

Отчёт по лабораторной работе №2

Управление версиями

Матвеева Анастасия Сергеевна

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
4	Выводы	13
5	Ответы на контрольные вопросы	14

List of Figures

3.1	Создание репозитория	6
3.2	Инициализация репозитория	7
3.3	Создание SSH-ключа	7
3.4	Добавление ключа на github.com	8
3.5	Загрузка файлов	8
3.6	Отправка в сетевой репозиторий по SSH	9
3.7	Инициализация git-flow и начало релиза	10
3.8	Завершение релиза и отправка изменений в сетевой репозиторий	11
3.9	Объединение веток в сетевом репозитории	12

1 Цель работы

Целью данной работы является изучение идеологии и применения средств контроля версий.

2 Задание

1. Создать учетную запись на github.com.
2. Настроить репозиторий и организовать доступ по ssh.
3. Изучить механизм управления версиями.



3 Выполнение лабораторной работы

1. Создаем учетную запись на github.com и репозиторий.

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 *


 asmatveeva158 / OS-Labs 

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

Description (optional)

Лабораторные работы по операционным системам

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

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

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

Figure 3.1: Создание репозитория

2. Инициализируем локальный репозиторий и создаю в нем файл README.md.

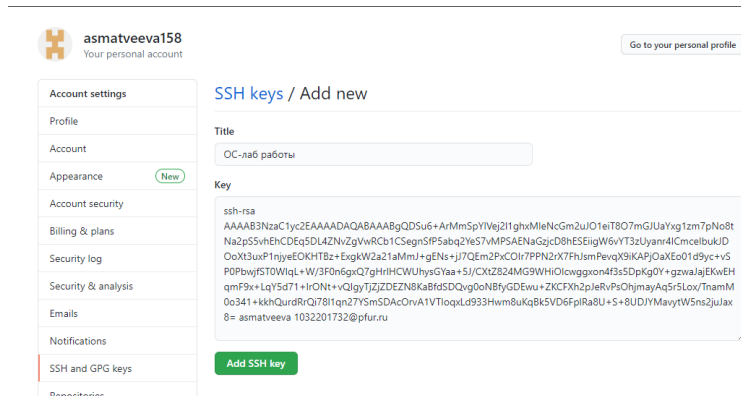


Figure 3.4: Добавление ключа на github.com

4. Загружаем файлы лицензионного соглашения и gitignore. Отправляем все файлы в сетевой репозиторий.

```

Терминал - asmatveeva@asmatveeva: ~/laboratory
Файл  Правка  Вид  Терминал  Вкладки  Справка
asmatveeva@asmatveeva:~/laboratory$ wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O LICENSE
--2021-04-28 12:28:18--  https://creativecommons.org/licenses/by/4.0/legalcode.t
xt
Распознаётся creativecommons.org (creativecommons.org)... 172.67.34.140, 104.20.15
0.16, 104.20.151.16, ...
Подключение к creativecommons.org (creativecommons.org)|172.67.34.140|:443... со
единение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: нет данных [text/plain]
Сохранение в: «LICENSE»

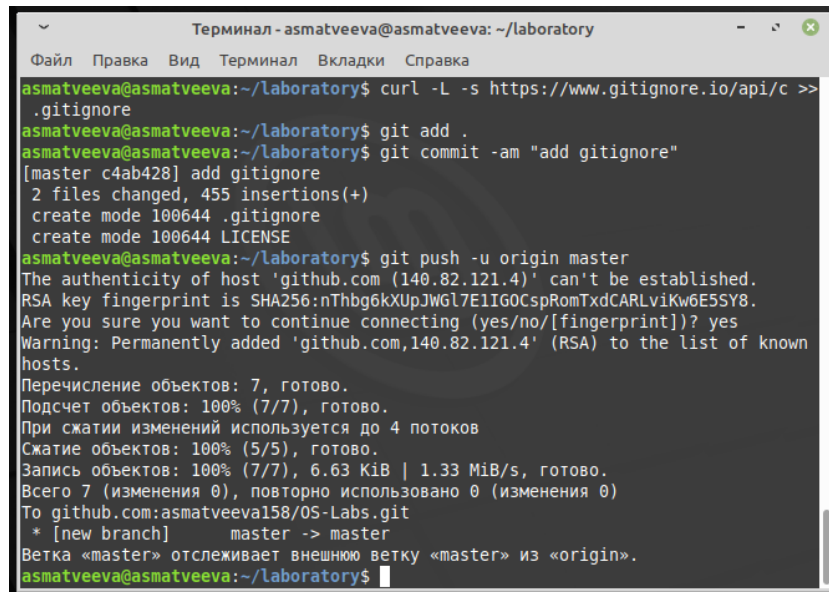
LICENSE      [ <=> ] 18,22K  --.-KB/s  за 0,001s

2021-04-28 12:28:19 (26,0 MB/s) - «LICENSE» сохранён [18657]

asmatveeva@asmatveeva:~/laboratory$ curl -L -s https://www.gitignore.io/api/list
1c,1c-bitrix,a-frame,actionscript,ada
adobe,advancedinstaller,adventuregamestudio,agda,al
alteraquartusii,altium,amplify,android,androidstudio
angular,anjuta,ansible,apachecordova,apachehadoop
appbuilder,appcelerator titanium,appcode,appcode+all,appcode+iml
appengine,aptanastudio,arcanist,archive,archives
archlinuxpackages,aspnetcore,assembler,ate,atmelstudio

```

Figure 3.5: Загрузка файлов



```
Терминал - asmatveeva@asmateveeva: ~/laboratory
Файл  Правка  Вид  Терминал  Вкладки  Справка
asmateveeva@asmateveeva:~/laboratory$ curl -L -s https://www.gitignore.io/api/c >>
.gitignore
asmateveeva@asmateveeva:~/laboratory$ git add .
asmateveeva@asmateveeva:~/laboratory$ git commit -am "add gitignore"
[master c4ab428] add gitignore
2 files changed, 455 insertions(+)
create mode 100644 .gitignore
create mode 100644 LICENSE
asmateveeva@asmateveeva:~/laboratory$ git push -u origin master
The authenticity of host 'github.com (140.82.121.4)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWGL7E1IG0CspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com,140.82.121.4' (RSA) to the list of known
hosts.
Перечисление объектов: 7, готово.
Подсчет объектов: 100% (7/7), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (5/5), готово.
Запись объектов: 100% (7/7), 6.63 KiB | 1.33 MiB/s, готово.
Всего 7 (изменения 0), повторно использовано 0 (изменения 0)
To github.com:asmateveeva158/OS-Labs.git
* [new branch]      master -> master
Ветка «master» отслеживает внешнюю ветку «master» из «origin».
asmateveeva@asmateveeva:~/laboratory$
```

Figure 3.6: Отправка в сетевой репозиторий по SSH

5. Использование системы управления версиями. Создаем ветку, начинаем и завершаем в ней релиз.

```
Терминал - asmatveeva@asmateveeva: ~/laboratory
Файл  Правка  Вид  Терминал  Вкладки  Справка
Branch name for "next release" development: [develop]
How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? [] v
Hooks and filters directory? [/home/asmateveeva/laboratory/.git/hooks]
asmateveeva@asmateveeva:~/laboratory$ git branch
* develop
  master
asmateveeva@asmateveeva:~/laboratory$ git flow release start 1.0.0
Переключено на новую ветку «release/1.0.0»

Summary of actions:
- A new branch 'release/1.0.0' was created, based on 'develop'
- You are now on branch 'release/1.0.0'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

    git flow release finish '1.0.0'

asmateveeva@asmateveeva:~/laboratory$ echo "1.0.0" >> VERSION
asmateveeva@asmateveeva:~/laboratory$ git add .
asmateveeva@asmateveeva:~/laboratory$ git commit -am "chore(main): add version"
[release/1.0.0 5860fae] chore(main): add version
1 file changed, 1 insertion(+)
 create mode 100644 VERSION
asmateveeva@asmateveeva:~/laboratory$
```

Figure 3.7: Инициализация git-flow и начало релиза

```
Терминал - asmatveeva@asmateveeva: ~/laboratory
Файл  Правка  Вид  Терминал  Вкладки  Справка
Merge made by the 'recursive' strategy.
VERSION | 1 +
1 file changed, 1 insertion(+)
create mode 100644 VERSION
Ветка release/1.0.0 удалена (была 5860fae).

Summary of actions:
- Release branch 'release/1.0.0' has been merged into 'master'
- The release was tagged 'v1.0.0'
- Release tag 'v1.0.0' has been back-merged into 'develop'
- Release branch 'release/1.0.0' has been locally deleted
- You are now on branch 'develop'

asmateveeva@asmateveeva:~/laboratory$ git push --all
Warning: Permanently added the RSA host key for IP address '140.82.121.3' to the
list of known hosts.
Перечисление объектов: 6, готово.
Подсчет объектов: 100% (6/6), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (4/4), готово.
Запись объектов: 100% (5/5), 489 bytes | 244.00 KiB/s, готово.
Всего 5 (изменения 3), повторно использовано 0 (изменения 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To github.com:asmateveeva158/OS-Labs.git
   c4ab428..e9463c3  master -> master
   * [new branch]      develop -> develop
asmateveeva@asmateveeva:~/laboratory$ git push --tags
Перечисление объектов: 1, готово.
Подсчет объектов: 100% (1/1), готово.
Запись объектов: 100% (1/1), 160 bytes | 160.00 KiB/s, готово.
Всего 1 (изменения 0), повторно использовано 0 (изменения 0)
To github.com:asmateveeva158/OS-Labs.git
   * [new tag]         v1.0.0 -> v1.0.0
asmateveeva@asmateveeva:~/laboratory$
```

Figure 3.8: Завершение релиза и отправка изменений в сетевой репозиторий

6. Выполним объединение веток.

Merge tag 'v1.0.0' into develop #1

Merged asmatveeva158 merged 1 commit into `master` from `develop` 1 minute ago

Conversation 0 Commits 1 Checks 0 Files changed 0

asmatveeva158 commented 1 minute ago

ver 1 v1.0.0

Merge tag 'v1.0.0' into develop 3058bdd

asmatveeva158 merged commit b123fe6 into `master` now

Pull request successfully merged and closed
You're all set—the `develop` branch can be safely deleted.

Delete branch

Write Preview

Leave a comment

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

Comment

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

Figure 3.9: Объединение веток в сетевом репозитории

4 Выводы

Мы приобрели практические навыки работы с системой контроля версий git и создали свой репозиторий.

5 Ответы на контрольные вопросы

1. Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` с различными опциями. Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом.
2. В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять неполную версию изменённых файлов, а производить так называемую дельта-компрессию—сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви.

Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

3. Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. Пример - Wikipedia.

В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. Пример — Bitcoin.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером.

4. Создадим локальный репозиторий. Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория:

```
git config --global user.name "Имя Фамилия" git config --global user.email "work@mail"
и настроив utf-8 в выводе сообщений git:
```

```
git config --global core.quotePath false
```

Для инициализации локального репозитория, расположенного, например, в каталоге ~/tutorial, необходимо ввести в командной строке:

```
cd
```

```
mkdir tutorial
```

```
cd tutorial
```

```
git init
```

5. Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый):

```
ssh-keygen -C“Имя Фамилия work@mail”
```

Ключи сохраняются в каталоге ~/.ssh/.

Скопировав из локальной консоли ключ в буфер обмена

```
cat ~/.ssh/id_rsa.pub | xclip -sel clip
```

вставляем ключ в появившееся на сайте поле.

6. У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.

7. Основные команды git:

Наиболее часто используемые команды git: – создание основного дерева репозитория: `git init` – получение обновлений (изменений) текущего дерева из центрального репозитория: `git pull` – отправка всех произведённых изменений локального дерева в центральный репозиторий: `git push` – просмотр списка изменённых файлов в текущей директории: `git status` – просмотр текущих изменений: `git diff` – сохранение текущих изменений: – добавить все изменённые и/или созданные файлы и/или каталоги: `git add .` – добавить конкретные изменённые и/или созданные файлы и/или каталоги: `git add имена_файлов` – удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): `git rm имена_файлов` – сохранение добавленных изменений: – сохранить все добавленные изменения и все изменённые файлы: `git commit -am ‘Описание коммита’` – сохранить добавленные изменения с внесением комментария через встроенный редактор: `git commit` – создание новой ветки, базирующейся на текущей: `git checkout -b имя_ветки` – переключение на некоторую ветку: `git checkout имя_ветки` (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой) – отправка изменений конкретной ветки в центральный репозиторий: `git push origin имя_ветки` – слияние ветки стекущим деревом: `git merge --no-ff имя_ветки` – удаление ветки: – удаление локальной уже слитой с основным деревом ветки: `git`

`branch -d имя_ветки`–принудительное удаление локальной ветки:`git branch -D имя_ветки`–удаление ветки с центрального репозитория: `git push origin :имя_ветки`

8. Использование `git` при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий):

```
git add hello.txt
```

```
git commit -am 'Новый файл'
```

9. Проблемы, которые решают ветки `git`:

- нужно постоянно создавать архивы с рабочим кодом
- сложно “переключаться” между архивами
- сложно перетаскивать изменения между архивами
- легко что-то напутать или потерять

10. Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл `.gitignore` с помощью сервисов. Для этого сначала нужно получить список имеющихся шаблонов:
- ```
curl -L -s https://www.gitignore.io/api/list
```

Затем скачать шаблон, например, для C и C++

```
curl -L -s https://www.gitignore.io/api/c » .gitignore
```

```
curl -L -s https://www.gitignore.io/api/c++ » .gitignore
```