

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ»**

Факультет физико-математических и естественных наук

Кафедра информационных технологий

**ОТЧЕТ**  
**по лабораторной работе №2**  
по дисциплине «Операционные системы»

Выполнил:

Студент группы НПИбд-02-21

Студенческий билет № 1032211220

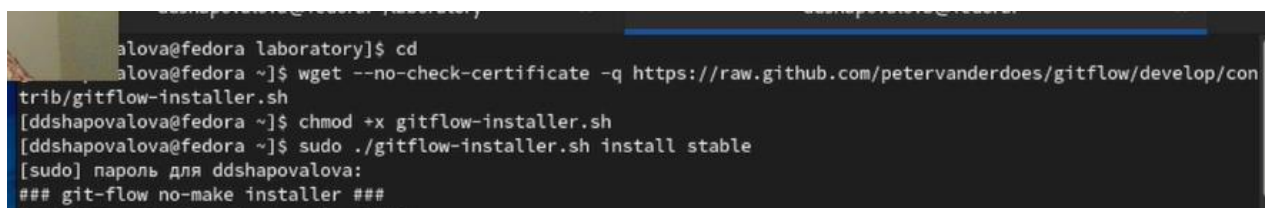
Шаповалова Диана Дмитриевна

**Цель работы:** Изучить идеологию и применение средств контроля версий.

Освоить умения по работе с git.

**Ход работы:** Устанавливаем git-flow в Fedora Linux.

```
wget --no-check-certificate -q  
https://raw.githubusercontent.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh  
  
chmod +x gitflow-installer.sh  
  
sudo ./gitflow-installer.sh install stable
```

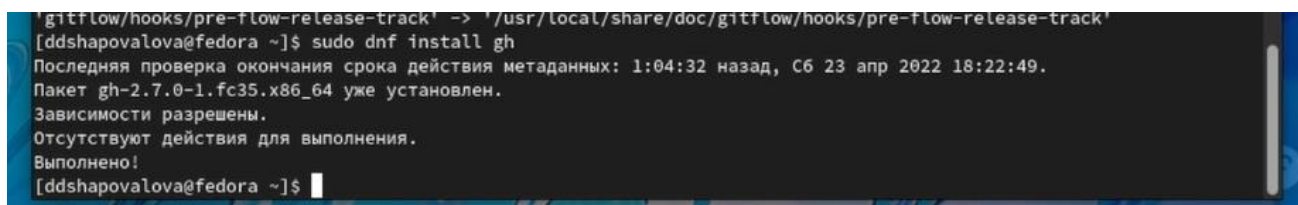


```
ddshapovalova@fedora laboratory]$ cd  
ddshapovalova@fedora ~]$ wget --no-check-certificate -q https://raw.githubusercontent.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh  
[ddshapovalova@fedora ~]$ chmod +x gitflow-installer.sh  
[ddshapovalova@fedora ~]$ sudo ./gitflow-installer.sh install stable  
[sudo] пароль для ddshapovalova:  
## git-flow no-make installer ##
```

*Рис.1 Установка git-flow*

Далее мы устанавливаем gh.

```
sudo dnf install gh
```



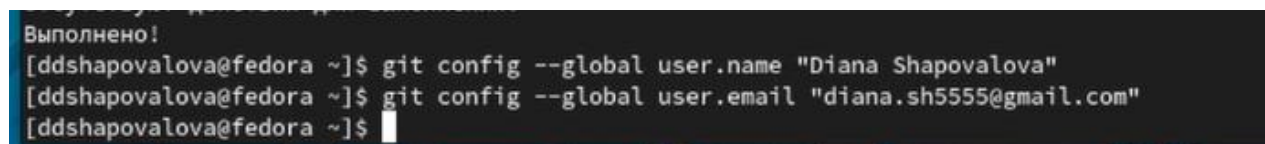
```
'gitflow/hooks/pre-flow-release-track' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-release-track'  
[ddshapovalova@fedora ~]$ sudo dnf install gh  
Последняя проверка окончания срока действия метаданных: 1:04:32 назад, Сб 23 апр 2022 18:22:49.  
Пакет gh-2.7.0-1.fc35.x86_64 уже установлен.  
Зависимости разрешены.  
Отсутствуют действия для выполнения.  
Выполнено!  
[ddshapovalova@fedora ~]$
```

*Рис.2 Установка gh.*

Переходим к базовой настройке git. Зададим имя и email владельца репозитория

```
git config --global user.name "Name Surname"
```

```
git config --global user.email "work@mail"
```



```
Выполнено!  
[ddshapovalova@fedora ~]$ git config --global user.name "Diana Shapovalova"  
[ddshapovalova@fedora ~]$ git config --global user.email "diana.sh5555@gmail.com"  
[ddshapovalova@fedora ~]$
```

*Рис. 3 Задаем имя и почту владельца*

Создаем ключи ssh и gpg. Прикрепляем к github.

```
ssh-keygen -t rsa -b 4096
```

```
gpg --full-generate-key
```

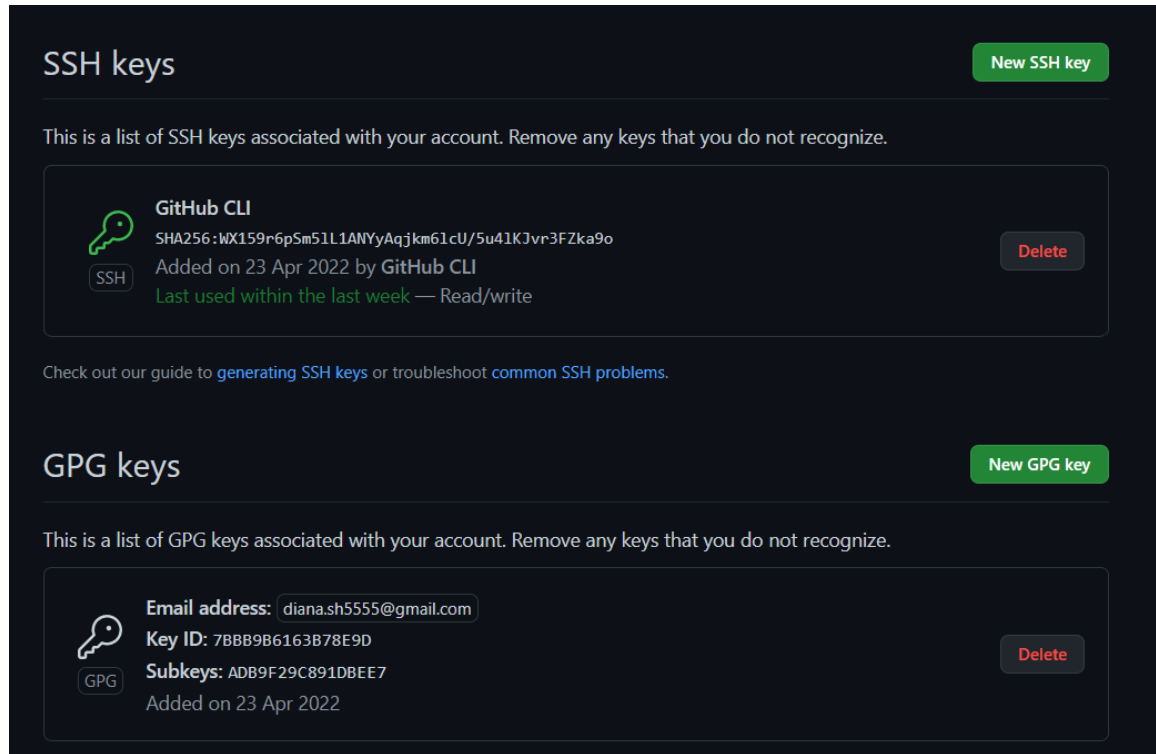


Рис.4 Ключи ssh и gpg.

Выполняем настройку gh.

```
gh auth login
```

Создаем репозиторий курса на основе шаблона.

```
mkdir -p ~/work/study/2021-2022/"Операционные системы"
```

```
cd ~/work/study/2021-2022/"Операционные системы"
```

```
gh repo create study_2021-2022_os-intro --template=yamadharma/course-directory-student-template --public
```

```
git clone --recursive git@github.com:<owner>/study_2021-2022_os-intro.git os-intro
```

Переходим в каталог, удаляем файл package.json, создаем каталог, добавляем файлы на сервер.

```
cd ~/work/study/2021-2022/"Операционные системы"/os-intro
```

```
rm package.json
```

```
make COURSE=os-intro
```

```
git add .
```

```
git commit -am 'feat(main): make course structure'
```

```
git push
```

```
[ddshapovalova@fedora ~]$ mkdir -p ~/work/study/2021-2022/"Операционные системы"
[ddshapovalova@fedora ~]$ cd ~/work/study/2021-2022/"Операционные системы"
[ddshapovalova@fedora Операционные системы]$ gh repo create study_2021-2022_os-intro --template=yamadharma/course-d
irectory-student-template --public
✓ Created repository ddshapovalova/study_2021-2022_os-intro on GitHub
[ddshapovalova@fedora Операционные системы]$ 
[ddshapovalova@fedora Операционные системы]$ cd ~/work/study/2021-2022/"Операционные системы"/os-intro
[ddshapovalova@fedora os-intro]$ rm package.json
[ddshapovalova@fedora os-intro]$ make COURSE=os-intro
[ddshapovalova@fedora os-intro]$ git add .
[ddshapovalova@fedora os-intro]$ git commit -am 'feat(main): make course structure'
[master cbfc601] feat(main): make course structure
149 files changed, 16590 insertions(+), 14 deletions(-)
create mode 100644 labs/lab01/presentation/Makefile
create mode 100644 labs/lab01/presentation/presentation.md
create mode 100644 labs/lab01/report/Makefile
create mode 100644 labs/lab01/report/bib/cite.bib
create mode 100644 labs/lab01/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab01/report/pandoc/csl/gost-r-7-0-5-2008-russian.numeric.csl
create mode 100644 labs/lab01/report/report.md
create mode 100644 labs/lab02/presentation/Makefile
create mode 100644 labs/lab02/presentation/presentation.md
create mode 100644 labs/lab02/report/Makefile
create mode 100644 labs/lab02/report/bib/cite.bib
create mode 100644 labs/lab02/report/image/placeimg_800_600_tech.jpg
[ddshapovalova@fedora os-intro]$ git push
Перечисление объектов: 20, готово.
Подсчет объектов: 100% (20/20), готово.
Сжатие объектов: 100% (16/16), готово.
Запись объектов: 100% (19/19), 265.88 КиБ | 1.98 МиБ/с, готово.
Всего 19 (изменений 2), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To github.com:ddshapovalova/study_2021-2022_os-intro.git
   3c23f62..cbfc601  master -> master
[ddshapovalova@fedora os-intro]$
```

Рис.5 Настраиваем репозиторий.

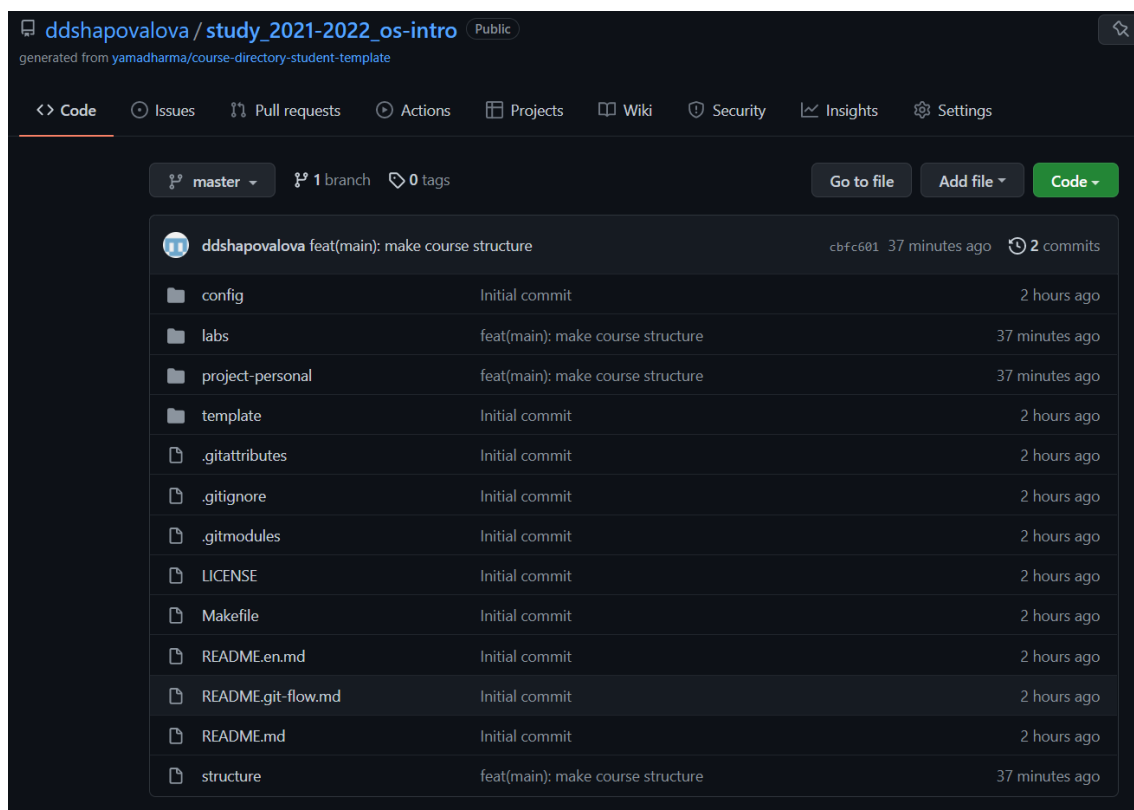


Рис.6 Проверяем репозиторий

**Вывод:** Мы успешно выполнили работу, у нас вышло создать репозиторий для дальнейших лабораторных работ и освоили умения по работе с git.

## Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.
4. Опишите действия с VCS при единоличной работе с хранилищем.
5. Опишите порядок работы с общим хранилищем VCS.
6. Каковы основные задачи, решаемые инструментальным средством git?
7. Назовите и дайте краткую характеристику командам git.
8. Приведите примеры использования при работе с локальным и удалённым репозиториями.

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

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

### Ответы:

1. Version Control System, VCS или Revision Control System) — программное обеспечение для облегчения работы с изменяющейся информацией.

Это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии.

2. - *Хранилище* — это содержимое скрытой папки .git. В этой папке хранятся все версии рабочей области и служебная информация. Этим версиям система автоматически даёт название, состоящее из букв и цифр.

- Команда *commit* позволяет сохранить текущее состояние проекта. В файле с сохранением отображаются: все изменения, которые происходили в рабочей области, автор изменений и краткий комментарий, описывающий суть изменений. Каждый коммит хранит полное состояние рабочей области, её папок и файлов проекта.

- История — история изменений. Обычно доступна информация о том, кто из участников, когда и какие изменения вносил.

- Рабочая копия - копия проекта, связанная с репозиторием

### 3. Централизованные VCS:

► Одно основное хранилище всего проекта

► Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем добавляет свои изменения обратно.

