

Отчёт по лабораторной работе №2

Управление версиями

Станиловский Эрик Кириллович НПМбд-02-20

Содержание

1	Цель работы	1
2	Выполнение лабораторной работы	1
3	Вывод	7

1 Цель работы

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

2 Выполнение лабораторной работы


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 *

 ekstanilovskiij ▾


 /

labs-os ✓


Great repository names are short and memorable. Need inspiration? How about [didactic-octo-eureka?](#)

Description (optional)

Операционные системы

☒  **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

Рисунок 1: Создание репозитория

Инициализируем локальный репозиторий и создаю в нем файл README.md

```
Терминал - екстанилловский@екстанилловский: ~/laboratory
Файл  Правка  Вид  Терминал  Вкладки  Справка
екстанилловский@екстанилловский:~$ mkdir laboratory
екстанилловский@екстанилловский:~$ cd laboratory
екстанилловский@екстанилловский:~/laboratory$ git init
Инициализирован пустой репозиторий Git в /home/екстанилловский/laboratory/.git/
екстанилловский@екстанилловский:~/laboratory$ echo "# Лабораторные работы" >> README.md
екстанилловский@екстанилловский:~/laboratory$ git add README.md
екстанилловский@екстанилловский:~/laboratory$ git config --global user.name "екстанилловский"
екстанилловский@екстанилловский:~/laboratory$ git config --global user.email "1032201731@pfur.ru"
екстанилловский@екстанилловский:~/laboratory$ git commit -m "first commit"
[master (корневой коммит) 838a4ae] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md
екстанилловский@екстанилловский:~/laboratory$
```

Рисунок 2: Инициализация репозитория

Создаем SSH-ключ и прописываем его в настройках на github.com

```
Терминал - екстанилловский@екстанилловский: ~/laboratory
Файл  Правка  Вид  Терминал  Вкладки  Справка
Enter same passphrase again:
Your identification has been saved in /home/екстанилловский/.ssh/id_rsa
Your public key has been saved in /home/екстанилловский/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:NvL0VdPk1smm44H8VwYqwsC78l5cHQNIr3Trw8S9Y0A екстанилловский 1032201731@pfur.ru
The key's randomart image is:
+---[RSA 3072]-----+
|      .      |
|     . . .   |
|    o o o   ++=|
|   * =.+. =o+ |
|  o E 0o=+  o |
| B X +o.o..   |
| . . + = .o . |
|  o . . .    |
| .o          |
+---[SHA256]-----+
екстанилловский@екстанилловский:~/laboratory$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAgQDNPH8/JBFPKd46w3MnAXEgLFx6LLvHBmhZoPnAlM08b+e7y/nrntmSgnTWLa+WXRIg1S71XZwc
BxN0xBrXuJT0TFEZZTSK5Ap1FWgIrLM3doNHfp/wLrt+Y5cMqGDFt2cAhOWcuUSciwnsnhetm+uMBY0g10AkxDpNv3R5K1A5s4NustHCR7K8UNcE
pkjhFDUPXroRULwECU/SKYMqpxDm8rkGv6UoLTsLKH1uQdUG80TmmhLThr3qAtEyzfWBeOedhy007Idx27zjh32Dqwb0GkgcJozG7TnwG23M2hCq
514zdxEcsh6n2Q60Uw3UuiBSVUqSZdc6aonwYX2/6pgeHJF0eaYrYW+UwDTguFiRSuxjZMbKo2ptUfzphASXaShleNR6GTwSqvRmjvqZjdqjWsU4
0+lPi/3brr5+TGeXebAELW+58x+z+qFedbwMfufMinz08ghclY6KnsFtIVAZDNfmktwPKV7mJMvdPaL/2Hm/rKICipogT0/J3ZTqbM= екстани
ловский 1032201731@pfur.ru
екстанилловский@екстанилловский:~/laboratory$ git remote add origin git@github.com:екстанилловский/labs-os.git
екстанилловский@екстанилловский:~/laboratory$
```

Рисунок 3: Создание SSH-ключа

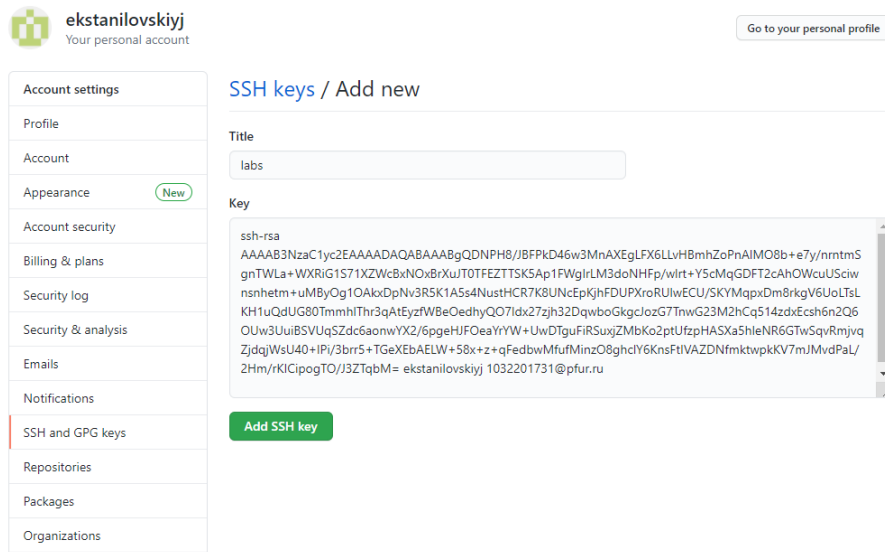


Рисунок 4: Добавление ключа на github.com

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

```
Терминал - ekstanilovskiyj@ekstanilovskiyj: ~/laboratory
Файл  Правка  Вид  Терминал  Вкладки  Справка
ekstanilovskiyj@ekstanilovskiyj:~/Laboratory$ wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O
LICENSE
--2021-04-23 01:06:34--  https://creativecommons.org/licenses/by/4.0/legalcode.txt
Распознаётся creativecommons.org (creativecommons.org)... 104.20.151.16, 172.67.34.140, 104.20.150.16, ...
Подключение к creativecommons.org (creativecommons.org)|104.20.151.16|:443... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: нет данных [text/plain]
Сохранение в: «LICENSE»

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

2021-04-23 01:06:34 (23,3 MB/s) - «LICENSE» сохранён [18657]

ekstanilovskiyj@ekstanilovskiyj:~/Laboratory$ curl -L -s https://www.gitignore.io/api/list
1c,1c-bitrix,a-frame,actionsript,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
ats,audio,automationstudio,autotools,autotools+strict
awr,azurefunctions,backup,ballerina,basercms
basic,batch,bazaar,bazel,bitrise
bitrix,bittorrent,blackbox,bloop,bluej
bookdown,bower,bricxcc,buck,c
```

Рисунок 5: Загрузка файлов

```
Терминал - екстанилловский@екстанилловский: ~/laboratory
Файл Правка Вид Терминал Вкладки Справка
xcodeinjection,xilinx,xilinxise,xilinxvivado,xill
xojo,xtext,y86,yarn,yeoman
yii,yii2,zendframework,zephir,zig
zsh,zukencr8000екстанилловский@екстанилловский:~/laboratory$
екстанилловский@екстанилловский:~/laboratory$ curl -L -s https://www.gitignore.io/api/c >> .gitignore
екстанилловский@екстанилловский:~/laboratory$ git add .
екстанилловский@екстанилловский:~/laboratory$ git commit -am "add gitignore"
[master 6ee7ebb] add gitignore
2 files changed, 455 insertions(+)
create mode 100644 .gitignore
create mode 100644 LICENSE
екстанилловский@екстанилловский:~/laboratory$ git push -u origin master
The authenticity of host 'github.com (140.82.121.3)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWGL7EIIG0CspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com,140.82.121.3' (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), 6.65 KiB | 1.33 MiB/s, готово.
Всего 7 (изменения 0), повторно использовано 0 (изменения 0)
To github.com:екстанилловский/labs-os.git
* [new branch] master -> master
Ветка «master» отслеживает внешнюю ветку «master» из «origin».
екстанилловский@екстанилловский:~/laboratory$
```

Рисунок 6: Отправка в сетевой репозиторий по SSH

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

```
Терминал - екстанилловский@екстанилловский: ~/laboratory
Файл Правка Вид Терминал Вкладки Справка
create mode 100644 VERSION
екстанилловский@екстанилловский:~/laboratory$ git flow release finish -m "ver 1" 1.0.0
Переключено на ветку «master»
Ваша ветка обновлена в соответствии с «origin/master».
Merge made by the 'recursive' strategy.
VERSION | 1 +
1 file changed, 1 insertion(+)
create mode 100644 VERSION
Уже на «master»
Ваша ветка опережает «origin/master» на 2 коммита.
(используйте «git push», чтобы опубликовать ваши локальные коммиты)
Переключено на ветку «develop»
Merge made by the 'recursive' strategy.
VERSION | 1 +
1 file changed, 1 insertion(+)
create mode 100644 VERSION
Ветка release/1.0.0 удалена (была 5d4c96f).

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'

екстанилловский@екстанилловский:~/laboratory$
```

Рисунок 7: Инициализация git-flow и начало релиза

```
Терминал - ekstanilovskiyj@ekstanilovskiyj: ~/laboratory
Файл  Правка  Вид  Терминал  Вкладки  Справка
- 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'

ekstanilovskiyj@ekstanilovskiyj:~/laboratory$ git push --all
Warning: Permanently added the RSA host key for IP address '140.82.121.4' 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), 494 bytes | 494.00 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:ekstanilovskiyj/labs-os.git
  6ee7ebb..6dde815  master -> master
  * [new branch]    develop -> develop
ekstanilovskiyj@ekstanilovskiyj:~/laboratory$ git push --tags
Перечисление объектов: 1, готово.
Подсчет объектов: 100% (1/1), готово.
Запись объектов: 100% (1/1), 164 bytes | 164.00 KiB/s, готово.
Всего 1 (изменения 0), повторно использовано 0 (изменения 0)
To github.com:ekstanilovskiyj/labs-os.git
  * [new tag]       v1.0.0 -> v1.0.0
ekstanilovskiyj@ekstanilovskiyj:~/laboratory$
```

Рисунок 8: Завершение релиза и отправка изменений в сетевой репозиторий

Выполним объединение веток

Merge tag 'v1.0.0' into develop #1

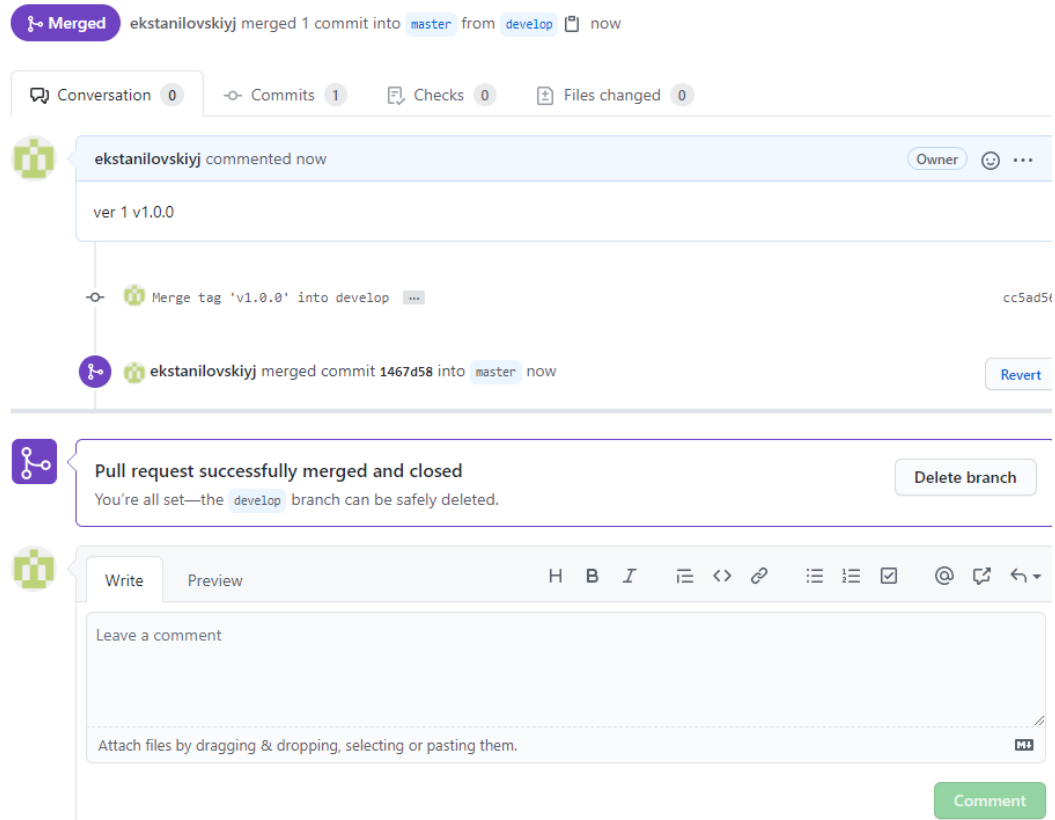


Рисунок 9: Объединение веток в сетевом репозитории

3 Вывод

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

4 Ответы

1) Система контроля версий (Version Control System, VCS) – программное обеспечение для облегчения работы с изменяющейся информацией. VCS позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое. Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется. Сервер может сохранять не полную

версию изменённых файлов, а производить так называемую дельта-компрессию – сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.

2) Хранилище (репозиторий) – в нем хранятся все документы вместе с историей их изменения и другой служебной информацией.

Рабочую копию необходимо периодически синхронизировать с репозиторием, эта операция предполагает отправку в него изменений, которые пользователь внес в свою рабочую копию. За это отвечает команда `commit`. С помощью этой команды можно также передать комментарий о сделанных изменениях.

История – список предыдущих изменений/коммитов.

Рабочая копия – копия проекта, связанная с репозиторием, в которой непосредственно работает пользователь.

3) Централизованные VCS

Клиент-серверная модель: один центральный репозиторий, с которым разработчики взаимодействуют по сети.

Примеры:

CVS

Subversion (SVN)

Perforce

Децентрализованная VCS

Клиенты полностью копируют репозиторий, вместо простого скачивания снимка всех файлов (состояния файлов в определенный момент времени). В этом случае, если сервер выйдет из строя, то клиентский репозиторий можно будет скопировать на другой, рабочий, сервер, ведь данный репозиторий является полным бэкапом всех данных.

Примеры:

Git

Mercurial

Bazaar

4) Действия с VCS при единоличной работе с хранилищем:

Создать новый репозиторий на локальном устройстве (если он не был создан)

Внести изменения в исходные файлы

Выполнить индексацию необходимых файлов

Проверить внесенные изменения

Выполнить commit

Отправить локальный репозиторий на удаленный сервер, при необходимости.

5) Действия при работе с общим хранилищем VCS:

Обычно проект уже создан и его нужно загрузить из общего удаленного хранилища

Необходимо создать свою рабочую ветку

Внести изменения внутри своей рабочей ветки

Выполнить индексацию необходимых файлов на своем локальном устройстве

Проверить внесенные изменения

Выполнить commit

Свою рабочую ветку отправить в общее хранилище

При необходимости внести изменения и отправить снова

После администратор объединит вашу ветку с master branch.

6) Git – это система управления версиями. 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. Git запоминает не все изменения, а только те, которые вы скажите.

Обычно работа с Git выглядит так:

сверстали шапку сайта – сделали commit;

сверстали контент страницы – сделали второй commit;

закончили верстать страницу – сделали третий commit и отправили код на сервер, чтобы вашу работу могли увидеть коллеги, либо чтобы опубликовать страницу с помощью Capistrano.

7) 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 изменения: `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`

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 с текущим деревом: `git merge --no-ff имя_ветки`

удаление ветки: о удаление локальной уже слитой с основным деревом ветки: `git branch -d имя_ветки` о принудительное удаление локальной ветки: `git branch -D имя_ветки` о удаление ветки с центрального репозитория: `git push origin :имя_ветки`

8) Примеры использования VCS при работе с локальными репозиториями: Создание небольшого проекта на своем локальном устройстве, работа с файлами (например, каталог, содержащий документы, презентации, которые будут часто редактироваться), работа с фотографиями, видео и т.д. Примеры использования VCS при работе с удаленными репозиториями: Примеры могут быть те же, но теперь над ними работают несколько человек. Такая система позволяет следить за работой других пользователей.

9) Ветвление («ветка», branch) – один из параллельных участков истории в одном хранилище, исходящих из одной версии (точки ветвления).

Обычно есть главная ветка (master), или ствол (trunk).

Между ветками, то есть их концами, возможно слияние. Ветки используются для разработки одной части функционала изолированно от других. Каждая ветка представляет собой отдельную копию кода проекта. Ветки позволяют одновременно работать над разными версиями проекта. Каждый репозиторий по-умолчанию имеет ветку master. Всякий раз, когда требуется разработка нового функционала, не внося при этом изменений в основную рабочую версию, можно создавать новую ветку, на основе рабочей, и вести разработку в ней – новой копии кода проекта. Когда функционал доделан и протестирован, можно сделать merge – слить отдельную ветку обратно с основной. При слиянии этой временной ветки в основную, все её коммиты разом перенесутся из одной в другую. Ветвление позволяет обеспечить процесс, при котором всегда в наличии будет рабочая версия проекта, в которой не будет частично завершённого функционала находящегося в активной разработке или же непротестированных фич.

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`