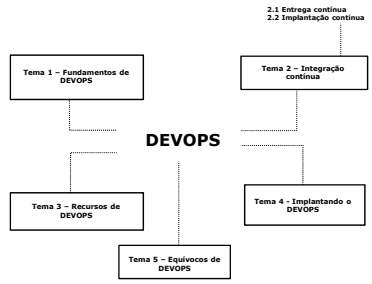
Aula 6

Engenharia de Software

Prof. Alex Mateus Porn

Conversa Inicial



Fundamentos de DevOps

DevOps

- **Conforme Muniz et al. (2020)**
  - **Desenvolvimento (Dev): equipe responsável pela identificação dos requisitos com o cliente, pela análise, projeto, pela codificação e pelos testes**
  - **Operações (Ops): equipe responsável pela implantação em produção, pelo monitoramento e pela solução de incidentes e problemas**

- **DevOps é uma cultura fortemente colaborativa entre as equipes de desenvolvimento e operações, para entregar o software funcionando em produção de forma ágil, segura e estável. Mais do que um conceito, é importante destacar que DevOps é uma jornada de aproximação entre as pessoas, com ações práticas de automação para acelerar as implantações com qualidade, considerando o ponto de vista de todos os envolvidos, a tão falada empatia (Muniz et al., 2020)**

Pilares da cultura DevOps

- Segundo Davis e Daniels (2016)
  - Colaboração: construir resultados específicos com interações de pessoas com diferentes experiências e um propósito comum
  - Afinidade: construção de relações interdependentes entre os times interfuncionais para aprendizagem contínua

- Ferramentas: funcionam como um acelerador, impulsionando a mudança com base na cultura atual
- Escala: o dimensionamento leva em conta como os outros três pilares podem ser aplicados à medida que as organizações crescem, amadurecem e até encolhem, considerando questões técnicas e culturais

- Conforme Soares e Silveira (2020), o DevOps influencia o ciclo de vida do aplicativo em todas as fases
  1. Planejamento
  2. Desenvolvimento
  3. Entrega
  4. Operação

1. Planejamento
   - Idealizam, definem e descrevem os recursos e funcionalidades dos aplicativos e sistemas que estão construindo
     - Criar lista de pendências, acompanhar bugs, gerenciar o desenvolvimento de software ágil com Scrum, usar quadros Kanban e visualizar o progresso com dashboards

2. Desenvolvimento
   - Inclui todos os aspectos da codificação: gravação, teste, revisão e integração do código
     - As equipes DevOps buscam inovar rapidamente sem sacrificar a qualidade, a estabilidade e a produtividade

3. Entrega
   - É o processo de implantação de aplicativos nos ambientes de produção, de maneira consistente e confiável. Também inclui a implantação e a configuração da infraestrutura fundamental
     - As equipes definem um processo de gerenciamento de versão com estágios claros de aprovação manual

4. Operação

- Envolve manter, monitorar e solucionar problemas de aplicativos em ambientes de produção
  - As equipes trabalham para garantir a confiabilidade do sistema, a alta disponibilidade e o objetivo de tempo de inatividade igual a zero, reforçando a segurança e a governança

Princípios DevOps

- Desenvolver e testar sistemas semelhantes à produção
  - Sistemas semelhantes à produção
- Implantar com processos confiáveis e repetíveis
  - Suportar um processo de desenvolvimento de software ágil do início ao fim

- Monitorar e validar a qualidade operacional
  - Monitorar o início do ciclo de vida
  - Teste automatizado no início e frequentemente no ciclo de vida
- Amplificar os loops de feedback
  - Permitir que as organizações reajam e façam mudanças mais rapidamente. Na entrega, requer um feedback rápido e aprendizagem rápida com cada ação

Integração contínua

- A integração contínua é uma prática do desenvolvimento de software em que cada participante do time integra seu trabalho pelo menos uma vez no dia. Cada integração é verificada por um build automatizado para detectar erros de imediato e permitir que as atividades de desenvolvimento de software tenham mais qualidade e agilidade (Fowler, 2006)

Princípios da integração contínua

- Conforme Fowler (2006)
  - Manter um repositório de origem único
  - Automatizar a versão
  - Fazer o autoteste de construção (testes unitários)
  - Todos devem fazer commit pelo menos uma vez por dia
  - Todo commit deve ser centralizado em uma máquina de integração

- Corrigir quebra de código imediatamente
- Manter a construção rápida
- Testar em um ambiente que seja o mais próximo possível do ambiente de produção
- Garantir que qualquer um possa obter o executável mais recente
- Permitir que todo mundo possa ver o que está acontecendo
- Automatizar a implantação

- Já para Humble e Farley (2014), a integração contínua é praticada com as seguintes ações:
  - O time integra o código pelo menos uma vez por dia em um único tronco
  - A segmentação de implementação é iniciada automaticamente a cada mudança de código e são executadas validações, análise estática de padrões de codificação e testes
  - Quando sua versão falha, na maioria das vezes é corrigida em até dez minutos

Exemplificando

- Implementar uma pequena função de um sistema
- O primeiro passo consiste em obter uma cópia do código-fonte atual
  - Exemplo: obter uma cópia do GitHub, conforme vimos em aulas anteriores
- O próximo passo consiste no desenvolvimento e testes da nova função

- Finalizada a etapa anterior, é necessário criar uma nova versão automatizada (build)
  - Essa etapa somente é finalizada se não forem encontrados erros em nenhum dos testes
- Com a nova versão pronta, o próximo passo consiste em adicioná-la ao repositório do GitHub
  - Se houver mudanças feitas por outros desenvolvedores, primeiro é necessário atualizar a versão e criar um novo build

- Com o novo build criado e sincronizado com a versão principal do software, o próximo passo consiste em realizar o commit e atualizar o repositório com a versão mais atual do sistema

Entrega contínua

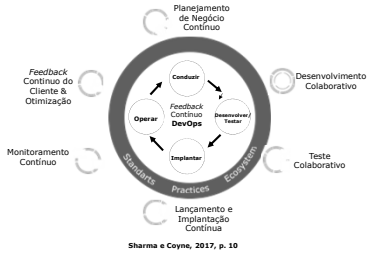
- É uma evolução natural quando existe o interesse de expandir os benefícios da automação dos testes e feedback imediato para os próximos estágios (Muniz et al., 2020)
- É a capacidade de disponibilizar mudanças de forma segura e rápida, garantindo que o código esteja sempre pronto para implantação, mesmo diante de milhares de desenvolvedores fazendo alterações diariamente (Humble; Farley, 2014)

Implantação contínua

- Conforme Muniz et al. (2020)
  - É a evolução natural da entrega e consiste na implantação automática em produção após a execução dos testes e das validações previstas
  - A ideia é disponibilizar pequenas mudanças em produção com a maior frequência possível
  - É também possível replicar os ambientes de forma confiável

Recursos do DevOps

Arquitetura de referência DevOps



Sharma e Coyne, 2017, p. 10

Caminhos DevOps a seguir

- Conduzir: estabelecer os objetivos do negócio e ajustá-los com base no feedback dos clientes
- Desenvolver/testar: envolve duas práticas
  - Desenvolvimento colaborativo: permite que os profissionais trabalhem juntos, fornecendo um conjunto comum de práticas e plataforma
  - Teste contínuo: testar mais cedo e continuamente, resultando em ciclos de teste mais curtos e feedback contínuo

- Implantar: abrange a prática da liberação e implantação contínuas
  - Liberação e implantação contínuas: lançar novos recursos para clientes e usuários o mais rápido possível

- Operar: inclui duas práticas que permitem às empresas monitorar o desempenho dos aplicativos em produção e receber feedback
  - Monitoramento contínuo: fornece dados e métricas para o pessoal de operações, controle de qualidade e desenvolvimento
  - Feedback e otimização contínua do cliente: permite que diferentes partes interessadas tomem as ações apropriadas para melhorar os aplicativos e a experiência do cliente

Implantando o DevOps

Implantando o DevOps

- Embora o nome DevOps sugira recursos baseados em desenvolvimento e operações, trata-se de um recurso corporativo que abrange todas as partes interessadas em uma organização, incluindo proprietários de negócios, arquitetura, design, desenvolvimento, garantia de qualidade, operações, segurança, parceiros e fornecedores. A exclusão de qualquer parte interessada – interna ou externa – leva a uma implementação incompleta do DevOps (Sharma; Coyne, 2017)

Fontes de ineficiência na entrega do software

- Segundo Sharma e Coyne (2017)
  - Sobrecarga desnecessária: necessidade de comunicar várias vezes a mesma informação e conhecimento
  - Retrabalho desnecessário: defeitos são descobertos em testes ou produção, forçando a equipe a voltar ao desenvolvimento
  - Superprodução: funcionalidades desenvolvidas que não foram requeridas

Cultura DevOps

- Construir uma cultura não é como adotar um processo ou uma ferramenta. Requer engenharia social de equipes de pessoas, cada uma com predisposições, experiências e preconceitos únicos. Essa diversidade pode tornar a construção de uma cultura desafiadora e difícil (Sharma; Coyne, 2017)

Principais técnicas

- Conforme Sharma e Coyne (2017)
  - Melhoria contínua
    - Foco no que se deseja melhorar
    - Métricas para medir o desempenho
    - Padronização
  - Planejamento de liberação

- Integração contínua
- Entrega contínua
- Teste contínuo
- Monitoramento e feedback contínuos

Equívocos DevOps

Equívocos DevOps

- Conforme Davis e Daniels (2016), DevOps:
  - É apenas para desenvolvedores e administradores de sistemas
  - É um time
  - É uma função
  - É relevante apenas para startups web
  - Necessita de uma certificação
  - Significa fazer todo o trabalho com metade da equipe

Equívocos DevOps

- Conforme Davis e Daniels (2016)
  - Existe um jeito certo e um errado de aplicar o DevOps
  - São necessárias X semanas para implantá-lo
  - DevOps:
    - É todo o conjunto de ferramentas
    - É sobre automação
    - É passageiro