Git - Managing Branches

Branch operation allows creating another line of development. We can use this operation to fork off the development process into two different directions. For example, we released a product for 6.0 version and we might want to create a branch so that the development of 7.0 features can be kept separate from 6.0 bug fixes.

Create Branch

Tom creates a new branch using the git branch <branch name> command. We can create a new branch from an existing one. We can use a specific commit or tag as the starting point. If any specific commit ID is not provided, then the branch will be created with HEAD as its starting point.

[jerry@CentOS src]$ git branch new\_branch

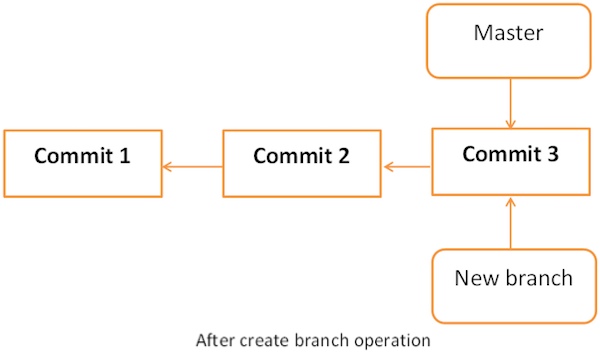
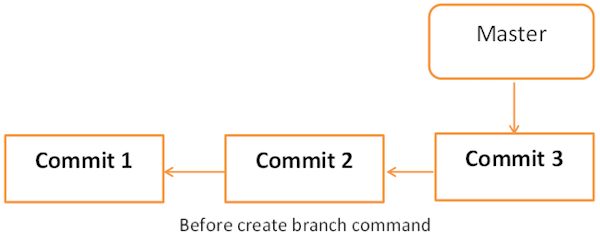
[jerry@CentOS src]$ git branch

\* master

new\_branch

A new branch is created; Tom used the git branch command to list the available branches. Git shows an asterisk mark before currently checked out branch.

The pictorial representation of create branch operation is shown below:



Switch between Branches

Jerry uses the git checkout command to switch between branches.

[jerry@CentOS src]$ git checkout new\_branch

Switched to branch 'new\_branch'

[jerry@CentOS src]$ git branch

master

\* new\_branch

Shortcut to Create and Switch Branch

In the above example, we have used two commands to create and switch branches, respectively. Git provides **–b** option with the checkout command; this operation creates a new branch and immediately switches to the new branch.

[jerry@CentOS src]$ git checkout -b test\_branch

Switched to a new branch 'test\_branch'

[jerry@CentOS src]$ git branch

master

new\_branch

\* test\_branch

Delete a Branch

A branch can be deleted by providing –D option with git branch command. But before deleting the existing branch, switch to the other branch.

Jerry is currently on **test\_branch** and he wants to remove that branch. So he switches branch and deletes branch as shown below.

[jerry@CentOS src]$ git branch

master

new\_branch

\* test\_branch

[jerry@CentOS src]$ git checkout master

Switched to branch 'master'

[jerry@CentOS src]$ git branch -D test\_branch

Deleted branch test\_branch (was 5776472).

Now, Git will show only two branches.

[jerry@CentOS src]$ git branch

\* master

new\_branch

Rename a Branch

Jerry decides to add support for wide characters in his string operations project. He has already created a new branch, but the branch name is not appropriate. So he changes the branch name by using **–m** option followed by the **old branch name** and the **new branch name**.

[jerry@CentOS src]$ git branch

\* master

new\_branch

[jerry@CentOS src]$ git branch -m new\_branch wchar\_support

Now, the git branch command will show the new branch name.

[jerry@CentOS src]$ git branch

\* master

wchar\_support

Merge Two Branches

Jerry implements a function to return the string length of wide character string. New the code will appear as follows:

[jerry@CentOS src]$ git branch

master

\* wchar\_support

[jerry@CentOS src]$ pwd

/home/jerry/jerry\_repo/project/src

[jerry@CentOS src]$ git diff

The above command produces the following result:

t a/src/string\_operations.c b/src/string\_operations.c

index 8ab7f42..8fb4b00 100644

--- a/src/string\_operations.c

+++ b/src/string\_operations.c

@@ -1,4 +1,14 @@

#include <stdio.h>

+#include <wchar.h>

+

+size\_t w\_strlen(const wchar\_t \*s)

+

{

+

const wchar\_t \*p = s;

+

+

while (\*p)

+ ++p;

+ return (p - s);

+

}

After testing, he commits and pushes his changes to the new branch.

[jerry@CentOS src]$ git status -s

M string\_operations.c

?? string\_operations

[jerry@CentOS src]$ git add string\_operations.c

[jerry@CentOS src]$ git commit -m 'Added w\_strlen function to return string lenght of wchar\_t

string'

[wchar\_support 64192f9] Added w\_strlen function to return string lenght of wchar\_t string

1 files changed, 10 insertions(+), 0 deletions(-)

Note that Jerry is pushing these changes to the new branch, which is why he used the branch name **wchar\_support** instead of **master** branch.

[jerry@CentOS src]$ git push origin wchar\_support  **<−−− Observer branch\_name**

The above command will produce the following result.

Counting objects: 7, done.

Compressing objects: 100% (4/4), done.

Writing objects: 100% (4/4), 507 bytes, done.

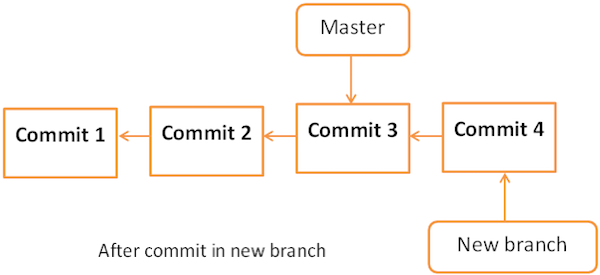
Total 4 (delta 1), reused 0 (delta 0)

To gituser@git.server.com:project.git

\* [new branch]

wchar\_support -> wchar\_support

After committing the changes, the new branch will appear as follows:



Tom is curious about what Jerry is doing in his private branch and he checks the log from the **wchar\_support** branch.

[tom@CentOS src]$ pwd

/home/tom/top\_repo/project/src

[tom@CentOS src]$ git log origin/wchar\_support -2

The above command will produce the following result.

commit 64192f91d7cc2bcdf3bf946dd33ece63b74184a3

Author: Jerry Mouse <jerry@tutorialspoint.com>

Date: Wed Sep 11 16:10:06 2013 +0530

Added w\_strlen function to return string lenght of wchar\_t string

commit 577647211ed44fe2ae479427a0668a4f12ed71a1

Author: Tom Cat <tom@tutorialspoint.com>

Date: Wed Sep 11 10:21:20 2013 +0530

Removed executable binary

By viewing commit messages, Tom realizes that Jerry implemented the strlen function for wide character and he wants the same functionality in the master branch. Instead of re-implementing, he decides to take Jerry’s code by merging his branch with the master branch.

[tom@CentOS project]$ git branch

\* master

[tom@CentOS project]$ pwd

/home/tom/top\_repo/project

[tom@CentOS project]$ git merge origin/wchar\_support

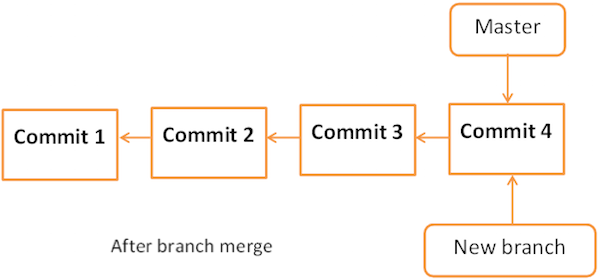
Updating 5776472..64192f9

Fast-forward

src/string\_operations.c | 10 ++++++++++

1 files changed, 10 insertions(+), 0 deletions(-)

After the merge operation, the master branch will appear as follows:



Now, the branch **wchar\_support** has been merged with the master branch. We can verify it by viewing the commit message or by viewing the modifications done into the string\_operation.c file.

[tom@CentOS project]$ cd src/

[tom@CentOS src]$ git log -1

commit 64192f91d7cc2bcdf3bf946dd33ece63b74184a3

Author: Jerry Mouse

Date: Wed Sep 11 16:10:06 2013 +0530

Added w\_strlen function to return string lenght of wchar\_t string

[tom@CentOS src]$ head -12 string\_operations.c

The above command will produce the following result.

#include <stdio.h>

#include <wchar.h>

size\_t w\_strlen(const wchar\_t \*s)

{

const wchar\_t \*p = s;

while (\*p)

++p;

return (p - s);

}

After testing, he pushes his code changes to the master branch.

[tom@CentOS src]$ git push origin master

Total 0 (delta 0), reused 0 (delta 0)

To gituser@git.server.com:project.git

5776472..64192f9 master −> master

Rebase Branches

The Git rebase command is a branch merge command, but the difference is that it modifies the order of commits.

The Git merge command tries to put the commits from other branches on top of the HEAD of the current local branch. For example, your local branch has commits A−>B−>C−>D and the merge branch has commits A−>B−>X−>Y, then git merge will convert the current local branch to something like A−>B−>C−>D−>X−>Y

The Git rebase command tries to find out the common ancestor between the current local branch and the merge branch. It then pushes the commits to the local branch by modifying the order of commits in the current local branch. For example, if your local branch has commits A−>B−>C−>D and the merge branch has commits A−>B−>X−>Y, then Git rebase will convert the current local branch to something like A−>B−>X−>Y−>C−>D.

When multiple developers work on a single remote repository, you cannot modify the order of the commits in the remote repository. In this situation, you can use rebase operation to put your local commits on top of the remote repository commits and you can push these changes.