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.
O gerenciamento das configurações de servidores é essencial para garantir que os sistemas permaneçam seguros e funcionem conforme o esperado. Isso envolve várias atividades, como: Rastreamento de Configurações: Monitorar e registrar todas as configurações atuais dos servidores. Atualizações de Software: Garantir que todas as versões de software estejam atualizadas para evitar vulnerabilidades.

2. No que se refere aos conceitos de gestão de configuração, julgue o item subsequente.
comando *git clone* incorpora as alterações de um repositório remoto no ramo atual. E vai explicar como vai funcionar O item é falso. O comando git clone é utilizado para criar uma cópia local de um repositório remoto. Ele não incorpora alterações no ramo atual; em vez disso, ele cria uma nova cópia do repositório, incluindo todos os seus ramos e histórico de commits. Para incorporar alterações de um repositório remoto em um ramo local existente, utiliza-se o comando git pull ou git fetch seguido por git merge.

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.

Uma das boas práticas do DevOps é, de fato, a adoção de uma cultura livre de culpa. Em uma cultura DevOps, o foco é na melhoria contínua e na resolução de problemas, em vez de culpar indivíduos por erros

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.

   Correto. Cada desenvolvedor deve trabalhar em sua própria branch e fazer merge com a branch master quando o código estiver testado e revisado.

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.

    Incorreto. A comunicação assíncrona é geralmente mais adequada para cenários de alta concorrência.

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.
Incorreto No Git, a primeira versão não é armazenada de forma completa; apenas as mudanças são armazenadas nas versões seguintes.

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.
Correto. Uma branch no Git é uma coleção de referências e um banco de dados de objetos.
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.
A IaC declarativa especifica as propriedades dos recursos e a ferramenta descobre como alcançá-las.

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

Correto. Na criação de um release, o código executável e arquivos associados devem ser coletados e identificados, e descrições de configuração podem ser necessárias.

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.

Ferramentas de workbenches abertas fornecem recursos integrados para controle de versões, construção de sistemas e rastreamento de mudanças.

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.

Incorreto. Para criar uma nova branch, usa-se git branch e não git add -b.

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.

Incorreto. Para compartilhar um commit, usa-se git push e não git checkout.

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.

Incorreto. O repositório de artefatos pode manter várias versões, não apenas as duas últimas.

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.

Incorreto. Subversion não é um sistema genérico para gerenciar qualquer coleção de arquivos.

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.

Correto A gestão de configurações ajuda a evitar que artefatos corrigidos reapareçam.

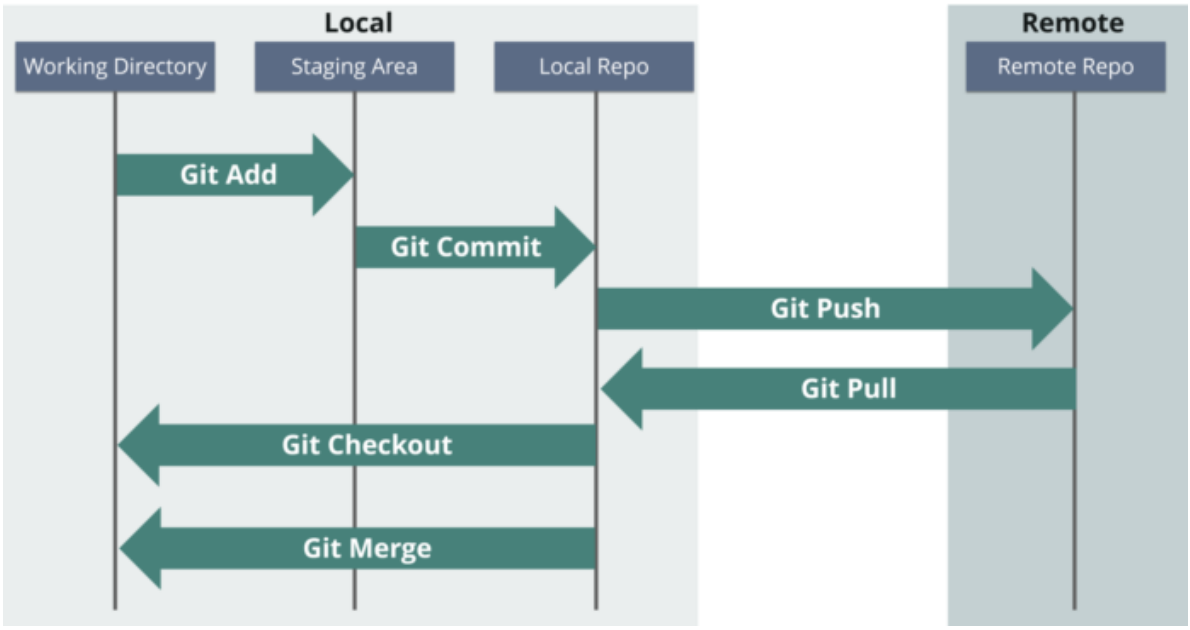
25. No Subversion, um projeto pronto para ser liberado e considerado como uma versão estável é copiado para uma pasta *branch* e fica congelado para que seja testado.

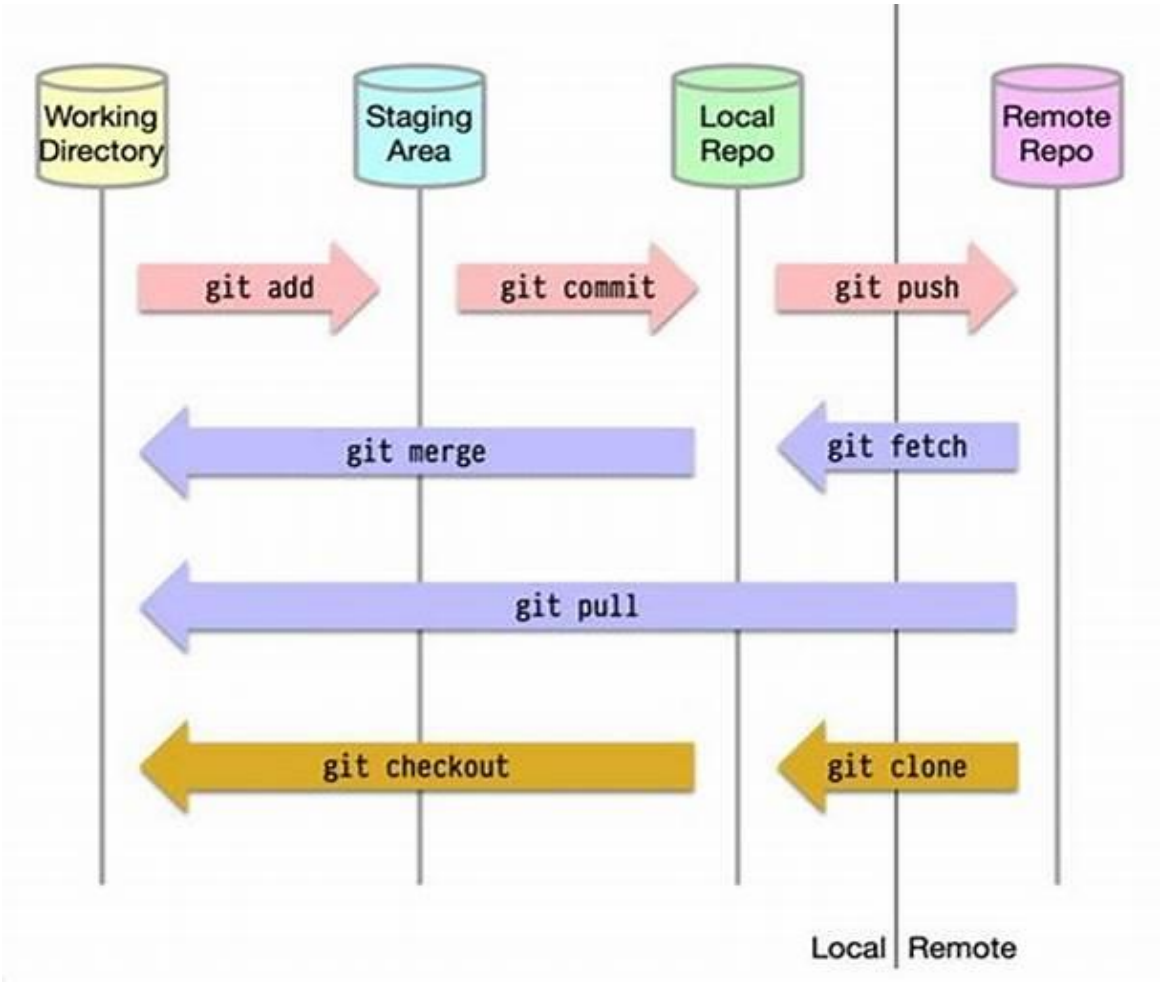
Correto. No Subversion, um projeto pronto para ser liberado é copiado para uma branch e congelado para testes.

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 :


Local
Working Directory
Staging Area
Local Repo
Remote
Remote Repo
Git Add
Git Commit
Git Push
Git Pull
Git Checkout
Git Merge


Working Directory
Staging Area
Local Repo
Remote Repo
git add
git commit
git push
git merge
git fetch
git pull
git checkout
git clone
Local
Remote

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.