/tutorial/git

Thomas Rausch
thomas.rausch@tuwien.ac.at

Institute for Information Systems
Distributed Systems Group
TU Wien

http://git-scm.com

Of Linus and learning curves

“Linus is a guy who delights in being cruel to people . . . ”

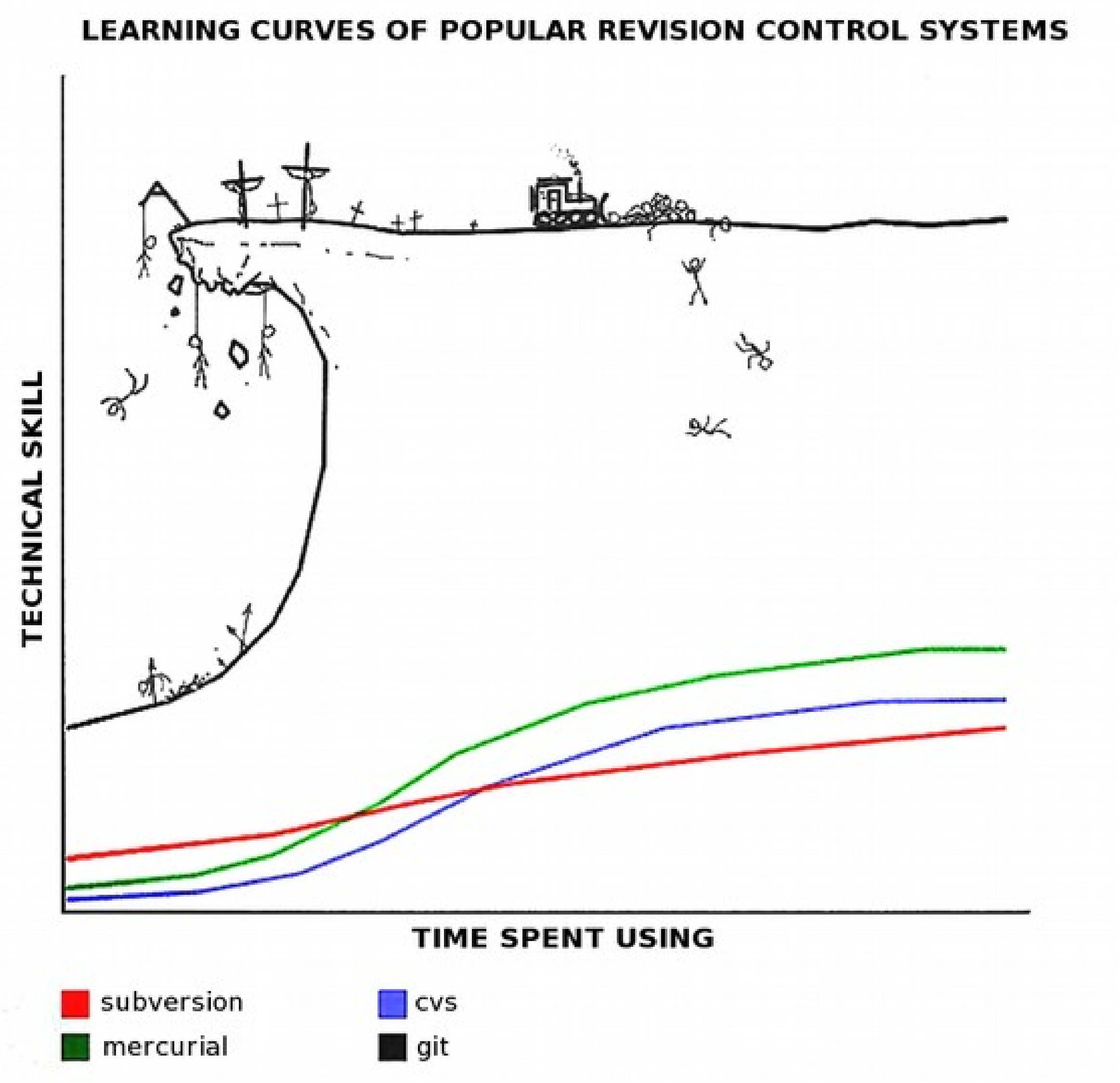
Of Linus and learning curves

“Linus is a guy who delights in being cruel to people . . . ”

“His latest cruel act is to create a revision control system which is ***expressly designed*** *to* ***make you feel less intelligent*** *than you thought you were”* [1]

[1]: Tech Talk: Linus Torvalds on git - http://youtu.be/4XpnKHJAok8

Of Linus and learning curves



What you should take from today

What you should take from today

git =

basics stages branch merge remotes advanced

Configure your user

```
$ git config --global user.name "Thomas Rausch"
$ git config --global user.email thomas@rauschig.org
```

Initialize an empty repository

```
$ git init
Initialized empty Git repository in /home/thomas/git-tutorial/.git/
```

Clone a remote repository

```
$ git clone <repo> [<directory>]
```

Check the status of your repository

```
$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#    README.md
#    src/
nothing added to commit but untracked files present (use "git add" to track)
```

```
$ git status -sb
# Initial commit on master
?? README.md
?? src
```

Start tracking files

```
$ git add README.md
```

```
$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#    new file:   README.md
#
```

Commit changes

```
$ git commit -m "add readme file"
[master (root-commit) d4c59ff] add readme file
 1 file changed, 3 insertions(+)
 create mode 100644 README.md
```

```
$ git status
# On branch master
nothing to commit (working directory clean)
```

View differences of current unstaged modifications

```
$ git diff --color
diff --git a/src/Main.java b/src/Main.java
index 66c8e93..dcdf6cb 100644
--- a/src/Main.java
+++ b/src/Main.java
@@ -1,5 +1,7 @@
 class Main {
+    static int status = 0;
+
     public static void main(String[] args) {
-        System.exit(0);
+        System.exit(status);
     }
 }
```

Unstaging files

```
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#    new file:   src/HelloWien.java
#    new file:   src/HelloWorld.java
```

```
$ git reset HEAD src/HelloWien.java
```

```
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#    new file:   src/HelloWorld.java
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#    src/HelloWien.java
#
```

Undoing local unstaged changes

```
$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#    modified:   README.md
```

```
$ git checkout -- README.md
```

```
$ git status
# On branch master
nothing to commit (working directory clean)
```

Viewing the history

```
$ git log
commit 9c3cb834c43d67cc37b15e74b64dc830c1e78199
Author: Thomas Rausch <thomas@rauschig.org>
Date:   Mon Oct 14 00:04:53 2013 +0100

    modified readme file

commit d4c59ffd7e676dad6aef2cc244b87e3c579aa904
Author: Thomas Rausch <thomas@rauschig.org>
Date:   Sun Oct 13 23:45:39 2013 +0100

    add readme file
```

```
$ git log --oneline
9c3cb83 modified readme file
d4c59ff add readme file
```

```
$ git log --graph --pretty=format:'%Cred%h%Creset %an: %s - %Creset %C(yellow)%d
%Creset %Cgreen(%cr)%Creset' --abbrev-commit -date=relative
* 9c3cb83 Thomas Rausch: modified readme file -    (HEAD, master) (4 minutes ago)
* d4c59ff Thomas Rausch: add readme file -    (24 minutes ago)
```

basics

stages

branch

merge

remotes

advanced

working directory

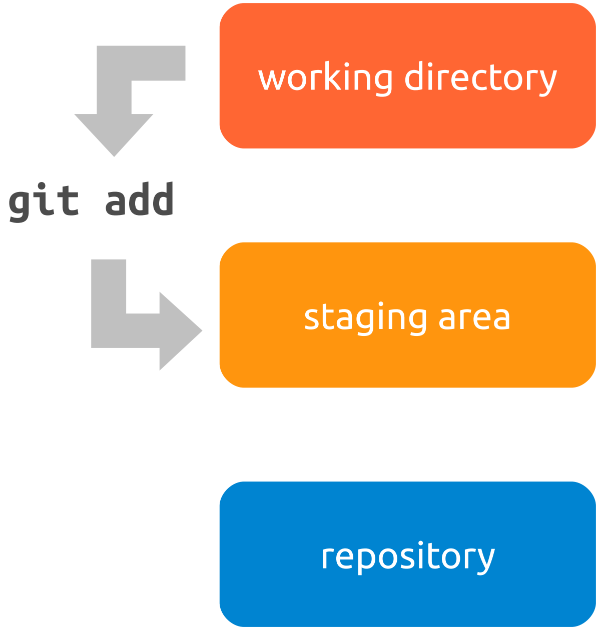
working directory

staging area

working directory

staging area

repository

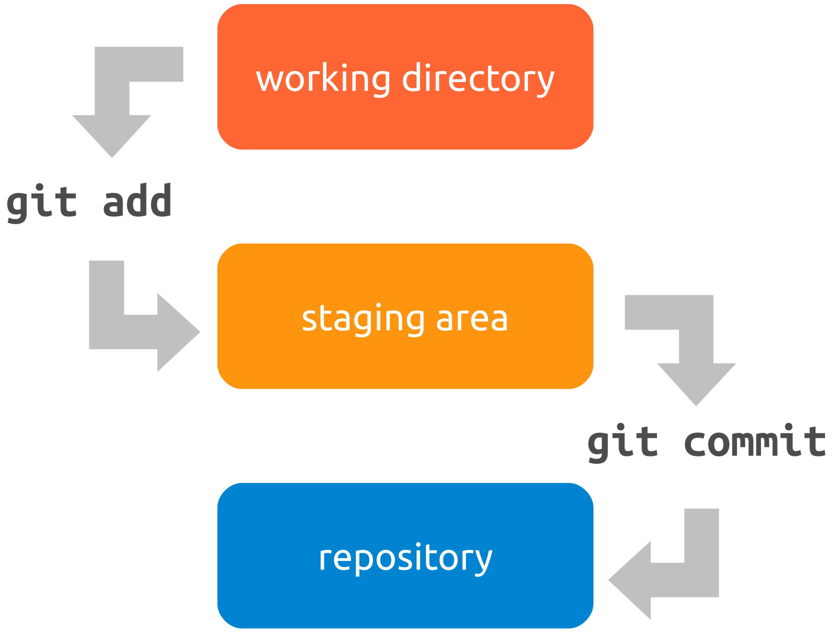


working directory

`git add`

staging area

repository



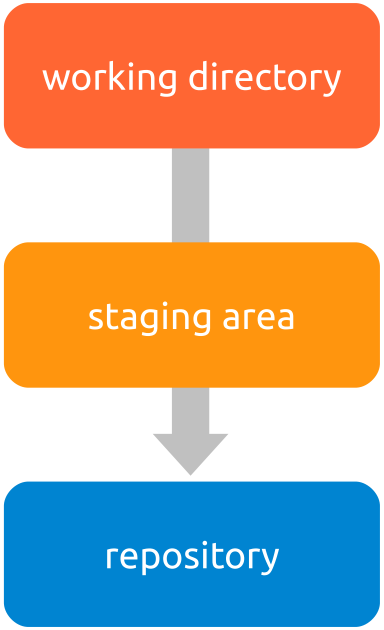
working directory

`git add`

staging area

`git commit`

repository

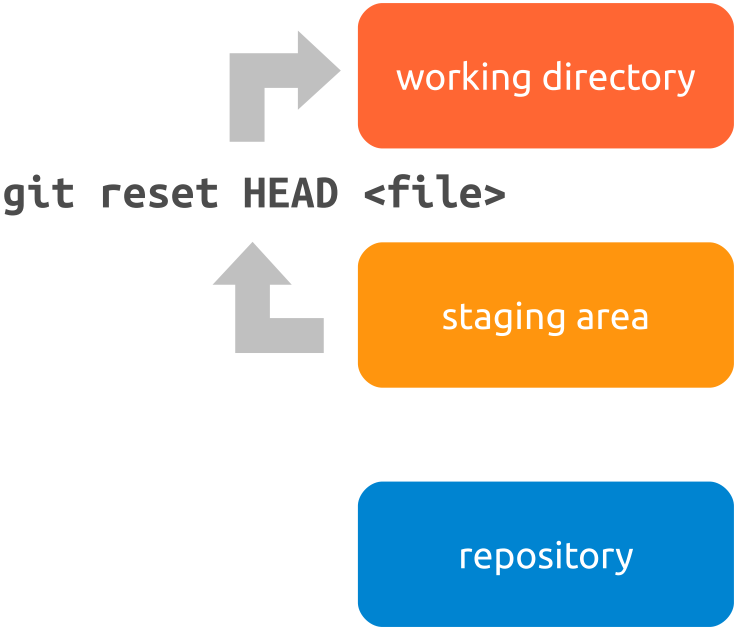


working directory

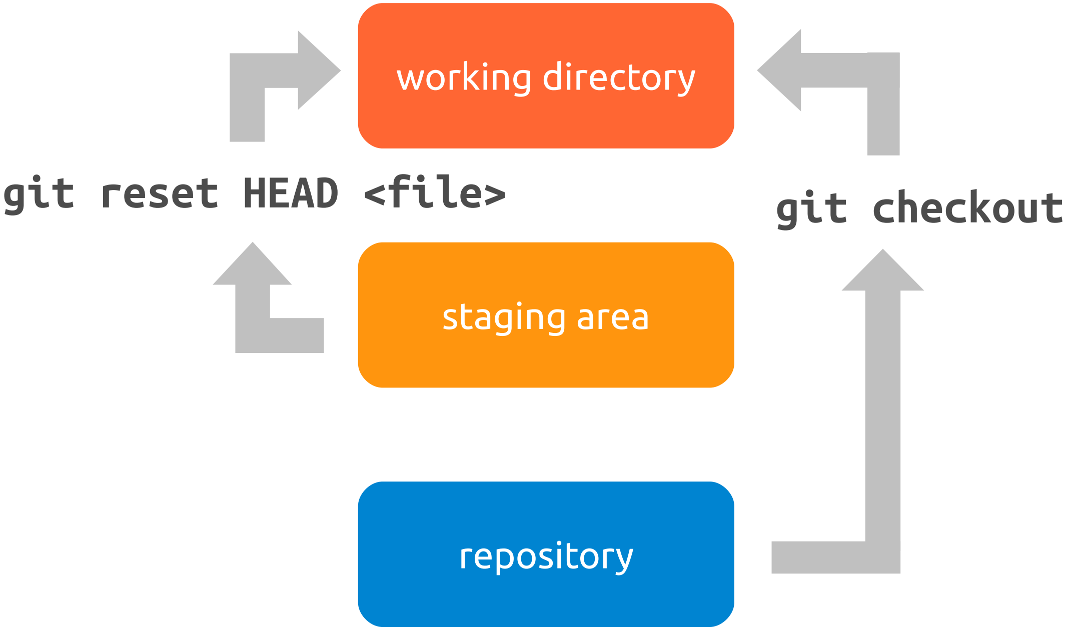
staging area

repository

```
git commit -a
```



```
git reset HEAD <file>
```



working directory

`git reset HEAD <file>`

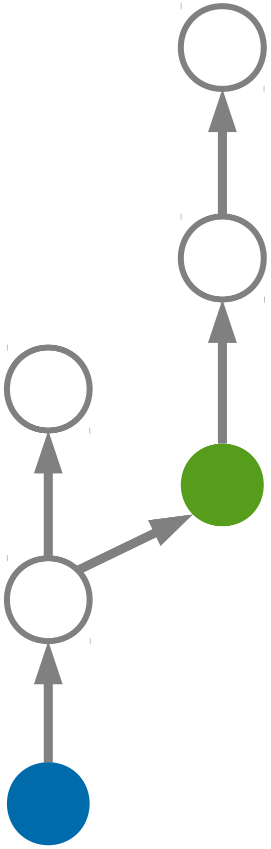
`git checkout`

staging area

repository

basics stages branch merge remotes advanced

basics stages branch merge remotes advanced



List branches

```
$ git branch
  develop
* master
  new-feature
```

List branches

```
$ git branch
  develop
* master
  new-feature
```

Branch you have currently checked out

Change into a branch

```
$ git checkout develop
Switched to branch 'develop'
```

Manage branches

Create a new branch from the one you have currently checked out

```
$ git branch <branch>
```

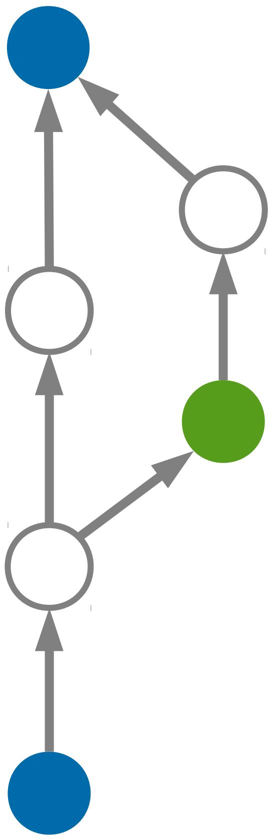
Rename a branch

```
$ git branch -m <oldbranch> <newbranch>
```

Delete a branch

```
$ git branch -D <branch>
```

basics stages **branch** merge remotes advanced basics stages branch **merge** remotes advanced

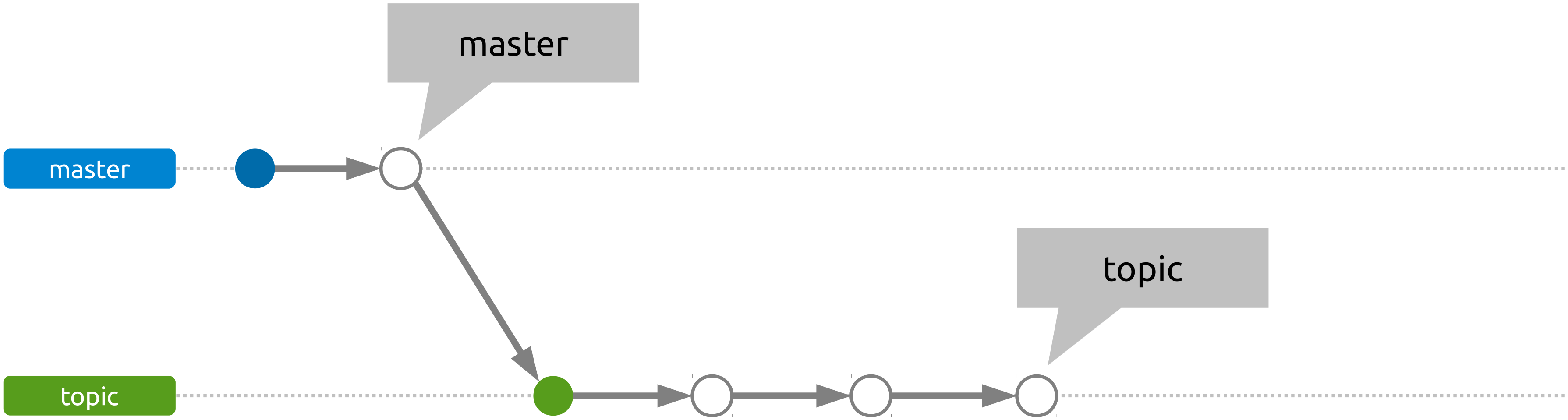


Merge branches

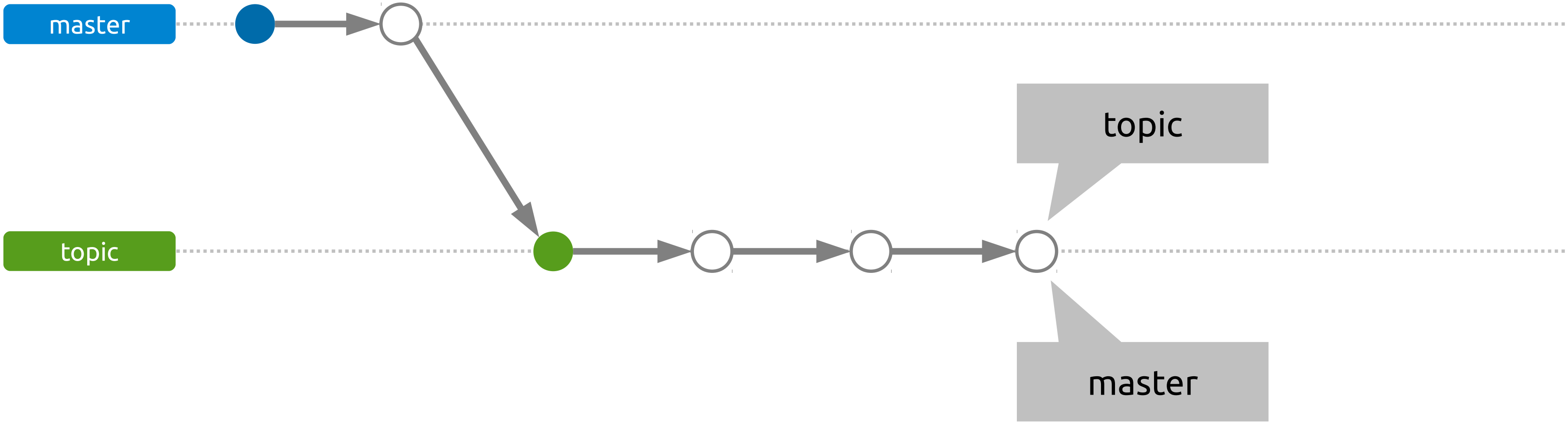
Merge the specified branch into the current branch (the one you have checked out)

```
$ git checkout master
$ git merge topic
```

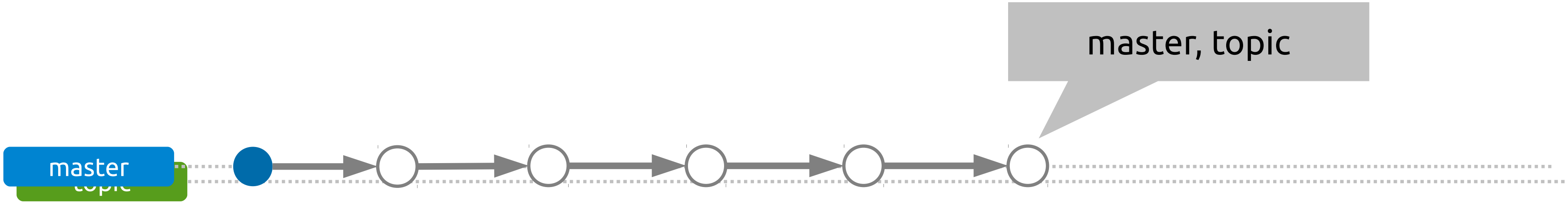
Merges topic into master

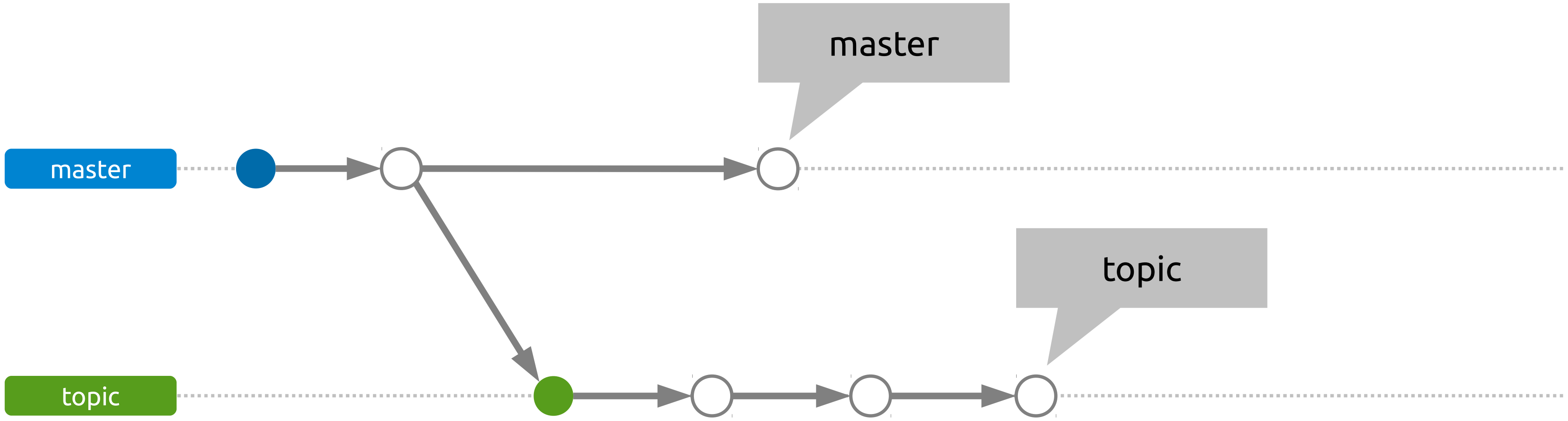

master
master
topic
topic

Fast-forward

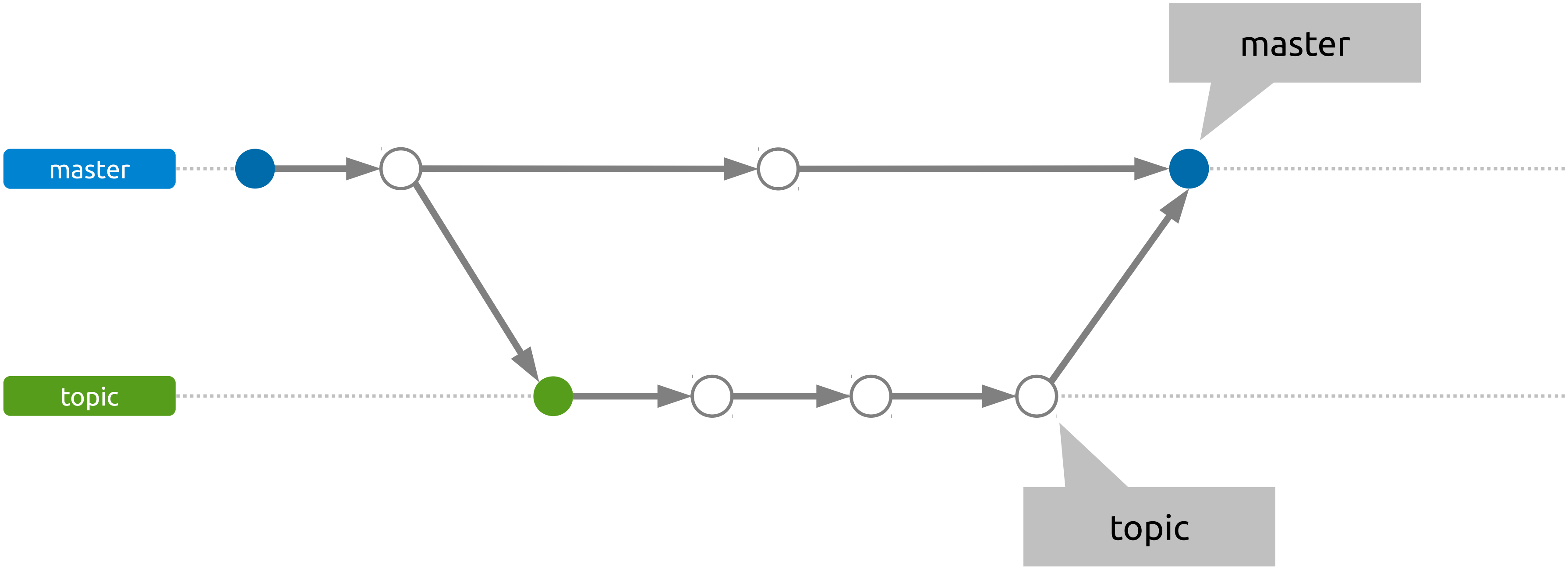


Fast-forward

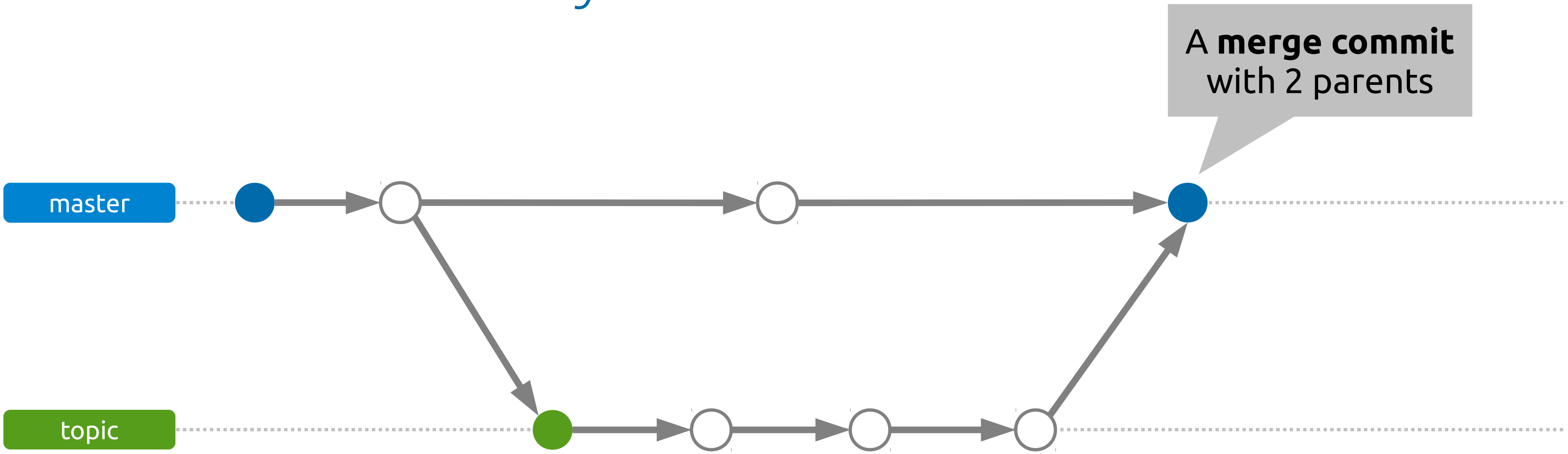



master
master
topic
topic

Recursive three-way

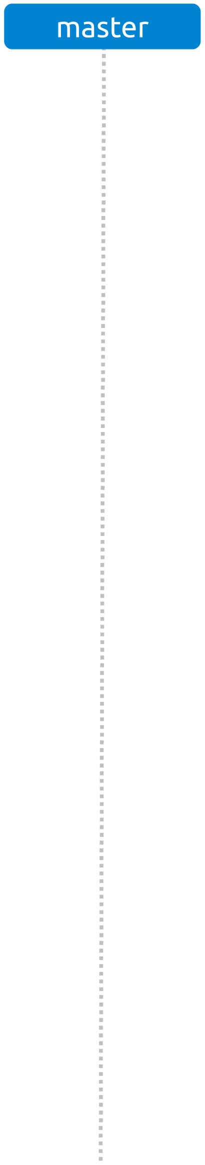


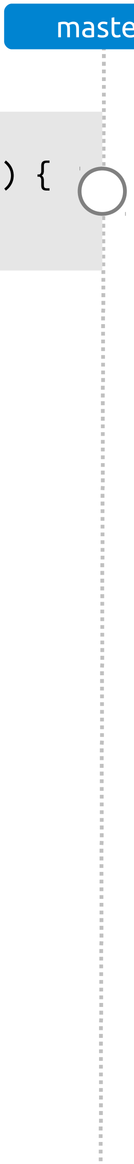
Recursive three-way



A **merge commit** with 2 parents

Conflicts

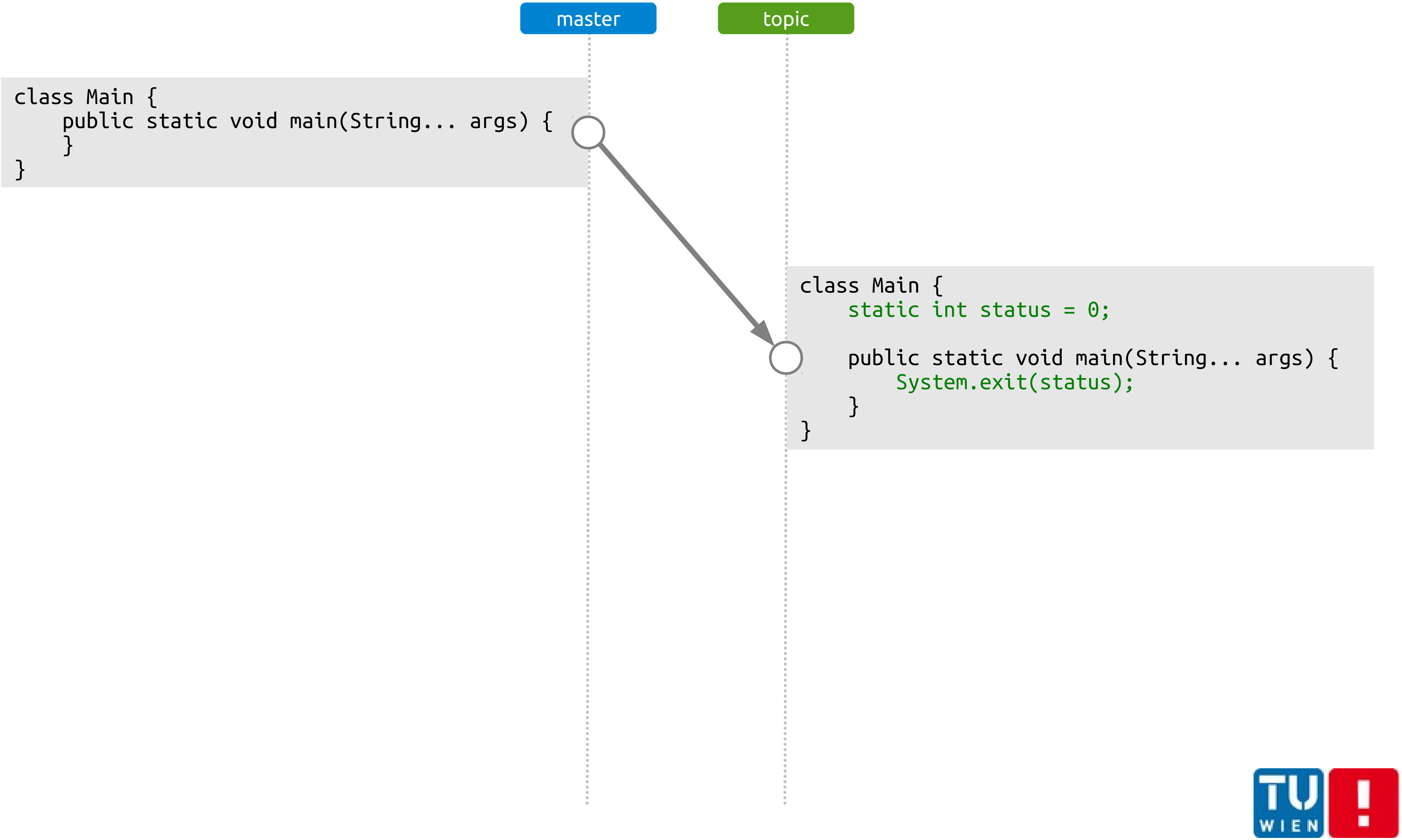

master



```
class Main {
    public static void main(String... args) {
    }
}
```

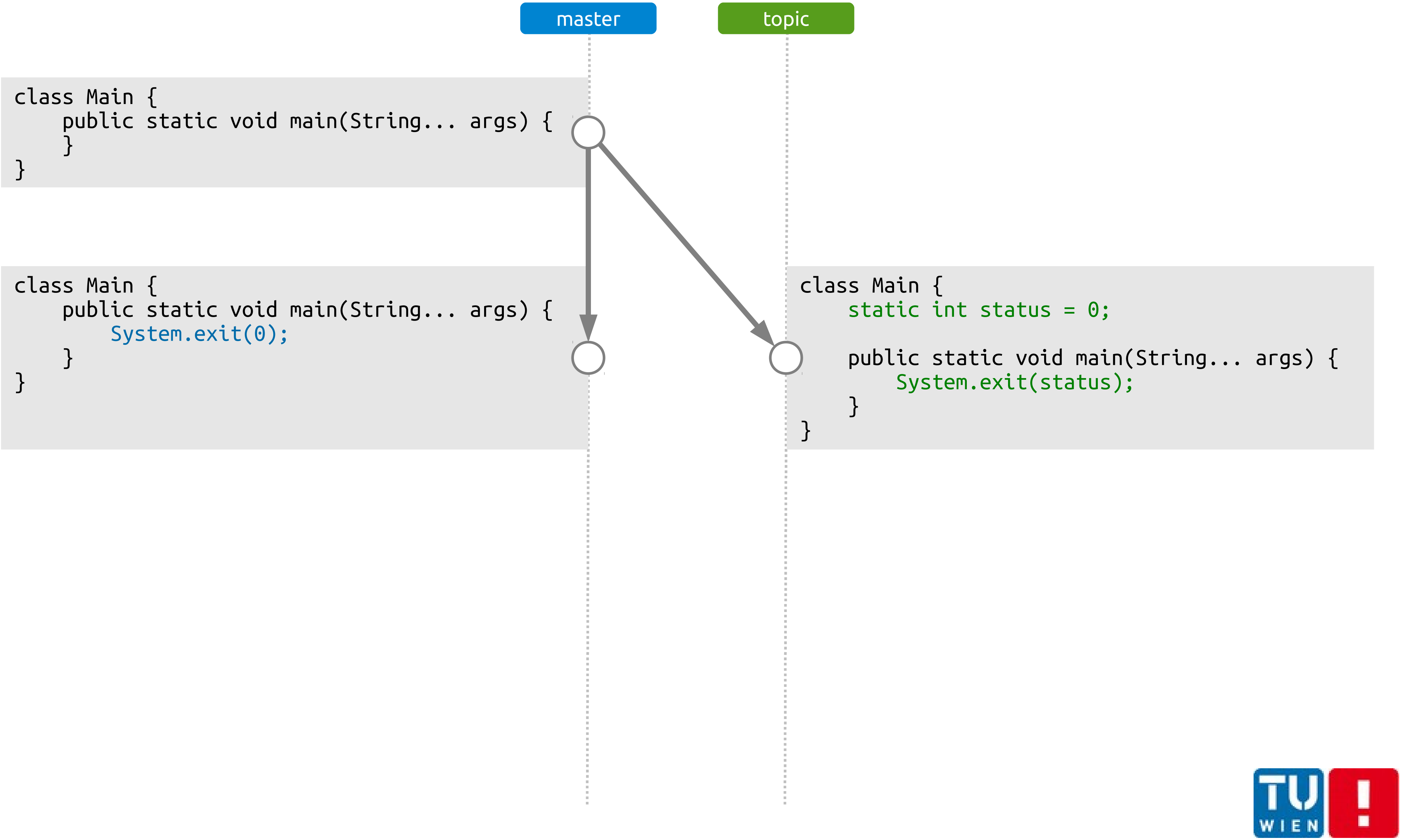
master
topic

```
class Main {
    public static void main(String... args) {
    }
}
```



```
class Main {
    public static void main(String... args) {
    }
}
```

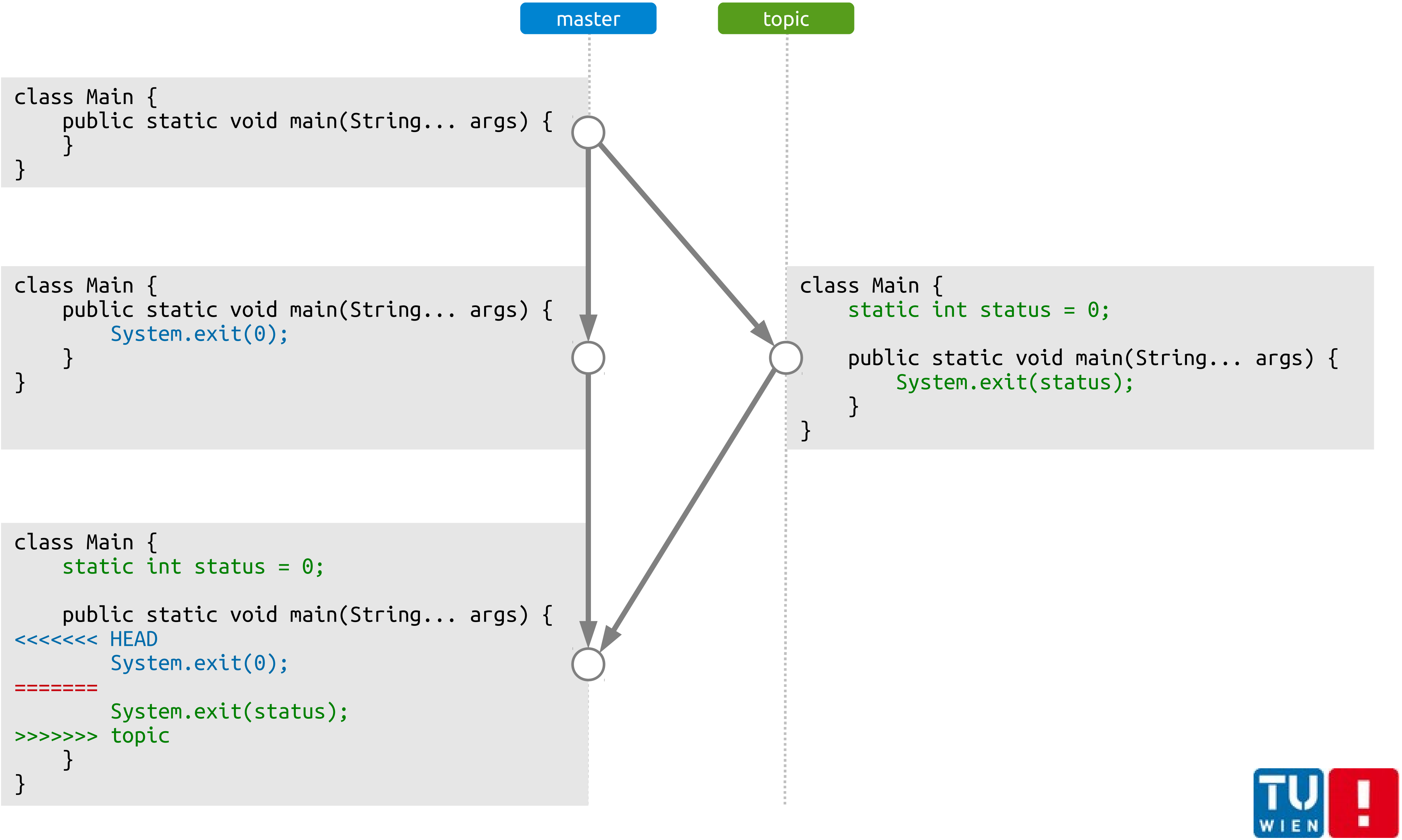
```
class Main {
    static int status = 0;

    public static void main(String... args) {
        System.exit(status);
    }
}
```



```
class Main {
    public static void main(String... args) {
    }
}
```

```
class Main {
    public static void main(String... args) {
        System.exit(0);
    }
}
```

```
class Main {
    static int status = 0;

    public static void main(String... args) {
        System.exit(status);
    }
}
```



```
class Main {
    public static void main(String... args) {
    }
}
```

```
class Main {
    public static void main(String... args) {
        System.exit(0);
    }
}
```

```
class Main {
    static int status = 0;

    public static void main(String... args) {
        System.exit(status);
    }
}
```

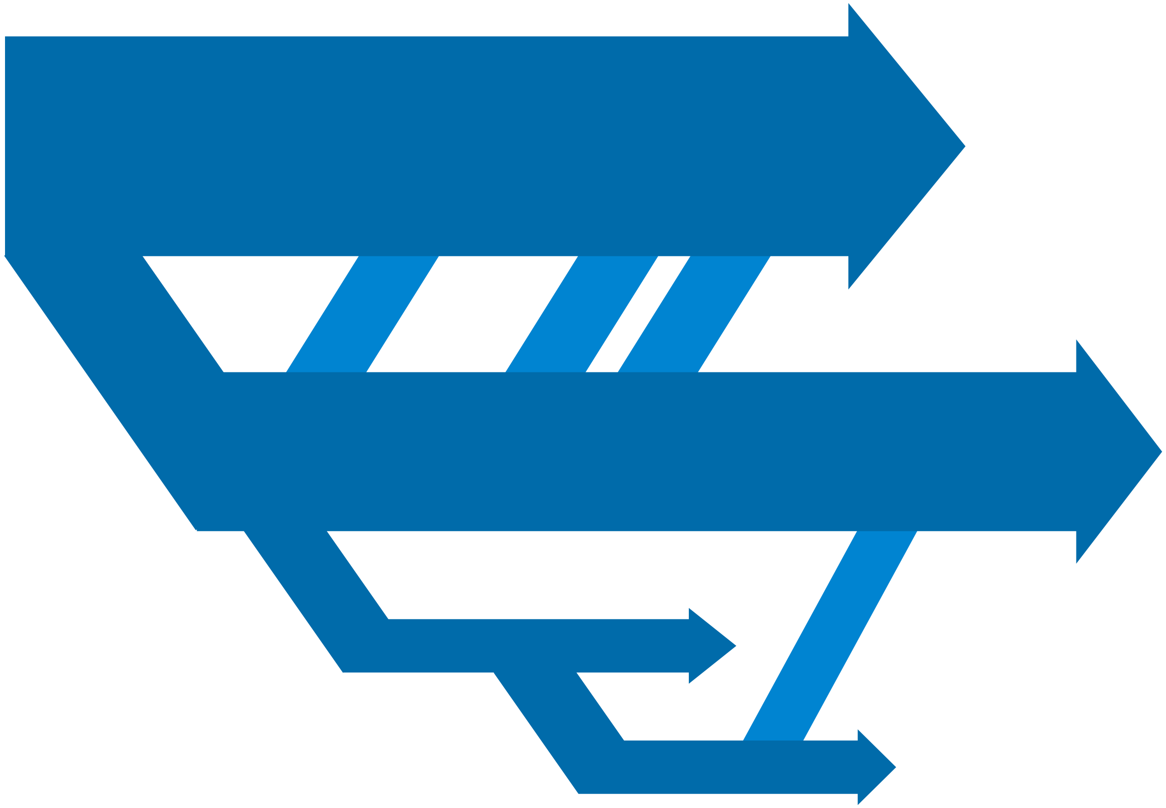
```
class Main {
    static int status = 0;

    public static void main(String... args) {
<<<<<<< HEAD
        System.exit(0);
=======
        System.exit(status);
>>>>>>> topic
    }
}
```

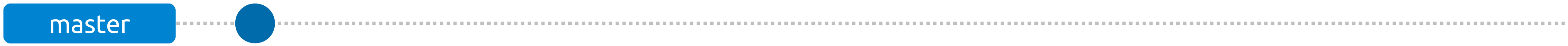
basics stages branch **merge** remotes advanced

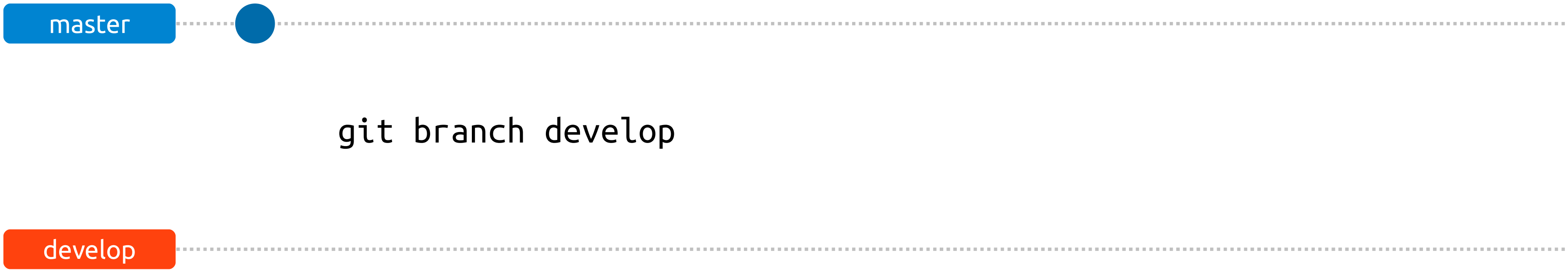
master

develop

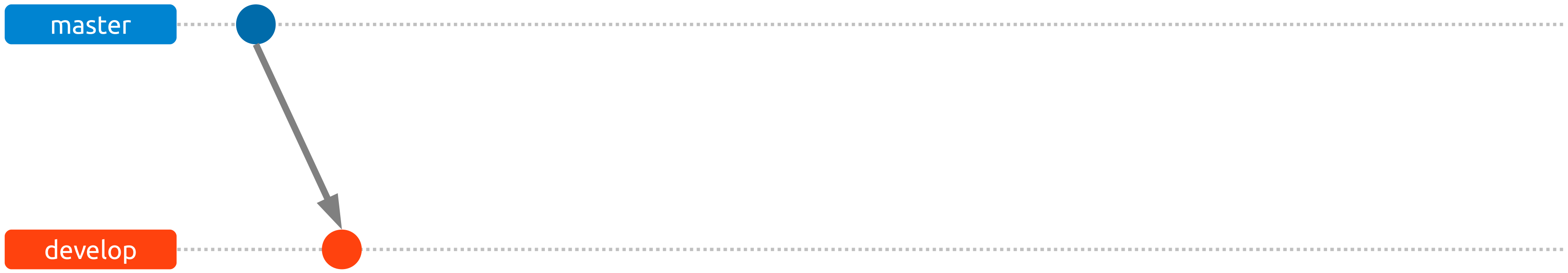
topic (features)

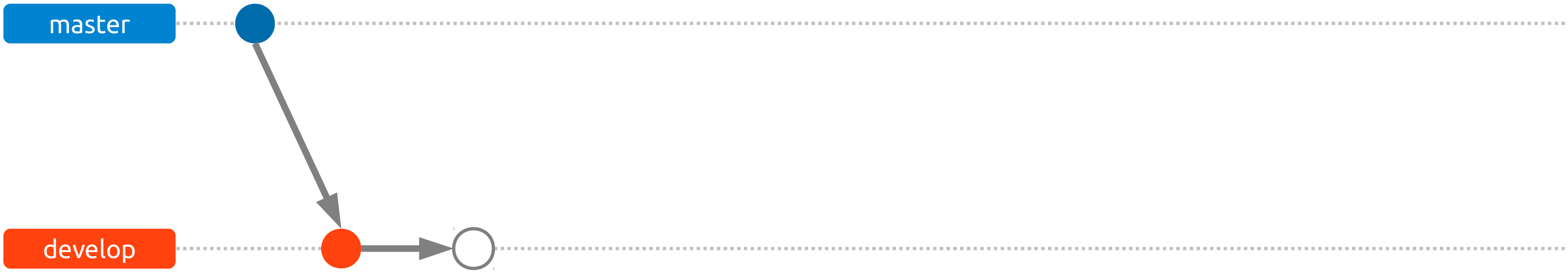


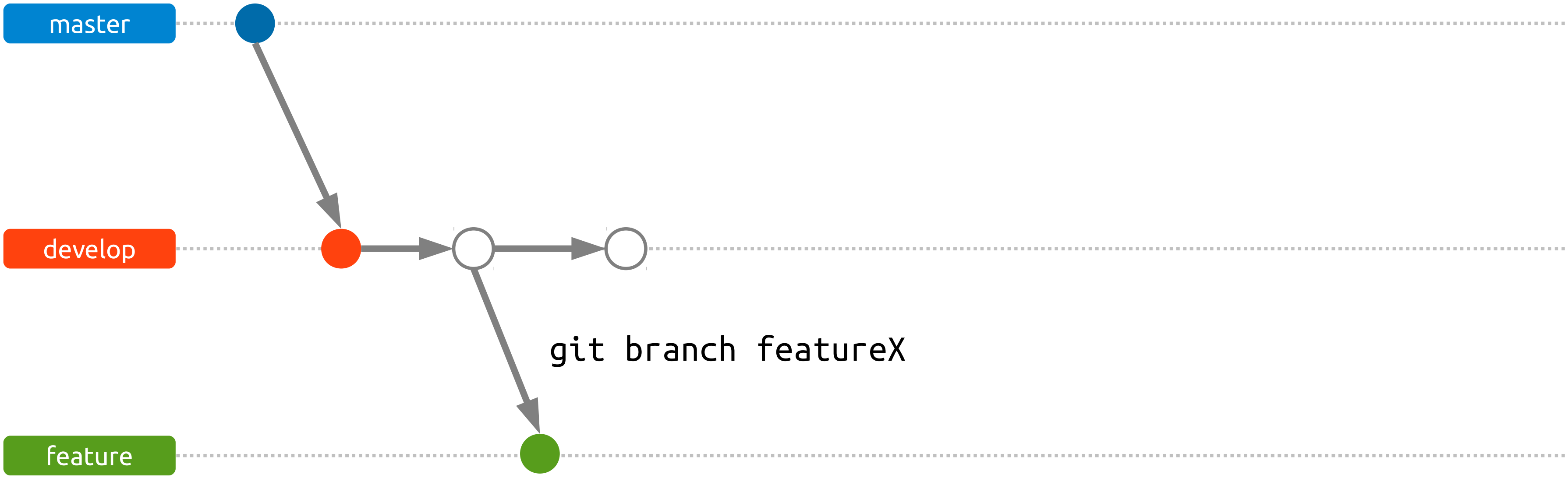

master



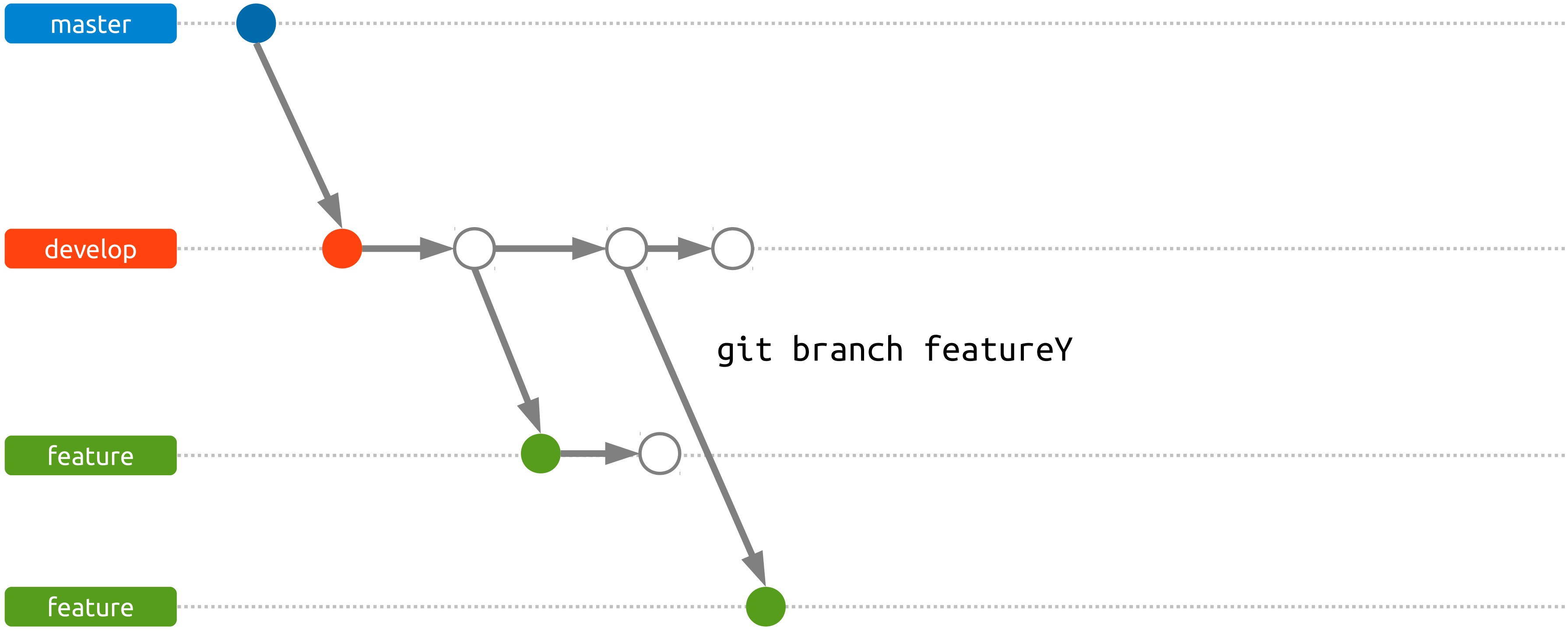
```
git branch develop
```


master
develop

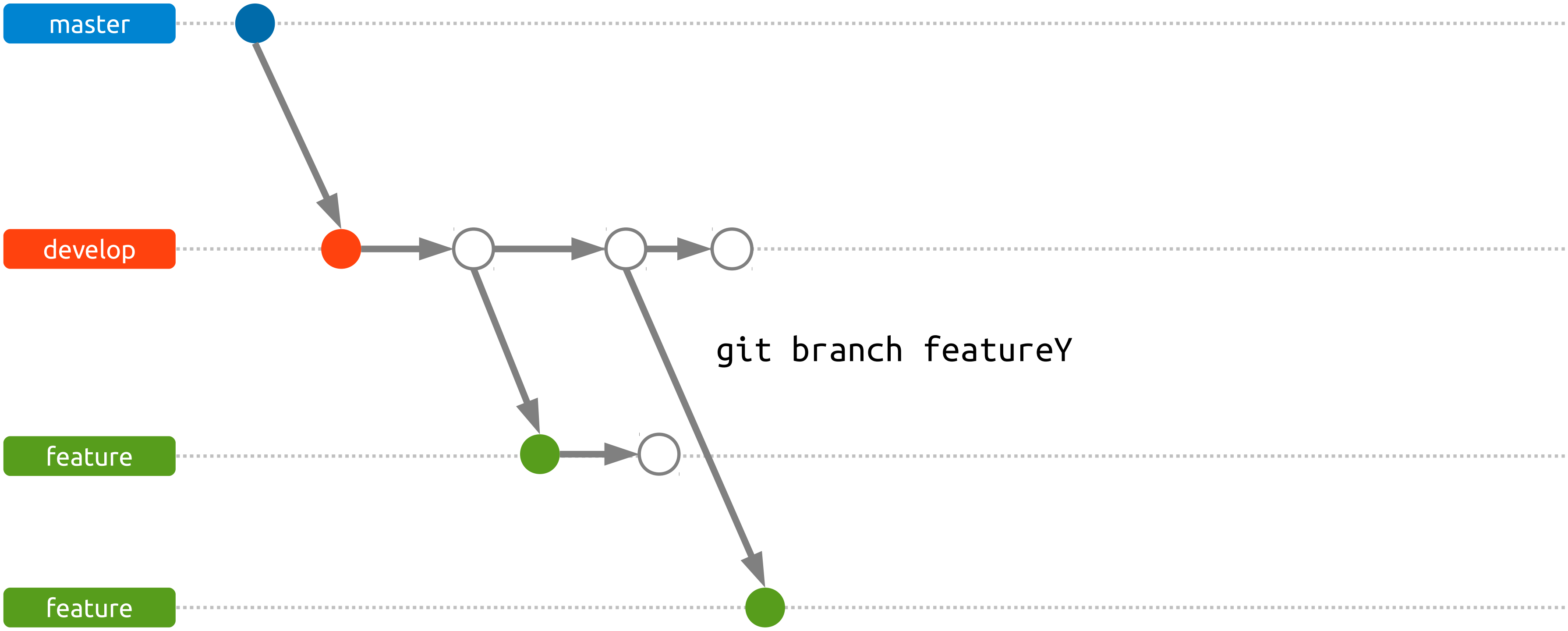

master
develop



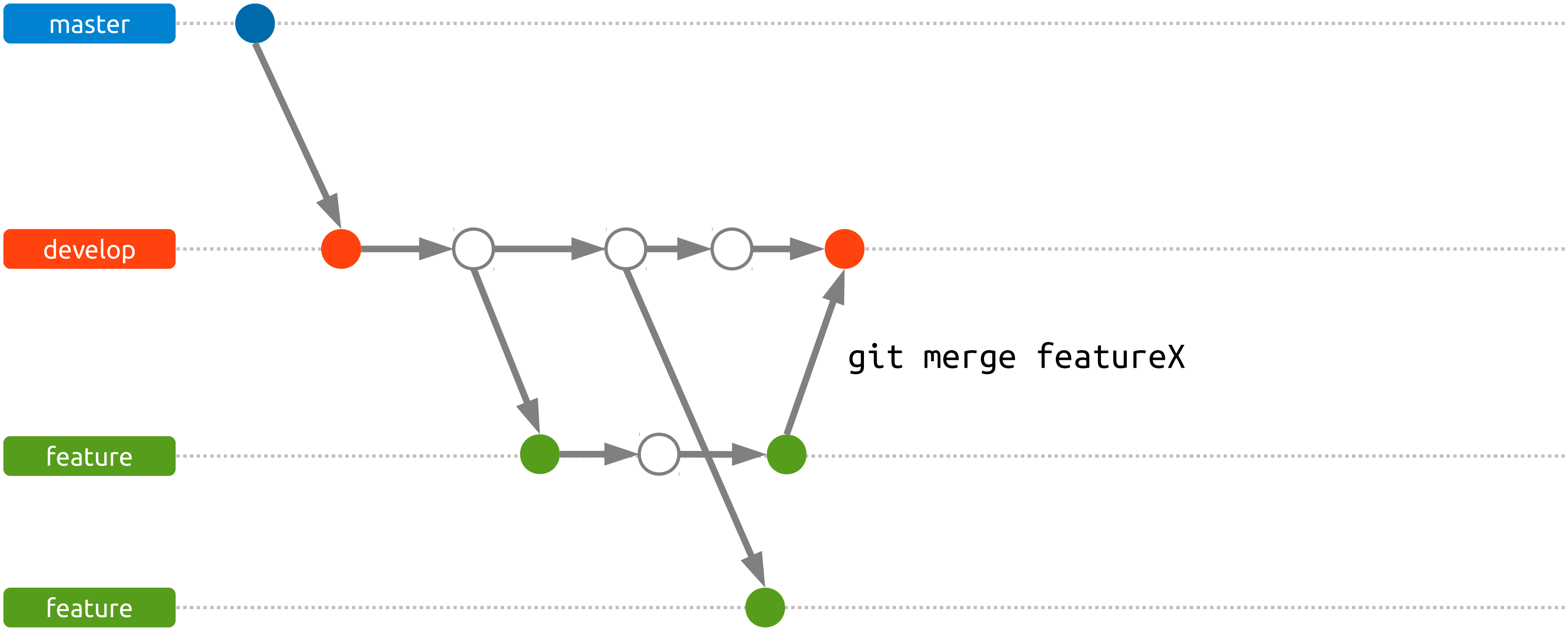
```
git branch featureX
```

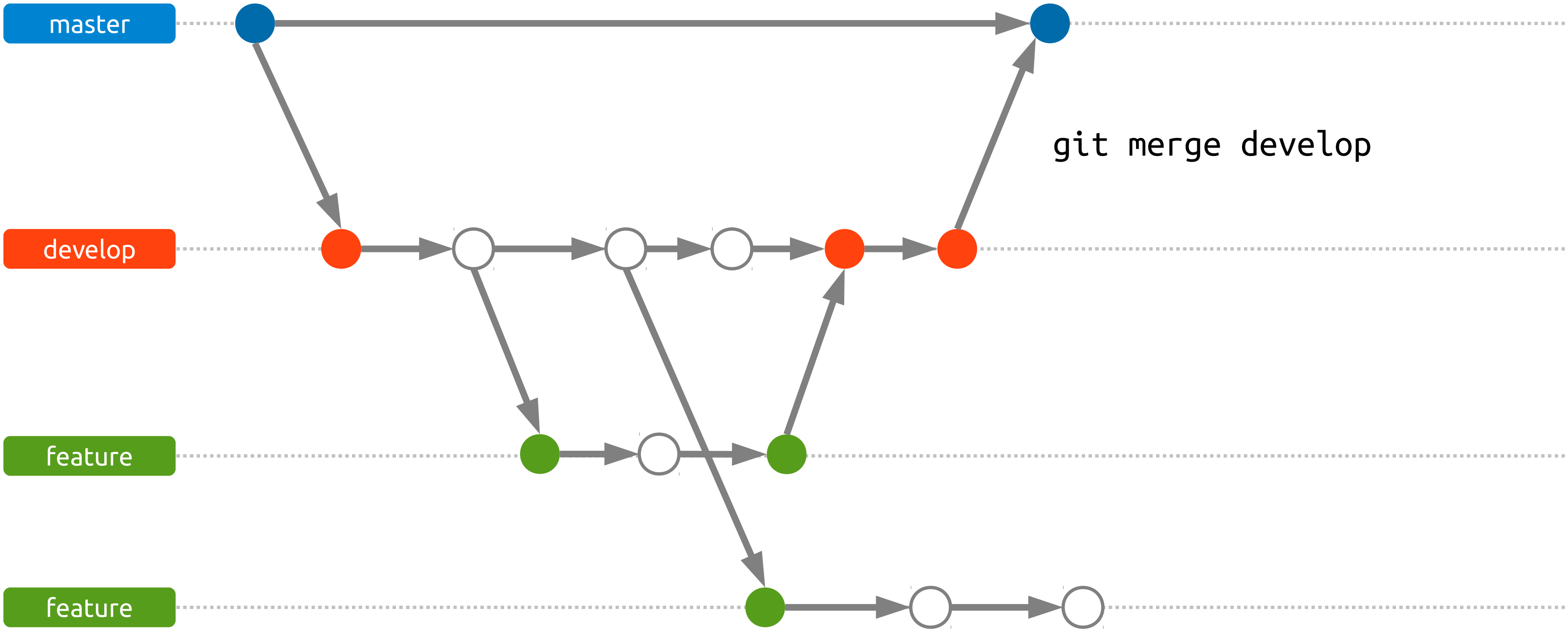


```
git branch featureY
```



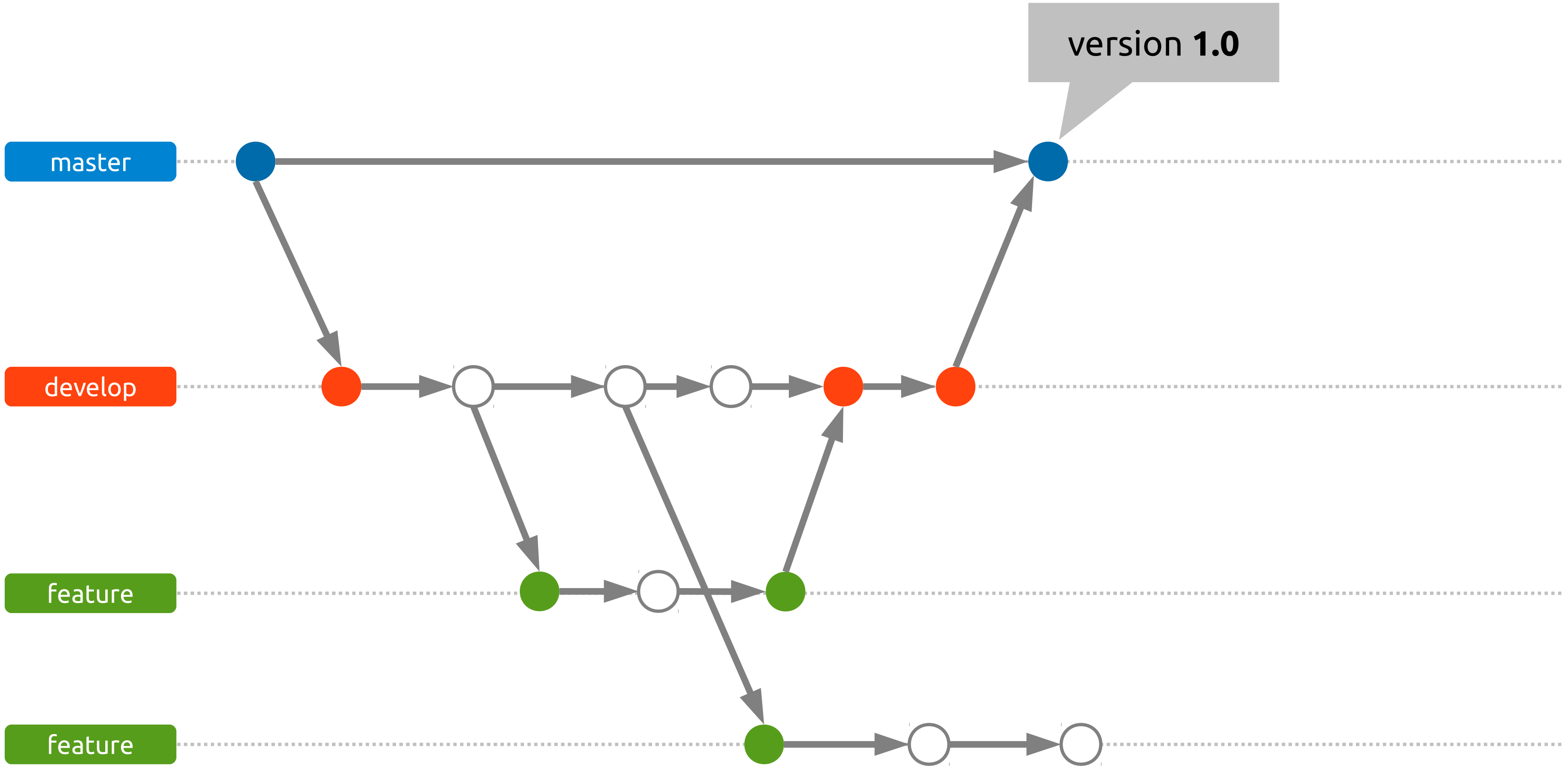
```
git branch featureY
```


master
develop
feature
feature
git merge featureX

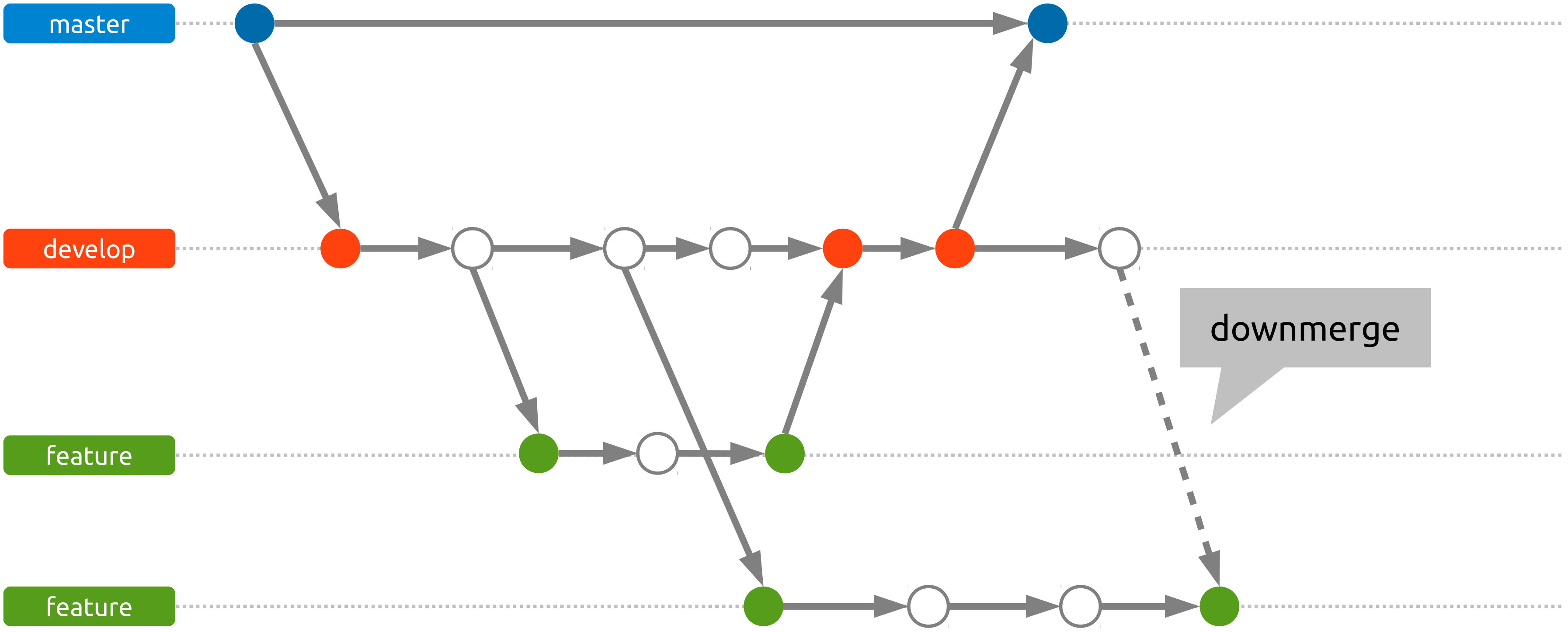


```
git merge develop
```

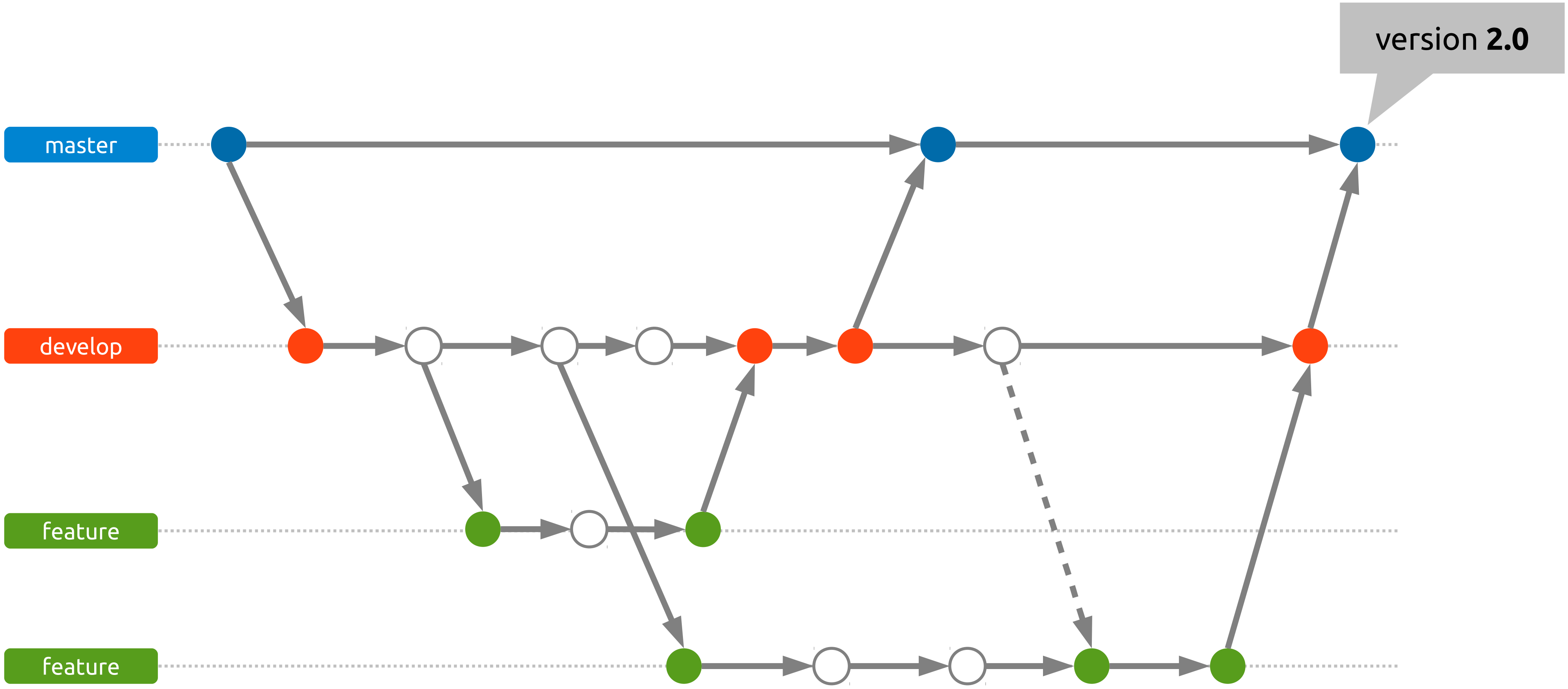
basics stages **branch** **merge** remotes advanced

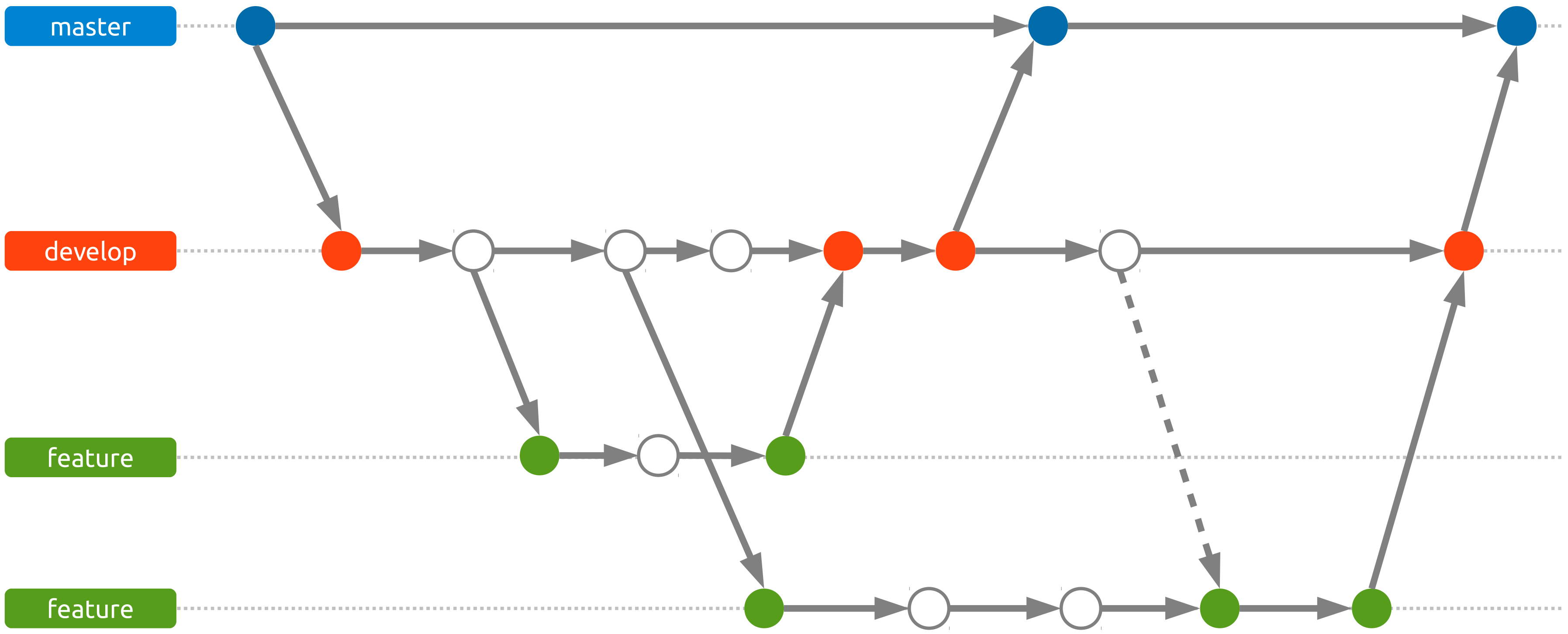


basics stages **branch** **merge** remotes advanced



`git merge develop`


version 2.0
master
develop
feature
feature


master
develop
feature
feature

basics

stages

branch

merge

remotes

advanced

working directory

staging area

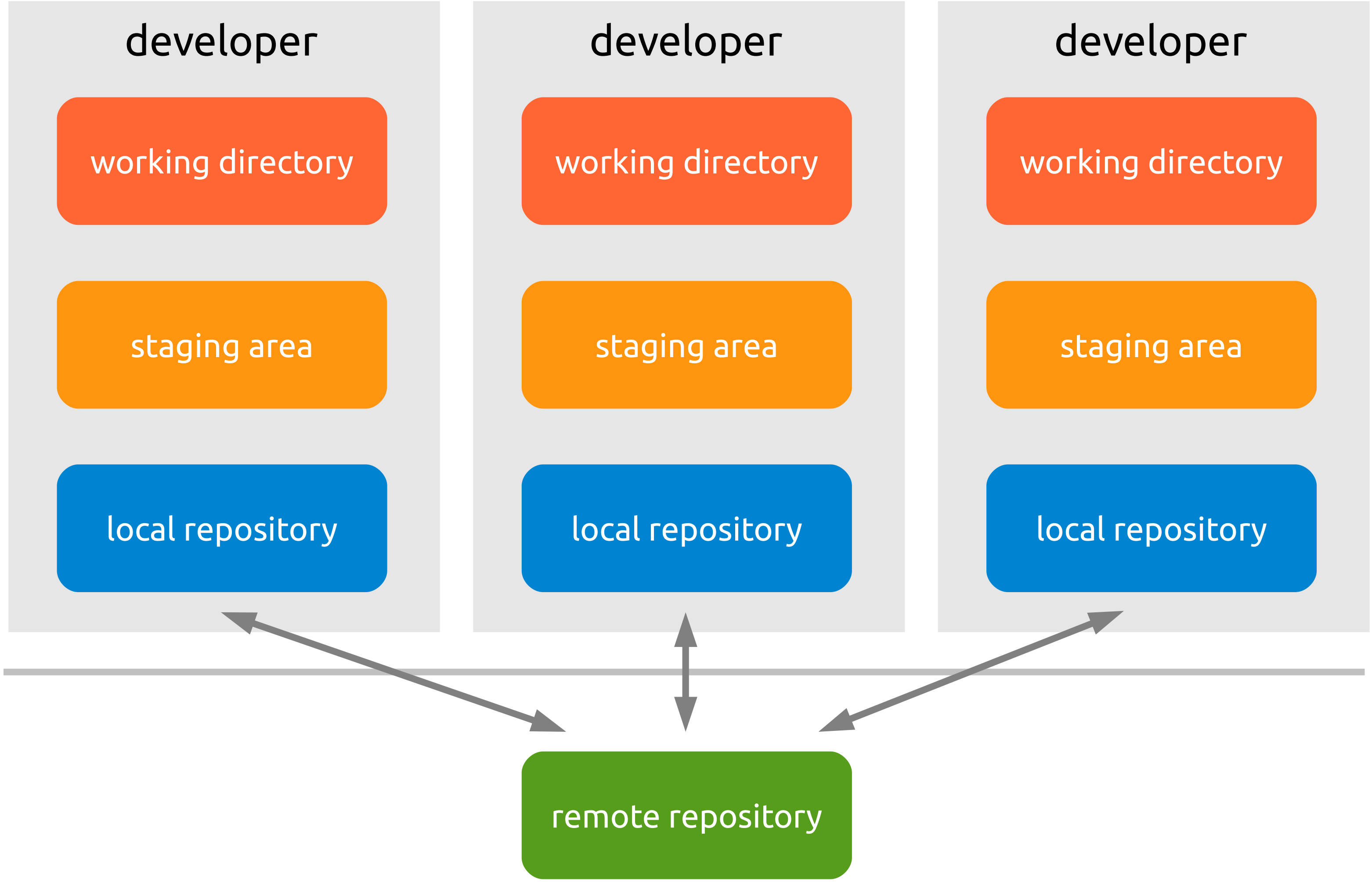
repository

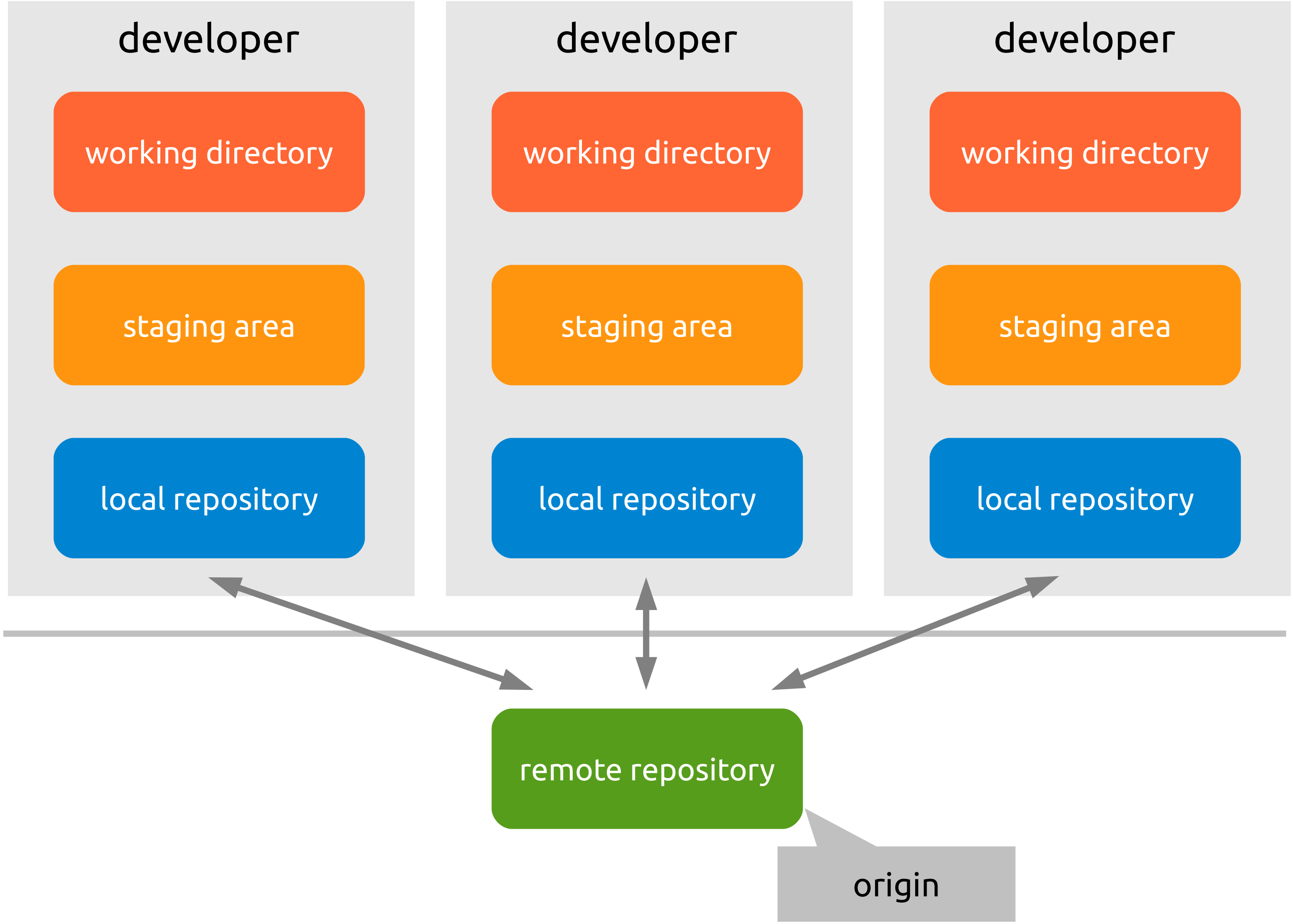
working directory

staging area

local repository

remote repository


developer
working directory
staging area
local repository
developer
working directory
staging area
local repository
developer
working directory
staging area
local repository
remote repository



developer

working directory

staging area

local repository

developer

working directory

staging area

local repository

developer

working directory

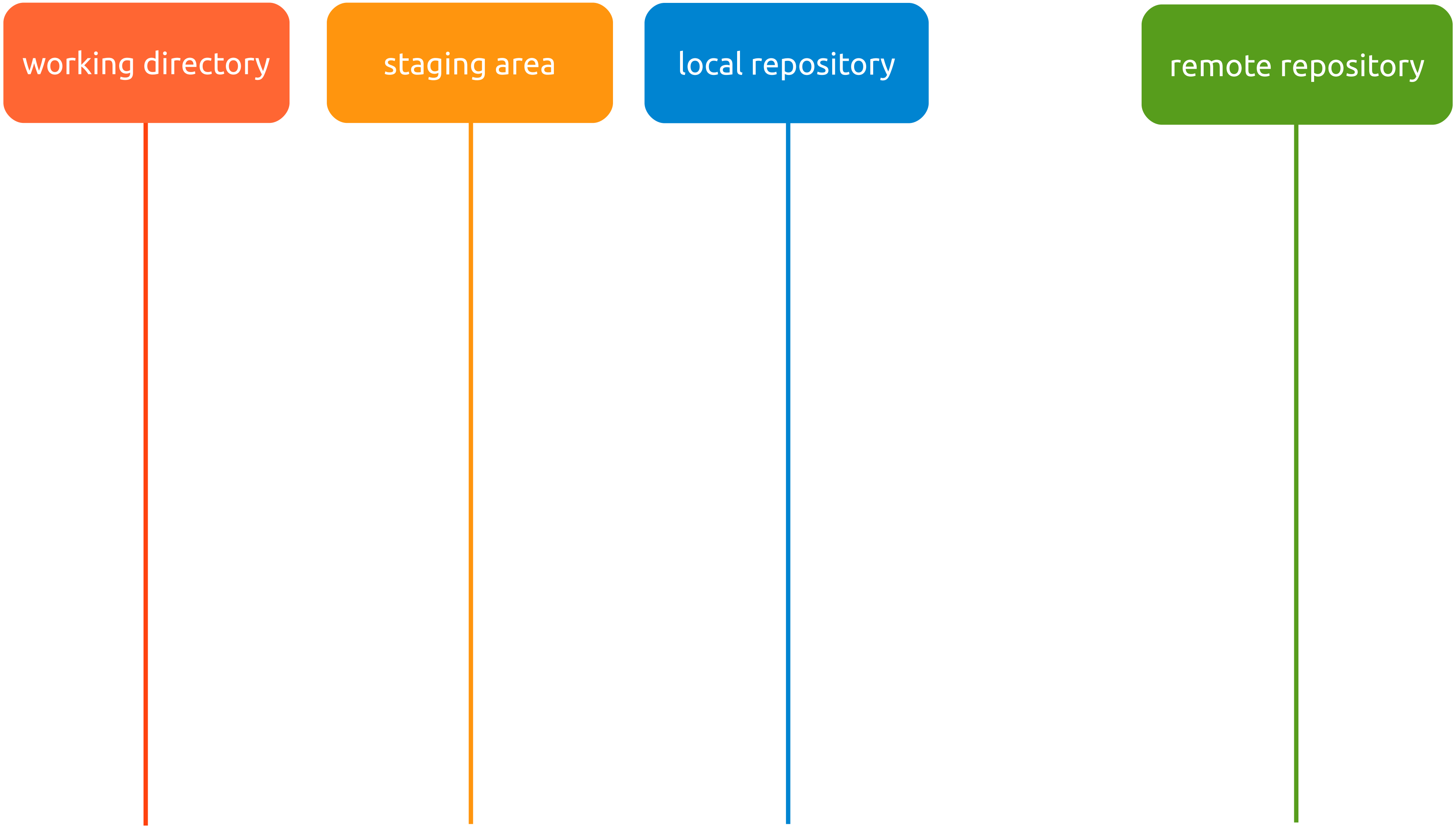
staging area

local repository

remote repository

origin

basics stages branch merge **remotes** advanced

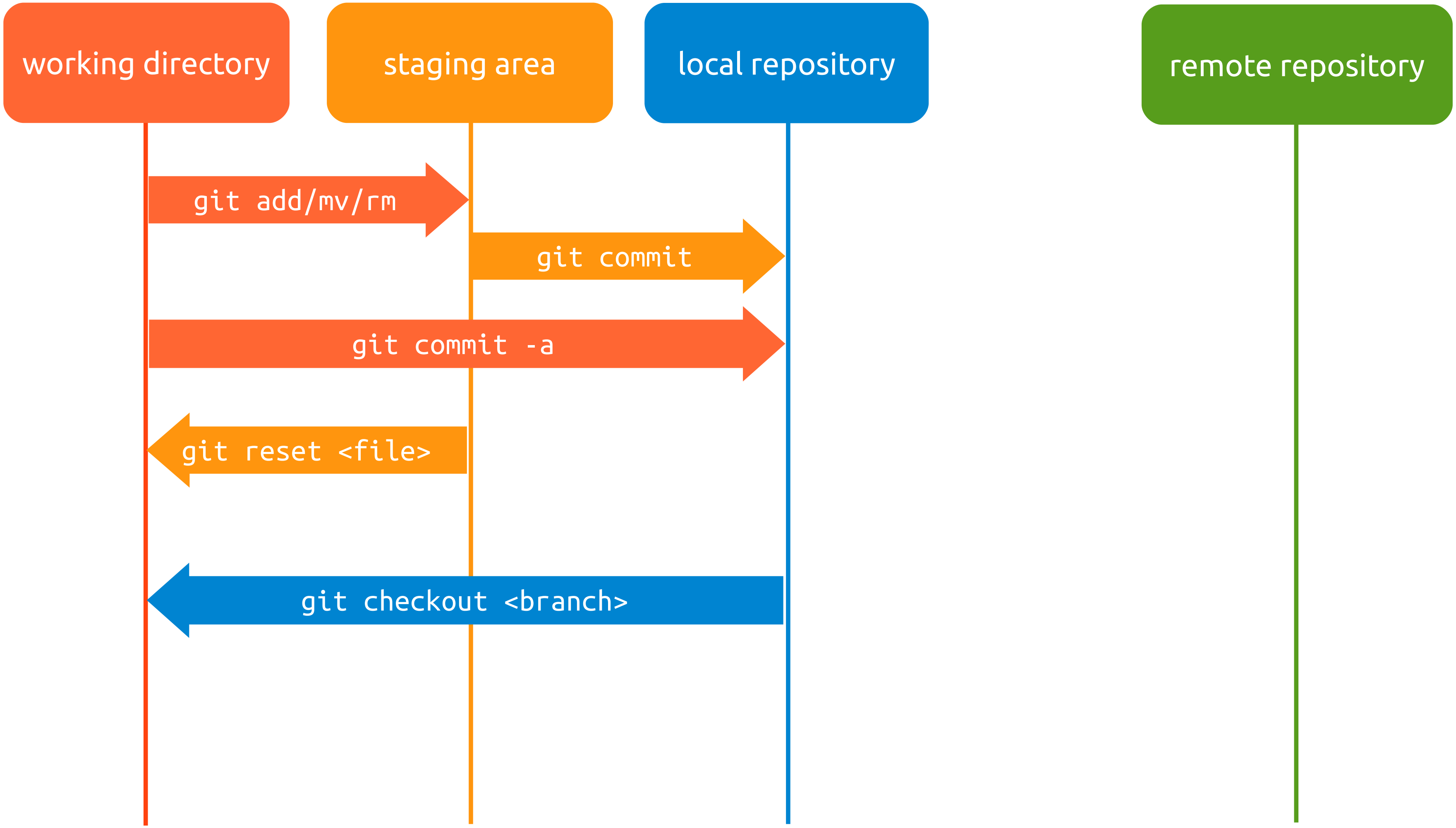


working directory

staging area

local repository

remote repository



working directory

staging area

local repository

remote repository

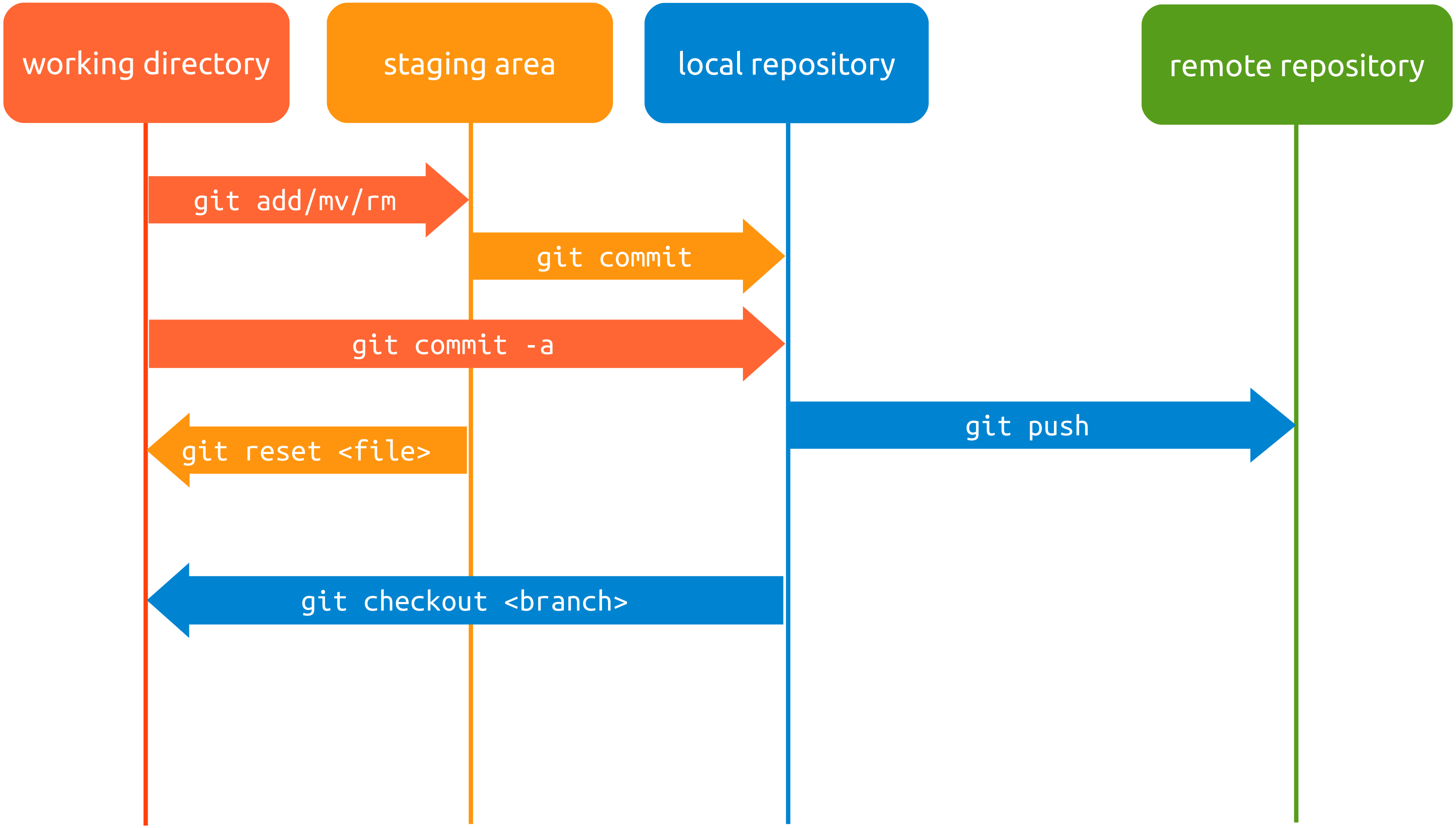
git add/mv/rm

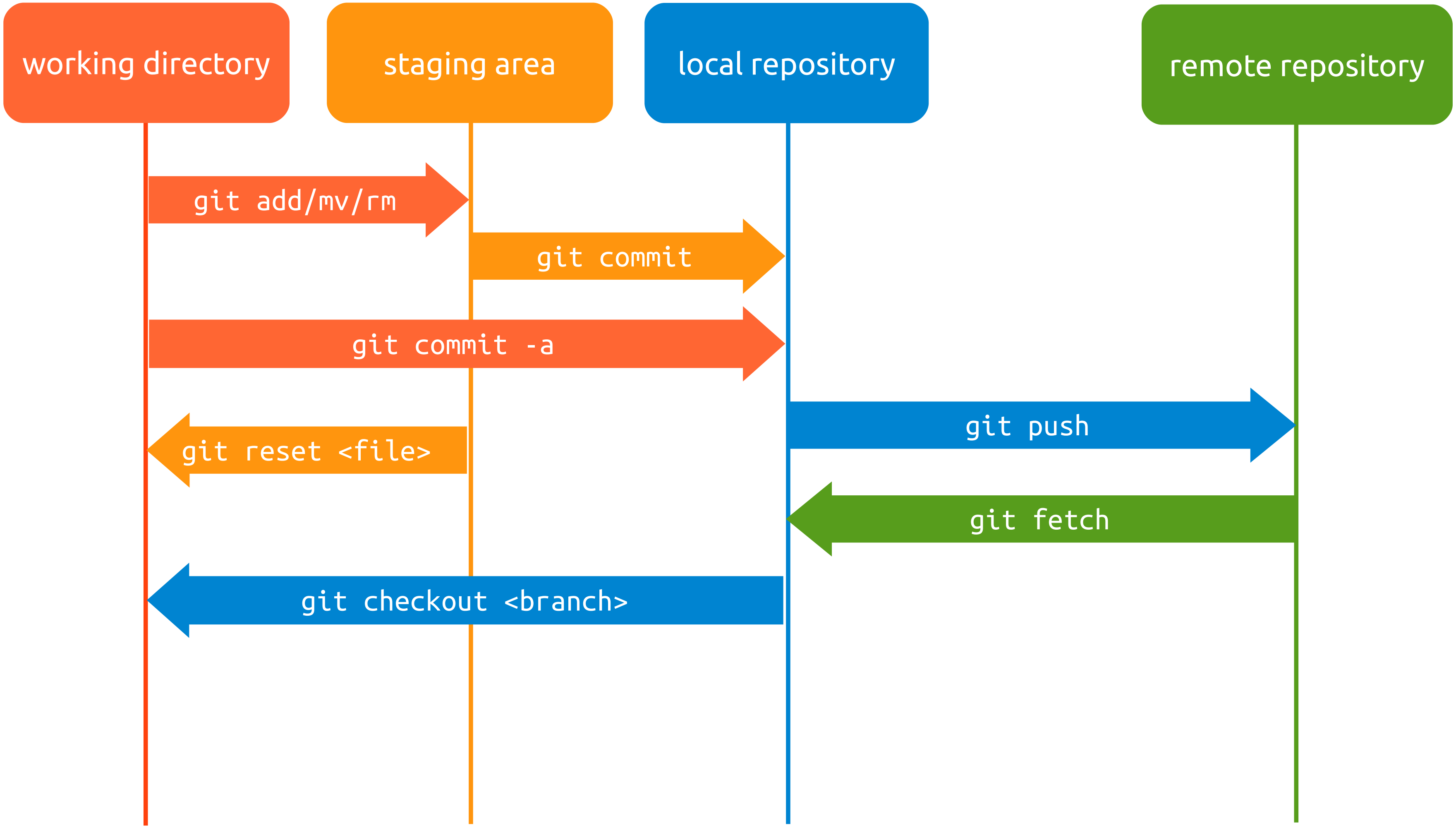
git commit

git commit -a

git reset <file>

git checkout <branch>


working directory
staging area
local repository
remote repository
git add/mv/rm
git commit
git commit -a
git push
git reset <file>
git checkout <branch>



working directory

staging area

local repository

remote repository

git add/mv/rm

git commit

git commit -a

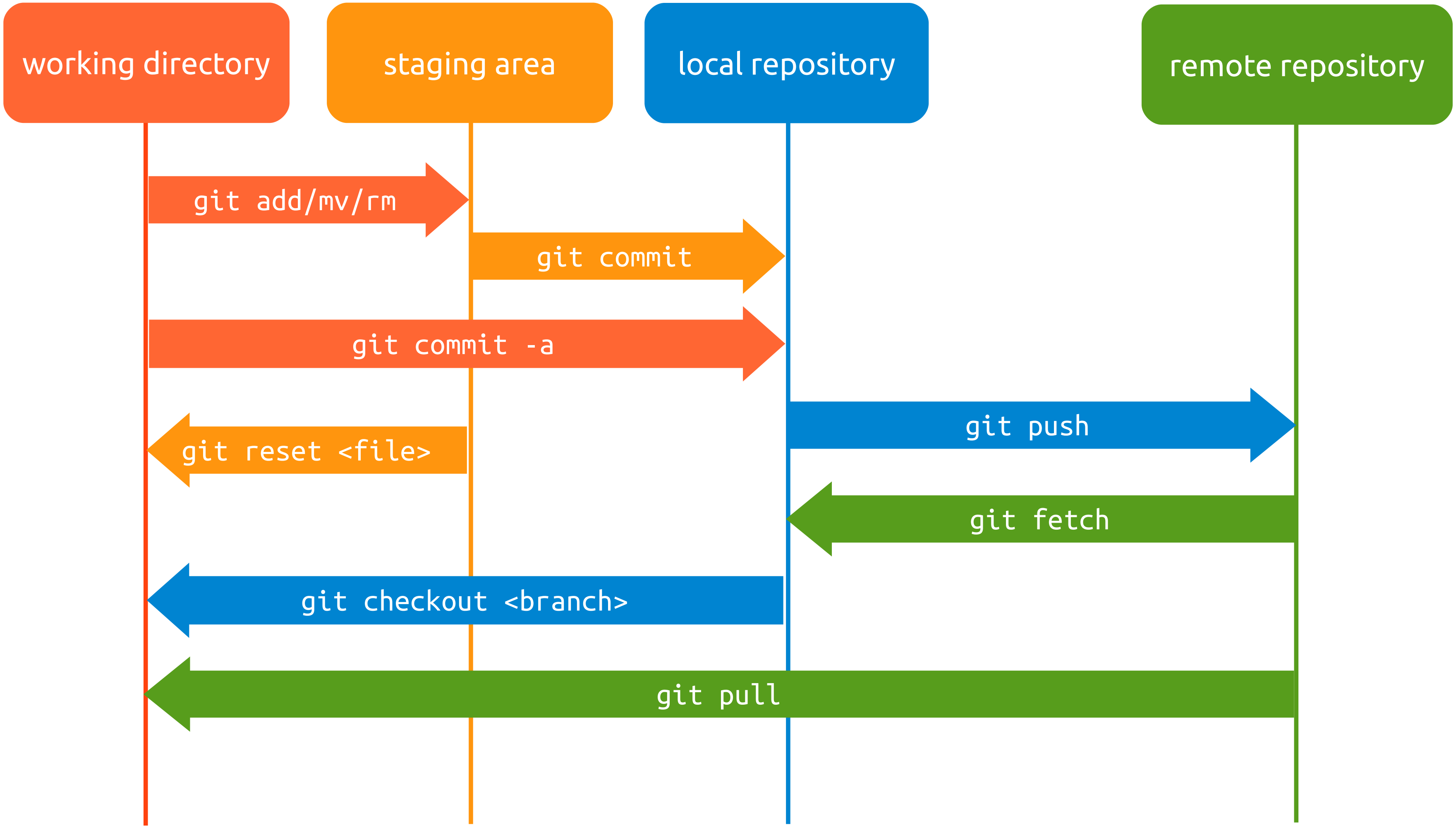
git push

git reset <file>

git fetch

git checkout <branch>

basics stages branch merge remotes advanced



working directory

staging area

local repository

remote repository

git add/mv/rm

git commit

git commit -a

git push

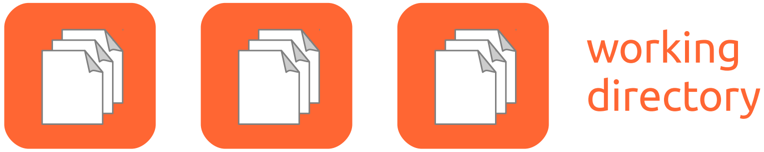
git reset <file>

git fetch

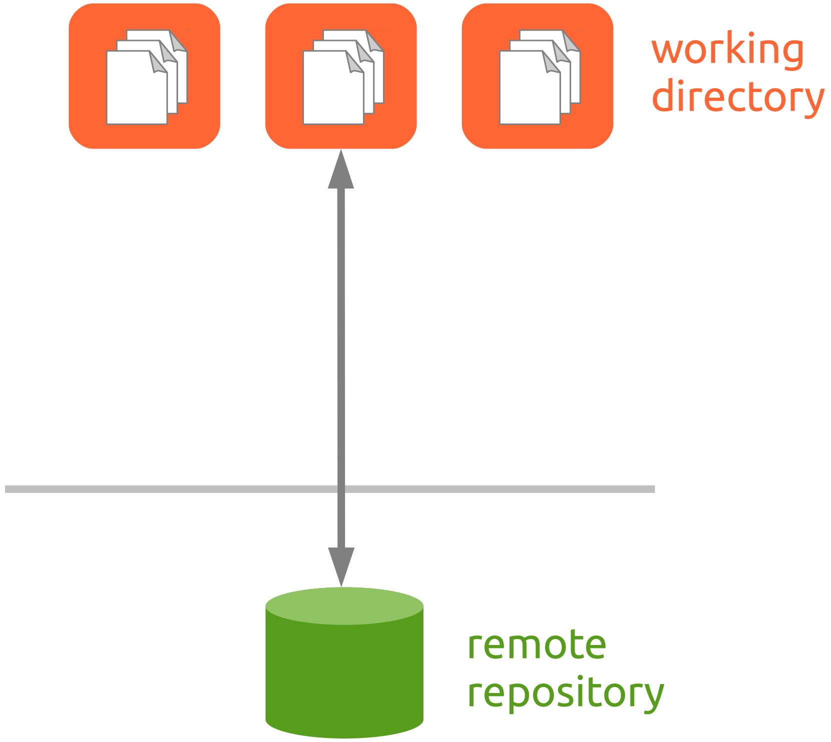
git checkout <branch>

git pull

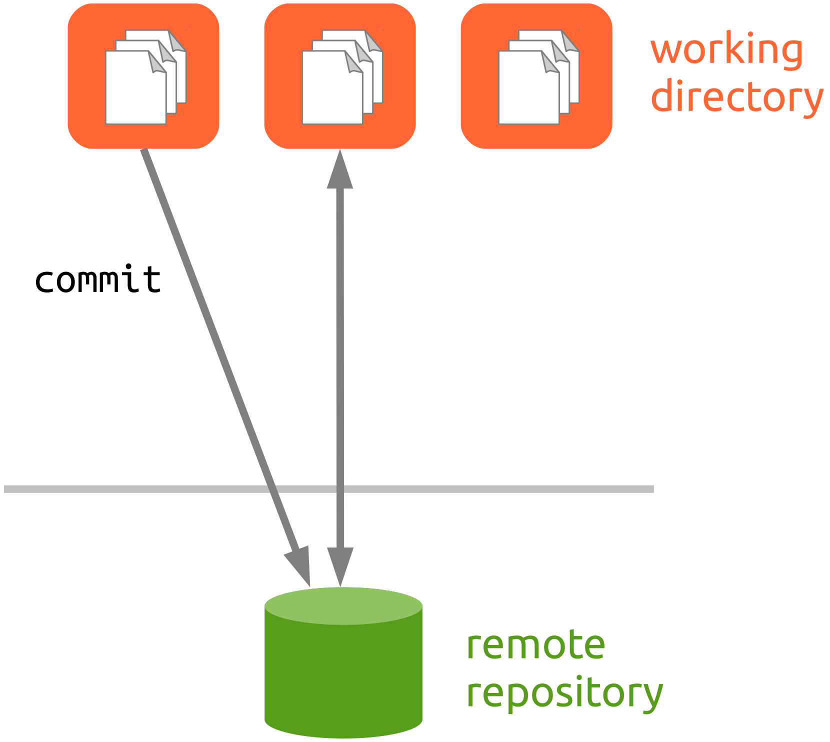
Subversion



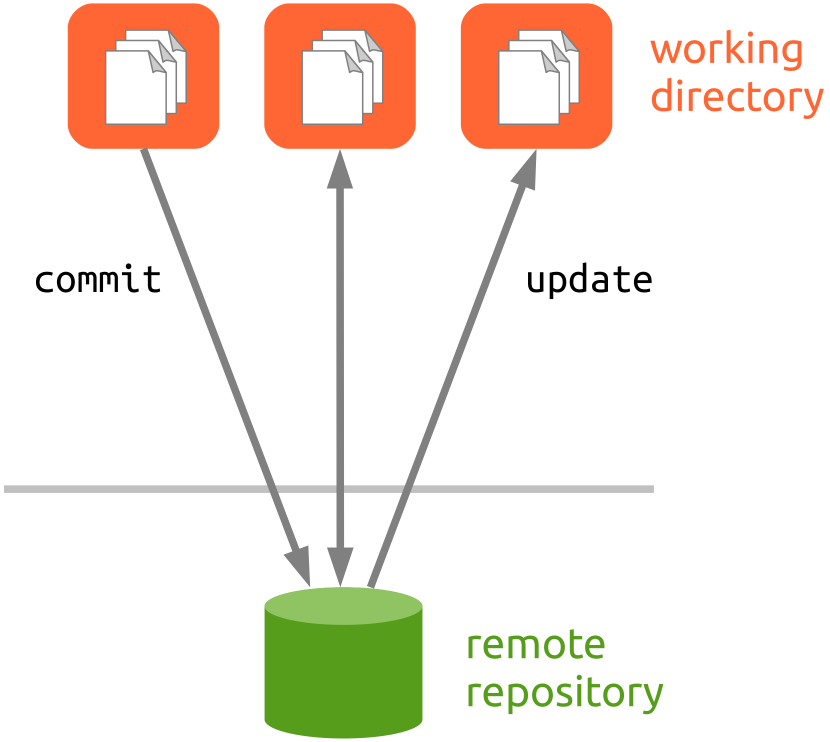
Subversion



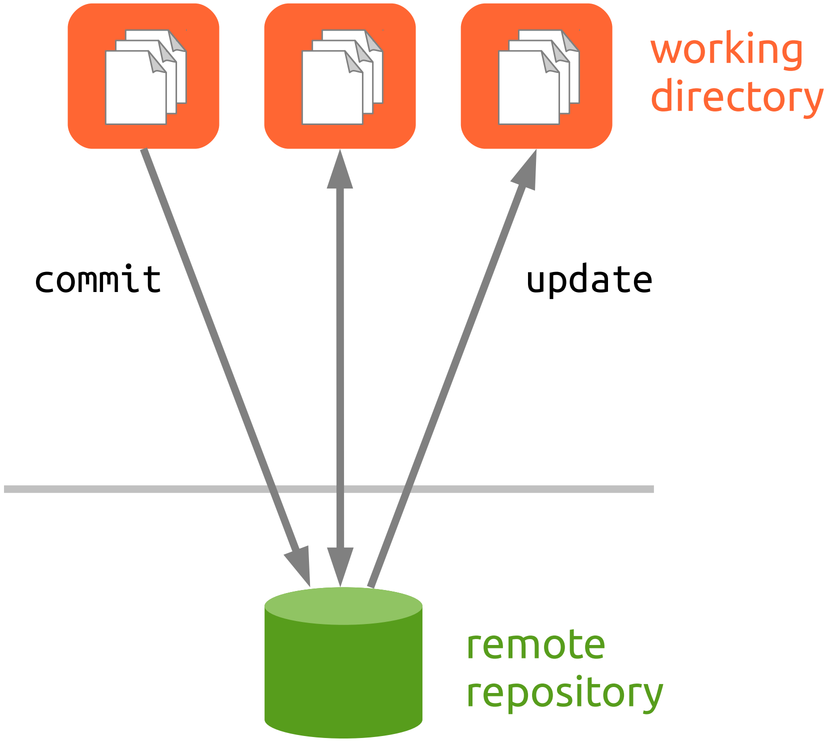
Subversion



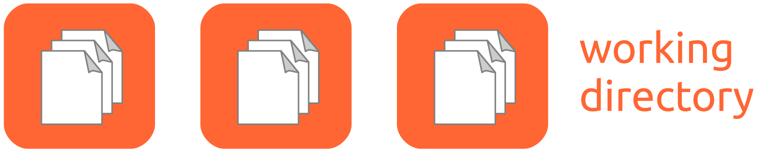
Subversion



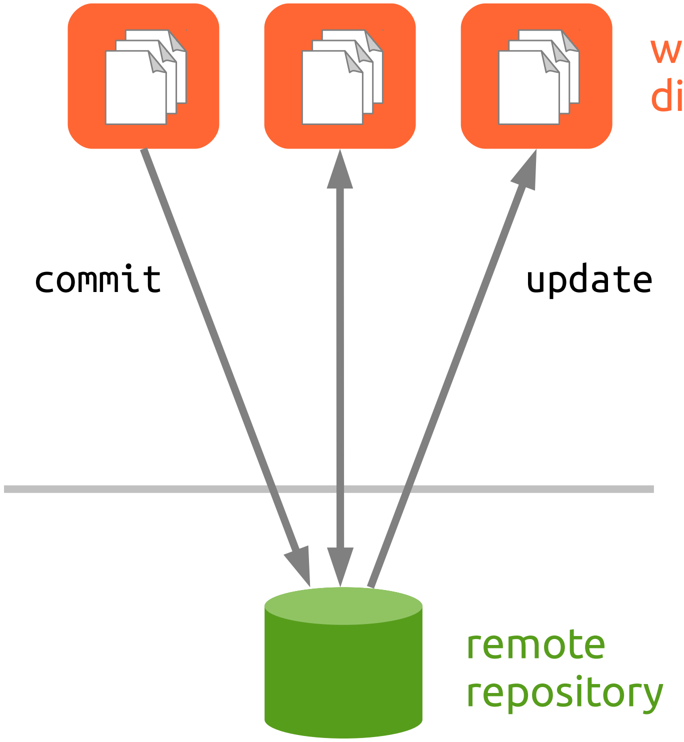
Subversion



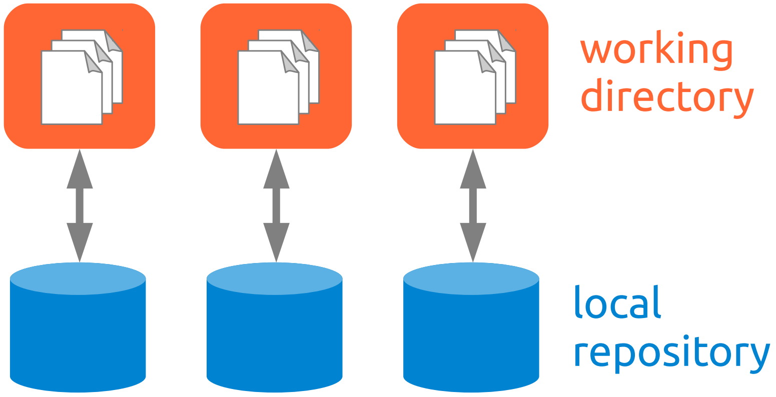
Git



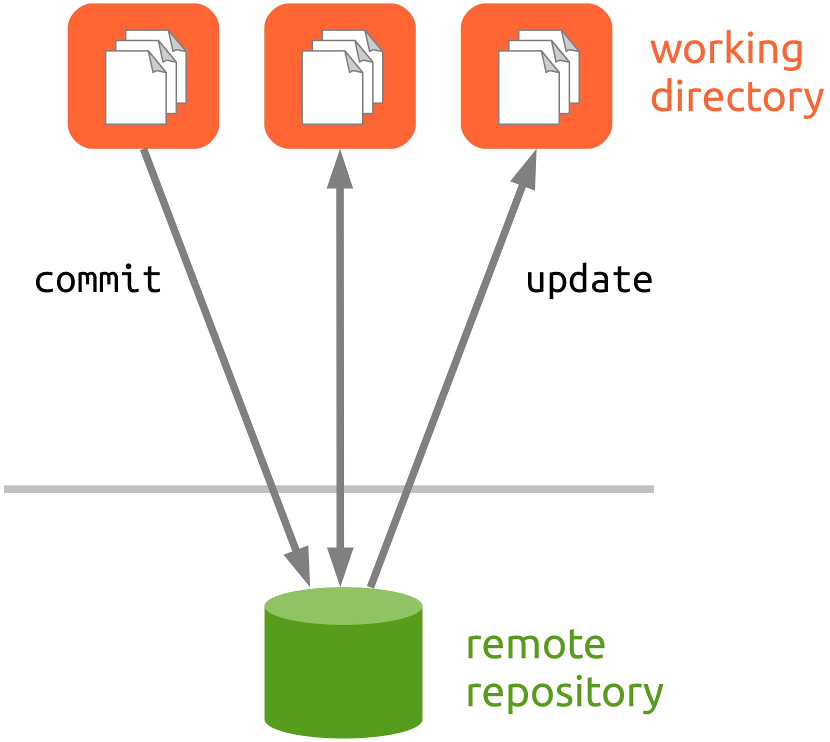
Subversion



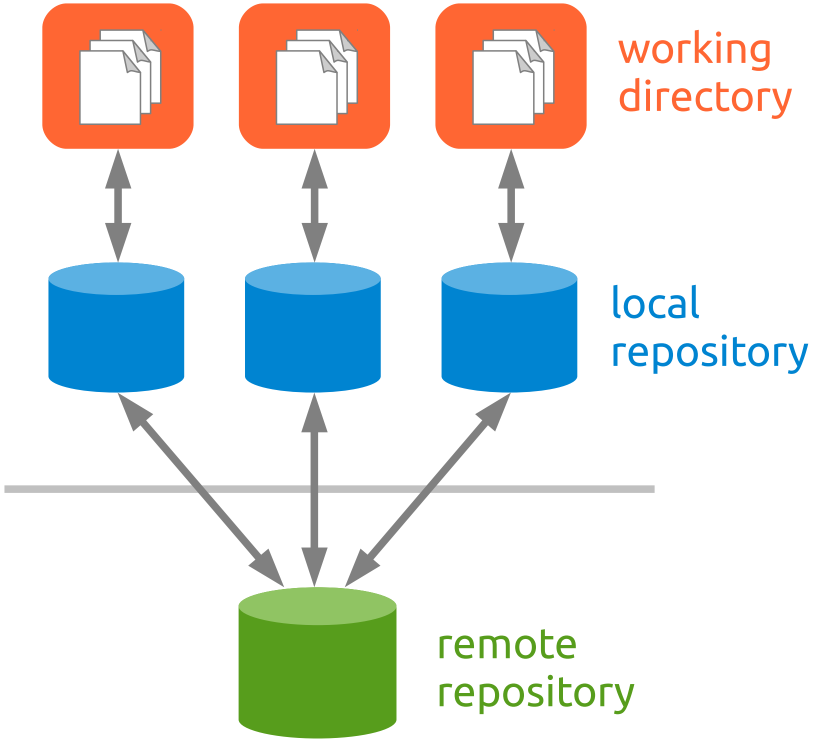
Git



Subversion



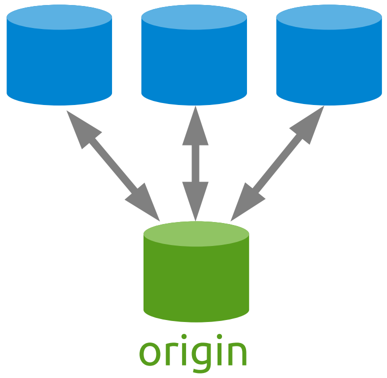
Git



Distribution models

Distribution models

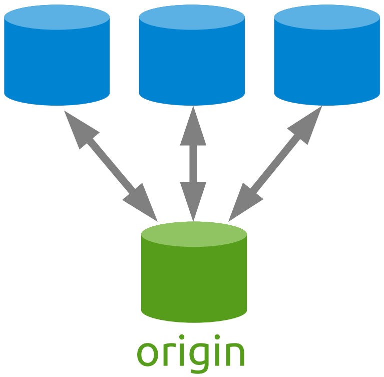
Centralized workflow



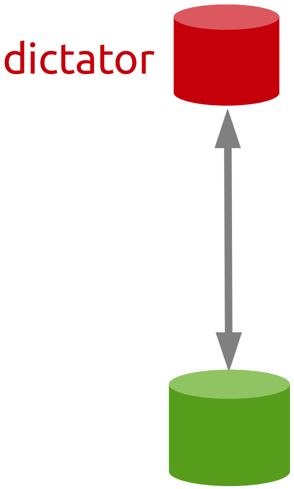
origin

Distribution models

Centralized workflow

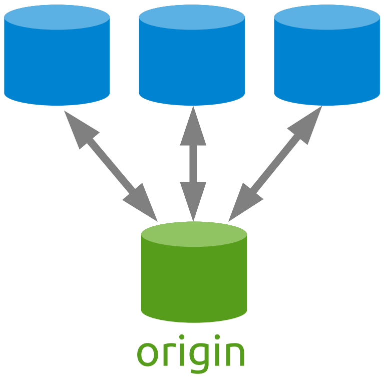


GitHub Forking/Pull Request



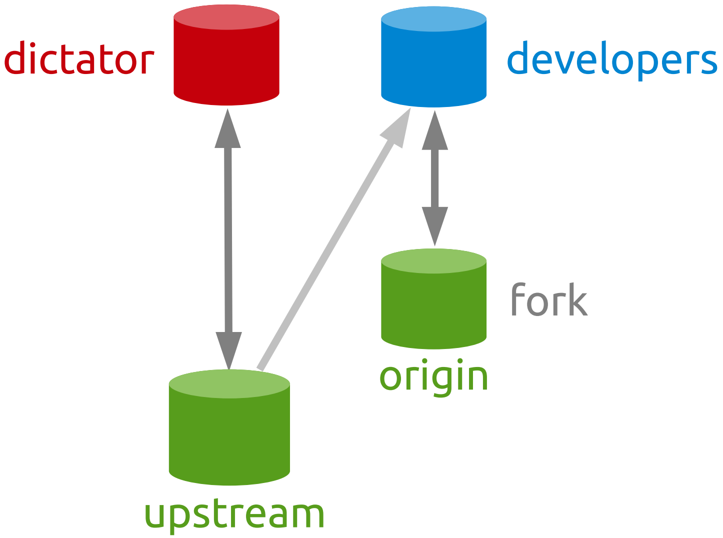
Distribution models

Centralized workflow



origin

GitHub Forking/Pull Request



dictator

developers

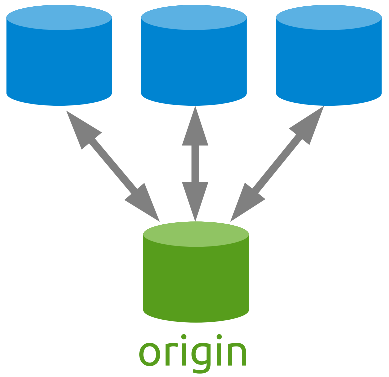
fork

origin

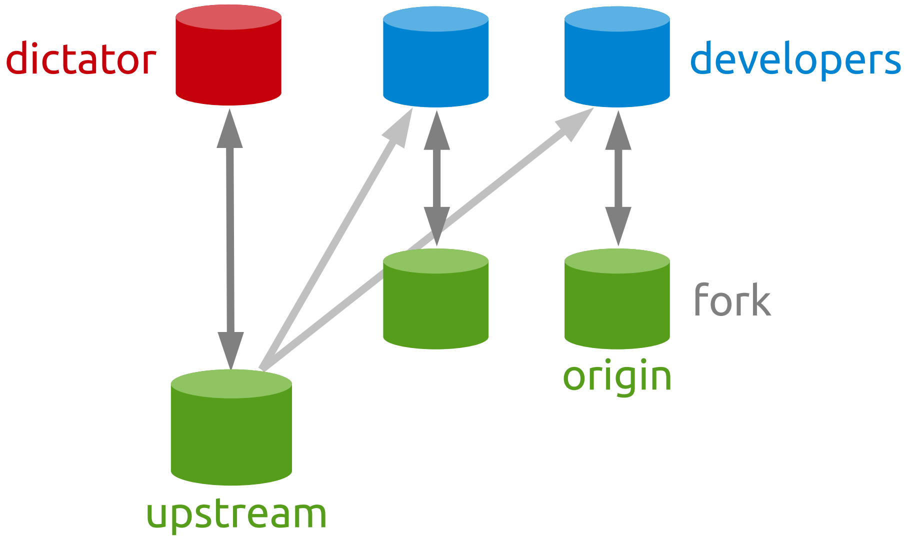
upstream

Distribution models

Centralized workflow

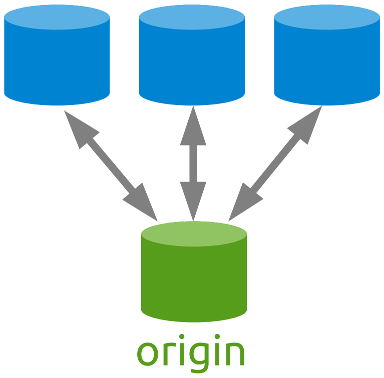


GitHub Forking/Pull Request

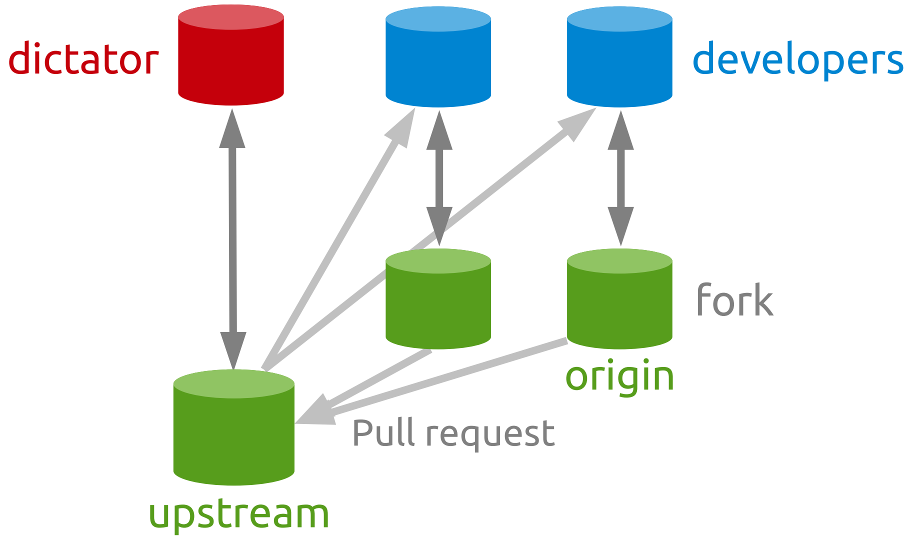


Distribution models

Centralized workflow

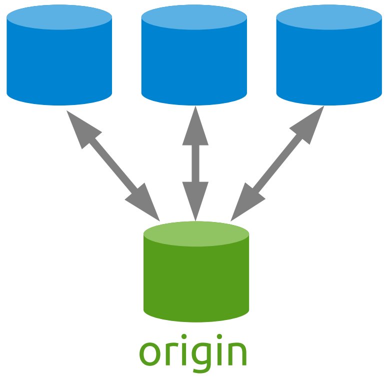


GitHub Forking/Pull Request

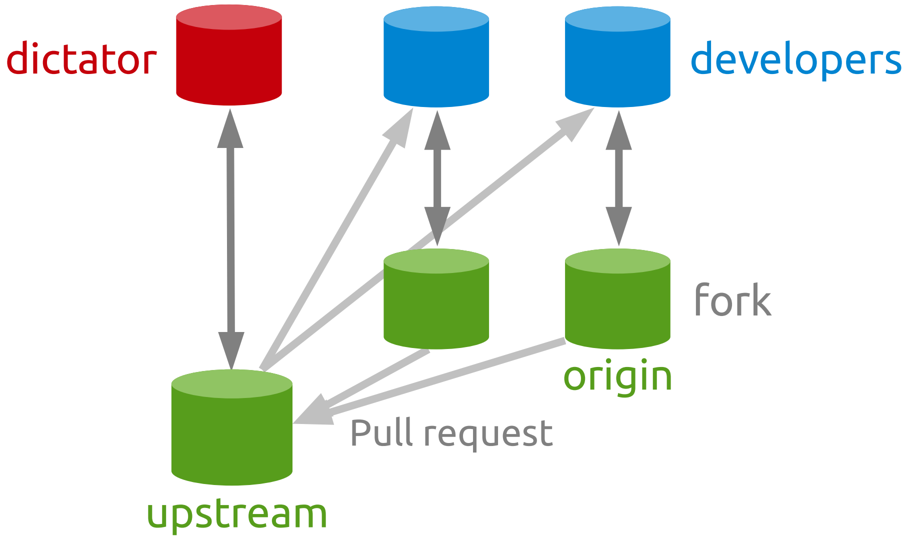


Distribution models

Centralized workflow



GitHub Forking/Pull Request



Many others

Managing remotes

```
$ git remote -v
origin    git@github.com:thrau/openengsb-framework.git (fetch)
origin    git@github.com:thrau/openengsb-framework.git (push)
upstream  git@github.com:openengsb/openengsb-framework.git (fetch)
upstream  git@github.com:openengsb/openengsb-framework.git (push)
```

```
$ git remote add <name> <url>
```

```
$ git remote rm <name>
```

Remote tracking branches

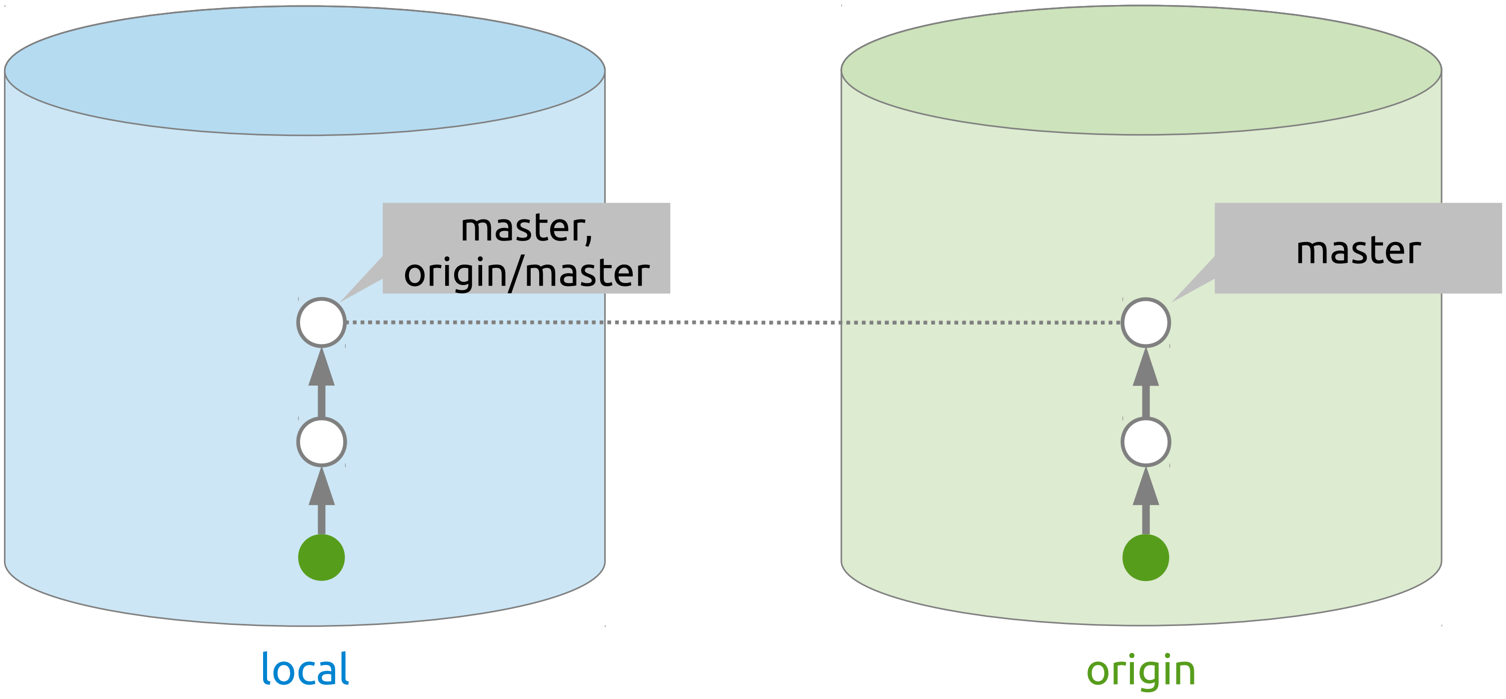
```
$ git branch -a
```

Remote tracking branches

```
$ git branch -a
* master
  my-local-feature
  remotes/origin/master
```

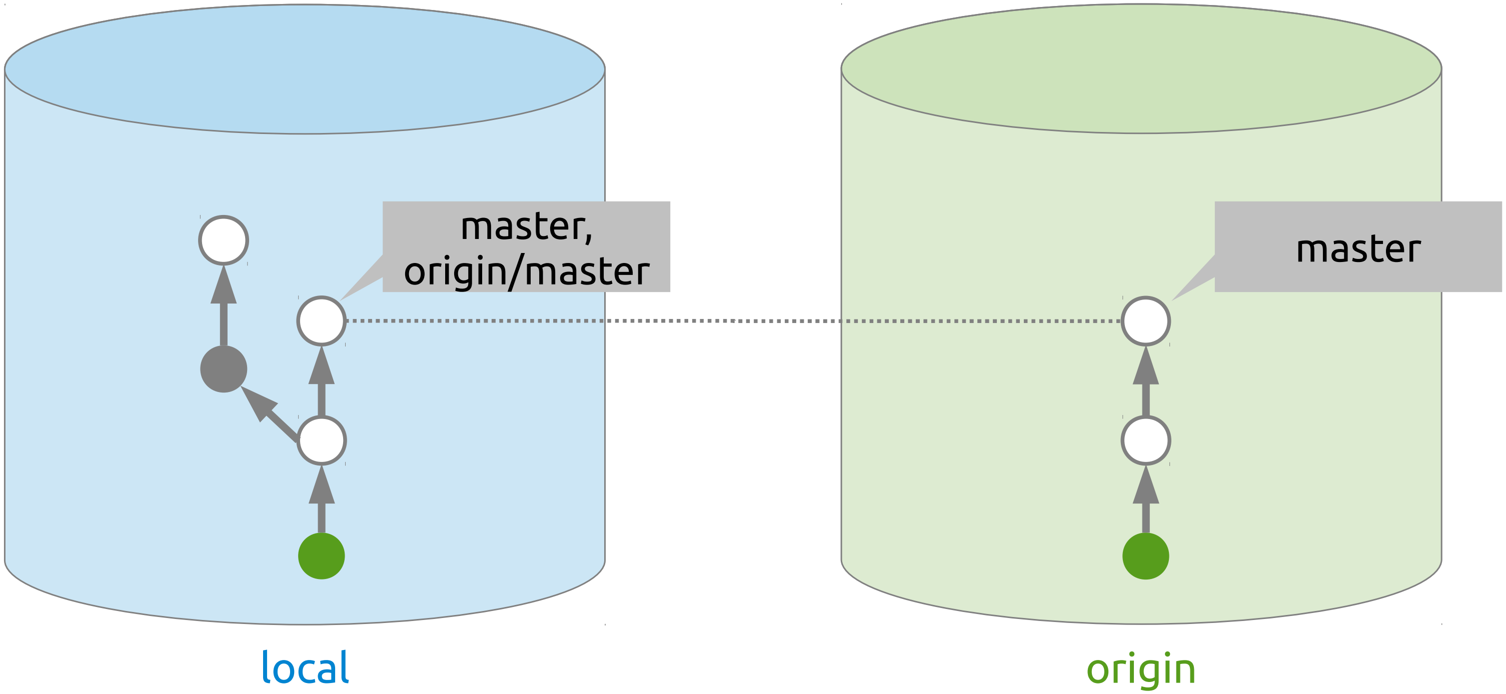
Remote tracking branches

```
$ git branch -a
* master
  my-local-feature
  remotes/origin/master
```



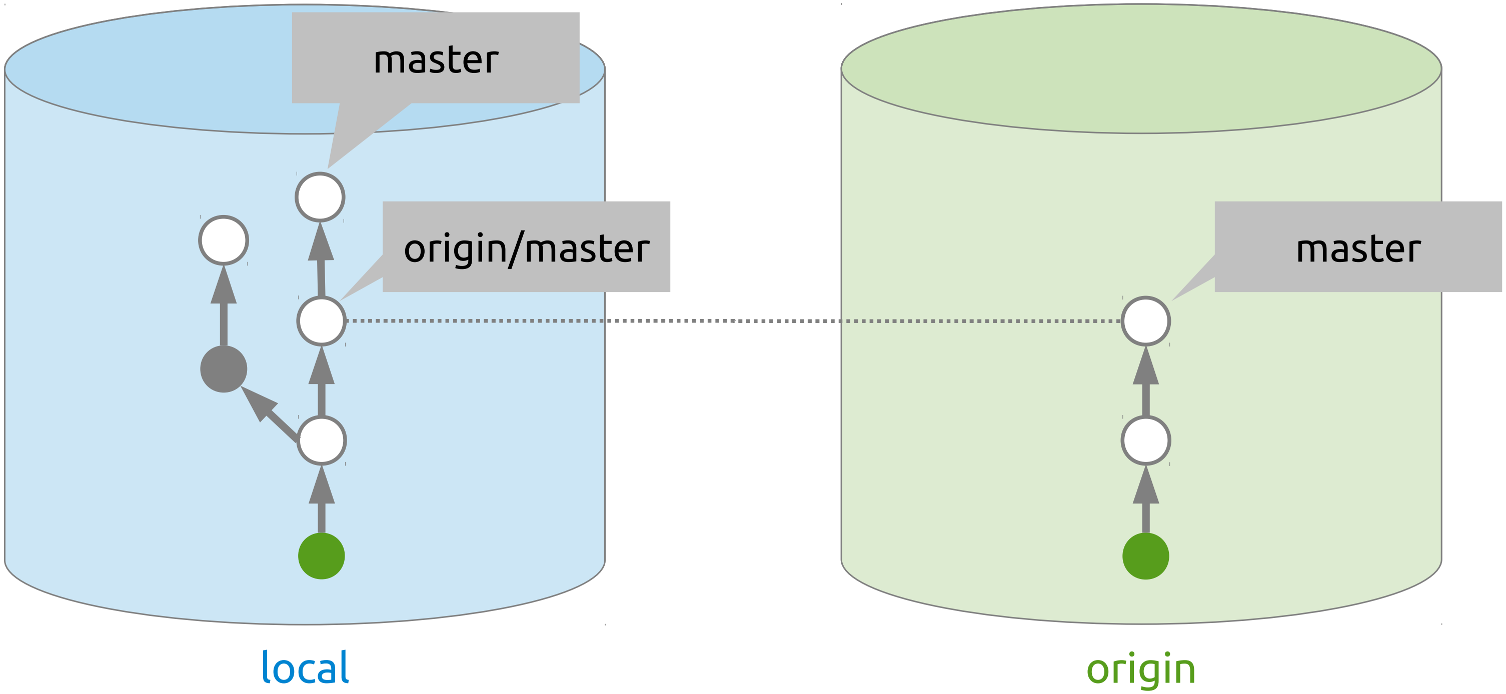
Remote tracking branches

```
$ git branch -a
* master
  my-local-feature
  remotes/origin/master
```



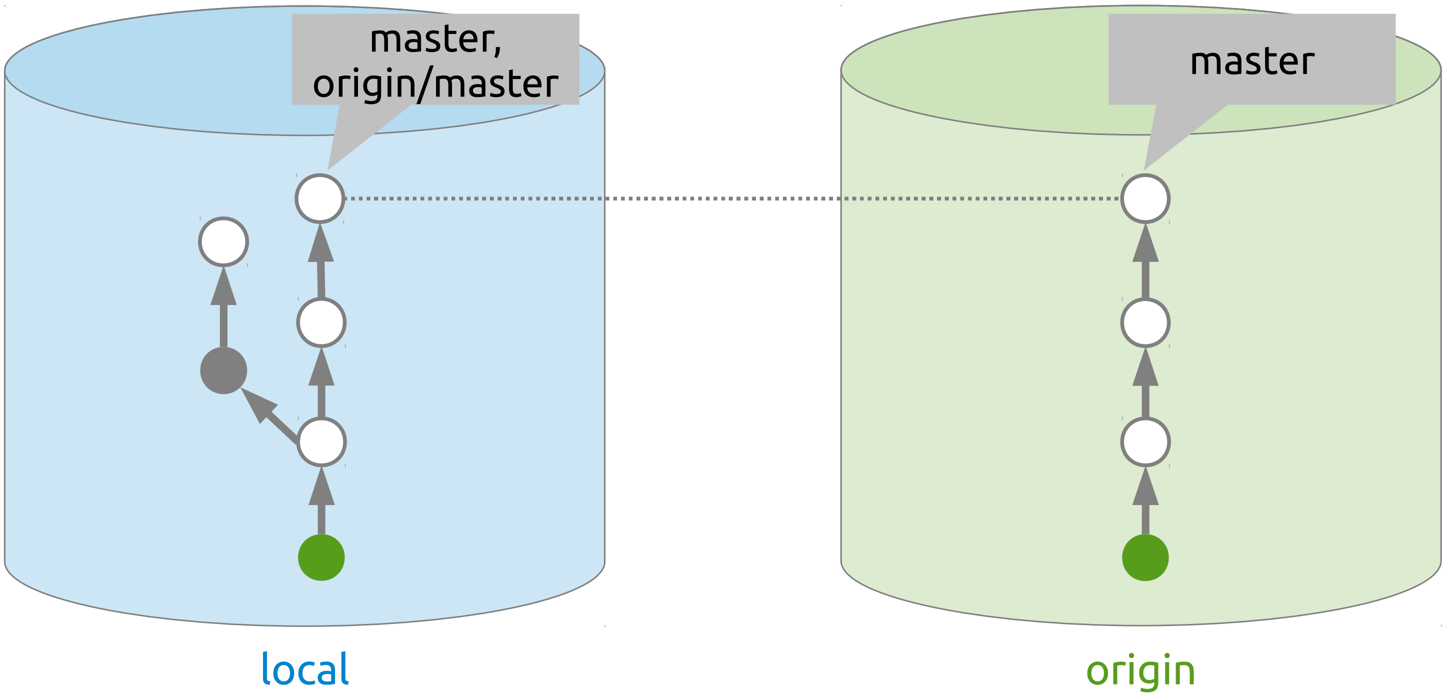
Remote tracking branches

```
$ git branch -a
* master
  my-local-feature
  remotes/origin/master
```



Remote tracking branches

```
$ git push
```



Delete remote branches

```
$ git push origin :branchname
```

Delete local tracking branches

```
$ git fetch origin --prune
```

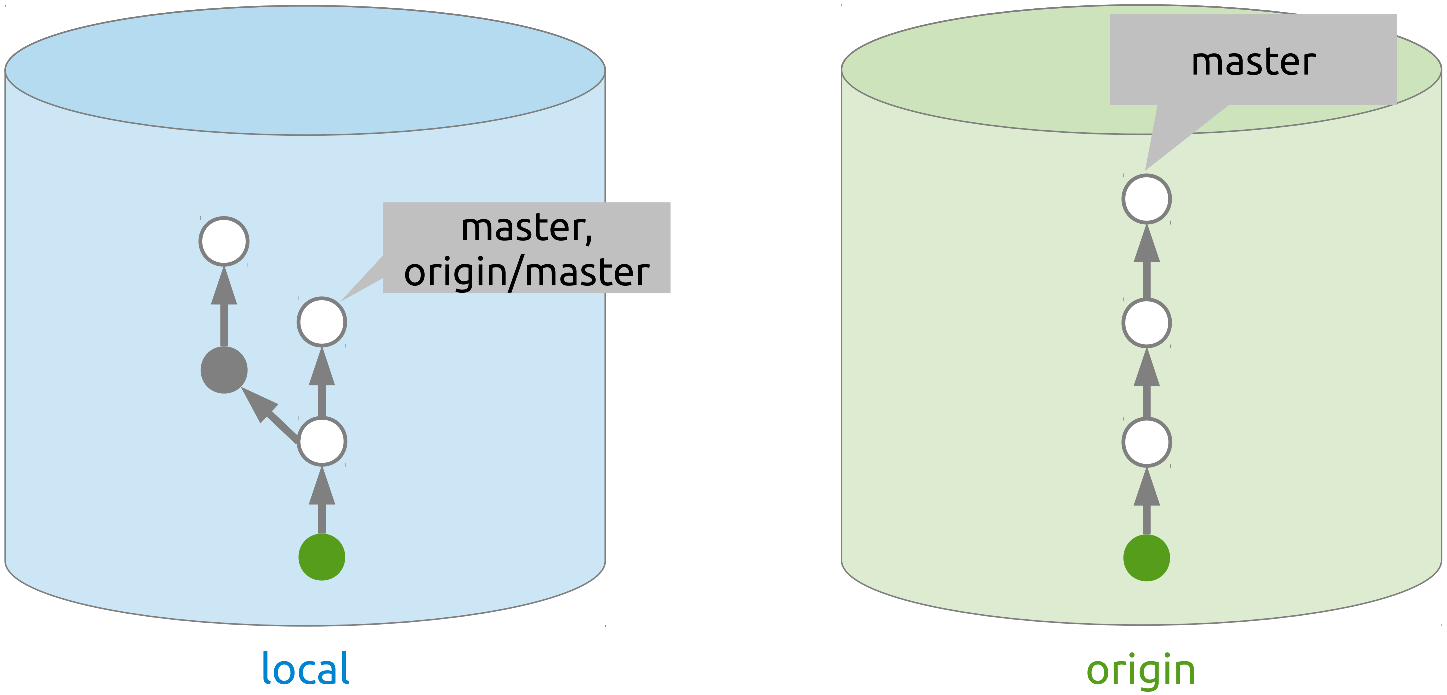
Dealing with remote conflicts

Dealing with remote conflicts

```
$ git push origin master
```

Dealing with remote conflicts

```
$ git push origin master
```



Dealing with remote conflicts

```
$ git push origin master
To ssh://thomas@localhost/home/thomas/git-remote
 ! [rejected]        HEAD -> master (non-fast-forward)
error: failed to push some refs to 'ssh://thomas@localhost/home/thomas/git-remote'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Merge the remote changes (e.g. 'git pull')
hint: before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Dealing with remote conflicts

```
$ git push origin master
To ssh://thomas@localhost/home/thomas/git-remote
 ! [rejected]        HEAD -> master (non-fast-forward)
error: failed to push some refs to 'ssh://thomas@localhost/home/thomas/git-remote'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Merge the remote changes (e.g. 'git pull')
hint: before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

```
$ git pull origin master
```

Dealing with remote conflicts

```
$ git push origin master
To ssh://thomas@localhost/home/thomas/git-remote
 ! [rejected]        HEAD -> master (non-fast-forward)
error: failed to push some refs to 'ssh://thomas@localhost/home/thomas/git-remote'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Merge the remote changes (e.g. 'git pull')
hint: before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

```
$ git pull origin master
```

… fix merge conflicts ...

Dealing with remote conflicts

```
$ git push origin master
To ssh://thomas@localhost/home/thomas/git-remote
 ! [rejected]        HEAD -> master (non-fast-forward)
error: failed to push some refs to 'ssh://thomas@localhost/home/thomas/git-remote'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Merge the remote changes (e.g. 'git pull')
hint: before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

```
$ git pull origin master
```

… fix merge conflicts …

```
$ git commit -am "Merge remote branch 'master'"
$ git push origin master
```

Dealing with remote conflicts

```
$ git push origin master
To ssh://thomas@localhost/home/thomas/git-remote
 ! [rejected]        HEAD -> master (non-fast-forward)
error: failed to push some refs to 'ssh://thomas@localhost/home/thomas/git-remote'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Merge the remote changes (e.g. 'git pull')
hint: before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

```
$ git pull origin master
```

… fix merge conflicts ...

```
$ git commit -am "Merge remote branch 'develop'"
$ git push origin master
```

Produces a merge commit

Dealing with remote conflicts

```
$ git push origin master
To ssh://thomas@localhost/home/thomas/git-remote
 ! [rejected]        HEAD -> master (non-fast-forward)
error: failed to push some refs to 'ssh://thomas@localhost/home/thomas/git-remote'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Merge the remote changes (e.g. 'git pull')
hint: before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

```
$ git pull --rebase origin master
```

repeat

… fix merge conflicts …

```
$ git rebase --continue
```

```
$ git push origin master
```

Rewrites the history.

basics stages branch merge **remotes** advanced

basics

stages

branch

merge

remotes

advanced

Rewriting history

Rewriting history

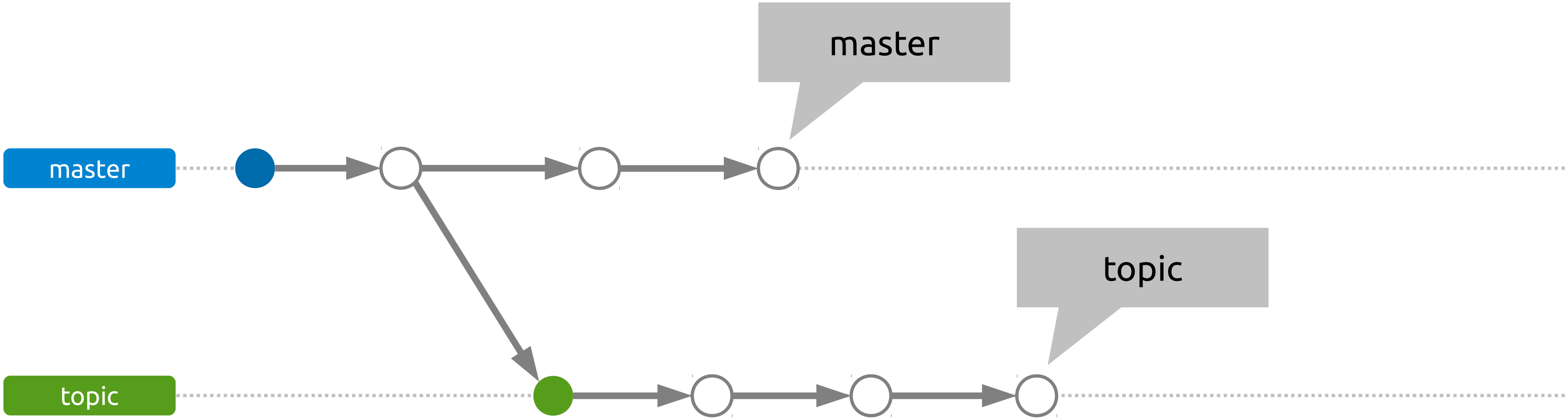
Rebase the current branch on to the tip of a different one

```
$ git rebase master
```

Rewriting history

Rebase the current branch on to the tip of a different one

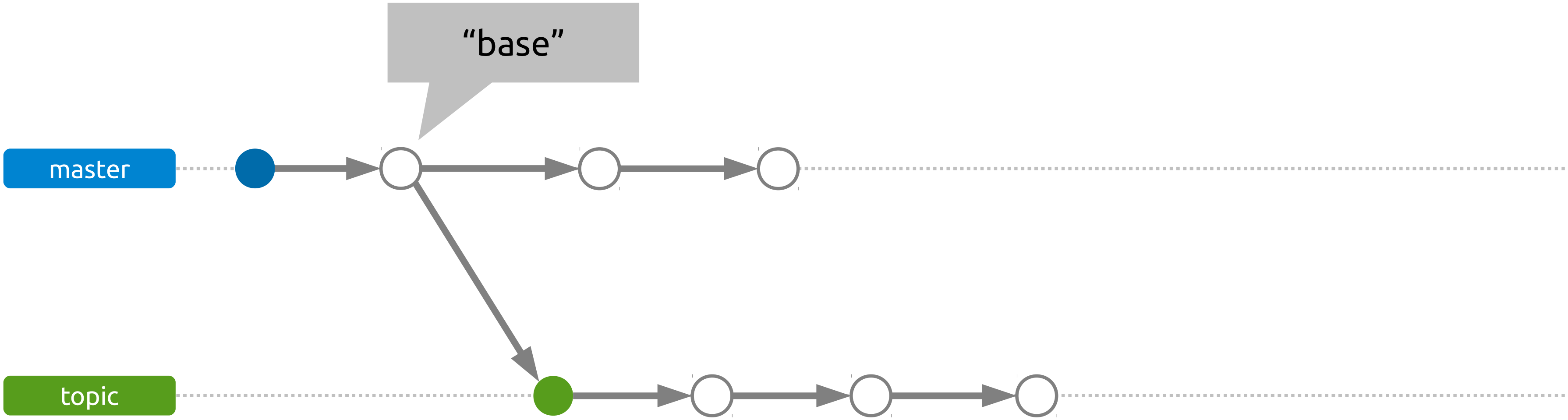
```
$ git rebase master
```



Rewriting history

Rebase the current branch on to the tip of a different one

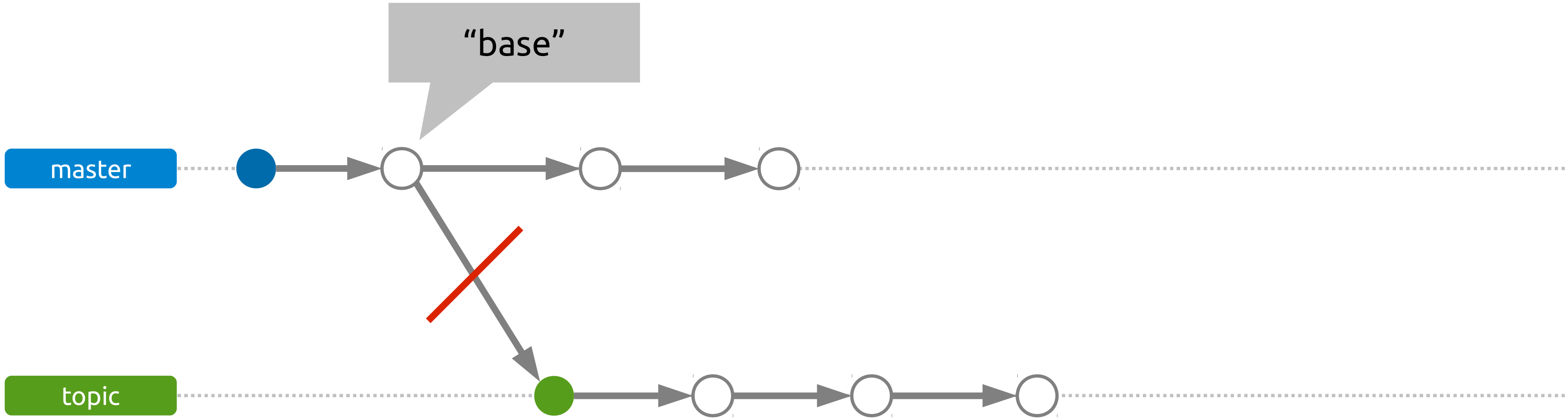
```
$ git rebase master
```



Rewriting history

Rebase the current branch on to the tip of a different one

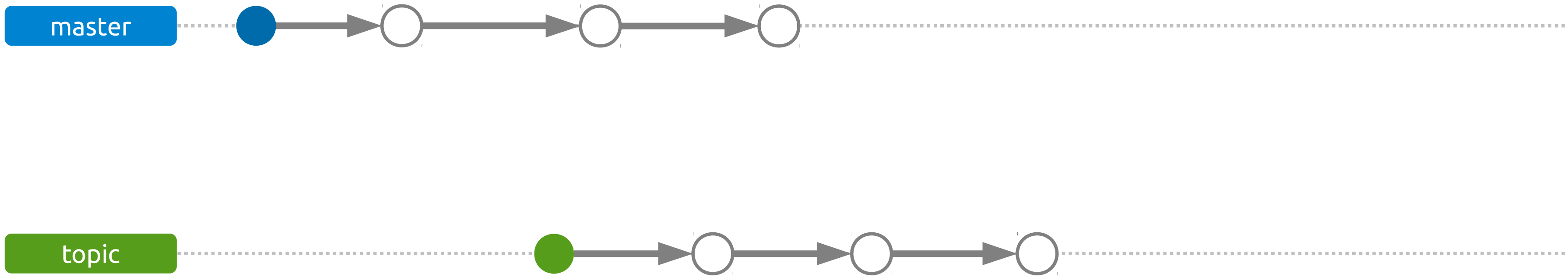
```
$ git rebase master
```



Rewriting history

Rebase the current branch on to the tip of a different one

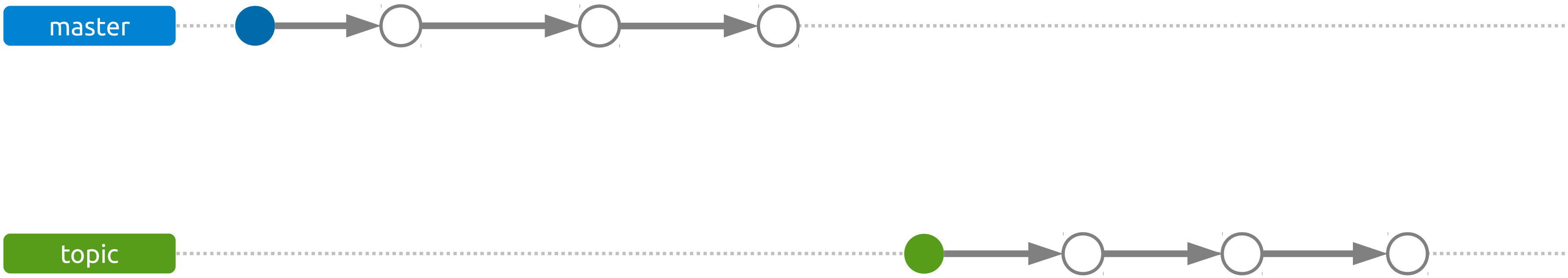
```
$ git rebase master
```



Rewriting history

Rebase the current branch on to the tip of a different one

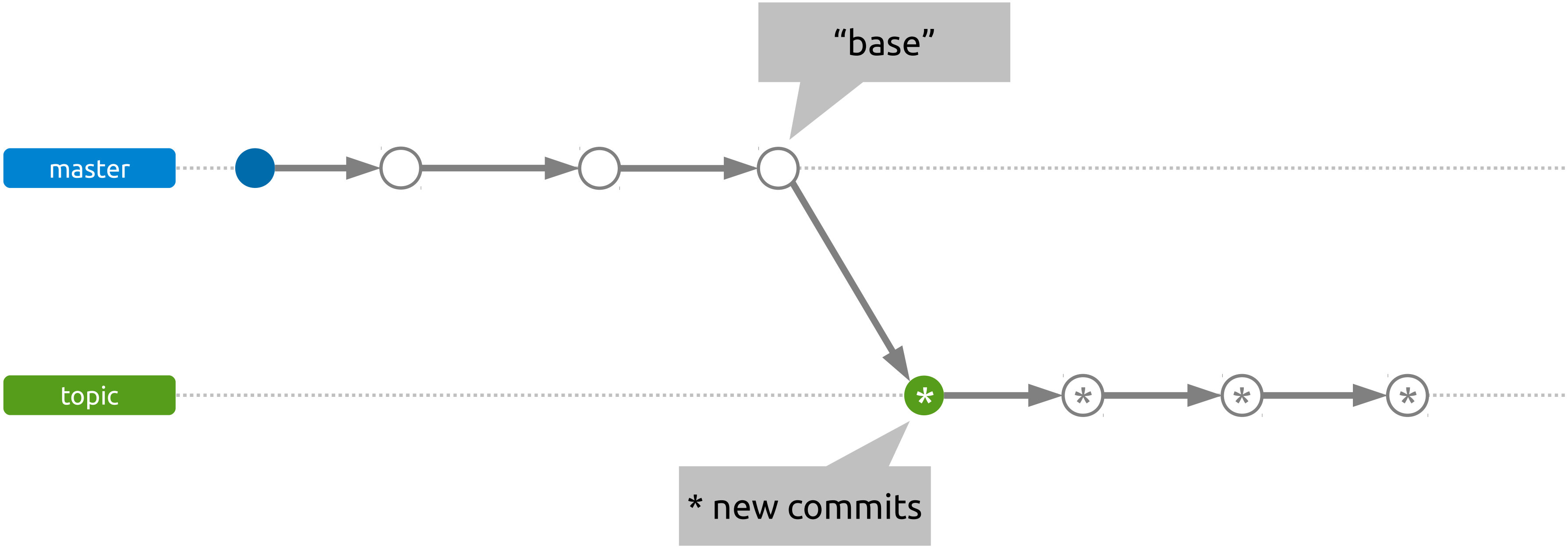
```
$ git rebase master
```



Rewriting history

Rebase the current branch on to the tip of a different one

```
$ git rebase master
```

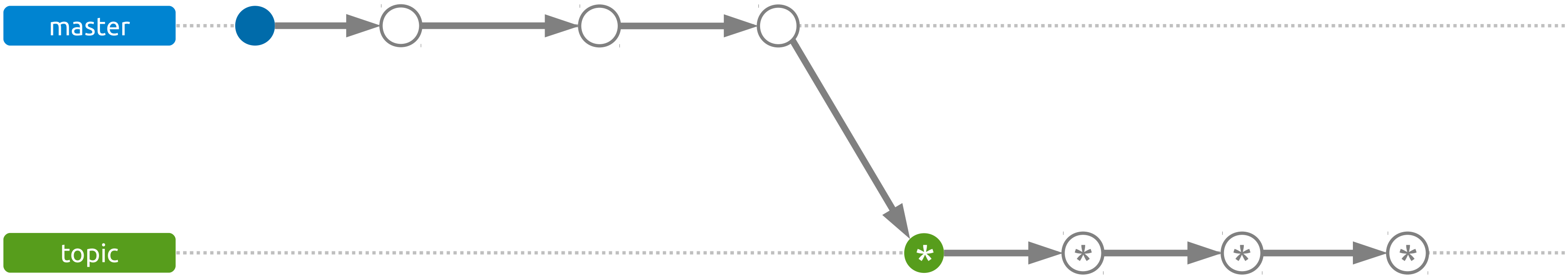


Rewriting history

Avoid changing published histories!

Rebase the current branch on to the tip of a different one

```
$ git rebase master
```



Ups, I forgot something

Ups, I forgot something

Add all your current changes to the previous commit

```
$ git commit --amend -a
```

Ups, I forgot something

Add all your current changes to the previous commit

```
$ git commit --amend -a
```

Reword the last commit

```
$ git commit --amend -m "new commit message"
```

Ups, I forgot something

Avoid changing published histories!

Add all your current changes to the previous commit

```
$ git commit --amend -a
```

Reword the last commit

```
$ git commit --amend -m "new commit message"
```

Pushing a rewritten history

Danger zone! Overwrites the remote history with your local one. *Remote* commits may get lost!

```
$ git push --force origin <branch>
```

When that merge came in like a wrecking ball

Abort an initiated merge or rebase

```
$ git merge --abort
```

```
$ git rebase --abort
```

Git tag

Tag (mark) important points in the commit history, e.g. when a working version is released

Show all tags

```
$ git tag
```

Tag the current commit with an annotation

```
$ git tag -a v2.0
```

Push tags to the remote

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

Ignoring files

Create a file in your repository (and add & commit it) named `.gitignore`, containing paths and rules that tell git which files to ignore.

```
# Maven files
target/
bin/

# Eclipse project files
.project
.classpath
.settings

# Mac OS
.DS_Store

# IntelliJ IDEA files
*.iml
*.ipr
*.iws
.idea

# backup-files
*~
```

Writing good commit messages

Writing *bad* commit messages

Writing *bad* commit messages

”fix”

Writing *bad* commit messages

”fix”

“:(:(“

Writing *bad* commit messages

”fix”

“:(:(“

“changes”

Writing *bad* commit messages

”fix”

“:(:(“

“changes”

“it works!”

Writing *bad* commit messages

”fix”

“:(:(“

“changes”

“it works!”

“final commit”

Writing *bad* commit messages

”fix”

“:(:(“

“changes”

“it works!”

“final commit”

“Testing in progress ;-)”

Writing *bad* commit messages

”fix”

“:(:(“

“changes”

“it works!”

“final commit”

“Testing in progress ;-)”

“TODO: write meaningful commit message”

Writing *bad* commit messages

”fix”

“:(:(“

“changes”

“it works!”

“final commit”

“Testing in progress ;-)”

“TODO: write meaningful commit message”

“Your commit is writing checks your merge can't cash”

Writing good commit messages

Write commit messages as if you're giving *commands* to the codebase

Writing good commit messages

Write commit messages as if you're giving *commands* to the codebase

“Add DAO interfaces for Entities“

Writing good commit messages

Write commit messages as if you're giving *commands* to the codebase

“Add DAO interfaces for Entities“

“Implement basic version of AddressDAO”

Writing good commit messages

Write commit messages as if you're giving *commands* to the codebase

“Add DAO interfaces for Entities“

“Implement basic version of AddressDAO”

“Fix bug in delete method of UserDAO”

“Move package security to at.ac.tuwien.service”

“Add generated fxml files for UI”

basics

stages

branch

merge

remotes

advanced

Other great tutorials

- Official Git Documentation
  http://git-scm.com/doc
- TryGit – An interactive Git tutorial
  http://try.github.io
- Git Immersion
  http://gitimmersion.com/
- Atlassian Git Tutorials
  https://www.atlassian.com/git
- Git Cheatsheet – Command categorisation
  http://ndpsoftware.com/git-cheatsheet.html
- LearnGitBranching
  http://pcottle.github.io/learnGitBranching

Further reading

- A successful Git branching model
  http://nvie.com/posts/a-successful-git-branching-model/
- Changing history, or How to Git pretty
  http://justinhileman.info/article/changing-history
- Reset Demystified
  http://git-scm.com/blog/2011/07/11/reset.html
- Avoiding Git Disasters: A Gory Story
  http://randyfay.com/node/89
- A Rebase Workflow for Git
  http://randyfay.com/node/91

Headache?

Questions?

fin

Thanks for listening - Enjoy git!

Feedback appreciated

thomas.rausch@tuwien.ac.at