Towards Reproducible Jupyter Notebooks

Ludovic Courtès

Webinaire du Groupe de travail Notebook

4 octobre 2024

Inria

Jupyter = reproducible science

Jupyter = reproducible science?

In [1]:

```
%matplotlib inline
from matplotlib import pyplot as plt
from matplotlib import style
import random
x = random.sample(range(1, 5000), 1000)
num_bins = 100
n, bins, patches = plt.hist(x, num_bins, facecolor='green', alpha=0.5)

plt.title('Histogram Example')
plt.xlabel('Values')
plt.xlabel('Counts')
plt.show()
```

Daniel S. Katz
@danielskatz

Follow

When I see a jupyter notebook that starts with
pip install
I get a little scared

6:37 AM - 15 Jul 2019

Turn a Git repo into a collection of interactive notebooks

Have a repository full of Jupyter notebooks? With Binder, open those notebooks in an executable environment, making your code immediately reproducible by anyone, anywhere.

- **`environment.yml`** - Install a Python environment
- **`Pipfile`** and/or **`Pipfile.lock`** - Install a Python environment
- **`requirements.txt`** - Install a Python environment
- **`setup.py`** - Install Python packages
- **`Project.toml`** - Install a Julia environment
- **`REQUIRE`** - Install a Julia environment (legacy)
- **`install.R`** - Install an R/RStudio environment
- **`apt.txt`** - Install packages with apt-get
- **`DESCRIPTION`** - Install an R package
- **`manifest.xml`** - Install Stencila
- **`postBuild`** - Run code after installing the environment
- **`start`** - Run code before the user sessions starts
- **`runtime.txt`** - Specifying runtimes
- **`default.nix`** - the nix package manager
- **`Dockerfile`** - Advanced environments

Deploying JupyterHub with Kubernetes on OpenStack

Loïc Gouarin Follow
Oct 15, 2018 · 13 min read

https://blog.jupyter.org/how-to-deploy-jupyterhub-with-kubernetes-on-openstack-f8f6120d4b1

What To Expect

This guide will help you deploy and customize your own JupyterHub on a cloud. While doing this, you will gain valuable experience with:

- **A cloud provider** such as Google Cloud, Microsoft Azure, Amazon EC2, IBM Cloud…
- **Kubernetes** to manage resources on the cloud
- **Helm v3** to configure and control the packaged JupyterHub installation
- **JupyterHub** to give users access to a Jupyter computing environment
- **A terminal interface** on some operating system

It’s also possible you end up getting some experience with:

- **Docker** to build customized image for the users
- **Domain registration** to make the hub available at https://your-domain-name.com

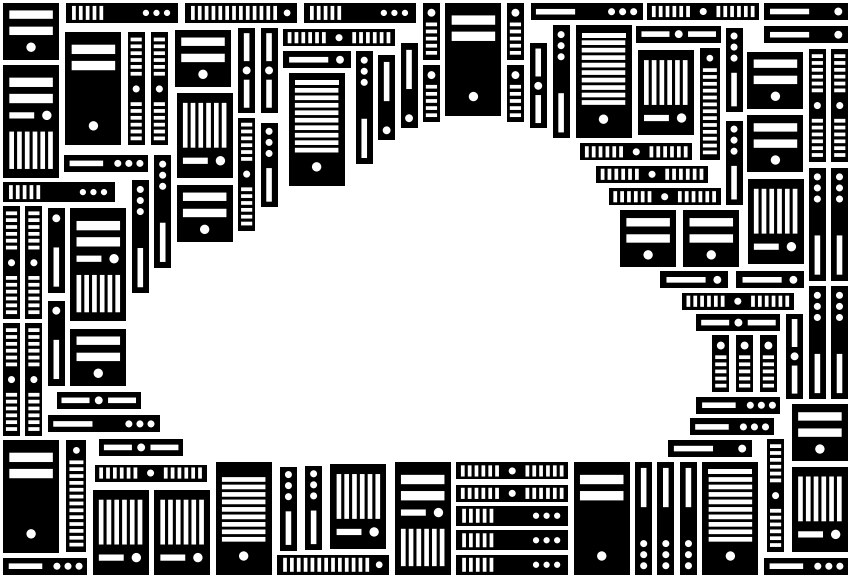
`https://zero-to-jupyterhub.readthedocs.io`

Notebook as a service, reproducibile research, & autonomy

Hinsen: **The four possibilities of reproducible scientific computations**

1. **inspect** data & source code
2. **run** code on computer of choice
3. **explore** behavior of the code
4. **verify** that published results correspond to code

https://blog.khinsen.net/posts/2020/11/20/the-four-possibilities-of-reproducible-scientific-computations/

