

Отчёт о выполнении лабораторной работы №2

Управление версиями

Российский Университет Дружбы Народов

Факультет Физико-Математических и Естественных Наук

Дисциплина: *Операционные системы*

Работу выполняла: Живцова Анна

1032201673

НКН60-01-20

Москва. Дисплейный класс РУДН. 2021г.

Цель работы

Изучить идеологию и применение средств контроля версий.

Задание

1. Создайте учётную запись на <https://github.com>.
2. Настройте систему контроля версий git, как это описано выше с использованием сервера репозитория <https://github.com/>.
3. Создайте структуру каталога лабораторных работ согласно пункту М.2.
4. Подключение репозитория к github
 - Создайте репозиторий на GitHub. Для примера назовём его os-intro.
 - Рабочий каталог будем обозначать как laboratory. Вначале нужно перейти в этот каталог:
 - Инициализируем систему git:
 - Создаём заготовку для файла README.md:

```
echo "# Лабораторные работы" >> README.md
```
 - ```
git add README.md
```

– Делаем первый коммит и выкладываем на github:

```
git commit -m "first commit"
```

```
git remote add origin
```

```
↪ git@github.com:sciproc-intro.git
```

5. Первичная конфигурация

– Добавим файл лицензии: 

```
wget https://creativecommons.org/licenses/by/4.0/legalcode.txt
```

– Добавим шаблон игнорируемых файлов. Просмотрим список имеющихся шаблонов:

```
curl -L -s https://www.gitignore.io/api/list
```

Затем скачаем шаблон, например, для C:

```
curl -L -s https://www.gitignore.io/api/c >> .gitignore
```

Можно это же сделать через web-интерфейс на сайте <https://www.gitignore.io/>.

io/. – Добавим новые файлы:

```
git add .
```

– Выполним коммит:

```
git commit -a
```

– Отправим на github:

```
git push
```

## 6. Конфигурация git-flow

– Инициализируем git-flow

```
git flow init
```

Префикс для ярлыков установим в v.

– Проверьте, что Вы на ветке develop:

```
git branch
```

– Создадим релиз с версией 1.0.0

```
git flow release start 1.0.0
```

– Запишем версию:

```
echo "1.0.0" >> VERSION
```

– Добавим в индекс:

```
git add .
```

```
git commit -am 'chore(main): add version'
```

– Зальём релизную ветку в основную ветку

```
git flow release finish 1.0.0
```

– Отправим данные на github

```
git push --all
```

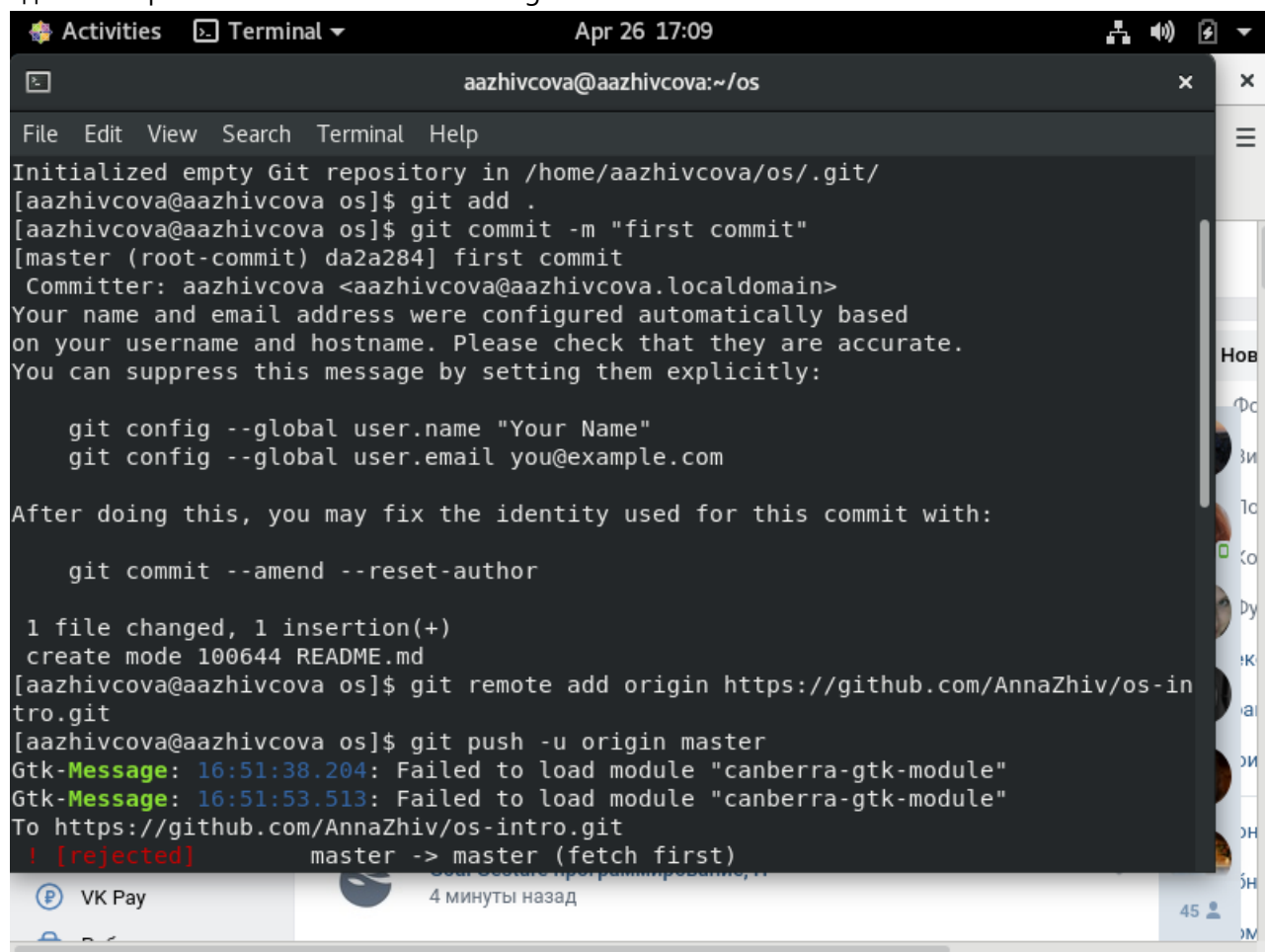
```
git push --tags
```

– Создадим релиз на github.

## Выполнение работы

1. Создала учётную запись на <https://github.com>.
2. Настройте систему контроля версий git добавила ssh ключ
  - Создала репозиторий на GitHub и файл README.md.

сделала первый коммит и выложила на github:



The screenshot shows a terminal window titled "aazhivcova@aazhivcova:~/os". The terminal output is as follows:

```
File Edit View Search Terminal Help
Initialized empty Git repository in /home/aazhivcova/os/.git/
[aazhivcova@aazhivcova os]$ git add .
[aazhivcova@aazhivcova os]$ git commit -m "first commit"
[master (root-commit) da2a284] first commit
Committer: aazhivcova <aazhivcova@aazhivcova.localdomain>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

 git config --global user.name "Your Name"
 git config --global user.email you@example.com

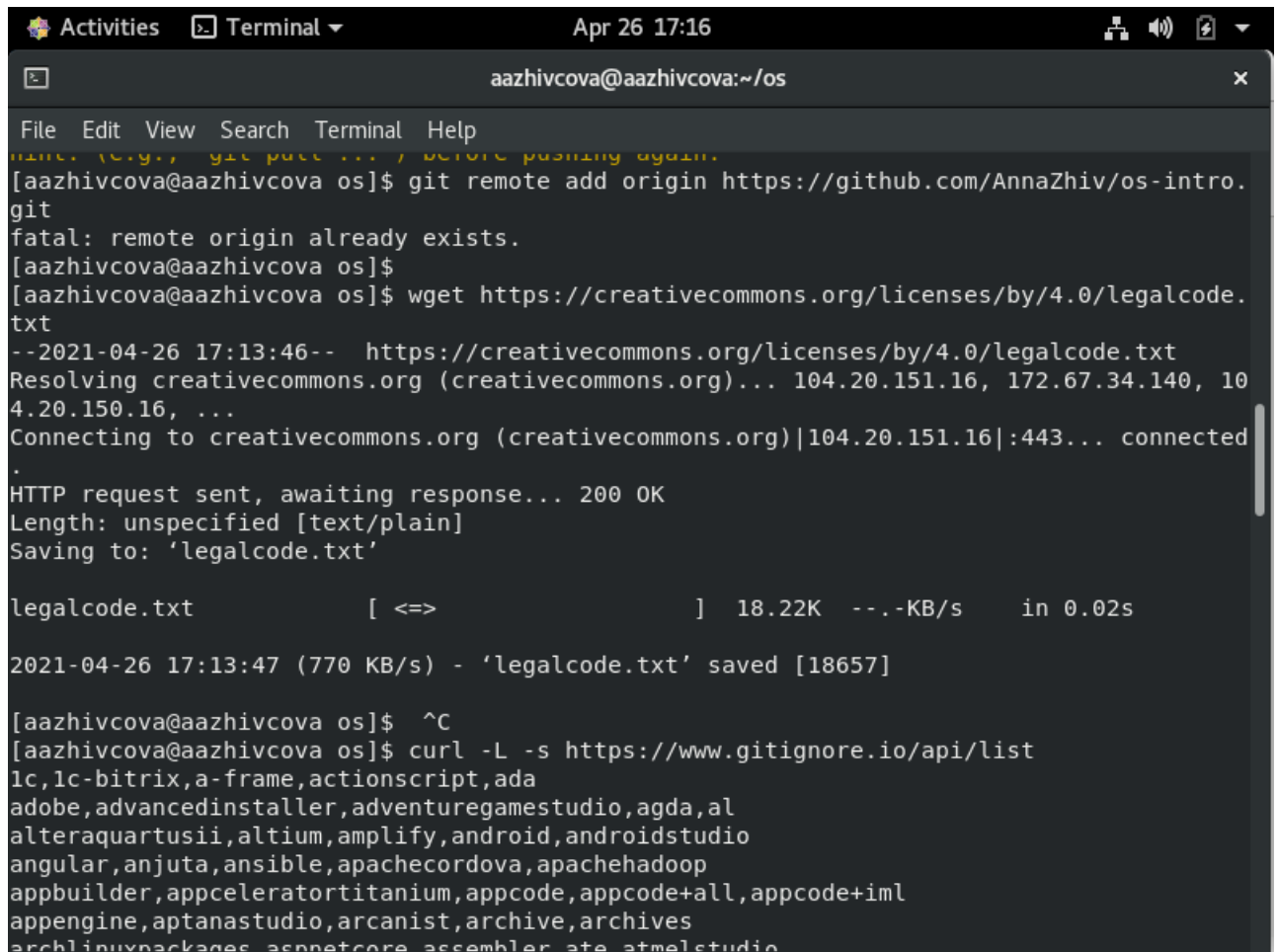
After doing this, you may fix the identity used for this commit with:

 git commit --amend --reset-author

1 file changed, 1 insertion(+)
create mode 100644 README.md
[aazhivcova@aazhivcova os]$ git remote add origin https://github.com/AnnaZhiv/os-intro.git
[aazhivcova@aazhivcova os]$ git push -u origin master
Gtk-Message: 16:51:38.204: Failed to load module "canberra-gtk-module"
Gtk-Message: 16:51:53.513: Failed to load module "canberra-gtk-module"
To https://github.com/AnnaZhiv/os-intro.git
! [rejected] master -> master (fetch first)
```

