Estratégia Concursos

Aula 00

BNB (Especialista Técnico - Analista de Sistemas Perfil 2: Infraestrutura e Segurança da Informação) Desenvolvimento de Software

Autor:
Diego Carvalho

03 de Março de 2023

Índice

FERRAMENTAS DE CONTROLE DE VERSÃO

GIT

O que é "controle de versão", e por que eu deveria me importar? **Controle de versão** é um sistema que **registra alterações em um arquivo ou conjunto de arquivos** ao longo do tempo para que você possa lembrar as versões específicas mais tarde. É possível realizar o **Controle de versão** com quase qualquer tipo de arquivo em um computador.

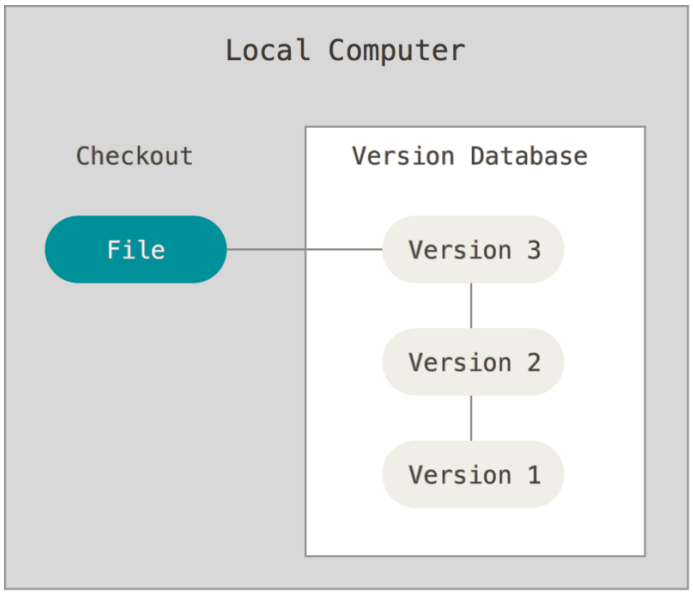
Se você é um designer gráfico ou web designer e todas as versões de uma imagem ou layout (o que você certamente deve querer), um **sistema de controle de versão (SCV)** é a coisa correta a ser usada.

Ele permite que você reverta para um estado anterior determinados arquivos ou um projeto inteiro, compare as mudanças ao longo do tempo, veja quem modificou pela última vez algo que pode estar causando um problema, quem introduziu um problema, quando, e muito mais.

Usar um SCV também significa que **se você estragar tudo ou perder arquivos, você pode facilmente recuperar.** 😆. Além disso, você tem tudo isso com **muito pouco trabalho**. Não é totalmente incrível? 🤩

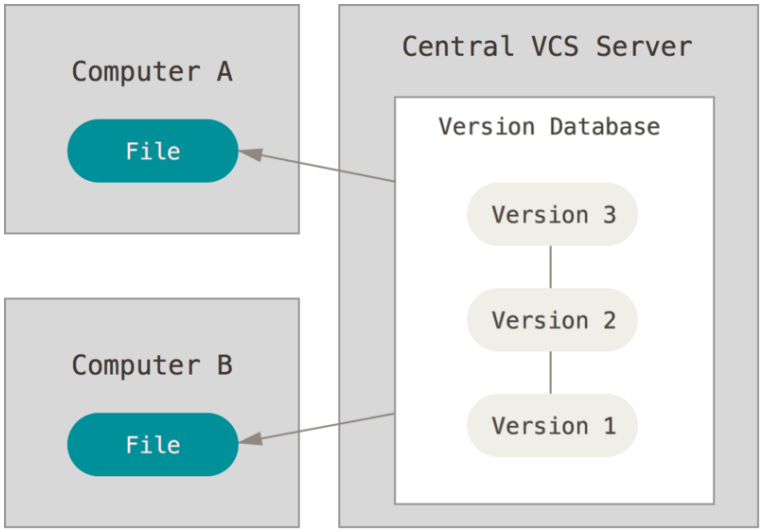
O método de controle da versão de muitas pessoas é **copiar os arquivos para outro diretório** (talvez um diretório com carimbo de tempo, se eles são espertos). Esta abordagem é muito comum, porque é incrivelmente propensa a erros. É fácil esquecer em qual diretório você está e acidentalmente sobrescrever o arquivo errado ou copiar arquivos que não quer.

Para lidar com este problema, programadores há muito tempo desenvolveram SCVs locais que tem um banco de dados simples que mantêm todas as alterações nos arquivos sob controle de revisão.



Uma das ferramentas SCV mais populares foi um sistema chamado **RCS**, que ainda é distribuído para muitos computadores. Até mesmo o popular sistema operacional Mac OS X inclui o comando **rcs** quando você instala as Ferramentas de Desenvolvimento. RCS funciona mantendo conjuntos de alterações (ou seja, as diferenças entre os arquivos) em um formato especial no disco; ele pode, em seguida, recriar como qualquer arquivo se parecia em qualquer ponto no tempo, adicionando-se todas as alterações.

Sistemas Centralizados de Controle de Versão



A próxima questão importante que as pessoas encontram é que elas precisam colaborar com desenvolvedores em outros sistemas. Para lidar com este problema, Sistemas Centralizados de Controle de Versão (**CVCSs**) foram desenvolvidos. Estes sistemas, tais como **CVS, Subversion e Perforce**, têm **um único servidor** que contém todos os arquivos de controle de versão, e um número de clientes que usam arquivos a partir desse lugar central. Por muitos anos, este tinha sido o padrão para controle de versão.

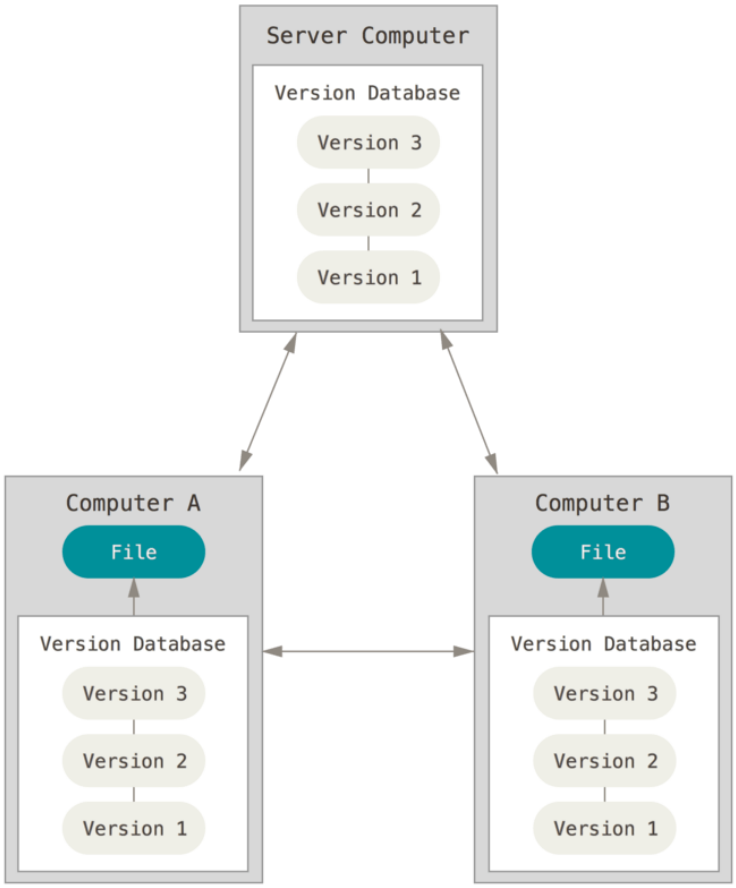
Esta configuração oferece muitas vantagens, especialmente sobre SCVs locais. Por exemplo, todos sabem, até certo ponto o que cada um no projeto está fazendo. Os administradores têm controle refinado sobre quem pode fazer o que; e é **muito mais fácil de administrar um CVCS** do que lidar com bancos de dados locais em cada cliente.

No entanto, esta configuração também tem **algumas desvantagens graves**. O mais óbvio é o **ponto único de falha** que o **servidor centralizado** representa. Se esse servidor der problema por uma hora, durante essa hora ninguém pode colaborar ou salvar as alterações de versão para o que quer que eles estejam trabalhando. Se o disco rígido do banco de dados central for corrompido, e backups apropriados não foram mantidos, você perde absolutamente tudo - toda a história do projeto, exceto imagens pontuais que desenvolvedores possam ter em suas máquinas locais. Sistemas SCV

locais sofrem com esse mesmo problema - sempre que você possui toda a história do projeto em um único lugar, há o risco de perder tudo.

Sistemas Distribuídos de Controle de Versão

É aqui que **Sistemas Distribuídos de Controle de Versão (DVCS)** entram em cena. Em um DVCS (como **Git, Mercurial, Bazaar ou Darcs**), clientes não somente usam o estado mais recente dos arquivos: eles duplicam localmente o repositório completo. Assim, se qualquer servidor morrer, e esses sistemas estiverem colaborando por meio dele, qualquer um dos repositórios de clientes pode ser copiado de volta para o servidor para restaurá-lo. **Cada clone é de fato um backup completo de todos os dados.**



Além disso, muitos desses sistemas trabalham muito bem com vários repositórios remotos, tal que você possa colaborar em diferentes grupos de pessoas de maneiras diferentes ao mesmo tempo dentro do mesmo projeto. Isso permite que você configure vários tipos de fluxos de trabalho que não são possíveis em sistemas centralizados, como modelos hierárquicos.

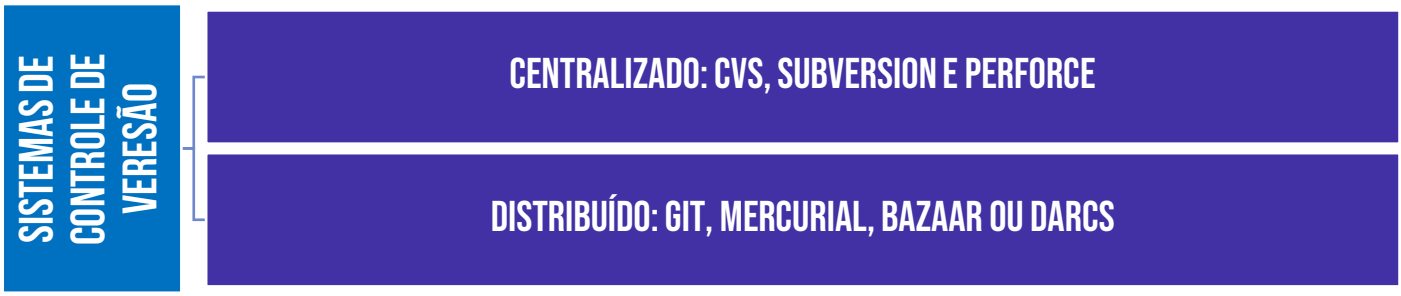
Como muitas coisas na vida, o Git começou com um pouco de destruição e uma ardente controvérsia.

O núcleo (kernel) do Linux é um projeto de código aberto com um escopo bastante grande. A maior parte da manutenção do núcleo do Linux 1991-2002), as mudanças no código eram compartilhadas como parte da vida em arquivos. Em 2020, o projeto do Linux começou a usar um núcleo DVCS chamado0 do BitKeeper.

Em 2005, **a relação** entre a comunidade que desenvolveu o núcleo do Linux e a empresa que desenvolveu BitKeeper **quebrou em pedaços**, e a ferramenta passou a ser paga. Isto alertou a comunidade que desenvolvia o Linux (e especialmente Linux Torvalds, o criador do Linux) a desenvolver a sua própria ferramenta baseada em lições aprendidas ao usar o BitKeeper. Algumas metas do novo sistema era os seguintes:

- Velocidade
- Projeto simples

- Forte suporte para desenvolvimento não-linear (milhares de ramos paralelos)
- Completamente distribuído
- Capacidade de lidar com projetos grandes como o núcleo o Linux com eficiência (velocidade e tamanho dos dados)



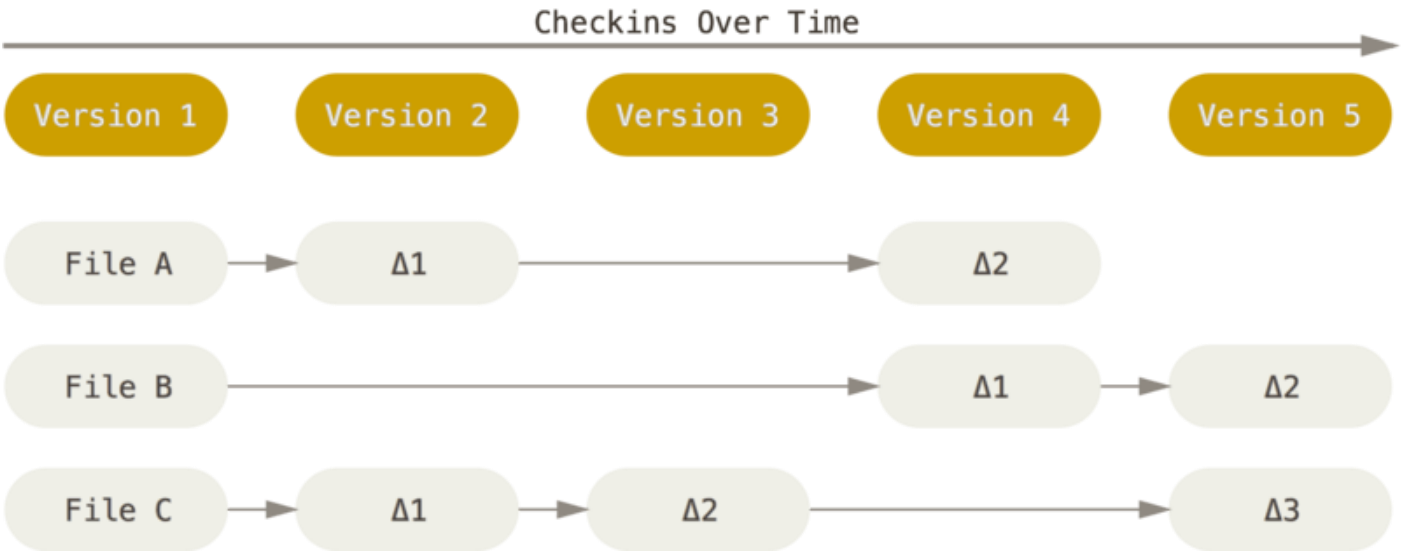
Desde seu nascimento em 2005, **Git** evoluiu e amadureceu para ser **fácil de usar** e ainda reter essas qualidades iniciais. Ele é **incrivelmente rápido, é muito eficiente com projetos grandes,** e ele tem um **incrível sistema de ramos para desenvolvimento não linear.**

O Básico do Git

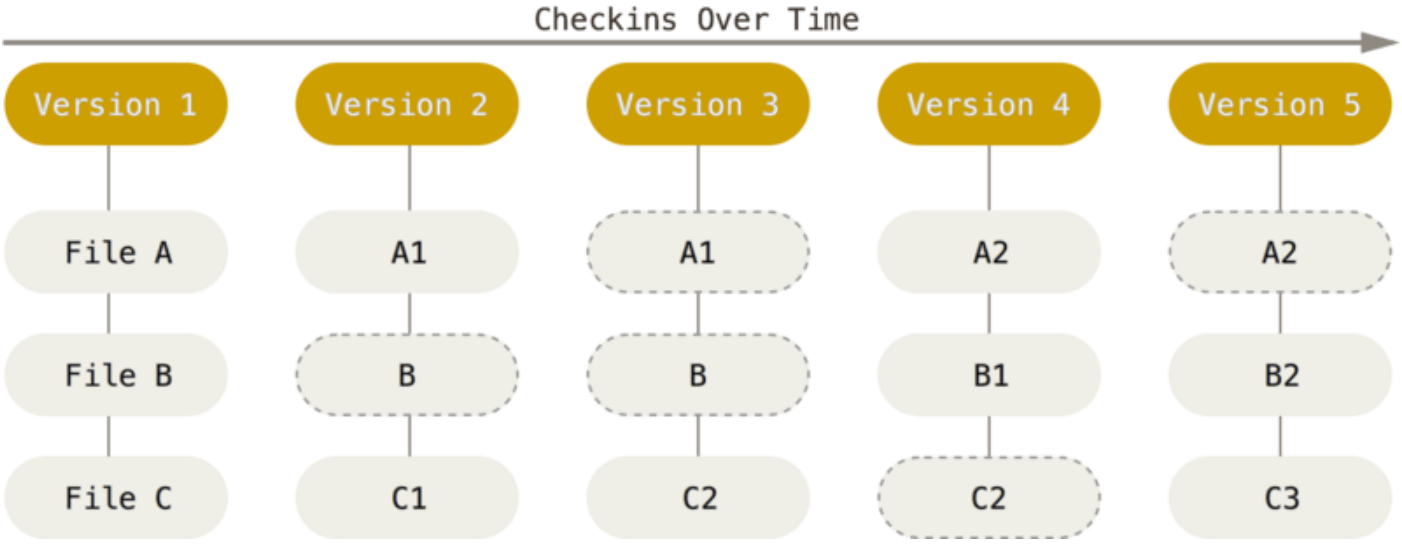
Então, em poucas palavras, o que é o Git? Esta é uma parte que é importante aprender, porque se você entender o que o Git é e os fundamentos de como ele funciona, em seguida, provavelmente usar efetivamente o Git será muito mais fácil para você.

O Git armazena e vê informações de forma muito diferente do que esses outros sistemas, mesmo que a interface do usuário seja bem semelhante, e entender essas diferenças o ajudará a não ficar confuso.

A **principal diferença** entre o Git e qualquer outro SCV (Subversion e similares) **é a maneira como o Git trata seus dados.** Conceitualmente, a maioria dos outros sistemas armazenam informação como uma lista de mudanças nos arquivos. Estes sistemas (CVS, Subversion, Perforce, Bazaar, e assim por diante) **tratam a informação como um conjunto de arquivos** e as mudanças feitas em cada arquivo ao longo do tempo.



O Git não trata nem armazena seus dados desta forma. Em vez disso, o Git trata seus dados mais como um **conjunto de imagens de um sistema de arquivos em miniatura**. Toda vez que você fizer um commit, ou salvar o estado de seu projeto no Git, ele **basicamente captura uma foto de todos os seus arquivos e armazena uma referência para esse conjunto de arquivos**. Para ser eficiente, se os arquivos não foram alterados, o Git não armazena o arquivo novamente, apenas um link para o arquivo idêntico anterior já armazenado. O Git trata seus dados mais como um fluxo do estado dos arquivos.



Esta é uma diferença importante entre o Git e quase todos os outros SCVs. Isto faz o Git reconsiderar quase todos os aspectos de controle de versão que a maioria dos outros sistemas copiaram da geração anterior. Isso faz com que o Git seja mais como um minissistema de arquivos com algumas **ferramentas incrivelmente poderosas**, ao invés de **simplesmente um SCV**. Vejamos alguns dos benefícios que você ganha ao tratar seus dados desta forma quando cobrirmos ramificações no Git.

Quase Todas as Operações são locais

A maioria das operações no Git só precisa de arquivos e recursos locais para operar - geralmente nenhuma informação é necessária de outro computador da rede. Se você estiver acostumado com um CVCS onde a maioria das operações têm aquela demora causada pela latência da rede, este aspecto do Git vai fazer você pensar que **os deuses da velocidade abençoaram o Git** com poderes extraterrestres. 😅 Como você tem toda a história do projeto ali mesmo em seu disco local, **a maioria das operações parecem quase instantâneas.**

Por exemplo, para pesquisar o histórico do projeto, o Git não precisa ir para o servidor para obter a história e exibi-la para você - ele simplesmente lê diretamente do seu banco de dados local. Isto significa que você vê o histórico do projeto quase que instantaneamente. Se você quiser ver as alterações introduzidas entre a versão atual de um arquivo e o arquivo de um mês atrás, o Git pode procurar o arquivo de um mês atrás e fazer um cálculo de diferença local, em vez de ter que pedir a um servidor remoto para fazê-lo ou puxar uma versão mais antiga do arquivo do servidor remoto para fazê-lo localmente.

Isto também significa que há muito pouco que você não pode fazer se você estiver desconectado ou sem VPN. Se você estiver em um avião ou um trem e quiser trabalhar um pouco, você pode fazer commits alegremente até conseguir conexão de rede e enviar os arquivos. Se você chegar em casa e não conseguir conectar ao VPN, você ainda poderá trabalhar. Em muitos outros sistemas, isso é impossível ou "doloroso". No Perforce, por exemplo, você não pode fazer quase nada se você não estiver conectado ao servidor; e no Subversion e CVS, você pode editar os arquivos, mas não poderá enviar commits das alterações ao seu banco de dados (porque você não está conectado ao seu banco de dados). Isso pode não parecer muito, mas você poderá se surpreender com a grande diferença que isso pode fazer.

Git tem Integridade

Tudo no Git passa por uma soma de verificações (checksum) antes de ser armazenado e é referenciado por esse checksum. Isto significa que é impossível mudar o conteúdo de qualquer arquivo ou pasta sem que Git saiba. Esta funcionalidade está incorporada no Git nos níveis mais baixos e é parte integrante de sua filosofia. Você não perderá informação durante a transferência e não receberá um arquivo corrompido sem que o Git seja capaz de detectar.

O mecanismo que o Git utiliza para esta soma de verificação é chamado um hash SHA-1. Esta é uma sequência de 40 caracteres composta de caracteres hexadecimais (0-9 e-f) e é calculada com base no conteúdo de uma estrutura de arquivo ou diretório no Git. Um hash SHA-1 é algo como o seguinte:

```
24b9da6552252987aa493b52f8696cd6d3b00373
```

Você vai ver esses valores de hash em todo o lugar no Git porque ele os usa com frequência. Na verdade, **o Git armazena tudo em seu banco de dados** não pelo nome do arquivo, mas pelo valor de hash do seu conteúdo.

O Git geralmente somente adiciona dados

Quando você faz algo no Git, quase sempre dados são adicionados no banco de dados do Git - e não removidos. É difícil fazer algo no sistema que não seja reversível ou fazê-lo apagar dados de forma alguma. Como em qualquer SCV, você pode perder alterações que ainda não tenham sido adicionadas em um commit; mas depois de fazer o commit no Git do estado atual das alterações, é muito difícil que haja alguma perda, especialmente se você enviar regularmente o seu banco de dados para outro repositório. Isso faz com que o uso do Git seja somente alegria, porque sabemos que podemos experimentar sem o perigo de estragar algo. 😀

Os Três Estados

Agora, preste atenção. **Esta é a principal coisa a lembrar sobre Git** se você quiser que o resto do seu processo de aprendizagem ocorra sem problemas**. O Git tem três estados principais** que seus arquivos podem estar: **committed, modificado (modified) e preparado (staged)**.

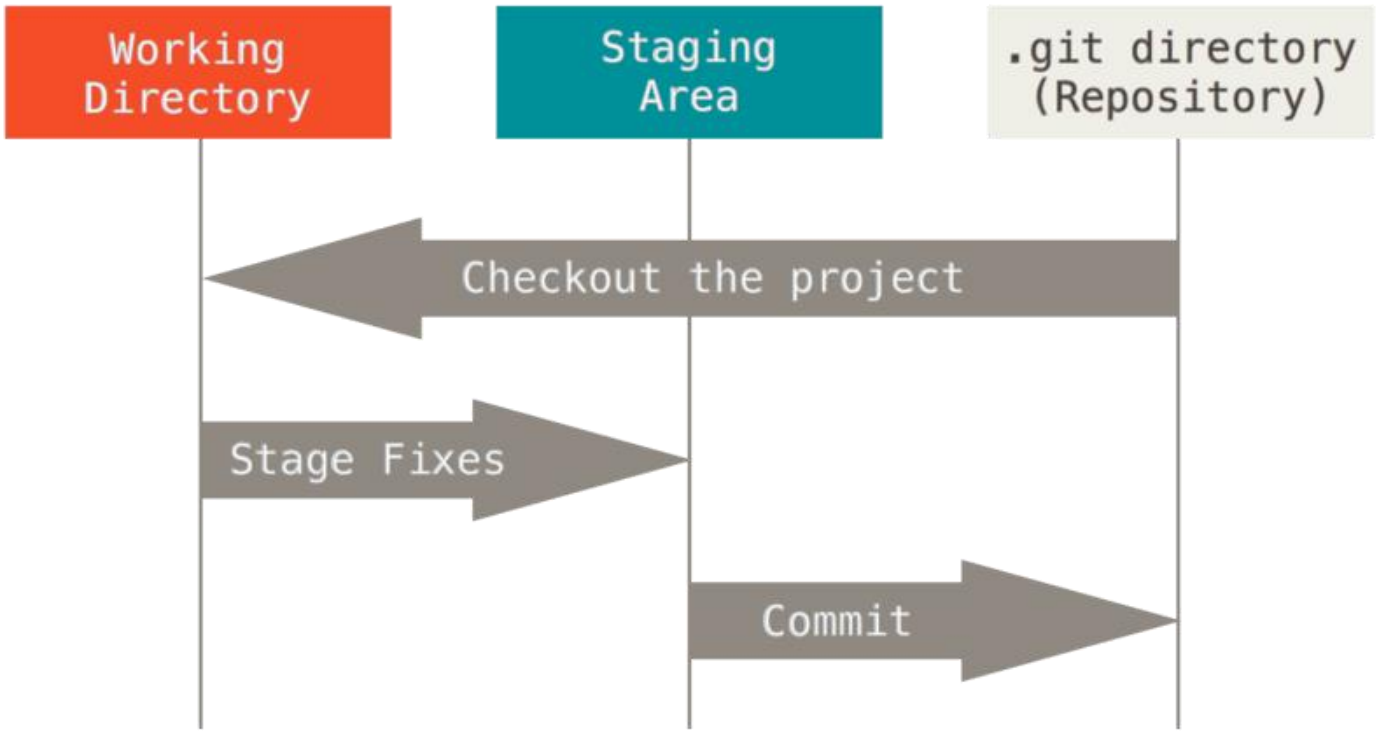
- **Committed** significa que os dados estão armazenados de forma segura em seu banco de dados local.
- **Modified**: significa que você alterou o arquivo, mas ainda não fez o commit no seu banco de dados.
- **Staged**: significa que você marcou a versão atual de um arquivo modificado para fazer parte de seu próximo commit.

(VUNESP – FUNDUNESP– 2016) O Git, ao tratar os arquivos que devem sofrer o processo de controle de versões, classifica o estado desses arquivos em 3 categorias, definidas como

a) checked (verificado), tracked (acompanhado) e identified (identificado).
b) committed (consolidado), modified (modificado) e staged (preparado).
c) identified (identificado), ignored (ignorado) e committed (consolidado).
d) ignored (ignorado), ready (pronto) e staged (preparado).
e) ready (pronto), cloned (clonado) e modified (modificado).

Comentários: O Git tem três estados principais que seus arquivos podem estar: committed, modificado (modified) e preparado (staged). (Gabarito: Letra B)

Isso nos leva a três seções principais de um projeto Git: o diretório Git, o diretório de trabalho e área de preparo.



O diretório Git é onde o Git armazena os metadados e o banco de dados de objetos de seu projeto. Esta é a parte mais importante do Git, e é o que é copiado quando você clona um repositório de outro computador.

O diretório de trabalho é uma simples cópia de uma versão do projeto. Esses arquivos são pegos do banco de dados compactado no diretório Git e colocados no disco para você usar ou modificar.

A área de preparo é um arquivo, geralmente contido em seu diretório Git, que armazena informações sobre o que vai entrar em seu próximo commit. É por vezes referido como o "índice", mas também é comum referir-se a ele como área de preparo (staging area).

O fluxo de trabalho básico Git é algo assim:

1. Você **modifica** arquivos no seu diretório de trabalho.
2. Você **prepara** os arquivos, adicionando imagens deles à sua área de preparo.
3. Você faz **commit**, o que leva os arquivos como eles estão na área de preparo e armazena essas imagens de forma permanente para o diretório do Git.

Se uma versão específica de um arquivo está no diretório Git, é considerado commited. Se for modificado, mas foi adicionado à área de preparo, é considerado preparado. E se ele for alterado depois de ter sido carregado, mas não foi preparado, ele é considerado modificado.

Comandos GIT

Pessoal, os comandos, com toda certeza, são os mais cobrados em provas! Portanto, tenha atenção redobrada!

Comando	Descrição
Criar novo repositório	git init
Verificar estado dos arquivos/diretórios	git status
Adicionar um arquivo em específico (staged area)	git add meu_arquivo.txt
Adicionar um diretório em específico	git add meu_diretorio
Adicionar todos os arquivos/diretórios	git add .
Adicionar um arquivo que esta listado no .gitignore (geral ou do repositório)	git add -f arquivo_no_gitignore.txt
Comitar um arquivo	git commit meu_arquivo.txt
Comitar vários arquivos	git commit meu_arquivo.txt meu_outro_arquivo.txt
Comitar informando mensagem	git commit meuarquivo.txt -m "minha mensagem de commit"
Remover arquivo	git rm meu_arquivo.txt
Remover diretório	git rm -r diretorio
Exibir histórico	git log
Exibir histórico com diff das duas últimas alterações	git log -p -2
Exibir resumo do histórico (hash completa, autor, data, comentário e qtde de alterações (+/-))	git log --stat
Exibir informações resumidas em uma linha (hash completa e comentário)	git log --pretty=oneline
Exibir histórico com formatação específica (hash abreviada, autor, data e comentário)	git log --pretty=format:"%h - %an, %ar : %s" %h: Abreviação do hash; %an: Nome do autor; %ar: Data; %s: Comentário.
Exibir histório de um arquivo específico	git log -- <caminho_do_arquivo>
Exibir histórico de um arquivo específico que contêm uma determinada palavra	git log --summary -S<palavra> [<caminho_do_arquivo>]
Exibir histórico modificação de um arquivo	git log --diff-filter=M -- <caminho_do_arquivo>

Exibir revisão e autor da última modificação de uma bloco de linhas	git blame -L 12,22 meu_arquivo.txt
Desfazendo alteração local (working directory)[1]	git checkout -- meu_arquivo.txt
Desfazendo alteração local (staging area)[2]	git reset HEAD meu_arquivo.txt
Exibir os repositórios remotos	git remote git remote -v
Vincular repositório local com um repositório remoto	git remote add origin git@github.com:usuario/repositorio
Exibir informações dos repositórios remotos	git remote show origin
Renomear um repositório remoto	git remote rename origin curso-git
Desvincular um repositório remoto	git remote rm curso-git
Enviar arquivos/diretórios para o repositório remoto	git push -u origin master
Listar configurações	git config --list
Setar usuário	git config --global user.name "nome"
Setar editor	git config --global core.editor vim
Setar ferramenta de merge	git config --global merge.tool vimdiff
Atualizar os arquivos no branch atual	git pull
Buscar as alterações, mas não aplica-las no branch atual	git fetch
Clonar um repositório remoto já existente	git clone git@<caminho_do_arquivo>
Usa busca binária para encontrar o commit que introduziu um bug	git-bisect

(CESPE - 2020) No GIT, o comando git pull é usado para enviar ao repositório a alteração que foi efetivada no computador local.

Comentários:
Pessoal, na verdade o comando usado para enviar ao repositório a alteração que foi efetivada no computador local é git push. (Gabarito: Errado)

Pessoal, vejamos um arquivo muito cobrado em provas que é o gitignore. Basicamente, ele especifica arquivos não rastreados intencionalmente para ignorar. Um arquivo gitignore especifica arquivos não rastreados intencionalmente que o Git deve ignorar. Arquivos já rastreados pelo Git não são afetados.

[1] Este comando deve ser utilizando enquanto o arquivo não foi adicionado na staged area.
[2] Este comando deve ser utilizando quando o arquivo já foi adicionado na staged area.

Cada linha em um arquivo gitignore especifica um padrão. Ao decidir ignorar um caminho, o Git normalmente verifica padrões gitignore de várias fontes, com a seguinte ordem de precedência, do maior para o menor (dentro de um nível de precedência, o último padrão correspondente decide o resultado):

Padrões lidos de um arquivo. gitignore no mesmo diretório que o caminho, ou em qualquer diretório pai (até o nível superior da árvore de trabalho), com padrões nos arquivos de nível superior sendo substituídos por aqueles em arquivos de nível inferior no diretório contendo o arquivo. Esses padrões correspondem à localização do arquivo .gitignore. Um projeto normalmente inclui esses arquivos .gitignore em seu repositório, contendo padrões para arquivos gerados como parte da construção do projeto.

Questões Comentadas

1. (FCC – PGE-AM – 2022) Um Técnico utilizou corretamente um comando git para modificar a mensagem do commit mais recente, ou seja, o último commit feito por ele no projeto. Trata-se do comando git

a) add merge.
b) push.
c) commit --amend.
d) add message.
e) checkout master.

Comentários: Pessoal, o comando commit --amend é usado para alterar o commit mais recente, veja: Alterar seu commit mais recente é provavelmente a reescrita mais comum do histórico que você fará. Muitas vezes você vai querer fazer duas coisas básicas no seu último commit: simplesmente mudar a mensagem do commit, ou mudar o conteúdo real do commit adicionando, removendo e modificando arquivos. Se você simplesmente deseja modificar sua última mensagem de confirmação, é só útilizar o comando: git commit --amend. Para complementar o conhecimento, o comando git commit --amend carrega a mensagem de confirmação anterior em uma sessão do editor, onde você pode fazer alterações na mensagem, salvar essas alterações e sair. Quando você salva e fecha o editor, o editor escreve um novo commit contendo essa mensagem de commit atualizada e o torna seu novo último commit.

Gabarito: Letra C

2. (FGV – TJDFT – 2022) O analista Mateus configurou um pipeline CI/CD para o projeto TJApp no GitLab. O repositório de TJApp denomina-se TJAppRepo. Mateus precisou controlar o comportamento do pipeline de TJApp condicionando o início de sua execução aos eventos de push de tags para o TJAppRepo.

Para aplicar essa condição ao pipeline de TJApp, Mateus precisou modificar o arquivo gitlab-ci.yml na raiz de TJAppRepo, adicionando uma regra na seção:

a) default;
b) include;
c) stages;
d) variables;

e) workflow.

Comentários: Pessoal, vejamos o que pede a questão: Uma configuração de pipeline GitLab CI/CD inclui: Palavras- chave globais que configuram o comportamento do pipeline. default: Valores padrão personalizados para palavras-chave de trabalho. include: Importe a configuração de outros arquivos YAML. stages: Os nomes e a ordem dos estágios do pipeline. variables: Defina variáveis CI/CD para todos os trabalhos no pipeline. workflow: Controle quais tipos de pipeline são executados. Pessoal, a questão foi contextualizada, mas é necessário se ater ao que foi solicitado: controlar o comportamento do pipeline, dessa forma, temos nosso gabarito na letra e) workflow: Controla quais tipos de pipeline são executados.

Gabarito: Letra E

3. (FGV – TJ TO – 2022) O técnico em informática José está desenvolvendo o software TJTOPlugin com o apoio da ferramenta de versionamento Git. José criou o branch local pluginConnector e efetuou alguns commits neste branch, mas não replicou os commits em um repositório remoto.

   A fim de replicar os commits e criar o branch pluginConnector no repositório remoto origin, utilizando um único comando no terminal de comandos do sistema operacional, José deve executar o comando git com os argumentos:

   a) mv pluginConnector origin;
   b) diff pluginConnector origin;
   c) merge origin/pluginConnector;
   d) push origin pluginConnector;
   e) remote add origin pluginConnector.

Comentários: Pessoal, o comando git-push atualiza referências remotas junto com objetos associados o técnico em informática José deseja replicar os commits e criar o branch pluginConnector no repositório remoto origin. Assim ele deve usar o comando git push origin pluginConnector. Já que esse comando atualiza referências remotas usando referências locais, enquanto envia objetos necessários para completar as referências fornecidas.

Gabarito: Letra D

4. (UFRPE – UFRPE – 2022) O git é um sistema de controle de versão distribuído e utilizado amplamente pela comunidade de desenvolvimento de software. Esse sistema possui um conjunto de comandos utilizados para o versionamento de código. Dito isso, qual o comando utilizado para enviar as alterações do repositório local para o repositório remoto?

a) git commit
b) git push
c) git add
d) git pull
e) git send

Comentários: Vamos relembrar cada comando: a) git commit: Comitar um arquivo. b) git push: Enviar arquivos/diretórios para o repositório remoto. c) git add: Adicionar um arquivo. d) git pull: Atualizar os arquivos no branch atual. e) git send: provavelmente o examinador quis dizer: git send-email: envia um e-mail.

Gabarito: Letra B

5. (CESPE – MPE CE – 2020) GitHub é uma plataforma de hospedagem de código que permite realizar o controle de versão de software, de modo que várias pessoas contribuam simultaneamente no mesmo projeto, editando e criando novos arquivos, sem o risco de suas alterações serem sobrescritas.

Comentários: Pessoal, definicao perfeita do Git. GitHub é uma plataforma de hospedagem de código-fonte e arquivos com controle de versão usando o Git. Ele permite que programadores, utilitários ou qualquer usuário cadastrado na plataforma contribuam em projetos privados e/ou Open Source de qualquer lugar do mundo. GitHub é amplamente utilizado por programadores para divulgação de seus trabalhos ou para que outros programadores contribuam com o projeto, além de promover fácil comunicação através de recursos que relatam problemas ou mesclam repositórios remotos

Gabarito: Correto

6. (CESPE – Ministério da Economia – 2020) O comando git clone permite baixar o repositório do GitHub para o computador do usuário.

Comentários: Pessoal, vejamos a definição do comando citado: git clone git@<caminho_do_arquivo>: Clonar um repositório remoto já existente. além disso, git-clone - Clona um repositório em um novo diretório.

Gabarito: Correto

7. (CESPE – Ministério da Economia – 2020) No GIT, o comando git pull é usado para enviar ao repositório a alteração que foi efetivada no computador local.

Comentários: Pessoal, na verdade o comando usado para enviar ao repositório a alteração que foi efetivada no computador local é git push.

Gabarito: Errado

8. (UFRN – UFRN – 2020) O git é um sistema de controle de versão muito utilizado em desenvolvimento de sistemas de software. Sobre o git, é correto afirmar:

a) O comando git push é utilizado para envio das alterações confirmadas no diretório local para o repositório remoto.
b) O comando git clone faz a cópia apenas dos arquivos, sendo as informações do repositório inicializadas como no comando git init.
c) O comando git add faz a confirmação das alterações de forma definitiva.
d) O comando git checkout faz uma atualização do diretório local com o diretório remoto.

Comentários: Pessoal vamos ver as definições dos comandos: git push envia arquivos/diretórios para o repositório remoto - Nosso gabarito! vejamos os demais comandos. git clone clona um repositório remoto. git add Adicionar um arquivo. git checkout desfaz alteração local.

Gabarito: Letra A

9. (UFC – UFC – 2019) Qual arquivo é necessário ser configurado para especificar intencionalmente que determinados arquivos não sejam rastreados (tracked) e que o Git deve ignorar no repositório Git local?

a) ignore.git
b).gitignore
c) git.ignore

d) gitignore.txt
e) .gitignore.txt

Comentários: O comando gitignore especifica arquivos não rastreados intencionalmente para ignorar. Arquivos já rastreados pelo Git não são afetados. Cada linha em um arquivo gitignore especifica um padrão. Portanto, o arquivo a ser configurado para especificar intencionalmente que determinados arquivos não sejam rastreados é o gitignore.

Gabarito: Letra B

10. (UFRN – Câmara de Parnamirim – 2019) A utilização do conceito de branches em sistemas de controle de versão permite ao desenvolvedor divergir da linha principal de desenvolvimento. Ao criar várias branches, o desenvolvedor deverá tomar cuidado para escolher a branch correta para iniciar qualquer modificação. Utilizando o sistema de versionamento de código GiT, para entrar em uma branch denominada mobile, utiliza-se o comando

a) git branch mobile
b) git checkout mobile
c) git status mobile
d) git change mobile

Comentários: Pessoal, o mais importante é saber o que cada comando faz: git-branch: Lista, cria ou exclui branches. git-checkout: Troque ramificações ou restaure os arquivos da árvore de trabalho - nosso gabarito. git-status: Mostra o status da árvore de trabalho. Por fim, git change não foi identificado.

Gabarito: Letra B

11. (Quadrix – CRM-PR – 2018) A ferramenta SVN (subversion) realiza o controle de versão de software por meio do uso da plataforma Mercurial.

Comentários: Mercurial e SVN são sistemas de controle de versão. Ademais, o SVN não usa a plataforma Mercurial.

Gabarito: Errado

12. (FCC – Prefeitura de São Luís - MA – 2018) Um Auditor Fiscal fez uma pesquisa na internet e obteve as seguintes informações:

Há vários critérios para escolher uma ferramenta para esta finalidade, como popularidade, eficácia, desempenho, adequação e simplicidade. Este tipo de ferramenta serve para resolver três problemas:
I. registrar a evolução do projeto;
II. possibilitar o trabalho em equipe;
III. criar e manter variações do projeto.

Tanto o Subversion, quanto o Git e o Mercurial atendem estas necessidades.

O Auditor estava pesquisando sobre ferramentas de

a) projeto e governança de portais corporativos.
b) controle de workflows e Business Process Management (BPM).
c) Gerenciamento Eletrônico de Documentos (GED) de projetos.
d) controle e gerenciamento de versão.
e) projetos de auditoria com base no PMBOK 5ª edição.

Comentários: Bem contextualizada a questão (a cara da FCC). Bom, há a descrição das características dos controle e gerenciamento de versão e a banca ainda solta dois sistemas muito comhecidos: Git e o Mercurial. Daí podemos marcar nosso gabarito na letra D.

Gabarito: Letra D

13. (CESPE – TRE TO – 2017) 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'

Comentários: O comando correto é: git checkout -b new_branch. Especificar -b faz com que um novo branch seja criado como se git-branch fosse chamado. Vejamos a definição de cada um deles. git log: Exibe o histórico. git clone: Clonar um repositório remoto. git init: Cria um novo repositório. git commit: Comita um arquivo.

Gabarito: Letra C

14. (UFPE – UFPE– 2017) Quando se usa o controle de versão através da ferramenta GIT, é possível interromper o fluxo de trabalho por meio da funcionalidade <STASH>. Pelo comando <git stash>, se faz possível:

a) criar uma ramificação (branching).
b) unificar ramificações (merging).
c) gerar uma solicitação de integração (pull request).
d) reverter a versão do código a uma versão específica (cherry pick).
e) exibir as diferenças entre duas versões quaisquer do código (diff).

Comentários: git-stash esconde as alterações em um diretório de trabalho sujo, ou seja, retira toda sujeira do seu diretório de trabalho. Use git stashquando quiser gravar o estado atual do diretório de trabalho e o índice, mas quiser voltar para um diretório de trabalho limpo. O comando salva suas modificações locais e reverte o diretório de trabalho para corresponder ao HEAD commit.

Gabarito: Letra A

15. (UFPE – UFPE – 2017) O GIT é um sistema de controle de versão distribuído, e também um gerenciamento de código fonte. Projetado e desenvolvido por Linus Torvalds, a ferramenta foi utilizada por muitos projetos ao redor do mundo. A respeito do Git, é incorreto afirmar que:

a) o comando 'git commit' grava as mudanças realizadas no repositório atual.
b) o comando 'git clone' copia um repositório existente em um novo diretório.
c) o comando 'git reset' modifica a HEAD atual para um estado específico.
d) o comando 'git init' pode reinicializar um repositório já criado.
e) o comando 'git bisect' divide o repositório atual em um ou mais repositórios diferentes.

Comentários: Pessoal, vejamos a descricao de cada comando: git commit: Grava alterações no repositório. git clone: Clona/copia um repositório em um novo diretório. git-reset: Redefine/modifica o HEAD atual para o estado especificado. git-init: Cria um repositório Git vazio ou reinicializa um existente. git bisect: Use busca binária para encontrar o commit que introduziu um bug. Como a banca pede a alternativa incorreta, nosso gabarito é a letra E.

Gabarito: Letra E

16. (UFPE – UFPE– 2017) A respeito de sistemas de controle de versão, assinale a alternativa correta.

a) O SVN pode ser considerado um sistema de controle de versão distribuído.
b) O GIT pode ser considerado um sistema de controle de versão centralizado.
c) Um conflito representa uma situação onde dois usuários modificam o mesmo arquivo em intervalos de tempo distintos.
d) A operação de mesclagem (merge, em inglês) consiste na junção automática de diferentes versões de um arquivo.
e) Uma cópia local sempre estará atualizada quando se usa o controle de versões GIT.

Comentários: Vamos comentar cada alternativa: a letra a está errada, porque na verdade, o SVN é centralizado. O Git é distribuído. Letra B está incorreta. Na verdade, um conflito representa uma situação onde dois usuários modificam o mesmo arquivo no mesmo intervalo de tempo. Por outro lado, a letra C está correta: A operação de mesclagem (merge, em inglês) consiste na junção automática de diferentes versões de um arquivo. A letra E está incorreta, na verdade é preciso executar o comando git pull ou o git fetch para atualizar.

Gabarito: Letra C

17. (VUNESP – FUNDUNESP– 2016) O Git, ao tratar os arquivos que devem sofrer o processo de controle de versões, classifica o estado desses arquivos em 3 categorias, definidas como

a) checked (verificado), tracked (acompanhado) e identified (identificado).
b) committed (consolidado), modified (modificado) e staged (preparado).
c) identified (identificado), ignored (ignorado) e committed (consolidado).
d) ignored (ignorado), ready (pronto) e staged (preparado).
e) ready (pronto), cloned (clonado) e modified (modificado).

Comentários: O Git tem três estados principais que seus arquivos podem estar: committed, modificado (modified) e preparado (staged).

Gabarito: Letra B

18. (BIO-RIO – Pref São Gonçalo – 2016) No que diz respeito às características das ferramentas de controle de versão SVN e GIT, analise as afirmativas a seguir.

I. SVN opera de forma centralizada, enquanto o GIT de forma distribuída.

II. SVN opera exclusivamente em distribuições Linux, enquanto o GIT exclusivamente em ambientes Windows.

III. SVN suporta a operação de commit de forma atômica, ou seja, se a operação for interrompida pelo meio ela é desconsiderada, como por exemplo, em situações de queda de energia, diferentemente do GIT.

Assinale a alternativa correta:

a) somente a afirmativa I está correta.
b) somente a afirmativa II está correta.
c) somente a afirmativa III está correta.
d) somente as afirmativas I e II estão corretas.
e) todas as afirmativas estão corretas.

Comentários: Novamente, os Sistemas Centralizados de Controle de Versão (CVCSs) são: CVS, Subversion (SVN) e Perforce. Já, Sistemas Distribuídos de Controle de Versão (DVCS) temos como exemplo Git, Mercurial, Bazaar ou Darcs. Sabendo disso, temos que o item I está correto, o item II está incorreto porque SVN e Git são multiplataforma. Ademais, ambos suportam commit atômico.

Gabarito: Letra A

19. (CESPE – STJ – 2015) O Git, sistema de controle de versões que mantém um histórico completo de todas as alterações, permite a recuperação das versões do projeto na busca de informações sobre o estado dos arquivos em versões anteriores.

Comentários: O Git é o sistema de controle de versão mais usado atualmente e está rapidamente se tornando o padrão para o controle de versão. Git é um sistema de controle de versão distribuído, o que significa que sua cópia local do código é um repositório completo de controle de versão. Esses repositórios locais totalmente funcionais facilitam o trabalho off-line ou remotamente. Você faz o commit do seu trabalho localmente e sincroniza sua cópia do repositório com a cópia no servidor. Esse paradigma difere do controle de versão centralizado, em que os clientes devem sincronizar o código com um servidor antes de criar novas versões de código. Ademais, o git permite a recuperação das versões do projeto na busca de informações sobre o estado dos arquivos em versões anteriores. Perfeita questão!

Gabarito: Correto

20. (CESPE – ANATAQ – 2014) As ferramentas de controle de versão Git e SVN oferecem o mesmo grau de confiabilidade no armazenamento das informações e são ambas implantadas conforme o conceito de sistemas de controle de versão distribuído.

Comentários: Em um DVCS (tais como Git, Mercurial, Bazaar ou Darcs), os clientes não apenas fazem cópias das últimas versões dos arquivos: eles são cópias completas do repositório. Assim, se um servidor falha, qualquer um dos repositórios dos clientes pode ser copiado de volta para o servidor para restaurá-lo. Cada checkout (resgate) é na prática um backup completo de todos os dados. Vimos que o Git é distribuído. Por outro lado, o SVN é centralizado, vejamos: Apache Subversion, também conhecido como Subversion, SVN representa o sistema de controle de versão centralizado mais popular do mercado. Com um sistema centralizado, todos os arquivos e dados históricos são armazenados em um servidor central. Os desenvolvedores podem confirmar suas alterações diretamente nesse repositório do servidor central.

▪

Gabarito: Errado

21. (CESPE – ANATEL – 2014) Os comandos da ferramenta Git são relativamente simples: para adicionar, por exemplo, um arquivo novo ao repositório no Git, basta utilizar o comando commit depois de efetuar o comando add.

Comentários: Pessoal, o comando para adicionar um arquivo novo ao repositório é o git add

Gabarito: Correto

Lista de Questões

1. (FCC – PGE-AM – 2022) Um Técnico utilizou corretamente um comando git para modificar a mensagem do commit mais recente, ou seja, o último commit feito por ele no projeto. Trata-se do comando git

   a) add merge.
   b) push.
   c) commit --amend.
   d) add message.
   e) checkout master.

2. (FGV – TJDFT – 2022) O analista Mateus configurou um pipeline CI/CD para o projeto TJApp no GitLab. O repositório de TJApp denomina-se TJAppRepo. Mateus precisou controlar o comportamento do pipeline de TJApp condicionando o início de sua execução aos eventos de push de tags para o TJAppRepo.

   Para aplicar essa condição ao pipeline de TJApp, Mateus precisou modificar o arquivo gitlab-ci.yml na raiz de TJAppRepo, adicionando uma regra na seção:

   a) default;
   b) include;
   c) stages;
   d) variables;
   e) workflow.

3. (FGV – TJ TO – 2022) O técnico em informática José está desenvolvendo o software TJTOPlugin com o apoio da ferramenta de versionamento Git. José criou o branch local pluginConnector e efetuou alguns commits neste branch, mas não replicou os commits em um repositório remoto.

   A fim de replicar os commits e criar o branch pluginConnector no repositório remoto origin, utilizando um único comando no terminal de comandos do sistema operacional, José deve executar o comando git com os argumentos:

   a) mv pluginConnector origin;
   b) diff pluginConnector origin;

c) merge origin/pluginConnector;
d) push origin pluginConnector;
e) remote add origin pluginConnector.

4. (UFRPE – UFRPE – 2022) O git é um sistema de controle de versão distribuído e utilizado amplamente pela comunidade de desenvolvimento de software. Esse sistema possui um conjunto de comandos utilizados para o versionamento de código. Dito isso, qual o comando utilizado para enviar as alterações do repositório local para o repositório remoto?

a) git commit
b) git push
c) git add
d) git pull
e) git send

5. (CESPE – MPE CE – 2020) GitHub é uma plataforma de hospedagem de código que permite realizar o controle de versão de software, de modo que várias pessoas contribuam simultaneamente no mesmo projeto, editando e criando novos arquivos, sem o risco de suas alterações serem sobrescritas.

6. (CESPE – Ministério da Economia – 2020) O comando git clone permite baixar o repositório do GitHub para o computador do usuário.

7. (CESPE – Ministério da Economia – 2020) No GIT, o comando git pull é usado para enviar ao repositório a alteração que foi efetivada no computador local.

8. (UFRN – UFRN – 2020) O git é um sistema de controle de versão muito utilizado em desenvolvimento de sistemas de software. Sobre o git, é correto afirmar:

a) O comando git push é utilizado para envio das alterações confirmadas no diretório local para o repositório remoto.
b) O comando git clone faz a cópia apenas dos arquivos, sendo as informações do repositório inicializadas como no comando git init.
c) O comando git add faz a confirmação das alterações de forma definitiva.
d) O comando git checkout faz uma atualização do diretório local com o diretório remoto.

9. (UFC – UFC – 2019) Qual arquivo é necessário ser configurado para especificar intencionalmente que determinados arquivos não sejam rastreados (tracked) e que o Git deve ignorar no repositório Git local?

a) ignore.git
b).gitignore
c) git.ignore
d) gitignore.txt
e) .gitignore.txt

10. (UFRN – Câmara de Parnamirim – 2019) A utilização do conceito de branches em sistemas de controle de versão permite ao desenvolvedor divergir da linha principal de desenvolvimento. Ao criar várias branches, o desenvolvedor deverá tomar cuidado para escolher a branch correta para iniciar qualquer modificação. Utilizando o sistema de versionamento de código GiT, para entrar em uma branch denominada mobile, utiliza-se o comando

a) git branch mobile
b) git checkout mobile
c) git status mobile
d) git change mobile

11. (Quadrix – CRM-PR – 2018) A ferramenta SVN (subversion) realiza o controle de versão de software por meio do uso da plataforma Mercurial.

12. (FCC – Prefeitura de São Luís - MA – 2018) Um Auditor Fiscal fez uma pesquisa na internet e obteve as seguintes informações:

Há vários critérios para escolher uma ferramenta para esta finalidade, como popularidade, eficácia, desempenho, adequação e simplicidade. Este tipo de ferramenta serve para resolver três problemas:
I. registrar a evolução do projeto;
II. possibilitar o trabalho em equipe;
III. criar e manter variações do projeto.

Tanto o Subversion, quanto o Git e o Mercurial atendem estas necessidades.

O Auditor estava pesquisando sobre ferramentas de

a) projeto e governança de portais corporativos.
b) controle de workflows e Business Process Management (BPM).
c) Gerenciamento Eletrônico de Documentos (GED) de projetos.
d) controle e gerenciamento de versão.
e) projetos de auditoria com base no PMBOK 5ª edição.

13. (CESPE – TRE TO – 2017) 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'

14. (UFPE – UFPE– 2017) Quando se usa o controle de versão através da ferramenta GIT, é possível interromper o fluxo de trabalho por meio da funcionalidade <STASH>. Pelo comando <git stash>, se faz possível:

a) criar uma ramificação (branching).
b) unificar ramificações (merging).
c) gerar uma solicitação de integração (pull request).
d) reverter a versão do código a uma versão específica (cherry pick).
e) exibir as diferenças entre duas versões quaisquer do código (diff).

15. (UFPE – UFPE – 2017) O GIT é um sistema de controle de versão distribuído, e também um gerenciamento de código fonte. Projetado e desenvolvido por Linus Torvalds, a ferramenta foi utilizada por muitos projetos ao redor do mundo. A respeito do Git, é **incorreto** afirmar que:

a) o comando 'git commit' grava as mudanças realizadas no repositório atual.
b) o comando 'git clone' copia um repositório existente em um novo diretório.
c) o comando 'git reset' modifica a HEAD atual para um estado específico.

d) o comando 'git init' pode reinicializar um repositório já criado.
e) o comando 'git bisect' divide o repositório atual em um ou mais repositórios diferentes.

16. (UFPE – UFPE– 2017) A respeito de sistemas de controle de versão, assinale a alternativa correta.

a) O SVN pode ser considerado um sistema de controle de versão distribuído.
b) O GIT pode ser considerado um sistema de controle de versão centralizado.
c) Um conflito representa uma situação onde dois usuários modificam o mesmo arquivo em intervalos de tempo distintos.
d) A operação de mesclagem (merge, em inglês) consiste na junção automática de diferentes versões de um arquivo.
e) Uma cópia local sempre estará atualizada quando se usa o controle de versões GIT.

17. (VUNESP – FUNDUNESP– 2016) O Git, ao tratar os arquivos que devem sofrer o processo de controle de versões, classifica o estado desses arquivos em 3 categorias, definidas como

a) checked (verificado), tracked (acompanhado) e identified (identificado).
b) committed (consolidado), modified (modificado) e staged (preparado).
c) identified (identificado), ignored (ignorado) e committed (consolidado).
d) ignored (ignorado), ready (pronto) e staged (preparado).
e) ready (pronto), cloned (clonado) e modified (modificado).

18. (BIO-RIO – Pref São Gonçalo – 2016) No que diz respeito às características das ferramentas de controle de versão SVN e GIT, analise as afirmativas a seguir.

I. SVN opera de forma centralizada, enquanto o GIT de forma distribuída.

II. SVN opera exclusivamente em distribuições Linux, enquanto o GIT exclusivamente em ambientes Windows.

III. SVN suporta a operação de commit de forma atômica, ou seja, se a operação for interrompida pelo meio ela é desconsiderada, como por exemplo, em situações de queda de energia, diferentemente do GIT.

Assinale a alternativa correta:

a) somente a afirmativa I está correta.
b) somente a afirmativa II está correta.
c) somente a afirmativa III está correta.
d) somente as afirmativas I e II estão corretas.
e) todas as afirmativas estão corretas.

19. (CESPE – STJ – 2015) O Git, sistema de controle de versões que mantém um histórico completo de todas as alterações, permite a recuperação das versões do projeto na busca de informações sobre o estado dos arquivos em versões anteriores.

20. (CESPE – ANATAQ – 2014) As ferramentas de controle de versão Git e SVN oferecem o mesmo grau de confiabilidade no armazenamento das informações e são ambas implantadas conforme o conceito de sistemas de controle de versão distribuído.

21. (CESPE – ANATEL – 2014) Os comandos da ferramenta Git são relativamente simples: para adicionar, por exemplo, um arquivo novo ao repositório no Git, basta utilizar o comando commit depois de efetuar o comando add.

GABARITO

1. LETRA C
2. LETRA E
3. LETRA D
4. LETRA B
5. CORRETO
6. CORRETO
7. ERRADO
8. LETRA A
9. LETRA B
10. LETRA B
11. ERRADO
12. LETRA D
13. LETRA C
14. LETRA A
15. LETRA E
16. LETRA C
17. LETRA B
18. LETRA A
19. CORRETO
20. ERRADO
21. CORRETO

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.