

Iniciando em DevOpS

O início da jornada, uma visão holística



O JEITO FEMININO DE FALAR DE TI

FACILITADORA: ALESSANDRA MARTINS

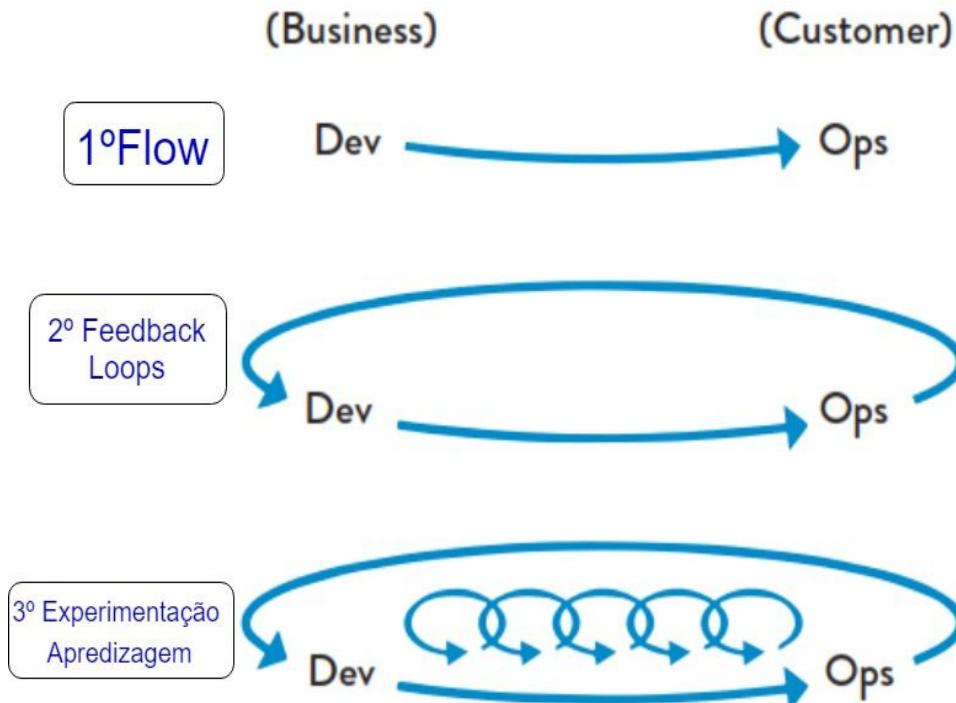
29/ 06 /2019
SÃO PAULO

Roteiro

1. Contexto
2. Definições de Engenharia & Arquitetura
3. Arquitetura de Software X Arquitetura DevOps - Diferenças
4. Conceitos, Princípios por dentro da Cultura DevOps
5. DevOps Versus Outros Modelos
6. Diferença Entre DevOps X CI X CD X DC
7. Fases de Implantação do DevOps + Processos + Papéis
8. Níveis de Maturidade DevOps – Como medir?
9. Visão Geral das Possíveis Ferramentas
10. Pipeline DevOps com Ferramentas
11. Por onde Começar? Governança – Padrões e Boas Práticas
12. DevOps e outras Boas Práticas (ITIL e COBIT)
13. CheckList CI e CD
14. Gitflow – Entendendo um pouco sobre Branch, Git Flow, Versionamento
15. Monitoração Contínua
16. DEVSECOPs – Adicionando Segurança ao Ciclo de Desenvolvimento e Operações
17. Mind Map e extras

CONTEXTO: FORMAS

DEVOPS 3 Formas:



CONTEXTO: Importância da Adoção da Agilidade

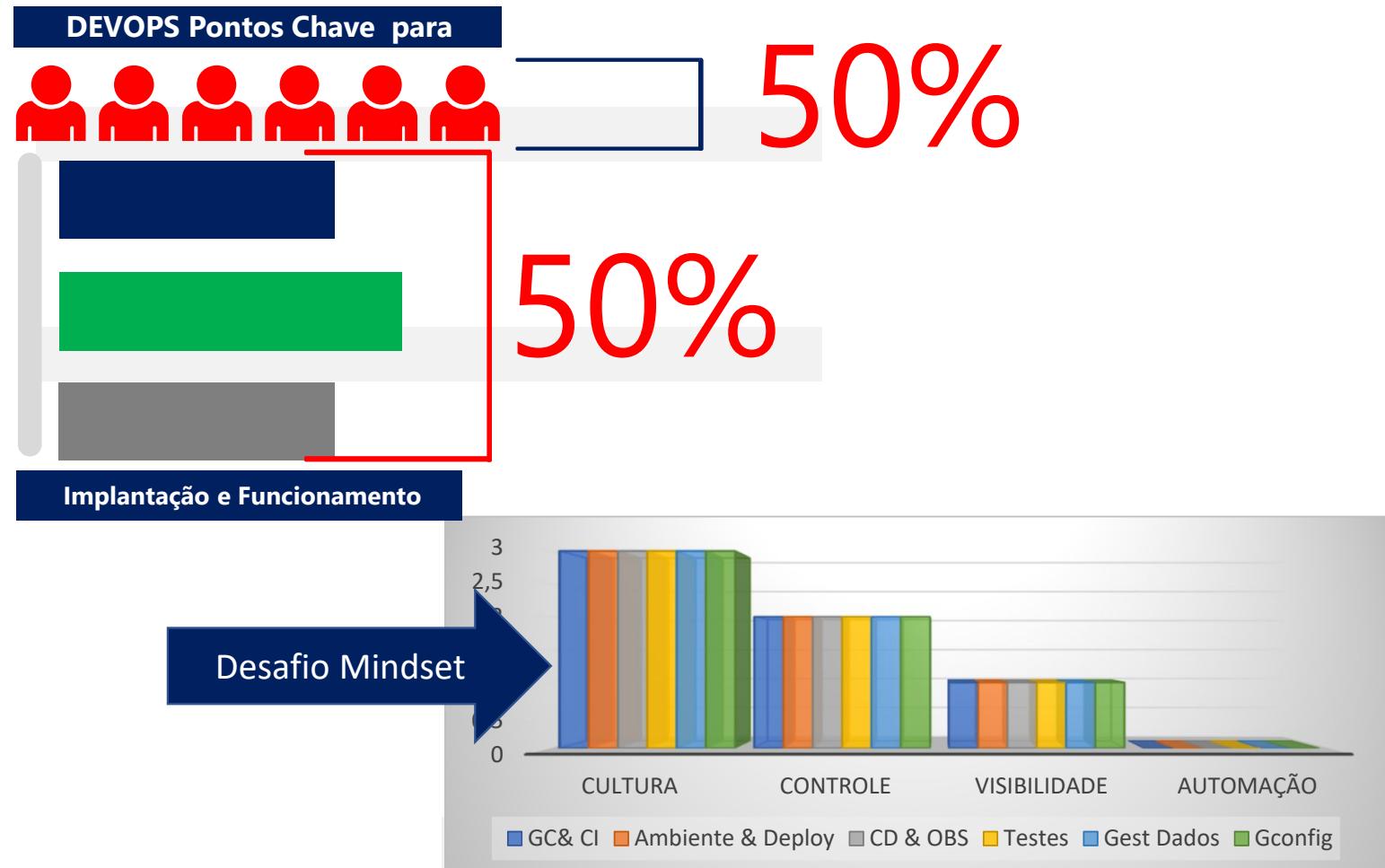
Pontos Chaves para uma Implementação de Sucesso

- 1 Mentalidade
- 2 Valores e Princípios
- 3 Processos e Práticas
- 4 Ferramentas



CONTEXTO: DEVOPS E DEVSECOPS

- 1 | Cultura
- 2 | Controle
- 3 | Automação
- 4 | Visibilidade



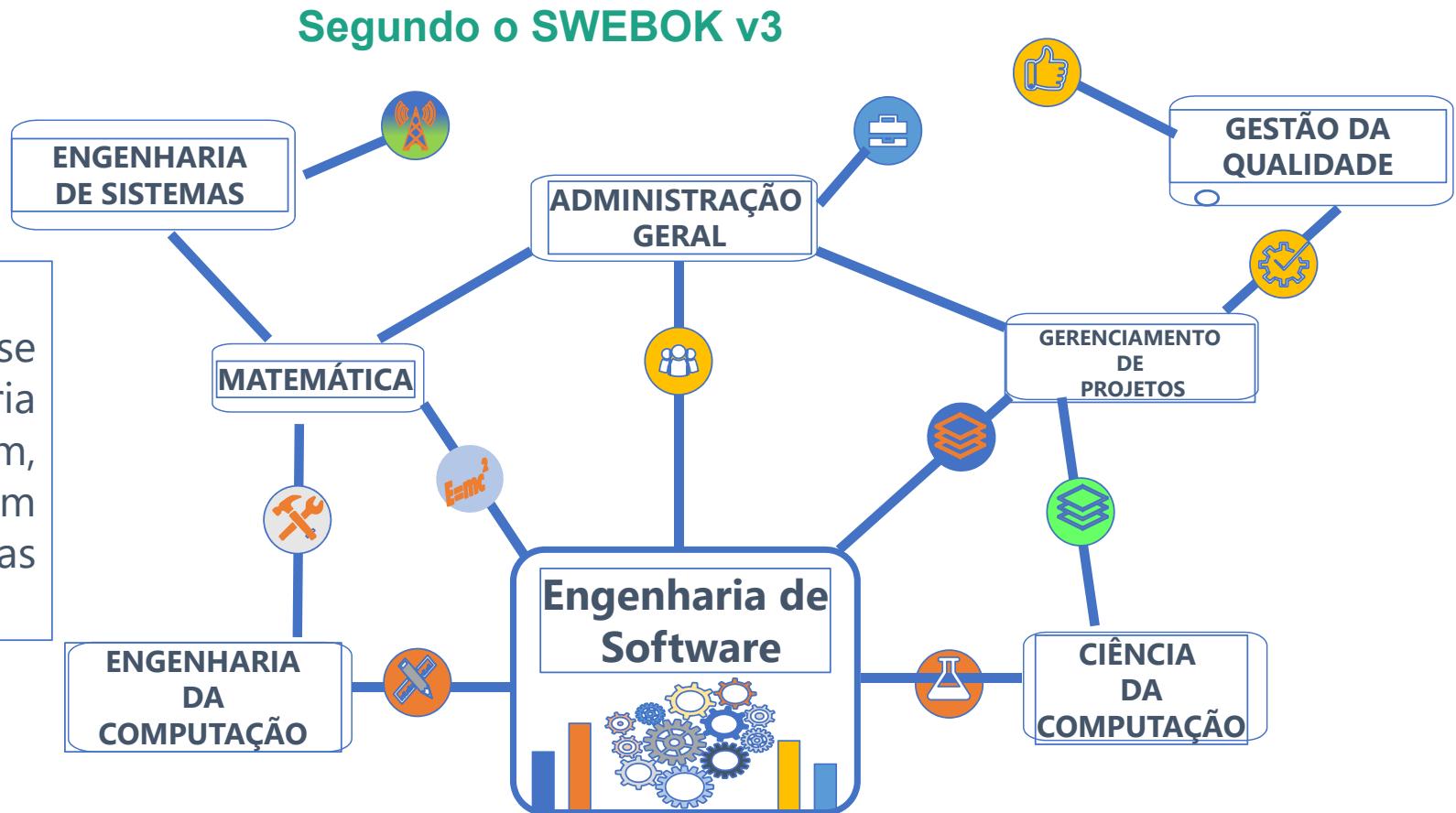
1 – Definições: Knowledge Area (KA) SWEBOK

Áreas de Conhecimento da Engenharia de Software

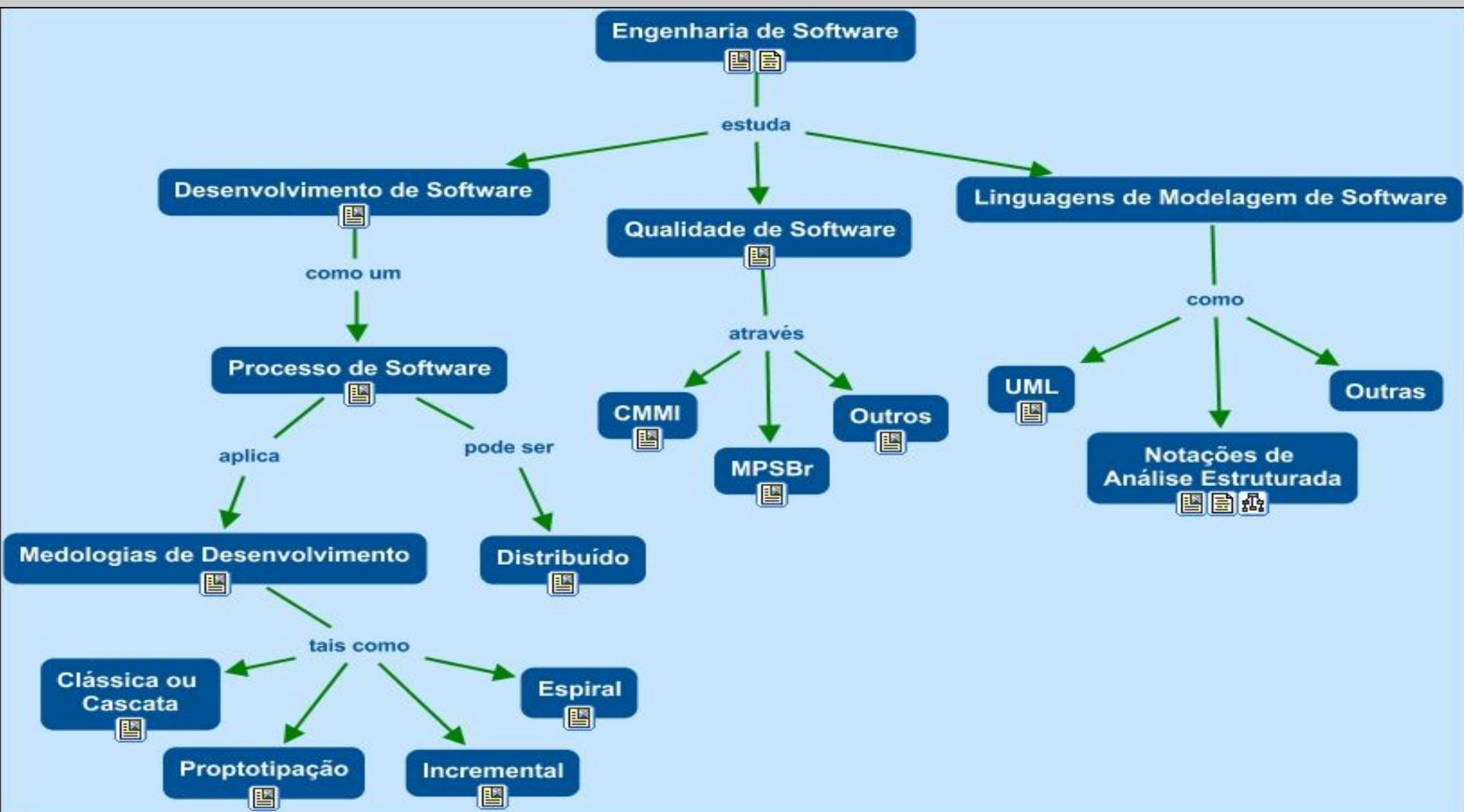


1 – Definições: Disciplinas Relacionadas

É importante identificar as disciplinas que se cruzam com a Engenharia de Software. Para este fim, o SWEBOK V3 também reconhece sete disciplinas relacionadas ao lado:



Para o SWEBOK Guide V3, os editores da SWEBOK receberam e responderam aos comentários de aproximadamente 150 revisores em 33 países.



2 – Arquiteturas:



Negócios



Estrutura e comportamento de um sistema de negócios (não necessariamente relacionado a computadores). Abrange objetivos de negócios, funções ou recursos de negócios, funções e processos de negócios, etc. As funções de negócios e os processos de negócios geralmente são mapeados para os aplicativos e dados de que precisam.



Dados



As estruturas de dados usadas por uma empresa e / ou seus aplicativos. Descrições de dados no armazenamento e dados em movimento. Descrições de armazenamentos de dados, grupos de dados e ítems de dados. Mapeamentos desses artefatos de dados para qualidades de dados, aplicativos, locais etc.



Aplicações



Estrutura e comportamento de aplicativos usados em um negócio, focados em como eles interagem entre si e com os usuários. Focado nos dados consumidos e produzidos por aplicativos e não em sua estrutura interna. No gerenciamento de portfólio de aplicativos, os aplicativos geralmente são mapeados para funções de negócios e para tecnologias de plataforma de aplicativos.



Tecnológica



Estrutura e comportamento da infraestrutura de TI. Abrange os nós de cliente e servidor da configuração de hardware, os aplicativos de infraestrutura que são executados neles, os serviços de infraestrutura que eles oferecem aos aplicativos, os protocolos e as redes que conectam aplicativos e nós.

2 - Arquitetura x Engenharia

SOFTWARE

PREOCUPA-SE

COM A DEFINIÇÃO DOS COMPONENTES DE SOFTWARE, SUAS PROPRIEDADES EXTERNAS, E SEUS RELACIONAMENTOS COM OUTROS SOFTWARES. SE REFERE TAMBÉM À DOCUMENTAÇÃO DA ARQUITETURA DE SOFTWARE DO SISTEMA, QUE FACILITA: A COMUNICAÇÃO ENTRE OS STAKEHOLDERS, REGISTRA AS DECISÕES INICIAIS ACERCA DO PROJETO DE ALTO NÍVEL, E PERMITE O REUSO DO PROJETO DOS COMPONENTES E PADRÕES ENTRE PROJETOS.

ENVOLVE-SE

COM AS TECNOLOGIAS E A MODELAGEM DO SOFTWARE. O ARQUITETO AJUDAR A EVITAR O DÉBITO TÉCNICO, AS QUEDAS DE PERFORMANCE, FALTA DE ESCALABILIDADE, QUE PODE SER CAUSADO PELOS DESENVOLVEDORES. ARQUITETOS TEM UM BOM CONHECIMENTO DE SOLUÇÕES EM ALTO NÍVEL, CONHECEM DESIGN PATTERNS, CONCEITOS COMO SOLID, DRY, YAGNI E TENTAM APLICÁ-LOS ONDE FOR CABÍVEL E POSSÍVEL.

PREOCUPA-SE

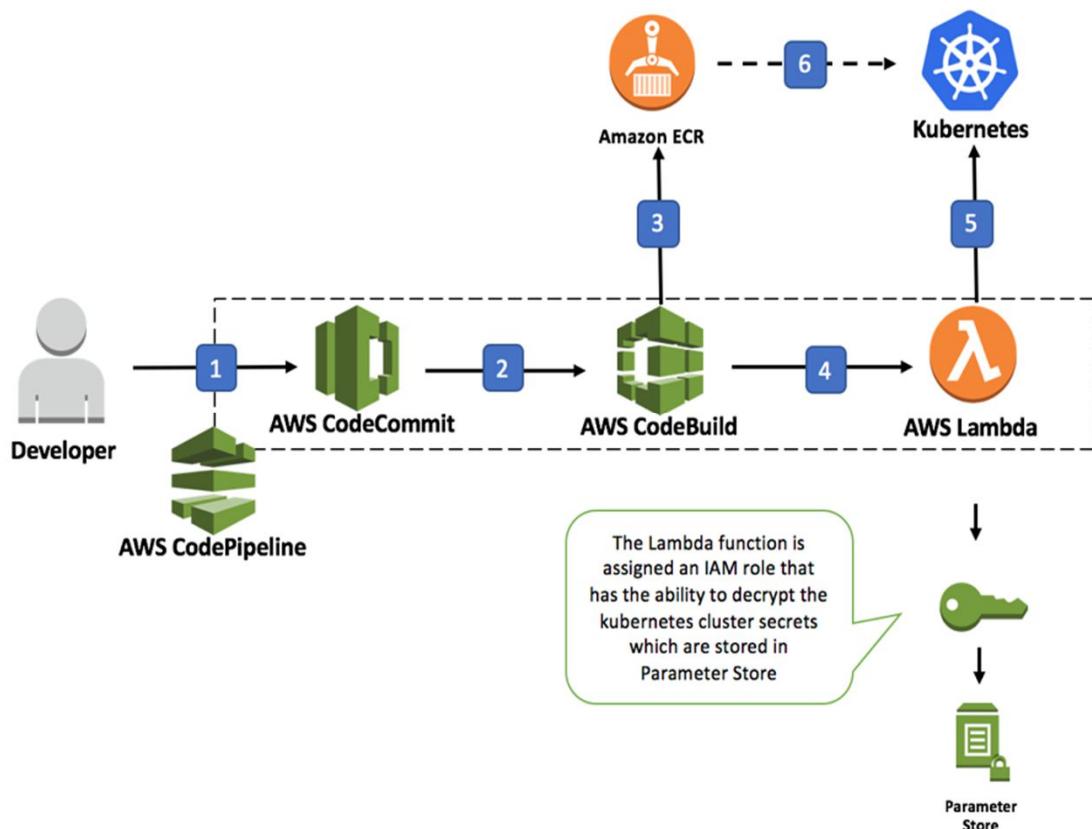
COM A ESPECIFICAÇÃO, DESENVOLVIMENTO E MANUTENÇÃO DE SISTEMAS DE SOFTWARE, COM APLICAÇÃO DE TECNOLOGIAS E PRÁTICAS DE GERÊNCIA DE PROJETOS E OUTRAS DISCIPLINAS, VISANDO ORGANIZAÇÃO, PRODUTIVIDADE E QUALIDADE

ENVOLVE-SE

COM AS TÉCNICAS DE DESENVOLVIMENTO ÁGIL, LIDERANÇA DE EQUIPES. A FUNÇÃO DE UM ENGENHEIRO DE SOFTWARE É MANTER A EQUIPE, COM SEU MELHOR ÍNDICE DE PRODUÇÃO POSSÍVEL, SANANDO PROBLEMAS DO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE.



2 - Arquitetura de Software X Arquitetura DevOps Diferenças

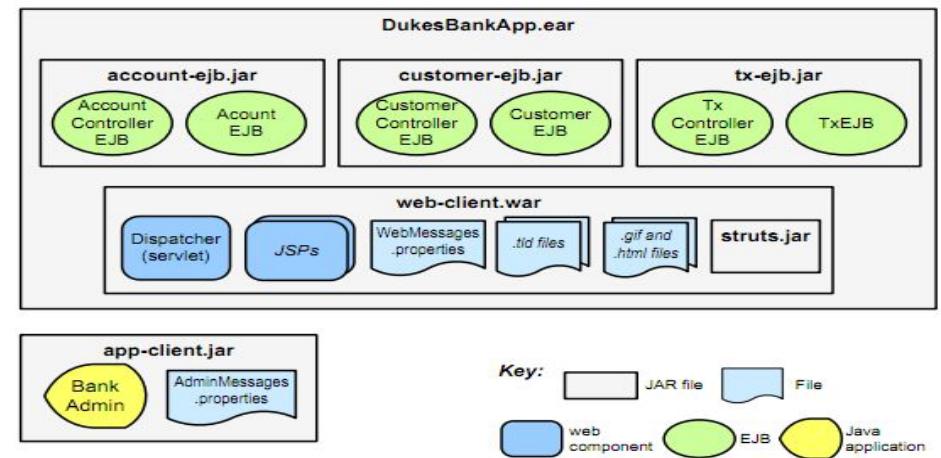
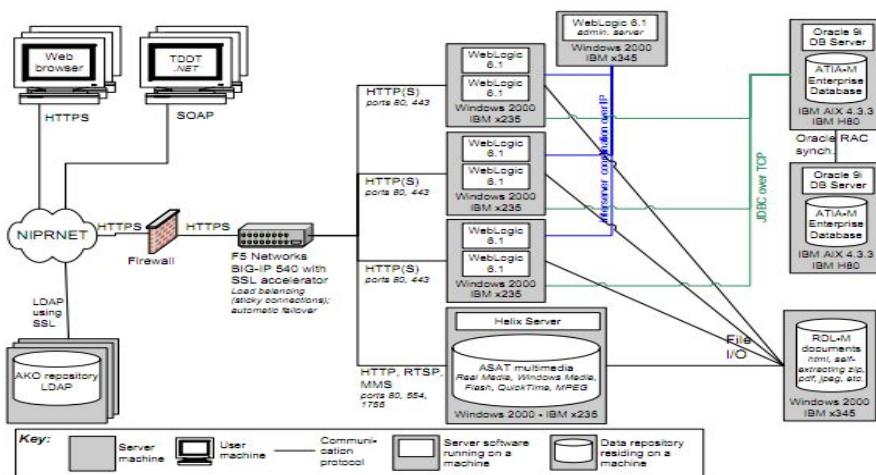


“Arquitetura de Software consiste na definição dos componentes de software, suas propriedades externas, e seus relacionamentos com outros softwares. O termo também se refere à documentação da arquitetura de software do sistema. A documentação da arquitetura do software facilita: a comunicação entre os stakeholders, registra as decisões iniciais acerca do projeto de alto-nível, e permite o reuso do projeto dos componentes e padrões entre projetos.” [Wikipedia](#)

2 - ARQUITETURA DE SOFTWARE X ARQUITETURA DEVOPS

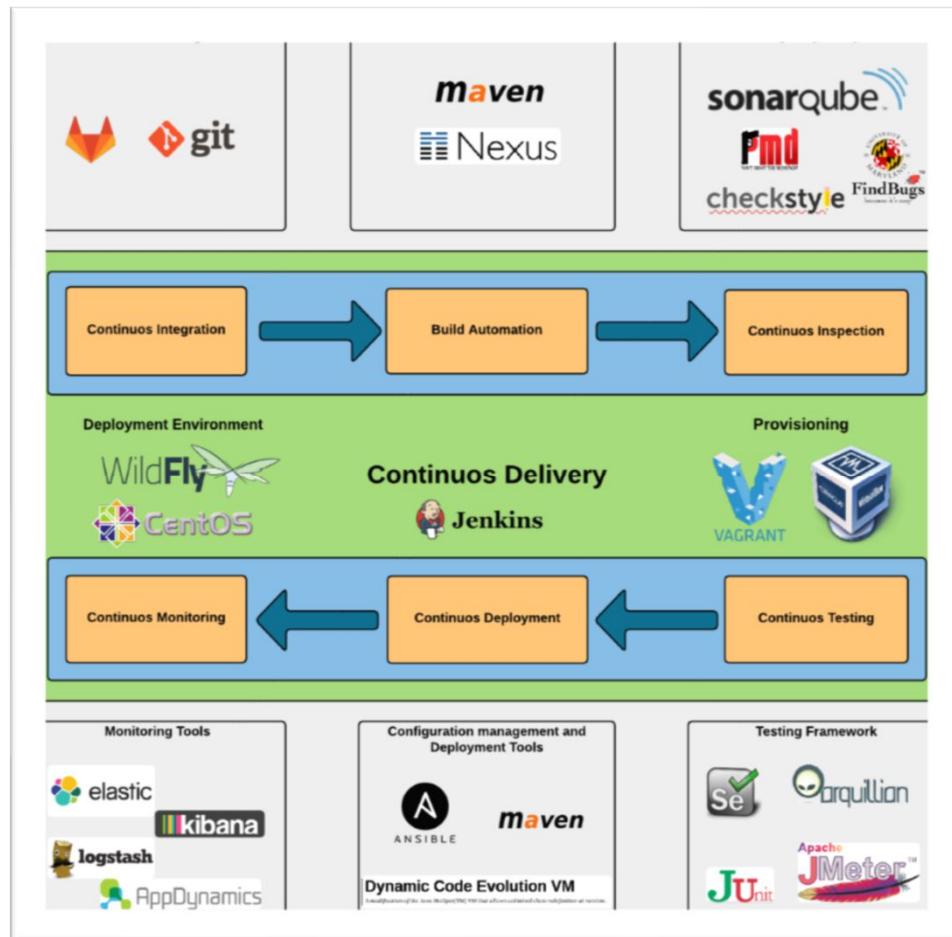
A disciplina de arquitetura de software é centrada na ideia da redução da complexidade através da abstração e separação de interesses.

A arquitetura de software serve como modelo para o sistema e o projeto que o desenvolve, definindo as atribuições de trabalho que devem ser executadas pelas equipes de projeto e implementação. A arquitetura é a principal portadora de qualidades do sistema, como desempenho, modificabilidade e segurança, e nenhuma delas pode ser obtida sem uma visão arquitetônica unificadora. A arquitetura é um artefato para a análise inicial para garantir que uma abordagem de projeto produza um sistema aceitável. Ao construir uma arquitetura eficiente, é possível identificar riscos de projeto e mitigá-los no início do processo de desenvolvimento.



2 - Arquitetura de Software X Arquitetura DevOps

Diferenças



Arquitetura DevOps engloba ganhos de produtividade através do aprimoramento e automação de fluxos (workflows) de trabalho, por meio de técnicas práticas como Continuous Integration e Continuous Delivery.

Desenvolvedores – e arquitetos – têm de assumir a responsabilidade para a construção de seus testes e implantação de vias automatizadas de testes.

Em DevOps, o monitoramento se torna um fator muito mais importante do que arquitetura e design, a fim de atender aos requisitos de operações.

Mapa Mental

<https://coggle.it/diagram/WjLmDrbZhwABkhlu/t/arquitetura-de-sistemas-com-devops>

2 - Arquitetura de software



3 – Conceitos: Soa - Arquitetura Orientada a Serviços

Before SOA

Closed - Monolithic - Brittle

Application Dependent Business Functions



After SOA

Shared services - Collaborative - Interoperable - Integrated

Composite Applications



Reusable Business Services



Data Repository



3 – ConceitoS: Soa X Micro Serviços: Diferenças

1. O SOA RA deve ser uma solução genérica que seja neutra do fornecedor.
2. O SOA RA é baseado em um modelo de conformidade com padrões.
3. O SOA RA deve ser capaz de ser instanciado para produzir:
 - a) Arquiteturas intermediárias do setor
 - b. Arquiteturas de soluções concretas
4. A Arquitetura de Referência SOA deve promover e facilitar o alinhamento de TI para negócios.
5. O RA SOA deve abordar múltiplas perspectivas das partes interessadas.

Arquitetura Orientada a Serviços

O termo "Service-Oriented Architecture" (SOA) ou Arquitetura Orientada a Serviços expressa um conceito no qual aplicativos ou rotinas são disponibilizadas como serviços em uma rede de computadores (Internet ou Intranets) de forma independente e se comunicando através de padrões abertos. A maior parte das implementações de SOA se utilizam de Web services (SOAP , REST e WSDL). Entretanto, uma implementação de SOA pode se utilizar de qualquer tecnologia padronizada baseada em web.

Monolithic vs. SOA vs. Microservices



3 – Conceitos: Soa X Micro Serviços: Diferenças

Arquitetura Orientada a Micro Serviços

Quando você toma o domínio de sua empresa e o divide em vários aplicativos, você está criando vários serviços que fazem parte da mesma Arquitetura Orientada a Serviços. Quando esses serviços são extremamente leves, minúsculos, “faça uma coisa e faça bem”... aí você tem micros serviços.

Então, no final do dia, você tem a ideia de dividir sua arquitetura em vários serviços (SOA) e, no mundo da tecnologia de hoje, a tendência é manter esses serviços extremamente finos (micros serviços).

Building Blocks for Containerized Microservices

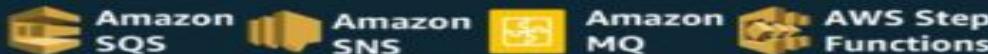
Compute



Storage & Database



Application Integration



Networking & API Proxy



Developer Tools



Logging & Monitoring



© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

3 – Conceitos: Soa X Micro Serviços: Diferenças

5 ORIENTAÇÃO À EVENTOS
www.nginx.com/blog/event-driven-data-management-microservices

6 ESTRATÉGIA DE IMPLANTAÇÃO
www.nginx.com/blog/deploying-microservices

BOM TRABALHO! QUE TAL ANALISAR OS TRADE-OFFS E OS PRÉ-REQUISITOS DA ADOÇÃO DE MICROSERVIÇOS?

7 TRADE-OFFS DA ABORDAGEM DE MICROSERVIÇOS
martinfowler.com/articles/microservice-trade-offs.html

8 PRÉ-REQUISITOS DA ABORDAGEM DE MICROSERVIÇOS
martinfowler.com/bikiki/MicroservicePrerequisites.html

4 DESCOPERTA DE SERVIÇOS
www.nginx.com/blog/service-discovery-in-a-microservices-architecture



Arquitetura Orientada a Micro Serviços

3 COMUNICAÇÃO INTER-PROCESSO
www.nginx.com/blog/building-microservices-inter-process-communication

2 USO DE API GATEWAY
www.nginx.com/blog/building-microservices-using-an-api-gateway

1 INTRODUÇÃO À ARQUITETURA DE MICROSERVIÇOS
www.nginx.com/blog/introduction-to-microservices

INICIE A JORNADA APRENDENDO OS CONCEITOS BÁSICOS LIGADOS À ARQUITETURA DE MICROSERVIÇOS!

14 EXEMPLOS DE CÓDIGO
eventuate.io/examples.html

13 PILHA DE TECNOLOGIAS UTILIZADAS PARA MICROSERVIÇOS
netflix.github.io

12 CONSTRUÇÃO PARA FALHAS
martinfowler.com/bikiki/CircuitBreaker.html

9 REFATORAÇÃO DE MONOLITOS
www.nginx.com/blog/refactoring-a-monolith-into-microservices

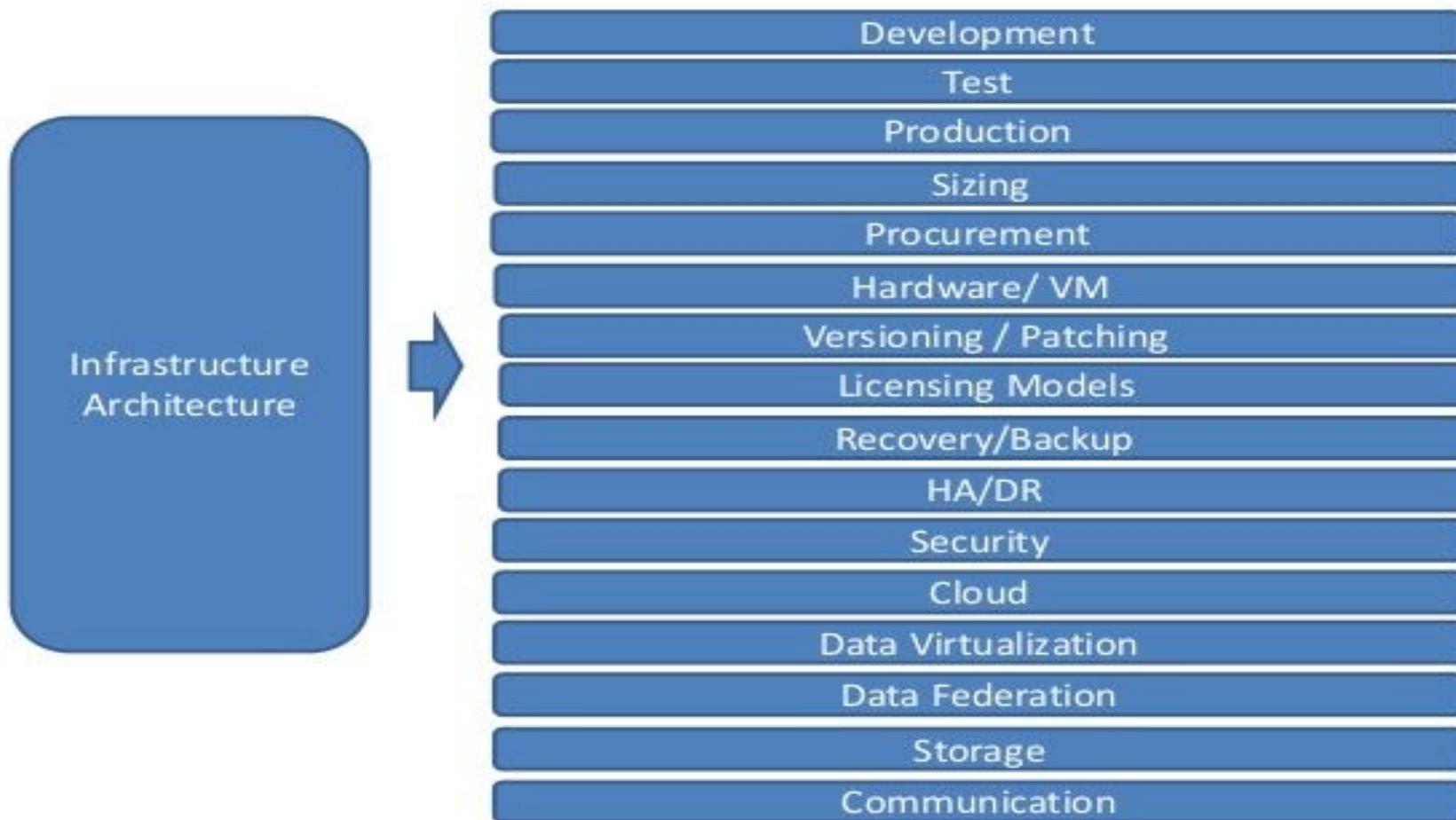
10 CRIAÇÃO DE MICROSERVIÇOS COM DDD BOUNDED CONTEXT
martinfowler.com/bikiki/BoundedContext.html

11 PADRÕES PARA A ABORDAGEM DE MICROSERVIÇOS
microservices.io/patterns

PARABÉNS! AGORA VOCÊ ESTÁ PREPARADO PARA INICIAR SUA JORDA RUMO À ARQUITETURA DE MICROSERVIÇOS!

ATENÇÃO! CASO ESTEJA CONFECIONANDO UM SOFTWAREE A PARTIR DO ZERO, SEM CONSIDERAR UM LEGADO, PULE PARA O PASSO 10

Infrastructure Architecture



3 – conceitos: Serverless X Containers X Virtualização (VM):Diferenças

Serverless

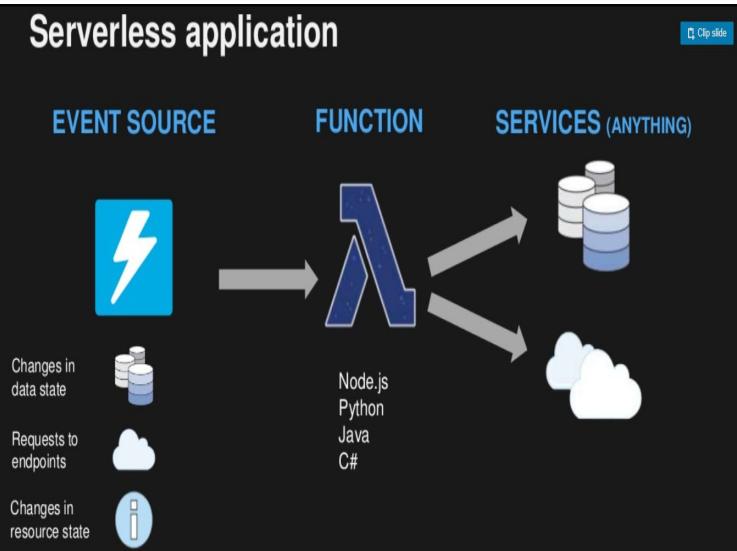
Não pague pela inatividade
Nenhum servidor para provisionar ou gerenciar
Escalas com uso
Disponibilidade e Tolerância a falhas incorporadas
Orientado a eventos

Containers

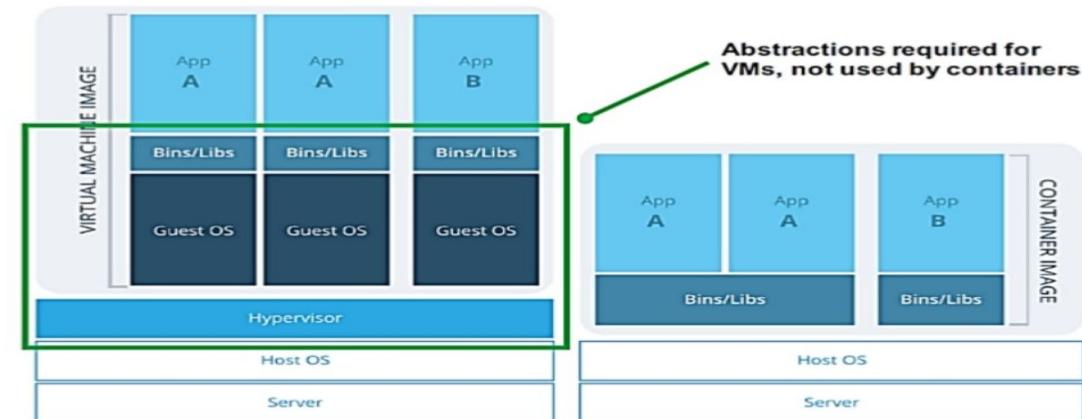
Abstrações de pacotes de aplicativos comuns
Ecossistema de distribuição de aplicativos
Orquestração de múltiplas aplicações
Fluxo fácil através do desenvolvimento, teste, preparação e produção
Filosofias de gerenciamento centradas no aplicativo
Emergindo como método preferido para aplicativos web escaláveis

Virtual Machine -VM

Abstração da infraestrutura física subjacente, incluindo um número de recursos X definidos por software
Isolamento seguro de cargas de trabalho
Tecnologia conhecida e entendida, desenvolvida a duas décadas
Particionamento e gerenciamento de recursos conhecido
Base de instalação existente macia e força de trabalho qualificada



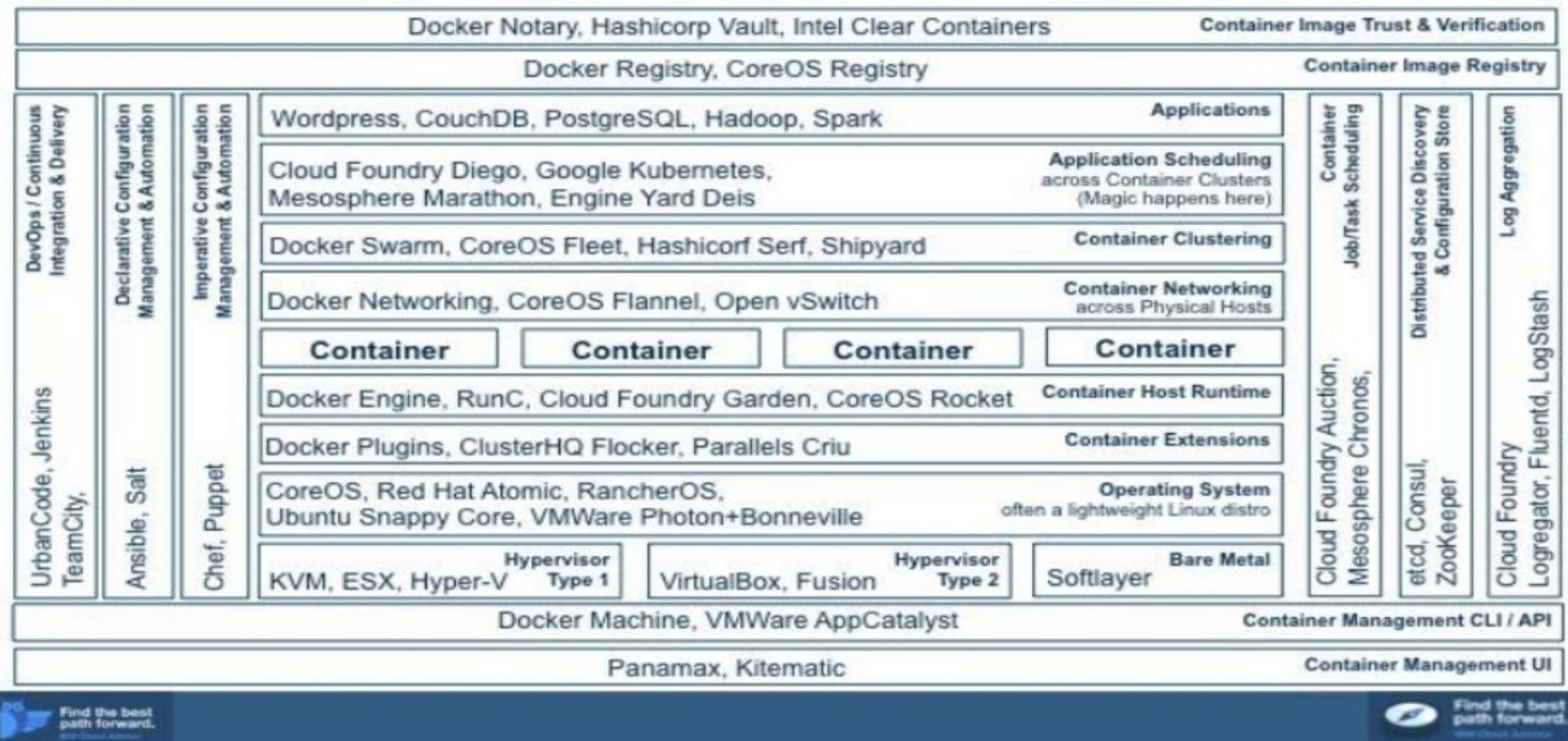
Containers vs. Virtual Machines



3 – conceitos: Serveless X Containers X Virtualização (VM):Diferenças



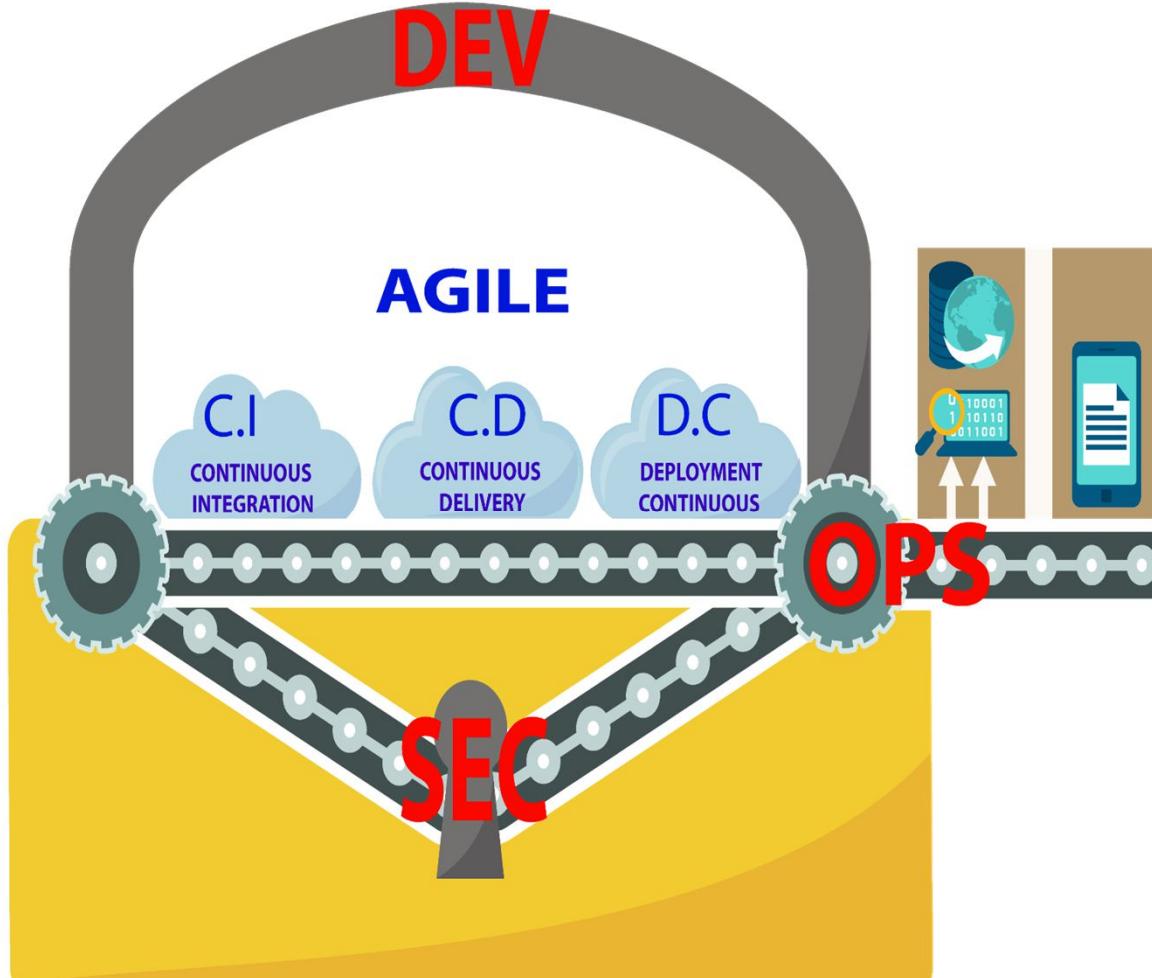
Open Source Container Ecosystem



Source: [CloudsWithCarl](#)

3 – Conceitos:

O que é DevSecOps ?

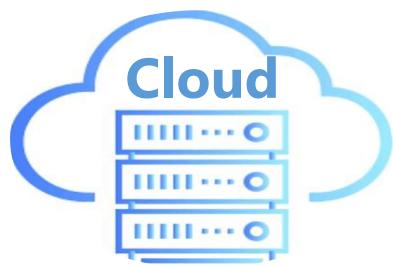


DevSecOps é um termo criado para descrever um conjunto de práticas para integração entre os times de Desenvolvimento de Software, Segurança e Operações e a adoção de processos automatizados para produção rápida e segura de aplicações e serviços

3 – Conceitos:

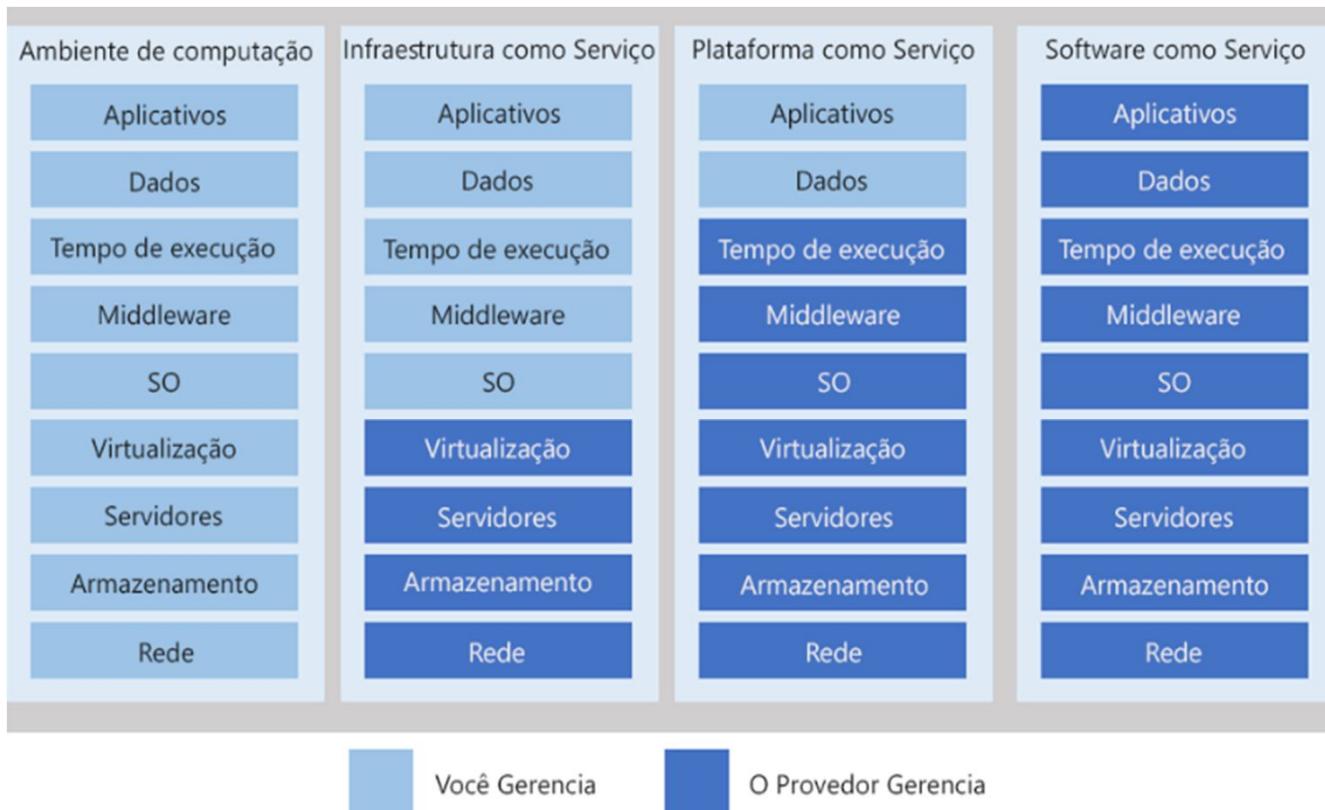
O que é Cloud ?

Cloud é um modelo computacional que permite acesso a recursos compartilhados e configuráveis de infraestrutura, plataforma e soluções de tecnologia (servidores, armazenamento, bancos de dados, rede, software, analytics, etc.) de forma flexível, on demand e escalável.



3 – Conceitos:

Cloud Modelos de Contratação



IaaS – Infrastructure as a Service



PaaS – Platform as a Service

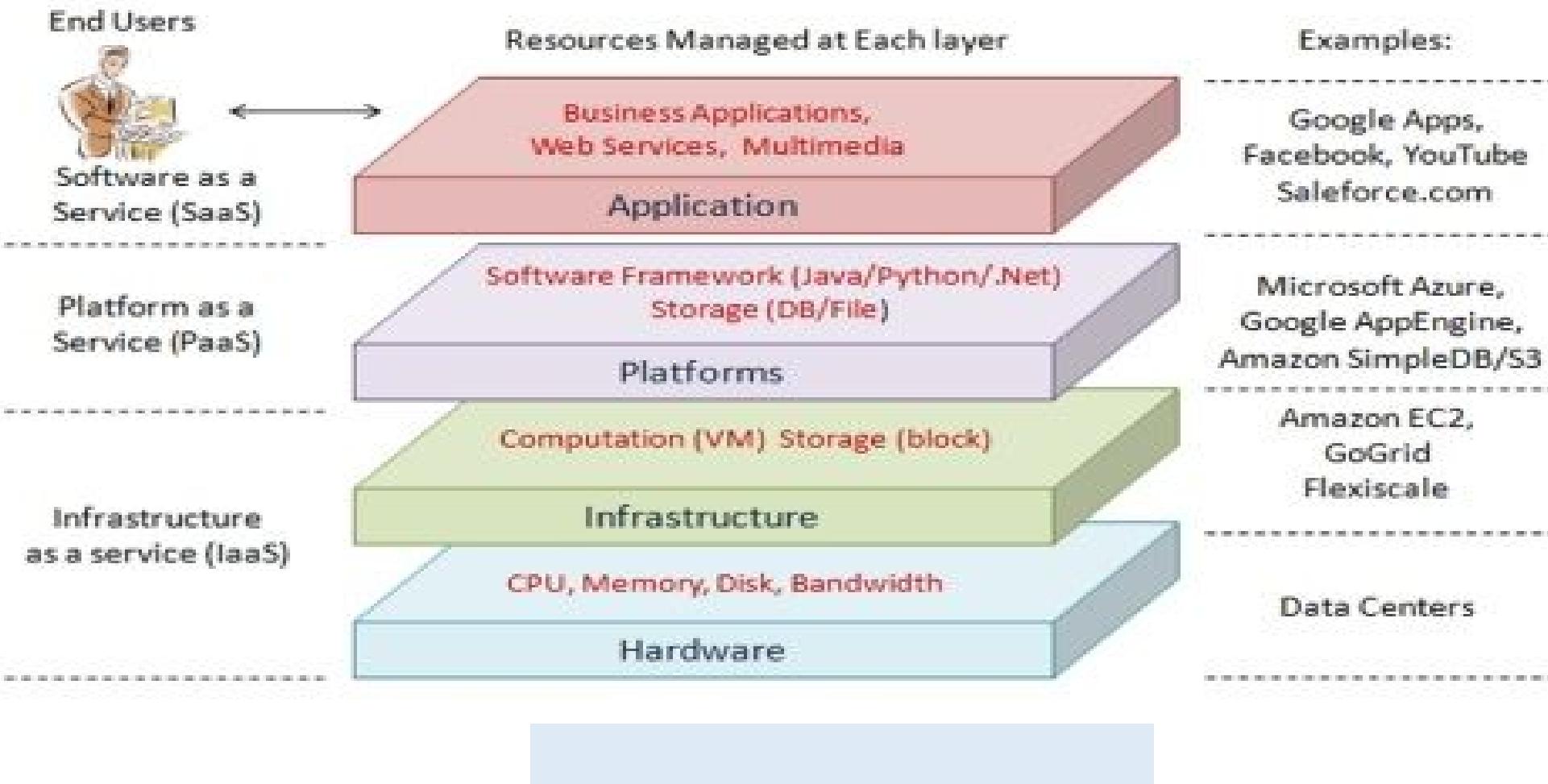


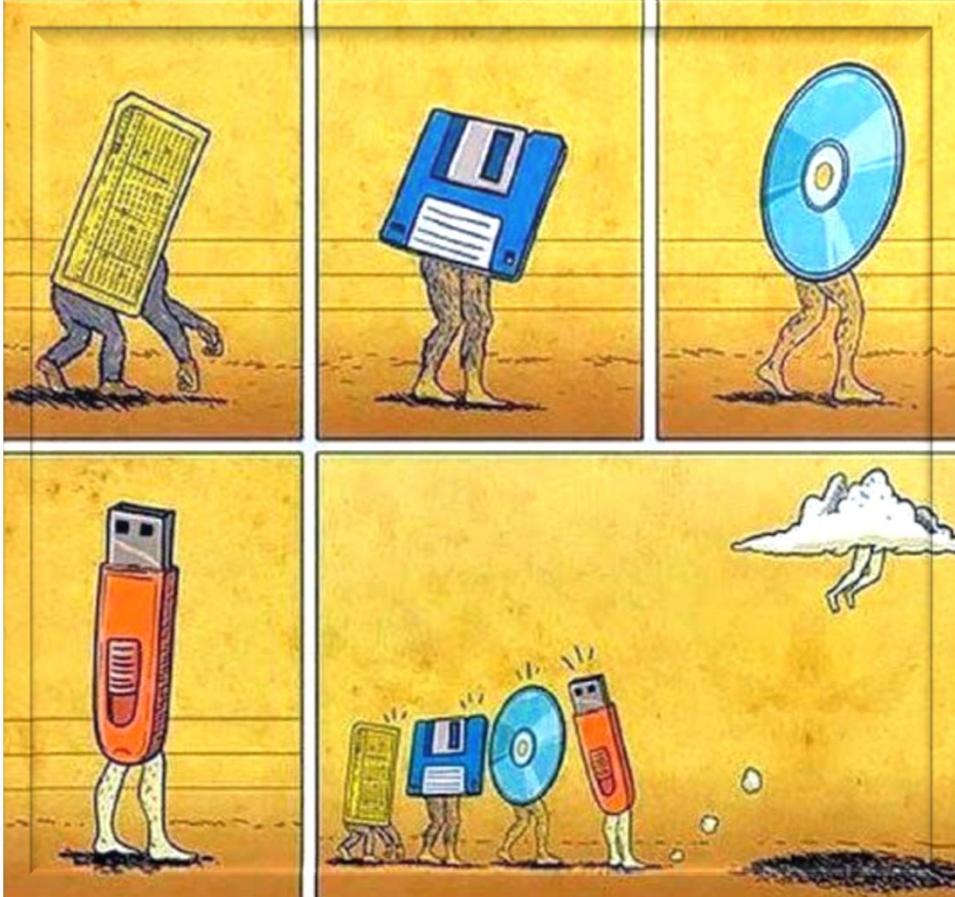
SaaS – Software as a Service



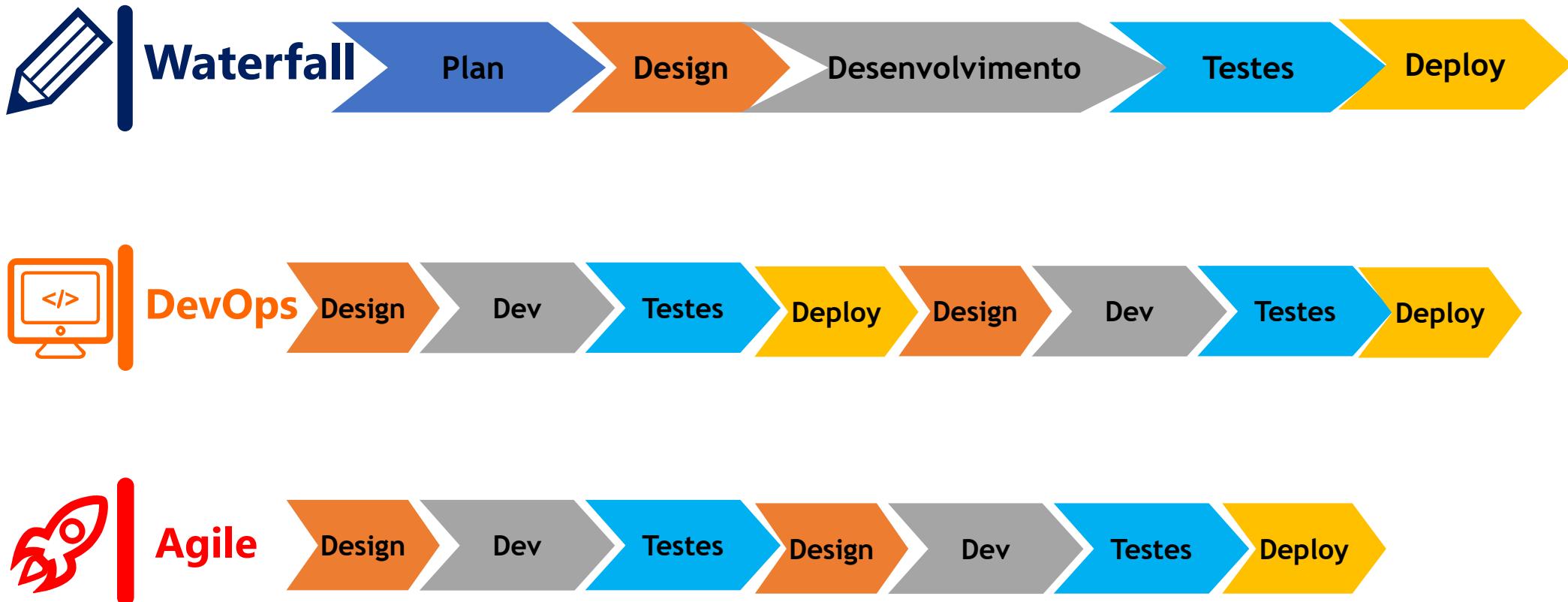
3 – Conceitos:

Cloud Modelos de Contratação

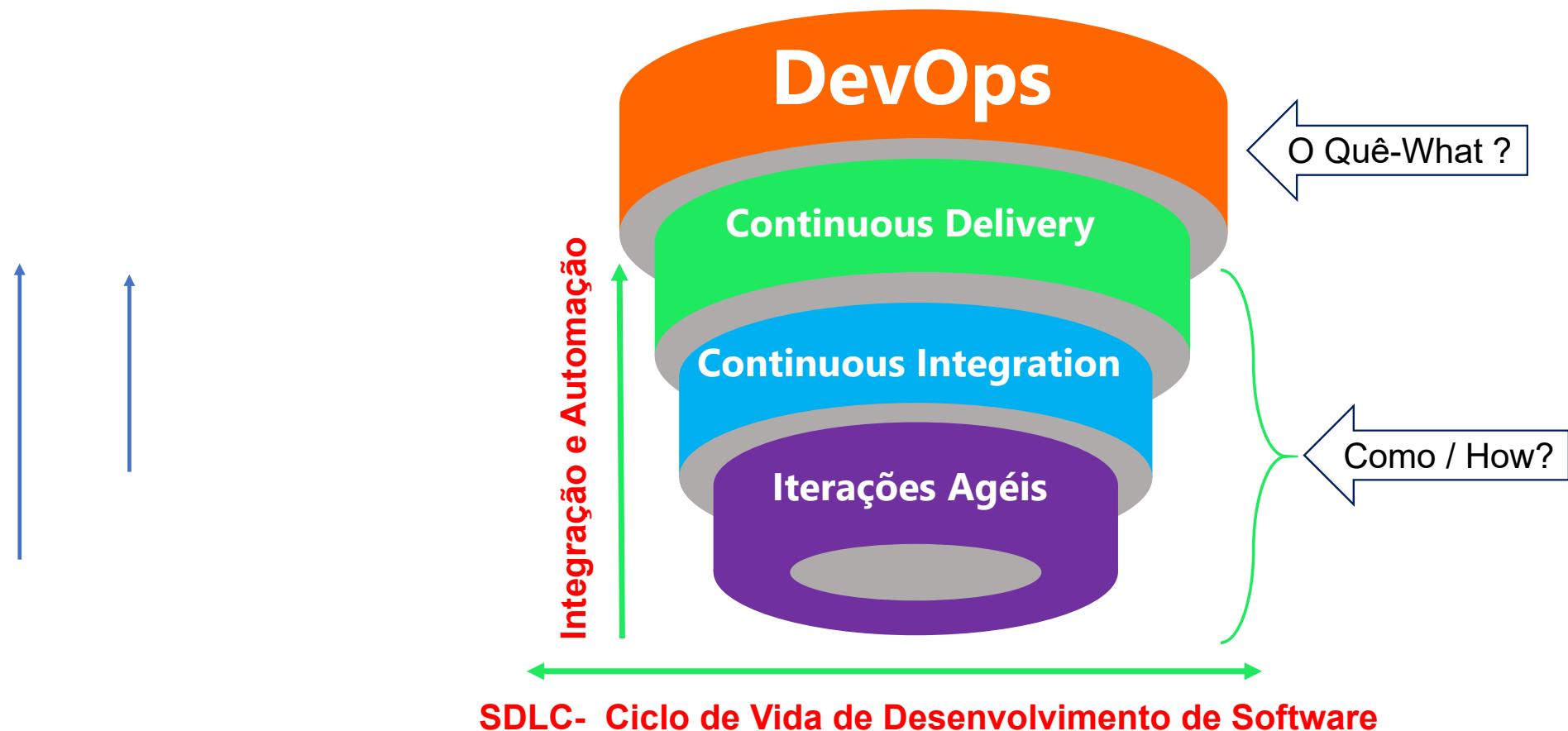




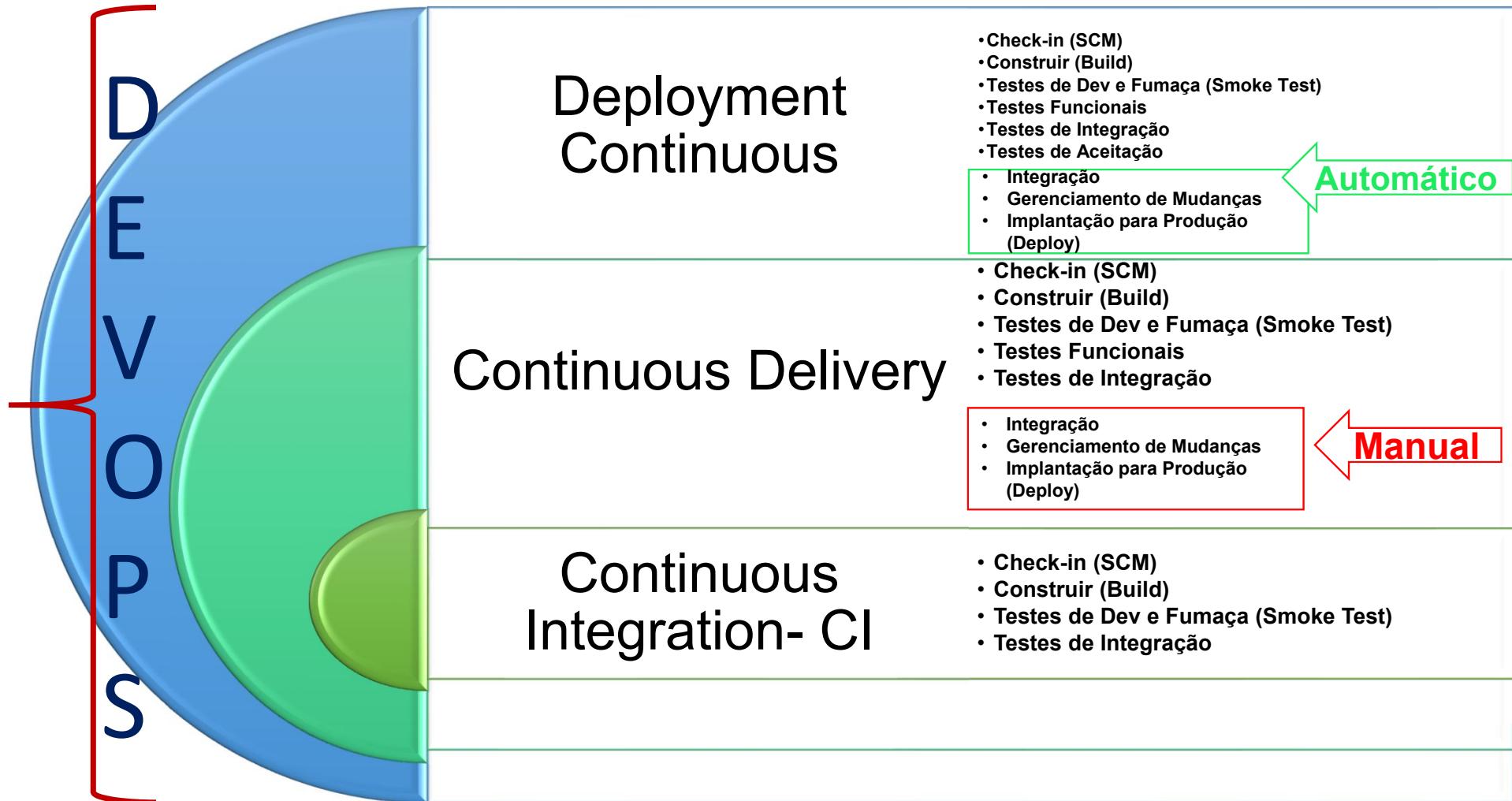
4 - DevOps Versus Outros Modelos:



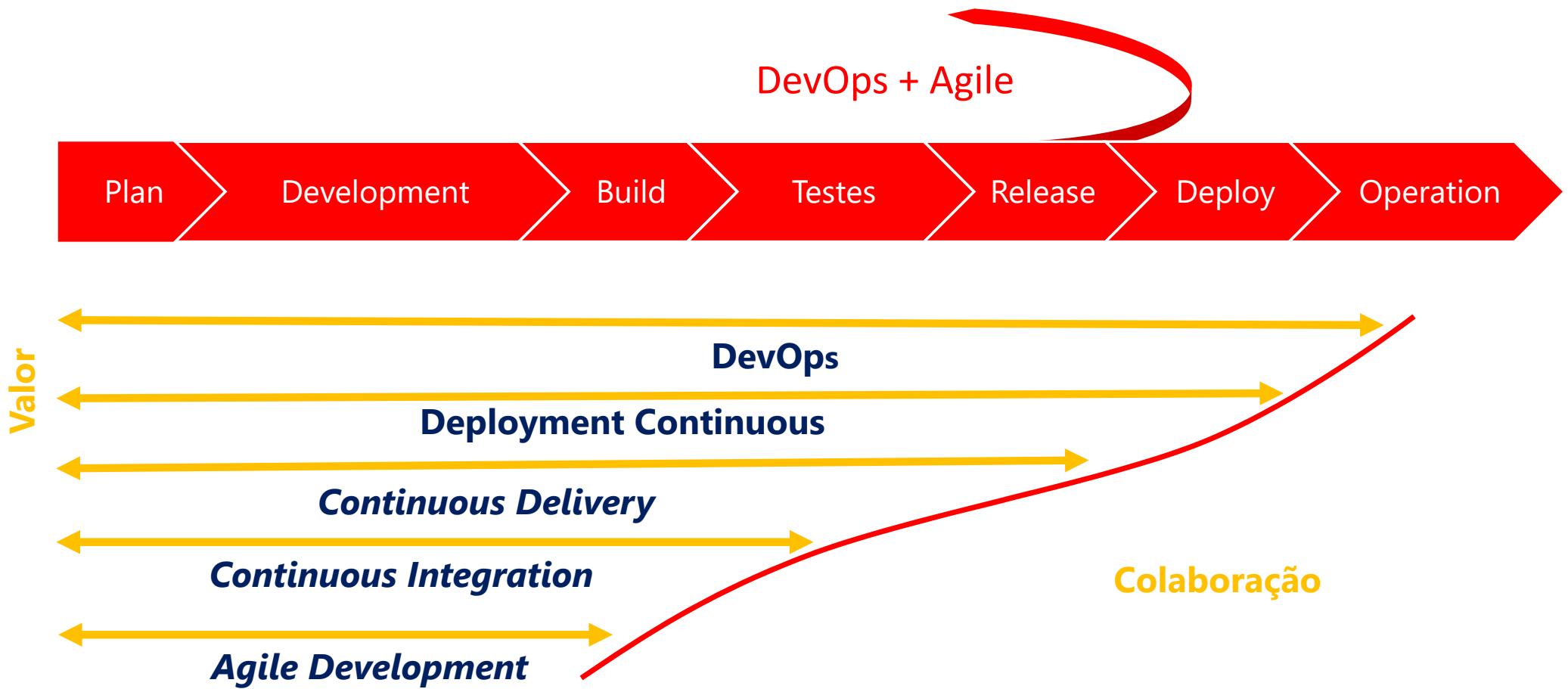
5 - DevOps vs CI & CD:



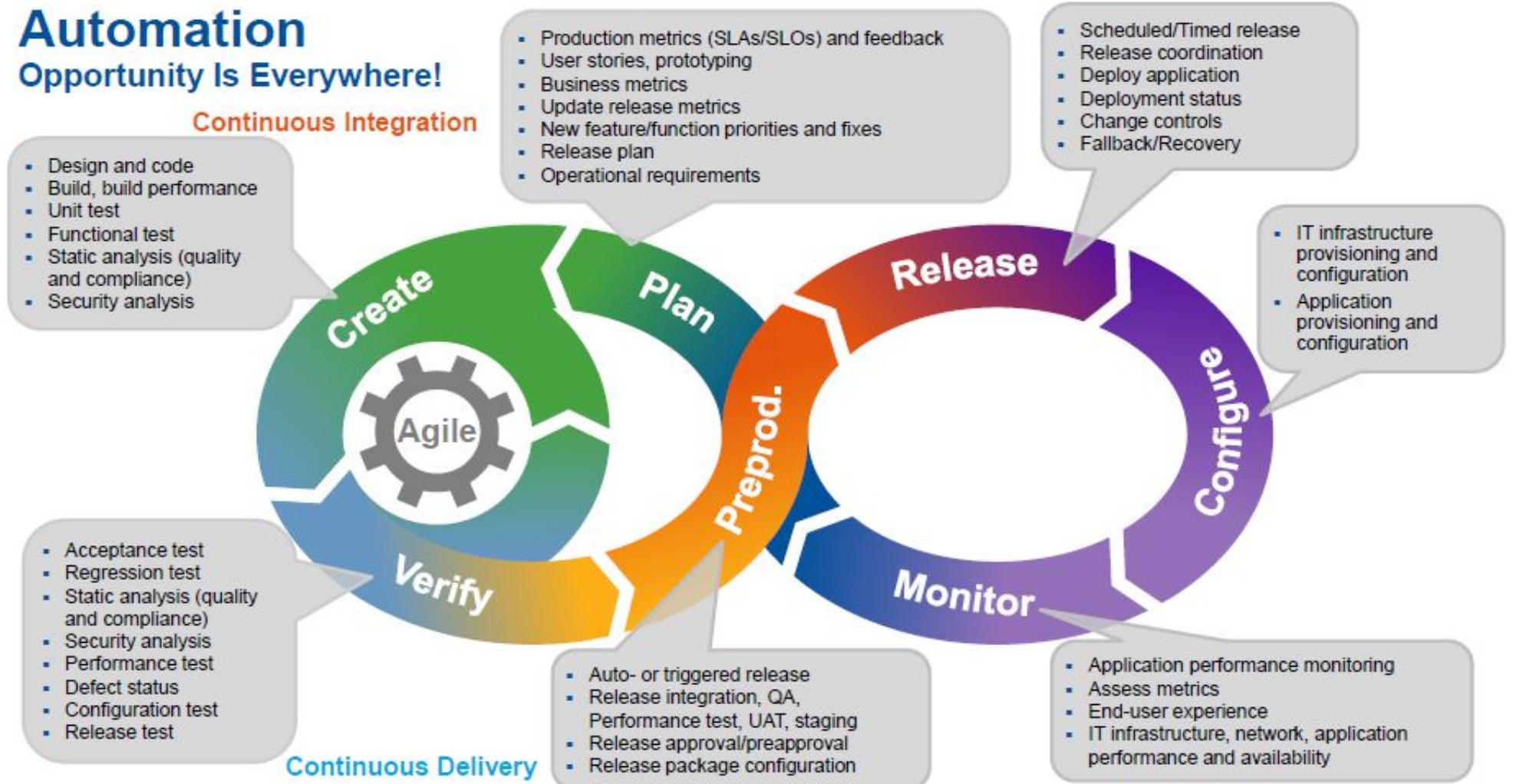
5 - Diferença Entre DevOps X CI & CD:



5 - Diferença Entre DevOps X CI X CD X DC:



5 - Diferença Entre DevOps X CI & CD:



6 - FASES de implantação: Descrição dos Papéis e Processos para Implantação DevOps

Papéis:

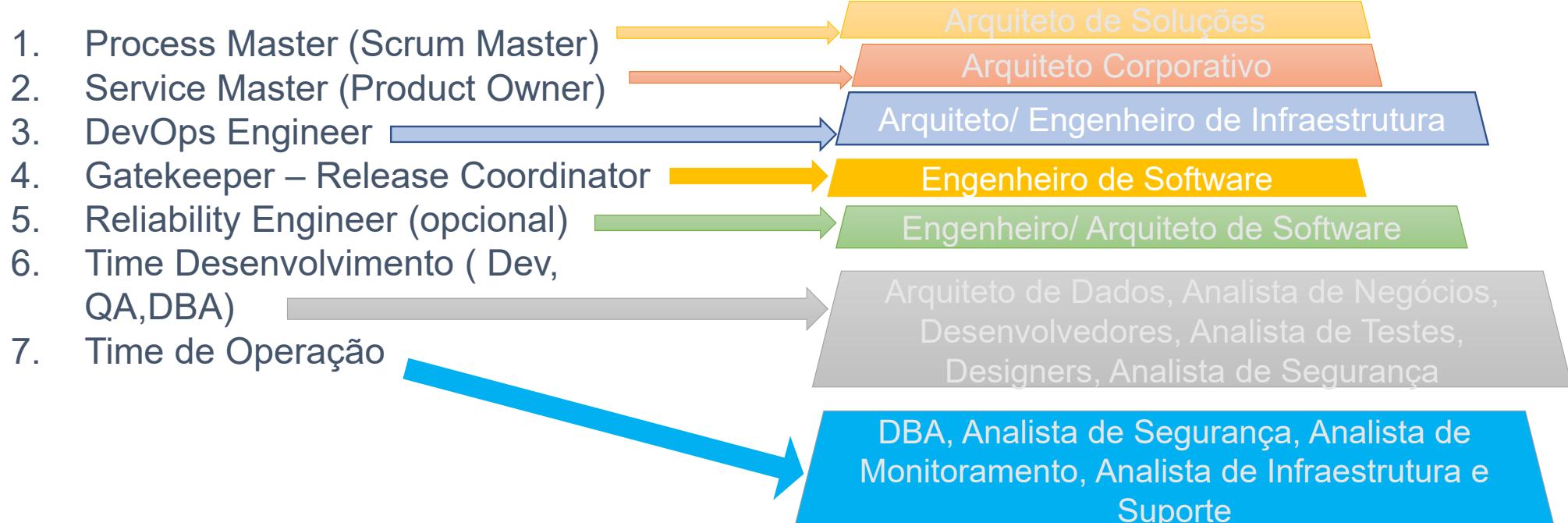
1. **Process Master (Scrum Master)**
2. **Service Master (Product Owner)**
3. **DevOps Engineer**
4. **Gatekeeper – Release Coordinator**
5. **Reliability Engineer (opcional)**
6. **Time/Squad Desenvolvimento (AN, Dev, QA,DBA)**
7. **Time/Squad Operação**

Processos:

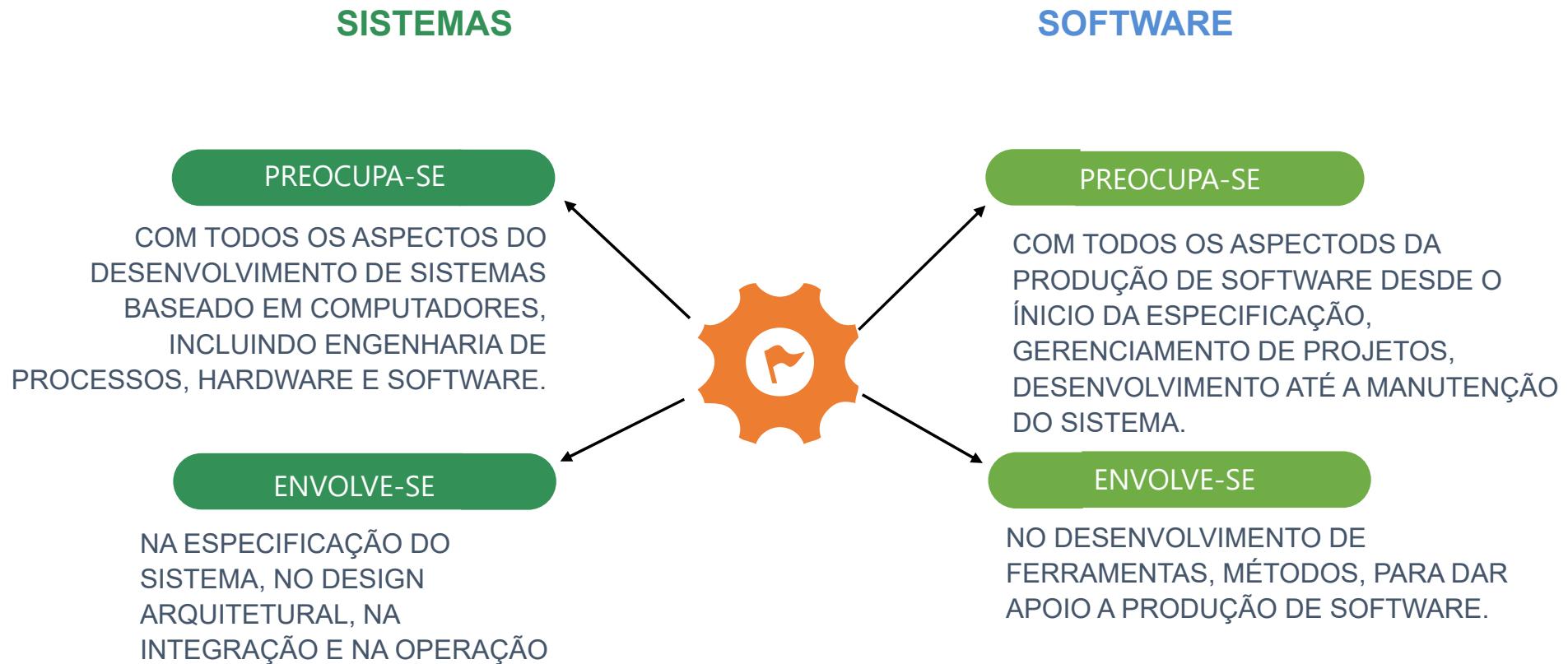
1. **Planejamento & Estratégia de Negocio**
2. **Marketing & Vendas**
3. **Administração**
4. **Planejamento de Projeto**
5. **Requerimentos & Desenho**
6. **Desenvolvimento**
7. **Implantação (Deployment)**
8. **Operação**
9. **Manutenção**
10. **Serviço do Cliente**
11. **Encerrar Ciclo de Vida**

6 - Sugestão de Composição de Papéis para Implantação DeSECvOps :

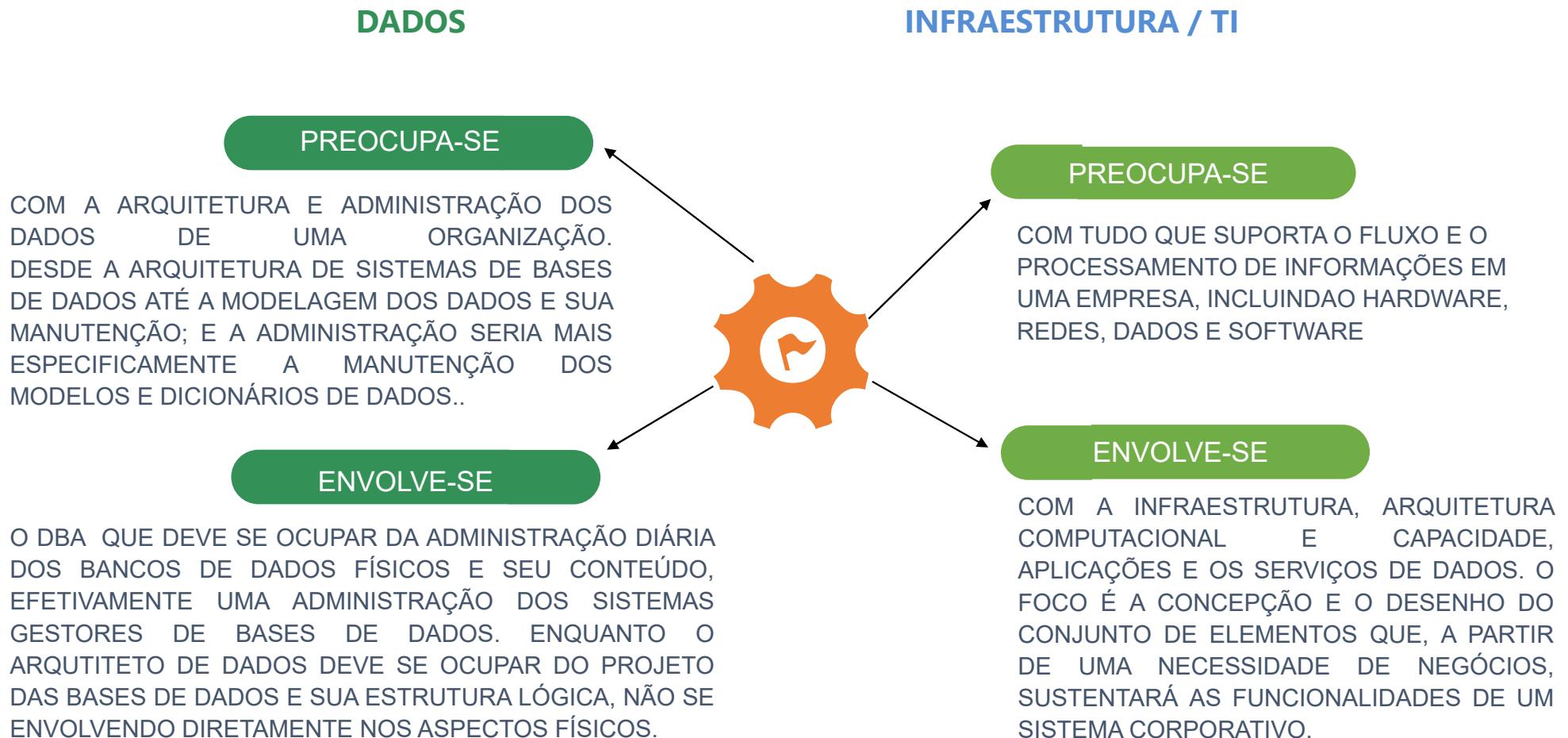
Papéis:



6 - Papéis Engenharia:



6 - Papéis Arquitetura:



6 - Papéis Arquitetura:

SOLUÇÕES

PREOCUPA-SE

COM GRUPOS DE APLICAÇÕES ESPECÍFICOS, OU MAIS PRECISAMENTE: TRATA-SE DE UMA ESPECIALIZAÇÃO VOLTADA PARA A ARQUITETURA DE SISTEMAS QUE SOLUCIONAM NECESSIDADES ESPECÍFICAS EM DETERMINADOS CENÁRIOS DE NEGÓCIOS.

ENVOLVE-SE

COM CONHECIMENTOS SOBRE INFRAESTRUTURA DE REDES, SISTEMAS, SEGURANÇA, ETC. SEJAM TRADICIONAIS OU EM NUVEM. DEVE CONHECER TAMBÉM DE NEGÓCIOS E TER VIÉS FORTE DE RELACIONAMENTO COM PESSOAS UMA VEZ QUE, NA MAIORIA DAS EMPRESAS ONDE ATUA, ACABARÁ APOIANDO A ÁREA COMERCIAL NA DEFESA DE SOLUÇÕES TECNOLÓGICAS JUNTO A POTENCIAIS CLIENTES.

CORPORATIVA

PREOCUPA-SE

COM A VISÃO DE NEGÓCIOS DO PONTO DE VISTA ESTRATÉGICO, ATUARÁ NO FUNCIONAMENTO E ORGANIZAÇÃO DA EMPRESA, REALIZANDO MUDANÇAS TÉCNICAS E PROCESSUAIS ONDE FOR NECESSÁRIO. É IMPRESCINDÍVEL TER DOMÍNIO DE FRAMEWORKS – PADRÕES ESTABELECIDOS DE ARQUITETURA CORPORATIVA – PARA QUE SEJA POSSÍVEL ALINHAR AS AÇÕES ESTRATÉGICAS DA EMPRESA DE MANEIRA MAIS AMPLA, INCLUINDO TI.

ENVOLVE-SE

COM A INFRAESTRUTURA, ARQUITETURA COMPUTACIONAL E CAPACIDADE, APLICAÇÕES E OS SERVIÇOS DE DADOS. O FOCO É A CONCEPÇÃO E O DESENHO DO CONJUNTO DE ELEMENTOS QUE, A PARTIR DE UMA NECESSIDADE DE NEGÓCIOS, SUSTENTARÁ AS FUNCIONALIDADES DE UM SISTEMA CORPORATIVO.

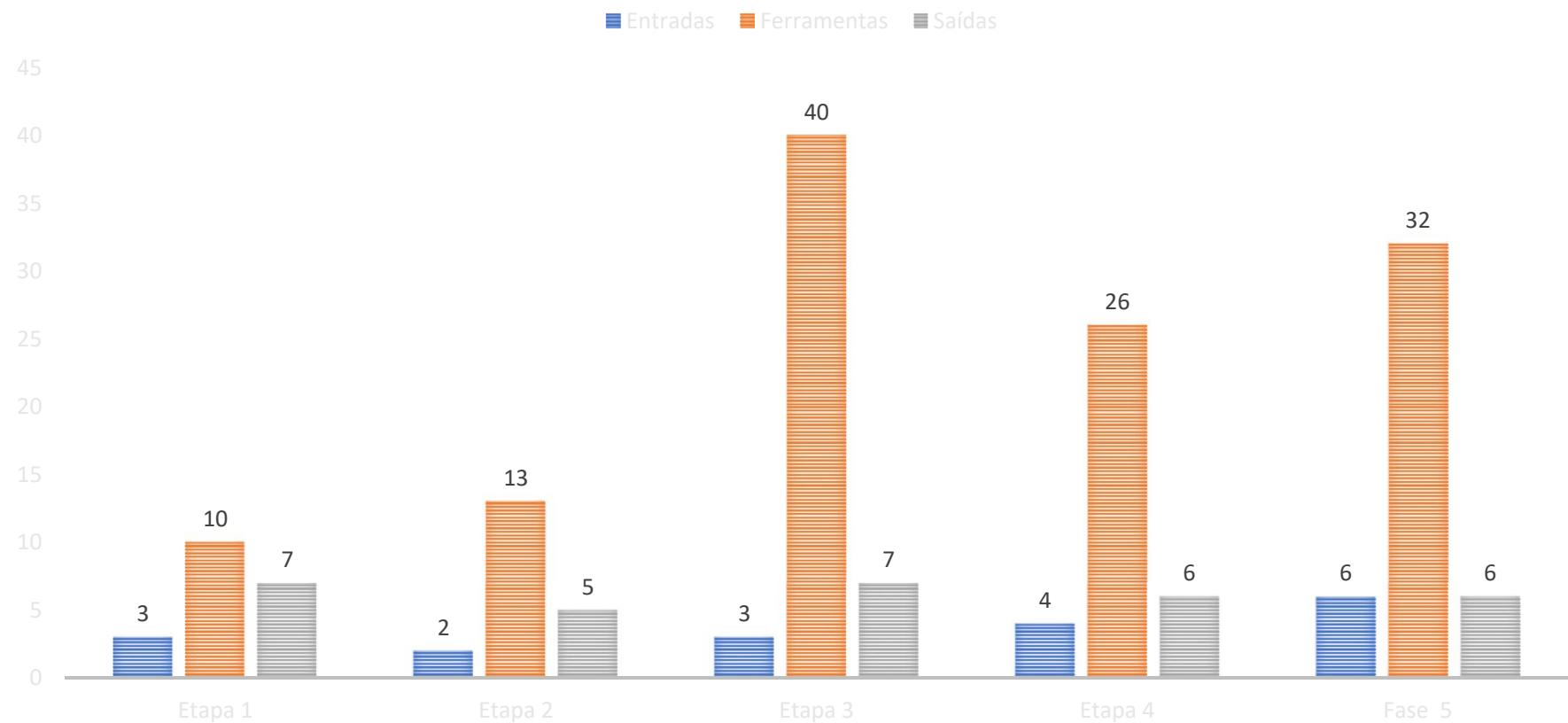


6 - Descrição das Etapas de Implantação DevOps:

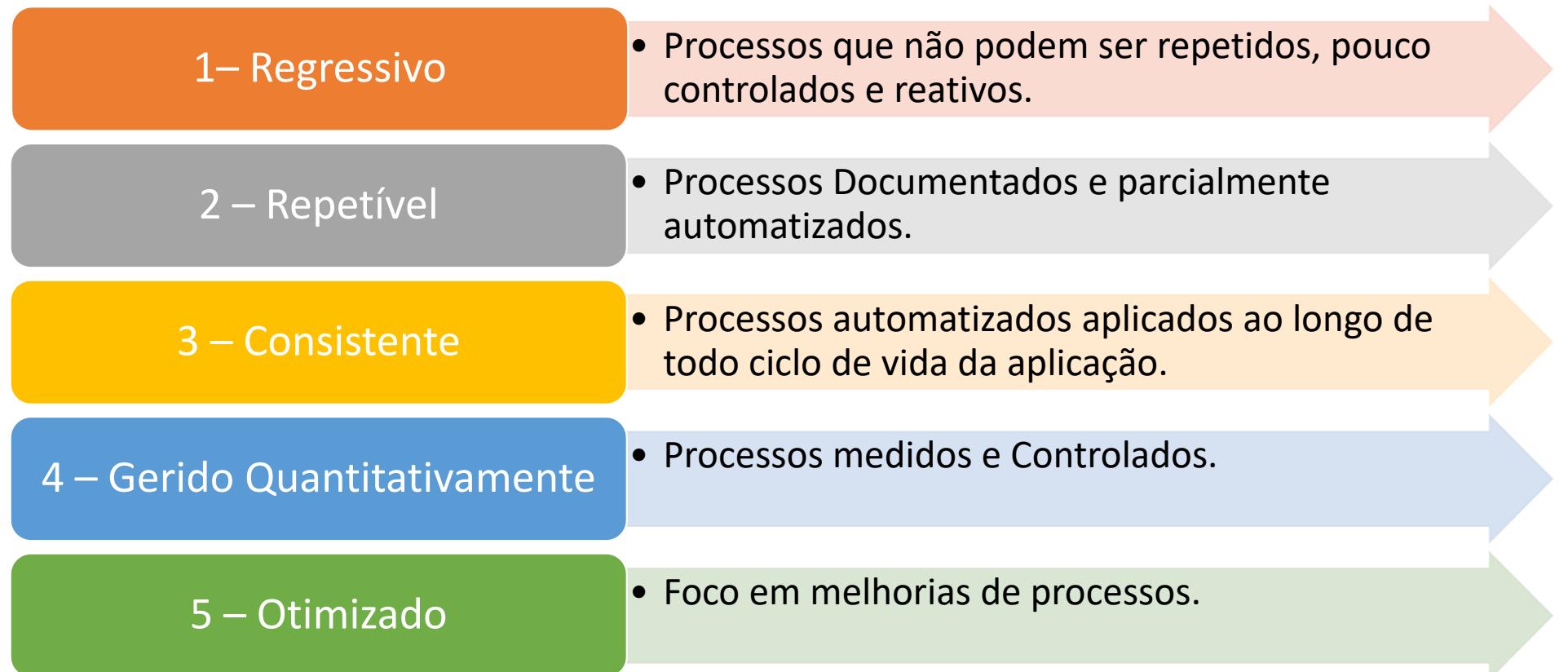
1. Gerenciar o Ciclo de Vida da Aplicação
2. Termo de Abertura do Projeto e Controle Visual de Atividades
3. Desenho da Arquitetura e Infraestrutura de TI
4. Requisitos e Acordos de Nível de Serviço
5. Implementação da Estratégia de Testes

Etapas	Papéis	Processos
Gerenciar ...	2	4
Termo de Abertura...	7	4
Desenho..	5	7
Requisitos...	7	2
Implementação..	7	7

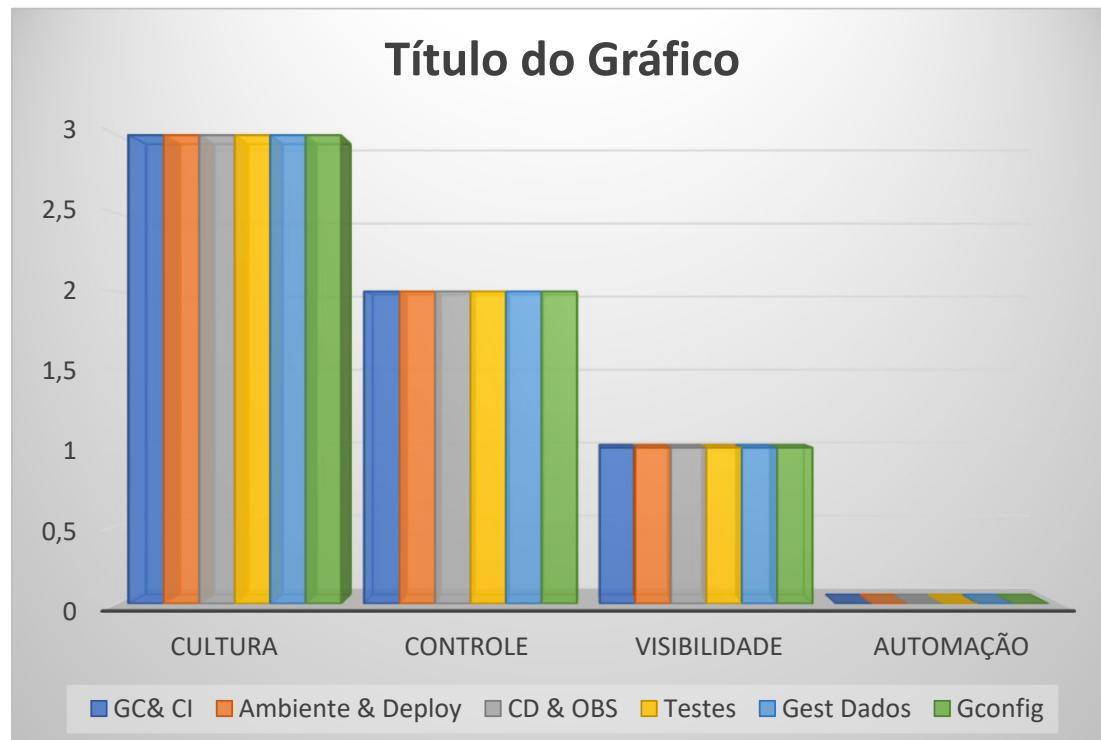
6 - Etapas de Implantação DevOps



7 - Níveis de Maturidade DevOps:



Dimensões	Gerência de Configuração	Gestão de Compilação e Integração Contínua	Gestão de Entrega de Versão e Observância	Testes	Gestão de Dados	Gestão de Ambientes e Deployment
Cultura	3	2	1	3	2	1
Controle	3	2	1	3	2	1
Visibilidade	3	2	1	3	2	1
Automação	3	2	1	3	2	1
Score Parcial	12	8	4	12	8	4



Legenda	Score Mínimo	Score Máximo
Expert	63	120
Avançado	49	62
Intermediário	25	48
Iniciante	0	24

1 – Cultura: MINDSET - PESSOAS

DEVSECOPS

NEGÓCIOS QUE PERMITEM RECEPVIDADE

REDUZIR O TEMPO DE ESPERA PARA MUDANÇAS

MONITORAMENTO CONTÍNUO

CONTINUOUS DELIVERY

INFRAESTRUTURA AUTOMATIZADA

INTEGRAÇÃO CONTÍNUA

TESTES AUTOMATIZADOS

CONTROLE DE VERSÕES

Práticas

ALTA CONFIANÇA

INOVAÇÃO

ORIENTAÇÃO A DESEMPENHO

EMPODERAMENTO DE TIMES

REDUÇÃO DE VARIAÇÃO

ALTA COOPERAÇÃO

Cultura



FLUXO CONTÍNUO & VISIBILIDADE

PRINCÍPIOS LEAN & AGILE

FOCO EM PRODUTOS

FLUXO DO SISTEMA

AMPLIAR FLUXO DE FEEDBACK

EXPERIMENTAÇÃO CONTÍNUA

2 – Controle
3 – Automação
4 -Visibilidade

Engenharia de Software

CAMADAS DE DESENVOLVIMENTO DE SOFTWARE

MONITORIA
SEGURANÇA
DEPLOYMENT
TESTE TOOLS
REPOSITORIO
PLUG-INS
SDK
IDE
SCM
CASE

MANUTENÇÃO
IMPLEMENTAÇÃO
TESTES
DESENVOLVIMENTO
DESENHO
ANÁLISE
REQUISITOS
PROJETO

MÉTODOS

FERRAMENTAS

DEVOPS
CRYSTALL
KANBAN
XP
SCRUM
WATERFALL
RUP
PROCESSOS

CONTROLE DE
QUALIDADE
SGQ
MPS-BR
CMMI
MATURIDADE
TQM
QUALIDADE

4 – Visibilidade: Processos e Métricas





COFFEE BREAK

8 - Visão Geral das Possíveis Ferramentas a serem utilizadas:

1	Fm.	PERIODIC TABLE OF DEVOPS TOOLS (V2)												2	Fm.																				
Gh	Github													Aws	Amazon Web Services																				
3	Os	4	En																																
Gt	Git	Dm	DBmaestro																																
11	Fm.	12	Os																																
Bb	Bitbucket	Lb	Liquibase																																
19	Os	20	En	21	Os	22	Os	23	Os	24	Os	25	Fr	26	Os	27	Fr	28	Os	29	Pd	30	Os	31	Pd	32	Os	33	Os	34	Os	35	Os	36	En
Gl	GitLab	Rg	Redgate	Mv	Gradle	Gr	ANT	At	Fn	FitNesse	Se	Selenium	Ga	Gettling	Dh	Docker Hub	Jn	Jenkins	Ba	Bamboo	Tr	Travis CI	Gd	Deployment Manager	Sf	SmartFrog	Cn	Consul	Bc	Bcfg2	Mo	Mesos	Rs	Rackspace	
37	Os	38	En	39	Os	40	Os	41	Os	42	Fr	43	Os	44	Fr	45	Os	46	Fm	47	Pd	48	Fm	49	Fr	50	Fr	51	Os	52	Os	53	Fr	54	Os
Sv	Subversion	Dt	Datical	Gt	Gulp	Gp	Broccoli	Br	Cu	Cucumber	Cj	Cucumber.js	Qu	Qunit	Npm	npm	Cs	Codeship	Vs	Visual Studio	Cr	CircleCI	Cp	Capistrano	Ju	Juju	Rd	Rundeck	Cf	CFEngine	Ds	Swarm	Op	OpenStack	
Hg	Mercurial	Dp	Delphix	Sb	Make	Mk	CMake	Ck	Jt	JUnit	Jm	JMeter	Tn	TestNG	Ay	Artifactory	Tc	TeamCity	Sh	Shippable	Cc	CruiseControl	Ry	RapidDeploy	Cy	CodeDeploy	Oc	Octopus Deploy	No	CA Nolio	Kb	Kubernetes	Hr	Heroku	
73	En	74	En	75	Os	76	Os	77	Fr	78	Os	79	Fr	80	Os	81	Os	82	Os	83	Fm	84	Pd	85	En	86	En	87	Fr	88	En	89	Os	90	En
Cw	ISPW	Id	Idera	Msb	MSBuild	Rk	Packer	Pk	Mc	Mocha	Km	Karma	Jm	Jasmine	Nx	Nexus	Co	Continuum	Ca	Shipable	So	Continuum CI	Xld	XL Deploy	Eb	ElasticBox	Dp	Deploybot	Ud	UrbanCode Denali	Nm	Nomad	Os	OpenShift	



 Follow @xebialabs

8 - Visão Geral das Possíveis Ferramentas a serem utilizadas:

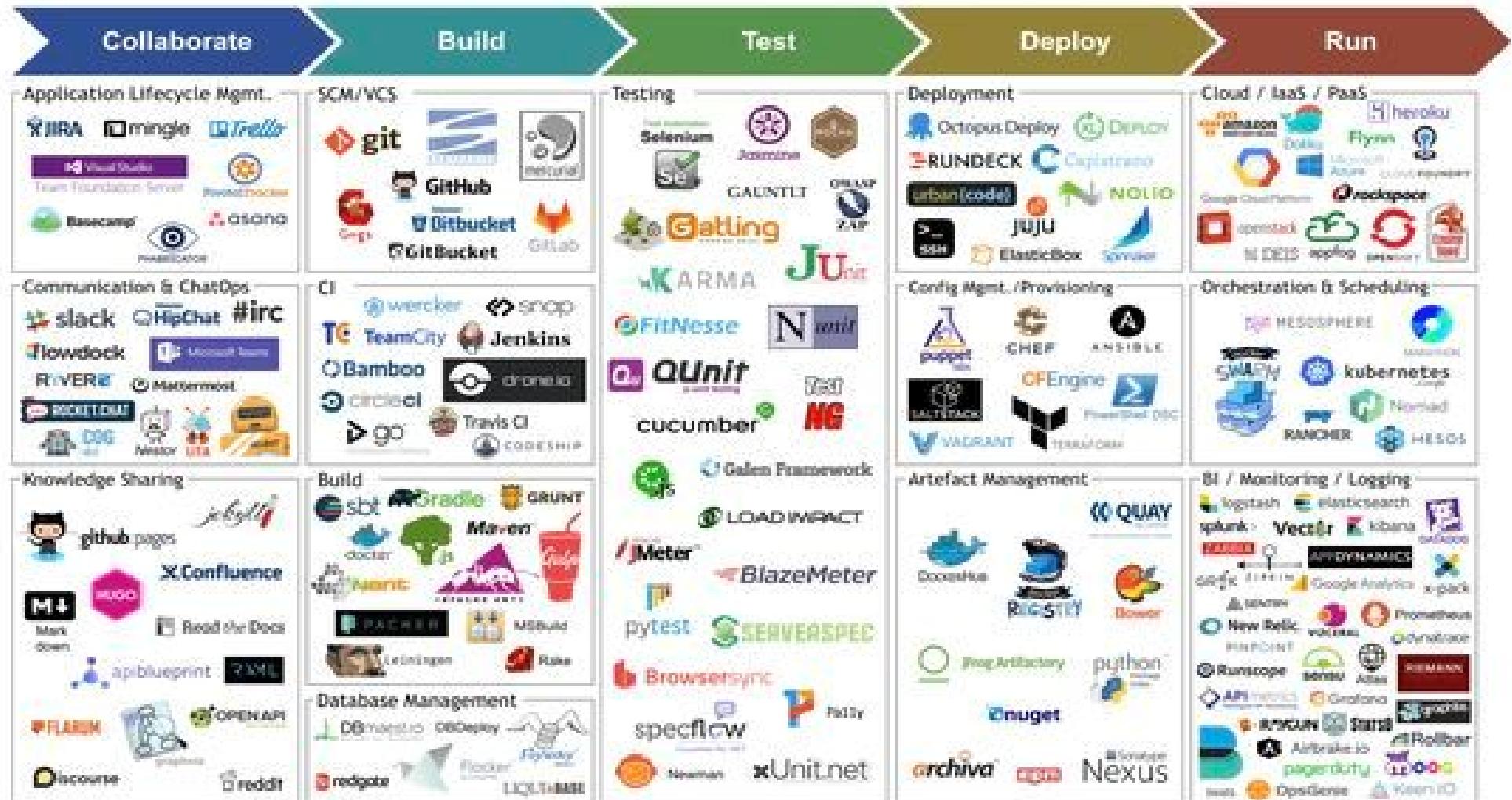
1	Os	PERIODIC TABLE OF DEVOPS TOOLS (V3)														2	En																						
GI																Sp																							
GitLab																Splunk																							
3	Fm	4	En																																				
Gh		Dt																																					
GitHub		Datical																																					
Sv	Os	Db																																					
Subversion		DBMaestro																																					
Cw	En	Dp		21	Os	22	Fm	23	Os	24	Fr	25	Fr	26	Fm	27	En	28	Fr	29	En	30	En	31	Os	32	Fm	33	En	34	Pd	35	Os	36	Os				
ISPW		Delphix		Jn	Jenkins	Cs	Codeship	Fn	FitNesse	Ju	JUnit	Ka	Karma	Su	SoapUI	Ch	Chef	Tf	Terraform	XLd	Xebialabs XL Deploy	Ud	UrbanCode Deploy	Ku	Kubernetes	Cc	CA CD Director	Pr	Plutora Release	Al	Alibaba Cloud	Os	OpenStack	Ps	Prometheus				
At	Pd	Rg		37	Pd	38	Fm	39	Pd	40	Fm	41	Fr	42	Fr	43	Os	44	Pd	45	En	46	Os	47	En	48	Os	49	Os	50	Pd	51	Fm	52	Pd	53	Pd	54	En
Artifactory		Redgate		Ba	Bamboo	Vs	VSTS	Se	Selenium	Jm	JMeter	Ja	Jasmine	Sl	Sauce Labs	An	Ansible	Ru	Rudder	Oc	Octopus Deploy	Go	GoCD	Ms	Mesos	Gke	GKE	Om	OpenMake	Cp	AWS CodePipeline	Cy	Cloud Foundry	It	ITRS				
Nx	Pd	Fw		55	Os	56	Os	57	Os	58	Fm	59	Os	60	Fr	61	Fm	62	Pd	63	En	64	Os	65	Fm	66	En	67	Os	68	Pd	69	Os	70	Os	71	Pd	72	Pd
Nexus		Flyway		Tr	Travis CI	Tc	TeamCity	Ga	Gatling	Tn	TestNG	Tt	Tricentis Tosca	Pe	Perfecto	Pu	Puppet	Pa	Packer	Cd	AWS CodeDeploy	Ec	ElectricCloud	Ra	Rancher	Aks	AKS	Rk	Rkt	Sp	Spinnaker	Ir	Iron.io	Mg	Moogsoft				
Bb	Fm	Pf		73	Fm	74	En	75	Fm	76	Pd	77	Fr	78	Os	79	Os	80	En	81	Os	82	Os	83	En	84	En	85	En	86	Pd	87	Fm	88	Os	89	Os	90	Os
BitBucket		Perforce		Cr	Circle CI	Cb	AWS CodeBuild	Cu	Cucumber	Mc	Mocha	Lo	Locust.io	Mf	Micro Focus UFT	Sa	Salt	Ce	CFEngine	Eb	ElasticBox	Ca	CA Automic	De	Docker Enterprise	Ae	AWS ECS	Cf	Codefresh	Hm	Helm	Aw	Apache OpenWhisk	Ls	Logstash				



Enterprise DevOps

 Follow @xebialabs

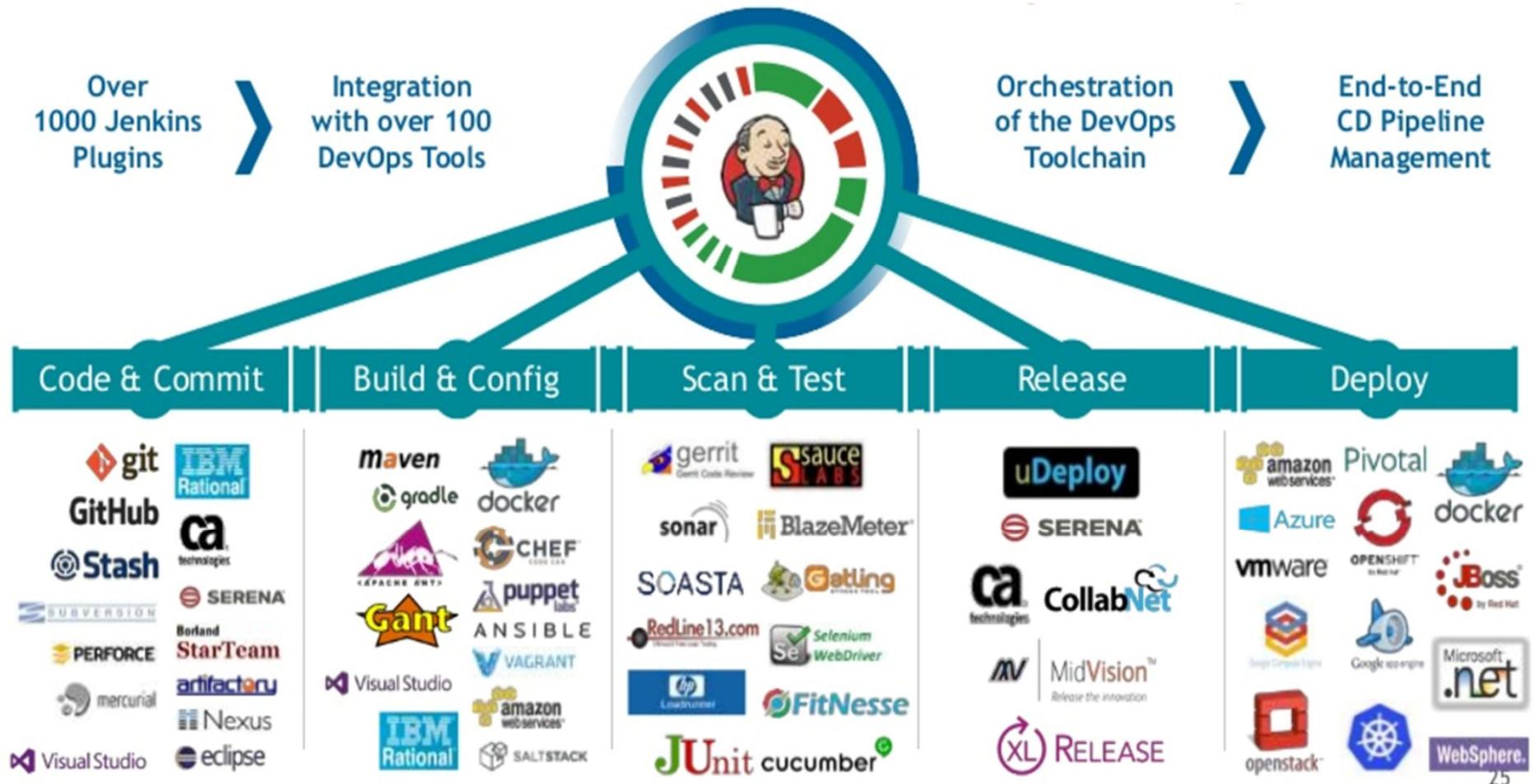
8 - Visão Geral das Possíveis Ferramentas a serem utilizadas:



8 - Visão Geral das Possíveis Ferramentas a serem utilizadas:



8 - Visão Geral das Possíveis Ferramentas a serem utilizadas:



8 - Visão Geral das Possíveis Ferramentas a serem utilizadas: DevSecOps

Dev & AppSec Tool Integration



klocwork
a Rogue Wave Company



VERACODE



VERACODE

Gitrob



RAPTOR



VERACODE



Chef Audit Mode



Code



Manage



Store



Build



Deploy

8 - Visão Geral das Possíveis Ferramentas a serem utilizadas: DevSecOps



8 - Visão Geral das Possíveis Ferramentas a serem utilizadas: APIS Back-End

DESIGNED BY
Mehdi Medjaoui

POWERED BY
spoke

The API Landscape

Last Update: February, 2017

Business Processes as an API/API-as-a-Product/Transactional APIs (43)



API Lifecycle platforms (37)



Backend Building Tools/Microservices (13)



API Abstraction / Integration Platforms (24)

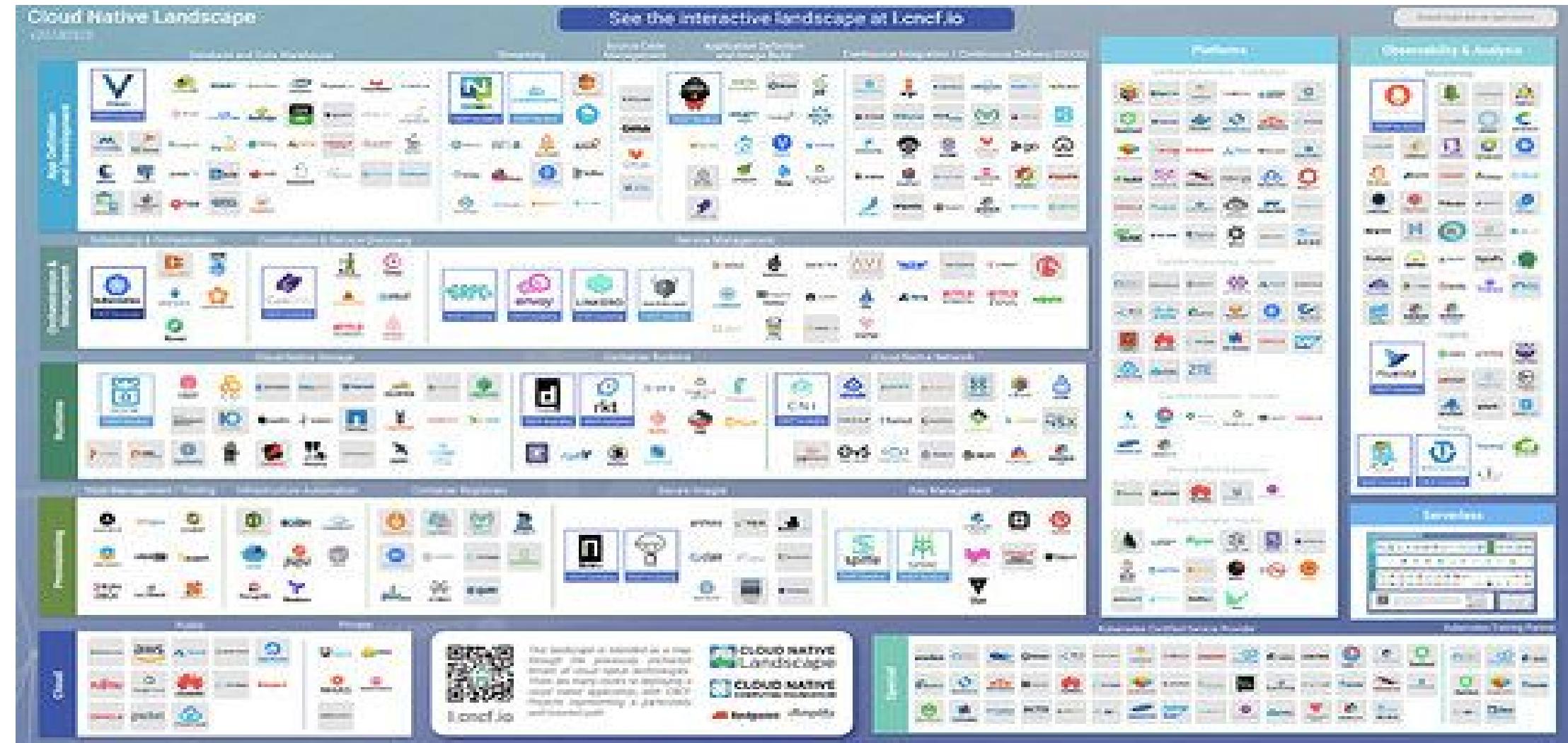


Visão Geral das Possíveis Ferramentas a serem utilizadas: OPEN SSource

OPEN SOURCE TECHNOLOGY LANDSCAPE v2

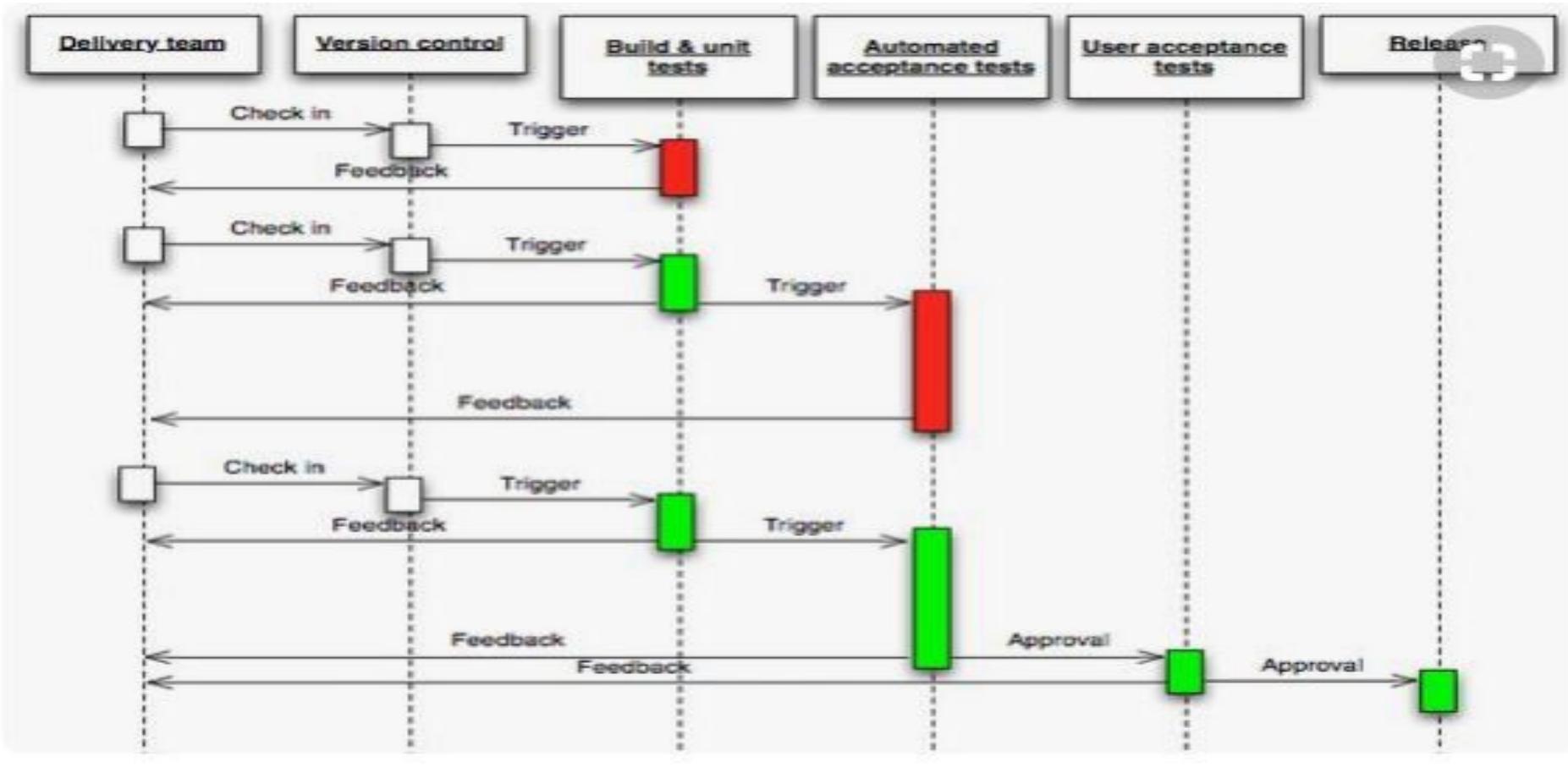


8 - Visão Geral das Possíveis Ferramentas a serem utilizadas: CLOUD



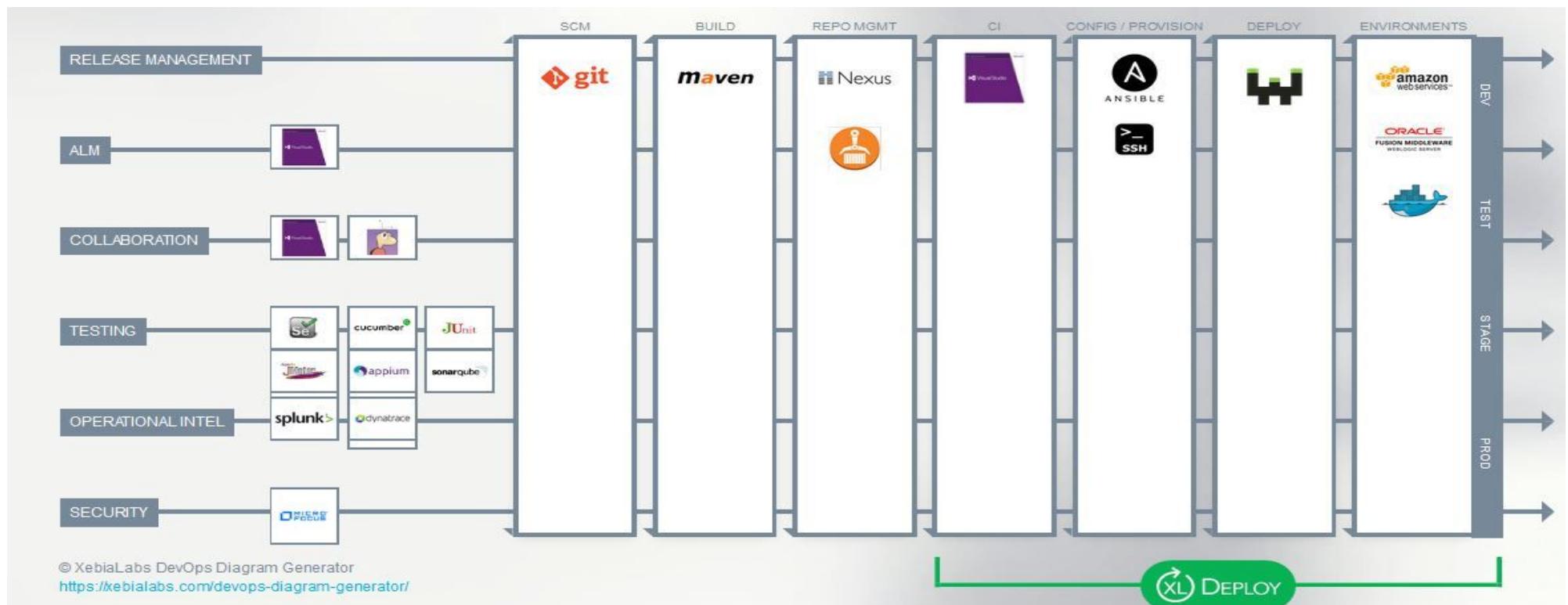
9 - Pipeline DevOps com Ferramentas

Exemplo de Fluxo com Automação para CI & CD



9 - Pipeline DevOps com Ferramentas

Exemplo de Pipeline com Ferramentas e Automação



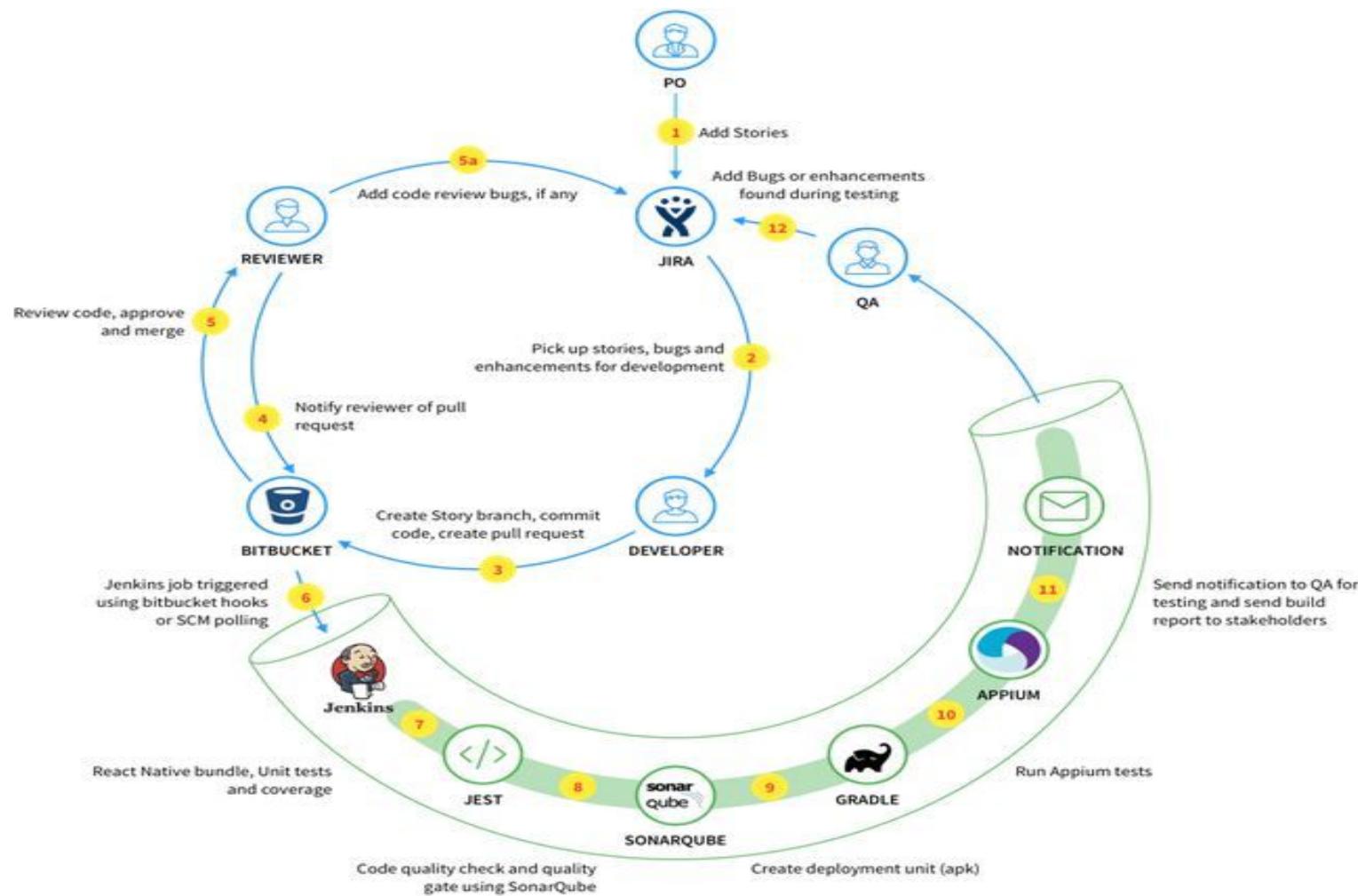
What tools do you currently use?

SHARE FAQ DOWNLOAD PRINT HIDE XL CLEAR ALL

Git × Maven × Team Foundation Server × Nexus × AWS ECR × Selenium × Cucumber × JUnit × JMeter × Appium × SonarQube × Jasmine × Cobertura × webhook × Ansible × SSH × Amazon Web Services × Team Foundation Server × Splunk × Dynatrace × WebLogic × Bugzilla × Team Foundation Server × Fortify WebInspect × Splunk × Dynatrace × Docker × Liquibase ×

9 - Pipeline DevOps com Ferramentas

Exemplo de Pipeline e Arquitetura com Ferramentas Atlassian



CATEGORIES

Apps
DevOps
Digital Transformation
GDPR
Productivity
Security
Splunk

META

[Log in](#)
[Entries RSS](#)
[Comments RSS](#)
[WordPress.org](#)



**Para onde ir, diante
de tantas opções e
DESAFIOS**

?

?

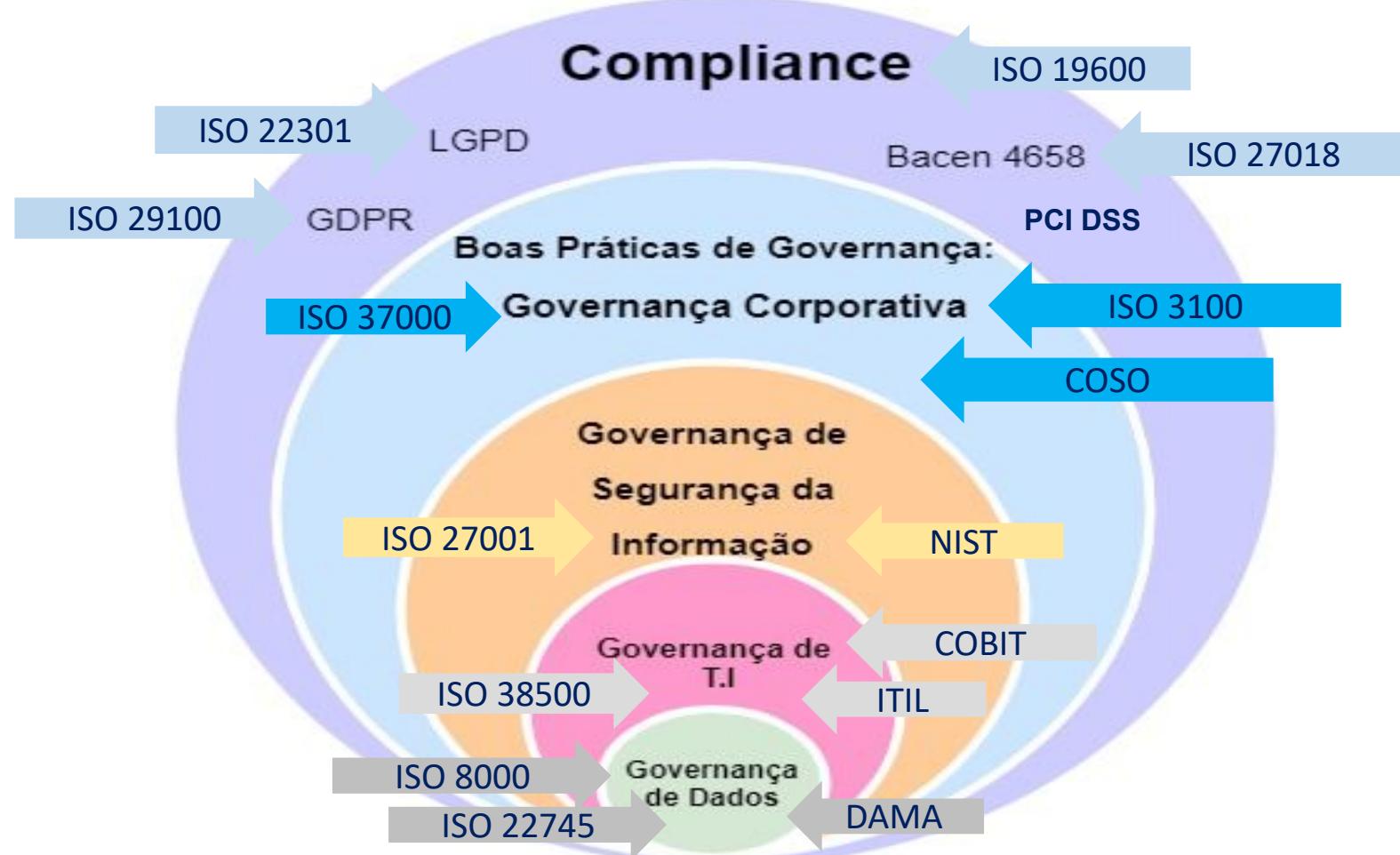
?

