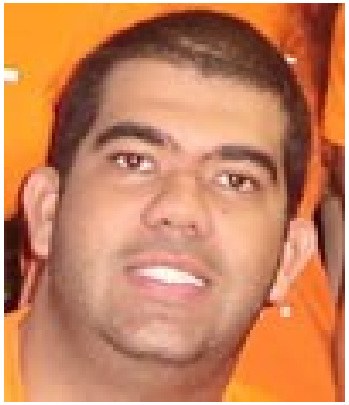


THE DEVELOPER'S CONFERENCE

"O combinado não sai caro: escrevendo testes de contrato consumer driven em Pact + Node"

Marcelo Bezerra (linkedin.com/in/bovino/)

Especialista em APIs – LuizaLabs SP – Tribo Engenharia APIs

Agenda

THE DEVELOPER'S CONFERENCE

- O que (não) é um contrato e um teste de contrato
- Porque precisamos de testes de contrato
- Consumer driven x Producer driven
- Dando nome aos bois: Pact, Pact Specification + ferramentas (libs pact, PactBroker / PactFlow)
- Considerações finais
- *"Demo"*
- *Quer saber mais sobre isso?*

Gauro é a maior espécie de gado selvagem do mundo

O que é um contrato

THE DEVELOPER'S CONFERENCE

- *Contrato = Compromisso entre as partes (cliente e fornecedor)*
- *Um registro "formal" (um documento) de **expectativas** das partes*
- *Declara quais **interações** estão previstas*
- ***acessível** por ambas as partes*
- *Violação = **prejuízo** para uma das partes*

- *Um contrato é uma forma de **expressar necessidades dos clientes** e **estabelecer comunicação** :-)*

O que é um teste de contrato

THE DEVELOPER'S CONFERENCE

- *verifica se uma API (provider) atendeu as **expectativas***
- *verifica se a API atende **bem** todas as **interações** necessárias para os clientes*
- *utiliza **técnicas e ferramentas de mocking** (server / stub runner como o Wiremock) para testar integração e não conectividade*
  → Não é um teste E2E!!!
- *Uma etapa nas pipelines de testes para permitir detecção de problemas o quanto antes (encontre um erro antes de commitar)*
  - *Violei um contrato! E agora?*
    *Interrompe a build? **Depende ("Pending Pacts" + "WIP Pacts")***

O que é um teste de contrato

THE DEVELOPER'S CONFERENCE

- **Benefícios que você pode esperar:**
  - *mais velocidade de desenvolvimento*
  - *segurança de que não irá quebrar seus consumers*
  - *encontrar erros mais rapidamente (antes de commitar)*
  - *Rodar testes sem precisar ter o provider rodando (teste de contrato sobe um mock server local na hora de rodar os testes) → Não é E2E*
  - *Ferramentas amigáveis e integráveis ao seu pipeline CI/CD (web ui, webhook, API, CLI, libs)*
  - *Open source (e multi-linguagem no caso do Pact)*
  - *Boa documentação, exemplos e suporte na comunidade (**dependendo da linguagem e versão de Pact specification adotada….)***

O que NÃO é um teste de contrato

THE DEVELOPER'S CONFERENCE

- ***Não** é um teste que requer que o API provider esteja rodando ou acessível para poder ser utilizado pelo consumer (não é um teste fim a fim)*
- ***Não** é um substituto para uma spec OpenAPI / Swagger*
- ***Não** é uma idéia nova (2006) e não é exclusivo para uso com micro serviços*
- ***Não** necessariamente é o fim dos testes de integração e E2E*
- ***Não é um teste que** antecipa todos os possíveis problemas*
- ***Não** é uma bala de prata*

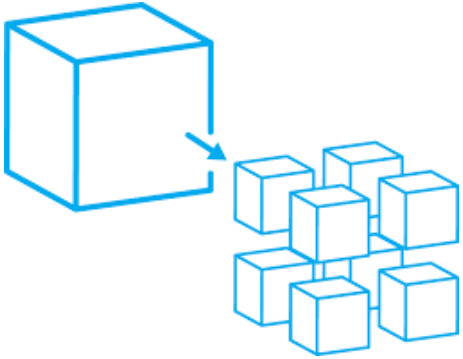
Porque precisamos de teste de contrato

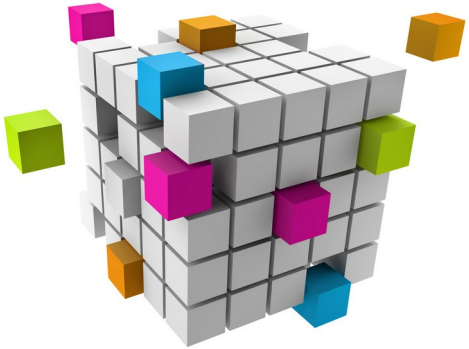
THE DEVELOPER'S CONFERENCE

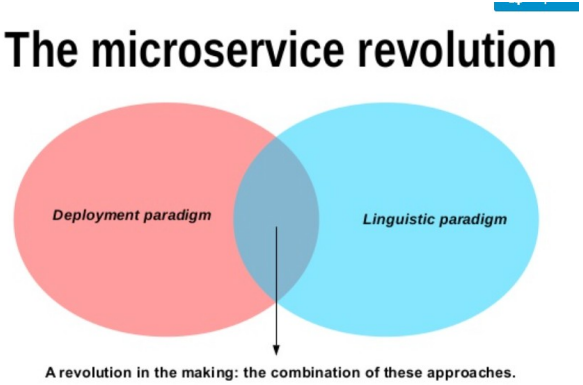
*Quebramos **(e xingamos)** o monolito!!! YAYYYY*

