

# **Отчёт по лабораторной работе №2**

**Управление версиями**

Магамадов Асхаб Ахиатович

# Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Вывод	11
4	Контрольные вопросы	12
	Список литературы	16

## Список иллюстраций

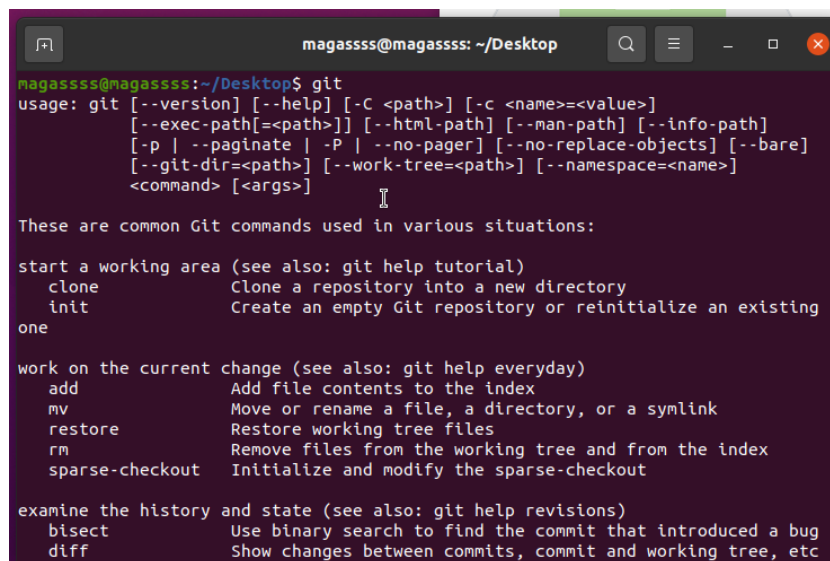
2.1	Загрузка пакетов . . . . .	5
2.2	Параметры репозитория . . . . .	6
2.3	rsa-4096 . . . . .	6
2.4	ed25519 . . . . .	7
2.5	GPG ключ . . . . .	7
2.6	GPG ключ . . . . .	8
2.7	Параметры репозитория . . . . .	8
2.8	Связь репозитория с аккаунтом . . . . .	9
2.9	Загрузка шаблона . . . . .	9
2.10	Первый коммит . . . . .	10

# 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.



```
magassss@magassss: ~/Desktop
magassss@magassss:~/Desktop$ 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>]
        <command> [<args>]

These are common Git commands used in various situations:


start a working area (see also: git help tutorial)
  clone                Clone a repository into a new directory
  init                 Create an empty Git repository or reinitialize an existing
  one

work on the current change (see also: git help everyday)
  add                  Add file contents to the index
  mv                   Move or rename a file, a directory, or a symlink
  restore              Restore working tree files
  rm                   Remove files from the working tree and from the index
  sparse-checkout       Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
  bisect               Use binary search to find the commit that introduced a bug
  diff                 Show changes between commits, commit and working tree, etc
```

Рис. 2.1: Загрузка пакетов

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

```
magassss@magassss: ~/Desktop
reset          Reset current HEAD to the specified state
switch         Switch branches
tag            Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
fetch          Download objects and refs from another repository
pull           Fetch from and integrate with another repository or a local
branch         branch
push           Update remote refs along with associated objects

'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.
magassss@magassss:~/Desktop$
magassss@magassss:~/Desktop$
magassss@magassss:~/Desktop$ git config --global user.name "ashabmagassss"
magassss@magassss:~/Desktop$ git config --global user.email "1132220821@pfur.ru"
magassss@magassss:~/Desktop$ git config --global core.quotePath false
magassss@magassss:~/Desktop$ git config --global init.defaultBranch master
magassss@magassss:~/Desktop$ git config --global core.autocrlf input
magassss@magassss:~/Desktop$ git config --global core.safecrlf warn
magassss@magassss:~/Desktop$
magassss@magassss:~/Desktop$
```

Рис. 2.2: Параметры репозитория

Создаем SSH ключи

```
magassss@magassss:~/Desktop$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/magassss/.ssh/id_rsa):
/home/magassss/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/magassss/.ssh/id_rsa
Your public key has been saved in /home/magassss/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Mk2RyEpl7pfsFeW4qUIdDgAqMl8X2uG0LUUZ7AWckv4 magassss@magassss
The key's randomart image is:
+---[RSA 4096]---+
| . o*.o0==. |
| . ..Bo==.+ |
|= .+*.oo.. |
|oo ..+ =.o.+ |
| . * S.+ |
| . O oE |
| . + |
| |
+---[SHA256]-----+
magassss@magassss:~/Desktop$
```

Рис. 2.3: rsa-4096

```
magassss@magassss: ~/Desktop
+----[SHA256]-----+
magassss@magassss:~/Desktop$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/magassss/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/magassss/.ssh/id_ed25519
Your public key has been saved in /home/magassss/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:nwsSTrhro9vZ+gGzHLgQCSeezUAKjVRrzF5HhJc/JRM magassss@magassss
The key's randomart image is:
+--[ED25519 256]--+
|BBo. oo.E. |
|+=+o ...o o . |
|oo B .... + |
|.o.... o |
|. ..= o S . |
|. o O . . . |
|. + + . o |
|.o+ o . . |
|o+=+o . |
+----[SHA256]-----+
magassss@magassss:~/Desktop$
```

Рис. 2.4: ed25519

Создаем GPG ключ

```
magassss@magassss: ~/Desktop
"ashabmagassss <113220821@pfur.ru>"
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
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.
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/magassss/.gnupg/trustdb.gpg: trustdb created
gpg: key 48C204F2762ADFB5 marked as ultimately trusted
gpg: directory '/home/magassss/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/magassss/.gnupg/openpgp-revocs.d/AE
EE08ACBDAF0BD347E5F4D948C204F2762ADFB5.rev'
public and secret key created and signed.

pub   rsa4096 2023-02-15 [SC]
       AEEE08ACBDAF0BD347E5F4D948C204F2762ADFB5
uid           ashabmagassss <113220821@pfur.ru>
sub   rsa4096 2023-02-15 [E]
magassss@magassss:~/Desktop$
```

Рис. 2.5: GPG ключ

Добавляем GPG ключ в аккаунт

## SSH keys

[New SSH key](#)


There are no SSH keys associated with your account.

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

## GPG keys

[New GPG key](#)

This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.



1

Email address: 1132220821@pfur.ru

Key ID: 48C204F2762ADFB5

Subkeys: 8E45D99777C8DAFD

Added on Feb 15, 2023

Delete

[Learn how to generate a GPG key and add it to your account.](#)

Рис. 2.6: GPG ключ

## Настройка автоматических подписей коммитов git

```
magassss@magassss:~/Desktop$
magassss@magassss:~/Desktop$
magassss@magassss:~/Desktop$ gpg --list-secret-keys --keyid-format LONG
/home/magassss/.gnupg/pubring.kbx
-----
sec   rsa4096/48C204F2762ADFB5 2023-02-15 [SC]
      AEEE08ACBDAF0BD347E5F4D948C204F2762ADFB5
uid           [ultimate] ashabmagassss <1132220821@pfur.ru>
ssb   rsa4096/8E45D99777C8DAFD 2023-02-15 [E]

magassss@magassss:~/Desktop$ git config --global commit.gpgSign true
magassss@magassss:~/Desktop$ git config --global user.signingKey 48C204F2762ADFB5
magassss@magassss:~/Desktop$ git config --global gpg.program $(which gpg2)
magassss@magassss:~/Desktop$
```

Рис. 2.7: Параметры репозитория

## Настройка gh



```
magassss@magassss:~/Desktop$ git config --global gpg.program $(which gpg2)
magassss@magassss:~/Desktop$
magassss@magassss:~/Desktop$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? SSH
? Upload your SSH public key to your GitHub account? /home/magassss/.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: 6AE6-FBA6
Press Enter to open github.com in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
✓ Uploaded the SSH key to your GitHub account: /home/magassss/.ssh/id_rsa.pub
✓ Logged in as ashabmagassss
magassss@magassss:~/Desktop$
```

Рис. 2.8: Связь репозитория с аккаунтом

## Загрузка шаблона репозитория и синхронизация

```
magassss@magassss: ~/work/study/2022-2023/Операционн...
Cloning into '/home/magassss/work/study/2022-2023/Операционные системы/os-intro/
template/presentation'...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 82 (delta 28), reused 77 (delta 23), pack-reused 0
Cloning into '/home/magassss/work/study/2022-2023/Операционные системы/os-intro/
template/report'...
remote: Enumerating objects: 101, done.
remote: Counting objects: 100% (101/101), done.
remote: Compressing objects: 100% (70/70), done.
remote: Total 101 (delta 40), reused 88 (delta 27), pack-reused 0
Receiving objects: 100% (101/101), 327.25 KiB | 3.34 MiB/s, done.
Resolving deltas: 100% (40/40), done.
Submodule path 'template/presentation': checked out 'b1be380ee91f5809264cb755d3
16174540b753e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a83917b82e3aef11a3
3b1e3b2'
magassss@magassss:~/work/study/2022-2023/Операционные системы$
magassss@magassss:~/work/study/2022-2023/Операционные системы$ cd os-intro/
magassss@magassss:~/work/study/2022-2023/Операционные системы/os-intro$ ls
CHANGELOG.md  COURSE  Makefile  README.en.md  README.md
config        LICENSE  package.json  README.git-flow.md  template
magassss@magassss:~/work/study/2022-2023/Операционные системы/os-intro$
```

Рис. 2.9: Загрузка шаблона

## Подготовка репозитория и коммит изменений

```
magassss@magassss: ~/work/study/2022-2023/Операционн...
remote: Enumerating objects: 101, done.
remote: Counting objects: 100% (101/101), done.
remote: Compressing objects: 100% (70/70), done.
remote: Total 101 (delta 40), reused 88 (delta 27), pack-reused 0
Receiving objects: 100% (101/101), 327.25 KiB | 3.34 MiB/s, done.
Resolving deltas: 100% (40/40), done.
Submodule path 'template/presentation': checked out 'b1be3800ee91f5809264cb755d3
16174540b753e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a83917b82e3aef11a3
3b1e3b2'
magassss@magassss:~/work/study/2022-2023/Операционные системы$
magassss@magassss:~/work/study/2022-2023/Операционные системы$ cd os-intro/
magassss@magassss:~/work/study/2022-2023/Операционные системы/os-intro$ ls
CHANGELOG.md  COURSE  Makefile  README.en.md  README.md
config        LICENSE  package.json  README.git-flow.md  template
magassss@magassss:~/work/study/2022-2023/Операционные системы/os-intro$ rm packa
ge.json
magassss@magassss:~/work/study/2022-2023/Операционные системы/os-intro$ make COU
RSE=os-intro
magassss@magassss:~/work/study/2022-2023/Операционные системы/os-intro$ ls
CHANGELOG.md  labs  prepare  README.en.md  template
config        LICENSE  presentation  README.git-flow.md
COURSE       Makefile  project-personal  README.md
magassss@magassss:~/work/study/2022-2023/Операционные системы/os-intro$
```

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:

# Список литературы

1. Лекция Системы контроля версий
2. GitHub для начинающих