
Resen

Release v2019.1.0

Dec 10, 2019

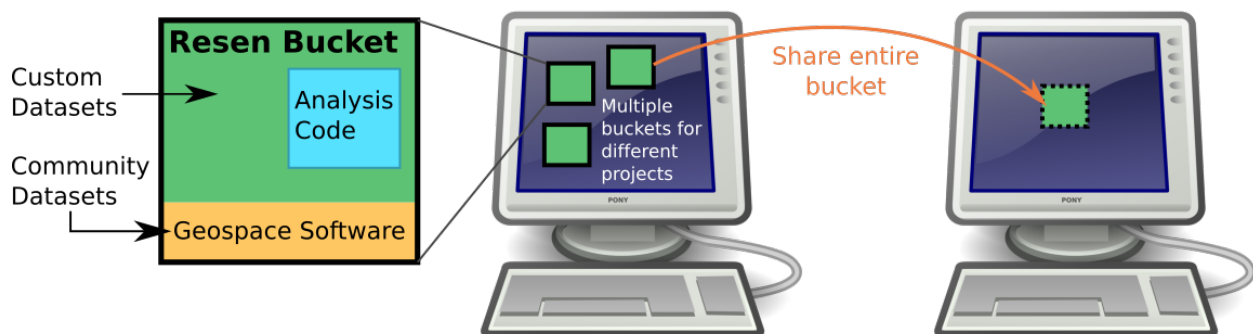
Contents

1	Resen	1
1.1	Quickstart	2
1.1.1	Installation	2
1.1.2	Usage	2
1.2	Documentation	2
2	Installation	3
2.1	General Instructions	3
2.1.1	Python 3	3
2.1.2	Docker	3
2.1.3	Resen	4
2.2	Windows Gotchas	4
2.2.1	Install Anaconda and Resen	4
2.2.2	Docker	4
3	Usage	7
3.1	Resen Workflow	7
3.1.1	Setup a New Bucket	8
3.1.2	Work with a Bucket	9
3.1.3	Remove a Bucket	11

CHAPTER 1

Resen

Resen (REproducible Software ENvironment), is a tool that enables reproducible scientific data analysis, built using python and docker. It is designed to make it easier for geospace researchers to share analysis and results, as well as build off of work others have done. Resen was developed under the InGeO project, currently supported by the National Science Foundation's Cyberinfrastructure for Sustained Scientific Innovation (CSSI) program (Grant #1835573). For more information about the InGeO project, please visit the [InGeO website](#).



Resen is based on the concept of portable environments, or buckets, where code can be developed and run independent of a users system. When you start a resen bucket, it has a variety of common geospace software packages preinstalled and ready for use. This means you have easy access to common models and datasets, and can start using them in your analysis immediately. You can also set up your bucket to access your own datasets, locally stored on your machine.

After you have completed your analysis, you can share an entire bucket with other researchers. Within the bucket, your analysis code will always run exactly the same way, regardless of what system the bucket is on. This means that other researchers should be able to reproduce your work and start building off of it immediately, instead of spending time configuring their system, installing new packages, and setting up file paths so their environment is compatible with your code.

1.1 Quickstart

1.1.1 Installation

Resen requires both python 3 and docker to be installed.

1. Install [Python 3](#)
2. Install [docker](#)
3. Install Resen with pip `pip install git+https://github.com/EarthCubeInGeo/resen.git@v2019.1.1`

Please refer to the [installation documentation](#) for more detailed instructions.

1.1.2 Usage

Resen is a command line tool. To start resen, simply enter `resen` at a command prompt:

```
$ resen
```

For a list of available commands, use the `help` command:

```
[resen] >>> help
```

[Resen Workflow Example](#)

1.2 Documentation

Complete documentation for Resen is available at <https://resen.readthedocs.io/>.

2.1 General Instructions

Resen is built off of both python 3 and docker, so you must have both of these installed for Resen to function.

2.1.1 Python 3

Python (<https://www.python.org/>) is an open source, interpreted programming language that is both powerful and easily to learn. There are many ways you can install python on your system. For new users, we recommend downloading and installing the latest Python 3 Anaconda Distribution (<https://www.anaconda.com/distribution/>) for your system. This will save you the trouble of building a python distribution from scratch.

2.1.2 Docker

Docker CE is the recommended version of Docker to use with Resen. [Installation instructions](#) can be found in the docker documentation. Please read installation instructions carefully! For convenience, some OS specific links are provided below:

MacOS: [Install Docker Desktop for Mac](#)

Important! Docker Desktop only supports the latest two versions of MacOS. Earlier versions of MacOS should install [Docker Toolbox](#).

Windows: [Install Docker Desktop for Windows](#)

Important! If you are already using virtualbox, do NOT install Docker Desktop. Instead, install [Docker Toolbox](#).

CentOS: [Get Docker CE for CentOS](#)

Debian: [Get Docker CE for Debian](#)

Fedora: [Get Docker CE for Fedora](#)

Ubuntu: [Get Docker CE for Ubuntu](#)

2.1.3 Resen

Install Resen from a python 3 environment using pip:

```
pip install git+https://github.com/EarthCubeInGeo/resen.git@v2019.1.1
```

2.2 Windows Gotchas

Resen requires both python 3 and docker to function. Here we provide a basic guide for installing both python 3 and docker. We have tested this procedure and know it works. Python 3 and Resen are easy to install. Docker is also fairly easy, but there are some subtle details that need to be emphasized for a smooth installation process.

2.2.1 Install Anaconda and Resen

Anaconda: We recommend downloading and installing the Python 3 Anaconda Distribution (<https://www.anaconda.com/distribution/>). This simplifies the installation and usage of several common software tools needed to install and run Resen.

Resen: Using the start menu search, open the “Anaconda Powershell Prompt” and install Resen using pip:

```
pip install git+https://github.com/EarthCubeInGeo/resen.git@v2019.1.1
```

Once complete, this will provide the command line command `resen`. Next, we need to install Docker.

2.2.2 Docker

For Windows, there are 2 options for installing Docker, which depends on what else you use and do with your Windows system:

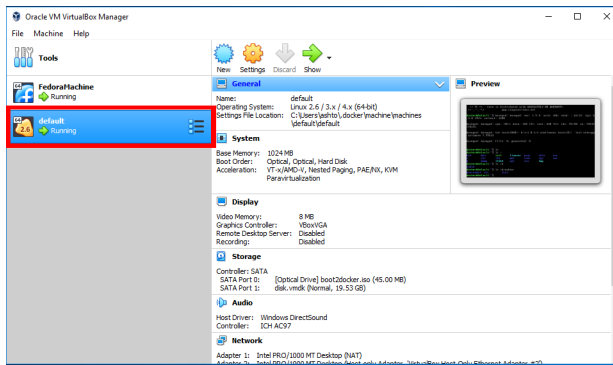
1. [Docker Desktop for Windows](#)
2. [Docker Toolbox](#)

If you use [Oracle VM VirtualBox](#) to run virtual machines on your Windows system, DO NOT install Docker Desktop. You must instead install Docker Toolbox. Docker Desktop uses Hyper-V, which is not compatible with VirtualBox.

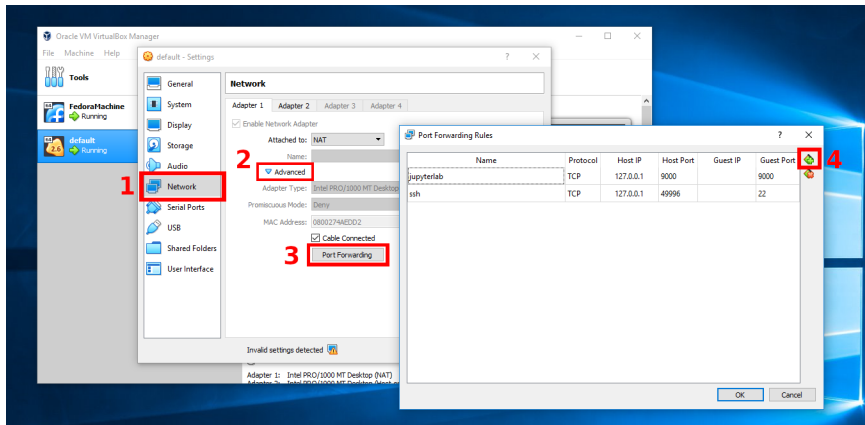
Docker Desktop TODO. If you can help fill this in, please make a PR to the develop branch for resen!

Docker Toolbox Docker Toolbox essentially works by running docker inside of a Linux virtual machine using VirtualBox. The VM that gets installed is name “default” and we will refer to this “default” Docker virtual machine as the “Docker VM”. To install it Docker Toolbox, do the following:

1. Shutdown any VirtualBox VMs that you currently have running and take note of the VirtualBox version you have installed. Docker Desktop installs an older version of VirtualBox on your system, but this version you are currently running can be upgraded back to the version you are currently running.
2. [Follow the instructions on here to install Docker Toolbox](#). Once installed, restart your computer and then run the Docker Quickstart Terminal from the start menu. TODO: insert screenshot
3. Now we need to add port forwarding and check the shared folders for the Docker VM in VirtualBox. To do this, open VirtualBox and open the “Settings” for the “default” VM, like so:

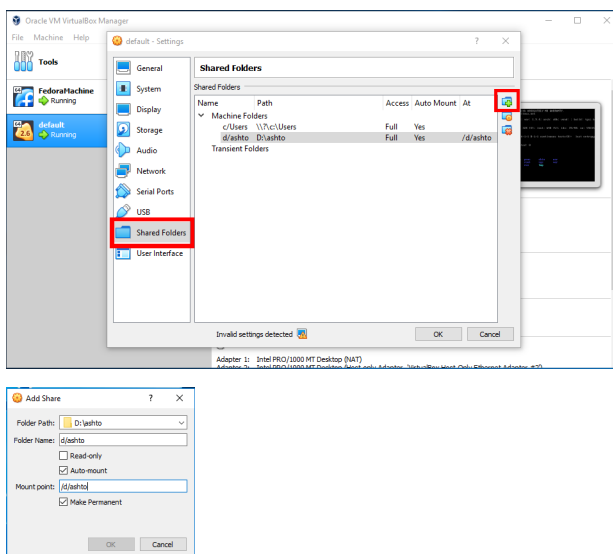


Add a new port forwarding rule by navigating to Settings->Network->Adapter 1->Advanced->Port Forwarding:



Here, we need to add a port forwarding rule for each bucket we create in Resen. Resen requires port 9000 for one bucket and then increments by 1 for every new bucket created. This means that if you have 5 buckets, you will need to make a port forward rule for ports 9000, 9001, 9002, 9003, and 9004. Change both the Host and Guest Ports as seen in the above screenshot.

Now we can optionally add Shared Folders. By default, Docker Toolbox shares the C:\Users directory with the Docker VM at /c/Users. If additional shared directory locations are desired add them. For example:



makes an additional location, D:\ashto available to the Docker VM at the location /d/ashto. After adding or

removing Shared Folders, you must restart the Docker VM. This can be done by running:

```
docker-machine restart
```

in the “Docker Quickstart Terminal”.

4. Optionally, you can now re-install the newer versions of VirtualBox that you had previously installed. Before doing this, shutdown the Docker Toolbox VM. After re-installing VirtualBox, restart your computer and then open the “Docker Quickstart Terminal” again.

Running Resen

Now you can run Resen! To do this, open an “Anaconda Powershell Prompt” and type “resen” and hit enter. A prompt should appear that asks if you are using Docker Toolbox:

```
Resen appears to be running on a Windows system. Are you using Docker Toolbox? (y/n):
```

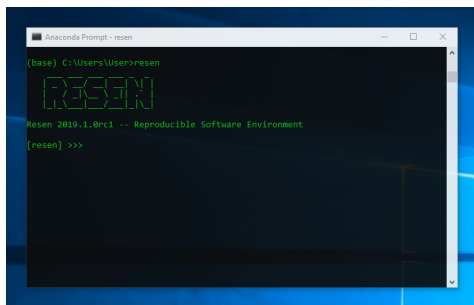
If you installed Docker Desktop for Windows, enter `n`. If you set up Docker Toolbox as described above enter `y`. If you respond `y`, you will be asked to specify the mapping between shared folders on the host machine and the Docker VM. This is referring to the Shared Folders set up in step 3 above. If you did not modify the default shared folder, the correct response should be:

```
Please specify the mapping between shared folders on the host machine and the Docker_
↵VM.
Host machine path: C:\Users
Docker VM path: /c/Users
```

This will be different if you have made a different location available to the Docker VM, such as `D:\ashto` as described above. In this case, the correct response will be:

```
Please specify the mapping between shared folders on the host machine and the Docker_
↵VM.
Host machine path: D:\ashto
Docker VM path: /d/ashto
```

Now Resen should be configured and ready to go! You should see something similar to:



CHAPTER 3

Usage

To use resen, simply enter `resen` at the command line:

```
$ resen
```

This will open the resen tool:

```
  _ _ _ _ _
 | _ \ _ / _ | _ | \ | |
 | / _ \ _ \ _ | | . ` |
 | _ \ _ | _ / _ | _ | _ |

Resen 2019.1.0 -- Reproducible Software Environment

[resen] >>>
```

Type `help` to see available commands:

```
[resen] >>> help
```

This will produce a list of resen commands you will use to manage your resen buckets:

```
Documented commands (type help <topic>):
=====
EOF      exit      help      list      remove    status
create   export   import    quit      start     stop
```

To get more information about a specific command, enter `help <command>`.

3.1 Resen Workflow

To create, import, export, and remove buckets, we use Resen. Buckets are portable, system independent environments where code can be developed and run. Buckets can be shared between Windows, Linux, and macos systems and all

analysis within the bucket will be run exactly the same. Resen buckets come preinstalled with a variety of common geospace software that can be used immediately in analysis.

The interface to a resen bucket is a jupyter lab server and access to the bucket is provided through a web browser. The user home directory is `/home/jovyan`. Any mounted storage directories are available in `mount`.

Below are instructions on how to use Resen to work with buckets. A typical workflow will involve: creation of a bucket, performing scientific data analysis inside the bucket, exporting the bucket and sharing it with colleagues. Collaborators can then import the bucket, perform additional analysis, and then export the bucket for publication in an open access citable repository, such as [Zenodo](#).

3.1.1 Setup a New Bucket

1. Creating a new bucket is performed with the command:

```
[resen] >>> create
```

The `create` command queries the user for several pieces of information required to create a bucket. First it asks for the bucket name. Creating a bucket named `amber`:

```
Please enter a name for your bucket.
Valid names may not contain spaces and must start with a letter and be less than
↳20 characters long.``
>>> Enter bucket name: amber
```

Next, the user is asked to specify the version of `resen-core` to use:

```
Please choose a version of resen-core.
Available versions: 2019.1.0
>>> Select a version: 2019.1.0
```

Optionally, one may then specify a local directory to mount into the bucket at `/home/jovyan/mount`:

```
Local directories can be mounted to either /home/jovyan/work or /home/jovyan/
↳mount/ in
a bucket. The /home/jovyan/work location is a workspace and /home/jovyan/mount/
↳is intended
for mounting in data. You will have rw privileges to everything mounted in work,
↳but can
specify permissions as either r or rw for directories in mount. Code and data
↳created in a
bucket can ONLY be accessed outside the bucket or after the bucket has been
↳deleted if it is
saved in a mounted local directory.
>>> Mount storage to /home/jovyan/mount? (y/n): y
>>> Enter local path: /some/local/path
```

Finally, the user is asked if they want jupyterlab to be started:

```
>>> Start bucket and jupyterlab? (y/n): y
```

after which resen will begin creating the bucket. Example output for a new bucket named `amber` with jupyterlab started is:

```
...adding core...
...adding ports...
...adding mounts...
```

(continues on next page)

(continued from previous page)

```

Bucket created successfully!
...starting jupyterlab...
Jupyter lab can be accessed in a browser at: http://localhost:9002/?
↪token=e7a11fc1ea42a445807b4e24146b9908e1abff82bacbf6f2

```

2. Check the status of the bucket:

```

[resen] >>> status amber

amber
=====

Resen-core Version:  2019.1.0
Status:  running
Jupyter Token:  e7a11fc1ea42a445807b4e24146b9908e1abff82bacbf6f2
Jupyter Port:  9002
Jupyter lab URL: http://localhost:9002/?
↪token=e7a11fc1ea42a445807b4e24146b9908e1abff82bacbf6f2

Storage:
Local                                Bucket                                ↪
↪Permissions
/some/local/path                    /home/jovyan/mount/path                rw

Ports:
Local      Bucket
9002      9002

```

At this point, the bucket should have a name, an image, at least one port, and optionally one or more storage locations. Status should be running if the user decided to have jupyterlab started, otherwise the status will be None.

3.1.2 Work with a Bucket

1. Check what buckets are available with list:

```

[resen] >>> list
Bucket Name      Docker Image      Status
amber            2019.1.0          running

```

If a bucket is running, it will consume system resources accordingly.

2. Stop the bucket amber:

```

[resen] >>> stop amber

```

The status of amber should now be exited:

```

[resen] >>> list
Bucket Name      Docker Image      Status
amber            2019.1.0          exited

```

The bucket will still exist and can be restarted at any time, even after quitting and restarting resen.

3. Start the bucket amber that was just stopped:

```
[resen] >>> start amber
```

The status of amber should now be running:

```
[resen] >>> status
Bucket Name      Docker Image      Status
amber            2019.1.0          running
```

4. Export bucket amber:

```
[resen] >>> export amber
```

The export command will ask a series of question. First, provide a name for the output *.tar file:

```
>>> Enter name for output tar file: /path/for/output/amber.tar
```

If desired, change the default name and tag for the exported image:

```
By default, the output image will be named "amber" and tagged "latest".
>>> Would you like to change the name and tag? (y/n): y
>>> Image name: custom_name
>>> Image tag: custom_tag
```

Specify if you want all mounted directories to be included in the exported bucket. Answering *n* to this query will allow you to see how large each mount is and specify which you would like to include. Consider excluding any mounts that are not necessary for the analysis to reduce the size of the output file:

```
The following local directories are mounted to the bucket (total 2212 MB):
/home/usr/mount1
/home/usr/mount2
/home/usr/mount3
>>> Would you like to include all of these in the exported bucket? (y/n): n
>>> Include /home/usr/mount1 [154.68095 MB]? (y/n): y
>>> Include /home/usr/mount2 [2005.28493 MB]? (y/n): y
>>> Include /home/usr/mount3 [53.59823 MB]? (y/n): y
```

Confirm that you want to continue with the export. The values shown should be considered a “high-side” approximation and may not be the actual final size:

```
This export could require up to 13337 MB of disk space to complete and will
→produce an output file up to 4600 MB.
>>> Are you sure you would like to continue? (y/n): y
Exporting bucket amber. This will take several minutes.
```

5. Import a new bucket, amber2, from a tar file amber.tar:

```
[resen] >>> import
```

This command will also ask a series of questions. First provide a name for the imported bucket:

```
Please enter a name for your bucket.
Valid names may not contain spaces and must start with a letter and be less
→than 20 characters long.
>>> Enter bucket name: amber2
```

Specify the *.tar file to import the bucket from:

```
>>> Enter name for input tar file: /path/to/file/amber.tar
```

If desired, enter a custom image name and tag. If not provided, the name an image saved on export will be used:

```
>>> Would you like to keep the default name and tag for the imported image?_
↪ (y/n): n
>>> Image name: amber2
>>> Image tag: new_tag
```

When a bucket that had mounts is imported, the mounted directories must be extracted and saved on the local machine. Resen will do this automatically, but you have the option to specify where these files should be saved instead of the default location:

```
The default directory to extract the bucket metadata and mounts to is /
↪ default/save/path/resen_amber2.
>>> Would you like to specify and alternate directory? (y/n): y
>>> Enter path to directory: /new_save_path
```

Aside from the existing mounts, you can add new mounts to a imported bucket. This is useful if you would like to repeat the analysis with a different dataset:

```
>>> Mount additional storage to the imported bucket? (y/n): y
>>> Enter local path: /new/local/path/new_mount
>>> Enter bucket path: /home/jovyan/mount/new_mount
>>> Enter permissions (r/rw): r
>>> Mount additional storage to /home/jovyan/mount? (y/n): n
```

Similar to create, you have the option to start jupyter lab immediately after the bucket is imported:

```
>>> Start bucket and jupyterlab? (y/n): y
...starting jupyterlab...
Jupyter lab can be accessed in a browser at: http://localhost:9003/?
↪ token=70532767bab0ddc4febe2790efaaf974961e961e78e6025a
```

3.1.3 Remove a Bucket

WARNING: This will permanently delete the bucket. Any work that was not saved in a mounted storage directory or downloaded from the bucket will be **permanently lost**.

The user can delete a bucket with the following command:

```
[resen] >>> remove amber
```

A bucket that is running needs to be stopped before being removed.