

Containers in HPC

Azat Khuziyakhmetov

GWGD

Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen

02.02.2022

GöHPCoffee



Overview

Containers

- Introduction
- Docker vs. Singularity/Apptainer

Singularity

- Architecture
- Use cases

Jupyterhub

- Goals
- Scheme

Conclusion

Containers

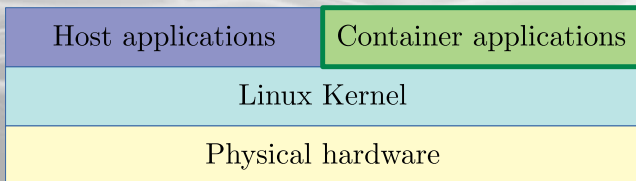
Why containers?

Management: easy to deploy software in HPC

Environment: users can install a software stack they want

Security: containers can be isolated from host OS

Lightweight: compared to other virtualization techniques



Container comparison

Why Singularity¹ is more preferable in HPC than Docker²?

	Docker	Singularity
Rooted daemon in Host OS	Yes	No
Root in the container	Yes	No
Host OS filesystems	Restricted	Mounted
Network	NS & NAT	no isolation

Docker needs additional steps to ensure security such as *user namespace*.

Without extra security steps, running user defined containers is not safe.

¹Singularity/Apptainer. URL: <https://apptainer.org>.

²Docker. URL: <https://www.docker.com>.



Architecture

The architecture of Singularity allows containers to be executed as if they were native programs or scripts on a host system.³

Important features for HPC

- Since v3 is written in GO (i.e. compilation and dependency resolving)
- Can be installed in shared FS, containers run on local FS
- Minimum isolation, network and devices are accessible in containers
- Processes run as users without ns (batch systems can account them)

Security

- Only root from Host OS can be root in a container
- Container FS mounted with `nosuid`
- Processes are executed with `PR_SET_NO_NEW_PRIVS`

³*Singularity architecture*. URL: https://www.sylabs.io/guides/3.0/admin-guide/admin_quickstart.html.



Use cases

Running applications in Singularity can be easier than installing it in HPC

Tensorflow Containers with matching versions of CUDA driver, cuDNN, Tensorflow and native support of GPU

Docker Docker has a large repository of images which can be easily converted to Singularity

MPI Compile and run MPI programs using self installed MPI implementation within the container

Jupyter Create Jupyter instance which runs custom notebooks. Parallelization can be done using IPython Parallel



JupyterHub

The main goal is to have an easy interface for Jupyter users in HPC.

JupyterHub⁴ implements spawning notebooks on demand for users.

IPython Parallel⁵ can be used to run parallel workers on whole HPC.

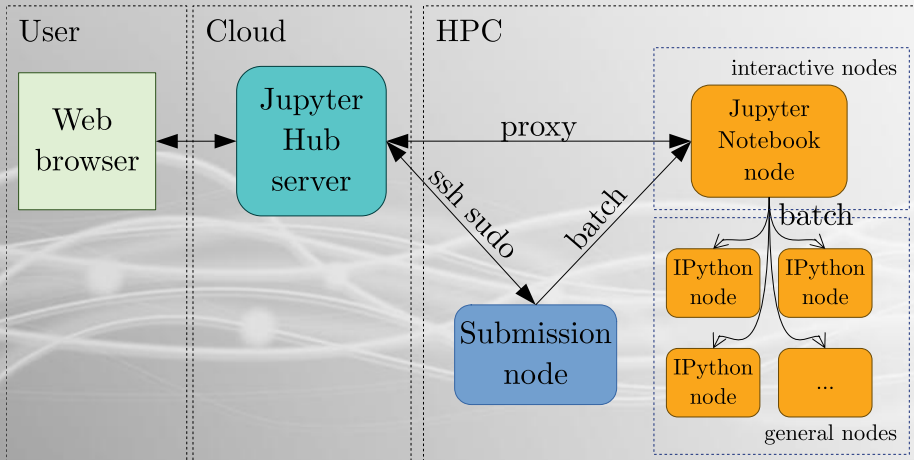
Benefits of using Jupyter in a container

- There are ready to use containers with preconfigured scientific Jupyter
- Can run multiple versions of Jupyter with the same hub
- The necessary environment for Jupyter is encapsulated
- Users can run their own Jupyter instances
- Containers can be prepared by users without admin privileges in HPC

⁴ *JupyterHub*. URL: <https://jupyter.org/hub>.

⁵ *IPython Parallel*. URL: <https://ipyparallel.readthedocs.io/en/latest/>.

Implementation in GWDG



Example - spawn

Spawner Options

Select a job profile:

Set the duration (in hours):

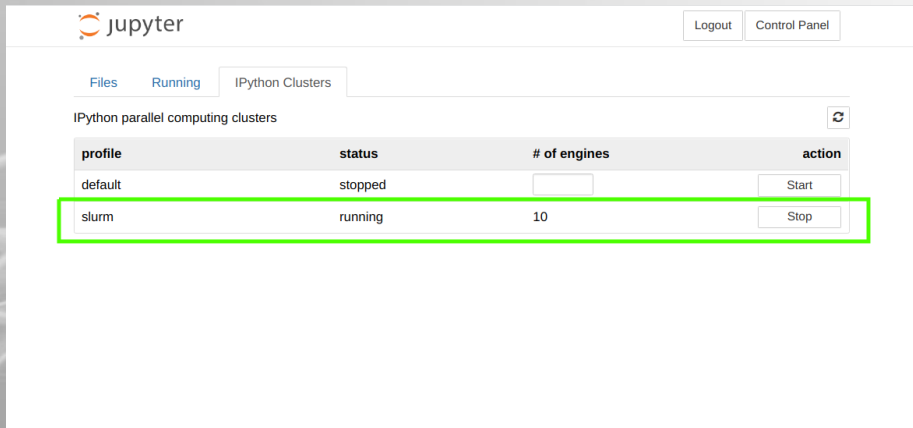
Set the number of cores:

Set the amount of memory (in GB):

Jupyter Notebook's Home directory

[Documentation](#)

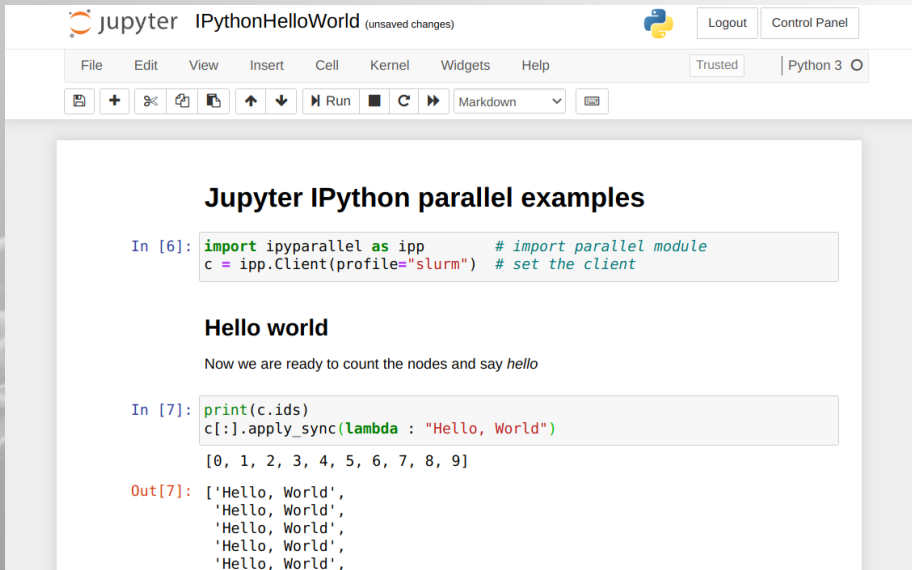
Example - set cluster



The screenshot shows the JupyterLab interface with the 'IPython Clusters' tab selected. The 'slurm' cluster is highlighted with a green box. The table below shows the details of the clusters.

profile	status	# of engines	action
default	stopped	<input type="text"/>	Start
slurm	running	10	Stop

Example - run



The screenshot shows a Jupyter Notebook interface with the following elements:

- Header: "jupyter IPythonHelloWorld (unsaved changes)" with a Python logo, "Logout", and "Control Panel" buttons.
- Menu: "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", "Help".
- Trust: "Trusted" button and "Python 3" dropdown.
- Toolbar: Includes icons for home, add, undo, redo, save, and a "Run" button.
- Code Cell 6:

```
In [6]: import ipyparallel as ipp # import parallel module
c = ipp.Client(profile="slurm") # set the client
```
- Section Header:

Hello world
- Text: "Now we are ready to count the nodes and say *hello*"
- Code Cell 7:

```
In [7]: print(c.ids)
c[:].apply_sync(lambda : "Hello, World")
```
- Output:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```
- Output 7:

```
Out[7]: ['Hello, World',
'Hello, World',
'Hello, World',
'Hello, World',
'Hello, World',
```

Conclusion

Advantages and disadvantages of using Singularity in HPC

Advantages

- Easier to deploy compared to other containerization systems
- Upgrade of Singularity is easy, even to major releases
- Less security issues because containers run with user permissions
- Only necessary isolation, devices and network are available

Disadvantages

- HPC specific hardware libraries should be manually installed/bound
- Containers cannot be built in HPC (version 3.x and later)
- Not all Docker images can be directly used in Singularity
- No long term support version is currently available

Thank you!

A decorative graphic at the bottom of the slide consisting of several overlapping, wavy, light grey lines that create a sense of motion and depth.