ITIL COBIT
DevSecOps

AGILE CLOUD

10 – Por Onde Comerçar: Governança

Modelos, Padrões e Boas



11 - COBIT 5

PERSPECTIVA DE IMPACTO POR DEVOPS

STAKEHOLDERS DRIVERS – DIRECIONADORES DE PARTES INTERESSADAS

AMBIENTE, EVOLUÇÃO TECNOLÓGICA, ROI,
ETC

INFLUÊNCIAS

STAKEHOLDERS NEEDS – NECESSIDADES DAS PARTES INTERESSADAS

OTIMIZAÇÃO DE
RISCOS

OTIMIZAÇÃO DE
RECURSOS

REALIZAÇÃO DE
BENEFÍCIOS

1 – CULTURA

2 – Controle

3 – Automação

4 -Visibilidade

ENTERPRISE GOALS - OBJETIVOS DA EMPRESA

CASCATEIA PARA

IT RELATED GOALS- OBJETIVOS RELACIONADOS A TI

CASCATEIA PARA

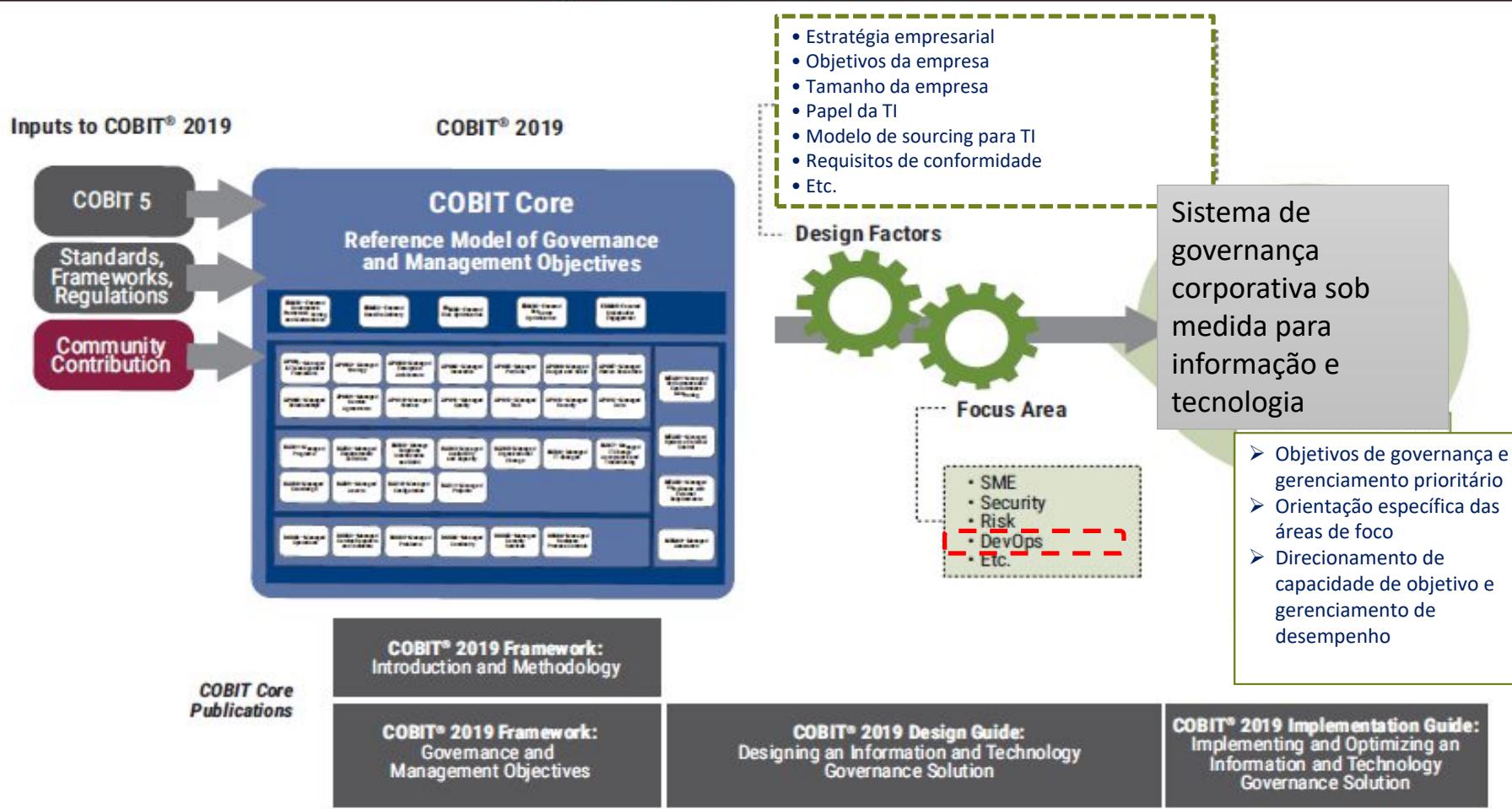
ENABLER GOALS- HABILITADORES DE OBJETIVOS

DEVSECOPS

11 - COBIT 2019

PERSPECTIVA DE IMPACTO POR DEVOPS

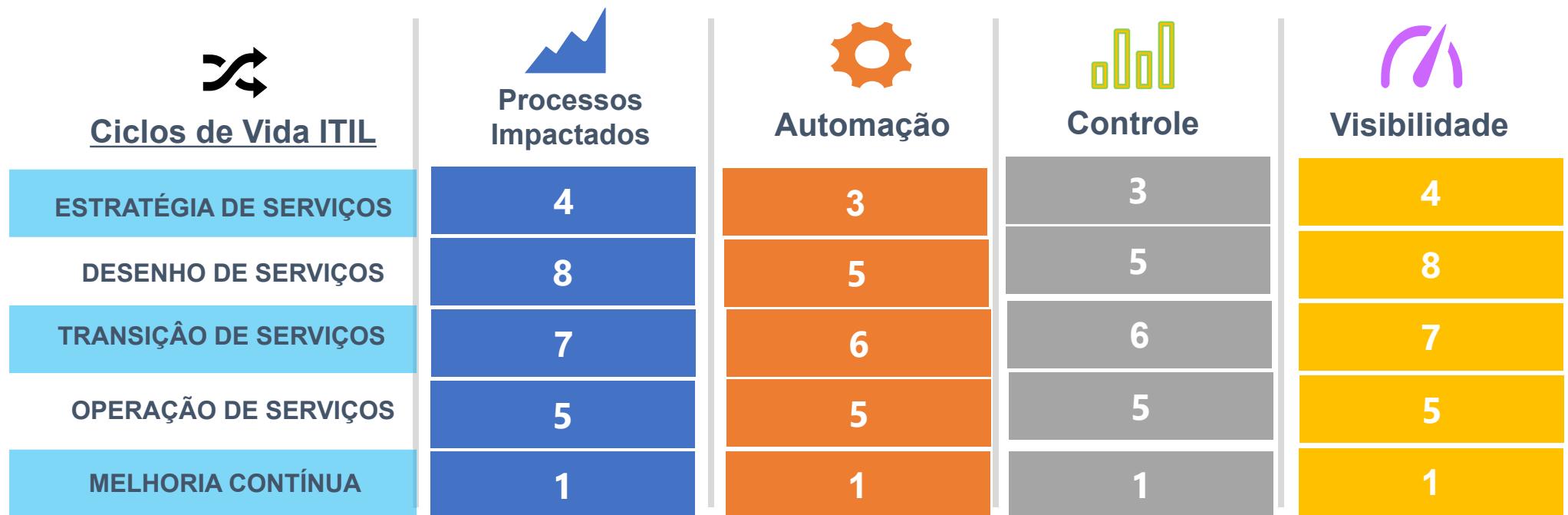
Figure 1.1—COBIT Overview



- 2 – Controle
- 3 – Automação
- 4 -Visibilidade

11 - Ciclos de Vida ITIL V3

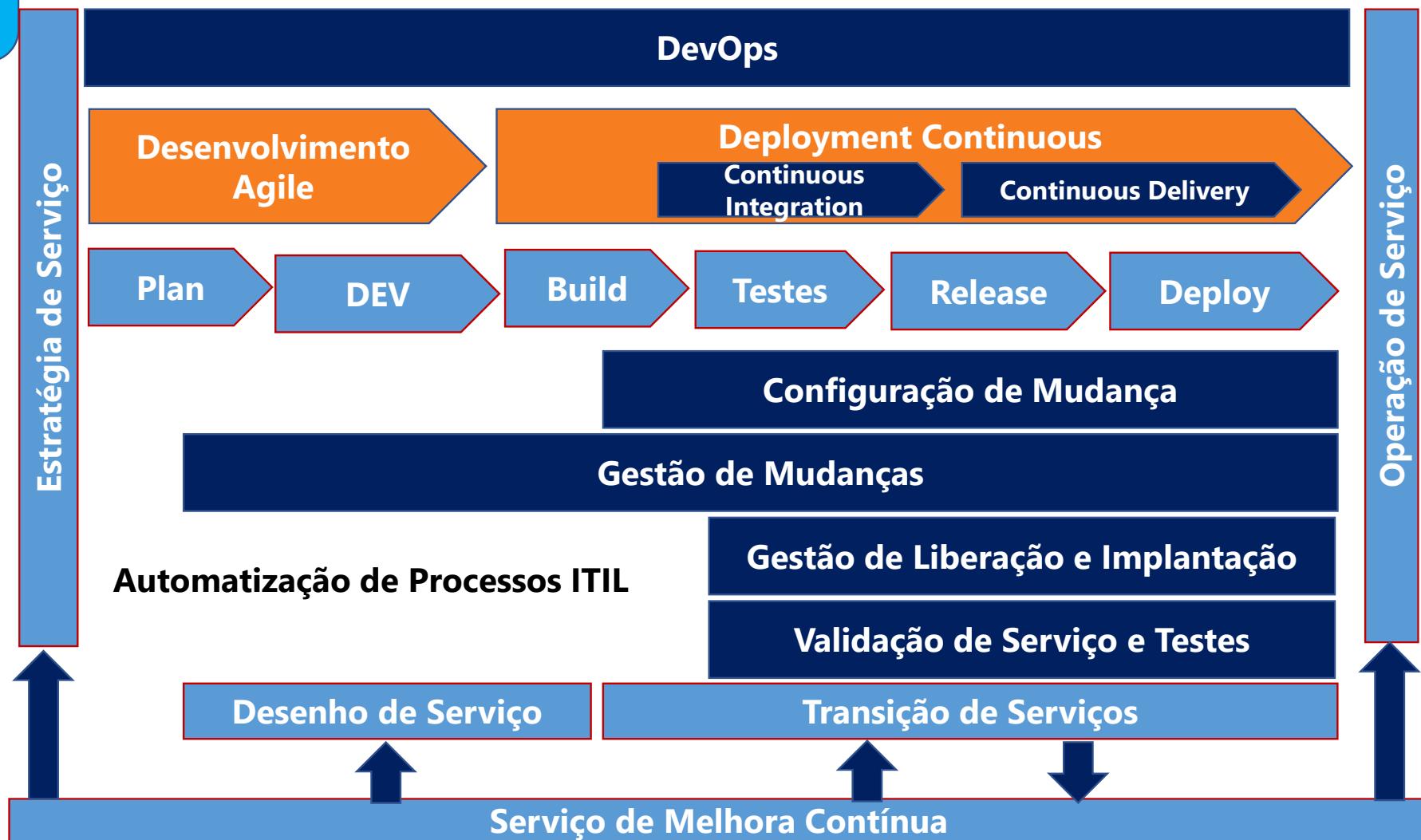
PERSPECTIVA DE IMPACTO POR DEVOPS



- 2 – Controle
- 3 – Automação
- 4 -Visibilidade

11 - Ciclos de Vida ITIL V3

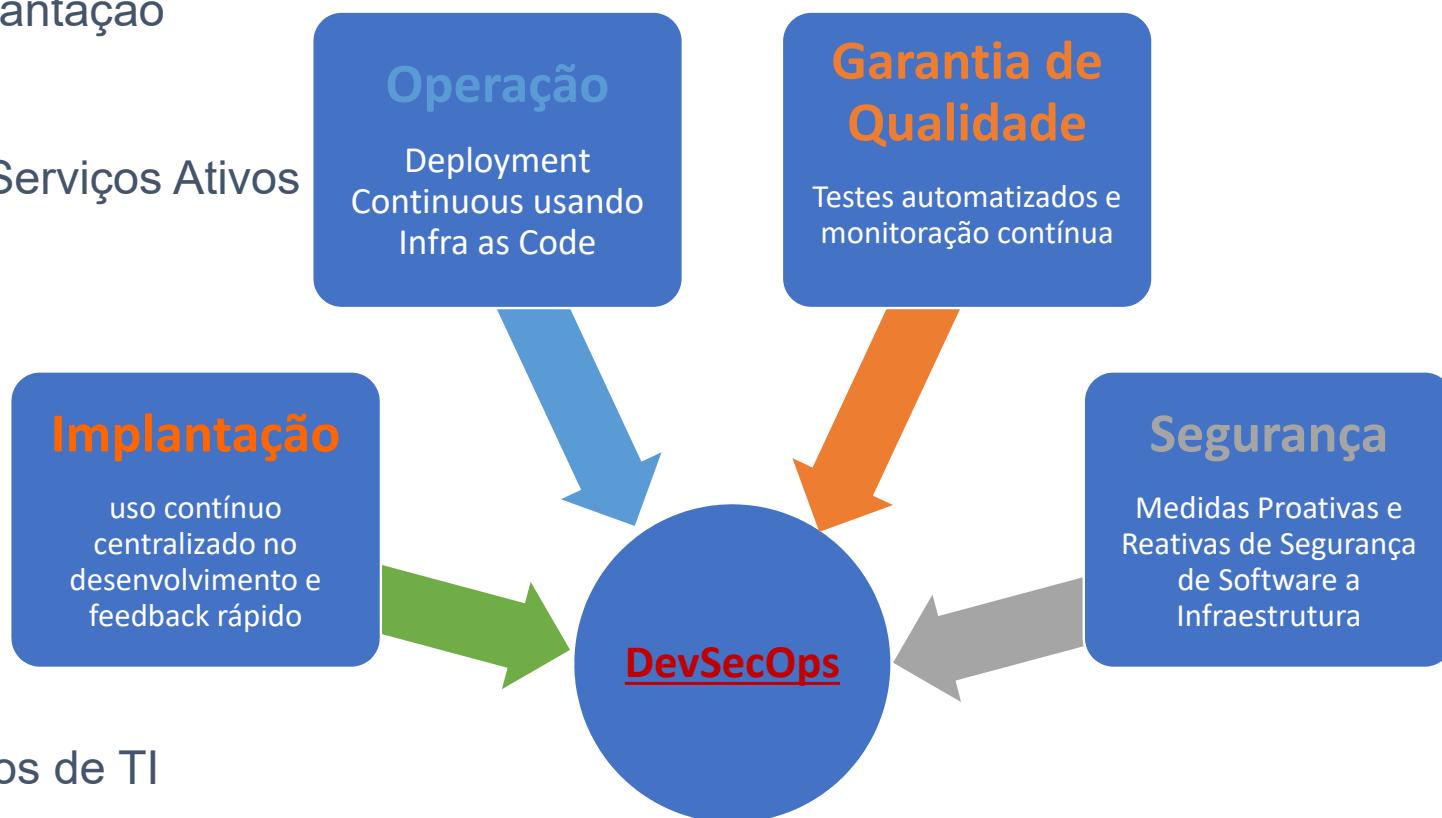
PERSPECTIVA DE IMPACTO POR DEVOPS



Processos Afetados / Modificados:

Gestão de Mudanças
Configuração de Mudanças
Gerenciamento de Liberação e Implantação
Validação e Teste de Serviço
Gerenciamento de Configuração e Serviços Ativos
Gestão de Eventos
Gerenciamento de Incidentes
Gerenciamento de Problemas
Gerenciamento de Acesso
Gerenciamento de Capacidade
Gerenciamento de Disponibilidade
Gestão da Continuidade dos Serviços de TI
Gerenciamento de Segurança da Informação

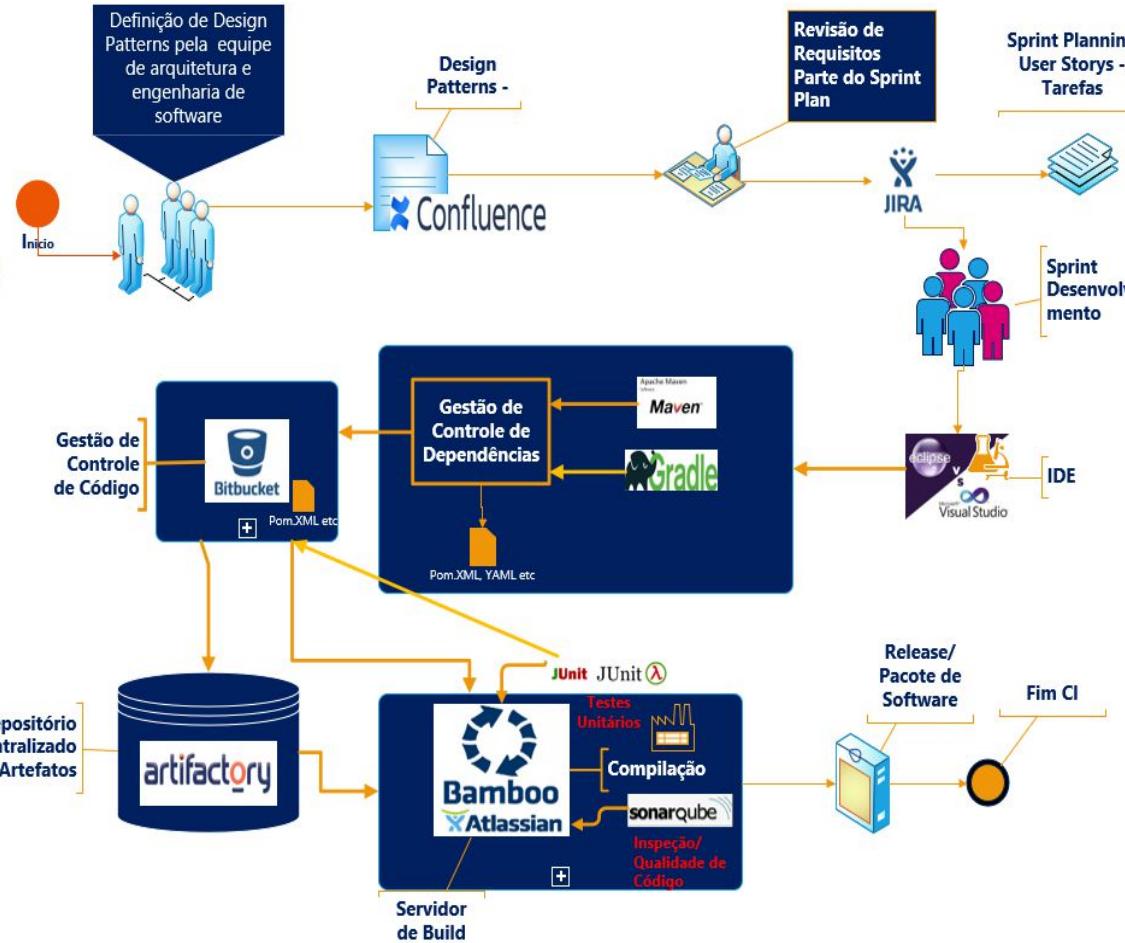
11 - Processos ITIL v3 + DevSecOps:



12 - Checklist Continuous Integration:

- Design Patterns para Escrita de Código
- Aderência do Código a Arquitetura de Software planejada (Controle de Qualidade, Análise estática)
- Gestão de Controle de Configuração
- Gestão de Controle de Código (Git Flow e SCM)
- Gestão e Controle de Dependências
- Repositório Centralizado de Artefatos e Componentes versionados
- Testes unitários automatizados
- Testes de Integração automatizados
- Servidor de build

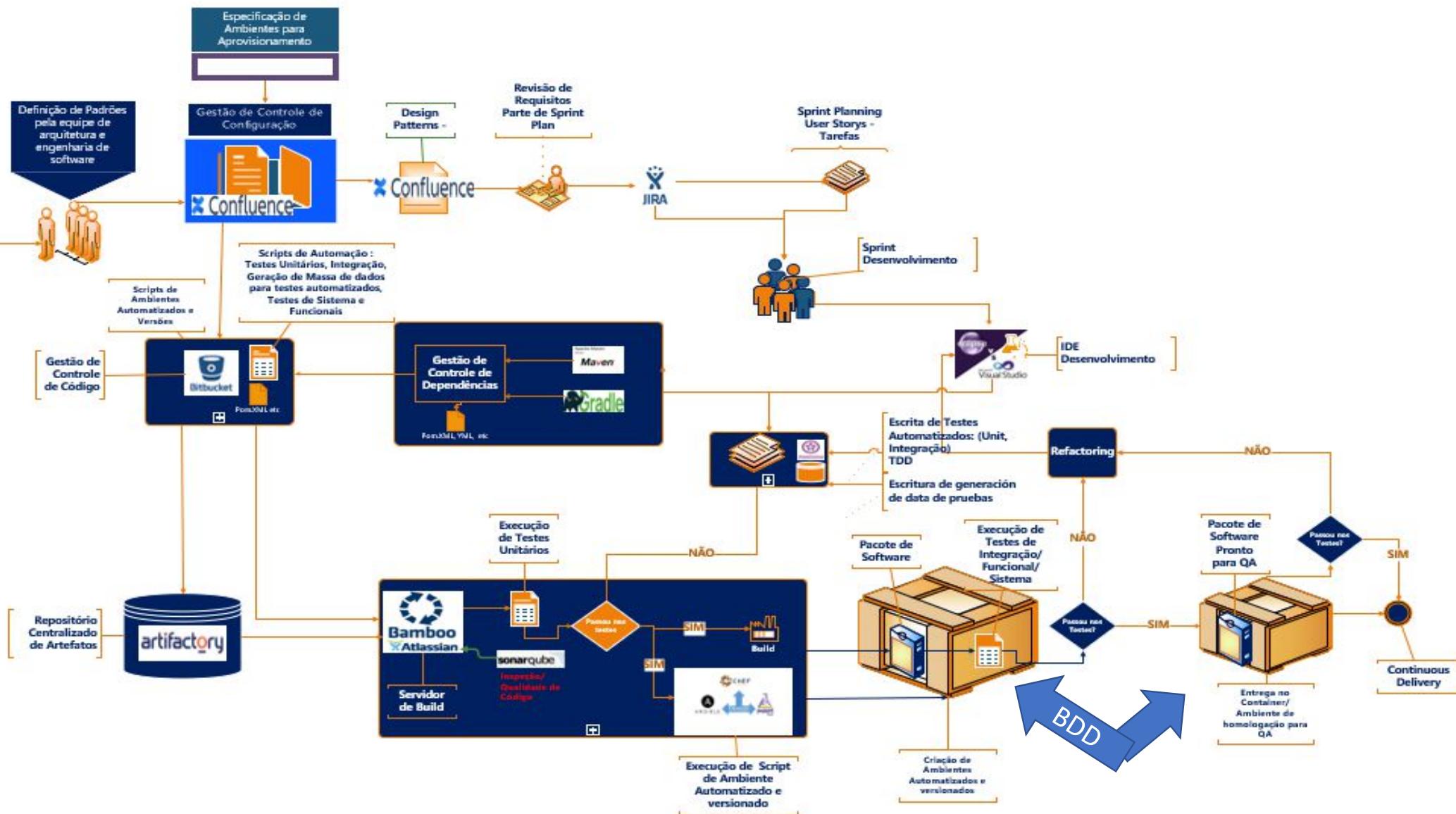
Continuous Integration



12 - Checklist Continuous Delivery:

- Integração Contínua
- Controle de Defeitos Documentado (gestão de problemas e incidentes)
- Controle de código de infraestrutura (ambientes)
- Testes funcionais Automatizados
- Testes de Segurança automatizados e manuais
- Geração de massa de testes de forma automatizada
- Fluxo de verificação automática de defeitos corrigidos
- Ambiente de Testes apartado e versionado
- Criação de ambientes de forma versionada e automatizada (conteinerização)
- Processo automatizado de solicitações de mudança
- Criação de Registros

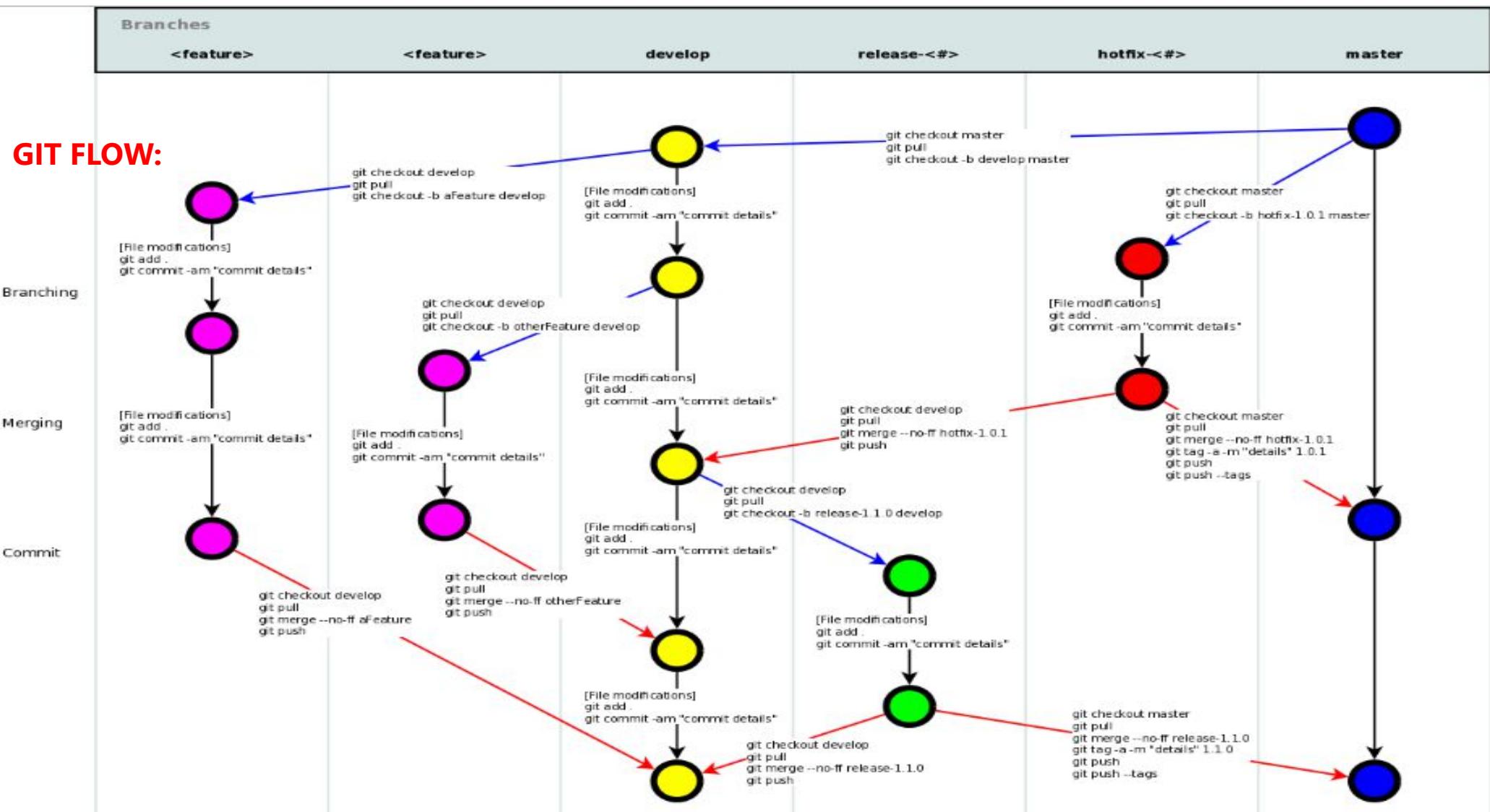
Continuous Delivery



13 - Pipeline DevOps GitFlow – Um modelo para organização de Branches

O GIT FLOW ESTABELECE ALGUMAS REGRAS DE **NOMENCLATURAS** PARA TIPOS DE BRANCHES ENQUANTO, AO MESMO TEMPO, DEFINE O QUE **CADA TIPO DE BRANCH FAZ**. PARA REFERÊNCIA, SEGUE UMA LISTA DOS TIPOS DE BRANCHES DEFINIDOS PELO GIT FLOW E SUAS RESPECTIVAS DESCRIÇÕES:





13 - Pipeline DevOps GitFlow – Um modelo para organização de Branches

- **Branch master** - É a branch que contém código em nível de **produção**, ou seja, o código mais maduro existente na sua aplicação. Todo o código novo produzido eventualmente é juntado com a branch **master**, em algum momento do desenvolvimento.
- **Branch develop** - É a branch que contém código em nível preparatório para o próximo deploy. Ou seja, quando features são terminadas, elas são juntadas com a branch **develop**, testadas (em conjunto, no caso de mais de uma feature), e somente depois as atualizações da branch **develop** passam por mais um processo para então ser juntadas com a branch **máster**.
- **Branches feature/*** - São branches no qual são desenvolvidos recursos novos para o projeto em questão. Essas branches tem por convenção nome começando com **feature/** (exemplo: **feature/new-layout**) e são criadas a partir da branch **develop** (pois um recurso pode depender diretamente de outro recurso em algumas situações), e, ao final, são juntadas com a branch **develop**.

13 - Pipeline DevOps GitFlow – Um modelo para organização de Branches

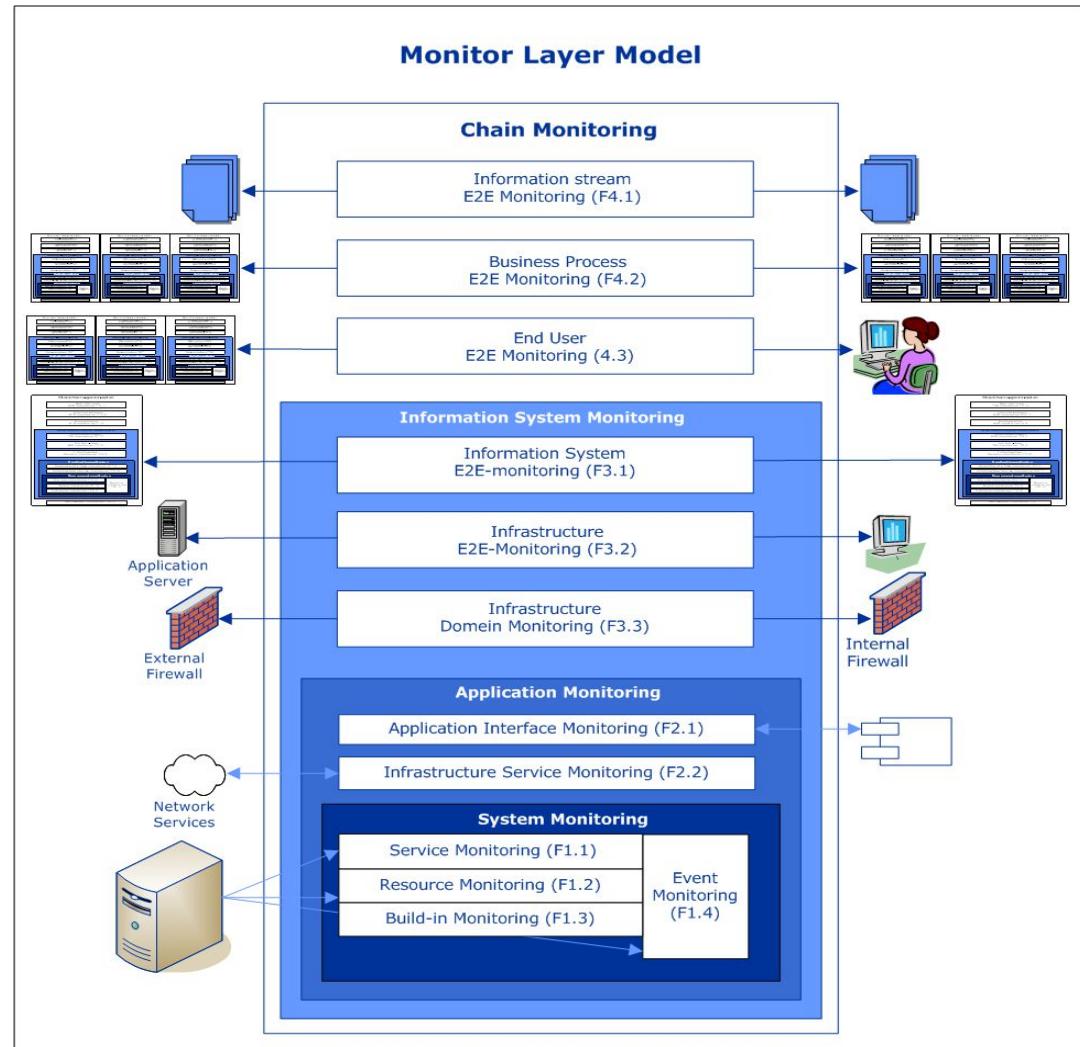
- **Branches hotfix/*** - São branches no qual são realizadas correções de bugs críticos encontrados em ambiente de produção, e que por isso são criadas a partir da branch **master**, e são juntadas diretamente com a branch **master** e com a branch **develop** (pois os próximos deploys também devem receber correções de bugs críticos, certo?). Por convenção, essas branches tem o nome começando com **hotfix/** e terminando com o próximo sub-número de versão (exemplo: **hotfix/2.31.1**), normalmente seguindo as regras de algum padrão de versionamento, como o padrão do versionamento semântico.
- **Branches release/*** - São branches com um nível de confiança maior do que a branch **develop**, e que se encontram em nível de preparação para ser juntada com a branch **master** e com a branch **develop** (para caso tenha ocorrido alguma correção de bug na branch **release/*** em questão). Note que, nessas branches, bugs encontrados durante os testes das *features* que vão para produção podem ser corrigidos mais tranquilamente, **antes** de irem efetivamente para produção. Por convenção, essas branches tem o nome começando com **release/** e terminando com o número da próxima versão do software (seguindo o exemplo do *hotfix*, dado acima, seria algo como **release/2.32.0**), normalmente seguindo as regras do versionamento semântico, como falado acima.

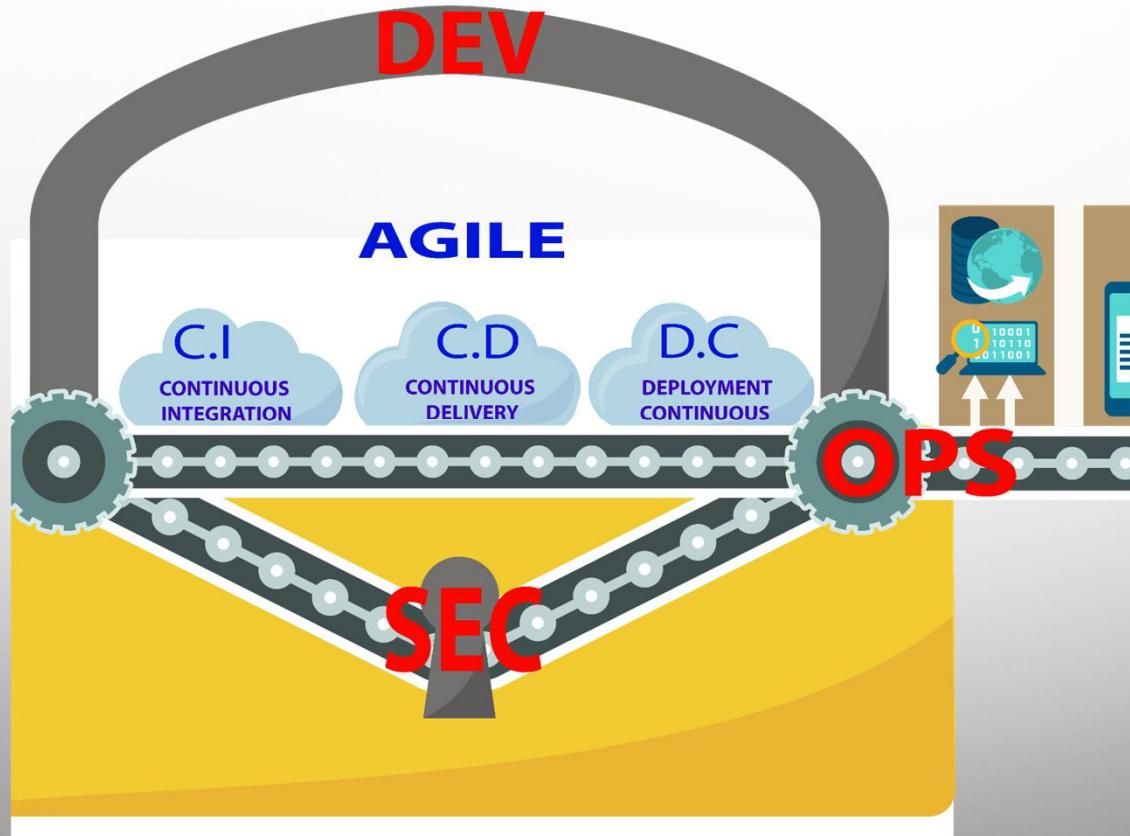
14 - Boas Práticas de Monitoração Contínua

Os seguintes padrões de melhores práticas se aplicam a um dispositivo de monitoramento DevOps limpo durante a programação:

- S1. Cada evento tem um número único
- S2. Cada evento refere-se ao item de configuração do software que fez a exceção.
- S3. Cada evento possui um código de gravidade atribuído.
- S4. Cada evento também define a ação de recuperação.
- S5. Cada novo evento será registrado no backlog do produto da equipe OPS.

Durante a fase de compilação, deve-se saber quais funções de monitor são aplicáveis. A equipe de atendimento e operações são partes interessadas importantes para serem envolvidas.





Incluindo o
SEC no
DevOps:
DEVSECOPS



Mitos

15 - Princípios: Security by Design

1 - Minimizar a superfície de área de ataque através da utilização de patterns de desenvolvimento de código e boas práticas de desenvolvimento seguro.

2 - Estabelecimento de Padrões seguro através da utilização de senhas fortes, ciclo de vida de senhas, autenticação multifator e tokens.

3 - Princípio do Menor Privilégio através da criação de contas com a menor quantidade de privilégios necessários para executar seus processos de negócios. Isso engloba direitos de usuário, permissões de recursos e outros.

4 -Princípio da defesa em profundidade utilizando um controle que seria razoável, mais controles que abordam riscos de diferentes maneiras são melhores.

5 - Falhar com segurança, ou seja, os aplicativos geralmente não processam transações por vários motivos. A forma como eles falham podem determinar se um aplicativo é seguro ou não, por exemplo se expõe, endpoints, paths, strings de conexão etc.

15 - Princípios: Security by Design

6 - Não Confie nos Serviços, ou seja, todos os sistemas externos com parceiros, integradores, brokers, devem ser tratados de maneira semelhante, os dados devem ser sempre verificados.

7 - Separação de deveres através da determinação de papéis que têm diferentes níveis de confiança do que usuários normais. Em particular, os administradores são diferentes dos usuários normais, utilizando RBAC

8 - Evitar a segurança por obscuridade, ou seja, a segurança de um aplicativo não deve depender do conhecimento do código-fonte mantido em segredo. A segurança deve se basear em muitos outros fatores, incluindo políticas razoáveis de senha, defesa em profundidade.

9 - Mantenha a Segurança simples, onde os desenvolvedores devem evitar o uso de negativos duplos e arquiteturas complexas quando uma abordagem mais simples seria mais rápida e simples.

10 - Correção de Problemas de Segurança da maneira correta, quando um problema de segurança for identificado, é importante desenvolver um teste para ele e entender a causa raiz do problema. Quando padrões de design são usados, é provável que o problema de segurança seja difundido entre todas as bases de código, portanto é essencial desenvolver a correção correta sem introduzir regressões.

15 - Princípios: Security by Design

7. Evitar a segurança por obscuridade, ou seja, a segurança de um aplicativo não deve depender do conhecimento do código-fonte mantido em segredo. A segurança deve se basear em muitos outros fatores, incluindo políticas razoáveis de senha, defesa em profundidade.
8. Mantenha a Segurança simples, onde os desenvolvedores devem evitar o uso de negativos duplos e arquiteturas complexas quando uma abordagem mais simples seria mais rápida e simples.

15 - Checklist para Desenvolvimento Seguro de Software - SAST

1. Validação de entrada - 16
2. Codificação de saída - 6
3. Autenticação e Gerenciamento de Senhas - 34
4. Gerenciamento de Sessão - 18
5. Controle de Acesso - 23
6. Práticas Criptográficas - 6
7. Tratamento de erros e registro - 35
8. Segurança de Comunicação - 8
9. Configuração do Sistema - 15
10. Segurança do banco de dados - 13
11. Gerenciamento de arquivos - 14
12. Gerenciamento de Memória - 9
13. Práticas Gerais de Codificação -12

Total 214

15 - Incluindo o SEC no DevOps:

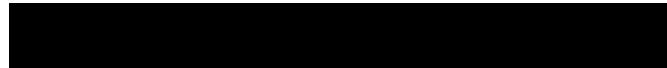
- Integre os testes de segurança e conformidade ao DevOps - os desenvolvedores podem gerenciar a segurança de dados sem precisar deixar suas ferramentas de CI / CD;
- Torne mais fácil para os desenvolvedores usar ferramentas de teste de segurança ou, melhor, automatize-os;
- Origem de components, qual sua procedencia - se for de código aberto, verifique vulnerabilidades e erros de configuração (nunca deixar o default).;
- Mente aberta para novas ferramentas e abordagens;
- Análise de código - quanto menor o pedaço de código, mais rapidamente podem ser identificados os pontos fracos;
- Investigação de ameaças - identifique ameaças potenciais com cada atualização de código e seja capaz de responder rapidamente;
- Avaliação de vulnerabilidade - identificar novas vulnerabilidades com análise de código;

Pontos de Atenção para adoção DevSecOps:

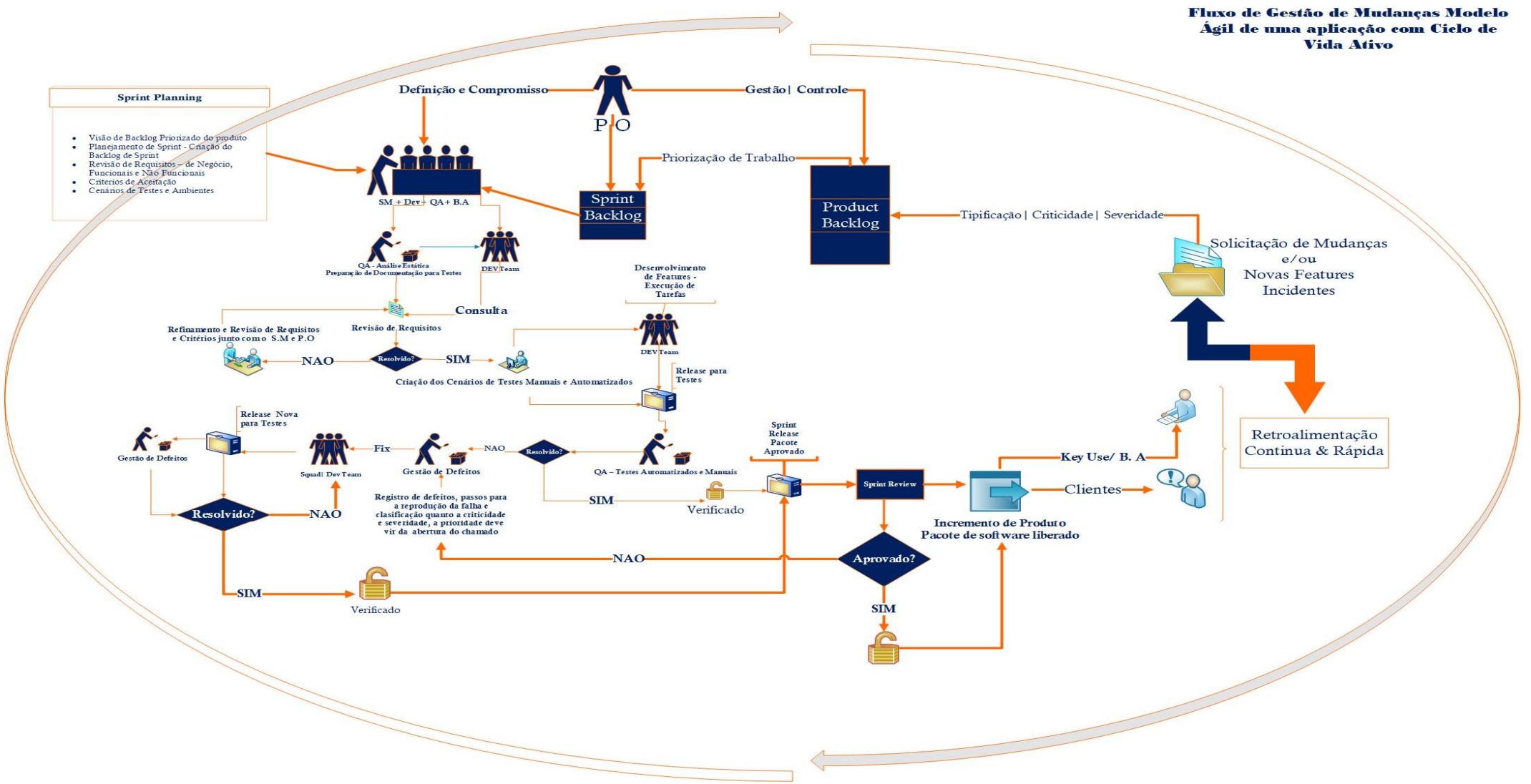
- Com que rapidez são as respostas e os patches? Gestão de Atualizações
- Treinamento - treinamento contínuo em segurança para os envolvidos no ciclo de vida do desenvolvimento;
- O DevSecOps ainda é uma tendência emergente, com alguns estudos indicando que isso poderia ser incorporado em 80% das equipes de desenvolvimento rápido até 2021, mas ainda não há um roteiro claro sobre como ele deve ser implementado, e o número de profissionais com conhecimento de funções ainda é relativamente baixa.

15 - Gestão de Mudanças:

- Sua função é garantir que todas as alterações que possam afetar a infraestrutura sejam feitas de uma forma controlada, e também que esta mudança seja refletida na Base de Dados do Gerenciamento de Configuração, ou BDGC.
- A tarefa Planejar Controle de Mudança e Configuração do Projeto estabelece um plano adequado para gerenciar e controlar mudanças para os artefatos desenvolvidos como produtos de trabalho do processo de desenvolvimento do software.
- A tarefa Criar Ambientes para CM (Comitê de Mudanças) do Projeto estabelece um ambiente, no qual o produto geral possa ser desenvolvido, construído e disponibilizado aos envolvidos.
- A tarefa Monitorar e Relatar o Status de Configuração fornece visibilidade para a atividade de alteração de configuração pelo monitoramento e relatório contínuos.
- A tarefa Alterar e Liberar Itens de Configuração gerencia os artefatos do projeto e o trabalho envolvido a partir da criação inicial como artefatos particulares por meio da entrega e da disponibilidade geral para a equipe do projeto e dos envolvidos.
- A tarefa Gerenciar Baselines e Releases garante que conjuntos consistentes de artefatos dependentes ou relacionados possam ser identificados como parte de uma "baseline" para várias partes de uma "baseline" para várias finalidades, como a identificação de release, versões do produto, maturidade do artefato ou integralidade.



15 - Gestão de Mudanças:



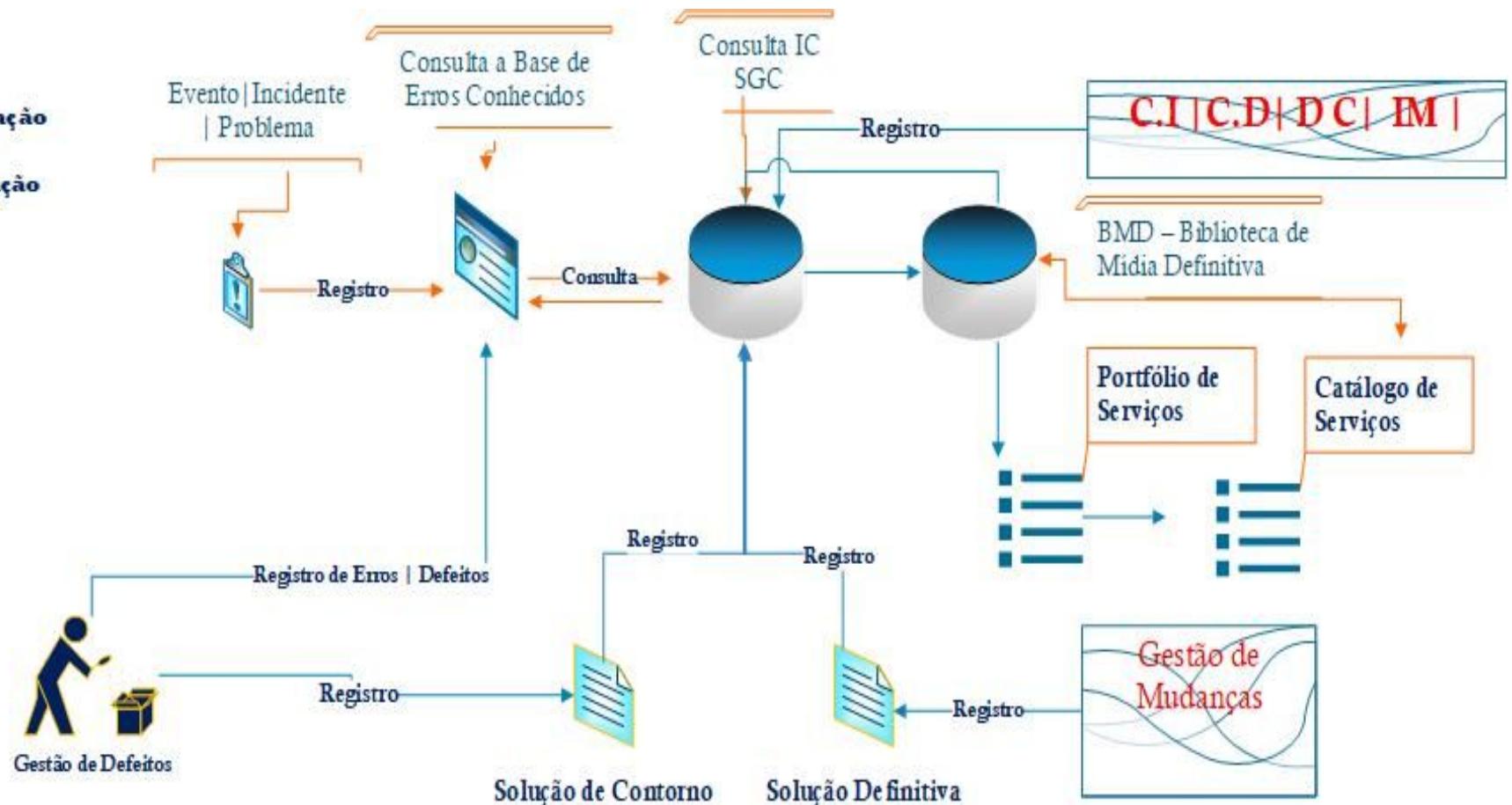
15 - Gestão de Configuração

- **GCS** ou em inglês *Software Configuration Management (SCM)*, que é um conjunto de atividades de apoio que permitem controlar as mudanças que ocorrem no desenvolvimento de software, mantendo a estabilidade na evolução do projeto.
- A **Gerência de Configuração de Software** responde a algumas questões como: Quais mudanças aconteceram no sistema? Por que essas mudanças aconteceram? O sistema continua íntegro mesmo após as mudanças?
- Para se entender a Gerência de Configuração de Software é interessante definir-se o que vem a ser configuração.
- Configuração de um sistema é uma coleção de versões específicas de itens de configuração como hardware ou software que são combinados de acordo com procedimentos específicos de construção para servir a uma finalidade particular. A Gerência de Configuração de Software por sua vez é uma disciplina que identifica a configuração de um sistema em diferentes pontos no tempo com a finalidade de controlar sistematicamente as mudanças realizadas, mantendo a integridade e rastreabilidade da configuração através do ciclo de vida do sistema.



15 - Gestão de Configuração:

**Fluxo de Gestão de Configuração
Modelo Agil**
Ciclo de Vida Ativo | Produção



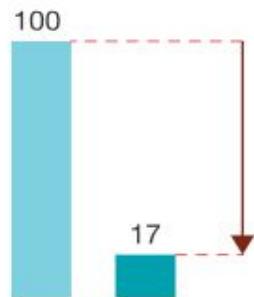
15 - Ganhos com Adoção de DevOps:

The value of adopting DevOps can be significant.

Indexed to 100

Improvement in time to market

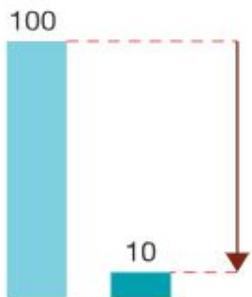
Average number of days from code completion to live production



- Eliminate rework through integrated change management and automated deployment and testing

Reduction in cycle time

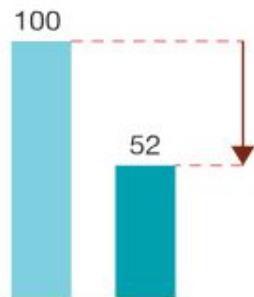
Number of days to update servers and the IT environment



- Eliminate wait time and rework through standardized processes
- Eliminate non-value-added work through automation

Improvement in productivity

Average number of DevOps handoffs per processing activity



- Eliminate wait time and rework through improved development and operations communication

Fonte: McKinsey

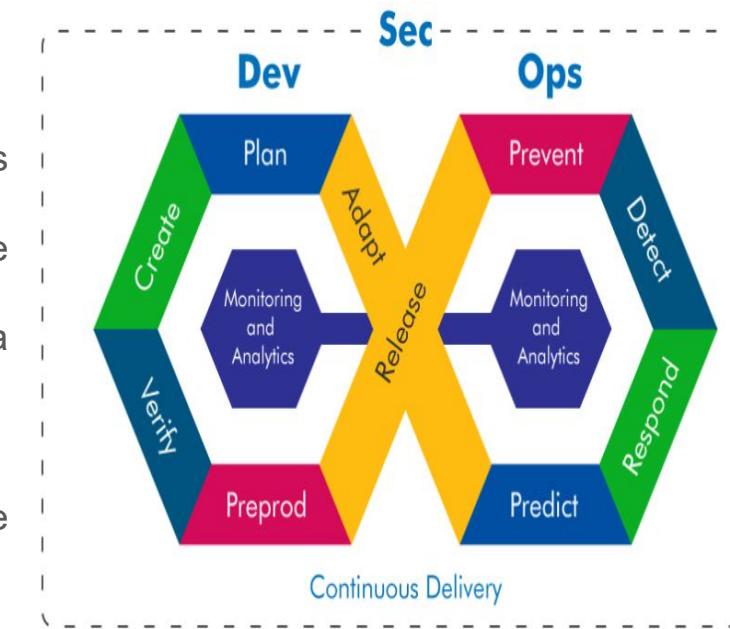
A pesquisa ouviu 1.425 executivos e líderes de TI em 15 países: Alemanha, Austrália, Brasil, Canadá, China, Coréia do Sul, Espanha, Estados Unidos, França, Índia, Inglaterra, Itália, Japão, Suíça e Cingapura.

