

# Отчёта по лабораторной работе №2

Управление версиями

Смирнов-Мальцев Егор Дмитриевич

## Цель работы

- Изучить идеологию и применение средств контроля версий.
- Освоить умения по работе с git.

# Задание

- Создать базовую конфигурацию для работы с git.
- Создать ключ SSH.
- Создать ключ PGP.
- Настроить подписи git.
- Зарегистрироваться на Github.
- Создать локальный каталог для выполнения заданий по предмету.

# Выполнение лабораторной работы

1. Создал учетную запись на Github и заполнил основные данные.

Let's begin the adventure

**Enter your email**  
✓ mathanalime@gmail.com

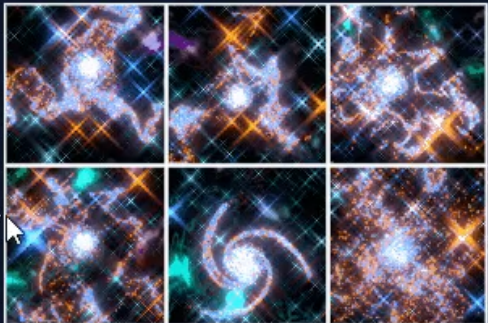
**Create a password**  
✓ .....

**Enter a username**  
✓ EgorSmM

**Would you like to receive product updates and announcements via email?**  
Type "y" for yes or "n" for no  
✓ n

**Verify your account**

Выберите спиральную галактику



⏮ ⏭

Рис. 1: Страница регистрации на Github

2. Установил Git-flow и gh на виртуальной машине.

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

Установка git-flow

```
[edsmirnovmaljce@edsmirnovmaljce ~]$ sudo dnf install gh
[sudo] пароль для edsmirnovmaljce:
```

Установка gh

3. Настроил git. (рис. [-@fig:004], [-@fig:005])

```
[edsmirnovmaljce@edsmirnovmaljce tmp]$ git config --global core.quotePath false
[edsmirnovmaljce@edsmirnovmaljce tmp]$ git config --global init.defaultBranch master
[edsmirnovmaljce@edsmirnovmaljce tmp]$ git config --global core.autocrlf input
```

Создание основной ветки

```
[edsmirnovmaljce@edsmirnovmaljce ~]$ git config --global user.signingkey 71491FC5CB026826
[edsmirnovmaljce@edsmirnovmaljce ~]$ git config --global commit.gpgsign true
[edsmirnovmaljce@edsmirnovmaljce ~]$ git config --global gpg.program $(which gpg2)
[edsmirnovmaljce@edsmirnovmaljce ~]$
```

Унифицирование коммитов

4. Создал 2 ssh ключа: по алгоритму rsa размером 4096 бит и по алгоритму ed25519.

```

[edsmirnovmaljce@edsmirnovmaljce tmp]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/edsmirnovmaljce/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/edsmirnovmaljce/.ssh/id_ed25519
Your public key has been saved in /home/edsmirnovmaljce/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:BJbd1NTUbtD9GtCv5Af0YqG//ja5d/vSswu5bPSoxB4 edsmirnovmaljce@edsmirnovmalj
ce
The key's randomart image is:
+--[ED25519 256]--+
|      oo o.o.+o.o.|
|      .... . o * +|
|      .      + * .|
|      .      . * *|
|      S      = B |
|      .      . * .|
|      E.oo+.|
|      o oo===|
|      oo+.o&|
+-----[SHA256]-----+

```

Создание ключа по алгоритму ed25519

```

[edsmirnovmaljce@edsmirnovmaljce tmp]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/edsmirnovmaljce/.ssh/id_rsa):
Created directory '/home/edsmirnovmaljce/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/edsmirnovmaljce/.ssh/id_rsa
Your public key has been saved in /home/edsmirnovmaljce/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:dPzXNb5jwI9tkTKYna5CnrytslQHg442DjPWxFsvb6A edsmirnovmaljce@edsmirnovmaljce
The key's randomart image is:
+---[RSA 4096]---+
|
|      .  ..
|      o o.oo   ..|
|      o =...o.= o.+|
|      = * +So +.B.+.|
|      . * o =.. ..B o|
|      E .+o.  o B |
|      ...=. . o .|
|      .oo+o
|
+-----[SHA256]-----+

```

Создание ключа по алгоритму rsa размером 4096 бит

5. Сгенерировал ргр-ключ.



```
[edsmirnovmaljce@edsmirnovmaljce ~]$ gpg --full-generate-key
gpg (GnuPG) 2.3.2; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Выберите тип ключа:
  (1) RSA and RSA
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (9) ECC (sign and encrypt) *default*
 (10) ECC (только для подписи)
 (14) Existing key from card
Ваш выбор? 1
длина ключей RSA может быть от 1024 до 4096.
Какой размер ключа Вам необходим? (3072) 4096
Запрошенный размер ключа - 4096 бит
Выберите срок действия ключа.
  0 = не ограничен
  <n> = срок действия ключа - n дней
  <n>w = срок действия ключа - n недель
  <n>m = срок действия ключа - n месяцев
  <n>y = срок действия ключа - n лет
Срок действия ключа? (0) 0
Срок действия ключа не ограничен
Все верно? (y/N) y

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: EgorSmM
Адрес электронной почты: mathanalime@gmail.com
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "EgorSmM <mathanalime@gmail.com>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? O
```

## Генерация pgp-ключа

6. Скопировал pgp-ключ и ввел его на сайте Github.

```
[edsmirnovmaljce@edsmirnovmaljce ~]$ gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: глубина: 0 достоверных: 3 подписанных: 0 доверие: 0-, 0q, 0n, 0m, 0f, 3u
/home/edsmirnovmaljce/.gnupg/pubring.kbx
-----
sec   rsa4096/C976E6CC4CC6F0BE 2022-04-23 [SC]
      08BF228B061282E50241D394C976E6CC4CC6F0BE
uid           [ абсолютно ] EgorSmM <mathanalime@gmail.com>
ssb   rsa4096/E7F3830193036243 2022-04-23 [E]
```

## Копирование pgp-ключа

7. Скопировал ssh-ключ и ввел его на Github.

```
cat: /home/edsmirnovmaljce/.ssh/id_rsa.pub: нет такого файла или каталога
[edsmirnovmaljce@edsmirnovmaljce ~]$ cat ~/.ssh/id_rsa.pub | xclip -sel clip
[edsmirnovmaljce@edsmirnovmaljce ~]$
```

## Копирование ssh-ключа

8. Указал Git применять ранее введенный email в качестве подписи коммитов.

```
[edsmirnovmaljce@edsmirnovmaljce ~]$ git config --global user.signingkey 71491FC5CB026826
[edsmirnovmaljce@edsmirnovmaljce ~]$ git config --global commit.gpgsign true
[edsmirnovmaljce@edsmirnovmaljce ~]$ git config --global gpg.program $(which gpg2)
[edsmirnovmaljce@edsmirnovmaljce ~]$
```

Настройка автоматических подписей

9. Авторизировался на Github.

```
edsmirnovmaljce@edsmirnovmaljce ~]$ gh auth login
What account do you want to log into? GitHub.com
You're already logged into github.com. Do you want to re-authenticate? Yes
What is your preferred protocol for Git operations? HTTPS
How would you like to authenticate GitHub CLI? Login with a web browser

First copy your one-time code: FCA8-D77D
Press Enter to open github.com in your browser...
estorecon: Could not stat /home/edsmirnovmaljce/.mozilla/firefox/mngi6i5l.default-release/sessionstore-backups/recovery.jsonlz4.tmp: No such file or directory.
estorecon: Could not stat /home/edsmirnovmaljce/.mozilla/firefox/mngi6i5l.default-release/sessionstore-backups/recovery.jsonlz4.tmp: No such file or directory.
Authentication complete.
gh config set -h github.com git_protocol https
Configured git protocol
Logged in as EgorSmM
```

Переход к авторизации через командную строку

10. Создал репозиторий на основе шаблона рабочего пространства.

```
[edsmirnovmaljce@edsmirnovmaljce ~]$ mkdir -p ~/work/study/2021-2022/"Операционные системы"
[edsmirnovmaljce@edsmirnovmaljce ~]$ cd work/study/2021-2022/"Операционные системы"
[edsmirnovmaljce@edsmirnovmaljce Операционные системы]$ gh repo create study_2021-2022_os-intro
```

Создание директории и переход в нее

```
[edsmirnovmaljce@edsmirnovmaljce Операционные системы]$ gh repo create study_2021-2022_os-intro --template=yamadharma/course-directory-student-template --public
GraphQL: Could not clone: Name already exists on this account (cloneTemplateRepository)
[edsmirnovmaljce@edsmirnovmaljce Операционные системы]$ git clone --recursive git@github.com:edsmirnovmaljce/study_2021-2022_os-intro.git
```

Копирование шаблона репозитория

11. Настроил репозиторий.

```
[edsmirnovmaljce@edsmirnovmaljce Операционные системы]$ cd os-intro
[edsmirnovmaljce@edsmirnovmaljce os-intro]$ rm package.json
rm: невозможно удалить 'package.json': Нет такого файла или каталога
```

Удаление лишнего файла

```
[edsmirnovmaljce@edsmirnovmaljce os-intro]$ git add .  
[edsmirnovmaljce@edsmirnovmaljce os-intro]$ git commit -am 'feat(main): make course structure'  
На ветке master
```

Создание каталогов

```
не есть коммитов, нет изменений в рабочем каталоге  
[edsmirnovmaljce@edsmirnovmaljce os-intro]$ git push  
Everything up-to-date
```

Отправление файла на сервер

## Выводы

- Я научился создавать репозиторий на Github с помощью командной строки.
- Я ознакомился с Git структурой и понял ее преимущества при работе в команде, а именно удобство отслеживания изменений.

# Ответы на контрольные вопросы

1. VCS – это системы контроля версий. Они используются при работе в команде. 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.
2. Хранилище – место, в котором хранятся все версии проекта. Для уменьшения объема используемой памяти можно хранить только изменения проекта. Commit – добавленные и измененные файлы по сравнению с предшествующей версией проекта. История – последовательность изменений проекта. Рабочая копия – копия над которой сейчас идет работа.
3. Централизованная система – система, в которой существует центральное хранилище, которое доступно всем участникам проекта. В децентрализованной системе у каждого участника есть свой репозиторий, что позволяет работать, не подключаясь к сети. Пример централизованной системы: CVS. Децентрализованной: Git.
4. При единоличной работе с VCS берешь нужную версию, вносишь правки и добавляешь коммит.
5. При совместной работе в централизованном хранилище, добавляется работа по устранению конфликтов. Также необходимо сливать версии, и вообще следить за структурированностью истории. Нельзя забывать проверять последнюю

- версию программы, потому что напарник мог ее изменить.
6. Git помогает работать одновременно над одним и тем же проектом независимо, а потом совмещать достижения разработчиков. Также он помогает хранить историю версий, что позволяет быстро откатиться в случае необходимости.
  7. Основные 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 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: `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. Пример использования локального репозитория: В Git создать репозиторий привязанный к имени пользователя и email и работать в нем не подключаясь к сети. Пример использования удаленного репозитория: создать репозиторий на Github, настроить авторизацию через ssh-ключ и ргр-ключ, авторизоваться и работать с ним через командную строку.

9. Ветви необходимы в случае, если есть несколько путей развития программы, поэтому из одной версии получаются сразу несколько.
10. Игнорировать файлы можно с помощью команды `gitignore` для того, чтобы не забивать хранилище мусором.