

Лабораторная работа 2

Настройка git

Ерёмин Даниил

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	11

Список иллюстраций

2.1	создание ключа pgr	6
2.2	ключи на гитхаб	7
2.3	набор команд для настройки автоматических подписей коммитов гит	7
2.4	создание репозитория	7
2.5	настройка репозитория	8

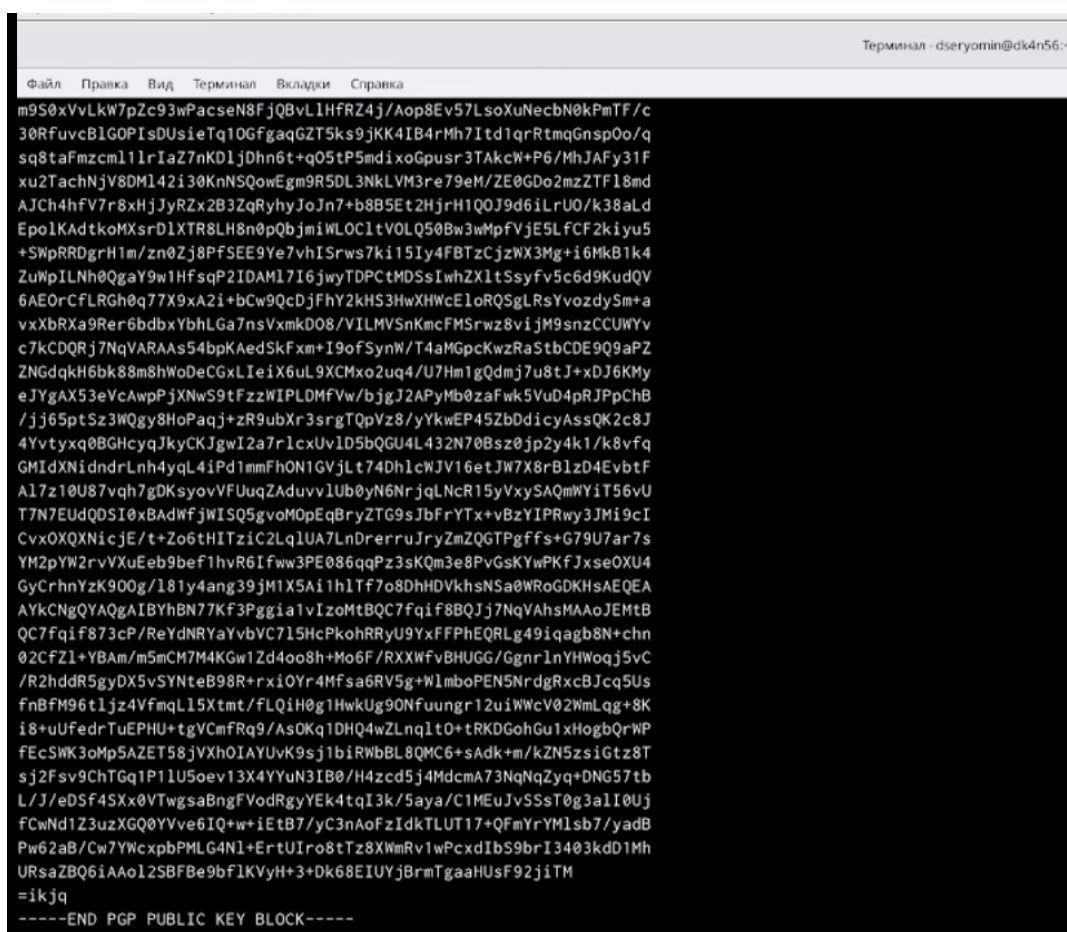
Список таблиц

1 Цель работы

Изучить идеологию и применение средств контроля версий. Освоить умения по работе с git.

