Advogados do diabo

Como a arquitetura emergente da sua aplicação pode jogar contra a entrega contínua

Gleicon Moraes

https://github.com/gleicon
https://twitter.com/gleicon
http://opamp.io

Renato Lucindo

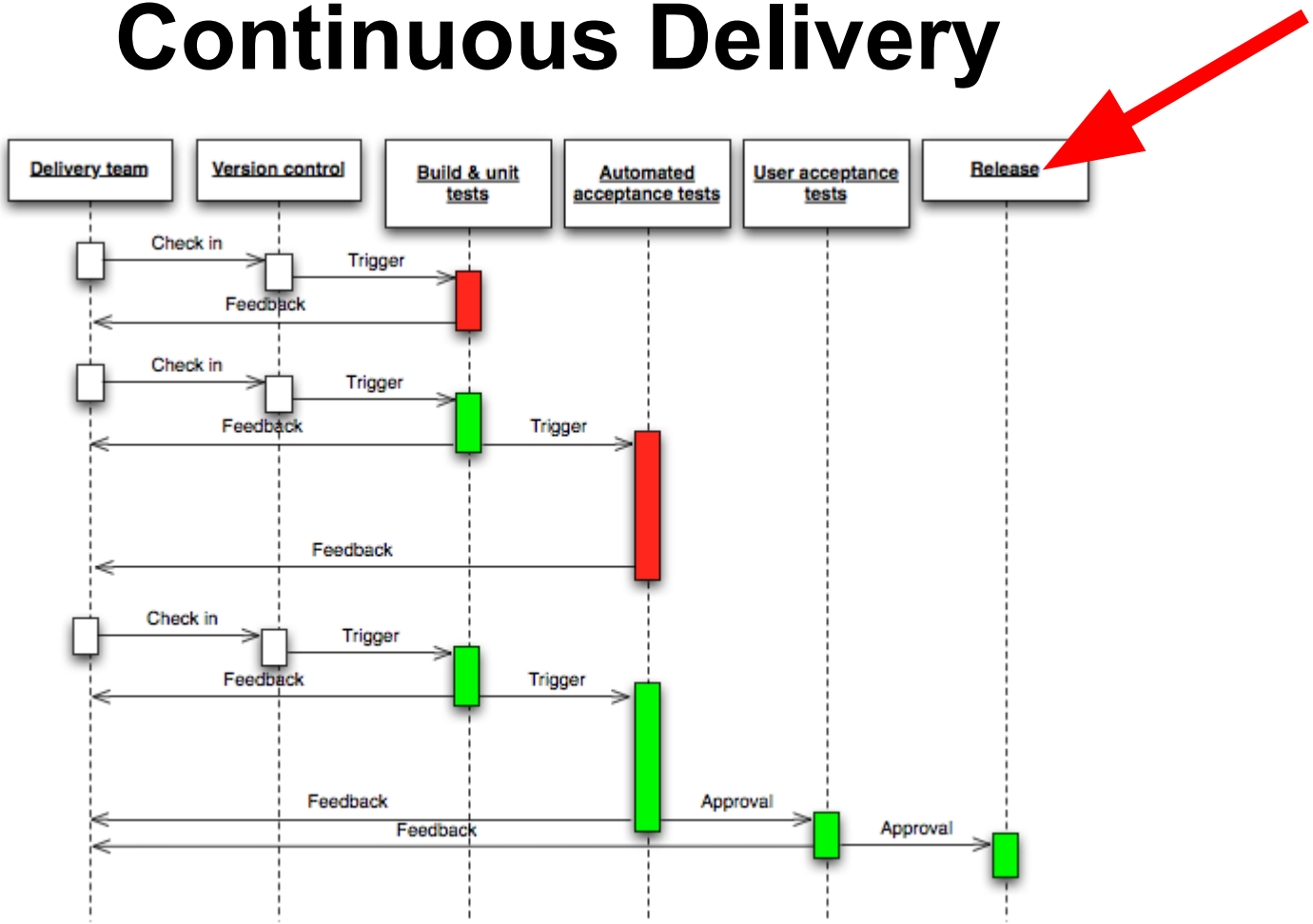
https://github.com/lucindo

https://twitter.com/rlucindo

http://opamp.io

https://intelie.com

Continuous Delivery



Continuous Delivery

Mudança incremental (check-in/push):

- Build ✓
- Testes unitários ✓
- Testes de integração ✓
- Testes de aceitação ✓
- Testes de interface ✓
- **Deploy** ✓

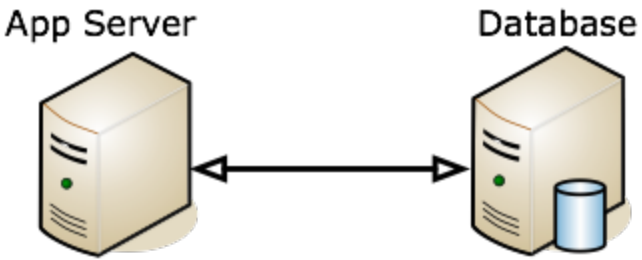
Tudo Automágico!

Desvios de percepção

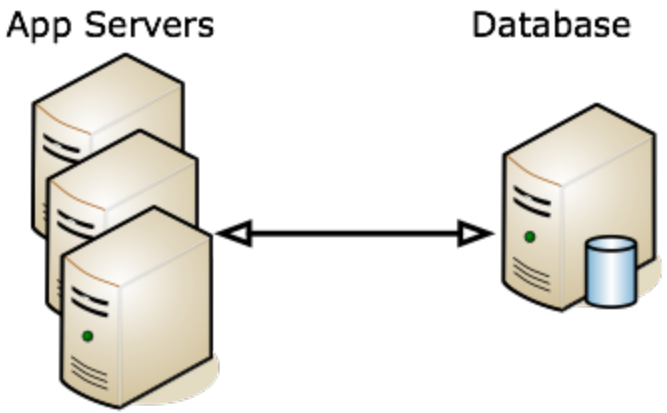
*"**Please, don’t drive a school bus blindfolded**." - Nassim Nicholas Taleb*

*"(...) there are known knowns; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns -- **the ones we don't know we don't know.** (...) **it is the latter category that tend to be the difficult ones**." - Donald Rumsfeld*

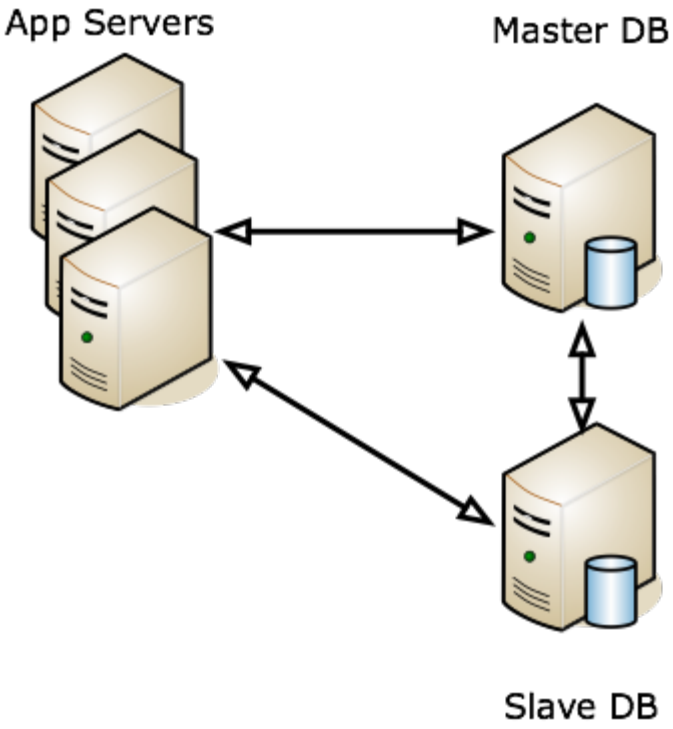
Sua aplicação começa assim



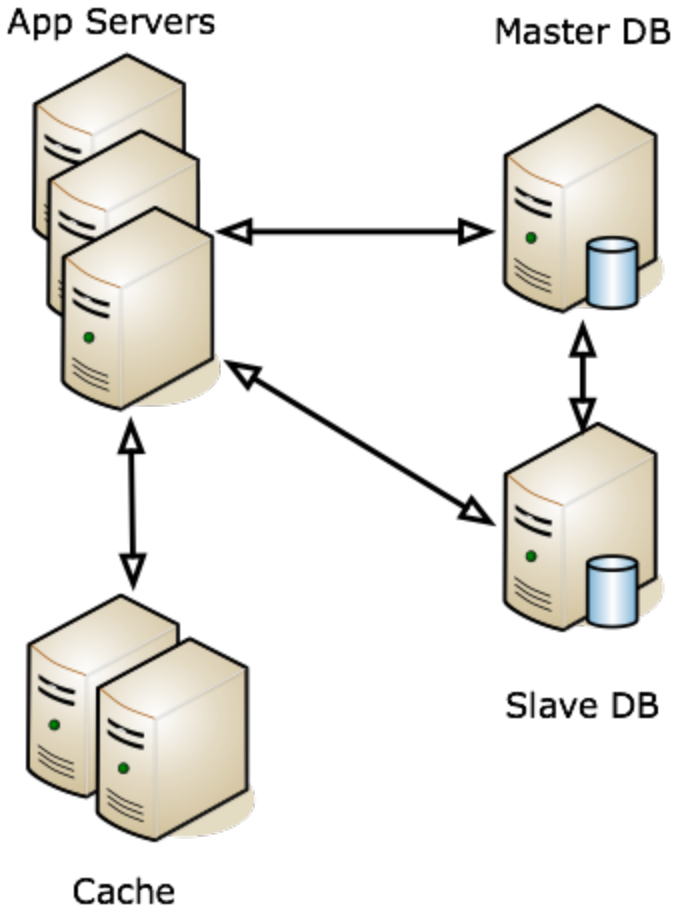
e

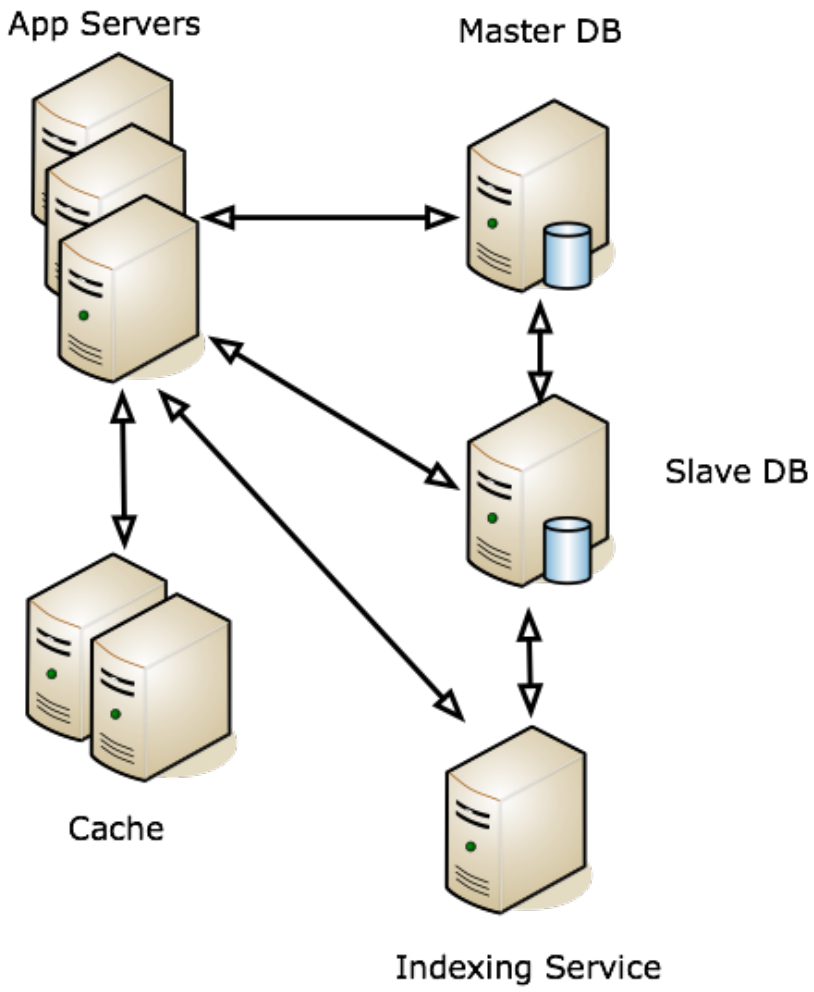


cresce



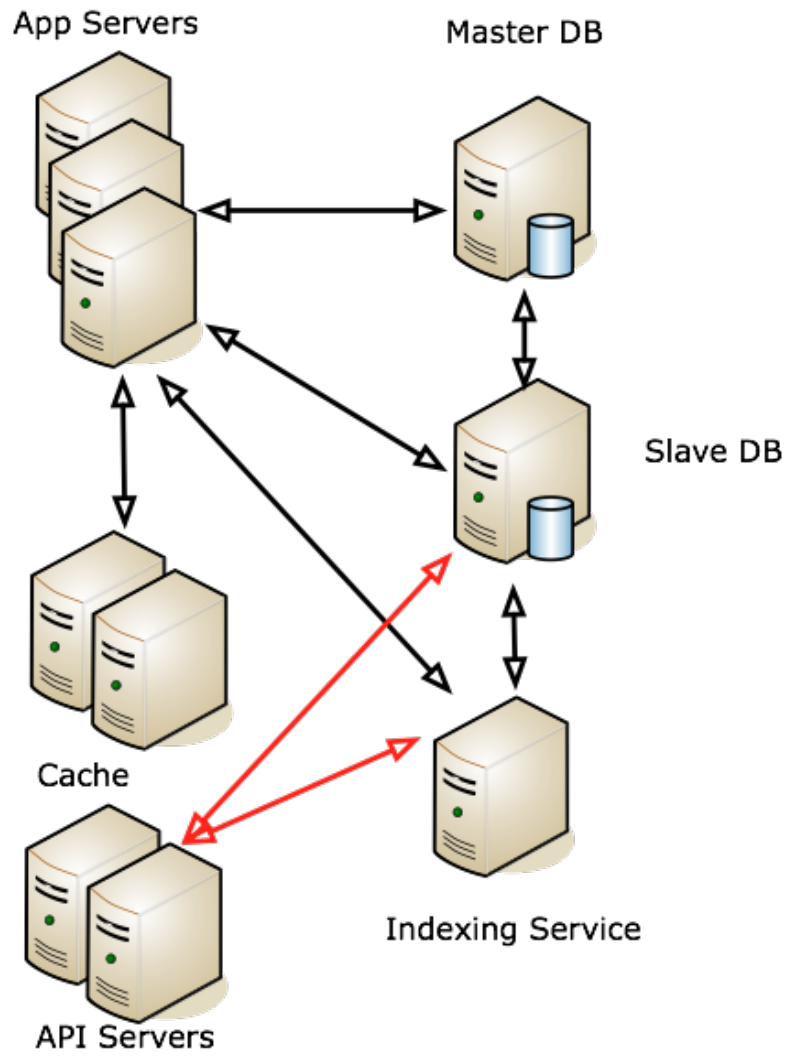
mais



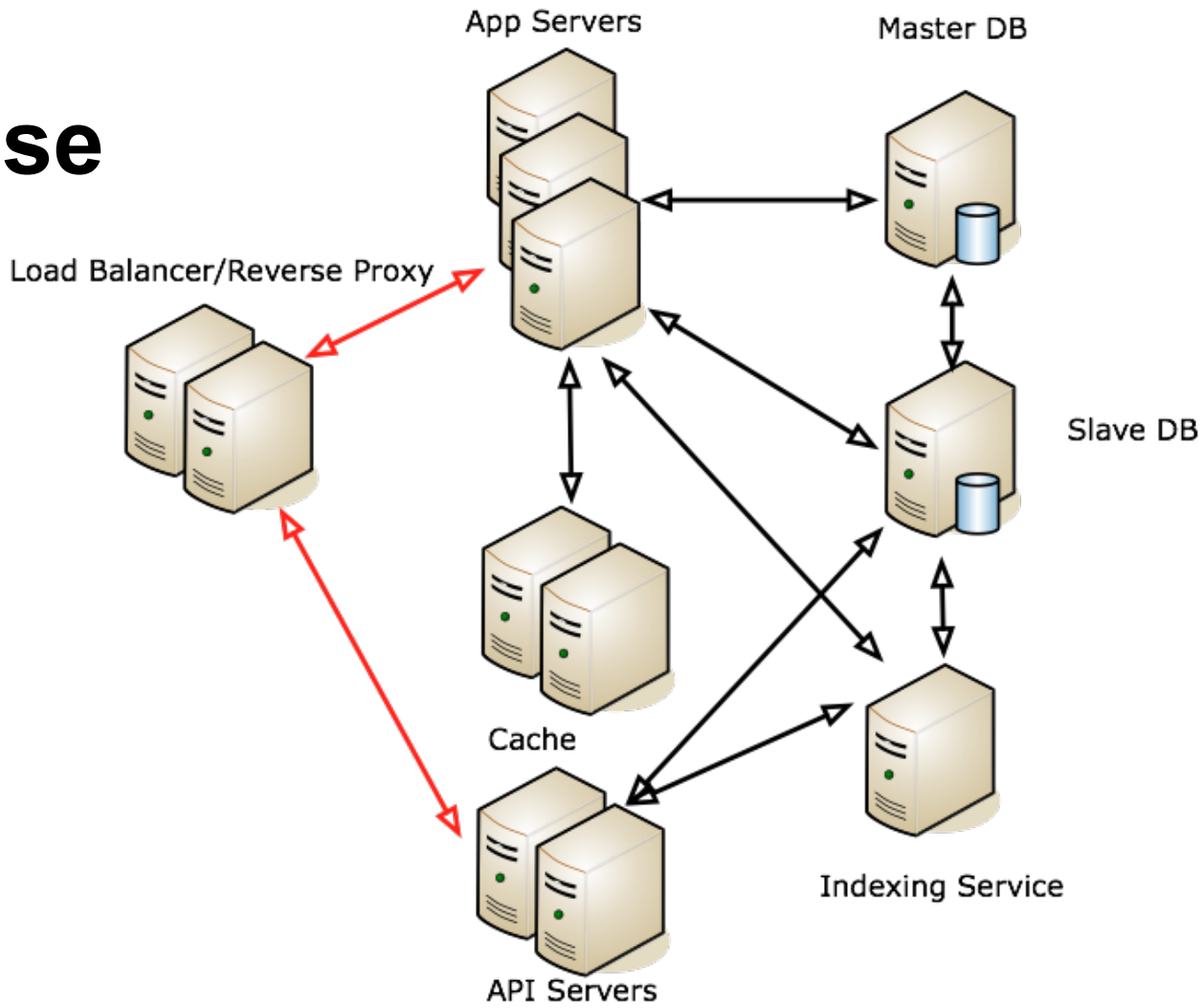


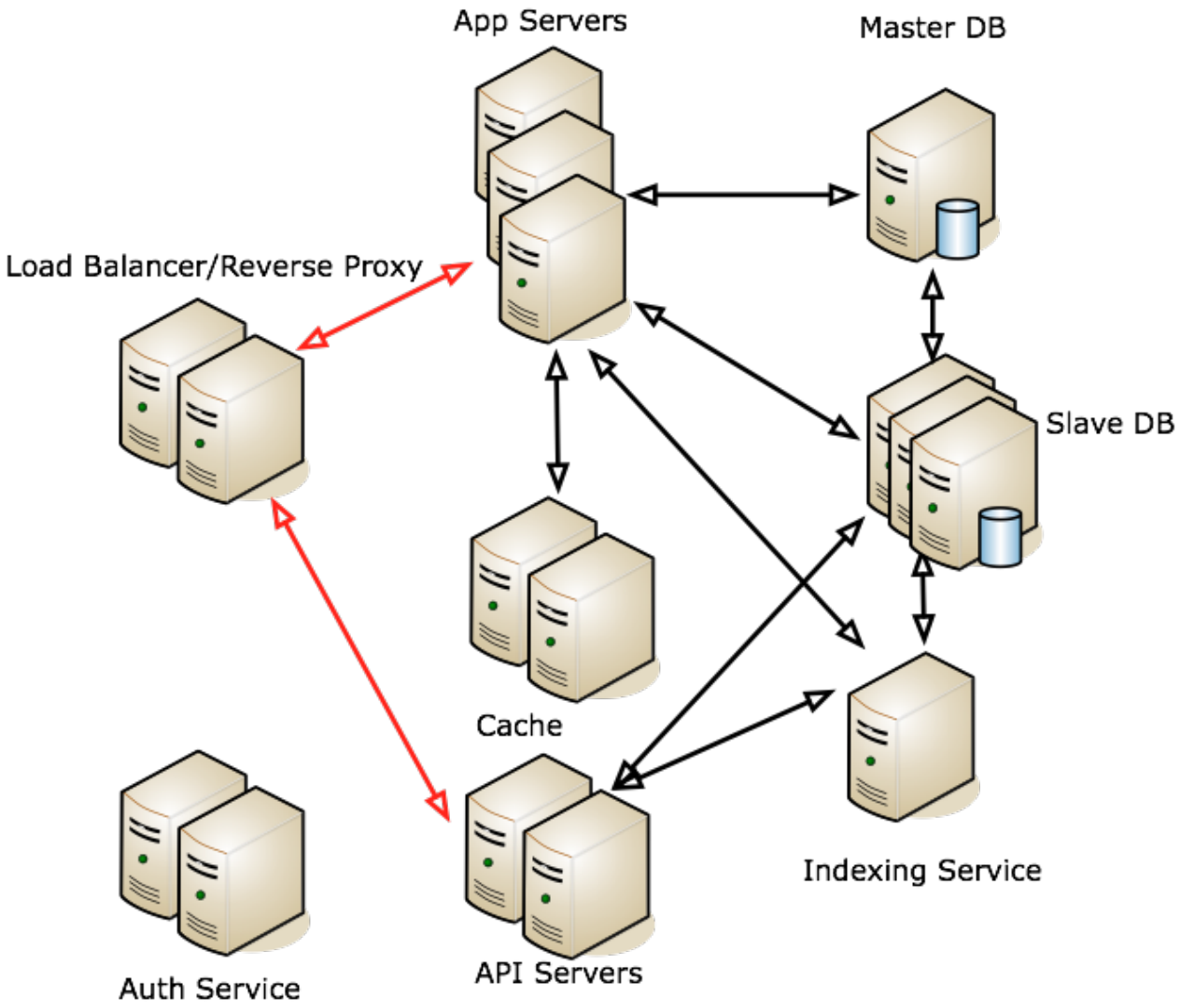
ou

menos


App Servers
Master DB
Slave DB
Cache
Indexing Service
API Servers

desse


App Servers
Master DB
Load Balancer/Reverse Proxy
Slave DB
Cache
Indexing Service
API Servers


App Servers
Master DB
Load Balancer/Reverse Proxy
Slave DB
Cache
Indexing Service
Auth Service
API Servers

jeito

Por que as coisas dão errado ?

Lit. Popular: Falácias de Sistemas (distribuídos)

"*Essentially everyone, when they first build a distributed application, makes the following eight assumptions. All prove to be false in the long run and all cause big trouble and painful learning experiences.*" -- L Peter Deutsch (1994)

1. The network is reliable.
2. Latency is zero.
3. Bandwidth is infinite.
4. The network is secure.
5. Topology doesn't change.
6. There is one administrator.
7. Transport cost is zero.
8. The network is homogeneous.

Lit. Popular: Algumas regras para Engenharia

1. A melhor solução para um problema é não tê-lo
2. Hacks são permanentes (principalmente os feios)
3. Não existe infraestrutura em stand-by: existe o que você usa e o que não vai funcionar quando você precisar
4. A primeira falácia de automação é fazer máquinas executar passos de um processo manual humano
5. Não são *features* (não são negociáveis): **Segurança, Disponibilidade e Performance**.

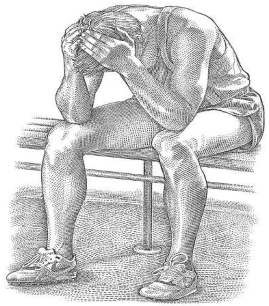
(http://blog.b3k.us/2012/01/24/some-rules.html)

Lit. Popular: Leis da interweb

- **Lei de Parkinson:** O trabalho se expande até preencher todo o tempo disponível.
- **W. Edwards Demming:** Não é suficiente fazer o seu melhor; Você precisa **saber o que fazer**, e então fazer o seu melhor.
- **Cisne Negro (Black swan):** Um evento que é uma surpresa para o observador, tem um efeito enorme e posteriormente é racionalizado como algo esperado.
- **Lei de Conway:** Organizações que projetam sistemas (...) são limitadas a produzir sistemas que são cópias das estruturas de comunicação destas organizações.

Por que as coisas dão errado ?

- **Saber o que fazer**: Desenvolvimento baseado em features
- **Black Swan**: Quais as chances de situações ruins acontecerem ?
- **Lei de Parkinson**: Quais são os deadlines. O que vamos deixar de fazer e aprender para alcança-los ?
- **Lei de Parkinson**: ReReReReReReReRepriorizações
- **Lei de Parkinson:** Impressão de nunca terminar nada após entender o problema.
- **Black Swan**: Excesso de confiança em ferramentas e racionalização de problemas. Soluções quebra galho acumuladas.
- **Black Swan**: Inocência ao abordar uma feature ou problema
- **Saber o que fazer**: Inexperiência do time trabalhando ou das pessoas
- **Black Swans**: Empolgação



Parkinson's law: **work expands to fill the time available**
It is not enough to do your best; you must ***know what to do,*** and then do your best. **W. Edwards Demming**
Black swan: An event that is a surprise to the observer, has a major effect and afterwards is rationalized in hindsight as it could have been expected

Desmistificando o PaaS genérico

Indice complicometrico de mudanças

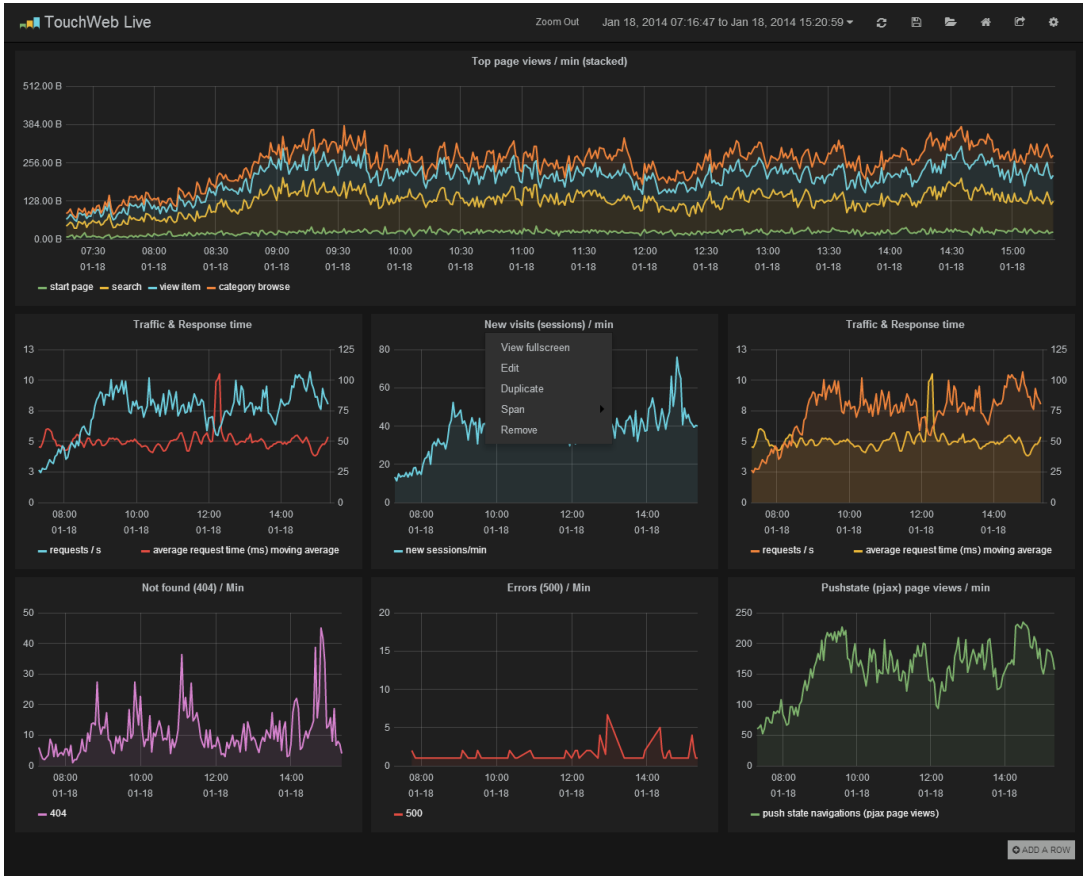
$$\frac{[(\textit{legado} + \textit{sistemas novos incompletos} + \textit{reescritas}) * \textit{idade}\ ]\ ^{\wedge}\ \textit{urgência do negócio}}{\text{número de desenvolvedores} + \text{número de sysadmins}}$$

Componentes importantes dos sistemas de sua empresa: auth, serviço interno de ordens, backoffice de clientes, logistica. Qual o custo adicional de criar um PaaS genérico e depois o seu PaaS™ ?

E agora ?

Casos e Sugestões

Métricas, métricas everywhere



(a soma das partes equivale ao timeout do todo)

Trabalhar com requisitos não funcionais

- Logs normalizados
  - com contexto (user, host, operation, ...)
- Reload automatico (configs, templates, ...)
- Ferramentas administrativas
  - quanto mais melhor, não são *features*
- **Contadores** e **estatísticas**
- **Gráficos** e **dashboards**
- **Requisitos também vem da operação**

Trabalhar com requisitos não funcionais

- Logs normalizados
  - com contexto (user, host, operation, ...)
- Reload automatico (configs, templates, ...)
- Ferramentas administrativas
  - quanto mais melhor, não são *features*
- **Contadores** e **estatísticas**
- **Gráficos** e **dashboards**
- **Requisitos também vem da operação**

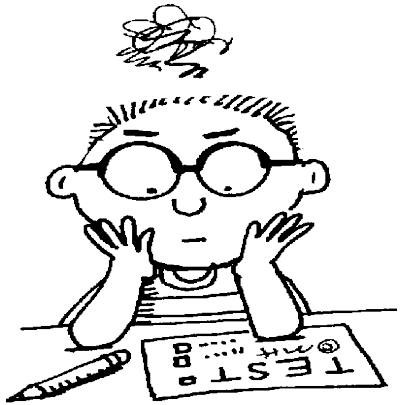
Lit. Popular: Algumas regras para Engenharia

1. A melhor solução para um problema é não tê-lo
2. Hacks são permanentes (principalmente os feios)
3. Não existe infraestrutura em stand-by: existe o que você usa e o que não vai funcionar quando você precisar
4. A primeira falácia de automação é fazer máquinas executar passos de um processo manual humano
5. **Não são *features* (não são negociáveis): Segurança, Disponibilidade e Performance.**

(http://blog.b3k.us/2012/01/24/some-rules.html)

Testes que importam

- Teste funcional totalmente automatizado
- CI: Sem mocks
- Log replay
- Teste de concorrência/carga
- Bullet-proof tests
  - Kill -STOP test
  - Load test (iozone, gcc test)
  - Kill VM/Proc test (chaos monkey)
- Quanto custa um restart quando as sessões estão amarradas a servidores ou entradas em cache sustentam seu dia a dia



Caso drama: Cade meu cache? Cade meu I/O?

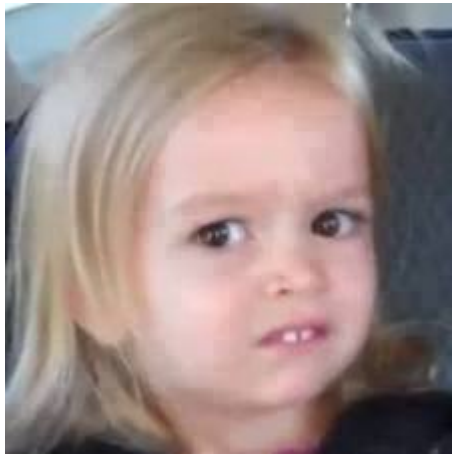
Cache (memcached, db, fs cache, app cache)

Read 1 MB sequentially from memory: 0.25 ms

Read 1 MB sequentially from SSD*: 1ms (4X memory)

Disk seek: 10ms 40x memory, 10x SSD

Read 1 MB sequentially from disk: 20ms 80x memory, 20x SSD



1GB em cache -> 1024 * 0.25ms = 256ms

1GB em SSD -> 1024 * 1ms = 1024ms = 1.24s

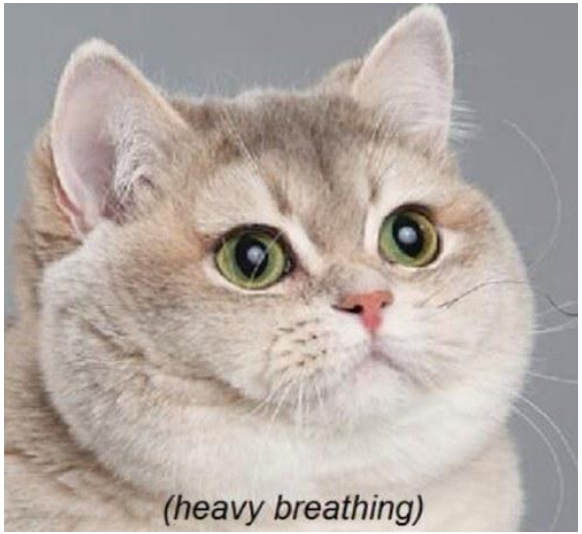
1GB em disco -> 1024 * 20ms = 20.480ms = 20.48s

- **+ Roundtrip de rede**
- **+ Timeouts**

(fonte: http://chu.pe/4sd)

Mudanças de schema de banco de dados

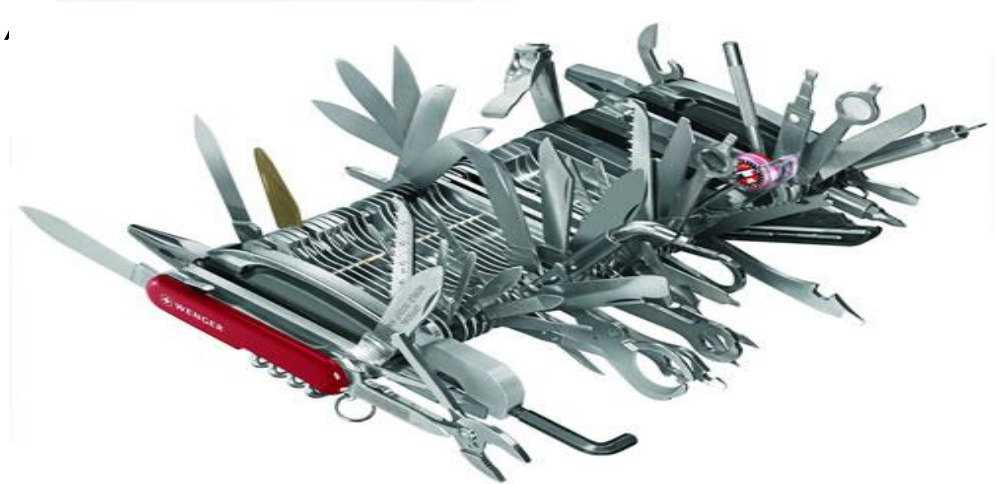
1. Conheça seu ORM e seu problema de impedância
2. Não conte com suas otimizações no banco
3. Alterações incrementais de esquema (db schema)
4. Faça um teste de restaurar seu esquema + dados de produção com carga sintética
5. Lembre do I/O e da concorrência entre mudanças no banco e produção/clientes



Caso drama: Migrations na minha tabela gigante

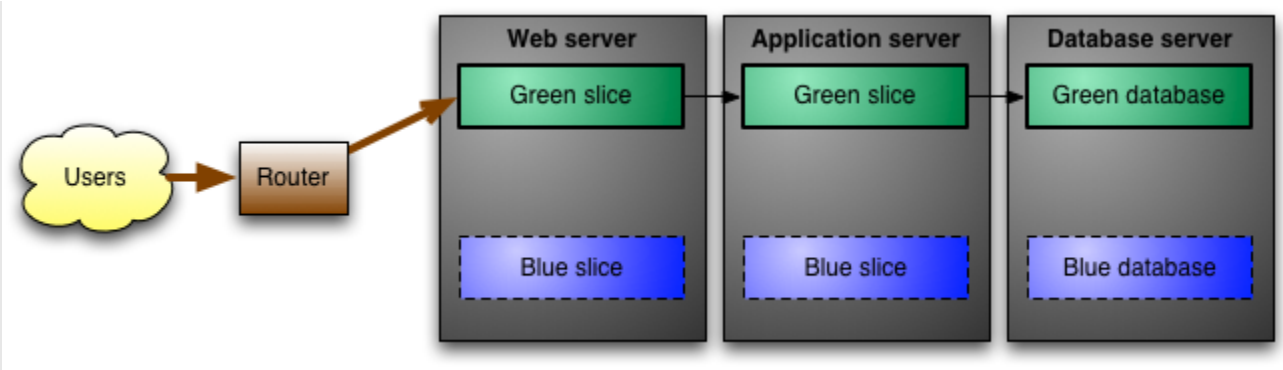
Como introduzir uma nova feature

- On/Off switch
- Testes de regressão (banco de dados)
- Testes funcionais completos
- Teste do Upgrade (versões intermediárias)
- Teste do Downgrade (rollback)



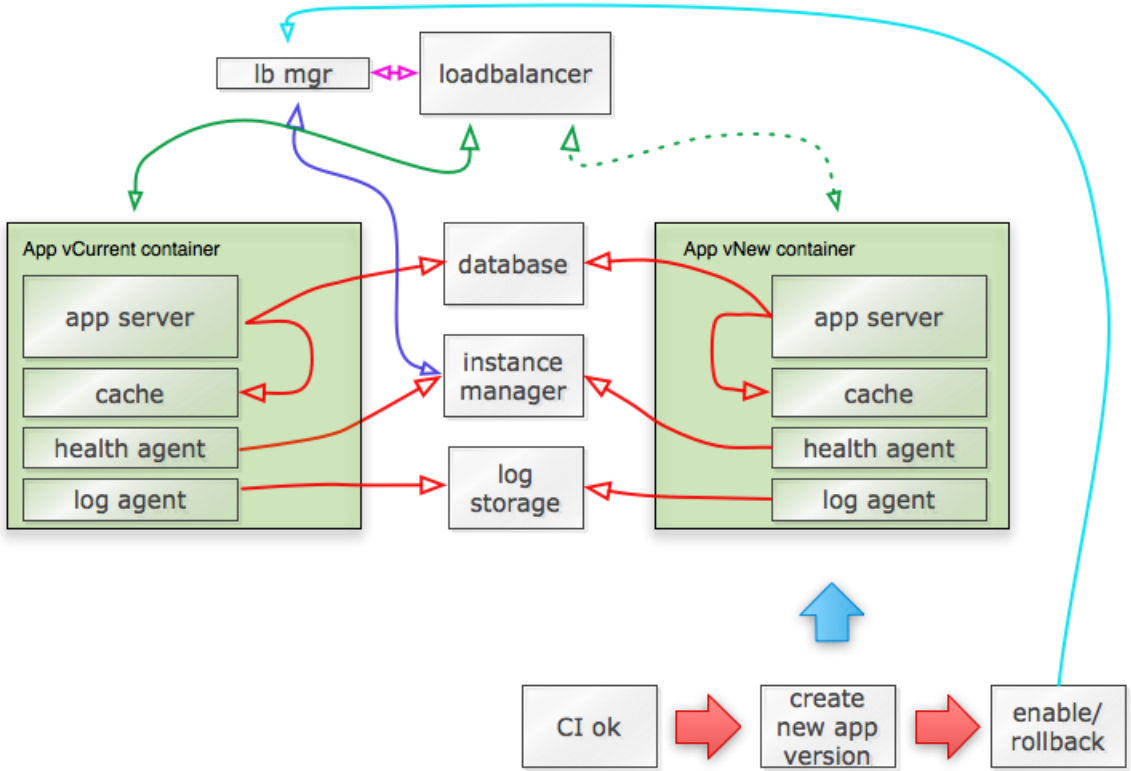
Blue-green Deployment e Loadbalancer

Blue-green deployment (sem variações) *



* http://martinfowler.com/bliki/BlueGreenDeployment.html

Blue-green Deployment e Loadbalancer



The Distributed Developer Stack Field Guide

- The cloud is the default platform.
- The codebase is in git.
- **The environment is automated in the code.**
- **Tests done in code, not by a QA department.**
- **Realtime chat and chatbots.**
- CI servers deploy code, not ops.
- The application runs locally on development.
- **The monitoring infrastructure is critical.**
- Containers are the default deployment target.

Bibliografia sugerida

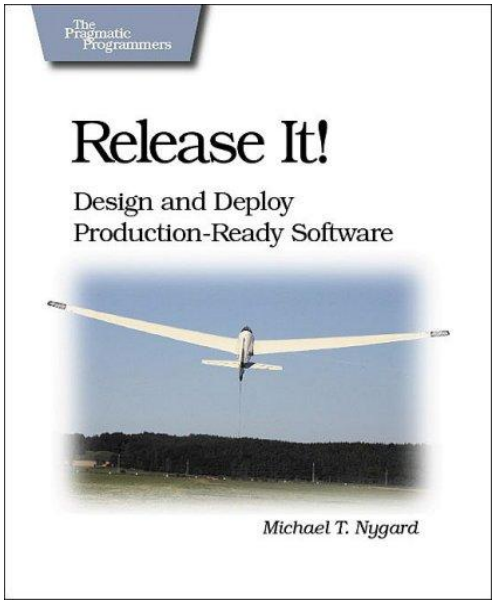
Bib: Distributed Developer Stack Field Guide

Uma introdução para o desenvolvedor de sistemas distribuidos (basicamente todos)

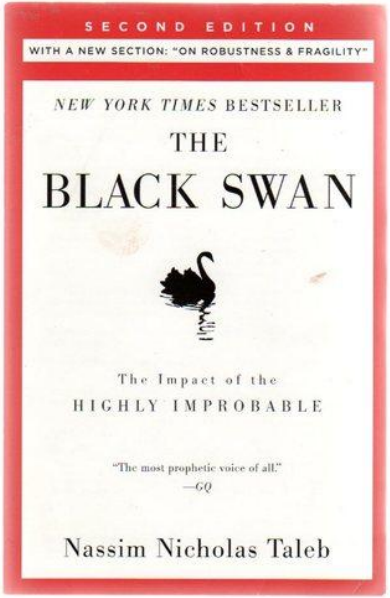
http://sites.oreilly.com/odewahn/dds-field-guide/

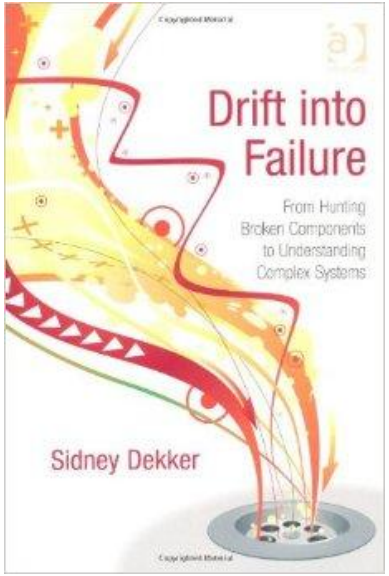
Bib: Stability Patterns

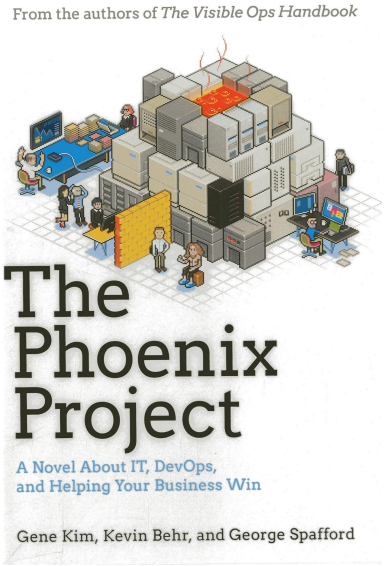
- **Use Timeouts**
- **Circuit Breaker**
- Bulkheads
- Steady State
- **Fail Fast**
- Handshaking
- **Test Harness**
- **Decoupling Middleware**



Bibliografia extra







Julho/2015
Editora Casa do Código
Gleicon Moraes

BE EXCELLENT TO EACH OTHER

Perguntas ?

obg dnad