Scripting Git

CB Bailey

@hashpling

Bloomberg

Saturday, 14th April 2018
Where would you extend Git?

- Helper scripts
- Hooks
- Bisecting
- Automated updates to a repository
Why script?

- No “first-tier” API for Git
- A relatively stable interface
- Low barrier to working results
Know your plumbing

- The most stable Git commands
- Listed in `git help git`
- But sometimes it's not that simple
porcelain:

`git reset --soft <commit id>`

plumbing:

`git update-ref HEAD <commit id>`
**porcelain:**

`git reset <commit id> .`

**plumbing:**

`git read-tree -m <commit id>`
`porcelain:`

`git branch`

`plumbing:`

`git symbolic-ref -q HEAD`
Check your exit codes

Most Git commands:

- Exit with status 0 on success
- Exit with non-zero status (usually 128) on failure
Delimiting with NUL

- Many commands use \(-z\) (sometimes \(--nul\))
- Uses ASCII NUL instead of ASCII LF as delimiter
- May also turn off escaping and quoting
Consider concurrency and sharing

- Do you need the working tree?
- Or just a working tree? (git worktree)
- Can you use a temporary index? (GIT_INDEX_FILE)
- What parallel operations do you support?
Use “atomic” operations where possible

- `git push --force-with-lease`
- `git update-ref <ref> <newval> <oldvalue>`
A quick sample: `git ff`

- Fast-forward a branch that isn't checked out
The shell function I always write

die () {
    echo >&2 "*$"
    exit 1
}

Do we have both parameters?

```bash
if [[ $# -ne 2 ]]
then
die "Syntax: $0 <branch-name> <target-commit-id>"
fi
```
Is it a branch?

current=$(git rev-parse --verify --quiet "refs/heads/$1") ||
die "$1 is not a valid branch name"
Is it the current branch?

```bash
if [[ "$(git symbolic-ref --quiet HEAD)" = "refs/heads/$1" ]] then
die "$1 is currently checked out, use 'git merge --ff-only' to fast-forward"
fi
```
Can this be resolved as a commit?

target=$(git rev-parse --verify --quiet "$2^{}" ) ||
die "Could not resolve $2 as a commit"
Is this really a fast-forward?

git merge-base --is-ancestor "$current" "$target" ||
die "Cannot fast forward, $1 is not an ancestor of $2"
git update-ref -m "Fast-forwarded by script to $2"
"refs/heads/$1" "$target" "$current"
git bisect
A brief primer on git bisect

- You identify one “bad” commit
- ... and one or more “good” commits
- Test commits as prompted
- Use `git bisect run`
Considerations for automation

- You need to be sure about your test
- Your test should be as fast as possible
Testing for a regression

• You fixed a bug...
• ...but now it's back. What went wrong?
Things to consider scripting

- Patching around known issues
- Use exit code 125 to signal ‘skip’
- Marking good or bad based on commit position
Scripting merges
Never script `git merge`

- ...for automated updates
- ...or be careful if you do
git merge deconstructed

1. Populate the index with three trees
2. Collapse "easy" cases into stage 0
3. Perform file level merges on the remaining entries
Merging into the index

```
git read-tree -m
```

Other useful options:

- `--trivial` / `--aggressive`
- `-u` update the working tree
- `-i` don't check the working tree
File level merges

- `git merge-index`
- `git merge-one-file`
git merge -s resolve

recursive is more sophisticated.
Key points

• Follow best practices for your language
• Choose the most suitable Git commands for your task
• Understand the whole context for your scripts
Tech at Bloomberg

Explore our repo: github.com/bloomberg

Check out our blog: TechAtBloomberg.com

We're hiring: bloomberg.com/careers/technology