Лабораторная работа № 2

Операционные системы

Ильина Любовь Александровна

Содержание

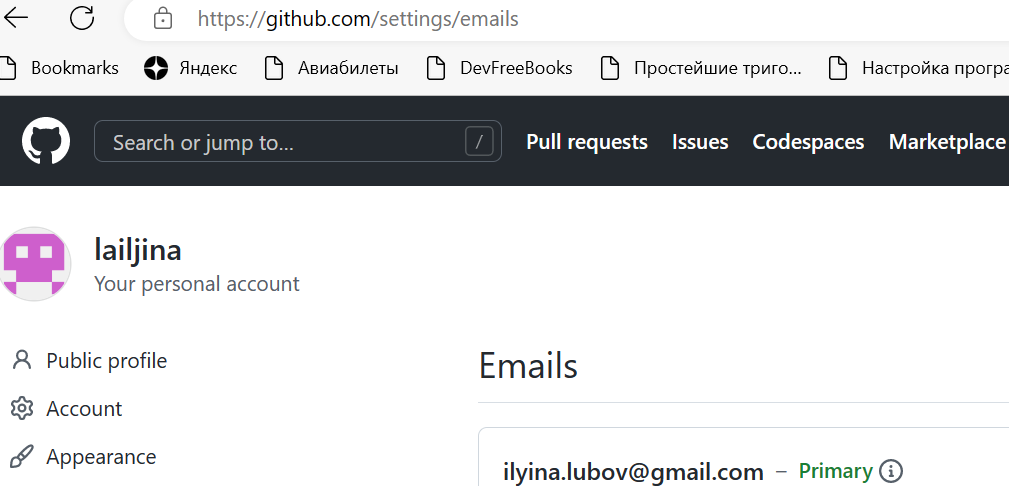
# Цель работы

Цель работы: - изучить идеологию и применение средств контроля версий; - освоить умения по работе с git.

# Задание

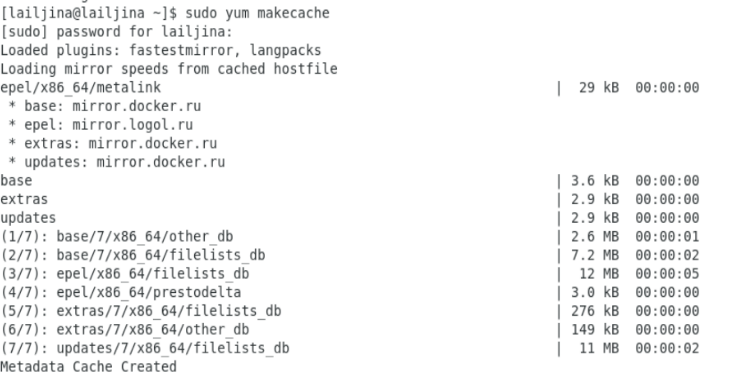
# Выполнение лабораторной работы

1. Создаем учётную запись на https://github.com – с логином lailjina и email: ilyina.lubov@gmail.com – (рис. @fig:001)



Создана учетная запись на https://github.com

2.Установим программное обеспечение после обновления $ yum makecache (рис. @fig:002) с помощью команд $ yum install epel-release (рис. @fig:003) и $ yum -y install gitflow (рис. @fig:004)



Обновление базы makecache

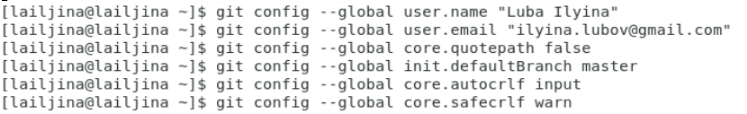


Установка epel-release



Установка gitflow

1. Базовая настройка git (рис. @fig:006)



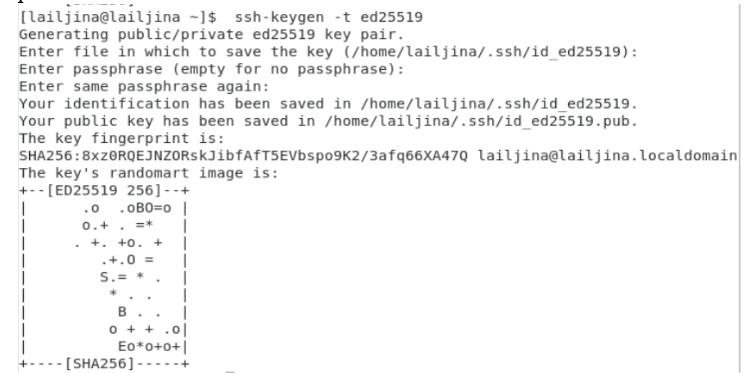
Настройка git

1. Создаем ключ SSH с помощью $ ssh-keygen -t rsa -b 4096 (рис. @fig:006)



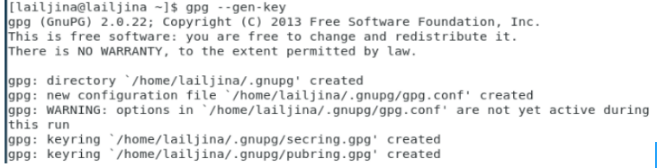
Создание ключа SSH

1. Cоздаем SSH ключ по алгоритму ed25519 с помощью команды $ ssh-keygen -t ed2551 (рис. @fig:007)



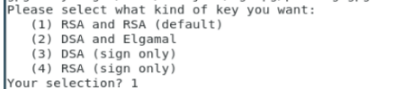
Создание ключа SSH по алгоритму ed25519

1. Генерируем ключ pgp с помощью команды $ gpg –gen-key (рис. @fig:008)



Генерируем ключ pgp

6.1. Выбрали тип ключа: RSA and RSA (рис. @fig:009)



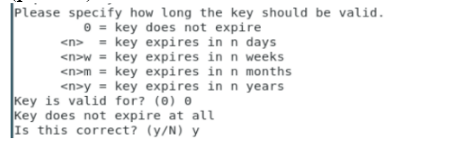
Выбор типа RSA and RSA

6.2. Выбрали размер ключа 4096 (рис. @fig:010)

Выбор размера ключа 4096

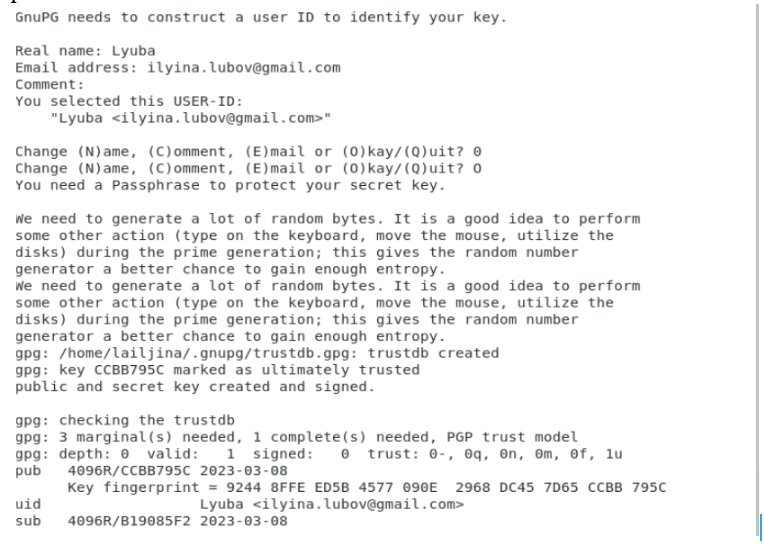
Выбор размера ключа 4096

6.3. Выбрали срок действия; значение по умолчанию — 0 (срок действия не истекает никогда) (рис. @fig:011)



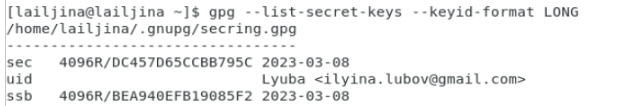
Выбор срока действия ключа

6.4. Введем личную информацию, которую GPG запросит и которая сохранится в ключе (рис. @fig:012):  
– Имя (не менее 5 символов).  
– Адрес электронной почты (соответствует на GitHub).



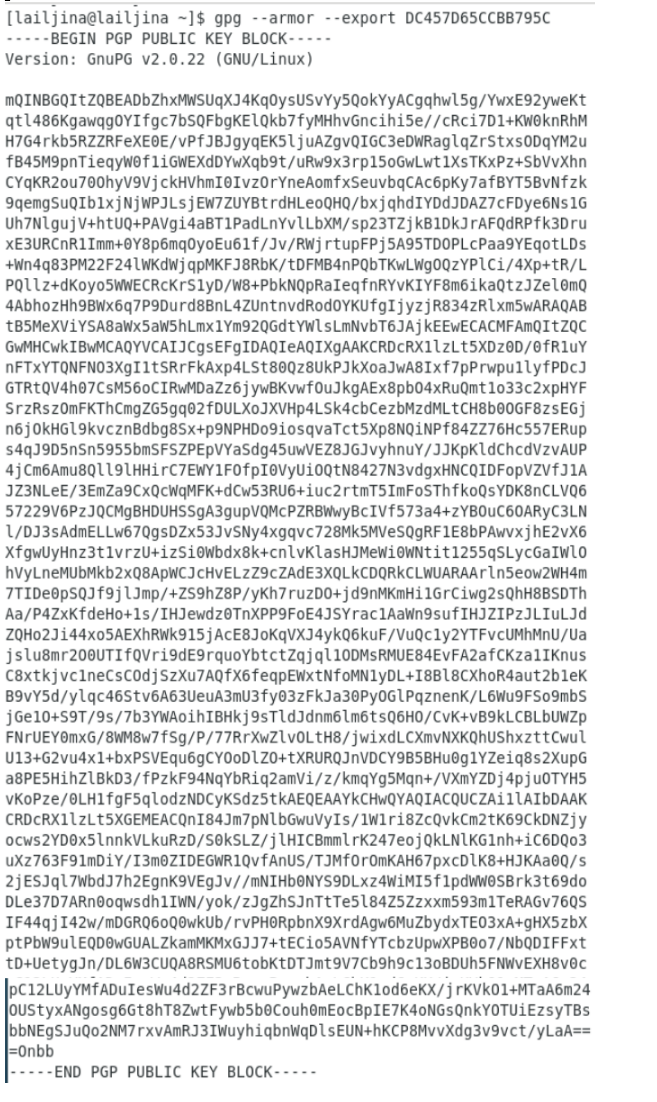
Введем личную информацию владельца ключа

1. Генерируем отпечатка ключа PGP с помощью $ gpg –list-secret-keys –keyid-format LONG (рис. @fig:013)



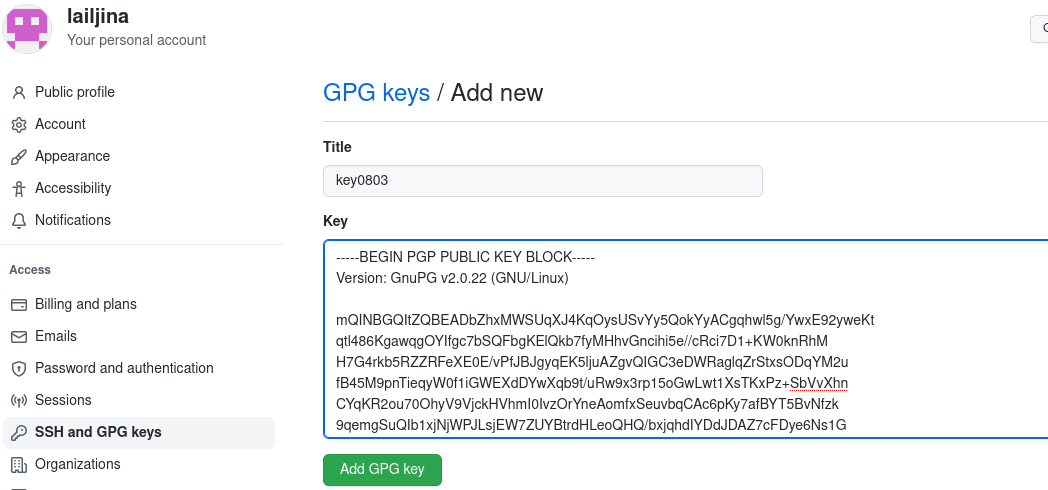
Генерация отпечатка ключа PGP

1. Получим ключ для загрузки на github c $ gpg –armor –export (рис. @fig:014)



Получение ключа для загрузки на github

1. Подгрузим ключ pgp на https://github.com/settings/keys (рис. @fig:015)



Подгрузка ключа PGP на github.com

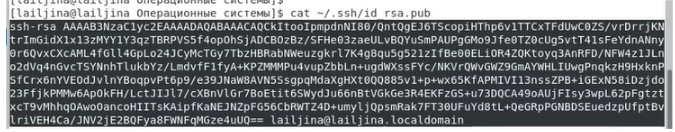
1. Используя введёный email, укажем Git применять его при подписи коммитов (рис. @fig:016)

Настройка подписи коммитов

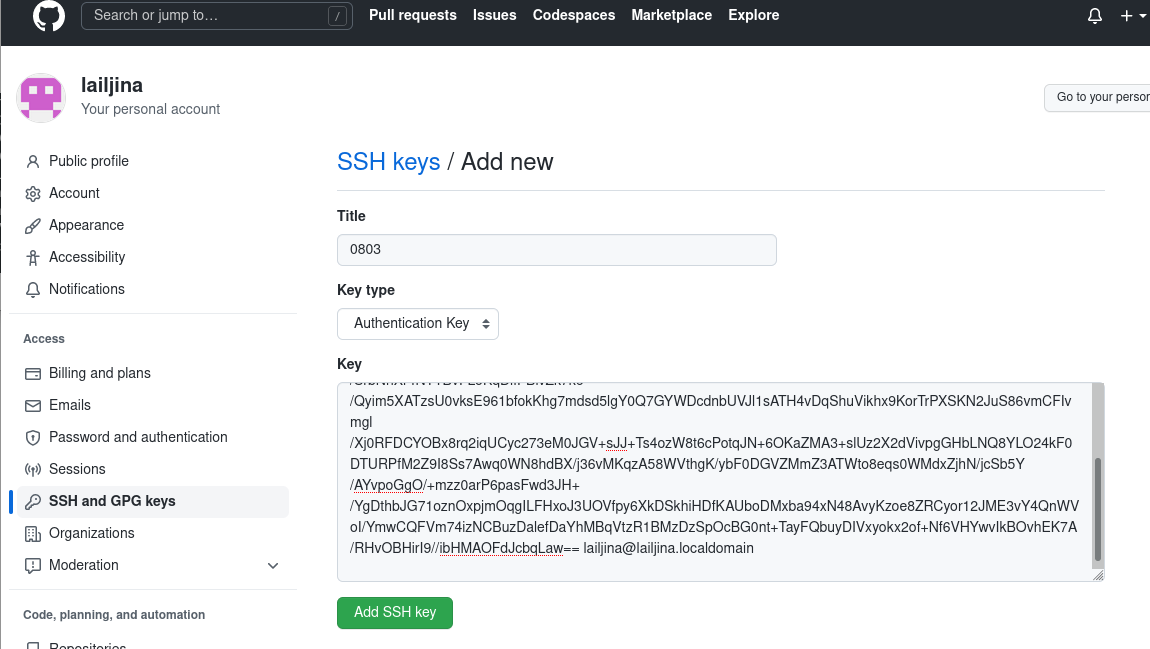
Настройка подписи коммитов

1. Создадим репозиторий курса на основе шаблона

11.1. Сначала выгрузим SSH ключ с $ cat ~/.ssh/id\_rsa.pub (рис. @fig:017) и загрузим на https://github.com/ (рис. @fig:018)

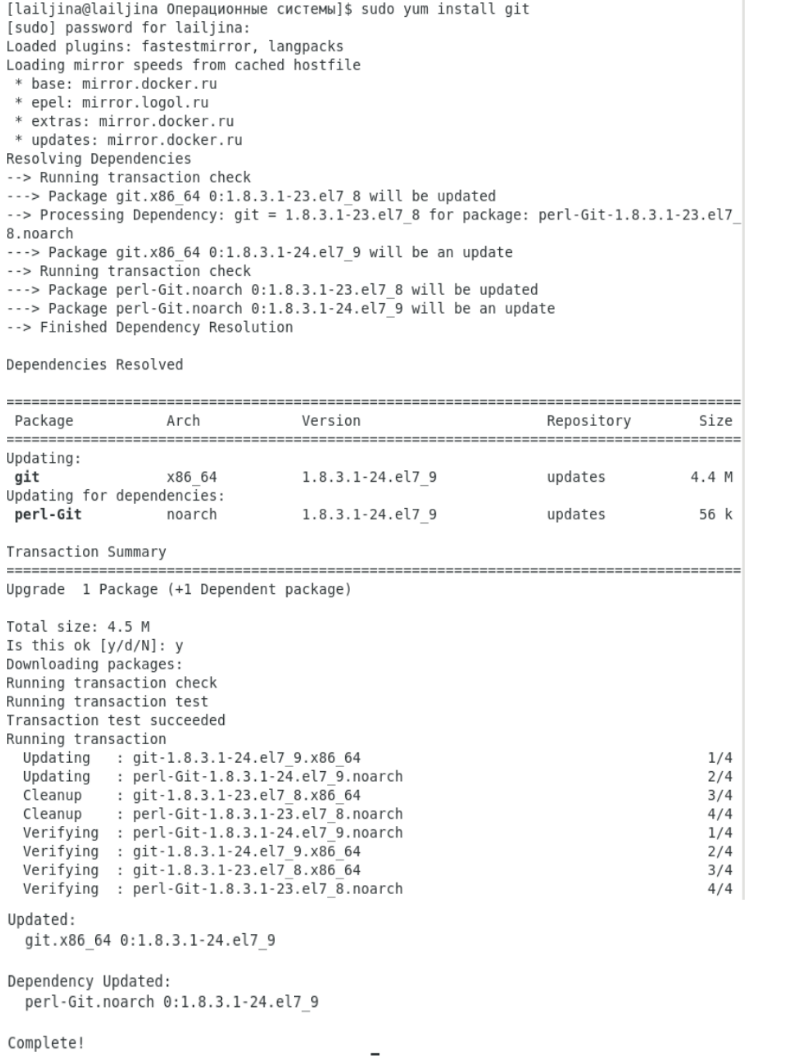


Получение отпечатка SSH ключа для загрузки на github



Загрузка SSH ключа для загрузки на github

11.2. Установим $ yum install git (рис. @fig:019)

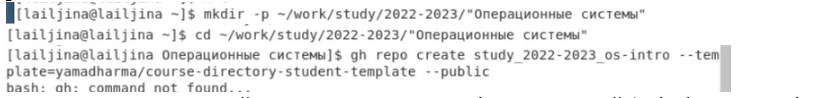


Установка git

11.3. Создаем папки курса (рис. @fig:020) с помощью: $ mkdir -p ~/work/study/2021-2022/“Операционные системы”

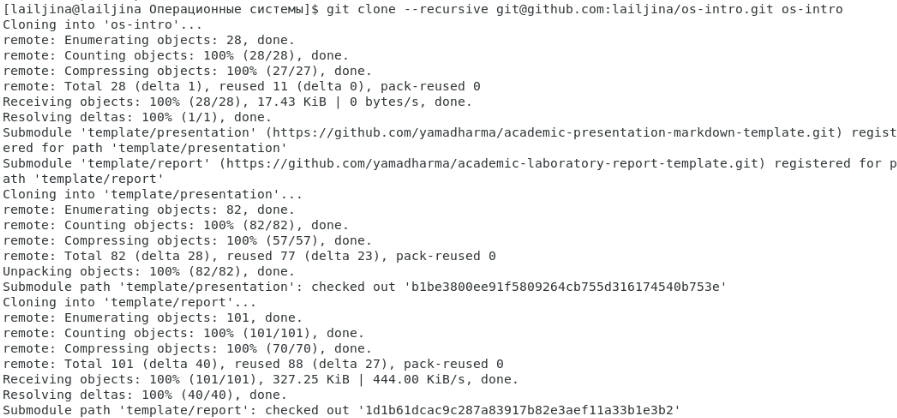
$ cd ~/work/study/2021-2022/“Операционные системы”

$ gh repo create study\_2021-2022\_os-intro –template=yamadharma/course-directory-student-template –public



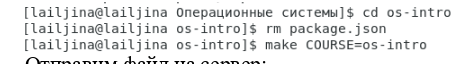
Создание папки курса

11.4. Клонируем репозиторий с созданием папки os-intro командой $ git clone –recursive git@github.com:lailjina/os-intro.git os-intro (рис. @fig:021)



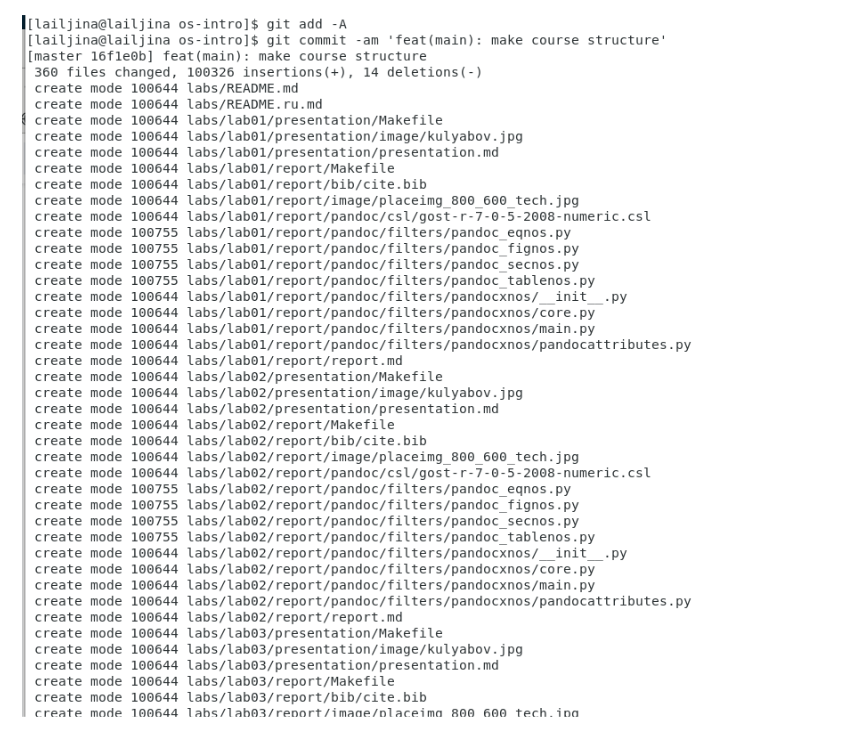
Клонируем репозиторий с созданием папки os-intro

1. Настроим каталог курса (рис. @fig:022): заходим в директорию(cd os-intro), удаляем файлы ($ rm package.json) и создаем файлы (make COURSE=os-intro).



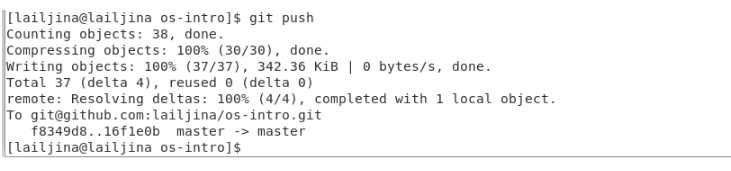
Настройка каталога курса

1. Отправим файлы на сервер: $ git add -A. и задаем структуру курса $ git commit -am ‘feat(main): make course structure’ (рис. @fig:023)



Отправка файлов на сервер и настройка структуры курса

1. Изменения синхронизируем с репозиторием гитхаб с помощью git push(рис. @fig:024)



Синхронизация с репозиторием гитхаб

# Выводы

Вывод: Мной изучена идеология и применение средств контроля версий, а также приобретены навык работы с git.

Ответы на контрольные вопросы:

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

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом.

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

Хранилище (repository), или репозитарий, — место хранения файлов и их версий, служебной информации.

Хранилище (repository), или репозитарий, — место хранения файлов и их версий, служебной информации.

История - состояние всего хранилища или отдельных файлов в момент времени.

Рабочая копия (working copy) — текущее состояние файлов проекта (любой версии), полученных из хранилища и, возможно, измененных.

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

В централизованных СКВ все файлы хранятся в одном репозитории, и каждый пользователь может вносить изменения. В децентрализованных их несколько, и они могут обмениваться изменениями между собой, а центрального репозитория может не существовать вообще. Среди классических (т.е. централизованных) VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial.

1. Опишите действия с VCS при единоличной работе с хранилищем.

Получить нужную версию проекта (рабочую копию), внести в неё необходимые изменения, сделать нужный коммит, создав при этом новую версию проекта (старые не удаляются).

1. Опишите порядок работы с общим хранилищем VCS.

Аналогично единоличной работе, но также можно объединить внесённые разными пользователями изменения, отменить изменения или заблокировать некоторые файлы для изменения, обеспечив привилегированный доступ конкретному разработчику.

1. Каковы основные задачи, решаемые инструментальным средством git?

Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями. Git позволяет создавать локальные репозитории и вносить в них изменения, а также работать с удалёнными репозиториями.

1. Назовите и дайте краткую характеристику командам git.

git. 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 изменения: git diff 6)сохранение текущих изменений: а)добавить все изменённые и/или созданные файлы и/или каталоги: git add . б)добавить конкретные изменённые и/или созданные файлы и/или каталоги: git add имена\_файлов в)удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): git rm имена\_файлов 7)сохранение добавленных изменений: а)сохранить все добавленные изменения и все изменённые файлы: git commit -am ‘Описание коммита’ б)сохранить добавленные изменения с внесением комментария через встроенный редактор: git commit 8)создание новой ветки, базирующейся на текущей: git checkout -b имя\_ветки 9)переключение на некоторую ветку: git checkout имя\_ветки (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой) 10)отправка изменений конкретной ветки в центральный репозиторий: git push origin имя\_ветки 11)слияние ветки с текущим деревом: git merge –no-ff имя\_ветки 12)удаление ветки: а)удаление локальной уже слитой с основным деревом ветки: git branch -d имя\_ветки б)принудительное удаление локальной ветки: git branch -D имя\_ветки в)удаление ветки с центрального репозитория: git push origin :имя\_ветки

1. Приведите примеры использования при работе с локальным и удалённым репозиториями.

Допустим, нужно добавить в проект новый файл file.txt Загрузим нужную версию из удалённого репозитория: git checkout last (last – имя нужной нам ветки) Добавим файл в локальный репозиторий: git add file.txt (файл лежит в том же каталоге, что и репозиторий) Сохраним изменения: git commit –am “file.txt was added” Отправим изменения в удалённый репозиторий: git push

1. Что такое и зачем могут быть нужны ветви (branches)?

Допустим, нужно добавить в проект новый файл file.txt Загрузим нужную версию из удалённого репозитория: git checkout last (last – имя нужной нам ветки) Добавим файл в локальный репозиторий: git add file.txt (файл лежит в том же каталоге, что и репозиторий) Сохраним изменения: git commit –am “file.txt was added” Отправим изменения в удалённый репозиторий: git push

1. Как и зачем можно игнорировать некоторые файлы при commit

Допустим, нужно добавить в проект новый файл file.txt Загрузим нужную версию из удалённого репозитория: git checkout last (last – имя нужной нам ветки) Добавим файл в локальный репозиторий: git add file.txt (файл лежит в том же каталоге, что и репозиторий) Сохраним изменения: git commit –am “file.txt was added” Отправим изменения в удалённый репозиторий: git push