AULA 5: DevOps: conceitos básicos

Professora: **Rosana T. Vaccare Braga**

Material elaborado com base no Capítulo 10 do livro online: Engenharia de Software Moderna - Princípios e Práticas para Desenvolvimento de Software com Produtividade, autor: Marco Tulio Valente, disponível em: https://engsoftmoderna.info/

Motivação

- **Desenvolvimento tradicional adotado pela maioria das empresas:**
  - **Departamento de Sistemas (ou Desenvolvimento): desenvolvedores, programadores, analistas, arquitetos, etc.**
  - **Departamento de Suporte (ou Operações): administradores de rede, administradores de bancos de dados, técnicos de suporte, técnicos de infraestrutura, etc.**
- **Problemas?**

Motivação

- **Problemas?**
  - **urgência em implantar um novo sistema (pelos ops)**
  - **desconhecimento das especificidades da plataforma utilizada no desenvolvimento (pelos ops) ou**
  - **desconhecimento das especificidades da plataforma utilizada na produção (pelos devps)**
  - **Consequências:**
    - falta de hardware para executar o novo sistema ou a nova funcionalidade
    - problemas de desempenho
    - incompatibilidades com o banco de dados de produção
    - vulnerabilidades de segurança
    - Resultado: atraso ou cancelamento da implantação ou abandono do sistema

DevOps

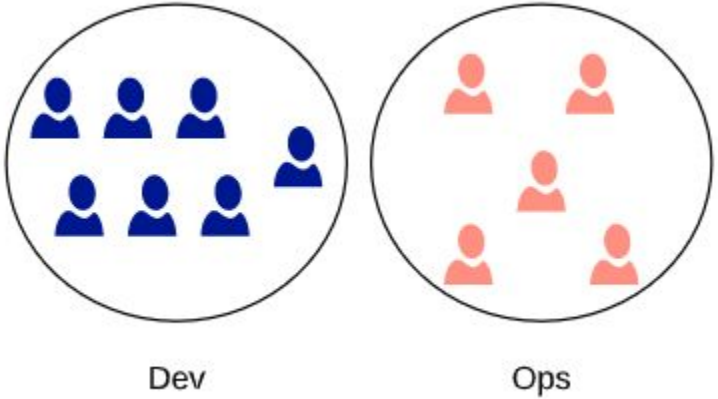
- **Termo recente (+- 2009)**
  - **Movimento que visa unificar as culturas de desenvolvimento (Dev) e de operação (Ops) para permitir a implantação mais rápida e ágil de um sistema**
  - **Reúne desenvolvedores, TI e organizações para criar e implantar aplicativos de maneira contínua para fornecer software de qualidade.**

DevOps

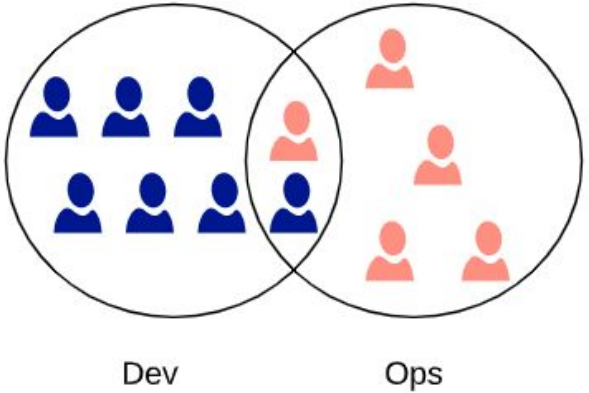
- **DevOps reúne desenvolvedores e operadores.**
  - **Essas duas pessoas têm pontos de vista diferentes no ciclo de desenvolvimento do projeto.**
  - **Todas as partes interessadas envolvidas no ciclo de vida de entrega de software são obrigadas a se comunicar e colaborar melhor e com frequência.**
  - Curiosidade: Patrick Debois cunhou o termo “DevOps” em 2009. Ele é um consultor belga, praticante ágil e gerente de projetos, que se tornou um dos primeiros líderes DevOps e formou esta palavra combinando “Dev” como em desenvolvimento, e “Ops” como em operações.

DevOps

- **Não advoga a criação de um profissional novo**
  - **Aproximação entre o pessoal de desenvolvimento e o pessoal de operações e vice-versa**



Organização que não é baseada em DevOps, pois existe pouca comunicação entre Dev e Ops.



Organização baseada em DevOps. Frequentemente, Devs e Ops sentam juntos para discutir questões sobre a entrega do sistema.

Princípios de DevOps

Baseado nos princípios para entrega de software do livro: “*Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*, de Humble e Farley, 2010.

1. Crie um processo repetível e confiável para entrega de software: colocar um software em produção deve ser tão simples como apertar um botão.
2. Automatize tudo que for possível (pré-requisito do princípio anterior):
    a. todos os passos para entrega de um software devem ser automáticos, incluindo seu build, a execução dos testes, a configuração e ativação dos servidores e da rede, a carga do banco de dados, etc.
3. Mantenha tudo em um sistema de controle de versões:
    a. não apenas a todo o código fonte, mas também arquivos e scripts de administração do sistema, documentação, páginas Web, arquivos de dados, etc.
    b. deve ser simples restaurar e voltar o sistema para um estado anterior.

Princípios de DevOps

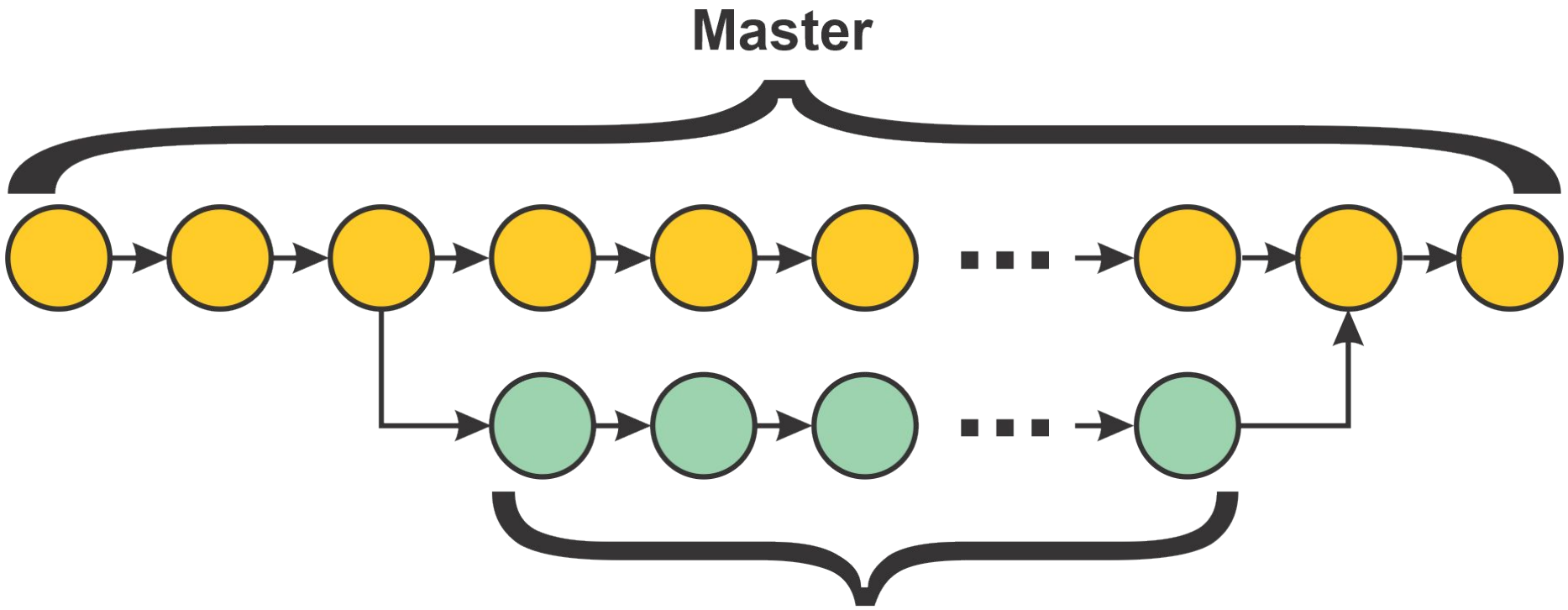
Baseado nos princípios para entrega de software do livro: “*Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*, de Humble e Farley, 2010.

4. Se um passo causa dor, execute-o com mais frequência e o quanto antes:
    a. antecipar os problemas antes que eles se acumulem e as soluções fiquem complicadas (**Integração Contínua**)
5. Concluído significa pronto para entrega:
    a. Não pode haver pendências como: a implementação ainda não foi testada com dados reais, ela ainda não foi documentada, ela ainda não foi integrada com o sistema X, etc.
6. Todos são responsáveis pela entrega do software:
    a. não admite-se mais que os times de desenvolvimento e de operações trabalhem em silos independentes e troquem informações apenas na véspera de uma implantação.

Integração Contínua

- **Problema que levou à proposta**
  - Branches ou mais especificamente Branches de funcionalidades (*feature branches*)
    - podem levar meses para se integrar à linha principal (main)
  - Merge
    - pode ocorrer conflitos de integração ou conflitos de merge

Integração Contínua



Integração Contínua

- **Problemas causados por conflitos**
  - **Integration hell ou merge hell**
    - **Para pensar: conseguem dar exemplos de alterações feitas numa branch que afetem a main, ou vice versa?**
- **Cada nova funcionalidade passa a ter um dono**
  - **silos de conhecimento**
  - **adoção de padrões próprios**

Integração Contínua

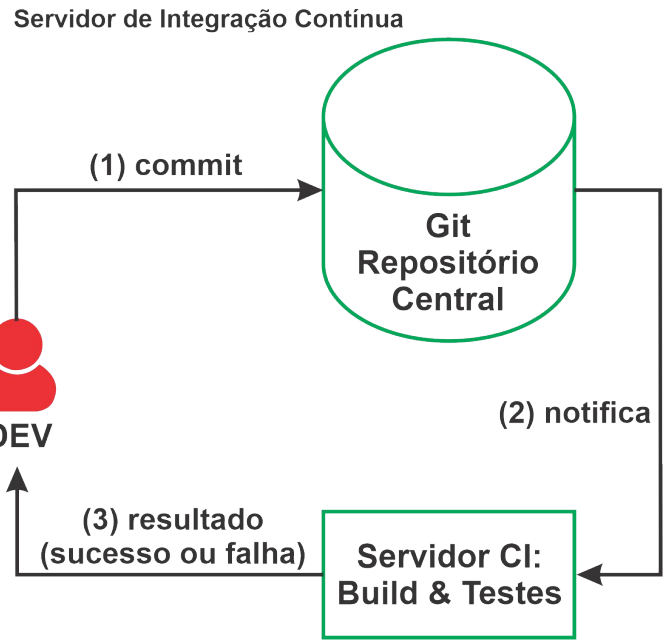
- **O que é Integração Contínua (Continuous Integration ou CI)?**
  - **Prática de desenvolvimento proposta por Extreme Programming (XP)**
    - **“Se uma tarefa causa dor, não podemos deixar que ela acumule”... devemos quebrá-la em subtarefas**
  - **CI recomenda integrar o código de forma frequente**

Boas Práticas para Uso de CI

- **Build Automatizado**
  - Build: compilação de todos os arquivos de um sistema para gerar uma versão executável.
  - Automatizado: sem nenhum passo manual
- **Testes Automatizados**
  - garantir que o sistema continua com o comportamento esperado após o novo build, executando testes automaticamente
- **Servidores de Integração Contínua**
  - responsável por executar o build automatizado e os testes sempre que houver mudanças

Servidores de Integração Contínua

- **Builds e testes automatizados devem ser executados com frequência**
- **Se possível após cada novo commit realizado no master**



Integração Contínua

- **Evitar a integração de código com problemas**
  - Quando o build falha, costuma-se dizer que ele "quebrou"
  - Alguns motivos: o desenvolvedor esqueceu de realizar o commit de algum arquivo, dependências incorretas (biblioteca de versão anterior), etc…
- **Parar tudo o que está fazendo e providenciar a correção (ou reverter o código para a versão anterior)**
- **Somente avançar para próxima tarefa de programação após receber o resultado do servidor de CI**

Integração Contínua

- **Diversos servidores de integração contínua no mercado (o próprio Github é um deles)**
- **Uma dúvida comum é se CI é compatível com o uso de branches.**
  - Vamos discutir: o que acham?

Integração Contínua

- **CI é compatível com o uso de branches?**
  - Branches devem durar no máximo um dia de trabalho
  - Será que vale a pena criá-las?
  - Mas como fazemos para adotar CI se o código ainda não está pronto?
    - voltaremos nesse assunto daqui a pouco!

Desenvolvimento Baseado no Trunk

- **Quando migram para CI, é comum que as organizações usem também desenvolvimento baseado no trunk (Trunk Based Development ou TBD)**
- **Com TBD, não existem mais branches para implementação de novas funcionalidades ou para correção de bugs**
- **Exemplos: Google e Facebook**

Deployment Contínuo

- **Com integração contínua, código novo é frequentemente integrado no branch principal**
  - **Esse código não precisa estar pronto para entrar em produção (veremos como daqui a pouco!)**
- **Existe mais um passo da cadeia de automação proposta por DevOps**
  - **Deployment Contínuo (Continuous Deployment ou CD)**

