Integrating Scientific Cyberinfrastructures to Improve Reproducibility in Computational Hydrology: Example for HydroShare and GeoTrust

Bakinam T. Essawy [a], Jonathan L. Goodall [a*], Wesley Zell[b], Daniel Voce[c], Mohamed M. Morsy[a,d], Jeffrey Sadler[a], Zhihao Yuan[e], and Tanu Malik[e]

[a] Department of Civil and Environmental Engineering, University of Virginia, 351 McCormick Road, PO Box 400742, Charlottesville, VA, 22904, USA

[b] Earth Systems Modeling Branch, US Geological Survey, 12201 Sunrise Valley Dr., Reston, VA, USA

[c] Department of Electrical and Computer Engineering, University of Virginia, 351 McCormick Road, PO Box 400743, Charlottesville, VA, 22904, USA

[d] Irrigation and Hydraulics Engineering Department, Faculty of Engineering, Cairo University, P.O. Box 12211, Giza 12613, Egypt

[e] College of Computing and Digital Media, DePaul University, Chicago, IL 60604, USA

[*] To whom correspondence should be addressed (E-mail: goodall@virginia.edu; Address: University of Virginia, Department of Civil and Environmental Engineering, PO Box 400742, Charlottesville, Virginia 22904; Tel: (434) 243-5019)

Highlights:

- Method for packaging and publishing scientific workflows
- Integration between GeoTrust and HydroShare projects
- GeoTrust is used to easily package environmental models as containers
- HydroShare is used to document and share packaged workflows
- An example application is provided for using a MODFLOW-NWT model

Abstract

The reproducibility of computational environmental models is an important challenge that calls for open and reusable code and data, well-documented workflows, and controlled environments that allow others to verify published findings. This requires an ability to document and share raw datasets, data preprocessing scripts, model inputs, outputs, and the specific model code with all associated dependencies. HydroShare and GeoTrust, two scientific cyberinfrastructures under development, can be used to improve reproducibility in computational hydrology. HydroShare is a web-based system for sharing hydrologic data and models as digital resources including detailed, hydrologic-specific resource metadata. GeoTrust provides tools for scientists to efficiently reproduce and share geoscience applications. This paper outlines a use case example, which focuses on a workflow that uses the MODFLOW model, to demonstrate how the cyberinfrastructures HydroShare and GeoTrust can be integrated in a way that easily and efficiently reproduces computational workflows.

Keywords:

Computational reproducibility; hydrologic modeling; MODFLOW; metadata

1. Software availability

The software created in this research is free and open source. The software information and availability are as follows:

Developers: Bakinam T. Essawy, Daniel Voce, and Wesley Zell

Programming language: Python, Bash

GitHub link: https://github.com/uva-hydroinformatics-lab/AWS_MODFLOW.

2. Introduction

The challenge of creating more open and reusable code, data, and formal workflows that allow others to verify published findings is gaining attention in the scientific community (Borgman, 2012; David et al., 2016; Gorgolewski and Poldrack, 2016; Meng et al., 2015; Peng, 2011; Qin et al., 2016). Reproducibility is important for both verifying previous results as well as building upon the prior computational research of other scientists. Although we can achieve standard reproducibility for most computational research, there are certain cases in which reproducibility remains difficult to achieve. This challenge is not caused only by technical barriers but also by limited documentation of the research to be replicated and the potentially complex requirements for how the software is packaged, installed, and executed (Piccolo and Frampton, 2016). Recent papers have argued the need and have proposed approaches to improve reproducibility, both within geosciences generally and the hydrologic sciences specifically (David et al., 2016; Essawy et al., 2016; Gil et al., 2016; Hutton et al., 2016). Reproducibility of research is said to be achieved if the scientist was able to preserve sufficient computational artifacts in a way that can be replicated in the future (Meng et al., 2015).

Here we consider reproducibility to be the ability to repeat in the same exact form and then document and share digital resources previously used to complete an analysis. These digital resources include (1) initial raw, unprocessed datasets; (2) data preprocessing scripts used to clean and organize the data; (3) model inputs; (4) model results; and (5) the specific model code along with all of its dependencies. Figure 1 shows a typical conceptual workflow that needs to be repeated for computational reproducibility. These data, software, and environments are often integrated into workflows (as computational experiments) that allow scientists to re-run an analysis from raw initial datasets and obtain the same model results.

There are different requirements for reproducibility depending on the nature of the research. For example, laboratory experiments require capturing descriptive information about protocols and methods, leading to empirical reproducibility. Computational reproducibility, on the other hand, requires descriptive information about the software and workflow details of model-based research (Todden, 2013). Any workflow that is computationally reproducible must be general and able to address the heterogeneous landscape of tools and approaches used within the target scientific community. In hydrology, scientists use a large variety of computational models, many of which have decades of development effort behind them (Singh et al., 2002). Computational modeling can often require a significant amount of effort and time to prepare model inputs and to calibrate and validate model parameters. Depending on the complexity of the system being modeled and the experience of the modeler, these aspects can make reproducing computational hydrologic experiments particularly challenging.

Addressing the challenges for achieving reproducibility in computational workflow has been the topic of many studies. Until now, most approaches have either focused on the logical preservation (i.e., sufficient documentation of a workflow and its components to allow for reproduction later on) or physical preservation (i.e., workflow conservation by packaging all of its components allowing identical replication) (Santana-Perez et al., 2017). It is hard to achieve a high level of reproducibility while using one of these approaches in isolation; rather, the integration of both physical and logical preservation is required to achieve a high level of reproducibility. Some efforts have been made to integrate both logical and physical preservation for computational workflows, such as the Topology and Orchestration Specification for Cloud Applications (TOSCA). The TOSCA framework supports documentation for both the top-level structure of the abstract workflow and the execution environment details (logical). TOSCA also provides packaging functionality for the workflow (physical) (Qasha et al., 2016). In a similar way, our approach provides both logical and physical preservation. However, the functionality is extended to allow for automated creation, documentation, publication, and cloud-based execution of scientific workflow packages.

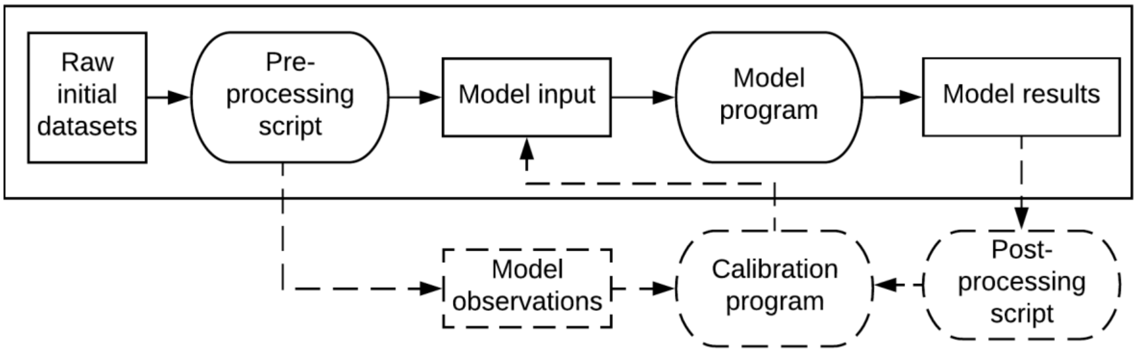


Figure 1 A typical conceptual workflow that needs to be repeated for computational reproducibility. Dashed lines indicate processes for model calibration that are not discussed in this study.

This research presents a solution for achieving a higher level of reproducibility by using GeoTrust's *Sciunit-CLI* tool and HydroShare. HydroShare (http://www.hydroshare.org) and GeoTrust (http://geotrusthub.org) are two new cyberinfrastructures under active development that aim to improve reproducibility in computational hydrology. The methods described in this paper can be used to assist scientists to more easily repeat, reproduce, and verify a computational experiment (Malik, 2017). This method goes beyond open source and simply shared by allowing portability in different hardware and software environments and reproducible analyses with different datasets. This level of reproducibility is not easily achieved by using HydroShare or GeoTrust in isolation. For example, GeoTrust does not provide a community of users who can verify analyses or the variety of datasets that are required for verification; HydroShare, however, does provide these. Similarly, while HydroShare simplifies the process of sharing code, data, and descriptive metadata, it does not address the challenge of sharing the computational environment required for the workflow and then repeating the computational workflow with different datasets. This paper presents the design and implementation of a workflow that takes advantage of the complementary strengths of the two systems. HydroShare is used to share key digital resources in the workflow, while GeoTrust is used to capture, encapsulate, and make portable model execution. An example application of the approach is presented using MODFLOW-NWT, a version of the United States Geological Survey's groundwater model, MODFLOW (Niswonger et al., 2011).

The remainder of the paper is organized as follows. First, additional background on the HydroShare and GeoTrust projects is provided. This background section is meant to orient readers on key aspects of these projects. Next, the methodology section shows the system design and the use case application for the MODFLOW-NWT model. In the results section, the system implementation of the HydroShare and GeoTrust integration approach is presented and demonstrated by using the use case results as an example application. Finally, a discussion and conclusions section summarizes the key aspects of the approach and outlines opportunities for future research to advance on known limitations of the approach.

3. Background

3.1. HydroShare

HydroShare is an open source web-based system developed for hydrologic scientists to easily share, collaborate around, and publish all types of scientific data and models including detailed, hydrologic-specific resource metadata (Tarboton et al., 2014a, 2014b). HydroShare has been developed with the support of the United States National Science Foundation (NSF). Following the completion of the original NSF grant, the Consortium of Universities for the

Advancement of Hydrologic Sciences Incorporated (CUAHSI) (also funded by the NSF) assumed long-term support for HydroShare’s operation and maintenance. In HydroShare, digital content is stored and referred to as a "resource." Each resource is a unit used for management and access control within HydroShare. Every resource has a resource type (Horsburgh et al., 2015). HydroShare assigns a unique identifier for each newly created resource; this identifier is known as the Resource ID. The "generic" resource type supports the Dublin Core metadata standard (Weibel et al., 1998) and more specific resource types expand on this metadata standard for well-defined data types. For example, "Model Operating System" is one of the extended metadata terms for the "Model Program" resource type, which is used for sharing a computational model programs in HydroShare (Morsy et al., 2017).

HydroShare provides a Representational State Transfer (REST) Application Program Interface (API) that allows third-party applications to interact with HydroShare resources. (https://github.com/hydroshare/hydroshare/wiki/HydroShare-REST-API#design-document). Developers can create web-apps that use HydroShare’s REST API to interact with HydroShare resources. Web-app developers can catalogue their apps in HydroShare via the "Web-app" resource type (Swain et al., 2016). When a developer creates a web-app resource in HydroShare, the developer specifies which resource types are relevant to the web-app and the URL that will be called when the web-app is executed from the landing page of the resource that the web-app is acting on. After a developer adds a web-app as a resource in HydroShare, HydroShare users can execute that app through HydroShare's web interface to act on relevant resources that they have access to.

Although there are several different resource types supported by HydroShare, two of the main resource types relevant to this paper deal with computational models. HydroShare divides computational models into two separate but linked resource types: a) the model program and b) the model instance. The model program includes the software for executing a specific instance of the model and the model instance are the input files required for executing the model and, optionally, the output files after a model instance has been executed by a model program (Horsburgh et al., 2015; Morsy et al., 2017, 2014). Additionally, a Model Instance Resource type can be linked to a model program resource type using the "ExecutedBy" term, assisting with reproducibility of the model instance (Morsy et al., 2017). Other HydroShare resource types used in this paper include the Composite resource type, which allows uploading metadata files at both file and resource level; the collections resource type, which stores any number of individual resources within HydroShare as a single, aggregate resource; and the web-app resource type, which is the Digital content stored in HydroShare and referred to it as a "resource."

3.2. GeoTrust

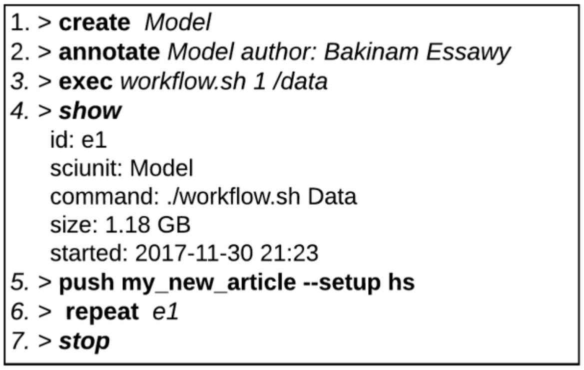
The GeoTrust project, also funded by the NSF through their EarthCube program, aims to create cyberinfrastructure that assists scientists to efficiently reproduce and share geoscience applications used in research (Malik et al., 2017). The project has done this primarily by developing the concept of a "sciunit" (https://sciunit.run/), an efficient, lightweight, self-contained digital package of an ad-hoc computational workflow that can be repeated in other environments. The sciunit advances the concept of a research object, an aggregation of digital artifacts such as code, data, scripts, and temporary experiment results associated with a research paper. The sciunit provides an authoritative and far more complete record of a piece of research (Hai et al., 2017). To create, maintain, and publish sciunits, the GeoTrust project has developed a software tool for Linux environments called *Sciunit-CLI*.

One of the main advantages of a sciunit is its portability, which allows it to be easily run on various computing environments. To accomplish this, *Sciunit-CLI* creates sciunits using Docker, a widely used containerization software. Docker wraps a piece of software in a complete filesystem that contains everything needed to run the software, including code, software runtime, system tools, and system libraries in a Docker container (Owsiak et al., 2017). By leveraging Docker, sciunits are packaged with all of their dependencies. In this way, any sciunit can be executed in any environment in which both Docker and the *Sciunit-CLI* tool are installed regardless of other computer configurations (Hai et al., 2017). This capability eliminates the burden of configuring a running environment with all software dependencies, which can be complex, in order to reuse a scientific workflow and reproduce its results.

In addition to ensuring the portability of sciunits, *Sciunit-CLI* automates some documentation of the workflow packaged into a sciunit, including environment dependencies. The automation of documenting all code, data, and environment dependencies alleviates what is typically a burdensome task for scientists. Importantly, *Sciunit-CLI* also records retrospective provenance of the workflow execution, which can be used for re-running containers (Pham et al., 2014). Because it contains all of the required dependencies, the sciunit can be rerun, and the outputs reproduced, using any other deployment configuration that also has *Sciunit-CLI* installed. When *Sciunit-CLI* creates a sciunit, it includes three types of metadata: annotation metadata (populated by the user) and provenance and version metadata (generated automatically by *Sciunit-CLI*).

Figure 2 shows an example user interaction with the *Sciunit-CLI* tool. The user runs the *create* command and provides a name, *"Model"* in the example. To create a container or a package within the sciunit, the user runs the *package* command and provides the workflow name (e.g., "workflow.sh") along with any inputs for the workflow (e.g., "data"). The user application can be written in any combination of programming languages, and many containers can be created within the same sciunit.

Sciunit-CLI works in a distributed fashion, similar to the Git version control philosophy, such that the sciunits are stored only locally until explicitly shared with a remote repository. This method of operation allows distributed collaborators to work offline on the same sciunit. When a user is ready to share, they can publish the sciunit container to any remote web-repository using the *publish* command. To use the publish command, the remote repository must be configured within the *Sciunit-CLI* tool. This command line prompts first-time users to provide their remote web-repository credentials. The remote repository reads the container's contents, stores the container's digital artifacts in the appropriate remote sciunit, and associates the container with an appropriate cloud execution server on which it can potentially re-execute. In our case, we used HydroShare as the remote repository to publish our packaged sciunit in order to use HydroShare's support for rich metadata and its ability to integrate third-party applications. The latter allowed us to automate the cloud-based execution of this packaged sciunit.



```
1. > create  Model
2. > annotate Model author: Bakinam Essawy
3. > exec workflow.sh 1 /data
4. > show
     id: e1
     sciunit: Model
     command: ./workflow.sh Data
     size: 1.18 GB
     started: 2017-11-30 21:23
5. > push my_new_article --setup hs
6. >  repeat  e1
7. > stop
```

Figure 2 A example user interaction with sciunit client.

4. Methodology

4.1. System Design

The combined GeoTrust and Hydroshare system is designed to connect a repeatable computational workflow with its input data in a reproducible way. As such, both the computational workflow and the data must be stored in a public repository that has extensive metadata support. In addition to public accessibility of the data and the computational workflow, the execution of the workflow must also be made publicly available to ensure reproducibility and transparency. The technology for producing a repeatable computational workflow is provided by the GeoTrust *Sciunit-CLI,* while the technology for public storage and metadata support is provided by CUAHSI's HydroShare. Therefore, the main design aspect of this work consisted of designing a publicly accessible method of execution in which sciunits built with the *Sciunit-CLI* and stored in HydroShare could be executed using input data also stored in HydroShare. This was done in two parts. The first was to build in functionality for publishing a sciunit through HydroShare. The second part was to automate the execution of a sciunit from HydroShare using HydroShare web-apps.

4.1.1. Integrating Sciunit-CLI with HydroShare

Figure 3 shows an activity diagram of the system design for integrating GeoTrust *Sciunit-CLI* and HydroShare. To achieve this integration, *Sciunit-CLI* was extended to support sharing of sciunits through HydroShare. This functionality was implemented using HydroShare's REST API. To publish their sciunit on HydroShare, the user must provide valid HydroShare credentials. In the current implementation, the sciunit resource is published on HydroShare as a Composite Resource Type. Once the resource for the sciunit is created within HydroShare, the user can log into HydroShare and edit the metadata fields to more fully describe the sciunit resource.

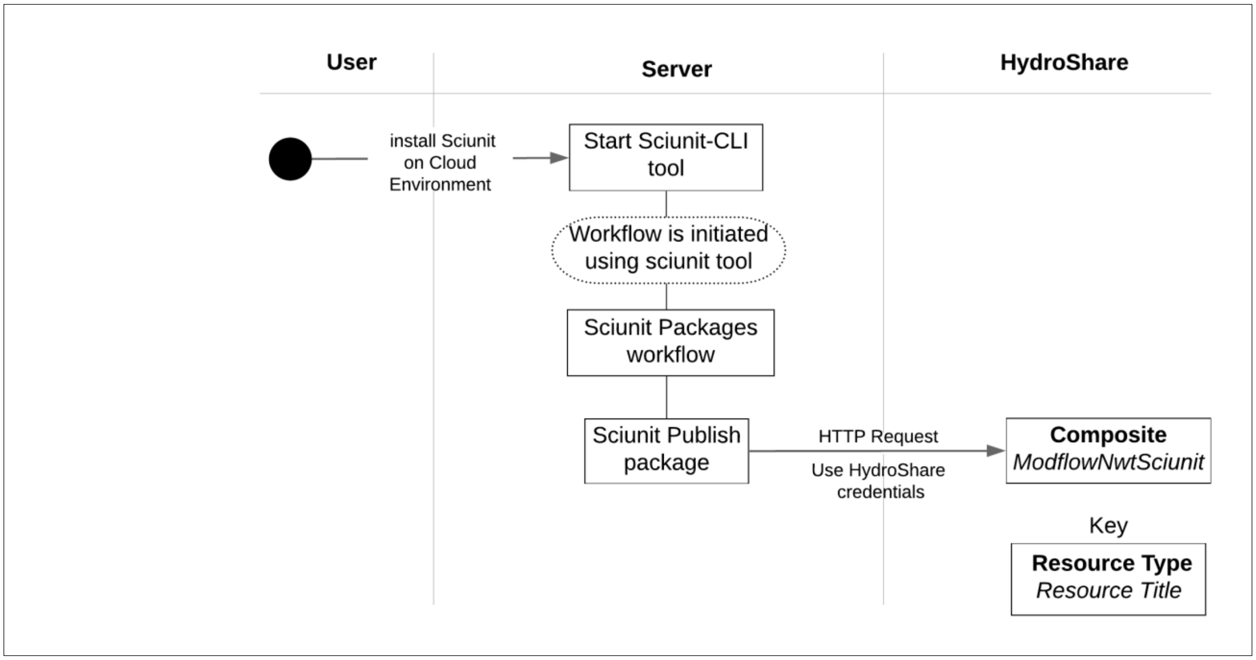


Figure 3 Activity diagram showing creating a sciunit using GeoTrust and publishing that sciunit on HydroShare.

4.1.2. Automating sciunit execution through HydroShare

Integrating the cloud-based sciunit execution from the HydroShare user interface was done using a HydroShare web-app. This web-app directs Hyper Text Transfer Protocol (HTTP) request to a web server where sciunits can be executed. The web-app configured to run a particular sciunit can be accessed through the "Open with" button on the landing page for the resource that stores the raw input data. When the scientist clicks on the web-app button from the "Open with" menu, an HTTP request containing the raw input data's resource ID will be sent to the server. With the resource ID, the HydroShare REST API can be used to download the raw input data and the sciunit to the server. The server can then execute the sciunit using the raw data, and return the output to the scientist as a new HydroShare resource.

Figure 4 shows the steps done in a generic form for the integration between the two cyberinfrastructures, GeoTrust and HydroShare, to improve reproducibility by automating the execution of the published sciunit. The figure shows how the "Open with" app will perform a HTTP GET request to a remote server, which has already been configured with the *Sciunit-CLI*. This automation process is done using a Python script created on the web server machine. This Python script uses the flask library to act as a web server with NGINX (https://www.nginx.com/) used as a proxy to forward all HTTP requests from the user browser to the Python script, which can handle multiple users simultaneously. The Python script is using the POST request to create a new resource and upload the output generated from running the sciunit on this resource. Simultaneously, a webserver is running on the remote machine, which handles the HTTP request and automatically executes a Python script. This script uses the HydroShare user authentication to download the input data from the resource and downloads the Composite resource that includes the sciunit container. Once both resources are downloaded, the resources are unzipped and moved to the working directory for the analysis. The *Sciunit-CLI* executes the downloaded sciunit package. After the sciunit is executed, a new resource is created in HydroShare and the output from the *Sciunit-CLI* execution is uploaded into this new resource. A new collection resource is also created on HydroShare to group all resources that were included during this execution. In this paper we used HydroShare API. Our Python script uses the Python Client Library for the REST API (http://hs-restclient.readthedocs.io/en/latest/).

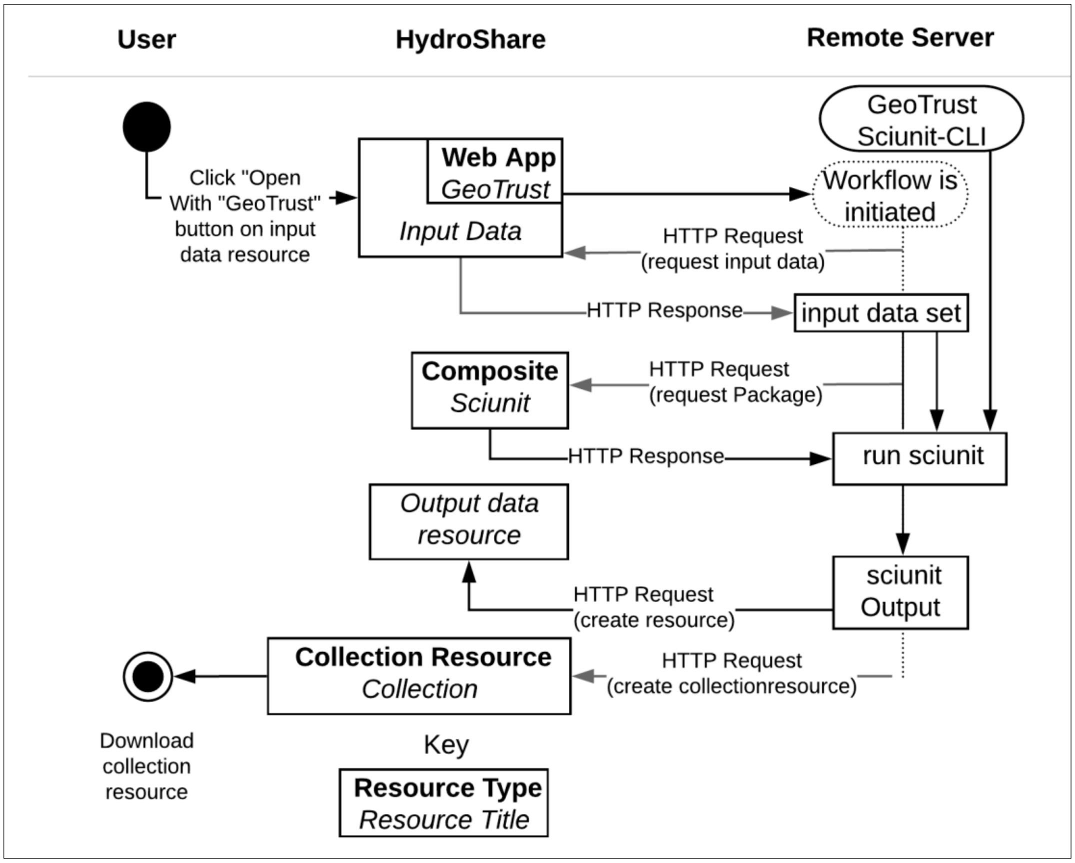


Figure 4 The generic implementation for automating the execution of the published sciunit from the HydroShare web-app

4.2. Use Case Application

A use case application was designed to demonstrate the integration of GeoTrust *Sciunit-CLI* and HydroShare. This integration allows GeoTrust to package and publish a sciunit through HydroShare, after which HydroShare automates the execution of this sciunit. Execution of the packaged sciunit through HydroShare was demonstrated using EC2 instances from Amazon Web Services (AWS). A Linux-based, micro-sized machine (t2) was used for prototyping and demonstration purposes; this machine had 1 Gb of memory, 1 vCPU, 32 Gb of Solid State Drive (SSD)-based local instance storage, and a 64-bit platform ("Amazon EC2 Instances," 2015). This use case consisted of a workflow used for preprocessing model input data, running a computational model, and handling the model outputs. The computational model used for the use case was MODFLOW-NWT.

4.2.1. MODFLOW-NWT Use Case

MODFLOW-NWT is a standalone version of MODFLOW, a commonly used groundwater model (Niswonger et al., 2011). The concept of "packages" is key to the modularity of the different versions of MODFLOW (including MODFLOW-NWT); packages are input files that define some individual component of the groundwater-flow conceptual model or specify the solution method used for the flow equation that is collectively formulated from the individual components. For example, the basic (BAS) and discretization (DIS) packages define the spatial and temporal framework of the model, including the grid dimensions and the location of active and inactive grid cells, while the recharge (RCH) package defines the spatial-distribution and rate of recharge to the water-table. For our use case using MODFLOW-NWT, the Newton-Raphson (NWT) package defines the variables required to implement the Newton-Raphson solution method.

For this study, MODFLOW-NWT was used to simulate the shallow groundwater flow in the James River watershed upstream of Richmond, VA, USA. The model includes recharge to the water table, subsurface flow through the saturated zone, and base-flow discharge to surface water bodies including the James, Rivanna, and Hardware Rivers and several smaller-order streams. Depth-integrated effective transmissivity was assumed to be constant throughout the active model area and spatially-distributed recharge was derived from the national recharge dataset developed by Reitz et al. (2017). Base-flow discharge was simulated using the MODFLOW drain (DRN) package with all drain elevations (i.e., the water-table elevation required to discharge base-flow to a receiving stream) extracted from the National Elevation Dataset. The model runs to completion and is unconstrained by calibration; as such it is to be only used as an example for the workflow processes described in this paper (i.e., no hydrologic or management conclusions were drawn from the results of the model). This workflow could be extended to include calibration (Figure 1). For example, a HydroShare resource for a parameter estimation program such as PEST (Doherty and Hunt, 2010) could be created and included in the sciunit container. Similarly, the pre-processing script could include data retrieval from web services such as the USGS water services API (https://waterservices.usgs.gov/) and the automated generation of PEST input files.

The FloPy library was used to create the MODFLOW-NWT model from raw input datasets (Bakker et al., 2016). FloPy is a library of Python modules that allows scripting of the various steps in MODFLOW model development, execution, and analysis. By combining FloPy with GeoTrust and HydroShare, the workflow used to create and execute MODFLOW model (e.g., the steps shown in Figure 1) can be stored within a reproducible container with descriptive metadata in HydroShare.

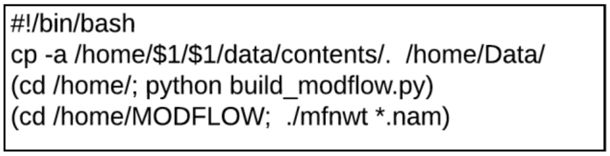
5. Results

5.1. System Implementation

The system was implemented using the following steps. First, the script downloads raw input data and the sciunit resources from HydroShare. Second, the script will unzip both the data and sciunit, pass the data to the sciunit as an argument (this is how the sciunit accepts the input data), and then run the sciunit with the downloaded data. Last, after the execution is completed, the Python script will upload the results to HydroShare by using a POST request to create two new resources: one for the sciunit output, which has the MODFLOW-NWT Model Instance Resource type, and the other the collection resource that will include all the resources used within the study. The script then returns the command status (including any errors) to the user.

5.2. Use Case Results

A digital workflow (bash script) was packaged into a sciunit using the *Sciunit-CLI* tool. The digital workflow runs a Python script to prepare the MODFLOW-NWT input data files and then executes a single run of the model. Figure 5 shows the component of the packaged digital workflow.



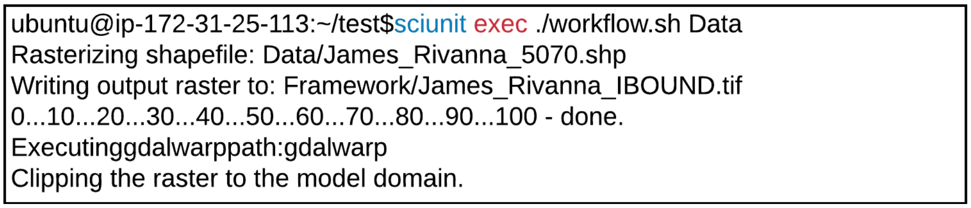
```
#!/bin/bash
cp -a /home/$1/$1/data/contents/.  /home/Data/
(cd /home/; python build_modflow.py)
(cd /home/MODFLOW;  ./mfnwt *.nam)
```

Figure 5 component of the packaged digital workflow.

Figure 6 outlines the first steps taken in the process to start and create a new sciunit through the GeoTrust *Sciunit-CLI* tool for the example workflow while Figure 7 shows the execution and packaging of the digital workflow into a sciunit package. This package command traces all dependencies for the workflow and includes them in a single Docker file. Figure 8 shows how the *publish* command is used to publish a sciunit package on HydroShare. If this is the user's first time connecting to HydroShare, *Sciunit-CLI* will ask for HydroShare user credentials, otherwise the credentials stored will be used. Once the package is published, metadata can be provided by the user via the HydroShare Graphical User Interface (GUI). Future implementations of the *Sciunit-CLI* may expand this functionality by automatically populating more detailed metadata for describing resources.

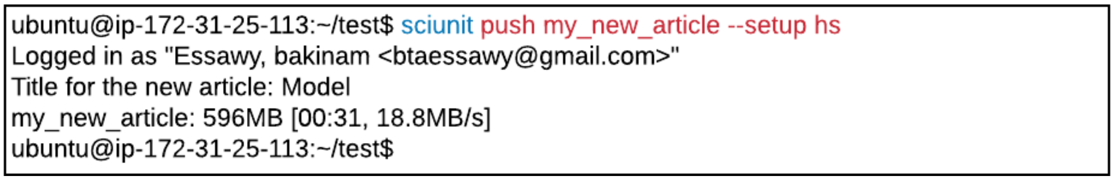
```
ubuntu@ip-172-31-25-113:~/test$sciunit create Model
Opened empty sciunit at /home/ubuntu/sciunit/Model
```

Figure 6 The creation of a new sciunit through the GeoTrust *Sciunit-CLI* tool for the use case



```
ubuntu@ip-172-31-25-113:~/test$sciunit exec ./workflow.sh Data
Rasterizing shapefile: Data/James_Rivanna_5070.shp
Writing output raster to: Framework/James_Rivanna_IBOUND.tif
0...10...20...30...40...50...60...70...80...90...100 - done.
Executinggdalwarppath:gdalwarp
Clipping the raster to the model domain.
```

Figure 7 Execution of the use case workflow through sciunit to create a package

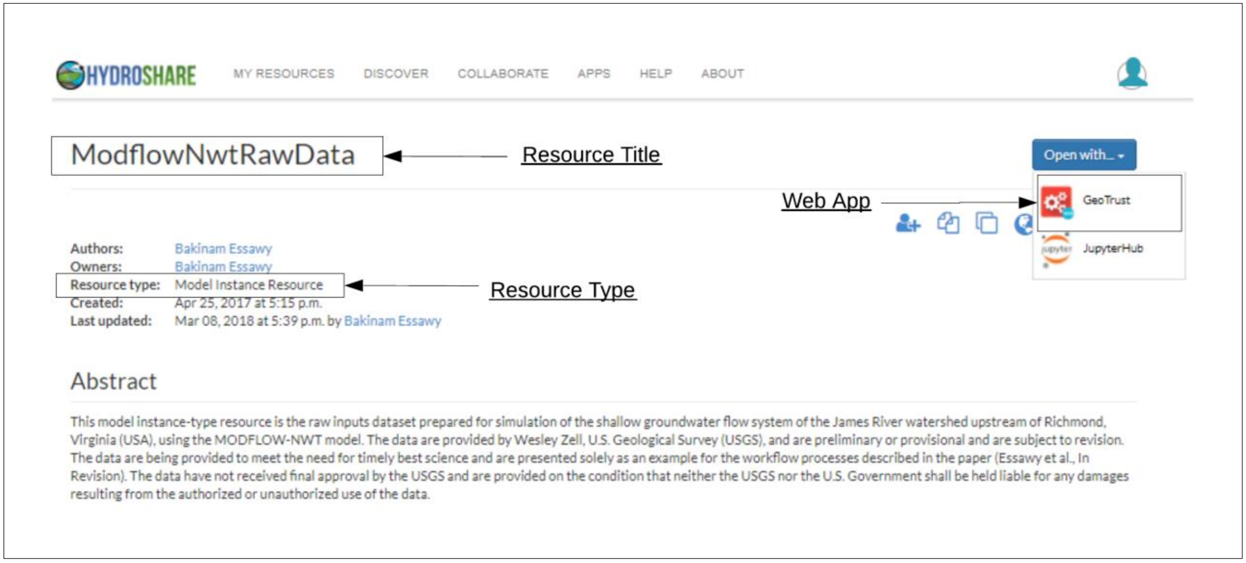


```
ubuntu@ip-172-31-25-113:~/test$ sciunit push my_new_article --setup hs
Logged in as "Essawy, bakinam <btaessawy@gmail.com>"
Title for the new article: Model
my_new_article: 596MB [00:31, 18.8MB/s]
ubuntu@ip-172-31-25-113:~/test$
```

Figure 8 Publishing the use case sciunit to HydroShare

The newly created resource on HydroShare is a Composite Resource Type. This resource type allows the resource to include multiple files without file format limitations and with metadata associated at a file level within the resource. The Composite resource contains two files. The first is the provenance metadata file created while packaging the workflow; this metadata file contains information concerning the creation and version history of the managed data. The second file is the zipped package for the sciunit itself.

Once the sciunit is available as a HydroShare resource, HydroShare's integration with third-party web apps is used to execute the sciunit. In order to store data and make it accessible to be used as the input required by the sciunit, we made a new model instance-type resource titled "ModflowNwtRawData" (Essawy, 2018b). We also created a web-app resource titled "GeoTrust" (Essawy, 2018a). This web-app pointed to the AWS-EC2 instance where the *Sciunit-CLI* tool and our Python script were installed. The connection between the HydroShare resource and the web server was made by providing the web server's URL as the "App-launching URL Pattern" metadata term in the resource. The GeoTrust web-app resource is linked to the ModflowNwtRawData resource by the SupportedResourceType metadata property. This metadata property was set to include the Composite Resource Type, which allowed the web-app to appear in a drop-down list in the "Open with" menu on the ModflowNwtRawData resource landing page. Figure 9 shows the Model Instance Resource type that includes the raw data, and the web apps linked to this resource type to automate the sciunit execution. When the GeoTrust web-app on this page is selected, the HTTP request is sent to server and the workflow is executed. The output is written back to HydroShare as a new resource with the MODFLOW Model Instance Resource type. This resource type is used because the resource can be executed by a MODFLOW model program and it allows for adding extended metadata specific to MODFLOW (Morsy et al., 2017).



ModflowNwtRawData

Authors: Bakinam Essawy
Owners: Bakinam Essawy
Resource type: Model Instance Resource
Created: Apr 25, 2017 at 5:15 p.m.
Last updated: Mar 08, 2018 at 5:39 p.m. by Bakinam Essawy

Abstract

This model instance-type resource is the raw inputs dataset prepared for simulation of the shallow groundwater flow system of the James River watershed upstream of Richmond, Virginia (USA), using the MODFLOW-NWT model. The data are provided by Wesley Zell, U.S. Geological Survey (USGS), and are preliminary or provisional and are subject to revision. The data are being provided to meet the need for timely best science and are presented solely as an example for the workflow processes described in the paper (Essawy et al., In Revision). The data have not received final approval by the USGS and are provided on the condition that neither the USGS nor the U.S. Government shall be held liable for any damages resulting from the authorized or unauthorized use of the data.

Figure 9 The raw data within the Model Instance Resource type, and the web apps linked to this resource type to automate the sciunit execution.

Figure 10 presents the activity diagram for the steps that occur when the "Open with" button is clicked and the "GeoTrust" app is selected on the ModflowNwtRawData resource landing page. The "GeoTrust" app will perform an HTTP GET request to the AWS-EC2 machine, which has already been configured with the *Sciunit-CLI*. The webserver running on the AWS-EC2 machine handles the HTTP request and automatically executes a Python script. The script uses the

HydroShare user authentication to download both the raw data of the ModflowNwtRawData resource and the sciunit container included within the ModflowNwtSciunit resource (Essawy, 2018c). Once the ModflowNwtSciunit and the ModflowNwtRawData resources are downloaded, the script unzips the resources and moves them to the working directory for the analysis. The *Sciunit-CLI* tool executes the downloaded sciunit package, which pre-processes the raw input data for the model and executes the MODFLOW-NWT model. After the model is executed, a new resource is created in HydroShare with the MODFLOW Model Instance Resource type named ModflowNwtSciunitOutput (Essawy, 2018d) and the output from the *Sciunit-CLI* execution is uploaded into this new resource. A new collection resource is also created on HydroShare to group all the resources: the ModflowNwtRawData generic Model Instance Resource (the resource type is a generic model instance because the data uploaded have no specific metadata or format that could be tied to a specific resource type), the web-app GeoTrust resource, the ModflowNwtSciunit, MODFLOW Model Instance Resource, the ModflowNwtSciunit Composite resource, and the ModflowNwtSciunitOutput resource that includes the output resulting from executing the sciunit package.

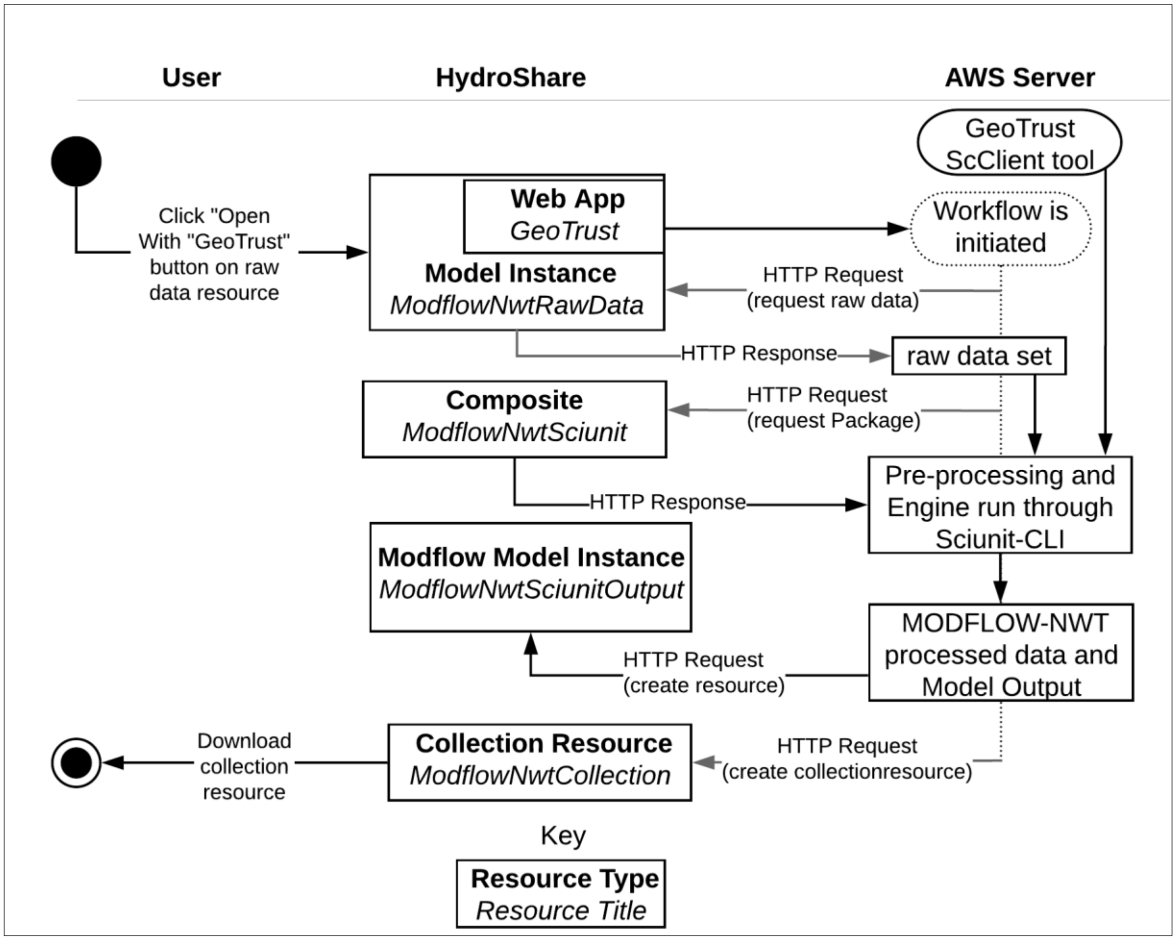
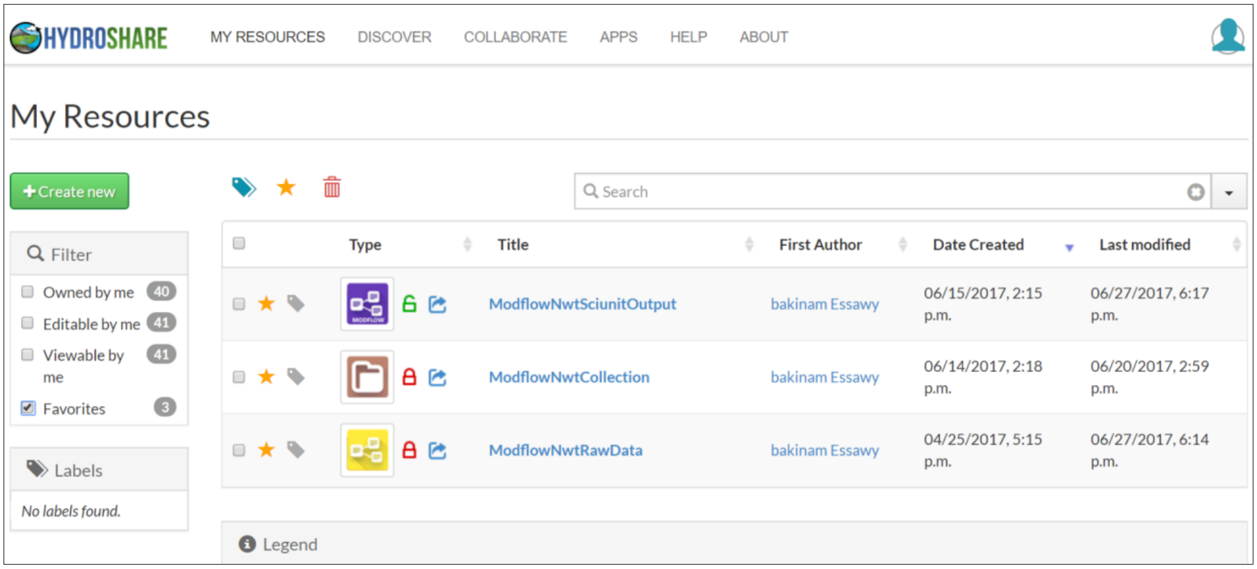


Figure 10 Activity diagram showing the steps for the online execution of the sciunit through HydroShare.

Figure 11 shows HydroShare user "My Resources page" after using the "Open with" action button on the GeoTrust web-app on the ModflowNwtRawData resource for the online execution. Two new resources are created. The first resource in the workflow is the ModflowNwtSciunitOutput resource, which includes the input files for the MODFLOW-NWT model program that are prepared through the preprocessing script and the output from the model run. This resource is given the MODFLOW Model Instance Resource type, because the resource has the inputs that are required by the MODFLOW-NWT model. This resource type allows for extended metadata specific to a MODFLOW model instance. The second resource created is the

ModflowNwtCollection resource (Essawy, 2018e), which includes all the resources used in the online execution for the MODFLOW-NWT. This provides a grouping of resources used for an analysis and allows the user to share or download this collection of resources more easily.



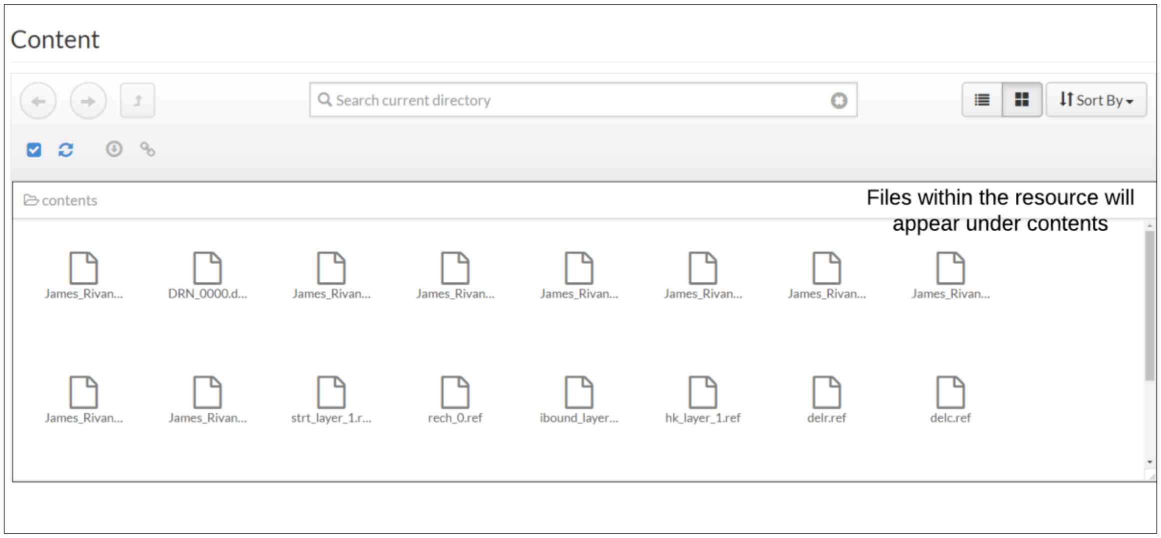
☐ Owned by me 40
☐ Editable by me 41
☐ Viewable by me 41
☑ Favorites 3

No labels found.

☐	Type	Title	First Author	Date Created	Last modified
☐ ★		ModflowNwtSciunitOutput	bakinam Essawy	06/15/2017, 2:15 p.m.	06/27/2017, 6:17 p.m.
☐ ★		ModflowNwtCollection	bakinam Essawy	06/14/2017, 2:18 p.m.	06/20/2017, 2:59 p.m.
☐ ★		ModflowNwtRawData	bakinam Essawy	04/25/2017, 5:15 p.m.	06/27/2017, 6:14 p.m.

Figure 11 HydroShare user My Resources page after using the GeoTrust web app for the online execution.

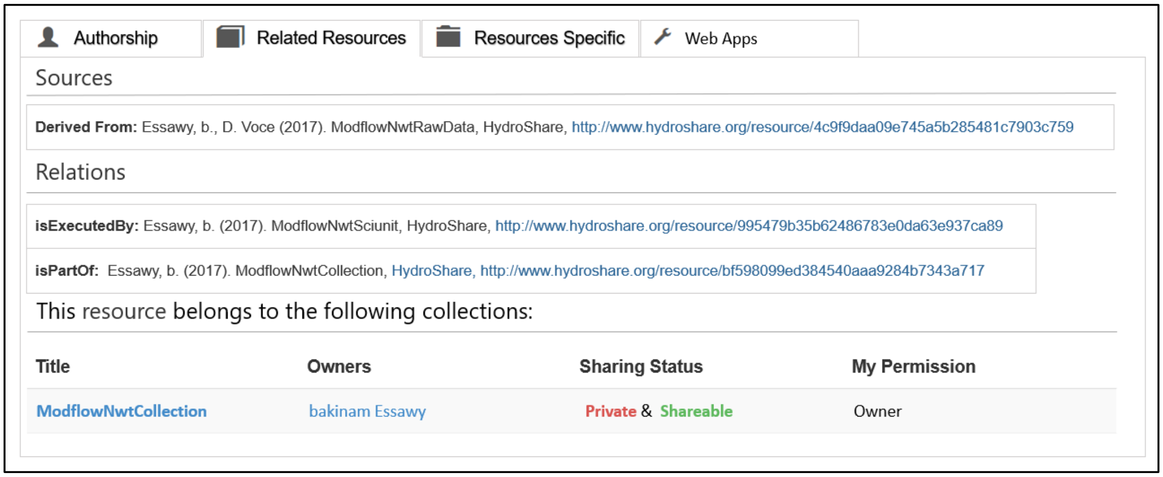
Figure 12 shows the output files within ModflowNwtSciunitOutput resource as viewed on this resource's HydroShare landing page. The resource contains the output generated from running the sciunit that prepares the model input for MODFLOW-NWT and the output from running the MODFLOW-NWT model program itself. The MODFLOW Model Instance Resource type includes extended metadata terms specific for MODFLOW. In this use case the model has eight packages. In addition to the packages already described, this model instance includes: the output control (OC) package, which specifies how the model output is written; the upstream-weighting (UPW) groundwater flow package, which describes the system properties (e.g., transmissivity/conductivity); and the one output listing file (LIST), which contains all the information about the current run (e.g., stress period, time step and the number of active and inactive cells, the recharge, drains, and any errors). The name file (NAM) specifies the name of the input and output files for the model instance.



Files within the resource will appear under contents

Figure 12 The output files within the ModflowNwtSciunitOutput resource landing page in HydroShare.

Additional metadata associated with the MODFLOW output resource is divided into four categories: 1) Authorship, 2) Related resources, 3) Resource Specific, and 4) Web Apps. Figure 13 shows the "Related Resources" metadata. Here all resources linked to the MODFLOW output resource through formal relationships are listed. In this case, the MODFLOW output resource is linked to the ModflowNwtRawData resource through the "Derived From" relationship and to the MODFLOW-NWT resource through the "isExecutedBy" relationship. Figure 14 shows the "Resource Specific" metadata. These are non-null metadata terms that apply only to the MODFLOW Model Instances' such as grid attributes, solver, and boundary condition package choices. Additional metadata terms not previously populated by the user can be populated later within the edit mode and will appear in this section once populated.



Sources

Derived From: Essawy, b., D. Voce (2017). ModflowNwtRawData, HydroShare, http://www.hydroshare.org/resource/4c9f9daa09e745a5b285481c7903c759

Relations

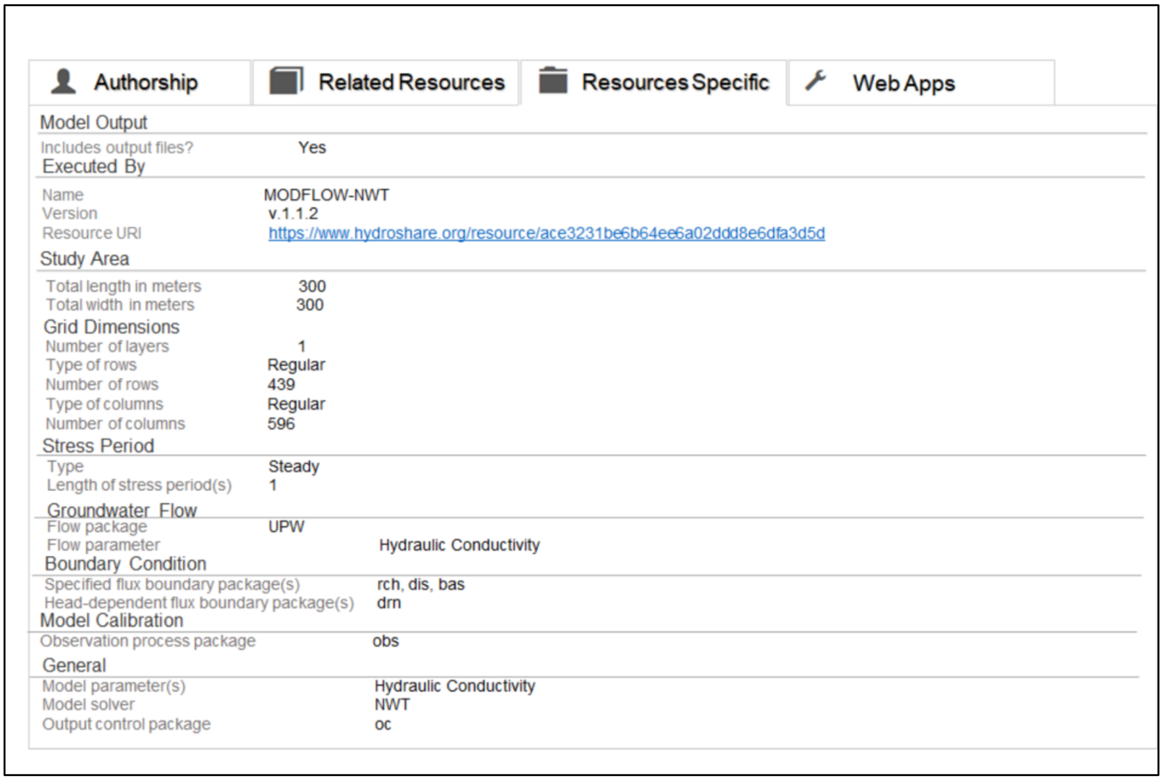
isExecutedBy: Essawy, b. (2017). ModflowNwtSciunit, HydroShare, http://www.hydroshare.org/resource/995479b35b62486783e0da63e937ca89

isPartOf: Essawy, b. (2017). ModflowNwtCollection, HydroShare, http://www.hydroshare.org/resource/bf598099ed384540aaa9284b7343a717

This resource belongs to the following collections:

Title	Owners	Sharing Status	My Permission
ModflowNwtCollection	bakinam Essawy	Private & Shareable	Owner

Figure 13 The ModflowNwtSciunitOutput Related Resources metadata tracking the resource's provenance within HydroShare.



Model Output

Includes output files?	Yes

Executed By

Name	MODFLOW-NWT
Version	v.1.1.2
Resource URI	https://www.hydroshare.org/resource/ace3231be6b64ee6a02ddd8e6dfa3d5d

Study Area

Total length in meters	300
Total width in meters	300

Grid Dimensions

Number of layers	1
Type of rows	Regular
Number of rows	439
Type of columns	Regular
Number of columns	596

Stress Period

Type	Steady
Length of stress period(s)	1

Groundwater Flow

Flow package	UPW
Flow parameter	Hydraulic Conductivity

Boundary Condition

Specified flux boundary package(s)	rch, dis, bas
Head-dependent flux boundary package(s)	drn

Model Calibration

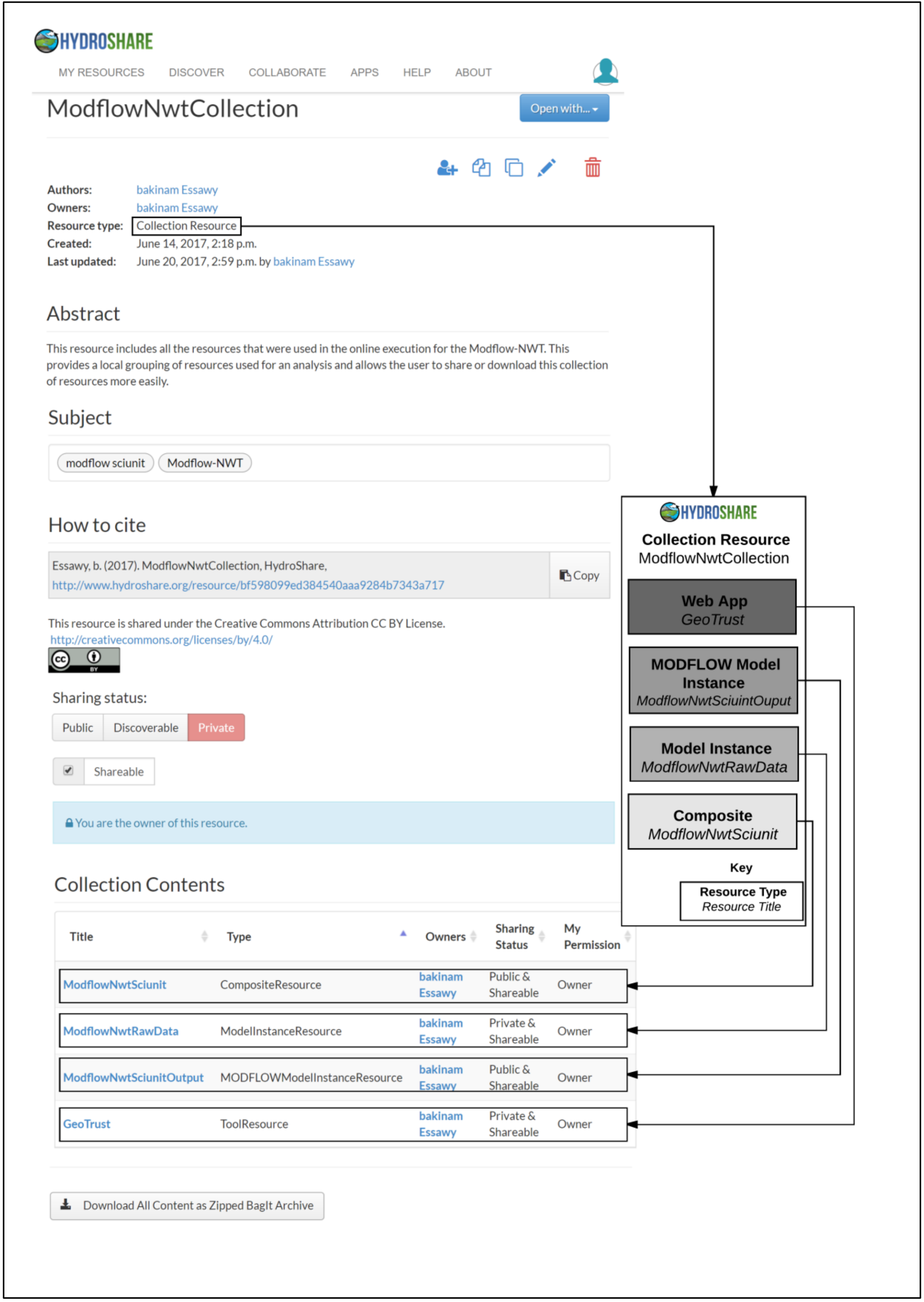
Observation process package	obs

General

Model parameter(s)	Hydraulic Conductivity
Model solver	NWT
Output control package	oc

Figure 14 ModflowNwtSciunitOutput specific metadata capturing key MODFLOW model properties.

Figure 15 shows details for the resulting ModflowNwtCollection resource as viewed on this resource's landing page. The collection resource contains four sub-resources: 1) the ModflowNwtRawData resource with the raw input data ready to be prepared for the MODFLOW-NWT model engine; 2) the ModflowNwtSciunit resource with the sciunit pre-processing workflow, which also includes running the MODFLOW-NWT model; 3) the ModflowNwtSciunitOutput resource, which stores the output generated from running the sciunit workflow; and 4) the GeoTrust web app used to perform the online model execution using AWS-EC2. By organizing all these resources into a single collection, it is possible to have one landing page where users can, referring back to the stated goals in the introduction of this paper, view, obtain, and execute (1) raw initial datasets, (2) data preprocessing scripts used to clean and organize the data, (3) model inputs, (4) model results, and (5) the specific model code along with of all its dependencies used for a computational analysis.



ModflowNwtCollection

Authors: bakinam Essawy
Owners: bakinam Essawy
Resource type: Collection Resource
Created: June 14, 2017, 2:18 p.m.
Last updated: June 20, 2017, 2:59 p.m. by bakinam Essawy

Abstract

This resource includes all the resources that were used in the online execution for the Modflow-NWT. This provides a local grouping of resources used for an analysis and allows the user to share or download this collection of resources more easily.

Subject

modflow sciunit | Modflow-NWT

How to cite

Essawy, b. (2017). ModflowNwtCollection, HydroShare, http://www.hydroshare.org/resource/bf598099ed384540aaa9284b7343a717

This resource is shared under the Creative Commons Attribution CC BY License.
http://creativecommons.org/licenses/by/4.0/

Sharing status:

You are the owner of this resource.

Collection Contents

Title	Type	Owners	Sharing Status	My Permission
ModflowNwtSciunit	CompositeResource	bakinam Essawy	Public & Shareable	Owner
ModflowNwtRawData	ModelInstanceResource	bakinam Essawy	Private & Shareable	Owner
ModflowNwtSciunitOutput	MODFLOWModelInstanceResource	bakinam Essawy	Public & Shareable	Owner
GeoTrust	ToolResource	bakinam Essawy	Private & Shareable	Owner

Figure 15 The collection resource that includes all resources used within the study.

6. Discussion and Conclusions

In this paper, we demonstrated how HydroShare and GeoTrust can be integrated to easily and efficiently package, share, and publish model workflows. MODLFOW-NWT was used as an example application to demonstrate the functionality provided by these cyberinfrastructures for creating open, reusable data analysis and cloud-based model execution services. The approach showed how containers built using GeoTrust tools can be shared as HydroShare resources. A cloud-based service was created to automatically retrieve raw input data from HydroShare, execute a sciunit container that both prepares and runs a MODFLOW-NWT model, and share the results on HydroShare using a MODFLOW Model Instance Resource type. All the resources are aggregated in HydroShare into one collection resource with domain-specific metadata.

The integration of scientific cyberinfrastructures such as the HydroShare and GeoTrust projects can improve reproducibility in computational hydrology. New MODFLOW models can be directly built from unprocessed input data (e.g., land-surface DEMs or stream-network shapefiles) by running a sciunit container that includes automated data preparation steps implemented using the FloPy Python package. The container is run online using AWS resources initiated directly through the HydroShare user interface. A particular advantage of this approach is that the GeoTrust *Sciunit-CLI* tool provides scientists a method for efficiently creating containers for script-driven modeling workflows. Thus, the general approach demonstrated here for the MODFLOW-NWT use case could be applied for any workflow that can be automated and that is compatible with Docker requirements. For example, in prior work we have constructed pre- and post-processing workflows for the Variable Infiltration Capacity (VIC) hydrologic model (Liang et al., 1996) that could directly benefit from this method for packaging, sharing, and publishing resources (Billah et al., 2016; Essawy et al., 2016). These containers are efficient, lightweight, self-contained packages of computational experiments that can be repeated or reproduced regardless of deployment configurations.

In addition to integration with HydroShare for storing and publishing a sciunit, cloud resources were used to execute sciunits directly through the HydroShare user interface. While only AWS was presented, we evaluated as part of this work three different cloud computing services: EarthCube Integration and Testing Environment (ECITE), CyVerse, and Amazon Web Services (AWS). ECITE and CyVerse are funded by NSF and both are under active development. One main advantage for using ECITE or CyVerse is that they are free of charge for scientific studies. AWS, though not free, does offer a competitive grant program for researchers. From our experience, the AWS platform made the process of obtaining computer resources the simplest when compared to ECITE and CyVerse. The AWS user simply logs in to the console, selects the type of the machine needed, and launches it. When using ECITE, we had to contact the developer and ask for an instance with the required specifications and a short paragraph summarizing the project we are working on to justify the allocation of compute resources. We also needed to contact the developer each time we wanted to open a port (e.g., port 22 to SSH or port 80 for HTTP). The service did not support Elastic IPs like AWS, so each time we restarted an instance and wanted to use SSH to access to the machine, we needed to report the IP address used to access the machine to the developer to add this address to the security rules. CyVerse is a more mature service, but allows each user only a certain allocation of computational time. Once the user exceeds this allocation the instance is suspended and the user needs to request more time from the administrators. This feature was problematic for our use case of a continually available cloud-based resource for online model execution. For these reasons, we used AWS-EC2 for much of the testing work described in this paper, but ECITE and CyVerse are in active development and will likely be good options for this use case in the future.

While this approach shows great promise, it is not without limitations: (1) the *Sciunit-CLI* tool must be installed in order to re-execute a sciunit container and (2) HydroShare lacks methods for uniquely identifying and managing web-app resources that will be needed as the number of these resources continues to increase. Regarding the latter limitation, without a more organized structure, naming conflicts could cause confusion when using the "Open with" button over which app is to be requested. Also, this work does not fully explore computational challenges associated with the proposed methodology. Using cloud services like AWS provides the opportunity for scalability as more users are added. For example, this solution used small EC2 instances for prototyping. Future work could explore AWS EC2 Container Service (ECS) as an alternative for a more scalable solution to support multiple concurrent users. Data movement between HydroShare and AWS is another potential issue as data volumes increase, which is not uncommon for hydrologic modeling. HydroShare is built on iRODS (Integrated Rule-Oriented Data System), which includes the ability to interface with AWS S3 storage resources. Future work could explore using this functionality to automate the movement of large files between HydroShare and AWS to support computation within AWS and still maintain access through the HydroShare user interface. iRODS is specifically designed to handle such data federation needs and should provide a robust solution for managing the large data flows common in hydrologic modeling. Lastly, future work should explore scaling of the general approach presented here to use cases in which multiple sciunits are available for execution within a remote, cloud-based resource. In this case, a user could select from available sciunits to process input data stored with HydroShare, making for a potentially very powerful general approach applicable to many different modeling and analysis use cases that require remote data processing.

7. Acknowledgements and Disclaimer

We gratefully acknowledge the National Science Foundation for support of this work under awards ACI-0940841, ICER-1343800, and ICER-1440323. Any use of trade, firm, or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government.

8. References

Amazon EC2 Instances [WWW Document], 2015. URL http://aws.amazon.com/ec2/instance-types/ (accessed 6.7.15).

Bakker, M., Post, V., Langevin, C.D., Hughes, J.D., White, J.T., Starn, J.J., Fienen, M.N., 2016. Scripting MODFLOW Model Development Using Python and FloPy. Groundwater 54, 733–739. https://doi.org/10.1111/gwat.12413

Billah, M.M., Goodall, J.L., Narayan, U., Essawy, B.T., Lakshmi, V., Rajasekar, A., Moore, R.W., 2016. Using a data grid to automate data preparation pipelines required for regional-scale hydrologic modeling. Environ. Model. Softw. 78, 31–39. https://doi.org/10.1016/j.envsoft.2015.12.010

Borgman, C.L., 2012. The conundrum of sharing research data. J. Am. Soc. Inf. Sci. Technol. 63, 1059–1078.

David, C.H., Famiglietti, J.S., Yang, Z.-L., Habets, F., Maidment, D.R., 2016. A decade of RAPID—Reflections on the development of an open source geoscience code. Earth Sp. Sci. 226–244. https://doi.org/10.1002/2014EA000014.Received

Doherty, J.E., Hunt, R.J., 2010. Approaches to highly parameterized inversion-A guide to using PEST for groundwater-model calibration. US Geol. Surv. Sci. Investig. Rep.

Essawy, B., 2018a. GeoTrust,. HydroShare, http://www.hydroshare.org/resource/126701df868e4da9872d9b533db34ae6.

Essawy, B., 2018b. ModflowNwtRawData, HydroShare, http://www.hydroshare.org/resource/4c9f9daa09e745a5b285481c7903c759.

Essawy, B., 2018c. ModflowNwtSciunit. HydroShare, http://www.hydroshare.org/resource/995479b35b62486783e0da63e937ca89.

Essawy, B., 2018d. ModflowNwtSciunitOutput, HydroShare, http://www.hydroshare.org/resource/19605cf6e91e415fb98b7a28cad263d6.

Essawy, B., 2018e. ModflowNwtCollection, HydroShare, http://www.hydroshare.org/resource/bf598099ed384540aaa9284b7343a717.

Essawy, B.T., Goodall, J.L., Xu, H., Rajasekar, A., Myers, J.D., Kugler, T.A., Billah, M.M., Whitton, M.C., Moore, R.W., 2016. Server-side workflow execution using data grid technology for reproducible analyses of data-intensive hydrologic systems. Earth Sp. Sci. 3, 163–175. https://doi.org/10.1002/2015EA000139

Gil, Y., David, C.H., Demir, I., Essawy, B.T., Fulweiler, R.W., Goodall, J.L., Karlstrom, L., Lee, H., Mills, H.J., Oh, J.-H., Pierce, S.A., Pope, A., Tzeng, M.W., Villamizar, S.R., Yu, X., 2016. Towards the Geoscience Paper of the Future : Best Practices for Documenting and Sharing Research from Data to Software to Provenance. Earth Sp. Sci. 1–75. https://doi.org/10.1002/2015EA000136

Gorgolewski, K.J., Poldrack, R.A., 2016. A Practical Guide for Improving Transparency and Reproducibility in Neuroimaging Research. PLoS Biol. 14, 1–13. https://doi.org/10.1371/journal.pbio.1002506

Hai, D., That, T., Fils, G., Yuan, Z., Malik, T., 2017. Sciunits : Reusable Research Objects. Tech.

Report, DBGroup, Sch. Comput. DePaul Univ.

Horsburgh, J.S., Morsy, M.M., Castronova, A.M., Goodall, J.L., Gan, T., Yi, H., Stealey, M.J., Tarboton, D.G., 2015. Hydroshare: Sharing diverse environmental data types and models as social objects with application to the hydrology domain. JAWRA J. Am. Water Resour. Assoc. 52. https://doi.org/10.1111/1752-1688.12363

Hutton, C., Wagener, T., Freer, J., Han, D., Duffy, C., Arheimer, B., 2016. Most computational hydrology is not reproducible, so is it really science? Water Resour. Res. 50. https://doi.org/10.1002/ 2016WR019285

Liang, X., Lettenmaier, D.P., Wood, E.F., 1996. One-dimensional statistical dynamic representation of subgrid spatial variability of precipitation in the two-layer variable infiltration capacity model. J. Geophys. Res. Atmos. 101(D16), 21403–21422.

Malik, T., 2017. GeoTrust: Improving Sharing and Reproducibility of Geoscience Applications [WWW Document]. EOL Semin. Ser. URL https://www2.ucar.edu/for-staff/daily/announcement-calendar-event/eol-seminar-series-dr-tanu-malik (accessed 6.6.17).

Malik, T., Valescu, C., Pham, Q., 2017. Sciunit, a system for creating, sharing, and running light-weight containers. [WWW Document]. URL http://www.geotrusthub.org/ (accessed 1.1.17).

Meng, H., Kommineni, R., Pham, Q., Gardner, R., Malik, T., Thain, D., 2015. An invariant framework for conducting reproducible computational science. J. Comput. Sci. 9, 137–142. https://doi.org/10.1016/j.jocs.2015.04.012

Morsy, M.M., Goodall, J.L., Castronova, A.M., Bandaragoda, C., Greenberg, J., 2014. Metadata for Describing Water Models, in: In Proceedings of the 7th International Congress on Environmental Modelling and Software, DP Ames, NWT QuinnMorsy, M.M., Goodall, J.L., Castronova, A.M., Bandaragoda, C., Greenberg, J., 2014. Metadata for Describing Water Models, in: In Proceedings of the. pp. 978–988.

Morsy, M.M., Goodall, J.L., Castronova, A.M., Dash, P., Merwade, V., Sadler, J.M., Rajib, M.A., Horsburgh, J.S., Tarboton, D.G., 2017. Design of a metadata framework for environmental models with an example hydrologic application in HydroShare. Environ. Model. Softw. 93, 13–28. https://doi.org/10.1016/j.envsoft.2017.02.028

Niswonger, R.G., Panday, S., Motomu, I., 2011. MODFLOW-NWT , A Newton Formulation for MODFLOW-2005. U.S. Geol. Surv. Tech. Methods 6, 44.

Peng, R.D., 2011. Reproducible research in computational science. Science. 334, 1226–1227.

Pham, Q., Malik, T., Foster, I., 2014. Auditing and maintaining provenance in software packages. Int. Proven. Annot. Work. 97–109.

Piccolo, S.R., Frampton, M.B., 2016. Tools and techniques for computational reproducibility. Gigascience 5, 1–13. https://doi.org/10.1186/s13742-016-0135-4

Qasha, R., Cala, Jacek, Watson, P., 2016. A Framework for Scientific Workflow Reproducibility in the Cloud. IEEE 12th Int. Conf. e-Science 81–90.

Qin, J., Dobreski, B., Brown, D., 2016. Metadata and Reproducibility : A Case Study of Gravitational Wave Research Data. J. Digit. Curation 11, 218–231. https://doi.org/10.2218/ijdc.v11i1.399

Reitz, M., Sanford, W.E., Senay, Gabriel B., and Cazenas, J., 2017. Annual estimates of recharge, quick-flow runoff, and ET for the contiguous US using empirical regression equations, 2000-2013. U.S. Geol. Surv. data release. https://doi.org/https://doi.org/10.5066/F7PN93P0.

Reproducibility Guide - The rOpenSci Project [WWW Document], n.d. . 2017. URL http://ropensci.github.io/reproducibility-guide/sections/introduction/ (accessed 6.16.17).

Santana-Perez, I., Ferreira da Silva, R., Rynge, M., Deelman, E., Pérez-Hernández, M.S., Corcho, O., 2017. Reproducibility of execution environments in computational science using Semantics and Clouds. Futur. Gener. Comput. Syst. 67, 354–367. https://doi.org/10.1016/j.future.2015.12.017

Singh, V.P., Asce, F., Woolhiser, D.A., Asce, M., 2002. Mathematical Modeling of Watershed Hydrology. J. Hydrol. Eng. 7, 270–292.

Stodden, V., 2013. Resolving Irreproducibility in Empirical and Computational Research. IMS Bull. Online.

Swain, N.R., Christensen, S.D., Snow, A.D., Dolder, H., Espinoza-Dávalos, G., Goharian, E., Jones, N.L., Nelson, E.J., Ames, D.P., Burian, S.J., 2016. A new open source platform for lowering the barrier for environmental web app development. Environ. Model. Softw. 85, 11–26. https://doi.org/10.1016/j.envsoft.2016.08.003

Tarboton, D.G., Horsburgh, J.S., Idaszak, R., Heard, J., Valentine, D., Couch, A., Ames, D., Goodall, J.L., Band, L., Merwade, V., Arrigo, J., Hooper, R., Maidment, D., 2014a. a Resource Centric Approach for Advancing Collaboration Through Hydrologic Data and Model Sharing. 11th Int. Conf. Hydroinformatics, HIC 2014, New York City, USA.

Tarboton, D.G., Idaszak, R., Horsburgh, J.S., Heard, J., Ames, D., Goodall, J.L., Band, L., Merwade, V., Couch, A., Arrigo, J., Hooper, R., Valentine, D., Maidment, D., 2014b. HydroShare: Advancing Collaboration through Hydrologic Data and Model Sharing. Int. Environ. Model. Softw. Soc. 7th Int. Congr. Environ. Model. Software, San Diego, California, USA. www. iemss. org/society/index/php/iemss-2014-proceedings. https://doi.org/10.13140/2.1.4431.6801

Weibel, S., Kunze, J., Carl Lagoze, A., Wolf, M., 1998. Dublin core metadata for resource

discovery. No. RFC 2413.