Отчет по лабораторной работе 2

Первоначальная настройка git

Власов Артем Сергеевич

Содержание

# 1 Цель работы

Освоить систему контроля версий и научиться работать с git.

# 2 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.

# 3 Выполнение лабораторной работы 2.

Сформируем отчет лабораторной работы номер 2.

Делаем предварительную конфигурацию git. (рис. fig. 1).

Задаем имя и email репозитория

Рис. 1: Задаем имя и email репозитория

Настраиваем utf-8 в выводе сообщения git. (рис. fig. 2).

Настраиваем utf-8

Рис. 2: Настраиваем utf-8

Задаем имя начальной ветки. (рис. fig. 3).

Задаем имя начальной ветки, как master

Рис. 3: Задаем имя начальной ветки, как master

Устанавливаем настройку autocrlf

Рис. 4: Устанавливаем настройку autocrlf

Устанавливаем параметр safecrlf

Рис. 5: Устанавливаем параметр safecrlf

Создаем SSH ключ (рис. fig. 6).

Генерируем пару ключей командой keygen

Рис. 6: Генерируем пару ключей командой keygen

Копируем ключ из локальной консоли в буфер обмена

Рис. 7: Копируем ключ из локальной консоли в буфер обмена

Заходим в свой аккаунт на сайте github. Переходим в настройки, SSH ключи. (рис. fig. 8).

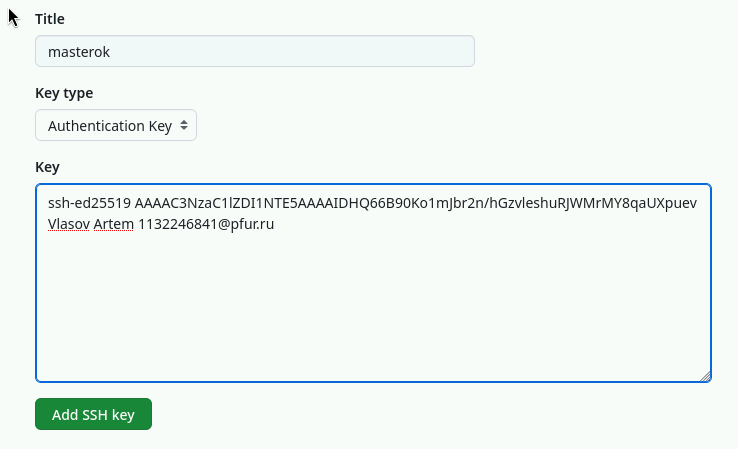


Рис. 8: Вставляем ключ и сохраняем

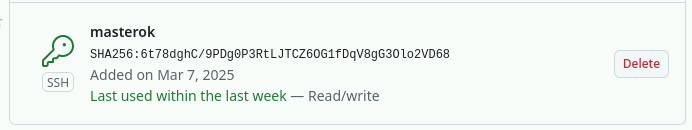


Рис. 9: Проверяем добавление ключа

Открываем терминал и создаем каталоги для предмета “Архитектура компьютера” (рис. fig. 10).

Создаем каталоги последовательно

Рис. 10: Создаем каталоги последовательно

Переходим на страницу репозитория с шаблоном (рис. fig. 11).

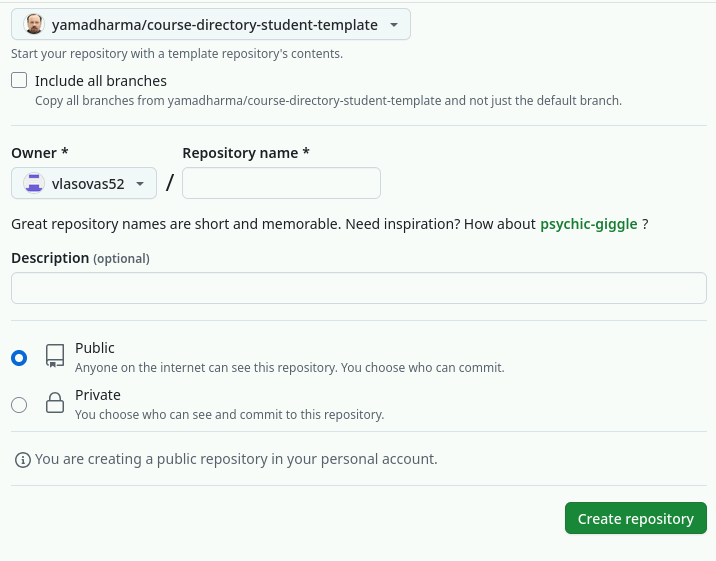


Рис. 11: Создаем репозиторий по шаблону

Переходим в папку с предметом (рис. fig. 12).

Переходим в каталог курса

Рис. 12: Переходим в каталог курса

Клонируем созданный репозиторий

Рис. 13: Клонируем созданный репозиторий

Переходим в каталог arch-pc (рис. fig. 14).

Переходим в нужный каталог

Рис. 14: Переходим в нужный каталог

Удаляем лишние файлы

Рис. 15: Удаляем лишние файлы

Создаем папки по образцу (рис. fig. 16).

Создаем необходимые каталоги

Рис. 16: Создаем необходимые каталоги

Отправляем файлы на сервер (рис. fig. 17).

Отправляем файлы на git

Рис. 17: Отправляем файлы на git

Отправляем файлы на git

Рис. 18: Отправляем файлы на git

Отправляем файлы на git

Рис. 19: Отправляем файлы на git

# 4 Ответы на контрольные вопросы

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

* Системы контроля версий (VCS) — это инструменты для управления изменениями в файлах. Они решают задачи:
  + Хранения истории изменений.
  + Совместной работы над проектами.
  + Ветвления и слияния кода.
  + Отслеживания изменений и их авторов.

1. **Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.**
   * **Хранилище (репозиторий)**: База данных, где хранятся все версии файлов и их история.
   * **Commit**: Фиксация изменений в репозитории. Каждый коммит сохраняет изменения и имеет уникальный идентификатор.
   * **История**: Последовательность коммитов, показывающая, как изменялись файлы.
   * **Рабочая копия**: Текущие файлы, с которыми работает разработчик, извлечённые из репозитория.
2. **Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.**
   * **Централизованные VCS**: Один сервер хранит всю историю. Разработчики работают с локальными копиями, но для фиксации изменений требуется подключение к серверу. Пример: **SVN (Subversion)**.
   * **Децентрализованные VCS**: Каждый разработчик имеет полную копию репозитория, включая всю историю. Примеры: **Git**, **Mercurial**.
3. **Опишите действия с VCS при единоличной работе с хранилищем.**
   * Создать репозиторий: git init.
   * Добавить файлы в индекс: git add <файл>.
   * Зафиксировать изменения: git commit -m "Сообщение".
   * Просматривать историю: git log.
4. **Опишите порядок работы с общим хранилищем VCS.**
   * Клонировать репозиторий: git clone.
   * Создать ветку для работы: git branch <имя\_ветки>.
   * Переключиться на ветку: git checkout <имя\_ветки>.
   * Зафиксировать изменения: git commit -m "Сообщение".
   * Отправить изменения на сервер: git push.
   * Получить изменения с сервера: git pull.
5. **Каковы основные задачи, решаемые инструментальным средством Git?**
   * Управление версиями файлов.
   * Ветвление и слияние кода.
   * Совместная работа над проектами.
   * Отслеживание изменений и их авторов.
6. **Назовите и дайте краткую характеристику командам Git.**
   * git init: Создать новый репозиторий.
   * git add: Добавить файлы в индекс для последующего коммита.
   * git commit: Зафиксировать изменения в репозитории.
   * git push: Отправить изменения в удалённый репозиторий.
   * git pull: Получить изменения из удалённого репозитория.
   * git branch: Управление ветками (создание, удаление, просмотр).
   * git checkout: Переключение между ветками или коммитами.
   * git merge: Слияние веток.
   * git log: Просмотр истории коммитов.
7. **Приведите примеры использования при работе с локальным и удалённым репозиториями.**
   * Локальный репозиторий:
   * git init  
     git add .  
     git commit -m "Initial commit"
   * Удалённый репозиторий:
   * git clone <URL>  
     git push origin main  
     git pull origin main
8. **Что такое и зачем могут быть нужны ветви (branches)?**

* Ветви (branches) — это отдельные линии разработки в репозитории. Они нужны для:
  + Параллельной работы над разными задачами.
  + Изоляции экспериментальных изменений.
  + Упрощения слияния изменений после завершения работы.

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

* Для игнорирования файлов используется файл .gitignore. В него добавляются шаблоны файлов или папок, которые не должны отслеживаться Git. Это полезно для исключения временных файлов, бинарных данных или конфиденциальной информации.

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

* (Повтор вопроса 9 для полноты.) Ветви (branches) — это отдельные линии разработки в репозитории. Они нужны для:
  + Параллельной работы над разными задачами.
  + Изоляции экспериментальных изменений.
  + Упрощения слияния изменений после завершения работы.

# 5 Выводы

Мы освоили систему контроля версий и получили превичные навыки работы с git. Создали ключи, связали локальные репозитории.

# Список литературы