

Отчет по лабораторной работе 2

Управление версиями

Вишняков Родион Сергеевич

Содержание

| | |
|--------------------------------|----|
| Цель работы | 5 |
| Выполнение лабораторной работы | 6 |
| Вывод | 10 |
| Контрольные вопросы | 11 |

Список иллюстраций

| | |
|--|---|
| 0.1. Загрузка пакетов | 6 |
| 0.2. Параметры репозитория | 6 |
| 0.3. rsa-4096 | 7 |
| 0.4. ed25519 | 7 |
| 0.5. GPG ключ | 8 |
| 0.6. GPG ключ | 8 |
| 0.7. Параметры репозитория | 8 |
| 0.8. Связь репозитория с аккаунтом | 8 |
| 0.9. Загрузка шаблона | 9 |
| 0.10. Первый коммит | 9 |

Список таблиц

Цель работы

Целью данной работы является изучение идеологии и применения средств контроля версий и освоение умений работать с git.

Выполнение лабораторной работы

Устанавливаем git, git-flow и gh.

```
[rodvish@vbox report]$ git
usage: git [--version] [--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>]

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    Restore working tree files
  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, отметка и настройка 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
  reset      Сброс текущего состояния HEAD на указанное состояние
  switch     Switch branches
  tag        Создание, вывод списка, удаление или проверка метки, подписанной с помощью GPG

совместная работа (смотрите также: git help workflows)
  fetch      Загрузка объектов и ссылок из другого репозитория
  pull       Извлечение изменений и объединение с другим репозиторием или локальной веткой
  push       Обновление внешних ссылок и связанных объектов

'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.
```

Рис. 0.1.: Загрузка пакетов

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

```
[rodvish@vbox ~]$ git config --global user.name "Rodion Vishnyakov"
[rodvish@vbox ~]$ git config --global user.email "1132241588@pfur.ru"
[rodvish@vbox ~]$ git config --global core.quotePath false
[rodvish@vbox ~]$ git config --global init.defaultBranch master
[rodvish@vbox ~]$ git config --global core.autocrlf input
[rodvish@vbox ~]$ git config --global core.safecrlf warn
```

Рис. 0.2.: Параметры репозитория

Создаем SSH ключи

```
[rodvish@vbox ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/rodvish/.ssh/id_rsa):
Created directory '/home/rodvish/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/rodvish/.ssh/id_rsa
Your public key has been saved in /home/rodvish/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:cnM5u39E6wh5y4LY71Roa7pxs1nTTQh0jpZgLpuag4E rodvish@vbox
The key's randomart image is:
+---[RSA 4096]-----+
|
| o . . |
| o o = |
| . . = . |
| . + + ... |
| E . . S *.....|
| o = +o=..oo |
| . +o..B+o=.. |
| ..o*.*+.o |
| o*+o.. |
+-----[SHA256]-----+
```

Рис. 0.3.: rsa-4096

```
[rodvish@vbox ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/rodvish/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/rodvish/.ssh/id_ed25519
Your public key has been saved in /home/rodvish/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:Fsg7qT6T6Bj0dcuWU3U6P0R2vkFe4c6cBUkfu2jgEhw rodvish@vbox
The key's randomart image is:
+--[ED25519 256]--+
|
| E .. |
| . . . . .oo|
| o .o + .oo.|
| o .B + o.o|
| . .+.So O = +.|
| . . .oo+ o O = o|
| . o.. * + = |
| o..+ . . o |
| ... .o . |
+-----[SHA256]-----+
```

Рис. 0.4.: ed25519

Создаем GPG ключ

```

Ваше полное имя: rodvish
Адрес электронной почты: 1132241588@pfur.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
"rodvish <1132241588@pfur.ru>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход?
Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)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: /home/rodvish/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/home/rodvish/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/rodvish/.gnupg/openpgp-revocs.d/0A5072F4155E360A86E61590D70387FEBB76AC70.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2025-09-15 [SC]
      0A5072F4155E360A86E61590D70387FEBB76AC70
uid    rodvish <1132241588@pfur.ru>
sub    rsa4096 2025-09-15 [E]

```

Рис. 0.5.: GPG ключ

Добавляем GPG ключ в аккаунт

```

[rodvish@vbox ~]$ 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
/home/rodvish/.gnupg/pubring.kbx
-----
sec   rsa4096/D70387FEBB76AC70 2025-09-15 [SC]
      0A5072F4155E360A86E61590D70387FEBB76AC70
uid    [ абсолютно ] rodvish <1132241588@pfur.ru>
ssb    rsa4096/E68AAA262319BCA1 2025-09-15 [E]

[rodvish@vbox ~]$ gpg --armor --export D70387FEBB76AC70 | xclip -sel clip

```

Рис. 0.6.: GPG ключ

Настройка автоматических подписей коммитов git

```

[rodvish@vbox ~]$ gpg --armor --export D70387FEBB76AC70 | xclip -sel clip
[rodvish@vbox ~]$ git config --global user.signingkey D70387FEBB76AC70
[rodvish@vbox ~]$ git config --global commit.gpgsign true
[rodvish@vbox ~]$ git config --global gpg.program $(which gpg2)

```

Рис. 0.7.: Параметры репозитория

Настройка gh

```

[rodvish@vbox ~]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 9935-E172
Press Enter to open github.com in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as rodvish

```

Рис. 0.8.: Связь репозитория с аккаунтом

Загрузка шаблона репозитория и синхронизация


```
[rodvish@vbox ~]$ mkdir -p ~/work/study/2024-2025/Операционные системы
[rodvish@vbox ~]$ cd ~/work/study/2024-2025/Операционные системы
[rodvish@vbox Операционные системы]$ gh repo create os-intro --template=yamadharma/course-directory-student-template --public
GraphQL: Could not clone: Name already exists on this account (cloneTemplateRepository)
[rodvish@vbox Операционные системы]$ gh repo create study_2022-2023_os-intro --template yamadharma/course-directory-student-template --public
Created repository rodvish/study_2022-2023_os-intro on GitHub
[rodvish@vbox Операционные системы]$ git clone --recursive https://github.com/rodvish/study_2022-2023_os-intro.git os-intro
Клонирование в «os-intro»...
remote: Repository not found.
fatal: repository 'https://github.com/rodvish/study_2022-2023_os-intro.git/' not found
[rodvish@vbox Операционные системы]$ git clone --recursive https://github.com/rodvish/study_2022-2023_os-intro.git os-intro
Клонирование в «os-intro»...
remote: Enumerating objects: 39, done.
remote: Counting objects: 100% (39/39), done.
remote: Compressing objects: 100% (36/36), done.
remote: Total 38 (delta 1), reused 27 (delta 1), pack-reused 0 (from 0)
Получение объектов: 100% (39/39), 23.45 Киб | 328.00 Киб/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «~/home/rodvish/work/study/2024-2025/Операционные системы/os-intro/template/presentation»...
remote: Enumerating objects: 161, done.
remote: Counting objects: 100% (161/161), done.
remote: Compressing objects: 100% (111/111), done.
remote: Total 161 (delta 60), reused 142 (delta 41), pack-reused 0 (from 0)
Получение объектов: 100% (161/161), 2.65 Миб | 1.34 Миб/с, готово.
Определение изменений: 100% (60/60), готово.
Клонирование в «~/home/rodvish/work/study/2024-2025/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 221, done.
remote: Counting objects: 100% (221/221), done.
remote: Compressing objects: 100% (152/152), done.
remote: Total 221 (delta 98), reused 180 (delta 57), pack-reused 0 (from 0)
Получение объектов: 100% (221/221), 765.46 Киб | 819.00 Киб/с, готово.
Определение изменений: 100% (98/98), готово.
Submodule path 'template/presentation': checked out '6ef5c4ee78e4456caff3dc7062cfad26058ca6'
Submodule path 'template/report': checked out '89a9022199046f8822709b3fa0d4714c85f68dd2'
```

Рис. 0.9.: Загрузка шаблона

Подготовка репозитория и коммит изменений

```
[rodvish@vbox os-intro]$ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 953 байта | 953.00 Киб/с, готово.
Всего 3 (изменений 1), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/rodvish/study_2022-2023_os-intro.git
8885b8f..7d10df5 master -> master
```

Рис. 0.10.: Первый коммит

Вывод

Мы приобрели практические навыки работы с сервисом github.

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 осуществляется 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: