

Отчёт по лабораторной работе №2

Дисциплина: Архитектура компьютера

Эзиз Хатамов

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Задания для самостоятельной работы	9
5	Выводы	16
6	Ответы на контрольные вопросы для самопроверки	17

Список иллюстраций

4.1	Рисунок 1. Регистрация в Github	9
4.2	Рисунок 2. Настройка utf-8 в выводе сообщения гит	10
4.3	Рисунок 3. Создания SSH ключа	10
4.4	Рисунок 4. Копирования ключа	10
4.5	Рисунок 5. Создания SSH ключа в Github	11
4.6	Рисунок 7.Создание рабочего пространства	11
4.7	Рисунок 8.Создание рабочего пространства	12
4.8	Рисунок 9. Переход на созданный каталог	12
4.9	Рисунок 10.Копирования ссылки репозитория	13
4.10	Рисунок 11.Клонирования с помощью git clone –recursive	13
4.11	Рисунок 12.Переход на каталог и удаления лишних файлов	13
4.12	Рисунок 13. Создания каталога echo	14
4.13	Рисунок 14. Команды git add.;git commit -am	14
4.14	Рисунок 15. Отправления файлов в Github	14
4.15	Рисунок 16. Создания утилиты с помощью touch	14
4.16	Рисунок 17. Копирования файла в другой каталог	15
4.17	Рисунок 18. Отправления файла в Github	15

Список таблиц

1 Цель работы

Целью работы является изучить идеологию и применение средств контроля версий. Приобрести практические навыки по работе с системой git

2 Задание

1 Настройка github 2 Базовая настройка git 3 Создание SSH ключа 4 Создание рабочего пространства и репозитория курса на основе шаблона 5 Создание репозитория курса на основе шаблона 6 Настройка каталога курса 7 Задания для самостоятельной работы

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 зави-

симости 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

4 Задания для самостоятельной работы

Настройка github.

Чтобы создать учетную запись в Github- е я зашёл в яндекс

браузер и написал эту ссылку в браузере <https://github.com/> и написал свои данные для создания учетной записи

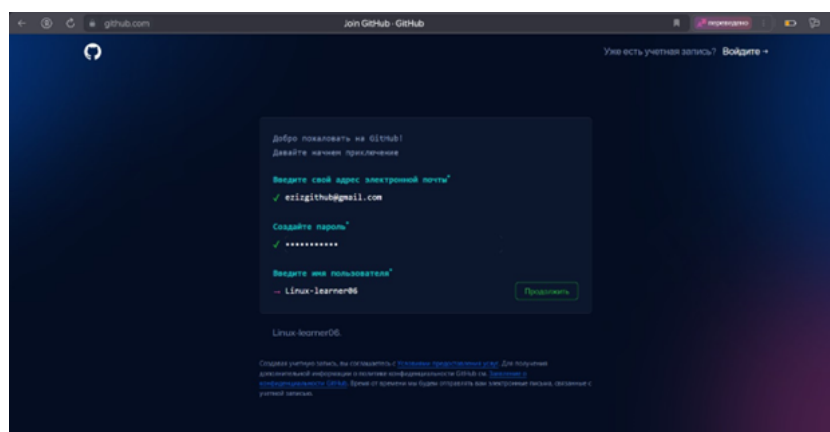


Рис. 4.1: Рисунок 1. Регистрация в Github

Базовая настройка git

Впервую очередь я сделал предварительную конфигурацию git. В терминале Я ввел нужные команды, указав свое имя и емейл. Потом настроил utf-8 в выводе сообщения git. Ещё я задал имя начальной ветки(назвал её мастер). Потом внес параметр autocrlf и carecrlf

```
Смотрите «git help git» для получения общего обзора системы.
ekhatamov@vbox:~$ git config --global user.name "<Ezizhatamov>"
ekhatamov@vbox:~$ git config --global user.email "<ezizhatamov01@gmail.com>"
ekhatamov@vbox:~$ git config --global core.quotepath false
ekhatamov@vbox:~$ git config --global init.defaultBranch master
ekhatamov@vbox:~$ git config --global core.autocrlf input
ekhatamov@vbox:~$ git config --global core.safecrlf warn
ekhatamov@vbox:~$
```

Рис. 4.2: Рисунок 2. Настройка utf-8 в выводе сообщения гит

Создание SSH ключа Чтобы создать SSH ключ я зашёл в терминал и ввел команду `ssh-keygen -C "User.nameyour_email@example.com"`. сюда написал свои данные и ввел в терминал

```
ekhatamov@vbox:~$ ssh-keygen -C "Linux-learner06<ezizgithub@gmail.com>"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ekhatamov/.ssh/id_ed25519):
/home/ekhatamov/.ssh/id_ed25519 already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ekhatamov/.ssh/id_ed25519
Your public key has been saved in /home/ekhatamov/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:t6YVDgv/gfyD2AoWYQLLHaC6thSsZp3crGqNw7W68 Linux-learner06<ezizgithub@gmail.com>
The key's randomart image is:
+--[ED25519 256]--+
|+ . |
|o=+ |
|+ +o+ |
|.oo.. |
|o= .o. S o |
|=.+ +. + * o |
| + *o .o=* |
|. +.oo..o*.. |
|oo o+Eoo o. |
+----[SHA256]-----+
ekhatamov@vbox:~$
```

Рис. 4.3: Рисунок 3. Создания SSH ключа

Потом скопировал этот ключ с помощью команды `cat ~/.ssh/id_ed25519.pub | xclip -sel clip`

```
|+ .oo..o*.. |
|oo o+Eoo o. |
+----[SHA256]-----+
ekhatamov@vbox:~$ cat ~/.ssh/id_ed25519.pub | xclip -sel clip
ekhatamov@vbox:~$
```

Рис. 4.4: Рисунок 4. Копирования ключа

После того как скопировал ключ я перешел в свой аккаунт в гитхаб и зашел на настройки и там есть раздел `ssh and gpg keys` зашел туда и создал SSH ключ

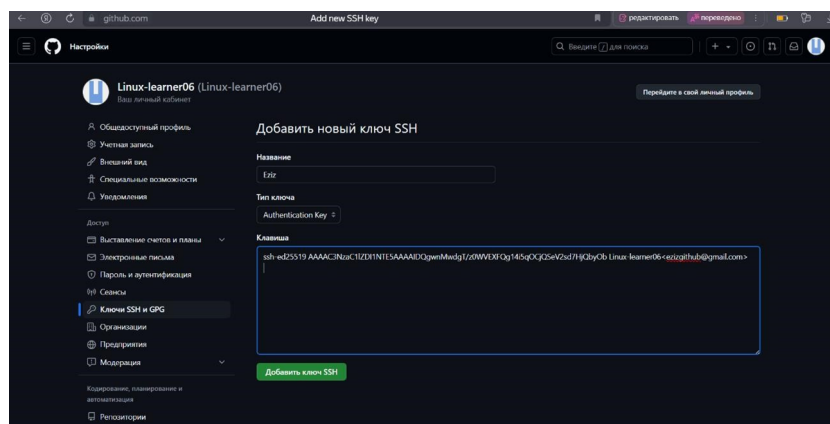


Рис. 4.5: Рисунок 5. Создания SSH ключа в Github

Создание рабочего пространства и репозитория курса на основе шаблона

Я открыл терминал и ввел нужную команду и создал каталог для предмета Архитектура компьютеров и проверил правильность своих действий с помощью команды ls.

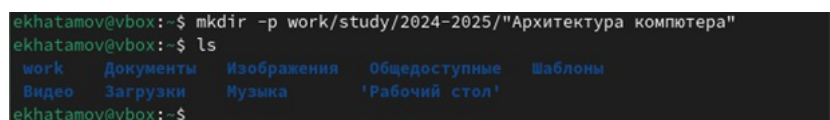


Рис. 4.6: Рисунок 7.Создание рабочего пространства

Создание репозитория курса на основе шаблона Я зашел В браузер и перешел на страницу репозитория с шаблоном курса по адресу <https://github.com/yamadharm/course-directorystudent-template> . Далее выбрал «Use this template», чтобы использовать этот шаблон для своего репозитория

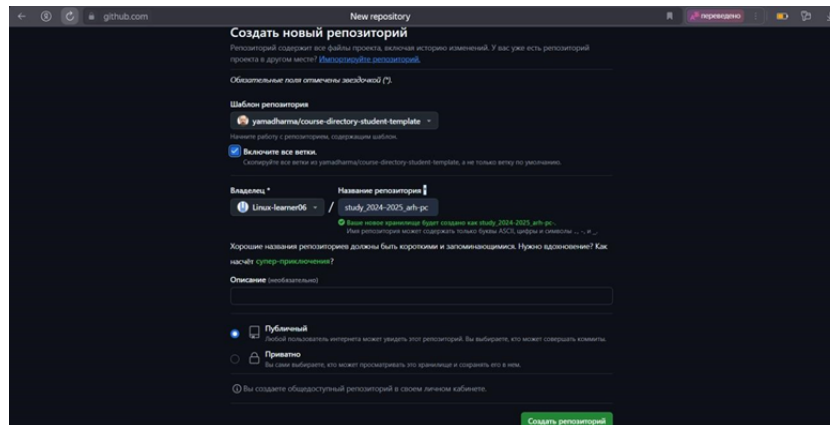


Рис. 4.7: Рисунок 8.Создание рабочего пространства

Теперь перехожу в созданный каталог курса с помощью утилиты cd:

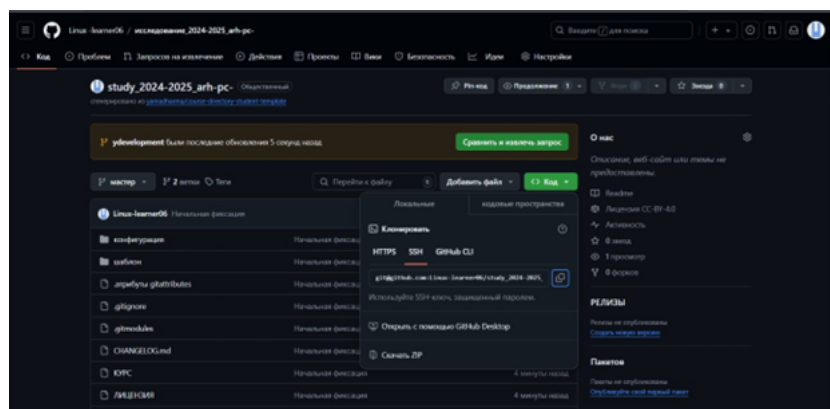


Рис. 4.8: Рисунок 9. Переход на созданный каталог

Для начало я зашел на Github и скопировал ссылку для клонирования на странице созданного репозитория Code -> SSH:

```

ekhatamov@vbox: $ cd ~/work/study/2024-2025/"Архитектура компьютера"
ekhatamov@vbox:~/work/study/2024-2025/Архитектура компьютера$ git clone --recursive git@github.com:Linux-learner06/study_2024-2025_arh-pc-.git
Клонирование в «study_2024-2025_arh-pc»...
remote: Enumerating objects: 34, done.
remote: Counting objects: 100% (34/34), done.
remote: Compressing objects: 100% (33/33), done.
remote: Total 34 (delta 1), reused 18 (delta 0), pack-reused 0 (from 0)
Получение объектов: 100% (34/34), 19.58 КиБ | 527.00 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/ekhatamov/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc-/template/presentation»...
remote: Enumerating objects: 111, done.
remote: Counting objects: 100% (111/111), done.
remote: Compressing objects: 100% (77/77), done.
remote: Total 111 (delta 42), reused 100 (delta 31), pack-reused 0 (from 0)
Получение объектов: 100% (111/111), 102.17 КиБ | 1.13 МиБ/с, готово.
Определение изменений: 100% (42/42), готово.
Клонирование в «/home/ekhatamov/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc-/template/report»...
remote: Enumerating objects: 142, done.
remote: Counting objects: 100% (142/142), done.
remote: Compressing objects: 100% (97/97), done.
remote: Total 142 (delta 60), reused 121 (delta 39), pack-reused 0 (from 0)
Получение объектов: 100% (142/142), 341.09 КиБ | 2.42 МиБ/с, готово.
Определение изменений: 100% (60/60), готово.
Submodule path 'template/presentation': checked out 'c9b2712b4b2d431ad5086c9c72a02bd2fca1d4a6'
Submodule path 'template/report': checked out 'c26e22effe7b3e0495707d82ef561ab185f5c748'
ekhatamov@vbox:~/work/study/2024-2025/Архитектура компьютера$

```

Рис. 4.9: Рисунок 10. Копирования ссылки репозитория

Потом зашел на терминал и ввел `git clone --recursive` и вставил то что скопировал

```

ekhatamov@vbox: ~/work/study/2024-2025/Архитектура компьютера$ cd ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc
ekhatamov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ rm package.json
ekhatamov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$

```

Рис. 4.10: Рисунок 11. Клонирования с помощью `git clone --recursive`

Настройка каталога курса

Сперва, в терминале я перехожу в каталог курса с помощью утилиты `cd` в каталог `arch-pc` и удаляю лишний файл `package.json` с помощью `rm`

```

ekhatamov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ echo arch-pc > COURSE
ekhatamov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare       Generate directories structure
  submodule     Update submodules
ekhatamov@vbox:~/work/study/2024-2025/Архитектура компьютера$

```

Рис. 4.11: Рисунок 12. Переход на каталог и удаления лишних файлов

Затем создаю необходимые каталоги с помощью `echo arch-pc > COURSE`

```

ekhatamov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ echo arch-
pc > COURSE
ekhatamov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare       Generate directories structure
  submodule     Update submules
ekhatamov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$

```

Рис. 4.12: Рисунок 13. Создания каталога echo

Теперь отправляю файлы на сервер с помощью `git add .; git commit -am; git push`

```

ekhatamov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git push
Перечисление объектов: 5, готово.
Сжатие объектов: 100% (5/5), готово.
При скатив изменений используется до 4 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (1/1), 292 байта | 292.00 KiB/c, готово.
Total: 3 (delta 1), reused 0 (delta 0), pack-reused 9 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:linux-learner06/study_2024-2025_arch-pc.git
 c884113..f8649a0 master -> master

```

Рис. 4.13: Рисунок 14. Команды `git add.`; `git commit -am`

```

make: Целей «Makefile» не требует выполнения команд.
ekhatamov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ touch ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab01/report/"00
1_Хатамов.отчет.pdf"
ekhatamov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ ls ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab01/report
001_Хатамов.отчет.pdf

```

Рис. 4.14: Рисунок 15. Отправления файлов в Github

Задания для самостоятельной работы

1. Создал отчет по выполнению лабораторной работы в `labs/lab02/report/` с помощью утилиты `touch`

```

ekhatamov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ cp ~/Документы/"001_Хатамов.отчет.pdf" ~/work/study/2024-2025/Архитектура компью
тера/arch-pc/labs/lab01/report/
ekhatamov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ ls ~/Документы/"001_Хатамов.отчет.pdf" ~/work/study/2024-2025/Архитектура компью
тера/arch-pc/labs/lab01/report/
/home/ekhatamov/Документы/001_Хатамов.отчет.pdf
/home/ekhatamov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab01/report/"001_Хатамов.отчет.pdf"
ekhatamov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$

```

Рис. 4.15: Рисунок 16. Создания утилиты с помощью `touch`

2. Потом скопировал предыдущую лабораторную работу в соответствующий каталог. Предыдущая работа находилась в каталоге Документы. Я воспользовался командой `cp` и скопировал файл и проверил его наличие с помощью `ls`.

```
skhatamov@ubuntu: ~/work/study/2024-2025/ispstesttype_kompyutera/arch-pc$ git add .
skhatamov@ubuntu: ~/work/study/2024-2025/ispstesttype_kompyutera/arch-pc$ git commit -m "Added lab02 report, its directory and others"
[master f5aaaae] Added lab02 report, its directory and others
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 labs/lab01/report/001_khatamov_order.pdf
create mode 100644 labs/lab02/report/002_khatamov_order.pdf
skhatamov@ubuntu: ~/work/study/2024-2025/ispstesttype_kompyutera/arch-pc$ git push
Перечисление объектов: 10, готово.
Подсчет объектов: 100% (10/10), готово.
При скатии изменений используется до 4 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (9/9), 1.09 МБ | 5.26 МБ/с, готово.
total 9 (delta 1), reused 1 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:linux-learner06/study_2024-2025_arh-pc.git
 f8649a0..f5aaaae master -> master
skhatamov@ubuntu: ~/work/study/2024-2025/ispstesttype_kompyutera/arch-pc$
```

Рис. 4.16: Рисунок 17. Копирования файла в другой каталог

3.Теперь загружаю файлы на github используя git add .; git commit - m; git push

Рисунок 18. Отправления файла в Github

Рис. 4.17: Рисунок 18. Отправления файла в Github

5 Выводы

При выполнении данной лабораторной работы я изучил применение средств контроля версий, а также приобрел практические навыки по работе с системой git и узнал как пересылать файлы на github.

6 Ответы на контрольные вопросы для самопроверки

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Системы контроля версий (Version Control Systems, VCS) — это инструменты, позволяющие отслеживать изменения в файлах и координировать работу над проектами, особенно в командной среде. Основные задачи VCS включают: 1.Хранение и управление версиями файлов и папок. 2.Отслеживание изменений, внесённых в проект.
 2. Возможность отката к предыдущим версиям файлов.
 3. Упрощение коллаборации между разработчиками.
 4. Управление конфликтами, возникающими при параллельной работе нескольких людей. 2.Объясните следующие понятия VCS и их отношения:хранилище,commit, история,рабочая копия.
- Хранилище (Repository) — это место, где сохраняются все версии файлов проекта, а также информация о всех изменениях и их авторах. Хранилище может быть локальным или удалённым.
 - Commit — это команда, которая сохраняет изменения в хранилище. Каждый commit включает описание изменений и метаданные (например, автор и дата). Это позволяет отслеживать историю изменений.
 - История — это последовательность всех коммитов в хранилище, в которой зафиксированы изменения, даты их внесения и авторы.
 - Рабочая копия (Working Copy) — это локальная версия файлов, с которой

пользователь работает. В ней находятся актуальные файлы, включая изменения, которые еще не были закоммичены.

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

- Централизованные VCS (например, Subversion, CVS) работают с центральным сервером, где хранится полная версия проекта. Пользователи работают с копией проекта и по окончании вносят изменения в центральное хранилище. При работе требуется постоянное соединение с сервером.
- Децентрализованные VCS (например, Git, Mercurial) позволяют каждому пользователю иметь полную копию репозитория на своем устройстве. Изменения могут сохраняться локально, а синхронизация с удалённым репозиторием происходит по мере необходимости. Это позволяет работать без постоянного соединения, поддерживает более гибкие и дистрибутивные потоки работы.

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

1. Создание репозитория — инициализация нового проекта с помощью команды `git init`.
2. Добавление файлов — добавление файлов в индекс с помощью команды `git add`.
3. Коммит изменений — сохранение изменений в репозитории с помощью команды `git commit -m "Комментарий"`.
4. Просмотр истории — команда `git log` позволяет отслеживать все коммиты.
5. Откат изменений — использование команды `git checkout` для возврата к предыдущим версиям.

5. Опишите порядок работы с общим хранилищем VCS.

1. Клонирование репозитория — получение локальной копии общего репозитория с помощью команды `git clone`.

2. Получение изменений — команда `git pull` для синхронизации локальной копии с удалённой (обновления).
 3. Работа над новыми изменениями — внесение изменений в рабочую копию и создание коммитов.
 4. Обновление локальной версии — решение конфликтов, если они возникли, при попытке синхронизации.
 5. Отправка изменений — отправка коммитов на удалённый репозиторий с помощью команды `git push`.
6. Каковы основные задачи, решаемые инструментальным средством `git`?
1. Отслеживание изменений и версий файлов.
 2. Управление различными ветками разработки.
 3. Поддержка коллаборации между разработчиками.
 4. Анализ истории изменений.
 5. Управление конфликтами и слиянием веток.
7. Назовите и дайте краткую характеристику командам `git`.
1. `git init` — инициализация нового репозитория.
 2. `git clone` — клонирование удалённого репозитория на локальную машину.
 3. `git add` — добавление изменений в индекс для следующего коммита.
 4. `git commit -m ""` — сохранение изменений в репозитории с сообщением.
 5. `git status` — просмотр статуса рабочей копии (изменённые, добавленные файлы).
 6. `git push` — отправка коммитов на удалённый репозиторий.
 7. `git pull` — получение последних изменений из удалённого репозитория и их слияние с локальной копией.
 8. `git branch` — управление ветками (просмотр, создание, удаление).
 9. `git checkout` — переключение между ветками или возврат к предыдущим версиям.

Компьютерные науки и технологии программирования. Раздел “Архитектура компьютеров”. (esystem.rudn.ru) <https://esystem.rudn.ru/course/view.php?id=133>
<https://esystem.rudn.ru/mod/resource/view.php?id=1030822>