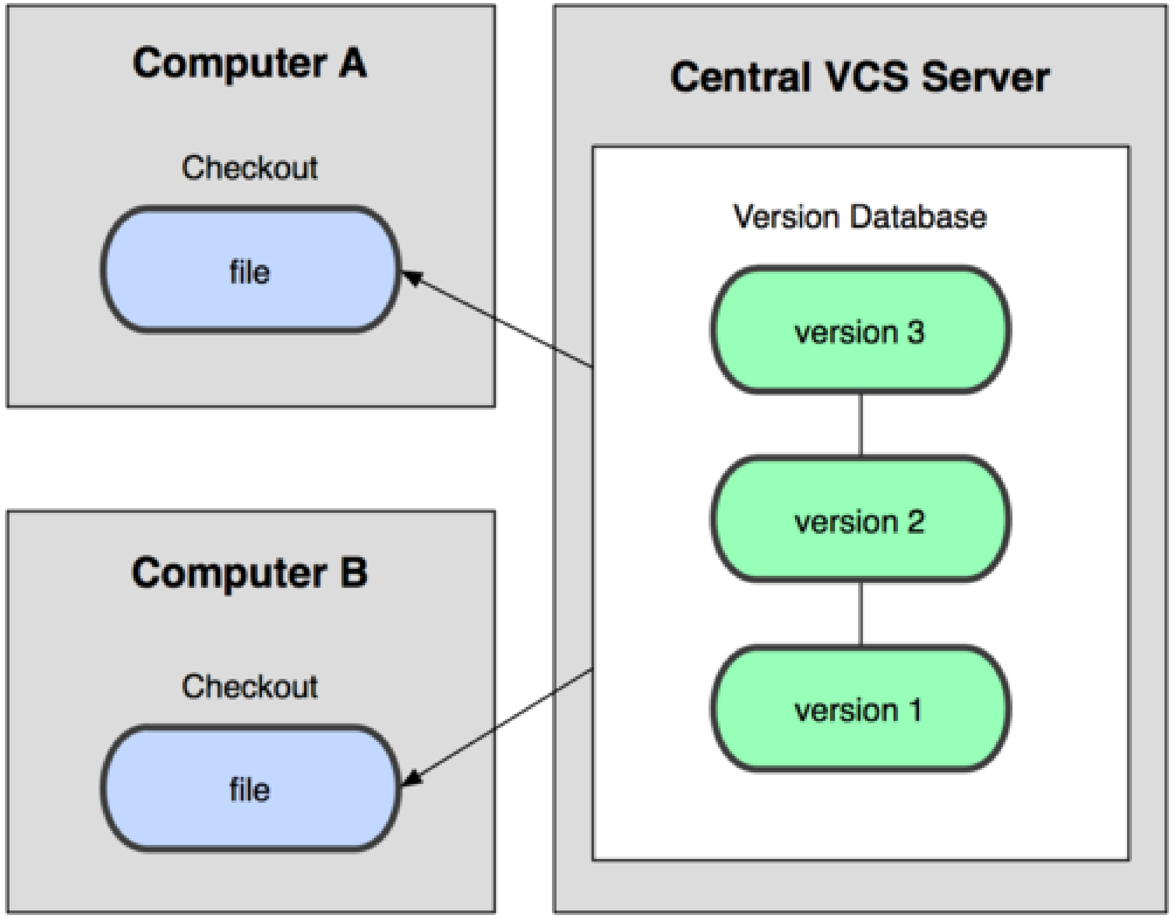
GIT

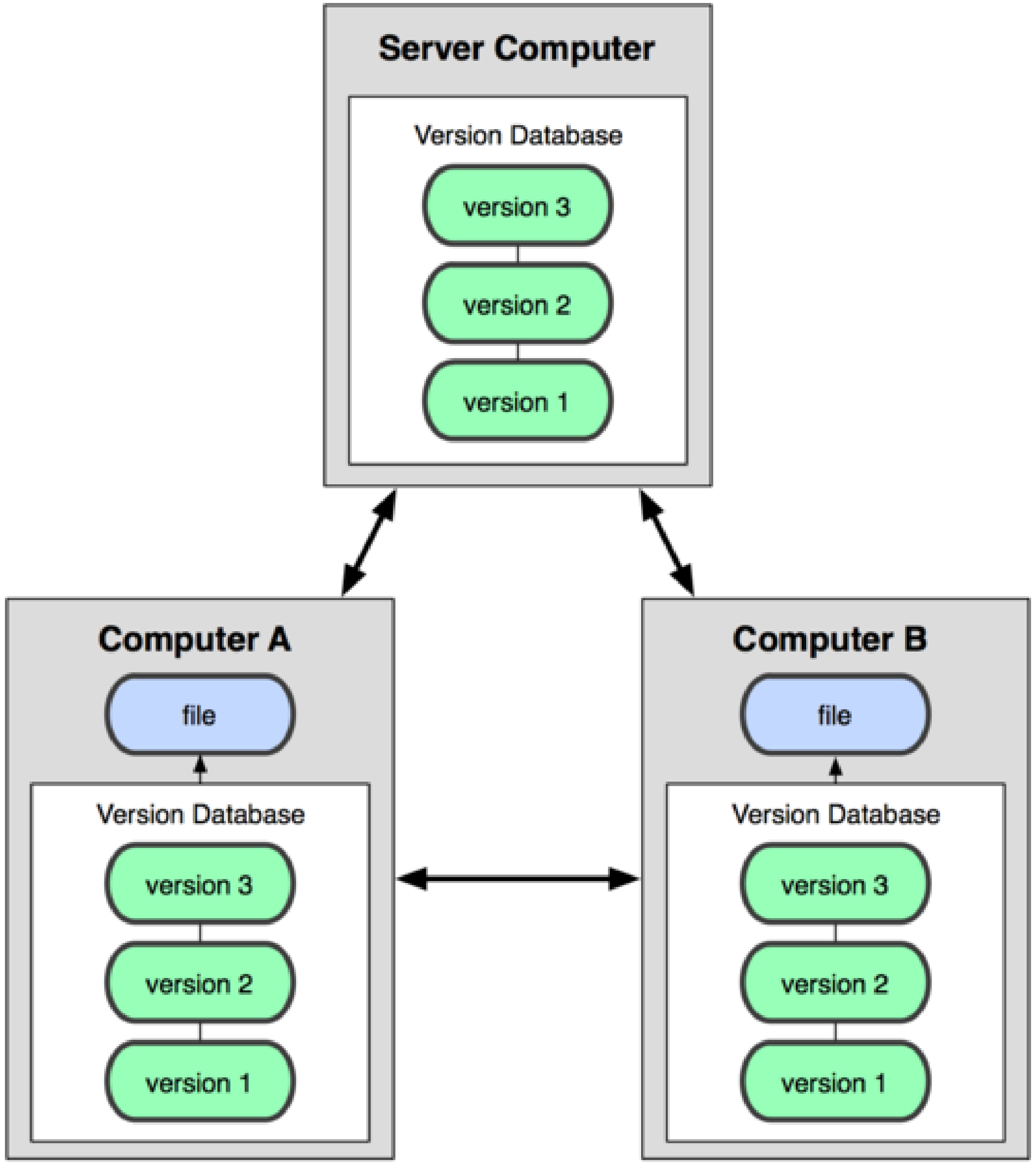
Introduction VCS

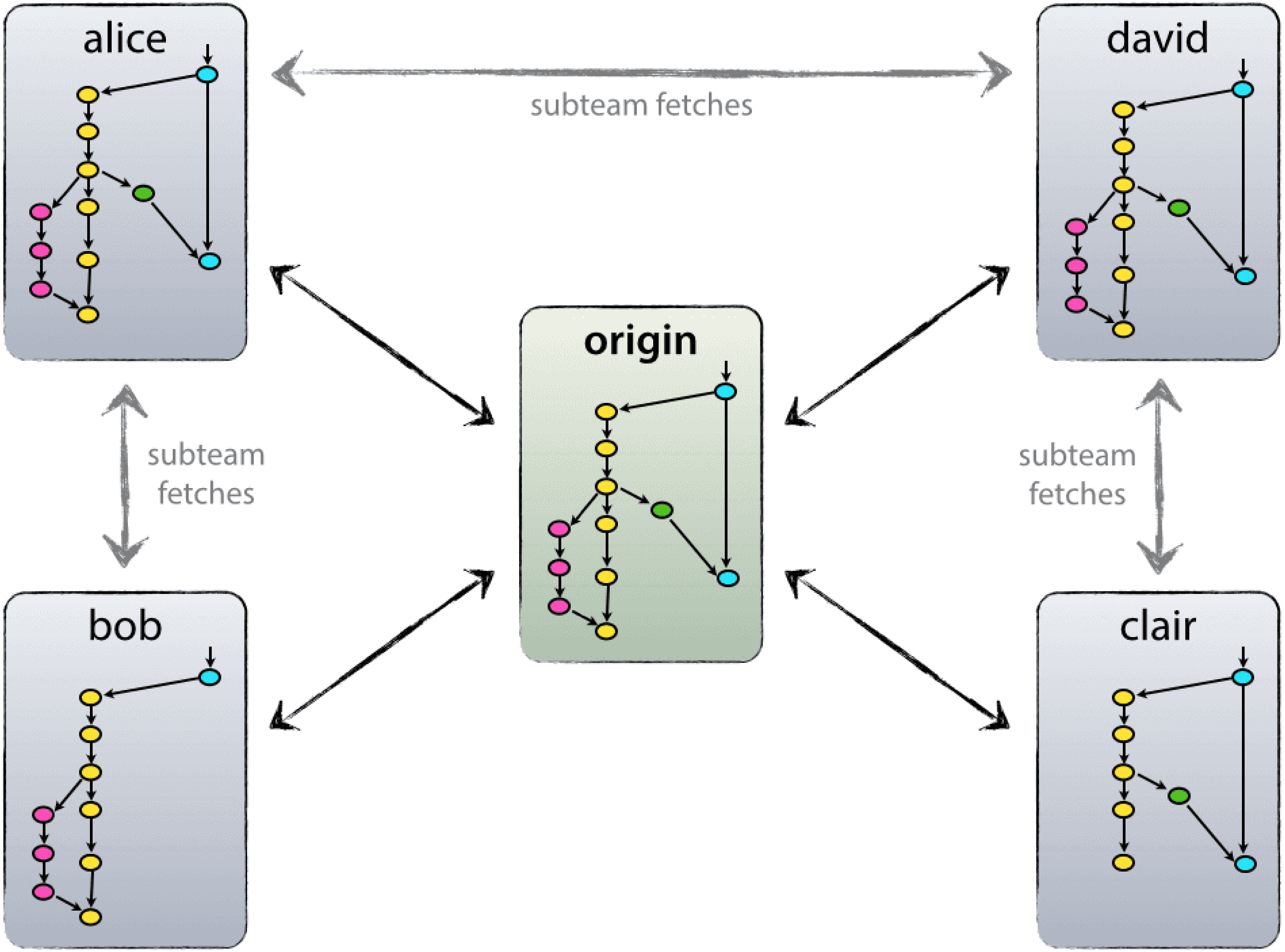
Centralized VCS
–Subversion (SVN)
- checkout
- update
- commit



Introduction VCS

- Distributed VCS
  - GIT





alice

david

subteam fetches

origin

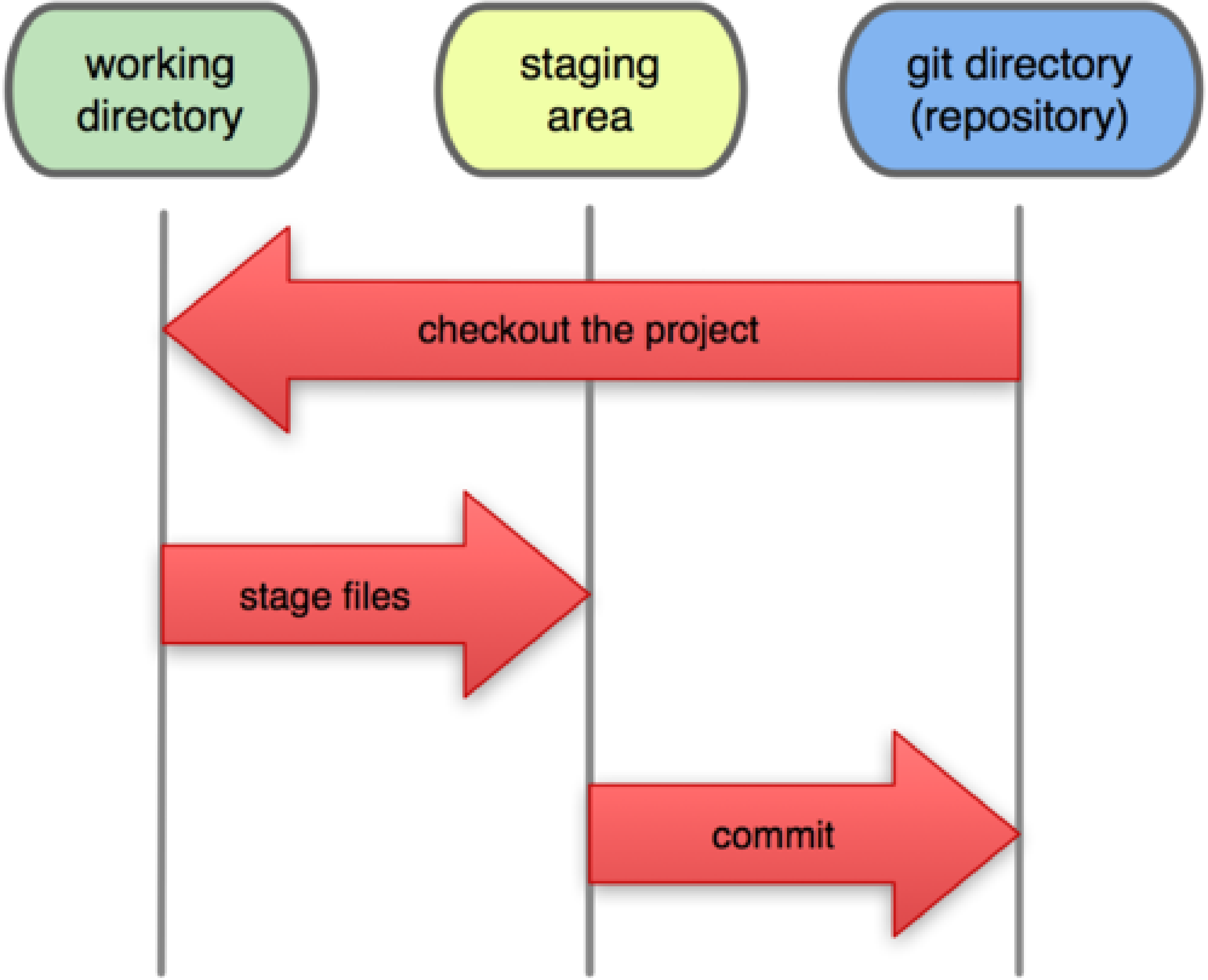
subteam fetches

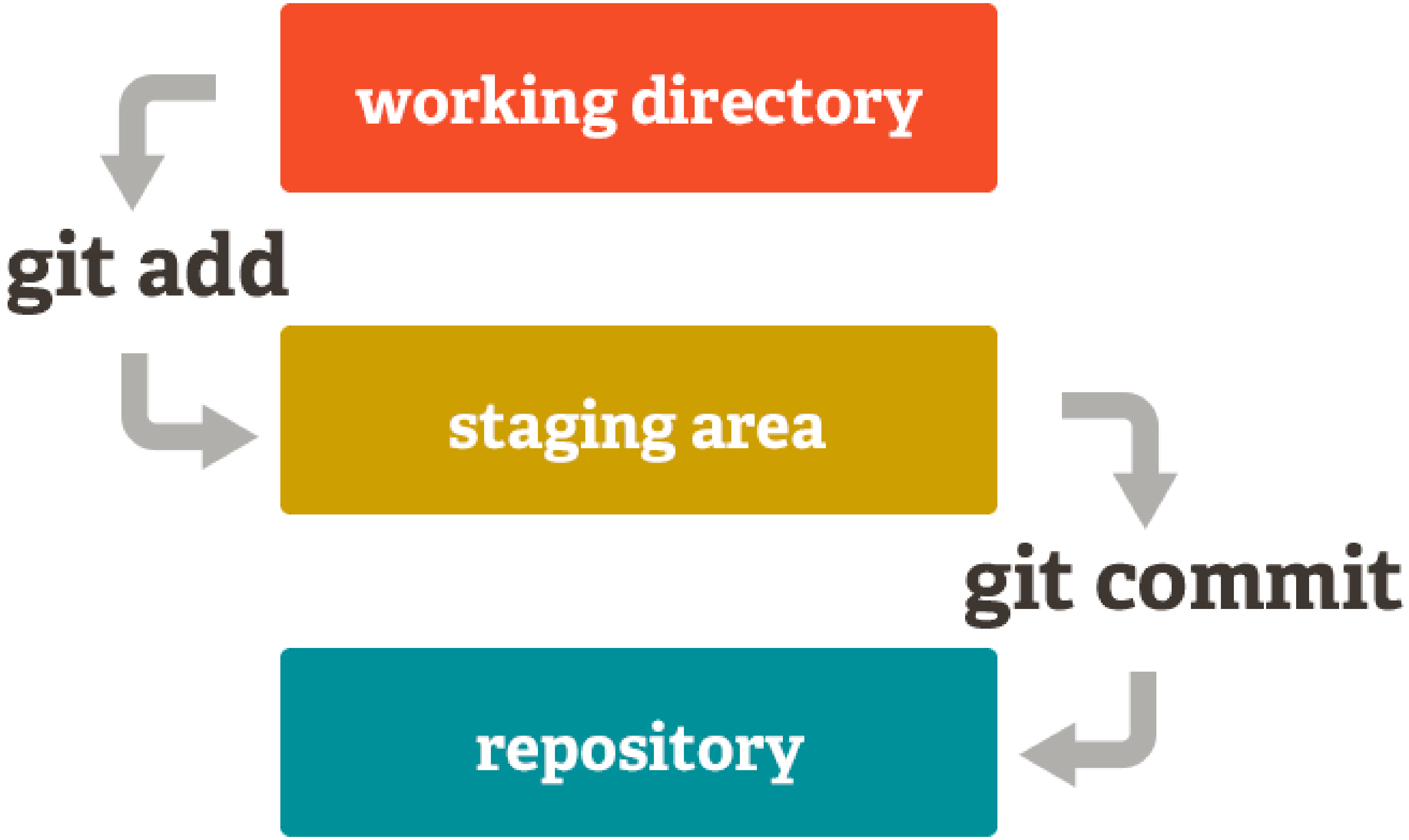
subteam fetches

bob

clair

Local Operations





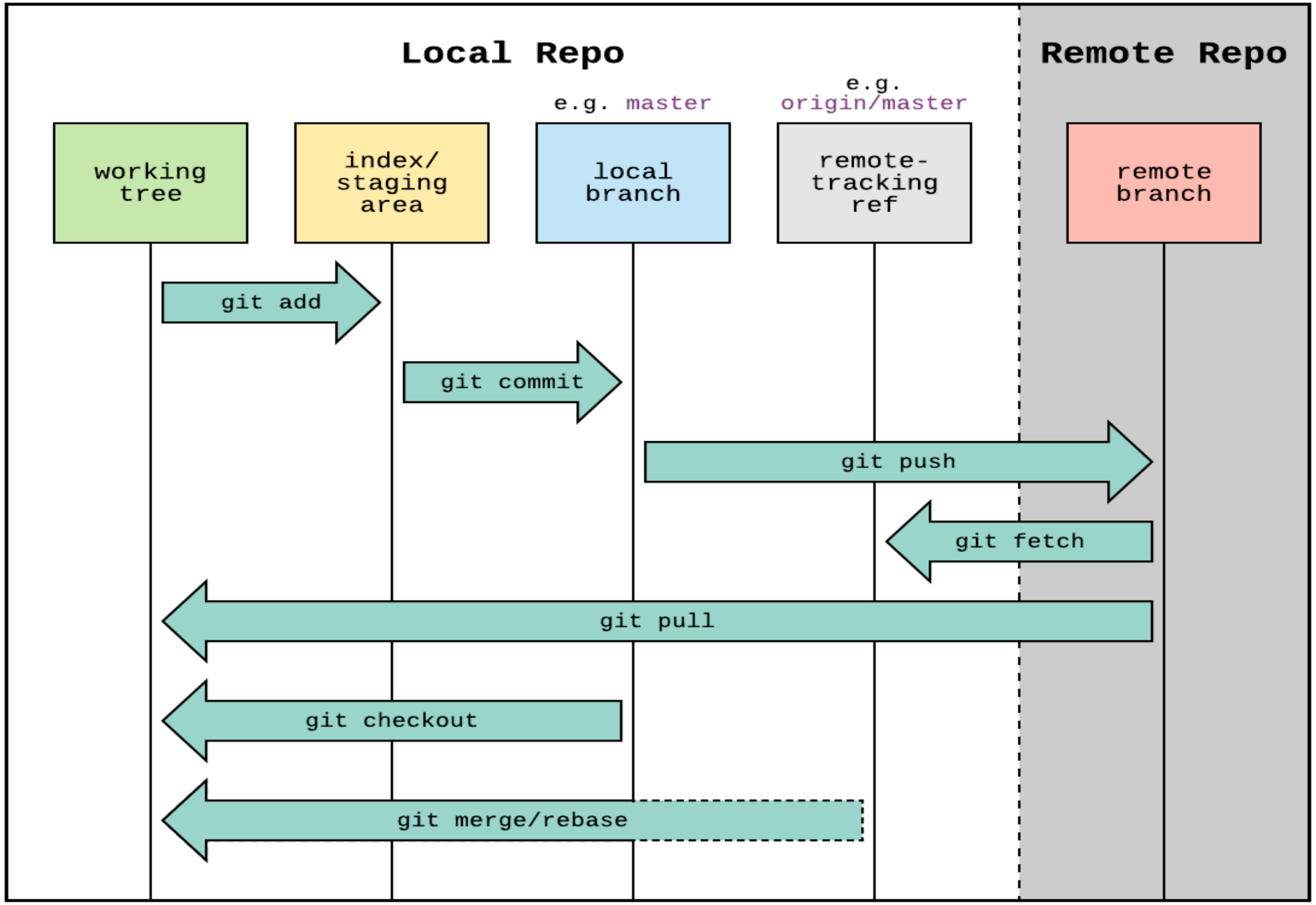
working directory

git add

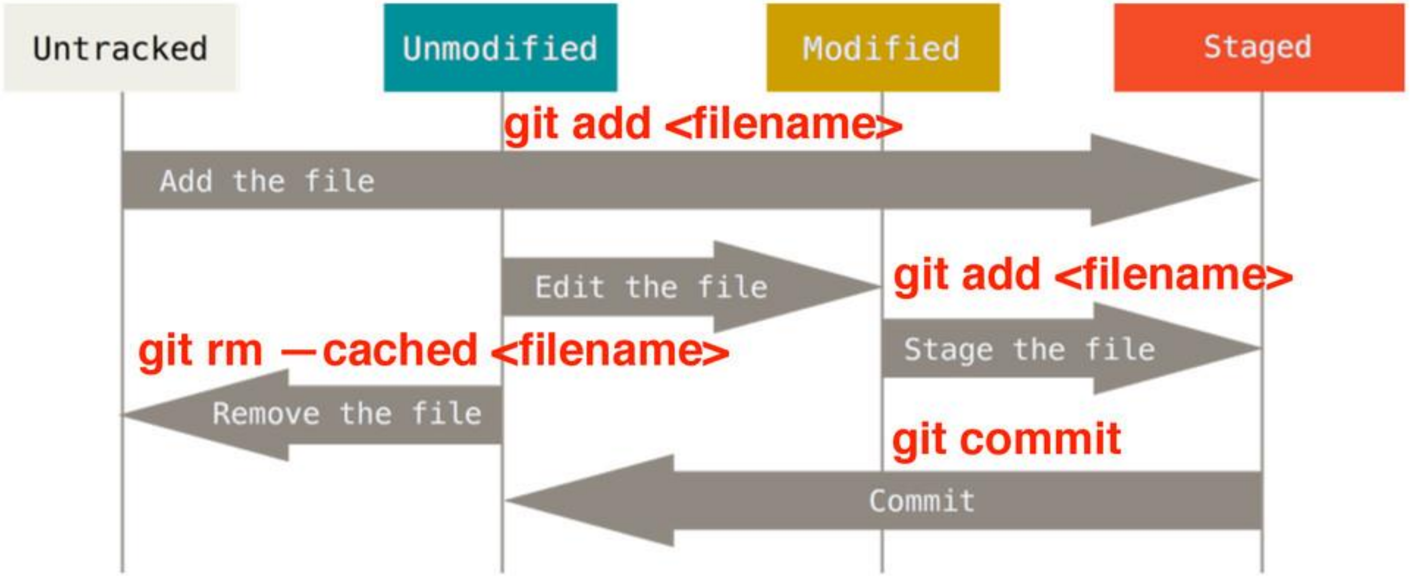
staging area

git commit

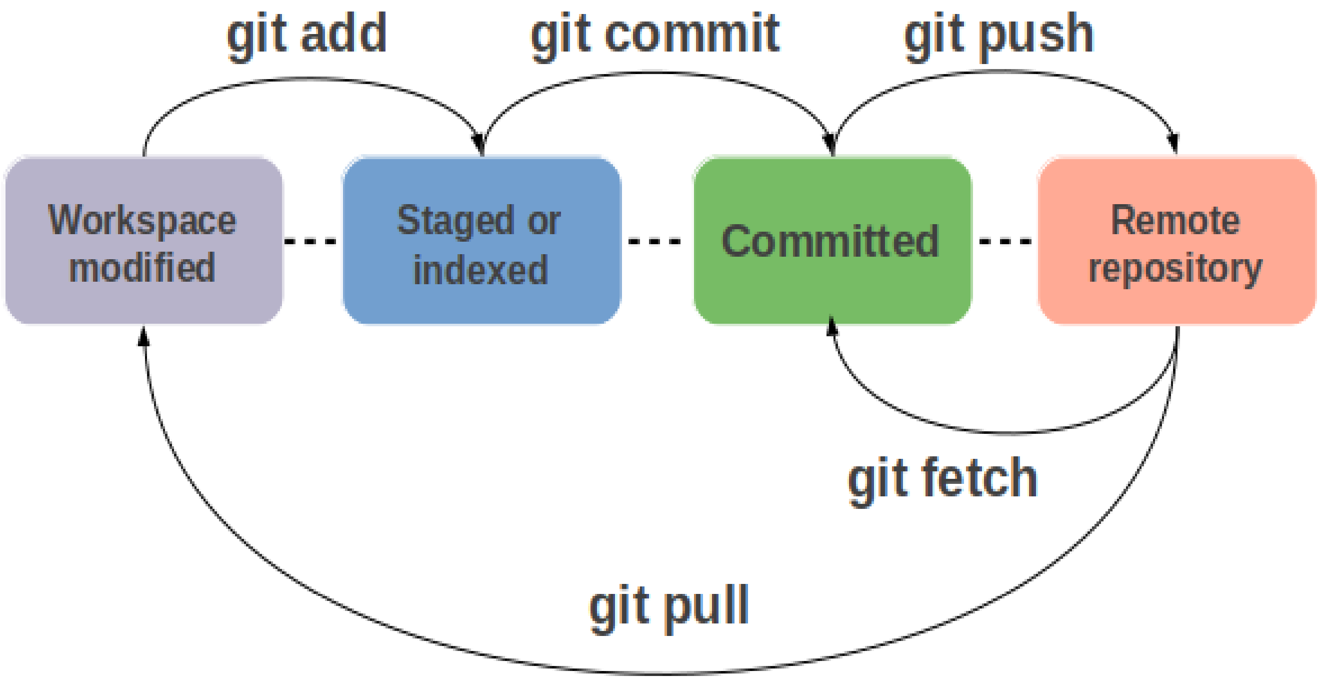
repository

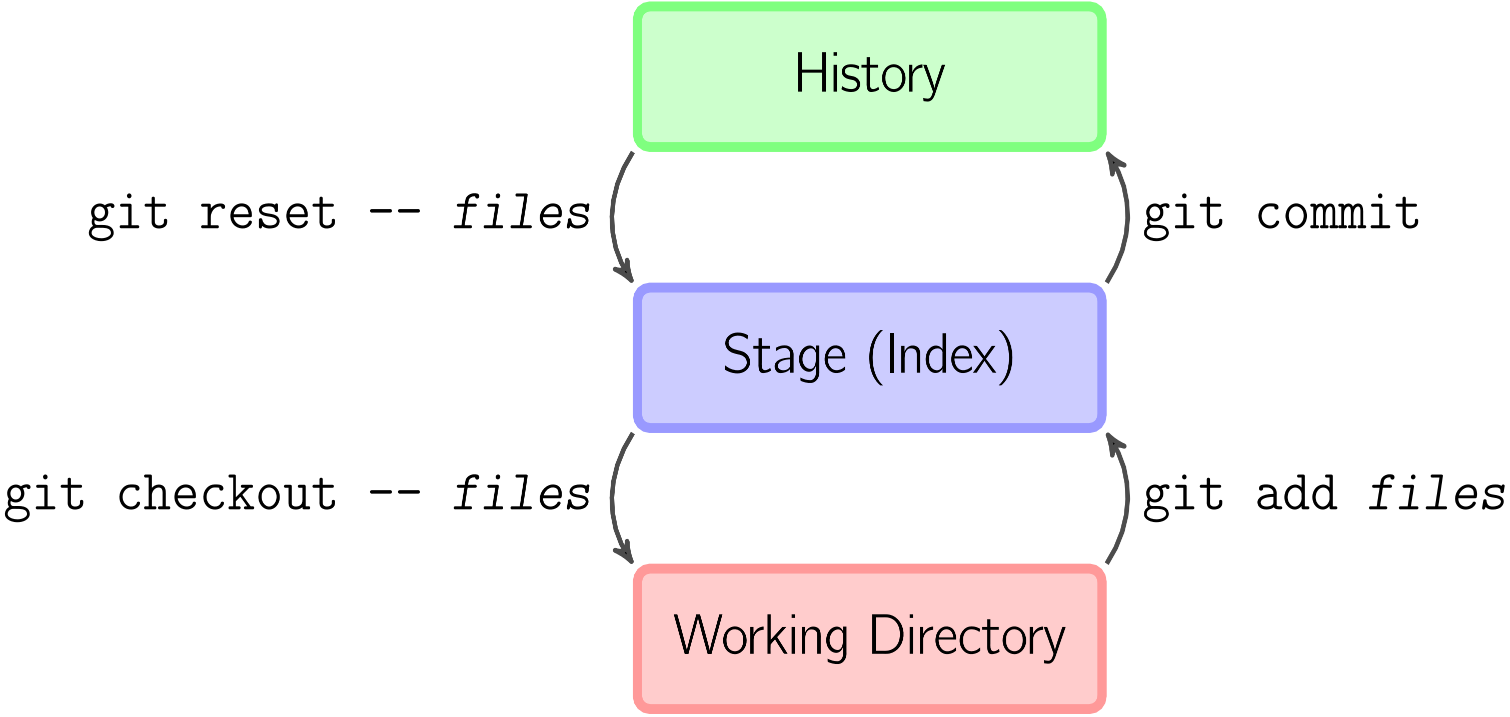

Local Repo
Remote Repo
e.g. master
e.g. origin/master
working tree
index/ staging area
local branch
remote-tracking ref
remote branch
git add
git commit
git push
git fetch
git pull
git checkout
git merge/rebase

Git lifecycle



- It’s the git life-cycle





History

git reset -- *files*

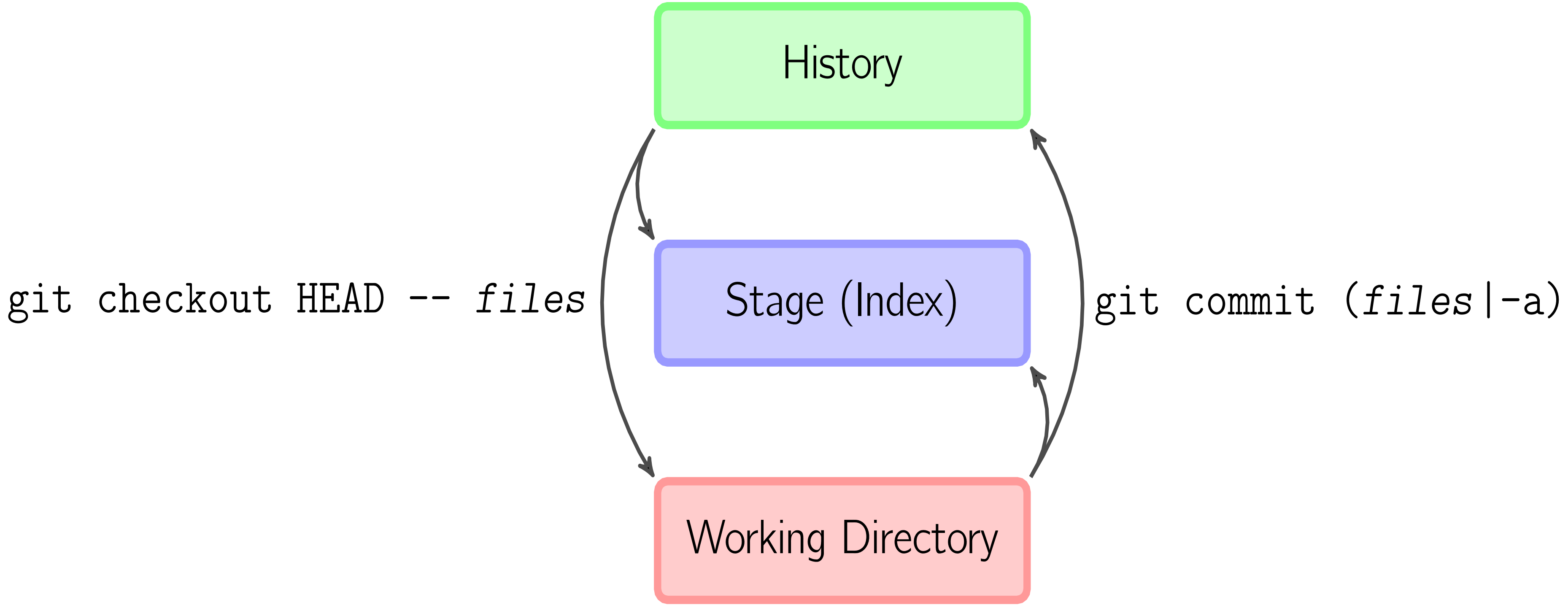
git commit

Stage (Index)

git checkout -- *files*

git add *files*

Working Directory



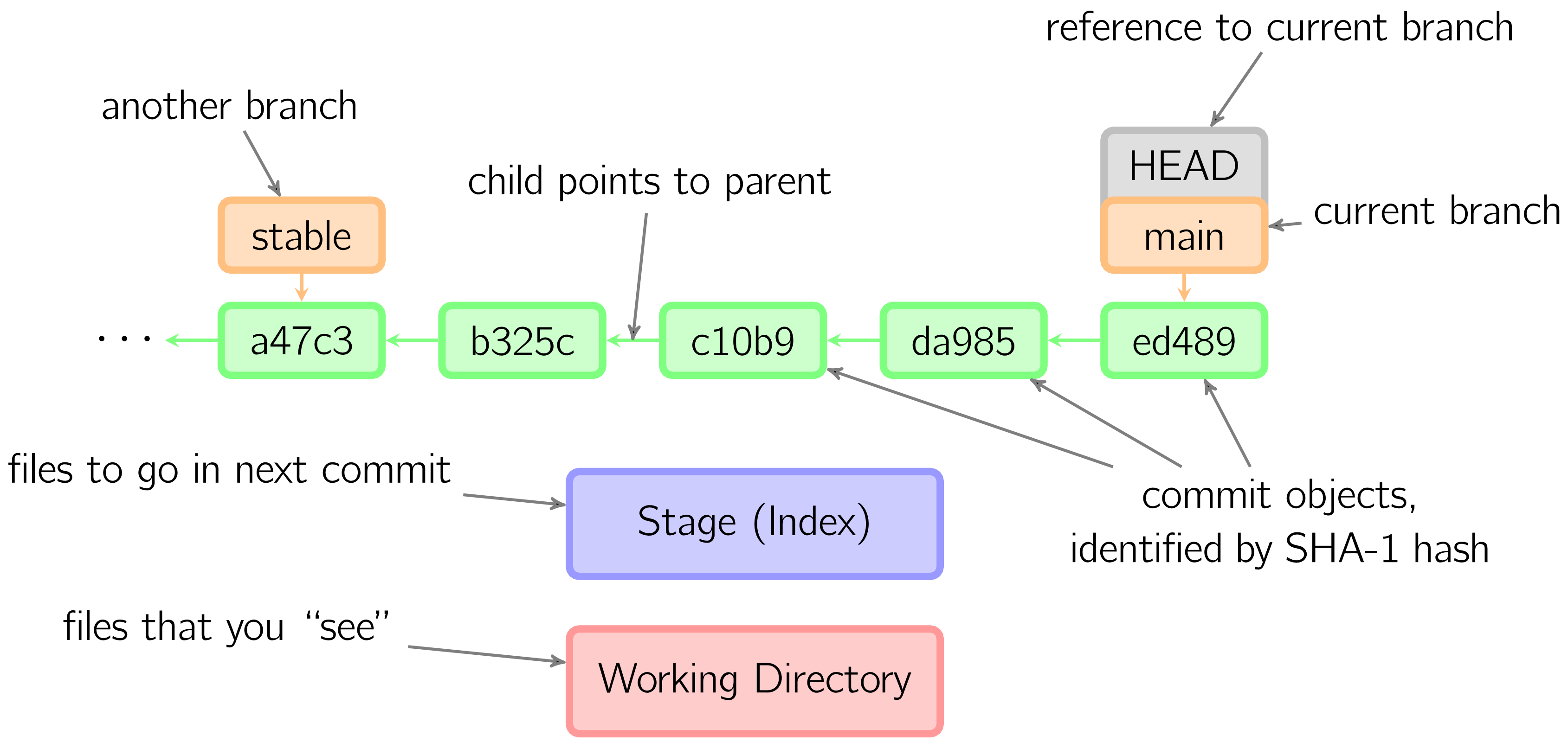
History

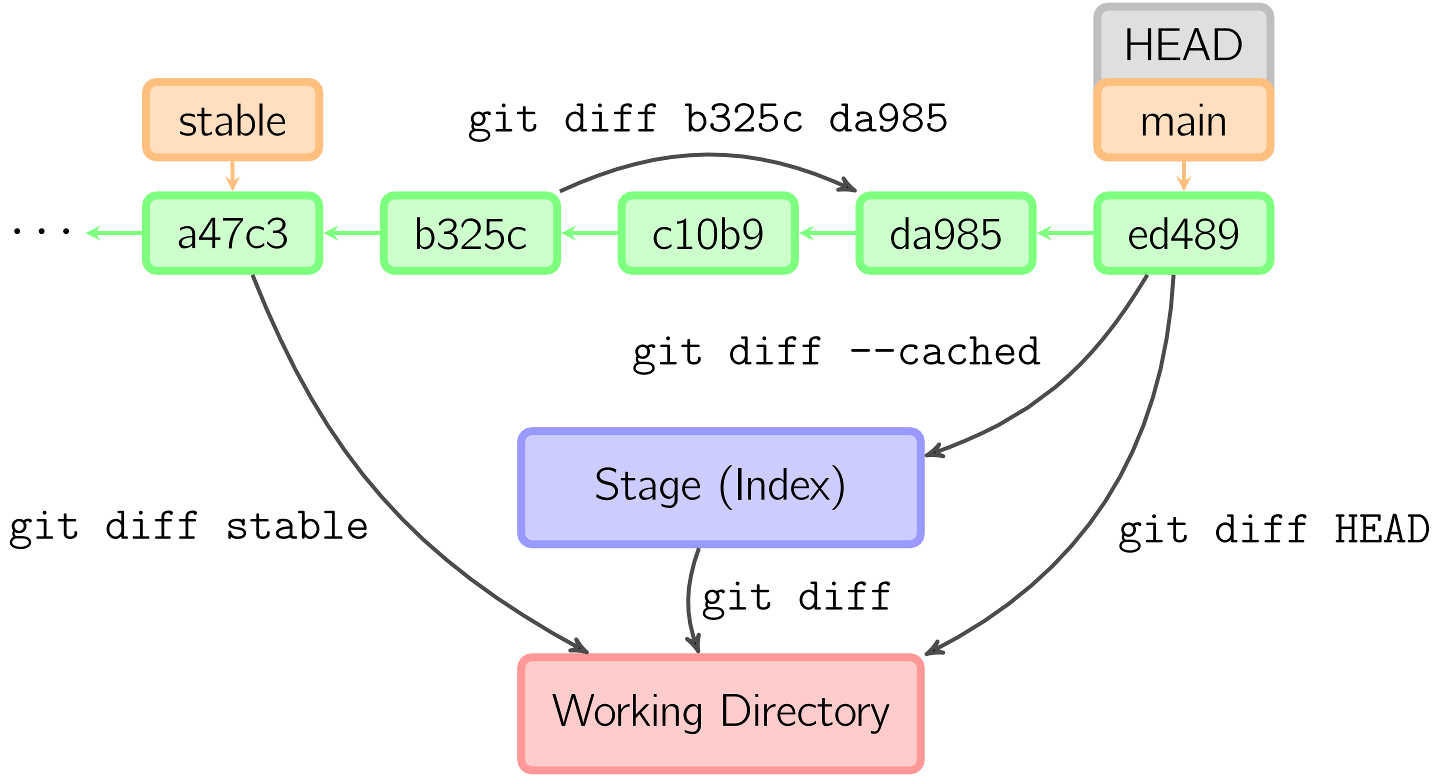
Stage (Index)

Working Directory

`git checkout HEAD -- `*`files`*

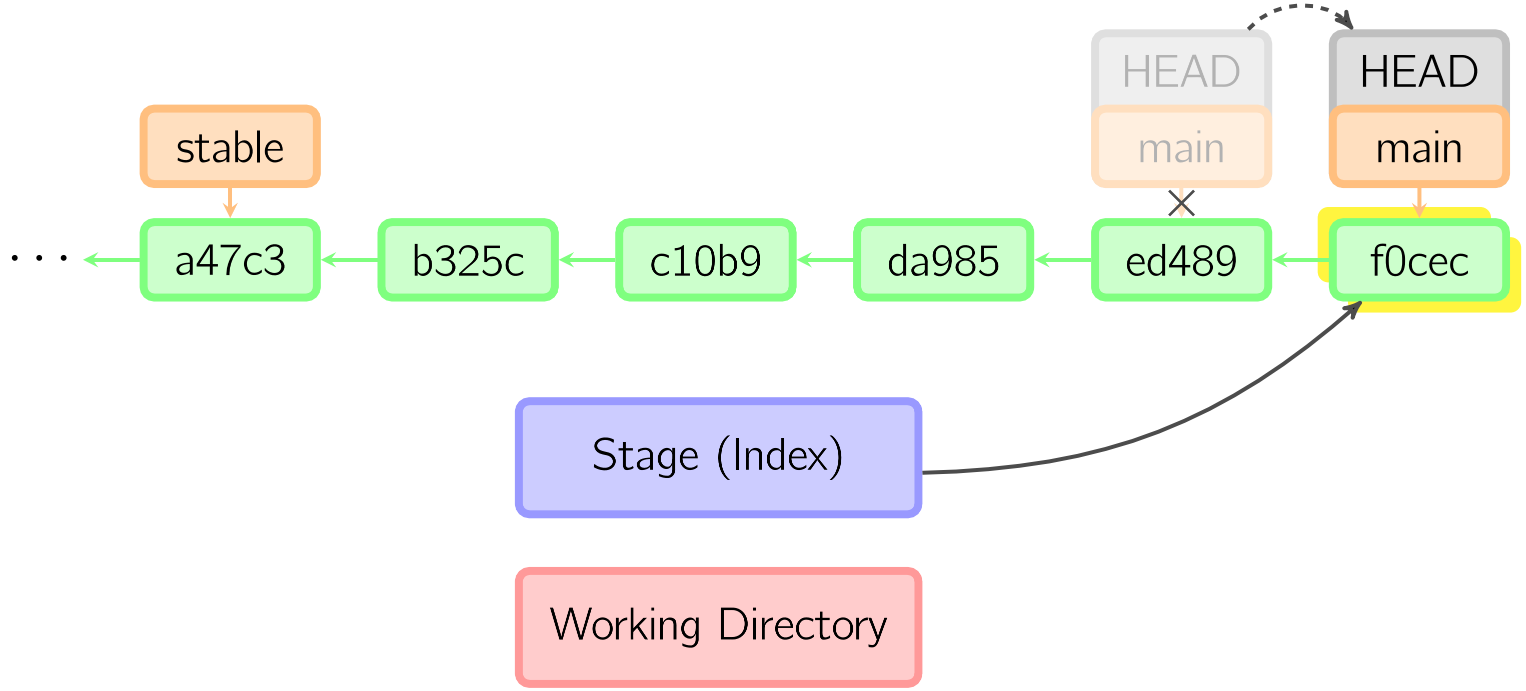
`git commit (`*`files`*`|-a)`


reference to current branch
another branch
HEAD
child points to parent
current branch
stable
main
. . .
a47c3
b325c
c10b9
da985
ed489
files to go in next commit
Stage (Index)
commit objects,
identified by SHA-1 hash
files that you "see"
Working Directory

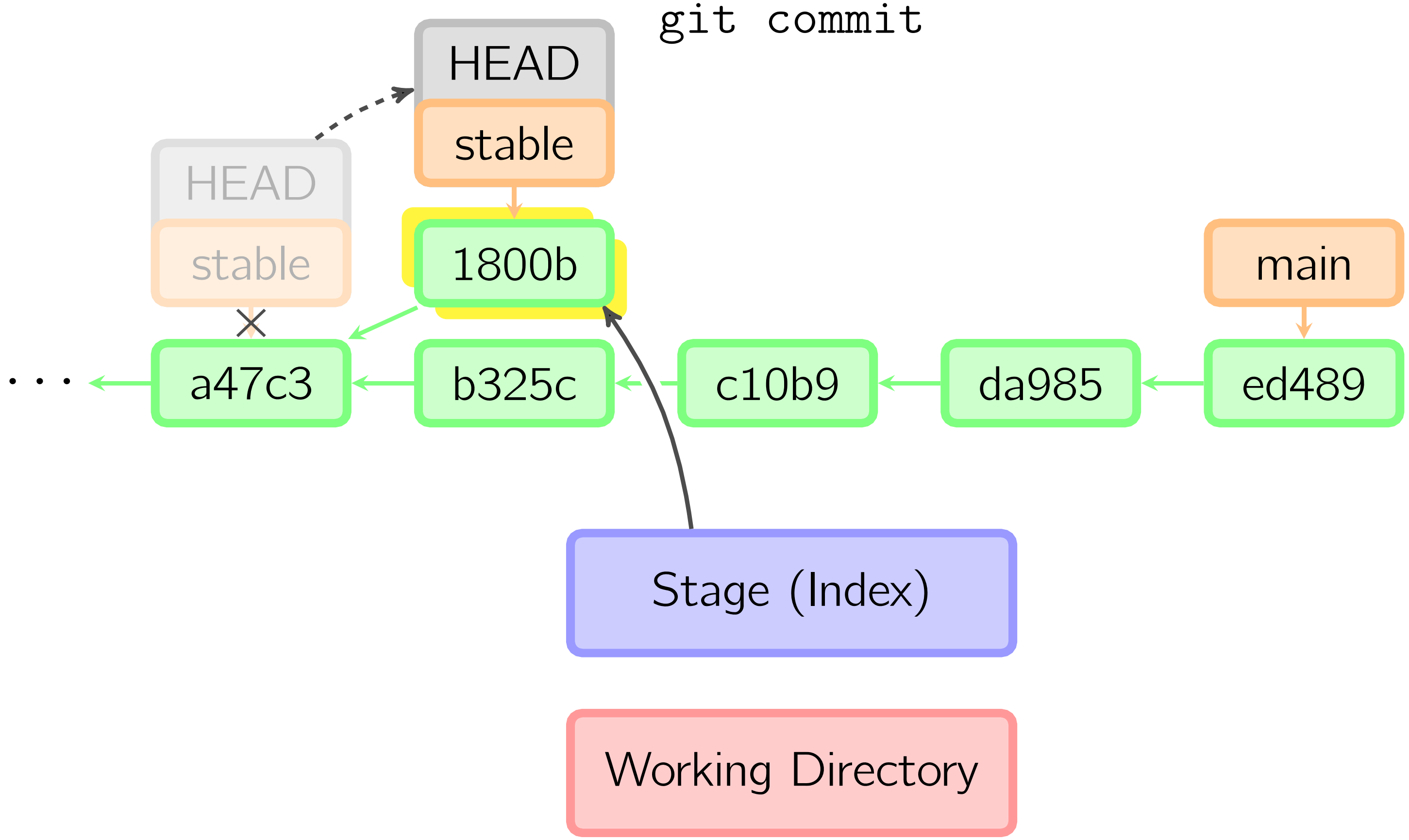


HEAD
stable
main
...
a47c3
b325c
c10b9
da985
ed489
git diff b325c da985
git diff --cached
Stage (Index)
git diff stable
git diff HEAD
git diff
Working Directory

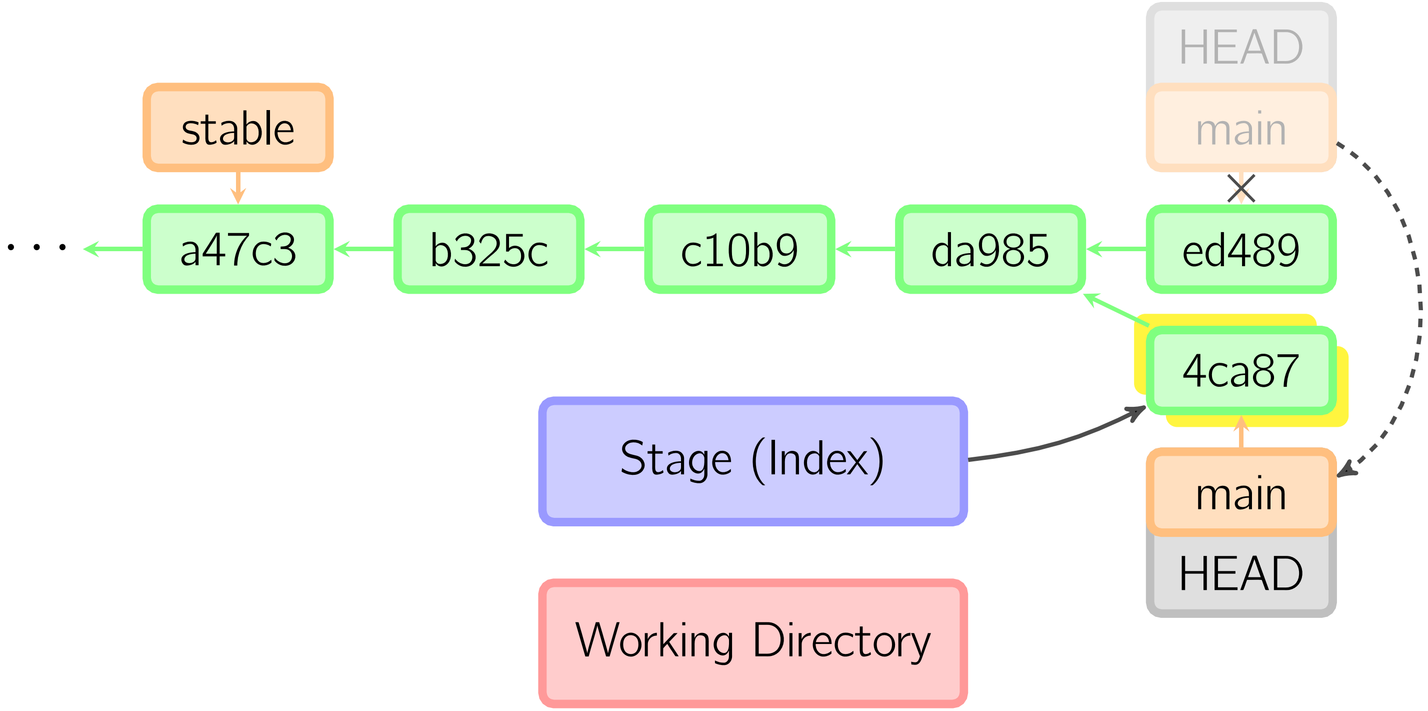
```
git commit
```


HEAD
HEAD
stable
main
main
. . .
a47c3
b325c
c10b9
da985
ed489
f0cec
Stage (Index)
Working Directory

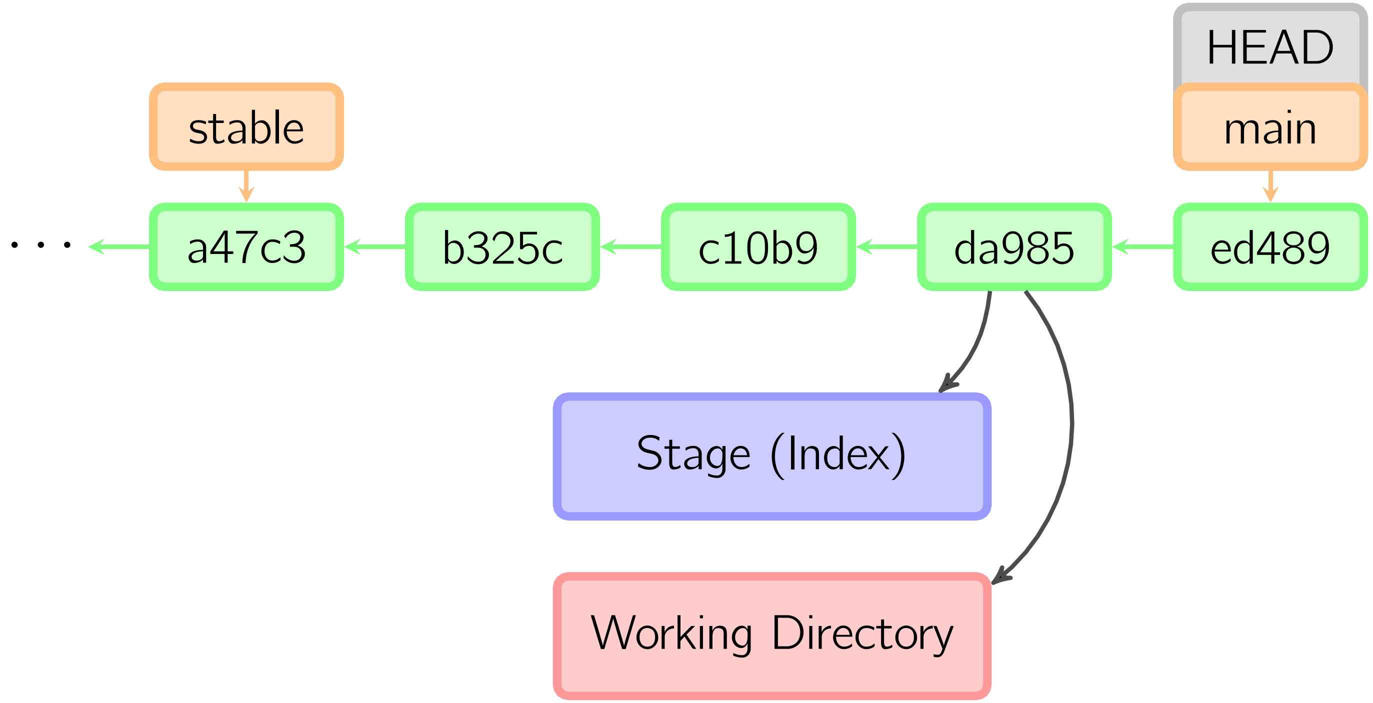
`git commit`


HEAD
stable
HEAD
stable
1800b
main
· · ·
a47c3
b325c
c10b9
da985
ed489
Stage (Index)
Working Directory

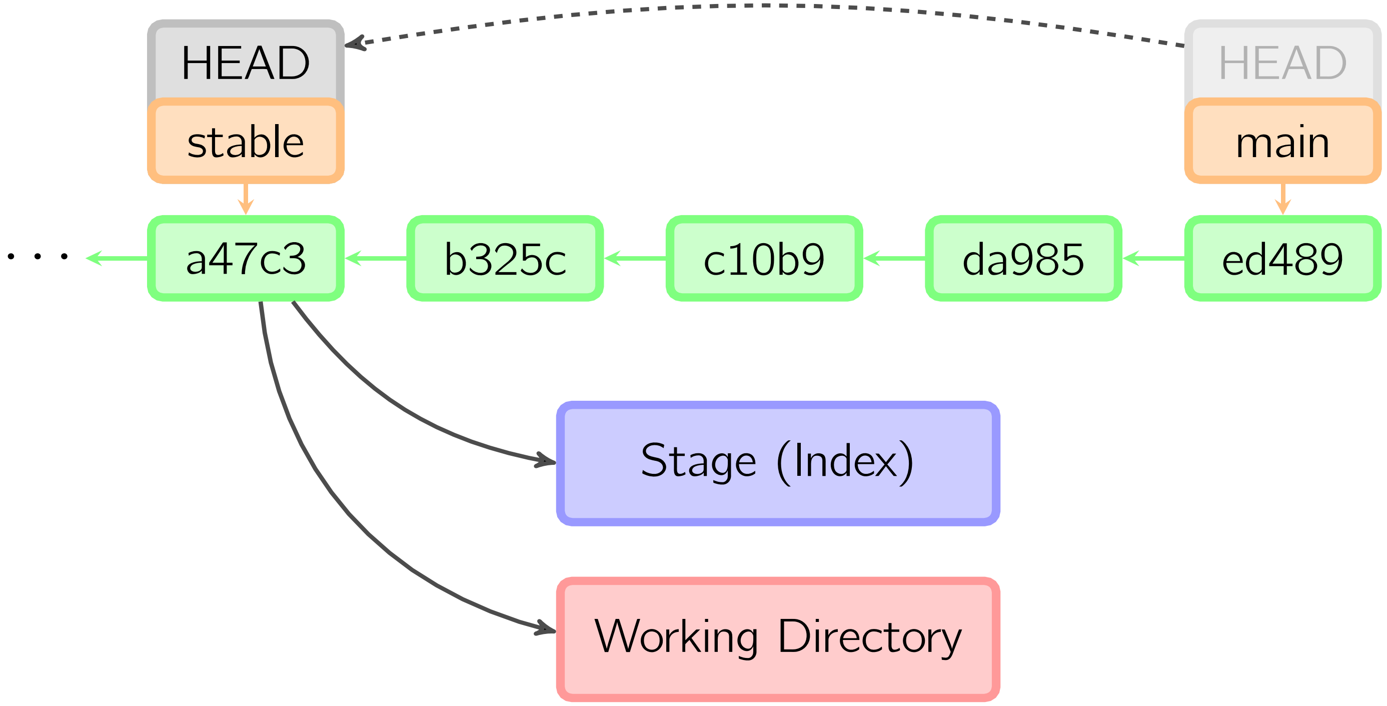
```
git commit --amend
```



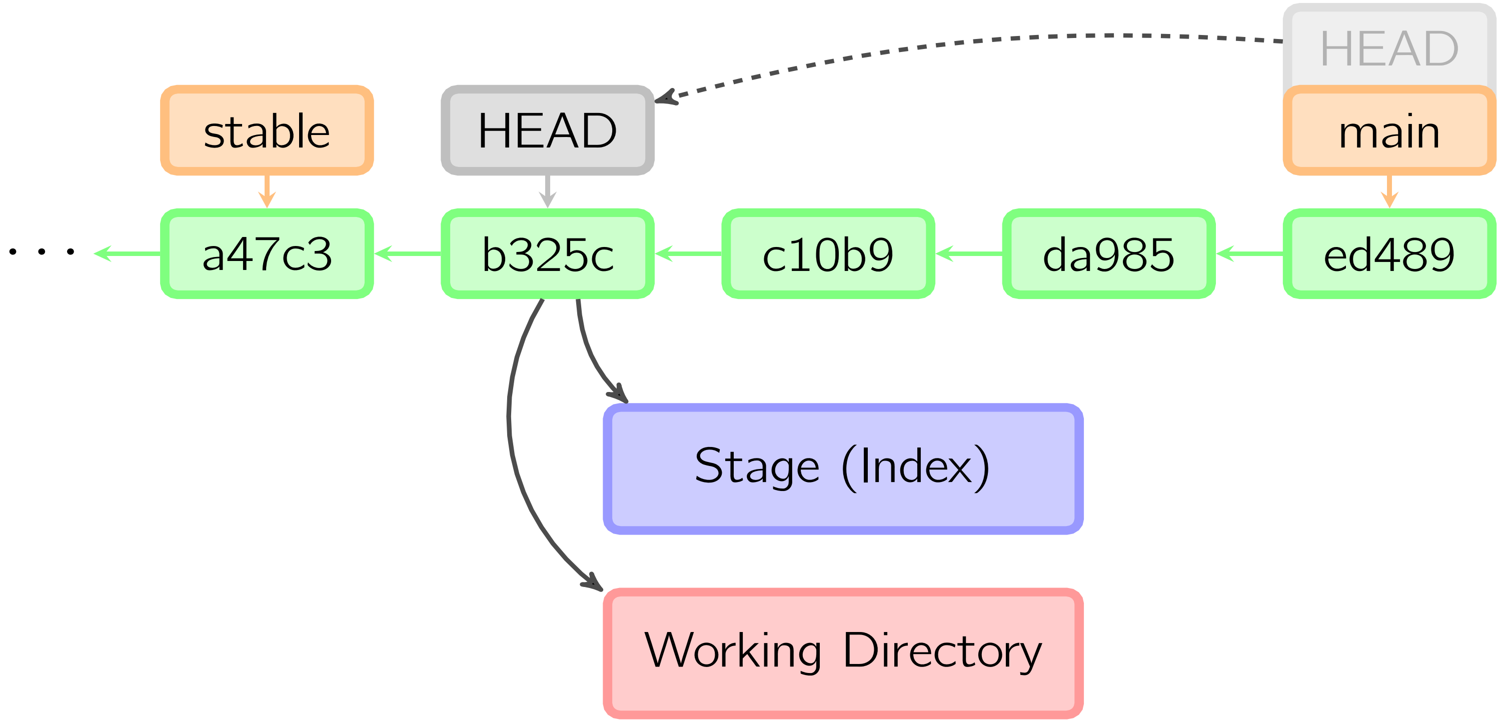
```
git checkout HEAD~ files
```



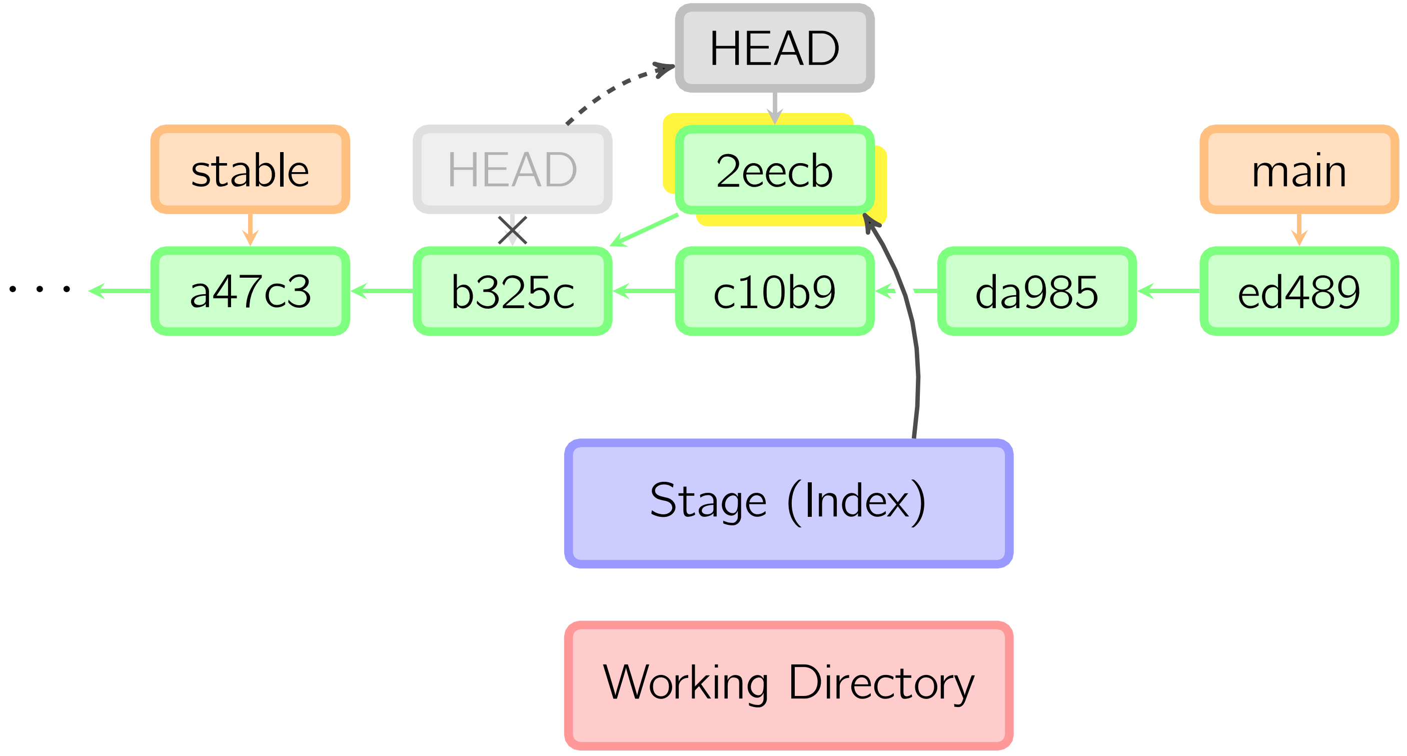
```
git checkout stable
```



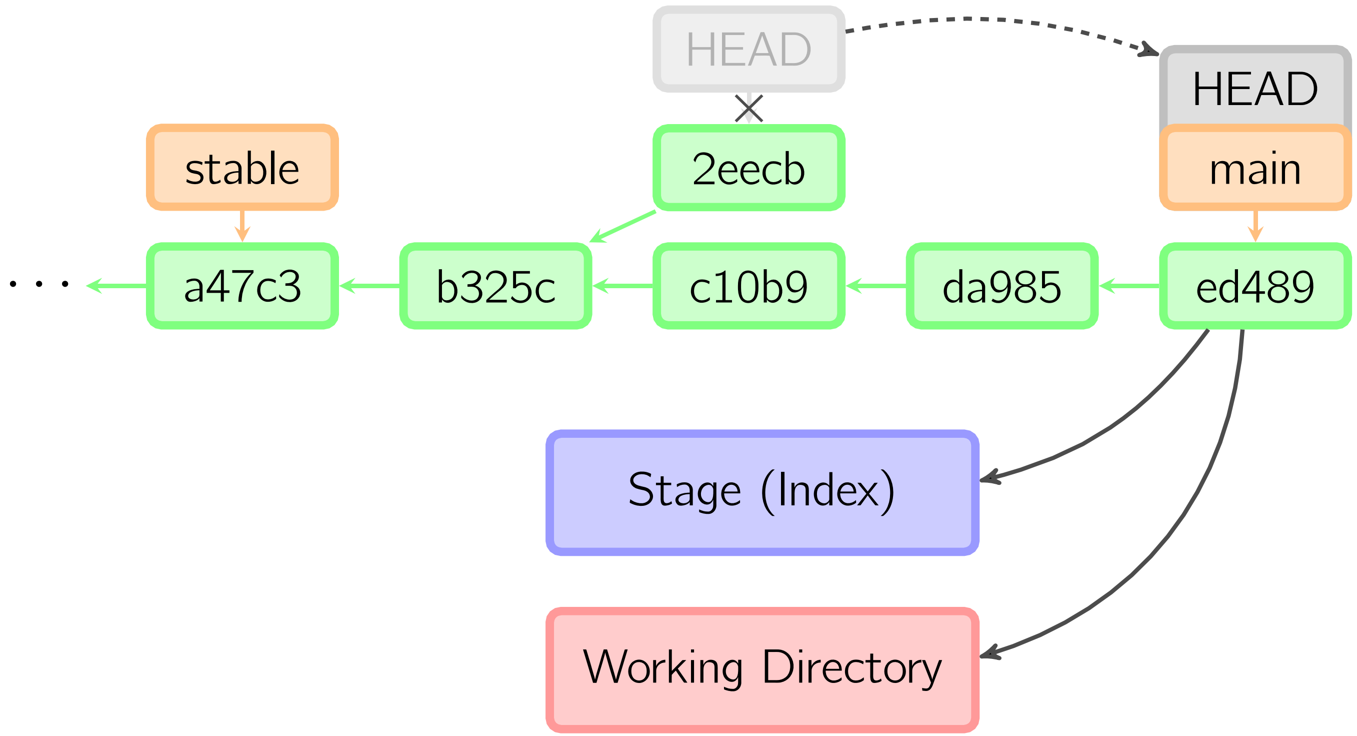
```
git checkout main~3
```



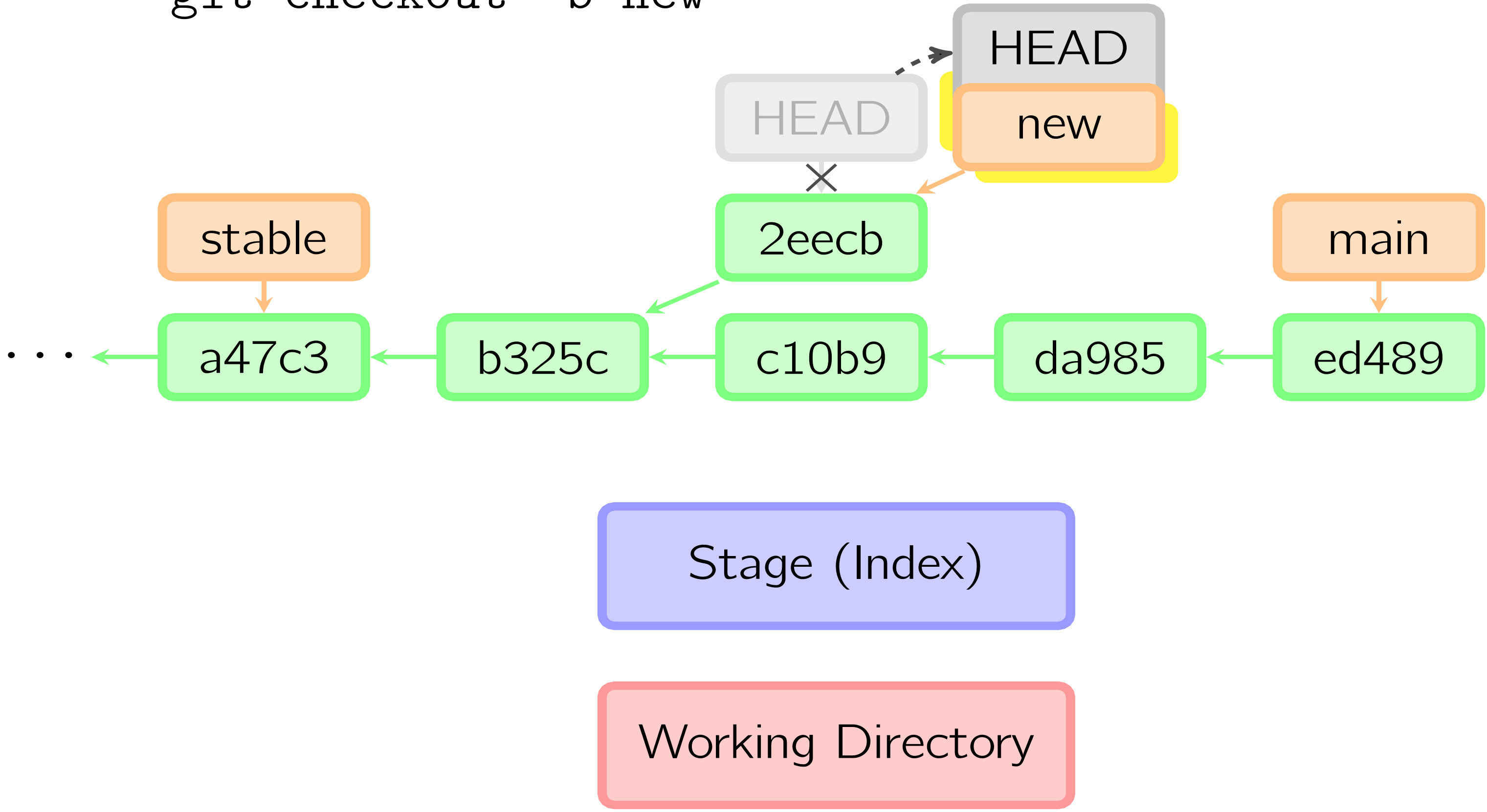
`git commit`


HEAD
stable
HEAD
2eecb
main
. . .
a47c3
b325c
c10b9
da985
ed489
Stage (Index)
Working Directory

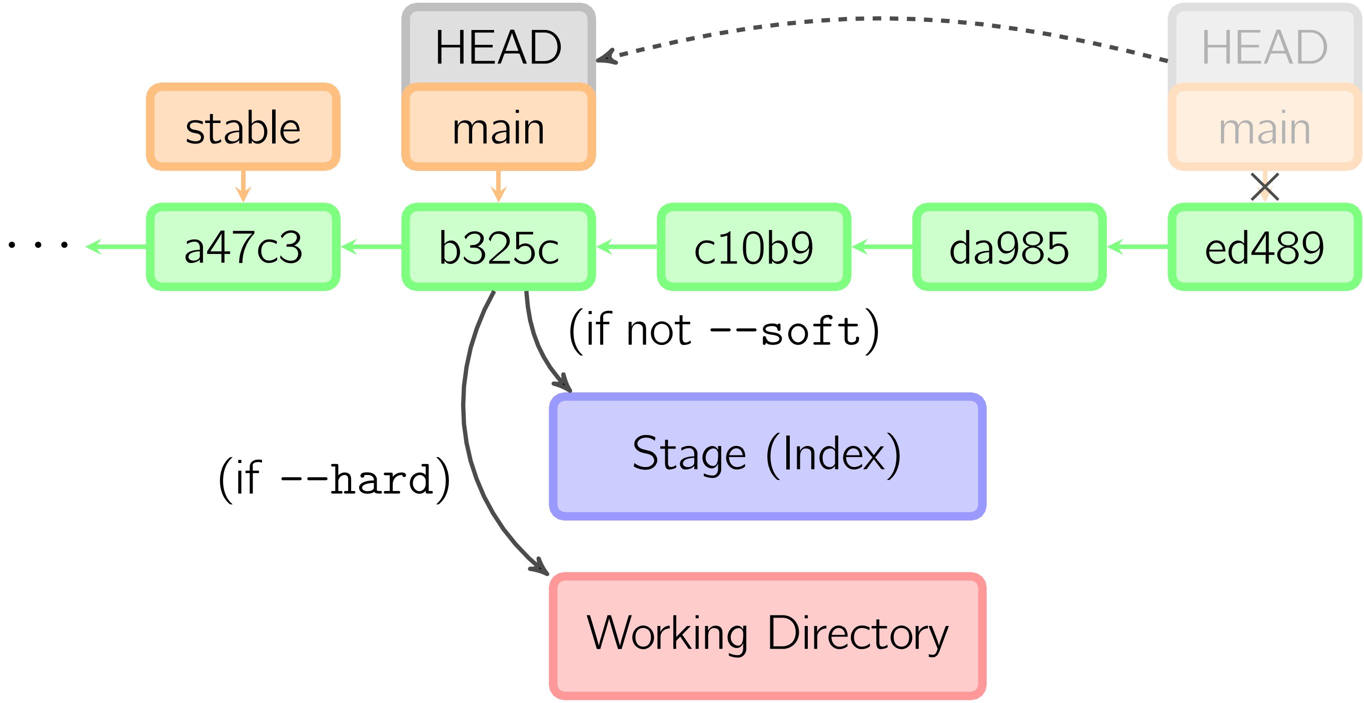
```
git checkout main
```



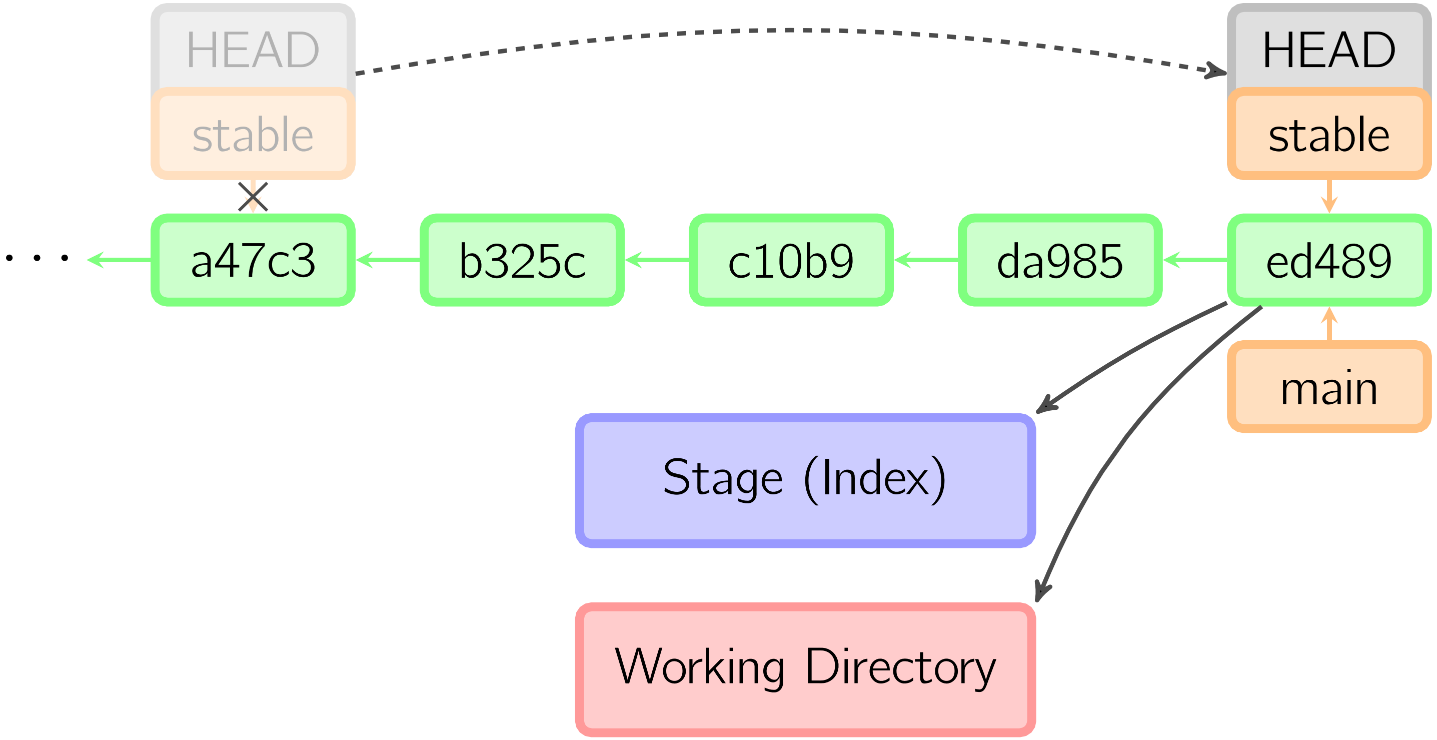
```
git checkout -b new
```



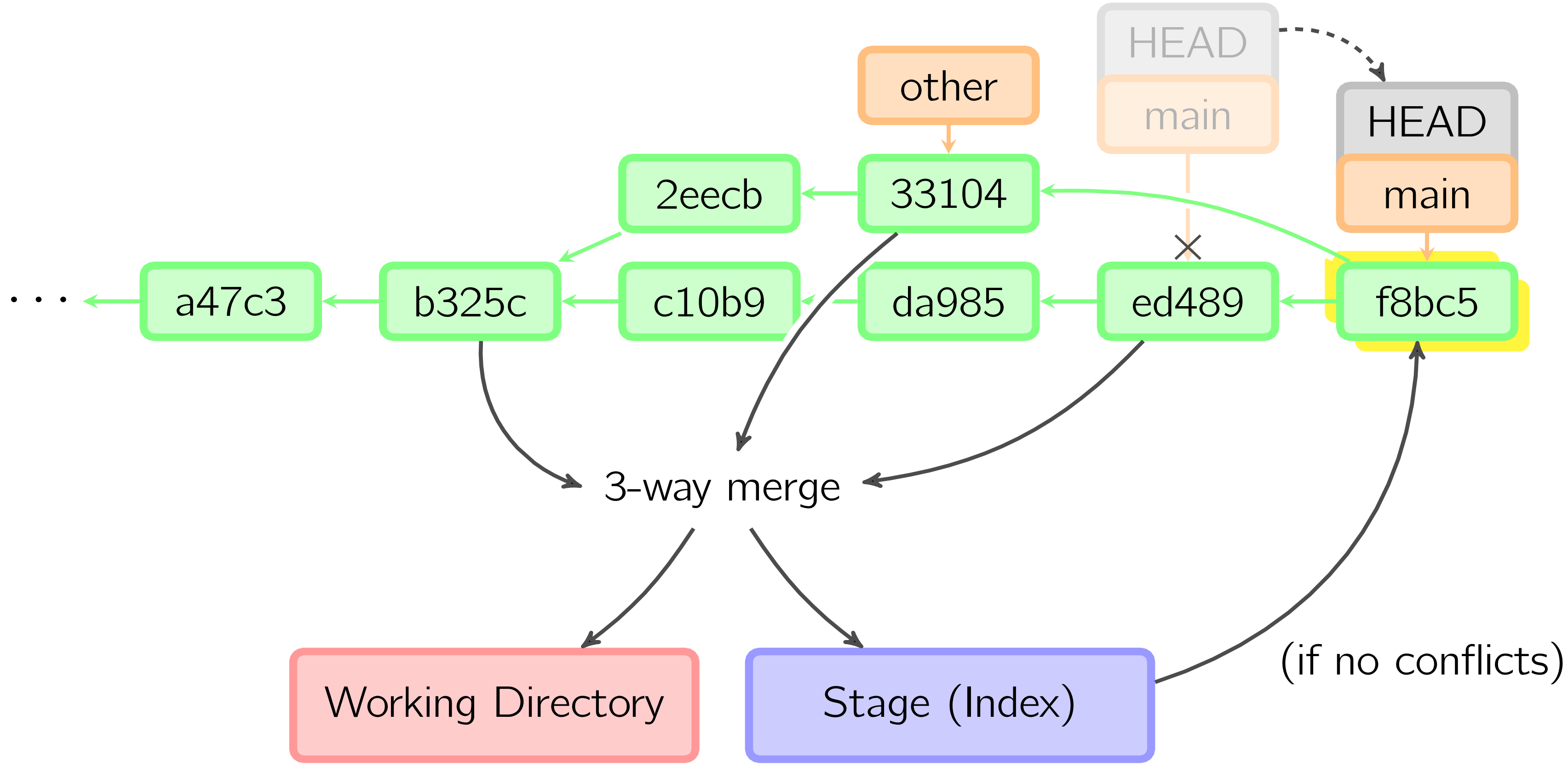
```
git reset HEAD~3
```



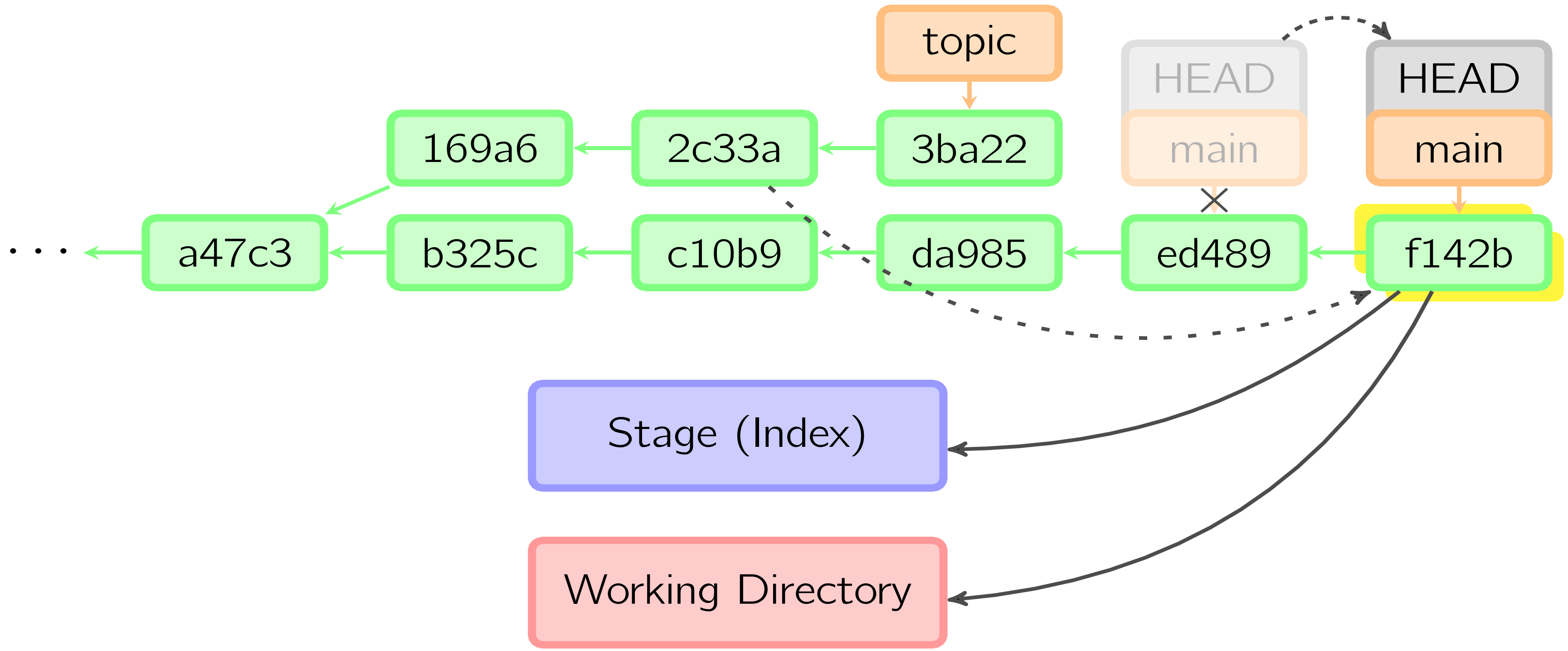
```
git merge main
```

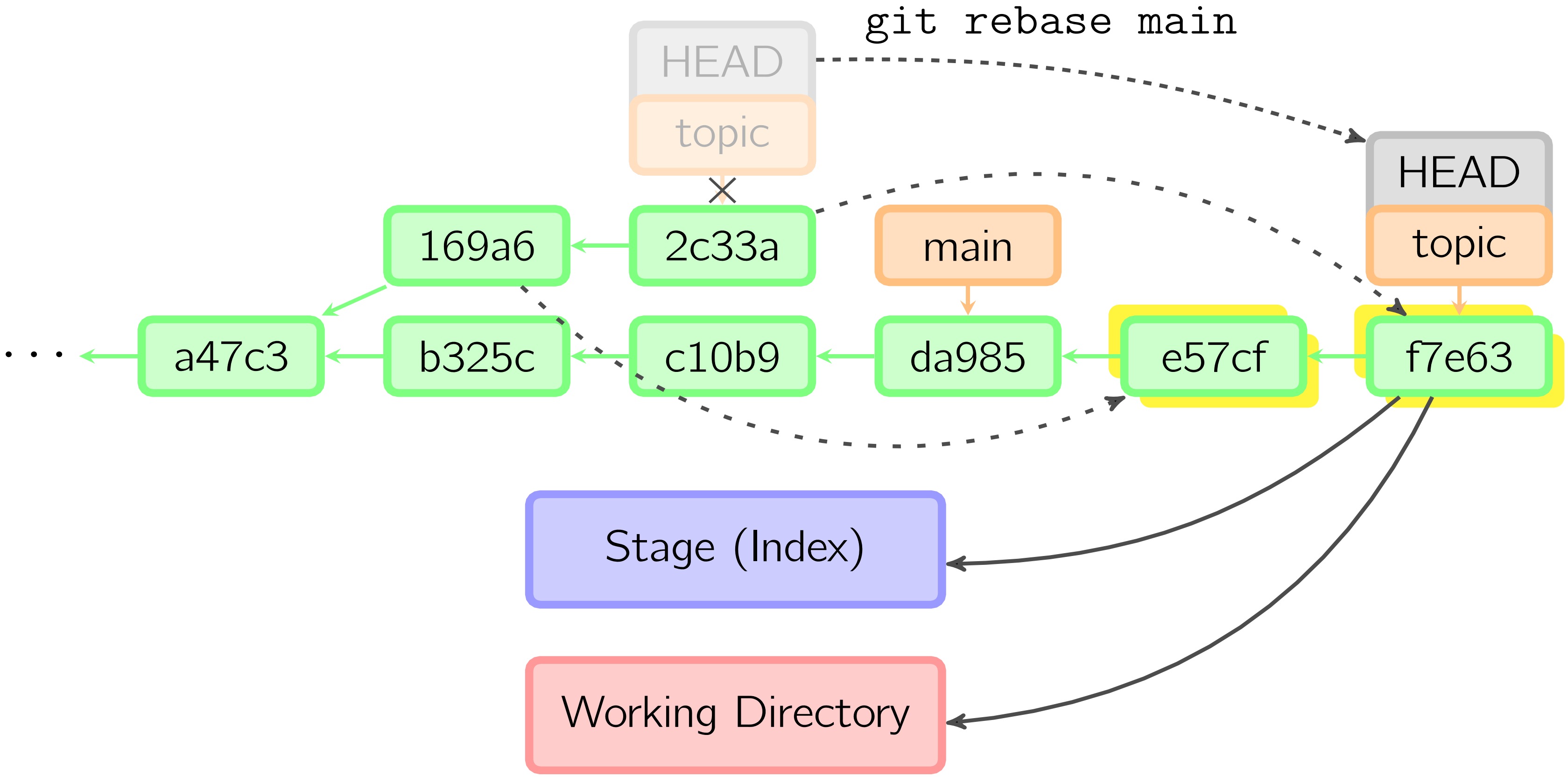


```
git merge other
```

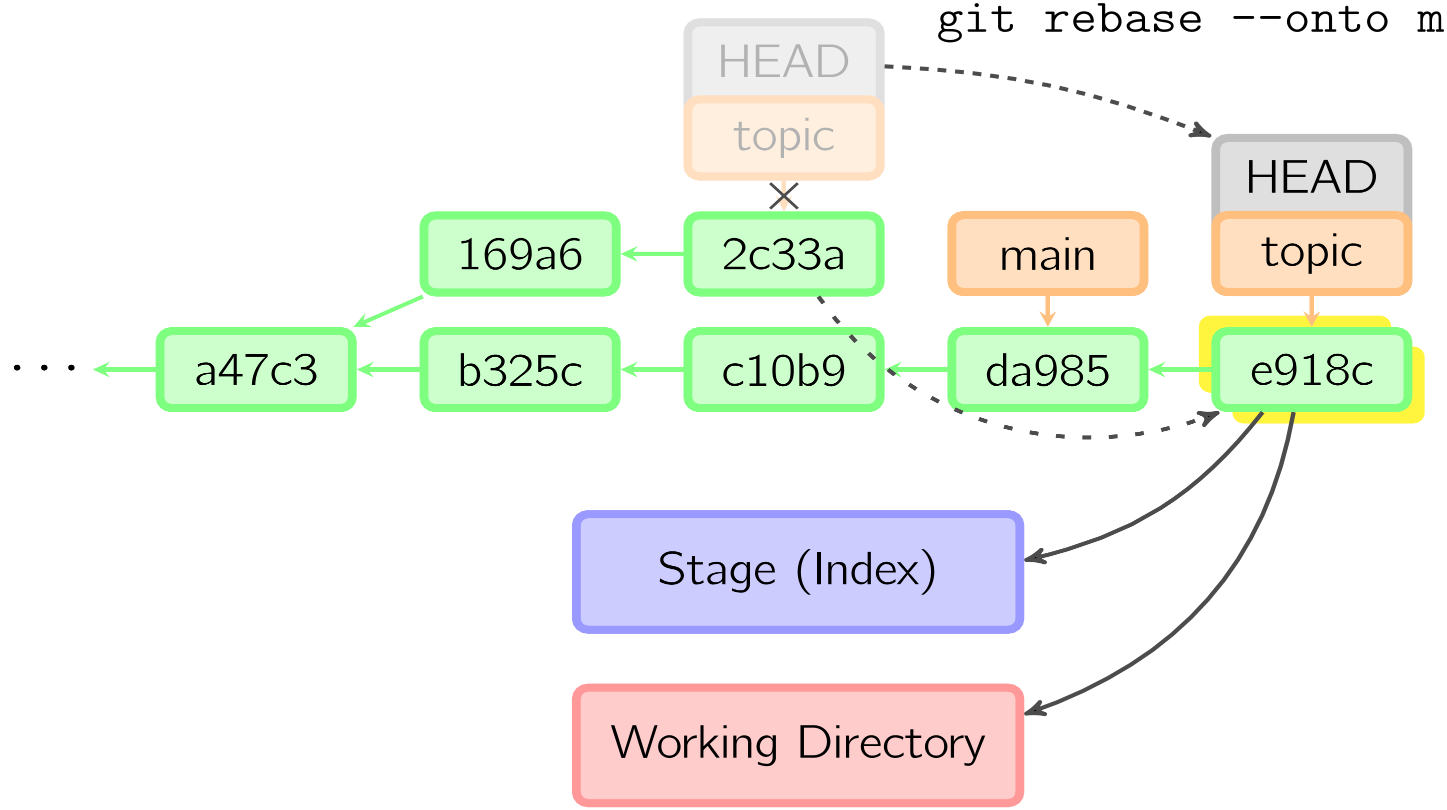


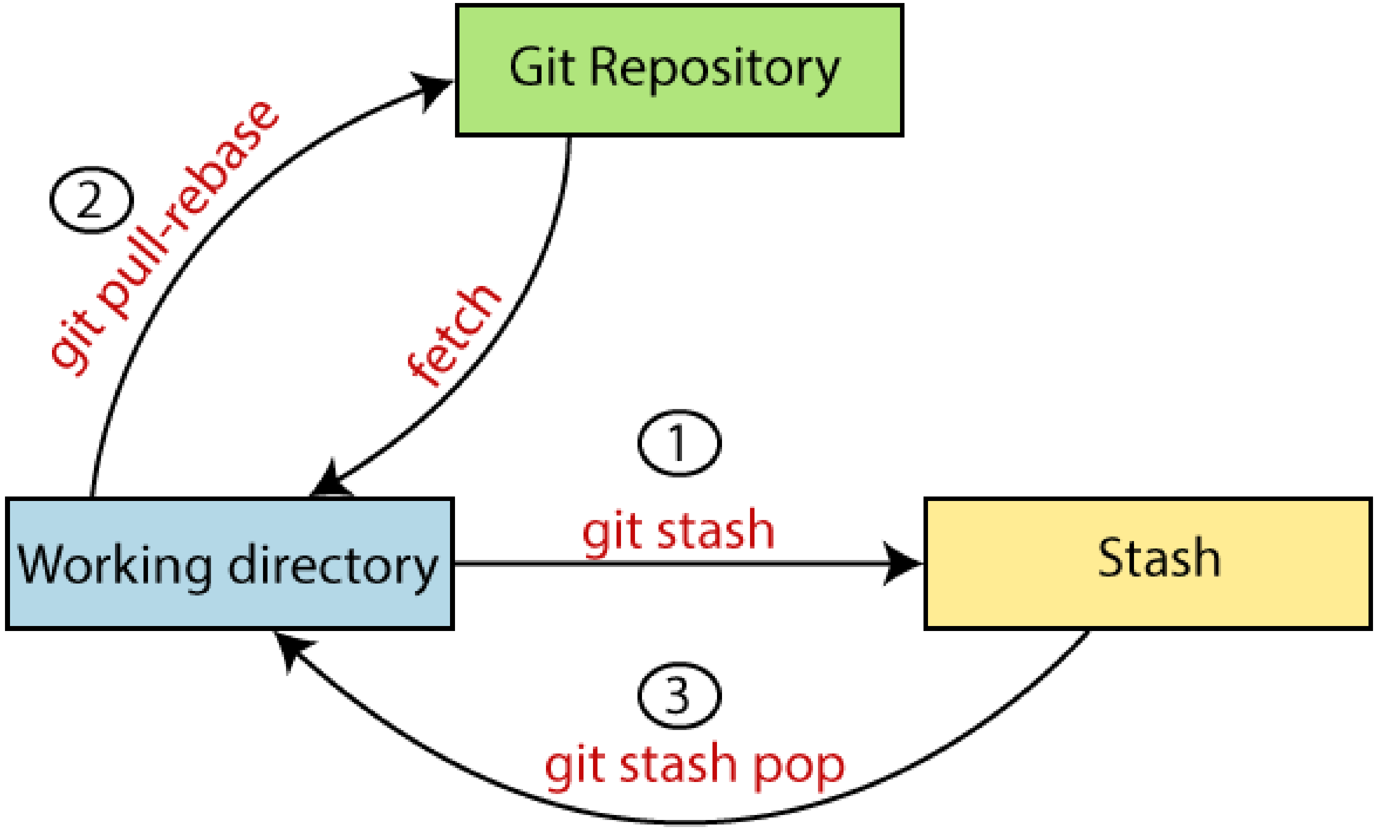
```
git cherry-pick 2c33a
```



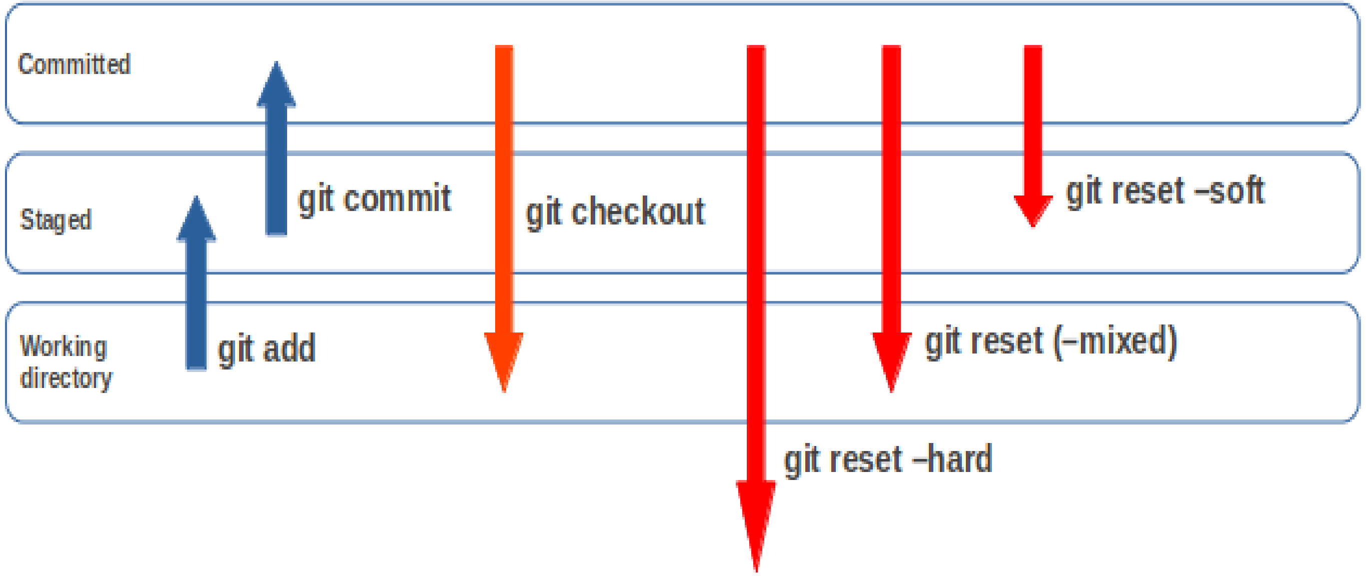

git rebase main
HEAD
topic
HEAD
topic
169a6
2c33a
main
...
a47c3
b325c
c10b9
da985
e57cf
f7e63
Stage (Index)
Working Directory

```
git rebase --onto main 169a6
```




Git Repository
2
git pull-rebase
fetch
1
git stash
Working directory
Stash
3
git stash pop

- The diagram of command `git reset`



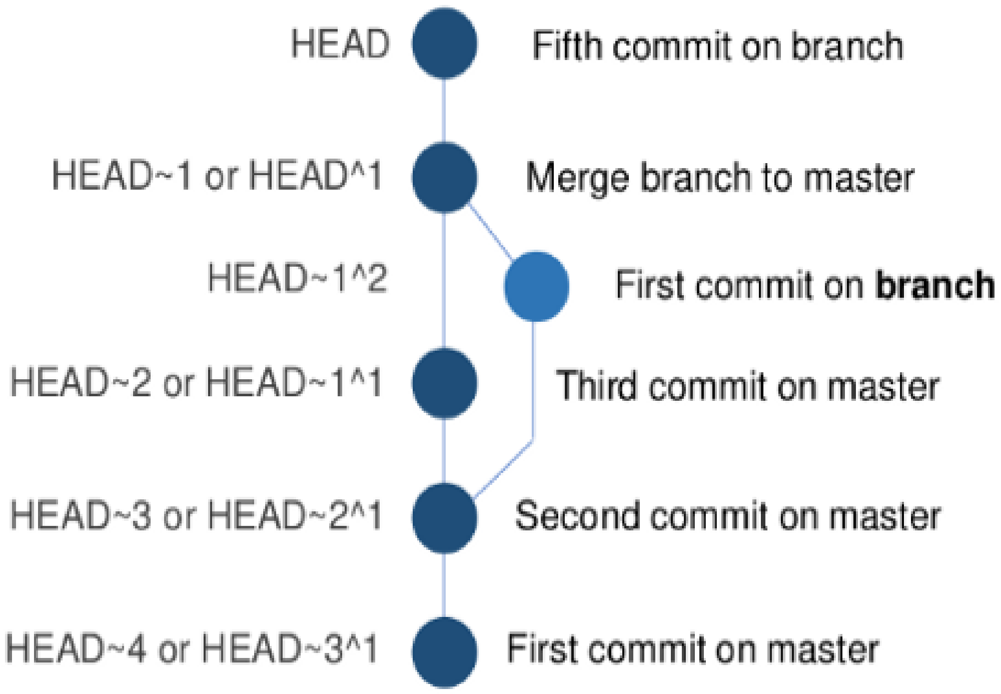
`--soft`

`--mixed`

`--hard`

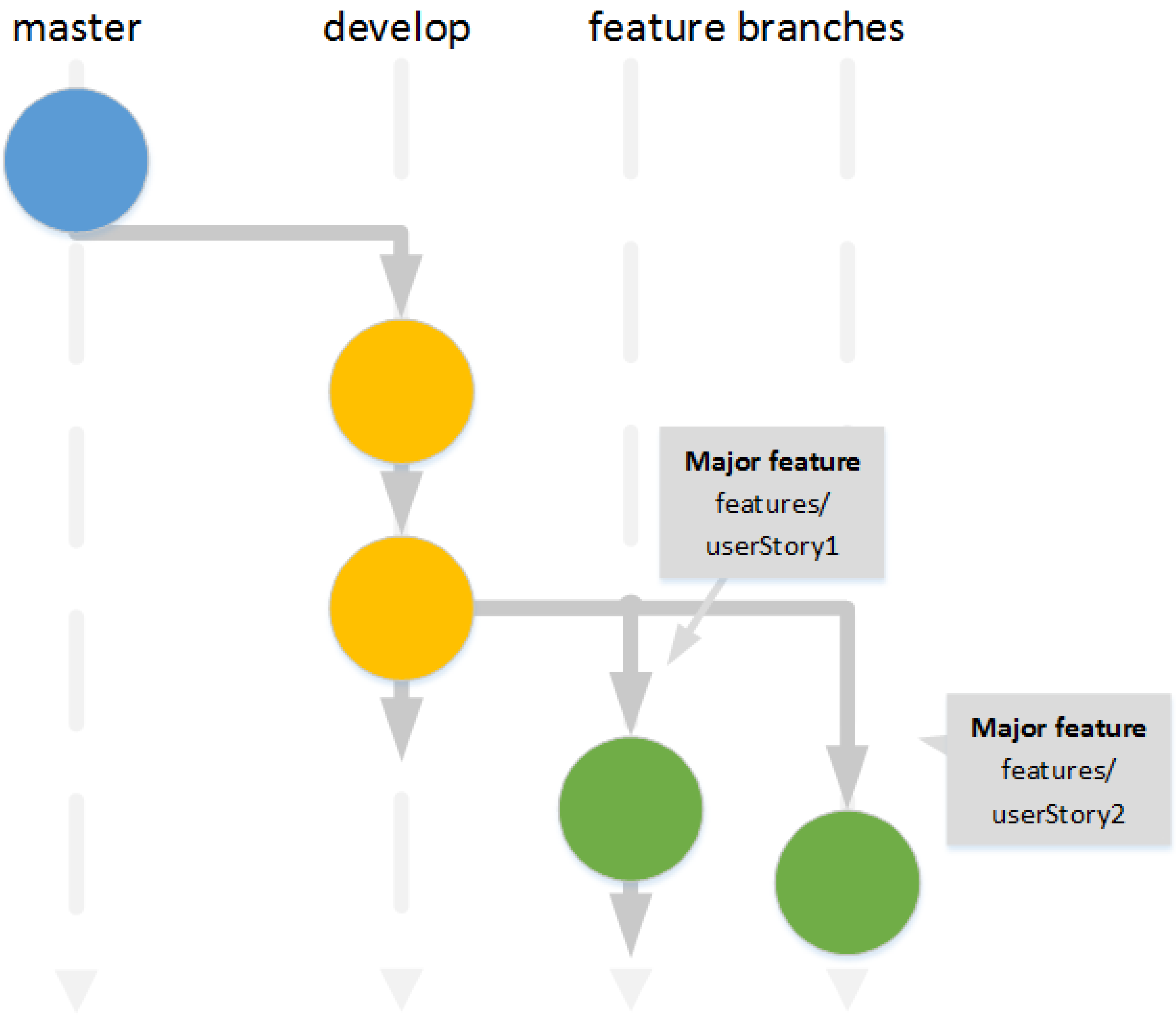
!! We should never use git **`reset <commit>`** when any snapshots after `<commit>` have been pushed to a public repository.

- **Tilde (~) and caret (^)**



tilde (~) ~N indicates the previous N-th commit

caret (^) ^N indicates the N-th parent commit



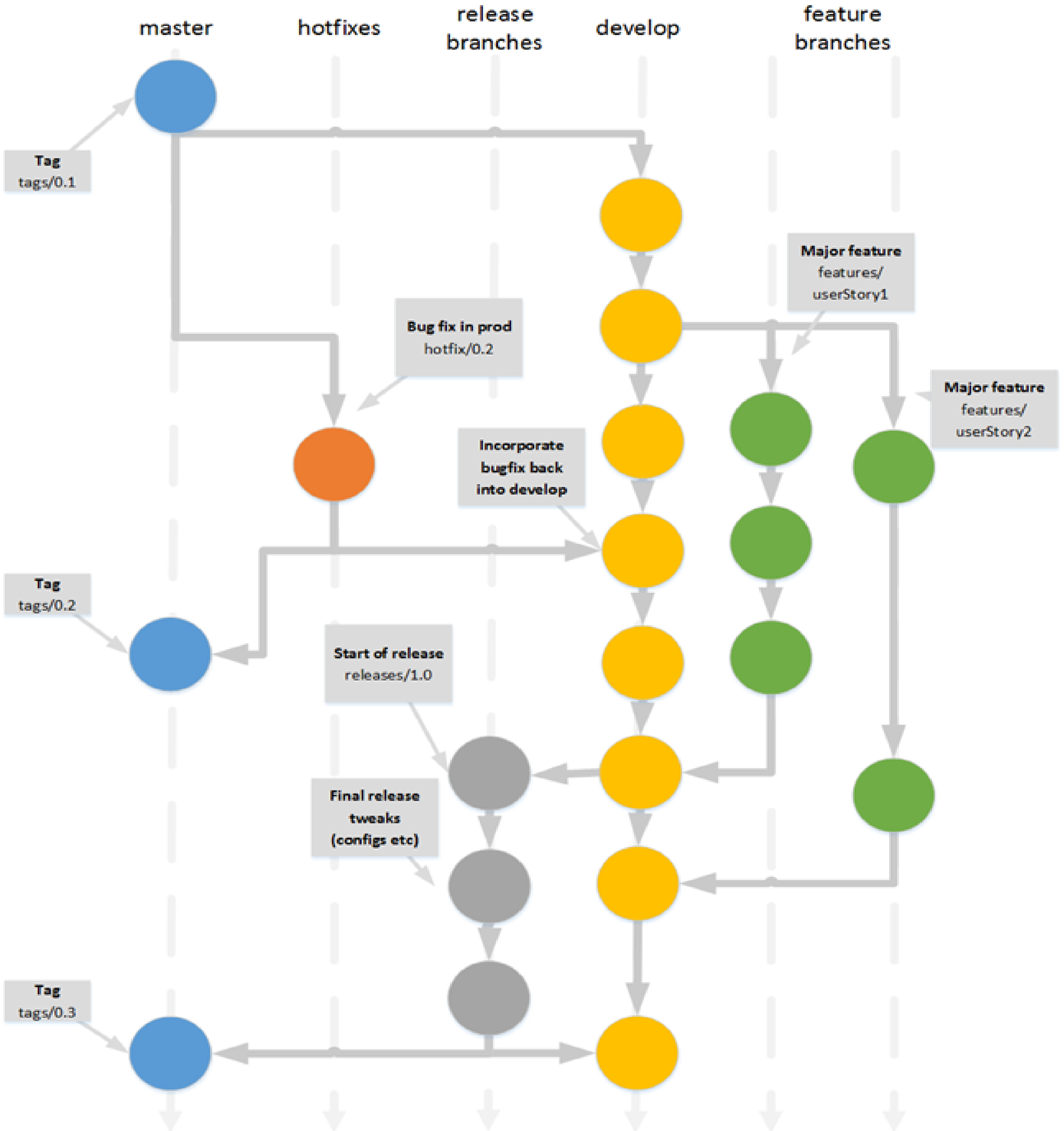
master

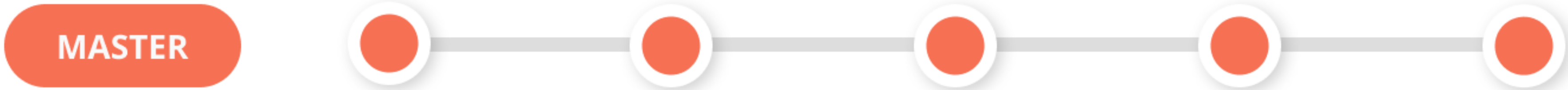
develop

feature branches

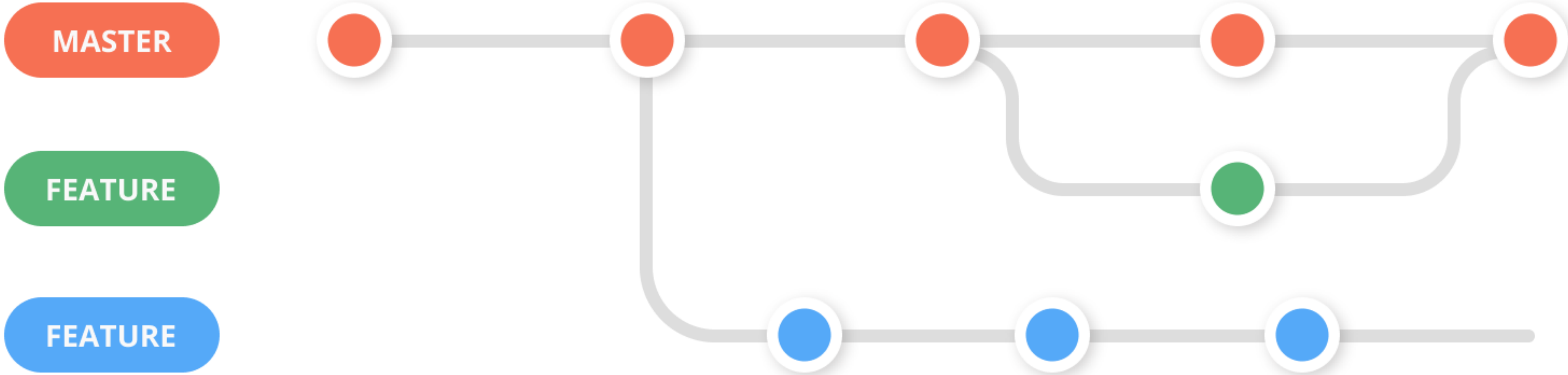
Major feature
features/
userStory1

Major feature
features/
userStory2


master
hotfixes
release branches
develop
feature branches
Tag
tags/0.1
Major feature
features/
userStory1
Bug fix in prod
hotfix/0.2
Major feature
features/
userStory2
Incorporate bugfix back into develop
Tag
tags/0.2
Start of release
releases/1.0
Final release tweaks (configs etc)
Tag
tags/0.3



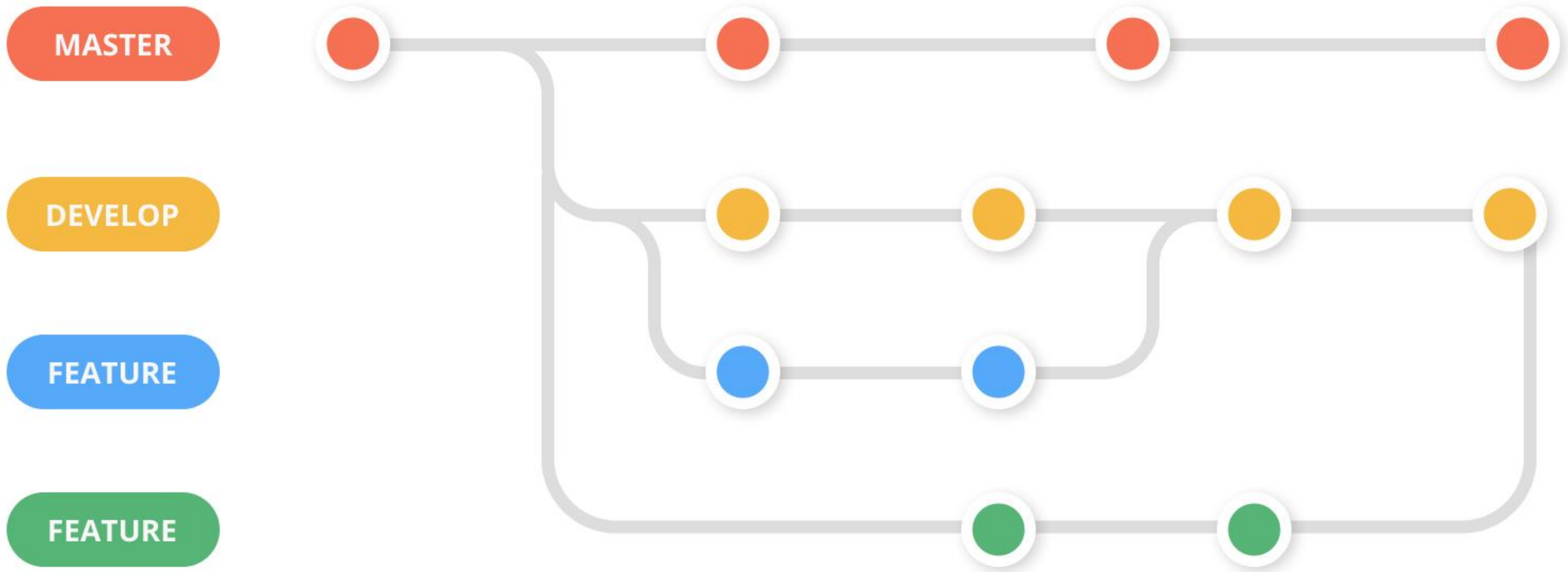
MASTER

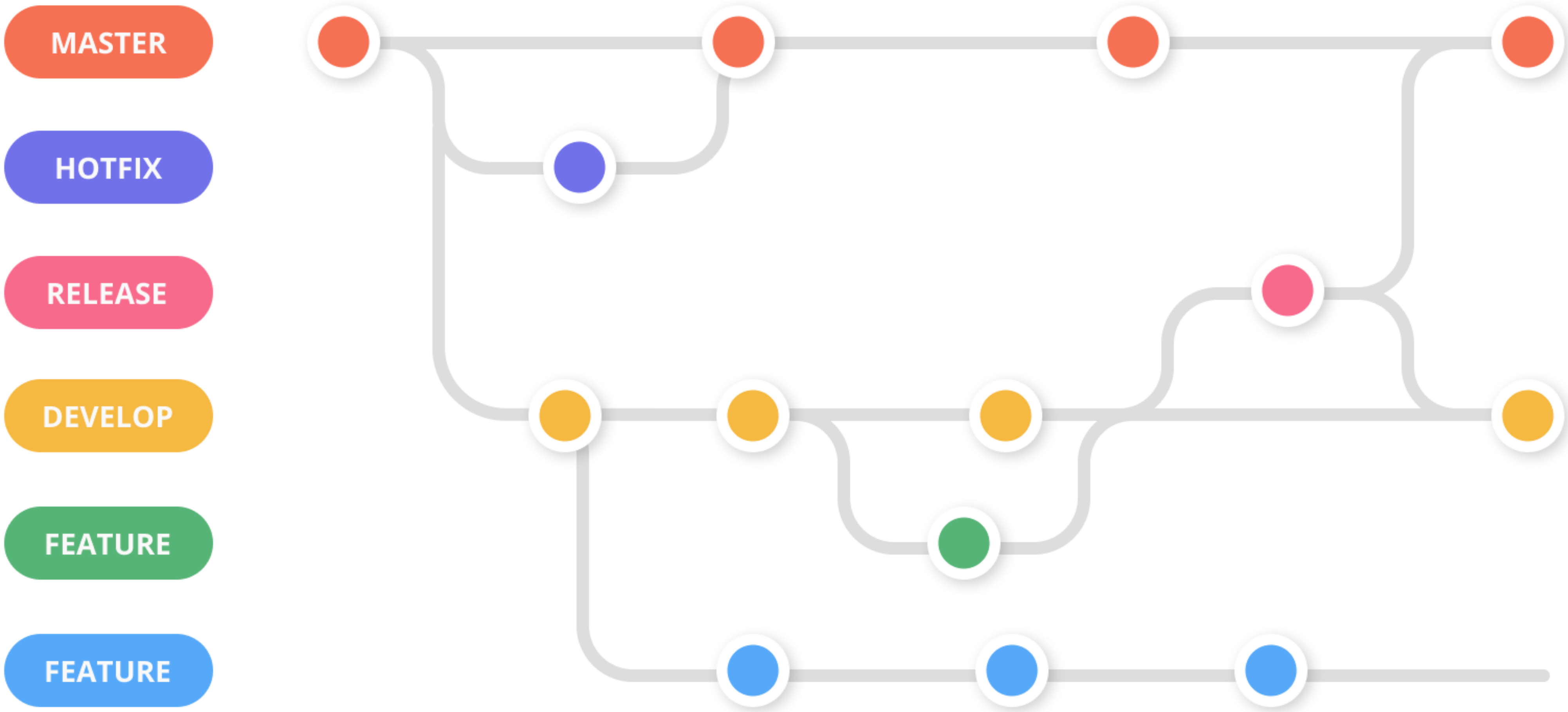


MASTER

FEATURE

FEATURE


MASTER
DEVELOP
FEATURE
FEATURE



MASTER

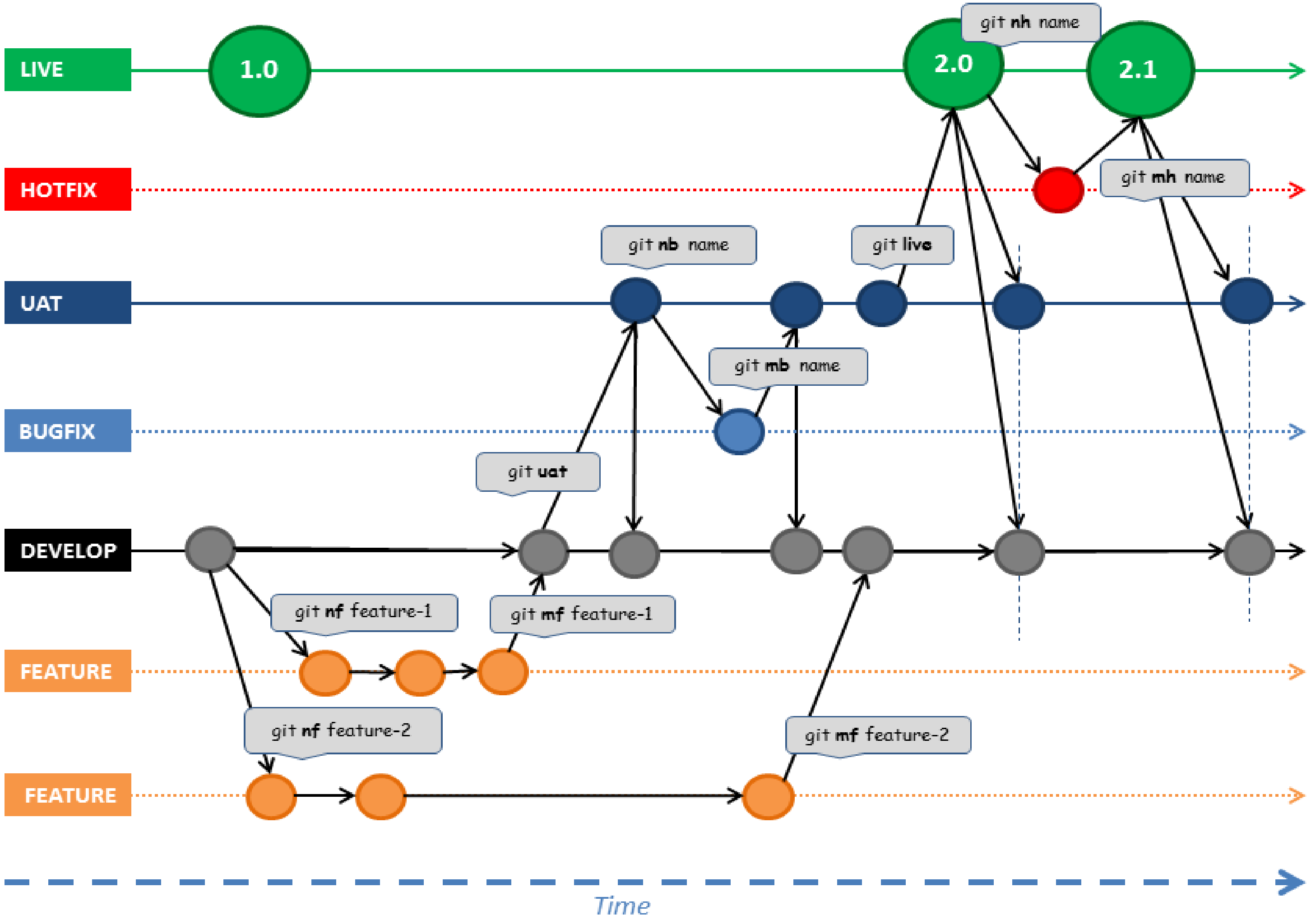
HOTFIX

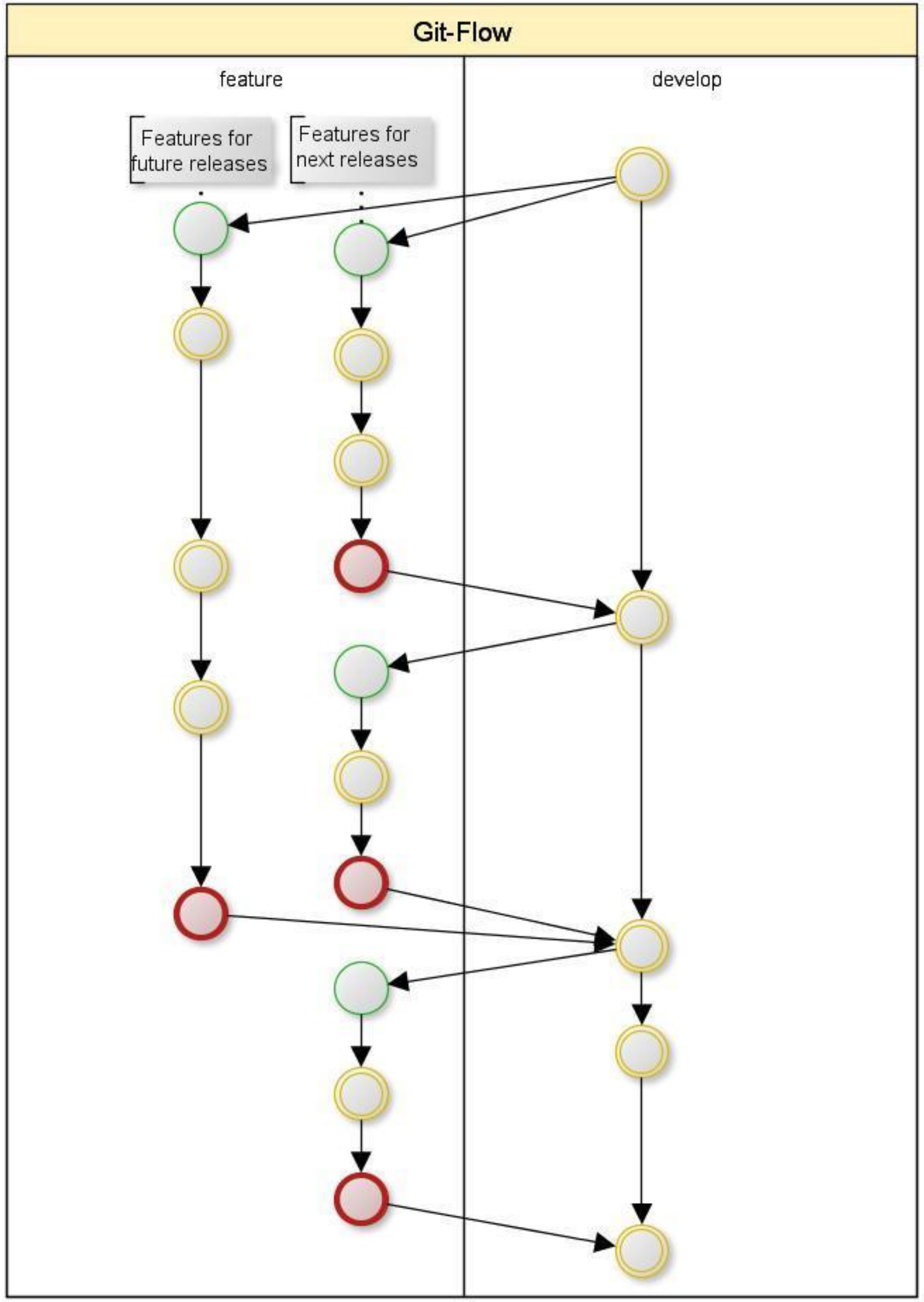
RELEASE

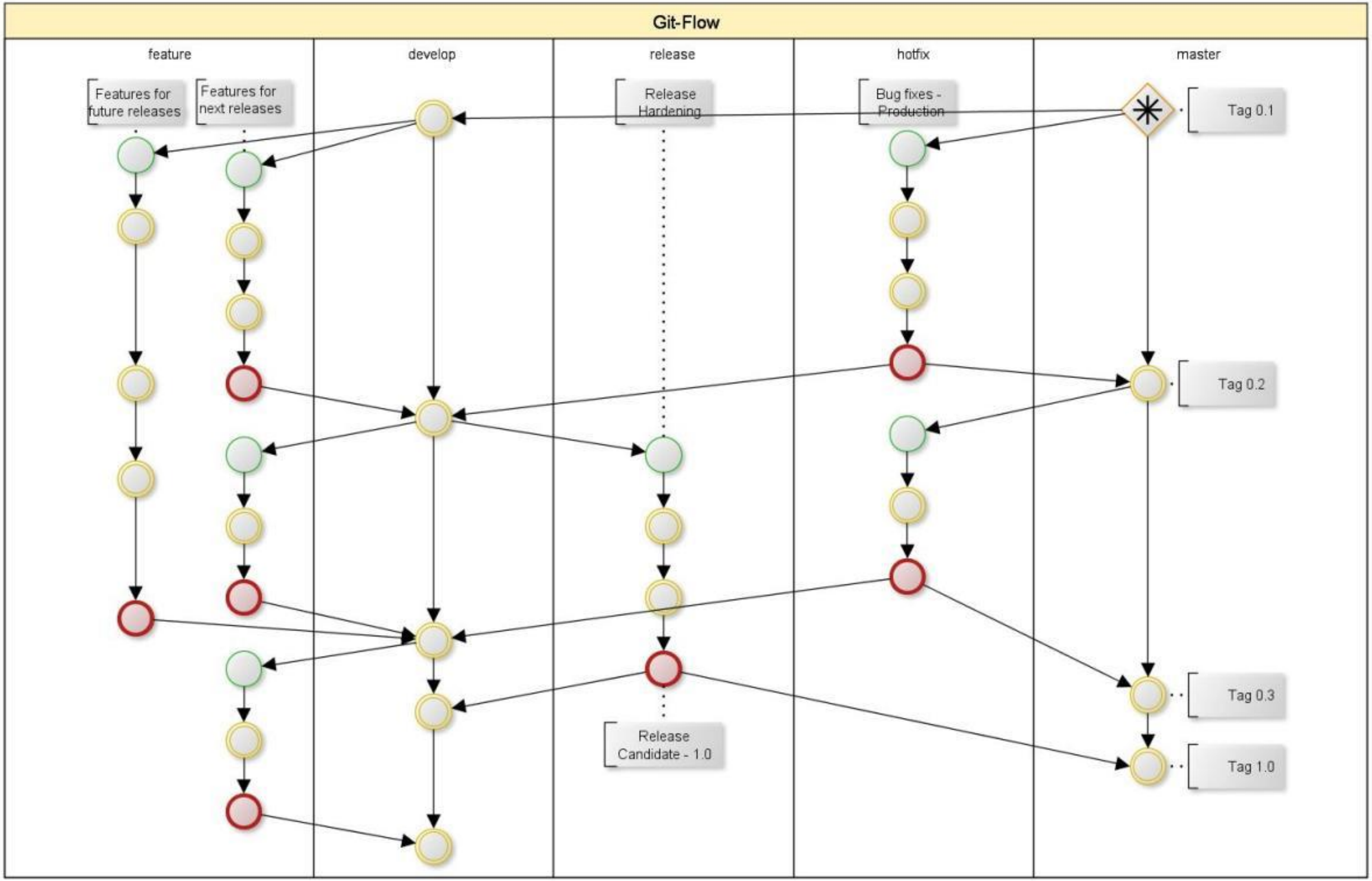
DEVELOP

FEATURE

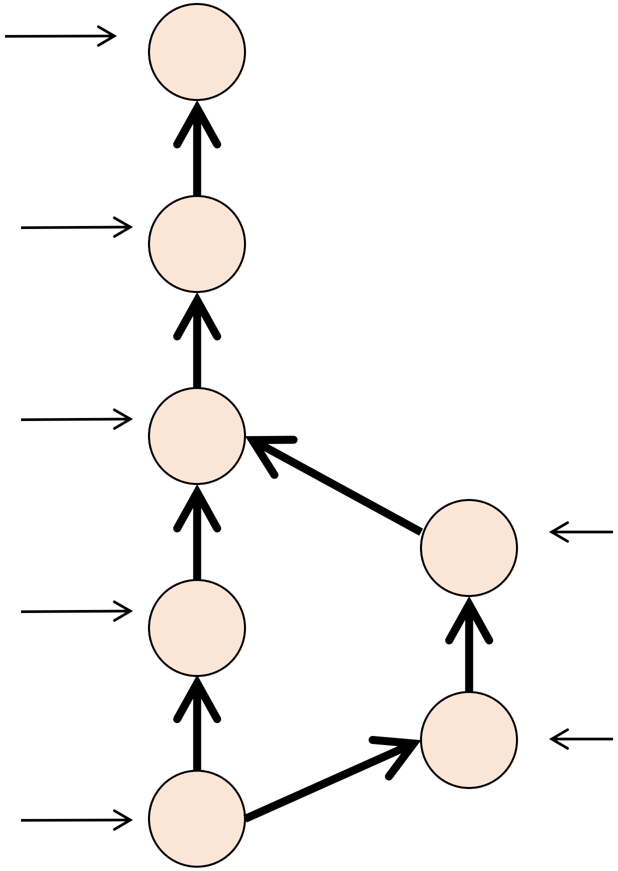
FEATURE


LIVE
1.0
2.0
2.1
git nh name
HOTFIX
git mh name
git nb name
git live
UAT
git mb name
BUGFIX
git uat
DEVELOP
git nf feature-1
git mf feature-1
FEATURE
git nf feature-2
git mf feature-2
FEATURE
Time


