人非圣贤孰能。所以每个 VCS 都提供一个功能来修复错误，直到 Git 控制的某一点上。 Git 提供了一个功能，可用于撤消对本地存储库所做的修改。

假设用户意外地对本地存储库进行了一些更改，然后想要撤消这些更改。 在这种情况下，恢复操作起着重要的作用。

恢复未提交的更改

假设我们不小心修改了本地存储库中的一个文件，此时想撤销这些修改。为了处理这种情况，我们可以使用 `git checkout` 命令。可以使用此命令来还原文件的内容。

为了更好的演示，我们首先在 `sample/src` 目录下创建一个文件：*string.py*，其代码如下所示 -

```
#!/usr/bin/python3

var1 = 'Hello World!'
var2 = "Python Programming"

print ("var1[0]: ", var1[0])
print ("var2[1:5]: ", var2[1:5]) # 切片加索引
```

并使用以下命令将此文件推送到远程存储库 -

```
$ pwd
/D/worksp/sample

Administrator@MY-PC /D/worksp/sample (master)
$ git add src/

Administrator@MY-PC /D/worksp/sample (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   src/string.py
```

```
Administrator@MY-PC /D/worksp/sample (master)
$ git add src/string.py

Administrator@MY-PC /D/worksp/sample (master)
$ git commit -m "add new file string.py"
[master 44ea8e4] add new file string.py
 1 file changed, 7 insertions(+)
 create mode 100644 src/string.py

Administrator@MY-PC /D/worksp/sample (master)
$ git push origin master
Username for 'http://git.oschina.net': 769728683@qq.com
Password for 'http://769728683@qq.com@git.oschina.net':
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 443 bytes | 0 bytes/s, done.
Total 4 (delta 0), reused 0 (delta 0)
```

现在，已经将 *string.py* 添加到远程存储库中了。

假设我们不小心/或者有心修改了本地存储库中的一个文件。但现在不想要这些修改的内容了，也就是说想要撤销修改。要处理这种情况，那么可以使用 `git checkout` 命令。可以使用此命令来还原文件的内容。

```
$ pwd
/D/worksp/sample

Administrator@MY-PC /D/worksp/sample (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   src/string.py
```

```
no changes added to commit (use "git add" and/or "git commit -a")

Administrator@MY-PC /D/worksp/sample (master)
$ git checkout src/string.py

Administrator@MY-PC /D/worksp/sample (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

nothing to commit, working directory clean

Administrator@MY-PC /D/worksp/sample (master)
$
```

此外，还可以使用`git checkout`命令从本地存储库获取已删除的文件。假设我们从本地存储库中删除一个文件，我们想要恢复这个文件。那么可以通过使用`git checkout`命令来实现这一点。

```
$ ls -l
total 1
-rw-r--r--    1 Administ Administ       57 Jul  7 05:37 README.md
drwxr-xr-x    1 Administ Administ        0 Jul 10 21:16 src

Administrator@MY-PC /D/worksp/sample (master)
$ cd src/

Administrator@MY-PC /D/worksp/sample/src (master)
$ ls -l
total 1
-rwxr-xr-x    1 Administ Administ      156 Jul 10 21:16 string.py

Administrator@MY-PC /D/worksp/sample/src (master)
$ rm string.py

Administrator@MY-PC /D/worksp/sample/src (master)
$ ls -l
```

```
total 0


Administrator@MY-PC /D/worksp/sample/src (master)
$ git status -s
 D string.py
```

Git 在文件名前显示字母 `D`，这表示该文件已从本地存储库中删除。

```
$ git checkout string.py


Administrator@MY-PC /D/worksp/sample/src (master)
$ ls -l
total 1
-rwxr-xr-x    1 Administ Administ      156 Jul 10 21:24 string.py

Administrator@MY-PC /D/worksp/sample/src (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

nothing to commit, working directory clean
```

注意：可以在提交操作之前执行这些操作。

删除分段区域的更改

我们已经看到，当执行添加操作时，文件将从本地存储库移动到暂存区域。 如果用户意外修改文件并将其添加到暂存区域，则可以使用 `git checkout` 命令恢复其更改。

在 Git 中，有一个 `HEAD` 指针总是指向最新的提交。 如果要从分段区域撤消更改，则可以使用 `git checkout` 命令，但是使用 `checkout` 命令，必须提供一个附加参数，即 `HEAD` 指针。 附加的提交指针参数指示 `git checkout` 命令重置工作树，并删除分段更改。

让我们假设从本地存储库修改一个文件。 如果查看此文件的状态，它将显示该文件已修改但未添加到暂存区域。

```
$ pwd
/D/worksp/sample/src


Administrator@MY-PC /D/worksp/sample/src (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
```

```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   string.py

no changes added to commit (use "git add" and/or "git commit -a")

Administrator@MY-PC /D/worksp/sample/src (master)
$ git add string.py
```

Git 状态显示该文件存在于暂存区域，现在使用 `git checkout` 命令恢复该文件，并查看还原文件的状态。

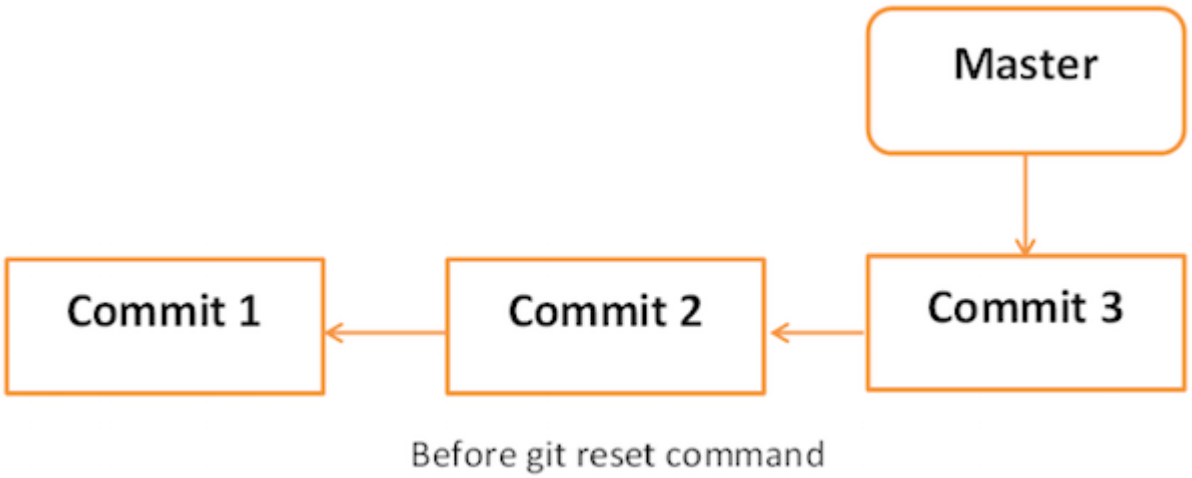
```
$ git checkout head -- string.py

Administrator@MY-PC /D/worksp/sample/src (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

nothing to commit, working directory clean
```

用 Git 复位移动头指针

经过少量更改后，可以决定删除这些更改。 `git reset` 命令用于复位或恢复更改。 我们可以执行三种不同类型的复位操作。

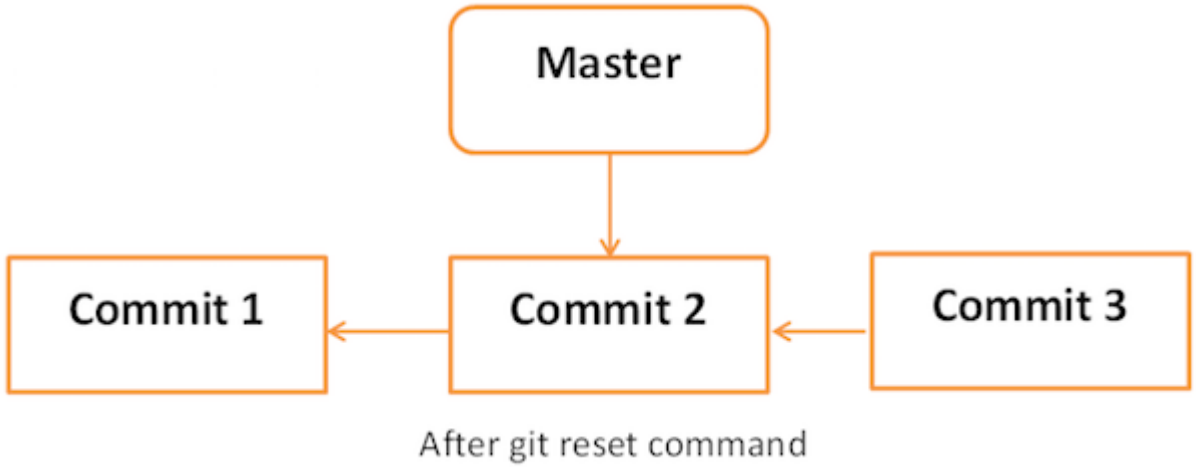
下图显示了 `git reset` 命令的图示。

`git reset` 命令之前 -



Before git reset command

`git reset` 命令之后 -



After git reset command

—soft 选项

每个分支都有一个 `HEAD` 指针，它指向最新的提交。 如果用`--soft` 选项后跟提交 ID 的 `Git reset` 命令，那么它将仅重置 `HEAD` 指针而不会破坏任何东西。

`.git/refs/heads/master` 文件存储 `HEAD` 指针的提交 ID。 可使用 `git log -1` 命令验证它。

```
$ pwd
/D/worksp/sample


Administrator@MY-PC /D/worksp/sample (master)
$ cat .git/refs/heads/master
44ea8e47307b47c9a80b44360e09f973e79312b0
```

现在，查看最新前两个的提交 ID，最近一次 ID 将与上述提交 ID 一致。

```
$ git log -2
commit 44ea8e47307b47c9a80b44360e09f973e79312b0
```

```
Author: maxsu <your_email@mail.com>
Date:   Mon Jul 10 21:09:35 2017 +0800

    add new file string.py

commit 7d8162db36723b8523c56ad658a07808ae7fb64c
Author: minsu <minsu@yiibai.com>
Date:   Mon Jul 10 17:51:11 2017 -0700

    remove/delete module.py

Administrator@MY-PC /D/worksp/sample (master)
$
```

下面我们重置 `HEAD` 指针。

```

```

现在，只需将 HEAD 指针重新设置一个位置。现在查看`.git/refs/heads/master` 文件的内容。

```
Administrator@MY-PC /D/worksp/sample (master)
$ cat .git/refs/heads/master
7d8162db36723b8523c56ad658a07808ae7fb64c
```

来自文件的提交 ID 已更改，现在通过查看提交消息进行验证。

```
$ git log -2
commit 7d8162db36723b8523c56ad658a07808ae7fb64c
Author: minsu <minsu@yiibai.com>
Date:   Mon Jul 10 17:51:11 2017 -0700

    remove/delete module.py

commit 6bdbf8219c60d8da9ad352c23628600faaefbe13
Author: maxsu <your_email@mail.com>
Date:   Mon Jul 10 20:34:28 2017 +0800

    renamed main.py to module.py
```

mixed 选项

使用`--mixed`选项的 Git 重置将从尚未提交的暂存区域还原这些更改。它仅从暂存区域恢复更改。对文件的工作副本进行的实际更改不受影响。 默认 Git 复位等效于执行`git reset – mixed`。

hard 选项

如果使用`--hard`选项与 Git 重置命令，它将清除分段区域; 它会将 HEAD 指针重置为特定提交 ID 的最新提交，并删除本地文件更改。

让我们查看提交 ID。

```
Administrator@MY-PC /D/worksp/sample (master)
$ pwd
/D/worksp/sample

Administrator@MY-PC /D/worksp/sample (master)
$ git log -1
commit 7d8162db36723b8523c56ad658a07808ae7fb64c
Author: minsu <minsu@yiibai.com>
Date:   Mon Jul 10 17:51:11 2017 -0700

    remove/delete module.py
```

通过在文件开头添加单行注释来修改文件或者往文件里添加其它代码。

```
$ head -2 src/string.py
#!/usr/bin/python3



Administrator@MY-PC /D/worksp/sample (master)
$ git status -s
 M src/string.py
```

将修改的文件添加到暂存区域，并使用`git status`命令进行验证。

```
$ git add src/string.py

Administrator@MY-PC /D/worksp/sample (master)
$ git status
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 1 different commit each, respectively.
  (use "git pull" to merge the remote branch into yours)
```

```
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   src/string.py
```

Git 状态显示该文件存在于暂存区域中。 现在，重置 HEAD 与`--hard`选项。

```
$ git reset --hard 7d8162db36723b8523c56ad658a07808ae7fb64c
HEAD is now at 7d8162d remove/delete module.py

Administrator@MY-PC /D/worksp/sample (master)
```

`git reset`命令成功，这将从分段区域还原文件，并删除对文件所做的任何本地更改。

```
Administrator@MY-PC /D/worksp/sample (master)
$ git status -s
```

Git 状态显示该文件已从暂存区域还原，当前恢复到了删除`moudle.py`时的版本了。