

Отчёт по лабораторной работе №2

Управление версиями

Давыдов Сергей Арсентьевич

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Вывод	9
4	Контрольные вопросы	10

List of Figures

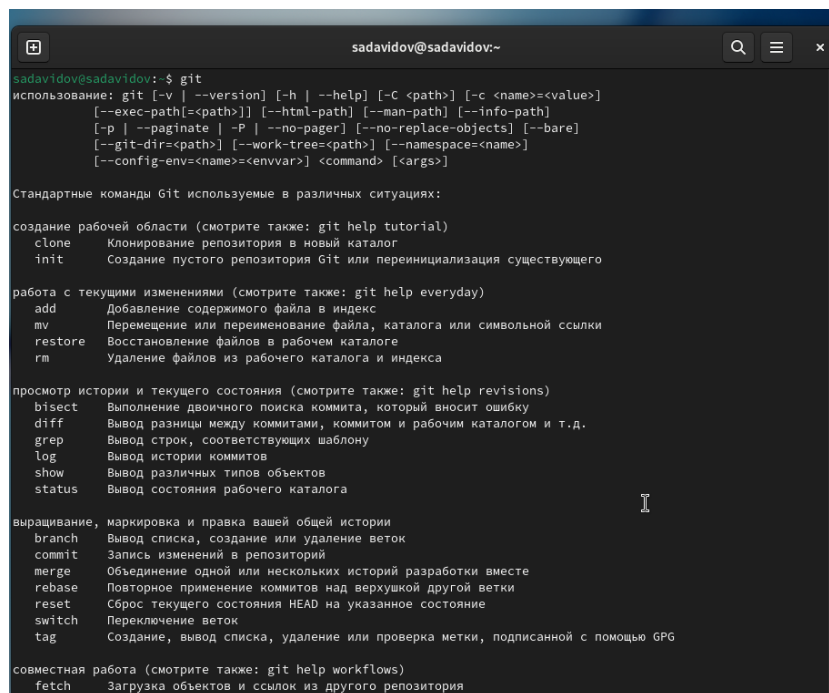
2.1	Загрузка пакетов	5
2.2	Параметры репозитория	5
2.3	rsa-4096	6
2.4	ed25519	6
2.5	GPG ключ	7
2.6	GPG ключ	7
2.7	Параметры репозитория	7
2.8	Связь репозитория с аккаунтом	8
2.9	Загрузка шаблона	8
2.10	Первый коммит	8

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.



```
sadavidov@sadavidov:~$ 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] [--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

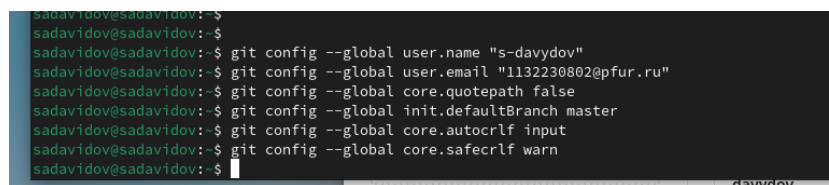
просмотр истории и текущего состояния (смотрите также: git help revisions)
bisect     Выполнение двоичного поиска коммита, который вносит ошибку
diff       Вывод разницы между коммитами, коммитом и рабочим каталогом и т.д.
grep       Вывод строк, соответствующих шаблону
log        Вывод истории коммитов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

совместная работа (смотрите также: git help workflows)
fetch      Загрузка объектов и ссылок из другого репозитория
```

Figure 2.1: Загрузка пакетов

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



```
sadavidov@sadavidov:~$
sadavidov@sadavidov:~$ git config --global user.name "s-davydov"
sadavidov@sadavidov:~$ git config --global user.email "1132230802@pfur.ru"
sadavidov@sadavidov:~$ git config --global core.quotepath false
sadavidov@sadavidov:~$ git config --global init.defaultBranch master
sadavidov@sadavidov:~$ git config --global core.autocrlf input
sadavidov@sadavidov:~$ git config --global core.safecrlf warn
sadavidov@sadavidov:~$
```

Figure 2.2: Параметры репозитория

Создаем SSH ключи

```

sadavidov@sadavidov:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/sadavidov/.ssh/id_rsa):
Created directory '/home/sadavidov/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/sadavidov/.ssh/id_rsa
Your public key has been saved in /home/sadavidov/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:ERCmuOW06gVT7GZlhh/nddm6rLm7cW/P2BiP2mWQIAo sadavidov@sadavidov
The key's randomart image is:
+---[RSA 4096]-----+
|          +o.          |
| . o o  +             |
| . E B o = .          |
| + & + o + .          |
| * 0   S o            |
| *   . . .            |
| . . . + . o          |
| . .   = o . @         |
| .   *+.o*.=          |
+---[SHA256]-----+
sadavidov@sadavidov:~$

```

Figure 2.3: rsa-4096

```

sadavidov@sadavidov:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/sadavidov/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/sadavidov/.ssh/id_ed25519
Your public key has been saved in /home/sadavidov/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:pabNJ7Z4hozxDf/ii3G3lLaMxkzA1znLqyJPOHBR0UY sadavidov@sadavidov
The key's randomart image is:
+---[ED25519 256]---+
|      o+E      |
|      . o      |
|      .. . ...  |
|      .o .o+    |
|      . . oS. o  |
|      o...=. o.  |
|      o=+0* *.   |
|      oo=*XX.o   |
|      o***==    |
+---[SHA256]-----+
sadavidov@sadavidov:~$

```

Figure 2.4: ed25519

Создаем GPG ключ

```
Срок действия ключа? (0)
Срок действия ключа не ограничен
Все верно? (y/N) y

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: s-davydov
Адрес электронной почты: 1132230802@pfur.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
"s-davydov <1132230802@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 (печатать
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 (печатать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
gpg: /home/sadavidov/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/home/sadavidov/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/sadavidov/.gnupg/openpgp-revocs.d/041B872DA85C828B6076975E27359220E8DD09E6.rev'
открытый и секретный ключи созданы и подписаны.

pub  rsa4096 2024-06-21 [SC]
     041B872DA85C828B6076975E27359220E8DD09E6
uid          s-davydov <1132230802@pfur.ru>
sub  rsa4096 2024-06-21 [E]

sadavidov@sadavidov:~$
```

Figure 2.5: GPG ключ

Добавляем GPG ключ в аккаунт

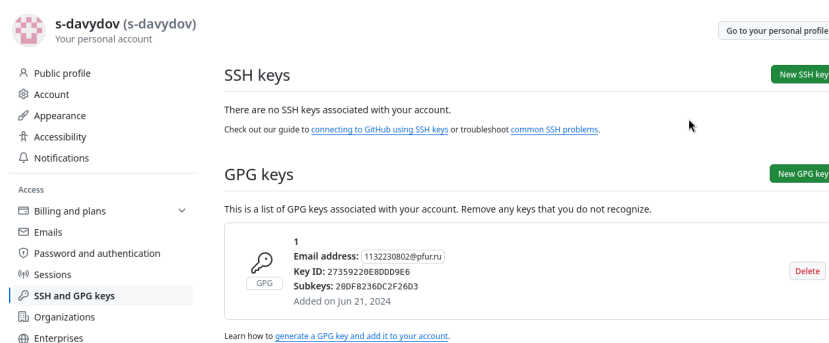


Figure 2.6: GPG ключ

Настройка автоматических подписей коммитов git

```
sadavidov@sadavidov:~$
sadavidov@sadavidov:~$
sadavidov@sadavidov:~$ git config --global user.signingkey 27359220E8DD09E6
sadavidov@sadavidov:~$ git config --global commit.gpgsign true
sadavidov@sadavidov:~$ git config --global gpg.program $(which gpg2)
sadavidov@sadavidov:~$
```

Figure 2.7: Параметры репозитория

Настройка gh

```
sadavidov@sadavidov:~$ git config --global user.signingkey 27359220E8DD9E6
sadavidov@sadavidov:~$ git config --global commit.gpgsign true
sadavidov@sadavidov:~$ git config --global gpg.program $(which gpg2)
sadavidov@sadavidov:~$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /home/sadavidov/.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: 9890-9FCB
Press Enter to open github.com 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/sadavidov/.ssh/id_rsa.pub
✓ Logged in as s-davydov
sadavidov@sadavidov:~$
```

Figure 2.8: Связь репозитория с аккаунтом

Загрузка шаблона репозитория и синхронизация

```
remote: Compressing objects: 100% (67/67), done.
remote: Total 95 (delta 34), reused 87 (delta 26), pack-reused 0
Получение объектов: 100% (95/95), 96.99 КиБ | 1.07 МБ/с, готово.
Определение изменений: 100% (34/34), готово.
Клонирование в «/home/sadavidov/work/study/2023-2024/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 126, done.
remote: Counting objects: 100% (126/126), done.
remote: Compressing objects: 100% (87/87), done.
remote: Total 126 (delta 52), reused 108 (delta 34), pack-reused 0
Получение объектов: 100% (126/126), 335.80 КиБ | 2.13 МБ/с, готово.
Определение изменений: 100% (52/52), готово.
Submodule path 'template/presentation': checked out '40a1761813e197d00e8443ff1ca72c60a304f24c'
Submodule path 'template/report': checked out '7c31ab8e5dfa8c2d67caeb8a19ef8028ced88e'
sadavidov@sadavidov:~/work/study/2023-2024/Операционные системы$
sadavidov@sadavidov:~/work/study/2023-2024/Операционные системы$
sadavidov@sadavidov:~/work/study/2023-2024/Операционные системы$ cd ~/work/study/2023-2024/"Операционные системы
"/os-intro
sadavidov@sadavidov:~/work/study/2023-2024/Операционные системы/os-intro$ rm package.json
sadavidov@sadavidov:~/work/study/2023-2024/Операционные системы/os-intro$ make COURSE=os-intro prepare
sadavidov@sadavidov:~/work/study/2023-2024/Операционные системы/os-intro$ ls
CHANGELOG.md  COURSE  LICENSE  prepare  project-personal  README.git-flow.md  template
config        laba    Makefile  presentation  README.en.md  README.md
sadavidov@sadavidov:~/work/study/2023-2024/Операционные системы/os-intro$
```

Figure 2.9: Загрузка шаблона

Подготовка репозитория и коммит изменений

```
create mode 100644 project-personal/stage6/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_fignos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_fignos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_sagynos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_sagynos.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/core.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/main.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 project-personal/stage6/report/report.md
sadavidov@sadavidov:~/work/study/2023-2024/Операционные системы/os-intro$ git push
Перечисление объектов: 38, готово.
Подсчет объектов: 100% (38/38), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (38/38), готово.
Запись объектов: 100% (37/37), 342.06 КиБ | 2.74 МБ/с, готово.
Total 37 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:s-davydov/os-intro.git
7dd9e8c..e4bf128 master -> master
sadavidov@sadavidov:~/work/study/2023-2024/Операционные системы/os-intro$
```

Figure 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: