DevOps

Aula 1

Eidil Santos
Eidil.santos@faculdadeimpacta.com.br

O que é DevOps

Surgiu a partir da comunidade de desenvolvimento;

- Visavam suprir necessidades do Agile;

Visa unir equipes de Dev e Ops;

O que é DevOps

Agile x DevOps
2001: Manifesto
2003: SRE (Google)
2005: Git, Puppet
2009: Velocity (DevOps days)
2012: grandes empresas (CA, HP, IBM)
2013: Project Phoenix (livro)

O que é DevOps

Agile Manifesto
Indivíduos e interações mais que processos e ferramentas
Software em funcionamento mais que documentação abrangente
Colaboração com o cliente mais que negociação de contratos
Responder a mudanças mais que seguir um plano

O que é DevOps

Lean
Valor (na perspectiva do cliente)
Fluxo de valor (etapas que agregam valor ao produto)
Fluxo contínuo (fluxo sem interrupções para agregar valor)
Produção puxada (fazer apenas o solicitado)
Perfeição (melhoria contínua – kaizen - de processos, pessoas e produtos)

O que é DevOps

Pilares do DevOps (CAMS)

Cultura (Culture)

- Colaboração;
- Melhor comunicação entre times de desenvolvimento e operações;
- Adequação ao Agile;

O que é DevOps

Pilares do DevOps (CAMS)

Automação (Automation)

- Ferramentas (CI e CD);
- Pipelines (conceito do Lean);
- Evitar interação humana nas ações da pipeline;

O que é DevOps

Pilares do DevOps (CAMS)

Monitoração (Monitoring)

- Métricas;
- Logs;
- Ambientes;

O que é DevOps

Pilares do DevOps (CAMS)

- Compatilhamento (Sharing)
  - Feedback;
  - Compartilhar responsabilidades (p. ex. Infra-as-a-code);

Por que “DevOps”

Velocidade;
Inovação;
Agilidade;

Continuous Integration

Identificar erros mais rapidamente;
Acelerar correções;
Melhorar a qualidade (shift-left);
Automação do processo de testes;

Continuous Delivery

Deploy a qualquer momento;
Automação;
Padronização e disponibilidade;

Pipeline

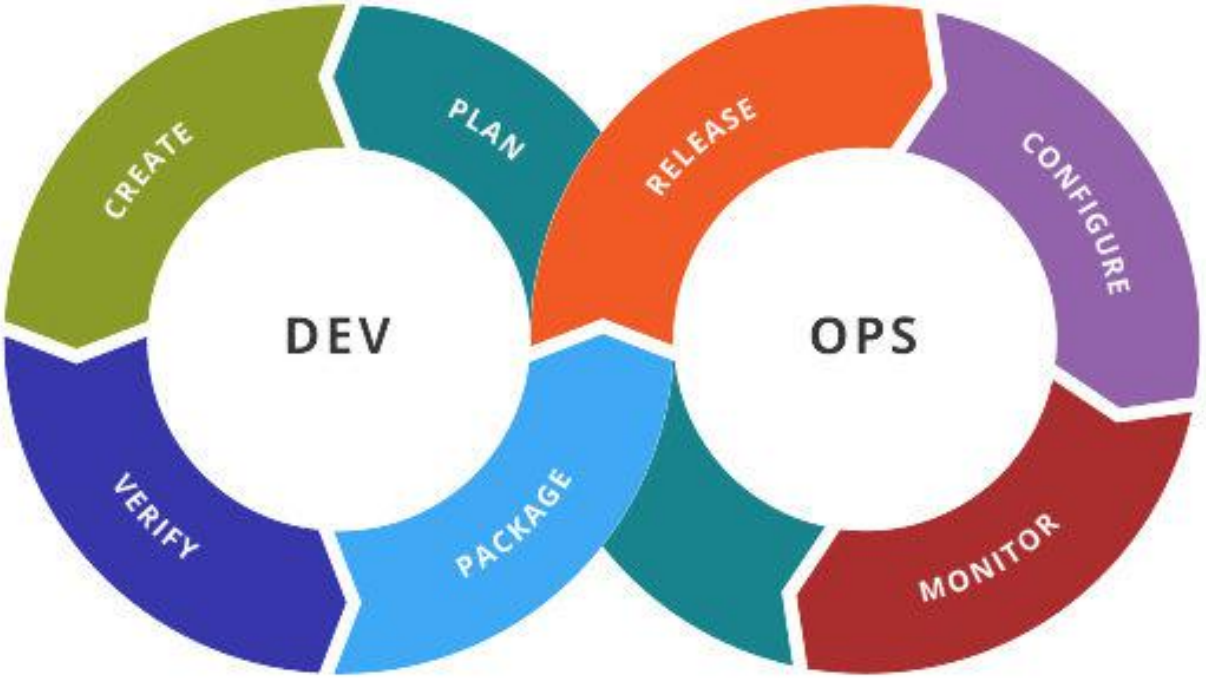
Ferramentas que compoem um pipeline:

- Repositório de código;
- Ferramentas de testes;
- Resolução de dependências;
- Ferramentas de CI;
- Ferramentas de CD;
- Análise estática de código;
- Análise dinâmica de código;
- Repositório de artefatos;

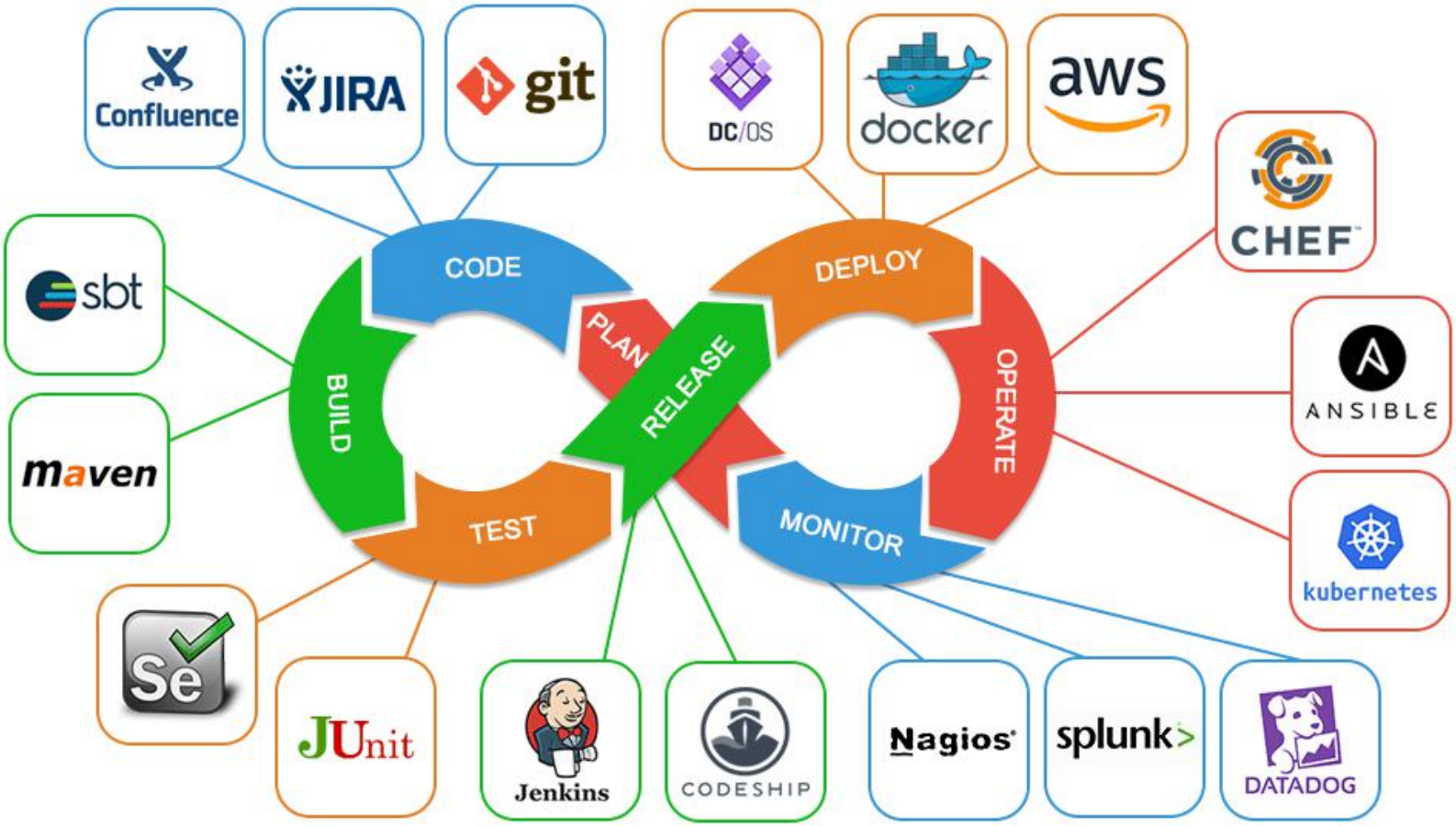
‘Plataformas DevOps’:

- GitLab CI;
- Atlassian;
- Azure DevOps;

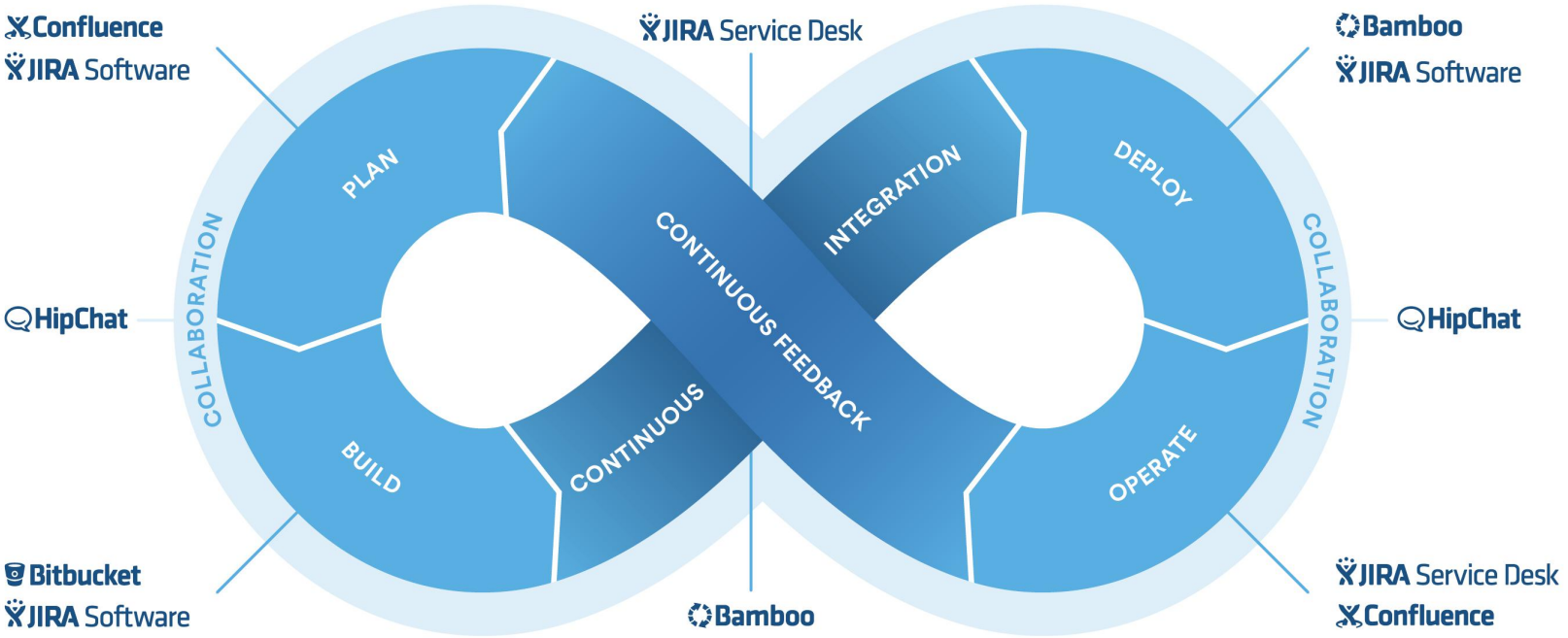
Pipeline


CREATE
PLAN
RELEASE
CONFIGURE
DEV
OPS
VERIFY
PACKAGE
MONITOR

Pipeline

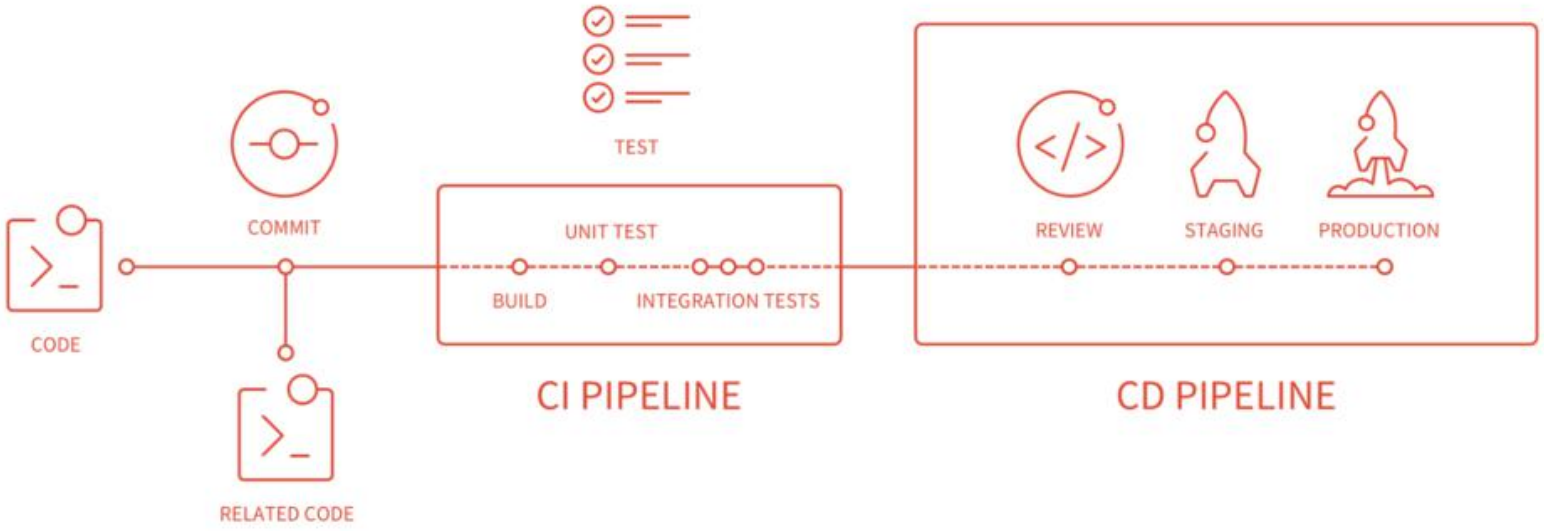

Confluence
JIRA
git
DC/OS
docker
aws
CHEF
sbt
maven
CODE
PLAN
BUILD
TEST
RELEASE
DEPLOY
OPERATE
MONITOR
ANSIBLE
kubernetes
Se
JUnit
Jenkins
CODESHIP
Nagios
splunk>
DATADOG

Pipeline (Atlassian)



Pipeline (GitLab CI)

GitLab CI



Pipeline (Azure)

Start using Azure DevOps

Build GitHub projects using Azure Pipelines

Set up continuous integration and continuous delivery (CI/CD) for your GitHub repository.

Start using Azure DevOps

Sign up and get started using Azure DevOps Services.

What's new

Learn about new features under development and recently released.

Web portal navigation

Learn how to work effectively within the web portal.

Azure Pipelines

Manage CI/CD to deploy your code with high-performance pipelines that work with any language, platform, and cloud.

Azure Repos

Use Git repositories, pull requests, and then integrate with CI/CD to build and deploy your apps.

Azure Boards

Plan and track your work using interactive, highly-customizable backlogs and boards.

Azure Artifacts

Share code with others across your teams or company, and support CI/CD of your apps.

Azure Test Plans

Improve the overall code quality of your apps by using manual, exploratory, or load-based testing services.

Settings

Configure resources and settings for users, teams, projects, and organizations.

Create pipelines to build and deploy applications to any platform, cloud, or app store.

Pipeline (Open)

- GitLab (repositório de código);
- SonarQube (análise de código);
- Nexus (repositório de artefatos;
- Maven (build);
- Jenkins (CI e CD);

Na prática

Problemas que cada tipo de ferramenta deve resolver:

- Repositório de código: integridade do código, acesso/distribuição;
- Ferramentas de testes: automatizar a qualidade;
- Resolução de dependências: padronizar;
- Ferramentas de CI: automatizar a geração de pacotes;
- Ferramentas de CD: automatizar a entrega;
- Análise estática de código: avaliar o código antes de ‘buildado’;
- Análise dinâmica de código: analisar código depois de ‘buildado’;
- Repositório de artefatos: disponibilizar rapidamente qualquer pacote, auditoria;

Na prática

Por quê:

- Docker;
- Microsserviços;
- Cloud;

Como seria um Pipeline ideal

Iniciado a partir de um commit em determinada branch;
Avaliação de cobertura de código durante o build;
Testes automatizados;
Avaliação de segurança antes e depois da geração do pacote/entregável;
Deploy em ambientes de homologação e produção automatizados;
Monitoramento automatizado;