### 3. Первичная конфигурация

- Добавила файл лицензии:
- Добавила шаблон игнорируемых файлов. Просмотрела список имеющихся шаблонов:



```
Activities Terminal Apr 26 17:16
aazhivcova@aazhivcova:~/os
File Edit View Search Terminal Help
fatal: remote origin already exists.
[aazhivcova@aazhivcova os]$ git remote add origin https://github.com/AnnaZhiv/os-intro.
git
fatal: remote origin already exists.
[aazhivcova@aazhivcova os]$
[aazhivcova@aazhivcova os]$ wget https://creativecommons.org/licenses/by/4.0/legalcode.
txt
--2021-04-26 17:13:46-- https://creativecommons.org/licenses/by/4.0/legalcode.txt
Resolving creativecommons.org (creativecommons.org)... 104.20.151.16, 172.67.34.140, 10
4.20.150.16, ...
Connecting to creativecommons.org (creativecommons.org)|104.20.151.16|:443... connected
.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: 'legalcode.txt'

legalcode.txt [<=>] 18.22K --.-KB/s in 0.02s

2021-04-26 17:13:47 (770 KB/s) - 'legalcode.txt' saved [18657]

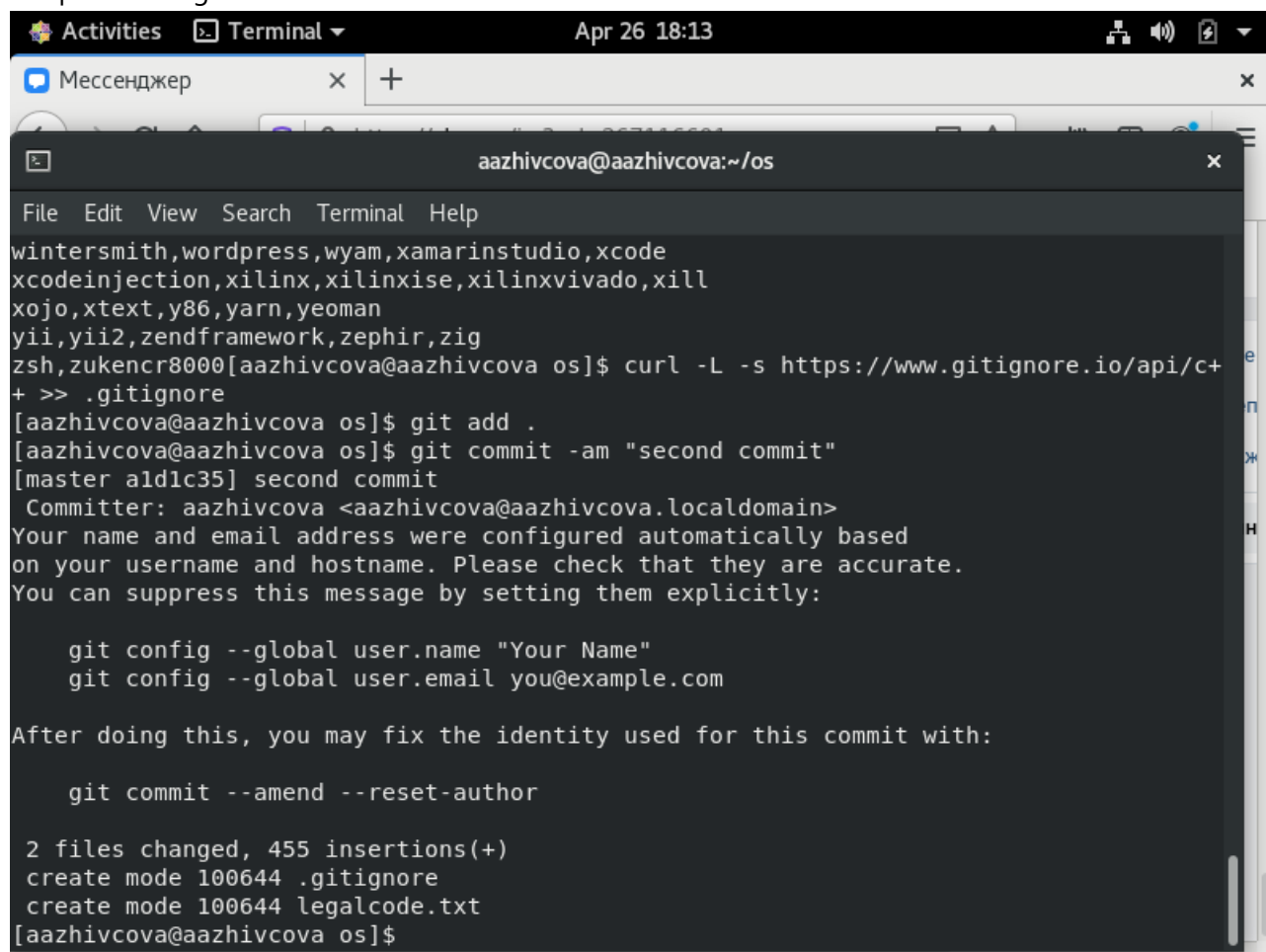
[aazhivcova@aazhivcova os]$ ^C
[aazhivcova@aazhivcova os]$ curl -L -s https://www.gitignore.io/api/list
1c,1c-bitrix,a-frame,actionscript,ada
adobe,advancedinstaller,adventuregamestudio,agda,al
alteraquartusii,altium,amplify,android,androidstudio
angular,anjuta,ansible,apachecordova,apachehadoop
appbuilder,appcelerator titanium,appcode,appcode+all,appcode+iml
appengine,aptanastudio,arcanist,archive,archives
archlinuxpackages,arduino,armnetcore,assembler,ata,atmelstudio
```

Затем скачала шаблон, например, для C++:

Добавила новые файлы:

Выполнила коммит:

Отправила на github:



The screenshot shows a terminal window titled 'aazhivcova@aazhivcova:~/os'. The terminal output shows the following commands and their results:

```
File Edit View Search Terminal Help
wintersmith,wordpress,wyam,xamarinstudio,xcode
xcodeinjection,xilinx,xilinxise,xilinxvivado,xill
xojo,xtext,y86,yarn,yeoman
yii,yii2,zendframework,zephir,zig
zsh,zukencr8000[aazhivcova@aazhivcova os]$ curl -L -s https://www.gitignore.io/api/c+
+ >> .gitignore
[aazhivcova@aazhivcova os]$ git add .
[aazhivcova@aazhivcova os]$ git commit -am "second commit"
[master ald1c35] second commit
Committer: aazhivcova <aazhivcova@aazhivcova.localdomain>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

 git config --global user.name "Your Name"
 git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

 git commit --amend --reset-author

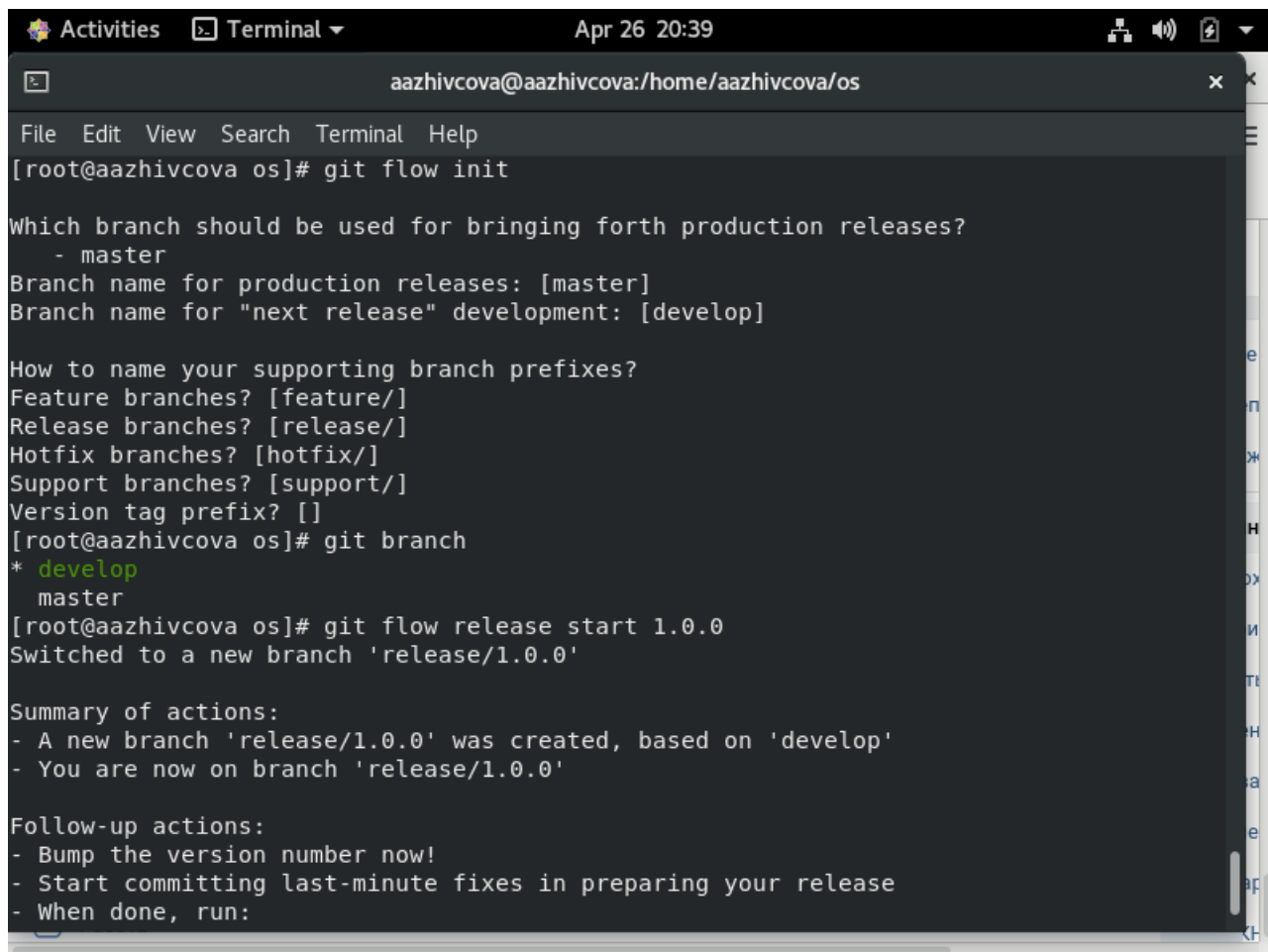
2 files changed, 455 insertions(+)
create mode 100644 .gitignore
create mode 100644 legalcode.txt
[aazhivcova@aazhivcova os]$
```

#### 4. Конфигурация git-flow

Инициализировала git-flow

Проверила, что я на ветке develop:

Создала релиз с версией 1.0.0



```
Activities Terminal Apr 26 20:39
aazhivcova@aazhivcova:/home/aazhivcova/os
File Edit View Search Terminal Help
[root@aazhivcova os]# git flow init

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

How to name your supporting branch prefixes?
Feature branches? [feature/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
[root@aazhivcova os]# git branch
* develop
 master
[root@aazhivcova os]# git flow release start 1.0.0
Switched to a new branch 'release/1.0.0'

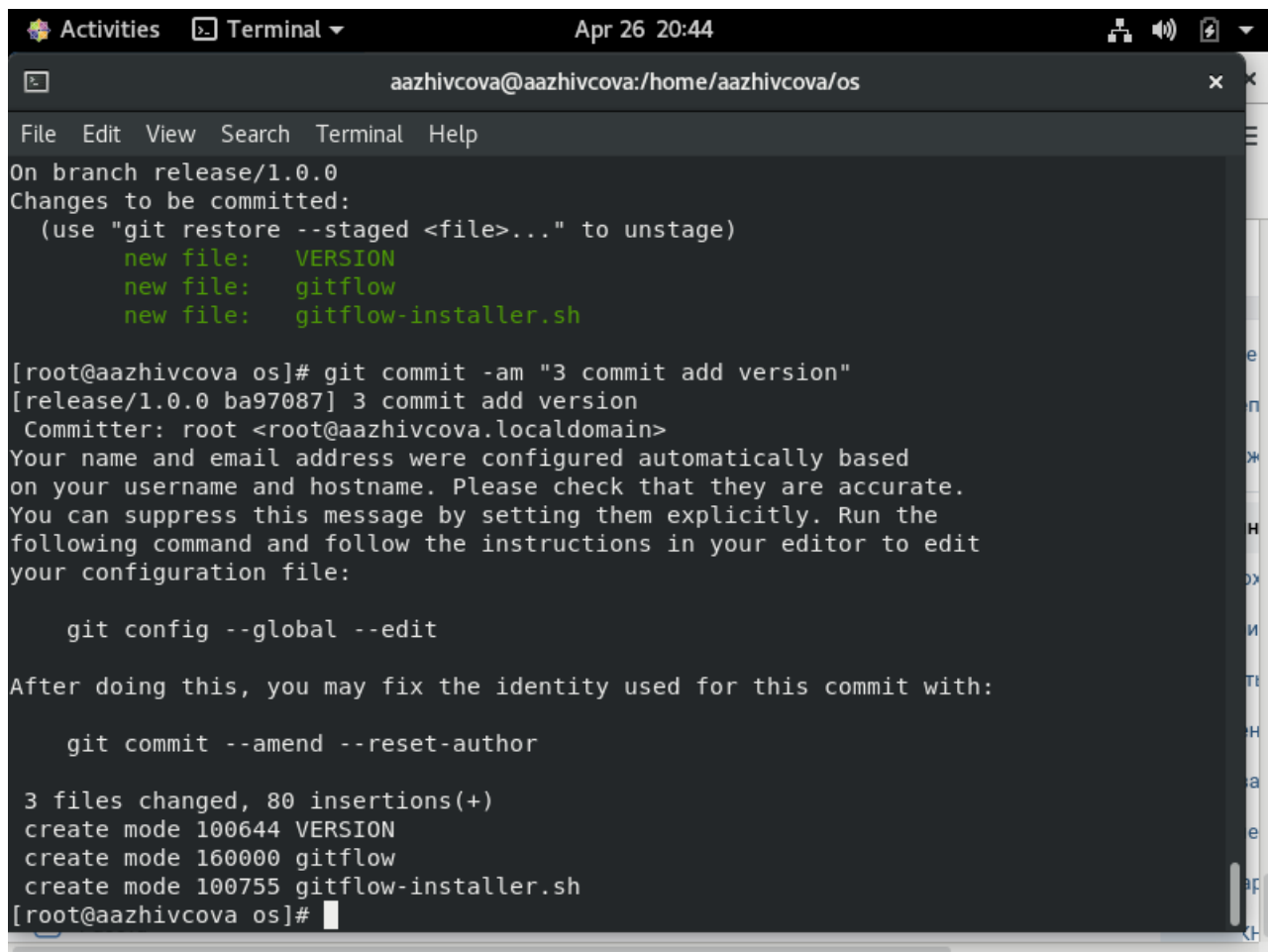
Summary of actions:
- A new branch 'release/1.0.0' was created, based on 'develop'
- You are now on branch 'release/1.0.0'

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

Записала версию:

Добавила в индекс:

Залила релизную ветку в основную ветку



```

Activities Terminal Apr 26 20:44
aazhivcova@aazhivcova:/home/aazhivcova/os

File Edit View Search Terminal Help
On branch release/1.0.0
Changes to be committed:
 (use "git restore --staged <file>..." to unstage)
 new file: VERSION
 new file: gitflow
 new file: gitflow-installer.sh

[root@aazhivcova os]# git commit -am "3 commit add version"
[release/1.0.0 ba97087] 3 commit add version
Committer: root <root@aazhivcova.localdomain>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

 git config --global --edit

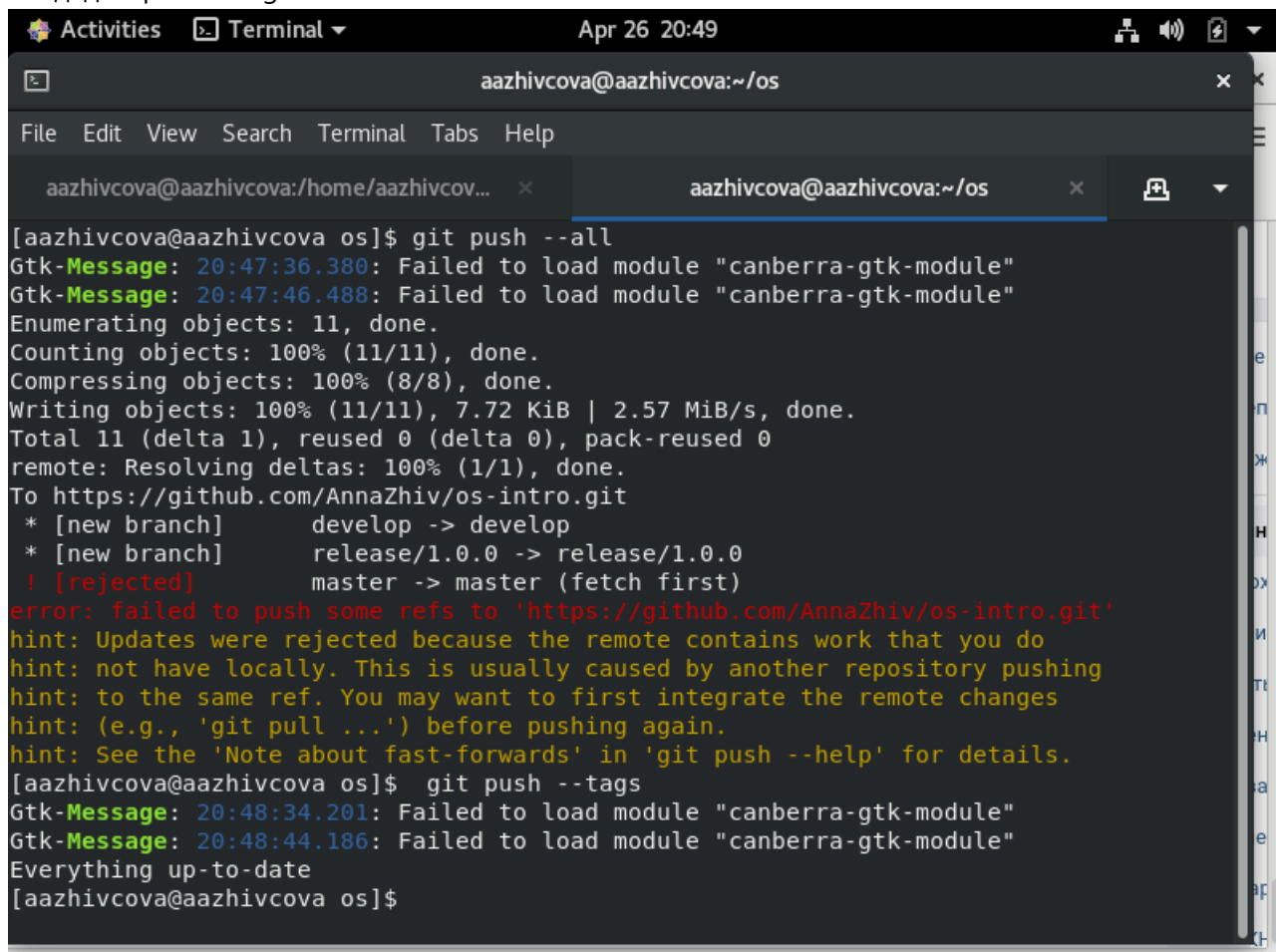
After doing this, you may fix the identity used for this commit with:

 git commit --amend --reset-author

3 files changed, 80 insertions(+)
create mode 100644 VERSION
create mode 160000 gitflow
create mode 100755 gitflow-installer.sh
[root@aazhivcova os]#

```

Отправила данные на github  
Создадим релиз на github.



```

Activities Terminal Apr 26 20:49
aazhivcova@aazhivcova:~/os

File Edit View Search Terminal Tabs Help

aazhivcova@aazhivcova:/home/aazhivcov... x aazhivcova@aazhivcova:~/os x

[aazhivcova@aazhivcova os]$ git push --all
Gtk-Message: 20:47:36.380: Failed to load module "canberra-gtk-module"
Gtk-Message: 20:47:46.488: Failed to load module "canberra-gtk-module"
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Compressing objects: 100% (8/8), done.
Writing objects: 100% (11/11), 7.72 KiB | 2.57 MiB/s, done.
Total 11 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/AnnaZhiv/os-intro.git
 * [new branch] develop -> develop
 * [new branch] release/1.0.0 -> release/1.0.0
 ! [rejected] master -> master (fetch first)
error: failed to push some refs to 'https://github.com/AnnaZhiv/os-intro.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
[aazhivcova@aazhivcova os]$ git push --tags
Gtk-Message: 20:48:34.201: Failed to load module "canberra-gtk-module"
Gtk-Message: 20:48:44.186: Failed to load module "canberra-gtk-module"
Everything up-to-date
[aazhivcova@aazhivcova os]$

```

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. 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.
2. В Git один коммит (англ. commit) представляет из себя ссылку на объект tree, соответствующий корневой директории, и ссылку на родительский коммит (кроме самого первого коммита в репозитории). Также в коммите есть информация об авторе и UNIX timestamp от времени создания. В Git единицей хранения данных является объект (англ. object), который однозначно определяется 40-символьным хешем sha1. В объектах Git хранит почти всё: коммиты, содержимое файлов, их иерархию. Сначала объекты представляют из себя обычные файлы в папке .git/objects, а после git gc упаковываются в .pack-файлы, о которых будет рассказано чуть ниже. Для экономии дискового пространства содержимое всех объектов дополнительно сжимается с помощью zlib. В Git нет отдельного хранилища истории. Всю историю можно развернуть, но лишь пройдя по ссылкам на родителя из нужного вам коммита. Если необходимо просмотреть историю только по одному файлу (или по поддиректории), Git всё равно должен проделать то же самое, но он будет возвращать отфильтрованные результаты. Стоит иметь это в виду, когда вы делаете интеграцию с Git, и не заставлять Git делать полный просмотр истории на каждый файл. Рабочая копия является снимком одной версии проекта. Эти файлы извлекаются из сжатой базы данных в каталоге Git и помещаются на диск, для того чтобы их можно было использовать или редактировать.
3. 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. 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.
5. 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. Задачи решаемые git: Как не потерять файлы с исходным кодом? Как защититься от случайных исправлений и удалений? Как отменить изменения, если они оказались некорректными? Как одновременно поддерживать рабочую версию и разработку новой?
7. – 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

#### 8. `git push --all`

9. Ветви функциональностей (feature branches), также называемые иногда тематическими ветвями (topic branches), используются для разработки новых функций, которые должны появиться в текущем или будущем релизах.

10. Игнорируемые файлы – это, как правило, специфичные для платформы файлы или автоматически созданные файлы из систем сборки. Некоторые общие примеры включают в себя: Файлы времени выполнения, такие как журнал, блокировка, кэш или временные файлы. Файлы с конфиденциальной информацией, такой как пароли или ключи API. Скомпилированный код, такой как .class или .o.

Каталоги зависимостей, такие как /vendor или /node\_modules.

Создавать папки, такие как /public, /out или /dist.

Системные файлы, такие как .DS\_Store или Thumbs.db

Конфигурационные файлы IDE или текстового редактора.

## Вывод

Изучила идеологию и применение средств контроля версий. Создала аккаунт и репозиторий на github. Выполнила важнейшие команды.