Отчёт по лабораторной работе №2: Работа с Git

Дисциплина: Операционные системы

Дарья Эдуардовна Ибатулина

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Задание

1. Установить Git;
2. Сделать базовую настройку;
3. Сгенерировать ssh и gpg ключи;
4. Авторизоваться на Github;
5. Настроить каталог курса;
6. Удалить лишние файлы и отправить файлы на сервер.

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.

4 Выполнение лабораторной работы

- **Установка Git и gh**

  При помощи команды dnf install git устанавливаем систему контроля версий - Гитхаб и gh, необходимый для облегчения дальнейшей работы с гитом. (рис. 4.1).

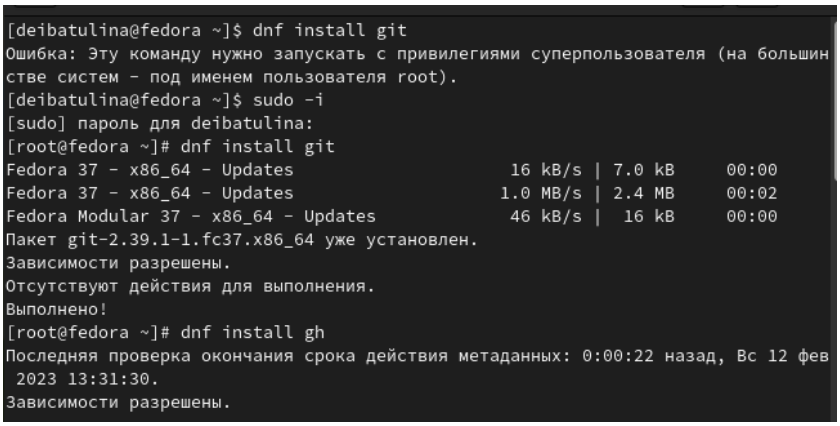

Рис. 4.1: Установка гита

- **Базовая настройка Git**

  Зададим имя и email владельца репозитория, а также кодировку utf-8 в выводе сообщений git, имя начальной ветки и параметры, создадим ключ ssh размером 4096 бит (рис. 4.2).:

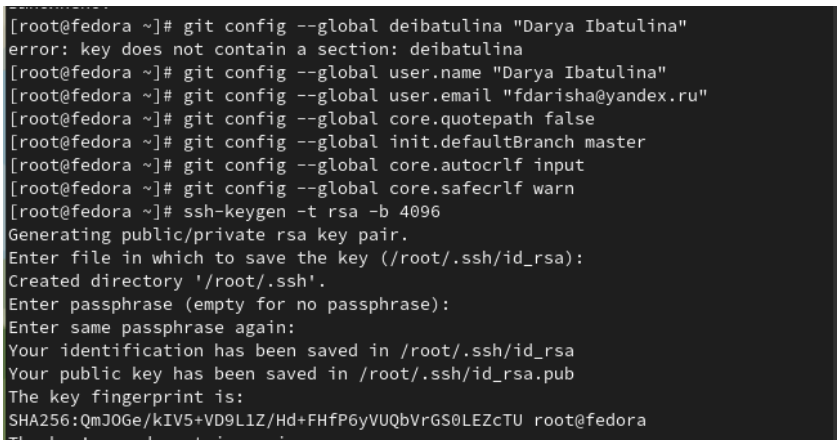

Рис. 4.2: Базовая настройка Git

Создадим ключ ssh по алгоритму ed-25519 (рис. 4.3):

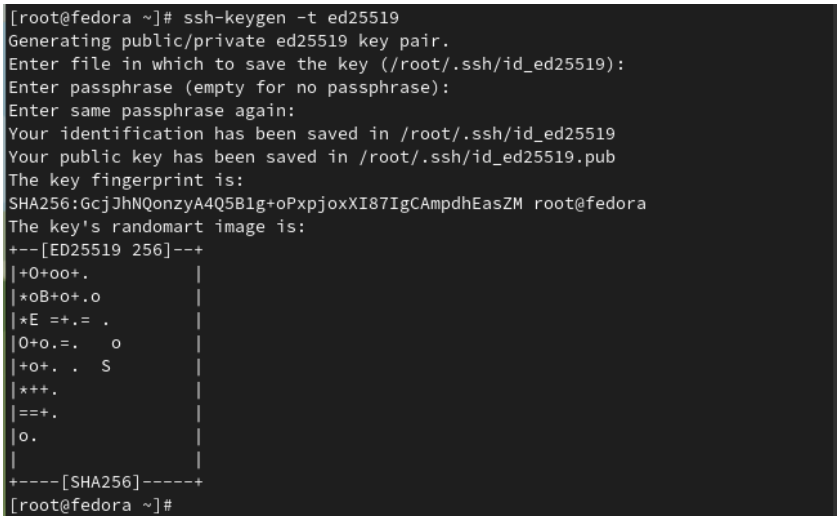

Рис. 4.3: Создание ssh ключа по алгоритму ed-25519

При помощи команды gpg –full-generate-key генерируем gpg ключ, отвечаем на вопросы, которые задаёт нам система относительно ключа (рис. 4.4):

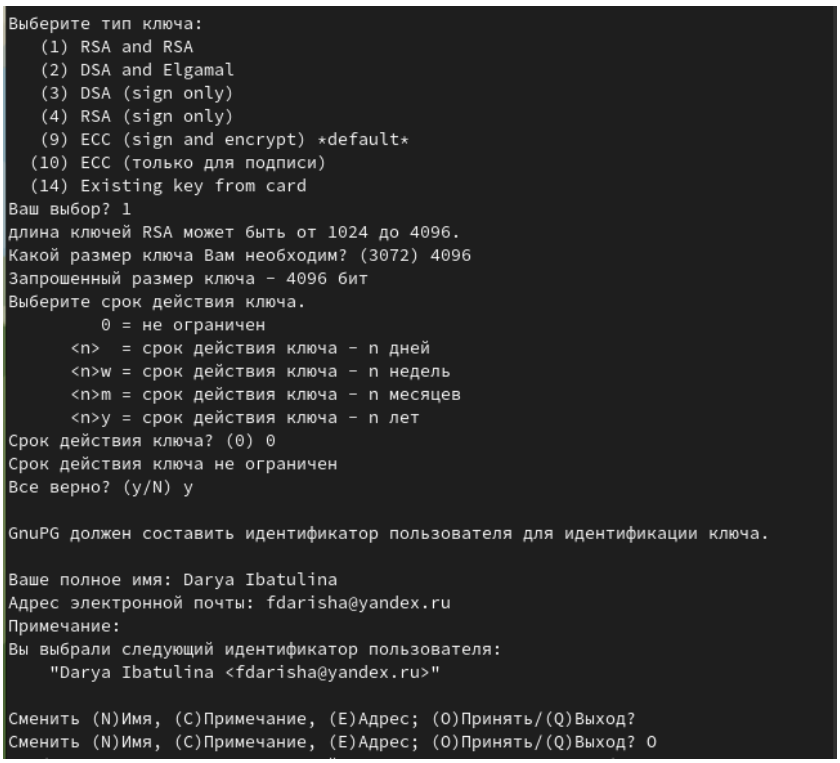

Рис. 4.4: Создание gpg ключа

Далее необходимо настроить учётную запись на сайте github. Я это уже проделала в прошлом семестре.

- **Добавление PGP ключа в GitHub**

  Выводим список ключей и копируем отпечаток приватного ключа (рис. 4.5):

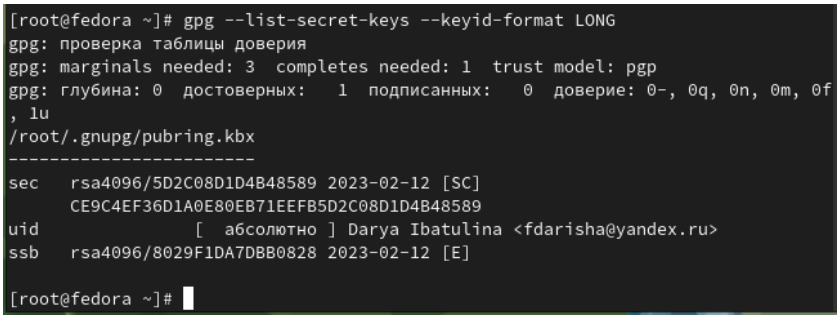

Рис. 4.5: Вывод списка всех ключей, генерация отпечатка приватного ключа

Отпечаток ключа — это последовательность байтов, используемая для идентификации более длинного, по сравнению с самим отпечатком ключа. Он идё после rsa/... . Копируем отпечаток в буфер обмена, настраиваем автоматические подписи коммитов (рис. 4.6):

Рис. 4.6: Копирование отпечатка в буфер обмена, настройка авто-подписей коммитов

Вставляем скопированный ключ в поле ввода для gpg ключа на сайте github. Авторизуемся на Github через браузер (рис. 4.7):

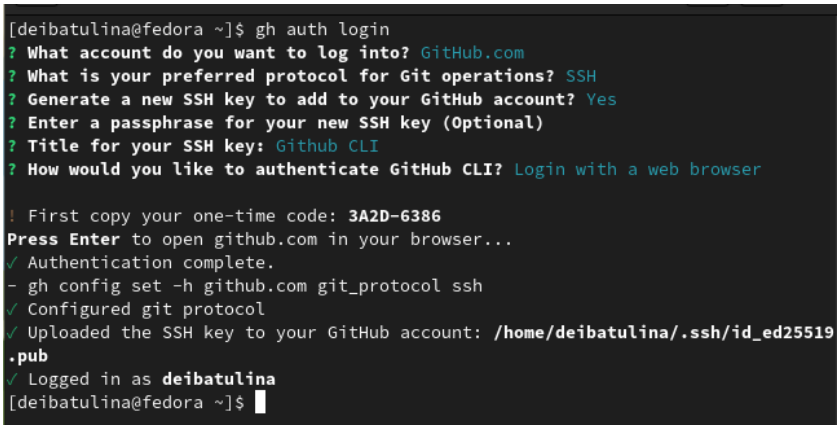

Рис. 4.7: Авторизация через браузер

Авторизация прошла успешно (рис. 4.8), (рис. 4.9):

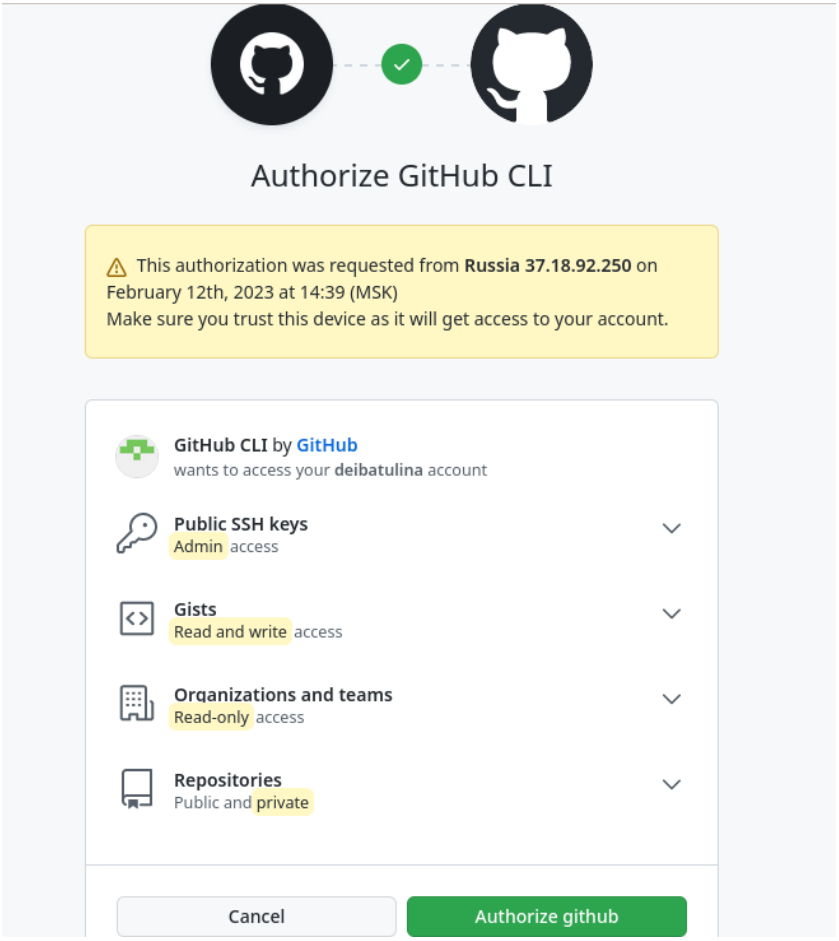

Рис. 4.8: Авторизация прошла успешно

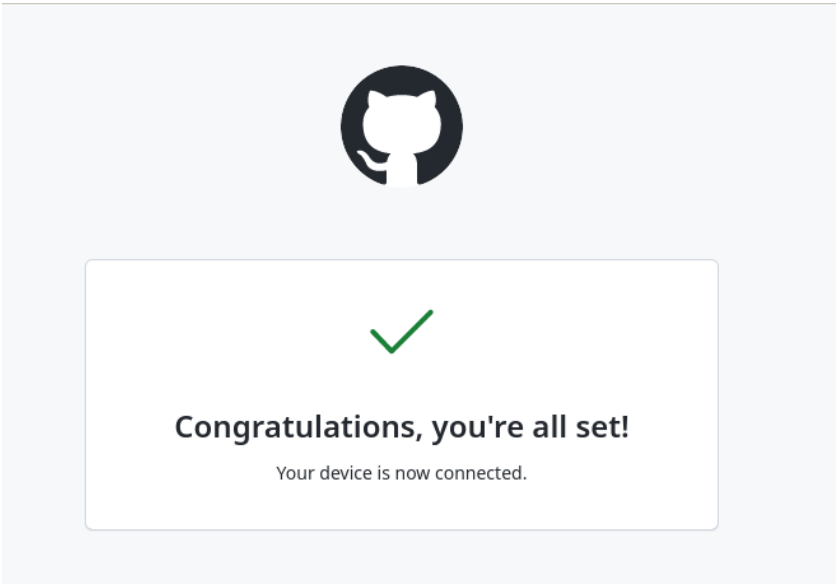

Рис. 4.9: Авторизация прошла успешно

- **Настройка каталога курса**

  Необходимо настроит каталог курса, как указано в соглашении об именовании (рис. 4.10):

```
[deibatulina@fedora ~]$ cd ~/work/study/2022-2023/"Операционные системы"
[deibatulina@fedora Операционные системы]$ gh repo create study_2022-2023_os-int
ro --template=yamadharma/course-directory-student-template --public
✓ Created repository deibatulina/study_2022-2023_os-intro on GitHub
[deibatulina@fedora Операционные системы]$ git clone --recursive git@github.com:
deibatulina/study_2022-2023_os-intro.git os-intro
Клонирование в «os-intro»...
The authenticity of host 'github.com (140.82.121.4)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
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 | 16.93 МиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presen
tation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-r
eport-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/deibatulina/work/study/2022-2023/Операционные системы/os-i
ntro/template/presentation»...
```

Рис. 4.10: Настройка каталог курса, переход в него

Удаляем лишние файлы, создаём необходимые каталоги (рис. 4.11):

```
[deibatulina@fedora Операционные системы]$ cd ~/work/study/2022-2023/"Операционн
ые системы"/os-intro
[deibatulina@fedora os-intro]$ rm package.json
[deibatulina@fedora os-intro]$ echo os-intro > COURSE
[deibatulina@fedora os-intro]$ make
[deibatulina@fedora os-intro]$ git add .
[deibatulina@fedora os-intro]$ git commit -am 'feat(main): make course structure
'
Author identity unknown

*** Пожалуйста, скажите мне кто вы есть.

Запустите

  git config --global user.email "you@example.com"
  git config --global user.name "Ваше Имя"

для указания идентификационных данных аккаунта по умолчанию.
Пропустите параметр --global для указания данных только для этого репозитория.

fatal: не удалось выполнить автоопределение адреса электронной почты (получено «
deibatulina@fedora.(none)»)
[deibatulina@fedora os-intro]$ git config --global user.email "fdarisha@yandex.r
u"
```

Рис. 4.11: Удление лишних файлов, создание необходимых каталогов

Как мы видим, система просит авторизоваться заново, поскольку я закрыла вкладку браузера с авторизацией. Делаем это ещё раз и переходим к следующему шагу (рис. 4.12):

```
[deibatulina@fedora os-intro]$ git config --global user.email "fdarisha@yandex.r
u"
[deibatulina@fedora os-intro]$ git config --global user.name "Darya Ibatulina"
[deibatulina@fedora os-intro]$ git commit -am 'feat(main): make course structure
'
[master 55bac58] feat(main): make course structure
 361 files changed, 100327 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/placeimg_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
```

Рис. 4.12: Повторная авторизация

Отправляем файлы на сервер (рис. 4.13):

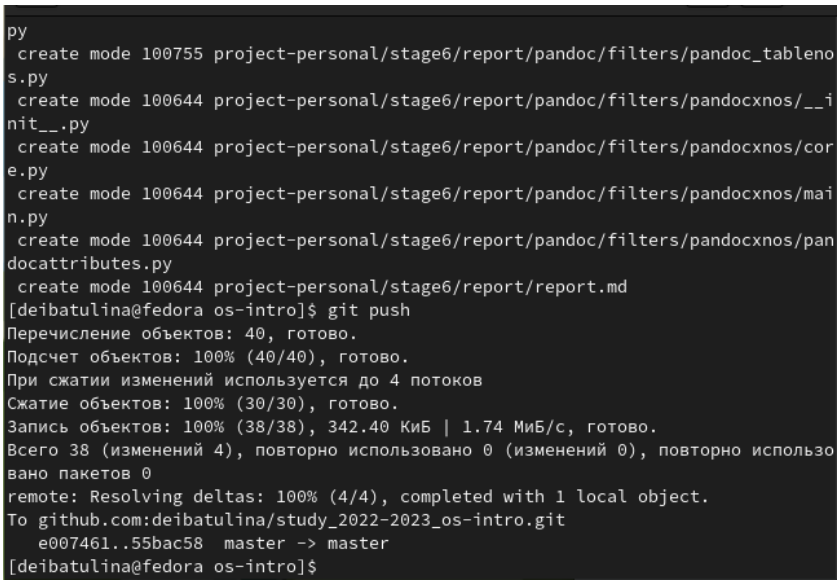

Рис. 4.13: Отправка файлов на сервер

Далее, проверяем работу команды make (она отвечает за конвертацию файла в markdown в docx и pdf), а также команды make clean (она отвечает за удаление docx и pdf файлов, если в них есть какие-либо недочёты) (рис. 4.14):

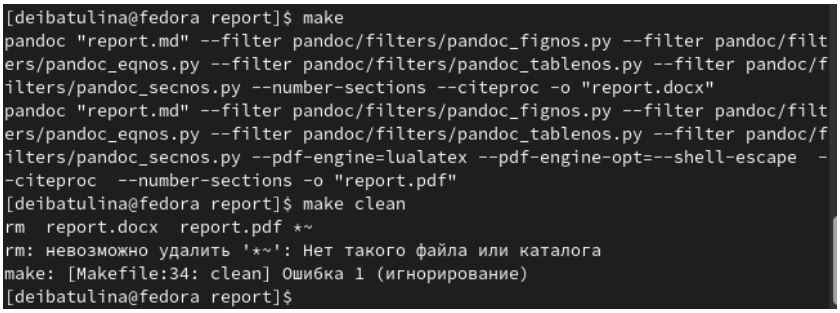

Рис. 4.14: Проверка работы команд make и make clean

Всё работает! Значит, всё настроено правильно.

5 Выводы

Я научилась работать с системой контроля версий Git. Узнала о ней новую информацию, сделала базовую настройку, научилась базовым командам.

6 Ответы на контрольные вопросы

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

Система контроля версий - это системы, записывающие все внесённые в них изменения. Они активно применяются в разработке программ, совместных проектов, когда несколько участников работают над выполнением одного проекта и каждый выполняет свою задачу. Каждый участник получает доступ к системе контроля версий и репозиторию, где расположен проект, и вносит туда свои изменения. [1]

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

Хранилище - это большое пространство, где хранятся данные. Commit - запись внесённого изменения. История - это последовательность всех вносимых в проект изменений. Рабочая копия - снимок какого-либо этапа проекта.

В хранилище расположено много проектов, данных и т.д. Получая доступ к нему, участник проекта вносит в него изменения (commit), эти изменения вносятся в историю, сохраняется рабочая копия проекта (проект на этапе какого-либо изменения).

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

Централизованная система контроля версий предназначена для решения основной проблемы локальной системы контроля версий. Для организации такой системы контроля версий используется единственный сервер, который содержит все версии файлов. Если с сервером возникнут какие-либо неполадки, то это грозит потерей всех данных (CVS, Subversion, Perforce).

В децентрализованных системах контроля версий при каждом копировании удалённого репозитория (расположенного на сервере) происходит полное копирование данных в локальный репозиторий (установленный на рабочем компьютере). Каждая копия содержит все данные, хранящиеся в удалённом репозитории. В случае, возникновения технической неисправности на стороне сервера, удаленный репозиторий можно перезаписать с любой сохраненной копии (Git). [2]

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

Для начала создаём удалённый репозиторий и подключаем его к основному. Затем вносим изменения в проект посредством локального репозитория, и отправляем внесённые изменения на сервер.

5. Опишите порядок работы с общим хранилищем VCS.

Посредством определённых команд пользователь получает нужную ему версию проекта, вносит туда необходимые изменения, отправляет файлы на сервер (при этом предыдущие и последующие версии не удаляются из общего репозитория).

6. Каковы основные задачи, решаемые инструментальным средством git?

Внесение изменений в проект несколькими участниками, просмотр истории изменений, возможность вернуться к любой версии проекта, при этом другие версии не удаляются.

7. Назовите и дайте краткую характеристику командам git.

Перечислим 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` Стандартные процедуры работы при наличии центрального репозитория Работа пользователя со своей веткой начинается с проверки и получения изменений из центрального репозитория (при этом в локальное дерево до начала этой процедуры не должно было вноситься изменений):

`git checkout master git pull git checkout -b имя_ветки`

Затем можно вносить изменения в локальном дереве и/или ветке.

После завершения внесения какого-то изменения в файлы и/или каталоги проекта необходимо разместить их в центральном репозитории. Для этого необходимо проверить, какие файлы изменились к текущему моменту:

`git status`

При необходимости удаляем лишние файлы, которые не хотим отправлять в центральный репозиторий.

Затем полезно просмотреть текст изменений на предмет соответствия правилам ведения чистых коммитов:

`git diff`

Если какие-либо файлы не должны попасть в коммит, то помечаем только те файлы, изменения которых нужно сохранить. Для этого используем команды добавления и/или удаления с нужными опциями:

`git add ...` `git rm ...` Если нужно сохранить все изменения в текущем каталоге, то используем:

```
git add .
```

Затем сохраняем изменения, поясняя, что было сделано:

```
git commit -am "Some commit message"
```

Отправляем изменения в центральный репозиторий:

```
git push origin имя_ветки
```

или

`git push` [3]

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

git push /-all (origin master/любая ветка)

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

Ветки нужны для удобства работы над разными проектами. Однако, ветки можно объединять (посредством построения соединений между ними).

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

Во время выполнения проекта могут создаваться файлы, которые не нужно добавлять в проект (например, объектные файлы, создаваемые компиляторами при написании программ, различные архивы с исходными материалами).

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.
3. Системы контроля версий [Электронный ресурс]. 2016. URL: http://uii.mpei.ru/study/courses