

Шаблон отчёта по лабораторной работе

Простейший вариант

Дзаки Рафли Зайдан

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Контрольные вопросы	13
5	Выводы	16

Список иллюстраций

3.1	Базовая настройка git	8
3.2	Создание ключа SSH	8
3.3	Создание ключа GPG	9
3.4	Ключ SSH создан	9
3.5	Ключ GPG создан	9
3.6	Отпечаток приватного ключа	10
3.7	Настройка подписей	10
3.8	Настро	10
3.9	Создание репозитория	11
3.10	Настраиваем каталог курса	11
3.11	Отправляем наши файлы на сервер	12

Список таблиц

1 Цель работы

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

2 Задание

1. Создать базовую конфигурацию для работы с git.
2. Создать ключ SSH.
3. Создать ключ PGP.
4. Настроить подписи git.
5. Зарегистрироваться на Github.
6. Создать локальный каталог для выполнения заданий по предмету.

3 Теоретическое введение

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. 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.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Базовая настройка git:

1. Задаём имя и email владельца репозитория (1 и 2 строка на рисунке)
2. Настраиваем utf-8 в выводе сообщений git (3 строка на рисунке)

3. Настраиваем верификацию и подписание коммитов git. Зададим имя начальной ветки (будем называть её master) (4 строка на рисунке)
4. Параметр autocrlf (5 строка на рисунке)
5. Параметр safecrlf (6 строка на рисунке)

```
raflzaa@raflzaa:~$ git config --global user.name "rafzai"
raflzaa@raflzaa:~$ git config --global user.email "1032235550@pfur.ru"
raflzaa@raflzaa:~$ git config --global core.quotepath false
raflzaa@raflzaa:~$ git config --global init.defaultBranch master
raflzaa@raflzaa:~$ git config --global core.autocrlf input
raflzaa@raflzaa:~$ git config --global core.safecrlf warn
```

Рис. 3.1: Базовая настройка git

Создаём ключ SSH. В терминале вводим данную команду: `ssh-keygen -t rsa -b 4096`

Далее во всех пунктах пользуемся клавишей Enter и получаем наш ключ.

```
raflzaa@raflzaa:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/raflzaa/.ssh/id_rsa):
/home/raflzaa/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/raflzaa/.ssh/id_rsa
Your public key has been saved in /home/raflzaa/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:XenJ5RMxaCgBartP2Ug7VEJPiKQByKMzAv5DIdoe27Y raflzaa@raflzaa
The key's randomart image is:
+----[RSA 4096]-----+
|+.....ooo. . .o |
|oo o.o.o. . o. o |
|+ + + . o. .o o |
|=o + . o . + + . |
|o+o . o S . + o |
|. oo + = . |
|. +o * . |
| o oo . |
|.E.. |
+-----[SHA256]-----+
```

Рис. 3.2: Создание ключа SSH


```

raflzaa@raflzaa:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/raflzaa/.ssh/id_ed25519):
/home/raflzaa/.ssh/id_ed25519 already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/raflzaa/.ssh/id_ed25519
Your public key has been saved in /home/raflzaa/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:J/umcbuLphj5YpnD9gBE/wu0b3zqp501W0dwNcqME7E raflzaa@raflzaa
The key's randomart image is:
+--[ED25519 256]--+
| .      o. o      |
| .      * o .      |
| .      E =        |
| .      +          |
| . . . S . .       |
| + o . + .         |
| .o=o. = o .        |
| .@*oo.o.o         |
| =*o*o+o=o         |
+-----[SHA256]-----+

```

Рис. 3.3: Создание ключа GPG

Ключ нужно добавить на github. Для этого переходим на сайте в раздел “Settings” и выбираем “SSH and GPG keys”.

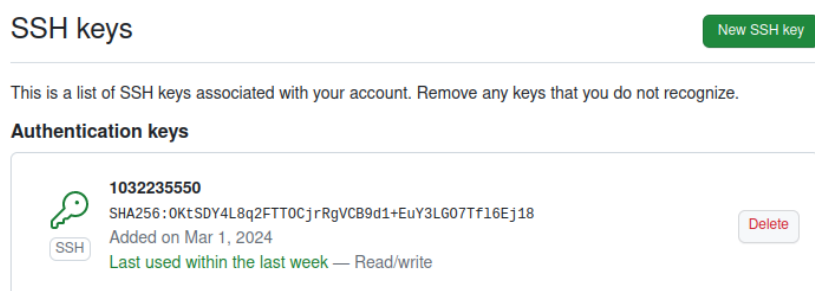


Рис. 3.4: Ключ SSH создан

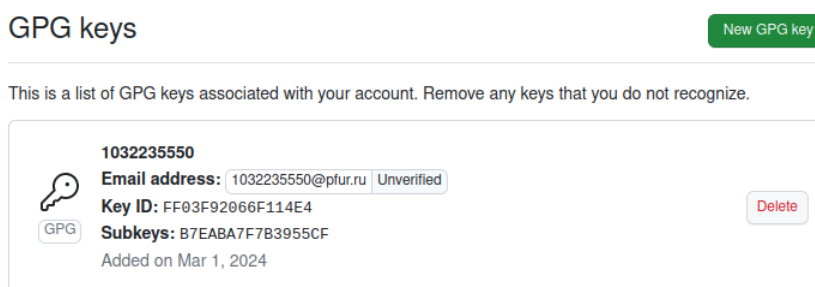


Рис. 3.5: Ключ GPG создан

Выводим список ключей и копируем отпечаток приватного ключа

```
raflzaa@raflzaa:~$ gpg --list-secret-keys --keyid-format LONG
/home/raflzaa/.gnupg/pubring.kbx
-----
sec   rsa4096/FF03F92066F114E4 2024-02-29 [SC]
      B7ADFEA636B19E04300E9D9BFF03F92066F114E4
uid    [ultimate] rafzai <1032235550@pfur.ru>
ssb    rsa4096/B7EABA7F7B3955CF 2024-02-29 [E]
```

Рис. 3.6: Отпечаток приватного ключа

Настройка автоматических подписей коммитов git

```
raflzaa@raflzaa:~$ git config --global user.signingkey FF03F92066F114E4
raflzaa@raflzaa:~$ git config --global commit.gpgsign true
raflzaa@raflzaa:~$ git config --global gpg.program $(which gpg2)
```

Рис. 3.7: Настройка подписей

Возвращаемся в наш терминал и настраиваем gh командой: gh auth login.

Во всех пунктах выбираем у(yes). По полученной ссылке переходим в браузер на виртуальной машине и вводим код из терминала (находится перед ссылкой).

```
raflzaa@raflzaa:~$ 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: 641D-166E
- Press Enter to open github.com in your browser...
Gtk-Message: 08:42:52.492: Not loading module "atk-bridge": The functionality is provided by GTK natively. Please try to not load it.
✓ Authentication complete. Press Enter to continue...

- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as rafzai
```

Рис. 3.8: Настро

Создаём репозиторий курса на основе шаблона. Все нужные команды для создания были в указаниях к лабораторной работе. В 4 команде, вместо , указываем своё имя профиля на github.

1. `mkdir -p ~/work/study/2021-2022/“Операционные системы”`
2. `cd ~/work/study/2021-2022/“Операционные системы”`

3. `gh repo create study_2021-2022_os-intro --template=yamadharma/course-directory-student-template --public`
4. `git clone --recursive git@github.com:/study_2021-2022_os-intro.git os-intro`

```

raflzaa@raflzaa:~$ mkdir -p ~/work/study/2023-2024/"Операционные системы"
raflzaa@raflzaa:~$ cd ~/work/study/2023-2024/"Операционные системы"
raflzaa@raflzaa:~/work/study/2023-2024/Операционные системы$ gh repo create study_20
23-2024_os-intro --template=yamadharma/course-directory-student-template --public
GraphQL: Could not clone: Name already exists on this account (cloneTemplateRepository)
raflzaa@raflzaa:~/work/study/2023-2024/Операционные системы$ git clone --recursive g
it@github.com:raflzaa/study_2023-2024_os-intro.git os-intro
fatal: destination path 'os-intro' already exists and is not an empty directory.

```

Рис. 3.9: Создание репозитория

Настраиваем каталог курса. Для этого переходим в него командой: `cd ~/work/study/2021-2022/"Операционные системы"/os-intro`

Далее командой `ls` проверяем, что мы в него перешли. В каталоге “os-intro” нам потребуется удалить файл “package.json”. Выполняем данную задачу командой: `rm package.json`

Снова командой `ls` проверяем успешное выполнение удаления файла.

```

raflzaa@raflzaa:~/work/study/2023-2024/Операционные системы$ cd ~/work/study/2023-20
24/"Операционные системы"/os-intro
raflzaa@raflzaa:~/work/study/2023-2024/Операционные системы/os-intro$ rm package.js
on
rm: cannot remove 'package.json': No such file or directory
raflzaa@raflzaa:~/work/study/2023-2024/Операционные системы/os-intro$ echo os-intro
> COURSE
raflzaa@raflzaa:~/work/study/2023-2024/Операционные системы/os-intro$ make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare        Generate directories structure
  submodule      Update submodules

```

Рис. 3.10: Настраиваем каталог курса

Создаём необходимые каталоги и отправляем наши файлы на сервер `make COURSE=os-intro`

1. `git add .`
2. `git commit -am 'feat(main): make course structure'`
3. `git push`

```
raflzaa@raflzaa:~/work/study/2023-2024/Операционные системы/os-intro$ git add .
raflzaa@raflzaa:~/work/study/2023-2024/Операционные системы/os-intro$ git commit -am
'feat(main): make course structure'
[master 80ea56a] feat(main): make course structure
9 files changed, 4 insertions(+), 45 deletions(-)
create mode 100644 labs/lab02/report/r.docx
create mode 100644 labs/lab02/report/r.pdf
create mode 100644 labs/lab02/report/report.docx
create mode 100644 labs/lab02/report/report.pdf
create mode 100644 labs/lab03/report/report.docx
create mode 100644 labs/lab03/report/report.pdf
raflzaa@raflzaa:~/work/study/2023-2024/Операционные системы/os-intro$ git push
```

Рис. 3.11: Отправляем наши файлы на сервер

4 Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Это программное обеспечение для облегчения работ с изменяющейся информацией. VCS позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Хранилище (repository), или репозиторий, — место хранения всех версий и служебной информации. Commit («[трудовой] вклад», не переводится) — синоним версии; процесс создания новой версии. История — место, где сохраняются все коммиты, по которым можно посмотреть данные о коммитах. Рабочая копия — текущее состояние файлов проекта, основанное на версии, загруженной из хранилища.
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные VCS: одно основное хранилище всего проекта и каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно. Децентрализованные VCS: у каждого пользователя свой вариант (возможно не один) репозитория.

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? Git — это система управления версиями. У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.
7. Назовите и дайте краткую характеристику командам git. `git -version` (Проверка версии Git) `git init` (Инициализировать ваш текущий рабочий каталог как Git-репозиторий) `git clone https://www.github.com/username/reponame` (Скопировать существующий удаленный Git-репозиторий) `git remote` (Просмотреть список текущих удалённых репозиториях Git) `git remote -v` (Для более подробного вывода) `git add my_script.py` (Можете указать в команде конкретный файл). `git add .` (Позволяет охватить все файлы в текущем каталоге, включая файлы, чье имя начинается с точки) `git commit "Commit message"` (Вы можете сжать все индексированные файлы и отправить коммит). `git branch` (Просмотреть список текущих веток можно с помощью команды `branch`) `git -help` (Чтобы узнать больше обо всех доступных параметрах и командах) `git push origin master` (Передать локальные коммиты в ветку удаленного репозитория).
8. Приведите примеры использования при работе с локальным и удалённым репозиториями.
9. Что такое и зачем могут быть нужны ветки (branches)? Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций: чтобы не повредить основному проекту, создается новая ветка

специально для экспериментов.

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

Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашей репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты.

5 Выводы

В ходе выполнения лабораторной работы изучили идеологию и применение средств контроля версий, а также освоили умения по работе с git