# Git (exercícios)

### Responda as questões abaixo e, para cada uma delas, exercite os comandos do git correspondentes (referenciados). Lembre-se de que o “interessante” não é exatamente o conjunto de respostas, mas o processo de obtê-las e a experiência obtida com a execução dos comandos.

1. Qual o comando para obter a versão instalada do Git?
2. Qual o efeito da execução de cada um dos comandos abaixo?
   1. **git help**
   2. **git help reset**
   3. **git init**
   4. **git add --all**
   5. **git add -u**
   6. **git config -l**
   7. **git mv a.txt b.txt**
   8. **git reset --hard**
   9. **git log -27**
3. O fluxo “clássico” de interação com o Git é algo como “alterar um ou mais arquivos”, “acrescentar essas mudanças para serem contemplados no próximo commit” e, finalmente, executar um “commit”. Quais os comandos necessários para realizar os dois últimos “passos” desse fluxo?
4. Qual o comando deve ser executado para identificar o que foi alterado desde o último “commit”?
5. Em um dado repositório, arquivos simplesmente copiados para lá, ou seja, *untracked*, podem ser exibidos/identificados com que comando?
6. Qual o comando para efetuar um *commit*?
7. Qual o comando que devemos empregar para descartar mudanças ocorridas no arquivo teste.txt, por exemplo?
8. O que deve ser feito para que um determinado diretório do seu repositório seja ignorado pelo Git? Faça uma busca por **.gitignore**.
9. O que acontece se o seu repositório local for acidentalmente removido?
10. Como clonar um repositório remoto?
11. Em alguns cenários **git log** pode produzir extensos resultados. Se houver interesse em visualizar o histórico de um repositório, onde cada mudança é fornecida exatamente em uma única linha, qual o comando que deve ser empregado?
12. Em qual arquivo o Git armazena informações de configuração empregadas por usuário?
13. Qual o comando para criar um repositório local?
14. Qual o nome do diretório criado pelo Git quando se executa o comando **git init**?
15. Qual o comando para adicionar todos os arquivos modificados? (Aqueles para os quais **git status** identificam como **modified**?)
16. O Git faz uso do valor de *hash* conhecido por SHA1. O que isto significa? Qual o propósito? O que é SHA1?
17. Qual a palavra para indicar o último *commit* em vez do valor de *hash* SHA1 correspondente?
18. Quando se cria dois arquivos usando um editor de texto qualquer e, na sequência, executamos o comando **git add -u**, os dois arquivos criados passam de *untracked* para *new file*?
19. Qual o efeito da execução dos dois comandos abaixo, nesta ordem, em um dado repositório?  
    **git reset --soft HEAD~1  
    git reset --hard**
20. Após o emprego de um ambiente integrado de desenvolvimento (IDE), é comum a criação de arquivos e diretórios. Qual o comando que podemos empregar para remover arquivos e diretórios *untracked*?
21. Qual o nome do arquivo no qual podemos inserir a indicação para o Git de arquivos e diretórios a serem ignorados?
22. Quando se cria o arquivo *MinhaClasse.class* em um dado diretório e desejamos que arquivos com a extensão .class, como neste caso, sejam ignorados por todos os membros de uma equipe que estão contribuindo com um dado projeto, como devemos proceder?
23. jQuery é uma famosa biblioteca em JavaScript. Consulte detalhes em <http://jquery.com>. O repositório correspondente encontra-se em <https://github.com/jquery/jquery.git>. Faça o clone deste repositório.
24. No repositório **jqueryrepo**, criado no passo anterior, qual o efeito do comando   
    **git shortlog -sne**?
25. No repositório **jqueryrepo**, qual o efeito de **git remote -v**?
26. Um repositório Git pode ser etiquetado ao longo do tempo. Ou seja, *commits* específicos podem ser “marcados” ou “etiquetados” para facilitar referências posteriores. Para listar todas as “etiquetas” (*tags*) estabelecidas para um dado repositório, qual comando deve ser executado?
27. Caso um dato repositório retorne muitas “marcas” ou “etiquetas” para o comando **git tag**, como retornar apenas aquelas que atendem a determinado padrão, por exemplo, iniciadas por 2.0?
28. Qual o efeito do comando **git tag -a 3.4-gold -m “minha versão ouro”**?
29. Após executado o comando acima, qual o efeito de **git show 3.4-gold**?
30. O que o comando **git push origin 3.4-gold** teria como efeito?
31. Após executar um *commit,* qual o efeito de **git commit --amend**?
32. Após executar **git add x.txt**, qual o efeito de **git reset HEAD x.txt**?
33. Após alterar o conteúdo de um arquivo *committed* em passo anterior, qual o efeito do comando **git checkout -- a.txt**?
34. Qual a diferença entre os comandos **git reset HEAD a.txt** e **git checkout -- a.txt**?
35. Veja como interpretar o resultado de git diff [aqui](https://medium.com/therobinkim/how-to-read-a-git-diff-6c87a9dc47c5). Execute, em um dos seus projetos, o comando **git diff HEAD~1 HEAD** e certifique-se de que entende o resultado apresentado.