Отчёт по лабораторной работе №2

дисциплина: Операционные системы

Латаева Гюзелия Андреевна

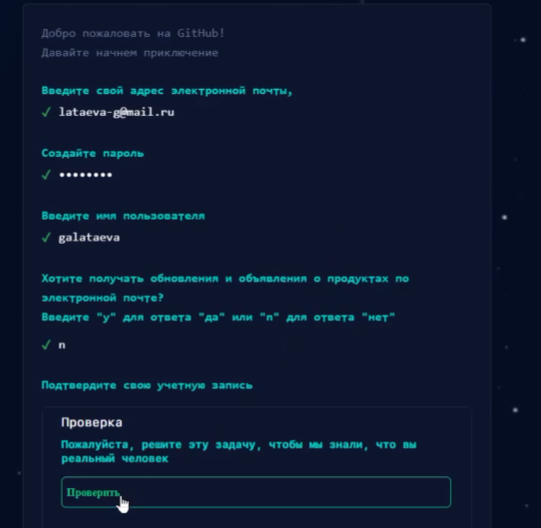
Содержание

# 1 Цель работы

Изучить идеологию и применение средств контроля версий. Освоить умения по работе с git.

# 2 Последовательность выполнения работы

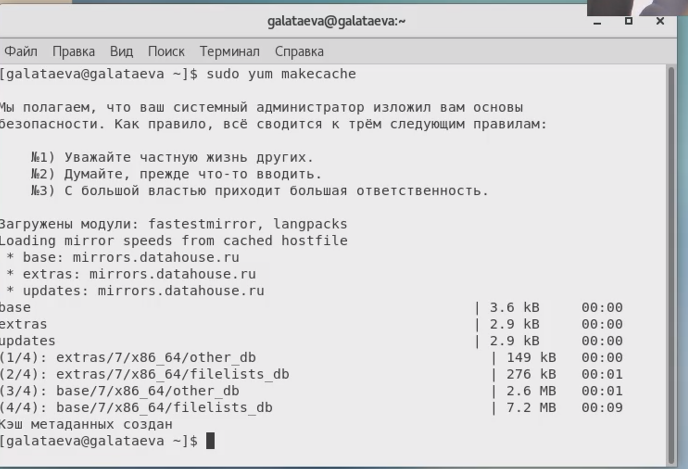
**1) Создаю учётную запись на https://github.com и заполняю даные.**



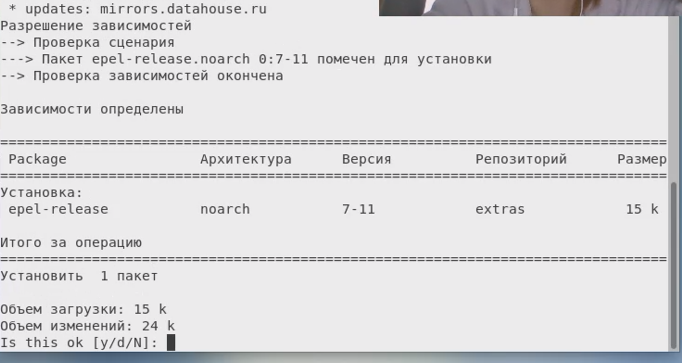
Регистрация

**2) Установка программного обеспечения**

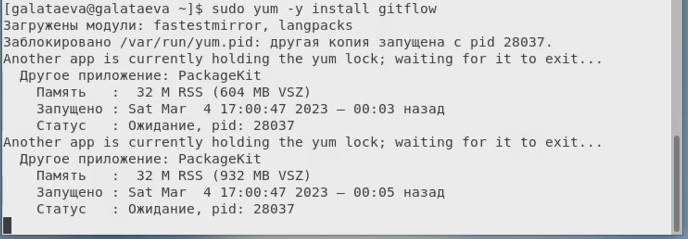
* Установка git-flow: сначала необходимо установить makecache а потом уже сам gitflow



Установка makecache

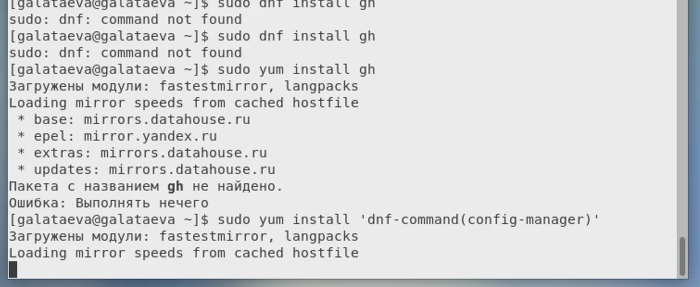


Установка epel\_release

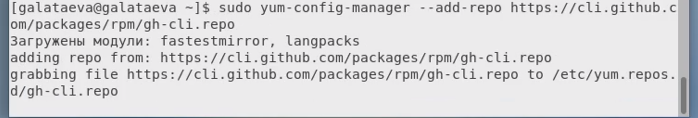


Установка gitflow

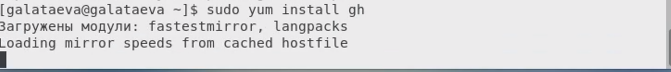
* Установка gh: на Centos 7 необходимо провести ряд манипуляций перед тем, как установить gh



Установка config-manager



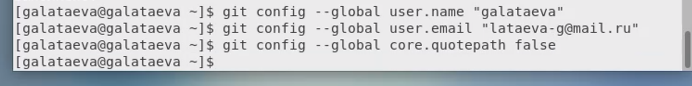
Установка репозитория с gh



Установка gh

**3) Базовая настройка git**

Тут мы задали имя пользователя, почту и настроили utf-8 в выводе сообщений



Настройка git

**4) Зададим имя начальной ветки:**

Начальная ветка master

Начальная ветка master

Параметры autocrlf и safecrlf

Параметры autocrlf и safecrlf

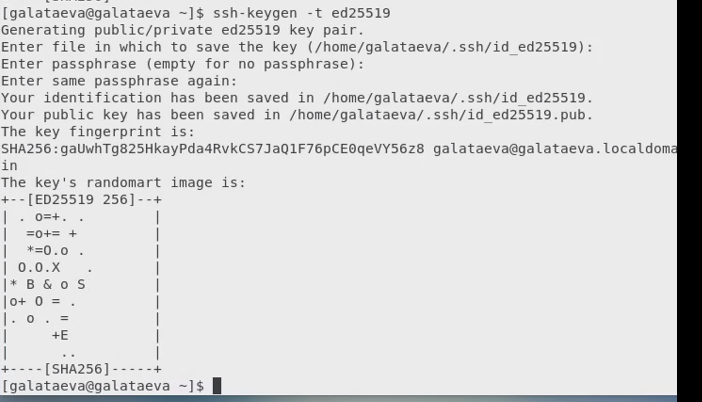
**5) Создаем ключи ssh**

* по алгоритму rsa с ключём размером 4096 бит:



Алгоритм rsa

* по алгоритму ed25519:



Алгоритм ed25519

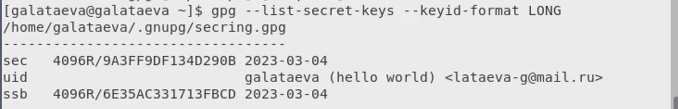
**6) Создаем ключи pgp**



Создание ключа

**7) Добавляем PGP ключ в GitHub**

* Выводим список ключей:



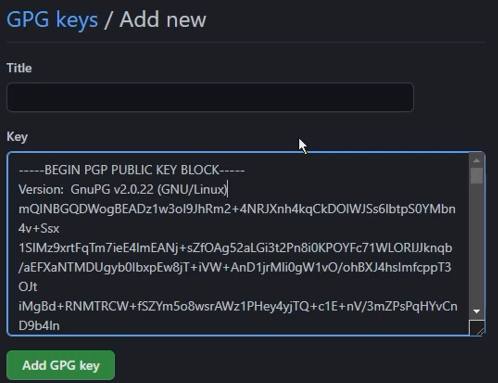
Список ключей

* Выводим ключ:



Список ключей

* Вставляем полученный ключ на сайте:



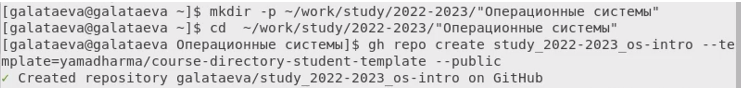
Добаление ключа GitHub

**8) Настраиваем автоматические подписи коммитов git**

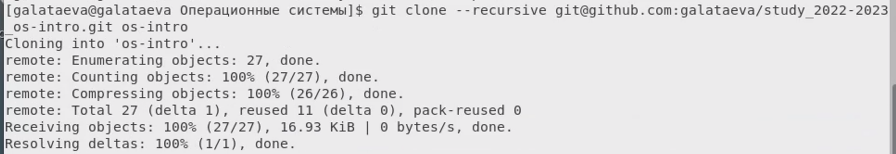
Настройка подписей

Настройка подписей

**9) Создаем репозиториий курса на основе шаблона**



Создание репозитория



Копирование репозитория по шаблону

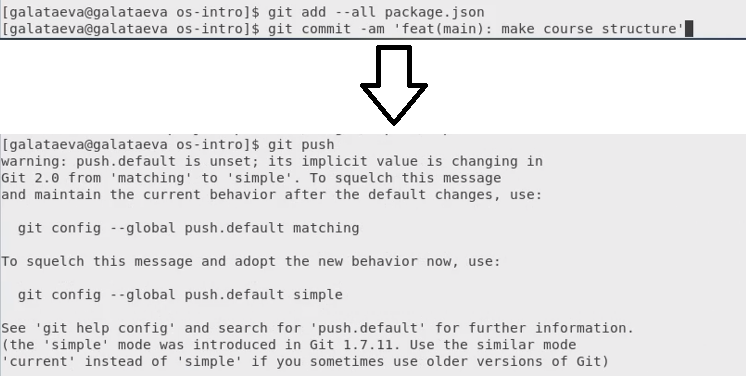
**10) Настраиваем каталог курса**

Переходим в каталог курса

Переходим в каталог курса

Удаление лишних фалов и создание каталогов

Удаление лишних фалов и создание каталогов



Отправка даных на сервер

# 3 Заключение

Я изучила применение средств контроля версий и в какой-то мере освоила навыки по работе с git.

# 4 Контрольные вопросы

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

Система контроля версий (от англ. Version Control System, VCS) — это, грубо говоря, место хранения кода. Она создана для разработки продуктов: на хранение кода, синхронизацию работы нескольких человек, создание релизов и т.д.

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

Хранилище (репозиторий) – это хранилище всех версий кода. Он бывает трех видов: Локальный — расположен на одном компьютере, и работать с ним может только один человек. Централизованный — расположен на сервере, куда имеют доступ сразу несколько программистов. Распределенный — самый удобный вариант с облачным хранилищем. Главный репозиторий хранится в облаке, а его локальные копии — у разработчиков на компьютерах.

Commit - запись изменений.

История - список предыдущих изменений.

Рабочая копия – текущее состояние файлов проекта, основанное на версии из хранилища (обычно на последней).

Рабочую копию получают из репозитория. Изменения вносятся в рабочую версию, потом посредством коммитов они заносятся в хранилище. История позволяет просмотреть изменения, которые вносились в репозиторий.

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

См. ответ на вопрос 1.

Среди централизованных VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial.

*4. Опишите действия с VCS при единоличной работе с хранилищем.*

1. Получение нужной рабочей копии
2. Внесение в неё необходых изменений
3. Сделать нужный коммит.

*5. Опишите порядок работы с общим хранилищем VCS.*

См. ответ на вопрос 4 + можно объединить/отменить внесённые другими пользователями изменения, или заблокировать некоторые файлы для изменения ими.

*6. Каковы основные задачи, решаемые инструментальным средством git?*

У Git две основных задачи: первая — хранить информацию о всех изменениях в коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.

*7. Назовите и дайте краткую характеристику командам git.*

git init - инициализация репозитория

git status - проверка статуса репозитория

git pull - получение обновлений (изменений)

git push - отправка изменений

git diff - просмотр текущих изменений

git add . -добавить все изменения

git add имена\_файлов - добавить конкретные изменения

git rm имена\_файлов - удалить файл и/или каталог из репозитория

git commit -am ‘Описание коммита’ - сохранение всех добавленных изменений

git log –p - просмотр истории коммитов с изменениями

git branch - просмотр списка веток

git branch -d имя\_ветки - удаление ветки

git branch имя\_ветки - создание новой ветки

git merge имя\_ветки – слияние ветки с основной

git merge origin - слияние удалённого репозитория с локальным

git push -u origin имя\_ветки - отправка новой ветки в удалённый репозиторий

*8. Приведите примеры использования при работе с локальным и удалённым репозиториями.*

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

Система контроля версий может поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Это удобно при работе над одним проектом нескольких человек.

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

Игнорируемые файлы отслеживаются в специальном файле .gitignore, который регистрируется в корневом каталоге репозитория. В Git нет специальной команды для указания игнорируемых файлов: вместо этого необходимо вручную отредактировать файл .gitignore, чтобы указать в нем новые файлы, которые должны быть проигнорированы. Файлы .gitignore содержат шаблоны, которые сопоставляются с именами файлов в репозитории для определения необходимости игнорировать эти файлы. Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты.