

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

## Лабораторная работа № 3

дисциплина: *Операционные системы*

Студент: Алших маслем Ахмад

Группа: НФИБД-02-20

### Цель работы:


Научиться оформлять отчёты с помощью легковесного языка разметки Markdown.

### Выполнение задания

Оформление лабораторной работы №2 на Markdown.

### Ход работы:

1. Создал учетную запись на github.



**AHMAD ALSHEIKH MUSLEM**

AHMADALSHIKHMUSLEM

Edit profile

Joined 11 days ago

Overview

Repositories2

Projects

Packages

Popular repositories

AHMADALSHIKHMUSLEM

Config files for my GitHub profile.

labaratornaya2

Shell

3 contributions in the last year

MayJunJulAugSepOctNovDecJanFebMarAprMay

Mon

Wed

Fri

Learn how we count contributions

Contribution settings

LessMore

Contribution activity

May 2021

Created their first repository

May 1

2. Обозначил рабочий каталог как test, создав ее командой mkdir. После перешел в данный каталог командой cd. Инициализировал систему git командой git init. Создаю заготовку для файла README.md: echo "# lab02" >> README.md git add README.md – Делаю первый коммит и выкладываем на github: git commit -m "first commit" git remote add origin [git@github.com:/sciproc-intro.git](https://github.com/sciproc-intro.git) git push -u origin master

```
ahmet@localhost:~/test1
File Edit View Search Terminal Help
[ahmet@localhost test1]$ echo "# labaratornaya2" >> README.md
[ahmet@localhost test1]$ git init
Initialized empty Git repository in /home/ahmet/test1/.git/
[ahmet@localhost test1]$ git add README.md
[ahmet@localhost test1]$ git commit -m "first commit"
[master (root-commit) ffb03b9] first commit
Committer: Ahmed <ahmet@localhost.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
[ahmet@localhost test1]$ git branch -M main
[ahmet@localhost test1]$ git remote add origin git@github.com:AHMADALSHIKHMUSLEM
/labaratornaya2.git
[ahmet@localhost test1]$ git push -u origin main
```

3. Добавил файл лицензий:

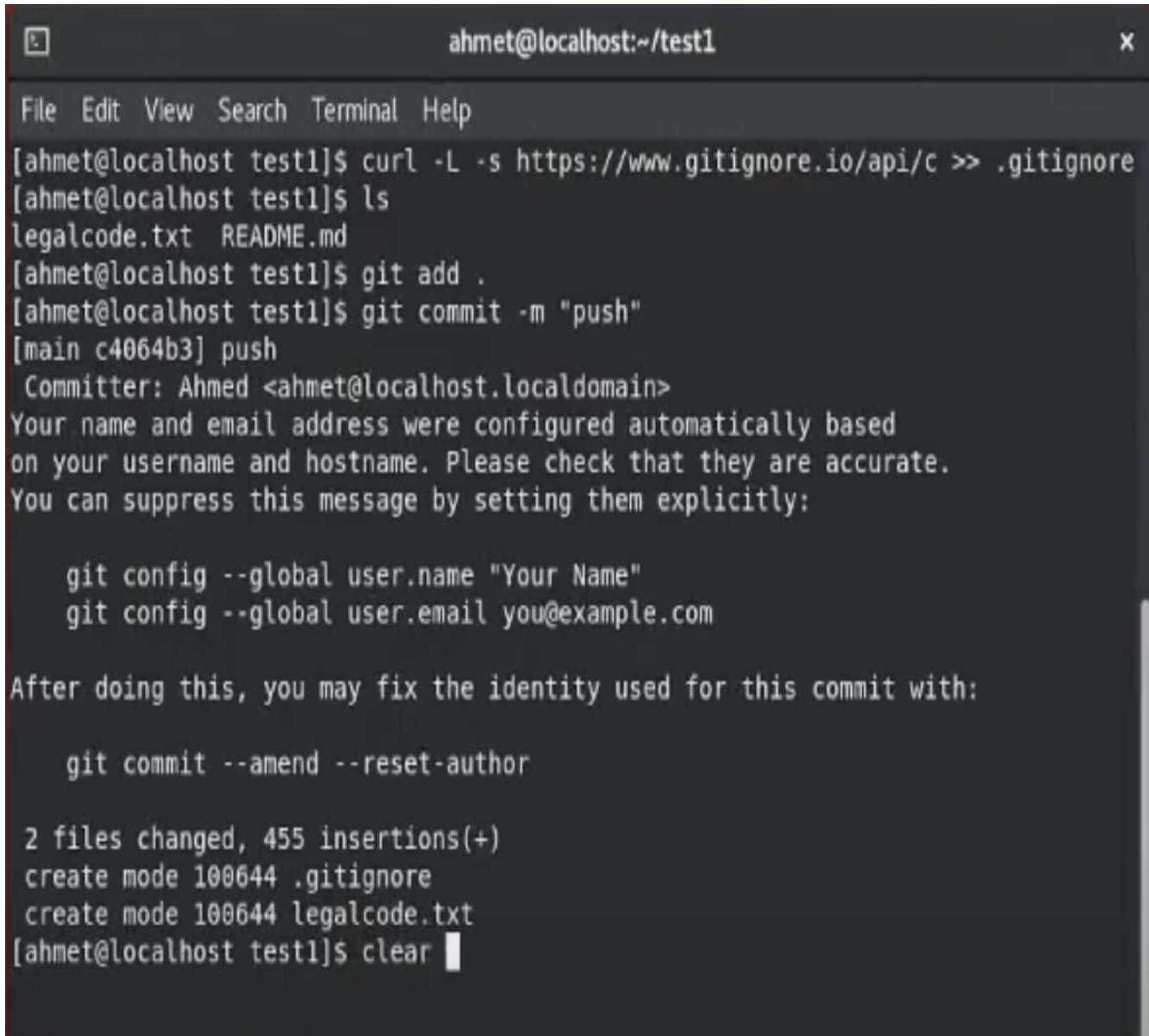
```
ahmet@localhost:~/test1
File Edit View Search Terminal Help
[ahmet@localhost test1]$ wget https://creativecommons.org/licenses/by/4.0/legalcode.txt
--2021-05-05 14:32:49-- https://creativecommons.org/licenses/by/4.0/legalcode.txt
Resolving creativecommons.org (creativecommons.org)... 104.20.150.16, 104.20.151.16, 172.67.34.140, ...
Connecting to creativecommons.org (creativecommons.org)|104.20.150.16|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: 'legalcode.txt'

legalcode.txt          [ <=> ] 18.22K --.-KB/s in 0s

2021-05-05 14:32:50 (62.6 MB/s) - 'legalcode.txt' saved [18657]

[ahmet@localhost test1]$ curl -L -s https://www.gitignore.io/api/list
```

4. Теперь скачиваем шаблон для C и Добавил новые файлы и выполнил коммит

A terminal window titled 'ahmet@localhost:~/test1' with a close button in the top right corner. The window contains a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal output shows the following sequence of commands and their results:

```
[ahmet@localhost test1]$ curl -L -s https://www.gitignore.io/api/c >> .gitignore
[ahmet@localhost test1]$ ls
legalcode.txt  README.md
[ahmet@localhost test1]$ git add .
[ahmet@localhost test1]$ git commit -m "push"
[main c4064b3] push
Committer: Ahmed <ahmet@localhost.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
[ahmet@localhost test1]$ clear
```

## 5. Отправим на github

```
ahmet@localhost:~/test1
File Edit View Search Terminal Help
[ahmet@localhost test1]$ 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 0), reused 0 (delta 0), pack-reused 0
To github.com:AHMADALSHIKHMUSLEM/labaratornaya2.git
   ffb03b9..c4064b3  main -> main
[ahmet@localhost test1]$
```

6. Инициализировал git-flow, установив префикс для ярлыков в v.

```
[ahmet@localhost test1]$ 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/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? [] v
```



## 7. Создаю релиз с версией 1.0.0

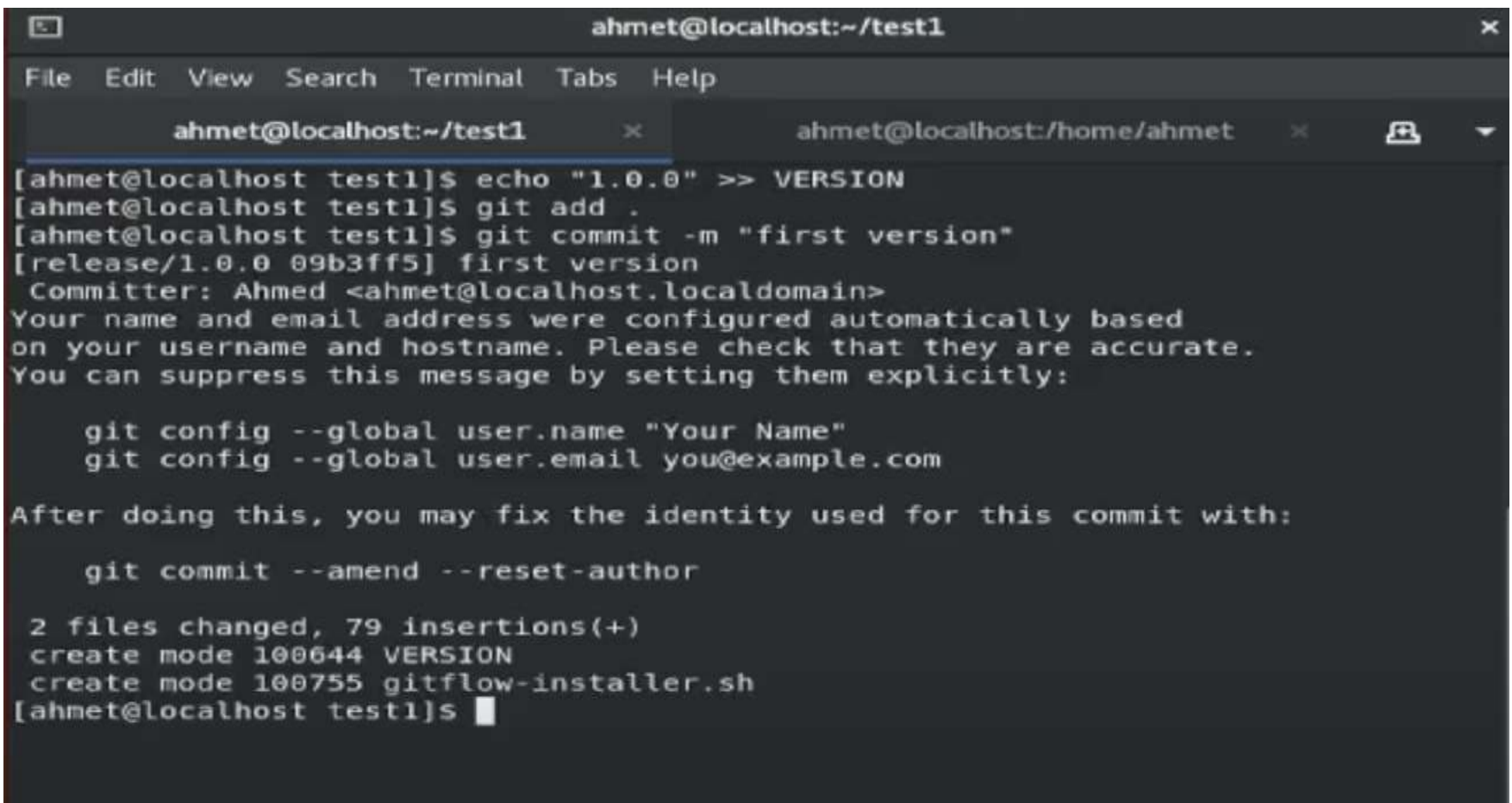
```
[ahmet@localhost test1]$ git flow release start 1.0.0
Switched to a new branch 'release/1.0.0'

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'
```

## 8. Записал версию и Добавил в индекс



```
ahmet@localhost:~/test1
File Edit View Search Terminal Tabs Help
ahmet@localhost:~/test1 x ahmet@localhost:/home/ahmet x
[ahmet@localhost test1]$ echo "1.0.0" >> VERSION
[ahmet@localhost test1]$ git add .
[ahmet@localhost test1]$ git commit -m "first version"
[release/1.0.0 09b3ff5] first version
Committer: Ahmed <ahmet@localhost.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, 79 insertions(+)
create mode 100644 VERSION
create mode 100755 gitflow-installer.sh
[ahmet@localhost test1]$
```

## 9. Залил релизную ветку в основную ветку

A screenshot of a terminal window titled "ahmet@localhost:~/test1". The terminal shows a successful git merge operation. The first line is the command "git merge release/1.0.0". The subsequent lines are the output from git, indicating that branch 'release/1.0.0' is being merged into 'main'. It prompts the user to enter a commit message, explaining that it's needed especially if merging an updated upstream into a topic branch. It also notes that lines starting with '#' will be ignored and an empty message aborts the commit. The prompt "~" appears at the end of the last visible line.

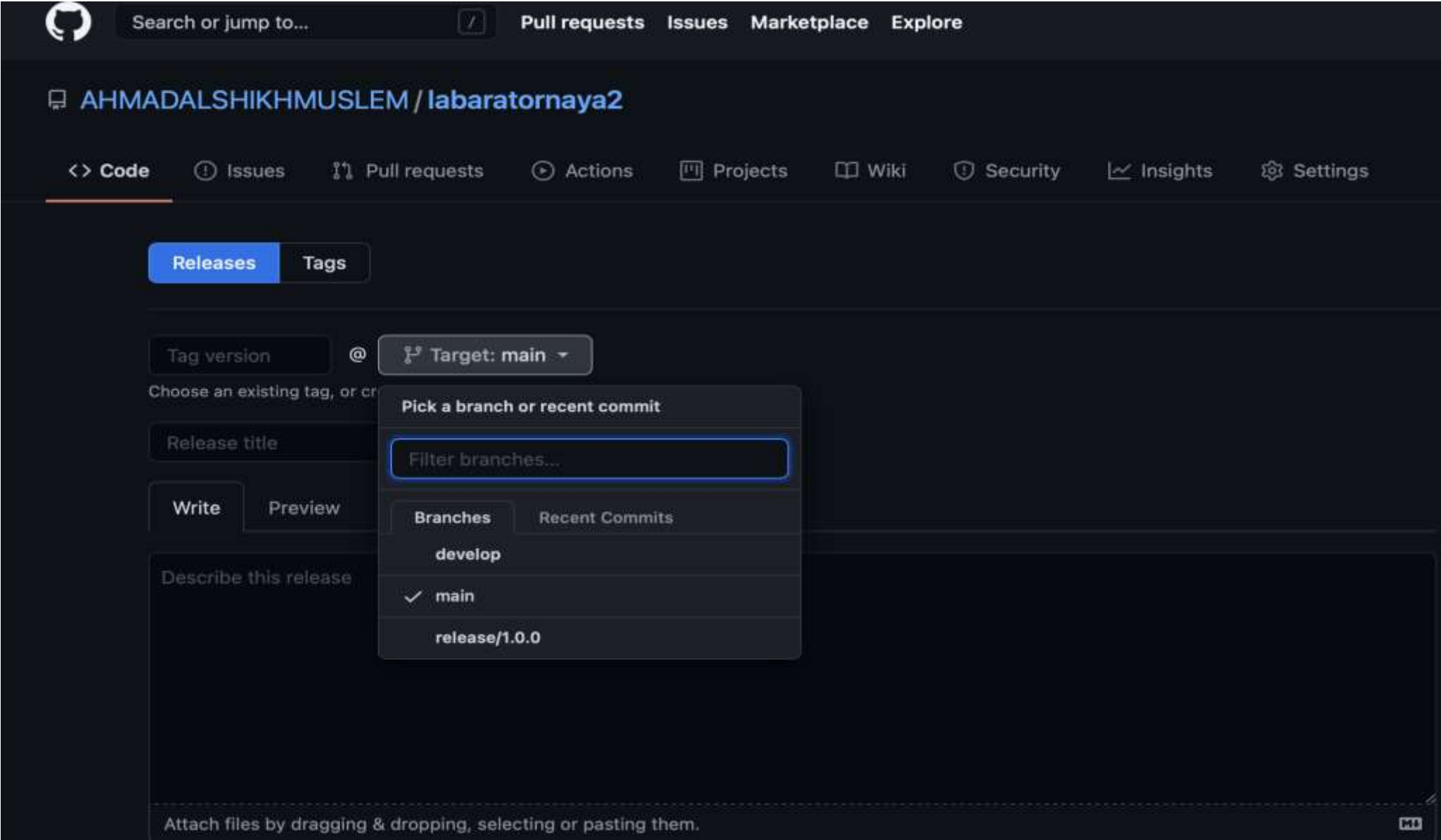
```
ahmet@localhost:~/test1
File Edit View Search Terminal Tabs Help

ahmet@localhost:~/test1 x ahmet@localhost:/home/ahmet x [icon] v

Merge branch 'release/1.0.0' into main
# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.
~
```

## 10. Отправил данные на github

```
[ahmet@localhost test1]$ git push --all
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 1.30 KiB | 1.30 MiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To github.com:AHMADALSHIKHMUSLEM/labaratornaya2.git
   c4064b3..b808c01  main -> main
   * [new branch]      develop -> develop
   * [new branch]      release/1.0.0 -> release/1.0.0
[ahmet@localhost test1]$ git push --tags
Everything up-to-date
[ahmet@localhost test1]$
```



**Вывод:**

Изучил и понял как работать с системой контроля версий, с помощью командной строки, а именно с Git. Разобралкоманды.

**Контрольные вопросы:**

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

Система контроля версий (Version Control System, VCS) представляет собой программное обеспечение для облегчения работы с изменяющейся информацией. VCS нужны для хранения полной истории изменений; Описания причин всех производимых изменений; Отката изменений, если что-то пошло не так; Поиска причины и ответственного за появления ошибок в программе; Совместной работы группы над одним проектом; Возможности изменять код, не мешая работе других пользователей.

2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Хранилище (repository), или репозиторий, — место хранения файлов и их версий, служебной информации. Версия (revision), или ревизия, — состояние всего хранилища или отдельных файлов в момент времени («пункт истории»). Commit («[трудовой] вклад», не переводится) — процесс создания новой версии; иногда синоним версии. Рабочая копия (working copy) — текущее состояние файлов проекта (любой версии), полученных из хранилища и, возможно, измененных.

3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере. Доступ к нему осуществлялся через специальное клиентское приложение. Пример: CVS-одна из первых систем второго поколения (1986г.). Обладает множеством недостатков и считается устаревшей.

Децентрализованные системы контроля версий, в отличие от централизованной модели, может существовать несколько экземпляров репозитория, которые время от времени синхронизируются между собой. Пример: Git- распределенная система управления версиями, созданная Л. Торвальдсом для управления разработкой ядра Linux.

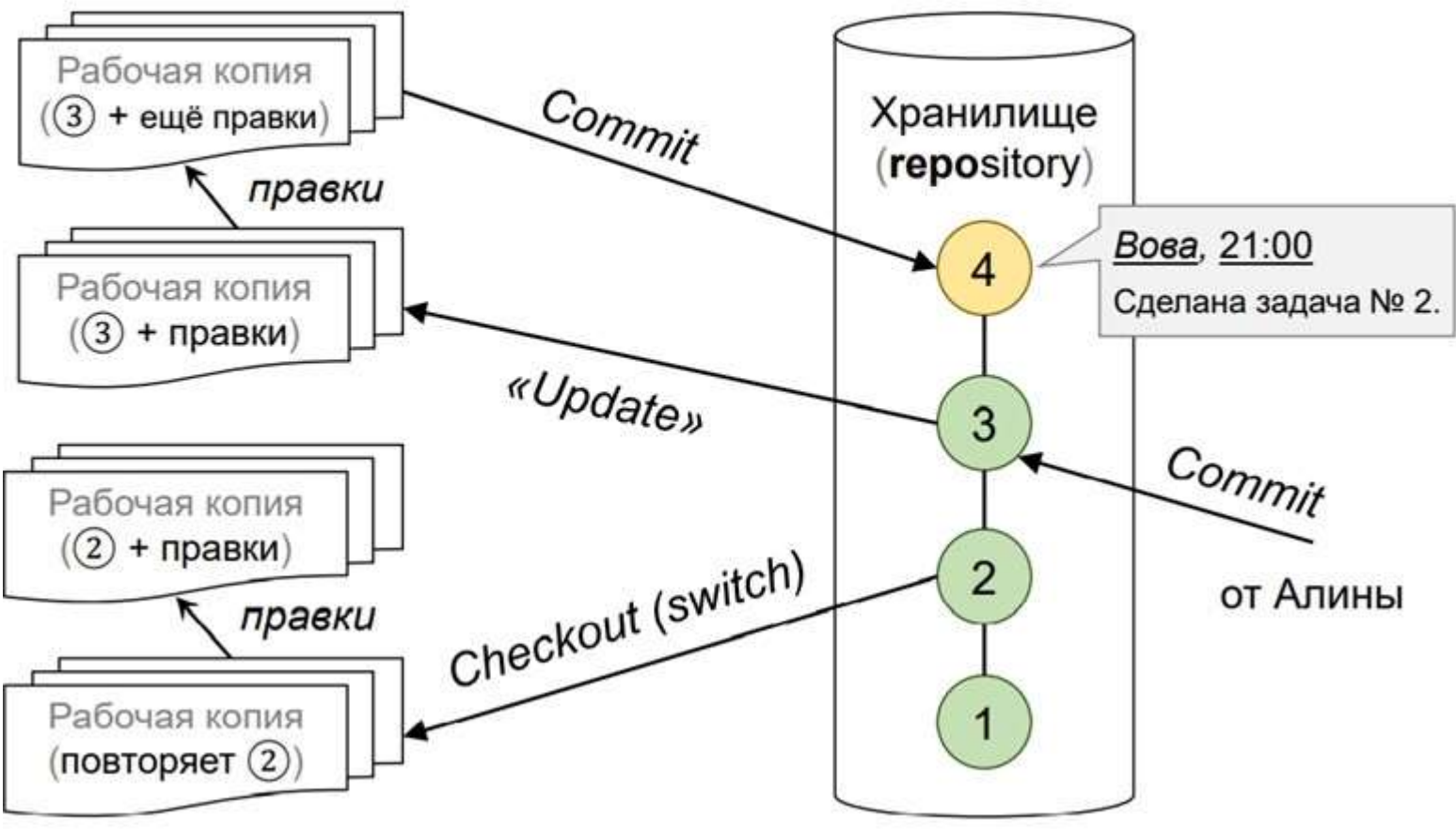
Отличия между централизованными и децентрализованными VCS. Централизованные:

- Простота использования.
  - Вся история — всегда в едином общем хранилище.
  - Нужно подключение к сети.
  - Резервное копирование нужно только одному хранилищу.
  - Удобство разделения прав доступа к хранилищу.
  - Почти все изменения навсегда попадают в общее хранилище. Децентрализованные:
  - Двухфазный commit:
    - 1. запись в локальную историю;
    - 2. пересылка изменений другим.
  - Подключение к сети не нужно.
  - Локальные хранилища могут служить резервными копиями.
  - Локальное хранилище контролирует его владелец,
  - но общее — администратор.
- Возможна правка локальной истории перед отправкой на сервер.

4. Опишите действия с VCS при единоличной работе с хранилищем. 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. Опишите порядок работы с общим хранилищем VCS. Работа с общим хранилищем выглядит так:

# Работа с общим хранилищем



6. Каковы основные задачи, решаемые инструментальным средством git? 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. Назовите и дайте краткую характеристику командам git.

- add - добавить файл или папку в репозиторий git;
- am - применить все патчи из email;
- archive - создать архив файлов;
- bisect - использовать бинарный поиск для поиска нужного коммита;
- branch - управление ветками проекта;
- bundle - перемещение объектов и ссылок в архиве;
- checkout - переключение между ветками;
- cherry-pick - внести изменения в уже существующие коммиты;
- clean - удалить все неотслеживаемые файлы и папки проекта;
- clone - создать копию удаленного репозитория в папку;
- commit - сохранить изменения в репозиторий;
- diff - посмотреть изменения между коммитами;
- fetch - скачать удаленный репозиторий;
- init - создать репозиторий;
- merge - объединить две ветви;
- pull - интегрировать удаленный репозиторий с локальным;
- push - отправить изменения в удаленный репозиторий;
- tag - управление тегами;
- worktree - управление деревьями разработки.

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

9. Что такое и зачем могут быть нужны ветви (branches)? Ветки нужны для того, чтобы программисты могли вести совместную работу над проектом и не мешать друг другу при этом. При создании проекта, Git создает базовую ветку. Она называется master веткой. Она считается центральной веткой, т.е. в ней содержится основной код приложения.

10. Как и зачем можно игнорировать некоторые файлы при commit? 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 к 3 лабораторной работе.

Я изучив базовые сведения о Markdown, научился оформлять отчеты. Освоил синтаксис данного языка разметки. Выполнил 2 лабораторную работу на Markdown

---