Лекция 0

Влюбляемся в терминал

Цели этого курса

- Научиться создавать адаптивные сайты в том числе с использованием Bootstrap;
- Получить навыки быстрого создания прототипов с использованием готовых компонентов;
- Научиться решать задачи используя JavaScript.

Особенности этого курса

- Интенсивность курса очень высокая, пропуск одного занятия означает полное непонимание следующего;
- Количество домашних заданий довольно большое, это сделано для большего погружения в незнакомую среду;
- Процесс работы с Git и BASH несколько отличаются от Начального курса;
- У вас будет линтер, который будет проверять код на соответствие некоторым требованиям;
- Структуру каталогов менять нельзя;

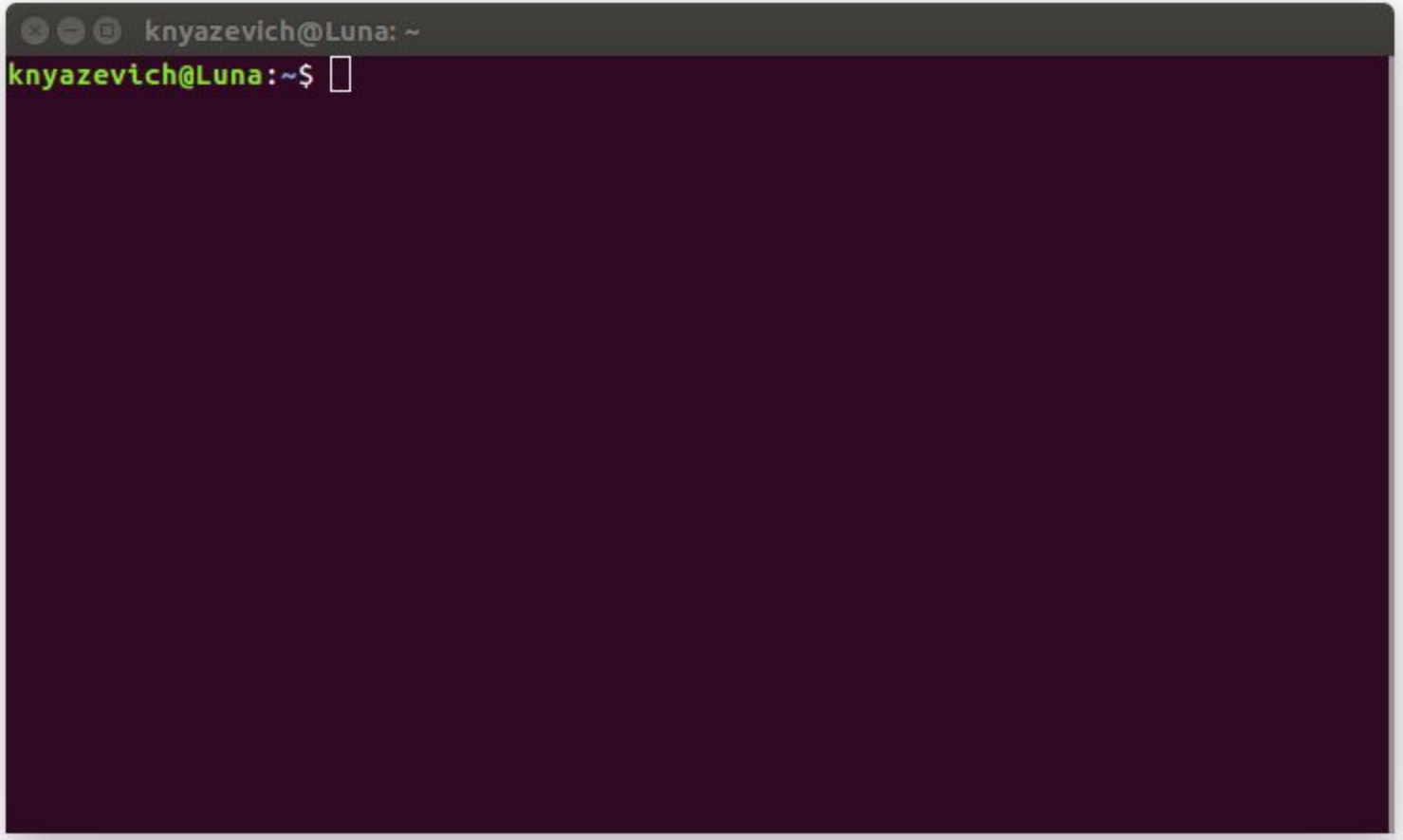
Цели этого занятия

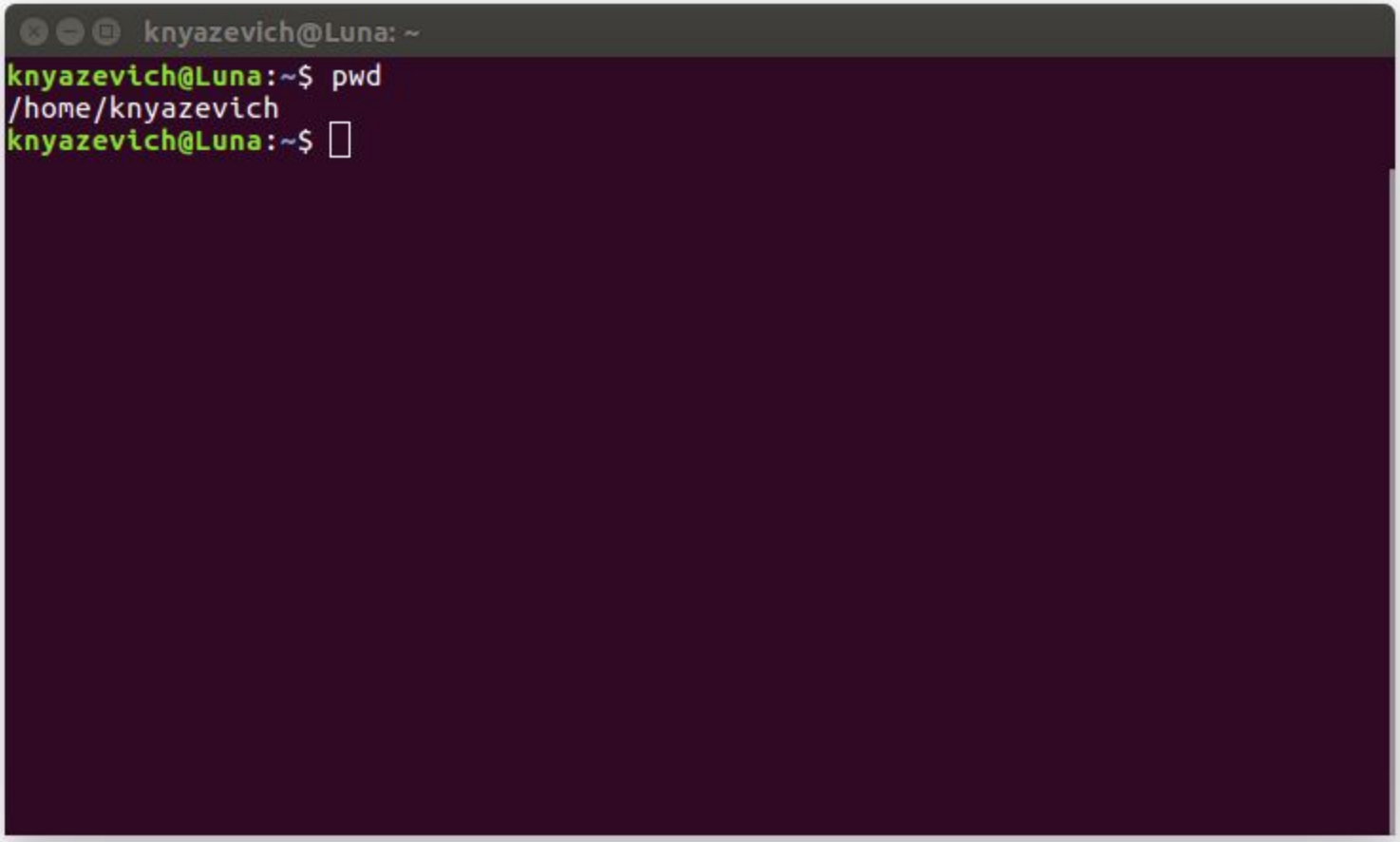
- Познакомиться с самыми основными командами BASH;
- Вспомнить основы Git и Github;
- Познакомиться с ветками;
- Настроить рабочее окружение.

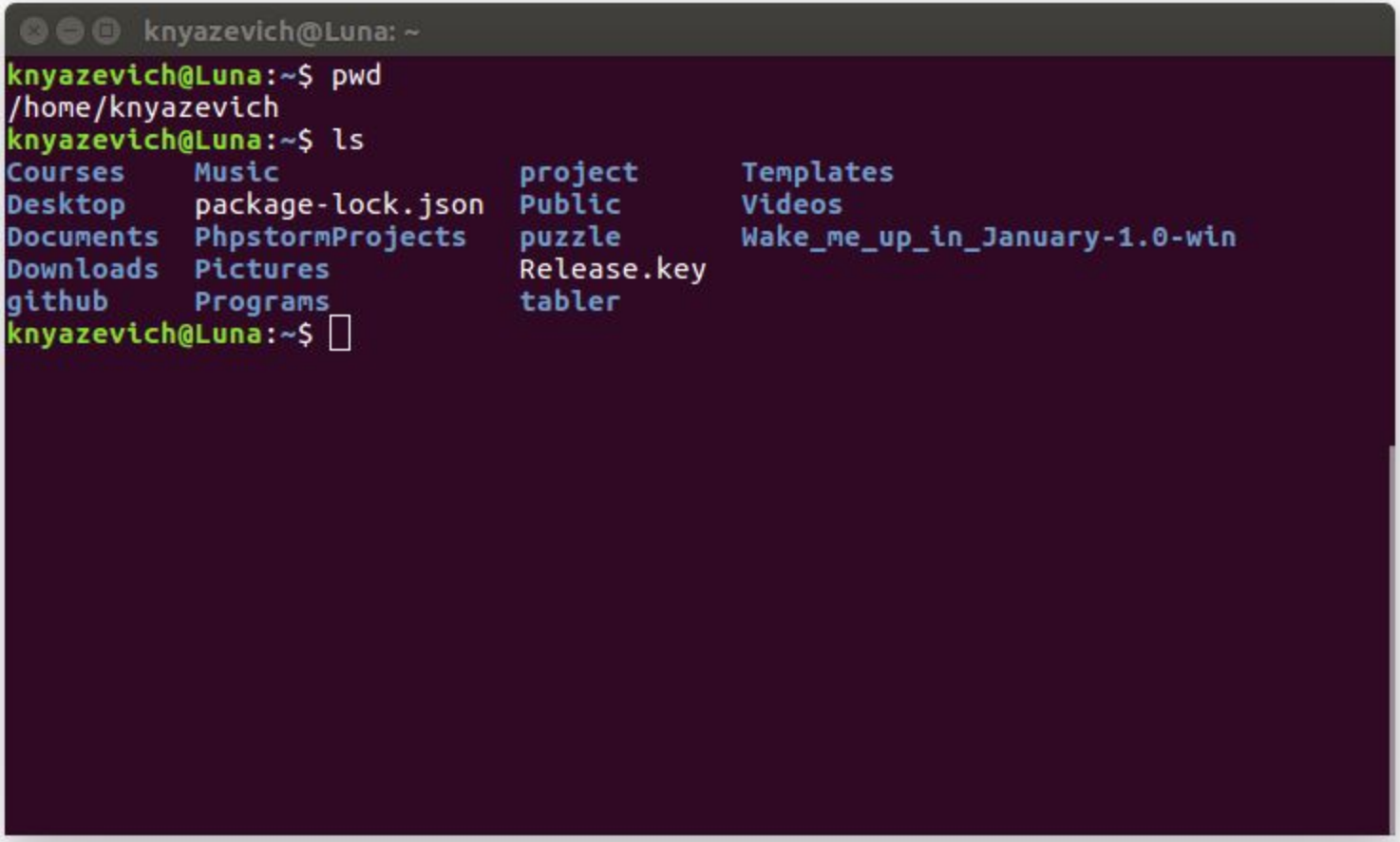
BASH –

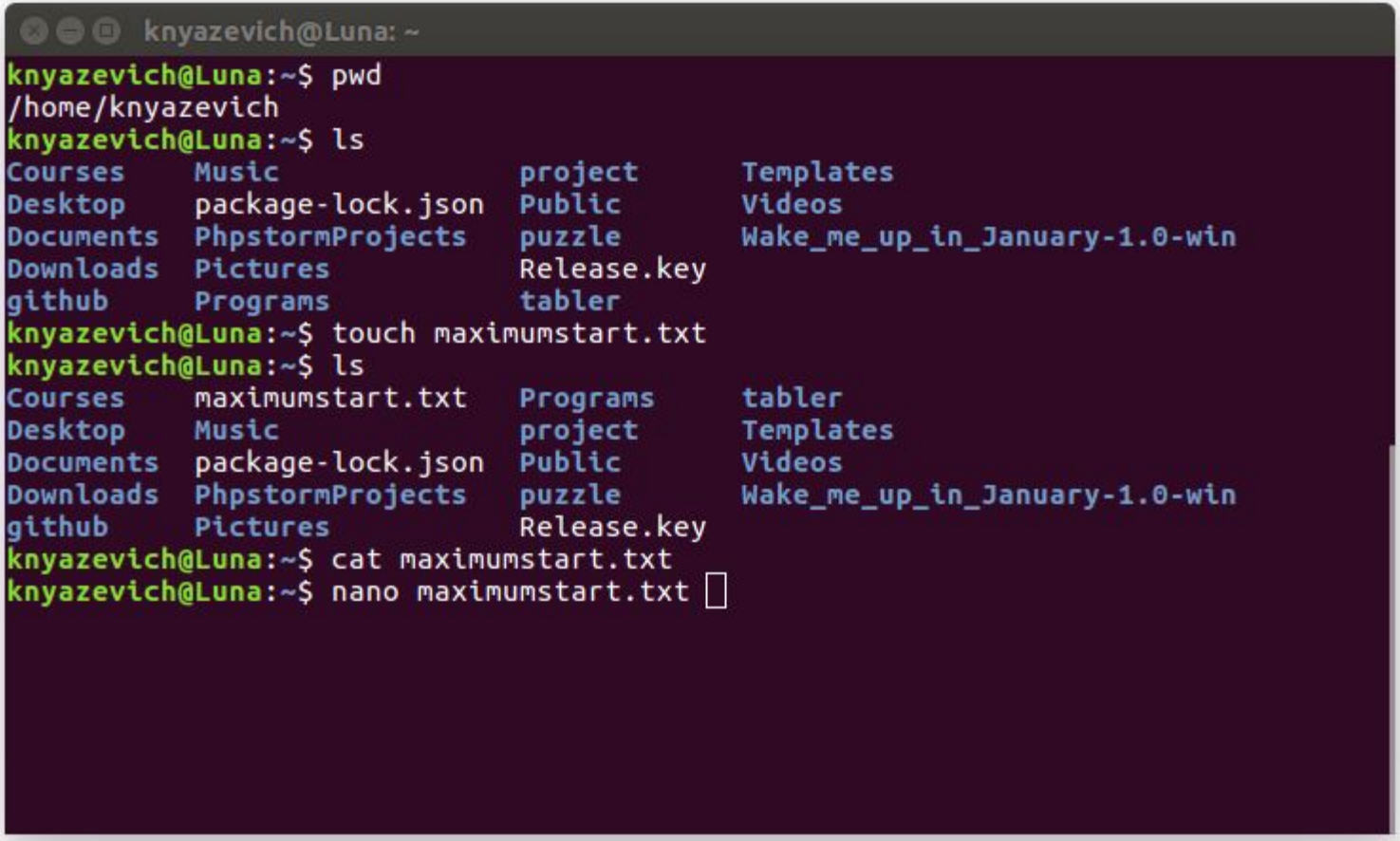
одна из наиболее популярных современных разновидностей командной оболочки UNIX. Особенно популярна в среде Linux, где она часто используется в качестве предустановленной командной оболочки.

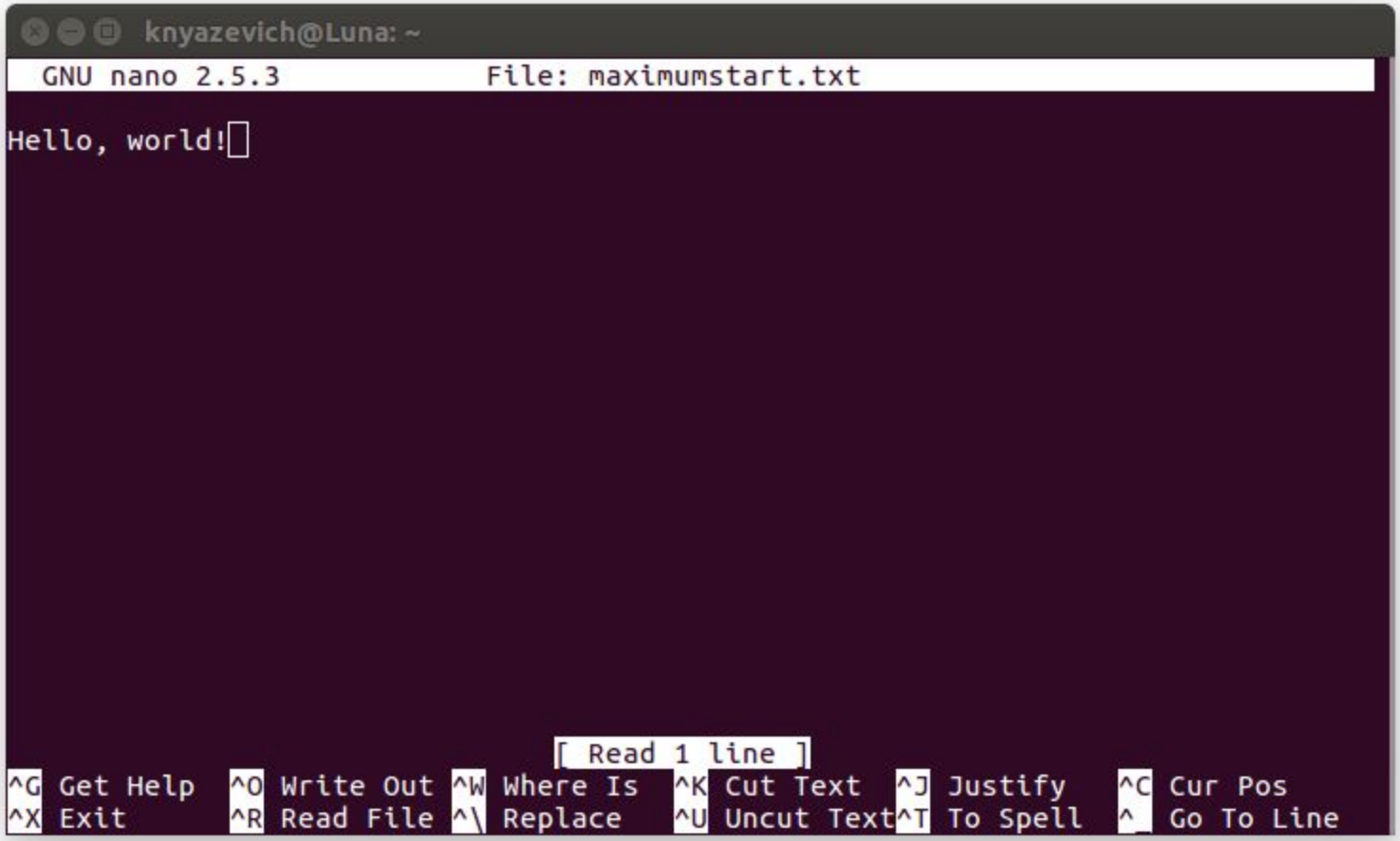
Основы работы с BASH


knyazevich@Luna: ~
knyazevich@Luna:~$

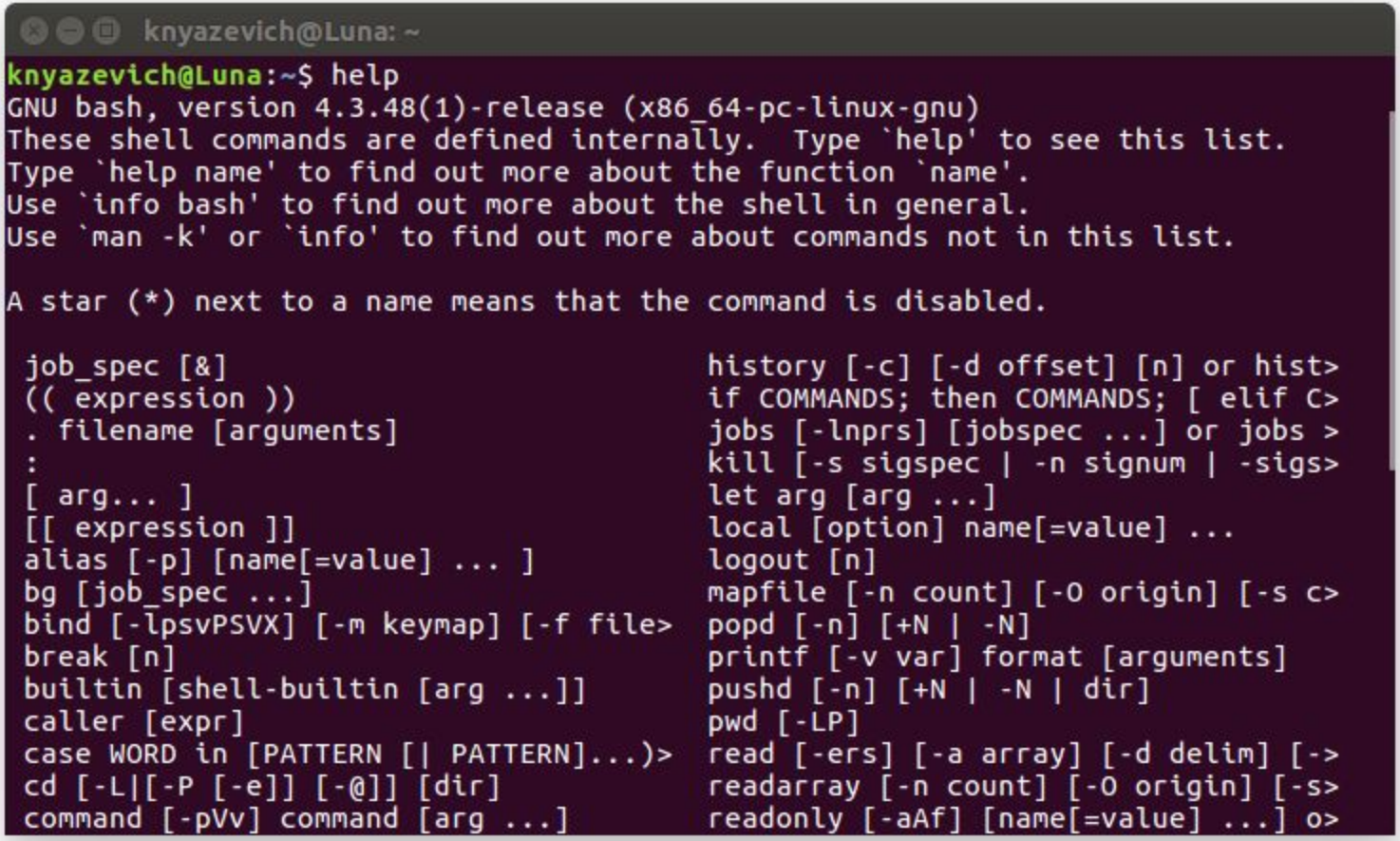

knyazevich@Luna: ~
knyazevich@Luna:~$ pwd
/home/knyazevich
knyazevich@Luna:~$


knyazevich@Luna: ~
knyazevich@Luna:~$ pwd
/home/knyazevich
knyazevich@Luna:~$ ls
Courses Music package-lock.json project Templates
Desktop Public Videos
Documents PhpstormProjects puzzle Wake_me_up_in_January-1.0-win
Downloads Pictures Release.key
github Programs tabler
knyazevich@Luna:~$

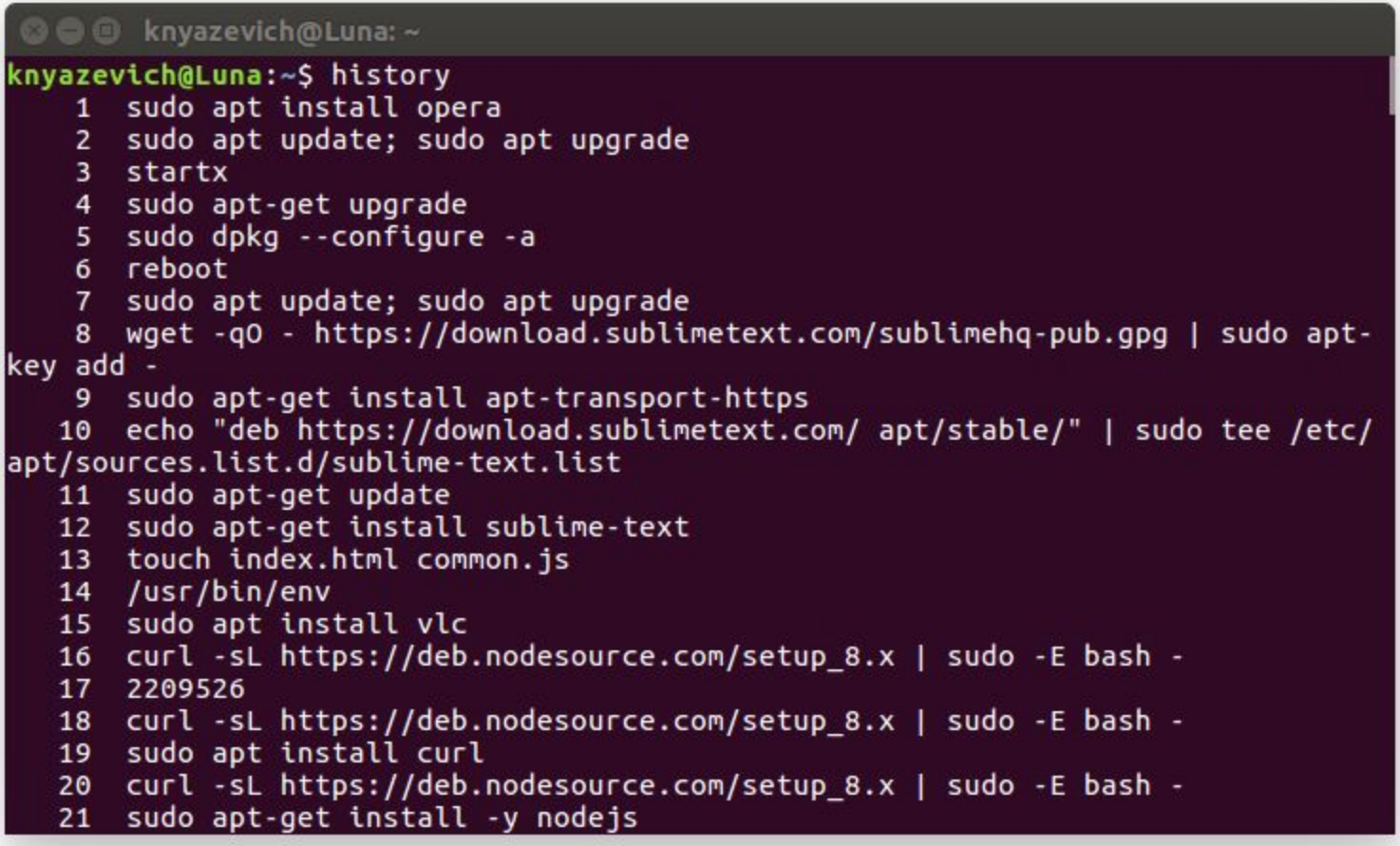

knyazevich@Luna: ~
knyazevich@Luna:~$ pwd
/home/knyazevich
knyazevich@Luna:~$ ls
Courses Music project Templates
Desktop package-lock.json Public Videos
Documents PhpstormProjects puzzle Wake_me_up_in_January-1.0-win
Downloads Pictures Release.key
github Programs tabler
knyazevich@Luna:~$ touch maximumstart.txt
knyazevich@Luna:~$ ls
Courses maximumstart.txt Programs tabler
Desktop Music project Templates
Documents package-lock.json Public Videos
Downloads PhpstormProjects puzzle Wake_me_up_in_January-1.0-win
github Pictures Release.key
knyazevich@Luna:~$ cat maximumstart.txt
knyazevich@Luna:~$ nano maximumstart.txt


knyazevich@Luna: ~
GNU nano 2.5.3 File: maximumstart.txt
Hello, world!
[Read 1 line]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line

Помощь


knyazevich@Luna: ~
knyazevich@Luna:~$ help
GNU bash, version 4.3.48(1)-release (x86_64-pc-linux-gnu)
These shell commands are defined internally. Type `help' to see this list.
Type `help name' to find out more about the function `name'.
Use `info bash' to find out more about the shell in general.
Use `man -k' or `info' to find out more about commands not in this list.
A star (*) next to a name means that the command is disabled.
job_spec [&]
((expression))
. filename [arguments]
:
[arg...]
[[expression]]
alias [-p] [name[=value] ...]
bg [job_spec ...]
bind [-lpsvPSVX] [-m keymap] [-f file>
break [n]
builtin [shell-builtin [arg ...]]
caller [expr]
case WORD in [PATTERN [| PATTERN]...)>
cd [-L|[-P [-e]] [-@]] [dir]
command [-pVv] command [arg ...]
history [-c] [-d offset] [n] or hist>
if COMMANDS; then COMMANDS; [elif C>
jobs [-lnprs] [jobspec ...] or jobs >
kill [-s sigspec | -n signum | -sigs>
let arg [arg ...]
local [option] name[=value] ...
logout [n]
mapfile [-n count] [-O origin] [-s c>
popd [-n] [+N | -N]
printf [-v var] format [arguments]
pushd [-n] [+N | -N | dir]
pwd [-LP]
read [-ers] [-a array] [-d delim] [->
readarray [-n count] [-O origin] [-s>
readonly [-aAf] [name[=value] ...] o>

История команд


knyazevich@Luna: ~
knyazevich@Luna:~$ history
1 sudo apt install opera
2 sudo apt update; sudo apt upgrade
3 startx
4 sudo apt-get upgrade
5 sudo dpkg --configure -a
6 reboot
7 sudo apt update; sudo apt upgrade
8 wget -qO - https://download.sublimetext.com/sublimehq-pub.gpg | sudo apt-key add -
9 sudo apt-get install apt-transport-https
10 echo "deb https://download.sublimetext.com/ apt/stable/" | sudo tee /etc/apt/sources.list.d/sublime-text.list
11 sudo apt-get update
12 sudo apt-get install sublime-text
13 touch index.html common.js
14 /usr/bin/env
15 sudo apt install vlc
16 curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
17 2209526
18 curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
19 sudo apt install curl
20 curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
21 sudo apt-get install -y nodejs

Немного обобщим

- **pwd (print working directory) –** показывает, откуда запущено окно терминала;
- **ls (list) –** выводит перечень файлов и папок;
- **touch –** создает или обновляет файл;
- **mkdir (make directory)** -- создает папку;
- **nano –** запускает текстовый редактор;
- **cat –** выводит содержимое файла;
- **du –** выводит размер файлов в папке;

Немного обобщим

- **cd (change directory) –** переходит по указанному пути;
- **rm (remove) –** удаляет файл;
- **rm -r (remove) –** удаляет папку;
- **more / less –** выводит содержимое файла с удобным чтением;
- **clear –** очищает текущее окно терминала;
- **cp –** Позволяет копировать файл;
- **mv –** позволяет переместить файл;
- **rmdir (remove directory) –** удалить пустую папку;

Немного обобщим

- **history –** показывает историю команд за все время;
- **help –** показывает версию терминала и доступные команды;
- **Tab один раз –** подсказывает путь/имя файла/команду;
- **Tab два раза –** показывает возможные дополнения;
- **Стрелка вверх –** показывает предыдущие команды;
- **Ctrl+C –** прекратить выполнение команды;
- **Q –** выйти из режима more или less;
- **ESC + :q –** выйти из редактора Vim

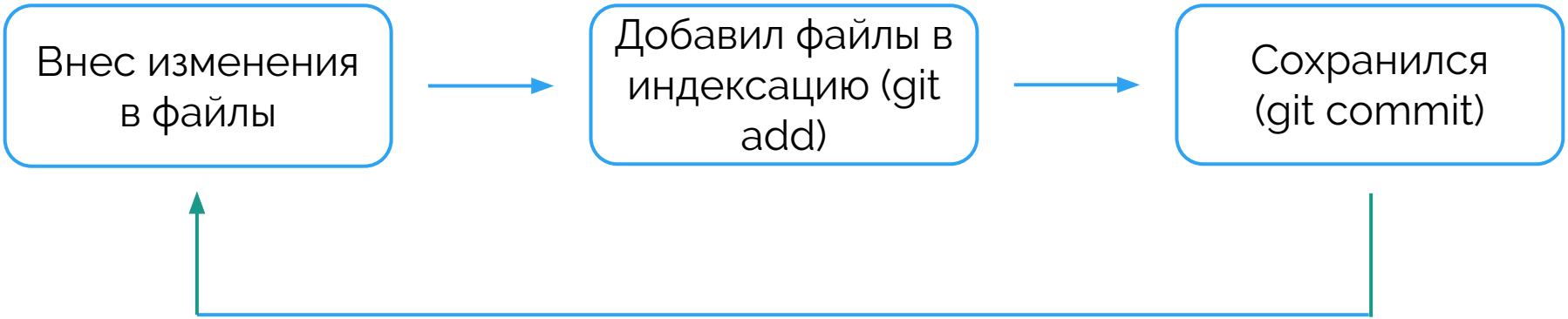
Git

Git –

распределенная система контроля версий, которая рассчитана на командную работу над проектом.

Как работает Git?

Схема работы с git



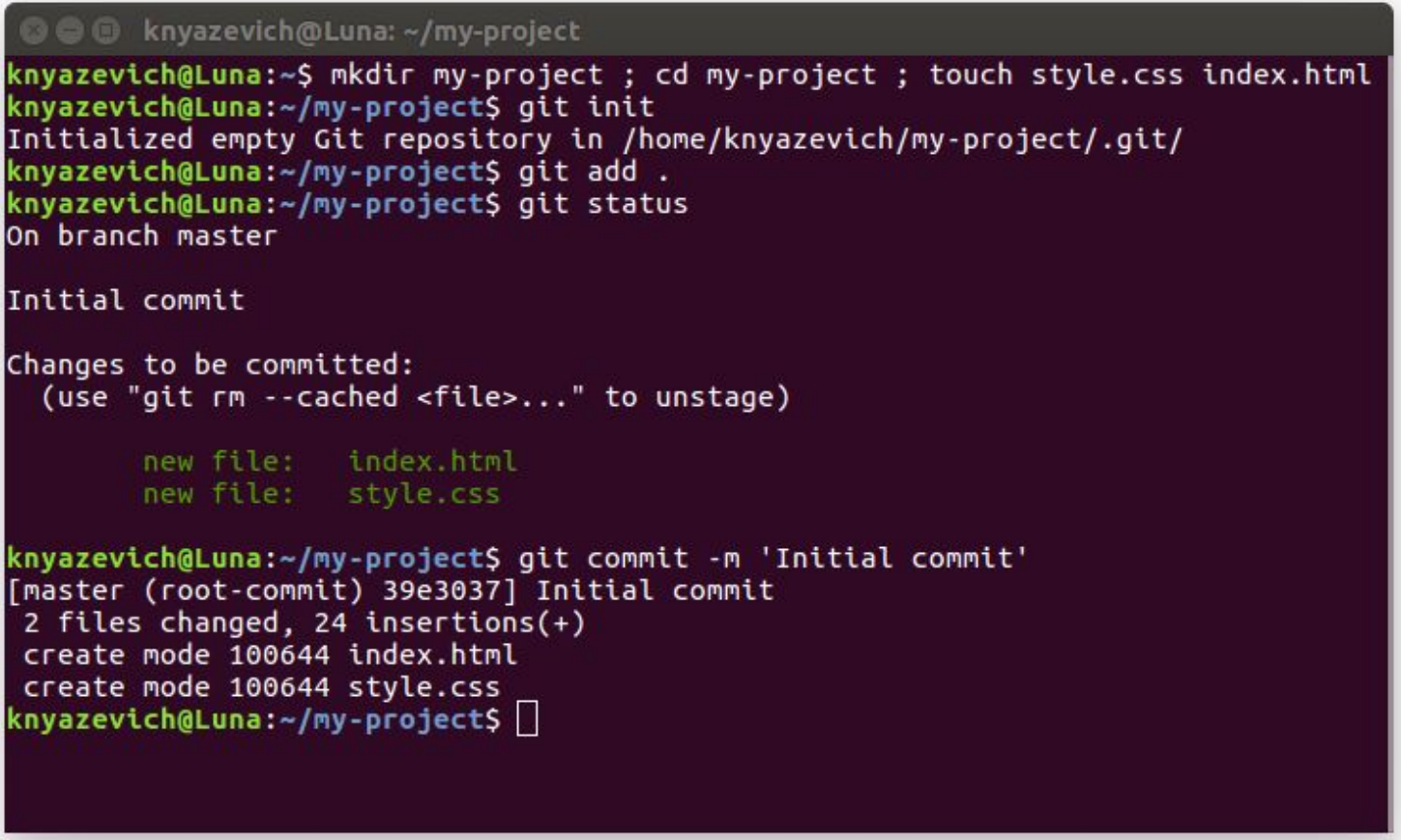
Начало работы

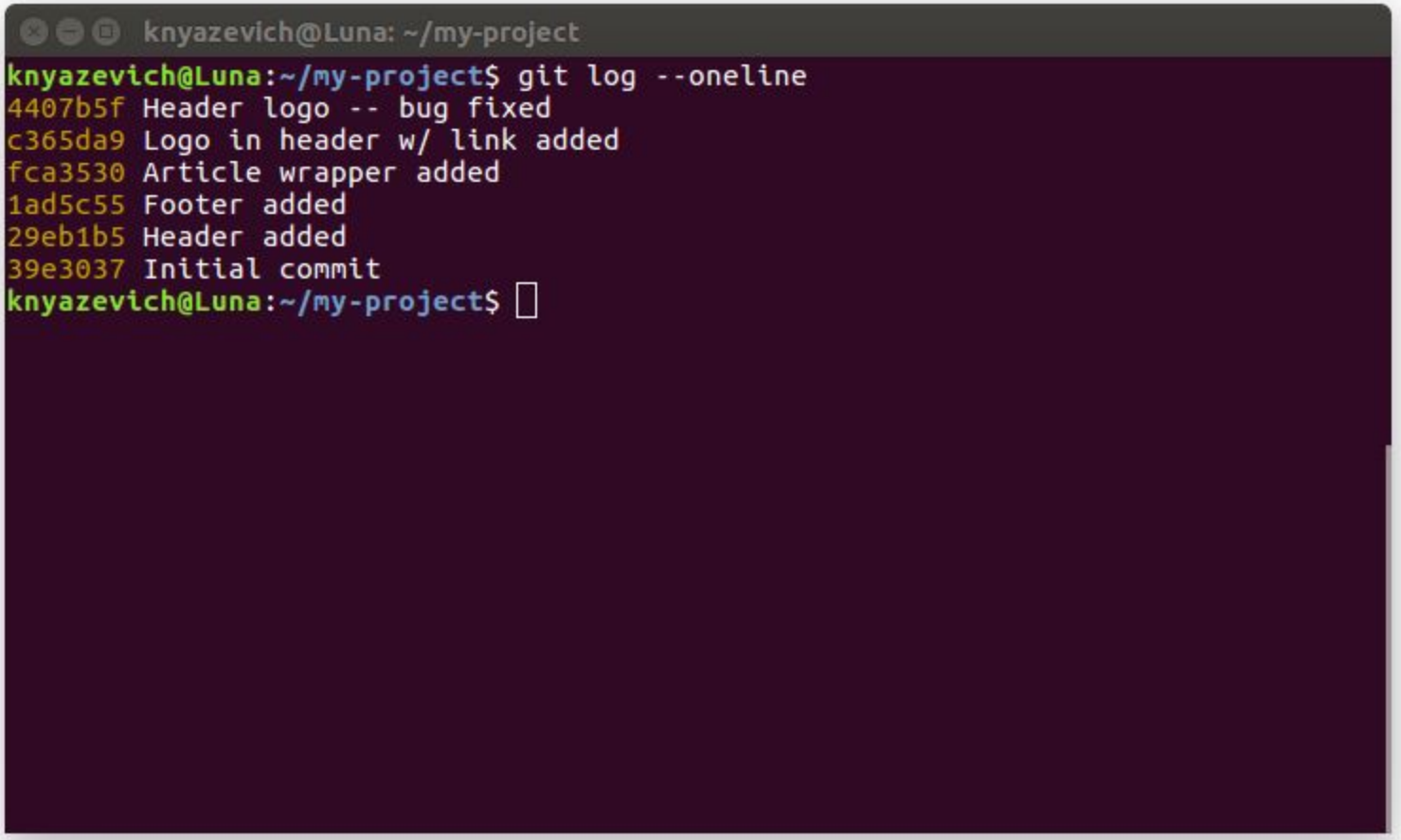
Перед началом работы, необходимо указать в конфиге имя и email, которые потом будут указываться в каждом коммите. Делается это с помощью команд:

git config --global user.name “Your Name”

git config --global user.email “your_email@example.com”

Локальный репозиторий


knyazevich@Luna: ~/my-project
knyazevich@Luna:~$ mkdir my-project ; cd my-project ; touch style.css index.html
knyazevich@Luna:~/my-project$ git init
Initialized empty Git repository in /home/knyazevich/my-project/.git/
knyazevich@Luna:~/my-project$ git add .
knyazevich@Luna:~/my-project$ git status
On branch master
Initial commit
Changes to be committed:
(use "git rm --cached <file>..." to unstage)
new file: index.html
new file: style.css
knyazevich@Luna:~/my-project$ git commit -m 'Initial commit'
[master (root-commit) 39e3037] Initial commit
2 files changed, 24 insertions(+)
create mode 100644 index.html
create mode 100644 style.css
knyazevich@Luna:~/my-project$


knyazevich@Luna: ~/my-project
knyazevich@Luna:~/my-project$ git log --oneline
4407b5f Header logo -- bug fixed
c365da9 Logo in header w/ link added
fca3530 Article wrapper added
1ad5c55 Footer added
29eb1b5 Header added
39e3037 Initial commit
knyazevich@Luna:~/my-project$

В Git мы всегда можем вернуться к любому состоянию (коммиту), используя команду:

git checkout <hash>

.gitignore –

позволяет указать файлы или папки, которые не нужно индексировать. То есть, эти файлы никогда не улетят на GitHub.

Метки (теги)

Метки позволяют отмечать важные состояния. Обычно их используют для отметки момента выпуска версий (0.1, 0.2, 1.0 и т. д).

Для просмотра меток:

git tag

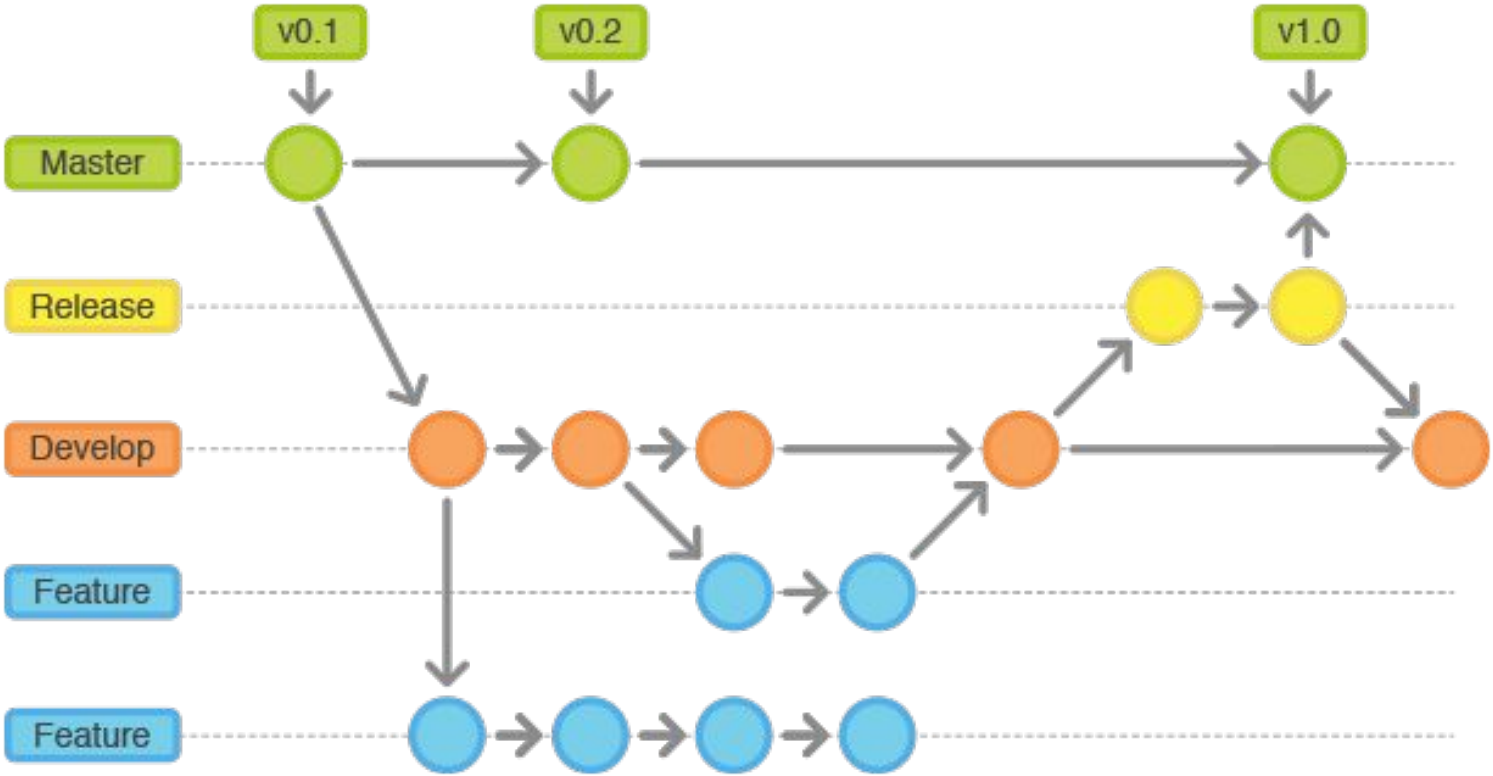
Для создания метки:

git tag -a <название> -m <описание>

Какие команды мы уже знаем?

- **git init** – инициирует пустой репозиторий;
- **git add <*file*> / <.>** – добавляет файл / все файлы;
- **git commit -m ‘Message’** – создает коммит;
- **git log --oneline (-1 --all --graph)** – выводит историю коммитов;
- **git remote add <*name*> <*path*>** – добавляет внешний путь;
- **git push** – отправляет коммиты на сервер;
- **git pull** – загружает коммиты с сервера;
- **git config …** – вносит изменения в конфигурационный файл;

Ветки


v0.1
v0.2
v1.0
Master
Release
Develop
Feature
Feature

Для создания ветки используется команда:

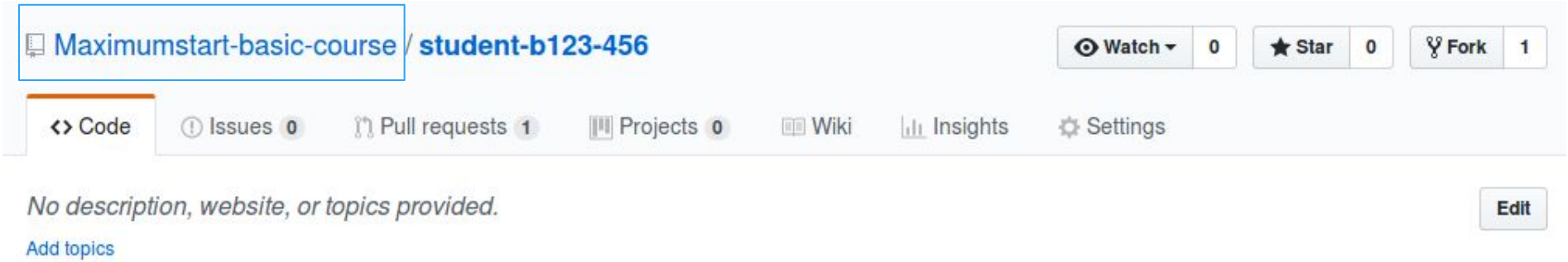
git checkout -b <имя ветки> <хеш коммита>

При этом мы сразу попадаем в ветку <имя ветки>, для того, чтобы вернуться в главную, используем команду:

git checkout master

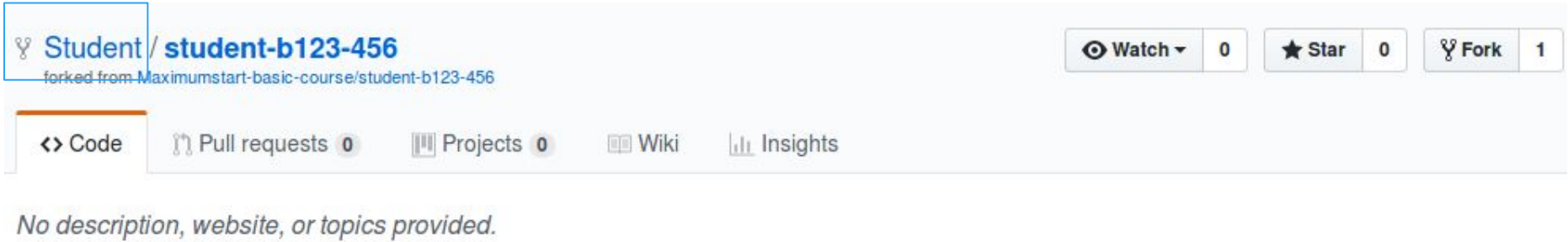
Как мы будем работать?

Основной репозиторий



это репозиторий, в котором будет находиться чистый код, в который **вы не можете** пушить свои коммиты непосредственно. Изменения в него попадают через **pull request**.

Личный репозиторий (форк)

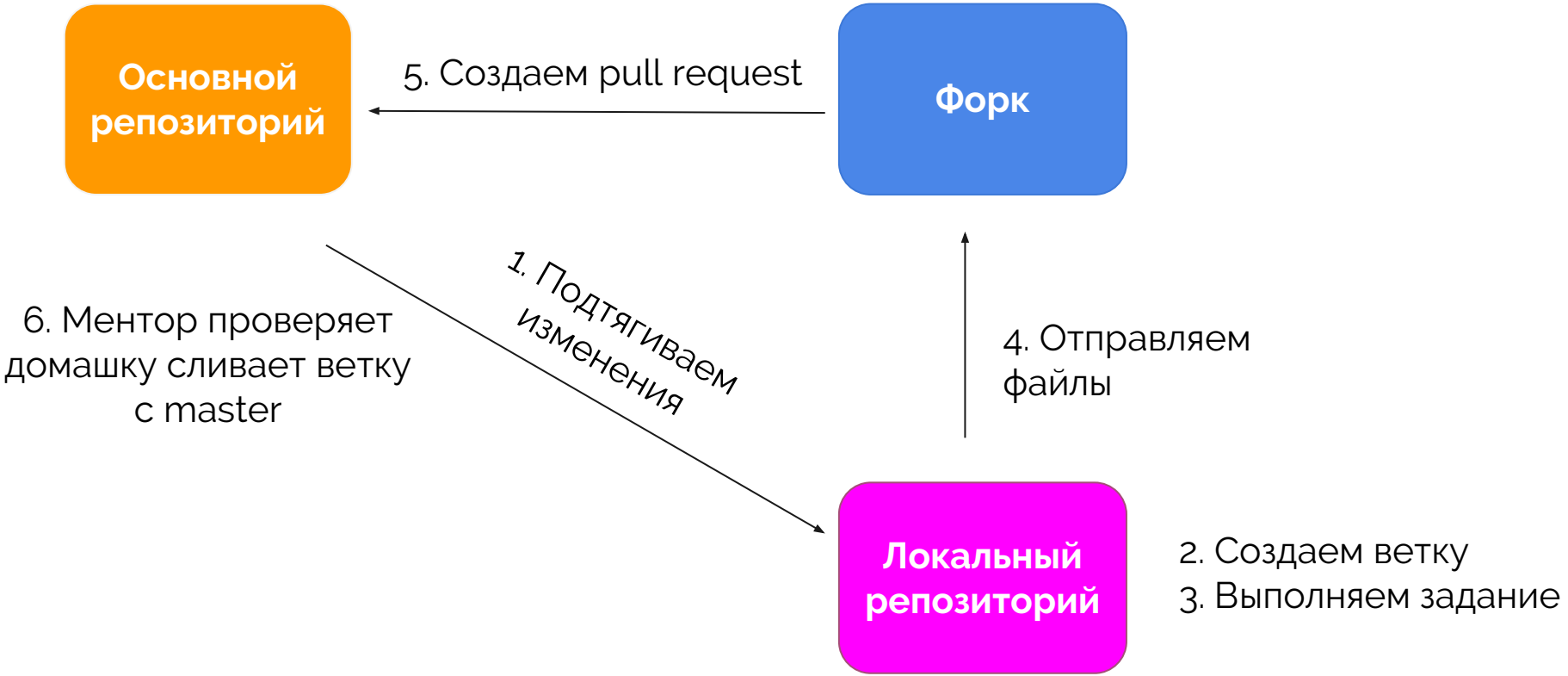


это ваш личный репозиторий, которые является форком основного и в который вы **можете** пушить. Именно его вы клонируете на компьютер и в созданной папке будете работать.

Локальный репозиторий (клонированный)

папка у вас на компьютерах – это клон личного репозитория, в нем вы будете **вести всю непосредственную работу**, которая затем будет отправляться в личный репозиторий.

Как сдается домашка?



Отправка задания на проверку

после того, как домашнее задание готово – необходимо создать **pull request**:

Отправка задания на проверку