

Отчёт по лабораторной работе №2

дисциплина: Операционные системы

Курникова Елизавета

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Контрольные вопросы	14
5	Выводы	17

Список иллюстраций

3.1	Базовая настройка git	8
3.2	Создание ключа SSH	9
3.3	Ключ SSH создан	9
3.4	Ключ GPG создан	9
3.5	Отпечаток приватного ключа	10
3.6	Настройка подписей	10
3.7	Создание репозитория	12
3.8	Настраиваем каталог курса	12
3.9	Отправляем наши файлы на сервер	13

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)

```
edkurnikova@dk3n40 ~$ git config --global user.name "elizavetakurnikovanibid-01-23"
edkurnikova@dk3n40 ~$ git config --global user.email "1132239095@pfur.ru"
edkurnikova@dk3n40 ~$ git config --global core.quotepath false
edkurnikova@dk3n40 ~$ git config --global init.defaultBranch master
edkurnikova@dk3n40 ~$ git config --global core.autocrlf input
edkurnikova@dk3n40 ~$ git config --global core.safecrlf warn
```

Рис. 3.1: Базовая настройка git

Создаём ключ SSH. В терминале вводим данную команду:

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

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”.

Рис. 3.3: Ключ SSH создан

Рис. 3.4: Ключ GPG создан

Выводим список ключей и копируем отпечаток приватного ключа

```

edkurnikova@dk3n40 ~$ gpg --armor --export 13BA924E39C7ACFB7B4B2428584387369E38D28
-----BEGIN PGP PUBLIC KEY BLOCK-----

m0INBGXdmkBEACcBGga+L7n+4769MGPdx5k+HnJBGaQh/pWqITkr7e71K9Zkww
23U4xclLwLnJi1tCmPJJP0J03Z00fG073Ybd4gY7+V1Pdcl1fDSEF+V2FUUj/
G4GRfm89yoQfAYKd+mzJ1x2y0kzXfFy+TieeoafkBRbokuCs1j19C951NpUj/
ZgWxhts4Z9B9ROSesQ0SUAaBpzvV0ATcmKP/fPaMJ3RzhrrEFZbPbG5cQ08KDP
1XK8RByaW57T3CdftavMLYGGPdjdr+/FH2A6dcokBf1Wc8MZWkowZHO5Qj5HB1U
C4g0l3r35t9UjaU3H6nvIMhELj+Q8qh31lP0lp/e56ef2q8A4h69+m9pDl1X0kXq
t4o4m1sAP3CcXelaj3gFEPdtdYeRcq0mwa8UJ6ZwJobb/tottp19Fb1fnXpW8
4JLS7j1vtrU4jAO2/vI4D7MY8pujgYyUw8FzH3T2XYsr4/LBRhSeF9NscYn7xa
8gI9tBFyNSWNk3w8yLjXvWwDge7UcrG6BMej7rvuuMNQITebHPk5sT4pZ0lhgCK
Tud57B3s+82tIuwTiYcBH6Tw801bddyLQwd5tgzV0ws19Ewj6HpG3VktcB6Y+Az
eAk+IN2XySuqy/6ctBn66UjQWZtS8aU42Ys9+nxFBII1Q10FFsxB6GIQARAQAB
tDJ1bG16YXZldGFrZXJuaWtvdmlFyYml1ZC0wMS0yMyA8MTEzOTAwNUBwZnVy
LnJ1PokCTgQTAg0BYhBB06kk45x/rPt7SyQoWEOHnp440oBQj13c5pAhs0BQsJ
CacCBhUKCQgLAgQWAgMBAh4BAheAAoJETWEOHnp440oBQj13c5pAhs0BQsJ
64Pcsyvw+PQerZ17rkWqHh/k8THKXk16PYay4511Q3DcmtNe60ITHuXONJO1xk/
0w911JP/bGfu85qKZZVdQTmcSPctGKC2Pn/AxggFmeBHLhb2RjEr2Ho8p9j1naZe
vyZQuYAXWjvnmhK0rGpJHVfNO8Y+CBZ0tF172RPKX2k+8REJVM9zH2/Y1AV/TTZ
NWWo9I2MyjvzFpm5ebJpjjpgbFFymWF2PtVynn0GP1nBtNlgZ15cxPC4Yf07HS1
q1xursZvrFZhy30ba2YfQd5XL5npQvU7rjoogKuaweCTsdCwrcPDPUwdbFhkjPL
1mkZCZE/FeLj0nAoBdUTDAeT0Gz3ekQkLesnbBOH5mp09CjhtV0GXf9CmoNjQ
PnQw/c+GZ/PI50a6hkaKz+u+FRSSPDJTMxBP1YMsW04kpWV+jfTWJdghGPs1K4
MDC5LSL07Wb3uvRpXbkWuWdQZUERy707mqTF0uxCinQbLB0huhLJF471xHfWVJL9
pIIMhVcvb5RrFHLGyFrZBcJ99J0JK+NI+aoX0qWxAcB/70bWZH5bcuvpNd1++u
2i5V017bB4o7ZsJp853LHV5nMtIGfCuXfwqf1xRdSj110CvcFb90U1Gf08/4W
GnwDhr122FJRBUvgeEau1230UjgMuQINBGXdmkBEADrsn0fEYkr2s/M6w0wH1M
3xCCzQH0QDqCkYyVbVg/bD2g1W0iFt0oAZKFE40PQ2se1tR2+0mPYkeEGHr1M2
1j6Gj7jVghnh1D7BdCyXtZ1Y+bjg9oSdCmqFdN30qB0wH3L13839gZSG1Z8I7aa
FKv47bFdvNvUnfzyoWURCSyLUuAm22+rB4IcXPSVQq7BhouOVkTtw7cEJUmYqJ
3MIMJsYxygFe4uEC7qwpbxxvMXiLXfC8hGSjPwBGRS5j6KT0NDH/zveqm/cswPn
NiAaoSh3VcjEGLZkF4c/X0o1+2Hfm6DhIsDQ60TTP/pYb+jXEO/k0HfGr4u1CxdQ
d44LhJTLNLRsaOPvc64yqHtVUhfVwSvKmb6ZVDtQD1FC1dtUpwRICs8HtWepZG8
LERIKKXdaY1++dv30q540w4hh4ikMwFJWDYc0n9v1MwckU2npK1bR1hX7iRF6q4K
02PnZyBwUvKKNIeQcZ8gFACnSpzu500CRAC4BTsu0AcB7D0spjrwon2Vks39M
eMhDDjgDW10rOjXpEgmc/rzY3GwW0cxpWJxhngF7ITPELTkDSUdIy86KNhR1r
7MXADJt+EBidpXaFHAVXRwaQcJr9mmz46AKwu4AoNVChSjDcDdIunGmIEbYpW1
KVt08/8xR5Qs6q6PtjY+vwARAQABQI28BgBCAAgFIEE7qStJnH++3tLJChYQ4
c2njJgSfAMxdznkCGwACGkQH4YQ4c2njJgShqEG//cLJH/H63xxQXicMr51ydlKAZ
3wFigrGbe1QFV7F4UuVervQHlUah6advRabK615DF++nPW101qmqZ1Gc38A1IPXWb
SjPcnWrr5T/OAwHlZPyx3G7hNRwUTQI0CqCve51BV/5A7ARAs5qKoo3+69xwL.f
NrotOKK2NM1Jidu1CkMr6FEA0SunmGr/xxF1jkkZMgnP/qsNjHax0o1qhJavRVG
QqxqpnXFLw10mnoUHQ4hzVCAdz8Rbp82ZvPKn101taKVDW1iemwjbE68+FRKIu
oX/y5/ojKtNHK0z1239pcnns5xQrkd22tWKRpr6SPirJnpXKA0VEu6Kco2VLR

```

Рис. 3.5: Отпечаток приватного ключа

Настройка автоматических подписей коммитов git

```

1j6Gj7jVghnh1D7BdCyXtZ1Y+bjg9oSdCmqFdN30qB0wH3L13839gZSG1Z8I7aa
FKv47bFdvNvUnfzyoWURCSyLUuAm22+rB4IcXPSVQq7BhouOVkTtw7cEJUmYqJ
3MIMJsYxygFe4uEC7qwpbxxvMXiLXfC8hGSjPwBGRS5j6KT0NDH/zveqm/cswPn
NiAaoSh3VcjEGLZkF4c/X0o1+2Hfm6DhIsDQ60TTP/pYb+jXEO/k0HfGr4u1CxdQ
d44LhJTLNLRsaOPvc64yqHtVUhfVwSvKmb6ZVDtQD1FC1dtUpwRICs8HtWepZG8
LERIKKXdaY1++dv30q540w4hh4ikMwFJWDYc0n9v1MwckU2npK1bR1hX7iRF6q4K
02PnZyBwUvKKNIeQcZ8gFACnSpzu500CRAC4BTsu0AcB7D0spjrwon2Vks39M
eMhDDjgDW10rOjXpEgmc/rzY3GwW0cxpWJxhngF7ITPELTkDSUdIy86KNhR1r
7MXADJt+EBidpXaFHAVXRwaQcJr9mmz46AKwu4AoNVChSjDcDdIunGmIEbYpW1
KVt08/8xR5Qs6q6PtjY+vwARAQABQI28BgBCAAgFIEE7qStJnH++3tLJChYQ4
c2njJgSfAMxdznkCGwACGkQH4YQ4c2njJgShqEG//cLJH/H63xxQXicMr51ydlKAZ
3wFigrGbe1QFV7F4UuVervQHlUah6advRabK615DF++nPW101qmqZ1Gc38A1IPXWb
SjPcnWrr5T/OAwHlZPyx3G7hNRwUTQI0CqCve51BV/5A7ARAs5qKoo3+69xwL.f
NrotOKK2NM1Jidu1CkMr6FEA0SunmGr/xxF1jkkZMgnP/qsNjHax0o1qhJavRVG
QqxqpnXFLw10mnoUHQ4hzVCAdz8Rbp82ZvPKn101taKVDW1iemwjbE68+FRKIu
oX/y5/ojKtNHK0z1239pcnns5xQrkd22tWKRpr6SPirJnpXKA0VEu6Kco2VLR
JqTHjNagywPF0L5+79hR8mUk8Jk1XCxR+KvRLWe0UeZfZCTZ1fayz/Z2KhZVX0E
ELxZ+P6z+asZn9z+Rahiv2qR7u9hdWT6QLZRFQVZ3CUsOfnEWvx8Rao5NKyAgS2
C7z3oW6Kd354keedkoqDZ73YCOFeJbugF1JSFkMfNumB2MujfsAph6FrSVLAhRV
ZnWeyqlaRPCLSGXp6fQz550RxlqnvV80p61O+MoQGpUBcaZ1A2fzz+9FUeoLks
1PX0/7USrcx/JXSWFvx3OMYIz9oPF4IqZsc5QVeg/hsYVW7BA110NoLuryTRwBtz
n7dI0NgiIi52Fb5uEIs=
=ISaZ
-----END PGP PUBLIC KEY BLOCK-----

edkurnikova@dk3n40 ~$ git config --global user.signingkey 8584387369E38D28
edkurnikova@dk3n40 ~$ git config --global commit.gpgsign true
edkurnikova@dk3n40 ~$ git config --global gpg.program $(which gpg2)
edkurnikova@dk3n40 ~$ git 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/e/d/edkurnikova/.ssh/id_rsa.pub
? Title for your SSH key: GitHub CLI
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 1059-7E00
Press Enter to open github.com in your browser...

/ Authentication complete.
- gh config set -h github.com git_protocol ssh
/ Configured git protocol
/ Uploaded the SSH key to your GitHub account: /afs/.dk.sci.pfu.edu.ru/home/e/d/edkurnikova/.ssh/id_rsa.pub
/ Logged in as elizavetakurnikovanibd-01-23
! You were already logged in to this account
edkurnikova@dk3n40 ~$
edkurnikova@dk3n40 ~$

```

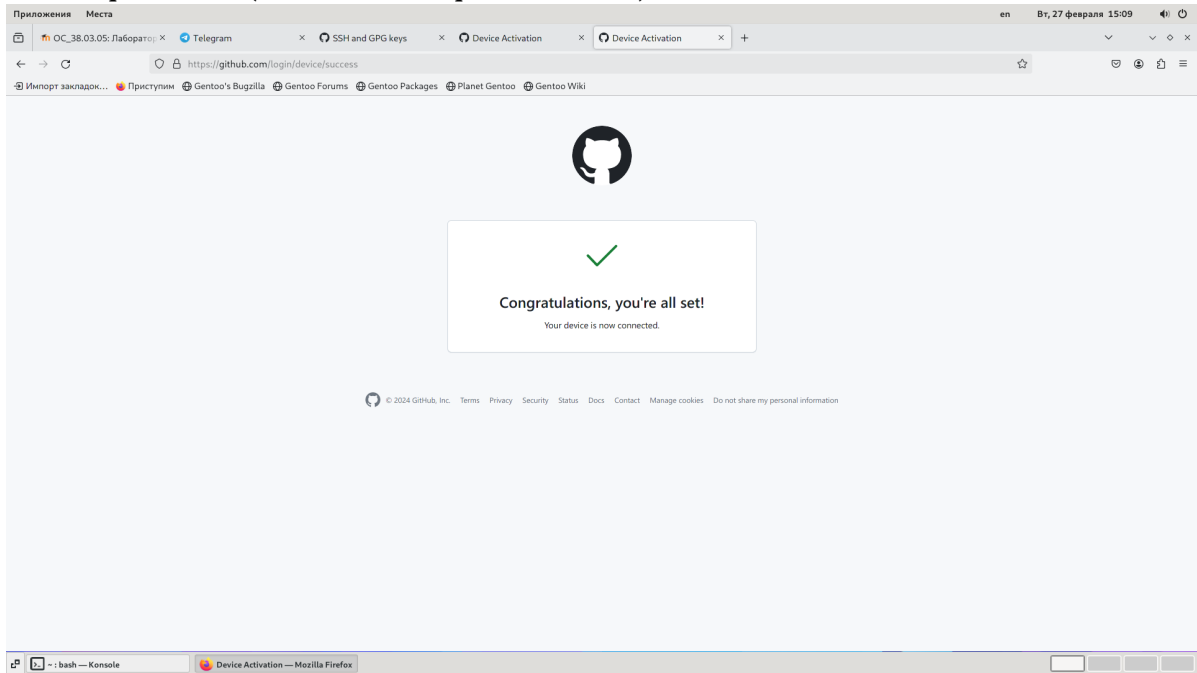
Рис. 3.6: Настройка подписей

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).



#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`

```

? Upload your SSH public key to your GitHub account? /afs/.dk.sci.pfu.edu.ru/home/e/d/edkurnikova/.ssh/id_rsa.pub
? Title for your SSH key: GitHub CLI
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 1059-7E00
Press Enter to open github.com in your browser...

/ Authentication complete.
- gh config set -h github.com git_protocol ssh
/ Configured git protocol
/ Uploaded the SSH key to your GitHub account: /afs/.dk.sci.pfu.edu.ru/home/e/d/edkurnikova/.ssh/id_rsa.pub
/ Logged in as elizavetakurnikovbibd-01-23
! You were already logged in to this account
edkurnikova@dk3n40 ~$
edkurnikova@dk3n40 ~$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
edkurnikova@dk3n40 ~$ cd ~/work/study/2022-2023/"Операционные системы"
edkurnikova@dk3n40 ~/work/study/2022-2023/Операционные системы $ gh repo create study_2022-2023-os-intro --template=y
/ Created repository elizavetakurnikovbibd-01-23/study_2022-2023-os-intro on GitHub
edkurnikova@dk3n40 ~/work/study/2022-2023/Операционные системы $ git clone --recursive git@github.com:elizavetakurnik
Клонирование в «study_2022-2023.git-os-intro»...
remote: Enumerating objects: 32, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 32 (delta 1), reused 18 (delta 0), pack-reused 0
Получение объектов: 100% (32/32), 18.61 КиБ | 476.00 КиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) записан
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован п
Клонирование в «/afs/.dk.sci.pfu.edu.ru/home/e/d/edkurnikova/work/study/2022-2023/Операционные системы/study_2022-202
remote: Enumerating objects: 95, done.
remote: Counting objects: 100% (95/95), done.
remote: Compressing objects: 100% (67/67), done.
remote: Total 95 (delta 34), reused 87 (delta 26), pack-reused 0
Получение объектов: 100% (95/95), 96.99 КиБ | 1.03 МБ/с, готово.
Определение изменений: 100% (34/34), готово.
Клонирование в «/afs/.dk.sci.pfu.edu.ru/home/e/d/edkurnikova/work/study/2022-2023/Операционные системы/study_2022-202
remote: Enumerating objects: 126, done.
remote: Counting objects: 100% (126/126), done.
remote: Compressing objects: 100% (87/87), done.
remote: Total 126 (delta 52), reused 108 (delta 34), pack-reused 0
Получение объектов: 100% (126/126), 335.80 КиБ | 1.77 МБ/с, готово.
Определение изменений: 100% (52/52), готово.
Submodule path 'template/presentation': checked out '40a1761813e197d00e8443ff1ca72c60a304f24c'
Submodule path 'template/report': checked out '7c31ab8e5dfa8cdb2d67caeb8a19ef8028ced88e'
edkurnikova@dk3n40 ~/work/study/2022-2023/Операционные системы $

```

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.

```

Create mode 100644 project/personal/stage/report/report.md
edkurnikova@dk3n40 ~/work/study/2022-2023/Операционные системы/study_2022-2023.git-os-intro $ git push
Перечисление объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 342.12 КиБ | 2.87 МБ/с, готово.
Всего 38 (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:elizavetakurnikovbibd-01-23/study_2022-2023.git-os-intro.git
 28acf10..d52e112 master -> master
edkurnikova@dk3n40 ~/work/study/2022-2023/Операционные системы/study_2022-2023.git-os-intro $

```

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

```
edkurnikova@dk3n40 ~/work/study/2022-2023/Операционные системы/study_2022-2023.git-os-intro $ make prepare
edkurnikova@dk3n40 ~/work/study/2022-2023/Операционные системы/study_2022-2023.git-os-intro $ ls
CHANGELOG.md  config  COURSE  labs  LICENSE  Makefile  prepare  presentation  project-personal  README.en.md  README.git-flow.md  README.md  template
edkurnikova@dk3n40 ~/work/study/2022-2023/Операционные системы/study_2022-2023.git-os-intro $ git add .
edkurnikova@dk3n40 ~/work/study/2022-2023/Операционные системы/study_2022-2023.git-os-intro $ git commit -am 'feat(main): make course structure'
[master d52e112] feat(main): make course structure
351 files changed, 98413 insertions(+), 14 deletions(-)
create mode 100644 labs/README.md
create mode 100644 labs/README.ru.md
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/placement_600_600_tech.jpg
create mode 100644 labs/lab01/report/pandoc/csl/gost-r-7-8-5-2008-numeric.csl
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_fignos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_scnos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/_init_.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/core.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/main.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/pandocattributes.py
```

Рис. 3.9: 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.