

Отчёт по лабораторной работе №2

Управление версиями

Фатима Халилова

Содержание

1 Цель работы	5
2 Выполнение лабораторной работы	6
3 Вывод	17
4 Контрольные вопросы	18

Список иллюстраций

2.1	Загрузка пакетов	7
2.2	Параметры репозитория	8
2.3	rsa-4096	9
2.4	ed25519	10
2.5	GPG ключ	11
2.6	GPG ключ	12
2.7	Параметры репозитория	13
2.8	Связь репозитория с аккаунтом	14
2.9	Загрузка шаблона	15
2.10	Первый коммит	16

Список таблиц

1 Цель работы

Целью данной работы является изучение идеологии и применения средств контроля версий и освоение умений работать с git.

2 Выполнение лабораторной работы

Устанавливаем git, git-flow и gh.

```
frhalolova@frhalilova:~$ git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--no-lazy-fetch]
           [--no-optional-locks] [--no-advice] [--bare] [--git-dir=<path>]
           [--work-tree=<path>] [--namespace=<name>] [--config-env=<name>=<envvar>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
clone      Clone a repository into a new directory
init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
add        Add file contents to the index
mv         Move or rename a file, a directory, or a symlink
restore    Restore working tree files
rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
bisect    Use binary search to find the commit that introduced a bug
diff      Show changes between commits, commit and working tree, etc
grep      Print lines matching a pattern
log       Show commit logs
show      Show various types of objects
status    Show the working tree status
```

Рисунок 2.1: Загрузка пакетов

Зададим имя и email владельца репозитория, кодировку и прочие параметры.

```
frhalolova@frhalilova:~$  
frhalolova@frhalilova:~$ git config --global user.name "fr-halolova"  
>  
frhalolova@frhalilova:~$  
frhalolova@frhalilova:~$ git config --global user.name "fr-halolova"  
frhalolova@frhalilova:~$ git config --global user.email "1132243815@pfur.ru"  
frhalolova@frhalilova:~$ git config --global core.quotepath false  
frhalolova@frhalilova:~$ git config --global init.defaultBranch master  
frhalolova@frhalilova:~$ git config --global core.autocrlf input  
frhalolova@frhalilova:~$ git config --global core.safecrlf warn  
frhalolova@frhalilova:~$
```

Рисунок 2.2: Параметры репозитория

Создаем SSH ключи

```
frhalolova@frhalilova:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/frhalolova/.ssh/id_rsa):
Created directory '/home/frhalolova/.ssh'.
Enter passphrase for "/home/frhalolova/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/frhalolova/.ssh/id_rsa
Your public key has been saved in /home/frhalolova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:44MZU31/pBcuE+mBUh07Is+B5RmIi3Tb2oGvsZn0LQI frhalolova@frhalilova
The key's randomart image is:
+---[RSA 4096]----+
|      .+... |
|     . o o= +.o |
|    . o *+.*.* ..|
|   . = o=.+=.o. |
|    o S .o +o.o |
|   E X +     oo |
|     = X .       |
|     * + .       |
|     . .       |
+---[SHA256]----+
frhalolova@frhalilova:~$
```

Рисунок 2.3: rsa-4096

```
frhalolova@frhalilova:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/frhalolova/.ssh/id_ed25519):
Enter passphrase for "/home/frhalolova/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/frhalolova/.ssh/id_ed25519
Your public key has been saved in /home/frhalolova/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:oPOTgEsIMxLjaDYLQ5DrEw8SBM7PHexHUX+mbTXyrYg frhalolova@frhalilova
The key's randomart image is:
+--[ED25519 256]--+
|0+   ... |
|0o . . . |
|OX 0.. . + o |
|OB=.o.o. = + o|
|+.*o+o .S . o ..|
| + o +.. . . . |
|  o + E . . |
|   . |
+---[SHA256]---+
frhalolova@frhalilova:~$
```

Рисунок 2.4: ed25519

Создаем GPG ключ

```
frhalolova@frhalilova:~$ gh auth login
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /home/frhalolova/.ssh/id_rsa.pub
? Title for your SSH key: GitHub CLI
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: B302-A18C
Press Enter to open https://github.com/login/device in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
✓ Uploaded the SSH key to your GitHub account: /home/frhalolova/.ssh/id_rsa.pub
✓ Logged in as fr=halolova
frhalolova@frhalilova:~$
```

Рисунок 2.5: GPG ключ

Добавляем GPG ключ в аккаунт

```
pub rsa4096 2025-11-12 [SC]
  0F0776EA9D3A53776844FD9FCC6FB4EA807D0786
uid          Fatima Halilova <1132243815@pfur.ru>
sub rsa4096 2025-11-12 [E]

frhalilova@frhalilova:~$ gpg --list-secret-keys --keyid-format LONG
gpg: checking the trustdb
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
[keyboxd]
-----
sec  rsa4096/CC6FB4EA807D0786 2025-11-12 [SC]
  0F0776EA9D3A53776844FD9FCC6FB4EA807D0786
uid          [ultimate] Fatima Halilova <1132243815@pfur.ru>
ssb  rsa4096/E491C196E7FCECBF 2025-11-12 [E]

frhalilova@frhalilova:~$ gpg --armor --export CC6FB4EA807D0786
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGkUjrUBEAC9C5reJ4UCvBR909Ay+3vNzmIdsXuWDUeXC1bFHLKkbL5UEQji
dqbHQbvNpJgnv4E/dwAffDKgfgCd016cmzyM8lYEB8sLiyganZeUu0gPwq+W0B6r
WYpl/gle3CzmAeqPD+OMoeFEomjAc0KWBrBXpnwH5V/myjJdAKVlv4eaBdBRCit
RzPVERZnSnyWDZqoJReMcZ9mKvKi95ZJuCKdZy4sVVjqlttY18MuEETD/yo/vUu
tOZminwCLJH2yvg5PsreHbq0IvR4Mwr67XIR55x9H4NIooiw3hwmM5hFsHacJFwP
1ZVLTVEQWhYVzKJ5yFM/QhLXn6CZePjuGuHpGRr4Hzb+K8IOZVsYs4eKeupeFRa+
mzIsQQOy47mNbMF+BiFOBsFmpFP07JGv+Mt8NE35xqmsnHceVIhkzOEWOh5umyt
K8y1UPWg1YXFRIwko/TWuMtLbH4YQvt967yzQ0i1GorxkKF2BsxzeydyLdz1d6xJ
```

Рисунок 2.6: GPG ключ

Настройка автоматических подписей коммитов git

```
-----BEGIN PGP PUBLIC KEY BLOCK-----  
3hBz  
=xU3v  
-----END PGP PUBLIC KEY BLOCK-----  
frhalolova@frhalilova:$  
frhalolova@frhalilova:$  
frhalolova@frhalilova:$  
frhalolova@frhalilova:$ git config --global user.signkey C256F576006A4852  
frhalolova@frhalilova:$ git config --global commit.gpgsign true  
frhalolova@frhalilova:$ git config --global gpg.program $(which gpg2)  
frhalolova@frhalilova:$
```

Рисунок 2.7: Параметры репозитория

Настройка gh

```
frhalolova@frhalilova:~$ gh auth login
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /home/frhalolova/.ssh/id_rsa.pub
? Title for your SSH key: GitHub CLI
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: B392-A18C
Press Enter to open https://github.com/login/device in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
✓ Uploaded the SSH key to your GitHub account: /home/frhalolova/.ssh/id_rsa.pub
✓ Logged in as fr-halolova
frhalolova@frhalilova:~$
```

Рисунок 2.8: Связь репозитория с аккаунтом

Загрузка шаблона репозитория и синхронизация

```
frhalolova@frhalilova:~$  
frhalolova@frhalilova:~$ mkdir -p ~/work/study/2025-2026/"Операционные системы"  
frhalolova@frhalilova:~$ cd ~/work/study/2025-2026/"Операционные системы"  
frhalolova@frhalilova:~/work/study/2025-2026/Операционные системы$ gh repo create os-intro -  
-template=yamadharma/course-directory-student-template --public  
✓ Created repository fr-halolova/os-intro on github.com  
https://github.com/fr-halolova/os-intro  
frhalolova@frhalilova:~/work/study/2025-2026/Операционные системы$ git clone --recursive git  
@github.com:fr-halolova/os-intro.git os-intro  
Cloning into 'os-intro'...  
The authenticity of host 'github.com (140.82.121.3)' can't be established.  
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhpZisF/zLDA0zPMsvHdkr4UvC0qU.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

Рисунок 2.9: Загрузка шаблона

Подготовка репозитория и коммит изменений

```
create mode 100644 project-personal/stage6/report/.gitkeep
create mode 100644 project-personal/stage6/report/.projectile
create mode 100644 project-personal/stage6/report/Makefile
create mode 100644 project-personal/stage6/report/_assets/preamble.tex
create mode 100644 project-personal/stage6/report/_quarto.yml
create mode 100644 project-personal/stage6/report/_resources/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 project-personal/stage6/report/bib/cite.bib
create mode 100644 project-personal/stage6/report/image/solvay.jpg
create mode 100644 project-personal/stage6/report/os-intro--project-personal--stage6--report.qmd
frhalolova@frhalilova:~/work/study/2025-2026/Операционные системы/os-intro$ git push
Enumerating objects: 106, done.
Counting objects: 100% (106/106), done.
Delta compression using up to 4 threads
Compressing objects: 100% (87/87), done.
Writing objects: 100% (103/103), 703.81 KiB | 4.96 MiB/s, done.
Total 103 (delta 42), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (42/42), completed with 1 local object.
To github.com:fr-halolova/os-intro.git
 ff2414c..05b4bca master -> master
frhalolova@frhalilova:~/work/study/2025-2026/Операционные системы/os-intro$
```

Рисунок 2.10: Первый коммит

3 Вывод

Мы приобрели практические навыки работы с сервисом github.

4 Контрольные вопросы

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

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

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

- хранилище - пространство на накопителе где расположен репозиторий
- commit - сохранение состояния хранилища
- история - список изменений хранилища (коммитов)
- рабочая копия - локальная копия сетевого репозитория, в которой работает программист. Текущее состояние файлов проекта, основанное на версии, загруженной из хранилища (обычно на последней)

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

Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере. Доступ к нему осуществлялся через специальное клиентское приложение. В качестве примеров таких программных продуктов можно привести CVS, Subversion.

Распределенные системы контроля версий (Distributed Version Control System, DVCS) позволяют хранить репозиторий (его копию) у каждого разработчика, работающего с данной системой. При этом можно выделить центральный репозиторий (условно), в который будут отправляться изменения из локальных и, с ним же эти локальные репозитории будут синхронизироваться. При работе с такой системой, пользователи периодически синхронизируют свои локальные репозитории с центральным и работают непосредственно со своей локальной копией. После внесения достаточного количества изменений в локальную копию они (изменения) отправляются на сервер. При этом сервер, чаще всего, выбирается условно, т.к. в большинстве DVCS нет такого понятия как «выделенный сервер с центральным репозиторием».

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

Один пользователь работает над проектом и по мере необходимости делает коммиты, сохраняя определенные этапы.

5. Опишите порядок работы с общим хранилищем VCS.

Несколько пользователей работают каждый над своей частью проекта. При этом каждый должен работать в своей ветки. При завершении работы ветка пользователя слиивается с основной веткой проекта.

6. Каковы основные задачи, решаемые инструментальным средством git?

- Ведение истории версий проекта: журнал (log), метки (tags), ветвления (branches).
- Работа с изменениями: выявление (diff), слияние (patch, merge).
- Обеспечение совместной работы: получение версии с сервера, загрузка обновлений на сервер.

7. Назовите и дайте краткую характеристику командам git.

- `git config` - установка параметров
- `git status` - полный список изменений файлов, ожидающих коммита
- `git add .` - сделать все измененные файлы готовыми для коммита.
- `git commit -m «[descriptive message]»` - записать изменения с заданным сообщением.
- `git branch` - список всех локальных веток в текущей директории.
- `git checkout [branch-name]` - переключиться на указанную ветку и обновить рабочую директорию.
- `git merge [branch]` – соединить изменения в текущей ветке с изменениями из заданной.
- `git push` - запушить текущую ветку в удаленную ветку.
- `git pull` - загрузить историю и изменения удаленной ветки и произвести слияние с текущей веткой.

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

- `git remote add [имя] [url]` – добавляет удалённый репозиторий с заданным именем;
- `git remote remove [имя]` – удаляет удалённый репозиторий с заданным именем;
- `git remote rename [старое имя] [новое имя]` – переименовывает удалённый репозиторий;

- `git remote set-url [имя] [url]` – присваивает репозиторию с именем новый адрес;
- `git remote show [имя]` – показывает информацию о репозитории.

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

Ветвление — это возможность работать над разными версиями проекта: вместо одного списка с упорядоченными коммитами история будет расходиться в определённых точках. Каждая ветвь содержит легковесный указатель HEAD на последний коммит, что позволяет без лишних затрат создать много веток. Ветка по умолчанию называется `master`, но лучше назвать её в соответствии с разрабатываемой в ней функциональностью.

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

Зачастую нам не нужно, чтобы Git отслеживал все файлы в репозитории, потому что в их число могут входить: