# Migrações de dados sem downtimes!

## Truques e lições aprendidas com PostgreSQL na Olist

Jéssica Bonson, desenvolvedora no **Olist**
**PythonBrasil 2019**

# Desenvolvedora na **Olist**


Graduação/Mestrado em Ciências da Computação

+-7 anos como pesquisadora / desenvolvedora

Foco em backend, big data e machine learning

Curto jogos, RPG e artes marciais

# Olist

Maior loja nos principais marketplaces do Brasil.

Arquitetura de micro serviços e serverless.

Python. Go. PostgreSQL. AWS. Heroku.

olist
empowering commerce

- As migrações de dados são parte do processo de deploy.
  - Alterações nas tabelas ou nos dados

- Como executar as migrações?
  - Downtime X Runtime

# Conceitos

# DELETEs e UPDATEs no Postgres

- DELETEs não deletam dados, matam
- UPDATEs não atualizam dados, duplicam

- Por quê?
  - Rollback de transações
  - Diferentes visibilidades de dados
  - Permitir escrita em paralelo com leitura
- Resultado: Cemitério de linhas mortas

# VACUUM

- É quem deleta ‘de verdade’

- Nunca causa exclusive locks nas tabelas
- Só marca dead rows quando não estão mais sendo usadas

- Na verdade, não deleta, só marca para reuso
  - Alternativa: VACUUM FULL

# Autovacuum

- PostgreSQL sabe se virar
  - Um daemon checa as tabelas de tempos em tempos
  - Se necessário, roda o VACUUM

- Porém… transações lentas são um problema
  - VACUUM não consegue deletar as linhas
  - Causa table bloat

# Table Bloat

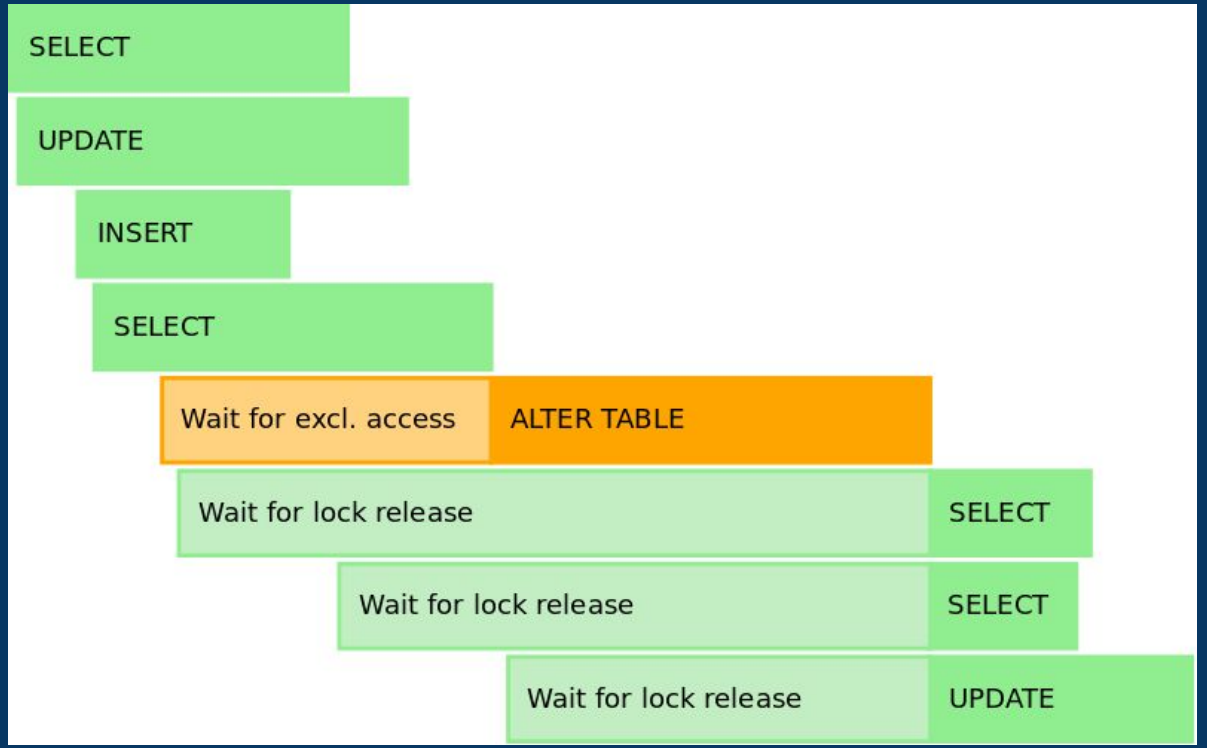
- Live rows X Dead rows
- Se o VACUUM continuar sem conseguir rodar…
  - A tabela fica cada vez maior
  - VACUUM leva mais tempo
  - Performance da API piora

- Se for um problema recorrente, tunar o autovacuum

heroku pg:vacuum_stats DATABASE_URL --app <app_name>

| table | last_vacuum | last_autovacuum | rowcount | dead_rowcount | autovacuum_threshold | expect_autovacuum |
|---|---|---|---|---|---|---|
| jobs | | 2013-05-20 16:54 | 82,617 | 36,056 | 16,573 | yes |
| logs | | 2013-05-20 16:27 | 1 | 18 | 50 | |
| reviews | | 2013-05-20 01:36 | 87 | 0 | 67 | |
| users | | 2013-05-20 16:28 | 0 | 23 | 50 | |
| ... | | | | | | |

# Locks


SELECT
UPDATE
INSERT
SELECT
Wait for excl. access
ALTER TABLE
Wait for lock release
SELECT
Wait for lock release
SELECT
Wait for lock release
UPDATE

# Migrações para atualização de tabelas

# Antes

```
UPDATE product
SET currency = 'BRL'
WHERE currency is Null;
```

# Como resolver?

- Fazer a atualização registro a registro
- Dividir a migração em lotes

# Depois

```
SELECT id
FROM product
WHERE currency is Null
LIMIT :limit;
```

```
UPDATE product
SET currency = 'BRL'
WHERE id = :id;
```

# Migrações para alteração de tabelas

# Antes

```
ALTER TABLE freight
ADD COLUMN enable_subsidy boolean
NOT NULL DEFAULT FALSE;
```

# Como resolver?

- Dividir a migração em partes:
  - Exemplo: Criar campo com default False
    i. Criar campo novo nullable
    ii. Setar default da coluna para False
    iii. Setar valores do campo para False
    iv. Remover nullable

# Depois

```
ALTER TABLE freight ADD COLUMN enable_subsidy boolean;

ALTER TABLE freight ALTER COLUMN enable_subsidy SET DEFAULT FALSE;

< Migração para alterar os valores do campo para FALSE >

ALTER TABLE freight ALTER COLUMN enable_subsidy SET NOT NULL;
```

# Réplicas

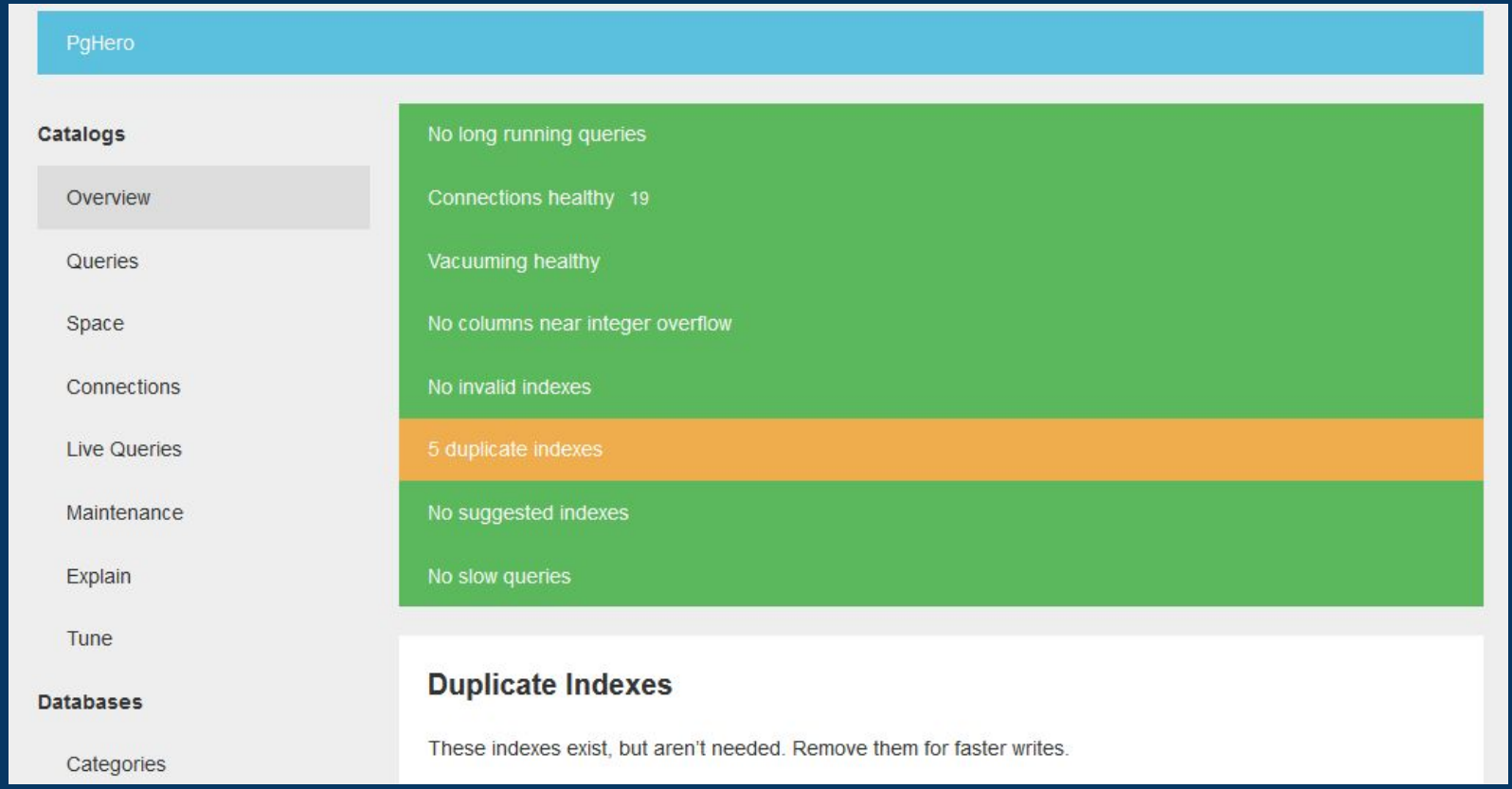
# Trade-offs

- Custo x Benefício
  - Melhor performance de leitura
  - Alta disponibilidade
  - Tolerância a falhas
  - Custo maior
  - Performance de escrita vs Consistência dos dados

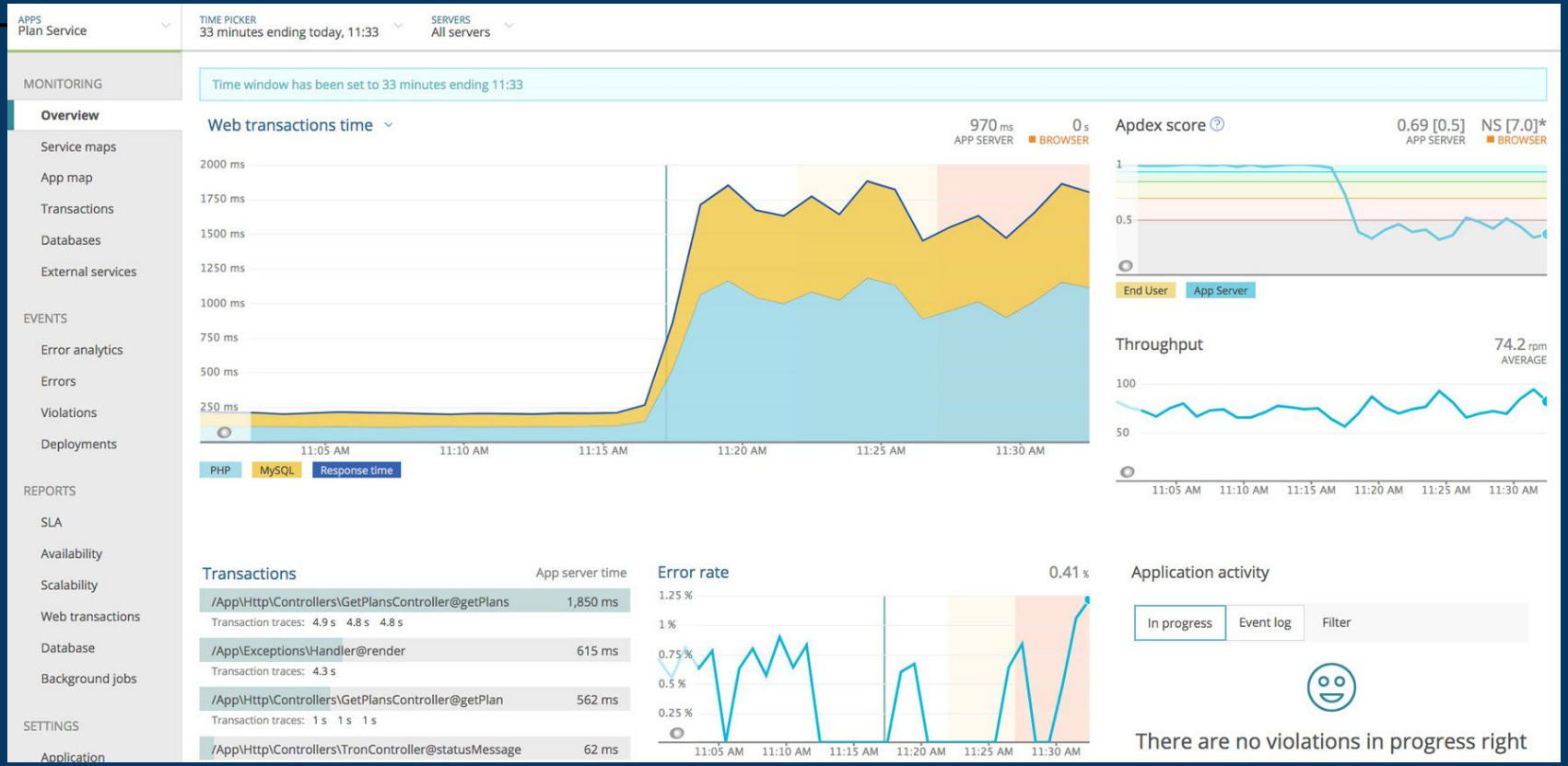
# Caso da API de Cálculo de Frete

- 1 Master Postgres na Amazon RDS com 2 Réplicas assíncronas
  - Read/Write na Master para sistemas internos
  - Read-only nas Réplicas
    - Sistemas externos
    - BI

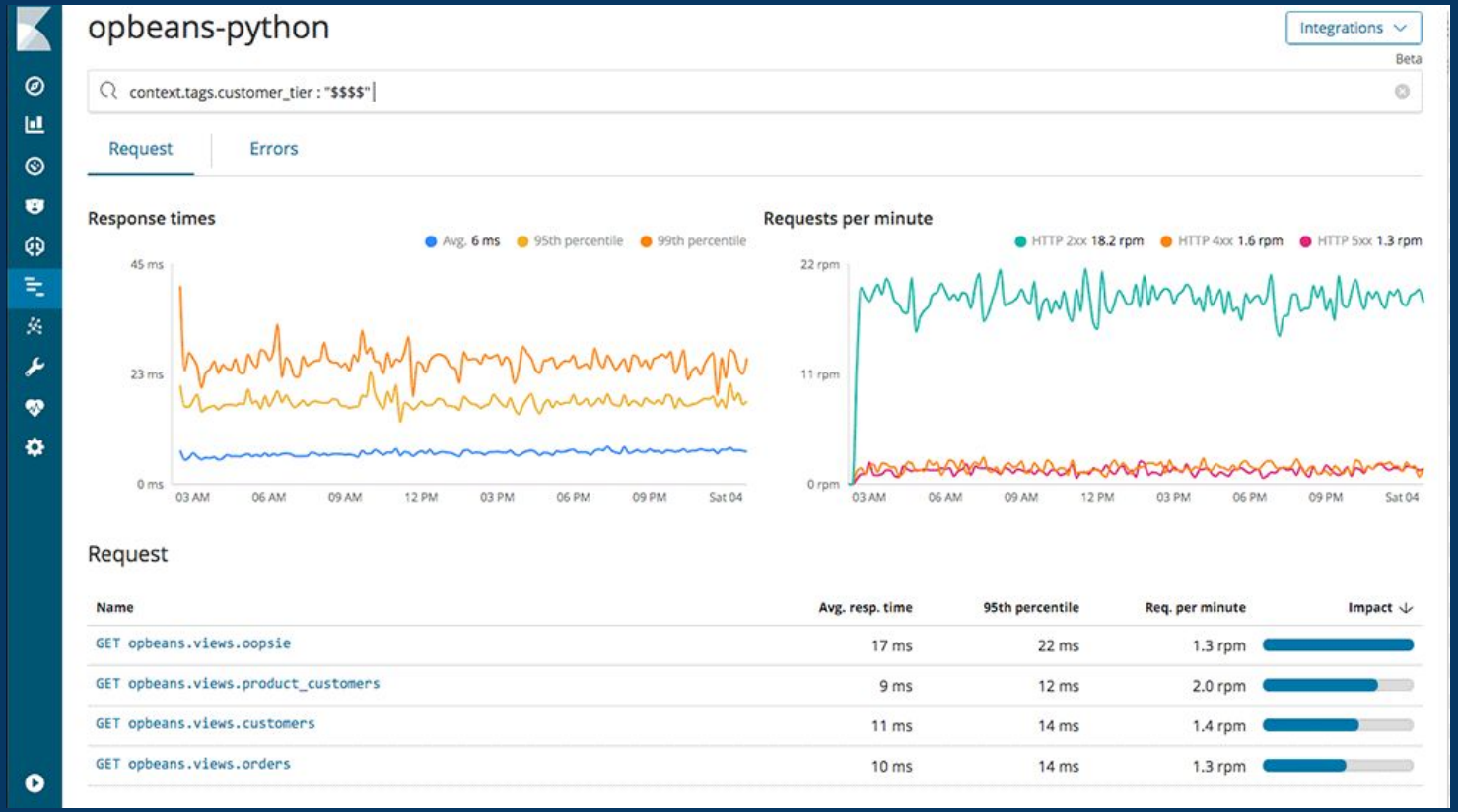
# Monitoramento


PgHero
Catalogs
Overview
Queries
Space
Connections
Live Queries
Maintenance
Explain
Tune
Databases
Categories
No long running queries
Connections healthy 19
Vacuuming healthy
No columns near integer overflow
No invalid indexes
5 duplicate indexes
No suggested indexes
No slow queries
Duplicate Indexes
These indexes exist, but aren't needed. Remove them for faster writes.

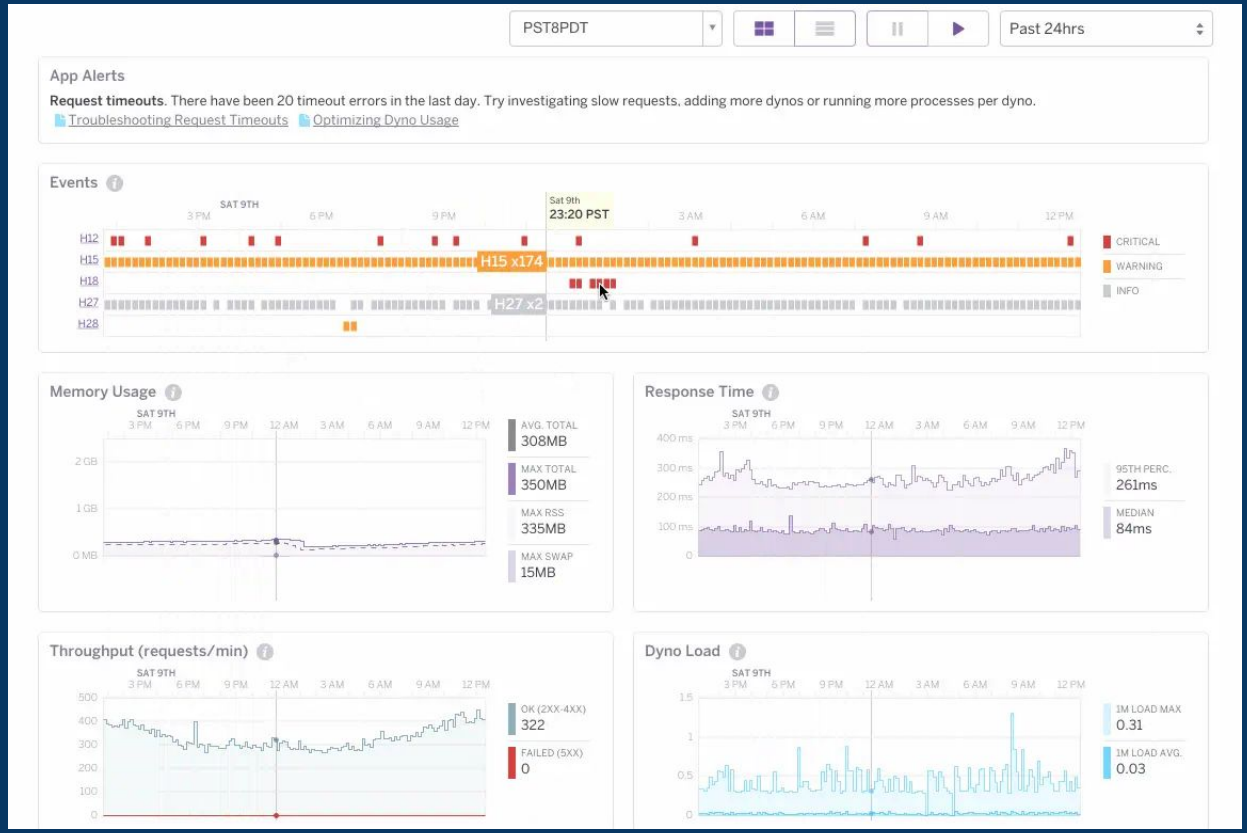
PgHero




NewRelic APM


opbeans-python
Integrations
Beta
context.tags.customer_tier : "$$$$"
Request
Errors
Response times
Avg. 6 ms
95th percentile
99th percentile
45 ms
23 ms
0 ms
03 AM
06 AM
09 AM
12 PM
03 PM
06 PM
09 PM
Sat 04
Requests per minute
HTTP 2xx 18.2 rpm
HTTP 4xx 1.6 rpm
HTTP 5xx 1.3 rpm
22 rpm
11 rpm
0 rpm
Request
Name
Avg. resp. time
95th percentile
Req. per minute
Impact
GET opbeans.views.oopsie
17 ms
22 ms
1.3 rpm
GET opbeans.views.product_customers
9 ms
12 ms
2.0 rpm
GET opbeans.views.customers
11 ms
14 ms
1.4 rpm
GET opbeans.views.orders
10 ms
14 ms
1.3 rpm

Elastic APM


PST8PDT
Past 24hrs
App Alerts
Request timeouts. There have been 20 timeout errors in the last day. Try investigating slow requests, adding more dynos or running more processes per dyno.
Troubleshooting Request Timeouts
Optimizing Dyno Usage
Events
SAT 9TH
Sat 9th
23:20 PST
H12
H15
H18
H27
H28
H15 x174
H27 x2
CRITICAL
WARNING
INFO
Memory Usage
AVG. TOTAL
308MB
MAX TOTAL
350MB
MAX RSS
335MB
MAX SWAP
15MB
Response Time
95TH PERC.
261ms
MEDIAN
84ms
Throughput (requests/min)
OK (2XX-4XX)
322
FAILED (5XX)
0
Dyno Load
1M LOAD MAX
0.31
1M LOAD AVG.
0.03

Heroku

# Dicas e Precauções

# Django

- Testar tempo da migração localmente
  - time python manage.py migrate

- Verificar o SQL que será executado
  - python manage.py sqlmigrate <app> <número da migração>

# Modelagem do BD

- UPDATE vs INSERT
- Estrutura das tabelas de histórico
- CREATE INDEX CONCURRENTLY vs CREATE INDEX

# Precauções

- Antes da migração...
  - Evitar executar em horário de pico
  - Lidar com queries longas em execução
  - Criar backup do banco de dados
  - Avisar os outros developers

WE'RE HIRING

# desenvol-vedor(a) python

olist

# Valeu!

## Perguntas?

@jpbonson

# Referências


- https://www.citusdata.com/blog/2018/02/15/when-postgresql-blocks/
- http://pankrat.github.io/2015/django-migrations-without-downtimes/
- https://momjian.us/main/writings/pgsql/nulls.pdf
- https://www.citusdata.com/blog/2018/02/22/seven-tips-for-dealing-with-postgres-locks/
- https://www.cybertec-postgresql.com/en/a-beginners-guide-to-postgresqls-update-and-autovacuum/
- https://devcenter.heroku.com/articles/managing-vacuum-on-heroku-postgres
- https://www.datadoghq.com/blog/postgresql-vacuum-monitoring/
- https://www.cybertec-postgresql.com/en/reasons-why-vacuum-wont-remove-dead-rows/
- https://www.percona.com/blog/2018/08/06/basic-understanding-bloat-vacuum-postgresql-mvcc/
- https://blog.timescale.com/scalable-postgresql-high-availability-read-scalability-streaming-replication-fb95023e2af/
- https://realpython.com/create-django-index-without-downtime/