

< Teach
Me
Skills />

Занятие 7.

GIT

Что такое GIT

GIT

Что такое git?

- система управления версиями с распределенной архитектурой
- он отслеживает и фиксирует изменения в файлах

Зачем?

- сохранение новых версий файлов
- возможность откатиться на старую версию файла ()
- совместная разработка
- возможность независимой разработки новых функций
- хранение кода проекта в облаке (на случай потери локальных данных)

Репозиторий

Зачем?

- хранилище кода и истории его изменений

Как создать?

- при помощи команды git init, при этом создается папка .git, где хранится все что связано с гитом

Где хранится код?

- локально
- GitHub
- Bitbucket
- GitLab

Коммит

Что это такое?

- точка сохранения проекта

Из чего состоит коммит?

- из уникального hash, который генерируется самостоятельно, а также из комментария

```
alena@alena-rog:~/techmeskills$ git commit -m "Add github instruction"
[main bb89e55] Add github instruction
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Например, на данном этапе у нас в проект добавилась инструкция по git

Коммит

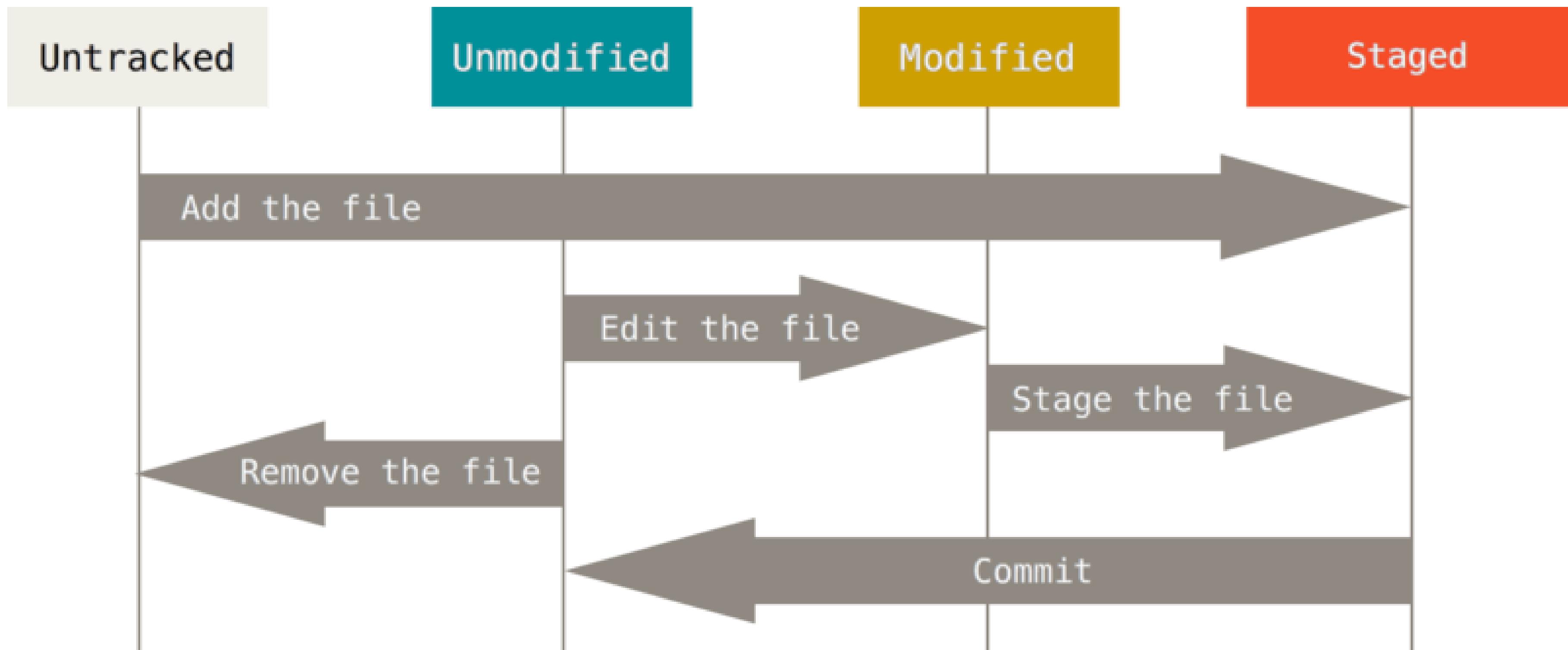
Как сделать? 2 этапа

- `git add <filename>` - добавляем нужные файлы
- `git commit -m "message"` - коммитим изменения

Что должно быть в коммите? Каким должен быть комментарий?

- Коммит должен содержать в себе небольшое количество изменений на одну тематику, если в комментарии хочется написать "сделано это И это", то это значит, что коммит следует разбить на несколько
- Комментарий должен быть осмысленным и отражать суть изменений
- Комментарии и содержание изменений должны быть в одном стиле (стиль кода, стиль написания комментариев в коммиту)

Состояния файлов

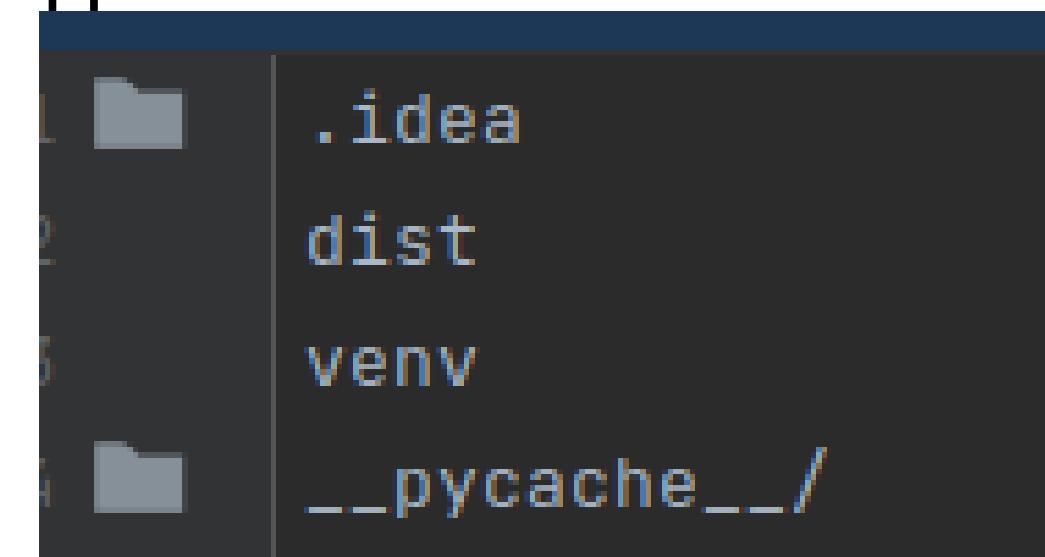


Неотслеживаемые файлы

- те, которые мы не добавили при помощи `git add <filename>` после изменения
- они будут подсвечиваться **красным** цветом
- можно из намеренно игнорировать, потому что в репозитории нет смысла их отслеживать, либо это является плохой практикой

.gitignore

- сюда мы пишем файлы или папки, которые хотим игнорировать, то есть не отслеживать их состояние НИКОГДА, будут подсвечиваться **желтым**
- сюда идут `.idea/`, `venv`, `__pycache__/`, `.env` и тд

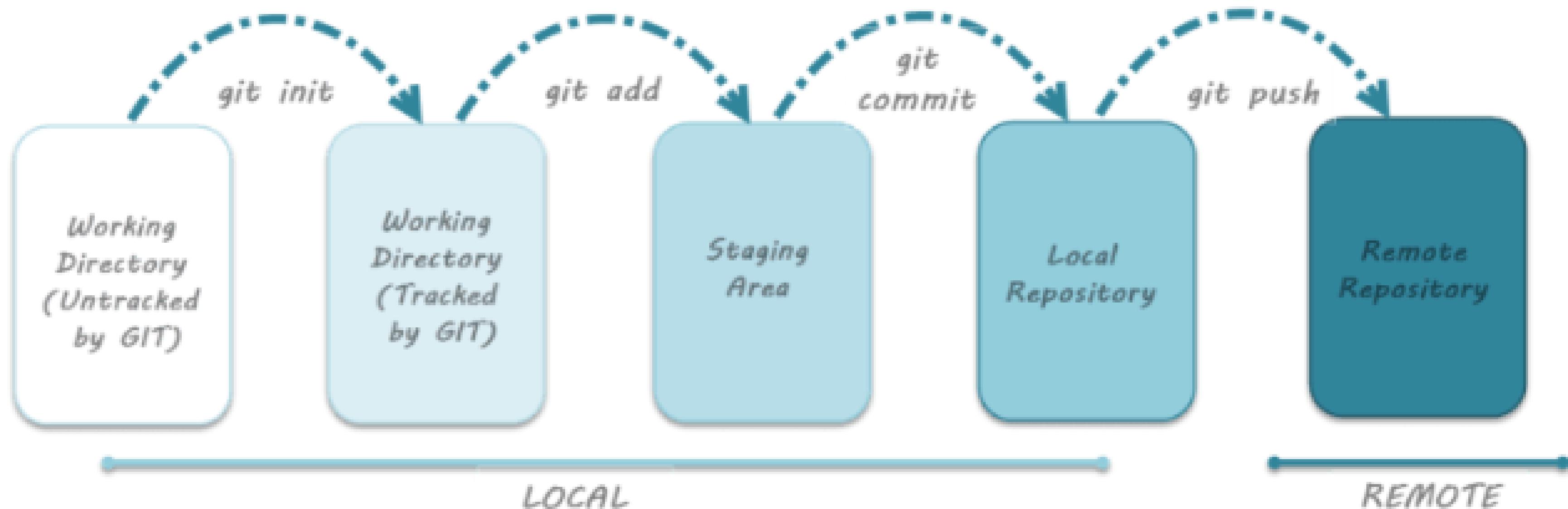


Индексация измененных файлов

- **git add <filename>** - добавляем нужные файлы после того как изменили их
- будут подсвечиваться **зеленым**
- если проиндексировали случайно, то при помощи команды **git reset <file>** можно это отменить
- проиндексированные файлы попадут в слепок - в коммит после команды **git commit -m "comment"**

неотслеживаемое (untracked);
измененное (modified);
подготовленное (staged);
закомиченное (committed).

Жизненный цикл



Что можно делать с коммитами

- **git log** - показать историю коммитов, тут будут их hash-ы
- **git commit --amend -m "comment"** - исправить последний коммит
- **git show <hash>** - показать детали коммита
- **git diff** - показать разницу между текущим состоянием и предыдущим коммитом
- **git diff 09bd8cc..ba25c0ff** - посмотреть разницу между двумя коммитами
- **git checkout 09bd8cc1 hello.txt** - вернуть файл к состоянию коммита 09bd8cc1

Когда нужно отменить коммит

- **git revert HEAD** - отменить последний коммит, при этом создается коммит, который отменяет изменения
- **git revert <commit_hash>** - откатиться до коммита с определенным SHA
- **git reset --soft <commit_hash>** - отменит все до коммита <commit_hash>, при этом изменения, которые были после, окажутся в индексе
- **git reset --mixed <commit_hash>** - отменит все до коммита <commit_hash>, при этом изменения, которые были после, окажутся not staged
- **git reset --hard <commit_hash>** - отменит все до коммита <commit_hash>, при этом изменения, которые были после, исчезнут

Очень осторожно с git push -f и git reset --hard



2

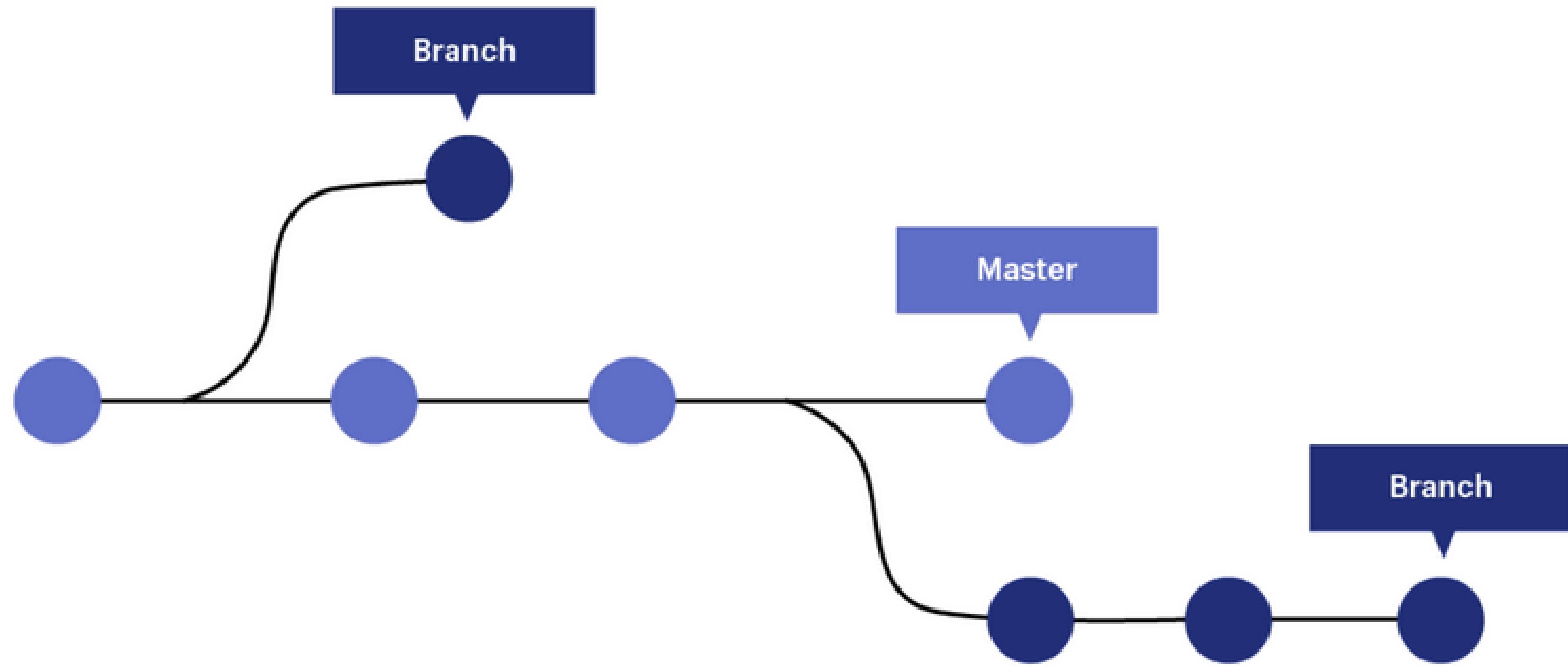
Ветки

Кто такие ветки

Ветка - обособленное направление разработки. Есть основной код, который в main, который сейчас у пользователей, а мы разрабатываем новые функции в отдельной ветке, чтобы мы не могли поломать то, чем сейчас пользуются люди. Чем-то похоже на черновик



Как это выглядит



Что можно делать с ветками

- **git branch** - показать список веток
- **git checkout -b <имя_новой_ветки>** - создать ветку и переключиться на нее
- **git checkout <имя_существующей_ветки>** - переключиться на существующую ветку
- **git branch -d <имя_ветки>** - удалить ветку
- **git merge <имя_ветки>** - сделать слияние текущей ветки с <имя_ветки> (делается одним коммитом)
- **git rebase master** - переместить коммиты текущей ветки в мастер

GIT Flow

3

GIT Flow

Методология работы с Git, определяет какие ветки нужно создать и как производить их слияние

- Создается репозиторий
- Репозиторий инициализируется
- Начинается работа на ветке `develop`
- Возникает необходимость опробовать новую штуку - создается `feature`-ветка и делаются коммиты
- Закончив работу на `feature`-ветке, вы сливаете ее с `develop`
- Если вы довольны текущей версией, но хотите продолжить работу, создается ветка `release`, куда перемещается текущая версия. Правка багов будет происходить на этой же ветке.
- Когда с веткой `release` покончено, время слить ее в `master` и продолжить работу с `develop`
- Кроме того, этот момент можно отметить на `master`-ветке

GitHub

GitHub — сервис онлайн-хостинга репозиториев, обладающий всеми функциями распределённого контроля версий и функциональностью управления исходным кодом — всё, что поддерживает Git и даже больше.

Также GitHub может похвастаться контролем доступа, багтрекингом, управлением задачами и вики для каждого проекта.

GitLab

GitLab — веб-приложение и система управления репозиториями программного кода для Git.

GitLab является конкурентом GitHub, в котором среди многих других проектов размещается разработка ядра Linux Линуса Торвальдса. Поскольку GitLab разрабатывается на той же основе управления версиями (Git), принцип их работы схож. GitLab поддерживает как публичные, так и неограниченное количество частных ветвей разработки.

BitBucket

Веб-сервис для хостинга проектов и их совместной разработки, основанный на системах контроля версий Mercurial и Git. По назначению и основным предлагаемым функциям аналогичен GitHub, от которого отличается с одной стороны меньшей пользовательской базой, а с другой, имеет определённые преимущества в плане размещения непубличных репозиториев — возможностью их бесплатного хостинга с ограничением на размер команды не более пяти человек и меньшей арендной платой при большем размере команды, а также управлением правами доступа на уровне отдельных ветвей проекта.

Pull Request

4

Pull Request

PR - запрос на включение сделанных вами изменений в ветку , сделать его можно при помощи графического интерфейса в GitHub

The screenshot shows a GitHub repository page for 'tomjoht/testrepo888'. At the top, there are navigation links for 'Code', 'Issues 0', 'Pull requests 0' (which is highlighted in orange), 'Projects 0', 'Wiki', 'Insights', and 'Settings'. To the right of the repository name are buttons for 'Unwatch 1', 'Star 0', and 'Fork 0'. A modal window titled 'Label issues and pull requests for new contributors' is displayed, explaining that GitHub will help potential first-time contributors discover issues labeled with 'help wanted' or 'good first issue'. Below the modal, there is a search bar with the query 'is:pr is:open', filter buttons for 'Labels' and 'Milestones', and a large green button labeled 'New pull request'. Further down, there are filters for '0 Open' and '3 Closed' pull requests, and dropdown menus for 'Author', 'Labels', 'Projects', 'Milestones', 'Reviews', 'Assignee', and 'Sort'. At the bottom, a message states 'There aren't any open pull requests.' and provides a link to a new search query.

tomjoht / testrepo888

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Label issues and pull requests for new contributors

Now, GitHub will help potential first-time contributors discover issues labeled with [help wanted](#) or [good first issue](#)

Dismiss

Filters is:pr is:open Labels Milestones

New pull request

0 Open 3 Closed Author Labels Projects Milestones Reviews Assignee Sort

There aren't any open pull requests.

Use the links above to find what you're looking for, or try [a new search query](#). The Filters menu is also super helpful for

5

домашнее задание

Домашнее задание

1. Создать новый репозиторий

а. создать репозиторий локально

б. создать репозиторий на github

с. дать доступ elenadeykun

д. создать файлик REAMDE.md, в нем написать "Репозиторий для дз"

е. запушить это

ф. создать ветку lesson-7 и перейти в нее (git checkout -b lesson-7)

г. решить задачи из 2 части, запушить изменения (git push --set-upstream origin lesson-7)

х. открыть ПР и поставить меня в ревьюеры

2. Решать задачи из дополнительных материалов (ссылка в чате), минимум по 2 задачи из тем функции и декораторы

Команды, которые нужны для выполнения дз

1. Инструкция по GIT (ссылка в чате), оттуда выполнить все 3 шага, на этом этапе добавляем README.md файлик
2. Работа с ветками
 - a. `git checkout -b lesson-7` - переключиться на новую ветку
 - b. `git add <filename>`
 - c. `git commit -m "осмысленный комментарий на английском"`
 - d. `git push --set-upstream origin lesson-7` - запушить изменения
3. В графическом интерфейсе GitHub создать pull request и добавить elenadeykun в reviewers

Спасибо за внимание!