软件开发中，bug就像家常便饭一样。有了bug就需要修复，在Git中，由于分支是如此的强大，所以，每个bug都可以通过一个新的临时分支来修复，修复后，合并分支，然后将临时分支删除。

当你接到一个修复一个代号101的bug的任务时，很自然地，你想创建一个分支issue-101来修复它，但是，等等，当前正在dev上进行的工作还没有提交：

$ git status

On branch dev

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

new file: hello.py

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: readme.txt

并不是你不想提交，而是工作只进行到一半，还没法提交，预计完成还需1天时间。但是，必须在两个小时内修复该bug，怎么办？

幸好，Git还提供了一个stash功能，可以把当前工作现场“储藏”起来，等以后恢复现场后继续工作：

$ git stashSaved working directory **and** index state WIP on dev: f52c633 add merge

现在，用git status查看工作区，就是干净的（除非有没有被Git管理的文件），因此可以放心地创建分支来修复bug。

首先确定要在哪个分支上修复bug，假定需要在master分支上修复，就从master创建临时分支：

$ git checkout masterSwitched to branch 'master'Your branch is ahead of 'origin/master' by 6 commits.

(use "git push" to publish your local commits)

$ git checkout -b issue-101Switched to a new branch 'issue-101'

现在修复bug，需要把“Git is free software ...”改为“Git is a free software ...”，然后提交：

$ git add readme.txt $ git commit -m "fix bug 101"

[issue-101 4c805e2] fix bug 101

1 file changed, 1 insertion(+), 1 deletion(-)

修复完成后，切换到master分支，并完成合并，最后删除issue-101分支：

$ git checkout masterSwitched to branch 'master'Your branch is ahead of 'origin/master' by 6 commits.

(use "git push" to publish your local commits)

$ git merge --no-ff -m "merged bug fix 101" issue-101Merge made by the 'recursive' strategy.

readme.txt | 2 +-

1 file changed, 1 insertion(+), 1 deletion(-)

太棒了，原计划两个小时的bug修复只花了5分钟！现在，是时候接着回到dev分支干活了！

$ git checkout devSwitched to branch 'dev'

$ git statusOn branch dev

nothing to commit, working tree clean

工作区是干净的，刚才的工作现场存到哪去了？用git stash list命令看看：

$ git stash list

stash@{0}: WIP on dev: f52c633 add merge

工作现场还在，Git把stash内容存在某个地方了，但是需要恢复一下，有两个办法：

一是用git stash apply恢复，但是恢复后，stash内容并不删除，你需要用git stash drop来删除；

另一种方式是用git stash pop，恢复的同时把stash内容也删了：

$ git stash pop

On branch dev

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

new file: hello.py

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: readme.txt

Dropped refs/stash@{0} (5d677e2ee266f39ea296182fb2354265b91b3b2a)

再用git stash list查看，就看不到任何stash内容了：

$ git stash list

你可以多次stash，恢复的时候，先用git stash list查看，然后恢复指定的stash，用命令：

$ git stash apply stash@{0}

### 小结

修复bug时，我们会通过创建新的bug分支进行修复，然后合并，最后删除；

当手头工作没有完成时，先把工作现场git stash一下，然后去修复bug，修复后，再git stash pop，回到工作现场。