A alta no faturamento associada ao DevOps nas empresas brasileiras supera a média global, que foi de 19%. A manutenção e a correção dos aplicativos ficou 21% mais rápida no mundo, e 30% no Brasil. Já a colaboração entre departamentos aumentou 21% no mundo e cresceu 29% no Brasil.

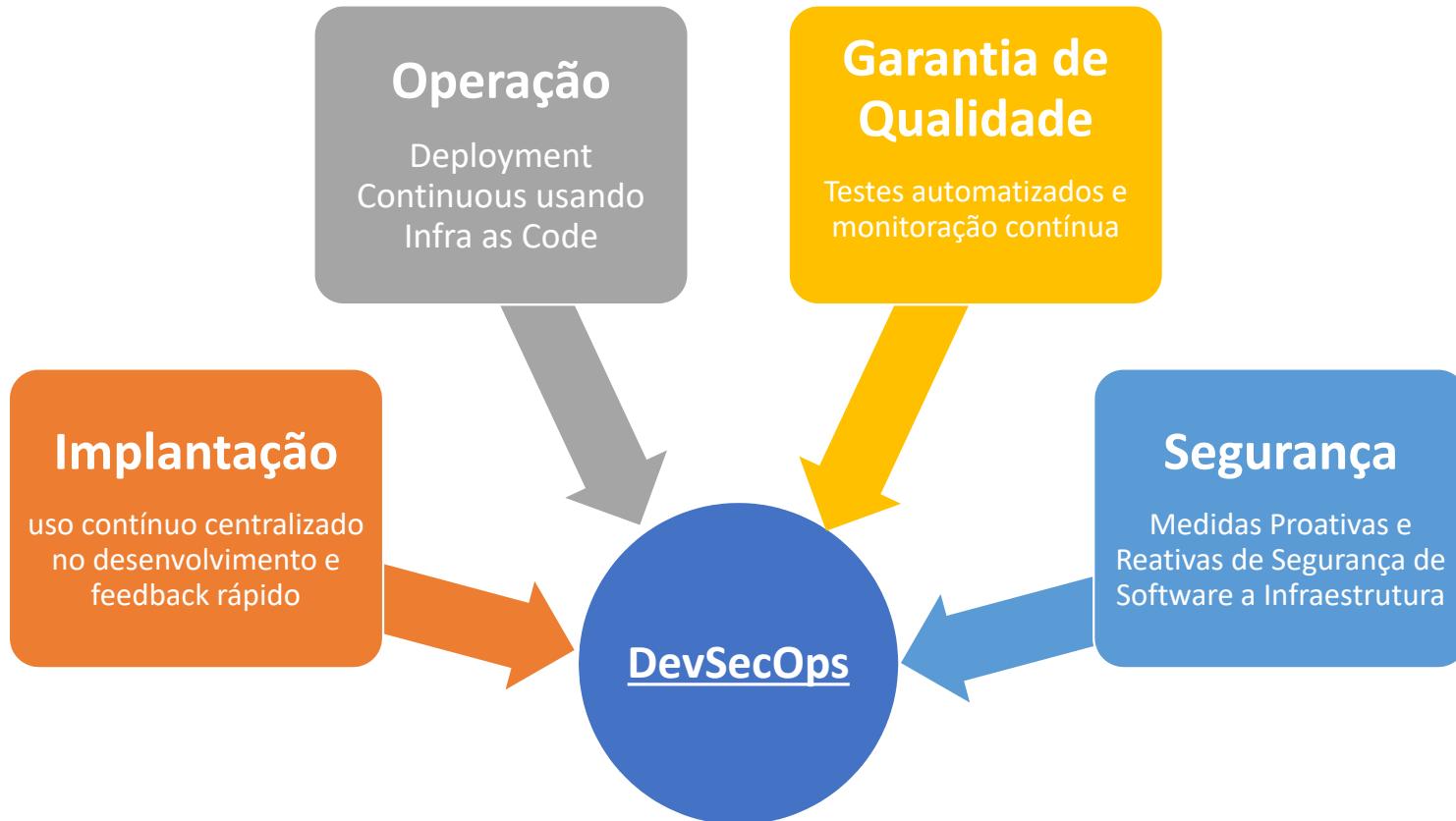
Para 78% dos entrevistados brasileiros, o maior ganho com a adoção às práticas de DevOps foi o aumento na frequência de lançamentos de softwares (78%), seguido pela redução no tempo gasto com manutenção e correção de aplicativos (71%) e a diminuição do *time-to-market* (70%). São esperados ainda resultados como a disponibilidade do software em mais plataformas (72%), a melhoria do desempenho dos aplicativos (59%) e mais colaboração entre os departamentos (48%).

15 - Ganhos com Adoção de DevSecOps:

- Mais automação em todos os estágios do ciclo de vida reduz a probabilidade de erro humano;
- Redução no tempo de inatividade ou os ataques;
- Criar segurança no ciclo de vida do produto desde o início garante que os processos de segurança sejam ativados automaticamente, com o objetivo de permitir a criação de produtos inovadores e, ao mesmo tempo, garantir a segurança do código;
- É uma continuação da abordagem colaborativa do DevOps e promove a idéia de responsabilidade coletiva pela segurança;
- Criação de ferramentas e técnicas para facilitar essa colaboração e permitir que a segurança acompanhe os processos de construção iterativos.



15 - Ganhos com Adoção de DevSecOps:



16 - Mind MAP DevSECOps

16 - Dev+OPS

X

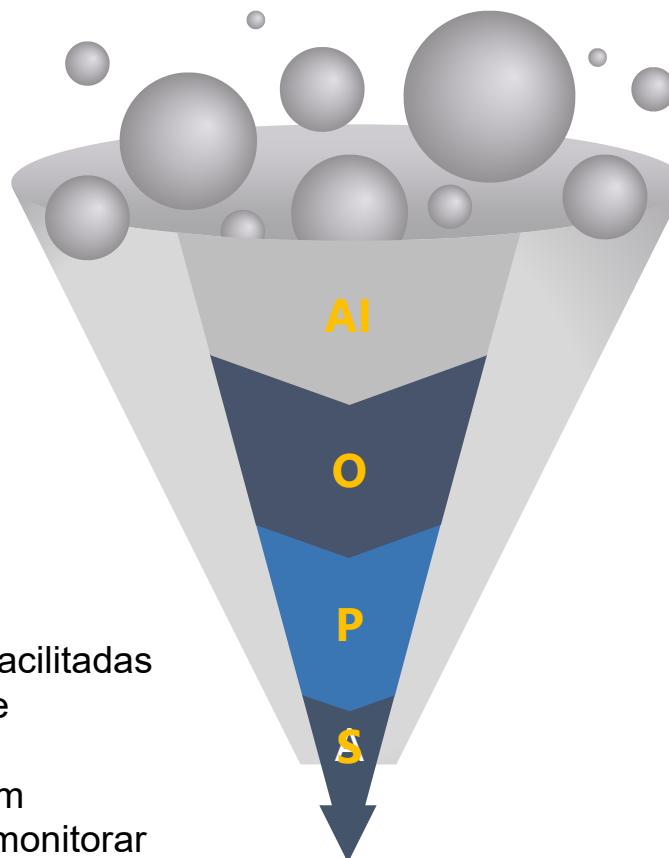
AI+OPS

Combinação de:

- Filosofia Cultural, Práticas e Ferramentas
- Entrega de aplicações em serviços em alta velocidade
- Nada de silos, feedback rápido
- DevOps inclui pensamento de sistemas holísticos
- Automação de trabalho “penoso”, repetitivo

Habilita:

- Metodologias de Desenvolvimento Ágil facilitadas por ciclos de liberação e implantação
- Operações colabora com Desenvolvimento para monitorar Operações de auto serviço

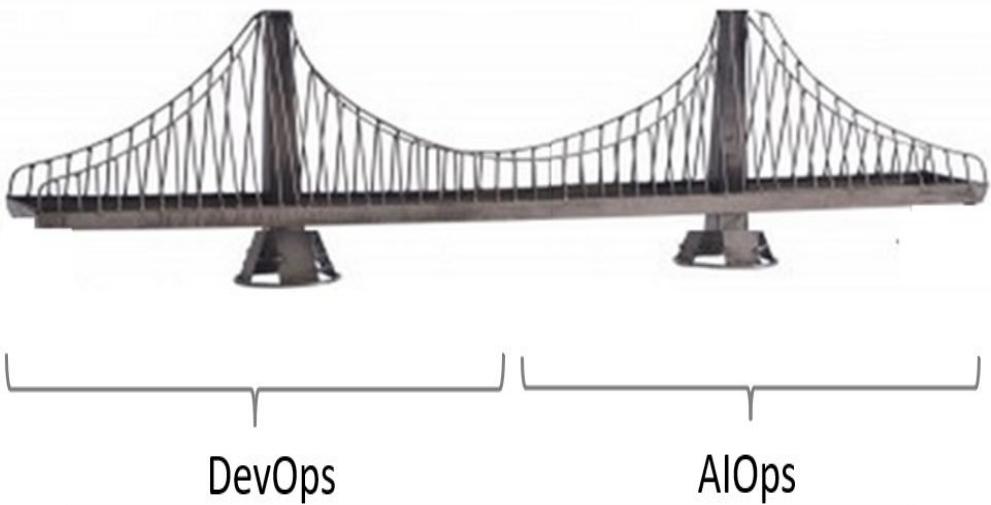


Promove:

- Plataformas de Tecnologias multi camadas que automatizam e melhoram as Operações de TI
- Eliminação de erros humanos e economia de tempo
- Componentes chave incluindo big data e machine learning

Ações:

- Detecta automaticamente e reage a problemas em tempo real e de forma inteligente automatizando devops e cloudops
- incluiu a utilização de insumos de ferramentas de monitoramento existentes, aplicação de técnicas de algoritmos, análise e produção de uma saída que é um item de ação com insight para a equipe de operação



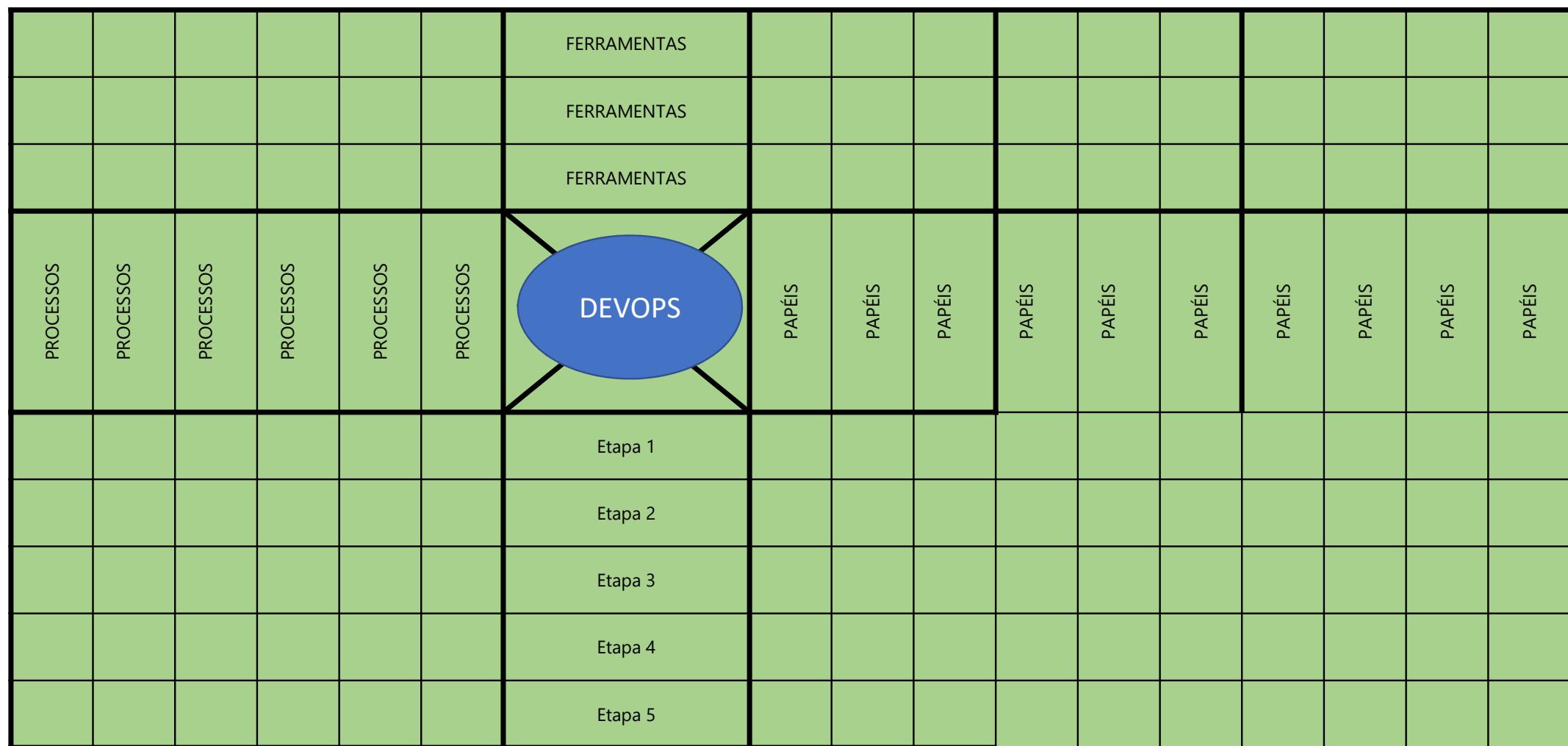
Plataformas AIOps utilizam bigdata, aprendizagem de máquinas moderna e outras tecnologias avançadas de análise para direta ou indiretamente melhorar as operações de TI (monitoração, automação e service desk) funcionando proativamente, com insights dinâmicos e personalizados.

Gartner

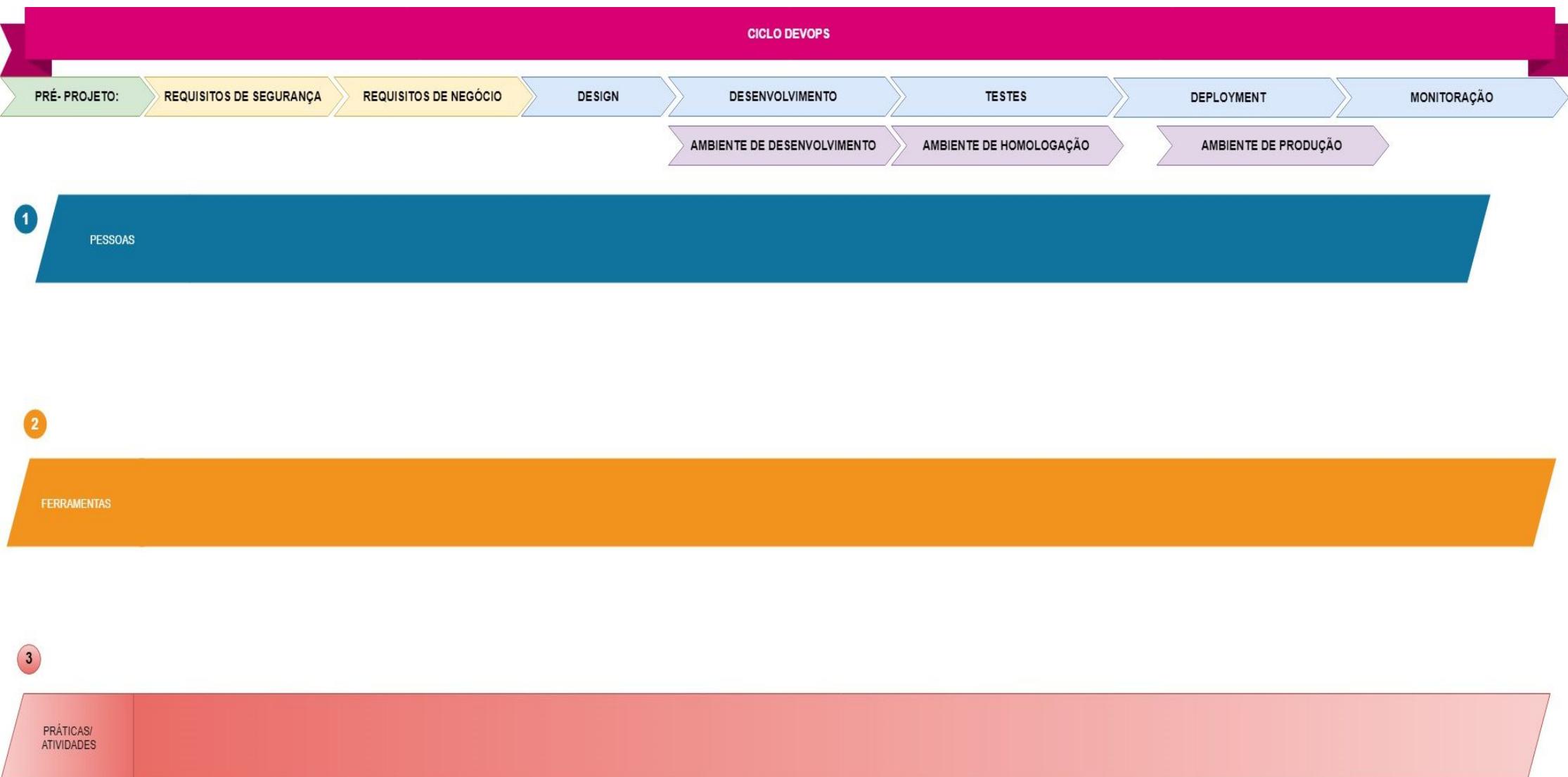


16 - Perspectivas



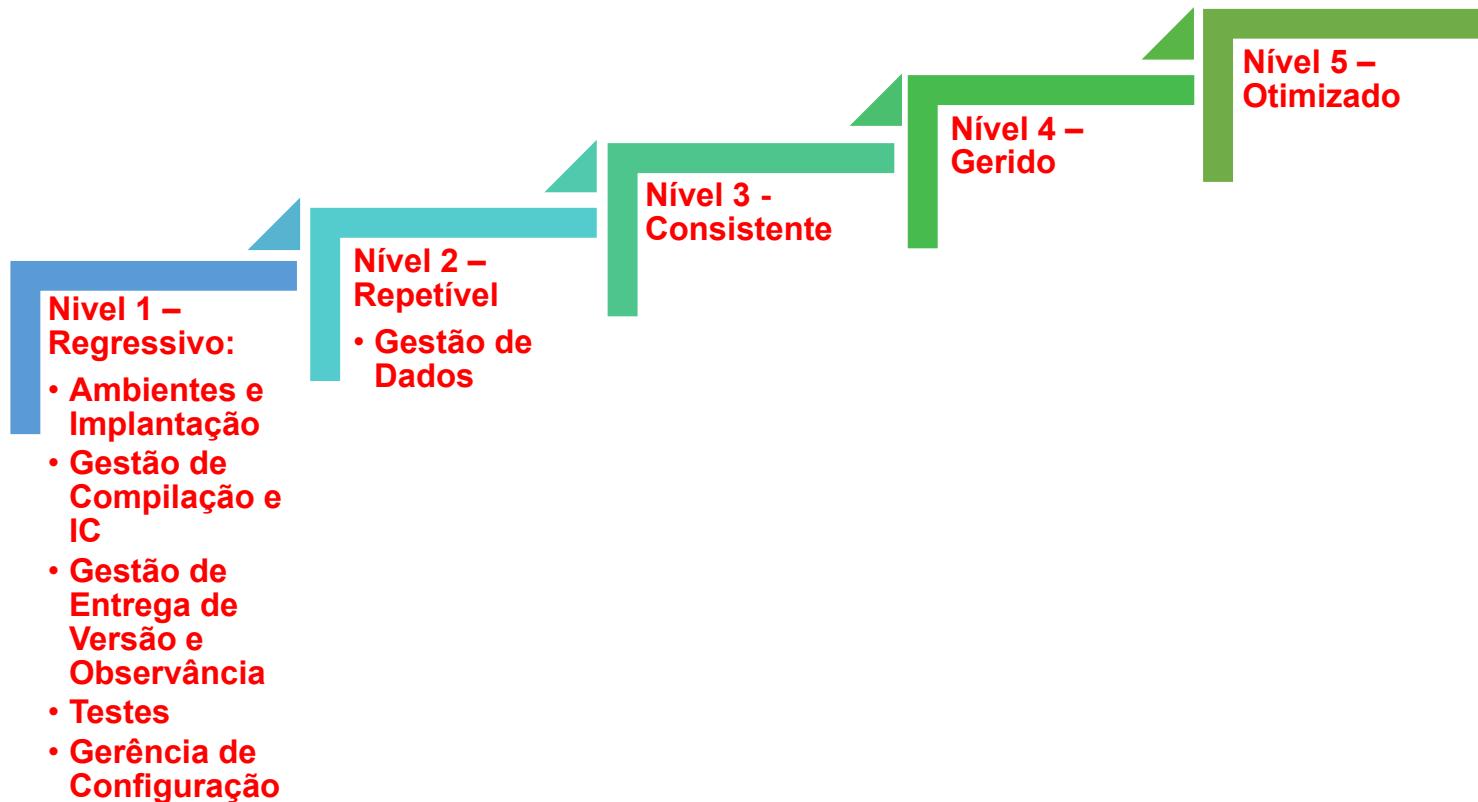


16 - Resolução do Problema com DevOps: Ciclo de vida e fases



Cenário 1

Problema: A Compilação demora muito e processo de implantação é frágil



Cenário 1:

Sintomas	Causas
1 - Os testadores demoram muito para encerrar os relatórios de defeitos	Não há hardware suficiente disponível
2 - Os clientes demoram muito para testar ou confirmar que histórias estão completas	O processo de implantação não é automatizado
3 - Os testadores encontram problemas que os desenvolvedores corrigiram há muito tempo	Os desenvolvedores não estão sendo disciplinados o suficiente em manter a aplicação funcional por meio de pequenas mudanças incrementais e frequentemente causam problemas em funcionalidades existentes
4 - Ninguém confia nos ambientes de UAT, de desempenho ou de IC, e as pessoas demonstram ceticismo em relação a quando uma nova entrega estará disponível	1 - A configuração de hardware e do sistema operacional não é gerenciada corretamente 2 - O processo de implantação depende de sistemas que estão fora do controle da equipe
5 - Demonstrações acontecem raramente	Não há pessoas suficientes que entendem dos processos de compilação e implantação
6 - A aplicação raramente é demonstrada de maneira funcional	Os desenvolvedores não estão sendo disciplinados o suficiente em manter a aplicação funcional por meio de pequenas mudanças incrementais e frequentemente causam problemas em funcionalidades existentes
7 - A velocidade da equipe (taxa de progresso) é menor que a esperada.	Testadores, desenvolvedores, analistas e equipe de operações não estão colaborando o suficiente durante o desenvolvimento

Material hand's on SUGESTÃO

<https://www.qwiklabs.com/quests/37>

<https://www.qwiklabs.com/focuses/269?locale=en&parent=catalog>

<https://www.qwiklabs.com/quests/51>

<https://docs.docker.com/install/linux/docker-ce/ubuntu/>

<https://docs.docker.com/docker-for-windows/>

<https://docs.docker.com/compose/overview/>

<https://devopsinstitute.com/certifications/continuous-delivery-architecture-cda/>

<https://cloudsecurityalliance.org/artifacts/>

Referências:

- Exin White Paper
- MORAIS, Gleicon - "CAIXA DE FERRAMENTAS DEVOPS – Casa do Código, 2017 São Paulo, SP.
- https://www.ibm.com/developerworks/community/blogs/a9ba1efe-b731-4317-9724-a181d6155e3a/entry/accelerating_maximo_development_with_continuous_delivery?lang=en
- <https://www.gp4us.com.br/backlog-do-produto/>
- <http://www.rightstar.com/the-devops-handbook-part-1/>
- <http://www.datacenterdynamics.com.br/focus/archive/2015/02/empresas-brasileiras-faturam-29-mais-ao-adotar-devops-diz-pesquisa-da-ca-techn>
- <https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/beyond-agile-reorganizing-it-for-faster-software-delivery>
- https://devops.com/integrating-itil-change-management-and-devops/?__hstc=82446857.f17651f71ffeced04f16650f088009bd.1524442511418.1524442511418.1524442511418.1&__hssc=82446857.1.1524442511420&__hsfp=3548929705
- <https://gaea.com.br/tudo-que-voce-precisa-saber-sobre-integracao-itil-e-devops/>
- <https://novacontext.com/azure-strategy-and-implementation/>
- <http://www.forumdaconstrucao.com.br/conteudo.php?a=0&Cod=1860>
- <https://www.mandic.com.br/artigos/devops-significado-do-termo/>
- <http://www.franklinfitch.com/blog/2018/04/24/devsecops-making-sec-sexy/>
- <https://www.slideshare.net/AmazonWebServices/local-testing-and-deployment-best-practices-for-serverless-applications-aws-online-tech-talks-83591383>
- <https://www.itpedia.nl/pt/2017/07/04/devops-architecture-monitoring/>
- <https://hostadvice.com/blog/devops-toolbox-jenkins-ansible-chef-puppet-vagrant-saltstack/>
- https://www.google.com.br/imgres?imgurl=https%3A%2F%2Fwww.owasp.org%2Fimages%2Fb%2Fb5%2FDevOps_AppSec_Tool_Integration.png&imgrefurl=https%3A%2F%2Fwww.owasp.org%2Findex.php%2FOWASP_AppSec_Pipeline&docid=eszrd-74tR3LXM&tbnid=c_dRo0IKqmDLPM%3A&vet=10ahUKEwj6ufGq_qXaAhVFvZAKHTzDDrYQMwhdKBcwFw..i&w=1051&h=525&client=firefox-b-ab&bih=626&bih=1366&q=devops%20X%20CI%2FCD&ved=0ahUKEwj6ufGq_qXaAhVFvZAKHTzDDrYQMwhdKBcwFw&iact=mrc&uact=8#h=525&imgdii=pFHJ8zTrDGZULM:&vet=10ahUKEwj6ufGq_qXaAhVFvZAKHTzDDrYQMwhdKBcwFw..i&w=1051
- <https://www.besthouse.site/agile-and-waterfall-combined/>
- <https://br.pinterest.com/pin/671106781944196356/>
- <https://br.pinterest.com/pin/671106781944196356/>
- <https://br.pinterest.com/pin/440367669805394832/>
- <https://xebialabs.com/devops-diagram-generator>
- https://pt.wikipedia.org/wiki/Test_Driven_Development
- <https://theleanapps.com/category/devops/>
- <http://www.visualworkplaceinc.com/continuous-improvement-resources/kpi-boards/>
- <https://denpeakacademy.com/2015/12/21/the-kanban-board/>
- <https://felipelucioquirino.wordpress.com/2013/02/19/estrutura-de-um-documento-de-arquitetura/>
- https://www.sei.cmu.edu/research-capabilities/all-work/display.cfm?customel_datapageid_4050=21328
- https://pt.wikipedia.org/wiki/Arquitetura_de_software
- <https://www.boozallen.com/s/insight/blog/how-to-avoid-the-devsecops-technology-trap.html>
- <https://aws.amazon.com/pt/blogsopensource/microservices-on-aws-using-containers-serverless/>
- https://www.citrix.com/blogs/wp-content/uploads/2016/05/syn210blog1_1.png
- <https://www.slideshare.net/welkaim/from-vm-to-container-to-serverless>
- <https://www.quora.com/What-is-the-advantage-of-Microservices-architecture-over-Service-Oriented-Architecture-SOA>
- https://pt.wikipedia.org/wiki/Service-oriented_architecture

