# Introduction & Story

# Linus Torvalds

Former creator of Linux Kernel (at 22) in 1991 and still its ***“Benevolent Dictator for life”***

Creator of Git in 2005 for the development of Linux Kernel.

*“I'm an egotistical bastard, and I name all my projects after myself. First ‘Linux', now ‘git’”*

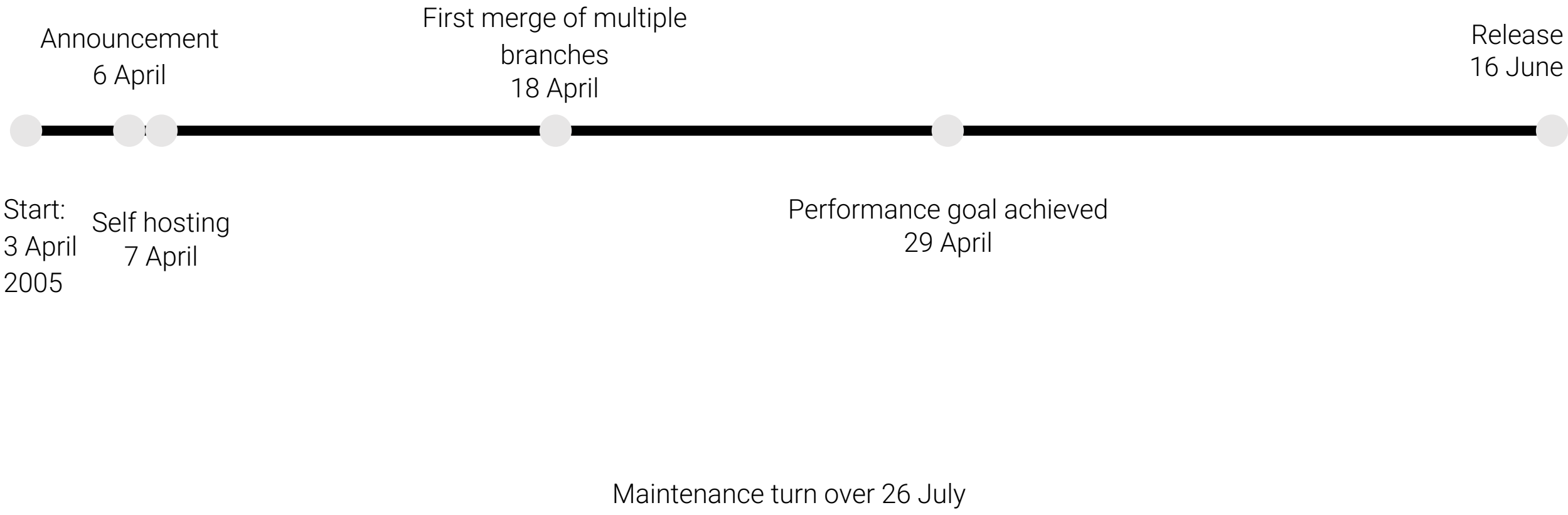
# Git, long story short

> Linux developers were using BitKeeper a proprietary source-control management system. The copyright holder withdrawn free use of it because he thought that they had reversed engineered the BitKeeper protocols.

> Linus Tovalds then benchmarked a lot of other tools but his expectations were so high that none fitted to his needs.

> They finally decided to create their own just after releasing the latest Linux kernel release.

# Project development timeline - Fast & Furious



# Who is using git?

> Basically any “tech” company (Facebook, Netflix, LinkedIn, Google etc.) and all companies that contains a dev team should use it.

> Git is also the go-to-guy when we talk about open-source projects. Nearly every open source project is hosted on git, mostly on github.

# What is Git?

# To keep it simple

A code versioning tool

A collaboration tool

And some branching policies 😍

To use it, two options:

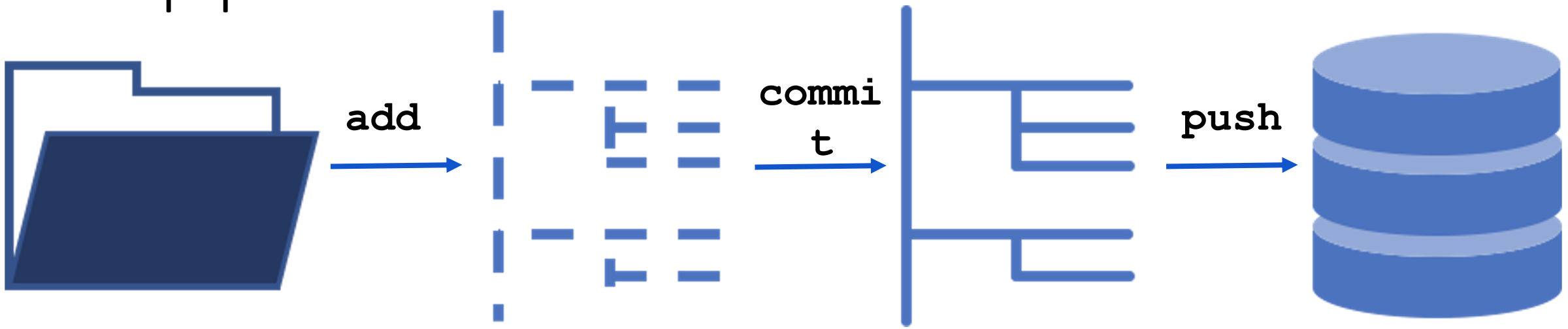
Using git commands from any Terminal

Using git softwares with windows and buttons

Because we all are IT-people, developers and proud to be so, we will only focus on the first point.

**It acts like a little database that will save with different stages all your modifications and enables you to work**

# THE pipeline




Working Directory

Index

Head

Remote Server

Local (on your computer)

Remote (on the git platform)

You can do multiple add for one commit.
And multiple commits for one push.

# Git platforms you may use

Azure DevOps

Bitbucket

GitHub

GitLab

# Key Principles

# Core concept: Branching

Git projects always have a master branch or equivalent.
This is the default branch that (in theory) contains the code with the best quality.

Usually (and depending on companies) projects also have some other branches with specific policies (dev, pre-prod, hotfix/ …)

Branching is an enabler for team-working.

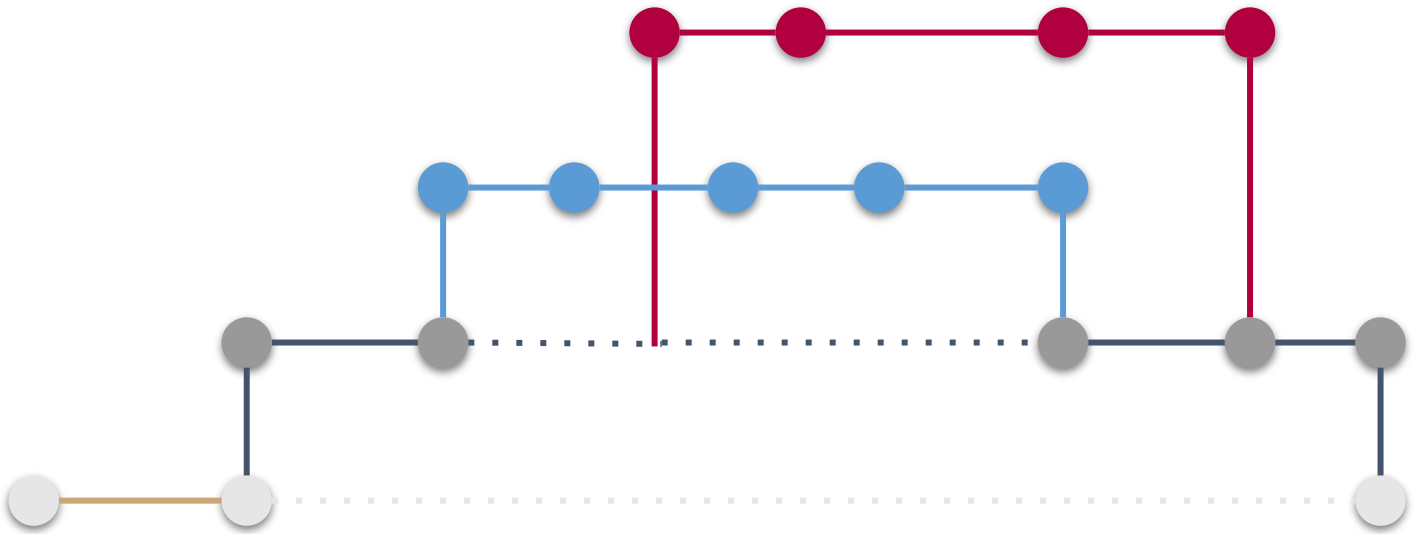
# A git scenario

feature/red

feature/green

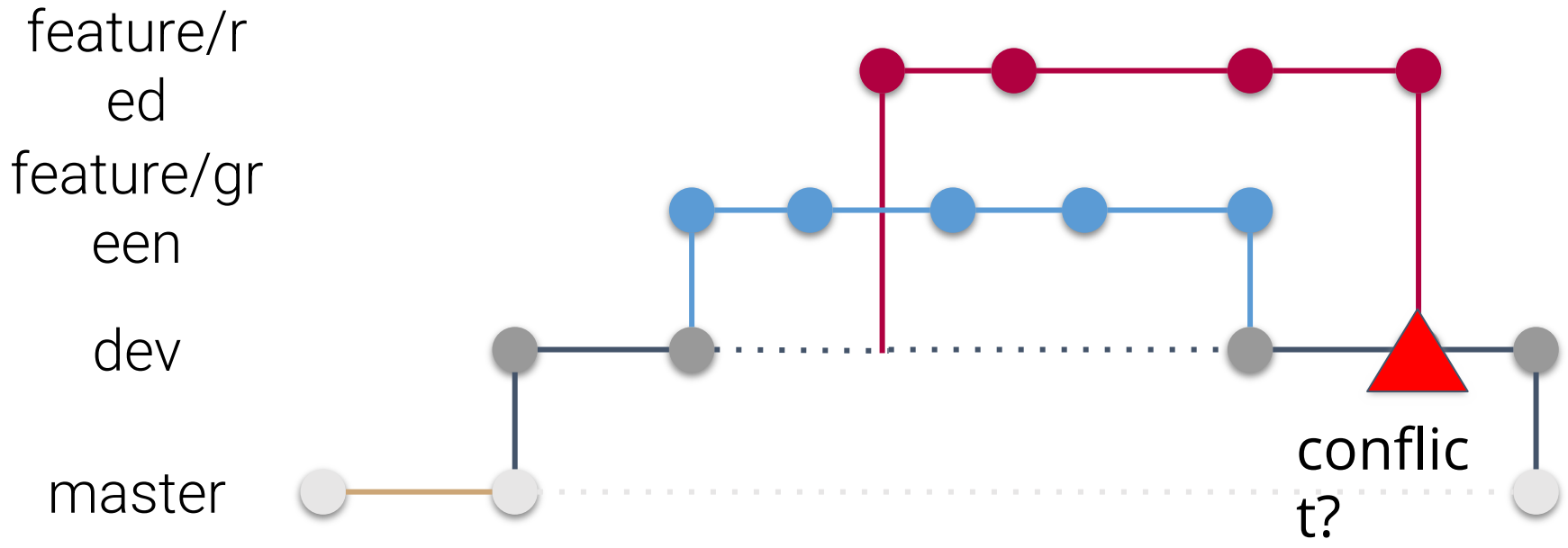
dev

master



```
git add, commit, push
git checkout -b dev
git add, commit, push
git checkout -b feature/green
git add, commit, push
git checkout -b feature/red
git add, commit, push
git add, commit, push
git add, commit, push
git add, commit, push
git merge feature/green
git add, commit, push
git add, commit, push
git merge feature/red
git add, commit, push
git merge dev
```

# Conflicts can happen on the way




```
git add, commit, push
git checkout -b dev
git add, commit, push
git checkout -b feature/green
git add, commit, push
git checkout -b feature/red
git add, commit, push
git add, commit, push
git add, commit, push
git add, commit, push
git merge feature/green
git add, commit, push
git add, commit, push
git merge feature/red
git add, commit, push
git merge dev
```

# Git commands

# Toolkit commands

```
$ git
```

+

```
init
```

Creates the .git directory in project

```
clone
```

Download a git repository from a git platform

```
status
```

Get informations about status of project's files

```
diff
```

Get differences on a file between last and current commit

# Updating commands

```
$ git
```

+

| | |
|---|---|
| `add` | Add a file and its modifications to index |
| `commit` | Create a commit (add files to HEAD) |
| `push` | Upload files to remote server |
| `revert` | Go back in time, go back to specified commit. |

# Branching commands

```
$ git
```

+

```
checkout
```

Navigate from current branch to specified one

```
branch
```

List all created branches, or create a new one if specified

# Merging commands

```
$ git
```

+

```
merge
```

Aggregates specified branch into the current one in one single commit.

```
rebase
```

Aggregates specified branch into the current one by replaying all the commits

# Pulling commands

```
$ git
```

+

```
pull
```

Update specified branch by downloading it from Remote

```
fetch
```

Update branches (and creates new ones) by downloading them from Remote.

