

Отчёт по лабораторной работе №2

Управление версиями

Павел Фудоткин

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.

```

pavelfedotkin@pavelfedotkin:~$ git
использование: 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>]

Стандартные команды Git используемые в различных ситуациях:

создание рабочей области (смотрите также: git help tutorial)
clone    Клонирование репозитория в новый каталог
init     Создание пустого репозитория Git или переинициализация существующего

работа с текущими изменениями (смотрите также: git help everyday)
add      Добавление содержимого файла в индекс
mv       Перемещение или переименование файла, каталога или символической ссылки
restore  Восстановление файлов в рабочем каталоге
rm       Удаление файлов из рабочего каталога и индекса

просмотр истории и текущего состояния (смотрите также: git help revisions)
bisect   Выполнение двоичного поиска коммита, который вносит ошибку
diff     Вывод разницы между коммитами, коммитом и рабочим каталогом и т.д.
grep     Вывод строк, соответствующих шаблону
log      Вывод истории коммитов
show     Вывод различных типов объектов
status   Вывод состояния рабочего каталога

выращивание, маркировка и правка вашей общей истории
branch   Вывод списка, создание или удаление веток
commit   Запись изменений в репозиторий
merge    Объединение одной или нескольких историй разработки вместе
rebase   Повторное применение коммитов над верхушкой другой ветки
reset    Сброс текущего состояния HEAD на указанное состояние
switch   Переключение веток
tag      Создание, вывод списка, удаление или проверка метки, подписанной с помощью GPG

```

Рис. 2.1: Загрузка пакетов

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

```
pavel-fedotkin@pavel-fedotkin:~$  
pavel-fedotkin@pavel-fedotkin:~$  
pavel-fedotkin@pavel-fedotkin:~$ git config --global user.name "pavel-fedotkin"  
pavel-fedotkin@pavel-fedotkin:~$ git config --global user.email "1032244328@pfur.ru"  
pavel-fedotkin@pavel-fedotkin:~$ git config --global core.quotePath false  
pavel-fedotkin@pavel-fedotkin:~$  
pavel-fedotkin@pavel-fedotkin:~$ git config --global init.defaultBranch master  
pavel-fedotkin@pavel-fedotkin:~$ git config --global core.autocrlf input  
pavel-fedotkin@pavel-fedotkin:~$ git config --global core.safecrlf warn  
pavel-fedotkin@pavel-fedotkin:~$
```

Рис. 2.2: Параметры репозитория

Создаем SSH ключи


```

pavelfedotkin@pavelfedotkin:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/pavelfedotkin/.ssh/id_rsa):
Created directory '/home/pavelfedotkin/.ssh'.
Enter passphrase for "/home/pavelfedotkin/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pavelfedotkin/.ssh/id_rsa
Your public key has been saved in /home/pavelfedotkin/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:q4Z/RFnNu/jAAR2lPPZ3oNjElMOisKBo0klIa/z+5PM pavelfedotkin@pavelfedotkin
The key's randomart image is:
+---[RSA 4096]-----+
| .o      ..*o.  |
| o o . . .++*  |
| B o . o ==.oo. |
| +.= . +...*.. |
| o . .S..o+... |
| . . .+ .. . |
| . o .. o  |
| =.... . |
| ==E  |
+----[SHA256]-----+
pavelfedotkin@pavelfedotkin:~$

```

Рис. 2.3: rsa-4096

```

[snr259]
pavelfedotkin@pavelfedotkin:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/pavelfedotkin/.ssh/id_ed25519):
Enter passphrase for "/home/pavelfedotkin/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pavelfedotkin/.ssh/id_ed25519
Your public key has been saved in /home/pavelfedotkin/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:RVnrKhTiaGFewvR+9ITNkWaugBgrAoIF7q/HeSI4jqw pavelfedotkin@pavelfedotkin
The key's randomart image is:
+--[ED25519 256]--+
|00.0= . 0+. |
|= .0.B . ==. . |
|0. ++. + += . |
|+ 0.... =.. |
|.0 ..S.. . |
|. . 0. . |
|. ... . . |
|=..= . . |
|E=0 0 |
+----[SHA256]-----+
pavelfedotkin@pavelfedotkin:~$ █

```

Рис. 2.4: ed25519

Создаем GPG ключ

```

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: pavel-fedotkin
Адрес электронной почты: 1032244328@pfur.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
    "pavel-fedotkin <1032244328@pfur.ru>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? O
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
gpg: /home/pavelfedotkin/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/home/pavelfedotkin/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/pavelfedotkin/.gnupg/openpgp-revocs.d/6D0E41F876C3C33D0236C2D86B8D86DA4D4A889
6.rev'.
открытый и секретный ключи созданы и подписаны.


pub  rsa4096 2025-09-06 [SC]
      6D0E41F876C3C33D0236C2D86B8D86DA4D4A8896
uid          pavel-fedotkin <1032244328@pfur.ru>
sub  rsa4096 2025-09-06 [E]

pavelfedotkin@pavelfedotkin:~$

```

Рис. 2.5: GPG ключ

Добавляем GPG ключ в аккаунт



pavel-fedotkin (pavel-fedotkin)
Your personal account

Go to your personal profile

Public profile

Account

Appearance

Accessibility

Notifications

Access

Billing and licensing

Emails

Password and authentication

Sessions

SSH and GPG keys

Organizations

Enterprises

Moderation

SSH keys

New SSH key


There are no SSH keys associated with your account.
Check out our guide to [connecting to GitHub using SSH keys](#) or troubleshoot [common SSH problems](#).

GPG keys

New GPG key

This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.

1



Email address: 1032244328@pfur.ru

Key ID: 680086DA404A8896

Subkeys: 4585376B8AED4E70

Added on Sep 6, 2025

Delete

Learn how to [generate a GPG key and add it to your account](#).

Рис. 2.6: GPG ключ

Настройка автоматических подписей коммитов git

12

```

pavelfedotkin@pavelfedotkin:~$ 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: 1 подписанных: 0 доверие: 0-, 0q, 0n, 0m, 0f, 1u
[keyboard]
-----
sec  rsa4096/6B8D86DA4D4A8896 2025-09-06 [SC]
     6D0E41F876C3C3D0236C2D86B8D86DA4D4A8896
uid          [ абсолютно ] pavelfedotkin <1032244328@pfur.ru>
ssb  rsa4096/4585376B9AED4E70 2025-09-06 [E]

pavelfedotkin@pavelfedotkin:~$
pavelfedotkin@pavelfedotkin:~$
pavelfedotkin@pavelfedotkin:~$ gpg --armor --export 6B8D86DA4D4A8896 | xclip -sel clip
pavelfedotkin@pavelfedotkin:~$ git config --global user.signingkey 6B8D86DA4D4A8896
pavelfedotkin@pavelfedotkin:~$ git config --global commit.gpgsign true
pavelfedotkin@pavelfedotkin:~$ git config --global gpg.program $(which gpg2)
pavelfedotkin@pavelfedotkin:~$

```

Рис. 2.7: Параметры репозитория

Настройка gh

```
pavelfedotkin@pavelfedotkin:~$ 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/pavelfedotkin/.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: 3323-983A
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/pavelfedotkin/.ssh/id_rsa.pub
✓ Logged in as pavel-fedotkin
pavelfedotkin@pavelfedotkin:~$
```

Рис. 2.8: Связь репозитория с аккаунтом

Загрузка шаблона репозитория и синхронизация

```
pavelfedotkin@pavelfedotkin:~$  
pavelfedotkin@pavelfedotkin:~$ mkdir -p ~/work/study/2024-2025/"Операционные системы"  
pavelfedotkin@pavelfedotkin:~$ cd ~/work/study/2024-2025/"Операционные системы"  
pavelfedotkin@pavelfedotkin:~/work/study/2024-2025/Операционные системы$ gh repo create os-intro --template=yamadharma/course-directory-student-template --public  
✓ Created repository pavel-fedotkin/os-intro on GitHub  
https://github.com/pavel-fedotkin/os-intro  
pavelfedotkin@pavelfedotkin:~/work/study/2024-2025/Операционные системы$ git clone --recursive git@github.com:pavel-fedotkin/os-intro.git os-intro  
Клонирование в «os-intro»...  
The authenticity of host 'github.com (140.82.121.4)' can't be established.  
ED25519 key fingerprint is SHA256:+DiY3wvV6TujhbpZisF/zLDA0zPMSvHdkr4UvCOqJ.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Рис. 2.9: Загрузка шаблона

Подготовка репозитория и коммит изменений

```

remote: Total 207 (delta 92), reused 170 (delta 55), pack-reused 0 (from 0)
Получение объектов: 100% (207/207), 760.70 КиБ | 3.73 МБ/с, готово.
Определение изменений: 100% (92/92), готово.
Submodule path 'template/presentation': checked out '645759e4b104e93753637dedf8109adf24d071b7'
Submodule path 'template/report': checked out 'cbca6e80e9c8e9b190a4bbbb8595f82335ae634a'
pavelfedotkin@pavelfedotkin:~/work/study/2024-2025/Операционные системы$
pavelfedotkin@pavelfedotkin:~/work/study/2024-2025/Операционные системы$
pavelfedotkin@pavelfedotkin:~/work/study/2024-2025/Операционные системы$ cd ~/work/study/2024-2025/"Операционные системы"/os-intro
pavelfedotkin@pavelfedotkin:~/work/study/2024-2025/Операционные системы/os-intro$ make COURSE=os-intro prepare
pavelfedotkin@pavelfedotkin:~/work/study/2024-2025/Операционные системы/os-intro$ rm package.json
pavelfedotkin@pavelfedotkin:~/work/study/2024-2025/Операционные системы/os-intro$ ls
COURSE LICENSE prepare project-personal README.git-flow.md template
labs Makefile presentation README.en.md README.md
pavelfedotkin@pavelfedotkin:~/work/study/2024-2025/Операционные системы/os-intro$

```

Рис. 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 сливается 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 - запустить 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: