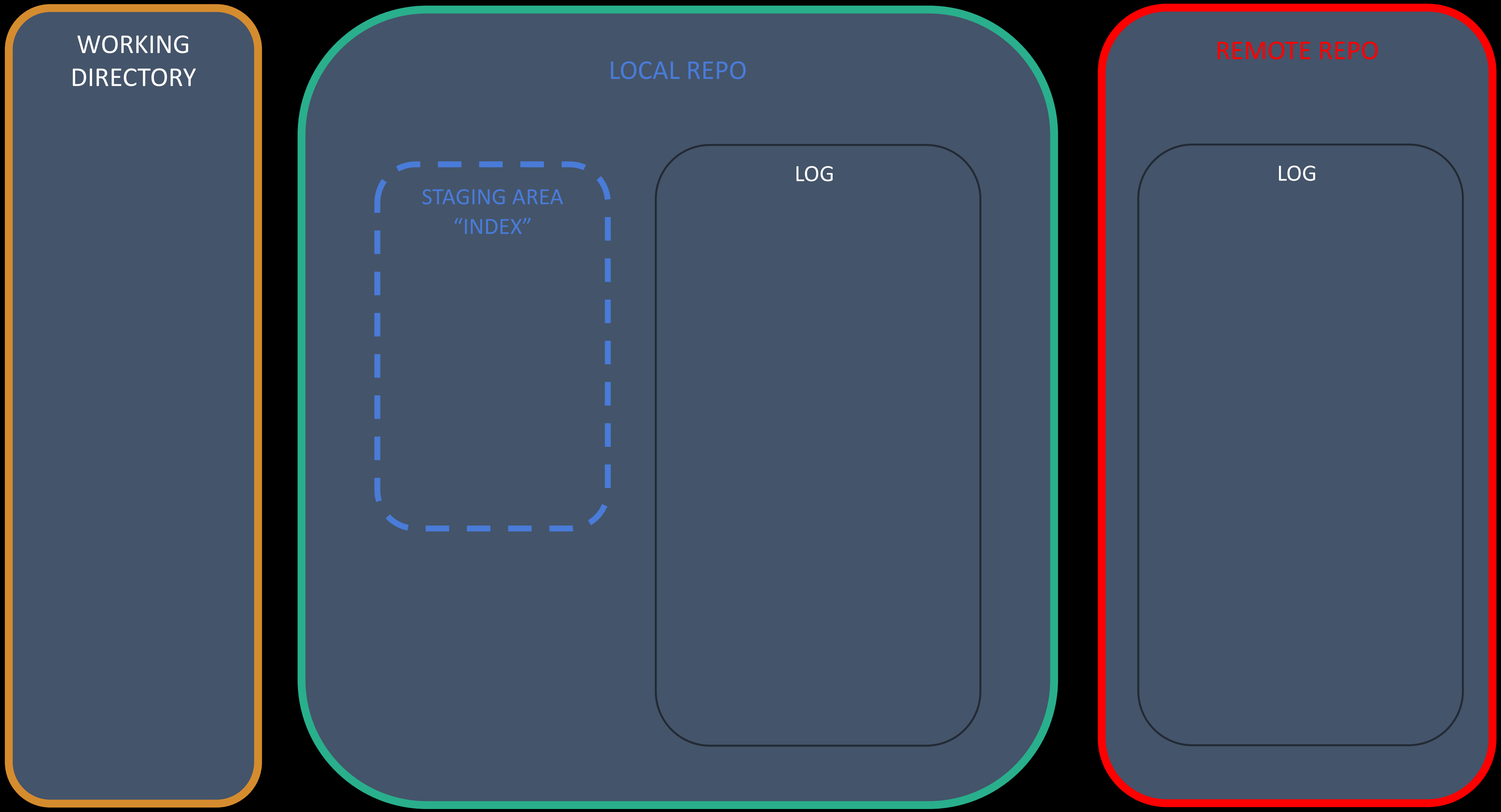
git Advanced

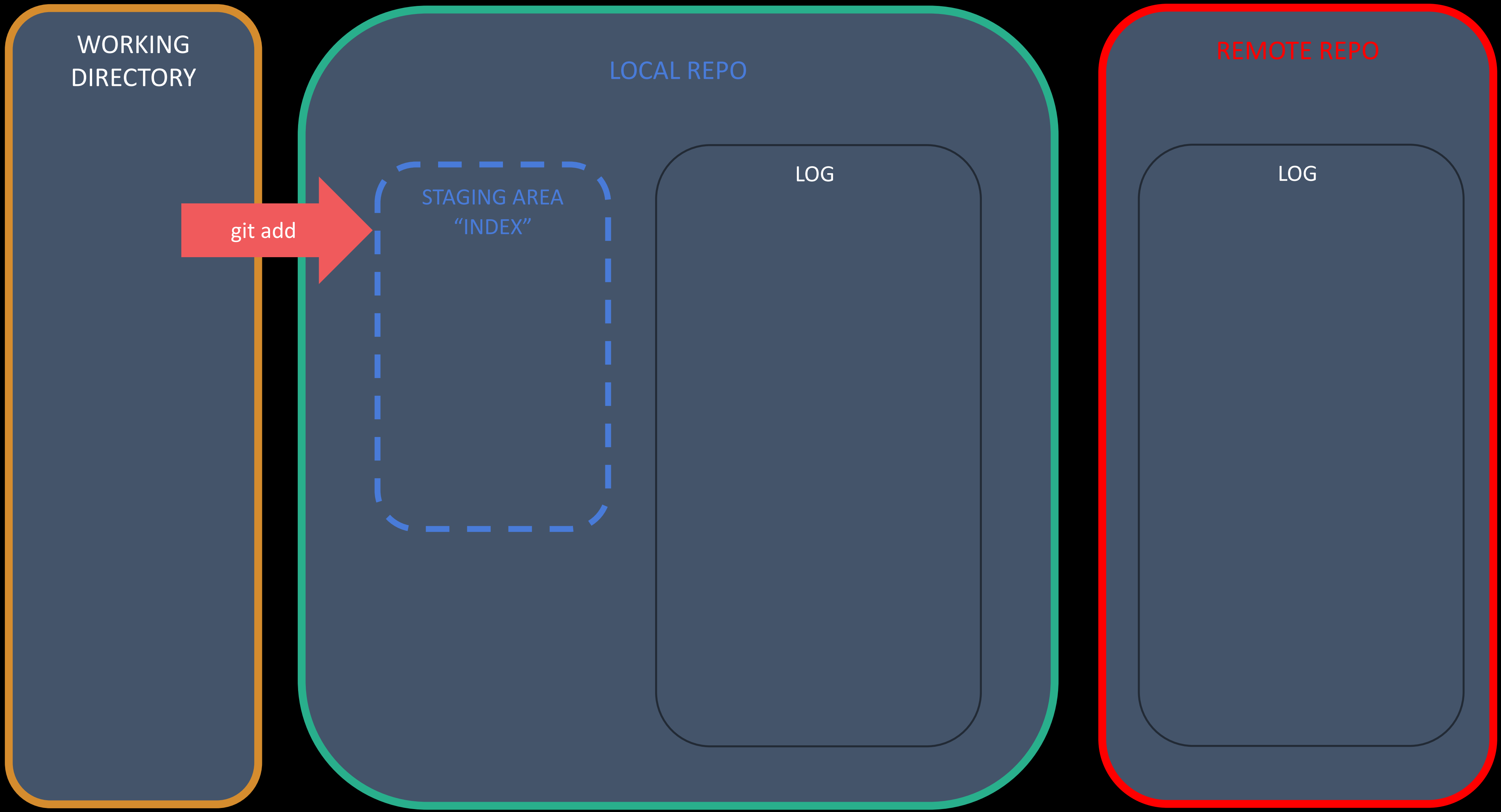
Ondřej Sucharda
sucharda@cngroup.dk

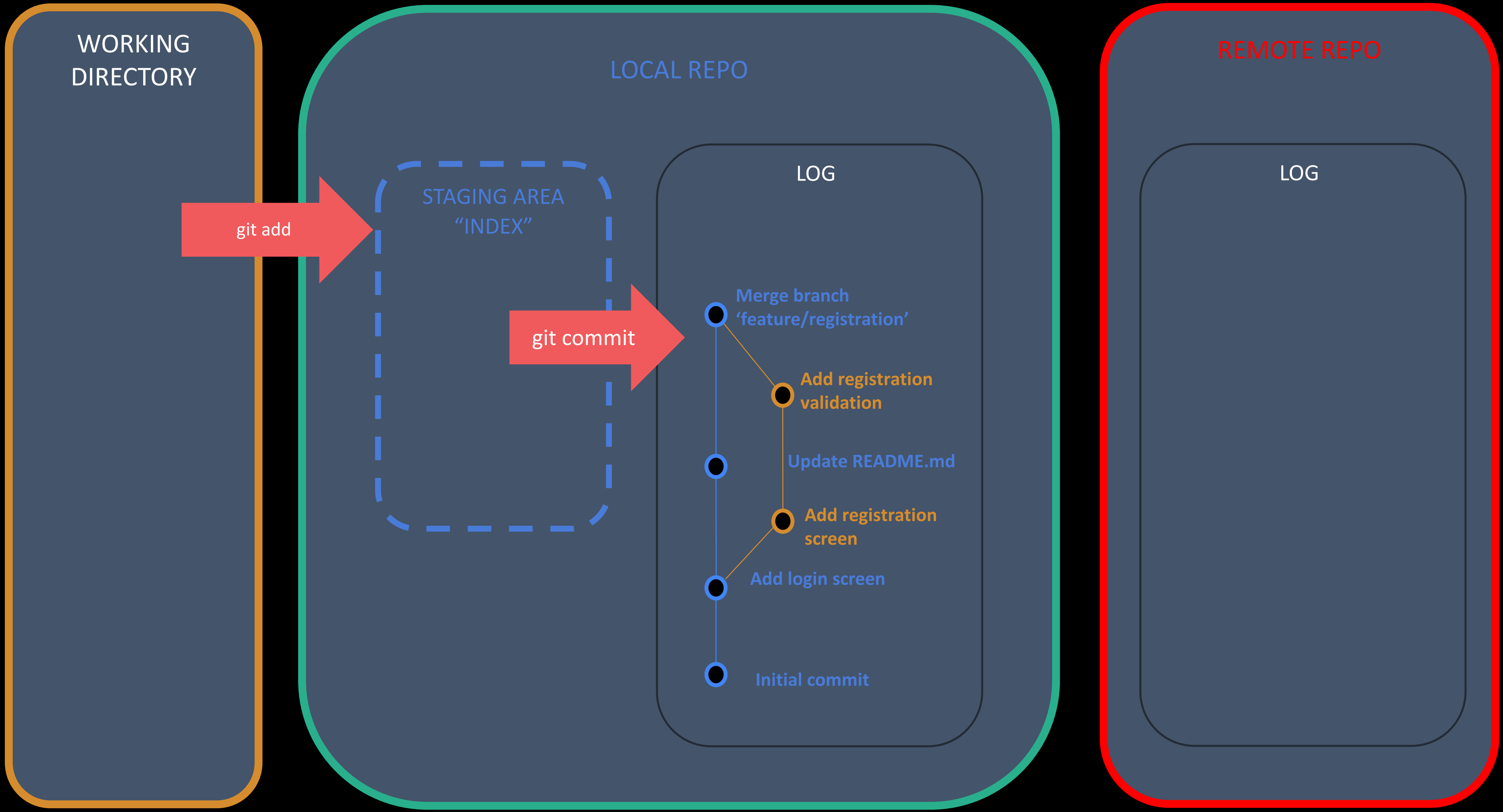
Agenda

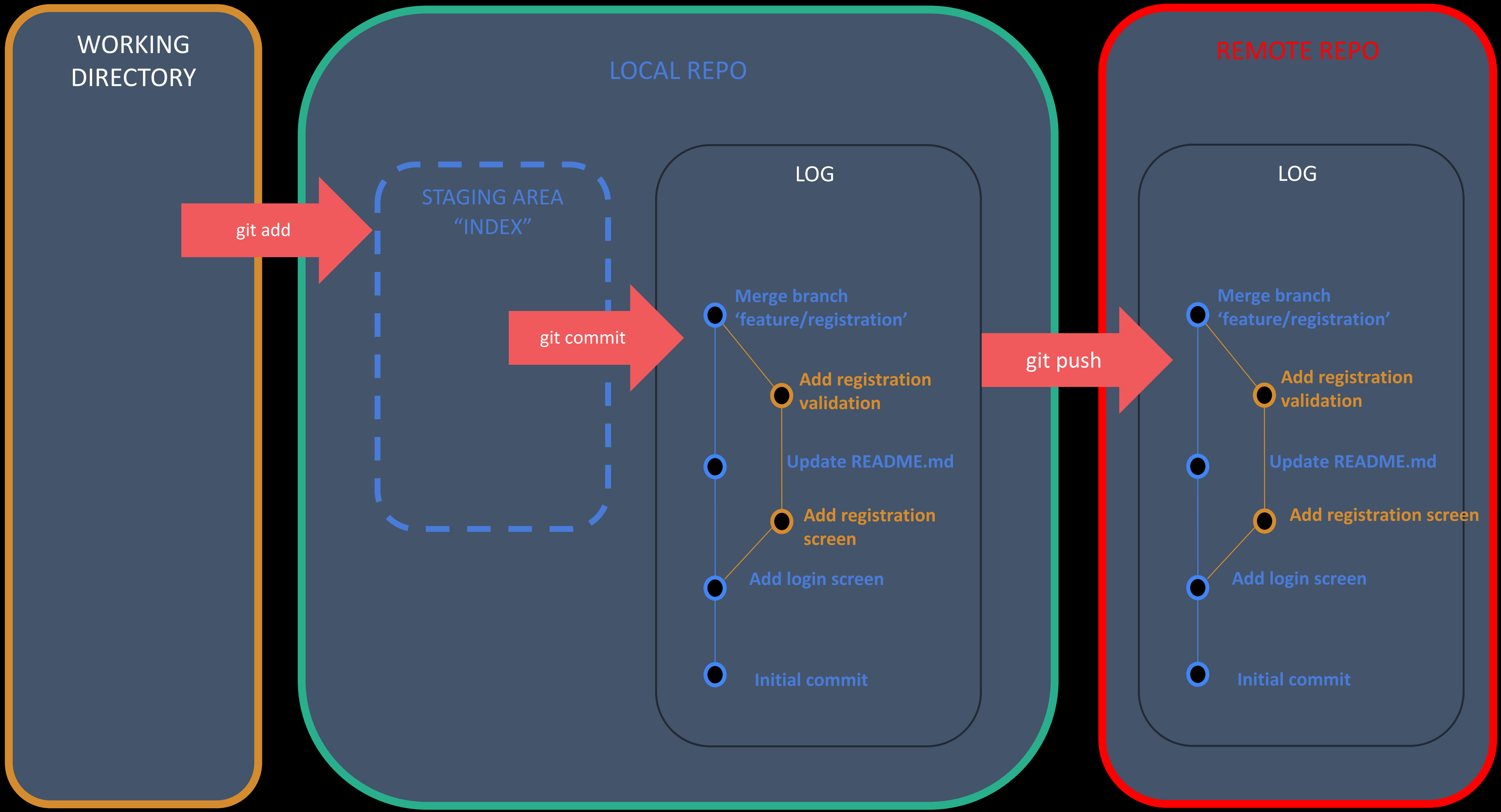
- Git Basics Recap
- Branches Strategies (Git Workflow)
- Pull Request
- Undo changes - Reset vs Revert
- Merge vs Rebase
- Interactive Rebase (Changing Git History)
- Merge conflicts

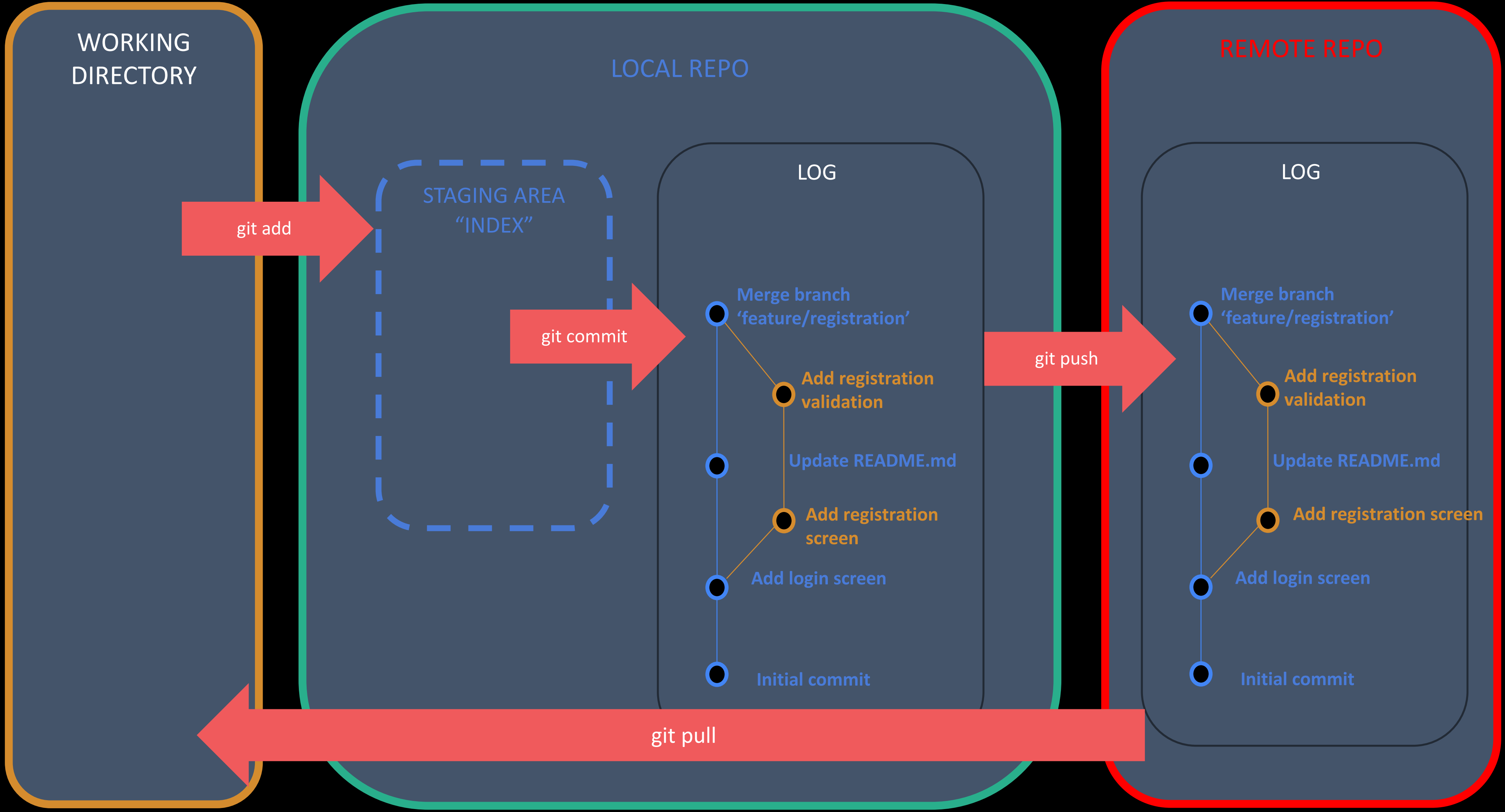
Git Basics Recap


WORKING DIRECTORY
LOCAL REPO
STAGING AREA “INDEX”
LOG
REMOTE REPO
LOG


WORKING DIRECTORY
LOCAL REPO
REMOTE REPO
git add
STAGING AREA
“INDEX”
LOG
LOG


WORKING DIRECTORY
LOCAL REPO
REMOTE REPO
git add
STAGING AREA "INDEX"
git commit
LOG
Merge branch 'feature/registration'
Add registration validation
Update README.md
Add registration screen
Add login screen
Initial commit
LOG


WORKING DIRECTORY
LOCAL REPO
STAGING AREA “INDEX”
git add
git commit
LOG
Merge branch ‘feature/registration’
Add registration validation
Update README.md
Add registration screen
Add login screen
Initial commit
git push
REMOTE REPO
LOG
Merge branch ‘feature/registration’
Add registration validation
Update README.md
Add registration screen
Add login screen
Initial commit


WORKING DIRECTORY
LOCAL REPO
REMOTE REPO
STAGING AREA
"INDEX"
LOG
LOG
git add
git commit
git push
git pull
Merge branch 'feature/registration'
Add registration validation
Update README.md
Add registration screen
Add login screen
Initial commit
Merge branch 'feature/registration'
Add registration validation
Update README.md
Add registration screen
Add login screen
Initial commit

Commands you already know

Create repo

Create new local repo (from scratch)

```
git init
```

Download from an existing repo

```
git clone my_url
```

Observe repo

List new or modified files

```
git status
```

Show changes to files not yet staged

```
git diff
```

Show full change history

```
git log
```

Branches

List all local branches

```
git branch
```

Create a new branch

```
git branch my_branch
```

Switch to a branch

```
git checkout my_branch
```

Merge branch into current active branch

```
git merge my_branch
```

Make a change

Stages the file, ready for commit

```
git add [file]
```

Staged all changed files, ready for commit

```
git add .
```

Commit all staged files

```
git commit –m “message”
```

Synchronize

Get the latest changes from origin (no merge)

```
git fetch
```

Fetch the latest changes from origin and merge

```
git pull
```

Push local changes to the origin

```
git push
```

Git Advanced

Let’s learn something new

Branching Strategies

a.k.a. “Git Workflow”

A Written Convention

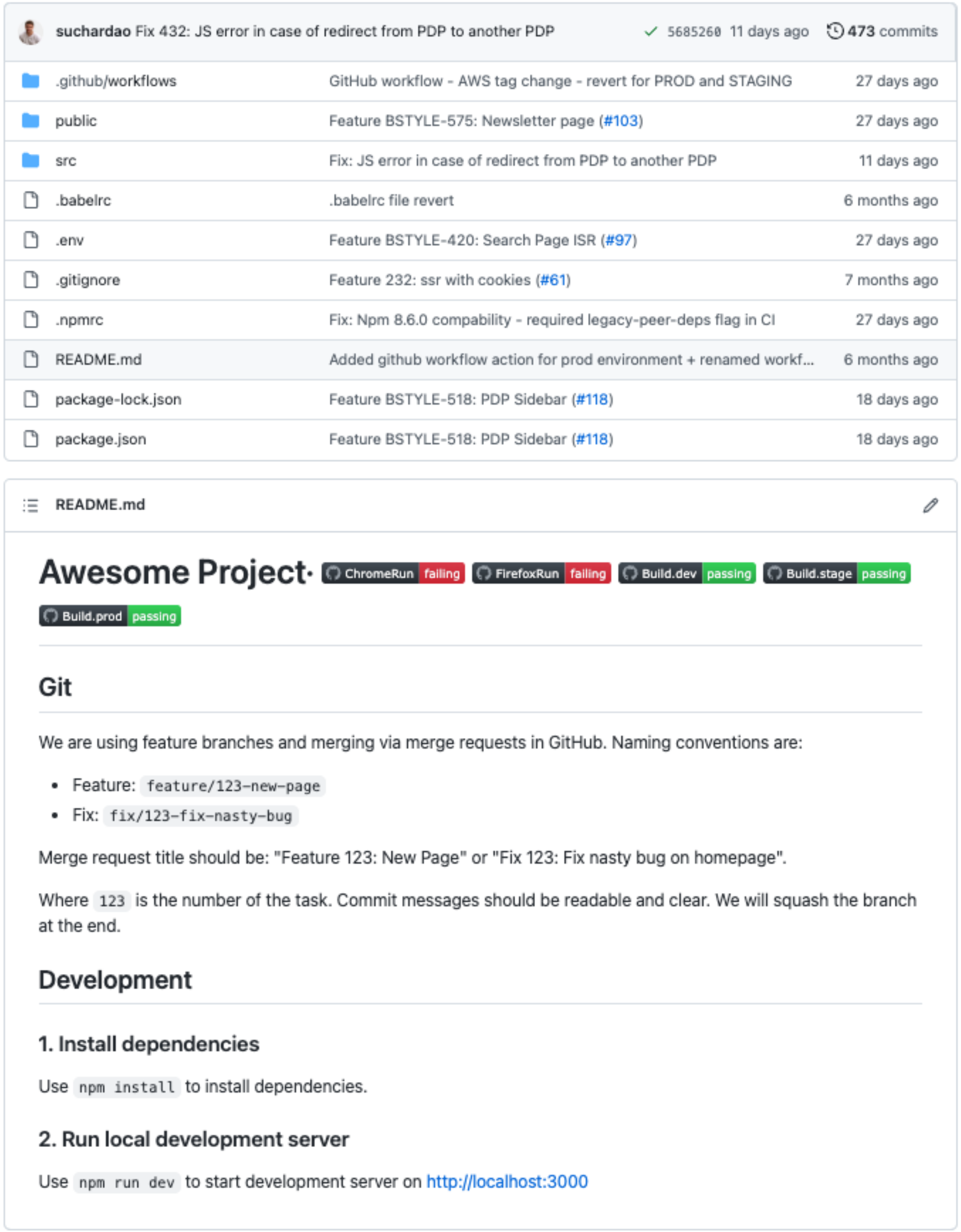
Agree on Branching Workflow in Your Team

- Git allows you to create branches – but it doesn't tell you how to use them!
- You need a written conventions – to avoid mistakes & collisions.
- It depends on your team / team size, on your project, and how you handle releases.
- It helps to onboard new team members ("this is how we work here").

Where to place conventions ?

For example README.md

- Is versioned with the code.
- Every repository manager (like GitHub, GitLab, …) can display it.



suchardao Fix 432: JS error in case of redirect from PDP to another PDP ✓ 5685260 11 days ago 473 commits

.github/workflows	GitHub workflow - AWS tag change - revert for PROD and STAGING	27 days ago
public	Feature BSTYLE-575: Newsletter page (#103)	27 days ago
src	Fix: JS error in case of redirect from PDP to another PDP	11 days ago
.babelrc	.babelrc file revert	6 months ago
.env	Feature BSTYLE-420: Search Page ISR (#97)	27 days ago
.gitignore	Feature 232: ssr with cookies (#61)	7 months ago
.npmrc	Fix: Npm 8.6.0 compatibility - required legacy-peer-deps flag in CI	27 days ago
README.md	Added github workflow action for prod environment + renamed workf...	6 months ago
package-lock.json	Feature BSTYLE-518: PDP Sidebar (#118)	18 days ago
package.json	Feature BSTYLE-518: PDP Sidebar (#118)	18 days ago

README.md

Awesome Project·

ChromeRun failing FirefoxRun failing Build.dev passing Build.stage passing Build.prod passing

Git

We are using feature branches and merging via merge requests in GitHub. Naming conventions are:

- Feature: `feature/123-new-page`
- Fix: `fix/123-fix-nasty-bug`

Merge request title should be: "Feature 123: New Page" or "Fix 123: Fix nasty bug on homepage".

Where `123` is the number of the task. Commit messages should be readable and clear. We will squash the branch at the end.

Development

1. Install dependencies

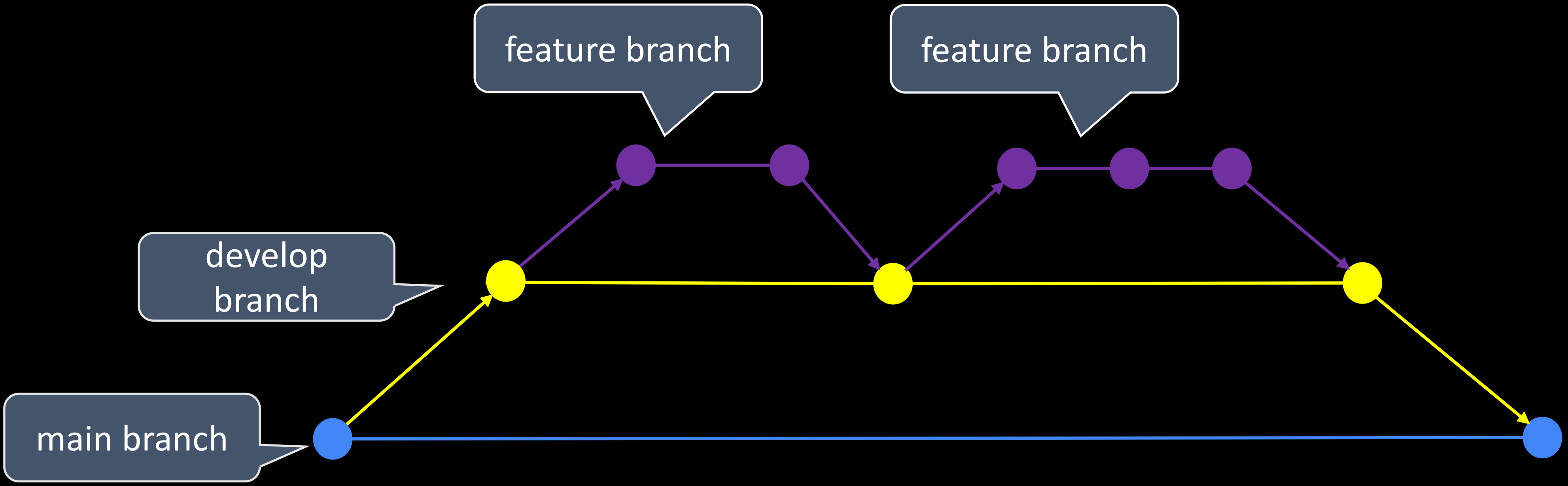
Use `npm install` to install dependencies.

2. Run local development server

Use `npm run dev` to start development server on http://localhost:3000

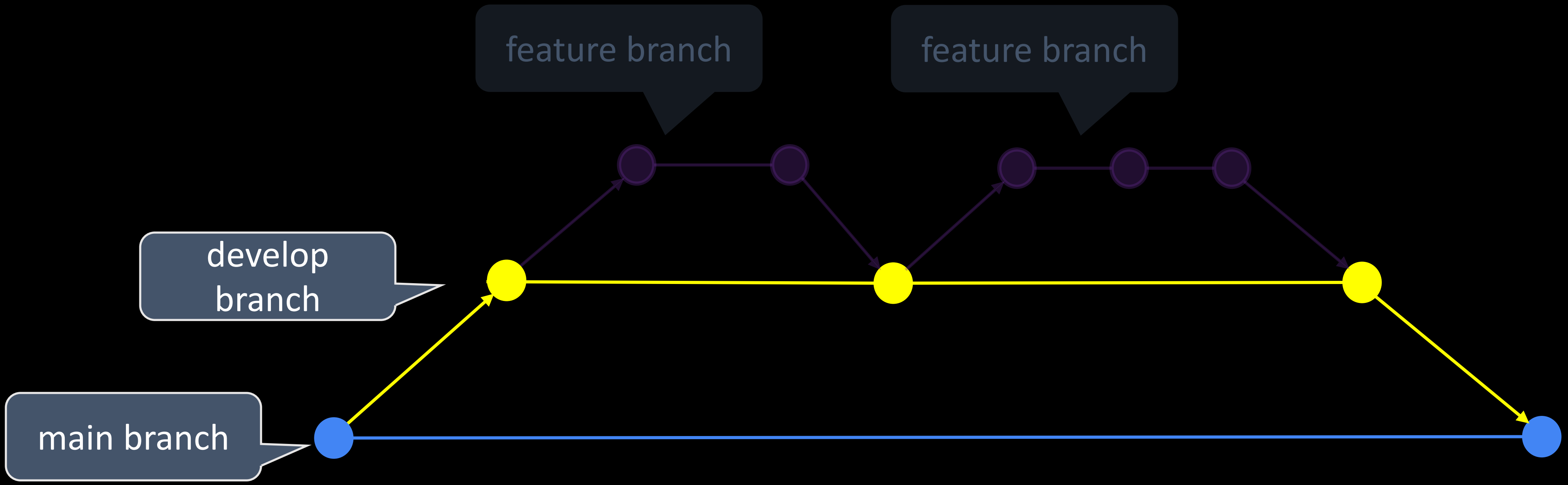
Long-Running & Short-Lived Branches

Two main types of branches



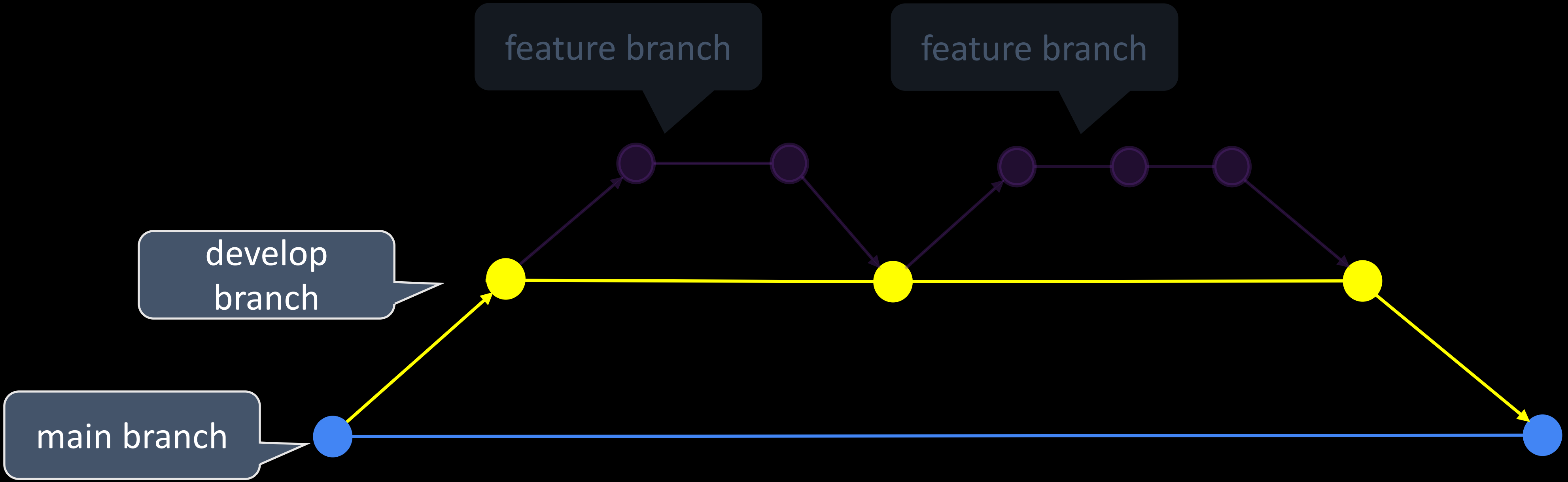
Long-Running & Short-Lived Branches

- Exists through the complete lifetime of the project
- Often, they mirror “stages” in your dev life cycle



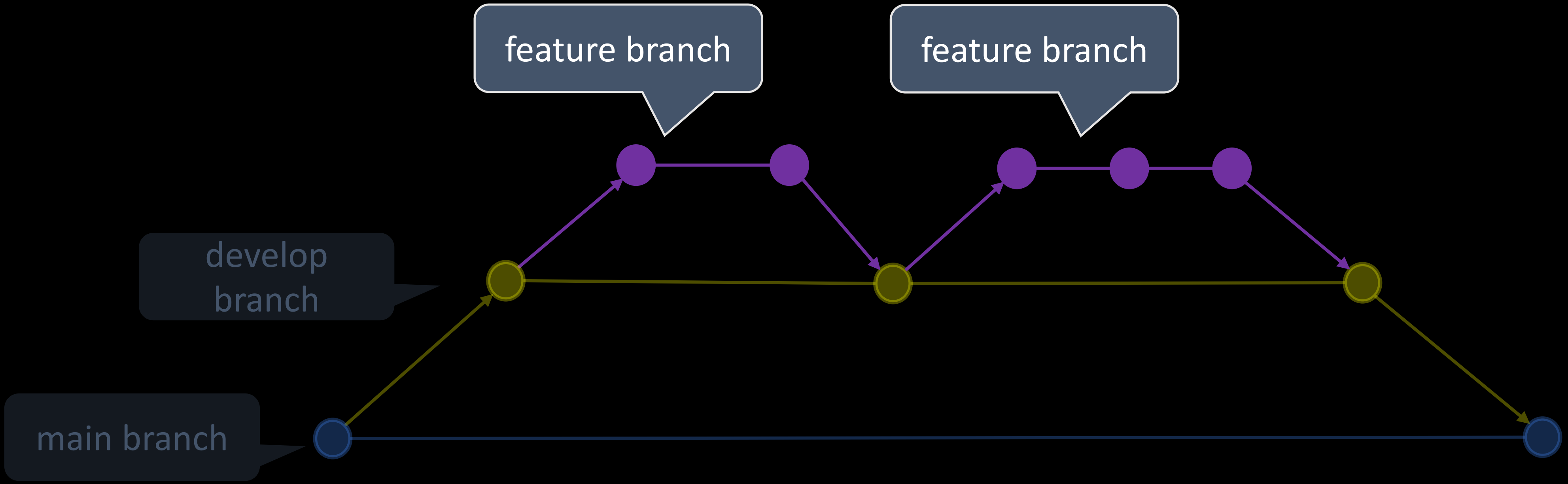
Long-Running & Short-Lived Branches

- Exists through the complete lifetime of the project
- Often, they mirror “stages” in your dev life cycle
- Common convention: no direct commits!



Long-Running & Short-Lived Branches

- For new features, bug fixes, refactorings, experiments, …
- Will be deleted after integration (merge/rebase)



Git Workflow

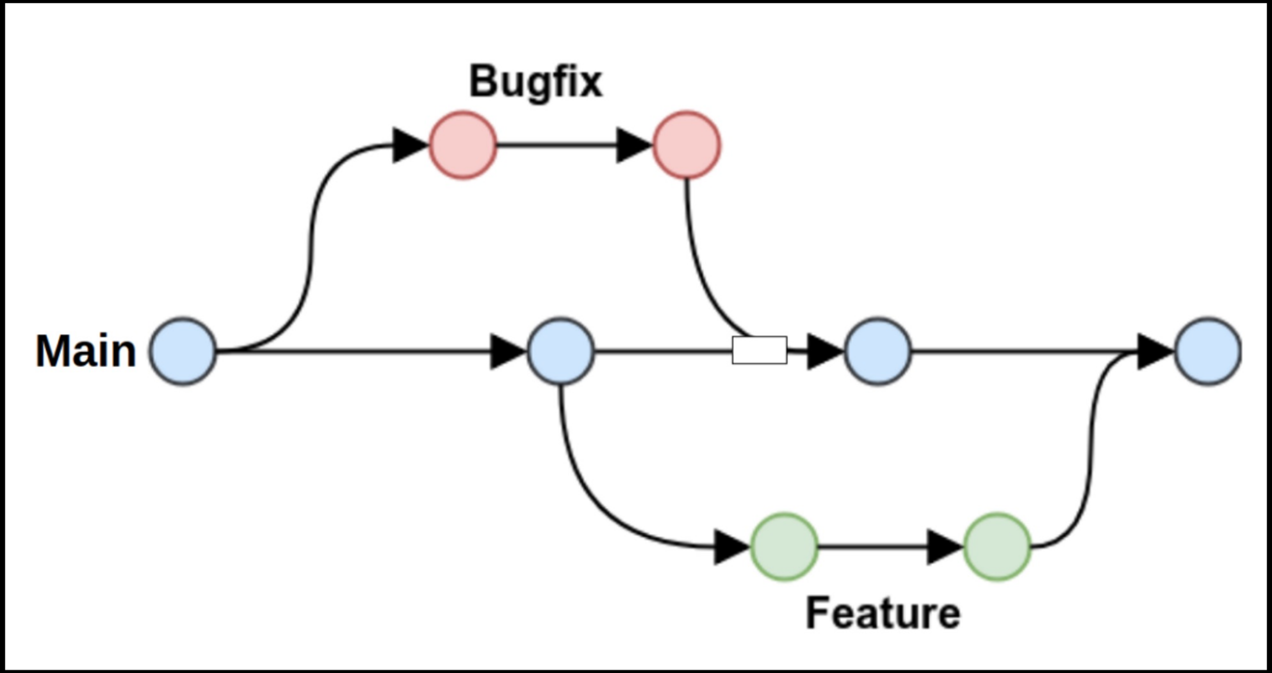
GitHub flow

VS

Git flow

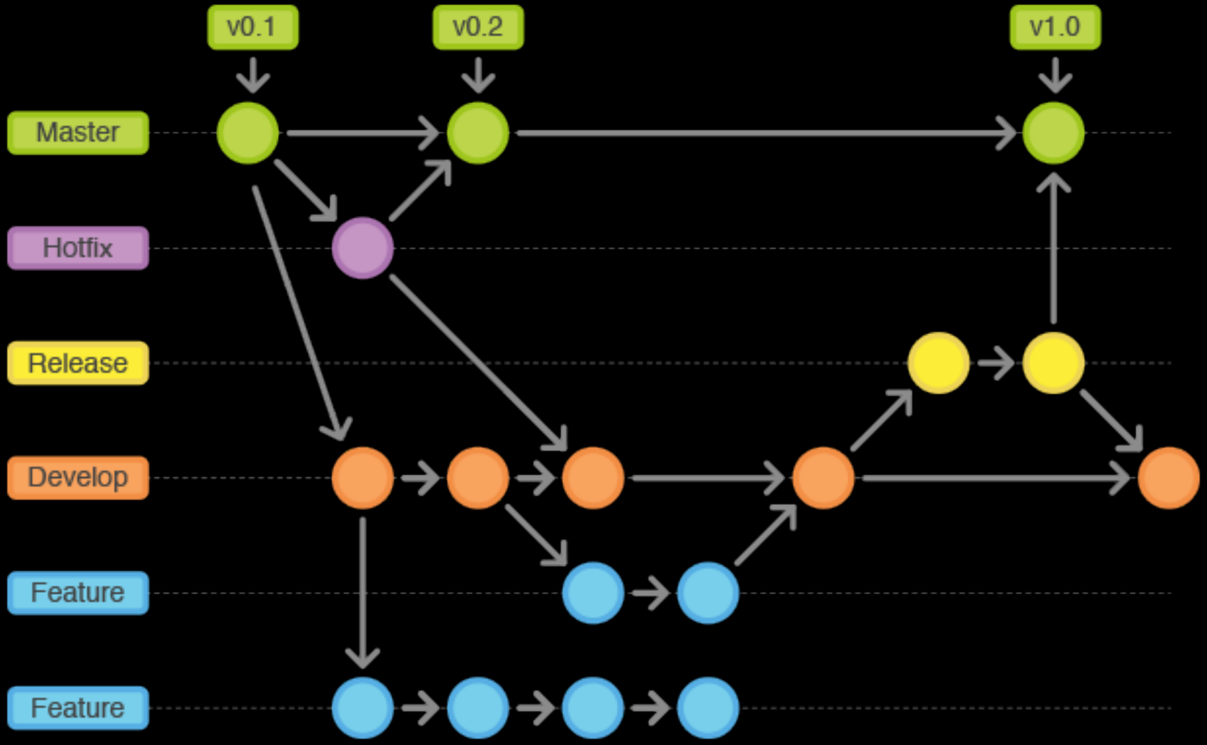
GitHub Flow

- Very simple: only one long-running branch (“main”) + feature branches



Git Flow

- More structure, more rules
- Long-Running: “main” + “develop”
- Short-Lived: features, releases, hotfixes

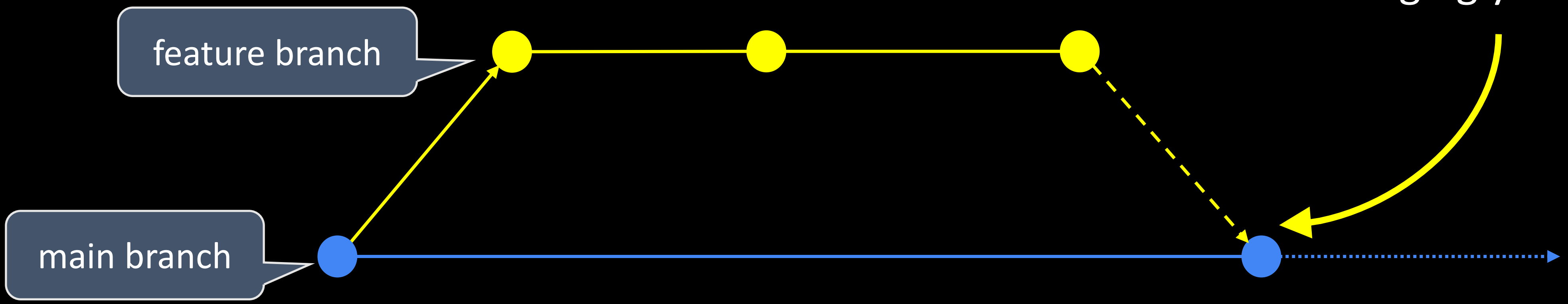


Pull Request

- Pull Request = Merge Request
- Provided by Git hosting platform like GitHub, GitLab (not git core feature)

Pull Request

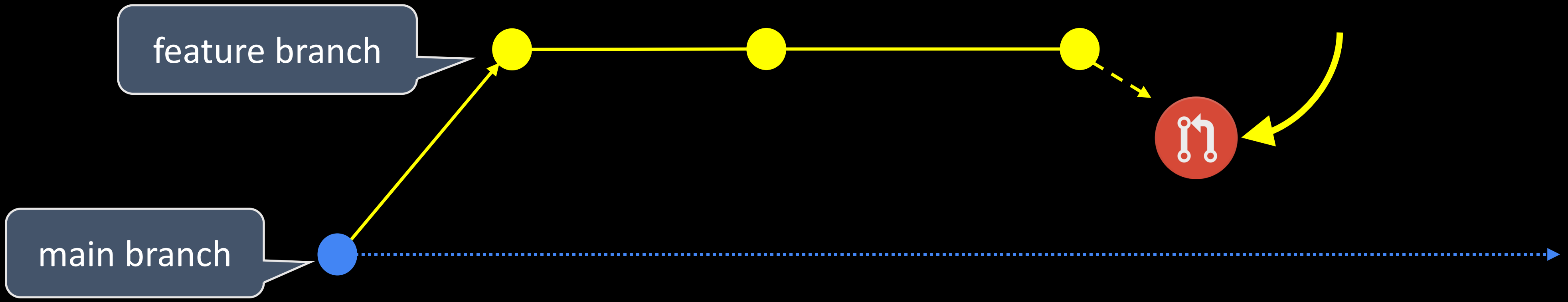
Communicating About and Reviewing Code



Without a Pull Request, you would jump right to merging your code…

Pull Request

Communicating About and Reviewing Code

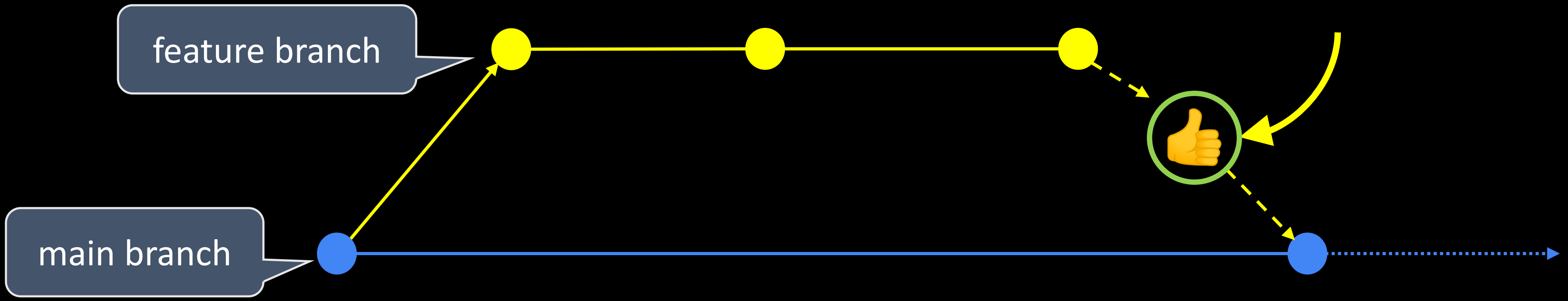


A Pull Request invites reviewers to provide feedback before merging.

Pull Request

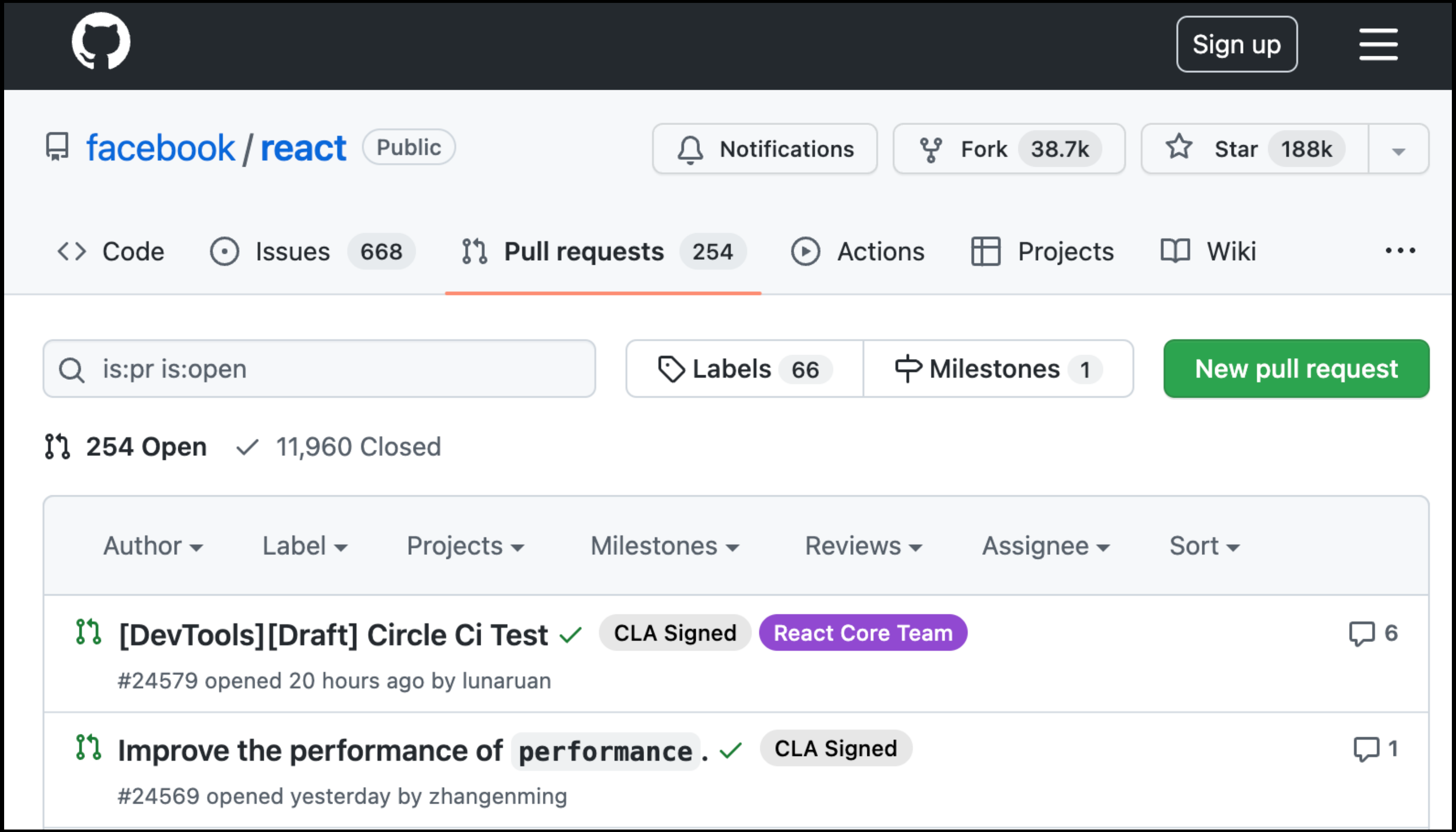
Communicating About and Reviewing Code

Your Pull Request can be merged after approval by the reviewer.



Pull Request

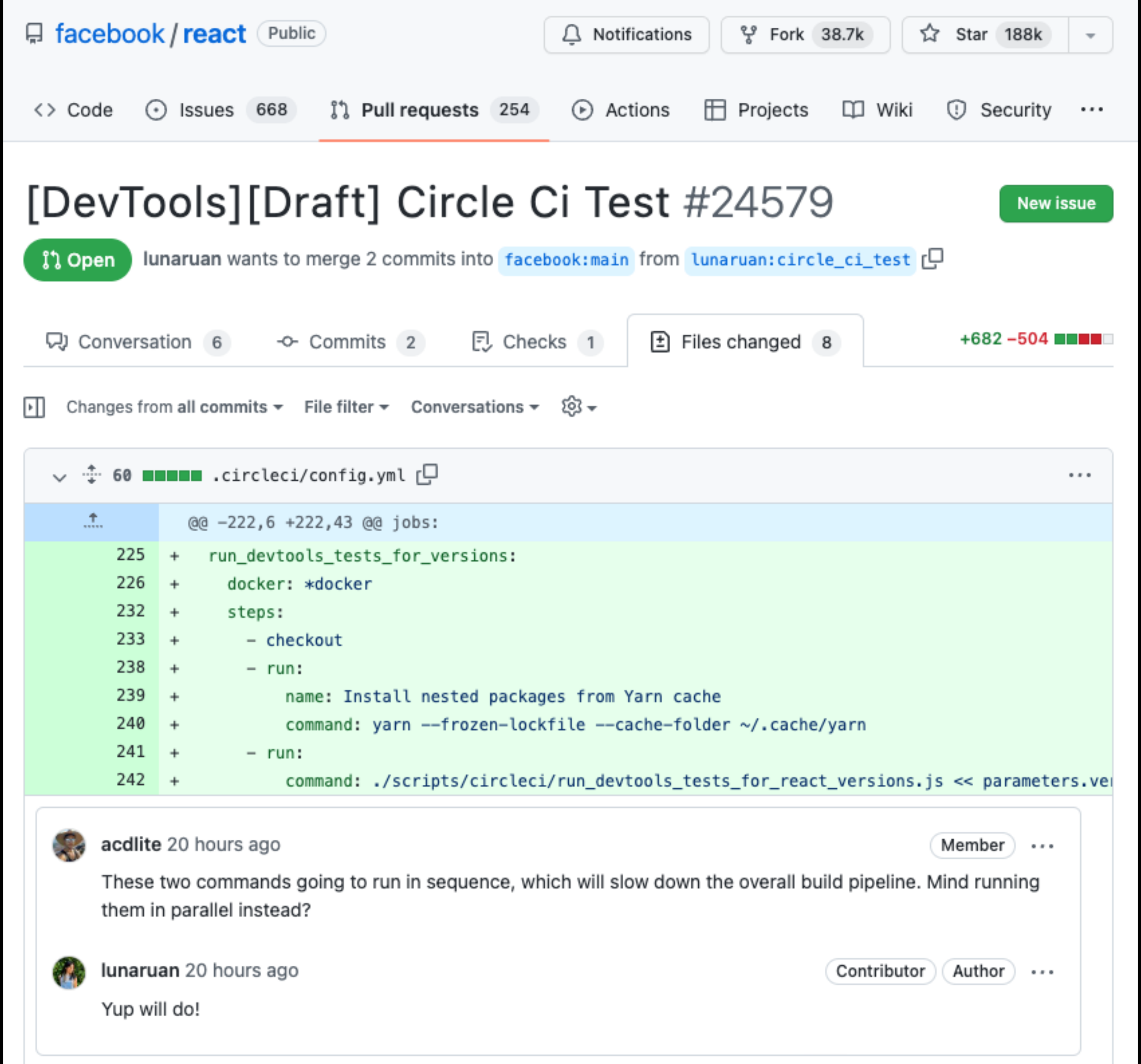
Example



254 Open ✓ 11,960 Closed

[DevTools][Draft] Circle Ci Test ✓ CLA Signed React Core Team 6
#24579 opened 20 hours ago by lunaruan

Improve the performance of `performance`. ✓ CLA Signed 1
#24569 opened yesterday by zhangenming



facebook / react Public

Notifications · Fork 38.7k · Star 188k

Code · Issues 668 · Pull requests 254 · Actions · Projects · Wiki · Security

[DevTools][Draft] Circle Ci Test #24579

Open — lunaruan wants to merge 2 commits into `facebook:main` from `lunaruan:circle_ci_test`

Conversation 6 · Commits 2 · Checks 1 · Files changed 8 · +682 −504

Changes from all commits · File filter · Conversations

60 .circleci/config.yml

```
@@ -222,6 +222,43 @@ jobs:
+  run_devtools_tests_for_versions:
+    docker: *docker
+    steps:
+      - checkout
+      - run:
+          name: Install nested packages from Yarn cache
+          command: yarn --frozen-lockfile --cache-folder ~/.cache/yarn
+      - run:
+          command: ./scripts/circleci/run_devtools_tests_for_react_versions.js << parameters.ve
```

acdlite 20 hours ago · Member

These two commands going to run in sequence, which will slow down the overall build pipeline. Mind running them in parallel instead?

lunaruan 20 hours ago · Contributor · Author

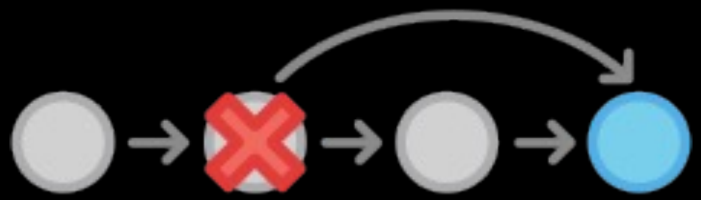
Yup will do!

Undo changes

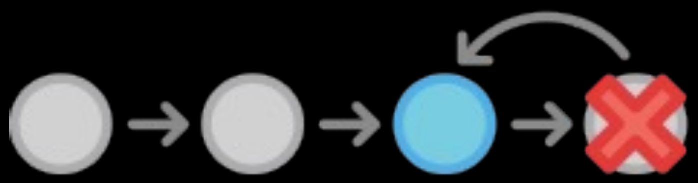
Revert vs Reset

Undo changes

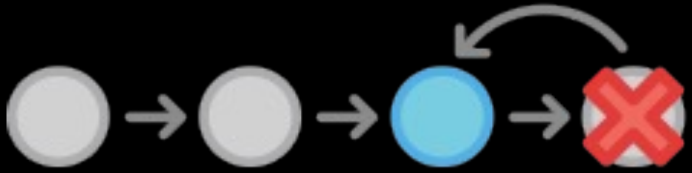
Reverting



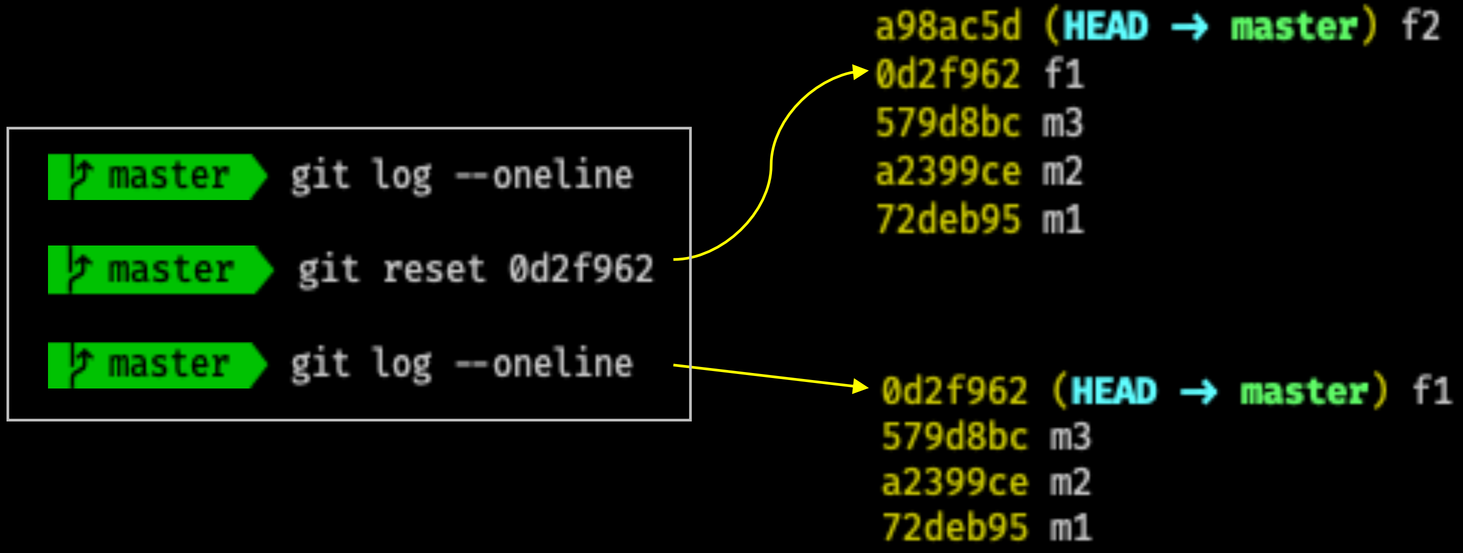
Resetting



Reset



```
git reset [commit_hash]
```

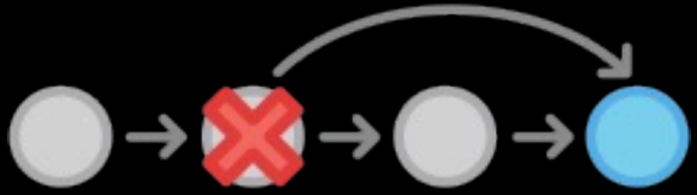


```
master  git log --oneline
master  git reset 0d2f962
master  git log --oneline
```

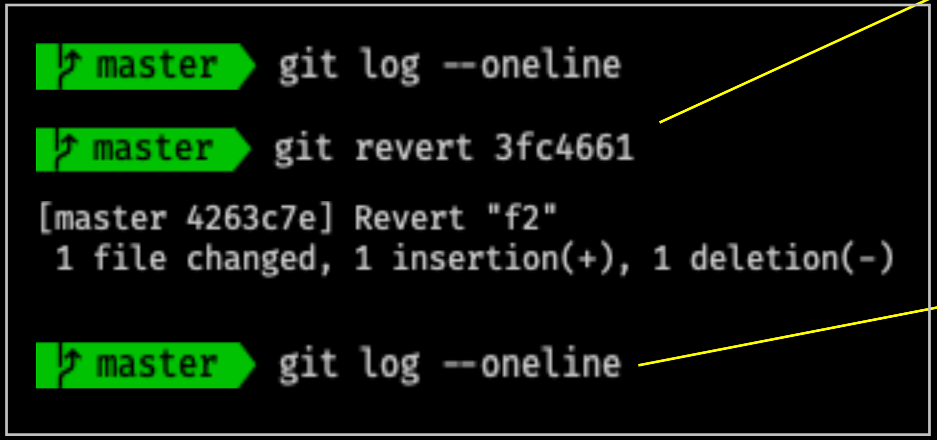
```
a98ac5d (HEAD → master) f2
0d2f962 f1
579d8bc m3
a2399ce m2
72deb95 m1
```

```
0d2f962 (HEAD → master) f1
579d8bc m3
a2399ce m2
72deb95 m1
```

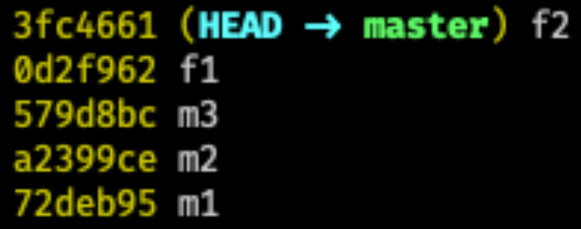
Revert



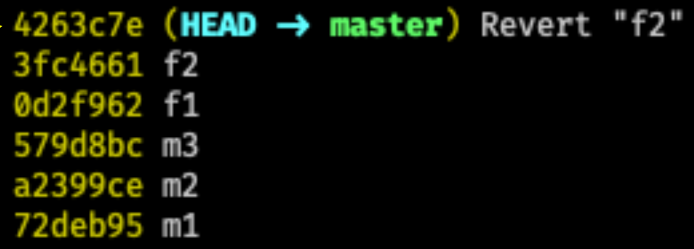
git revert [commit_hash]



```
master  git log --oneline

master  git revert 3fc4661

[master 4263c7e] Revert "f2"
 1 file changed, 1 insertion(+), 1 deletion(-)

master  git log --oneline
```



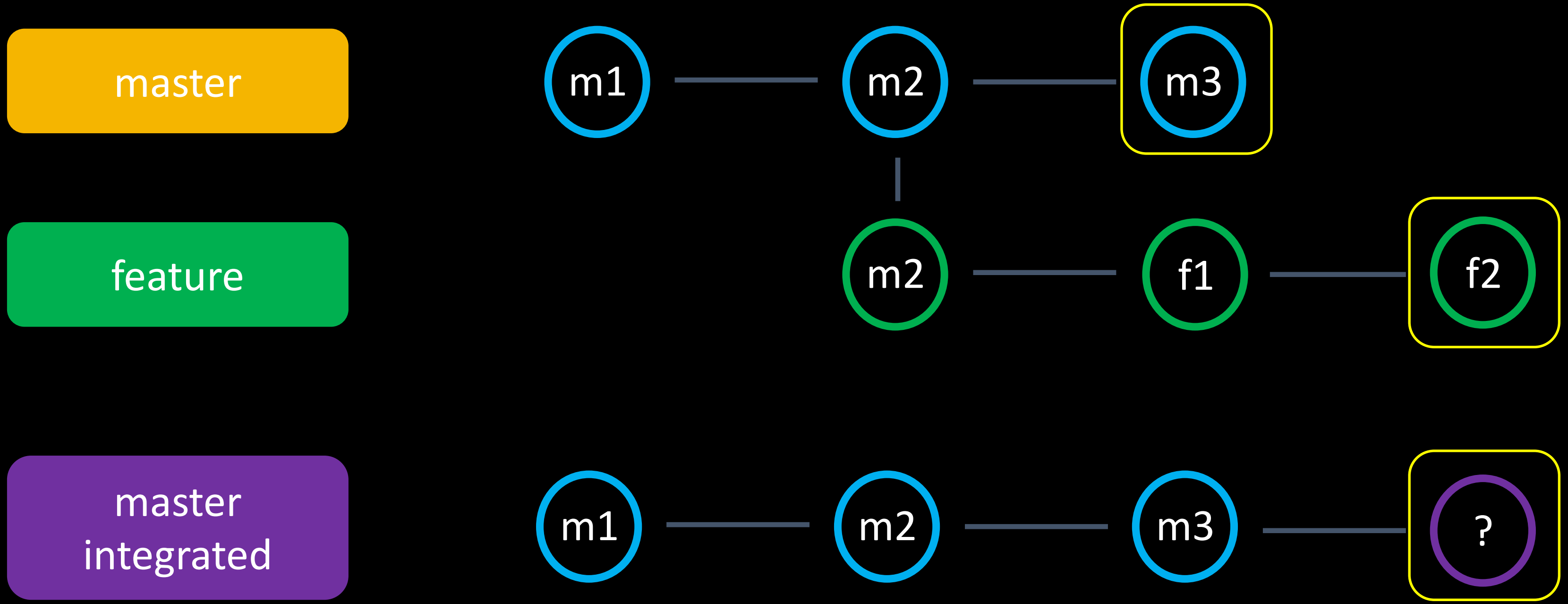
```
3fc4661 (HEAD → master) f2
0d2f962 f1
579d8bc m3
a2399ce m2
72deb95 m1
```



```
4263c7e (HEAD → master) Revert "f2"
3fc4661 f2
0d2f962 f1
579d8bc m3
a2399ce m2
72deb95 m1
```

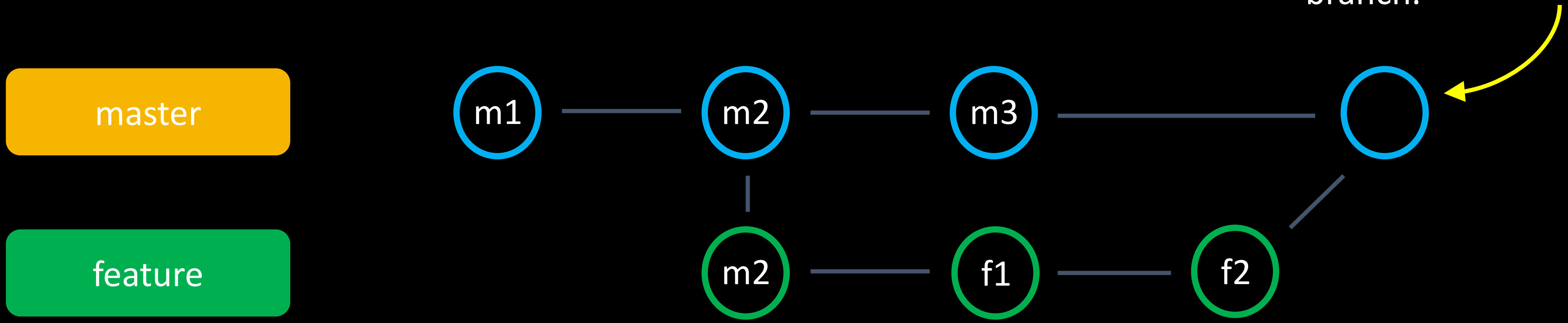
Merge vs Rebase

Different approach to branches integrations ("how to combine two branches").

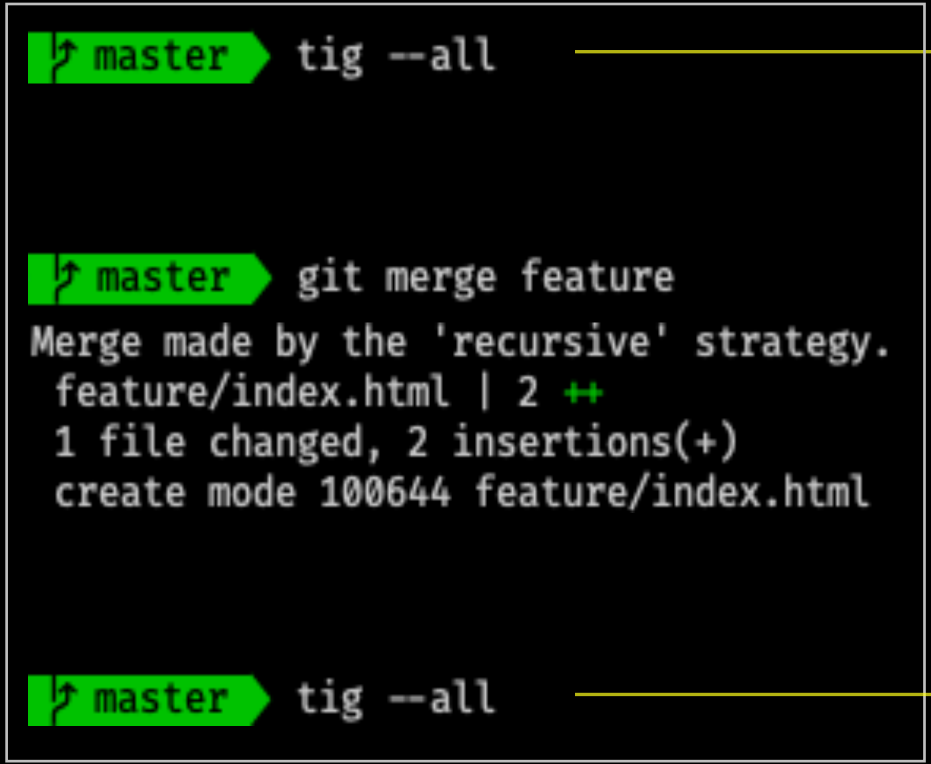


Merge

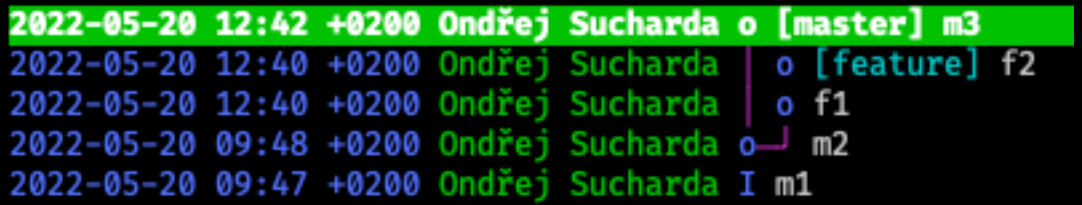
Merge creates new commit called “Merge commit”, which contains all changes from feature branch.



Merge

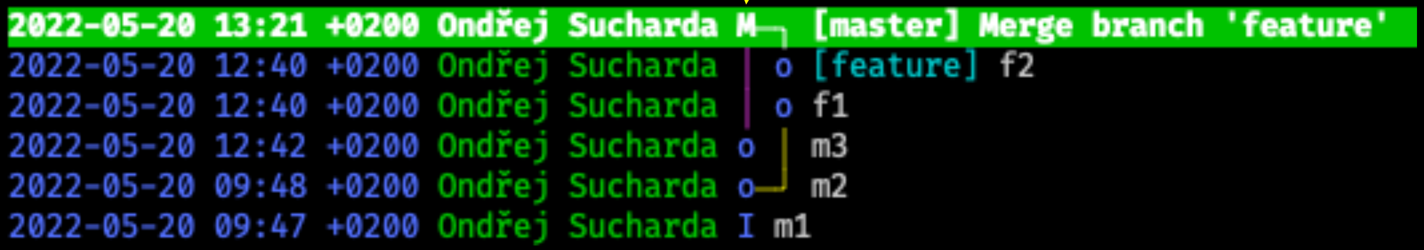


```
 master  tig --all


 master  git merge feature
Merge made by the 'recursive' strategy.
 feature/index.html | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 feature/index.html


 master  tig --all
```



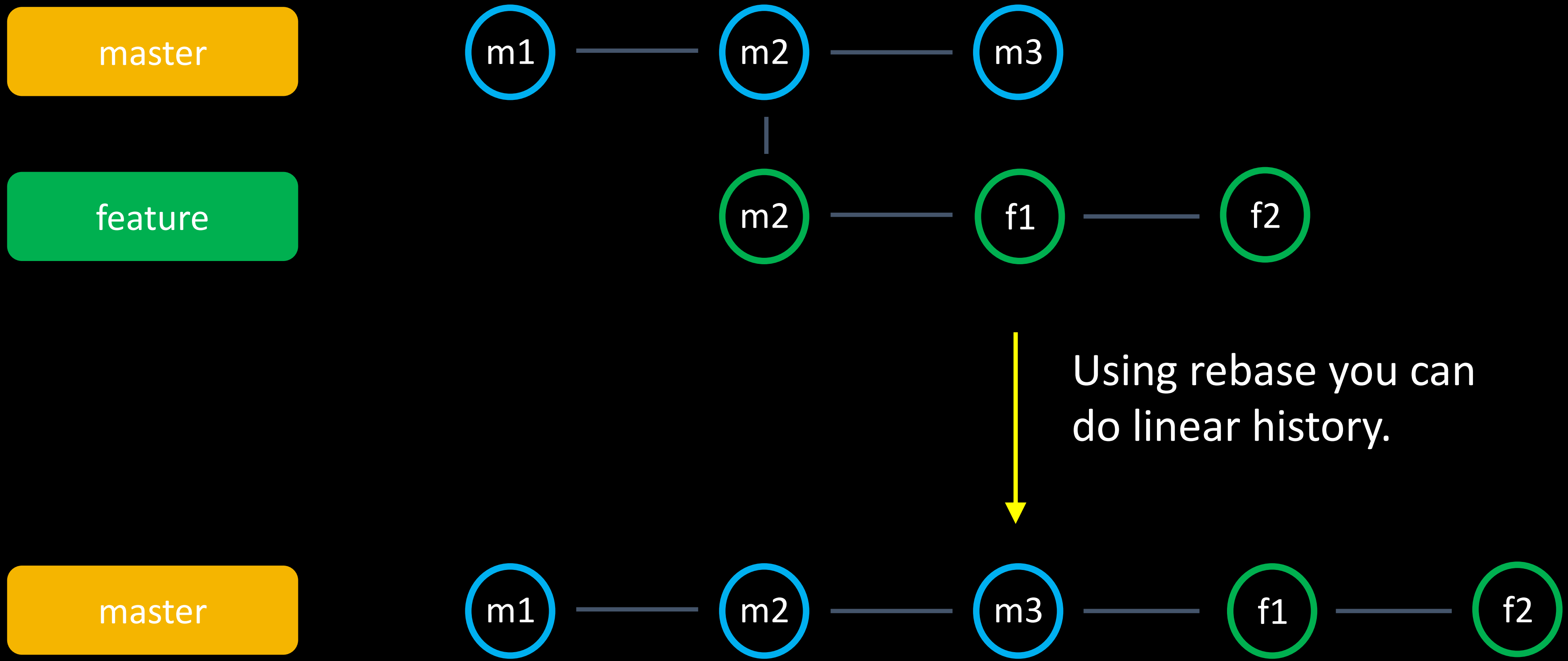
```
2022-05-20 12:42 +0200 Ondřej Sucharda o [master] m3
2022-05-20 12:40 +0200 Ondřej Sucharda │ o [feature] f2
2022-05-20 12:40 +0200 Ondřej Sucharda │ o f1
2022-05-20 09:48 +0200 Ondřej Sucharda o─┘ m2
2022-05-20 09:47 +0200 Ondřej Sucharda I m1
```

`git merge`
created this "Merge commit"



```
2022-05-20 13:21 +0200 Ondřej Sucharda M─┐ [master] Merge branch 'feature'
2022-05-20 12:40 +0200 Ondřej Sucharda │ o [feature] f2
2022-05-20 12:40 +0200 Ondřej Sucharda │ o f1
2022-05-20 12:42 +0200 Ondřej Sucharda o │ m3
2022-05-20 09:48 +0200 Ondřej Sucharda o─┘ m2
2022-05-20 09:47 +0200 Ondřej Sucharda I m1
```

Rebase

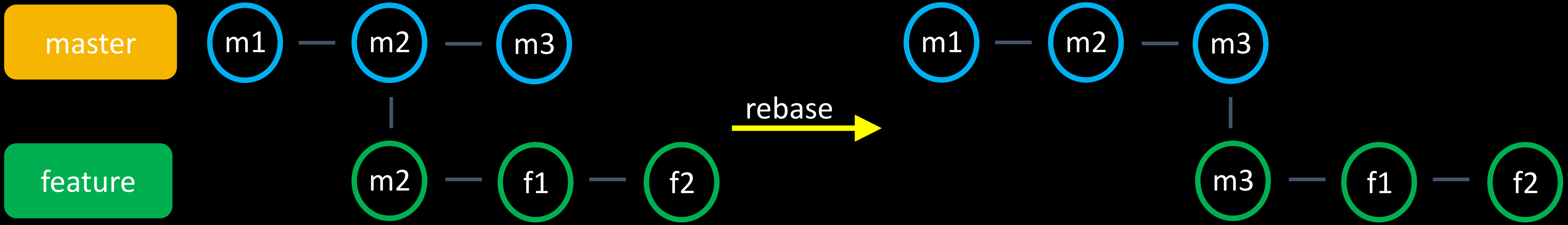


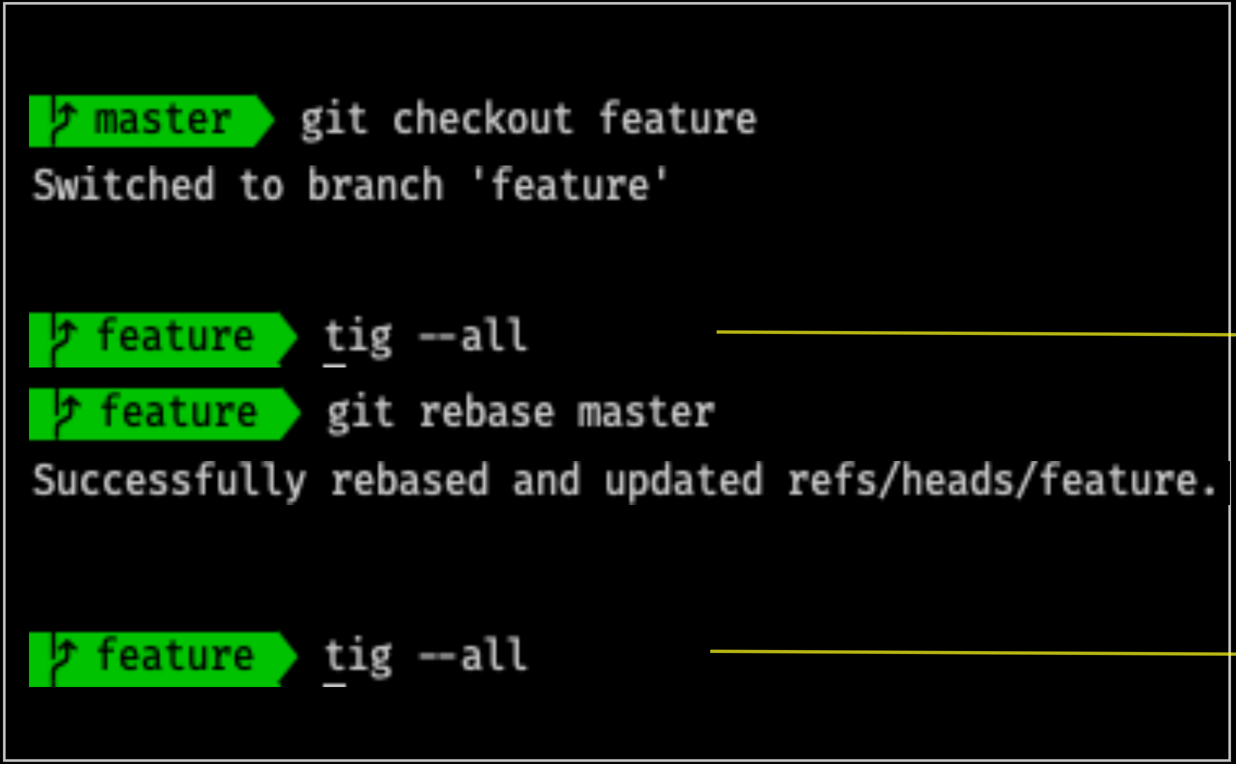
Using rebase you can do linear history.

Rebase

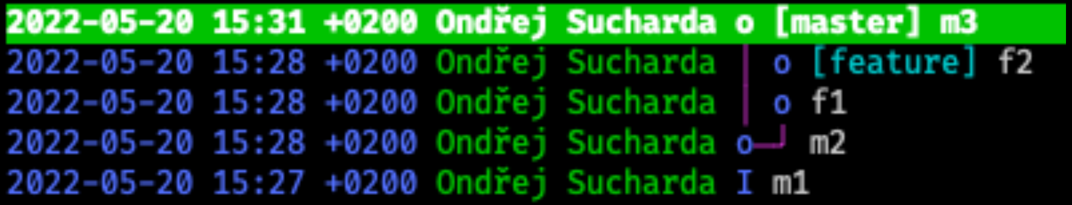
!

- Don’t use it on Long-Running branches (master/develop etc…)!
- Rebase is destructive operation! It changes git history!
- Use it only for your own Short-Lived branches (i.e. feature branch)!

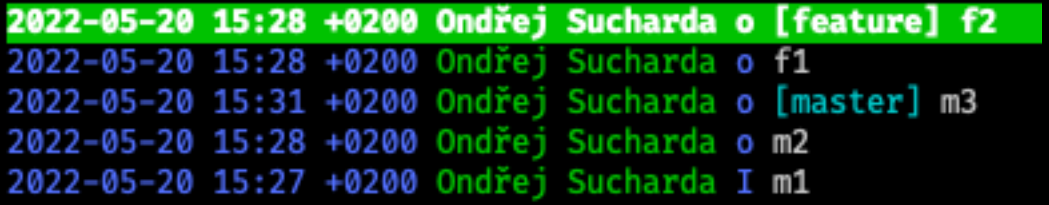




```
 master  git checkout feature
Switched to branch 'feature'

 feature  tig --all
 feature  git rebase master
Successfully rebased and updated refs/heads/feature.

 feature  tig --all
```

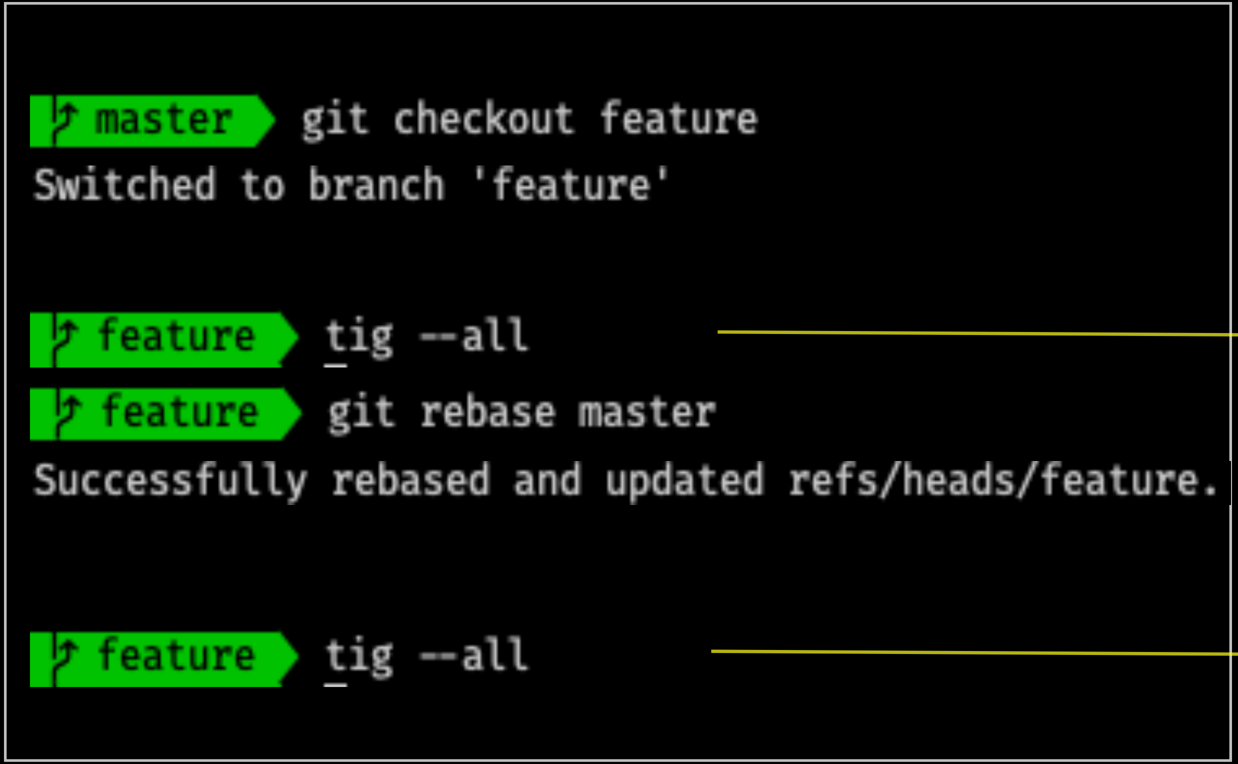


```
2022-05-20 15:31 +0200 Ondřej Sucharda o [master] m3
2022-05-20 15:28 +0200 Ondřej Sucharda │ o [feature] f2
2022-05-20 15:28 +0200 Ondřej Sucharda │ o f1
2022-05-20 15:28 +0200 Ondřej Sucharda o─┘ m2
2022-05-20 15:27 +0200 Ondřej Sucharda I m1
```

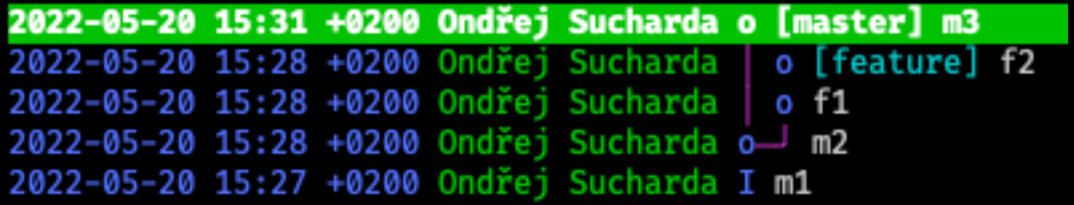


```
2022-05-20 15:28 +0200 Ondřej Sucharda o [feature] f2
2022-05-20 15:28 +0200 Ondřej Sucharda o f1
2022-05-20 15:31 +0200 Ondřej Sucharda o [master] m3
2022-05-20 15:28 +0200 Ondřej Sucharda o m2
2022-05-20 15:27 +0200 Ondřej Sucharda I m1
```

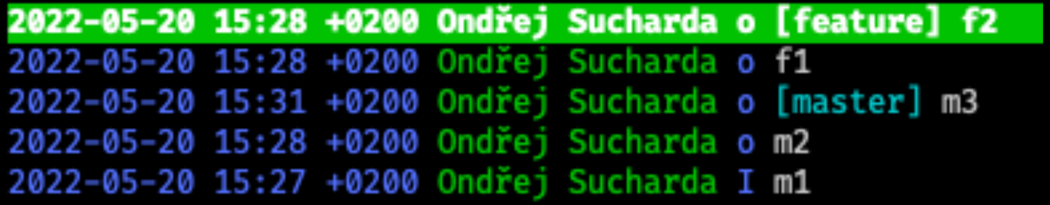
Our branch “feature” is on top of “master” due to Rebase.



```
 master  git checkout feature
Switched to branch 'feature'

 feature  tig --all
 feature  git rebase master
Successfully rebased and updated refs/heads/feature.

 feature  tig --all
```

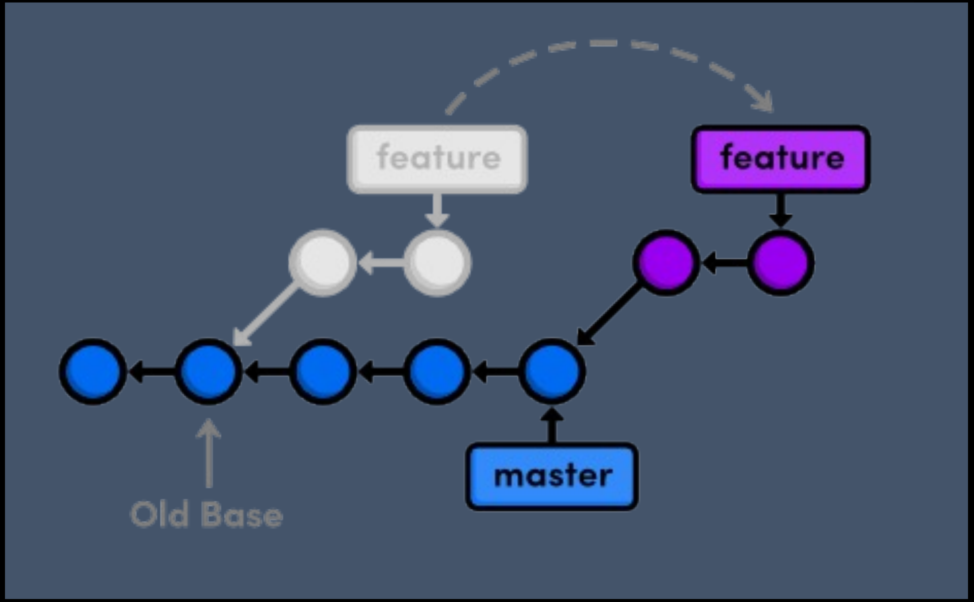


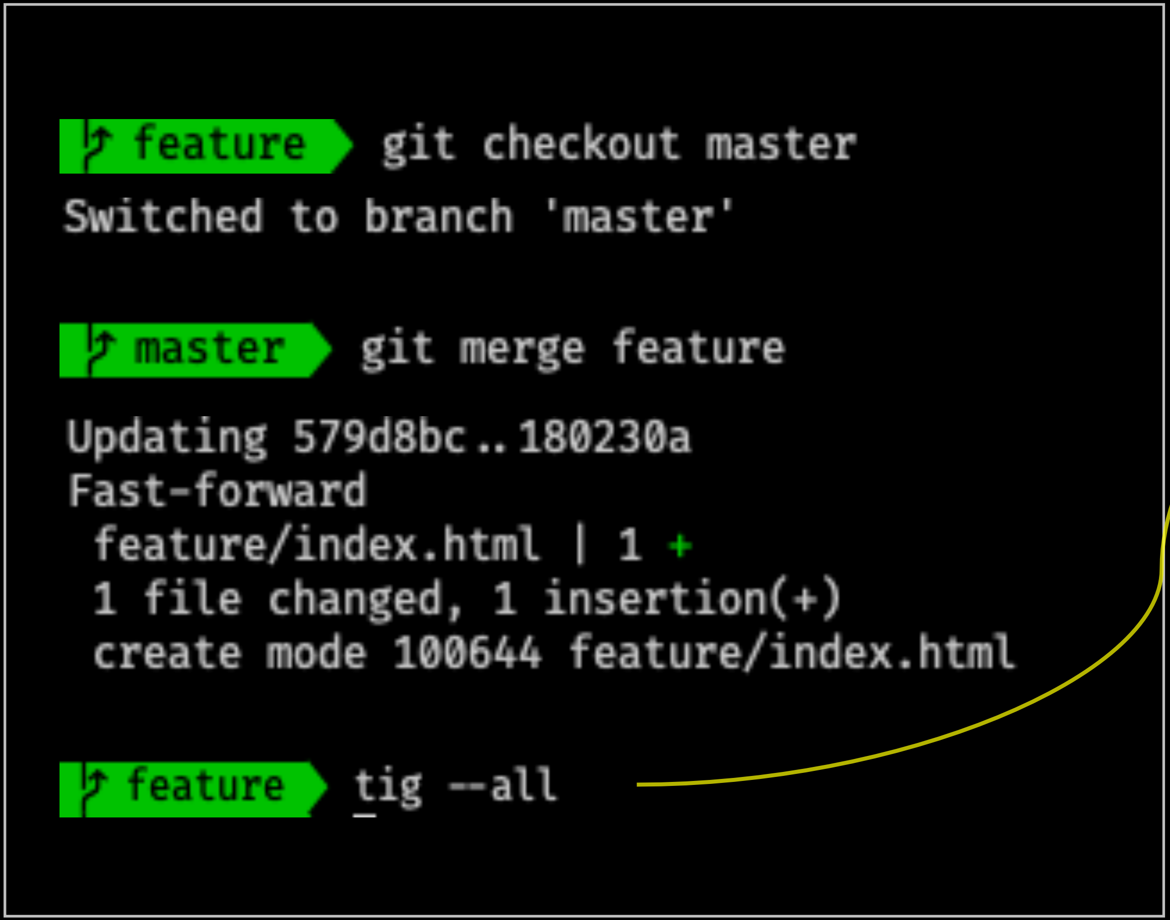
```
2022-05-20 15:31 +0200 Ondřej Sucharda o [master] m3
2022-05-20 15:28 +0200 Ondřej Sucharda │ o [feature] f2
2022-05-20 15:28 +0200 Ondřej Sucharda │ o f1
2022-05-20 15:28 +0200 Ondřej Sucharda o─┘ m2
2022-05-20 15:27 +0200 Ondřej Sucharda I m1
```



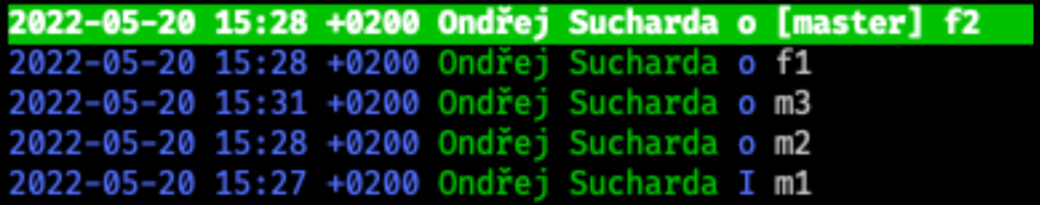
```
2022-05-20 15:28 +0200 Ondřej Sucharda o [feature] f2
2022-05-20 15:28 +0200 Ondřej Sucharda o f1
2022-05-20 15:31 +0200 Ondřej Sucharda o [master] m3
2022-05-20 15:28 +0200 Ondřej Sucharda o m2
2022-05-20 15:27 +0200 Ondřej Sucharda I m1
```

Our branch “feature” is on top of “master” due to Rebase.





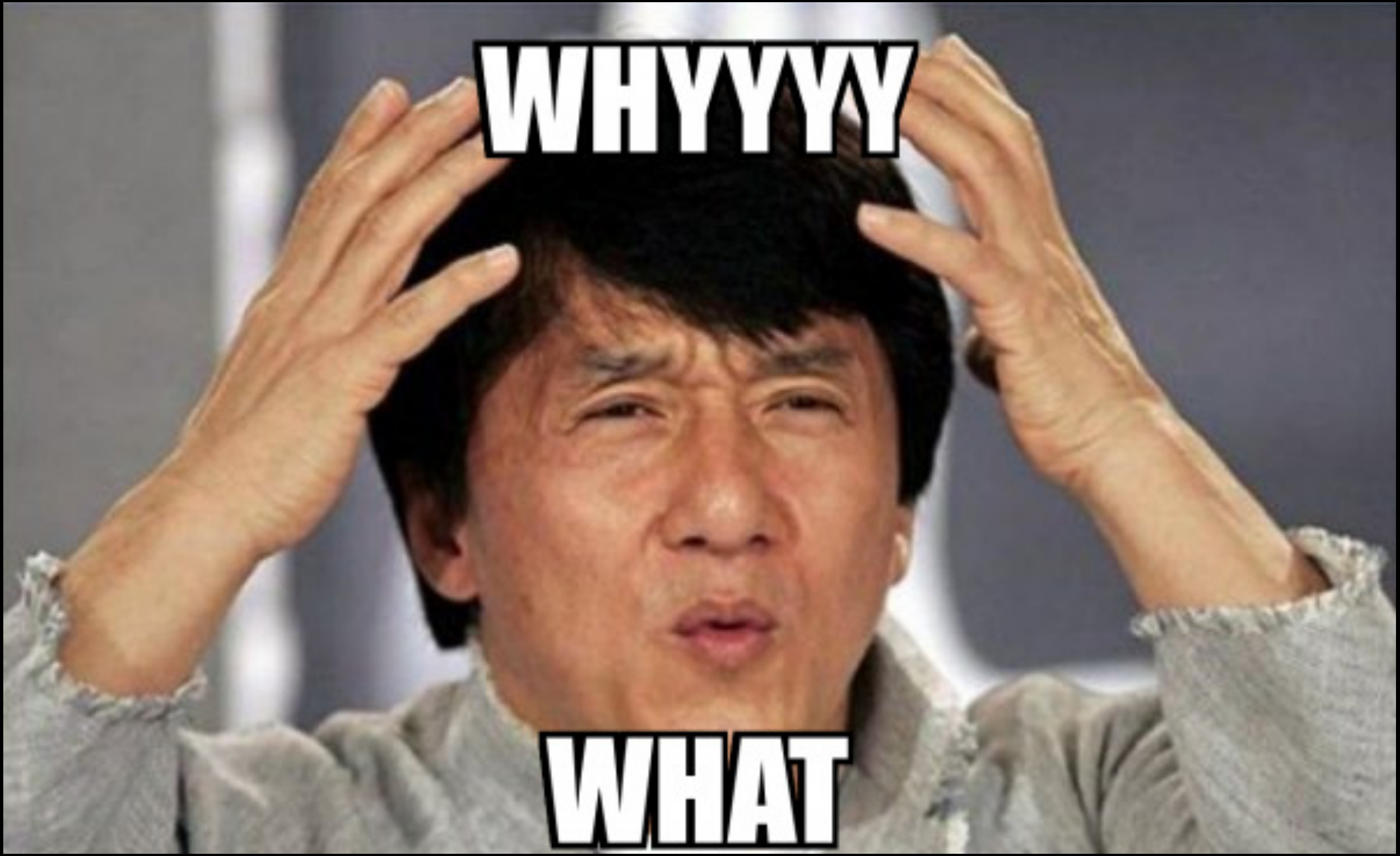
```
 feature  git checkout master
Switched to branch 'master'

 master  git merge feature
Updating 579d8bc..180230a
Fast-forward
 feature/index.html | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 feature/index.html

 feature  tig --all
```

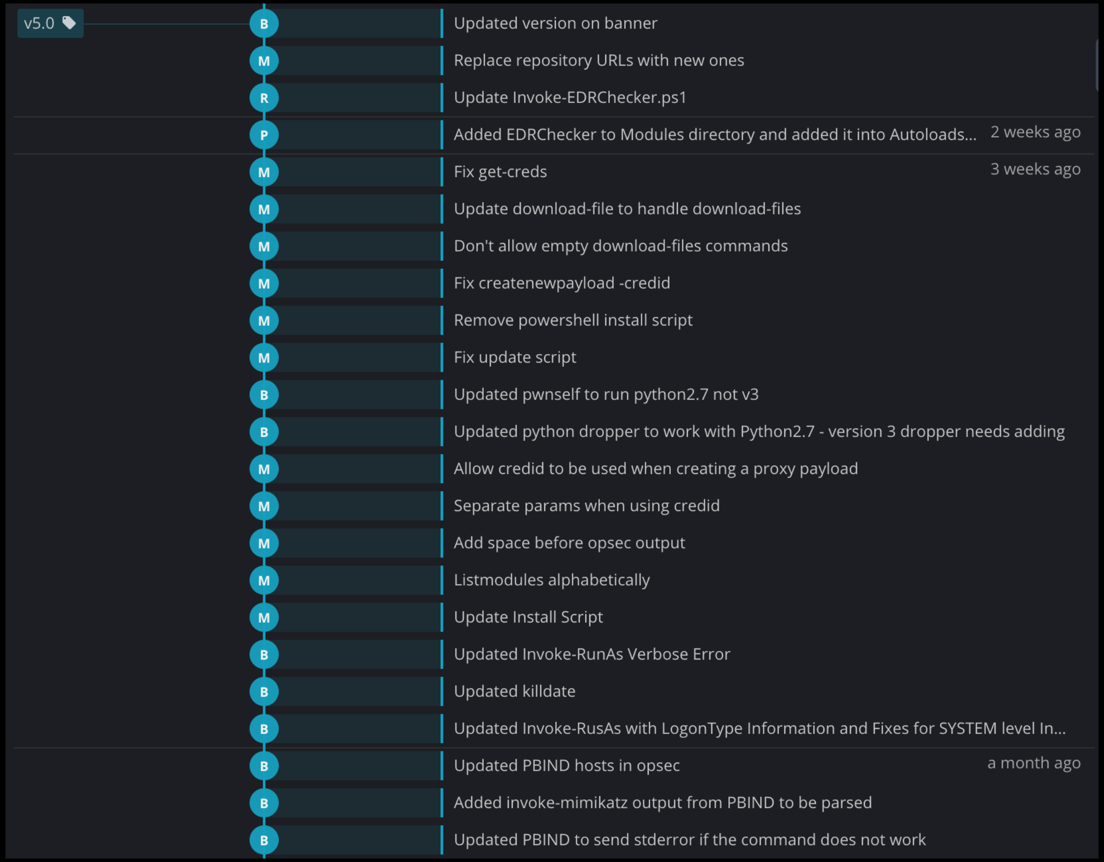


```
2022-05-20 15:28 +0200 Ondřej Sucharda o [master] f2
2022-05-20 15:28 +0200 Ondřej Sucharda o f1
2022-05-20 15:31 +0200 Ondřej Sucharda o m3
2022-05-20 15:28 +0200 Ondřej Sucharda o m2
2022-05-20 15:27 +0200 Ondřej Sucharda I m1
```

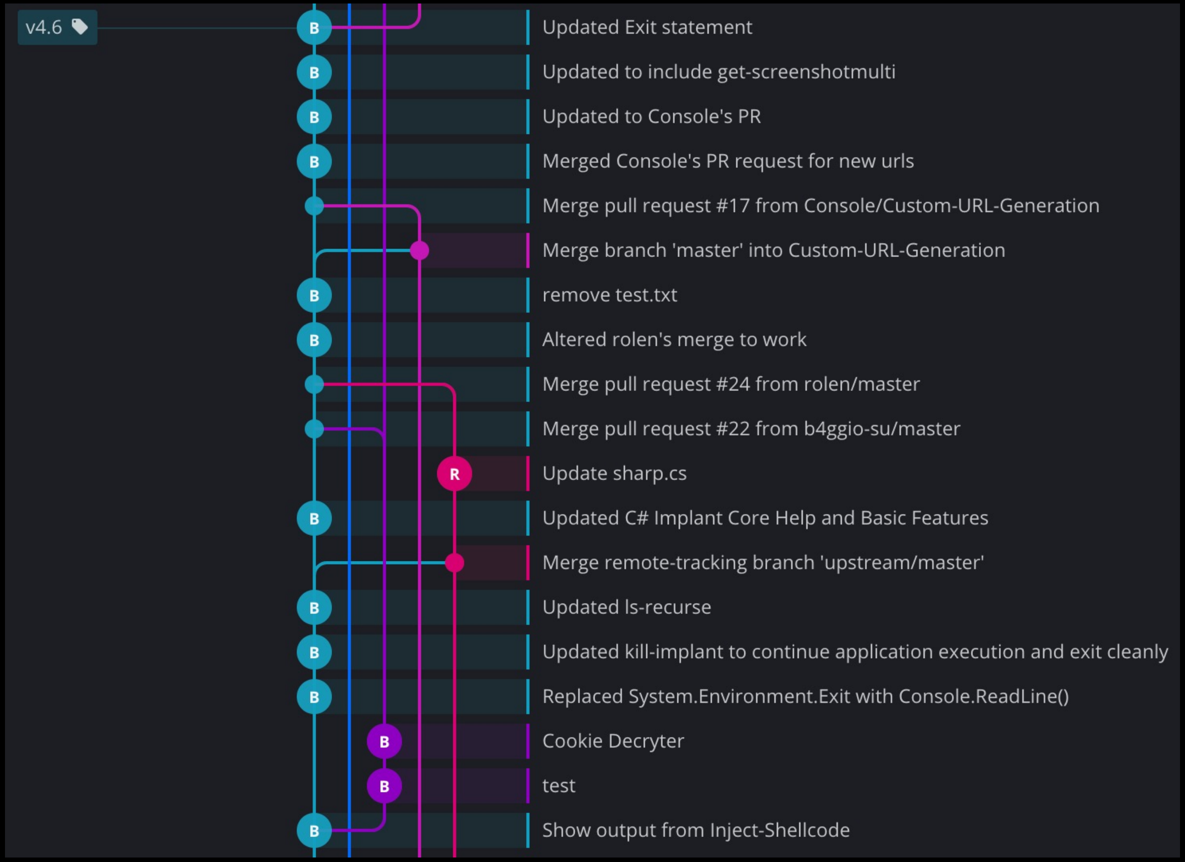
Git uses Fast forward merge, that’s why:

1. Now “git merge feature” doesn’t create merge commit
2. You have a linear history


WHYYYY
WHAT


v5.0

B Updated version on banner
M Replace repository URLs with new ones
R Update Invoke-EDRChecker.ps1
P Added EDRChecker to Modules directory and added it into Autoloads... 2 weeks ago
M Fix get-creds 3 weeks ago
M Update download-file to handle download-files
M Don't allow empty download-files commands
M Fix createnewpayload -credid
M Remove powershell install script
M Fix update script
B Updated pwnself to run python2.7 not v3
B Updated python dropper to work with Python2.7 - version 3 dropper needs adding
M Allow credid to be used when creating a proxy payload
M Separate params when using credid
M Add space before opsec output
M Listmodules alphabetically
M Update Install Script
B Updated Invoke-RunAs Verbose Error
B Updated killdate
B Updated Invoke-RusAs with LogonType Information and Fixes for SYSTEM level In...
B Updated PBIND hosts in opsec a month ago
B Added invoke-mimikatz output from PBIND to be parsed
B Updated PBIND to send stderror if the command does not work


v4.6

B Updated Exit statement
B Updated to include get-screenshotmulti
B Updated to Console's PR
B Merged Console's PR request for new urls
Merge pull request #17 from Console/Custom-URL-Generation
Merge branch 'master' into Custom-URL-Generation
B remove test.txt
B Altered rolen's merge to work
Merge pull request #24 from rolen/master
Merge pull request #22 from b4ggio-su/master
R Update sharp.cs
B Updated C# Implant Core Help and Basic Features
Merge remote-tracking branch 'upstream/master'
B Updated ls-recurse
B Updated kill-implant to continue application execution and exit cleanly
B Replaced System.Environment.Exit with Console.ReadLine()
B Cookie Decryter
B test
B Show output from Inject-Shellcode

Squash and Rebase policy

When development policy is set to squash and rebase:

1. Squash (everything is “squashed” into 1 commit)
2. Rebase feature branch onto master
3. Merge into master

Classic merging

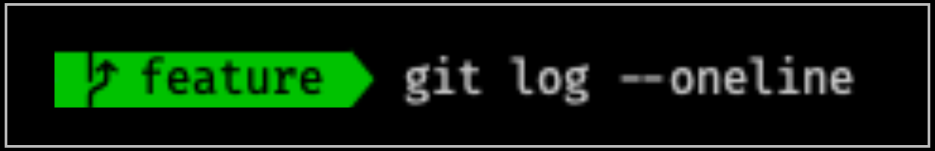
Interactive Rebase

Interactive Rebase

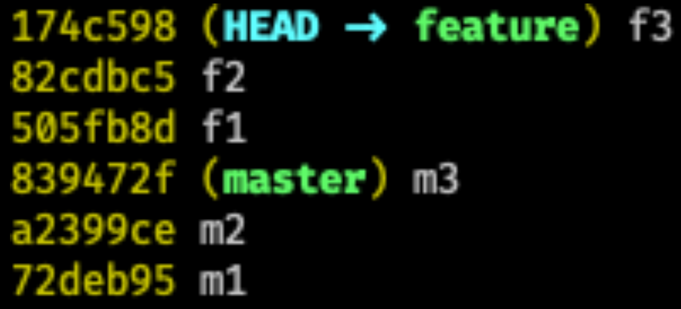
- Rewriting git history
- You can change commits (rename commit message, squash commits, …)

- Use it only for your own feature branch!

Interactive Rebase

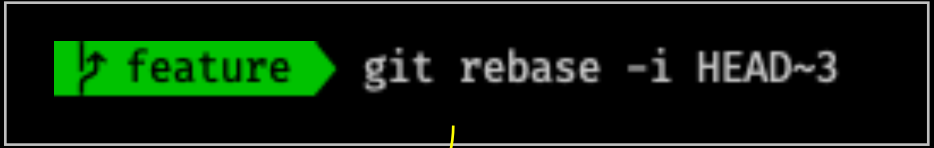


```
feature  git log --oneline
```

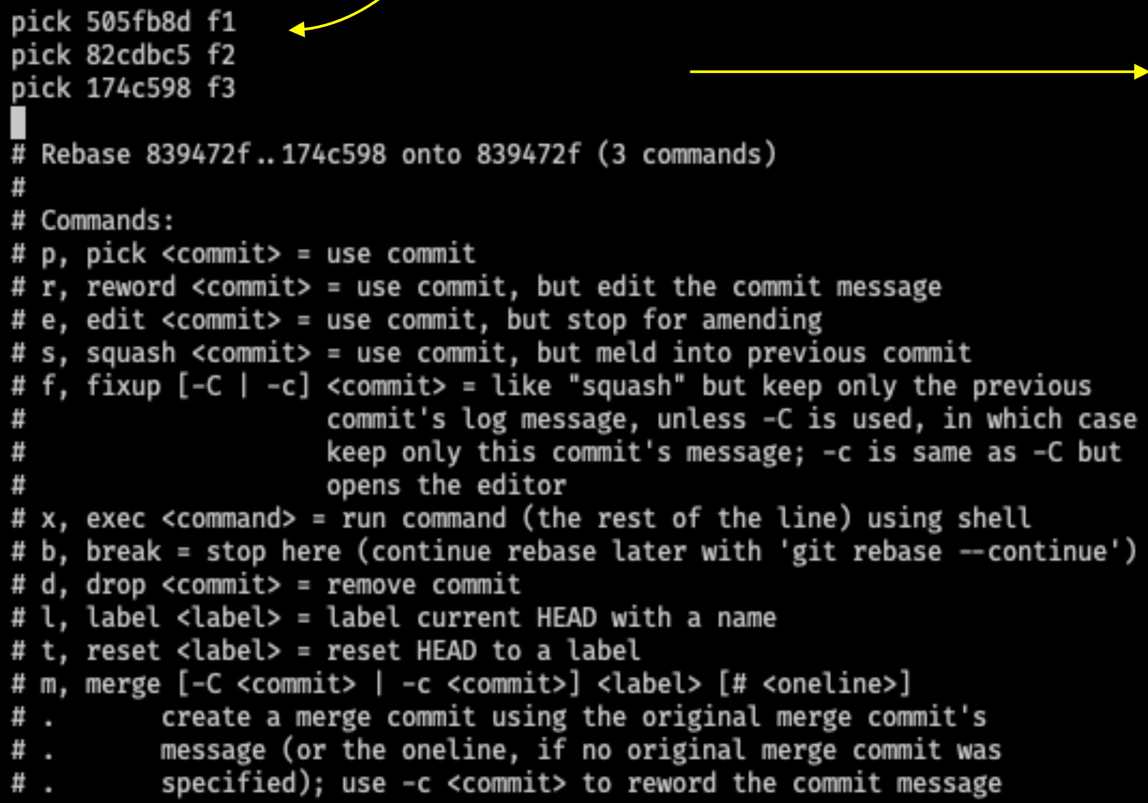


```
174c598 (HEAD -> feature) f3
82cdbc5 f2
505fb8d f1
839472f (master) m3
a2399ce m2
72deb95 m1
```

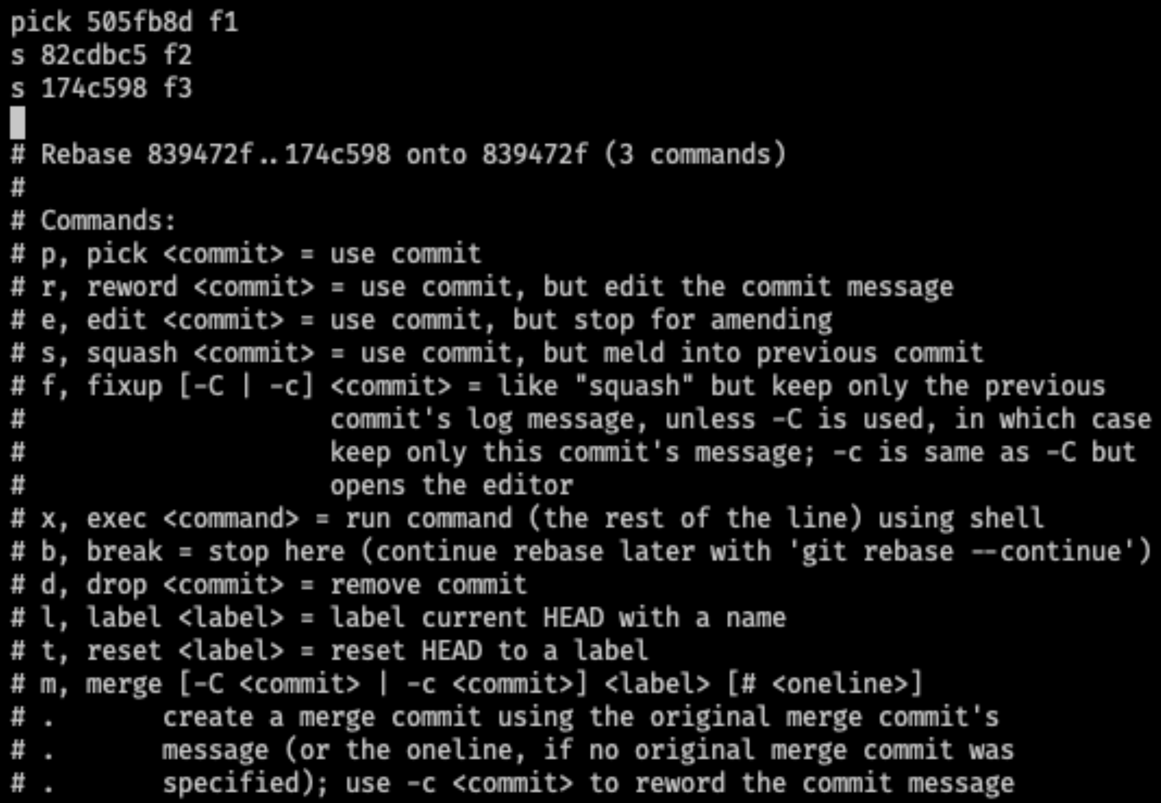
Branch ”feature” contains commits f1, f2, f3.
Let’s squash them into 1 commit.



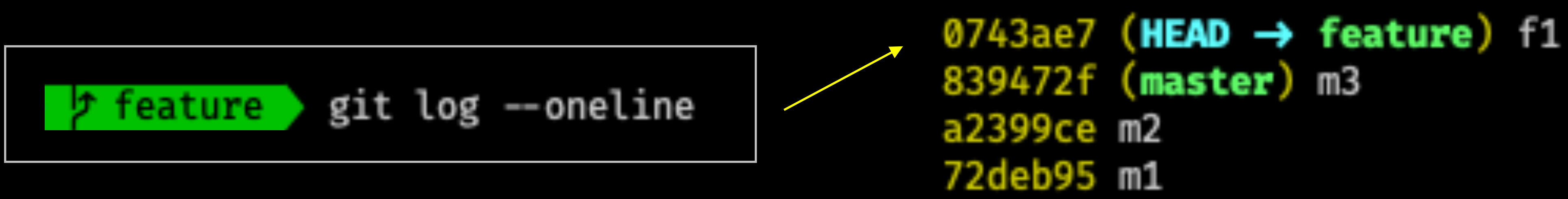
```
feature  git rebase -i HEAD~3
```



```
pick 505fb8d f1
pick 82cdbc5 f2
pick 174c598 f3

# Rebase 839472f..174c598 onto 839472f (3 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup [-C | -c] <commit> = like "squash" but keep only the previous
#                    commit's log message, unless -C is used, in which case
#                    keep only this commit's message; -c is same as -C but
#                    opens the editor
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
# .       create a merge commit using the original merge commit's
# .       message (or the oneline, if no original merge commit was
# .       specified); use -c <commit> to reword the commit message
```



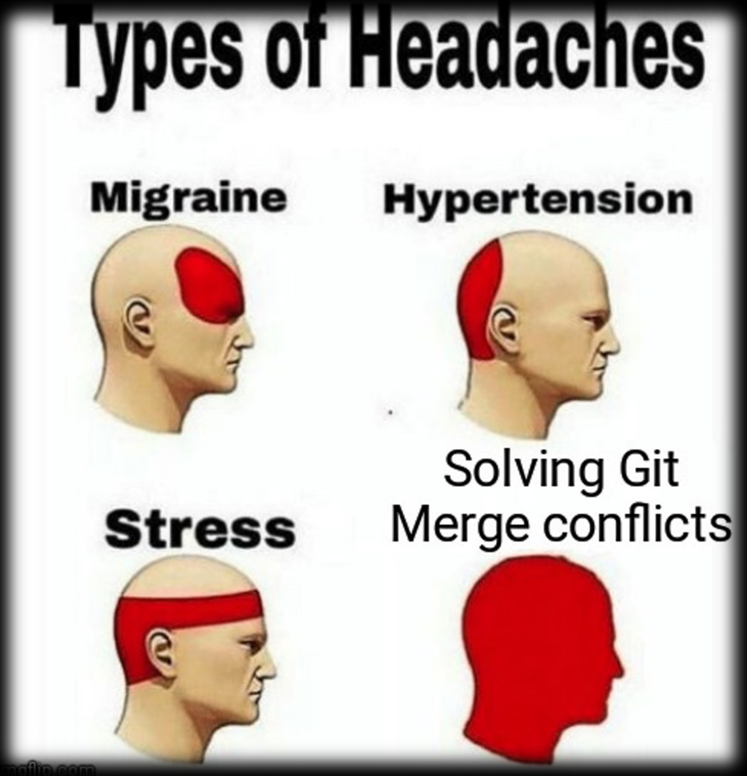
```
pick 505fb8d f1
s 82cdbc5 f2
s 174c598 f3

# Rebase 839472f..174c598 onto 839472f (3 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup [-C | -c] <commit> = like "squash" but keep only the previous
#                    commit's log message, unless -C is used, in which case
#                    keep only this commit's message; -c is same as -C but
#                    opens the editor
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
# .       create a merge commit using the original merge commit's
# .       message (or the oneline, if no original merge commit was
# .       specified); use -c <commit> to reword the commit message
```



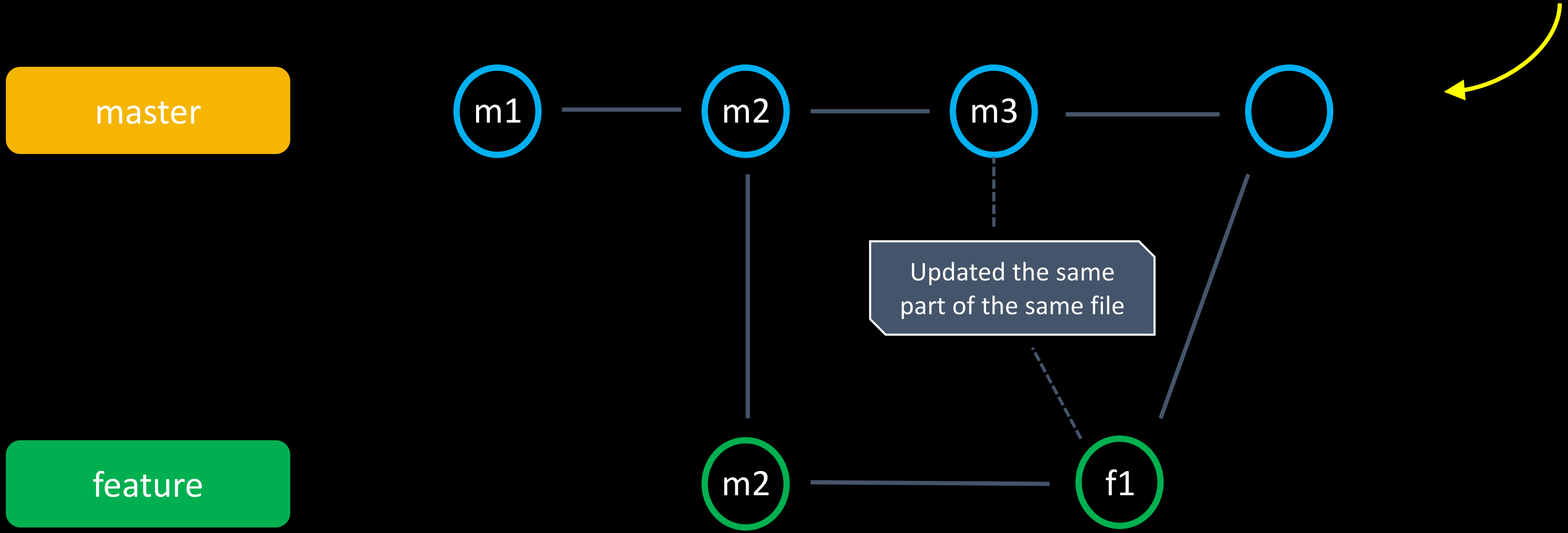
```
feature  git log --oneline
```

```
0743ae7 (HEAD → feature) f1
839472f (master) m3
a2399ce m2
72deb95 m1
```

Merge conflicts

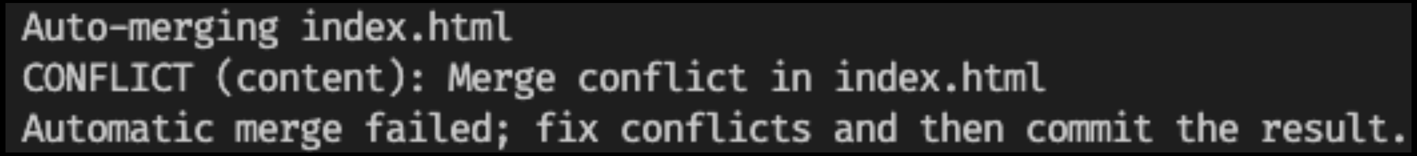


If you've changed the same part of the same file differently in the two branches you want to merge, Git won't be able to merge them automatically = MERGE CONFLICT.

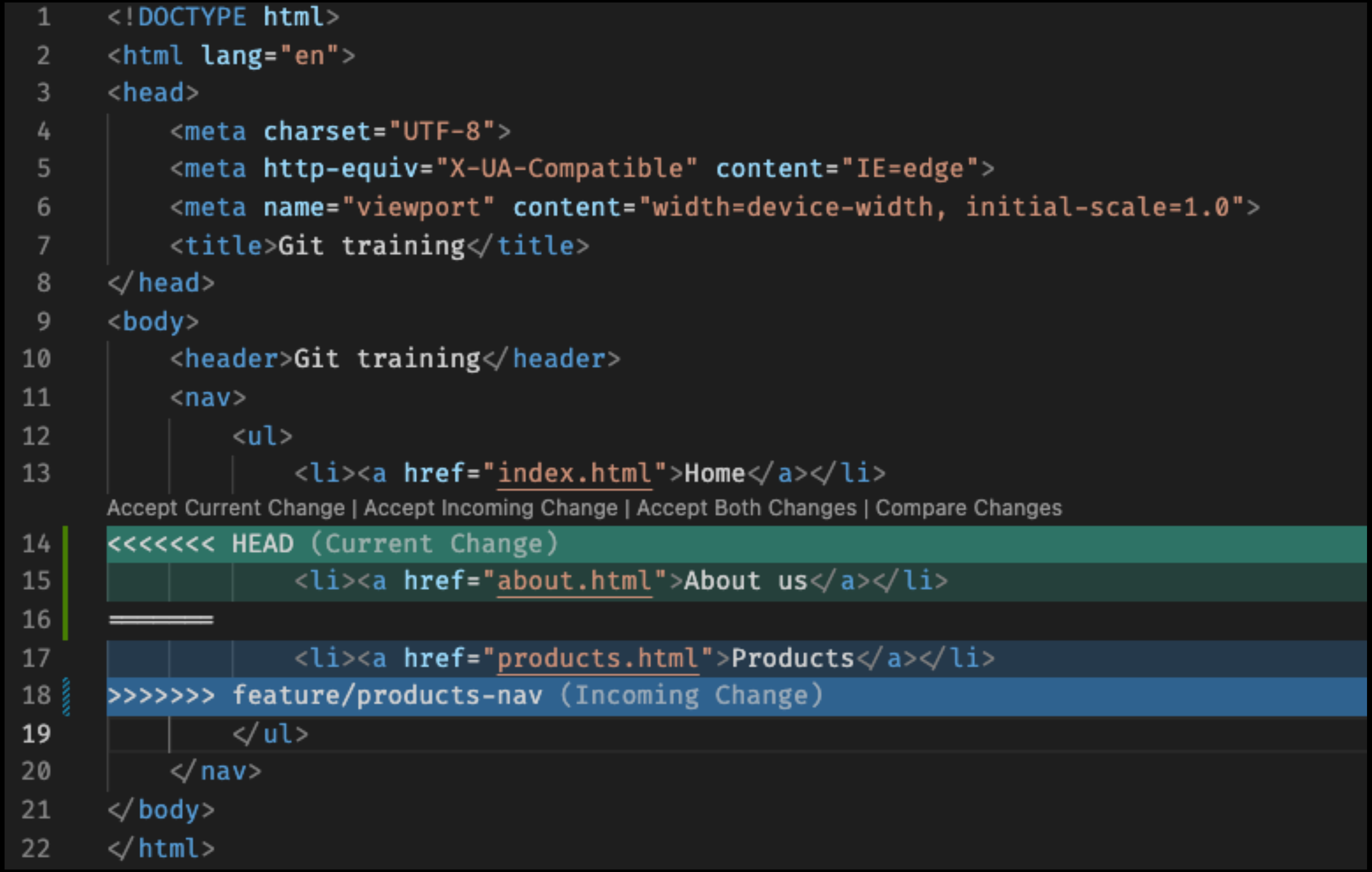


1

Git tells you about merge conflicts when you want to merge branches.



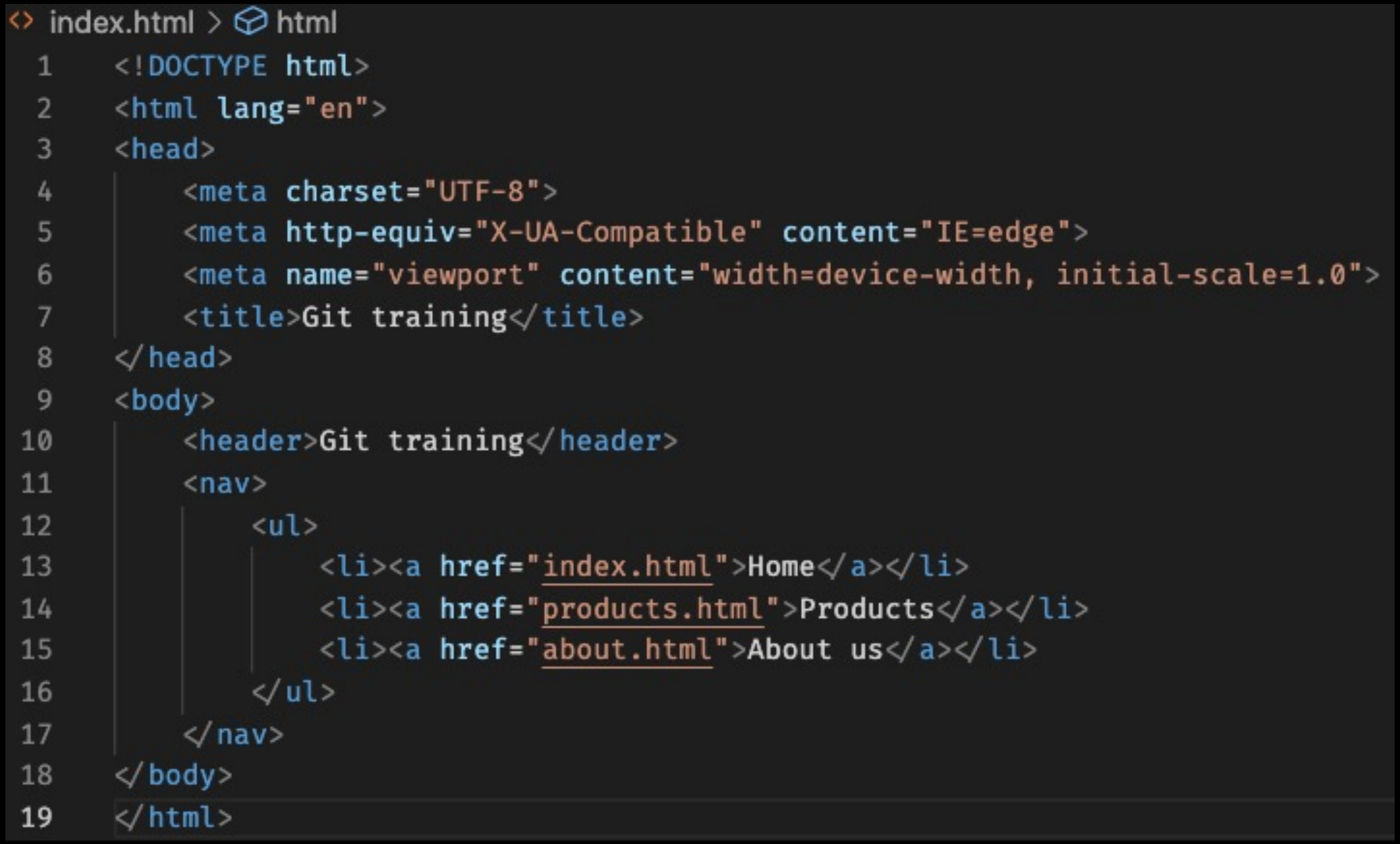
```
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```



Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Git training</title>
</head>
<body>
    <header>Git training</header>
    <nav>
        <ul>
            <li><a href="index.html">Home</a></li>
<<<<<<< HEAD (Current Change)
            <li><a href="about.html">About us</a></li>
=======
            <li><a href="products.html">Products</a></li>
>>>>>>> feature/products-nav (Incoming Change)
        </ul>
    </nav>
</body>
</html>
```

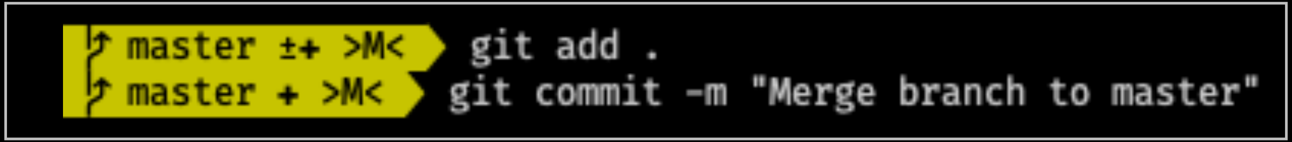
2 Resolve conflicts manually.



index.html > html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Git training</title>
</head>
<body>
    <header>Git training</header>
    <nav>
        <ul>
            <li><a href="index.html">Home</a></li>
            <li><a href="products.html">Products</a></li>
            <li><a href="about.html">About us</a></li>
        </ul>
    </nav>
</body>
</html>
```

3 Add changes to staging area and commit



```
master ±+ >M<  git add .
master + >M<  git commit -m "Merge branch to master"
```

Questions ?

Thanks
☺