Software Version Control - 2

David Camilo Delgado Arias
Ingeniero de Sistemas y Computación
dcdelgadoa@unal.edu.co
Universidad Nacional de Colombia

Topics

- ☐ Configuring Git
- ☐ Working with Tags
- ☐ Stashes
- ☐ Merging Branches
- ☐ Branching Strategies (Workflows)

Configuring Git

Git Configuration

- We can configure git in three ways: Local, Global and System Wide.
- Local configuration only impacts the current repository that you are working on.
- Global configuration impacts all repositories handled by the instance of git for the current user.
- System configuration impacts all the repositories handled by the instance of git for all the users on the system

Git Configuration

- To configure git we use the command:
  - System: git config - -system <parameter>
  - Global: git config - -global <parameter>
  - Local: git config <parameter>

Git Configuration

- Git stores the configurations in the following locations
  - /etc/gitconfig (System)
  - ~/.gitconfig (Global)
  - .git/config (Local)
- Each level overrides the configuration from the previous levels

**Git configuration should be edited carefully!
Otherwise you could damage the configuration
and should restore the original one or reinstall git**

Git Configuration Parameters

- To check all the possible parameters from the config that you can edit you should run: man git-config.
- user.name: Changes the name associated with the new commits.
- user.email: Changes the email address associated with the new commits.

Git Configuration Parameters

- http.proxy <proxy>: Changes the proxy used by git to connect to the internet. Usual Template:
  - http://<proxyuser>:<proxypwd>@<proxy.server.com>:<proxy_port>
- core.editor: Changes the default editor associated with git to edit the commit messages.
- commit.template <file_route>: Changes the default template file associated with the new commits.
- core.excludesfile <file_route>: Changes the default ignore file which contents the patterns for the files to be excluded

Git Aliases

- We can use aliases to create shortcuts for our most used commands in git, so we just type the alias in the console and not the entire command.

Git Aliases

- git config --global alias.co checkout
- git config --global alias.br branch
- git config --global alias.ci commit
- git config --global alias.st status

Working with Tags

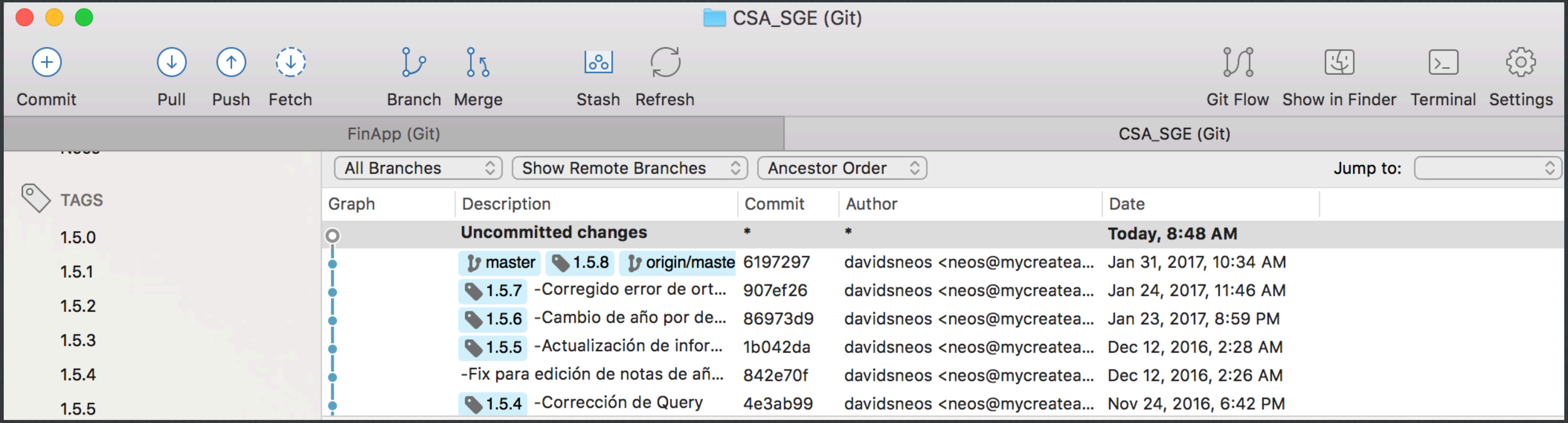
Tags

- Tags are a useful feature in git which allow us to mark a specified commit as important or relevant for the project using a unique string
- Tags are commonly used to mark commits which holds a release version of the software (v1.0, v2.2, v2.3) so we keep track of the versions of our software
- It can be used to assign a specific unique version number to a unique state of software (Commit) (Software Versioning)

Tags Common Operations

- List Tags: `git tag`
- List Filtered Tags: `git tag -l “<pattern>”`
- Create Tag on the current commit: `git tag -a <tag_name> -m “<tag_message>”`
- Create Tag on a specific commit: `git tag -a <tag_name> <commit_hash>`
- Show Tag and associated commit info: `git show <tag_name>`

Tags in SourceTree



Graph	Description	Commit	Author	Date
	Uncommitted changes	*	*	Today, 8:48 AM
	master 1.5.8 origin/maste	6197297	davidsneos <neos@mycreatea...	Jan 31, 2017, 10:34 AM
	1.5.7 -Corregido error de ort...	907ef26	davidsneos <neos@mycreatea...	Jan 24, 2017, 11:46 AM
	1.5.6 -Cambio de año por de...	86973d9	davidsneos <neos@mycreatea...	Jan 23, 2017, 8:59 PM
	1.5.5 -Actualización de infor...	1b042da	davidsneos <neos@mycreatea...	Dec 12, 2016, 2:28 AM
	-Fix para edición de notas de añ...	842e70f	davidsneos <neos@mycreatea...	Dec 12, 2016, 2:26 AM
	1.5.4 -Corrección de Query	4e3ab99	davidsneos <neos@mycreatea...	Nov 24, 2016, 6:42 PM

Stashes

Stashes

- A Stash takes the modified (dirty) state of the working directory (i.e. Modified tracked files and staged changes) and saves it on a stack on unfinished changes that can be reapplied when needed

Stashes

- We can create a new stash using: `git stash`
- We can list the available stashes using: `git stash list`

Stashes

- To apply the last stash created (Remember: Stack) we can use: git stash apply
- To apply a specific stash created we can use: git stash apply <stash_name>
- To apply a specific stash and try to reapply the previous staged changes, we can use: git stash apply - -index

Stashes

- To delete a specific stash we created before, we can use: git stash drop <stash_name>
- To apply the last stash created and immediately drop it from the stack, we can use: git stash pop

Stashes

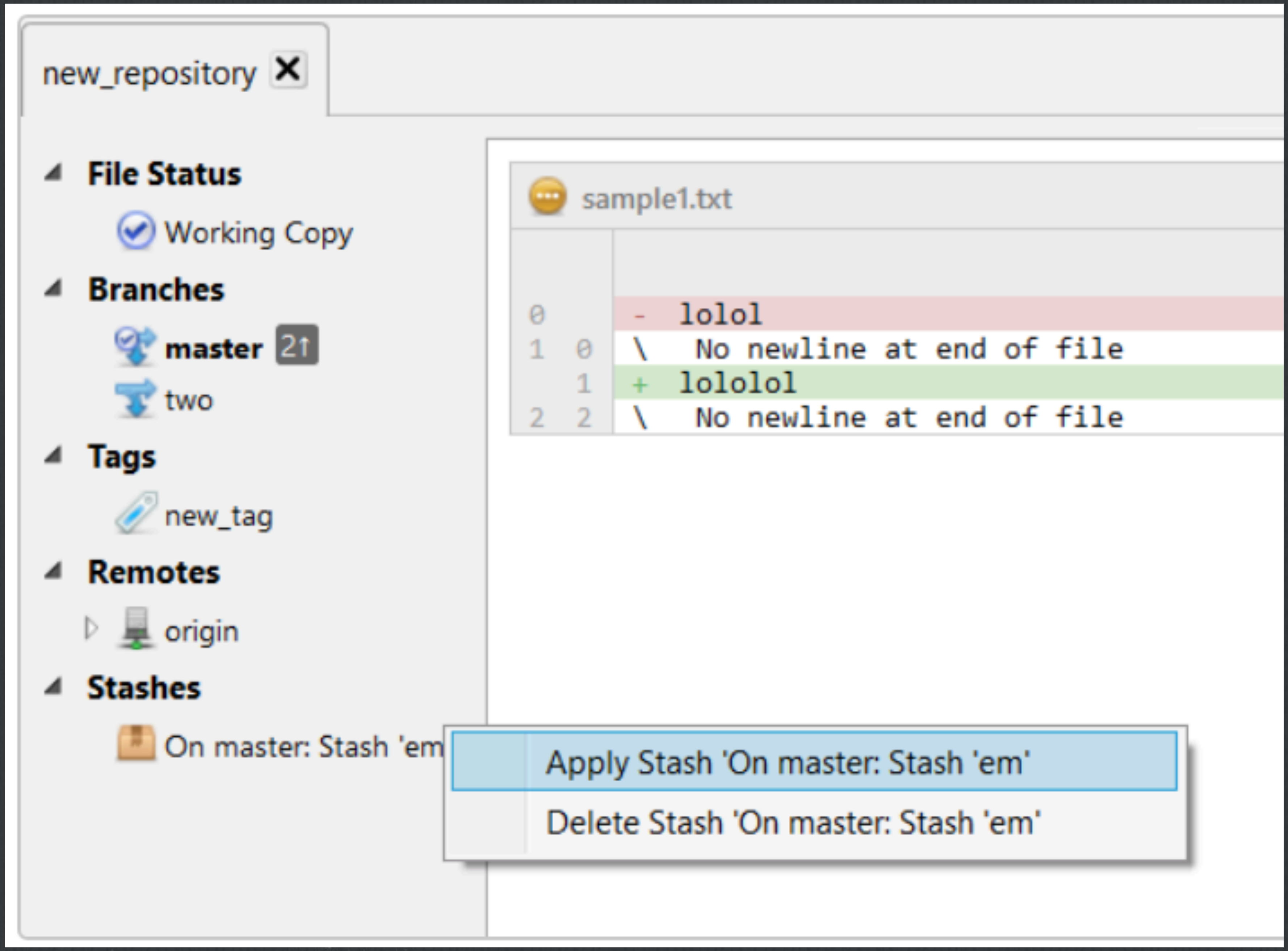
- Is not easy to unapplied a Stash, but we can achieve the effect retrieving the patch associated with the Stash and applied in reverse: git stash show -p <stash_name> | git apply -R

- Remember: We can create an alias in git to create a shortcut for the command

Stashes

- We can also **create a branch** starting from a **specific stash**. To achieve this we use: **git stash branch <branch_name>**

Stashes in SourceTree



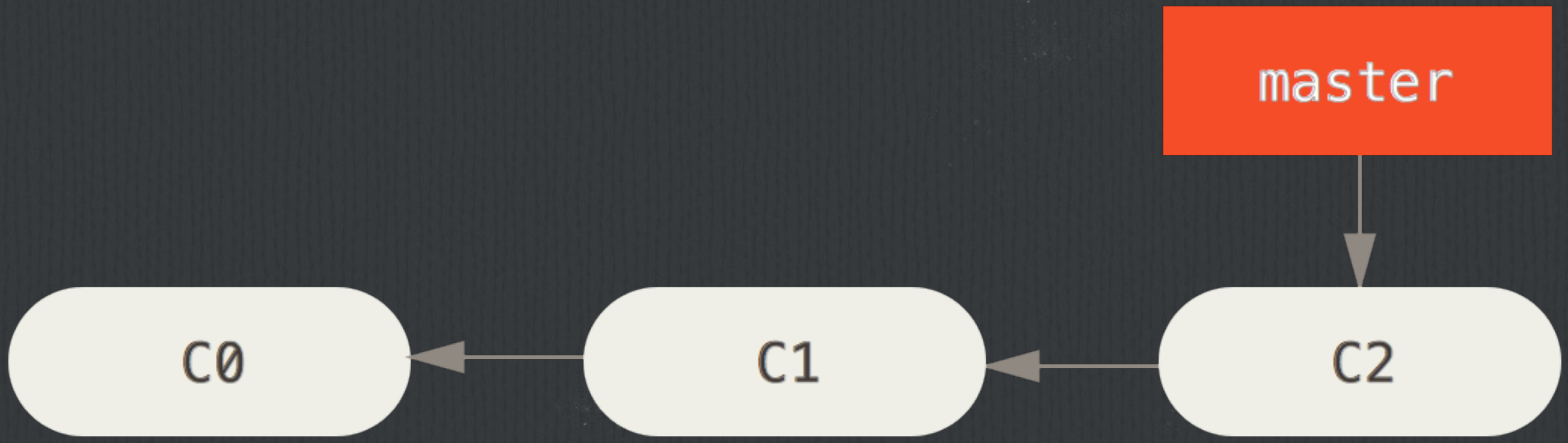
```
0      -  lolol
1  0   \   No newline at end of file
   1   +  lololol
2  2   \   No newline at end of file
```

Merging Branches

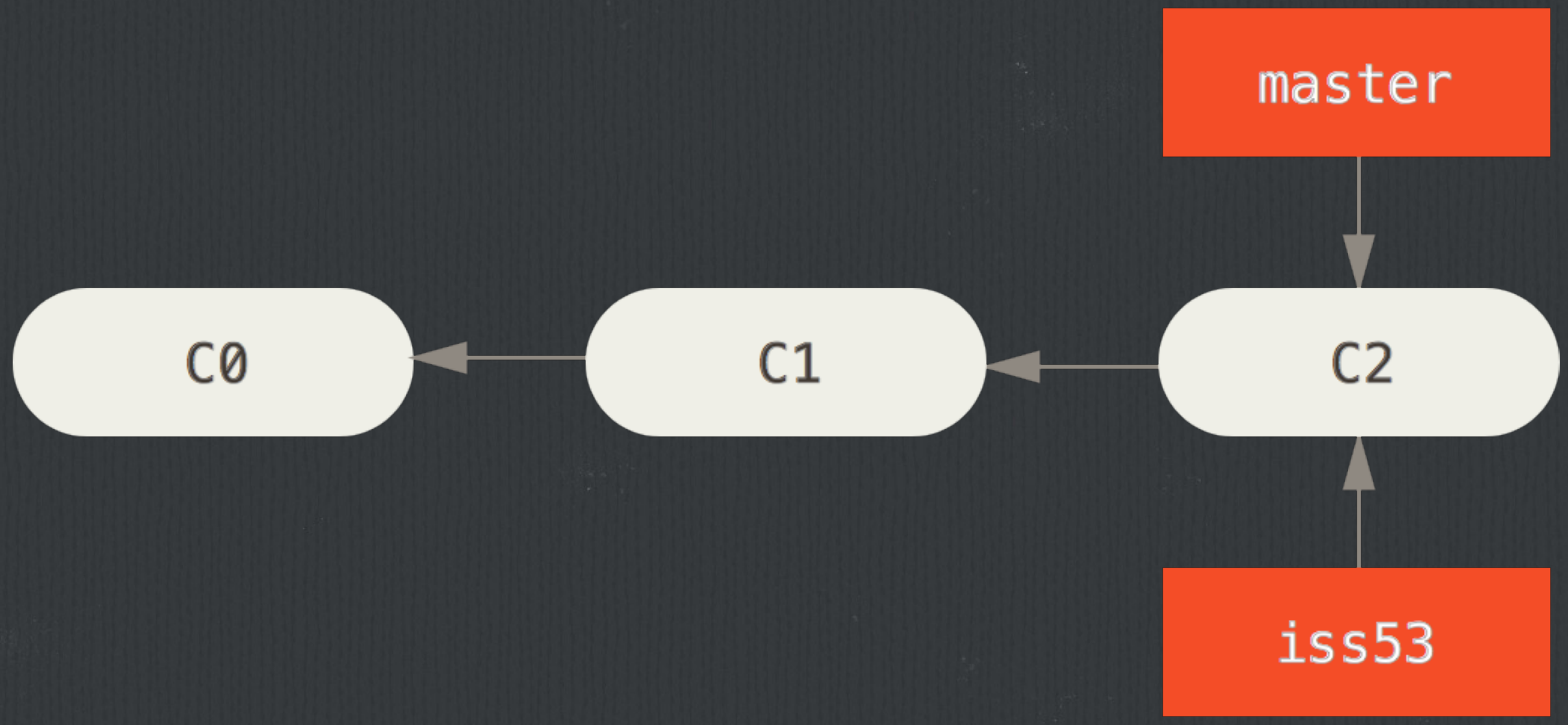
Basic Merging

- Remember: To merge a specific branch into our current branch we use: git merge <other_branch_name>

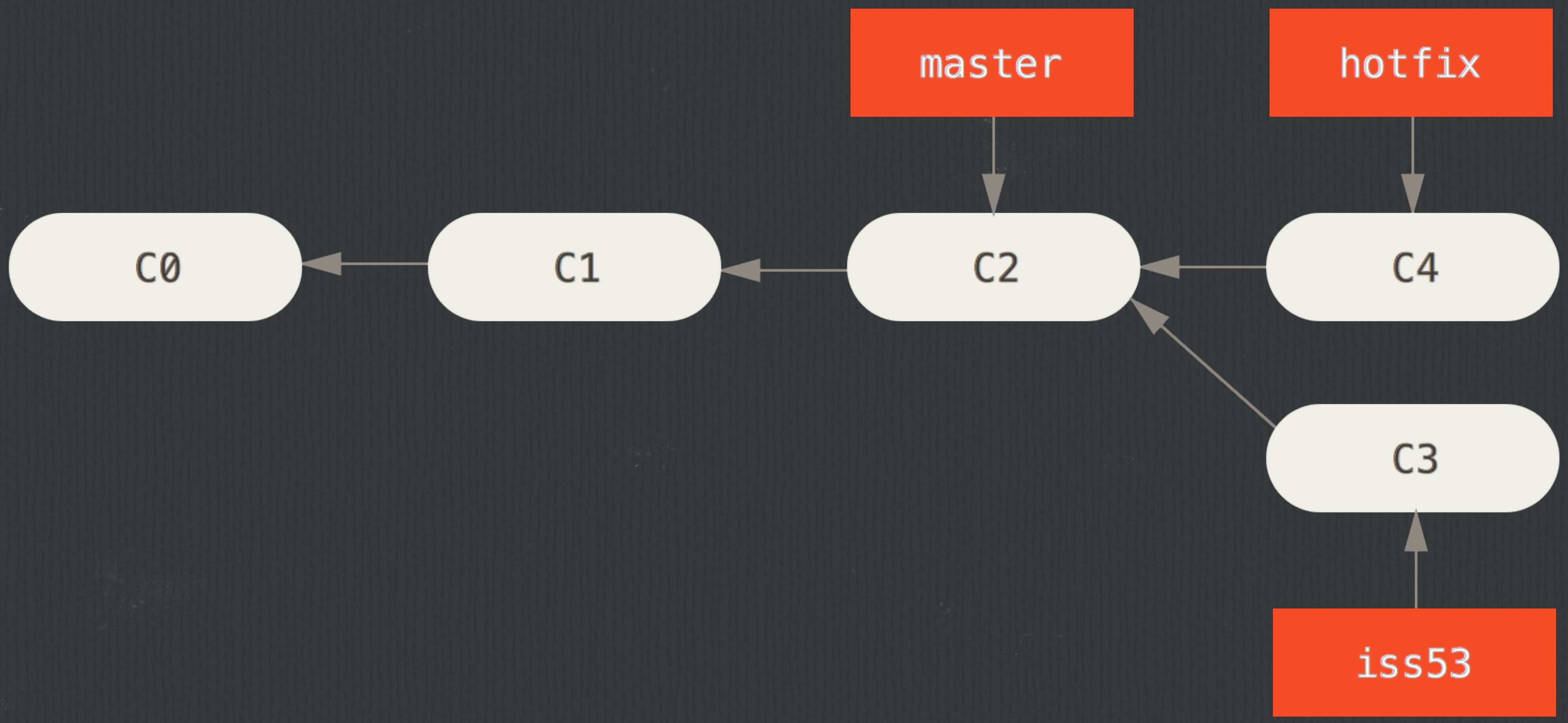
Basic Merging



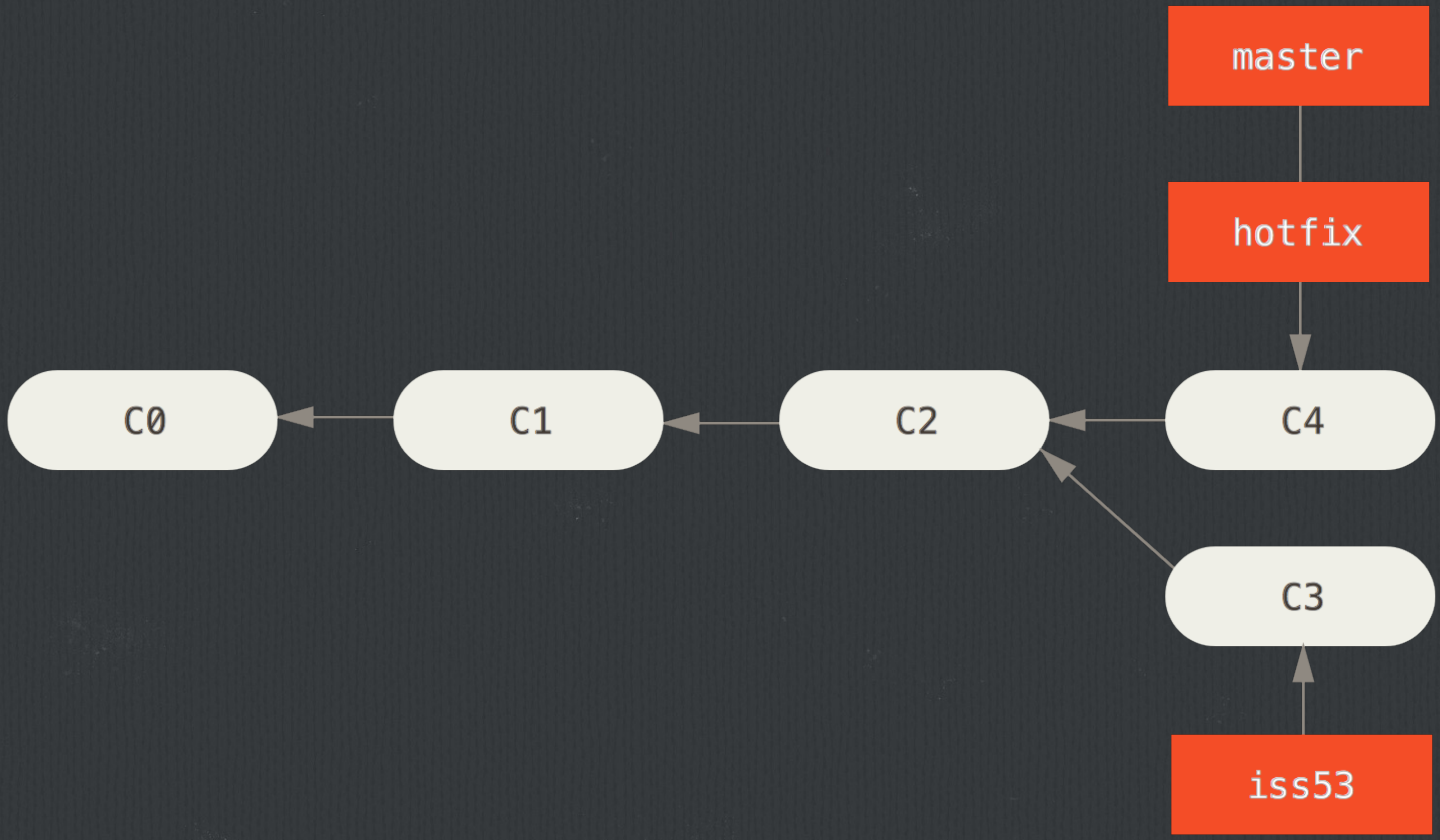
Basic Merging



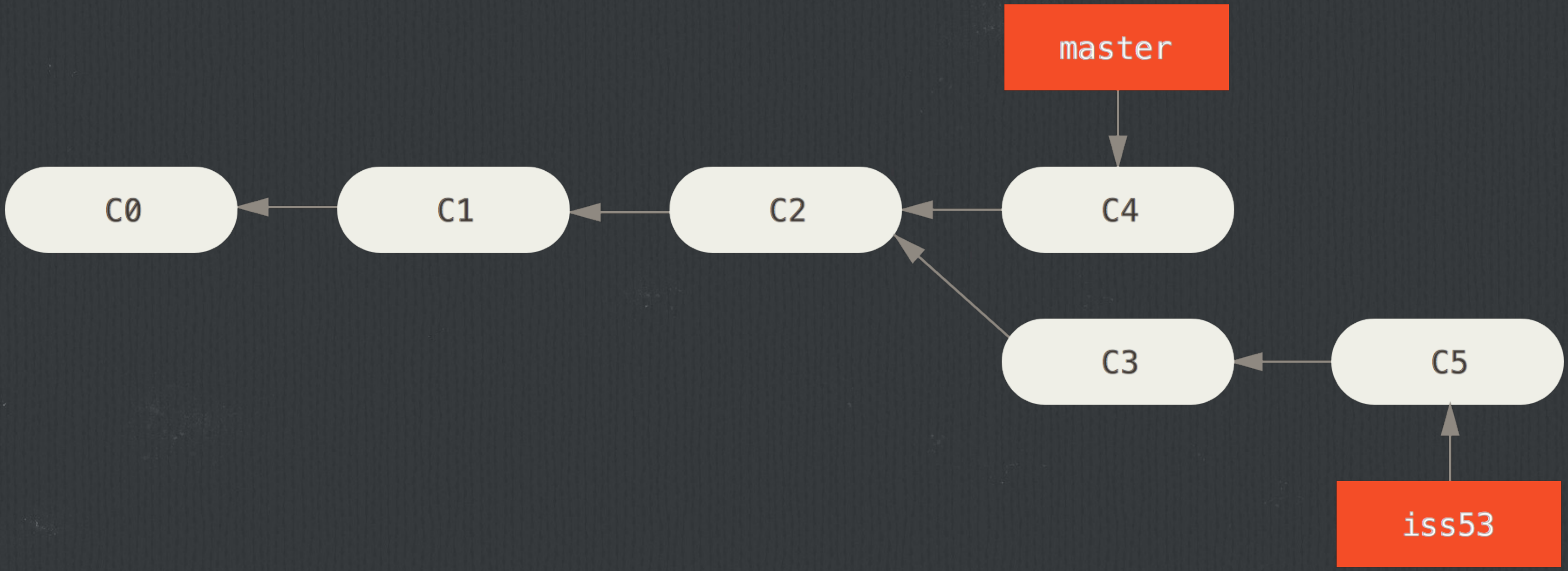
Basic Merging



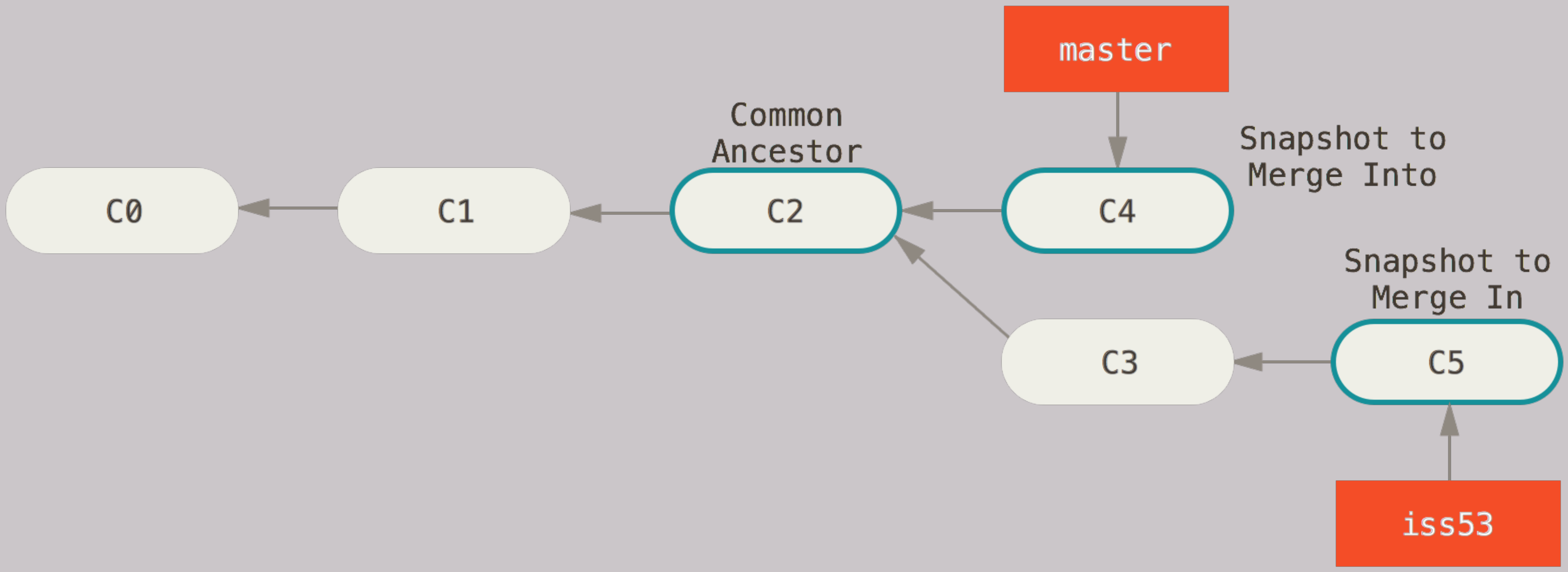
Basic Merging



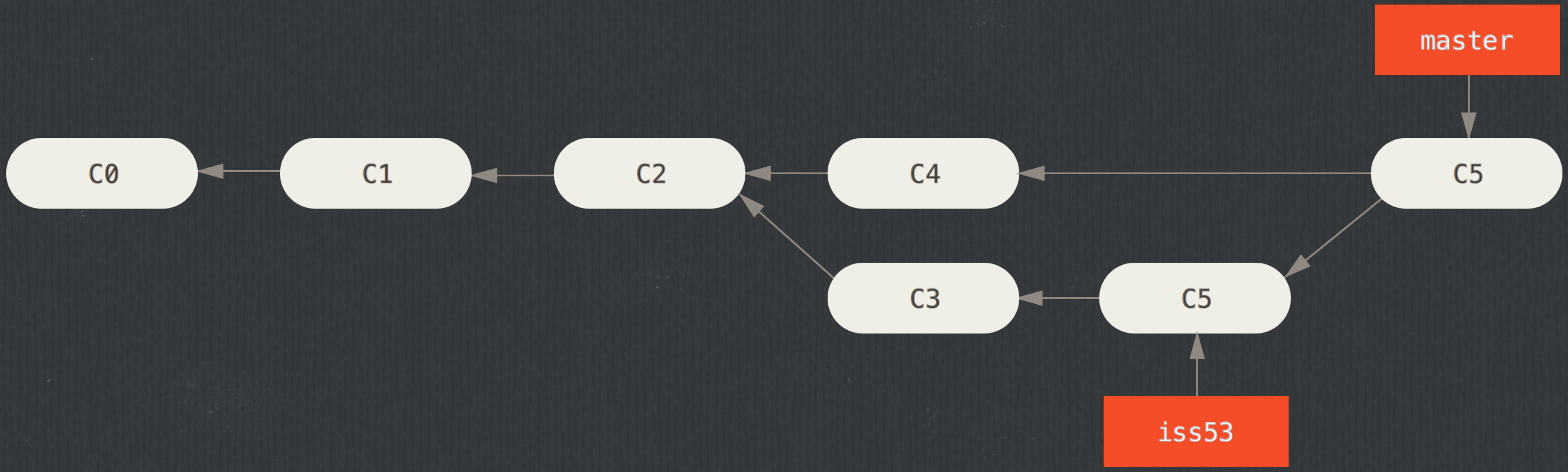
Basic Merging



Its time to merge!



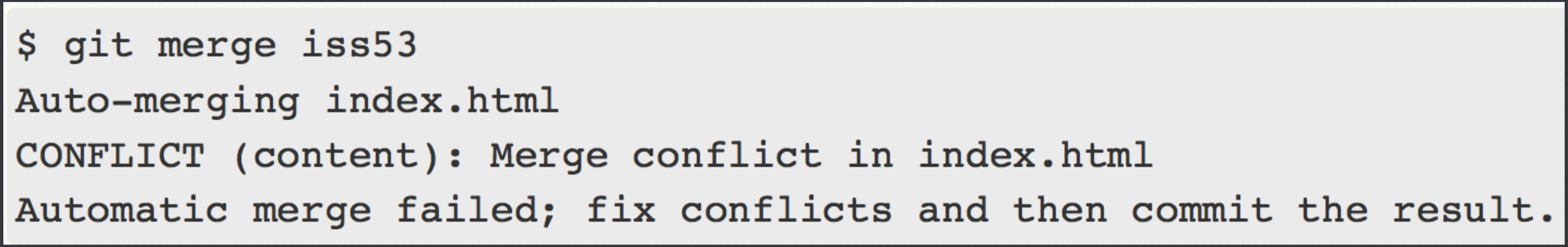
After three way merging …



Basic Merging

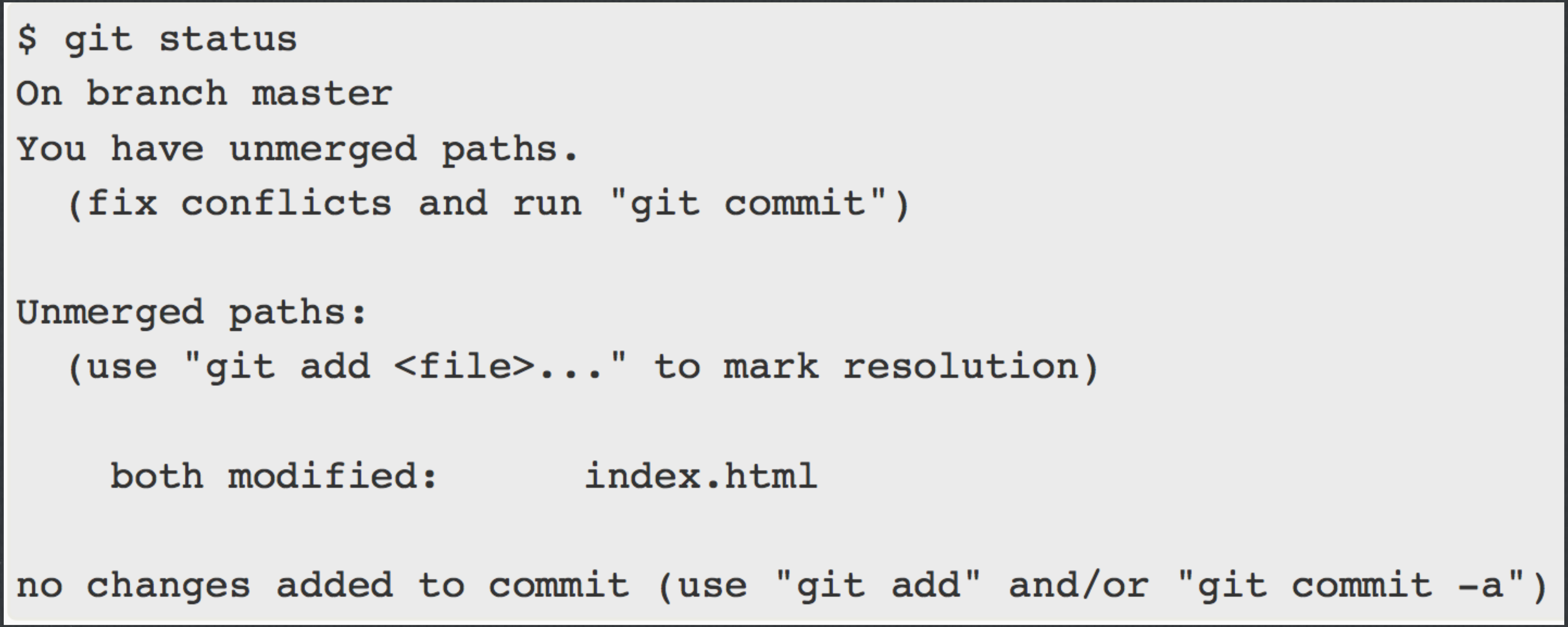
- We can **delete** a branch using the command: **git branch -d <branch_name>**

Basic Merge Conflicts
Conflict Notification



```
$ git merge iss53
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

Basic Merge Conflicts
Conflicts List



```
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")

Unmerged paths:
  (use "git add <file>..." to mark resolution)

    both modified:      index.html

no changes added to commit (use "git add" and/or "git commit -a")
```

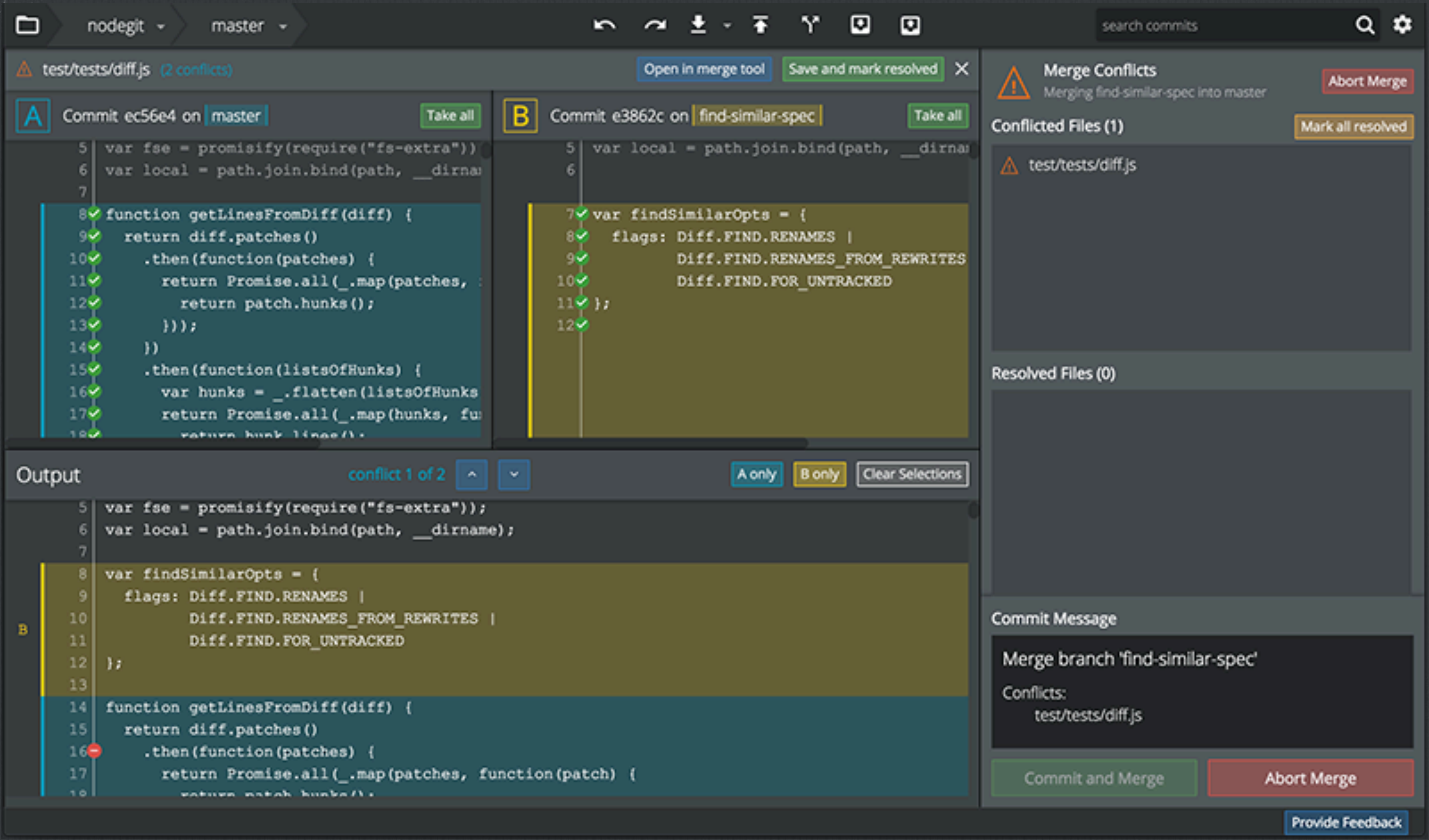
Basic Merge Conflicts
Conflict Visualization

```
<<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.com</div>
=======
<div id="footer">
 please contact us at support@github.com
</div>
>>>>>>> iss53:index.html
```

Basic Merge Conflicts
Resolving the conflict

```
<div id="footer">
please contact us at email.support@github.com
</div>
```

Graphic Diff Tools



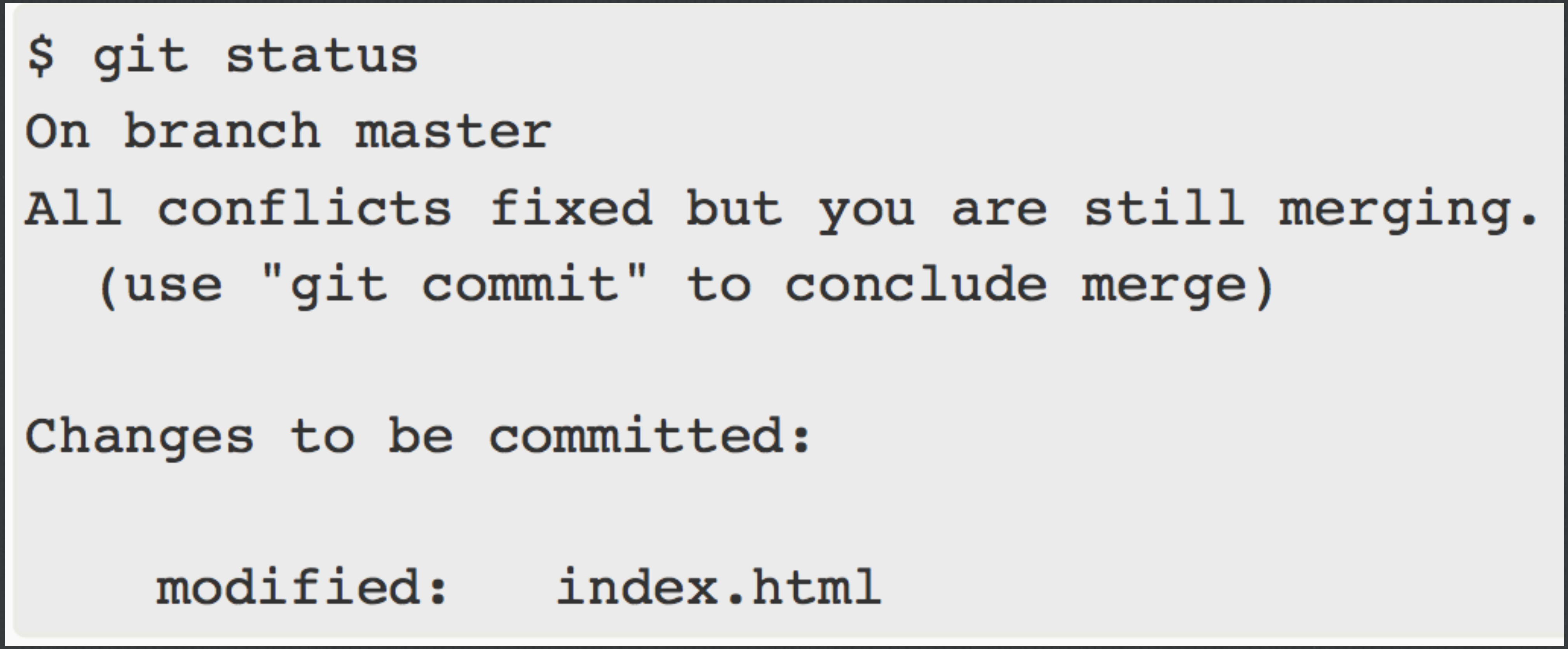
```
var fse = promisify(require("fs-extra"))
var local = path.join.bind(path, __dirna

function getLinesFromDiff(diff) {
  return diff.patches()
    .then(function(patches) {
      return Promise.all(_.map(patches,
        return patch.hunks();
      }));
    })
    .then(function(listsOfHunks) {
      var hunks = _.flatten(listsOfHunks
      return Promise.all(_.map(hunks, fu
        return hunk.lines();
```

```
var local = path.join.bind(path, __dirna

var findSimilarOpts = {
  flags: Diff.FIND.RENAMES |
         Diff.FIND.RENAMES_FROM_REWRITES
         Diff.FIND.FOR_UNTRACKED
};
```

```
var fse = promisify(require("fs-extra"));
var local = path.join.bind(path, __dirname);

var findSimilarOpts = {
  flags: Diff.FIND.RENAMES |
         Diff.FIND.RENAMES_FROM_REWRITES |
         Diff.FIND.FOR_UNTRACKED
};

function getLinesFromDiff(diff) {
  return diff.patches()
    .then(function(patches) {
      return Promise.all(_.map(patches, function(patch) {
        return patch.hunks();
```

```
Merge branch 'find-similar-spec'

Conflicts:
	test/tests/diff.js
```

Basic Merge Conflicts
After solving all conflicts



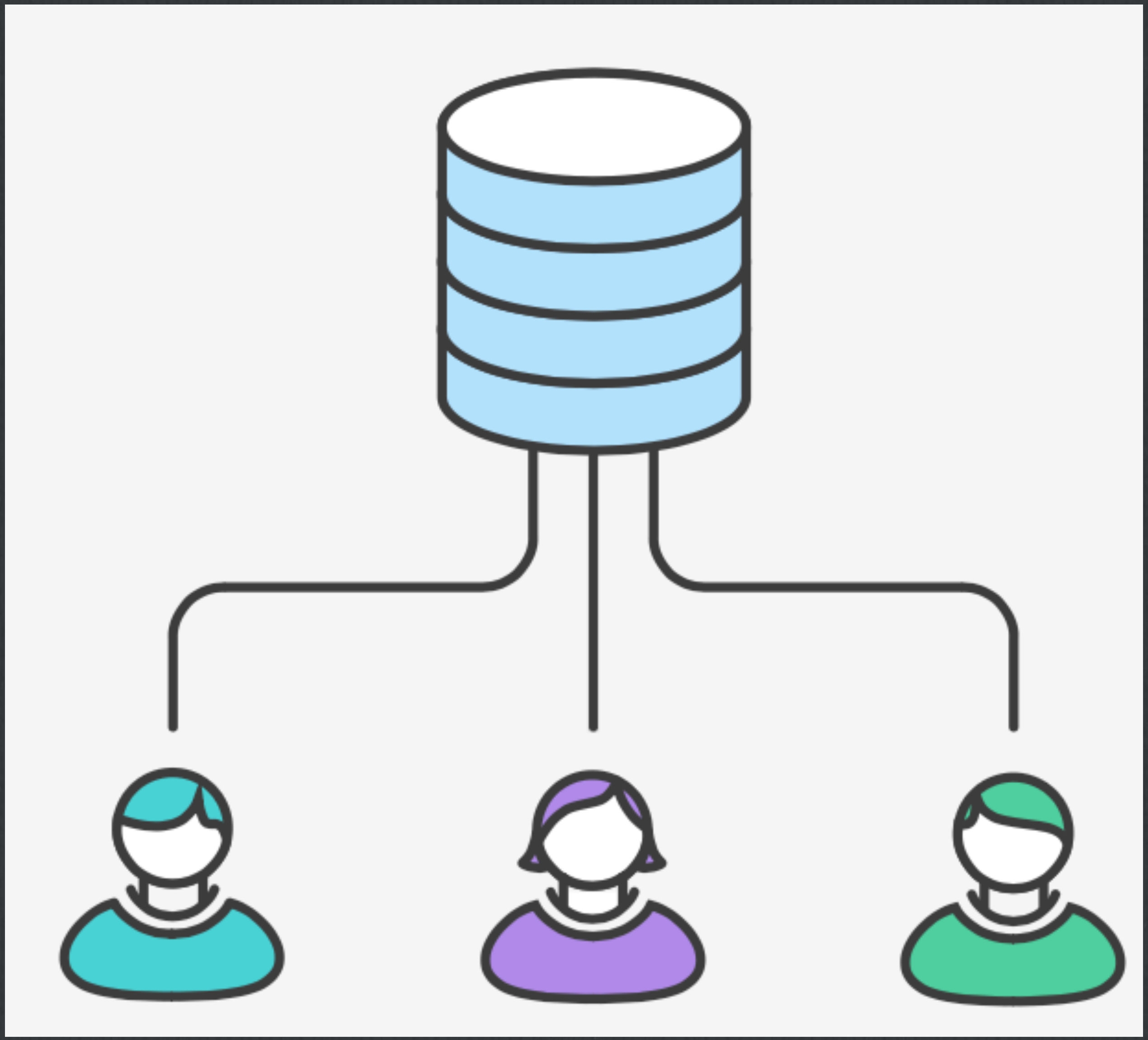
```
$ git status
On branch master
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:

    modified:   index.html
```

Basic Merge Conflicts Default Commit Message

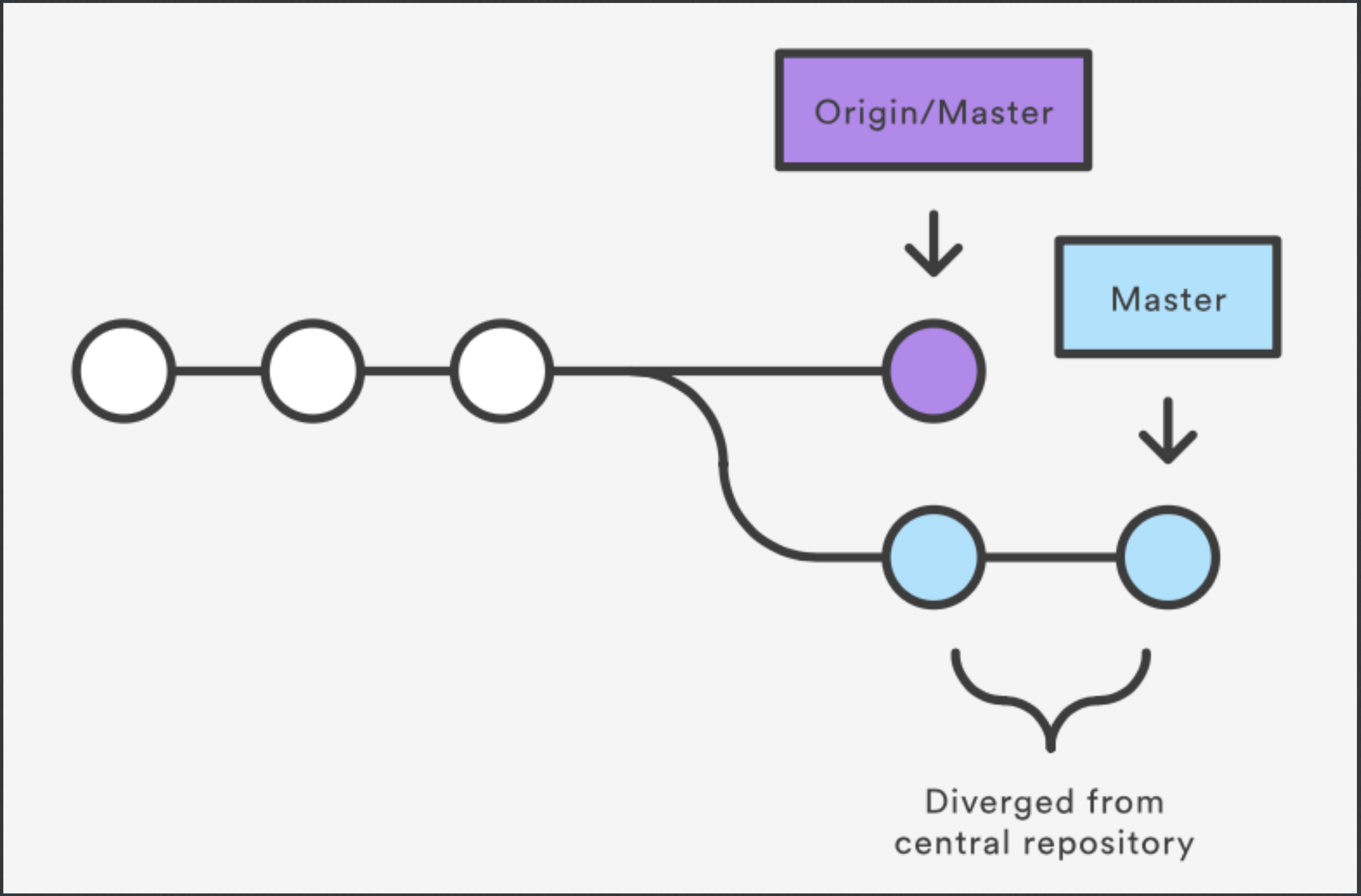
```
Merge branch 'iss53'

Conflicts:
    index.html
#
# It looks like you may be committing a merge.
# If this is not correct, please remove the file
#       .git/MERGE_HEAD
# and try again.


# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# All conflicts fixed but you are still merging.
#
# Changes to be committed:
#       modified:   index.html
#
```

Branching Strategies (Workflows)

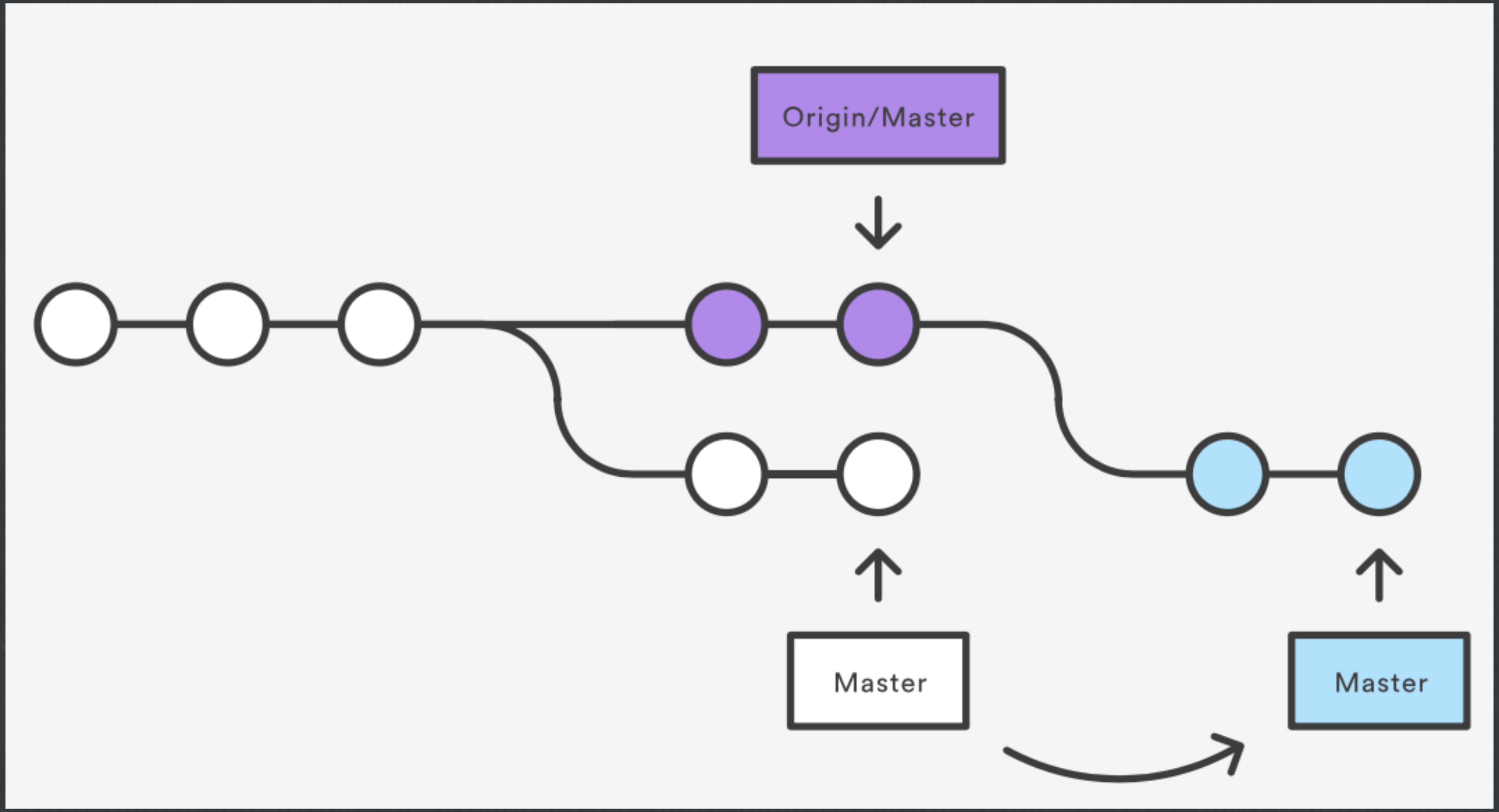
Centralized Workflow



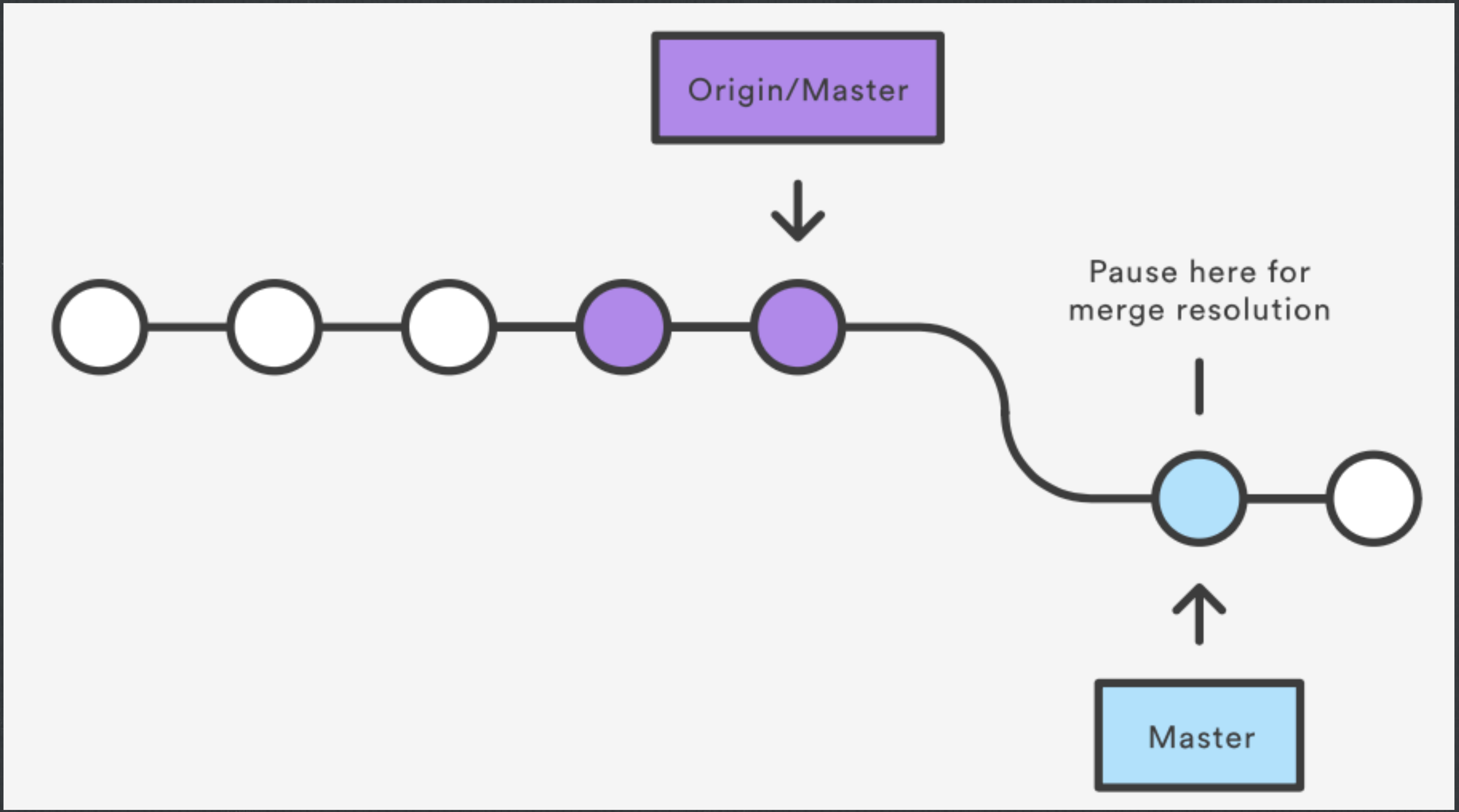
Centralized Workflow



Centralized Workflow



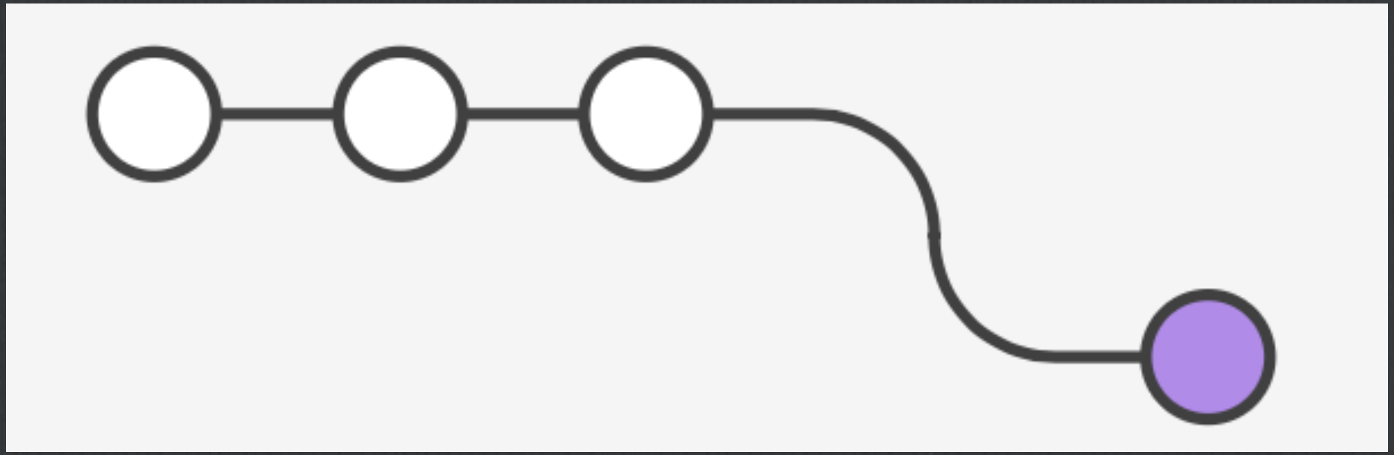
Centralized Workflow



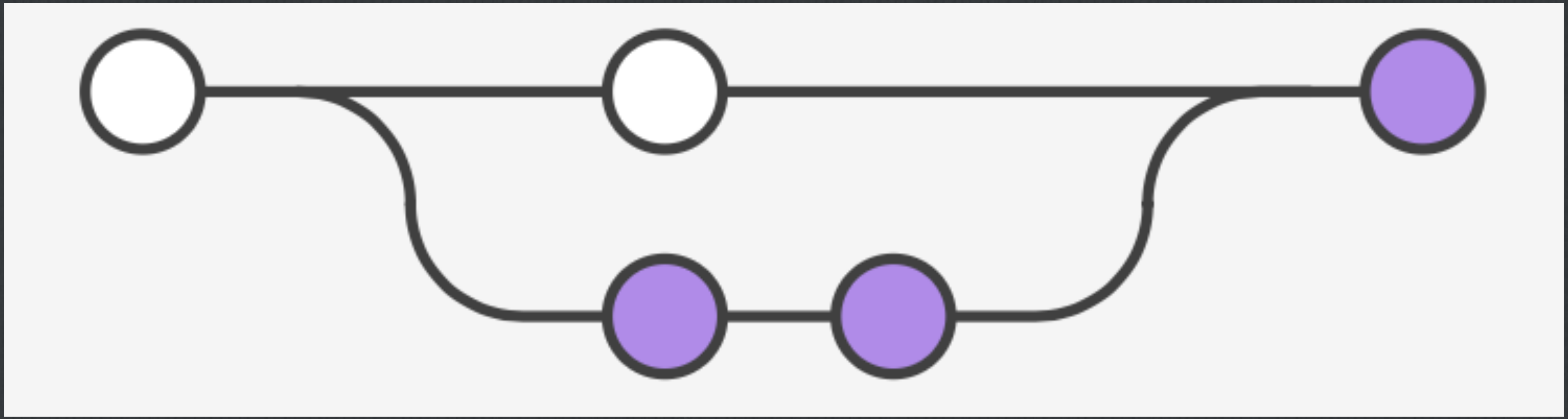
Feature Branch Workflow
Pull Request

- A pull request is a special type of request made to a repo admin, asking him to pull the changes from a specific branch into a objective branch (usually an important base branch as master)
- The developer who is making the request, must pull all the changes from the objective branch and resolve all the merge conflicts before issuing the PR

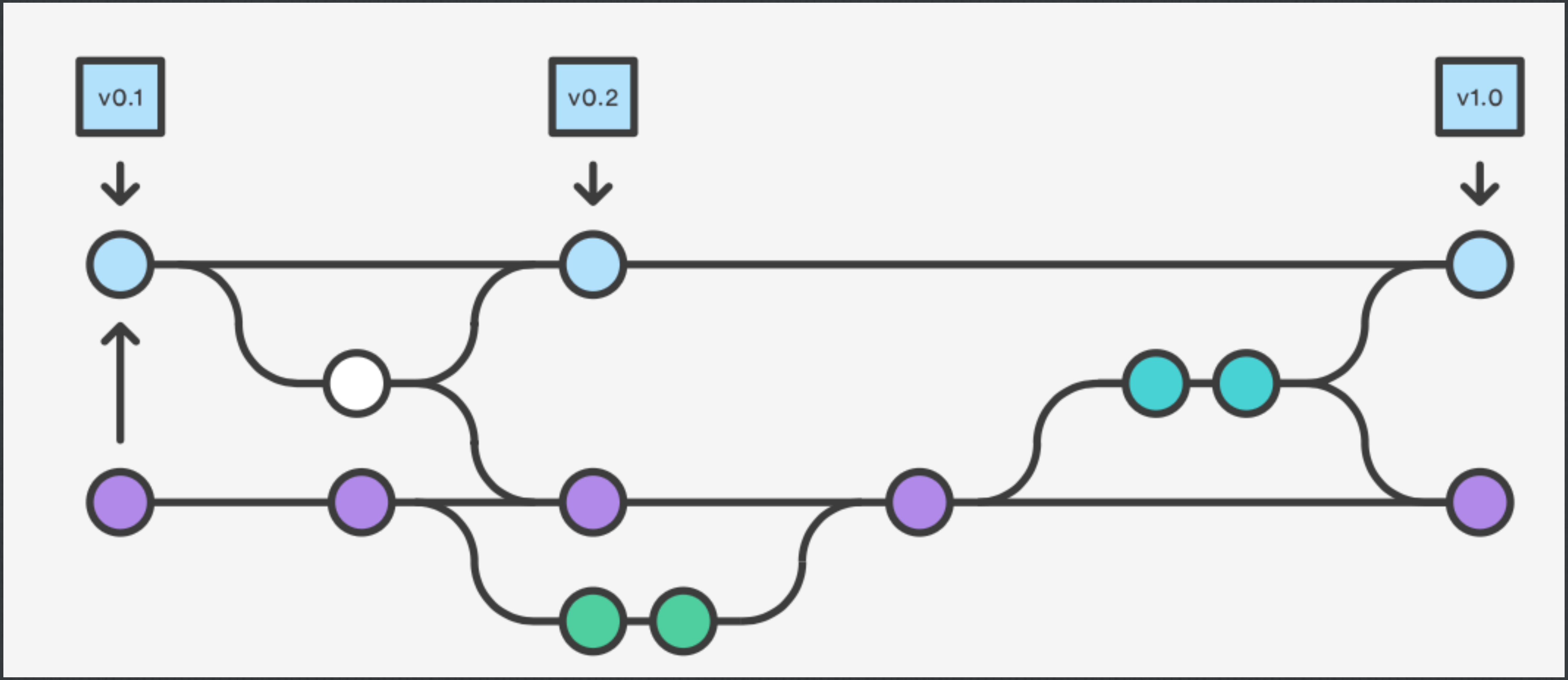
Feature Branch Workflow



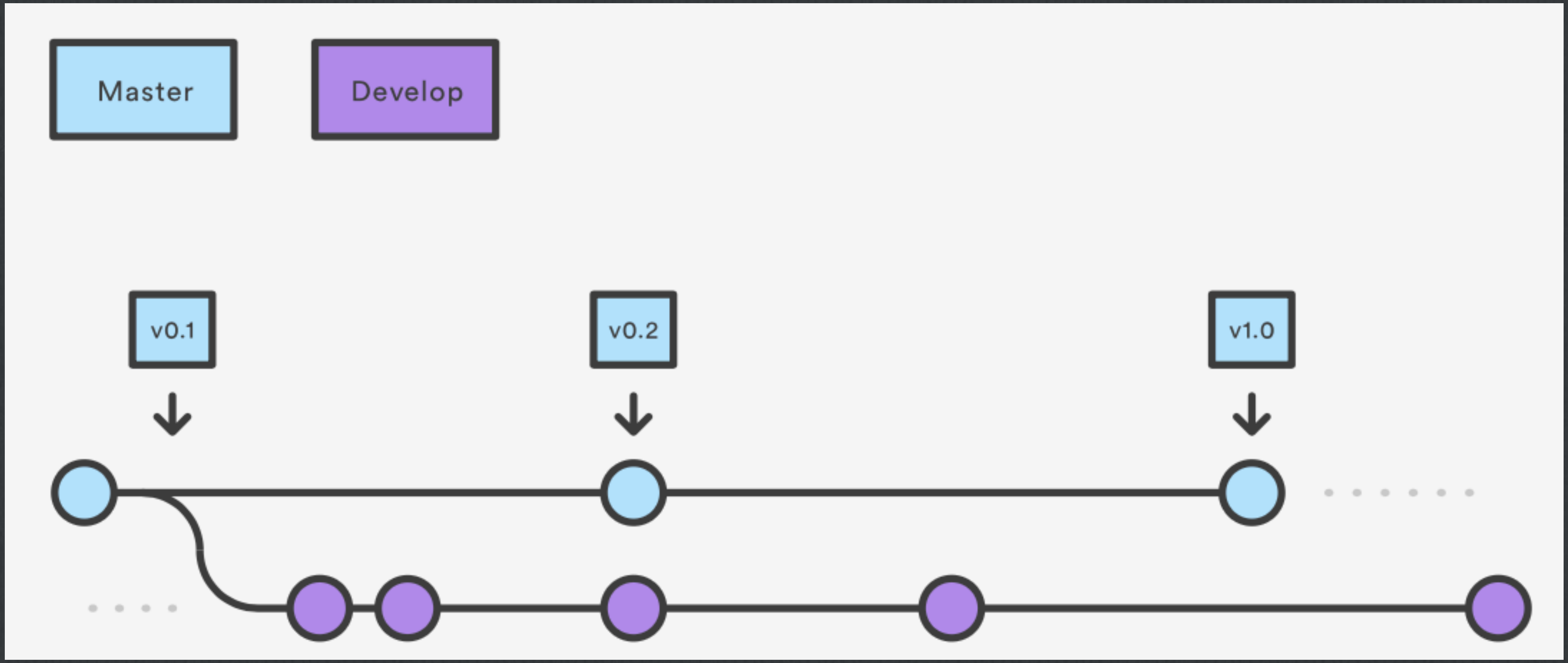
After the PR is accepted:



Git Workflow General View



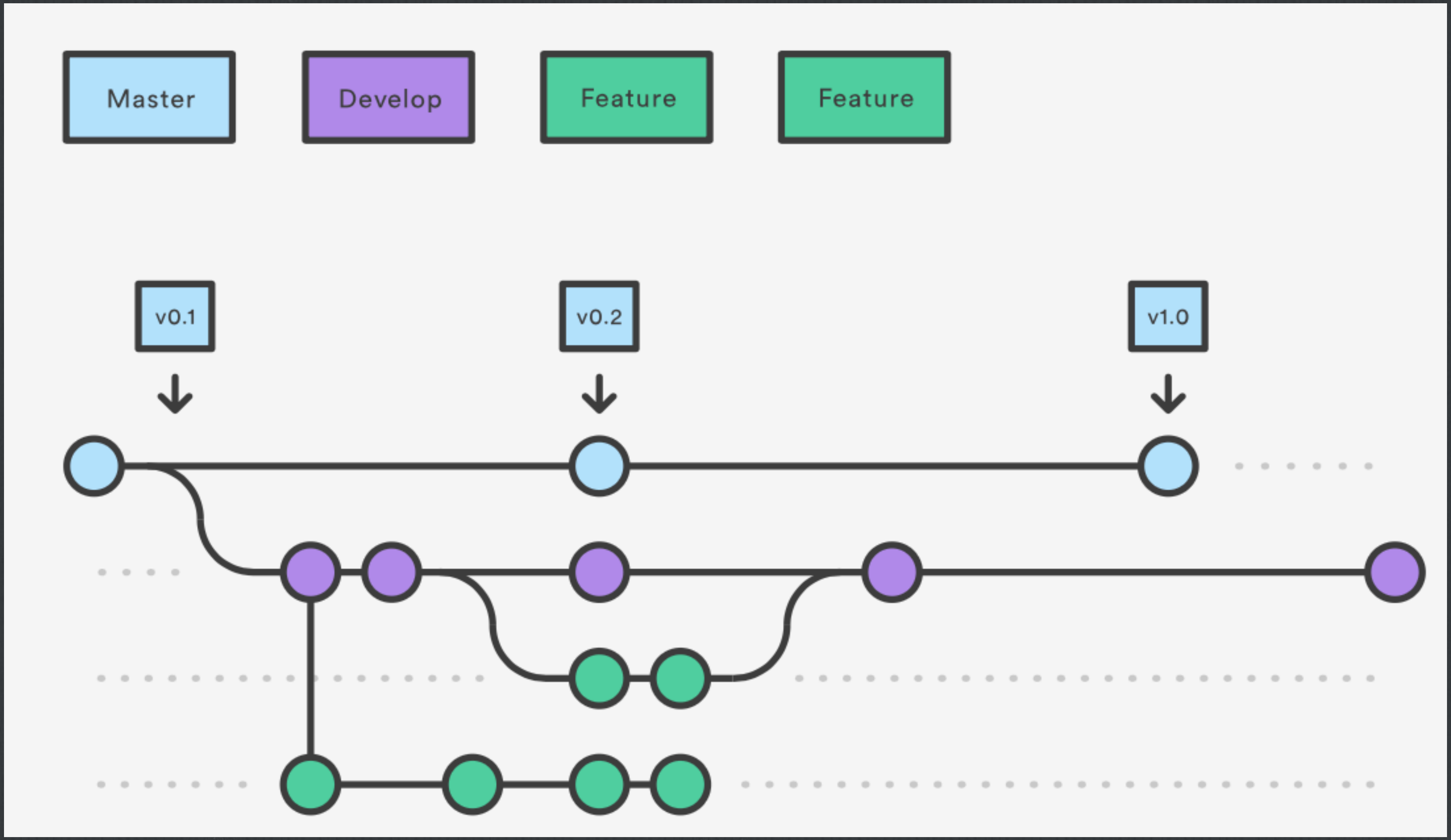
Git Workflow
Develop and Master



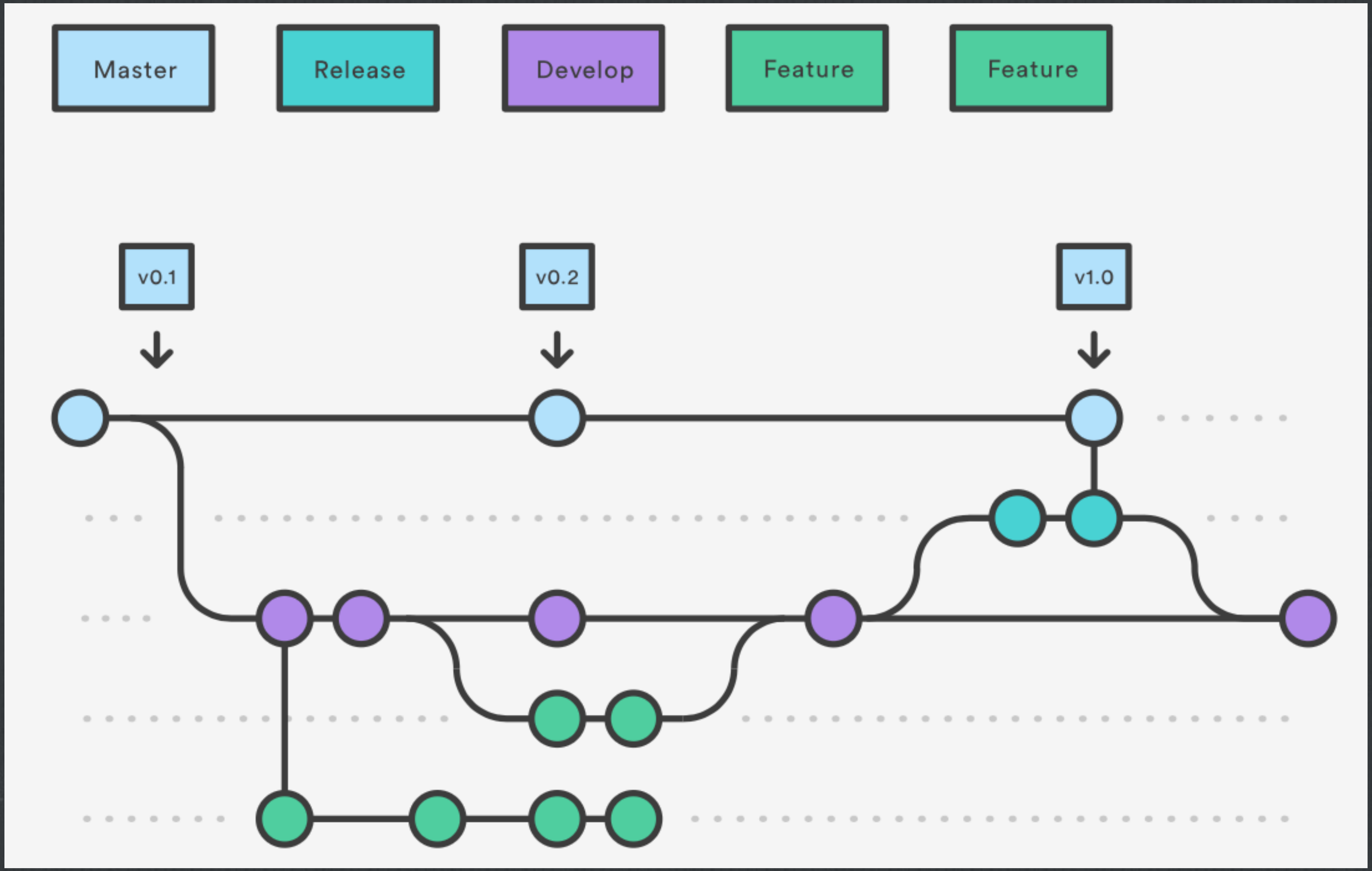
Develop: Integration Branch for features

Master: Official Release History

Git Workflow
Feature Branches

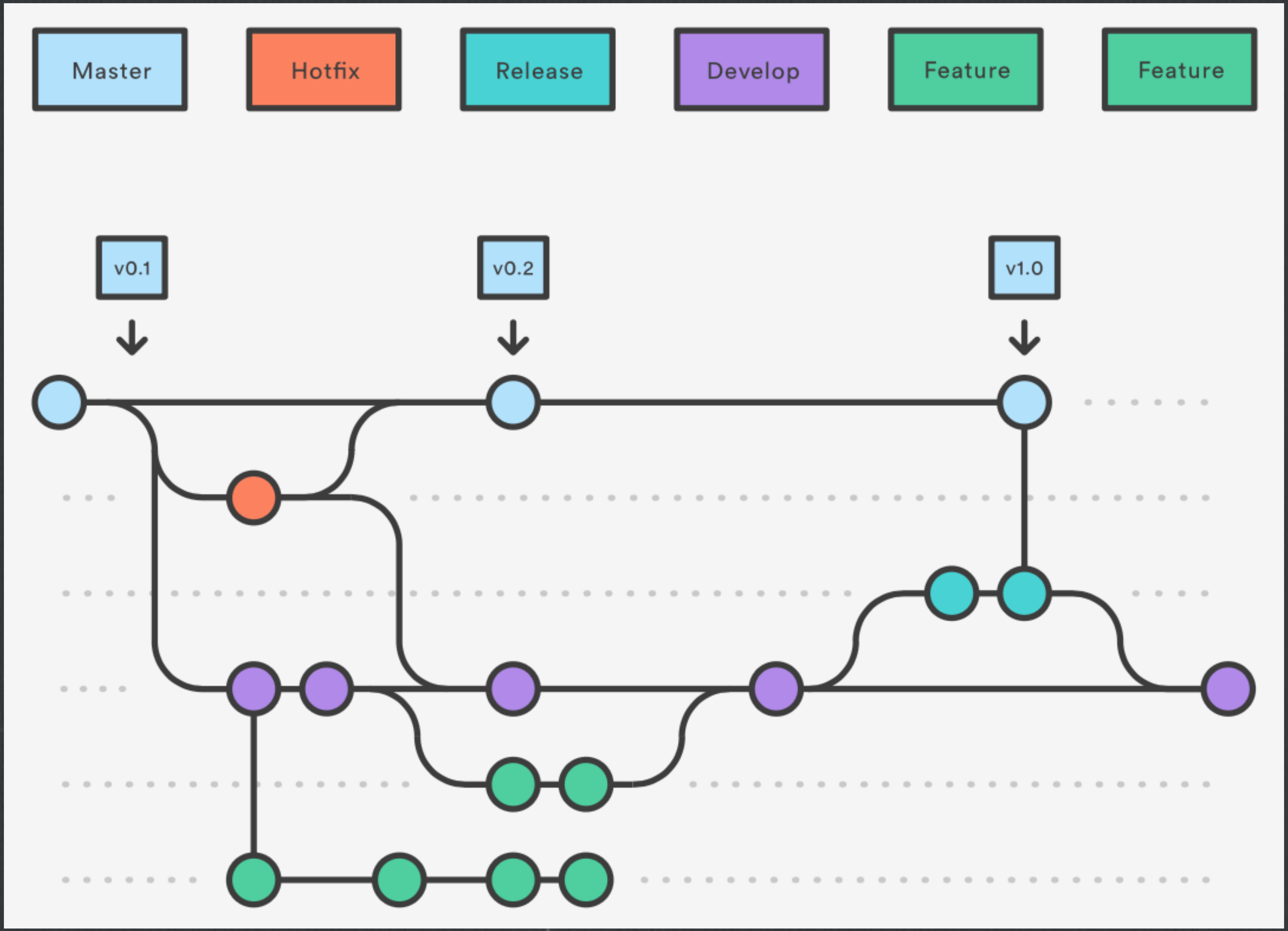


Git Workflow Release Branches

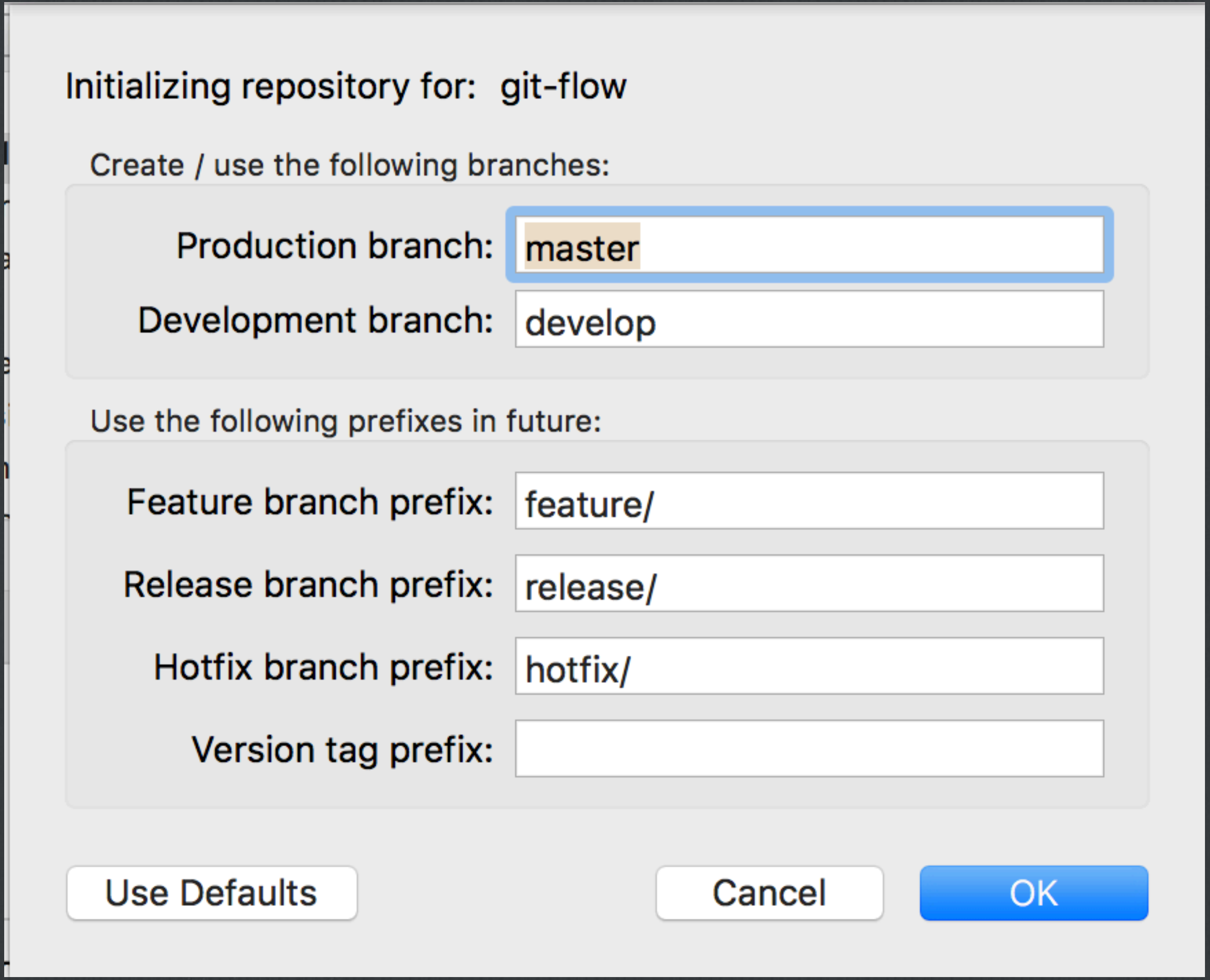


Release - Oriented Tasks (Documentation, Bug Fixes)

Git Workflow
HotFixes (Maintenance) Branches



Git Workflow in SourceTree



Initializing repository for: git-flow

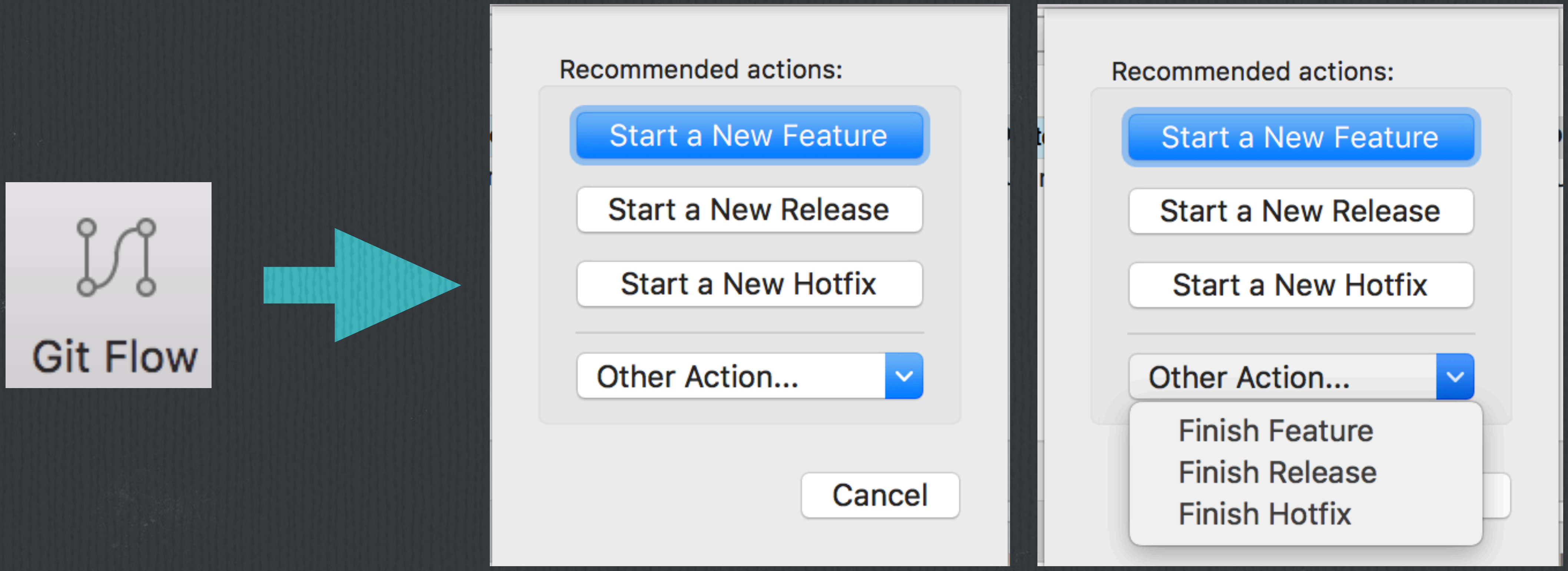
Create / use the following branches:

Production branch:	master
Development branch:	develop

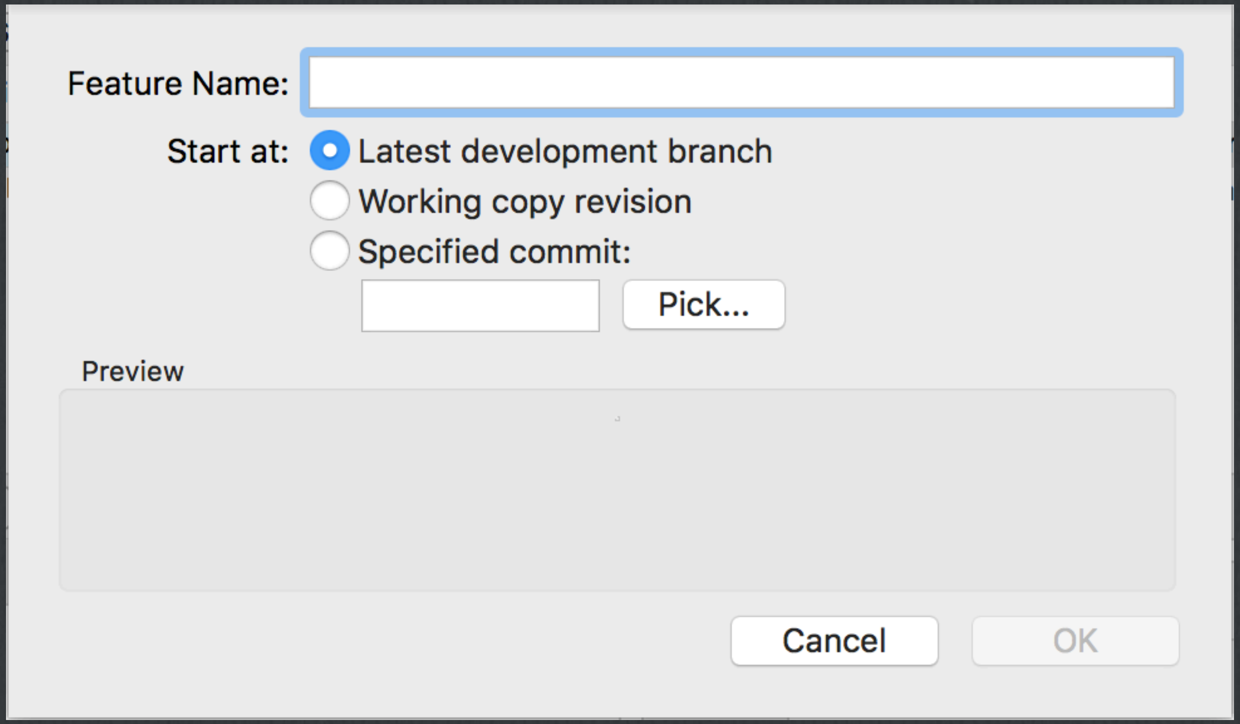
Use the following prefixes in future:

Feature branch prefix:	feature/
Release branch prefix:	release/
Hotfix branch prefix:	hotfix/
Version tag prefix:	

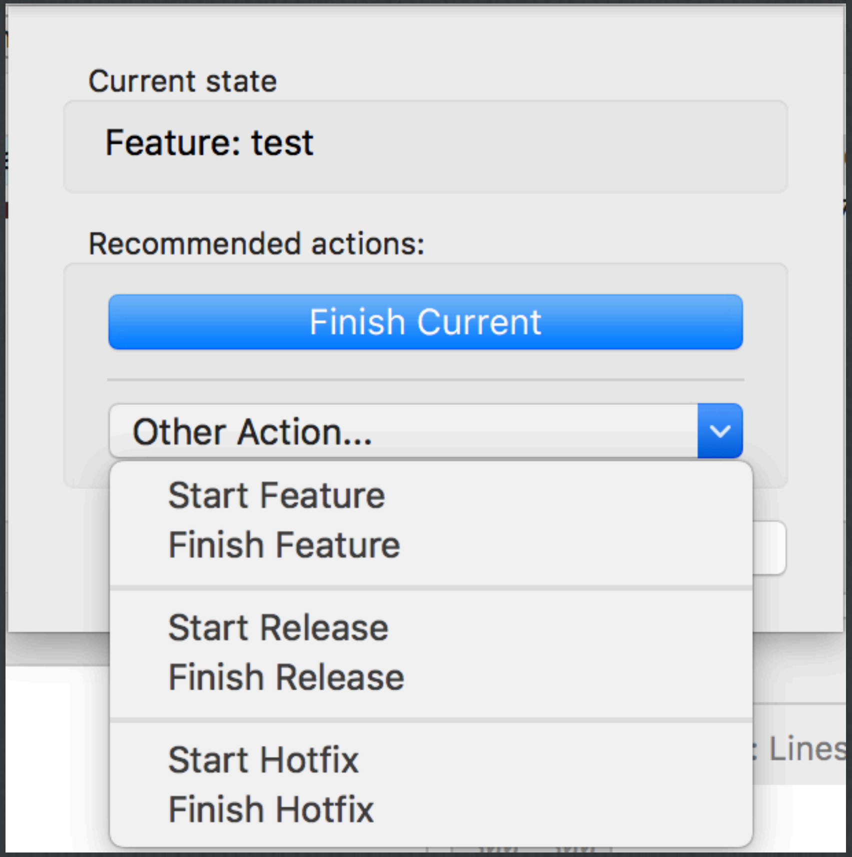
Git Workflow in SourceTree



Git Workflow in SourceTree


Feature Name:

Start at: [✓ Latest development branch] Working copy revision Specified commit:


Current state

Feature: test

Recommended actions:

Homework

- ☐ Final Project Idea Submission
- ☐ Github Profile Creation
- ☐ GitHub Organization Creation
- ☐ Project Repo Creation
- ☐ First Commit by each one

About Friday’s Revisions

- ☐ Friday's morning
- ☐ Schedule
- ☐ 20 min per group
- ☐ Last committed work - Thursday at 5 pm