

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.

```
Setting up git-man (1:2.39.5-4+deb12u2) ...
Setting up git (1:2.39.5-4+deb12u2) ...
Processing triggers for man-db (2.11.2-2) ...
root@debian:/home/kyratin# git
usage: 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>]
          [--super-prefix<path>] [--config-env=<name>]<envvar>]
          <command> [<args>]

These are common Git commands used in various situations:

Start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

Work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  restore    Restore working tree files
  rm         Remove files from the working tree and from the index

Examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  diff       Show changes between commits, commit and working tree, etc
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status

Grow, mark and tweak your common history
  branch     List, create, or delete branches
  commit     Record changes to the repository
  merge      Join two or more development histories together
  rebase     Reapply commits on top of another base tip
  reset      Reset current HEAD to the specified state
  switch     Switch branches
  tag        Create, list, delete or verify a tag object signed with GPG

Collaborate (see also: git help workflows)
  fetch      Download objects and refs from another repository
  pull       Fetch from and integrate with another repository or a local branch
  push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
root@debian:/home/kyratin#
```

Рис. 2.1: Загрузка пакетов

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

```
root@debian:/home/kyratin# git config --global user.name "kyratin"
root@debian:/home/kyratin# git config --global user.email "geller@kyratin.ru"
root@debian:/home/kyratin# git config --global core.quietpath false
root@debian:/home/kyratin# git config --global init.defaultBranch master
root@debian:/home/kyratin# git config --global core.autocrlf input
root@debian:/home/kyratin# git config --global core.safecrlf warn
root@debian:/home/kyratin#
```

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:tvvKQexaD0ppm8ncELM1sPEXtnE3adREekgrnR87S5I root@debian
The key's randomart image is:
+----[RSA 4096]-----+
|      .  .  .==0      |
|      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

```

You have not entered a passphrase - this is in general a bad idea!
Please confirm that you do not want to have any protection on your key.
Yes, protection is not needed                                <Enter new passphrase>

gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: directory '/root/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/root/.gnupg/openpgp-revocs.d/666B56D1B6A6207F6AE1750CC74552906E23236.rev'
public and secret key created and signed.

pub   rsa3072 2025-06-16 [SC]
      666B56D1B6A6207F6AE1750CC74552906E23236
uid    kyratin (test) <geller@kyratin.ru>
sub    rsa3072 2025-06-16 [E]

root@debian:/home/kyratin#

```

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-----

mQGnBGhQnLkBDACjXmfzCtxAdG1v5bMATbdQkJJpCVm15HF08qf30QfnsivtTfCn
m2zbrizuTfYK1YHDAZ4XscKIKQZZHmuqBQAzRG6tvAhC427iiXzDEazAET9fr0FV
0twEYMZK1FsF+9cd3V4h1bQFYs+iUmGPbq4FTm0hi+qLzlcW0e8n20ujr8AuIsEb
tsHgp5ilQIad45lUAlsqqp1zq+LYsfiVR09Al+a/Zfmm8oHkFzbodBP4jG/Pj7mK
Ngw50lUU4zKvMMgNWg00J9oR2lEXn57deXWMf8ZmruF9WX044h1WvfuHZGbFmvyE
zDNZzluTpNm6aKd3WP0nCDqTLYmR9/DLZXo184T4t1riIZdeJSTt64ilgbGL6s3c

```

Рис. 2.5: GPG ключ

Добавляем GPG ключ в аккаунт

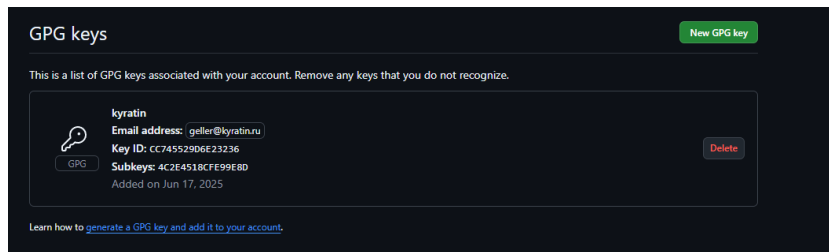


Рис. 2.6: GPG ключ

Настройка автоматических подписей коммитов git

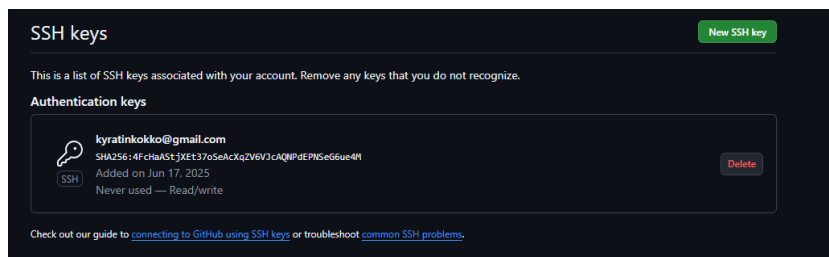


Рис. 2.7: Параметры репозитория

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/tokens
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

kyratinkokko Add files via upload		80f55a6 · 17 minutes ago	2 Commits
config	Initial commit	18 minutes ago	
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	Initial commit	18 minutes ago	
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

kyratinkokko Add files via upload	
Name	Last commit message
..	
presentation	Add files via upload
report	Add files via upload

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: