Desconstruindo Monolitos

Como construir micro serviços Delphi com agilidade e qualidade

Hello World!!

Felipe Caputo

Desenvolvedor sênior e líder técnico na Softplan, atualmente sou responsável pelo desenvolvimento de soluções para integração entre aplicações dos órgãos da justiça no País.

May Fernandes

Analista de Testes com mais de 08 anos de experiência com testes de aplicações desktop e micro-serviços. Atualmente trabalho com processos de automação de testes e QAOps na Softplan.

Softplan

- Principais sistemas em Delphi com mais de 20 anos
- Milhares de usuários e milhões de linhas de código
- Dezenas de versões
- Diversas integrações
- Evolução para plataforma
- Integração com outras linguagens

Agenda

- Monolitos e sistemas legados
- Porque micro serviços?
- Estratégias de adoção / migração
- QA
- DevOps

1.

MONOLITOS E SISTEMAS LEGADOS

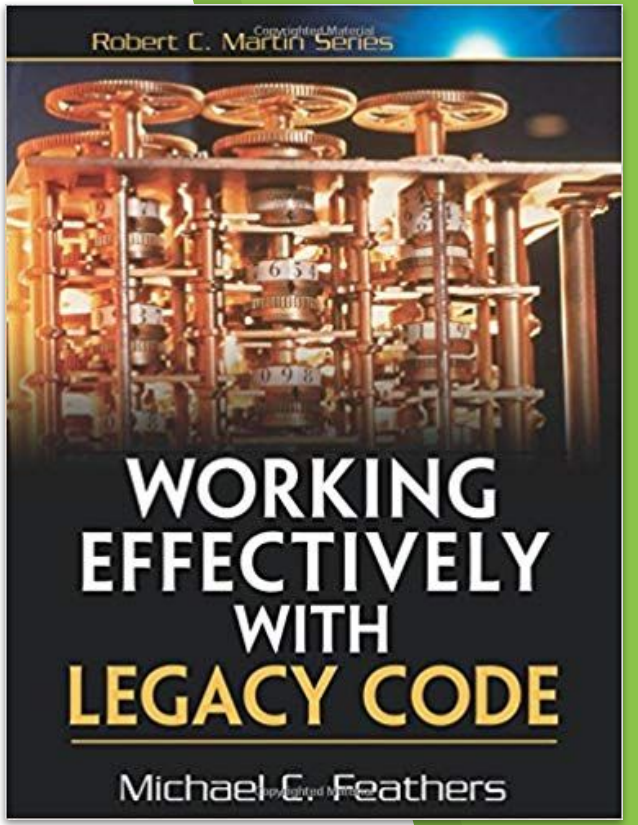
Você sabe o que significa legado?

É o que você deixa para seus filhos e para os filhos dos seus filhos.

É o que resta de você depois que você se for.

Evolução do legado

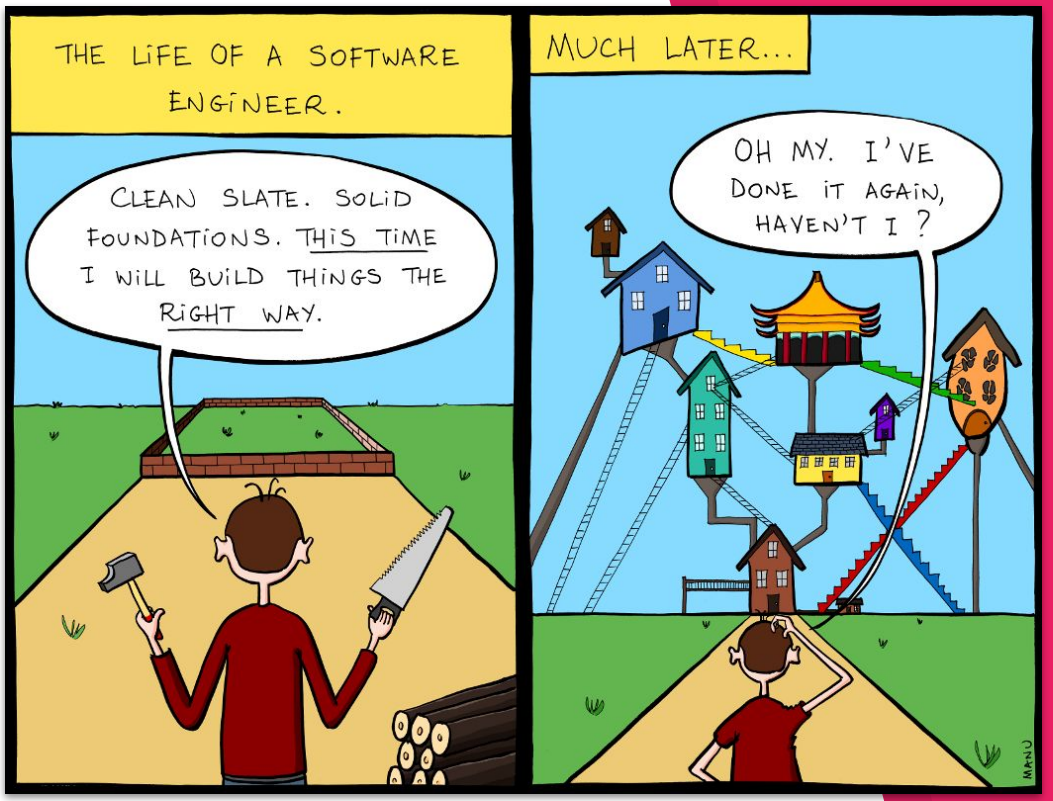
- Evolução gradativa
- Refactoring e testes
- Modernização da aplicação
- Adoção de novas tecnologias de comunicação
- Lento e gradual
- Longo prazo para refletir as mudanças



Fonte: https://www.amazon.com/FEATHERS-WORK-EFFECT-LEG-CODE/dp/0131177052

Evolução do legado: Fazer o novo

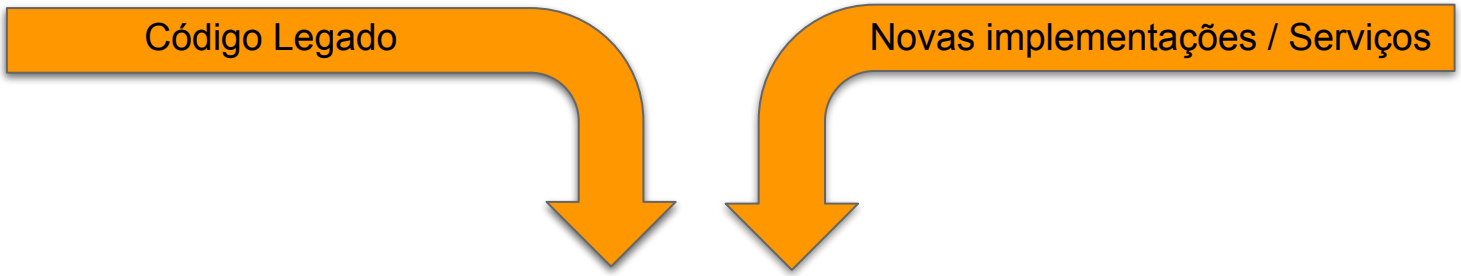
- O novo legado?
- O valor do legado
- Investimento ao longo do tempo
- Existência conjunta
- Riscos



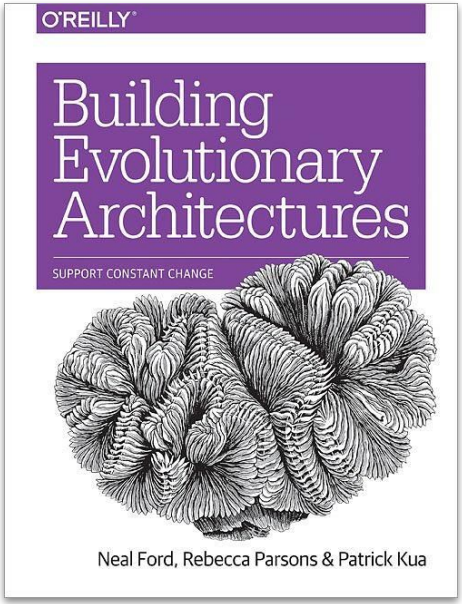
Fonte: http://bonkersworld.net/building-software

O melhor dos dois mundos

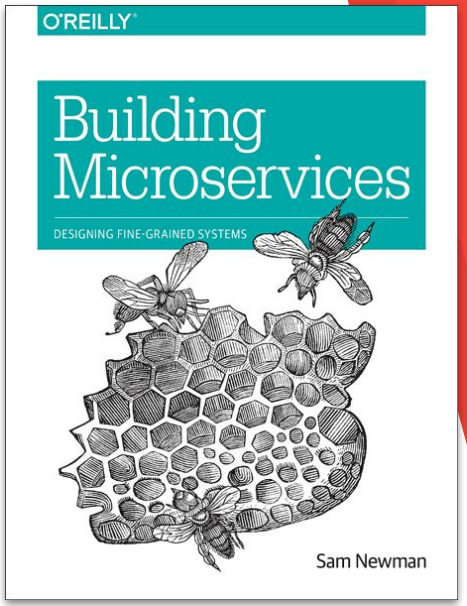
E se fosse possível fazer um novo, evoluindo o velho, adotando novas tecnologias e evoluindo o legado?



Mas e a parte de microserviços?



Fonte: https://www.thoughtworks.com/books/building-evolutionary-architectures



Fonte: https://www.thoughtworks.com/books/building-microservices

2.

POR QUE MICRO SERVIÇOS?

Pontos positivos

- Escalabilidade
- Possibilidade de usar a ferramenta certa para cada serviço
- Menor tempo de implementação e correções
- Serviços de escopo contido e controlado (testabilidade)
- *Continuous Delivery*

Pontos negativos

- Necessidade de infraestrutura maior
- Dificuldade maior nos testes de integração
- Necessidade de monitoramento e automação

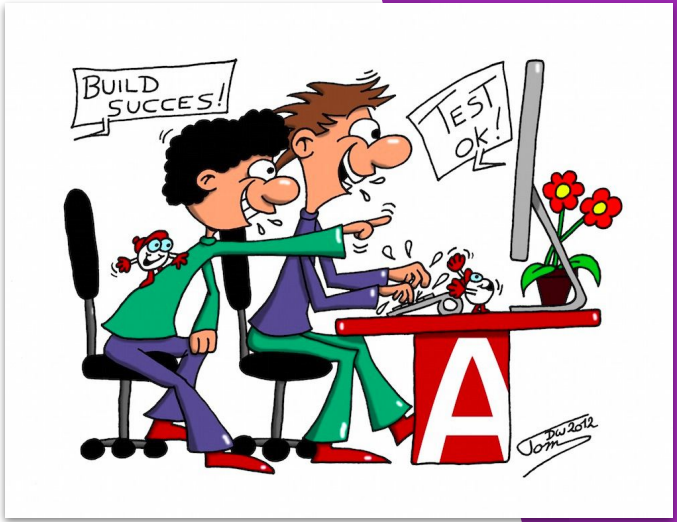
3.

ESTRATÉGIAS DE ADOÇÃO

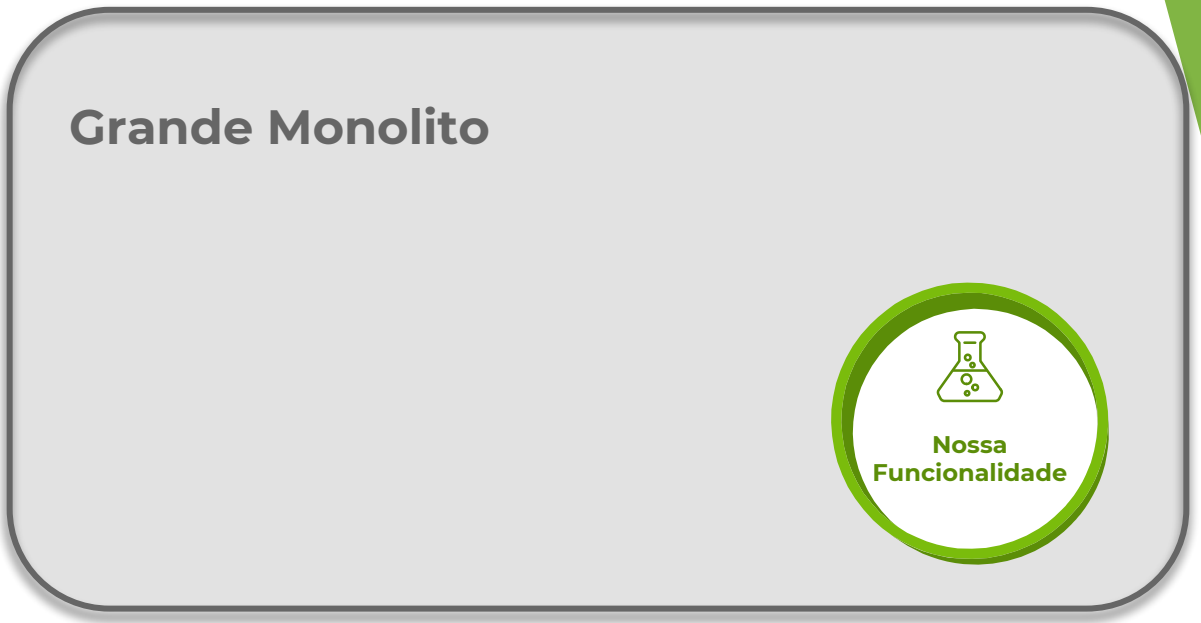
“

Lei de Conway

Organizações que desenvolvem sistemas de software tendem a produzir sistemas que são cópias das estruturas de comunicação dessas organizações.



Cenário Inicial



Cenário Inicial - Visão completa

Passo 1
Lidando com as dependências

Extrair as dependências comuns para serem utilizadas por ambas as aplicações.

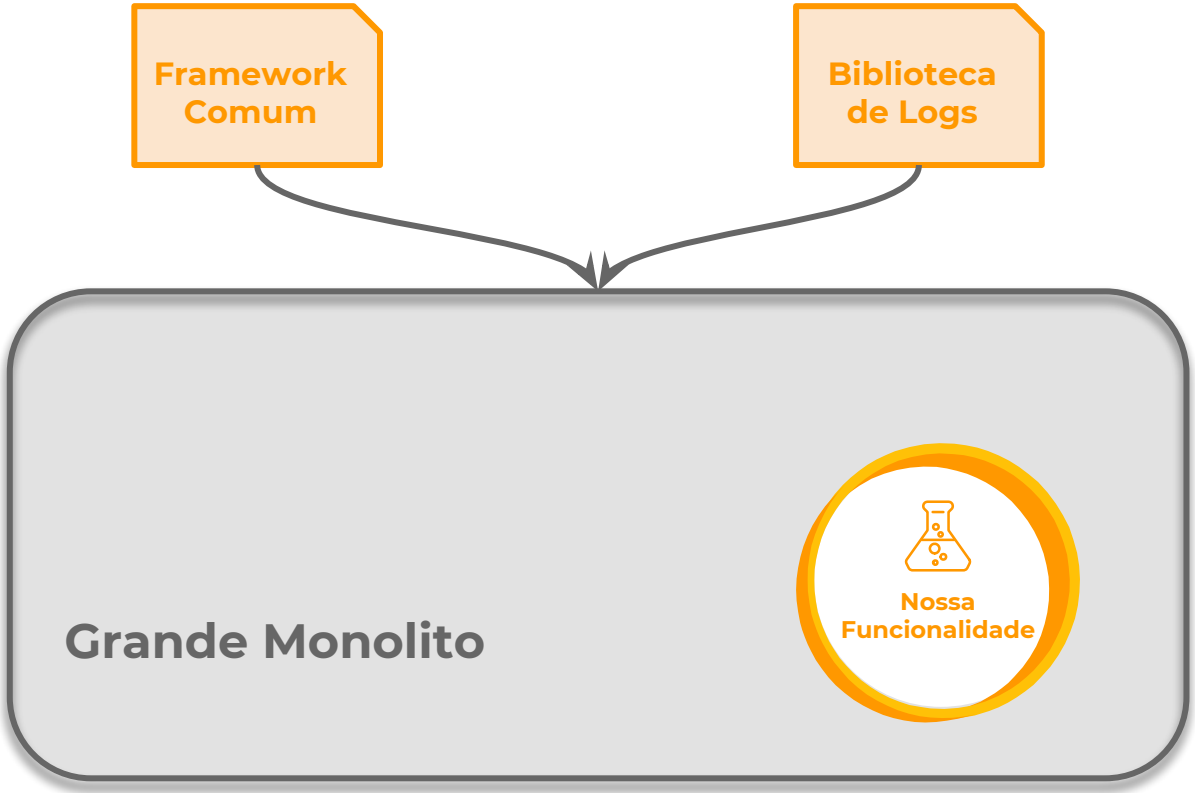
Como exportar com *NuGet*

package.nuspec

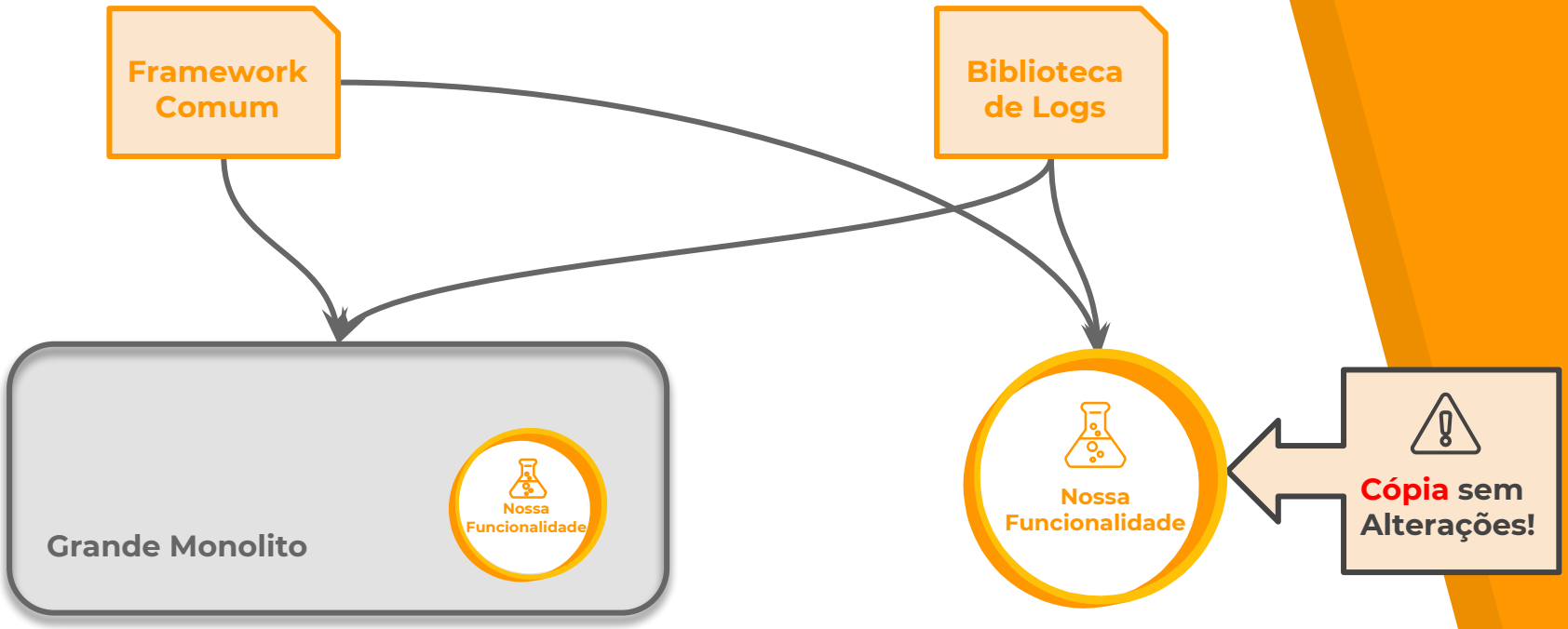
```xml
<?xml version="1.0" encoding="utf-8"?>
<package xmlns="http://schemas.microsoft.com/packaging/2011/08/nuspec.xsd">
  <metadata>
    <id>MEUPROJETO</id>
    <version>0.0.1</version>
    <authors>mygitlab</authors>
    <owners>MY TEAM</owners>
    <description>MEUPROJETO pack</description>
    <releaseNotes>.</releaseNotes>
    <copyright>Copyright 2018</copyright>
    <dependencies>
      <dependency id="DEPENDENCYALFA" version="1.0.1" />
      <dependency id="DEPENDENCYBETA" version="1.0.0" />
    </dependencies>
  </metadata>
  <files>
    <file src="lib\Win32\**" target="lib\Win32\" />
    <file src="lib\Win64\**" target="lib\Win64\" />
    <file src="lib\Linux64\**" target="lib\Linux64\" />
    <file src="bin\Win32\**" target="bin\Win32\" />
    <file src="bin\Win64\**" target="bin\Win64\" />
    <file src="bin\Linux64\**" target="bin\Linux64\" />
  </files>
</package>
```

```
- nuget.exe pack -version %CI_COMMIT_TAG%
- nuget.exe push "MEUPROJETO.%CI_COMMIT_TAG%.nupkg" -source %project_artifactory_url% -APIKey %NugetAPIKey%
```

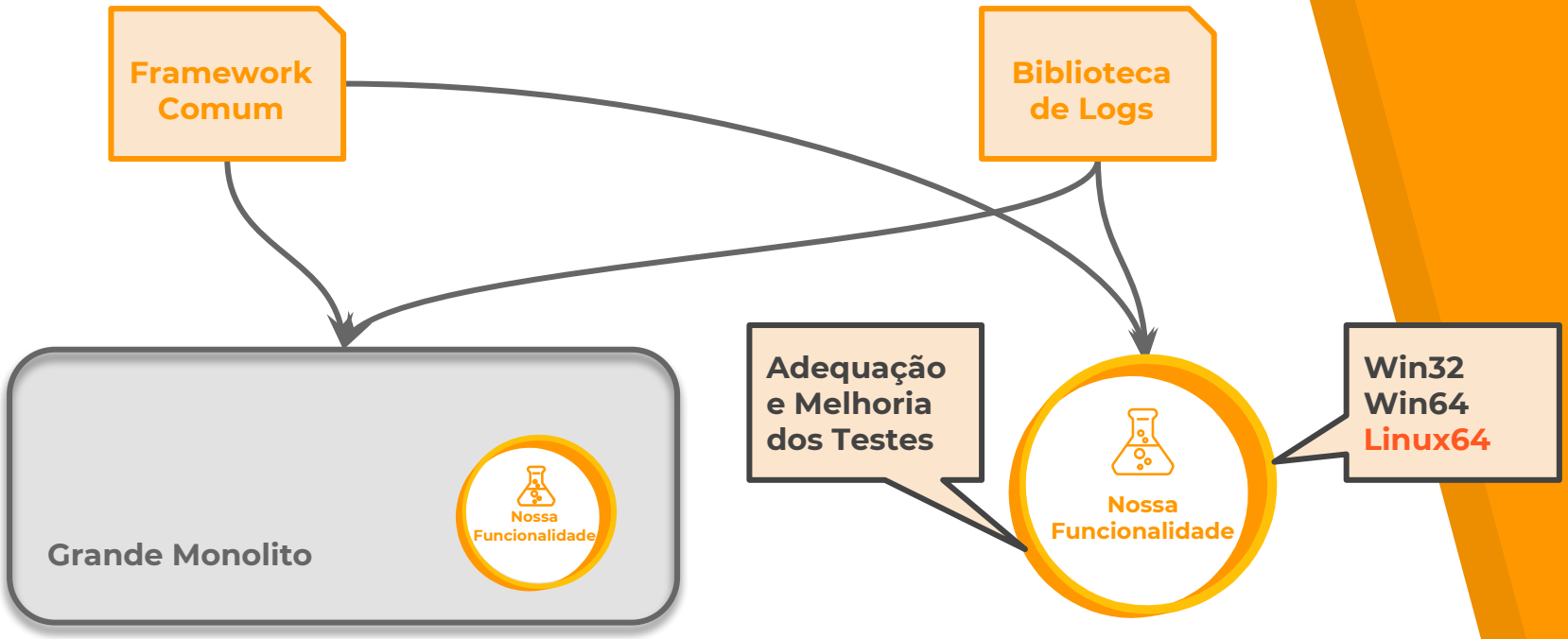
Extração das bibliotecas



Passo 2
Extraindo os fontes



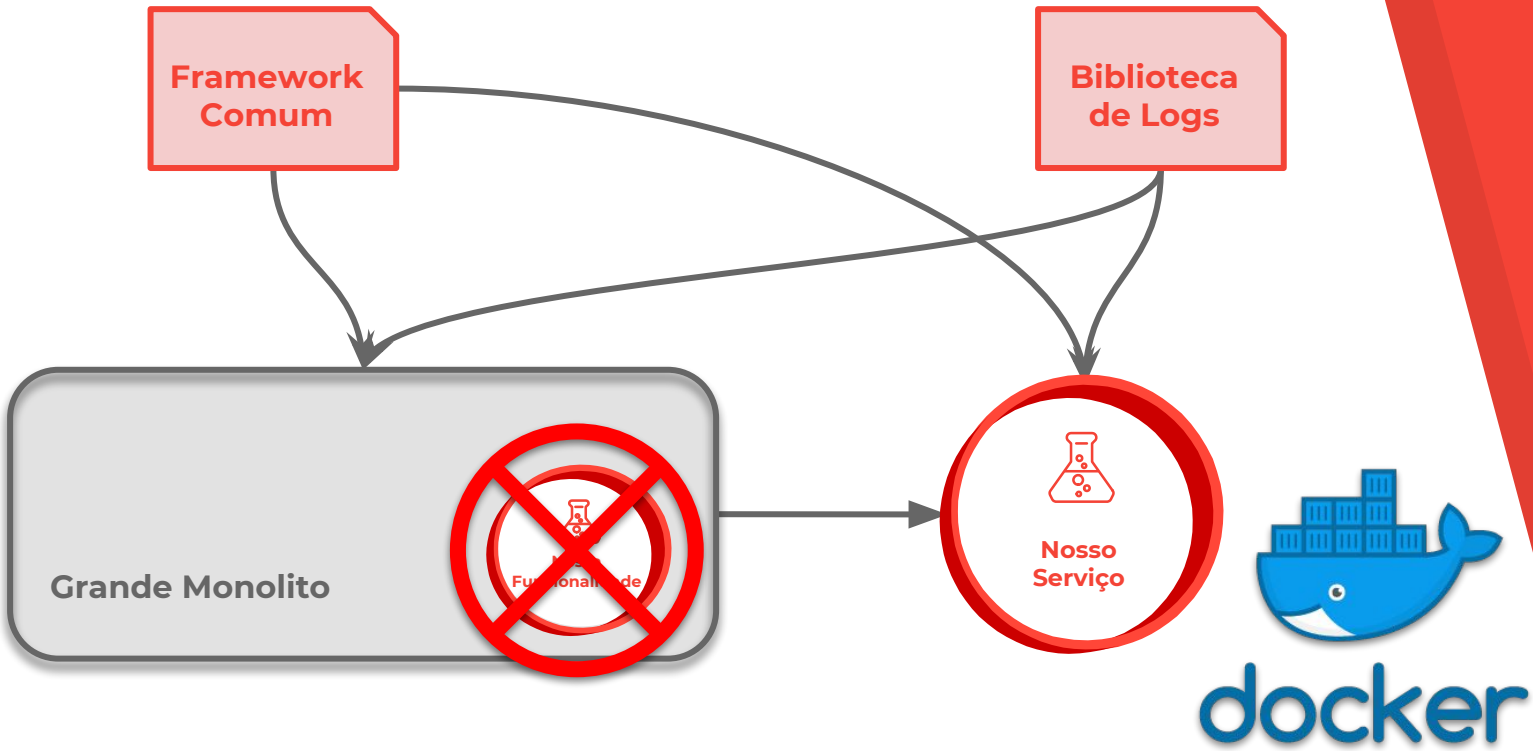
Passo 2
Migrando os fontes para versão *Delphi Tokyo*



📖 Passo 2
O que aprendemos migrando os fontes para a versão *Delphi Tokyo*

- Uso de namespaces é excelente para identificar e gerir cada dependência e serviço.
- Multiplataforma ainda precisa de {**$IFDEF PLATFORM**}
- Testes executados em cada plataforma (**DUnitX**)
- ***TObjectList*** não é multiplataforma, mas o ***TObjectList<TObject>*** é.

Passo 3
Transformando funcionalidade em Serviço

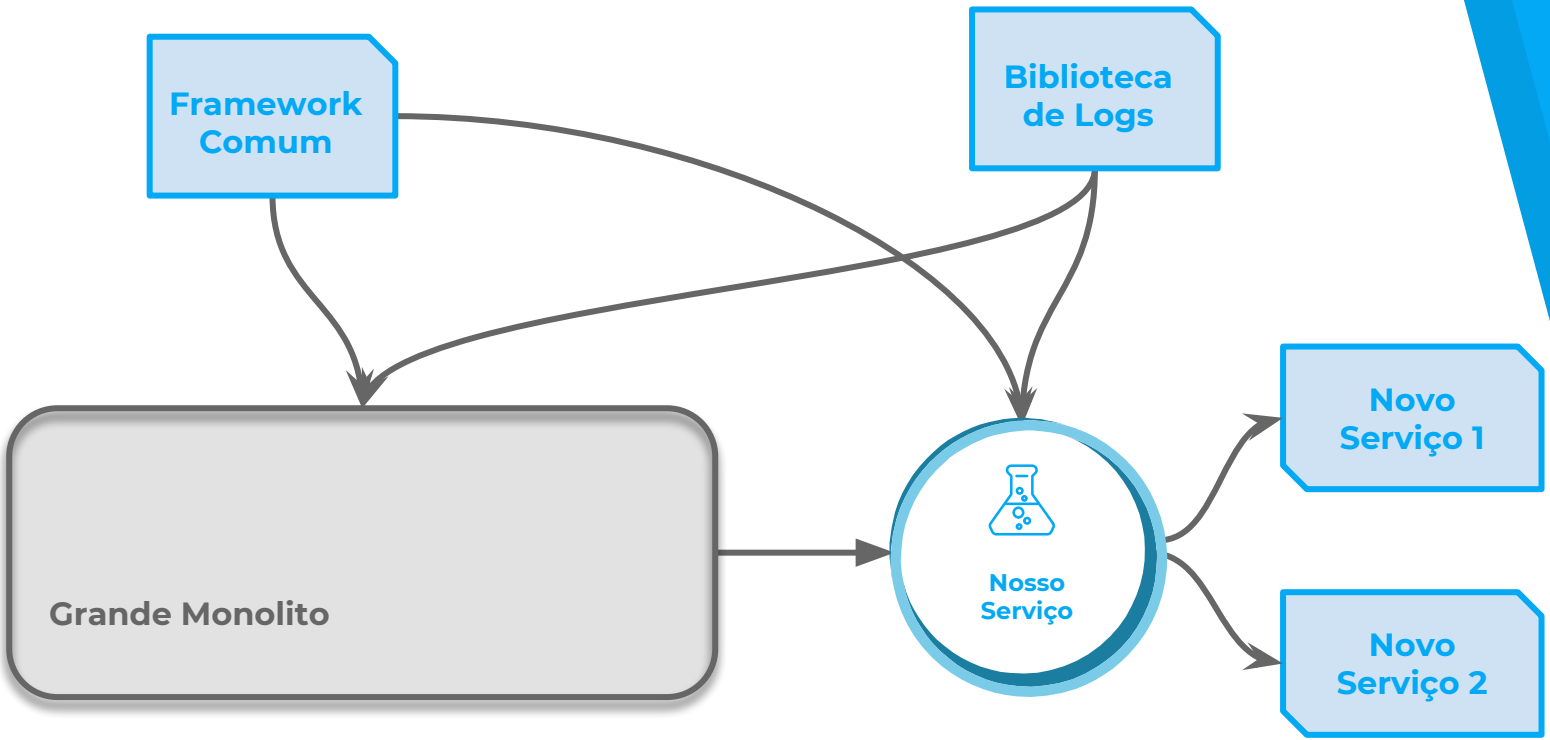


Passo 3
DockerFile

```
FROM ubuntu:xenial

RUN apt-get update && apt-get install -y libmongoc-1.0-0 && \
    ln -s /usr/lib/x86_64-linux-gnu/libmongoc-1.0.so.0 /usr/lib/x86_64-linux-gnu/libmongoc-1.0.so && \
    ln -s /usr/lib/x86_64-linux-gnu/libbson-1.0.so.0 /usr/lib/x86_64-linux-gnu/libbson-1.0.so && \
    apt-get install tzdata && \
    apt-get clean && rm -rf /var/lib/apt/lists/*

RUN echo "America/Sao_Paulo" > /etc/timezone && \
    dpkg-reconfigure -f noninteractive tzdata
```

```
FROM docker-unj-repo.softplan.com.br/unj-integracoes/ubuntu-delphi-mongo:1.1.0

RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app

#Adding dependencies
ADD dependencies/dlls/Linux64/ /usr/src/app
ADD dependencies/dlls/Linux64/libicu55/ /usr/lib/x86_64-linux-gnu/

COPY bin/Linux64/Release/UnjMniService /usr/src/app/

RUN chmod +x UnjMniService

CMD [ "./UnjMniService"]
```

Passo 4 - Mais serviços surgiram



4.

