Ustore

Desmistificando Microservices: Projetando Arquiteturas Efetivamente Escaláveis

Prof. Vinicius Cardoso Garcia
vcg@cin.ufpe.br :: vinicius3w.com :: assertlab.com

FORTALEZA, 18 A 22 DE SETEMBRO DE 2017

AssertLab
Advanced Software and Systems Engineering Research Technologies

Centro de Informática UFPE 40 anos

Ustore

Licença do material

Este Trabalho foi licenciado com uma Licença

Creative Commons - Atribuição-NãoComercial-CompartilhaIgual 3.0 Não Adaptada

Mais informações visite

http://creativecommons.org/licenses/by-nc-sa/3.0/deed.pt

AssertLab
Advanced Software and Systems Engineering Research Technologies

Centro de Informática 40 anos

Ustore

Download

Apresentação disponível na URL:

http://bit.ly/microservices-cbsoft-2017

AssertLab
Advanced Software and Systems Engineering Research Technologies

Centro de Informática 40 anos

Ustore

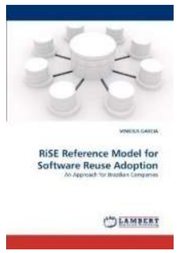
Quem sou eu?

- **Professor** @ Cin-UFPE [Sistemas de Informação, Pós-Graduação em Ciência da Computação], Aug-2010
  - 1/44 MSc; 5/9 (+2 co) DSc :: http://viniciusgarcia.me
  - Diretor de Sistemas @ NTI (2013)
- **ASSERT** [Advanced System and Software Engineering Research Technologies] Lab :: http://assertlab.com
- **Ikewai** [http://www.ikewai.com]: rede de business designers
  - **Netweaver**
- **USTO.RE** [http://usto.re] :: Smart Cloud Storage; Multicloud Environment
  - **Chief Scientist Officer** (CSO)
- **INES** [Instituto Nacional para Ciência e Tecnologia em Engenharia de Software] :: http://www.ines.org
- Engenheiro de Sistemas :: 2005 ~ 2010 @ CESAR
- D.Sc. em Engenharia de Software, Feb-2010

ikewai

Ustore

INES



http://amzn.to/ILF8k

AssertLab
Advanced Software and Systems Engineering Research Technologies

Centro de Informática

40 anos

Ustore

Mais recursos

- https://www.infoq.com/articles/cloud-native-architectures-matt-stine/ (May 2, 2015), InfoQ
- Cloud-native architectures will be the default soon, By Andy Patrizio, Network World | JUN 9, 2017
- Learning Path: Modern DevOps, UDEMY, Criado por Packt Publishing, Última atualização em 8/2017
- [IF1004] Desmistificando Microsserviços e DevOps: Projetando Arquiteturas Efetivamente Escaláveis, Sistemas de Informação, CIn-UFPE, 2017.2

AssertLab
Advanced Software and Systems Engineering Research Technologies

Centro de Informática UFPE 40 anos

Ustore

Migrando Aplicações para Arquiteturas Nativas de Nuvem

AssertLab
Advanced Software and Systems Engineering Research Technologies

Centro de Informática 40 anos

Ustore

Sumário

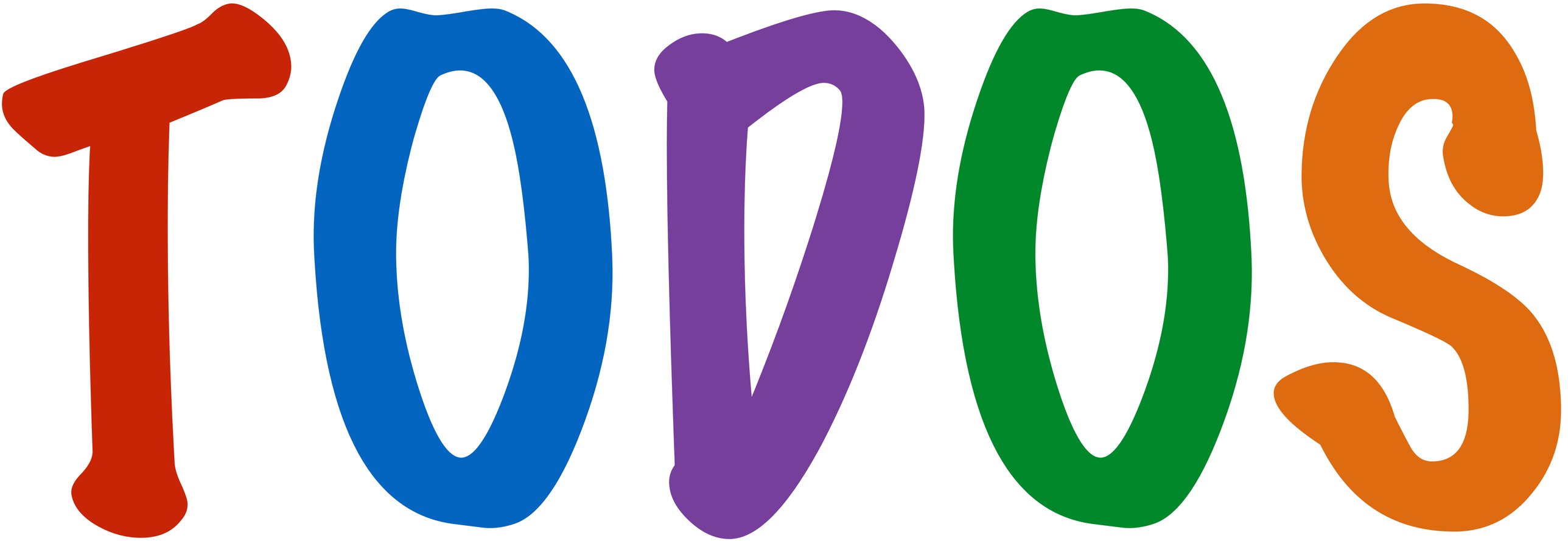
- O surgimento da "Cloud-Native"
  - Por que Arquiteturas de Aplicações Cloud-Native?
  - Definindo Arquiteturas Cloud-Native
- Mudanças necessárias
  - Culturais
  - Organizacionais
  - Técnicas
- Livro de Receitas da Migração para Nuvem
  - Receitas de Decomposição
  - Receitas de Sistemas Distribuídos

Ustore

O surgimento da “Cloud-Native"

Software is eating the world.
— Mark Andreessen

Ustore

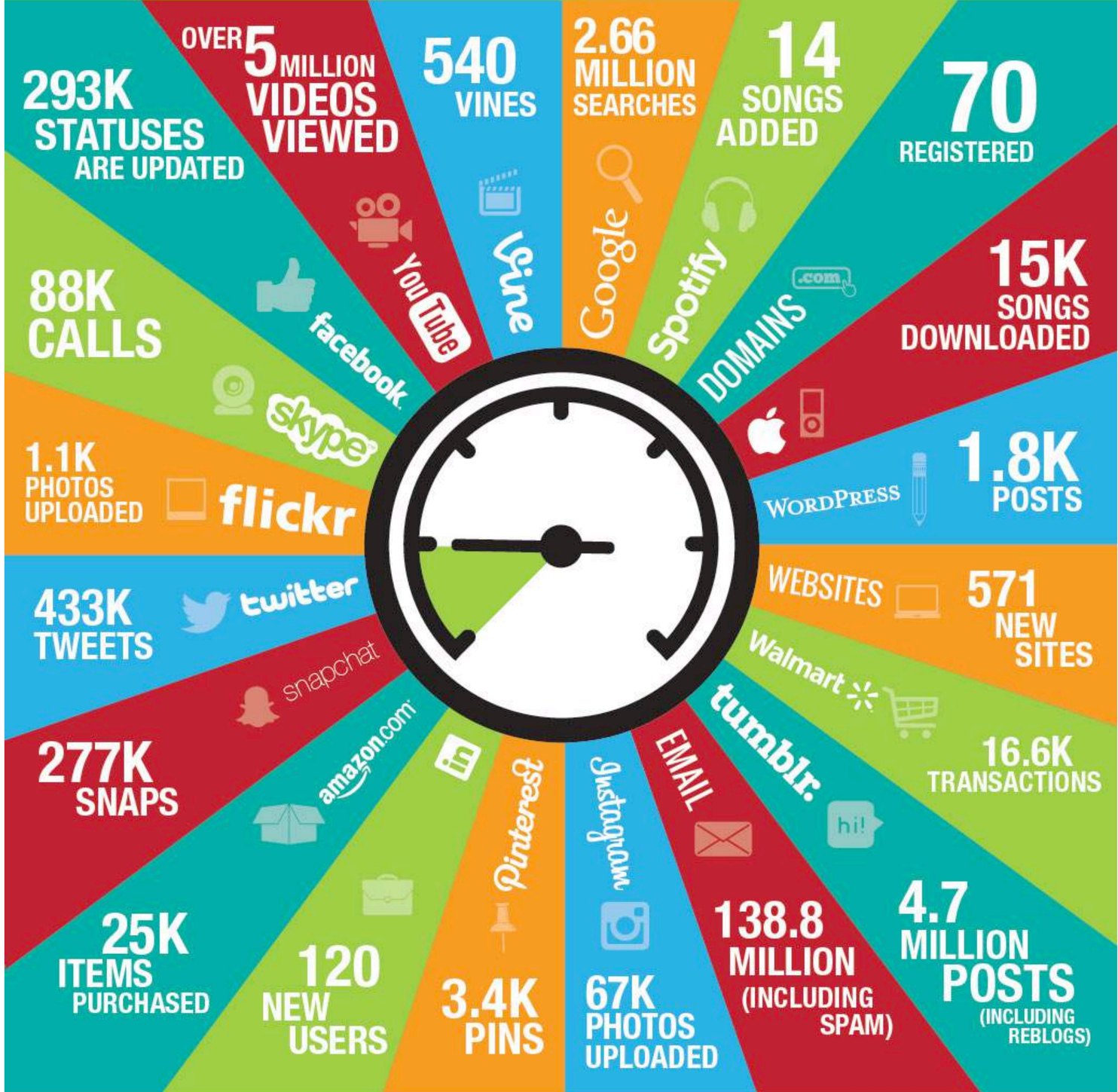


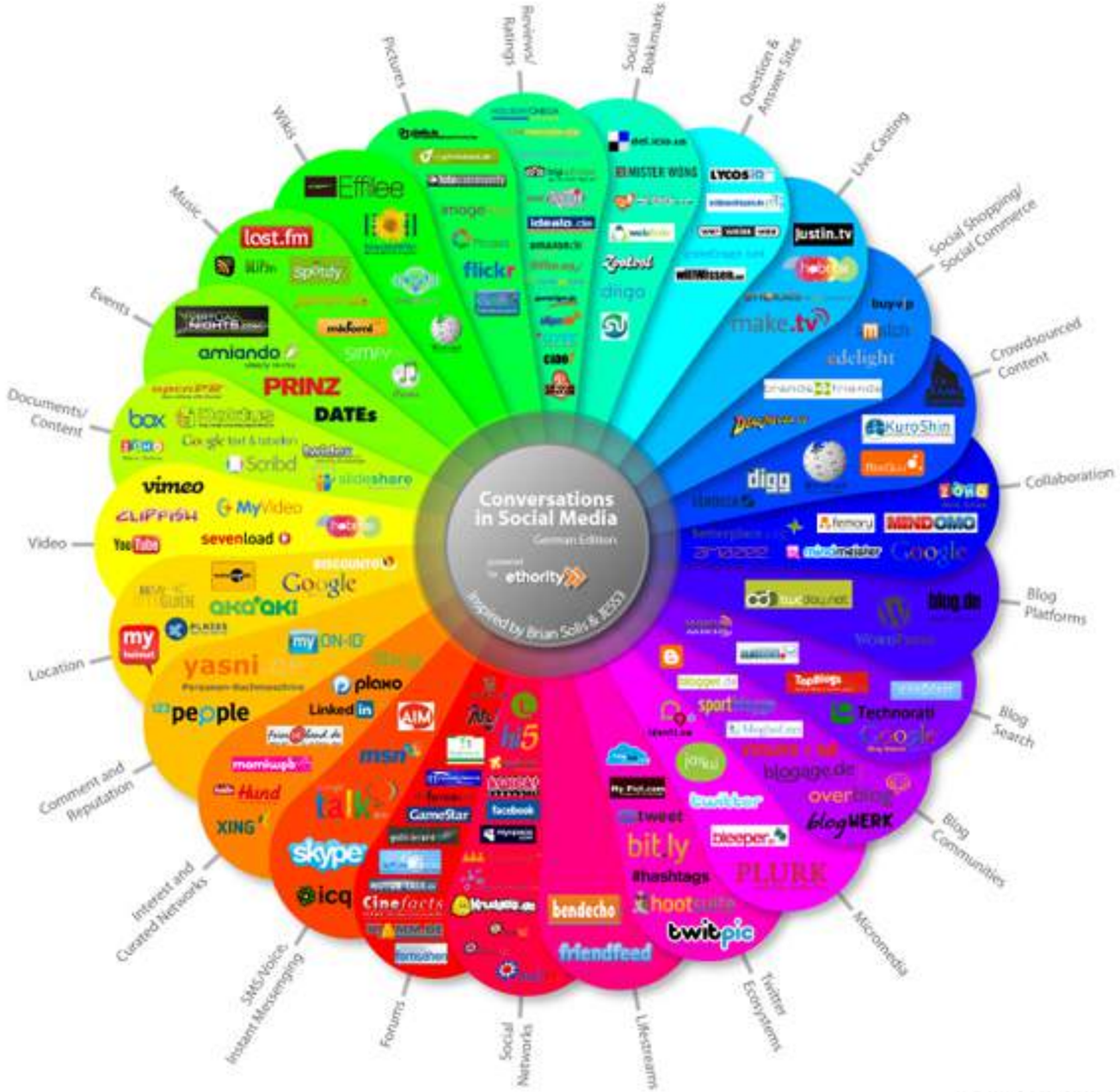
Ustore

Conectados

Online in 60 seconds

2014





Mudança nas formas de relacionamento

"Computing means **CONNECTING.**"

Wade Roush (2006)

Ustore

"Every company is a technology company" – more and more evidence..

by **Mark Raskino** | November 28, 2013 | Submit a Comment

The quote in the title of this post comes from Peter Sondergaard's section of the Gartner Symposium keynote 2013. I just keep on seeing more examples that show how it is true. This week I noticed 3 news stories that provide more evidence.

Argos Digital Store

Reuters reports that Argos is introducing "digital stores". Argos is a major UK home goods retailer, that for many years has operated an unusual catalog shopping in-store model. Now it is doing away with paper catalogs and replacing them with tablet based versions. Take a look at the picture – perhaps this is one way store based retailers can compete with Amazon.

Marlborough maker to introduce an e-cigarette in 2014

The Wall Street Journal reports that PMI – the manufacturers of Marlborough and other big brand cigarettes has announced it will enter the e-cigarette market next year. It will be joining Lorrilard, BAT and others who have already responded to the rapid growth in this category that has been largely developed by start-ups. Don't be fooled into thinking this is just an "electrical" product. Lorrilard's "blu" brand already has wireless and social features. Digital innovation in this area will become very lively indeed.

Ustore

"Software is eating the world”

- As indústrias estáveis há anos estão sendo substituídas pelas empresas baseadas em software

- O que essas empresas inovadoras têm em comum?
  - Velocidade da inovação
  - Serviços sempre disponíveis
  - Escala da Web
  - Experiências de usuário centradas no celular

AssertLab
Advanced Software and Systems Engineering Research Technologies

Centro de Informática 40 anos

Ustore

Surgimento de um novo modelo de TI

A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., servers, storage, networks, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

– NIST

"Computing may someday be organized as a public utility, just as the electricity is organized as a public utility"

– John McCarthy, speech at MIT in 1961

Características essenciais da Nuvem

- Autosserviço sob demanda
- Amplo acesso à rede
- Pool de recursos
- Rápida elasticidade
- Serviços mensuráveis

Modelos de Serviço

- SaaS
- PaaS
- IaaS
- XaaS

Modelos de Implantação

- Público
- Privado
- Comunitário
- Híbrido

Ustore

Por que Aplicações com Arquiteturas Nativas para Nuvem?

"Cloud is about **how** computing is done, not **where**"

AssertLab
Advanced Software and Systems Engineering Research Technologies

Centro de Informática 40 anos

Ustore

Velocidade

- Velocidade para inovar, experimentar e entregar é fundamental no mercado competitivo
  - O custo de corrigir um erro também é medido em uma escala de tempo.

- Várias implantações por dia
  - Se você pode implantar centenas de vezes por dia, você pode se recuperar de erros quase que instantaneamente!

- Nuvem [elasticidade & self-service]
  - Chamadas para uma API de serviço em nuvem <> processo de implantação de ambiente "tradicional"

Ustore

Velocidade

“How long would it take your organization to deploy a change that involves just one single line of code?”

—Mary Poppendick

Ustore

Segurança

- Às vezes, ser rápido não é o suficiente
- Arquiteturas de aplicativos nativos em nuvem equilibram a necessidade de velocidade com estabilidade, disponibilidade e durabilidade
- Então, como fazemos para ser rápidos e seguros?

AssertLab
Advanced Software and Systems Engineering Research Technologies

Centro de Informática UFPE 40 anos

Ustore

Velocidade e Segurança

- Visibilidade
  - Identificar a falha quando ela acontecer ~> measure everything
- Isolamento de falhas
  - Limitar o alcance de componentes ou recursos que podem ser afetados por uma falha ~> microservices
- Tolerância a falhas
  - Previnir falhas em cascata ~> circuit breaker [Release It!, Mike Nygard]
- Recuperação automatizada
  - health check

Ustore

Escala

- À medida que a demanda aumenta, devemos escalar nossa capacidade de atender essa demanda

- Não podemos comprar hardware indefinidamente ~> não podemos desenvolver como se não houvesse limites no cartão de crédito
  - escalabilidade horizontal
  - virtualização
  - conteinerização

Ustore

Escala

- Como o descartável interage com o persistente?
  - Tradicionalmente a escalabilidade da persistência é vertical…
- Aplicações stateless podem ser rapidamente criadas e destruídas

Ustore

Aplicações móveis e diversidade de clientes

- Em janeiro de 2014, os dispositivos móveis representavam 55% do uso da Internet nos Estados Unidos.

- Devemos assumir que nossos usuários estão caminhando com supercomputadores multicore nos bolsos [Mi Mix 2 com 6GB de RAM e Octa-core]

- Quais são os desafios arquiteturais com este cenário?!

Ustore

Aplicações móveis e diversidade de clientes

- As aplicações móveis muitas vezes têm que interagir com vários sistemas legados
- Bem como com vários microsserviços em uma aplicação nativa da nuvem
- Mais uma vez, quais os desafios arquiteturais?
  - API Gateway [design pattern]

Ustore

Definindo Arquiteturas Nativas para Nuvem

What are the key characteristics of cloud-native application architectures?

AssertLab
Advanced Software and Systems Engineering Research Technologies

Centro de Informática 40 anos

Ustore

heroku

Twelve-Factor Applications

- twelve-factor app é uma coleção de padrões para aplicações nativas pra nuvem, originalmente desenvolvido pelo time de engenheiros da Heroku

- Cloud Foundry, Heroku, e Amazon Elastic Beanstalk são otimizados para implantação de aplicações twelve-factor

- Se refere a uma única unidade de implantação

Twelve-Factor Applications

I. Codebase: One codebase tracked in revision control, many deploys

II. Dependencies: Explicitly declare and isolate dependencies

III. Config: Store config in the environment

IV. Backing services: Treat backing services as attached resources

V. Build, release, run: Strictly separate build and run stages

VI. Processes: Execute the app as one or more stateless processes

VII. Port binding: Export services via port binding

VIII. Concurrency: Scale out via the process model

IX. Disposability: Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity: Keep development, staging, and production as similar as possible

XI. Logs: Treat logs as event streams

XII. Admin processes: Run admin/ management tasks as one-off processes

Ustore

Características 12factor

- Fazem poucas ou nenhuma suposição sobre os ambientes nos quais serão implantados
- Mecanismo simples e consistente, facilmente automatizado, para fornecer rapidamente novos ambientes e implantar as apps neles
- Também se prestam bem à idéia de efemeridade, ou aplicações que podemos "jogar fora" com muito pouco custo.
  - Recuperação automática de eventos de falha muito rapidamente

Ustore

Microsserviços

- Representam a decomposição de sistemas de negócios monolíticos em [micro]serviços que podem ser implementados de forma independente, que fazem "uma coisa bem".

- Essa única coisa geralmente representa uma capacidade de negócio, ou a unidade de serviço "atômica" pequena mas que oferece valor comercial.

Ustore

Microsserviços: speed, safety, and scale

- Desacoplamento do contexto do domínio leva ao desacoplamento dos ciclos de mudança associados ~> CI/CD

- O desenvolvimento pode ser acelerado ao dar escalabilidade ao próprio desenvolvimento
  - Fred Brooks, The Mythical Man-Month
  - Paralelização do trabalho

- Melhor entendimento e adoção de novas tecnologias

Ustore

Infraestrutura ágil de autoatendimento

- Normalmente, o time que desenvolve, é o time responsável pelas atividades de implantação e operação

- O time não precisa pensar em onde seu código está sendo executado ou como chegou lá, pois a plataforma cuida dessas preocupações de forma transparente.
  - O mesmo para serviços de backend

Ustore

Infraestrutura ágil de autoatendimento

- Essas plataformas também oferecem uma ampla gama de recursos operacionais adicionais
  - Escalabilidade automatizada e sob demanda de instâncias de aplicações
  - Gerenciamento de saúde de aplicativos
  - Roteamento dinâmico e balanceamento de carga de solicitações para e entre instâncias de aplicações
  - Agregação de logs e métricas

Ustore

Colaboração Baseada em API

- Interação entre serviços acontece via APIs publicadas e versionadas
  - HTTP REST-style com JSON

- Novas funcionalidades são implantadas sempre que houver necessidade, sem sincronizar com outras equipes, desde que não quebrem nenhum contrato de API existente

Ustore

Antifragilidade

- O conceito de antifragilidade foi introduzido no livro Antifragile (Random House) de Nassim Taleb: Se a fragilidade é a qualidade de um sistema que fica mais fraco ou quebra quando submetido a estressores, então, o que é o oposto disso?
- Muitos responderiam com a idéia de robustez ou resiliência - coisas que não quebram ou ficam mais fracas quando submetidas a estressores.
- No entanto, o Taleb apresenta o oposto da fragilidade como antifragilidade, ou a qualidade de um sistema que se torna mais forte quando submetido a estressores.

Ustore

Resumindo

- As motivações para migrar para arquiteturas de aplicações nativas da nuvem em termos de habilidades que queremos fornecer ao nosso negócio via software
  - Velocidade, Segurança, Escala, Mobilidade, Aplicações Twelve-factor, Microsserviços, Infraestrutura ágil de autoatendimento, Colaboração Baseada em API e Antifragilidade

Ustore

Mudanças necessárias

"All we are doing is looking at the timeline from the moment a customer gives us an order to the point when we collect the cash. And we are reducing that timeline by removing the nonvalue-added wastes."
— Taichi Ohno

Ustore

Mudanças Culturais

AssertLab
Advanced Software and Systems Engineering Research Technologies

Centro de Informática 40 anos

Ustore

De Silos para DevOps

- Empresas de TI são normalmente organizadas em silos
  - Desenvolvimento de Software
  - Garantia de Qualidade
  - Administração de Banco de Dados
  - Administração de Sistemas
  - Operações de TI
  - Administração/Gestão de Entregas/Releases
  - Gestão de Projetos

Peculiaridades
Hierarquias
Técnicas de gestã
Suítes de ferrament
Processos
Comunicação
Vocabulário
Habilidades

AssertLab
Advanced Software and Systems Engineering Research Technologies

Centro de Informática UFPE 40 anos

Ustore

Conflitos :: Desenvolvimento

- A missão do desenvolvimento geralmente é vista como fornecer valor adicional à organização através do desenvolvimento de recursos de software.

- Assim, a missão do desenvolvimento pode ser descrita como "entregar mudanças" e é muitas vezes incentivada em relação à quantidade de mudança que ela entrega.

Ustore

Conflitos :: Operações

- A missão do time de operações de TI pode ser descrita como a de "evitar mudanças".
- Como? As operações de TI geralmente são encarregadas de manter os níveis desejados de disponibilidade, resiliência, desempenho e durabilidade dos sistemas de TI.
- Portanto, eles são muitas vezes incentivados a manter os principais indicadores de desempenho (KPIs) como o tempo médio entre falhas (MTBF) e tempo médio para recuperação (MTTR).

Ustore

Conflitos :: Dev vs Ops

- Colaboração, comunicação e transferência simples do produto do trabalho tornam-se tediosas e dolorosas na melhor das hipóteses, e absolutamente caóticas (mesmo perigosas) na pior das hipóteses.

- Solução?
  - Criação de processos [pesados] impulsionados por sistemas baseados em tickets e reuniões de comitês…

Ustore

Dev & Ops

- Ideias opostas aos princípios de velocidade providos pelos ambientes nativos de nuvem

- Silos e processos são motivados, usualmente, pela falsa ideia de segurança… mas de fato, oferecem muito pouca segurança adicional ou ainda tornam as coisas piores!

Ustore

DevOps

- Derrubar estes silos e construir conjuntos de ferramentas, vocabulários e estruturas de comunicação compartilhados em serviço de uma cultura focada em um único objetivo: entregar valor de forma rápida e segura.
- As estruturas de incentivo são então criadas reforçam e atribuem comportamentos que lideram a organização na direção dessa meta.
- A burocracia e o processo são substituídos por confiança e responsabilidade.

Ustore

Do equilíbrio pontuado à entrega contínua

- As empresas normalmente adotam estruturas de governança centralizadas em torno da arquitetura de aplicativos e gerência de dados
- Foco em evitar:
  - Inconsistências generalizadas em pilhas de tecnologia, diminuindo o custo geral de manutenção
  - Inconsistências generalizadas em escolhas arquitetônicas, permitindo uma visão comum do desenvolvimento de aplicativos em toda a organização
  - As preocupações transversais, como a conformidade regulamentar, podem ser manipuladas de maneira consistente para toda a organização.
  - A propriedade dos dados pode ser determinada por aqueles que têm uma visão ampla de todas as preocupações organizacionais.

Ustore

Do equilíbrio pontuado à entrega contínua

- Assim como as arquiteturas de aplicativos monolíticos podem criar gargalos que limitam a velocidade da inovação técnica, estruturas de governança monolíticas podem fazer o mesmo

- A adoção de arquiteturas de aplicações nativas da nuvem é quase sempre acompanhada de um movimento para a governança descentralizada.

- Estruturas mínimas e leves que são impostas aos padrões de integração utilizados entre serviços independentemente desenvolvidos e implantados (por exemplo, eles preferem as API HTTP REST JSON em vez de muitos estilos diferentes de RPC).

Mudanças Organizacionais

Ustore

Conway's Law

Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.

—Melvyn Conway

Ustore

E se formos construir um novo software?

- Canada de acesso aos dados
- Camada de serviços
- Camada Web MVC
- Camada de mensagens
- etc

Inverse Conway Maneuver by Thoughtworks (http://bit.ly/inverse-conway)

Em vez de construir uma arquitetura que corresponde ao seu organograma, construir a arquitetura desejada e reestruturar a organização

Ustore

Capacidades do Time de Negócios

- Como parte da cultura DevOps, devemos desenvolver produtos e não projetos

- “two-pizza teams”

Ustore

A Equipe de Operações

- "Capacidade de desenvolver, implantar e operar capacidades empresariais”

- A equipe de operações gerencia a plataforma de infraestrutura ágil self-service alavancada pelas equipes de negócio

Ustore

Mudanças Técnicas

AssertLab
Advanced Software and Systems Engineering Research Technologies

Centro de Informática 40 anos

Ustore

Decompondo Monolíticos

- As aplicações monolíticas tradicionais de n-camadas raramente funcionam bem quando implantadas em uma infraestrutura de nuvem
  - Acesso a sistemas de arquivos montados e compartilhados
  - Agrupamento (clustering) de servidores de aplicações peer-to-peer
  - Bibliotecas compartilhadas
  - Entre outras…

Ustore

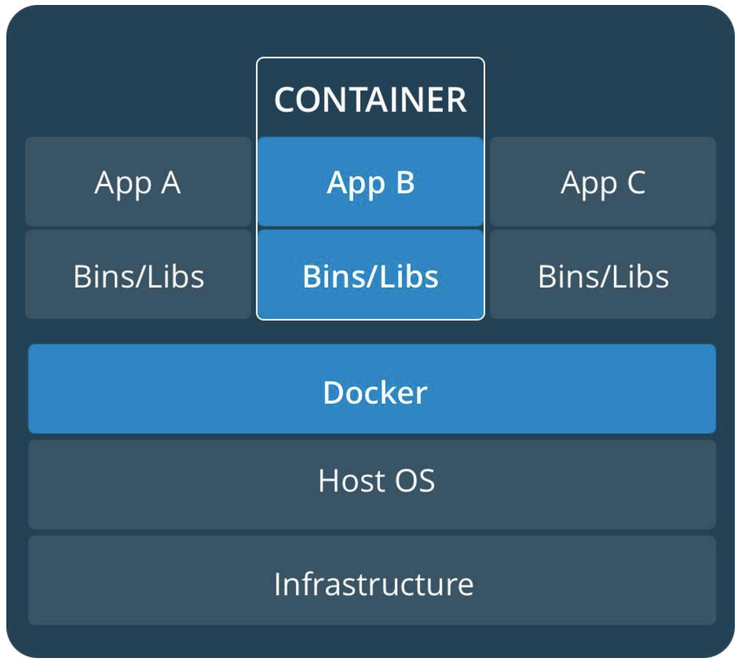
Decompondo Dados

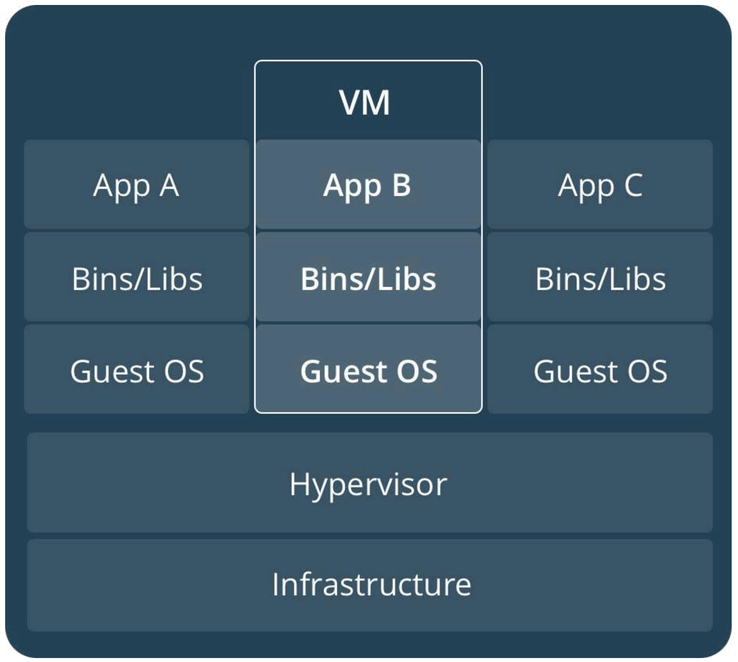
- Toda arquitetura de um produto, deveria começar pela arquitetura da informação
- O sucesso de um produto é em grande parte governado pela qualidade do modelo de domínio (e a linguagem ubíqua que o sustenta). [Domain-Driven Design, by Eric Evans]
  - para ser efetivo, precisa também ser consistente!
- Bounded context
  - Os contextos delimitados ou bounded contexts buscam delimitar o seu domínio complexo em contextos baseados nas intenções do negócio.
  - Isto significa que você deve delimitar as intenções de suas entidades com base no contexto que ela pertence.
    - Padrão de projeto: database per service: cada microsserviço encapsula, governa e protege seu próprio modelo de domínio e armazenamento persistente

Ustore

Containerization

- Uma imagem de um contêiner é um pacote leve, autônomo e executável de um [pedaço de] software que inclui tudo o que é necessário para executá-lo: código, scripts, artefatos, ferramentas do sistema, bibliotecas de sistema, configurações [Docker.com]





AssertLab
Advanced Software and Systems Engineering Research Technologies

Ustore

Da orquestração à coreografia

- Não só a distribuição de serviços, modelagem de dados e governança devem ser descentralizadas, mas também integração de serviços
  - Enterprise Service Bus (ESB): orquestração

- Microsserviços preferem coreografia
  - Ao invés de colocar a inteligência no mecanismo de integração, ela é colocada nos endpoints, como os dutos e os filtros inteligentes da arquitetura Unix

Ustore

Resumindo

- Discutimos algumas das mudanças que a maioria das empresas precisará fazer para adotar arquiteturas de aplicações nativas da nuvem.
- Culturalmente, o tema geral é de descentralização e autonomia
  - DevOps; Integração Contínua; Autonomia; Capacidade dos times de Negócio, Time de Operações…
- Tecnicamente, também descentralização do controle
  - Monolíticos para microsserviços; Bounded Contexts; Containerization; Coreografia

Ustore

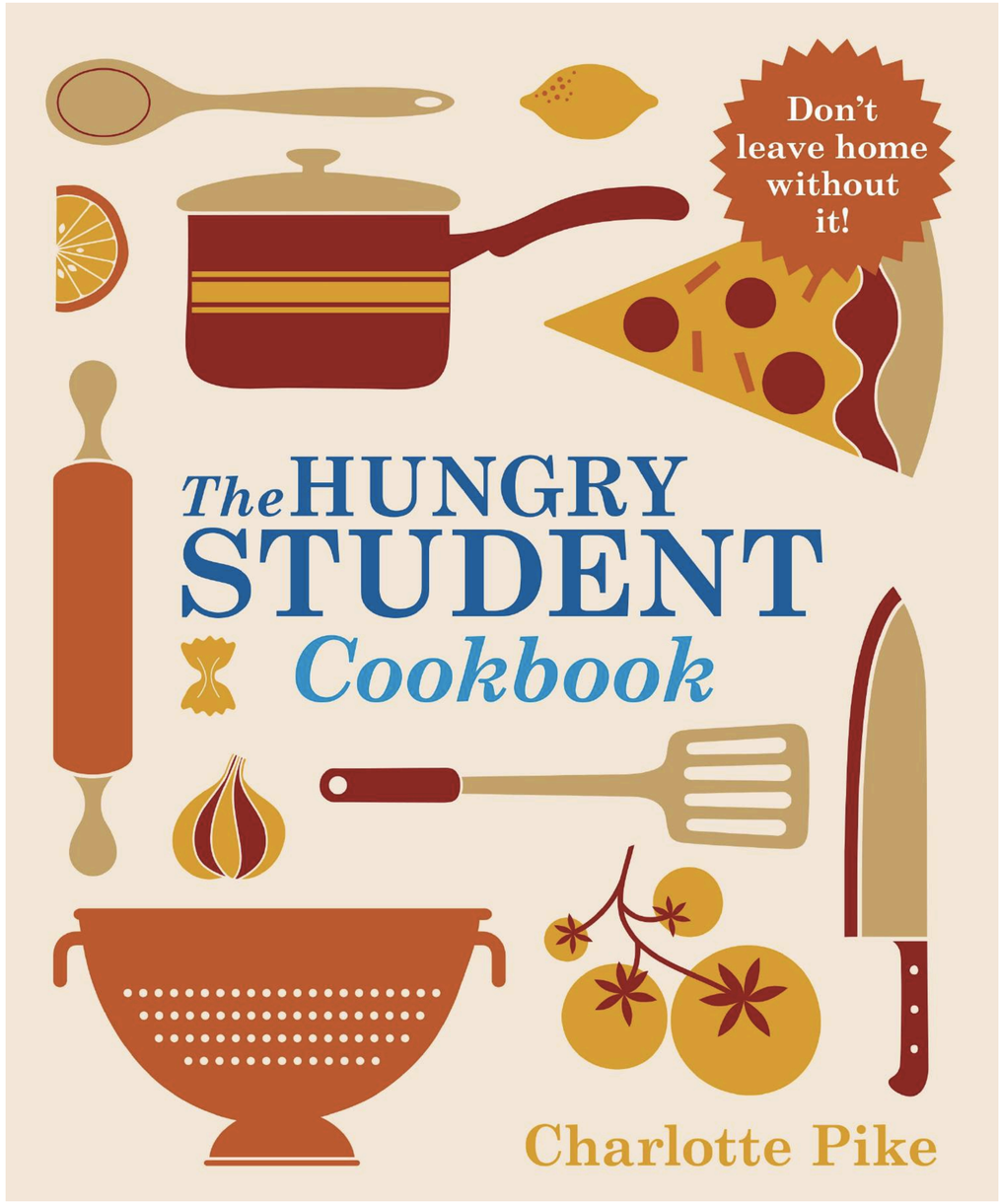
Livro de Receitas da Migração

Migrating to Cloud-Native Application Architectures (O'Reilly)
by Matt Stine

Ustore

Receitas de Decomposição

"Great! How do we get there from here?"



Ustore

First things first

- Building Products at SoundCloud — Part I: Dealing with the Monolith
- Building Products at SoundCloud—Part II: Breaking the Monolith
- How we build microservices at Karma

Novas Características como Microsserviços

- Surpreendentemente, o primeiro passo não é começar a desfazer-se do próprio monólito.
  - Começaremos com a suposição de que você ainda possui um backlog de recursos a serem construídos dentro do monolito.

...the team decided that the best approach to deal with the architecture changes would not be to split the Mothership immediately, but rather to not add anything new to it. All of our new features were built as microservices...

Phil Calcado, SoundCloud

Ustore

A Camada Anti-Corrupção

- Prover a integração de dois sistemas sem permitir que o modelo de domínio de um sistema corrompa o modelo de domínio do outro [Domain-Driven Design (DDD), by Eric Evans]

- Uma forma de criar contratos via API para tratar os sistemas monolíticos como outro microsserviço

Ustore

A Camada Anti-Corrupção

- Facade
  - Simplificar o processo de integração com a interface monolítica
- Adapter
  - Onde são definidos “serviços" que provêem coisas necessárias para os nossos microsserviços
- Translator
  - Converte requisições e respostas entre os microsserviços e a Facade do monolítico

Integração de Sistemas

Tradução de Protocolos

Tradução de Modelos

Ustore

A Camada Anti-Corrupção

- E o link de comunicação?
  - Facade para o sistema, para quando você não tem permissão para acessar ou alterar o sistema legado
  - Adapter para a facade, uma vez que a facade é construída para o monolítico, já permitindo essa comunicação

Ustore

Estrangulando o Monolítico

- A ideia é, gradualmente, ir criando um novo sistema ao redor do antigo, até que finalmente o antigo seja estrangulado [“StranglerApplication”, by Martin Fowler (2004)]
- Dois critérios ajudam na extração de componentes
  1. Identificar contextos delimitados dentro do legado.
  2. Qual dos nossos candidatos extraímos primeiro?
     - Qual microsserviço candidato se beneficiará mais com a velocidade da inovação?

Ustore

Potenciais Estados Finais

- Quando sabemos que terminamos?
  - Quando o legado foi completamente estrangulado até o fim do seu ciclo de vida. Todos os contextos delimitados foram extraídos para microsserviços. Agora é ir eliminando aos poucos as camadas anti-corrupção
  - O legado foi estrangulado até um ponto onde o custo para extrair mais serviços excede o retorno no esforço necessário de desenvolvimento

Ustore

Receitas de Sistemas Distribuídos

Ustore

Requisitos não-funcionais

- Em SD encontramos requisitos que não existem no contexto dos sistemas monolíticos
- Alguns são resolvidos pelas leis da física como, por exemplo, latência, consistência particionamento de redes
- Outros devemos aplicar o uso de padrões de projeto específicos na aplicação
- Tomemos como exemplo os casos dos projetos Spring Cloud e Netflix OSS

Ustore

Configuração Versionada e Distribuída

- Proper configuration management (Twelve-Factor Applications) com adições para sistemas em larga escala
  - Alterar os níveis de log de um aplicativo em execução para depurar um problema de produção
  - Alterar o número de threads que recebem mensagens de um broker de mensagens
  - Relatar todas as alterações de configuração feitas em um sistema em produção para suportar auditorias regulatórias
  - Ativar / desativar funções em um aplicativo em execução
  - Proteja segredos (como senhas) embutidos na configuração

Versionamento

Auditabilidade

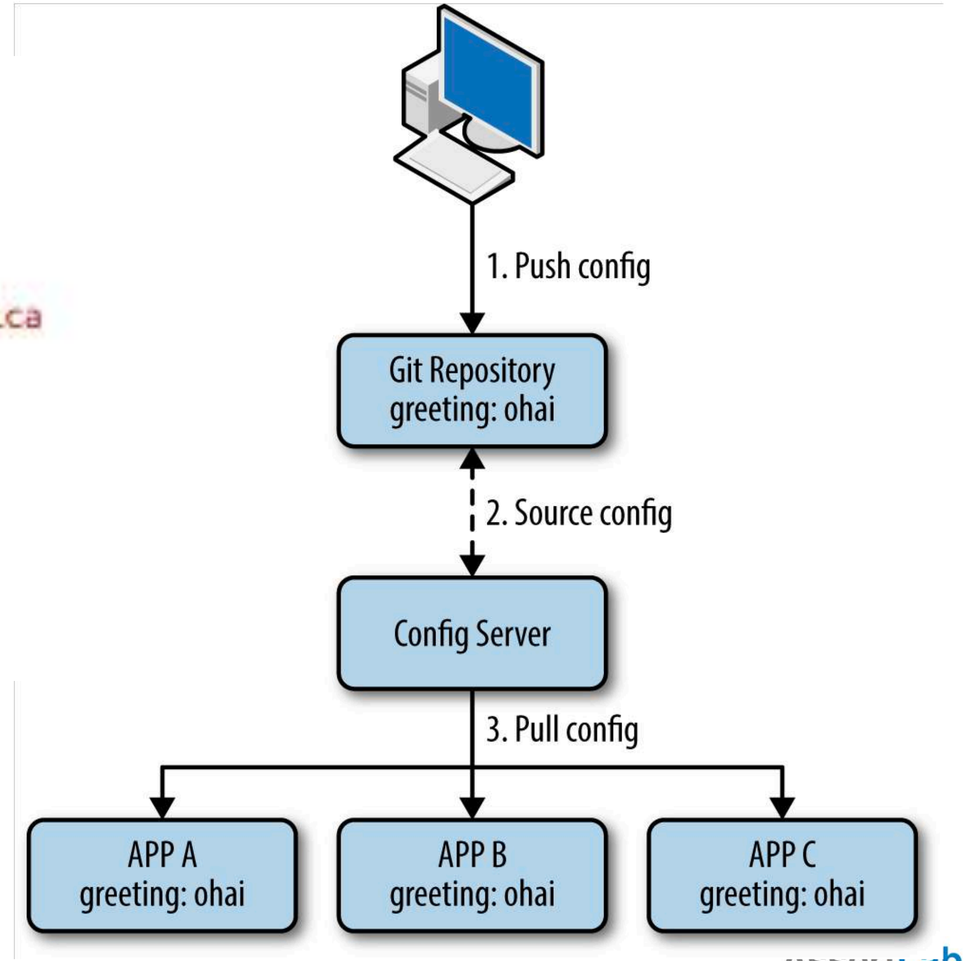
Criptografia

Atualizar sem reinicia

Spring Cloud Config Server

```
{
    "label": "",
    "name": "default",
    "propertySources": [
      {
          "name": "https://github.com/mstine/config-repo.git/applica
tion.yml",
        "source": {
          "greeting": "ohai"
        }
      }
    ]
}
```

```
"configService:https://github.com/mstine/config-repo.git/applica
tion.yml": {
  "greeting": "ohai"
},
```

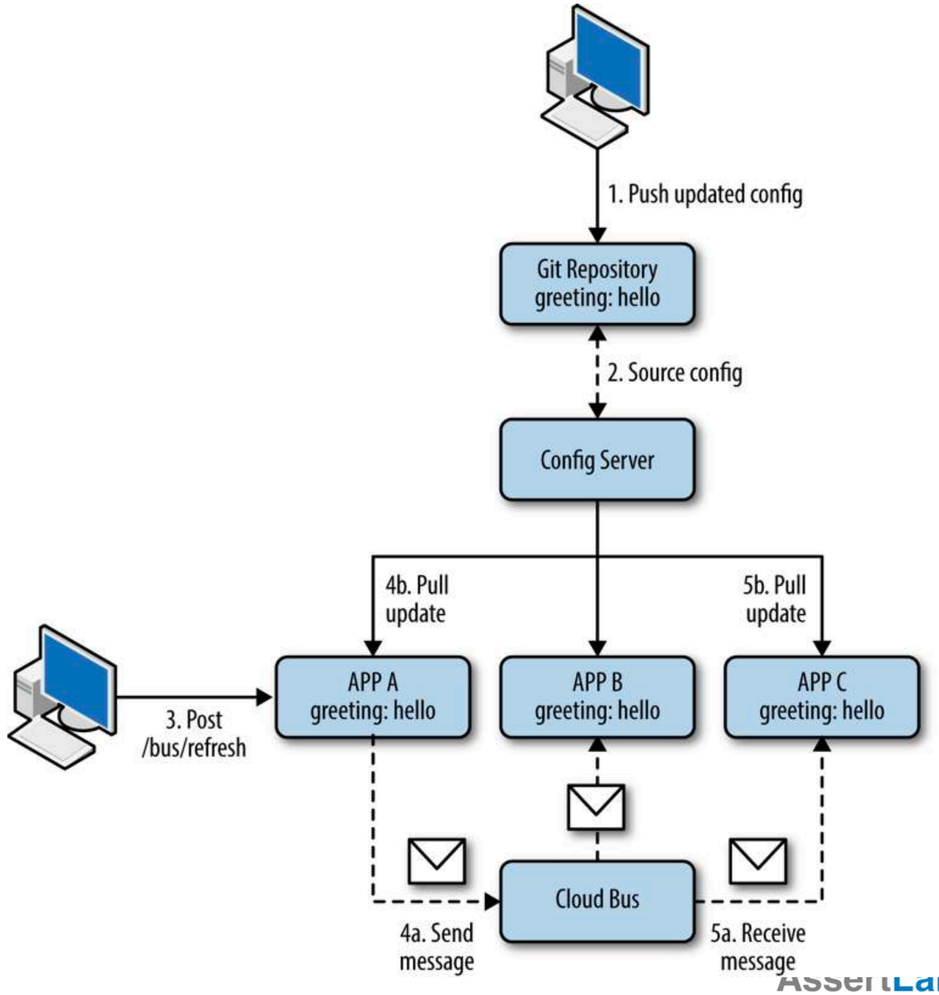


Ustore

Spring Cloud Bus

- Este projeto liga nós de um sistema distribuído com um broker leve de mensagens, que pode então ser usado para transmitir mudanças de estado

- HTTP POST to the /bus/refresh endpoint

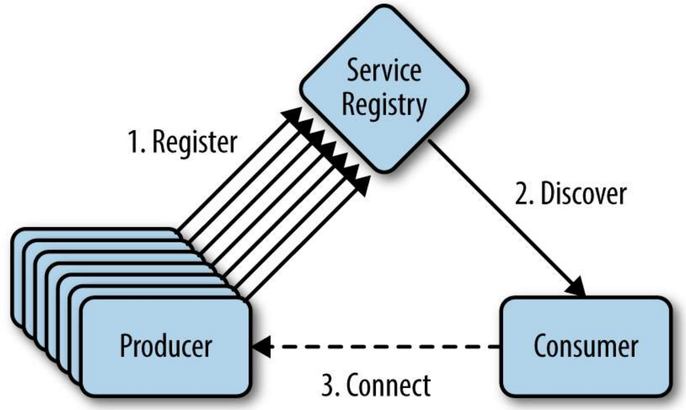


AssertLab
Advanced Software and Systems Engineering Research Technologies

Ustore

Registro e Descoberta de Serviços

- Como realizamos a conexão necessária para permitir que todos os microsserviços dentro de um sistema se comuniquem uns com os outros?
- Um padrão de arquitetura comum na nuvem é ter serviços frontend (application) e backend (business).
- Resolvemos esse problema anteriormente usando os padrões de Service Locator e Dependency Injection, e as arquiteturas orientadas a serviços usaram várias outras formas de registros de serviços.
- O Eureka (projeto do Netflix OSS) é uma implementação de solução para este problema, cujo consumo desse serviço é simplificado pelo Spring Cloud Netflix




```
@SpringBootApplication
@EnableDiscoveryClient
public class Application {

  public static void main(String[] args) {
    SpringApplication.run(Application.class, args);
  }

}
```

Ustore

Aplicação Spring Boot

- A aplicação consegue se comunicar com seus clientes através do uso do DiscoveryClient
- A aplicação procura uma instância de um serviço registrado com o nome de PRODUCER, obtém sua URL e, em seguida, aproveita o RestTemplate do Spring para se comunicar com ele.

```
@Autowired
DiscoveryClient discoveryClient; ❶

@RequestMapping("/")
public String consume() {
    InstanceInfo instance = discoveryClient.getNextServerFromEur
eka("PRODUCER", false); ❷

  RestTemplate restTemplate = new RestTemplate();
       ProducerResponse response = restTemplate.getForOb
ject(instance.getHomePageUrl(), ProducerResponse.class);

  return "{\"value\": \"" + response.getValue() + "\"}";
}
```

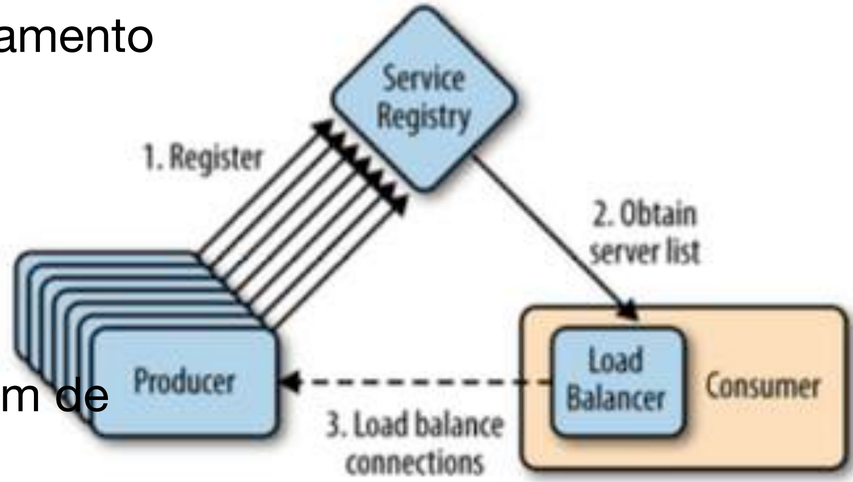
AssertLab
Advanced Software and Systems Engineering Research Technologies

Centro de Informática 40 anos

Ustore

Roteamento e Balanceador de Carga

- Round-robin básico é efetivo para muitos cenários, mas em se tratando de sistemas na nuvem, precisamos de soluções mais avançadas de roteamento e balanceamento
- Ribbon (Netflix OSS) é outra implementação de solução
  - Várias regras de balanceamento de carga incorporadas:
    - Round-robin, Tempo de resposta médio ponderado, Aleatória,
    - Disponibilidade filtrada (evitar circuitos trocados ou alta contagem de conexões)
  - Sistema personalizado de regras de balanceamento de carga
  - Integração plugável com soluções de descoberta de serviços (incluindo Eureka)
  - Inteligência nativa da nuvem, como afinidade de zona e evitar zona não saudável (unhealthy)
  - Resiliência de falha interna



Ustore

Aplicação Spring Boot

- O LoadBalancerClient ativado é injetado pelo Spring.

- O método choose fornece a localização de uma instância do serviço usando o algoritmo de balanceamento de carga ativado

```
@Autowired
LoadBalancerClient loadBalancer; ❶

@RequestMapping("/")
public String consume() {
  ServiceInstance instance = loadBalancer.choose("producer"); ❷
      URI    producerUri    =    URI.create("http://${instance.host}:$
{instance.port}");

  RestTemplate restTemplate = new RestTemplate();
   ProducerResponse  response  =  restTemplate.getForObject(producer
Uri, ProducerResponse.class);

  return "{\"value\": \"" + response.getValue() + "\"}";
}
```

AssertLab
Advanced Software and Systems Engineering Research Technologies

Ustore

Spring Cloud Netflix + Ribbon

- RestTemplate é injetado ao invés do LoadBalancerClient.
- O RestTemplate injetado automaticamente resolve http://producer para a URI da instância atual do serviço.

```
@Autowired
RestTemplate restTemplate; ❶

@RequestMapping("/")
public String consume() {
    ProducerResponse response = restTemplate.getForObject("http://
producer", ProducerResponse.class); ❷
  return "{\"value\": \"" + response.getValue() + "\"}";
}
```

Ustore

Tolerância a Falhas

- Os sistemas distribuídos possuem mais potenciais modos de falha do que monolíticos.
- Cada solicitação recebida pode potencialmente chamar dezenas (ou mesmo centenas) de diferentes microsserviços, algumas falhas em uma ou mais dessas dependências são praticamente garantidas.

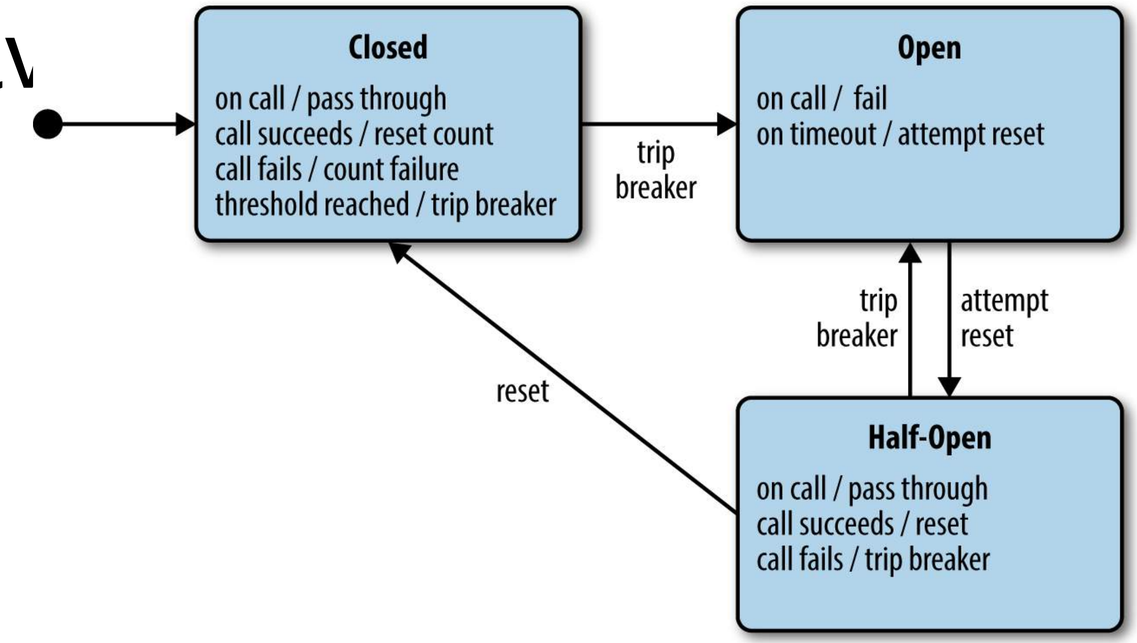
Without taking steps to ensure fault tolerance, 30 dependencies each with 99.99% uptime would result in 2+ hours downtime/month (99.99%^30^ = 99.7% uptime = 2+ hours in a month).

—Ben Christensen, Netflix Engineer

Ustore

Tolerância a Falhas

- Circuit breakers isolam um serviço de suas dependências evitando chamadas remotas quando uma dependência é determinada como não saudáv



AssertLab
Advanced Software and Systems Engineering Research Technologies

Ustore

Tolerância a Falhas

- Bulkheads particionam um serviço para limitar erros e faltas e evitar que o serviço inteiro falhe devido a falhas em uma área.
  - Microsserviços e contêineres são exemplos práticos

Ustore

Hystrix

- Hystrix (Netflix OSS) permite que o código seja envolvido (wrapped) em objetos HystrixCommand para que esse código seja quebrad

```
public class CommandHelloWorld extends HystrixCommand<String> {

    private final String name;

    public CommandHelloWorld(String name) {
                super(HystrixCommandGroupKey.Factory.asKey("Exam
pleGroup"));
        this.name = name;
    }

    @Override
    protected String run() { ❶
        return "Hello " + name + "!";
    }
}
```

AssertLab
Advanced Software and Systems Engineering Research Technologies

Ustore

Spring Boot e Hystrix

- Spring Cloud Netflix adiciona a anotação @EnableCircuitBreaker para habilitar o componente de runtime Hystrix em uma aplicação Spring Boot
- O método anotado com @HystrixCommand é envolvido em um circuit breaker
- O método [illegible] lo dentro da anotação [illegible] etorno enquanto [illegible] semi-aberto.

```
@Autowired
RestTemplate restTemplate;

@HystrixCommand(fallbackMethod = "getProducerFallback") ❶
public ProducerResponse getProducerResponse() {
   return  restTemplate.getForObject("http://producer",  ProducerRes
ponse.class);
}

public ProducerResponse getProducerFallback() { ❷
  return new ProducerResponse(42);
}
```

