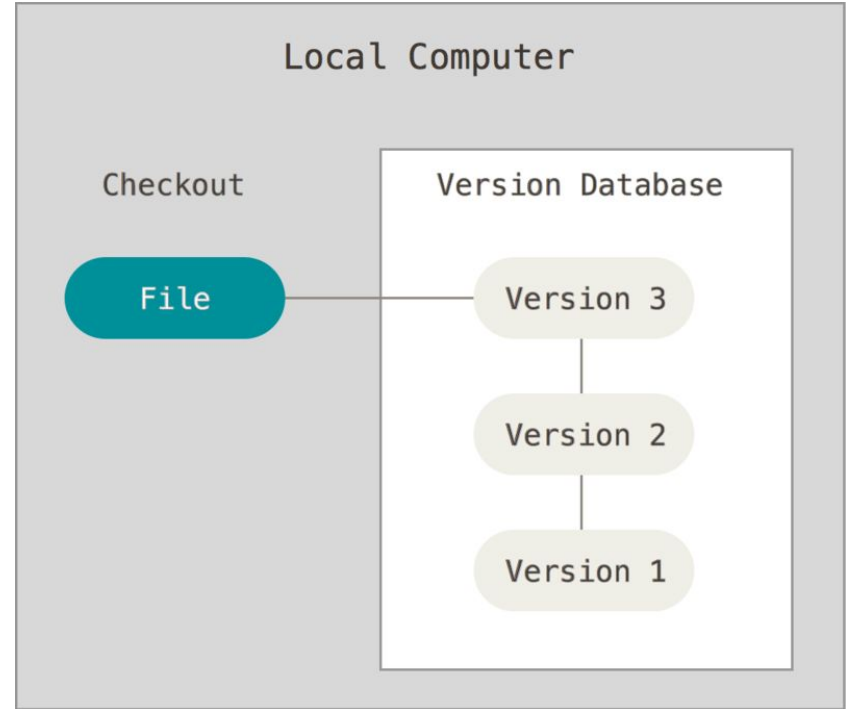


Git

Version Control System (VCS)

Система контроля версий

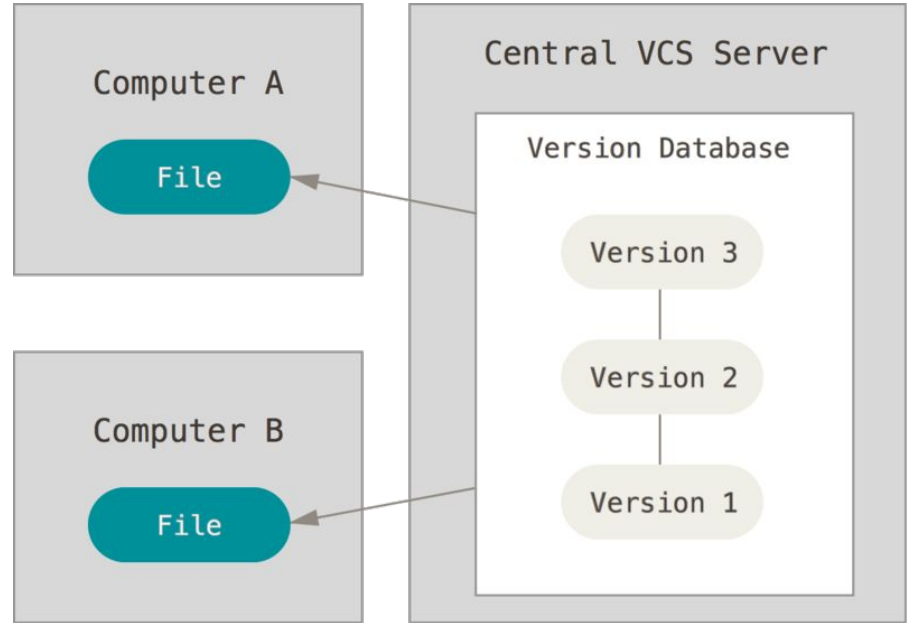
записывает изменения файлов и позволяет вернуться к определенной версии (если, например, приложение упало, то можно откатиться на стабильную версию).



Centralized Version Control Systems

Централизованные системы контроля версий (CVCS) были придуманы для взаимодействия нескольких разработчиков.

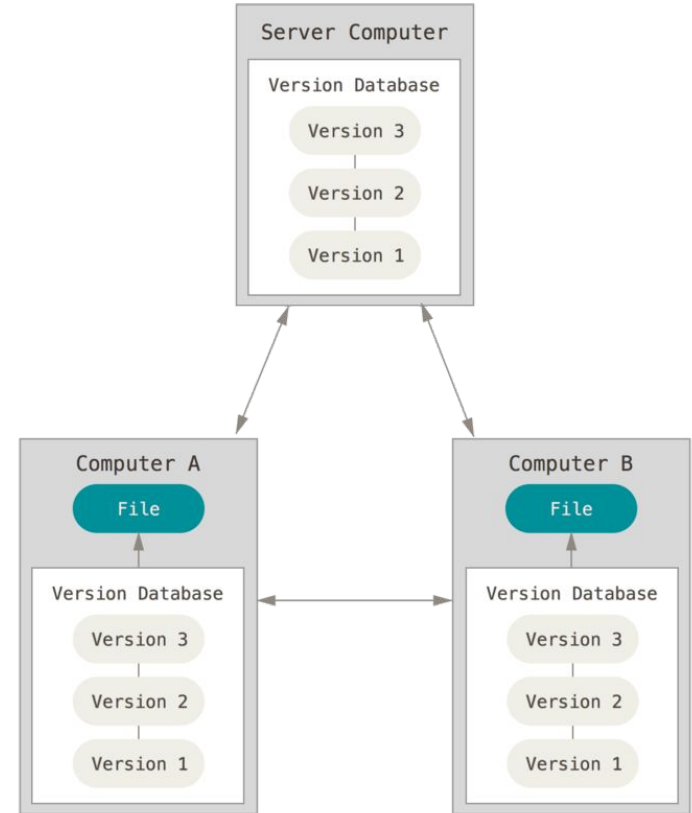
В таких системах существует единый сервер, хранящий все версии файлов, куда разработчики отправляют свои изменения.



Distributed Version Control Systems

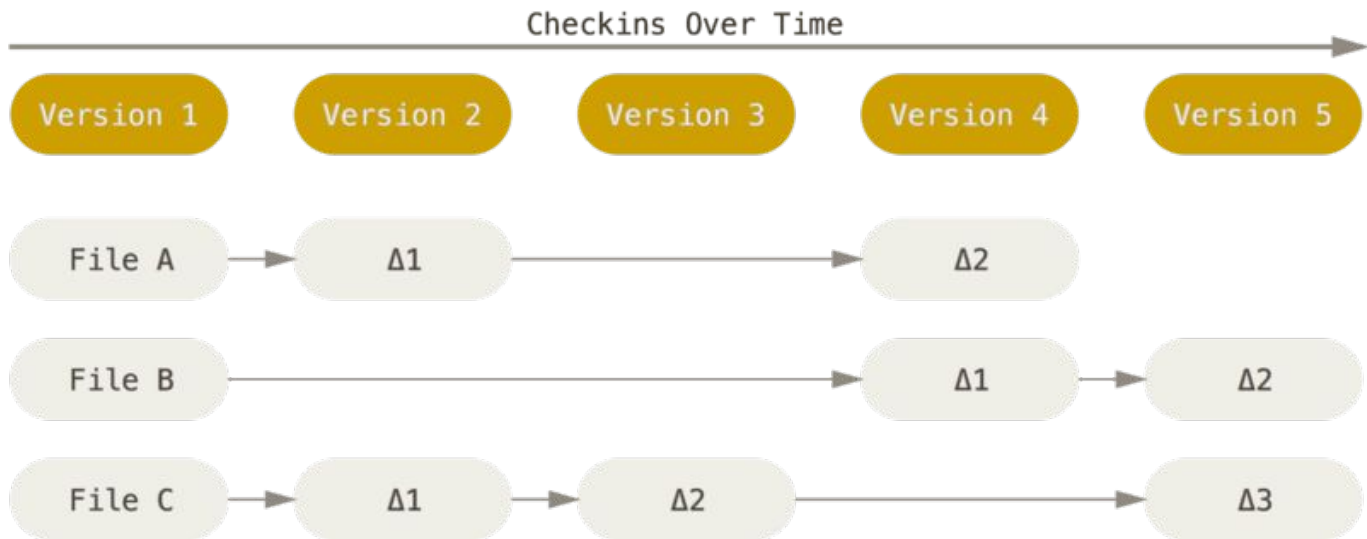
В распределенных системах контроля версий **DVCS** у каждого разработчика локально сохранена полная история изменений проекта.

- Различные операции выполняются гораздо быстрее
- Нет единой точки отказа
- DVCS не требуют постоянного подключения к интернету



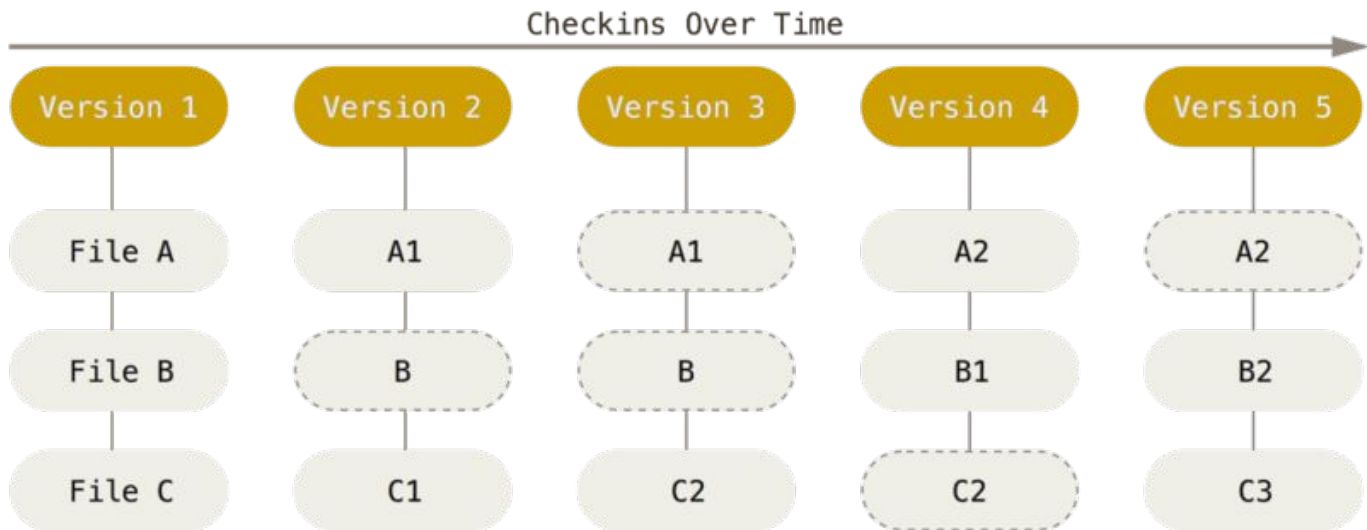
Other VCS

Другие системы хранят набор файлов и изменения этих файлов (delta-based version control).



Git

Git хранит данные в виде “снимков” (**snapshot**) файловой системы.



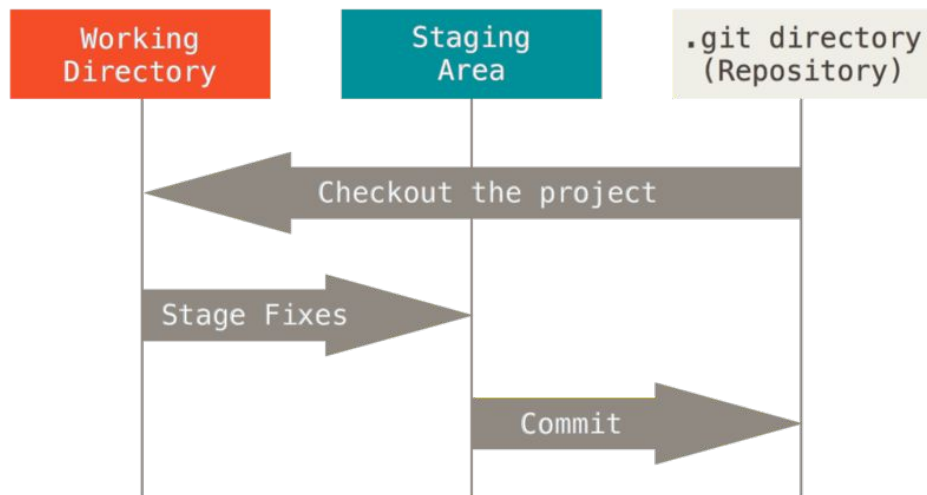
Git

Git – распределенная система контроля версий.

- Git хранит информацию о файлах, а не о изменениях в них
- Почти все операции не требуют подключения к сети
- Для всех объектов вычисляется хеш-сумма и обращение идёт по ней
- В Git обычно только добавляются данные

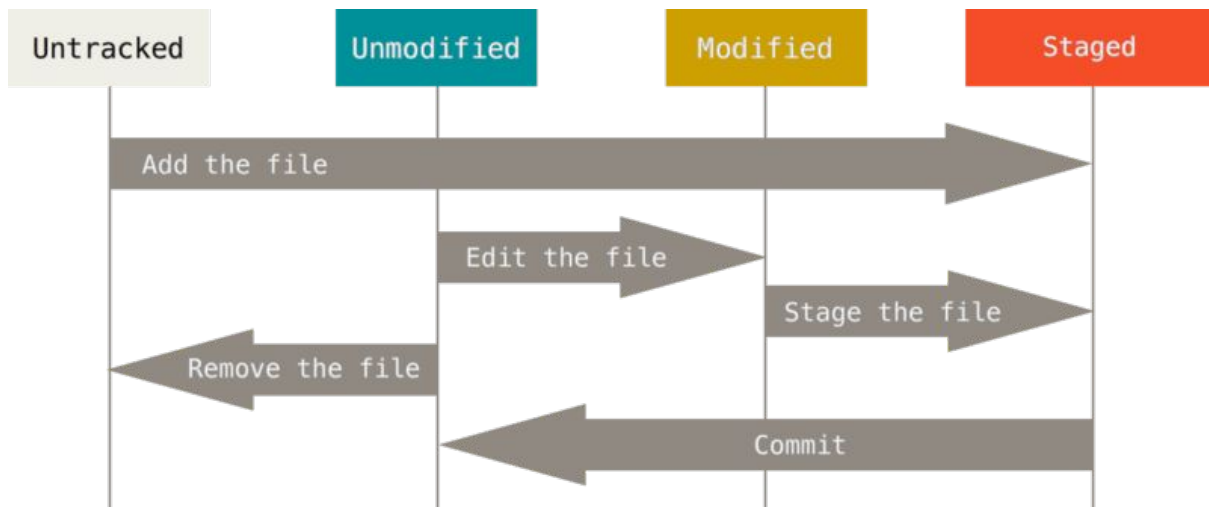
Git sections

- Рабочая директория (working tree)
- Индекс (staging area)
- Репозиторий



File states

- **Неотслеживаемые (untracked)** – не находящиеся под версионным контролем
- **Неизмененный (unmodified)** – находится и в рабочей директории, и в репозитории
- **Измененные (modified)** – поменялись, но изменения не были зафиксированы
- **Индексированные (staged)** – подготовленные к коммиту



Configuration

Доступны 3 уровня конфигурации: **--local**, **--global**, **--system**

```
git config --global user.name "Ivan Ivanov"
```

```
git config --global user.email ivan.ivanov@email.com
```

Имя пользователя и его почта являются частью каждого коммита

Initializing repository

git init – инициализация нового локального репозитория

git clone <url> – получение удаленного репозитория (например, с github/gitlab)

```
$ git clone https://github.com/git/git.git
Cloning into 'git'...
remote: Enumerating objects: 329570, done.
remote: Counting objects: 100% (2/2), done.
remote: Total 329570 (delta 1), reused 1 (delta 1), pack-reused 329568
Receiving objects: 100% (329570/329570), 193.65 MiB | 4.83 MiB/s, done.
Resolving deltas: 100% (246405/246405), done.
Updating files: 100% (4167/4167), done.
```

Checking file status

git status – отображает файлы, отличные от последнего коммита, и файлы, не отслеживаемые Git

```
USER@windows-uj49s6b MINGW64 ~/Desktop/git-demo (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    index.html

nothing added to commit but untracked files present (use "git add" to track)
```

Tracking and indexing

git add <файл> – добавление файла к индексу.

git commit -m “<сообщение>” – фиксирование изменений.

```
USER@windows-uj49s6b MINGW64 ~/Desktop/git-demo (master)
$ git add .

USER@windows-uj49s6b MINGW64 ~/Desktop/git-demo (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   index.html

USER@windows-uj49s6b MINGW64 ~/Desktop/git-demo (master)
$ git commit -m "Init"
[master (root-commit) f6876ae] Init
1 file changed, 9 insertions(+)
create mode 100644 index.html
```

Undoing

git commit --amend – замена предыдущего коммита

git reset HEAD <файл> – удаление файла из индекса

git restore <файл> – сброс изменений в файле

git revert <коммит> – создание нового коммита, отменяющего предыдущий

Listing history and comparing changes

git diff <файл> – отображение изменений в файле

git log – отображение коммитов в истории

- **--oneline**
- **--graph**
- **--patch (-p)**

```
USER@windows-uj49s6b MINGW64 ~/Desktop/git-demo (master)
$ git diff index.html
diff --git a/index.html b/index.html
index 9305ae7..8274e3c 100644
--- a/index.html
+++ b/index.html
@@ -1,7 +1,7 @@
<html lang="en">
  <head>
    <meta charset="UTF-8" />
-    <title>Git Demo</title>
+    <title>Git Demo!</title>
  </head>
  <body>
    <div>Hello, world!</div>
```

```
USER@windows-uj49s6b MINGW64 ~/Desktop/git-demo (master)
$ git log
commit 1bb18e5ea0f9c5005f1c8753e77d15350dd93bd1 (HEAD -> master)
Author: Sergey Koloydenko <skoloydenko@mail.ru>
Date:   Wed Jul 13 16:14:22 2022 +0300

    Init
```

.gitignore

Не все файлы стоит отслеживать!

Файлы и папки внутри **.gitignore** не отслеживаются системой контроля версий

```
# .gitignore

# dependencies
/node_modules
/node
/.pnp
.pnp.js

# testing
/coverage

# production
/build

# env
.env
```


Commits

Git хранит набор коммитов

Коммит – “снимок” состояния проекта

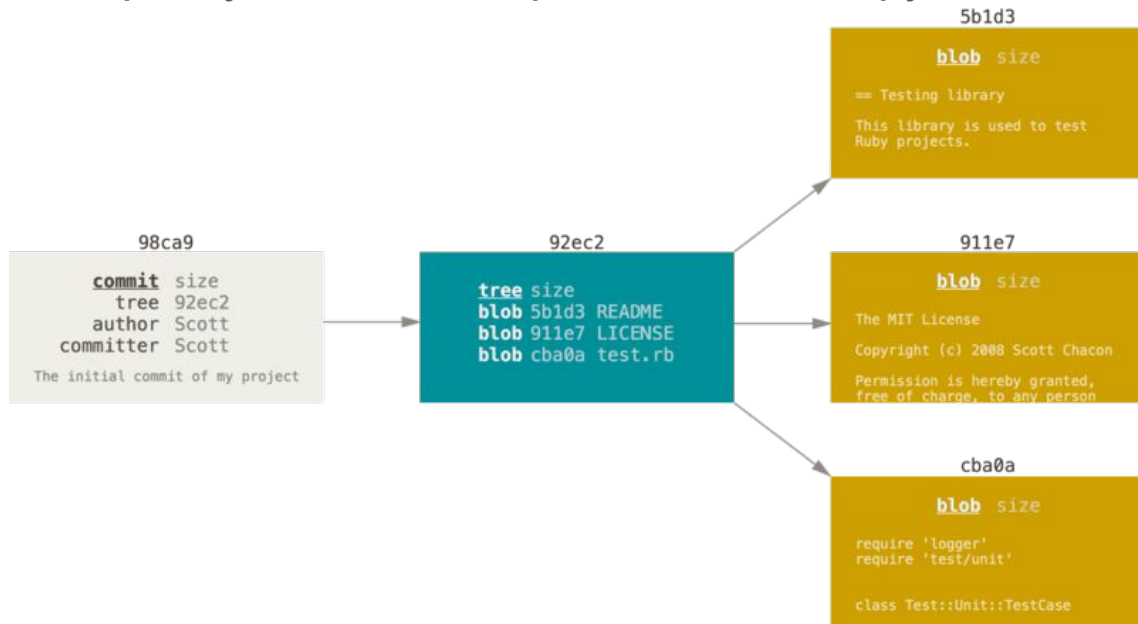
Каждый коммит ссылается на предыдущий, формируя цепочку



Git objects

Blob – объект для хранения содержимого файлов

Tree – объект, который указывает на разные blob и другие tree

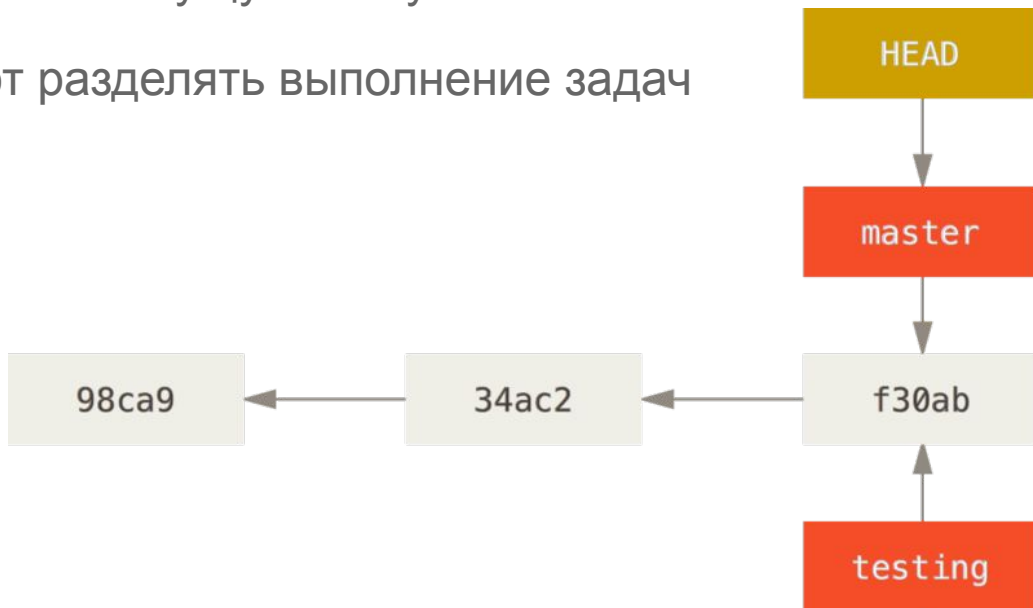


Branches

Ветка – указатель на коммит

HEAD указывает на текущую ветку

Ветки позволяют разделять выполнение задач



Branches

`git branch <имя_ветки>` – создание ветки

`git checkout <имя_ветки>` – переключение на другую ветку

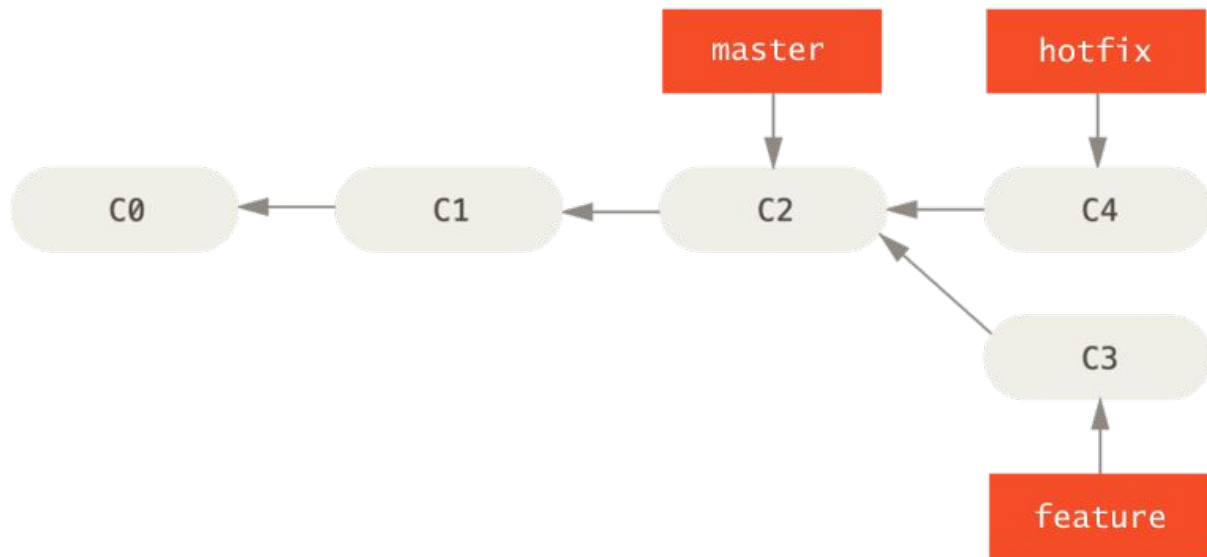
`git checkout -b <имя_ветки>`

```
USER@windows-uj49s6b MINGW64 ~/Desktop/git-demo (master)
$ git branch develop
```

```
USER@windows-uj49s6b MINGW64 ~/Desktop/git-demo (master)
$ git checkout develop
Switched to branch 'develop'
```

Merge

По завершению работы в ветке её стоит слить с главной веткой

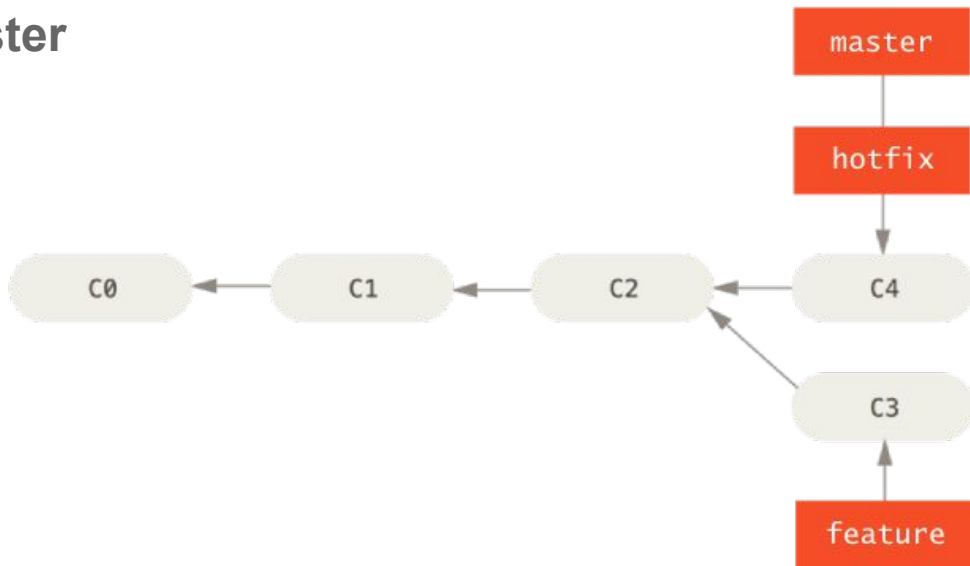


Merge

Сольем ветку hotfix с master. Произошел “fast-forward”, так как коммит C4 был прямым потомком C2

`git checkout master`

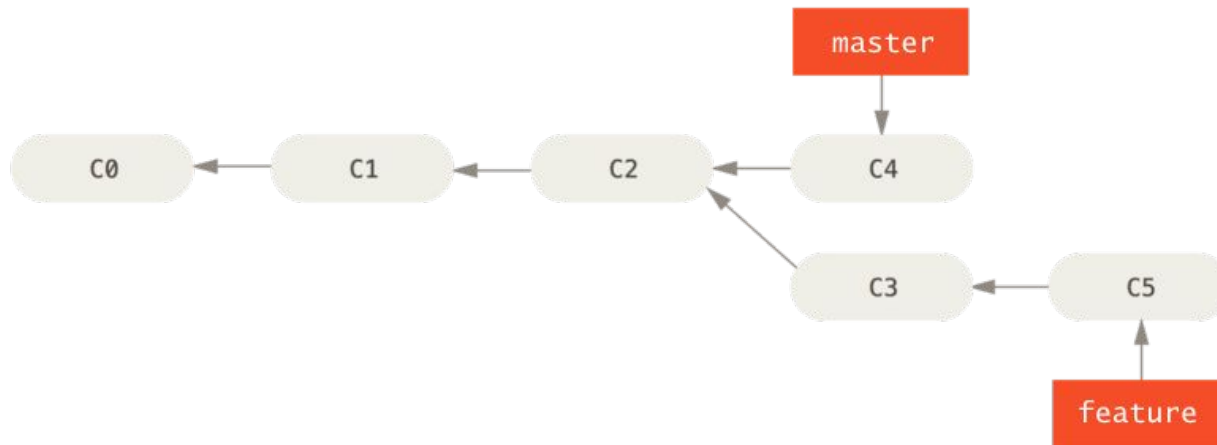
`git merge hotfix`



Merge

Удалим ненужную ветку hotfix

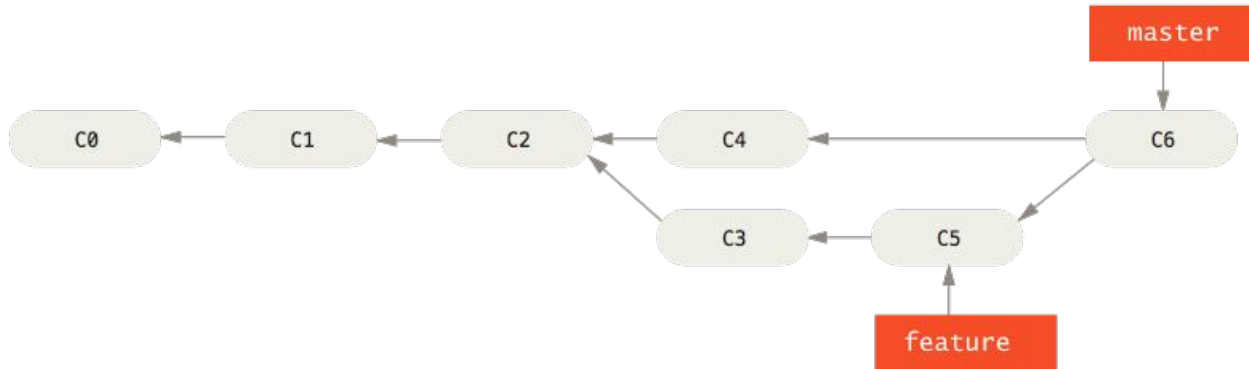
git branch -d hotfix



Merge

git checkout master

git merge feature



Merge conflict

Конфликт слияния возникает когда Git не может автоматически слить два коммита (например, при изменении одной и той же части файла)

```
<head>
<meta charset="UTF-8" />
<title>Git Demo!</title>
</head>
```

```
<head>
<meta charset="UTF-8" />
<title>Git Demo?</title>
</head>
```

```
USER@windows-uj49s6b MINGW64 ~/Desktop/git-demo (master)
$ git merge develop
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

Merge conflict

Для разрешения конфликта нужно изменить содержимое конфликтующих строк и закоммитить изменения

Можно использовать **git mergetool** или графический интерфейс среды разработки

```
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Git Demo!</title>
</head>
<body>
  <div>Hello, world!</div>
</body>
</html>
~
~
AL_2143.html [dos] (16:59 13/07/2022)1,1 Бесъ
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Git Demo!</title>
</head>
<body>
  <div>Hello, world!</div>
</body>
</html>
~
~
index.html [dos] (16:59 13/07/2022)
"index.html" [dos] 9L, 149B
```

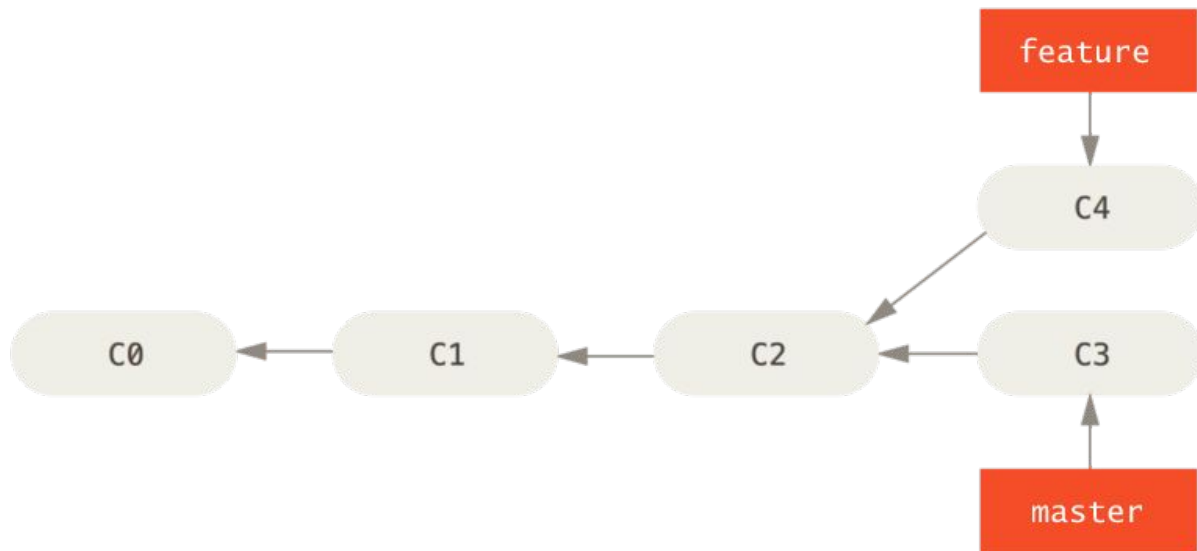
Git in IDE

```
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
4 <<<<<< HEAD (Current Change)
5   <title>Git Demo!</title>
6   =====
7   <title>Git Demo?</title>
8 >>>>>> develop (Incoming Change)
9   </head>
```

<pre>1 <html lang="en"> 2 <head> 3 <meta charset="UTF-8" /> 4- <title>Git Demo!</title> 5 </head> 6 <body> 7 <div>Hello, world!</div> 8 </body> 9 </html> 10</pre>	→	<pre>1 <html lang="en"> 2 <head> 3 <meta charset="UTF-8" /> 4+ <title>Git Demo?</title> 5 </head> 6 <body> 7 <div>Hello, world!</div> 8 </body> 9 </html> 10</pre>
--	---	--

Rebase

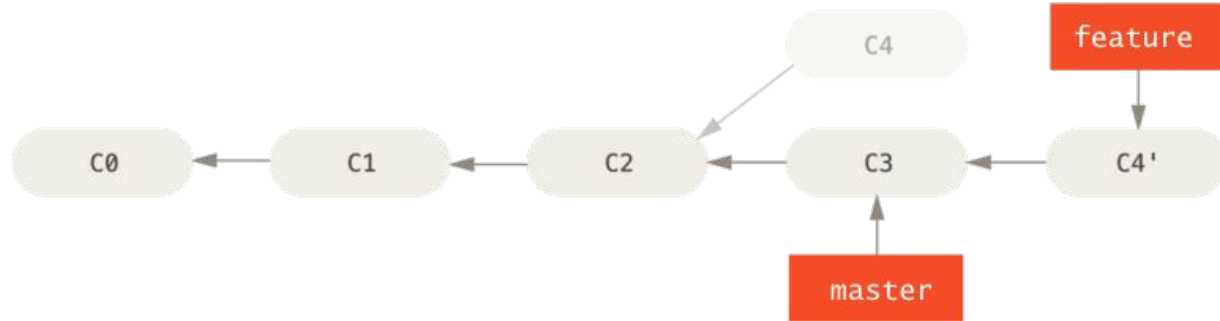
Альтернатива merge, позволяет сохранять историю коммитов линейной



Rebase

git checkout feature

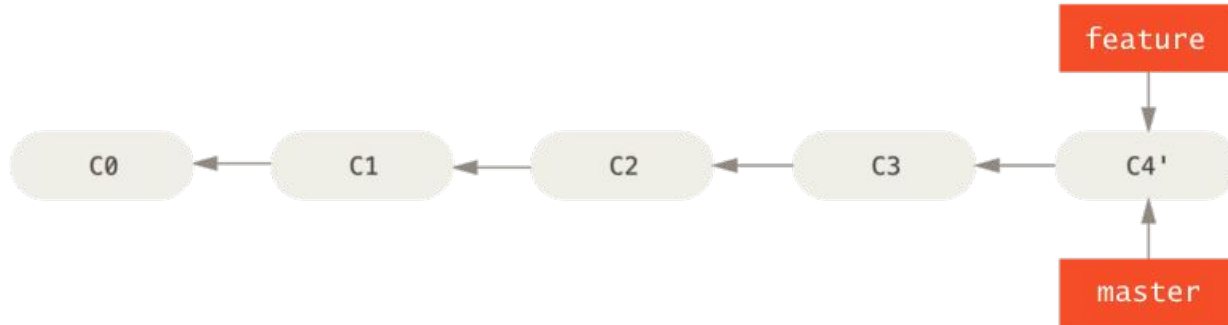
git rebase master



Rebase

git checkout master

git merge feature



Rebase

Rebase стоит использовать только в локальном репозитории. Если перебазировать файлы, уже находящиеся в удаленном репозитории, то другим разработчикам, забравшим себе копию кода, придется делать merge КОММИТ

Remote repositories

Удаленные репозитории позволяют взаимодействовать с другими разработчиками

- <https://github.com/>
- <https://gitlab.com/>
- <https://bitbucket.org/>

Remote repositories

`git remote add <name> <url>` – создание подключения к удаленному репозиторию

`git push origin <branch>` – отправка изменений в удаленный репозиторий

```
USER@windows-uj49s6b MINGW64 ~/Desktop/git-demo (main)
$ git remote add origin https://github.com/SKoloydenko/git-demo.git

USER@windows-uj49s6b MINGW64 ~/Desktop/git-demo (main)
$ git push -u origin main
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (10/10), 913 bytes | 456.00 KiB/s, done.
Total 10 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/SKoloydenko/git-demo.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

Remote repositories

`git pull origin <branch>` – получение изменений с удаленного репозитория

```
$ git pull origin main
From https://github.com/SKoloydenko/git-demo
 * branch                main          -> FETCH_HEAD
Updating bab0a50..10ed038
Fast-forward
 index.html | 2 ++
 1 file changed, 2 insertions(+)
```

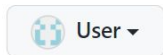
Pull request

Pull request – предложение создателю репозитория внести в него ваши правки.

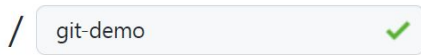
Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Owner *



Repository name *



By default, forks are named the same as their parent repository. You can customize the name to distinguish it further.

Description (optional)

 You are creating a fork in your personal account.

Create fork

Pull request

После внесения изменений можно создать pull request

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)

Create pull request

1 commit

1 file changed

1 contributor

Commits on Jul 13, 2022

Added header

SKoloydenko committed 1 minute ago

8499816

Showing 1 changed file with 1 addition and 0 deletions.

Split Unified

	1	index.html
4	4	<title>Git Demo!</title>
5	5	</head>
6	6	<body>
7	+	<header>Header</header>
7	8	<div>Hello, world!</div>
8	9	</body>
9	10	</html>

Pull request

Создатель репозитория
может принять ваш pull
request

Added header #1



Open

User wants to merge 1 commit into [SKoIoydenko:main](#) from [User:main](#)



Conversation 0



Commits 1



Checks 0



Files changed 1



User commented 4 minutes ago

First-time contributor



No description provided.



Added header

8499816

Add more commits by pushing to the **main** branch on **User/git-demo**.



Continuous integration has not been set up

[GitHub Actions](#) and [several other apps](#) can be used to automatically catch bugs and enforce style.



This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request



You can also [open this in GitHub Desktop](#) or view [command line instructions](#).



Write

Preview

H

B

I

≡

<>

🔗

≡

☑

@

📎

↶

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.



Close pull request

Comment

Reference

- <https://git-scm.com>
- <https://learngitbranching.js.org>
- <https://dev.to/lydiahallie/cs-visualized-useful-git-commands-37p1>
- <https://www.youtube.com/watch?v=USjZcfj8yxE>
- <https://www.youtube.com/watch?v=nhNq2klvi9s>

Homework

- Основное дз будет скинуто в отдельном файле
- Опциональное дз:
 - Создайте собственный репозиторий с Hello World
 - Попрактикуйтесь в переключении между коммитами
 - Изучите концепции GitFlow и GitHub flow
 - Попрактикуйтесь в работе с ветками в git и переключении между ними
 - Попробуйте слить (merge) две ветки (напр. ветку feature и master)
 - Залейте определенную ветку на GitHub. Создайте новый репозиторий в другом месте на вашем диске и подгрузите эту ветку с GitHub.