

Отчет по лабораторной работе №2

Операционные Системы

Марцев Аркадий

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	13
5	Контрольные вопросы	14
	Список литературы	17

Список иллюстраций

3.1	установка git	7
3.2	установка gh	7
3.3	user.email/user.name	7
3.4	utf-8 настройка	8
3.5	настройка верификации и подписи коммитов	8
3.6	ssh rsa -b 4096	8
3.7	ssh ed25519	9
3.8	создание pgr ключа	9
3.9	github	10
3.10	отпечаток pgr ключа	10
3.11	копируем pgr ключ	10
3.12	копирование ключа	10
3.13	автоматические подписи коммитов	11
3.14	авторизация в gh	11
3.15	создание и клонирование репозитория	11
3.16	создаем структуру курса	11
3.17	отправляем изменения	12

Список таблиц

1 Цель работы

1. Изучить идеологию и применение средств контроля версий.
2. Освоить умения по работе с git.

2 Задание

1. Создать базовую конфигурацию для работы с git.
2. Создать ключ SSH.
3. Создать ключ PGP.
4. Настроить подписи git.
5. Зарегистрироваться на Github.
6. Создать локальный каталог для выполнения заданий по предмету.

3 Выполнение лабораторной работы

Для начала я запускаю в терминале супер-пользователя и устанавливаю git, чтобы использовать его функционал в терминале.

```
aamartsev@aamarcev:~$ sudo -i
[sudo] пароль для aamartsev:
[root@aamarcev ~]# dnf install git
Последняя проверка окончания срока действия метаданных: 0:01:26 назад, Вт 27 фев
 2024 22:46:45.
Пакет git-2.41.0-2.fc39.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
[root@aamarcev ~]#
```

Рис. 3.1: установка git

После того, как установился git, я скачиваю gh, который упростит авторизацию и сделает возможной авторизацию через терминал.

```
[root@aamarcev ~]# dnf install gh
Последняя проверка окончания срока действия метаданных: 0:01:50 назад, Вт 27 фев
 2024 22:46:45.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий  Размер
=====
Установка:
gh          x86_64       2.43.1-1.fc39  updates      9.1 М
Результат транзакции
=====
Установка: 1 Пакет
```

Рис. 3.2: установка gh

Задаю двумя командами имя и email владельца репозитория.

```
[root@aamarcev ~]# git config --global user.name "Arkadiy Marcev"
[root@aamarcev ~]# git config --global user.email "1132239100@pfur.ru"
[root@aamarcev ~]#
```

Рис. 3.3: user.email/user.name

Настраиваю utf-8 в выводе сообщений git.

```
[root@aamarcev ~]# git config --global core.quotepath false
```

Рис. 3.4: utf-8 настройка

Далее я настраиваю верификацию и подпись коммитов. Задаю начальную ветку master и включаю параметры autocrlf и safecrlf.

```
[root@aamarcev ~]# git config --global init.defaultBranch maste
[root@aamarcev ~]# git config --global core.autocrlf input
[root@aamarcev ~]# git config --global core.safecrlf warn
```

Рис. 3.5: настройка верификации и подписи коммитов

Создаю два secure shell ключа для верификации моего устройства. В создании этих ключей участвуют два разных алгоритма в первом случае – rsa 4096, во втором ed25519.

```
[root@aamarcev ~]# 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:c9lZcbWug6negwRy3zWUn/TwTC9/lW6pDWUSK35cFZA root@aamarcev
The key's randomart image is:
+----[RSA 4096]-----+
|          .=. = |
|          E.++ |
|          . *** |
|       . o  + BoB* |
|      o S = * B+o |
|         = oo+. +o |
|        . .o.o= . |
|       .o. ... |
|      .o ..   |
+----[SHA256]-----+
```

Рис. 3.6: ssh rsa -b 4096


```

[root@aamarcev ~]# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519
Your public key has been saved in /root/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:xky6p1xwhsxMEMjR+dIicsURAOhtv0gImdsrRGAvSwg root@aamarcev
The key's randomart image is:
+--[ED25519 256]--+
|  =.++..O+O      |
| E.B +.O..       |
|==.. =.+         |
|*+.. =+*         |
|*O    B S        |
|O+.  *           |
|. . . O          |
|. . . +          |
|. . O            |
+-----[SHA256]-----+

```

Рис. 3.7: ssh ed25519

Теперь я дополнительно создаю pgr ключ, выбирая требуемые варианты и вводя свои данные.

```

-----[SHA256]-----
[root@aamarcev ~]# gpg --full-generate-key
gpg: invalid option '--full-generate-key'
[root@aamarcev ~]# gpg --full-generate-key
gpg (GnuPG) 2.4.3; Copyright (C) 2023 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.

gpg: создан каталог '/root/.gnupg'
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
Вопрошенный размер ключа - 4096 бит
Выберите срок действия ключа.
  0 = не ограничен
  <n> = срок действия ключа - n дней
  <nw> = срок действия ключа - n недель
  <nM> = срок действия ключа - n месяцев
  <ny> = 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: Martsev Arkadiy
Адрес электронной почты: 1132239190@pfur.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Martsev Arkadiy <1132239190@pfur.ru>"
Сменить (N) Имя, (C) Примечание, (E) Адрес, (O) Принять/(Q) Выход? o

```

Рис. 3.8: создание pgr ключа

Регистрирую аккаунт на гитхабе и ввожу туда свои основные данные.

Настраиваем автоматические подписи коммитов. Используя введенный email, указываем git применять его при подписи коммитов.

```
[root@aamarcev .ssh]# git config --global user.signingkey 250D9BCA5B0FAFCE
[root@aamarcev .ssh]# git config --global commit.gpgsign true
[root@aamarcev .ssh]# git config --global gpg.program $(which gpg2)
[root@aamarcev .ssh]#
```

Рис. 3.13: автоматические подписи коммитов

Авторизуемся в gh через терминал.

```
What account do you want to log into? github.com
What is your preferred protocol for git operations on this host? 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? Login with a web browser

First copy your one-time code: 6257-DE86
Press Enter to open github.com in your browser...
Running Firefox as root in a regular user's session is not supported. (XAUTHORITY is /run/user/1000/.mutter-kwaylandauth.72FQ32 which is owned by aamarcev)
```

Рис. 3.14: авторизация в gh

Создаем папки в которые будем клонировать шаблон нашего репозитория, переходим в них. Копируем репозиторий вставляя свое имя пользователя в ссылку и клонируем его в заранее созданные папки.

```
[root@aamarcev .ssh]# mkdir -p ~/work/study/2022-2023/"Операционные системы"
[root@aamarcev .ssh]# cd ~/work/study/2022-2023/"Операционные системы"
[root@aamarcev Операционные системы]# gh repo create study_2022-2023_os-intro --template=yamadharma/course-directory-student-template --public
Created repository aamarcev/study_2022-2023_os-intro on GitHub
https://github.com/aamarcev/study_2022-2023_os-intro
[root@aamarcev Операционные системы]# git clone --recursive git@github.com:aamarcev/study_2022-2023_os-intro.git os-intro
bash: owner: нет такого файла или каталога
[root@aamarcev Операционные системы]# git clone --recursive git@github.com:aamarcev/study_2022-2023_os-intro.git os-intro
Клонирование в «os-intro»...
The authenticity of host 'github.com (148.82.121.4)' can't be established.
6025519 key fingerprint is SHA256:10Y13wvv6TuJ2hbpZisf/zLDA82zMSvHdkr4UvCQgu.
This key is not known by any other names.
```

Рис. 3.15: создание и клонирование репозитория

Создаем структуру курса добавляя в репозиторий папки для лабораторных работ и индивидуального проекта.

```
[root@aamarcev Операционные системы]# cd ~/work/study/2022-2023/"Операционные системы"/os-intro
[root@aamarcev os-intro]# rm package.json
rm: удалить обычный файл 'package.json'? yes
[root@aamarcev os-intro]# echo os-intro > COURSE
[root@aamarcev os-intro]# make
Usage:
make <target>

Targets:
list             List of courses
prepare          Generate directories structure
submodule        Update submodules
```

Рис. 3.16: создаем структуру курса

Когда мы убедились, что структура создана, отправляем изменения на github.

```
[root@aamarcev os-intro]# git add .  
[root@aamarcev os-intro]# git commit -am 'feat(main: make course structure)'  
Текущая ветка: master  
Эта ветка соответствует «origin/master».  
  
нечего коммитить, нет изменений в рабочем каталоге  
[root@aamarcev os-intro]# git push  
Everything up-to-date
```

Рис. 3.17: отправляем изменения

4 Выводы

мы научились создавать собственные репозитории по шаблону, выполнять первоначальную настройку git и пользоваться рядом его функций.

5 Контрольные вопросы

1. Системы контроля версий (VCS) - программное обеспечение для облегчения работы с изменяющейся информацией. Они позволяют хранить несколько версий изменяющейся информации, одного и того же документа, может предоставить доступ к более ранним версиям документа. Используется для работы нескольких человек над проектом, позволяет посмотреть, кто и когда внес какое-либо изменение и т. д. VCS применяются для: Хранения полной истории изменений, сохранения причин всех изменений, поиска причин изменений и совершивших изменение, совместной работы над проектами.
2. Хранилище – репозиторий, хранилище версий, в нем хранятся все документы, включая историю их изменения и прочей служебной информацией. commit – отслеживание изменений, сохраняет разницу в изменениях. История – хранит все изменения в проекте и позволяет при необходимости вернуться/обратиться к нужным данным. Рабочая копия – копия проекта, основанная на версии из хранилища, чаще всего последней версии.
3. Централизованные VCS (например: CVS, TFS, AccuRev) – одно основное хранилище всего проекта. Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет, затем добавляет изменения обратно в хранилище. Децентрализованные VCS (например: Git, Bazaar) – у каждого пользователя свой вариант репозитория (возможно несколько вариантов), есть возможность добавлять и забирать изменения из любого

репозитория. В отличие от классических, в распределенных (децентрализованных) системах контроля версий центральный репозиторий не является обязательным.

4. Сначала создается и подключается удаленный репозиторий, затем по мере изменения проекта эти изменения отправляются на сервер.
5. Участник проекта перед началом работы получает нужную ему версию проекта в хранилище, с помощью определенных команд, после внесения изменений пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются. К ним можно вернуться в любой момент.
6. Хранение информации о всех изменениях в вашем коде, обеспечение удобства командной работы над кодом.
7. Создание основного дерева репозитория: `git init`

Получение обновлений (изменений) текущего дерева из центрального репозитория: `git pull`

Отправка всех произведённых изменений локального дерева в центральный репозиторий: `git push`

Просмотр списка изменённых файлов в текущей директории: `git status`

Просмотр текущих изменений: `git diff`

Сохранение текущих изменений: добавить все изменённые и/или созданные файлы и/или каталоги: `git add` .

добавить конкретные изменённые и/или созданные файлы и/или каталоги: `git add имена_файлов`

удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): `git rm имена_файлов`

Сохранение добавленных изменений:

сохранить все добавленные изменения и все изменённые файлы: `git commit -am 'Описание коммита'`

сохранить добавленные изменения с внесением комментария через встроенный редактор: `git commit`

создание новой ветки, базирующейся на текущей: `git checkout -b имя_ветки`

переключение на некоторую ветку: `git checkout имя_ветки` (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой)

отправка изменений конкретной ветки в центральный репозиторий: `git push origin имя_ветки`

слияние ветки с текущим деревом: `git merge --no-ff имя_ветки`

Удаление ветки:

удаление локальной уже слитой с основным деревом ветки: `git branch -d имя_ветки`

принудительное удаление локальной ветки: `git branch -D имя_ветки`

удаление ветки с центрального репозитория: `git push origin :имя_ветки`

8. `git push -all` отправляем из локального репозитория все сохраненные изменения в центральный репозиторий, предварительно создав локальный репозиторий и сделав предварительную конфигурацию.
9. Ветвление - один из параллельных участков в одном хранилище, исходящих из одной версии, обычно есть главная ветка. Между ветками, т. е. их концами возможно их слияние. Используются для разработки новых функций.
10. Во время работы над проектом могут создаваться файлы, которые не следуют добавлять в репозиторий. Например, временные файлы. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл `.gitignore` с помощью сервисов.

Список литературы