РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

дисциплина: Операционные системы

Студент: Матюшкин Денис Владимирович

Группа: НПИбд-02-21

МОСКВА

2022 г.

Цель работы:

– Изучить идеологию и применение средств контроля версий.

– Освоить умения по работе с git.

Ход работы:

1. Создадим учетную версию на https://github.com и заполним основные данные (рис. 1). Примечание: я ранее регистрировался на данном сайте, поэтому не буду повторяться.

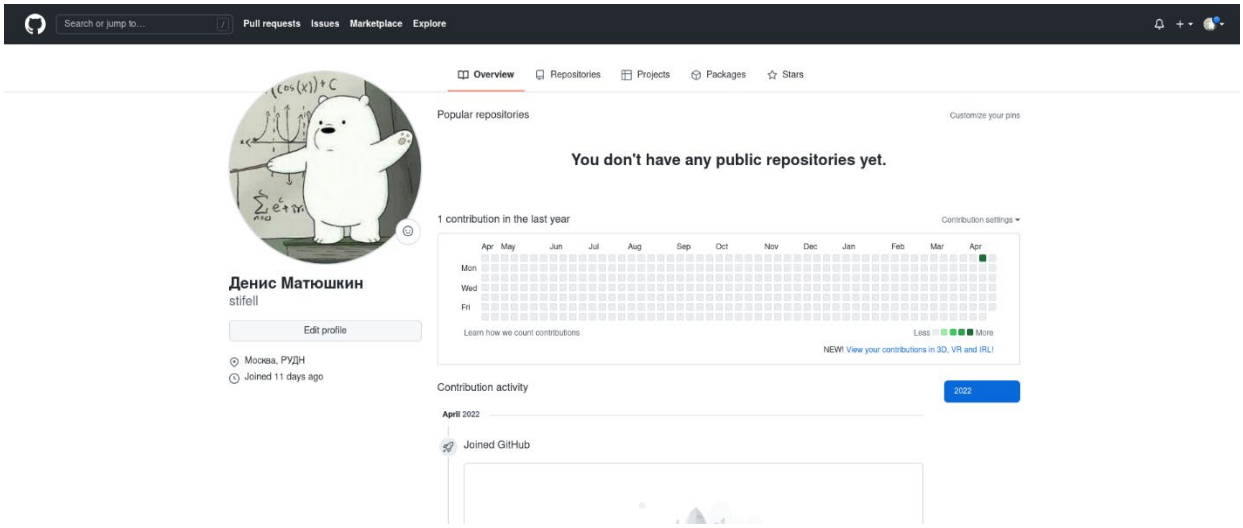

Рис. 1: Создание учетной записи на Github

2. Установим программное обеспечение git-flow через терминал (рис. 2).

```
[dvmatyushkin@dvmatyushkin ~]$ cd /tmp
[dvmatyushkin@dvmatyushkin tmp]$ wget --no-check-certificate -q https://raw.github.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh
[dvmatyushkin@dvmatyushkin tmp]$ chmod +x gitflow-installer.sh
[dvmatyushkin@dvmatyushkin tmp]$ sudo ./gitflow-installer.sh install stable
[sudo] пароль для dvmatyushkin:
### 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 | 971.00 КиБ/с, готово.
Определение изменений: 100% (2533/2533), готово.
Уже обновлено.
Ветка «master» отслеживает внешнюю ветку «master» из «origin».
Переключено на новую ветку «master»
install: создание каталога '/usr/local/share/doc'
install: создание каталога '/usr/local/share/doc/gitflow'
install: создание каталога '/usr/local/share/doc/gitflow/hooks'
'gitflow/git-flow' -> '/usr/local/bin/git-flow'
'gitflow/git-flow-init' -> '/usr/local/bin/git-flow-init'
'gitflow/git-flow-feature' -> '/usr/local/bin/git-flow-feature'
'gitflow/git-flow-bugfix' -> '/usr/local/bin/git-flow-bugfix'
'gitflow/git-flow-hotfix' -> '/usr/local/bin/git-flow-hotfix'
'gitflow/git-flow-release' -> '/usr/local/bin/git-flow-release'
'gitflow/git-flow-support' -> '/usr/local/bin/git-flow-support'
'gitflow/git-flow-version' -> '/usr/local/bin/git-flow-version'
'gitflow/gitflow-common' -> '/usr/local/bin/gitflow-common'
'gitflow/gitflow-shFlags' -> '/usr/local/bin/gitflow-shFlags'
```

Рис. 2: Установка git-flow на Linux Fedora

3. Установим программное обеспечение gh через терминал (рис. 3).

```
[dvmatyushkin@dvmatyushkin tmp]$ sudo dnf install gh
Последняя проверка окончания срока действия метаданных: 0:39:57 назад, Пт 22 апр 2022 00:54:43.
Зависимости разрешены.
================================================================================================================================
 Пакет                  Архитектура              Версия                        Репозиторий                Размер
================================================================================================================================
Установка:
 gh                     x86_64                   2.7.0-1.fc35                  updates                    6.8 M

Результат транзакции
================================================================================================================================
Установка  1 Пакет

Объем загрузки: 6.8 M
Объем изменений: 32 M
Продолжить? [д/Н]: y
Загрузка пакетов:
gh-2.7.0-1.fc35.x86_64.rpm                                                      4.4 MB/s | 6.8 MB     00:01
--------------------------------------------------------------------------------------------------------------------------------
Общий размер                                                                    2.7 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

Установлен:
  gh-2.7.0-1.fc35.x86_64

Выполнено!
```

Рис. 3: Установка gh на Linux Fedora

4. Совершим базовую настройку git (рис. 4):

    1) Зададим имя и email владельца репозитория.

    2) Настроим utf-8 в выводе сообщений git.

    3) Настроим верификацию и подписание коммитов git.

    4) Зададим имя начальной ветки (будем называть её master).

    5) Зададим параметр autocrlf и safecrlf.

```
[dvmatyushkin@dvmatyushkin tmp]$ git config --global user.name "Denis Matyushkin"
[dvmatyushkin@dvmatyushkin tmp]$ git config --global user.email "matyushkin_d@list.ru"
[dvmatyushkin@dvmatyushkin tmp]$ git config --global core.quotepath false
[dvmatyushkin@dvmatyushkin tmp]$ git config --global init.defaultBranch master
[dvmatyushkin@dvmatyushkin tmp]$ git config --global core.autocrlf input
[dvmatyushkin@dvmatyushkin tmp]$ git config --global core.safecrlf warn
[dvmatyushkin@dvmatyushkin tmp]$
```

Рис. 4: Базовая настройка git

5. Создадим ключ SSH (рис. 5): 1) по алгоритму rsa с ключём размером 4096 бит; 2) по алгоритму ed25519.

```
[dvmatyushkin@dvmatyushkin tmp]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/dvmatyushkin/.ssh/id_rsa):
Created directory '/home/dvmatyushkin/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/dvmatyushkin/.ssh/id_rsa
Your public key has been saved in /home/dvmatyushkin/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:m3H/vT/9qX0aOGs1W1wsRw9XbRQ/3A5SYWyMmgJJJD4 dvmatyushkin@dvmatyushkin
The key's randomart image is:
+---[RSA 4096]----+
|    .oo.     +++*|
|   . .o     .o* B|
|    E  .   o...O+|
|     .  . o  ..o*|
|        S..    +o|
|         = . .o o|
|        o   +..+.|
|            .+o++|
|           ...+=X|
+----[SHA256]-----+
[dvmatyushkin@dvmatyushkin tmp]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/dvmatyushkin/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/dvmatyushkin/.ssh/id_ed25519
Your public key has been saved in /home/dvmatyushkin/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:rmfDdpww1JNe8TIUaGhZUYWhnHgno7IvotZW0wI9Ix8 dvmatyushkin@dvmatyushkin
The key's randomart image is:
+--[ED25519 256]--+
|         +o++=.  |
|        +ooo+    |
|     . ..oBo.o   |
|    o E .o+++ .  |
|     +.*S. o o   |
|      =++ .      |
|   . ..+.+ .     |
|  . + .o* +      |
| ..o ..=.o       |
+----[SHA256]-----+
[dvmatyushkin@dvmatyushkin tmp]$
```

Рис. 5: Создание ключа ssh

6. Создадим ключ GPG (рис. 6.1 и 6.2). Выберем опции, описанные в лабораторной:

– тип RSA and RSA;

– размер 4096;

– выберите срок действия; значение по умолчанию — 0

– GPG запросит личную информацию, которая сохранится в ключе.

– Имя.

– Адрес электронной почты.

```
[dvmatyushkin@dvmatyushkin 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.

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: Denis
Адрес электронной почты: matyushkin_d@list.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
    "Denis <matyushkin_d@list.ru>"

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

Рис. 6.1: Создание ключа gpg

```
gpg: /home/dvmatyushkin/.gnupg/trustdb.gpg: создана таблица доверия
gpg: ключ A7BCA013E5F9D27B помечен как абсолютно доверенный
gpg: создан каталог '/home/dvmatyushkin/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/dvmatyushkin/.gnupg/openpgp-revocs.d/DE9ECBD7892C15F6F1DDA4DDA7BCA013E5F9D27B.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2022-04-21 [SC]
      DE9ECBD7892C15F6F1DDA4DDA7BCA013E5F9D27B
uid                      Denis <matyushkin_d@list.ru>
sub   rsa4096 2022-04-21 [E]
```

Рис. 6.2: Продолжение вывода

7. Выводим список ключей и копируем отпечаток приватного ключа (рис. 7). Отпечаток ключа – A7BCA013E5F9D27B

```
[dvmatyushkin@dvmatyushkin ~]$ gpg --list-secret-keys --keyid-format LONG
/home/dvmatyushkin/.gnupg/pubring.kbx
------------------------------------
sec   rsa4096/A7BCA013E5F9D27B 2022-04-21 [SC]
      DE9ECBD7892C15F6F1DDA4DDA7BCA013E5F9D27B
uid                 [  абсолютно ] Denis <matyushkin_d@list.ru>
ssb   rsa4096/C38BC8F74CE53312 2022-04-21 [E]

[dvmatyushkin@dvmatyushkin ~]$
```

Рис. 7: Вывод списка приватных ключей

8. Скопируем наш сгенерированный GPG ключ в буфер обмена (рис. 8.1) и вставим его в настройках личнего кабинета Github (рис. 8.2).

```
[dvmatyushkin@dvmatyushkin tmp]$ gpg --armor --export A7BCA013E5F9D27B | xclip -sel clip
[dvmatyushkin@dvmatyushkin tmp]$
```

Рис. 8.1: Копирования gpg ключа в буфер обмена

9. Сгенерированный SSH ключ и скопируем его в буфер обмена (рис. 9) и вставим его в настройках личнего кабинета Github (рис. 8.2).

```
[dvmatyushkin@dvmatyushkin tmp]$ cat ~/.ssh/id_rsa.pub | xclip -sel clip
[dvmatyushkin@dvmatyushkin tmp]$
```

Рис. 9: Копирования ssh ключа в буфер обмена

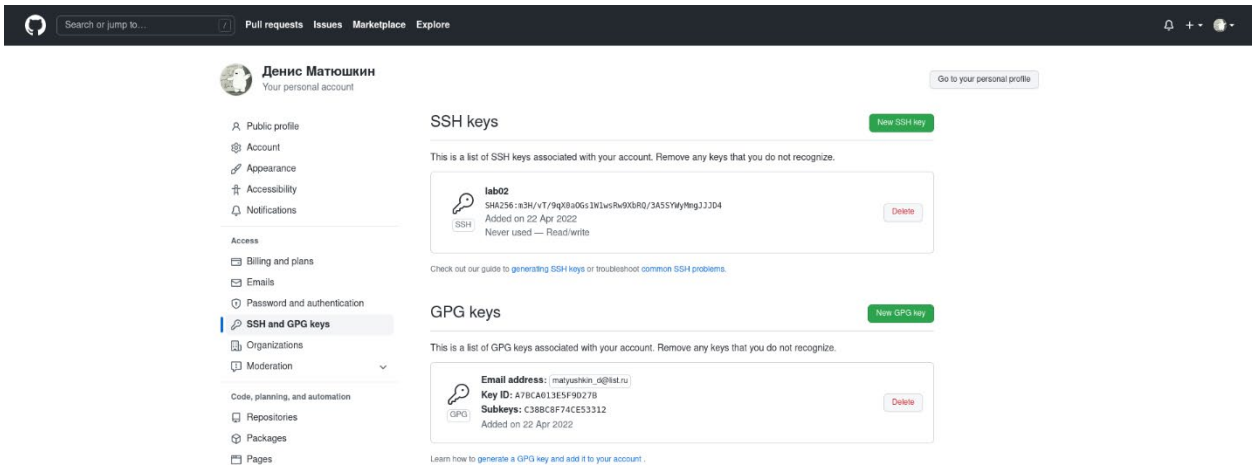

Рис. 8.2: Заполнение ключей SSH и GPG в личном аккаунте

10. Настроим автоматические подписи коммитов Git. Используя введенный email, укажем Git применять его при подписи коммитов (рис. 10).

```
[dvmatyushkin@dvmatyushkin tmp]$ git config --global user.signingkey A7BCA013E5F9D27B
[dvmatyushkin@dvmatyushkin tmp]$ git config --global commit.gpgsign true
[dvmatyushkin@dvmatyushkin tmp]$ git config --global gpg.program $(which gpg2)
[dvmatyushkin@dvmatyushkin tmp]$
```

Рис. 10: Настроение автоматических подписей коммитов Git

11. Настроим gh. Для начала необходимо авторизоваться, можно через браузер (рис. 11).

```
[dvmatyushkin@dvmatyushkin ~]$ 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: 25D0-242C
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 stifell
```

Рис. 11: Настройка gh

12. Создадим репозитория курса на основе шаблона. Для этого создадим каталоги work/study/2021-2022/”Операционные системы” (рис. 12)

```
[dvmatyushkin@dvmatyushkin ~]$ mkdir -p ~/work/study/2021-2022/"Операционные системы"
[dvmatyushkin@dvmatyushkin ~]$ cd ~/work/study/2021-2022/"Операционные системы"
[dvmatyushkin@dvmatyushkin Операционные системы]$ gh repo create study_2021-2022_os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository stifell/study_2021-2022_os-intro on GitHub
[dvmatyushkin@dvmatyushkin Операционные системы]$ git clone --recursive git@github.com:stifell/study_2021-2022_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: 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), 12.52 КиБ | 61.00 КиБ/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/dvmatyushkin/work/study/2021-2022/Операционные системы/os-intro/template/presentation»…
remote: Enumerating objects: 42, done.
remote: Counting objects: 100% (42/42), done.
remote: Compressing objects: 100% (34/34), done.
Получение объектов: 100% (42/42), 31.19 КиБ | 76.00 КиБ/с, готово.
Определение изменений: 100% (9/9), готово.
remote: Total 42 (delta 9), reused 40 (delta 7), pack-reused 0
Клонирование в «/home/dvmatyushkin/work/study/2021-2022/Операционные системы/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 | 406.00 КиБ/с, готово.
Определение изменений: 100% (31/31), готово.
Подмодуль по пути «template/presentation»: забрано состояние «3eaebb7586f8a9aded2b506cd1018e625b228b93»
Подмодуль по пути «template/report»: забрано состояние «df7b2ef80f8def3b9a496f8695277469a1a7842a»
[dvmatyushkin@dvmatyushkin Операционные системы]$
```

Рис. 12: Создание репозитория курса

13. Настроим каталог курса (рис. 13.1, 13.2 и 13.3). Для этого:

1) Перейдем в каталог курса.

2) Удалим лишние файлы.

3) Создадим необходимые каталоги.

4) Отправим файлы на сервер.

```
[dvmatyushkin@dvmatyushkin Операционные системы]$ cd ~/work/study/2021-2022/"Операционные системы"/os-intro
[dvmatyushkin@dvmatyushkin os-intro]$ rm package.json
[dvmatyushkin@dvmatyushkin os-intro]$ make COURSE=os-intro
```

Рис. 13.1: Переход в каталог, удаление лишних файлов и создание необходимых каталогов

```
[dvmatyushkin@dvmatyushkin os-intro]$ git add .
[dvmatyushkin@dvmatyushkin os-intro]$ git commit -am 'feat(main): make course structure'
[master 05a84c8] 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
```

Рис. 13.2: Отправка файлов на сервер

```
[dvmatyushkin@dvmatyushkin os-intro]$ git push
Перечисление объектов: 20, готово.
Подсчет объектов: 100% (20/20), готово.
Сжатие объектов: 100% (16/16), готово.
Запись объектов: 100% (19/19), 266.52 КиБ | 1.56 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:stifell/study_2021-2022_os-intro.git
   7b2a67e..05a84c8  master -> master
```

Рис. 13.3: Продолжение вывода

Заключение:

В ходе этой лабораторной работы мы изучили идеологию и применение средств контроля версий. Освоили умения по работе с git.

Контрольные вопросы:

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

Системы контроля версий (Version Control System, VCS) – система, использующаяся для хранения и управления своим программным продуктом более удобным способом (разработкой). 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-система, которая обеспечивает хранение всех существовавших вариантов файлов Commit-фиксация изменений. История-список предыдущих ревизий. Рабочая копия-копия другой ветки. Команде commit можно передать сообщение, описывающее изменения в ревизии. Она также записывает идентификатор пользователя, текущее время и временную зону, плюс список измененных файлов и их содержимого. Сообщение, описывающее изменения, определяется через опцию -m, или – message. Можно также вводить сообщения, состоящие из нескольких строк; в большинстве оболочек вы можете сделать это оставив открытую кавычку в конце строки. commit -m "добавлен первый файл.

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

Системы контроля версий. Централизованная система контроля версий Subversion и децентрализованная система контроля версий Mercurial. Существуют СКВ централизованные, в которых имеется один репозиторий, в который собираются изменения со всех рабочих копий разработчиков, и децентрализованные,

когда репозиториев много, и они могут обмениваться изменениями между собой. централизованные СКВ - Репозиторий один. У каждого разработчика своя рабочая копия. Время от времени разработчик может затягивать к себе в рабочую копию новые изменения из репозитория, или проталкивать свои изменения из своей рабочей копии в репозиторий. Прочие особенности централизованных СКВ зависят от реализации.

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

Создание хранилища (git init). Настройка хранилища. Коммиты нужных файлов (git add , git commit). Пуш файлов (git push). В случае неработоспособности отменить вернуть рабочее состояние исходя из хранимых в хранилище прошлых версий.

5. Опишите порядок работы с общим хранилищем VCS.

Создание хранилища (git init). Проведение работы локально. Добавление в коммит различных файлов (git add , git commit), коммит ресурсов. Пуш файлов (git push). В случае неработоспособности можно вернуть всё обратно. Если кто-то другой изменил репозиторий - пул ветки.

6. Каковы основные задачи, решаемые инструментальным средством git?

Предоставление удаленного доступа к коду, хранение всех версий продукта, предоставление возможности отката в случае неработоспособности кода.

7. Назовите и дайте краткую характеристику командам 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 изменения: 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.

Локальные репозитории могут быть полезны когда разработчик один. Удаленные - когда работает целая команда. Так же это может быть и для одного человека - он может иметь доступ к хранилищу удаленно.

Как и обычно мы должны создать репозиторий, загрузить его. Тут уже либо у себя, либо удаленно. После этого человек может работать у себя локально и загружать данные на удаленный репозитории. Используя при этом команды commit, add, push , pull и другие

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

Ветви - отдельные области основного хранилища. Ветви могут быть полезны во многих случаях. Начиная от того, что различные модули программы разрабатываются на одной ветке, заканчивая тем, что ветки могут быть для будущего "код ревью". Так же могут быть какие-то экспериментальные ветви. Применений множество. Но единый смысл - возможность распараллелить хранилище

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

Можно установить специальные настройки в .gitignore. Это может быть полезно, если при работе с файлами могут появляться временные файлы, которые загружать в хранилище нет необходимости.