

работа № 2

Лабораторная работа: Базовая настройка Git

ФИО студента:

Рафи Кази ар

НКАбд-03-24

1032238132@pfur.ru

Кулябов Дмитрий Сергеевич

Moscow

1 — "Базовая настройка Git"

1. Установка программного обеспечения

```
sudo apt install git gh -y
```

```
[& krafia@pop-os ] → [🕒 04:50 PM] → www.krafi.info
└─ > ~/Downloads/SUPERHOT - MIND CONTROL DELETE
> sudo apt install git gh -y
[sudo] password for krafia:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.34.1-1ubuntu1.15).
git set to manually installed.
The following package was automatically installed and is no longer required:
  nvidia-firmware-570-570.153.02
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  gh
0 upgraded, 1 newly installed, 0 to remove and 76 not upgraded.
Need to get 6,242 kB of archives.
After this operation, 33.7 MB of additional disk space will be used.
Get:1 http://apt.pop-os.org/ubuntu jammy/universe amd64 gh amd64 2.4.0+dfsg1-2 [6,242 kB]
Fetched 6,242 kB in 1s (8,709 kB/s)
Selecting previously unselected package gh.
(Reading database ... 250543 files and directories currently installed.)
Preparing to unpack .../gh_2.4.0+dfsg1-2_amd64.deb ...
Unpacking gh (2.4.0+dfsg1-2) ...
Setting up gh (2.4.0+dfsg1-2) ...
Processing triggers for man-db (2.10.2-1) ...

[& krafia@pop-os ] → [🕒 04:50 PM] → www.krafi.info
└─ > ~/Downloads/SUPERHOT - MIND CONTROL DELETE      took 6s
└─ █
```

2. Базовая настройка Git

git config --global user.name "Kazi Ar Rafi"

git config --global user.email "krafi.info@gmail.com"

git config --global core.quotepath false

git config --global init.defaultBranch master

git config --global core.autocrlf input

```

[& krafi@pop-os ] [🕒 04:51 PM]
└─ > ~/Downloads/SUPERHOT - MIND CO
└─ > git config --global user.name "Kazi Ar
[& krafi@pop-os ] [🕒 04:51 PM]
└─ > ~/Downloads/SUPERHOT - MIND CON
└─ > git config --global user.email "krafi.
[& krafi@pop-os ] [🕒 04:51 PM]
└─ > ~/Downloads/SUPERHOT - MIND CON
└─ > git config --global core.quotepath fal
git config --global init.defaultBranch mas
git config --global core.autocrlf input
git config --global core.safecrlf warn
[& krafi@pop-os ] [🕒 04:52 PM]
└─ > ~/Downloads/SUPERHOT - MIND CON
└─ >

```

git config --global core.safecrlf warn

3. Создание SSH-ключей

RSA (4096 бит): `ssh-keygen -t rsa -b 4096 -C "ваш_email@example.com"`

```

➤ ssh-keygen -t rsa -b 4096 -C "krafi.info@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/krafi/.ssh/id_rsa):
Created directory '/home/krafi/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/krafi/.ssh/id_rsa
Your public key has been saved in /home/krafi/.ssh/id_rsa.pub
The key fingerprint is:

```

Ed25519 (современный \ modern): `ssh-keygen -t ed25519 -`

```
[krafi@pop-os] [04:52 PM]
└─ ~/Downloads/SUPERHOT - MIND CONTROL DELETE
-> ssh-keygen -t ed25519 -C "krafi.info@gmail.co
generating public/private ed25519 key pair.
enter file in which to save the key (/home/krafi/
enter passphrase (empty for no passphrase):
enter same passphrase again:
your identification has been saved in /home/krafi/
your public key has been saved in /home/krafi/.ss
the key fingerprint is:
```

С "ваш_email@example.com"

Копирование публичного ключа: `cat ~/.ssh/id_ed25519.pub`
`| xclip -sel clip`

1. Settings → SSH and GPG keys .
2. Нажмите New SSH key , вставьте скопированный

```
[krafi@pop-os] [04:53 PM] www.krafi.in
└─ ~/Downloads/SUPERHOT - MIND CONTROL DELETE too
-> cat /home/krafi/.ssh/id_ed25519.pub | xclip -sel clip

[krafi@pop-os] [04:55 PM] www.krafi.in
└─ ~/Downloads/SUPERHOT - MIND CONTROL DELETE
-> cat /home/krafi/.ssh/id_ed25519.pub
```

Ключ.

4. Создание PGP-ключа (create):

```
[A krafi@pop-os ] [04:55 PM] www.krafi
~/Downloads/SUPERHOT - MIND CONTROL DELETE
gpg --full-generate-key
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: directory '/home/krafi/.gnupg' created
gpg: keybox '/home/krafi/.gnupg/pubring.kbx' created
Please select what kind of key you want:
  (1) RSA and RSA (default)
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (14) Existing key from card
Your selection?
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072)
Requested keysize is 3072 bits
Please specify how long the key should be valid.
    0 = key does not expire
    <n> = key expires in n days
    <n>w = key expires in n weeks
    <n>m = key expires in n months
    <n>y = key expires in n years
Key is valid for? (0)
Key does not expire at all
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: rafi
Name must be at least 5 characters long
Real name: krafi
Email address: krafi.info@gmail.com
Comment: hello
You selected this USER-ID:
    "krafi (hello) <krafi.info@gmail.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, use the mouse,
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, use the mouse,
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: /home/krafi/.gnupg/trustdb.gpg: trustdb created
gpg: key 6E2FDB98EEA550AD marked as ultimately trusted
gpg: directory '/home/krafi/.gnupg/openpgp-revocs.d' created
gpg --full-generate-key
```

- Тип: RSA and RSA
- Размер: 4096
- Срок действия: 0 (без ограничений)
- Укажите имя, email (должен совпадать с тем, что указан в `git config --global user.email`), комментарий.

Получение отпечатка ключа:

Get key fingerprint: `gpg --list-secret-keys --keyid-format LONG`

Экспорт публичного ключа: Export public key: `gpg --armor --export 1234ABCD5678EFGH90IJKLMNOPQ2RSTUVWXYZ123 | xclip -sel clip`

Добавление ключа в GitHub: Settings → SSH and GPG keys

.

New GPG key , вставьте содержимое из буфера обмена.

5. Настройка автоматических подписей коммитов

Configuring Automatic Commit Signing

`git config --global user.signingkey 1234ABCD5678EFGH90IJKLMNOPQ2RSTUVW`

`git config --global commit.gpgsign true`

`git config --global gpg.program $(which gpg)`

6. Авторизация через GitHub CLI

`gh auth login`

7. Создание рабочего пространства

Создание каталога проекта: `mkdir -p ~/work/study/2022-2023/"Операционные системы"`

```
cd ~/work/study/2022-2023/"Операционные системы"
```

Создание репозитория на основе шаблона:

Create repo based on template: `gh repo create study_2022-2023_os-intro --template=yamadharm/course-directory-student-template --public`

```
git clone --recursive git@github.com:<your_username>/study_2022-2023_os-intro.git os-intro
```

Настройка структуры курса:

Configure course structure: `cd ~/work/study/2022-2023/"Operating Systems"/os-intro`

```
rm package.json
```

```
echo os-intro > COURSE
```

```
make
```

Отправка изменений:

Push changes: `git add .`

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

```
git push
```

□ Контрольные вопросы

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

Ответ:

Системы контроля версий (VCS) позволяют отслеживать изменения файлов проекта, сохранять историю изменений, работать над проектом нескольким разработчикам одновременно, а также восстанавливать предыдущие версии.

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

Ответ:

- Хранилище (репозиторий) — это место, где хранится вся история изменений проекта.
- Commit — это зафиксированное состояние изменений.
- История — это последовательность коммитов, показывающая развитие проекта.
- Рабочая копия — это локальная копия проекта, с которой работает пользователь.

3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

Ответ:

- Централизованные VCS : есть один главный сервер, все работают с ним (например, SVN).
- Децентрализованные VCS : каждый участник имеет полную копию репозитория (например, Git, Mercurial).

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

Ответ:

При единоличной работе вы:

- Создаете локальный репозиторий (git init).
- Добавляете файлы (git add).
- Сохраняете изменения (git commit).
- Может быть, отправляете их на удалённый сервер (git push).

5. Опишите порядок работы с общим хранилищем VCS.

Ответ:

- Клонирование репозитория (`git clone`).
- Переключаетесь на нужную ветку (`git checkout`).
- Вносите изменения, добавляете их (`git add`) и делаете коммит (`git commit`).
- Отправляете изменения (`git push`).
- Если есть изменения у других участников — получаете их (`git pull`).
-

6. Каковы основные задачи, решаемые инструментальным средством `git`?

Ответ:

- Управление версиями кода.
- Совместная разработка.
- Откат изменений.
- Работа с ветками.
- Поддержка истории изменений.

7. Назовите и дайте краткую характеристику командам `git`.

Ответ:

- `git init` — Инициализация нового репозитория
- `git clone` — Клонирование существующего репозитория
- `git add` — Добавление файлов в индекс (подготовка к коммиту)
- `git commit` — Сохранение изменений с комментарием (фиксация изменений в истории)

- `git status` — Показывает текущее состояние репозитория (какие файлы изменены, добавлены и т.д.)
- `git push` — Отправка локальных коммитов на удалённый репозиторий
- `git pull` — Получение и слияние изменений из удалённого репозитория с локальной веткой
- `git branch` — Управление ветками (создание, удаление, просмотр списка)
- `git checkout` — Переключение между ветками или восстановление файлов
- `git merge` — Слияние одной ветки с текущей

8. Приведите примеры использования при работе с локальным и удалённым репозиториями.

Ответ:

- Локальный репозиторий :

`git init`

`git add .`

`git commit -m "Initial commit"`

Удалённый репозиторий :

`git remote add origin git@github.com:user/repo.git`

`git push -u origin master`

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

Ответ:

Ветви позволяют работать над разными частями проекта независимо друг от друга. Например, одна ветка может использоваться для разработки, другая — для тестирования, третья — для исправления ошибок.

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

Ответ:

Файлы, не относящиеся к исходному коду (временные, объектные, системные), можно игнорировать, создав файл .gitignore. Это позволяет избежать загрузки лишних данных в репозиторий.

Пример содержания .gitignore:

```
*.log  
__pycache__  
env/  
.DS_Store
```

Выводы по работе:

Выполнив данную лабораторную работу, я:

- Настроил Git под свои нужды.
- Создал и зарегистрировал SSH- и PGP-ключи.
- Настроил автоматическую подпись коммитов.
- Зарегистрировался на GitHub и создал репозиторий курса.
- Изучил и применил базовые команды Git для работы с локальным и удалённым репозиторием.
- Разобрался с принципами управления версиями и безопасностью.