Git & Github Tutorial

Daniel Schädler & Jean-Pierre Hotz

DHBW Karlsruhe

22. October 2018

General

General

- Git is a **V**ersion **C**ontrol **S**ystem

General

- Git is a **V**ersion **C**ontrol **S**ystem
- created in 2005 by Linus Torvalds

Terminology

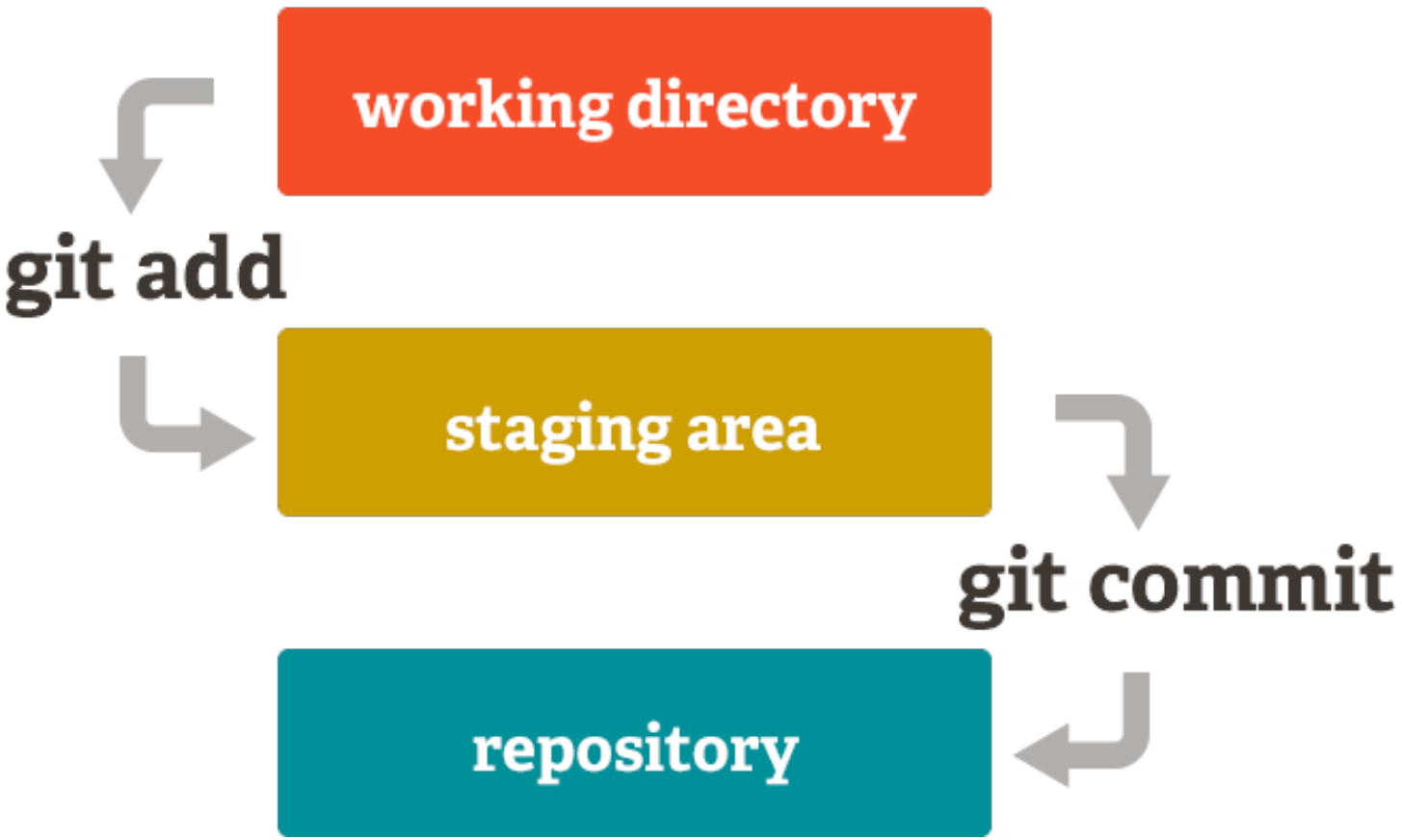
Terminology

- Commit (History)

Terminology

- Commit (History)
- (Remote) Repository

Git areas


working directory
git add
staging area
git commit
repository

Use your Github user in Git

Use your Github user in Git

- set your username and e-mail

```
$ git config --global user.name "<username>"
$ git config --global user.email <e-mail>
```

Ignoring files

Ignoring files

- exclude files with `.gitignore`-file

Ignoring files

- exclude files with `.gitignore`-file
- contains patterns

Ignoring files

- exclude files with `.gitignore`-file
- contains patterns
- `*` matches anything

Ignoring files

- exclude files with `.gitignore`-file
- contains patterns
- `*` matches anything
- postceding `/` matches a folder

Ignoring files

- exclude files with `.gitignore`-file
- contains patterns
- `*` matches anything
- postceding `/` matches a folder
- preceding `/` matches only in the root folder

Ignoring files

- exclude files with `.gitignore`-file
- contains patterns
- `*` matches anything
- postceding `/` matches a folder
- preceding `/` matches only in the root folder
- preceding `!` negates the pattern (i.e. includes files)

What do the following rules exclude?

```
/src/
!/src/working/
```

Local repositories

Local repositories

- create new local repository

  ```
  $ git init
  ```

Local repositories

- create new local repository

  ```
  $ git init
  ```

- clone a remote repository

  ```
  $ git clone <URL>
  ```

Staging and committing

Staging and committing

- stage changes

  `$ git add (--all / <files>)`

Staging and committing

- stage changes

  `$ git add (--all / <files>)`

- commit all the staged changes

  `$ git commit -m "<Message>"`

Staging and committing

- stage changes

  `$ git add (--all / <files>)`
- commit all the staged changes

  `$ git commit -m "<Message>"`
- Commit messages are important!

Display changes and commits

Display changes and commits

- see changes in code

  ```
  $ git diff <commit1> <commit2> <files>
  ```

Display changes and commits

- see changes in code

  ```
  $ git diff <commit1> <commit2> <files>
  ```

- see staged changes

  ```
  $ git status
  ```

Display changes and commits

- see changes in code

  ```
  $ git diff <commit1> <commit2> <files>
  ```

- see staged changes

  ```
  $ git status
  ```

- see commit history

  ```
  $ git log
  ```

Remote repositories

Remote repositories

- add remote repository

```
$ git remote add <Name> <URL>
```

Remote repositories

- add remote repository

  ```
  $ git remote add <Name> <URL>
  ```

- not needed when repository was cloned

Remote repositories

- add remote repository

  ```
  $ git remote add <Name> <URL>
  ```

- not needed when repository was cloned
- set remote repository as up-stream

  ```
  $ git branch -u <Name> <Branch>
  $ git push -u <Name> <Branch>
  ```

Remote repositories

- add remote repository

  ```
  $ git remote add <Name> <URL>
  ```

- not needed when repository was cloned
- set remote repository as up-stream

  ```
  $ git branch -u <Name> <Branch>
  $ git push -u <Name> <Branch>
  ```

- push all commits to remote repository

  ```
  $ git push
  ```

Remote repositories

- add remote repository

  ```
  $ git remote add <Name> <URL>
  ```

- not needed when repository was cloned
- set remote repository as up-stream

  ```
  $ git branch -u <Name> <Branch>
  $ git push -u <Name> <Branch>
  ```

- push all commits to remote repository

  ```
  $ git push
  ```

- pulls all commits from remote repository

  ```
  $ git pull
  ```

Tagging versions

Tagging versions

- list tags

```
$ git tag --list
```

Tagging versions

- list tags

  ```
  $ git tag --list
  ```

- tag a commit

  ```
  $ git tag -a <version> -m "<message>"<commit>
  ```

Tagging versions

- list tags

  ```
  $ git tag --list
  ```

- tag a commit

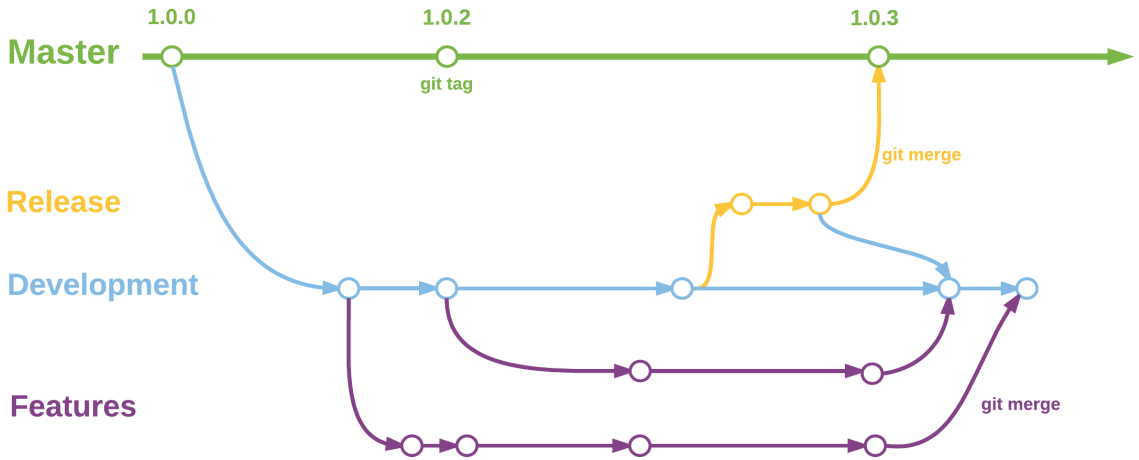
  ```
  $ git tag -a <version> -m "<message>"<commit>
  ```

- push all tags

  ```
  $ git push origin --tags
  ```

Branches

Branches



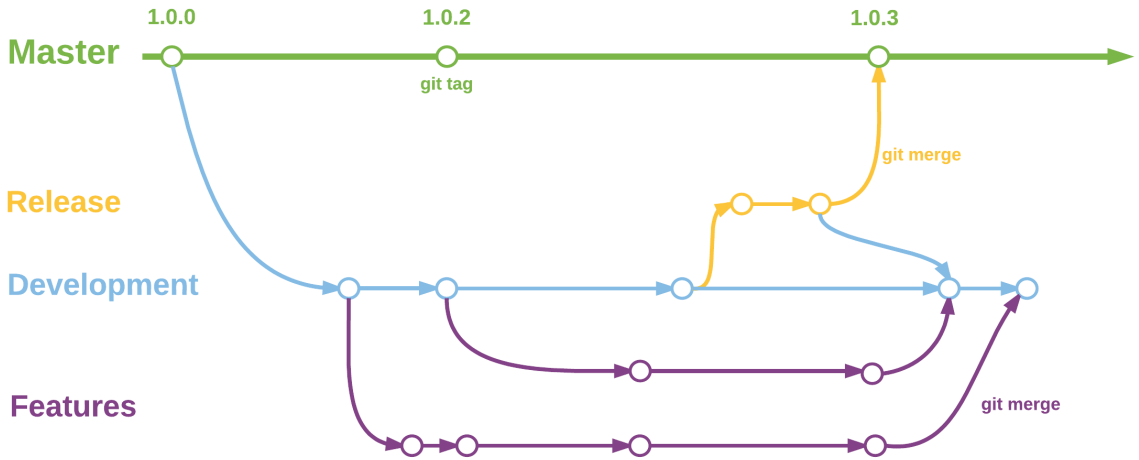
official release history with commits tagged with version numbers

integration branch for features

release branch that will get merged to master

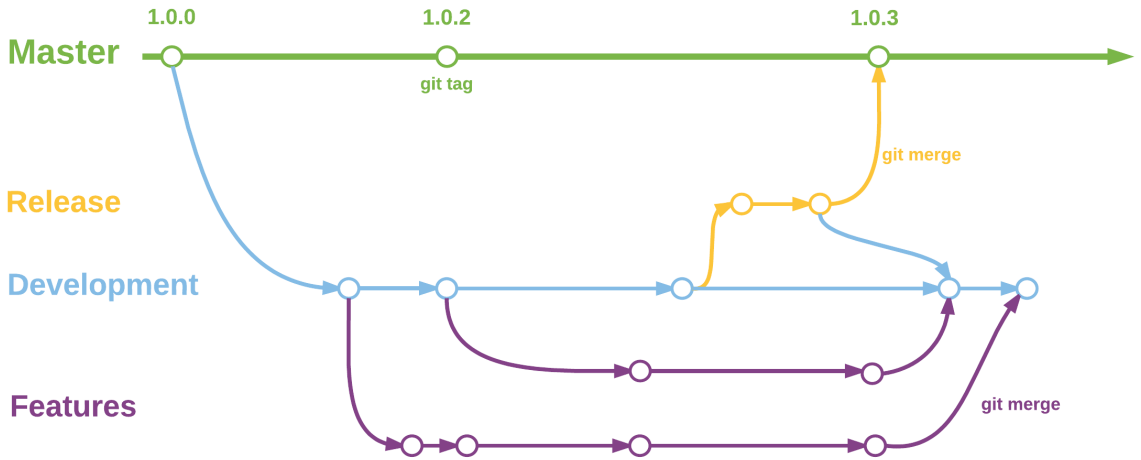
local branches for each feature that will get merged back to development branch

Branches



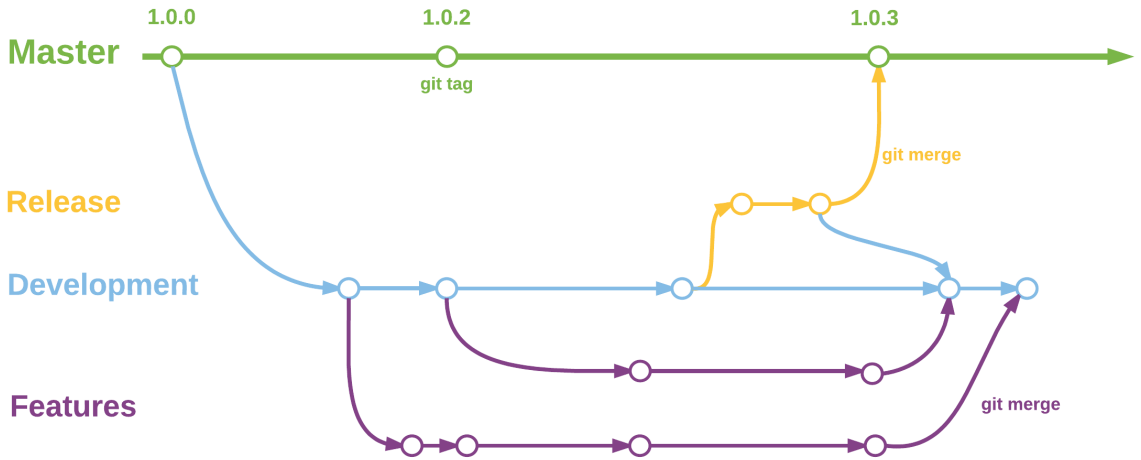
- different directions the project is heading of into

Branches



- different directions the project is heading of into
- different features (feature branches)

Branches



- different directions the project is heading of into
- different features (feature branches)
- can be merged again

merging

merging

- bringing two branches back together

```
$ git merge <branchname>
```

merging

- bringing two branches back together

  ```
  $ git merge <branchname>
  ```

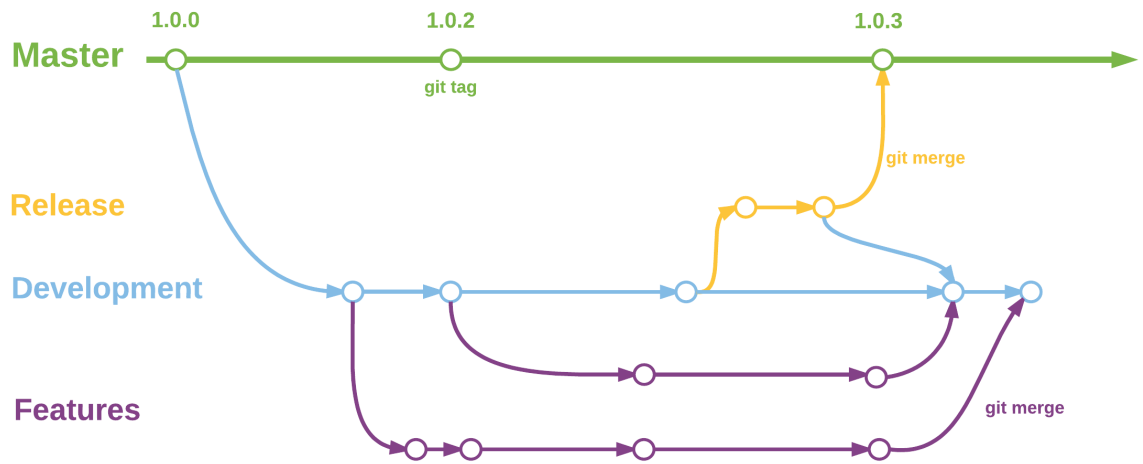
- merges `<branchname>` into current branch

merging

- bringing two branches back together

  ```
  $ git merge <branchname>
  ```

- merges `<branchname>` into current branch



conflict resolving

conflict resolving

- conflicts happen when two people edit the same line

conflict resolving

- conflicts happen when two people edit the same line

```
To help you with git,
<<<<<<< HEAD
this line has many infos
=======
I write many different tutorials
>>>>>>> conflicting branch
```

- resolve them manually by editing the files

conflict resolving

- conflicts happen when two people edit the same line

```
To help you with git,
<<<<<<< HEAD
this line has many infos
=======
I write many different tutorials
>>>>>>> conflicting branch
```

- resolve them manually by editing the files

- `git add <conflicting-file>`

rebasing

rebasing

- reapply commits on top of another base tip

rebasing

- reapply commits on top of another base tip
- keeps the git history straight

rebasing

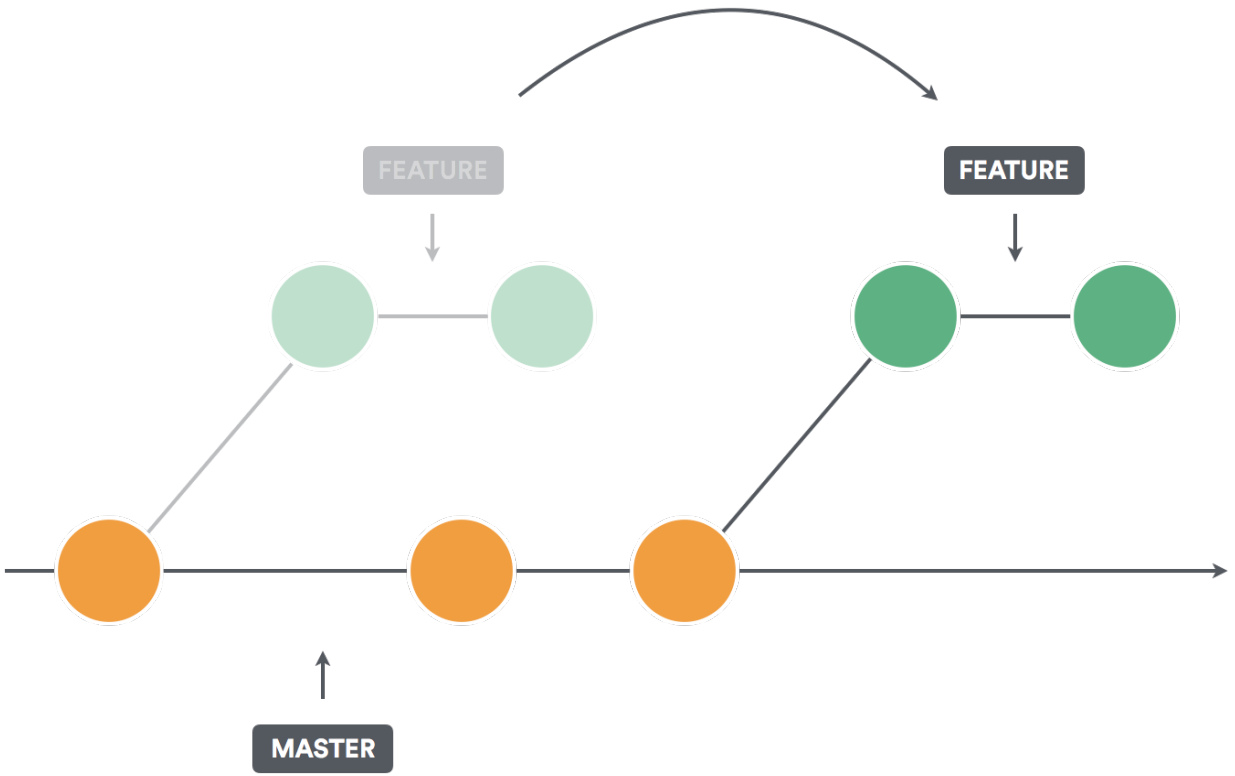
- reapply commits on top of another base tip
- keeps the git history straight
- allows for `--ff-only` merge

```
$ git merge --ff-only <branchname>
```

rebasing

- reapply commits on top of another base tip
- keeps the git history straight
- allows for `--ff-only` merge

```
$ git merge --ff-only <branchname>
```



Rewriting history

Rewriting history

- You can rewrite git history

Rewriting history

- You can rewrite git history
- `$ git rebase -i HEAD~<number>`
  `<number>` is the number of commits you want to go back and edit

Rewriting history

Rewriting history

```
$ git rebase -i HEAD~<number>
```

```
pick ce28e71 change gitignore intellij
pick 4067bbb Add chapter about tags
pick 78580c3 Change some slides
pick 1641359 Complete my slides
pick 091611d add current pdf version
pick db3e8b4 add Daniels part to slides

# Rebase 3a4a271..db3e8b4 onto 3a4a271 (7 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop to amend
# s, squash <commit> = use commit, but meld into previous commit
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a commit here THAT COMMIT WILL BE LOST.
```

GitHub (& others)

GitHub (& others)

complete copy

branch protection

pull requests

issues
forks
statistics