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

De longe, o sistema de controle de versão moderno mais usado no mundo hoje é o Git. 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. Os desenvolvedores que trabalharam com o Git estão bem representados no pool de talentos de desenvolvimento de software disponíveis e funcionam bem em uma ampla variedade de sistemas operacionais e IDEs (Ambientes de Desenvolvimento Integrado).

Tendo uma arquitetura distribuída, o Git é um exemplo de DVCS (portanto, Sistema de Controle de Versão Distribuído). Em vez de ter apenas um único local para o histórico completo da versão do software, como é comum em sistemas de controle de versão outrora populares como CVS ou Subversion (também conhecido como SVN), no Git, a cópia de trabalho de todo desenvolvedor do código também é um repositório que pode conter o histórico completo de todas as alterações.

<https://www.atlassian.com/br/git/tutorials/what-is-git#:~:text=O%20Git%20%C3%A9%20um%20projeto,e%20de%20c%C3%B3digo%2Dfonte%20aberto.>

***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***

[***https://www.atlassian.com/br/git/tutorials/setting-up-a-repository#:~:text=Para%20criar%20um%20novo%20reposit%C3%B3rio,comando%20cria%20um%20novo%20subdiret%C3%B3rio%20.***](https://www.atlassian.com/br/git/tutorials/setting-up-a-repository#:~:text=Para%20criar%20um%20novo%20reposit%C3%B3rio,comando%20cria%20um%20novo%20subdiret%C3%B3rio%20.)

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

***Olá! git add significa Adicionar, ou seja, você pode adicionar um arquivo qualquer. Exemplo: git add nomedomeuarquivoaqui.***

***Já em git commit é muito importante, pois indica as alterações que você fez no seu projeto. Ou seja, o commit é a realização de um conjunto de mudanças, alterações que você realizou em seu projeto.***

[***https://pt.stackoverflow.com/questions/572430/comandos-git-git-add-e-git-commit#:~:text=git%20add%20significa%20Adicionar%2C%20ou,voc%C3%AA%20realizou%20em%20seu%20projeto.***](https://pt.stackoverflow.com/questions/572430/comandos-git-git-add-e-git-commit#:~:text=git%20add%20significa%20Adicionar%2C%20ou,voc%C3%AA%20realizou%20em%20seu%20projeto.)

***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?***

[***https://blog.betrybe.com/git/git-push/***](https://blog.betrybe.com/git/git-push/)

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

***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.***

[***https://git-scm.com/book/pt-br/v2/Branches-no-Git-Branches-em-poucas-palavras#:~:text=Um%20branch%20no%20Git%20%C3%A9,novo%20commit%2C%20ele%20avan%C3%A7a%20automaticamente.***](https://git-scm.com/book/pt-br/v2/Branches-no-Git-Branches-em-poucas-palavras#:~:text=Um%20branch%20no%20Git%20%C3%A9,novo%20commit%2C%20ele%20avan%C3%A7a%20automaticamente.)

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

**O comando git status exibe as condições do diretório de trabalho e da área de staging. Ele permite que você veja quais alterações foram despreparadas, quais não foram e quais arquivos não estão sendo monitorados pelo Git. Os resultados de status não exibem qualquer informação sobre o histórico de projetos que recebeu commit. Para tal, você precisa usar o** [**git log**](https://www.atlassian.com/git/tutorials/inspecting-a-repository/git-log)

[***https://www.atlassian.com/br/git/tutorials/inspecting-a-repository***](https://www.atlassian.com/br/git/tutorials/inspecting-a-repository)

***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.***

## ***Tipos de conflitos de merge***

***O merge pode entrar em estado de conflito em dois momentos diferentes. No começo ou durante o processo de merge. A seguir, a gente discute como enfrentar esses cenários de conflito.***

[***https://www.atlassian.com/br/git/tutorials/using-branches/merge-conflicts***](https://www.atlassian.com/br/git/tutorials/using-branches/merge-conflicts)

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

***Em resumo, o Git é uma ferramenta de controle de versão local, enquanto o GitHub é uma plataforma online que permite hospedar projetos Git e colaborar com outras pessoas. Ambos são muito úteis para o trabalho de desenvolvedores, e é sempre bom conhecer bem cada uma dessas ferramentas para aproveitá-las ao máximo!***

[***https://www.dio.me/articles/entendendo-as-diferencas-entre-git-e-github***](https://www.dio.me/articles/entendendo-as-diferencas-entre-git-e-github)

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

***É possível restringir quem tem acesso a um repositório escolhendo a visibilidade de um repositório: público ou privado.***

***Ao criar um repositório, você pode optar por tornar o repositório público ou privado. Os repositórios em organizações que usam GitHub Enterprise Cloud e são propriedade de uma conta corporativa também podem ser criados com visibilidade interna. Para obter mais informações, confira*** [***a documentação do GitHub Enterprise Cloud***](https://docs.github.com/pt/enterprise-cloud@latest/repositories/creating-and-managing-repositories/about-repositories)***.***

* ***Os repositórios públicos podem ser acessados por todos na Internet.***
* ***Os repositórios só podem ser acessados por você, pelas pessoas com as quais você compartilha explicitamente o acesso e, para repositórios da organização, por determinados integrantes da organização.***

***Os proprietários da organização sempre têm acesso a todos os repositórios criados em uma organização. Para obter mais informações, confira "***[***Funções de repositório para uma organização***](https://docs.github.com/pt/organizations/managing-user-access-to-your-organizations-repositories/managing-repository-roles/repository-roles-for-an-organization)***".***

***As pessoas com permissões de administrador para um repositório podem alterar a visibilidade de um repositório existente. Para obter mais informações, confira "***[***Definir a visibilidade do repositório***](https://docs.github.com/pt/repositories/managing-your-repositorys-settings-and-features/managing-repository-settings/setting-repository-visibility)***".***

[***https://docs.github.com/pt/repositories/creating-and-managing-repositories/about-repositories***](https://docs.github.com/pt/repositories/creating-and-managing-repositories/about-repositories)

***10. Como podemos realizar o processo de clonagem (clone) de um repositório do GitHub para o***

***seu computador local?***

***Como clonar um repositório do GitHub?***

***Abra TerminalTerminalGit Bash. Crie um clone bare do repositório. Substitua o exemplo de nome de usuário pelo nome da pessoa ou da organização a quem pertence o repositório e substitua o exemplo de nome de repositório pelo nome do repositório que você deseja duplicar.***

[***https://docs.github.com/pt/repositories/creating-and-managing-repositories/duplicating-a-repository***](https://docs.github.com/pt/repositories/creating-and-managing-repositories/duplicating-a-repository)

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

***colaborativo?***

***Os pull requests permitem que você proponha alterações nos projetos, forneça feedback e revisões e mescle alterações em projetos. Para obter mais informações, confira "***[***Sobre solicitação de pull***](https://docs.github.com/pt/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests)***".***

***Quando alguém cria uma solicitação de pull, ele faz alterações em um "branch head" e sugere essas alterações em um "branch base", como main. Em GitHub Desktop, você pode abrir (ou "fazer check-out") o branch head de uma solicitação de pull para exibir as alterações que um colaborador está sugerindo. Por exemplo, você pode ver um histórico dos commits que o colaborador fez e ver quais arquivos os commits modificaram, adicionaram ou excluíram.***

[***https://docs.github.com/pt/desktop/working-with-your-remote-repository-on-github-or-github-enterprise/viewing-a-pull-request-in-github-desktop***](https://docs.github.com/pt/desktop/working-with-your-remote-repository-on-github-or-github-enterprise/viewing-a-pull-request-in-github-desktop)

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

***Você pode criar um arquivo .gitignore no diretório raiz do repositório para informar o Git de quais arquivos e diretórios devem ser ignorados durante um commit. Para compartilhar as regras de ignorar com outros usuários que clonam o repositório, faça commit do arquivo .gitignore no repositório.***

***O GitHub mantém uma lista oficial de arquivos .gitignore recomendados para vários sistemas operacionais, ambientes e linguagens populares no repositório público "github/gitignore". Use também o gitignore.io para criar um arquivo .gitignore para seu sistema operacional, sua linguagem de programação ou seu IDE. Para obter mais informações, confira "***[***github/gitignore***](https://github.com/github/gitignore)***" e o site "***[***gitignore.io***](https://www.gitignore.io/)***".***

[***https://docs.github.com/pt/get-started/getting-started-with-git/ignoring-files***](https://docs.github.com/pt/get-started/getting-started-with-git/ignoring-files)

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

***Os forks permitem fazer alterações em um projeto sem afetar o repositório original, também conhecido como repositório "upstream". Após criar fork em um repositório, você pode buscar atualizações do repositório upstream para manter seu fork atualizado e propor alterações do fork para o repositório upstream com solicitações de pull. Um fork pode ser propriedade de uma conta pessoal ou de uma organização.***

[***https://docs.github.com/pt/pull-requests/collaborating-with-pull-requests/working-with-forks/about-forks***](https://docs.github.com/pt/pull-requests/collaborating-with-pull-requests/working-with-forks/about-forks)

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

***Por padrão, sem argumentos, git log lista os commits feitos neste repositório em ordem cronológica inversa; isto é, o commit mais recente aparece primeiro. Como você pode ver, esse comando lista cada commit com o seu checksum SHA-1, o nome e email do autor, data de inserção, e a mensagem do commit.***

***Está disponível um enorme número e variedade de opções para o comando git log a fim de lhe mostrar exatamente aquilo pelo que está procurando. Aqui, vamos mostrar a você algumas das mais populares.***

***Uma das opções que mais ajuda é -p, que mostra as diferenças introduzidas em cada commit. Você pode também usar -2, que lista no retorno apenas os dois últimos itens:***

***$ git log -p -2***

***commit ca82a6dff817ec66f44342007202690a93763949***

***Author: Scott Chacon <schacon@gee-mail.com>***

***Date: Mon Mar 17 21:52:11 2008 -0700***

***changed the version number***

***diff --git a/Rakefile b/Rakefile***

***index a874b73..8f94139 100644***

***--- a/Rakefile***

***+++ b/Rakefile***

***@@ -5,7 +5,7 @@ require 'rake/gempackagetask'***

***spec = Gem::Specification.new do |s|***

***s.platform = Gem::Platform::RUBY***

***s.name = "simplegit"***

***- s.version = "0.1.0"***

***+ s.version = "0.1.1"***

***s.author = "Scott Chacon"***

***s.email = "schacon@gee-mail.com"***

***s.summary = "A simple gem for using Git in Ruby code."***

***commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7***

***Author: Scott Chacon <schacon@gee-mail.com>***

***Date: Sat Mar 15 16:40:33 2008 -0700***

***removed unnecessary test***

***diff --git a/lib/simplegit.rb b/lib/simplegit.rb***

***index a0a60ae..47c6340 100644***

***--- a/lib/simplegit.rb***

***+++ b/lib/simplegit.rb***

***@@ -18,8 +18,3 @@ class SimpleGit***

***end***

***end***

***-***

***-if $0 == \_\_FILE\_\_***

***- git = SimpleGit.new***

***- puts git.show***

***-end***

***\ No newline at end of file***

[***https://git-scm.com/book/pt-br/v2/Fundamentos-de-Git-Vendo-o-hist%C3%B3rico-de-Commits#:~:text=A%20ferramenta%20mais%20b%C3%A1sica%20e,%C3%A9%20o%20comando%20git%20log%20.&text=Por%20padr%C3%A3o%2C%20sem%20argumentos%2C%20git,commit%20mais%20recente%20aparece%20primeiro.***](https://git-scm.com/book/pt-br/v2/Fundamentos-de-Git-Vendo-o-hist%C3%B3rico-de-Commits#:~:text=A%20ferramenta%20mais%20b%C3%A1sica%20e,%C3%A9%20o%20comando%20git%20log%20.&text=Por%20padr%C3%A3o%2C%20sem%20argumentos%2C%20git,commit%20mais%20recente%20aparece%20primeiro.)

***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 é usado para buscar e baixar conteúdo de repositórios remotos e fazer a atualização imediata ao repositório local para que os conteúdos sejam iguais. Fazer o merge de alterações upstream remotas no repositório local é algo comum em fluxos de trabalho de colaboração baseados em Git. O comando git pull é uma combinação de outros dois comandos,*** [***git fetch***](https://www.atlassian.com/br/git/tutorials/syncing/git-fetch) ***seguido por*** [***git merge***](https://www.atlassian.com/br/git/tutorials/using-branches/git-merge)***. No primeiro estágio da operação, o git pull executa o comando git fetch, que abrange a ramificação local para qual a HEAD aponta. Quando o conteúdo é baixado, o git pull insere o fluxo de trabalho de merge. O commit de merge é criado e a HEAD é atualizada para apontar o novo commit.***

[***https://www.atlassian.com/br/git/tutorials/syncing/git-pull#:~:text=O%20comando%20git%20pull%20%C3%A9,de%20colabora%C3%A7%C3%A3o%20baseados%20em%20Git.***](https://www.atlassian.com/br/git/tutorials/syncing/git-pull#:~:text=O%20comando%20git%20pull%20%C3%A9,de%20colabora%C3%A7%C3%A3o%20baseados%20em%20Git.)