```
---
## Front matter
title: "Лабораторная работа №2"
subtitle: "Управление версиями"
author: "Pavlova Polina"
## Generic otions
lang: ru-RU
toc-title: "Содержание"
## Bibliography
bibliography: bib/cite.bib
csl: pandoc/csl/gost-r-7-0-5-2008-numeric.csl
## Pdf output format
toc: true # Table of contents
toc-depth: 2
lof: true # List of figures
lot: true # List of tables
fontsize: 12pt
linestretch: 1.5
papersize: a4
documentclass: scrreprt
## I18n polyglossia
polyglossia-lang:
  name: russian
  options:
      - spelling=modern
      - babelshorthands=true
polyglossia-otherlangs:
  name: english
## I18n babel
babel-lang: russian
babel-otherlangs: english
## Fonts
mainfont: PT Serif
romanfont: PT Serif
sansfont: PT Sans
monofont: PT Mono
mainfontoptions: Ligatures=TeX
romanfontoptions: Ligatures=TeX
sansfontoptions: Ligatures=TeX,Scale=MatchLowercase
monofontoptions: Scale=MatchLowercase,Scale=0.9
## Biblatex
biblatex: true
biblio-style: "gost-numeric"
biblatexoptions:
  - parentracker=true
  - backend=biber
  - hyperref=auto
  - language=auto
  - autolang=other*
  - citestyle=gost-numeric
## Pandoc-crossref LaTeX customization
figureTitle: "Рис."
tableTitle: "Таблица"
listingTitle: "Листинг"
lofTitle: "Список иллюстраций"
lotTitle: "Список таблиц"
lolTitle: "Листинги"
## Misc options
indent: true
header-includes:
  - \usepackage{indentfirst}
```

```
  - \usepackage{float} # keep figures where there are in the text
  - \floatplacement{figure}{H} # keep figures where there are in the text
---
# Цель работы
Изучить идеологию и применение средств контроля версий. Освоить умения по работе с git.

# Задание
Сделать базовую настройку git для дальнейшей работы.

# Теоретическое введение
Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями.

# Выполнение лабораторной работы
Установить git-flow в Fedora Linux (рис. 1.2)
```

```
cd /tmp
wget --no-check-certificate -q https://raw.github.com/petervanderdoes
 ↪  /gitflow/develop/contrib/gitflow-installer.sh
chmod +x gitflow-installer.sh
sudo ./gitflow-installer.sh install stable
```

Рис1.1 Листинг установки git-flow

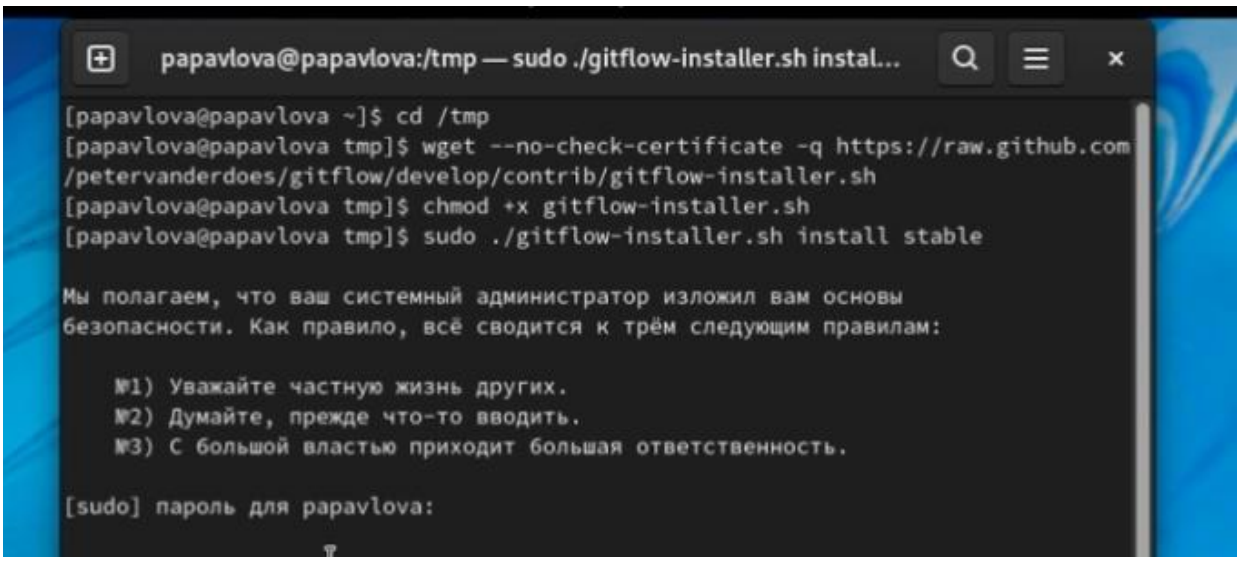

Рис1.2 Установка git-flow в Fedora Linux

Установить gh в fedora Linux (рис. 1.4)

```
sudo dnf install gh
```

Рис1.3 Листинг установки gh

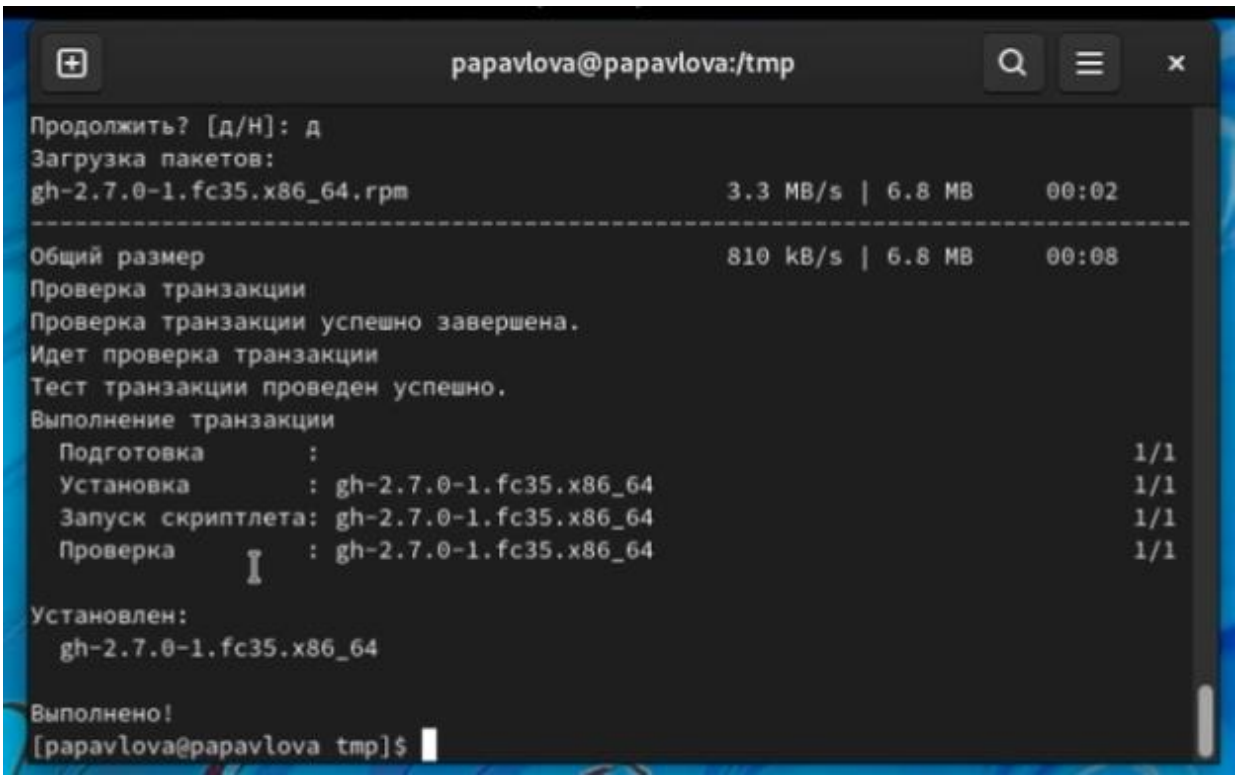

Рис1.4 Установка gh

Задать имя и email владельца репозитория (рис. 1.6)

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

Рис1.5 Листинг задания имени и email владельца репозитория

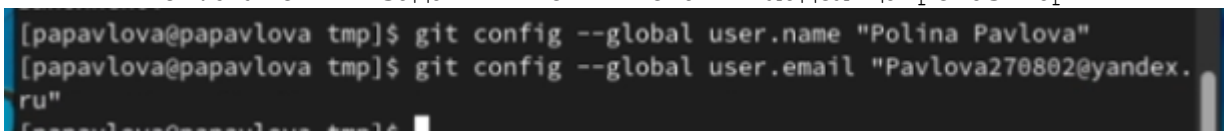

Рис1.6 Указывание имени и email владельца репозитория

Настроить utf-8 в выводе сообщений git (рис. 1.8)

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

Рис1.7 Листинг настройки utf-8

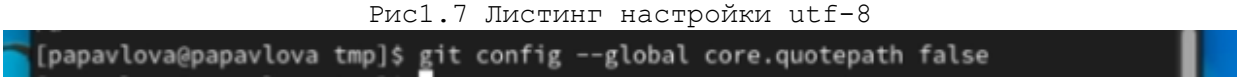

Рис1.8 Настройка utf-8 в выводе сообщений

Настройте верификацию и подписание коммитов git. – Зададим имя начальной ветки (будем называть её master). (рис. 1.10)

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

Рис1.9 Листинг задания начальной ветки

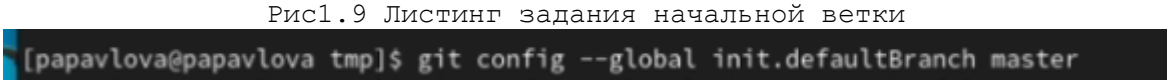

Рис1.10 Задание имени начальной ветки

Параметр autocrlf: (рис. 1.12)

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

Рис1.11 Листинг параметра autocrlf

```
[papavlova@papavlova tmp]$ git config --global core.autocrlf input
```

Рис1.12 Параметр autocrlf

Параметр safecrlf:

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

Рис1.13 Листинг параметра safecrlf

```
[papavlova@papavlova tmp]$ git config --global core.safecrlf warn
```

Рис 1.14 Параметр safecrlf

Создайте ключи ssh. По алгоритму rsa с ключём размером 4096 бит: (рис. 2.2)

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

Рис2.1 Листинг создания ключа ssh

```
Created directory '/home/papavlova/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/papavlova/.ssh/id_rsa
Your public key has been saved in /home/papavlova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:VFBmLVGjFNPDqG19vnaODm8GkY2+XAbAkZbifatwyKw papavlova@papavlova
The key's randomart image is:
+---[RSA 4096]----+
|        .+@%o    |
|        .=Oo=.   |
|       ..*.* =   |
|       .o + B o  |
|       oSo o *   |
|        = . + +  |
|       . o o.= . |
|      E   . oo=..|
|             =+o.|
+----[SHA256]-----+
```

Рис2.2 Создание ключа ssh

По алгоритму ed25519 (рис. 2.4)

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

Рис2.3 Листинг алгоритма ed25519

```
Enter file in which to save the key (/home/papavlova/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/papavlova/.ssh/id_ed25519
Your public key has been saved in /home/papavlova/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:fjAXUnI/wEZI8yK+fVBUt9O2mBCMto5/VZ2iyl0iq9w papavlova@papavlova
The key's randomart image is:
+--[ED25519 256]--+
|       .++Oo. .  |
|        .Xo+.. o |
|       . +.=.o o =|
|      . . = ..oo=o|
|       . S . .oo. |
|        = B o o   |
|       . = B +    |
|       . .B o     |
|        o.E.      |
+----[SHA256]-----+
```

Рис2.4 Алгоритм ed25519

Создайте ключи pgp. Генерируем ключ. (рис.3.2)

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

Рис3.1 Листинг генерации ключа

– Из предложенных опций выбираем: – тип RSA and RSA; – размер 4096; – выберите срок действия; значение по умолчанию— 0 (срок действия не истекает никогда). – GPG запросит личную информацию, которая сохранится в ключе: – Имя (не менее 5 символов). – Адрес электронной почты. – При вводе email убедитесь, что он соответствует адресу, используемому на GitHub. – Комментарий. Можно ввести что угодно или нажать клавишу ввода, чтобы оставить это поле пустым.

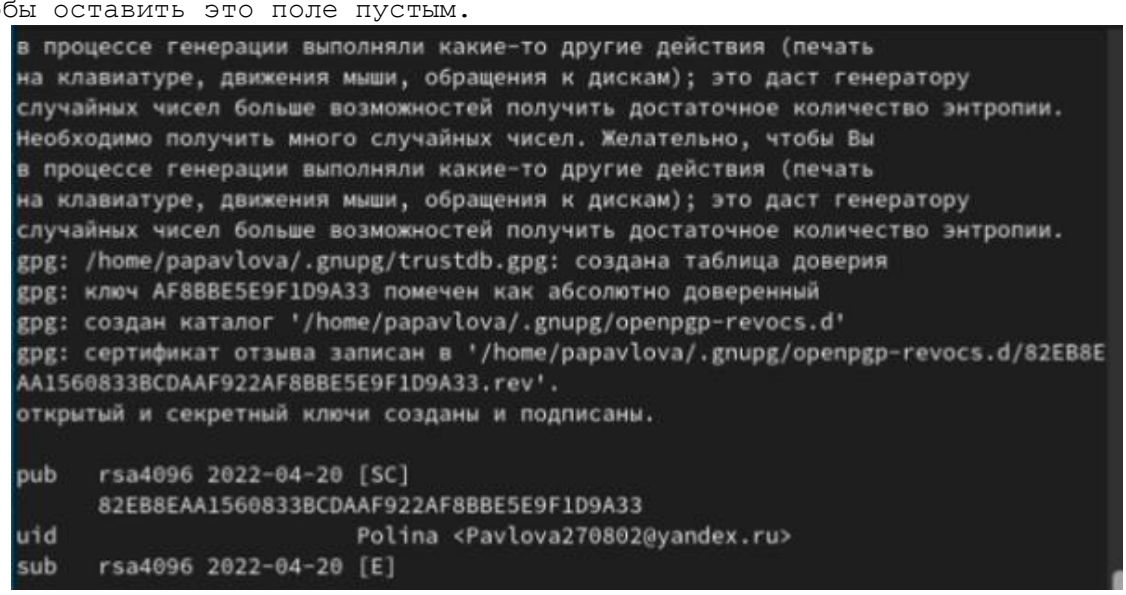

Рис3.2 Генерация ключа pgp

Добавление ключа pgp в GitHub. Выводим список ключей и копируем отпечаток приватного ключа: (рис. 4.2)

```
gpg --list-secret-keys --keyid-format LONG
```

Рис4.1 Листинг вывода списка ключей

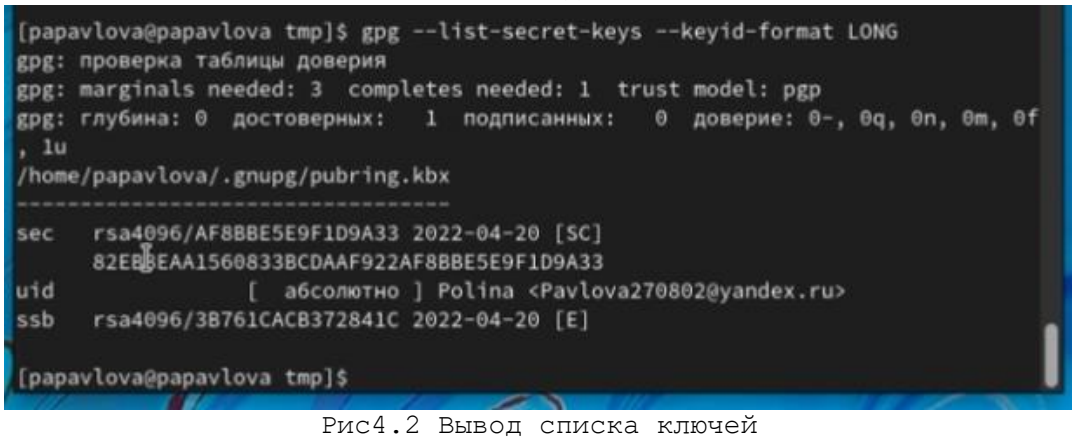

Рис4.2 Вывод списка ключей

Скопируйте ваш сгенерированный PGP ключ в буфер обмена: (рис. 4.3)

```
gpg --armor --export <PGP Fingerprint> | xclip -sel clip
```

Рис4.3 Листинг копии сгенерированного ключа в буфер обмена

Перейдите в настройки GitHub (https://github.com/settings/keys), нажмите на кнопку New GPG key и вставьте полученный ключ в поле ввода. (рис. 4.4)

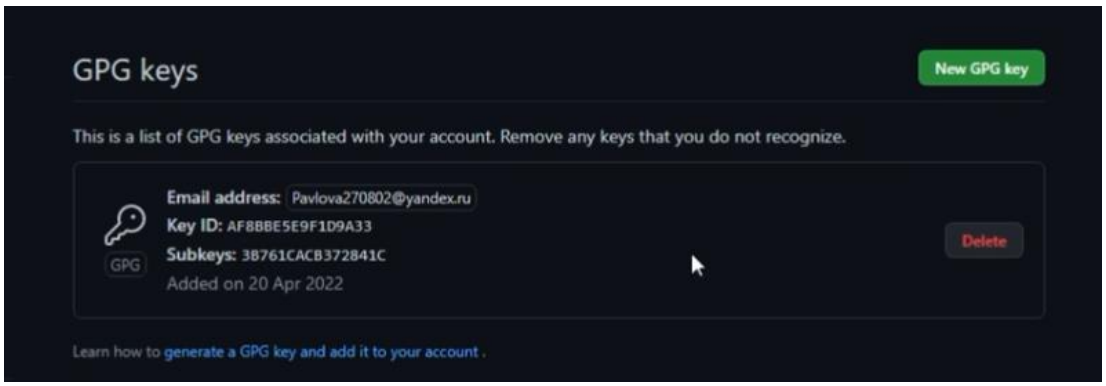

Рис4.4 Вставка полученного ключа в GitHub

Настройка автоматических подписей коммитов git. Используя введёный email, укажите Git применять его при подписи коммитов: (рис. 5.2)

```
git config --global user.signingkey <PGP Fingerprint>
git config --global commit.gpgsign true
git config --global gpg.program $(which gpg2)
```

Рис5.1 Листинг настройки автоматических подписей коммитов git

```
[papavlova@papavlova tmp]$ git config --global user.signingkey AF8BBE5E9F1D9A33
[papavlova@papavlova tmp]$ git config --global commit.gpgsign true
[papavlova@papavlova tmp]$ git config --global gpg.programm $(which gpg2)
```

Рис5.2 Настройка автоматических подписей коммитов git.

Настройка gh. Авторизация (рис 6.2).

```
gh auth login
```

Рис6.1 Листинг авторизации

```
[papavlova@papavlova Операционные системы]$ 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/papavlova/.ssh/id_ed25519.pub
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: D134-8775
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/papavlova/.ssh/id_ed25519.pub
✓ Logged in as Terrorochka
```

Рис6.2 Авторизация

Создание репозитория курса на основе шаблона. (рис. 7.2) Необходимо создать шаблон рабочего пространства. – Например, для 2021–2022 учебного года и предмета «Операционные системы» (код предмета os-intro) создание репозитория примет следующий вид:

```
mkdir -p ~/work/study/2021-2022/"Операционные системы"
cd ~/work/study/2021-2022/"Операционные системы"
gh repo create study_2021-2022_os-intro
 ↪  --template=yamadharma/course-directory-student-template --public
git clone --recursive
 ↪  git@github.com:<owner>/study_2021-2022_os-intro.git os-intro
```

Рис7.1 Листинг создания репозитория курса

Настройка каталога курса. Перейдите в каталог курса. Удалите лишние файлы. Создайте необходимые каталоги. Отправьте файлы на сервер. (рис. 7.5-7.6)

```
cd ~/work/study/2021-2022/"Операционные системы"/os-intro
```

Рис7.1 Листинг перехода в каталог курса

```
rm package.json
```

Рис7.2 Листинг удаления лишних файлов

```
make COURSE=os-intro
```

Рис7.3 Листинг создания необходимых каталогов

```
git add .
git commit -am 'feat(main): make course structure'
git push
```

Рис7.4 Листинг отправки файлов на сервер

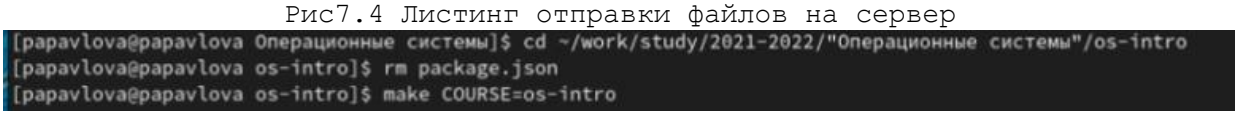

Рис7.5 Переход в каталог курса. Удаление лишних файлов. Создание необходимых каталогов.

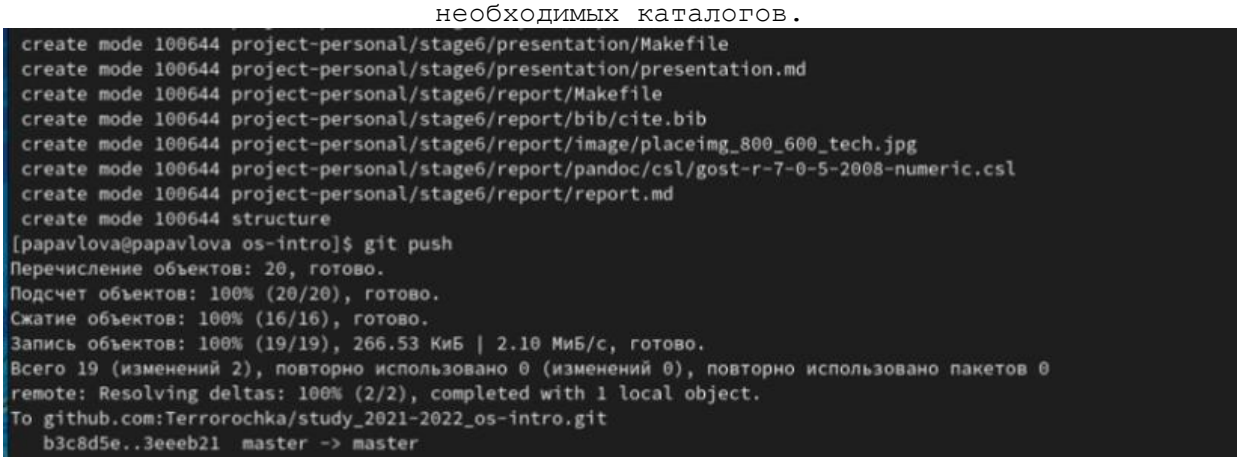

Рис7.6 Отправка файлов на сервер

Выводы

Базовая конфигурация git была создана. Был создан локальный каталог для выполнения заданий по предмету.

Ответы на контрольные вопросы

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

VCS –
программное обеспечение для облегчения работы с изменяющейся информацией. Применяется при работе нескольких человек над одним проектом. 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 - место хранения всех версий и служебной информации.
Commit - синоним версии; процесс создания новой версии.
История – изменения в репозитории.
Рабочая копия - 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: Простота использования. Вся история — всегда в едином общем хранилище. Нужно подключение к сети. Резервное копирование нужно только одному хранилищу. Удобство разделения прав доступа к хранилищу. Почти все изменения навсегда попадают в общее хранилище.
Децентрализованные: Двухфазный commit: 1) запись в локальную историю; 2) пересылка изменений другим. Подключение к сети не нужно. Локальные хранилища могут служить резервными копиями. Локальное хранилище контролирует его владелец, но общее — администратор. Возможна правка локальной истории перед отправкой на сервер.

4. Опишите действия с VCS при единоличной работе с хранилищем.
Создаётся новая версия в хранилище, делаем правки, после чего создаётся новая версия с правками. Так до конца работы над проектом.

5. Опишите порядок работы с общим хранилищем VCS.
Создаётся новая версия в хранилище. Каждый участник репозитория может получить локальную версию проекта и вносить изменения.

6. Каковы основные задачи, решаемые инструментальным средством git?
Хранение информации о всех изменениях в коде. Обеспечение удобства для командной работы над кодом.

7. Назовите и дайте краткую характеристику командам 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 изменения: git diff
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 локальной директории): git rm имена_файлов
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
отправка изменений конкретной ветки в центральный репозиторий: git push origin имя_ветки
слияние ветки с текущим деревом: git merge --no-ff имя_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

8. Приведите примеры использования при работе с локальным и удалённым репозиториями.
Локальный репозиторий – репозиторий, расположенный на локальном компьютере разработчика. Именно с ним работает программист. Удаленный репозиторий — репозиторий, находящийся на сервере. Это общий репозиторий, в который приходят все изменения.

9. Что такое и зачем могут быть нужны ветви (branches)?
Ветка — 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.

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

Игнорируемые файлы обычно представляют собой файлы для конкретной платформы или автоматически созданные файлы из систем сборки. Игнорируемые файлы отслеживаются в специальном файле. gitignore, который регистрируется в корневом каталоге репозитория. Он использует шаблоны подстановки для сопоставления имен файлов с подстановочными знаками.