→ *Vision of a microservice revolution - https://bit.ly/3z6hksU*

→ *Are microservices evolutionary or revolutionary? https://bit.ly/3cnypVk*





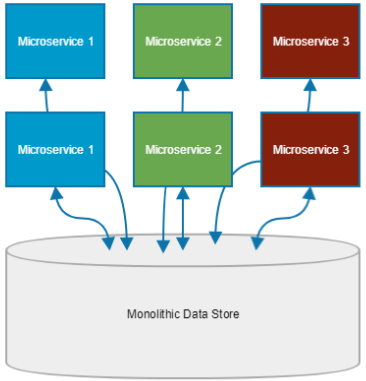


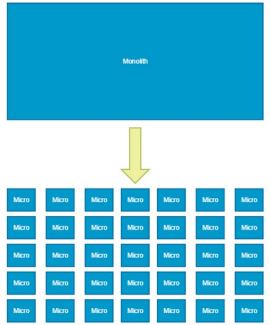
Porque precisamos de teste de contrato

THE DEVELOPER'S CONFERENCE

YAYY.. MAS..... Microservice ***TRETAS*** *e* ***ANTIPATTERNS*** *(trace, log, observabilidade, deploy, cloud specific tretas, teste, escalabilidade, latência, DB/data store monolítica, muitos microserviços, multiplas linguagens e frameworks na empresa,* ***integration hell,*** *comunicação entre os times…*

Enfim, a lista de desafios numa arquitetura [distribuída] de microserviços é extensa...