QA

Story of developers in my team.

^^

A importância da automação dos testes End2End

- Garantem os contratos
- Garantem que a integração entre os serviços continue funcionando
- Continuam a existir mesmo com mudança de tecnologia
- Servem de documentação

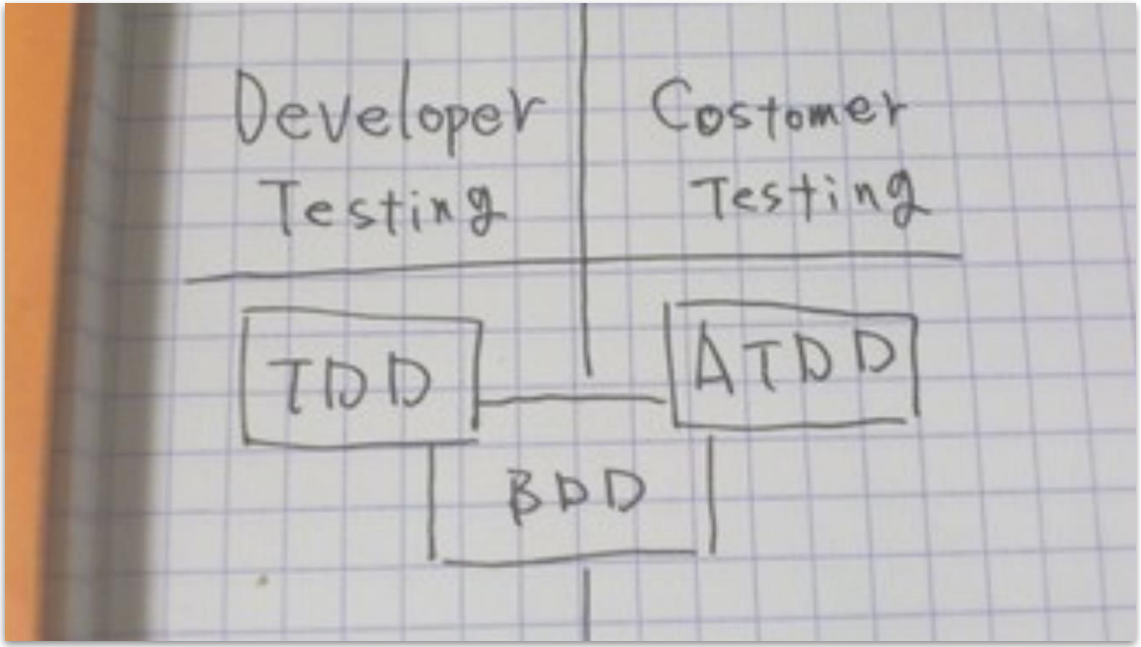
Estratégias de QA

- Metodologia Ágil - ATDD
- Testes unitários (TDD) e *Code Review*
- Testes funcionais com framework com ***Robot Framework***
- Documentação viva (BDD executável)
- Testes não-funcionais com ***Locust e Python***

Metodologia ATDD
Acceptance Test-Driven Development

- Critérios de Aceitação definidos em BDD (*Behavior Driven Development*)
- Refinamento dos critérios em time (*PO + DEV + QA*)
- TDD
- Os testes automatizados funcionais se tornam a documentação viva do produto
- *Definition of Done*: 100% dos critérios de aceitação entregue automatizados

Metodologia ATDD
Acceptance Test-Driven Development



Metodologia ATDD
Acceptance Test-Driven Development

```
*** Settings ***
Resource          ResourceBDD.robot
Resource          BDDpt-br.robot
Suite Teardown    Fechar Navegador

*** Test Cases ***
Cenário 01: Pesquisar postagem Season Premiere
    Dado que esteja na tela HOME do blog robotizando testes
    Quando pesquisar pela palavra "introdução"
    Então a postagem "Season Premiere: Introdução ao Robot Framework" deve ser listada no resultado da pesquisa

Cenário 02: Ler postagem Season Premiere
    Dado que esteja na tela de resultado da pesquisa pela postagem "Season Premiere: Introdução ao Robot Framework"
    Quando clicar no link da postagem
    Então a tela da postagem "Season Premiere: Introdução ao Robot Framework" deve ser mostrada
```

Testes Unitários e *Code Review*

- Testes unitários com DUnitX (multiplataforma)
- Implementação suportando testes
  - Uso intensivo de interfaces para facilitar *Mocks*
- TDD e *Test First*
- ***Code Review:*** Oportunidade para outros DEVs acompanharem o projeto
- ***Code Review***: Boas práticas compartilhadas e feedbacks de possíveis problemas

Automação dos Testes Funcionais
Robot Framework

- **Open Source**: Framework Python, muito bem mantido pela comunidade e documentação abrangente
- **Genérico**: permite automação de qualquer sistema (web, serviço, desktop, mobile, etc)
- **Produtivo**: Possui diversas bibliotecas com códigos prontos para automação de testes (*RequestsLibrary*, *Collections*, *RabbitMQLibrary*, *RedisLibrary*, *MongoDBLibrary*, etc)
- **Documentação viva**: A automação é feita por linguagem mais humana e suporta o BDD
- **Virtualização dos Serviços**: Utilização de *mocks* para testes *End2End*

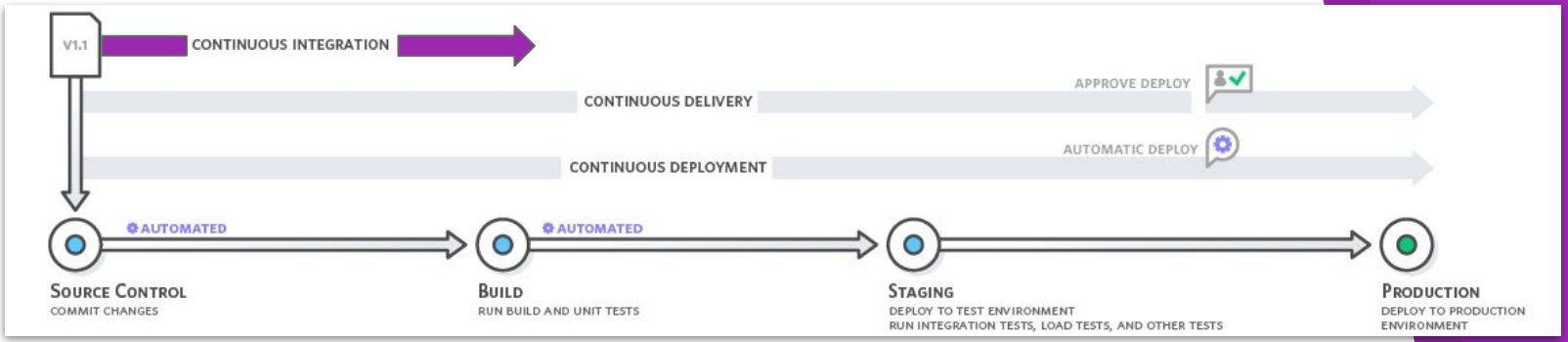
Automação dos Testes Não-Funcionais *Locust e Python*

- ***Load Testing***: testes de carga contra os micro serviços
  - Utilizamos o ***LOCUST*** open source em python
  - Se o micro serviço se comunicava via mensageria, utilizamos API REST auxiliar para enviar e receber requisições
- ***Memory check***:
  - Captura do status da memória do container após várias rodadas de teste (Linux) e FastMM4 (Windows)
  - Análise do vazamento de memória via script Python (Linux)

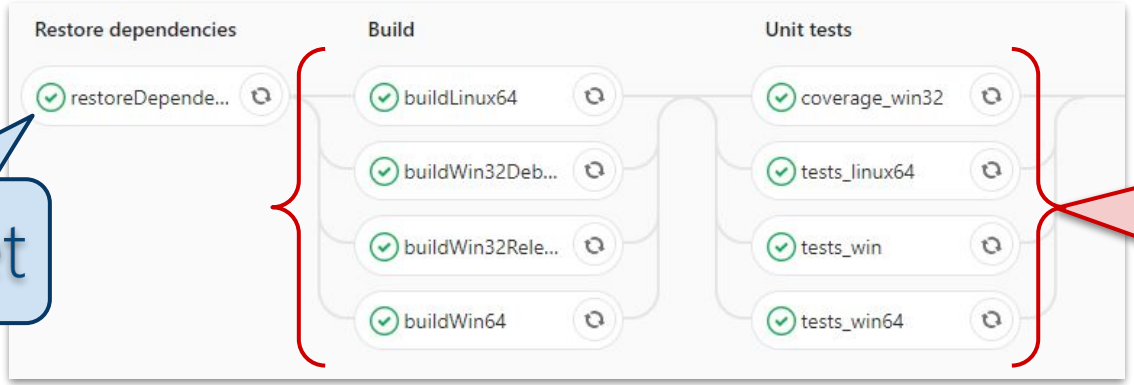
5.

DEVOPS

Integração Contínua
Continuous Integration



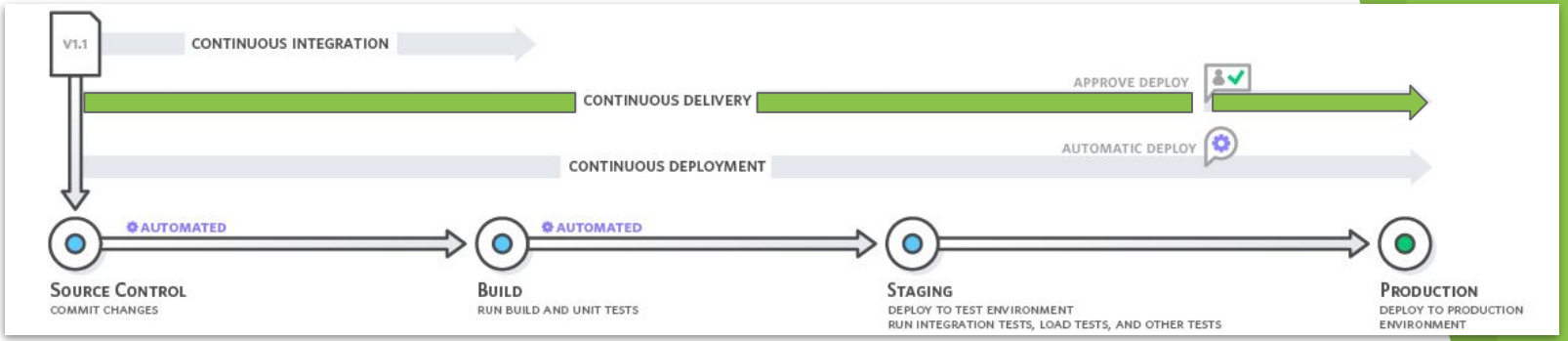
https://aws.amazon.com/pt/devops/continuous-integration/



