

# **Отчёт по лабораторной работе №2**

## **Управление версиями**

Довлетов Довлет

# **Содержание**

<b>1 Цель работы</b>	<b>5</b>
<b>2 Выполнение лабораторной работы</b>	<b>6</b>
<b>3 Вывод</b>	<b>17</b>
<b>4 Контрольные вопросы</b>	<b>18</b>

# 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.

```
ddovletov@ddovletov:~$ git
usage: 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>]

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

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
  grep      Print lines matching a pattern
  log       Show commit logs
  show      Show various types of objects
  status    Show the working tree status
```

Рисунок 2.1: Загрузка пакетов

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

```
ddovletov@ddovletov:~$  
ddovletov@ddovletov:~$ git config --global user.name "ddovletov"  
ddovletov@ddovletov:~$ git config --global user.email "10325469@rudn.ru"  
ddovletov@ddovletov:~$ git config --global core.quotePath false  
ddovletov@ddovletov:~$ git config --global init.defaultBranch master  
ddovletov@ddovletov:~$ git config --global core.autocrlf input  
ddovletov@ddovletov:~$ git config --global core.safecrlf warn  
ddovletov@ddovletov:~$
```

Рисунок 2.2: Параметры репозитория

Создаем SSH ключи

```
ddovletov@ddovletov:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ddovletov/.ssh/id_rsa):
Created directory '/home/ddovletov/.ssh'.
Enter passphrase for "/home/ddovletov/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ddovletov/.ssh/id_rsa
Your public key has been saved in /home/ddovletov/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:UkG/g8YJidzWp+On0upzGwW2rcDk+Ee+18G2Z+mfeuY ddovletov@ddovletov
The key's randomart image is:
+---[RSA 4096]---+
|          .0      |
|         . o o    |
|        + B o o   |
|       = o B * .  |
|      . + + S *  |
|     . + * + + . |
|    . * o + +   |
|   ..o+ o +  o.  |
|  .o+o+. o*E    |
+---[SHA256]---+
ddovletov@ddovletov:~$
```

Рисунок 2.3: rsa-4096

```
ddovletov@ddovletov:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ddovletov/.ssh/id_ed25519):
Enter passphrase for "/home/ddovletov/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ddovletov/.ssh/id_ed25519
Your public key has been saved in /home/ddovletov/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:m0wTFZZEfL0EzFB8hi9vyCqCfDP3r2B9dRiz9hNw/M8 ddovletov@ddovletov
The key's randomart image is:
+--[ED25519 256]--+
|      +B0o+   |
|      oo * *   |
|      . . *o+   |
|      . . ==.   |
|      S . +=.o  |
|      o = 000+o  |
|      . . * ....E|
|      o =.0...   .|
|      . = 000.   |
+---[SHA256]----+
ddovletov@ddovletov:~$
```

Рисунок 2.4: ed25519

Создаем GPG ключ

```
ddovletov@ddovletov:~$ gpg --full-generate-key
gpg (GnuPG) 2.4.9; Copyright (C) 2025 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: directory '/home/ddovletov/.gnupg' created
Please select what kind of key you want:
  (1) RSA and RSA
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (9) ECC (sign and encrypt) *default*
  (10) ECC (sign only)
  (14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long. [I]
What keysize do you want? (3072) 4096
Requested keysize is 4096 bits
Please specify how long the key should be valid.
  0 = key does not expire
  <n>  = key expires in n days
  <n>w = key expires in n weeks
  <n>m = key expires in n months
  <n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.
```

Рисунок 2.5: GPG ключ

Добавляем GPG ключ в аккаунт

```
ddovletov@ddovletov:~$ gpg --list-secret-keys --keyid-format LONG
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid: 1  signed: 0  trust: 0-, 0q, 0n, 0m, 0f, 1u
[keyboxd]
-----
sec rsa4096/8372EF4AD2694693 2026-02-17 [SC]
      3E2273AFBEF56C6687DFC17F8372EF4AD2694693
uid          [ultimate] ddovletov <1032254169@rudn.ru>
ssb rsa4096/4C488693B3389666 2026-02-17 [E]

ddovletov@ddovletov:~$ git config --global user.signingkey 8372EF4AD2694693
ddovletov@ddovletov:~$ git config --global commit.gpgsign true
ddovletov@ddovletov:~$ git config --global gpg.program $(which gpg2)
ddovletov@ddovletov:~$ gpg --armor --export 8372EF4AD2694693 | xclip -sel clip
ddovletov@ddovletov:~$
```

Рисунок 2.6: GPG ключ

Настройка автоматических подписей коммитов git

The screenshot shows the GitHub account settings page for the user 'ddovletov'. The top navigation bar includes links for 'Public profile', 'Your personal account', 'Go to your personal profile', and a green 'New SSH key' button. The left sidebar lists account management options: 'Public profile', 'Account', 'Appearance', 'Accessibility', 'Notifications', 'Access', 'Billing and licensing' (with a dropdown arrow), 'Emails', 'Password and authentication', 'Sessions', 'SSH and GPG keys' (which is selected and highlighted in grey), 'Organizations', and 'Enterprises'. The main content area is divided into two sections: 'SSH keys' and 'GPG keys'. The 'SSH keys' section displays a message: 'There are no SSH keys associated with your account.' It includes a link to a guide on connecting GitHub using SSH keys and troubleshooting common SSH problems. The 'GPG keys' section displays a message: 'This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.' It shows one key entry: '1' (Key ID: 8372EF4A02694693, Email address: 1032254169@ruuhr.ru, Subkeys: 4C488693B3389666, Added on Feb 17, 2026). A 'Delete' button is located next to the key entry. Below the GPG keys section, there is a link to learn how to generate a GPG key and add it to the account.

Рисунок 2.7: Параметры репозитория

Настройка gh

```
ddovletov@ddovletov:~$ gh auth login
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /home/ddovletov/.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: DD58-A0CC
Press Enter to open https://github.com/login/device 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/ddovletov/.ssh/id_rsa.pub
✓ Logged in as ddovletov
ddovletov@ddovletov:~$
```

Рисунок 2.8: Связь репозитория с аккаунтом

Загрузка шаблона репозитория и синхронизация

```
Receiving objects: 100% (287/287), 785.00 KiB | 4.00 MiB/s, done.  
Resolving deltas: 100% (125/125), done.  
Submodule path 'template/presentation': checked out '60a0091ebedbb01a8d98f2c5fae0f98dc63ad031'  
Submodule path 'template/report': checked out '708b8c51acd94d4b78e7634198f221e5decb6355'  
ddovletov@ddovletov:~/work/study/2025-2026/Операционные системы$  
ddovletov@ddovletov:~/work/study/2025-2026/Операционные системы$  
ddovletov@ddovletov:~/work/study/2025-2026/Операционные системы$  
ddovletov@ddovletov:~/work/study/2025-2026/Операционные системы$  
ddovletov@ddovletov:~/work/study/2025-2026/Операционные системы$ cd ~/work/study/2025-2026/"Операцио  
нныe системы"/2026-1--study--os-intro  
ddovletov@ddovletov:~/work/study/2025-2026/Операционные системы/2026-1--study--os-intro$ echo os-int  
ro > COURSE  
ddovletov@ddovletov:~/work/study/2025-2026/Операционные системы/2026-1--study--os-intro$ make prepar  
e  
Synchronizing submodule url for 'template/report'  
Synchronizing submodule url for 'template/presentation'  
Synchronizing submodule url for 'template/presentation'  
Synchronizing submodule url for 'template/report'  
ddovletov@ddovletov:~/work/study/2025-2026/Операционные системы/2026-1--study--os-intro$
```

Рисунок 2.9: Загрузка шаблона

## Подготовка репозитория и коммит изменений

```
create mode 100644 project-personal/stage06/report/_macro1.tcl
create mode 100644 project-personal/stage06/report/_extensions/yamadharma/minted-quarto/_extension.yml
create mode 100644 project-personal/stage06/report/_extensions/yamadharma/minted-quarto/minted-quarto.lua
create mode 100644 project-personal/stage06/report/_quarto.yml
create mode 100644 project-personal/stage06/report/_resources/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 project-personal/stage06/report/_resources/tex/preamble.tex
create mode 100644 project-personal/stage06/report/bib/cite.bib
create mode 100644 project-personal/stage06/report/image/solvay.jpg
create mode 100644 project-personal/stage06/report/os-intro-project-personal-stage06-report.qmd
ddovletov@ddovletov:~/work/study/2025-2026/Операционные системы/2026-1--study--os-intro$ git push
Enumerating objects: 109, done.
Counting objects: 100% (109/109), done. ▄
Delta compression using up to 8 threads
Compressing objects: 100% (90/90), done.
Writing objects: 100% (106/106), 705.57 KiB | 4.93 MiB/s, done.
Total 106 (delta 41), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (41/41), completed with 1 local object.
To github.com:ddovletov/2026-1--study--os-intro.git
    715b193..0a1d8bc master -> master
ddovletov@ddovletov:~/work/study/2025-2026/Операционные системы/2026-1--study--os-intro$
```

Рисунок 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 слиивается 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 - запушить 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: