

Непрерывная оптимизация

Занятие 2. GIT

Зачем нужны системы контроля версий?

- 1) Возврат к старым версиям
- 2) Отслеживание изменений
- 3) Хранений истории
- 4) Совместная работа

Как обычно выглядит репозиторий?

The screenshot shows a GitHub repository named 'nn_ml_practice' which is public. At the top, there are buttons for 'Unpin', 'Unwatch' (6), 'Fork' (28), and 'Star' (15). Below this, the repository is set to the 'spring2024' branch, showing 2 branches and 0 tags. A search bar 'Go to file' and buttons for 'Add file' and 'Code' are present. The main content area lists the repository's files and folders:

File/Folder	Commit Message	Time Ago
data	added 03 seminar	6 months ago
notebooks	added 04 lab	6 months ago
pics	init commit	7 months ago
presentations	added alter hw	6 months ago
.gitignore	init commit	7 months ago
README.md	init commit	7 months ago

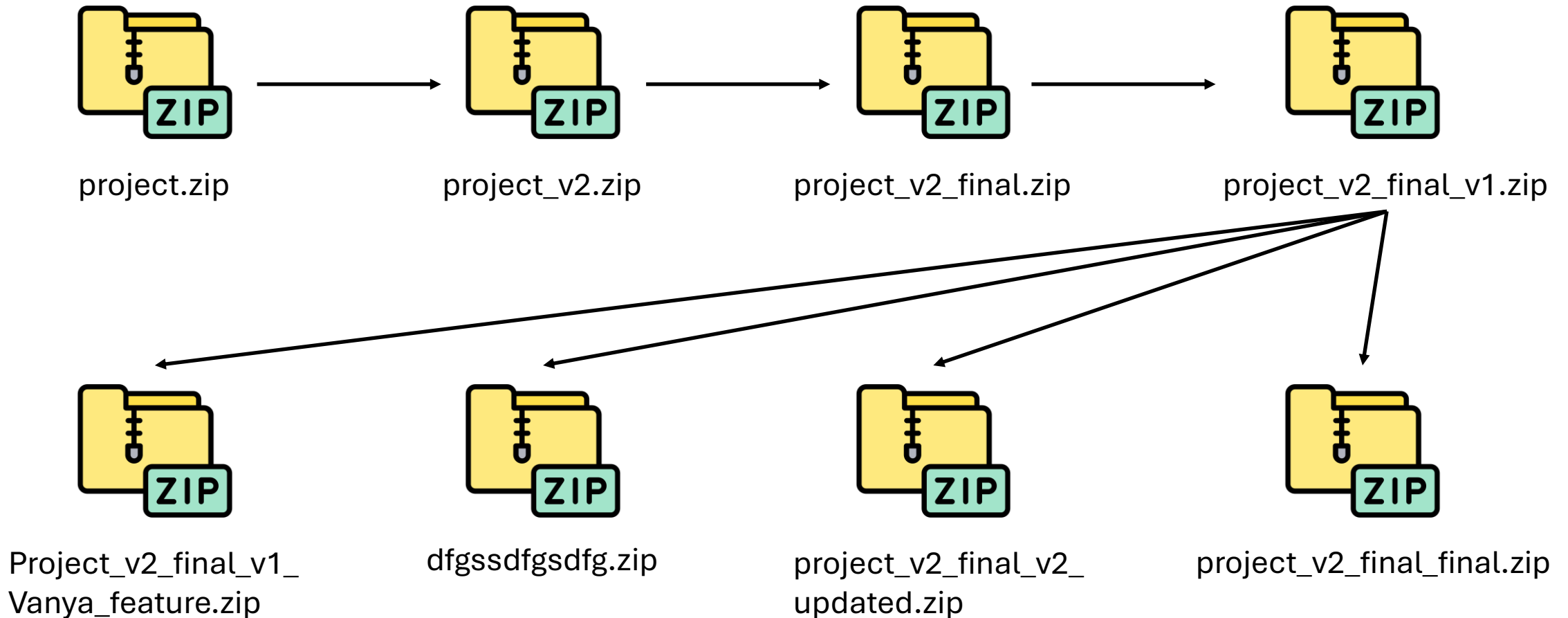
Below the file list is the 'README' section, which contains the following text:

Репозиторий курса "Искусственные нейронные сети и машинное обучение" для групп БИВТ-21


[Google таблица с вашими успехами](#)

On the right side of the repository page, there is an 'About' section stating: 'Repo for course "Neural Network and Machine Learning"'. It also lists 'Readme', 'Activity', '15 stars', '6 watching', and '28 forks'. Below this are sections for 'Releases' (no releases published, with a link to 'Create a new release') and 'Packages' (no packages published, with a link to 'Publish your first package'). At the bottom right, the 'Languages' section shows 'Jupyter Notebook' at 100.0%.





Если не гит, то что?




Версионирование файлов

 This repository Search

Explore Gist Blog Help

 mdo + -   

 github / github PRIVATE

Watch 32 Unstar 44 Fork 36

Diff view body class toggle #32247

Merged josh merged 6 commits into master from diff-view-body-class-toggle about 19 hours ago

Conversation 0 Commits 6 Files changed 4

+10 -1

Showing 4 changed files with 10 additions and 1 deletion.

Unified Split

5 app/assets/javascripts/github/pages/diffs/split.coffee

...	...	@@ -1,9 +1,12 @@
1	1	# Everything here is terrible
2	2	
3	3	syncBodyClass = ->
4		- enabled = \$('#files').is(':visible') and \$('.file-diff-split')[0]?
	4	+ enabled = \$('meta[name=diff-view]').prop('content') is 'split' and
	5	+ \$('.file-diff-split').is(':visible')
5	6	document.body.classList.toggle 'split-diff', enabled
6	7	
7	8	# resync body class
	9	+\$_.observe 'meta[name=diff-view]', add: syncBodyClass, remove: syncBodyClass
8	10	\$.observe '.file-diff-split', add: syncBodyClass, remove: syncBodyClass
9	11	\$.observe '.js-pull-request-tab.selected', add: syncBodyClass, remove: syncBodyClass
	12	+\$_.observe '.js-compare-tabs .tabnav-tab.selected', add: syncBodyClass, remove: syncBodyClass

View

2 app/views/commit/show.html.erb

		@@ -10,6 +10,8 @@
10	10	<% content_for :head do %>

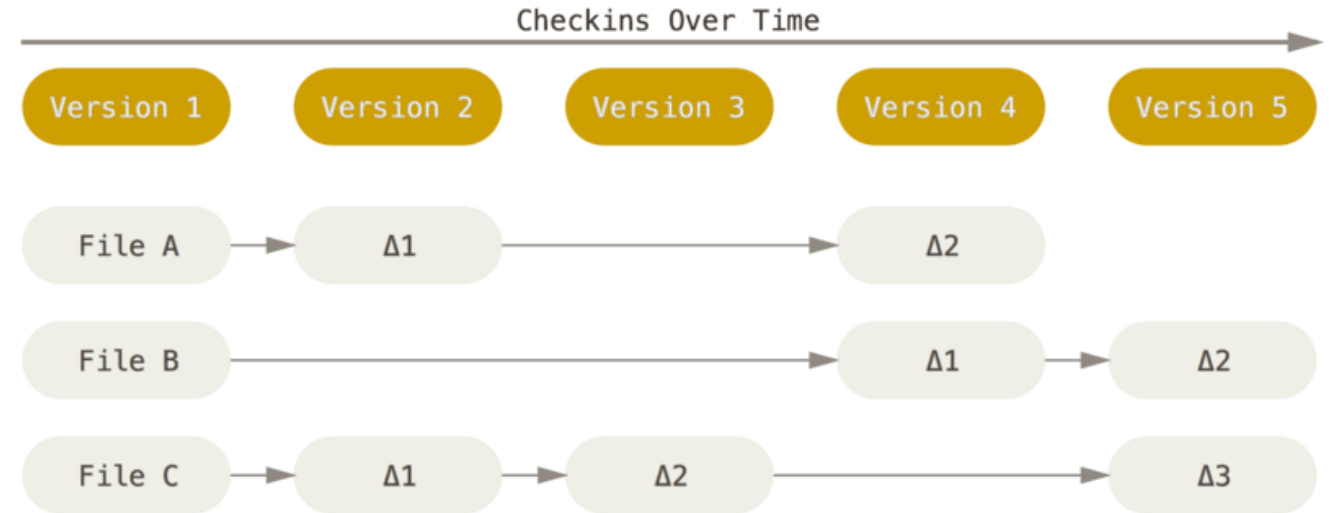
View

Как Git хранит файлы

Другие VCS:

