Utrecht University

FACULTY OF SCIENCES

Programming Café

Managing coding (R & Python) environments with Conda and Docker

Libio Goncalves Braz
ICT Developer – Information and Computing Sciences department

Summary

- Do I need dependency management?
- Virtual environments (Python and R)
- Conda environments
- Docker containers and chroot

Do I need dependency management?

gpt4all 2.7.0

```
pip install gpt4all
```

✓ Dernière version

Dernière version : 31 mai 2024

Licence : MIT License

Created by: Nomic and the Open Source Community

Requires: Python >=3.8

Provides-Extra: `all`, `cuda`, `dev`

Python bindings for GPT4All

Do I need dependency management?

README License

AQ-GT / AQ-GT-A

This project is the official pytorch implementation of:
AQ-GT: a Temporally Aligned and Quantized GRU-Transformer for Co-Speech Gesture Synthesis

with the extension (AQ-GT-A) from the paper:
Augmented Co-Speech Gesture Generation: Including Form and Meaning Features to Guide Learning-Based Gesture Synthesis

State of the Art | Gesture Generation on TED Gesture Dataset

Install without Docker

This repository is developed and tested on Ubuntu 20.04, Python 3.7, and PyTorch 2.0+.

```
python=3.7
Pytorch
```

Do I need dependency management?

README

Self-Tuning Networks

This repository contains the code used for the paper Self-Tuning Networks: Bilevel Optimization of Hyperparameters using Structured Best-Response Functions (ICLR 2019).

Requirements

- Python 3.7
- Pytorch 1.8.x
- Stable Baselines 3

Do I need dependency management?

GPT4All	AQGT	STN
Python >= 3.8	Python = 3.7	Python = 3.7
	• pytorch > 2.0	• pytorch = 1.8 • Linux dependent?

- Avoid breaking old projects dependencies
- Have a clean an easy installation for testing purposes
- Improve reusability of software

Virtual environments (Python)

https://docs.python.org/3/library/venv.html

A virtual environment, Venv, is (amongst other things):

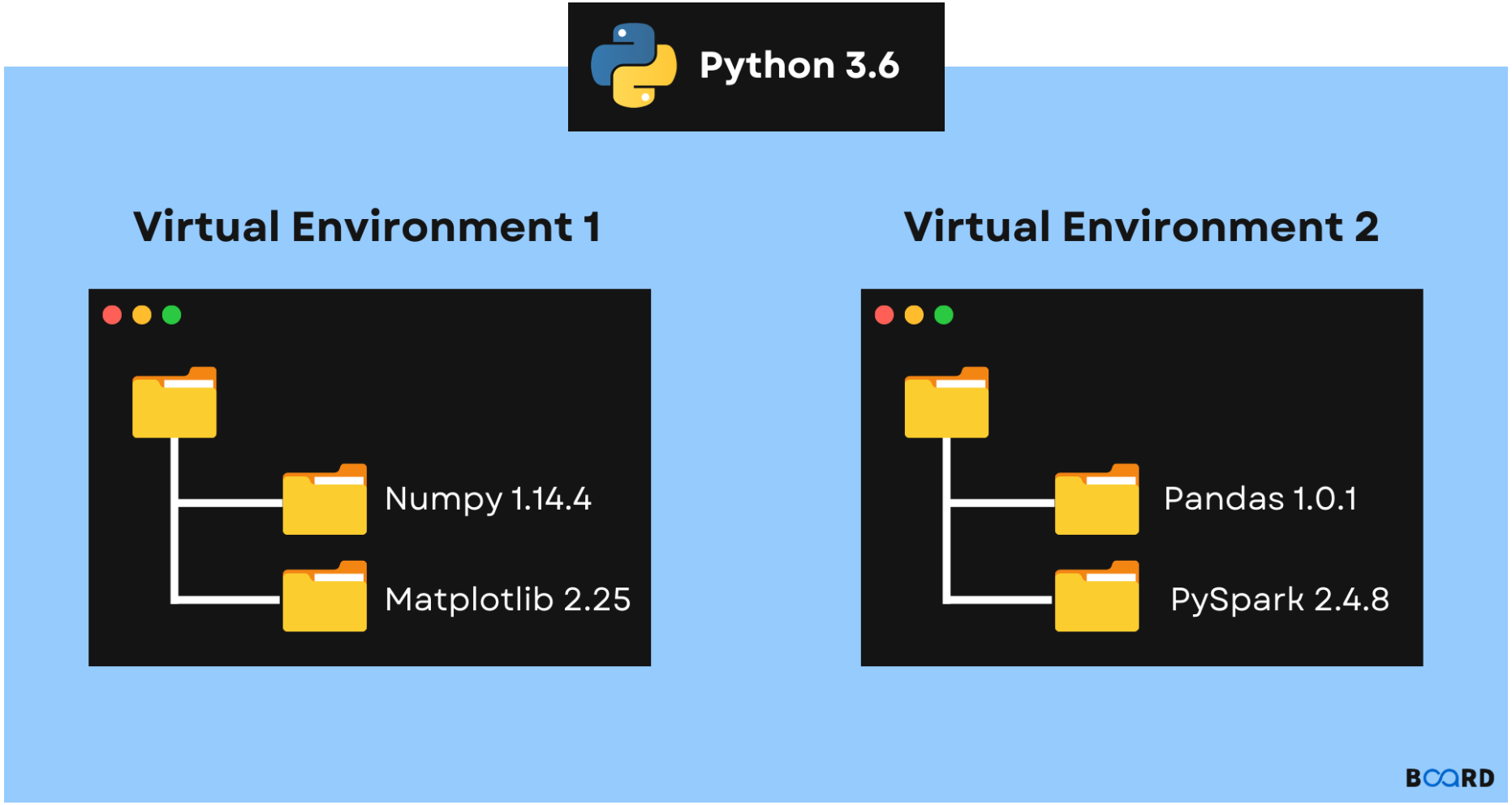
- Used to contain a specific Python interpreter and software libraries and binaries which are needed to support a project (library or application). These are by default isolated from software in other virtual environments and Python interpreters and libraries installed in the operating system.

- Contained in a directory, conventionally either named venv or .venv in the project directory, or under a container directory for lots of virtual environments, such as ~/.virtualenvs.

- Considered as disposable – it should be simple to delete and recreate it from scratch. You don't place any project code in the environment

Virtual environments (Python)

https://www.boardinfinity.com/blog/python-virtual-environment/



Virtual environments (Python)

https://packaging.python.org/en/latest/guides/installing-using-pip-and-virtual-environments/#create-and-use-virtual-environments

Create a new virtual environment

venv (for Python 3) allows you to manage separate package installations for different projects. It creates a "virtual" isolated Python installation. When you switch projects, you can create a new virtual environment which is isolated from other virtual environments. You benefit from the virtual environment since packages can be installed confidently and will not interfere with another project's environment.

> **Tip**
>
> It is recommended to use a virtual environment when working with third party packages.

To create a virtual environment, go to your project's directory and run the following command. This will create a new virtual environment in a local folder named `.venv`:

Unix/macOS Windows

```
python3 -m venv .venv
```

Virtual environments (Python)

https://packaging.python.org/en/latest/guides/installing-using-pip-and-virtual-environments/#create-and-use-virtual-environments

Activate a virtual environment

Before you can start installing or using packages in your virtual environment you'll need to `activate` it. Activating a virtual environment will put the virtual environment-specific `python` and `pip` executables into your shell's `PATH`.

Unix/macOS Windows

```
source .venv/bin/activate
```

To confirm the virtual environment is activated, check the location of your Python interpreter:

Unix/macOS Windows

```
which python
```

Virtual environments (Python)

https://packaging.python.org/en/latest/guides/installing-using-pip-and-virtual-environments/#create-and-use-virtual-environments

Install packages using pip

When your virtual environment is activated, you can install packages. Use the `pip install` command to install packages.

Install a package

For example,let's install the Requests library from the Python Package Index (PyPI):

Unix/macOS Windows

```
python3 -m pip install requests
```

Virtual environments (R)

https://rstudio.github.io/renv/

The same principle also exists in R:

- Isolated: Installing a new or updated package for one project won't break your other projects, and vice versa. That's because renv gives each project its own private library.
- Portable: Easily transport your projects from one computer to another, even across different platforms. renv makes it easy to install the packages your project depends on.
- Reproducible: renv records the exact package versions you depend on, and ensures those exact versions are the ones that get installed wherever you go.

Virtual environments (R)

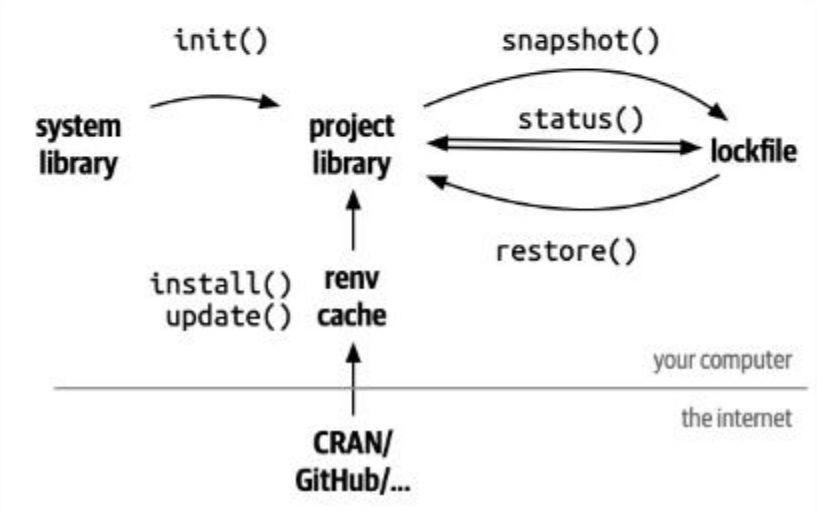
https://rstudio.github.io/renv/

Installation

Install the latest version of renv from CRAN with:

```
install.packages("renv")
```

Workflow



Virtual environments

https://mindthevirt.com/venv-vs-conda-choosing-the-right-python-environment-manager-for-you/

Why to use venv/renv:

- Simplicity: Venv/Renv is straightforward, focusing solely on creating isolated Python/R environments.
- Lightweight: It's part of the Python/R standard library, so there's nothing extra to install if you have Python/R.
- Specificity: Venv/Renv is Python/R-specific, making it a focused tool for Python/R developers.

Virtual environments

Does Venv solve our initial problem?

GPT4All	AQGT	STN
Python >= 3.8	Python = 3.7	Python = 3.7
	• pytorch > 2.0	• pytorch = 1.8 • Linux dependent?

Conda environments

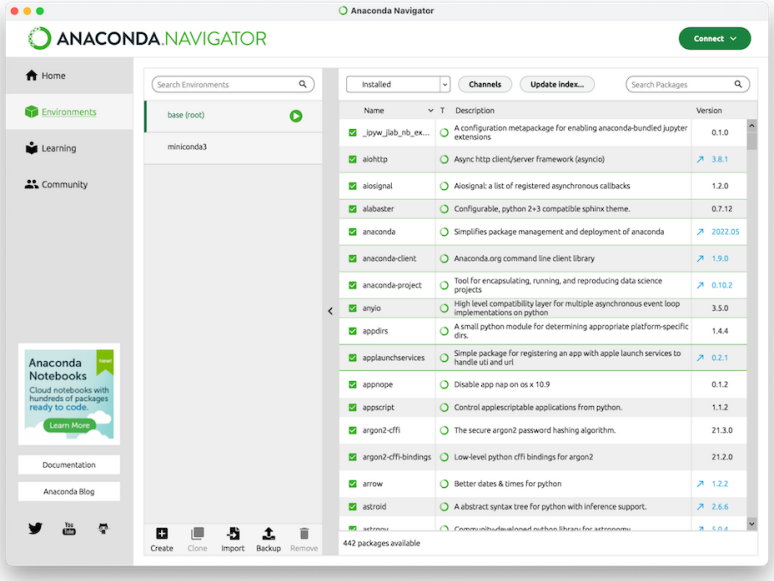
The best of both worlds(?)

https://www.anaconda.com/download

Conda

Open-source package and environment management system that runs on Windows, macOS, and Linux. Install, run, and update packages and their dependencies

```
[bash-3.2$ conda install pandas
Collecting package metadata (current_repodata.json) : done
Solving environment: done

## Package Plan ##

    environment location: /opt/anaconda3

    added / updated specs:
        - pandas

The following packages will be downloaded:

package                |        build
-------------------------------------
pandas- 1.             | py39he9d5cce_1         9.4 MB
-------------------------------------------------------
                                  Total:         9.4 MB

The following packages will be UPDATED:
```



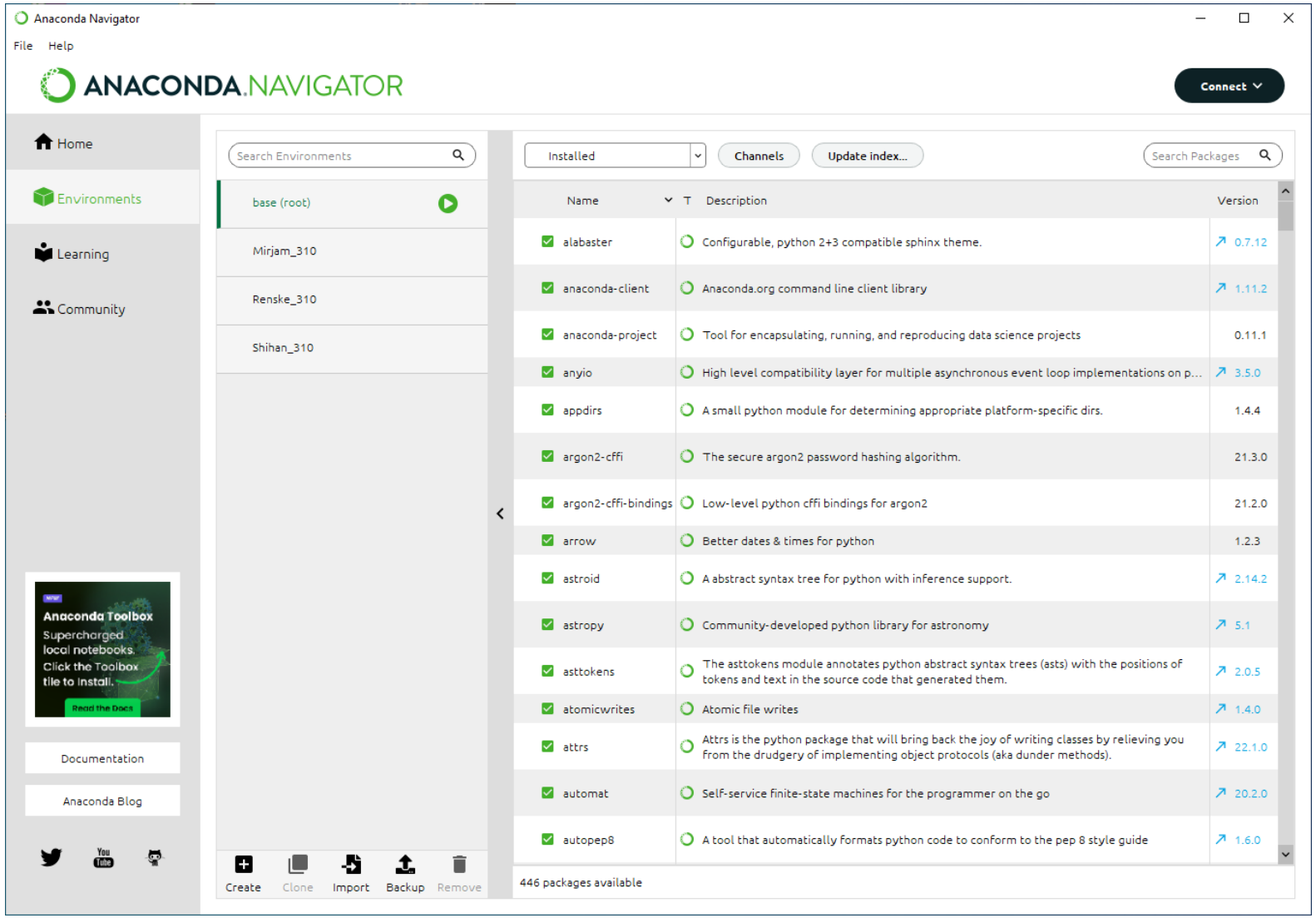
Navigator

Desktop application lets you easily manage integrated applications, packages, and environments without using the command line.

Conda environments

The best of both worlds(?)

https://www.anaconda.com/download/success

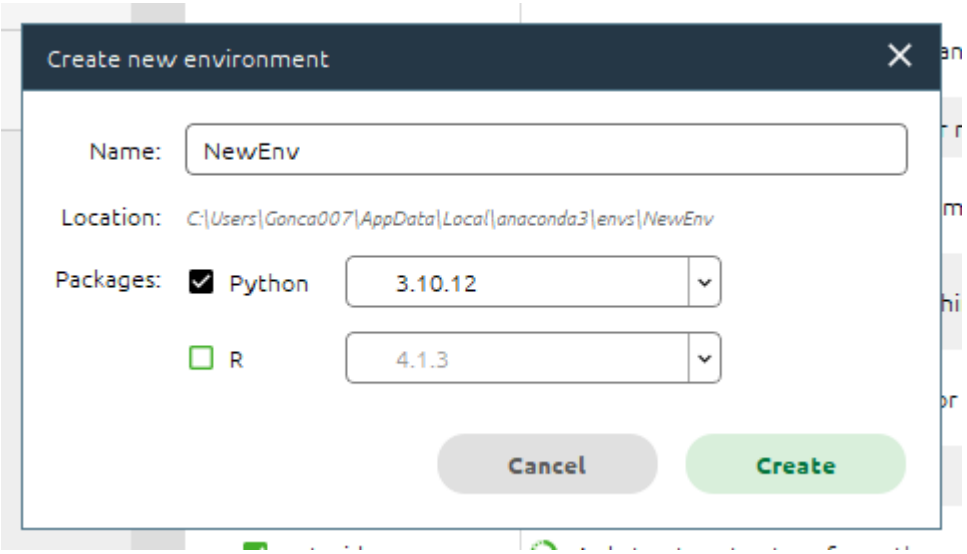


Conda environments

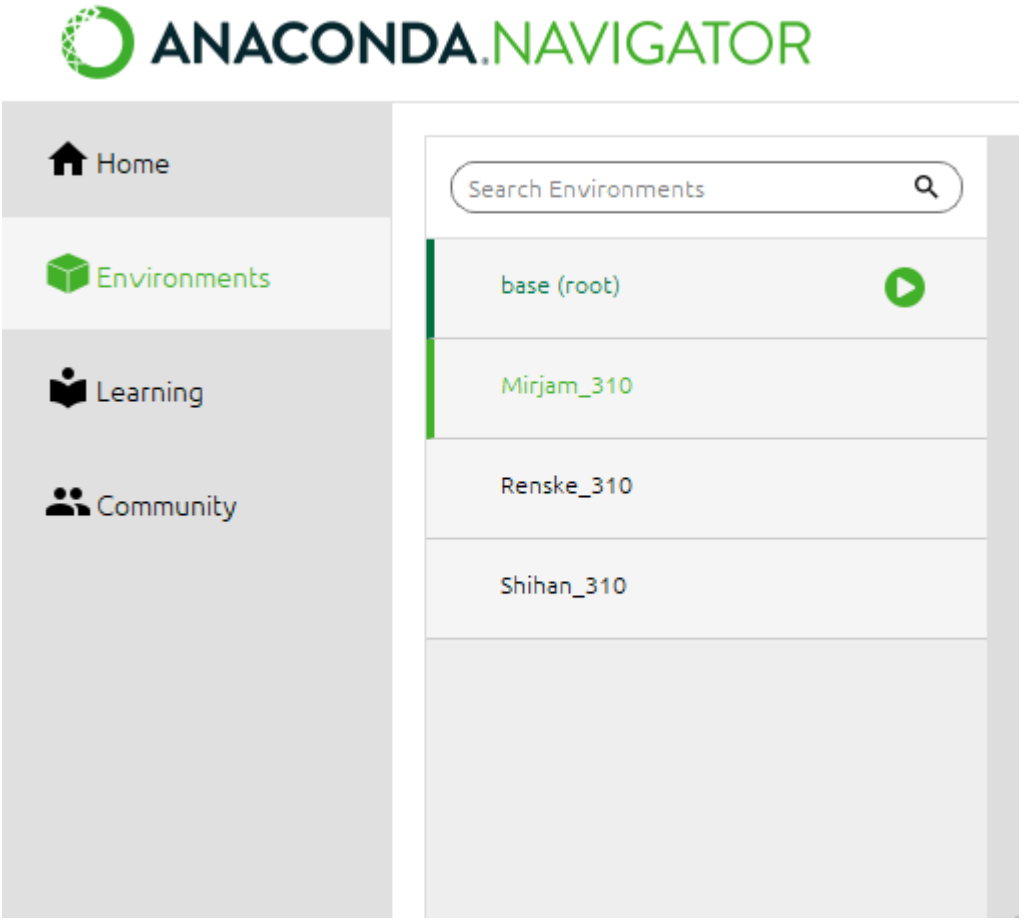
The best of both worlds(?)

https://www.anaconda.com/download/success

Easily create new environments…



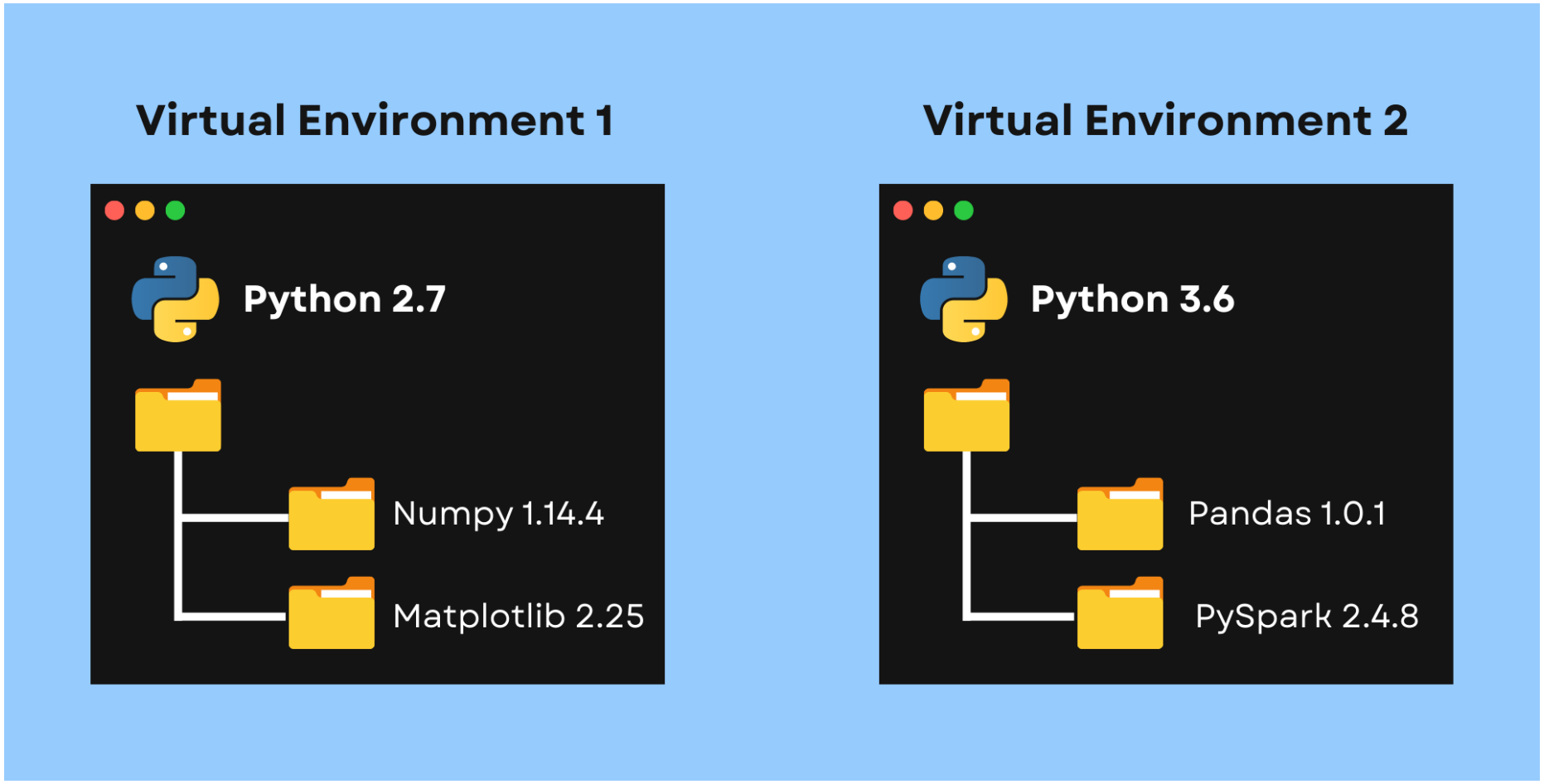
…and get an overview on your environments



Conda environments

The best of both worlds(?)

https://www.boardinfinity.com/blog/python-virtual-environment/



Conda environments

The best of both worlds(?)

https://docs.anaconda.com/free/anacondaorg/user-guide/packages/installing-packages/

Installed | Channels | Update index... | Search Packages

Name	T	Description	Version
alabaster		Configurable, python 2+3 compatible sphinx theme.	0.7.12
anaconda-client		Anaconda.org command line client library	1.11.2
anaconda-project		Tool for encapsulating, running, and reproducing data science projects	0.11.1
anyio		High level compatibility layer for multiple asynchronous event l...	3.5.0
appdirs		A small python module for determining appropriate platform-specific dirs.	1.4.4
argon2-cffi		The secure argon2 password hashing algorithm.	21.3.0
argon2-cffi-bindings		Low-level python cffi bindings for argon2	21.2.0
arrow		Better dates & times for python	1.2.3
astroid		A abstract syntax tree for python with inference support.	2.14.2
astropy		Community-developed python library for astronomy	5.1
asttokens		The asttokens module annotates python abstract syntax trees (asts) with the positions of tokens and text in the source code that generated them.	2.0.5

You can filter search results using three filter controls:

- `Type`: All, conda only, Standard Python only, Standard R only, or Notebooks
- `Access`: All, Public, Private (only available if you are logged in and have specific permissions), or Authenticated (only available if you are logged in)
- `Platform`: All, source, linux-32, linux-64, linux-aarch64, linux-armv61, linux-armv71, linux-ppc641e, linux-s390x, noarch, osx-32, osx-64, win-32, or win-64

Tip

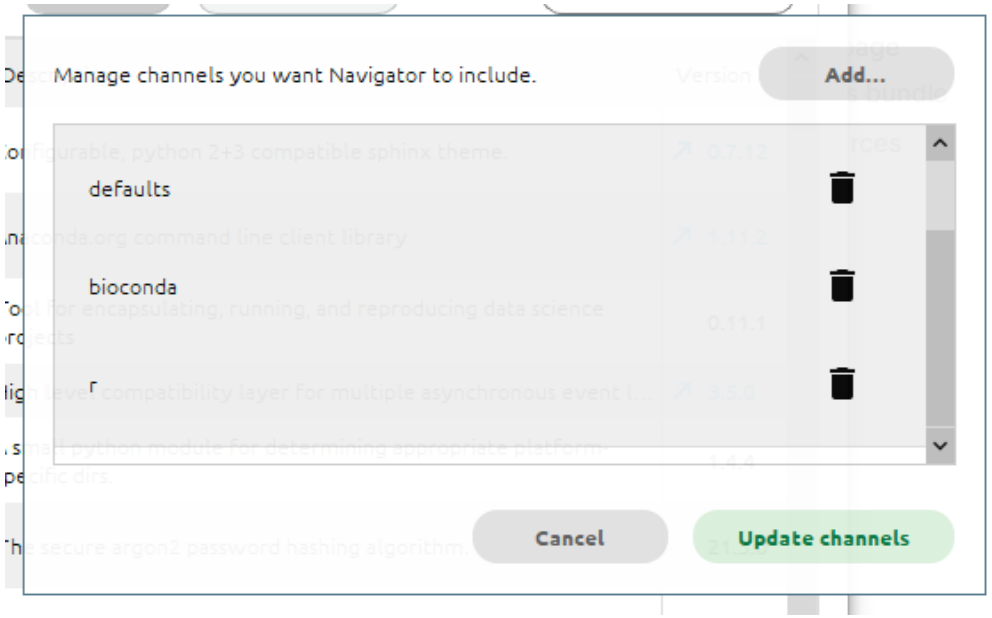
`Source` packages are source code only, not yet built for any specific platform.

`Noarch` packages are built to work on all platforms.

Conda environments

The best of both worlds(?)

https://docs.anaconda.com/free/anacondaorg/user-guide/packages/installing-packages/



Channels

Get additional packages when not supported by the standard repository

Python Package Index

Or use pip as you would do with venv

To install a non-conda package:

1. Activate the environment where you want to put the program:
   - In your terminal window, run `conda activate myenv`.
2. To use pip to install a program such as See, in your terminal window, run:

```
pip install see
```

3. To verify the package was installed, in your terminal window, run:

```
conda list
```

Conda environments

Does Conda solve our initial problem?

GPT4All	AQGT	STN
Python >= 3.8	Python = 3.7	Python = 3.7
	• pytorch > 2.0	• pytorch = 1.8 • Linux dependent?

Conda is nice but

- What if my Python and R libraries are dependent on C/C++ libraries that I can not modify?
- What if I need any other language than Python and R?
- What if that library is not available on my OS?

What is a container?

A container is a sandboxed process running on a host machine that is isolated from all other processes running on that host machine. Docker makes these capabilities approachable and easy to use. To summarize, a container:

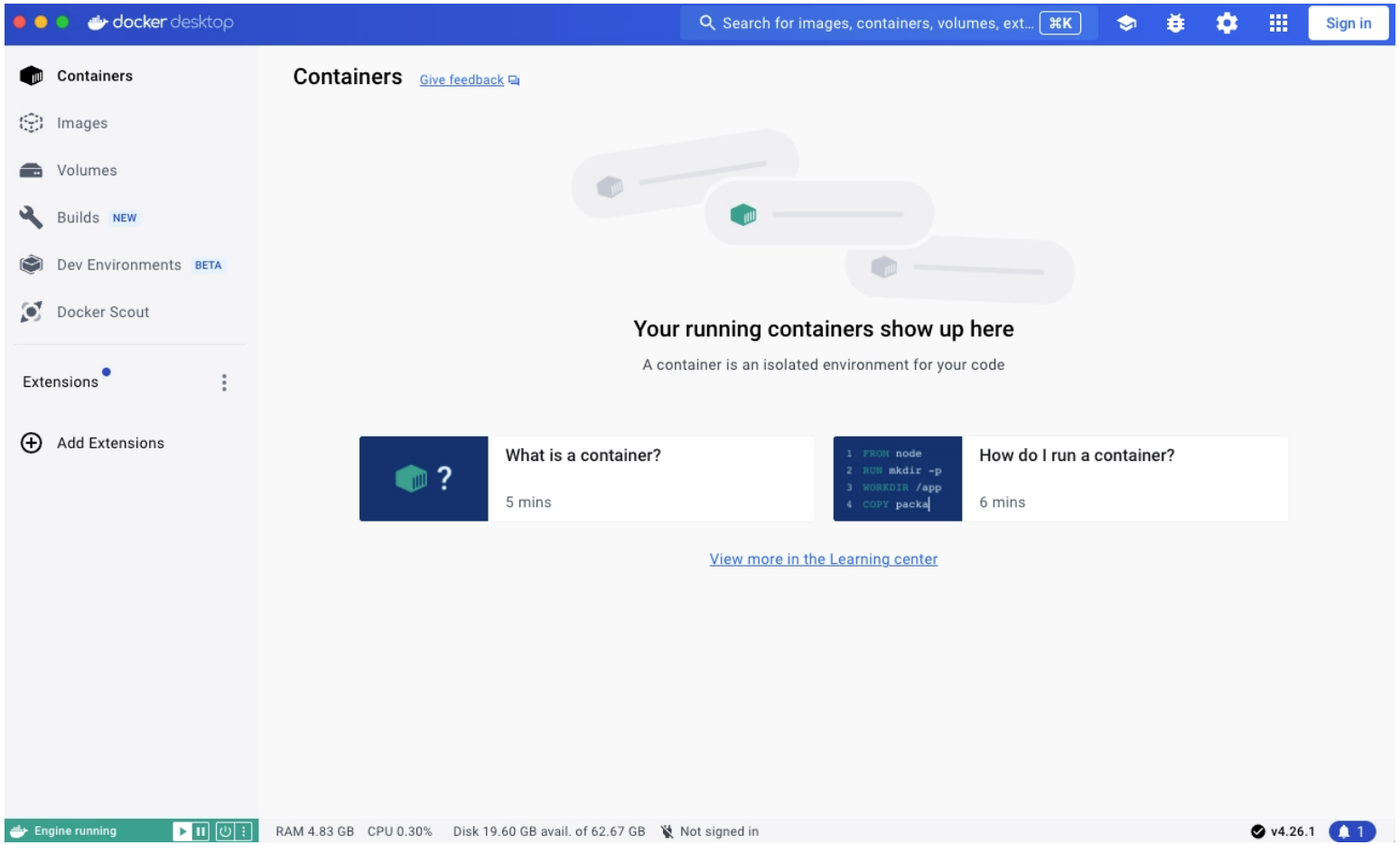
- Is a runnable instance of an image. You can create, start, stop, move, or delete a container.
- Can be run on local machines, virtual machines, or deployed to the cloud.
- Is portable (and can be run on any OS).
- Is isolated from other containers and runs its own software, binaries, configurations, etc.

Docker containers and chroot

https://docs.docker.com/get-started/

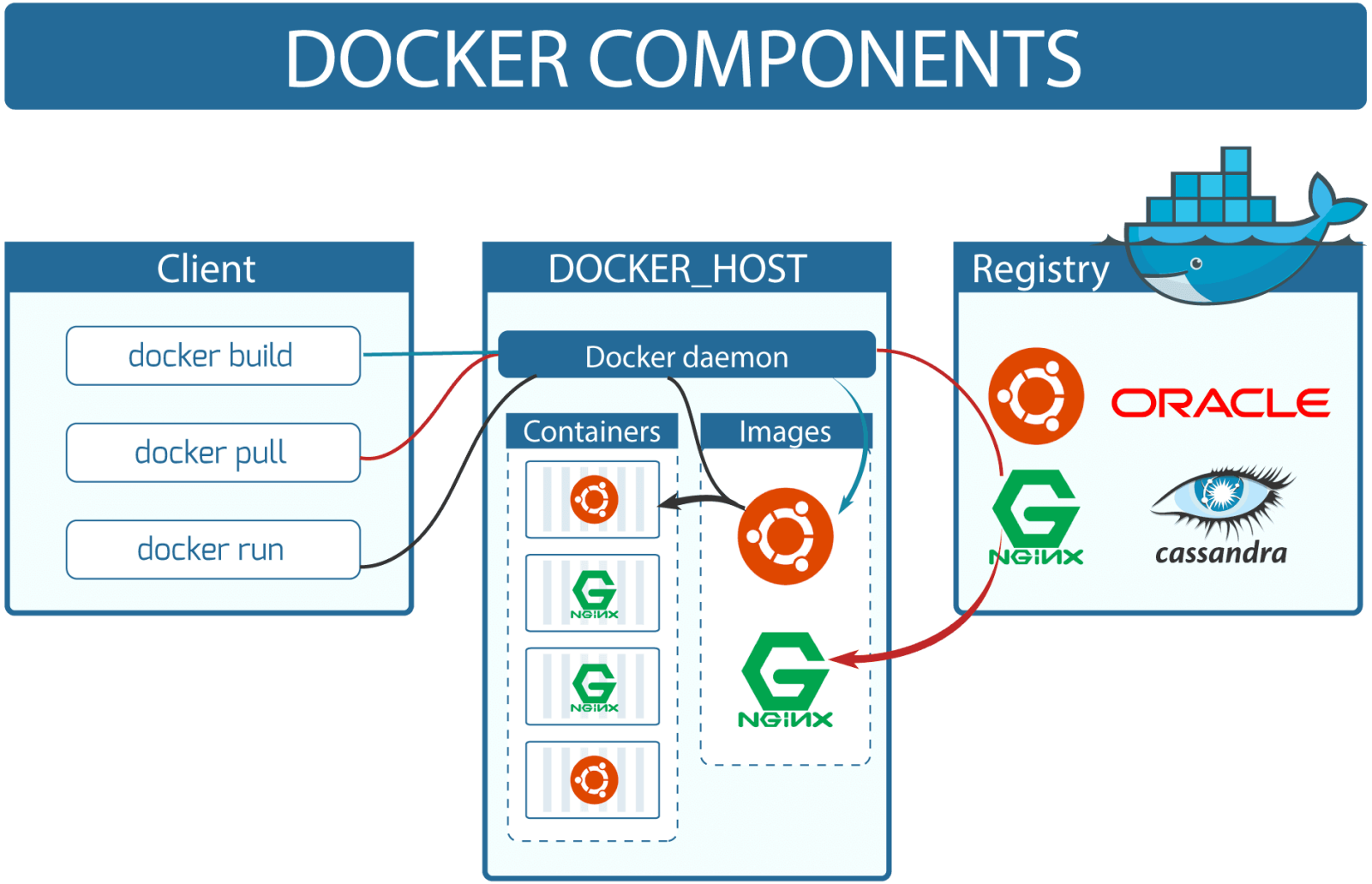
What is an image?

A running container uses an isolated filesystem. This isolated filesystem is provided by an image, and the image must contain everything needed to run an application - all dependencies, configurations, scripts, binaries, etc.



Docker containers and chroot

https://docs.docker.com/get-started/



Docker containers and chroot

https://docs.docker.com/get-started/

Dockerfile

Docker can build images automatically by reading the instructions from a Dockerfile. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. This page describes the commands you can use in a Dockerfile.

Docker containers and chroot

https://docs.docker.com/get-started/

Dockerfile

```
FROM python:3.10-slim-bullseye

# Prepare
RUN apt-get update ; \
        apt-get upgrade -y ; \
        apt-get install -y --no-install-recommends build-essential cmake g++
ffmpeg python3 python3-pip python3-dev
```

Docker containers and chroot

https://docs.docker.com/get-started/

```
RUN mkdir /gpt4all

# Build backend
COPY gpt4all-backend /gpt4all/gpt4all-backend
RUN mkdir /gpt4all/gpt4all-backend/build
WORKDIR /gpt4all/gpt4all-backend/build
RUN cmake ..
RUN cmake --build . --parallel
```

Docker containers and chroot

https://docs.docker.com/get-started/

```
# Confirm that libllmodel.* exists in gpt4all-backend/build.

# Build python bindings
RUN mkdir /gpt4all/gpt4all-bindings
COPY gpt4all-bindings/python /gpt4all/gpt4all-bindings/python
WORKDIR /gpt4all/gpt4all-bindings/python
RUN pip install --upgrade pip && pip install -e .
```

Docker containers and chroot

https://docs.docker.com/get-started/

```
# Prepare, copy and execute cli
RUN pip install --upgrade pip && pip install typer

RUN mkdir /cli
WORKDIR /cli
COPY gpt4all-bindings/cli/app.py .

ENTRYPOINT ["python3", "app.py"]
```

Docker containers and chroot

https://www.howtogeek.com/441534/how-to-use-the-chroot-command-on-linux/
https://domjudge.uu.nl/

Docker is very nice but

- It needs to run as a process, on top of your OS

What is chroot?

- You can use chroot to set up and run programs or interactive shells such as Bash in an encapsulated filesystem that is prevented from interacting with your regular filesystem. Everything within the chroot environment is penned in and contained.

When should you use a chroot?

- A chroot environment provides functionality similar to that of a virtual machine, but it is a lighter solution. The captive system doesn't need a hypervisor to be installed and configured, such as VirtualBox or Virtual Machine Manager.

It's important to note that chroot provides process isolation but is not as secure as more advanced containerisation technologies like Docker or virtualisation solutions.

Virtual environments

https://packaging.python.org/en/latest/guides/installing-using-pip-and-virtual-environments/#create-and-use-virtual-environments
https://rstudio.github.io/renv/

Now is your turn!

- Create a new folder called GPT4All
- Install python, if not already installed
- Open a terminal and create a new environment
  - Linux/Mac -> **python -m venv .venv**
  - Windows -> **py -m venv .venv**
- Activate your new environment
  - Linux/Mac -> **source .venv/bin/activate**
  - Windows -> **.venv\Scripts\activate**
- Install GPT4All **pip install gpt4all typer**
- Run **python app.py repl**
- Now you have your very own chatbot!

- Install RStudio, if not already installed
- Create a new project in a new folder called NetworkD3
- Open the console and install renv -> **install.packages("renv")**
- Activate your new environment -> **renv::init()**
- Install networkD3 -> **install.packages("networkD3")**
- Now you can see a folder called renv in your project that contains all the installed packages

Conda environments

The best of both worlds(?)

How to use the R programming language in Jupyter Notebook

https://docs.anaconda.com/free/navigator/tutorials/r-lang/

Docker containers and chroot

https://www.docker.com/101-tutorial/

Docker Desktop

Docker Desktop is a native application that delivers all of the Docker tools to your Mac or Windows Computer.

1. Open Docker Desktop
2. Type the following command in your terminal: **docker run -dp 80:80 docker/getting-started**
3. Open your browser to **http://localhost**

Utrecht University

Sharing science,
shaping tomorrow