

Отчёт по лабораторной работе №2

Управление версиями

Сардор Турсунбоев

Содержание

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

List of Figures

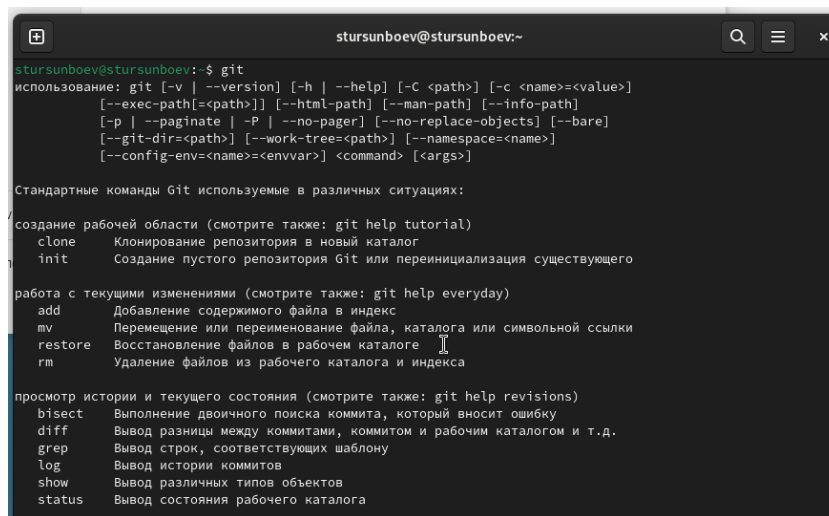
2.1	Загрузка пакетов	5
2.2	Параметры репозитория	5
2.3	rsa-4096	6
2.4	ed25519	6
2.5	GPG ключ	7
2.6	GPG ключ	7
2.7	Параметры репозитория	7
2.8	Связь репозитория с аккаунтом	8
2.9	Загрузка шаблона	8
2.10	Первый коммит	8

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.



```
stursunboev@stursunboev:~$ 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:

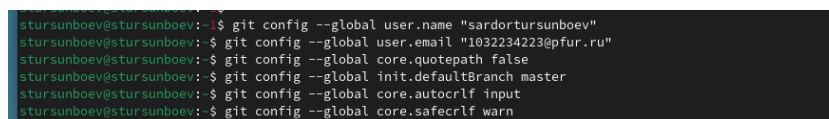
создание рабочей области (смотрите также: git help tutorial)
clone      Клонирование репозитория в новый каталог
init       Создание пустого репозитория Git или переинициализация существующего

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
```

Figure 2.1: Загрузка пакетов

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



```
stursunboev@stursunboev:~$ git config --global user.name "sardortursunboev"
stursunboev@stursunboev:~$ git config --global user.email "1032234223@pfur.ru"
stursunboev@stursunboev:~$ git config --global core.quotepath false
stursunboev@stursunboev:~$ git config --global init.defaultBranch master
stursunboev@stursunboev:~$ git config --global core.autocrlf input
stursunboev@stursunboev:~$ git config --global core.safecrlf warn
```

Figure 2.2: Параметры репозитория

Создаем SSH ключи

```

stursunboev@stursunboev:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/stursunboev/.ssh/id_rsa):
Created directory '/home/stursunboev/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/stursunboev/.ssh/id_rsa
Your public key has been saved in /home/stursunboev/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:p9301GfBPP320wVEZhK/RTDftqDWTLbd50bsiSkN9Ww stursunboev@stursunboev
The key's randomart image is:
+---[RSA 4096]-----+
|          oOB..|
|          *O+O|
|          =O=*|
|          B *B*|
|      S .+.oE%|
|          +.000+*B|
|          . ...+.O*|
|          . ..|
|          |
+---[SHA256]-----+
stursunboev@stursunboev:~$

```

Figure 2.3: rsa-4096

```

stursunboev@stursunboev:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/stursunboev/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/stursunboev/.ssh/id_ed25519
Your public key has been saved in /home/stursunboev/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:/3l405I4hJthBtI7ERMFmFExoN3oY/to+bFkveJB898 stursunboev@stursunboev
The key's randomart image is:
+---[ED25519 256]---+
|      BB+      |
|      oo= o     |
|      ..000     |
|      ..+       |
|      .S+o.     |
|      .o==+.    |
|      .*=00 o   |
|      o=+=++*.. |
|      ..o=.++oE |
+---[SHA256]-----+
stursunboev@stursunboev:~$

```

Figure 2.4: ed25519

Создаем GPG ключ

```
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/stursunboev/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/home/stursunboev/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/stursunboev/.gnupg/openpgp-revocs.d/8AB03DE69A559479D33B0A6D2C4CC8A9D2D06734.rev'.
спол: открытый и секретный ключи созданы и подписаны.

pub  rsa4096 2024-02-22 [SC]
     8AB03DE69A559479D33B0A6D2C4CC8A9D2D06734
uid          sardortursunboev <1032234223@pfur.ru>
sub  rsa4096 2024-02-22 [E]

stursunboev@stursunboev:~$ 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, 0f, 1u
[keyboxd]
-----
sec  rsa4096/2C4CC8A9D2D06734 2024-02-22 [SC]
     8AB03DE69A559479D33B0A6D2C4CC8A9D2D06734
uid          [ абсолютно ] sardortursunboev <1032234223@pfur.ru>
ssb  rsa4096/897351848C72B4CC 2024-02-22 [E]

stursunboev@stursunboev:~$
```

Figure 2.5: GPG ключ

Добавляем GPG ключ в аккаунт

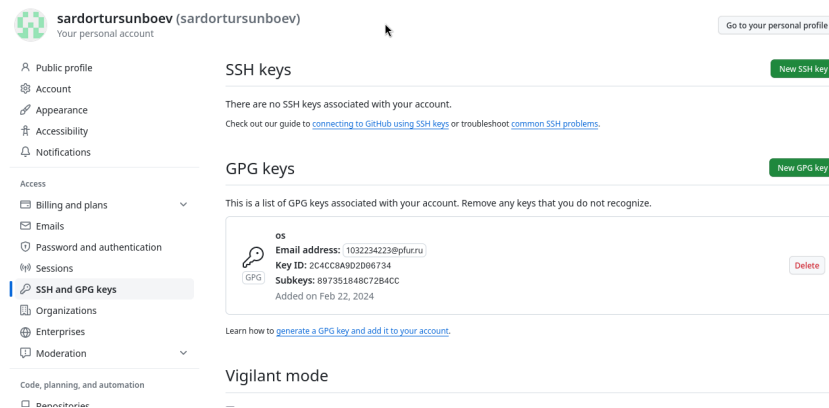


Figure 2.6: GPG ключ

Настройка автоматических подписей коммитов git

```
UN31E0HC1YU1YLOqBXN9NVG21TQETMRZT0TWZCU40ZKYUJ1G16T1SWZZXCSd6UV
eZME2Q==
=Dv8
-----END PGP PUBLIC KEY BLOCK-----
stursunboev@stursunboev:~$
stursunboev@stursunboev:~$
stursunboev@stursunboev:~$ git config --global user.signingkey 2C4CC8A9D2D06734
stursunboev@stursunboev:~$ git config --global commit.gpgsign true
stursunboev@stursunboev:~$ git config --global gpg.program $(which gpg2)
stursunboev@stursunboev:~$
```

Figure 2.7: Параметры репозитория

Настройка gh

```
stursunboev@stursunboev:~$ 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/stursunboev/.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: 7607-52C1
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/stursunboev/.ssh/id_rsa.pub
✓ Logged in as sardortursunboev
stursunboev@stursunboev:~$ mkdir -p ~/work/study/2023-2024/"Операционные системы"
stursunboev@stursunboev:~$ cd ~/work/study/2023-2024/"Операционные системы"
stursunboev@stursunboev:~/work/study/2023-2024/Операционные системы$ gh repo create os-intro --template=yam
adharma/course-directory-student-template --public
✓ Created repository sardortursunboev/os-intro on GitHub
stursunboev@stursunboev:~/work/study/2023-2024/Операционные системы$
```

Figure 2.8: Связь репозитория с аккаунтом

Загрузка шаблона репозитория и синхронизация

```
remote: Total 126 (delta 52), reused 108 (delta 34), pack-reused 0
Получение объектов: 100% (126/126), 335.80 КиБ | 2.42 МиБ/с, готово.
Определение изменений: 100% (52/52), готово.
Submodule path 'template/presentation': checked out '40a1761813e197d0e8443ff1ca72c60a304f24c'
Submodule path 'template/report!': checked out '7c31ab8e5dfa8cdb2d67cae88a19ef8028ced88e'
stursunboev@stursunboev:~/work/study/2023-2024/Операционные системы$ cd ~/work/study/2023-2024/"Операционны
е системы"/os-intro
stursunboev@stursunboev:~/work/study/2023-2024/Операционные системы/os-intro$ rm package.json
stursunboev@stursunboev:~/work/study/2023-2024/Операционные системы/os-intro$ ls
CHANGELOG.md  config  COURSE  LICENSE  Makefile  README.en.md  README.git-flow.md  README.md  template
stursunboev@stursunboev:~/work/study/2023-2024/Операционные системы/os-intro$ make COURSE=os-intro prepare
stursunboev@stursunboev:~/work/study/2023-2024/Операционные системы/os-intro$ ls
CHANGELOG.md  COURSE  LICENSE  prepare  project-personal  README.git-flow.md  README.md  template
config  labs  Makefile  presentation  README.en.md  README.md
stursunboev@stursunboev:~/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/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
stursunboev@stursunboev:~/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% (30/30), готово.
Запись объектов: 100% (37/37), 342.07 КиБ | 3.32 МиБ/с, готово.
Всего 37 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:sardortursunboev/os-intro.git
cccd8ee..71f3987 master -> master
stursunboev@stursunboev:~/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: