ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

дисциплина: Архитектура компьютера

Газизянов Владислав Альбертович

Содержание

Список иллюстраций

1 Цель работы

Изучить и понять принципы работы системы контроля версий Git. - Освоение основных команд Git, таких как init, add, commit, push, pull, clone, branch, checkout и других. - Понимание, как Git отслеживает изменения в файлах и как он хранит эти изменения. - Изучение ветвления и слияния в Git, а также способов решения конфликтов при слиянии. - Практическое применение Git для управления версиями проекта, включая работу с удаленными репозиториями. В конечном итоге, целью является развитие навыков, которые позволят эффективно использовать Git для управления версиями кода.

2 Задание

Задание данной работы является освоение идеологии и использование инструментов контроля версий. Также получение практического опыта в работе с системой git.

3 Теоретическое введение

3.1. Системы контроля версий.

Общие понятия 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 дельта-компрессию — 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.

3.2. Система контроля версий 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. Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией 3.3. Основные команды git

Команда	Описание
`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 изменения
`git add .`	добавить все изменённые и/или созданные файлы и/или каталоги
`git add имена_файлов`	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)

Рис. 3.1: table

`git commit -am 'Описание коммита'`	сохранить все добавленные изменения и все изменённые файлы
`git checkout -b имя_ветки`	создание новой ветки, базирующейся на текущей
`git checkout имя_ветки`	переключение на некоторую ветку (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой)
`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

Рис. 3.2: table2

4 Выполнение лабораторной работы

4.1. Настройка GitHub Создаём учётную запись на сайте https://github.com/ и заполняем основные данные.(рис.1)

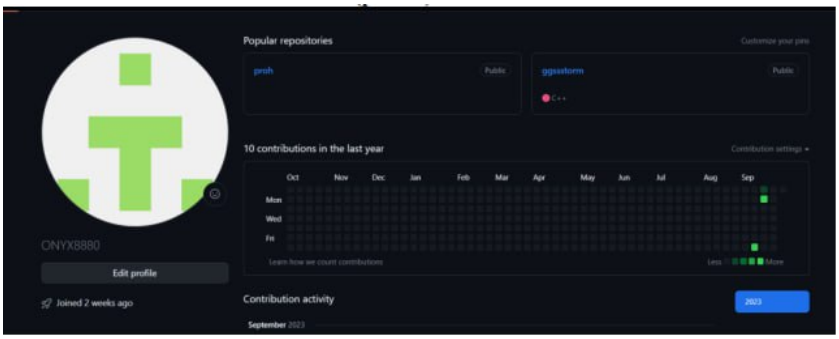


Рис. 4.1: GitHub

4.2 Базовая настройка git. Начнем с базовой настройки git. Запустим терминал и выполним следующие команды(

git config –global user.name “” git config –global user.email “”

), указав имя и электронную почту владельца репозитория(рис.2)

```
vagazizyanov@ONY-X:~$ git config --global user.name ONYX8880
vagazizyanov@ONY-X:~$ git config --global user.email ony.resset@gmail.com
```

Рис. 4.2: Настройка git

Настроим utf-8 в выводе сообщений git: (Рис.3)

```
vagazizyanov@ONY-X:~$ git config --global core.quotepath false
```

Рис. 4.3: Utf-8

Зададим имя начальной ветки (будем называть её master): (Рис.4)

```
vagazizyanov@ONY-X:~$ git config --global init.defaultBranch mas
```

Рис. 4.4: Начальная ветка

Параметр autocrlf: (рис.5)

```
vagazizyanov@ONY-X:~$ git config --global core.autocrlf input
```

Рис. 4.5: Параметр autocrlf

Параметр safecrlf: (Рис.6)

```
vagazizyanov@ONY-X:~$ git config --global core.safecrlf warn
```

Рис. 4.6: Параметр safecrlf

4.3. Создание SSH ключа генерируем пару ключей (приватный и открытый): 9 (Рис.7)

```
vagazizyanov@ONY-X:~$ ssh-keygen -C "GazizynovVladislav ony.resset@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/vagazizyanov/.ssh/id_rsa):
Created directory '/home/vagazizyanov/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match.  Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/vagazizyanov/.ssh/id_rsa
Your public key has been saved in /home/vagazizyanov/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:/YVfCXSMM2iP2igpNxwYFngldkrbNS7MKlb1CIcceZc GazizynovVladislav ony.resset@gmai
The key's randomart image is:
+---[RSA 3072]----+
|     oB*= o. ..o. |
|    .oB%.=E.o.+.. |
|     o++*.o. o.o  |
|     ..... . o. .|
|    o .. S = . ...|
|   . .. * o o o . |
|       o o   . .  |
|                  |
|                  |
+----[SHA256]-----+
```

Рис. 4.7: Генерация ключей

Смотрим сгенерированный публичный ключ командой cat ~/.ssh/id_rsa.pub (Рис.8)

Публичный ключ

Скопировав из локальной консоли ключ в буфер обмена вставляем ключ в появившееся на сайте поле и указываем для ключа имя (Title).(Рис.9)

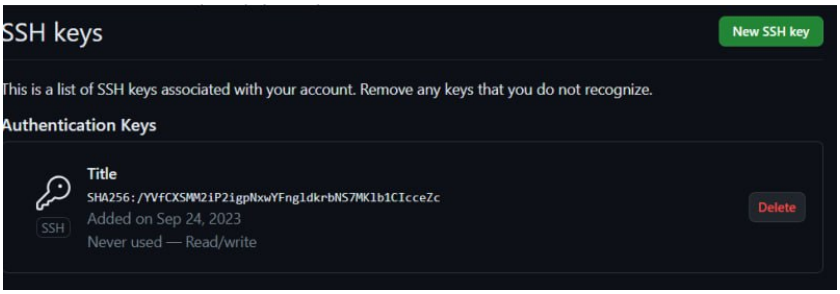


Рис. 4.8: Ключ на сайте

4.4. Создание рабочего пространства и репозитория курса на основе шаблона Открываем терминал и создаём каталог для предмета «Архитектура компьютера»: (рис 10) 10

Рис. 4.9: Создание каталога

4.5. Сознание репозитория курса на основе шаблона Создаём новый репозиторий. (Рис 11)

Рис. 4.10: Репозиторий

Открываем терминал и переходим в каталог курса. (Рис 12)

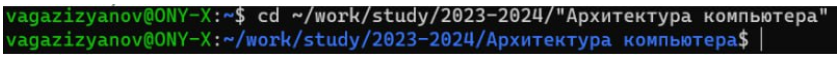


Рис. 4.11: Переход в каталог

Клонируем созданный репозиторий. (рис 13)

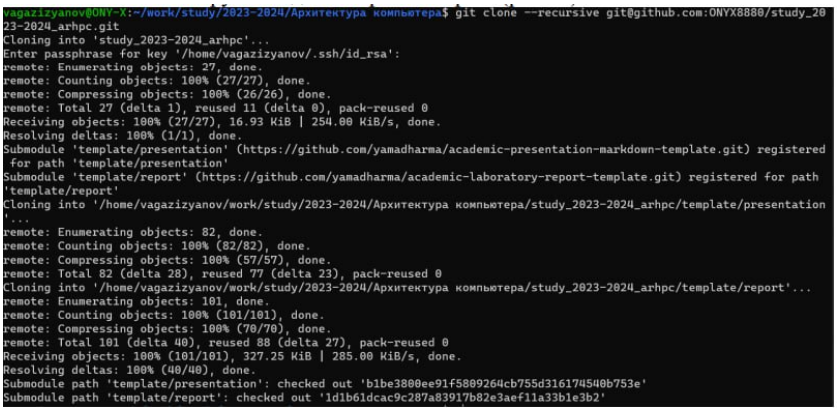


Рис. 4.12: Клон

4.6. Настройка каталога курса Переходим в каталог курса и удаляем лишние файлы. (Рис 14)

Рис. 4.13: удалиение файлов

Создаём необходимые каталоги.(Рис.15)

Рис. 4.14: Каталоги

Отправляем файлы на сервер.(Рис.16)

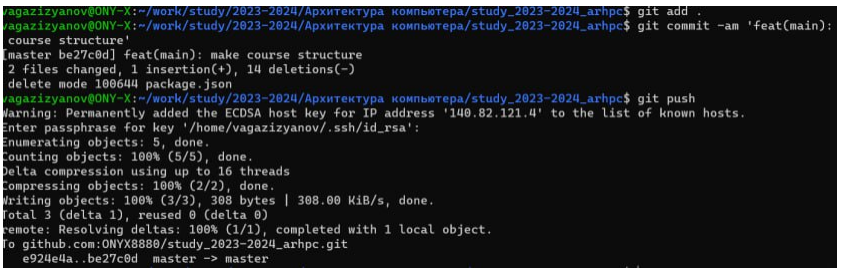
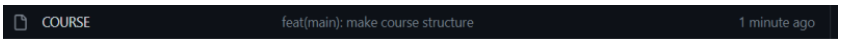


Рис. 4.15: Отправка

Проверяем правльность(Рис.17)



5.Самостаятельная работа Создаём нужный каталог.(Рис.18)

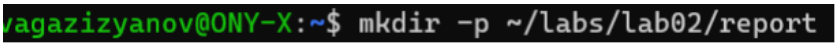


Рис. 4.16: нужный каталог

Скопируем отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданногорабочего пространства. (Рис.19, 20)

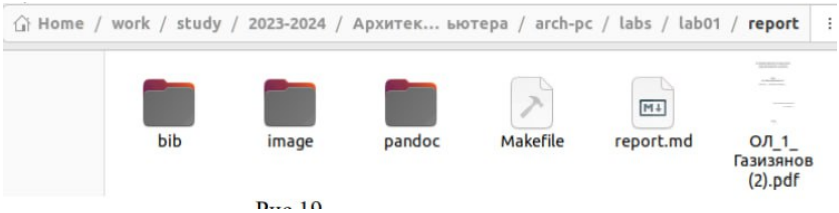


Рис. 4.17: Рис.19

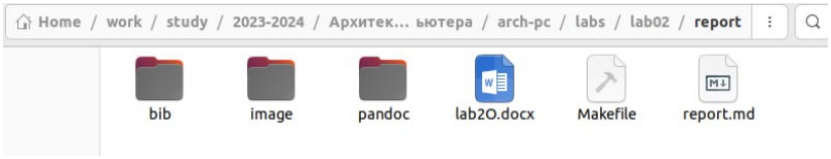


Рис. 4.18: Рис. 20

Загружаем файлы на github(Рис.21)

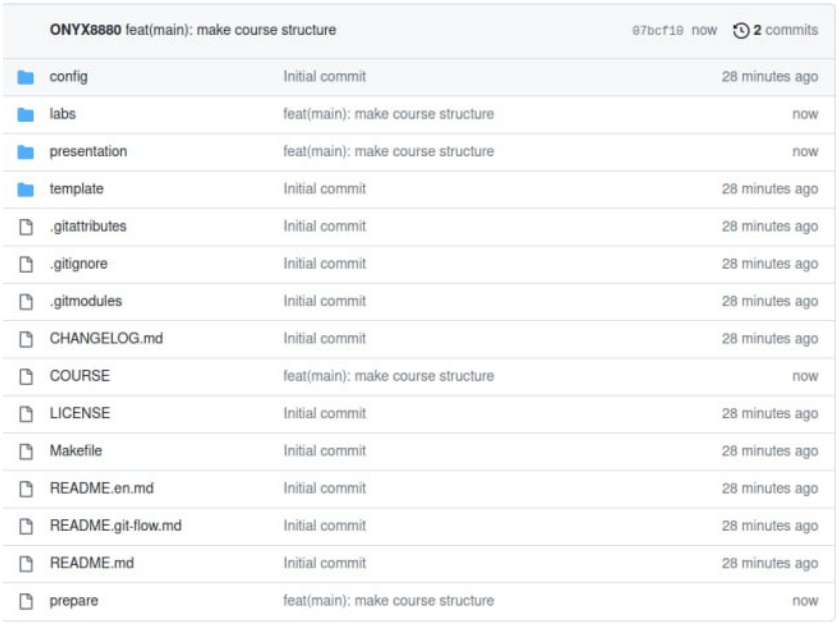


Рис. 4.19: Рис.21

5 Выводы

1. Освоены основные команды Git, что позволяет уверенно использовать систему контроля версий.
2. Понято, как Git отслеживает изменения в файлах, а также как он хранит и отображает эти изменения.
3. Изучены принципы ветвления и слияния, а также способы решения конфликтов, возникающих при слиянии веток.
4. На практике применен Git для управления проектом, в том числе для работы с удаленными репозитариями.
5. Развиты навыки, необходимые для эффективного использования Git для управления версиями кода, что позволяет улучшить качество работы и снизить возможные ошибки.