

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

## ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

дисциплина: Архитектура компьютеров “Операционные системы”

Студент: Оганнисян Г.А.

Группа: НБИбд-03-24

№ ст. билета: 1132243806

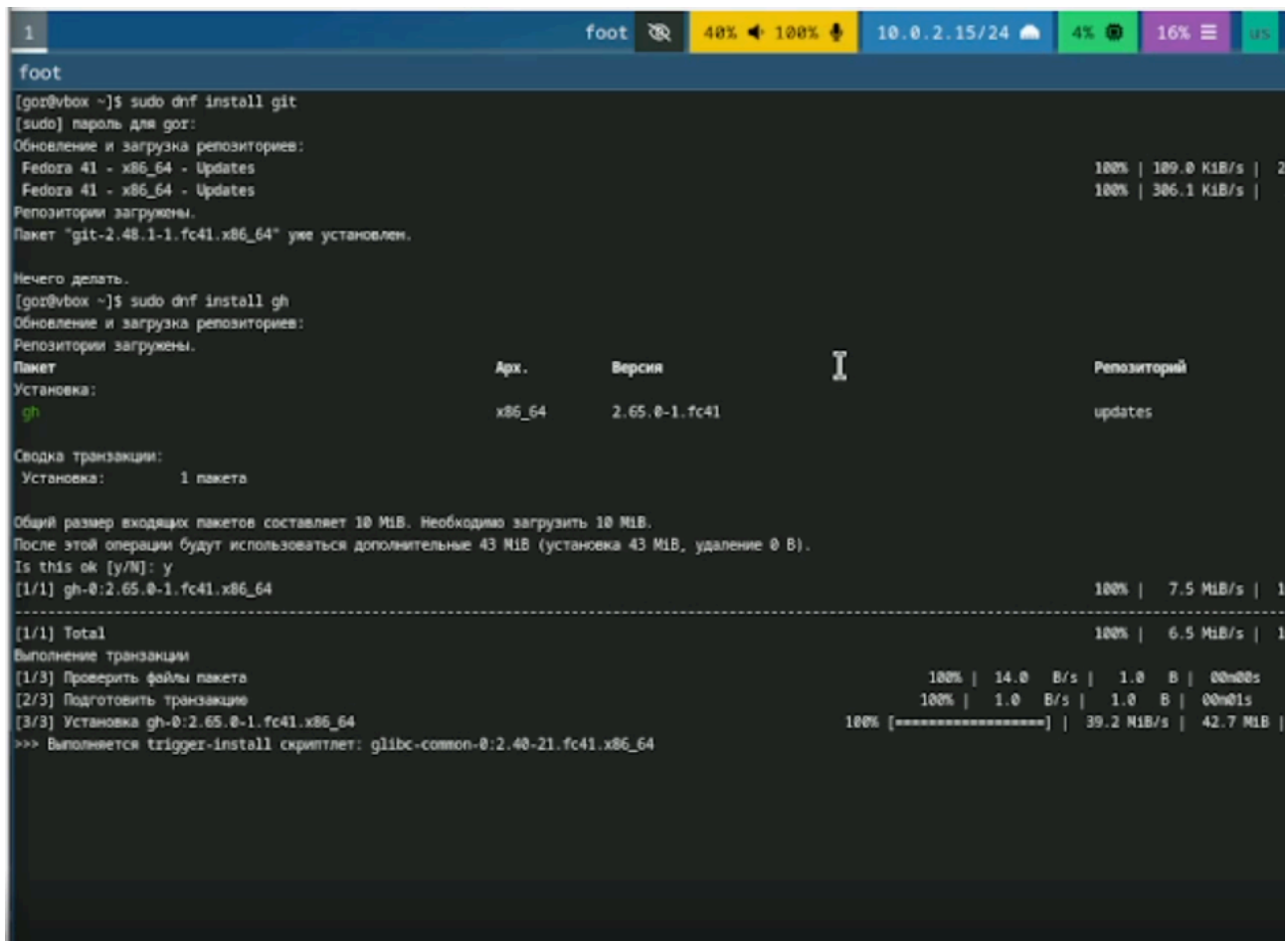
МОСКВА

2025 г.

# Цель работы

Целью данной работы является изучение идеологии git, применение средств контроля версий и освоение умений работы с git.

## Описание результатов выполнения работы



```
1 foot 48% 100% 10.0.2.15/24 4% 16% us
foot
[gor@vbox ~]$ sudo dnf install git
[sudo] пароль для gor:
Обновление и загрузка репозитория:
Fedora 41 - x86_64 - Updates 100% | 109.0 KiB/s | 2
Fedora 41 - x86_64 - Updates 100% | 306.1 KiB/s |
Репозитории загружены.
Пакет "git-2.48.1-1.fc41.x86_64" уже установлен.

Нечего делать.
[gor@vbox ~]$ sudo dnf install gh
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет Арх. Версия Репозиторий
Установка:
gh x86_64 2.65.0-1.fc41 updates

Сводка транзакции:
Установка: 1 пакета

Общий размер входящих пакетов составляет 10 MiB. Необходимо загрузить 10 MiB.
После этой операции будут использоваться дополнительные 43 MiB (установка 43 MiB, удаление 0 B).
Is this ok [y/N]: y
[1/1] gh-0:2.65.0-1.fc41.x86_64 100% | 7.5 MiB/s | 1
-----
[1/1] Total 100% | 6.5 MiB/s | 1
Выполнение транзакции
[1/3] Проверить файлы пакета 100% | 14.0 B/s | 1.0 B | 00m00s
[2/3] Подготовить транзакцию 100% | 1.0 B/s | 1.0 B | 00m01s
[3/3] Установка gh-0:2.65.0-1.fc41.x86_64 100% [=====] | 39.2 MiB/s | 42.7 MiB |
>>> Выполняется trigger-install скриплет: glibc-common-0:2.40-21.fc41.x86_64
```

Устанавливаем Git и gh.

```

foot
[gor@vbox ~]$ git config --global user.name "Tononari"
[gor@vbox ~]$ git config --global user.email "ogannisyan5476@gmail.com"
>
> ;2;13-
> "
[gor@vbox ~]$ git config --global user.name "Tononari"
[gor@vbox ~]$ git config --global user.email "ogannisyan5476@gmail.com"
[gor@vbox ~]$ git config --global core.quotepath false
[gor@vbox ~]$ git config --global init.defaultBranch master
[gor@vbox ~]$ git config --global core.autocrlf input
[gor@vbox ~]$ git config --global core.safecrlf warn
[gor@vbox ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/gor/.ssh/id_rsa):
Created directory '/home/gor/.ssh'.
Enter passphrase for "/home/gor/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/gor/.ssh/id_rsa
Your public key has been saved in /home/gor/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:I5cQqp7nAVeMp5yMUCk4P38sUnD15lwQa5TDCqEUphU gor@vbox
The key's randomart image is:
+---[RSA 4096]-----+
|oB60o .           |
|%o@.=. .          |
|=#o@ .           |
|o./ . . .         |
| + * .. S         |
| + o oo .         |
| . o              |
|                  |
|                  |
+-----[SHA256]-----+
[gor@vbox ~]$ ssh-keygen -t ed

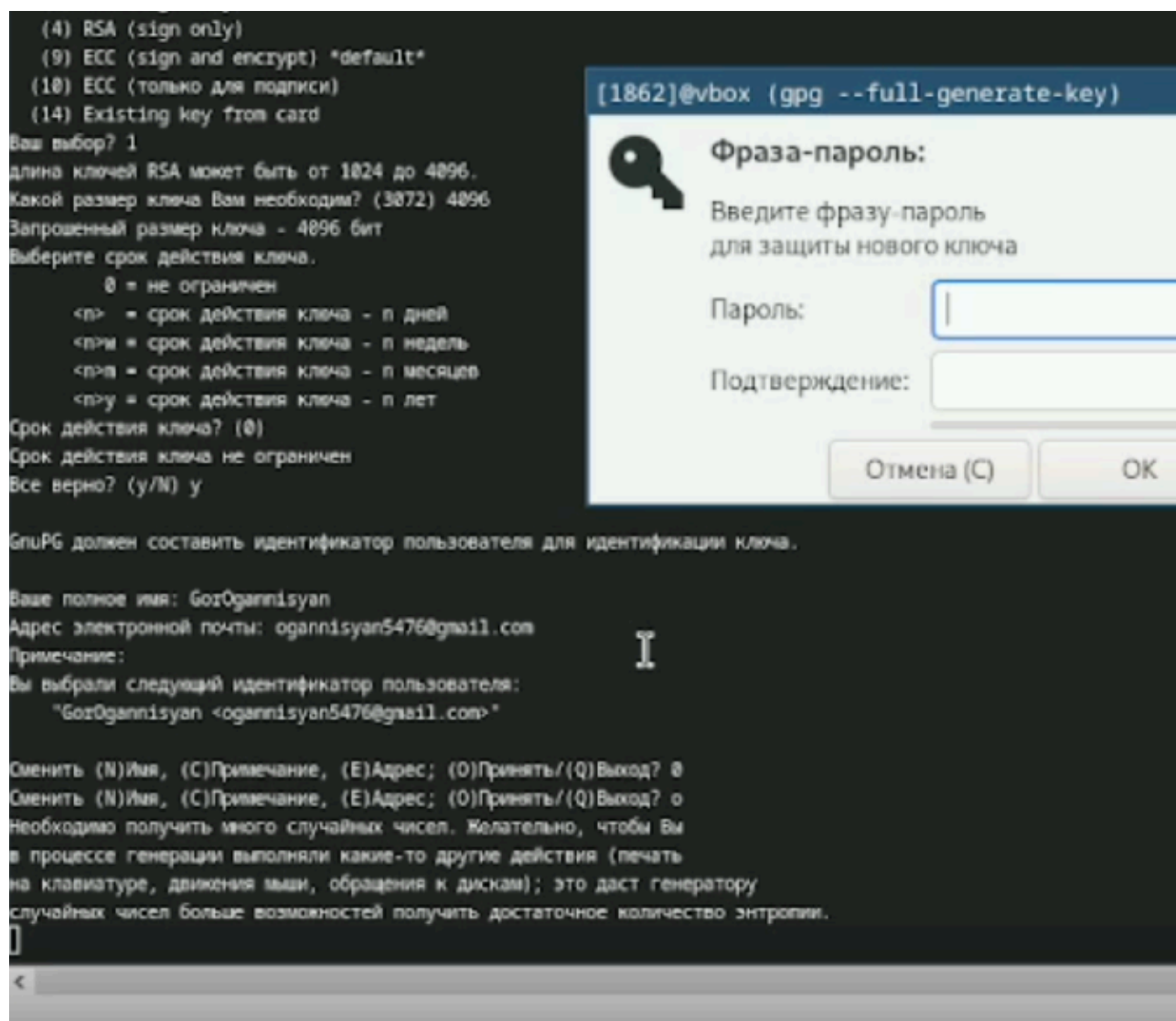
```

Делаем базовую настройку git.

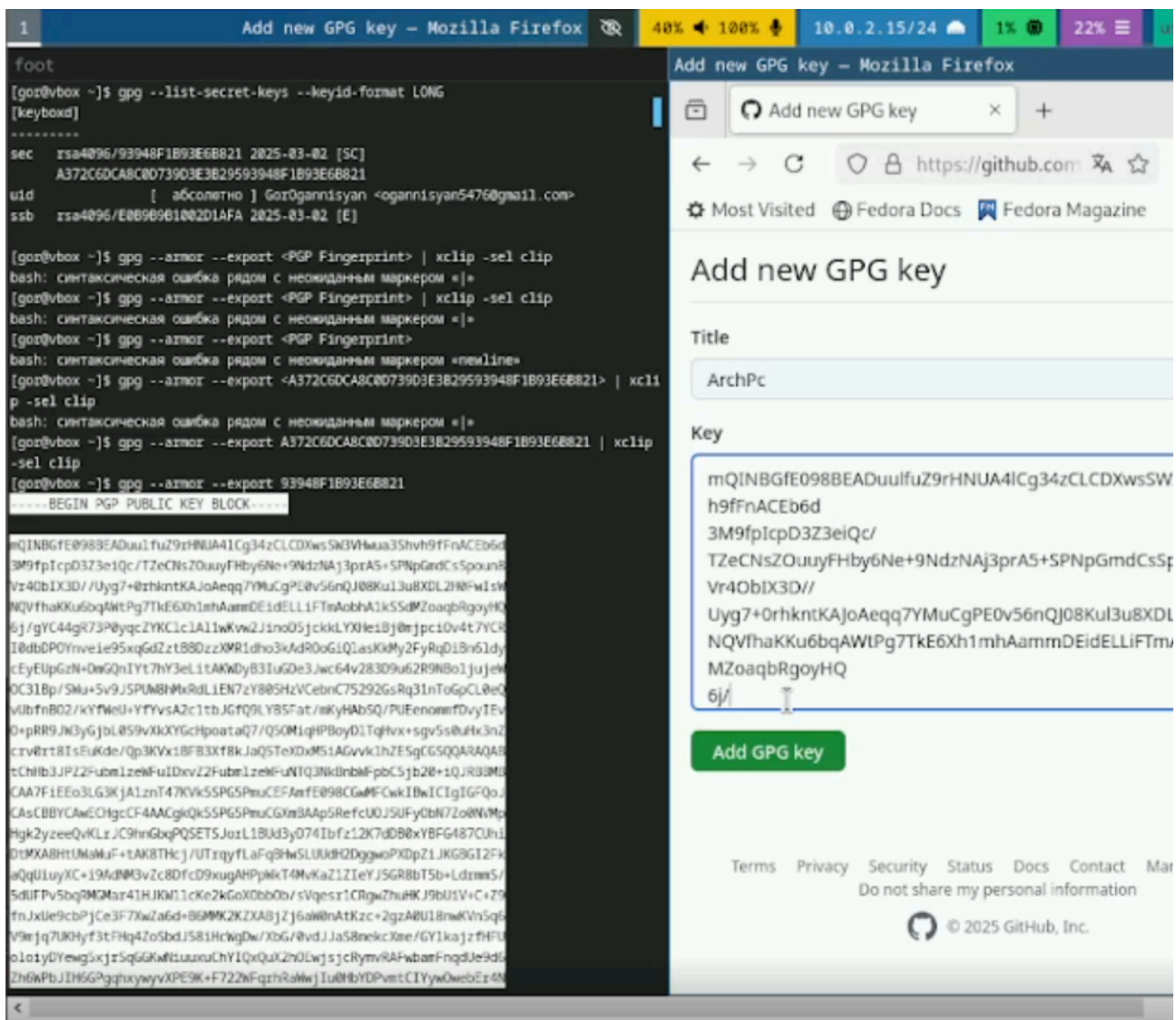
```
foot
|=#00 . |
|0./ . . |
| + ^ .. 5 |
| + 0 00 . |
| . 0 |
| |
|-----[SHA256]-----+
[gor@vbox ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/gor/.ssh/id_ed25519):
Enter passphrase for "/home/gor/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/gor/.ssh/id_ed25519
Your public key has been saved in /home/gor/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:FmH9uH1uCBb61RiVB+KqXRXKJBWkK5Ba8oVhxdTE40w gor@vbox
The key's randomart image is:
+---[ED25519 256]---+
| +.oE+. . .o |
|..+.o=B o..o |
| = ..++ + o+ . |
| . . . . + ++ . |
| S. ^ . |
| +. = + . |
| . o o + |
| . . o |
| . |
+-----[SHA256]-----+
[gor@vbox ~]$ gpg --full-generate-key
gpg (GnuPG) 2.4.5; 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: создан каталог '/home/gor/.gnupg'
Выберите тип ключа:
  (1) RSA and RSA
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (9) ECC (sign and encrypt) "default"
  (10) ECC (только для подписи)
  (14) Existing key from card
Ваш выбор? 1
```

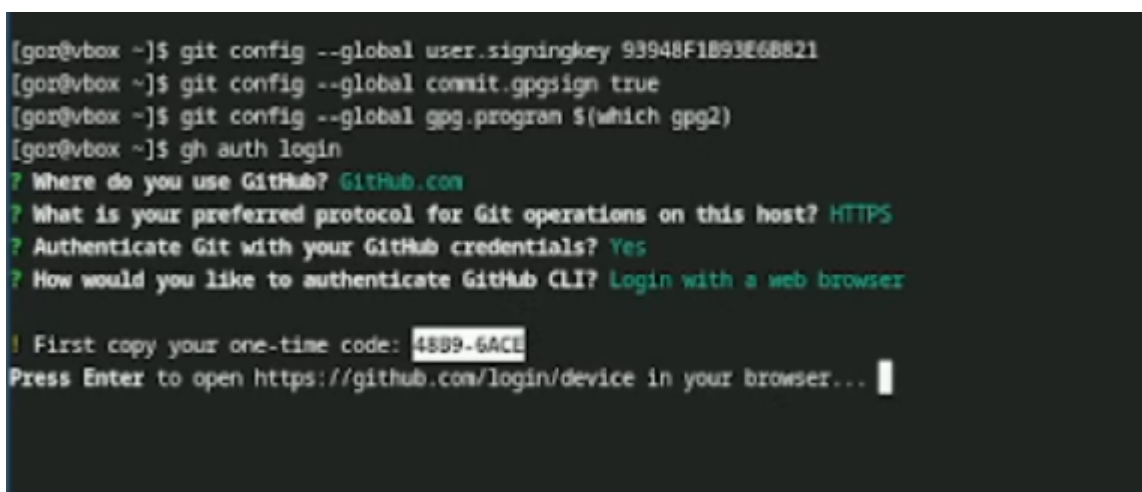
Создаем ключи ssh и pgp.



Делаем настройку github.



Добавляем pgr ключ в github.



Делаем настройку автоматический подписей коммитов и авторизуемся в gh.



```

! First copy your one-time code: 48B9-6ACE
Press Enter to open https://github.com/login/device in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as Tonomari
[gor@vbox ~]$ mkdir -p ~/work/study/2024-2025/"Operations System"
[gor@vbox ~]$ cd ~/work/study/2024-2025/Operations\ System/
[gor@vbox Operations System]$ ls
[gor@vbox Operations System]$ gh repo create study_2024-2025_os_intro --template=yamadharma/course-directory-student-template --public
✓ Created repository Tonomari/study_2024-2025_os_intro on GitHub
https://github.com/Tonomari/study_2024-2025_os_intro
[gor@vbox Operations System]$ git clone --recursive ogannisyan5476@gmail.com:Tonomari/study_2024-2025_

```

Создаем репозиторий курса на основе шаблона.

```

remote: Total 36 (delta 1), reused 21 (delta 0), pack-reused 0 (from 0)
Получение объектов: 100% (36/36), 19.38 КиБ | 431.00 КиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/igor/work/study/2024-2025/Operations System/os-intro/template/presentation»...
remote: Enumerating objects: 111, done.
remote: Counting objects: 100% (111/111), done.
remote: Compressing objects: 100% (77/77), done.
remote: Total 111 (delta 42), reused 100 (delta 31), pack-reused 0 (from 0)
Получение объектов: 100% (111/111), 102.17 КиБ | 1.16 МБ/с, готово.
Определение изменений: 100% (42/42), готово.
Клонирование в «/home/igor/work/study/2024-2025/Operations System/os-intro/template/report»...
remote: Enumerating objects: 142, done.
remote: Counting objects: 100% (142/142), done.
remote: Compressing objects: 100% (97/97), done.
remote: Total 142 (delta 60), reused 121 (delta 39), pack-reused 0 (from 0)
Получение объектов: 100% (142/142), 341.09 КиБ | 1.76 МБ/с, готово.
Определение изменений: 100% (60/60), готово.
Submodule path 'template/presentation': checked out 'c9b2712b4b2d431ad5086c9c72a82bd2fca1d4a6'
Submodule path 'template/report': checked out 'c26e22effe7b3e0495707d82ef561ab185f5c748'
[gor@vbox Operations System]$ ls
os-intro
[gor@vbox Operations System]$ cd os-intro/
[gor@vbox os-intro]$ ls
CHANGELOG.md config COURSE LICENSE Makefile package.json README.en.md README.git-flow.md README.md template
[gor@vbox os-intro]$ rm package.json
rm: удалить записанный от записи обычный файл 'package.json'?
[gor@vbox os-intro]$ ls
CHANGELOG.md config COURSE LICENSE Makefile package.json README.en.md README.git-flow.md README.md template
[gor@vbox os-intro]$ rm -rf package.json
rm: невозможно удалить 'package.json': Отказано в доступе
[gor@vbox os-intro]$ sudo rm -rf package.json
[gor@vbox os-intro]$ ls
CHANGELOG.md config COURSE LICENSE Makefile README.en.md README.git-flow.md README.md template
[gor@vbox os-intro]$ sudo -i
[root@vbox ~]# echo os-intro > COURSE
[root@vbox ~]# make
make: *** Не заданы цели и не найден make-файл. Останов.
[root@vbox ~]# git add .
fatal: не найден git репозиторий (или один из родительских каталогов): .git
[root@vbox ~]# ls
anaconda-ks.cfg COURSE
[root@vbox ~]#

```

Делаем настройку каталога курса.

## Выводы, согласованные с заданием работы

В данной лабораторной работе мы настроили git и gh для работы с github. Создали репозиторий для добавления своих работ туда.

## Ответы на контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Системы контроля версий (VCS) — это инструменты, которые помогают разработчикам управлять изменениями в исходном коде и других файлах проекта. Они предназначены для отслеживания изменений, сохранения истории изменений, совместной работы над проектом и обеспечения возможности возврата к предыдущим версиям.
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.
  - Хранилище (Repository): Центральное место, где хранятся все версии файлов проекта.
  - Commit (Коммит): Зафиксированное изменение в хранилище, содержащее набор изменений и сообщение о том, что было изменено.
  - История (History): Последовательность всех коммитов, показывающая, как проект изменялся со временем.
  - Рабочая копия (Working Copy): Локальная копия проекта на компьютере разработчика, с которой он работает и вносит изменения.
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.
  - Централизованные VCS (CVCS): Все изменения и история хранятся на центральном сервере. Примеры: Subversion (SVN), Perforce.



- Децентрализованные VCS (DVCS): Каждый разработчик имеет полную копию хранилища, включая всю историю изменений. Примеры: Git, Mercurial.
4. Опишите действия с VCS при единоличной работе с хранилищем. При единоличной работе разработчик создает локальное хранилище, вносит изменения, фиксирует их с помощью коммитов и может возвращаться к предыдущим версиям при необходимости.
  5. Опишите порядок работы с общим хранилищем VCS. Разработчики клонируют общее хранилище, работают с локальной копией, вносят изменения, коммитят их локально, а затем отправляют (push) изменения в общее хранилище. При необходимости они могут получать (pull) обновления от других разработчиков.
  6. Каковы основные задачи, решаемые инструментальным средством git? Git решает задачи управления версиями, обеспечивая возможность отслеживания изменений, совместной работы, управления ветками и слияния изменений.
  7. Назовите и дайте краткую характеристику командам git.
    - git init: Инициализация нового локального хранилища.
    - git clone: Клонирование удаленного хранилища.
    - git add: Добавление изменений в индекс (staging area).
    - git commit: Создание коммита.
    - git push: Отправка изменений в удаленное хранилище.
    - git pull: Получение изменений из удаленного хранилища.
    - git branch: Работа с ветками.
    - git merge: Слияние веток.
  8. Приведите примеры использования при работе с локальным и удаленным репозиторием.
    - Локальный репозиторий: git init, git add, git commit.
    - Удаленный репозиторий: git clone, git push, git pull.
  9. Что такое и зачем могут быть нужны ветви (branches)? Ветви позволяют разработчикам работать над различными задачами или функциями параллельно, не мешая друг другу. Они используются для разработки новых функций, исправления ошибок и экспериментов.
  10. Как и зачем можно интегрировать некоторые файлы при commit? Интеграция определенных файлов при коммите

позволяет разработчикам выборочно включать изменения в коммит, что помогает управлять изменениями более гибко и избегать включения ненужных или незавершенных изменений. Это делается с помощью команды `git add` перед `git commit`.