Atividade

CI/CD e Git

1. O que é CI/CD e qual é o seu objetivo?

CI/CD significa Integração Contínua e Entrega Contínua. O objetivo é automatizar testes e deploy, facilitando entregas rápidas e seguras de software..

2. Quais são os benefícios do CI/CD?

- Redução de erros em produção
- Feedback rápido sobre falhas
- Entregas frequentes e seguras
- Automatização de testes e builds
- Menor tempo de entrega
- Mais confiança no processo de liberação

3. Quais são as diferenças entre integração contínua (CI) e entrega contínua (CD)?

CI automatiza o teste do código sempre que algo é alterado. CD automatiza a entrega dessas alterações para ambientes de produção ou homologação.

4. Como o CI/CD pode melhorar a qualidade do software?

- Identifica bugs mais cedo com testes automatizados
- Evita acúmulo de problemas no código
- Garante consistência entre os ambientes
- Melhora a cobertura de testes
- Facilita a verificação de padrões de código

5. Como o CI/CD pode melhorar a colaboração entre os desenvolvedores?

- Permite commits pequenos e frequentes
- Reduz conflitos de código
- Feedback mais rápido para todos
- Facilita revisões e integração de código
- Ajuda a manter um fluxo de trabalho organizado

6. Como o controle de versão está relacionado ao CI/CD?

O Git aciona os pipelines de CI/CD sempre que um commit ou push acontece, garantindo rastreabilidade e controle do processo.

7. Quais ferramentas e tecnologias são comumente usadas em uma implementação de CI/CD?

- Servidores CI/CD: Jenkins, GitHub Actions, GitLab CI, CircleCI, Travis CI
- Containers: Docker, Kubernetes
- Controle de versão: Git
- Automação: Ansible, Terraform
- Testes: JUnit, PyTest, Selenium
- Monitoramento: Prometheus, Grafana

8. O que é o Git e por que é usado em desenvolvimento de software?

Git é um sistema de controle de versão distribuído. Ele ajuda a rastrear mudanças, trabalhar em equipe e manter o histórico do projeto

9. Como o Git difere de outros sistemas de controle de versão?

Git é distribuído, funciona offline, é rápido, permite múltiplos fluxos de trabalho e tem ótima gestão de branches.

10. O que é um repositório do Git?

É onde o projeto e seu histórico ficam salvos. Pode estar no seu computador ou em um servidor como o GitHub.

11. Quais são os principais comandos do Git que são usados com mais frequência?

- `git init`: inicia um repositório
- `git clone`: clona um repositório
- `git add`: adiciona arquivos à área de stage
- `git commit`: salva mudanças no repositório
- `git push`: envia mudanças para o repositório remoto
- `git pull`: baixa atualizações do repositório remoto
- `git branch`: lista ou cria branches
- `git merge`: une branches

12. O que é um commit no Git?

É um registro de mudanças feitas no código, com uma mensagem explicando o que foi alterado.

13. O que é um branch no Git e como ele é usado?

É uma linha de desenvolvimento separada. Serve pra desenvolver algo novo sem mexer no código principal.

14. O que é um merge no Git e como é realizado?

É quando você junta um branch com outro. Normalmente, se une uma feature com o branch principal.

15. O que é um conflito de merge no Git e como ele é resolvido?

Acontece quando dois branches mudam a mesma parte de um arquivo. Você precisa editar manualmente e decidir qual versão manter.

16. O que é o GitHub e como ele se relaciona com o Git?

GitHub é uma plataforma online que usa Git para hospedar repositórios e facilitar a colaboração entre devs.

17. Quais são algumas melhores práticas para trabalhar com o Git em equipes de desenvolvimento?

Commits frequentes com mensagens claras, uso de branches, pull requests para revisar, e manter o main estável com CI/CD rodando.