

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

дисциплина: Архитектура компьютера

Газизянов Владислав Альбертович

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	9
5	Выводы	15

Список иллюстраций

3.1	table	8
3.2	table2	8
4.1	GitHub	9
4.2	Настройка git	9
4.3	Utf-8	9
4.4	Начальная ветка	10
4.5	Параметр autocrlf	10
4.6	Параметр safecrlf	10
4.7	Генерация ключей	10
4.8	Ключ на сайте	11
4.9	Создание каталога	11
4.10	Репозиторий	11
4.11	Переход в каталог	12
4.12	Клон	12
4.13	удаление файлов	12
4.14	Каталоги	12
4.15	Отправка	13
4.16	нужный каталог	13
4.17	Рис.19	13
4.18	Рис. 20	13
4.19	Рис.21	14

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
<code>git init</code>	создание основного дерева репозитория
<code>git pull</code>	получение обновлений (изменений) текущего дерева из центрального репозитория
<code>git push</code>	отправка всех произведённых изменений локального дерева в центральный репозиторий
<code>git status</code>	просмотр списка изменённых файлов в текущей директории
<code>git diff</code>	просмотр текущих изменения
<code>git add .</code>	добавить все изменённые и/или созданные файлы и/или каталоги
<code>git add имена_файлов</code>	добавить конкретные изменённые и/или созданные файлы и/или каталоги
<code>git rm имена_файлов</code>	удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории)

Рис. 3.1: table

<code>git commit -am 'Описание коммита'</code>	сохранить все добавленные изменения и все изменённые файлы
<code>git checkout -b имя_ветки</code>	создание новой ветки, базирующейся на текущей
<code>git checkout имя_ветки</code>	переключение на некоторую ветку (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой)
<code>git push origin имя_ветки</code>	отправка изменений конкретной ветки в центральный репозиторий
<code>git merge --no-ff имя_ветки</code>	слияние ветки с текущим деревом
<code>git branch -d имя_ветки</code>	удаление локальной уже слитой с основным деревом ветки
<code>git branch -D имя_ветки</code>	принудительное удаление локальной ветки
<code>git push origin :имя_ветки</code>	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)

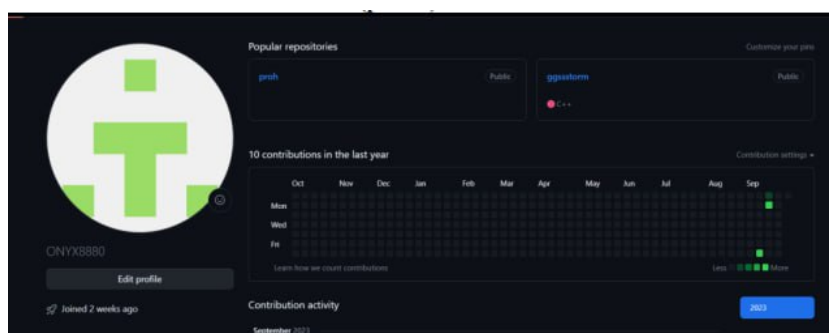


Рис. 4.1: GitHub

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.reset@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:YVfCXSM2iP2igpNxwVFngldkrbNS7MKlb1CicceZc GazizynovVladislav ony.reset@gmail
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)

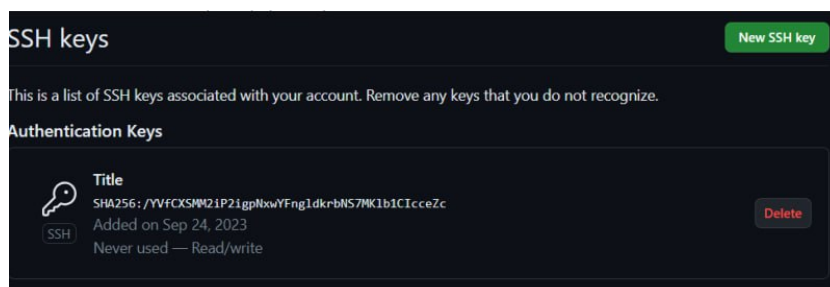


Рис. 4.8: Ключ на сайте

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

```
vagazizyanov@ONY-X:~$ mkdir -p ~/work/study/2023-2024/"Архитектура компью
```

Рис. 4.9: Создание каталога

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

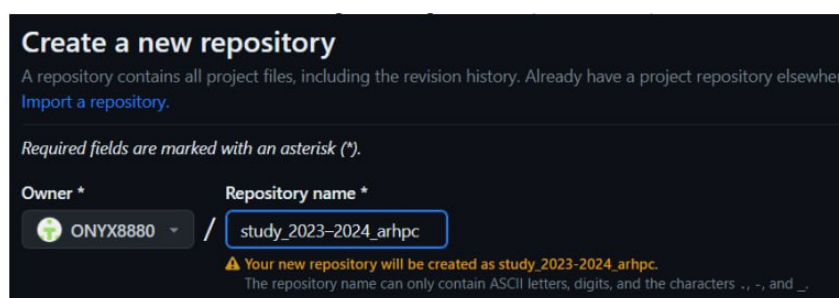


Рис. 4.10: Репозиторий

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

```
vagazizyanov@ONY-X:~$ cd ~/work/study/2023-2024/"Архитектура компьютера"
vagazizyanov@ONY-X:~/work/study/2023-2024/Архитектура компьютера$
```

Рис. 4.11: Переход в каталог

Клонируем созданный репозиторий. (рис 13)

```
vagazizyanov@ONY-X:~/work/study/2023-2024/Архитектура компьютера$ git clone --recursive git@github.com:ONYX8888/study_2023-2024_arhpc.git
Cloning into 'study_2023-2024_arhpc'...
Enter passphrase for key '/home/vagazizyanov/.ssh/id_rsa':
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 27 (delta 1), reused 11 (delta 0), pack-reused 0
Receiving objects: 100% (27/27), 16.93 KiB | 254.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.
Submodule 'template/presentation' (https://github.com/yamadharma/academic-presentation-markdown-template.git) registered for path 'template/presentation'
Submodule 'template/report' (https://github.com/yamadharma/academic-laboratory-report-template.git) registered for path 'template/report'
Cloning into '/home/vagazizyanov/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/template/presentation'...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 82 (delta 28), reused 77 (delta 23), pack-reused 0
Cloning into '/home/vagazizyanov/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/template/report'...
remote: Enumerating objects: 101, done.
remote: Counting objects: 100% (101/101), done.
remote: Compressing objects: 100% (70/70), done.
remote: Total 101 (delta 40), reused 88 (delta 27), pack-reused 0
Receiving objects: 100% (101/101), 327.25 KiB | 285.00 KiB/s, done.
Resolving deltas: 100% (40/40), done.
Submodule path 'template/presentation': checked out 'b1be3800ee91f5809264cb755d316174540b753e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a83917b82e3aef1a33b1e3b2'
```

Рис. 4.12: Клон

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

```
vagazizyanov@ONY-X:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc$ rm packag
```

Рис. 4.13: удаление файлов

Создаём необходимые каталоги.(Рис.15)

```
vagazizyanov@ONY-X:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc$ echo study_2023-2024_arhpc > COU
RSE make
```

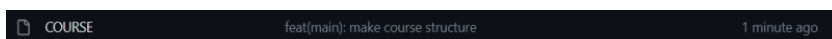
Рис. 4.14: Каталоги

Отправляем файлы на сервер.(Рис.16)

```
vagazizyanov@ONY-X:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc$ git add .
vagazizyanov@ONY-X:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc$ git commit -am 'feat(main):
course structure'
[master be27c8d] feat(main): make course structure
2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
vagazizyanov@ONY-X:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc$ git push
Warning: Permanently added the ECDSA host key for IP address '140.82.121.4' to the list of known hosts.
Enter passphrase for key '/home/vagazizyanov/.ssh/id_rsa':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 308 bytes | 308.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:ONYX8888/study_2023-2024_arhpc.git
e924e4a..be27c8d master -> master
```

Рис. 4.15: Отправка

Проверяем правильность(Рис.17)



5.Самостоятельная

работа Создаём нужный каталог.(Рис.18)

```
vagazizyanov@ONY-X:~$ mkdir -p ~/labs/lab02/report
```

Рис. 4.16: нужный каталог

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

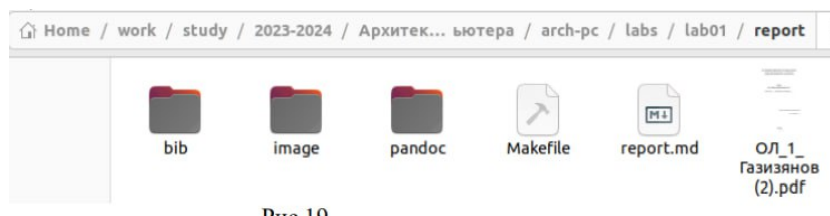


Рис. 4.17: Рис.19

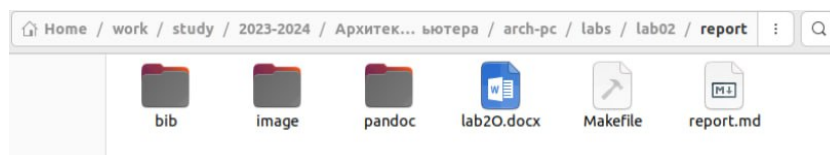


Рис. 4.18: Рис. 20

Загружаем файлы на github(Рис.21)

ONYX8880 feat(main): make course structure		07bcf10 now 2 commits
config	Initial commit	28 minutes ago
labs	feat(main): make course structure	now
presentation	feat(main): make course structure	now
template	Initial commit	28 minutes ago
.gitattributes	Initial commit	28 minutes ago
.gitignore	Initial commit	28 minutes ago
.gitmodules	Initial commit	28 minutes ago
CHANGELOG.md	Initial commit	28 minutes ago
COURSE	feat(main): make course structure	now
LICENSE	Initial commit	28 minutes ago
Makefile	Initial commit	28 minutes ago
README.en.md	Initial commit	28 minutes ago
README.git-flow.md	Initial commit	28 minutes ago
README.md	Initial commit	28 minutes ago
prepare	feat(main): make course structure	now

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 репозиториями.
5. Развиты навыки, необходимые для эффективного использования Git для управления версиями кода, что позволяет улучшить качество работы и снизить возможные ошибки.