

# **Отчёта по лабораторной работе №2**

**НКНбд-02-21**

Акондзо Жордани Лади Гаэл

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Ход работы</b>	<b>9</b>
<b>5</b>	<b>Выводы</b>	<b>16</b>
<b>6</b>	<b>Контрольные вопросы</b>	<b>17</b>

## Список иллюстраций

4.1	Регистрация в github . . . . .	9
4.2	Установка git-flow в Fedora Linux . . . . .	10
4.3	Установка gh в Fedora Linux . . . . .	10
4.4	Базовая настройка GIT . . . . .	11
4.5	Генерирование ключа ssh . . . . .	11
4.6	Генерирование ключа pgr . . . . .	12
4.7	Добавление pgr ключа в Github . . . . .	12
4.8	Подписание коммитов git . . . . .	13
4.9	Настройка gh . . . . .	13
4.10	Настройка gh . . . . .	14
4.11	Создание репозитория курса . . . . .	14
4.12	Настройка каталога курса 1 . . . . .	15
4.13	Настройка каталога курса 2 . . . . .	15
4.14	Настройка каталога курса 2 . . . . .	15

# Список таблиц

3.1	Описание основных команд git . . . . .	7
-----	--	---

# 1 Цель работы

Цель данной работы — Изучить идеологию и применение средств контроля версий. Освоить умения по работе с git.

## 2 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

По определению система контроля версий (VCS) это программное обеспечение для облегчения работы с изменяющейся информацией. Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Git является распределённой системой контроля версий. Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` с различными опциями. В табл. 3.1 приведено краткое описание наиболее часто используемых команд `git`.

Таблица 3.1: Описание основных команд `git`

Команда	Описание действия
<code>git init</code>	создание основного дерева репозитория
<code>git pull</code>	получение обновлений (изменений) текущего дерева из центрального репозитория
<code>git status</code>	просмотр списка изменённых файлов в текущей директории
<code>git push</code>	отправка всех произведённых изменений локального дерева в центральный репозиторий
<code>git diff</code>	просмотр текущих изменения
<code>git add .</code>	добавить все изменённые и/или созданные файлы и/или каталоги
<code>git add</code> имена_файлов	добавить конкретные изменённые и/или созданные файлы и/или каталоги

Команда	Описание действия
git rm имена_файлов	удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории)
git commit -am 'Коммит'	сохранить все добавленные изменения и все изменённые файлы
git commit	сохранить добавленные изменения с внесением комментария через встроенный редактор
git checkout -b имя_ветки	создание новой ветки, базирующейся на текущей
git checkout имя_ветки	переключение на некоторую ветку
git push origin имя_ветки	отправка изменений конкретной ветки в центральный репозиторий
git branch -d имя_ветки	удаление локальной уже слитой с основным деревом ветки
git branch -D имя_ветки	принудительное удаление локальной ветки
git push origin :имя_ветки	удаление ветки с центрального репозитория



## 4 Ход работы

### 1. Настройка github

Зарегистрировался на Github. (сайт <https://github.com>) (рис.4.1)

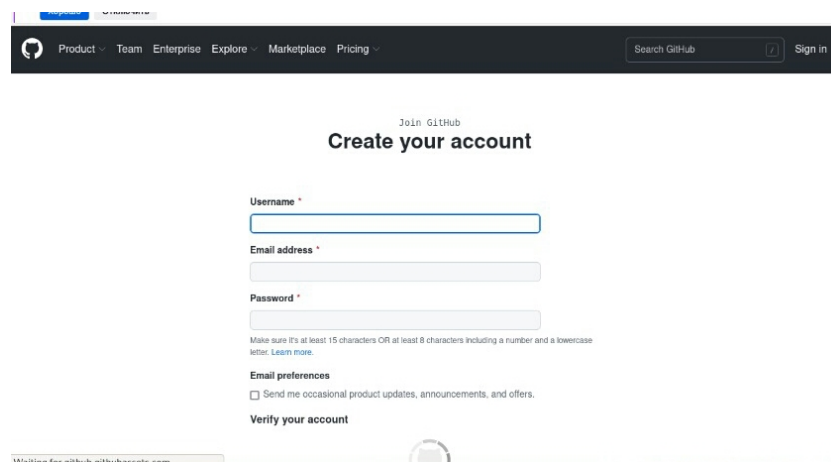


Рис. 4.1: Регистрация в github

### 2. Установка git-flow в Fedora Linux

Установил его вручную, введя следующие команды: (рис. 4.2)

```
zlkondzo@jordan1 ~$ cd /tmp
zlkondzo@jordan1 tmp$ wget --no-check-certificate -q https://raw.githubusercontent.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh
zlkondzo@jordan1 tmp$ chmod +x gitflow-installer.sh
zlkondzo@jordan1 tmp$ sudo ./gitflow-installer.sh install stable
[sudo] Mot de passe de zlkondzo :
## git-flow no-make installer ##
installing git-flow to /usr/local/bin
Cloning repo from Github to gitflow
Clonage dans 'gitflow'...
remote: Enumerating objects: 4270, done.
remote: Total 4270 (delta 0), reused 0 (delta 0), pack-reused 4270
Réception d'objets: 100% (4270/4270), 1.74 Mio | 602.00 Kio/s, fait.
Résolution des deltas: 100% (2533/2533), fait.
Déjà à jour.
La branche 'master' est paramétrée pour suivre la branche distante 'master' depuis 'origin'.
Basculement sur la nouvelle branche 'master'
install: création du répertoire '/usr/local/share/doc'
install: création du répertoire '/usr/local/share/doc/gitflow'
install: création du répertoire '/usr/local/share/doc/gitflow/hooks'
'gitflow/git-flow' -> '/usr/local/bin/git-flow'
'gitflow/git-flow-init' -> '/usr/local/bin/git-flow-init'
'gitflow/git-flow-feature' -> '/usr/local/bin/git-flow-feature'
'gitflow/git-flow-bugfix' -> '/usr/local/bin/git-flow-bugfix'
'gitflow/git-flow-hotfix' -> '/usr/local/bin/git-flow-hotfix'
'gitflow/git-flow-release' -> '/usr/local/bin/git-flow-release'
'gitflow/git-flow-support' -> '/usr/local/bin/git-flow-support'
'gitflow/git-flow-version' -> '/usr/local/bin/git-flow-version'
'gitflow/gitflow-common' -> '/usr/local/bin/gitflow-common'
'gitflow/gitflow-shFlags' -> '/usr/local/bin/gitflow-shFlags'
'gitflow/git-flow-config' -> '/usr/local/bin/git-flow-config'
'gitflow/hooks/filter-flow-hotfix-finish-tag-message' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-hotfix-finish-tag-message'
'gitflow/hooks/filter-flow-hotfix-start-version' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-hotfix-start-version'
'gitflow/hooks/filter-flow-release-branch-tag-message' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-release-branch-tag-message'
'gitflow/hooks/filter-flow-release-finish-tag-message' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-release-finish-tag-message'
'gitflow/hooks/filter-flow-release-start-version' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-release-start-version'
'gitflow/hooks/post-flow-bugfix-delete' -> '/usr/local/share/doc/gitflow/hooks/post-flow-bugfix-delete'
'gitflow/hooks/post-flow-bugfix-finish' -> '/usr/local/share/doc/gitflow/hooks/post-flow-bugfix-finish'
'gitflow/hooks/post-flow-bugfix-publish' -> '/usr/local/share/doc/gitflow/hooks/post-flow-bugfix-publish'
'gitflow/hooks/post-flow-bugfix-pull' -> '/usr/local/share/doc/gitflow/hooks/post-flow-bugfix-pull'
'gitflow/hooks/post-flow-bugfix-start' -> '/usr/local/share/doc/gitflow/hooks/post-flow-bugfix-start'
'gitflow/hooks/post-flow-bugfix-track' -> '/usr/local/share/doc/gitflow/hooks/post-flow-bugfix-track'
'gitflow/hooks/post-flow-feature-delete' -> '/usr/local/share/doc/gitflow/hooks/post-flow-feature-delete'
'gitflow/hooks/post-flow-feature-finish' -> '/usr/local/share/doc/gitflow/hooks/post-flow-feature-finish'
'gitflow/hooks/post-flow-feature-publish' -> '/usr/local/share/doc/gitflow/hooks/post-flow-feature-publish'
'gitflow/hooks/post-flow-feature-pull' -> '/usr/local/share/doc/gitflow/hooks/post-flow-feature-pull'
'gitflow/hooks/post-flow-feature-start' -> '/usr/local/share/doc/gitflow/hooks/post-flow-feature-start'
'gitflow/hooks/post-flow-feature-track' -> '/usr/local/share/doc/gitflow/hooks/post-flow-feature-track'
'gitflow/hooks/post-flow-hotfix-delete' -> '/usr/local/share/doc/gitflow/hooks/post-flow-hotfix-delete'
```

Рис. 4.2: Установка git-flow в Fedora Linux

### 3. Установка gh в Fedora Linux

Использовал следующую команду: (рис. 4.3)

```
zlkondzo@jordan1 tmp$ sudo dnf install gh
Dernière vérification de l'expiration des métadonnées effectuée il y a 0:00:24 le mer. 27 avril 2022 04:45:24.
Le paquet gh-2.7.0-1.fc35.x86_64 est déjà installé.
Dépendances résolues.
Rien à faire.
Terminé !
zlkondzo@jordan1 tmp$ git config --global user.name Jordaniakondzo
```

Рис. 4.3: Установка gh в Fedora Linux

### 4. Базовая настройка GIT

- Задал имя и email владельца репозитория.
- Настроил utf-8 в выводе сообщений git.
- Задал имя начальной ветки, настроил параметры autocrlf и safecrlf.gpg (рис. 4.4)

```
rien à faire.
terminé !
zlakondzo@jordani tmp]$ git config --global user.name Jordaniakondzo
zlakondzo@jordani tmp]$ git config --global user.email jordaniakondzo3438@gmail.com
zlakondzo@jordani tmp]$ git config --global core.quotepath false
zlakondzo@jordani tmp]$ git config --global init.defaultBranch master
zlakondzo@jordani tmp]$ git config --global core.autocrlf input
zlakondzo@jordani tmp]$ git config --global core.safecrlf warn
zlakondzo@jordani tmp]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/zlakondzo/.ssh/id_rsa):
Created directory '/home/zlakondzo/.ssh'.
```

Рис. 4.4: Базовая настройка GIT

## 5. Создание ключа gpg

- Сгенерировал ключ ssh по алгоритму rsa и ed25519 (рис. 4.5)

```
zlakondzo@jordani tmp]$ git config --global core.safecrlf warn
zlakondzo@jordani tmp]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/zlakondzo/.ssh/id_rsa):
Created directory '/home/zlakondzo/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/zlakondzo/.ssh/id_rsa
Your public key has been saved in /home/zlakondzo/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:PIRIZTruFv6m2b1xSn1q/754qL+9+H2CmE+WBXh6RM zlakondzo@jordani
The key's randomart image is:
----[RSA 4096]-----
      .+..+ .
      o + +
      = + +
      S . + + +
      o . + + +
      oo+.8
      o+.o+o+
      g++8*
----[SHA256]-----
zlakondzo@jordani tmp]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/zlakondzo/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/zlakondzo/.ssh/id_ed25519
Your public key has been saved in /home/zlakondzo/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:d8EqS3hn96H3TVRc3o9J854PKYK/5r3qx131W6iuzEE zlakondzo@jordani
The key's randomart image is:
---[ED25519 256]---
      .
      o o +o

```

Рис. 4.5: Генерирование ключа ssh

- Сгенерировал ключ pgp. (рис. 4.6)

```

-----[SHA256]-----
zlkondzo@jordani tmp$ gpg --full-generate-key
gpg (GnuPG) 2.3.2; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: répertoire = /home/zlkondzo/.gnupg = créé
gpg: le trousseau local = /home/zlkondzo/.gnupg/pubring.kbx = a été créé
selectionnez le type de clef désiré :
  (1) RSA and RSA
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (9) ECC (sign and encrypt) «default»
  (10) ECC (signature seule)
  (14) Existing key from card
Quel est votre choix ? 1
les clefs RSA peuvent faire une taille comprise entre 1024 et 4096 bits.
Quelle taille de clef désirez-vous ? (3072) 4096
La taille demandée est 4096 bits
Veuillez indiquer le temps pendant lequel cette clef devrait être valable.
  0 = la clef n'expire pas
  <n> = la clef expire dans n jours
  <nm> = la clef expire dans n semaines
  <nm> = la clef expire dans n mois
  <ny> = la clef expire dans n ans
Pendant combien de temps la clef est-elle valable ? (0) 0
La clef n'expire pas du tout
Est-ce correct ? (o/N) o

GnuPG doit construire une identité pour identifier la clef.

Nom réel : Jordaniakondzo
Adresse électronique : jordaniakondzo3438@gmail.com
Commentaire :
Vous avez sélectionné cette identité :
  « Jordaniakondzo <jordaniakondzo3438@gmail.com> »

Changer le (N)om, le (C)ommentaire, l'(A)ddresse électronique
ou (O)ui/(Q)uitter ? o
De nombreux octets aléatoires doivent être générés. Vous devriez faire
autre chose (taper au clavier, déplacer la souris, utiliser les disques)
pendant la génération de nombres premiers ; cela donne au générateur de

```

Рис. 4.6: Генерирование ключа ргр

## 6. Добавление ргр ключа в Github

- Выводил список ключей и копировал отпечаток приватного ключаю.
- Скопировал мой сгенерированный ргр ключ в буфер обмена. (рис. 4.7)

```

Activités Terminal 27 avril 05:20
zlkondzo@jordani/tmp

gpg: clef 591C803CBF489599 marquée de confiance ultime.
gpg: répertoire = /home/zlkondzo/.gnupg/openpgp-revocs.d = créé
gpg: revocation certificate stored as '/home/zlkondzo/.gnupg/openpgp-revocs.d/1EDBC1159C203C7FD0959A6A591C803CBF489599.rev'
les clefs publique et secrète ont été créées et signées.

pub  rsa4096 2022-04-27 [SC]
     1EDBC1159C203C7FD0959A6A591C803CBF489599
uid  Jordaniakondzo <jordaniakondzo3438@gmail.com>
sub  rsa4096 2022-04-27 [E]

[zlkondzo@jordani tmp]$ gpg --list-secret-keys --keyid-format LONG
gpg: invalid option '--list-secret-keys'
[zlkondzo@jordani tmp]$ gpg --list-secret-keys --keyid-format LONG
bash: gpg: command not found...
Install package 'general-purpose-preprocessor' to provide command 'gpg'? [N/y] N

[zlkondzo@jordani tmp]$ gpg --list-secret-keys --keyid-format LONG
gpg: vérification de la base de confiance
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: profondeur : 0 valables : 1 signées : 0
     confiance : 0 t., 0 n.d., 0 j., 0 m., 0 t., 1 u.
/home/zlkondzo/.gnupg/pubring.kbx
-----
sec  rsa4096/591C803CBF489599 2022-04-27 [SC]
     1EDBC1159C203C7FD0959A6A591C803CBF489599
uid  Jordaniakondzo <jordaniakondzo3438@gmail.com>
sub  rsa4096/3343850E776F9A3D 2022-04-27 [E]

[zlkondzo@jordani tmp]$ gpg --armor --export ^C
[zlkondzo@jordani tmp]$ gpg --armor --export 591C803CBF489599 | xclip -sel clip
bash: xclip: command not found...
Install package 'xclip' to provide command 'xclip'? [N/y] y

* Waiting in queue...
* Loading list of packages...
The following packages have to be installed:
xclip-0.13-15.git11cha61.fc35.x86_64 Command line clipboard grabber
Proceed with changes? [N/y] n

The transaction did not proceed.
Failed to install packages: user declined simulation

[zlkondzo@jordani tmp]$ gpg --armor --export 591C803CBF489599 | xclip -sel clip
bash: xclip: command not found...
Install package 'xclip' to provide command 'xclip'? [N/y] y

```

Рис. 4.7: Добавление ргр ключа в Github

## 7. Настройка автоматических подписей коммитов git

- Настроил автоматические подписи коммитов git: (рис. 4.8)

```
zlakeondzo@jordani tmp$ gpg --armor --export 591C803CBF489599 | xclip -sel c1tp
zlakeondzo@jordani tmp$ git config --global user.signingkey -----BEGIN PGP PUBLIC KEY BLOCK-----
^C
zlakeondzo@jordani tmp$ gpg config --global user.signingkey 591C803CBF489599
gpg: Remarque : « --global » n'est pas considéré comme une option
gpg: WARNING: no command supplied. Trying to guess what you mean ...
Utilisation : gpg [options] [filename]
zlakeondzo@jordani tmp$ git config --global user.signingkey 591C803CBF489599
zlakeondzo@jordani tmp$ git config --global commit.gpgsign true
zlakeondzo@jordani tmp$ git config --global gpg.program $ZLAKONDZO
zlakeondzo@jordani tmp$ gh auth login
? What account do you want to log into? (Use arrows to move, type to filter)
```

Рис. 4.8: Подписание коммитов git

## 8. Настройка gh

- Затем настроил gh (рис. 4.9, 4.10)

```
zlakeondzo@jordani tmp$ 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/zlakeondzo/.ssh/id_rsa.pub
? How would you like to authenticate GitHub CLI? Login with a web browser

First copy your one-time code: EF0F-D979
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/zlakeondzo/.ssh/id_rsa.pub
/ Logged in as Jordaniakondzo
zlakeondzo@jordani tmp$ gh auth login
? What account do you want to log into? GitHub.com
? You're already logged into github.com. Do you want to re-authenticate? Yes
? What is your preferred protocol for Git operations? SSH
? Upload your SSH public key to your GitHub account? /home/zlakeondzo/.ssh/id_rsa.pub
? How would you like to authenticate GitHub CLI? Paste 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', 'admin:public_key'.
? Paste your authentication token:
? Sorry, your reply was invalid: Value is required
? Paste your authentication token:
zlakeondzo@jordani tmp$
```

Рис. 4.9: Настройка gh

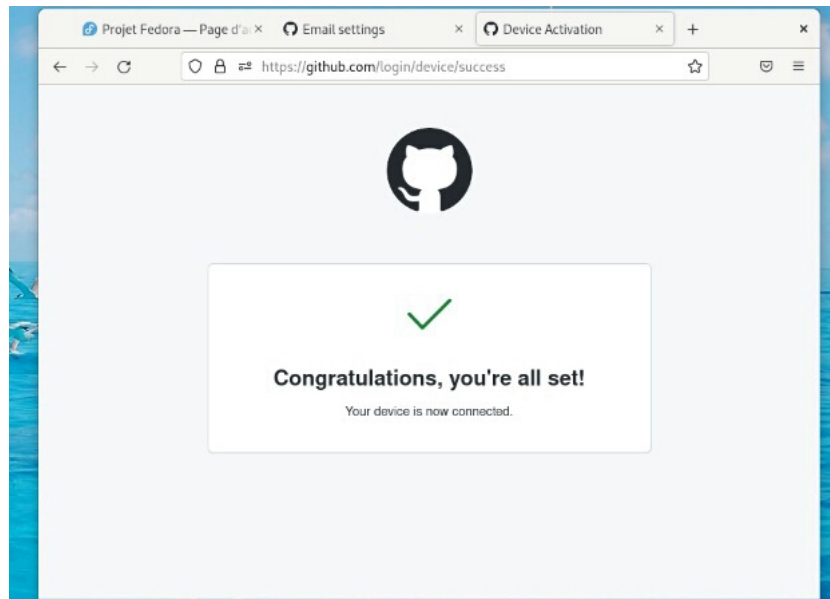


Рис. 4.10: Настройка gh

## 9. Создание репозитория курса на основе шаблона

- Создал шаблон рабочего пространства (рис. 4.11)

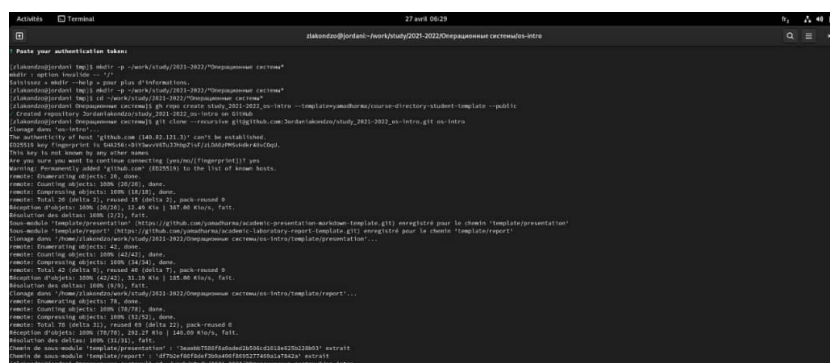


Рис. 4.11: Создание репозитория курса

## 10. Настройка каталога курса

- Перейдил в катог курса.
- Удалил лишние файлы.
- Создал необходимо каталоги. (рис. 4.12)

[illegible]

Рис. 4.12: Настройка каталога курса 1

- Отправил файлы на сервер. (рис. 4.13, 4.14)

[illegible]

Рис. 4.13: Настройка каталога курса 2

[illegible]

Рис. 4.14: Настройка каталога курса 2

## 5 Выводы

Таким образом, мы только что создали репозиторий курса “Операционные Системы”, основанный на шаблоне профессора на github. Это позволит всем студентам класса а одновременно работать над одним проектом.



## 6 Контрольные вопросы

### 1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначены?

- Система контроля версий (VCS) - программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.
- Такие системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы. Однако они могут с успехом применяться и в других областях, в которых ведётся работа с большим количеством непрерывно изменяющихся электронных документов.

### 2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

- Хранилище – репозиторий - место хранения всех версий и служебной информации.
- Commit - это команда для записи индексированных изменений в репозиторий.
- История – место, где сохраняются все коммиты, по которым можно посмотреть данные о коммитах.
- Рабочая копия – текущее состояние файлов проекта, основанное на

версии, загруженной из хранилища.

**3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.**

- Централизованные системы – это системы, в которых одно основное хранилище всего проекта, и каждый пользователь копирует необходимые ему файлы, изменяет и вставляет обратно.
- Пример – Subversion.
- Децентрализованные системы – система, в которой каждый пользователь имеет свой вариант репозитория и есть возможность добавлять и забирать изменения из репозитория.
- Например, Git.

**4. Опишите действия с VCS при единоличной работе с хранилищем.**

- Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория: 1 `git config --global user.name "Имя Фамилия"` 2 `git config --global user.email "work@mail"` и настроив utf-8 в выводе сообщений git: 1 `git config --global quotePath false` Для инициализации локального репозитория, расположенного, например, в каталоге `~/tutorial`, необходимо ввести в командной строке: Кулябов Д. С. и др. Операционные системы 25 1 `cd` 2 `mkdir tutorial` 3 `cd tutorial` 4 `git init` После это в каталоге `tutorial` появится каталог `.git`, в котором будет храниться история изменений. Создадим тестовый текстовый файл `hello.txt` и добавим его в локальный репозиторий:

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

- Воспользуемся командой `status` для просмотра изменений в рабочем каталоге, сделанных с момента последней ревизии: 1 `git status` Во время

работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами.

- Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл `.gitignore` с помощью сервисов. Для этого сначала нужно получить список имеющихся шаблонов:

1. `curl -L -s https://www.gitignore.io/api/list` Затем скачать шаблон, например, для C и C++
2. `curl -L -s https://www.gitignore.io/api/c » .gitignore`
3. `curl -L -s https://www.gitignore.io/api/c++ » .gitignore`

## 5. Опишите порядок работы с общим хранилищем VCS.

- Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый):

1. `ssh-keygen -C "Имя Фамилия"`

- Ключи хранятся в каталоге `~/.ssh/`. Существует несколько доступных серверов репозитория с возможностью бесплатного размещения данных. Например, <https://github.com/>. Для работы с ним необходимо сначала зайти на сайте <https://github.com/> учётную запись. Затем необходимо загрузить сгенерённый нами ранее открытый ключ. Для этого зайти на сайт <https://github.com/> под своей учётной записью и перейти в меню GitHub setting . После этого выбрать в боковом меню GitHub setting SSH-ключи и нажать кнопку Добавить ключ . Скопировав из локальной консоли ключ в буфер обмена

1. `cat ~/.ssh/id_rsa.pub | xclip -sel clip`

- И так вставляем ключ в появившееся на сайте поле. После этого можно создать на сайте репозиторий, выбрав в меню Репозитории Создать

репозиторий , дать ему название и сделать общедоступным (публичным). Для загрузки репозитория из локального каталога на сервер выполняем следующие команды: 1 `git remote add origin`

2. `ssh://git@github.com//.git`

3. `git push -u origin master`

- Далее на локальном компьютере можно выполнять стандартные процедуры для работы с git при наличии центрального репозитория.

#### 6. Каковы основные задачи, решаемые инструментальным средством git?

- У Git две основных задачи:

1. Первая — хранить информацию обо всех изменениях в вашем коде, начиная с самой первой строчки,
2. А вторая — обеспечение удобства командной работы над кодом.

#### 7. Назовите и дайте краткую характеристику командам git.

Команда	Описание действия
<code>git init</code>	создание основного дерева репозитория
<code>git pull</code>	получение обновлений (изменений) текущего дерева из центрального репозитория
<code>git status</code>	просмотр списка изменённых файлов в текущей директории
<code>git push</code>	отправка всех произведённых изменений локального дерева в центральный репозиторий
<code>git diff</code>	просмотр текущих изменения
<code>git add .</code>	добавить все изменённые и/или созданные файлы и/или каталоги

Команда	Описание действия
<code>git add</code> имена_файлов	добавить конкретные изменённые и/или созданные файлы и/или каталоги
<code>git rm</code> имена_файлов	удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории)
<code>git commit</code> <code>-am 'Коммит'</code>	сохранить все добавленные изменения и все изменённые файлы
<code>git commit</code>	сохранить добавленные изменения с внесением комментария через встроенный редактор
<code>git checkout</code> <code>-b имя_ветки</code>	создание новой ветки, базирующейся на текущей
<code>git checkout</code> имя_ветки	переключение на некоторую ветку
<code>git push</code> origin имя_ветки	отправка изменений конкретной ветки в центральный репозиторий
<code>git branch</code> <code>-d имя_ветки</code>	удаление локальной уже слитой с основным деревом ветки
<code>git branch</code> <code>-D имя_ветки</code>	принудительное удаление локальной ветки
<code>git push</code> origin :имя_ветки	удаление ветки с центрального репозитория

## 8. Приведите примеры использования при работе с локальным и удалённым репозиториями

- Для инициализации локального репозитория, расположенного, напри-

мер, в каталоге ~/tutorial, необходимо ввести в командной строке:

1. cd
  2. mkdir tutorial
  3. cd tutorial
  4. git init
- После это в каталоге tutorial появится каталог .git, в котором будет храниться история изменений. Создадим тестовый текстовый файл hello.txt и добавим его в локальный репозиторий:
1. echo 'hello world' > hello.txt
  2. git add hello.txt 3 git commit -am 'Новый файл' Воспользуемся командой status для просмотра изменений в рабочем каталоге, сделанных с момента последней ревизии:
  3. git status

#### **9. Что такое и зачем могут быть нужны ветви (branches)?**

- Ветка (англ. branch) — это последовательность коммитов, в которой ведётся параллельная разработка какого-либо функционала. Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций: чтобы не повредить основному проекту, создается новая ветка специально для экспериментов.

#### **10. Как и зачем можно игнорировать некоторые файлы при commit?**

- Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты. В Git нет специальной команды для указания игнорируемых файлов: вместо этого необходимо вручную отредактировать файл .

- Временно игнорировать изменения в файле можно с командой `git update-index-assumeunchanged`