6 handy Bash scripts for Git

These six Bash scripts will make your life easier when you're working with Git repositories.

| 13 min read

371 readers like this

Image by: *Opensource.com*

More on Git

- What is Git?
- Git cheat sheet
- Markdown cheat sheet
- New Git articles

I wrote a bunch of Bash scripts that make my life easier when I'm working with Git repositories. Many of my colleagues say there's no need; that everything I need to do can be done with Git commands. While that may be true, I find the scripts trying to figure out the appropriate Git command

: of current patches against the master version. It est and shows the author and description, with **H HEAD~2,** and so forth. For example:

```
----[ recovery25
-----

11 340d27a33895 Bob Peterson     gfs2: drain the
ail2 list after io errors
10 9b3c4e6efb10 Bob Peterson     gfs2: clean up
iopen glock mess in gfs2_create_inode
 9 d2e8c22be39b Bob Peterson     gfs2: Do proper
error checking for go_sync family of glops
 8 9563e31f8bfd Christoph Hellwig gfs2: use
page_offset in gfs2_page_mkwrite
 7 ebac7a38036c Christoph Hellwig gfs2: don't use
buffer_heads in gfs2_all
 6 f703a3c27874 Andreas
```

```
mmap write vs. punch_hole consistency
 5 a3e86d2ef30e Andreas Gruenbacher gfs2: Multi-
block allocations in gfs2_page_mkwrite
 4 da3c604755b0 Andreas Gruenbacher gfs2: Fix end-
of-file handling in gfs2_page_mkwrite
 3 4525c2f5b46f Bob Peterson     Rafael Aquini's
slab instrumentation
 2 a06a5b7dea02 Bob Peterson     GFS2: Add
go_get_holdtime to gl_ops
 ^ 8ba93c796d5c Bob Peterson     gfs2: introduce new
function remaining_hold_time and use it in dq
 H e8b5ff851bb9 Bob Peterson     gfs2: Allow rgrps
to have a minimum hold time
```


e on a different branch, I can specify an alternate

SHA1 IDs:

```
----[ recovery25
-----
56908eeb6940 2ca4a6b628a1 fc64ad5d99fe 02031a00a251
f6f38da7dd18 d8546e8f0023 fc3cc1f98f6b 12c3e0cb3523
76cce178b134 6fc1dce3ab9c 1b681ab074ca 26fed8de719b
802ff51a5670 49f67a512d8c f04f20193bbb 5f6afe809d23
2030521dc70e dada79b3be94 9b19a1e08161 78a035041d3e
f03da011cae2 0d2b2e068fcd 2449976aa133 57dfb5e12ccd
53abedfdcf72 6fbdda3474b3 49544a547188 187032f7a63c
6f75dae23d93 95fc2a261b00 ebfb14ded191 f653ee9e414a
0e2911cb8111 73968b76e2e
7c9ef68388ed 71ca19d0cba
d2e8c22be39b 9563e31f8b1
```

```
a3e86d2ef30e da3c604755b0 4525c2f5b46f a06a5b7dea02
8ba93c796d5c e8b5ff851bb9
```

Again, it assumes the current branch, but I can specify a different branch if I want.

3. gitlog.id2

gitlog.id2 is the same as **gitlog.id** but without the branch line at the top. This is handy for cherry-picking all patches from one branch to the current branch:

```
$ # create a new branch
$ git branch --track origin/master
                    iew branch I just created
                    overy26
                    . patches from the old branch to

                    id2 recovery25` ; do git cherry-
```


within that collection of patches. For example, if I
atch that has a reference to function

```
$ gitlog.grep inode_go_sync
-----------------------[ recovery25 - 50 patches
]-----------------------
(snip)
11 340d27a33895 Bob Peterson     gfs2: drain the
ail2 list after io errors
10 9b3c4e6efb10 Bob Peterson     gfs2: clean up
iopen glock mess in gfs2_create_inode
 9 d2e8c22be39b Bob Peterson     gfs2: Do proper
error checking for go_sy
152:-static void inode_g
```

```
*gl)
153:+static int inode_go_sync(struct gfs2_glock *gl)
163:@@ -296,6 +302,7 @@ static void
inode_go_sync(struct gfs2_glock *gl)
 8 9563e31f8bfd Christoph Hellwig gfs2: use
page_offset in gfs2_page_mkwrite
 7 ebac7a38036c Christoph Hellwig gfs2: don't use
buffer_heads in gfs2_allocate_page_backing
 6 f703a3c27874 Andreas Gruenbacher gfs2: Improve
mmap write vs. punch_hole consistency
 5 a3e86d2ef30e Andreas Gruenbacher gfs2: Multi-
block allocations in gfs2_page_mkwrite
 4 da3c604755b0 Andreas Gruenbacher gfs2: Fix end-
of-file handling in gfs2_page_mkwrite
 ) Peterson     Rafael Aquini's
on
 ) Peterson     GFS2: Add
 gl_ops
 ) Peterson     gfs2: introduce new
_hold_time and use it in dq
 ) Peterson     gfs2: Allow rgrps
old time
```


D~9 is the one that needs fixing. I use **git rebase -i ommit -a --amend**, then **git rebase --continue** ents.

gitbranchcmp3 lets me compare my current branch to another branch, so I can compare older versions of patches to my newer versions and quickly see what's changed and what hasn't. It generates a compare script (that uses the KDE tool Kompare, which works on GNOME3, as well) to compare the patches that aren't quite the same. If there are no differences other than line numbers, it prints **[SAME]**. If there are only comment differences, it prints **[same]** (in lower case). For example:

```
$ gitbranchcmp3 recovery24
Branch recovery24 has 47 patches
Branch recovery25 has 50 patches

(snip)
38 87eb6901607a 340d27a33895 [same] gfs2: drain the
ail2 list after io errors
39 90fefb577a26 9b3c4e6efb10 [same] gfs2: clean up
iopen glock mess in gfs2_create_inode
40 ba3ae06b8b0e d2e8c22be39b [same] gfs2: Do proper
error checking for go_sync family of glops
41 2ab662294329 9563e31f8bfd [SAME] gfs2: use
page_offset in gfs2_page_mkwrite
42 0adc6d817b7a ebac7a38036c [SAME] gfs2: don't use
                s2_allocate_page_backing
                03a3c27874 [SAME] gfs2: Improve
                h_hole consistency
                e86d2ef30e [SAME] gfs2: Multi-
                n gfs2_page_mkwrite
                3c604755b0 [SAME] gfs2: Fix end-
                gfs2_page_mkwrite
                25c2f5b46f [SAME] Rafael Aquini's
                on
                5a5b7dea02 [    ] GFS2: Add
                gl_ops
                193c796d5c [    ] gfs2: introduce
                ing_hold_time and use it in dq
                05ff851bb9 [    ] gfs2: Allow
                imum hold time

Missing from recovery25:
The missing:
Compare script generated at:
/tmp/compare_mismatches.sh
```


6. gitlog.find

Finally, I have **gitlog.find**, a script to h
of my patches are and each patch's cu

patch description. It also generates a compare script (again, using Kompare) to compare the current patch to the upstream counterpart:

```
$ gitlog.find
-----------------------[ recovery25 - 50 patches
]-----------------------
(snip)
11 340d27a33895 Bob Peterson     gfs2: drain the
ail2 list after io errors
lo 5bcb9be74b2a Bob Peterson     gfs2: drain the
ail2 list after io errors
10 9b3c4e6efb10 Bob Peterson     gfs2: clean up
iopen glock mess in gfs2_create_inode
                 ) Peterson     gfs2: clean up
                 ) gfs2_create_inode
                 ) Peterson     gfs2: Do proper
                  go_sync family of glops
                 ) Peterson     gfs2: Do proper
                  go_sync family of glops
                 ristoph Hellwig gfs2: use
                 _page_mkwrite
                 ristoph Hellwig gfs2: use
                 _page_mkwrite
                 ristoph Hellwig gfs2: don't use
                 2_allocate_page_backing
                 ristoph Hellwig gfs2: don't use
                 2_allocate_page_backing
                 lreas Gruenbacher gfs2: Improve
mmap write vs. punch_hole consistency
fn 39c3a948ecf6 Andreas Gruenbacher gfs2: Improve
mmap write vs. punch_hole consistency
 5 a3e86d2ef30e Andreas Gruenbacher gfs2: Multi-
block allocations in gfs2_page_mkwrite
fn f53056c43063 Andreas Gruenbacher gfs2: Multi-
block allocations in gfs2_page_mkwrite
 4 da3c604755b0 Andreas Gruenbacher gfs2: Fix end-
of-file handling in gfs2_page_mkwrite
fn 184b4e60853d Andreas
of-file handling in gfs2
```

```
 3 4525c2f5b46f Bob Peterson     Rafael Aquini's
slab instrumentation
   Not found upstream
 2 a06a5b7dea02 Bob Peterson     GFS2: Add
go_get_holdtime to gl_ops
   Not found upstream
 ^ 8ba93c796d5c Bob Peterson     gfs2: introduce new
function remaining_hold_time and use it in dq
   Not found upstream
 H e8b5ff851bb9 Bob Peterson     gfs2: Allow rgrps
to have a minimum hold time
   Not found upstream
Compare script generated: /tmp/compare_upstream.sh
```


ines, the first of which is your current patch, upstream patch, and a 2-character abbreviation to

e local upstream Git repo only (i.e., not pushed

inus Torvald's master branch.

ned to my "for-next" development branch, intended ge window.

ptions based on how I normally work with Git. For stream patches, it uses my well-known Git tree's location. So, you will need to adjust or improve them to suit your conditions. The **gitlog.find** script is designed to locate GFS2 and DLM patches only, so unless you're a GFS2 developer, you will want to customize it to the components that interest you.

Source code

Here is the source for these scripts.

1. gitlog

```
#!/bin/bash
branch=$1

if test "x$branch" = x; then
    branch=`git branch -a | grep "*" | cut -d ' '
-f2`
fi

patches=0
tracking=`git rev-parse --abbrev-ref --symbolic-
full-name @{u}`

LIST=`git log --reverse --abbrev-commit
  pretty=oneline $tracking..$branch | cut -d ' ' -f1
```



```
    patches=$(echo $patches + 1 |

:for-next.* ]]

:or-next

ıster

ıarse --abbrev-ref --symbolic-
```

```
/usr/bin/echo "----------------------[" $branch
"]----------------------"
patches=$(echo $patches - 1 | bc);
for i in $LIST; do
    if [ $patches -eq 1 ]; then
        cnt=" ^"
    elif [ $patches -eq 0 ]; then
        cnt=" H"
    else
        if [ $patches -l
            cnt=" $patch
```

```
        else
            cnt="$patches"
        fi
    fi
    /usr/bin/git show --abbrev-commit -s
--pretty=format:"$cnt %h %<|(32)%an %s %n" $i
    patches=$(echo $patches - 1 | bc)
done
#git log --reverse --abbrev-commit
--pretty=format:"%h %<|(32)%an %s"
$tracking..$branch
#git log --reverse --abbrev-commit
--pretty=format:"%h %<|(32)%an %s" ^origin/master
^linux-gfs2/for-next $branch
```



```
= x; then
nch -a | grep "*" | cut -d ' '


arse --abbrev-ref --symbolic-
```

```
/usr/bin/echo "----------------------[" $branch
"]----------------------"
git log --reverse --abbrev-commit --pretty=oneline
$tracking..$branch | cut -d ' ' -f1 |paste -s -d ' '
```

3. gitlog.id2

```
#!/bin/bash
branch=$1
```

```
if test "x$branch" = x; then
    branch=`git branch -a | grep "*" | cut -d ' '
-f2`
fi

tracking=`git rev-parse --abbrev-ref --symbolic-
full-name @{u}`
git log --reverse --abbrev-commit --pretty=oneline
$tracking..$branch | cut -d ' ' -f1 |paste -s -d ' '
```

4. gitlog.grep


```
= x; then
nch -a | grep "*" | cut -d ' '
```

```
arse --abbrev-ref --symbolic-
full-name @{u}`

LIST=`git log --reverse --abbrev-commit
--pretty=oneline $tracking..$branch | cut -d ' ' -f1
|paste -s -d ' '`
for i in $LIST; do patches=$(echo $patches + 1 |
bc);done
/usr/bin/echo "----------------------[" $branch "-"
$patches "patches ]----------------------"
patches=$(echo $patches
for i in $LIST; do
```

```
    if [ $patches -eq 1 ]; then
        cnt=" ^"
    elif [ $patches -eq 0 ]; then
        cnt=" H"
    else
        if [ $patches -lt 10 ]; then
            cnt=" $patches"
        else
            cnt="$patches"
        fi
    fi
    /usr/bin/git show --abbrev-commit -s
--pretty=format:"$cnt %h %<|(32)%an %s" $i
    /usr/bin/git show --pretty=email --patch-with-
                                  "$string"
                                   $patches - 1 | bc)
```



```
                                  d branch> [<new_branch>]


                                  _mismatches.sh
```

```
/usr/bin/rm -f $script
echo "#!/bin/bash" > $script
/usr/bin/chmod 755 $script
echo "# Generated by gitbranchcmp3.sh" >> $script
echo "# Run this script to compare the mismatched
patches" >> $script
echo " " >> $script
echo "function compare_them()" >> $script
echo "{"  >> $script
echo "    git show --pre
\$1 > /tmp/gronk1" >> $s
```

```
echo "    git show --pretty=email --patch-with-stat
\$2 > /tmp/gronk2" >> $script
echo "    kompare /tmp/gronk1 /tmp/gronk2" >>
$script
echo "}" >> $script
echo " " >> $script

if test "x$newbranch" = x; then
    newbranch=`git branch -a | grep "*" | cut -d ' '
-f2`
fi

tracking=`git rev-parse --abbrev-ref --symbolic-
full-name @{u}`
```



```
=(`git log --reverse --abbrev-
eline $tracking..$oldbranch | cut
s -d ' '`)
=(`git log --reverse --abbrev-
eline $tracking..$newbranch | cut
s -d ' '`)

ha1s
s[@]}
anch has $oldcount patches"
dcount - 1 | bc)
#oldsha1s[@]}`; do
ha1s[$o]} " "
 $i | head -5 | tail -1|cut -b5-`
done

#echo "new: " $newsha1s
newcount=${#newsha1s[@]}
echo "Branch $newbranch has $newcount patches"
newcount=$(echo $newcount - 1 | bc)
#for o in `seq 0 ${#newsha1s[@]}`; do
#    echo -n ${newsha1s[$o]} " "
#    desc=`git show $i | head -5 | tail -1|cut -b5-`
#done
echo
```

```
for new in `seq 0 $newcount`; do
    newsha=${newsha1s[$new]}
    newdesc=`git show $newsha | head -5 | tail
-1|cut -b5-`
    oldsha="            "
    same="[    ]"
    for old in `seq 0 $oldcount`; do
        if test "${oldsha1s[$old]}" = "match"; then
            continue;
        fi
        olddesc=`git show ${oldsha1s[$old]} | head
-5 | tail -1|cut -b5-`
        if test "$olddesc" = "$newdesc" ; then
            oldsha=${oldsha1s[$old]}
            "  '  'oldsha
            ow $oldsha |tail -n +2 |grep -v
            ) -v "@@" > /tmp/gronk1
            ow $newsha |tail -n +2 |grep -v
            ) -v "@@"  > /tmp/gronk2
            :mp/gronk1 /tmp/gronk2 &>

            ' -eq 0 ] ;then

            ie="[SAME]"
            lsha1s[$old]="match"
            eak

            ow $oldsha |sed -n '/diff/,$p'
             \." |grep -v "@@" > /tmp/gronk1
            g-- show $newsha |sed -n '/diff/,$p'
|grep -v "index.*\.\." |grep -v "@@" > /tmp/gronk2
            diff /tmp/gronk1 /tmp/gronk2 &>
/dev/null
            if [ $? -eq 0 ] ;then
# Differences in comments only
                same="[same]"
                oldsha1s[$old]="match"
                break
            fi
            oldsha1s[$ol
            echo "compa
```

```
$script
        fi
    done
    echo "$new $oldsha $newsha $same $newdesc"
done

echo
echo "Missing from $newbranch:"
the_missing=""
# Now run through the olds we haven't matched up
for old in `seq 0 $oldcount`; do
    if test ${oldsha1s[$old]} != "match"; then
        olddesc=`git show ${oldsha1s[$old]} | head
-5 | tail -1|cut -b5-`
        ... "${...}sha1s[$old]} $olddesc"
        ...=`echo "$the_missing
```



```
" $the_missing
)t generated at: $script"
--abbrev-commit --pretty=oneline
| cut -d ' ' -f1 |paste -s -d ' '
```

```
#
# Find the upstream equivalent patch
#
# gitlog.find
#
cwd=$PWD
param1=$1
ubranch=$2
patches=0
script=/tmp/compare_upst
echo "#!/bin/bash" > $sc
```

```
/usr/bin/chmod 755 $script
echo "# Generated by gitbranchcmp3.sh" >> $script
echo "# Run this script to compare the mismatched
patches" >> $script
echo " " >> $script
echo "function compare_them()" >> $script
echo "{"  >> $script
echo "    cwd=$PWD" >> $script
echo "    git show --pretty=email --patch-with-stat
\$2 > /tmp/gronk2" >> $script
echo "    cd ~/linux.git/fs/gfs2" >> $script
echo "    git show --pretty=email --patch-with-stat
\$1 > /tmp/gronk1" >> $script
echo "    cd $cwd" >> $script
                    'tmp/gronk1 /tmp/gronk2" >>
```



```
                    stream patch info. Please wait."
                    -a | grep "*" | cut -d ' ' -f2`
                    arse --abbrev-ref --symbolic-


                    }" = "X"; then
                    anch -a | grep "*" | cut -d ' '

                    parse --abbrev-ref --symbolic-
full-name @{u}`
#
# gather a list of gfs2 patches from master just in
case we can't find it
#
#git log --abbrev-commit --pretty=format:"   %h
%<|(32)%an %s" master |grep -i -e "gfs2" -e "dlm" >
/tmp/gronk
git log --reverse --abbr
--pretty=format:"ms %h %
fs/gfs2/ > /tmp/gronk.g1
```

```
# ms = in Linus's master
git log --reverse --abbrev-commit
--pretty=format:"ms %h %<|(32)%an %s" master fs/dlm/
> /tmp/gronk.dlm

cd $cwd
LIST=`git log --reverse --abbrev-commit
--pretty=oneline $tracking..$branch | cut -d ' ' -f1
|paste -s -d ' '`
for i in $LIST; do patches=$(echo $patches + 1 |
bc);done
/usr/bin/echo "----------------------[" $branch "-"
$patches "patches ]----------------------"
patches=$(echo $patches - 1 | bc);
```



```
eq 1 ]; then

 -eq 0 ]; then


ies -lt 10 ]; then
patches"

atches"


ow --abbrev-commit -s
nt %h %<|(32)%an %s" $i
'git show --abbrev-commit -s
" $i`
    cd ~/linux.git
    cmp=1
    up_eq=`git log --reverse --abbrev-commit
--pretty=format:"lo %h %<|(32)%an %s"
$utracking..$ubranch | grep "$desc"`
# lo = in local for-next
    if test "X$up_eq" = "X"; then
        up_eq=`git log --reverse --abbrev-commit
--pretty=format:"fn %h %
master..$utracking | gre
# fn = in for-next for
```

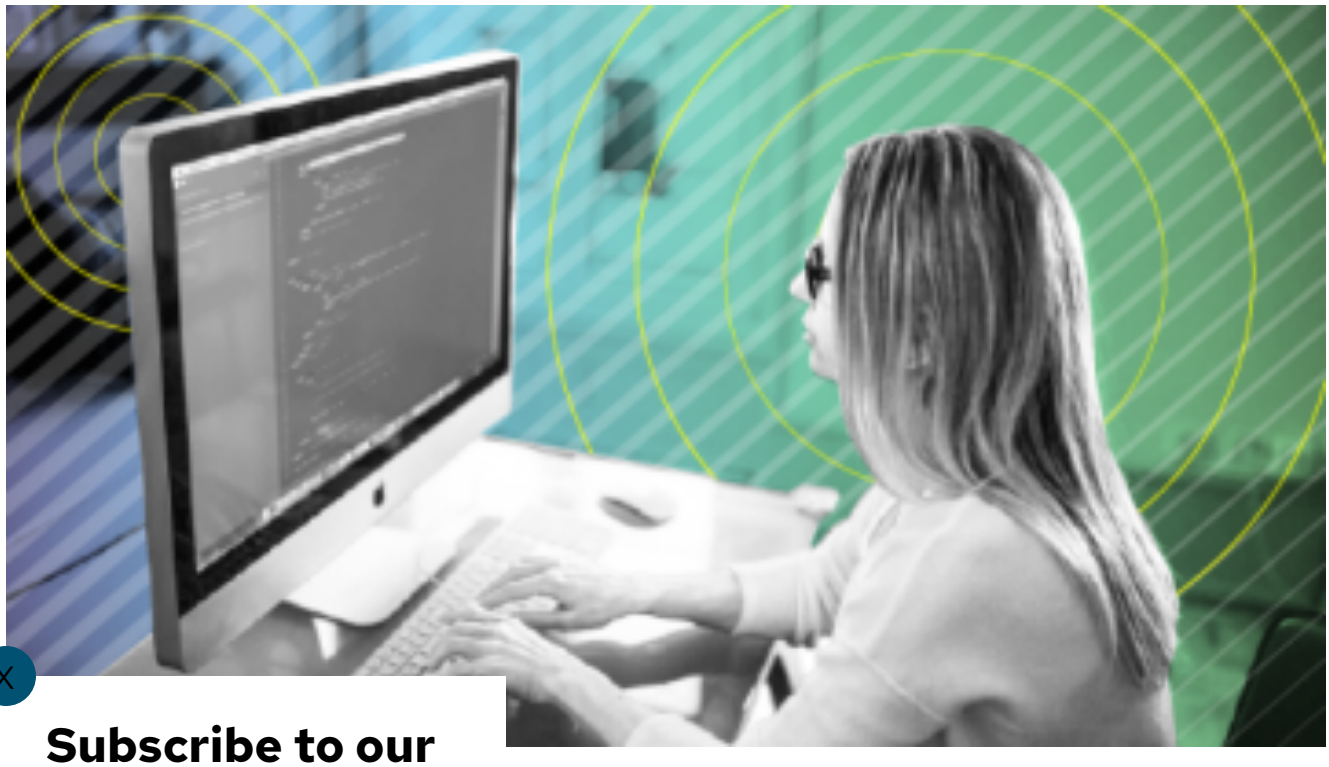
```
        if test "X$up_eq" = "X"; then
            up_eq=`grep "$desc" /tmp/gronk.gfs2`
            if test "X$up_eq" = "X"; then
                up_eq=`grep "$desc" /tmp/gronk.dlm`
                if test "X$up_eq" = "X"; then
                    up_eq="   Not found upstream"
                    cmp=0
                fi
            fi
        fi
    fi
    echo "$up_eq"
    if [ $cmp -eq 1 ] ; then
        UP_SHA1=`echo $up_eq|cut -d' ' -f2`
          '   "     ıre_them $UP_SHA1 $i" >> $script

                    $patches - 1 | bc)

                ıt generated: $script"
```

ombos and special syntax

for Bash commands and shortcuts you need to talk

ur Git skills

of the best in Git. Here are the top 10 articles about in the new year.



mplate

es, create a fairly simple template that you can use h programs, then test it.

GIT

Bob Peterson

Bob Peterson is a Linux kernel developer working for Red Hat on the GFS2 file system.

More about me

Comments

, however you can Register or Login to post a

:020

ay find built-in ġit range-diff ùseful

2020

ɔts have not been checked with 'shellcheck' ->
s (on Debian/Ubuntu there is /bin/echo, no
/usr/bin/echo like on RedHat), inconsistencies (git and /usr/bin/git),...

Related Content

Automate image processing with this Bash script

Use this Python script to simulate Babbage's Difference Engine

Using Bash traps in your scripts

X

Subscribe to our newsletter.

Stay on top of the latest thoughts, strategies and insights from enterprising peers.

kly newsletter

elect your country o

SUBSCRIBE

ABOUT THIS SITE

The opinions expressed on this website are those of each author, not of the author's employer or of Red Hat.

Opensource.com aspires to publish all content under a **Creative Commons license** but may not be able to do so in all cases. You are responsible for ensuring that you have the necessary permissic
and the Red Hat logo are trademarks

States and other countries.

A note on advertising: Opensource.com does not sell advertising on the site or in any of its newsletters.

CONTACT

Follow us @opensource.com on Twitter

Like Opensource.com on Facebook

Watch us at Opensource.com

Contact