

# **Отчет по лабораторной работе №2**

**Управление версиями**

Беличева Д.М.

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>5</b>	<b>Выводы</b>	<b>14</b>
<b>6</b>	<b>Ответы на контрольные вопросы</b>	<b>15</b>
	<b>Список литературы</b>	<b>19</b>

## Список иллюстраций

4.1	Создание учетной записи на GitHub . . . . .	7
4.2	Установка git-flow в Fedora Linux . . . . .	8
4.3	Установка gh в Fedora Linux . . . . .	8
4.4	Базовая настройка git . . . . .	8
4.5	Создания ключа ssh по алгоритму rsa с ключем размером 4096 . .	9
4.6	Создания ключа ssh по алгоритму ed25519 . . . . .	9
4.7	Создание ключа gpg . . . . .	10
4.8	Вывод списка ключей . . . . .	10
4.9	Копирование сгенерированного PGP ключа в буфер обмена . . . .	10
4.10	Добавление PGP ключа в GitHub . . . . .	11
4.11	Настройка автоматических подписей коммитов git . . . . .	11
4.12	Настройка gh . . . . .	11
4.13	Подтверждение авторизации . . . . .	11
4.14	Добавление ssh ключа в GitHub . . . . .	12
4.15	Создание репозитория курса на основе шаблона . . . . .	12
4.16	Настройка каталога курса . . . . .	12
4.17	Настройка каталога курса . . . . .	13

# 1 Цель работы

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

## 2 Задание

1. Зарегистрироваться на Github;
2. Создать базовую конфигурацию для работы с git;
3. Создать ключ SSH;
4. Создать ключ PGP;
5. Настроить подписи git;
6. Создать локальный каталог для выполнения заданий по предмету.

### 3 Теоретическое введение

В этой лабораторной работе мы познакомимся с системами контроля версий. 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. Существуют классические и распределённые системы контроля версий (РСКВ). Сегодня мы будем работать с распределённой VSC – Git. В РСКВ (таких как Git, Mercurial, Bazaar или Darcs) клиенты не просто скачивают снимок всех файлов - они полностью копируют репозиторий. В этом случае, если один из серверов, через который разработчики обменивались данными, умрёт, любой клиентский репозиторий может быть скопирован на другой сервер для продолжения работы. Каждая копия репозитория является полным бэкапом всех данных. Более того, многие РСКВ могут одновременно взаимодействовать с несколькими удалёнными репозиториями, благодаря этому вы можете работать с различными группами людей, применяя различные подходы единовременно в рамках одного проекта. Это позволяет применять сразу несколько подходов в разработке, например, иерархические модели, что совершенно невозможно в централизованных системах. [1]

## 4 Выполнение лабораторной работы

№1 Создаем учетную запись на Github и заполняем основные данные. (рис. 4.1)

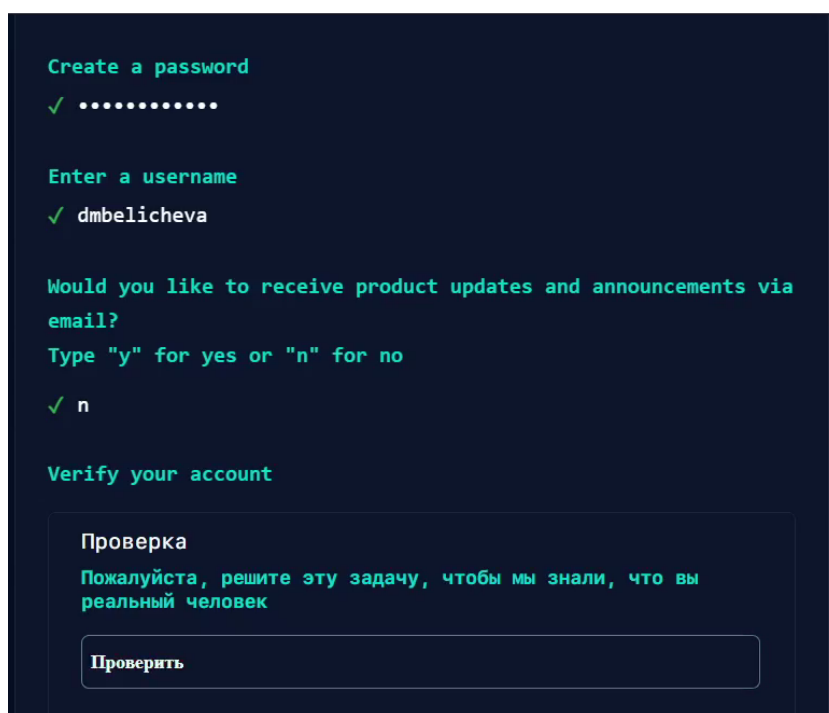
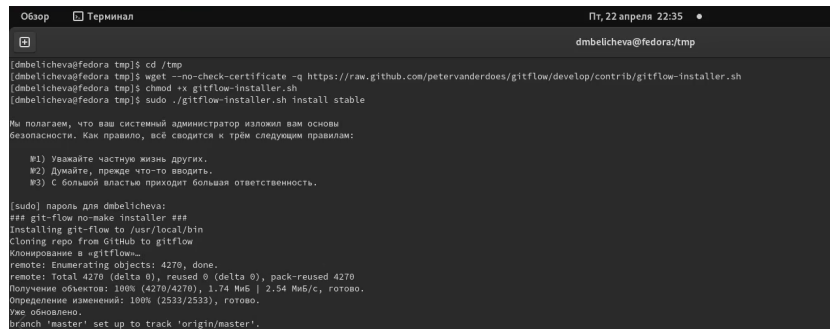
The image shows a terminal window with a dark background and light green text. It displays the steps for creating a GitHub account. The first step is 'Create a password', followed by a green checkmark and a series of dots representing a password. The second step is 'Enter a username', followed by a green checkmark and the username 'dmbelicheva'. The third step is a question: 'Would you like to receive product updates and announcements via email?' with the instruction 'Type "y" for yes or "n" for no'. This is followed by a green checkmark and the answer 'n'. The final step is 'Verify your account', which includes a sub-section titled 'Проверка' (Check) with the text 'Пожалуйста, решите эту задачу, чтобы мы знали, что вы реальный человек' (Please solve this task so we know you are a real person). Below this text is a button labeled 'Проверить' (Check).

Рис. 4.1: Создание учетной записи на GitHub

№2 Далее установим программное обеспечение git-flow в Fedora Linux (сделаем это вручную). (рис. 4.2)



```
Обзор Терминал Пн, 22 апреля 22:35
dmbelicheva@fedora/tmp

[dmbelicheva@fedora tmp]$ cd /tmp
[dmbelicheva@fedora tmp]$ wget --no-check-certificate -q https://raw.githubusercontent.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh
[dmbelicheva@fedora tmp]$ chmod +x gitflow-installer.sh
[dmbelicheva@fedora tmp]$ sudo ./gitflow-installer.sh install stable

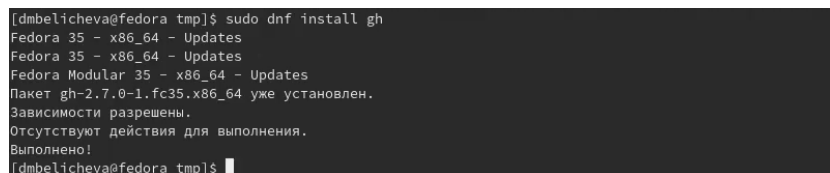
Мы полагаем, что ваш системный администратор изложил вам основы
безопасности. Как правило, всё сводится к трем следующим правилам:

#1 Уважайте частную жизнь других.
#2 Думайте, прежде что-то ввести.
#3 С большой властью приходит большая ответственность.

[sudo] пароль для dmbelicheva:
## 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 Мб | 2.54 Мб/с, готово.
Определение изменений: 100% (2533/2533), готово.
Все обновлено.
branch 'master' set up to track 'origin/master'.
```

Рис. 4.2: Установка git-flow в Fedora Linux

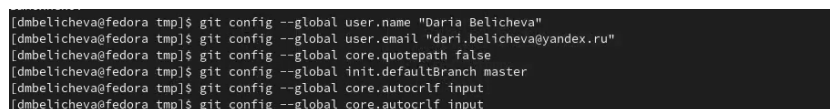
Установим gh в Fedora Linux. (рис. 4.3)



```
[dmbelicheva@fedora tmp]$ sudo dnf install gh
Fedora 35 - x86_64 - Updates
Fedora 35 - x86_64 - Updates
Fedora Modular 35 - x86_64 - Updates
Пакет gh-2.7.0-1.fc35.x86_64 уже установлен.
Зависимости разрешены.
Отсутствуют действия для выполнения.
Выполнено!
[dmbelicheva@fedora tmp]$
```

Рис. 4.3: Установка gh в Fedora Linux

Перейдем к базовой настройке Git: зададим имя и почту владельца репозитория; настроим utf-8 в выводе сообщений git; настроим верификацию и подписание коммитов git (Зададим имя начальной ветки (будем называть её master), параметр autocrlf, параметр safecrlf). (рис. 4.4)



```
[dmbelicheva@fedora tmp]$ git config --global user.name "Daria Belicheva"
[dmbelicheva@fedora tmp]$ git config --global user.email "dari.belicheva@yandex.ru"
[dmbelicheva@fedora tmp]$ git config --global core.quotePath false
[dmbelicheva@fedora tmp]$ git config --global init.defaultBranch master
[dmbelicheva@fedora tmp]$ git config --global core.autocrlf input
[dmbelicheva@fedora tmp]$ git config --global core.safecrlf input
```

Рис. 4.4: Базовая настройка git

№3 Создаем ключ ssh: по алгоритму rsa с ключём размером 4096 бит: (рис. 4.5)



```
[dmbelicheva@fedora tmp]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/dmbelicheva/.ssh/id_rsa):
Created directory '/home/dmbelicheva/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/dmbelicheva/.ssh/id_rsa
Your public key has been saved in /home/dmbelicheva/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:yuqk0S0k8kVW7ySjphDZ+FdI1Wj5KWDZJbSYbXpVpLQ dmbelicheva@fedora
The key's randomart image is:
+---[RSA 4096]-----+
|          ==+         |
|          =+++        |
| + oo=Eo .           |
| + . +O= =           |
| | o o.O+S           |
| + o == ..           |
| ,= *o o             |
| =+..                |
| O+O                 |
+---[SHA256]-----+
```

Рис. 4.5: Создания ключа ssh по алгоритму rsa с ключем размером 4096

по алгоритму ed25519: (рис. 4.6)

```
[dmbelicheva@fedora tmp]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/dmbelicheva/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/dmbelicheva/.ssh/id_ed25519
Your public key has been saved in /home/dmbelicheva/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:MbY2H236SmhPdyWpBzjkw3ulhx3j/3vmT8RZC5aswoE dmbelicheva@fedora
The key's randomart image is:
+---[ED25519 256]---+
|          .           |
|          .. .. .     |
| Eo.o o=. +          |
| .+.o+.+=+          |
| So .o O=+          |
| . .&=              |
| .-o@               |
| .X*                |
| ..=&                |
+---[SHA256]-----+
```

Рис. 4.6: Создания ключа ssh по алгоритму ed25519

№4 Создаем ключ gpg. Генерируем ключ и 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. (рис. 4.7)

```
[dmbelicheva@fedora tmp]$ gpg --full-generate-key
gpg (GnuPG) 2.3.4; 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 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

GnuPG должен составить идентификатор пользователя для идентификации ключа.
Ваше полное имя: Daria
Адрес электронной почты: dari.belicheva@yandex.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
"Daria <dari.belicheva@yandex.ru>"

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

Рис. 4.7: Создание ключа gpg

Добавим pgr ключ в GitHub. Выводим список ключей и копируем отпечаток приватного ключа. (рис. 4.8)

```
[dmbelicheva@fedora 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/dmbelicheva/.gnupg/pubring.kbx
-----
sec rsa4096/A1EA2071843DF9CA 2022-04-22 [SC]
33A19588DDC6F96A8F4D338AA1EA2071843DF9CA
uid [ абсолютно ] Daria <dari.belicheva@yandex.ru>
ssb rsa4096/072FCC407133FF5F 2022-04-22 [E]
```

Рис. 4.8: Вывод списка ключей

Скопируем сгенерированный PGP ключ в буфер обмена. (рис. 4.9)

```
[dmbelicheva@fedora tmp]$ gpg --armor --export <PGP Fingerprint> xclip -sel clip
bash: PGP: Нет такого файла или каталога
[dmbelicheva@fedora tmp]$ gpg --armor --export A1EA2071843DF9CA | xclip -sel clip
```

Рис. 4.9: Копирование сгенерированного PGP ключа в буфер обмена

Перейдем в настройки GitHub (<https://github.com/settings/keys>), нажмем на кнопку New GPG key и вставим полученный ключ в поле ввода. (рис. 4.10)

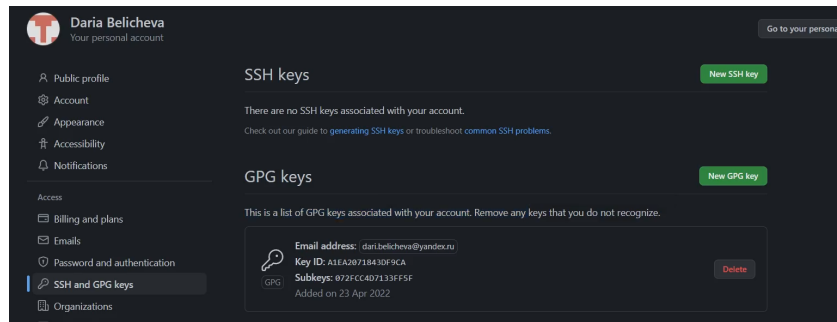


Рис. 4.10: Добавление PGP ключа в GitHub

№5 Настроим автоматические подписи коммитов git. Используя введённый email, укажем Git применять его при подписи коммитов. (рис. 4.11)

```
[dmbelicheva@fedora tmp]$ git config --global user.signingkey A1EA2071843DF9CA
[dmbelicheva@fedora tmp]$ git config --global commit.gpgsign true
[dmbelicheva@fedora tmp]$ git config --global gpg.program $(which gpg2)
```

Рис. 4.11: Настройка автоматических подписей коммитов git

Настроим gh. (рис. 4.12)

```
[dmbelicheva@fedora 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? /home/dmbelicheva/.ssh/id_rsa.pub
? How would you like to authenticate GitHub CLI? Login with a web browser

First copy your one-time code: 6C92-467B
Press Enter to open github.com in your browser...
restorecon: SELinux: Could not get canonical path for /home/dmbelicheva/.mozilla/firefox/* restorecon: No such file or directory.
```

Рис. 4.12: Настройка gh

Для начала необходимо авторизоваться. Ответим на несколько наводящих вопросов, которые задаст утилита. Авторизуемся можно через браузер. (рис. 4.13)

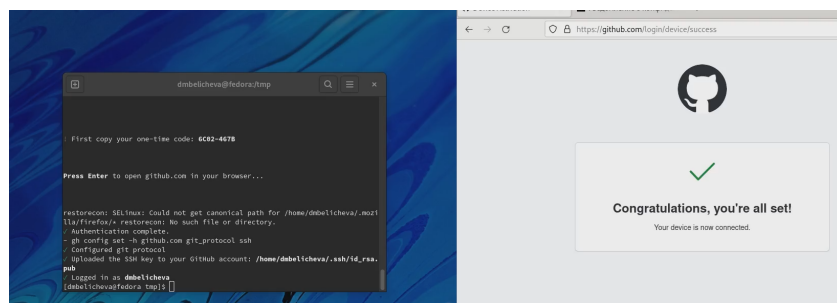


Рис. 4.13: Подтверждение авторизации

Покажем, что ключ SSH добавился. (рис. 4.14)

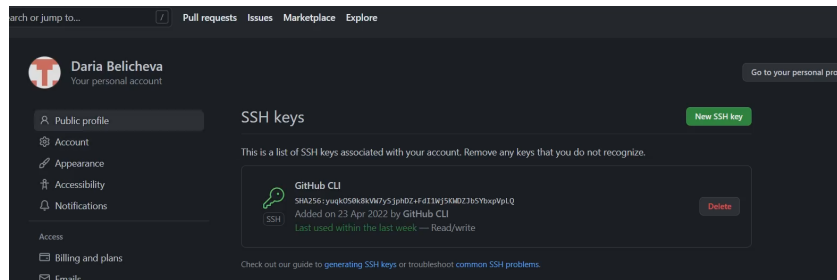


Рис. 4.14: Добавление ssh ключа в GitHub

## №6 Необходимо создать шаблон рабочего пространства. (рис. 4.15)

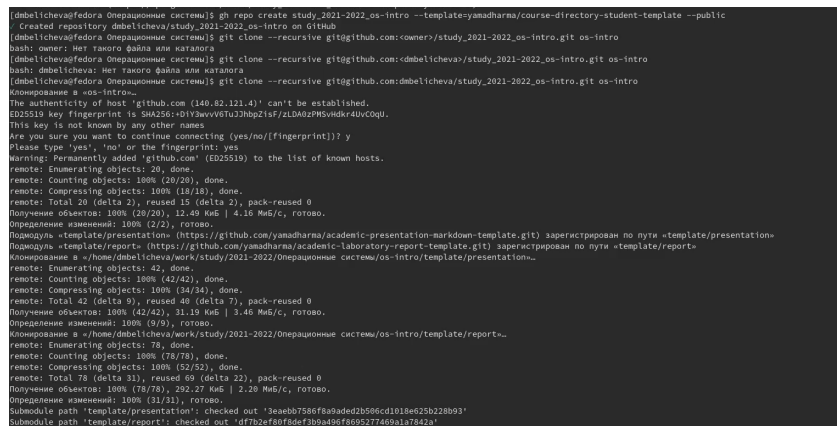


Рис. 4.15: Создание репозитория курса на основе шаблона

Перейдем в каталог курса. Удалим лишние файлы. Создадим необходимые каталоги. Отправим файлы на сервер. (рис. 4.16)

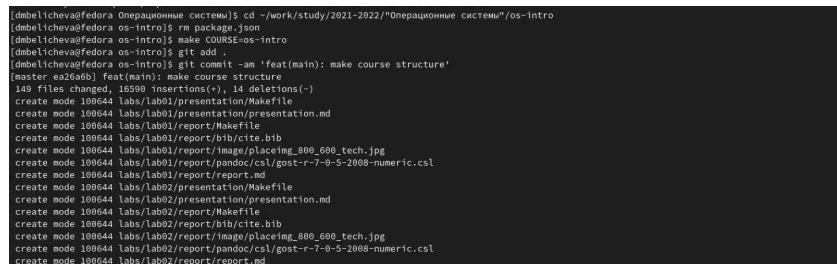


Рис. 4.16: Настройка каталога курса

Продолжаем настройку каталога. (рис. 4.17)

```
[dmbelicheva@fedora 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), 266.53 КиБ | 2.26 МБ/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:dmbelicheva/study_2021-2022_os-intro.git
  7898575..ea26a6b  master -> master
```

Рис. 4.17: Настройка каталога курса

## 5 Выводы

Изучила средства контроля версий и научилась применять их. Освоила работу с git, научилась подключать репозитории, добавлять и удалять необходимые файлы.

## 6 Ответы на контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначены? Система контроля версий — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями 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. Системы контроля версий (Version Control System, VCS) применяются для:
  - Хранение полной истории изменений
  - причин всех производимых изменений
  - Откат изменений, если что-то пошло не так
  - Поиск причины и ответственного за появления ошибок в программе
  - Совместная работа группы над одним проектом
  - Возможность изменять код, не мешая работе других пользователей
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Репозиторий - хранилище версий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией. Commit — отслеживание изменений, сохраняет разницу в изменениях Рабочая копия - копия проекта, связанная с репозиторием (текущее состояние файлов проекта, основанное на версии из хранилища (обычно на последней)) История хранит все изменения в проекте и позволяет при необходимости обратиться к нужным данным.
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные

VCS (Subversion; CVS; TFS; VAULT; AccuRev): • Одно основное хранилище всего проекта • Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно

Децентрализованные VCS (Git; Mercurial; Bazaar): • У каждого пользователя свой вариант (возможно не один) репозитория • Присутствует возможность добавлять и забирать изменения из любого репозитория [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. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

4. Опишите действия с VCS при единоличной работе с хранилищем. Сначала создаем и подключаем удаленный репозиторий. Затем по мере изменения проекта отправлять эти изменения на сервер.
5. Опишите порядок работы с общим хранилищем VCS. 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.
6. Каковы основные задачи, решаемые инструментальным средством git? 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. Наиболее часто используемые команды git: • создание основного дерева репозитория: git init • получение обновлений (изменений) текущего дерева из центрального



репозитория: `git pull` • отправка всех произведённых изменений локального дерева в центральный репозиторий: `git push` • просмотр списка изменённых файлов в текущей директории: `git status` • просмотр текущих изменений: `git diff` • сохранение текущих изменений: – добавить все изменённые и/или созданные файлы и/или каталоги: `git add`. – добавить конкретные изменённые и/или созданные файлы и/или каталоги: `git add имена_файлов` • удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): `git rm имена_файлов` • сохранение добавленных изменений: – сохранить все добавленные изменения и все изменённые файлы: `git commit -am 'Описание коммита'` – сохранить добавленные изменения с внесением комментария через встроенный редактор `git commit` • создание новой ветки, базирующейся на текущей: `git checkout -b имя_ветки` • переключение на некоторую ветку: `git checkout имя_ветки` (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой) • 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. Приведите примеры использования при работе с локальным и удалённым репозиториями. `git push --all` (`push origin master/любой branch`)
9. Что такое и зачем могут быть нужны ветви (branches)? Ветвление («ветка», `branch`) — один из параллельных участков истории в одном хранилище, исходящих из одной версии (точки ветвления). [3] • Обычно есть главная ветка (`master`), или ствол (`trunk`). • Между ветками, то есть их концами, возможно слияние. Используются для разработки новых функций.

10. Как и зачем можно игнорировать некоторые файлы при commit? 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.

## Список литературы

1. О системе контроля версий [Электронный ресурс]. 2016. URL: <https://git-scm.com/book/ru/v2/Введение-О-системе-контроля-версий>.
2. Евгений Г. Системы контроля версий [Электронный ресурс]. 2016. URL: [https://glebradchenko.susu.ru/courses/bachelor/engineering/2016/SUSU\\_SE\\_2016\\_REP\\_3\\_VCS.pdf](https://glebradchenko.susu.ru/courses/bachelor/engineering/2016/SUSU_SE_2016_REP_3_VCS.pdf).
3. Системы контроля версий [Электронный ресурс]. 2016. URL: [http://uii.mpei.ru/study/courses/sdt/16/lecture02.2\\_vcs.slides.pdf](http://uii.mpei.ru/study/courses/sdt/16/lecture02.2_vcs.slides.pdf).