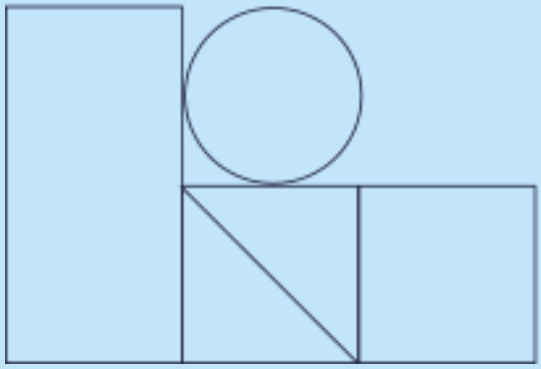
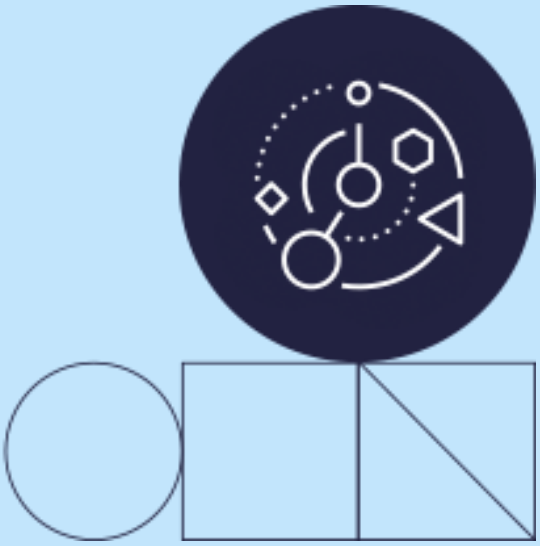
IBM SkillsBuild

IBM

Git e Github

Commits





Índice

Definição

O comando git commit captura um instantâneo das mudanças feitas atualmente em seu projeto. Instantâneos criados com compromisso podem ser considerados versões "seguras" de um projeto. Git nunca os modificará a menos que o solicitemos explicitamente.

Antes de executar o git commit, você usa o comando git add para "preparar" mudanças no projeto para ser armazenado em seu repositório.

Estes dois comandos de comando, git commit e git add, são dois dos comandos mais comumente usados.

Uso do commit

No tópico anterior, tínhamos deixado nosso arquivo na área de *stage*.

Para fazer um instantâneo de seu conteúdo, usaremos o comando "commit".

Na primeira vez que usaremos o comando "commit", seremos questionados sobre nosso e-mail e nome.

Git só solicitará estes detalhes na primeira vez que o utilizarmos em um computador

```
$ git commit -m "Version 1.0.0"
Author identity unknown

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.
```

Uma vez inseridos os dados seguindo as instruções fornecidas pela GIT, voltamos a executar o “commit”. Devemos também acrescentar uma descrição para o instantâneo, em nosso caso colocaremos, por exemplo, a "Versão 1.0.0".

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git commit -m “Version 1.0.0”
[master (root-commit) 00350f8] Version
1.0.0
 1 file changed, 0 insertions(+), 0
deletions(-)
 Créate mode 100644 index.html
```

Já temos uma primeira foto do nosso projeto, ou seja, uma cópia do nosso código como está no momento atual.

Se mais tarde precisarmos voltar à versão do projeto como era na época atual, podemos recuperá-la.

Se agora fizermos um “git status -s”:

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git status -s

?? estilo.css
?? main.js
```

Esta é a pasta que a GIT utilizará para criar as áreas locais de encenação e repositório.

Veremos que nosso arquivo desapareceu da lista. Isto porque assim que um arquivo é adicionado ao repositório e já existe um backup, o arquivo não está mais na área do palco e, portanto, não será mais mostrado. Com o comando "git status -s" só obteremos informações sobre o que não está no repositório.

Se fizermos qualquer mudança no conteúdo de nosso arquivo e executarmos um "git status -s" novamente, obteremos informações sobre o que não está no repositório:

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git status -s

M index.html
?? estilo.css
?? main.js
```

Vemos que agora ele nos informa sobre o status. Especificamente, o **M** indica que o arquivo foi modificado e tem mudanças que ainda não foram apoiadas.

Se eu quiser fazer um backup do arquivo com as novas modificações, teremos que colocar o arquivo na área de encenação e executar o comando: git commit -m “nova descrição”

Isto é, para cada mudança que fazemos, temos que passar novamente pela área do palco (para que possamos vê-la com um **M**) e depois fazer um *commit*.

Stage área:

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git add index.html
```

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git status -s

M index.html
?? css/
?? js/
```

Você verá agora um **A** indicando que ele está rastreando o arquivo index.html. Este arquivo está agora na área de encenação, a área de encenação.

Commit:

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git reset –hard 093f1c5

HEAD is now at 093f1c5 Version 1.0.0
```

Neste ponto, teríamos armazenado dois instantâneos de nosso arquivo index.html.

Para obter uma lista de todas as cópias que temos no repositório local, usaremos o comando:

git log –oneline

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git log oneline

934978a (HEAD -> master) Version 1.0.1
093f1c5 Version 1.0.0
```

Como podemos ver, em nosso caso temos duas versões do arquivo, V1.0.0 e V1.0.1.

Se quiséssemos restaurar o arquivo para a primeira versão que tínhamos, ou seja, restaurar o arquivo, deveríamos usar o comando:

git reset –hard 093f1c5

Como podemos ver, temos que passar o código alfanumérico do instantâneo que queremos recuperar.

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git commit -m “Version 1.0.1”
[master 934978a] Version 1.0.1
 1 file changed, 1 insertion(+), 1
deletion(-)
```

Com isso, agora teríamos nosso arquivo index.html restaurado para a primeira versão.

Deve-se notar que, tendo voltado ao primeiro instantâneo que tivemos (V1.0.0), a segunda versão do meu arquivo (V1.0.1) não existe mais, não está mais disponível, desaparece. Em outras palavras, podemos voltar a uma versão anterior, mas uma vez que fazemos isso, perdemos todas as versões após essa versão.

Já vimos como rastrear um arquivo específico.

Para rastrear todos os arquivos em nosso projeto, usaremos o comando:

git add .

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git add .
```

Isto adicionará todos os meus arquivos à *stage area*.

Se fizermos um git status -s veremos que Git está rastreando todos os arquivos em nosso diretório de trabalho.

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git status -s

A  estilo.css
A  main.js
```

Se agora fizermos mudanças no conteúdo dos arquivos index.html e style.cs, salvando essas mudanças e fazendo um git status -s.

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git status -s
```

Veremos que Git indica com **M** os arquivos que modificamos.

Para fazer um novo instantâneo do projeto incluindo as últimas mudanças feitas, devemos fazer duas etapas:

- Faça uma adição para adicionar ambos os arquivos de volta à staging área.
- Faça um "commit" para mover os arquivos para nosso repositório.

Em casos como este, onde temos que fazer uma add e um commit, podemos fazer ambas as operações em uma única etapa, usando o comando:

git commit -am “descripción”

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git commit -am “Version 1.0.2”

[master fd8ada9] Version 1.0.2
 3 files changed, 5 insertions(+)
 create mode 100644 estilo.css
 create mode 100644 main.js
```

Se fizermos uma verificação de status agora, nada deve aparecer porque todos os arquivos já estão adicionados ao repositório.

Neste ponto, temos vários instantâneos de nosso código no repositório local, ou seja, em nosso computador. Mas eles não são carregados para a nuvem.

Para isso, usaremos o GitHub mais tarde.