**QUESTÕES:**

1. **Quais são os principais benefícios da prática de Integração Contínua?​**
2. **Explique como a prática de Entrega Contínua se diferencia da Integração Contínua.​**
3. **Quais são os componentes-chave de um processo de Entrega Contínua?​**
4. **Como a Gerência de Configuração de Software contribui para a qualidade e consistência do software?​**
5. **Quais são as vantagens de utilizar um sistema de controle de versão distribuído como o Git em comparação com um sistema centralizado como o Subversion?**
6. **Qual é a importância da automação na implementação de práticas de Integração Contínua e Entrega Contínua?​**
7. **Quais são as melhores práticas para gerenciar dependências de software em um ambiente de desenvolvimento ágil?​**
8. **Como as baselines de configuração são usadas no contexto da Gerência de Configuração de Software?​**
9. **Quais são os principais desafios enfrentados ao implementar práticas de Integração Contínua e Entrega Contínua em equipes de desenvolvimento?​**
10. **Por que a rastreabilidade é um aspecto importante da Gerência de Configuração de Software, e como ela é alcançada na prática?**
11. **Qual é a principal diferença entre Integração Contínua e Entrega Contínua?​**
12. **Por que a prática de Integração Contínua é considerada fundamental no desenvolvimento de software?​**
13. **Quais são os benefícios de realizar testes automatizados no contexto de Integração Contínua?​**
14. **Explique a importância da implantação contínua no contexto de Entrega Contínua.​**
15. **Como a Gerência de Configuração de Software contribui para a integridade e rastreabilidade do código-fonte?​**
16. **Quais são os objetivos da Gerência de Configuração de Software?**
17. **Como a automação de builds está relacionada à prática de Integração Contínua?​**
18. **Cite pelo menos três ferramentas comumente utilizadas para controle de versão no desenvolvimento de software.**

**​**

1. **Qual é a importância de manter baselines de configuração em projetos de software?​**
2. **Como a Gerência de Configuração de Software pode ajudar na identificação e correção de problemas em diferentes versões do software?**

**RESOLUÇÃO:**

1. ***Integração Contínua (IC)****: Os principais benefícios da IC incluem detecção precoce de problemas de integração, redução de tempo de retrabalho e aumento da confiança no software produzido.*
2. ***Entrega Contínua vs Integração Contínua****: A Entrega Contínua estende a IC ao garantir que o software possa ser lançado a qualquer momento. Enquanto a IC se concentra na integração e teste de código, a Entrega Contínua se concentra na entrega de versões de software prontas para produção.*
3. ***Componentes-chave da Entrega Contínua****: Incluem integração contínua, ambientes de teste automatizados, implantação automatizada e monitoramento de aplicativos.*
4. ***Gerência de Configuração de Software (GCS)****: A GCS contribui para a qualidade e consistência do software, garantindo que todas as mudanças sejam rastreadas, gerenciadas e aplicadas de maneira controlada.*
5. ***Git vs Subversion****: O Git, um sistema de controle de versão distribuído, permite que os desenvolvedores trabalhem offline, suporta fluxos de trabalho não lineares e oferece melhor desempenho e segurança em comparação com o Subversion, um sistema centralizado.*
6. ***Automação na IC e Entrega Contínua****: A automação é crucial para a implementação eficaz de práticas de IC e Entrega Contínua, pois permite a execução rápida e confiável de tarefas repetitivas, como compilação, teste e implantação de software.*
7. ***Gerenciamento de dependências em desenvolvimento ágil****: As melhores práticas incluem o uso de ferramentas de gerenciamento de dependências, a verificação regular de atualizações de dependências e a garantia de que as dependências sejam compatíveis entre si.*
8. ***Baselines de configuração na GCS****: As baselines de configuração são usadas para definir um ponto no desenvolvimento do software que serve como referência para futuras atividades de desenvolvimento e manutenção.*
9. ***Desafios na implementação de IC e Entrega Contínua****: Os desafios incluem a necessidade de uma cultura de colaboração, a necessidade de testes automatizados abrangentes e a necessidade de infraestrutura adequada.*
10. ***Rastreabilidade na GCS****: A rastreabilidade é importante para entender as relações e dependências entre diferentes elementos de configuração. É alcançada mantendo registros detalhados de todas as mudanças.*
11. ***Diferença entre IC e Entrega Contínua****: A principal diferença é que a IC se concentra na integração e teste de código, enquanto a Entrega Contínua se estende a isso para garantir que o software possa ser lançado a qualquer momento.*
12. ***Importância da IC no desenvolvimento de software****: A IC é considerada fundamental porque ajuda a detectar e corrigir problemas mais cedo, reduz o tempo de retrabalho e aumenta a confiança no software produzido.*
13. ***Testes automatizados na IC****: Os benefícios incluem a detecção precoce de defeitos, a redução do esforço manual e a capacidade de executar testes com frequência.*
14. ***Implantação contínua na Entrega Contínua****: A implantação contínua é importante porque permite que as organizações entreguem novos recursos e melhorias aos usuários mais rapidamente e com menos risco.*
15. ***GCS e integridade do código-fonte****: A GCS contribui para a integridade e rastreabilidade do código-fonte, garantindo que todas as mudanças sejam rastreadas e aplicadas de maneira controlada.*
16. ***Objetivos da GCS****: Os objetivos da GCS incluem o controle de mudanças, o gerenciamento de versões e a garantia da integridade e rastreabilidade do software.*
17. ***Automação de builds e IC****: A automação de builds é uma parte fundamental da IC, pois permite a compilação rápida e confiável do software sempre que as mudanças são integradas.*
18. ***Ferramentas de controle de versão****: Algumas ferramentas comumente utilizadas para controle de versão incluem Git, Subversion e Mercurial.*
19. ***Importância das baselines de configuração****: As baselines de configuração são importantes porque fornecem um ponto de referência estável para o desenvolvimento e manutenção do software.*
20. ***GCS e identificação de problemas****: A GCS pode ajudar na identificação e correção de problemas, permitindo que os desenvolvedores rastreiem e revertam as mudanças que causaram problemas.*