

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

дисциплина: Операционные системы

Студент: Воронцов Павел Васильевич

Группа: НПМбв-02-19

МОСКВА

2023 г.

Цель работы

Целью данной работы является изучение идеологии и методов применения средств контроля версий и освоения умения по работе с git.

Задание

- Создать базовую конфигурацию для работы с git.
- Создать ключ SSH.
- Создать ключ PGP.
- Настроить подписи git.
- Зарегистрироваться на Github.
- Создать локальный каталог для выполнения заданий по предмету.

Лабораторная работа выполняется в Ubuntu.

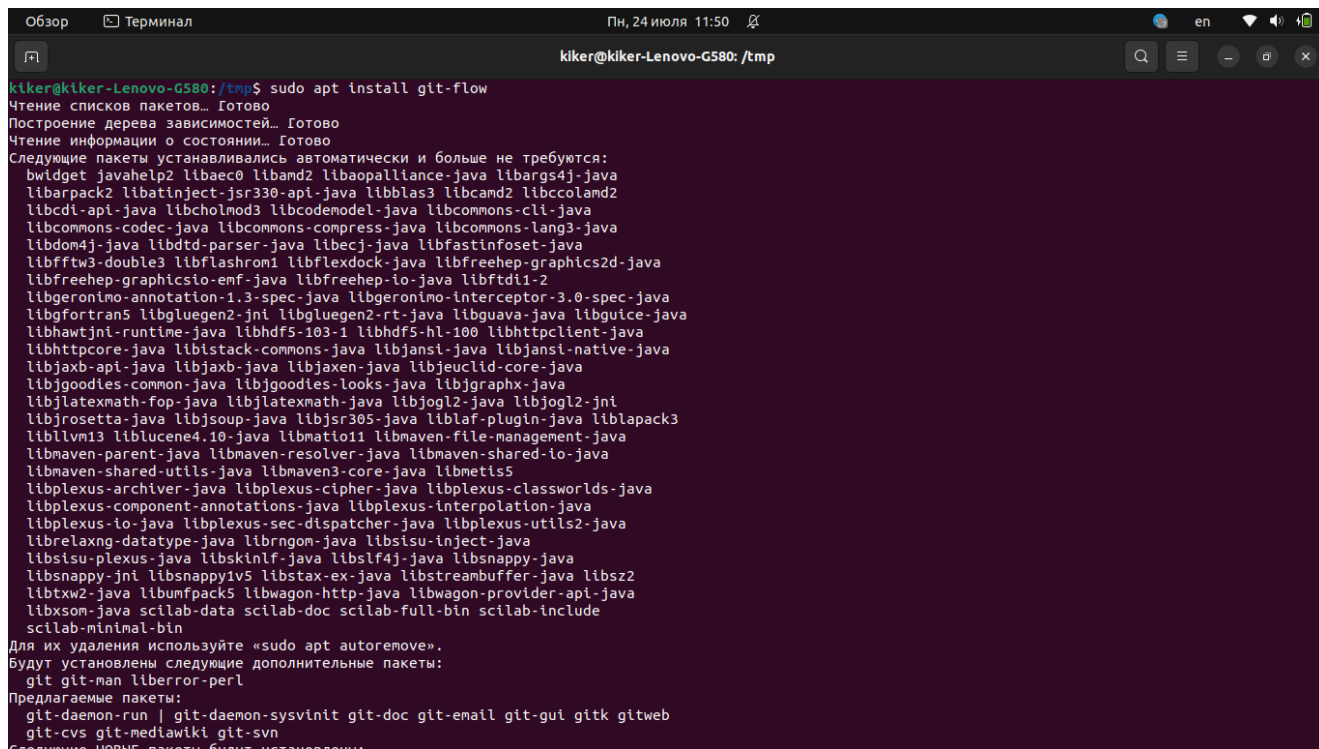
Выполнение лабораторной работы

Настройка github

1. Создайте учётную запись на <https://github.com>.
2. Заполните основные данные на <https://github.com>.