# Git related files

# The .git directory

When you set up git into a project repository you automatically generate a .git directory

This directory will store all the modifications of your code into a bunch of .txt files

It will also store everything you need for the correspondence between your local branches and the ones that are on the remote server.

**Basically you don’t need to and won’t go into this directory**

# The .gitconfig file

This file is used to store all your configuration variables such as proxy settings, git credentials etc.

In fact you have two types of .gitconfig files, one is the global one, valid for your computer user, the other one is for each project, valid for the project where it's located.

Use them as you want, You can directly edit them or run commands to do so...

For example:

```
$ git config --local user.name=James.Bond
$ gti config --global http.proxy=10.1.1.1:8080
```

# The .gitignore file

This file has to be created if needed.
It’s a really simple text file where you can add all the patterns of filenames that should be ignored by git.

For example if we want to avoid to follow with git the .idea les generated by your IDE:

```
$ echo “*.idea” >> .gitignore
```

*.gitignore*

```
*.idea
```

Much more to come in hands-on session

https://github.com/ecaman/git_lesson

# Python Ecosystem For Data Science

## Virtual environments & Jupyter notebook

# Deep dive into a python package

# Huge amount of files

Example with pandas

For most of files, you don't really need to know what they are doing. We will quickly cover the basics.

20,133 commits | 11 branches | 106 releases | 1,580 contributors | BSD-3-Clause

Branch: master | New pull request | Find File | Clone or download

jorisvandenbossche and simonjayhawkins DOC: fix the editable install command (#28445) — Latest commit ad9fe5d 4 hours ago

| | | |
|---|---|---|
| .github | Add tidelift sponsor to FUNDING (#27823) | last month |
| LICENSES | Add reader for SPSS (.sav) files (#26537) | 3 months ago |
| asv_bench | PERF: Speed up Spearman calculation (#28151) | 7 days ago |
| ci | BLD: Add pyproject.toml (#28374) | 21 hours ago |
| conda.recipe | Run tests in conda build [ci skip] (#22190) | last year |
| doc | DOC: fix the editable install command (#28445) | 4 hours ago |
| pandas | CLN: Exception in core.dtypes (#28387) | 17 hours ago |
| scripts | Revert "TMP: pin openssl for doc build (#28404)" (#28405) | 23 hours ago |
| .binstar.yml | update conda recipe to make import only tests | 4 years ago |
| .gitattributes | CI: use versioneer, for PEP440 version strings #9518 | 4 years ago |
| .gitignore | BLD: Add pyproject.toml (#28374) | 21 hours ago |
| .pep8speaks.yml | Remove duplicate config .pep8speaks.yml (#26226) | 5 months ago |
| .pre-commit-config.yaml | DEV: Remove seed-isort-config hook (#28272) | 10 days ago |
| .travis.yml | [BLD] Add script that fails build if git tags do not exist (#27... | last month |
| AUTHORS.md | CLN: make license file machine readable (#16649) | 2 years ago |
| LICENSE | CLN: make license file machine readable (#16649) | 2 years ago |
| MANIFEST.in | BLD: Add pyproject.toml (#28374) | 21 hours ago |
| Makefile | BLD: Add pyproject.toml (#28374) | 21 hours ago |
| README.md | BLD: Add pyproject.toml (#28374) | 21 hours ago |
| RELEASE.md | BUG: replace of numeric by string / dtype coversion (GH157... | 3 years ago |
| azure-pipelines.yml | CI: Fix setting PATH in azure pipelines (#27787) | last month |
| codecov.yml | Admin: Disable codecov comments and re-enable results in ... | 3 months ago |
| environment.yml | Revert "TMP: pin openssl for doc build (#28404)" (#28405) | 23 hours ago |
| pyproject.toml | BLD: Add pyproject.toml (#28374) | 21 hours ago |
| release_stats.sh | add args to release_stats.sh | 4 years ago |
| requirements-dev.txt | DEP: Bump Cython to 0.29.13 (#28391) | 2 days ago |
| setup.cfg | STYLE: run pre-commit filters on the repo (#27915) | 15 days ago |
| setup.py | BLD: Add pyproject.toml (#28374) | 21 hours ago |
| test.bat | BLD Added --strict and -r sxX to test scripts (#18598) | 2 years ago |
| test.sh | BLD Added --strict and -r sxX to test scripts (#18598) | 2 years ago |
| test_fast.bat | CI: fix db usage in CI (#24529) | 9 months ago |
| test_fast.sh | CI: fix db usage in CI (#24529) | 9 months ago |
| test_rebuild.sh | TST: Use pytest | 3 years ago |
| versioneer.py | STYLE: Apply black formatting | 2 months ago |

# README.md

Md stands for markdown. This is an easy-to-write and easy-to-read format.
We used it in the git hands-on session.

Pretty straight forward, it’s name speaks for itself. READ-IT!

Usually it contains information about the package, how to install it, how to use it with quickstarts.

# Setup.py

> Setup.py is present in all python packages. Otherwise it could not be called a package.

> When you install a package, setup.py in launched and will install everything needed.

```
from setuptools import setup

setup(
   name='foo',
   version='1.0',
   description='A useful module',
   author='Man Foo',
   author_email='foomail@foo.com',
   packages=['foo'],  #same as name
   install_requires=['bar', 'greek'], #external packages as dependencies
   scripts=[
            'scripts/cool',
            'scripts/skype',
           ]
)
```

# requirements.txt

> Most packages contains a requirements.txt

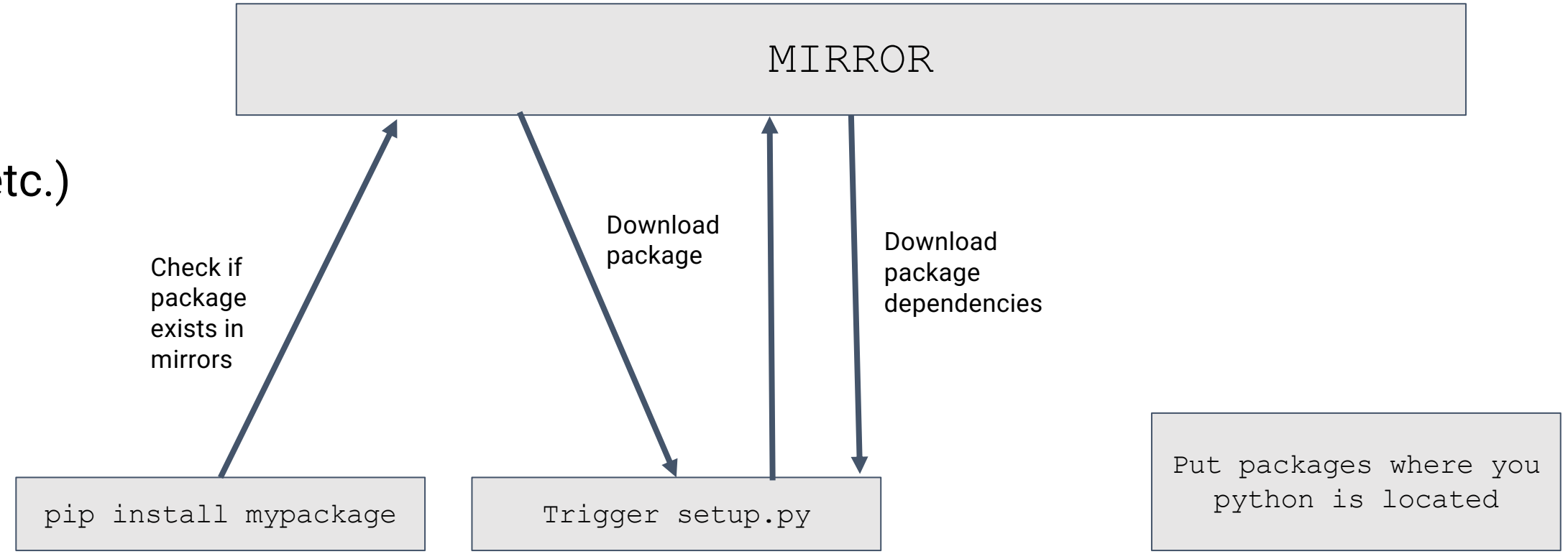
> Hopefully yours would also contain one.

> All the packages necessary to run the code are specified in this file, with their version needed

# What happens when I download a python package?

# Install pipeline

Mirrors (pypi, conda forge, etc.)




Loc al

# Why using virtual environments?

Scenarios

Awesome_Package now released a cool new feature that could help you with your current work

# Available starting at version 2.0!

But you installed the 1.4 ...

No big deal, I’ll update the package to the latest version

# Now, my million dollar project can’t run anymore.

Why did I used this damn awesome_deprecated_function in the first place to build this project?

# The solution - Virtual Environments

Multiple Virtual environments can be used with different python version at the same time

All the packages installed in a virtual env are only available inside this environment.

My advice is to create a virtual environment for each project you are working on.
(and maybe keep one as your “dev” or “sandbox” environment)

# Conda & virtualenv

# Two different tools - well, much more in reality...


# An eternal debate among python developers


But let’s try to be objective!

# Pros & cons


+ Official python tool
+ Simple to understand
+ Uses pip
+ Easy to install

- No env manager
- Can lead to anarchy
- Hard to configure if needed


+ Contains an env manager
+ Easy to configure
+ Works with pip & conda
+ Works on Windows easily

- Relies on anaconda
- Could be heavy
- Works on Windows easily

Seems that there is a new contender...

$ pipenv

# Conda install & setup

> Anaconda is heavy and will install so much things you will never use… Please don’t use anaconda distribution.

> Instead, if you want to benefit from the conda environments. Download and install miniconda.

> **Install**: https://docs.conda.io/en/latest/miniconda.html

> **Setup**: Edit everything you need (proxy settings, ssl certificates etc.) in the .condarc file located at the root of your user in your computer.

# virtualenv install & setup

It relies on pip, so you'll have to install pip first.

```
pip install virtualenv
```

**Setup**: No setup needed, you'll have to setup pip instead. Check pip.ini, pip.conf and such files if needed.

# Conda commands

## Manage environments

Create env

```
conda create --name myenv
```

```
conda activate
```

Go inside env (activate)

```
myenv
```

```
source activate
```

List all created envs

```
rconda env
```

```
list
```

## Configure conda (edit .condarc)

Set pip as package manager

```
conda config --set use_pip true
```

Set proxy

```
conda config --set http.proxy
```

```
0.0.0.0:8080
```

## Manage packages

List all packages installed

```
conda
```

```
list
```

Install a package

```
conda install
```

Update package version

```
conda update scikit-
```

```
learn
```

Install package with pip

```
pip install
```

```
boltons
```

```
conda install -c conda-forge --file requirements.txt
```

# virtualenv commands

## Manage environments

| | |
|---|---|
| Create env | *virtualenv* -p python3 new_p3_env |
| Go inside env (activate) | *source* ENV/bin/activate |
| List all created envs | *NOT AVAILABLE* |

## Manage packages

| | |
|---|---|
| List all packages installed | *pip* freeze |
| Install a package | *pip* install |
| Update package version | *pip* update scikit-learn |

*pip* install -r requirements.txt

# Jupyter notebooks

# What is Jupyter

It stands for Julia, python and R, the three languages used in Jupyter in the first place.

Jupyter notebook app is a client-server based app used to build notebooks documents (.ipynb). It’s based on ipython. The app can be run without internet connection locally or on a remote server and accessed through internet.

Notebooks are documents which contains both computer code and rich text elements (in Markdown or HTML format). Useful for tracking experiments and tutorials.

# How to use Jupyter

Install it using your favorite python package manager.

```
pip install notebook
conda install
notebook
```

```
pip install jupyter
conda install
jupyter
```

Oddly, installing notebook install 10 packages less
Than installing jupyter...

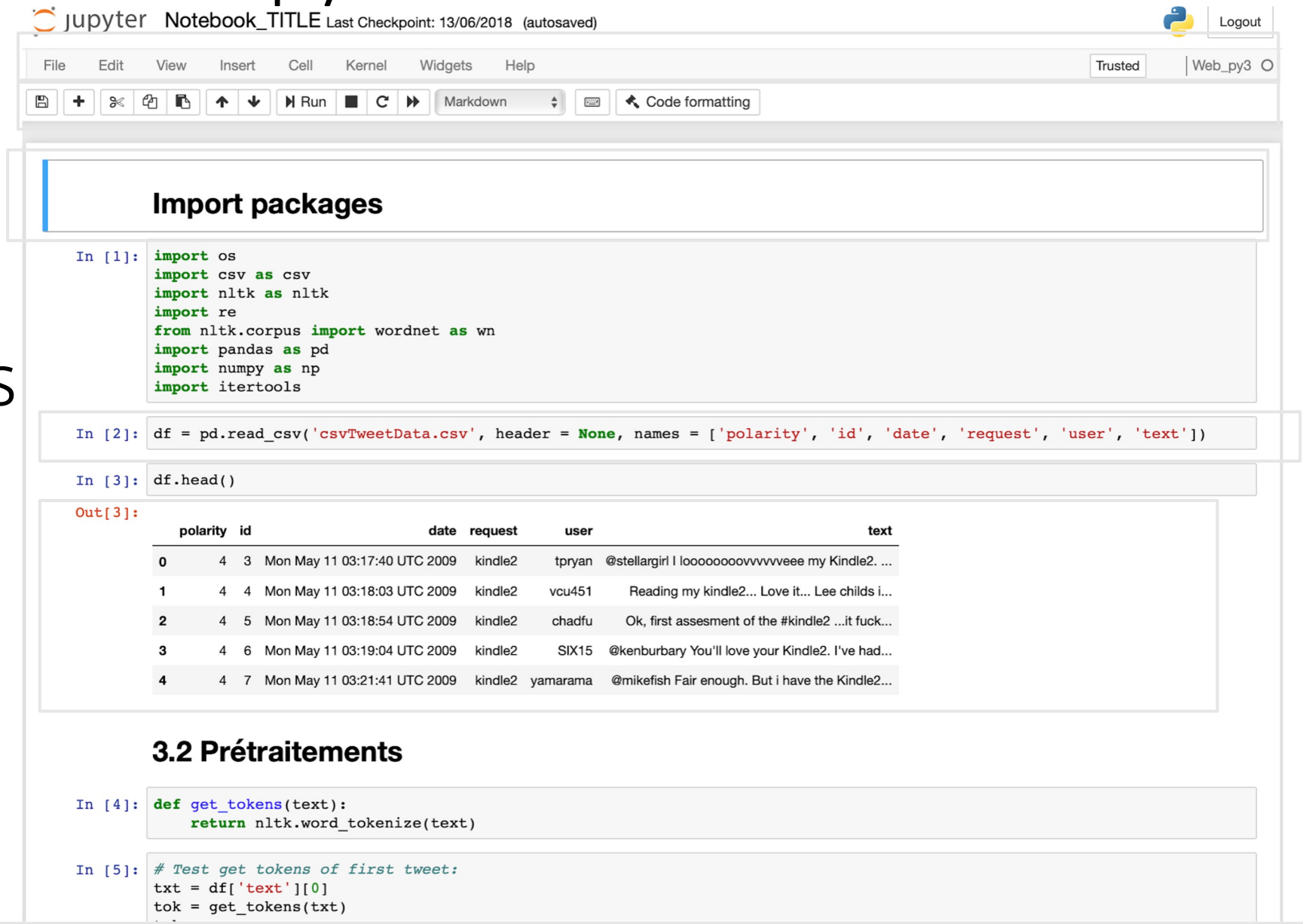
Start jupyter server with `jupyter notebook`

Copy and paste running address from your terminal to your browser.

Note that ip adress, ports, and some parameters could be edited while starting the server if needed.

# How to use Jupyter

LEARN THE SHORTCUTS !!!



# Pro-tips

You can use “magic functions” in jupyter. For example an “!” before a line will execute it as bash. “%%SQL” at the begining of a cell will execute the code as SQL commands, %%time at the begining of a cell will print out execution time.

You can change Jupyter Kernel (the executor) to make it point to another version of python, or another language (Java, R, C, bash) if you installed them.

You can install several extensions (using nbextensions) to change the appearence of your notebooks and add features to them. Try it out!

# What Jupyter is not!

> It’s not a IDE. if you want to write code in an easy and efficient way, prefer PyCharm, VSCode or equivalent. For example there is no auto-completion in Jupyter.

> It’s not something that you should deliver to your client. They can freak out by seeing code in a report...

> It’s not that easy… Watch out for hidden states and stuff. Restart Kernel if something is weird.

Watch this presentation to have interesting point of view: Why I don't like notebooks

# Hands-on

**It's your turn, try to:**

- Install conda or virtualenv (or both)
- Create a virtual environment with python 3.7
- Install jupyter
- Create a requirements.txt with some packages and specify versions of them
- Install them in your virtual environment