Git-Flow
feature
develop
Features for future releases
Features for next releases


Git-Flow
feature
develop
release
hotfix
master
Features for future releases
Features for next releases
Release Hardening
Bug fixes - Production
Tag 0.1
Tag 0.2
Tag 0.3
Tag 1.0
Release Candidate - 1.0

Multiple versions



Initial commit

Second commit

Third commit

Fourth commit

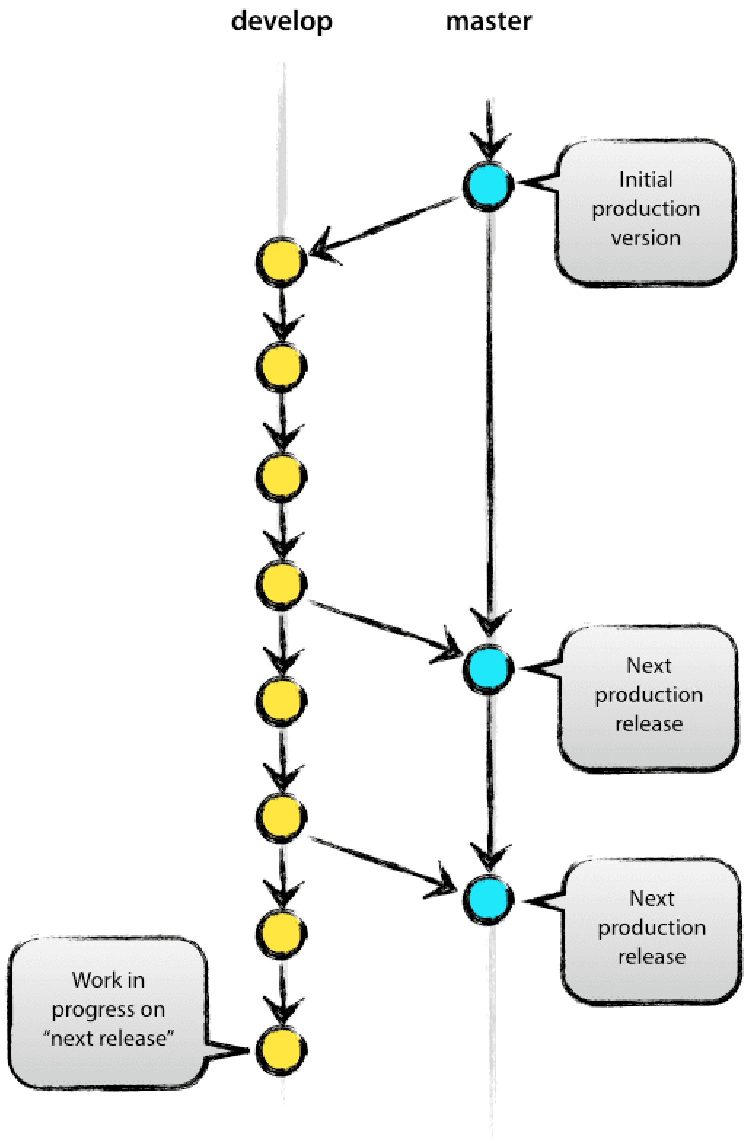
Merge

Bob gets a copy

Bob’s commit

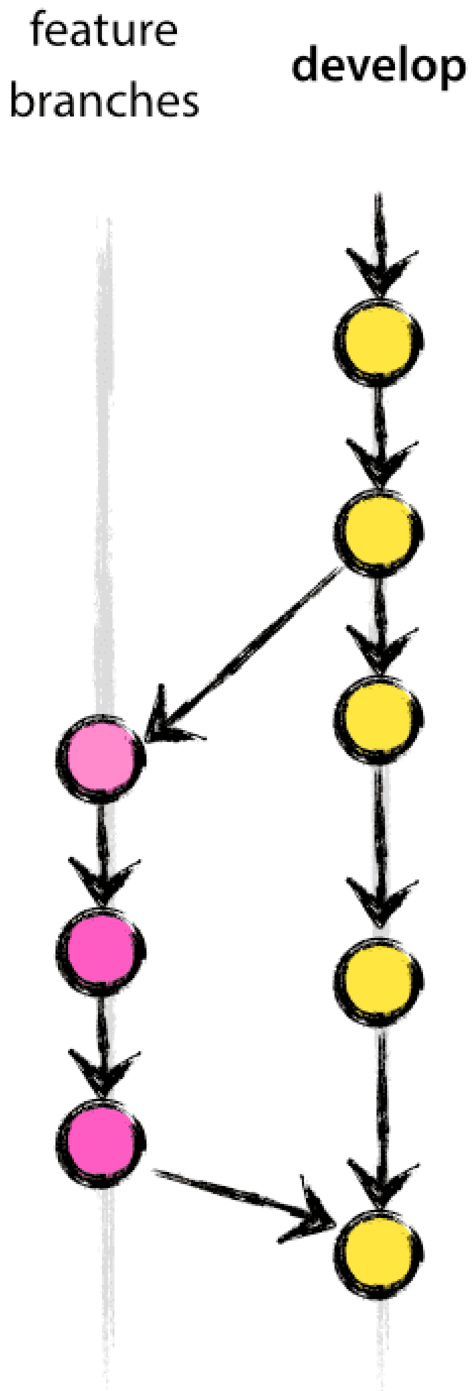
Master and Develop

- The two primary branches
- Master is the default branch in git
    - Use for stable releases
- Create develop branch
    - Use for untested code
    - Primary working branch for the team
    - Contains the latest features
- Close to a release
    - Test develop branch for stability
    - Merge commits in develop into master
    - Test and release master
    - Continue to code on develop through the release


develop
master
Initial production version
Next production release
Next production release
Work in progress on “next release”

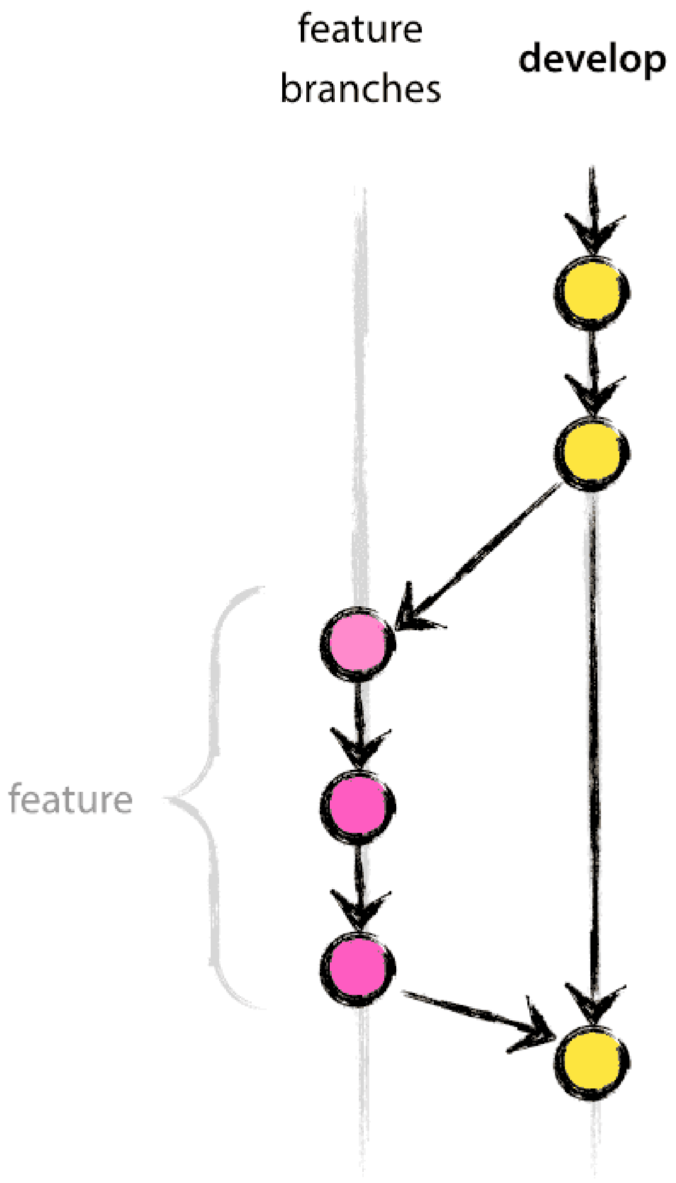
Feature Branch

- Branch off of develop
- Primary working branches for an individual(s)
- When new feature is finished
  - Merge into develop
  - Code reviews (If applicable)
  - Hope it doesn’t break develop (it might)
    - nightly builds to find out
- Temporary branches
- Can be many feature branches being developed in parallel
- If the feature is a failure
  - Delete the branch without merging into develop

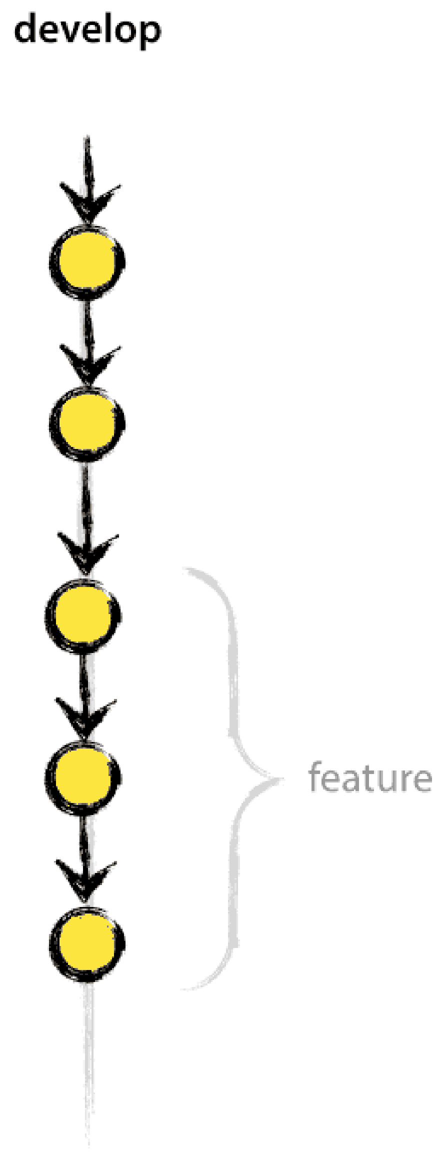

feature branches
develop

Merging

- To retain branch information
  - git merge --no-ff
  - no fast-forward
- With fast-forward
  - Existence of the branch is lost
  - Without meaningfully tagging commit messages
    - looking through history is confusing
    - Especially with many features developed in parallel



`git merge --no-ff`

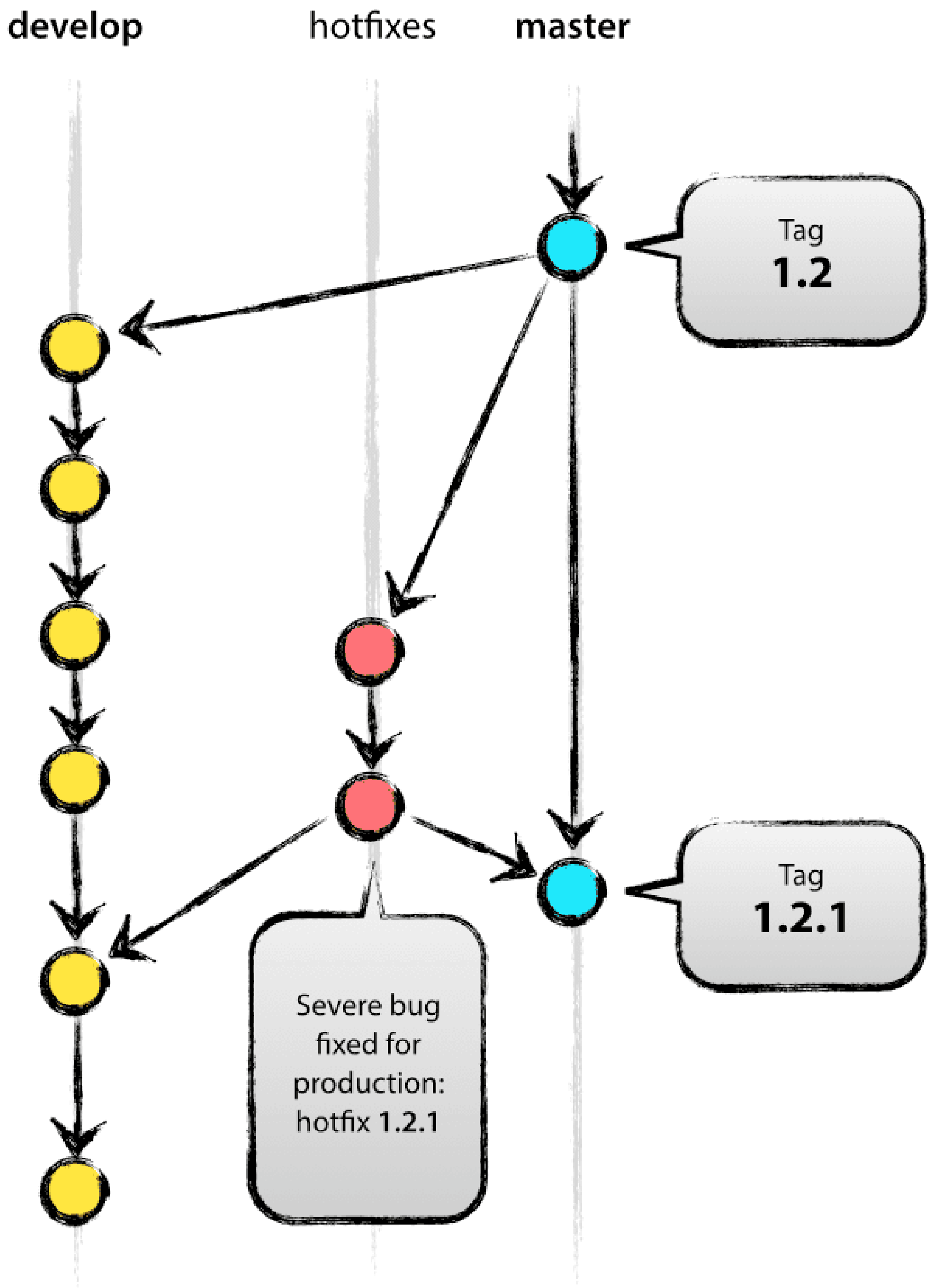


`git merge`

(plain)

Hotfixes

- Production release contains a bug
  - Create a hotfix branch from master
  - Fix the bug
  - merge fix into:
    - Next production release
    - Develop

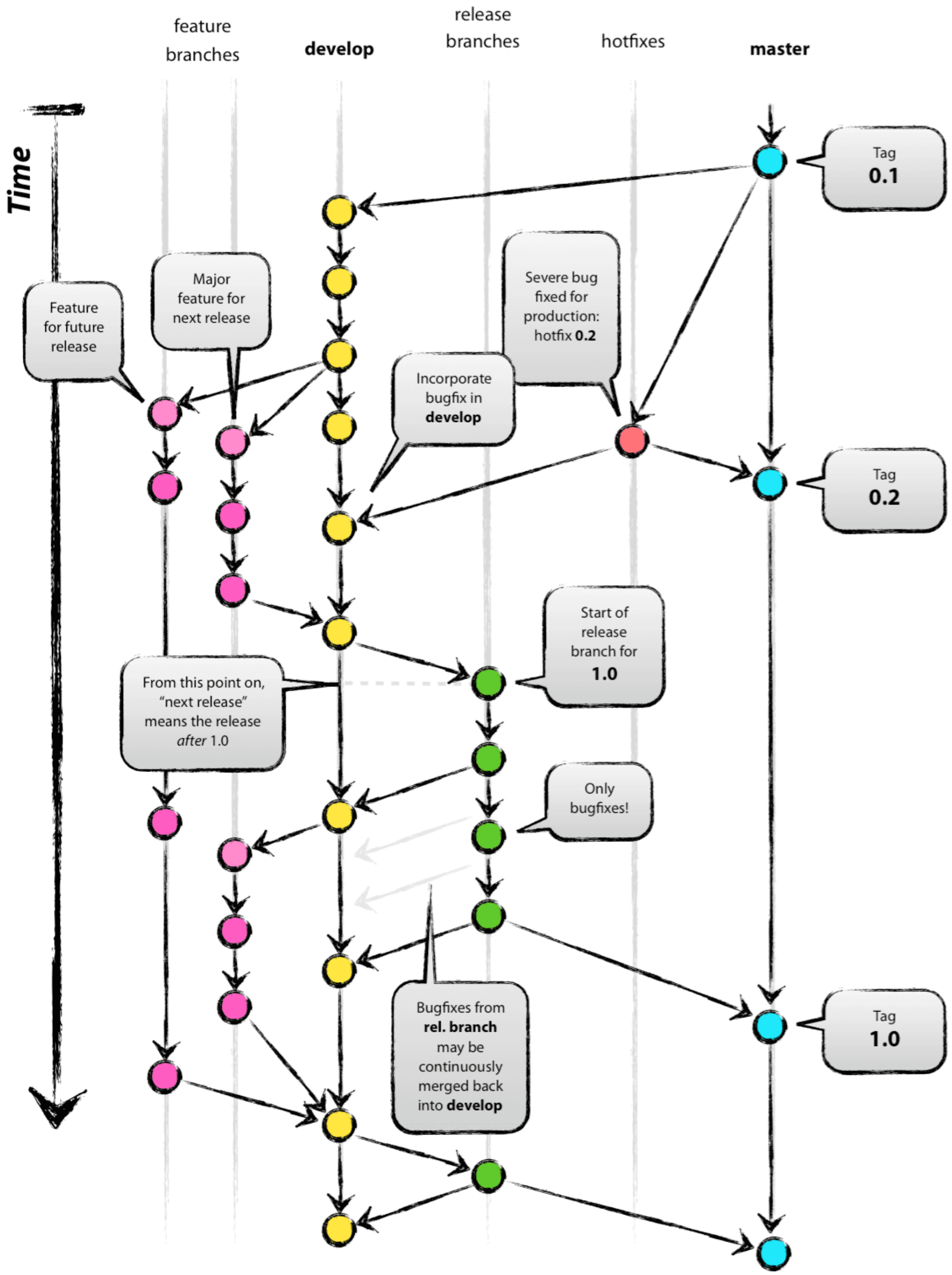

develop
hotfixes
master
Tag
1.2
Tag
1.2.1
Severe bug fixed for production: hotfix 1.2.1

Release Branch

- Branch off develop when approaching a release
- No features added
- Extensive testing and bug fixes
  - Merge all changes back to develop
- When confidently stable
  - Merge into master as a release

Versioning

- Tag the current state of the code
- Can easily work with a previous version if needed
- Use versioning on master



Feature for future release

Major feature for next release

Severe bug fixed for production: hotfix **0.2**

Incorporate bugfix in **develop**

Start of release branch for **1.0**

From this point on, "next release" means the release *after* 1.0

Only bugfixes!

Bugfixes from **rel. branch** may be continuously merged back into **develop**