datalad_container Documentation

Release 0.5.2

DataLad team

Dec 04, 2019

Contents

1 Documentation 3

2 API Reference 9

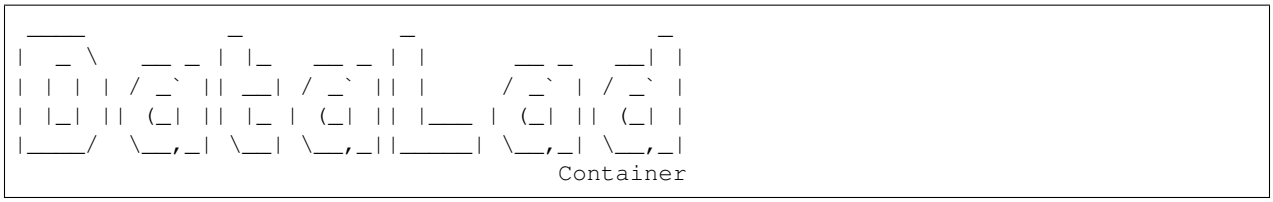
This extension equips DataLad's run/rerun functionality with the ability to transparently execute commands in containerized computational environments. On re-run, DataLad will automatically obtain any required container at the correct version prior execution.

CHAPTER 1

Documentation

- Documentation index
- *Getting started*
- *API reference*

1.1 Change log



This is a high level and scarce summary of the changes between releases. We would recommend to consult log of the DataLad git repository for more details.

1.1.1 0.5.2 (Nov 12, 2019) –

Fixes

- The Docker adapter unconditionally called `docker run` with `--interactive` and `--tty` even when stdin was not attached to a TTY, leading to an error.

1.1.2 0.5.1 (Nov 08, 2019) –

Fixes

- The Docker adapter, which is used for the "dhub://" URL scheme, assumed the Python executable was spelled "python".
- A call to DataLad's `resolve_path` helper assumed a string return value, which isn't true as of the latest DataLad release candidate, 0.12.0rc6.

1.1.3 0.5.0 (Jul 12, 2019) – damn-you-malicious-users

New features

- The default result renderer for `containers-list` is now a custom renderer that includes the container name in the output.

Fixes

- Temporarily skip two tests relying on SingularityHub – it is down.

1.1.4 0.4.0 (May 29, 2019) – run-baby-run

The minimum required DataLad version is now 0.11.5.

New features

- The call format gained the "{img_dspath}" placeholder, which expands to the relative path of the dataset that contains the image. This is useful for pointing to a wrapper script that is bundled in the same subdataset as a container.
- `containers-run` now passes the container image to `run` via its `extra_inputs` argument so that a run command's "{inputs}" field is restricted to inputs that the caller explicitly specified.
- During execution, `containers-run` now sets the environment variable `DATALAD_CONTAINER_NAME` to the name of the container.

Fixes

- `containers-run` mishandled paths when called from a subdirectory.
- `containers-run` didn't provide an informative error message when `cmdexec` contained an unknown placeholder.
- `containers-add` ignores the `--update` flag when the container doesn't yet exist, but it confusingly still used the word "update" in the commit message.

1.1.5 0.3.1 (Mar 05, 2019) – Upgrayeddd

Fixes

- `containers-list` recursion actually does recursion.

1.1.6 0.3.0 (Mar 05, 2019) – Upgrayedd

API changes

- `containers-list` no longer lists containers from subdatasets by default. Specify `--recursive` to do so.
- `containers-run` no longer considers subdataset containers in its automatic selection of a container name when no name is specified. If the current dataset has one container, that container is selected. Subdataset containers must always be explicitly specified.

New features

- `containers-add` learned to update a previous container when passed `--update`.
- `containers-add` now supports Singularity's "docker://" scheme in the URL.
- To avoid unnecessary recursion into subdatasets, `containers-run` now decides to look for containers in subdatasets based on whether the name has a slash (which is true of all subdataset containers).

1.1.7 0.2.2 (Dec 19, 2018) – The more the merrier

- list/use containers recursively from installed subdatasets
- Allow to specify container by path rather than just by name
- Adding a container from local filesystem will copy it now

1.1.8 0.2.1 (Jul 14, 2018) – Explicit lyrics

- Add support `datalad run --explicit`.

1.1.9 0.2 (Jun 08, 2018) – Docker

- Initial support for adding and running Docker containers.
- Add support `datalad run --sidecar`.
- Simplify storage of `call_fmt` arguments in the Git config, by benefiting from `datalad run` being able to work with single-string compound commands.

1.1.10 0.1.2 (May 28, 2018) – The docs

- Basic beginner documentation

1.1.11 0.1.1 (May 22, 2018) – The fixes

New features

- Add container images straight from singularity-hub, no need to manually specify `--call-fmt` arguments.

API changes

- Use "name" instead of "label" for referring to a container (e.g. `containers-run -n ...` instead of `containers-run -l`.

Fixes

- Pass relative container path to `datalad run`.
- `containers-run` no longer hides `datalad run` failures.

1.1.12 0.1 (May 19, 2018) – The Release

- Initial release with basic functionality to add, remove, and list containers in a dataset, plus a `run` command wrapper that injects the container image as an input dependency of a command call.

1.2 Acknowledgments

DataLad development is being performed as part of a US-German collaboration in computational neuroscience (CRCNS) project "DataGit: converging catalogues, warehouses, and deployment logistics into a federated 'data distribution'" (Halchenko/Hanke), co-funded by the US National Science Foundation (NSF 1429999) and the German Federal Ministry of Education and Research (BMBF 01GQ1411). Additional support is provided by the German federal state of Saxony-Anhalt and the European Regional Development Fund (ERDF), Project: Center for Behavioral Brain Sciences, Imaging Platform

DataLad is built atop the git-annex software that is being developed and maintained by Joey Hess.

1.3 Getting started

1.3.1 Getting started

The Datalad container extension provides a few commands to register containers with a dataset and use them for execution of arbitray commands. In order to get going quickly, we only need a dataset and a ready-made container. For this demo we will start with a fresh dataset and a demo container from Singularity-Hub.

```
# fresh dataset
datalad create demo
cd demo

# register container straight from Singularity-Hub
datalad containers-add my1st --url shub://datalad/datalad-container:testhelper
```

This will download the container image, add it to the dataset, and record basic information on the container under its name "my1st" in the dataset's configuration at `.datalad/config`.

Now we are all set to use this container for command execution. All it needs is to swap the command *datalad run* with *datalad containers-run*. The command is automatically executed in the registered container and the results (if there are any) will be added to the dataset:

```
datalad containers-run cp /etc/debian_version proof.txt
```

If there is more than one container registered, the desired container needs to be specifed via the `--name` option. Containers do not need to come from Singularity-Hub, but can be local images too. Via the `containers-add --call-fmt` option it is possible to configure how exactly a container is being executed, or which local directories shall be made available to a container.

At the moment there is built-in support for Singularity and Docker, but other container execution systems can be used together with custom helper scripts.

The Datalad container extension provides a few commands to register containers with a dataset and use them for execution of arbitray commands. In order to get going quickly, we only need a dataset and a ready-made container. For this demo we will start with a fresh dataset and a demo container from Singularity-Hub.

```
# fresh dataset
datalad create demo
cd demo

# register container straight from Singularity-Hub
datalad containers-add my1st --url shub://datalad/datalad-container:testhelper
```

This will download the container image, add it to the dataset, and record basic information on the container under its name "my1st" in the dataset's configuration at `.datalad/config`.

Now we are all set to use this container for command execution. All it needs is to swap the command *datalad run* with *datalad containers-run*. The command is automatically executed in the registered container and the results (if there are any) will be added to the dataset:

```
datalad containers-run cp /etc/debian_version proof.txt
```

If there is more than one container registered, the desired container needs to be specifed via the `--name` option. Containers do not need to come from Singularity-Hub, but can be local images too. Via the `containers-add --call-fmt` option it is possible to configure how exactly a container is being executed, or which local directories shall be made available to a container.

At the moment there is built-in support for Singularity and Docker, but other container execution systems can be used together with custom helper scripts.

CHAPTER 2

API Reference

2.1 Command manuals

2.2 Python API

containers_add
containers_remove
containers_list
containers_run