UTFPR **Especialização em Desenvolvimento Web com *Frameworks* Modernos**

WEB11 — Versionamento de código com Git **Lista de exercícios 02**
2022

Aluno: Kauan Marques de Moraes Polydoro **Nota:**

1. Julgue as afirmações a seguir, prestando bastante **atenção aos detalhes**:
(marque V para verdadeiro, F para falso; é recomendado testar cada comando)(/ 7)

(V) Dentre os comandos **git add .**, **git log**, **git checkout a1b2c3d4e5** e **git init**, **apenas** o último **faz sentido** em um diretório **sem** subdiretório **.git/**
(F) O diretório **.git/** é produzido em resposta ao **git checkout**
(V) Após efetuar alterações no *working directory*, pode-se utilizar os comandos **git add .** e **git commit 'Mensagem de commit'** para criar um novo *commit*
(V) São consideradas "alterações" a inclusão, a exclusão e a modificação de arquivos no *working directory*
(V) O índice (*index*) pode ser entendido como a lista de modificações a compor o próximo *commit*
(V) As modificações, quando exibidas no **git status**, podem representar tanto entradas do índice quanto alterações não adicionadas para *commit*
(F) Uma vez que alterações forem adicionadas ao índice, não podem mais ser removidas (pois essa é justamente a função do Git, criar *snapshots* do código)

2. Julgue as alternativas a seguir quanto à finalidade de cada comando:
(marque V para verdadeiro, F para falso) (/ 6)

(V) **clone** obtém uma cópia do repositório especificado
(F) **checkout** gera uma visualização do grafo do repositório
(V) **log** exibe **por padrão** uma representação linear do histórico do repositório
(V) **gitk** exibe uma visualização gráfica do repositório
(V) **init** transforma, por consequência, um diretório comum em um repositório Git
(F) **commit inclui** arquivos no índice de *commit*

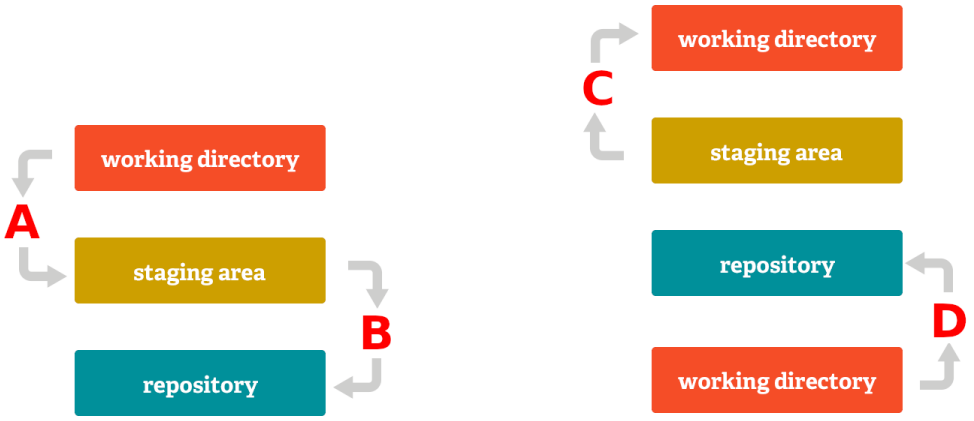
3. Liste, em ordem, as ações e os comandos necessários para transformar um diretório comum já existente em um repositório Git, criar um ***commit* inicial**, efetuar modificações nos arquivos, salvar as alterações na forma de um **novo *commit*** (total de 2 *commits*).
Atenção: as **mensagens** de ambos os *commits* devem ser inseridas em editor de texto, não via *flag* no comando de *commit*.
(pressuponha Git já instalado, configurado e pronto para uso) (/ 5)

1. git init
2. git add .
3. git commit
4. Inserir a mensagem principal do commit na primeira linha, inserir resumo do commit nas demais linhas
5. Efetuar modificações nos arquivos já existentes
6. git add .
7. git commit

8. Inserir a mensagem principal do commit na primeira linha, inserir resumo do commit nas demais linhas

4. Indique a alternativa que preenche corretamente as lacunas (considere apenas arquivos **rastreados**):

(/ 2)



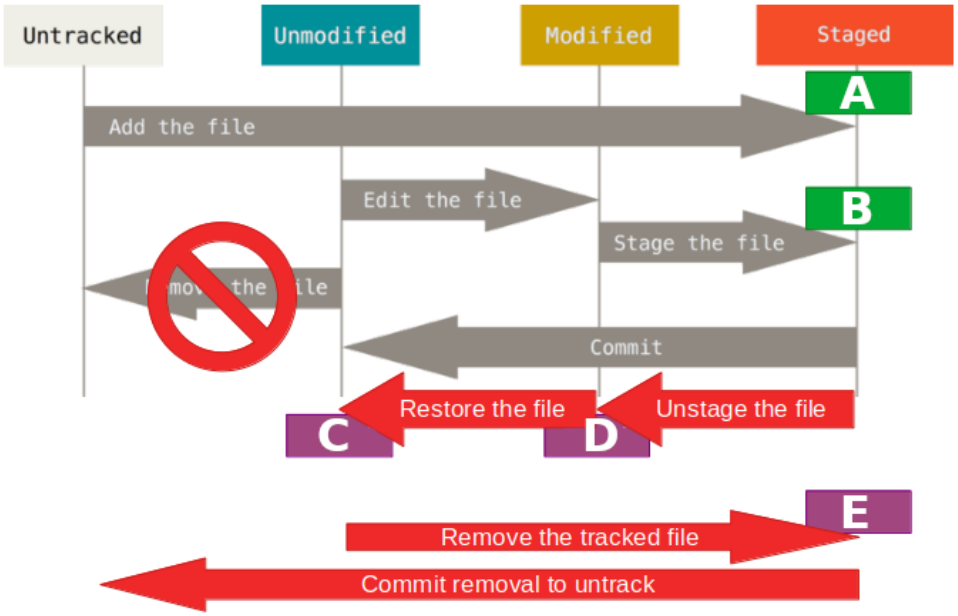
a)	A = **init**	B = **add**	C = **reset**	D = **checkout**
b)	A = **add**	B = **commit**	C = **remove**	D = **restore**
c)	A = **add**	B = **commit**	C = **reset**	D = **checkout**
d)	A = **add**	B = **reset**	C = **remove**	D = **checkout**
e)	A = **init**	B = **commit**	C = **reset**	D = **clone**

5. Sobre as lacunas C e D da imagem da questão anterior, porém agora considerando apenas arquivos **não rastreados**, assinale a alternativa correta:

(/ 2)

a)		C = **reset**	D = **clean**
b)		C = **remove**	D = **clean**
c)		C = **reset**	D = **remove**
d)		C = **clean**	D = **reset**
e)		C = **reset**	D = **checkout**

6. Analise a imagem a seguir e preencha as lacunas de B a E com os comandos adequados.
(ex.: **A = git add arquivo.txt**) (/ 8)



B = git add modificado.txt
C = git checkout modificado.txt
D = git reset modificado.txt
E = git rm –cached arquivo.txt