Отчёт по лабораторной работе №2

Управление версиями

Геллер Михаил Андреевич

Содержание

1 Цель работы **5**

2 Выполнение лабораторной работы **6**

3 Вывод **10**

4 Контрольные вопросы **11**

Список иллюстраций

2.1 Загрузка пакетов 6
2.2 Параметры репозитория 6
2.3 rsa-4096 7
2.4 ed25519 7
2.5 GPG ключ 8
2.6 GPG ключ 8
2.7 Параметры репозитория 8
2.8 Связь репозитория с аккаунтом 9
2.9 Загрузка шаблона 9
2.10 Первый коммит 9

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.

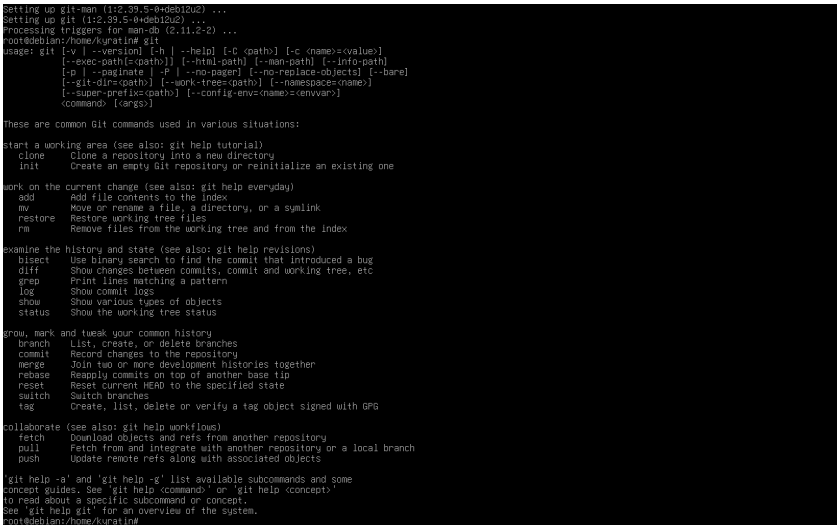

Рис. 2.1: Загрузка пакетов

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

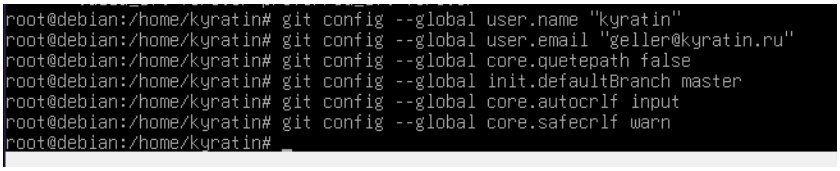

Рис. 2.2: Параметры репозитория

Создаем SSH ключи

```
root@debian:/home/kyratin# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:tvvKQexaDOppm8ncELM1sPExTnE3adREekgrnR87S5I root@debian
The key's randomart image is:
+---[RSA 4096]----+
|       . ..==o    |
|        o +o*.    |
|    o + ..* o     |
|     B + . + o    |
|    + * S E =     |
|     * B . o o    |
|    +   *   .     |
|   +.* + o        |
|   .O.o +o.       |
+----[SHA256]-----+
root@debian:/home/kyratin#
```

Рис. 2.3: rsa-4096

Рис. 2.4: ed25519

Создаем GPG ключ

```
root@debian:/home/kyratin# gpg --list-secret-keys
/root/.gnupg/pubring.kbx
------------------------
sec   rsa3072 2025-06-16 [SC]
      066B56D1B6A6287F6AEA1750CC745529D6E23236
uid           [ultimate] kyratin (test) <geller@kyratin.ru>
ssb   rsa3072 2025-06-16 [E]

root@debian:/home/kyratin# gpg --armor --export geller@kyratin.ru
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQGNBGhQnLkBDACjXmfzCtxADg1v5bMATbdQkJJpCVml5HF08qf3OQfnsivtTFcn
m2zbrizuTfYK1YHDAZ4XscKIKQZZHmuqBQAzRG6tvAhC427iiXzDEazAET9frOFV
0twEYMZK1FsF+9cd3V4h1bQFYs+iUmGPbq4FTm0hi+qLzlcW0e8n20ujr8AuIsEb
tsHgp5ilQIad45lUAlsqqp1zq+LYsfiVRO9Al+a/Zfmm8oHkFzbodBP4jG/Pj7mK
Ngw50lUU4zKvMMgNWgOOJ9oR2lEXn57deXWMf8ZmruF9WXO44h1WvfuHZGbFmvyE
zDNZzluTpNm6aKd3WP0nCDqTLYmR9/DLZXo184T4tlriIZdeJSTt64ilgbGL6s3c
```

Рис. 2.5: GPG ключ

Добавляем GPG ключ в аккаунт

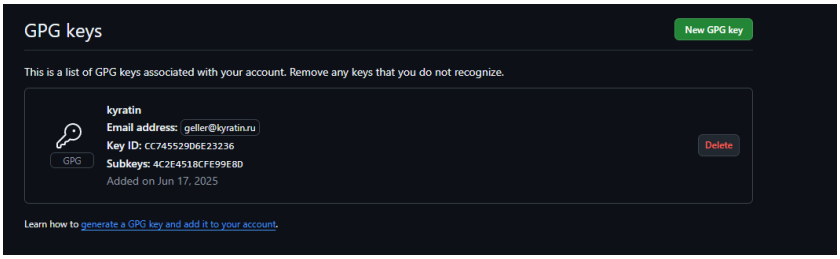

Рис. 2.6: GPG ключ

Настройка автоматических подписей коммитов git

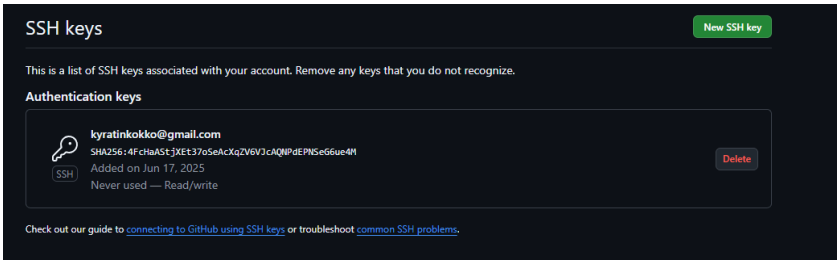

Рис. 2.7: Параметры репозитория

Настройка gh

```
root@debian:/home/kyratin# gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? SSH
? Upload your SSH public key to your GitHub account? /root/.ssh/id_rsa.pub
? Title for your SSH key: GitHub CLI
? How would you like to authenticate GitHub CLI? Paste an authentication token
Tip: you can generate a Personal Access Token here https://github.com/settings/t
okens
The minimum required scopes are 'repo', 'read:org', 'admin:public_key'.
? Paste your authentication token: ****************************************
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
✓ Uploaded the SSH key to your GitHub account: /root/.ssh/id_rsa.pub
✓ Logged in as kyratinkokko
root@debian:/home/kyratin#
```

Рис. 2.8: Связь репозитория с аккаунтом

Загрузка шаблона репозитория и синхронизация

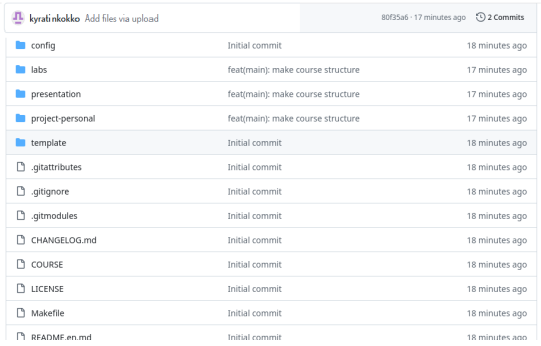

Рис. 2.9: Загрузка шаблона

Подготовка репозитория и коммит изменений

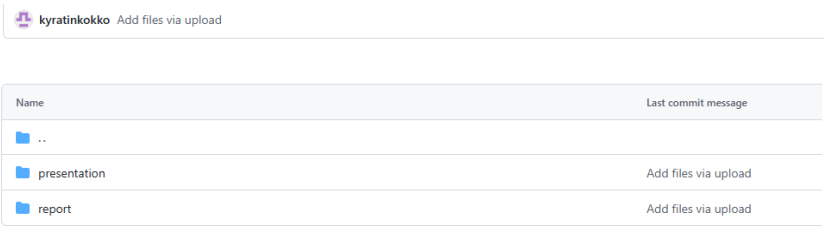

Рис. 2.10: Первый коммит

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: