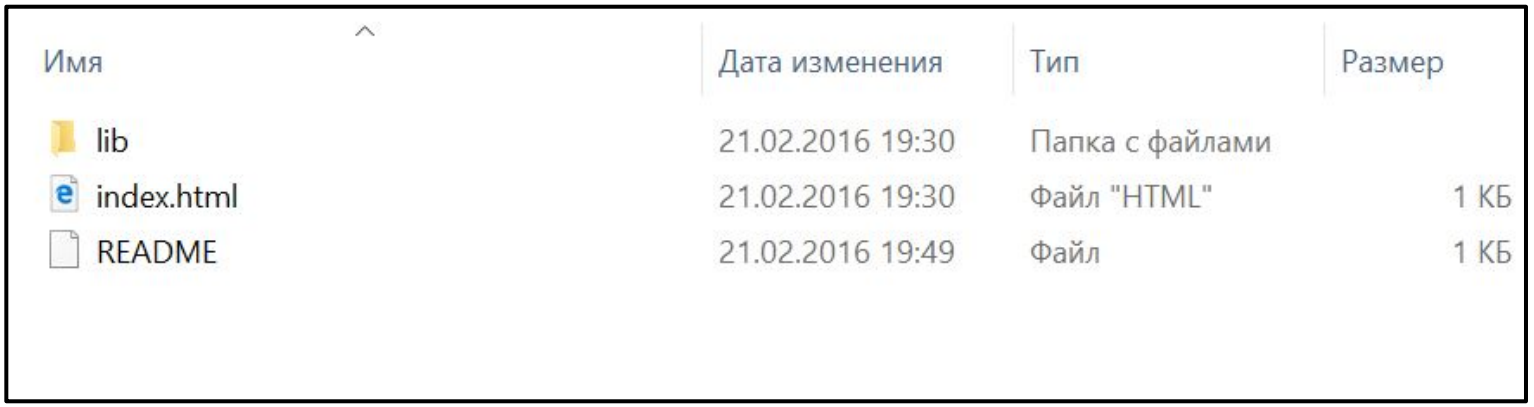
Git

Иван Домашних

Базовые понятия

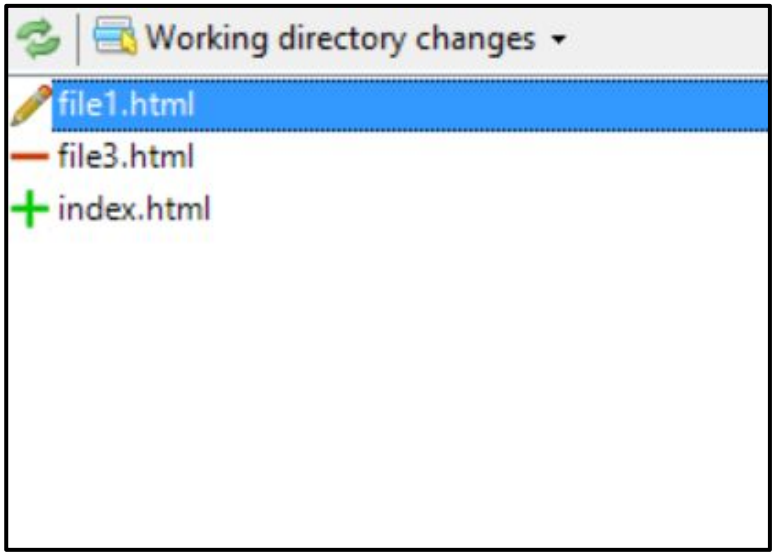
Working directory



Modification

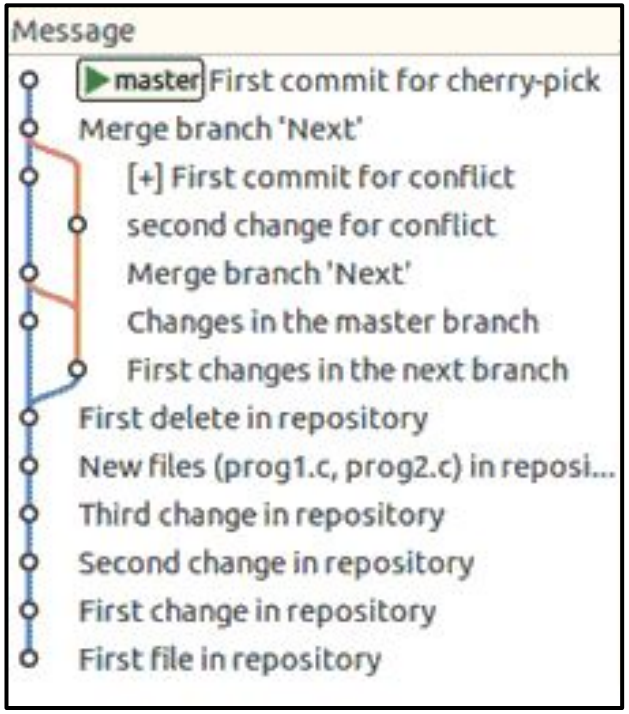
```
diff --git a/file1.html b/file1.html
index 05c9afd..36f0a56 100644
--- a/file1.html
+++ b/file1.html
@@ -1 +1,2 @@
-<h1>File1</h1>
\ No newline at end of file
+<h1>File1</h1>
+Content
\ No newline at end of file
```

Commit

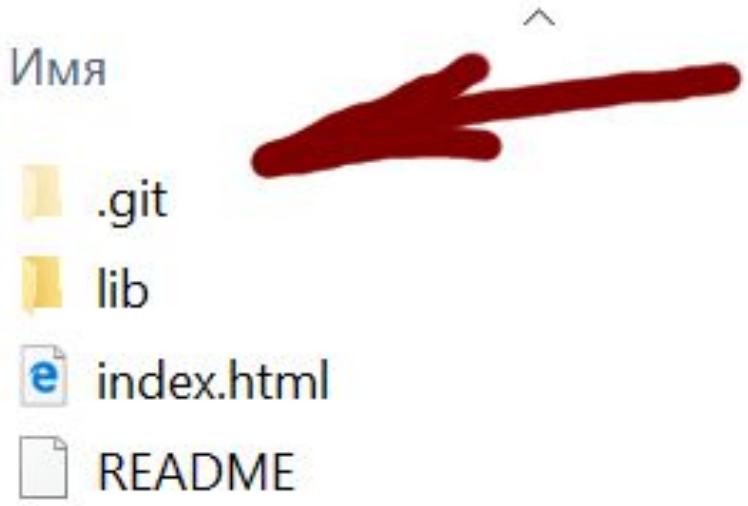


Commit hash
Commit message
Author
Date

History


Message
master First commit for cherry-pick
Merge branch 'Next'
[+] First commit for conflict
second change for conflict
Merge branch 'Next'
Changes in the master branch
First changes in the next branch
First delete in repository
New files (prog1.c, prog2.c) in reposi...
Third change in repository
Second change in repository
First change in repository
First file in repository

Repository



repository + working directory

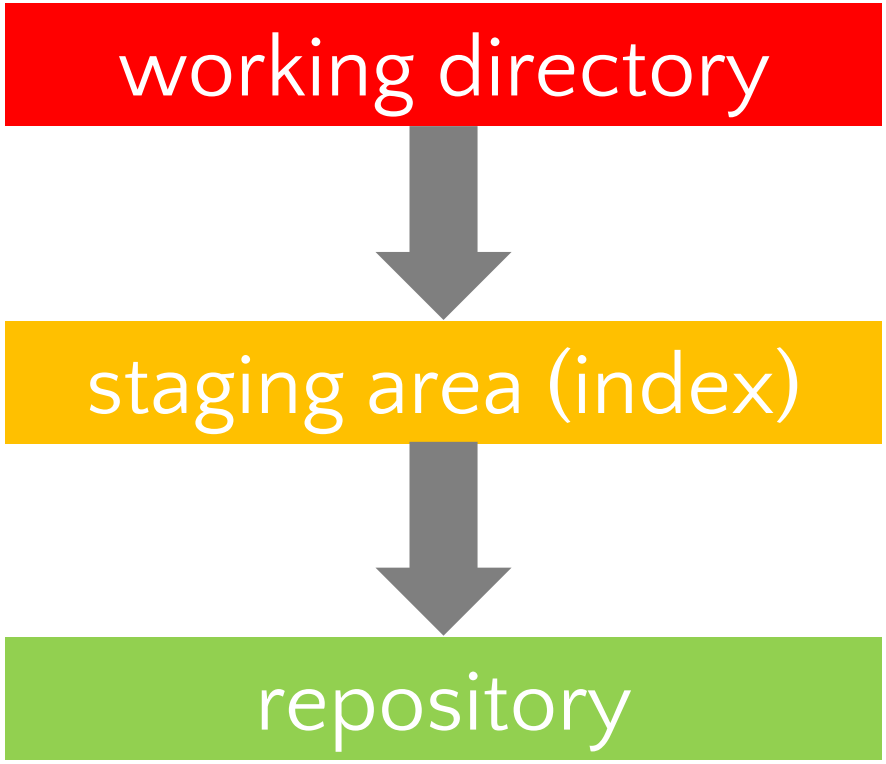
test/.git – repository + working directory
test.git – bare repository

Внесение изменений

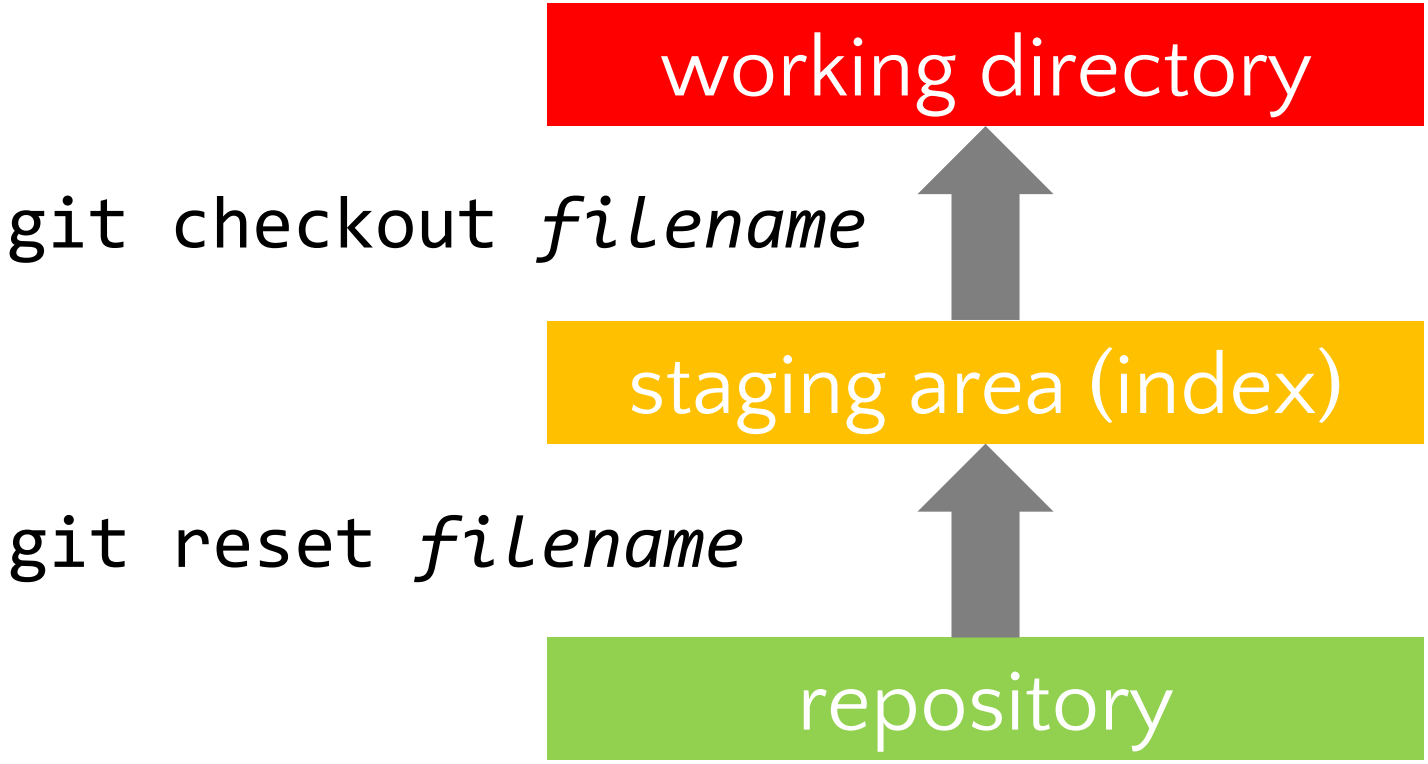
add to staging area and commit

```
git add .
```

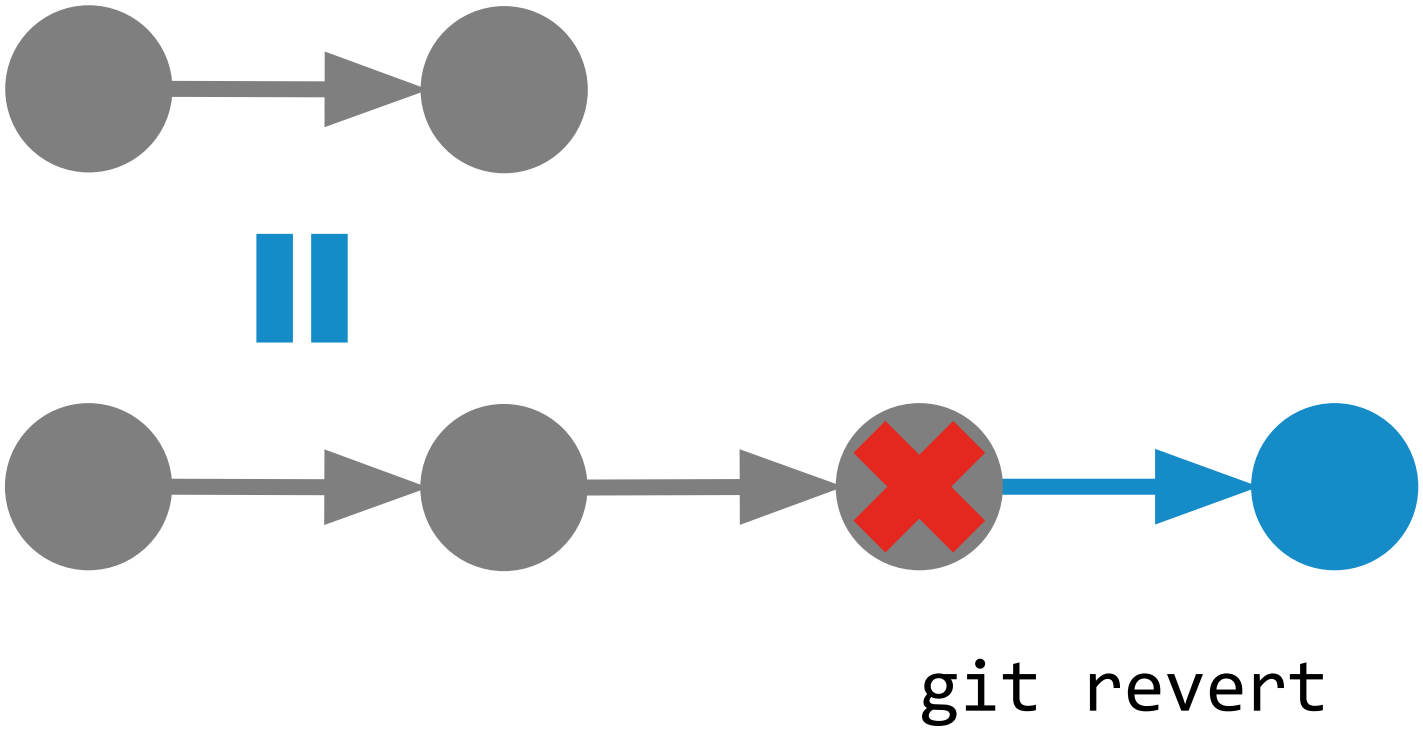
```
git commit
```



checkout file or reset file



revert commit



move & remove

`git mv` – переименовывает/перемещает файл*

`git rm` – удаляет файл

* Git прекрасно умеет отслеживать изменения сам,
поэтому можно перемещаться файлы любимым способом

get status of working directory

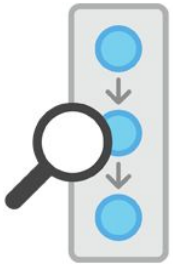
```
# Edit hello.py
git status
# hello.py is listed under "Changes not staged for commit"
git add hello.py
git status
# hello.py is listed under "Changes to be committed"
git commit
git status
# nothing to commit (working directory clean)
```

log history

```
git log -n <limit>
git log --oneline
git log --author="<pattern>“

git log --graph --decorate --oneline
git log --graph --decorate --oneline --all

git log --pretty=format:\"%h %ad | %s%d [%an]\"
--graph --date=short
```



