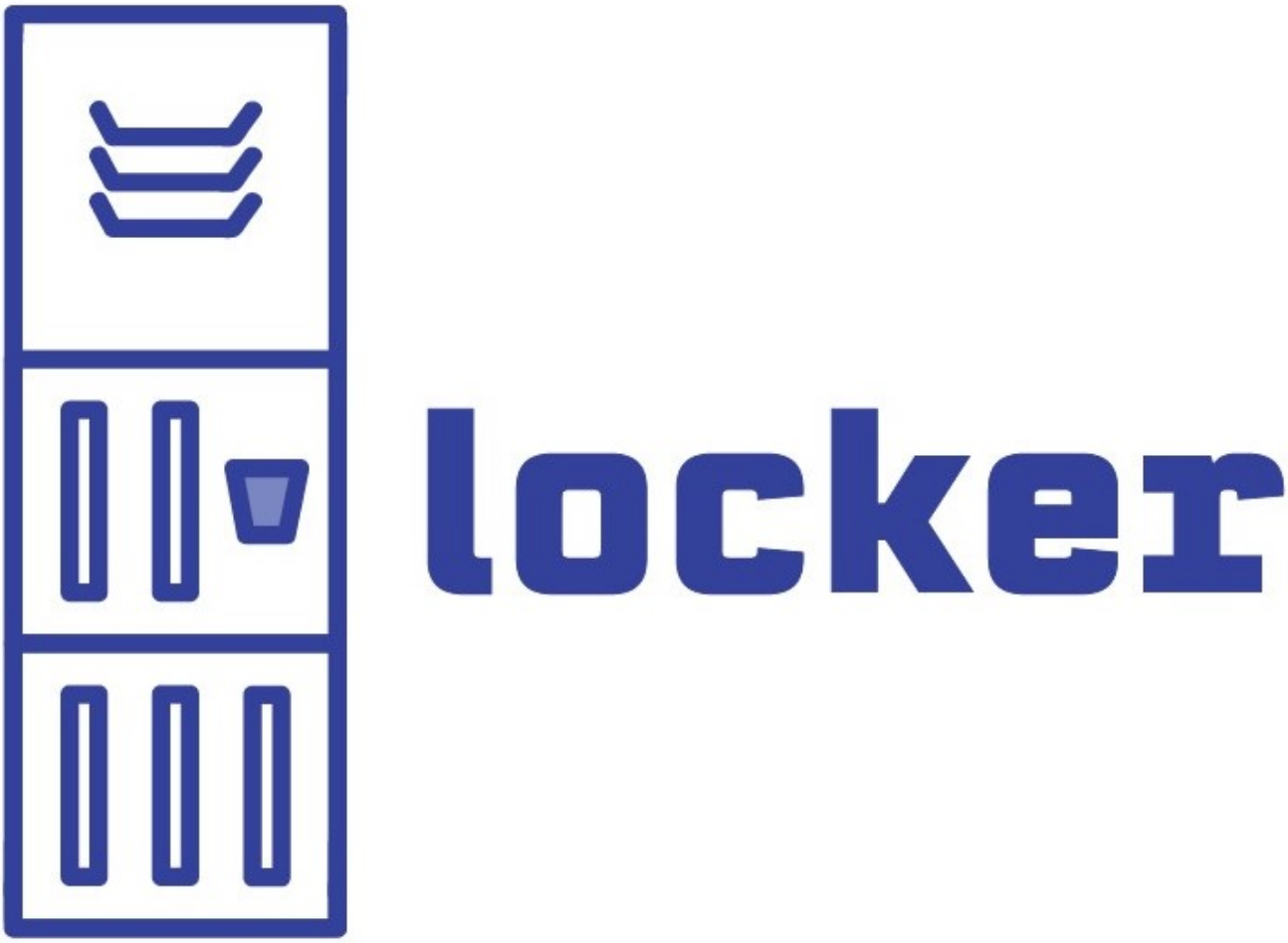
Locker

Benjamin Allan-Rahill, Intern



A command-line tool to run Domino environments a local machine

Purpose

This command-line interface was developed to add syntactic sugar to the regular Docker commands. This would allow users to run environments, usually run on Domino, locally on their machine. This would allow users to run tests and runs offline (e.g. on the train!). Locker takes care of all the defaults used to run our BMS base images. It also offers the functionality to run images retrieved from DockerHub (experts only!). The simple commands allow users to get running quickly!

Installation

1. Clone this repo

```
git clone https://biogit.pri.bms.com/allanrab/docker_local_cli.git
```

2. Navigate to the directory

```
cd docker_local_cli
```

3. Install

Make sure you have your proxys set.

```
python3 -m pip3 install .
```

Local Installation Dependencies

Locker asserts that a few things are installed:

- python3
- Docker instructions here

Please make sure these are installed before you try to run Locker.

Getting Help

Help for Locker is available using the `--help` option. This lists the high-level options, as well as the available subcommands.

```

```

```
$ locker --help
usage: locker [-h] [-V] {add,clean-up,drop-in,grab,list,run,ssh,stop} ...

optional arguments:
  -h, --help            show this help message and exit
  -V, --version         print Locker CLI version

subcommands:
  {add,clean-up,drop-in,grab,list,run,ssh,stop}
                        Available sub-commands
    add                 Add a file or dir to the container
    clean-up            Clean up running containers
    drop-in             Run a command inside the container
    grab                Grab a file or dir from the container
    list                list all the running containers or images
    run                 Run an environment on your local machine.
    ssh                 Ssh into a running container
    stop                Stop a running environment.
```

Subcommands

If no container is provided, it will default to the latest created

`add`

The `add` subcommand allows the addition of files (from the local machine) into the running container.

Git Bash

If you are using Git Bash, you will need to format you POSIX file paths correctly.

See stack overflow article here: HERE

EXAMPLE: /tmp/test.txt ==> //tmp\test.txt

```
$ locker add --help
usage: locker add [-h] [--container ID] source dest

positional arguments:
  source          The local file
  dest            Where you want the file to end up

optional arguments:
  -h, --help      show this help message and exit
  --container ID  The container to add the files to
```

Be careful of file paths if you are using **git bash** or other shells on Windows.

`clean-up`

The `clean-up` subcommand allows for the removal of stopped or running containers.

This is a **dangerous** command

```
$ locker clean-up --help
usage: locker clean-up [-h] [-a] [--container ID] [-q]

optional arguments:
  -h, --help      show this help message and exit
  -a, --all       [Optional] Stop all the containers
  --container ID  The container to add the files to
  -q, --quiet     [Optional] Don't prompt; just do.
```

drop-in

The `drop-in` subcommand uses `docker exec` to jump into a container. This can be done interactively or not.

```
usage: locker drop-in [-h] [--cmd ENTRYPOINT] [--container ID] [--mode {d,ti}]

optional arguments:
  -h, --help        show this help message and exit
  --cmd ENTRYPOINT  The command you would like to start in the container.
  --container ID    The container to add the files to
  --mode {d,ti}     [Optional] Run the command detached or interactive.
```

mode

`d` This allows a command to be run without interaction. An ssh connection will be established and then the command will be run, then the connection is close.

The command to run can me specified with the `--cmd` option.

`ti` This allows for interactive mode (essentially ssh-ing in).

grab

The `grab` subcommand allows for retrieval of files from a running container.

```
$ locker grab --help
usage: locker grab [-h] [--container ID] source dest

positional arguments:
  source          The path
  dest            Where you want the file to end up

optional arguments:
  -h, --help      show this help message and exit
  --container ID  The container to grab the files from
```

`list`

The `list` subcommand prints out the running containers.

```
$ locker list --help
usage: locker list [-h] [-a] [-i] [-r [REGISTRY]]

optional arguments:
  -h, --help            show this help message and exit
  -a, --all             [Optional] List all the containers
  -i, --images          [Optional] List the local images
  -r [REGISTRY], --registry [REGISTRY]
                        [Optional] List the images at a registry
```

`-i, --images`

This flag will list the images downloaded on the machine

`-r, --registry`

This flag prints the images and tags at registry.

The `REGSITRY` can be specified in the command or in `./settings.py`

The default `REGISTRY` is `docker.rdcloud.bms.com:443`

`run`

The `run` subcommand is the main command of locker. **All arguments are optional.** This command spins up a Docker environment for you.

The `run` command does the following:

Steps

1. Checks to see if the image/environment is downloaded.
    1. If not, the script will attempt to pull from the registry.
        - This requires network connection
    2. You have the ability to search for similar images if you would like to.
2. Checks to see if there is a running container with that environment.
    1. If so, asks if you want to start a separate one.
3. Check the port mappings
    - locker will make sure you're not missing any ports that should be mapped
    - It will also make sure you don't have any conflicting mappings
        - Can change randomly or manually for conflicts
4. Start the containers

BMS Specific

5. Locate your ssh keys
    1. Ask for location if they cannot be found at default location
6. Copy them to the container to allow ssh and mounting `/stash/`
7. Mount `/stash/`

Usage

```
$ locker run --help
usage: locker run [-h] [--cap-add CAP_ADD] [--cmd ENTRYPOINT]
                  [--device DEVICE] [--env IMAGE] [--keys KEYPATH]
                  [--label key val] [--mode {d,ti}] [-p inside outside]
                  [--user USER]

optional arguments:
  -h, --help            show this help message and exit
  --cap-add CAP_ADD     Add linux capabilities
  --cmd ENTRYPOINT      The command you would like to start in the container.
  --device DEVICE       Add device to the container
  --env IMAGE, --image IMAGE
                        [Optional] The environment that you would like to run
                        locally.
  --keys KEYPATH        [Optional] The location in which your SSH keys are
                        stored.
  --label key val       [Optional] A label to append to your container < key,
                        val >
  --mode {d,ti}         [Optional] Run the environment detached or
                        interactive.
  -p inside outside, --ports inside outside
                        [Optional] The ports you would like to use to run the
                        servers on [ssh, RStudio server].
  --user USER           Your BMS username
```

`ssh`

The `ssh` subcommand allows for a user to ssh into a running container.

Usage

```
$ locker ssh --help
usage: locker ssh [-h] [--cmd ENTRYPOINT] [--container ID] [--mode {d,ti}]

optional arguments:
  -h, --help        show this help message and exit
  --cmd ENTRYPOINT  The command you would like to start in the container.
  --container ID    The container to add the files to
  --mode {d,ti}     [Optional] Run the command detached or interactive.
```

`mode`

`d` This allows a command to be run without interaction. An ssh connection will be established and then the command will be run, then the connection is close.

The command to run can me specified with the `--cmd` option.

`ti`

This interactive mode allows the use of `ssh` right into the running container. The shell will prompt for `domino's` password. The password is **domino**.

`stop`

The `stop` subcommand stops the containers specified.

```
$ locker stop --help
usage: locker stop [-h] [-a] [-r REGISTRY] [--halt]

optional arguments:
  -h, --help            show this help message and exit
  -a, --all             [Optional] Stop all the containers
  -r REGISTRY, --registry REGISTRY
                        [Optional] Stop the images labeled with a particular
                        registry
  --halt, --slam        [Optional] Force the stop of a running container (uses
                        SIGKILL)
```

`--halt, --slam`

This option sends a SIGKILL to the processes running in the container.