Deployment Contínuo

- **Diferença entre CI e CD**
- **Fluxo de trabalho quando se usa CD:**
  - **O desenvolvedor desenvolve e testa na sua máquina local**
  - **Ele realiza um commit e o servidor de CI executa novamente um build e os testes de unidade**
  - **Algumas vezes no dia, o servidor de CI realiza testes mais exaustivos com os novos commits que ainda não entraram em produção**
  - **Se todos os testes passarem, os commits entram imediatamente em produção → CD !**

Vantagens do Deployment Contínuo

- **redução no tempo de entrega de novas funcionalidades, que são liberadas assim que ficam prontas, diminuindo o intervalo entre releases e o número de releases.**
- **CD torna novas entregas um não-evento (não existe mais um dia D ou um deadline para entrega de novas releases) e evita que a perda de um deadline atrase a entrega de uma funcionalidade por meses.**
- **melhora a motivação dos desenvolvedores, que rapidamente recebem retorno — vindo de usuários reais — sobre o sucesso ou não de suas tarefas.**
- **favorece a experimentação e um estilo de desenvolvimento orientado por dados e feedback dos usuários.**

Entrega Contínua (Continuous Delivery)

- **Deployment Contínuo (CD) não é recomendável para certos tipos de sistemas**
  - desktop, app móvel ou embarcada que precisa de instalação para ser atualizada.
- **Entrega Contínua (EC)**
  - Todo commit pode entrar em produção imediatamente
  - Gerente de projetos ou de releases, por exemplo, toma a decisão sobre quando os commits, de fato, serão liberados para os usuários finais

Entrega Contínua (Continuous Delivery)

- **Diferença entre CD e EC**
  - **Deployment** é o processo de liberar uma nova versão de um sistema para seus usuários.
  - **Delivery** é o processo de liberar uma nova versão de um sistema para ser objeto de deployment.
  - Quando adota-se Deployment Contínuo (CD), ambos os processos são automáticos e contínuos. Porém, com Entrega Contínua (EC), a entrega é realizada com frequência, mas o deployment depende de uma autorização manual.

Feature Flags

- ***Se novas releases são liberadas quase todo dia, como evitar que minhas implementações parciais, que ainda não foram devidamente testadas ou que têm problemas de desempenho, cheguem até os usuários finais?***

Feature Flags

- **Nem sempre todo commit estará pronto para entrar imediatamente em produção**
  - **Integration (ou merge) hell**
  - **Solução para esse problema:**
    - Variável booleana (um flag)

Solução: integre continuamente o código parcial da funcionalidade X, mas com ela desabilitada, isto é, qualquer código relativo a X estará guardado por uma variável booleana (um flag) que, enquanto a implementação de X não estiver concluída, vai ser falsa

```
featureX = false;

...
if (featureX)
        “aqui tem código incompleto de X”

...
if (featureX)
        “mais código incompleto de X”
```

Feature Flags ou Feature Toggles

- **Suponha que você esteja trabalhando em uma nova página de um certo sistema**
- **Esse é o código que vai para produção enquanto a nova página não estiver pronta**

```
nova_pag = false;

...
if (nova_pag)
        “carregue nova página”

else
        “carregue página antiga”
```

Feature Flags ou Feature Toggles

- **A duplicação de código entre as duas páginas pode ser temporária, ou seja, removida quando a nova página estiver pronta**
- **Exemplo: estudo feito no navegador Chrome em 2016: 2400 flags; uma nova release adicionava 73 novos flags e removia 43 flags.**

```
AMBIENTE DE DESENVOLVIMENTO
nova_pag = true;

...
if (nova_pag)
        “carregue nova página”

else
        “carregue página antiga”
```

```
AMBIENTE DE PRODUÇÃO
nova_pag = false;

...
if (nova_pag)
        “carregue nova página”

else
        “carregue página antiga”
```

Feature Flags

- **Release canário:**
  - nova funcionalidade, guardada por um feature flag, é disponibilizada inicialmente para um grupo pequeno de usuários
- **Testes A/B**
  - libera-se simultaneamente duas versões de uma funcionalidade (antiga e nova, por exemplo) para grupos distintos de usuários, com o objetivo de verificar se a nova funcionalidade de fato agrega valor ao sistema

Feature Flags

- **Como implementar features canário ou testes A/B:**
  - usar uma estrutura de dados para armazenar as flags ou
  - Bibliotecas dedicadas a gerenciar feature flags: flags podem ser setados externamente ao código, por exemplo, em um arquivo de configuração

```
FeatureFlagsTable fft = new FeatureFlagsTable ( );
fft.addFeature(“novo-carrinho-compras”, false);
...
if (fft.IsEnabled(“novo-carrinho-compras”))
        // processa compra usando novo carrinho

else
        // processa compra usando carrinho atual
...
```

Próxima aula

Ferramentas DevOps