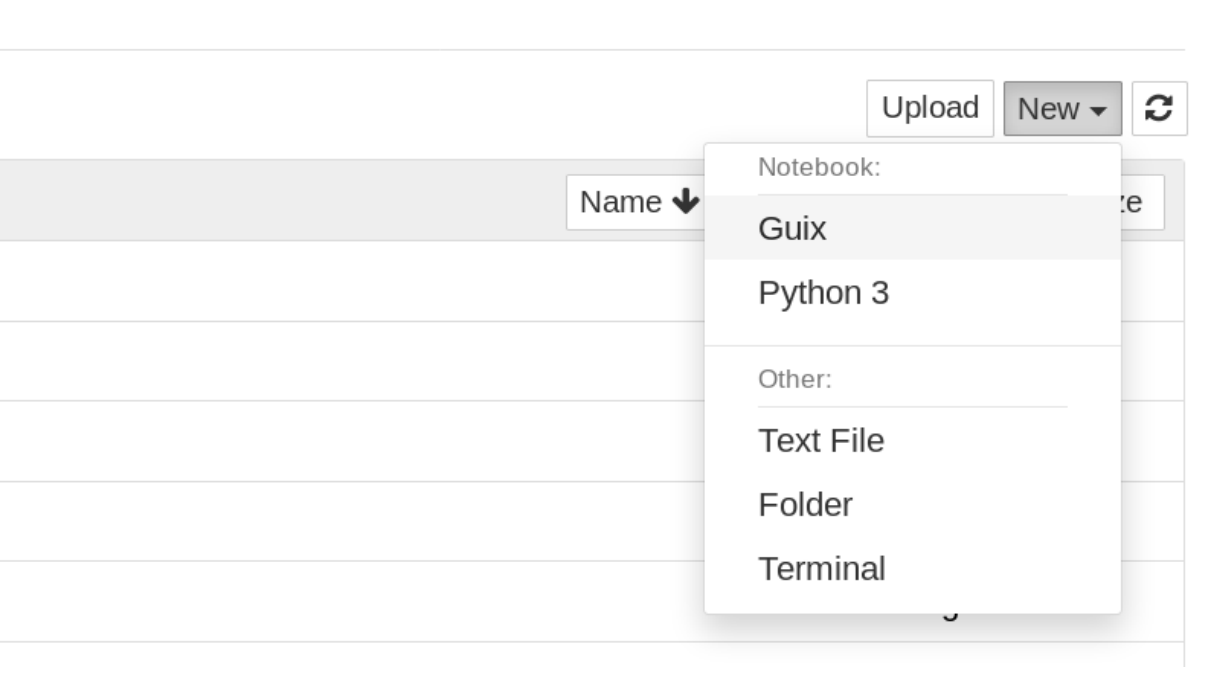


There is NO CLOUD, just other people's computers

What if notebooks were self-contained, “deployment-aware”?

```
$ guix shell \
      python python-numpy python-scipy \
      -- python3
```

https://hpc.guix.info/blog/2019/10/towards-reproducible-jupyter-notebooks


Upload
New
Notebook:
Guix
Python 3
Other:
Text File
Folder
Terminal
Name

In [4]:
```
;;guix environment matplotlib-env <- python-ipykernel python-ipywidgets python-matplotlib
```

Out[4]:

Preparing environment `matplotlib-env` with these packages:

- python-ipykernel 5.1.1
- python-ipywidgets 5.2.2
- python-matplotlib 3.1.1

Out[3]:

Running Python 3 kernel.

In [1]:
```
%matplotlib inline
from matplotlib import pyplot as plt
from matplotlib import style
import random
x = random.sample(range(1, 5000), 1000)
num_bins = 100
n, bins, patches = plt.hist(x, num_bins, facecolor='green', alpha=0.5)

plt.title('Histogram Example')
plt.xlabel('Values')
plt.xlabel('Counts')
plt.show()
```

In [2]:
```
;;guix search jupyter kernel
```

Out[2]:

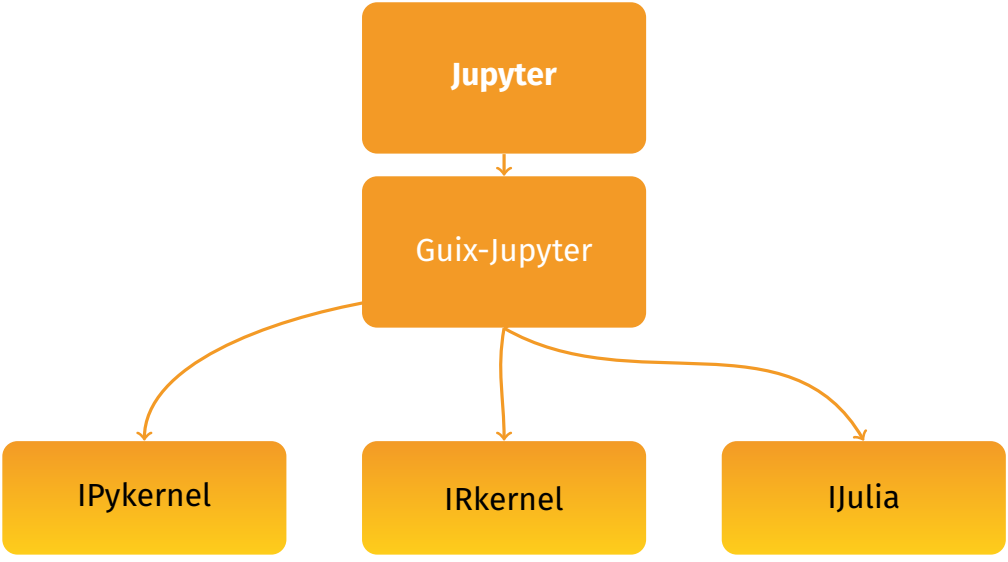
python-jupyter-kernel-test	0.3	Test Jupyter kernels
xeus	0.23.2	C++ implementation of the Jupyter Kernel protocol
python2-jupyter-client	5.2.4	Jupyter protocol implementation and client libraries
python-jupyter-kernel-mgmt	0.4.0	Discover, launch, and communicate with Jupyter kernels
python-jupyter-client	5.2.4	Jupyter protocol implementation and client libraries
guix-jupyter	0.1.0	Guix kernel for Jupyter
jupyter-guile-kernel	0.0.0-2.f25fb90	Guile kernel for the Jupyter Notebook
r-irkernel	1.1.1	Native R kernel for Jupyter
python-jupyter-protocol	0.1.1	Jupyter protocol implementation

First, jump back to Guix as it existed in January 2019:

In [1]:
```
;;guix pin 0791437f972caa7e48de91ad5cb150a614f617c2
```

Out[1]: Switched to these Guix channels:

```
guix 0791437f972caa7e48de91ad5cb150a614f617c2
```


Jupyter
Guix-Jupyter
IPykernel
IRkernel
IJulia

I've stored all the files at text files in a directory called articles and I wanted to grab all their names.

In [41]:
```
file_list=glob.glob('articles/*.txt')
```

The basic idea is to read each file, split it into sentences, and then process each sentence. The processing begins by splitting the sentence into words and removing punctuation. Then for each word that doesn't begin the sentence, I figure out if it is capitalized or not as part of the hunt for proper nouns. Then, I estimate whether the

In [6]:
```
import os
os.getcwd()
```

Out[6]:
```
'/home/jupyter'
```

In [7]:
```
os.getuid()
```

Out[7]:
```
1000
```

In [8]:
```
os.getpid()
```

Out[8]:
```
1
```

In [9]:
```
os.listdir('.')
```

Out[9]:
```
['.ipython']
```

In [6]:
```
;;guix environment R <- r r-irkernel
```

Out[6]:

Preparing environment R with these packages:

- r 3.6.1
- r-irkernel 1.0.2

Out[5]: Running R kernel.

In [8]:
```
;;guix download https://ftp.gnu.org/gnu/coreutils/coreutils-8.30.tar.xz e831b3a86091496cdba720411f9748de8
```

Out[8]: File `coreutils-8.30.tar.xz` from https://ftp.gnu.org/gnu/coreutils/coreutils-8.30.tar.xz is now available in environment `R`.

In [2]:
```
file.info('coreutils-8.30.tar.xz')
```

A data.frame: 1 × 10

	size	isdir	mode	mtime	ctime	atime	uid	gid	uname	grname
	<dbl>	<lgl>	<octmode>	<dttm>	<dttm>	<dttm>	<int>	<int>	<chr>	<chr>
coreutils-8.30.tar.xz	5359532	FALSE	444	1970-01-01 00:00:01	2019-10-09 20:42:28	1970-01-01 00:00:01	1000	1000	jupyter	users

Imposing a Memory Management Discipline on Software Deployment

Eelco Dolstra, Eelco Visser and Merijn de Jonge
Utrecht University, P.O. Box 80089,
3508 TB Utrecht, The Netherlands
{eelco, visser, mdejonge}@cs.uu.nl

Abstract

The deployment of software components frequently fails because dependencies on other components are not declared explicitly or are declared imprecisely. This results in an incomplete reproduction of the environment necessary for proper operation, or in interference between incompatible variants. In this paper we show that these deployment hazards are similar to pointer hazards in memory models of programming languages and can be countered by imposing a memory management discipline on software deployment.

cies between the components being deployed. Dependencies on other components are not declared explicitly, causing an incomplete reproduction of the environment necessary for proper operation of the components. Furthermore, dependency information that *is* declared, is often not precise enough, allowing incompatible variants of a component to be used, or causing interference between such variants.

In this paper, we present a simple and effective solution to such deployment problems. In Section 2 we analyse the problems that occur in software deployment. We then show

Wrap-up.

Open issues

- how can we improve the **user interface**?
- should deployment be **built into Jupyter**?
- what about **interoperability**?
- ...

Guix-Jupyter =

- **self-contained** notebooks
- automatic & **reproducible deployment**
- code runs in **isolated environment**

https://hpc.guix.info

ludovic.courtes@inria.fr | @civodul@toot.aquilenet.fr