

# Отчет по лабораторной работе №2

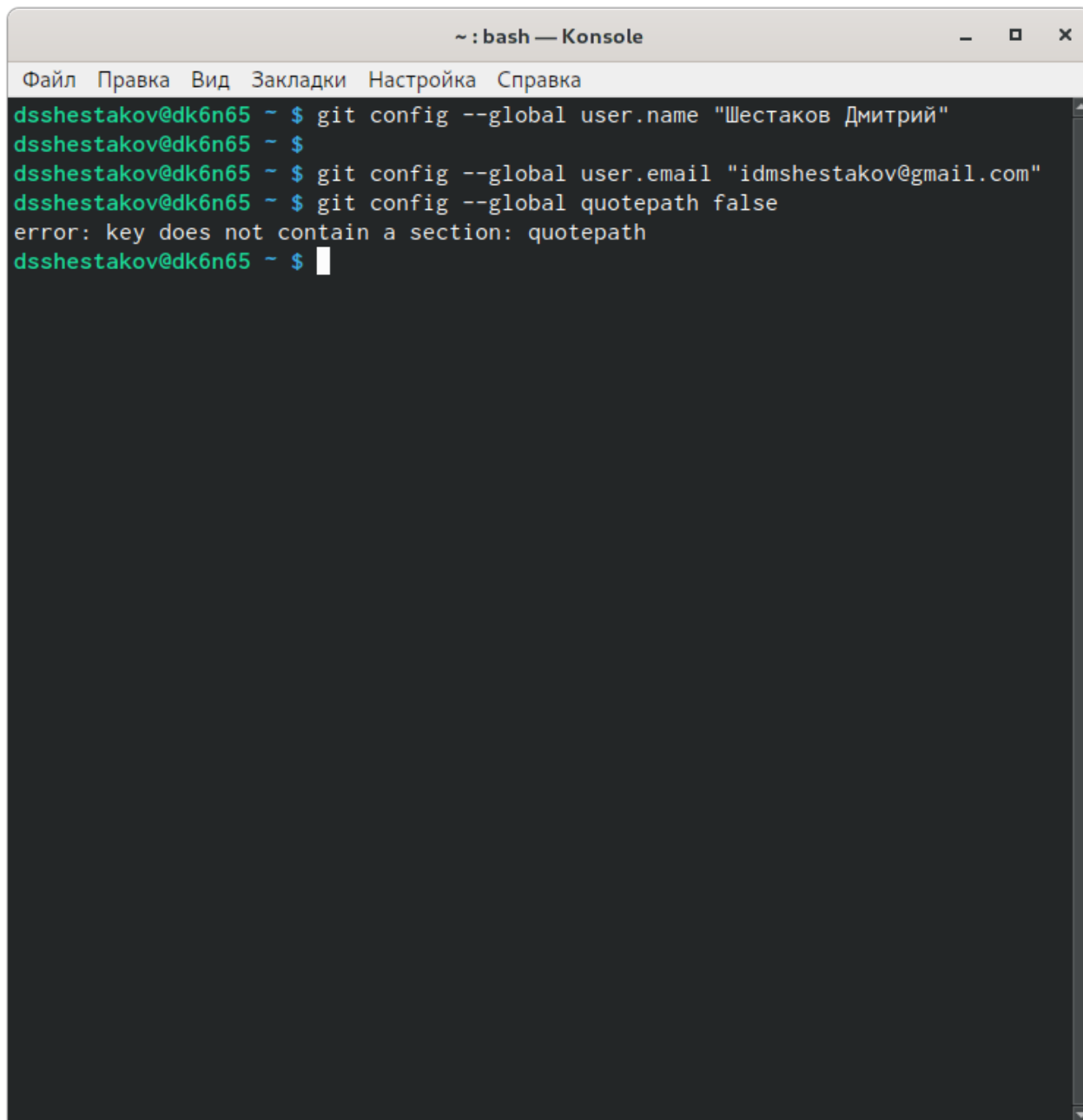
---

## Цель работы

Изучить идеологию и применение средств контроля версий.

## Ход работы

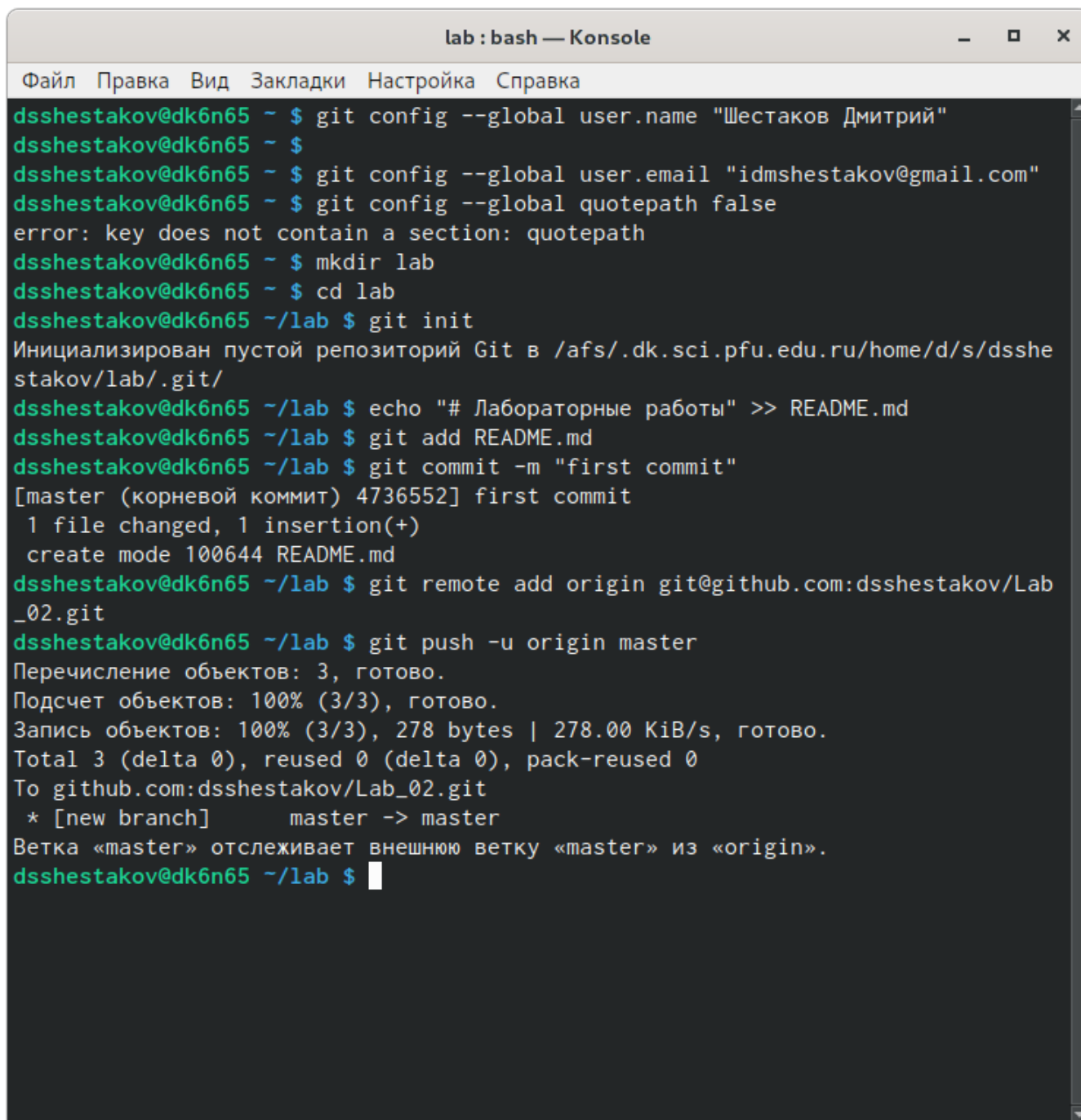
1. Создали учетную запись на github.com, настроили git и создали структуру каталога



```
~ : bash — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
dsshestakov@dk6n65 ~ $ git config --global user.name "Шестаков Дмитрий"
dsshestakov@dk6n65 ~ $
dsshestakov@dk6n65 ~ $ git config --global user.email "idmshestakov@gmail.com"
dsshestakov@dk6n65 ~ $ git config --global quotepath false
error: key does not contain a section: quotepath
dsshestakov@dk6n65 ~ $
```

2. Создали репозиторий на github.com, назвали его Lab\_02
3. Перешли в рабочий каталог
4. Инициализировали систему контроля версий `git init`
5. Создали заготовку файла README.md `echo "Лабораторная работа №2" >> README.md`  
`git add README.md`
6. Сделали первый коммит и выложили на github

```
git commit -m "first commit"
git remote add origin git@github.com:dsshestakov/Lab_02
git push -u origin master
```



```
lab : bash — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
dsshestakov@dk6n65 ~ $ git config --global user.name "Шестаков Дмитрий"
dsshestakov@dk6n65 ~ $
dsshestakov@dk6n65 ~ $ git config --global user.email "idmshestakov@gmail.com"
dsshestakov@dk6n65 ~ $ git config --global quotepath false
error: key does not contain a section: quotepath
dsshestakov@dk6n65 ~ $ mkdir lab
dsshestakov@dk6n65 ~ $ cd lab
dsshestakov@dk6n65 ~/lab $ git init
Инициализирован пустой репозиторий Git в /afs/.dk.sci.pfu.edu.ru/home/d/s/dsshe
stakov/lab/.git/
dsshestakov@dk6n65 ~/lab $ echo "# Лабораторные работы" >> README.md
dsshestakov@dk6n65 ~/lab $ git add README.md
dsshestakov@dk6n65 ~/lab $ git commit -m "first commit"
[master (корневой коммит) 4736552] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md
dsshestakov@dk6n65 ~/lab $ git remote add origin git@github.com:dsshestakov/Lab
_02.git
dsshestakov@dk6n65 ~/lab $ git push -u origin master
Перечисление объектов: 3, готово.
Подсчет объектов: 100% (3/3), готово.
Запись объектов: 100% (3/3), 278 bytes | 278.00 KiB/s, готово.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:dsshestakov/Lab_02.git
 * [new branch]      master -> master
Ветка «master» отслеживает внешнюю ветку «master» из «origin».
dsshestakov@dk6n65 ~/lab $
```

7. Добавили файл лицензии `wget`

`https://creativecommons.org/licenses/by/4.0/legalcode.txt -O LICENSE` 8. Скачали шаблон для C `curl -L -s https://www.gitignore.io/api/c >> .gitignore` 9. Добавили новые файлы, выполнили коммит, отправили на github

```
git add .
git commit -a
git push
```

```
lab : bash — Konsole
Файл Правка Вид Закладки Настройка Справка
var screenReaderText = {"expand":"expand child menu","collapse":"collapse child menu"};
</script>
<script src='https://d15omoko64skxi.cloudfront.net/wp-content/themes/twentyseventeen/js/functions.js?ver=20181217' id='twentyseventeen-script-js'></script>
<script src='https://d15omoko64skxi.cloudfront.net/wordpress/wp-includes/js/wp-embed.min.js?ver=5.5' id='wp-embed-js'></script>
<script type='text/javascript' src='https://stats.wp.com/e-202117.js' async='async' defer='defer'></script>
<script type='text/javascript'>
    _stq = window._stq || [];
    _stq.push([ 'view', {v:'ext',j:'1:8.8.2',blog:'78836240',post:'0',tz:'-7',srv:'creativecommons.org'} ]);
    _stq.push([ 'clickTrackerInit', '78836240', '0' ]);
</script>
</body>
</html>
dsshestakov@dk6n65 ~/lab $ curl -L -s https://www.gitignore.io/api/c >> .gitignore
ore
dsshestakov@dk6n65 ~/lab $ git add .
dsshestakov@dk6n65 ~/lab $ git commit -a
[master 08f6633] commit
 2 files changed, 455 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 LICENSE
dsshestakov@dk6n65 ~/lab $ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (4/4), готово.
Запись объектов: 100% (4/4), 6.46 KiB | 6.46 MiB/s, готово.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:dsshestakov/Lab_02.git
 4736552..08f6633 master -> master
dsshestakov@dk6n65 ~/lab $
```

10. Инициализировали gitflow `git flow init`

```
lab : bash — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
dsshestakov@dk6n65 ~/lab $ curl -L -s https://www.gitignore.io/api/c >> .gitignore
ore
dsshestakov@dk6n65 ~/lab $ git add .
dsshestakov@dk6n65 ~/lab $ git commit -a
[master 08f6633] commit
 2 files changed, 455 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 LICENSE
dsshestakov@dk6n65 ~/lab $ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (4/4), готово.
Запись объектов: 100% (4/4), 6.46 KiB | 6.46 MiB/s, готово.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:dsshestakov/Lab_02.git
 4736552..08f6633  master -> master
dsshestakov@dk6n65 ~/lab $ 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/] v
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [/afs/.dk.sci.pfu.edu.ru/home/d/s/dsshestakov/lab/.git/hooks]
dsshestakov@dk6n65 ~/lab $ git branch
dsshestakov@dk6n65 ~/lab $
```

11. Проверили, что мы на ветке develop `git branch`



14. Залили релизную ветку в основную ветку `git flow release finish 1.0.0`

```
git push --all
git push --tags
```

```
lab : bash — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
dsshestakov@dk6n65 ~/lab $ 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'

dsshestakov@dk6n65 ~/lab $ echo "1.0.0" >> VERSION
dsshestakov@dk6n65 ~/lab $ git add .
dsshestakov@dk6n65 ~/lab $ git commit -am 'chore(main): add version'
[release/1.0.0 1241ac7] chore(main): add version
1 file changed, 1 insertion(+)
create mode 100644 VERSION
dsshestakov@dk6n65 ~/lab $ git flow release finish 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)
fatal: нет описания метки?
Fatal: Tagging failed. Please run finish again to retry.
dsshestakov@dk6n65 ~/lab $
```

## Контрольные вопросы

1. **Система контроля версий** (Version Control System, VCS) представляет собой программное обеспечение, которое позволяет отслеживать изменения в документах, при необходимости производить их откат, определять, кто и когда внес исправления и т. п.
2. **Репозиторий** – специальное хранилище файлов и папок проекта, изменения в которых отслеживаются. **Коммит** – операция отправки в репозиторий изменений, которые пользователь внес в свою рабочую копию. **История** – все коммиты совершенные к данному моменту. **Рабочая копия** – версия проекта, с которой работает разработчик.
3. Централизованная система контроля версий(CVS, Subversion) – репозиторий существует в единственном экземпляре и хранится на сервере. Распределенная система контроля версий (Git) – у каждого разработчика есть собственный репозиторий, при этом существует условно центральный репозиторий, куда будут отправляться изменения из локальных

4. При работе с общим хранилище,, надо собственную ветку. Затем надо получить информацию об изменениях из центрального репозитория. После можно вносить свои изменения. Для того, чтобы отправить изменения в центральный репозитория, надо проверить какие файлы притерпели изменения и при необходимости удалить какие-то файлы, которые не должны быть отправлены в центральный репозиторий. Затем создаем коммит и загружаем файлы в центарльный репозиторий.
5. Возврат к ранним версиям проектам, отслеживание истории изменений, устраняет опасность удаление чужих файлов и дает возможность не бояться за потерю данных.
6. 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` – 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 с текущим 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`
7. При помощи веток в VCS можно:

Реализовать фичу, не мешая остальным. Проводить модерацию (кодревью) нового кода перед непосредственным добавлением в кодовую базу. Отвлечься от реализации фичи и починить баг в другом месте. Вовсе отложить начатую фичу до лучших времен. Получить запрос на доработку старой версии программы от заказчика и поддерживать далее несколько версий ПО.

Поэкспериментировать с кодом без страха сломать билд. Организовать процесс поэтапного выпуска программы (разработка - тестирование - релиз), не блокируя разработку следующей версии. Организовать работу с open source сообществом или подрядчиком. Запилить постоянную автоматическую сборку с рабочей ветки с прогоном тестов и ручную авторизованную сборку релиза с релиз-ветки.