
An Introduction to Git and Github

Poruri Sai Rahul,
Software Developer,
Enthought Inc.

Why?

-
- Why?
 - What?
 - Git
 - GitHub
 - How?
 - Basic
 - Advanced

@rahulporuri

Why?

- Made a change to code, realised it was a mistake and wanted to revert back?
 - Lost code or had a backup that was too old?
 - Had to maintain multiple versions of a product?
 - Wanted to see the difference between two (or more) versions of your code?
 - Wanted to prove that a particular change broke or fixed a piece of code?
-

-
- Why?
 - What?
 - Git
 - GitHub
 - How?
 - Basic
 - Advanced

@rahulporuri

Why? (continued)

- Wanted to review the history of some code?
- Wanted to submit a change to someone else's code?
- Wanted to share your code, or let other people work on your code?
- Wanted to see how much work is being done, and where, when and by whom?
- Wanted to experiment with a new feature without interfering with working code?

Source - <https://goo.gl/GoW6My>

What?

-
- Why?
 - What?
 - **Git**
 - GitHub
 - How?
 - Basic
 - Advanced

What? - Git

- One of many Distributed Version Control Software
- [Hg](#) and [SVN](#) are two other popular alternatives

@rahulporuri

-
- Why?
 - What?
 - Git
 - **GitHub**
 - How?
 - Basic
 - Advanced

What? - Github

- For a beginner - it's a place to backup all of your code
- For a professional - it's a place to collaboratively develop software

@rahulporuri

How?

-
- Why?
 - What?
 - Git
 - GitHub
 - **How?**
 - Basic
 - Advanced

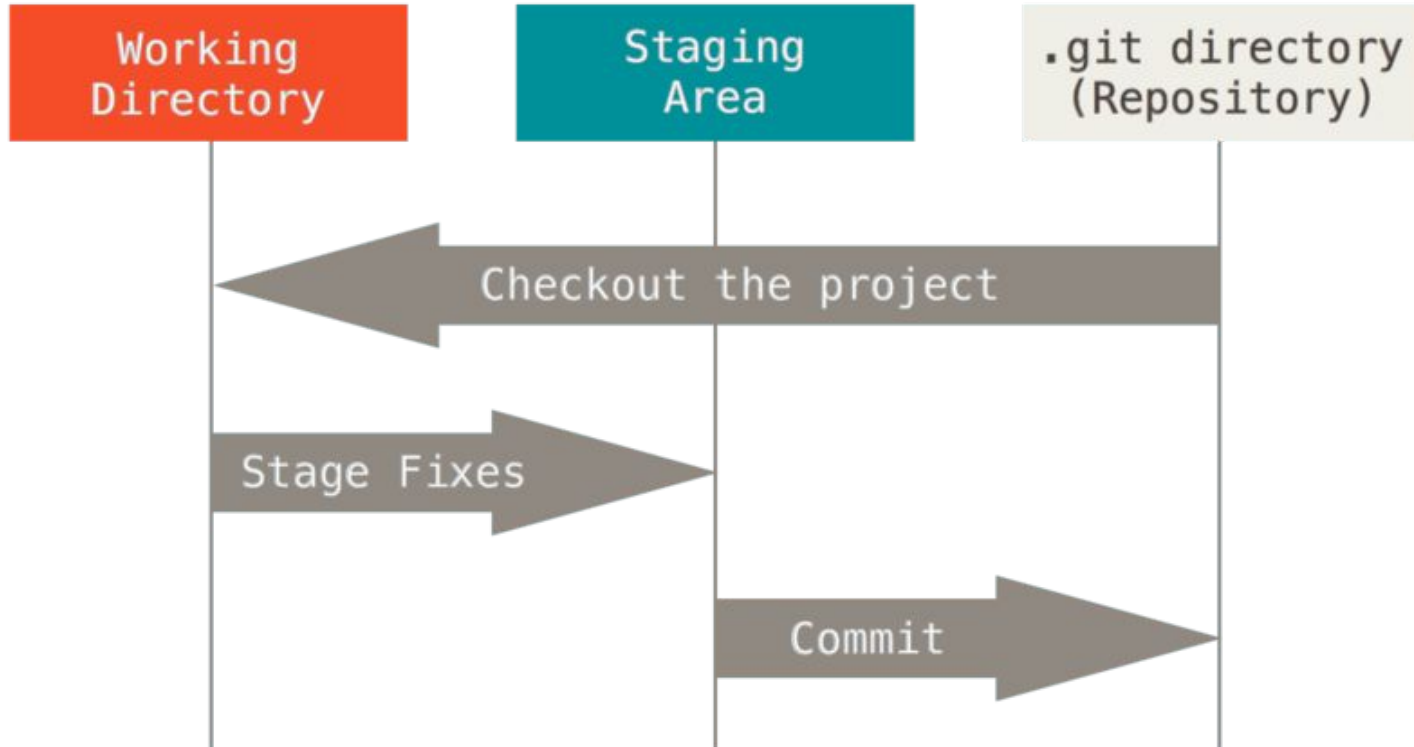
How? - Workflows

- A Basic workflow, which introduces commands for single user to get up and running with Git.
- An advanced workflow, which introduces commands that allow multiple users to work on the same repository(?).

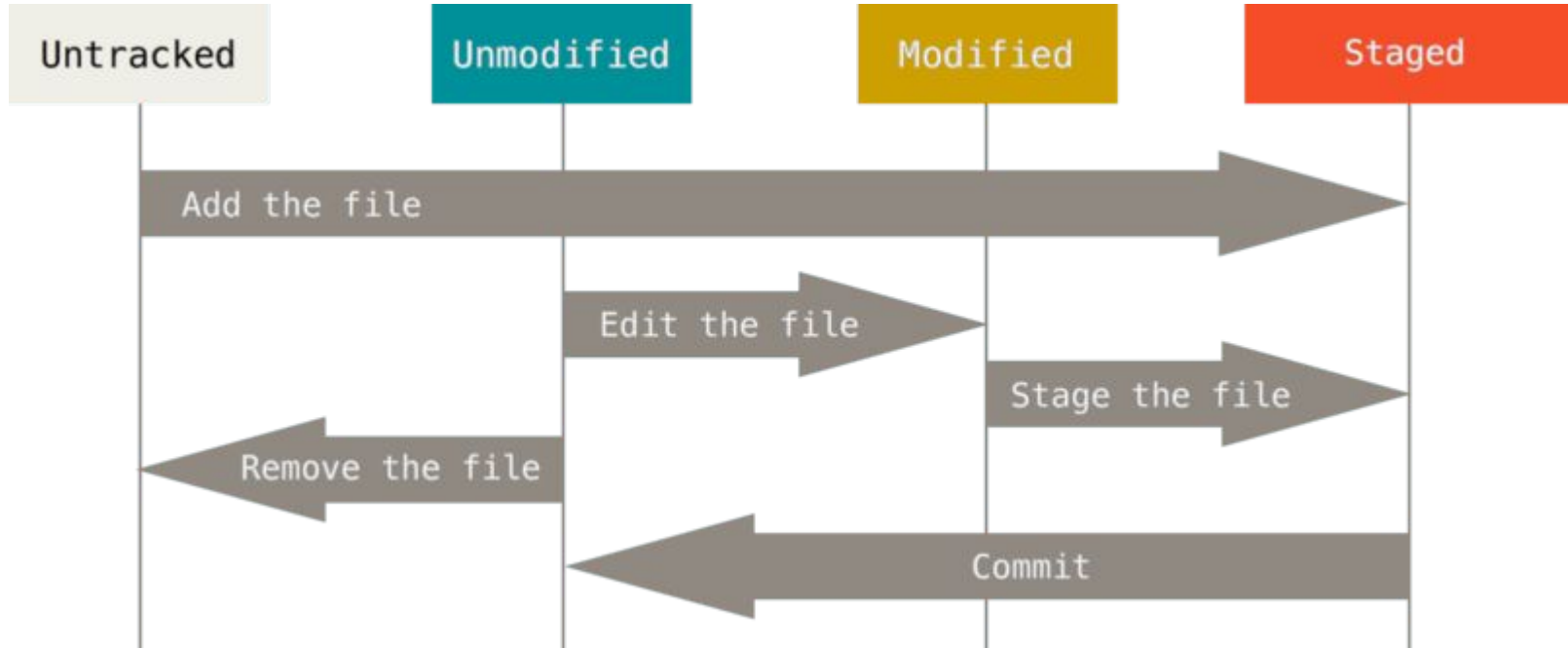
@rahulporuri

repository - a directory being tracked/versioned using Git.

How? - Basic Workflow



How? - Basic Workflow (continued)



-
- Why?
 - What?
 - Git
 - GitHub
 - How?
 - Basic
 - Advanced

How? - Basic Workflow

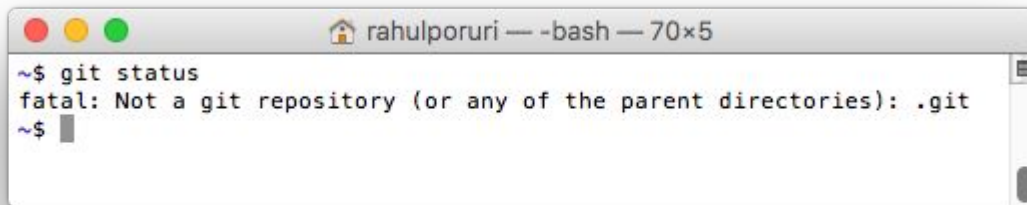
- `git add 'file-name'`
- `git commit -m "commit message"`
- `git push`

@rahulporuri

-
- Why?
 - What?
 - Git
 - GitHub
 - How?
 - Basic
 - Advanced

First mistake - Not a Git repository

The repository needs to be initialized - Git needs to be told to track changes in this repository.

A terminal window with a title bar that reads 'rahulporuri — -bash — 70x5'. The window contains the following text: '~\$ git status', 'fatal: Not a git repository (or any of the parent directories): .git', and '~\$' followed by a cursor.

```
~$ git status
fatal: Not a git repository (or any of the parent directories): .git
~$
```

@rahulporuri

-
- Why?
 - What?
 - Git
 - GitHub
 - How?
 - Basic
 - Advanced

@rahulporuri

Second mistake - defaults aren't set

Git needs to know your name and email, which are stored in each commit. This helps with the blame game.

```
ab-tmp (master)$ git commit
[master (root-commit) cfa7b85] ealsdjaldfkj
Committer: Poruri Sai Rahul <rahulporuri@Prashants-MacBook-Pro.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:
```

```
git config --global user.name "Your Name"
git config --global user.email you@example.com
```

After doing this, you may fix the identity used for this commit with:

```
git commit --amend --reset-author
```

```
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 me
```

-
- Why?
 - What?
 - Git
 - GitHub
 - How?
 - Basic
 - Advanced

Third mistake - Origin hasn't been set

Git needs to be told what the origin is - where to push the changes to. The origin can be set to anything - a GitHub/GitLab/BitBucket repository or your own personal server.

```
[ab-tmp (master)]$ git push
fatal: No configured push destination.
Either specify the URL from the command-line or configure a remote repository using

    git remote add <name> <url>

and then push using the remote name

    git push <name>
```

-
- Why?
 - What?
 - Git
 - GitHub
 - How?
 - Basic
 - Advanced

Helpful commands

- `git clone url` and `git fork url`
- `git log`
- `git show 'commit-hash'`
- `git commit --amend`
- `git reset HEAD^`
- `git diff 'file-name'`
- `git stash`
- Add a `.gitignore` file to your repository.
- Add `git-completion` and `git-prompt` for autocompletion and fancy prompt.

@rahulporuri

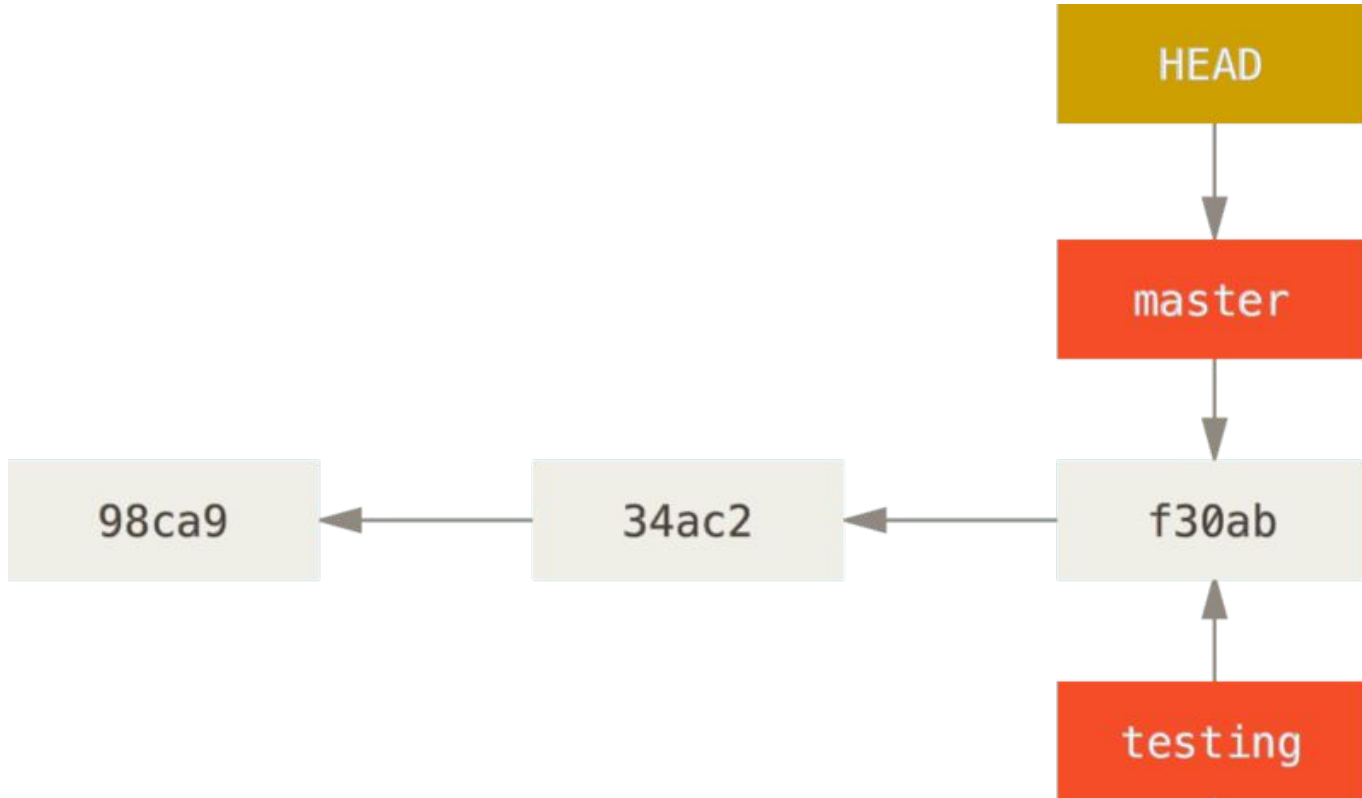
-
- Why?
 - What?
 - Git
 - GitHub
 - How?
 - Basic
 - **Advanced**

Advanced Workflow

- `git branch 'branch-name'`
- `git checkout 'branch-name'`
- `git add`
- `git commit`
- `git push origin 'branch-name'`
- `git merge 'branch-name'`

What? - Advanced Workflow (continued)

When we create a new branch



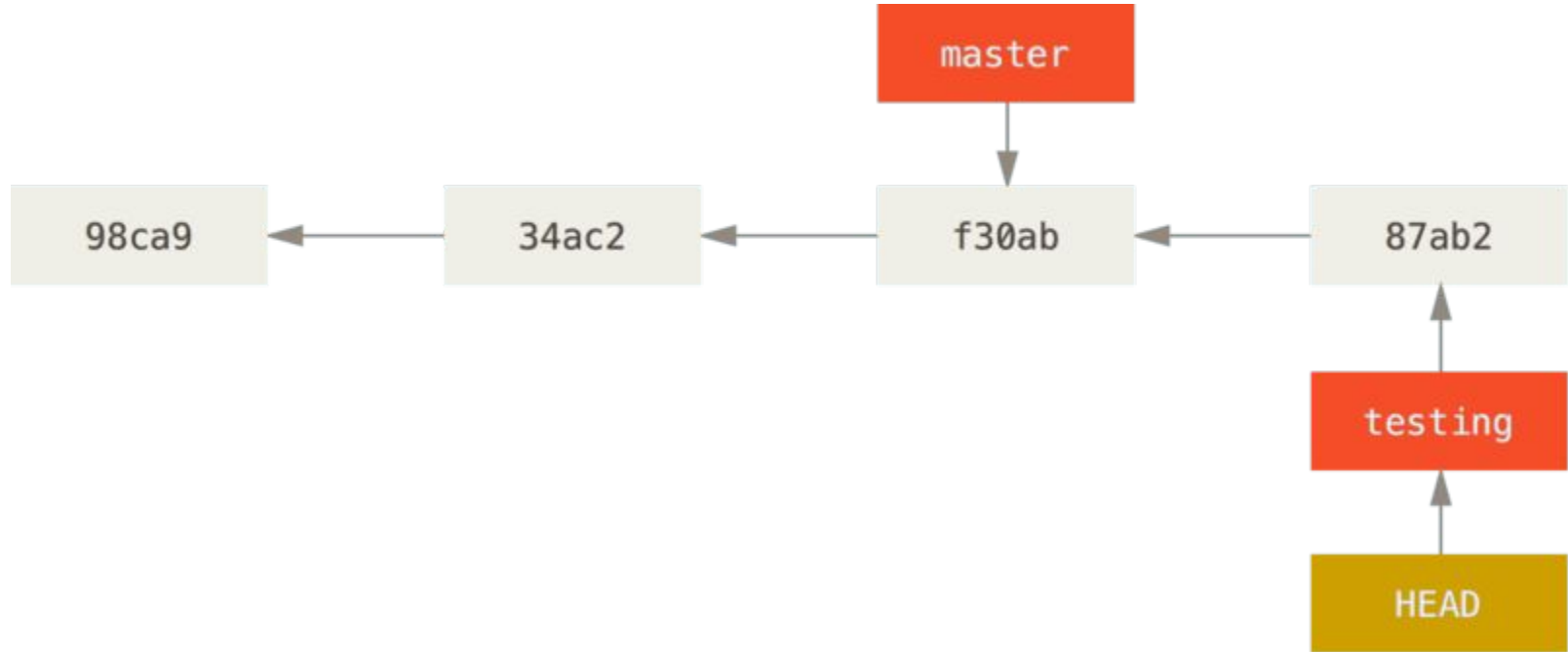
What? - Advanced Workflow (continued)

After we checkout the newly created branch



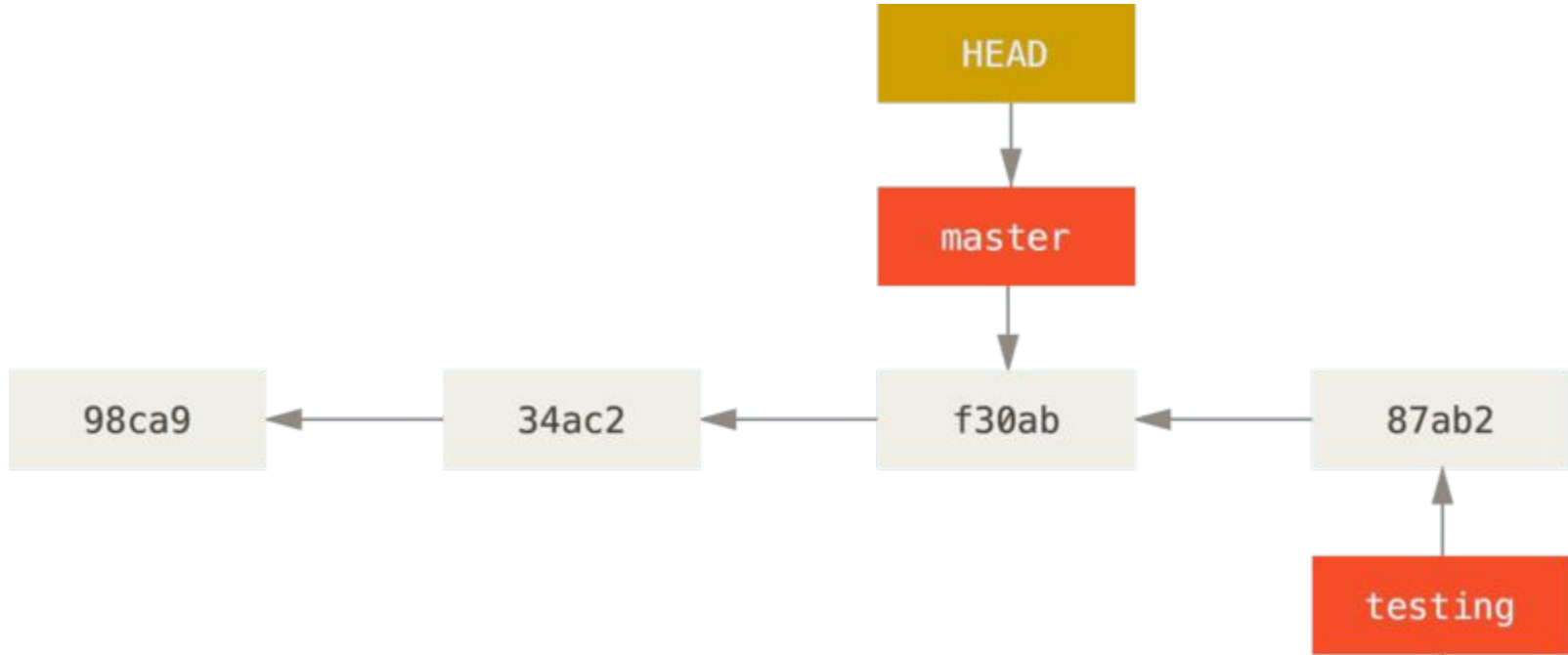
What? - Advanced Workflow (continued)

After you make commits on the new branch



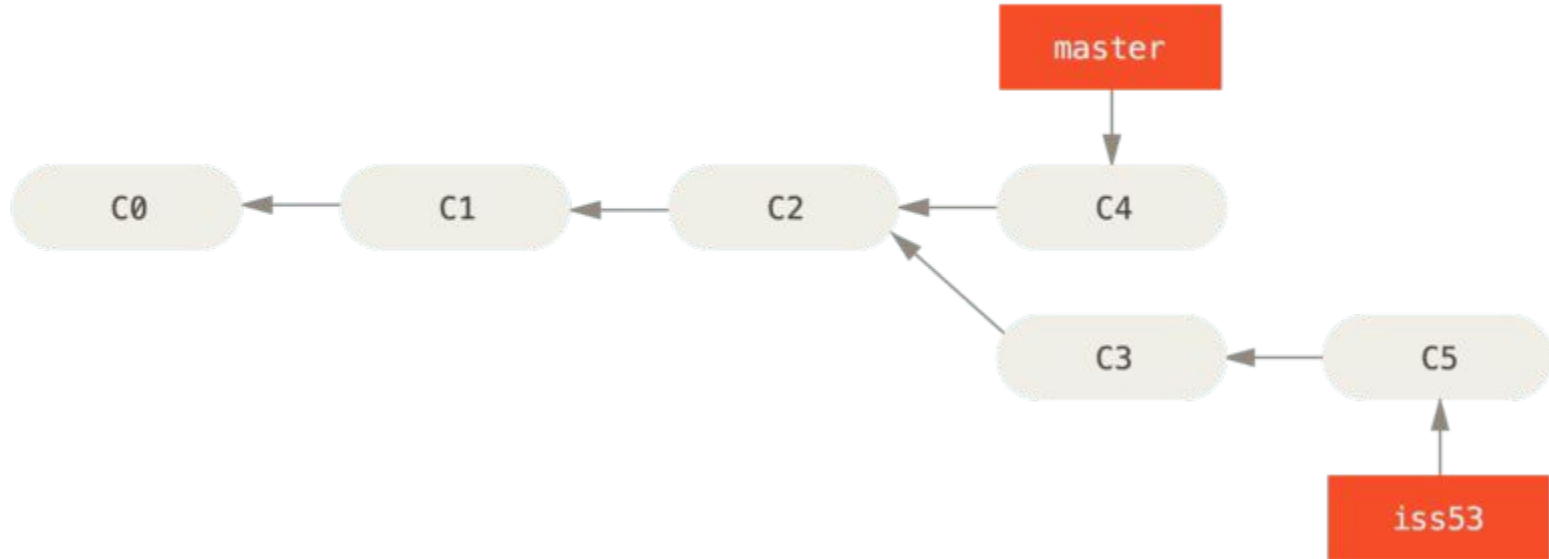
What? - Advanced Workflow (continued)

When you checkout master



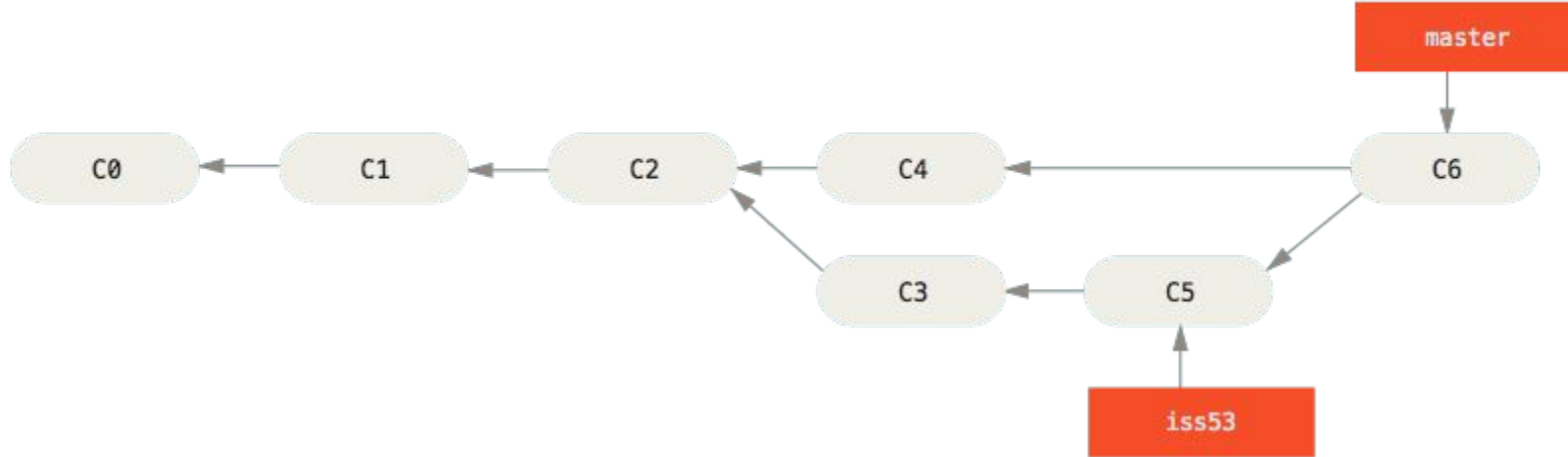
What? - Advanced Workflow (continued)

How a merge conflict might arise



What? - Advanced Workflow (continued)

Creating a merge commit



-
- Why?
 - What?
 - Git
 - GitHub
 - How?
 - Basic
 - Advanced

How? - Advanced Workflow (continued)

Helpful commands

- `git branch -l`
- `git branch -d 'branch-name'`

-
- Why?
 - What?
 - Git
 - GitHub
 - How?
 - Basic
 - **Advanced**

@rahulporuri

Fourth mistake - Merge conflicts

Changes were made to the same pieces of code on both your branch and the master branch.

```
[ab-tmp (test)]$ git checkout master
Switched to branch 'master'
[ab-tmp (master)]$
[ab-tmp (master)]$ git merge test
Auto-merging me
CONFLICT (content): Merge conflict in me
Automatic merge failed; fix conflicts and then commit the result.
[ab-tmp (master|MERGING)]$ 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:   me

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

-
- Why?
 - What?
 - Git
 - GitHub
 - How?
 - Basic
 - **Advanced**

@rahulporuri

Fourth mistake - Merge conflicts (continued)

Resolve your merge conflicts by selecting which pieces of the code should remain.

```
ab-tmp — vim me — 33x13
...a-import — -bash  ...ab-tmp — vim me >> +
1 <<<<<<< HEAD
2 asdfasdfasdf
3 asdf
4 asdfasdf
5 asdf
6 =====
7 asdfasd
8 asdasdff
9 >>>>>> test
~
me 1,8 All
```

-
- Why?
 - What?
 - Git
 - GitHub
 - How?
 - Basic
 - Advanced

Time travel using Git

- Using `git checkout` to travel back in time.

@rahulporuri

-
- Why?
 - What?
 - Git
 - GitHub
 - How?
 - Basic
 - Advanced

Correcting your mistakes

- Using `git revert 'commit-hash'` to correct your mistakes

@rahulporuri

-
- Why?
 - What?
 - Git
 - GitHub
 - How?
 - Basic
 - Advanced

Rewriting history

- Using `git reset 'commit-hash'` to undo commits and bring back changes to the staging area.

-
- Why?
 - What?
 - Git
 - GitHub
 - How?
 - Basic
 - Advanced

Prepare for mind=blown

- `git bisect` and how it can be used to pinpoint breaking changes.
- Followed by `git blame` to play the blame game

@rahulporuri

Credits

- All images are from - <https://git-scm.com/book/en/v2/>, which is awesome.