Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Информационных технологий и программирования

Работа: Лабораторная работа №2

Выполнил: Ерёмин Владимир Ильич

Проверил: Ельников Сергей Сергеевич

Санкт-Петербург

2022 г.

Лабораторная работа 2: Изучение системы управления версиями Git

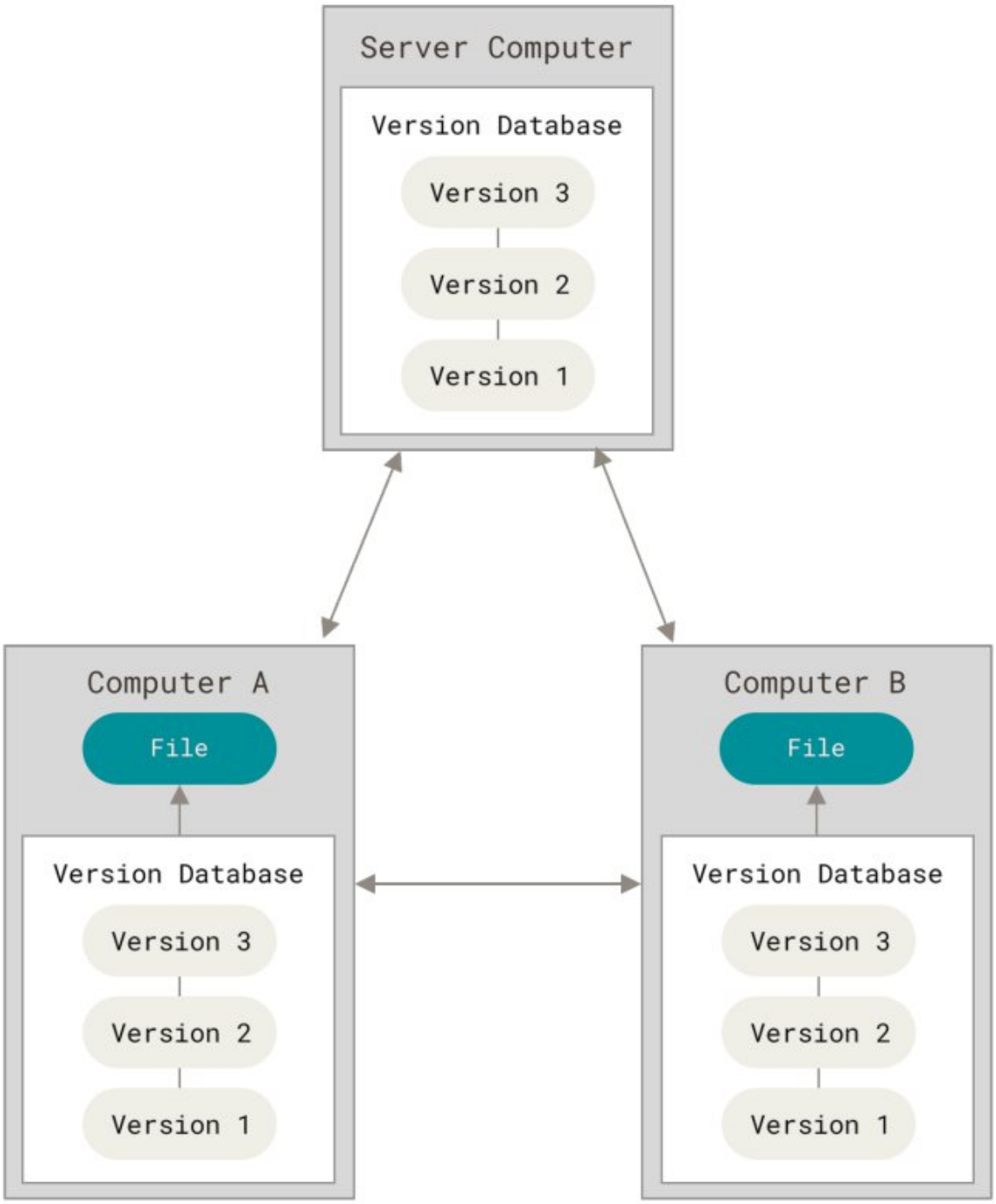
Система контроля версий — это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии.

СКВ можно разбить на 3 категории:

- Локальные системы контроля версий
- Централизованные системы контроля версий
- Распределённые системы контроля версий < ***Git***

Очевидно, локальная система не подходит для совместной работы. А централизованные системы, помимо того, что требуют постоянного подключения к центральному узлу, уязвимы к полной потере данных. **Локальные СКВ** страдают от той же самой проблемы: когда вся история проекта хранится в одном месте, вы рискуете потерять всё.

Преимущество распределенного подхода состоит в том, что СКВ полностью копируют репозиторий. В этом случае, если один из серверов, через который разработчики обменивались данными, умрет, любой клиентский репозиторий может быть скопирован на другой сервер для продолжения работы. Каждая копия репозитория является полным бэкапом всех данных. Более того, многие РСКВ могут одновременно взаимодействовать с несколькими удалёнными репозиториями, благодаря этому вы можете работать с различными группами людей, применяя различные подходы единовременно в рамках одного проекта. Это позволяет применять сразу несколько подходов в разработке, например, иерархические модели, что совершенно невозможно в централизованных системах



Хранение данных в Git похоже на набор снимков миниатюрной файловой системы. Каждый раз, когда вы делаете коммит, то есть сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок. Для увеличения эффективности, если файлы не были изменены, Git не запоминает эти файлы вновь, а только создает ссылку на предыдущую версию идентичного файла, который уже сохранен. Git представляет свои данные как, скажем, поток снимков.

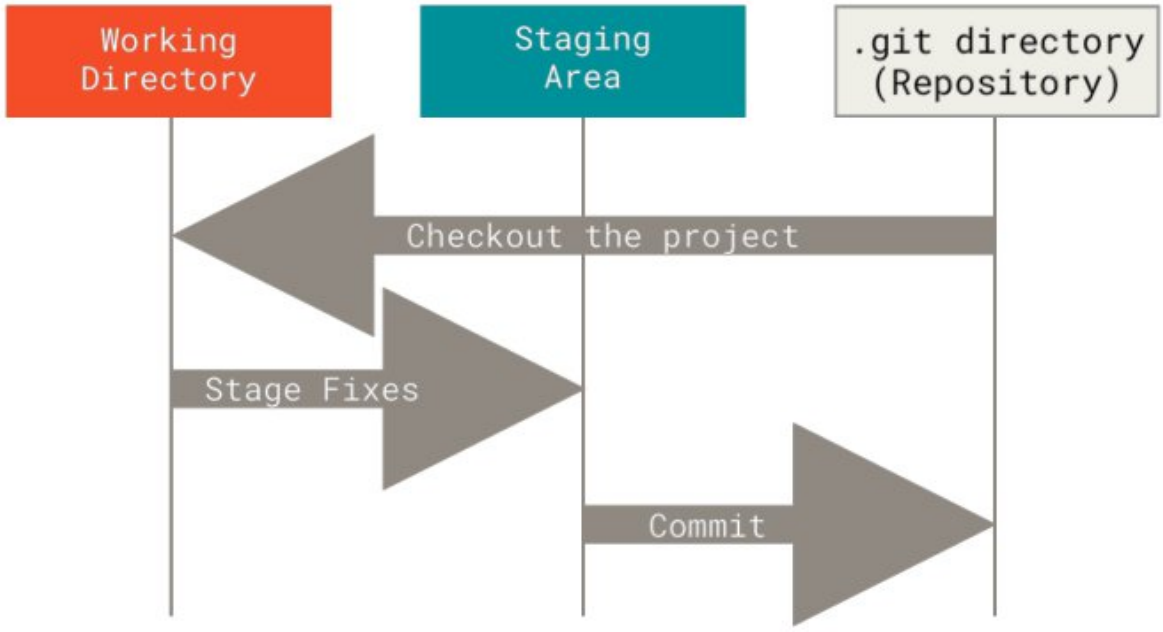
Когда вы производите какие-либо действия в Git, практически все из них только добавляют новые данные в базу Git. Очень сложно заставить систему удалить данные либо сделать что-то, что нельзя впоследствии отменить. Как и в любой другой СКВ, вы можете потерять или испортить свои изменения, пока они не зафиксированы, но после того, как вы фиксируете снимок в Git, будет очень сложно что-либо потерять, особенно, если вы регулярно синхронизируете свою базу с другим репозиторием.

Для работы большинства операций в Git достаточно локальных файлов и ресурсов — в основном, системе не нужна никакая информация с других компьютеров в вашей сети. Единственное, когда требуется подключение, это выгрузка в/загрузка из **Remote.**

В Git для всего вычисляется хеш-сумма, и только потом происходит сохранение. В дальнейшем обращение к сохраненным объектам происходит по этой хеш-сумме.

Все изменения, которые происходят в Git построены на трех состояниях:

- Modified, которые изменились, но еще не были зафиксированы.
- Staged — это изменённый файл в его текущей версии, отмеченный для включения в следующий коммит.
- Committed Зафиксированный значит, что файл уже сохранен в вашей локальной базе.



Рабочая копия (checkout) является снимком одной версии проекта. Эти файлы извлекаются из сжатой базы данных в каталоге Git и помещаются на диск, для того чтобы их можно было использовать или редактировать.

Задание 1

Репозиторий

Создатель - Я.

Drumnom - Кириленко Илья М3100.

Был успешно произведен Fork, а затем Pull Request в моём репозитории.

Задание 2.

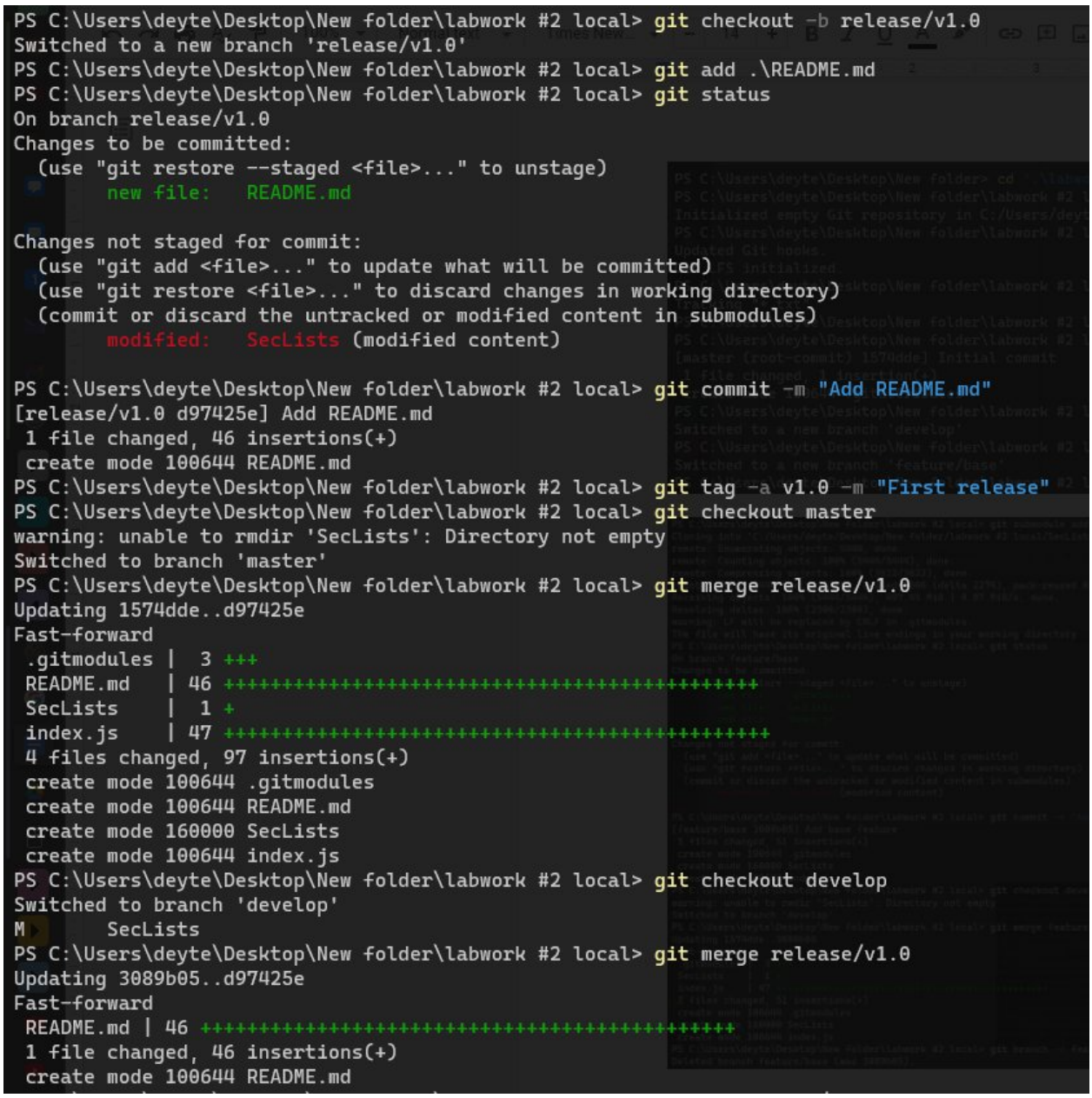
Выполнено, но github все локальные сливания записывает в master....

```
PS C:\Users\deyte\Desktop\New folder> cd '.\labwork #2 local\'
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git init
Initialized empty Git repository in C:/Users/deyte/Desktop/New folder/labwork #2 local/.git/
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git lfs install
Updated Git hooks.
Git LFS initialized.
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git lfs track "*.txt"
Tracking "*.txt"
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git add .\.gitattributes
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git commit -m "Initial commit"
[master (root-commit) 1574dde] Initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 .gitattributes
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git checkout -b develop
Switched to a new branch 'develop'
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git checkout -b feature/base
Switched to a new branch 'feature/base'
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git add .\index.js
```

```
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git submodule add --depth 1 https://github.com/danielmiessler/SecLists.git SecLists
Cloning into 'C:/Users/deyte/Desktop/New folder/labwork #2 local/SecLists'...
remote: Enumerating objects: 5444, done.
remote: Counting objects: 100% (5444/5444), done.
remote: Compressing objects: 100% (3022/3022), done.
remote: Total 5444 (delta 2300), reused 5006 (delta 2274), pack-reused 0
Receiving objects: 100% (5444/5444), 497.45 MiB | 9.87 MiB/s, done.
Resolving deltas: 100% (2300/2300), done.
warning: LF will be replaced by CRLF in .gitmodules.
The file will have its original line endings in your working directory
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git status
On branch feature/base
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .gitmodules
        new file:   SecLists
        new file:   index.js

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)
  (commit or discard the untracked or modified content in submodules)
        modified:   SecLists (modified content)

PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git commit -m "Add base feature"
[feature/base 3089b05] Add base feature
 3 files changed, 51 insertions(+)
 create mode 100644 .gitmodules
 create mode 160000 SecLists
 create mode 100644 index.js
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git checkout develop
warning: unable to rmdir 'SecLists': Directory not empty
Switched to branch 'develop'
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git merge feature/base
Updating 1574dde..3089b05
Fast-forward
 .gitmodules |  3 +++
 SecLists    |  1 +
 index.js    | 47 +++++++++++++++++++++++++++++++++++++++++++++++
 3 files changed, 51 insertions(+)
 create mode 100644 .gitmodules
 create mode 160000 SecLists
 create mode 100644 index.js
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git branch -d feature/base
Deleted branch feature/base (was 3089b05).
```



(переделал, чтобы сохранить скрины)

```
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git branch -d release/v1.0
Deleted branch release/v1.0 (was d97425e).
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git checkout master
Switched to branch 'master'
M       SecLists
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git checkout -b hotfix/license
Switched to a new branch 'hotfix/license'
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git add .\LICENSE
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git commit -m "Add license"
[hotfix/license 3e39f6f] Add license
 1 file changed, 21 insertions(+)
 create mode 100644 LICENSE
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git tag -a v1.1 -m "First hotfix"
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git checkout master
Switched to branch 'master'
M       SecLists
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git merge hotfix/license
Updating d97425e..3e39f6f
Fast-forward
 LICENSE | 21 ++++++++++++++++++++
 1 file changed, 21 insertions(+)
 create mode 100644 LICENSE
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git checkout develop
Switched to branch 'develop'
M       SecLists
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git merge hotfix/license
Updating d97425e..3e39f6f
Fast-forward
 LICENSE | 21 ++++++++++++++++++++
 1 file changed, 21 insertions(+)
 create mode 100644 LICENSE
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git -d hotfix/license
unknown option: -d
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [--super-prefix=<path>] [--config-env=<name>=<envvar>]
           <command> [<args>]
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git branch -d hotfix/license
Deleted branch hotfix/license (was 3e39f6f).
```

```
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git remote add origin https://github.com/unknowableshade/Git-Flow-Playground.git
PS C:\Users\deyte\Desktop\New folder\labwork #2 local> git push --all origin
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 12 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (13/13), 2.70 KiB | 690.00 KiB/s, done.
Total 13 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/unknowableshade/Git-Flow-Playground.git
 * [new branch]      develop -> develop
 * [new branch]      master -> master
PS C:\Users\deyte\Desktop\New folder\labwork #2 local>
```

Задание 3. Команды GIT.

`git init`
initialize an existing directory as a Git repository
`git clone [url]`
retrieve an entire repository from a hosted location via URL

`git status`
show modified files in working directory, staged for your next commit
`git add [file]`
add a file as it looks now to your next commit (stage)
`git reset [file]`
unstage a file while retaining the changes in working directory
`git diff`
diff of what is changed but not staged
`git diff --staged`
diff of what is staged but not yet committed
`git commit -m "[descriptive message]"`
commit your staged content as a new commit snapshot

`git branch`
list your branches. a * will appear next to the currently active branch
`git branch [branch-name]`
create a new branch at the current commit
`git checkout`
switch to another branch and check it out into your working directory
`git merge [branch]`
merge the specified branch's history into the current one
`git log`
show all commits in the current branch's history

`git log`
show the commit history for the currently active branch

`git remote add [alias] [url]`
add a git URL as an alias
`git fetch [alias]`
fetch down all the branches from that Git remote
`git merge [alias]/[branch]`
merge a remote branch into your current branch to bring it up to date
`git push [alias] [branch]`
Transmit local branch commits to the remote repository branch
`git pull`
fetch and merge any commits from the tracking remote branch