1. Установка программного обеспечения

Установка git-flow в Ubuntu (рис. 1)



```
Обзор Терминал Пн, 24 июля 11:50 kiker@kiker-Lenovo-G580: /tmp$ sudo apt install git-flow
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Следующие пакеты устанавливались автоматически и больше не требуются:
bwidget javahelp2 libaec0 libamd2 libaopalliance-java libargs4j-java
libarpack2 libatinject-jsr330-api-java libblas3 libcamd2 libccolamd2
libcdi-api-java libcholmod3 libcodemodel-java libcommons-cli-java
libcommons-codec-java libcommons-compress-java libcommons-lang3-java
libdom4j-java libdttd-parser-java libecj-java libfastinfoset-java
libfft3-double3 libflashrom1 libflexdock-java libfreehep-graphics2d-java
libfreehep-graphicsio-emf-java libfreehep-io-java libftdi1-2
libgeronimo-annotation-1.3-spec-java libgeronimo-interceptor-3.0-spec-java
libgfortran5 libgluegen2-jni libgluegen2-rt-java libguava-java libguice-java
libhawtjni-runtime-java libhdf5-103-1 libhdf5-hl-100 libhttpclient-java
libhttpcore-java libstack-commons-java libjansi-java libjansi-native-java
libjaxb-api-java libjaxb-java libjaxen-java libjeuclid-core-java
libjgoodies-common-java libjgoodies-looks-java libjgraphx-java
libjlatexmath-fop-java libjlatexmath-java libjogl2-java libjogl2-jni
libjrosetta-java libjsoup-java libjsr305-java liblaf-plugin-java liblapack3
liblvm13 liblucene4.10-java libmatio11 libmaven-file-management-java
libmaven-parent-java libmaven-resolver-java libmaven-shared-io-java
libmaven-shared-utils-java libmaven3-core-java libmetis5
libplexus-archiver-java libplexus-cipher-java libplexus-classworlds-java
libplexus-component-annotations-java libplexus-interpolation-java
libplexus-io-java libplexus-sec-dispatcher-java libplexus-utils2-java
librelaxng-datatype-java librngom-java libsisu-inject-java
libsisu-plexus-java libskinlf-java libslf4j-java libsnappy-java
libsnappy-jni libsnappy1v5 libstax-ex-java libstreambuffer-java libsz2
libtxw2-java libumfpack5 libwagon-http-java libwagon-provider-api-java
libxson-java scilab-data scilab-doc scilab-full-bin scilab-include
scilab-minimal-bin
Для их удаления используйте «sudo apt autoremove».
Будут установлены следующие дополнительные пакеты:
git git-man liberror-perl
Предлагаемые пакеты:
git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb
git-cvs git-mediawiki git-svn
Следующие ЧВРНЕ пакеты будут установлены:
```

Установка gh в Ubuntu (рис.2, рис.3, рис.4)

```
kiker@kiker-Lenovo-G580:/tmp$ curl -fsSL http://cli.github.com/packages/githubcli-archive-keyring.gpg | sudo dd of=/usr/share/keyrings/githubcli-archive-keyring.gpg && sudo chmod go+r /usr/share/keyrings/githubcli-archive-keyring.gpg && echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/githubcli-archive-keyring.gpg] http://cli.github.com/packages stable main" | sudo tee /etc/apt/sources.list.d/github-cli.list > /dev/null
&& sudo apt update && sudo apt install gh -y
4+1 записей получено
4+1 записей отправлено
2270 байт (2,3 kB, 2,2 KiB) скопирован, 0,479785 s, 4,7 kB/s
Сущ:1 http://ru.archive.ubuntu.com/ubuntu jammy InRelease
Пол:2 http://ru.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Пол:3 http://ru.archive.ubuntu.com/ubuntu jammy-backports InRelease [108 kB]
Пол:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Пол:6 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metadata [99,6 kB]
Пол:5 https://cli.github.com/packages stable InRelease [3 917 B]
Пол:7 http://ru.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 DEP-11 Metadata [274 kB]
Пол:8 http://ru.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 DEP-11 Metadata [940 B]
Пол:9 http://ru.archive.ubuntu.com/ubuntu jammy-backports/main amd64 DEP-11 Metadata [7 972 B]
Пол:10 http://ru.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 DEP-11 Metadata [15,4 kB]
Пол:11 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [41,5 kB]
Пол:12 https://cli.github.com/packages stable/main amd64 Packages [344 B]
Пол:13 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Metadata [21,9 kB]
Получено 802 kB за 2с (522 kB/s)
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Может быть обновлено 3 пакета. Запустите «apt list --upgradable» для их показа.
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
```

```
kiker@kiker-Lenovo-G580:/tmp$ type -p curl >/dev/null || sudo apt install curl -y
[sudo] пароль для kiker:
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Следующие пакеты устанавливались автоматически и больше не требуются:
  bwidget javaahelp2 libaec0 libamd2 libaoalliance-java libargs4j-java libarpack2 libatinject-jsr330-api-java libblas3 libcamd2 libccolamd2
  libcdt-api-java libcholmod3 libcodemodel-java libcommons-cli-java libcommons-codec-java libcommons-compress-java libcommons-lang3-java
  libdom4j-java libdtd-parser-java libecj-java libfastinfoset-java libfftw3-double3 libflashrom1 libflexdock-java libfreehep-graphics2d-java
  libfreehep-graphicsio-enf-java libfreehep-io-java libftdi1-2 libgeronimo-annotation-1.3-spec-java libgeronimo-interceptor-3.0-spec-java
  libgfortran5 libgluegen2-jni libgluegen2-rt-java libguava-java libguice-java libhawtjni-runtime-java libhdfs-103-1 libhdfs-hl-100
  libhttpclient-java libhttpcore-java libistack-commons-java libjansi-java libjansi-native-java libjaxb-api-java libjaxb-java libjaxen-java
  libjeuclid-core-java libjgoodies-common-java libjgoodies-looks-java libjgraphx-java libjlatexmath-fop-java libjlatexmath-java libjogl2-java
  libjogl2-jni libjrosetta-java libjsoup-java libjsr305-java liblaf-plugin-java liblapack3 liblvm13 liblucene4.10-java libmatio11
  libmaven-file-management-java libmaven-parent-java libmaven-resolver-java libmaven-shared-io-java libmaven-shared-utils-java libmaven3-core-java
  libmetis5 libplexus-archiver-java libplexus-cipher-java libplexus-classworlds-java libplexus-component-annotations-java
  libplexus-interpolation-java libplexus-io-java libplexus-sec-dispatcher-java libplexus-utils2-java librelaxng-datatype-java librngom-java
  libsisu-inject-java libsisu-plexus-java libskinlf-java libslf4j-java libsnappy-java libsnappy-jni libsnappy1v5 libstax-ex-java
  libstreambuffer-java libs22 libtxw2-java libumfpack5 libwagon-http-java libwagon-provider-api-java libxson-java scilab-data scilab-doc
  scilab-full-bin scilab-include scilab-minimal-bin
Для их удаления используйте «sudo apt autoremove».
Следующие НОВЫЕ пакеты будут установлены:
  curl
Обновлено 0 пакетов, установлено 1 новых пакетов, для удаления отмечено 0 пакетов, и 3 пакетов не обновлено.
Необходимо скачать 194 kB архивов.
После данной операции объем занятого дискового пространства возрастет на 454 kB.
Пол:1 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 curl amd64 7.81.0-1ubuntu1.13 [194 kB]
Получено 194 kB за 0с (777 kB/s)
Выбор ранее не выбранного пакета curl.
(Чтение базы данных ... на данный момент установлено 252504 файла и каталога.)
Подготовка к распаковке ./curl_7.81.0-1ubuntu1.13_amd64.deb ...
Распаковывается curl (7.81.0-1ubuntu1.13) ...
Настраивается пакет curl (7.81.0-1ubuntu1.13) ...
Обрабатываются триггеры для man-db (2.10.2-1) ...
```

```
Следующие НОВЫЕ пакеты будут установлены:
  gh
Обновлено 0 пакетов, установлено 1 новых пакетов, для удаления отмечено 0 пакетов, и 3 пакетов не обновлено.
Необходимо скачать 10,9 MB архивов.
После данной операции объем занятого дискового пространства возрастет на 41,5 MB.
Пол:1 https://cli.github.com/packages stable/main amd64 gh amd64 2.32.0 [10,9 MB]
Получено 10,9 MB за 5с (2 128 kB/s)
Выбор ранее не выбранного пакета gh.
(Чтение базы данных ... на данный момент установлено 252511 файлов и каталогов.)
Подготовка к распаковке ./archives/gh_2.32.0_amd64.deb ...
Распаковывается gh (2.32.0) ...
Настраивается пакет gh (2.32.0) ...
Обрабатываются триггеры для man-db (2.10.2-1) ...
kiker@kiker-Lenovo-G580:/tmp$ sudo apt update
Сущ:1 http://ru.archive.ubuntu.com/ubuntu jammy InRelease
Сущ:2 http://ru.archive.ubuntu.com/ubuntu jammy-updates InRelease
Сущ:3 http://ru.archive.ubuntu.com/ubuntu jammy-backports InRelease
Сущ:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Сущ:5 https://cli.github.com/packages stable InRelease
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Может быть обновлено 3 пакета. Запустите «apt list --upgradable» для их показа.
kiker@kiker-Lenovo-G580:/tmp$ sudo apt install gh
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Уже установлен пакет gh самой новой версии (2.32.0).
```

Базовая настройка git

Рисунок 5.

– Зададим имя и email владельца репозитория:

```
git config --global user.name "Name Surname"
```

```
git config --global user.email "work@mail"
```

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

– Настройте верификацию и подписание коммитов git.

– Зададим имя начальной ветки (будем называть её master): 30 Лабораторная работа No 2.
Управление версиями

```
git config --global init.defaultBranch master
```

 – Параметр `autocrlf`:

```
git config --global core.autocrlf input
```

 – Параметр `safecrlf`:

```
git config --global core.safecrlf warn
```

```
kiker@kiker-Lenovo-G580:/tmp$ git config --global user.name "PoWeeDlo"
kiker@kiker-Lenovo-G580:/tmp$ git config --global user.email "woodpeker2015@yandex.ru"
kiker@kiker-Lenovo-G580:/tmp$ git config --global core.quotepath false
kiker@kiker-Lenovo-G580:/tmp$ git config --global init.defaultBranch master
kiker@kiker-Lenovo-G580:/tmp$ git config --global core.autocrlf input
kiker@kiker-Lenovo-G580:/tmp$ git config --global core.safecrlf warn
kiker@kiker-Lenovo-G580:/tmp$
```

Создайте ключи ssh

Рисунок 6, рисунок 7.

– по алгоритму rsa с ключём размером 4096 бит:

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

– по алгоритму ed25519:

```
ssh-keygen -t ed25519
```

```
kiker@kiker-Lenovo-G580:/tmp$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kiker/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kiker/.ssh/id_rsa
Your public key has been saved in /home/kiker/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Zgdn5DsndR8xma+zTgz5qgR1g0kh3Z2QPEzBcz6V1Tw kiker@kiker-Lenovo-G580
The key's randomart image is:
+---[RSA 4096]-----+
|  . 0=+...=.o      |
| . = 0o+B=   E      |
| o ..=0.*.   .      |
| .. oo0 ..         |
| .  o.S..          |
| .  *o.            |
| .  +o             |
| .  o.             |
| .....            |
+---[SHA256]-----+
```

```
kiker@kiker-Lenovo-G580:/tmp$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/kiker/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kiker/.ssh/id_ed25519
Your public key has been saved in /home/kiker/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:Gg6V8kFkCcV9GjMA2mrdq49HVzWAMW7GTuP83ffwA5U kiker@kiker-Lenovo-G580
The key's randomart image is:
+---[ED25519 256]---+
| o+= oo..          |
| ..B *.   o        |
| oo B X   . .      |
| . o X o.   E       |
| o..o S.   .        |
| o..oo=.. o        |
| . .+o . . + .     |
| . o           =.    |
| o+.           +     |
+---[SHA256]-----+
```

Создайте ключи pgp

Рисунки 8 и 9. – Генерируем ключ `gpg --full-generate-key`

Из предложенных опций выбираем:

- тип RSA and RSA;
- размер 4096;
- выберите срок действия; значение по умолчанию — 0 (срок действия не истекает никогда).

GPG запросит личную информацию, которая сохранится в ключе:

- Имя (не менее 5 символов).
- Адрес электронной почты.
- При вводе email убедитесь, что он соответствует адресу, используемому на GitHub.
- Комментарий. Можно ввести что угодно или нажать клавишу ввода, чтобы оставить это поле пустым.

```
kiker@kiker-Lenovo-G580:/tmp$ gpg --full-generate-key
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
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: Pavel Voroncov
Адрес электронной почты: woodpeker2015@yandex.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Pavel Voroncov <woodpeker2015@yandex.ru>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? 0
Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? 0
Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? 0
Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? 0
```

```
pub  rsa4096 2023-07-24 [SC]
     C2F397C30383ABE91AF73E84E2D384207B80FFF8
uid          Pavel Voroncov <woodpeker2015@yandex.ru>
sub  rsa4096 2023-07-24 [E]
```

Добавление PGP ключа в GitHub

– Выводим список ключей и копируем отпечаток приватного ключа: (рис.10)

`gpg --list-secret-keys --keyid-format LONG`

```
kiker@kiker-Lenovo-G580:/tmp$ gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: глубина: 0 достоверных: 1 подписанных: 0 доверие: 0-, 0q, 0n, 0m, 0f, 1u
/home/kiker/.gnupg/pubring.kbx
-----
sec  rsa4096/E2D384207B80FFF8 2023-07-24 [SC]
     C2F397C30383ABE91AF73E84E2D384207B80FFF8
uid          [ абсолютно ] Pavel Voroncov <woodpeker2015@yandex.ru>
ssb  rsa4096/2F39D30F0AC4FCCC 2023-07-24 [E]
```

– Формат строки:

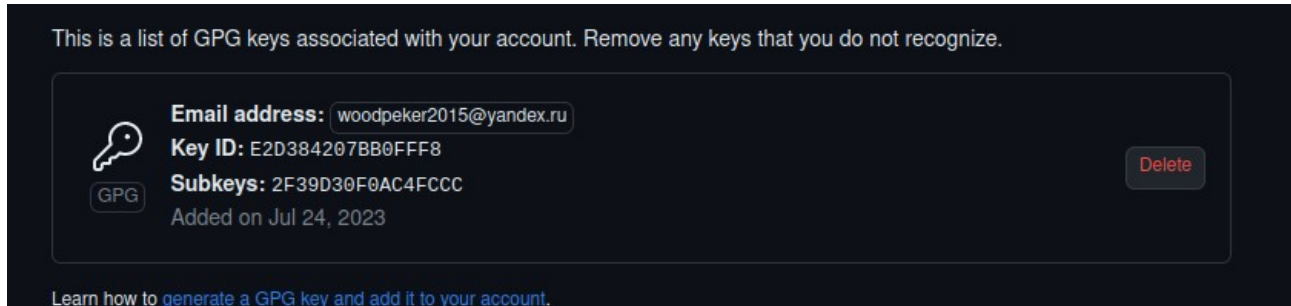
sec Алгоритм/Отпечаток_ключа Дата_создания [Флаги]\ [Годен_до]ID_ключа

– Скопируйте ваш сгенерированный PGP ключ в буфер обмена:

```
gpg --armor --export | xclip -sel clip
```

```
kiker@kiker-Lenovo-G580:/tmp$ gpg --armor --export 2F39D30F0AC4FCCC | xclip -sel clip
kiker@kiker-Lenovo-G580:/tmp$ gpg --armor --export | xclip -sel clip
```

– Перейдите в настройки GitHub (<https://github.com/settings/keys>), нажмите на кнопку New GPG key и вставьте полученный ключ в поле ввода. (рис. 12)



Настройка автоматических подписей коммитов git

– Используя введенный email, укажите Git применять его при подписи коммитов (рис. 13):

```
git config --global user.signingkey
git config --global commit.gpgsign true
git config --global gpg.program $(which gpg2)
```

```
kiker@kiker-Lenovo-G580:/tmp$ gpg --armor --export 2F39D30F0AC4FCCC | xclip -sel clip
kiker@kiker-Lenovo-G580:/tmp$ git config --global user.signingkey 2F39D30F0AC4FCCC
kiker@kiker-Lenovo-G580:/tmp$ git config --global commit.gpgsign true
kiker@kiker-Lenovo-G580:/tmp$ git config --global gpg.program $(which gpg2)
```

Создание репозитория курса на основе шаблона

Шаблон для рабочего пространства

– Репозиторий: <https://github.com/yamadharma/course-directory-student-template>.

– Необходимо создать шаблон рабочего пространства.

– Например, для 2022–2023 учебного года и предмета «Операционные системы» (код предмета os-intro) создание репозитория примет следующий вид:

```
mkdir -p ~/work/study/2021-2022/"Операционные системы"
cd ~/work/study/2021-2022/"Операционные системы"
gh repo create study_2021-2022_os-intro--template=yamadharmacourse-directory-student-template
--public
git clone --recursive git@github.com:/study_2021-2022_os-intro.git os-intro
```

При создании репозитория нам напоминают, что надо авторизоваться (рис.14). Для этого добавляем SSH ключ (рис.15) в аккаунте гитхаба, через `gh auth login` авторизовываемся: Выбираем `github.com`, `SSH`, `Skip`, `Paste an authentication token` (формируем в гх), вставляем его в консоли. (рис. 16) Завершаем создание и клонирование репозитория (рис.17)



```
kiker@kiker-Lenovo-G580:~/work/study/2022-2023/операционные системы$ git clone --recursive git@github.com:PoWeeDlo/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 КиБ | 619.00 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/kiker/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), 92.90 КиБ | 951.00 КиБ/с, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в «/home/kiker/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% (70/70), done.
remote: Total 101 (delta 40), reused 88 (delta 27), pack-reused 0
Получение объектов: 100% (101/101), 327.25 КиБ | 962.00 КиБ/с, готово.
Определение изменений: 100% (40/40), готово.
Submodule path 'template/presentation': checked out 'b1be3800ee91f5809264cb755d316174540b753e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a83917b82e3aef11a33b1e3b2'
```

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



kiker@kiker-Lenovo-G580

SHA256: Gg6VBkfkCcv9GjWA2mrdq49HVzWAMW7GtUP83ffwA5U

Added on Jul 24, 2023

Never used — Read/write

Delete

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

```
kiker@kiker-Lenovo-G580:~/work/study/2022-2023/операционные системы$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? SSH
? Upload your SSH public key to your GitHub account? Skip
? How would you like to authenticate GitHub CLI? Paste an authentication token
Tip: you can generate a Personal Access Token here https://github.com/settings/tokens
The minimum required scopes are 'repo', 'read:org'.
? Paste your authentication token: *****
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
✓ Logged in as PoWeeDlo
```

```
kiker@kiker-Lenovo-G580:~/work/study/2022-2023/операционные системы$ git clone --recursive git@github.com:PoWeeDlo/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 КиБ | 619.00 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/kiker/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), 92.90 КиБ | 951.00 КиБ/с, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в «/home/kiker/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% (70/70), done.
remote: Total 101 (delta 40), reused 88 (delta 27), pack-reused 0
Получение объектов: 100% (101/101), 327.25 КиБ | 962.00 КиБ/с, готово.
Определение изменений: 100% (40/40), готово.
Submodule path 'template/presentation': checked out 'b1be3800ee91f5809264cb755d316174540b753e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a83917b82e3aef11a33b1e3b2'
```

Настройка каталога курса

Рисунок 18.

– Перейдите в каталог курса:

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

– Удалите лишние файлы:

```
rm package.json
```

– Создайте необходимые каталоги:

Лабораторная работа No 2. Управление версиями

```
make COURSE=os-intro
```

```
kiker@kiker-Lenovo-G580:~/work/study/2022-2023/операционные системы$ cd os-intro
kiker@kiker-Lenovo-G580:~/work/study/2022-2023/операционные системы/os-intro$ rm package.json
kiker@kiker-Lenovo-G580:~/work/study/2022-2023/операционные системы/os-intro$ make COURSE=os-intro
kiker@kiker-Lenovo-G580:~/work/study/2022-2023/операционные системы/os-intro$ git add .
kiker@kiker-Lenovo-G580:~/work/study/2022-2023/операционные системы/os-intro$ git commit -am 'feat(main): make course structure'
[master ee04e84] feat(main): make course structure
360 files changed, 100326 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_800_600_tech.jpg
create mode 100644 labs/lab01/report/pandoc/csl/gost-r-7-0-5-2008-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_secnos.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
create mode 100644 labs/lab01/report/report.md
create mode 100644 labs/lab02/presentation/Makefile
create mode 100644 labs/lab02/presentation/image/kulyabov.jpg
create mode 100644 labs/lab02/presentation/presentation.md
create mode 100644 labs/lab02/report/Makefile
create mode 100644 labs/lab02/report/bib/cite.bib
create mode 100644 labs/lab02/report/image/placement_800_600_tech.jpg
create mode 100644 labs/lab02/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_fignos.py
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_secnos.py
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/core.py
```

– Отправьте файлы на сервер (рис. 17, рис.18):

```
git add .
```

```
git commit -am 'feat(main): make course structure'
```

```
git push
```

```
kiker@kiker-Lenovo-G580:~/work/study/2022-2023/операционные системы/os-intro$ git push
Перечисление объектов: 38, готово.
Подсчет объектов: 100% (38/38), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (37/37), 342.36 КиБ | 5.90 МиБ/с, готово.
Всего 37 (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:PoWeeDlo/study_2022-2023_os-intro.git
  5386b5a..ee04e84  master -> master
```

Выводы

По итогу выполнения лабораторной работы удалось познакомиться с идеологией и инструментами системы управления версиями git

Контрольные вопросы

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

VCS — это практика отслеживания изменений программного кода и управления им. Системы контроля версий — это программные инструменты, помогающие командам разработчиков управлять изменениями в исходном коде с течением времени. В свете усложнения сред разработки они помогают командам разработчиков работать быстрее и эффективнее.

2. **Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.**

Хранилище версий – или репозиторий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией. commit делает для проекта снимок текущего состояния изменений, добавленных в раздел проиндексированных файлов. Такие подтвержденные снимки состояния можно рассматривать как «безопасные» версии проекта — VCS не будет их менять, пока вы явным образом не попросите об этом.

log или история перечисляет коммиты, сделанные в репозитории в обратном к хронологическому порядку — последние коммиты находятся вверху. Тут же можно увидеть различие одного коммита от другого

Рабочая копия является снимком одной версии проекта. Эти файлы извлекаются из сжатой базы данных в каталоге Git и помещаются на диск, для того чтобы их можно было использовать или редактировать

Отношения: правки вносятся в рабочую копию, делаете коммит. Коммиты хранятся в репозиториях, log (история) позволяет посмотреть историю коммитов в репо.

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

Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере. Доступ к нему осуществлялся через специальное клиентское приложение. Примеры - CVS, Subversion.

Децентрализованные VCS позволяют хранить репозиторий у каждого разработчика, работающего с данной системой. При этом можно выделить центральный репозиторий (условно), в который будут отправляться изменения из локальных и, с ним же эти локальные репозитории будут синхронизироваться. При работе с такой системой, пользователи периодически синхронизируют свои локальные репозитории с центральным и работают непосредственно со своей локальной копией. Примеры – Git, Mercurial.

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

Установить и настроить VCS клиента. Создать репозиторий. Это можно сделать с помощью команды "git init" (если используется Git)

Добавить файлы в репозиторий. Это можно сделать с помощью команды "git add"

Создать коммит. Коммит можно создать с помощью команды "git commit" (или аналогичной команды в другой VCS).

Просматривать историю коммитов. Это можно сделать с помощью команды "git log"

Восстановить предыдущую версию проекта. Это можно сделать с помощью команды "git checkout".

Создавать и удалять ветки (branch).

5. **Опишите порядок работы с общим хранилищем VCS.**

Получение копии проекта из общего хранилища. Для этого нужно выполнить команду

"git clone" (если используется Git).

Создание новой ветки. Если вы планируете внести изменения в проект, то для этого необходимо создать новую ветку (branch) в вашем локальном репозитории

Внесение изменений. Коммит изменений. После внесения изменений в файлы проекта, необходимо выполнить команду "git commit"

Отправка изменений на сервер. Для этого выполните команду "git push"

Обновление локальной копии проекта. Для этого выполните команду "git pull"

6. Каковы основные задачи, решаемые инструментальным средством git?

- Возврат к любой версии кода из прошлого.
- Просмотр истории изменений.
- Совместная работа без боязни потерять данные или затереть чужую работу.

7. Назовите и дайте краткую характеристику командам git.

Описано в вопросах 4, 5.

8. Приведите примеры использования при работе с локальным и удалённым репозиториями.

Локально:

Создание локального репозитория. "git init" в терминале. Добавление файлов в репозиторий. После создания репозитория вы можете добавить файлы проекта в него, используя команду "git add".

Создание коммита. После добавления файлов вы можете создать коммит, используя команду "git commit -m 'Commit message'". Просмотр истории коммитов. Вы можете просмотреть историю коммитов, используя команду "git log".

Восстановление предыдущей версии. Если в проекте была допущена ошибка или нужно вернуться к предыдущей версии проекта, это можно сделать с помощью команды "git checkout".

Удаленно: Клонирование удаленного репозитория. Чтобы получить локальную копию проекта, вы можете клонировать репозиторий с помощью команды "git clone".

Добавление изменений в локальный репозиторий. После того, как вы получили копию проекта, вы можете вносить изменения и добавлять их в локальный репозиторий с помощью команд "git add" и "git commit -m 'Commit message'".

Отправка изменений в удаленный репозиторий. После добавления изменений в локальный репозиторий вы можете отправить их в удаленный репозиторий, используя команду "git push". Получение изменений из удаленного репозитория. Если в удаленном репозитории были внесены изменения, вы можете получить их и обновить свою локальную копию проекта, используя команду "git pull".

Восстановление предыдущей версии. Если в проекте была допущена ошибка или нужно вернуться к предыдущей версии проекта, это можно сделать с помощью команды "git checkout" в локальном репозитории. Если нужно откатить изменения в удаленном репозитории, можно использовать команду "git revert".

9. Что такое и зачем могут быть нужны ветви (branches)?

Ветви нужны для того, чтобы разделять код. Например одна ветка у нас может быть основная для разработки. Если мы делаем новый функционал, то мы создаем новую ветку под него, а после окончания работы сливаем то, что мы сделали в основную ветку. Это дает нам возможность легко откатывать код, если вдруг мы передумаем его сливать в основную ветку, либо делать несколько различных изменений в разных ветках.

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.

Игнорируемые файлы отслеживаются в специальном файле `.gitignore`, который регистрируется в корневом каталоге репозитория. В Git нет специальной команды для указания игнорируемых файлов: вместо этого необходимо вручную отредактировать файл `.gitignore`, чтобы указать в нем новые файлы, которые должны быть проигнорированы. \ Файлы `.gitignore` содержат шаблоны, которые сопоставляются с именами файлов в репозитории для определения необходимости игнорировать эти файлы.