Leitura de Software - Engenharia de software - UFG

Aluno: Guilherme Alves Rosa e Silva

Lista 1

1. git version  
   1. Lista as configurações.;
   2. Move ou renomeia arquivos;
   3. Desfaz todas as modificações retornando para o último commit;
   4. Mostra os ultimos 27 commits;
   5. Exibe informações de ajuda sobre o Git;
   6. Exibe ajuda sobre o comando “reset”;
   7. Adiciona todos os arquivos arquivos modificados;
   8. Adiciona apenas os arquivos modificados rastreados pelo git
2. git merge e git push
3. git log
4. git ls-files . --exclude-standard --others
5. git commit
6. git checkout teste.txt
7. Adicionando o nome do diretório no .gitgnore seguido de barra, por exemplo: “pasta/”
8. Você perderá as modificações não enviadas por um push para o repositório, podendo baixar novamente aquelas versões presentes no repositório.
9. “git clone” + o url
10. git log --oneline
11. Arquivo “.gitconfig”
12. git init
13. .git
14. git add -u
15. Significa que as trocas de informação no git são validadas por um hash SHA1, que é uma função hash que gera uma chave, que serve para validar a consistência dos dados transmitidos.
16. git log -1
17. Não, pois **git add -u** não adiciona untrackeds.
18. Irá voltar para o último commit, as mudanças no entanto se manterão. Em seguida retorna para o commit anterior e remove completamente as mudanças.
19. git clean -df
20. gitignore
21. Deve se adicionar a extensão no gitignore para que ela seja ignorada

|  |
| --- |
| 1714 John Resig <jeresig@gmail.com>  657 Timmy Willison <4timmywil@gmail.com>  579 Dave Methvin <dave.methvin@gmail.com>  336 Jörn Zaefferer <joern.zaefferer@gmail.com>  332 Julian Aubourg <aubourg.julian@gmail.com>  315 Rick Waldron <waldron.rick@gmail.com>  267 Oleg Gaidarenko <markelog@gmail.com>  262 Richard Gibson <richard.gibson@gmail.com>  250 Brandon Aaron <brandon.aaron@gmail.com>  230 Michał Gołębiowski-Owczarek <m.goleb@gmail.com>  200 Ariel Flesler <aflesler@gmail.com>  85 Mike Sherov <mike.sherov@gmail.com>  71 Colin Snover <github.com@zetafleet.com>  67 David Serduke <davidserduke@gmail.com>  59 Yehuda Katz <wycats@gmail.com>  55 Corey Frang <gnarf37@gmail.com>  47 Louis-Rémi Babé <lrbabe@gmail.com>  35 Anton Matzneller <obhvsbypqghgc@gmail.com>  34 Scott González <scott.gonzalez@gmail.com>  27 Gilles van den Hoven <gilles0181@gmail.com>  25 Timo Tijhof <krinklemail@gmail.com>  22 Robert Katić <robert.katic@gmail.com>  17 Dan Heberden <danheberden@gmail.com>  15 Carl Fürstenberg <azatoth@gmail.com>  12 Ben Alman <cowboy@rj3.net>  12 Jason Bedard <jason+github@jbedard.ca>  12 Karl Swedberg <kswedberg@gmail.com>  11 Jordan Boesch <jboesch26@gmail.com>  11 Mike Alsup <malsup@gmail.com> |



|  |
| --- |
| origin https://github.com/jquery/jquery.git (fetch)  origin https://github.com/jquery/jquery.git (push) |

1. git tag -l
2. git tag --points-at 2.0
3. Cria uma tag “3.4-gold”.
4. Exibe a mensagem “minha versão ouro”, contida nesta tag.
5. Mostra a tag e os objetos a ela referenciados.
6. Edita a mensagem do último commit.
7. Retira o arquivo da lista para commit.
8. Retorna o arquivo a.txt para a ultima versão que foi feita o commit.
9. Git reset remove da lista para commit, enquanto o git checkout restaura a ultima versão feita commit.

Lista 2

1. Feito
2. git push --tags
3. Master
4. Cria um branch com esse nome
5. git branch <nome do branch> HEAD~<numero do commit>  
   Ou git branch <nome do branch> <hash do commit>
6. Cria um branch com o nome erro1234
7. git checkout -b experimento2
8. Retorna:

|  |
| --- |
| b2 \*b1 |

1. Coloca o novobranch como branch corrente
2. Deletar o branch teste, se ele já foi merged

|  |
| --- |
| git branch novobranch git checkout -b novobranch git add Teste.txt  git commit  git checkout master  git merge novobranch |