

Отчёт по лабораторной работе №2

Управление версиями

Максим Мигачев

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.

```
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`)

<code>clone</code>	Клонирование репозитория в новый каталог
<code>init</code>	Создание пустого репозитория Git или переинициализация существующего

работа с текущими изменениями (смотрите также: `git help everyday`)

<code>add</code>	Добавление содержимого файла в индекс
<code>mv</code>	Перемещение или переименование файла, каталога или символической ссылки
<code>restore</code>	Восстановление файлов в рабочем каталоге
<code>rm</code>	Удаление файлов из рабочего каталога и индекса

просмотр истории и текущего состояния (смотрите также: `git help revisions`)

<code>bisect</code>	Выполнение двоичного поиска коммита, который вносит ошибку
<code>diff</code>	Вывод разницы между коммитами, коммитом и рабочим каталогом и т.д.
<code>grep</code>	Вывод строк, соответствующих шаблону
<code>log</code>	Вывод истории коммитов
<code>show</code>	Вывод различных типов объектов
<code>status</code>	Вывод состояния рабочего каталога

Рис. 2.1: Загрузка пакетов

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

```
~$ git config --global user.email "1132243103@rudn.university"
~$ git config --global core.quotepath false
~$ git config --global init.defaultBranch master
~$ git config --global core.autocrlf input
~$ git config --global core.safecrlf warn
~$
```

Рис. 2.2: Параметры репозитория

Создаем SSH ключи


```
The key's randomart image is:
+---[RSA 4096]-----+
|          +=...|
|      . +=+.+0|
|      . +=00+.0|
|      . +.B0+. .|
|      .S*.++0 ..|
|      ==+00..E |
|      ..+0. 0. |
|      .. . . |
|              |
+----[SHA256]-----+
```

Рис. 2.3: rsa-4096

```

+---[ED25519 256]---+
| E..*B+ . .+.0.|
| =00.0 0 ...|
| 0 . . 0 0..|
| . .... .+|
| =S 0 . =.0|
| 0 =+ 0 *|
| +.0. 0 0|
| ...=. . 0|
| . =00 ...|
+-----[SHA256]-----+

```

Рис. 2.4: ed25519

Создаем GPG ключ

```
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

```

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  rsa4096/3314511D52086862 2025-02-13 [SC]
      DD6D5E4AB971227C246E94833314511D52086862
uid          [ абсолютно ] amirahakimova <1132243103@rudn.university>
ssb  rsa4096/DCD42CD992A95560 2025-02-13 [E]

```

```

-----BEGIN PGP PUBLIC KEY BLOCK-----

```

```

mQINBGet1iYBEADZGh2yAEXj1DeRBUUkLEbtFK4w6Hr08im38KpvVbKbfhEAL7+
Ym5IZYCTs0cqtuSGcvh/9gT+LTkNVof5vbhwQbrAT1X106JeYq6H9+F9A8eaxD80
NPHcRe/tIVjkzriP/u29P7nImoQLQJ7uePC4JtCQt+Z82AkJoLCWtJv1PtLQUJ8f
3F6X4M+Z+Jb2CRUaEpXf1o+No4dI77jNXpGUJifRtEnQWCS/L91NEumAdWvJOH50
ZnA9s3XxrIh5P/eLuEiHYKYZp0PI3uq4KYGpaeByDY8J1aaZmb4qhmyNzcoHBY+o
/VVsasNuCB3BnUjkb2M6joCY8NoFi7ryiLss7Gy/X3e09nnt5EQbUvk0aU0qRE0
5mISMxbC2yZH0eJnME9EyRQ3eK5FFInwtcGjA7BdTu9s1rrtIyW7jVZXgkVJBz6N
U5uxYyZTbmQEpl0zFxV46RPuvDHG9RbTsERwiELuif7W4zQyIN5/LFpBtAyFIhL
A+pqRDXC0uH+QYzJenLAUqCLWtr++WZW7hBXLbzsF/KtDdgxS6MBjcu/aigawvD
uwGMU70xjYaI2bs+MQFFXJaRtqeUBebvUy2r2gQsZoZZStT7DU4ihTaA3U9huJZy
rIg31/7/coW6Vm8vpDUS8saJnR5n6mwYoTPdU4KPb96veU392+026/7/rwARAQAB
tCphbWlyYWwha2ltb3ZhIDwxMTMyMjQzMTAzQHNj1ZG4udW5pdmVyc2l0eT6JALEE
EwEIAQsWIQTdbv5KuXEifCRuLIMzFFedUghoYgUCZ63WJgIbAwULCQgHAgIAgYV
CgkICwIEFgIDAQIEBwIXgAAKCRAzFFedUghoYltdD/9v29Y/qGzLU9JaiiXx9/SU
8iiVo1zhW70xyvud9iCYK04V0es7C16fYpSk6hEaPEd0EUGR7nkr0TWE8S2vdlhr

```

Рис. 2.6: GPG ключ

Настройка автоматических подписей коммитов git

```

0HbfuYfPGGGQnHkfyLOQ6IVhRvfgL/NRaisc7ozDXTHzGpGVySpxb5Lipf9wLUS0
k1t56xKz50g+u0l+vI6BVEoBgP/ooBCF9WsxUHDpfQMX0/sc2Me9gA1VA16ynlih
iKxbB/Mt9kM1HN0M8ES0TWuaAzKiDdU257E0rIHT3ojzcUr8Nzn0BFpsYoeiesvE
HBWJKJF0ic6T4f0bVwZak370XArm3RQGC5FBu8Hr1kY6EimMRga87Br/V3Za5pW1
0LdTvifwC28LbXsJWxvdktmCMXWr09oErk0FSbvpQ7dVKUHc09j0gTSXdgv6nuBI
c0ia4XRgosHbA86KuNKqLNgNQFpWnQU0d70dukmvQRXyr0oRpbW62EF2auXTY0p0
yi1Crz+ik4jE2EU/E+uq5BsikPw12dT0Jz6QjLd5eoFcIhGZsngJA24cxlIzeW7Y
vMNnjJrGLiU0tbRGaDZ8Rckio5NBm1xzRoMXGkKZwTm76DdEsP7WAUqEivfLYJS
zI9DdG7/iALnZYonmyzZiBft7ATL2YiUs5SAs2TzyvQfWq9Tdcvm2XjGyafDf4Ho
kR1mgv2inpaKDZ7MH87DmSZnJb0SnRny53vHw4TMbwOPPFdzfgMHTKB+0bvaf6Is
PzlsA9yX1I6Q
=ldaj

```

```

BLOCK-----
$ git config --global user.signingkey 3314511D52086862
$ git config --global commit.gpgsign true
$ git config --global gpg.program $(which gpg2)
$

```

Рис. 2.7: Параметры репозитория

Настройка gh


```
va/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:+DiY3mVV6TujJhbpZisF/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.
remote: Enumerating objects: 36, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 36 (delta 1), reused 21 (delta 0), pack-reused 0 (from 0)
Получение объектов: 100% (36/36), 19.37 КиБ | 19.38 Миб/с, готово.
Определение изменений: 100% (1/1), готово.
Получение шаблона/presentation (https://github.com/umaydharma/academic-presentation-examples-template.git)...
```

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 - 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: