

# Bootcamp

# Arquiteto de Soluções - Cloud

Módulo 4 - Soluções para Desenvolvimento

Capítulo 1. Introdução – Soluções para desenvolvimento

**ANALÍA IRIGOYEN**



# Arquiteto de Soluções

---

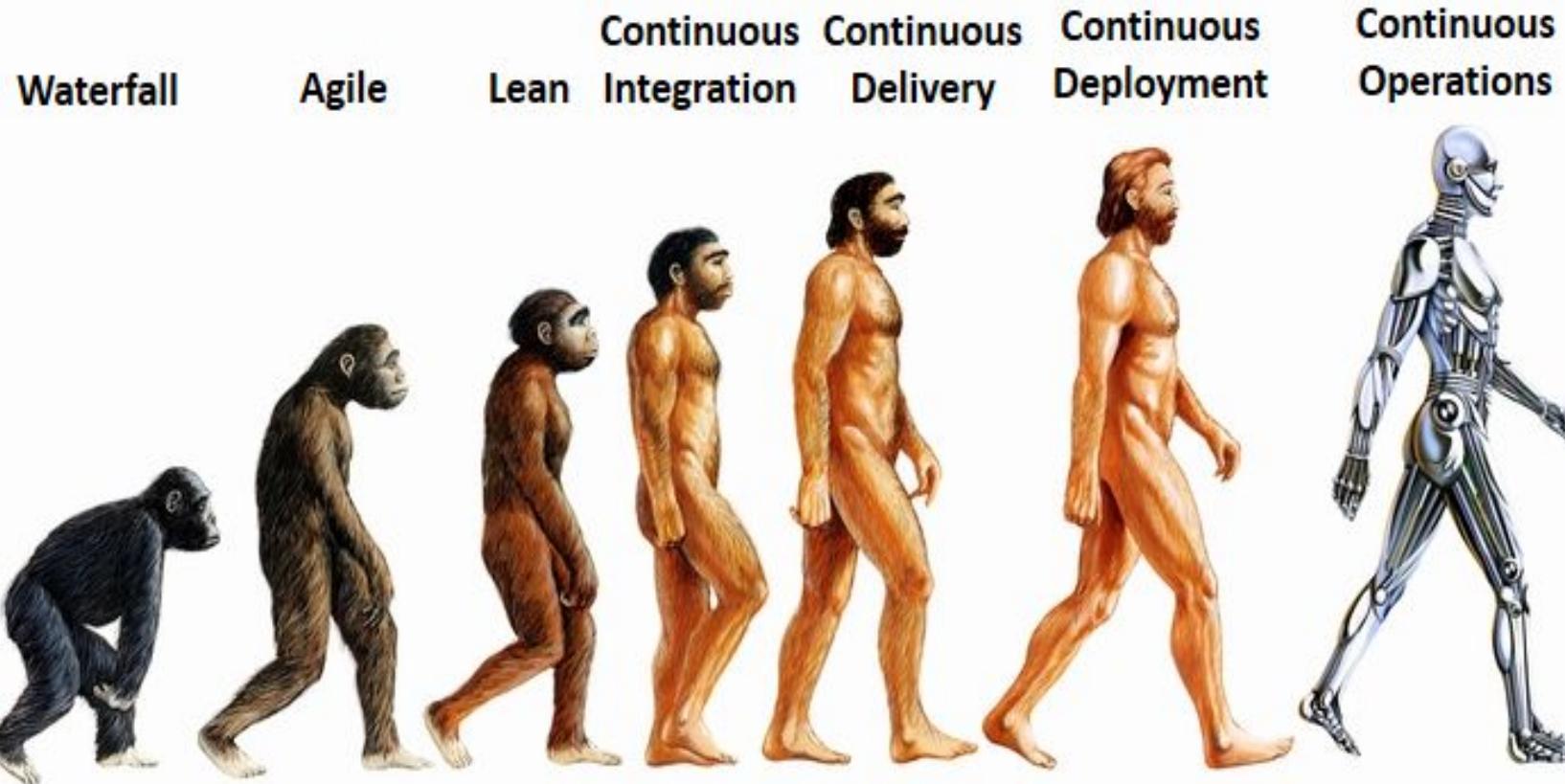
Aula 1.1. Motivação – Por que DevOps?

ANALIA IRIGOYEN

# Terceira Maneira - DevOps



- Motivação:
  - Por que DevOps?
  - Introdução ao DevOps.
  - O que é DevOps Calms.
  - Princípio das 3 maneiras.



# Falhas inaceitáveis



HSBC - 2016

- Milhões de clientes sem acesso a conta por 2 dias

```
ts/www$ sudo npm install
ts/www$ npm -v
ts/www$ sudo apt-get re
?
Done
ed, so not removed
4, 0 to remove and 14 no
ts/www$ npm -v
ts/www$
```

Hospital NHS - 2018

- Dados dos pacientes inacessíveis



Vazados 988 mil arquivos -2019

- Falha de cartórios expõe dados de ao menos 1 milhão de pais, mães e filhos...

IGTI

# **“Código mal escrito” gera perdas de US\$ 85 bi por ano às empresas**

40% do tempo das equipes dedicado para correções

Estudo com mil Desenvolvedores e mil executivos C-level

<https://www.itforum365.com.br/carreira/codigo-mal-escrito-gera-perdas-de-us-85-bi-por-ano-as-empresas/>



## Mindset aceitável no passado

Filas no checkin para embarcar

Blockbuster multa por atraso

TV domina conteúdo e audiência

Guia 4 rodas para viagem

Momento kodak em papel

Comunicação por email

Filas para transações bancárias

Filas no orelhão

Táxi acessível para poucos

Anúncio caro: TV, jornal e revista

Reembolso médico em papel

Linear e analógico

## Mindset digital exige DevOps

Checkin via app

Netflix entrega vídeo sob demanda

Netflix domina conteúdo e audiência

Google compra Waze por U\$ 1 Bi

Facebook compra instagram por U\$ 1 Bi

Facebook compra WhatsApp por U\$ 22 Bi

Transações na palma da mão

Qtd de celular supera qtd de pessoas

Uber acessível para muitos

Domínio do Google e mídias sociais

**Reembolso digital e ágil**

Exponencial e digital

## Modelo tradicional

Cultura do medo e grito

Causa da falha foi o fulano

Competição entre departamentos

Cada um no seu quadrado

Teste manual no final

Merge complexo e demorado

Tudo funciona na minha máquina

Implantação manual

Implantação demorada

Big bang

Acho que o problema foi xxx

## Cultura DevOps

Confiança e experimentação



Resolvemos a falha sistêmica

Colaboração multidisciplinar

Visão mais horizontal

Teste automatizado na origem

Integração em lotes pequenos

Ambiente similar de produção

Implantação automatizada

Implantação contínua

Teste A/B

Fatos e dados com telemetria

# Jornada DevOps

Lean, Scrum,  
Definição de Pronto,  
Kanban, XP, Kata,  
Teoria das Restrições

Cultura Colaborativa,  
Facilitação,  
Transparência,  
Confiança

Revisão de Código,  
Programação em  
Pares, Gestão de  
Mudança e  
Segurança Contínua

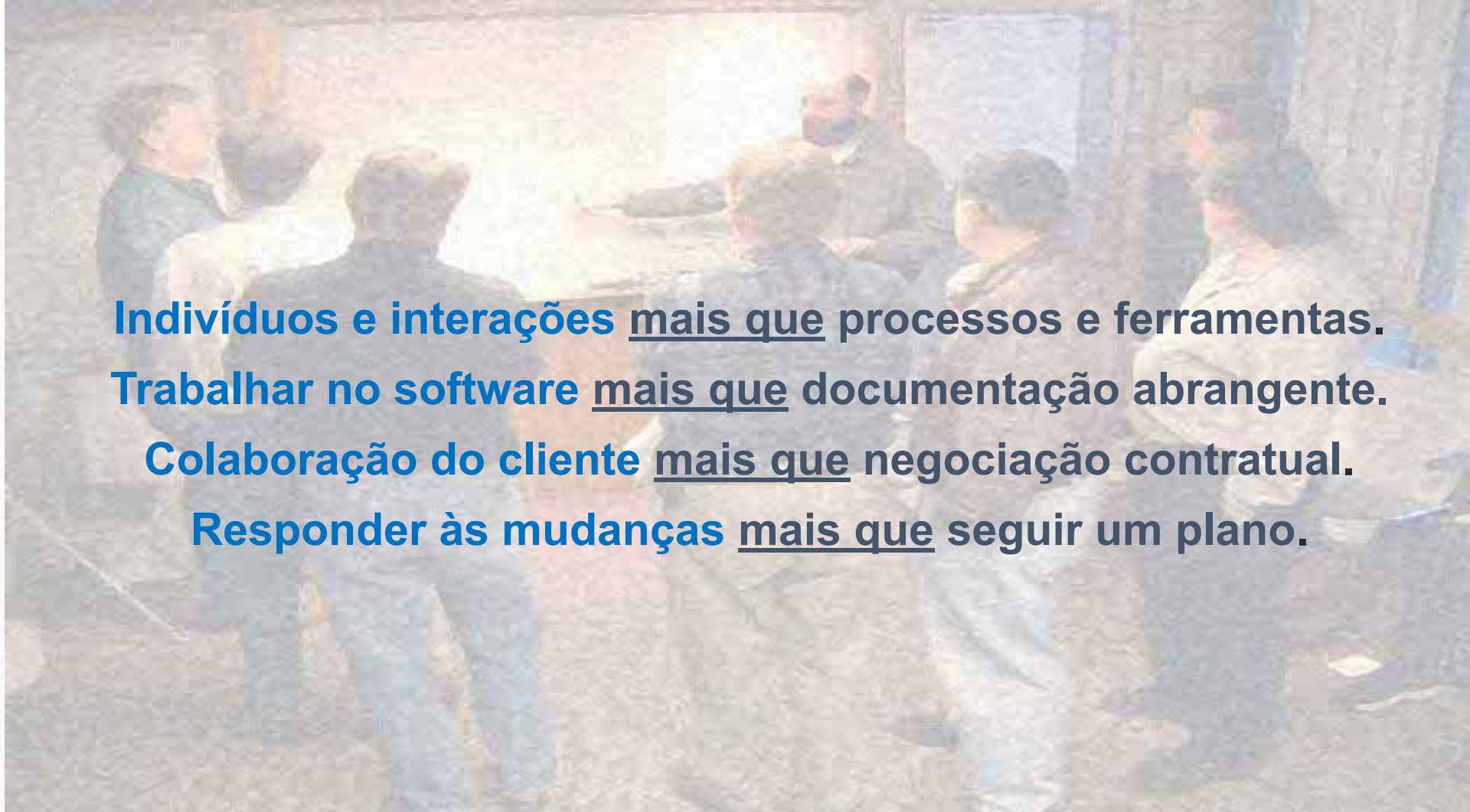
Integração Contínua,  
Entrega Contínua,  
Implantação Contínua

Pipeline de  
Implantação,  
Automação de  
Testes, TDD

Telemetria,  
Experimentação,  
Feedback Contínuo,  
Testes A/B

Microserviço, APIs,  
Infraestrutura como código,  
Releases de Baixo Risco,  
Injeção de Falhas



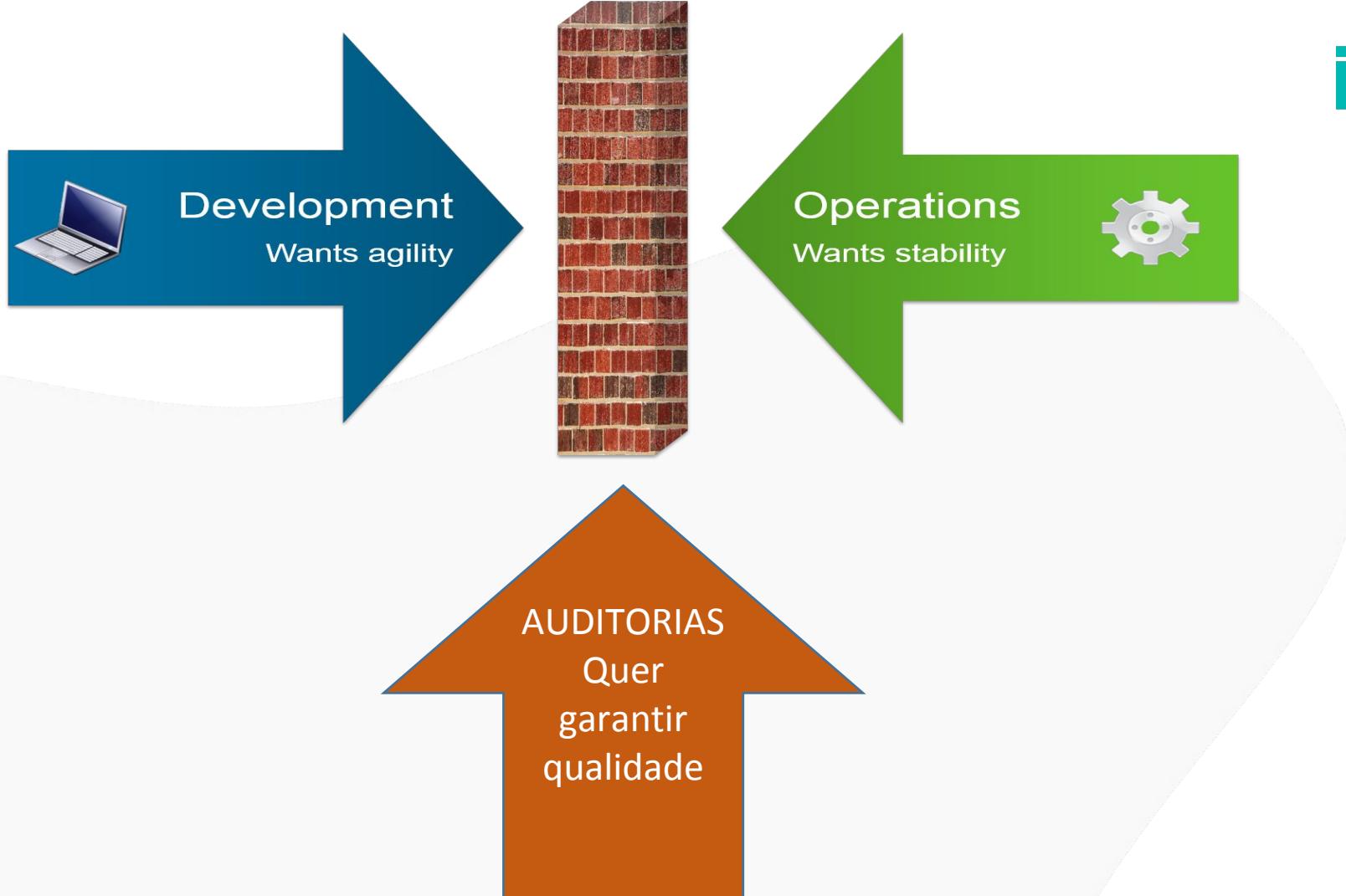


**Indivíduos e interações mais que processos e ferramentas.**

**Trabalhar no software mais que documentação abrangente.**

**Colaboração do cliente mais que negociação contratual.**

**Responder às mudanças mais que seguir um plano.**



# Introdução e adoção do Devops – Conceitos Básicos

1	Os	2	En
<b>Gl</b> GitLab		<b>Sp</b> Splunk	
3 Fm	4 En	5 En	6 Fm
<b>Gh</b> GitHub	<b>Dt</b> Datical	<b>Aws</b> AWS	7 Pd
11 Os	12 En	<b>Az</b> Azure	8 En
<b>Sv</b> Subversion	<b>Db</b> DBMaestro	<b>Gc</b> Google Cloud	9 Fm
19 En	20 En	<b>Op</b> OpenShift	10 Fm
<b>Cw</b> ISPW	<b>Dp</b> Delphix	<b>Sg</b> Sumo Logic	
21 Os	22 Fm	13 Os	
<b>Jn</b> Jenkins	<b>Cs</b> Codeship	<b>XLr</b> XebiaLabs XL Release	14 En
23 Os	24 Fr	<b>Aws</b> AWS	15 Pd
<b>Fn</b> FitNesse	<b>Ju</b> JUnit	<b>Az</b> Azure	16 Pd
25 Fr	26 Fm	<b>Gc</b> Google Cloud	17 Fm
<b>Ka</b> Karma	<b>Su</b> SoapUI	<b>Op</b> OpenShift	18 Os
27 En	<b>Ch</b> Chef	<b>Sg</b> Sumo Logic	
28 Fr	<b>Tf</b> Terraform	19 En	
<b>XLd</b> XebiaLabs XL Deploy	<b>Ud</b> UrbanCode Deploy	20 En	
31 Os	32 Fm	21 Os	
<b>Ku</b> Kubernetes	<b>Cc</b> CA CD Director	<b>Pr</b> Plutora Release	22 Os
33 En	34 Pd	<b>Al</b> Alibaba Cloud	23 Os
<b>Cc</b> CA CD Director	<b>Ur</b> UrbanCode Release	<b>Os</b> OpenStack	24 Os
35 Os	36 Os	<b>Ps</b> Prometheus	
37 Pd	38 Fm	25 Os	
<b>At</b> Artifactory	<b>Rg</b> Redgate	<b>Pd</b> Pd	37 En
39 Pd	<b>Ba</b> Bamboo	<b>S1</b> Rm	
40 Fm	<b>Vs</b> VSTS	<b>52</b> Pd	
41 Fr	<b>Se</b> Selenium	<b>Cp</b> AWS CodePipeline	
42 Fr	<b>Jm</b> JMeter	<b>Cy</b> Cloud Foundry	
43 Os	<b>Ja</b> Jasmine	<b>It</b> ITRS	
44 Pd	<b>Sl</b> Sauce Labs	44 En	
45 En	<b>An</b> Ansible	<b>Ms</b> Mesos	45 En
46 Os	<b>Ru</b> Rudder	<b>Gke</b> GKE	
47 En	<b>Oc</b> Octopus Deploy	<b>Om</b> OpenMake	
48 Os	<b>Go</b> GoCD	<b>Cp</b> AWS CodePipeline	
49 Os	<b>Rs</b> Go	<b>Cy</b> Cloud Foundry	
50 Pd	<b>Ms</b> Mesos	<b>It</b> ITRS	
51 Rm	<b>Gke</b> GKE	53 Pd	
52 Pd	<b>Om</b> OpenMake	54 En	
53 Pd	<b>Cp</b> AWS CodePipeline	55 En	
54 En	<b>Cy</b> Cloud Foundry	56 En	
55 Pd	<b>Nx</b> Nexus	<b>It</b> ITRS	
56 Os	<b>Fw</b> Flyway	57 Os	
57 Os	<b>Tr</b> Travis CI	<b>Tc</b> TeamCity	58 Fm
58 Fm	<b>Ga</b> Gatling	<b>Tn</b> TestNG	59 Os
59 Os	<b>Tt</b> Tricentis Tosca	<b>Pe</b> Perfecto	60 Fr
60 Fr	<b>Pu</b> Puppet	<b>Pa</b> Packer	61 Fm
61 Fm	<b>Cd</b> AWS CodeDeploy	<b>Ec</b> ElectricCloud	62 Pd
62 Pd	<b>Ra</b> Rancher	<b>Rk</b> Rkt	63 En
63 En	<b>Aks</b> AKS	<b>Sp</b> Spinnaker	64 Os
64 Os	<b>Ec</b> ElectricCloud	<b>Rk</b> Rkt	65 Fm
65 Fm	<b>Ra</b> Rancher	<b>Sp</b> Spinnaker	66 En
66 En	<b>Aks</b> AKS	<b>Ir</b> Iron.io	67 Os
67 Os	<b>Cd</b> AWS CodeDeploy	<b>Mg</b> Moogsoft	68 Pd
68 Pd	<b>Ec</b> ElectricCloud	<b>Ir</b> Iron.io	69 Os
69 Os	<b>Ra</b> Rancher	<b>Mg</b> Moogsoft	70 Os
70 Os	<b>Rk</b> Rkt	<b>Sp</b> Spinnaker	71 Pd
71 Pd	<b>Sp</b> Spinnaker	<b>Ir</b> Iron.io	72 Pd
72 Pd	<b>Sp</b> Spinnaker	<b>Mg</b> Moogsoft	
73 Fm	<b>Bb</b> BitBucket	<b>Pf</b> Perforce	74 En
74 En	<b>Cr</b> Circle CI	<b>Cb</b> AWS CodeBuild	75 Fm
75 Fm	<b>Cu</b> Cucumber	<b>Mc</b> Mocha	76 Pd
76 Pd	<b>Lo</b> Locust.io	<b>Mf</b> Micro Focus UFT	77 Fr
77 Fr	<b>Sa</b> Salt	<b>Ce</b> CFEngine	78 Os
78 Os	<b>Eb</b> ElasticBox	<b>Ca</b> CA Automic	79 Os
79 Os	<b>De</b> Docker Enterprise	<b>Ac</b> AWS ECS	80 En
80 En	<b>Ae</b> AWS ECS	<b>Cf</b> Codefresh	81 Os
81 Os	<b>Hm</b> Helm	<b>Hm</b> Helm	82 Os
82 Os	<b>De</b> Docker Enterprise	<b>Cf</b> Codefresh	83 En
83 En	<b>Ae</b> AWS ECS	<b>Hm</b> Helm	84 En
84 En	<b>De</b> Docker Enterprise	<b>Cf</b> Codefresh	85 En
85 En	<b>Ae</b> AWS ECS	<b>Hm</b> Helm	86 Pd
86 Pd	<b>De</b> Docker Enterprise	<b>Cf</b> Codefresh	87 Fm
87 Fm	<b>Ae</b> AWS ECS	<b>Hm</b> Helm	88 Os
88 Os	<b>Cf</b> Codefresh	<b>Hm</b> Helm	89 Os
89 Os	<b>Aw</b> Apache OpenWhisk	<b>Ls</b> Logstash	90 Os
90 Os	<b>Aw</b> Apache OpenWhisk	<b>Ls</b> Logstash	



Follow @xebialabs

91 En	92 Os	93 Fm	94 En	95 En	96 Fm	97 Os	98 Os	99 Os	100 En	101 En	102 En	103 En	104 Os	105 Os
<b>XLi</b> XebiaLabs XL Impact	<b>Ki</b> Kibana	<b>Nr</b> New Relic	<b>Dt</b> Dynatrace	<b>Dd</b> Datadog	<b>Ad</b> AppDynamics	<b>Ei</b> ElasticSearch	<b>Ni</b> Nagios	<b>Zb</b> Zabbix	<b>Zn</b> Zenoss	<b>Cx</b> Checkmarx SAST	<b>Sg</b> Signal Sciences	<b>Bd</b> BlackDuck	<b>Sr</b> SonarQube	<b>Hv</b> HashiCorp Vault
106 En	107 Pd	108 Fm	109 Fm	110 Fm	111 En	112 En	113 En	114 Pd	115 Pd	116 Os	117 Fm	118 En	119 En	120 En
<b>Sw</b> ServiceNow	<b>Jr</b> Jira	<b>Tl</b> Trello	<b>Sk</b> Slack	<b>St</b> Stride	<b>Cn</b> CollabNet VersionOne	<b>Ry</b> Remedy	<b>Ac</b> Agile Central	<b>Og</b> OpsGenie	<b>Pd</b> Pagerduty	<b>Sn</b> Snort	<b>Tw</b> Tripwire	<b>Ck</b> CyberArk	<b>Vc</b> Veracode	<b>Ff</b> Fortify SCA

<https://softwarezen.me/campaigns/warmup/gravacao/>

A má implementação ou excesso de customização pode destruir uma ótima ferramenta...

**Incentive a opinião das equipes no chão de fábrica!**  
**Vá ao gemba !**



NÃO EXISTEM MAIS TESTES MANUAIS

O CÓDIGO SÓ VAI PARA PRODUÇÃO SE ESTIVER COM ALTA QUALIDADE, VALIDADO AUTOMATICAMENTE

NINGUÉM TEM ACESSO ÀS MÁQUINAS DE PRODUÇÃO

TODA INSTALAÇÃO É AUTOMATIZADA

OS DESENVOLVEDORES POSSUEM UM AMBIENTE IDÊNTICO AO PRODUÇÃO EM SUAS MÁQUINAS LOCAIS

OS ANALISTAS DE INFRAESTRUTURA SE TORNAM DESENVOLVEDORES DE CÓDIGO

OS DESENVOLVEDORES SE TORNAM ANALISTAS DE INFRAESTRUTURA

TODOS TEM VISIBILIDADE POR TODO O AMBIENTE

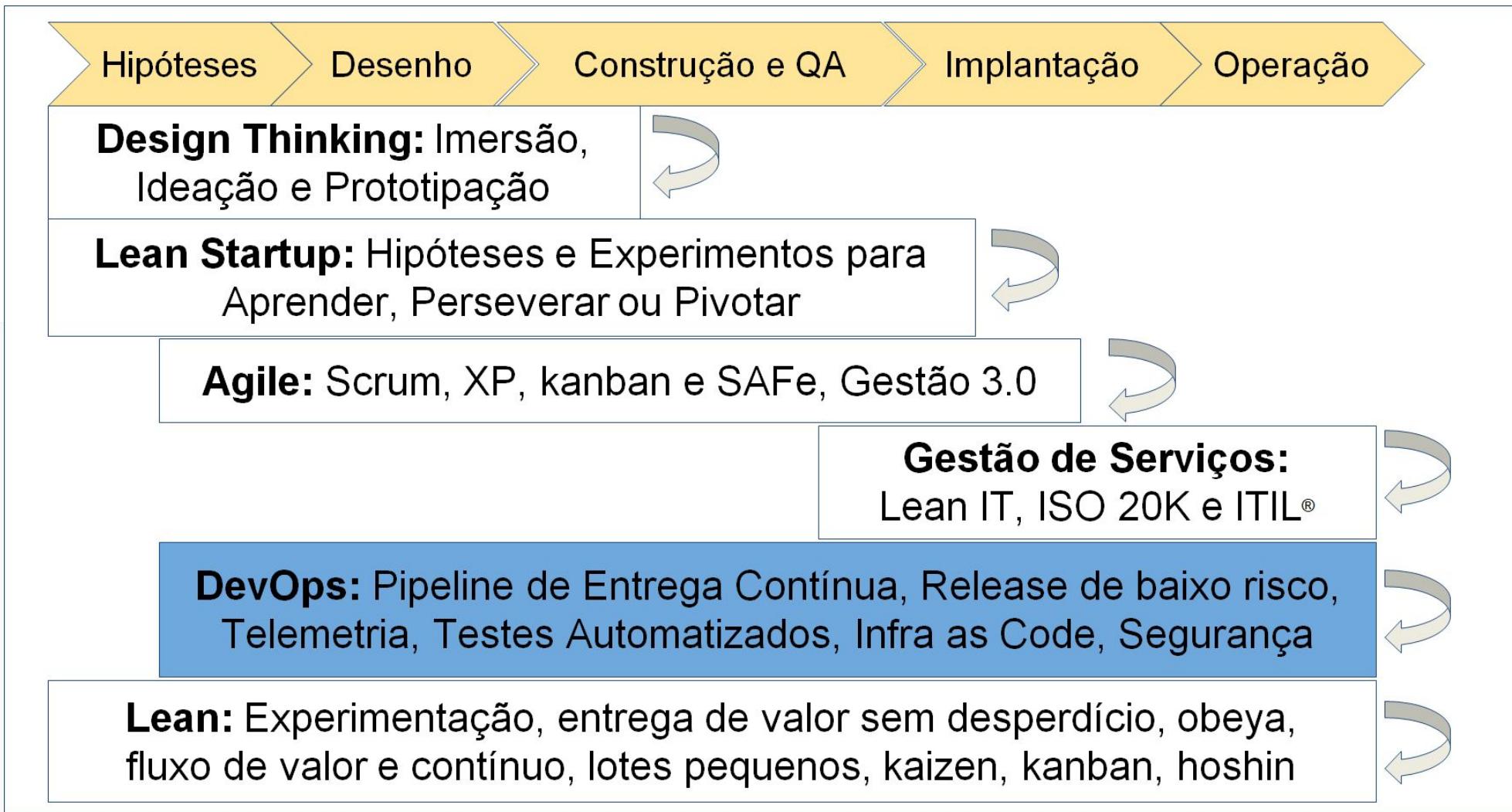
NÃO EXISTEM BARREIRAS ENTRE EQUIPES DE TI, TODOS SÃO DESENVOLVEDORES DE PROJETOS

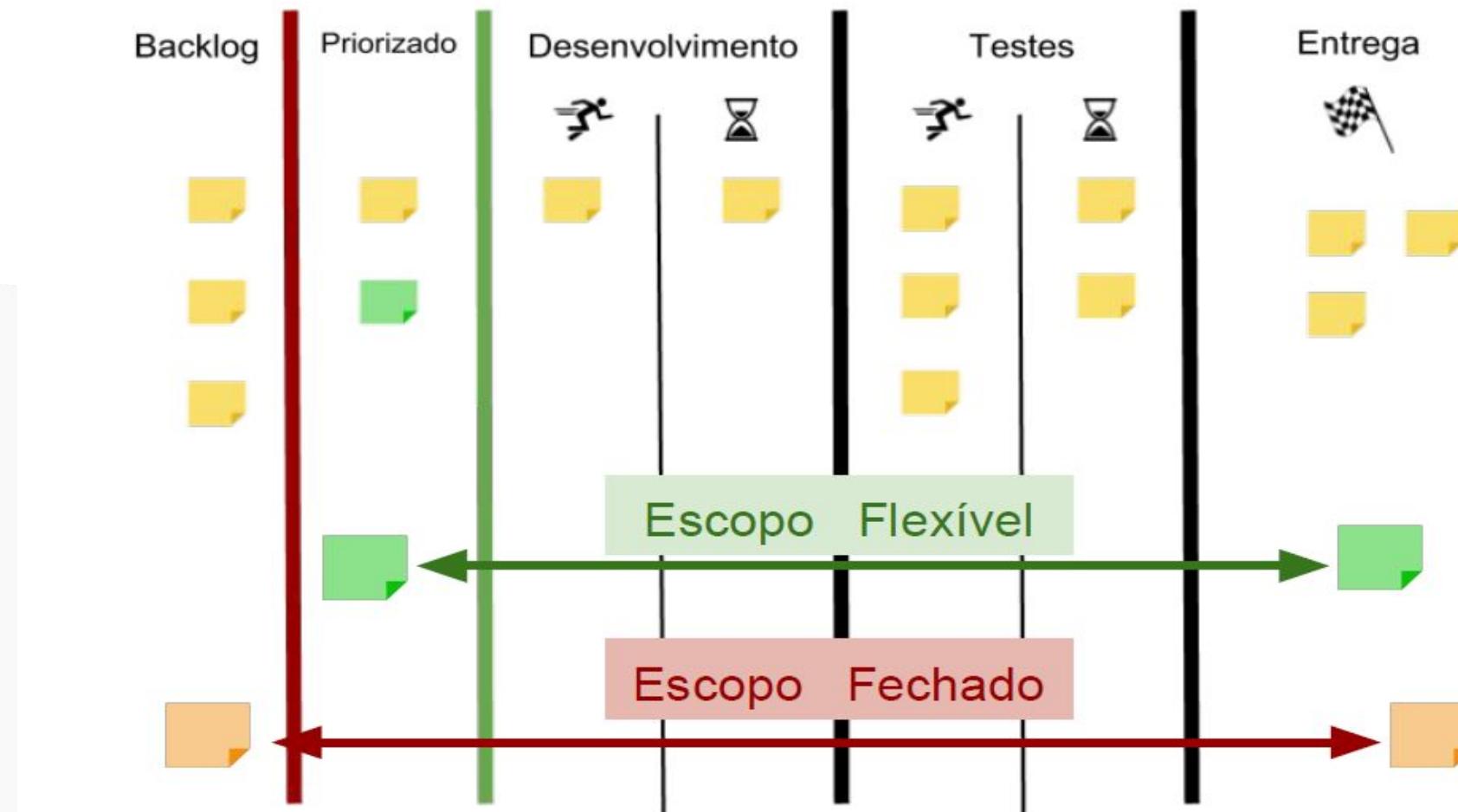
NINGUÉM TEM CULPA POR UM ERRO DE PRODUÇÃO

TODO O AMBIENTE DE PRODUÇÃO É RECONSTRUÍDO EM MINUTOS

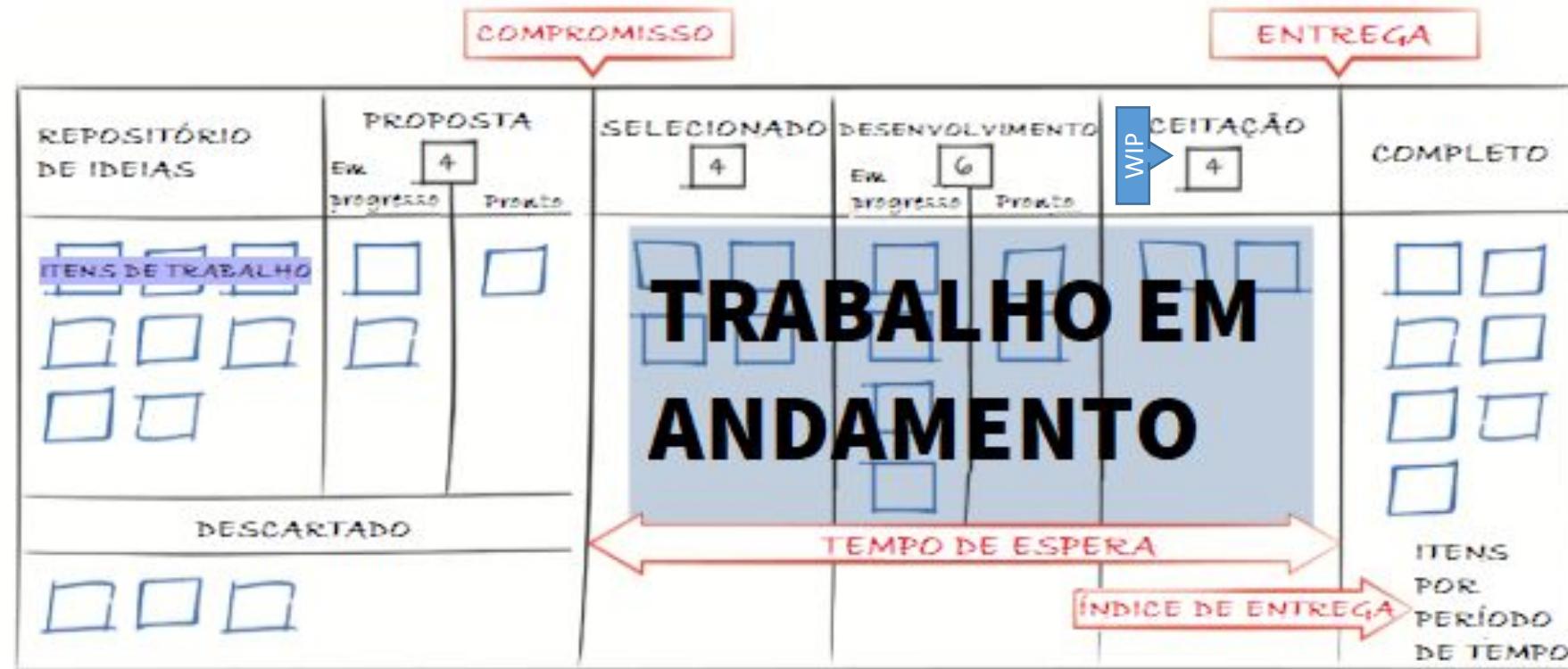
ESCALONAMENTO DE AMBIENTE É FEITO EM MINUTOS

Benefício	
Aumento da frequência de deployments	Código sempre entregável
Aumento da colaboração entre departamentos	Visibilidade do Pipeline
Redução de tempo construindo e mantendo aplicações	Sistema sempre validado
Aumento de qualidade e performance de aplicações	Arquitetura orientada a padrões
Redução do “time-to-market”	Equipes trabalhando de forma igual
Redução de custos de desenvolvimento	Desenvolvimento otimizado
Redução de pessoas trabalhando no processo de deploy	Deploy contínuo
Paralelização de desenvolvimento	Separação do desenvolvimento
Facilidade na realocação de recursos entre projetos	Projetos padrões
Redução do tempo de treinamento de novos funcionários	Ambientes prontos
.... entre outros ...	



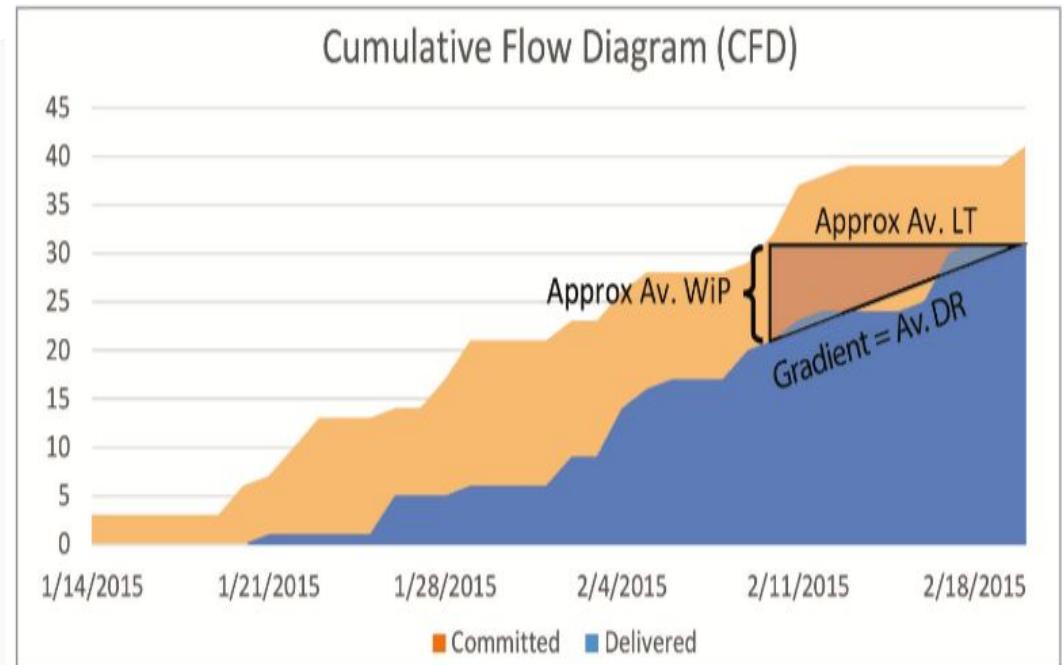


## KANBAN – Produção Puxada



### Framework Cynefin

- CFD
- Dependências externas ao time - Gargalo
- Eficiência do Fluxo
- Leadtime (Lembrar do ponto de compromisso)
- Tamanho dos itens (P/M/G)
- Touch time (tempo que eu levo efetivamente desenvolvendo)
- WIP (Lei do Little – limitar tarefas em andamento)
- Waiting time (tempo de espera)



## Dívida técnica (Débito técnico)



- ✓ **Representa o resultado das decisões do curto prazo que geram problemas que se tornam cada vez mais difíceis de resolver com o passar do tempo**
- ✓ Reduz a performance da equipe e aumenta o TCO
- ✓ Os objetivos conflitantes das áreas Dev e Ops contribuem para o aumento do débito técnico

## Algumas causas da dívida técnica

- ✓ Ausência de testes
- ✓ Requisitos não claros
- ✓ Documentação pobre
- ✓ Pouco foco em refatoração
- ✓ Equipes não colaborativas

# Princípios Lean



**Fluxo de valor:** É o processo que concretiza uma necessidade de negócio em um produto ou serviço para entrega de valor ao cliente.

**Mapeamento do fluxo de valor:** Visa entender como o processo funciona com foco na entrega de valor ao cliente e identifica gargalos ou desperdícios.

**Gembá:** É o local onde as coisas acontecem e todos deveriam ir ao gembá com frequência para conhecer o “chão de fábrica” e evitar suposições sem dados e fatos.

**Obeya:** Também conhecida nas organizações como “sala de guerra”, o objetivo é facilitar a gestão visual e a coordenação para solução de problemas sem os entraves das estruturas organizações clássicas.

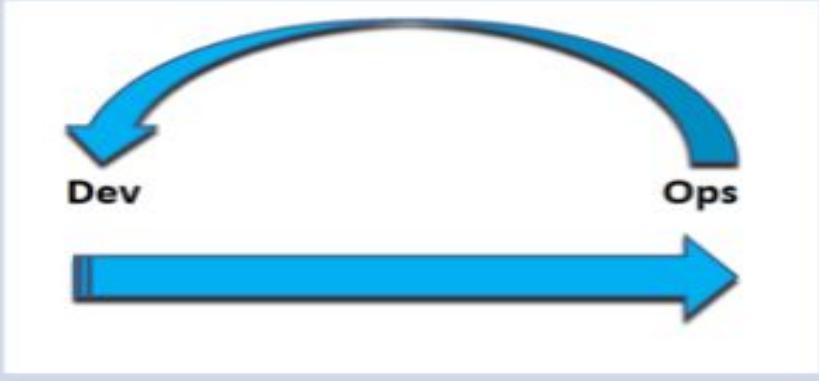
### Desperdício

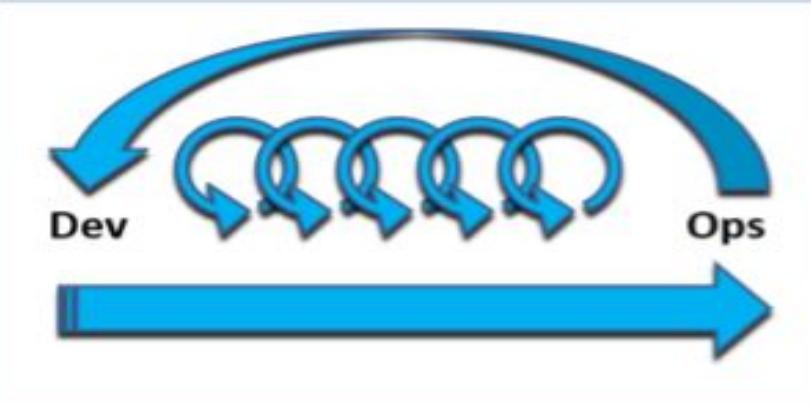
Existem 8 tipos de desperdícios identificados no Lean:

- Defeito e retrabalho: Desfazer, refazer algo.
- Movimentação: Caminhadas, deslocamentos, viagens.
- Espera: Pessoas aguardando informações, materiais ou outras equipes.
- Transporte: Transferências desnecessárias de materiais ou informações.
- Estoques: Informações ou materiais sem uso.
- Processamento: Etapa redundante ou desnecessária.
- Desconexão ou superprodução: Fluxo deficiente ou falta de sincronismo entre etapas (antes ou depois do necessário).
- Conhecimento: Não aproveitar as habilidades das pessoas adequadamente.

**Corda de Andon:** Dispositivo que existe nas fábricas da Toyota para interromper a linha de produção quando é encontrado algum defeito nos produtos. O objetivo é aglomerar imediatamente todas as equipes e líderes que podem ajudar a resolver o problema na origem, podendo mobilizar os executivos e a alta administração.

Objetivo da Primeira Maneira	Princípios e Práticas
<p>Acelerar o fluxo dos desenvolvedores (Esquerda) para operação e clientes (Direita)</p> <p><b>Empresa</b>                           <b>Cliente</b></p> <p><b>Dev</b>  <b>Ops</b></p>	<ol style="list-style-type: none"><li>1. Tornar o trabalho visível</li><li>2. Reduzir o tamanho dos lotes e intervalos</li><li>3. Aplicar teoria das restrições e otimizar o fluxo</li><li>4. Remover desperdícios e foco no cliente</li><li>5. Reduzir o número de transferências (handoff)</li><li>6. Incorporar qualidade na origem</li><li>7. Limitar o trabalho em andamento (WIP)</li><li>8. Infraestrutura como código e self service</li><li>9. Integração, entrega e implantação contínua</li><li>10. Testes automatizados e TDD</li><li>11. Arquitetura e releases de baixo risco</li></ol>

Objetivo da Segunda Maneira	Princípios e Práticas
<p>Rápido feedback em todos os estágios do fluxo de valor (Direita para a Esquerda)</p>  <p>The diagram illustrates a flow from right to left. A thick blue arrow points from the word 'Ops' on the right towards the word 'Dev' on the left. Above this main arrow, a curved blue arrow loops back from 'Dev' back towards 'Ops', representing a feedback mechanism.</p>	<ol style="list-style-type: none"><li>1. Ver problemas quando ocorrem (“ir ao gemba”)</li><li>2. Aglomerar quando problema aparece (Andon)</li><li>3. Qualidade próxima da fonte (menos aprovações)</li><li>4. Telemetria self service e irradiadores de informação disponível para todos</li><li>5. Desenvolvimento por hipóteses e Testes A/B</li><li>6. Equipes Dev e Ops compartilham o trabalho diário e plantões de suporte 24 x7</li><li>7. Revisão de código usando as técnicas: Programação em pares, sobre os ombros, divulgação por email, assistida por ferramentas</li></ol>

Objetivo da Terceira Maneira	Princípios e Práticas
<p>Cultura de alta confiança que permite correr riscos e potencializar o aprendizado contínuo</p> 	<ol style="list-style-type: none"><li>1. Cultura justa e segura para aprender e evoluir com os erros</li><li>2. Injeção de falhas na produção para aumentar resiliência</li><li>3. Converter descobertas locais em melhorias globais</li><li>4. Reservar tempo para melhorar o trabalho diário (Kata e blitz de melhoria)</li><li>5. Reunião post-mortem sem culpa</li><li>6. Instituir dias de jogos para ensaiar falhas</li><li>7. Difundir conhecimento usando testes automatizados como documentação</li></ol>

## Sistemas de registro e engajamento



Sistema	Tipo Bimodal (Gartner)	Ritmo de mudanças
Registro	Tipo 1: Fazer direito com foco em estabilidade e confiabilidade	Mais lento e costuma ter requisitos normativos e conformidade
Engajamento	Tipo 2: Fazer rápido com foco em flexibilidade e inovação	Mais rápido e permite experimentos em ambientes incertos

# Papéis DevOps

Papel	Responsabilidades
<b>Dono do Produto</b>	<b>A voz interna da empresa que define as funcionalidades</b>
Desenvolvimento	Criar as funcionalidades dos aplicativos
<b>QA</b>	<b>Realiza loops de feedback para garantir qualidade</b>
Operações	Manter o ambiente de produção e alcance do SLA
Segurança	Manter a segurança de sistemas e dados
Gerente de release	Administrar e coordenar a implantação em produção
Gerente fluxo	Garantir que alcance os requisitos do cliente

## Lei de Conway: O design do sistema sempre é uma cópia da estrutura de comunicação da organização



Tipo de Estrutura	Características	Resultados
Organizações hierarquizadas, comando e controle	Pouca colaboração e comunicação ineficaz	Sistemas centralizados, processos engessados e lentidão de resposta ao mercado e clientes
<b>Organizações flexíveis e equipes com grande autonomia</b>	<b>Forte colaboração e objetivos compartilhados</b>	<b>Sistemas modulares, fluxo de valor efetivo e adaptabilidade para atender mercado e clientes</b>

# Etapas para a Transformação DevOps



Etapa	Características importantes
1. Equipe dedicada	Coloque em um espaço físico separado os melhores generalistas que tenham uma relação respeitosa e de longa data
2. Meta SMART	Equipes tem autonomia para combinar meta compartilhada
3. Sprints curtos	Entregas constantes e adaptabilidade do plano
4. Requisitos não funcionais	Reservar pelo menos 20% do ciclo de melhoria para reduzir a dívida técnica
5. Visibilidade	Disponibilizar informação atual com evolução das melhorias
6. Ferramentas	Backlog unificado entre todas as equipes gera mais empatia

Gestão Funcional (Clássica)	Gestão do Fluxo de Valor
Informação fica restrita a poucos (silos)	Informações compartilhadas (interfaces)
Foco maior nos departamentos	Foco nos objetivos dos processos
Comunicação é vertical	Comunicação é transversal
Metas departamentais (chefe/subordinado)	Objetivos organizacionais
Pouco foco no cliente dos processos	Foco total nos clientes dos processos
Delegação de autoridade limitada	Alto grau de <i>empowerment</i>
Visão restrita à tarefa departamental	Visão macro e organizacional
Processos podem não agregar valor	Melhoria contínua nos processos
Estruturado nas habilidades e poderes	Estruturado no modo de fazer o trabalho

# Integrando Ops no trabalho diário de Dev



- ✓ Participar das reuniões diárias: O que foi feito ontem, o que será feito hoje e os impedimentos
  - ✓ Participar da retrospectiva para aumentar aprendizado: O que teve êxito, o que pode melhorar e como aplicar as lições nos próximos sprints
  - ✓ Deixar visível o trabalho de Ops necessário para que a aplicação funcione em produção
- Livro Jornada DevOps: MUNIZ; SANTOS; IRIGOYEN; MOUTINHO (Brasport, 2019)



- Primeira Maneira



# Bootcamp

# Arquiteto de Soluções - Cloud

Módulo 4 - Soluções para Desenvolvimento

Capítulo 2. Primeira Maneira

ANALÍA IRIGOYEN



# Arquiteto de Soluções

---

Aula 2.1. Primeira Maneira

ANALIA IRIGOYEN



É o processo de automação do fluxo de valor que leva o software do controle de versão até o ambiente de produção (cliente\$)



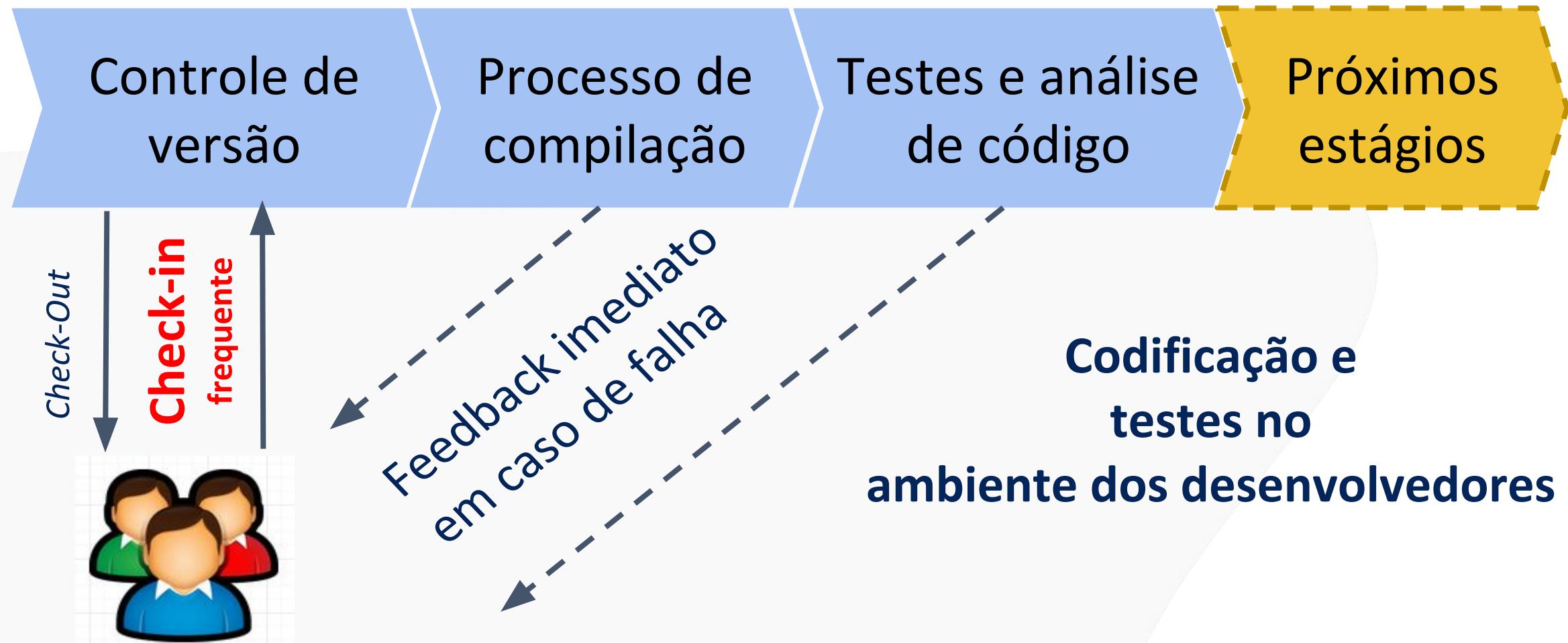
Seu objetivo principal é fornecer feedback rápido a todos no fluxo de valor sobre o status das mudanças (principalmente para equipe Dev)



Colabora com a correção imediata quando ocorre problema

Livro Jornada DevOps: MUNIZ; SANTOS; IRIGOYEN; MOUTINHO (Brasport, 2019)

## Resumo da Integração contínua



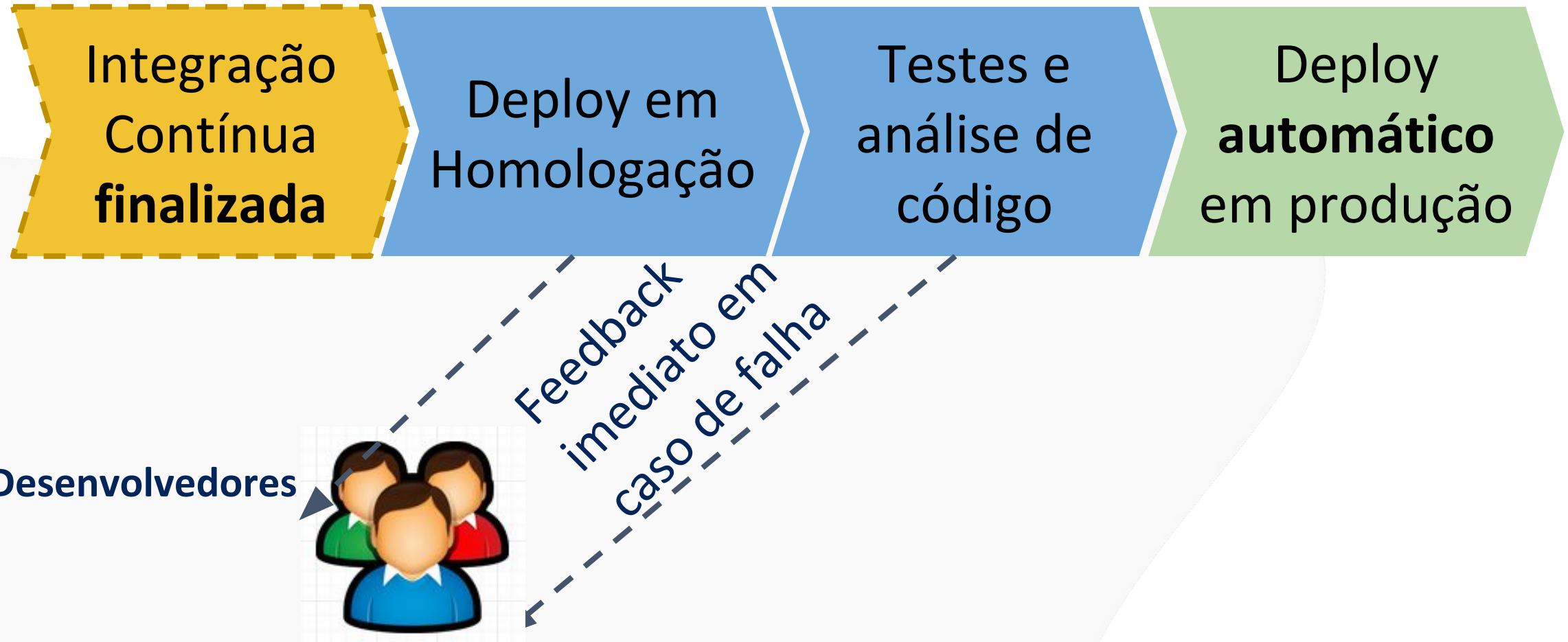
## Resumo da Entrega contínua

IGTI



## Resumo da Implantação contínua

IGTi



# Infraestrutura ágil



**É a aplicação dos princípios ágeis na infraestrutura**

**Foco principal é a infraestrutura como código:**

Gerência e provisão de ambientes automaticamente

Tudo fica centralizada no controle de versões

Nuvem/virtualização é a base e existem ferramentas para otimizar (Puppet, Terraform, Chef, Ansible, etc.)

## Infraestrutura como código (IaC)



- ✓ Código e configurações dentro do controle de versão
- ✓ Criação automatizada e sob demanda (self-service) em todos os ambientes, evitando trabalho manual
- ✓ Todos os estágios do fluxo de valor com ambientes iguais ou semelhantes ao de produção
- ✓ **Infraestrutura imutável:** Foco em recriar todo o ambiente de produção de forma rápida em vez de realizar alterações

## IaaS, SaaS e PaaS



✓ IaaS é a mais simples

1. Oferece infraestrutura de TI automatizada e escalonável- armazenamento, hospedagem, redes - de seus próprios servidores globais, cobrando apenas pelo o que o usuário consome.
2. Não precisa adquirir licença de software ou servidores e alocar recursos conforme a necessidade.
3. Dominado pela Amazon (AWS – 38% do mercado), Microsoft (18%) , Google (9%) e Alibaba (6%)

## IaaS, SaaS e PaaS



PaaS é mais difícil de ser definido

1. Oferece os conceitos básicos de IaaS.
2. Além disso, oferecem ferramentas e recursos para desenvolver e gerenciar aplicativos com segurança sem precisar se preocupar com a infraestrutura: Sistema Operacional, Compilação testes.
3. Exemplo: Os servidores que hospedam sites são exemplos de PaaS.
4. A solução Azure para desenvolvimento de software: Azure DevOps +

## IaaS, SaaS e PaaS



✓ SaaS é mais difícil de ser definido

1. O Software como Serviço (SaaS) é o local onde um software é hospedado por terceiros e pode ser acessado pela web, geralmente bastando um login
2. Plano de assinatura e utiliza os programas necessários para os negócios.
3. É interessante para o uso de aplicativos específicos.

## IaaS, SaaS e PaaS



Fonte: <https://azure.microsoft.com/>

# PaaS AzureDevOps + Azure Services



O Azure DevOps gerencia o processo de desenvolvimento.

[Azure DevOps](#)



Os agentes de build e lançamento do Microsoft Release Management implantam o modelo do Azure Resource Manager e o código associado nos vários ambientes.

[Microsoft  
Release  
Management](#)



Os grupos de recursos do AzureDevOps são usados para definir todos os serviços necessários para implantar a solução em um ambiente de desenvolvimento e teste ou de produção.

[Grupos de  
recursos](#)



Um aplicativo Web executa o site e é implantado em todos os ambientes. Slots de preparo são usados para alternar entre versões de pré-produção e de produção.

[Aplicativos Web](#)

Ciclo de Desenvolvimento

Criação de Releases

Criação de Pipelines

Criação de Ambiente  
para hospedar um site

## Vantagens do PaaS



Ao fornecer infraestrutura como serviço, PaaS oferece as mesmas vantagens que o IaaS. Seus recursos adicionais – middleware, ferramentas de desenvolvimento e outras ferramentas de negócios – dão ainda mais vantagens:



Reduza o tempo de codificação. As ferramentas de desenvolvimento PaaS podem reduzir o tempo levado para codificar novos aplicativos com componentes de aplicativos pré-codificados inseridos na plataforma, como fluxo de trabalho, serviços de diretório, recursos de segurança, pesquisa etc.



Adicione funcionalidades de desenvolvimento sem adicionar funcionários. Componentes da Plataforma como Serviço dão à sua equipe de desenvolvimento novas funcionalidades sem precisar adicionar funcionários com as habilidades necessárias.

Fonte: <https://azure.microsoft.com/>

Desenvolvimento simplificado para diversas plataformas, incluindo móveis. Alguns provedores

## Vantagens do PaaS



Use ferramentas sofisticadas de forma acessível. Um modelo pago conforme o uso permite que pessoas ou organizações usem software de desenvolvimento sofisticado e ferramentas de análise e business intelligence que não poderiam comprar por completo.



Suporte a equipes de desenvolvimento distribuído geograficamente. Como o ambiente de desenvolvimento é acessado pela Internet, equipes de desenvolvimento podem trabalhar em conjunto em problemas mesmo quando os membros da equipe estiverem em locais remotos.



Gerencie com eficácia o ciclo de vida do aplicativo. PaaS fornece todas as funcionalidades que você precisa para dar suporte ao ciclo de vida completo do aplicativo Web: compilação, teste, implantação, gerenciamento e atualizações no mesmo ambiente integrado.

# Serverless



-  Ainda existem servidores apesar do nome 😊
-  O desenvolvedor não se preocupa em configurar ou outros aspectos de infra onde a sua aplicação vai rodar.
-  Podem ser dividido em: Backend as a Service (BaaS) and Function as a Service (FaaS).
-  Você paga pelo uso (cada vez que a função é executada)

# Serverless



✓ Serverless é um framework web e open-source - Node.js.

- Inicialmente era destinado exclusivamente para a criação de aplicações para o AWS Lambda, a plataforma da Amazon Web Services de serverless cloud computing.
- Agora está compatibilizado com outros fornecedores de Cloud: Microsoft Azure, IBM BlueMix, Google Cloud, Oracle Cloud, entre outros.

Cuidado: ao usar não se esqueça de que é diferente de um desenvolvimento para Cloud Tradicional

# Serverless



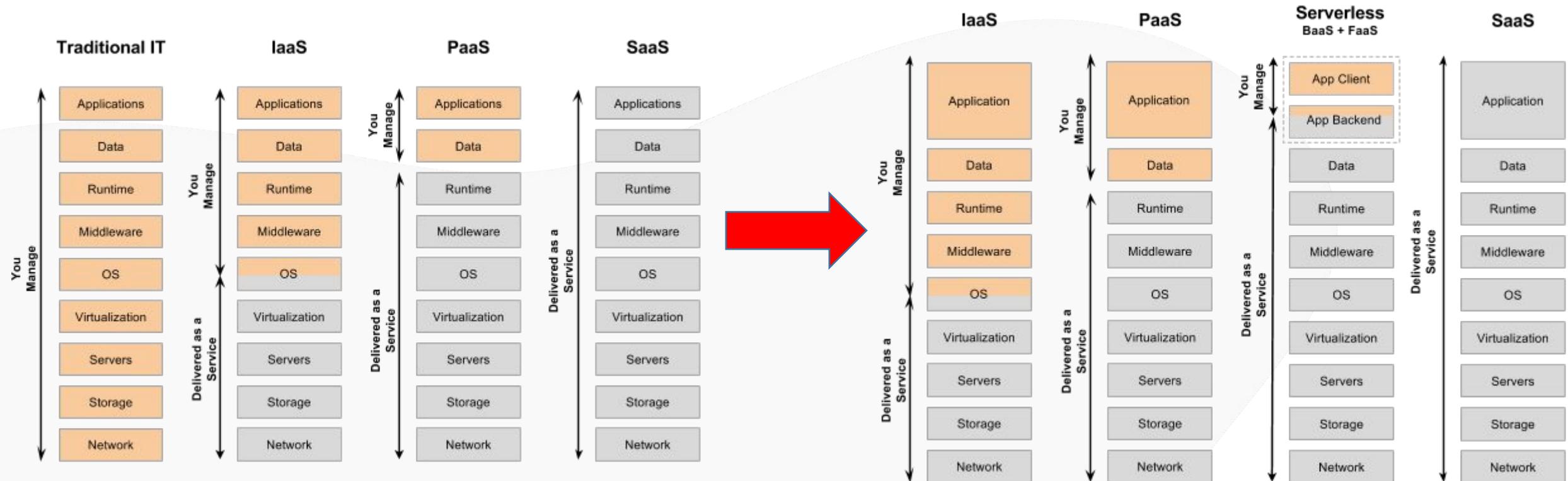
Up é um framework web e open-source - Node.js.

- Não tem funcionalidades.
- Adapta seu código para rodar na arquitetura serverless.

Cuidado: seu código não poderá utilizar as funcionalidades especiais dos provedores serverless.

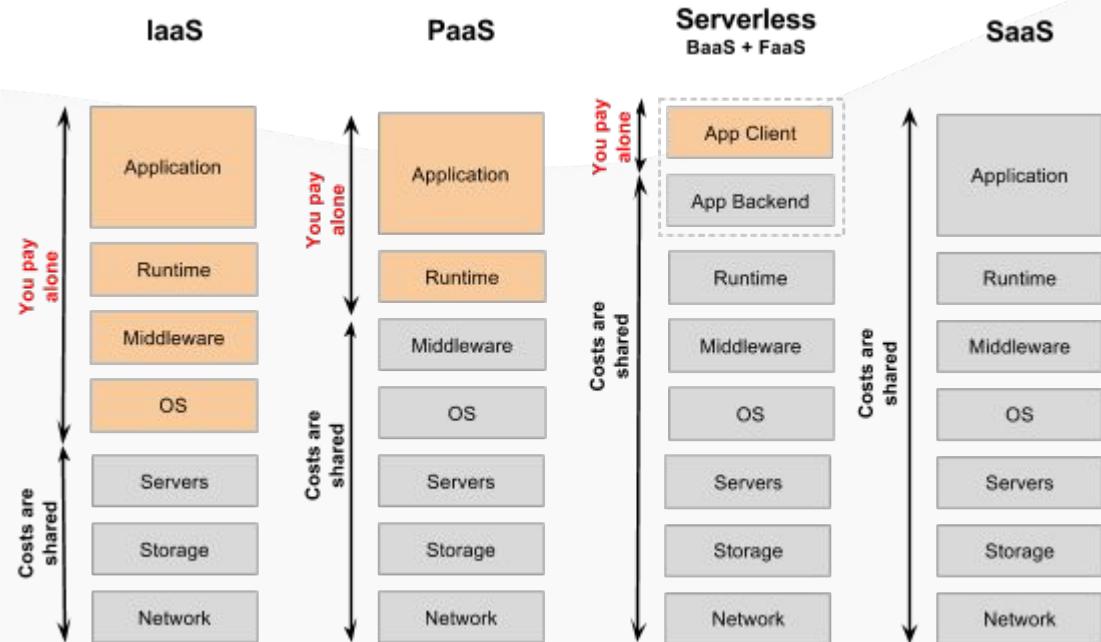
# Serverless –

## Backend as a Service (BaaS) and Function as a Service (FaaS).



Fonte: <https://specify.io/concepts/serverless-baas-faas>

# Serverless – Backend as a Service (BaaS) and Function as a Service (FaaS).



Quando você está usando FaaS, você compartilha o tempo de execução com outras pessoas. Por exemplo: quando sua "função" é escrita em JavaScript, esta parte do código será executada no mesmo servidor node.js que as "funções" de outros usuários deste FaaS (isso pode ser diferente para alguns fornecedores).

Ao usar o BaaS, você compartilha o mesmo BaaS com outros usuários deste BaaS (isso pode ser diferente para alguns fornecedores).

Fonte: <https://specify.io/concepts/serverless-baas-faas>

## Adaptar a Definição de Pronto com uso de ambientes sendo criados sob demanda, dentro do controle de versão

- ✓ “Ao final de cada intervalo de desenvolvimento temos código integrado, testado, funcionando e que pode ser entregue,
- ✓ demonstrado em um ambiente do tipo produção”

## Adaptar a Definição de Pronto com uso de **ambientes sendo criados sob demanda, dentro do controle de versão e pipeline automatizado**

- ✓ “Ao final de cada intervalo de desenvolvimento temos código integrado, testado, funcionando e que pode ser entregue,
- ✓ **demonstrado em um ambiente do tipo produção,**
- ✓ **criado a partir do trunk com um processo de um clique e validado com testes automatizados”**

## Controle de versão



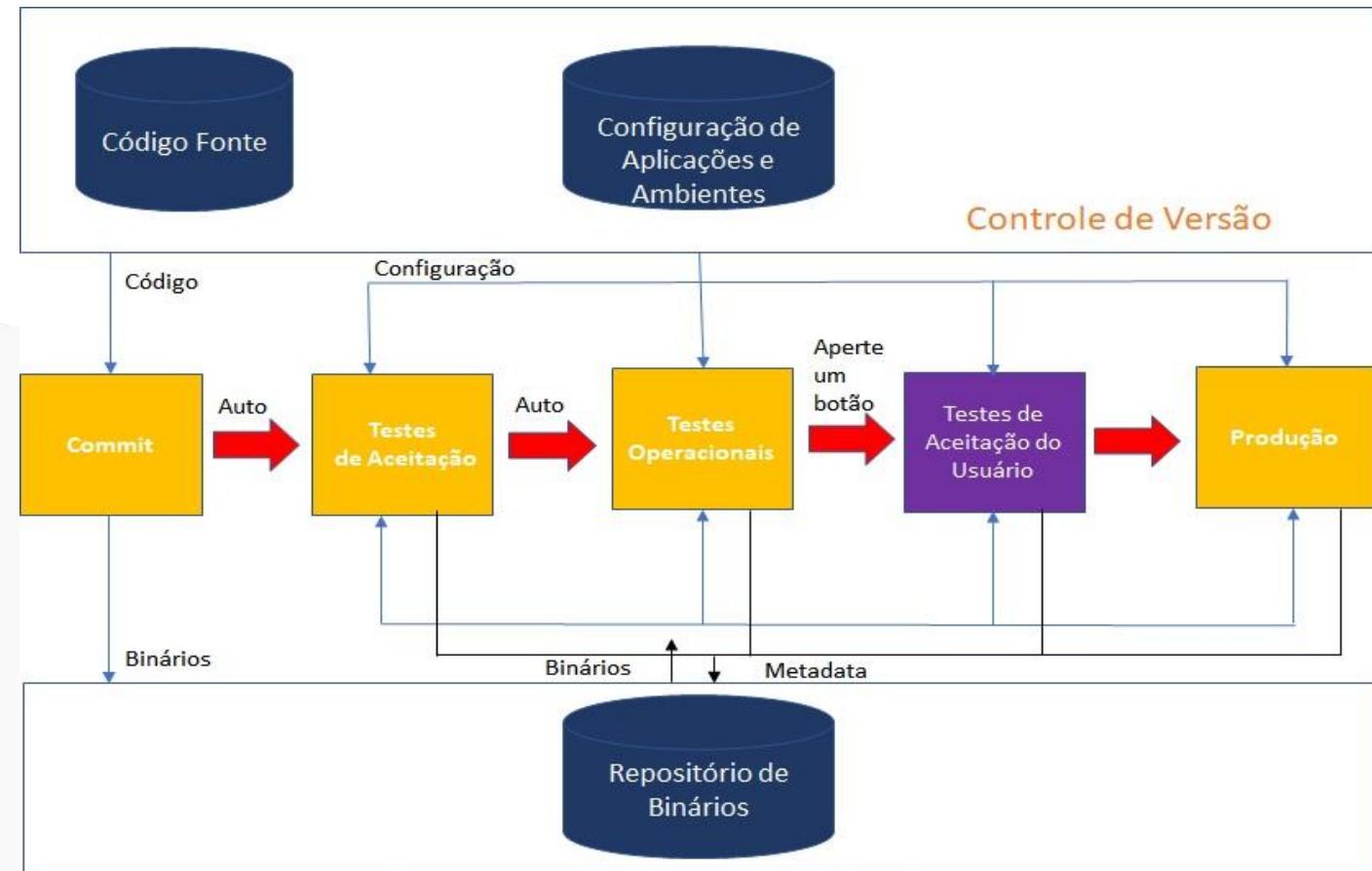
- ✓ Um sistema de controle de versão registra mudanças em arquivos ou conjuntos de arquivos
- ✓ Pode ser código-fonte, recursos ou outros documentos que façam parte do software
- ✓ Permite rastrear tudo que é realizado, assim como efetivar, comparar, mesclar e restaurar versões
- ✓ Importante incluir informações do ambiente que o software é executado e deve ser compartilhado por todos do fluxo de valor: QA, Dev, Ops, Security

## Testes automatizados

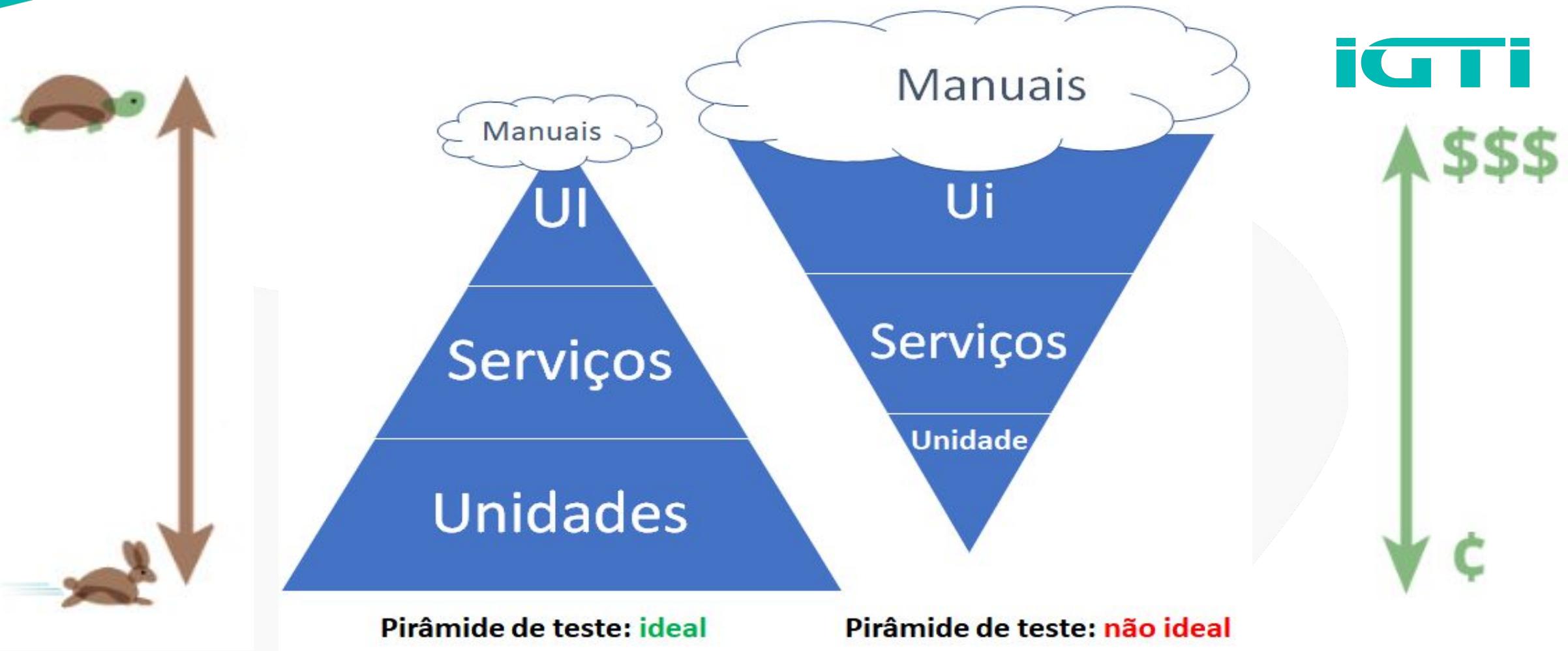


- ✓ Gary Gruver: “Sem testes automatizados, quanto mais código escrevemos, mais tempo e dinheiro são necessários para testá-lo”
- ✓ Testes apenas no final do desenvolvimento tiram a oportunidade de aprendizado e impactam a entrega de valor
- ✓ É fundamental adicionar testes contínuos no pipeline de implementação

# Testes automatizados



Livro Jornada DevOps: MUNIZ; SANTOS; IRIGOYEN; MOUTINHO  
(Brasport, 2019)



## Testes automatizados: Case Google



- ✓ 40 mil implantações por dia
- ✓ 50 mil construções por dia
- ✓ 120 mil conjuntos de testes
- ✓ 75 milhões de casos de testes executados diariamente
- ✓ Mais de 100 engenheiros com foco em testes (0,5% de P&D)

## Recomendações

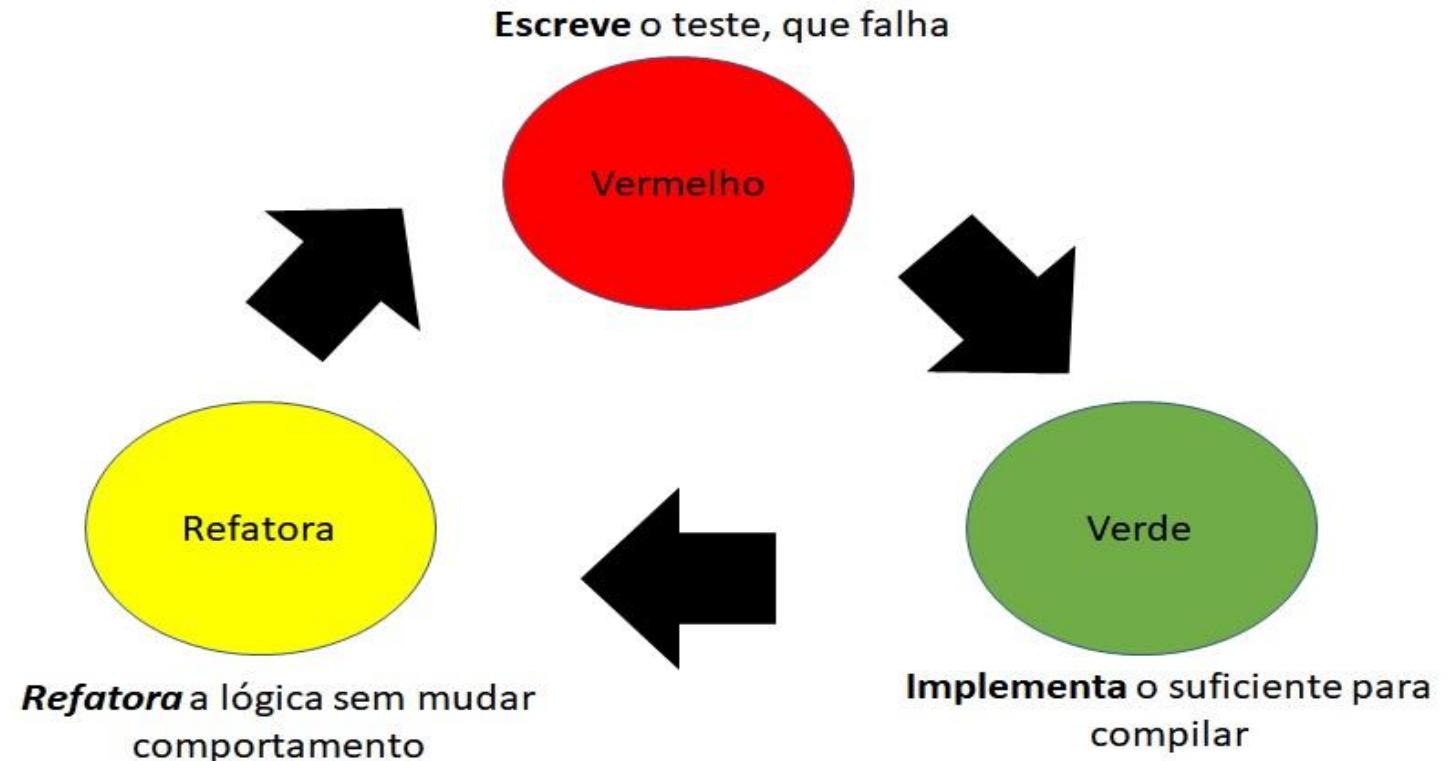
- ✓ Devemos **priorizar os testes unitários** que tem a execução mais rápida do que os testes de aceite e integração
- ✓ Além de economizar tempo, outro grande benefício é que os desenvolvedores recebem **feedback de imediato no teste unitário** e fica mais eficiente a correção na origem

## Desenvolvimento guiado por testes (TDD: Test-Driven Development)



- ✓ Técnica criada por Kent Beck na década de 90 e faz parte do XP (Programação extrema)
- ✓ Objetivo é escrever primeiro o teste automatizado que valide as falhas de comportamento e depois escreve-se o código para fazer os testes passarem
- ✓ É uma forma eficaz de escrever testes automatizados como parte do trabalho diário

# Sequência básica do TDD

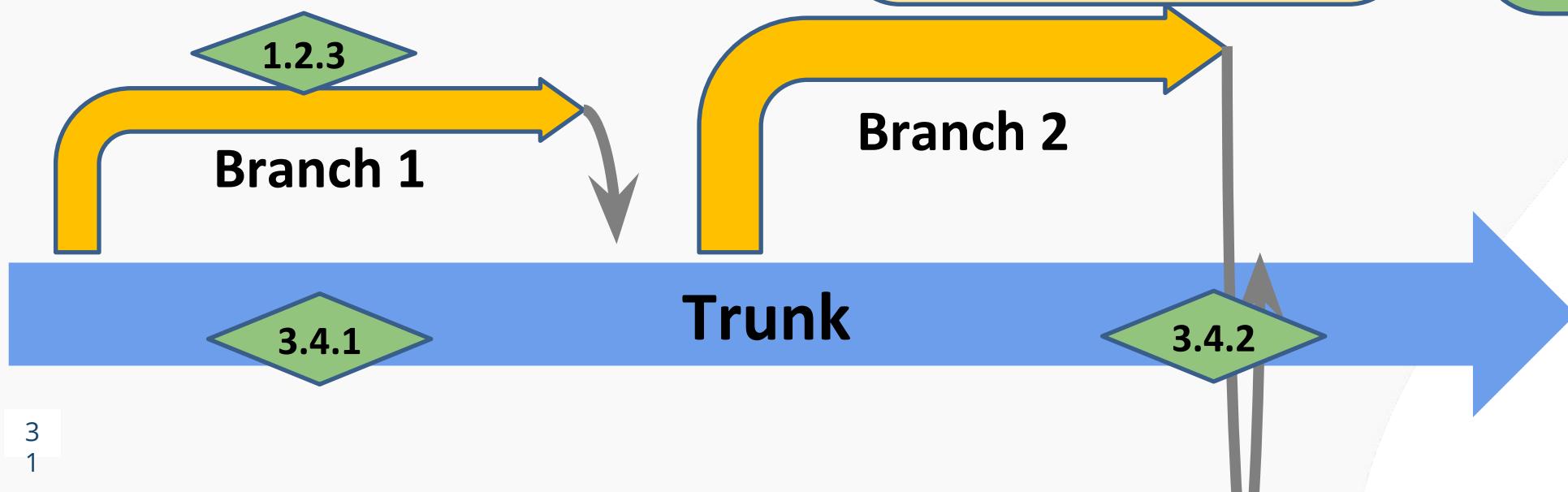


# Ambiente de Desenvolvimento

**TRUNK:** É a linha principal ou master que o projeto sempre deve se basear até o ambiente de produção

**BRANCH:** Cópia do código atual do trunk em dado momento

**TAG:** Marcador do estado do código em dado momento

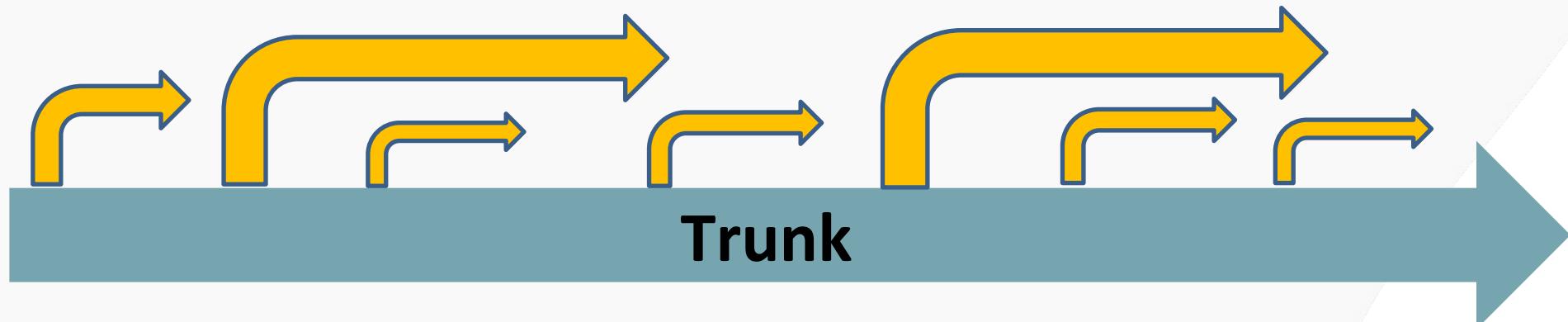


# O problema do merge (Integração)

As mudanças em cada branch devem ser integradas ao Trunk

Quanto mais tempo demora o merge, maior o conflito e complexidade

O uso de muitas branches costuma aumentar a dívida técnica



# Estratégias de branching (ramificação)

Estratégia	Vantagens	Desvantagens
Produtividade individual	Projeto privado que não atrapalha outras equipes	Merge do código ocorre no final do projeto <b>e gera muitos problemas</b>
Produtividade da equipe (Desenv. Baseado no trunk)	Fila única com todos trabalhando no trunk e commit frequente  Não há o estresse de merge no final do projeto	Difícil de implantar e cada commit pode quebrar o projeto inteiro  <b>Deve-se puxar a corda de andon para corrigir</b>

# Release de baixo risco



## Implantação (Deployment)

Instalação no respectivo ambiente para que seja possível a etapa de release (Teste, Produção)

## Liberação (Release)

Disponibiliza os recursos para os clientes ou conjunto de clientes após a implantação com sucesso e reduz riscos

# Categorias de liberação (Release)



## Baseado no Ambiente

Há 2 ou mais ambientes e apenas um fica ativo para os clientes (Ex.:平衡adores)

Azul verde (Blue-green) e Canário (Canary)

## Baseado no aplicativo

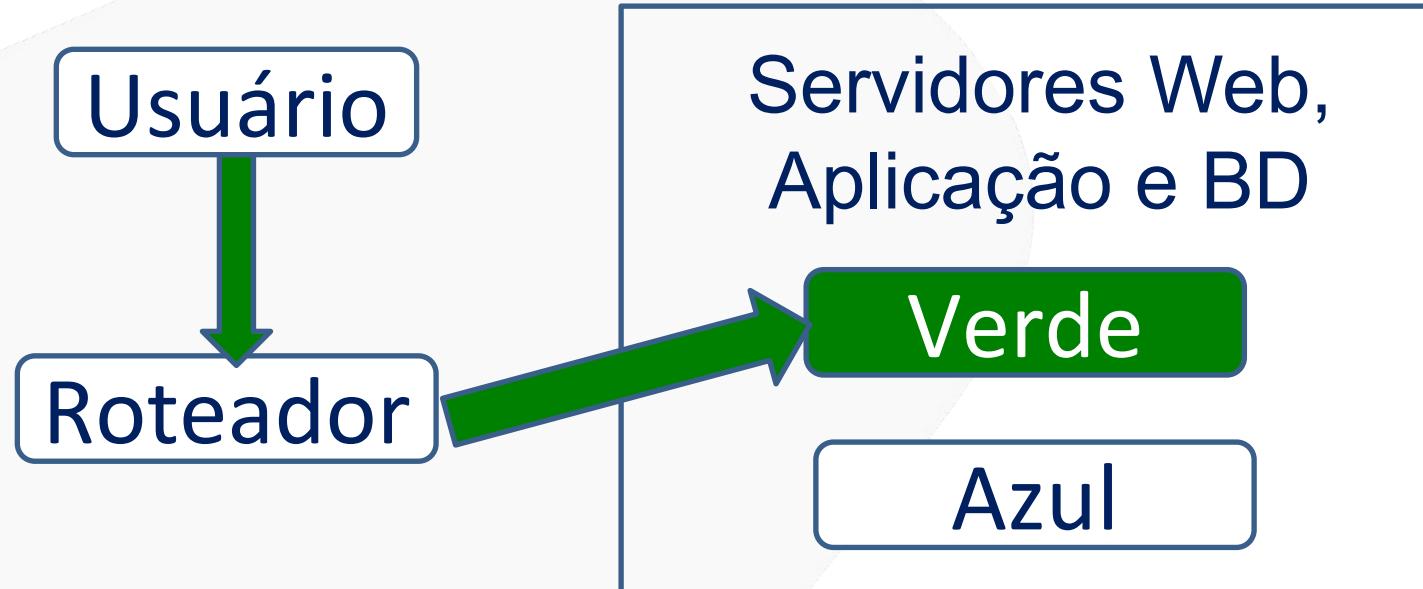
Novas funcionalidades de forma seletiva usando configurações simples (não precisa fazer deploy)

Alternância de recursos (Feature toggles) e Lançamento escuro



# Release Azul-Verde

- ✓ Existe um ambiente 100% ativo e outro fica em stand-by
- ✓ É o tipo mais simples de release de baixo risco, mas um problema que surge é como gerenciar BD com 2 versões do aplicativo em produção

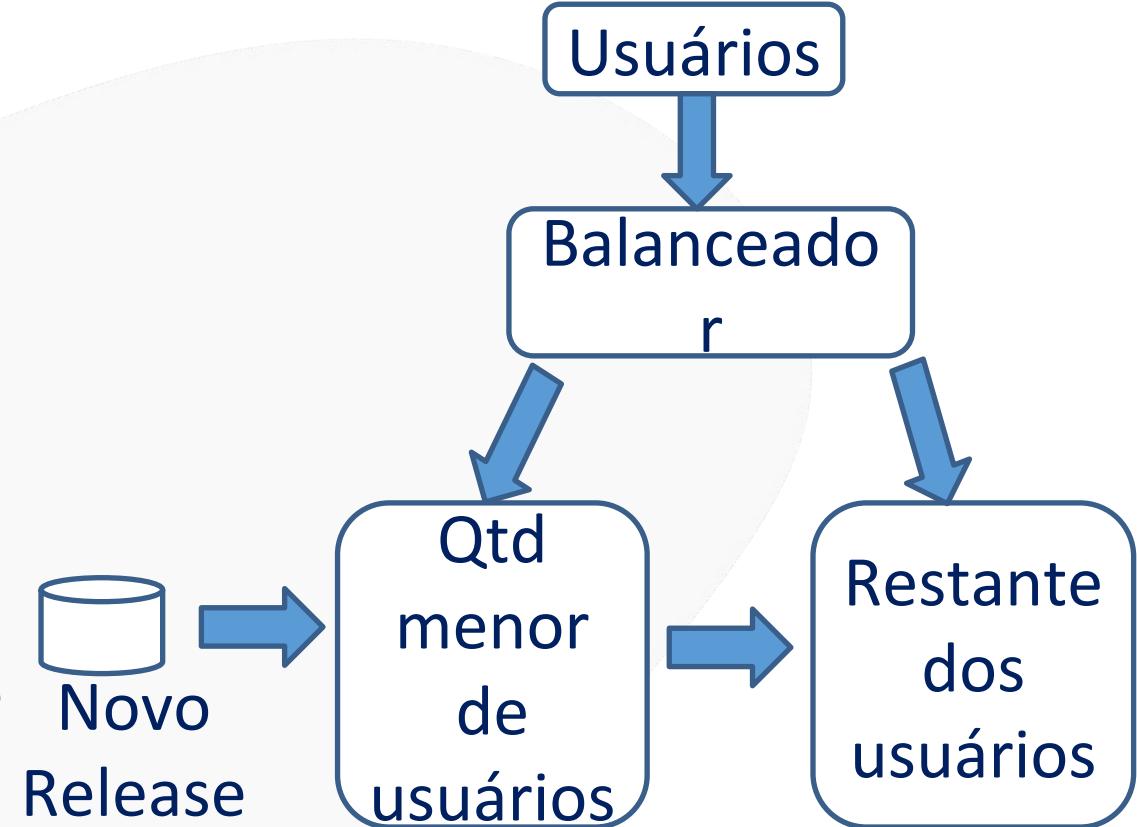


# Release Canário



✓ A implantação ocorre em um ambiente com poucos usuários (Ex.: 10%) e aumenta gradativamente

✓ O **sistema imune a cluster** é uma variação do Canário e permite reverter o deploy automaticamente quando ocorre falha em produção (Ex.: monitoramento indica maior tempo de resposta)



# Alternância de Recursos (Feature Toggle)



Habilita e desabilita recursos da aplicação de acordo com critérios definidos (Ex.: funcionário, localidade)

Usa instrução condicional através de parâmetro ou configuração sem necessidade de deploy

Permite o lançamento escuro:

Implantação de recursos na produção para avaliar resultados sem usuário perceber

## Benefícios: Release baseado no aplicativo iGTTi

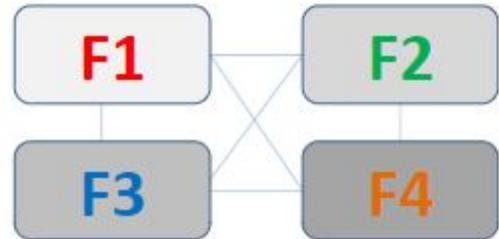
1. Permite reverter rápido quando ocorrem problemas
2. **Degradar o desempenho:**  
Reducir a qualidade do serviço quando existe o risco de falha em produção, (Netflix:  
aplicação no ar sem alguma funcionalidade)
3. Aumentar resiliência com arquitetura voltada a serviços:  
Permite ocultar recursos ainda não prontos ou com defeito

Livro Jornada DevOps: MUNIZ; SANTOS; IRIGOYEN; MOUTINHO  
(Brasport, 2019)

# Arquitetura de baixo risco

## Microsserviço

Funcionalidades em serviços independentes



## Aplicação Monolítica

Funcionalidades ficam em processo único



Arquitetura	Vantagens	Desvantagens
Monolítica	<p>Inicialmente simples</p> <p>Baixa latência entre processos</p> <p>Eficiente para recursos em pequena escala</p>	<p>Fraca escalabilidade e redundância</p> <p>Implantação big bang</p> <p>Longo período de build</p>
Microserviço	<p>Cada unidade é simples</p> <p>Escalonamento independente</p> <p>Teste e implantação independente</p> <p>Facilita visão por produto</p>	<p>Latência de rede</p> <p>Precisa de ferramentas para gerenciar dependências</p>

- Segunda Maneira



# Bootcamp

# Arquiteto de Soluções - Cloud

Módulo 4 - Soluções para Desenvolvimento

Capítulo 3. Segunda Maneira

ANALÍA IRIGOYEN



# Arquiteto de Soluções

---

Aula 3.1. Segunda Maneira

ANALIA IRIGOYEN

# Telemetria



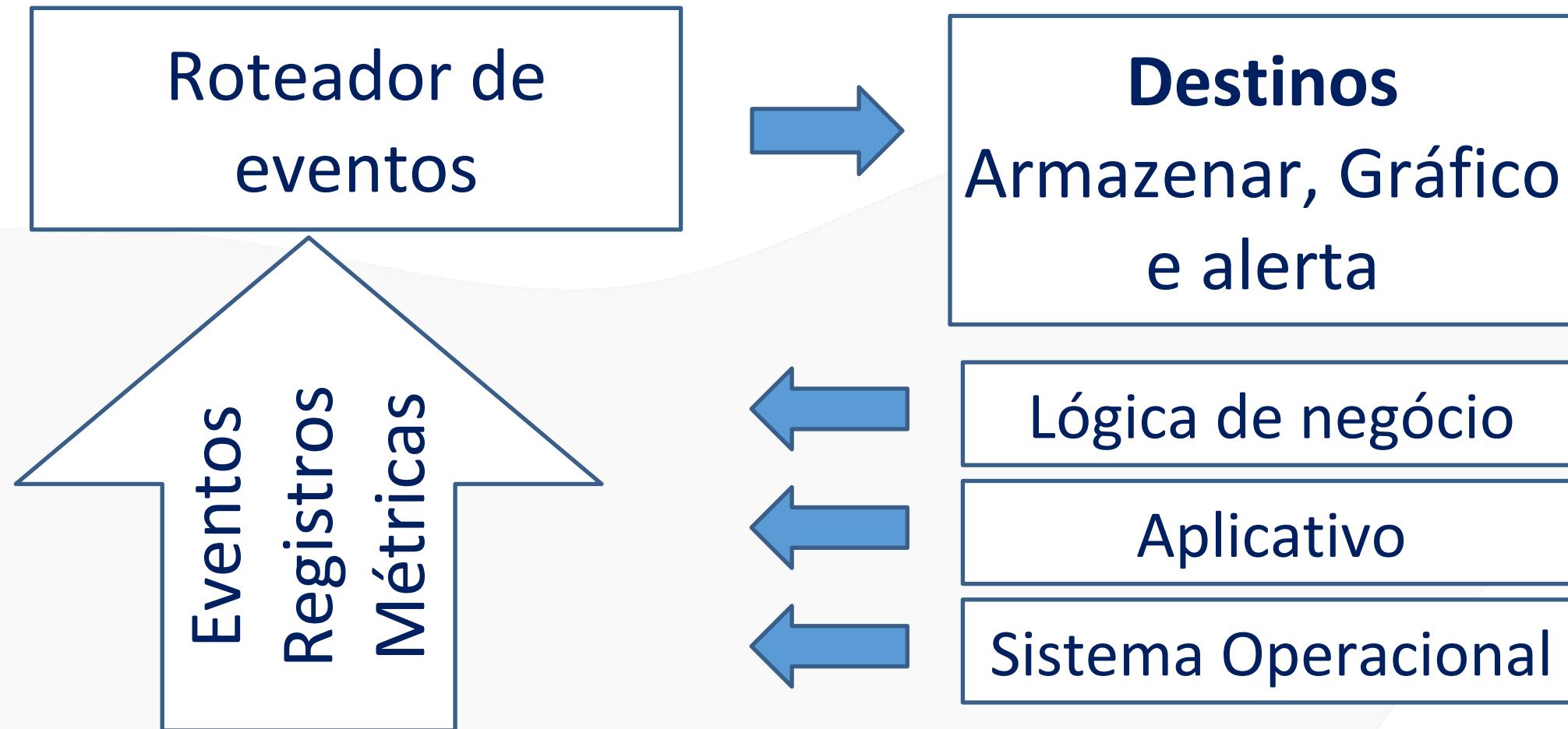
- ✓ Coleta remota e automática de dados para monitorar a saúde das aplicações
- ✓ Visão de todo o fluxo de valor permite otimizar e comunicar melhor

Fornece feedback efetivo:

- ✓ **Experiência do cliente**
- ✓ Erros no pipeline de implantação e após deploy
- ✓ Eventos proativos
- ✓ **Funcionalidades mais usadas pelos clientes**

# Framework de monitoramento

IGTI



Adaptado de DevOps Handbook

# Telemetria Self-service



Todos os participantes do fluxo de valor devem ter acesso ao resultado da telemetria, incluindo clientes

Métricas devem ser criadas como parte do trabalho diário das equipes

Acesso liberado sem necessidade de abertura de tickets: Usando APIs de autoatendimento

Sugerido usar irradiadores de informação visível para todos

## Feedback

Visão clara do fluxo de valor em todos os estágios do ciclo de vida do serviço, produto, desenvolvimento e implementação, entrada em operação e retirada

Dessa forma, todos os participantes do fluxo de valor tem a oportunidade de aprender desde os primeiros estágios do projeto

# Opções para resolver problemas



## Correção: Fix Forward

É uma mudança no código e deve ser executada em ambientes com grande maturidade:  
Teste automatizado, implantação rápida e telemetria

## Reversão: Rollback

Usuário deixa de receber a implantação e há 2 formas:

1. Implantar a versão anterior (Ex.: Canário)
2. Alternância de recursos:  
Mais fácil de reverter e menos arriscada

# Suporte compartilhado



A implantação de um pipeline com testes automatizado não garante zero erro, em função da complexidade dos sistemas

Erro de software depende que a equipe Dev priorize a solução para não impactar a operação por muito tempo

Todos os participantes do Fluxo de Valor devem compartilhar as responsabilidades para resolver os incidentes em produção

# Google SRE: Site Reliability Engineering



Dev representa a equipe de produto e gerencia o serviço em produção até que esteja saudável e faça transição para OPS (SRE)

SRE representa Ops e sustenta os serviços sob sua responsabilidade

Faz o papel de consultor durante o tempo em que o time Dev é responsável pela gestão do serviço em produção

## Itens do Checklist



- Severidade dos defeitos encontrados
- Tipo e frequência de alertas
- Cobertura de Monitoramento
- Arquitetura do Sistema
- Processo de Implementação
- Higiene de Produção: existem boas práticas de compartilhamento de conhecimento, automação e telemetria que permitam suporte em produção por outras pessoas?

# Experiência do usuário (UX) como feedback



**A pesquisa contextual** ocorre quando a equipe do produto observa um cliente usar a aplicação em seu ambiente natural

Mostra todas as dificuldades na prática (cliques numerosos, lentidão, transição de várias telas, migração de dados para outros sistemas, etc.)

Essa simples ação gera maior **empatia** e incentiva as equipes a melhorar a qualidade na origem

# Testes A/B



Técnica bastante usada em marketing para avaliar o resultado imediato e entender a experiência do usuário (UX)

**Controle (A):** Conteúdo original é mantido

**Tratamento (B):** Mudança em alguma parte do serviço para entender o impacto imediato

## Testes A/B e Experimentação

O Teste A/B é bastante útil quando desejamos experimentar novos recursos e tomar decisões com base em fatos e dados

Pode usar com a alternância de recursos (sem deploy):

- ✓ Habilitar nova funcionalidade para 10% dos usuários
- ✓ Se resultados positivos (mais vendas), expande para mais usuários até chegar a 100%

# Desenvolvimento orientado por hipóteses



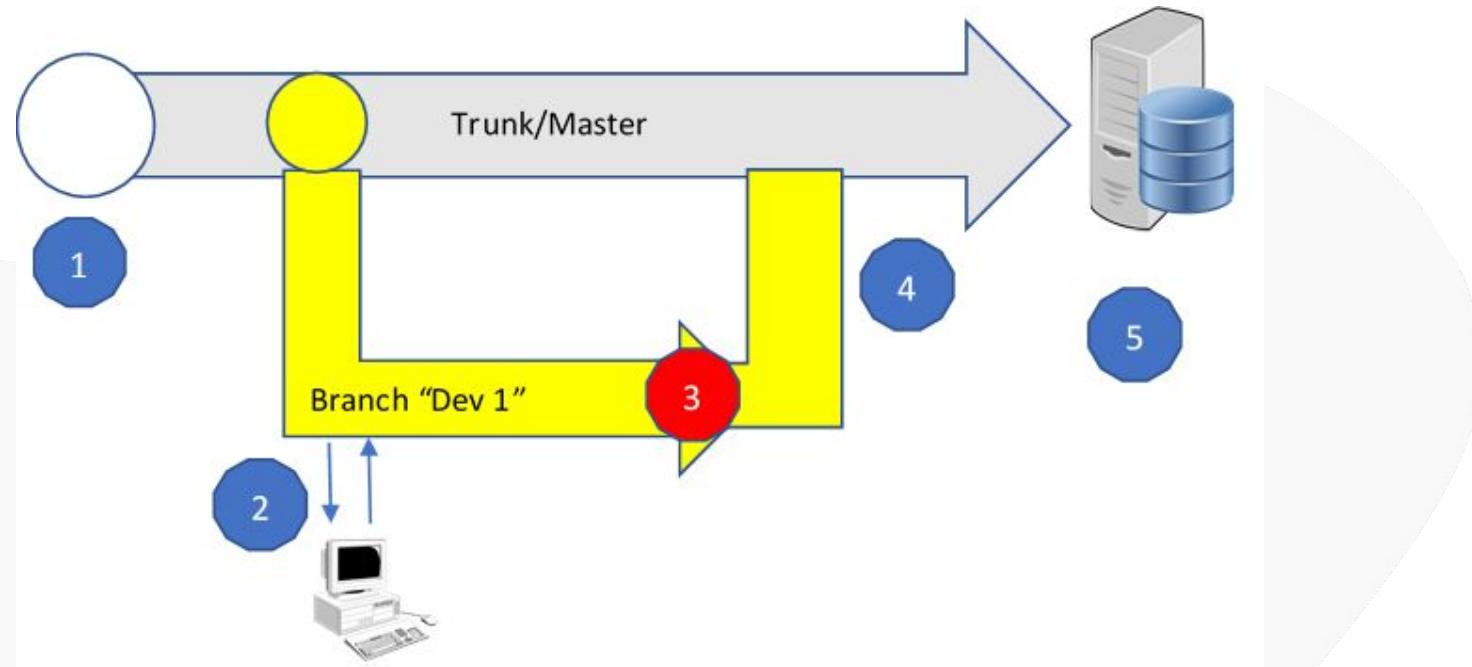
O uso de hipóteses é bastante importante para confirmar se novas funcionalidades serão realmente adotadas pelos clientes

O objetivo é refletir sobre uma série de experimentos para determinar se teve o resultado

O processo é realizado até alcançar a meta ou a ação mostre-se inviável (Pivatar)

# Pull Request

IGTI



# Importância do Pull Request



Desenvolvedor solicita revisão de forma colaborativa após concluir mudança no código.

Esse processo é muito usado no GitHub

- ✓ Experiência da Toyota mostra que as pessoas mais próximas do trabalho tem mais condições de colaborar com qualidade na fonte
- ✓ Resultados não melhoram com mais níveis de aprovação, novas perguntas na RFC, janelas longas

## Eficácia do Pull Request



Recomendação  
é que a revisão  
seja realizada  
pelo colega  
**antes** de efetivar  
o código no trunk

Revisão deve ser em pequenos lotes.  
Shoup diz: “Quando aumentamos o  
código de 10 linhas para 100 linhas, a  
chance de erro aumenta 10 vezes”

Mudanças críticas podem ter a  
revisão do especialista no assunto  
(segurança, BD, redes)

## Técnicas para revisão



Os próprios colegas analisam o código em busca de erros e melhora a qualidade das implantações

Funciona também como treinamento prático e para aperfeiçoar habilidades e aprendizado

Categorias: Programação em pares, sobre os ombros, E-mail repassado, Revisão de código assistida por ferramentas

## Programação em pares



O código é criado por duas pessoas trabalhando juntas em um único computador

A melhor maneira de parear o programa é sentar lado a lado na frente do monitor para revezar as atividades de codificação (piloto) e revisão (navegador)

Tempo gasto a mais é 15% e aumenta qualidade do software de 70% para 85%

## Outras técnicas de revisão



**Sobre os ombros:** Um membro da equipe observa sobre os ombros de quem executa e percorre todo o código

**E-mail repassado:** É usado um sistema de gerenciamento de código-fonte para enviar por e-mail de forma automática logo após que o código é inserido

**Ferramentas:** Tanto o autor quanto o revisor usam ferramentas especializadas para revisão em pares ou recursos fornecidos pelos repositórios de código-fonte (Ex.: GitHub)

- Terceira Maneira



# Bootcamp

# Arquiteto de Soluções - Cloud

Módulo 4 - Soluções para Desenvolvimento

Capítulo 4. Terceira Maneira

ANALÍA IRIGOYEN

# Arquiteto de Soluções

---

Aula 4.1. Terceira Maneira

ANALIA IRIGOYEN

# Terceira Maneira -DevOps



- ❑ Motivação
  - Nosso Cérebro é avesso às mudanças
  - Aprendizado
  - O profissional DevOps
  - DevOps é para todos
- ❑ A importância da Terceira Maneira para a Cultura DevOps
- ❑ O que é a Terceira Maneira mesmo?

# Motivação

IGTI

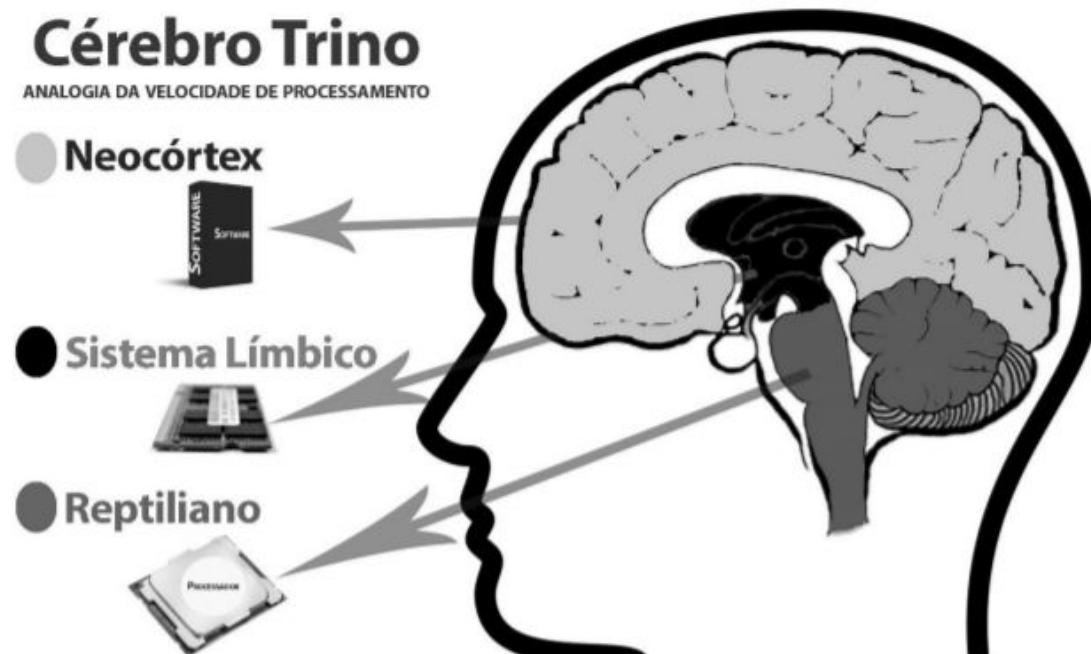


Figura 5.2. Analogia da velocidade de processamento. Fonte: adaptado de MACLEAN, 1990.

**Fonte:** Livro Jornada Ágil e Digital: MUNIZ;IRIGOYEN;(Brasport, 2019)

## **Somos, por natureza, avessos à mudanças**

Aprendizado gera desconforto no início, sair da zona de conforto é sempre um desafio

**O seu cérebro vai criar mil razões para impedi-lo de mudar os hábitos**

A sensação de prazer ao conseguir melhorar com fatos e dados não tem preço – Pare, reflita e mude

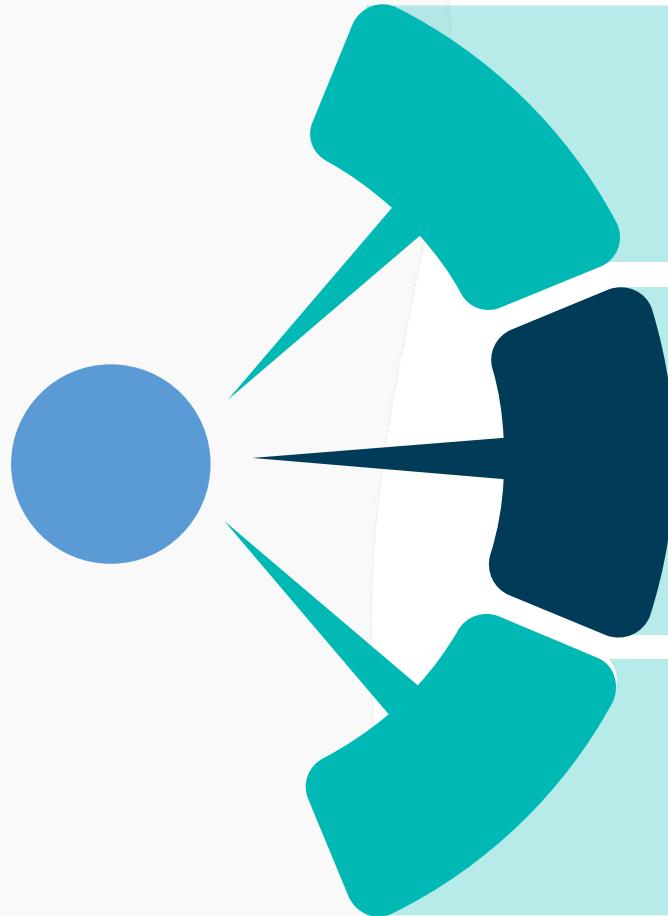
# Importância da Terceira Maneira



Correr Riscos (Cultura Justa – Fatos e Dados)

Loop de feedback contínuo

Melhoria contínua até a perfeição



# Aprendizado x Segurança psicológica

Segurança Psicológica



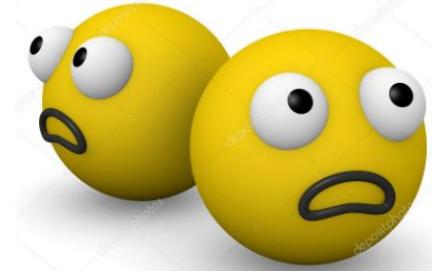
Zona de  
Conforto

Zona de  
Aprendizado



Zona da Apatia

Zona da  
Ansiedade



Motivação e Responsabilidade

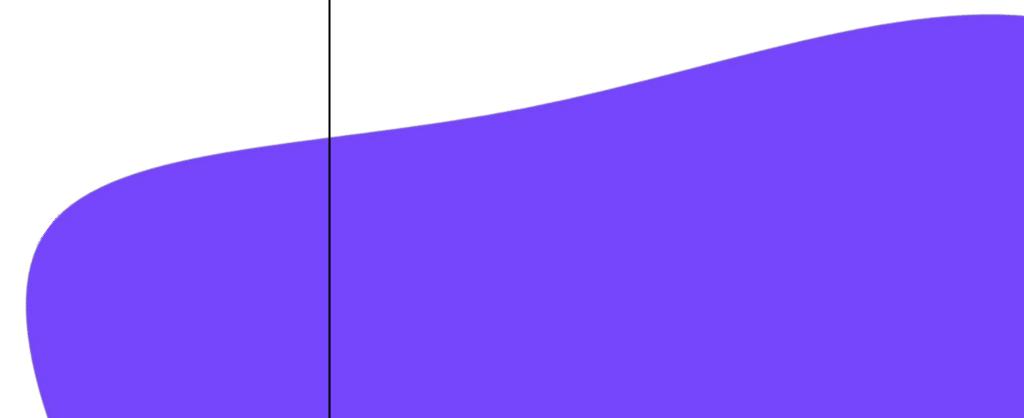
<https://www.youtube.com/watch?v=LhoLuui9gX8>:

Building a psychologically safe workplace | Amy Edmondson | TEDxHGSE

# O profissional DevOps



Forma	Características importantes
Especialista	<ul style="list-style-type: none"><li>✓ Expertise profunda em uma área de conhecimento</li><li>✓ Falta de interesse com impacto global do seu trabalho cria gargalos</li></ul>
Generalista	<ul style="list-style-type: none"><li>Pouca preocupação com desperdícios e visão limitada de silo</li><li>✓ Impede flexibilidade no planejamento e adaptação das ações</li></ul>
Experiência	<ul style="list-style-type: none"><li>✓ Expertise profunda em uma área de conhecimento</li><li>✓ Amplas habilidades em muitas áreas permite visão global</li><li>✓ Antecipa ações para evitar gargalos e desperdícios</li><li>✓ Colabora com flexibilidade e adaptação das ações</li></ul>



Livro Jornada Agil e Digital: MUNIZ;IRIGOYEN;(Brasport, 2019)

Colabora com flexibilidade e adaptação das ações

# Aprendizado contínuo

"Os analfabetos do século 21 não serão aqueles que não sabem ler e escrever, mas aqueles que não sabem aprender, desaprender e reaprender".

Alvin  
Toffler



FONTE: <https://image.aldeiasanecas.com/BaSc04e4-efb0-45e1-9425-b4d9eb07104-180212132347/95/como-ler-e-mais-e-pronunciar-uma-safra-de-investimentos-na-area-5-838.jpg?fbclid=IwAR0283567>

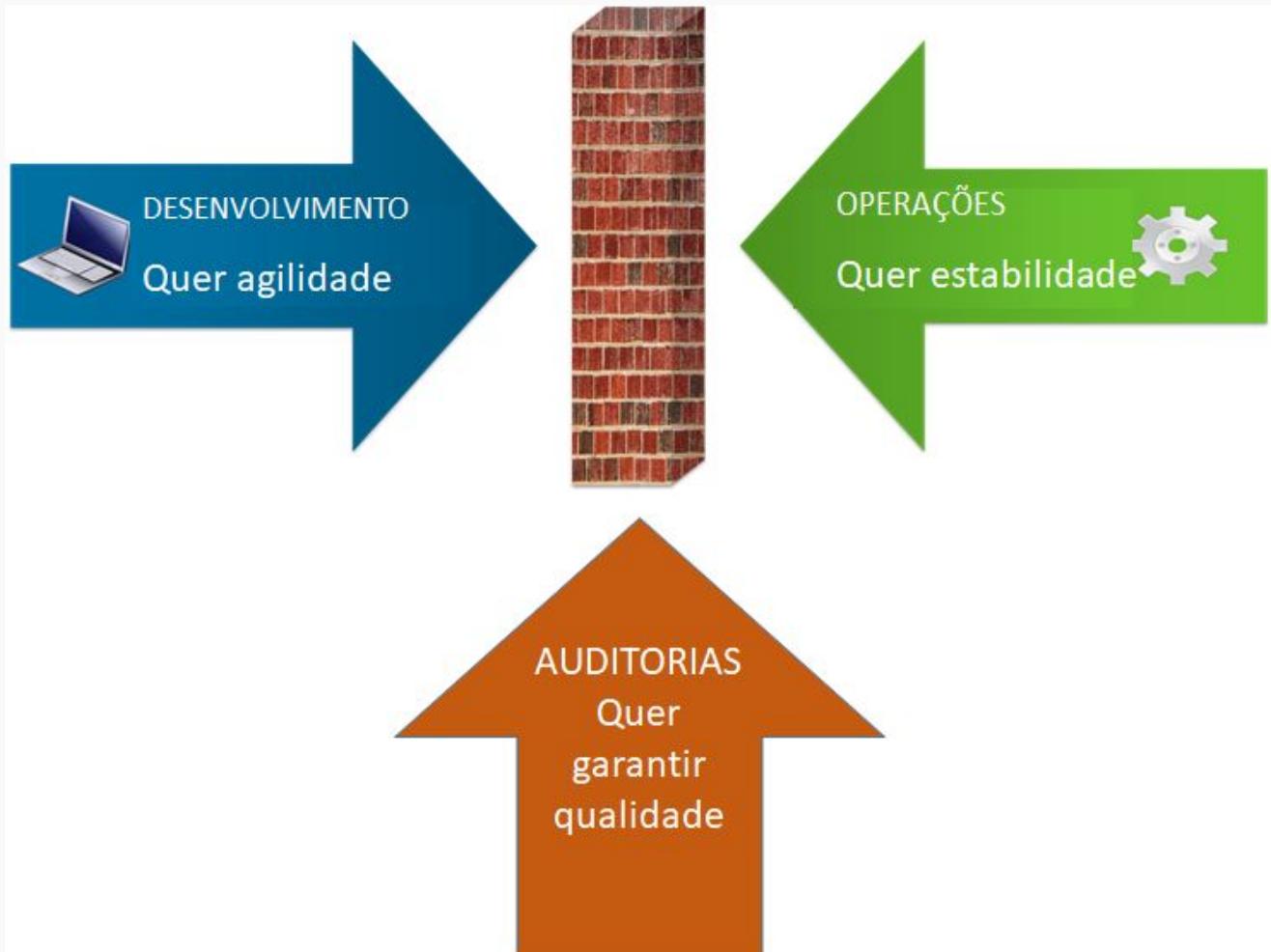
# Organizações exponenciais

IGTI

1	O	2	O	3	O	4	O	5	O	6	O	7	O	8	O	9	O	10	O
<b>amazon</b>																			
31.81%		13.79%		29.97%		4.73%		5.00%		16.21%		15.37%		-7.41%		-0.80%		17.62%	
\$415,855.00	M	\$352,206.00	M	\$326,544.00	M	\$323,601.00	M	\$186,809.00	M	\$152,525.00	M	\$150,978.00	M	\$147,190.00	M	\$129,321.00	M	\$108,129.00	M
11	O	12	O	13	O	14	O	15	O	16	O	17	O	18	O	19	O	20	O
 <b>AT&amp;T</b>		 <b>verizon</b> ✓		 <b>Coca-Cola</b>		 <b>IBM</b>		 <b>Marlboro</b>		 <b>The Home Depot</b>		 <b>SAP</b>		 <b>MOUTAI</b>		 <b>LOUIS VUITTON</b>		 <b>ups</b>	
-2.35%		0.07%		3.96%		-2.72%		-19.05%		7.62%		0.09%		58.46%		9.66%		-7.56%	
\$105,833.00	M	\$94,662.00	M	\$84,022.00	M	\$83,667.00	M	\$58,247.00	M	\$57,585.00	M	\$57,578.00	M	\$53,755.00	M	\$51,777.00	M	\$50,748.00	M

São as que mais aprendem!!

# DevOps é para todos



IGTI

# DevOps é para todos



**iGTD**

Mudança???

Auditoria????

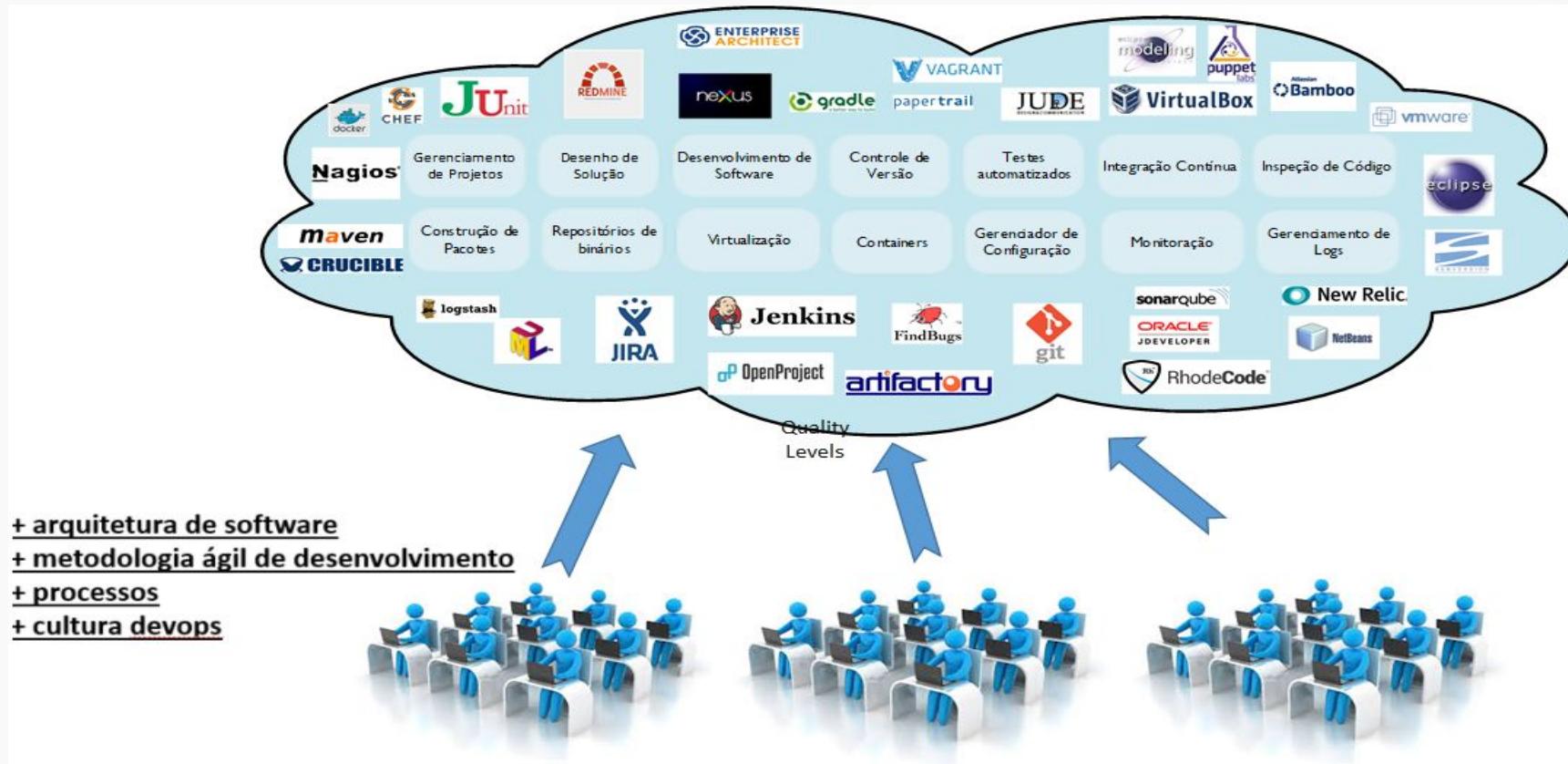
Evidência???

Compliance?

Segurança e LGPD??

SOX?

# DevOps = Engenharia de Software



IGTI

Para orquestrar isso tudo só com boas práticas e muito conhecimento  
de: **Gestão, Qualidade, Engenharia de Software, Serviços, Inovação**

# DevOps = Engenharia de Software



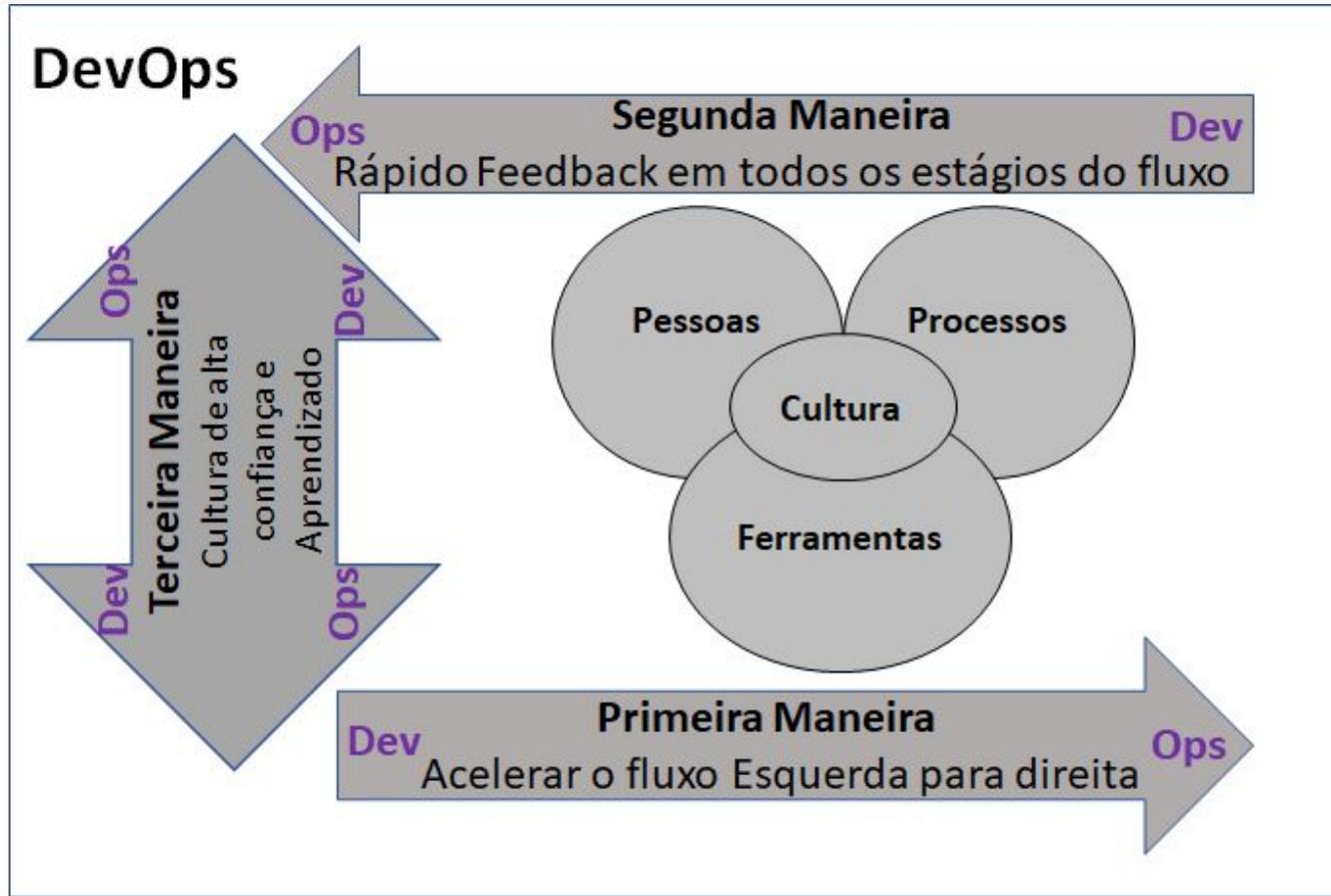
Cada implementação de pacote envolve:

1. Planejamento da melhor solução para o cliente
2. Implementação das ferramentas
3. Treinamento da cultura DevOps
4. Treinamento da equipe no uso das ferramentas
5. Treinamento de arquitetura de software orientado para as ferramentas
6. Acompanhamento de um projeto piloto
7. Adequação do processo da empresa de forma a levar em consideração as ferramentas em todo o processo de desenvolvimento

**Com gestão automatizada e segurança!**

# Importância da 3<sup>a</sup> Maneira

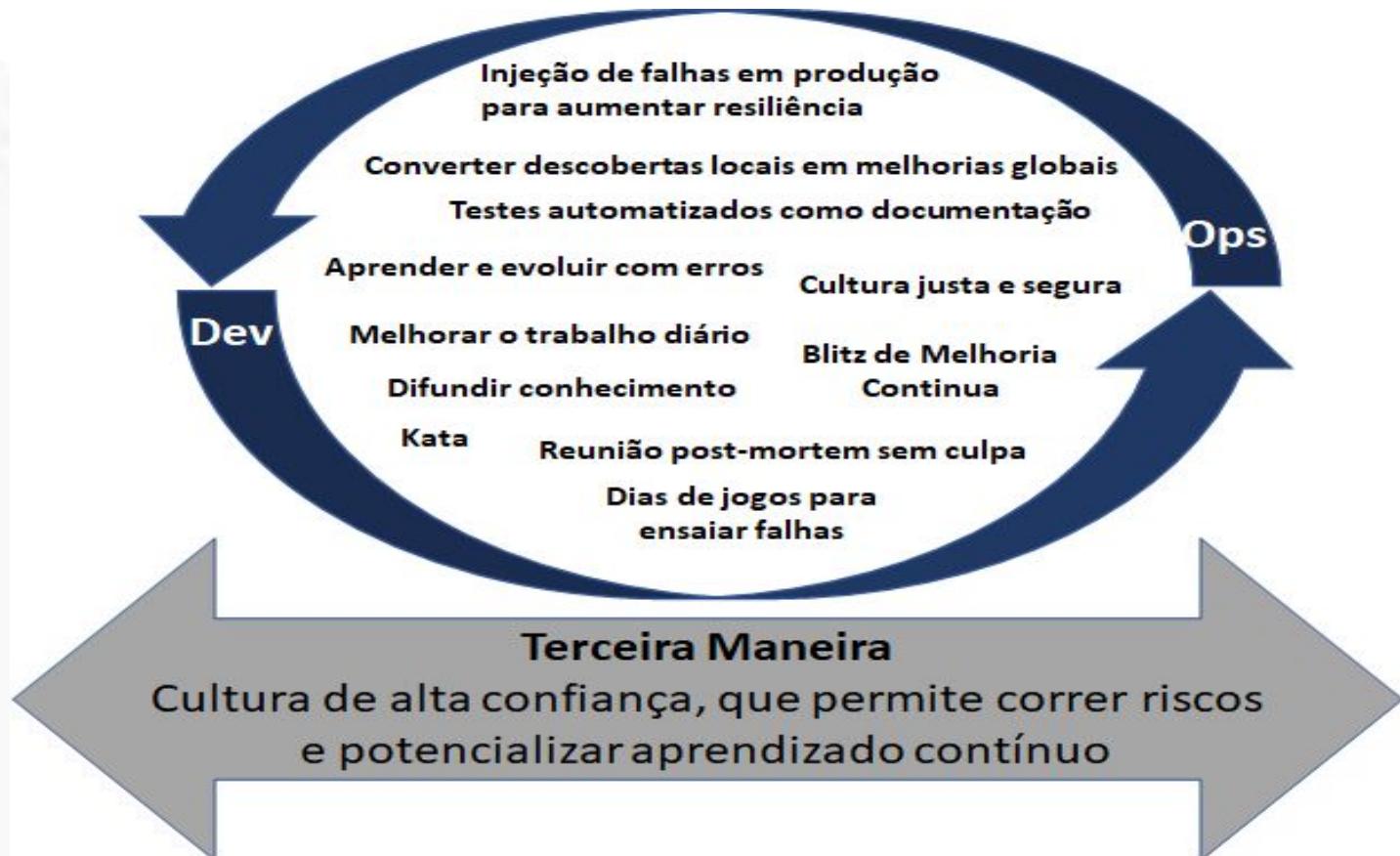
iG  
TI



Overview do DevOps. Fonte: Elaborado pelo autor.

# Introdução da Terceira Maneira

IGTI



Segunda Maneira. Fonte: adaptado de (MUNIZ; SANTOS; IRIGOYEN; MOUTINHO. 2019).

# Próximas aulas



01.

Macacos do exército simiano  
(Netflix)

02.

Reunião post-mortem

03.

Dias de jogos

04.

Descobertas

# Arquiteto de Soluções

---

Aula 4.2. Macacos do exército simiano (Netflix)

ANALIA IRIGOYEN

# Terceira Maneira - Aprendizado



- Tipos de Macaco do exército simiano (Case Netflix)
- Reunião post-mortem livre de culpa
- Dias de jogos
- Descobertas

# Aprendizado com falhas



✓ Netflix criou um processo de injeção de falhas chamado Chaos Monkey que aumenta a resiliência

✓ O foco é desenvolver aprendizado contínuo nas equipes e evitar a cultura de culpa e punição

# Exército de Sírios Netflix

IGTI

- ✓ O Chaos Monkey evoluiu para novos macacos chamado “Exército Sírio”
- ✓ Essa técnica ficou famosa quando houve uma queda de zona da AWS em 2011 e a Netflix não foi afetada porque tinha se preparado proativamente



# Exército de Sírios Netflix



**Chaos Gorilla:** Simula a falha de **uma zona inteira** de disponibilidade AWS (Serviço Web Amazon)



**Chaos Kong:** Simula indisponibilidade em **regiões inteiras** da AWS (Ex: Europa, América do Norte, África, etc)

# Exército de Sírios Netflix



## Macaco Janitor

### Macaco de Latência

Causa atrasos ou paralisações artificiais

Simula a **degradação de serviço** para garantir que serviços dependentes respondam de forma adequada



Responsável em garantir que o ambiente esteja livre de desperdício e desorganização



**Procura recursos não utilizados e desativa**

# Exército de Sírios Netflix



## Macaco de Conformidade

✓ Localiza e desliga instâncias AWS que **não seguem as melhores práticas** (Ex: falta de email para alerta)

✓ O Macaco de Segurança **é uma extensão para vulnerabilidades**

## Macaco Doutor

✓ Verifica a integridade de cada instância

✓ **Desliga instâncias não íntegras** quando o responsável não resolve a causa raiz no tempo combinado

# Próximas aulas



01.



Macacos do exército simiano  
(Netflix)

02.

Reunião post-mortem

03.

Dias de jogos

04.

Descobertas

# Arquiteto de Soluções

---

Aula 4.3. Reunião post-mortem

ANALIA IRIGOYEN

# Terceira Maneira - Aprendizado



- Tipos de Macaco do exército simiano (Case Netflix)
- Reunião post-mortem livre de culpa
- Dias de jogos
- Descobertas

# Aprendizado com falhas



Para o verdadeiro aprendizado, é necessário combater a teoria da maçã podre, que busca “eliminar as pessoas que causaram os erros”

✓ Segundo Dekker, o erro humano é a consequência do projeto de ferramentas que as pessoas recebem para trabalhar: **Deve-se buscar a causa sistêmica dos erros**

Um engenheiro do Google confessou: “Eu estraguei uma linha de código e isso nos custou um milhão de dólares em receita” Não foi demitido...

# Dicas para criar cultura justa



1. Nunca nomear ou envergonhar quem é o culpado da falha

2. Incentivar quem compartilha problemas do sistema

3. Criar confiança para que equipe aprenda com problemas

4. Lições aprendidas sem culpa e injeção controlada de falhas em produção

# Reunião post-mortem livre de culpa

**Após a solução de incidentes, objetivo é entender e divulgar:**

1. Ações que deram certo
2. Ações que podem ser aprimoradas
3. Erros e medidas para evitar recorrência

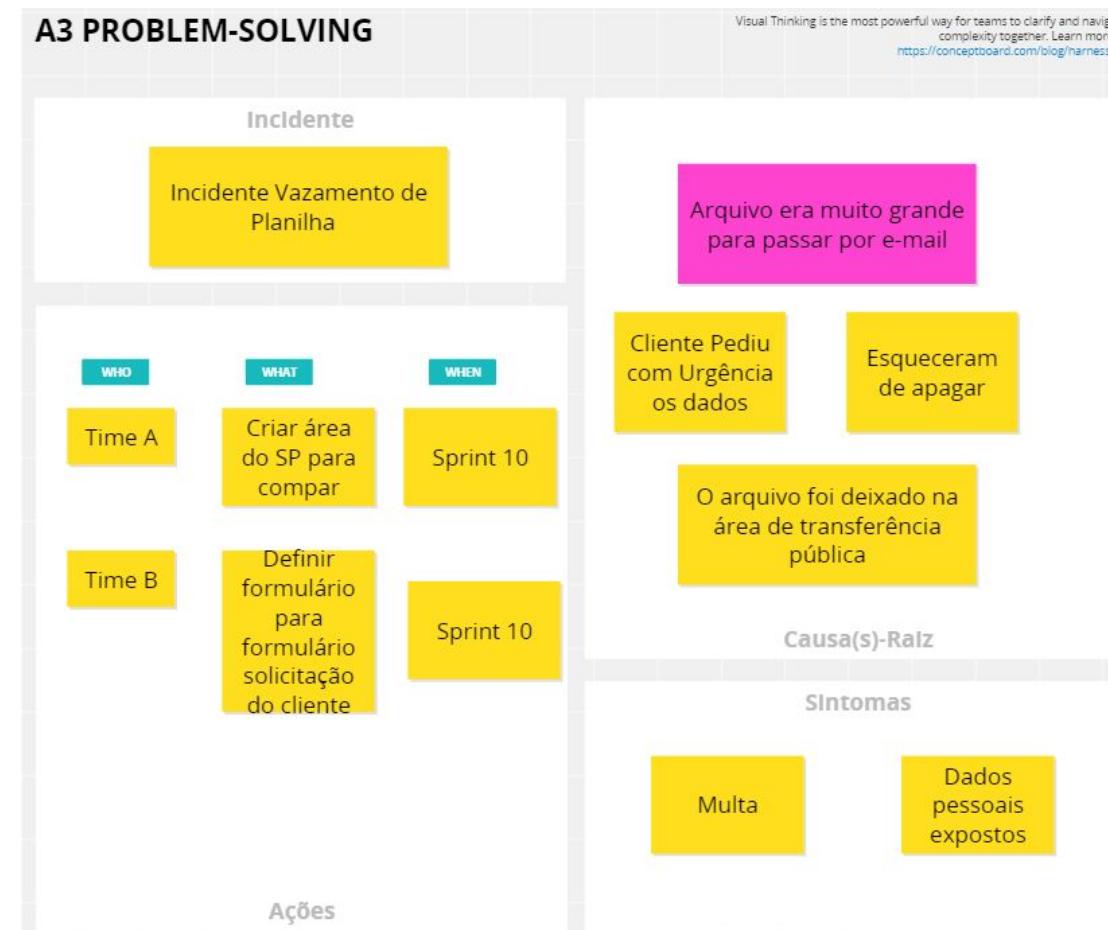
**Etapas importantes:**

1. Agendar a reunião com as pessoas envolvidas
2. Realizar a reunião
3. Publicar o resultado da reunião

# Retrospectiva Causa-Raiz (K21)



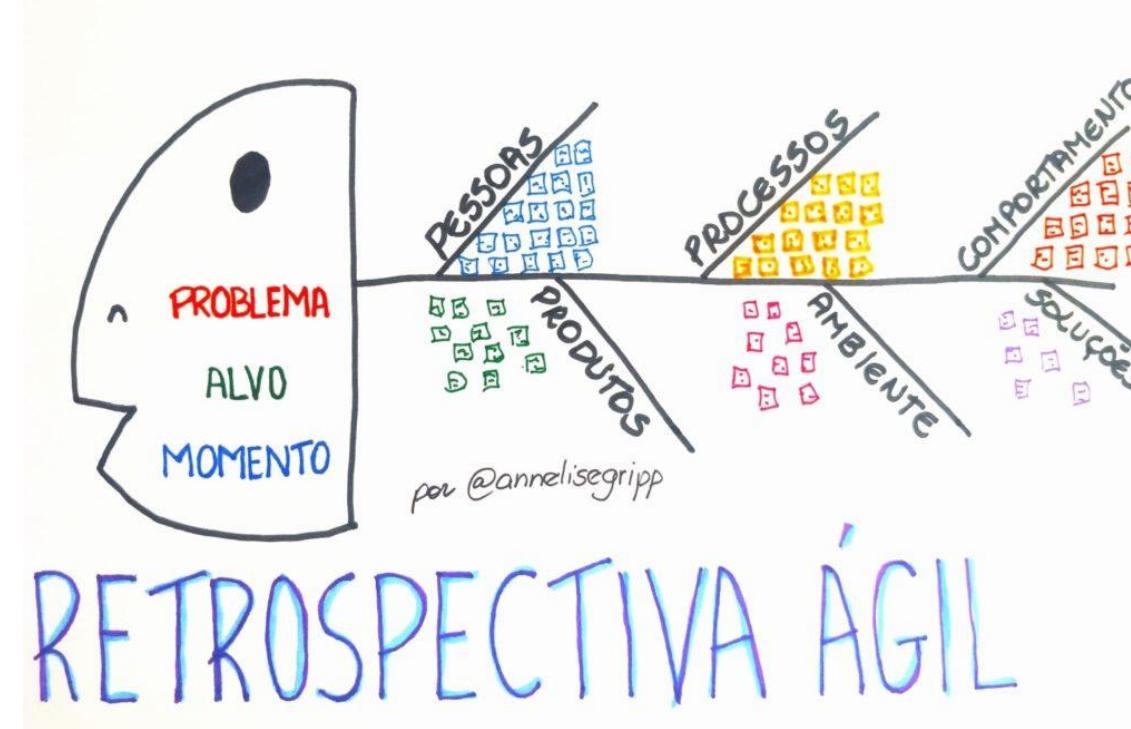
- Descobrir Sintomas (1)
- Chegando a causa-raiz
- Planos de Ação



# Retrospectiva Causa-Raiz

IGTI

- Chegando a causa-raiz



<https://annelisegripp.com.br/retrospectivas-ageis/>

# Retrospectiva - Timeline



# Próximas aulas



01.



Macacos do exército simiano  
(Netflix)

02.



Reunião post-mortem

03.

Dias de jogos

04.

Descobertas

# Arquiteto de Soluções

---

Aula 4.4. Dias de jogos

ANALIA IRIGOYEN

# Terceira Maneira - Aprendizado



- Tipos de Macaco do exército simiano (Case Netflix)
- Reunião post-mortem livre de culpa
- Dias de jogos
- Descobertas

# Dias de Jogos – Games Days



Esse conceito vem da engenharia da resiliência, cujo objetivo é criar exercícios programados para aumentar a resiliência através da injeção de falhas de grande escala em sistemas críticos

Foi popularizado por Jesse Robbins pelo trabalho que fez na Amazon para garantir a disponibilidade do site. Ele ficou conhecido como o “Mestre do Desastre” e defende que **“Um serviço não está realmente testado até o estragarmos em produção”**

# Dias de Jogos – Etapas



## 1. Planejar a interrupção

Ex: Simular perda completa de datacenter

## 2. Adotar medidas

Ex: Criar site de contingência

## 3. Testar medidas

Ex: Usar site de contingência

## 4. Executar interrupção

Ex: Avaliar resultados da contingência

## 5. Seguir plano e aprender

Ex: Avaliar situações não previstas

# Games Days: como é na AWS?



## 1. Defina seu cenário

### Dicas de cenários:

Falhas anteriores conhecidas  
fraquezas de processos e times,  
tarefas sazonais que são feitas  
sob demanda, etc.

Requisitos de Ambiente necessário  
(HW/SW)

Selecionar e Comunicar Pessoas que  
vão jogar e observar o jogo

Pessoas que vão “jogar” precisam  
conhecer ferramentas e processos

Múltiplos times vão precisar de  
múltiplos observadores

Defina um processo para a divulgação  
dos jogos

# Games Days: como é na AWS?



## 2. Prepare o ambiente para o jogo

Criar ambientes, que devem ser “igual” ao de produção

Pensar nas permissões, indicadores críticos e automatizar as verificações (runbooks)

- **Requisitos** - permissões, ferramentas e suas configurações, acesso e conectividade.
- **Restrições** - Janelas de manutenção, Recursos impactados e Identificar conflitos entre atividades de negócio e operações.
- **Procedimentos e saídas** - vulnerabilidades, fraquezas e/ou melhorias registradas em...
- **Procedimentos de escalamento** - jogadores não participarem, Timebox, para quem escalar, o que e quando
- Tomadores de Decisão (antes, durante e após os jogos).

Dicas de Runbooks  
O que documentar?

# Games Days: como é na AWS?



3. Defina um cronograma

Divulgue

Obtenha o comprometimento

4. Execute a simulação

- Use uma sala separada
- Anuncie que os jogos vão começar
- Verifique se o runbook está sendo executado
- Ouça o feedback dos observadores para decisões de adiamento das simulações
- Anuncie o fim do jogo

# Games Days: como é na AWS?



## 4. Analise o dia de jogo

Faça uma retrospectiva e anote melhorias nos processos, ferramentas e runbook !

Analise necessidades adicionais de treinamentos, ferramentas ou automações

Documente outras áreas para os próximos dias de jogos

# Próximas aulas

IGTI

01.



Macacos do exército simiano  
(Netflix)

02.



Reunião post-mortem

03.



Dias de jogos

04.

Descobertas



# Arquiteto de Soluções

---

Aula 4.5. Descobertas

ANALIA IRIGOYEN

# Terceira Maneira - Aprendizado



- Descobertas
  - Requisitos não-funcionais
  - Histórias de usuários de operações reutilizáveis
  - Como transformar melhorias locais em melhorias globais
  - Teste automatizado como documentação

## Próximo Capítulo

- Segurança
- Gestão de Mudanças

# Descobertas



- ❖ Como divulgar e compartilhar aprendizados (descobertas) entre os times?
  
- ❖ Empresas de alto desempenho obtém os mesmos resultados (ou melhores) refinando as operações diárias, introduzindo tensão continuamente para aumentar o desempenho, e assim gerando mais **resiliência** nos seus sistemas.

# Descobertas – Requisitos NF



Usando os requisitos não funcionais para projetar operações:

- ❖ Telemetria completa de produção: verifica através de métricas se o sistema está comportando conforme o esperado;
- ❖ Capacidade de monitorar dependências: verifica se os serviços que fornecem dados ao sistema estão operacionais, estão funcionando e na capacidade adequada.
- ❖ Serviços resilientes que degradam quando necessário: serviços que podem ser degradados de uma forma planejada para que a aplicação não caia dada uma carga excessiva no sistema;
- ❖ Compatibilidade com todas as versões: verifica se o sistema é compatível com outros sistemas nos quais ele é integrados. Geralmente aqui são implementados testes de contrato.

# Descobertas – Requisitos NF



**Usando os requisitos não funcionais para projetar operações:**

- ❖ Mensagens intuitivas de erros.: os logs de erros da aplicação devem ter toda informação necessária para que o time possa reagir de forma eficaz quando investiga um determinado problema e/ou incidente na aplicação;
- ❖ Monitorar pedidos dos usuários: ter uma ferramenta onde são acompanhadas as solicitações dos usuários, que geralmente chegam via suporte da aplicação, realizando ajustes se necessário ou dando um feedback de como contornar o problema até a correção definitiva. Também ser transparente ao usuário quanto aos feedbacks de melhorias que eles vão dando ao longo do percurso.
- ❖ Configurar tempo de execução: determinar uma medida de limite para acompanhar o tempo de execução dos processos e saber no exato momento em que eles estão sofrendo degradação do serviço ou o serviço se tornou indisponível.

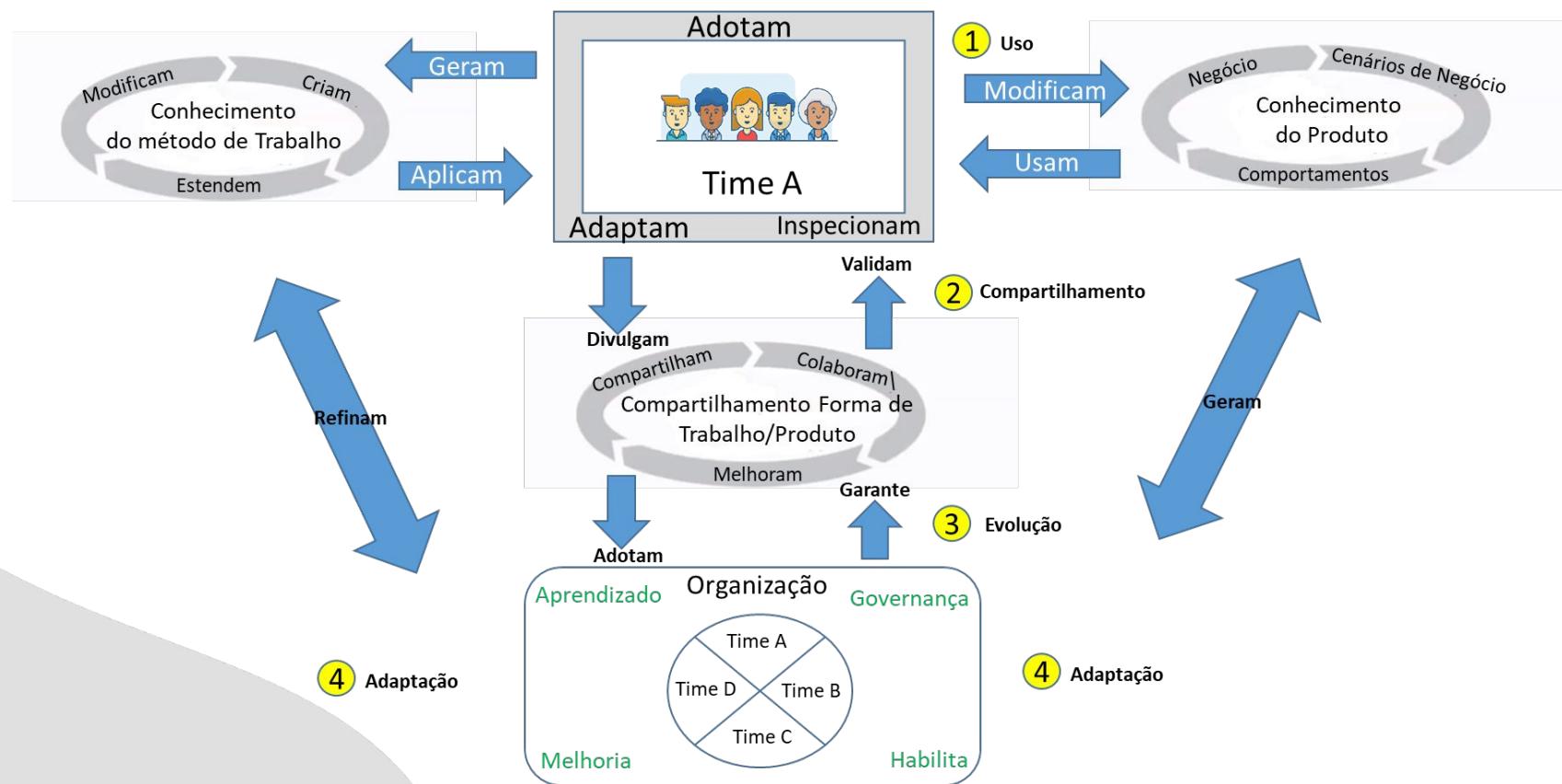
# Como transformar melhorias Locais em Globais



Transformar processos que estão documentados em editores de texto em *workflows* e/ou *scripts* automatizados é uma outra forma de transformar um conhecimento local em organizacional, permitindo a reutilização e a sua ampla utilização, fornecendo valor agregado para todos os que usam.

# Mas o que é conhecimento mesmo?

IGTI



# Descobertas – Histórias de Operações Reutilizáveis



## Benefício:

- ❖ Expõe o trabalho de Operações de TI passíveis de reprodução de forma que aparece ao lado do trabalho de Desenvolvimento, permitindo melhor planejamento e resultados mais passíveis de reprodução
- ❖ Manter as configurações de ambientes automatizadas, sendo assim, sempre que preciso é possível criar um ambiente novo de forma rápida e confiável, considerando que as mesmas configurações serão aplicadas em ambiente de produção.

# Descobertas – Histórias de Operações Reutilizáveis

## Como começar?

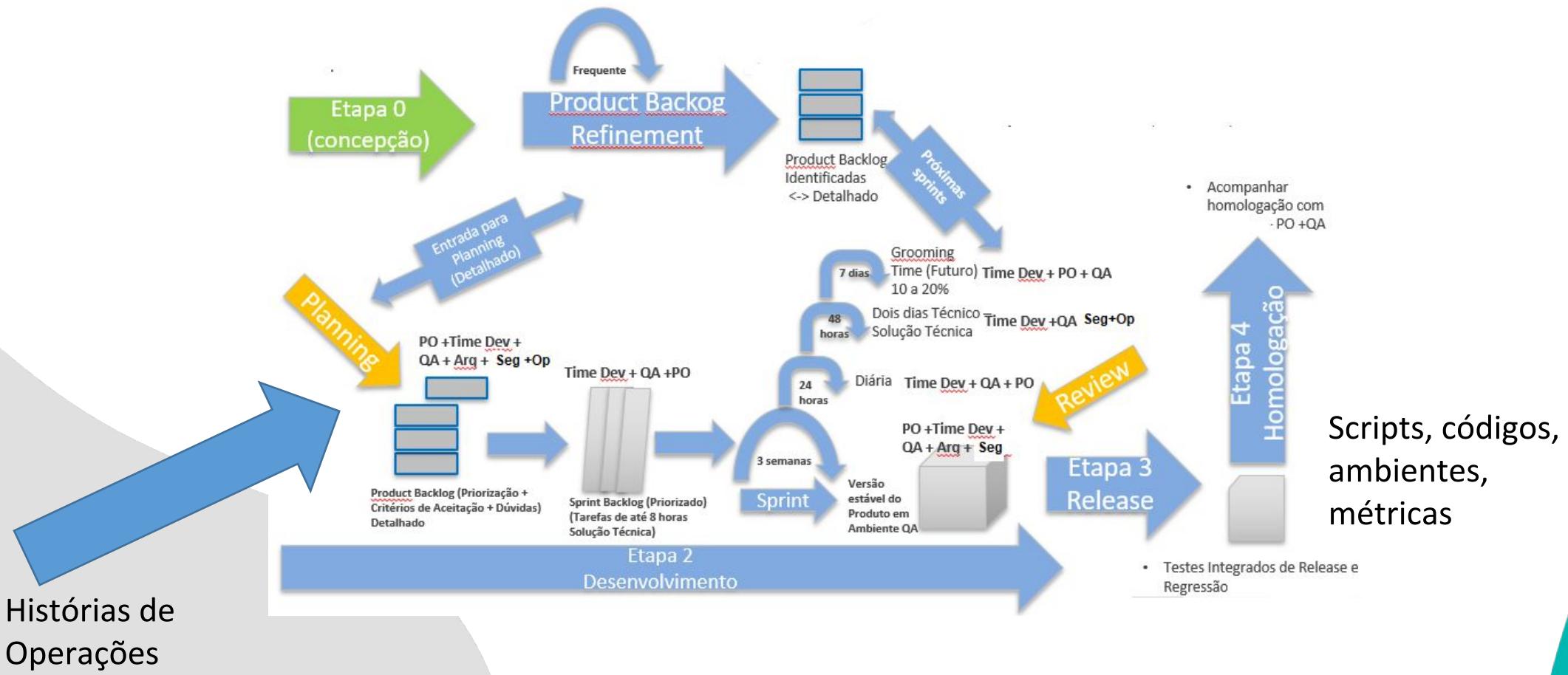
**Atividades:** levantar quais são as atividades do time de operação que o time de DEV pode ajudar a automatizar.

**Recursos Necessários:** determinar o que é preciso em termos de infra para que o sistema venha a funcionar em produção.

**Etapas planejadas:** para cada etapa do projeto ou do ciclo de desenvolvimento do software, é necessário entender em quais etapas o time de operação atua para agir sempre em conjunto com a evolução do software. Ou seja, realizar o *hand-off* das atividades, seja documentar de forma simples quando há atuação do desenvolvimento ou da operação e em que momento isso ocorre.

**Ferramentas:** disseminar o conhecimento acerca das ferramentas que o time de operações utiliza para poder atuar junto com eles seja no monitoramento, seja na infra ou em outras configurações necessárias. Embora cada time tenha autonomia para escolher suas ferramentas, é interessante disseminar o conhecimento sobre as ferramentas e tentar alinhar ferramentas que possam ser utilizadas pelo time de desenvolvimento e operações.

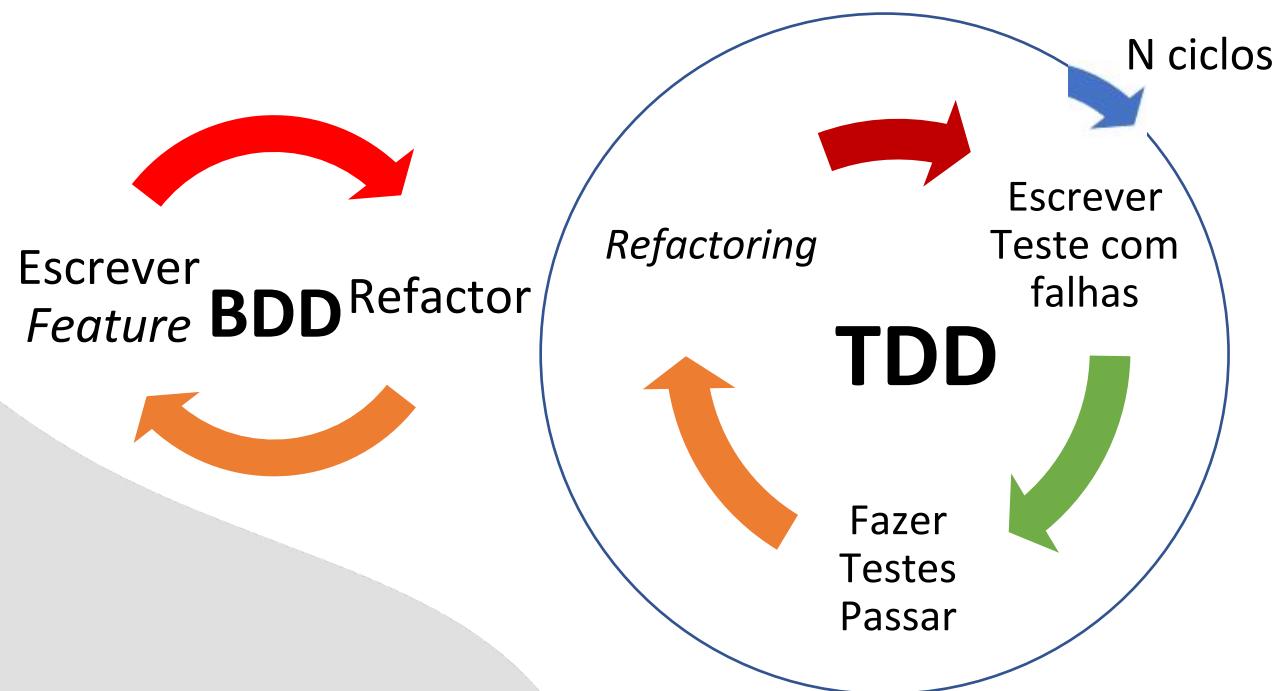
# Descobertas – Histórias de Operações Reutilizáveis -



# Testes Automatizados como Documentação - BDD

IGTI

Behavior Driven Development



# | Testes Automatizados como Documentação - BDD



## Behavior Driven Development

- ❖ Técnica de desenvolvimento Ágil
- ❖ Encoraja colaboração entre desenvolvedores, setores de qualidade e pessoas não-técnicas ou de negócios num projeto de software.
- ❖ Resposta ao TDD que é direcionado unicamente aos desenvolvedores.
- ❖ Frameworks de automatização conhecidos: Jbehave(Java), Rspec(Ruby), Cucumber, Nbehave(.Net), SpecFlow (.Net)

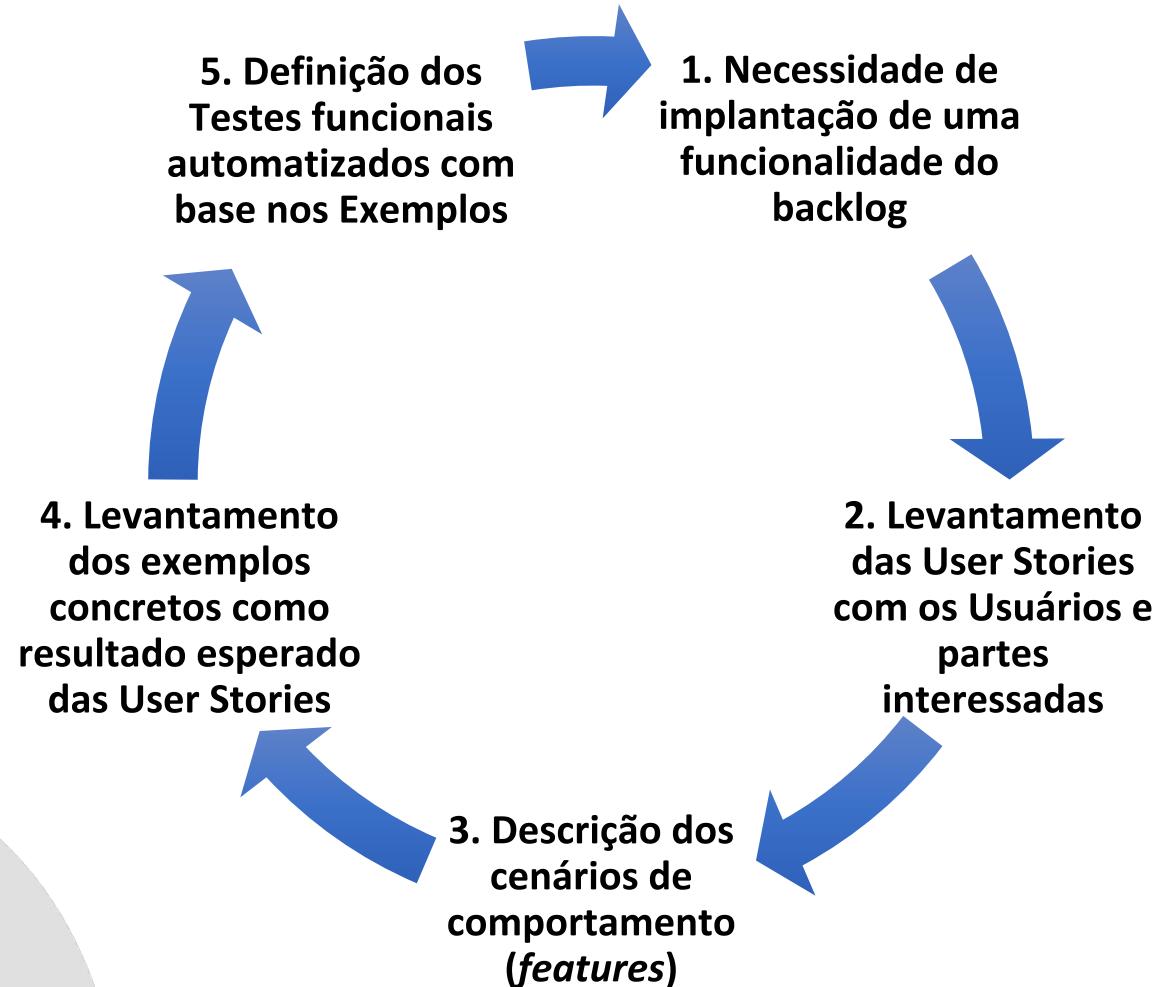
**“Behavior-driven development é sobre implementar uma aplicação através da descrição de seu comportamento pela perspectiva de seus stakeholders”**

-- Dan North

# Testes Automatizados como Documentação - BDD



## Passos do BDD



# Testes Automatizados como Documentação - BDD



Product Owner  
and Dev Team



- BDD sugere que os analistas/testadores escrevam os cenários antes mesmo dos testes serem implementados, e desta forma os desenvolvedores terão uma visão geral do objetivo do projeto antes de codificá-lo.

# Testes Automatizados como Documentação - BDD



- Em BDD, um desenvolvedor, algum profissional do setor de qualidade ou até mesmo o cliente podem esclarecer os requisitos quebrando-os em exemplos específicos.
- Exemplo de requisito: "Itens reembolsados ou substituídos devem ser retornados para o estoque". Como quebrar em BDD:

## Cenário 1: Itens reembolsados devem retornar para o estoque

**Dado** que um cliente compra um jumper preto  
**E** eu tenho três jumper pretos no estoque  
**Quando** ele retorna com o jumper preto para reembolso  
**Então** eu devo ter quatro jumpers pretos no estoque

## Cenário 2: Itens substituídos devem ser retornados ao estoque

**Dado** que uma cliente compra um vestido azul  
**E** eu tenho dois vestidos azuis no estoque  
**E** eu tenho três vestidos pretos no estoque  
**Quando** ela retorna com o vestido para uma troca por um preto  
**Então** eu devo ter três vestidos azuis no estoque  
**E** dois vestidos pretos no estoque

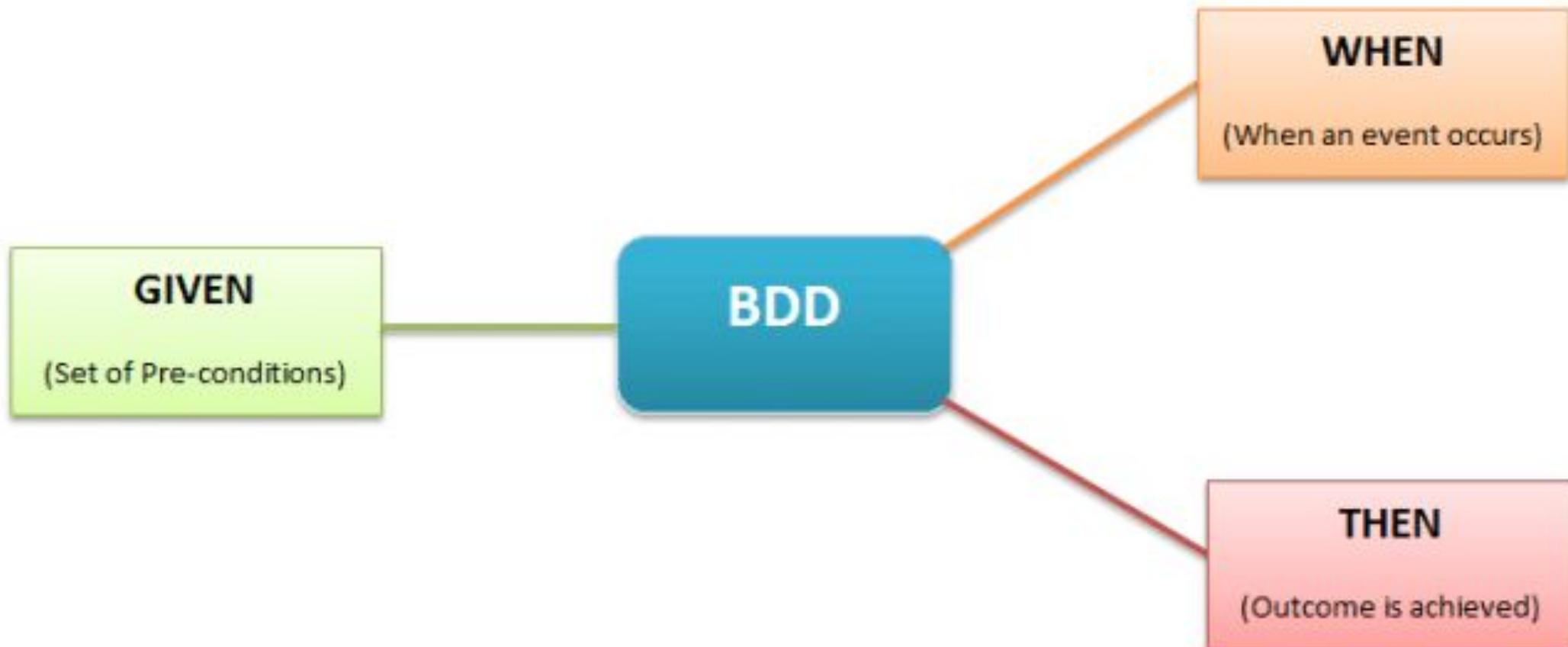
# Gherkin



- Linguagem **semi-formal** cujo objetivo é descrever cenários de teste inteligíveis, automatizáveis e em conformidade com uma necessidade de negócio.
- Linguagem orientada a **espaços**, ela usa **indentação** para definir a estrutura. Os fins de linha encerram as declarações (Passos) e espaços ou tabs também podem ser usados para **indentação**.
- Palavras reservadas:
  - Feature** -> Funcionalidade
  - Scenario** -> Cenário
  - Given** -> Dado
  - When** -> Quando

# Gherkin

IGTI



# Gherkin - Exemplos



```
1 #language: pt-BR
2
3 Funcionalidade: Receber feedback quando dados de compra estão errados
4     Para evitar que eu me frustre e desista da minha compra
5     Enquanto cliente
6     Desejo que o sistema previna a entrada de dados inválidos de cartão de crédito
7
8
9 Cenário: Digitei letras no campo de número do cartão
10    Dado que eu escolhi os itens que vou comprar
11        E que estou prestes a digitar os dados do meu cartão de crédito
12    Quando eu insiro letras no campo de número do cartão
13        Então nada aparece escrito
14            E uma mensagem me orienta com o formato correto de um número de cartão de crédito
```

# Gherkin - Exemplos



```
12  
13 Cenário: Digitei letras no campo de número do cartão  
14     Dado que eu escolhi os itens que vou comprar  
15         E que estou prestes a digitar os dados do meu cartão de crédito  
16 Quando eu insiro letras no campo de número do cartão  
17     Então nada aparece escrito  
18         E uma mensagem me orienta com o formato correto de um número de cartão de crédito  
19  
20 Cenário: Número do cartão de crédito com poucos dígitos  
21     Dado que eu escolhi os itens que vou comprar  
22         E que estou prestes a digitar os dados do meu cartão de crédito  
23 Quando eu insiro um número de cartão que tem apenas 15 dígitos  
24     Então o botão para confirmar a compra fica desabilitado  
25         E ao tentar clicar nesse botão, uma mensagem me orienta com o formato correto  
26
```

# Gherkin - Exemplos



```
12  
13 Cenário: Digitei letras no campo de número do cartão  
14     Dado que eu escolhi os itens que vou comprar  
15         E que estou prestes a digitar os dados do meu cartão de crédito  
16 Quando eu insiro letras no campo de número do cartão  
17     Então nada aparece escrito  
18         E uma mensagem me orienta com o formato correto de um número de cartão de crédito  
19  
20 Cenário: Número do cartão de crédito com poucos dígitos  
21     Dado que eu escolhi os itens que vou comprar  
22         E que estou prestes a digitar os dados do meu cartão de crédito  
23 Quando eu insiro um número de cartão que tem apenas 15 dígitos  
24     Então o botão para confirmar a compra fica desabilitado  
25         E ao tentar clicar nesse botão, uma mensagem me orienta com o formato correto  
26
```

# Gherkin - Contexto



Usar apenas as palavras reservadas básicas pode criar cenários repetitivos e de leitura massante para os stakeholders.

```
8 # assim incluímos comentários no arquivo
9 Contexto:
10   Dado que eu escolhi os itens que vou comprar
11     E que estou prestes a digitar os dados do meu cartão de crédito
12
13 Cenário: Digitei letras no campo de número do cartão |
14   Quando eu insiro letras no campo de número do cartão
15     Então nada aparece escrito
16       E uma mensagem me orienta com o formato correto de um número de cartão de crédito
17
18 Cenário: Número do cartão de crédito com poucos dígitos
19   Quando eu insiro um número de cartão que tem apenas 15 dígitos
20     Então o botão para confirmar a compra fica desabilitado
21       E ao tentar clicar nesse botão, uma mensagem me orienta com o formato correto
22
```

# Gherkin - Cenário



```
3  Contexto:  
4    Dado que Joana possui $200  
5  
6  Cenário: Cliente possui fundos  
7    Quando ela tentar sacar $100  
8    Então o sistema exibe a mensagem : 'Saque autorizado'  
9  
10 Cenário: Cliente não possui fundos  
11    Quando ela tentar sacar $250  
12    Então o sistema exibe a mensagem : 'Saque não autorizado'  
13  
14 Cenário: Cliente sacou exatamente  
15    Quando ela tentar sacar $200  
16    Então o sistema exibe a mensagem : 'Saque autorizado, mas se liga, que acabou a grana'
```

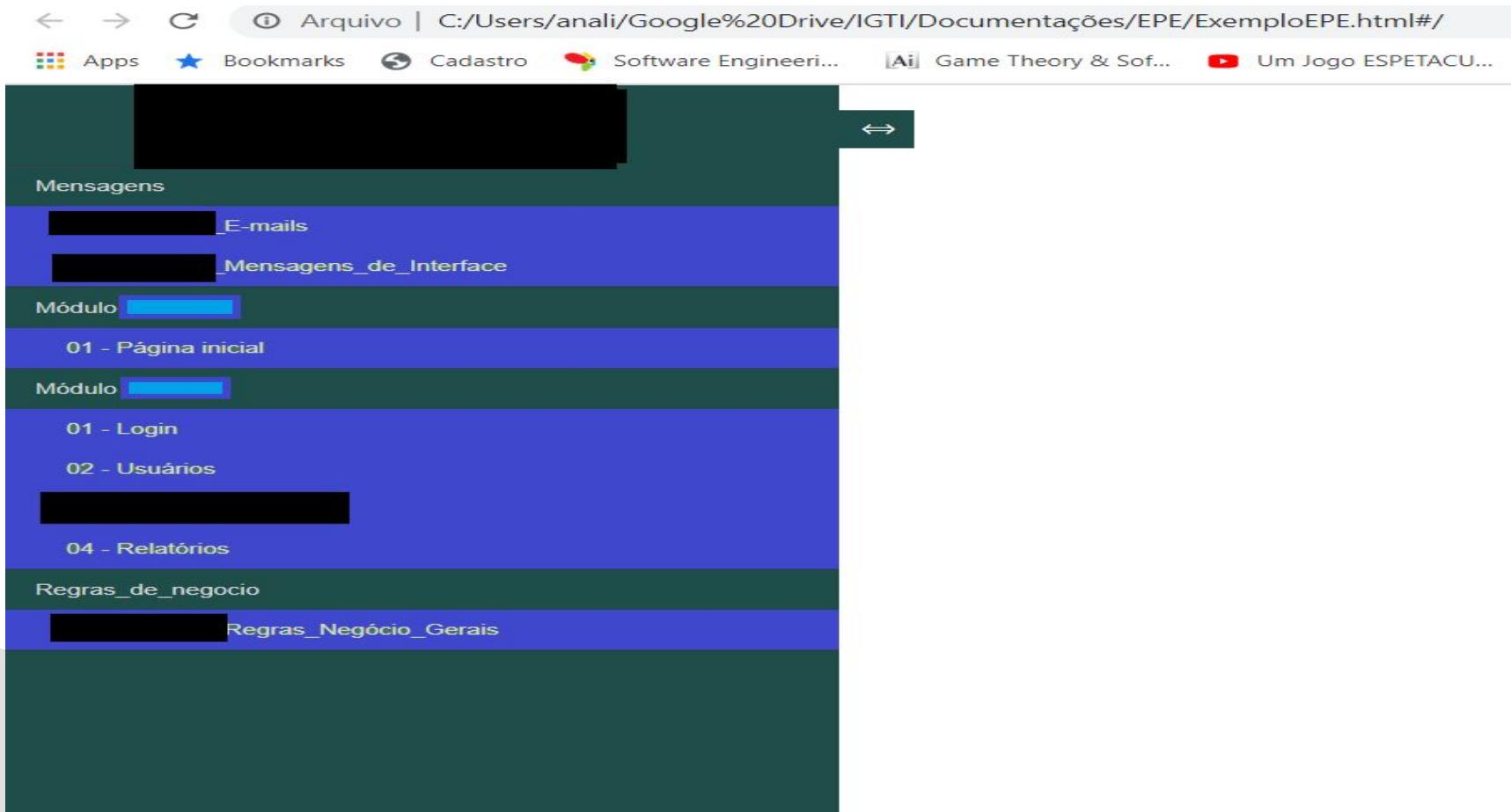
# Gherkin - Cenário



```
2 Contexto:  
3   Dado que Joana possui $200  
4  
5 Esquema de cenário: Mensagens de feedback para saque  
6   Quando ela tentar sacar <Quantidade>  
7   Então o sistema exibe a mensagem : <Mensagem>  
8  
9 Exemplos:  
10  | Quantidade | Mensagem |  
11  | $100      | Saque autorizado |  
12  | $250      | Saque não autorizado |  
13  | $200      | Saque autorizado, mas se liga, que acabou a grana |
```

# EPE

IGTI



# EPE – Exemplo Cadastro Histórico



## US02\_07\_Histórico

Como usuário do sistema Quero manter o histórico de usuários Para que tenha o controle das mudanças dos dados dos usuários

Critérios de Aceitação		
<p><b>Dado</b> que o usuário esteja logado no sistema para executar esta US</p>		
<p><b>E</b> que existam os seguintes usuários já cadastrados</p>		
Usuário	E-mail	Status
[REDACTED]	[REDACTED]	Ativo
[REDACTED]	[REDACTED]	Ativo
[REDACTED]	[REDACTED]	Ativo
[REDACTED]	[REDACTED]	Excluído

Exibição do histórico de usuário
<p><b>Dado</b> que o usuário acesse o menu <b>Usuários</b></p>
<p><b>E</b> o sistema apresente uma lista dos usuários cadastrados em ordem de alteração decrescente conforme <a href="#">RNG-07</a></p>
<p><b>E</b> o usuário acione a opção <b>Visualizar</b> em um usuário cadastrado</p>
<p><b>Quando</b> o usuário aciona a opção <b>Histórico</b></p>
<p><b>Então</b> o sistema exibe o histórico do usuário em ordem decrescente de data/hora</p>

# EPE – Exemplo Cadastro Histórico



## \_Histórico

Como usuário do sistema Quero manter o histórico de usuários Para que tenha o controle das mudanças dos dados dos usuários

Critérios de Aceitação		
<p><b>Dado</b> que o usuário esteja logado no sistema para executar esta US</p>		
<p><b>E</b> que existam os seguintes usuários já cadastrados</p>		
Usuário	E-mail	Status
[REDACTED]	[REDACTED]	Ativo
[REDACTED]	[REDACTED]	Ativo
[REDACTED]	[REDACTED]	Ativo
[REDACTED]	[REDACTED]	Excluído

Exibição do histórico de usuário	
<p><b>Dado</b> que o usuário acesse o menu <b>Usuários</b></p>	
<b>E</b> o sistema apresente uma lista dos usuários cadastrados em ordem de alteração decrescente conforme	RN - 01
<b>E</b> o usuário acione a opção <b>Visualizar</b> em um usuário cadastrado	
<b>Quando</b> o usuário aciona a opção <b>Histórico</b>	

# EPE – Exemplo Cadastrar Usuário



## Cadastrar usuário

**Dado** que o usuário acesse o menu **Usuários**

**E** o sistema apresente uma lista dos usuários cadastrados em ordem de alteração decrescente conforme [RN-02](#)

**E** o usuário ação a opção **Cadastrar**

**Quando** o sistema apresenta a tela de cadastro de usuário

**E** o usuário preenche os seguintes <Campos> de acordo com as [RN 03, RN 04 e RN 05](#)

Campos	Exemplo
Nome	[REDACTED]
E-mail	[REDACTED]
Lotação	[REDACTED]

**E** ação a opção **Salvar**

**Então** o sistema salva os dados do usuário de acordo com as [RN-06](#)

**E** envia o [E-mail 01](#) de acordo com a [RN-07](#)

**E** apresenta a [MSG-01](#)

**E** o sistema retorna para a lista de usuários

# Próximas Aulas



## Próximo Capítulo

- Segurança
- Gestão de Mudanças

# Bootcamp

# Arquiteto de Soluções - Cloud

Módulo 4 - Soluções para Desenvolvimento

Capítulo 5. Terceira Maneira – Segurança e Gestão de Mudanças

**ANALÍA IRIGOYEN**



# Arquiteto de Soluções

---

Aula 5.1. Segurança

ANALIA IRIGOYEN

# Terceira Maneira – Segurança e Gestão de Mudanças



- Segurança
- Gestão de Mudanças

# Terceira Maneira – Segurança e Gestão de Mudanças



- ❑ Segurança
  - Conceitos Básicos
  - DevSecOps ou só DevOps
- ❑ Gestão de Mudanças

# O que é Segurança da Informação

IGTI

- Integridade
- Confidencialidade
- Disponibilidade



# O que é Segurança da Informação



# Organização SI - Responsabilidades

# Aquisição, Desenvolvimento e manutenção de sistemas

# Criptografia



Controle  
de Acesso

## Relação com Fornecedores

## Segurança Física e Ambiental

# Segurança e RH

# Políticas de SI

# Continuidade de negócios

# Segurança de Comunicações

# Segurança de Operações

```
graph TD; Operacao[Operação] --> Dev[Dev]; Operacao --> RH[RH]; Operacao --> AltaDir[Alta Direção]; Operacao --> Juridica[Jurídica]
```

# O que é LGPD



DPO (Data Protection Officer)

Dados Pessoais

Portabilidade

Direito das pessoas singulares

Consentimento

Direito Digital

Âmbito

Notificação

Privacidade



Dev      Operação      RH  
Alta Direção      Jurídica

# LGPD x ISO 27

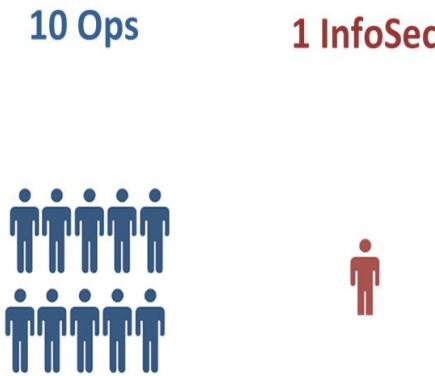
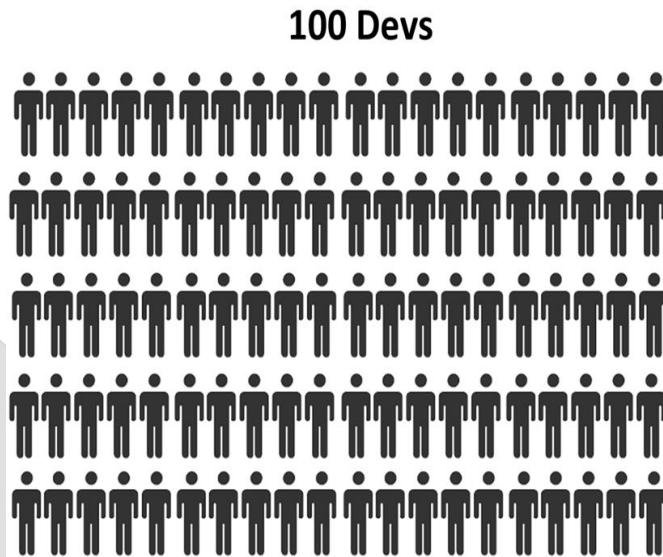


- **Integridade** - Propriedade da exatidão e completeza da informação.
- **Confidencialidade** - a informação não é disponibilizada ou divulgada a indivíduos, entidades ou processos autorizados.
- **Disponibilidade** - Propriedade de ser acessível e utilizável sob demanda por uma entidade autorizada.



Segurança para a privacidade dos dados pessoais - **todo** o ciclo de vida dos dados pessoais, inclusive os processos de negócio - além da **SI**.

# O gargalo da segurança no final do ciclo



Nosso objetivo é que a equipe de segurança participe ativamente desde o início do ciclo de desenvolvimento com grande foco em automatização dos controles: **Conformidade por demonstração**

# O gargalo da segurança no final do ciclo

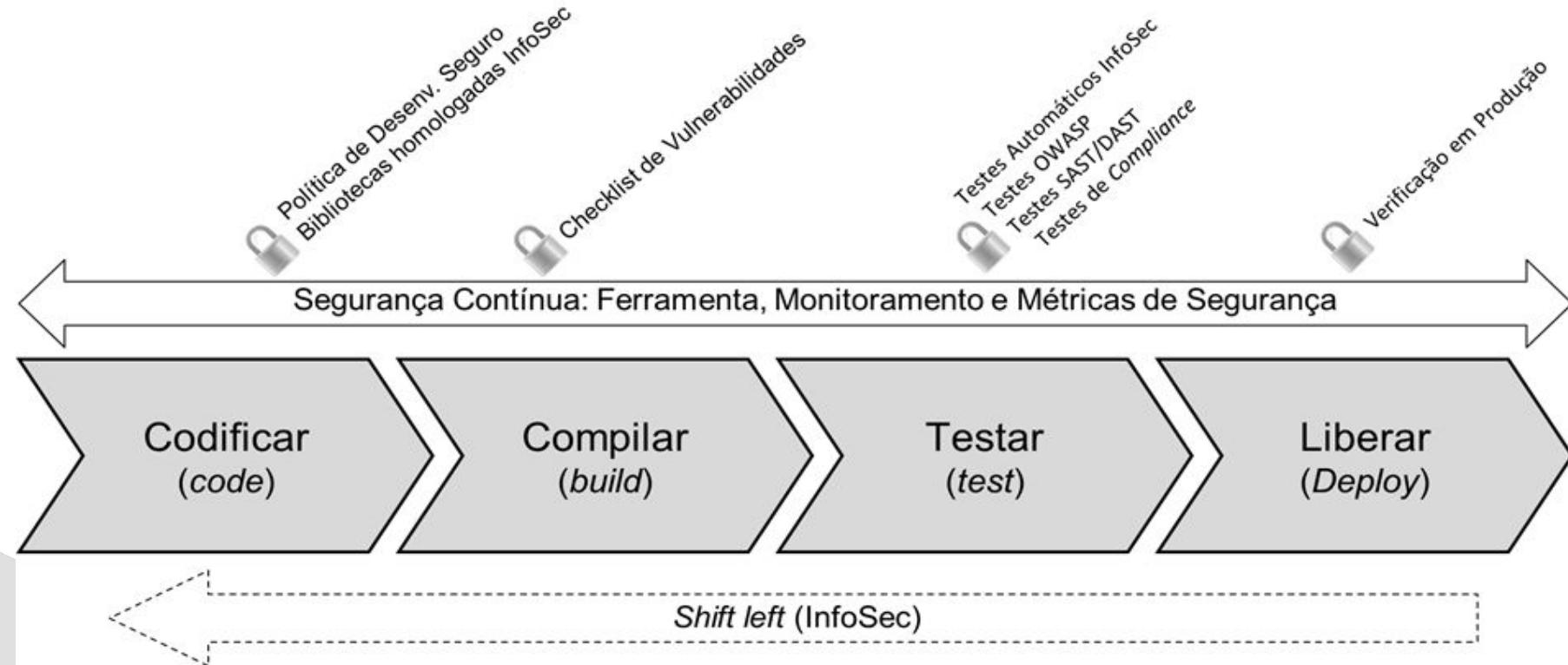


Figura 14.2 - *Shift left* e a Segurança Contínua (Fonte: Rodrigo Santos, Palestra DevSecOps Infnet e AdaptNow, 2018).

# Ações sugeridas para a equipe InfoSec



## Codificar (*code*)

- Política de Desenvolvimento Seguro.
- Bibliotecas homologadas de InfoSec.

## Compilar (*build*)

- *Checklist* de Vulnerabilidades.

## Testar (*test*)

- Testes Automáticos InfoSec.
- Testes OWASP.
- Testes SAST/DAST.
- Testes de *Compliance*.

## Liberar (*Deploy*)

- Verificação em Produção.

# Sete maneiras para melhor integração do Sec com o Dev e Ops



# Dicas de Segurança da Informação



## Desenvolvimento

1. Teste de código.
2. Revisão de código.
3. Teste de penetração.
4. Servidores de IC como código.
5. Análise estática pode ser realizada considerando critérios específicos de segurança, garantindo a remoção dos erros antes mesmo da execução do *build* (*SONAR*, *Veracode*, *Gauntlet* e *manual*).

## Evitar que usuários sem autorização acessem o ambiente:

1. Controle das configurações de ambiente.
2. Teste de ataques de injeção de SQL.
3. Use credenciais de IC somente leitura.
4. Use VM isoladas.

## Ambiente:

1. Testes de vulnerabilidade
2. Testes automatizados Segurança de BD e senhas, por exemplo.

# Dicas de Segurança da Informação



## Telemetria em ambiente

- Alterações de componentes e infraestrutura na nuvem.
- Alterações em configurações diversas (Puppet, Chef, entre outros).
- Alteração de usuários nos grupos privilegiados de admin e InfoSec.
- Erros http no servidor web, como os 4xx e 5xx, ex. 401 - Não autorizado; 403 - Permissão Negada; 502 - Bad Gateway.
- Abertura e fechamento de portas

**ATENÇÃO:** A abertura de uma porta no servidor, por exemplo, pode proporcionar a infecção por trojan, como a porta **80 711** que já é sabido, pode ser explorada por diversos trojans como AckCmd, BlueFire, WebDownloader, entre outros.

# Dicas de Segurança da Informação



## Telemetria em aplicativos

- Alteração de dados cadastrais e dados financeiros como número do cartão de crédito.
- Quantidade de logins mal sucedidos e bloqueio de usuário.
- Acesso a transações e opções bloqueadas para permissão do usuário corrente.
- Solicitação de redefinição de senhas.
- Criar um usuário privilegiado e apagar no mesmo dia sem uma autorização formal

# Exemplos práticos (DevSecOps)



Squad DevSecOps ★

Quadro Gráficos Agenda ...

Objetivos Release e Sprint (Informativo)

+ Adicionar tarefa

Melhoria Contínua... Segurança Organiz... Controle de Acesso... Desenvolvimento e... Continuidade de N...

Release de Abril

Objetivo da Release:  
1) Gestão de Segurança da Área de ...  
! 30/04

Melhoria Contínua... Segurança Organiz... Continuidade de N...

Release Maio

Objetivo da Release:  
1) Conscientização, Educação e Treinamento

Melhoria Contínua... Release Junho

Backlog Melhoria Continua

+ Adicionar tarefa

[SEGINFO]Email de boas vindas [SEG INFO] Criptografia [SEG INFO] Acessos aos servidores Programa Woops, phishing interno Rever a política de BYOD após a

Backlog de mudanças (não envolve infra)

+ Adicionar tarefa

Backlog Auditorias Internas

+ Adicionar tarefa

Desenvolvimento e... Detalhamento de Estórias 24/04 0/3 Melhoria Contínua... Classificação e Con... Definir Base para o SGSI

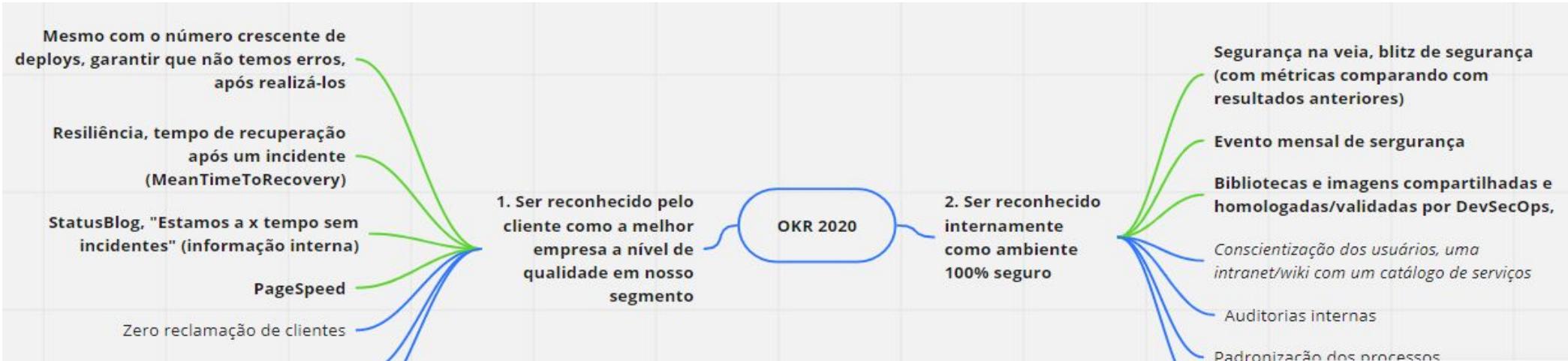
Controle de Acesso... Políticas de desenvolvimento s

SPS

+ Adicionar tarefa

Melhoria Contínua... [Validar] Ajustar o POP de Comunicação Interna 15/05 1 3/3 Melhoria Contínua... Garantir que nosso processo é consistente 15/05 0/2 Melhoria Contínua... Desenvolvimento e... Políticas de desenvolvimento s 1 3/6 Melhoria Contínua...

# Exemplos práticos (DevSecOps)



Objetivo 1	Objetivo 2	Objetivo 3
<p>Ser reconhecido pelo cliente como a melhor empresa a nível de qualidade em nosso segmento</p> <p>KR1: Mesmo com o número crescente de deploys, garantir que não temos erros, após realizá-los</p> <p>KR2: Resiliência, tempo de recuperação após um incidente (MeanTimeToRecovery)</p> <p>KR3: StatusBlog, "Estamos a x tempo sem incidentes" (informação interna)</p> <p>KR4: PageSpeed</p> <p>KR5:</p>	<p>Ser reconhecido internamente como ambiente 100% seguro.</p> <p>KR1: Segurança na veia, blitz de segurança (com métricas comparando com resultados anteriores)</p> <p>KR2: Evento mensal de sergurança</p> <p>KR3: Bibliotecas compartilhadas e homologadas/validadas por DevSecOps</p> <p>KR4:</p> <p>KR5:</p>	<p>Ter processos imperceptíveis viabilizados pela automação</p> <p>DEVSECOPS: % de Automação, SONAR, etc</p> <p>KR1: Pipelines CI/CD, Infra Automatizada, Docker, etc</p> <p>KR2:</p> <p>KR3:</p> <p>KR4:</p> <p>KR5:</p>



# OWASP x SONAR



Categories	漏洞 Vulnerabilities	热点 Security Hotspots		
		Open	In Review	Won't Fix
A1 - Injection	0 A	43	0	0
A2 - Broken Authentication	5 E	0	0	0
A3 - Sensitive Data Exposure	2 C	13	0	0
A4 - XML External Entities (XXE)	0 A	0	0	0
A5 - Broken Access Control	0 A	0	0	0
A6 - Security Misconfiguration	4 E	0	0	0
A7 - Cross-Site Scripting (XSS)	0 A	5	0	0
A8 - Insecure Deserialization	0 A	1	0	0
A9 - Using Components with Known Vulnerabilities	0 A	0	0	0
A10 - Insufficient Logging & Monitoring	0 A	0	0	0
Not OWASP	656 D	0	0	0

Activate Windows

# Próximas Aulas



- Gestão de Mudanças
- AIOps

**Caso Prático (GitHub, Visual Studio 2019, Azure DevOps, Azure Services), principais pontos:**

- Primeira Maneira
- Segunda Maneira
- Terceira Maneira



# Arquiteto de Soluções

---

Aula 5.2. Gestão de Mudanças

ANALIA IRIGOYEN

# Terceira Maneira – Segurança e Gestão de Mudanças



- Segurança
- Gestão de Mudanças

## Próximas Aulas

- AIOps
- Desafio DevOps - AzureDevOps

# Terceira Maneira – Segurança e Gestão de Mudanças



- ❑ Segurança
  - Conceitos Básicos
  - DevSecOps ou só DevOps
- ❑ Gestão de Mudanças

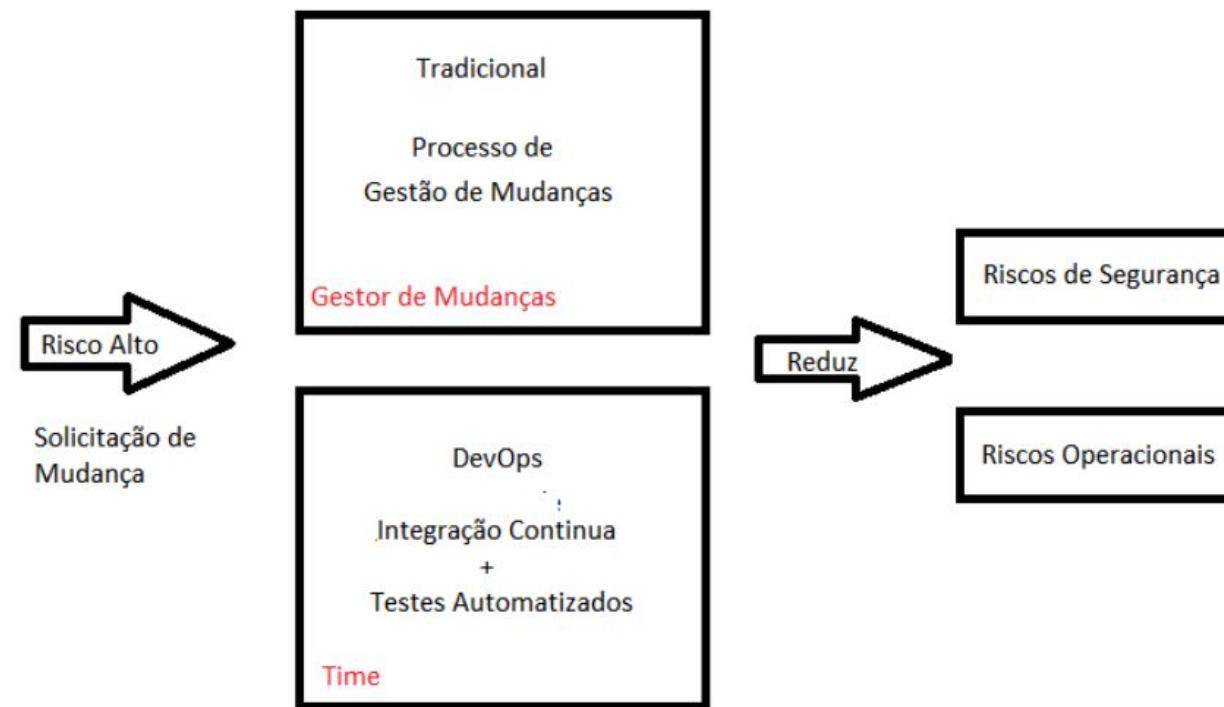
# DevOps é Engenharia de Software



Atividade	Nível E	Nível D	Nível C	Nível B	Nível A
Cultura & Organização	<ul style="list-style-type: none"> <li>Times organizados segundo uma plataforma/tecnologia</li> <li>Processos definidos e documentados</li> </ul>	<ul style="list-style-type: none"> <li>Um backlog por time</li> <li>Adotar metodologias ágeis</li> <li>Remover fronteiras entre os times</li> </ul>	<ul style="list-style-type: none"> <li>Colaboração do time estendida entre equipes</li> <li>Remover fronteiras dev/ops</li> <li>Processo comum para todas as mudanças</li> </ul>	<ul style="list-style-type: none"> <li>Melhoria contínua entre times</li> <li>Times responsáveis por todo o caminho até produção</li> </ul>	<ul style="list-style-type: none"> <li>Times Cross functional (Desenvolvimento e Operações)</li> </ul>
Build & Deploy	<ul style="list-style-type: none"> <li>Controle centralizado de versão</li> <li>Scripts de construção de pacotes automatizados</li> <li>Sem gestão de artefatos</li> <li>Deployment manual</li> <li>Ambientes são manualmente provisionados</li> </ul>	<ul style="list-style-type: none"> <li>Filas de execuções de integração contínua</li> <li>Qualquer build pode ser criado a partir do código-fonte</li> <li>Gerenciamento de artefatos</li> <li>Scripts de deployment automatizado</li> <li>Provisionamento de ambientes automatizado</li> </ul>	<ul style="list-style-type: none"> <li>Execução de integração contínua a partir de triggers</li> <li>Build falha se a qualidade não é atendida (análise de código, desempenho, etc.)</li> <li>Botão de release e deployment (automático)</li> <li>Deployment padrão para todos os ambientes</li> </ul>	<ul style="list-style-type: none"> <li>Prioridades do time em manter o código sempre "deployável" ao invés de melhorias</li> <li>Builds não são deixados quebrados</li> <li>Deployments orquestrados</li> <li>Blue Green Deployments (Dois ambientes com duas versões diferentes)</li> </ul>	<ul style="list-style-type: none"> <li>Entrega contínua com "Zero Touch", sem nenhuma pessoa interferindo</li> </ul>
Release	<ul style="list-style-type: none"> <li>Releases não-frequentes e não-confiáveis</li> <li>Processo manual</li> </ul>	<ul style="list-style-type: none"> <li>Releases não-frequentes e dolorosos, mas confiáveis</li> </ul>	<ul style="list-style-type: none"> <li>Releases não-frequentes, mas totalmente automatizadas e confiáveis em qualquer ambiente</li> </ul>	<ul style="list-style-type: none"> <li>Releases frequentes totalmente automatizadas</li> <li>Deployment desconectado da release</li> <li>Release Canary (Roteamento entre as versões)</li> </ul>	<ul style="list-style-type: none"> <li>Sem realizar nenhum rollback, sempre alterando para frente.</li> </ul>
Gestão de Dados	<ul style="list-style-type: none"> <li>Migração de dados é realizada manualmente, sem scripts</li> </ul>	<ul style="list-style-type: none"> <li>Migração de dados usando scripts de versionamento mas realizados manualmente</li> </ul>	<ul style="list-style-type: none"> <li>Mudanças em banco de dados automatizadas e versionadas</li> </ul>	<ul style="list-style-type: none"> <li>Mudanças em banco de dados realizadas automaticamente como parte do processo de deployment</li> </ul>	<ul style="list-style-type: none"> <li>Alterações e rollback de banco de dados testados automaticamente a cada deployment</li> </ul>
Teste & Verificação	<ul style="list-style-type: none"> <li>Testes de unidade automáticos</li> <li>Ambiente de teste separado</li> </ul>	<ul style="list-style-type: none"> <li>Testes de integração automáticos</li> <li>Análise estática de código</li> <li>Análise de cobertura de teste</li> </ul>	<ul style="list-style-type: none"> <li>Testes funcionais automáticos</li> <li>Testes de desempenho/segurança manuais</li> </ul>	<ul style="list-style-type: none"> <li>Testes de aceitação totalmente automáticos</li> <li>Testes de desempenho/segurança automáticos</li> <li>Testes de exploração manuais baseados em análise de risco em falhas</li> </ul>	<ul style="list-style-type: none"> <li>Testes de verificação valor de negócio esperado</li> <li>Defeitos encontrados e resolvidos imediatamente (roll forward) através de commits automáticos</li> </ul>
Informação & Relatório	<ul style="list-style-type: none"> <li>Métricas de baseline de processos</li> <li>Relatórios manuais</li> <li>Visibilidade apenas para o gerador do relatório</li> </ul>	<ul style="list-style-type: none"> <li>Medição do processo</li> <li>Relatórios automáticos</li> <li>Visibilidade para o time</li> </ul>	<ul style="list-style-type: none"> <li>Geração automática de release notes</li> <li>Rastreabilidade do pipeline de deployment desde o commit</li> <li>Histórico de relatórios</li> <li>Visibilidade entre as equipes</li> </ul>	<ul style="list-style-type: none"> <li>Reportar análise de tendência</li> <li>Gráficos em tempo real nas métricas do pipeline de deployment</li> </ul>	<ul style="list-style-type: none"> <li>Informações dinâmicas e fácil de serem buscadas por qualquer pessoa</li> <li>Dashboards customizados</li> <li>Cruzamentos de informações de vários times diferentes da empresa</li> </ul>

# Gestão de Mudanças

IGTI



# Gestão de Mudanças



Um pipeline de implementação adequado permite que a grande maioria das mudanças seja classificada como baixo risco

## Tipos de Mudança:

**Padrão:** Baixo risco e pré-aprovada

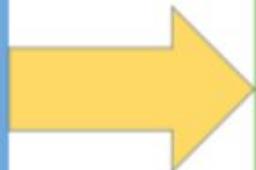
**Normal:** Risco alto e passam pela aprovação de comitê

**Urgente:** Risco alto para resolver incidentes críticos em produção e exigem aprovação da alta direção

# Simplificando a Gestão de Mudanças



Crie uma rotina de visita ao gemba (local onde as situações ocorrem na prática) para sempre conversar com as equipes visando eliminar a necessidade da maioria das mudanças ter que passar pelo comitê



1. Avalie constantemente a real necessidade das informações do formulário de mudanças (RFC)
2. Negocie com os envolvidos a redução das aprovações necessárias e o uso de Pull Request
3. Negocie com a auditoria o aceite de evidências automatizadas do Pipeline de Implementação e entenda a perspectiva deles

# Gerando transparência



- Lista de mudanças realizadas nos últimos três meses.
- Compartilhar a lista completa dos problemas encontrados nestas mudanças.
- Compartilhar os indicadores de mudanças, como por exemplo: tempo médio entre falhas ou tempo médio para reparar uma mudança.
- Demonstrar o controle do ambiente onde são realizadas as implementações e testes automatizados.
- Demonstrar como os erros são controlados e o processo de correção, neste ponto é importante mostrar as ferramentas de automação e o controle de acesso, quando necessário.
- Automatizar o máximo possível as solicitações e gestão de mudanças para que seja fácil mostrar as evidências automatizadas do controle e gerar confiança no processo DevOps para os stakeholders Gestor de *compliance*, auditores e Gestor de Mudanças. Muitas vezes, o diálogo e o treinamento de algumas ferramentas pode gerar a confiança necessária.

# Gerando transparência – Quality Gate



## Quality Gates summary

1 3 7

Some Quality Gates fail. The build can be stopped upon quality gate failure. [Online documentation](#).

Quality Gates that measure diff cannot be run on the baseline. Hence they have blank trend and baseline status.

Name	Trend	Baseline Value	Value	Group
Percentage Coverage		71.67 %	67.91 %	Project Rules \ Quality Gates
Percentage Coverage on New Code			70.73 %	Project Rules \ Quality Gates
Percentage Coverage on Refactored Code			69.52 %	Project Rules \ Quality Gates
Blocker Issues	=	0 issues	0 issues	Project Rules \ Quality Gates
Critical Issues	=	4 issues	4 issues	Project Rules \ Quality Gates
New Blocker / Critical / Major Issues			367 issues	Project Rules \ Quality Gates
Critical Rules Violated		7 rules	9 rules	Project Rules \ Quality Gates
Percentage Debt		13.09 %	14.39 %	Project Rules \ Quality Gates
New Debt since Baseline			7.2 man-days	Project Rules \ Quality Gates
Debt Rating per Namespace		6 namespaces	8 namespaces	Project Rules \ Quality Gates
New Annual Interest since Baseline			17.57 man-days	Project Rules \ Quality Gates

Showing 1 to 11 of 11 entries

# Exemplos práticos (Gestão de Mudanças)



USER STORY 88407

88407 [REDACTED]

1 comment Add tag Save & Close Follow ...

State: Doing Area: [REDACTED] Updated by: [REDACTED]

Reason: Moved out of state ... Iteration: [REDACTED]

Riscos de Segurança

Possibilidade de vazamento de dados (por código inseguro ou por inadequação de funcionalidade)

Impacto por vazamento de dados (por código inseguro ou por inadequação de funcionalidade)

Estratégia para Mitigação dos Riscos

Quando o resultado do Impacto for alto o Head de Desenvolvimento e Produto deve aprovar a mudança. Os impactos Médios e Baixos podem considerar a aprovação do PR como aprovação da mudança.

Tratamento de Dados

1) Existe a necessidade real de manipulação de dados sensíveis?  
 False

2) Há preocupação com quem vai manipular a informação e foi dado um controle de acesso específico?  
 False

3) Existem funcionalidades obrigatórias para esses dados? Quais? Ex: Logs, Motivo, Quem, Quando?

4) Foi definida a criptografia dos dados ou autenticação de 2 fatores? Qual?

5) Existem funcionalidades que gravam log de ações de segurança? Ex: dar/retirar acesso para ler/editar dados  
 False

6) As senhas armazenadas de forma seguras implementadas em todos os apps (int/ext)  
 False

7) Monitoração/log/restrição de Edição/inclusão/consulta a dados sensíveis  
 False

# Políticas de Pull Request



Policies for: [REDACTED] > [ all repositories ] > develop

Save changes  Discard changes

## Protect this branch

- Setting a Required policy will enforce the use of pull requests when updating the branch
- Setting a Required policy will prevent branch deletion

### Require a minimum number of reviewers

Require approval from a specified number of reviewers on pull requests.

Minimum number of reviewers

1

- Allow requestors to approve their own changes
- Prohibit the most recent pusher from approving their own changes
- Allow completion even if some reviewers vote to wait or reject
- Reset code reviewer votes when there are new changes

### Check for linked work items

Encourage traceability by checking for linked work items on pull requests.

Policy requirement

Required

Block pull requests from being completed unless they have at least one linked work item.

Optional

Warn if there are no linked work items, but allow pull requests to be completed.

### Check for comment resolution

Check to see that all comments have been resolved on pull requests.

### Limit merge types

Control branch history by limiting the available types of merge when pull requests are completed.

Allowed merge types:

- Basic merge (no fast-forward)  
Preserves nonlinear history exactly as it happened during development.
- Squash merge  
Creates a linear history by condensing the source branch commits into a single new commit on the target branch.
- Rebase and fast-forward  
Creates a linear history by replaying the source branch commits onto the target without a merge commit.
- Rebase with merge commit  
Creates a semi-linear history by replaying the source branch commits onto the target and then creating a merge commit.

### Build validation

Validate code by pre-merging and building pull request changes

+ Add build policy

AUTOMATICALLY INCLUDE CODE REVIEWERS

Include specific users or groups in the code review based on which files changed.

+ Add automatic reviewers

Reviewer(s)	Requirement	Path filter
GrupoDEVs	Required	No filter
GrupoLíderes	Optional	No filter
HEAD DEV	Optional	No filter

E  
E  
E

# Políticas de Pull Request



Policies for: [REDACTED] > [ all repositories ] > develop

Save changes  Discard changes

## Protect this branch

- Setting a Required policy will enforce the use of pull requests when updating the branch
- Setting a Required policy will prevent branch deletion

## Require a minimum number of reviewers

Require approval from a specified number of reviewers on pull requests.

Minimum number of reviewers

1

Allow requestors to approve their own changes

Prohibit the most recent pusher from approving their own changes

Allow completion even if some reviewers vote to wait or reject

Reset code reviewer votes when there are new changes

## Check for linked work items

Encourage traceability by checking for linked work items on pull requests.

Policy requirement

Required

Block pull requests from being completed unless they have at least one linked work item.

Optional

Warn if there are no linked work items, but allow pull requests to be completed.

## Check for comment resolution

Check to see that all comments have been resolved on pull requests.

## Limit merge types

Control branch history by limiting the available types of merge when pull requests are completed.

Allowed merge types:

Basic merge (no fast-forward)

Preserves nonlinear history exactly as it happened during development.

Squash merge

Creates a linear history by condensing the source branch commits into a single new commit on the target branch.

Rebase and fast-forward

Creates a linear history by replaying the source branch commits onto the target without a merge commit.

Rebase with merge commit

Creates a semi-linear history by replaying the source branch commits onto the target and then creating a merge commit.

## Build validation

Validate code by pre-merging and building pull request changes

+ Add build policy

### AUTOMATICALLY INCLUDE CODE REVIEWERS

Include specific users or groups in the code review based on which files changed.

+ Add automatic reviewers

Reviewer(s)	Requirement	Path filter
GrupoDEVs	Required	No filter
GrupoLíderes	Optional	No filter
HEAD DEV	Optional	No filter

E  
E  
E

# Próximas Aulas



- AIOPS
- Caso prático (GitHub, Visual Studio 2019, Azure DevOps, Azure Services), principais pontos:
  - Introdução
  - Primeira Maneira
  - Segunda Maneira
  - Terceira Maneira



# Bootcamp

# Arquiteto de Soluções - Cloud

Módulo 4 - Soluções para Desenvolvimento

Case Prático

**ANALÍA IRIGOYEN**



# Arquiteto de Soluções

---

Introdução

ANALIA IRIGOYEN

# Case Prático - DevOps

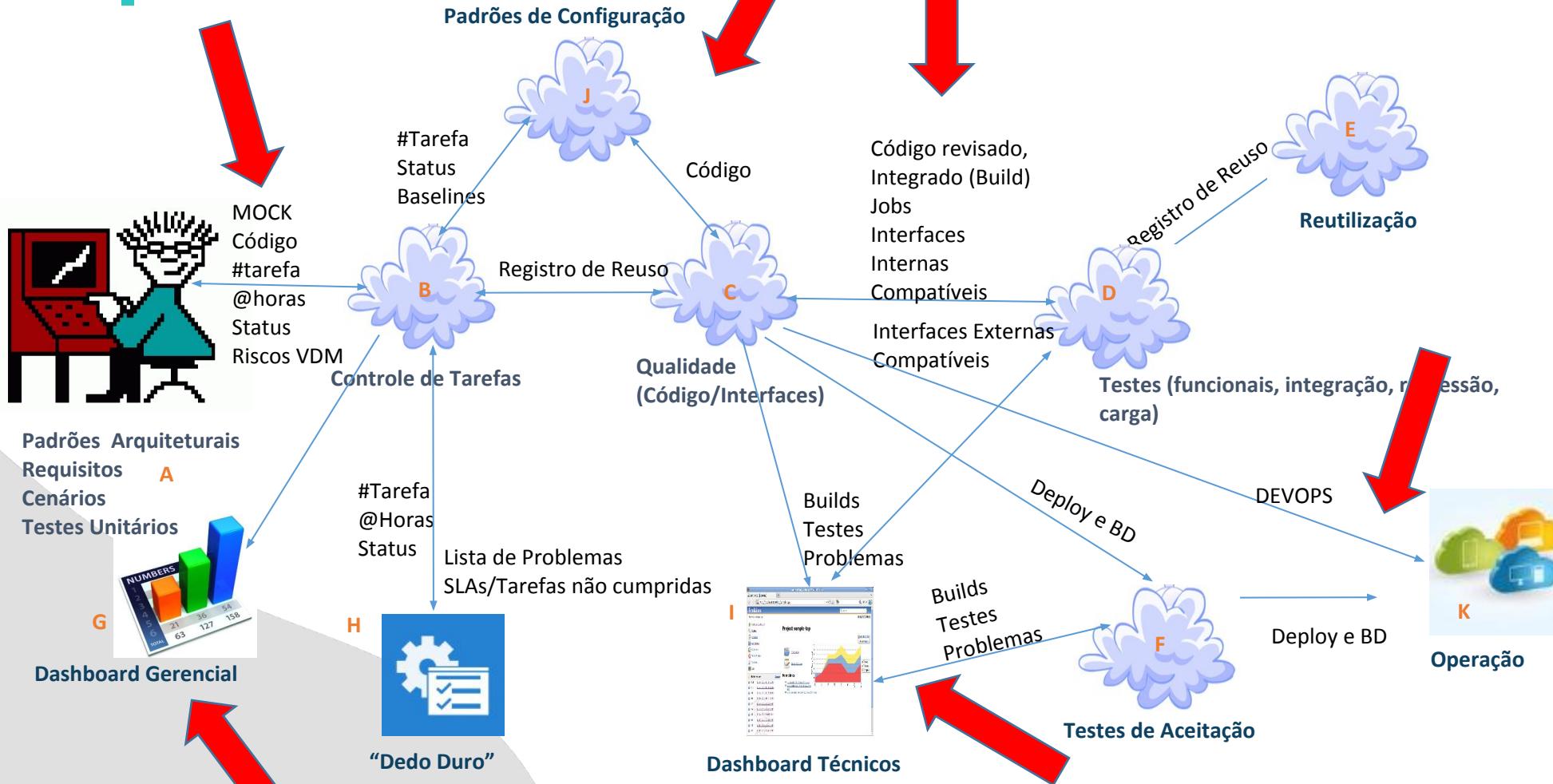


## □ Nosso Case (Visual Studio, GitHub, Azure DevOps, Azure Services)

- Introdução
- Primeira Maneira – Exercício
- Segunda Maneira – Exercício
- Terceira Maneira – Exercício

# Nosso Case

IGTI



# Nosso Case



Primeira Maneira -> Acelerar o fluxo dos desenvolvedores (esquerda) para a operação e cliente (direita) (1/2):

- Tornar o trabalho visível e reduzir o tamanho dos lotes e intervalos usando o template Scrum AzureDevOps.
- Entender a teoria das Restrições e como otimizar o fluxo: remover desperdícios; limitar o trabalho em andamento: usando os Dashboards do Azure DevOps.
- Reduzir o número de transferências (handoff) - Montagem das etapas do quadro no AzureDevOps.
- Conhecer infra como código e self-service – (Pipeline CI – AzureDevOps e Azure Services)
- Incorporar qualidade na origem (Critérios de Ready/Done, Testes Automatizados, análise de segurança)

# Nosso Case



Segunda Maneira (Rápido feedback em todos os estágios do fluxo de valor-direita para a esquerda) :

- Qualidade próxima da fonte (menos aprovações)
  - a. Revisão de Código (PR)
- Telemetria como informação para todos (explorar as métricas básicas no Azure Services) e melhorar o processo.

# Nosso Case



Terceira Maneira (Cultura de alta confiança que permite correr riscos e potencializar o aprendizado contínuo):

- Cultura de alta confiança que permite correr riscos e potencializar o aprendizado contínuo
  - Como incluir uma retrospectiva no Azure DevOps
  - Fazer uma dinâmica de Post-mortem com ConceptBoard

Difundir conhecimento usando testes automatizados

- Difundir conhecimento usando o BDD (usar a Wiki e estrutura para incluir o BDD)

# Nosso Case



## □ Primeiros Passos

- Ter uma conta gratuita Azure DevOps
- Ter uma conta gratuita Azure Services
- Ter uma conta no GitHub

# Próximas Aulas



**Caso Prático (GitHub, Visual Studio 2019, Azure DevOps, Azure Services)**

- Primeira Maneira**
- Segunda Maneira**
- Terceira Maneira**



# Arquiteto de Soluções

---

Primeira Maneira (Parte I)

ANALIA IRIGOYEN

# Case Prático - DevOps



## □ Nosso Case

- Introdução
- Primeira Maneira – Parte I
  - ❖ Tornar o trabalho visível e reduzir o tamanho dos lotes e intervalos usando o template Agile AzureDevOps.
  - ❖ Usando Dashboards no AzureDevOps
  - ❖ Customizando os Quadros no AzureDevOps
- Primeira Maneira – Parte II
- Segunda Maneira – Exercício
- Terceira Maneira – Exercício

# Azure DevOps



- Application Lifecycle Management, expressão utilizada para descrever a forma como uma aplicação é criada, construída, desenvolvida e aprimorada.
- Inicialmente utilizada pela Microsoft para designar os processos de criação, desenvolvimento e atualização a que o software “Visual Studio” fora submetido na versão 2010.
- Em uma visão mais prática é o ferramental, processo e pessoas envolvidas para apoiar todo o desenvolvimento do software nas organizações.

# Azure DevOps



- É basicamente focado nas principais atividades de programação que compreendem o início da vida de uma aplicação, a qual começa com a necessidade da existência de tal aplicação e termina quando a aplicação é criada e entregue
- É uma ferramenta localizada na nuvem derivada do VSTS (Visual Studio Team Services)
- É aberto e extensível com milhares de add-ins no Visual Studio Marketplace (<https://marketplace.visualstudio.com/azuredevops>)

# Azure DevOps



- Consegue se integrar a qualquer tipo de aplicação em qualquer plataforma, como GitHub, Azure e AWS
- Totalmente integrado ao GIT
- Por exemplo, pode ser compilado e testado um serviço em node.js a partir do Github, e instalado no AWS ou em container Docker no Azure.
- Suporta configurações públicas e privadas de nuvem

# Azure DevOps



- As principais áreas envolvidas no Azure DevOps:
  - Controle de fonte: suporta gerência de versionamento de artefatos de software
  - Controle de Projetos: suporta planejamento e monitoramento de projetos, defeitos de código, issues e mais
  - DevOps: suporta build, teste e release contínuo

# Azure DevOps

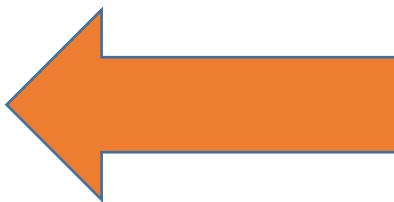


- <https://azure.microsoft.com/pt-br/services/devops/>
- Documentação
  - <https://docs.microsoft.com/en-us/azure/devops/index?view=vsts>

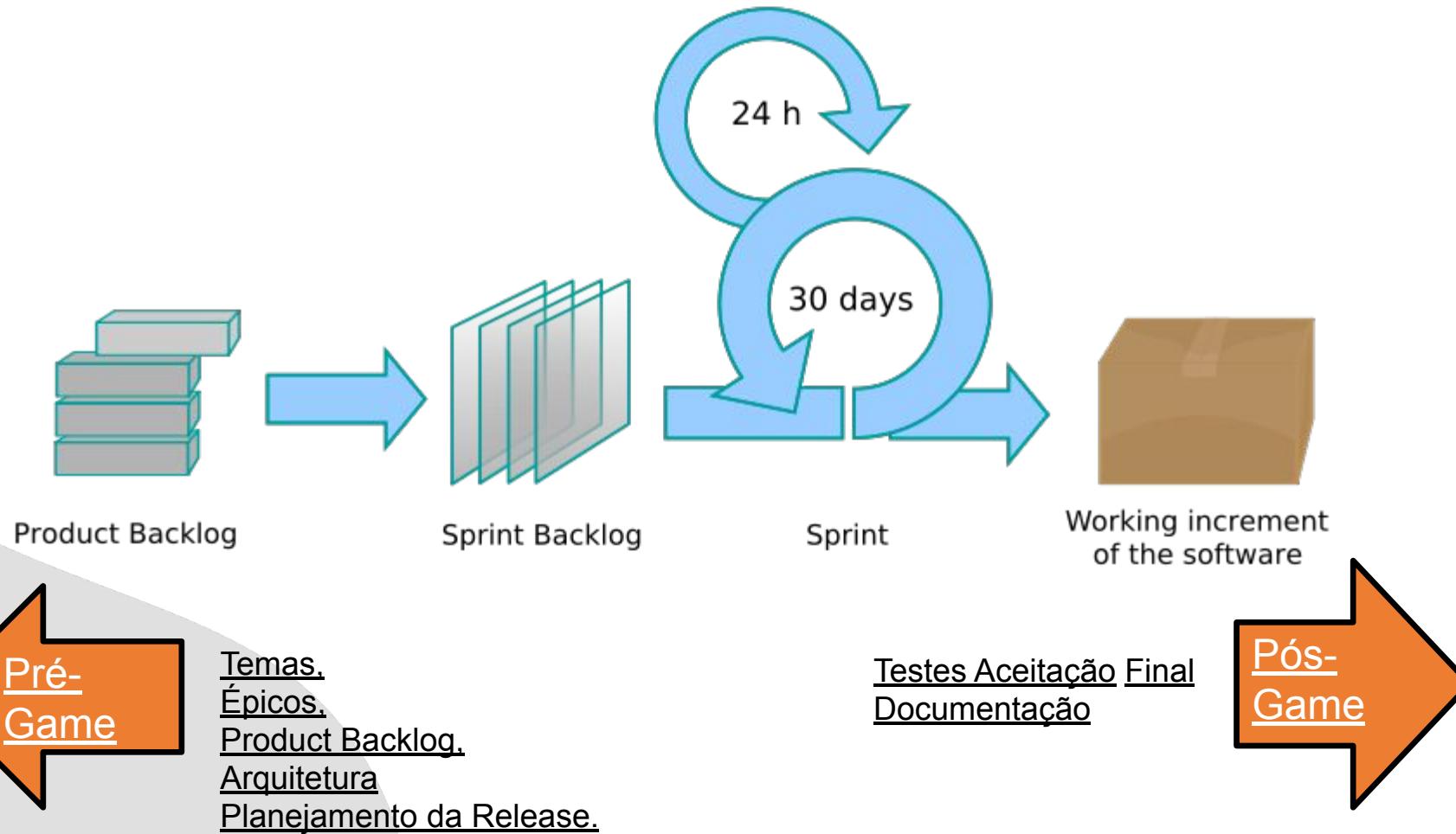
# Azure DevOps



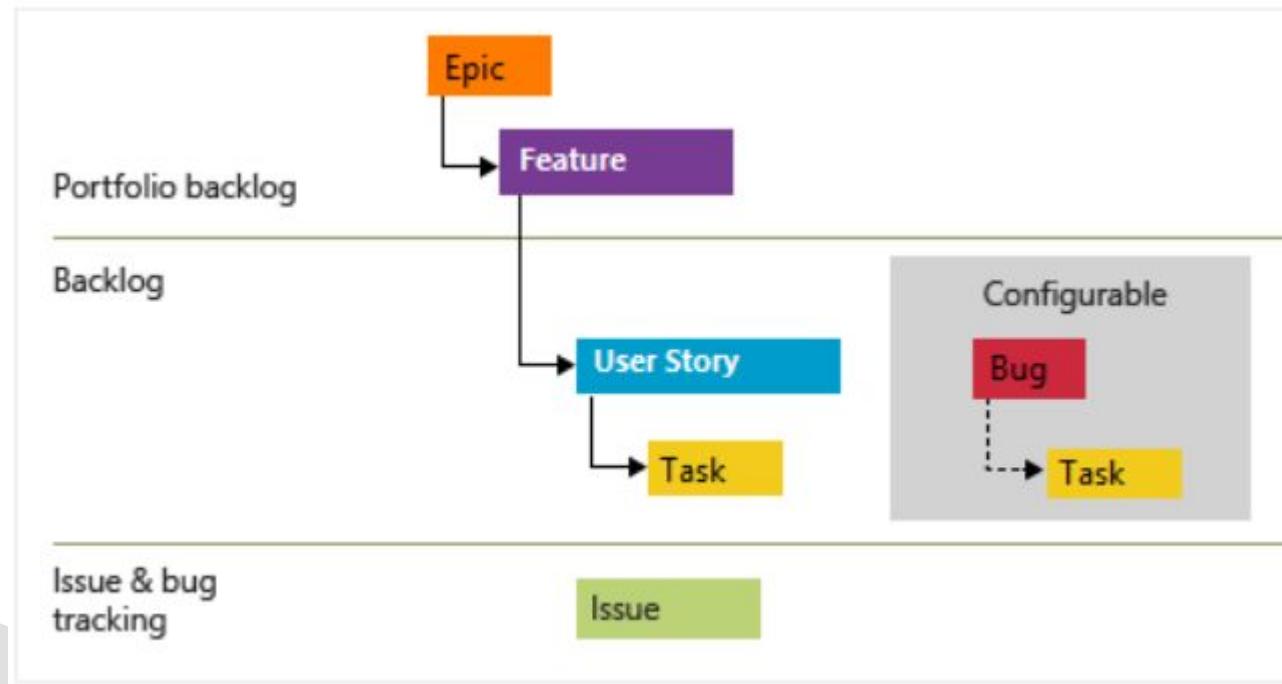
## ❑ Processo e Templates

- ❖ Agile
  - ❖ Scrum
  - ❖ CMMI
- 
- A large, solid orange arrow points from the bottom towards the top, positioned between the Agile and Scrum items in the list.

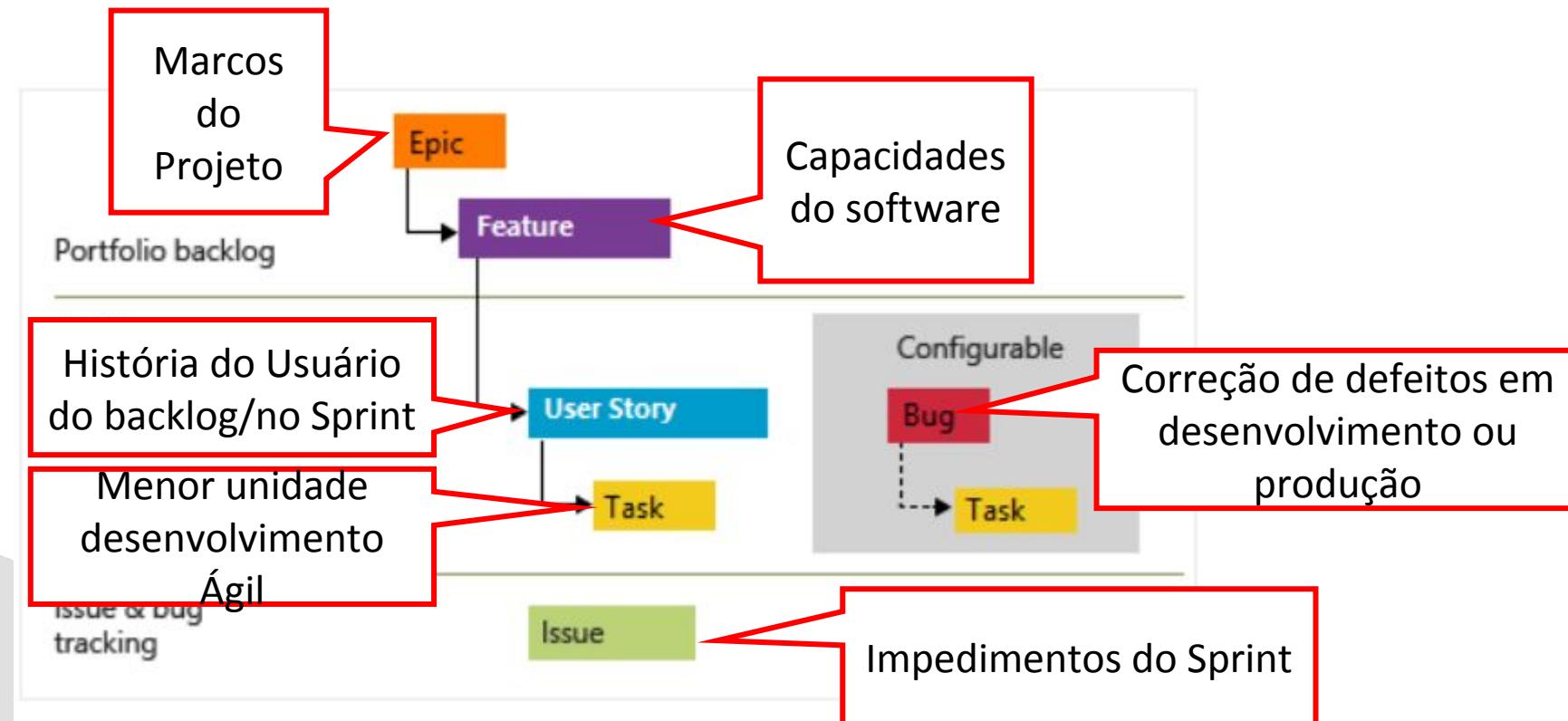
# Azure DevOps



# Azure DevOps - Agile



# Azure DevOps - Agile



# Azure DevOps

**promove**

Projects My work items My pull requests

CM

Contact Manager

Projeto exemplo de gerenciamento de contatos.



PS

ProMove Servicos

Serviços da ProMove.



Criação de  
Projetos

+ Create project

Filter projects

**IGTI**

# Azure DevOps



Create new project X

Project name \*

Description

Visibility

Public ⓘ  
Anyone on the internet can view the project. Certain features like TFVC are not supported.

Private  
Only people you give access to will be able to view this project.

Public projects are disabled for your organization. You can turn on public visibility with [organization policies](#).

Advanced

Version control ⓘ

Work item process ⓘ

Git

Agile

# Azure DevOps



The screenshot shows the Azure DevOps Contact Manager interface. On the left is a navigation sidebar with the following items:

- CM Contact Manager
- Overview
- Summary
- Dashboards
- Wiki
- Boards
- Repos
- Pipelines
- Test Plans
- Artifacts

Red callout boxes with arrows point from specific text descriptions to their corresponding items in the sidebar:

- "Visão Geral do Projeto" points to the "Overview" item.
- "Controle do Projeto com Kanban e Dashboards" points to the "Boards" item.
- "Controle de CI/CD em cloud" points to the "Pipelines" item.
- "Gestão de pacotes nuget/npm e Maven" points to the "Artifacts" item.
- "Repositório de arquivos do projeto" points to the "Repos" item.
- "Toda gestão de testes e testes exploratórios" points to the "Test Plans" item.

# Azure DevOps – Configurações



- Criar Histórias Entregas (Modulo Questões, Modulo Administrativo, Módulo do Jogo...)
- Criar Features ( Questões de PMP, Questões ITIL V3)
- Criar Tarefas ( Cadastrar Pergunta PMP, Cadastrar Respostas PMP, Próxima Página)
- Criar um bug e passar pelos campos

# Azure DevOps – Dashboards



- Criar Board e suas visões
- Customizar Quadros

# Nosso Case



Primeira Maneira -> Acelerar o fluxo dos desenvolvedores (esquerda) para a operação e cliente (direita) (1/2):

- Conhecer infra como código e self-service – (Pipeline CI – AzureDevOps e Azure Services)
- Incorporar qualidade na origem (Critérios de Ready/Done, Testes Automatizados, análise de segurança)

# Nosso Case



Segunda Maneira (Rápido feedback em todos os estágios do fluxo de valor-direita para a esquerda) :

- Qualidade próxima da fonte (menos aprovações)
  - a. Revisão de Código (PR)
- Telemetria como informação para todos (explorar as métricas básicas no Azure Services) e melhorar o processo.

# Nosso Case



Terceira Maneira (Cultura de alta confiança que permite correr riscos e potencializar o aprendizado contínuo)):

- Cultura de alta confiança que permite correr riscos e potencializar o aprendizado contínuo
  - Como incluir uma retrospectiva no Azure DevOps
  - Fazer uma dinâmica de Post-mortem com ConceptBoard

Difundir conhecimento usando testes automatizados

- Difundir conhecimento usando o BDD (usar a Wiki e estrutura para incluir o BDD)



# Arquiteto de Soluções

---

Primeira Maneira (Parte II)

ANALIA IRIGOYEN

# Case Prático - DevOps



## □ Nosso Case

- Introdução
- Primeira Maneira – Parte II
  - ❖ Usando Dashboards, Capacidade e WIP
  - ❖ Incorporar qualidade na origem (Critérios de Ready/Done, Testes Automatizados, análise de segurança)
- Primeira Maneira – Parte III
  - ❖ Subindo o Código Exemplo no GitHub
  - ❖ Github x Azure DevOps.
  - ❖ Configurando a Pipeline CI (YAML)

# Dashboards

IGTI

Criação de  
Dashboard

Azure DevOps marceloncosta / Treinamento Azure DevOps / Overview / Dashboards Search   

**TD Treinamento Azure De...** + 

**Dashboards** 

Name	Team	Description
My favorite dashboards	No favorites yet! Favorite  dashboards to quickly access...	
Treinamento Azure DevOps Team (1)	 Overview  Treinamento Azure DevOps Team	
All dashboards	 Overview  Treinamento Azure DevOps Team	

# Dashboards



Treinamento Azure DevOps Team Over...

Work assigned to Marcelo Costa (5)

2 Task 1 Feature 2 Other

ID	State	Title
38	New	CRUD do módulo de clientes
37	New	Módulo de clientes
39	New	Tela para cadastrar clientes
40	New	Alterar clientes
42	New	Realizar cadastro de clientes

New Work Item

Enter title

Bug

Create

Query Results

Build History

Configure widget

Gears icon

Chart for Work Items

Configure widget

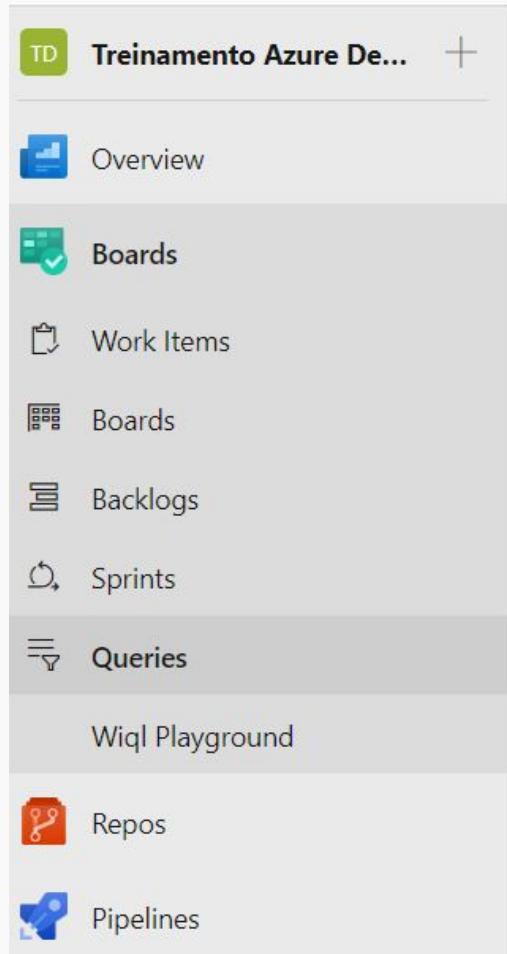
Gears icon

Iteration 1

26 de novembro - 26 de dezembro

# Dashboards

IGTI



# Dashboards



Queries > My Queries > Untitled query

Results Editor Charts > Save query Revert changes Column options Save items Email query Copy query URL

Type of query Flat list of work items

5 work items  
1 selected

Query across projects

Salvando a query

New query

Name \* Enter name

Folder \* My Queries

OK Cancel

Value

[Any] [Any] @Me

Save query as

Name \* Meus Work Items

Folder \* Shared Queries

OK Cancel

Compartilhada permite ser adicionada ao Dashboard

# Dashboard



Treinamento Azure DevOps Team Over... ▾ ★ 8 Edit Refresh

Work assigned to [REDACTED]

2 Task 1 Feature 2 Other

ID	State	Title
38	New	CRUD do módulo de clientes
37	New	Módulo de clientes
39	New	Tela para cadastrar clientes
40	New	Alterar clientes
42	New	Realizar cadastro de clientes

New Work Item

Enter title

Bug

Create

Build History

Configure widget



Chart for Work Items

Configure widget



Configuran  
do para  
uma  
consulta

# Dashboard



Configuration

Title: Meus Work Items by Work Item Type

Size: 2 x 2

Query: Meus Work Items

Chart type: Column

Group by: Work Item Type

Aggregation: Count of work items

Save Cancel

Meus Work Items by Work Item Type

Task User Story Feature Epic

Iteration 1  
26 de novembro - 26 de dezembro

Configurando para uma consulta

Work Item Type	Count
Task	2
User Story	1
Feature	1
Epic	1

# Capacidade da Equipe



⌚ Treinamento Azure DevOps Team ▾ ⭐ ⓘ

26 de novembro - 26 de dezembro  
21 work days remaining

Taskboard Backlog Capacity | + Add user Save Undo ...

Iteration 1 ▾ ⚙️⚙️ ⚙️

User	Days off	Activity	Capacity per day
[Redacted]	0 days	Development ▾	8

Team days off 0 days These days off apply to the whole team.

# BDC

IGTI



# WIP

IGTI

Settings

Cards

Fields

Styles

Tag colors

Annotations

Tests

Board

Columns

Swimlanes

Card reordering

Charts

Cumulative flow

General

Backlogs

Columns

Columns visualize the flow of work across the board.

+ Column

New ... Active ... Resolved ... Closed ...

Column name

Name Active

Work in progress limit  
Setting it to 0 specifies no limit.

WIP limit 5

Split column into doing and done

State mapping  
Specify the state this column maps to.

Save Cancel

# Case Prático - DevOps



## □ Exemplos de Critérios de Ready

- 1. Foi refinada no Grooming?
- 2. Foi descrita completamente, de acordo com os padrões de refinamento?
- 3. Possui detalhes suficientes para ser estimada?
- 4. Permite o entendimento claro do que precisa ser feito?
- 5. O tamanho da estória permite o desenvolvimento em um tempo aceitável?
- 6. Foram considerados requisitos não funcionais (ex: desempenho, usabilidade, segurança)?
- 7. Foram considerados interfaceamentos com outros sistemas e/ou módulos?

## □ INVEST

# Case Prático - DevOps



## Exemplos de Critérios de Done

- 1. 100% Testes unitários automatizados - ambiente de testes
- 2. SONAR
- 3. PR aprovado
- 4. Testes de Segurança

# Case Prático - DevOps



- Checklist de Segurança
  - Configurações da Organização
  - Boards – Processo
    - ◆ Criar um processo herdado
    - ◆ Escolher feature
    - ◆ Criar campos de Segurança
    - ◆ Propriedades – Projeto – Troca o template para o criado em Descrição

# Próximas Aulas



## Parte prática III

- Subir Código Exemplo no GitHub
- Github x Azure DevOps
- Criar a Pipeline CI
- Criar Ambiente Azure Services
- Criar a Release no Azure DevOps



# Arquiteto de Soluções

---

Primeira Maneira (Parte III)

ANALIA IRIGOYEN

# Case Prático - DevOps



## □ Nosso Case

- Primeira Maneira – Parte II
  - ❖ Usando Dashboards, Capacidade e WIP
  - ❖ Incorporar qualidade na origem (Critérios de Ready/Done, Testes Automatizados, análise de segurança)
  - ❖ Subindo o Código Exemplo no GitHub
  - ❖ Github x Azure DevOps.
  - ❖ Configurando a Pipeline CI (YAML)
  - ❖ Configurando Azure Services (Ambiente WEB)
  - ❖ Configurando uma Release

# Próximas Aulas



Segunda Maneira (Rápido feedback em todos os estágios do fluxo de valor-direita para a esquerda) :

- Qualidade próxima da fonte (menos aprovações)
  - a. Revisão de Código (PR)
  - b. BDD (somente a escrita não a codificação)
- Telemetria como informação para todos (explorar as métricas básicas no Azure DevOps e no Azure Services) e melhorar o processo.

# Próximas Aulas



Terceira Maneira (Cultura de alta confiança que permite correr riscos e potencializar o aprendizado contínuo)):

- Cultura de alta confiança que permite correr riscos e potencializar o aprendizado contínuo
  - Como incluir uma retrospectiva no Azure DevOps
  - Fazer uma dinâmica de Post-mortem com ConceptBoard

Difundir conhecimento usando testes automatizados

- Difundir conhecimento usando o BDD (usar a Wiki e estrutura para incluir o BDD)