

Отчёт по лабораторной работе №2

дисциплина: Операционные системы

Давыдов Сергей

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Контрольные вопросы	16
5	Выводы	19

Список иллюстраций

3.1	Базовая настройка git	9
3.2	Создание ключа SSH	9
3.3	Ключ SSH создан	10
3.4	Ключ GPG создан	10
3.5	Отпечаток приватного ключа	11
3.6	Настройка подписей	12
3.7	Создание репозитория	13
3.8	Настраиваем каталог курса	14
3.9	Отправляем наши файлы на сервер	14
3.10	Отправляем наши файлы на сервер	15

Список таблиц

1 Цель работы

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

2 Задание

1. Создать базовую конфигурацию для работы с git.
2. Создать ключ SSH.
3. Создать ключ PGP.
4. Настроить подписи git.
5. Зарегистрироваться на Github.
6. 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 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. # Выполнение лабораторной работы

Базовая настройка git:

1. Задаём имя и email владельца репозитория (1 и 2 строка на рисунке)
2. Настраиваем utf-8 в выводе сообщений git (3 строка на рисунке)
3. Настраиваем верификацию и подписание коммитов git. Зададим имя начальной ветки (будем называть её master) (4 строка на рисунке)


```
141 history
sadavihdov@dk4n62 ~ $ git config --global user.name "plugikx"
sadavihdov@dk4n62 ~ $ git config --global user.email "ser.dav2006@gmail.com"
sadavihdov@dk4n62 ~ $ git config --global core.quotepath false
sadavihdov@dk4n62 ~ $ git config --global core.quotepath false
^C^C
^Cc^C^Ccc^C
sadavihdov@dk4n62 ~ $ git config --global core.quotepath false
sadavihdov@dk4n62 ~ $ git config --global init.defaultBranch master
sadavihdov@dk4n62 ~ $ git config --global core.autocrlf input
sadavihdov@dk4n62 ~ $ git config --global core.safecrlf warn
sadavihdov@dk4n62 ~ $
```

Рис. 3.1: Базовая настройка git

Создаём ключ SSH. В терминале вводим данную команду:

```
ssh-keygen -t rsa -b 4096
```

Далее во всех пунктах пользуемся клавишей Enter и получаем наш ключ.

```
sadavihdov@dk4n62 ~ $ git config --global core.quotepath false
sadavihdov@dk4n62 ~ $ git config --global init.defaultBranch master
sadavihdov@dk4n62 ~ $ git config --global core.autocrlf input
sadavihdov@dk4n62 ~ $ git config --global core.safecrlf warn
sadavihdov@dk4n62 ~ $
```

Рис. 3.2: Создание ключа SSH

Ключ нужно добавить на github. Для этого переходим на сайте в раздел “Settings” и выбираем “SSH and GPG keys”.

```

sadavihdov@dk4n62 ~ $ gpg --full-generate-key
gpg (GnuPG) 2.2.42; Copyright (C) 2023 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Выберите тип ключа:
  (1) RSA и RSA (по умолчанию)
  (2) DSA и Elgamal
  (3) DSA (только для подписи)
  (4) RSA (только для подписи)
  (14) Имеющийся на карте ключ
Ваш выбор? 1
длина ключей RSA может быть от 1024 до 4096.
Какой размер ключа Вам необходим? (3072) 4096
Запрошенный размер ключа - 4096 бит
Выберите срок действия ключа.
  0 = не ограничен
  <n> = срок действия ключа - n дней
  <n>w = срок действия ключа - n недель
  <n>m = срок действия ключа - n месяцев
  <n>y = срок действия ключа - n лет
Срок действия ключа? (0) 0
Срок действия ключа не ограничен
Все верно? (y/N) y

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: plugikx
Адрес электронной почты: ser.dav2006@gmail.com
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "plugikx <ser.dav2006@gmail.com>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? o
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
gpg: создан каталог '/afs/.dk.sci.pfu.edu.ru/home/s/a/sadavihdov/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/afs/.dk.sci.pfu.edu.ru/home/s/a/sadavihdov/.gnupg/openpgp-revocs.d/2029
9022123984043B7AD864BF6C7.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2024-03-01 [SC]
      2029CA7B849E9209D22123984043B7AD864BF6C7
uid    plugikx <ser.dav2006@gmail.com>
sub    rsa4096 2024-03-01 [E]

sadavihdov@dk4n62 ~ $

```


Рис. 3.3: Ключ SSH создан

SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication keys



Sergei

SHA256:SPncId92e4f137E0/sOKTCcNxmFXTLmjB7Bh5oXMkeE

Added on Nov 23, 2023

Last used within the last 4 months — Read/write

Delete



Рис. 3.4: Ключ GPG создан

Выводим список ключей и копируем отпечаток приватного ключа

GPG keys

[New GPG key](#)

This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.



password

Email address: ser.dav2006@gmail.com Unverified

Key ID: 4043B7AD864BF6C7

Subkeys: EB76EC135CDA79A8

Added on Mar 1, 2024

[Delete](#)

Learn how to [generate a GPG key and add it to your account](#).

Рис. 3.5: Отпечаток приватного ключа

Настройка автоматических подписей коммитов git

```
NtkZHyrYSqknXvgttDvdqpPUqA7xhqkrWgGm5ByzaliZDcxGxzX2yNE11uNCvtCx
fX0HBsXUuq9B9LagNuicB0M5YXTPxDXExc1RQxmVv66/hSD4kDUibV1x1q331eFQ
7E9D7JX0SFdn+NwYaRyGtzzzzHbie6o2ec4JBsJwmcSfo6Ya0sSmkxPC3PXsCYlg
eKyXqwdW1QA1raEJEpHE0XnT2zrOcWPlRQ11JV1UUpok1/NGdT+ksU9uhJmi8N7
Ts0cCE8MGCFFA6B1JM4HAhy0HY4ORlPnRmY60mZcPfm/GouTSpuYdf1W+LKwQsRL
ACmSyFvJrTcfJ8ReaxzYJSzSNCqPikF91fow3r5U3IKSJUztUs4CHzpScbFJdoDC
SKIZ9bf/UxLH5ncWrbp2WsZ5+spe26FnGcsQuoel8+mWDiu7UuBh6s1tUNKYJMK4
Dvn3P4DSJD0FPAJW/pq+AYMo+tr4/xNO2twJvyhMyqYv4zvIJTVdiRLNbB5N2o4X
1/lwEkr24MKvWDAaXo1TRGURUM+cBiXxVhDDZG56Co9sM46By6z3RcsNBwARAQAB
tB9wbHvNawt4IDxzZXIuZGF2MjAwNkNbnWFpbC5jb20+iQJ0BBMBCAA4FiEEICnK
e4S0kgN5IS0YQE03rYZL9scFAMXh0/4CGwMFCwkIBWIGFQoJCA5CBBYCAwECHgEC
F4AACgkQQE03rYZL9sdWlxAAlXPK1qRqmfo5BoM4QT7s3ZxFS1UE8h/icJSGRHu
wqz0BaShMqnPpznLWKwSfYmTiFQ3DS7UEfSQzk8kcaaXMULeKttrFHVj70klu0Jh
iHwV09v0kgFWH5jAGYDJE0JB665Nd+zhBMHqHf+jAjHlRofFvYZQ/pw5BNaaXRei
bWpPBRXZhn/Nsdt6YVDF0aSU/0rcrQHXLj67RhqvjkH/bozofG3yaL3MxDkZ7E
3/2VtpKfGrP+8p3wI29qUIZxLG1J0oUZqzwsUjaF1KyK2tltFFMwcBn714yQ0JS9
e0PndCj9sIzkSDnvihoBtu+N1piXaR3/e7yLowXGCIpLjor7vq6qdiHuErqk8T+5
H62oq51o8/dmAp/wnEw0jEsj0w70Xj661810rEvxNxFrZXeIdUuqxw7kzcN3EzPQ
fdUvSLyyvi3fiX/FscP9IABeify0US6XcSHRMjNMPdFr2zkPxAUy/9ADDTgmfh
BHUX7x5+KsnzDSUR5PY89jCADdepF29jBz646E7VqbpAJ4vK2F1jzPz2qjflBz3
LYpYce6uLXinRXELqKXfrqc8IcHnj6Q9MUZF4XEury12YFJt5fIlLg9lNjNMH
GnANmwQ6LiXR9sNDH1oWFs215TnlN8FZ/LJvcRUyWuygu5g1UpkedQ/D6Fbtqt/
eRy5Ag0EZeHT/gEQAK2SZFXHeCyMkVTuICBNBesefN0QPjplHIAPEwc3c550iEvB
Lhh0YAnvFPoqs/KCnc/TEIGYrp0GqH584eN/OM91l61YRnroMZWzVF/sZ882BN+G
KZGzTOX3ojg4GRg4GZU6APyJ13o03ASr+KkZROdnXYsZCtQykuh7TYQP+8bMi+h
HWCmp0lMuRnOEwVUDSnpgpDoSAUdiRjdpT85QKgd9Hzf7fw9p07yeEwAvJujkw7Y
UEIf53f3VTdnPONG/kAm7PMF90Jem6x0h7LND+GD5/1Vilw27nvAe4CXHUFVt6uN
dXrKkMXV26nhlg1glkUBolzSeb8fCh9Aa7NXTFFNSgxhBoQVijZqjFrVG0v2qgx5
mxLw5gkjXoya4h0f+UbKb1nGpKvtKKHzNC7LYxigQgoAr5ISmLQ7Yk9JHutIDmhi
nmrW/gObRfTPeZJF53iMKy/jw2QBFTKfGkK16JeomVq6Ty7UDxsfgZaC2YsJEzry
KR7pSLP972ymW/FGZ0f50VCJVEgz0omocHsbl0meVKVufnvHhwKsUJ07QaSo968
LnWIbn13A9pDDc1EKKPdcdu18GzQHSEQc07ewdD97YMXgrAzJJxsGyEIGcKZ19L
AMBseOZUazpIHnCOcclBMKQ8s74KU9LQgqXQKbBWhXWSKcdj2keZKT5XiYpABEB
AAGJAjYEGAEIACAWIQQgKcp7hI6ScDIhI5hAQ7ethkv2xwUCZeHT/gIbDAACRBA
Q7ethkv2x8eBD/9GQgAJG1G8707u/etejrv+6LK6RmFqXoF22wbAt9MQP0FQ1fn7
1hkwtMG0Znu7tEqD8NA01vkdjuINECyAZXgbY6xCjw2z5XzA8d16sCyB2Ywds6km
4R9sc+Zbs00JScmTFXYZgLhC8Lw3sDGMlQ9B54FdzkRzFw01Ejsh59oJ2EWhdTXP
R1q80ZpsOK+N1vAPe58D3xmc1kWB8pWgmmtXEG9VdV6VwfoRo7Hp40rhVvLFfNP
f5eRAhzbA199jMn5XXeSPbmxsRX1oIMSnXjsCfrWwEX3/Os+UzJ1E1SkNRky5yrx
huedZ5k42nvBiM7b1oKj8zhmI2GKIxhStlP2mW0wCCOXLItrX/GQGwbaBEJk0/Hw
qtLMZSILSPWQEao9ILO/JnT+q+bYA60NbM7ux6pDWPtFa8ZU1QUhucRibkIqDt0q
pJkh9K3+pKilVaBoH+La8GQo/XWfcWfP0C9Wm70YjHkh7qBdHOqnCJWtSbg6E0BR
hbT/jMu+BQNUiHICTOCBryOdv6E0xGx1+H7q8Udt+Brk1G1MKXZ34xtN2Qff1M1X
hHRUGmr02KiaSe9Tx0o4fdwo+c20SE25cD2yJidPaQ9AxkrRZYIlMqfLJK+ut4nX
3kHxw3ug/8ZpXzY2K0qYQ8+Y5hEZ7HuKC9HPuJDtMREzgJ+AEUbYdLEdKQ==
```

Рис. 3.6: Настройка подписей

Возвращаемся в наш терминал и настраиваем gh командой:

gh auth login.

Во всех пунктах выбираем у(yes).

По полученной ссылке переходим в браузер на виртуальной машине и вводим код из терминала (находится перед ссылкой).

```
=a+/7
-----END PGP PUBLIC KEY BLOCK-----
sadavihdov@dk4n62 ~ $ git config --global user.signingkey 4043B7AD864BF6C7
sadavihdov@dk4n62 ~ $ git config --global commit.gpgsign true
sadavihdov@dk4n62 ~ $ git config --global gpg.program $(which gpg2)
```

#fig:007

width=70% }

Создаём репозиторий курса на основе шаблона. Все нужные команды для создания были в указаниях к лабораторной работе. В 4 команде, вместо , указываем своё имя профиля на github.

1. `mkdir -p ~/work/study/2021-2022/“Операционные системы”`
2. `cd ~/work/study/2021-2022/“Операционные системы”`
3. `gh repo create study_2021-2022_os-intro --template=yamadharma/course-directory-student-template --public`
4. `git clone --recursive git@github.com:/study_2021-2022_os-intro.git os-intro`

```
sadavihdov@dk4n62 ~ $ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /afs/.dk.sci.pfu.edu.ru/home/s/a/sadavihdov/.ssh/id_rsa.pub
? Title for your SSH key: Sergei
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: B483-544F
Press Enter to open github.com in your browser...
! Failed opening a web browser at https://github.com/login/device
exec: "/mnt/c/Windows/explorer.exe": stat /mnt/c/Windows/explorer.exe: no such file or directory
Please try entering the URL in your browser manually
✓ Authentication complete.
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
! Authentication credentials saved in plain text
✓ SSH key already existed on your GitHub account: /afs/.dk.sci.pfu.edu.ru/home/s/a/sadavihdov/.ssh/id_rsa.pub
Logged in as plugiky
```

Рис. 3.7: Создание репозитория

Настраиваем каталог курса. Для этого переходим в него командой:

```
cd ~/work/study/2021-2022/“Операционные системы”/os-intro
```

Далее командой `ls` проверяем, что мы в него перешли. В каталоге “os-intro” нам потребуется удалить файл “package.json”. Выполняем данную задачу командой:

```
rm package.json
```

Снова командой `ls` проверяем успешное выполнение удаления файла.

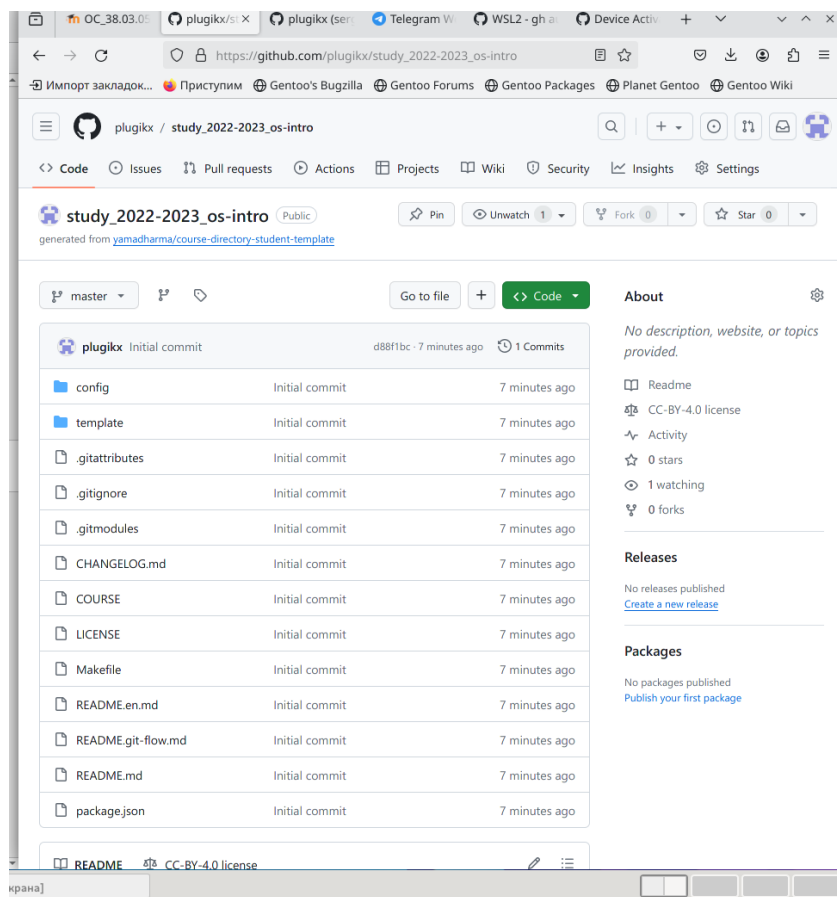


Рис. 3.8: Настраиваем каталог курса

Создаём необходимые каталоги и отправляем наши файлы на сервер
make COURSE=os-intro

1. git add .
2. git commit -am 'feat(main): make course structure'
3. git push

```

ределение изменений: 100% (52/52), готово.
bmodule path 'template/presentation': checked out '40a1761813e197d00e8443ff1ca72c60a304f24c'
bmodule path 'template/report': checked out '7c31ab8e5d5fa8c6b2d67caeb8a19ef8028ced88e'
davihdov@dk4n62: ~/work/study/2022-2023/Операционные системы $ ls
udy_2022-2023_os-intro
davihdov@dk4n62: ~/work/study/2022-2023/Операционные системы $ cd study_2022-2023_os-intro
davihdov@dk4n62: ~/work/study/2022-2023/Операционные системы/study_2022-2023_os-intro $ rm package.json
davihdov@dk4n62: ~/work/study/2022-2023/Операционные системы/study_2022-2023_os-intro $ echo os-intro > COURSE
davihdov@dk4n62: ~/work/study/2022-2023/Операционные системы/study_2022-2023_os-intro $ make prepare
davihdov@dk4n62: ~/work/study/2022-2023/Операционные системы/study_2022-2023_os-intro $ git add .
davihdov@dk4n62: ~/work/study/2022-2023/Операционные системы/study_2022-2023_os-intro $ git commit -am 'feat(main):
make course structure'
[aster 4d46851] feat(main): make course structure
61 files changed, 29433 insertions(+), 14 deletions(-)

```

Рис. 3.9: Отправляем наши файлы на сервер

```

create mode 100644 project-personal/stage5/presentation/image/kulyabov.jpg
create mode 100644 project-personal/stage5/presentation/presentation.md
create mode 100644 project-personal/stage5/report/Makefile
create mode 100644 project-personal/stage5/report/bib/cite.bib
create mode 100644 project-personal/stage5/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage5/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100755 project-personal/stage5/report/pandoc/filters/pandoc_eanos.py
create mode 100755 project-personal/stage5/report/pandoc/filters/pandoc_fignos.py
create mode 100755 project-personal/stage5/report/pandoc/filters/pandoc_secnos.py
create mode 100755 project-personal/stage5/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 project-personal/stage5/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 project-personal/stage5/report/pandoc/filters/pandocxnos/core.py
create mode 100644 project-personal/stage5/report/pandoc/filters/pandocxnos/main.py
create mode 100644 project-personal/stage5/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 project-personal/stage5/report/report.md
create mode 100644 project-personal/stage6/presentation/Makefile
create mode 100644 project-personal/stage6/presentation/image/kulyabov.jpg
create mode 100644 project-personal/stage6/presentation/presentation.md
create mode 100644 project-personal/stage6/report/Makefile
create mode 100644 project-personal/stage6/report/bib/cite.bib
create mode 100644 project-personal/stage6/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage6/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_eanos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_fignos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_secnos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/core.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/main.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 project-personal/stage6/report/report.md
sadavihdov@dk4n62 ~/work/study/2022-2023/Операционные системы/study_2022-2023_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% (30/30), 342.11 Киб | 13.16 МБ/с, готово.
Всего 30 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:plugix/study_2022-2023_os-intro.git
   d88f1bc..4d46851 master -> master
sadavihdov@dk4n62 ~/work/study/2022-2023/Операционные системы/study_2022-2023_os-intro $

```

Рис. 3.10: Отправляем наши файлы на сервер

4 Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Это программное обеспечение для облегчения работы с изменяющейся информацией. VCS 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.
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Хранилище (repository), или репозиторий, — место хранения всех версий и служебной информации. Commit («[трудовой] вклад», не переводится) — синоним версии; процесс создания новой версии. История — место, где сохраняются все коммиты, по которым можно посмотреть данные о коммитах. Рабочая копия — текущее состояние файлов проекта, основанное на версии, загруженной из хранилища.
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные VCS: одно основное хранилище всего проекта и каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно. Децентрализованные VCS: у каждого пользователя свой вариант (возможно не один) репозитория.
4. Опишите действия с VCS при единоличной работе с хранилищем.
5. Опишите порядок работы с общим хранилищем VCS.

6. Каковы основные задачи, решаемые инструментальным средством git?
Git — это система управления версиями. У Git две основных задачи: первая — хранить информацию о 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. git –version (Проверка версии Git) git init (Инициализировать ваш текущий рабочий каталог как Git-репозиторий) git clone <https://www.github.com/username/repo-name> (Скопировать существующий удаленный Git-репозиторий) git remote (Просмотреть список текущих удалённых репозиториях Git) git remote -v (Для более подробного вывода) git add my_script.py (Можете указать в команде конкретный файл). git add . (Позволяет охватить все файлы в текущем каталоге, включая файлы, чье имя начинается с точки) git commit -am “Commit message” (Вы можете сжать все индексированные файлы и отправить коммит). git branch (Просмотреть список текущих веток можно с помощью команды branch) git –help (Чтобы узнать больше обо всех доступных параметрах и командах) git push origin master (Передать локальные коммиты в ветку удаленного репозитория).
8. Приведите примеры использования при работе с локальным и удалённым репозиториями.
9. Что такое и зачем могут быть нужны ветки (branches)? Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций: чтобы не повредить основному проекту, создается новая ветка специально для экспериментов.
10. Как и зачем можно игнорировать некоторые файлы при commit? Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы,

которые по какой-либо иной причине не должны попадать в коммиты.

5 Выводы

В ходе выполнения лабораторной работы изучили идеологию и применение средств контроля версий, а также освоили умения по работе с git.