DevOps

UNIVERSIDADE DO VALE DO SAPUCAÍ – UNIVAS – POUSO ALEGRE/MG
PROFESSOR: RAFFAEL CARVALHO

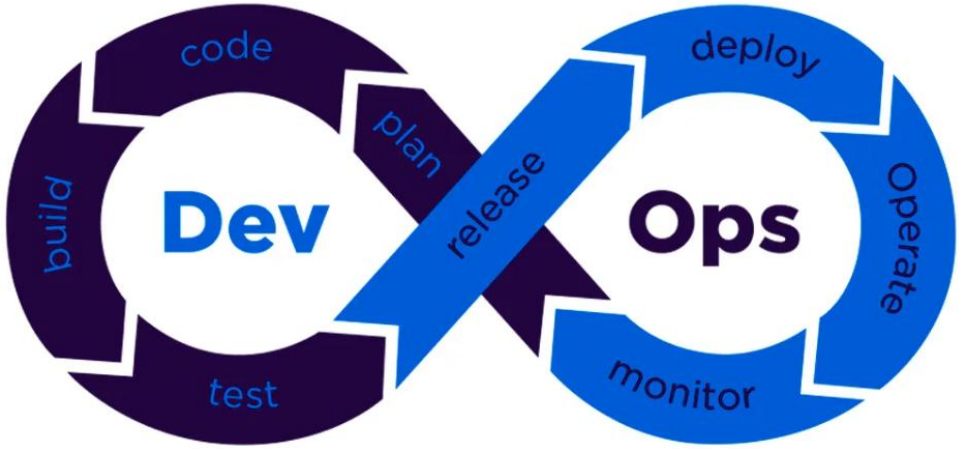
Antes do DevOps: Muros e Desafios

Pontos:

- **Dev (Desenvolvimento):** Foco em inovação, novas funcionalidades, velocidade nas entregas.
- **Ops (Operações):** Foco em estabilidade, segurança, infraestrutura, evitar falhas.
- **O Conflito:** Metas diferentes, prioridades opostas. Lentidão na entrega, "jogar a culpa", retrabalho.

A União que Transforma

DevOps é a união de cultura, práticas e ferramentas para aumentar a capacidade de uma organização de entregar aplicações e serviços em alta velocidade: evoluindo e melhorando produtos em um ritmo mais rápido do que as organizações que usam abordagens tradicionais de desenvolvimento de software e gerenciamento de infraestrutura.



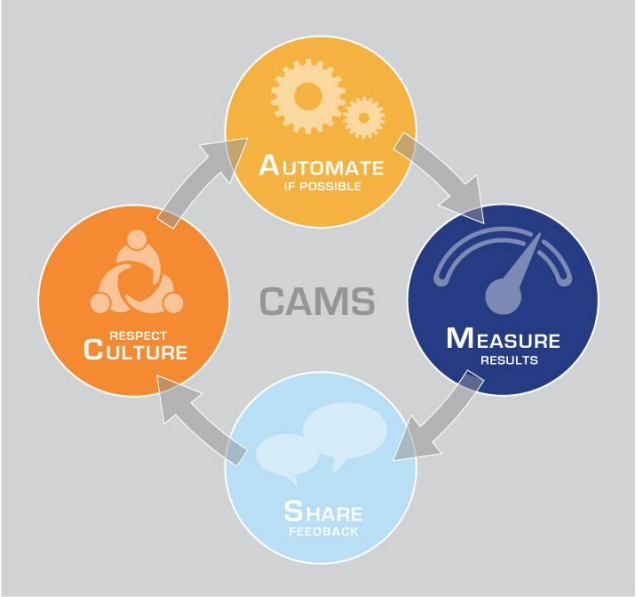
Os Pilares do DevOps: C.A.M.S

Cultura (Culture): Foco na colaboração, responsabilidade compartilhada e confiança entre equipes.

Automação (Automation): Automatizar tarefas repetitivas em todo o ciclo de vida (testes, deploy, infra).

Medição (Measurement): Coletar métricas, monitorar o desempenho e a saúde do sistema. "O que não é medido, não é gerenciado.

Compartilhamento (Sharing): Compartilhar conhecimento, feedback, sucessos e falhas entre as equipes. Aprender continuamente.



Benefícios do DevOps

Maior Velocidade de Entrega: Lançamentos mais frequentes e rápidos.

Maior Confiabilidade: Menos falhas, recuperações mais rápidas.

Melhor Colaboração e Comunicação: Equipes mais felizes e produtivas.

Melhoria Contínua: Ciclos de feedback que impulsionam a inovação.

Satisfação do Cliente: Entregar valor mais rápido e com mais qualidade.

Controle de Versão e Colaboração

Caos e Conflito: Trabalhar sem Controle de Versão

- **Cenário de Pesadelo:**
  - Arquivos chamados codigo_final.js, codigo_final_2.js, codigo_final_de_verdade.js.
  - **Perda de Trabalho:** Alguém apaga o código de outra pessoa.
  - **Conflitos na Fusão:** Mesclar o código é manual, demorado e cheio de erros.
  - **Impossibilidade de Reverter:** Se algo quebrar, é impossível voltar para a versão estável.
- **Consequência:** Lentidão, frustração e medo de fazer mudanças.

A Solução: Sistema de Controle de Versão (VCS)

- **Definição:** Um sistema que registra todas as mudanças em arquivos ao longo do tempo, permitindo que você recupere versões específicas mais tarde.
  - **Benefícios Principais:**
  - **Histórico Completo:** Saiba quem mudou o quê e quando.
  - **Rastreabilidade:** Volte a qualquer versão anterior com segurança.
  - **Trabalho Paralelo:** Múltiplas pessoas podem trabalhar no mesmo projeto sem se atrapalharem.
- **A Ferramenta Dominante:** Git é o padrão de mercado.

Como Colaboramos: O Ciclo de 3 Passos

- **Passos e Conceitos:**
  - **Clone:** Baixar uma cópia do repositório central (git clone).
  - **Commit:** Salvar uma mudança localmente (git commit). O registro oficial da sua alteração.
  - **Push:** Enviar o seu código para o repositório central (o servidor) (git push).
  - **Pull:** Baixar as mudanças feitas por outros colegas (git pull).
- **Conceito-Chave:** O repositório remoto é o ponto central de colaboração.

Branching: Trabalhando em Ramificações Isoladas

- **Conceito:** Uma branch (ramo) é uma linha de desenvolvimento isolada do código principal (geralmente chamada de main ou master).
- **Para que Serve:**
  - Permite que você trabalhe em uma nova funcionalidade (ou correção de bug) sem quebrar o código que está em produção.
  - É a base para as Merge Requests / Pull Requests.
- **Fluxo Típico:**
  1. Cria uma nova branch.
  2. Faz as mudanças e testes.
  3. Abre um Pull Request (PR) para a revisão do código.
  4. Merge: Mescla o código de volta à branch principal.

O Coração da Colaboração: Pull Requests (PRs)

- **Definição:** Um PR (em GitHub/GitLab/Bitbucket) é um pedido para mesclar seu código (sua branch) na branch principal.
- **Função no DevOps:**
  - **Revisão de Código:** Outros membros da equipe (ou especialistas) revisam o código antes que ele seja aceito.
  - **Gatilho do CI/CD:** Quando um PR é aberto, ele geralmente dispara automaticamente o pipeline de Integração Contínua (CI) para rodar testes.
  - **Discussão e Feedback:** É o local para discutir as mudanças e melhorias.

Conectando os Pontos: VCS e CI/CD

- O Controle de Versão (VCS), através do Git, é o Gatilho para o CI/CD.
- **Ação:** Um git push ou a abertura de um Pull Request.
- **Resultado:** O pipeline de CI é ativado, garantindo que o novo código funcione e possa ser integrado.
- Sem Git, não há CI/CD eficiente.

Exercício em grupo

1. Pré-Requisitos

Ferramenta Central: Utilizar uma plataforma de hospedagem de Git (GitHub, GitLab, etc.).

Repositório Base: Um membro do grupo deve criar um repositório com um único arquivo, por exemplo, apresentacoes.md.

Exercício em grupo

Cada membro do grupo deve adicionar um pequeno bloco de apresentação pessoal ao final do arquivo apresentacoes.md.

Instruções para todos os membros do grupo:

1. **Pull Inicial:** Faça um git pull no repositório principal (main).
2. **Branch Isolada:** Crie uma *branch* com o seu primeiro nome.
3. **Implementação:** Abra o arquivo apresentacoes.md e adicione a sua seção no **final** do arquivo
4. **Commit e Push:** Faça o commit das suas mudanças (com uma mensagem clara) e o push para a sua branch remota.

Exercício em grupo

Pull Request: *Cada membro do grupo deve ir para a interface da plataforma (GitHub/GitLab) e abrir um Pull Request (PR) da sua branch para a branch principal (main).*

Merge: *O membro que criou o repositório deve ser o único a mesclar (merge) os PRs. Como todos estão adicionando conteúdo no final do arquivo, não haverá conflito de código, e os merges serão rápidos e bem-sucedidos.*

Visualização: *Todos devem fazer um último git pull na branch main para verem que o arquivo apresentacoes.md agora contém o trabalho de todos os colegas.*

Bibliografia Básica

- PRESSMAN, Roger S. Engenharia de Software: uma abordagem profissional.8. ed. São Paulo: McCraw Hill – Artmedia, 2016.
- BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. UML
- Guia do Usuário. Rio de Janeiro: Campus, 2012.
- JOHNSON, Ralph; VLISSIDES, John; HELM, Richard; GAMMA, Erich. Padrões de Projeto. São Paulo: Bookman, 2008.