

[Skip to content](#)



 [AlexandrPriscepov](#) / [Lab-work](#) Public

[Code](#)

[Issues](#)

[Pull requests](#)

[Actions](#)


[Projects](#)

[Wiki](#)

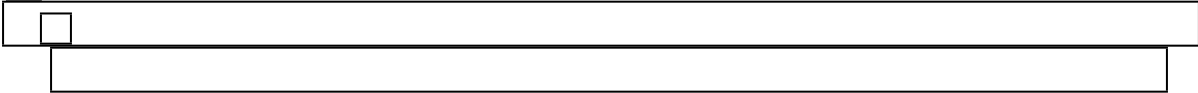
[Security](#)

[Insights](#)

[Settings](#)

 [main](#)

[Lab-work](#) / [Lab03](#) / [Lab02.md](#)



143 lines (91 sloc) 13.8 KB

 [Raw](#) [Blame](#)



2 Лабораторная Работа

Прищепов Александр НПМ-03-21

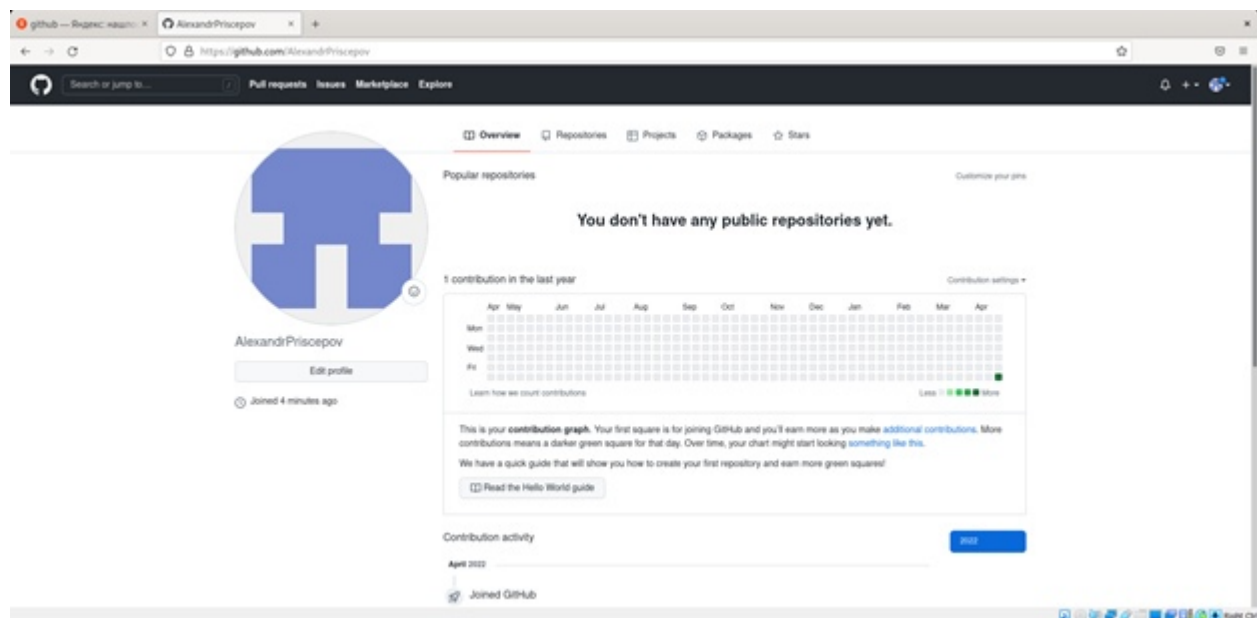
Введение:

- Цель работы:
 - Изучить идеологию и применение средств контроля версий.
 - Освоить умения по работе с git.

Ход Работы:

1. Создаём учётную запись на **Github** и заполняем основные данные на сайте.(рис 1)

рис 1:



2. Устанавливаем программное обеспечение(рис 2,3):

рис 2:

```
aprithepov@10:/tmp — sudo dnf install gh

[aprithepov@10 ~]$ cd /tmp
[aprithepov@10 tmp]$ wget --no-check-certificate -q https://raw.githubusercontent.com/peter
vanderdoes/gitflow/develop/contrib/gitflow-installer.sh
[aprithepov@10 tmp]$ chmod +x gitflow-intaller.sh
chmod: невозможно получить доступ к 'gitflow-intaller.sh': Нет такого файла или
каталога
[aprithepov@10 tmp]$ chmod +x gitflow-installer.sh
[aprithepov@10 tmp]$ sudo ./gitflow-installer.sh install stable

Мы полагаем, что ваш системный администратор изложил вам основы
безопасности. Как правило, всё сводится к трём следующим правилам:

    №1) Уважайте частную жизнь других.
    №2) Думайте, прежде что-то вводить.
    №3) С большой властью приходит большая ответственность.

[sudo] пароль для aprithepov:
### git-flow no-make installer ###
Installing git-flow to /usr/local/bin
Cloning repo from GitHub to gitflow
Клонирование в «gitflow»...
remote: Enumerating objects: 4270, done.
remote: Total 4270 (delta 0), reused 0 (delta 0), pack-reused 4270
Получение объектов: 100% (4270/4270), 1.74 МиБ | 5.62 МиБ/с, готово.
```

рис 3:

```
aprithepov@10:/tmp

'gitflow/hooks/pre-flow-release-start' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-release-start'
'gitflow/hooks/pre-flow-release-track' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-release-track'
[aprithepov@10 tmp]$ sudo dnf install gh
Fedora 35 - x86_64                2.6 MB/s | 79 MB     00:30
Fedora 35 openh264 (From Cisco) - x86_64 4.6 kB/s | 2.5 kB     00:00
Fedora Modular 35 - x86_64        1.7 MB/s | 3.3 MB     00:01
Fedora 35 - x86_64 - Updates      12 MB/s | 29 MB     00:02
Fedora Modular 35 - x86_64 - Updates 1.9 MB/s | 2.9 MB     00:01
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий  Размер
=====
Установка:
gh         x86_64       2.7.0-1.fc35 updates      6.8 М
=====
Результат транзакции
=====
Установка 1 Пакет

Объем загрузки: 6.8 М
Объем изменений: 32 М
Продолжить? [д/н]: д
Загрузка пакетов:
gh-2.7.0-1.fc35.x86_64.rpm          12 MB/s | 6.8 MB     00:00
-----
Общий размер                        5.6 MB/s | 6.8 MB     00:01
Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции проведен успешно.
Выполнение транзакции
Подготовка                          : 1/1
Установка      : gh-2.7.0-1.fc35.x86_64 1/1
Запуск скрипта : gh-2.7.0-1.fc35.x86_64 1/1
Проверка       : gh-2.7.0-1.fc35.x86_64 1/1
Установлен:
gh-2.7.0-1.fc35.x86_64
Выполнено!
[aprithepov@10 tmp]$
```

3. Базовая настройка(рис 4):

рис 4:

```
[aprithepov@10 tmp]$ git config --global user.name "Alexandr Priscepov"
[aprithepov@10 tmp]$ git config --global user.email "priscepov.alexandr@gmail.com"
[aprithepov@10 tmp]$ git config --global core.quotepath false
[aprithepov@10 tmp]$ git config --global init.defaultBranch master
[aprithepov@10 tmp]$ git config --global core.autocrlf input
[aprithepov@10 tmp]$ git config --global core.safecrlf warn
[aprithepov@10 tmp]$
```

4. Создаём ключи ssh(рис 5):

рис 5:

```
aprithepov@10:/tmp
[aprithepov@10 tmp]$ git config --global core.autocrlf input
[aprithepov@10 tmp]$ git config --global core.safecrlf warn
[aprithepov@10 tmp]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/aprithepov/.ssh/id_rsa):
Created directory '/home/aprithepov/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/aprithepov/.ssh/id_rsa
Your public key has been saved in /home/aprithepov/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:GyNztXIhtZ0RBW8W2fkU6swzhzqp0mEkTylXbHW0jzY aprithepov@10.0.2.15
The key's randomart image is:
+---[RSA 4096]-----+
|      o ==+*o|
|      . * ++o+|
|      . = o.+o.|
|      o *  +o..o|
|      o S o  BE..|
|      + @ .o.+ .|
|      = o+      |
|      . o. .    |
|      ..        |
+---[SHA256]-----+
[aprithepov@10 tmp]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/aprithepov/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/aprithepov/.ssh/id_ed25519
Your public key has been saved in /home/aprithepov/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:GU7d0kuchGuwtTbpgi+d7VRZiVYlemvjoTZ2pHWAevA aprithepov@10.0.2.15
The key's randomart image is:
+---[ED25519 256]---+
|      .. o..|
|      ..o= = o|
|      +++X + |
|      o.*B= = .|
|      .S+E.= *|
|      . ...o B o|
|      o +. B +|
|      . +..o +|
|      . ..    |
+---[SHA256]-----+
[aprithepov@10 tmp]$
```

5. Создаём ключи ргр(рис 6,7):

рис 6:

```

[aprithepov@10 tmp]$ gpg --full-generate-key
gpg (GnuPG) 2.3.2; 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.

gpg: создан каталог '/home/aprithepov/.gnupg'
gpg: создан щит с ключами '/home/aprithepov/.gnupg/pubring.kbx'
Выберите тип ключа:
  (1) RSA and RSA
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (9) ECC (sign and encrypt) *default*
 (10) ECC (только для подписи)
 (14) Existing key from card
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

```

рис 7:

```

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: Alexandr Priscepov
Адрес электронной почты: psashok9013@mail.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Alexandr Priscepov <psashok9013@mail.ru>"

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: /home/aprithepov/.gnupg/trustdb.gpg: создана таблица доверия
gpg: ключ 3A0B357C1EADB05C помечен как абсолютно доверенный
gpg: создан каталог '/home/aprithepov/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/aprithepov/.gnupg/openpgp-revocs.d/C682B40278B4A0300F5392293A0B357C1EADB05C.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2022-04-23 [SC]
      C682B40278B4A0300F5392293A0B357C1EADB05C
uid           Alexandr Priscepov <psashok9013@mail.ru>
sub   rsa4096 2022-04-23 [E]

```

6. Добавление PGP ключа в GitHub(рис 8,9,10):

рис 8:

```

[aprithepov@10 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/aprithepov/.gnupg/pubring.kbx
-----
sec   rsa4096/3A0B357C1EADB05C 2022-04-23 [SC]
      C682B40278B4A0300F5392293A0B357C1EADB05C
uid           [ абсолютно ] Alexandr Priscepov <psashok9013@mail.ru>
ssb   rsa4096/AB627268268EF861 2022-04-23 [E]

[aprithepov@10 tmp]$ gpg --armor --export <^C
[aprithepov@10 tmp]$ gpg --armor --export 3A0B357C1EADB05C | xclip -sel clip
bash: xclip: command not found...
Install package 'xclip' to provide command 'xclip'? [N/y] y

* Waiting in queue...
The following packages have to be installed:
  xclip-0.13-15.git11cba61.fc35.x86_64  Command line clipboard grabber
Proceed with changes? [N/y] y

* Waiting in queue...
* Waiting for authentication...
* Waiting in queue...
* Downloading packages...
* Requesting data...
* Testing changes...
* Installing packages...

[aprithepov@10 tmp]$ gpg --armor --export 3A0B357C1EADB05C | xclip -sel clip
[aprithepov@10 tmp]$

```

рис 9:

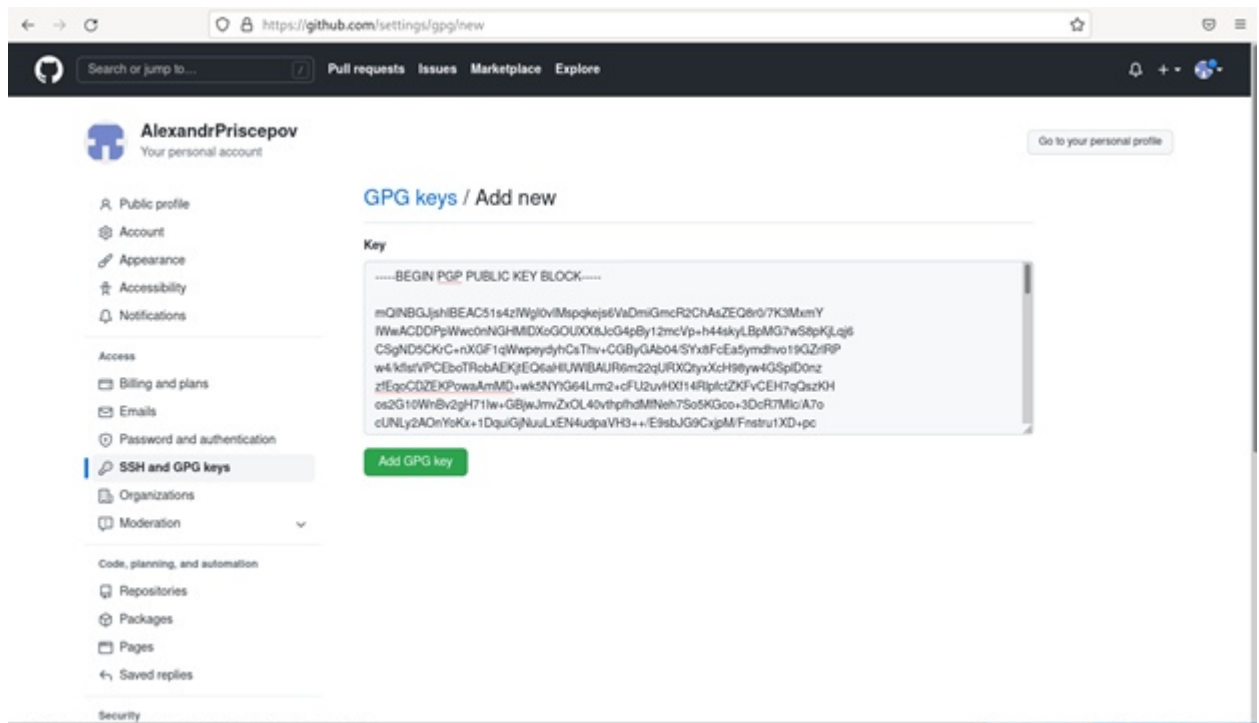
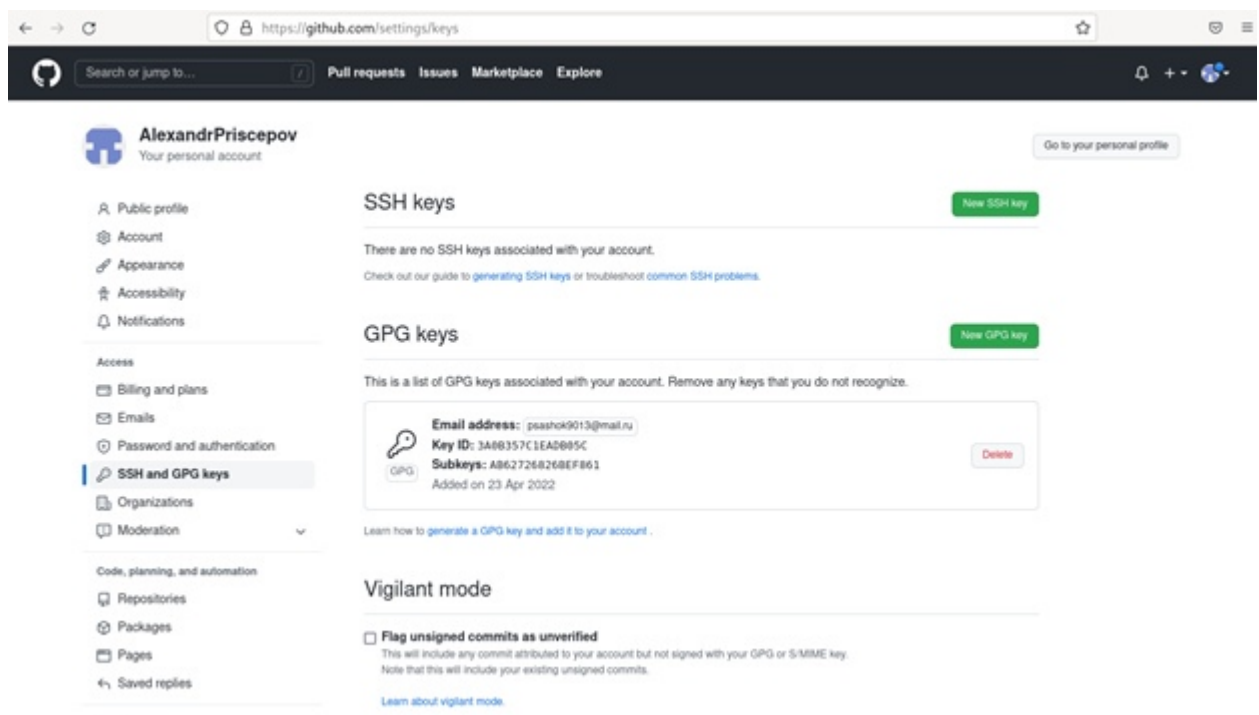


рис 10:



7. Настройка автоматических подписей коммитов git(рис 11):

рис 11:


```
[aprithepov@10 tmp]$ git config --global user.signingkey 3A0B357C1EADB05C
[aprithepov@10 tmp]$ git config --global commit.gpgsign true
[aprithepov@10 tmp]$ git config --global gpg.program $(which gpg2)
```

8. Настройка gh(рис 12):

рис 12:

```
[aprithepov@10 tmp]$ 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? Login with a web browser

! First copy your one-time code: 700E-FB37
Press Enter to open github.com in your browser...
```

9. Создание репозитория курса на основе шаблона(рис 13, 14):

рис 13:

```
[aprithepov@10 tmp]$ mkdir -p ~/work/study/2021-2022/"Операционные системы"
[aprithepov@10 tmp]$ cd ~/work/study/2021-2022/"Операционные системы"
[aprithepov@10 Операционные системы]$ gh repo create study_2021-2022_os-intro --template=yamadharma/course-directory-student-template --public
Created repository AlexandrPriscev/study_2021-2022_os-intro on GitHub
```

рис 14:

```
[aprithepov@10 ~]$ cat ~/.ssh/id_rsa.pub
cat: /home/aprithepov/: too katanor
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAQ0TYlXA714ruU4XSxrgHEXEJ55SVJld2Tn1PXL3lwz2mmHcEUvWI2laWPv7xp34sdmIPHWB/eQmsSnxedjvIk9keq6mjTrMa6bfuGGjm00
k9Hs3FtbrU0vMwBkxtMtJGYDWQah05wjK0zHhM00M0gI/3rDWYwrxxT6YLVGxPL6utoLXCEERgiYwV00s0niWxXWAGve1hdwLdnT8bgQb643I3ywnPa2ycTCQGmVsGtwXXdj6Hq3b3VzKA
/Vs+2PjDxKqWf6xqqAt67GHUeon5BSXGzytWg+pSMwdmv54LMj0AuURmvuGC4tcnjHV6TnJ1D59WCo4UadRyoBVMzbyyMtIZWz6Yz4EZaDoCsb1U9zq28FuohMU2M3asxlh5JT9C+3V7
CULMVz8HbgsWTTg+vWrvYJ28UGMBogurv/C8r0b2HjgYzsPHLmcowm+mkj7ruPJACjkk10xxYRQy0KVHhC15jtztlrzzN7TucxW1wBU7AmVJehgqyTHEvUia5tpptXEKnDAIiv/d7S1FRl
daOmF4Kk4542AaR3cfFBs9T4r+FTvPZrakmDLj3bv8W9wo1pHVPia12sKzIHxgOpTOQs5dC9RoofZ+X1c0hRisAEacScBVl7R/S4jqFj2uRBbvhtKslsTvuwLqetZII57wQKc0ngEqQn4
9sjH/8qlxr0w== aprithepov@10.0.2.15
[aprithepov@10 ~]$
```

10. Настройка каталога курса(рис 15, 16):

рис 15:

```
[aprithepov@10 ~]$ cd ~/work/study/2021-2022/"Операционные системы"/os-intro
[aprithepov@10 os-intro]$ rm package.json
rm: невозможно удалить 'package.json': Нет такого файла или каталога
[aprithepov@10 os-intro]$ ls
[aprithepov@10 os-intro]$ rm package.json
[aprithepov@10 os-intro]$ ls
config LICENSE Makefile README.en.md README.git-flow.md README.md template
[aprithepov@10 os-intro]$ make COURSE=oc-intro
Makefile:8: config/course/oc-intro: Нет такого файла или каталога
make: *** Нет правила для сборки цели «config/course/oc-intro». Останов.
[aprithepov@10 os-intro]$ make COURSE=os-intro
[aprithepov@10 os-intro]$ git add .
[aprithepov@10 os-intro]$ git commit -am 'feat(main): make course structure'
(master b8ec683) feat(main): make course structure
149 files changed, 16590 insertions(+), 14 deletions(-)
create mode 100644 labs/lab01/presentation/Makefile
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
create mode 100644 labs/lab02/presentation/Makefile
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/placeimg_800_600_tech.jpg
create mode 100644 labs/lab02/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab02/report/report.md
create mode 100644 labs/lab03/presentation/Makefile
create mode 100644 labs/lab03/presentation/presentation.md
create mode 100644 labs/lab03/report/Makefile
create mode 100644 labs/lab03/report/bib/cite.bib
create mode 100644 labs/lab03/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab03/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab03/report/report.md
create mode 100644 labs/lab04/presentation/Makefile
create mode 100644 labs/lab04/presentation/presentation.md
create mode 100644 labs/lab04/report/Makefile
create mode 100644 labs/lab04/report/bib/cite.bib
```

рис 16:

```
[aprithepov@10 os-intro]$ git push
Перечисление объектов: 20, готово.
Подсчет объектов: 100% (20/20), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (16/16), готово.
Запись объектов: 100% (19/19), 265.88 КиБ | 2.37 МиБ/с, готово.
Всего 19 (изменений 2), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To github.com:AlexandrPriscepov/study_2021-2022_os-intro.git
cb9c0ea..b8ec683 master -> master
```

Заключение:

Ответы на контрольные вопросы:

1). Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с 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 нескольких 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 quotepath 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 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). 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 к аталог 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) —

отправка изменений конкретной ветки в центральный репозиторий: `git push origin имя_ветки`—слияние ветки с текущим деревом: `git merge --no-ff имя_ветки`—удаление ветки: — удаление локальной уже слитой с основным деревом ветки:`git branch -d имя_ветки`—принудительное удаление локальной ветки:`git branch -D имя_ветки`—удаление ветки с центрального репозитория: `git push origin :имя_ветки`

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 требуется 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 сервисов. 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`

Вывод:

Я изучил идеологию и применение средств контроля версий, а так же освоил умения по работе с `git`.



© 2022 GitHub, Inc.

[Terms](#)

[Privacy](#)

[Security](#)

[Status](#)
[Docs](#)
[Contact GitHub](#)
[Pricing](#)
[API](#)
[Training](#)
[Blog](#)
[About](#)