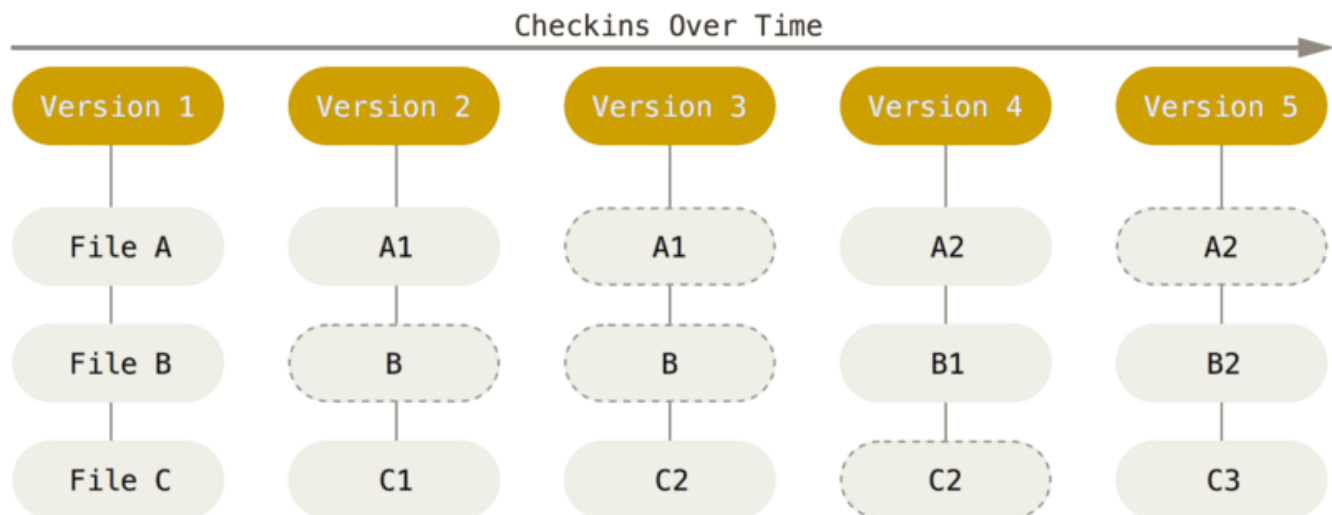
- Хранят информацию об изменениях в файле

$$File = File_{prev} + \Delta File$$



Git:

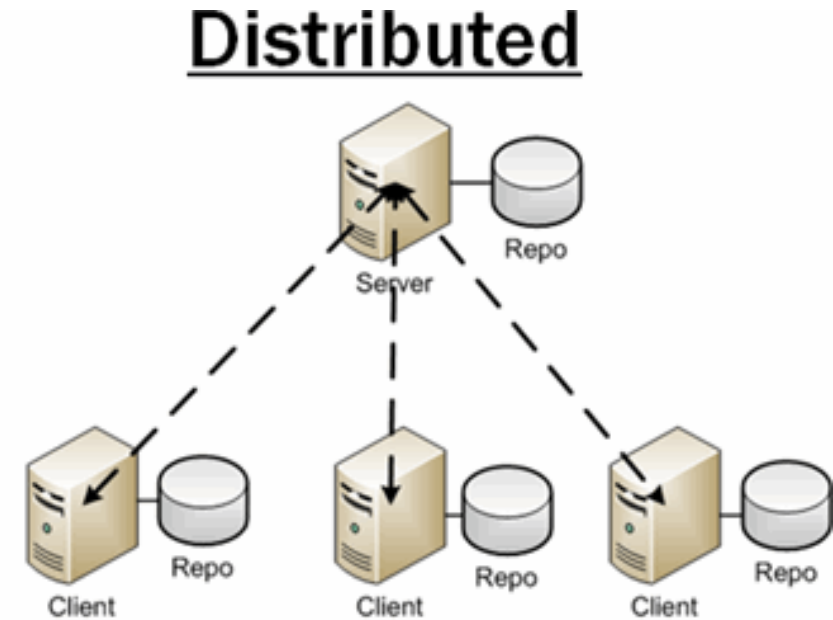
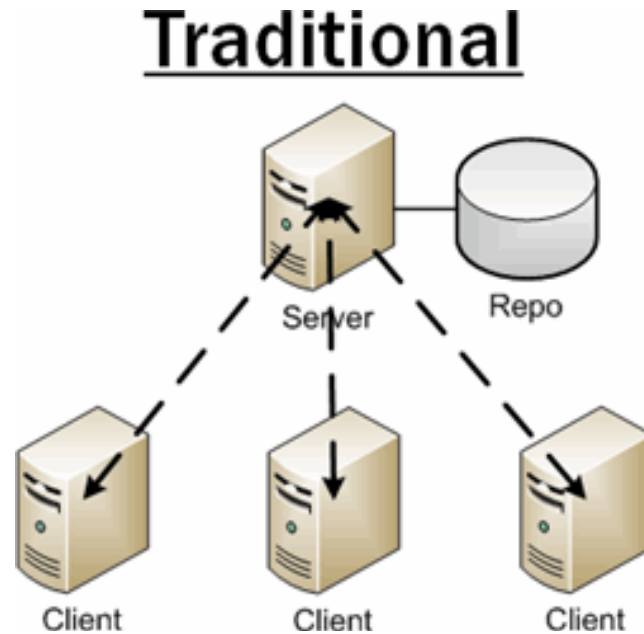
- Хранит снимоты файлов
- Если файл не поменялся, снимот делаться не будет



Git

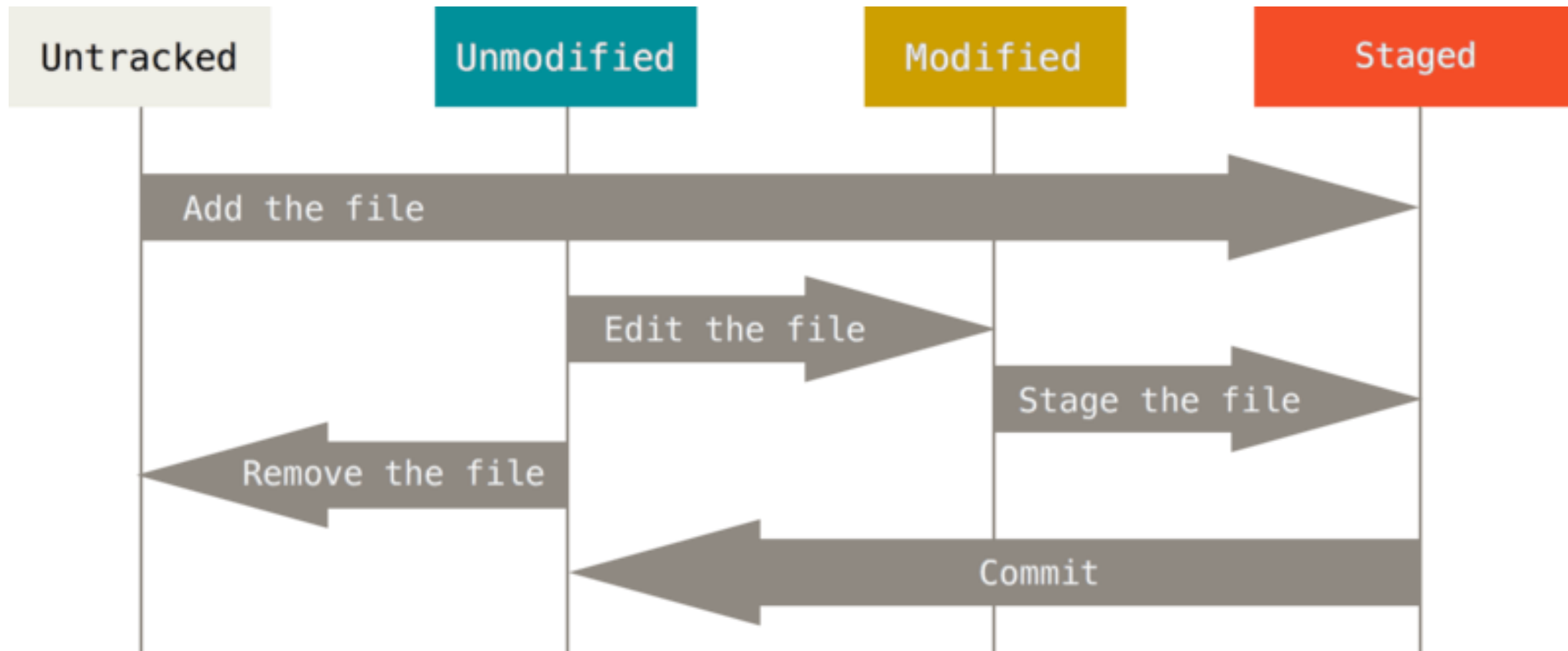
Реализован в:

- github.com
- gitlab.com
- bitbucket.com
- etc.



Traditional VS Distributed version
control system

Состояния файлов в Git



Практика

- 1) Создаем репозиторий
- 2) Добавляем файлы, делаем коммиты
- 3) Создаем еще одну ветку
- 4) Создаем пулл реквест в мастер
- 5) Создаем конфликт и решаем его

Инициализация репозитория

```
git init
```

Эта команда создаёт в текущем каталоге новый подкаталог с именем `.git`, содержащий все необходимые файлы репозитория — структуру Git репозитория. На этом этапе ваш проект ещё не находится под версионным контролем.

Подробное описание файлов, содержащихся в только что созданном вами каталоге `.git`, приведено в главе (<https://git-scm.com/book/ru/v2/Git-изнутри-Сантехника-и-Фарфор#ch10-git-internals>)

Создаем файл и добавляем его в git

Создаем sum_script.py

```
vim sum_script.py
```

Добавляем в гит с помощью
команды

```
git add sum_script.py
```

Узнать
репозитория

```
git status
```

состояние

Создаем коммит
изменениями с

```
git commit -m <MESSAGE>
```

Создаем удаленный репозиторий и пушим туда изменения

Обозначаем адрес для удаленного хранилища

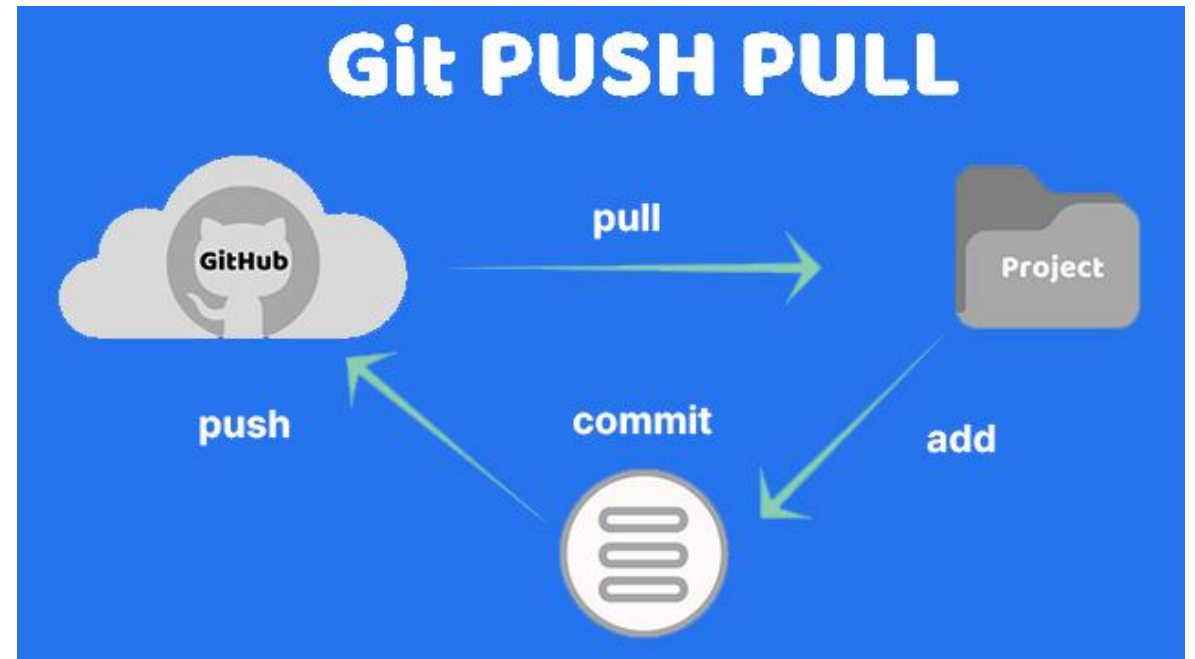
```
git remote add origin <URL>
```

Отправляем изменения на удаленный репозиторий

```
git push origin master
```

Подтягиваем изменения с удаленного репозитория

```
git pull
```



Еще команды

Откатиться в нужный коммит

```
git checkout <HASH>
```

Переименовать файл

```
git mv <FILE_FROM> <FILE_TO>
```

Удалить файл

```
git rm <FILENAME>
```

Удалить файл из индекса

```
git rm --cached <FILENAME>
```

Создаем ветки

Создать ветку

```
git branch <NAME>
```

Узнать, на какой ветке

```
git status
```

Переключиться на ветку

```
git checkout <NAME>
```

Создать + переключиться на ветку

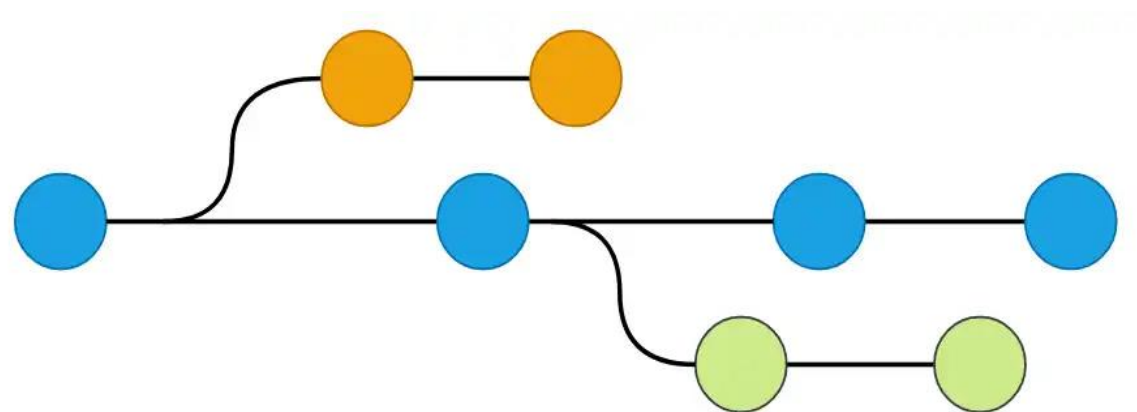
```
git checkout -b <BRANCH_NAME>
```

Переключиться на ветку

```
git switch <NAME>
```

Создать и переключиться на ветку

```
git switch -c <NAME>
```



Слияние веток

Влить изменения из ветки BRANCH в текущую

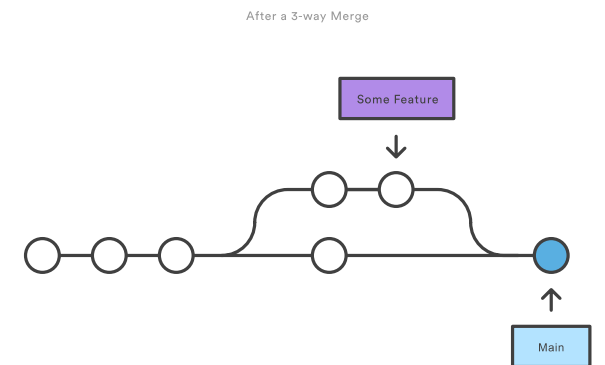
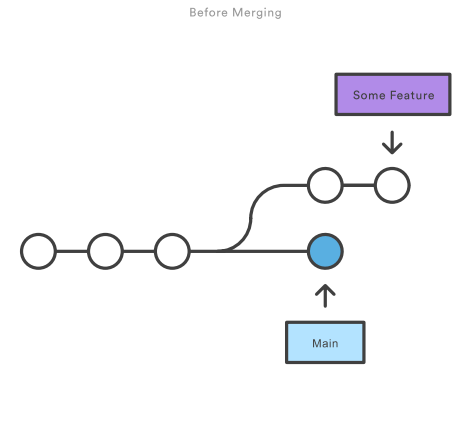
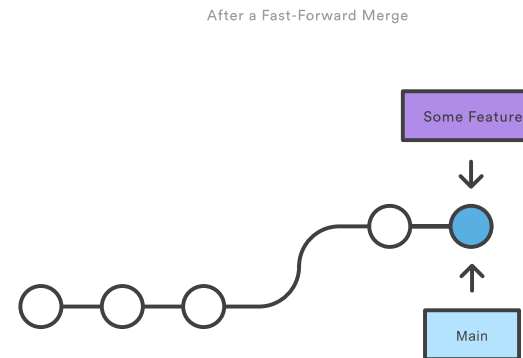
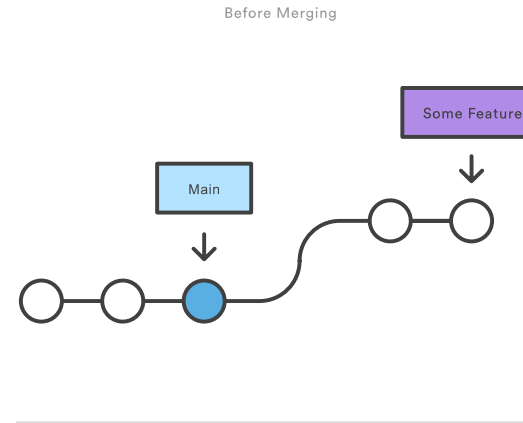
```
git merge <BRANCH>
```

Влить с помощью fast-forward (если BRANCH – продолжение текущей ветки)

```
git merge --ff <BRANCH>
```

Слияние через коммит слияния

```
git merge -no-ff <BRANCH>
```



Просмотр истории коммитов

Вся история коммитов

```
git log
```

Каждый коммит в 1 строку

```
git log --pretty=oneline
```

Показать историю с патчами

```
git log -p -2
```

В виде графа

```
git log --pretty=oneline --graph
```

Показать со статистикой

```
git log --stat
```

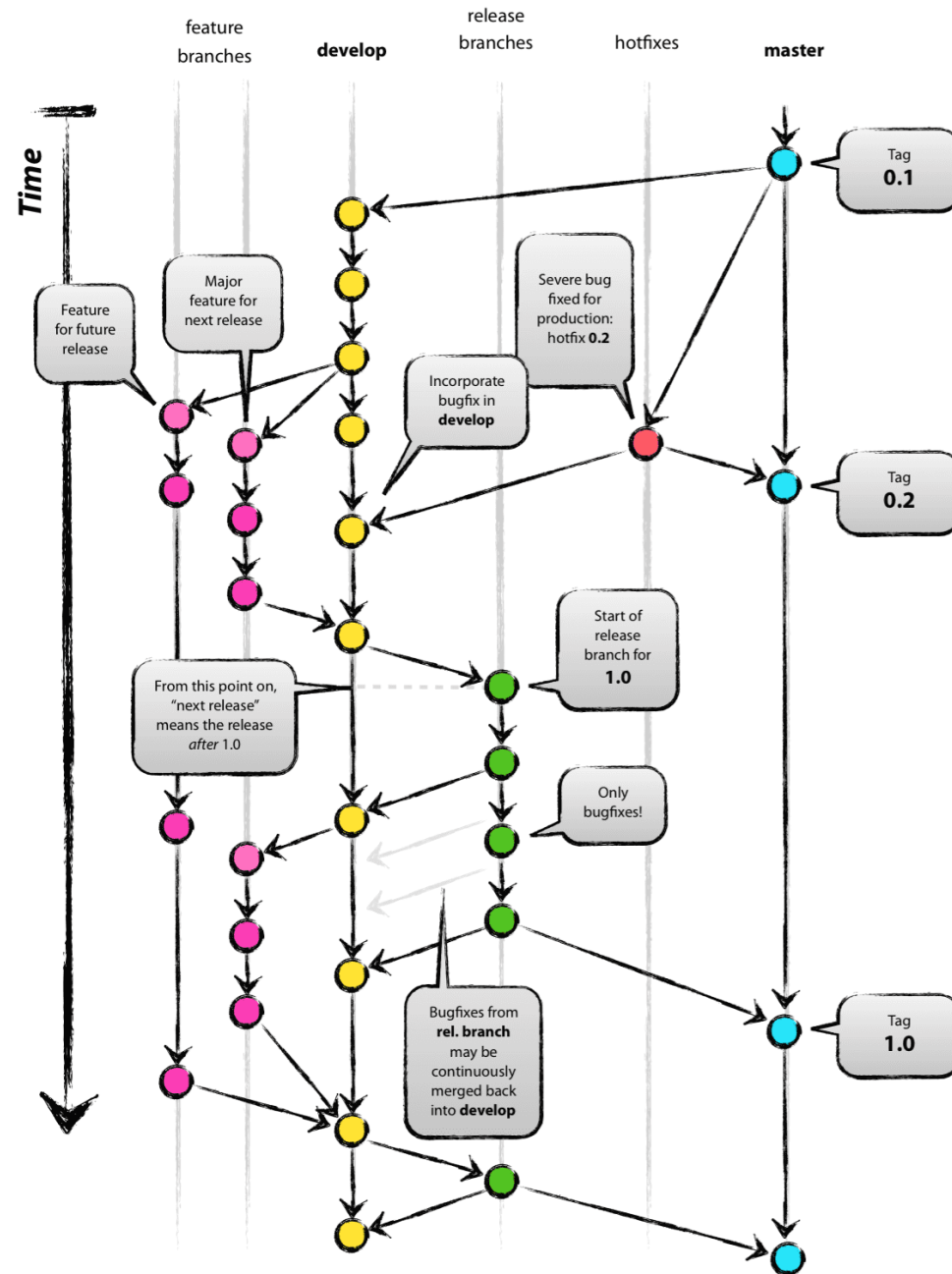
Показать коммиты, где менялся только интересующий файл

```
git log -- <PATH_TO_FILE>
```

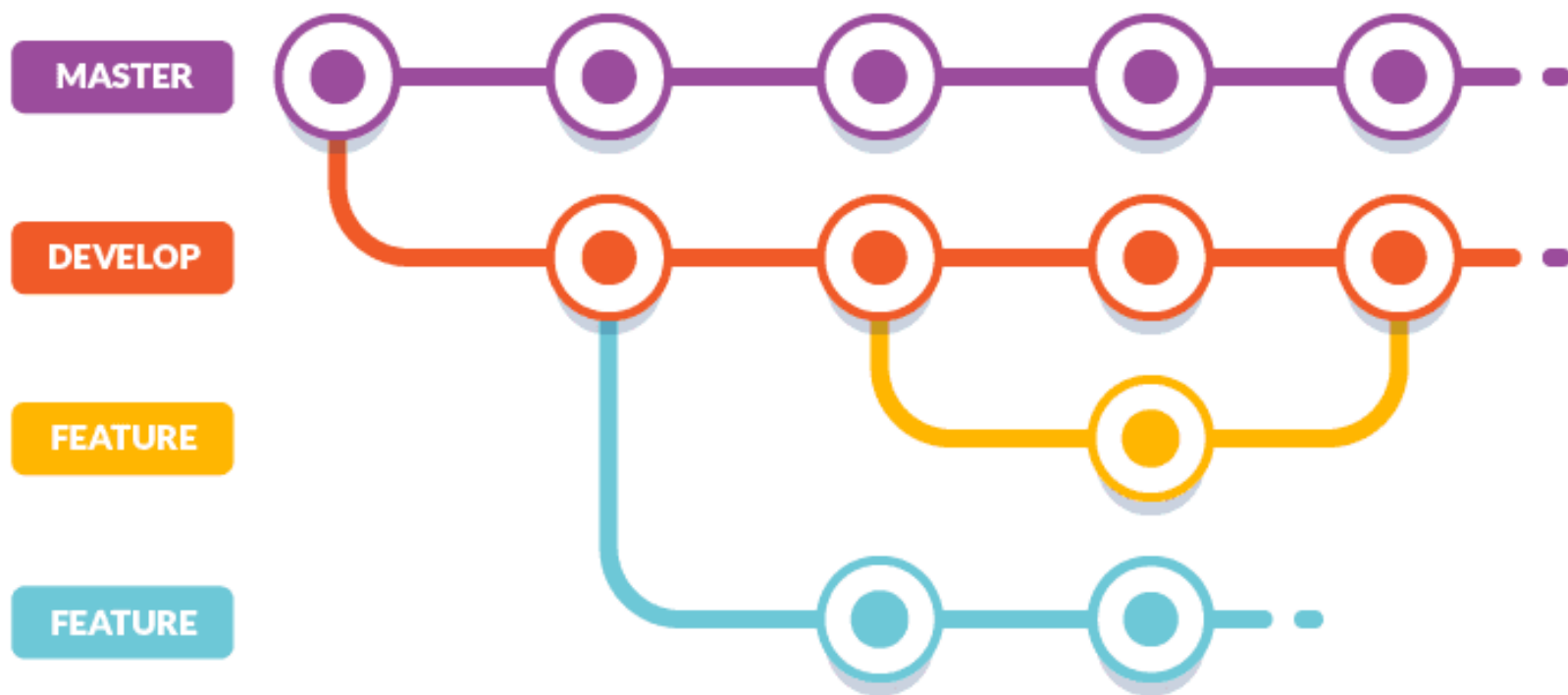

Поиграться с ветками гита онлайн

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

GitFlow



GitFlow поприятнее



Домашнее задание №1

- 1) Создать репозиторий в пространстве <https://github.com/misis-mlops2024> в виде <name-surname-group>, например alexey-myshlyanov-16
- 2) Создать локально пустую директорию, внутри которой создать README.md файл. Внутри написать заголовок «# Репозиторий курса MLOps». В строке ниже написать фамилию, имя и группу.
- 3) Создать файл main.py, внутри написать функцию для подсчета сумм двух чисел, используя main() функцию. Запустить в мастер.
- 4) Переписать функцию как сумму 3 чисел и запустить в мастер.
- 5) Перейти на ветку develop, изменить функцию как сумму N чисел (работает с любым количеством аргументов, посмотрите *args).
- 6) Сделать pull request из develop в мастер и решить существующий конфликт. Добавить меня (l3lush) в ревьюеры