-*Subversion*

-*CVS*

-*TFS, VAULT*

-*AccuRev*

### Децентрализованные VCS:

► У каждого пользователя свой вариант (возможно не один) репозитория

► Присутствует возможность добавлять и забирать изменения из любого репозитория

- *Git*

- *Mercurial*

- *Bazaar*

4. При единоличной работе с VCS каждое новое изменение в репозитории сохраняется не со всеми предыдущими версиями. Оно изменяется по системе: одно предыдущее + новая информация.

5. -Создать репозиторий (делается разово)

- Скачать проект из репозитория (делается разово)

- Обновить проект

- Внести изменения в проект

- Запустить код

- Создать ветку

6. -Сохранение файлов с исходным кодом

-Защита от случайных исправлений и удалений

- Отмена изменений и удалений, если они некорректны

-Возврат к любой прошлой версии кода

-Просмотр истории изменений

-Исключена возможность потери данных

7. – 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:

`1 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 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. Локальный репозиторий – в ней хранятся коммиты и другие проекты. Используем его, когда работаем в одиночку и нам нужно сохранить свои изменения.

Удаленный репозиторий- тот репозиторий, который считается общим, в который мы можем передать все коммиты из локального репозитория, чтобы остальные пользователи могли их увидеть. Используем для групповой работы, чтобы делиться своими изменениями и скачивать чужие изменения.

9. Ветка- подвижный указатель на один из коммитов. Обычно указывает на последний коммит в цепочке коммитов. В своей ветке мы можем как угодно ломать проект, основной код при этом не пострадает.

10. Игнорируемые файлы- это обычно специфичные для платформы файлы или автоматические созданные файлы из систем сборки.

- Файлы времени выполнения, такие как журнал, блокировка, кэш или временные файлы.
- Файлы с конфиденциальной информацией, такой как пароли или ключи API.
- Скомпилированный код, такой как .class или .o.
- Каталоги зависимостей, такие как /vendor или /node\_modules.
- Создавать папки, такие как /public, /out или /dist.
- Системные файлы, такие как .DS\_Store или Thumbs.db
- Конфигурационные файлы IDE или текстового редактора.

`.gitignore` Шаблоны

`.gitignore` — это простой текстовый файл, в каждой строке которого содержится шаблон, который файлы или каталоги следует игнорировать.

Он использует шаблоны подстановки для сопоставления имен файлов с подстановочными знаками. Если у вас есть файлы или каталоги, содержащие шаблон подстановки, вы можете использовать одиночную обратную косую черту ( `\` ) для экранирования символа.

Местный `.gitignore`

`.gitignore` файл `.gitignore` обычно помещается в корневой каталог репозитория.

Однако вы можете создать несколько файлов `.gitignore` в разных подкаталогах вашего репозитория. Шаблоны в файлах `.gitignore` сопоставляются относительно каталога, в котором находится файл.

Шаблоны, определенные в файлах, которые находятся в каталогах (подкаталогах) более низкого уровня, имеют приоритет над шаблонами в каталогах более высокого уровня. Локальные файлы `.gitignore` используются совместно с другими разработчиками и должны содержать шаблоны, полезные для всех других пользователей репозитория.

### Личные правила игнорирования

Шаблоны, специфичные для вашего локального репозитория и не подлежащие распространению в другие репозитории, должны быть установлены в файле `.git/info/exclude`.

Например, вы можете использовать этот файл, чтобы игнорировать файлы, сгенерированные из ваших личных инструментов проекта.

### Глобальный `.gitignore`

Git также позволяет вам создать глобальный файл `.gitignore`, в котором вы можете определить правила игнорирования для каждого репозитория Git в вашей локальной системе.

Файл можно назвать как угодно и хранить в любом месте. Чаще всего этот файл хранится в домашнем каталоге. Вам придется вручную создать файл и настроить Git для его использования.