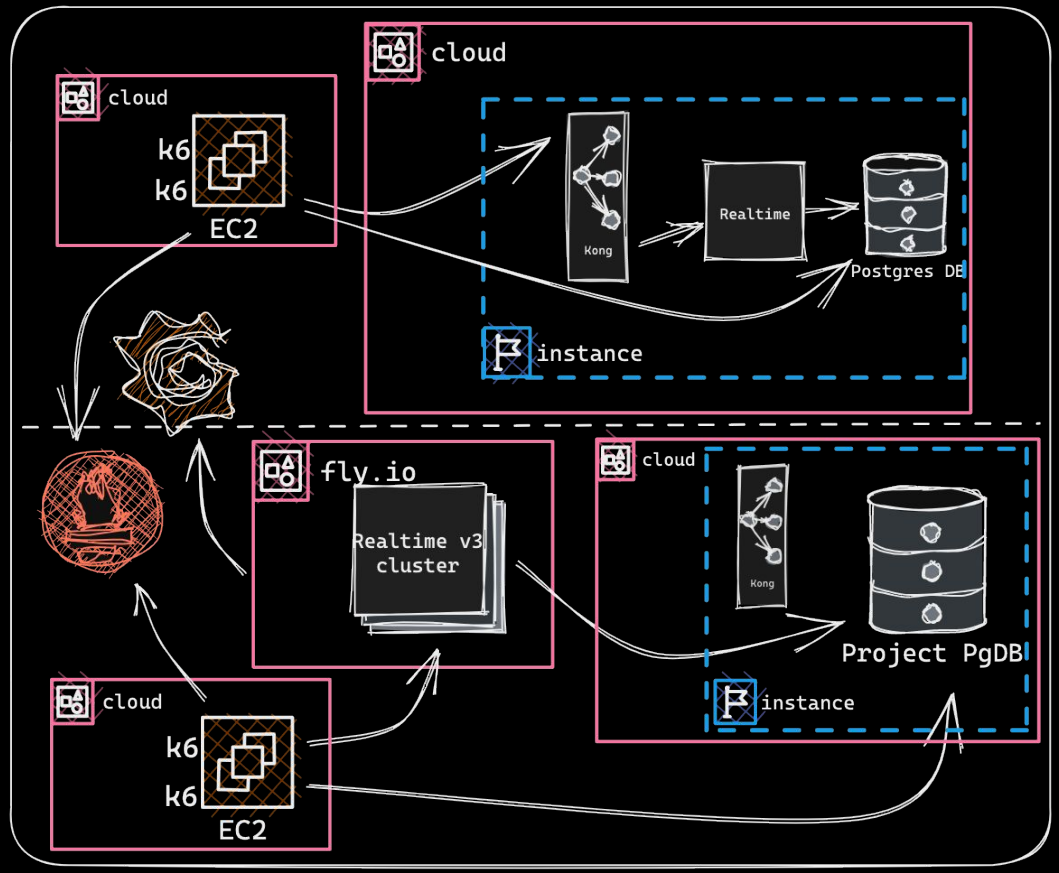
supabase

А что с нагрузчиком



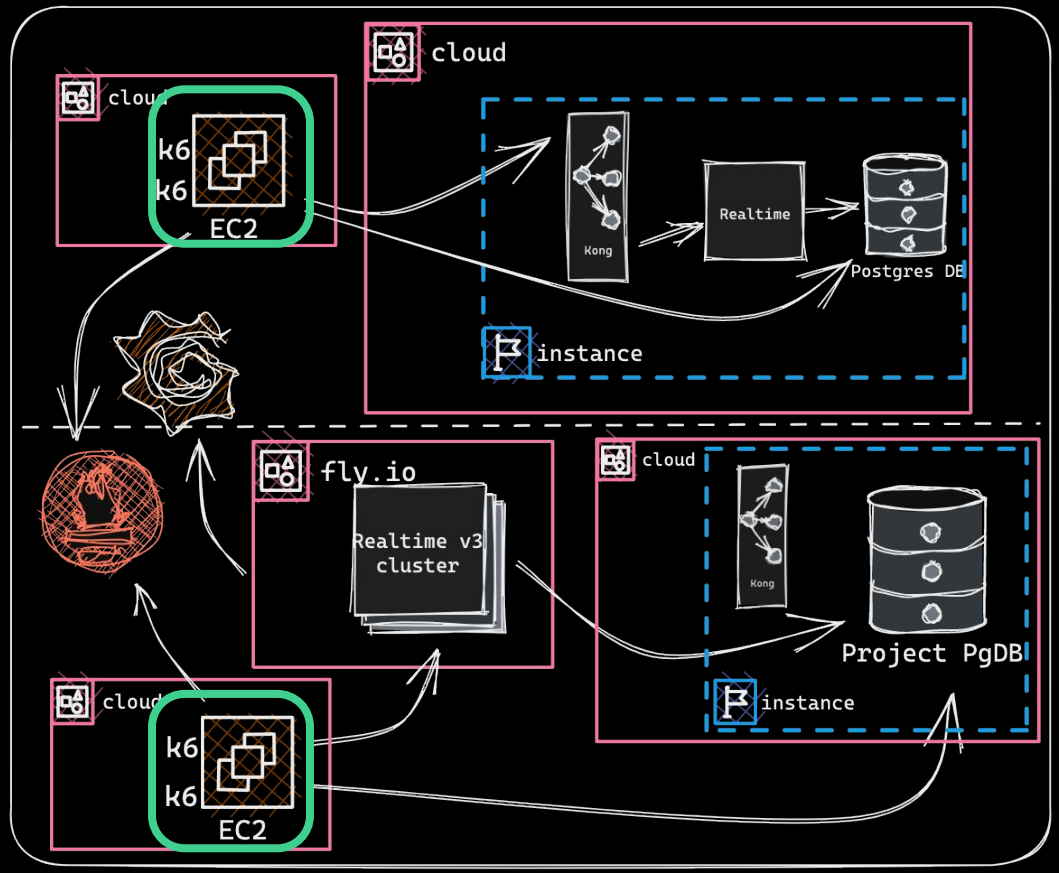
supabase

Всё вместе



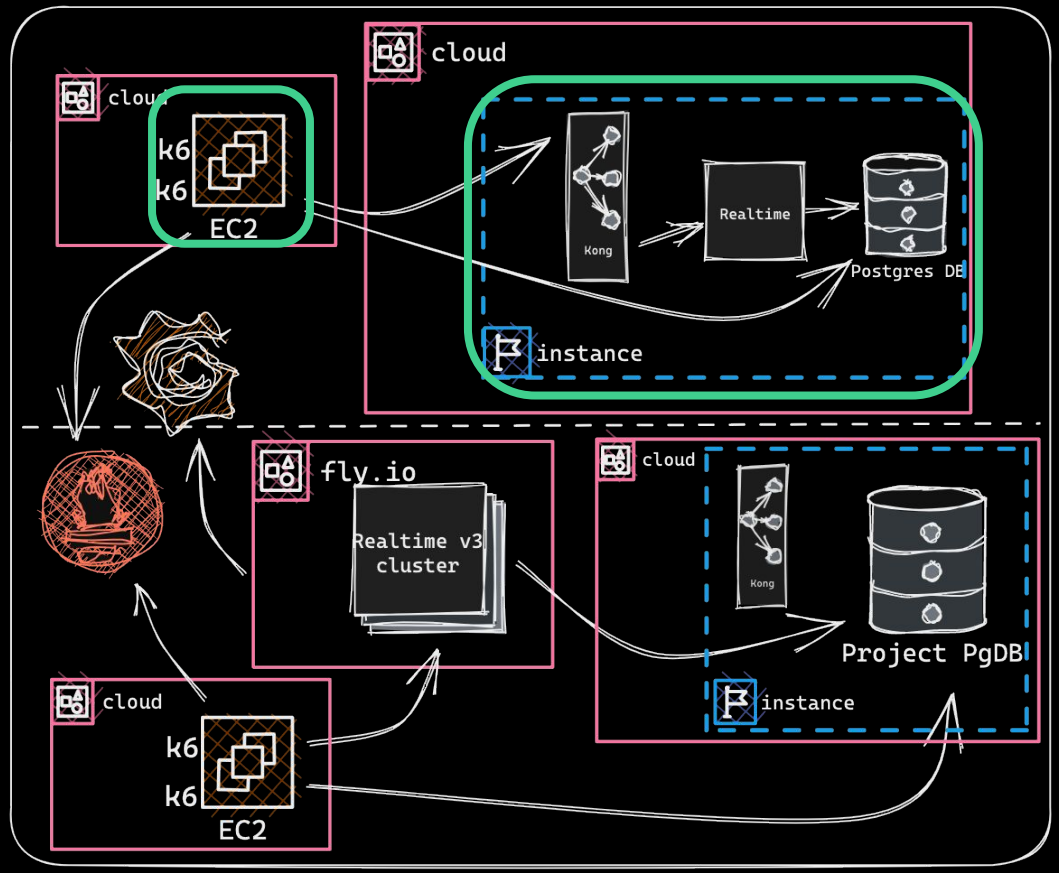
supabase

Всё вместе



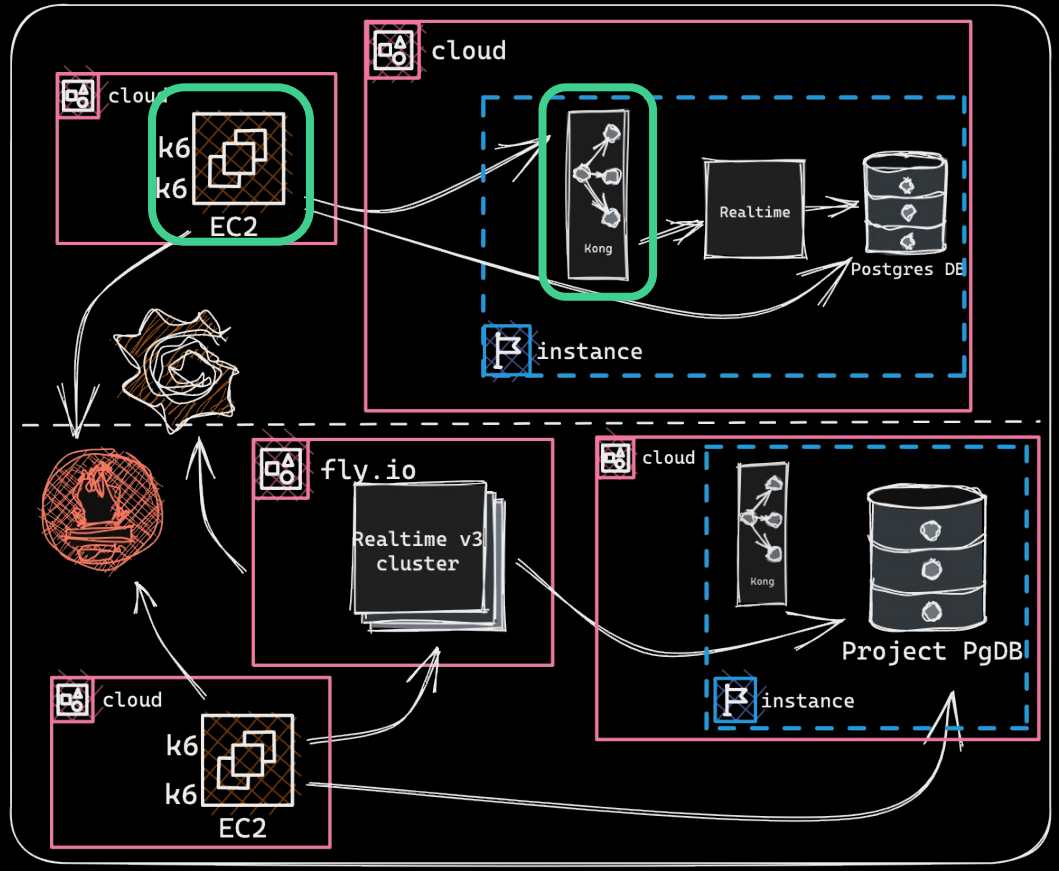
supabase

Всё вместе



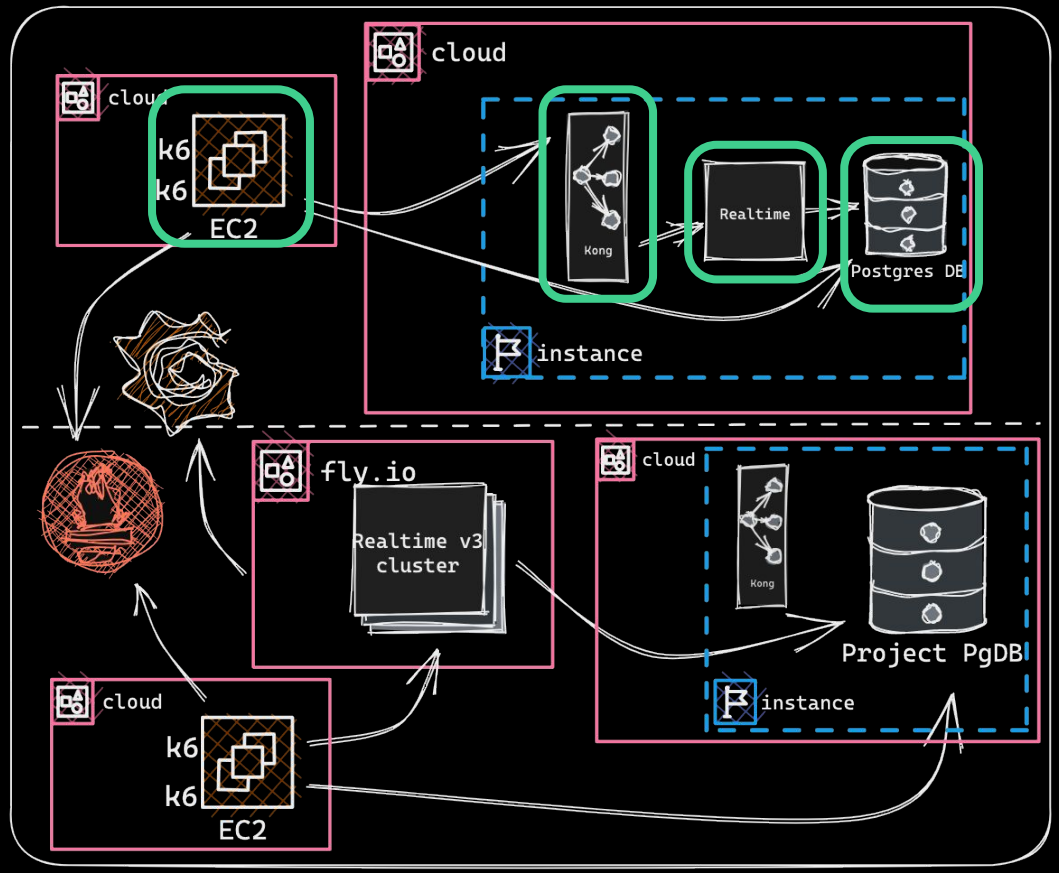
supabase

Всё вместе



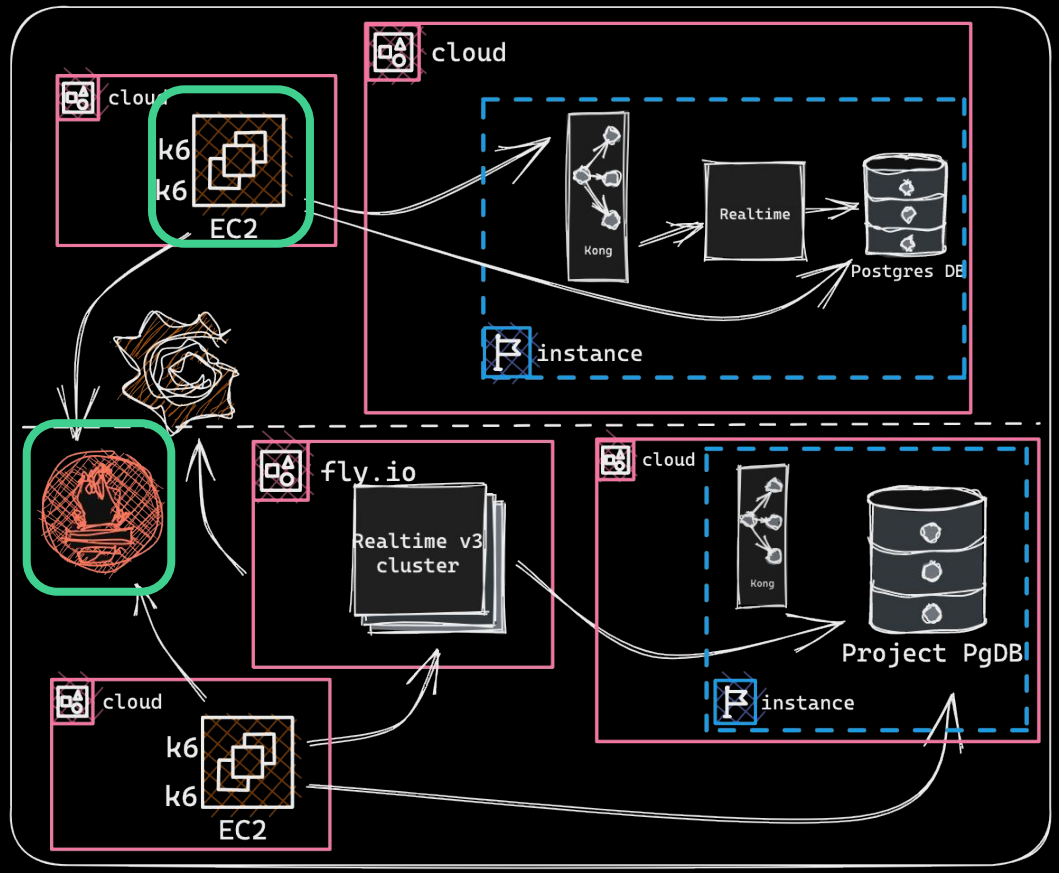
supabase

Всё вместе



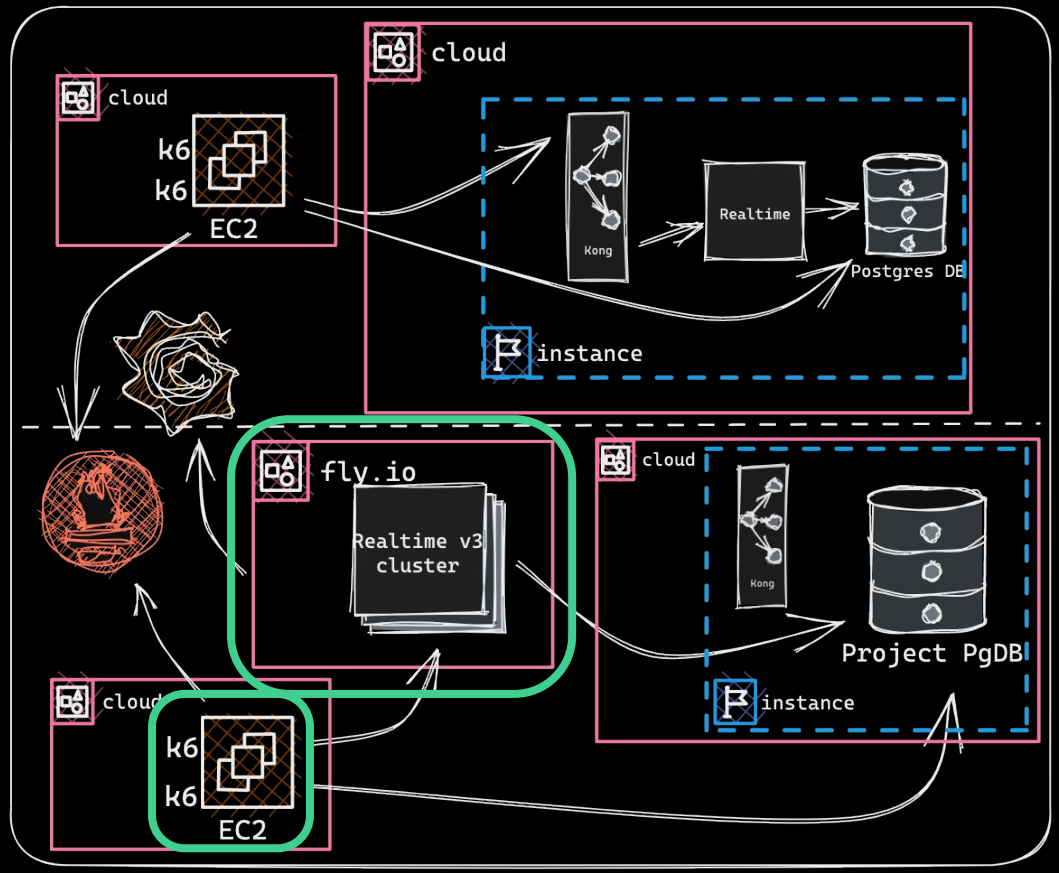
supabase

Всё вместе



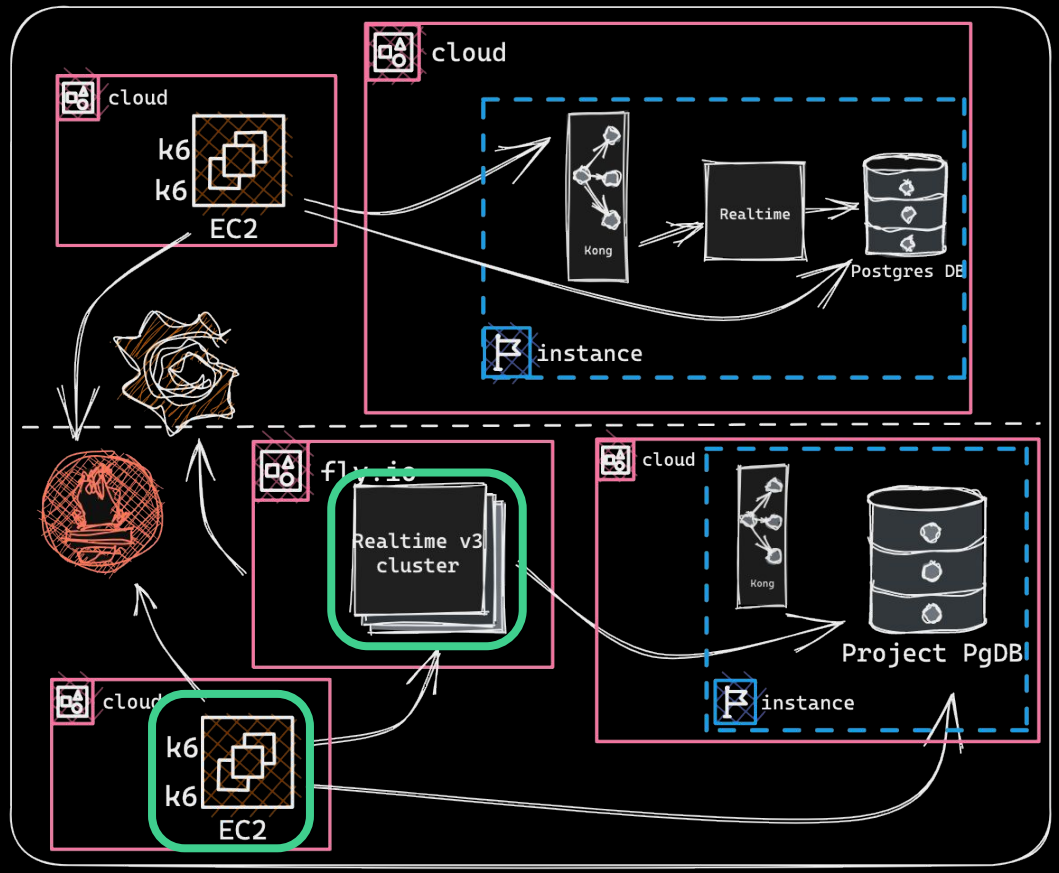
supabase

Всё вместе



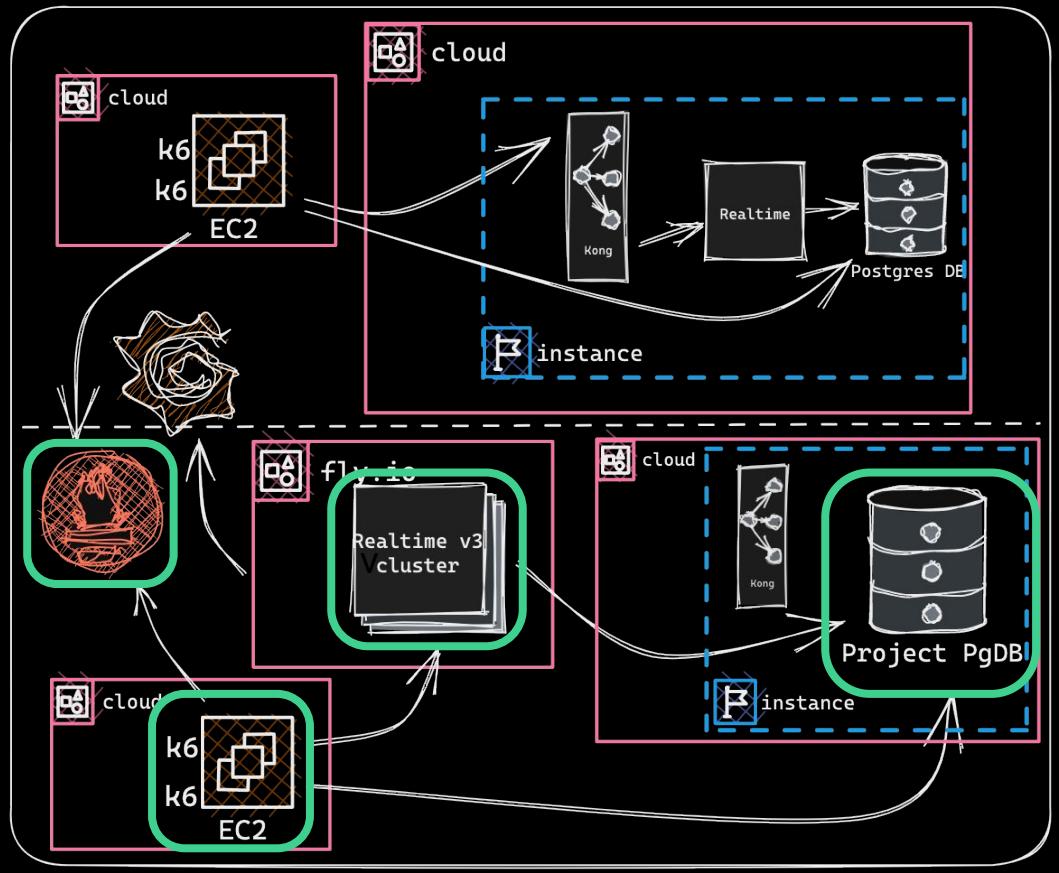
supabase

Всё вместе



supabase

Всё вместе



supabase

Зачем вообще это всё было?

→ Сравнить несколько решений

→ Определить скорость, производительность, стабильность

→ Понять как масштабируется

→ Найти возможные проблемы

→ Установить стандарты для приложения

→ Проверять на удовлетворение стандарту потом

supabase

Зачем вообще это всё было?

→ Сравнить несколько решений

→ Определить скорость, производительность, стабильность ✅

→ Понять как масштабируется

→ Найти возможные проблемы

→ Установить стандарты для приложения

→ Проверять на удовлетворение стандарту потом

supabase

История запусков

General / Realtime Benchmarks Runs

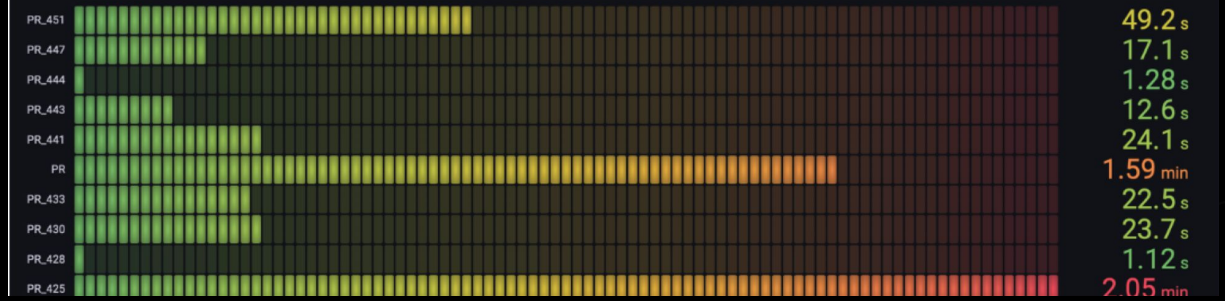
Last 14 days | 1m

scenario | All

Test run time range

testrun	Timestamp (min)	Timestamp (max)	started	finished
PR_451	2023-01-19 07:28:00	2023-01-19 07:46:00	1674102180000	1674103565000
PR_450	2023-01-19 06:42:00	2023-01-19 07:00:00	1674099420000	1674100805000
PR_447	2023-01-19 02:10:00	2023-01-19 06:40:00	1674083100000	1674099605000
PR_444	2023-01-18 22:46:00	2023-01-18 23:04:00	1674070860000	1674072245000
PR_443	2023-01-18 22:06:00	2023-01-18 22:24:00	1674068460000	1674069845000
PR_441	2023-01-18 18:32:00	2023-01-18 18:50:00	1674055620000	1674057005000
PR_433	2023-01-12 21:38:00	2023-01-12 21:56:00	1673548380000	1673549765000

Latency p95



supabase

Зачем вообще это всё было?

→ Сравнить несколько решений ✅

→ Определить скорость, производительность, стабильность ✅

→ Понять как масштабируется ✅

→ Найти возможные проблемы

→ Установить стандарты для приложения

→ Проверять на удовлетворение стандарту потом

supabase

Зачем вообще это всё было?

→ Сравнить несколько решений ✅

→ Определить скорость, производительность, стабильность ✅

→ Понять как масштабируется ✅

→ Найти возможные проблемы ✅

→ Установить стандарты для приложения ✅

→ Проверять на удовлетворение стандарту потом

supabase

Получается мы все решили?

→ Сравнить несколько решений ✅
→ Определить скорость, производительность, стабильность ✅
→ Понять как масштабируется ✅
→ Найти возможные проблемы ✅
→ Установить стандарты для приложения ✅

→ Проверять на удовлетворение стандарту потом

supabase

Или нет...

→ Сравнить несколько решений ✅
→ Определить скорость, производительность, стабильность ✅
→ Понять как масштабируется ✅
→ Найти возможные проблемы ✅
→ Установить стандарты для приложения ✅

→ Проверять на удовлетворение стандарту потом

→ Можно не остановить виртуалку или не остановить SUT

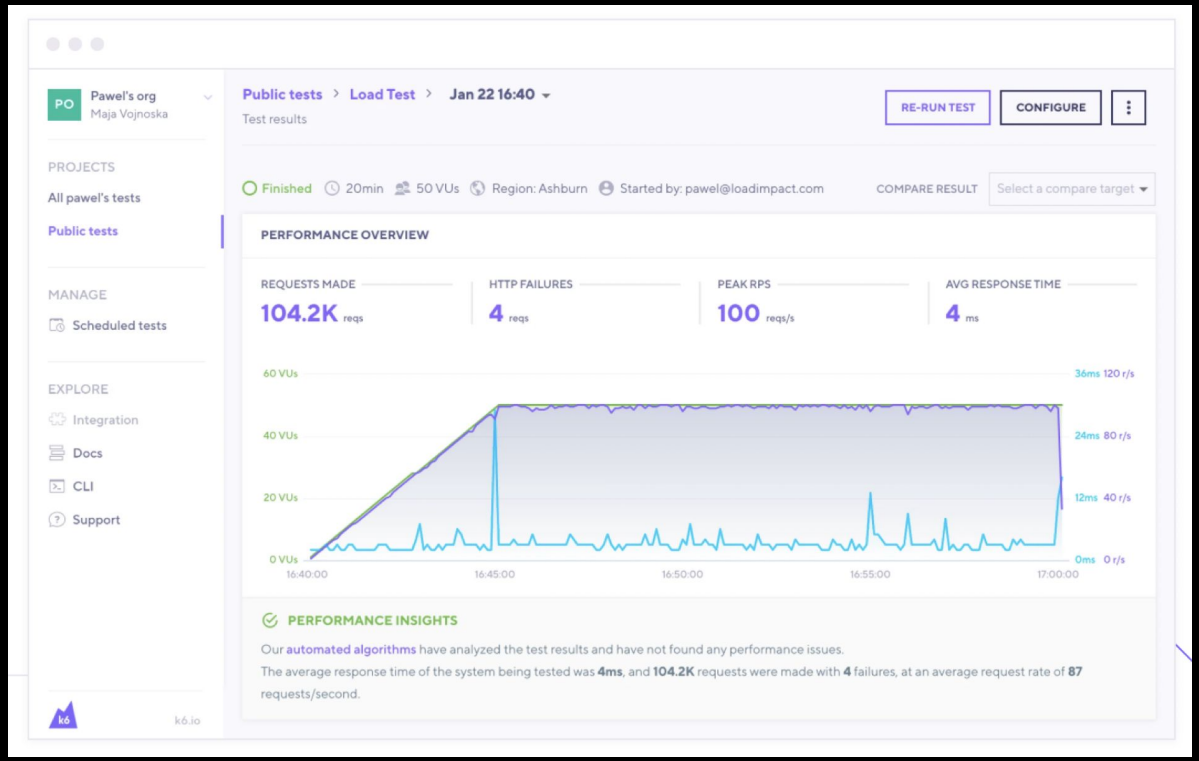
→ Сложно делить ресурсы с разработчиками

→ Неудобно обновлять код на нескольких нагрузчиках

→ История есть, но неудобная

Как бы нам все это
автоматизировать

k6 cloud


PO
Pawel's org
Maja Vojnoska
PROJECTS
All pawel's tests
Public tests
MANAGE
Scheduled tests
EXPLORE
Integration
Docs
CLI
Support
k6.io
Public tests > Load Test > Jan 22 16:40
Test results
RE-RUN TEST
CONFIGURE
Finished
20min
50 VUs
Region: Ashburn
Started by: pawel@loadimpact.com
COMPARE RESULT
Select a compare target
PERFORMANCE OVERVIEW
REQUESTS MADE
104.2K reqs
HTTP FAILURES
4 reqs
PEAK RPS
100 reqs/s
AVG RESPONSE TIME
4 ms
60 VUs
40 VUs
20 VUs
0 VUs
36ms 120 r/s
24ms 80 r/s
12ms 40 r/s
0ms 0 r/s
16:40:00
16:45:00
16:50:00
16:55:00
17:00:00
PERFORMANCE INSIGHTS
Our automated algorithms have analyzed the test results and have not found any performance issues.
The average response time of the system being tested was 4ms, and 104.2K requests were made with 4 failures, at an average request rate of 87 requests/second.

k6 cloud

Плюсы

- → Подходит для больших распределенных тестов
- → Хороший UX, работа с графиками
- → Интегрируется с CICD
- → Удобно сравнивать прогоны
- → Инструменты для коллаборации

Минусы

- → Цена
- → Плагины только на enterprise подписке
- → Небольшая история

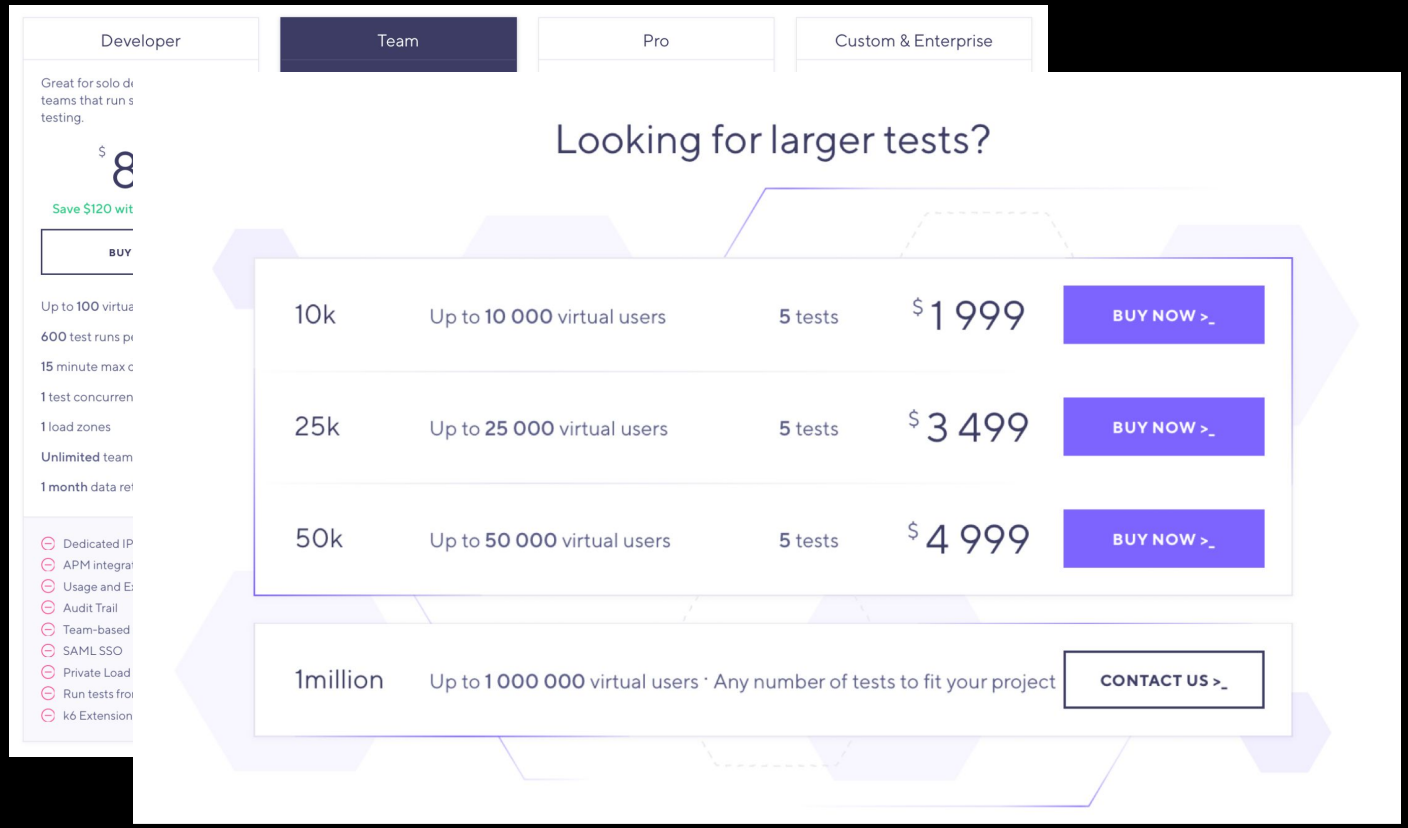
Developer	Team	Pro	Custom & Enterprise
Great for solo developers and teams that run small scale cloud testing.	Best for teams that run medium scale load tests as part of their CI & QA process.	For larger engineering teams that perform more frequent load testing with higher levels of traffic.	For companies with higher testing requirements or larger traffic.
$ 89 /mo	$ 424 /mo	$ 1199 /mo	
Save $120 with an annual plan	Save $900 with an annual plan	Save $3,600 with an annual plan	
BUY NOW >_	BUY NOW >_	BUY NOW >_	CONTACT US >_
Up to **100** virtual users	Up to **1000** virtual users	Up to **3000** virtual users	**Custom terms** based on your organization's unique needs.
600 test runs per year	**900** test runs per year	**1200** test runs per year	**Tiered plans.** Configure different virtual users and max duration for various types of tests.
15 minute max duration	**30** minute max duration	**60** minute max duration	Custom invoicing and payment options. Security & compliance review.
1 test concurrency	**2** test concurrency	**4** test concurrency	Starts at **$25000/annually**
1 load zones	**2** load zones	**10** load zones	
Unlimited team members	**Unlimited** team members	**Unlimited** team members	
1 month data retention	**1 month** data retention	**3 months** data retention	
⊖ Dedicated IPs ($30/IP/month)	⊕ Dedicated IPs ($30/IP/month)	⊕ Dedicated IPs ($30/IP/month)	⊕ Dedicated IPs ($30/IP/month)
⊖ APM integration	⊕ APM integration	⊕ APM integration	⊕ APM integration
⊖ Usage and Executive Reports	⊖ Usage and Executive Reports	⊕ Usage and Executive Reports	⊕ Usage and Executive Reports
⊖ Audit Trail	⊖ Audit Trail	⊕ Audit Trail	⊕ Audit Trail
⊖ Team-based access control	⊖ Team-based access control	⊕ Team-based access control	⊕ Team-based access control
⊖ SAML SSO	⊖ SAML SSO	⊖ SAML SSO	⊕ SAML SSO
⊖ Private Load Zones	⊖ Private Load Zones	⊖ Private Load Zones	⊕ Private Load Zones
⊖ Run tests from your K8s Cluster	⊖ Run tests from your K8s Cluster	⊖ Run tests from your K8s Cluster	⊕ Run tests from your K8s Cluster
⊖ k6 Extensions in k6 Cloud	⊖ k6 Extensions in k6 Cloud	⊖ k6 Extensions in k6 Cloud	⊕ k6 Extensions in k6 Cloud

supabase

k6 cloud pricing

Developer	Team	Pro	Custom & Enterprise
Great for solo developers and teams that run small scale cloud testing.	Best for teams that run medium scale load tests as part of their CI & QA process.	For larger engineering teams that perform more frequent load testing with higher levels of traffic.	For companies with higher testing requirements or larger traffic.
$ 89 /mo	$ 424 /mo	$ 1199 /mo	
Save $120 with an annual plan	Save $900 with an annual plan	Save $3,600 with an annual plan	
BUY NOW >_	BUY NOW >_	BUY NOW >_	CONTACT US >_
Up to **100** virtual users	Up to **1000** virtual users	Up to **3000** virtual users	**Custom terms** based on your organization's unique needs.
600 test runs per year	**900** test runs per year	**1200** test runs per year	**Tiered plans.** Configure different virtual users and max duration for various types of tests.
15 minute max duration	**30** minute max duration	**60** minute max duration	Custom invoicing and payment options. Security & compliance review.
1 test concurrency	**2** test concurrency	**4** test concurrency	Starts at **$25000/annually**
1 load zones	**2** load zones	**10** load zones	
Unlimited team members	**Unlimited** team members	**Unlimited** team members	
1 month data retention	**1 month** data retention	**3 months** data retention	
⊖ Dedicated IPs ($30/IP/month)	⊕ Dedicated IPs ($30/IP/month)	⊕ Dedicated IPs ($30/IP/month)	⊕ Dedicated IPs ($30/IP/month)
⊖ APM integration	⊕ APM integration	⊕ APM integration	⊕ APM integration
⊖ Usage and Executive Reports	⊖ Usage and Executive Reports	⊕ Usage and Executive Reports	⊕ Usage and Executive Reports
⊖ Audit Trail	⊖ Audit Trail	⊕ Audit Trail	⊕ Audit Trail
⊖ Team-based access control	⊖ Team-based access control	⊕ Team-based access control	⊕ Team-based access control
⊖ SAML SSO	⊖ SAML SSO	⊖ SAML SSO	⊕ SAML SSO
⊖ Private Load Zones	⊖ Private Load Zones	⊖ Private Load Zones	⊕ Private Load Zones
⊖ Run tests from your K8s Cluster	⊖ Run tests from your K8s Cluster	⊖ Run tests from your K8s Cluster	⊕ Run tests from your K8s Cluster
⊖ k6 Extensions in k6 Cloud	⊖ k6 Extensions in k6 Cloud	⊖ k6 Extensions in k6 Cloud	⊕ k6 Extensions in k6 Cloud

k6 cloud pricing



Как же нам решить оставшиеся проблемы?

→ Проверять на удовлетворение стандарту потом

→ Можно не остановить виртуалку или не остановить SUT

→ Сложно делить ресурсы с разработчиками

→ Неудобно обновлять код на нескольких нагрузчиках

→ История есть, но неудобная

supabase

Свой Велосипед!

supabase

Свое приложение

→ Запускать виртуалки для создания нагрузки

→ Запускать приложение для тестирования

→ Удобная работа с историей запусков

→ Отправлять актуальный код на нагрузчики

→ Управлять секретами и переменными

→ Интеграция с CI

→ Не зависеть от вендоров

→ Быстро и легко в разработке

supabase

Архитектура

Приложение на основе Pocketbase

→ хранит Summaries, мета информацию о запусках и секреты в sqlite

→ storage для кода тестов и инфры

→ кастомные ручки чтобы запускать тесты

→ встроенная админка для управления

Используем Terraform для развертывания SUT и нагрузчиков

Остальное входит в состав terraform скриптов и может быть заменено:

→ common package для k6, чтобы отправлять summary после теста в приложение

→ TF модули для развертывания SUT

→ VM template с golang и telegraf

→ Grafana и Prom через Grafana Cloud

Pocketbase - это open-source backend на golang, в который входят SQLite, realtime подписка, управление пользователями, дашборд админка, простой автоматический REST-ish API для данных. Огромный плюс - framework mode.

Terraform - это DSL и инструмент для создания и управления инфраструктурой as code. Может работать с огромным количеством облачных провайдеров и сервисов.

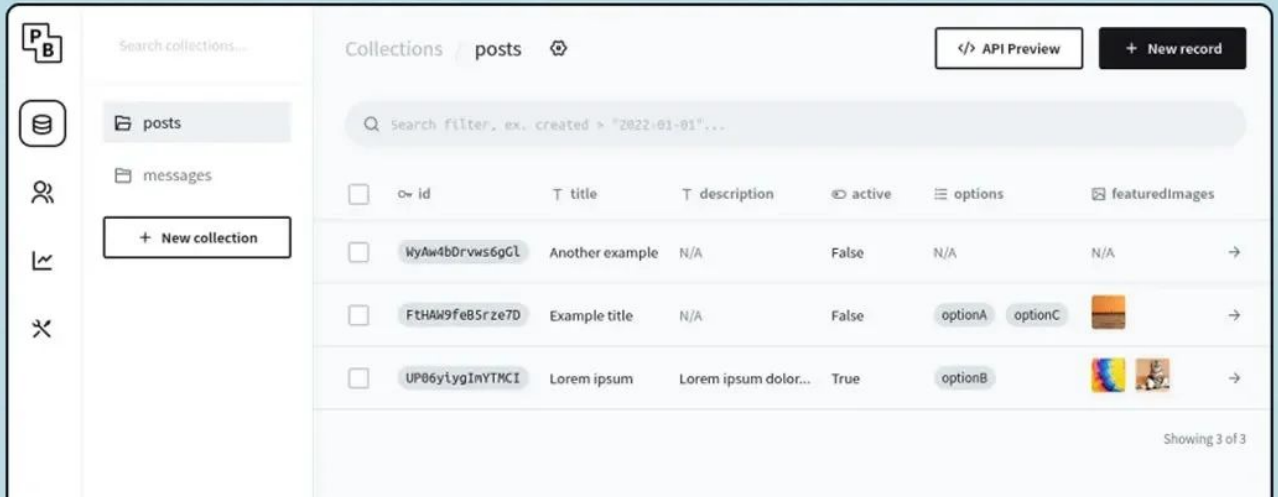
supabase

pocketbase

pocketbase.io

PocketBase

Open Source backend in **1** file



terraform

terraform.io

Deliver infrastructure as code

Terraform codifies cloud APIs into declarative configuration files.

Архитектура

Приложение на основе Pocketbase

→ хранит Summaries, мета информацию о запусках и секреты в sqlite

→ storage для кода тестов и инфры

→ кастомные ручки чтобы запускать тесты

→ встроенная админка для управления

Используем Terraform для развертывания SUT и нагрузчиков

Остальное входит в состав terraform скриптов и может быть заменено:

→ common package для k6, чтобы отправлять summary после теста в приложение

→ TF модули для развертывания SUT

→ VM template с golang и telegraf

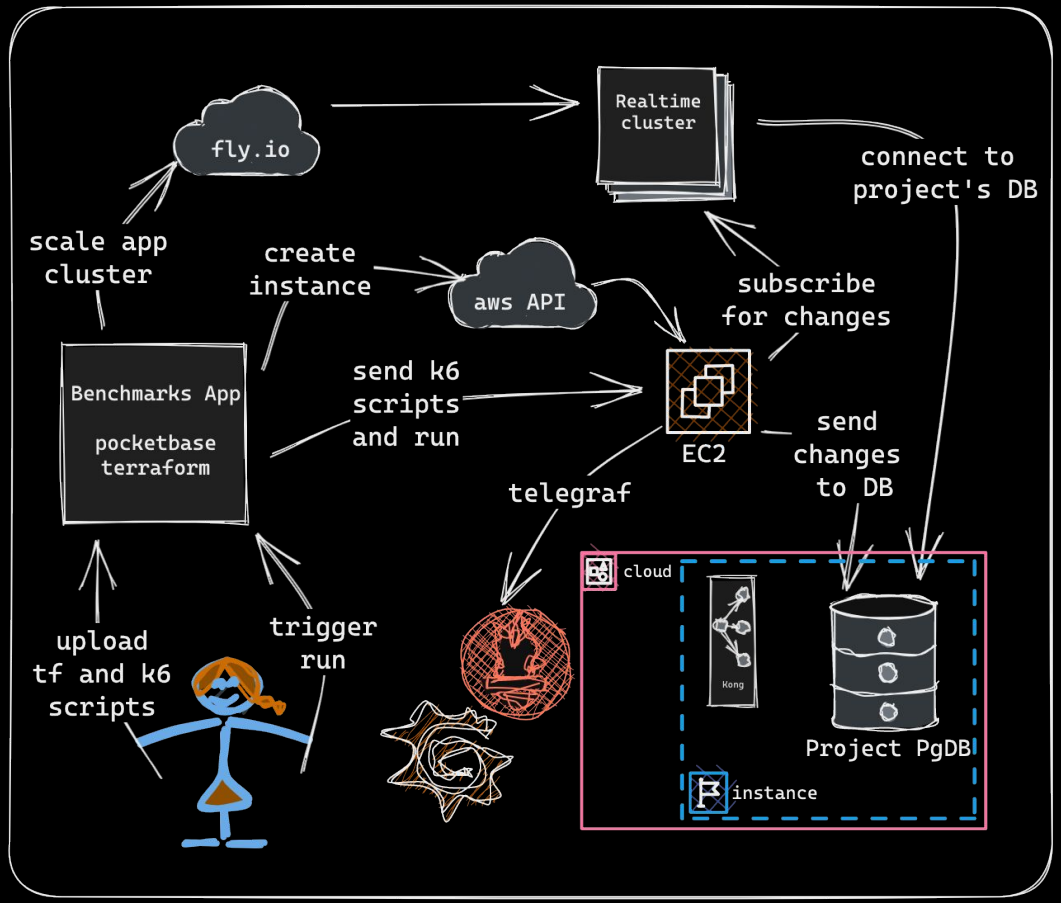
→ Grafana и Prom через Grafana Cloud

Pocketbase - это open-source backend на golang, в который входят SQLite, realtime подписка, управление пользователями, дашборд админка, простой автоматический REST-ish API для данных. Огромный плюс - framework mode.

Terraform - это DSL и инструмент для создания и управления инфраструктурой as code. Может работать с огромным количеством облачных провайдеров и сервисов.

supabase

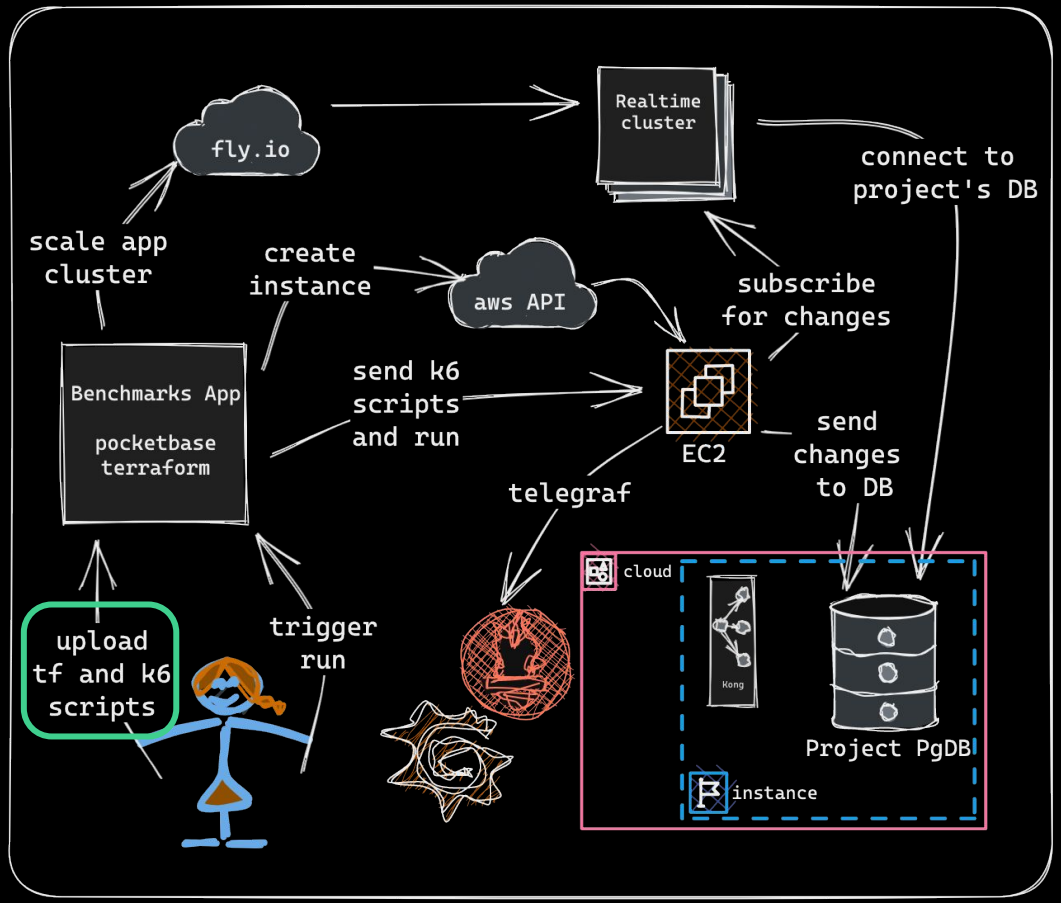
На картинке



supabase

Шаг 1

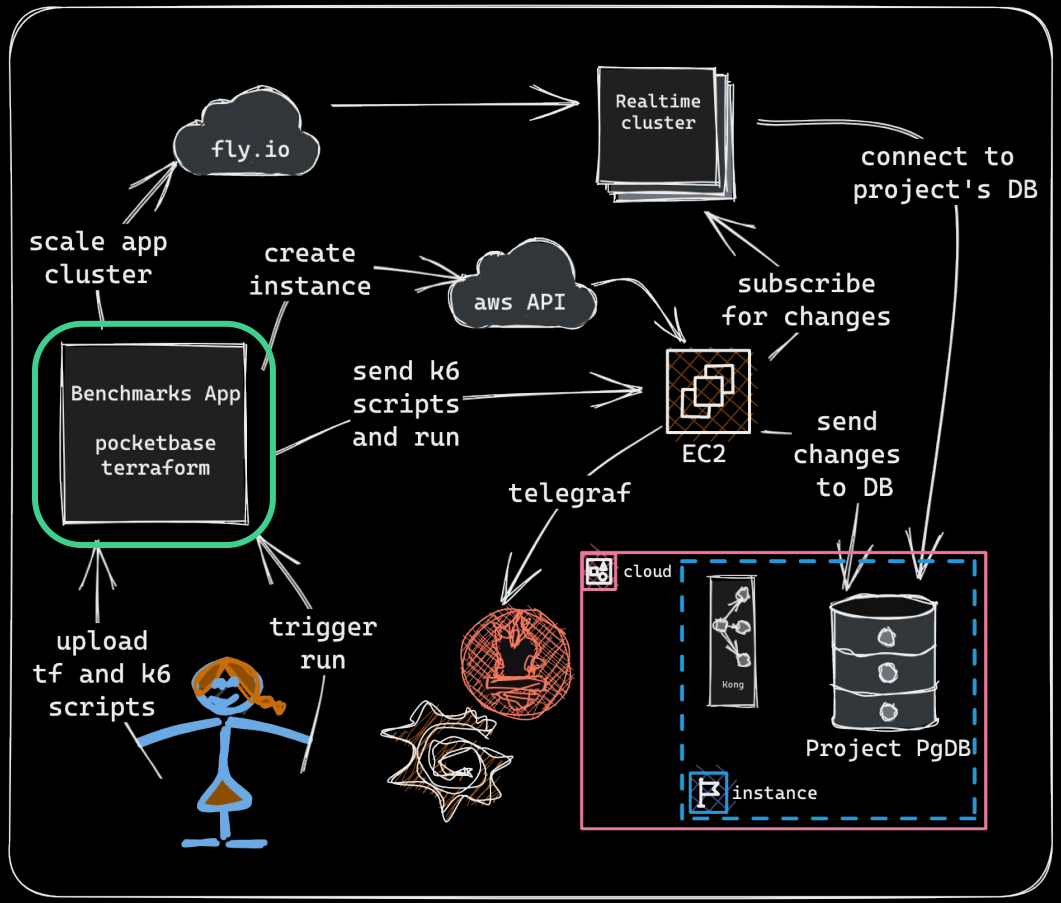
Пишем скрипты нагрузки и терраформы



supabase

Шаг 1

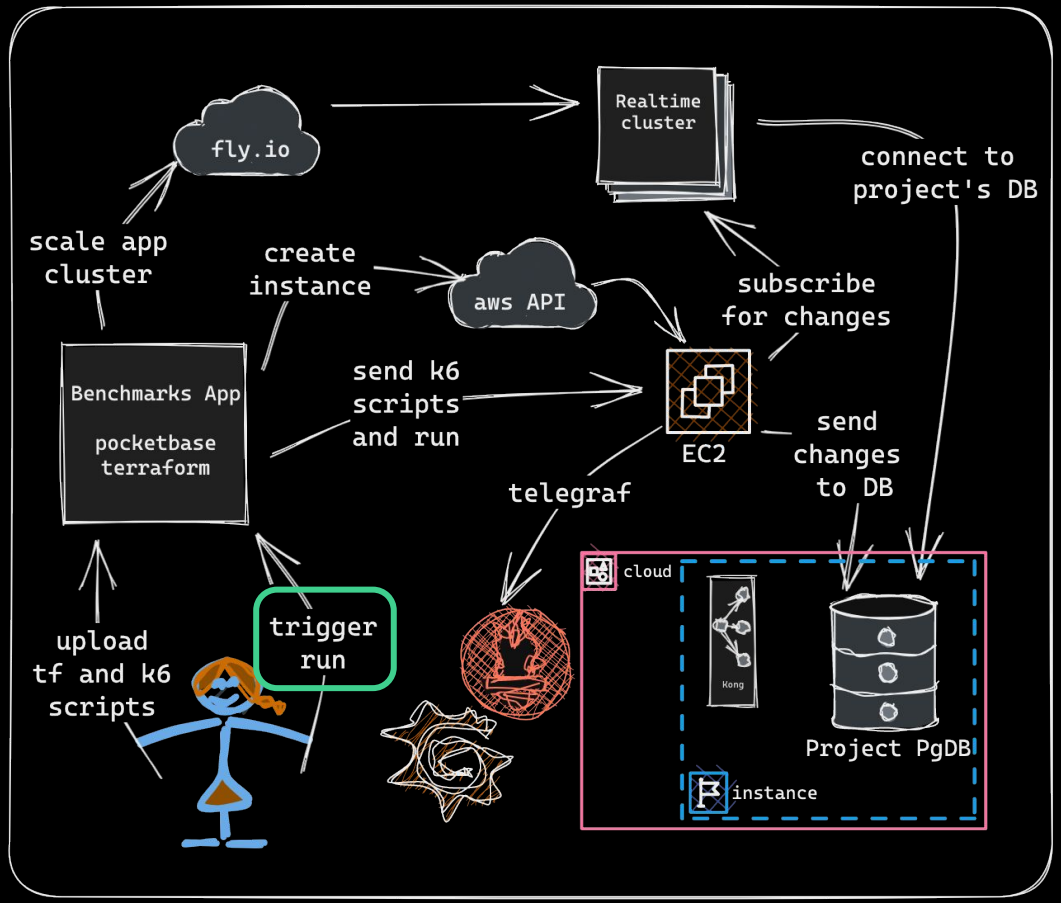
Пишем скрипты нагрузки и терраформы



supabase

Шаг 2

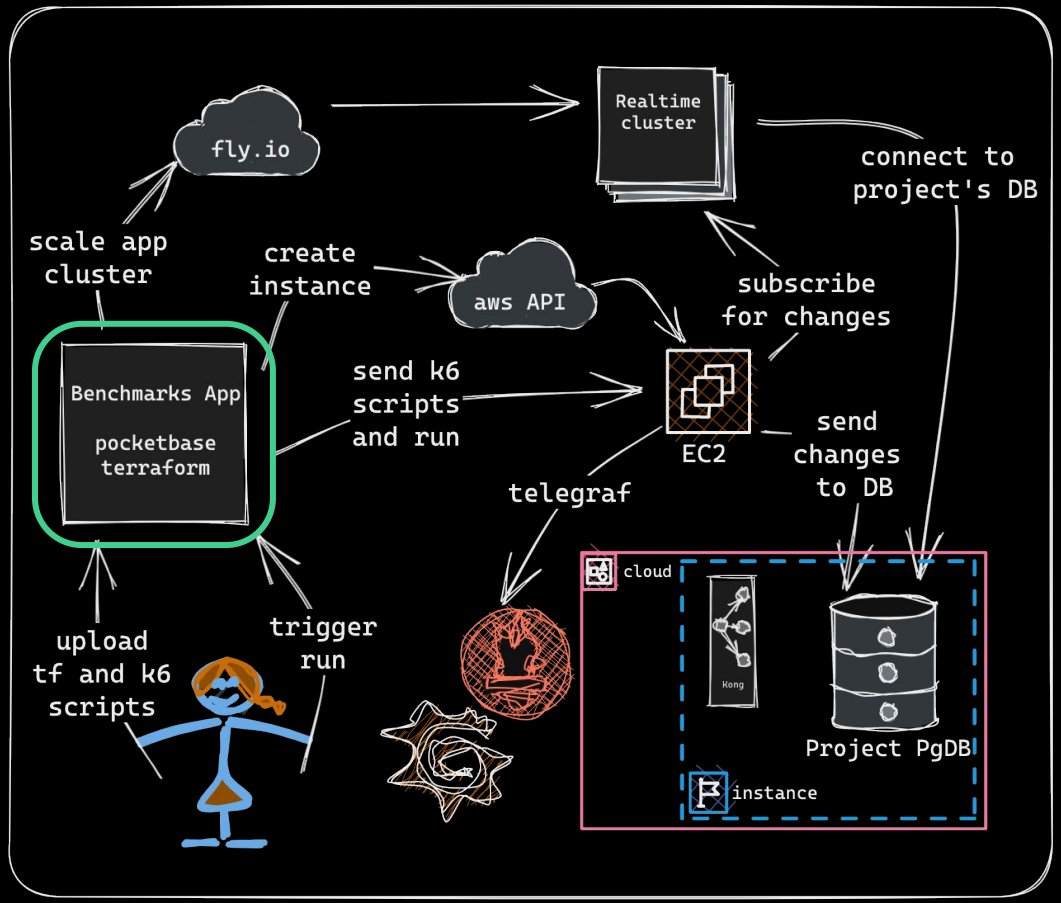
Запускаем тест



supabase

Шаг 2

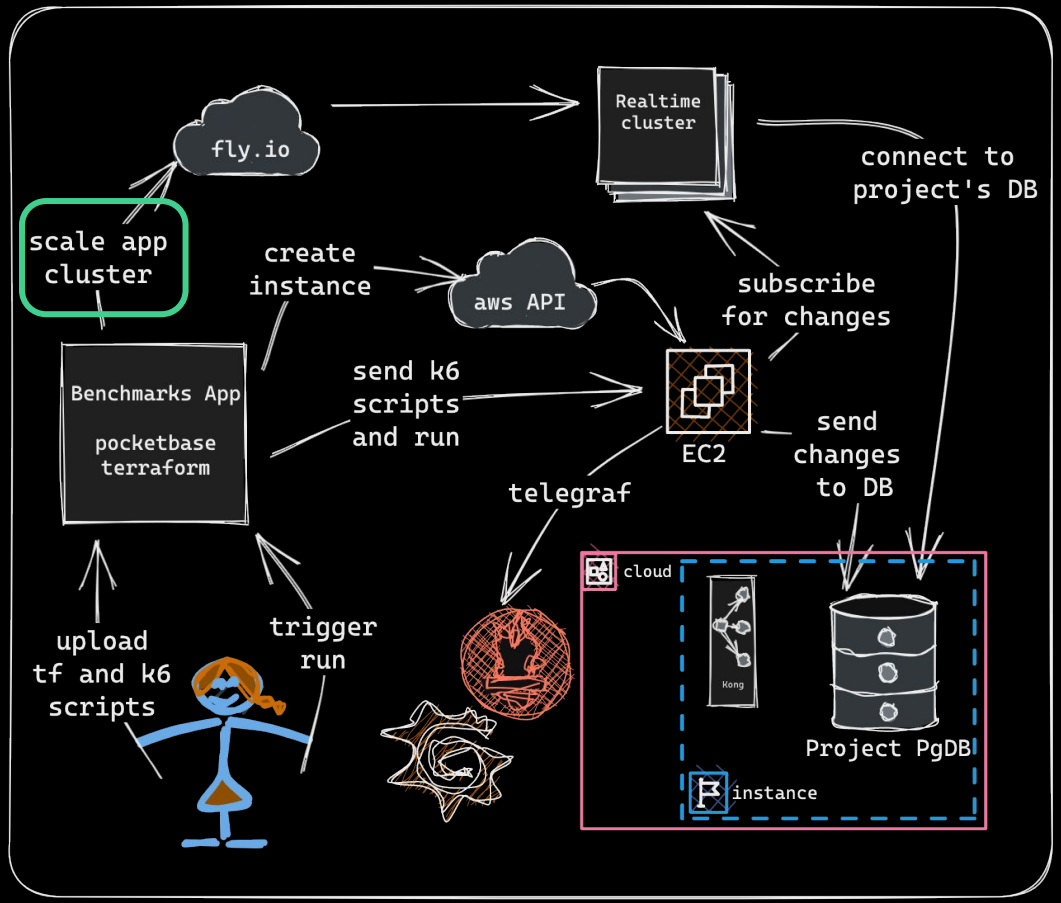
Запускаем тест



supabase

Шаг 3

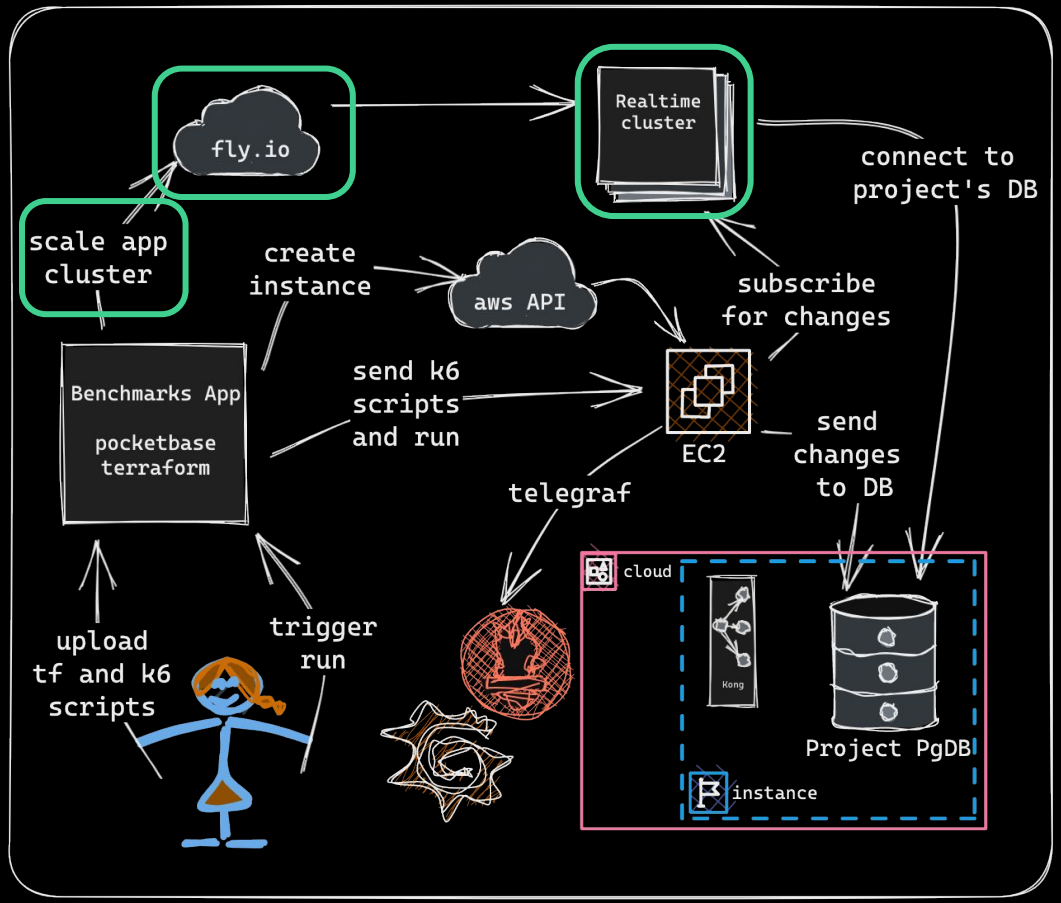
Развертывание инфраструктуры



supabase

Шаг 3

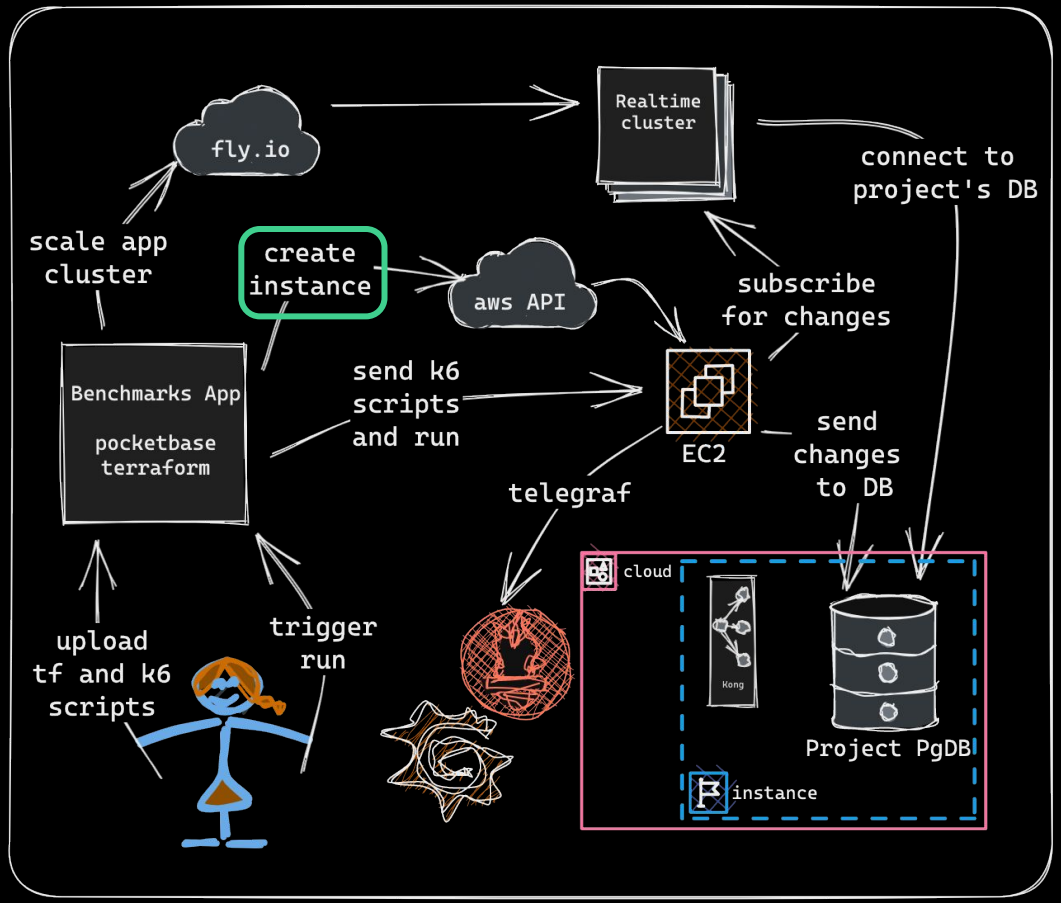
Развертывание инфраструктуры



supabase

Шаг 3

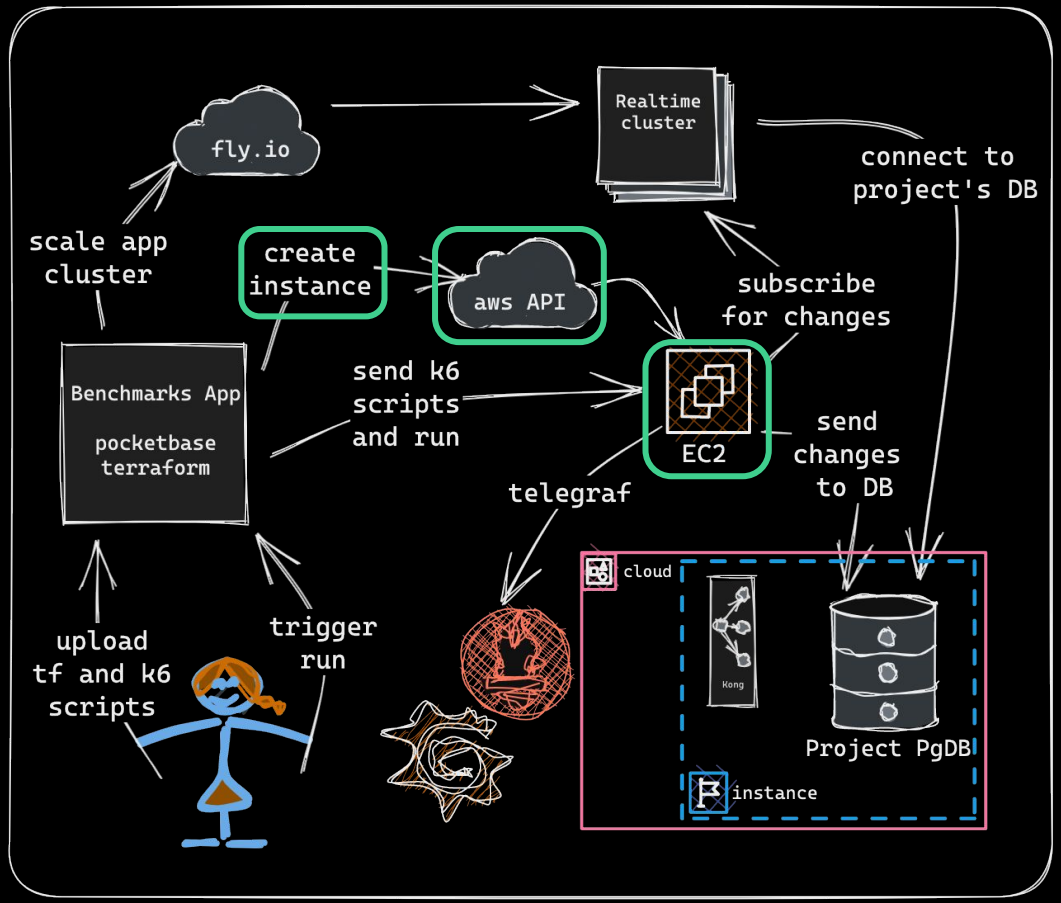
Развертывание инфраструктуры



supabase

Шаг 3

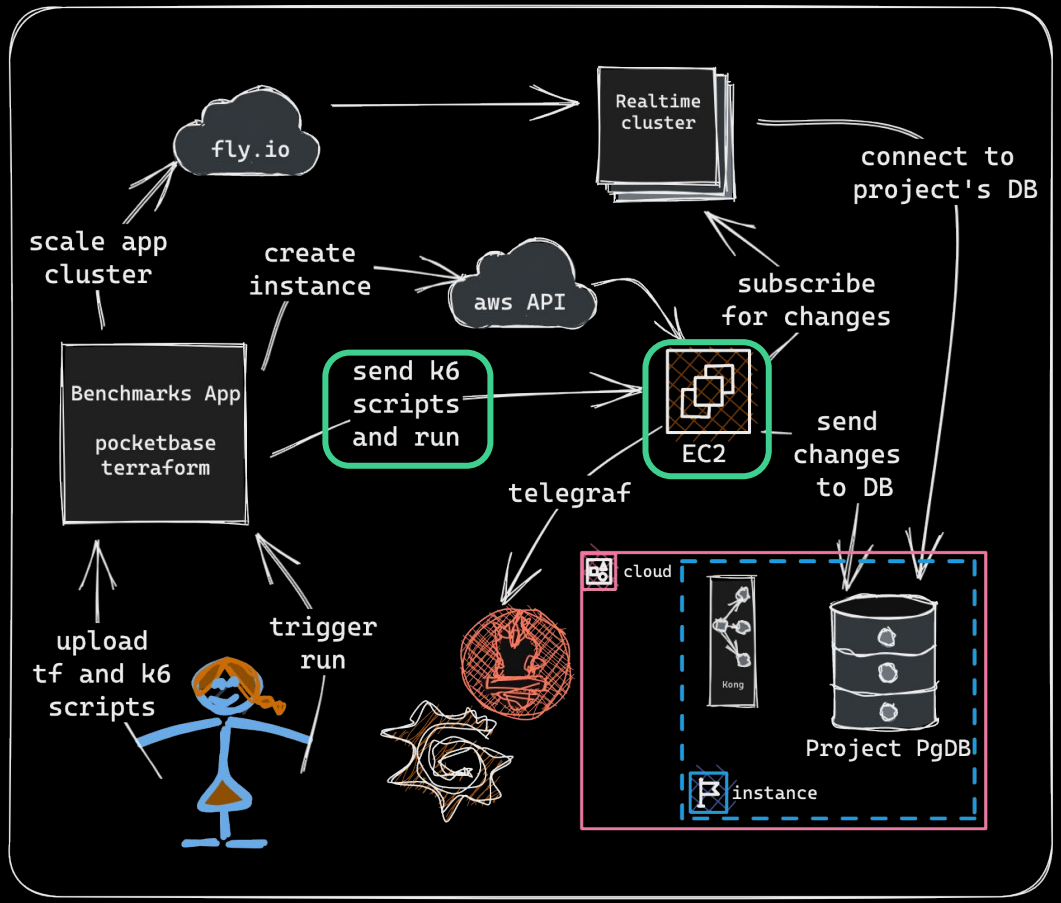
Развертывание инфраструктуры



supabase

Шаг 3

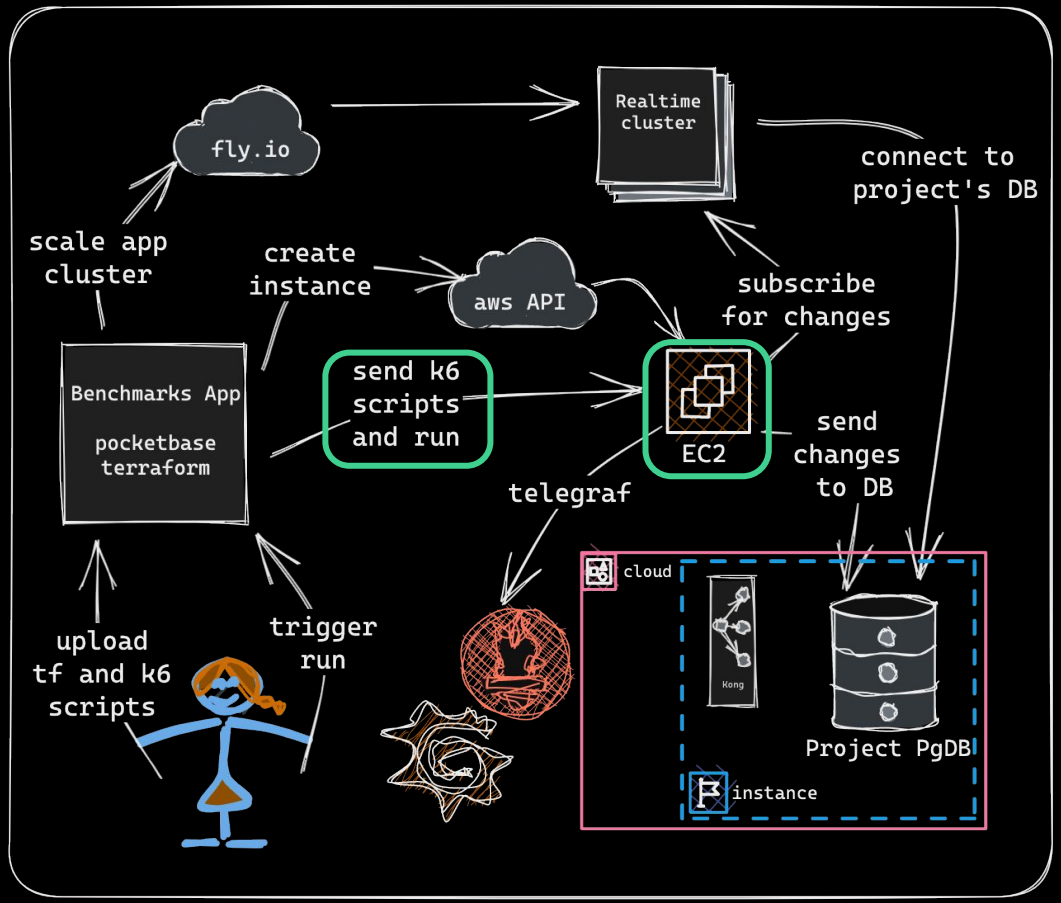
Развертывание инфраструктуры



supabase

Шаг 4

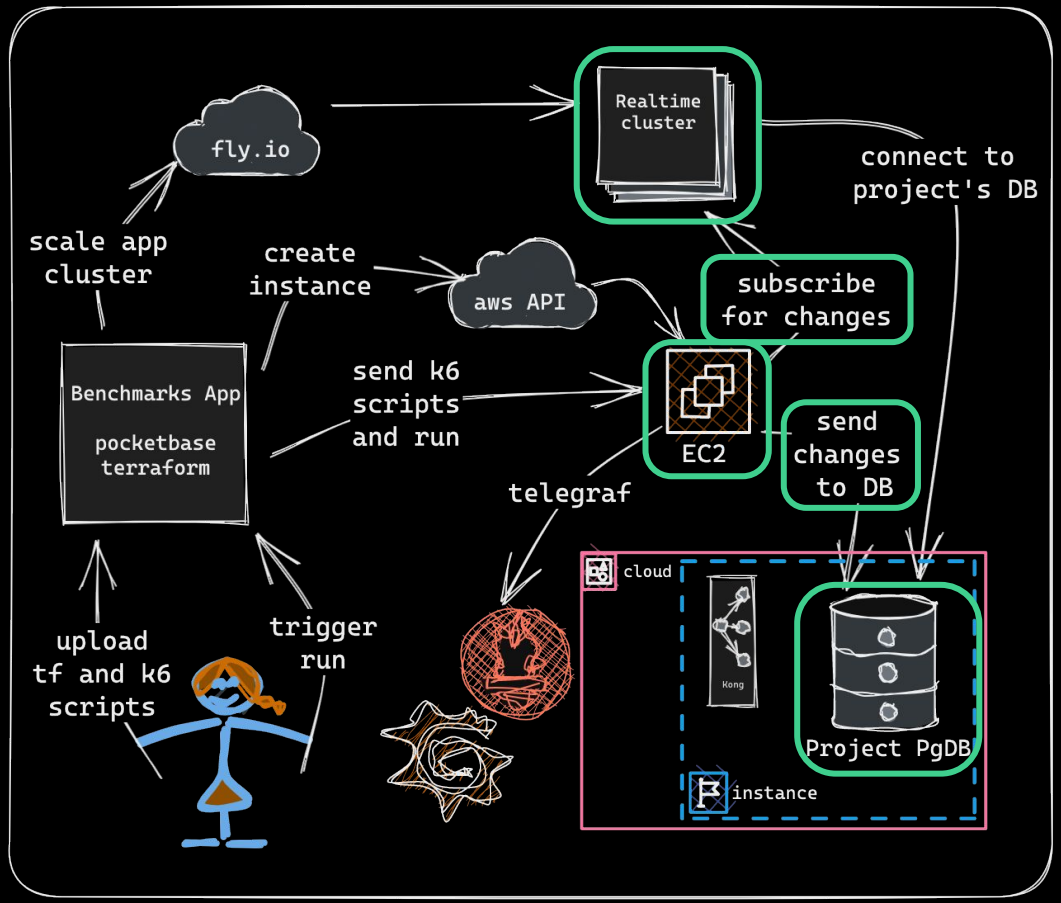
Запускается тест



supabase

Шаг 4

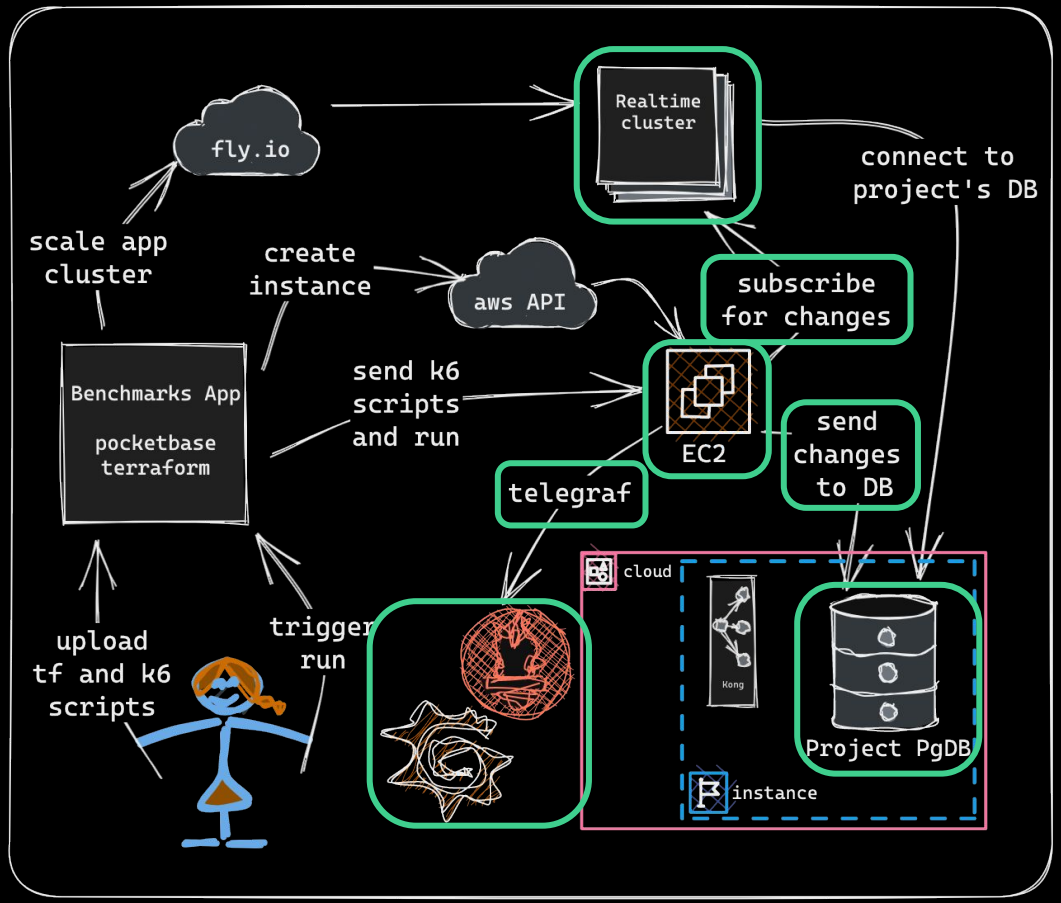
Запускается тест



supabase

Шаг 4

Запускается тест



supabase

Проблемы решены?

→ Проверять на удовлетворение стандарту потом

→ Можно не остановить виртуалку или не остановить SUT

→ Сложно делить ресурсы с разработчиками

→ Неудобно обновлять код на нескольких нагрузчиках

→ История есть, но неудобная

supabase

Интерфейс приложения

Search benchmarks..

storage

image_proxy

realtime-playground

200k

rls

replication-oldapi

replication-test

realtime

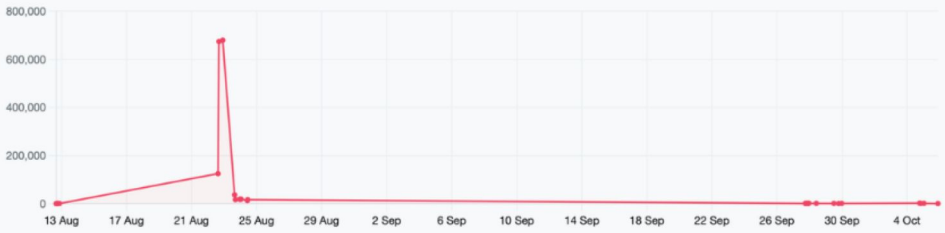
replication-with-...

replication

Projects / realtime / replication

New run

Search filter, ex. created > "2022-01-01"...



33 runs

id	name	origin	grafana_url	triggered
kqy2dokg2m5cfgb	PR	main	https://romanov.grafana.net/d/5D3riZ97k/...	2023-01-20 01:50:46 UTC
7nlt0j9je097f1j	PR_451	main	https://romanov.grafana.net/d/5D3riZ97k/...	2023-01-19 04:24:57 UTC
xx7cbibn3bk3tr6	PR_450	main	https://romanov.grafana.net/d/5D3riZ97k/...	2023-01-19 03:23:21 UTC
r44tx5dzmytbb2s	PR_447	main	https://romanov.grafana.net/d/5D3riZ97k/...	2023-01-19 03:19:14 UTC

Made by Supabase Powered by PocketBase Using Terraform and Grafana K6

supabase

Ресурсы создаются и убиваются

→ Проверять на удовлетворение стандарту потом

→ Можно не остановить виртуалку или не остановить SUT ✅

→ Сложно делить ресурсы с разработчиками ✅

→ Неудобно обновлять код на нескольких нагрузчиках

→ История есть, но неудобная ✅

supabase

Админка Pocketbase

Search collections...

Collections / secrets

projects

benchmarks

secrets

runs

+ New collection

Search filter, ex. created > "2022-

id	owner_id
lakxs3sm0bktsit	7chZgtupDwoRAq
hn8sv19y5f4wuop	7chZgtupDwoRAq
ua11u5xt4ujqsht	7chZgtupDwoRAq
tFLwKqH6HKLzVeN	7chZgtupDwoRAq
1UklZTDOV4ckmdA	7chZgtupDwoRAq
pKTUdNebc0QzCHC	7chZgtupDwoRAq
mwUuDIe2r5C7TOy	7chZgtupDwoRAq

Edit secrets record

BENCHMARK_ID *

ganaikvk4k2vp8x

name: image_proxy

SCRIPT

ip_terraform_4QL9Z2WzYD.zip

{} SCRIPT_LINK

{} ENV

{} VARS

```
"requests": "5",
"batchsize": "6",
"rampingduration": "300",
"consecutiveduration": "210",
```

Cancel

Save changes

supabase

Код всегда актуальный

→ Проверять на удовлетворение стандарту потом

→ Можно не остановить виртуалку или не остановить SUT ✅

→ Сложно делить ресурсы с разработчиками ✅

→ Неудобно обновлять код на нескольких нагрузчиках ✅

→ История есть, но неудобная ✅

supabase

Интеграция с CI

→ Запустить performance test из CI

```
curl --request POST \
  --url {supabench.url}/api/runs \
  --header 'Authorization: Admin ${JWT_TOKEN}' \
  --data '{"benchmark_id":"${BENCHMARK_ID}","name":"my-testrun","origin":"master"}'
```

→ Хук, чтобы получить результаты после теста:

Work In Progress

Можем запускать как часть CI пайплайна

→ Проверять на удовлетворение стандарту потом ✅

→ Можно не остановить виртуалку или не остановить SUT ✅

→ Сложно делить ресурсы с разработчиками ✅

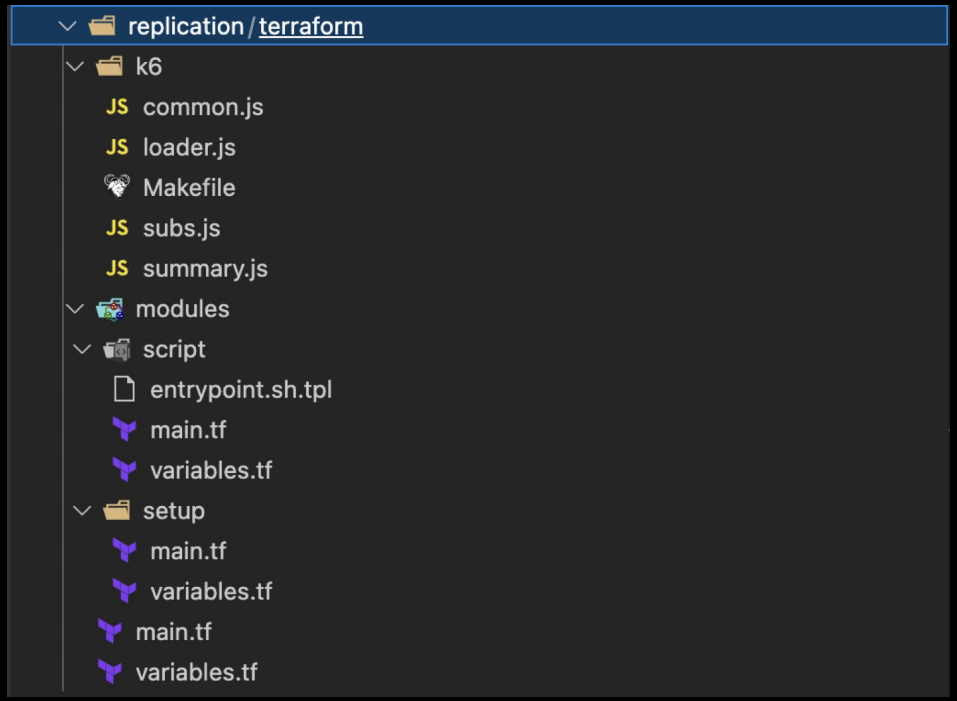
→ Неудобно обновлять код на нескольких нагрузчиках ✅

→ История есть, но неудобная ✅

supabase

Пример готового проекта

- → Пачка k6 скриптов и вспомогательных функций
- → Main терраформ модуль
- → Терраформ модуль для развертывания тестируемого приложения
- → Терраформ модуль для развертывания нагрузчиков и запуска теста



supabase

Main module

Run all submodules

```
{
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "4.0.0"
    }
  }
}

provider "aws" {
  region = "eu-central-1"
}

module "setup_infra" {
  source = "./modules/setup"

  app_name = var.app_name
  fly_access_token = var.fly_access_token
}

module "script" {
  source = "./modules/script"

  pg_pass  = var.pg_pass
  pg_host  = var.pg_host

  depends_on = [
    module.setup_infra.ready,
  ]
}
```

Setup SUT

Scale SUT cluster using CLI

```
resource "null_resource" "fly" {
  triggers = {
    app_name = var.app_name
    fly_access_token = var.fly_access_token
  }

  provisioner "local-exec" {
    command = "/flyctl scale -a ${var.app_name} count 1"
    environment = {
      HOME = path.module
      FLY_ACCESS_TOKEN = var.fly_access_token
    }
  }

  provisioner "local-exec" {
    when    = destroy
    command = "/flyctl scale -a ${self.triggers.app_name} count 0"
    environment = {
      HOME = path.module
      FLY_ACCESS_TOKEN = self.triggers.fly_access_token
    }
  }
}
```

supabase

Script

Create instance, ssh into, send k6 scripts

```
resource "aws_instance" "k6" {
  ami                    = var.ami_id
  instance_type          = var.instance_type
  vpc_security_group_ids = [var.security_group_id]
  subnet_id              = var.subnet_id

  key_name = var.key_name

  tags = {
    terraform   = "true"
    environment = "qa"
    app         = var.sut_name
    creator     = "supabench"
  }
}

resource "null_resource" "remote" {
  connection {
    type        = "ssh"
    user        = var.instance_user
    host        = aws_instance.k6.public_ip
    private_key = var.private_key_location
    timeout     = "1m"
  }

  provisioner "file" {
    source      = "${path.root}/k6"
    destination = "/tmp"
  }
```

Build entrypoint.sh, and execute it

```
  provisioner "file" {
    destination = "/tmp/k6/entrypoint.sh"

    content = templatefile(
      "${path.module}/entrypoint.sh.tpl",
      {
        pg_pass         = var.pg_pass
        pg_host         = var.pg_host
        conns           = var.conns
        duration        = var.duration
        testrun_name    = var.testrun_name
      }
    )
  }

  provisioner "remote-exec" {
    inline = [
      "#!/bin/bash",
      "source ~/.bashrc",
      "sudo chown -R ubuntu:ubuntu /tmp/k6",
      "sudo chmod +x /tmp/k6/entrypoint.sh",
      "/tmp/k6/entrypoint.sh",
    ]
  }

  depends_on = [
    aws_instance.k6,
  ]
}
```

supabase

entrypoint.sh.tpl

Можно сделать дополнительные шаги перед запуском теста

```
#!/bin/bash

wget https://golang.org/dl/go1.19.linux-amd64.tar.gz
sudo rm -rf /usr/local/go && sudo tar -C /usr/local -xzf go1.19.linux-amd64.tar.gz
export PATH=$PATH:/usr/local/go/bin

export K6_VERSION='v0.37.0'

~/go/bin/xk6 build --output /tmp/k6/k6 \
  --with github.com/jdheyburn/xk6-prometheus@v0.1.6 \
  --with github.com/grafana/xk6-sql@659485a

telegraf --config telegraf.conf &>/dev/null &

cd /tmp/k6 || exit 1

export PG_PASS="${pg_pass}"
export PG_HOST="${pg_host}"

make db_test \
  rate="${rate}" conns="${conns}" duration="${duration}" rooms="${rooms}" \
  testrun="${testrun_name}"
```

supabase

k6 script example

```
import ws from 'k6/ws'
import { Trend, Counter } from 'k6/metrics'

export { handleSummary } from './summary.js'

const baseDuration = __ENV.DURATION ? __ENV.DURATION : 60

const latencyTrend = new Trend('latency_trend')
const counterReceived = new Counter('received_updates')

export const options = {
  vus: 1,
  thresholds: to,
  summaryTrendStats: trends,
  scenarios: {
    replication: scenario(baseDuration, conns),
  },
}

export default () => {
  const res = ws.connect(URL, {}, (socket) => {
    socket.on('open', () => {
      // Join channel
      ...
    socket.on('message', (msg) => {
      // Add some metrics
      ...
  })

  check(res, { 'status is 101': (r) => r && r.status === 101 })
}
```

supabase

GitHub Workflow

Обновляем тестовый стенд и запускаем бенчмарк

```yaml
name: MT Supabench
on:
  push:
    branches:
      - main
env:
  FLY_API_TOKEN: ${{ secrets.FLY_API_TOKEN }}
jobs:
  deploy:
    name: Deploy QA application
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - uses: superfly/flyctl-actions@1.1
        with:
          args: "-c deploy/fly/qa.toml scale count 1"
      - uses: superfly/flyctl-actions@1.1
        with:
          args: "-c deploy/fly/qa.toml deploy --build-arg SLOT_NAME_SUFFIX=${GITHUB_SHA::7}"
      - uses: superfly/flyctl-actions@1.1
        if: always()
        with:
          args: "-c deploy/fly/qa.toml scale count 0"
      - name: Get PR into variable
        run: echo "PR_NUM=$(git show -s --format=%s | grep -Po '(?<=#)[0-9]*')" >> $GITHUB_ENV
      - name: Start Supabench
        run: |
          curl --request POST \
          --url https://supabench.fly.dev/api/runs \
          --header 'Authorization: User ${{ secrets.BENCHMARK_JWT }}' \
          --data '{"benchmark_id":"SyMKx","name":"PR ${{ env.PR_NUM }}","comment": "github.com/.../pull/${{ env.PR_NUM }}", "origin":"main"}'
```

supabase

Чего добились

→ Решаем задачи тестирования производительности ✅
→ Экономим ресурсы ✅
→ Разработчики могут пользоваться самостоятельно ✅
→ Всегда актуальный код ✅

supabase

vs. k6 cloud

	Наше приложение	k6 cloud
Решаем задачи тестирования производительности	✅	✅
Экономим ресурсы	✅	✅
Разработчики могут пользоваться самостоятельно	✅	✅
Всегда актуальный код	✅	✅

supabase

vs. k6 cloud

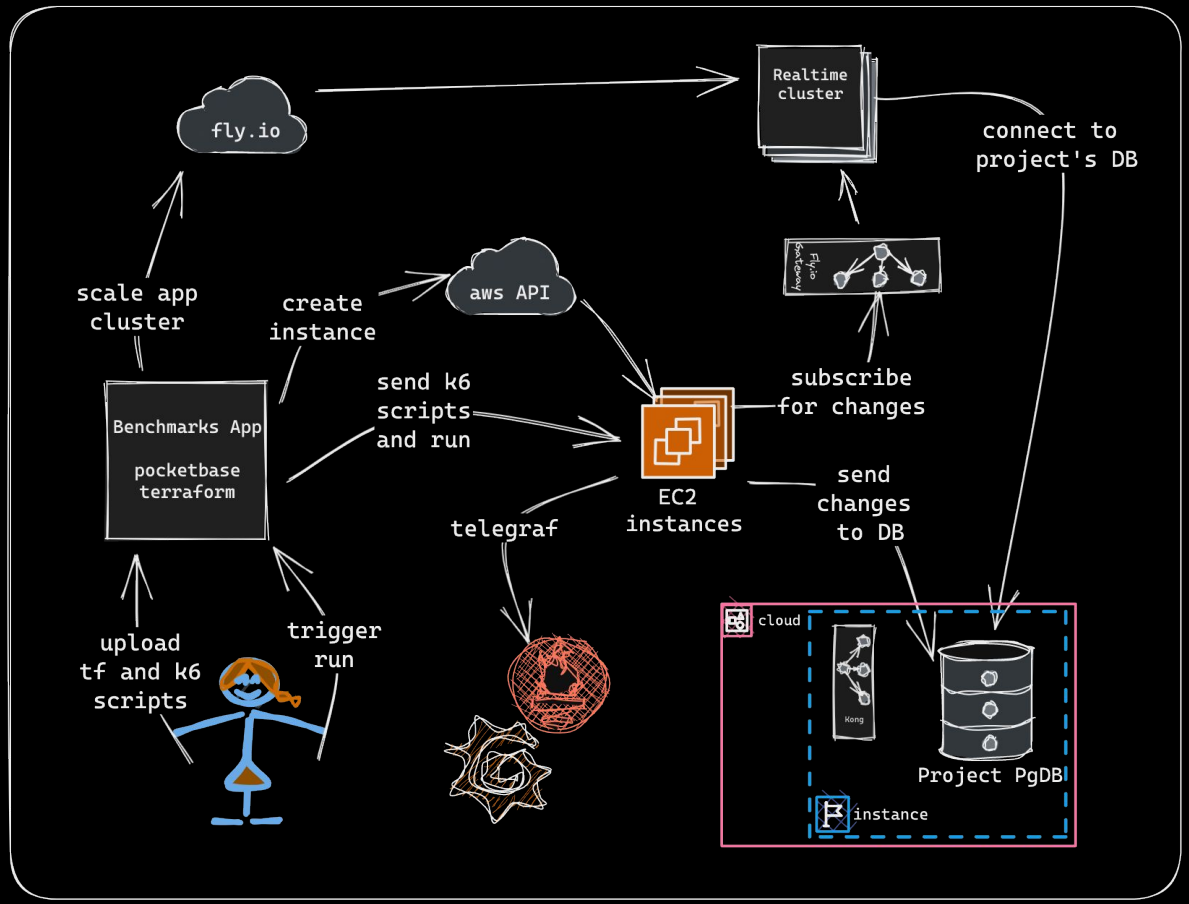
	Наше приложение	k6 cloud
Решаем задачи тестирования производительности	✅	✅
Экономим ресурсы	✅	✅
Разработчики могут пользоваться самостоятельно	✅	✅
Всегда актуальный код	✅	✅
Подходит для больших распределенных тестов	⁇	✅
Хороший UX, работа с графиками	±	✅
Интегрируется с CICD	✅	✅
Удобно сравнивать прогоны	±	✅
Инструменты для коллаборации	🚫	✅

Конец!

Привет, мы хотим 200 тысяч одновременных пользователей

Масштабируем вертикально

Нужно больше нагрузчиков
У вендоров могут быть ограничения



supabase

Script

В AWS добавляем всего одну строчку

```
resource "aws_instance" "k6" {
  ami                    = var.ami_id
  instance_type          = var.instance_type
  vpc_security_group_ids = [var.security_group_id]
  subnet_id              = var.subnet_id

  key_name = var.key_name

  tags = {
    terraform   = "true"
    environment = "qa"
    app         = var.sut_name
    creator     = "supabench"
  }
}

resource "null_resource" "remote" {
  connection {
    type        = "ssh"
    user        = var.instance_user
    host        = aws_instance.k6.public_ip
    private_key = var.private_key_location
    timeout     = "1m"
  }

  provisioner "file" {
    source      = "${path.root}/k6"
    destination = "/tmp"
  }
```

```
  provisioner "file" {
    destination = "/tmp/k6/entrypoint.sh"

    content = templatefile(
      "${path.module}/entrypoint.sh.tpl",
      {
        pg_pass         = var.pg_pass
        pg_host         = var.pg_host
        conns           = var.conns
        duration        = var.duration
        testrun_name    = var.testrun_name
      }
    )
  }

  provisioner "remote-exec" {
    inline = [
      "#!/bin/bash",
      "source ~/.bashrc",
      "sudo chown -R ubuntu:ubuntu /tmp/k6",
      "sudo chmod +x /tmp/k6/entrypoint.sh",
      "/tmp/k6/entrypoint.sh",
    ]
  }

  depends_on = [
    aws_instance.k6,
  ]
}
```

supabase

Script

В AWS добавляем всего одну строчку

```
resource "aws_instance" "k6" {
  ami                    = var.ami_id
  instance_type          = var.instance_type
  vpc_security_group_ids = [var.security_group_id]
  subnet_id              = var.subnet_id

  key_name = var.key_name

  tags = {
    terraform   = "true"
    environment = "qa"
    app         = var.sut_name
    creator     = "supabench"
  }
}

resource "null_resource" "remote" {
  connection {
    type        = "ssh"
    user        = var.instance_user
    host        = aws_instance.k6.public_ip
    private_key = var.private_key_location
    timeout     = "1m"
  }

  provisioner "file" {
    source      = "${path.root}/k6"
    destination = "/tmp"
  }
```

```
  provisioner "file" {
    destination = "/tmp/k6/entrypoint.sh"

    content = templatefile(
      "${path.module}/entrypoint.sh.tpl",
      {
        pg_pass         = var.pg_pass
        pg_host         = var.pg_host
        conns           = var.conns
        duration        = var.duration
        testrun_name    = var.testrun_name
      }
    )
  }

  provisioner "remote-exec" {
    inline = [
      "#!/bin/bash",
      "source ~/.bashrc",
      "sudo chown -R ubuntu:ubuntu /tmp/k6",
      "sudo chmod +x /tmp/k6/entrypoint.sh",
      "/tmp/k6/entrypoint.sh",
    ]
  }

  depends_on = [
    aws_instance.k6,
  ]
}
```

supabase

Script

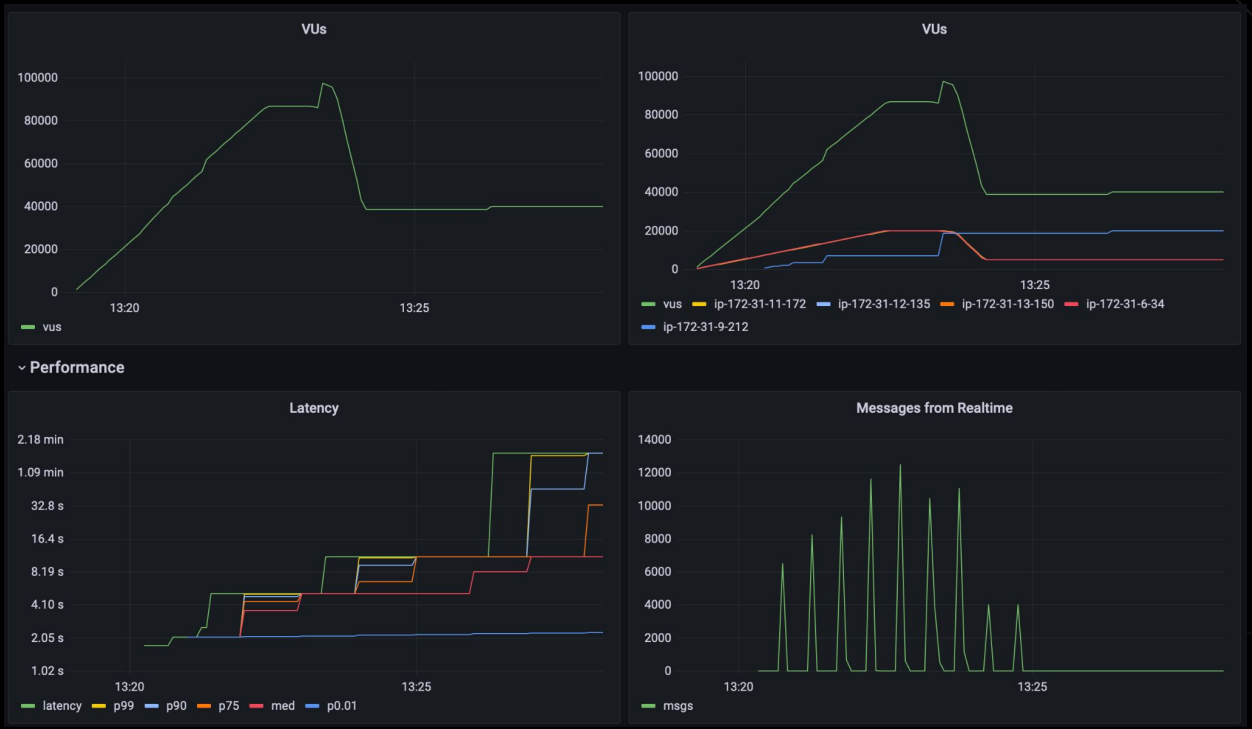
В AWS добавляем всего одну строчку

```
resource "aws_instance" "k6" {
  count = var.instances_count
  ami                    = var.ami_id
  instance_type          = var.instance_type
  vpc_security_group_ids = [var.security_group_id]
  subnet_id              = var.subnet_id

  key_name = var.key_name

  tags = {
    terraform   = "true"
    environment = "qa"
    app         = var.sut_name
    creator     = "supabench"
  }
}

resource "null_resource" "remote" {
  count = var.instances_count
  connection {
    type        = "ssh"
    user        = var.instance_user
    host        = aws_instance.k6.public_ip
    private_key = var.private_key_location
    timeout     = "1m"
  }

  provisioner "file" {
    source      = "${path.root}/k6"
    destination = "/tmp"
  }
```

```
  provisioner "file" {
    destination = "/tmp/k6/entrypoint.sh"

    content = templatefile(
      "${path.module}/entrypoint.sh.tpl",
      {
        pg_pass         = var.pg_pass
        pg_host         = var.pg_host
        conns           = var.conns
        duration        = var.duration
        testrun_name    = var.testrun_name
      }
    )
  }

  provisioner "remote-exec" {
    inline = [
      "#!/bin/bash",
      "source ~/.bashrc",
      "sudo chown -R ubuntu:ubuntu /tmp/k6",
      "sudo chmod +x /tmp/k6/entrypoint.sh",
      "/tmp/k6/entrypoint.sh",
    ]
  }

  depends_on = [
    aws_instance.k6,
  ]
}
```

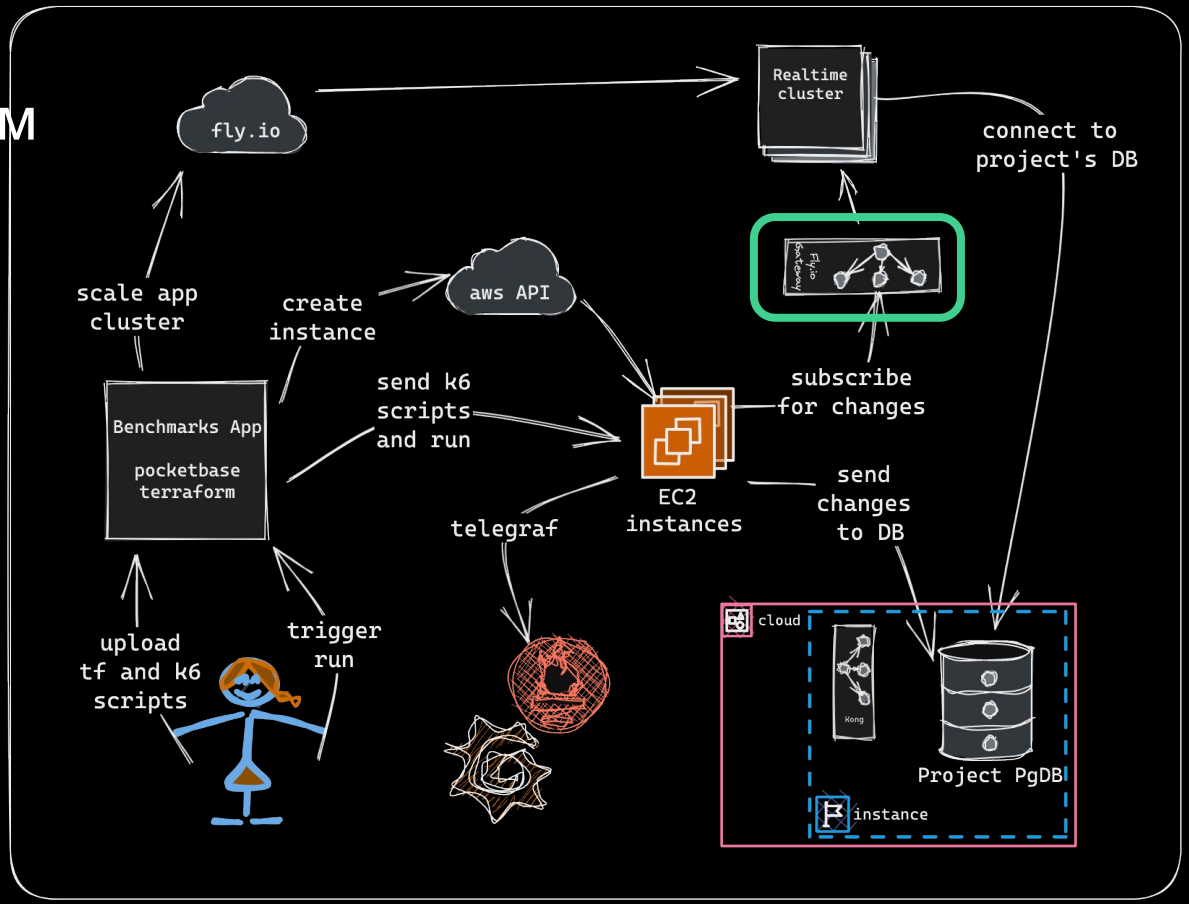
supabase

Проверяем реконнект



supabase

Размазываем нагрузку по регионам



supabase

Main module

AWS специфично, зависит от провайдера

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "4.0.0"
    }
  }
}

provider "aws" {
  region = "eu-central-1"
}

module "setup_infra" {
  source = "./modules/setup"

  app_name = var.app_name
  fly_access_token = var.fly_access_token
}
```

Добавляем по блоку на каждый регион

```
module "script" {
  source = "./modules/script"

  pg_pass         = var.pg_pass
  pg_host         = var.pg_host

  depends_on = [
    module.setup_infra.ready,
  ]
}
```

supabase

Main module

AWS специфично, зависит от провайдера

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "4.0.0"
    }
  }
}

provider "aws" {
  region = "eu-central-1"
}

module "setup_infra" {
  source = "./modules/setup"

  app_name = var.app_name
  fly_access_token = var.fly_access_token
}
```

Добавляем по блоку на каждый регион

```
module "script" {
  source = "./modules/script"

  pg_pass         = var.pg_pass
  pg_host         = var.pg_host

  depends_on = [
    module.setup_infra.ready,
  ]
}
```

supabase

Main module

AWS специфично, зависит от провайдера

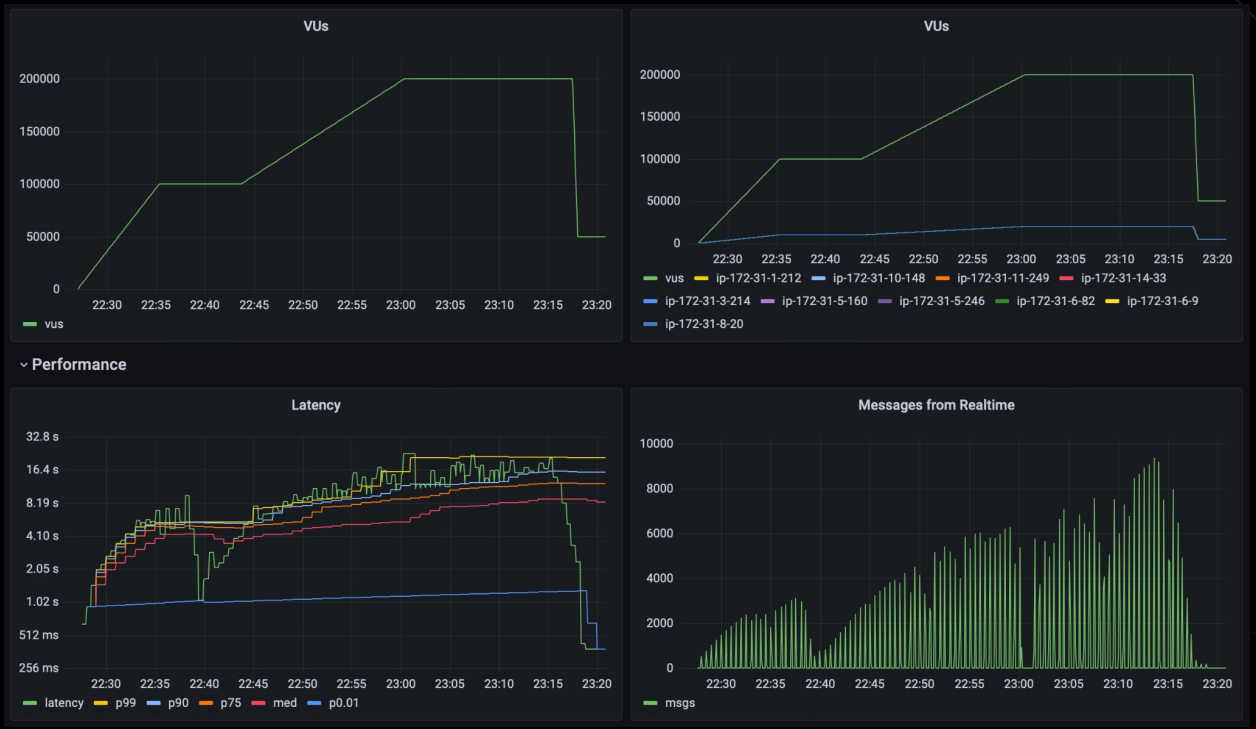
```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "4.0.0"
    }
  }
}


provider "aws" {
  alias = "eu-central-1"

  region = "eu-central-1"
}

provider "aws" {
  alias = "eu-west-2"

  region = "eu-west-2"
}

module "setup_infra" {
  source = "./modules/setup"

  app_name = var.app_name
  fly_access_token = var.fly_access_token
  app_nodes_count = var.app_nodes_count
}
```

Добавляем по блоку на каждый регион

```
module "eu-west-2" {
  source = "./modules/script"

  instances_count = var.instances_count / 2
  pg_pass         = var.pg_pass
  pg_host         = var.pg_host

  depends_on = [
    module.setup_infra.ready,
  ]

  providers = {
    aws = aws.eu-west-2
  }
}

module "eu-central-1" {
  source = "./modules/script"

  instances_count = var.instances_count / 2
  pg_pass         = var.pg_pass
  pg_host         = var.pg_host

  depends_on = [
    module.setup_infra.ready,
  ]

  providers = {
    aws = aws.eu-central-1
  }
}
```

supabase

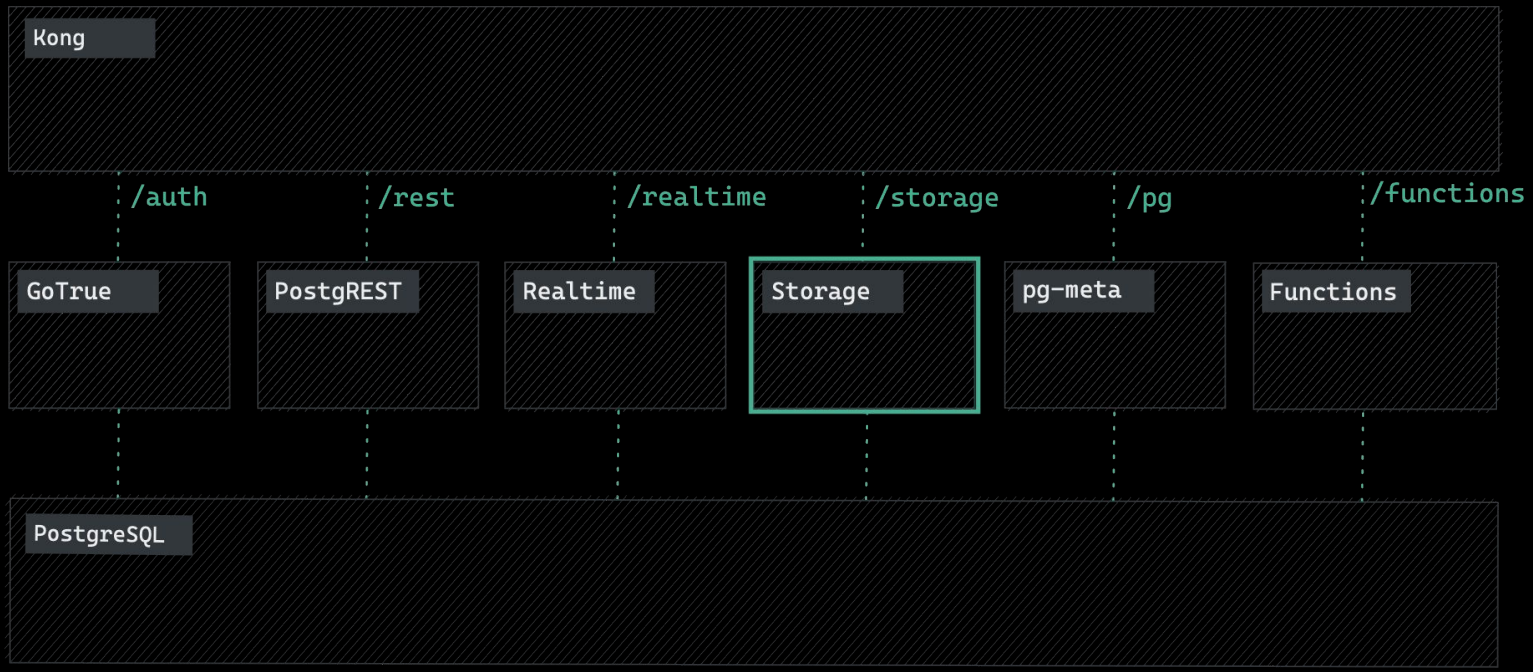
И все работает!



supabase

Привет, я видел, как тестируется Realtime, а можем мы сделать так же для Storage?

Storage API


Kong
/auth
/rest
/realtime
/storage
/pg
/functions
GoTrue
PostgREST
Realtime
Storage
pg-meta
Functions
PostgreSQL

New

Storage v2

Image Resizing and Smart CDN

Добавляем нагрузку для ImageProxy

- → Написать терраформ - 1 час
- → Написать k6 скрипты - 2 часа
- → Создать проект и тестовых юзеров - 30 минут
- → Эксперименты со сценарием нагрузки - 8 часов
- → Далее эксперименты над самим сервисом - 14 часов и 22 эксперимента
- → Эксперименты под большого клиента с крупными файлами - 4 часа и 14 экспериментов

supabase

Добавляем Storage API

→ Написать терраформ - 10 минут

→ Написать k6 скрипты - 15 минут

→ Эксперименты со сценарием нагрузки - 2 часа

→ Далее эксперименты над самим сервисом - 8 часов и 11 экспериментов

→ Результат:

- исправили проблемы со спайками CPU
- поменяли размер, при той же цене кластера увеличили его производительность в разы
- Починили автоскейлинг

Вернемся к k6 cloud

vs. k6 cloud

	Наше приложение	k6 cloud
Решаем задачи тестирования производительности	✅	✅
Экономим ресурсы	✅	✅
Разработчики могут пользоваться самостоятельно	✅	✅
Всегда актуальный код	✅	✅
Подходит для больших распределенных тестов	⁇	✅
Хороший UX, работа с графиками	±	✅
Интегрируется с CICD	✅	✅
Удобно сравнивать прогоны	±	✅
Инструменты для коллаборации	🚫	✅

vs. k6 cloud

	Наше приложение	k6 cloud
Решаем задачи тестирования производительности	✅	✅
Экономим ресурсы	✅	✅
Разработчики могут пользоваться самостоятельно	✅	✅
Всегда актуальный код	✅	✅
Подходит для больших распределенных тестов	✅	✅
Хороший UX, работа с графиками	±	✅
Интегрируется с CICD	✅	✅
Удобно сравнивать прогоны	±	✅
Инструменты для коллаборации	🚫	✅

vs. k6 cloud

	Наше приложение	k6 cloud
Подходит для больших распределенных тестов	✅	✅
Хороший UX, работа с графиками	±	✅
Интегрируется с CICD	✅	✅
Удобно сравнивать прогоны	±	✅
Инструменты для коллаборации	🚫	✅
В разы дольше история	✅	🚫
Плагины	✅	🛠️
Возможность управлять SUT	✅	🚫
Можно добавить на дашборды графики SUT	✅	🚫

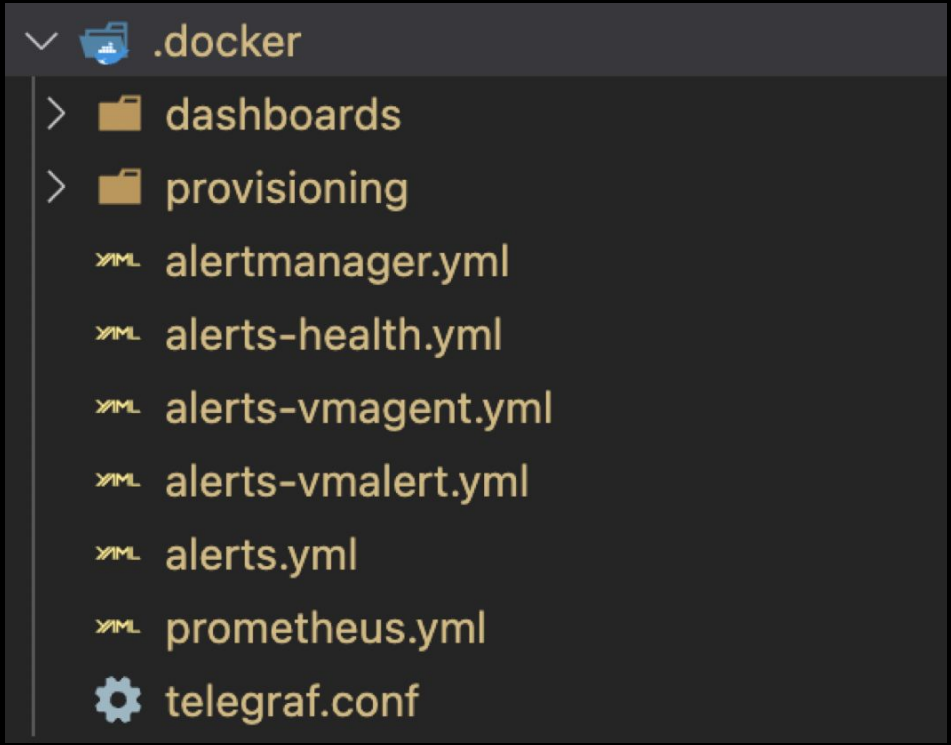
За полгода экономия

>150`000$

Demo

docker-compose

- → benchmarks app
- → victoriametrics
- → grafana
- → telegraf
- → vmagent
- → vmalert
- → alertmanager



supabase

Demo

video

supabase

benchmarks app

→ Решаем задачи тестирования производительности ✅
- Сравнить несколько решений ✅
- Определить скорость, производительность, стабильность ✅
- Понять как масштабируется ✅
- Найти возможные проблемы ✅
- Установить стандарты для приложения ✅
- Проверять на удовлетворение стандарту потом ✅

→ Экономим ресурсы ✅

→ Разработчики могут пользоваться самостоятельно ✅

→ Всегда актуальный код ✅

→ Подходит для больших распределенных тестов ✅

→ Возможность управлять SUT ✅

→ Большая история ✅

→ Дешево ✅

supabase

Ссылки:

- → Supabase Website
- → Benchmarks App GitHub Repo
- → Benchmarks App
- → Pocketbase
- → Terraform

supabase

Всем Спасибо :)