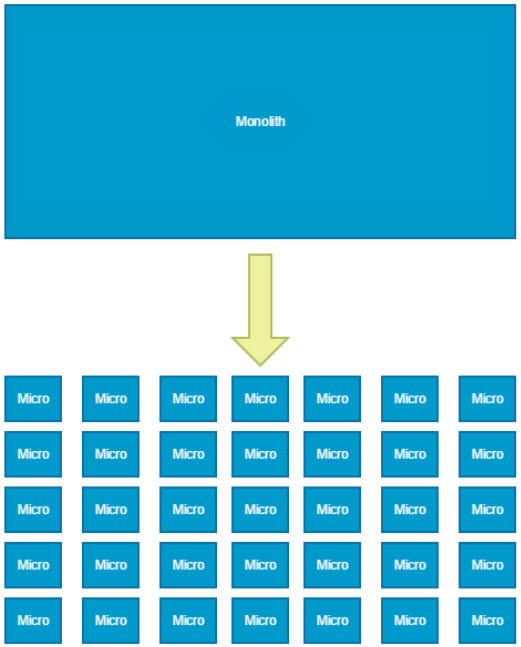


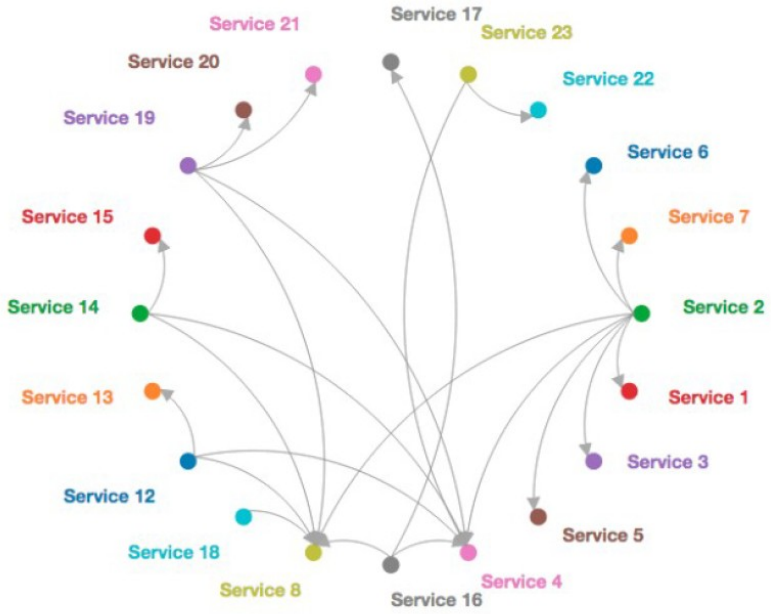


Porque precisamos de teste de contrato

THE DEVELOPER'S CONFERENCE

Microservices integration hell

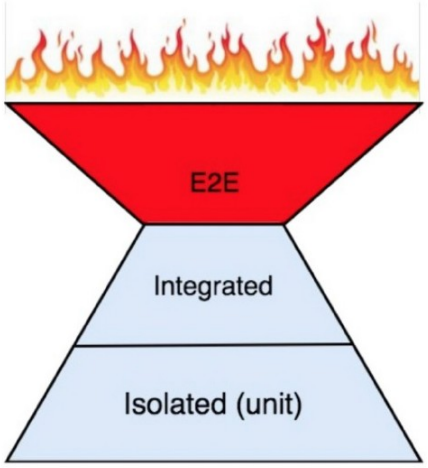


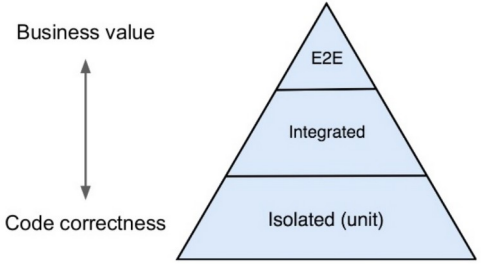


Porque precisamos de teste de contrato

THE DEVELOPER'S CONFERENCE

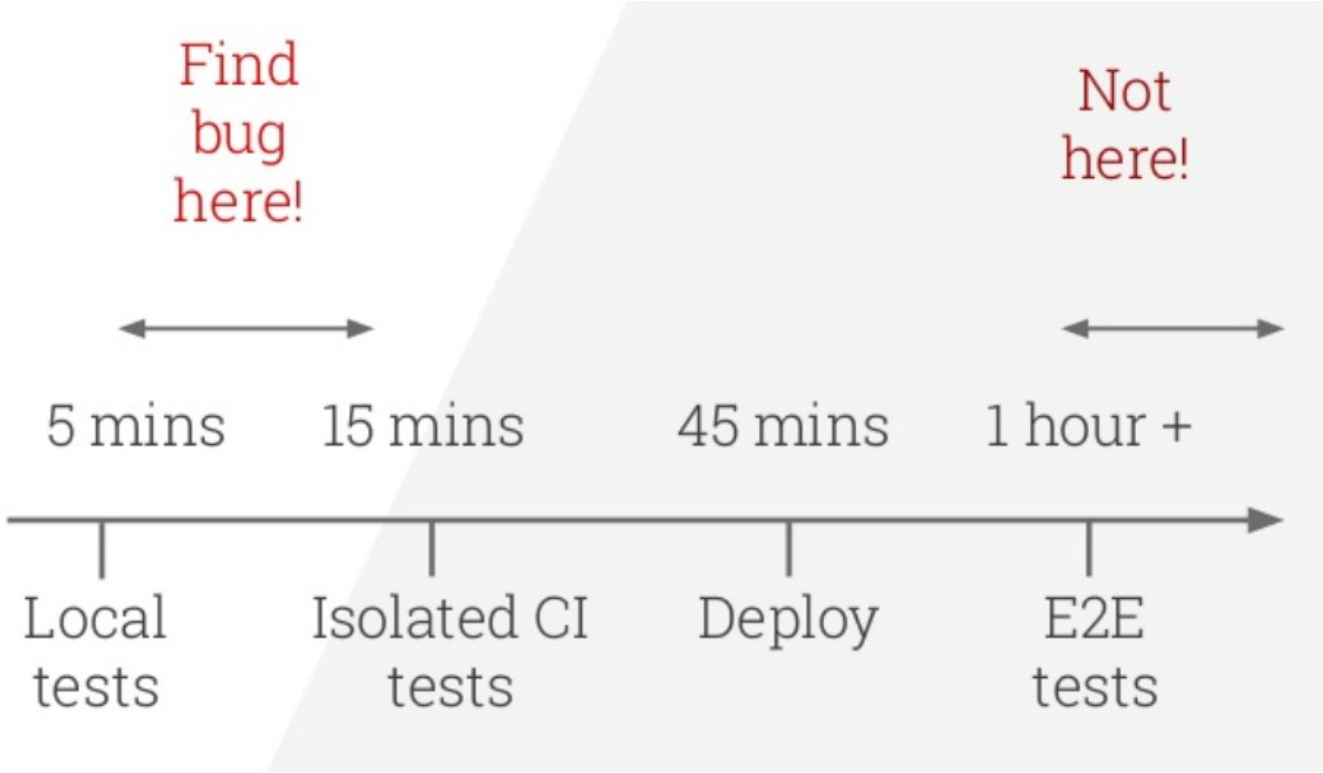
- *Qualidade requer testes… mas quais testes?*
- *Qual o problema dos testes E2E e integrados?*





	E2E Tests	Mock Tests	CDC Tests
Isolation	−	+	+
Complexity	−	+	+
Test Data Setup	−	+	+
Testing Data Semantics	+	−	−
Feedback Time	−	+	+
Stability	−	+	+
Reveal Unused Interfaces	−	−	+
Well-Fittedness	−	−	+
Unknown Consumers	+	+	−

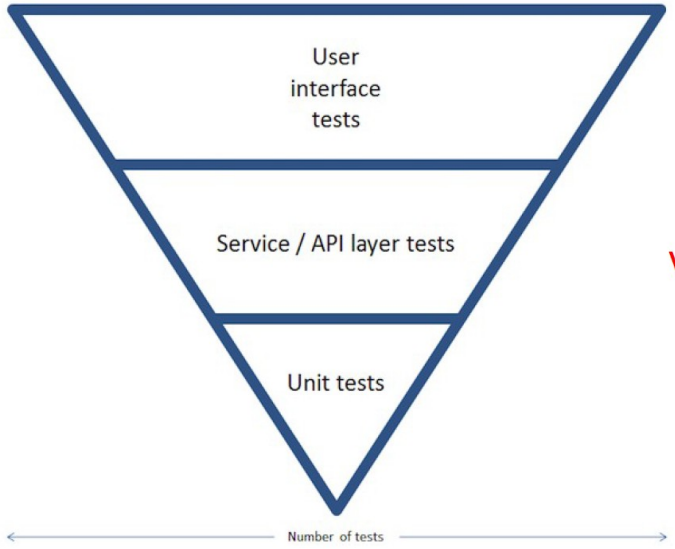
Porque precisamos de teste de contrato



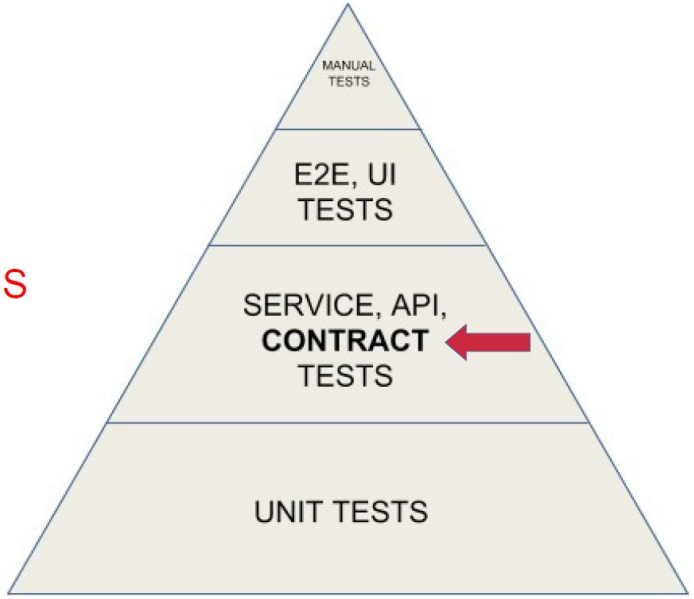
Porque precisamos

THE DEVELOPER'S CONFERENCE

- *Qualidade requer testes… mas quais testes?*



VERSUS



Consumer driven x Producer driven

THE DEVELOPER'S CONFERENCE

- ***Consumer driven:*** ***Cada consumer*** *escreve e disponibiliza o contrato (um contrato por consumer)*
- *Provider consome os contratos e os usa em seus testes de verificação (usando os contratos como insumo para fazer requests na API e ver se está tudo ok)*
- *Ferramentas → PACT + lib pact da sua linguagem + Pact Broker / Pact Flow*
  - *Sem "Pending Pacts", quando o provider viola algum contrato os testes dele irão quebrar e ele saberá de imediato qual consumer foi impactado.*
  - *Útil quando os consumers são conhecidos… mas e se forem MUITOS?*
  - *Escolha bem o que colocar no contrato!*
    Exemplo: *preciso* ***MESMO*** *colocar no contrato 100% dos status code de retorno possíveis?*

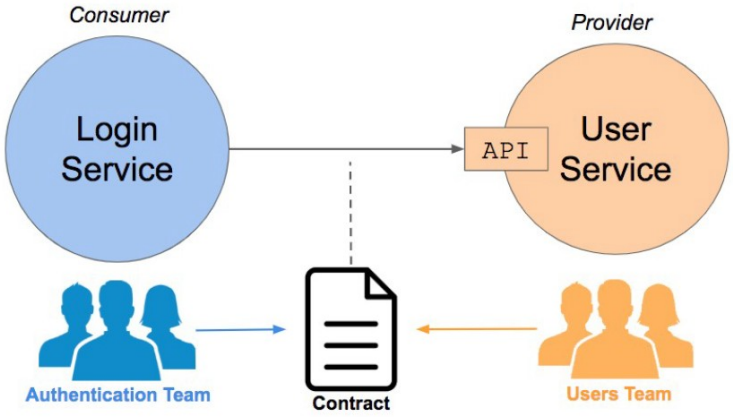
    devo quebrar o teste no provider se um campo novo for adicionado ao response?
  - *Geralmente "não dá" para aplicar em APIs públicas exceto se o provider da API fornecer também algum tipo de implementação de client de referência, SDK ou consumer release*
  - *O projeto Spring Cloud Contract iniciou mais orientado a abordagem producer driven, porém depois passou a suportar bem ambas as abordagens*

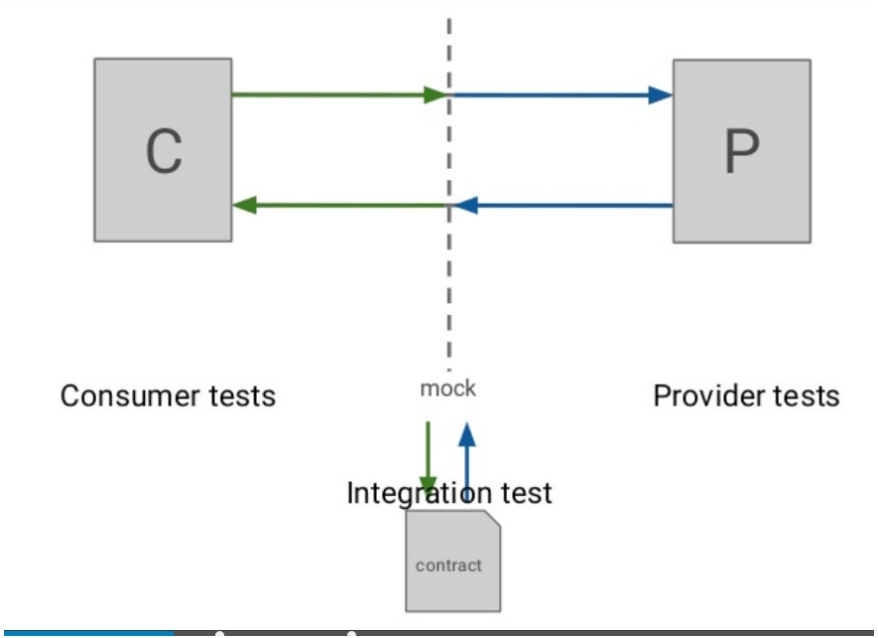
Consumer driven x Producer driven

THE DEVELOPER'S CONFERENCE

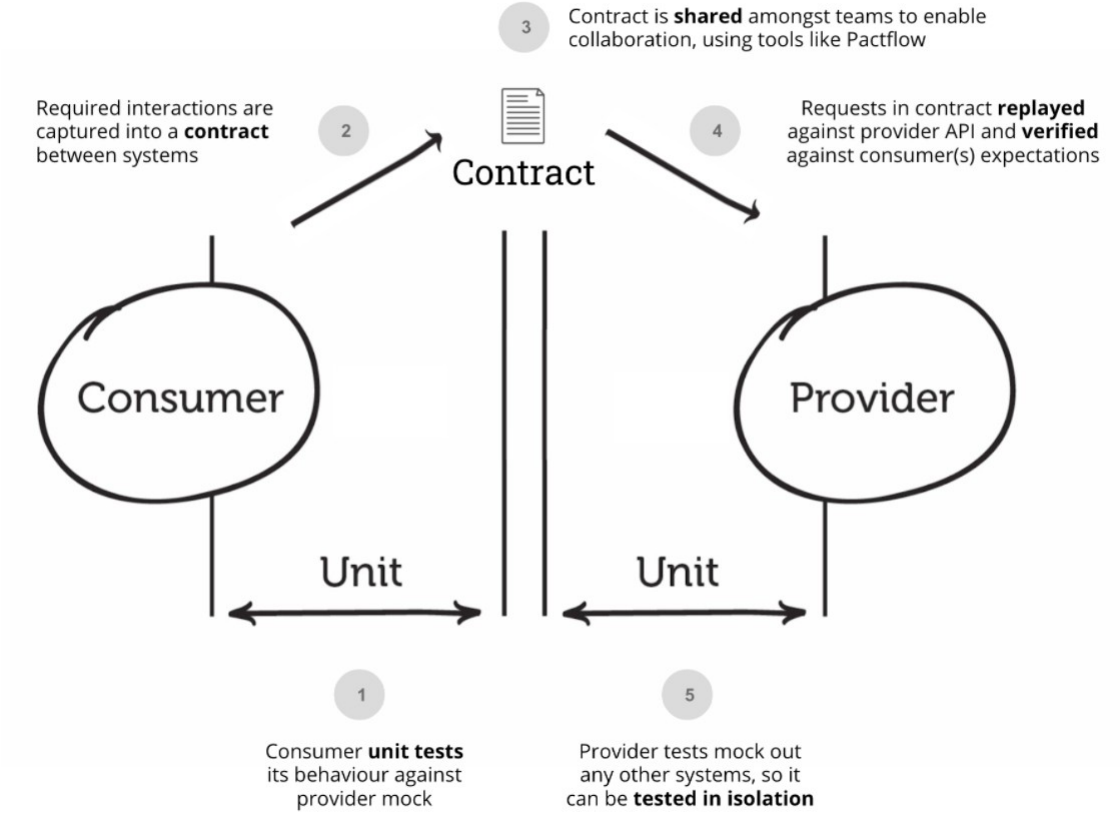
- ***Producer driven:*** *Provider é quem escreve e disponibiliza os contratos*
- *Ferramentas → Spring Cloud Contract + repositório Maven/Gradle (ou PactFlow) + Spring Cloud Stub Runner + Spring Cloud Contract Verifier*
- *Quando usar:*
  - *consumers são muitos*
  - *consumers estão fora da empresa (porém são bem definidos e conhecidos)*
  - *provider terá sua implementação iniciada antes dos consumers*
  - *não necessariamente útil para API públicas, não vai fechar o loop de feedback: Se o provider mudar o contrato, o cliente vai quebrar e o provider não saberá (até chegar um e-mail ou ligação raivosa de um consumer irritado e xingando muito no Twitter)*
  - ***The curious case for the Provider Driven Contract:*** https://pactflow.io/blog/the-curious-case-for-the-provider-driven-contract/

Consumer driven





Consumer driven com Pact

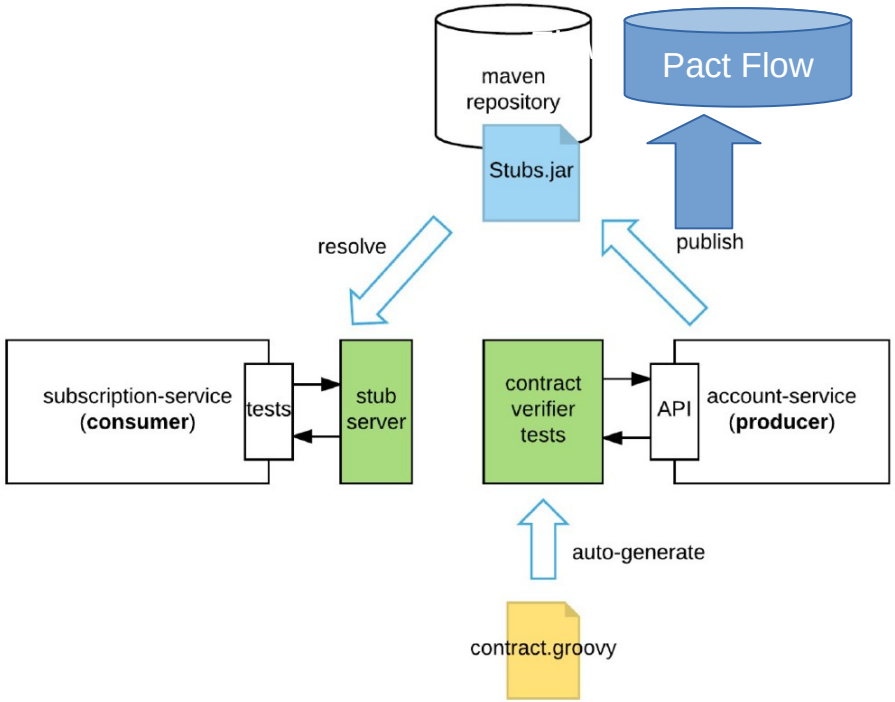


THE DEVELOPER'S CONFERENCE

Pact

Producer driven – Exemplo usando SpringCloud Contract

THE DEVELOPER'S CONFERENCE



Pact

THE DEVELOPER'S CONFERENCE

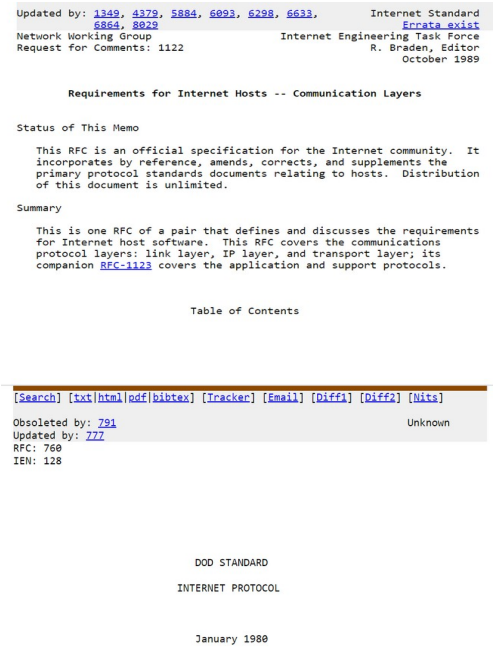
- Ferramenta REALMENTE multi-linguagem para testes de contrato consumer driven (ou producer driven com algum contorcionismo)
  - JVM (Java / Kotlin / Groovy / Swift / Scala), Ruby , Golang, **NodeJS / JavaScript**, Python, Objective-C / Swift, .NET, PHP, C++ (consumer only), Rust
- Pact specification: viabiliza que as implementações sejam compatíveis multi-linguagem: https://github.com/pact-foundation/pact-specification/
- Pact specifications existentes: v1, v2, v3 e v4
- Para usar com Node/JavaScript prefira por enquanto spec v2 (v3 em progresso no PactJS)
- Para usar com Java serve v2 ou v3
- Permite ao consumer publicar contratos num broker (PactBroker ou PactFlow) e ao provider registrar o resultado dos testes de verificação
- Possui API, webhook e CLI que podem e devem ser integradas ao seu pipeline CI/CD

Filosofia do Pact Specification (Lei de Postel)

Pact

THE DEVELOPER'S CONFERENCE

- Lei de Postel na RFC 760 (1980) "an implementation should be conservative in its sending behavior, and liberal in its receiving behavior"
- Lei de Postel reeditada na RFC 1122 (1989) "Be liberal in what you accept, and conservative in what you send"
- Seja o mais rigoroso possível com o que enviamos (requests)
- Seja o mais frouxo que pudermos com o que aceitamos (response)
- Valide o que precisa estar no requests e responses
- **NÃO** valide o que "não" deveria estar
  Exemplo: testar se o meu header NÃO contém uma certa chave



Updated by: 1349, 4379, 5884, 6093, 6298, 6633, 6864, 8029 — Internet Standard — Errata exist

Network Working Group — Internet Engineering Task Force

Request for Comments: 1122 — R. Braden, Editor — October 1989

Requirements for Internet Hosts -- Communication Layers

Status of This Memo

This RFC is an official specification for the Internet community. It incorporates by reference, amends, corrects, and supplements the primary protocol standards documents relating to hosts. Distribution of this document is unlimited.

Summary

This is one RFC of a pair that defines and discusses the requirements for Internet host software. This RFC covers the communications protocol layers: link layer, IP layer, and transport layer; its companion RFC-1123 covers the application and support protocols.

Table of Contents

[Search] [txt|html|pdf|bibtex] [Tracker] [Email] [Diff1] [Diff2] [Nits]

Obsoleted by: 791 — Unknown

Updated by: 777

RFC: 760

IEN: 128

DOD STANDARD

INTERNET PROTOCOL

January 1980

Pact Broker, Pact Flow

- *Repositório de contratos open source e gratuito*
- *Possui uma aplicação web com interface amigável, onde é possível ver contratos, consumers, providers, histórico de testes de verificação dentre outros (PactFlow com bem mais recursos na interface web)*
- *Fácil de subir localmente via Docker*
- *Pact Broker: Free, open source, simples de usar, porém possui poucas features "enterprise"*
  - ***Solução:** PactFlow (versão comercial e paga com mais features, pode ser usado como SaaS, melhores ferramentas de colaboração, melhor segurança, suporte a outros tipos de contrato como os gerados pelo Spring Cloud Contract etc...)*
  - *PactFlow possui trial grátis, limitado a 5 contratos*
- *Seu broker pode operar com ou sem autenticação*
  - *Opere com autenticação*
  - *contratos devem ser publicados sempre a partir de pipelines CI/CD (evite deixar os devs publicarem contratos direto das máquinas deles)*

Considerações finais

THE DEVELOPER'S CONFERENCE

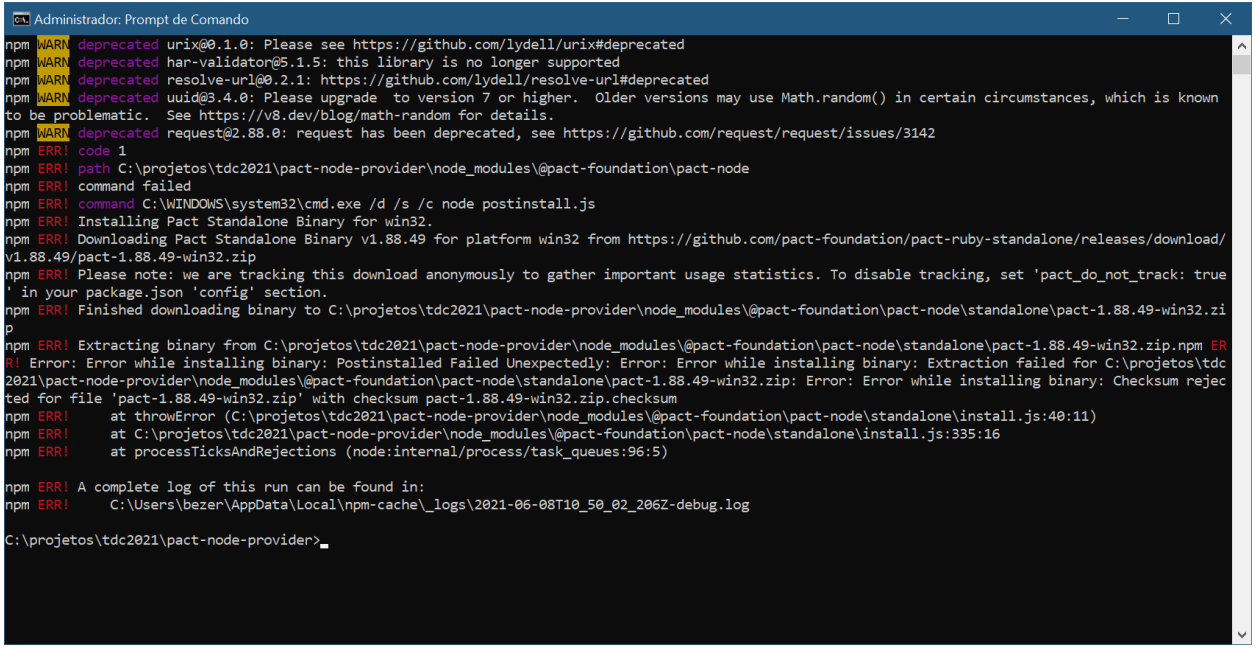
- *Consumer e provider devem estar na mesma versão de specification (v2 ou v3)*
  - *Ou então você vai experimentar erros pouco amigáveis :-)*
- *Escolha bem o que colocar nos testes de contrato (Lei de Postel **Exemplo prático:** o Provider adicionou um campo novo no corpo do response, isso não deveria quebrar um teste de contrato)*
- *Avalie qual abordagem usar (consumer driven x producer driven)*
- *Não dá para usar em qualquer cenário (APIs públicas sem SDK, muitos consumers, consumers desconhecidos, consumer ou provider escritos em linguagem de programação ainda sem suporte sólido ao Pact **Exemplos:** ColdFusion, PactElixir em estado pré-alfa)*
  - *Minha LP não suporta mas quer tentar usar assim mesmo? Ok, boa sorte champs →* https://docs.pact.io/implementation_guides/other_languages *(usar a implementação em C, usar um Pact verifier separado por CLI)*
- *Se puder evite múltiplos consumers subindo contratos idênticos ou redundantes*
- *Pode gerar algum nível de "treta" entre os times no início*

Considerações finais

Pact

THE DEVELOPER'S CONFERENCE

Podem haver alguns problemas envolvendo "Pact Standalone Binary checksum" ao rodar o npm install em Windows 10 + Node 16 + PactJS versão mais recente 9.15.5 (mais estável com Node 14)



Considerações finais

Pact

THE DEVELOPER'S CONFERENCE

SOLUÇÃO 1: *Usar um ambiente dockerizado sem rodar Node localmente*

SOLUÇÃO 2: *Se acalmar e ler a documentação rsrsrsr*

1) Baixar na mao o binario e extrair numa pasta →

https://github.com/pact-foundation/pact-ruby-standalone/releases/tag/v1.88.49

2) Configurar o path da pasta no package.json

3) Rodar (via GIT Bash) ***PACT_SKIP_BINARY_INSTALL=true*** *npm install*

```
{
        "name": "some-project",
        ...
        "config": {
                "pact_binary_location": "/home/some-user/Downloads"
        },
        ...
}
```

DEMO

THE DEVELOPER'S CONFERENCE

APLICAÇÃO PROVIDER: NodeJS+Express
APLICAÇÃO CONSUMER: Java+SpringBoot

- Subir uma instancia do PactBroker com o Docker

- Executar testes no consumer e gerar um contrato

- Publicar o contrato no PactBroker

- Provider baixará os contratos e rodará um teste de verificação que vai passar

- Provider vai fazer uma mudança na API que quebra o contrato, e vai rodar um teste de verificação e ver que não irá passar, detectando a quebra de contrato

- Ver contratos e resultados dos testes exibidos na interface web do PactBroker

*** Códigos de exemplo estarão disponíveis via meus perfis Linkedin+GitHub depois da palestra**

Pact

node + express

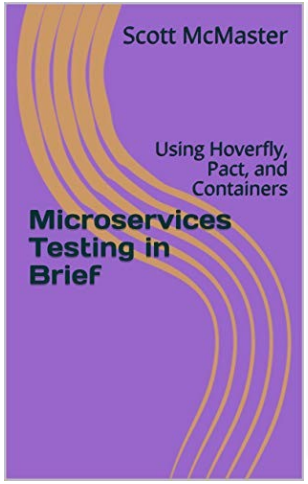
Java

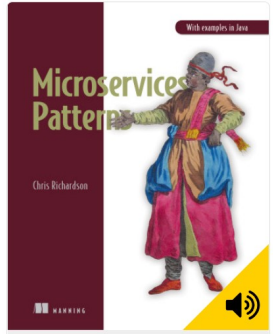
Spring Boot

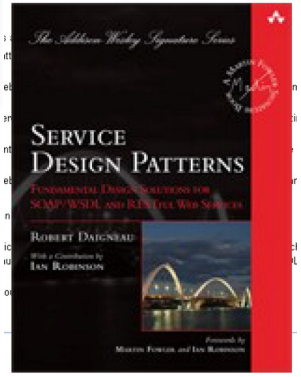
Quero saber mais!

THE DEVELOPER'S CONFERENCE

- ***Microservices. Test smarter, not harder. Voxxed Days 2019***
  https://bit.ly/3gcaz05
- ***ARCast - The Evolution of Architecture with Martin Fowler (2006)***
  https://bit.ly/3fURBMz
- ***Service Design Patterns: Web Service Evolution***
  https://bit.ly/2SYN1nr
- ***Blog Martin Fowler - Consumer-Driven Contracts: A Service Evolution Pattern (2006):*** https://bit.ly/34Xg2CC
- ***NordicAPIs:*** https://nordicapis.com/
- ***Microservices patterns:***
  https://www.manning.com/books/microservices-patterns
  https://microservices.io/patterns/index.html
- ***Pact specification:***
  https://github.com/pact-foundation/pact-specification
- ***Pact e PactBroker docs:*** https://docs.pact.io/
- ***Pact Flow:*** https://pactflow.io/features/
- ***Canal YouTube Pact Flow:*** https://bit.ly/2THtKHn
- ***[Advanced contract testing with Pact] Best practices for writing consumer tests***
  https://www.youtube.com/watch?v=oPuHb9RI8Zo
- ***7 Reasons to Choose Consumer-Driven Contract Tests Over End-to-End Tests:***
  https://reflectoring.io/7-reasons-for-consumer-driven-contracts/
- ***Mocks are not stubs***
  https://martinfowler.com/articles/mocksArentStubs.html
- ***Gauro:*** https://pt.wikipedia.org/wiki/Gauro







PACTFLOW

Pact

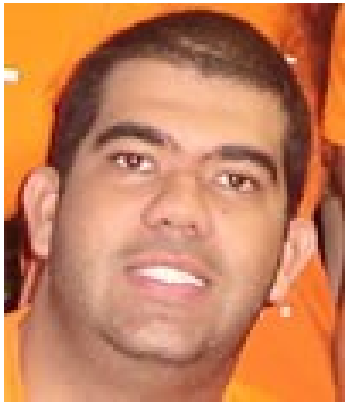
Quero saber mais = Trilha LuizaLabs!!!

THE DEVELOPER'S CONFERENCE

Obrigado pela participação!!! Perguntas?

THE DEVELOPER'S CONFERENCE

Códigos e slides estão disponíveis via meu perfil Linkedin+GitHub



linkedin.com/in/bovino

github.com/bovino

bezerra.mj@gmail.com