

Первоначальна настройка git.

НПИБд-01-22

Коннова Татьяна Алексеевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Самостоятельная работа	13
4	Выводы	17

Список иллюстраций

3.1	KONNOVAT	7
3.2	config	8
3.3	rsa	9
3.4	keysik	9
3.5	mkdir	10
3.6	git	10
3.7	public	11
3.8	cd	11
3.9	clone	12
3.10	settings	12
3.11	commit	12
3.12	git push	13

Список таблиц

1 Цель работы

Изучить идеологию и применение средств контроля версий.

Освоить умения по работе с git.

2 Задание

Изучить идеологию и применение средств контроля версий.

Освоить умения по работе с git.

3 Выполнение лабораторной работы

Существует несколько доступных серверов репозиториев с возможностью бесплатного размещения данных. Например, <http://bitbucket.org/>, <https://github.com/> и <https://gitflic.ru>. Для выполнения лабораторных работ воспользуюсь Github.

Создала учётную запись на сайте <https://github.com/> и заполнила основные данные. (рис. [3.1])

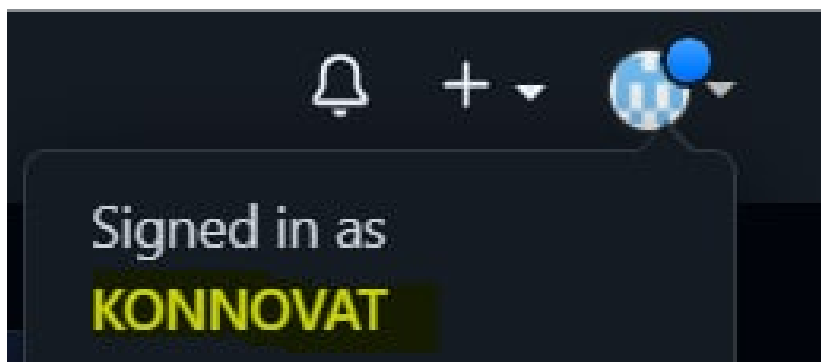


Рис. 3.1: KONNOVAT

Сначала сделаем предварительную конфигурацию git. Открываем терминал и введём следующие команды, указав мое имя и email как владельца репозитория:

```
git config --global user.name "Tatyana Konnova"
```

```
git config --global user.email "konnovav05@gmail.com"
```

```
Настроим utf-8 в выводе сообщений git: git config --global core.quotePath false
```

```
Зададим имя начальной ветки (будем называть её master):
```

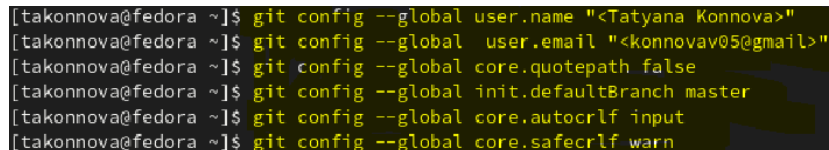
`git config --global init.defaultBranch master`

Параметр `autocrlf`:

`git config --global core.autocrlf input`

Параметр `safecrlf`:

`git config --global core.safecrlf warn` (рис. [3.2])



```
[takonnova@fedora ~]$ git config --global user.name "<Tatyana Konnova>"
[takonnova@fedora ~]$ git config --global user.email "<konnovav05@gmail>"
[takonnova@fedora ~]$ git config --global core.quotepath false
[takonnova@fedora ~]$ git config --global init.defaultBranch master
[takonnova@fedora ~]$ git config --global core.autocrlf input
[takonnova@fedora ~]$ git config --global core.safecrlf warn
```

Рис. 3.2: config

Создание SSH ключа Для последующей идентификации пользователя на сервере репозитория

необходимо сгенерировать пару ключей (приватный и открытый):

`ssh-keygen -C "Татьяна Коннова konnovav05@gmail.com"` Рис. 3

Ключи сохраняться в каталоге `~/.ssh/`.

Далее необходимо загрузить сгенерированный открытый ключ. Для этого зайдём на

сайт <http://github.org/> под своей учётной записью и перейдем в меню Settings .

После этого

выберем в боковом меню SSH and GPG keys и нажмем кнопку New SSH key .

Скопировав из локальной консоли ключ в буфер обмена

`cat ~/.ssh/id_rsa.pub` (рис. [3.3])


```
[takonnova@fedora ~]$ ssh-keygen -C "Татьяна Коннова <konnovav05@gmail.com>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/takonnova/.ssh/id_rsa):
Created directory '/home/takonnova/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/takonnova/.ssh/id_rsa
Your public key has been saved in /home/takonnova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:R/m/WdzgJlovGluemZHNlCY/HtTF74gSLJuGgaRPjg Татьяна Коннова <konnovav05@gmail.co
m>
The key's randomart image is:
+---[RSA 3072]-----+
|  .   . o |
| o   . . + . |
| o .   + +   o . |
| E +   . o o . . . o |
| . o . $ . ooo.+ |
|   o o . . +o==o |
|   . +   =oo+o= |
|   .   oB+.+ |
|   o+ o+ |
+---[SHA256]-----+
[takonnova@fedora ~]$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQDPKheP4XNyXrhDFL0DrZ9DCd60CuS830va5isDfV2CcgExZw0
BmM7ni0UesaxYeLyk14rkIoBYpZZQPBhVsz3Icsu5kK9n52ac/qhxV57jT7KY1hc9oT1GLN07Y/5gNfnwyMF3hZ
85L9SsQYjDgST94ylif20jInz1G+16Pp/UtTeFkXCgt5PzDbbvIyi4bIsRrUfo977Xa1lAmzT58GBMfqVJYjBat
rjRdrEwAd7rWcLQRdZILZzuG+wwLIILAmUvcQc36X0CzQornj2MZMkHhdbG92DoHtc10u0EhkJHNe/RRJ8jYsli
2lwe+ic/h3sfqh/dxJvLDjfgHwQ1+8X0raamijNFKXyjlNdXOWTuFjmZa8AbVQEqAAS1mR9lmH1fLbPQLA1naCp
UQAaUIwN03XiHnMeJfVfSdyBB7ZBlmL3Yqu0RTNFctInifQYwYzuiHbZDg/dzKdcS7dVXLQ5kmcbbGg4gSN4su6
+7PzpyaTtkeJ4fL7xiKDRtD0jBDrM= Татьяна Коннова <konnovav05@gmail.com>
```

Рис. 3.3: rsa

Вставляем ключ в появившееся на сайте поле и указываем для ключа имя (Keysik) (рис. [3.4])

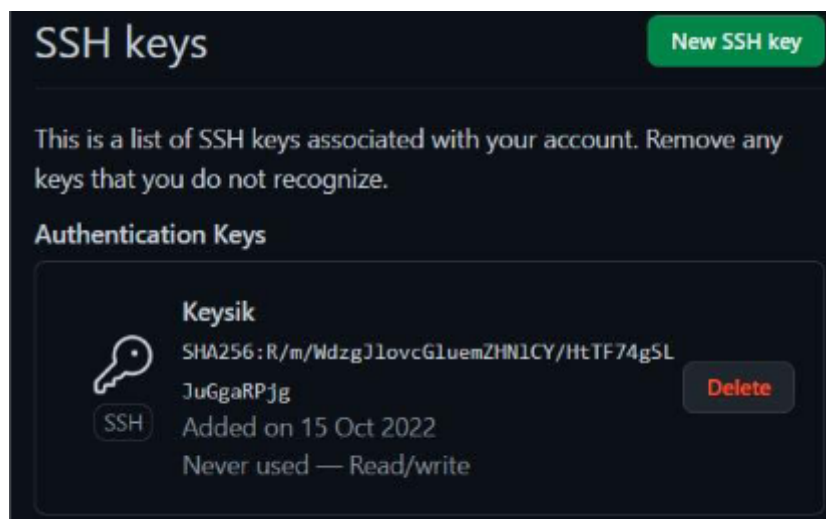


Рис. 3.4: keysik

ознание рабочего пространства и репозитория курса на основе шаблона При

выполнении лабораторных работ следует придерживаться структуры рабочего пространства. Рабочее пространство по предмету располагается в следующей иерархии: ~/work/study/ **XXXX** / ... • Каталог для лабораторных работ имеет вид labs.

- Каталоги для лабораторных работ имеют вид lab, например: lab01, lab02 и т.д. Название проекта на хостинге git имеет вид: study_ Например, для 2022–2023 учебного года и предмета ... (код предмета arch-рс) название проекта примет следующий вид: study_2022–2023... Откроем терминал и создадим каталог для предмета: `mkdir -p ~/work/study/2022-2023/....`(рис. [3.5])

```
[takonnova@fedora ~]$ mkdir -p ~/work/study/2022-2023/"Архитектура компьютера"
```

Рис. 3.5: mkdir

Создание репозитория курса на основе шаблона

Репозиторий на основе шаблона можно создать через web-интерфейс github.

Перейдём на страницу репозитория с шаблоном курса <https://github.com/yamadharma/course-directory-student-template>.(рис. [3.6])

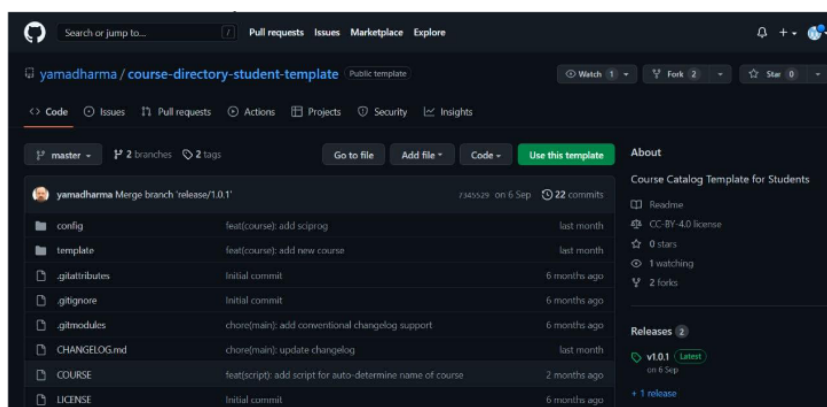


Рис. 3.6: git

Далее выберем Use this template. В открывшемся окне задаем имя репозитория (Repository name) 5 study_2022–2023_.... и создаем репозиторий (рис. [3.7])

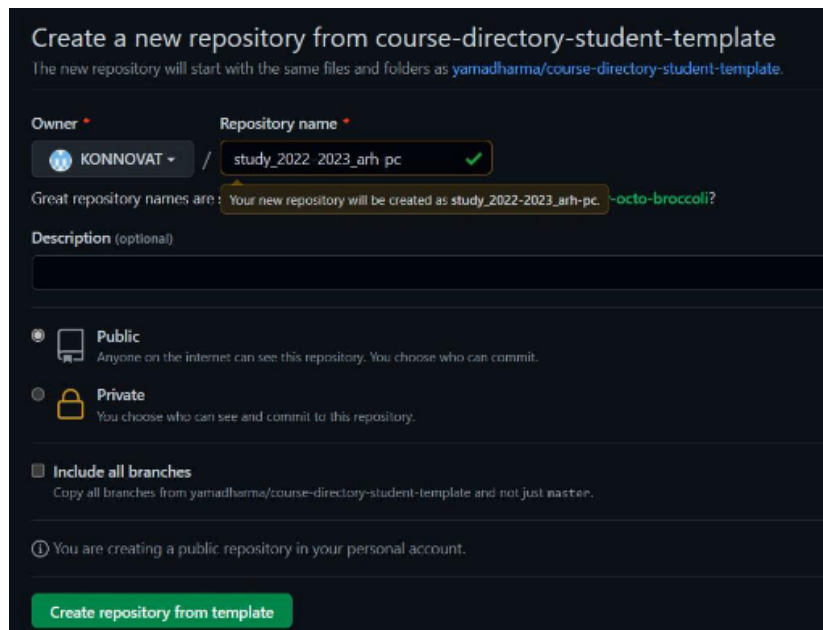


Рис. 3.7: public

Откроем терминал и перейдём в каталог курса: `cd ~/work/study/2022-2023/“Архитектура компьютера”`(рис. [3.8])

```
[takonnova@fedora Архитектура компьютера]$ git clone --recursive git@github.com:
KONNOVAT/study_2022-2023_arh-pc.git arch-pc
Клонирование в «arch-pc»...
The authenticity of host 'github.com (140.82.121.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvC0qU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
remote: Enumerating objects: 26, done.
remote: Counting objects: 100% (26/26), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 26 (delta 0), reused 17 (delta 0), pack-reused 0
Получение объектов: 100% (26/26), 16.39 КиБ | 16.39 Миб/с, готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presen
tation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-r
eport-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/takonnova/work/study/2022-2023/Архитектура компьютера/arch
-pc/template/presentation»...
remote: Enumerating objects: 71, done.
```

Рис. 3.8: cd

Клонируем созданный репозиторий: `git clone --recursive git@github.com:KONNOVAT/study_2022-2023_arh-pc.git arch-pc`(рис. [3.9])

```
[takonnova@fedora arch-pc]$ git commit -am 'feat(main): make course structure'
[master 98fec6b] feat(main): make course structure
91 files changed, 8229 insertions(+), 14 deletions(-)
create mode 100644 labs/lab01/presentation/Makefile
create mode 100644 labs/lab01/presentation/image/kulyabov.jpg
create mode 100644 labs/lab01/presentation/presentation.md
create mode 100644 labs/lab01/report/Makefile
create mode 100644 labs/lab01/report/bib/cite.bib
create mode 100644 labs/lab01/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab01/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab01/report/report.md
```

Рис. 3.9: clone

Настройка каталога курса

Перейдем в каталог курса:

`cd ~/work/study/2022-2023/...` Удаляем лишние файлы:

`rm package.json`

Создаем необходимые каталоги:

`echo arch-pc > COURSE`

`make`

Отправляем файлы на сервер:

`git add .` (рис. [3.10])

```
[takonnova@fedora Архитектура компьютера]$ cd ~/work/study/2022-2023/"Архитектура компью
тера"/arch-pc
[takonnova@fedora arch-pc]$ rm package.json
[takonnova@fedora arch-pc]$ echo arch-pc > COURSE
[takonnova@fedora arch-pc]$ make
[takonnova@fedora arch-pc]$ git add .
```

Рис. 3.10: settings

`git commit -am 'feat(main): make course structure'` (рис. [3.11])

```
[takonnova@fedora arch-pc]$ git push
Перечисление объектов: 22, готово.
Подсчет объектов: 100% (22/22), готово.
Сжатие объектов: 100% (16/16), готово.
Запись объектов: 100% (20/20), 310.95 КиБ | 2.19 МиБ/с, готово.
Всего 20 (изменений 1), повторно использовано 0 (изменений 0), повторн
кетов 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:KONNOVAT/study_2022-2023_arh-pc.git
  2b4687b..98fec6b master -> master
```

Рис. 3.11: commit

git push (рис. [3.12])

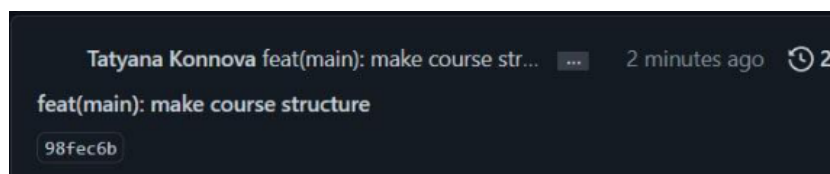


Рис. 3.12: git push

Проверяем правильность создания иерархии рабочего пространства в локальном репозитории и на странице github, проверила, изменения вступили в силу. (рис. [??])

github

3.1 Самостоятельная работа

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

Системы контроля версий (VCS) — это программные инструменты, которые помогают управлять изменениями исходного кода с течением времени. Они позволяют разработчикам отслеживать изменения, возвращаться к предыдущим версиям и сотрудничать с другими разработчиками. Они также позволяют хранить исходный код и делиться им с другими.

- 2) Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Хранилище — это место, где хранится и управляется исходный код. Это основная область хранения для всех версий проекта. Commit — это сохраненная версия проекта. Он представляет собой снимок проекта в определенный момент времени. Коммиты используются для отслеживания изменений кода с течением времени. История: история проекта — это запись всех коммитов, сделанных в репозиторий. Он показывает, как

проект развивался с течением времени и какие изменения были внесены разработчиками. Рабочая копия — это локальная копия репозитория. Она используется разработчиками для внесения изменений в код, не затрагивая основной репозиторий. Затем изменения могут быть зафиксированы в репозитории, когда они будут готовы.

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

Централизованные системы контроля версий (CVCS) — это системы, в которых все файлы и изменения хранятся на одном сервере. Все пользователи получают доступ к одному и тому же серверу для фиксации, отправки и получения изменений. Примеры CVCS включают Subversion (SVN) и Perforce. Децентрализованные системы контроля версий (DVCS) — это системы, в которых у каждого пользователя есть собственная локальная копия репозитория. Изменения можно зафиксировать в локальном репозитории, а затем отправить в репозитории других пользователей. Примеры DVCS включают Git и Mercurial.

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

При единоличной работе с системой контроля версий пользователь может фиксировать изменения в своем локальном репозитории. Это включает в себя добавление новых файлов, редактирование существующих файлов и удаление файлов. Затем пользователь может отправить эти изменения в центральный репозиторий, где они будут храниться для доступа других пользователей. Пользователь также может получать изменения из центрального репозитория, который будет обновлять их локальный репозиторий последними изменениями, сделанными другими пользователями.

- 5) Опишите порядок работы с общим хранилищем VCS.

1. Создаем локальный репозиторий. Основным шагом к работе с системой контроля версий является создание локального репозитория либо путем

- клонирования существующего репозитория с удаленного сервера, либо путем создания нового репозитория с нуля.
2. Вносим изменения. После создания локального репозитория пользователи могут вносить изменения в свои локальные файлы: добавление новых файлов, редактирование существующих файлов и удаление файлов.
 3. Фиксируем изменения: после внесения изменений в локальный репозиторий пользователи могут зафиксировать свои изменения. Мы сохраняем изменения в локальном репозитории и позволим их отслеживать.
 4. Нужно отправить изменения: после внесения изменений в локальный репозиторий пользователи могут отправить эти изменения в центральный репозиторий. Это сохранит изменения, чтобы другие могли получить доступ и обновить свои локальные репозитории последними изменениями.
 5. Нужно извлечь изменения: пользователи также могут извлекать изменения из центрального репозитория. Это обновит их локальный репозиторий последними изменениями, внесенными другими пользователями.
- 6) Каковы основные задачи, решаемые инструментальным средством git?
- Контроль версий: Git позволяет пользователям отслеживать изменения в своем коде с течением времени и при необходимости возвращаться к предыдущим версиям. Ветвление Git позволяет пользователям создавать несколько веток проекта для разных целей, таких как разработка, тестирование и производство. Слияние Git позволяет пользователям объединять разные ветки вместе. Совместная работа: Git упрощает совместную работу нескольких пользователей над проектом, позволяя им отправлять и извлекать изменения из репозитория друг друга. Безопасность. Git обеспечивает безопасный способ хранения исходного кода и управления им с помощью шифрования и аутентификации.
- 7) Назовите и дайте краткую характеристику командам git. git push: отправляет изменения в удаленный репозиторий. git pull: извлекает изменения из удаленного репозитория. git status: Проверяет текущий статус репозитория.

рия. `git init`: инициализирует новый локальный репозиторий Git. `git clone`: клонирует существующий репозиторий из удаленного источника. `git add`: Добавляет файлы в промежуточную область для фиксации. `git commit`: фиксирует изменения в локальном репозитории.

`git branch`: создает ветки в репозитории и управляет ими. 8) Приведите примеры использования при работе с локальным и удалённым репозиториями. Локальный репозиторий: `git add`: Добавляет файлы в промежуточную область для фиксации. `git commit`: фиксирует изменения в локальном репозитории. `git status`: Проверяет текущий статус репозитория. `git init`: инициализирует новый локальный репозиторий Git. ветка `git`: создает ветки в репозитории и управляет ими. Удаленный репозиторий: `git clone`: клонирует существующий репозиторий из удаленного источника. `git push`: отправляет изменения в удаленный репозиторий. `git pull`: извлекает изменения из удаленного репозитория. 9) Что такое и зачем могут быть нужны ветви (branches)? Разделение проектов Позволяют разработчикам работать над различными функциями или исправлениями ошибок, не затрагивая основной код. Ветки также можно использовать для экспериментов с новыми идеями или тестирования новых функций, не влияя на основной проект. Это упрощает отслеживание изменений и сотрудничество с другими разработчиками.

- 10) Как и зачем можно игнорировать некоторые файлы при `commit`? `.gitignore`. Эта команда указывает, какие файлы следует игнорировать при фиксации изменений. Это может быть полезно для игнорирования файлов, которые не нужно отслеживать, таких как временные файлы или файлы конфигурации. Его также можно использовать для предотвращения случайной фиксации конфиденциальной информации, такой как пароли.

4 Выводы

Благодаря данной лабораторной работе я приобрела практические навыки работы со средствами контроля версий git, изучила идеологию и применение данных средств ::: {#refs} :::