DevOps

Aula 01 – Parte 1

Senac

Ponto de Reflexão

Como funciona a criação de um site de e-commerce e de que maneira as pessoas conseguem colaborar e concluir suas tarefas mesmo trabalhando remotamente?

Experiências anteriores

Você já teve experiência ou conhece alguma metodologia de trabalho colaborativo?

Você já utilizou alguma ferramenta de software para trabalho colaborativo?

DevOps

Introdução

O QUE É DEVOPS

DevOps é uma metodologia que combina desenvolvimento de software (Dev) e operações de TI (Ops) em um único fluxo de trabalho contínuo.

Promove a colaboração entre equipes, automação de processos e entrega rápida de software de alta qualidade

O QUE É DEVOPS

DEVOPS é um modo de pensar ou trabalhar

DevOps é uma cultura.

ATIVIDADE - O QUE É DEVOPS?

Realize uma pesquisa na Internet e identifique os principais motivos para a adoção da cultura DevOps

Referências

- DevOps – Wikipédia, a enciclopédia livre (https://pt.wikipedia.org/wiki/DevOps)
- Conheça os benefícios do DevOps para a programação (https://www.hostgator.com.br/blog/conheca-os-beneficios-do-devops/)
- 5 motivos para usar a cultura DevOps sempre! – Blog Central Server (https://blog.centralserver.com.br/5-motivos-para-usar-cultura-devops-sempre/)

FINALIDADE DO DEVOPS

Acelera o ciclo de vida do desenvolvimento de software

⚙️ Automação:

Redução de tarefas manuais e repetitivas, como compilação, testes e implantação.

🤝 Colaboração:

Integração entre equipes de desenvolvimento e operações, eliminando silos e promovendo responsabilidades compartilhadas.

🚀 Entrega Contínua:

Lançamento frequente de pequenas atualizações, reduzindo riscos e melhorando a qualidade do software.

RELAÇÃO COM O CENÁRIO ATUAL DE TI

DevOps se tornou uma abordagem crucial.

Adaptação às Mudanças:
permite que as empresas lancem novos recursos rapidamente.

Integração com Novas Tecnologias:
se alinha com tendências como IA, IoT e computação em nuvem.

Segurança e Sustentabilidade:
DevSecOps integra a segurança desde o início do ciclo de desenvolvimento.

Cultura Organizacional:
promove uma cultura de colaboração e aprendizado contínuo, essencial para equipes multidisciplinares em ambientes de TI modernos.

ETAPAS DO DEVOPS

1. Planejamento e Colaboração;
2. Desenvolvimento;
3. Integração Contínua (CI);
4. Entrega Contínua (CD);
5. Monitoramento e Feedback;
6. Implantação em Produção.

PLANEJAMENTO E COLABORAÇÃO

Objetivo: Alinhar as equipes, definir objetivos e planejar as entregas.

Ferramentas: Jira, Trello, Azure DevOps.

Prática:

- ❑ A equipe se reúne para planejar o próximo sprint.
- ❑ Utilizam um quadro Kanban no Jira para visualizar as tarefas
- ❑ Definem métricas de sucesso, como tempo de entrega e taxa de falhas.
- ❑ Compartilham documentação para garantir que todos tenham acesso às informações necessárias.

DESENVOLVIMENTO

Objetivo: Escrever código de forma eficiente, seguindo boas práticas e integrando feedbacks.

Ferramentas: Git, GitHub, GitLab, VS Code.

Prática:

- ❑ Um desenvolvedor trabalha em uma nova funcionalidade
- ❑ Cria uma branch no Git para isolar suas alterações
- ❑ Segue práticas de código limpo e realiza revisões de código com colegas via pull requests no GitHub.
- ❑ Escreve testes unitários para garantir que o código funcione conforme o esperado.

INTEGRAÇÃO CONTÍNUA (CI)

Objetivo: Integrar o código frequentemente e detectar problemas cedo.

Ferramentas: GitLab CI e GitHub

Prática:

- ❑ O desenvolvedor faz um push da branch
- ❑ O pipeline de CI é acionado automaticamente:
  1. Build: Compila o código e verifica erros de sintaxe.
  2. Testes: Executa testes unitários e de integração.
  3. Análise de Código: Verifica a qualidade do código
- ❑ Se tudo passar, o código é mesclado na branch principal (`main`).

ENTREGA CONTÍNUA (CD)

Objetivo: Preparar o código para ser liberado em produção de forma automatizada.

Ferramentas: GitHub e GitLab.

Prática:

- ❑ Após a integração na branch `main`, o pipeline de CD é acionado:
  1. Build do Artefato: Gera uma imagem Docker da aplicação.
  2. Testes de Aceitação: Executa testes automatizados em um ambiente de staging.
  3. Deploy em Staging: Implanta a aplicação no ambiente de staging para validação manual.
- ❑ O código é marcado como pronto para produção.

INTEGRAÇÃO CONTÍNUA (CI)

Objetivo: Integrar o código frequentemente e detectar problemas cedo.

Ferramentas: GitLab CI e GitHub

Prática:

- ❑ O desenvolvedor faz um push da branch
- ❑ O pipeline de CI é acionado automaticamente:
  1. Build: Compila o código e verifica erros de sintaxe.
  2. Testes: Executa testes unitários e de integração.
  3. Análise de Código: Verifica a qualidade do código
- ❑ Se tudo passar, o código é mesclado na branch principal (`main`).

MONITORAMENTO E FEEDBACK

Objetivo: Coletar dados sobre o desempenho da aplicação e obter feedback para melhorias..

Ferramentas: Prometheus, Grafana, ELK Stack, New Reli.

Prática:

- ❑ A aplicação em produção é monitorada em tempo real:
  1. Métricas: Uso de CPU, memória, tempo de resposta.
  2. Logs: Erros e eventos são coletados e analisados no ELK Stack.
  3. Alertas: Configuração de alertas no Prometheus para notificar a equipe em caso de problemas.
- ❑ Feedback dos usuários é coletado via ferramentas como surveys.
- ❑ A equipe analisa os dados e planeja melhorias para o próximo ciclo.

IMPLANTAÇÃO EM PRODUÇÃO

Objetivo: Liberar o software para os usuários finais de forma segura e controlada

Ferramentas: Kubernetes, Terraform, Ansible

Prática:

- O código validado no staging é implantado em produção:
  - Rollout Gradual: estratégia de canary deployment - versão é liberada para uma pequena porcentagem de usuários.
  - Verificação: Monitora-se o comportamento da aplicação.
  - Rollback Automático: Se problemas forem detectados, o sistema reverte automaticamente para a versão anterior.
- Após confirmação de estabilidade, a nova versão é liberada

DevOps

DevOps

Aula 01 – Parte 1

Senac