diff

`git diff` – разница между working directory и index

`git difftool` – разница с помощью заданного тула

`git difftool --gui` – разница в GUI-туле

help

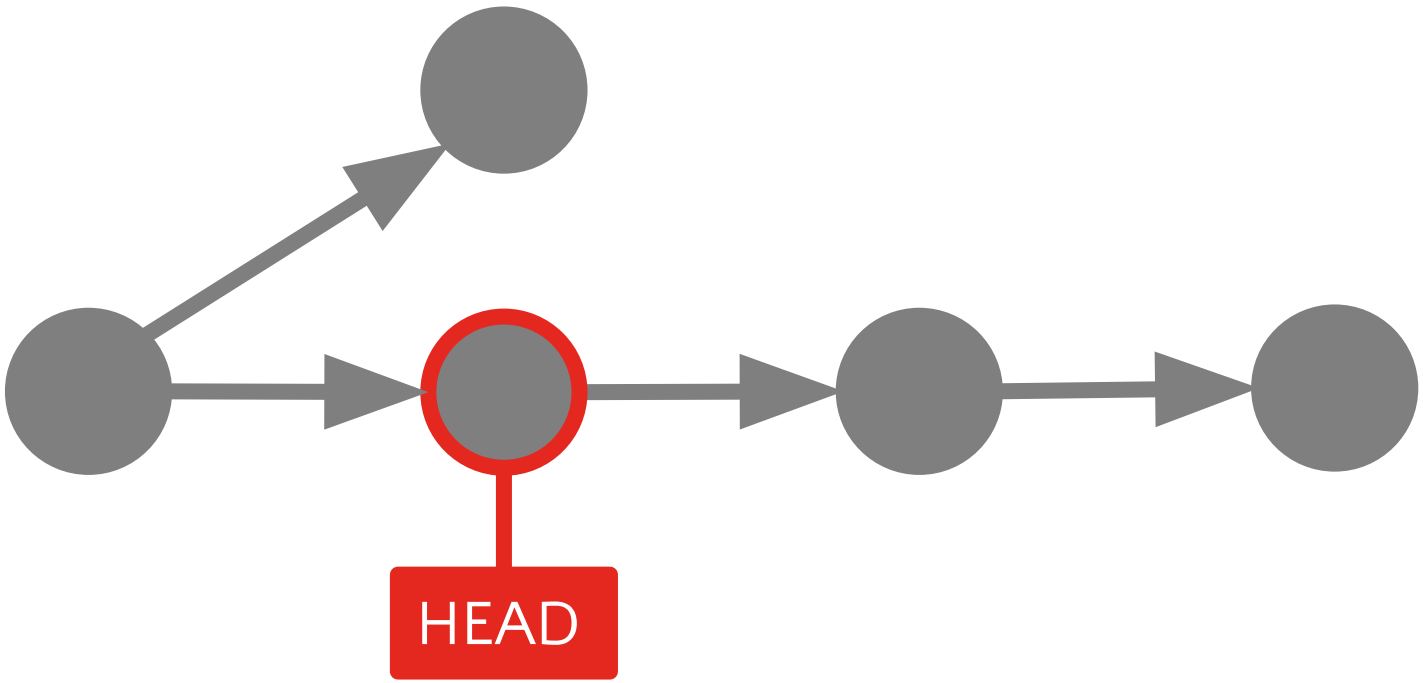
git help commit

git commit --help

git commit -h

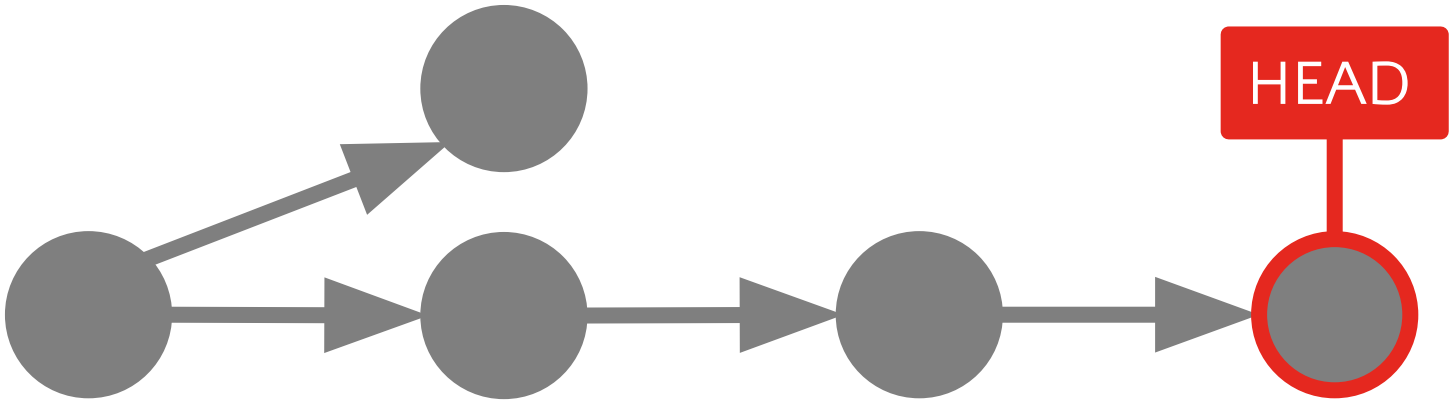
Метки и ветки

HEAD



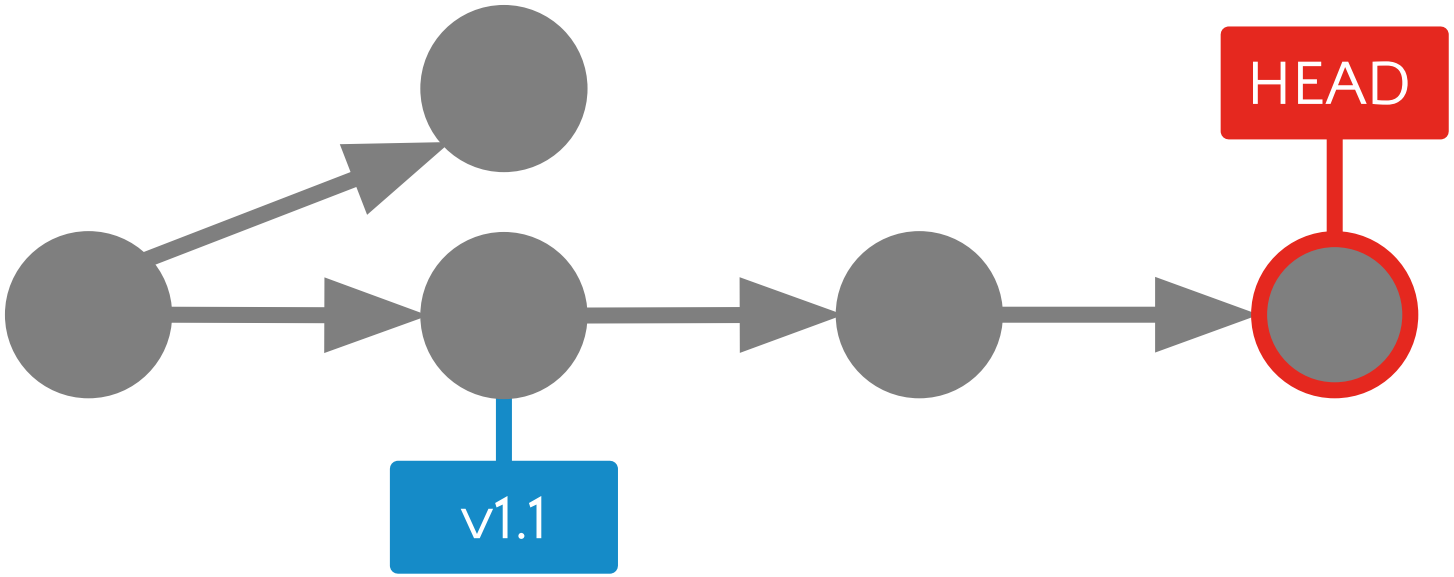
checkout to revision

```
git checkout <revision>
```

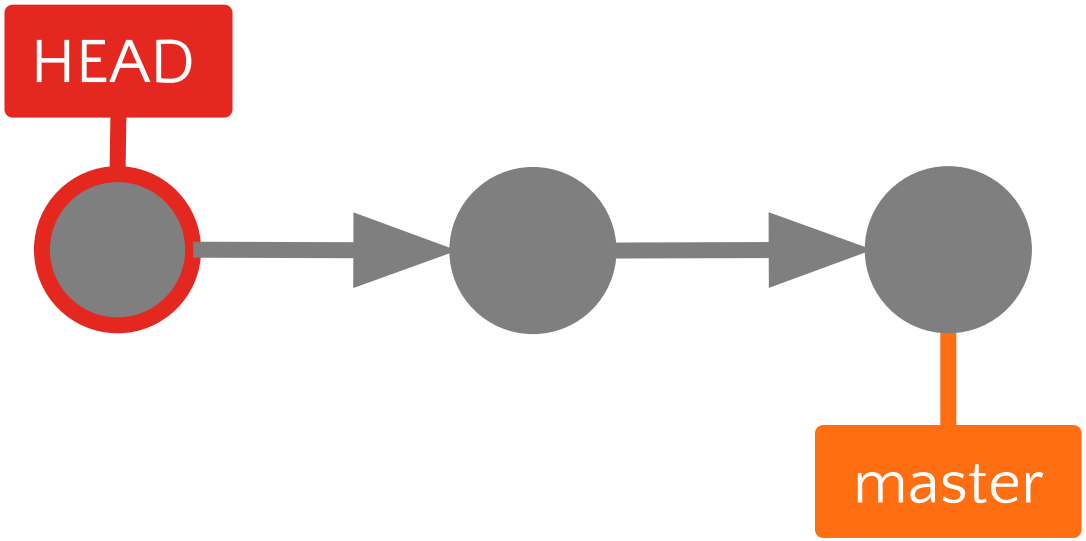


tag

```
git tag “v1.1”
```

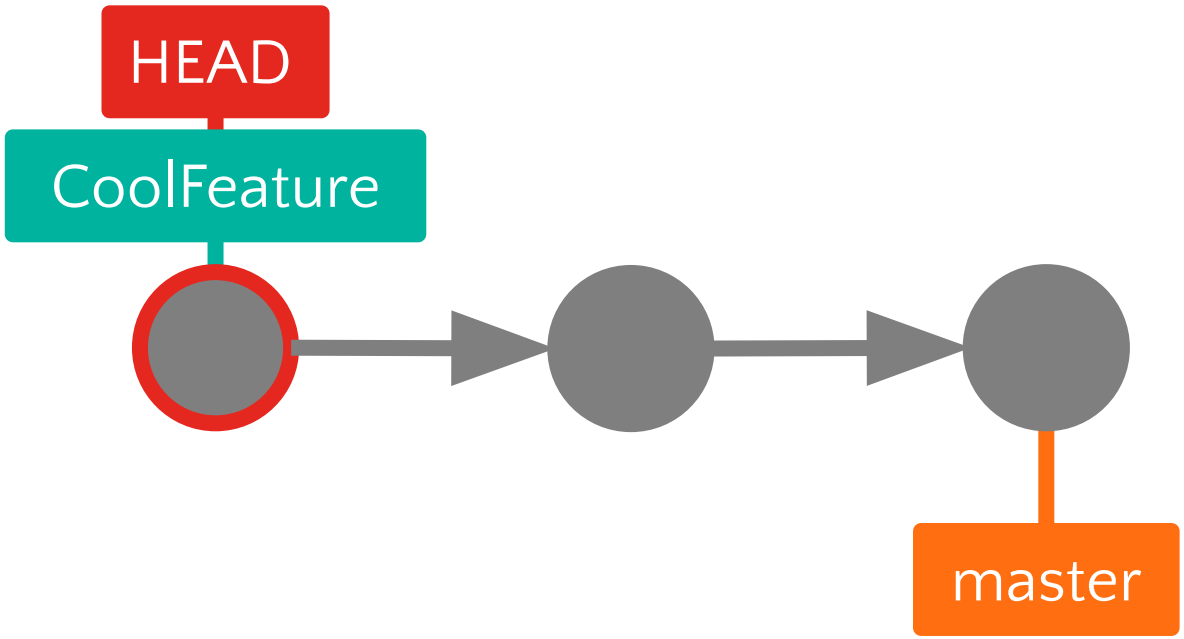


branch

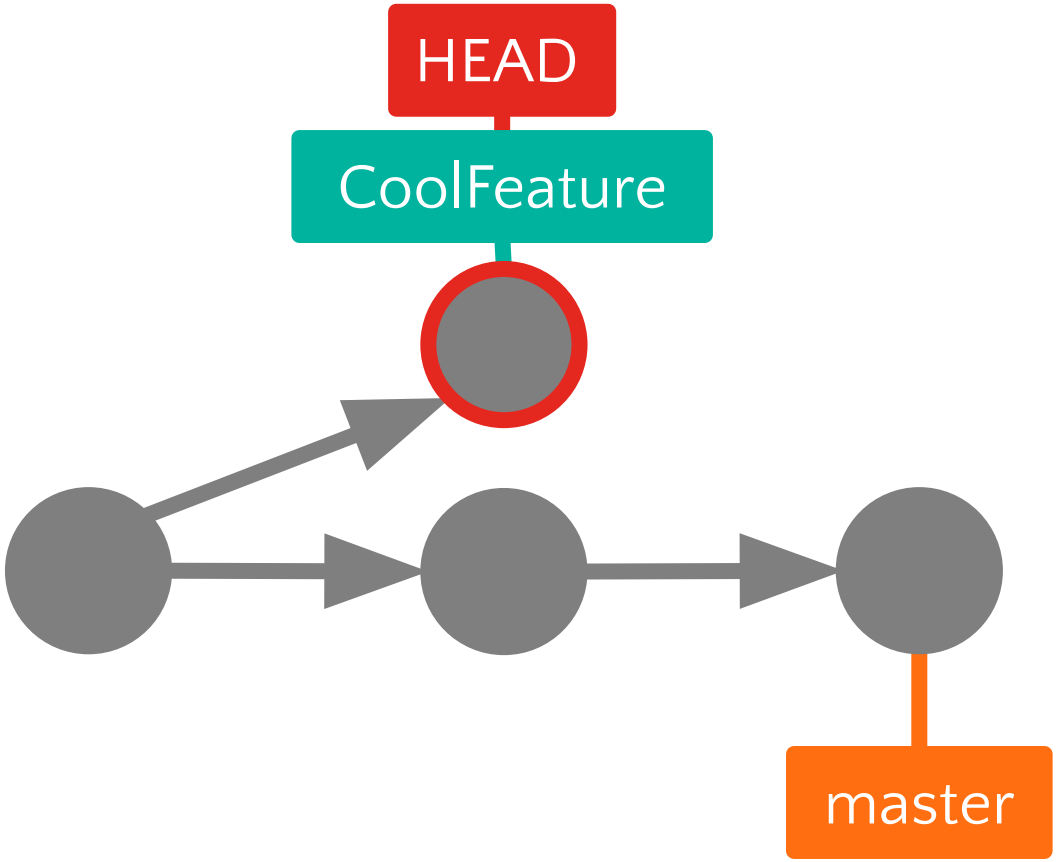


`master` – ветка по умолчанию

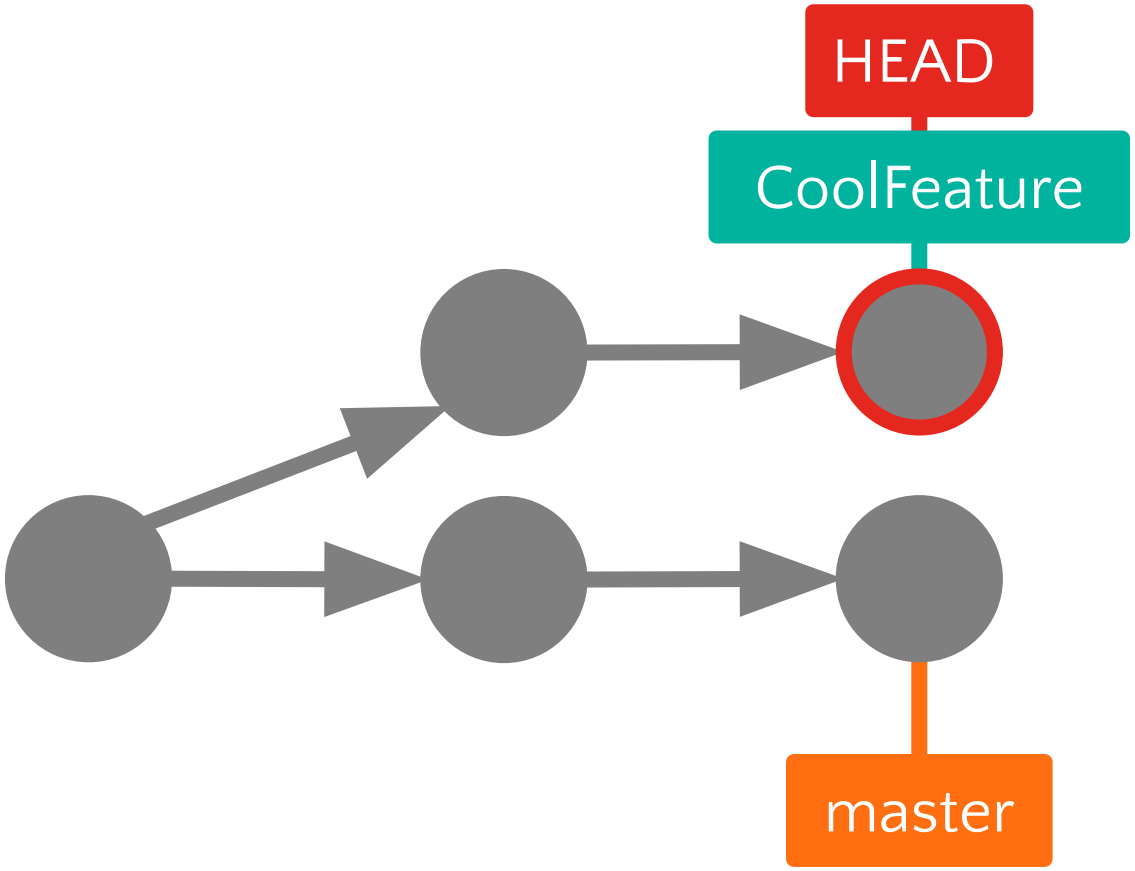
Развитие фичи



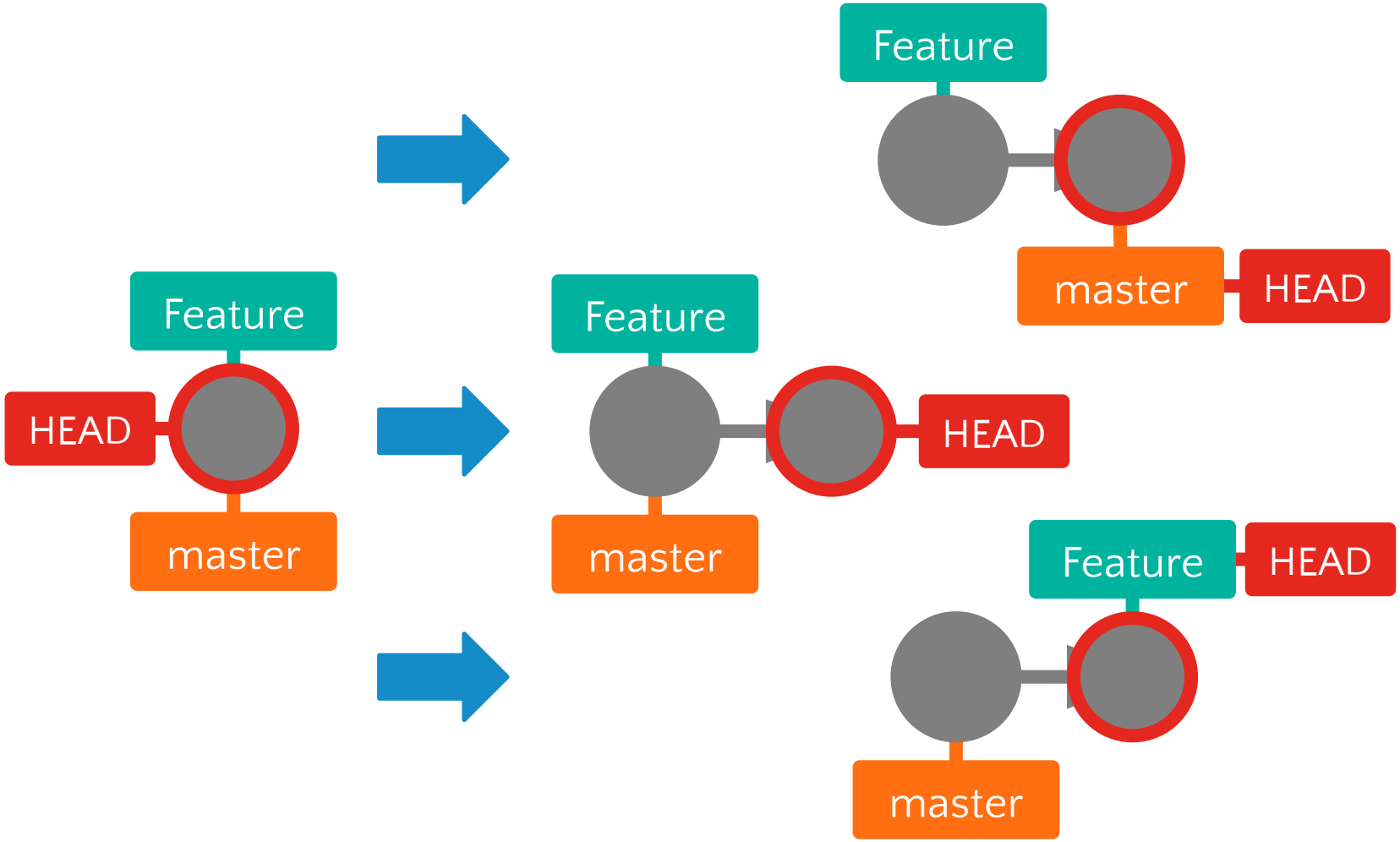
Развитие фичи



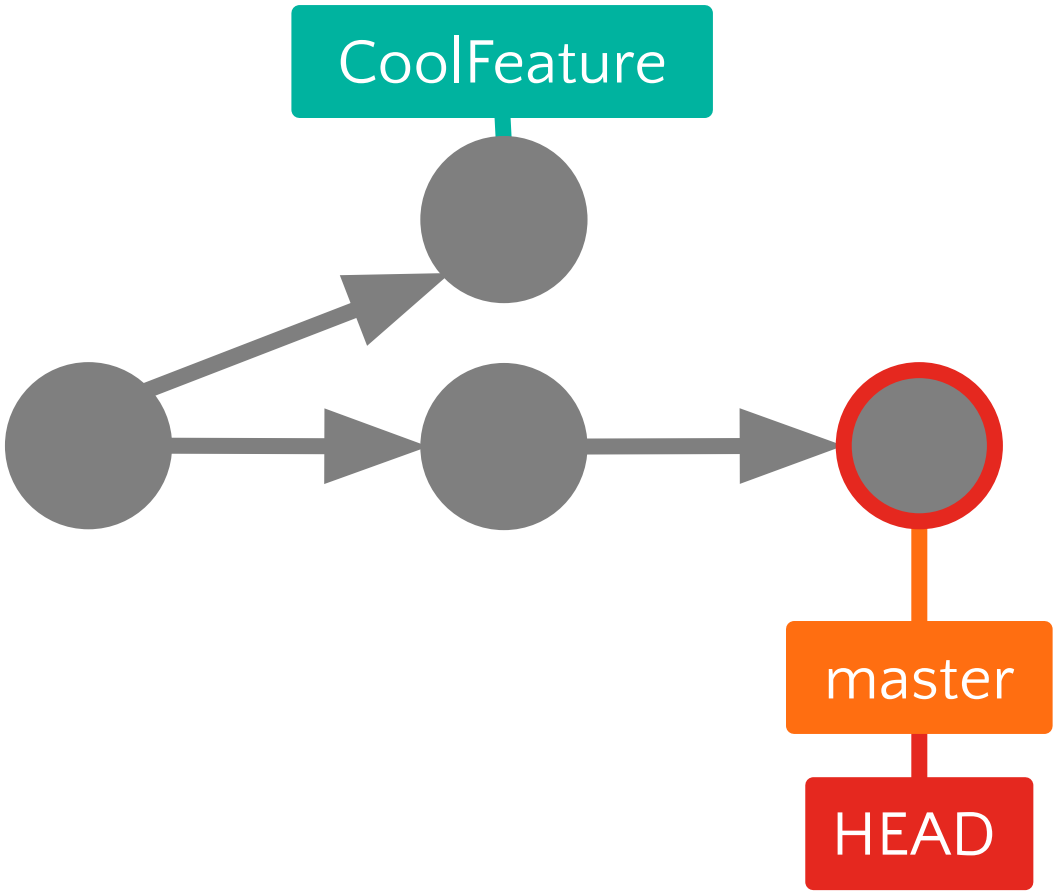
Развитие фичи



Что будет при коммите?

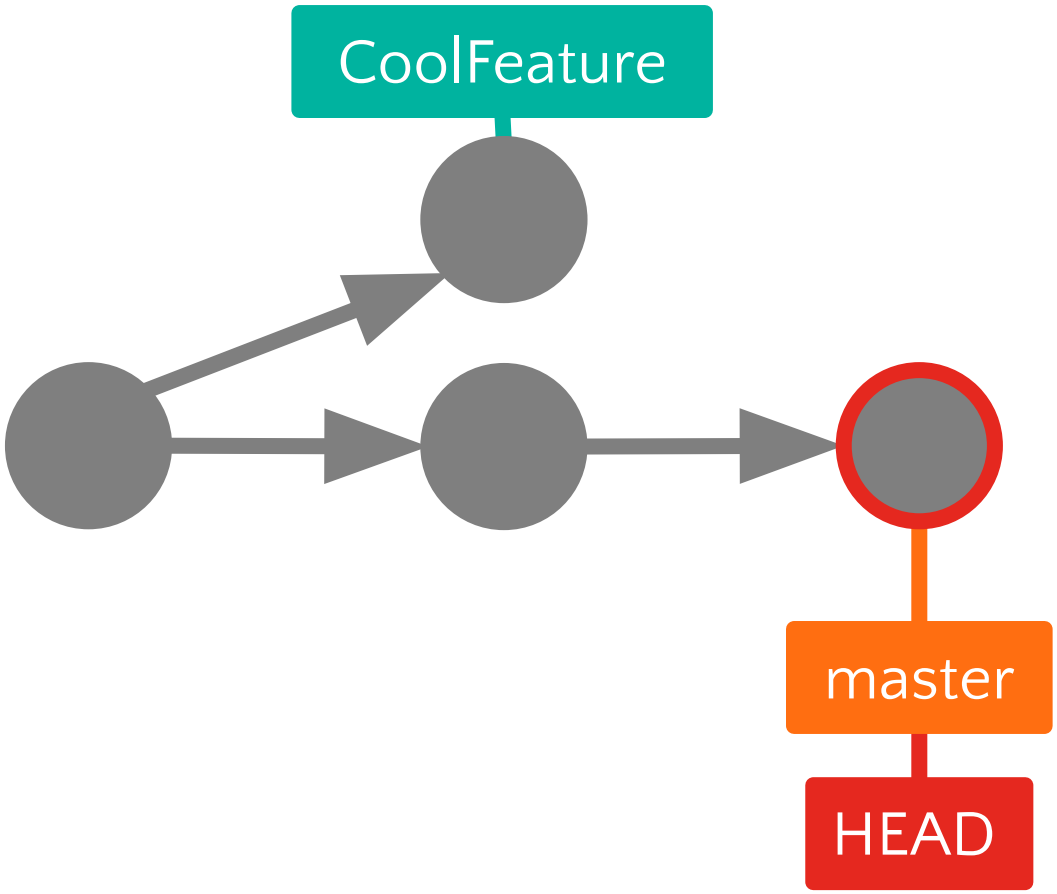


checkout branch



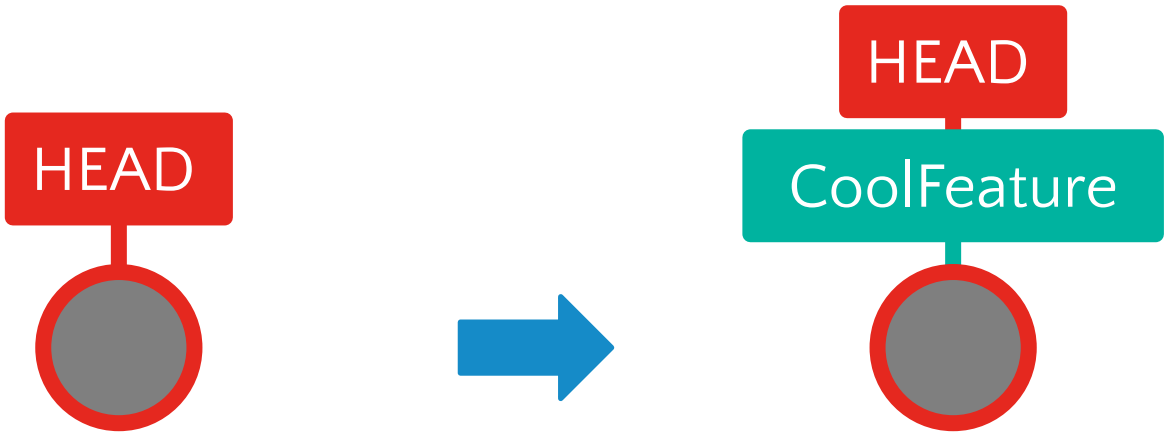
checkout branch

```
git checkout CoolFeature
```

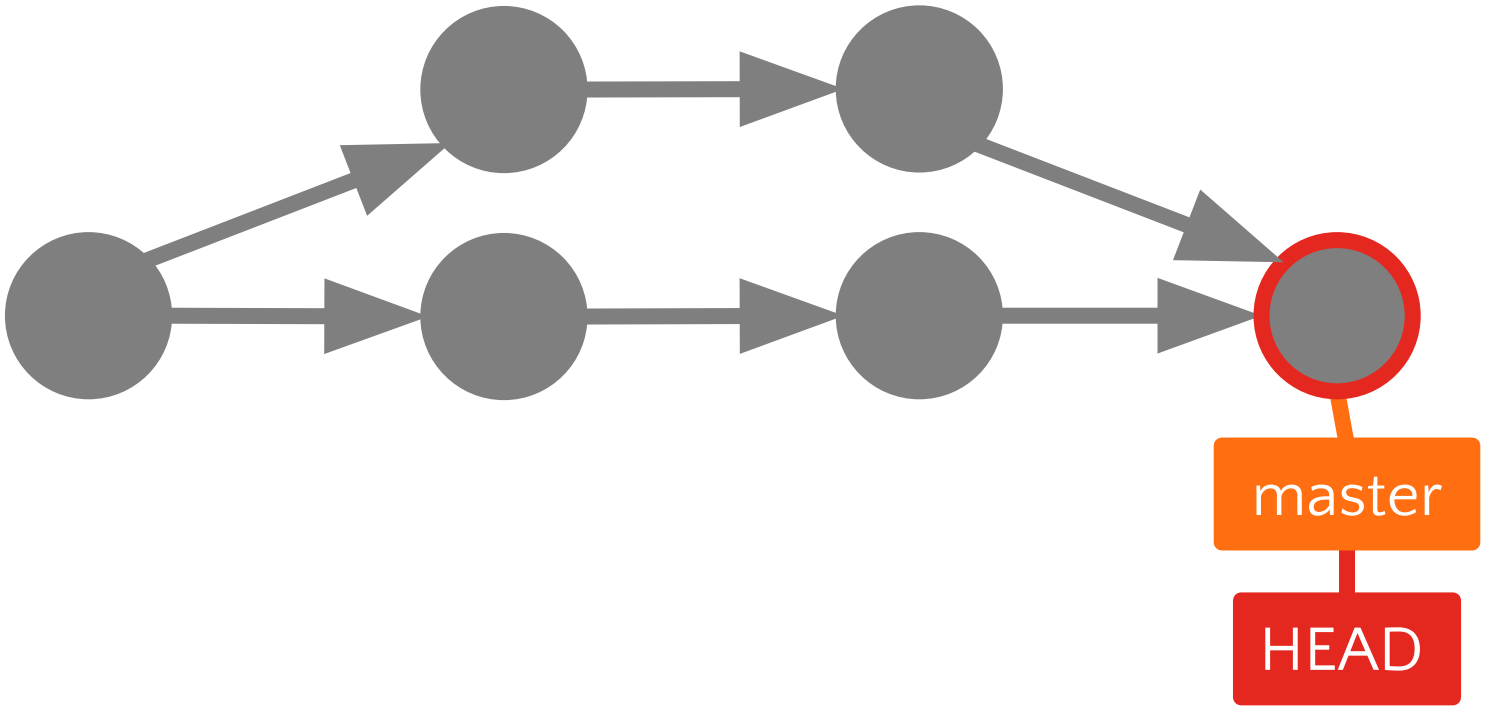


create and checkout branch

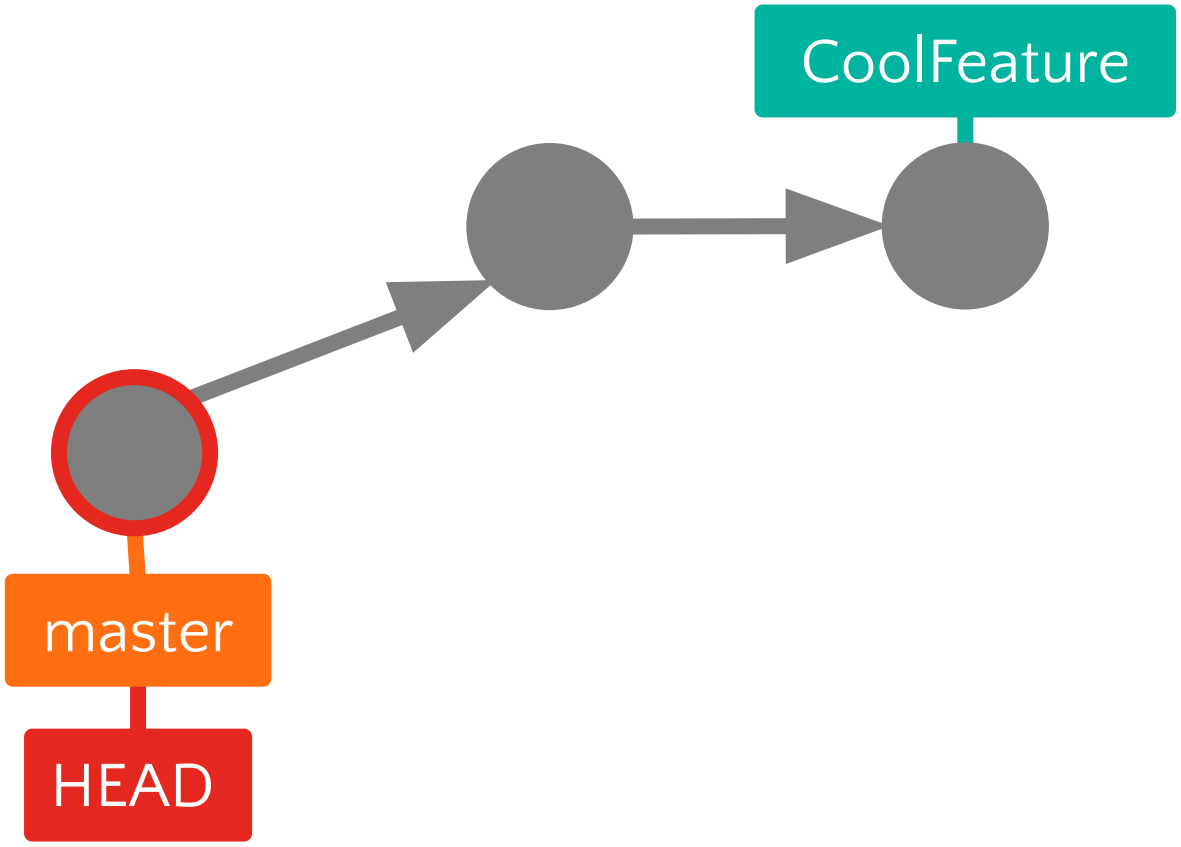
```
git checkout -b CoolFeature
```



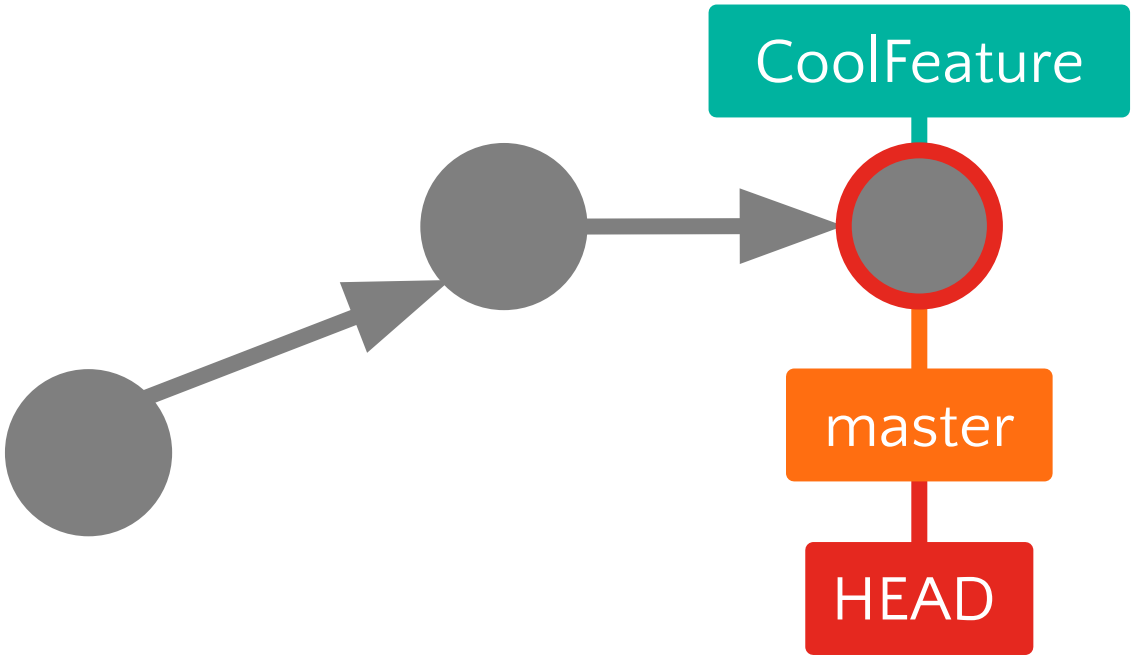
merge



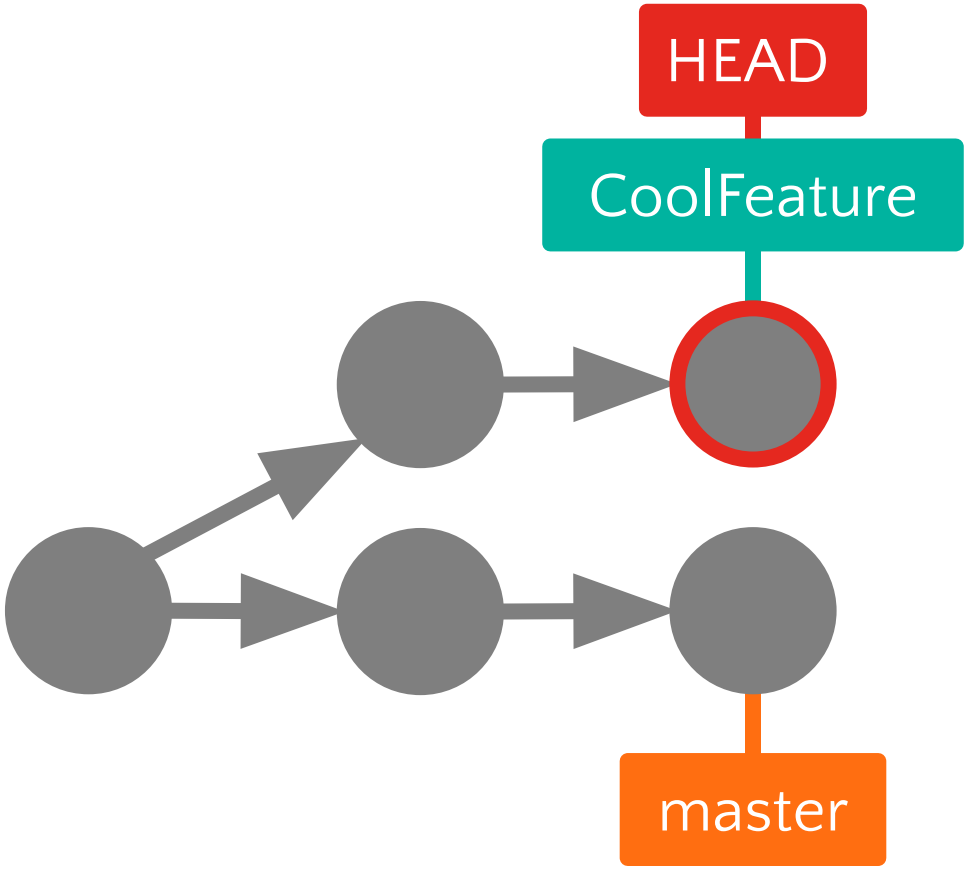
Fast Forward Merge



Fast Forward Merge

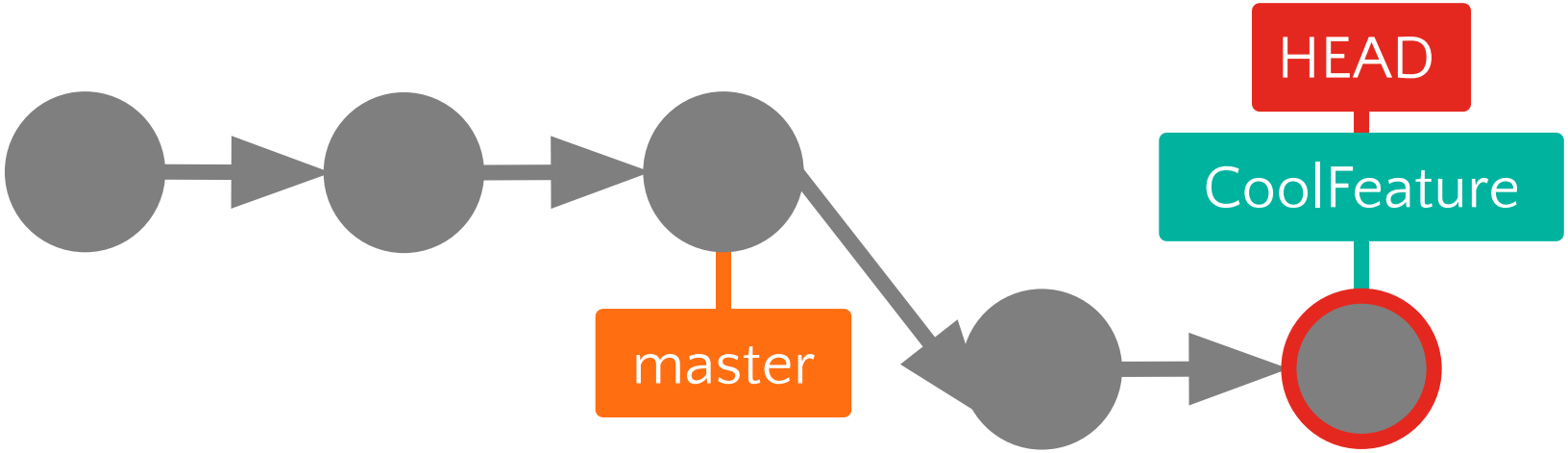


rebase

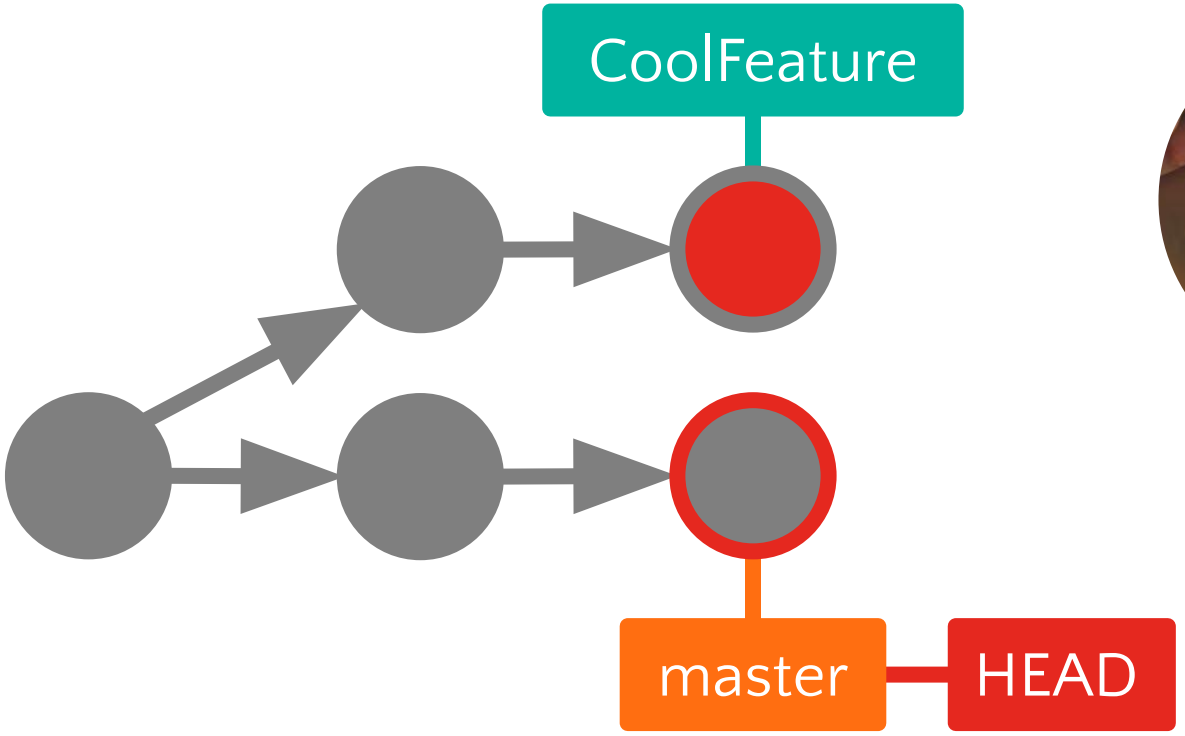


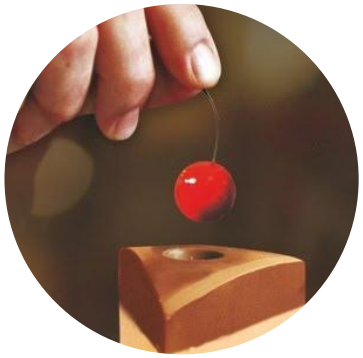
rebase

Achtung! Команда меняет историю



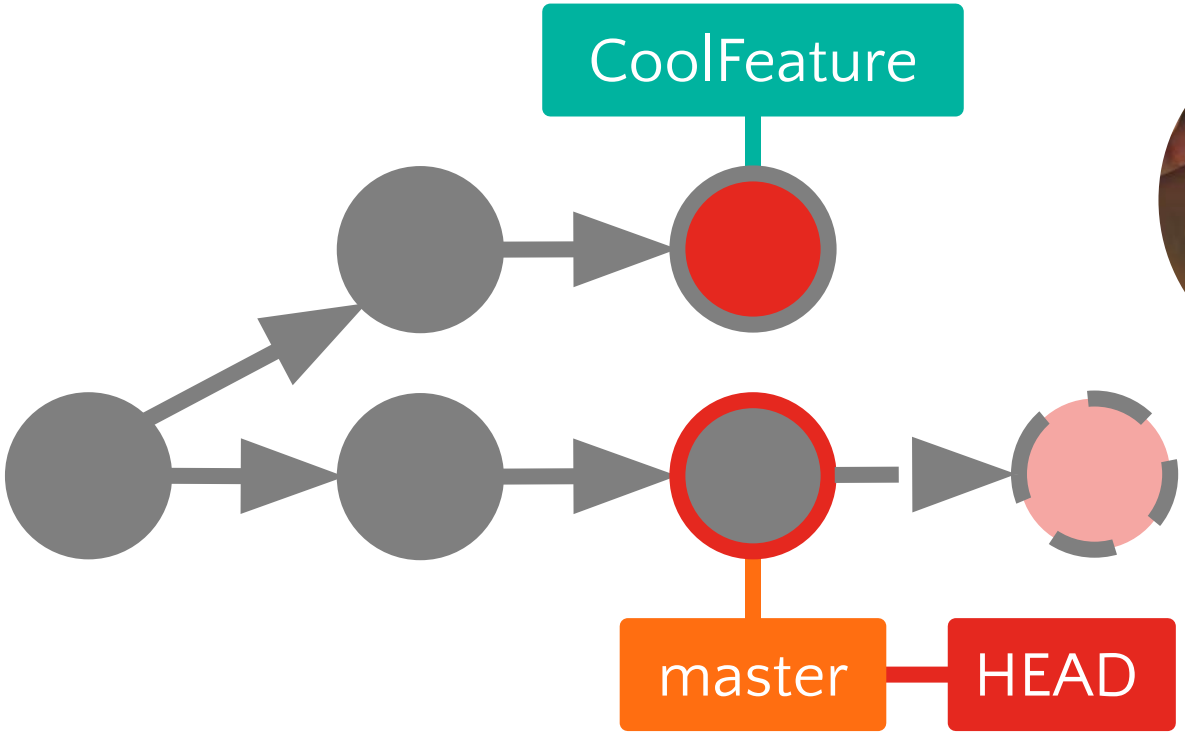
cherry-pick

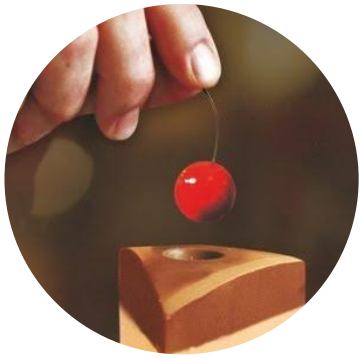




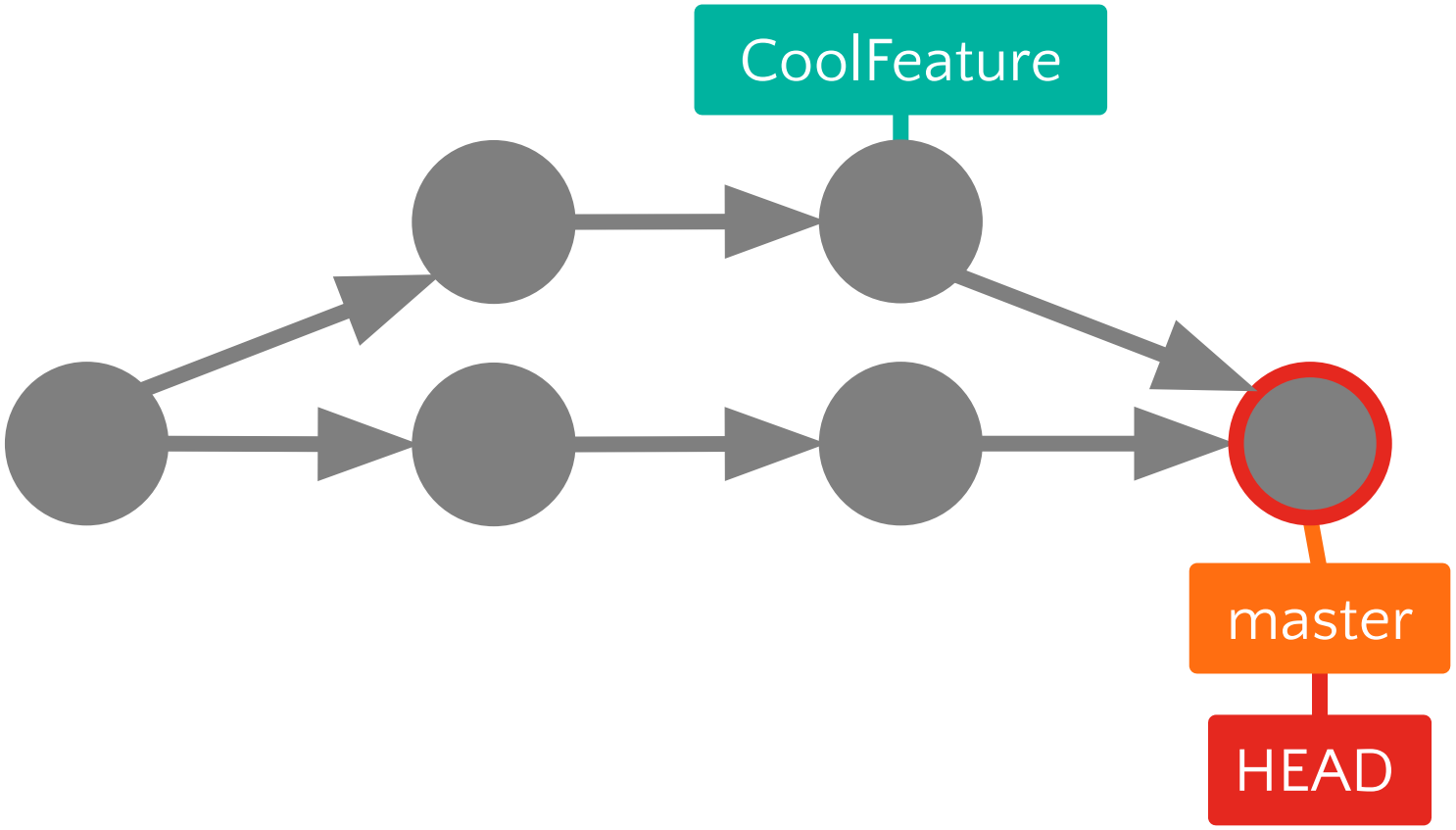
cherry-pick

```
git cherry-pick <commit-hash>
```



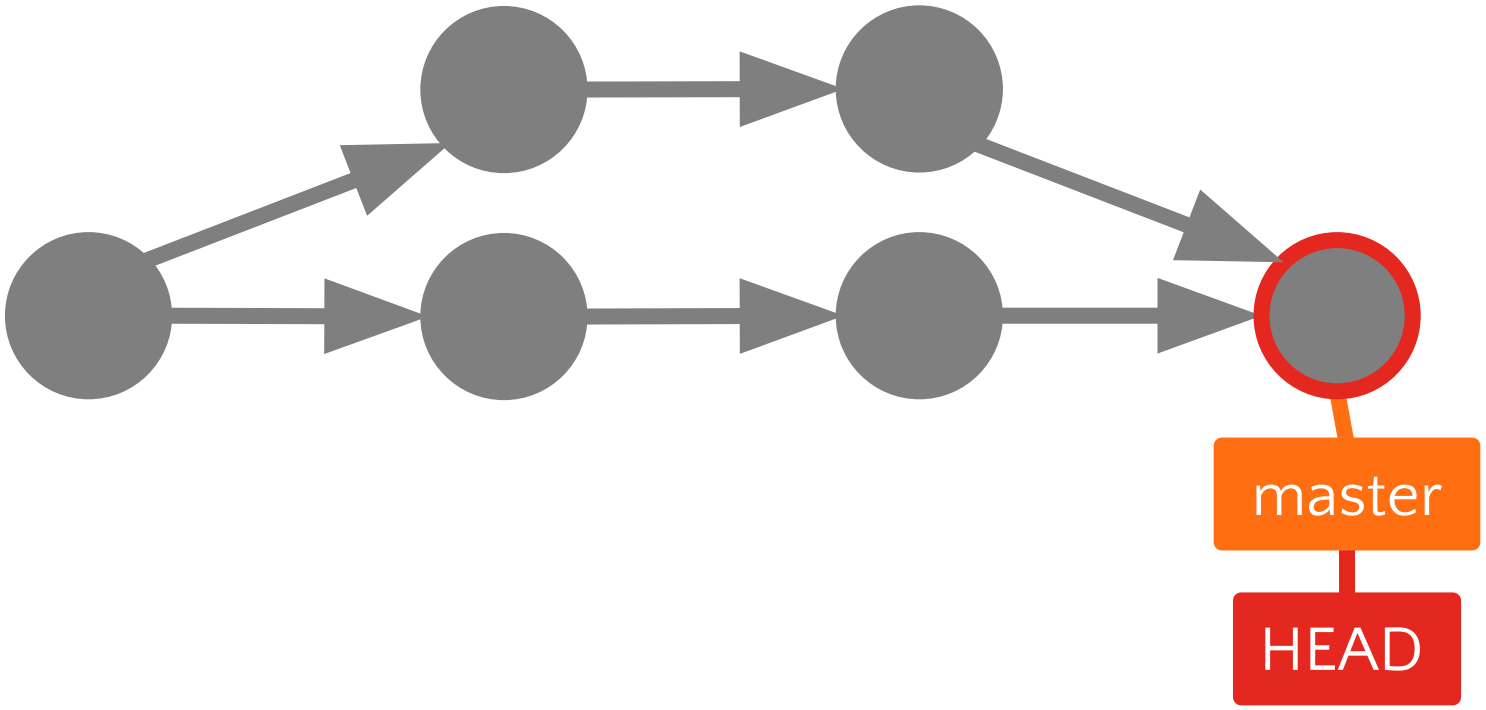


branch delete



branch delete

```
git branch –d “CoolFeature”
```



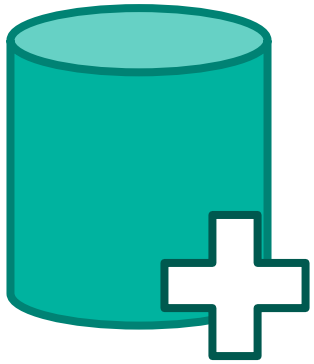
Конфликт

```
<<<<<<< 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
```

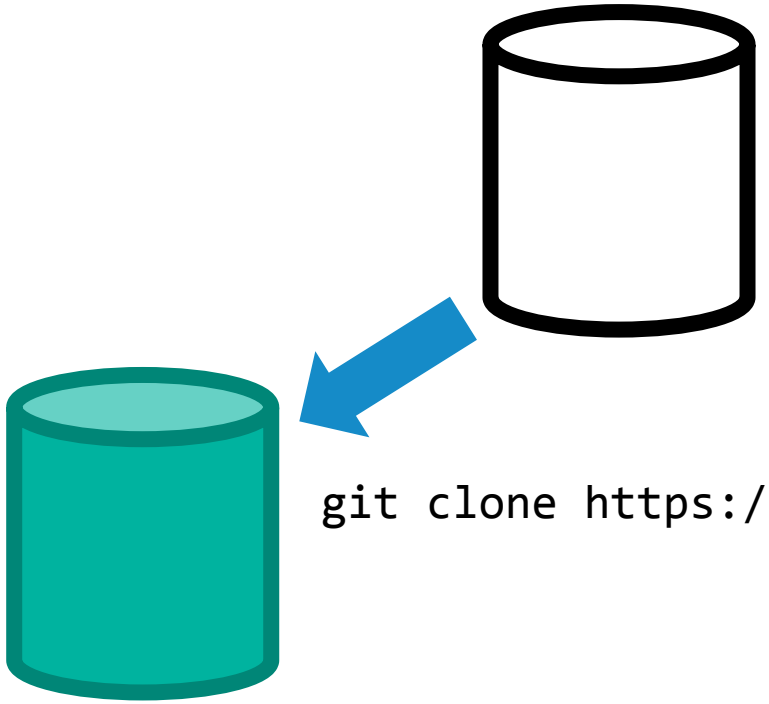
Reflog, gc

Операции над репозиториями

init



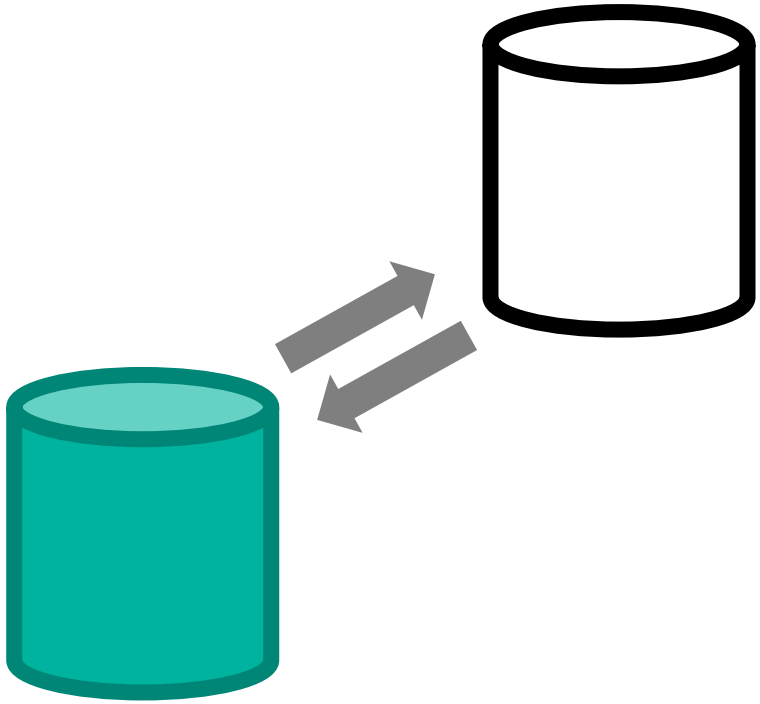
clone



```
git clone https://repos/repo.git
```

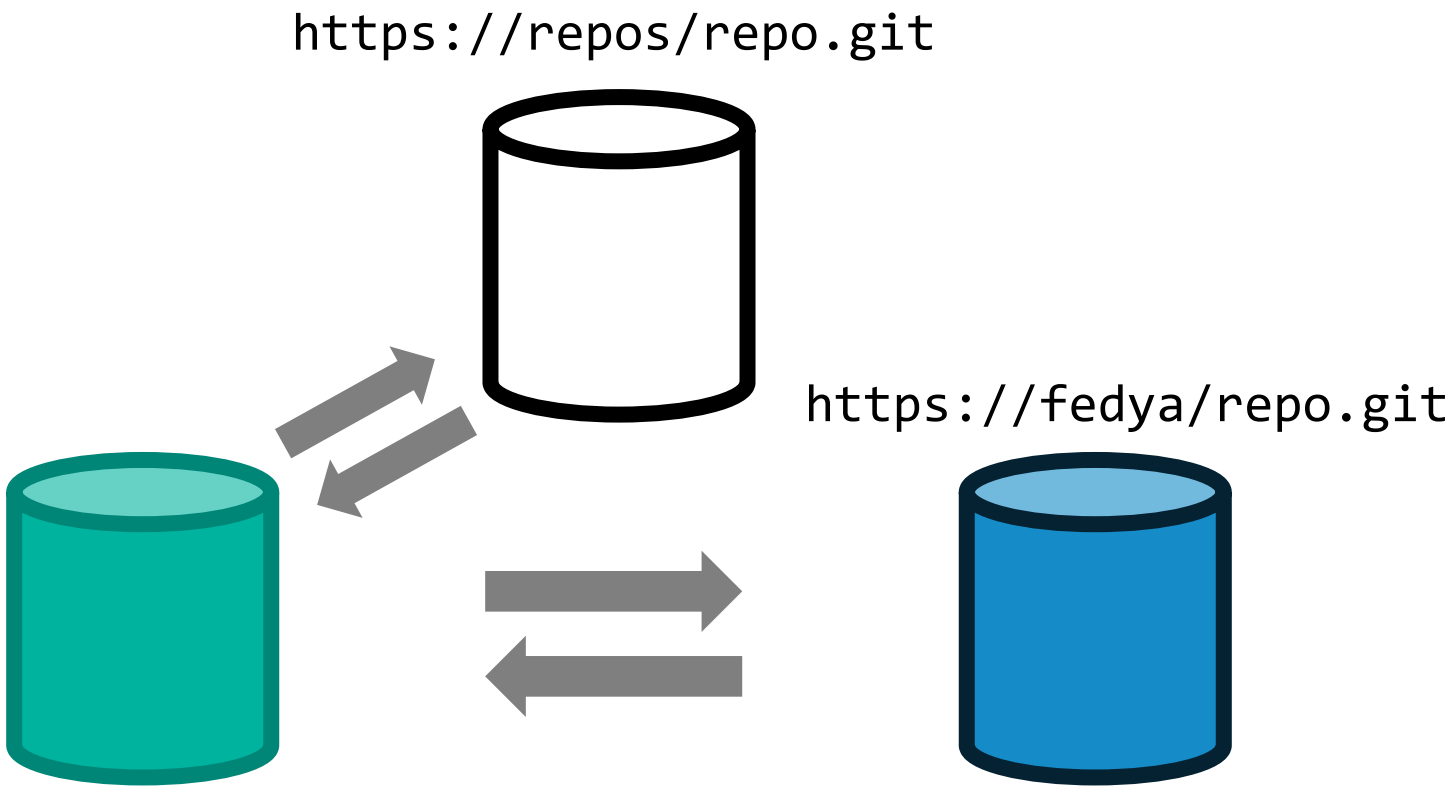
Remotes

https://repos/repo.git



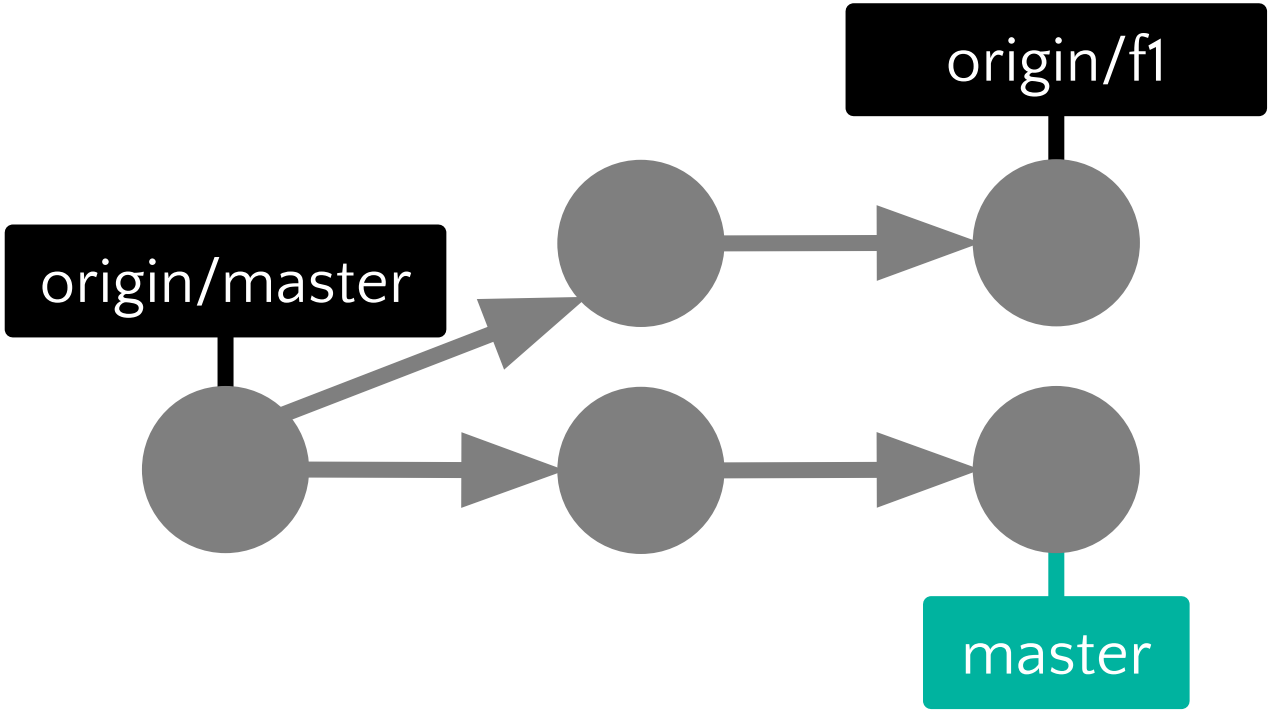
origin = https://repos/repo.git

Remotes

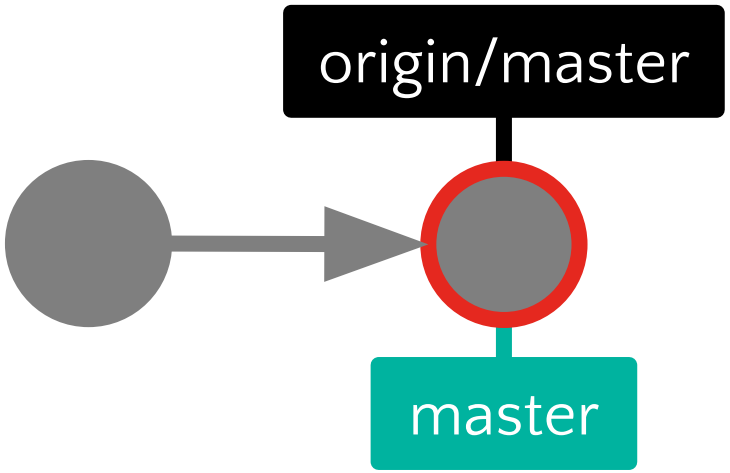



```
origin = https://repos/repo.git
fedya = https://fedya/repo.git
```

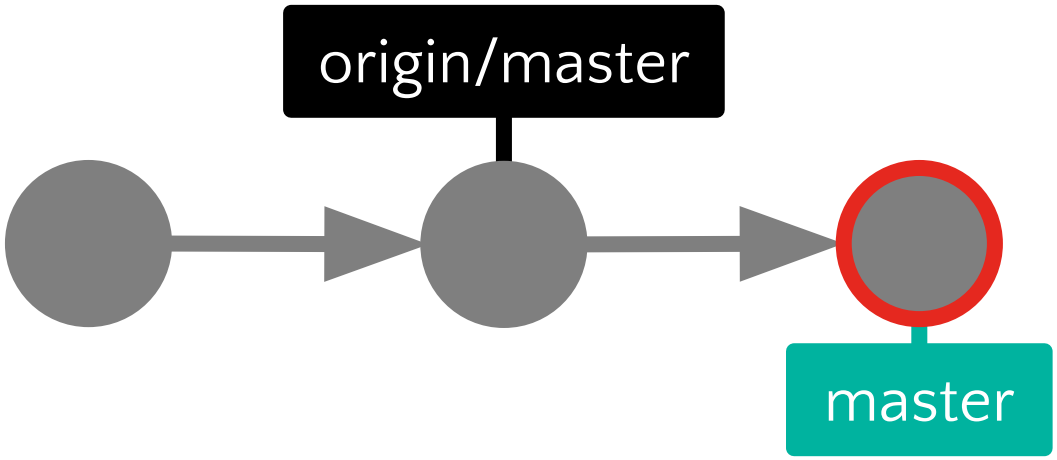
Remote branches



fetch

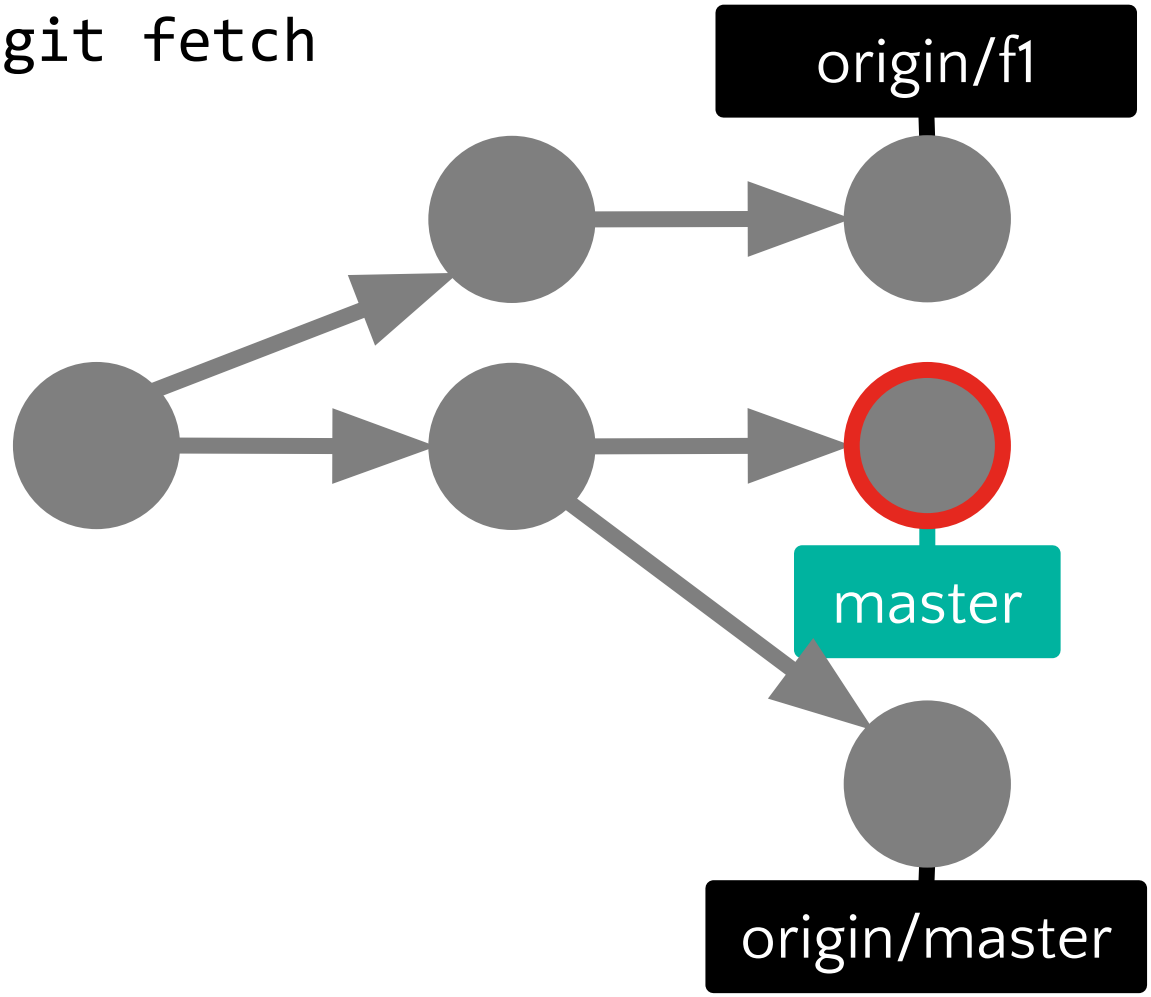


fetch

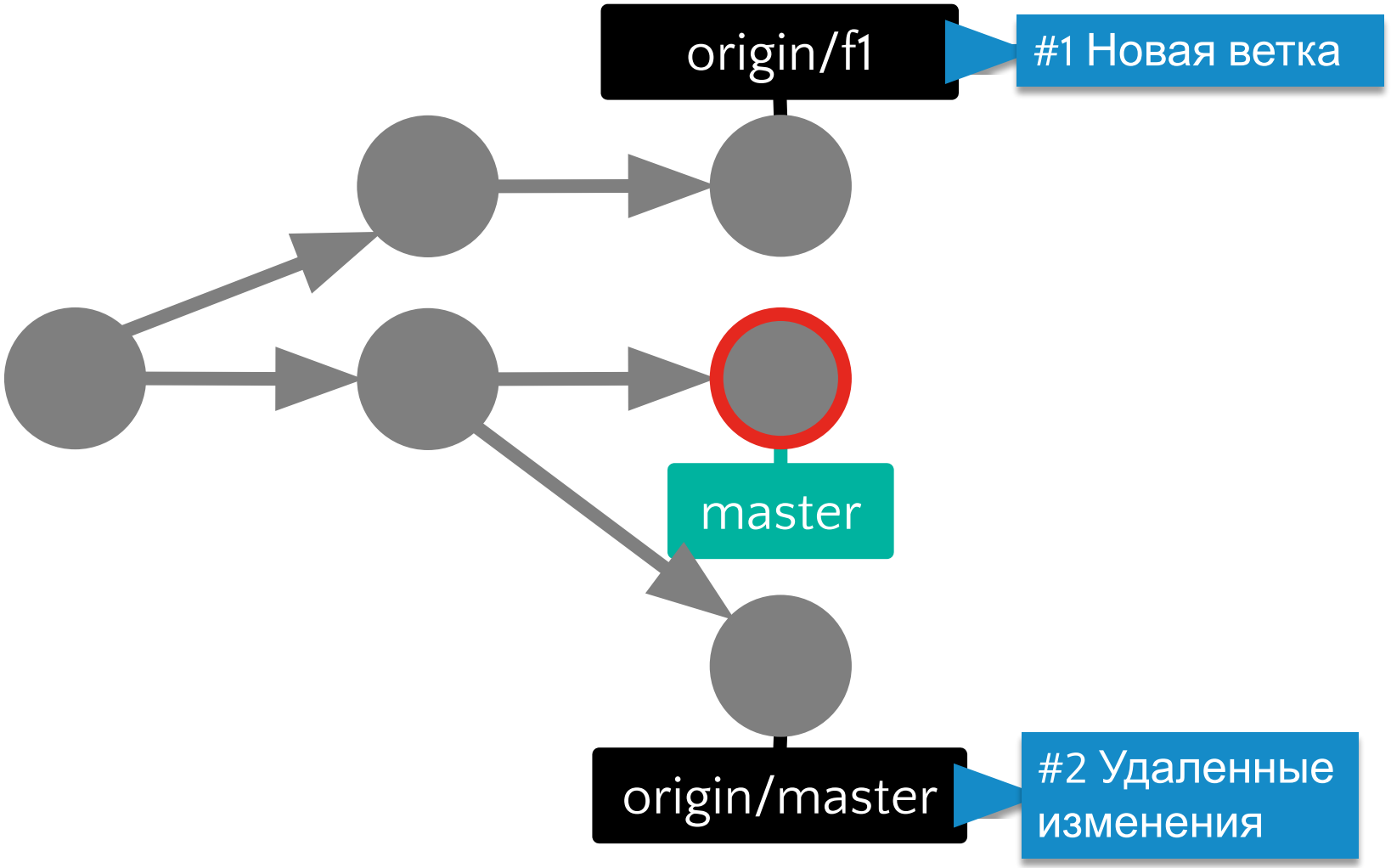


fetch

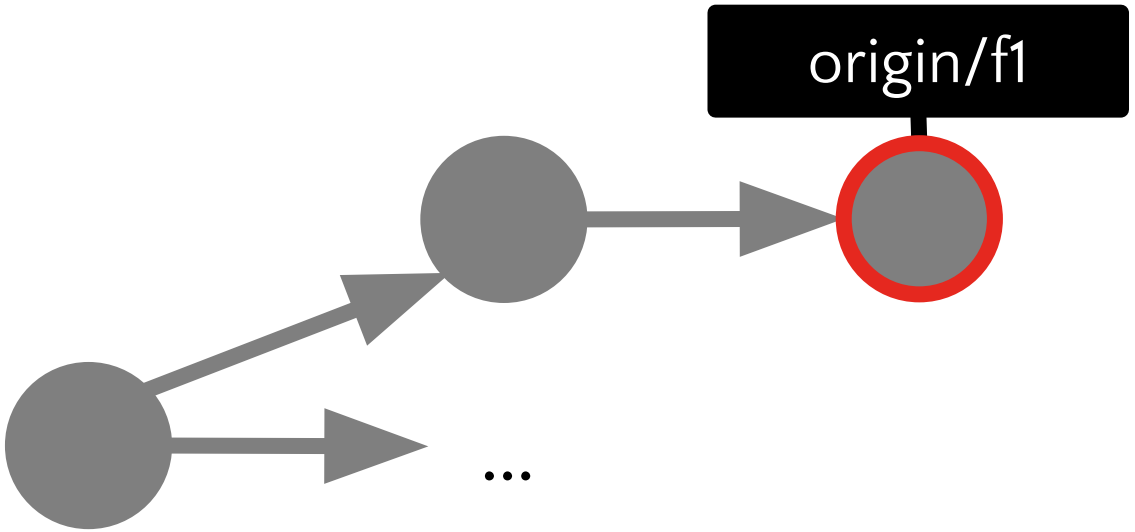
```
git fetch
```



Что делать?

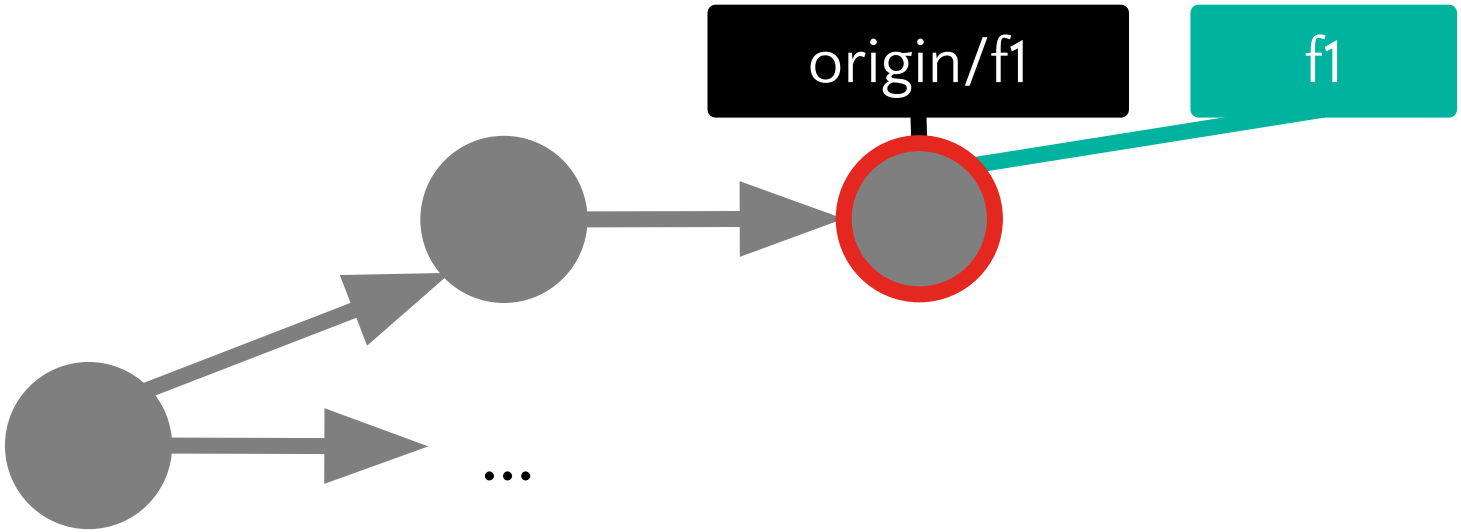

origin/f1
#1 Новая ветка
master
origin/master
#2 Удаленные изменения

#1 Новая ветка



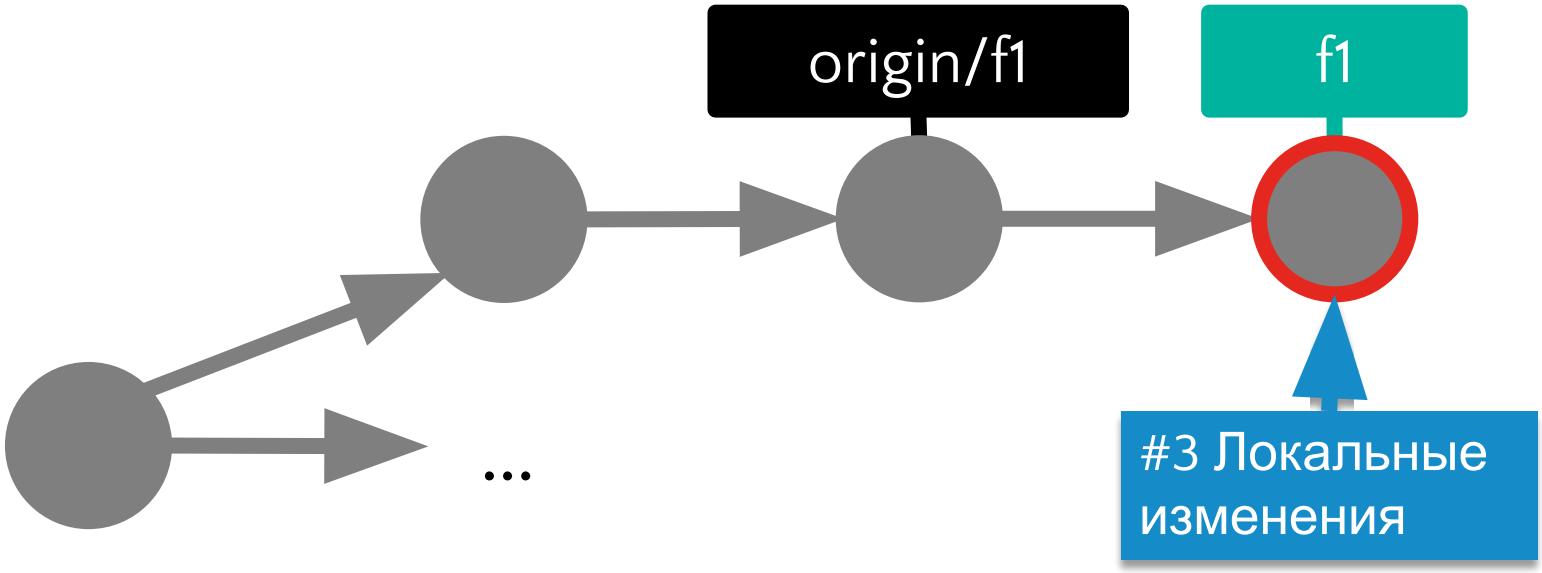
`git checkout -b f1` ?

#1 Новая ветка

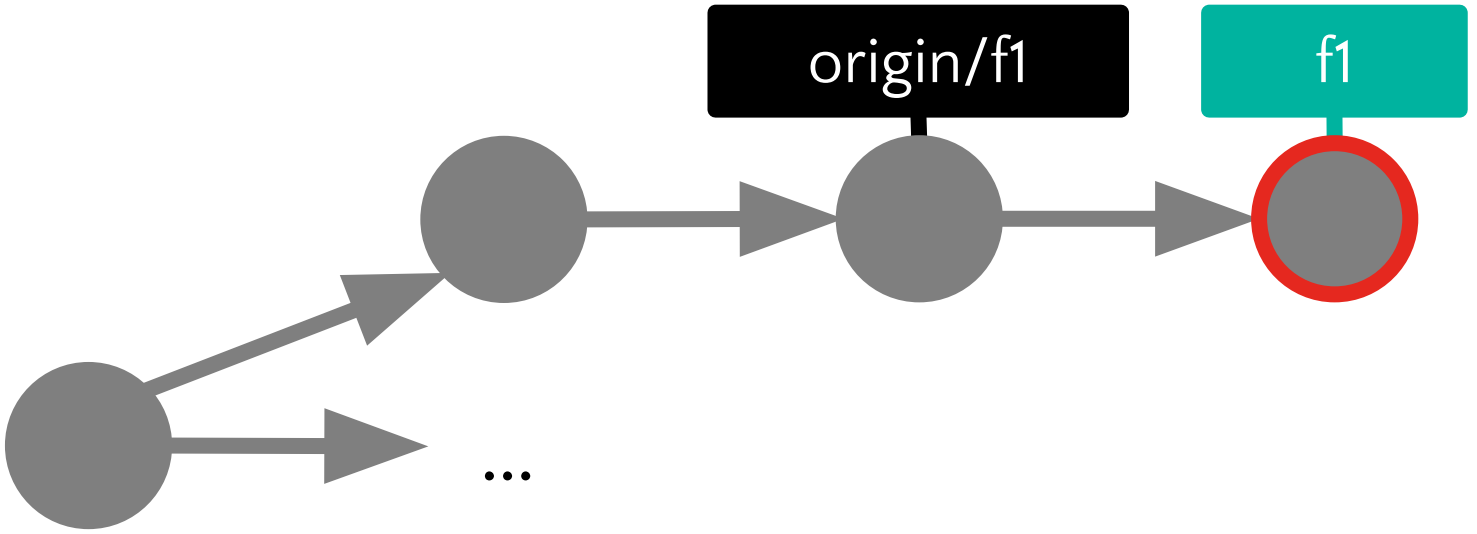



```
git checkout -b f1 ?
```

Начинаем править

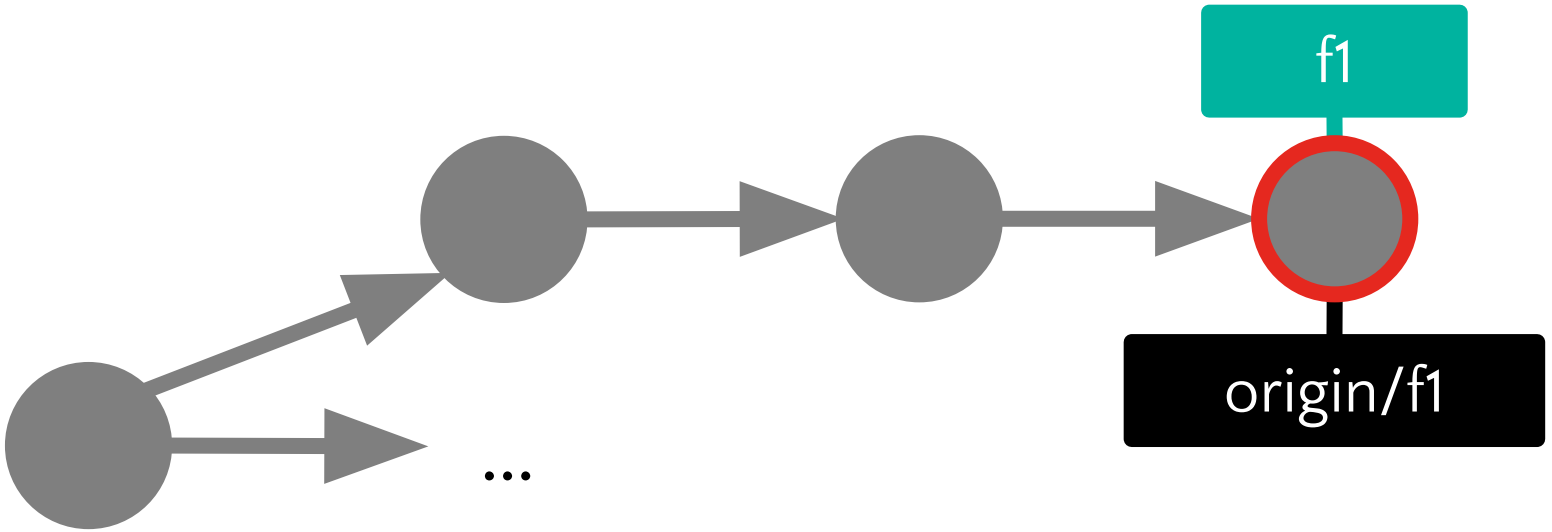


#3 Локальные изменения



`git` ?

#3 Локальные изменения



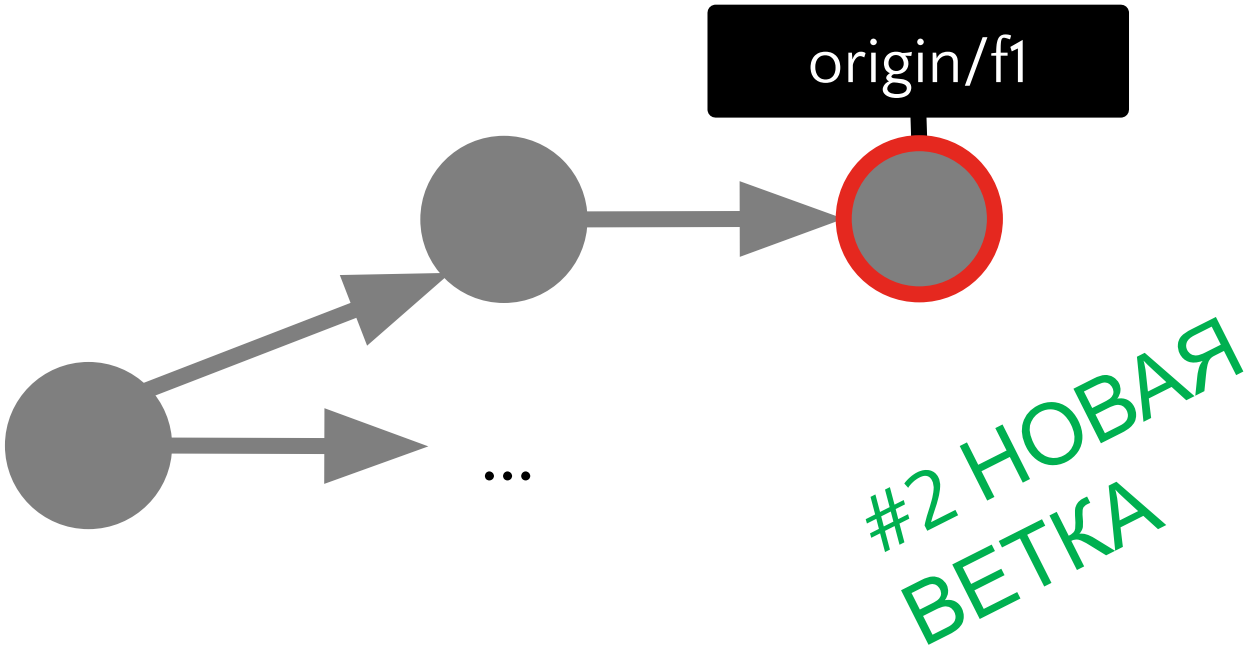
`git` ?

Таблица веток

```
$ git branch -vv
f1     7e424c3 [origin/f1]
master 1ae2a45 [origin/master]
f2     5ea463a
```

upstream branch or tracking branch

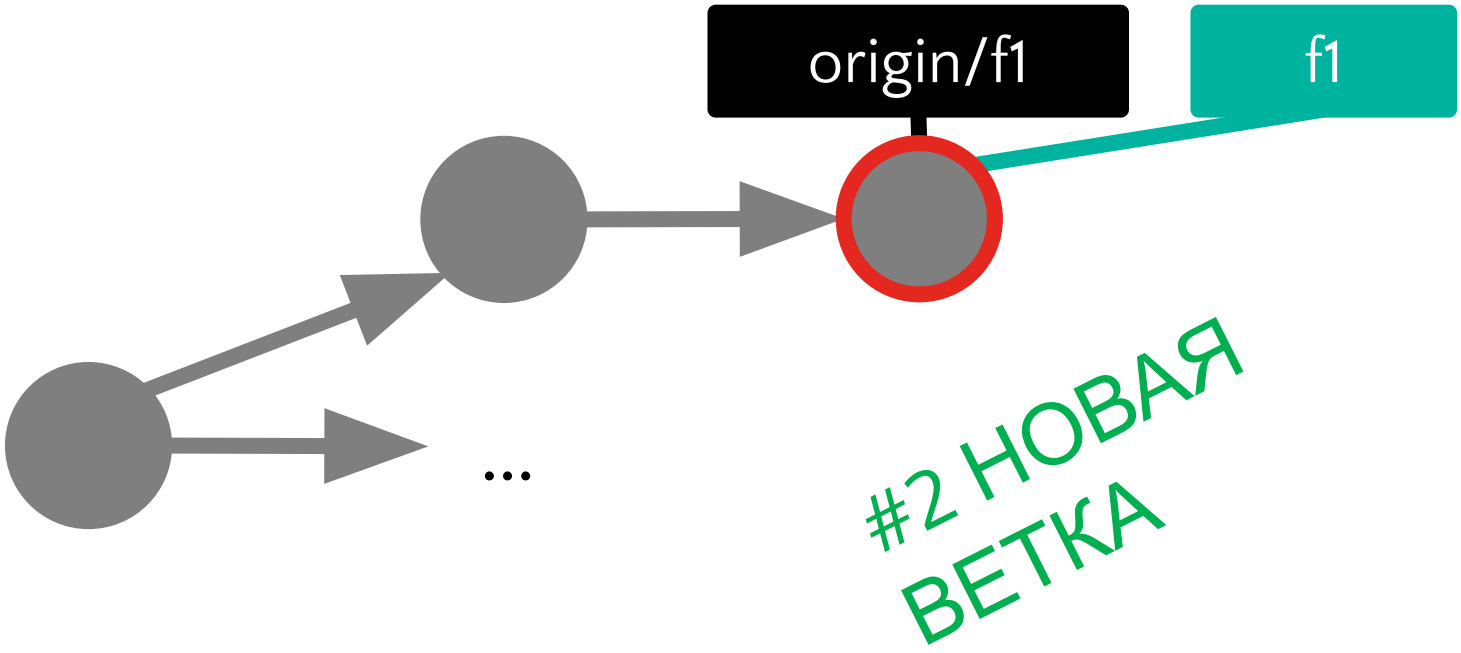
checkout upstream




```
git checkout -b f1 origin/f1
git checkout --track origin/f1

git checkout -b f1
```

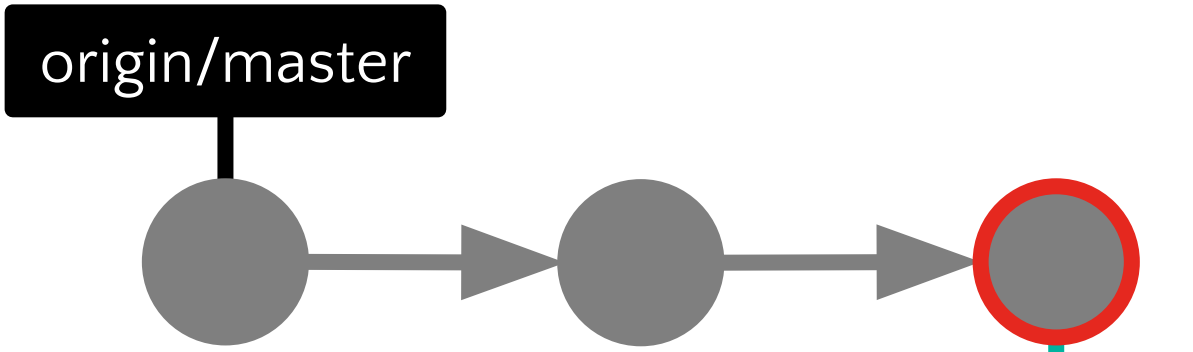
checkout upstream




```
git checkout -b f1 origin/f1
git checkout --track origin/f1

git checkout -b f1
```

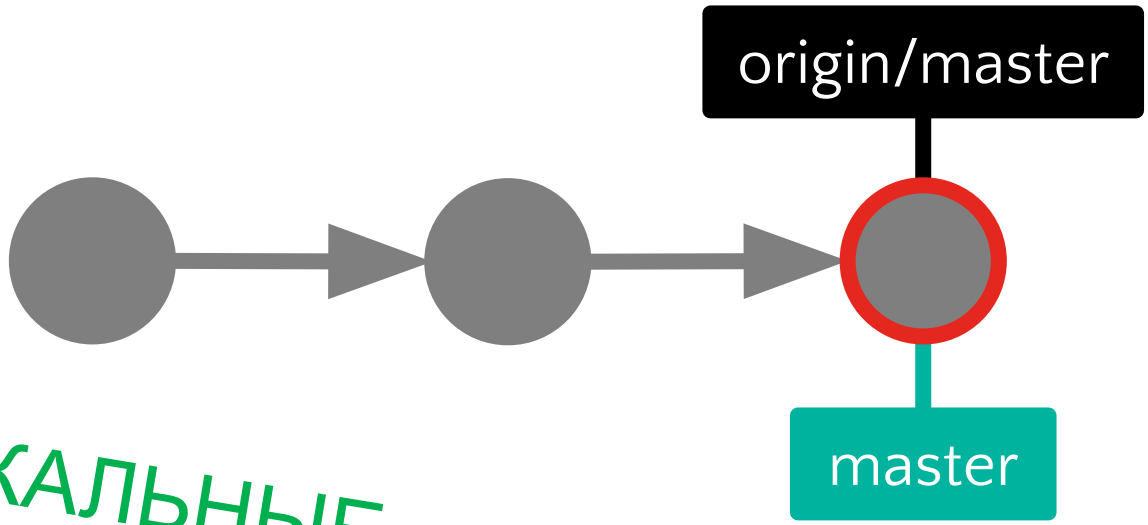
push



#3 ЛОКАЛЬНЫЕ ИЗМЕНЕНИЯ

```
git push
```

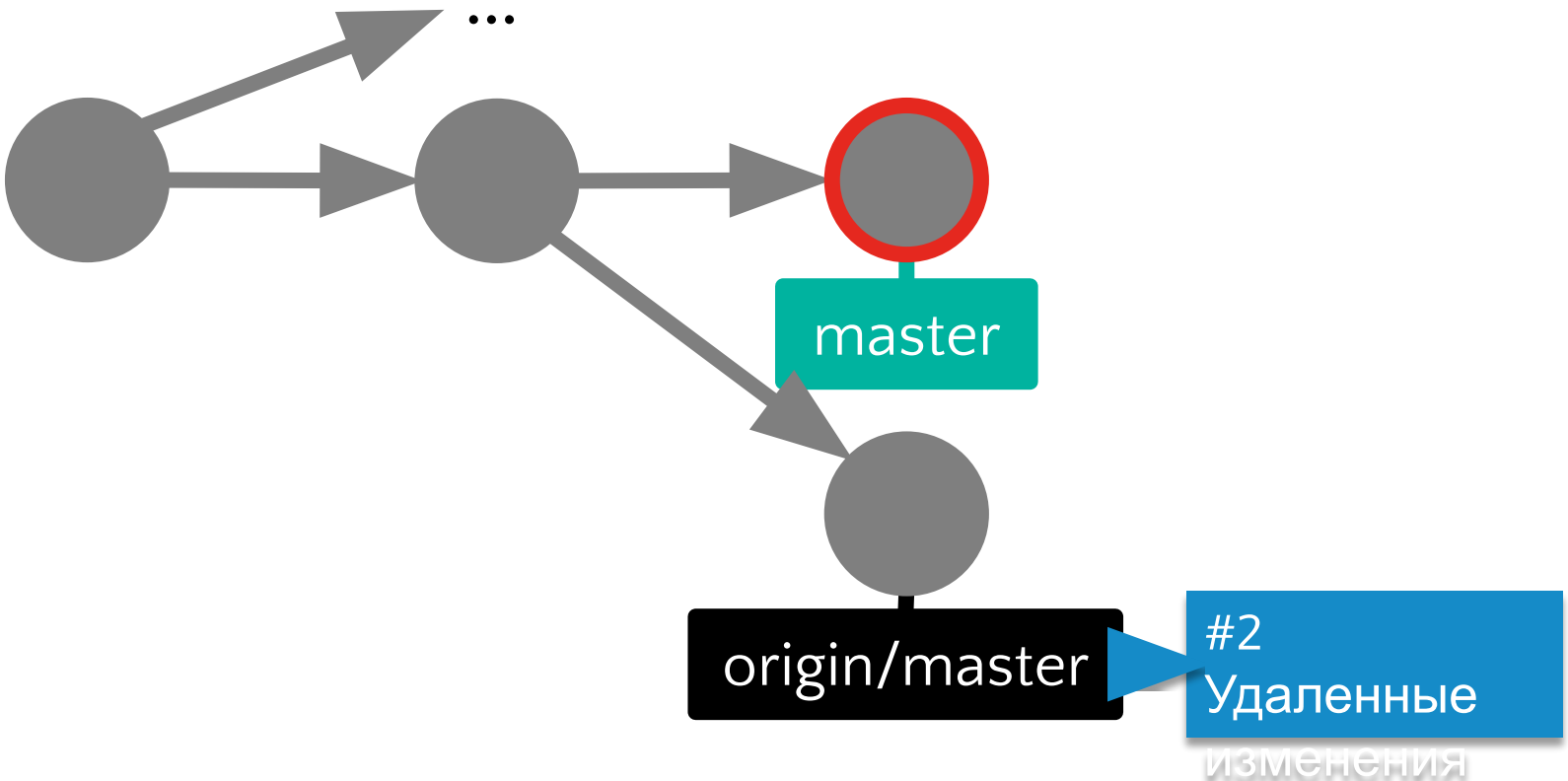
push



#3 ЛОКАЛЬНЫЕ ИЗМЕНЕНИЯ

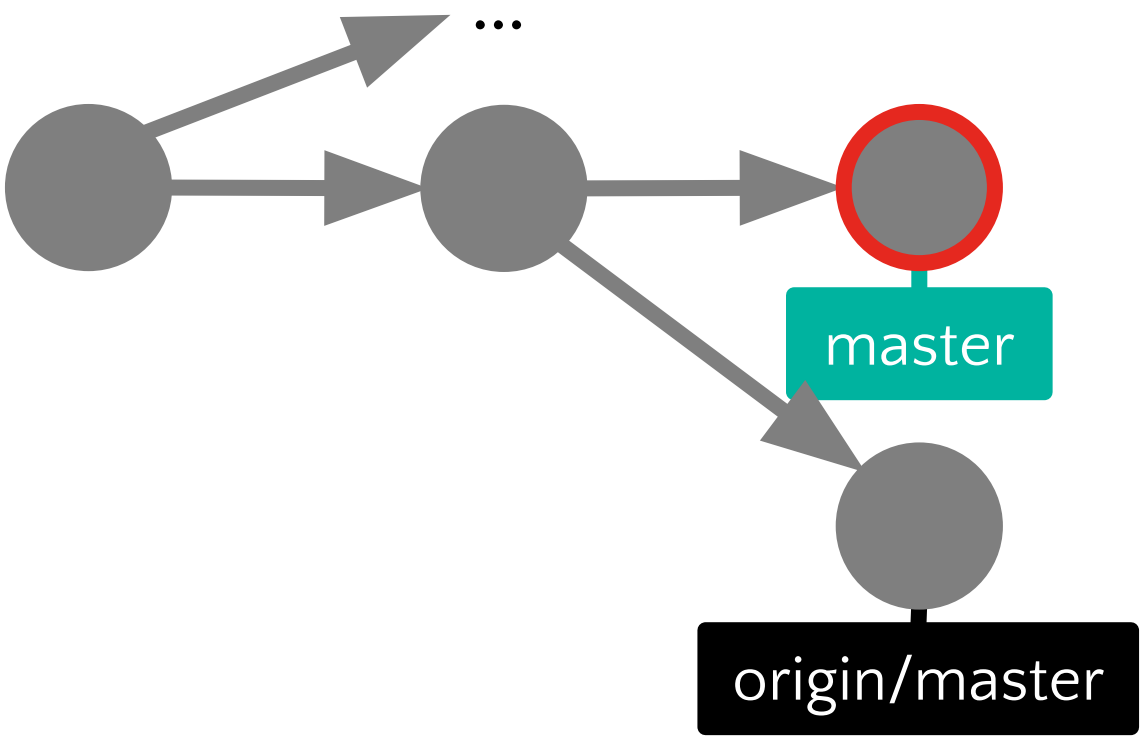
```
git push
```

Что делать?



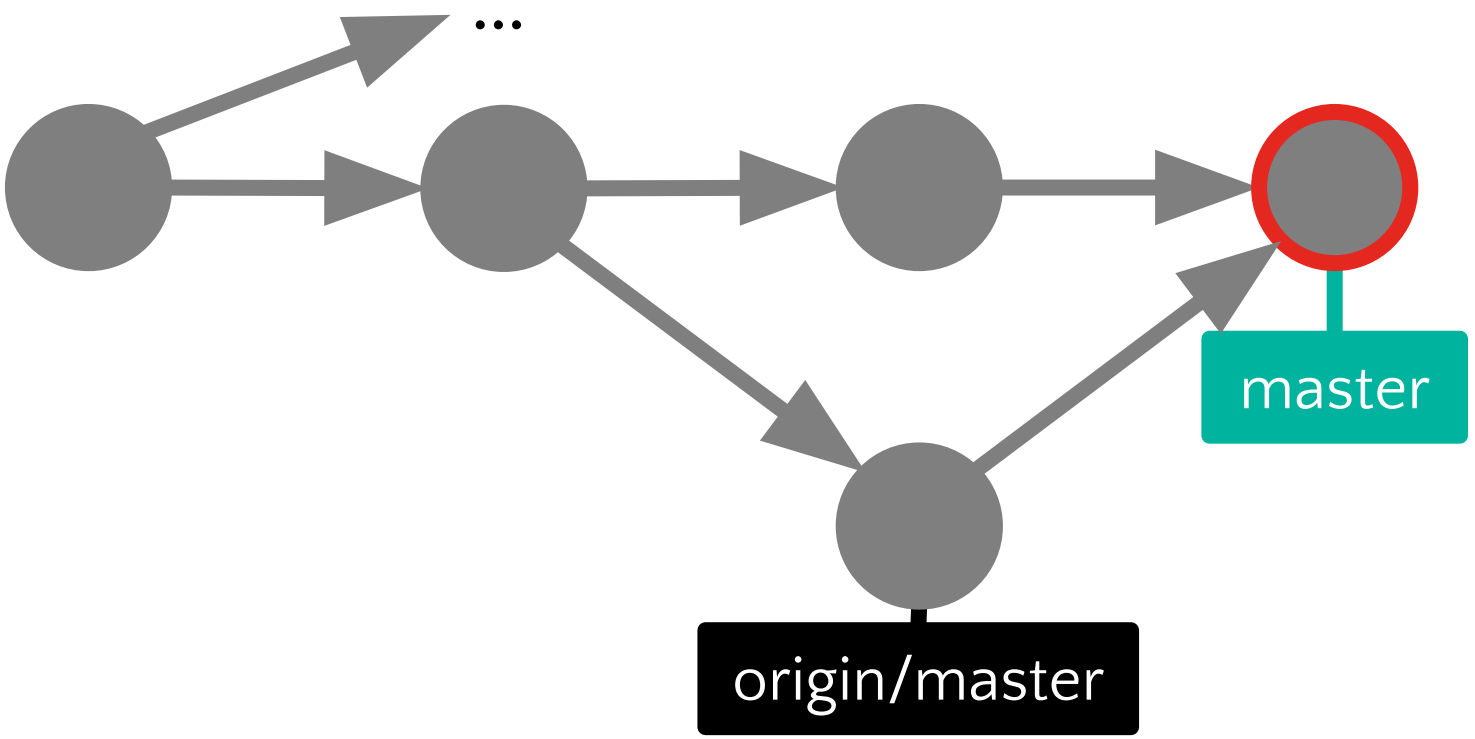
#2 Удаленные изменения

```
git merge origin/master
```
?

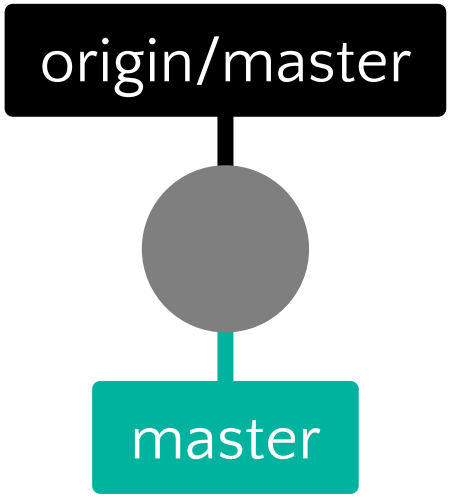


#2 Удаленные изменения

`git merge origin/master` ?



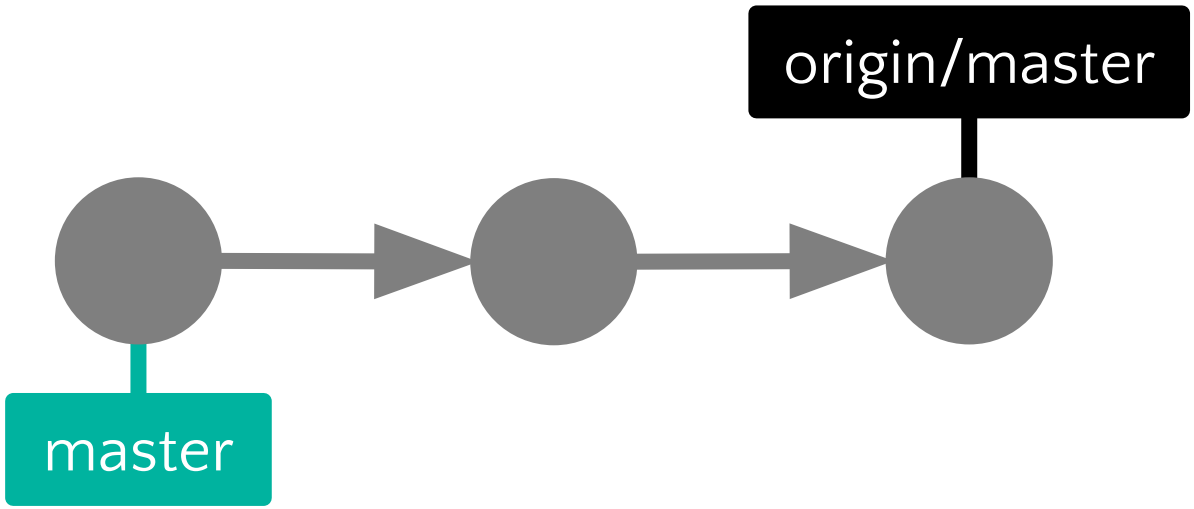
pull



#2 УДАЛЕННО УЕХАЛО

```
git pull origin =
    git fetch + git merge origin/<current_branch>
```

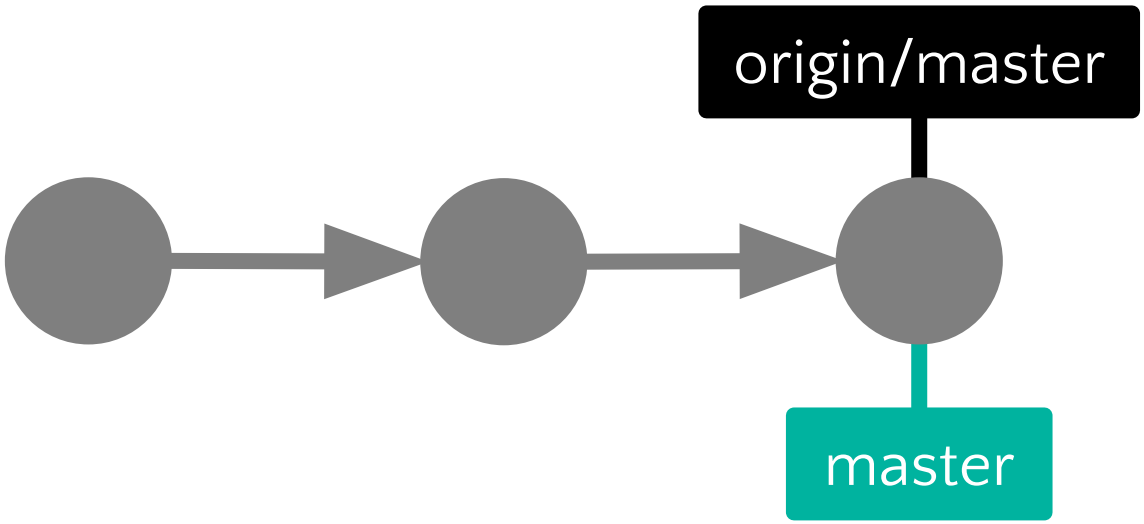
pull



#2 УДАЛЕННО УЕХАЛО

```
git pull origin =
    git fetch + git merge origin/<current_branch>
```

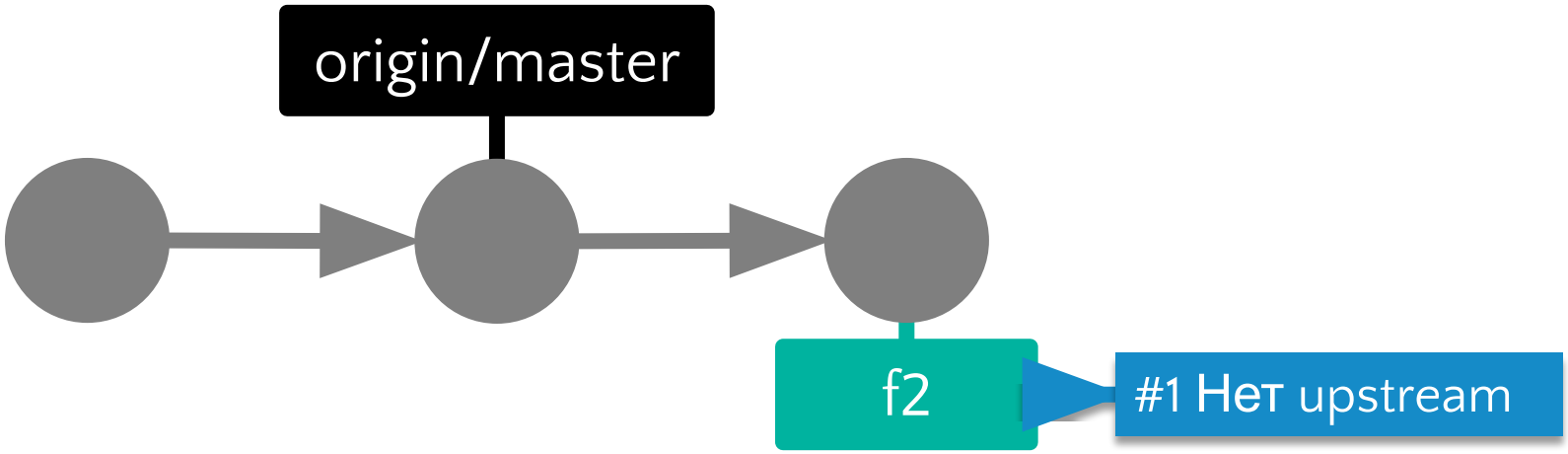
pull



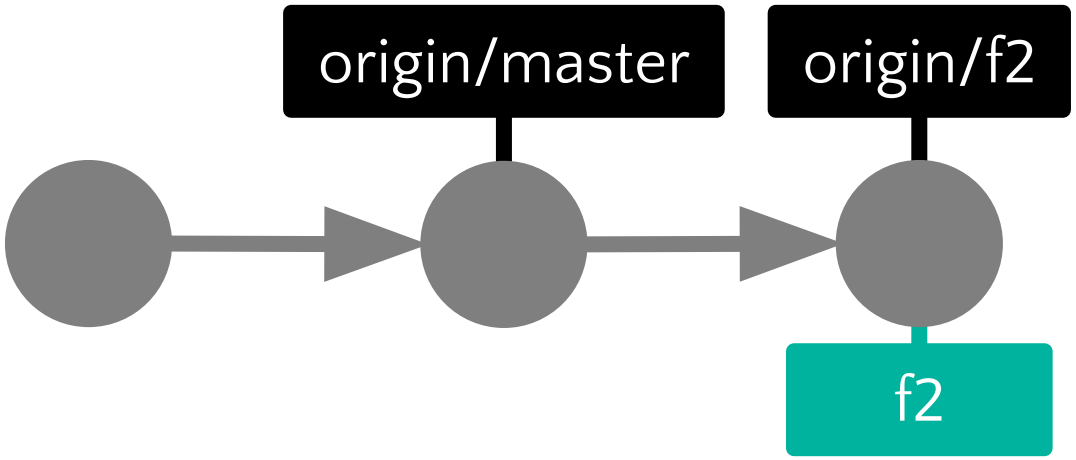
#2 УДАЛЕННО УЕХАЛО

```
git pull origin =
    git fetch + git merge origin/<current_branch>
```

Как начать фичу?

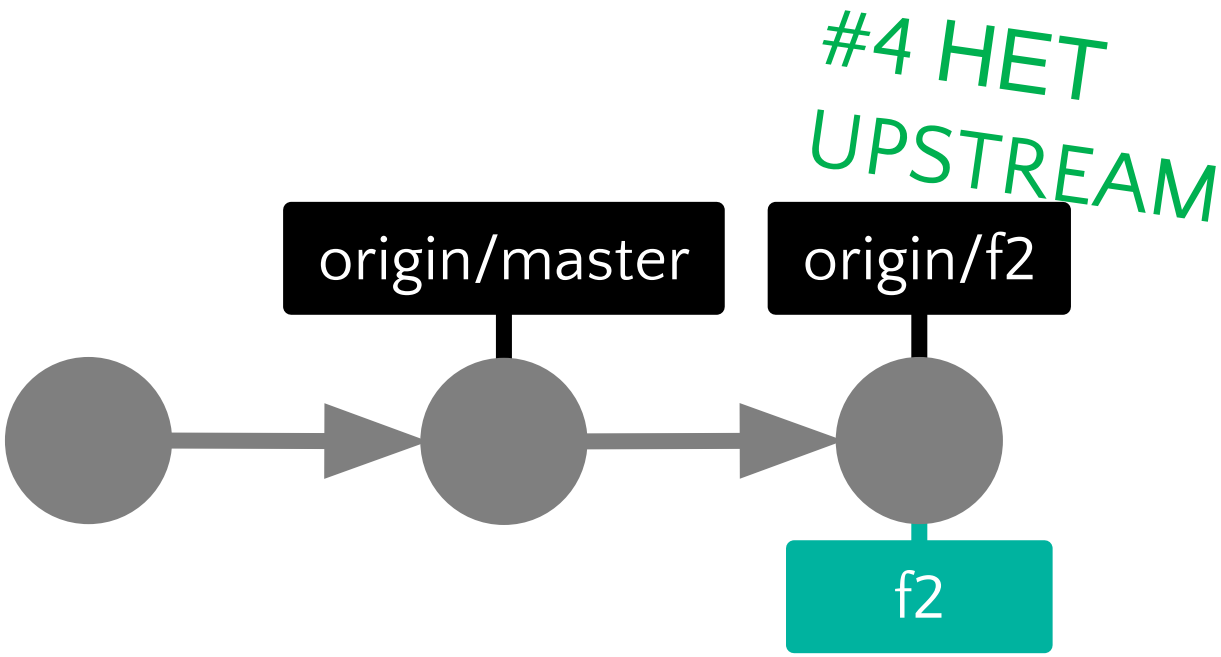


#4 Нет upstream



`git push` ?

push ветки




```
git push <remote> <localBranch>:<remoteBranch>
git push origin f2:f2
git push -u origin HEAD
```

Удаление ветки

Локально

```
git branch -d <branchName>
```

Удаленно

```
git push origin --delete <branchName>
```

```
git push origin :<branchName>
```

Конфигурирование

Конфигурирование

git config --system
    %ProgramFiles%\Git\mingw64\etc\gitconfig

git config --global
    %USERPROFILE%\.gitconfig

git config
    <repo>/.git/config

Настройка авторства

.gitconfig

```
[user]
    name = username
    email = username@mail.com
```

команды

```
git config --global user.name Ivan Domashnikh
git config --global user.email domashnikh@mail.com
```

Настройка для Windows

.gitconfig

```
[core]
    autocrlf = true
    safecrlf = true
```

команды

```
    git config --global core.autocrlf true
    git config --global core.safecrlf true
```

autocrlf – преобразование \r\n в \n

safecrlf – проверка обратимости преобразования \r\n в \n

Выбор текстового редактора

.gitconfig

```
[core]
    editor = notepad
```

будет использоваться для ввода commit message

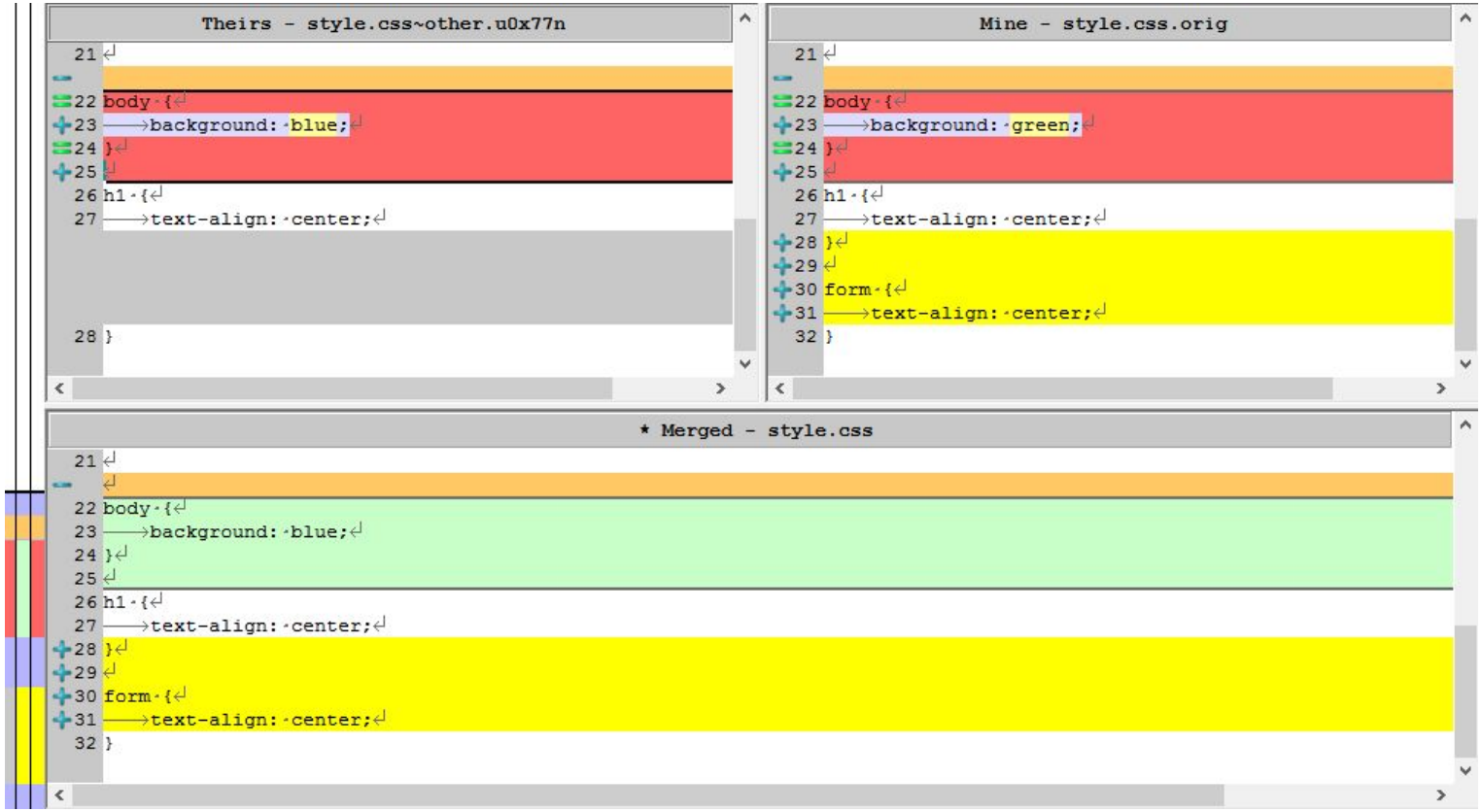
Выбор merge tool

.gitconfig

```
[merge]
   tool = TortoiseMerge
```

позволяет решать конфликты

Разрешение конфликта в TortoiseMerge



Git Alias

.gitconfig

```
[alias]
    co = checkout
    ci = commit
    st = status
    br = branch
    hist = log --pretty=format:\"%h %ad | %s%d
[%an]\" --graph --date=short
    type = cat-file -t
    dump = cat-file –p
    undo = reset --hard HEAD~1
    sm = submodule
```

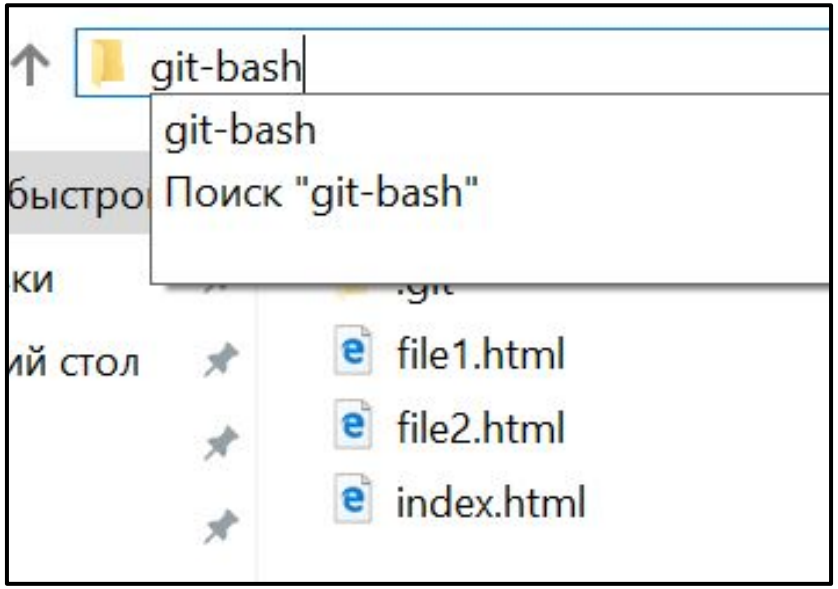
Bash Alias

%USERPROFILE%\.bashrc

```
alias less='less -r'
# --show-control-chars: help showing Korean or accented
characters
alias ls='ls -F --color --show-control-chars'
alias ll='ls -l'
alias gs='git status '
alias ga='git add '
alias gb='git branch '
alias gc='git commit'
alias gd='git diff'
alias go='git checkout '
alias gk='gitk --all&'
alias gx='gitx --all'
alias got='git '
alias get='git '
```

Git Bash везде

Если добавить C:\Program Files\Git в PATH,
то git-bash будет доступен через Win+R и адресную строку Проводника



Игнорирование файлов

.gitconfig

```
[core]
    excludesFile
```

для репозитория

```
    <repo>/.git/info/exclude
```

в любой папке и ее подпапках

```
    .gitignore
```

OpenSSH

1. выполнить в Git Bash
   `ssh-keygen -t rsa -C “keyname”`
2. найти открытый ключ `%USERPROFILE%\.ssh\id_rsa.pub`
3. Зарегистрировать открытый ключ на сервере

PuTTY SSH

Git Extensions может использовать как PuTTY, так OpenSSH

1. открыть в **Git Extensions /Tools/PuTTY/Generate or import key**
2. сгенерировать ключ SSH-2 RSA, нажав **Generate**, а затем поводив мышкой вплоть до генерации ключа
3. открытый ключ появится прямо в диалоговом окне – его следует сохранить и зарегистрировать на сервере
4. на диск следует сохранить файл с расширением .ppk
5. теперь при выполнении clone/fetch/push можно нажать **Load SSH key** и выбрать для использования закрытый ключ *.ppk

Доверенные узлы

может требоваться доверие к удаленному репозиторию

список доверенных

`%USERPROFILE%\.ssh\known_hosts`

для добавления в доверенные

выполнить в Git Bash

`ssh <host>`

e.g. *`ssh github.com`*

для подтверждения соединения ввести

`yes`

Напоследок

amend

Можно добавить изменения в уже сделанный коммит

```
git commit --amend
```

reflog

Позволяет отобразить историю перемещений HEAD

Спасает, когда на нужный коммит не указывает ни тэг, ни ветка

```
git reflog
```

```
247ed61 HEAD@{0}: commit: Fix bug
b7cad06 HEAD@{1}: checkout: moving from master to
cool-feature
b7cab06 HEAD@{2}: clone: from
https://github.com/awesome-application
```

clean

Удаление недобавленных в git файлов

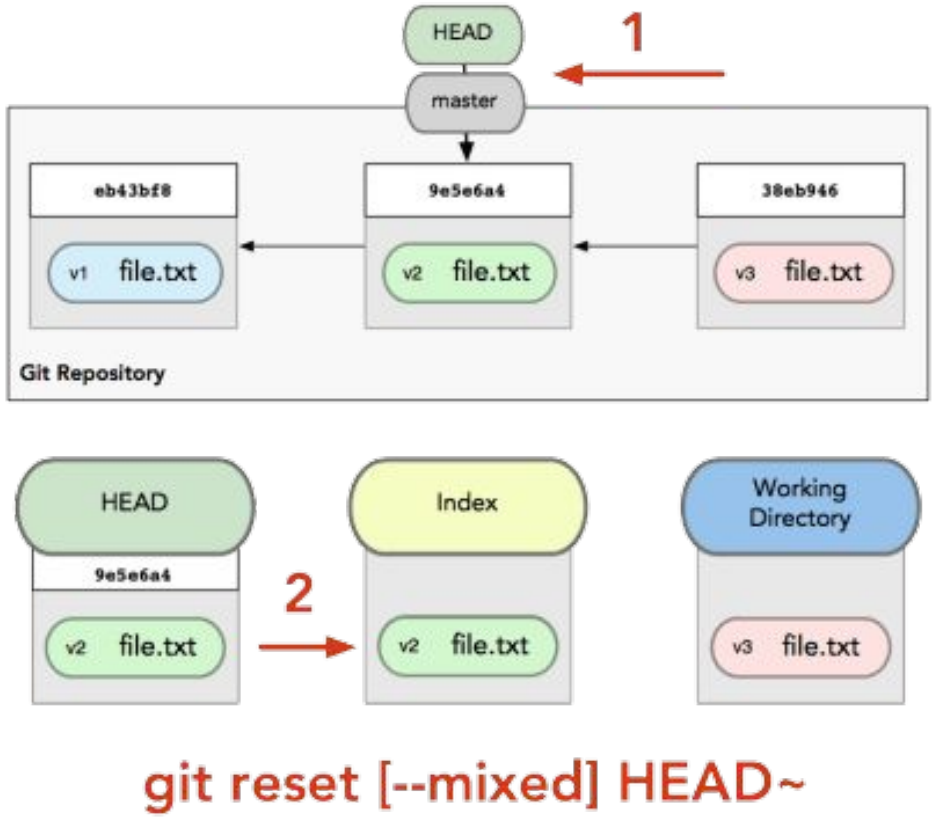
```
git clean
```

reset

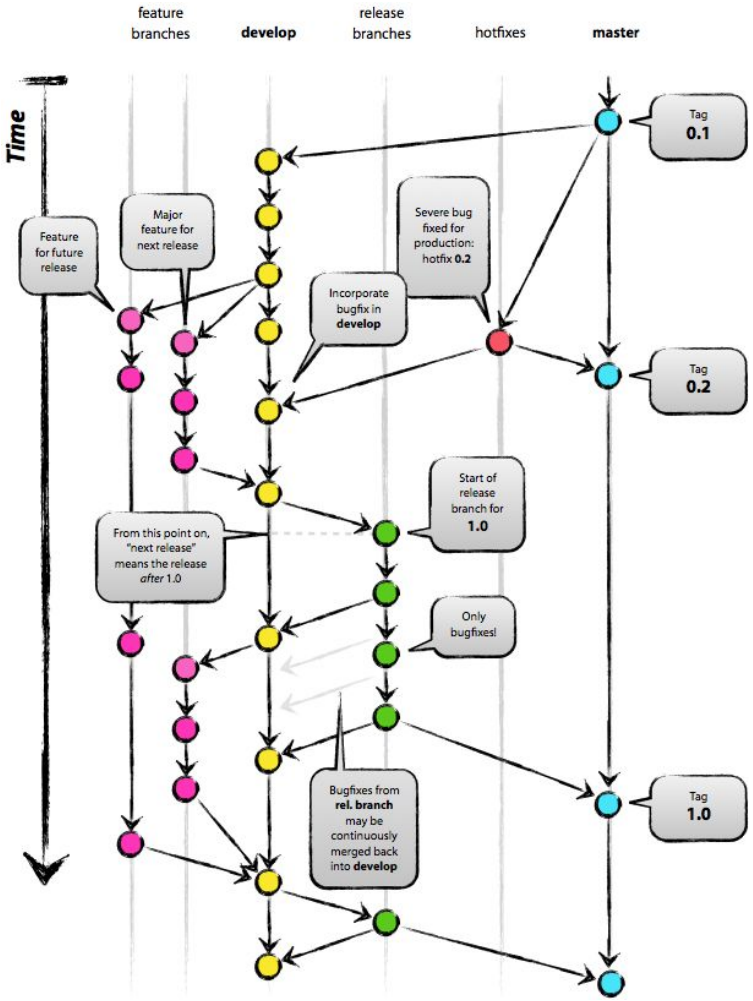
soft
переносит HEAD и ветку

mixed
+ переписывает Index

hard
+ переписывает WD



flow


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Чтиво

Классный практический курс на русском
https://githowto.com/ru

Про remote branches
Глава в Pro Git

Про reset
https://git-scm.com/blog/2011/07/11/reset.html

Основные фичи с картинками на английском
https://www.atlassian.com/pt/git/tutorial

Pro Git
https://git-scm.com/book/ru/v2

Кач

Git

https://git-scm.com/

Git Extensions

https://sourceforge.net/projects/gitextensions/

TortoiseGit

https://tortoisegit.org/

SourceTree

https://www.sourcetreeapp.com/

Вопросы?