

Отчет по лабораторной работе № 2

Цель работы

Изучить идеологию и применение средств контроля версий.

Ход работы

- . Создали учетную запись на github.com ,настроили `git` создали структуру каталога

```
~ : bash — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
CalixteGiordannyJo@junior ~ $ git config --global user.name "Jonathan Junior"
CalixteGiordannyJo@junior ~ $
CalixteGiordannyJo@junior ~ $ git config --global user.email "orderownitcontact@gmail.com"
CalixteGiordannyJo@junior ~ $ git config --global quotepath false
error: key does not contain a section: quotepath
CalixteGiordannyJo@junior ~ $
```

Создали репозиторий на github.com ,назвали его `Lab_02`

- . Перешли в рабочий каталог `git init`
- . Инициализировали систему контроля версий
- . Создали заготовку файла `README.md` `echo "Лабораторная работа №2" >> README.md` `git add README.md`
- . Сделали первый коммит и выложили на [github](https://github.com)

Файл Правка Вид Закладки Настройка Справка

```
CalixteGiordannyJo@junior ~ $ git config --global user.name "Jonathan Junior"
CalixteGiordannyJo@junior ~ $
CalixteGiordannyJo@junior ~ $ git config --global user.email "orderownitcontact@gmail.com"
CalixteGiordannyJo@junior ~ $ git config --global quotepath false
error: key does not contain a section: quotepath
CalixteGiordannyJo@junior ~ $ mkdir lab
CalixteGiordannyJo@junior ~ $ cd lab
CalixteGiordannyJo@junior ~/lab $ git init
Инициализирован пустой репозиторий Git в /afs/.dk.sci.pfu.edu.ru/home/d/s/Jonathan929/lab/.git/
CalixteGiordannyJo@junior ~/lab $ echo "# Лабораторные работы" >> README.md
CalixteGiordannyJo@junior ~/lab $ git add README.md
CalixteGiordannyJo@junior ~/lab $ git commit -m "first commit"
[master (корневой коммит) 4736552] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md
CalixteGiordannyJo@junior ~/lab $ git remote add origin git@github.com: Jonathan929/Lab_02.git
CalixteGiordannyJo@junior ~/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: Jonathan929/Lab_02.git
* [new branch]      master -> master
Ветка «master» отслеживает внешнюю ветку «master» из «origin».
CalixteGiordannyJo@junior ~/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
```

```
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/twentysext
  een/js/functions.js?ver=20181217' id='twentysexteen-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='as
  ync' defer='defer'></script>
  Jonathan929
  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>
CalixteGiordannyJo@junior ~/lab $ curl -L -s https://www.gitignore.io/api/c >> .gitign

CalixteGiordannyJo@junior ~/lab $ git add .
CalixteGiordannyJo@junior ~/lab $ git commit -a
[master 08f6633] commit
 2 files changed, 455 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 LICENSE
CalixteGiordannyJo@junior ~/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: Jonathan929/Lab_02.git
 4736552..08f6633 master -> master
CalixteGiordannyJo@junior ~/lab $
```

10.Инициализировали gitflow `git flow init`

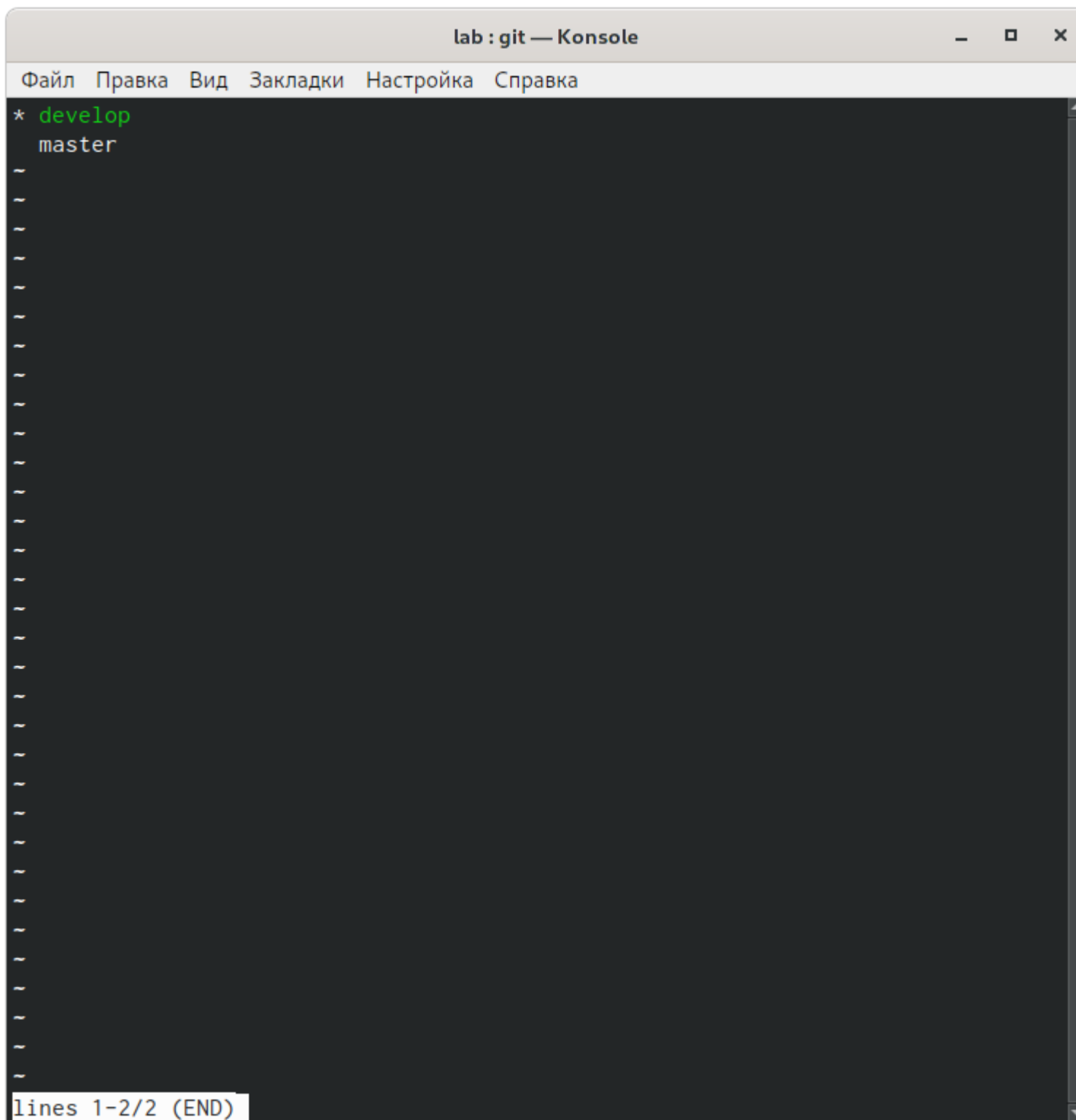

```
lab : bash — Konsole
Файл Правка Вид Закладки Настройка Справка
CalixteGiordannyJo@junior~/lab $ curl -L -s https://www.gitignore.io/api/c >> .gitignore
CalixteGiordannyJo@junior~/lab $ git add .
CalixteGiordannyJo@junior~/lab $ git commit -a
[master 08f6633] commit
2 files changed, 455 insertions(+)
create mode 100644 .gitignore
create mode 100644 LICENSE
CalixteGiordannyJo@junior~/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
CalixteGiordannyJo@junior~/lab $ git flow init

Which branch should be used for bringing forth production releases?

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/ Jonathan929 /lab/.git/hooks] Jonathan929
CalixteGiordannyJo@junior~/lab $ git branch
CalixteGiordannyJo@junior~/lab $
```

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



12.Создали релиз 1.0.0 `git flow release start 1.0.0` **13.Добавили индекс**

```
git add .
git commit -am "chore(main): add version"
```

. Залили релизную ветку в основную ветку `git flow release finish 1.0.0` **. Отправили данные на github**

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

```
lab : bash — Konsole
Файл Правка Вид Закладки Настройка Справка
CalixteGiordannyJo@junior~/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'

CalixteGiordannyJo@junior~/lab $ echo "1.0.0" >> VERSION
CalixteGiordannyJo@junior~/lab $ git add .
CalixteGiordannyJo@junior~/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
CalixteGiordannyJo@junior~/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.
CalixteGiordannyJo@junior~/lab $
```

Контрольные вопросы

- 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.
- 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.
- 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

.При работе с общ им хранилищ е,,надо собственную ветку.Затем надо получитьинформацию об измененияхизцентрального репозитория.После можно вноситьсвои изменения.Длятого, чтобы отправитьизмененияв центральный репозитория,надо проверитькакие файлы притерпели измененияи при необходимости удалитькакие-то файлы,которые не должны бытьотправлены в центральный репозиторий.Затем создаем коммити загружаем файлы в центарльный репозиторий.

.Возвраткранним версиям проектам,отслеживание истории изменений,устраняетопасность удаление чужихфайлов и даетвозможностьне боятьсяза потерю данных.

. Наиболее часто используемые команды git:– создание основного дерева репозитория:gitinit

– получение обновлений(изменений)текущ его дерева из центрального репозитория:gitpull–

отправка всех произведённых изменений локального дерева в центральный репозиторий:git

push – просмотр списка изменённых файлов в текущ ей директории:gitstatus – просмотр

текущ их изменения:gitdiff – сохранение текущ их изменений:– добавить все изменённые и/

или созданные файлы и/или каталоги:gitadd .– добавить конкретные изменённые и/или

созданные файлы и/или каталоги:gitadd имена_файлов – удалить файл и/или каталог из

индекса репозитория(при этом файл и/или каталог остаётся в локальной директории):gitrm

имена_файлов – сохранение добавленныхизменений:– сохранитьвсе добавленные

изменения и все изменённые файлы:gitcommit -m 'Описание коммита'– сохранить

добавленные изменения с внесением комментария через встроенный редактор:gitcommit -m –

создание новой ветки,базирующ ейся на текущ ей:gitcheckout -b имя_ветки – переключение на

некоторую ветку:gitcheckoutимя_ветки (при переключении на ветку,которой ещ ё нет в локальном репозитории,она будетсоздана и связана с удалённой)– отправка изменений конкретной ветки в

центральный репозиторий:gitpush origin имя_ветки – слияние ветки с текущ им деревом:gitmerge --no-ff имя_ветки – удаление ветки:–

удаление локальной уже слитой с основным деревом ветки:gitbranch -d имя_ветки – принудительное удаление локальной ветки:gitbranch -D

имя_ветки – удаление ветки с центрального репозитория: `git push origin :имя_ветки`

.При помощи веток в VCS можно:

Реализовать фичу, не мешая остальным. Проводить модерацию (код-ревью) нового кода перед

непосредственным добавлением в кодовую базу. Отвлечься от реализации фичи и починить баг в

другом месте. Во все 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 ей

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.