

Лабораторная работа 2

Управление версиями

Арина Олеговна Аристова

Содержание

Цель работы	3
Задание	4
Теоретическое введение	5
Выполнение лабораторной работы	7
Вывод	15
Ответы на контрольные вопросы	16

Цель работы

- Изучить идеологию и применение средств контроля версий.
- Освоить 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.

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 дельтакомпрессию — 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 на-

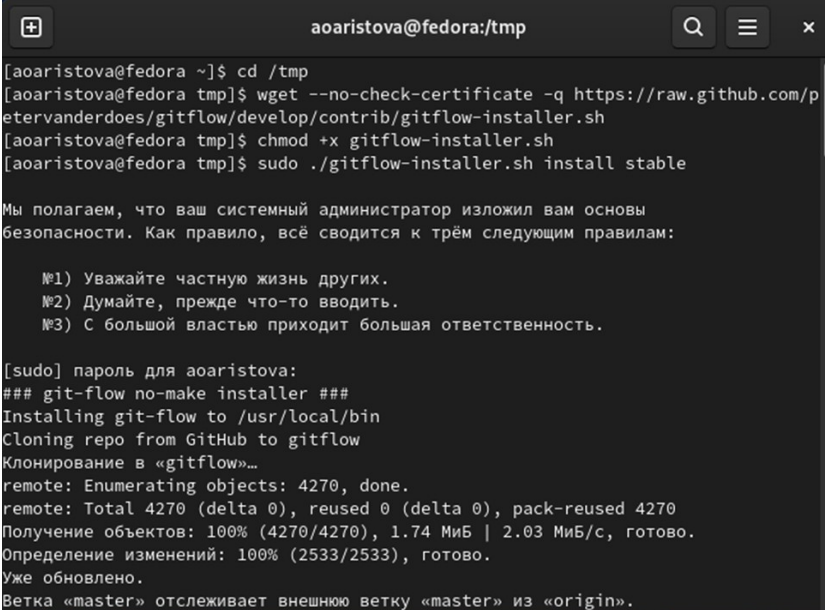
строек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.

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.

Выполнение лабораторной работы

Предварительно создаю учетную запись и заполняю основные данные на <https://github.com>. Устанавливаю git-flow в Fedora Linux.



```
aoaristova@fedora:~/tmp
[aoaristova@fedora ~]$ cd /tmp
[aoaristova@fedora tmp]$ wget --no-check-certificate -q https://raw.githubusercontent.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh
[aoaristova@fedora tmp]$ chmod +x gitflow-installer.sh
[aoaristova@fedora 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 aoaristova:
### 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 | 2.03 МиБ/с, готово.
Определение изменений: 100% (2533/2533), готово.
Уже обновлено.
Ветка «master» отслеживает внешнюю ветку «master» из «origin».
```

Рис. 1: Установка git-flow

Устанавливаю gh в Fedora Linux.

```
aoaristova@fedora:~ — sudo dnf install gh
[aoaristova@fedora ~]$ sudo dnf install gh
[sudo] пароль для aoaristova:
Fedora 35 openh264 (From Cisco) - x86_64      1.5 kB/s | 2.5 kB    00:01
Fedora Modular 35 - 93% [===== ] 518 kB/s | 3.2 MB    00:00 ETA
```

Рис. 2: Установка gh

```
aoaristova@fedora:~ — sudo dnf install gh
Установка:
gh          x86_64          2.7.0-1.fc35          updates          6.8 M

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          5.9 MB/s | 6.8 MB    00:01
-----
Общий размер          2.5 MB/s | 6.8 MB    00:02
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 скриптлета: gh-2.7.0-1.fc35.x86_64 1/1
Проверка             : gh-2.7.0-1.fc35.x86_64 1/1
```

Рис. 3: Процесс установки gh

Выполняю базовую настройку git.


```
[aoaristova@fedora ~]$ git config --global user.name "Arina Aristova"
[aoaristova@fedora ~]$ git config --global user.email "aristovarina@mail.ru"
[aoaristova@fedora ~]$ git config --global core.quotepath false
[aoaristova@fedora ~]$
```

Рис. 4: Выполнение базовой настройки git

Настраиваю utf-8 в выводе сообщений git

```
[aoaristova@fedora ~]$ git config --global user.name "Arina Aristova"
[aoaristova@fedora ~]$ git config --global user.email "aristovarina@mail.ru"
[aoaristova@fedora ~]$ git config --global core.quotepath false
[aoaristova@fedora ~]$
```

Рис. 5: Настройка utf-8 в выводе сообщений git

Настраиваю верификацию и подписание коммитов git. Задаю имя начальной ветки master, параметр autocrlf, параметр safecrlf.

```
[aoaristova@fedora ~]$ git config --global init.defaultBranch master
[aoaristova@fedora ~]$ git config --global core.autocrlf input
[aoaristova@fedora ~]$ git config --global core.safecrlf warn
[aoaristova@fedora ~]$
```

Рис. 6: Настройка верификацию и подписания коммитов git

Создаю ключи ssh:

-по алгоритму rsa с ключём размером 4096 бит;

```
[aoaristova@fedora ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/aoaristova/.ssh/id_rsa):
Created directory '/home/aoaristova/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/aoaristova/.ssh/id_rsa
Your public key has been saved in /home/aoaristova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:/V6A498gg1keg3IDH9wmkEiNLkHUJRpp113jI0mzQyc aoaristova@fedora
The key's randomart image is:
+---[RSA 4096]-----+
|o+oB=o|
|.oo=o . E +|
|o. . = * 0 .|
|. .. + * B +|
|. o = S * o|
| o * + o .|
| o o o o .|
|. + +|
| o .|
+---[SHA256]-----+
[aoaristova@fedora ~]$
```

Рис. 7: Создание ключа ssh по алгоритму rsa с ключём размером 4096 бит

- по алгоритму ed25519.

```
[aoaristova@fedora ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/aoaristova/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/aoaristova/.ssh/id_ed25519
Your public key has been saved in /home/aoaristova/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:m1KFfNskgkyNCVIZwsBxt470MGi+3XkqBt7qGdMsnwc aoaristova@fedora
The key's randomart image is:
+---[ED25519 256]---+
|*+Bo.. .|
|=*+. ....|
|+. + o ...|
|..= o.|
|o.o....S|
|o.oE=. o|
|.ooooo|
|.o+oo..|
|.o+o+o|
+---[SHA256]-----+
[aoaristova@fedora ~]$
```

Рис. 8: Создание ключа ssh по алгоритму ed25519

Создаю ключ ргр. Генерирую ключ. Из предложенных опций выбираю: тип RSA and RSA; размер 4096; срок действия - 0 (срок действия не истекает никогда), примечание оставляю пустым.

```
[aoaristova@fedora ~]$ 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/aoaristova/.gnupg'
gpg: создан щит с ключами '/home/aoaristova/.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
```

Рис. 9: Создание ключа gpg

```
GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: Арина Аристова
Адрес электронной почты: aristovarina@mail.ru
Примечание:
Используется таблица символов 'utf-8'.
Вы выбрали следующий идентификатор пользователя:
  "Арина Аристова <aristovarina@mail.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
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/aoaristova/.gnupg/trustdb.gpg: создана таблица доверия
gpg: ключ 11CAB8A54C14EB8C помечен как абсолютно доверенный
gpg: создан каталог '/home/aoaristova/.gnupg/openpgp-revocs.d'
```

Рис. 10: Создание ключа gpg.2

Добавляю PGP ключ в GitHub: Вывожу список ключей и копирую отпечаток приватного ключа.

```
aoaristova@fedora:~  
[aoaristova@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/aoaristova/.gnupg/pubring.kbx  
-----  
sec  rsa4096/11CAB8A54C14EB8C 2022-04-22 [SC]  
      393FFC41909F5EA8F9EF362411CAB8A54C14EB8C  
uid          [ абсолютно ] Арина Аристова <aristovarina@mail.ru>  
ssb  rsa4096/9D5C9A827E495759 2022-04-22 [E]  
  
[aoaristova@fedora ~]$
```

Рис. 11: Копирование отпечатка ключа

Копирую сгенерированный PGP ключ в буфер обмена и вставляю полученный ключ в поле ввода в GitHub.

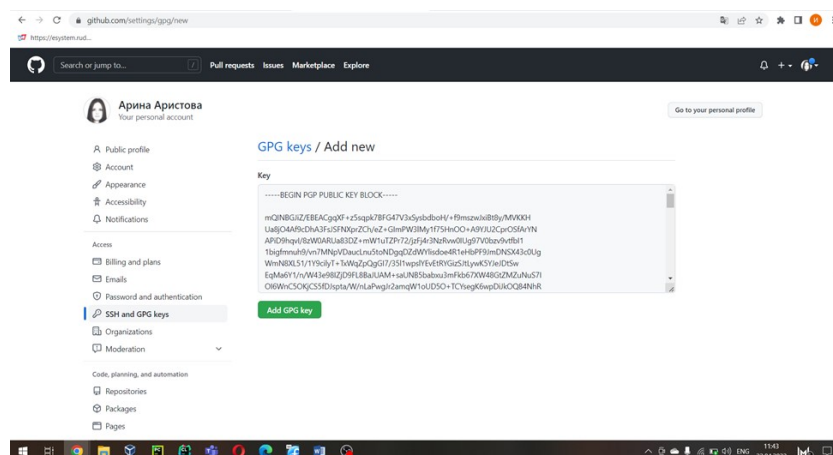


Рис. 12: Добавление PGP ключа в GitHub

Настраиваю автоматические подписи коммитов git. Используя введенный email, указываю Git применять его при подписи коммитов.

```
[aoaristova@fedora ~]$ git config --global user.signingkey 11CAB8A54C14EB8C
[aoaristova@fedora ~]$ git config --global commit.gpgsign true
[aoaristova@fedora ~]$ git config --global gpg.program $(which gpg2)
[aoaristova@fedora ~]$
```

Рис. 13: Настройка автоматических подписей коммитов git

Настраиваю gh: Авторизуюсь через браузер.

```
[aoaristova@fedora Операционные системы]$ 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: B6E0-1089
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 aoaristova
[aoaristova@fedora Операционные системы]$ gh repo create study_2021-2022_os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository aoaristova/study_2021-2022_os-intro on GitHub
[aoaristova@fedora Операционные системы]$ git clone --recursive git@github.com:<owner>/study_2021-2022_os-intro.git
os-intro
bash: owner: Нет такого файла или каталога
[aoaristova@fedora Операционные системы]$ git clone --recursive git@github.com:aoaristova/study_2021-2022_os-intro.git
git os-intro
Клонирование в «os-intro»...
git@github.com: Permission denied (publickey).
fatal: Не удалось прочитать из внешнего репозитория.

Удостоверьтесь, что у вас есть необходимые права доступа
и репозиторий существует.
[aoaristova@fedora Операционные системы]$ gh auth login

[aoaristova@fedora Операционные системы]$ gh repo create study_2021-2022_os-intro --template=yamadharma/course-directory-student-template --public
GraphQL: Could not clone: Name already exists on this account (cloneTemplateRepository)
[aoaristova@fedora Операционные системы]$ ls
[aoaristova@fedora Операционные системы]$
```

Рис. 14: Авторизация

Создаю репозиторий курса на основе шаблона.

```
[aoaristova@fedora Операционные системы]$ gh repo create study_2021-2022_os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository aoaristova/study_2021-2022_os-intro on GitHub
[aoaristova@fedora Операционные системы]$ git clone --recursive git@github.com:<owner>/study_2021-2022_os-intro.git
os-intro
```

Рис. 15: Переход в директорию

```
[aoaristova@fedora ~]$ cd ~/work/study/2021-2022/"Операционные системы"
```

Рис. 16: Создание репозитория

```
[aoaristova@fedora Операционные системы]$ git clone --recursive https://github.com/aoaristova/study_2021-2022_os-intro.git
Клонирование в «study_2021-2022_os-intro»...
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 20 (delta 2), reused 15 (delta 2), pack-reused 0
Получение объектов: 100% (20/20), 12.51 КиБ | 2.50 МиБ/с, готово.
Определение изменений: 100% (2/2), 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/aoaristova/work/study/2021-2022/Операционные системы/study_2021-2022_os-intro/template/presentation»...
remote: Enumerating objects: 42, done.
remote: Counting objects: 100% (42/42), done.
remote: Compressing objects: 100% (34/34), done.
remote: Total 42 (delta 9), reused 40 (delta 7), pack-reused 0
Получение объектов: 100% (42/42), 31.19 КиБ | 679.00 КиБ/с, готово.
Определение изменений: 100% (9/9), готово.
Клонирование в «/home/aoaristova/work/study/2021-2022/Операционные системы/study_2021-2022_os-intro/template/report»...
remote: Enumerating objects: 78, done.
remote: Counting objects: 100% (78/78), done.
remote: Compressing objects: 100% (52/52), done.
remote: Total 78 (delta 31), reused 69 (delta 22), pack-reused 0
Получение объектов: 100% (78/78), 292.27 КиБ | 1.99 МиБ/с, готово.
Определение изменений: 100% (31/31), готово.
Подмодуль по пути «template/presentation»: забрано состояние «3eae6b7586f8a9aded2b596cd1018e625b228b93»
Подмодуль по пути «template/report»: забрано состояние «df7b2ef80f8def3b9a496f8695277469a1a7842a»
[aoaristova@fedora Операционные системы]$
```

Рис. 17: Клонирование репозитория

Настраиваю каталог курса. Перехожу в каталог курса, удаляю лишние файлы, создаю необходимые каталоги и отправляю файлы на сервер.

```
[aoaristova@fedora Операционные системы]$ ls
study_2021-2022_os-intro
[aoaristova@fedora Операционные системы]$ cd study_2021-2022_os-intro
[aoaristova@fedora study_2021-2022_os-intro]$ ls
config LICENSE Makefile package.json README.en.md README.git-flow.md README.md template
[aoaristova@fedora study_2021-2022_os-intro]$ rm package.json
[aoaristova@fedora study_2021-2022_os-intro]$ ls
config LICENSE Makefile README.en.md README.git-flow.md README.md template
[aoaristova@fedora study_2021-2022_os-intro]$ make COURSE=os-intro
[aoaristova@fedora study_2021-2022_os-intro]$ git add .
[aoaristova@fedora study_2021-2022_os-intro]$ S
```

Рис. 18: Настройка каталога курса

Вывод

я изучила идеологию применения средств контроля версий, а также освоила умения по работе с git.

Ответы на контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначены? VCS – (Version Control System) системы контроля версий. Они 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.
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Хранилище – место расположения файлов и папок проекта, изменения в которых отслеживаются. Commit – операция, предполагающая отправку в репозиторий изменений, которые пользователь внес в свою рабочую копию. Рабочая копия - текущее состояние файлов проекта, полученных из хранилища и, возможно, изменённых, то есть разработчик имеет в распоряжении именно рабочую копию и с ней работает.
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные VCS имеют одно основное хранилище. Каждый пользователь копирует себе необходимые файлы из основного репозитория, изменяет их, а затем добавляет изменения обратно (Subversion). Децентрализованные VCS устроены так, что каждый пользователь имеет свой (или даже не один) репозиторий. Пользователь может добавлять и забирать изменения из любого репозитория (Git).

Соответственно, количеством основных репозиторий и различаются централизованные и децентрализованные VCS.

4. Опишите действия с VCS при единоличной работе с хранилищем. Разработчик работает с веткой master, при необходимости может создать ветки для отдельных частей проекта. При завершении изменений разработчик коммитит (commit) и пушит (push) их, то есть сохраняет изменения в общем хранилище.
5. Опишите порядок работы с общим хранилищем VCS. Каждый разработчик проекта работает над отдельной частью проекта в своей ветке. После завершения изменений разработчик коммитит (commit) и пушит (push) изменения на сервер. После окончания работы необходимо смирджить(merge), то есть выполнить слияние веток, например, с главной веткой. Также разработчик может работать с изменениями, сделанными другим разработчиком, если на одной ветке их работает несколько.
6. Каковы основные задачи, решаемые инструментальным средством git? Основные задачи git заключаются в удобной командной работе над проектом, а также в хранении информации обо всех изменениях проекта.
7. Назовите и дайте краткую характеристику командам git. git init – создание основного дерева репозитория; git pull - получение обновлений (изменений) текущего дерева из центрального репозитория; git push - отправка всех произведённых изменений локального дерева в центральный репозиторий; git status - просмотр списка изменённых файлов в текущей директории; git diff - просмотр текущих изменений; git add - сохранение текущих изменений: добавить все изменённые и/или созданные файлы и/или каталоги; git add имена_файлов - сохранение текущих изменений: добавить конкретные изменённые и/или созданные файлы и/или каталоги; git rm имена_файлов - удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории); git commit -am 'Описание коммита' - сохранение добавленных изменений: сохранить все добавленные изменения и все изменённые

файлы; `git commit` - сохранение добавленных изменений: сохранить добавленные изменения с внесением комментария через встроенный редактор; `git checkout -b имя_ветки` - создание новой ветки, базирующейся на текущей; `git checkout имя_ветки` - переключение на некоторую ветку; `git push origin имя_ветки` - отправка изменений конкретной ветки в центральный репозиторий; `git merge --no-ff имя_ветки` - слияние ветки с текущим деревом; `git branch -d имя_ветки` - удаление локальной уже слитой с основным деревом ветки; `git branch -D имя_ветки` - принудительное удаление локальной ветки; `git push origin :имя_ветки` - удаление ветки с центрального репозитория.

8. Что такое и зачем могут быть нужны ветви (branches)? Ветка – указатель на один из коммитов. Ветки используются для разработки одной части проекта отдельно от других его частей. Каждая ветка представляет собой отдельную копию кода проекта. Ветки позволяют работать одновременно над разными версиями проекта.
9. Как и зачем можно игнорировать некоторые файлы при commit? Игнорировать файлы при commit можно с помощью файла `.gitignore`. Туда обычно помещаются файлы, которые не нужны для проекта, например, временные файлы, создаваемые редакторами.