

Отчёт по лабораторной работе №2

Управление версиями

Амина Усманова

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.

```

abusmanova@abusmanova:~$ 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

```

Рис. 2.1: Загрузка пакетов

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

```
abusmatkova@abusmatkova:~$  
abusmatkova@abusmatkova:~$ git config --global user.name "abusby"  
abusmatkova@abusmatkova:~$ git config --global user.email "1132246823@rudn.university"  
abusmatkova@abusmatkova:~$ git config --global core.quotepath false  
abusmatkova@abusmatkova:~$ git config --global init.defaultBranch master  
abusmatkova@abusmatkova:~$ git config --global core.autocrlf input  
abusmatkova@abusmatkova:~$ git config --global core.safecrlf warn  
abusmatkova@abusmatkova:~$
```

Рис. 2.2: Параметры репозитория

Создаем SSH ключи


```

abusmanova@abusmanova:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/abusmanova/.ssh/id_rsa):
Created directory '/home/abusmanova/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/abusmanova/.ssh/id_rsa
Your public key has been saved in /home/abusmanova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:TYTgwnVoZCmJoBus9PUW5ZUUCxseKtEPKD9bkRAnfPg abusmanova@abusmanova
The key's randomart image is:
+---[RSA 4096]-----+
|.. ..B@=o*o+o |
|o .ooBBB*.*.. |
|oo ==+o++.. |
|oo. .+oE.+ |
|o . +oS . |
| .. |
| |
| |
+---[SHA256]-----+
abusmanova@abusmanova:~$ █

```

Рис. 2.3: rsa-4096

```

abusmanova@abusmanova:~$
abusmanova@abusmanova:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/abusmanova/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/abusmanova/.ssh/id_ed25519
Your public key has been saved in /home/abusmanova/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:cUKyzoVEC1TjtfIYe0NnQ+HD/TjujLPjBkBo508+oBs abusmanova@abusmanova
The key's randomart image is:
+--[ED25519 256]--+
|  o++* +          |
|  .o=.@ +         |
|  .o.X X o        |
|    *.0 = o       |
|    *.S o .       |
|   .. 0.00.       |
|  E. .. +..       |
|  .. .. .00       |
|  .. ...+0        |
+----[SHA256]-----+
abusmanova@abusmanova:~$ █

```

Рис. 2.4: ed25519

Создаем GPG ключ

```
abusmanova@abusmanova:~$  
abusmanova@abusmanova:~$ gpg --full-generate-key  
gpg (GnuPG) 2.4.5; Copyright (C) 2024 g10 Code GmbH  
This is free software: you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law.  
  
gpg: создан каталог '/home/abusmanova/.gnupg'  
Выберите тип ключа:  
  (1) RSA and RSA  
  (2) DSA and Elgamal  
  (3) DSA (sign only)  
  (4) RSA (sign only)  
  (9) ECC (sign and encrypt) *default*  
 (10) ECC (только для подписи)  
 (14) Existing key from card  
Ваш выбор? 1  
длина ключей RSA может быть от 1024 до 4096.  
Какой размер ключа Вам необходим? (3072) 4096  
Запрошенный размер ключа - 4096 бит  
Выберите срок действия ключа.  
  0 = не ограничен  
  <n> = срок действия ключа - n дней  
  <n>w = срок действия ключа - n недель  
  <n>m = срок действия ключа - n месяцев  
  <n>y = срок действия ключа - n лет  
Срок действия ключа? (0) 0  
Срок действия ключа не ограничен  
Все верно? (y/N) y
```

Рис. 2.5: GPG ключ

Добавляем GPG ключ в аккаунт

```

abusmanova@abusmanova:~$
abusmanova@abusmanova:~$ 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  rsa4096/39550FF62A411C5C 2025-02-13 [SC]
      98602E4CF29A38AE04FF080A39550FF62A411C5C
uid          [ абсолютно ] amusby <1132246823@rudn.university>
ssb  rsa4096/4E4295DE5C71F0FA 2025-02-13 [E]

abusmanova@abusmanova:~$
abusmanova@abusmanova:~$
abusmanova@abusmanova:~$ gpg --armor --export 39550FF62A411C5C
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGetu1QBEACxfYFe5MYL3kGRJeHReFMhNDzC9NYtvMcupa0EyQGSrr5V5hY0
5bjL8ErzdXwoth3z4Cyt8rifGpqvCKbhG2QLxA4KY9hncTJlTi+w40trxpQUc9by
AyAAIvPQ3GTyG9rqL/5GAGgT3cE+WacFjuCGDugy/dHvekRQTiMnW0xOrPegs4H0
ftT4WUm29R7h23Ssbmxt+SmKj3x6QlyKChC3bJCbDybu1kGIM3siLhqr1gRiyUY
GQ3GFu3lEtmcgf1NChjGKOUe0tugrUKK9H8iNavs1x1I5MwXwccSAYk/DP2T7c6k
-----END PGP PUBLIC KEY BLOCK-----

```

Рис. 2.6: GPG ключ

Настройка автоматических подписей коммитов git

```

VqK9dh105fxrVWFQyIQqasgmFx9bYFYHxDA1ehWjhAwSp7A1F0Mdx/0PCh2Ljnv
G Mv9lGw2h6bdhjuPwjcewh0nJd6+JUmDnz1awxRq+46nTyJHKwr4lzd/nH0eEF6Fn
pVQ7UMS4EvM82o1CAojoJMP0oPoiaCw2H79/PdVSfrQ/1gFHder4hDHTk/3pBpp
Q3IH816lHQQRqz2Uwl/3br0detcV0c0Pgobr200g81e4pbK+VbkHqic8iLIE6rd
DSKpsvDF4cDNnX+xoN911RZvVwSuqfkvzBgf8UGQK2hiHcf5x9t15FtA0eG8LHxp
yxk=
=0+NE
-----END PGP PUBLIC KEY BLOCK-----
abusmanova@abusmanova:~$
abusmanova@abusmanova:~$
abusmanova@abusmanova:~$ git config --global user.signingkey 39550FF62A411C5C
abusmanova@abusmanova:~$ git config --global commit.gpgsign true
abusmanova@abusmanova:~$ git config --global gpg.program $(which gpg2)
abusmanova@abusmanova:~$

```

Рис. 2.7: Параметры репозитория

Настройка gh

```
abusmanova@abusmanova:~$  
abusmanova@abusmanova:~$ 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/abusmanova/.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: 5A83-30A7  
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/abusmanova/.ssh/id_rsa.pub  
✓ Logged in as amusby  
abusmanova@abusmanova:~$
```

Рис. 2.8: Связь репозитория с аккаунтом

Загрузка шаблона репозитория и синхронизация

```

abusmanova@abusmanova:~$
abusmanova@abusmanova:~$ mkdir -p ~/work/study/2024-2025/"Операционные системы"
abusmanova@abusmanova:~$ cd ~/work/study/2024-2025/"Операционные системы"
abusmanova@abusmanova:~/work/study/2024-2025/Операционные системы$ gh repo create os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository amusby/os-intro on GitHub
https://github.com/amusby/os-intro
abusmanova@abusmanova:~/work/study/2024-2025/Операционные системы$ git clone --recursive git@github.com:amusby/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:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.

```

Рис. 2.9: Загрузка шаблона

Подготовка репозитория и коммит изменений

```

create mode 100644 project-personal/stage6/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
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_secnos.py
create mode 100755 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
abusmanova@abusmanova:~/work/study/2024-2025/Операционные системы/os-intro$ git push
Перечисление объектов: 38, готово.
Подсчет объектов: 100% (38/38), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (37/37), 342.27 КБ | 2.18 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:amusby/os-intro.git
   c3f91d1..46d97ba master -> master
abusmanova@abusmanova:~/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: