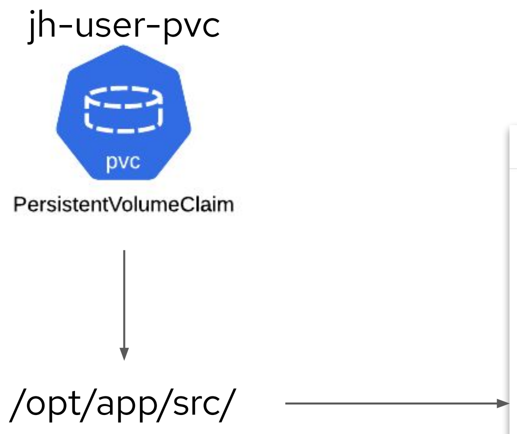
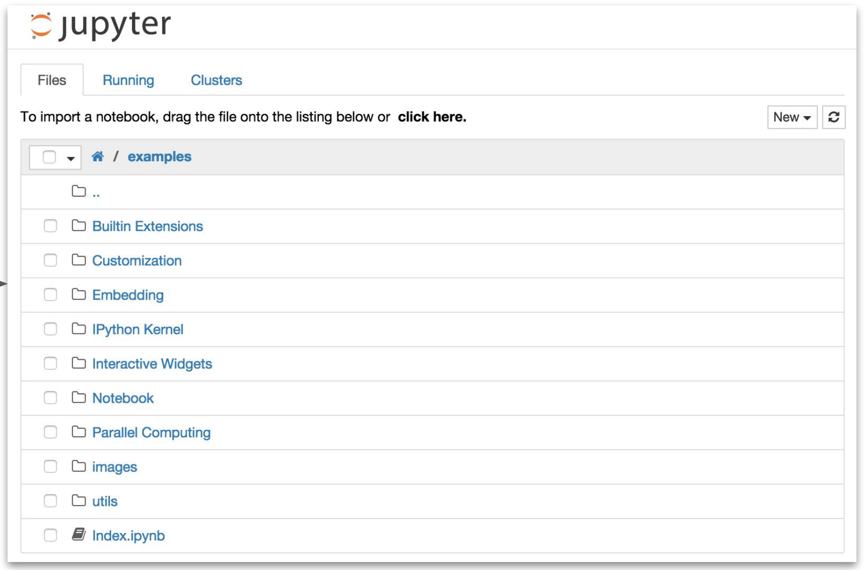
CONFIDENTIAL designator

Data Persistency

Current implementation in Jupyterhub (Open Data Hub)

- Each user gets its own personal space to store data.
- A new unique PVC is created the first time a user launches a notebook.
- This PVC is automatically mounted in the pod at notebook launch time.
- The mount point acts as the user's home and root of the notebook.





To import a notebook, drag the file onto the listing below or **click here.**

/ examples

..
Builtin Extensions
Customization
Embedding
IPython Kernel
Interactive Widgets
Notebook
Parallel Computing
images
utils
Index.ipynb

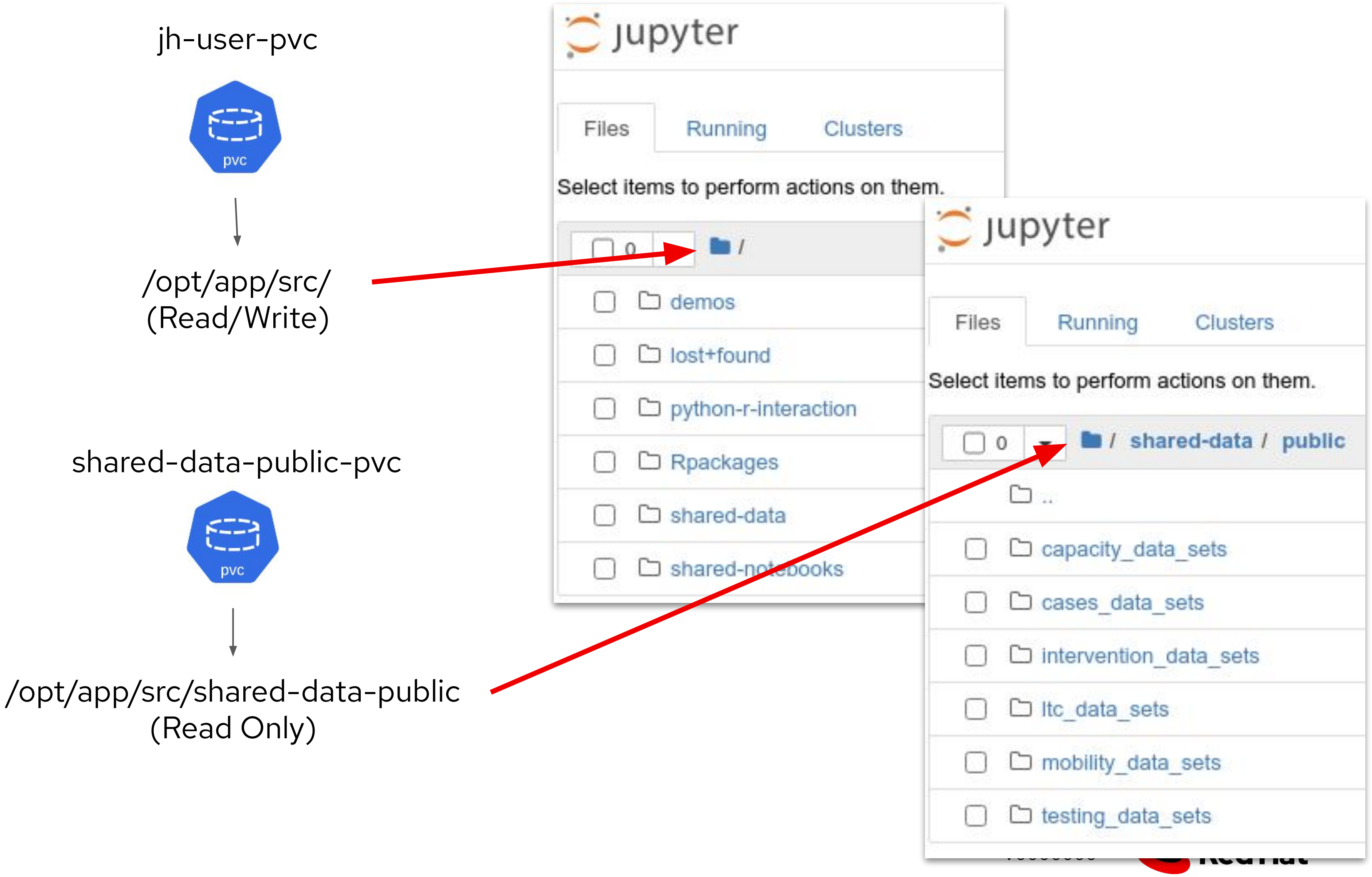
V0000000 Red Hat

Requirements and Limits

- Users want to share data and notebooks.
- They also want different access levels depending on data sensitivity (public, private, secret).
- For some environments, the Object Storage approach to data sharing may bring some limitations:
  - Overkill for small datasets (e.g. a few GBs of CSV files).
  - Not all applications or libraries support reading data from S3.
  - Existing notebooks referring to “paths” have to be modified:
    - Change management can hamper the adoption of the solution.
    - Work has to be done to make those modifications, which on top breaks the ability for further movements (local computer <-> Open Data Hub environment).

Solutions: RWX volumes!

- For Data and Notebooks sharing:
  - Create RWX PVCs for the shares.
  - Mount the PVCs as additional volumes at spawn time.
  - Handle access level control + R/W rights through config maps



Select items to perform actions on them.

0 /

- demos
- lost+found
- python-r-interaction
- Rpackages
- shared-data
- shared-notebooks

Select items to perform actions on them.

0 / shared-data / public

- ..
- capacity_data_sets
- cases_data_sets
- intervention_data_sets
- ltc_data_sets
- mobility_data_sets
- testing_data_sets

CONFIDENTIAL designator

Requirements and Limits

- Libraries or applications not in a notebook container image, or not at the required version can be:

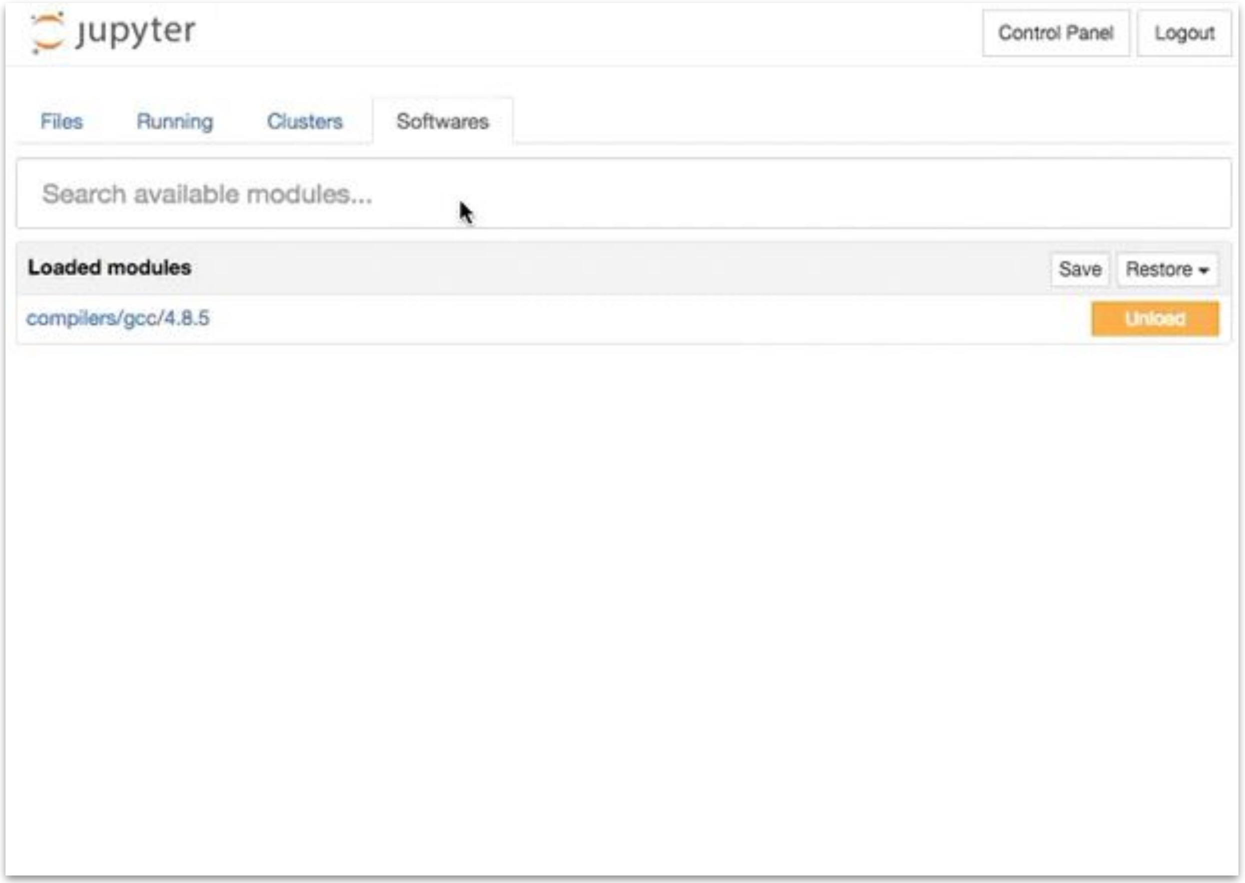
Solution	Pros	Cons
Installed temporarily in the environment (e.g. ‘pip install xyz’)	- Quick and easy - Direct control by the user (no other people involved)	- Has to be repeated each time the notebook is used, which can take time for some libraries or sets - Not everything can be installed this way (rpms,...)
Installed in new version of the container image	- No technical involvement for the user - Sanitized images handled by a dedicated team	- May take time: request to the images maintainers, new build, deployment… - Ever growing size of images to handle all requests - App or library may require update of other components which will: break the image unicity or multiply the images to satisfy all requests
Installed locally by the user in its own space	- Direct control of installation and updates	- Feasible for libraries, difficult for applications - Higher storage consumption (several 100’s of MB per user) - Reproducibility and notebook sharing can be difficult (version mismatch, conflicts...)

Red Hat

CONFIDENTIAL designator

Solutions: RWX volumes!

- For Libraries sharing (WIP):
  - Mount centralized apps/libraries collections as RWX volumes inside each container.
  - Leverage LMod to dynamically load apps and libraries.



compilers/gcc/4.8.5

V0000000

Red Hat