nuget

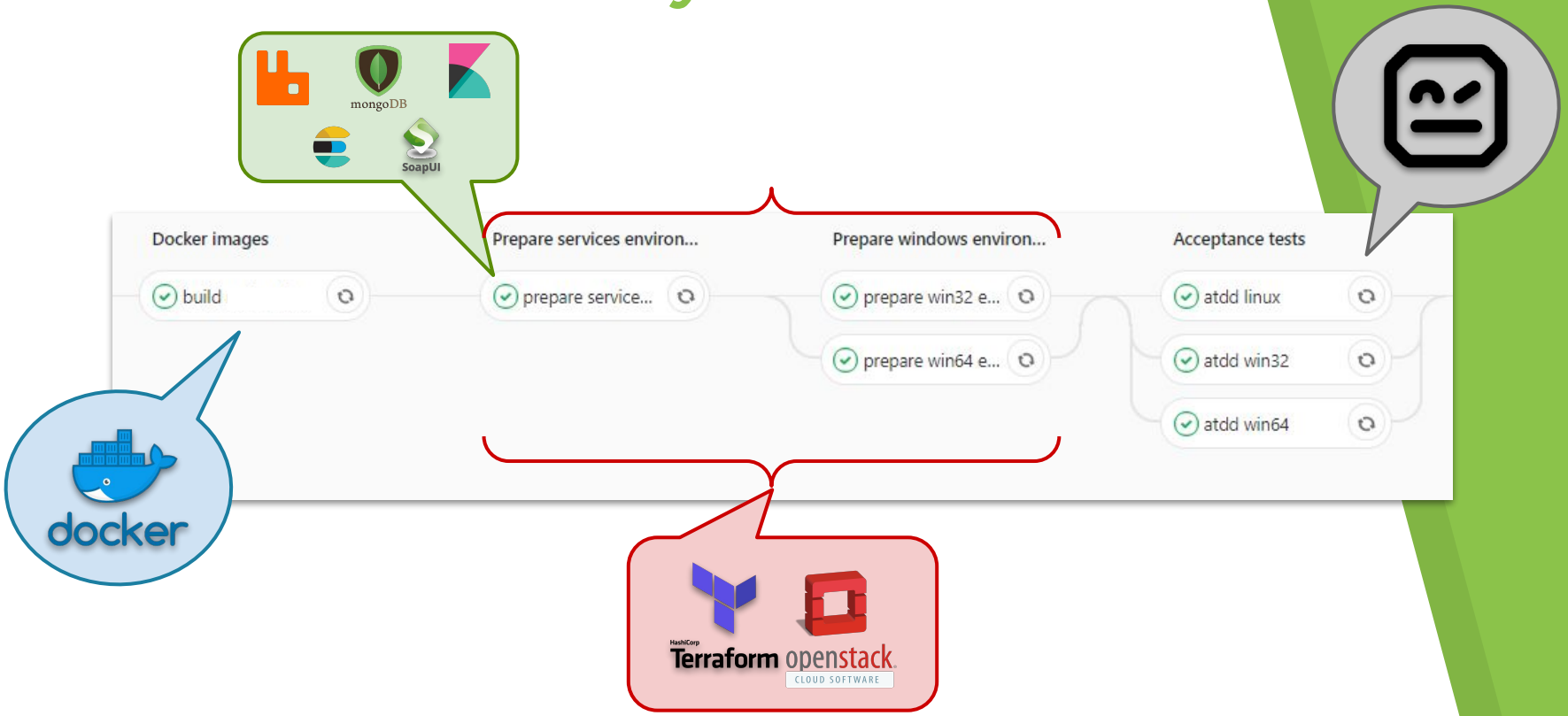
DX

Entrega Contínua
Continuous Delivery

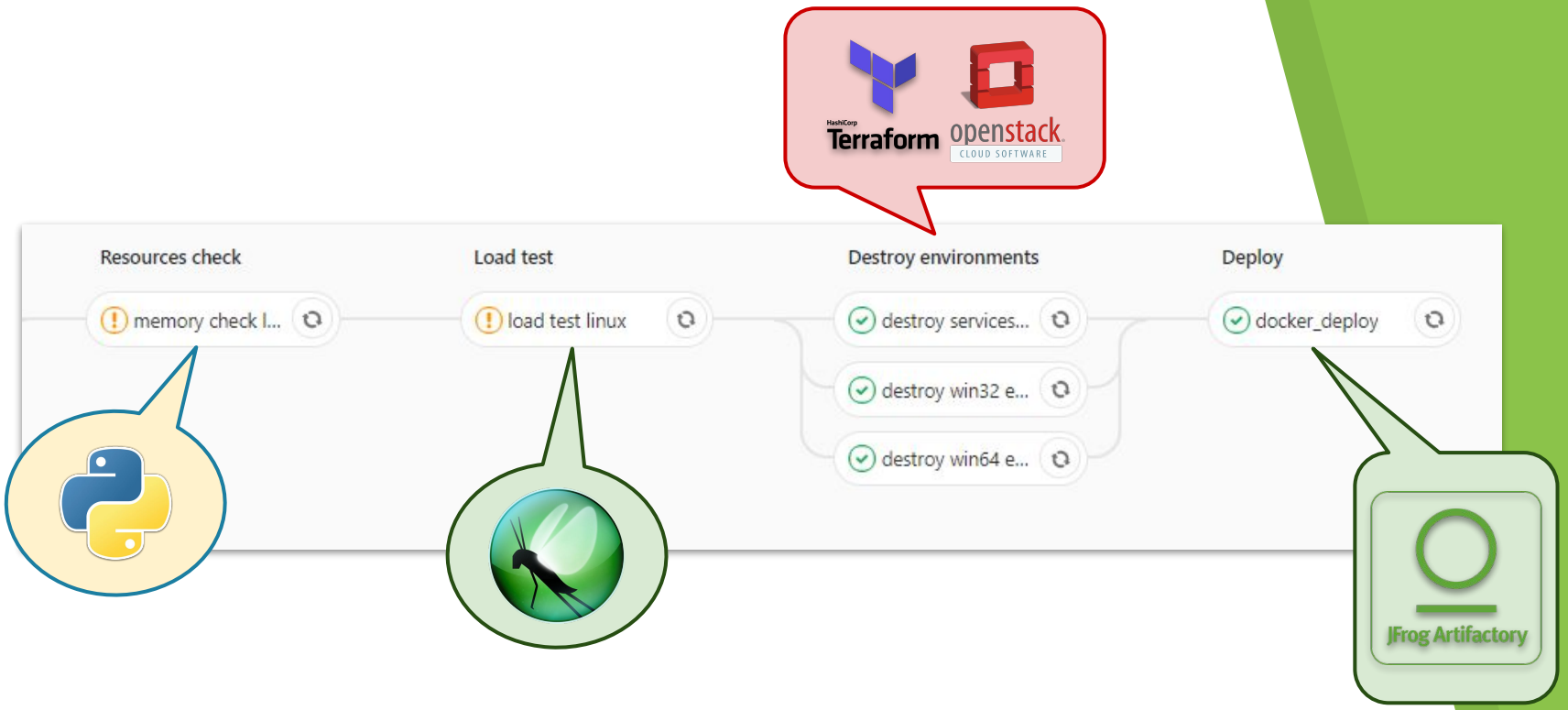


https://aws.amazon.com/pt/devops/continuous-integration/

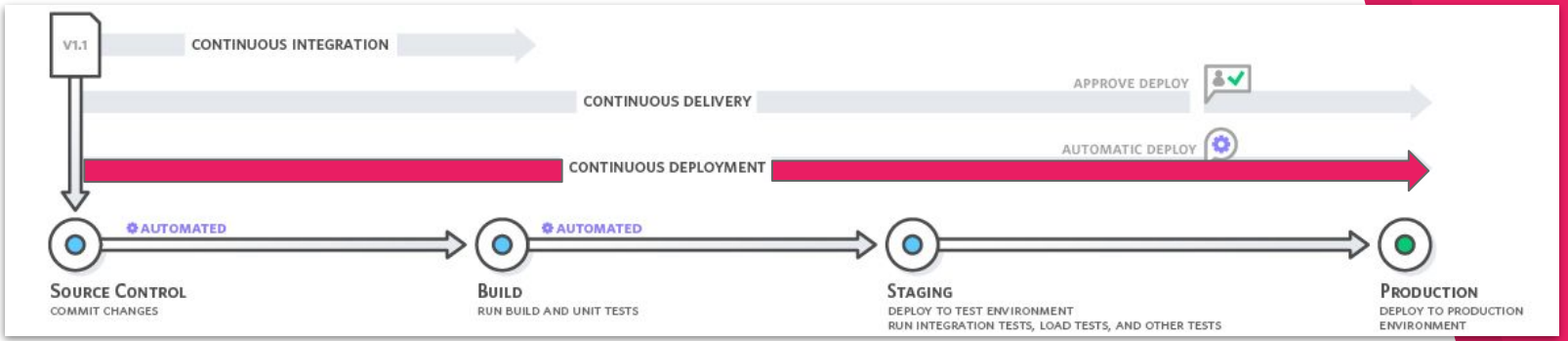
Entrega Contínua
Continuous Delivery



Entrega Contínua
Continuous Delivery



Implantação Contínua
Continuous Deployment



https://aws.amazon.com/pt/devops/continuous-integration/

O que utilizamos

THANKS!

Perguntas?

Vocês podem nos achar em:

felipe.caputo@gmail.com

ammmayara@hotmail.com

https://www.linkedin.com/in/felipewcaputo/

https://www.linkedin.com/in/mayfernandes/

https://github.com/felipecaputo

https://github.com/mayribeirofernandes