РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

дисциплина: Операционные системы

Студент: Куликова Юлия Викторовна

Группа: НПМбв-01-18

МОСКВА

2022 г.

Цель работы

Изучить идеологию и применение средств контроля версий.

Ход работы

Настройка git

1) Создание учётной записи на Github

2) Настройка системы контроля версий git

```
yulia@yulia-VirtualBox:~$ git config --global user.name "Yulia Kulikova"
yulia@yulia-VirtualBox:~$ git config --global user.email "jili.lina.z@gmail.com"
```

Подключение репозитория к GitHub

1) генерируем ssh ключ

2)Копируем полученный ключ и вставляем в гитхаб

3) Создаём репозиторий на GitHub

4) инициализируем репозиторий

5) Подключаемся к репозиторию os-intro

6) Создаём заготовку для файла README.md и отправляем на Github

Первичная конфигурация

Добавим файл лицензии:

```
yulia@yulia-VirtualBox:~/laboratory$ wget https://creativecommons.org/licenses/b
y/4.0/legalcode.txt -O LICENSE
--2022-06-17 16:29:45--  https://creativecommons.org/licenses/by/4.0/legalcode.t
xt
Resolving creativecommons.org (creativecommons.org)... 172.67.34.140, 104.20.150
.16, 104.20.151.16, ...
Connecting to creativecommons.org (creativecommons.org)|172.67.34.140|:443... co
nnected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: ‘LICENSE’

LICENSE                 [ <=>                ]  18,22K  --.-KB/s    in 0,009s

2022-06-17 16:29:46 (2,08 MB/s) - ‘LICENSE’ saved [18657]
```

Добавим шаблон игнорируемых файлов. Просмотрим список имеющихся шаблонов:

```
yulia@yulia-VirtualBox:~/laboratory$ curl -L -s https://www.gitignore.io/api/lis
t
1c,1c-bitrix,a-frame,actionscript,ada
adobe,advancedinstaller,adventuregamestudio,agda,al
alteraquartusii,altium,amplify,android,androidstudio
angular,anjuta,ansible,ansibletower,apachecordova
```

Затем скачаем шаблон, например, для C:

```
yulia@yulia-VirtualBox:~/laboratory$ curl -L -s https://www.gitignore.io/api/c >
> .gitignore
yulia@yulia-VirtualBox:~/laboratory$
```

Добавим новые файлы:

```
yulia@yulia-VirtualBox:~/laboratory$ git add .
yulia@yulia-VirtualBox:~/laboratory$ git commit -a
Aborting commit due to empty commit message.
yulia@yulia-VirtualBox:~/laboratory$ git commit -m "gitignore"
[main 5341744] gitignore
 2 files changed, 455 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 LICENSE
yulia@yulia-VirtualBox:~/laboratory$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 6.43 KiB | 6.43 MiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:UlyaKulikova/os-intro.git
   8ac4fd1..5341744  main -> main
yulia@yulia-VirtualBox:~/laboratory$
```

Конфигурация git-flow

Инициализируем git-flow

```
yulia@yulia-VirtualBox:~/laboratory$ git flow init

Which branch should be used for bringing forth production releases?
   - main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [/home/yulia/laboratory/.git/hooks]
```

Проверьте, что Вы на ветке develop:

```
yulia@yulia-VirtualBox:~/laboratory$ git branch
* develop
  main
```

Создадим релиз с версией 1.0.0

```
yulia@yulia-VirtualBox:~/laboratory$ git flow release start 1.0.0
Switched to a new branch 'release/1.0.0'

Summary of actions:
- A new branch 'release/1.0.0' was created, based on 'develop'
- You are now on branch 'release/1.0.0'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

     git flow release finish '1.0.0'

yulia@yulia-VirtualBox:~/laboratory$
```

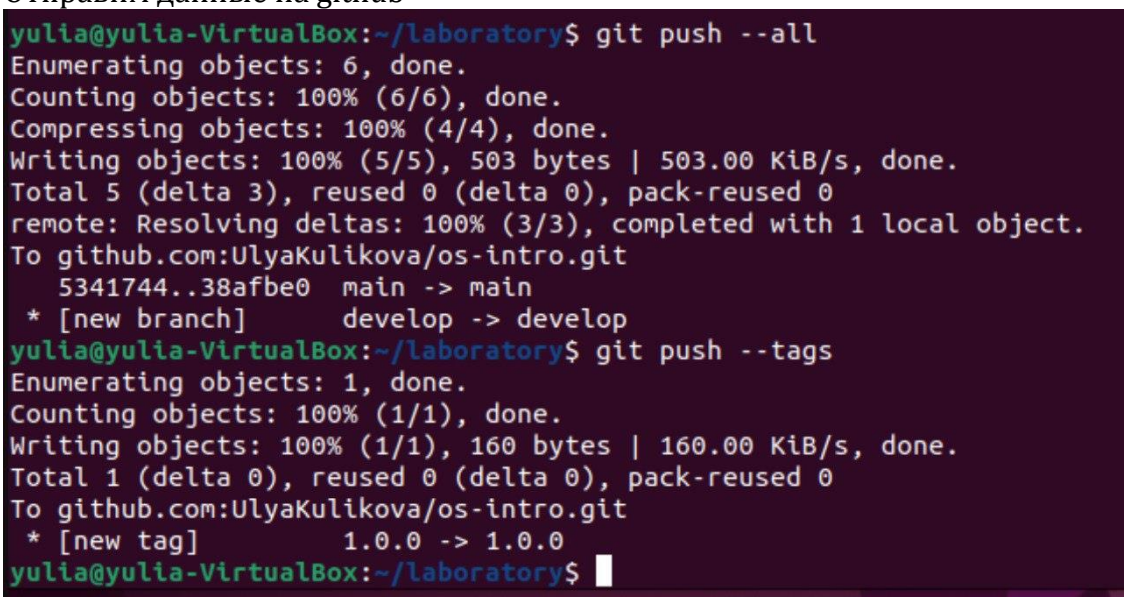
Запишем версию:

```
yulia@yulia-VirtualBox:~/laboratory$ echo "1.0.0" >> VERSION
yulia@yulia-VirtualBox:~/laboratory$
```

Добавим в индекс:

```
yulia@yulia-VirtualBox:~/laboratory$ echo "1.0.0" >> VERSION
yulia@yulia-VirtualBox:~/laboratory$ git add .
yulia@yulia-VirtualBox:~/laboratory$ git commit -am 'chore(main): add version'
[release/1.0.0 5798d6c] chore(main): add version
 1 file changed, 1 insertion(+)
 create mode 100644 VERSION
yulia@yulia-VirtualBox:~/laboratory$ git flow release finish 1.0.0
```

Отправим данные на github



Вывод

В ходе выполнения лабораторной работы №2 были освоены навыки администрирования и взаимодействия с децентрализованной системой и операционной системой git для параллельного контроля поддержки программного кода.

Контрольные вопросы

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

Системы контроля версий (VCS) – это программное обеспечение для облегчения работы с изменяющейся информацией. Предназначаются для работы нескольких человек над одним проектом, совместная работа путем изменения файлов в одном репозитории.

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

- Хранилище – это общее пространство для хранения файлов
- Commit – это команда для записи индексированных изменений в репозитории
- В истории сохраняются все коммиты, по которым можно отследить автора, сообщение, дату и хэш коммита
- Все файлы кроме .git/ называются рабочей копией, и принадлежат пользователю

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

Централизованные системы контроля версий сохраняют проект и его файлы на один общий сервер, а децентрализованные системы контроля версий при каждом копировании данных удаленного репозитория, происходит полное копирование данных в локальный репозиторий. К примеру, централизованные системы контроля версий – SVN, MS TFS, ClearCase; децентрализованные системы контроля версий – Git, Mercurial, Bazaar.

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

1) Создаем репозиторий и именуем его
2) Добавляем файлы в репозиторий
3) Фиксируем с помощью коммитов
4) Изменяем файлы репозитория и фиксируем изменения

5. Опишите порядок работы с общим хранилищем VCS.

1) Создаем репозиторий, именуем его или присоединяемся к нему в качестве contributor
2) Добавляем файлы в репозиторий
3) Фиксируем с помощью коммитов
4) Изменяем файлы репозитория и фиксируем изменения
5) Ждем проверки коммитов при участии других пользователей в общем репозитории

6. Каковы основные задачи, решаемые инструментальным средством git?

Систематизация, параллельность разработки программного обеспечения, единое месть для хранения файлов проекта

7. Назовите и дайте краткую характеристику командам git.

- Git init – создание репозитория,
- Git clone – клонирование репозитория,
- Git add – добавление изменений в индекс,
- Git reset – удаление изменений из индекса,
- Git commit – коммиты,
- Git rm – удаление файла.

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

К примеру, я использую локальные репозитории для черновых работ по лабораторным работам, а удаленный репозиторий git для их распространения и оценивания преподавателем.

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

Ветви служат для параллельной разработки программного обеспечения, тестирования, отладки и улучшения

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

Игнорирование можно установить для проекта, компьютера и репозитория, цель игнорирования заключается в том, чтобы не отслеживать файлы служебного типа, например временные файлы сборных утилит для проектов или только те файлы, которые полезны при взаимодействии только с очень ограниченным программным обеспечением