Jupyter Notebooks

Jupyter Notebooks

Publishing Jupyter Notebooks

The Jupyter tool is a useful place to develop code and analyses in a notebook format. Hub users can easily share their notebooks with other users by *publishing* their notebooks as tools. A published Jupyter notebook enables other users to interact with the notebook, stepping through its cells and even changing them. But, when users run your published notebook, any changes they have made to it will not persist.

This set of instructions takes you through publishing a hub tool based on your existing Jupyter notebook. Here, we'll assume that the short name for your tool is *toolname* and that you are a registered, logged-in user. To develop the notebook tool, all you need is access to Jupyter. You'll navigate between your Jupyter home directory, the Jupyter terminal, and your tool's status page from the Tool Pipeline (The Tool Pipeline is typically found at yourhub.org/tools/pipeline but this may vary by Hub).

Jupyter version

When developing Jupyter notebooks or Jupyter based tools, you should use the most recent version of Jupyter deployed on your Hub.

To deploy a Jupyter notebook:

1. CREATE THE TOOL

To create the tool for your Jupyter notebook, navigate to Tools and click "Create a New Tool" on the upper left. Fill in the Create Tool page that the system displays:

1. Give your tool a brief name (no spaces or hyphens), a full title, and the at-a-glance description.
2. Select the repository hosting option. If you select the external hosting option be sure to supply the appropriate URL.
3. Select "Deploy as Jupyter notebook", and add your username in the development team.
4. The Access section enables you to restrict tool access to a specific hub Group, if you wish.
5. For other fields, you may accept the defaults, and submit.
6. Finally, flip the tool status to Registered, and click Apply Change.

2. REGISTER AS A Debian10 TOOL

For some hubs, you will need to submit a Hub ticket, indicating the short name of your tool, and

asking that it be registered as a Debian10 tool. This will ensure that the new tool uses current packages and kernels. Ask your Hub administrator if this applies to you.

3. CHECK OUT THE TOOL REPO

Your notebook tool's code repo should now have status Created and be ready to use. To do so, we must first check out the repo.

Open the Jupyter tool, navigate to your home notebooks directory, and open a terminal by selecting New, and then Terminal. Using the terminal, check out the newly created tool repo locally. In the section below, *toolname* is the brief name you gave your tool on the Create Tool page.

Using Subversion (svn)

To use a HUB hosted subversion repository, specify this command:

```
svn checkout https://yourhub.org/tools/toolname/svn/trunk toolname
```

Using Git

To use a HUB hosted git repostitory, specify this command:

```
git clone https://yourhub.org/tools/toolname/git/toolname toolname
```

To use an externally hosted git repository, specify this command:

```
git clone https://yourURL toolname
```

4. ADD NOTEBOOK CODE

It's now time to add the code that will run for your notebook. Back in the Jupyter tool file listing, you should see the *toolname* directory under your home notebooks directory. Into that directory, copy a working notebook (or develop one in place).

You can configure your notebook to access additional Python packages by loading an alternate kernel in the Jupyter notebook UI. To do so, consult the Kernel dropdown in the Jupyter interface. Different kernels may be available now on your Hub with additional packages. File a ticket or get in touch to let us know what packages you need.

You may need additional data files or code to run the notebook. The Hubzero team recommends putting the main notebook in the top level tool directory. Other files your notebook

needs (say, pythonfile.py) can be organized in subdirectories such as bin/. Then, you can load any Python files in your notebook as if they were modules. Your notebook will load the Python source data/pythonfile.py this way:

```
import bin.pythonfile
```

5. EDIT INVOKE SCRIPT

Finally, to tell the hub how to launch the notebook, you need to edit the invoke script that was automatically created at tool creation time. The invoke shell script is found in the *toolname*/middleware/ directory. To edit it, double-click on the invoke script in the Jupyter file listing, and the editor will launch.

In the invoke script you specify the filename of your Jupyter notebook, the version of Anaconda to use, and other parameters. Here we suppose that your notebook is called *your-jupyter-notebook-name.ipynb*.

For a Jupyter notebook using anaconda-X, your script will look like this :

```
#!/bin/sh

/usr/bin/invoke_app "$@" -t toolname \
                         -C "start_jupyter -T @tool your-jupyter-
notebook-name.ipynb" \
                         -r none \
                         -w headless \
                         -u anaconda-X
```

If your notebook needs additional modules, list them as -u module pairs. Be sure to add line continuation as needed (\).

For details on invoke script command line options, refer to the Hubzero invoke documentation.

6. TESTING

Next, you'll test that your working notebook starts properly as a Hub tool. When the notebook passes testing, you are ready to proceed.

7. COMMIT CHANGES

Once you have saved your invoke script and your notebook, check them in to the repository management software. You'll use subversion or git.

Using Subversion (svn)

From a Jupyter terminal, navigate to your tool's directory (get there as we did in step 3. above). First, add the notebook to svn (similarly, add any other needed files, using "svn add"):

```
svn add your-jupyter-notebook-name.ipynb
```

then, once all files have been added in this way, commit the changes:

```
svn commit -m "commit message"
```

The commit message should briefly indicate why the commit is being done or what the commit accomplishes. Commit messages serve as documentation for your work.

Using Git

From a Jupyter terminal, navigate to your tool's directory (get there as we did in step 3. above). First, add the notebook to git (similarly, add any other needed files, using "git add"):

```
git add your-jupyter-notebook-name.ipynb
```

then, once all files have been added in this way, commit the changes:

```
git commit -m "commit message"
```

The commit message should briefly indicate why the commit is being done or what the commit accomplishes. Commit messages serve as documentation for your work.

Once all files have added and committed the changes need to be pushed to the repository accessed by yourhub.

```
git push
```

To alert the administrator that your tool is ready for installation, you can now visit your tool's status page, either from the Tool Pipeline, or specifying a URL like this:

`https://yourhub.org/tools/`*`toolname`*`/status`

Here, click the link that reads, "My code is committed, working, and ready to be installed." If you have special instructions, caveats, compile steps, or other dependencies for your installation, enter them in the available text box now. The tool administrators will be alerted about your tool status and perform the installation along with any required steps you describe.

8. INSTALL

It's time to install the tool source. This action will depend on your access privileges; you may need the help of an administrator. On the hub, visit your tool's status page, either from the Tool Pipeline, or specifying a URL like this:

`https://yourhub.org/tools/`*`toolname`*`/status`

Here you can click the Install button and then on success message, flip the status to Installed and apply the change.

If the Install button is not available to you, this task will be executed by an administrator. You will receive a status email when it is complete.

9. TEST AND PUBLISH

To test your tool, go to the hub's Tool Pipeline and select your tool's link, or specify the tool URL directly:

`https://yourhub.org/tools/`*`toolname`*

In the status page, click the button to test run the tool. If the tool does not display or otherwise fails your test, there is still work to do. Revisit your development steps, starting with the TEST section above.

If the notebook test is successful, and it displays and functions as expected, you are almost done! Return to the tool status page. There you can indicate to administrators that you Approve the tool for publication.

You will receive a status change email when the tool has transitioned to Published. When you receive word that your tool is Published, you should verify again that it works as expected.

That should do it--your Jupyter notebook is now a published tool available to other Hub users. If

you have questions, concerns, or run into a snag, please email the Hub administrator. Include any error messages you see, and we'll give you a hand.

10. MAKING CHANGES

To make changes to a published notebook, you must only revisit some of the steps outlined above.

To make edits to the tool:

- Change your notebook code as necessary, revisiting the TEST and COMMIT CHANGES steps above when complete.
- INSTALL your changed code as above
- TEST AND PUBLISH the notebook tool as above

Each time you make changes, be sure to test the notebook and confirm that it works properly.