Gerencia de configuração:

Resolva e justifique as respostas:

1. O gerenciamento das configurações de servidores é o processo pelo qual se pode rastrear, atualizar e manter configurações relacionadas a versões de *software*, segurança e rede para que o sistema funcione em uma linha de base predeterminada e permaneça seguro independentemente de quaisquer alterações.

Gerenciamento de configurações de servidores é o processo de rastreamento, atualização e manutenção das configurações de servidores relacionadas a software, segurança e rede para garantir que o sistema opere de acordo com uma linha de base predeterminada. Isso inclui garantir que o sistema permaneça seguro, mesmo diante de mudanças, aplicando práticas como controle de versões, automação de configurações e monitoramento contínuo para detectar e corrigir variações ou vulnerabilidades. Esse gerenciamento é essencial para manter a integridade, desempenho e segurança dos servidores ao longo do tempo.

2. No que se refere aos conceitos de gestão de configuração, julgue o item subsequente.
O comando *get clone* incorpora as alterações de um repositório remoto no ramo atual.

O comando mencionado está incorreto. O comando correto para incorporar as alterações de um repositório remoto no ramo

atual é git pull, não get clone. O comando git pull busca e integra as atualizações do repositório remoto no branch atual. Por outro lado, git clone é usado para criar uma cópia local de um repositório remoto, não para incorporar alterações em um branch existente. Portanto, o item está errado.

3. Assinale a opção em que é corretamente indicado o comando Git que permite armazenar as alterações feitas nos arquivos sem realizar o commit e que pode ser útil quando se precisa trocar de branch ou mesmo trabalhar numa tarefa diferente, mas não se deseja confirmar as alterações, ainda.
   a) Log
   b) Stash
   c) Rebase
   d) Bissect
   e) Restore

4. O objetivo principal da criação de uma *branch* em um sistema de controle de versão como o Git é:
   a) sincronizar automaticamente as alterações com um servidor remoto.
   b) comprimir os arquivos do repositório para economizar espaço em disco.
   c) reverter as alterações feitas em um arquivo.

d) facilitar o trabalho colaborativo, permitindo que várias pessoas trabalhem em diferentes funcionalidades simultaneamente.

e) excluir permanentemente um arquivo do repositório.

5. Julgue o item a seguir, a respeito de conceitos, prática e ferramentas relativos a DevOps e de integração contínua.

   Uma das boas práticas do DevOps é a adoção de uma cultura livre de culpa por erros nos processos apresentados pelos desenvolvedores ou pelo pessoal de operações.

   Está correto a afirmação porque uma das boas práticas do DevOps é, de fato, a adoção de uma cultura livre de culpa ("blameless culture"

6. Assinale a opção que apresenta o comando utilizado no Git para *versionar* o projeto com um pacote de alterações.
   a) Add
   b) Checkout
   c) Commit
   d) Clone
   e) Branch

7. No Git, o comando que envia as atualizações do repositório local para o repositório remoto é executado por meio da instrução
   a) Git push
   b) Git commit
   c) Git pull
   d) Git add
   e) Git merge

8. Um *dev* que trabalha com integração contínua, para garantir que suas implementações funcionem com o restante do código, deve, sequencialmente, ao final de sua tarefa,
   a) atualizar a cópia local do projeto, executar os testes localmente, executar um *build* local e fazer *commit* com o repositório central.
   b) atualizar a cópia local do projeto, executar um *build* local, executar os testes localmente e fazer *commit* com o repositório central.
   c) fazer *commit* com o repositório central, executar um *build* local, atualizar a cópia local do projeto e executar os testes localmente.
   d) executar um *build* local, atualizar a cópia local do projeto, executar os testes localmente e fazer *commit* com o repositório central.
   e) executar os testes localmente, executar um *build* local, atualizar a cópia local do projeto e fazer *commit* com o repositório central.

9. Quanto ao gerenciamento de configuração do *software* e aos serviços de mensageria, julgue o item a seguir.

Em um projeto de *software* que utilize a ferramenta Git para controle de versão, é recomendável que cada desenvolvedor trabalhe em sua própria *branch* local e faça *merge* com a *branch master* apenas quando o código estiver testado e revisado.

Está corretor dizer pois em projetos de software que utilizam Git para controle de versão, é uma prática recomendada que cada desenvolvedor trabalhe em sua própria branch local. Isso permite que o desenvolvedor faça alterações, realize testes e revise o código sem impactar o branch principal (geralmente chamado de "master" ou "main"). Somente após o código ter sido testado e revisado é que o desenvolvedor deve fazer o merge com a branch principal. Essa prática ajuda a manter a estabilidade do código no branch principal e facilita a colaboração entre diferentes desenvolvedores.

10. Quanto ao gerenciamento de configuração do *software* e aos serviços de mensageria, julgue o item a seguir.

Nos serviços de mensageria, a comunicação síncrona via HTTP é mais adequada para cenários de alta concorrência do que a comunicação assíncrona.

Está incorreto pois nos serviços de mensageria, a comunicação assíncrona é geralmente mais adequada para cenários de alta concorrência do que a comunicação síncrona via HTTP.

11. A respeito de interoperabilidade de sistemas, DevOps e configuração de *software*, julgue o item que se segue.

No Git, a informação é tratada como um conjunto de arquivos, sendo a primeira versão armazenada de forma completa, e apenas as mudanças são armazenadas nas versões seguintes.

Está incorreto pois no Git, a abordagem para o versionamento é diferente.

12. Julgue o seguinte item, relativo a DevOps, Jenkins e GIT.

> No ambiente GIT, uma *branch* é definida como uma coleção de referências junto com um banco de dados de objetos que contém todos os objetos que são acessíveis a partir das referências dos "ramos" do desenvolvimento.

Está correto porque em Git, uma branch é de fato uma referência para um commit específico. Cada branch é associada a um ponteiro (ou referência) que aponta para um commit no repositório.

13. Com relação ao desenvolvimento Java EE, a padrões e antipadrões de projeto Java EE, a *software* de versionamento e guarda de fontes e a conceitos de arquitetura monolítica e microsserviços, julgue o item subsequente.

14. A IaC declarativa especifica as propriedades dos recursos de infraestrutura que deseja provisionar e, em seguida, a ferramenta IaC descobre como alcançar esse resultado final por conta própria.

O item está correto, a infraestrutura como código (IaC) declarativa especifica o estado desejado da infraestrutura em termos de propriedades e configurações, e a ferramenta IaC é responsável por determinar como alcançar esse estado. Em contraste, a abordagem imperativa de IaC define uma série de passos específicos a serem seguidos para configurar a infraestrutura. Com IaC declarativa, você descreve o "o quê" você quer, e a ferramenta decide "como" alcançar esse objetivo.

15. Caso se pretenda criar, no desenvolvimento de um novo código em certo projeto de software, um espaço no repositório Git que seja independente do principal, a fim de fazer alterações sem interferências no código principal, então isso poderá ser feito por meio do uso do recurso denominado

a) branch.
b) commit.
c) release.
d) rollback.
e) *restore.*

16. Quanto a aspectos associados ao processo de gerenciamento de configurações de *softwares,* julgue o item subsecutivo

Na criação de um *release* de um sistema, o código executável de programas e todos os arquivos de dados associados devem ser coletados e identificados, e as descrições de configuração podem ter que ser escritas para *hardwares* diferentes e para instruções e sistemas operacionais preparados para clientes que necessitem configurar os próprios sistemas.

Está correto pois na criação de um release de um sistema, é importante coletar e identificar o código executável, arquivos de dados associados e quaisquer descrições de configuração necessárias. Essas descrições podem precisar ser adaptadas para diferentes tipos de hardware, sistemas operacionais e ambientes de execução, especialmente se os clientes forem responsáveis por configurar os sistemas por conta

própria. O gerenciamento de configurações de software envolve garantir que todos os componentes necessários estejam bem documentados e preparados para diferentes ambientes onde o sistema será executado.

17. Quanto a aspectos associados ao processo de gerenciamento de configurações de *softwares*, julgue o item subsecutivo.

As ferramentas de *workbenches* abertas fornecem recursos integrados para controlar versões de *software*, a construção de sistemas e o rastreamento de mudanças, facilitando e simplificando a troca de dados, incluindo um banco de dados integrado de controle de mudanças.

Está correto pois ferramentas de workbench abertas (ou ferramentas de desenvolvimento integradas) frequentemente fornecem recursos para controle de versões de software, construção de sistemas e rastreamento de mudanças. Elas facilitam a troca de dados e muitas vezes incluem um banco de dados integrado de controle de mudanças para gerenciar versões e alterações no código-fonte e em outros artefatos do projeto. Essas ferramentas ajudam a simplificar o processo de desenvolvimento e gerenciamento de configurações ao integrar várias funcionalidades em uma única plataforma.

18. Julgue o item seguinte, relativos às ferramentas de gestão de configuração.

No Git, havendo a necessidade de criar uma nova *branch* de nome systemmobile quando, por exemplo, se deseja adicionar código a um projeto, mas não se tem certeza se o código funciona corretamente, é possível criar a referida *branch* por meio do comando git add -b systemmobile.

O comando mencionado está incorreto para a criação de uma nova branch no Git.

Para criar uma nova branch chamada `systemmobile`, você deve usar o comando:

git branch systemmobile

Ou, se você deseja criar a branch e imediatamente mudar para ela, use:

git checkout -b systemmobile

O comando `git add` é utilizado para adicionar mudanças ao índice (staging area) e não para criar branches.

19. No Git, o usuário, para compartilhar um *commit* com membros de sua equipe de desenvolvimento, deve executar os três passos descritos a seguir: adicionar arquivos da cópia de trabalho à área de *staging*, usando o comando git add; enviar para seu repositório local, usando o comando git push; e enviar para um repositório remoto compartilhado, usando o comando git checkout.

Está errado pois para compartilhar um commit com membros de sua equipe de desenvolvimento no Git, os passos corretos são:

- Adicionar arquivos da cópia de trabalho à área de staging usando o comando git add.

- Criar um commit local com o comando git commit.

- Enviar o commit para um repositório remoto compartilhado usando o comando git push.

O comando git checkout não é usado para enviar commits para um repositório remoto; ele é usado para alternar entre branches ou restaurar arquivos. Portanto, o passo correto para enviar alterações para um repositório remoto é usar git push.

20. Com relação à arquitetura de *software*, julgue o próximo item.

Nos sistemas de versionamento de *software*, o repositório de artefatos deverá manter as duas últimas versões para *backup*, o que, por conseguinte, leva à exclusão das demais versões.

Está incorreto pois os sistemas de versionamento de software, o repositório de artefatos não é obrigado a manter apenas as duas últimas versões. A política de retenção de versões pode variar dependendo dos requisitos do projeto ou da organização. Muitas vezes, é desejável manter um histórico mais extenso de versões por várias razões, como rastreamento de mudanças, recuperação de versões anteriores e auditoria. A exclusão de versões anteriores pode ser feita com base em critérios específicos de retenção ou espaço, mas não há uma regra universal que imponha a retenção apenas das duas últimas versões.

21. A ferramenta de controle de versão Subversion (SVN)

a) é considerada um sistema *peer-to-peer* e, embora seja uma ferramenta proprietária que requer aquisição de licença para uso, pode rodar (executar) usando o servidor Apache.

b) suporta *commits* atômicos, sem deixar inconsistências, mesmo diante de problemas que ocorrem na rede ou no servidor.

c) tem como principal desvantagem o fato de não permitir as operações *file-lock* ou *checkout* reservado.

d) fornece automaticamente uma camada extra de proteção a arquivos e pastas adicionais armazenados por ela, os quais, por isso, não podem ser corrompidos pelo usuário.
e) mantém um histórico detalhado de todos os arquivos removidos, com exceção daqueles que foram renomeados.

22. Assinale a opção que apresenta a funcionalidade do Subversion que permite ao usuário criar um repositório remoto em determinado diretório em seu repositório.

a) Merge
b) Branching
c) Externals
d) Trunk
e) Tagging

23. Com relação a *subversion*, julgue o item subsecutivo.

Subversion é um sistema genérico para gerenciar qualquer coleção de arquivos, como, por exemplo, uma lista de compras.

Está incorreto pois o Subversion (SVN) é um sistema de controle de versão projetado para gerenciar e versionar coleções de arquivos e diretórios que fazem parte de um projeto de software ou de um repositório de dados estruturados. Embora tecnicamente possa ser usado para versionar qualquer tipo de arquivos, como uma lista de compras, não é o uso mais comum ou recomendado para a ferramenta. Subversion é especificamente desenvolvido para rastrear mudanças e controlar versões em projetos de software e outros conjuntos de arquivos com estruturas e relacionamentos complexos.

24. A respeito da engenharia de *software*, julgue o seguinte item.

Entre as disciplinas da engenharia de *software*, inclui-se a gestão de configurações, que, aliada à memória humana em pequenos projetos, consegue evitar que artefatos corrigidos reapareçam durante o desenvolvimento do projeto.

Está incorreto, a gestão de configurações é uma disciplina fundamental na engenharia de software que ajuda a gerenciar e controlar as mudanças em artefatos e componentes de software ao longo do ciclo de vida do desenvolvimento. Ela é essencial para garantir a integridade e a rastreabilidade das mudanças.

Embora a memória humana possa desempenhar um papel em pequenos projetos, confiar apenas nela não é suficiente para evitar que artefatos corrigidos reapareçam ou para manter o controle adequado das mudanças. A gestão de configurações, com o uso de ferramentas e práticas adequadas, é crucial para garantir que as correções e alterações sejam devidamente registradas e que não ocorram regressões ou problemas de versão durante o desenvolvimento.

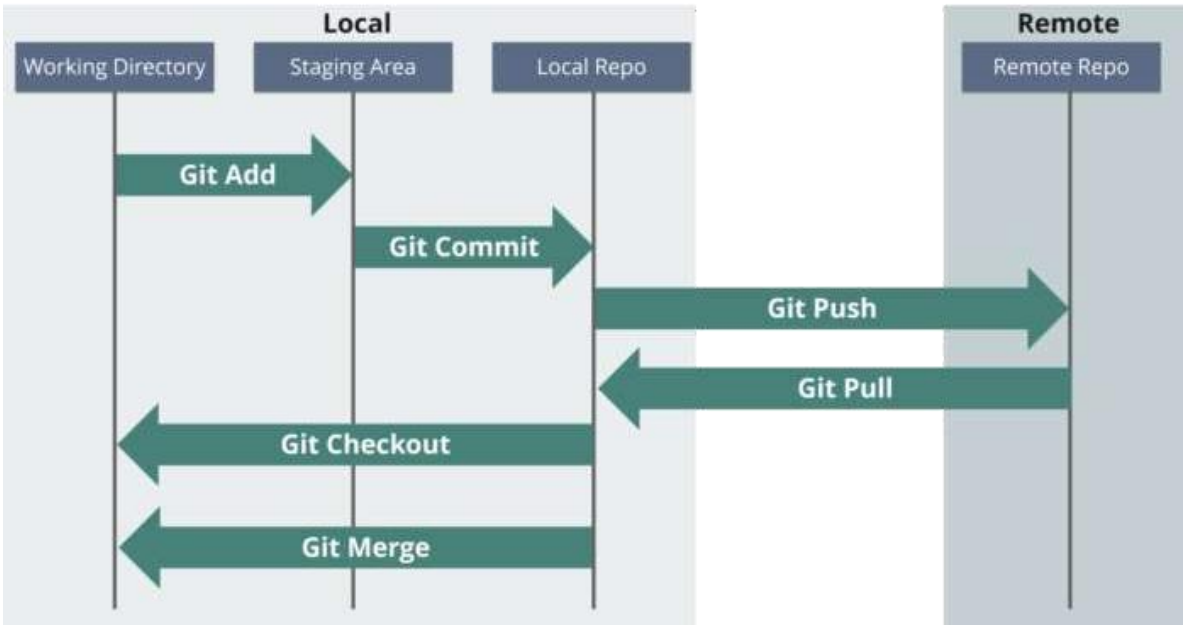
25. No Subversion, um projeto pronto para ser liberado e considerado como uma versão estável é copiado para uma pasta *branch* e fica congelado para que seja testado.

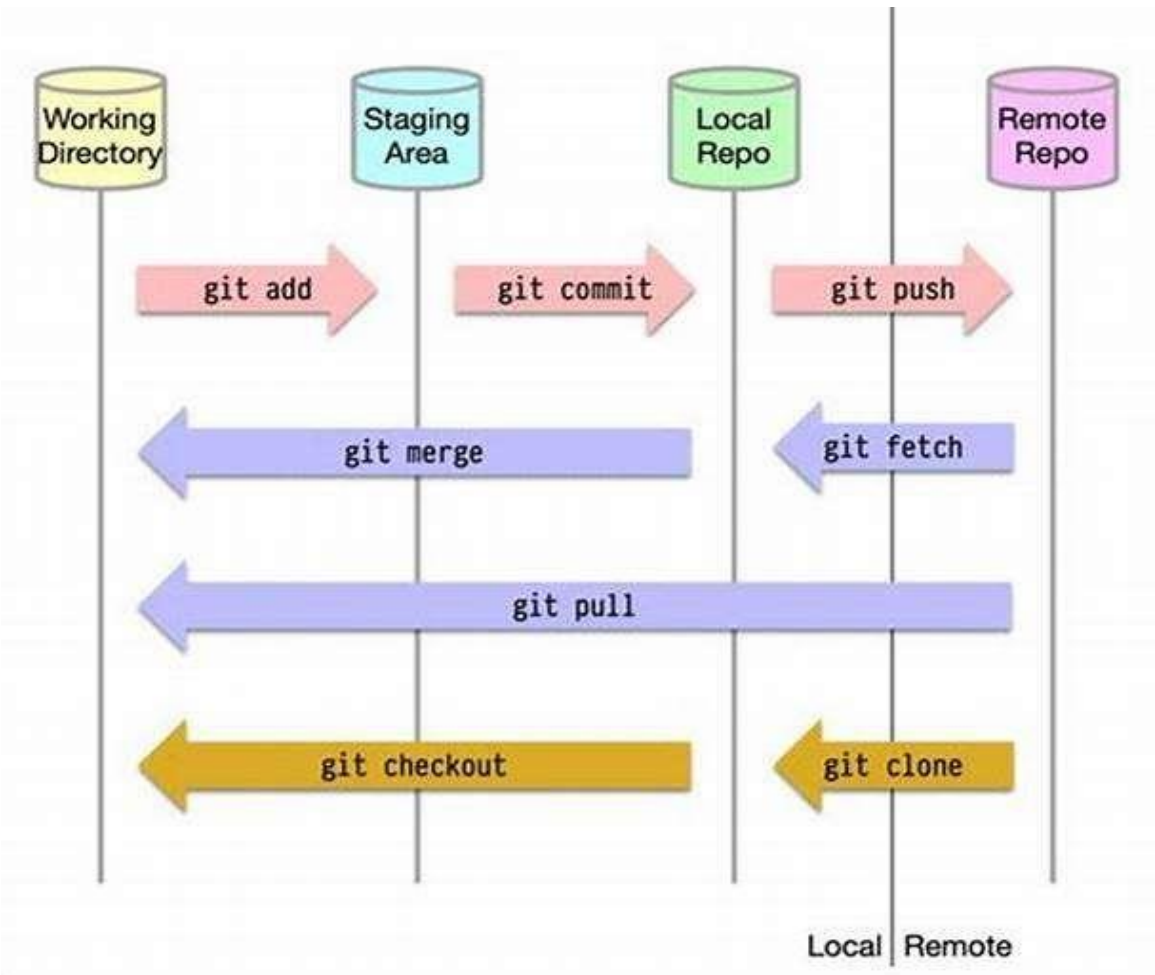
Está mais ou menos correto dizer pois o Subversion (SVN), a prática descrita refere-se à criação de uma branch para uma versão estável do projeto que está pronta para ser liberada, mas não é exatamente o que se faz com uma branch no SVN.

26. Considerando um programa em linguagem Java, assinale a opção que apresenta o comando do versionador Git que permite criar uma *branch* de nome new_branch e mudar para essa *branch* ao mesmo tempo.

a) git log new_branch
b) git clone new_branch
c) git checkout -b new_branch
d) git init new_branch
e) git commit -m 'new_branch'

Exemplos Git workflow :





No Git, o **diretório de trabalho** (ou *working directory*) é o local onde você faz as alterações nos arquivos do seu projeto. Ele contém uma cópia dos arquivos do projeto em um estado específico, geralmente o mais recente commit do branch em que você está trabalhando.

O git add atualiza o índice (index) com o conteúdo atual dos arquivos no diretório de trabalho. Isso significa que ele prepara os arquivos modificados, novos ou deletados para serem incluídos no próximo commit.

O comando git commit é usado para gravar as mudanças na área de stage no repositório local, criando um novo snapshot do estado atual do projeto.

O git push atualiza as referências remotas usando as referências locais, enviando os objetos necessários para completar as referências dadas. Isso é essencial para compartilhar seu trabalho com outros colaboradores ou para fazer backup do seu código em um servidor remoto.

O comando git fetch é usado para baixar objetos e referências (refs) de um repositório remoto para o seu repositório local, sem integrá-los automaticamente ao seu trabalho.

O comando git merge é usado para combinar mudanças de diferentes branches em um único branch no Git.

O comando git pull é usado para atualizar seu repositório local com as mudanças do repositório remoto. Ele é uma combinação de dois comandos: git fetch e git merge.

O git clone copia todo o conteúdo de um repositório remoto, incluindo todos os arquivos, branches e commits, para o seu repositório local. Isso é útil para começar a trabalhar em um projeto existente ou para colaborar com outros desenvolvedores.

O git checkout permite que você mude para um branch diferente ou restaure arquivos específicos para um estado anterior. É uma ferramenta versátil que pode ser usada para várias operações no Git

Uma branch (ou ramificação) no Git é uma cópia separada do código onde você pode fazer mudanças sem afetar a linha principal de desenvolvimento. Isso é útil para adicionar novas funcionalidades, corrigir erros ou testar novas ideias sem arriscar comprometer o que já está funcionando.

A **linha principal de desenvolvimento** no Git, frequentemente chamada de **branch principal** ou **branch padrão**, é o branch onde o código estável e pronto para produção é mantido. Em muitos projetos, essa linha principal é chamada de **main ou master**

Principais Características de uma Branch:

1. **Isolamento**: Permite que você trabalhe em novas funcionalidades ou correções de bugs de forma isolada.
2. **Segurança**: Evita que mudanças instáveis sejam mescladas diretamente no código principal.
3. **Colaboração**: Facilita o trabalho em equipe, permitindo que diferentes desenvolvedores trabalhem em diferentes branches simultaneamente.