## Отчёт по лабораторной работе №2

Управление версиями

Субанов Юсуф Жура угли

### 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 c git.

# 2 Выполнение лабораторной работы

Устанавливаем git, git-flow и gh.

```
yusufsubanov@yusufsubanov:~$ git
использование: git [-v | --version] [-h | --help] [-С <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
            Создание пустого репозитория Git или переинициализация существующего
работа с текущими изменениями (смотрите также: git help everyday)
            Добавление содержимого файла в индекс
            Перемещение или переименование файла, каталога или символьной ссылки
  restore Восстановление файлов в рабочем каталоге
            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
            Вывод разницы между коммитами, коммитом и рабочим каталогом и т.д.
            Вывод строк, соответствующих шаблону
  grep
            Вывод истории коммитов
            Вывод различных типов объектов
  show
  status Вывод состояния рабочего каталога
```

Рис. 2.1: Загрузка пакетов

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



Рис. 2.2: Параметры репозитория

Создаем SSH ключи

```
yusufsubanov@yusufsubanov:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/yusufsubanov/.ssh/id_rsa):
/home/yusufsubanov/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase for "/home/yusufsubanov/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/yusufsubanov/.ssh/id_rsa
Your public key has been saved in /home/yusufsubanov/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:ruilzuGXx7rNQr3sq7CdEWcoDFiaSzMQbiah35QbvuE yusufsubanov@yusufsubanov
The key's randomart image is:
+---[RSA 4096]----+
 |*B. +
 =00* 0 .
 . . B o.S
     E.o+..
    0 0+*=
    oO.BB=.
 ----[SHA256]----+
yusufsubanov@yusufsubanov:~$
```

Рис. 2.3: rsa-4096

```
yusufsubanov@yusufsubanov:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/yusufsubanov/.ssh/id_ed25519):
/home/yusufsubanov/.ssh/id_ed25519 already exists.
Overwrite (y/n)? y
Enter passphrase for "/home/yusufsubanov/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/yusufsubanov/.ssh/id_ed25519
Your public key has been saved in /home/yusufsubanov/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:b1fhPLId096uPmFDcloYt/P0TYhVC24sAdgpJ8HPelw yusufsubanov@yusufsubanov
The key's randomart image is:
+--[ED25519 256]--+
         o .oE*oo
         S o **0==|
         0 +.0=*=
          + =+.=0
              .00.
 ----[SHA256]----+
yusufsubanov@yusufsubanov:~$
```

Рис. 2.4: ed25519

#### Создаем GPG ключ

```
GnuPG должен составить идентификатор пользователя для идентификации ключа.
Baшe полное имя: yusufsubanov
Адрес электронной почты: 1032214217@rudn.university
Примечание:
Вы выбрали следующий идентификатор пользователя:
    "yusufsubanov <1032214217@rudn.university>"
Сменить (N)Имя, (C)Примечание, (E)Адрес; (О)Принять/(Q)Выход? О
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: сертификат отзыва записан в '/home/yusufsubanov/.gnupg/openpgp-revocs.d/5CFAF5B96A1ADB3DF506023096BAC0F
D5B1228F2.rev'.
открытый и секретный ключи созданы и подписаны.
pub rsa4096 2025-05-29 [SC]
      5CFAF5B96A1ADB3DF506023096BAC0FD5B1228F2
                         yusufsubanov <1032214217@rudn.university>
sub rsa4096 2025-05-29 [E]
yusufsubanov@yusufsubanov:~$
```

Рис. 2.5: GPG ключ

#### Добавляем GPG ключ в аккаунт

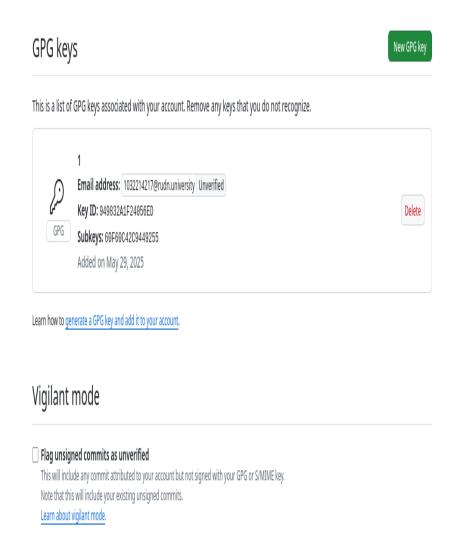


Рис. 2.6: GPG ключ

Настройка автоматических подписей коммитов git

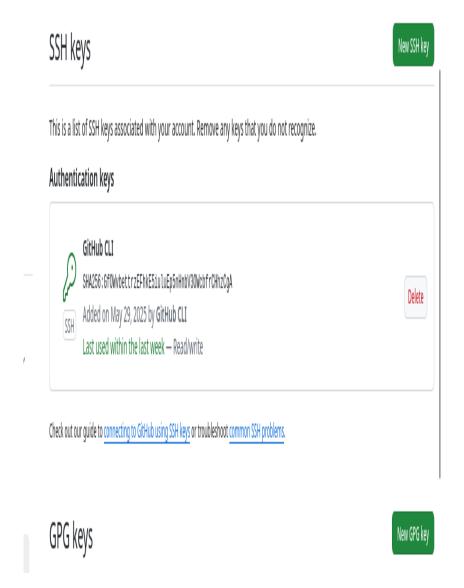


Рис. 2.7: Параметры репозитория

Настройка gh

```
yusufsubanov@yusufsubanov:~$ 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/yusufsubanov/.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: 9608-BC91
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/yusufsubanov/.ssh/id_rsa.pub
 Logged in as yusufsubanov
 You were already logged in to this account
yusufsubanov@yusufsubanov:~$
```

Рис. 2.8: Связь репозитория с аккаунтом

Загрузка шаблона репозитория и синхронизация

yusufsubanov feat(main): make course structure 80f35a6 · 17 minutes ago 🐧 2 Commits config 18 minutes ago Initial commit labs feat(main): make course structure 17 minutes ago presentation feat(main): make course structure 17 minutes ago project-personal feat(main): make course structure 17 minutes ago **template** Initial commit 18 minutes ago gitattributes Initial commit 18 minutes ago .gitignore Initial commit 18 minutes ago gitmodules [ 18 minutes ago Initial commit CHANGELOG.md Initial commit 18 minutes ago COURSE Initial commit 18 minutes ago LICENSE Initial commit 18 minutes ago ☐ Makefile Initial commit 18 minutes ago README.en.md Initial commit 18 minutes ago

Рис. 2.9: Загрузка шаблона

#### Подготовка репозитория и коммит изменений

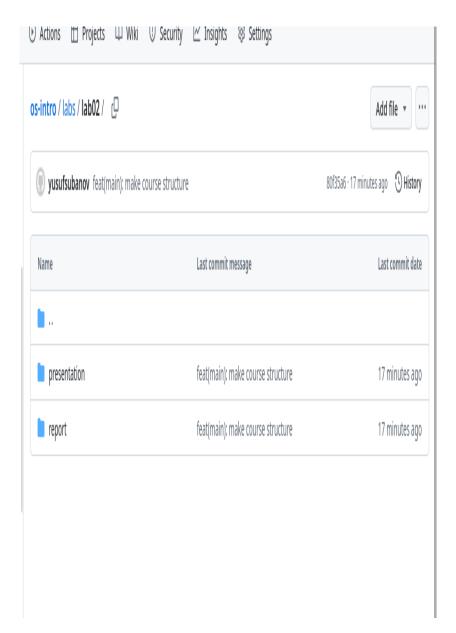


Рис. 2.10: Первый коммит

## 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 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: