Лабораторная работа №1

НКАбд-06-25

Хрисанова Ксения Олеговна

Содержание

1 Цель работы

Целью работы является изучение идеологии и применения средств контроля версий, приобретение практических навыков по работе с системой контроля версий git.

2 Задание

1. Создать отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства (labs/lab02/report).

2. Скопировать отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданного рабочего пространства.

3. Загрузить файлы на github.

- 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 Теоретическое введение

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 19 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.

4 Выполнение лабораторной работы

На первом этапе необходимо было выполнить базовую настройку git.

Сначала я сделала предварительную конфигурацию git. указав имя и e-mail

Далее настроила utf-8 в выводе сообщений и задала имя начальной ветки (master)

Следующим шагом задала правило преобразования символов конца строки: при добавлении файлов в репозиторий строки будут приводиться к формату LF(git config --global core.autocrlf input) и включила предупреждение при обнаружении потенциальных проблем с переводами строк (git config --global core.safecrlf warn) (рис.1)

```
kseniahrisanova@fedora:~$ git config --global user.name "<Ksenia Hrisanova>"
kseniahrisanova@fedora:~$ git config --global user.email "<kseniahrisanova07@gmail.com>"
kseniahrisanova@fedora:~$ git config --global core.quotepath false
kseniahrisanova@fedora:~$ git config --global init.defaultBranch master
kseniahrisanova@fedora:~$ git config --global core.autocrlf input
kseniahrisanova@fedora:~$ git config --global core.safecrlf warm
```

рис.1 – базовая настройка git

Далее я начала создание нового репозитория для работ по курсу “Операционные системы”. Сначала сгенерировала новый ssh-ключ (рис.2)

```
kseniahrisanova@fedora:~$ ssh-keygen -C "Хрисанова Ксения <kseniahrisanova07@gmail.com>"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/kseniahrisanova/.ssh/id_ed25519): ssh1
Enter passphrase for "ssh1" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ssh1
Your public key has been saved in ssh1.pub
The key fingerprint is:
SHA256:9q6JKRSRpN6YOBfX6EpWZ/q4iu8Mb4o+An/9EvBo3E8 Хрисанова Ксения <kseniahrisanova07@gmail.com>
The key's randomart image is:
+--[ED25519 256]--+
|    ...          |
|    .oo          |
|   o +.+         |
|  o Oo+          |
|o B.+*  S        |
|.= .=o+.E.       |
|o..o...+  .      |
|o*o o.oo.o       |
|**Bo..oo+..      |
+----[SHA256]-----+
kseniahrisanova@fedora:~$
```

рис.2 – начало генерации нового ssh-ключа

Далее:

1. С помощью команды $cat\ {\sim}/.\boldsymbol{ssh/id\_ed25519.pub}$ был выведен публичный SSH-ключ.
2. Скопированное содержимое ключа вставлено в форму на GitHub при добавлении нового SSH-ключа.
3. В поле **Title** задано имя ключа , в поле **Key** вставлен сам ключ.
4. После этого был выполнен переход к подтверждению добавления — кнопка **Add SSH key**.
   (рис.3, 4)

```
kseniahrisanova@fedora:~$ cat ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIFFG2M5c0nYjIuLIFUw2xCmqiE7vebsBZm/dYK8U0eBe Ксения Хрисанова <kseniahri
sanova07@gmail.com>
```

рис.3

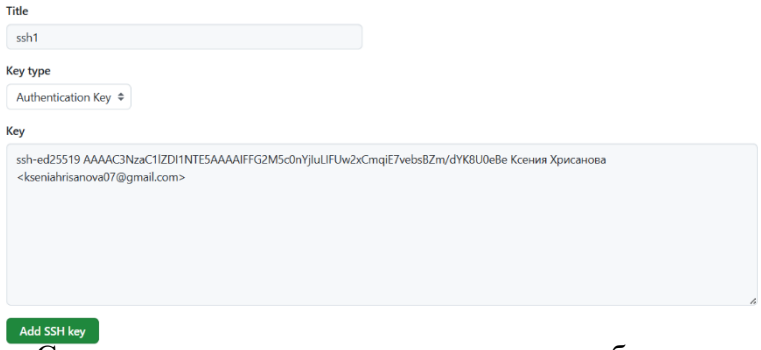


рис.4

Следующим шагом в домашнем каталоге была создана структура папок для хранения материалов курса:

$$mkdir\ -p\ {\sim}/work/study/2025-2026/\text{"Архитектура компьютера"}$$

$$cd\ {\sim}/work/study/2025-2026/\text{"Архитектура компьютера"}$$

Таким образом, был подготовлен каталог для дальнейшей работы с репозиторием.(рис.5)

```
mkdir -p ~/work/study/2025-2026/"Архитектура компьютера"
cd ~/work/study/2025-2026/"Архитектура компьютера"
```

рис.5

Потом на странице созданного репозитория на GitHub в разделе **Code → SSH** была скопирована ссылка для клонирования.(рис.6)

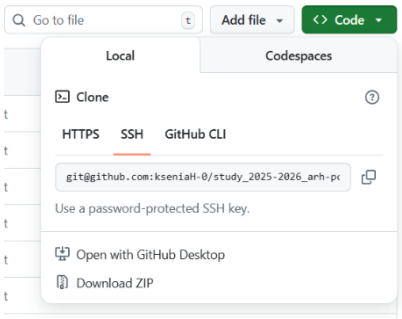


рис.6

Репозиторий был успешно склонирован в локальный каталог `arh-pc`.(рис.7)

```
kseniahrisanova@fedora:~/work/study/2025-2026/Архитектура компьютера$ git clone --recursive ssh://git@ssh.github.com:443/kseniah-0/study_2025-2026_arh-pc.git
 arh-pc
Клонирование в «arh-pc»...
The authenticity of host '[ssh.github.com]:443 ([140.82.121.36]:443)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[ssh.github.com]:443' (ED25519) to the list of known hosts.
remote: Enumerating objects: 38, done.
remote: Counting objects: 100% (38/38), done.
remote: Compressing objects: 100% (36/36), done.
remote: Total 38 (delta 1), reused 26 (delta 1), pack-reused 0 (from 0)
Получение объектов: 100% (38/38), 23.45 КиБ | 170.00 КиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/kseniahrisanova/work/study/2025-2026/Архитектура компьютера/arh-pc/template/presentation»...
remote: Enumerating objects: 161, done.
remote: Counting objects: 100% (161/161), done.
remote: Compressing objects: 100% (111/111), done.
remote: Total 161 (delta 60), reused 142 (delta 41), pack-reused 0 (from 0)
Получение объектов: 100% (161/161), 2.65 МиБ | 6.55 МиБ/с, готово.
Определение изменений: 100% (60/60), готово.
Клонирование в «/home/kseniahrisanova/work/study/2025-2026/Архитектура компьютера/arh-pc/template/report»...
remote: Enumerating objects: 221, done.
remote: Counting objects: 100% (221/221), done.
remote: Compressing objects: 100% (152/152), done.
remote: Total 221 (delta 98), reused 180 (delta 57), pack-reused 0 (from 0)
Получение объектов: 100% (221/221), 765.46 КиБ | 625.00 КиБ/с, готово.
Определение изменений: 100% (98/98), готово.
Submodule path 'template/presentation': checked out '6efd5c4ee78e4456caff3dc7062cfcad26058ca6'
Submodule path 'template/report': checked out '89a9622199b4df88227b9b3fa3d4714c85f68dd2'
```

рис.7

Далее выполнен переход в каталог с клонированным репозиторием *cd* $\sim/\boldsymbol{work}/\boldsymbol{study}/\boldsymbol{2025-2026}/$"Архитектура компьютера"$/\boldsymbol{arh-pc}$
Был выполнен запуск команды:

echo $\boldsymbol{arh-pc} > \boldsymbol{COURSE}$
$\boldsymbol{make\ prepare}$

На этапе выполнения `make prepare` система предложила установить пакет `make`, который был успешно установлен.(рис.8)

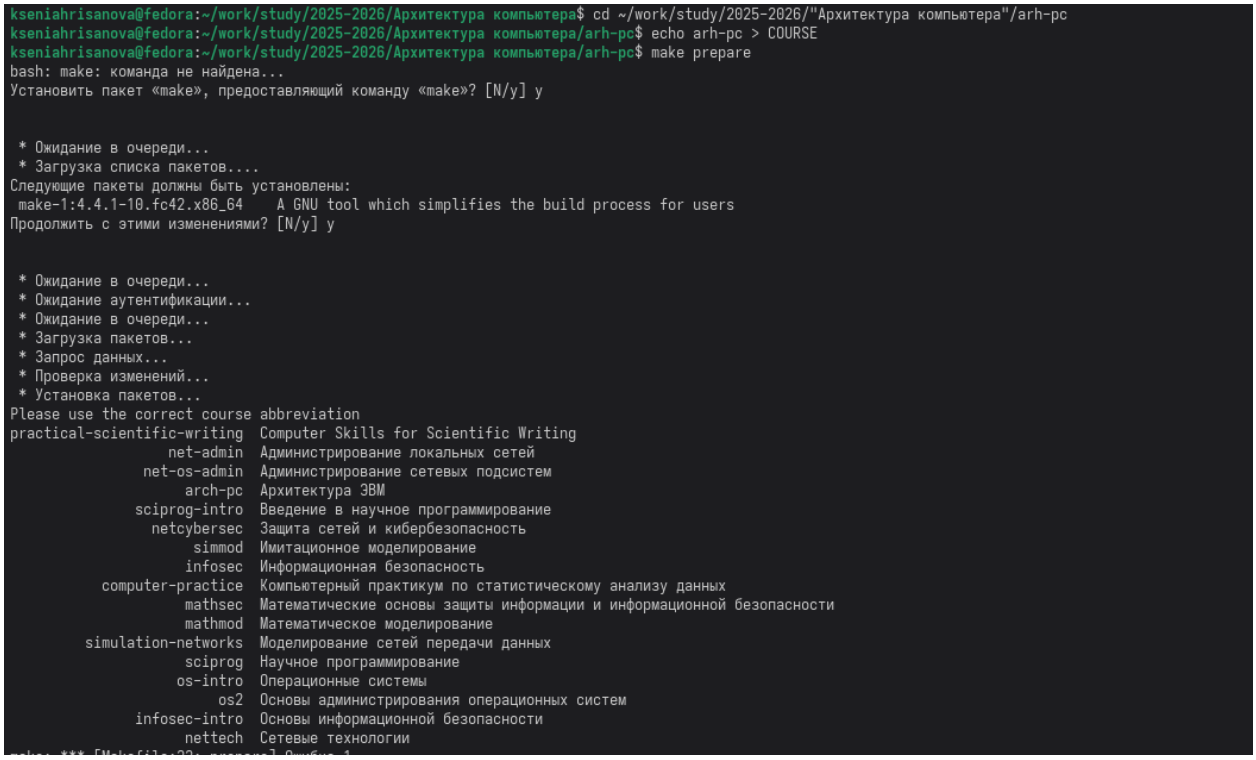


рис.8

После подготовки файлов они были добавлены в индекс Git, зафиксированы и отправлены в удалённый репозиторий:

git add .
git commit $-$ ***am*** $'feat(main)$: $make\ course\ structure'$
git push

Изменения успешно загрузились на GitHub в репозиторий `study_2025-2026_arh-pc`.(рис.9)

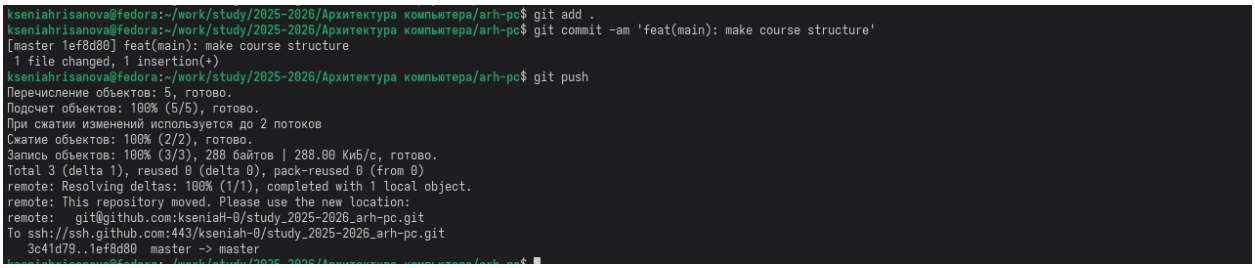


рис.9

Для проверки корректности иерархии локального репозитория была выполнена команда `ls -R`, которая отобразила все файлы и папки проекта. Структура полностью соответствует заданию. После отправки изменений командой `git push` структура была проверена на GitHub — она совпадает с локальной.

В ходе выполнения самостоятельной работы были выполнены следующие шаги:

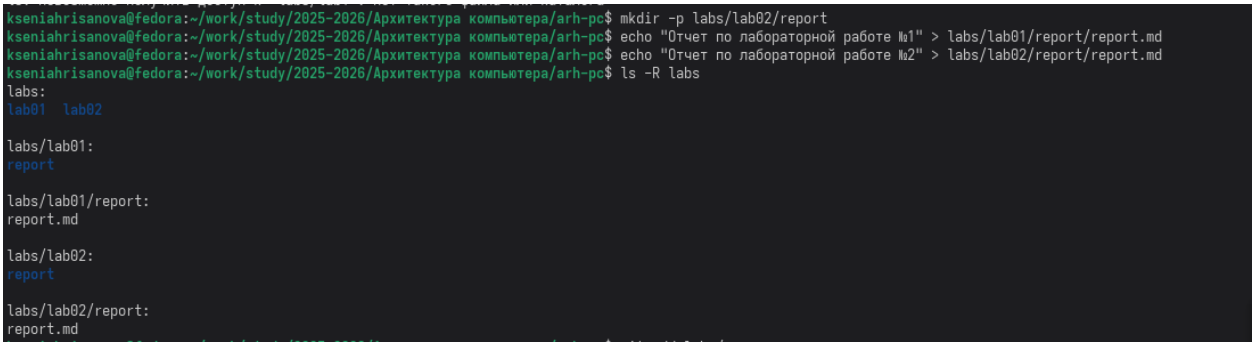
1. Создан каталог для отчёта по второй лабораторной работе:
   $mkdir - p\ labs/lab02/report$

2. В каталоги `labs/lab01/report` и `labs/lab02/report` были добавлены файлы отчётов:

   $echo$ "Отчёт по лабораторной работе №1" $> \boldsymbol{labs/lab01/report/report.md}$
   $echo$ "Отчёт по лабораторной работе №2" $> \boldsymbol{labs/lab02/report/report.md}$

3. Проверена структура каталогов с помощью команды: $ls\ - R\ labs$

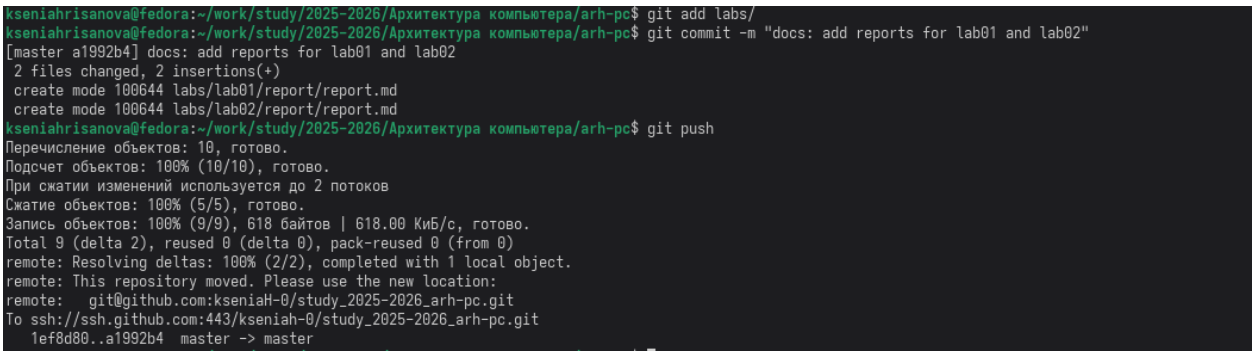
   В результате отобразились папки `lab01/report` и `lab02/report` с файлами `report.md`.



4. Все изменения были добавлены в систему контроля версий: $git\ add\ labs/$
5. Создан коммит с описанием изменений:

   $$git\ commit\ - m\ "docs: add\ reports\ for\ lab01\ and\ lab02"$$

6. Выполнена отправка изменений на сервер GitHub: $git\ push$
7. На странице удалённого репозитория проверено, что каталоги `labs/lab01/report` и `labs/lab02/report` корректно созданы, и в них присутствуют файлы отчётов.



5 Контрольные вопросы

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

Система контроля версий (VCS) - программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое. Такие системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы. Однако они могут с успехом применяться и в других областях, в которых ведётся работа с большим количеством непрерывно изменяющихся электронных документов.

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

Система контроля версий (Version Control System, VCS) — это программное обеспечение для облегчения работы с изменяющейся информацией. Репозиторий (или хранилище) - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией. Тэг commint позволяет сохранять и измененять изменения в репозитории. Рабочая копия - копия проекта, связанная с репозиторием.

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

Централизованные системы – это системы, в которых одно основное хранилище всего проекта, и каждый пользователь копирует необходимые ему файлы, изменяет и вставляет обратно. Пример – Subversion. Децентрализованные системы – система, в которой каждый пользователь имеет свой вариант репозитория и есть возможность добавлять и забирать изменения из репозиториев. Пример – Git.

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

В рабочей копии, которую исправляет человек, появляются правки, которые отправляются в хранилище на каждом из этапов. То есть в правки в рабочей копии появляются, только если человек делает их (отправляет их на сервер) и никак по-другому.

5. Опишите порядок работы с общим хранилищем VCS.

Если хранилище общее, то в рабочую копию каждого, кто работает над проектом, приходят изменения, отправленные на сервер одним из команды. Рабочая правка каждого может изменяться вне зависимости от того, делает ли конкретный человек правки или нет.

6. Каковы основные задачи, решаемые инструментальным средством git?

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

7. Назовите и дайте краткую характеристику командам git.

– создание основного дерева репозитория: git init – получение обновлений (изменений) текущего дерева из центрального репозитория: git pull – отправка всех произведённых изменений локального дерева в центральный репозиторий: git push – просмотр списка изменённых файлов в текущей директории: git status – просмотр текущих изменения: git diff – сохранение текущих изменений: – добавить все изменённые и/или созданные файлы и/или каталоги: git add . – добавить конкретные изменённые и/или созданные файлы и/или каталоги: git add – удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): git rm имена_файлов – сохранить все добавленные изменения и все изменённые файлы: git commit -am ‘Описание коммита’ – сохранить добавленные изменения с внесением комментария через встроенный редактор: git commit – создание новой ветки, базирующейся на текущей: git checkout -b имя_ветки – 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) – отправка изменений конкретной ветки в центральный репозиторий: 1 git push origin имя_ветки – слияние ветки с текущим деревом: 1 git merge –no-ff имя_ветки – удаление локальной уже слитой с основным деревом ветки: git branch -d имя_ветки – принудительное удаление локальной ветки: git branch -D имя_ветки – удаление ветки с центрального репозитория: git push origin :имя_ветки

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

Работа с удаленным репозиторием: git remote – просмотр списка настроенных удаленных репозиториев.

Работа с локальным репозиторием: git status - выводит информацию обо всех изменениях, внесенных в дерево директорий проекта по сравнению с последним коммитом рабочей ветки.

6 Выводы

В результате выполнения данной лабораторной работы я изучил идеологию и применение средств контроля версий, а также приобрел практические навыки по работе с системой git.

Список литературы

Лабораторная работа №2 (Архитектура ОС).