

Sciunit: A Reproducible Container for EarthCube Community

Raza Ahmad¹, Madeline Deeds¹, Tanu Malik¹, Young-Don Choi², Jonathan Goodall², and David Tarboton³

¹DePaul University

²University of Virginia

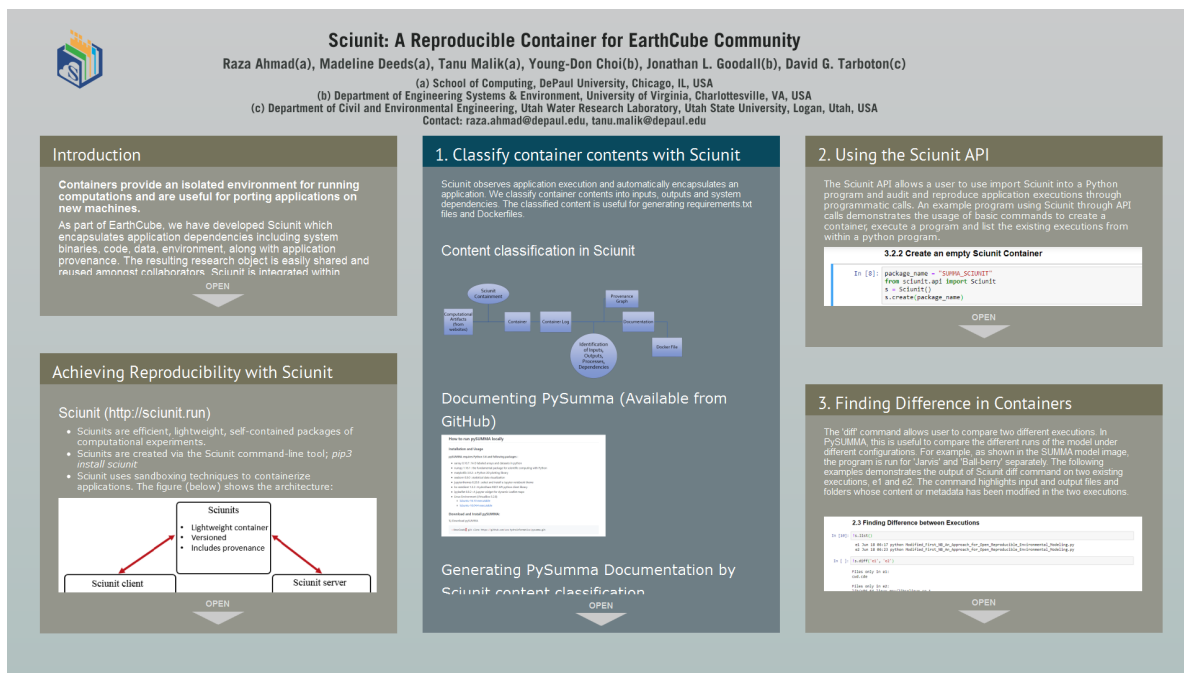
³Utah State University

November 26, 2022

Abstract

The conduct of reproducible science improves when computations are portable and verifiable. A container provides an isolated environment for running computations and thus is useful for porting applications on new machines. Current container engines, such as Linux Containers (LXC) and Docker, however, have a high learning curve, are resource-intensive, and do not address the entire reproducibility spectrum consisting of portability, repeatability, and replicability. As part of EarthCube, we have developed Sciunit (<https://sciunit.run>) which encapsulates application dependencies i.e, system binaries, code, data, environment, along with application provenance. The resulting research object can be easily shared and reused amongst collaborators. Sciunit can be used with HydroShare's JupyterHub CUAHSI notebook environment, and available to the entire community for use. In this poster, we will present three new features in Sciunit which have emerged based on community-provided use cases and discussion. Sciunit is available as a command-line utility. We will: (1) showcase the new Sciunit API. This will allow data facilities to integrate Sciunit as a reproducible environment on portals, (2) show how a Sciunit container can transition to a Docker container and vice versa, and finally, (3) demonstrate the ability to contrast two containers in terms of content and metadata. We will show these capabilities with the Hydrology use case of pySUMMA, a Python API for the Structure for Unifying Multiple Modeling Alternative (SUMMA) hydrologic model.

Sciunit: A Reproducible Container for EarthCube Community



Sciunit: A Reproducible Container for EarthCube Community
 Raza Ahmad(a), Madeline Deeds(a), Tanu Malik(a), Young-Don Choi(b), Jonathan L. Goodall(b), David G. Tarboton(c)
 (a) School of Computing, DePaul University, Chicago, IL, USA
 (b) Department of Engineering Systems & Environment, University of Virginia, Charlottesville, VA, USA
 (c) Department of Civil and Environmental Engineering, Utah Water Research Laboratory, Utah State University, Logan, Utah, USA
 Contact: raza.ahmad@depaul.edu, tanu.malik@depaul.edu

Introduction
 Containers provide an isolated environment for running computations and are useful for porting applications on new machines.
 As part of EarthCube, we have developed Sciunit which encapsulates application dependencies including system binaries, code, data, environment, along with application provenance. The resulting research object is easily shared and reused amongst collaborators. Sciunit is integrated within

1. Classify container contents with Sciunit
 Sciunit observes application execution and automatically encapsulates an application. We classify container contents into inputs, outputs and system dependencies. The classified content is useful for generating requirements bit files and Dockerfiles.
 Content classification in Sciunit
 Documenting PySUMMA (Available from GitHub)
 Generating PySUMMA Documentation by Sciunit content classification

2. Using the Sciunit API
 The Sciunit API allows a user to use import Sciunit into a Python program and audit and reproduce application executions through programmatic calls. An example program using Sciunit through API calls demonstrates the usage of basic commands to create a container, execute a program and list the existing executions from within a python program.
 3.2.2 Create an empty Sciunit Container
 3. Finding Difference in Containers
 The 'diff' command allows user to compare two different executions. In PySUMMA, this is useful to compare the different runs of the model under different configurations. For example, as shown in the SUMMA model image, the program is run for 'Janis' and 'Bailberry' separately. The following examples demonstrate the output of Sciunit diff command on two existing executions, e1 and e2. The command highlights input and output files and folders whose content or metadata has been modified in the two executions.

Achieving Reproducibility with Sciunit
 Sciunit (<http://sciunit.run>)
 • Sciunits are efficient, lightweight, self-contained packages of computational experiments.
 • Sciunits are created via the Sciunit command-line tool; pip3 install sciunit
 • Sciunit uses sandboxing techniques to containerize applications. The figure (below) shows the architecture:
 Sciunit client, Sciunit server, Sciunits (Lightweight container, Versioned, Includes provenance)

Raza Ahmad(a), Madeline Deeds(a), Tanu Malik(a), Young-Don Choi(b), Jonathan L. Goodall(b), David G. Tarboton(c)

(a) School of Computing, DePaul University, Chicago, IL, USA

(b) Department of Engineering Systems & Environment, University of Virginia, Charlottesville, VA, USA

(c) Department of Civil and Environmental Engineering, Utah Water Research Laboratory, Utah State University, Logan, Utah, USA

Contact: raza.ahmad@depaul.edu, tanu.malik@depaul.edu

PRESENTED AT:



EarthCube
 2020 EarthCube Annual Meeting
 Virtual – June 18, 2020

INTRODUCTION

Containers provide an isolated environment for running computations and are useful for porting applications on new machines.

As part of EarthCube, we have developed Sciunit which encapsulates application dependencies including system binaries, code, data, environment, along with application provenance. The resulting research object is easily shared and reused amongst collaborators. Sciunit is integrated within HydroShare's JupyterHub CUAHSI notebook environment, and available to the entire community for use.

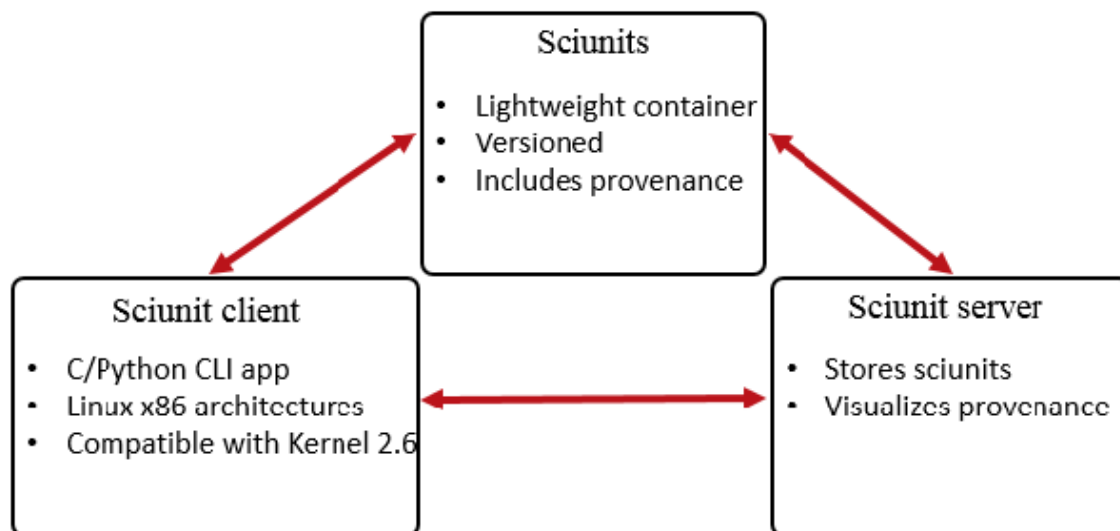
In this poster, we will present three new features in Sciunit with the Hydrology use case of pySUMMA, a Python API for the Structure for Unifying Multiple Modeling Alternative (SUMMA) hydrologic model. We describe how to:

1. Classify the contents of a Sciunit container into inputs, outputs, and system dependencies. The description of contents can generate a *requirements.txt* file or used for writing a *dockerfile*. We show how the container contents generate system dependencies for a Hydrology model.
2. Use the Sciunit API in a Notebook. Sciunit was so far available as a CLI. The API allows programmatic access to Sciunit commands. We show the use of API in a Hydrology notebook.
3. Use the *diff* command in a Sciunit container. We make parameter changes in a Hydrology model and use Sciunit *diff* to show which files (content and metadata) change.

ACHIEVING REPRODUCIBILITY WITH SCIUNIT

Sciunit (<http://sciunit.run>)

- Sciunits are efficient, lightweight, self-contained packages of computational experiments.
- Sciunits are created via the Sciunit command-line tool; `pip3 install sciunit`
- Sciunit uses sandboxing techniques to containerize applications. The figure (below) shows the architecture:

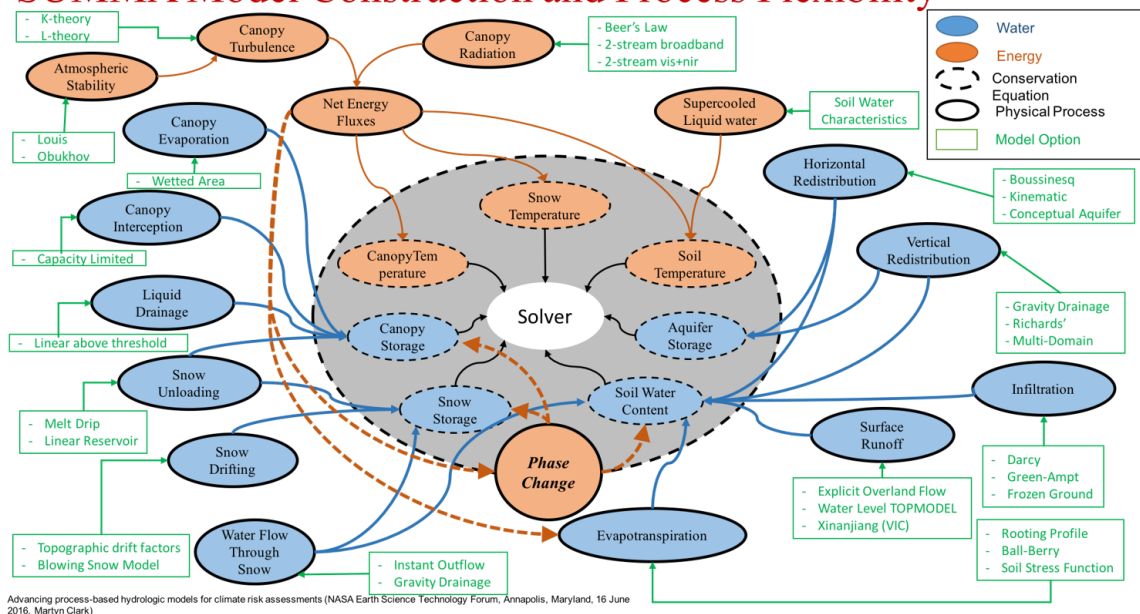


Hydrology Usecase

We demonstrate our results on the Python version of the SUMMA Model in Hydrology. The source code for auditing PySUMMA using sciunit can be found at the following notebook:

<https://www.hydroshare.org/resource/75f31565dbd24c198450b9d37c6fcf74/>
[\(https://www.hydroshare.org/resource/75f31565dbd24c198450b9d37c6fcf74/\)](https://www.hydroshare.org/resource/75f31565dbd24c198450b9d37c6fcf74/)

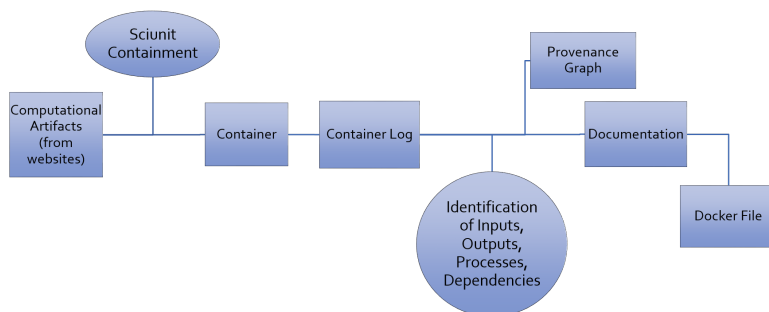
SUMMA Model Construction and Process Flexibility



1. CLASSIFY CONTAINER CONTENTS WITH SCIUNIT

Sciunit observes application execution and automatically encapsulates an application. We classify container contents into inputs, outputs and system dependencies. The classified content is useful for generating requirements.txt files and Dockerfiles.

Content classification in Sciunit



Documenting PySumma (Available from GitHub)

How to run pySUMMA locally

Installation and Usage

pySUMMA requires Python 3.6 and following packages :

- xarray 0.10.7 : N-D labeled arrays and datasets in python
- numpy 1.16.1 : the fundamental package for scientific computing with Python
- matplotlib 3.0.2 : a Python 2D plotting library
- seaborn 0.9.0 : statistical data visualization
- jupyterthemes 0.20.0 : select and install a Jupyter notebook theme
- hs-restclient 1.3.3 : HydroShare REST API python client library
- ipyleaflet 0.9.2 : A Jupyter widget for dynamic Leaflet maps
- Linux Environment (VirtualBox 5.2.8)
 - ubuntu-16.10 executable
 - ubuntu-16.04.4 executable

Download and Install pySUMMA:

1) Download pySUMMA

```
~/Downloads$ git clone https://github.com/vera-hydroinformatics/pysumma.git
```

Generating PySumma Documentation by Sciunit content classification

Listed Packages	Identified Packages
xarray{0.10.7} numpy{1.16.1} matplotlib{3.0.2} hs-restclient{1.3.3} ipyleaflet{0.9.2} seaborn{0.9.0} jupyterthemes{0.20.0}	xarray{0.10.7} numpy{1.16.1} matplotlib{3.0.2} hs-restclient{1.3.3} ipyleaflet{0.9.2}
Identified Sub-Packages	
Pygments{2.2.0} asyncio backcall{0.1.0} blinker{1.3} certifi{2018.10.15} cftime{1.0.2.1} geopandas{0.4.0} html http ipykernel{5.1.0} ipython- genutils{0.2.0} ipython{7.1.1} ipywidgets{7.4.2} jedi{0.13.1} jupyter- core{4.4.0} netCDF4{1.4.2} pandas{0.23.4} parso{0.3.1} pexpect{4.6.0} prompt-toolkit{2.0.7} ptyprocess{0.6.0} pyparsing{2.3.0} pysumma{0.1} pytz{2018.7} pyzmq{17.1.2} requests-oauthlib{1.0.0} requests-toolbelt{0.8.0} tornado{5.1.1} traitlets{4.3.2} traitlets{0.2.1} wcwidth{0.1.7}	
Python Built-In Packages	
chardet collections concurrent ctypes dateutil distutils email encodings idna importlib jinja2 json logging markupsafe multiprocessing oauthlib pkg_resources pydoc_data requests sqlite3 unittest urllib urllib3 xml	

2. USING THE SCIUNIT API

The Sciunit API allows a user to use import Sciunit into a Python program and audit and reproduce application executions through programmatic calls. An example program using Sciunit through API calls demonstrates the usage of basic commands to create a container, execute a program and list the existing executions from within a python program.

3.2.2 Create an empty Sciunit Container

```
In [8]: package_name = "SUMMA_SCIUNIT"
        from sciunit.api import Sciunit
        s = Sciunit()
        s.create(package_name)
```

Opened empty sciunit at /home/jovyan/data/sciunit/SUMMA_SCIUNIT

3.2.4 Execute Sciunit to create the Sciunit Container

```
In [*]: s.exec('python Modified_First_NB_An_Approach_for_Open_Reproducible_Environmental_Modeling.py')
```

```
[SUMMA_SCIUNIT e2] python Modified_First_NB_An_Approach_for_Open_Reproducible_Environmental_Modeling.py
Date: Thu, 18 Jun 2020 06:23:29 +0000
```

3.2.5 Show the Created Sciunit Container

```
In [21]: s.list()
```

```
e1 Jun 18 06:17 python Modified_First_NB_An_Approach_for_Open_Reproducible_Environmental_Modeling.py
e2 Jun 18 06:23 python Modified_First_NB_An_Approach_for_Open_Reproducible_Environmental_Modeling.py
```

```
In [22]: s.show('e1')
```

```
id: e1
sciunit: SUMMA_SCIUNIT
command: python Modified_First_NB_An_Approach_for_Open_Reproducible_Environmental_Modeling.py
size: 279.99 MB
started: 2020-06-18 06:17
```

3. FINDING DIFFERENCE IN CONTAINERS

The 'diff' command allows user to compare two different executions. In PySUMMA, this is useful to compare the different runs of the model under different configurations. For example, as shown in the SUMMA model image, the program is run for 'Jarvis' and 'Ball-berry' separately. The following examples demonstrates the output of Sciunit diff command on two existing executions, e1 and e2. The command highlights input and output files and folders whose content or metadata has been modified in the two executions.

2.3 Finding Difference between Executions

```
In [10]: !s.list()

e1 Jun 18 06:17 python Modified_First_NB_An_Approach_for_Open_Reproducible_Environmental_Modeling.py
e2 Jun 18 06:23 python Modified_First_NB_An_Approach_for_Open_Reproducible_Environmental_Modeling.py

In [ ]: !s.diff('e1', 'e2')

Files only in e1:
cwd.cde

Files only in e2:
lib/x86_64-linux-gnu/libselinux.so.1
lib/x86_64-linux-gnu/libdl.so.2
lib/
etc/ld.so.cache
bin/ls
root/sciunit/ls.cde

Files with changed size:
cde.full-environment.cde-root
cde.log

Files with changed modified time:
cde.log
provenance.cde-root.1.log

Files with changed permissions:
none
```

ABSTRACT

The conduct of reproducible science improves when computations are portable and verifiable. A container provides an isolated environment for running computations and thus is useful for porting applications on new machines. Current container engines, such as Linux Containers (LXC) and Docker, however, have a high learning curve, are resource-intensive, and do not address the entire reproducibility spectrum consisting of portability, repeatability, and replicability. As part of EarthCube, we have developed Sciunit (<https://sciunit.run>) which encapsulates application dependencies i.e, system binaries, code, data, environment, along with application provenance. The resulting research object can be easily shared and reused amongst collaborators. Sciunit can be used with HydroShare's JupyterHub CUAHSI notebook environment, and available to the entire community for use.

In this poster, we will present three new features in Sciunit which have emerged based on community-provided use cases and discussion. Sciunit is available as a command-line utility. We will: (1) showcase the new Sciunit API. This will allow data facilities to integrate Sciunit as a reproducible environment on portals, (2) show how a Sciunit container can transition to a Docker container and vice versa, and finally, (3) demonstrate the ability to contrast two containers in terms of content and metadata. We will show these capabilities with the Hydrology use case of pySUMMA, a Python API for the Structure for Unifying Multiple Modeling Alternative (SUMMA) hydrologic model.

REFERENCES

1. J. Chuah, M. Deeds, T. Malik, Y. Choi, J. Goodall, “Documenting Computing Environments for Reproducible Experiments”, In *Parallel Computing: Technology Trends*, 756-765, 2020, doi: 10.3233/APC200106.
2. D.H. Ton That, G. Fils, Z. Yuan, T. Malik, “Sciunits: Reusable Research Objects”, In *IEEE 13th International Conference on e-Science (e-Science) (eScience)*, 374-383, 2017, doi: 10.1109/eScience.2017.51.
3. Z. Yuan, D.H. Ton That, S. Kothari, G. Fils, T. Malik, “Utilizing Provenance in Reusable Research Objects”, In *MDPI Informatics, Special Issue on Using Computational Provenance*, Vol 5(1), 2018, doi: 10.3390/informatics5010014.