2 Выполнение лабораторной работы

- 1) создадим ключ pgr и загрузим его на гитхаб (рис. -2.1)



```
Терминал - dseryomin@dk4n56:~  
Файл  Правка  Вид  Терминал  Вкладки  Справка  
m9S0xVvLkK7pZc93wPacseN8FjQBvLlHFRZ4j/Aop8Ev57LsoXUecbN0kPmTF/c  
30RFuvclBGOPIsDUsieTq10GfgaqGZT5ks9jKK4IB4rMh7Itld1qrRtmqGnspOo/q  
sq8taFmzcm11lrIaZ7nKD1jDhn6t+q05tP5mdixGpusr3TAkcW+P6/MhJAFy31F  
xu2TachNjV8DM142i30KnNSQowEgm9R5DL3NkLVM3re79eM/ZE0GDo2mz2TF18md  
AJCh4hFV7r8xHjJyRZx2B3ZqRyhyJoJn7+b8B5Et2HjrH1Q0J9d6iLrU0/k38aLd  
EpolKAdtkoMXsrD1XTR8LH8n0pQbJmiWLOC1tVOLQ50Bw3wMpfVjE5LFCF2kiyu5  
+SWpRRDgrH1m/zn0Zj8PFSEE9Ye7vhISrws7ki15Iy4FBTzCjzWX3Mg+i6MkB1k4  
ZuWpILNh0QgaY9w1HfsqP2IDAM17I6jwyTDPCtMDSsIwhZX1tSsyfv5c6d9KudQV  
6AEOrCfLRGh0q77X9xA2i+bCw9QcDjFhY2kH53HwXHWcElORQSGLRsYvozdySm+a  
vxXbRXa9Rer6bdbxYbhlGa7nsVxmKD08/VILMVSnKmcFMSrWz8viJm9snzCCUWYv  
c7kCDQRj7NqVARAAs54bpKAedSkFxm+I9ofSynW/T4aMGpcKwzRaStbCDE9Q9aPZ  
ZNGdqkH6bk88m8hWoDeCGxLIEiX6uL9XCMxo2uq4/U7Hm1gQdmj7u8tJ+xDJ6KMy  
eJYgAX53eYcAwpjXNwS9tfzzWIPLDMfVw/bjgJ2APyMb0zaFwk5Vu04pRJPPChB  
/jj65ptSz3WQgy8HoPaqj+zR9ubXr3srgTQpVz8/yYkwEP45ZbDdicyAssQK2c8J  
4Yvtxyq0BGHcyqJkyCKJgwI2a7rlcxUv1D5bQGU4L432N70Bsz0jp2y4k1/k8vfq  
GMiXndrLn4yqL4iPd1mmFhON1GVjL74DhlcWJV16etJW7X8rBlzD4EvbtF  
A17z10U87vqh7gDKsyovVFUuqZAduvv1Ub0yN6NrqLNCr15yVxySAQmWYiT56vU  
T7N7EUDQDSi0xBadWfjWISQ5gvoMOpEqBryZTG9sJbFrYTx+vBzYIPrwy3JMi9cI  
CvxOXQXNcicJE/t+Zo6tHITziC2Lq1UA7LnDrerruJryZmZQGTpgffs+G79U7ar7s  
YM2pYW2rvVXuEeb9bef1hvr6Ifww3PE086qqPz3sKQm3e8PvGsKYwPKfJxseOXU4  
GyCrhnYzK900g/181y4ang39jM1X5Ai1h1Tf7o8DhHDVkhNSa0WROGDKHsAEQEA  
AYkCNgQYAQgAIBYhBN77Kf3Pggia1vIzoMtBQC7fqiF8BQJj7NqVAhsMAAoJEMtB  
QC7fqiF873cP/ReYdNRYaYvbVC7L5HcPkohRRyU9YxFFPhEQLg49iqagb8N+chn  
02CFZ1+YBAM/m5mCM7M4KGw1Zd4oo8h+Mo6F/RXXWfvBHUGG/GgnrInYHwoqj5vC  
/R2hddR5gyDX5vSYNteB98R+rxioYr4Mfsa6RV5g+Wlmb0PEN5NrdgRxcBJcq5Us  
fnBFM96t1jz4VfmqL15Xtmt/FLQIH0g1HwkUg9ONFuungr12uiWWcV02WmLqg+8K  
i8+uUfedrTuEPHU+tgVcmFRq9/AsOKq1DHQ4wZLnq1tO+trKDGohGu1xHogbQrWP  
FEcSWK3oMp5AZET58jVXhOIAyUvK9sj1biRWbBL8QMC6+sAdk+m/kZN5zsiGtz8T  
sJ2Fsv9ChTGq1P11U5oev13X4YYuN3IB0/H4zcd5j4MdcMA73NqNqZyq+DNG57tb  
L/J/eDSF4SXx0VTwsaBngFVodRgyYEK4tqI3k/5aya/C1MEuJvSSST0g3aII0Uj  
FCwNd1Z3uzXGQ0YVve6IQ+w+iEtB7/yC3nAoFzIdkTLUT17+QFmYrYMLsb7/yadB  
Pw62aB/Cw7YwCxbPMLG4N1+ErtUIro8tTz8XWmRv1wPcxdIb59brYI3403kdD1Mh  
URsaZBQ6iAAo12SBFB9bf1KVyH+3+Dk68EUYjBrmTgaaHUsF92jiTM  
=ikjq  
-----END PGP PUBLIC KEY BLOCK-----
```

Рис. 2.1: создание ключа pgr

(рис.-2.2)

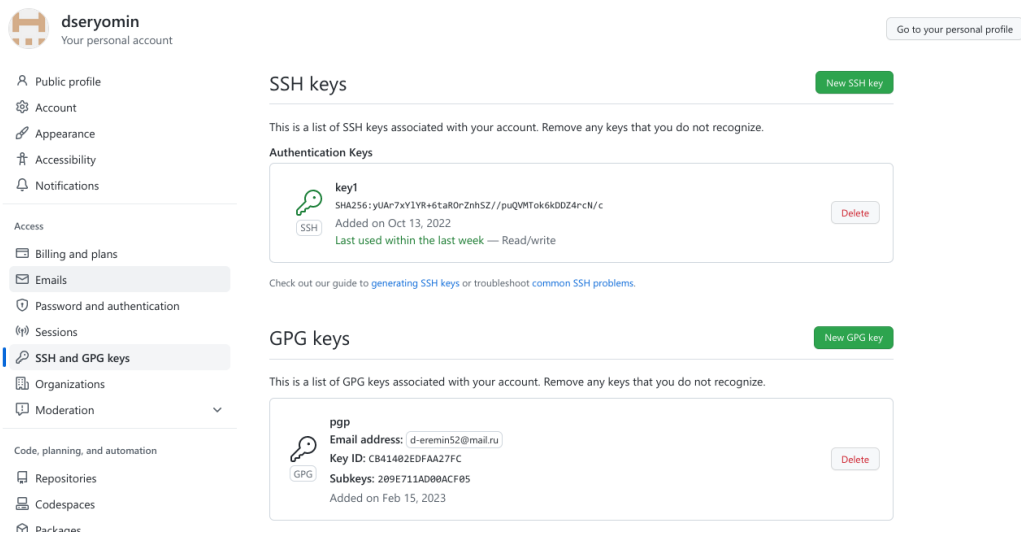


Рис. 2.2: ключи на гитхаб

2) Настроим автоматические подписи коммитов git (рис. -2.3)

```
dseryomin@dk4n56 ~/work/study/2022-2023/Операционные системы $ git config --global user.signingkey DEFB29FDCF82089AD6F233A0CB41402EDFAA27FC
dseryomin@dk4n56 ~/work/study/2022-2023/Операционные системы $ git config --global commit.gpgsign true
dseryomin@dk4n56 ~/work/study/2022-2023/Операционные системы $ git config --global gpg.program $(which gpg2)
```

Рис. 2.3: набор команд для настройки автоматических подписей коммитов гит

3) Создадим репозиторий курса на основе шаблона (рис. -2.4)

```
dseryomin@dk4n56 ~/work/study/2022-2023/Операционные системы $ gh repo create study_2022-2023-os-intro --template=yamadharma/course-directory-student-template --public
GraphQL: Could not clone: Name already exists on this account (cloneTemplateRepository)
dseryomin@dk4n56 ~/work/study/2022-2023/Операционные системы $ gh repo create study_2022-2023-os-intro --template=yamadharma/course-directory-student-template --public
Created repository dseryomin/study_2022-2023-os-intro on GitHub
dseryomin@dk4n56 ~/work/study/2022-2023/Операционные системы $ git clone --recursive git@github.com:dseryomin/study_2022-2023-os-intro.git os-intro
Клонирование в «os-intro»...
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
Получение объектов: 100% (27/27), 16.93 КнБ | 16.93 МнБ/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 «/afs/.dk.sci.pfu.edu.ru/home/d/s/dseryomin/work/study/2022-2023/Операционные системы/os-intro/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
Получение объектов: 100% (82/82), 92.98 КнБ | 786.00 МнБ/с, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в «/afs/.dk.sci.pfu.edu.ru/home/d/s/dseryomin/work/study/2022-2023/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 101, done.
remote: Counting objects: 100% (101/101), done.
remote: Compressing objects: 100% (78/78), done.
remote: Total 101 (delta 40), reused 88 (delta 27), pack-reused 0
Получение объектов: 100% (101/101), 327.25 КнБ | 1.69 МнБ/с, готово.
Определение изменений: 100% (40/40), готово.
Submodule path 'template/presentation': checked out 'b1be380ee91f5809264cb755d316174540b763e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a83917b32e3aef11a33b1e3b2'
dseryomin@dk4n56 ~/work/study/2022-2023/Операционные системы $
```

Рис. 2.4: создание репозитория

4) Настройка каталога курса и отправка файлов на сервер (рис. -2.5)

```
dseryomin@dk4n56 ~/work/study/2022-2023/Операционные системы/os-intro $ rm package.json
dseryomin@dk4n56 ~/work/study/2022-2023/Операционные системы/os-intro $ echo os-intro
os-intro
dseryomin@dk4n56 ~/work/study/2022-2023/Операционные системы/os-intro $ echo os-intro > COURSE
dseryomin@dk4n56 ~/work/study/2022-2023/Операционные системы/os-intro $ make
dseryomin@dk4n56 ~/work/study/2022-2023/Операционные системы/os-intro $ git add .
dseryomin@dk4n56 ~/work/study/2022-2023/Операционные системы/os-intro $ git commit -am 'feat(main): make course structure'
```

Рис. 2.5: настройка репозитория

```
dseryomin@dk4n56 ~/work/study/2022-2023/Операционные системы/os-intro $ git push
Перечисление объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 343.04 КиБ | 2.74 МиБ/с, готово.
Всего 38 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано п
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:dseryomin/study_2022-2023_os-intro.git
b02cede..54d19d3 master -> master
```

(рис. -??)

Контрольные вопросы: 1). Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` различными опциями. Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. 2). 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 неполную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. 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.

3). Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. Пример - Wikipedia. В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. Пример — Bitcoin. В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером.

4). Создадим локальный репозиторий. Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория: `git config --global user.name "Имя Фамилия"` `git config --global user.email "work@mail"` и настроив utf-8 в выводе сообщений: `git config --global path false`. Для инициализации локального репозитория, расположенного, например, в каталоге `~/tutorial`, необходимо ввести в командной строке: `cd mkdir tutorial cd tutorial git init`

5). Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый): `ssh-keygen -C "Имя Фамилия work@mail"`. Ключи сохраняются в каталоге `~/.ssh/`. Скопировав из локальной консоли ключ в буфер обмена `cat ~/.ssh/id_rsa.pub | xclip -sel clip` вставляем ключ в появившееся на сайте поле.

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

7). Основные команды git: Наиболее часто используемые команды 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 изменений: `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 имена_файлов` – удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): `git rm имена_файлов` – сохранение добавленных изменений: – 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). 8). Использование `git` при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий): `git add hello.txt` `git commit -am 'Новый файл'` 9). Проблемы, которые решают ветки `git`: нужно постоянно создавать архивы с рабочим кодом сложно “переключаться” между архивами сложно перетаскивать изменения между архивами легко что-то напутать или потерять 10). Во время работы над проектом так или иначе могут создаваться файлы, которые не требуются добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл `.gitignore` с помощью `е`рвисов. Для этого сначала нужно получить списки имеющихся шаблонов: `curl -L -s https://www.gitignore.io/api/list` Затем скачать шаблон, например, для C и C++ `curl -L -s https://www.gitignore.io/api/c » .gitignore` `curl -L -s https://www.gitignore.io/api/c++ » .gitignore`

3 Выводы

В рамках данной лабораторной работы я изучил идеологию и применение средств контроля версий. Освоил умения по работе с git.