Ustore

Hystrix

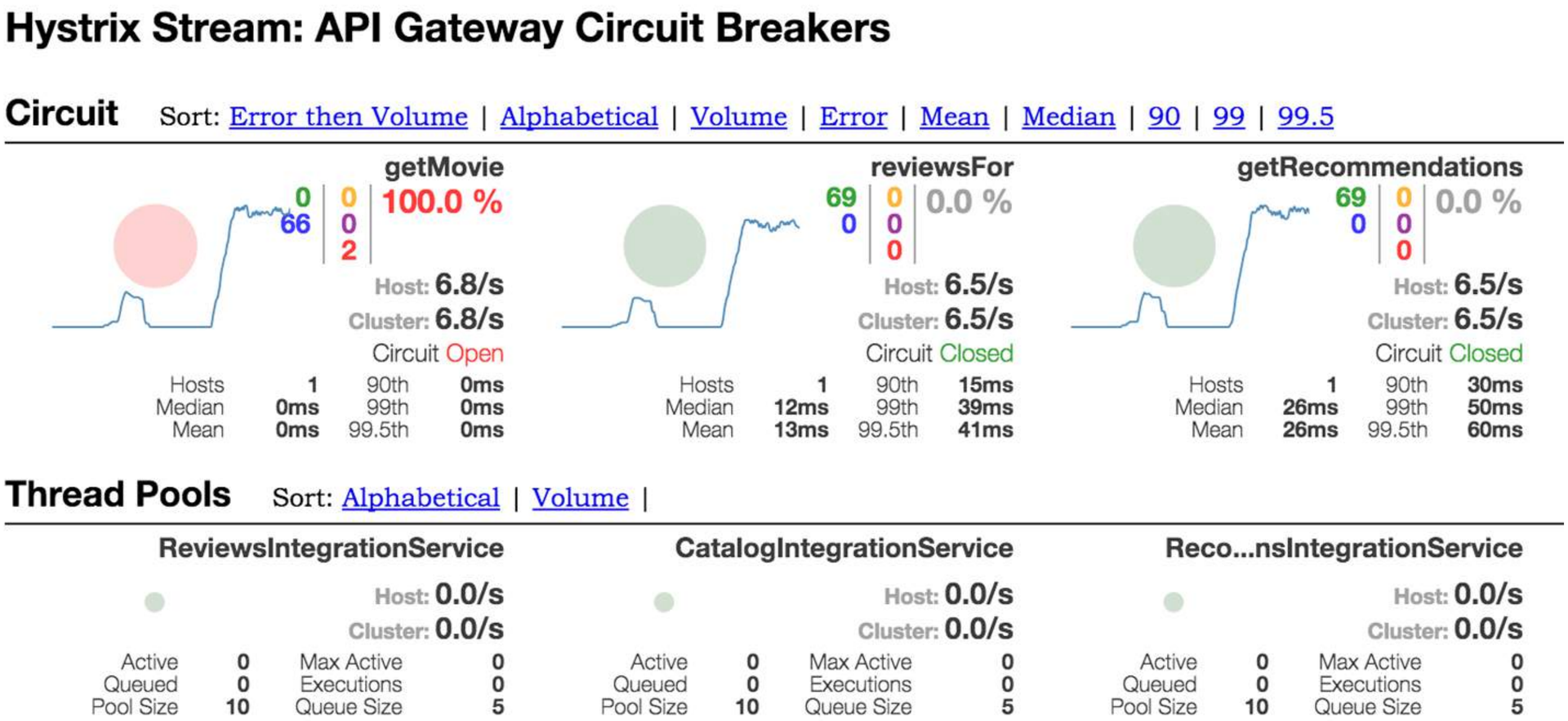
- Hystrix é única dentre as muitas outras implementações de circuit breaker, pois emprega bulkheads operando cada circuit breaker dentro de seu próprio grupo de threads.

- Ele também coleta muitas métricas úteis sobre o estado do circuit breaker, incluindo:
  - Volume de tráfego; Taxa de solicitação; Percentagem de erro; Relatório de hospedagem; Percentis de latência; Sucessos, falhas e rejeições

- Essas métricas são emitidas como um fluxo de eventos que pode ser agregado por outro projeto Netflix OSS chamado Turbine.

Ustore

Hystrix Dashboard

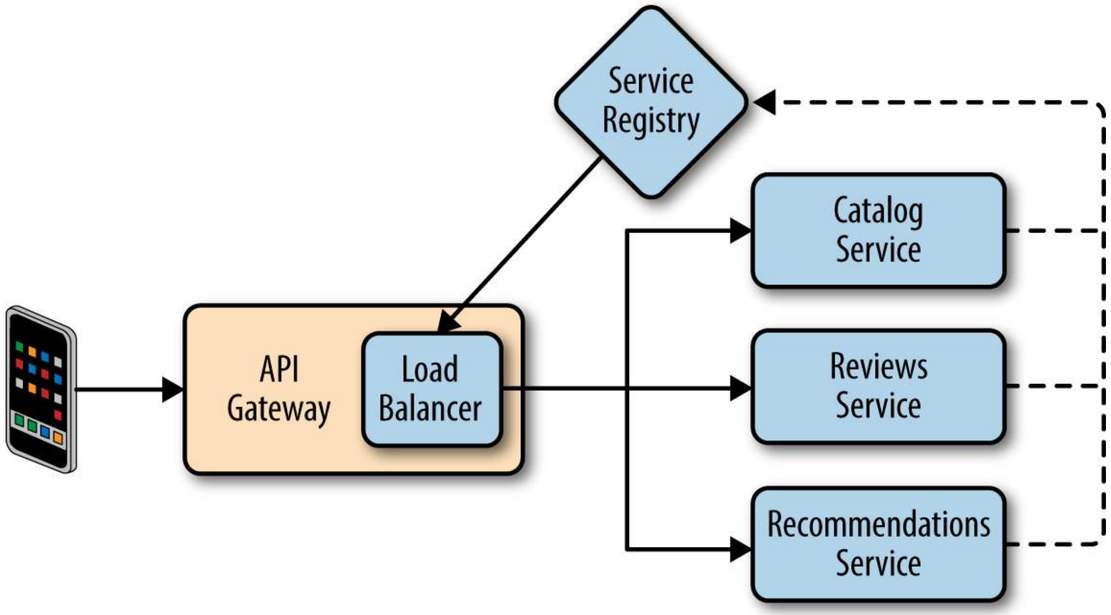


AssertLab
Advanced Software and Systems Engineering Research Technologies

Ustore

API Gateways/Edge Services

- O padrão API Gateway é direcionado para mudar a carga dos requisitos do desenvolvedor do dispositivo para o lado do servidor.
- API Gateways são simplesmente uma classe especial de microsserviços que atendem as necessidades de um aplicativo de cliente e fornecem um único ponto de entrada para o backend.



Ustore

RxJava uma implementação de Reactive Extensions

- RxJava nasceu no Netflix OSS, é uma biblioteca para compor programas assíncronos e baseados em eventos usando sequências observáveis para a JVM
- Exemplo: um site como Netflix com serviços de catálogo, de revisões e recomendações


```
{
    "mlId": "1",
    "recommendations": [
        {
            "mlId": "2",
            "title": "GoldenEye (1995)"
        }
    ],
    "reviews": [
        {
            "mlId": "1",
            "rating": 5,
            "review": "Great movie!",
            "title": "Toy Story (1995)",
            "userName": "mstine"
        }
    ],
    "title": "Toy Story (1995)"
}
```

Ustore

RxJava

- RxJava utiliza o método Observable para acesso simultâneo a cada um dos serviços.
- Depois de receber as três respostas, o código as passa para o Lambda do Java 8, que as usa para criar uma instância de MovieDetails.
- Estas instâncias de MovieDetails podem então ser serializadas para produzir a resposta

```
Observable<MovieDetails> details = Observable.zip(

  catalogIntegrationService.getMovie(mlId),
  reviewsIntegrationService.reviewsFor(mlId),
  recommendationsIntegrationService.getRecommendations(mlId),

  (movie, reviews, recommendations) -> {
    MovieDetails movieDetails = new MovieDetails();
    movieDetails.setMlId(movie.getMlId());
    movieDetails.setTitle(movie.getTitle());
    movieDetails.setReviews(reviews);
    movieDetails.setRecommendations(recommendations);
    return movieDetails;
  }
);
```

AssertLab
Advanced Software and Systems Engineering Research Technologies

Ustore

Resumindo

- Receitas de Decomposição
  - Construindo as novas características como microsserviços
  - Integrando novos microsserviços com o legado através de camadas anticorrupção
  - Estrangulando o legado a partir da identificação de contextos delimitados e extraindo serviços.

Ustore

Resumindo

- Receitas de Sistemas Distribuídos
  - Configuração versionadas, distribuídoas e atualizadas através de um servidor de configuração e barramento de gerenciamento.
  - Descobrimento dinâmico de dependências remotas.
  - Decentralizar decisões de balanceamento de carga.
  - Prevenção de falhas em cascata através de circuit breakers e bulkheads.
  - Integração de clientes específicos via API Gateways.

Ustore

Resumindo

- Existem tantos outros padrões para auxiliar nestas tarefas
- Uma boa fonte e sugestão de leitura são
  - "Testing Strategies in a Microservice Architecture" by Toby Clemson
  - Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation by Jez Humble and David Farley (Addison- Wesley).

Ustore

Referências

- DevOps: A Software Architect's Perspective (SEI Series in Software Engineering), by Len Bass, Ingo Weber , Liming Zhu
- Building Microservices: Designing Fine-Grained Systems, by Sam Newman
- The evolution of DevOps, By Mike LoukidesAugust 31, 2017
- Continous Delivery (https://www.continuousdelivery.com/)
- Continuous Integration: Improving Software Quality and Reducing Risk, by Paul M. Duvall, Steve Matyas, Andrew Glover
- Designing Data-Intensive Applications (http://www.dataintensive.net/)
- Systems Performance: Enterprise and the Cloud, by Brendan Gregg
- The Practice of Cloud System Administration
- Where to Start with DevOps Series, October 17, 2016 by Gene Kim

AssertLab
Advanced Software and Systems Engineering Research Technologies

Centro de Informática 40 anos