

Системи за управление на версиите

Git и GitHub

Съдържание

- Системи за управление на версиите
- Разлики между централизирана и децентрализирана система за управление на версиите
- Използване на Git и GitHub

Системи за управление на версиите

- Средство за проследяване и запис на промените по файл (или множество файлове) във времето
 - Можете да възстановите дадена версия във всеки един момент
 - Възможност за възстановяване на цял проект към даден предишен момент
 - Сравняване на промените във времето
 - Предоставя възможност много лица да работят по едни и същи файлове едновременно
 - Проследяване от кой са направени дадени промени и кога

Централизирана и децентрализирана система за управление на версиите

Централизирана система за управление на версиите

- Базирана на идеята за едно единствено “централно” копие на проекта, което се намира някъде (на сървър)
- Направените промени се изпращат до този централен сървър
- Промените се теглят отново от централния сървър, като при изтегляне се обновяват файловете, които потребителят има локално на машината си

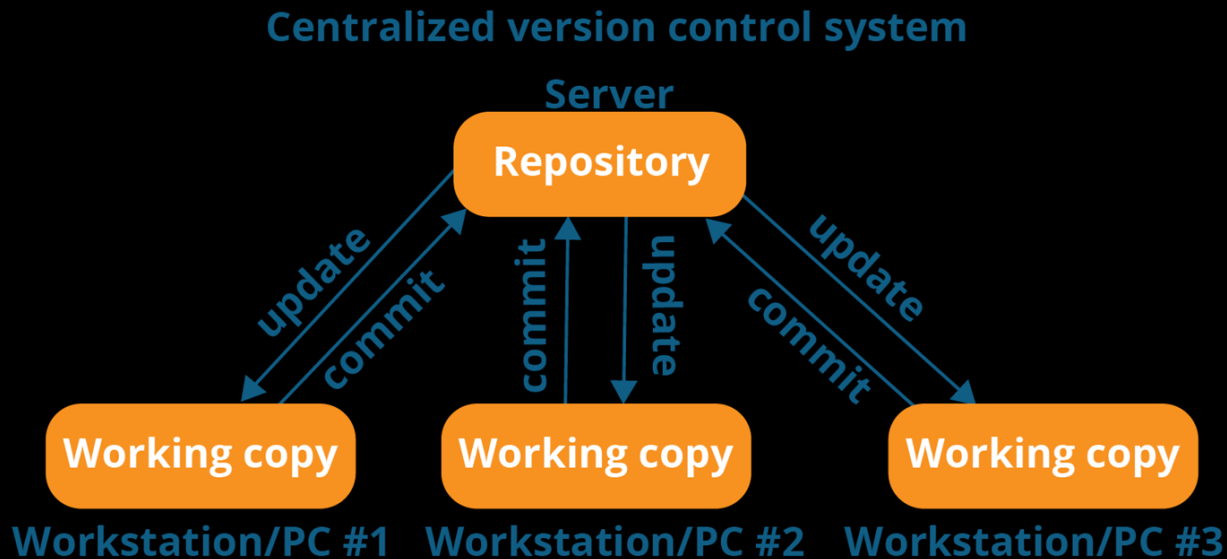
Централизирана и децентрализирана система за управление на версиите

Централизирана система за управление на версиите - механизъм на работа

1. Свлячане последната версия на проекта от централния сървър
2. Променяне файлове на проекта
3. Изпращане (commit) на промените към централния сървър, от където другите хора, работещи по проекта, могат да ги видят

Централизирана и децентрализирана система за управление на версиите

Централизирана система за управление на версиите



Централизирана и децентрализирана система за управление на версиите

Централизирана система за управление на версиите

- Решава въпроса със запазването на всяка нова версия на проекта локално
- При проблеми с централния сървър, ако не са направени копия, проектът е изгубен

Централизирана и децентрализирана система за управление на версиите

Децентрализирана система за управление на версиите

- Всеки разработчик “клонира” копие на репозитори и има пълна история на проекта локално на своята машина
- В програмирането повечето файлове съдържат просто текст, рядко снимки - затова не е много осезаема загубата на памет при този метод
 - Модерните системи също компресират файловете, за да спестят място

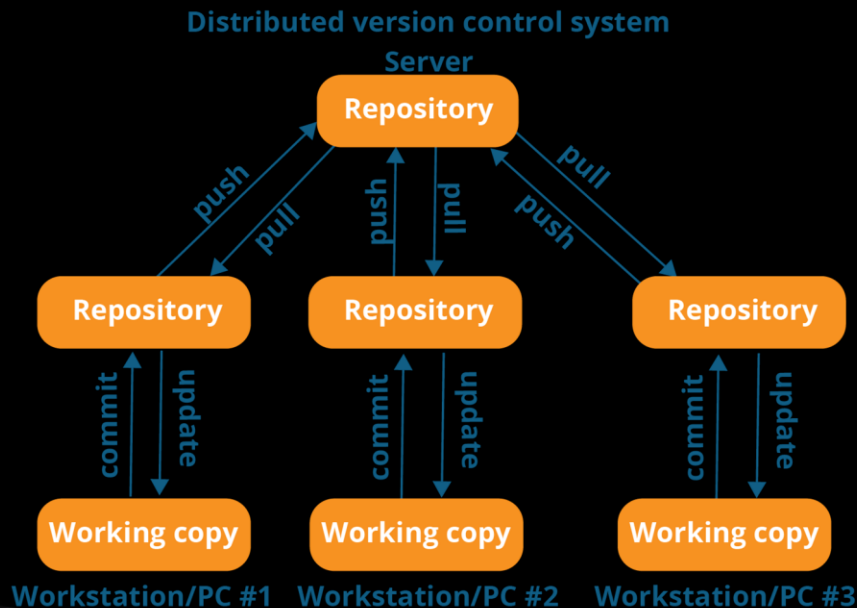
Централизирана и децентрализирана система за управление на версиите

Децентрализирана система за управление на версиите

- Метода за обновяване на променени файлове се нарича “pulling”
- Метода за изпращане направените от нас промени се нарича “pushing”

Централизирана и децентрализирана система за управление на версиите

Децентрализирана система за управление на версиите



Централизирана и децентрализирана система за управление на версиите

Децентрализирана сорс-контрол система - предимства пред централизирания вид системи

- Всички операции освен push и pull се извършват изключително бързо, понеже е нужен достъп само до твърдият диск, а не до сървър
- Промените, могат да се събират и изпращат до локално репозитори без никой да ги види, преди да се push-нат наведнъж
- Всичко освен push и pull може да се прави без наличието на интернет връзка
 - Може да работите дори от самолет

Централизирана и децентрализирана система за управление на версиите

Децентрализирана система за управление на версиите - недостатъци пред централизирания вид системи

- Ако проектът съдържа много големи файлове, които не могат да се компресират лесно, запазването на всички версии ще струва много памет
- Ако проектът има дълга история (например над 50,000 промени), тегленето на тази история ще отнеме прекалено много време

Git и GitHub

Git

- Децентрализирана система за управление на версиите
- Използва snapshot-и вместо разлики
 - Всеки път при commit на промени Git прави нов snapshot на това как изглежда проекта в този момент, като включва всички файлове
 - Ако даден файл не е променен Git е достатъчно умен да не го записва наново, а да предостави референция към последния snapshot, в който този файл е претърпял промени
- Почти всичко при Git се случва локално

Git и GitHub

Git

- Има интегритет
 - Всичко се проследява с чексуми преди да се запише
 - Обръщенията отново стават чрез чексумите
 - Това означава, че е невъзможно да се промени съдържанието на файл или директория без Git да знае за това
 - Механизмът, използван за чек-сумите е SHA1 хеш - 40 символен символен низ подобен на този

24b9da6552252987aa493b52f8696cd6d3b00373

Git и GitHub

Git

- Три състояния на файл
 - Committed - данните в този файл са запазени в локалната база данни на този проект
 - Modified - съдържанието на този файл е било променено
 - Staged - промените са маркирани за бъдещ commit

Git и GitHub

Git

- Три основни части на Git
 - Работна директория - единично копие наричано checkout - версия на проекта от източника (origin)
 - Промените от последния checkout насам не са добавени в индекса за commit
 - Зона за постановка (staging area)/Индекс - намира е между работната директория и .git директорията
 - Подготвя промяна или редица промени, които искате да commit-нете

Git и GitHub

Git

- Три основни части на Git
 - Зона за постановка (staging area)/Индекс
 - При операция commit, се изпращат само промените, които са в индекса
 - Не commit-нати промени остават в работната директория
 - Файловете в индекса са променени и маркирани за следващия commit snapshot

Git и GitHub

Git

- Три основни части на Git
 - .git директория (Репозитори)
 - Източника на данни за проекта
 - Това, което потребителят pull-ва от сървъра
 - Файловете в репозиторито са commit-нати в проекта като snapshot версия

Git и GitHub

Git - полезни команди за конзолата/терминал

- `pwd` - принтира текущата работна директория
- `cd` - променя директорията
- `ls` - списък на файловете в директорията (`dir` за Windows)
- `touch` - създава нов празен файл (`copy con` за Windows)
- `mkdir` - създава нова празна папка

Git и GitHub

Git - графичен потребителски интерфейс

- Препоръчва се избягването на графичен потребителски интерфейс за Git за потребители без опит със системи за управление на версиите
- GitHub Desktop
- SourceTree
- SmartGit
- и др.

Git и GitHub

Git - установка

- Windows
 - <https://git-scm.com/download/win>
- Mac
 - <https://git-scm.com/download/mac>

```
$ brew install git
```

- Linux

```
$ sudo apt-get install git  
$ sudo yum install git
```

Git и GitHub

Git - конфигурация

- Валидация за успешна инсталация

```
$ git --version
```

- Задаване на потребителско име и имейл адрес

```
$ git config --global user.name "My Name"
```

```
$ git config --global user.email "my@mail.eu"
```

Git и GitHub

Git - конфигурация

- Списък с пълната конфигурация за Git

```
$ git config --list
```

- Документация - <https://git-scm.com/doc>

```
$ git help
```

Git и GitHub


Git - инициализация на Git репозитори

- В случай на вече съществуващ проект - първо идете в директорията на проекта

```
$ git init
```


Git и GitHub

Git - инициализация на Git репозитори

 MINGW64:/c/Users/danai/Desktop/OurFirstRepo

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo
```

```
$ git init
```

```
Initialized empty Git repository in C:/Users/danai/Desktop/OurFirstRepo/.git/
```

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)
```

```
$ |
```

Git и GitHub

GitHub

- Предлага хостинг за сорс-контрол на проекти
- Базиран на Git
- Създаване на акаунт

Git и GitHub

GitHub - push на локално репозитори в GitHub

- Създаване на README файл
- Добавяне на файла като част от проекта
- Добавяне на проектите в commit и задаване на съобщение за този commit

```
$ echo "Our first GitHub repository" > README.md  
$ git add .  
$ git commit -m "First commit"
```

Git и GitHub

GitHub - push на локално репозитори в GitHub

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)
$ git commit -m "First commit"
[master (root-commit) 5d3053b] First commit
1 file changed, 1 insertion(+)
create mode 100644 README.md
```

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)
$ |
```

Git и GitHub

GitHub - push на локално репозитори в GitHub



- Създаване на репозитори в GitHub

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner

Repository name *

 / OurFirstRepo 

Great repository names are short and memorable. Need inspiration? How about [cuddly-enigma?](#)

Description (optional)

☐ Public

☒ Private

Anyone can see this repository. You choose who can commit.


You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer.

Add .gitignore: None ▼





Add a license: None ▼ 

Create repository

Git и GitHub

GitHub - push на локално репозитори в GitHub


Quick setup — if you've done this kind of thing before

 Set up in Desktop or  HTTPS  SSH `https://github.com/` `OurFirstRepo.git` 

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).


...or create a new repository on the command line

```
echo "# OurFirstRepo" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/ /OurFirstRepo.git
git push -u origin master
```



...or push an existing repository from the command line

```
git remote add origin https://github.com/ /OurFirstRepo.git
git push -u origin master
```



Git и GitHub

GitHub - push на локално репозитори в GitHub

- HTTPS - изисква име и парола
- SSH - изисква ключ и SSH парола, която трябва да се добави в GitHub акаунта

```
$ git remote add origin  
https://github.com/MyUsername/OurFirstRepo.git
```

```
$ git push -u origin master
```

Git и GitHub

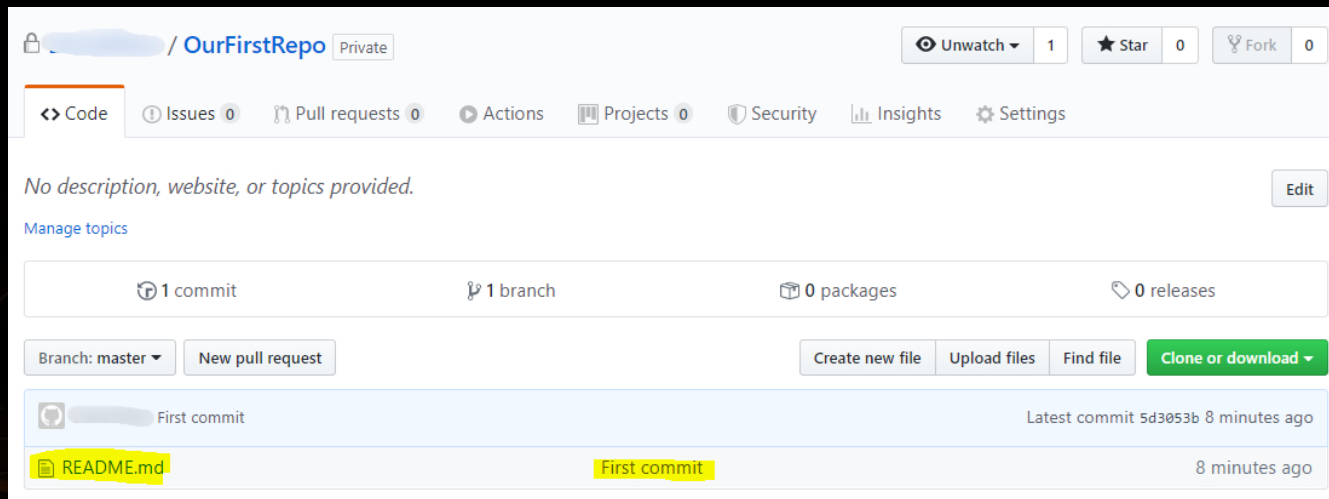
GitHub - push на локално репозитори в GitHub

```
MINGW64:/c:/Users/danai/Desktop/OurFirstRepo  
  
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)  
$ git remote add origin https://github.com/[REDACTED]/OurFirstRepo.git  
  
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)  
$ git push -u origin master  
Enumerating objects: 3, done.  
Counting objects: 100% (3/3), done.  
Writing objects: 100% (3/3), 243 bytes | 243.00 KiB/s, done.  
Total 3 (delta 0), reused 0 (delta 0)  
To https://github.com/[REDACTED]/OurFirstRepo.git  
 * [new branch]      master -> master  
Branch 'master' set up to track remote branch 'master' from 'origin'.  
  
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)  
$ |
```


Git и GitHub

GitHub - push на локално репозитори в GitHub

- Проверка дали всичко наред - промените са отразени в нашето GitHub репозитори



Git и GitHub

GitHub - най-често използвани команди

- `git status`
 - Извежда информация относно branch-а, в който се намираме
 - Дали branch-ът съдържа последната версия на файловете
 - Споменава, ако има нещо, което трябва да се commit-не

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

Git и GitHub

GitHub - най-често използвани команди

- `git add`
 - Обновява съществуващи файлове или добавя проследвяване на нови
 - `git add .`
 - Работи върху всички файлове
 - `git add <file_name>`
 - Работи върху файл с посоченото име

Git и GitHub

GitHub - най-често използвани команди

Нека добавим нов файл в папката на нашето локално репозитори и обновим съдържанието на вече съществуващият файл README.md - можете да направите това с любимия си текстов редактор.

След това ще проверим статуса на локалното ни репозитори преди и след като добавим промените с `git add` командата

Git и GitHub

GitHub - най-често използвани команди

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)
```

```
$ git status
```

```
On branch master
```

```
Your branch is up to date with 'origin/master'.
```

```
Changes not staged for commit:
```

```
(use "git add <file>..." to update what will be committed)
```

```
(use "git restore <file>..." to discard changes in working directory)
```

```
modified: README.md
```

```
Untracked files:
```

```
(use "git add <file>..." to include in what will be committed)
```

```
my-new-file.txt.txt
```

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

Git и GitHub

GitHub - най-често използвани команди

- Преди git add .

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)
```

```
$ git status
```

```
On branch master
```

```
Your branch is up to date with 'origin/master'.
```

```
Changes not staged for commit:
```

```
(use "git add <file>..." to update what will be committed)
```

```
(use "git restore <file>..." to discard changes in working directory)
```

```
modified: README.md
```

```
Untracked files:
```

```
(use "git add <file>..." to include in what will be committed)
```

```
my-new-file.txt.txt
```

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

Git и GitHub

GitHub - най-често използвани команди

- След `git add .` - забележете как състоянието на файловете се променя

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
       modified:   README.md
       new file:   my-new-file.txt.txt
```

Git и GitHub

GitHub - най-често използвани команди

- `git diff` - отговаря на 2 въпроса
 - Какви промени са в Staged състояние и са готови за `commit`?
 - Какви промени са направени, но не са в Staged състояние?
- `git diff --staged`
 1. Сравнява версиите на файловете
 2. Метаданни
 3. Маркери за промени за файл A/B
 4. Chunk Header
 5. Chunk промени

Git и GitHub

GitHub - най-често използвани команди

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)
$ git diff --staged
diff --git a/README.md b/README.md
index 2c77258..a1b83bc 100644
--- a/README.md
+++ b/README.md
@@ -1,2 @@
  Our first GitHub repository
+Edit 1.0
\ No newline at end of file
diff --git a/my-new-file.txt.txt b/my-new-file.txt.txt
new file mode 100644
index 0000000..e69de29
```

Git и GitHub

GitHub - най-често използвани команди

- `git commit`
 - Прави `commit` съдържащ в себе си променените файлове в даден момент от времето
 - `git commit -a`
 - Пропуска Staging фазата (изпуска `git add` командата) и директно прави `commit`
 - `git commit -a -m "Commit message"`
 - Флага `-m` задава съобщение за бъдещия `commit`

Git и GitHub

GitHub - най-често използвани команди

- Тъй като вече сме добавили промените ще изпуснем -а флага

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)
$ git commit -m "New file added; Changes to README introduced"
[master 52cf2f6] New file added; Changes to README introduced
2 files changed, 1 insertion(+)
create mode 100644 my-new-file.txt.txt
```

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

Git и GitHub

GitHub - най-често използвани команди

- `git push / git push origin master`
 - изпраща направените промени до източника (сървър)

```
MINGW64:/c:/Users/danai/Desktop/OurFirstRepo

danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)
$ git push origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 360 bytes | 180.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/[REDACTED]/OurFirstRepo.git
5d3053b..52cf2f6  master -> master
```

Git и GitHub

GitHub - най-често използвани команди

- `git log`
 - Връща историята на commit-ите
- `git log -1`
 - Връща последния commit
- `git log --oneline`
 - Връща всички commit-и форматирани на един ред
- `git log --patch`
 - Връща всеки commit детайлно + `git diff` за всеки от тях

Git и GitHub

GitHub - най-често използвани команди

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)
$ git log
commit 52cf2f6ef9d244be8e5a3a2e5fa80b07dc5b6f2f (HEAD -> master, origin/master)
Author: [REDACTED]
Date:   Wed Nov 20 17:51:19 2019 +0200
```

New file added; Changes to README introduced

```
commit 5d3053b402e5e41a762ea11a5f1340e84d648800
Author: [REDACTED]
Date:   Wed Nov 20 17:05:43 2019 +0200
```

First commit

Git и GitHub

GitHub - най-често използвани команди

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)
$ git log --oneline
52cf2f6 (HEAD -> master, origin/master) New file added; Changes to README introduced
5d3053b First commit
```

Git и GitHub

GitHub - най-често използвани команди

- `git rm <file_name>`
 - Премахва файл от проекта
- `git rm --cached <file_name>`
 - Премахва файл от проекта (спира да следи за промени), но не и от директорията

Git и GitHub

GitHub - най-често използвани команди

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)
$ ls
file-to-remove.txt.txt  my-new-file.txt.txt  README.md
```

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)
$ git rm file-to-remove.txt.txt
rm 'file-to-remove.txt.txt'
```

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.
```

```
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    file-to-remove.txt.txt
```

Git и GitHub

GitHub - най-често използвани команди

- `git mv <file_name> <new_file_name>`
 - Преименува файл

Git и GitHub

GitHub - първи стъпки с branch-ове

<http://git-school.github.io/visualizing-git/>

- Какво е branch (клон)?
 - Разклонение на проекта
 - Позволява работа по проекта без да се променя директно официалната му версия (master branch)
 - Полезно за тестване на нова функционалност
 - Практика е в master branch винаги да се държи работеща версия на проекта

Git и GitHub

GitHub - първи стъпки с branch-ове

- Създаване на нов клон
 - `git checkout -b <branch_name>`
- Преминаване между клоновете
 - `git checkout <branch_name>`
- Създаване на нов клон без да се преминава на него
 - `git branch <branch_name>`

Git и GitHub

GitHub - първи стъпки с branch-ове

Ще използваме инструмента за визуализация, за да онагледим как изглежда репозиторието ни, когато създадем други клонове.

```
$ git commit  
$ git commit  
$ git checkout -b new_branch  
$ git commit  
$ git commit
```

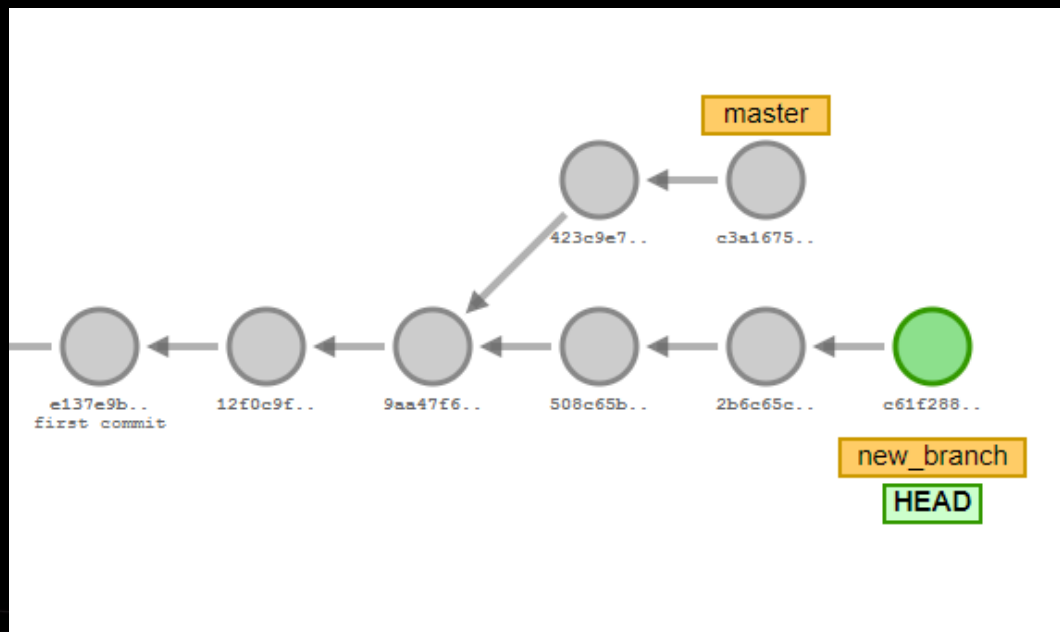
Git и GitHub

GitHub - първи стъпки с branch-ове

```
$ git checkout master  
$ git commit  
$ git commit  
$ git checkout new_branch  
$ git commit
```

Git и GitHub

GitHub - първи стъпки с branch-ове



Git и GitHub

GitHub - първи стъпки с branch-ове

- `git stash`
 - Позволява да се “избутат” настрана текущите промени, за да се върнем към чиста работната директория
 - Полезно в случаите, в които искаме да сменим клона и да работим по друг аспект, но работта на текущия клон се е объркала (не искаме да я commit-ваме)
- `git stash list`
 - Изкарва списък на stash-натите промени
- `git stash pop`
 - Връща промените обратно в работната директория

Git и GitHub

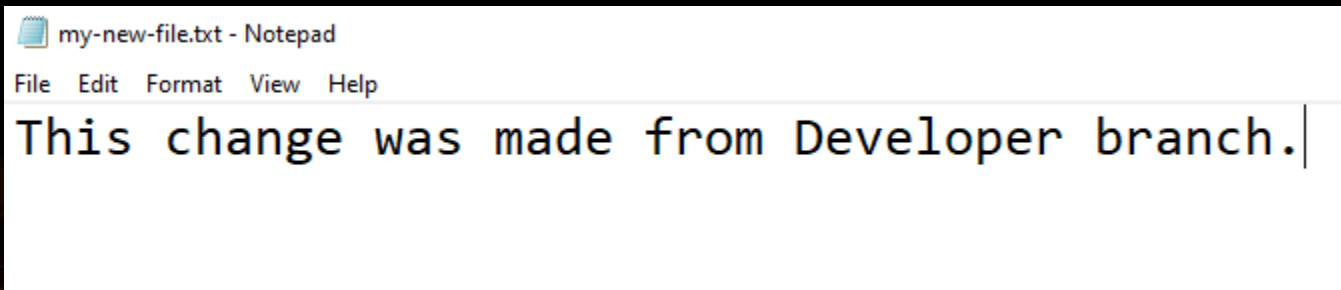
GitHub - първи стъпки с branch-ове

- `git merge <branch_name>`
 - Обединява няколко клона

Git и GitHub

GitHub - първи стъпки с branch-ове

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)
$ git checkout -b Developer
Switched to a new branch 'Developer'
```



Git и GitHub

GitHub - първи стъпки с branch-ове

- `git push --set-upstream origin <branch_name>`
 - Конфигурира как да се push-ват направените промени в новият
НИ КЛОН

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (Developer)
$ git push --set-upstream origin Developer
Total 0 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'Developer' on GitHub by visiting:
remote:   https://github.com/[redacted]/OurFirstRepo/pull/new/Developer
remote:
To https://github.com/[redacted]/OurFirstRepo.git
 * [new branch]      Developer -> Developer
Branch 'Developer' set up to track remote branch 'Developer' from 'origin'.
```

Git и GitHub

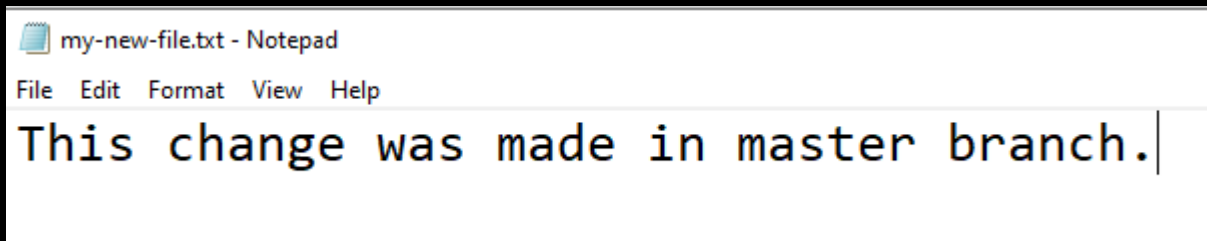
GitHub - първи стъпки с branch-ове

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (Developer)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 339 bytes | 169.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/[REDACTED]/OurFirstRepo.git
   aec9e93..9c29624  Developer -> Developer
```

Git и GitHub

GitHub - първи стъпки с branch-ове

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (Developer)  
$ git checkout master  
Switched to branch 'master'  
Your branch is up to date with 'origin/master'.
```



...commit и push на направените промени..

Git и GitHub

GitHub - първи стъпки с branch-ове

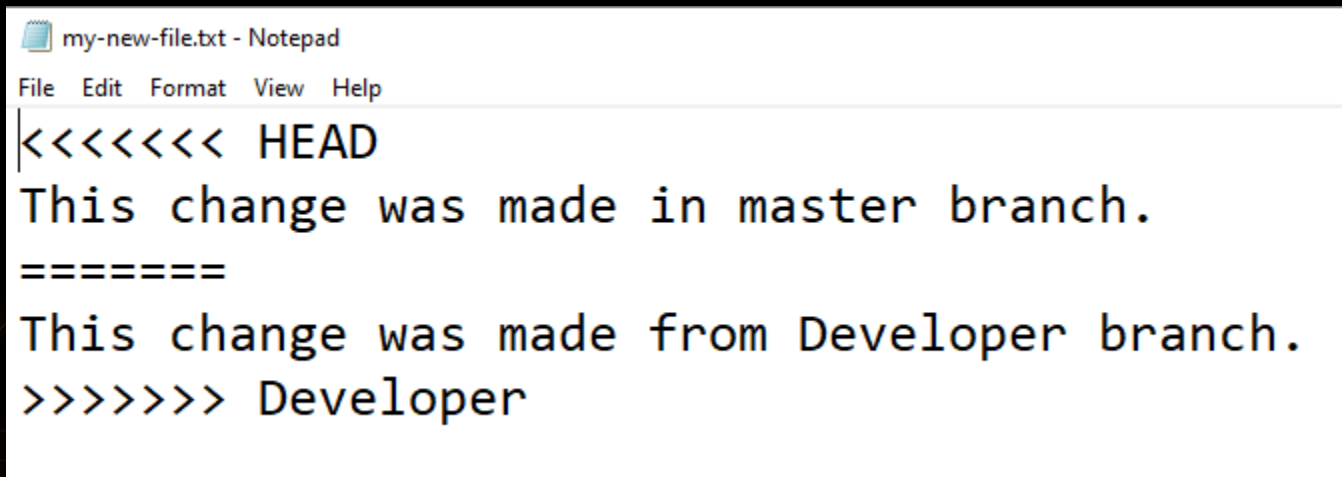
```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)
$ git merge Developer
Auto-merging my-new-file.txt.txt
CONFLICT (content): Merge conflict in my-new-file.txt.txt
Automatic merge failed; fix conflicts and then commit the result.
```

- В случаите, когато са направени промени по един и същи файл от различни клонове възникват конфликти
- Сливането на клоновете е невъзможно преди да се разрешат тези конфликти

Git и GitHub

GitHub - първи стъпки с branch-ове

- Как изглежда файла след опита за сливане на клоновете



```
my-new-file.txt - Notepad
File Edit Format View Help
|<<<<<<< HEAD
This change was made in master branch.
=====
This change was made from Developer branch.
>>>>>>> Developer
```

Git и GitHub

GitHub - първи стъпки с branch-ове

- Статуса на клона след опита за сливане

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master|MERGING)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

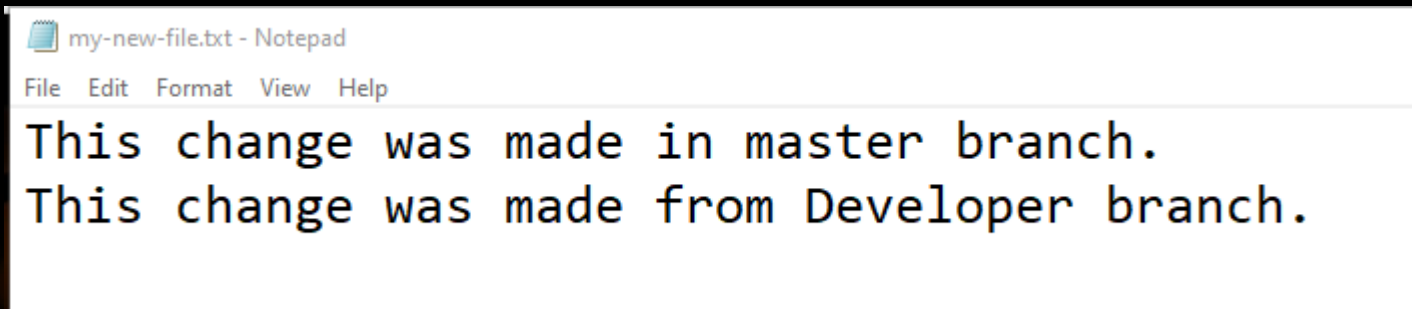
Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   my-new-file.txt.txt

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


Git и GitHub

GitHub - първи стъпки с branch-ове

- Разрешаване на конфликтите - могат да се разрешат и чрез уеб клиента на GitHub

A screenshot of a Notepad application window. The title bar reads "my-new-file.txt - Notepad". The menu bar includes "File", "Edit", "Format", "View", and "Help". The text area contains two lines of code in a monospaced font: "This change was made in master branch." and "This change was made from Developer branch."

```
my-new-file.txt - Notepad
File Edit Format View Help
This change was made in master branch.
This change was made from Developer branch.
```

Git и GitHub

GitHub - първи стъпки с branch-ове

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master|MERGING)
$ git add .
```

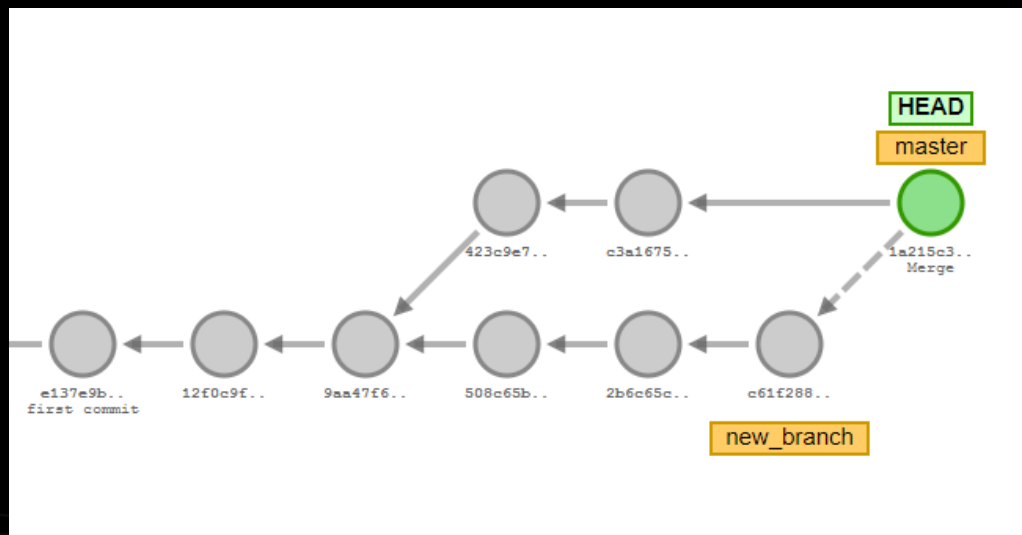
```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master|MERGING)
$ git commit -m "Merged Developer-master"
[master 5ff8d95] Merged Developer-master
```

```
danai@DESKTOP-2GF44TF MINGW64 ~/Desktop/OurFirstRepo (master)
$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 383 bytes | 383.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/[REDACTED]/OurFirstRepo.git
   355d245..5ff8d95  master -> master
```

Git и GitHub

GitHub - първи стъпки с branch-ове

- Как изглежда merge през инструмента за визуализация



Git и GitHub

GitHub - първи стъпки с branch-ове

- Преместване на commit-ите от историята обратно в работната директория или staging area
 - `git reset --soft <commit>`
 - В staging area
 - `git reset --mixed <commit>`
 - В работната директория
 - `git reset --hard <commit>`
 - Премества направените промени в коша

Git и GitHub

GitHub - първи стъпки с branch-ове

- Изтегляне нова версия на проекта (изтегляне на промените) от сървъра
 - `git clone <project_url>`
 - При неклонирано репозитори
 - `git pull`
 - При вече клонирано репозитори

Git и GitHub

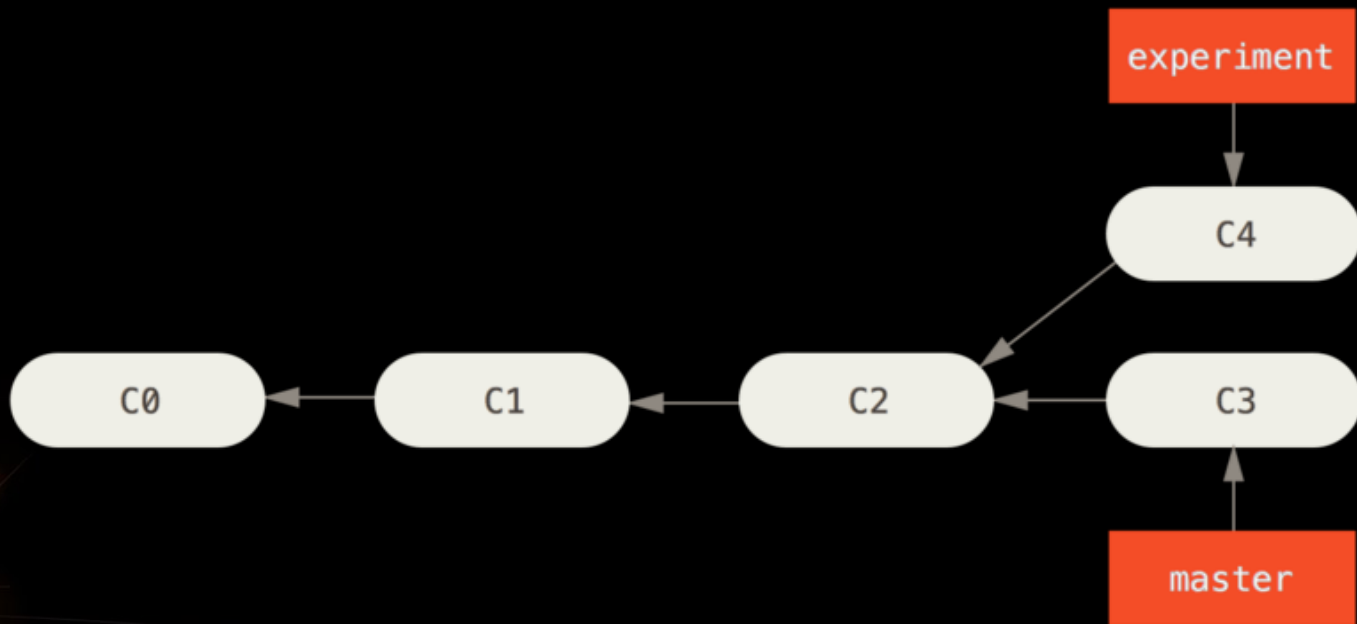
GitHub - rebase

<https://git-scm.com/book/en/v2/Git-Branching-Rebasing>

- Друг начин да се интегрират промени от един клон в друг
- В случая с merge двата клона experiment и master ще бъдат обединени двата последни snapshot-а C4 и C3, като заедно с последния наследник на двата (C2) ще бъде създаден нов snapshot (и commit)

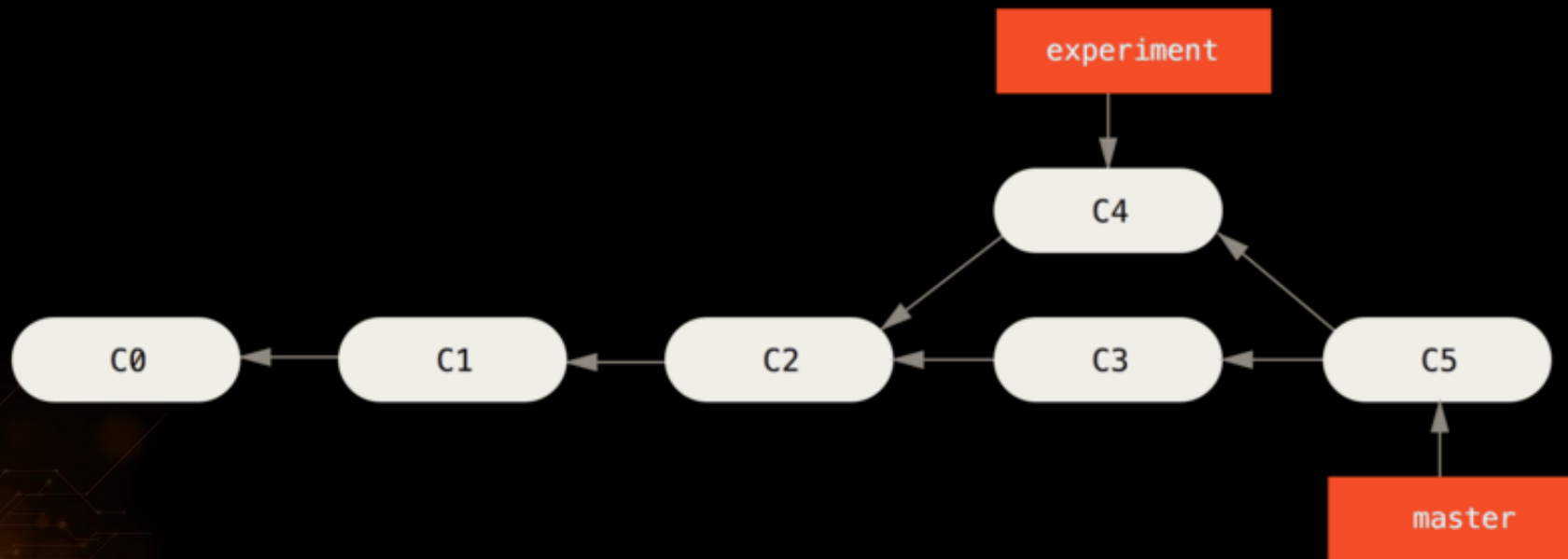
Git и GitHub

GitHub - rebase



Git и GitHub

GitHub - rebase



Git и GitHub

GitHub - rebase

- Ако вместо това се използва rebase - промените от C4 ще бъдат приложени върху C3
- В този пример бихте преминали на клона experiment и след това бихте го ребазирали върху master клона:

```
$ git checkout experiment  
$ git rebase master
```

Git и GitHub

GitHub - rebase

- Ако вместо това се използва rebase - промените от C4 ще бъдат приложени върху C3
- В този пример бихте преминали на клона experiment и след това бихте го ребазирали върху master клона:

```
$ git checkout experiment  
$ git rebase master
```

Git и GitHub

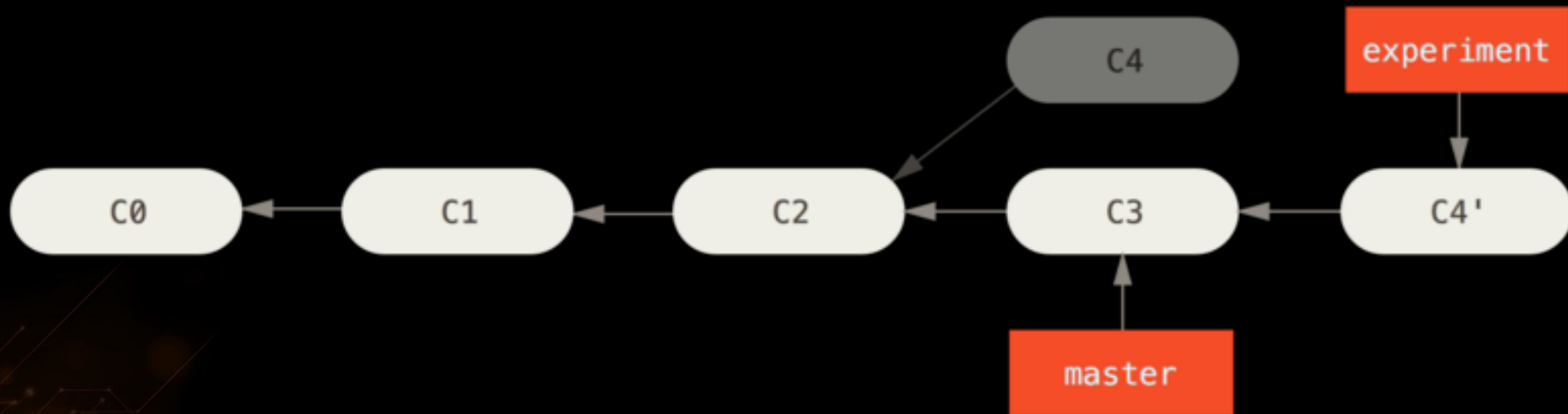
GitHub - rebase

- В този момент вече може да се върнете в master клона и да обедините клоновете като master ще “прескочи във времето” до experiment

```
$ git checkout master  
$ git merge experiment
```

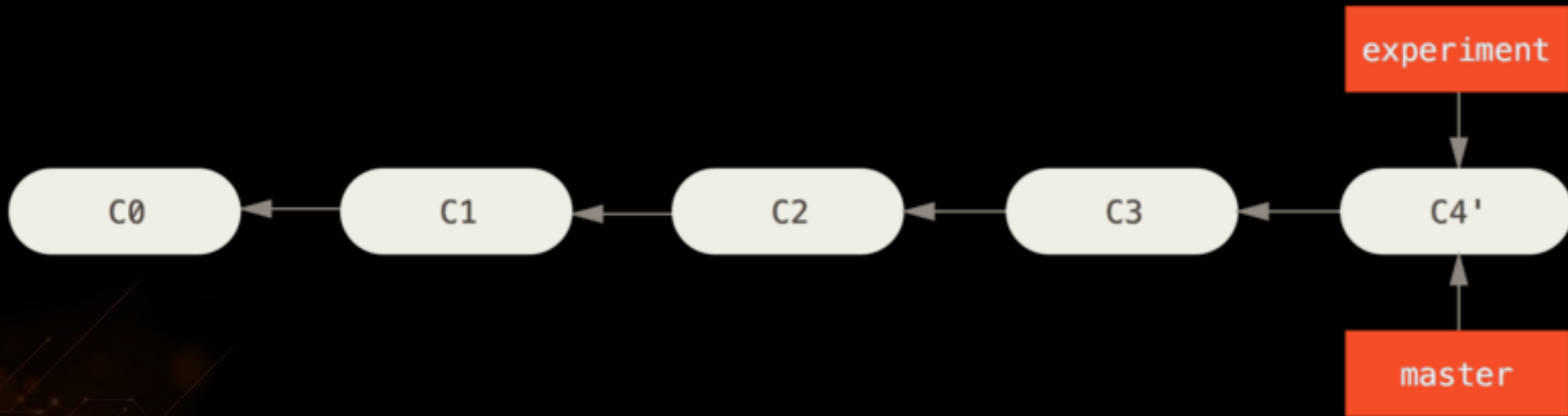
Git и GitHub

GitHub - rebase



Git и GitHub

GitHub - rebase



Обобщение

- Системи за управление на версиите
- Разлики между централизирана и децентрализирана система за управление на версиите
- Използване на Git и GitHub