

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

Студент: Евселев Д.

Группа: НПМбд-01-20

Преподаватель: Курячий Г.

Цель работы

Изучить идеологию и применение средств контроля версий.

Задание

Научиться пользоваться системой контроля версий git. Создать репозиторий и опубликовать его на github.

Выполнение работы

Создал учетную запись на github

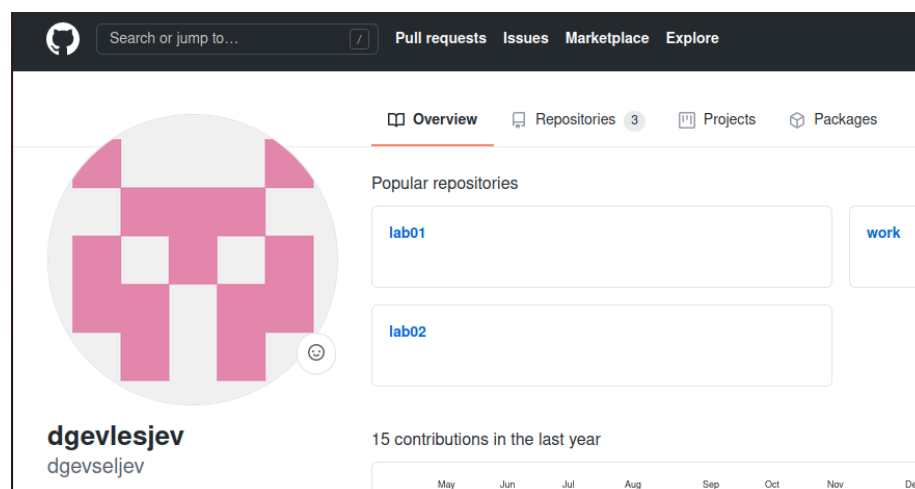


Рис. 1: Рис. 1. Учетная запись на github

```
dimon@vostro:~$ git config --global user.name "Dmitry Evselev"
dimon@vostro:~$ git config --global user.email "1032209458@pfur.ru"
dimon@vostro:~$ git config --global quotepath false
```

Рис. 2: Рис. 2. Предварительная конфигурация

Далее инициализировал локальный репозиторий, расположенный в tutorial. Создал текстовый файл hello.txt и добавил в репозиторий. Воспользовался командой git status для просмотра изменений.

```

dimon@vostro:~$ cd tutorial
dimon@vostro:~/tutorial$ git iniy
git: «iniy» не является командой git. Смотрите «git --help».

Самые похожие команды:
  init
dimon@vostro:~/tutorial$ git init
Инициализирован пустой репозиторий Git в /home/dimon/tutorial/.git/
dimon@vostro:~/tutorial$ ls -l
итого 0
dimon@vostro:~/tutorial$ ls -am
., .., .git
dimon@vostro:~/tutorial$ echo 'hello' > hello.txt
dimon@vostro:~/tutorial$ git add hello.txt
dimon@vostro:~/tutorial$ git commit -am 'new file'
[master (корневой коммит) 3022ac0] new file
 1 file changed, 1 insertion(+)
 create mode 100644 hello.txt
dimon@vostro:~/tutorial$ git status
На ветке master
ничего коммитить, нет изменений в рабочем каталоге

```

Рис. 3: Рис. 3. Добавление файла в локальный репозиторий

Прописал шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore

```

dimon@vostro:~/tutorial$ curl -L -s https://www.gitignore.io/api/c >> .gitignore
dimon@vostro:~/tutorial$ ls -am
., .., .git, .gitignore, hello.txt

```

Рис. 4: Рис. 4. Установка шаблонов игнорируемых типов файлов

Сгенерировал ключ для идентификации пользователя на сервере репозитариев.

Указал ключ в github

Загрузил репозиторий из локального каталога на сервер

Создал заготовку для файла README.md, закоммитил и выложил на github

Далее добавил файл лицензии, шаблон игнорируемых файлов, закоммитил и отправил на github

Далее идет работа с git-flow

Вывод

Научился пользоваться системой контроля версий git. Создал репозиторий и опубликовал его на github.

Ответы на контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

```

dimon@vostro:~/tutorial$ ssh-keygen -C "Dmitry Evselev 1032209458@pfur.ru"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/dimon/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/dimon/.ssh/id_rsa
Your public key has been saved in /home/dimon/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:72H0Gk0f4b+L80h0xFU1TJ3/4lkIZDH2RJivPQf8kmk Dmitry Evselev 1032209458@pfur.ru>
The key's randomart image is:
+---[RSA 3072]-----+
|
|  ==O*|
|  .+=00+|
|  o oo*+|
|  +.E.+|
|  S . .*0=+|
|  o +..000|
|  =000.+.|
|  o.+= +.|
|  +o o o.|
+----[SHA256]-----+


```

Рис. 5: Рис. 5. Генерирование ключа.

SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.



key

SHA256:72H0Gk0f4b+L80h0xFU1TJ3/4lkIZDH2RJivPQf8kmk

Added on 9 May 2021

Last used within the last week — Read/write

Delete

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

Рис. 6: Рис. 6. Указание ключа на github

```

dimon@vostro:~/tutorial$ git remote add origin ssh://git@github.com/dgevseljev/lab02.git
dimon@vostro:~/tutorial$ git push -u origin master
The authenticity of host 'github.com (140.82.121.3)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWGL7E1IGOCspRomTxdCARLviKw6E5SY8.
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: 3, готово.
Подсчет объектов: 100% (3/3), готово.
Запись объектов: 100% (3/3), 219 bytes | 219.00 KiB/s, готово.
Всего 3 (изменения 0), повторно использовано 0 (изменения 0)
To ssh://github.com/dgevseljev/lab02.git
 * [new branch]      master -> master
Ветка «master» отслеживает внешнюю ветку «master» из «origin».

```

Рис. 7: Рис. 7. Загрузка локального репозитория на сервер

```

dimon@vostro:~/tutorial$ echo "#Labs" >> README.md
dimon@vostro:~/tutorial$ git add README.md
dimon@vostro:~/tutorial$ git commit -m "first commit"
[master 79ab100] first commit
1 file changed, 1 insertion(+)
 create mode 100644 README.md
dimon@vostro:~/tutorial$ git status
На ветке master
Ваша ветка опережает «origin/master» на 1 коммит.
(используйте «git push», чтобы опубликовать ваши локальные коммиты)

Неотслеживаемые файлы:
(используйте «git add <файл>...», чтобы добавить в то, что будет включено в коммит)
.gitignore

ничего не добавлено в коммит, но есть неотслеживаемые файлы (используйте «git add», чтобы отслеживать их)
dimon@vostro:~/tutorial$ git remote add origin git@github.com:dgevseljje/lab02.git
fatal: внешний репозиторий origin уже существует
dimon@vostro:~/tutorial$ git push -u origin master
Warning: Permanently added the RSA host key for IP address '140.82.121.4' to the list of known hosts.
Перечисление объектов: 4, готово.
Подсчет объектов: 100% (4/4), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 283 bytes | 283.00 KiB/s, готово.
Всего 3 (изменения 0), повторно использовано 0 (изменения 0)
To ssh://github.com:dgevseljje/lab02.git
 3022ac0..79ab100  master -> master
Ветка «master» отслеживает внешнюю ветку «master» из «origin».

```

Рис. 8: Рис. 8. Добавление README.md в репозиторий на нитхабе

```

dimon@vostro:~/tutorial$ wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O LICENSE
--2021-05-09 17:19:43-- https://creativecommons.org/licenses/by/4.0/legalcode.txt
Распознаётся creativecommons.org (creativecommons.org)... 2606:4700:10::6814:9610, 2606:4700:10::ac43:228c, 2606:4700:10::6814:9710, ...
Подключение к creativecommons.org (creativecommons.org)[2606:4700:10::6814:9610]:443... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: нет данных [text/plain]
Сохранение в: «LICENSE»

LICENSE [ <=> ] 18,22K --.-KB/s за 0s
2021-05-09 17:19:45 (36,2 MB/s) - «LICENSE» сохранён [18657]
dimon@vostro:~/tutorial$ curl -L -s https://www.gitignore.io/api/c >> .gitignore
dimon@vostro:~/tutorial$ git add .

```

Рис. 9: Рис. 9.1. Скачивание лицензии и шаблонов игнорируемых файлов

```

dimon@vostro:~/tutorial$ git commit -m '#first configuration'
[master 7287d76] #first configuration
2 files changed, 514 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 LICENSE
dimon@vostro:~/tutorial$ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (4/4), готово.
Запись объектов: 100% (4/4), 6.48 KiB | 6.48 MiB/s, готово.
Всего 4 (изменения 0), повторно использовано 0 (изменения 0)
To ssh://github.com:dgevseljje/lab02.git
 79ab100..7287d76  master -> master

```

Рис. 10: Рис. 9.2. Коммит и отправка на github

```

dimon@vostro:~/tutorial$ git flow init

Which branch should be used for bringing forth production releases?
  - master
Branch name for production releases: [master]
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? []
Hooks and filters directory? [/home/dimon/tutorial/.git/hooks]
dimon@vostro:~/tutorial$ git branch
* develop
  master

```

Рис. 11: Рис. 10. Инициализация git-flow

```

dimon@vostro:~/tutorial$ 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'

```

Рис. 12: Рис. 11. Старт релиза версии 1.0.0

```

dimon@vostro:~/tutorial$ git flow release finish 1.0.0
Branches 'master' and 'origin/master' have diverged.
And local branch 'master' is ahead of 'origin/master'.
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 d4685aa).

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

```

Рис. 13: Рис. 12. Финиш релиза версии 1.0.0

```

dimon@vostro:~/tutorial$ git push --all
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), 491 bytes | 491.00 KiB/s, готово.
Всего 5 (изменения 3), повторно использовано 0 (изменения 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To ssh://github.com:dgevseljjevlab02.git
  7287d76..84746fc master -> master
* [new branch]      develop -> develop
dimon@vostro:~/tutorial$ 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 ssh://github.com:dgevseljjevlab02.git
* [new tag]         1.0.0 -> 1.0.0

```

Рис. 14: Рис. 13. Отправка данных на github

Программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями 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.

2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

Хранилище – место где хранятся файлы после внесения изменений,

commit – сохранение добавленных изменений,

история – история изменений файлов,

рабочая копия – копия, которая находится в работе.

3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS?

Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере. Доступ к нему осуществляется через специальное клиентское приложение.

Децентрализованные системы контроля версий позволяют хранить репозиторий (его копию) у каждого разработчика, работающего с данной системой.

4. Опишите действия с VCS при единоличной работе с хранилищем.

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 не полную версию изменённых файлов, а производить так называемую дельтакомпрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

5. Опишите порядок работы с общим хранилищем VCS.

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.

6. Каковы основные задачи, решаемые инструментальным средством git?

Git даёт возможность разработчикам отслеживать изменения в файлах и работать над одним проектом совместно с коллегами.

7. Назовите и дайте краткую характеристику командам 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`
- 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`
- 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. Приведите примеры использования при работе с локальным и удалённым репозиториями.

Если я забыл, в какой ветке нахожусь, то с помощью `git branch` могу посмотреть это. Если мне нужно подключить систему контроля версий к уже существующему проекту, то я инициализирую локальный репозиторий `git init` и подключаю удаленный `git remote add`, затем добавляю все файлы `git add` и коммичу их `git commit`, затем пушу на удаленный репозиторий `git push`. Теперь к моему проекту подключена система контроля версий

9. Что такое и зачем могут быть нужны ветви (branches)?

Ветки нужны для разделения разработки. Например, когда разрабатывается новая фича, не нужно, чтобы она присутствовала в основном проекте, поэтому для нее создают отдельную ветку. В случае успешной разработки фичи, эту ветку сливают с основной. Так убираются риски багов, ошибок, а также утечки данных

10. Как и зачем можно игнорировать некоторые файлы при `commit`?

Есть временные и системные файлы, которые засоряют проект и не нужны. Путь к ним можно добавить в файл `.gitignore`, тогда они не будут добавляться в проект.