

Лабораторная работа № 2

Первоначальная настройка git.

Ермаков Алексей

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	15
5	Выводы	19
	Список литературы	20

Список иллюстраций

4.1	Установка программного обеспечения git и gh	15
4.2	Базовая настройка git	17
4.3	Базовая настройка git	18
4.4	Название рисунка	18

Список таблиц

1 Цель работы

Изучить идеологию и применение средств контроля версий.

Освоить умения по работе с `git`.

2 Задание

Задание

Создать базовую конфигурацию для работы с git.

Создать ключ SSH.

Создать ключ PGP.

Настроить подписи git.

Зарегистрироваться на Github.

Создать локальный каталог для выполнения заданий по предмету.

3 Теоретическое введение

Системы контроля версий. Общие понятия

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.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными

участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.

Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

Примеры использования git

Система контроля версий Git представляет собой набор программ командной строки. Доступ Благодаря тому, что Git является распределённой системой контроля версий, резервную ко

Основные команды git

Перечислим наиболее часто используемые команды git.

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 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

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
```

Стандартные процедуры работы при наличии центрального репозитория

Работа пользователя со своей веткой начинается с проверки и получения изменений из центрального репозитория:

```
git checkout master
```

```
git pull
```

```
git checkout -b имя_ветки
```

Затем можно вносить изменения в локальном дереве и/или ветке.

После завершения внесения какого-то изменения в файлы и/или каталоги проекта необходимо проверить статус:

```
git status
```

При необходимости удаляем лишние файлы, которые не хотим отправлять в центральный репозиторий:

Затем полезно посмотреть текст изменений на предмет соответствия правилам ведения чистоты

```
git diff
```

Если какие-либо файлы не должны попасть в коммит, то помечаем только те файлы, изменения которых не хотим

```
git add ...
```

```
git rm ...
```

Если нужно сохранить все изменения в текущем каталоге, то используем:

```
git add .
```

Затем сохраняем изменения, поясняя, что было сделано:

```
git commit -am "Some commit message"
```

Отправляем изменения в центральный репозиторий:

```
git push origin имя_ветки
```

или

```
git push
```

Работа с локальным репозиторием

Создадим локальный репозиторий.

Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория

```
git config --global user.name "Имя Фамилия"  
git config --global user.email "work@mail"
```

Настроим utf-8 в выводе сообщений git:

```
git config --global quotepath false
```

Для инициализации локального репозитория, расположенного, например, в каталоге ~/tutorial

```
cd  
mkdir tutorial  
cd tutorial  
git init
```

После это в каталоге tutorial появится каталог .git, в котором будет храниться история

Создадим тестовый текстовый файл hello.txt и добавим его в локальный репозиторий:

```
echo 'hello world' > hello.txt  
git add hello.txt  
git commit -am 'Новый файл'
```

Воспользуемся командой status для просмотра изменений в рабочем каталоге, сделанных с

```
git status
```

Во время работы над проектом так или иначе могут создаваться файлы, которые не требуют

```
curl -L -s https://www.gitignore.io/api/list
```

Затем скачать шаблон, например, для С и С++

```
curl -L -s https://www.gitignore.io/api/c >> .gitignore
```

```
curl -L -s https://www.gitignore.io/api/c++ >> .gitignore
```

4 Выполнение лабораторной работы

Установка программного обеспечения(рис. 4.1).

Установка git

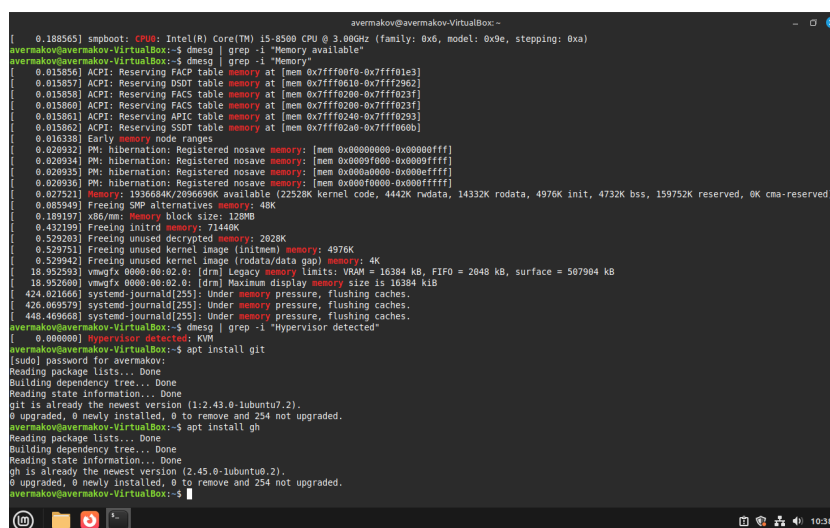
Установим git:

```
dnf install git
```

Установка gh

Fedora:

```
dnf install gh
```



```
0.188565] smpboot: CPU0: Intel(R) Core(TM) i5-8500 CPU @ 3.80GHz (family: 0x6, model: 0x9e, stepping: 0xa)
avermakov@avermakov-VirtualBox:~$ dmesg | grep -i "Memory available"
0.015856] ACPI: Reserving FACP table memory at [mem 0x7fff00f0-0x7fff01e3]
0.015857] ACPI: Reserving DSDT table memory at [mem 0x7fff0610-0x7fff2962]
0.015858] ACPI: Reserving FACS table memory at [mem 0x7fff0200-0x7fff023f]
0.015868] ACPI: Reserving FACS table memory at [mem 0x7fff0200-0x7fff023f]
0.015861] ACPI: Reserving APIC table memory at [mem 0x7fff0240-0x7fff0293]
0.015862] ACPI: Reserving SSDT table memory at [mem 0x7fff02a0-0x7fff060b]
0.016338] Early memory node ranges
0.020932] PM: hibernation: Registered nosave memory: [mem 0x00000000-0x00000fff]
0.020934] PM: hibernation: Registered nosave memory: [mem 0x0000f000-0x0000ffff]
0.020935] PM: hibernation: Registered nosave memory: [mem 0x00000000-0x0000ffff]
0.020936] PM: hibernation: Registered nosave memory: [mem 0x000f0000-0x000fffff]
0.027521] Memory: 1936684K/2096696K available (22528K kernel code, 4442K rdata, 14332K rodata, 4976K init, 4732K bss, 159752K reserved, 0K cma-reserved)
0.082949] Freeing SMP alternatives memory: 48K
0.189197] x86/mm: Memory block size: 128MB
0.432199] Freeing initrd memory: 71448K
0.529203] Freeing unused decrypted memory: 2028K
0.529751] Freeing unused kernel image (initmem) memory: 4976K
0.529942] Freeing unused kernel image (rodata/data gap) memory: 4K
18.352593] vmwgfx 0000:00:02.0: [drm] legacy memory limits: VRAM = 16384 KB, FIFO = 2048 KB, surface = 507904 KB
18.525680] vmwgfx 0000:00:02.0: [drm] Maximum display memory size is 16384 KiB
424.021666] systemd-journald[255]: Under memory pressure, flushing caches.
426.069579] systemd-journald[255]: Under memory pressure, flushing caches.
448.489668] systemd-journald[255]: Under memory pressure, flushing caches.
avermakov@avermakov-VirtualBox:~$ dmesg | grep -i "Hypervisor detected"
0.000000] Hypervisor detected: KVM
avermakov@avermakov-VirtualBox:~$ sudo dnf install git
[sudo] password for avermakov:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.43.0-1ubuntu0.2).
0 upgraded, 0 newly installed, 0 to remove and 254 not upgraded.
avermakov@avermakov-VirtualBox:~$ sudo dnf install gh
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
gh is already the newest version (2.45.0-1ubuntu0.2).
0 upgraded, 0 newly installed, 0 to remove and 254 not upgraded.
avermakov@avermakov-VirtualBox:~$
```

Рис. 4.1: Установка программного обеспечения git и gh

Базовая настройка git (рис. 4.2).

Зададим имя и email владельца репозитория:

```
git config --global user.name "Name Surname"
```

```
git config --global user.email "work@mail"
```

Настроим utf-8 в выводе сообщений git:

```
git config --global core.quotepath false
```

Настройте верификацию и подписание коммитов git (см. Верификация коммитов git с помощью gpg).

Зададим имя начальной ветки (будем называть её master):

```
git config --global init.defaultBranch master
```

Параметр autocrlf:

```
git config --global core.autocrlf input
```

Параметр safecrlf:

```
git config --global core.safecrlf warn
```



```
0.000000] Hypervisor detected: KVM
avermakov@avermakov-VirtualBox:~$ apt install git
[sudo] password for avermakov:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.43.0-lubuntu7.2).
0 upgraded, 0 newly installed, 0 to remove and 254 not upgraded.
avermakov@avermakov-VirtualBox:~$ apt install gh
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
gh is already the newest version (2.45.0-lubuntu7.2).
0 upgraded, 0 newly installed, 0 to remove and 254 not upgraded.
avermakov@avermakov-VirtualBox:~$ git config --global user.name "vn322"
avermakov@avermakov-VirtualBox:~$ git config --global user.email "bigbr@mail.ru"
avermakov@avermakov-VirtualBox:~$ git config --global core.quotepath false
avermakov@avermakov-VirtualBox:~$ git config --global init.defaultBranch master
avermakov@avermakov-VirtualBox:~$ git config --global core.autocrlf input
avermakov@avermakov-VirtualBox:~$ git config --global core.safecrlf warn
avermakov@avermakov-VirtualBox:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/avermakov/.ssh/id_rsa): rs1
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in rs1
Your public key has been saved in rs1.pub
The key fingerprint is:
SHA256:ENWQwZ0wZpNwtqWxJfKpkwix1zEU67CEP100YH0UM avermakov@avermakov-VirtualBox
The key's randomart image is:
+--[RSA 4096]--+
|
|.X.B =
|.+.0 0
|.0.+.
|..S.
|..
|.Bto 0.
|.00+0.0.
|..
|..
+-----[SHA256]-----
avermakov@avermakov-VirtualBox:~$ gpg --full-generate-key
```

Рис. 4.2: Базовая настройка git

Создайте ключи ssh

по алгоритму rsa с ключём размером 4096 бит:

```
ssh-keygen -t rsa -b 4096
```

по алгоритму ed25519:

```
ssh-keygen -t ed25519
```

Создайте ключи pgp

Генерируем ключ

```
gpg --full-generate-key
```

```
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: /home/avermakov/.gnupg/trustdb.gpg: trustdb created
gpg: directory '/home/avermakov/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/avermakov/.gnupg/openpgp-revocs.d/61852794AD6AD09510F6FCF2838C58599FCBE.rev'
public and secret key created and signed.

pub  rsa4096 2025-04-27 [SC]
     61852794AD6AD09510F6FCF2838C58599FCBE
uid      vn322 <bigbr@mail.ru>
sub  rsa4096 2025-04-27 [E]

avermakov@avermakov-VirtualBox:~$ gpg --list-secret-keys --keyid-format LONG
gpg: checking the trustdb
gpg: marginals needed: 3 completes needed: 1 trust model: gpg
gpg: depth: 0 valid: 1 signed: 0 trust: 0:, 0q, 0n, 0f, 1u
/home/avermakov/.gnupg/pubring.kbx
-----
sec  rsa4096/F2838C58599FCBE 2025-04-27 [SC]
     61852794AD6AD09510F6FCF2838C58599FCBE
uid      [ultimate] vn322 <bigbr@mail.ru>
ssb  rsa4096/F8CF4C663E1BE860 2025-04-27 [E]

avermakov@avermakov-VirtualBox:~$ gpg --armor --export 61852794AD6AD09510F6FCF2838C58599FCBE | xclip -sel clip
avermakov@avermakov-VirtualBox:~$ git config --global user.signingkey 61852794AD6AD09510F6FCF2838C58599FCBE
avermakov@avermakov-VirtualBox:~$ git config --global commit.gpgsign true
avermakov@avermakov-VirtualBox:~$ git config --global gpg.program $(which gpg2)
avermakov@avermakov-VirtualBox:~$ gh auth login
? What account do you want to log into? github.com
? What is your preferred protocol for git operations on this host? ssh
? Generate a new SSH key to add to your GitHub account? No
? How would you like to authenticate GitHub CLI? oauth with 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'.
? Paste your authentication token: *****
- gh config set -h github.com git protocol ssh
  Configured git protocol
  Logged in as vn322
avermakov@avermakov-VirtualBox:~$
```

Рис. 4.3: Базовая настройка git

Настройка каталога курса(рис. 4.4).

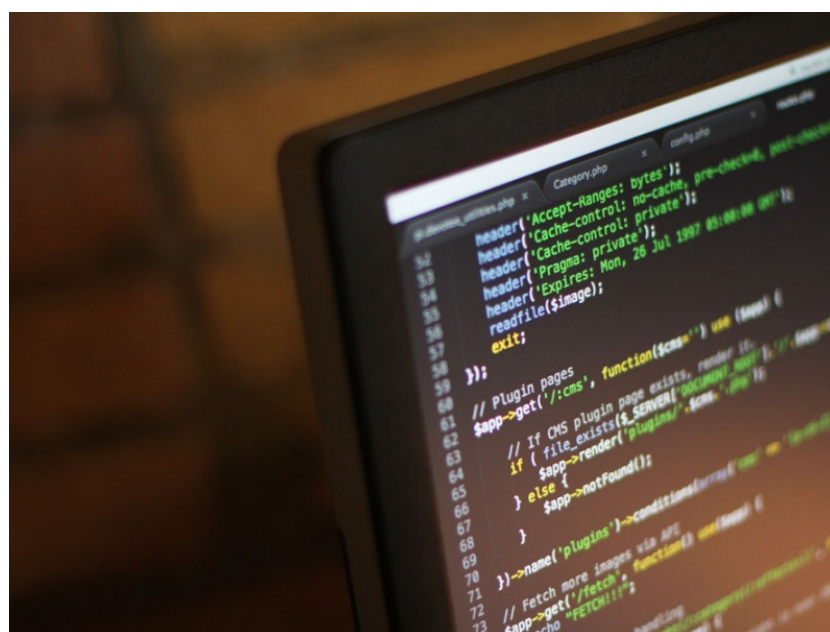


Рис. 4.4: Название рисунка

5 Выводы

В данной работе познакомились с системой контроля версий git. Создали клон учебного репозитория создали и отредактировали файлы с последующей отправкой в репозиторий и проверкой их наличия. Визуальный интерфейс выглядит привлекательнее, но может и в командной строке есть свои плюсы.

Список литературы

1. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016. URL: <https://www.gnu.org/software/bash/manual/>.
2. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.
6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с. {#refs} :::