

Отчёт по лабораторной работе №2

Управление версиями

Чернятин Артём Андреевич

Содержание

| | |
|---|-----------|
| 1 Цель работы | 5 |
| 2 Выполнение лабораторной работы | 6 |
| 3 Вывод | 17 |
| 4 Контрольные вопросы | 18 |

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.

```
aachernyatin@aachernyatin:~$ 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.

```
aachernyatin@aachernyatin:~$ git config -global user.name "aa-chertyanin"  
aachernyatin@aachernyatin:~$ git config -global user.email "1132246223@pfur.ru"  
aachernyatin@aachernyatin:~$ git config -global core.quotepath false  
aachernyatin@aachernyatin:~$ git config -global init.defaultBranch master  
aachernyatin@aachernyatin:~$ git config -global core.autocrlf input  
aachernyatin@aachernyatin:~$ git config -global core.safecrlf warn  
aachernyatin@aachernyatin:~$ [
```

Рисунок 2.2: Параметры репозитория

Создаем SSH ключи

```
aachernyatin@aachernyatin:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/aachernyatin/.ssh/id_rsa):
Created directory '/home/aachernyatin/.ssh'.
Enter passphrase for "/home/aachernyatin/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/aachernyatin/.ssh/id_rsa
Your public key has been saved in /home/aachernyatin/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:p+IyEOh+wRVTRXLv/ryrdjADAOz/nuXT5DAqARs92/U aachernyatin@aachernyatin
The key's randomart image is:
+---[RSA 4096]---+
| ..000+
| + .0 .
| . . + . .
| . . = 0 ...
| . .... = S +..
| ..0 . + +.* E
| . . . + .oX
| . .0. 0 0+00+
| . 0. 00.0+=0
+---[SHA256]---+
aachernyatin@aachernyatin:~$
```

Рисунок 2.3: rsa-4096

```
aachernyatin@aachernyatin:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/aachernyatin/.ssh/id_ed25519):
Enter passphrase for "/home/aachernyatin/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/aachernyatin/.ssh/id_ed25519
Your public key has been saved in /home/aachernyatin/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:QkRL3r3IShSROCYF1n0LZ+VgjPTnd0em0qzpk4I6rPo aachernyatin@aachernyatin
The key's randomart image is:
+--[ED25519 256]--+
| 0+.=B=0.. |
| .. +=*=+= |
| 0 .==0.= . . |
| 0 ..= 0 0 |
| o S o o |
| . o 0 0 |
| . ... ..+ |
| o . . 0+ |
| .oE..o 0+. |
+---[SHA256]---+
aachernyatin@aachernyatin:~$
```

Рисунок 2.4: ed25519

Создаем GPG ключ

```
aachernyatin@aachernyatin:~$ gpg --full-generate-key
gpg (GnuPG) 2.4.7; 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.

gpg: directory '/home/aachernyatin/.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.
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.

Real name: aa-chertyanin
Email address: 1132246223@pfur.ru
Comment:
You selected this USER-ID:
  "aa-chertyanin <1132246223@pfur.ru>"
```

Рисунок 2.5: GPG ключ

Добавляем GPG ключ в аккаунт

```
aachernyatin@aachernyatin:~$ gpg --list-secret-keys --keyid-format LONG  
[keyboxd]  
-----  
sec  rsa4096/4B4D81845A52CB0A 2025-12-08 [SC]  
      1F14C4299F3F8A8FDA308C8E4B4D81845A52CB0A  
uid          [ultimate] aa-chertyanin <1132246223@pfur.ru>  
ssb  rsa4096/A57908019E911CC5 2025-12-08 [E]  
  
aachernyatin@aachernyatin:~$ gpg --armor --export 4B4D81845A52CB0A  
-----BEGIN PGP PUBLIC KEY BLOCK-----  
  
mQINBGk22lwBEADYkSHy43o8Lx5+cGmXpq0nsnIM7VE6IuDJPoLWyCsfxLtgrJL  
gEDMMNNQuN0EvbcuIcNZ+nL983UrrNpJD3VrovFSdt9if8rE2QedGM/Y3j4Su0K4  
tbR3ZT6cXnDzFv1So4jH4MvMRM1ZYgYECKADieAuIlPKNRy1/j6mutx4L/Ynmra  
GUDE012440UWk/almAfMxmewVa1cM4owktAt3u1PZQTWYl6jEhekxwMjneDX6Xnb  
oTORqteWVMTG3y9vo4RFsFuvDWCoYuUzLugkvocZYU5XfKJcv8TVf+2KfdXbJC/6  
soRPAffH9hV5nK02AJ/sS/99jbIoDhjdGNBtSsKVMJvM139BAiHPrkY2qXrN6EPE  
X/JKaQmWK4Vt6RX5VFipj54EKxPhdeEWsSXzwv6ysVPPLcStP1vSRBR7R1YHLMkl  
RpuoQdDrjcZFDj2XpWR1L052P1UE3Xa0TAhqD6frVi j1m0T1MozADa0QgbgrHJEK  
LHhWIvydunJ930vw18KSixzvBNcqFCGdJK83TKBgxWA1Q7Kx5hDcCq5RM5UdNMl0  
2bPvNIbL8jJx14NCu900IyicsCImTmbEc0R2SDrzvnmrPJT34Meqt1Vs6b5n3zs  
21jMYfRP+o2HzqK+QwXPYS0ngH3e1z+WsDjrJSjm19I3t2+YLwve9cWVwARAQAB  
tCJhYS1jaGVydHlhbmluIDwxMTMyMjQ2MjIzQHBmdXIucnU+iQJRBMBcG A7F iEE  
HxTEKZ8/io/aMIyOS02BhFpSywoFAmk22lwCGwMFcwkIBwICigIGFQoJCA sCBBYC
```

Рисунок 2.6: GPG ключ

Настройка автоматических подписей коммитов git



Рисунок 2.7: Параметры репозитория

Настройка gh

```
aachernyatin@aachernyatin:~$  
aachernyatin@aachernyatin:~$ 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/aachernyatin/.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: 5CCA-1D9B  
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/aachernyatin/.ssh/id_rsa.pub  
✓ Logged in as aa-chertyanin  
aachernyatin@aachernyatin:~$
```

Рисунок 2.8: Связь репозитория с аккаунтом

Загрузка шаблона репозитория и синхронизация

```
Resolving deltas: 100% (1/1), done.
Submodule 'template/presentation' (https://github.com/yamadharma/academic-presentation-markdown-template.git) registered for path 'template/presentation'
Submodule 'template/report' (https://github.com/yamadharma/academic-laboratory-report-template.git) registered for path 'template/report'
Cloning into '/home/aachernyatin/work/study/2025-2026/Операционные системы/os-intro/template/presentation'...
remote: Enumerating objects: 219, done.
remote: Counting objects: 100% (219/219), done.
remote: Compressing objects: 100% (151/151), done.
remote: Total 219 (delta 86), reused 189 (delta 56), pack-reused 0 (from 0)
Receiving objects: 100% (219/219), 2.66 MiB | 9.99 MiB/s, done.
Resolving deltas: 100% (86/86), done.
Cloning into '/home/aachernyatin/work/study/2025-2026/Операционные системы/os-intro/template/report'...
remote: Enumerating objects: 251, done.
remote: Counting objects: 100% (251/251), done.
remote: Compressing objects: 100% (172/172), done.
remote: Total 251 (delta 111), reused 284 (delta 64), pack-reused 0 (from 0)
Receiving objects: 100% (251/251), 775.12 KiB | 3.67 MiB/s, done.
Resolving deltas: 100% (111/111), done.
Submodule path 'template/presentation': checked out '1c93acf9e731bf186384c85de4aff70037314240'
Submodule path 'template/report': checked out '8ee157c58b3362947b1c71492a65d4dc6882d5ad'
aachernyatin@aachernyatin:~/work/study/2025-2026/Операционные системы$ 
aachernyatin@aachernyatin:~/work/study/2025-2026/Операционные системы$ cd ~/work/study/2025-2026/"Операционные системы"/os-intro
aachernyatin@aachernyatin:~/work/study/2025-2026/Операционные системы$ make COURSE=os-intro prepare
aachernyatin@aachernyatin:~/work/study/2025-2026/Операционные системы$ ls
COURSE LICENSE package.json presentation README.en.md README.md
labs Makefile prepare project-personal README.git-flow.md template
aachernyatin@aachernyatin:~/work/study/2025-2026/Операционные системы$ 
```

Рисунок 2.9: Загрузка шаблона

Подготовка репозитория и коммит изменений

```
create mode 100644 project-personal/stage6/presentation/_assets/auto/beamer.el
create mode 100644 project-personal/stage6/presentation/_assets/beamer.tex
create mode 100644 project-personal/stage6/presentation/_quarto.yml
create mode 100644 project-personal/stage6/presentation/_resources/image/logo_rudn.png
create mode 100644 project-personal/stage6/presentation/image/kulyabov.jpg
create mode 100644 project-personal/stage6/presentation/os-intro--project-personal--stage6--presentation.qmd
create mode 100644 project-personal/stage6/report/.gitignore
create mode 100644 project-personal/stage6/report/.marksman.toml
create mode 100644 project-personal/stage6/report/.projectile
create mode 100644 project-personal/stage6/report/Makefile
create mode 100644 project-personal/stage6/report/_assets/preamble.tex
create mode 100644 project-personal/stage6/report/_quarto.yml
create mode 100644 project-personal/stage6/report/_resources/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 project-personal/stage6/report/bib/cite.bib
create mode 100644 project-personal/stage6/report/image/solvay.jpg
create mode 100644 project-personal/stage6/report/os-intro--project-personal--stage6--report.qmd
aachernyatin@aachernyatin:~/work/study/2025-2026/Операционные системы/os-intro$ git push
Enumerating objects: 106, done.
Counting objects: 100% (106/106), done.
Delta compression using up to 4 threads
Compressing objects: 100% (87/87), done.
Writing objects: 100% (103/103), 704.46 KiB | 4.96 MiB/s, done.
Total 103 (delta 42), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (42/42), completed with 1 local object.
To github.com:aa-chertyanin/os-intro.git
  2c4fe28..bbb0995 master -> master
aachernyatin@aachernyatin:~/work/study/2025-2026/Операционные системы/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: