

## Contexto de arquivos

Nós podemos utilizar os comandos `reset` e `checkout` em alguns arquivos específicos, porém as suas funcionalidades são bem diferentes. Isso é bem útil quando queremos reverter alguma alteração em poucos arquivos só, sem nos preocupar de voltar o estado completo do commit.

Outro ponto é que os modos `mixed`, `soft` e `hard` não tem efeito ao serem aplicados em arquivos, e muitas vezes quando estamos nesse contexto, estamos desfazendo algo em nosso `working dir`.

Nesse contexto, usamos o `reset` para restaurar as mudanças na área de `staging` enquanto o `checkout` restaura os arquivos no `working dir`.

Vejamos alguns exemplos dos comandos vistos até aqui:

# Descarta e elimina as alterações do último commit



1

```
git reset HEAD~1 --hard
```

# Move o HEAD para dois commits anteriores



1

```
git checkout HEAD~2
```

# Descarta as alterações desde o último commit da master, mas mantém as alterações no `working dir` e no `staging`



1

```
git reset master --soft
```

# Reverte as alterações de um commit específico e cria um novo commit



1

```
git revert ab32e2
```

# Restaura para o staging as alterações do arquivo foo.txt em dois commits atrás



1

```
git reset HEAD~2 foo.txt
```

# Recupera diretamente no working dir o arquivo xpto.txt



1

```
git checkout -- xpto.txt
```

Abaixo temos uma tabela de consulta rápida de quando usar cada um dos comandos:

Comando	Escopo	Uso
git reset	Commit	Descartar commits em branches locais ou alterações ainda não envi
git reset	Arquivo	Altera o staging do arquivo
git checkout	Commit	Troca entre branches ou commits pontuais
git checkout	Arquivo	Descarta alterações no working dir
git revert	Commit	Desfazer commits em branches públicas

