Notebook

June 21, 2019

Contents

List of Figures

List of Tables

List of Codes

Figure 0.1: coderefinery logo

0.1 Publication ready scientific reports and presentations with Jupyter notebooks

This workshop has been developed within the CodeRefinery project. CodeRefinery is funded by Nordic e-Infrastructure Collaboration (NeIC) and aims at advancing FAIRness of Software management and development practices so that research groups can collaboratively develop, review, discuss, test, share and reuse their codes. CodeRefinery also deliver 3-day workshops within the Nordic countries, namely in Iceland, Denmark, Norway, Sweden, Finland and Estonia.

- Introduction to JupyterLab
- A first notebook to get familiar with JupyterLab
- JupyterLab git extension
- Convert your notebook with ipypublish
- Share and Publish Jupyter notebooks

1 1 - Introduction to JupyterLab

1.1 What is Jupyterlab?

JupyterLab is the next-generation web-based user interface for Project Jupyter. It is made of modular building blocks:

- File explorer
- Text editor
- Diagram generator
- jupyter notebooks supporting more than 50 different Jupyter kernels
- Terminals
- Outputs

with an easy integration with Jupyterlab extension such as jupyterlab-latex for live-editing of LaTeX documents, jupyterlab-git and jupyterlab-nbdime for git integration or jupyterlab-drawio for creating diagrams.

See JupyterLab slides from JupyterLab Github demo repository for more information about JupyterLab.

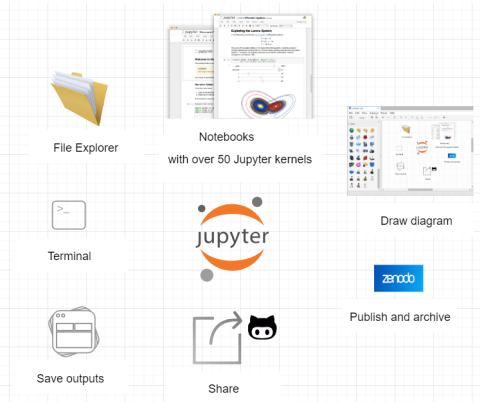

Figure 1.1: JupyterLab ecosystem

1.2 Get familiar with Jupyterlab

Make sure you have followed setup instructions and start jupyterlab:

You can also run the JupyterLab demo with Binder.

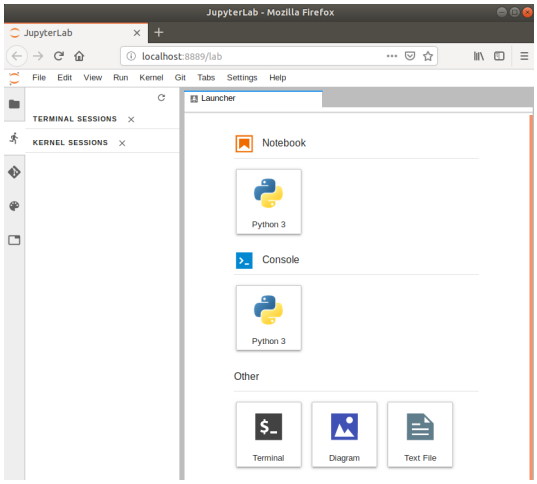

Figure 1.2: JupyterLab interface

2 Skagerrak Strait - 21 March 2013

Another Envisat image 2.1, acquired 31 January 2012, shows the Skagerrak Strait, which divides Norway and Denmark in this image. Clouds cover the North Sea and sweep down to the strait between Denmark (lower-right corner) and Norway (upper-centre) in this view.

```
from skimage import io
import matplotlib.pyplot as plt
%matplotlib inline

# https://images.eoportal.org/web/eoportal/images/snow-and-ice
url="https://images.eoportal.org/image/image_gallery?uuid=980fc01a-
    c2b5-43d1-a87c-1ce8c2d67763&groupId=163813&t=1363864717568"
image = io.imread(url)
plt.imshow(image)
plt.title("Credit: European Space Agency (ESA)")
plt.show()
```

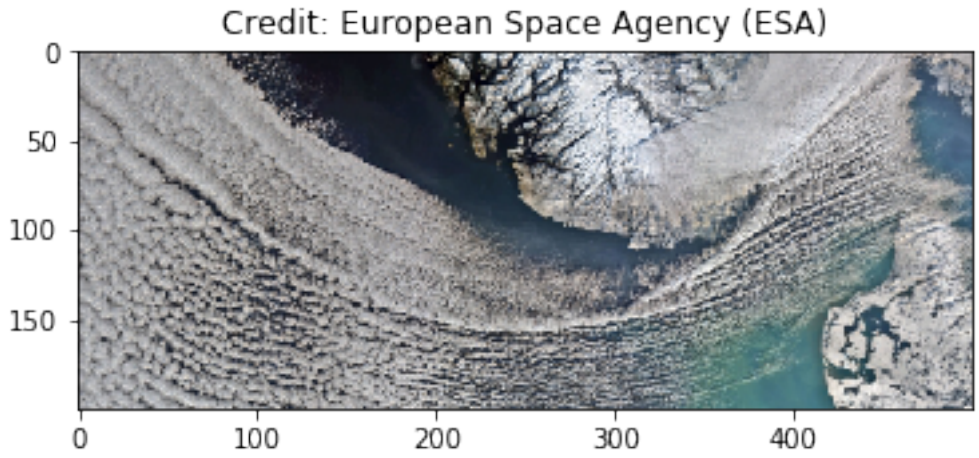

Figure 2.1: Envisat image, acquired 31 January 2012, showing the Skagerrak Strait

3 Using equation with LaTeX notation in a markdown cell

The well known Pythagorean theorem $x^2 + y^2 = z^2$ was proved to be invalid for other exponents. Meaning the next equation has no integer solutions:

$$x^n + y^n = z^n \tag{3.1}$$

The equation 3.1 can be referenced.

```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

# Data for plotting
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2 * np.pi * t)

fig, ax = plt.subplots()
ax.plot(t, s)

ax.set(xlabel='time (s)', ylabel='voltage (mV)',
       title='About as simple as it gets, folks')
ax.grid()

fig.savefig("test.png")
plt.show()
```

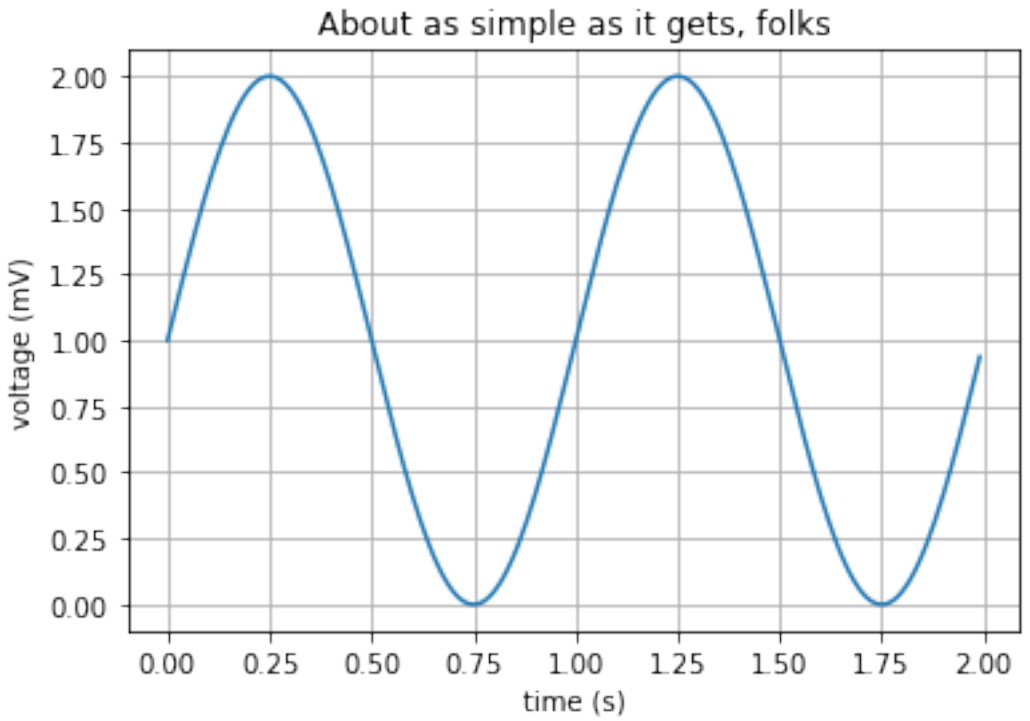

Figure 3.1: Simple plot with matplotlib

Example above is taken from https://matplotlib.org/gallery/lines_bars_and_markers/simple_plot.html.

You can also add your own notebooks and try them out. Please note that your favourite python packages may not be available in the provided environment.

Then you can reference your figure (fig. 2.1).

4 JupyterLab git extension

JupyterLab git extension is already installed in the Jupyter publish VM. If you want to install this extension to your local JupyterLab on your computer, follow installation instructions at https://github.com/jupyterlab/jupyterlab-git.

4.1 Check availability of JupyterLab git extension

A tab labeled "Git" (or with Git logo) as well as "Git panel" are available once JupyterLab git extension is properly installed, as shown on the figure below:

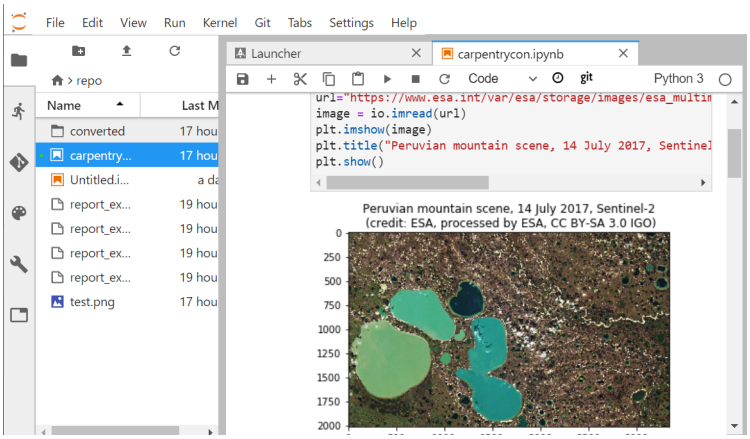

Figure 4.1: JupyterLab git

4.2 Git Terminal

Save and close your Jupyter notebook so you can start a terminal from the JupyterLab Launcher:

From this terminal, you can use `git` as a command line:

```
git status
```

Tips

Another way to open a Git Terminal is to use the “Git” Tab -> “Open Terminal” The copy/paste menu is “hidden” so to get it: - copy: select the text to copy with your mouse then SHIFT and right-click with your mouse so the copy menu will appear. - paste: SHIFT and right click to get the paste menu.

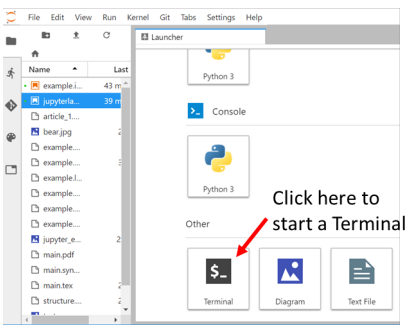

Figure 4.2: JupyterLab terminal

4.3 Git diff Jupyter notebook

We have installed nbdime and its JupyterLab extension so we can diff Jupyter notebooks:

Let's make a change in one of our jupyter notebook, for instance jupyter_publish-1.ipynb, by adding a new **markdown cell**:

```
# Add a new cell
```

Click on `git` to get the differences:

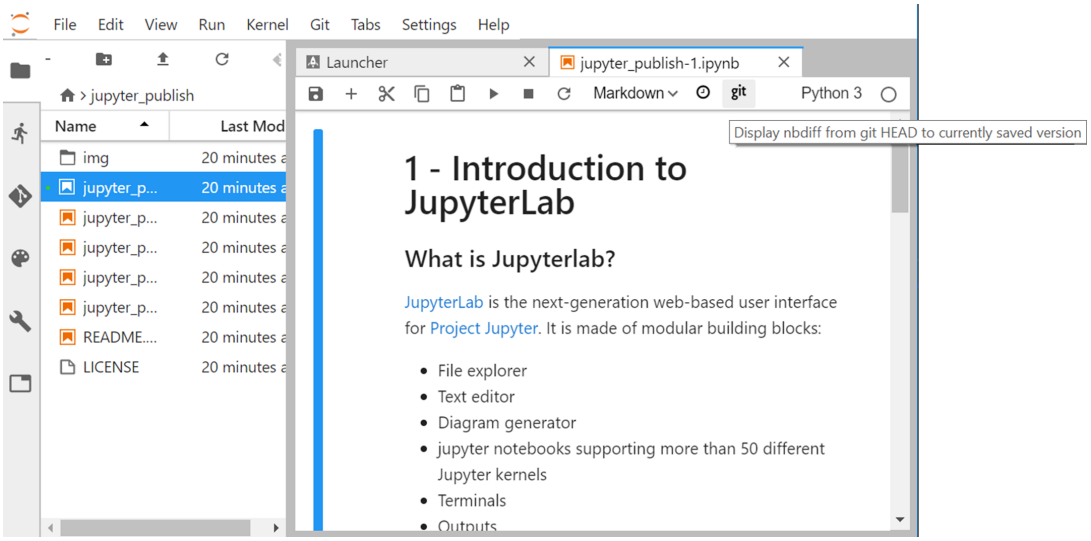

Figure 4.3: JupyterLab git diff

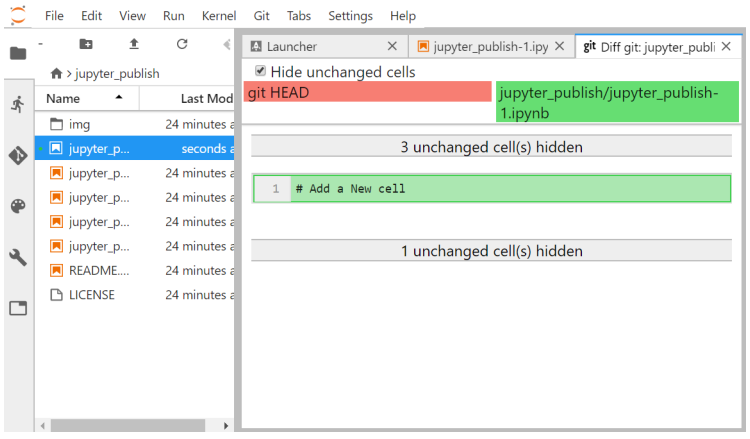

Figure 4.4: JupyterLab git diff with changes

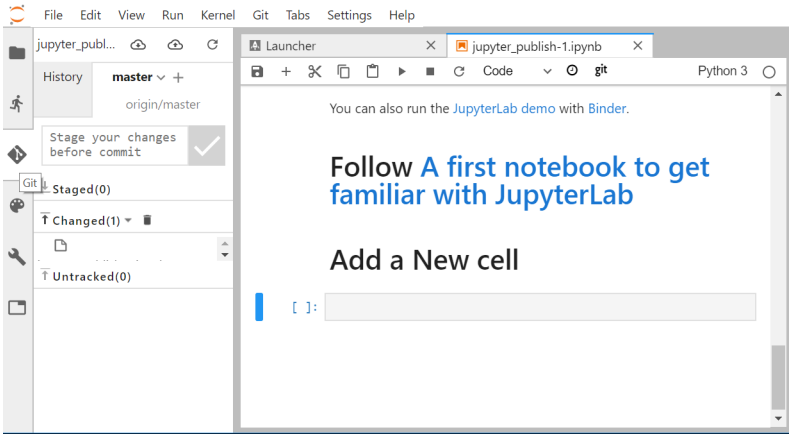

Figure 5.1: JupyterLab git staging

5 Staging and committing from JupyterLab

You can either use `git` Terminal from JupyterLab and run the git command line to manage your project or you can use the graphical user interface provided by JupyterLab:

6 Convert your notebook with ipypublish

We have been using JupyterLab to keep our research work and are now ready to write a scientific paper associated with our computing results. We will be using ipypublish to convert our notebook in various output formats. The main idea here is to have one source (that you can share in Github) that produces different kind of documents depending on the usage (oral presentation, paper publication, live demo, etc.).

6.0.1 Generate publication ready documents from our jupyter notebooks

Being able to create live documents with JupyterLab is great because we can create a reproducible research work, share it and publish it, but we usually need to communicate our research work (oral presentation, paper publication, live demo, etc.).

Let's go back to our jupyter notebook jupyter_publish-2.ipynb and let's create a nice LaTeX and pdf document from it.

Open a JupyterLab Terminal:

```
nbpublish -f latex_ipypublish_all -pdf jupyter_publish-2.ipynb
```

Execute this command and check `converted` directory. It should contain a list of files and in particular: - jupyter_publish-2.tex - jupyter_publish-2.pdf

`nbpublish` can be called directly from your jupyter notebook or from the jupyterLab Terminal.

6.0.2 Customize outputs with metadata

6.0.2.1 Ignore outputs It is sometimes convenient to inspect and modify metadata of a given cell. For instance, to ignore any cell for all outputs:

```
{
   "ipub": {
    "ignore": true
  }
}
```

Do not forget to commit changes to git if you want them to remain!

6.0.2.2 Metadata for Figures Each notebook cell can produce different kind of outputs and it is important to label your figure so you can reference them later in your notebook:

```
{
    "ipub": {
        "figure": {
            "caption": "A nice picture.",
            "label": "fig:example",
            "placement": "!bh"
        }
    }
}
```

Then you can reference your picture using latex notation:

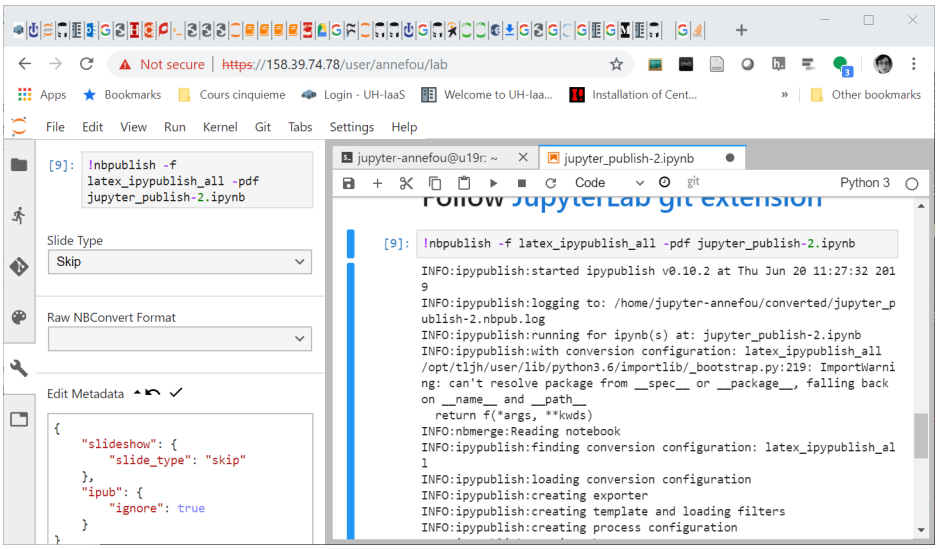

Figure 6.1: ipypublish ignore outputs

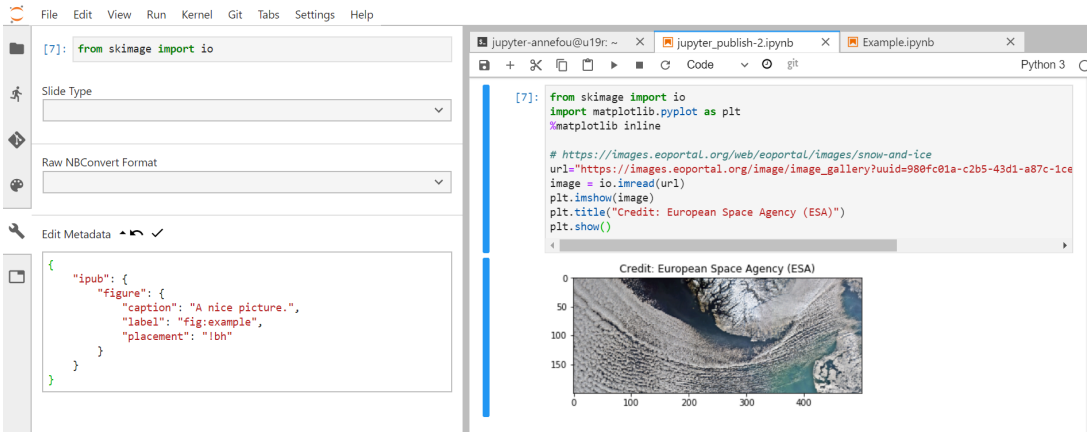

Figure 6.2: ipypublish figure

Figure 6.3: ipypublish slide

```
Then you can reference your figure (\cref{fig:example}).
```

Generate again the pdf and you should see your figure listed in the list of figures and the reference to your figure with the right numbering.

Metadata documentation: to get more information on ipypublish metadata, look here.

6.0.3 Generate slides from our jupyter notebooks

Still using jupyter_publish-2.ipynb, we can now generate a different output format, for instance for an oral presentation:

```
nbpublish -f slides_ipypublish_main jupyter_publish-2.ipynb
```

To view the resulting slides, go to the file explorer of JupyterLab and right click on the file to "Open in New Browser Tab".

As you can see many of the cell outputs do not appear by default in your slides. By default, an output that do not have metadata will not appear when producing slides!

Tips

If you are not running JupyterLab from a remote computer, you can also use `nbpresent`:

```
nbpresent converted/jupyter_publish-2.slides.html
```

6.1 More on ipypublish

Please not that this would not be possible to visualize the resulting slides either from binder or jupyterhub. You would first need to download the resulting file on your local computer.

There are several options (not only `latex_ipypublish_all` or `slides_ipypublish_main`) to generate and customize what you would like to see in your report.

Check the ipypublish documentation for more information. Please note that the ipypublish python package is still under development (beta version avaiable only).

6.1.1 Additional ipypublish example

In this short tutorial, all the functionalities of ipypublish are not demonstrated. However, feel free to browse the following example.

6.2 Add generated files into your Github repository

If you want to keep your work, do not forget to add your changes to your Github repository either using JupyterLab git extension and/or JupyterLab Git Terminal.

7 Share and publish your research work

To make publication ready scientific reports and presentations, we would need to share our research with Github with an objective to be as reproducible as possible.

In this section we will learn how to publish our research work using MyBinder and Zenodo.

Sharing your GitHub repository along with your jupyter notebooks and your publications is an important step for making your research reproducible. However, anyone willing to rerun your programs/notebooks need to get the same computational environment (python, LaTeX, additional python packages, etc.).

The next section (using Binder) will show you how to make your research "fully" reproducible, offering users the same computational environment as we used during this workshop and with very little extra efforts.

7.1 MyBinder reproducible environment

7.1.1 Important notice

This lesson has been taken from https://reproducible-science-curriculum.github.io/sharing-RR-Jupyter/ and is distributed under the Creative Commons Attribution license. The following is a human-readable summary of (and not a substitute for) the full legal text of the CC BY 4.0 license.

7.1.2 A short intro on Binder

Authors: Chris Holdgraf, M Pacer

Slideshow

7.2 Turn your github repository into a reproducible environment with mybinder

7.2.1 Preparing your github repository for Binder

We would like to publish all the codes, notebooks, reports, etc. in our repository with Binder. To be Binder-compliant, we need to add configurations files (one or more text files) that specify all the requirements for building your project's.

All these configuration files are placed in a directory called binder that needs to be in the root directory of your git repository.

Check `binder` directory in your repository. We have already prepared all the necessary files for running in Binder:

```
ls binder
```

You should see 3 files: - environment.yml
- apt.txt
- postBuild

7.2.1.1 Sharing our Python environment (environment.yml) This approach is recommended when all the additional packages/libraries you need are part of conda. Be aware that `conda` is a source package management system and is not only used for `python`. Many packages/libraries, independent of python/R are made available via conda, so the best is to first check online whether your package is already available via conda.

```
name: jupyter-publish

channels:
  - conda-forge
  - bioconda
  - defaults

dependencies:
  - python=3.7
  - numpy >=1.14
  - scikit-image
  - nbdime
  - ipypublish
  - nbconvert
  - pip:
      - jupyterlab_latex
      - jupyterlab-git
```

This file must be placed in the binder directory of your reprository on Github (e.g. it needs to be added to your repository). From a JupyterLab Terminal (or using the JupyterLab Git GUI.

7.2.1.2 Additional system packages Using environment.yml, we can run our jupyter notebooks but we cannot generate LaTeX and pdf documents as it is not available by default.

To share our computational environment, additional system packages (LaTeX, etc.) need to be installed.

When these packages are not available as `conda` packages, we install them with `apt-get install`.

- apt.txt : contains all the debian packages that should be installed for installing and compiling LaTeX documents. This file needs to be in `binder` directory too (same location as for `environment.yml`).

```
texlive-fonts-recommended
texlive-generic-recommended
latexmk
texlive-xetex
nano
vim
```

7.2.1.3 PostBuild Sometimes, additional commands need to be run after installing all system and cond packages. For instance, in our case to activate git and LaTeX jupyterLab extension. For this purpose, we created a file called `PostBuild` (also in `binder` directory).

```
#!/bin/bash

jupyter labextension install @jupyterlab/git
pip install --upgrade jupyterlab-git
jupyter serverextension enable --py jupyterlab_git
nbdime extensions --enable
```

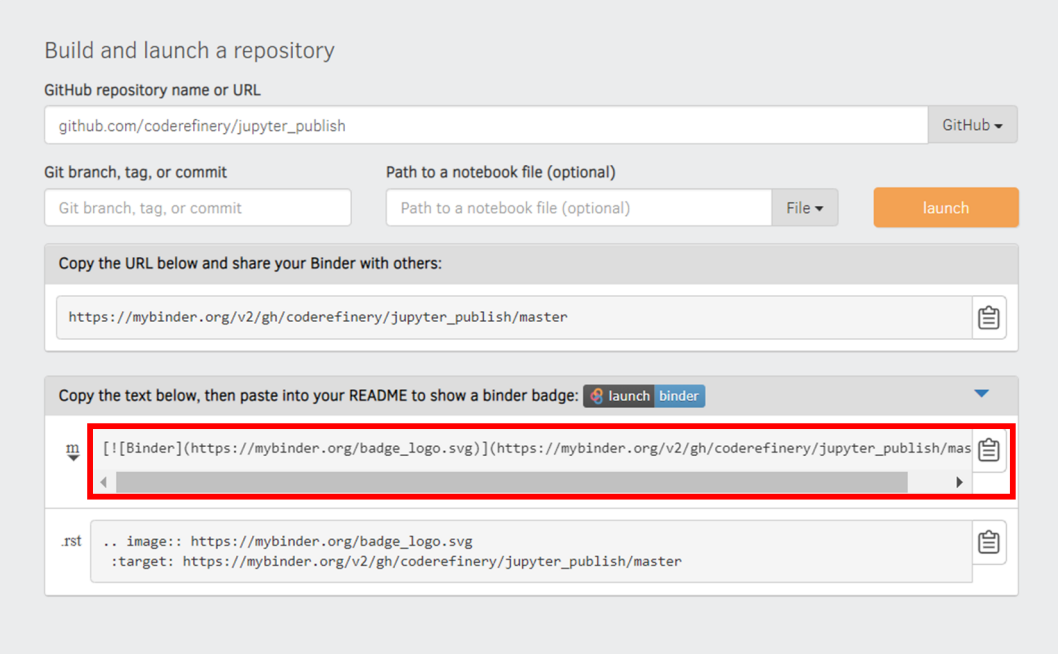

Figure 7.1: binder badge

```
jupyter labextension install jupyterlab-drawio

jupyter labextension install jupyterlab-jupytext
pip install jupyterlab_latex
jupyter labextension install @jupyterlab/latex
```

7.2.1.4 Note This file must be executable to be used with repo2docker.

To do this, run the following on Linux/Mac-OSX from a JupyterLab Terminal:

```
chmod +x postBuild
```

On Windows (to be done before you commit your file) and from a JupyterLab Terminal:

```
git update-index --chmod=+x postBuild
```

7.2.2 Launch your computational environment on Binder

- Start your complete computational environment on Binder
- Try to execute your notebook
- Check your notebook can run in your Binder environment

7.2.3 Get a shareable Binder Badge

- Create a shareable Binder link

Figure 7.2: display badge

To launch JupyterLab, you need to add `?urlpath=lab` after the branch name (here `master`) and to start from `index.ipynb`, add `index.ipynb` at the end:

https://mybinder.org/v2/gh/coderefinery/jupyter_publish/master?urlpath=lab/tree/index.ipynb

In your case, replace coderefinery by your github username.

- Update your README file in your github repository to display the shareable Binder badge with the proper github repository (yours and not the coderefinery one).

8 Make your github repository citable with Zenodo

8.1 Make your GitHub repository citable (DOI)

Your GitHub repository contains your scientific workflow, your programs/software, datasets (or links to your datasets) and jupyter dashboards so it is important to make the work you share on GitHub citable by archiving your GitHub repository to get a DOI. You may have a Data archive in your University or you may use the data archiving tool Zenodo.

8.1.1 Login to Zenodo

- Go to https://zenodo.org/ (or when testing only, you can use `https://sandbox.zenodo.org/`) and click on `Log in` (not `Sign up`)
- Choose `Log in with GitHub`
- Zenodo will redirect you back to GitHub and ask you to give Zenodo the permissions it needs. click `Authorize Application`:

Source: https://guides.github.com/activities/citable-code/zenodo-authorize.png

8.1.2 Get a DOI for your Github repository

- When sucessfully login to Zenodo, click on your username (top right) and select `GitHub`

Then - Select your repository `jupyter_publish` and flip the switch to `on` - Create a Release on Github - Then go to your GitHub repository and click on `settings` and select `Webhooks`

Your GitHub repository is now linked to Zenodo and you will automatically get a DOI:

8.1.3 Add your DOI to your GitHub repository

- Get your DOI badge on Zenodo and copy your DOI information (selection markdown)
- Go to your GitHub repository and edit your README file to add your DOI
- Create a new file CITATION in your GitHub repository and show how to cite your reprository with your DOI

9 Setup instructions

We strongly recommend using a virtual machine for the hands-on parts of the workshop.

A virtual machine, or VM, enables you to run a different operating system on your computer, from within your existing operating system. For these workshop we will use a VM image to run a fully configured Linux instance on participants' laptops. This will allow you to switch between your native/host operating system (Windows, Mac OS, Linux) and the guest Jupyterlab Linux operating system.

Using a virtual machine ensures that you will be running with the same configuration as the instructor. It also relieves you from having to setup your jupyterlab instance and installing prerequisites. With the Resbaz jupyterlab provided VMs you will also receive support from the instructors and your fellow attendees.

9.1 Getting ready for the workshop

Make sure the following steps are done before the workshop. These steps take some time, and require downloading large files, so please do not postpone this until you arrive. Contact us (organizers@swcarpentry.uio.no) before the workshop if you have any problems.

1. Install VirtualBox on your laptop

- Download and install the VirtualBox package from this link.
- You can install the latest version or at least version VirtualBox 5.2.

2. Install the needed VM on your laptop

You will want to download this VM before arriving at the workshop, because it is a large file and it will take too much time to download over the wifi network.

- Download Jupyter Publish Jupyterlab VM OVA file (10.GB)
- This VM contains:
  - Anaconda with python 3.7
  - Latex (basic version)
  - jupyter publish conda environment with Jupyterlab and other python packages we will be using for the workshop.

This section will include directions on how to start up, access, and use these virtual machines on your laptop.

9.1.1 Import the VM into VirtualBox

First, (if you have enough space on your laptop) make a backup copy of the downloaded .ova file(s). If something goes wrong you can always make a new copy.

Import the VM image into VirtualBox by either starting the downloaded .ova file directly, or by launching VirtualBox and navigating to File Import Appliance and opening the file.

This will display the Appliance Import Settings window. Click the Import button. It may then take several minutes for VirtualBox to import the VM. Once it is done, a new VM will appear in the left pane in the 'powered off' state. The VM is now installed.

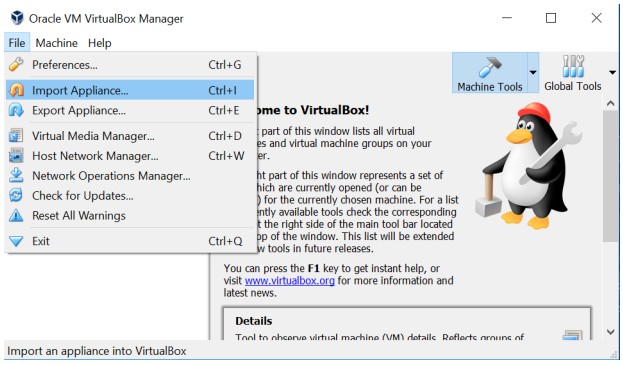

Figure 9.1: VirtualBox

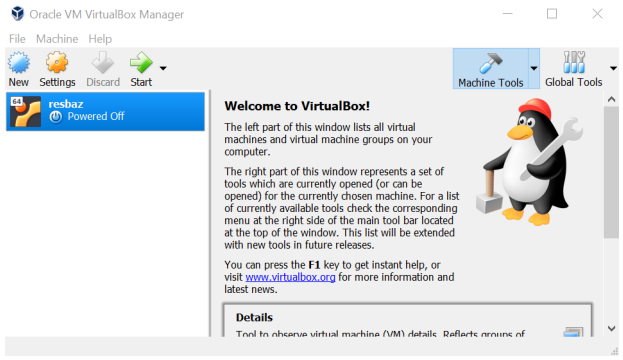

Figure 9.2: VirtualBox

9.1.2 Start VM

Double-click on the new VM in the left panel of VirtualBox. This starts the VM and displays two informational messages about regaining control of your keyboard and mouse from the VM. **Click OK for both**. This will log you in and show the ubuntu desktop.

The user password is 123456789

9.1.3 Start jupyterlab

- Open a terminal
- Enter `jupyter lab` to start jupyterlab. Your browser should open automatically with jupyterlab.

```
jupyter lab
```

You are done!

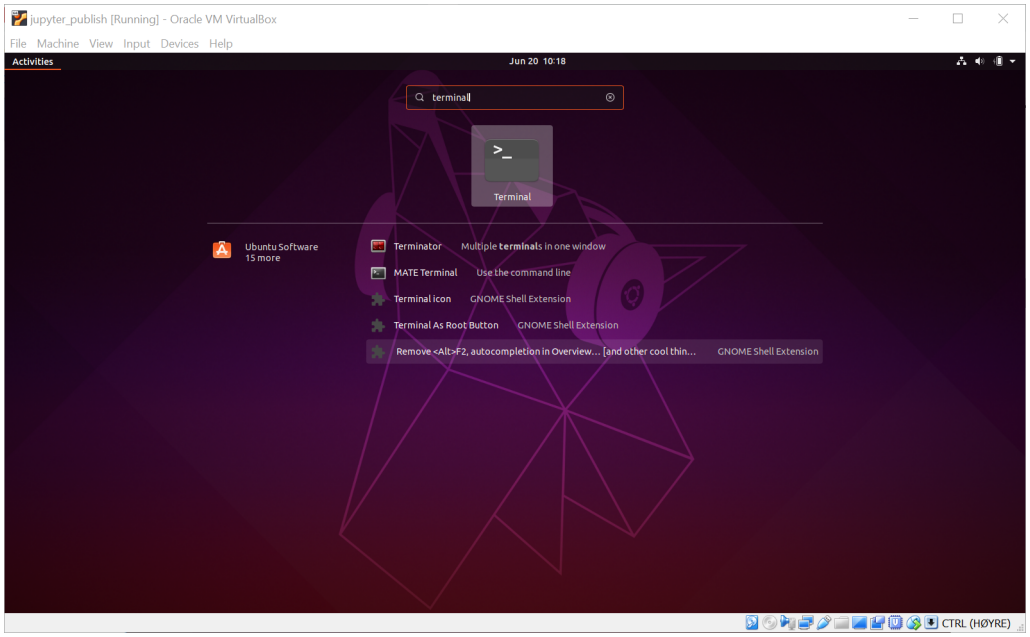

Figure 9.3: Open Terminal

9.2 Troubleshooting

VirtualBox does not show 32-bits images If VirtualBox is only showing 32-bit versions in the Version list make sure:

You have an x64 CPU installed. (Optimally, a 64-bit OS should also be installed to receive acceptable virtualization performance.) Hardware virtualization is enabled in the BIOS. (Your CPU must support it.) For Intel x64: VT-x (Intel Virtualization Technology) and VT-d are both enabled For AMD x64: AMD SVM (Secure Virtual Machine) is enabled Hyper-V (or any other form of bare-metal hypervisor) is not installed For more information follow this link.

VirtualBox 6.0 Installation problems on Fedora 29/28, CentOS/RHEL 7.5/6.10 Follow instructions given here

9.3 Github

9.3.1 Github Account

Note: If you already have a GitHub account you do NOT need to create a new one. Please skip this step.

Important notice

This setup instruction are from `https://reproducible-science-curriculum.github.io/sharing-RR-Jupyter/` and is distributed under the Creative Commons Attribution license. The following is a human-readable summary of (and not a substitute for) the full legal text of the CC BY 4.0 license.

To use GitHub one needs to register for a (free) account. To register for a GitHub account we need to:

- Open a web browser
- Navigate to github.com

We should see the the web page below.

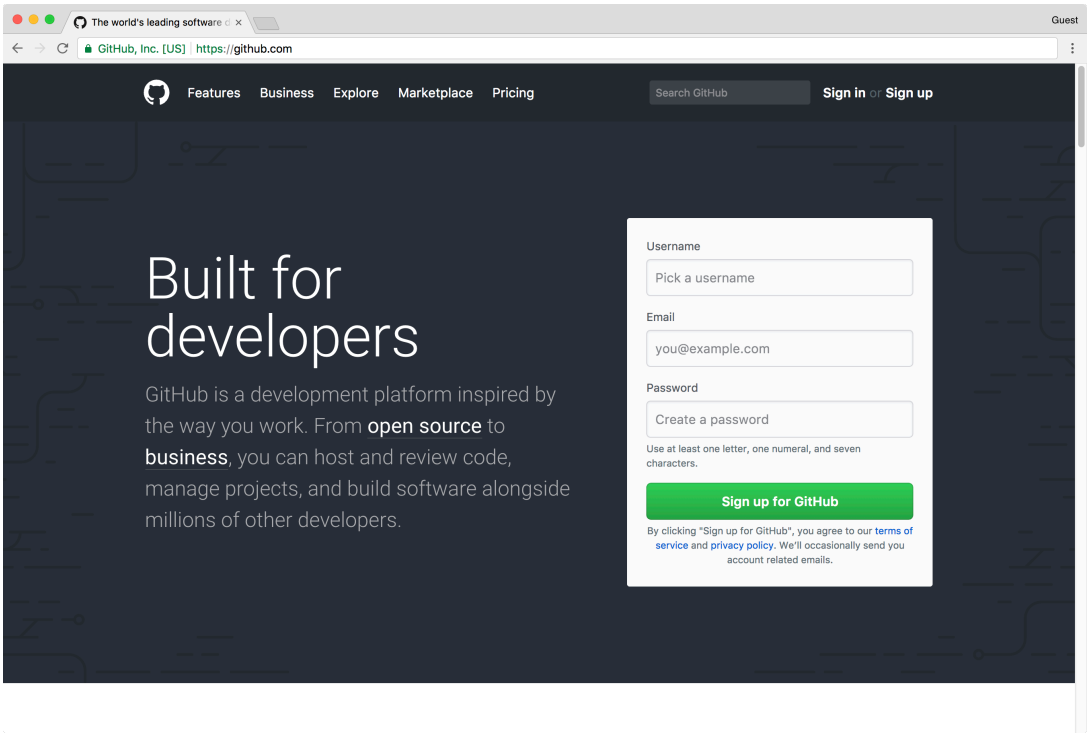

Figure 9.4: Github registration

9.3.2 To create a GitHub account

On the GitHub homepage enter:

- a username
- an email address
- a password

Click the green Sign up for GitHub button.

An example is shown in the screenshot below.

We are asked to confirm the email address that we used to sign. Please confirm the email address.

Select a plan

We have to choose a plan that we would like to use. We will use the personal (default) plan.

Click on *Continue*.

Completing our signup

We can select any options we would like (or none).

Click *Submit* to complete our `Github` account setup.

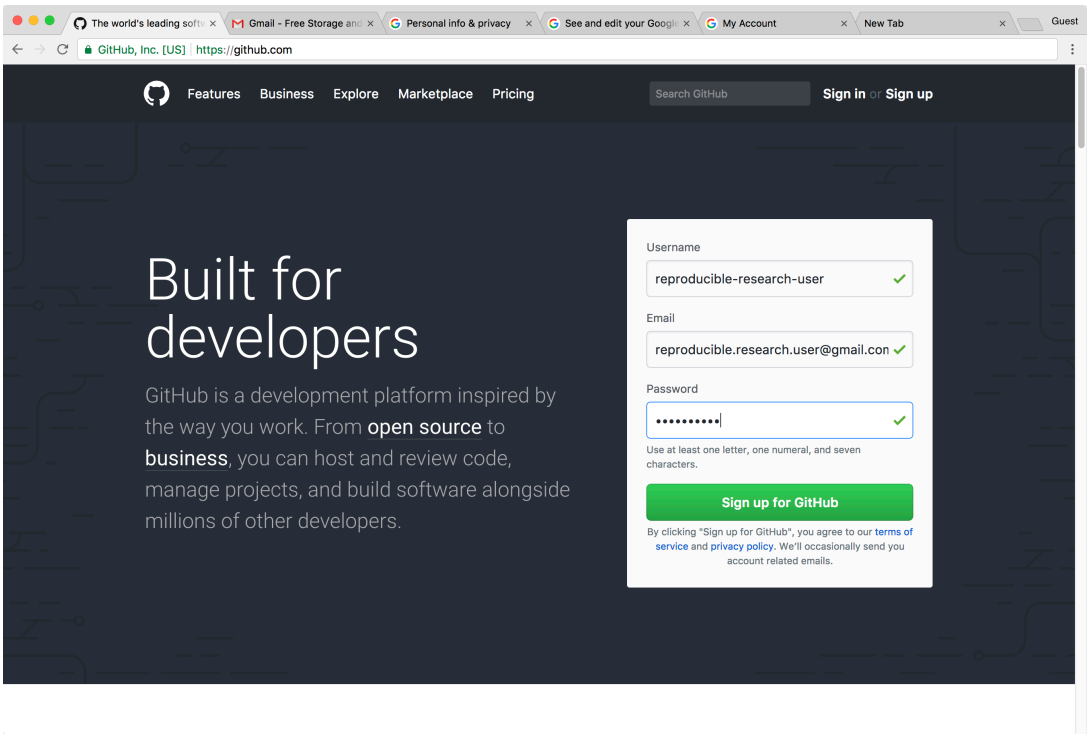

Figure 9.5: Github sign-up

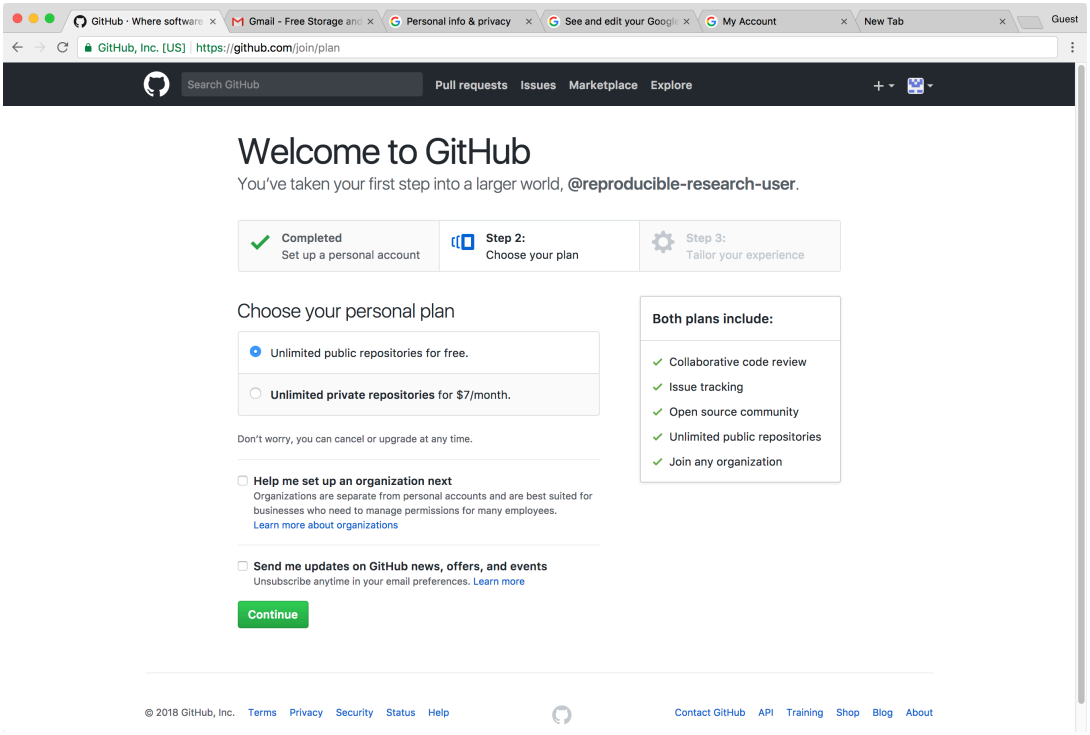

Figure 9.6: Github welcome

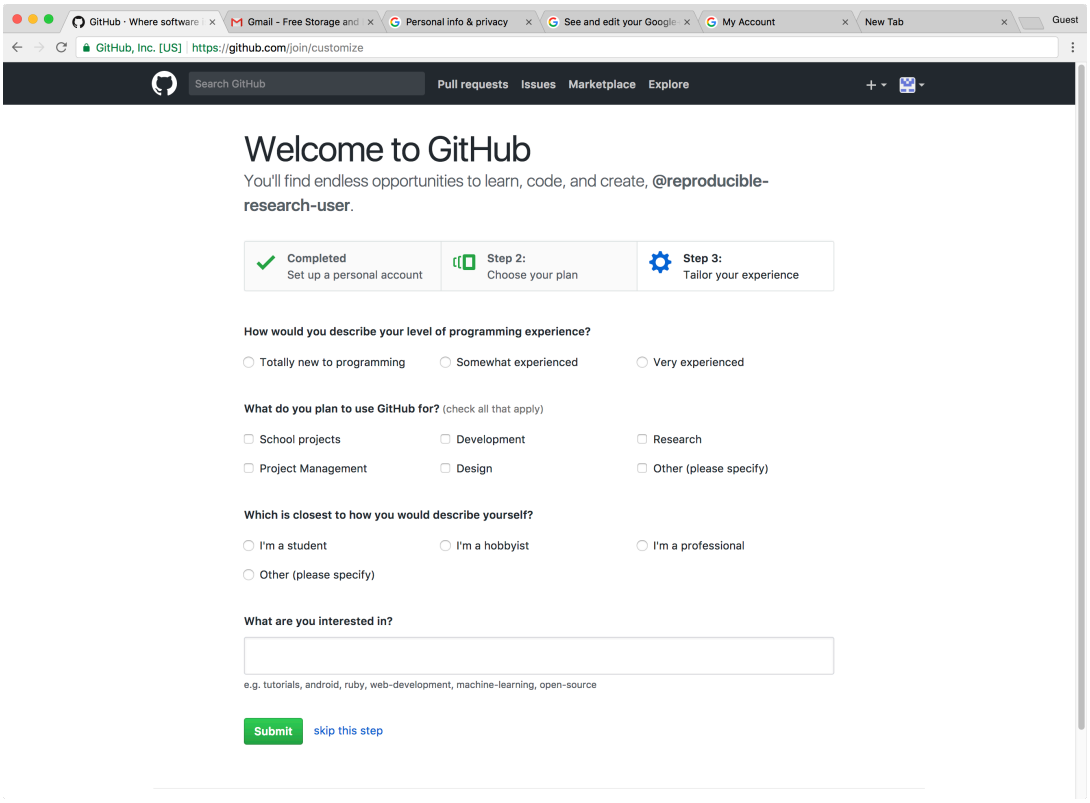

Figure 9.7: Github submit