Рабочее окружение ML разработчика

AI Education

План на сегодня

- **Основные инструменты**
- **Linux и терминал**
- **Знакомство с Git**

Рабочее окружение

- **Что делаем?**
  Пишем код на Python (версия 3.10)
- **Где пишем?**
  Jupyter Notebook, Google Colab
  PyCharm / VS Code
- **Как пишем?**
  Красиво! И складываем в Git!
- **А потом?**
  Открываем терминал и подключаемся к серверу (GNU/Linux)

Установка Python на локальный компьютер

- **Linux**
  Уже установлен! Проверяем версию командой:
  `$ python3 --version`

- **MacOS**
  Используем менеджер пакетов Homebrew
  `$ brew install python`

- **Windows**
  - python.org
  - WSL

Менеджер пакетов

- **Как установить программу на компьютер?**
- **А как надо?**
- **Что есть?**
  - DPKG (Debian)
  - RPM (Red Hat Package Manager)
  - APT (Ubuntu)

    ```
    $ apt install <название пакета>
    ```
  - …

Advanced Packaging Tool

- `$ sudo apt update`

  Обновление баз данных пакетов (указанных в **/etc/apt/sources.list**)

- `$ sudo apt upgrade`

  Обновление системы

- `$ sudo apt install <название пакета>`

  Установка пакета

- `$ sudo apt show <название пакета>`

  Вывести название пакета

Substitute User and do

- **Кто такой суперпользователь?**
  root
- **Некоторые команды можно выполнять только от имени** *root*
- **Раньше использовали su**
- **sudo**
  способ повышать привилегии в современных системах

Среда разработки

- **Самая популярная**
  VS Code
- **А в индустрии?**
  PyCharm
- **Для эстетов**
  Sublime Text, Vim (напишите в чатик, когда сможете выбраться!)

Работа в терминале

- **Где мы?**

```
$ pwd
```

- **Кто мы?**

```
$ whoami
```

- **Что здесь есть?**

```
$ ls
$ ls -l
$ ls -a
$ ls -h
```

Работа в терминале (продолжение)

- **Получить справку**
  `$ man <название команды>`

- **Отобразить историю ранее введенных команд**
  `$ history`

- **Очистить экран**
  `$ clear`

- **Какой дистрибутив Linux используется?**
  `$ lsb_release -a`

Виртуальное окружение

- **Создаем папку для проекта**

```
$ mkdir <название папки>
```

- **Переходим в нее**

```
$ cd <название папки>
```

- **Создаем виртуальное окружение**

```
$ python3 -m venv env
```

- **Активируем окружение**

```
$ source venv/bin/activate
```

Системы контроля версий

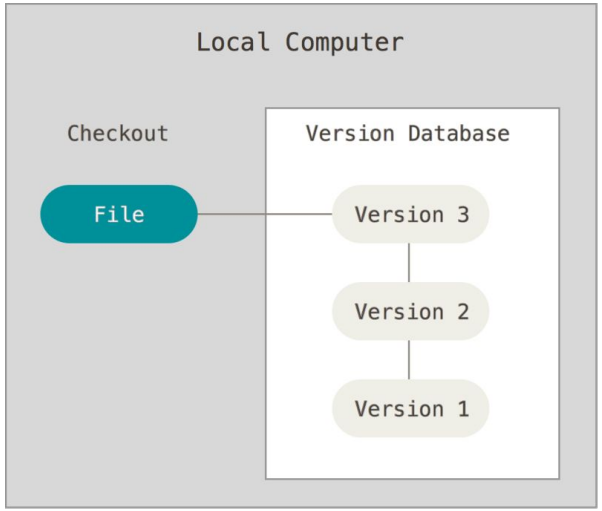
- Если вы работаете над проектом, тем более в составе команды, необходимо сохранять промежуточные этапы

- Система контроля версий - это система записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определенной версии

Локальные системы контроля версий

- Для каждого файла, зарегистрированного в системе, хранится полная история изменений

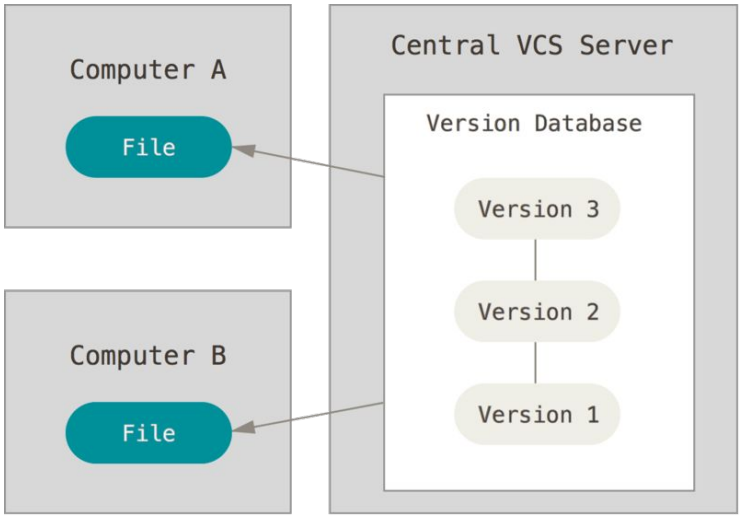
- Недостатки?



Централизованные системы контроля версий

- Используется единственный сервер, который содержит все версии файлов. Клиенты, обращаясь к этому серверу, получают код проекта

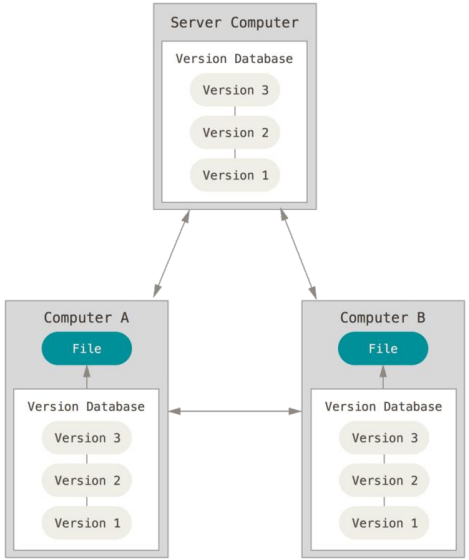
- Недостатки?



Распределенные системы контроля версий

- Клиенты не просто выгружают последние версии файлов, а полностью копируют весь *репозиторий* (место, где хранятся и поддерживаются какие-либо данные)

- Недостатки?



Установка git на локальный компьютер

- **Linux, Windows (WSL)**

  ```
  $ apt install git
  ```

- **MacOS**

  ```
  $ brew install git
  ```

- **Windows**

  Скачиваем дистрибутив отсюда!

Начальная настройка git

- **Локальная**
- **Глобальная**

```
$ git config --global user.name "The One"
$ git config --global user.email neo@example.com
```

- **Системная**

Домашнее задание

- **Создаем репозиторий проекта (с осмысленным названием)**
- **Заполняем README описанием задачи**
- **Присылаем ссылку на репозиторий в форму сдачи на Stepik**