Лабораторная работа 2

Управление версиями

Арина Олеговна Аристова

Содержание

[Цель работы 1](#_Toc101910831)

[Задание 1](#_Toc101910832)

[Теоретическое введение 1](#_Toc101910833)

[Выполнение лабораторной работы 2](#_Toc101910834)

[Вывод 10](#_Toc101910835)

[Ответы на контрольные вопросы 10](#_Toc101910836)

# Цель работы

-Изучить идеологию и применение средств контроля версий.

-Освоить умения по работе с git.

# 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.

# Теоретическое введение

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.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельтакомпрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

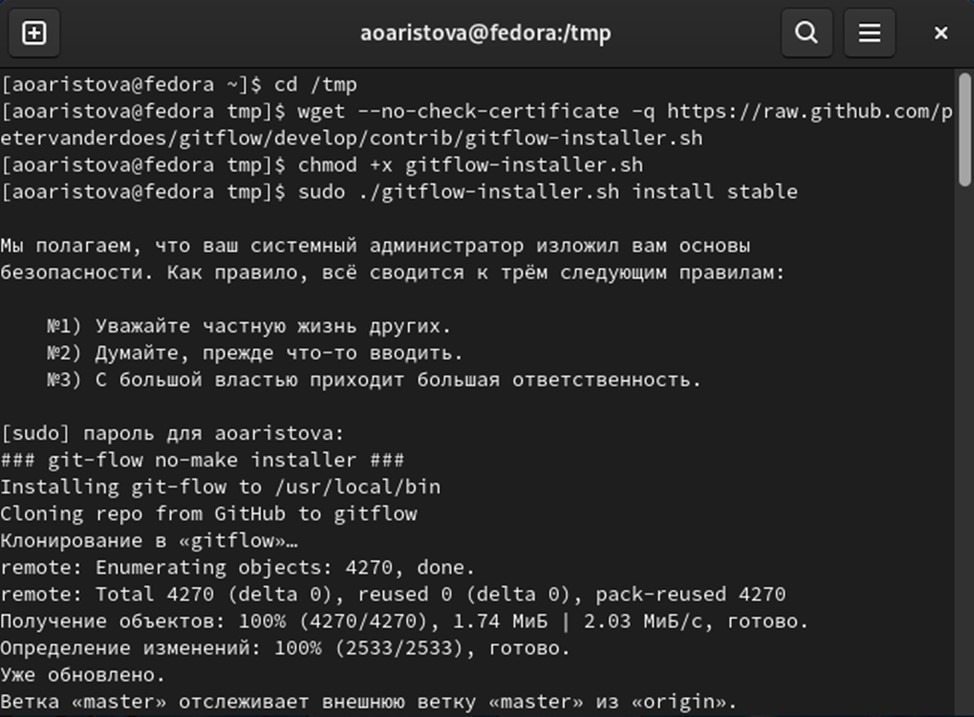
Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.

Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным. Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

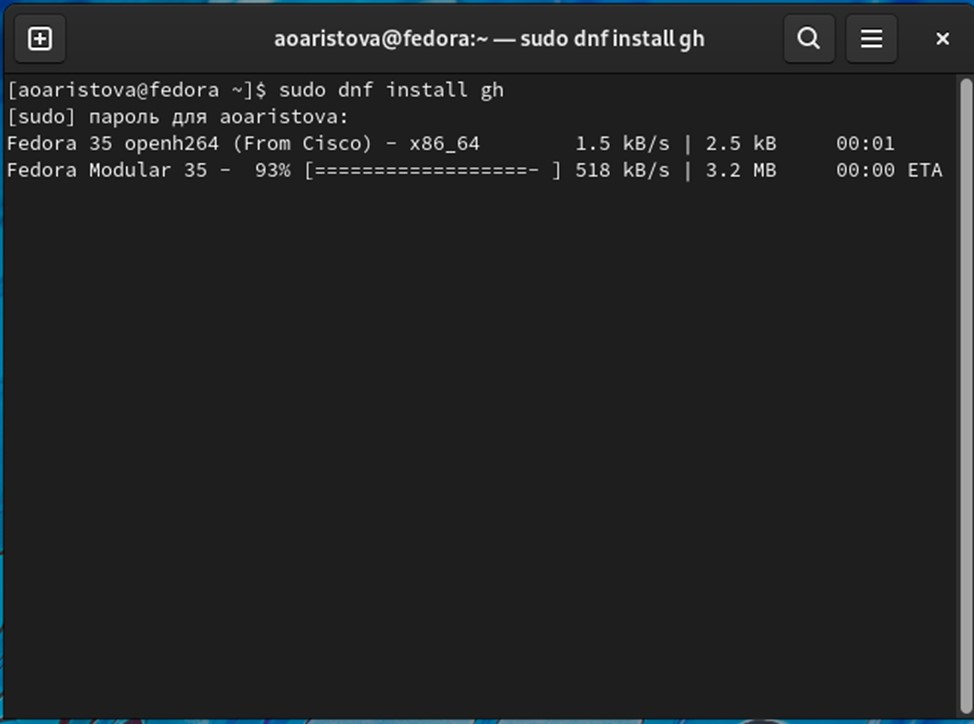
# Выполнение лабораторной работы

Предварительно создаю учетную запись и заполняю основные данные на https://github.com. Устанавливаю git-flow в Fedora Linux.

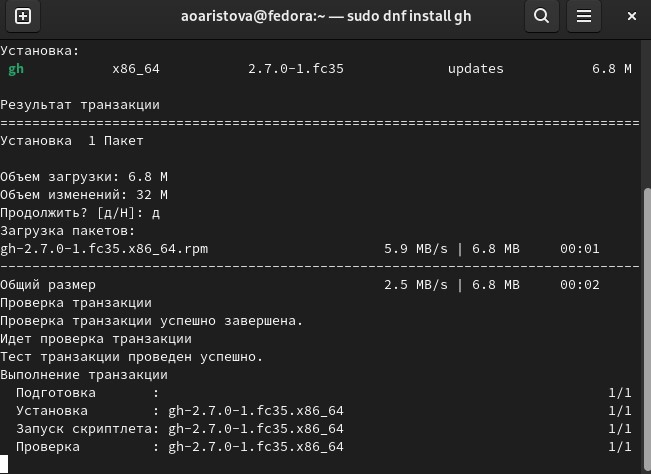


Установка git-flow

Устанавливаю gh в Fedora Linux.

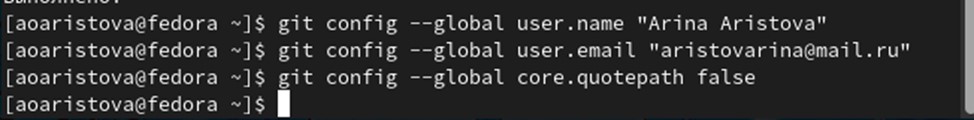


Установка gh



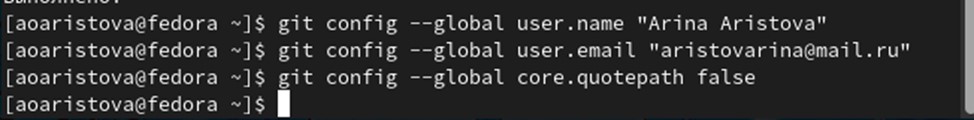
Процесс установки gh

Выполняю базовую настройку git.



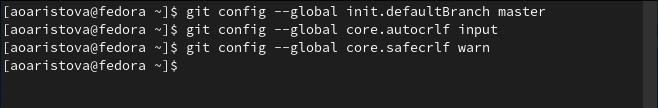
Выполнение базовой настройки git

Настраиваю utf-8 в выводе сообщений git



Настройка utf-8 в выводе сообщений git

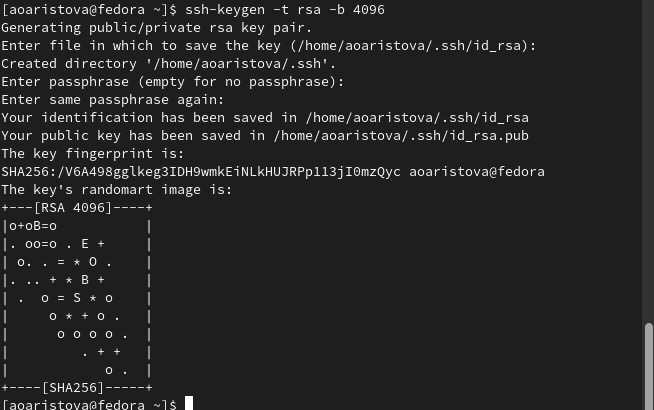
Настраиваю верификацию и подписание коммитов git. Задаю имя начальной ветки master, параметр autocrlf, параметр safecrlf.



Настройка верификацию и подписания коммитов git

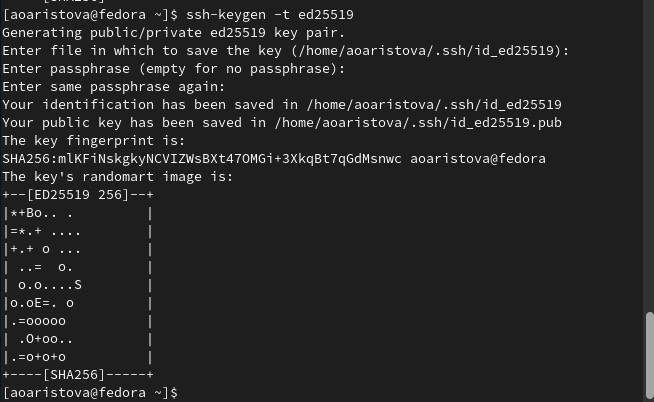
Создаю ключи ssh:

-по алгоритму rsa с ключём размером 4096 бит;



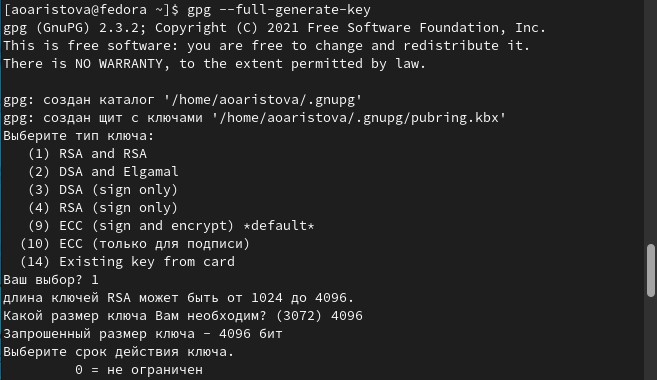
Создание ключа ssh по алгоритму rsa с ключём размером 4096 бит

* по алгоритму ed25519.

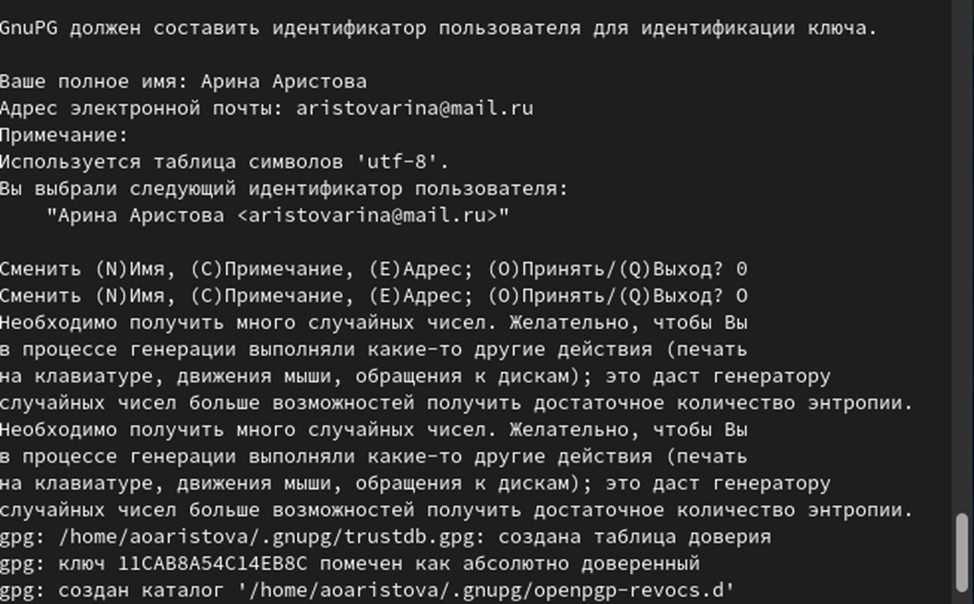


Создание ключа ssh по алгоритму ed25519

Создаю ключ pgp. Генерирую ключ. Из предложенных опций выбираю: тип RSA and RSA; размер 4096; срок действия - 0 (срок действия не истекает никогда), примечание оставляю пустым.

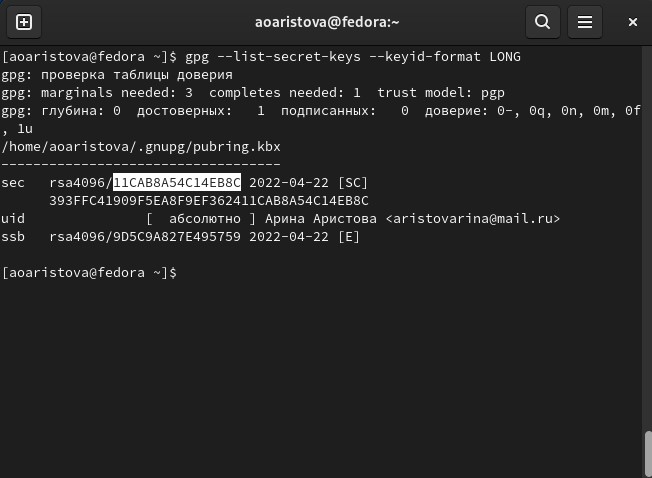


Создание ключа pgp



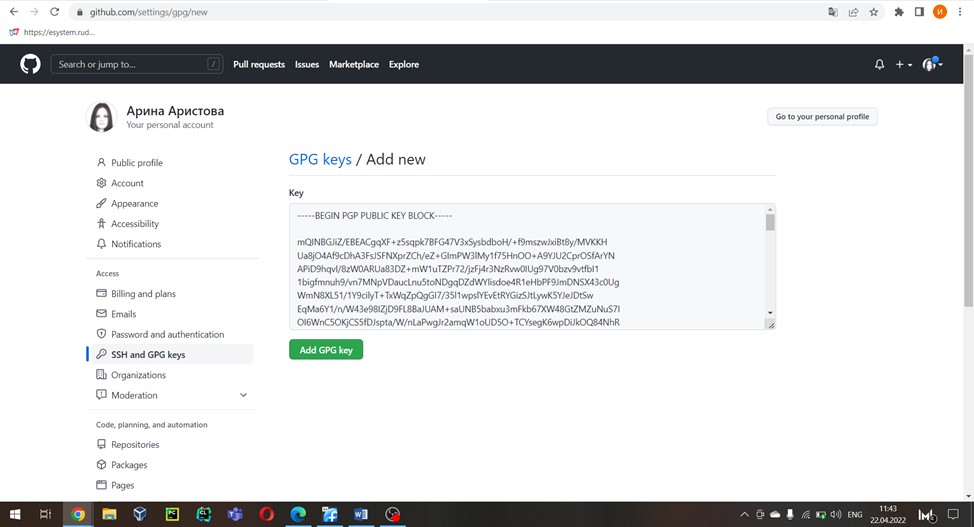
Создание ключа pgp.2

Добавляю PGP ключ в GitHub: Вывожу список ключей и копирую отпечаток приватного ключа.



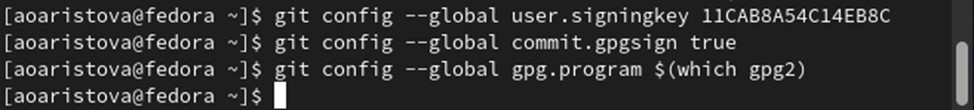
Копирование отпечатка ключа

Копирую сгенерированный PGP ключ в буфер обмена и вставляю полученный ключ в поле ввода в GitHub.



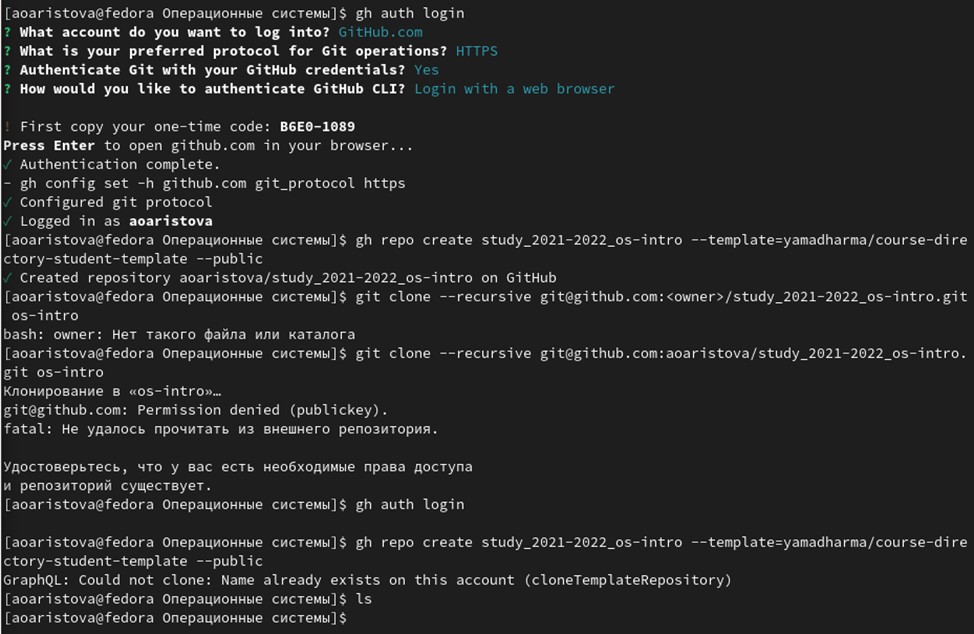
Добавление PGP ключа в GitHub

Настраиваю автоматические подписи коммитов git. Используя введённый email, указываю Git применять его при подписи коммитов.



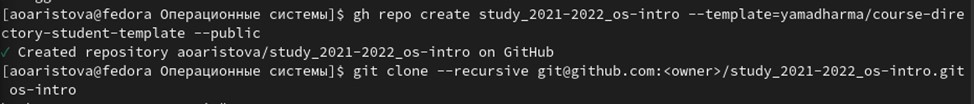
Настройка автоматических подписей коммитов git

Настраиваю gh: Авторизуюсь через браузер.



Авторизация

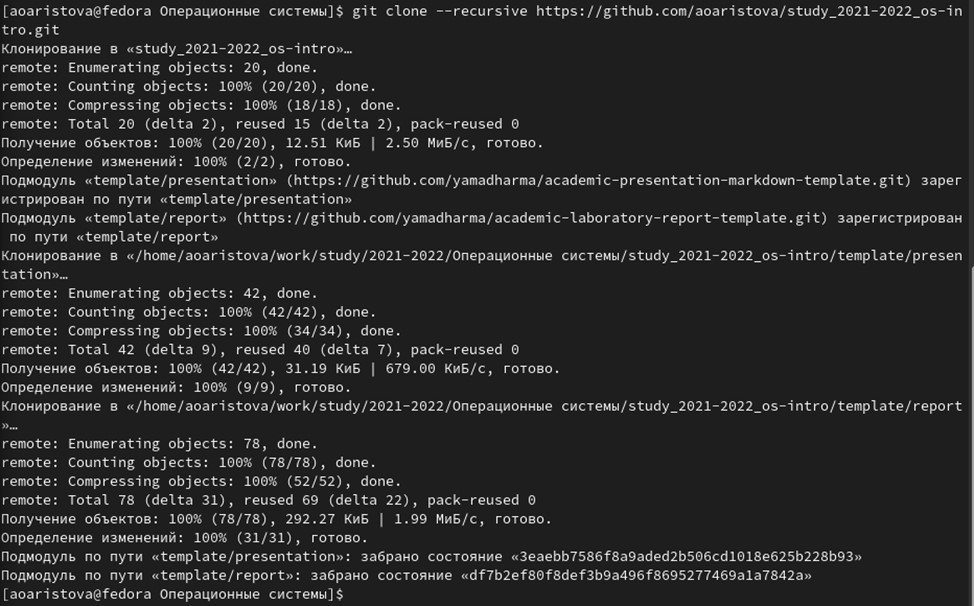
Создаю репозиторий курса на основе шаблона.



Переход в директорию

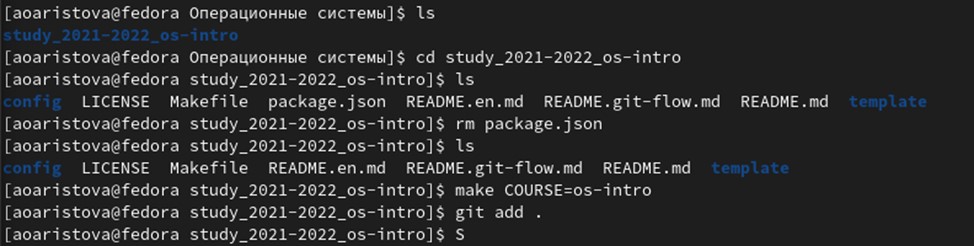
Создание репозитория

Создание репозитория



Клонирование репозитория

Настраиваю каталога курса. Перехожу в каталог курса, удаляю лишние файлы, создаю необходимые каталоги и отправляю файлы на сервер.



Настройка каталога курса

# Вывод

я изучила идеологию применения средств контроля версий, а также освоила умения по работе с git.

# Ответы на контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? VCS – (Version Control System) системы контроля версий. Они 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 – место расположения файлов и папок проекта, изменения в которых отслеживаются. Commit – операция, предполагающая отправку в репозиторий изменений, которые пользователь внес в свою рабочую копию. Рабочая копия - текущее состояние файлов проекта, полученных из хранилища и, возможно, измененных, то есть разработчик имеет в распоряжении именно рабочую копию и с ней работает.
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные VCS имеют одно основное хранилище. Каждый пользователь копирует себе необходимые файлы из основного репозитория, изменяет их, а затем добавляет изменения обратно (Subversion). Децентрализованные VCS устроены так, что каждый пользователь имеет свой (или даже не один) репозиторий. Пользователь может добавлять и забирать изменения из любого репозитория (Git). Соответственно, количеством основных репозиториев и различаются централизованные и децентрализованные VCS.
4. Опишите действия с VCS при единоличной работе с хранилищем. Разработчик работает с веткой master, при необходимости может создать ветки для отдельных частей проекта. При завершении изменений разработчик коммитит (commit) и пушит (push) их, то есть сохраняет изменения в общем хранилище.
5. Опишите порядок работы с общим хранилищем VCS. Каждый разработчик проекта работает над отдельной часть проекта в своей ветке. После завершения изменений разработчик коммитит (commit) и пушит (push) изменения на сервер. После окончания работы необходимо смерджить(merge), то есть выполнить слияние, веток, например, с главной веткой. Также разработчик может работать с изменениями, сделанными другим разработчиком, если на одной ветке их работает несколько.
6. Каковы основные задачи, решаемые инструментальным средством git? Основные задачи git заключаются в удобной командной работе над проектом, а также в хранении информации обо всех изменениях проекта.
7. Назовите и дайте краткую характеристику командам git. git init – создание основного дерева репозитория; git pull - получение обновлений (изменений) текущего дерева из центрального репозитория; git push - отправка всех произведённых изменений локального дерева в центральный репозиторий; git status - просмотр списка изменённых файлов в текущей директории; git diff - просмотр текущих изменений; git add - сохранение текущих изменений: добавить все изменённые и/или созданные файлы и/или каталоги; git add имена\_файлов - сохранение текущих изменений: добавить конкретные изменённые и/или созданные файлы и/или каталоги: 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 ‘Описание коммита’ - сохранение добавленных изменений:сохранить все добавленные изменения и все изменённые файлы; git commit - сохранение добавленных изменений: 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 merge –no-ff имя\_ветки - слияние ветки с текущим деревом; 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.
8. Что такое и зачем могут быть нужны ветви (branches)? Ветка – указатель на один из коммитов. Ветки используются для разработки одной части проекта отдельно от других его частей. Каждая ветка представляет собой отдельную копию кода проекта. Ветки позволяют работать одновременно над разными версиями проекта.
9. Как и зачем можно игнорировать некоторые файлы при commit? Игнорировать файлы при commit можно с помощью файла .gitignor . Туда обычно помещаются файлы, которые не нужны для проекта, например, временные файлы, создаваемые редакторами.