**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ  
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

*дисциплина: Операционные системы*

**Студент:** Кондратьев Арсений Вячеславович

**Группа:** НПИбд-01-21

**Ст. билет №:** 1132210645

Москва

2022

**Цель работы:** Изучить идеологию и применение средств контроля версий и освоить умения по работе с git.

**Ход работы:**

1. Создал учетную запись на Github(рис.1)

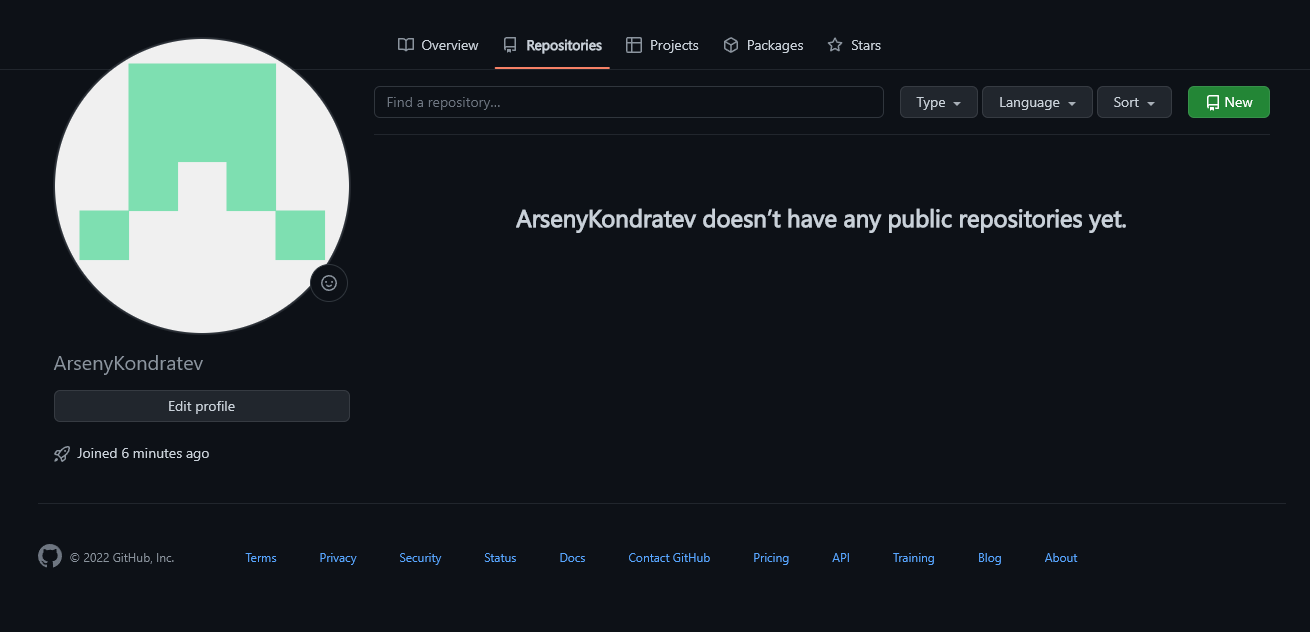


Рис. 1

1. Установил Git-flow(рис. 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

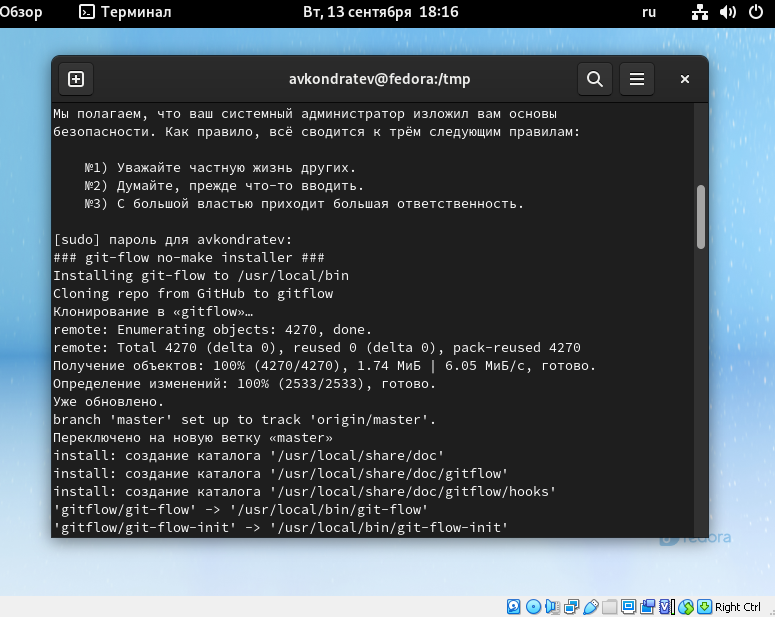


Рис. 2

1. Установил gh в Fedora Linux(рис.3)

> sudo dnf install gh

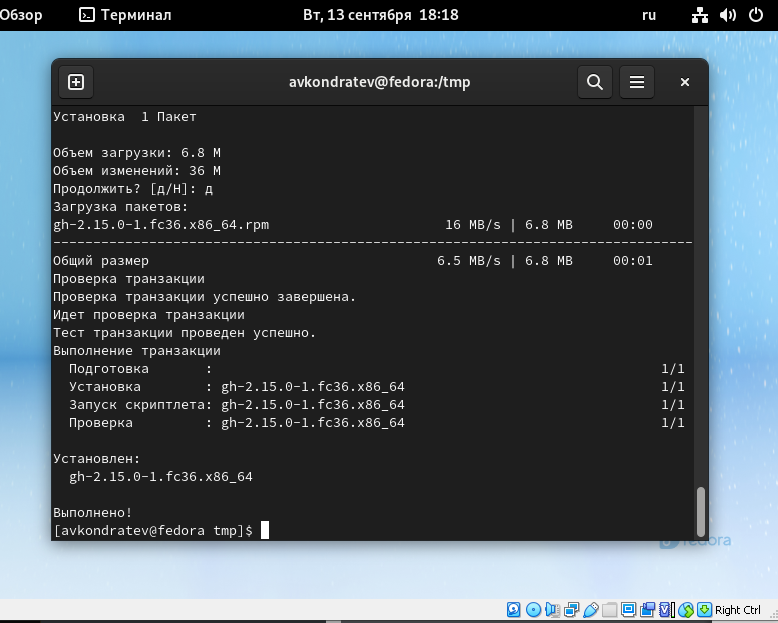


Рис. 3

1. Выполнил базовую настройку Git(рис. 4)

> git config --global user.name "KondratevArseny"

>git config --global user.email [avk2200@yandex.ru](mailto:avk2200@yandex.ru)

> git config --global core.quotepath false

> git config --global init.defaultBranch master

> git config --global core.autocrlf input> git config --global core.safecrlf warn

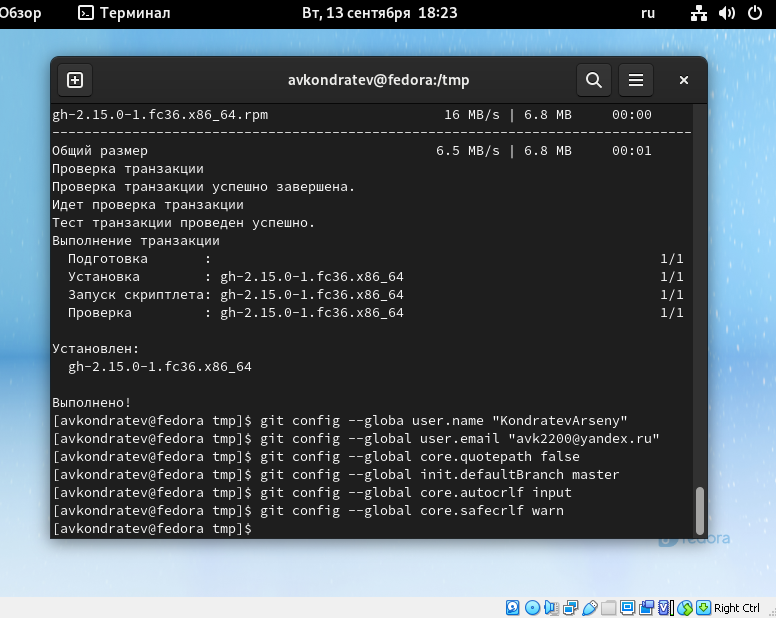


Рис. 4

1. Создал ключ SSH по алгоритму rsa(рис. 5)

> ssh-keygen -t rsa -b 4096

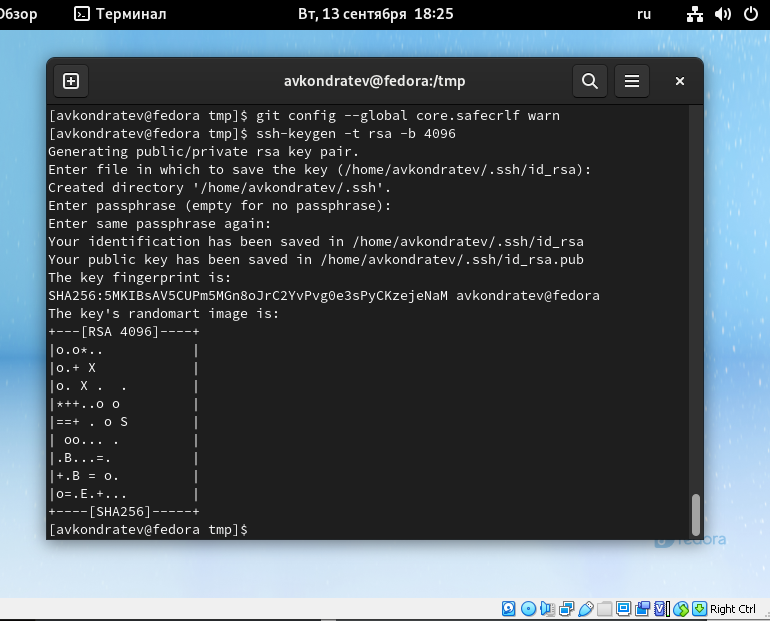


Рис. 5

1. Создал ключ SSH по алгоритму ed25519(рис. 6)

> ssh-keygen -t ed25519

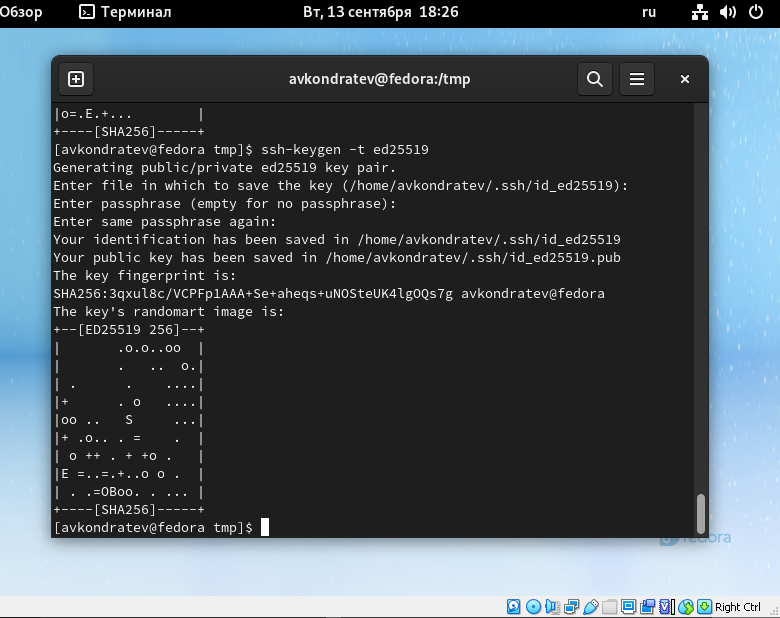


Рис. 6

1. Создал ключи pgp(рис. 7)

> gpg --full-generate-key

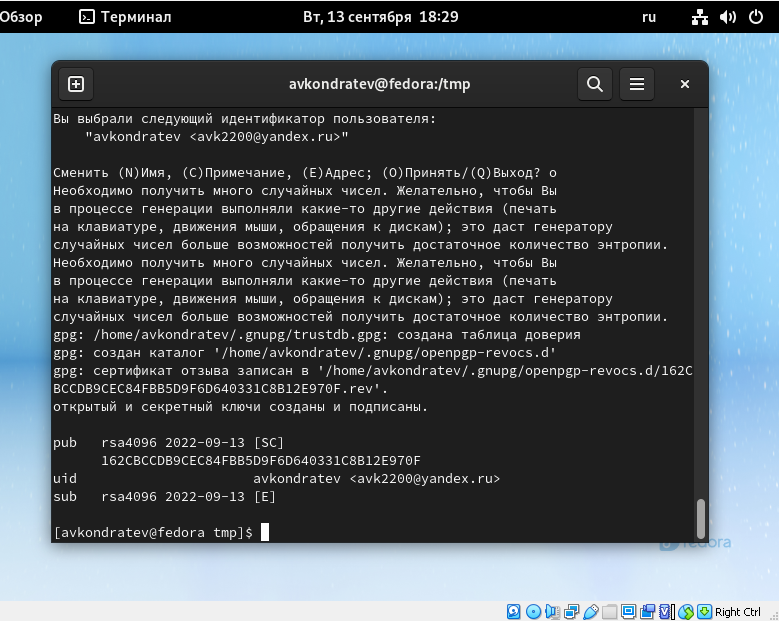


Рис. 7

1. Добавил ключ pgp в github(рис. 8, рис. 9)

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

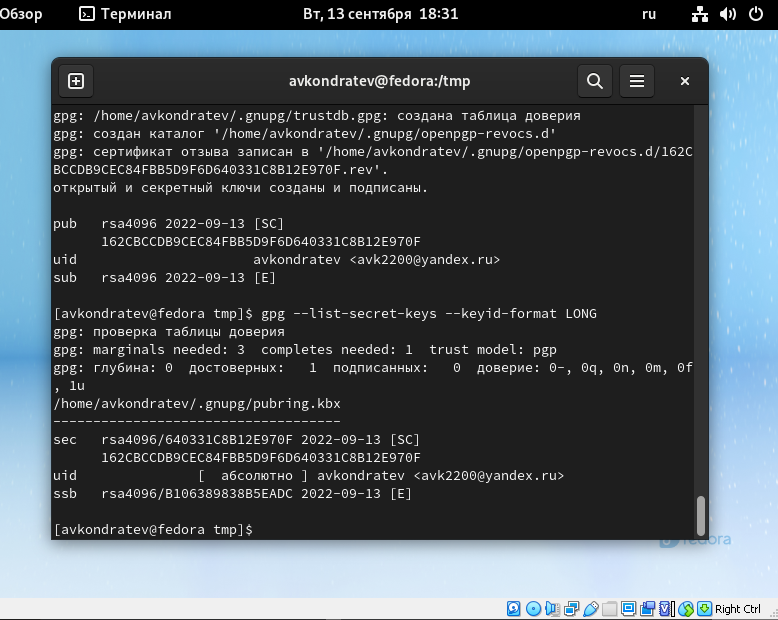


Рис.8(вывел список ключей и скопировал список приватного ключа)

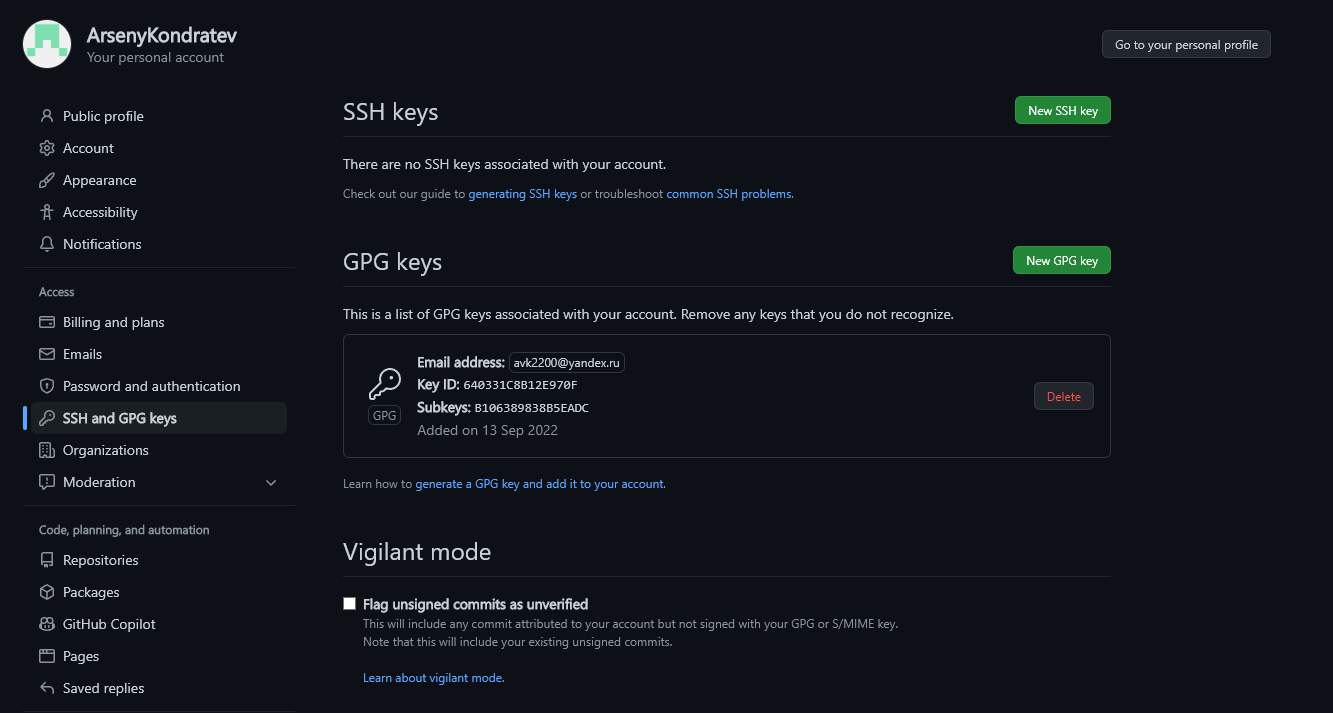


Рис. 9(Добавил ключ в Github)

1. Настроил автоматические подписи коммитов git(рис. 10)

> git config --global user.signingkey <PGP Fingerprint>

>git config --global commit.gpgsign true

>git config --global gpg.program $(which gpg2)

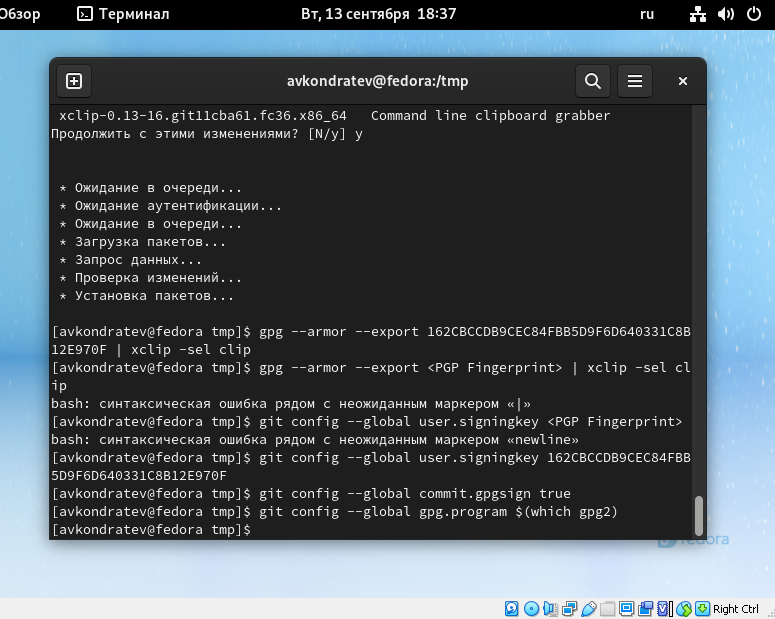


Рис. 10

1. Создал репозиторий курса на основе шаблона(рис. 11)

> 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:ArsenyKondratev study\_2021-2022\_os-intro.git os-intro

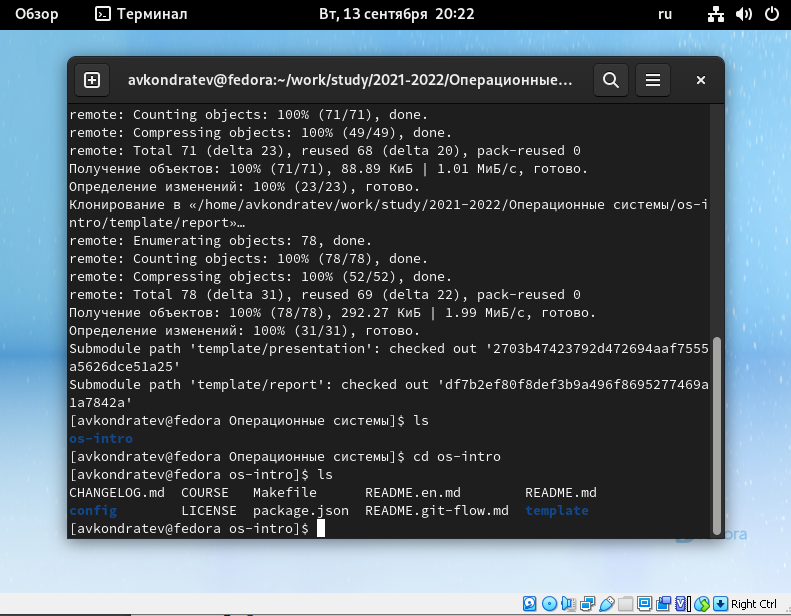


Рис. 11

1. Настроил каталог курса(рис. 12)

Удалил лишние файлы и создал необходимый каталог

>rm package.json

> make COURSE=os-intro

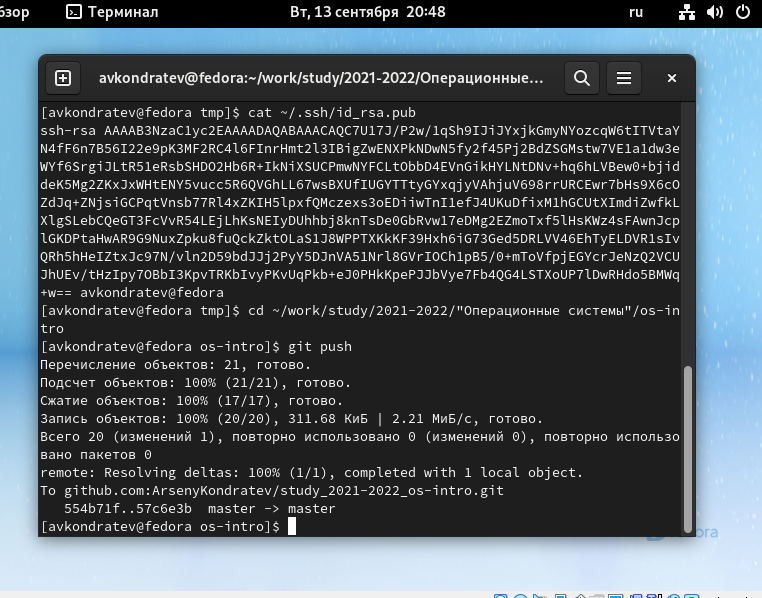


Рис. 12

**Вывод:** Я изучил идеологию и применение средств контроля версий и освоил умения по работе с git.

**Контрольные вопросы:**

1. Система контроля версий (Version Control System, VCS) — программное обеспечение для облегчения работы с изменяющейся информацией.

VCS 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.

1. Хранилище – место, где хранятся изменения кода.

Commit - снимок состояния проекта на текущий момент времени.

История – список снимков состояния проекта к которым можно при необходимости откатиться.

Рабочая копия - Рабочая копия является снимком одной версии проекта.

1. Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере.(CVS, Subversion)

Децентрализованные системы контроля версий (Distributed Version Control System, DVCS) позволяют хранить репозиторий (его копию) у каждого разработчика, работающего с данной системой.(Git, Mercurial)

1. Создаем свою ветку, базирующуюся на главной(git checkout -b имя\_ветки), вносим изменения, делаем снимок(git commit) и затем вносим эти изменения в свою ветку(git push)
2. Отдельные ветки разработчиков внедряются в общую master ветку
3. Git позволяет несокльким разработчикам с удобством работать над одним проектом. Возможность получать изменения, внесенные другим человеком и откатываться на прошлые версии в случае ошибок.
   1. создание основного дерева репозитория – git init
   2. получение обновлений (изменений) текущего дерева из центрального репозитория – git pull
   3. отправка всех произведённых изменений локального дерева в центральный репозиторий – git push
   4. просмотр списка изменённых файлов в текущей директории – git status
   5. просмотр текущих изменений – git diff
   6. добавить все изменённые и/или созданные файлы и/или каталоги – git add
   7. сохранить все добавленные изменения и все изменённые файлы – git commit
   8. создание новой ветки, базирующейся на текущей: - git checkout -b имя\_ветки
   9. переключение на некоторую ветку - git checkout имя\_ветки
   10. слияние ветки с текущим деревом - git merge --no-ff имя\_ветки
   11. удаление локальной уже слитой с основным деревом ветки - git branch -d имя\_ветки
   12. принудительное удаление локальной ветки - git branch -D имя\_ветки
4. С локальным: commit(снимок состояния проекта)

С удаленным: push(отправляем изменения) pull(загружаем изменения)

1. Это простой перемещаемый указатель на один из таких коммитов. Они нужны для того, чтобы разделять код. Например одна ветка у нас может быть основная для разработки. Если мы делаем новый функционал, то мы создаем новую ветку под него, а после окончания работы сливаем то, что мы сделали в основную ветку.
2. Во время работы могут появляться временные файлы, не несущие смысла для проекта. Их лучше не отправлять при использовании commit.