

Лабораторная работа №2

Доберштейн Алина Сергеевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Установка программного обеспечения	9
4.2	Базовая настройка git.	10
4.3	Создание ключей SSH	10
4.4	Создание ключей PGP	11
4.5	Настройка GitHub	12
4.6	Добавление PGP ключа на GitHub	13
4.7	Настройка автоматических подписей коммитов git	14
4.8	Настройка gh	14
4.9	Шаблон для рабочего пространства	15
4.9.1	Создание репозитория курса на основе шаблона	15
4.9.2	Настройка каталога курса	15
5	Выводы	18
6	Контрольные вопросы	19
	Список литературы	23

Список иллюстраций

4.1	Установка git	9
4.2	Установка gh	9
4.3	Владелец репозитория	10
4.4	Задание имени начальной ветки	10
4.5	Параметры autocrlf, safecrlf	10
4.6	SSH-ключ по алгоритму rsa с размером 4096 бит	11
4.7	SSH-ключ по алгоритму ed25519	11
4.8	Генерирование PGP-ключа	12
4.9	Генерирование PGP-ключа	12
4.10	Учетная запись	13
4.11	Вывод списка ключей и копирование отпечатка приватного ключа	13
4.12	Копирование ключа в буфер обмена	13
4.13	PGP-ключ в GitHub	14
4.14	Настройка автоматических подписей коммитов	14
4.15	Авторизация	14
4.16	Создание шаблона рабочего пространства	15
4.17	Создание шаблона рабочего пространства	15
4.18	Переход в каталог курса	15
4.19	Удаление лишних файлов	16
4.20	Создание необходимых каталогов	16
4.21	Отправка файлов на сервер	16
4.22	Репозиторий на GitHub	17

Список таблиц

1 Цель работы

Изучить идеологию и применения средств контроля версий, освоить умения по работе с git.

2 Задание

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

3 Теоретическое введение

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется. В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зави-

симости 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

4.1 Установка программного обеспечения

Установила git, перейдя на роль суперпользователя. (рис. 4.1).

```
[asdoershteyjn@fedora ~]$ sudo -i
[sudo] пароль для asdoershteyjn:
[root@fedora ~]# dnf install git
Последняя проверка окончания срока действия метаданных: 0:07:48 назад, Ср 15 фев
2023 16:28:22.
Пакет git-2.39.1-1.fc37.x86_64 уже установлен.
Зависимости разрешены.
Отсутствуют действия для выполнения.
Выполнено!
```

Рис. 4.1: Установка git

Установила gh (рис. 4.2).

```
[root@fedora ~]# dnf install gh
Последняя проверка окончания срока действия метаданных: 0:08:00 назад, Ср 15 фев
2023 16:28:22.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Резепозиторий  Размер
=====
Установка:
gh          x86_64       2.22.1-1.fc37  updates        8.3 М
=====
Результат транзакции
=====
Установка 1 Пакет

Объем загрузки: 8.3 М
Объем изменений: 42 М
Продолжить? [д/н]:
```

Рис. 4.2: Установка gh

4.2 Базовая настройка git.

Задала имя и email владельца репозитория (рис. 4.3).

```
[asdoershteyjn@fedora ~]$ git config --global user.name "Alina Doershteyjn"
[asdoershteyjn@fedora ~]$ git config --global user.email "1132226448@pfur.ru"
```

Рис. 4.3: Владелец репозитория

Настроила utf-8 в выводе сообщений гит (рис. ??).

Настройка utf-8

Задала имя начальной ветки (рис. 4.4).

```
[asdoershteyjn@fedora ~]$ git config --global init.defaultBranch master
```

Рис. 4.4: Задание имени начальной ветки

Параметры autocrlf, safecrlf (рис. 4.5).

```
[asdoershteyjn@fedora ~]$ git config --global core.autocrlf input
[asdoershteyjn@fedora ~]$ git config --global core.safecrlf warn
```

Рис. 4.5: Параметры autocrlf, safecrlf

4.3 Создание ключей SSH

По алгоритму rsa с ключем размером 4096 бит (рис. 4.6).

```
[asdoershteyjn@fedora ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/asdoershteyjn/.ssh/id_rsa):
Created directory '/home/asdoershteyjn/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/asdoershteyjn/.ssh/id_rsa
Your public key has been saved in /home/asdoershteyjn/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:07eYYQEncMBC0r8rF8a9xMticipTSH2xsAqppGSVrNr0 asdoershteyjn@fedora
The key's randomart image is:
+---[RSA 4096]-----+
|.O..000 O|
|.O.. O .|
|.O.. .|
|... . .|
|..+ B S + .|
|..X X O = .|
|..X X O .|
|...E* =|
|. . .|
+---[SHA256]-----+
```

Рис. 4.6: SSH-ключ по алгоритму rsa с размером 4096 бит

По алгоритму ed25519 (рис. 4.7).

```
[asdoershteyjn@fedora ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/asdoershteyjn/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/asdoershteyjn/.ssh/id_ed25519
Your public key has been saved in /home/asdoershteyjn/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:T0GGYlZSC6sdrVHdARLq4IWVtU27DD3WeiDYx0nhvVo asdoershteyjn@fedora
The key's randomart image is:
+--[ED25519 256]--+
|.O*==O..|
|.O*BO.O.|
|.O.+O=+O .|
|.++ ==.*|
|.O.+S =..|
|.O O E.|
|.O.|
|. .|
+---[SHA256]-----+
```

Рис. 4.7: SSH-ключ по алгоритму ed25519

4.4 Создание ключей PGP

Сгенерировала ключ (рис. 4.8), (рис. 4.9).

```
[asdobershteyjn@fedora ~]$ gpg --full-generate-key
gpg (GnuPG) 2.3.8; 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/asdobershteyjn/.gnupg'
gpg: создан шит с ключами '/home/asdobershteyjn/.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
```

Рис. 4.8: Генерирование PGP-ключа

```
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: Alina Dobershteyjn
Адрес электронной почты: 1132226448@pfur.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Alina Dobershteyjn <1132226448@pfur.ru>"
Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход?
```

Рис. 4.9: Генерирование PGP-ключа

4.5 Настройка GitHub

Созданная учетная запись на ГитХабе (рис. 4.10).

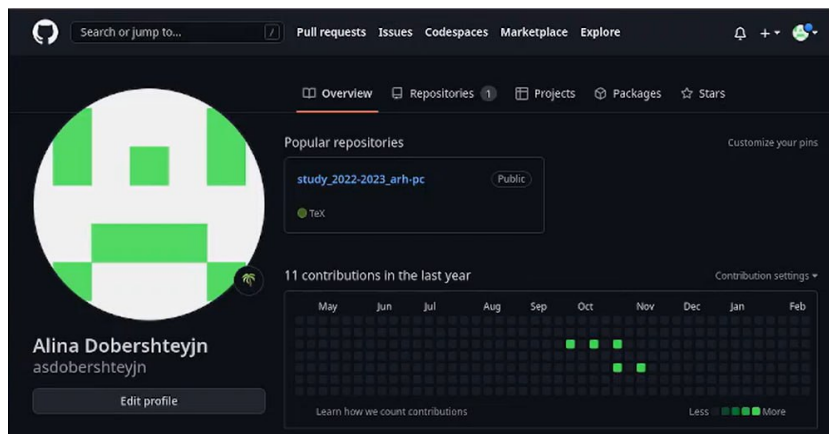


Рис. 4.10: Учетная запись

4.6 Добавление PGP ключа на GitHub

Вывела список ключей и скопировала отпечаток приватного ключа (рис. 4.11).

```
[asdobershteyjn@fedora ~]$ 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/asdobershteyjn/.gnupg/pubring.kbx
-----
sec   rsa4096/AC2421815930299D 2023-02-17 [SC]
      9F6B138E6E0EC42B3AFAC631AC2421815930299D
uid   [ а6солнѣтно ] Alina Dobershteyjn <1132226448@pfur.ru>
ssb   rsa4096/F83D23F1DA1606B5 2023-02-17 [E]
```

Рис. 4.11: Вывод списка ключей и копирование отпечатка приватного ключа

Скопировала сгенерированный PGP-ключ в буфер обмена (рис. 4.12).

```
[asdobershteyjn@fedora ~]$ gpg --armor --export F83D23F1DA1606B5 | xclip -sel clip
[asdobershteyjn@fedora ~]$
```

Рис. 4.12: Копирование ключа в буфер обмена

Перешла в настройки GitHub, вставила полученный ключ (рис. 4.13).

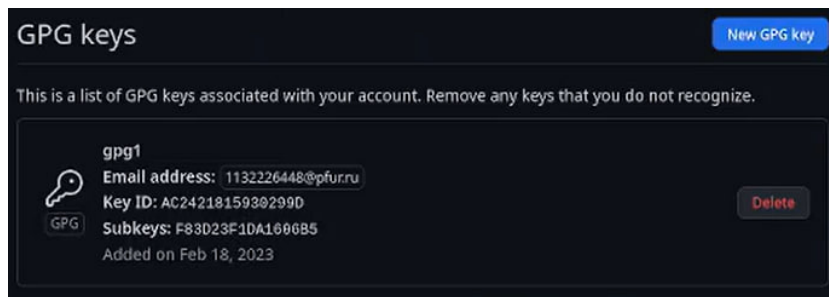


Рис. 4.13: PGP-ключ в GitHub

4.7 Настройка автоматических подписей коммитов git

Используя введенный email, указала Git применять его при подписи коммитов (рис. 4.14).

```
[asdoershteyjn@fedora ~]$ git config --global user.signingkey F83D23F1DA1606B5
[asdoershteyjn@fedora ~]$ git config --global commit.gpgsign true
[asdoershteyjn@fedora ~]$ git config --global gpg.program $(which gpg2)
```

Рис. 4.14: Настройка автоматических подписей коммитов

4.8 Настройка gh

Авторизовалась через браузер(рис. 4.15).

```
[asdoershteyjn@fedora ~]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: BB78-E039
Press Enter to open github.com in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as asdoershteyjn
```

Рис. 4.15: Авторизация

4.9 Шаблон для рабочего пространства

4.9.1 Создание репозитория курса на основе шаблона

Создала шаблон рабочего пространства (рис. 4.16), (рис. 4.17).

```
[asdobershteyjn@fedora ~]$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
[asdobershteyjn@fedora ~]$ cd ~/work/study/2022-2023/"Операционные системы"
[asdobershteyjn@fedora Операционные системы]$ gh repo create study_2022-2023_os-intro --tem
plate=yamadharm/course-directory-student-template --public
```

Рис. 4.16: Создание шаблона рабочего пространства

```
[asdobershteyjn@fedora Операционные системы]$ git clone --recursive git@github.com:asdobers
hteyjn/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:+DiY3wvV6TuJJhbpZisF/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), 16.94 КиБ | 867.00 КиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharm/academic-presentation-mark
down-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharm/academic-laboratory-report-templ
ate.git) зарегистрирован по пути «template/report»
Клонирование в «/home/asdobershteyjn/work/study/2022-2023/Операционные системы/os-intro/tem
plate/presentation»...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
```

Рис. 4.17: Создание шаблона рабочего пространства

4.9.2 Настройка каталога курса

Перешла в каталог курса (рис. 4.18).

```
[asdobershteyjn@fedora Операционные системы]$ cd ~/work/study/2022-2023/"Операционные систе
мы"/os-intro
```

Рис. 4.18: Переход в каталог курса

Удалила лишние файлы (рис. 4.19).


```
[asdoershteyjn@fedora os-intro]$ rm package.json
[asdoershteyjn@fedora os-intro]$ echo os-intro > C
```

Рис. 4.19: Удаление лишних файлов

Создала необходимые каталоги (рис. 4.20).

```
[asdoershteyjn@fedora os-intro]$ echo os-intro > COURSE
[asdoershteyjn@fedora os-intro]$ make
```

Рис. 4.20: Создание необходимых каталогов

Отправила файлы на сервер (рис. 4.21).

```
[asdoershteyjn@fedora os-intro]$ git add .
[asdoershteyjn@fedora os-intro]$ git commit -am 'feat(main): make course structure'
[master b668195] 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.py
create mode 100644 labs/lab01/report/report.md
create mode 100644 labs/lab02/presentation/Makefile
```

Рис. 4.21: Отправка файлов на сервер

Репозиторий на GitHub (рис. 4.22).

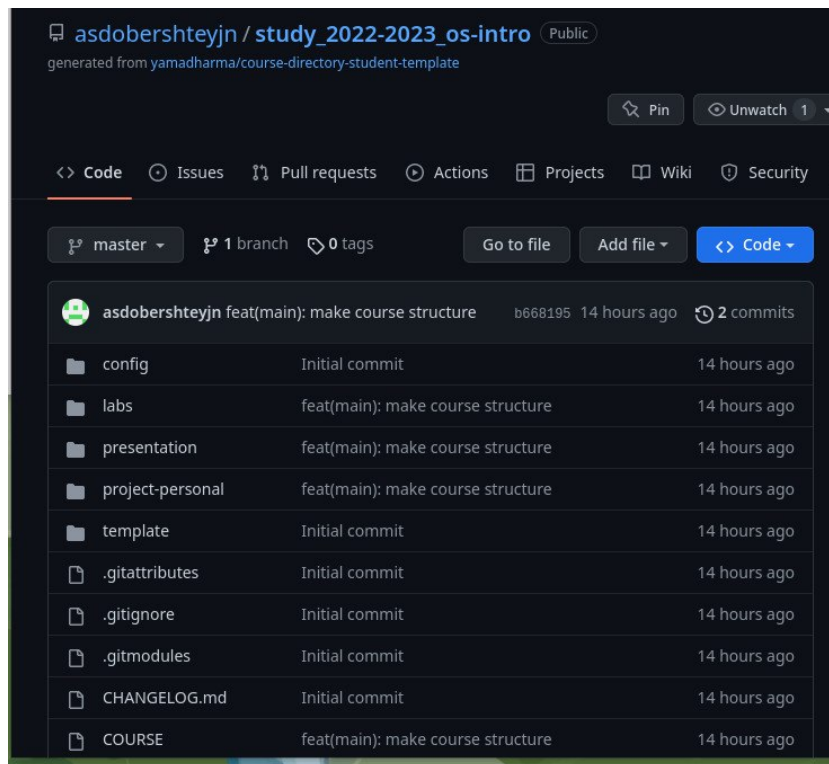


Рис. 4.22: Репозиторий на GitHub

5 Выводы

В ходе данной лабораторной работы я изучила идеологию и применения средств контроля версий, освоила умения по работе с git.

6 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. 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] - Режим доступа: <https://git-scm.com/book/ru/v2/Введение-О-системе-контроля-версий>
2. Системы контроля версий [Электронный ресурс] - Режим доступа: http://uii.mpei.ru/study/courses/sdt/16/lecture02.2_vcs.slides.pdf.