1. O que é o Git e qual é a sua principal função no controle de versão de código-fonte?

O Git é um projeto de código aberto maduro e com manutenção ativa desenvolvido em 2005 por Linus Torvalds, o famoso criador do kernel do sistema operacional Linux. Um número impressionante de projetos de software depende do Git para controle de versão, incluindo projetos comerciais e de código-fonte aberto.

**2)** Como podemos criar um novo repositório Git em um projeto existente no seu sistema local?

Para criar um novo repositório, você vai usar o comando git init. git init é um comando único que você usa durante a configuração inicial de um novo repositório. A execução desse comando cria um novo subdiretório. git no diretório de trabalho atual. Essa ação também vai criar uma ramificação principal.

**3)** Qual a diferença entre "git add" e "git commit" no contexto do Git?

Os instantâneos com commit podem ser considerados versões "seguras" de um projeto, o Git nunca os altera, a menos que você peça a ele. Antes da execução de git commit, o comando [git add](https://www.atlassian.com/br/git/tutorials/saving-changes) é usado para promover ou "preparar" alterações no projeto que são armazenadas em uma confirmação. Estes dois comandos, git commit e git add, estão entre os mais usados.

**4)** Explique o propósito do comando "git push" e quando você o utilizaria.

O Git push é um comando que possibilita que as alterações da sua máquina local sejam enviadas para uma máquina remota. Mas, vamos entender isso melhor:

* Imagine que você está desenvolvendo um projeto com uma equipe. O projeto que vocês estão trabalhando está no GitHub, mas você vai desenvolvê-lo no seu computador. Como fazer para “enviar” seu desenvolvimento para o GitHub?
* Para enviar suas alterações, você usará o comando git push. Antes dele, evidentemente, temos alguns outros comandos, mas é o git push que pegará tudo o que você fez e colocará em um local onde qualquer pessoa colaboradora possa ver e pegar suas alterações.
* Basicamente você desenvolve no repositório local, ou seja, seu computador. Quando realiza o comando git push, você envia essas alterações do seu repositório local para um repositório remoto e todas as pessoas que possuem acesso ao repositório remoto poderão acessar as suas alterações.

**5)** O que são branches no Git e por que são úteis em um projeto de desenvolvimento?

Um branch no Git é simplesmente um ponteiro móvel para um desses commits. O nome do branch padrão no Git é master. Conforme você começa a fazer commits, você recebe um branch master que aponta para o último commit que você fez. Cada vez que você faz um novo commit, ele avança automaticamente.

**6)** Como podemos verificar o status atual do repositório local no Git?

Inicialmente se estamos trabalhando em um arquivo local e temos um repositorio github teriamos que saber se a versão local e do repositório estão alinhadas. Para isso usamos: git status.

**7)** Explique o que é um conflito de merge no Git e como você resolveria esse problema.

É comum que surjam conflitos quando duas pessoas alteram as mesmas linhas no mesmo arquivo. Ou então quando algum desenvolvedor exclui arquivos enquanto outra pessoa faz alterações. Nesses casos, o Git não pode determinar qual está correto. Os conflitos afetam apenas o desenvolvedor que conduz o merge e o resto da equipe não fica ciente deles. O Git marca os arquivos em conflito e interrompe o processo de merge. A partir daí, é responsabilidade dos desenvolvedores resolver o conflito.

**8)** O que é o GitHub e como ele se diferencia do Git em termos de funcionalidades?

O Github é uma “rede social dev” em que é possível armazenar e compartilhar projetos de desenvolvimento de software.

O Git é um sistema de controle de versão de arquivos; em outras palavras, é responsável por guardar o histórico de alterações sempre que alguém modificar algum arquivo que está sendo monitorado por ele.

**9)** Qual a diferença entre um repositório público e privado no GitHub?

Com o GitHub Free para contas pessoais e de organizações, você pode trabalhar com colaboradores ilimitados em repositórios públicos ilimitados, com um conjunto completo de recursos, ou em repositórios privados ilimitados, com um conjunto limitado de recursos.

**10)** Como podemos realizar o processo de clonagem (clone) de um repositório do GitHub para o seu computador local?

git clone <url>

**11)** O que são Pull Requests no GitHub e qual é o seu propósito no desenvolvimento

colaborativo?

Os modelos de pull request permitem personalizar e padronizar as informações que você gostaria de incluir quando alguém cria uma pull request em seu repositório.

**12)** Como podemos adicionar um arquivo ao arquivo .gitignore e qual é a sua finalidade?

Abra o arquivo. gitignore no seu editor de texto favorito. Se o arquivo não existir, você pode criá-lo na raiz do seu repositório.

Adicione o nome do arquivo (ou padrão de nome) que você deseja ignorar em uma nova linha no arquivo. gitignore.

**13)** Explique o que é um fork no GitHub e em que situações você pode querer realizar um fork de um repositório.

Os forks geralmente são usados para iterar ideias ou alterações antes de serem propostas de volta para o repositório upstream, como em projetos código aberto ou quando um usuário não tem acesso de gravação ao repositório upstream. Para saber mais, confira "Trabalhar com bifurcações".

**14)** Como podemos visualizar o histórico de commits em um repositório Git?

A ferramenta mais básica e poderosa para fazer isso é o comando git log . Por padrão, sem argumentos, git log lista os commits feitos neste repositório em ordem cronológica inversa; isto é, o commit mais recente aparece primeiro.

**15)** Qual é a função do comando "git pull" e quando ele deve ser utilizado no desenvolvimento colaborativo com o Git e GitHub?

O comando git pull serve para recuperar e baixar o conteúdo de um repositório remoto e atualizar o repositório local assim que ele for baixado. Isso é muito útil em fluxos de trabalho de colaboração que precisam mesclar alterações upstream remotas no repositório local.

<https://blog.betrybe.com/git/git-push/>

<https://git-scm.com/book/pt-br/v2/Branches-no-Git-Branches-em-poucas-palavras#:~:text=Um%20branch%20no%20Git%20é,novo%20commit%2C%20ele%20avança%20automaticamente>.