Advanced Git

A Taster Menu

Abizer Nasir ◇ @abizern ◇ 365git.tumblr.com ◇ abizern.org

Scope

- This is not a full discussion about Git.
- Just a taster menu. A presentation of workflows and tips that should get you thinking about more than committing, merging, pushing and pulling.
- Git is there to support your workflow (within reason), you shouldn’t be the one jumping through hoops to work with it.

132 Commands

add
am
archive
bisect
branch
bundle
checkout
cherry-pick
citool
clean
clone
commit
describe
diff
fetch
format-patch
gc
grep
gui
init
log
merge
mv
notes
pull
push
rebase
reset
revert
rm
shortlog
show
stash
status
submodule
tag
gitk
apply
checkout-index
commit-tree
hash-object
index-pack
merge-file
merge-index
mktag
mktree
pack-objects
prune-packed
read-tree
symbolic-ref
unpack-objects
update-index
update-ref
write-tree
cat-file
diff-files
diff-index
diff-tree
for-each-ref
ls-files
ls-remote
ls-tree
merge-base
name-rev
pack-redundant
rev-list
show-index
show-ref
tar-tree
unpack-file
var
verify-pack
check-attr
check-ref-format
fmt-merge-msg
mailinfo
mailsplit
merge-one-file
patch-id
peek-remote
sh-setup
stripspace
daemon
fetch-pack
http-backend
send-pack
update-server-info
http-fetch
http-push
parse-remote
receive-pack
shell
upload-archive
upload-pack
config
fast-export
fast-import
filter-branch
lost-found
mergetool
pack-refs
prune
reflog
relink
remote
repack
replace
repo-config
annotate
blame
cherry
count-objects
difftool
fsck
get-tar-commit-id
help
instaweb
merge-tree
rerere
rev-parse
show-branch
verify-tag
whatchanged
archimport
cvsexportcommit
cvsimport
cvsserver
imap-send
quiltimport
request-pull
send-email
svn

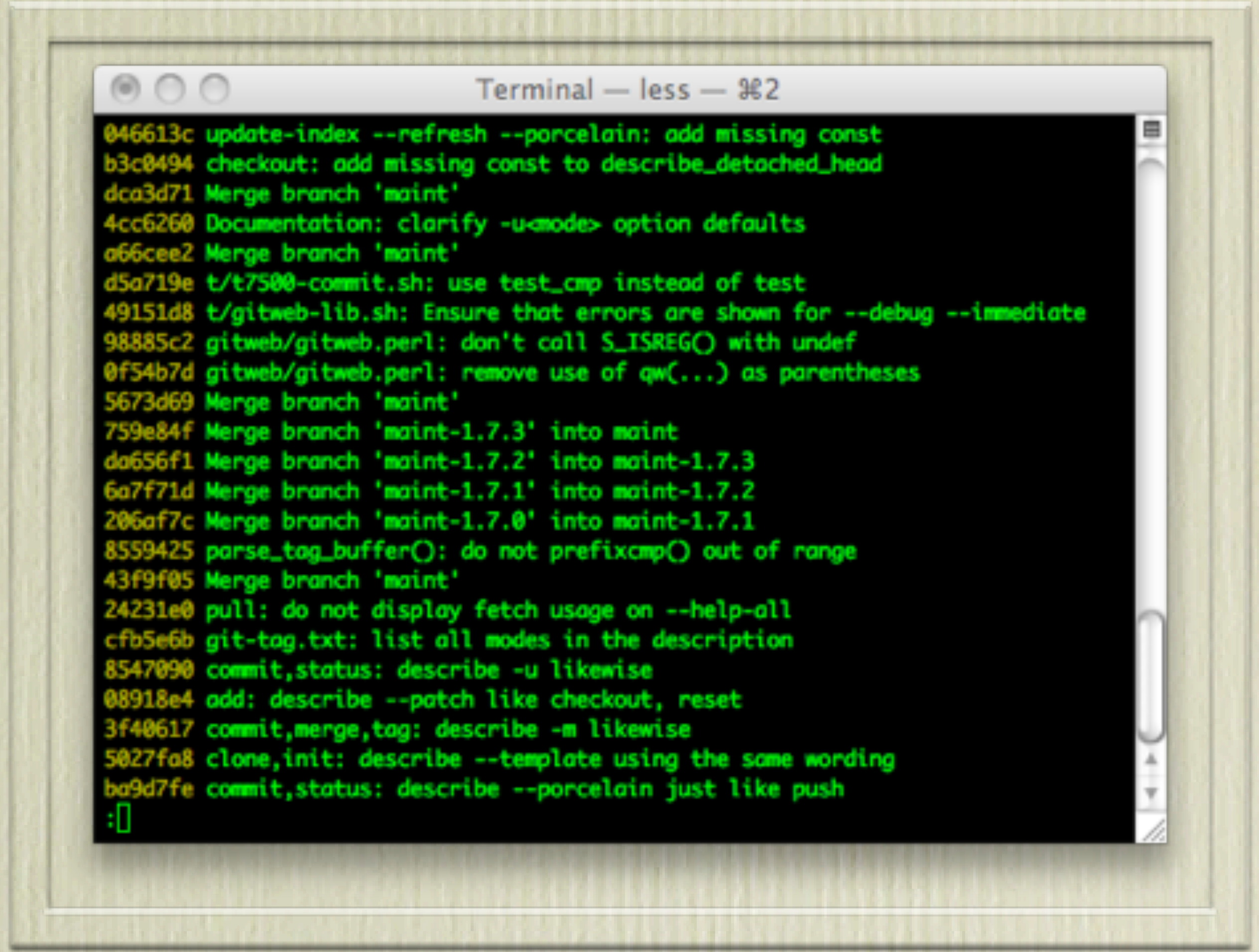
Porcelain / Plumbing

add
am
archive
bisect
branch
bundle
checkout
cherry-pick
citool
clean
clone
commit
describe
diff
fetch
format-patch
gc
grep
gui
init
log
merge
mv
notes
pull
push
rebase
reset
revert
rm
shortlog
show
stash
status
submodule
tag
gitk
apply
checkout-index
commit-tree
hash-object
index-pack
merge-file
merge-index
mktag
mktree
pack-objects
prune-packed
read-tree
symbolic-ref
unpack-objects
update-index
update-ref
write-tree
cat-file
diff-files
diff-index
diff-tree
for-each-ref
ls-files
ls-remote
ls-tree
merge-base
name-rev
pack-redundant
rev-list
show-index
show-ref
tar-tree
unpack-file
var
verify-pack
check-attr
check-ref-format
fmt-merge-msg
mailinfo
mailsplit
merge-one-file
patch-id
peek-remote
sh-setup
stripspace
daemon
fetch-pack
http-backend
send-pack
update-server-info
http-fetch
http-push
parse-remote
receive-pack
shell
upload-archive
upload-pack
config
fast-export
fast-import
filter-branch
lost-found
mergetool
pack-refs
prune
reflog
relink
remote
repack
replace
repo-config
annotate
blame
cherry
count-objects
difftool
fsck
get-tar-commit-id
help
instaweb
merge-tree
rerere
rev-parse
show-branch
verify-tag
whatchanged
archimport
cvsexportcommit
cvsimport
cvsserver
imap-send
quiltimport
request-pull
send-email
svn

Starter

Commit Messages

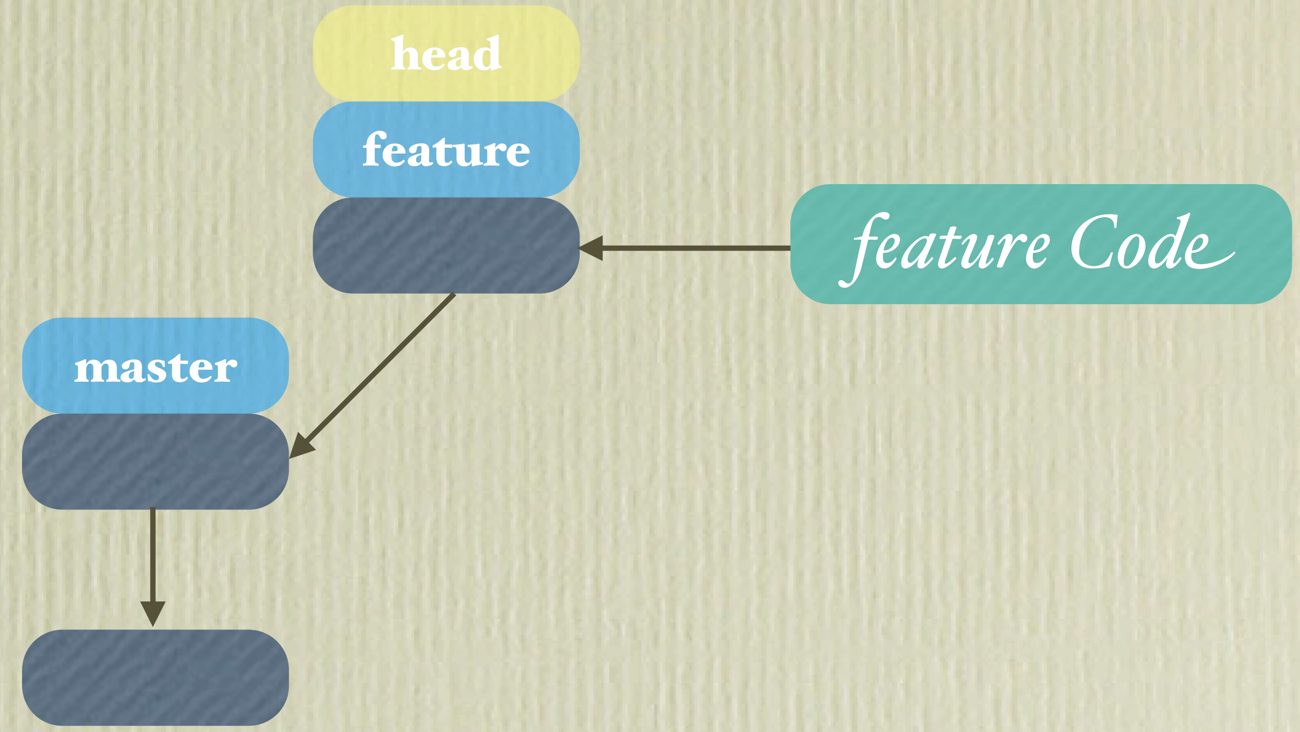
- Use the present imperative tense.
  - 'Change' not 'Changes', 'Add' not 'Adds' or 'added'.
- Describe what applying the commit will do, not what you did.
- Matches system generated output.

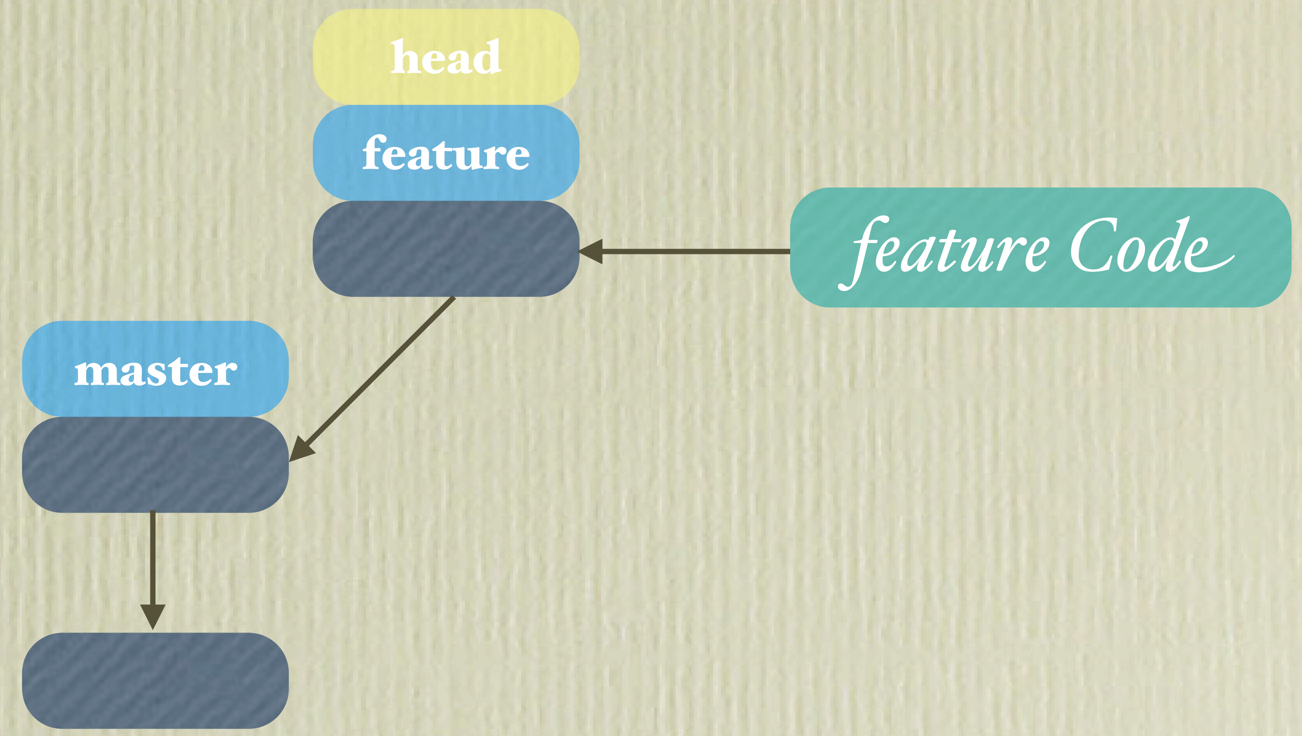


```
046613c update-index --refresh --porcelain: add missing const
b3c0494 checkout: add missing const to describe_detached_head
dca3d71 Merge branch 'maint'
4cc6260 Documentation: clarify -u<mode> option defaults
a66cee2 Merge branch 'maint'
d5a719e t/t7500-commit.sh: use test_cmp instead of test
49151d8 t/gitweb-lib.sh: Ensure that errors are shown for --debug --immediate
98885c2 gitweb/gitweb.perl: don't call S_ISREG() with undef
0f54b7d gitweb/gitweb.perl: remove use of qw(...) as parentheses
5673d69 Merge branch 'maint'
759e84f Merge branch 'maint-1.7.3' into maint
da656f1 Merge branch 'maint-1.7.2' into maint-1.7.3
6a7f71d Merge branch 'maint-1.7.1' into maint-1.7.2
206af7c Merge branch 'maint-1.7.0' into maint-1.7.1
8559425 parse_tag_buffer(): do not prefixcmp() out of range
43f9f05 Merge branch 'maint'
24231e0 pull: do not display fetch usage on --help-all
cfb5e6b git-tag.txt: list all modes in the description
8547090 commit,status: describe -u likewise
08918e4 add: describe --patch like checkout, reset
3f40617 commit,merge,tag: describe -m likewise
5027fa8 clone,init: describe --template using the same wording
ba9d7fe commit,status: describe --porcelain just like push
:
```

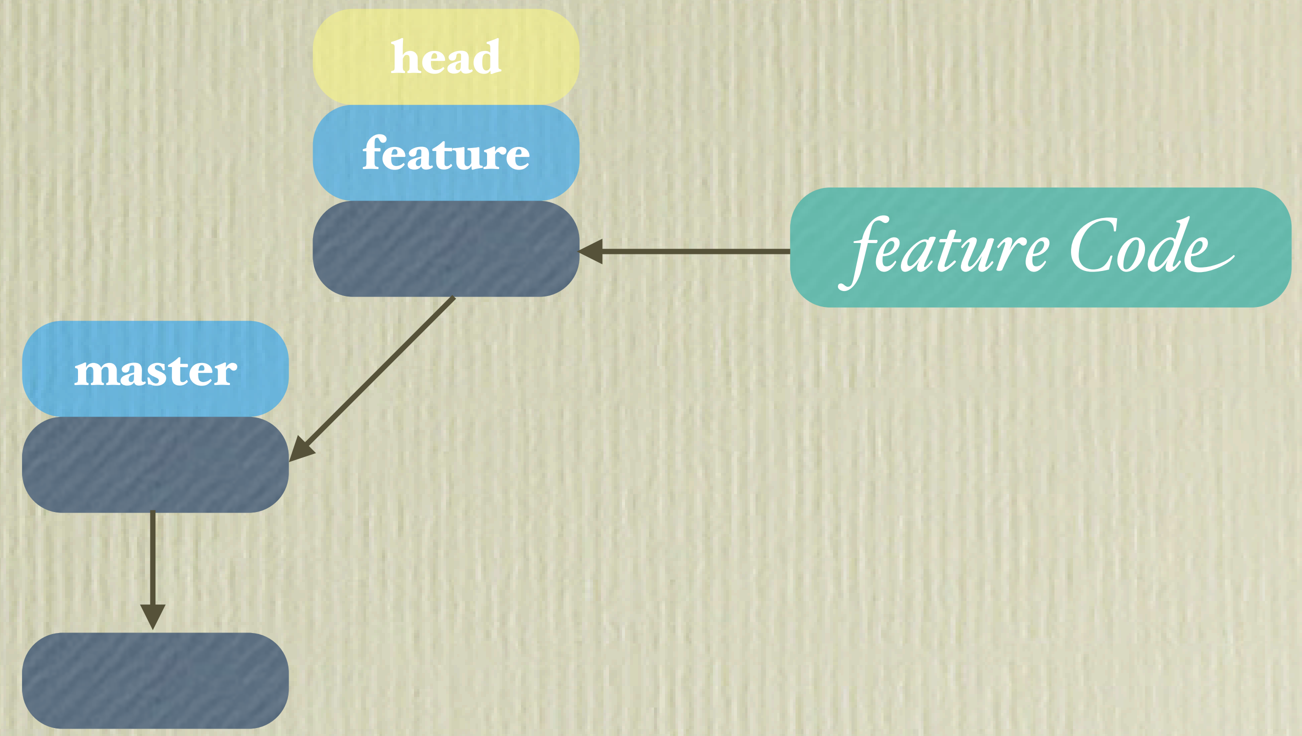
Mains

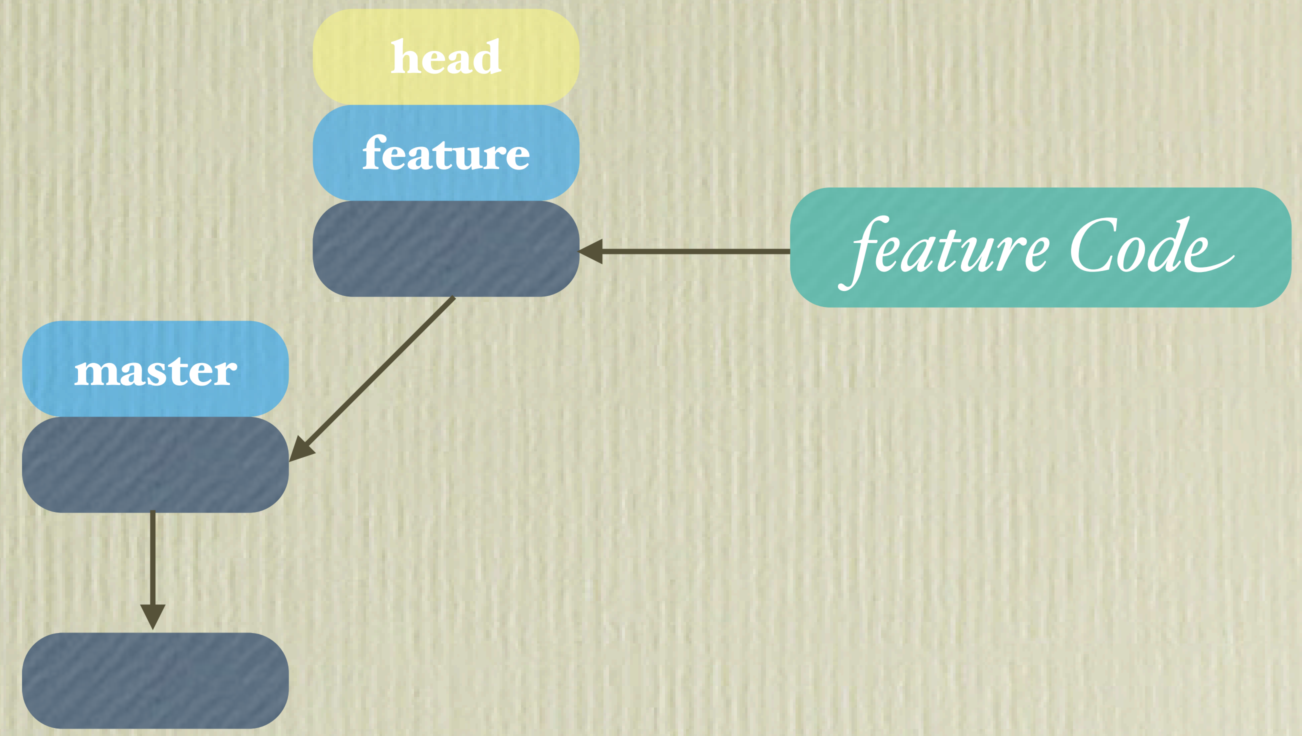
Late night bug fixing


head
feature
feature Code
master

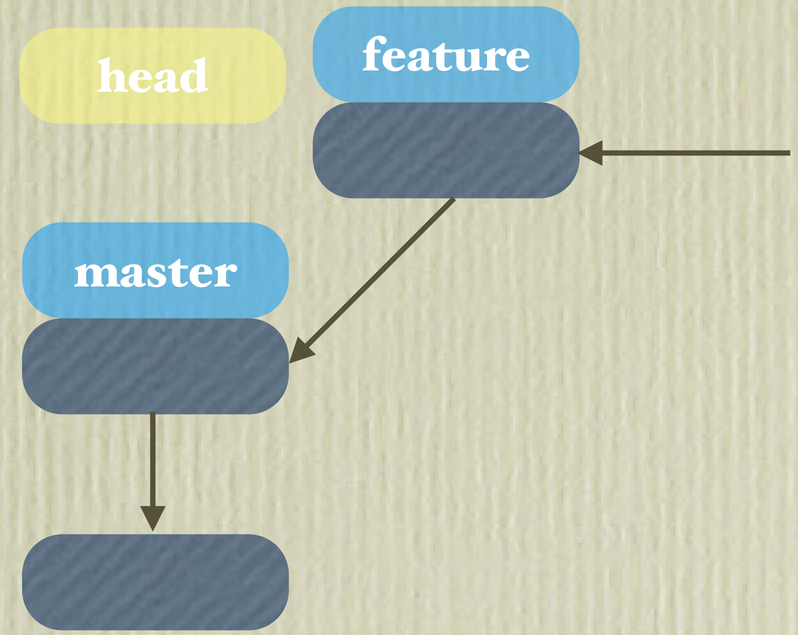


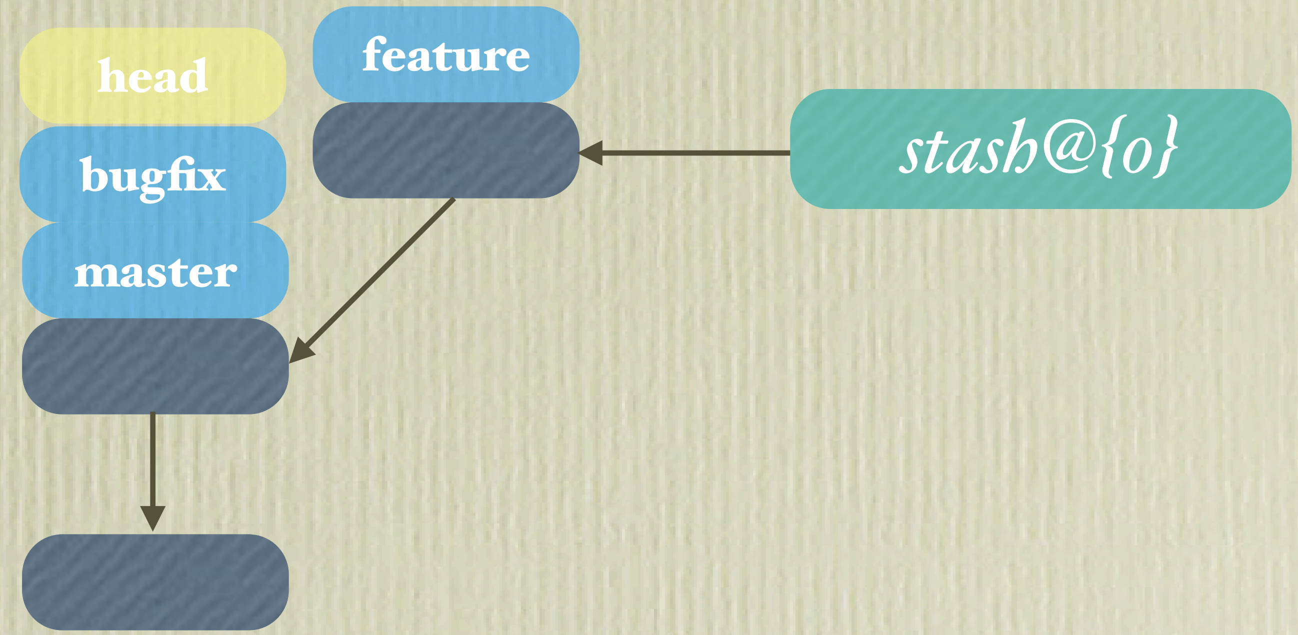
You’re working on a feature in it’s own branch.


head
feature
feature Code
master

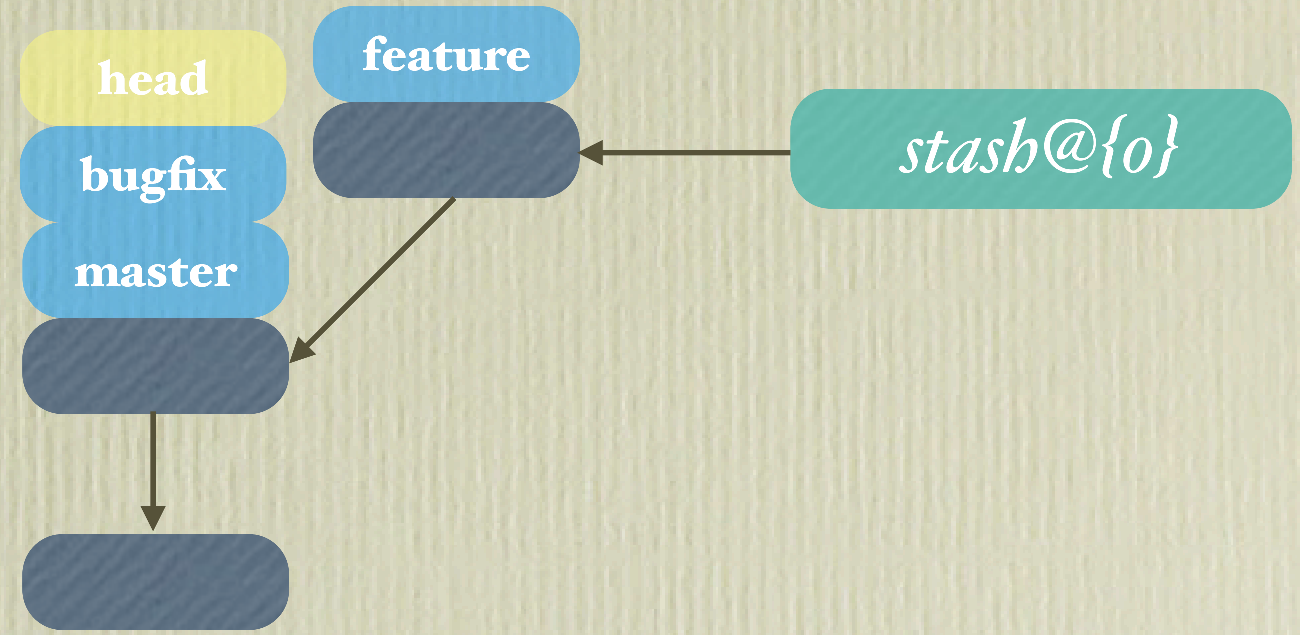


You realise that you have a bug to fix.


head
feature
master

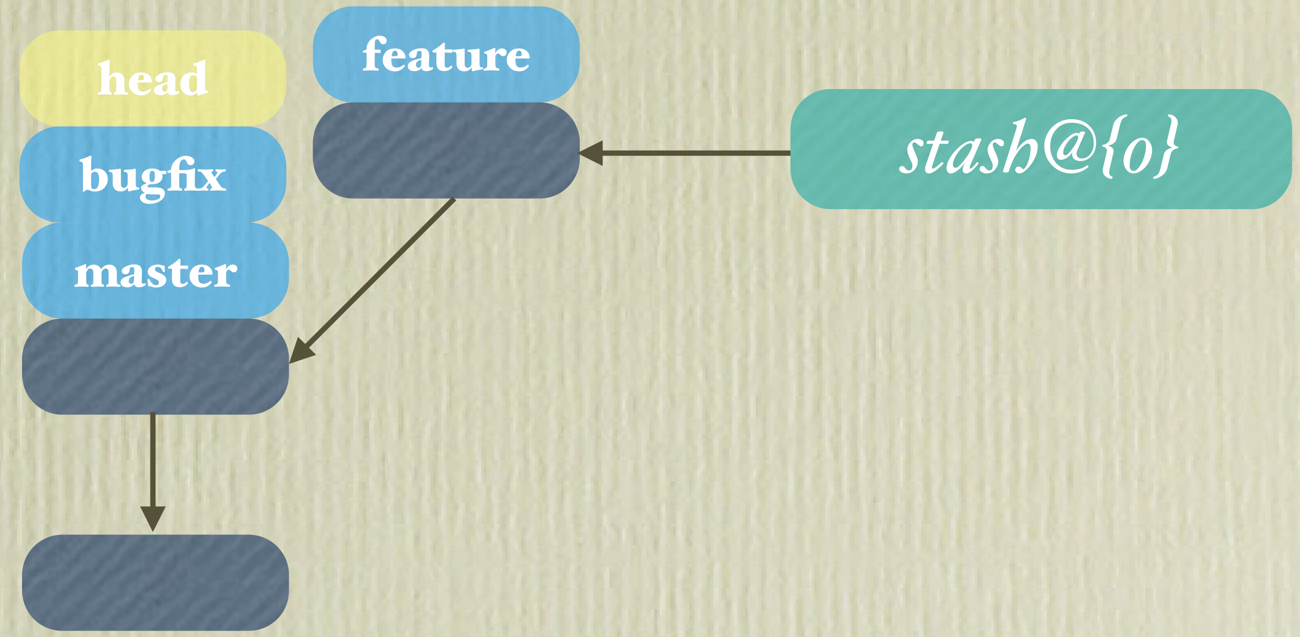


You stash your current work and create a bugfix branch off master


head
bugfix
master
feature
stash@{0}

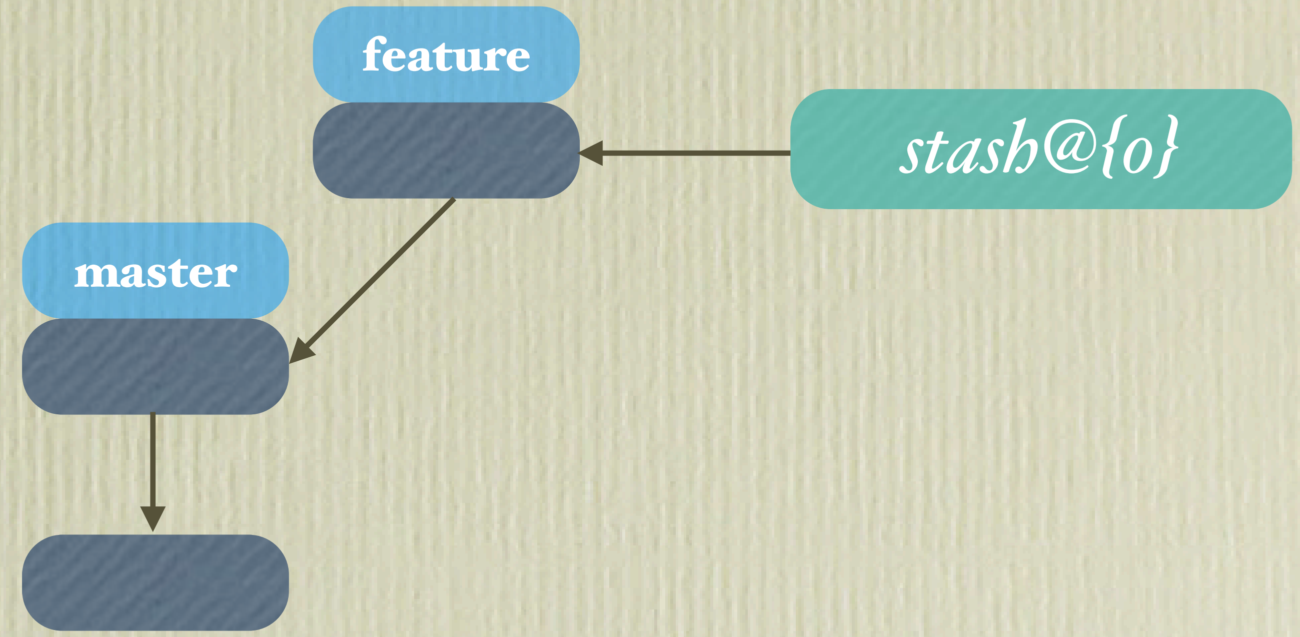
Fix the bug

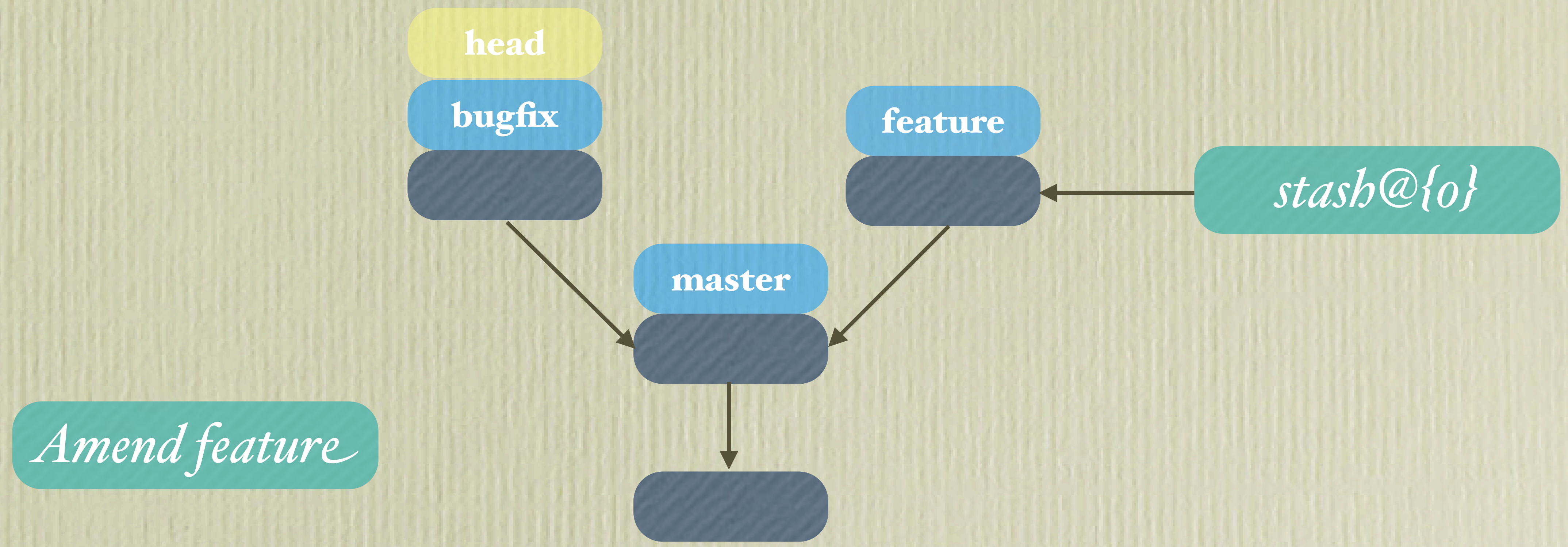
Amend feature



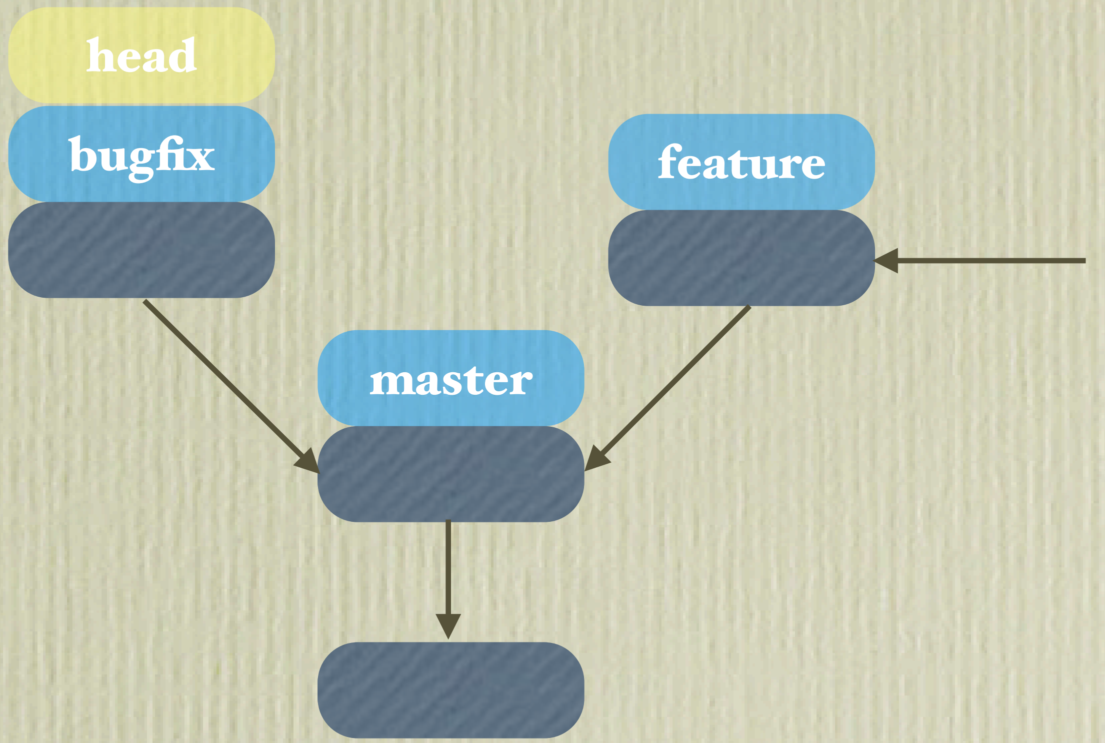
You get carried away with the fix and add a bit more to the feature.

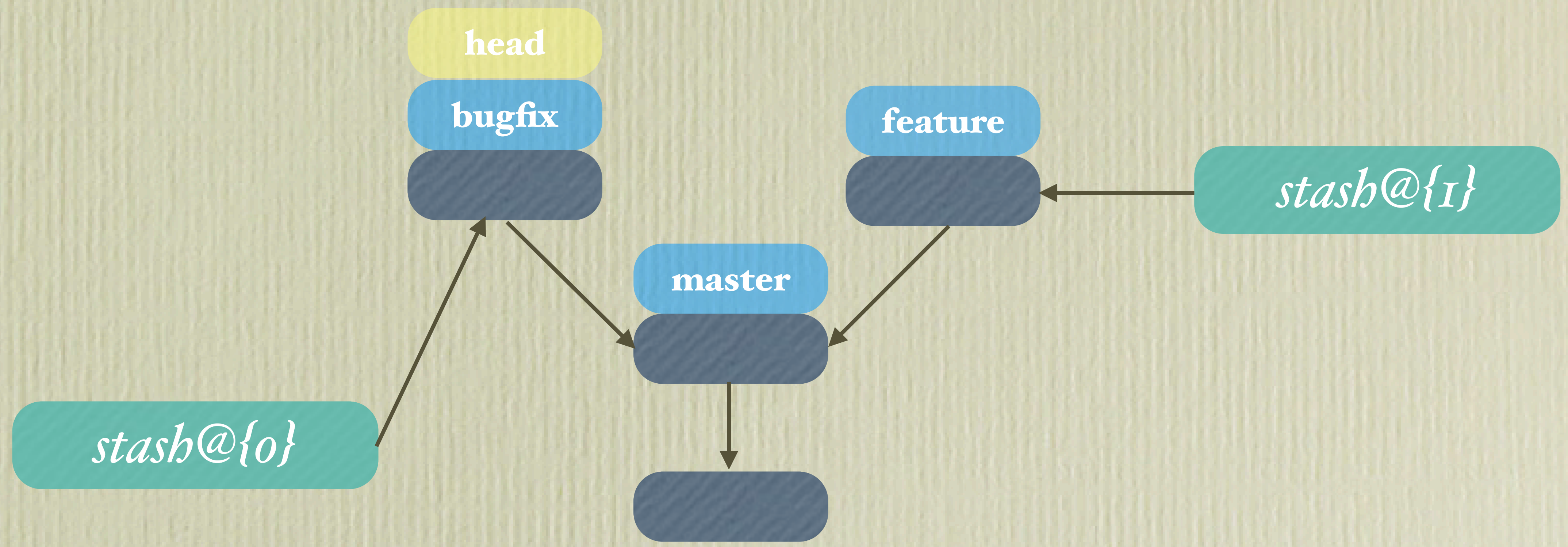
Amend feature



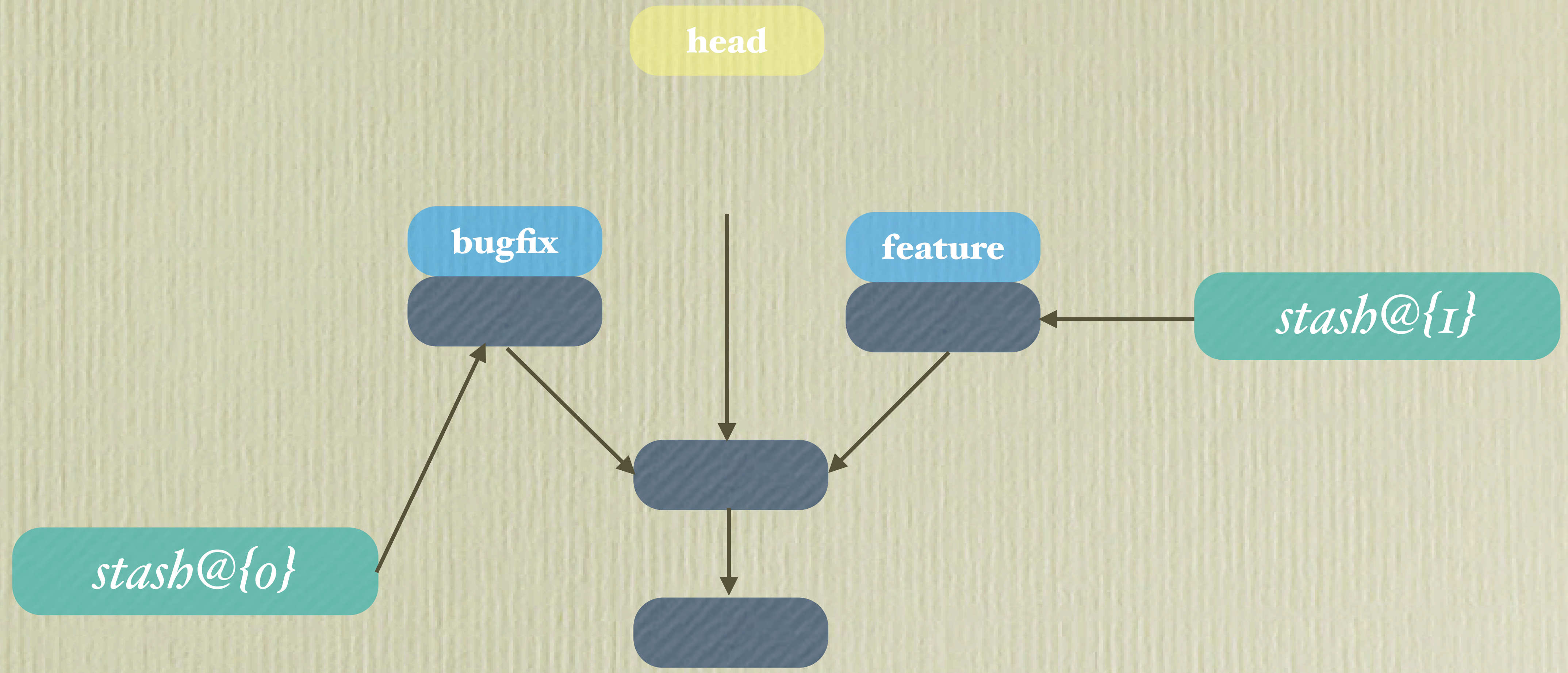


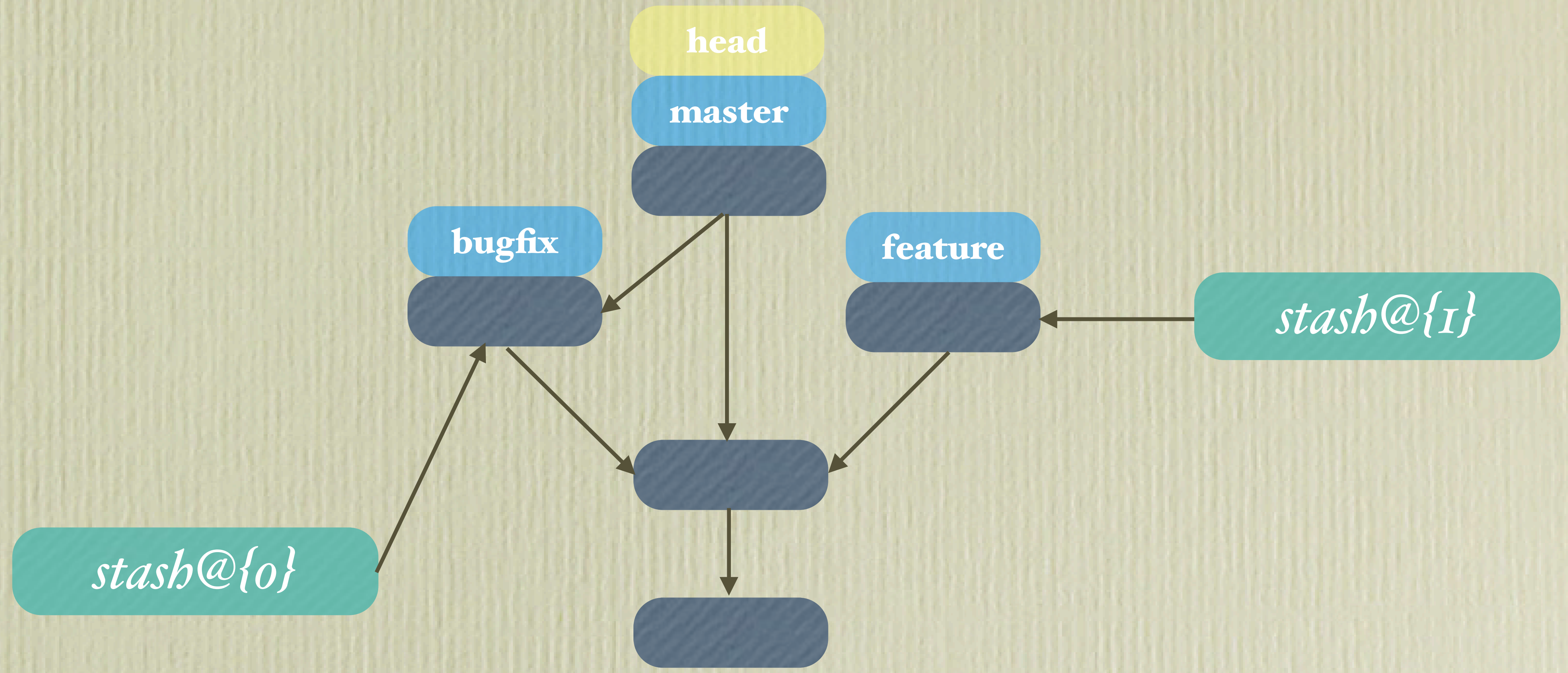
Using `git add -p` you add the bits of code that fix the bug to the index and commit just that.


head
bugfix
feature
master

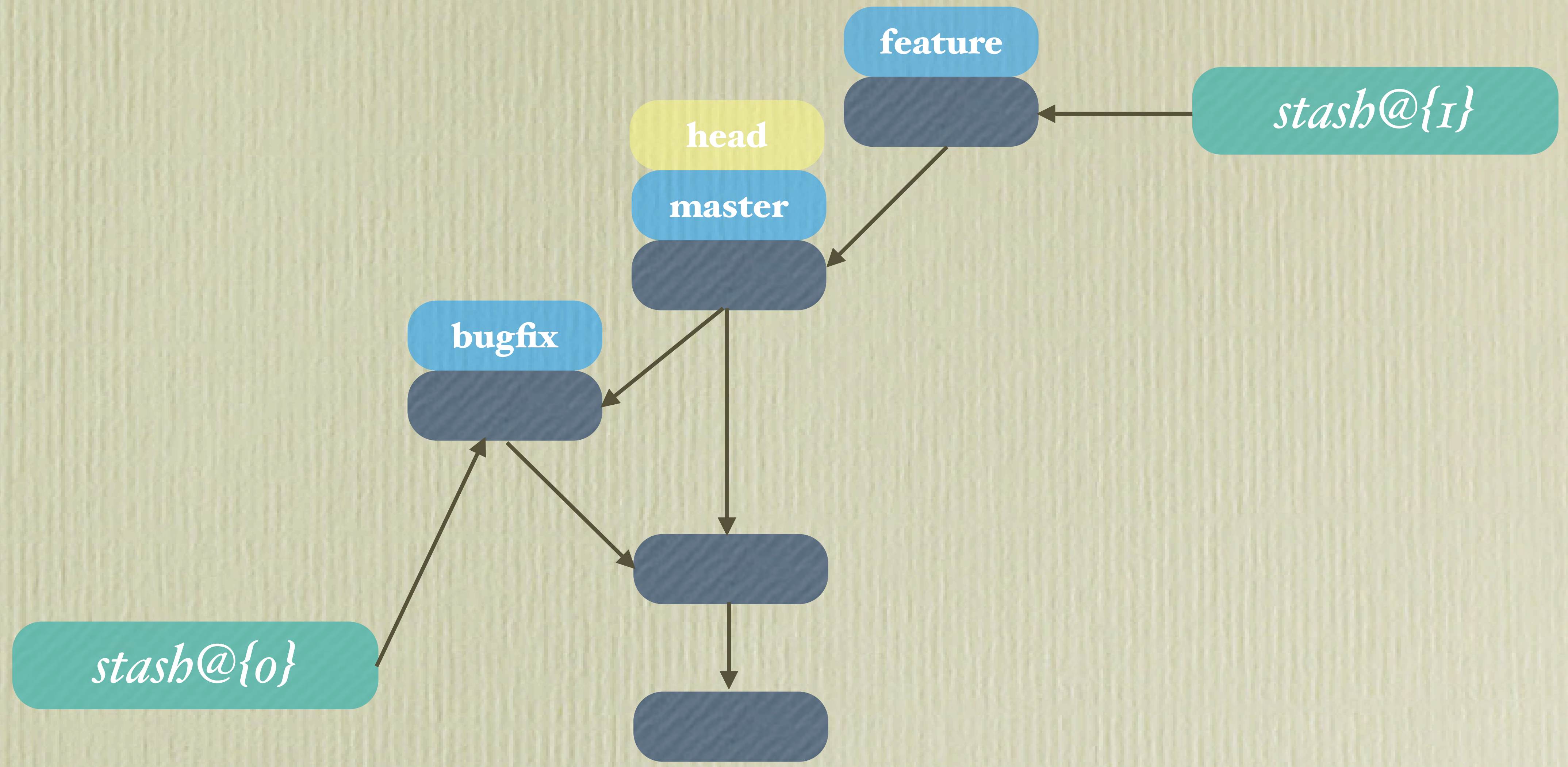


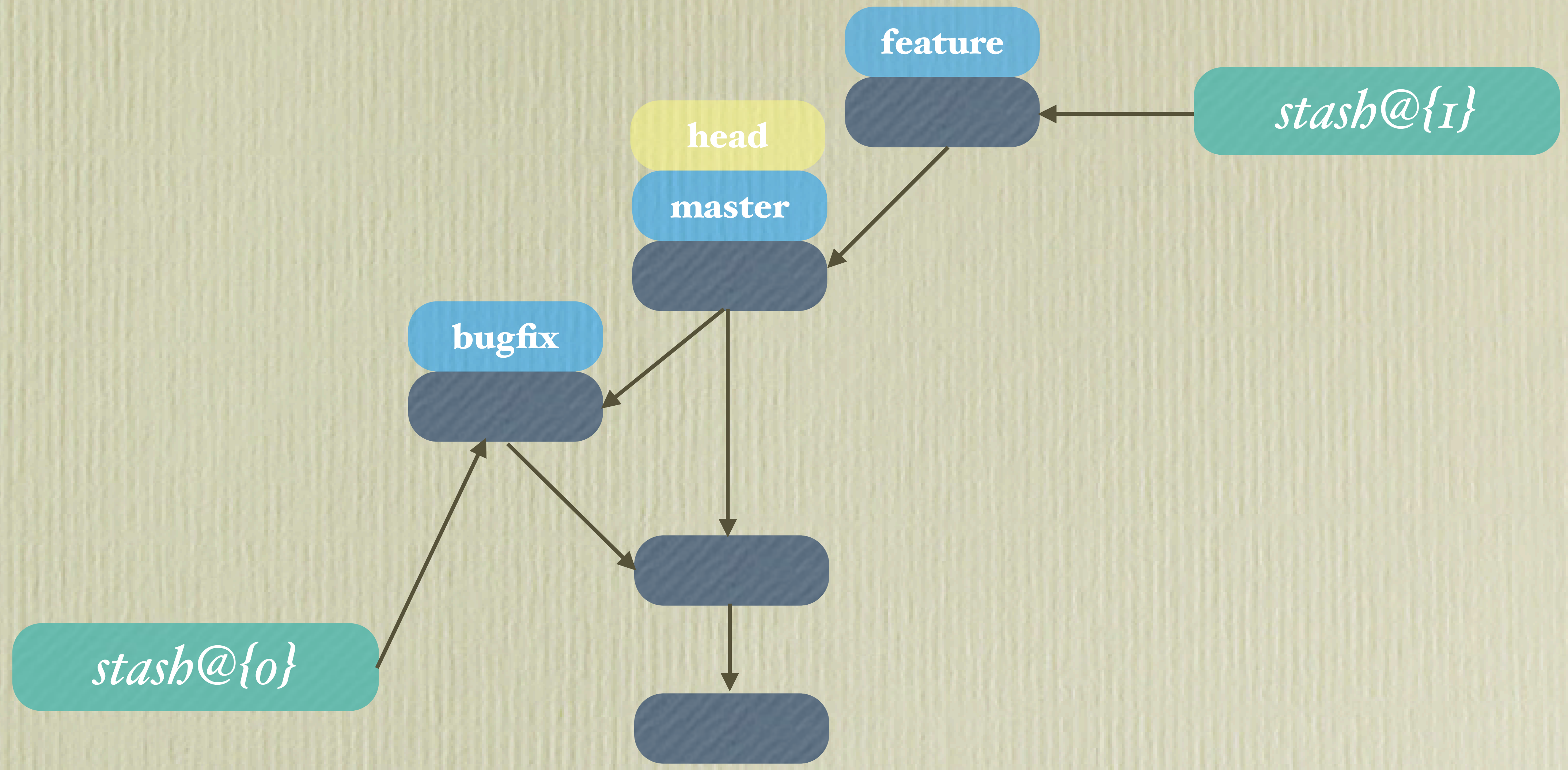
Stash the extra changes that add the feature.


head
bugfix
feature
stash@{1}
stash@{0}

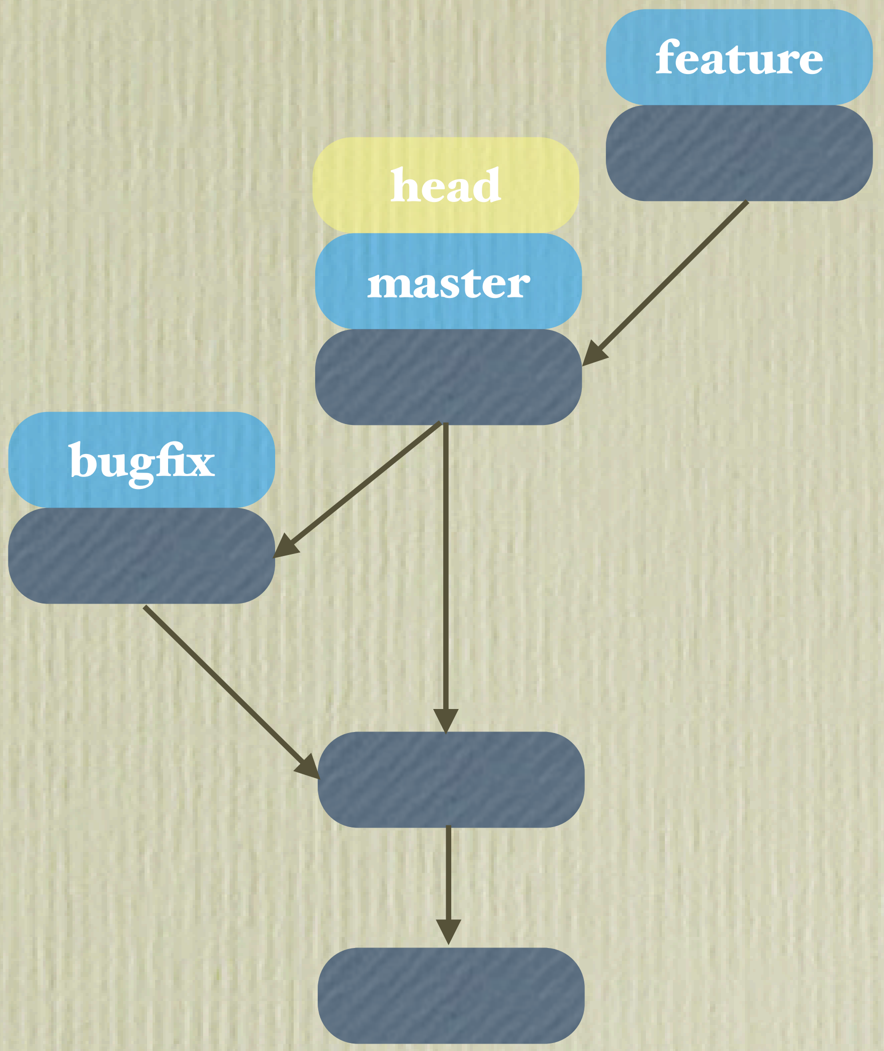


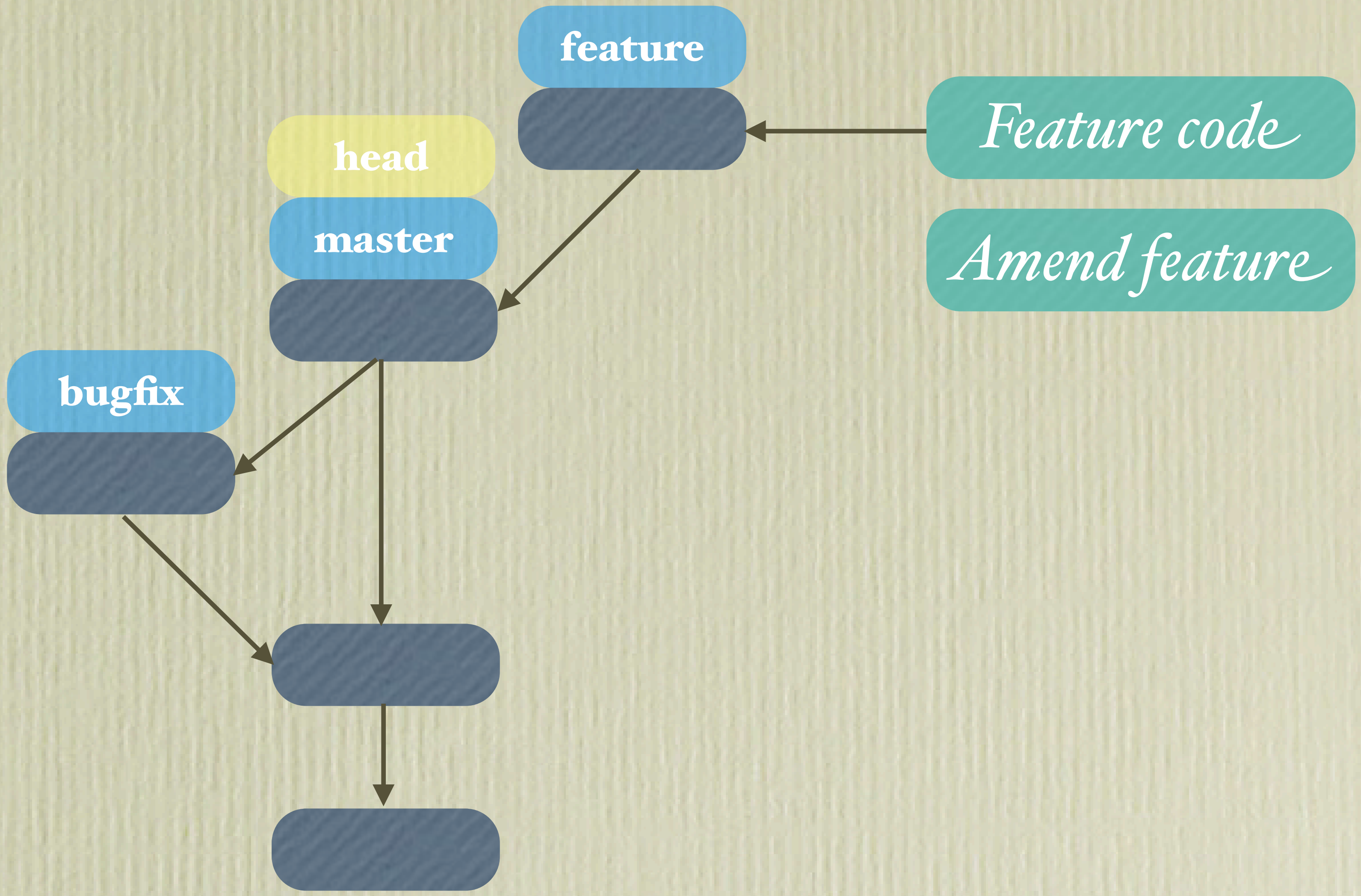
Merge the changes back into the master branch.


feature
stash@{1}
head
master
bugfix
stash@{0}



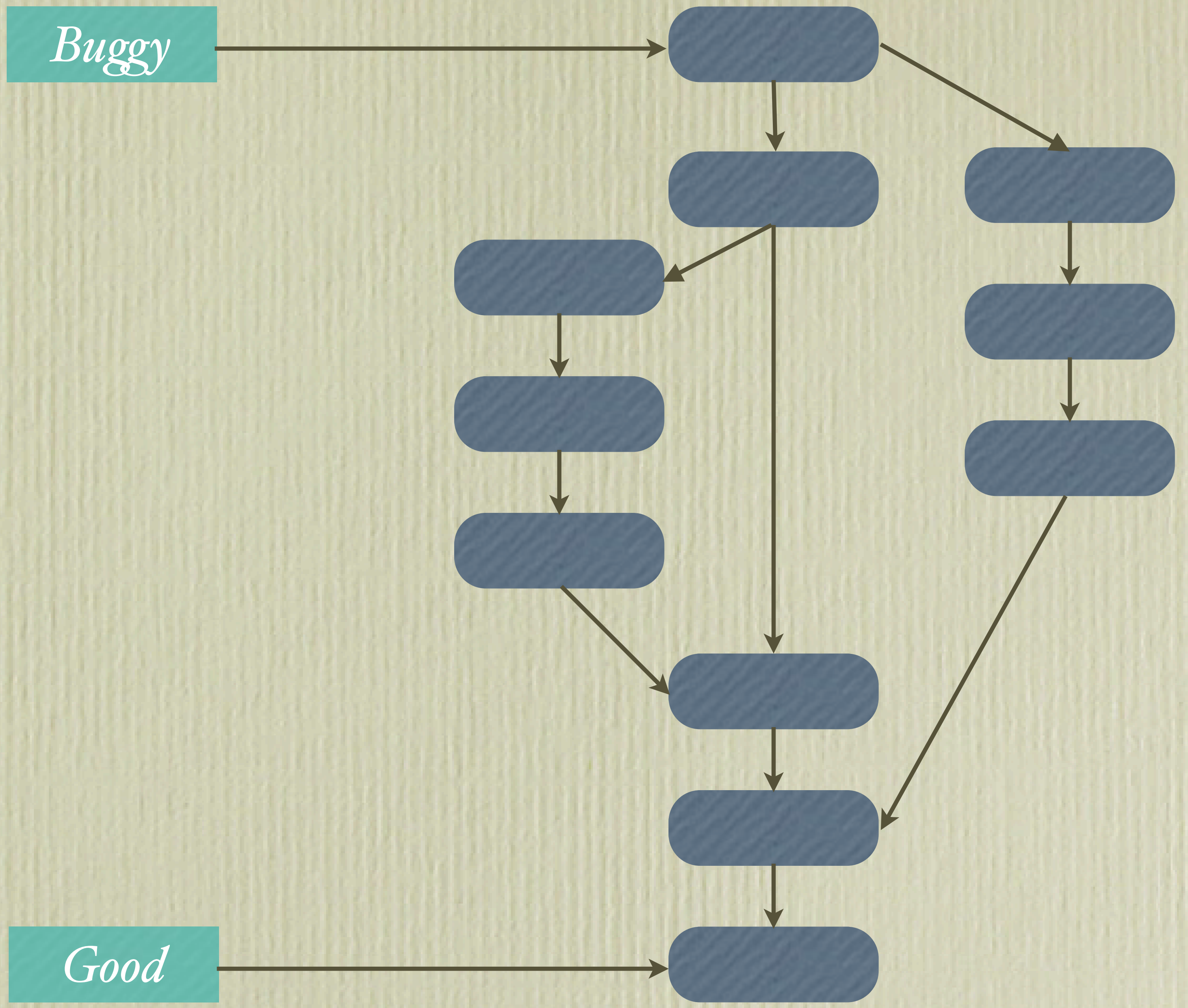
Rebase the feature branch onto the master branch

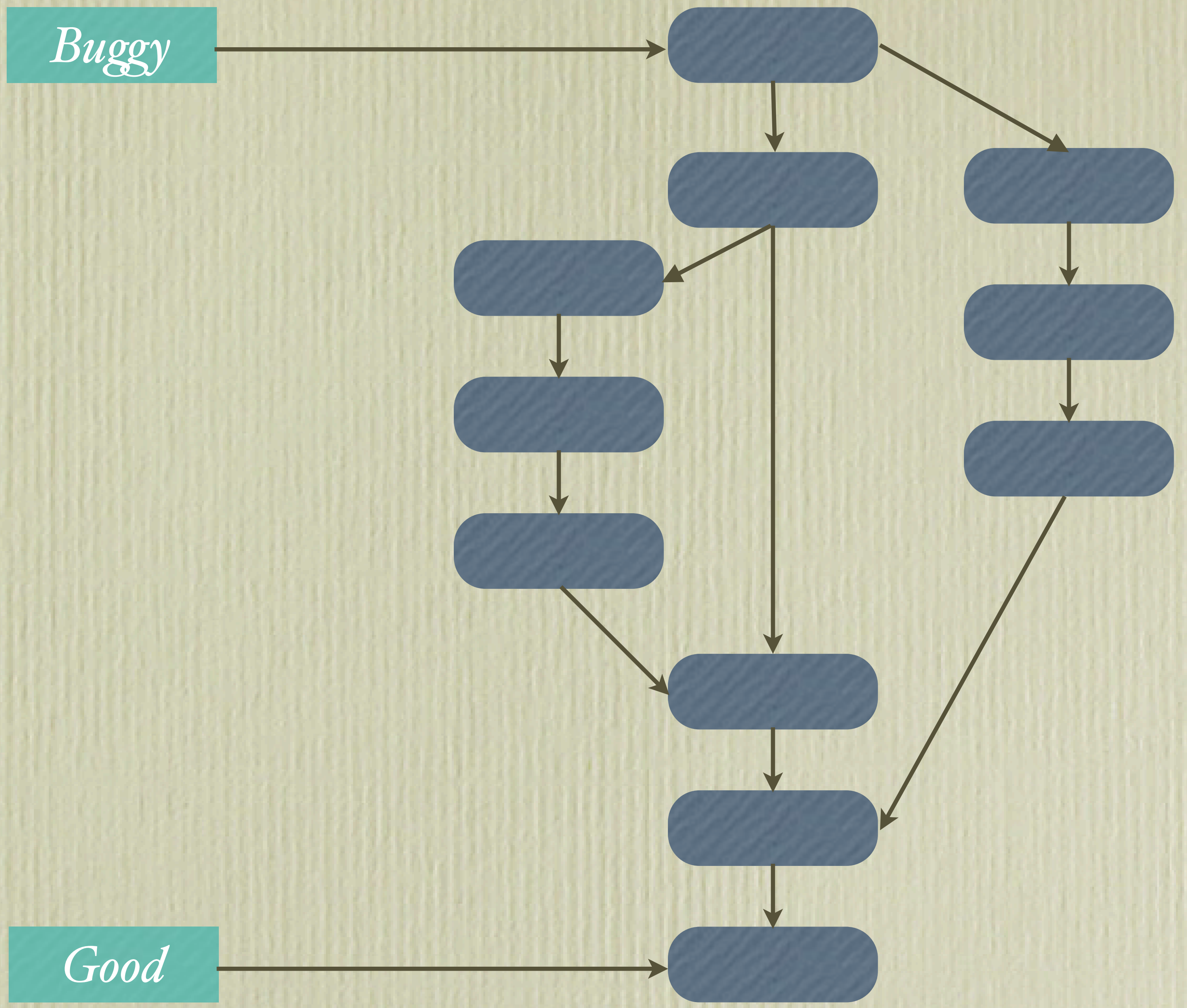

feature
head
master
bugfix



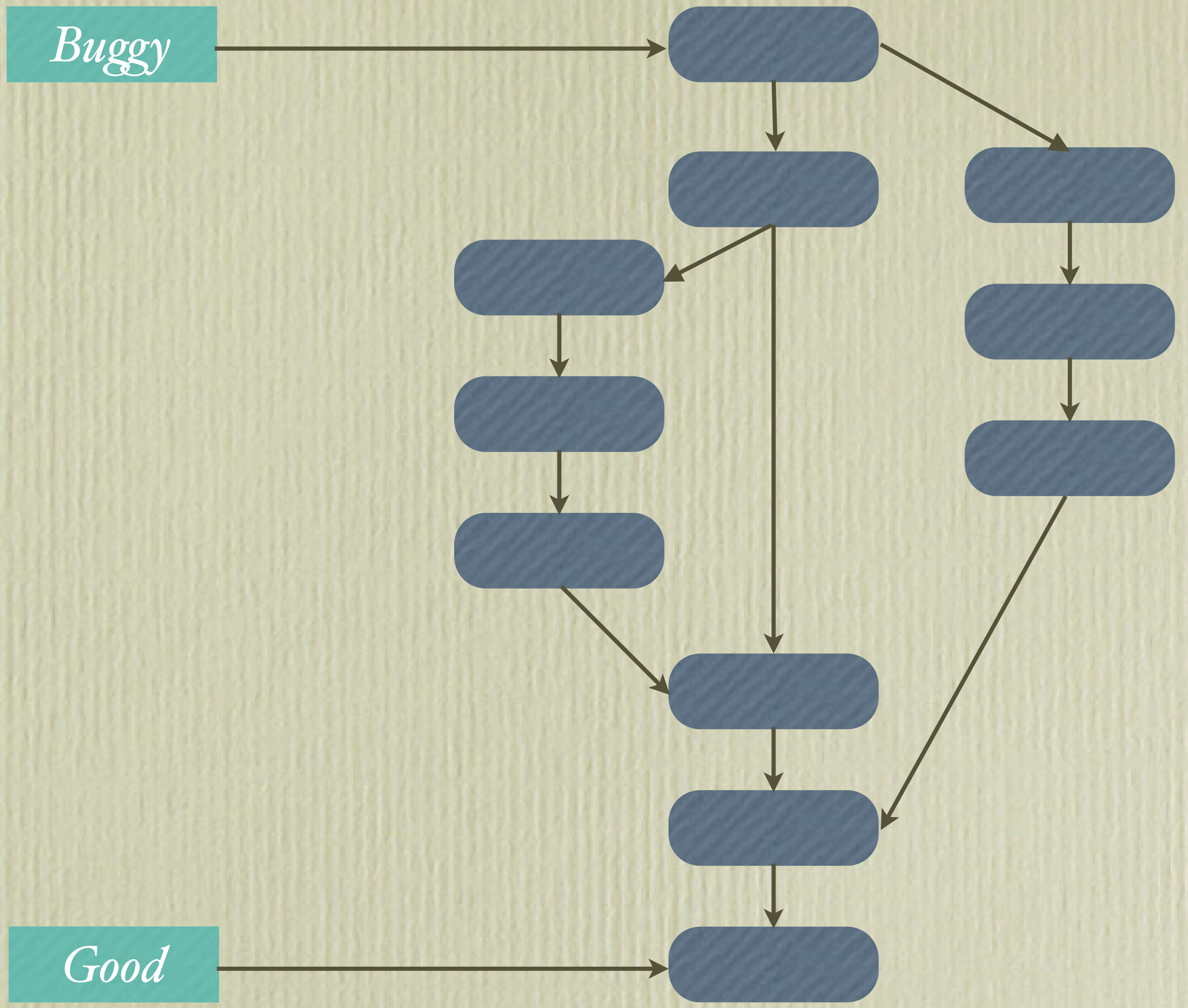
Pop the stashes onto the feature branch one at a time and fix any merge conflicts.

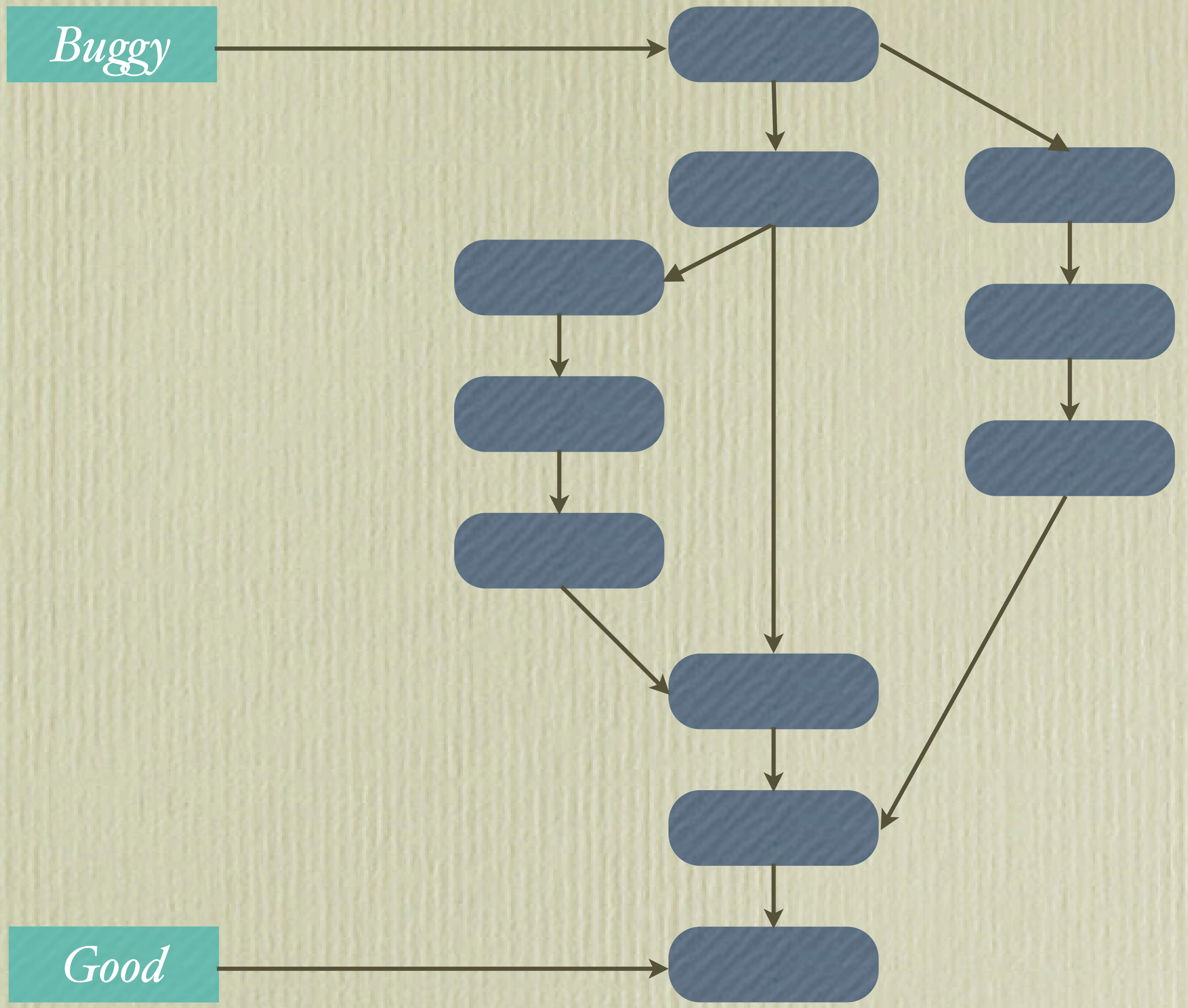
Early morning witch-hunt


Buggy
Good

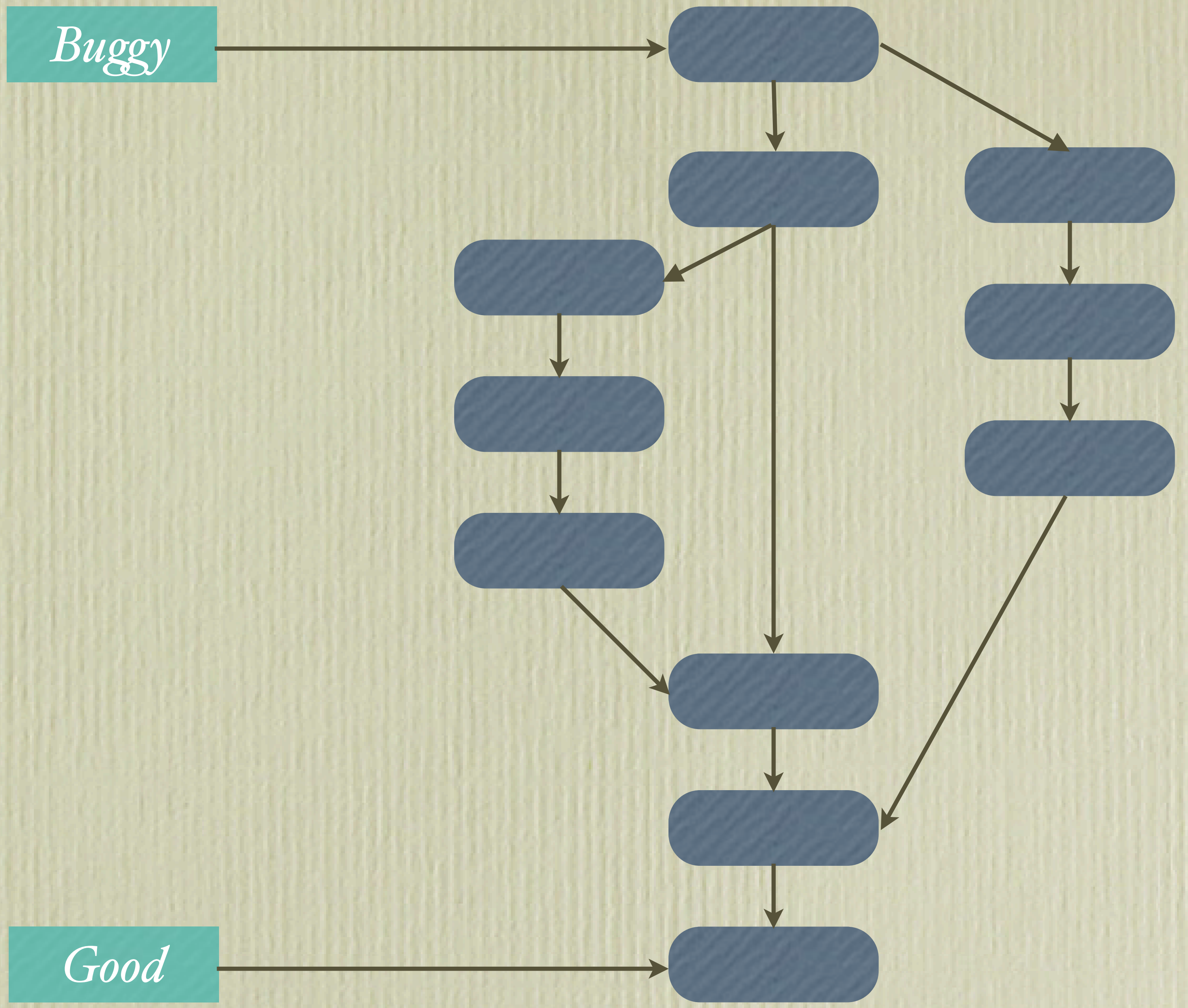


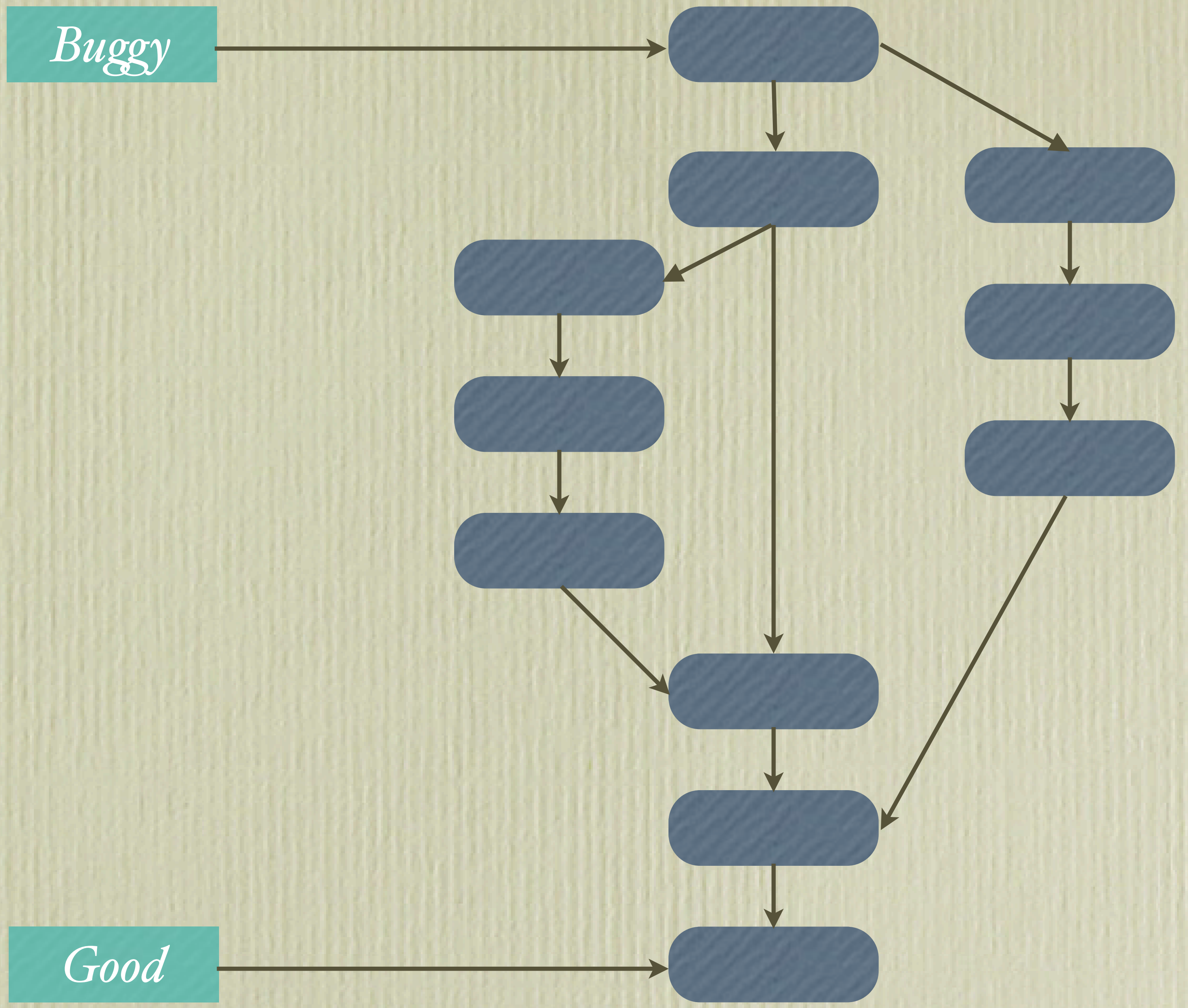
Somehow, a bug appeared between two states of the codebase. And you have to find out where.


Buggy
Good

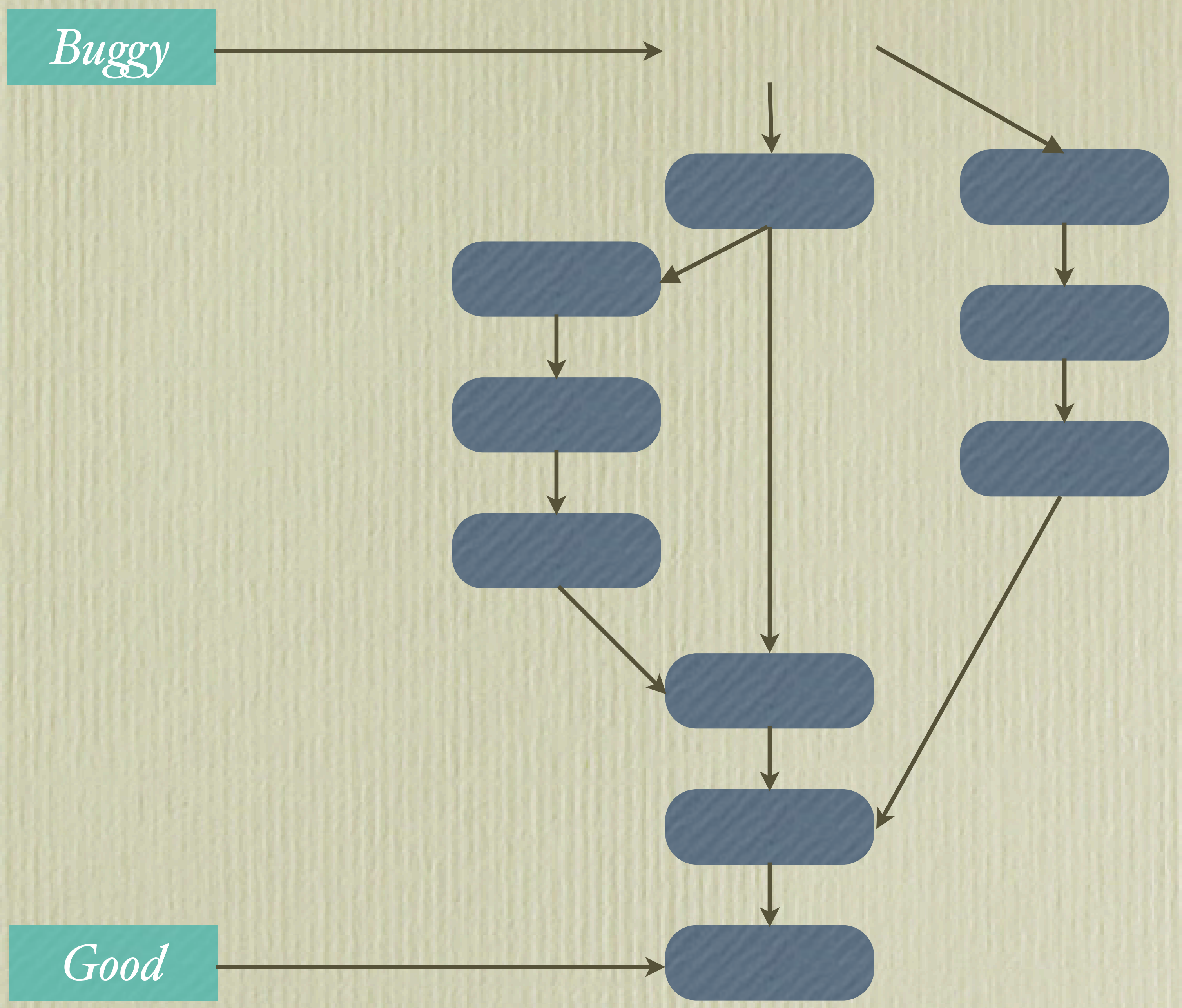


You create a test for the bug and run `git bisect`.

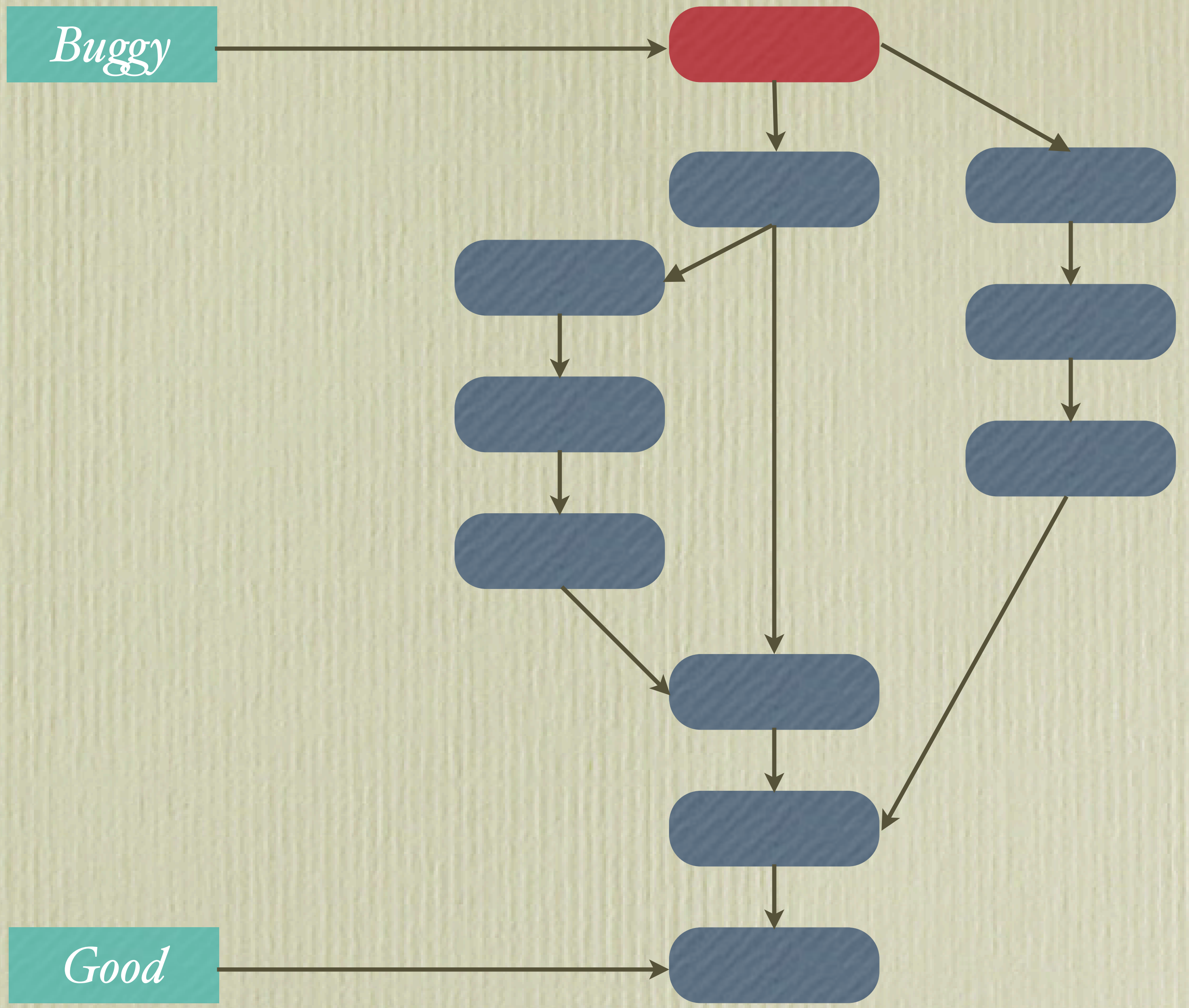

Buggy
Good



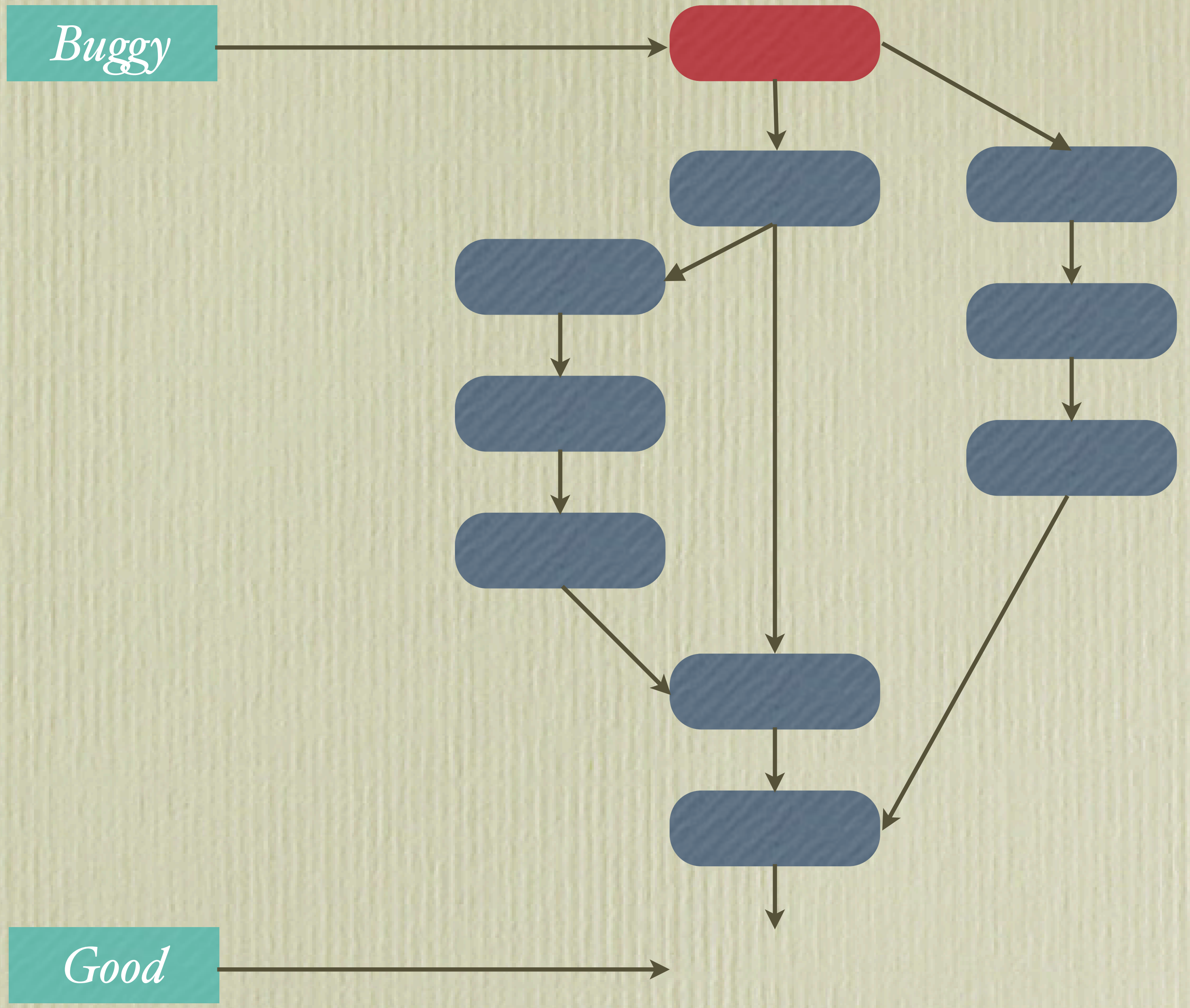
Git runs a binary search algorithm to check out commits and find the first one that causes the test to fail.



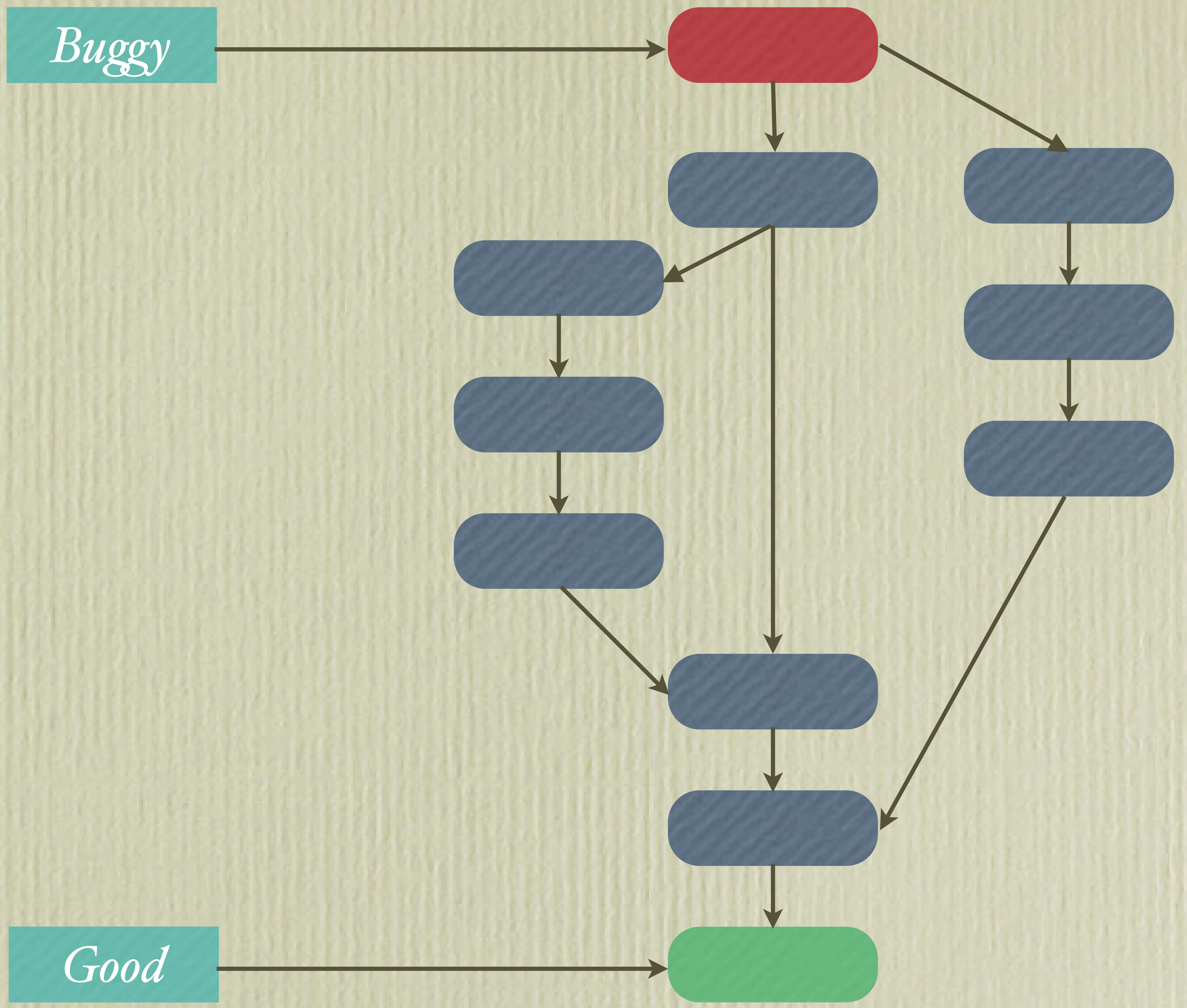
Git runs a binary search algorithm to check out commits and find the first one that causes the test to fail.



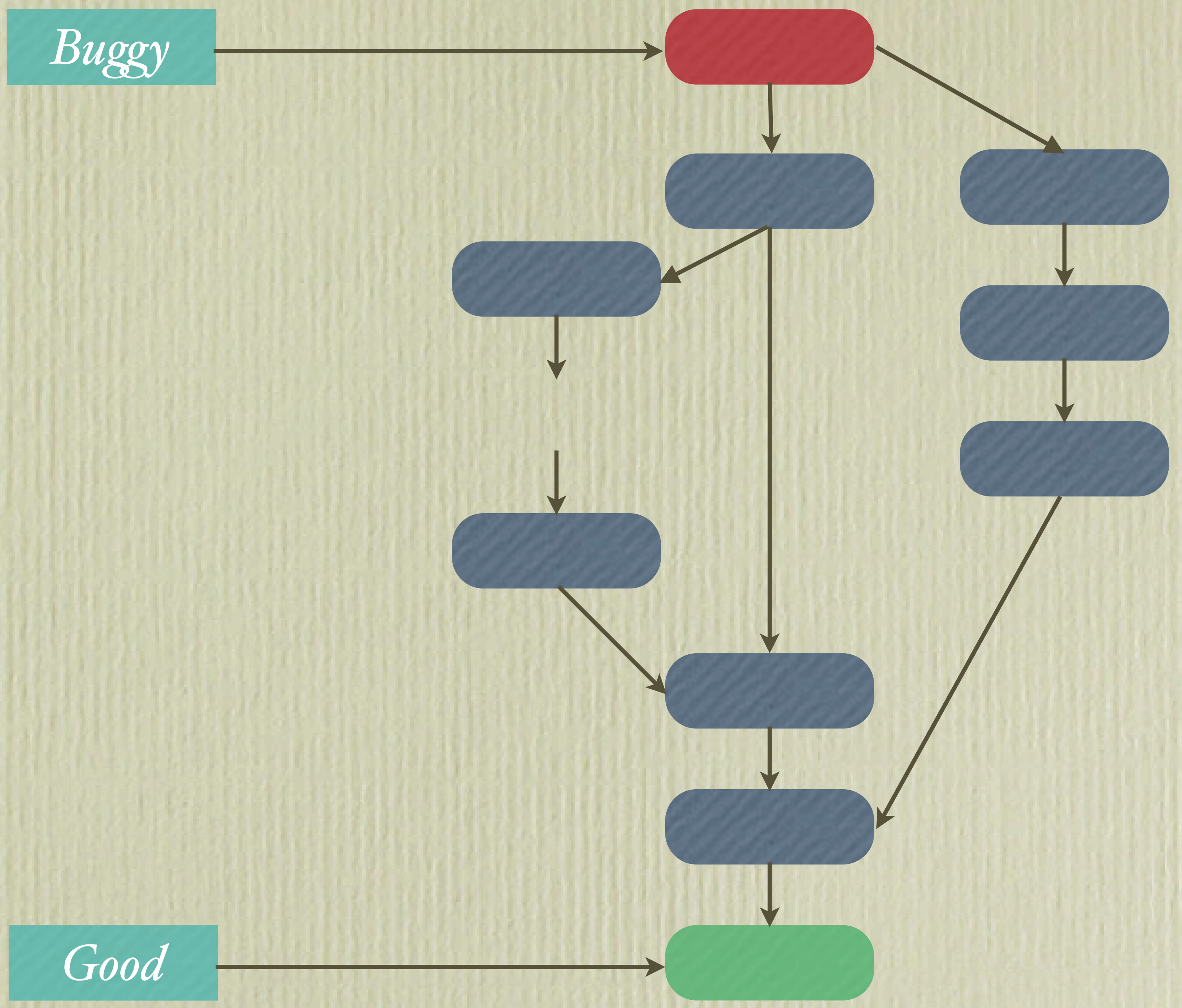
Git runs a binary search algorithm to check out commits and find the first one that causes the test to fail.



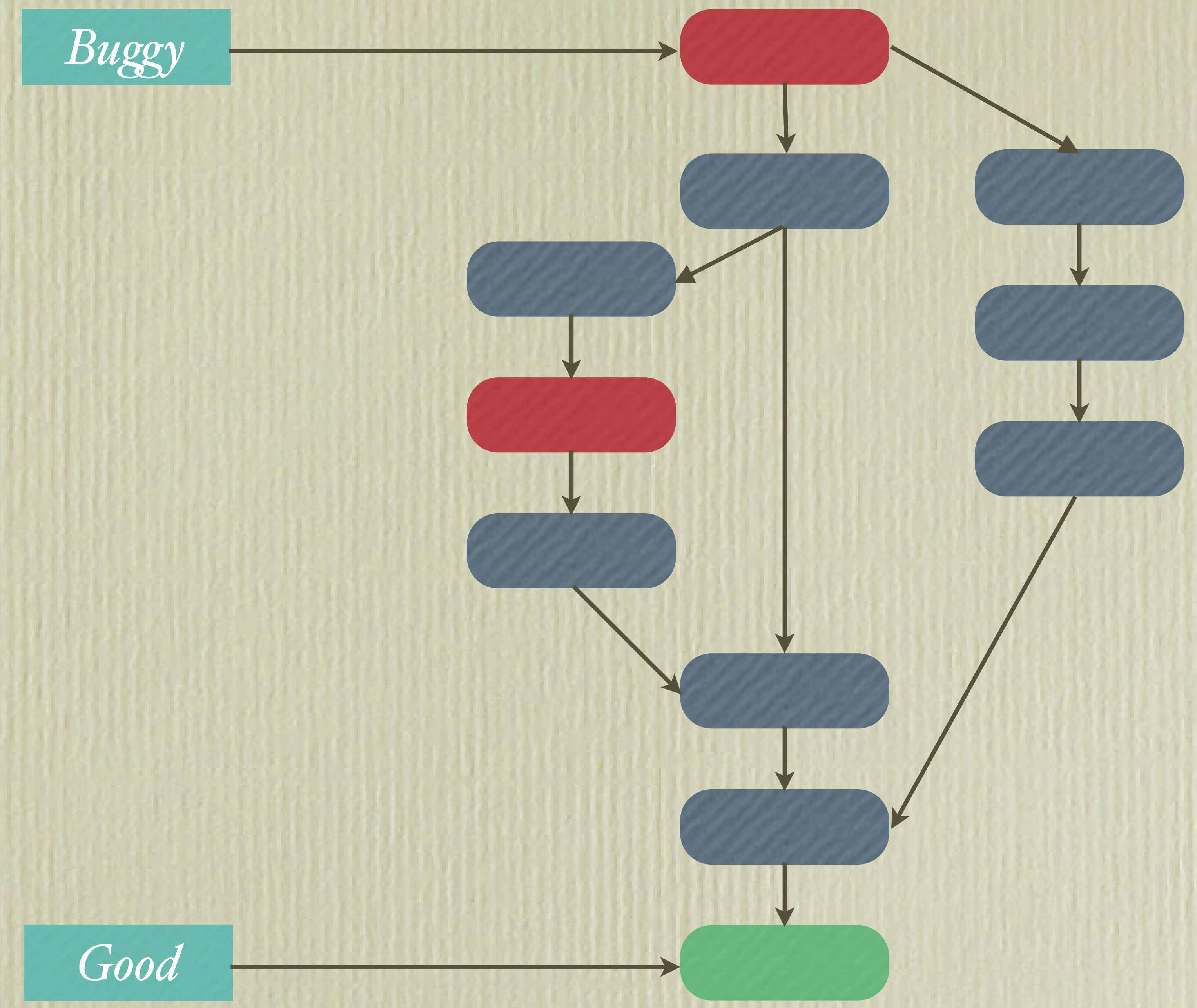
Git runs a binary search algorithm to check out commits and find the first one that causes the test to fail.



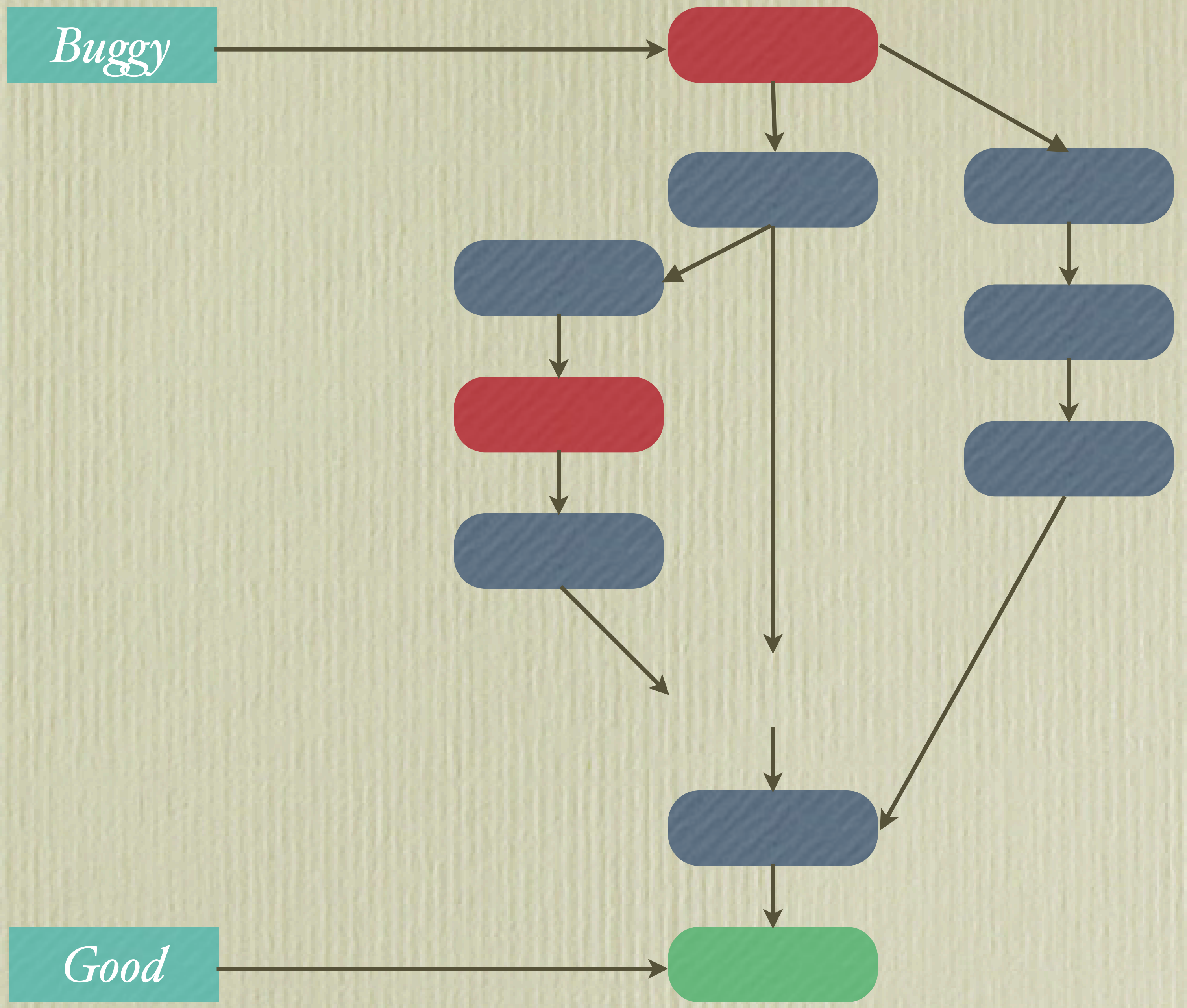
Git runs a binary search algorithm to check out commits and find the first one that causes the test to fail.



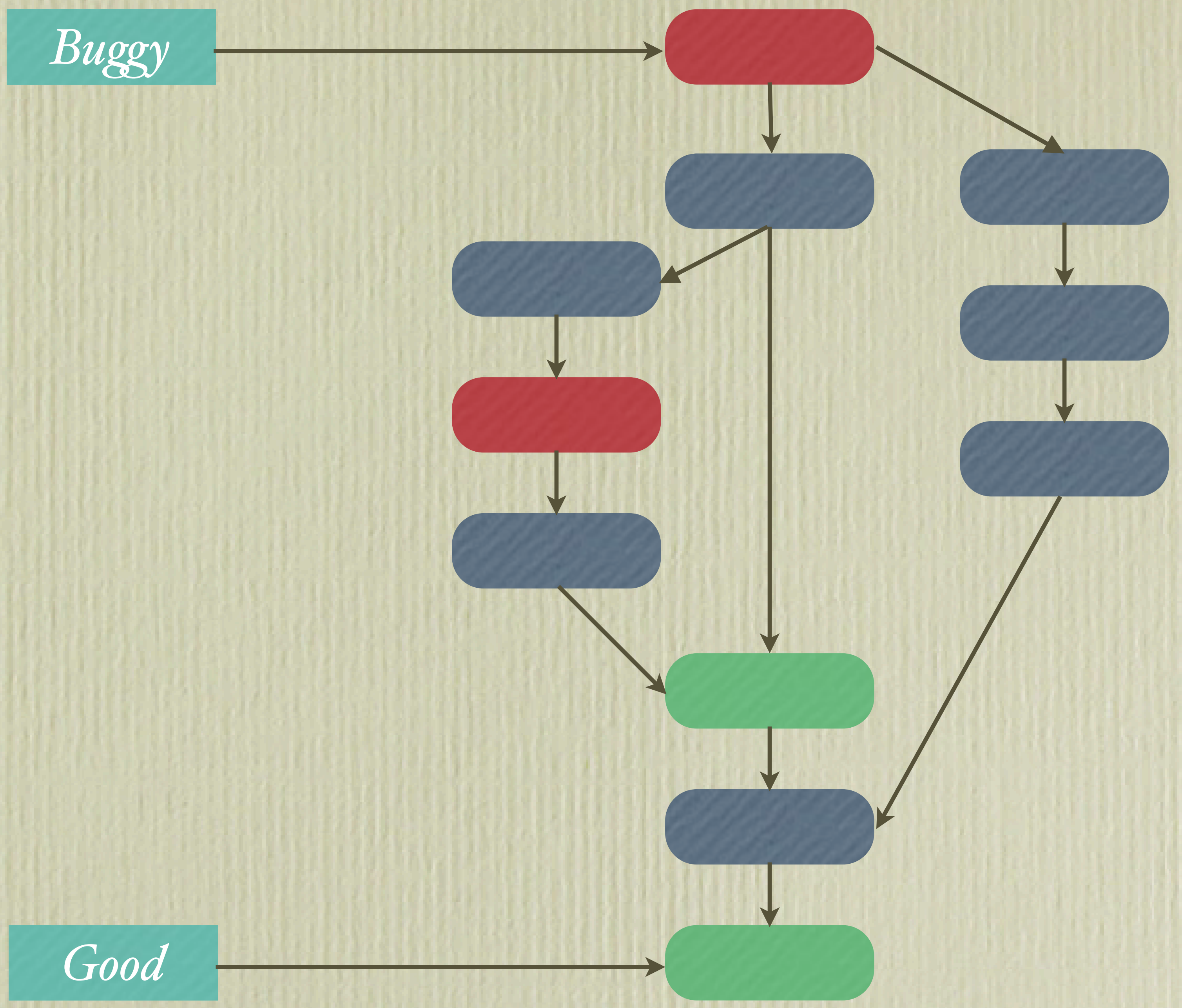
Git runs a binary search algorithm to check out commits and find the first one that causes the test to fail.



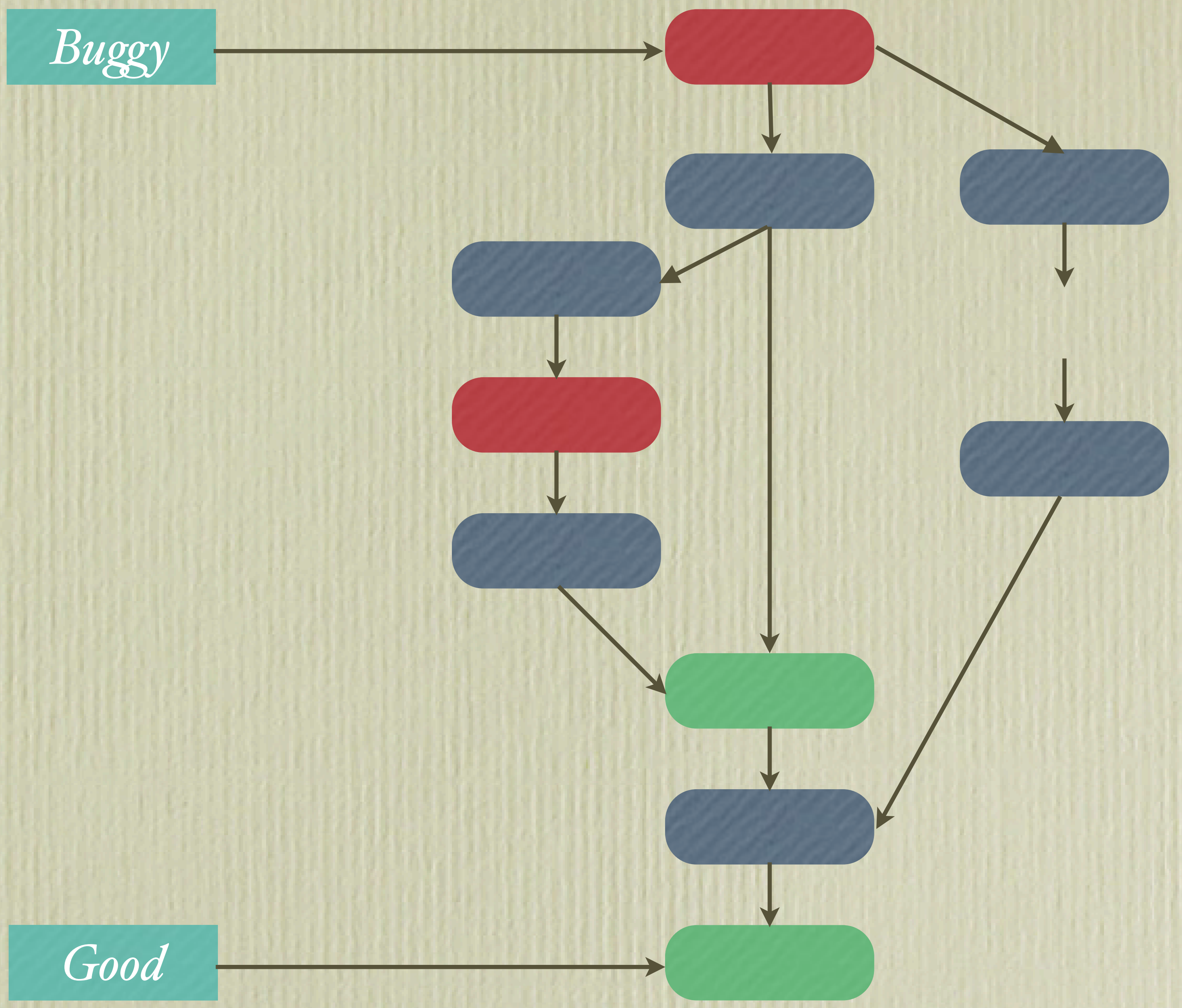
Git runs a binary search algorithm to check out commits and find the first one that causes the test to fail.



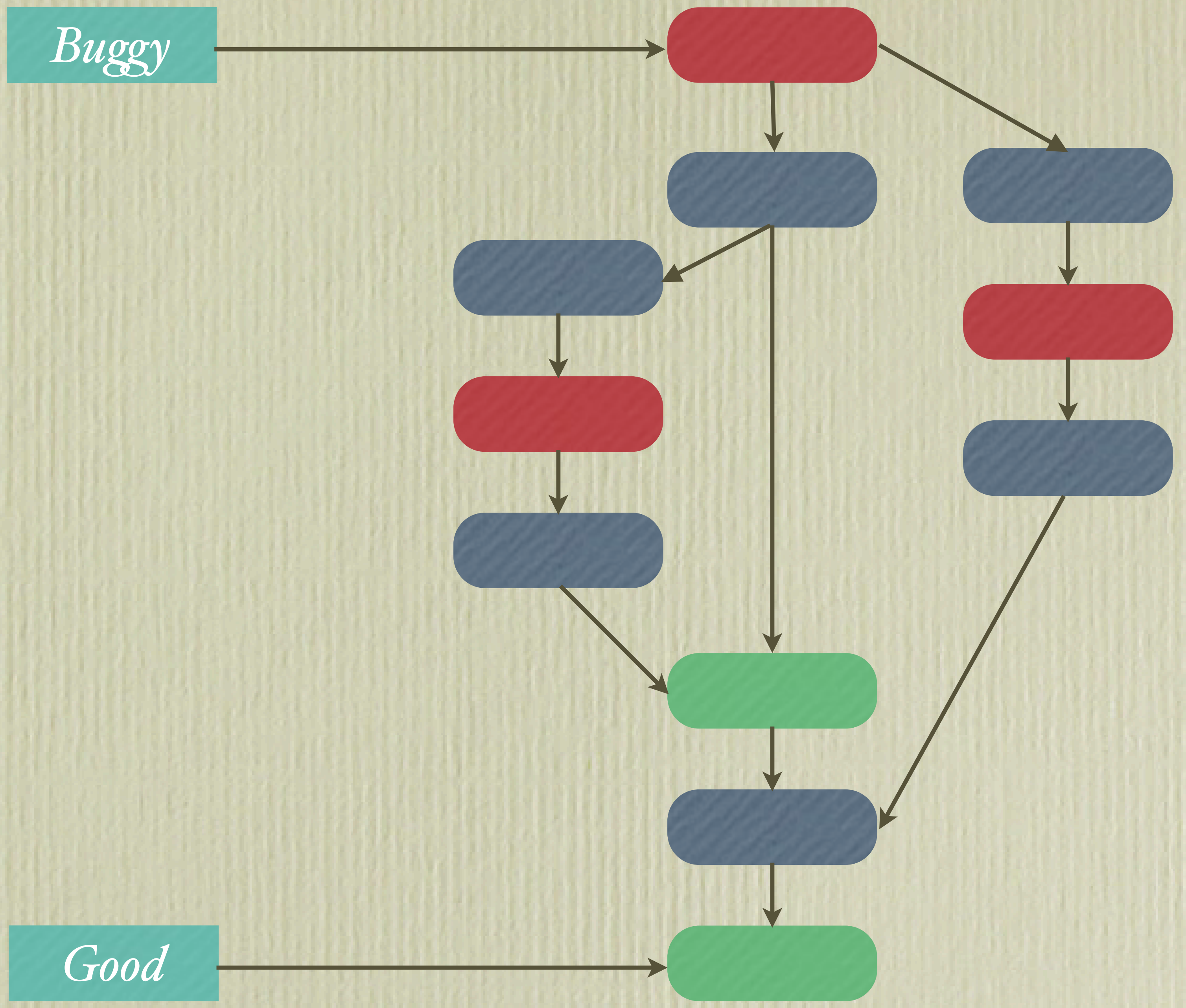
Git runs a binary search algorithm to check out commits and find the first one that causes the test to fail.



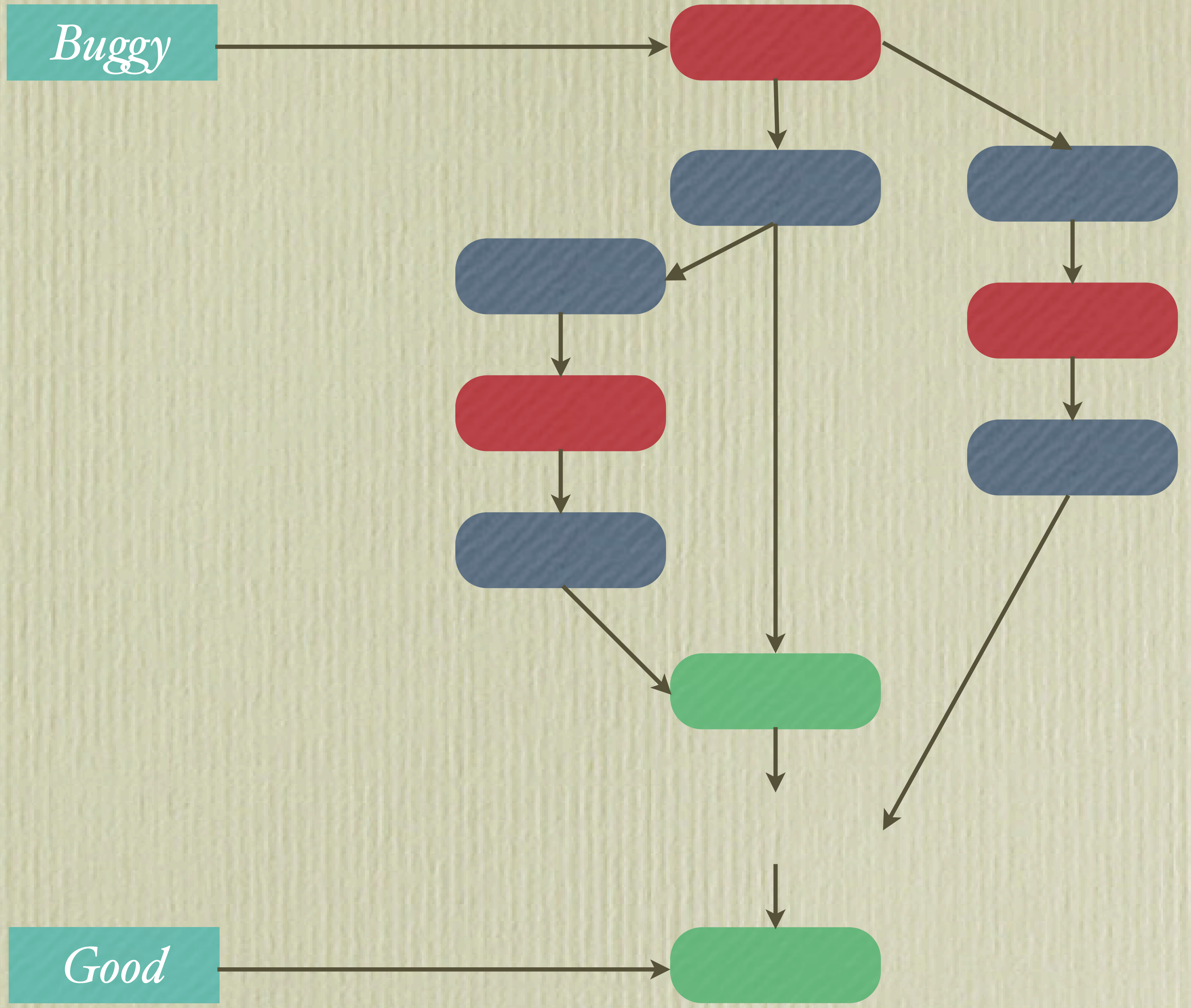
Git runs a binary search algorithm to check out commits and find the first one that causes the test to fail.



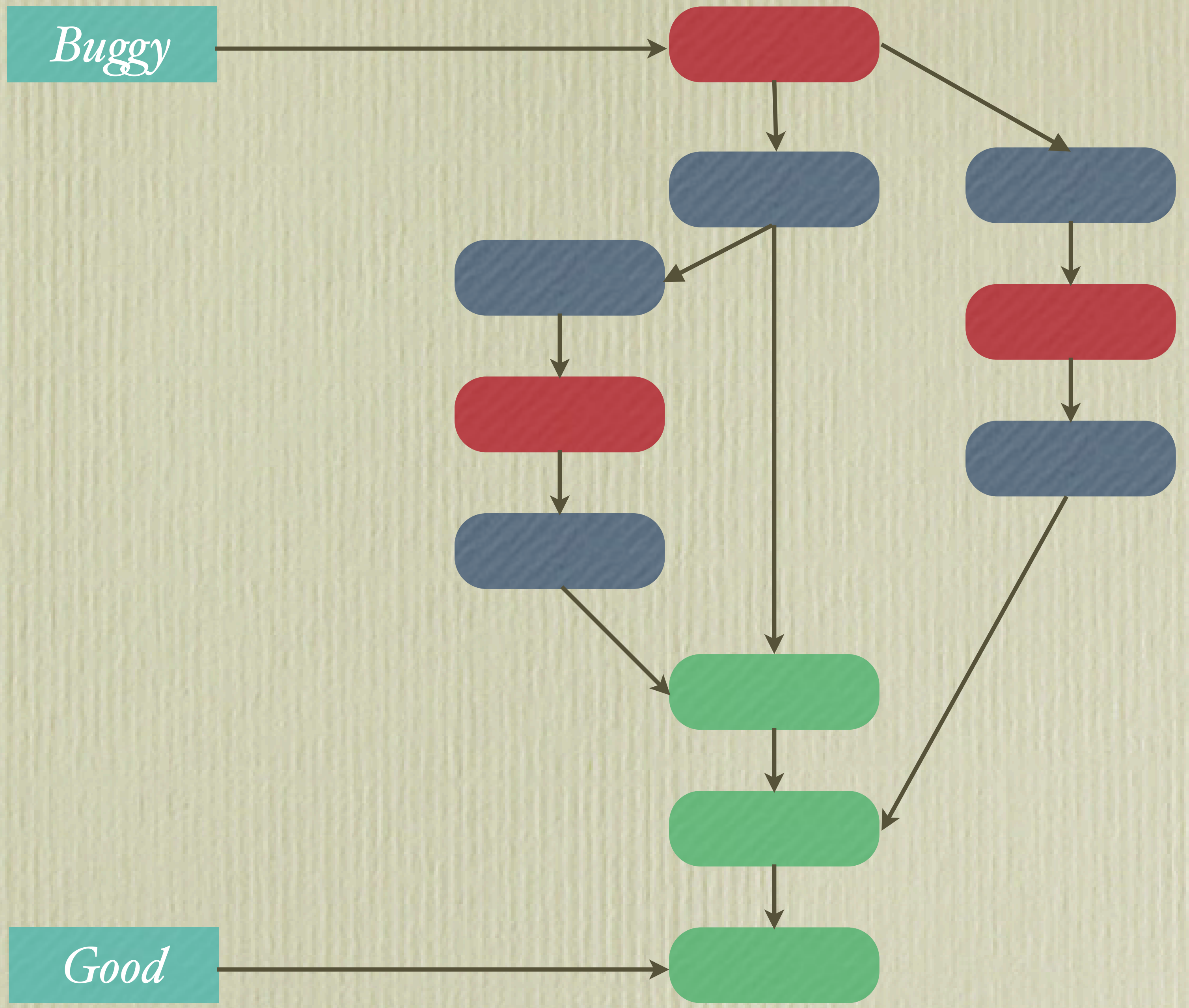
Git runs a binary search algorithm to check out commits and find the first one that causes the test to fail.



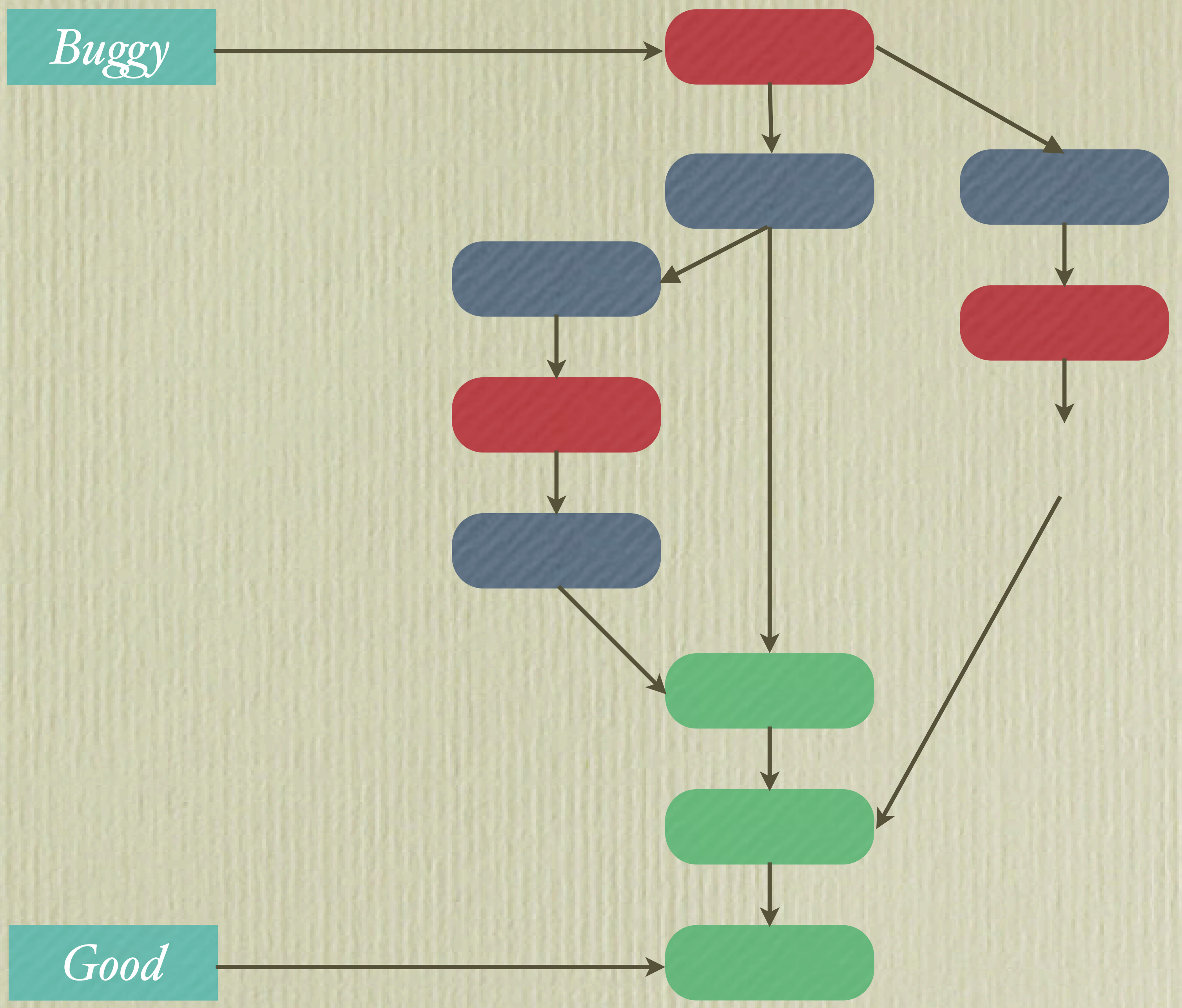
Git runs a binary search algorithm to check out commits and find the first one that causes the test to fail.



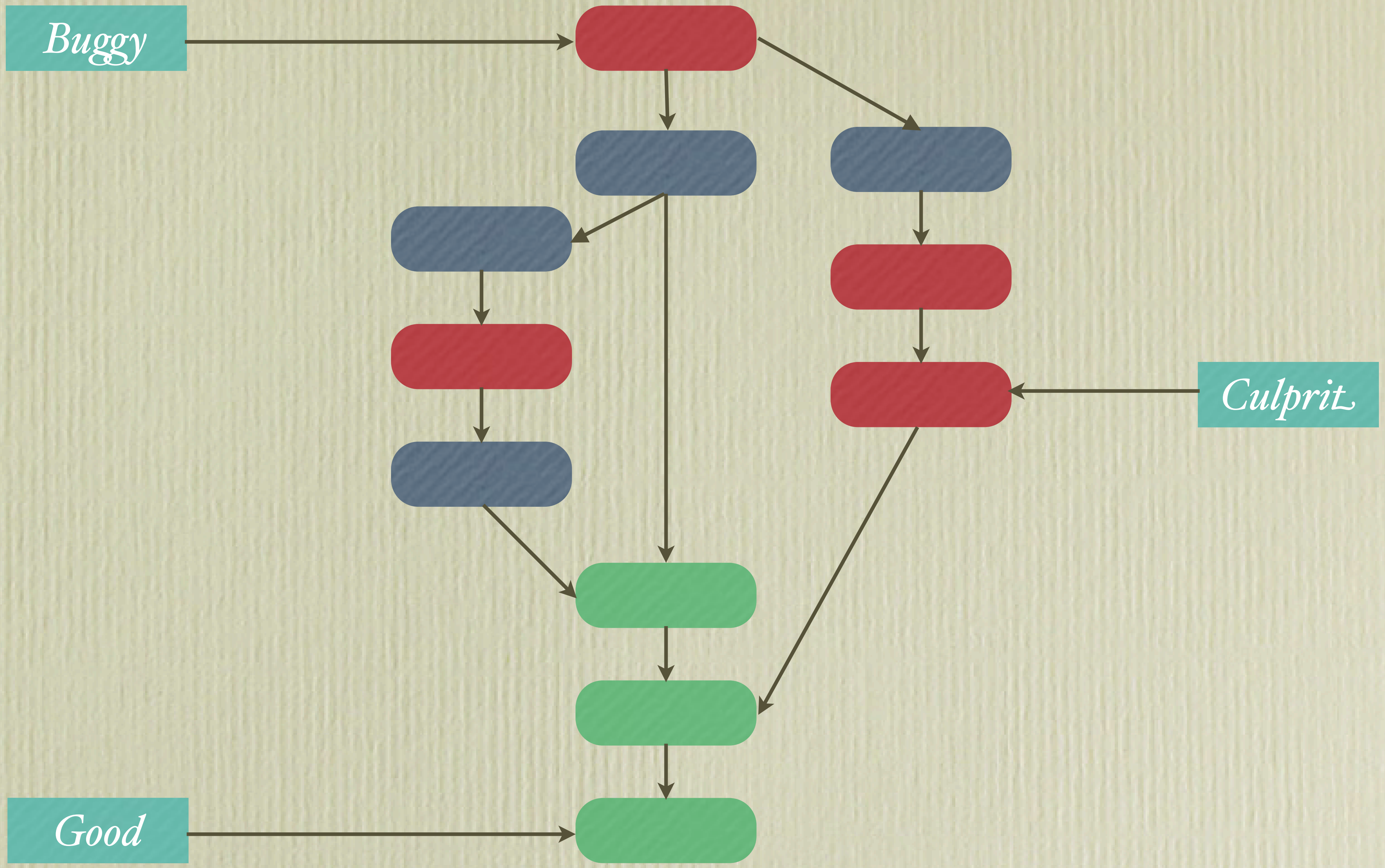
Git runs a binary search algorithm to check out commits and find the first one that causes the test to fail.



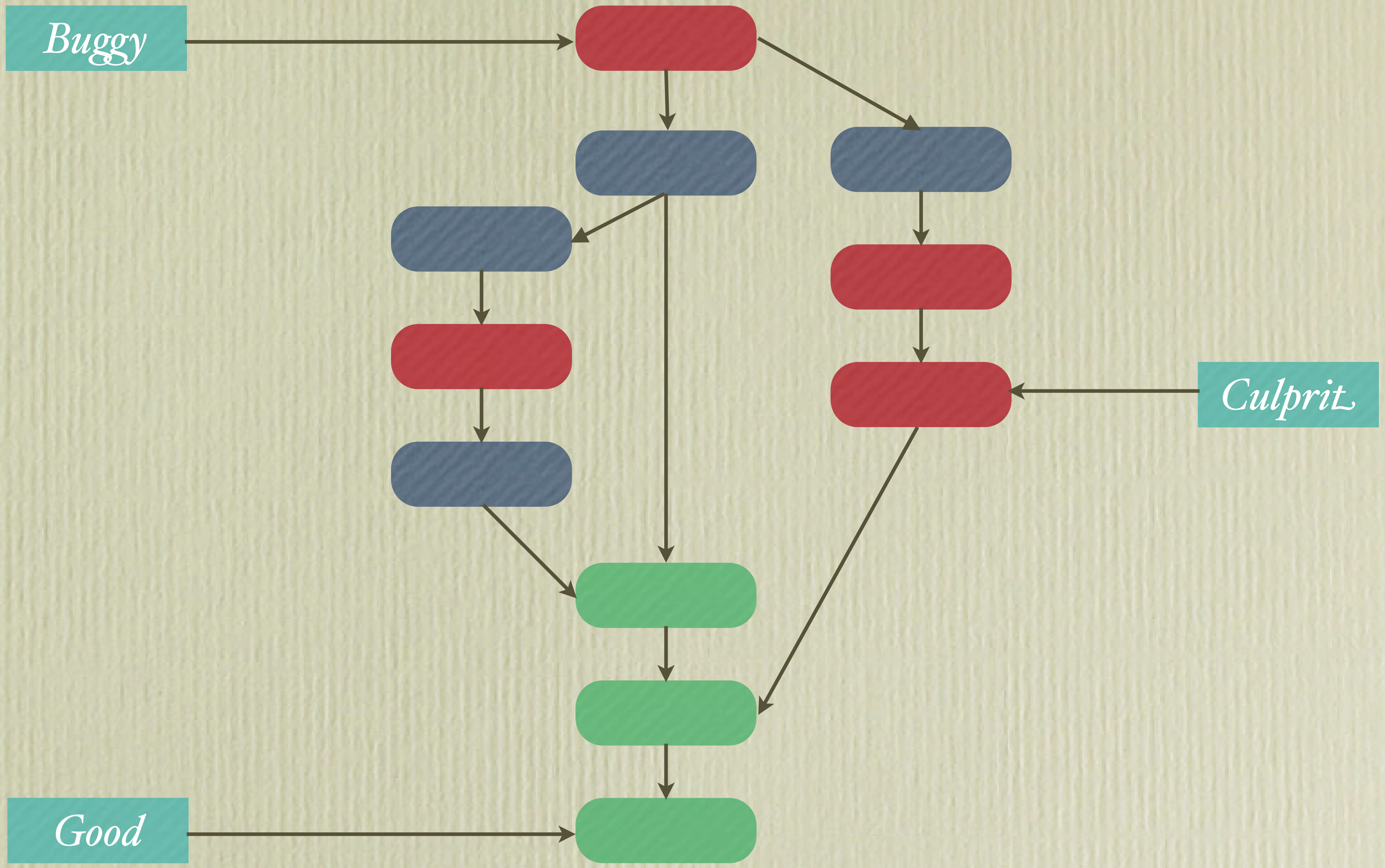
Git runs a binary search algorithm to check out commits and find the first one that causes the test to fail.

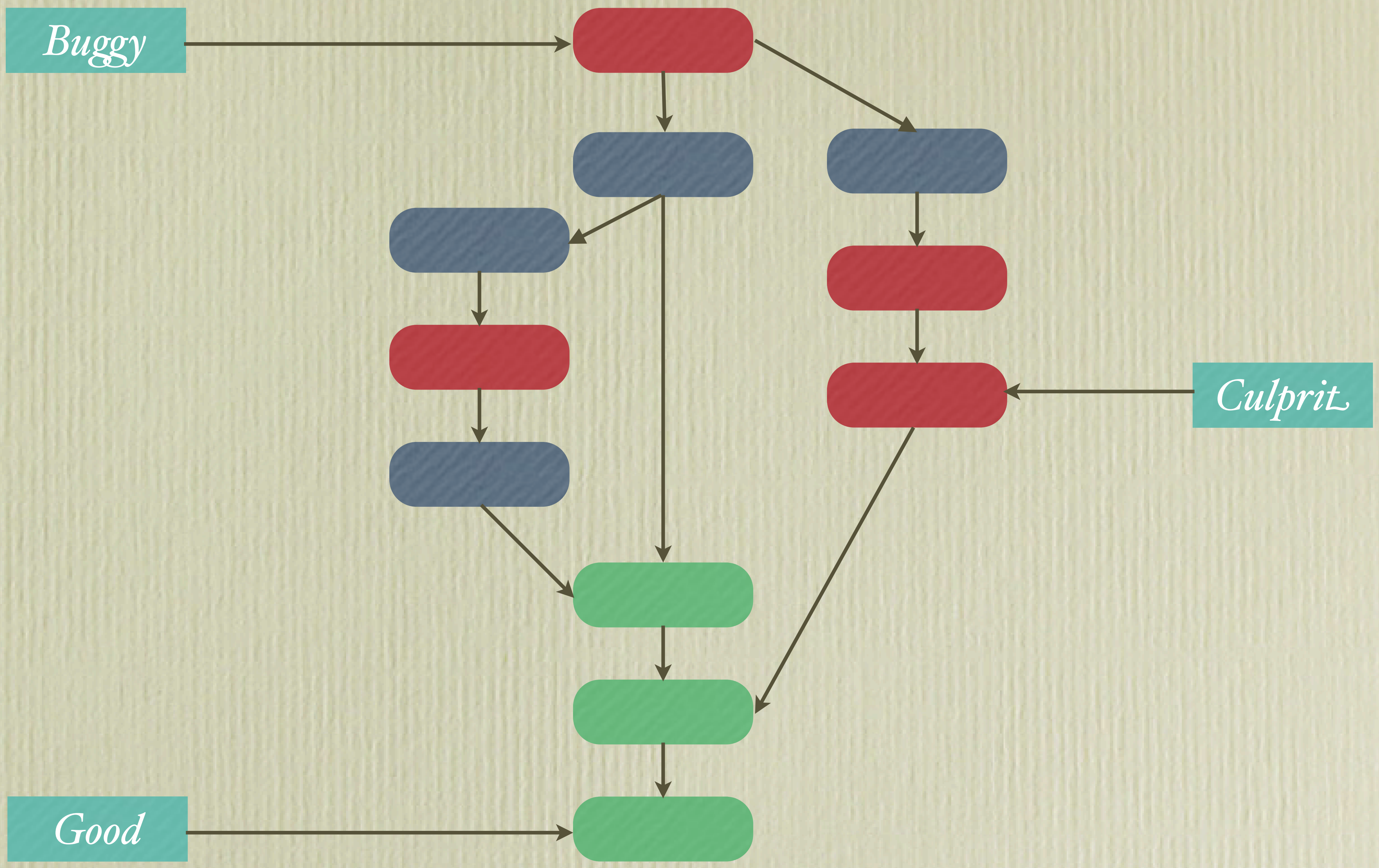


Git runs a binary search algorithm to check out commits and find the first one that causes the test to fail.



Git runs a binary search algorithm to check out commits and find the first one that causes the test to fail.


Buggy
Culprit
Good



Use `git log` to show who checked in the commit, and `git diff` to see the code that introduced the bug.

Tired of typing long commands?

Simple - shell aliases

- For simple one liners, use shell aliases. This is for zsh.
- I don't use too many; just the ones I use commonly.
- Don't need to prefix with \`git\`.
- Keep them simple so that other options may be added.

```
alias gst='git status'
alias gd='git diff'
alias gdt='git difftool'
alias gl='git pull'
alias gp='git push'
alias gc='git commit'
alias gca='git commit -a'
alias gb='git branch'
alias gco='git checkout'
alias gba='git branch -a'
alias gsb='git show-branch'
alias gka='gitk --all &'
alias glo='git log --oneline'
```

Complex - Git Aliases

- Git aliases are in the .gitconfig file.
- Can be local to the repository.
- Prefix with `!` to expand the shortcut in the shell. (useful for multiline commands)
- The last is an example of a local alias for git that will build the latest version and install the man files.

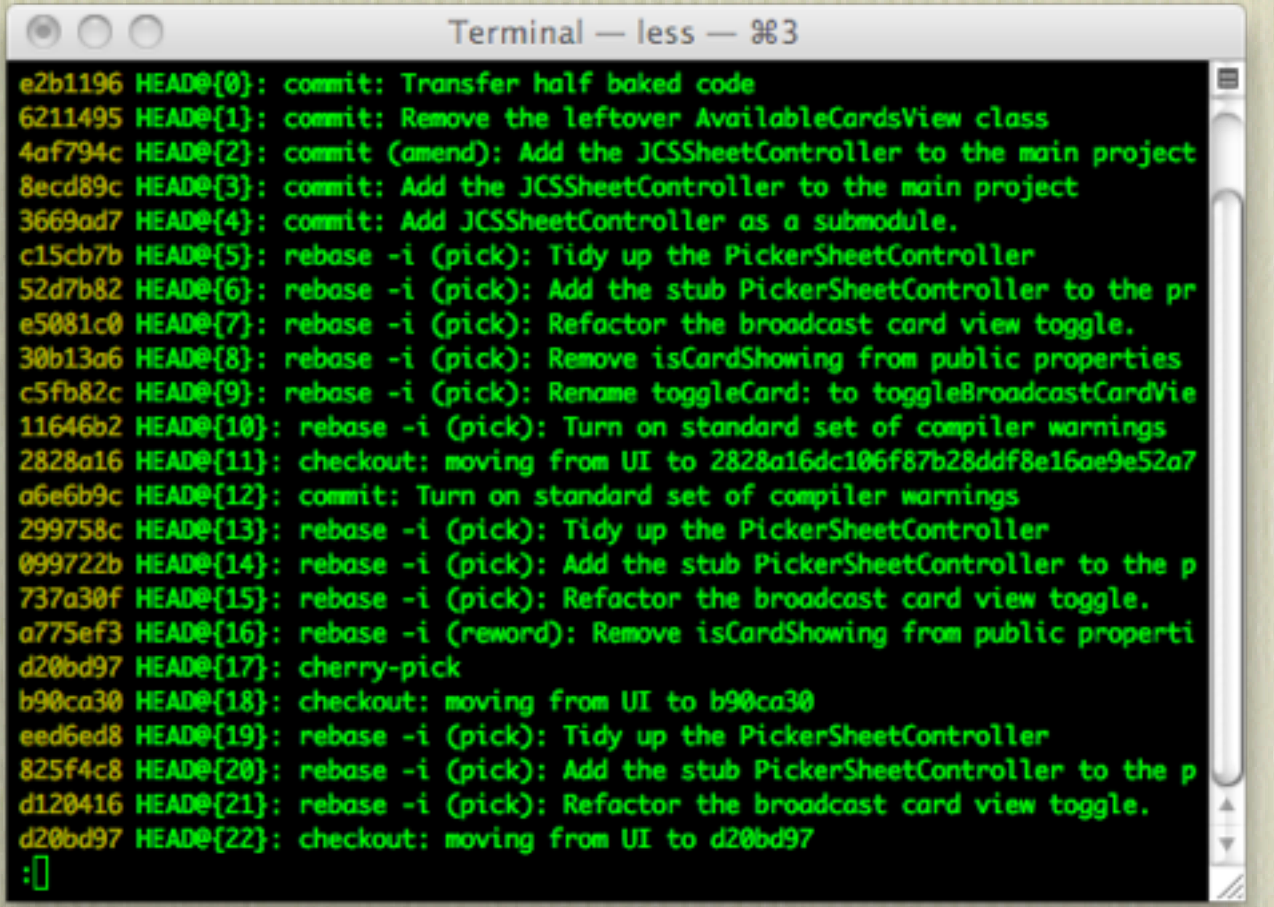
```
[alias]
    st = status
    co = checkout
    dt = difftool
    k = !gitk
```

```
git config alias.install '!
sudo -v && make -j5 all
prefix=/usr/local
NO_DARWIN_PORTS=yes && sudo
make install prefix=/usr/local
NO_DARWIN_PORTS=yes && git
archive origin/man | sudo tar
xvC /usr/local/share/man'
```

What have I done?

git reflog

- When the tip of a branch is updated, this is recorded in the reflog
- This is how you can track commits that are not on any branches, and why it is said that you don't lose data in Git.
- It can be pruned, as with repositories. Although, as with repositories, not everything is tidied up.
- The changes to the local repository are recorded. So history is not leaked.



```
e2b1196 HEAD@{0}: commit: Transfer half baked code
6211495 HEAD@{1}: commit: Remove the leftover AvailableCardsView class
4af794c HEAD@{2}: commit (amend): Add the JCSSheetController to the main project
8ecd89c HEAD@{3}: commit: Add the JCSSheetController to the main project
3669ad7 HEAD@{4}: commit: Add JCSSheetController as a submodule.
c15cb7b HEAD@{5}: rebase -i (pick): Tidy up the PickerSheetController
52d7b82 HEAD@{6}: rebase -i (pick): Add the stub PickerSheetController to the pr
e5081c0 HEAD@{7}: rebase -i (pick): Refactor the broadcast card view toggle.
30b13a6 HEAD@{8}: rebase -i (pick): Remove isCardShowing from public properties
c5fb82c HEAD@{9}: rebase -i (pick): Rename toggleCard: to toggleBroadcastCardVie
11646b2 HEAD@{10}: rebase -i (pick): Turn on standard set of compiler warnings
2828a16 HEAD@{11}: checkout: moving from UI to 2828a16dc106f87b28ddf8e16ae9e52a7
a6e6b9c HEAD@{12}: commit: Turn on standard set of compiler warnings
299758c HEAD@{13}: rebase -i (pick): Tidy up the PickerSheetController
099722b HEAD@{14}: rebase -i (pick): Add the stub PickerSheetController to the p
737a30f HEAD@{15}: rebase -i (pick): Refactor the broadcast card view toggle.
a775ef3 HEAD@{16}: rebase -i (reword): Remove isCardShowing from public properti
d20bd97 HEAD@{17}: cherry-pick
b90ca30 HEAD@{18}: checkout: moving from UI to b90ca30
eed6ed8 HEAD@{19}: rebase -i (pick): Tidy up the PickerSheetController
825f4c8 HEAD@{20}: rebase -i (pick): Add the stub PickerSheetController to the p
d120416 HEAD@{21}: rebase -i (pick): Refactor the broadcast card view toggle.
d20bd97 HEAD@{22}: checkout: moving from UI to d20bd97
:
```

Reading blobs

git show

- This is useful for presenting objects in a human readable format.
    - The contents of a file.
    - The contents of a tree (but not subtrees).
    - The message of a commit and the diff.
- Most useful for commits and tags.

git ls-tree

- Better than git show for viewing trees, because it gives the hashes of the trees and blobs that it points to.
  - -r recursively shows subtrees.
  - -t shows the hashes of the subtrees as well.

git cat-file

- Extracts the contents of individual blobs.
  - -t shows the type of the object instead of the contents.
  - -p pretty print the contents based on the type.

History is written by the victors

Changing history is bad.

- If the repository has been cloned and someone else is working on changes, modifying the history means that they will have to do some work to reconcile the differences.
- This is why I recommend `git fetch` and manually merging upstream changes.

Changing history is not bad.

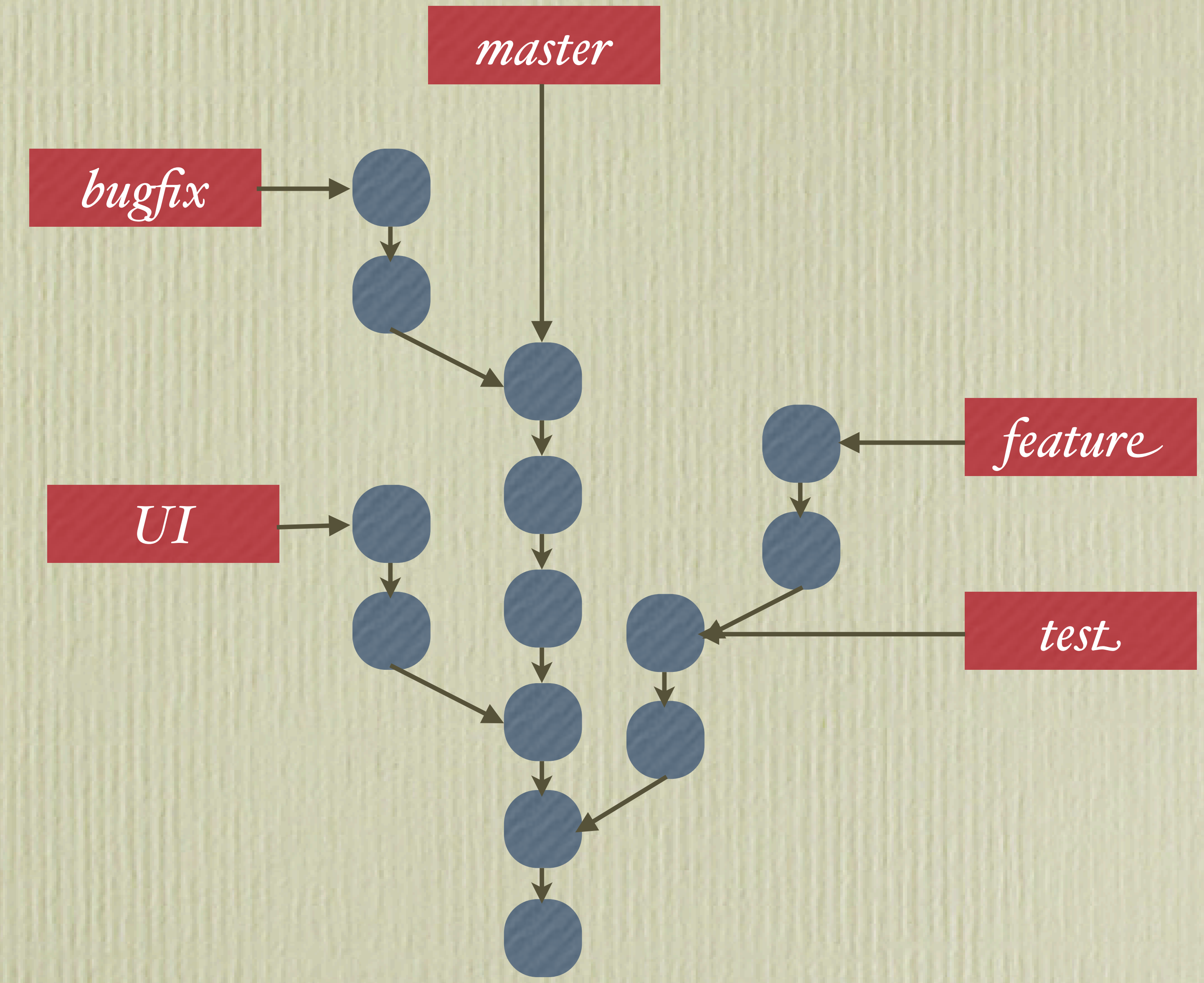
- local development branches – not shared with anyone.
- local branches used for syncing – your codebase, so you know what state it's in.
- A quick `git commit --amend` before anyone is likely to work on the push.
- You can always just re-clone the repository if it messes up.
- `git pull --rebase` is a handy command.

Playing hooky

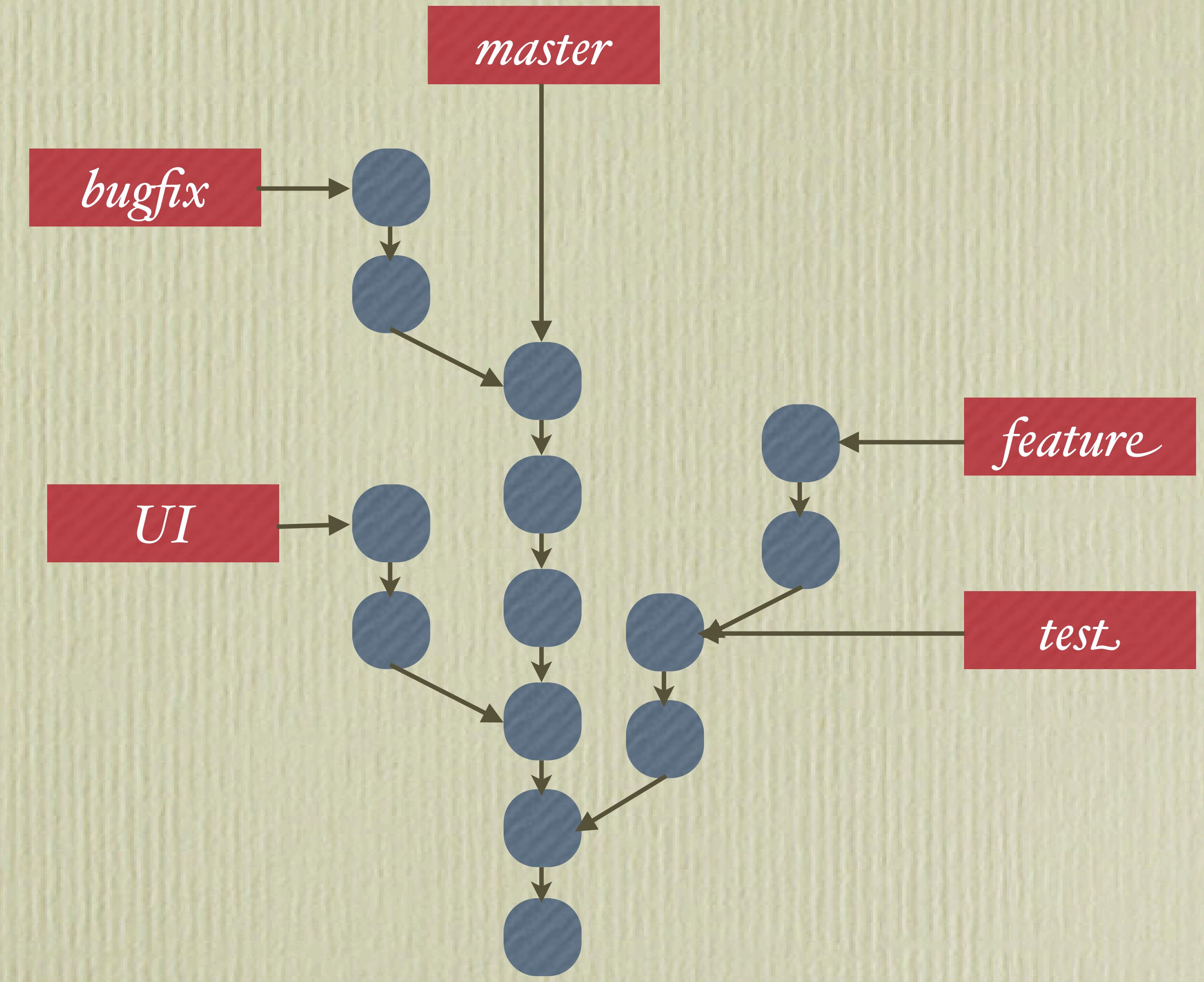
Triggering actions

- Installed in the .git/hooks directory, but disabled by default.
- Samples are shell scripts, but any executable will work.
- local to repository.
    - Run tests.
    - Check code layout.
    - Close tickets when written in the correct format.
    - Twitter, facebook, email.

Too many branches?

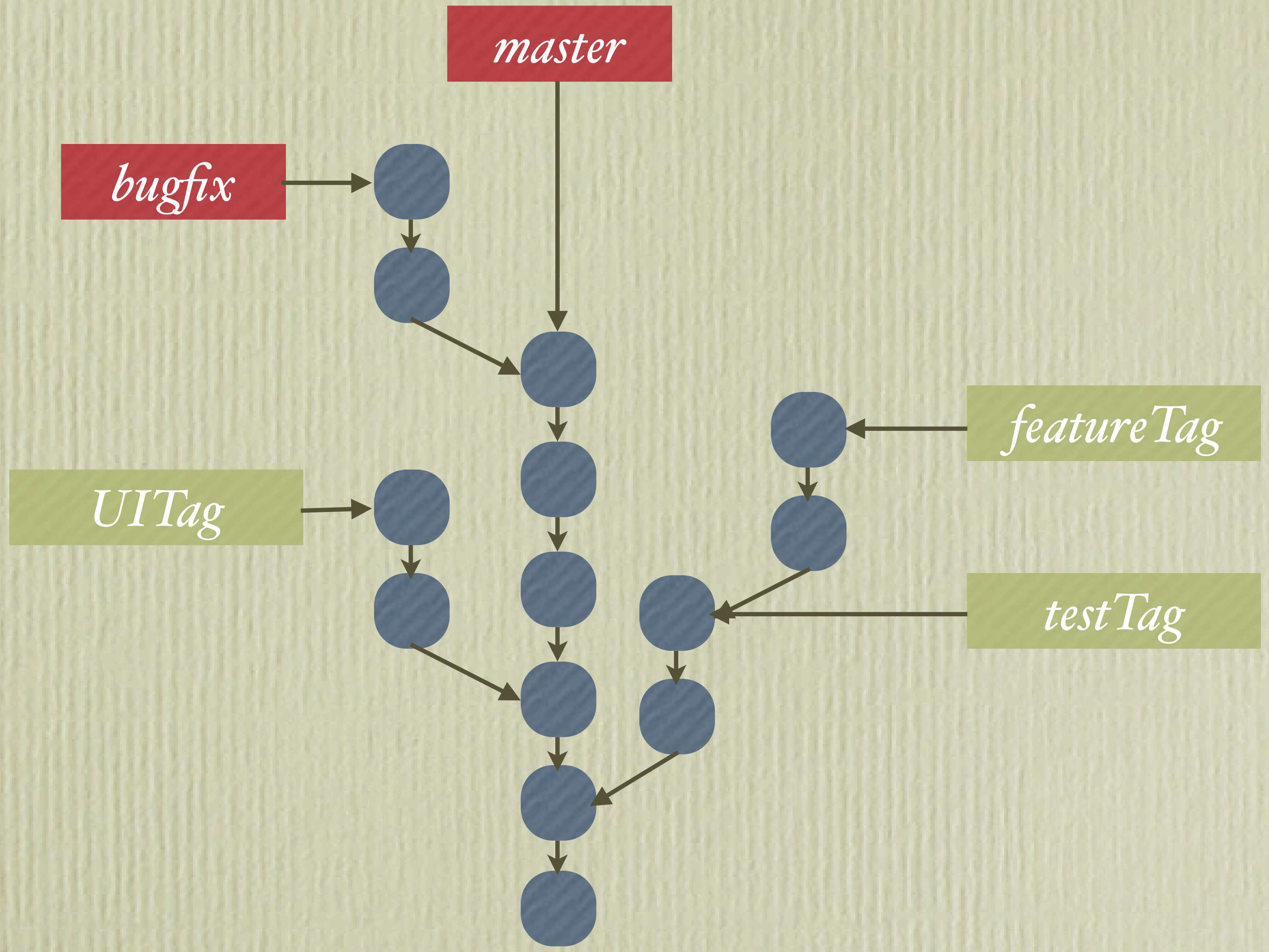


```
$ git branch
$ master
$ bugfix
$ feature
$ test
$ UI
```

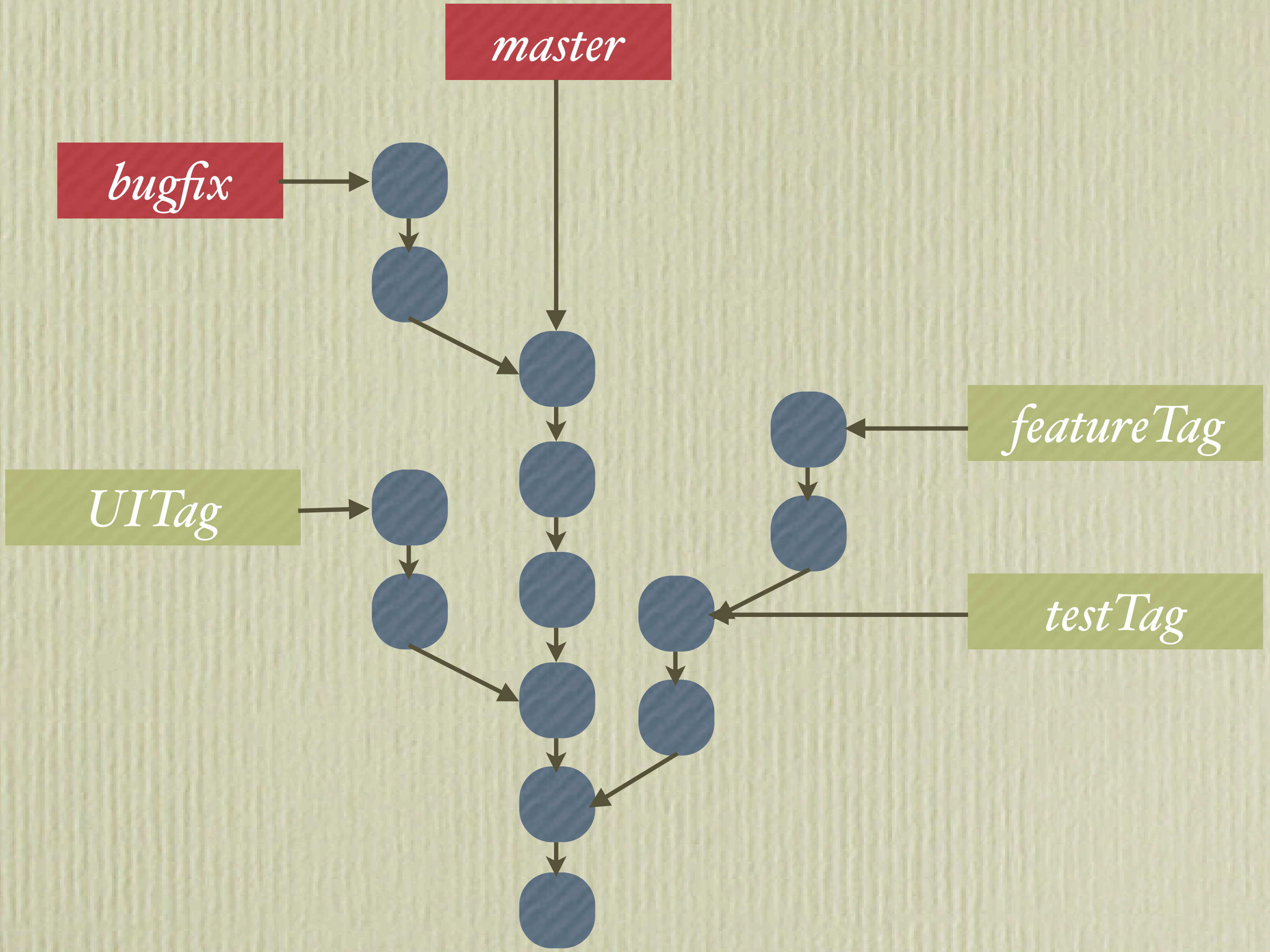


```
$ git branch
$ master
$ bugfix
$ feature
$ test
$ UI
```

Branches are cheap, but it can be distracting to have to deal with too many of them.



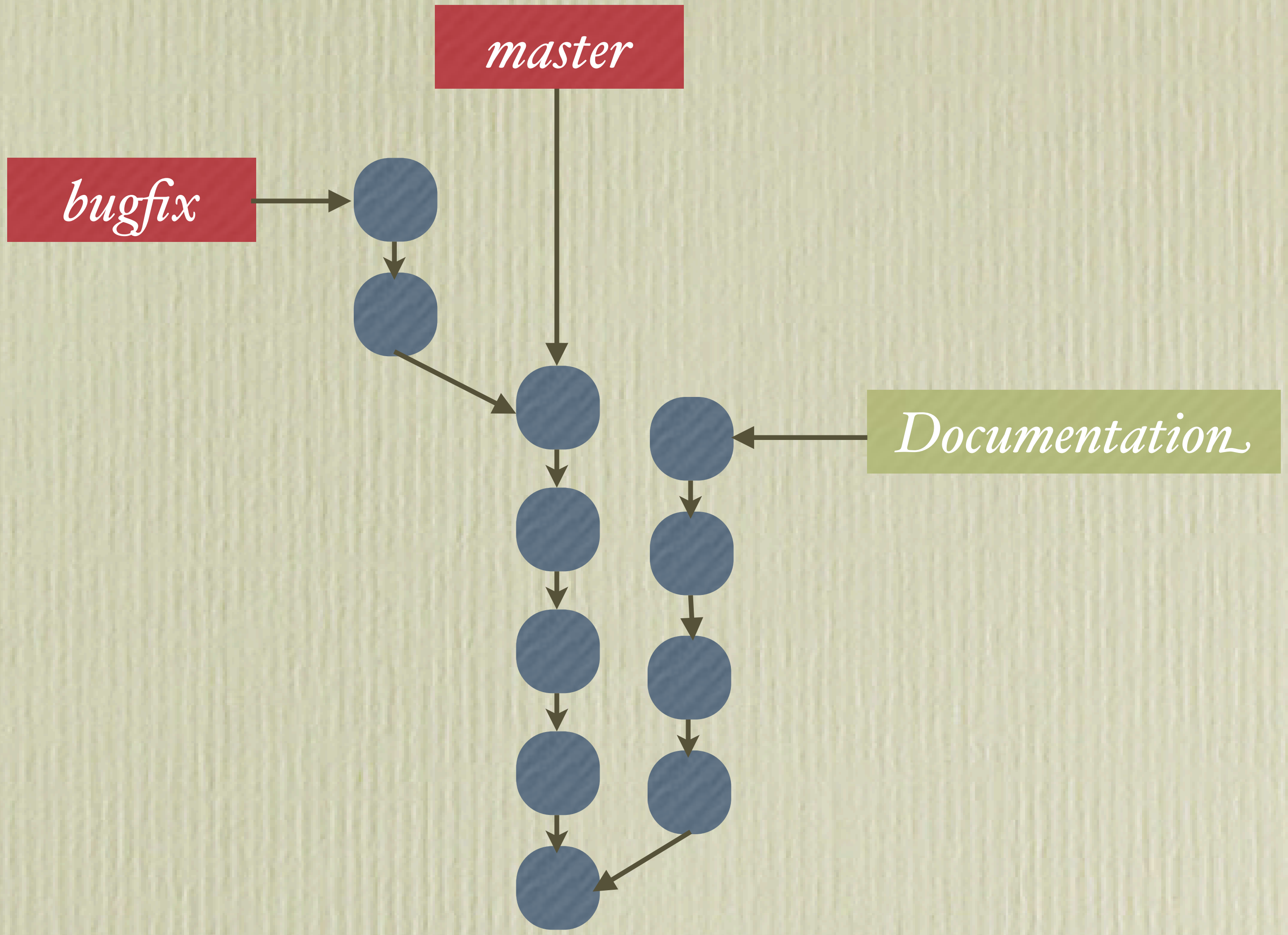
```
$ git branch
$ master
$ bugfix
```

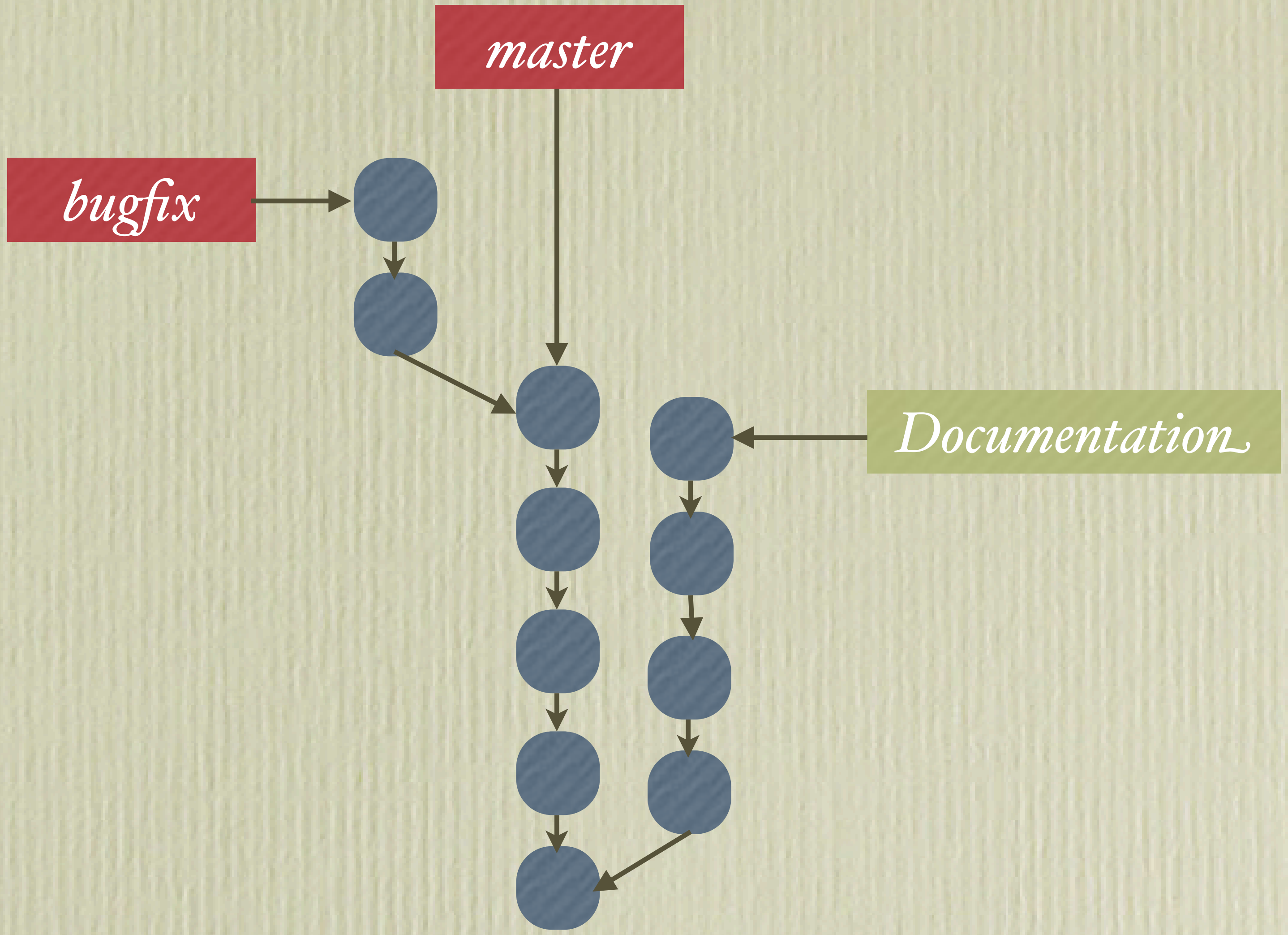


```
$ git branch
$ master
$ bugfix
```

Replace them with tags so they don't show up in the list of branches.
Recreate the branch by branching off the tag.

Independent branches


master
bugfix
Documentation



Branches can be independent of each other. Related items such as documentation, marketing screenshots, App Store copy etc. can be put into their own branches.

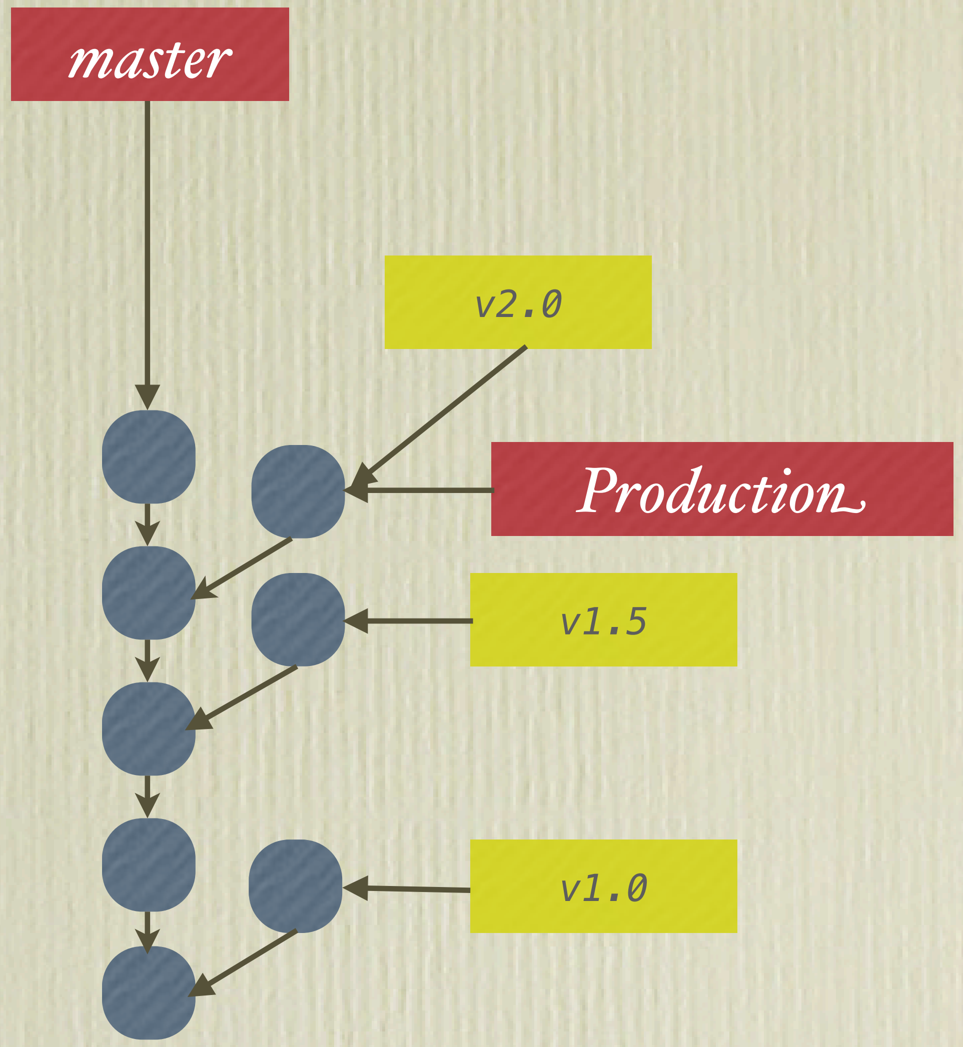
Composition not inheritance

git submodules

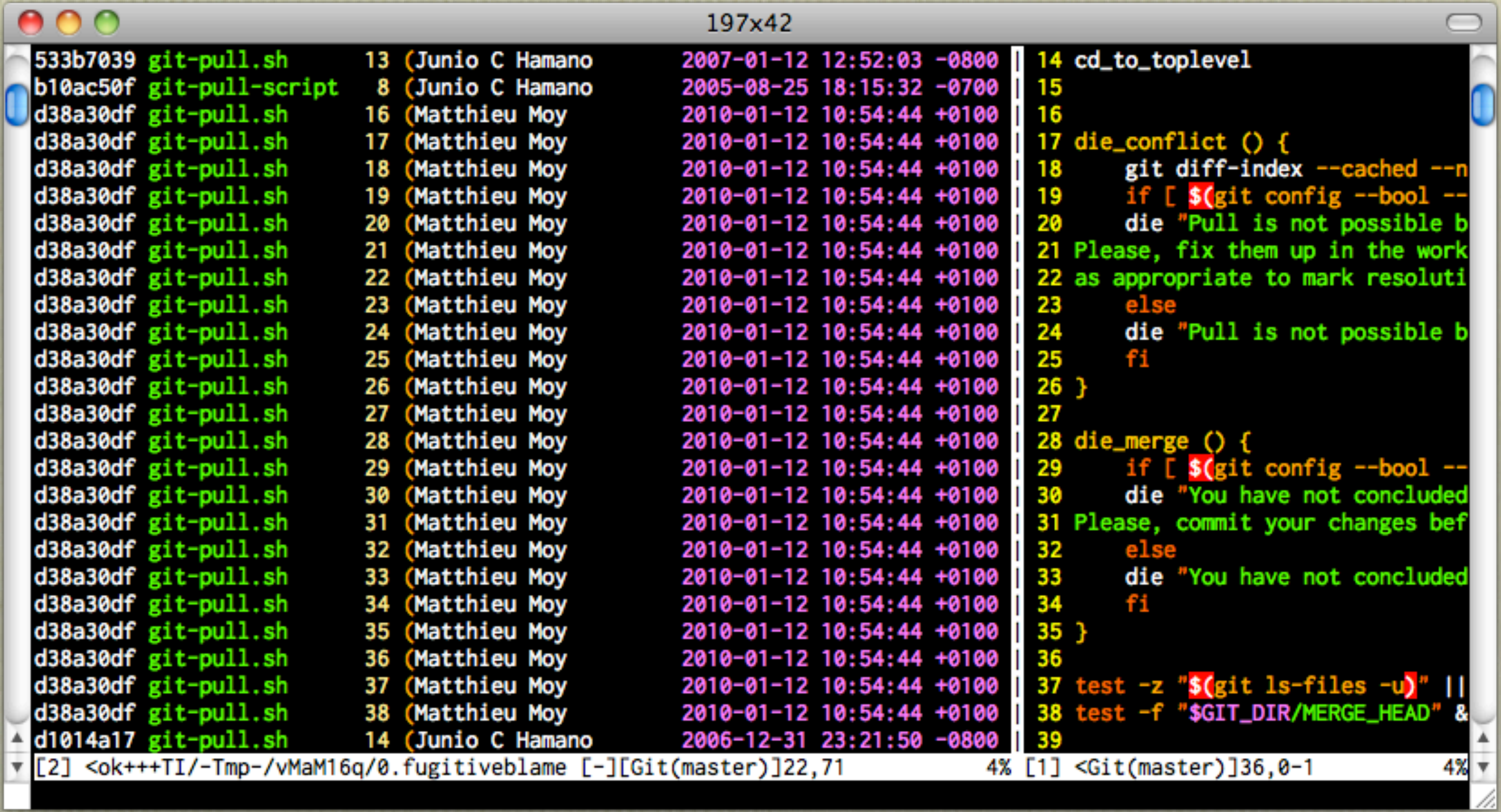
- A git repository within a git repository. Can be recursive
- The super repository checks out the submodule at a specific hash, so upstream changes will not suddenly appear.
- Two stage creation may seem odd, but it means that the submodule that is part of the general repository need not be the same as the local repository.
- Always push submodule changes before pushing super repository changes.

Clean production branches

- It is common to have helper classes that are developed with scaffolding code.
- Create a production branch that has this scaffolding removed.
- Keep it up to date by rebasing.
- Easy to import as submodules.



Dessert



```
533b7039 git-pull.sh      13 (Junio C Hamano      2007-01-12 12:52:03 -0800 | 14 cd_to_toplevel
b10ac50f git-pull-script   8 (Junio C Hamano      2005-08-25 18:15:32 -0700 | 15
d38a30df git-pull.sh      16 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 16
d38a30df git-pull.sh      17 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 17 die_conflict () {
d38a30df git-pull.sh      18 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 18     git diff-index --cached --n
d38a30df git-pull.sh      19 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 19     if [ $(git config --bool --
d38a30df git-pull.sh      20 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 20     die "Pull is not possible b
d38a30df git-pull.sh      21 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 21 Please, fix them up in the work
d38a30df git-pull.sh      22 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 22 as appropriate to mark resoluti
d38a30df git-pull.sh      23 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 23     else
d38a30df git-pull.sh      24 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 24     die "Pull is not possible b
d38a30df git-pull.sh      25 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 25     fi
d38a30df git-pull.sh      26 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 26 }
d38a30df git-pull.sh      27 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 27
d38a30df git-pull.sh      28 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 28 die_merge () {
d38a30df git-pull.sh      29 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 29     if [ $(git config --bool --
d38a30df git-pull.sh      30 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 30     die "You have not concluded
d38a30df git-pull.sh      31 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 31 Please, commit your changes bef
d38a30df git-pull.sh      32 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 32     else
d38a30df git-pull.sh      33 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 33     die "You have not concluded
d38a30df git-pull.sh      34 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 34     fi
d38a30df git-pull.sh      35 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 35 }
d38a30df git-pull.sh      36 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 36
d38a30df git-pull.sh      37 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 37 test -z "$(git ls-files -u)" ||
d38a30df git-pull.sh      38 (Matthieu Moy        2010-01-12 10:54:44 +0100 | 38 test -f "$GIT_DIR/MERGE_HEAD" &
d1014a17 git-pull.sh      14 (Junio C Hamano      2006-12-31 23:21:50 -0800 | 39
[2] <ok+++TI/-Tmp-/vMaM16q/0.fugitiveblame [-][Git(master)]22,71       4% [1] <Git(master)]36,0-1         4%
```

git blame

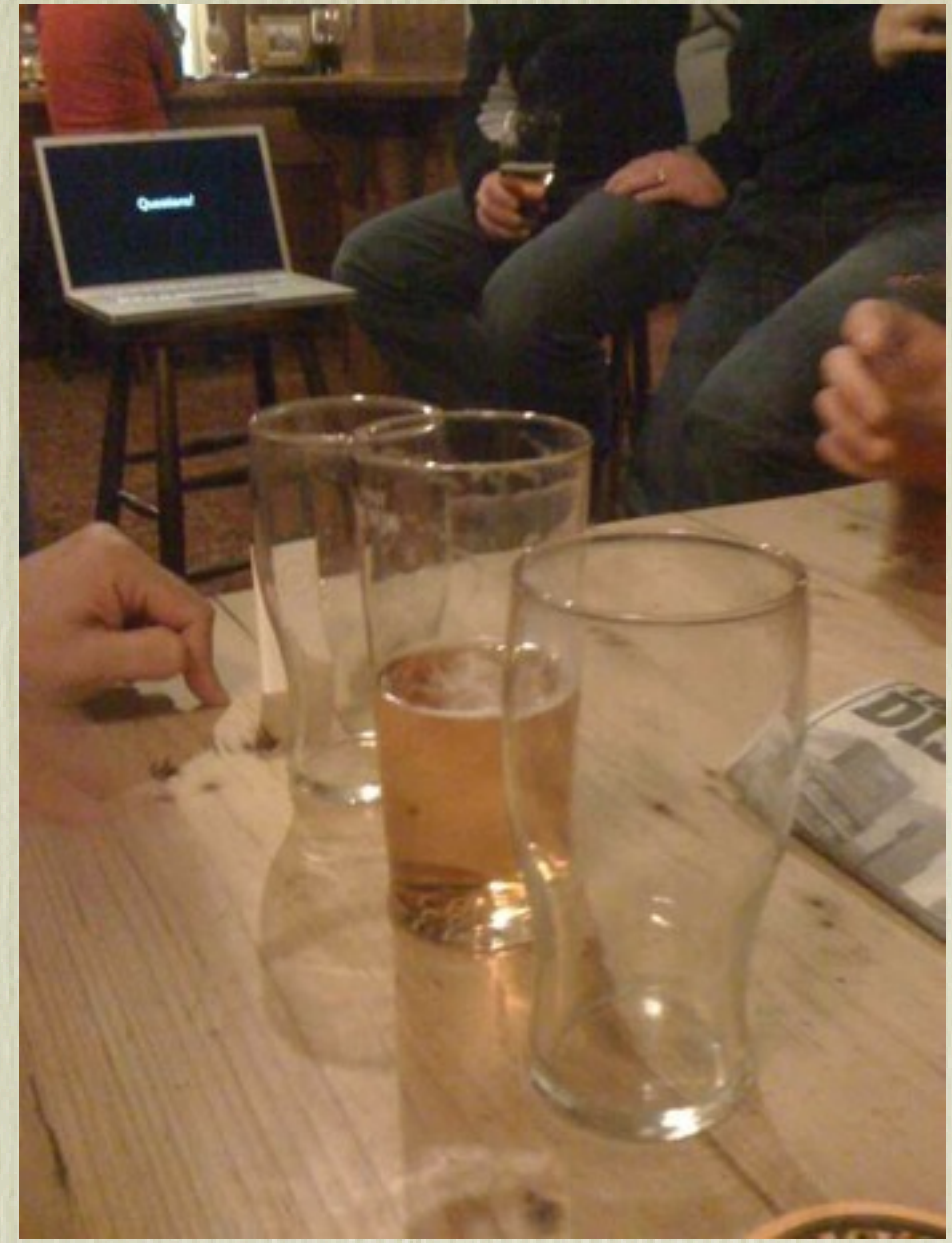
Cheese

Assert your mastery!

- You are capable of handling much bigger abstractions than git.
- You don't take the easy way out with your code; do the same with your version control.
- At the very least, remember the object model – blobs collected into trees collected into commits. Branches, tags, remotes, notes, are all simple constructs built on top of that.

Thank you!

- Coffee / liquers at the Red Lion in Kingly street



Abizer Nasir ◇ @abizern ◇ 365git.tumblr.com ◇ abizern.org