

Отчёт по лабораторной работе №2

Управление версиями

Хатамов Эзиз

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Вывод	15
4	Контрольные вопросы	16

Список иллюстраций

2.1	Загрузка пакетов	6
2.2	Параметры репозитория	6
2.3	rsa-4096	7
2.4	ed25519	8
2.5	GPG ключ	9
2.6	Параметры репозитория	11
2.7	Связь репозитория с аккаунтом	12
2.8	Загрузка шаблона	13
2.9	Первый коммит	14

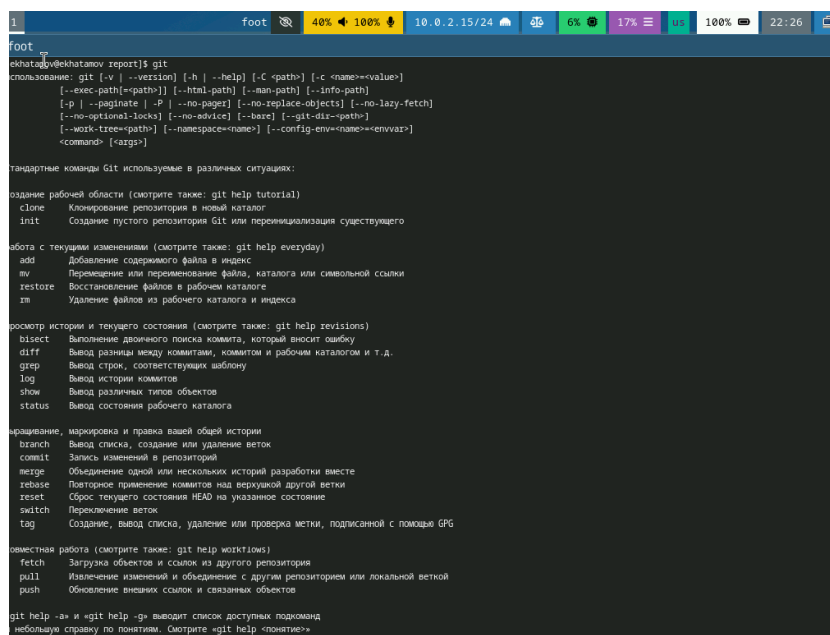
Список таблиц

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.



```
foot
ekhatamov@ekhatamov: report]$ git
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-lock>] [--no-device] [--bare] [--git-dir=<path>]
[<work-tree=<path>] [--namespace=<name>] [--config-env=<name>=<envvar>]
<command> [<args>]

стандартные команды Git используются в различных ситуациях:

создание рабочей области (смотрите также: git help tutorial)
clone Клонирование репозитория в новый каталог
init Создание пустого репозитория Git или перинициализация 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 Восстановление файлов в рабочем каталоге
rm Удаление файлов из рабочего каталога и индекса

просмотр истории и текущего состояния (смотрите также: git help revisions)
bisect Выполнение бинарного 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

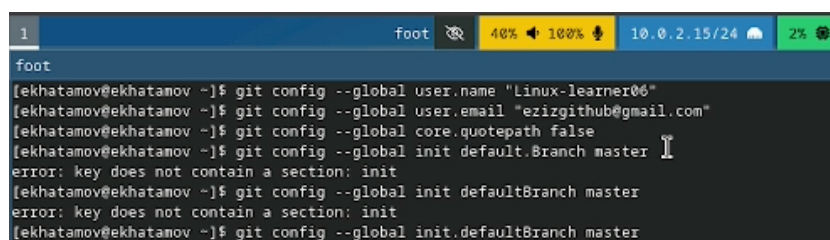
манипулирование, маркировка и правка вашей общей истории
branch Вывод списка, создание или удаление веток
commit Запись изменений в репозиторий
merge Объединение одной или нескольких историй разработки вместе
rebase Повторное применение коммитов над вершиной другой ветки
reset Сброс текущего состояния HEAD на указанное состояние
switch Переключение веток
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 «>» и «git help -f» выводит список доступных подкоманд
небольшую справку по понятию. Смотрите «git help «понятие»»
```

Рис. 2.1: Загрузка пакетов

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



```
foot
[ekhatamov@ekhatamov ~]$ git config --global user.name "Linux-learner06"
[ekhatamov@ekhatamov ~]$ git config --global user.email "ezizgithub@gmail.com"
[ekhatamov@ekhatamov ~]$ git config --global core.quotepath false
[ekhatamov@ekhatamov ~]$ git config --global init.defaultBranch master
error: key does not contain a section: init
[ekhatamov@ekhatamov ~]$ git config --global init.defaultBranch master
error: key does not contain a section: init
[ekhatamov@ekhatamov ~]$ git config --global init.defaultBranch master
```

Рис. 2.2: Параметры репозитория

Создаем SSH ключи

```

[ekhatanov@ekhatanov ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ekhatanov/.ssh/id_rsa):
/home/ekhatanov/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase for '/home/ekhatanov/.ssh/id_rsa' (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ekhatanov/.ssh/id_rsa
Your public key has been saved in /home/ekhatanov/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:9WFnIRTZYKWEAUSoE/mxxf44HtncV8d5QP2wgts6rI ekhatanov@ekhatanov
The key's randomart image is:
+----[RSA 4096]-----+
|      .  +..+8BB|
|      o o o .+o+o|
|      = =..+ .o|
|      o *.o+....^|
|      +S. B.o +|
|      .  +  + .|
|      .  . o .|
|      .  .  .|
|      Eo      |
+----[SHA256]-----+
[ekhatanov@ekhatanov ~]$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAQC7YQFSMP4fEpPA1FrqZgXAA4Ebo+scfo1v
ffDBIqHJIDcp0suQHJ/weBq179Z4EA87q8Iej0t/avdBu3P3/IaGIRQfBLH5v9ERjX78u4BM
E8SCXWK/6JiHirw+zx0gMsW0NBxYJR033D2Xqhbfd1rk0m1CLv+9zC8SC11SEVeqwV5j/U
pn0cnzIL/CNCNxm7ngya46QJSmrIy4qurQYckgCmQiLCSnTjQEKtq/wAnip8d2eMvtWe3x
Qq0wmj0yR40FG7s5tAk6TOD0DpJ0HpjwxAY39sosaryRbtKwR0FxmWbd1Yinq3FLnPhzjihs
d4dEBA+0SoroHHDaMNvS0c0FEdpqzQjGCGFdtH1Ago4GdK4fEZYno16817/09MSZxxAgv/i
Kgwz4ZCe/Ff3k7e5lnPr20jiX2wbvj0A1vh5ajdRe61cDI9qJoPTFsvUgBjEcNy7PumVf2H8
S01QP28TNeKnqn9sqH4/pz9YILfizMtsWKMNI+HMsE0RXkh5fHo19005E030g0vr64xUn49
rqa3lrIHkfjgY0+FeXZ+98biFnxuzGuF/FpAk2K5auvhACmZAqdjiEg+XZ70AXphiJ4MNpe
U/zax3itoHb7ZhuWm952rJaNpEtBcjynrVkyt3Kva6e1/EPMShfj3prakkQr165Z5+Q3s+fw
Mw-- ekhatanov@ekhatanov
[ekhatanov@ekhatanov ~]$

```

Рис. 2.3: rsa-4096

```
1 Add new SSH key - Mozilla Firefox 40% 100%
foot
[ekhatamov@ekhatamov ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ekhatamov/.ssh/id_ed25519):
/home/ekhatamov/.ssh/id_ed25519 already exists.
Overwrite (y/n)? y
Enter passphrase for "/home/ekhatamov/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ekhatamov/.ssh/id_ed25519
Your public key has been saved in /home/ekhatamov/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:BFIXpjGN4rvcOxWAKLtQ+ezqWtltFu5TCAHfG3PvDMk ekhatamov@ekhatamov
The key's randomart image is:
+--[ED25519 256]--+
| .+.o=o+. |
| .+.+oOB. |
| + +.o=.o |
| o = B.o |
| .. . oo.E.. |
| . ooo..o+ |
| oo.o=o o |
| .. o++ |
| .o. o+ |
+----[SHA256]-----+
[ekhatamov@ekhatamov ~]$ xclip -i < ~/.ssh/id_ed25519.pub
[ekhatamov@ekhatamov ~]$
```

Рис. 2.4: ed25519

Создаем GPG ключ


```
1 foot 40% 100% 10.0.
foot
(ekhatanov@ekhatanov ~)$ gpg --full-generate-key
gpg (GnuPG) 2.4.5; Copyright (C) 2024 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Выберите тип ключа:
  (1) RSA and RSA
  (2) RSA 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
Вопрошенный 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

GnuPG должен составить идентификатор пользователя для идентификации ключа.
Ваше полное имя: Eriz MAtanov
Адрес электронной почты: ezizgithub@gmail.com
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Eriz MAtanov <ezizgithub@gmail.com>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? 
```

Рис. 2.5: GPG ключ

Добавляем GPG ключ в аккаунт

```
[ekhatamov@ekhatamov ~]$ 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, 0a, 0f, 1u
[keyboard]
-----
sec   rsa4096/10677086628366E6 2025-03-01 [SC]
      3AC7883A3955DF8A804381E710677006620366E6
uid           [ абсолютно ] Eziz Hatamov <ezizgithub@gmail.com>
ssb   rsa4096/C441991388AE3373 2025-03-01 [E]
```

```
[ekhatamov@ekhatamov ~]$ gpg --armor
55DF8A804381E710677006620366E6
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGfDmAIBEAC1dVxSxojV8q59+c21Kb7p
VA9RUc+K
hitHgg yokFSYZHJMsa/D+ofK5ww2/ZTYySG
gP1qb1GI
KDKQZEe6NhPUBTG6TXSeYBEWqCcOmE5wCYC
1953jAOA
ONxQwMQtfK0z5bWsxjaHQdaJRJ1r5IF2BU7D
nrBCI7G+
rVKx+aSD9F7Z9WZVQ16gRa99wJKiyul/R1a
PyfrYpWd
eYgGKCjYlK1Aomse1fDaNKD3WYUwEnqYVuPF
W456p+Pi
fvfpY6jYhcVQYIW+GzTmaFww0153bUM0egsw
27Q9a0p8
8oAYGwsVBqDwkYN50jUMNwQAFU3fKmjGY8jC
K78EWqq7
4JxY6r+mGmn1gaNiPJl9FosLoKesZcxPTqLH
weLgg7pm
v1JVfqh4gg1HoxDGfXdQSp1zTWvchJu+Nxq
AYbU0Ntk
uMqKJ6r08vflKqatDguQbu7JKvC5lCe4BTdM
awARAQAB
```

Настройка автоматических подписей коммитов git

```

foot
VYZ7EwARAQABiQI2BBgBCAAgFiEE0se70j1V34qAQ4HnEGdwBmIDZuYF
AmfDmAIC
GwwACGkQEGdwBmIDZuZbzQ/9ExKj1NfDr0hjFstF/VGZsj1pU734MvfN
SJZQ5mCM
9990rhg/QN8de0a8jmnKu9TbEACDSOxfiWdSwyW4EwOVpAkdJjbcNDFt
CxmsNVhS
FTMd8frem8tgQ7e31uIsxGtZYQ0ZYksw32PDsRguurewRCz60n7U0arL
aGPArhmm
BVULR50rsYqRz2M7hXk8dxBYjDev3/IMd42Tf2qQPxWswxequJ8LZ277
GBYTAsGo
vL57fkXxoGrZbQDfDNhtXte3NquXHpbgQqEgVQeJfcxgwgXKgAphq85Y
4HofpDEJ
YdncThUpcjf/w+m3fJOn1RunXZu70T2w3/EURXIc6XgejUHgQo8DRmn6
h8LLhMHY
gT7nbh95dlmg2NSoo0PLNBg3AvXTiySh7/Z+z6RY0amXXnAzfzSLdpSl
tAylrY50
4F+WoxCGfFdRb1S5V7ahjGq+jLUy71w0YwL1B91JyoNJdibfbLv1Jm3l
ptn71hiW
QBjQjYBtGrtjMmOpdqxg0EHjUvhvn19ETBoyXtxAahBoABz0/k28054Q
28LxTX1P
XU2gzWVZh/yJktAscRrg78rSpWr24BecQ7KZNu0gKk2GjLSbDwPgp5u0
B3Yo//MZ
CYa0jf3I97Vi1pQjMzwveB67rVtevzvqH0dxsdWR7GJl6sAg8xDtM2P5
38VhyA8R
FG0=
=iMep
-----END PGP PUBLIC KEY BLOCK-----
[ekhatamov@ekhatamov ~]$ git config --global user.signin
gkey 3AC7BB3A3955DF8A804381E710677006620366E6
[ekhatamov@ekhatamov ~]$ git config --global commit.gpgs
ign true
[ekhatamov@ekhatamov ~]$ git config --global gpg.program
$(which gpg2)

```

Рис. 2.6: Параметры репозитория

Настройка gh

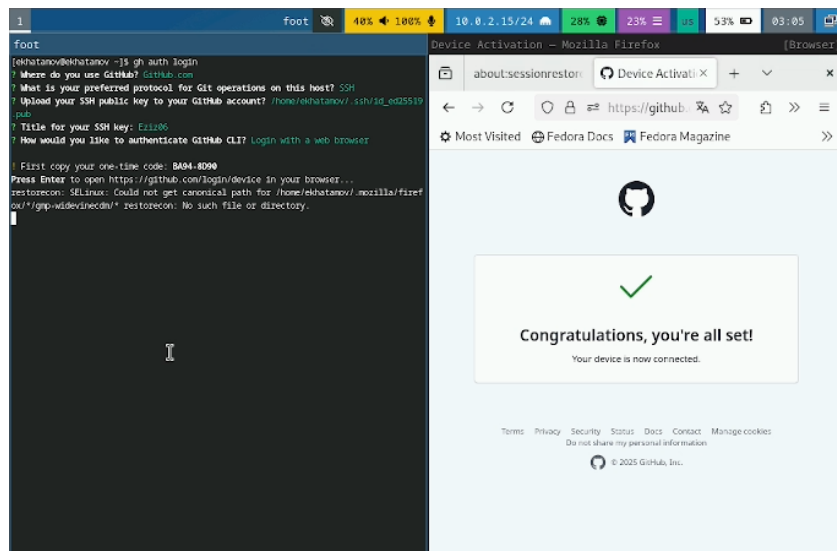


Рис. 2.7: Связь репозитория с аккаунтом

Загрузка шаблона репозитория и синхронизация

```
1 foot 40% 100%
foot
[ekhatamov@ekhatamov 2024-2025]$ cd "Operatsionnyye sistemy"
[ekhatamov@ekhatamov Operatsionnyye sistemy]$ gh repo create os-intro --template=yamadharma/course-directory-student-template --public

✓ Created repository Linux-learner06/os-intro on GitHub
https://github.com/Linux-learner06/os-intro
[ekhatamov@ekhatamov Operatsionnyye sistemy]$
[ekhatamov@ekhatamov Operatsionnyye sistemy]$ git clone --recursive git@github.com:Linux-learner06/os-intro.git os-intro
Клонирование в «os-intro»...
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 | 1.29 МиБ/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/ekhatamov/work/study/2024-2025/Operatsionnyye sistemy/os-intro/template/presentation»...
```

Рис. 2.8: Загрузка шаблона

Подготовка репозитория и коммит изменений

```

[ekhatamov@ekhatamov os-intro]$ git commit -am 'feat(main): make course structure'
[master 31738b6] feat(main): make course structure
 2 files changed, 1 insertion(+), 14 deletions(-)
 delete mode 100644 package.json
[ekhatamov@ekhatamov os-intro]$ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 949 байтов | 949.00 КиБ/с,
готово.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:Linux-learner06/os-intro.git
   74d98f9..31738b6 master -> master
[ekhatamov@ekhatamov os-intro]$

```

Рис. 2.9: 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: