Brigham Young University
BYU ScholarsArchive

Faculty Publications

2016-08-20

A new open source platform for lowering the barrier for environmental web app development

Nathan R. Swain

Scott D. Christensen
United States Army Engineer Research and Development Center

Alan D. Snow
United States Army Engineer Research and Development Center

Herman Dolder

Gonzola Espinoza-Dávalos
UNESCO-IHE

Follow this and additional works at: https://scholarsarchive.byu.edu/facpub

 Part of the Other Civil and Environmental Engineering Commons

See next page for additional authors

Original Publication Citation

Swain, N. R., S. D. Christensen, A. D. Snow, H. Dolder, G. Espinoza-Dávalos, E. Goharian, N. L. Jones, E. J. Nelson, D. P. Ames and S. J. Burian (2016). "A new open source platform for lowering the barrier for environmental web app development."Environmental Modelling & Software 85: 11-26.

BYU ScholarsArchive Citation

Swain, Nathan R.; Christensen, Scott D.; Snow, Alan D.; Dolder, Herman; Espinoza-Dávalos, Gonzola; Goharian, Erfan; Jones, Norman L.; Nelson, E. James; Ames, Daniel P.; and Burian, Steven J., "A new open source platform for lowering the barrier for environmental web app development" (2016). *Faculty Publications*. 4273.
https://scholarsarchive.byu.edu/facpub/4273

This Peer-Reviewed Article is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Faculty Publications by an authorized administrator of BYU ScholarsArchive. For more information, please contact ellen_amatangelo@byu.edu.

Authors

Nathan R. Swain, Scott D. Christensen, Alan D. Snow, Herman Dolder, Gonzola Espinoza-Dávalos, Erfan Goharian, Norman L. Jones, E. James Nelson, Daniel P. Ames, and Steven J. Burian

This peer-reviewed article is available at BYU ScholarsArchive: https://scholarsarchive.byu.edu/facpub/4273

Environmental Modelling & Software 85 (2016) 11–26

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

A new open source platform for lowering the barrier for environmental web app development

Nathan R. Swain [a, *], Scott D. Christensen [b], Alan D. Snow [c], Herman Dolder [a], Gonzalo Espinoza-Dávalos [d], Erfan Goharian [e], Norman L. Jones [f], E. James Nelson [f], Daniel P. Ames [f], Steven J. Burian [g]

[a] Aquaveo, LLC, Provo, UT, USA
[b] Information and Technology Laboratory, United States Army Engineer Research and Development Center, Vicksburg, MS, USA
[c] Coastal and Hydraulics Laboratory, United States Army Engineer Research and Development Center, Vicksburg, MS, USA
[d] Department of Integrated Water Systems and Governance, UNESCO-IHE, Delft, The Netherlands
[e] Department of Land, Air and Water Resource, University of California, Davis, CA, USA
[f] Civil and Environmental Engineering, Brigham Young University, Provo, UT, USA
[g] Civil and Environmental Engineering, University of Utah, Salt Lake City, UT, USA

ARTICLE INFO

Article history:
Received 12 February 2016
Received in revised form
2 August 2016
Accepted 8 August 2016
Available online 20 August 2016

Keywords:
Tethys platform
Web app development
Environmental
Decision support
Hydrologic modeling

ABSTRACT

The interactive nature of web applications or "web apps" makes them a well-suited medium for conveying complex scientific concepts to lay audiences and creating decision support tools that harness cutting edge modeling techniques and promote the work of environmental scientists and engineers. Despite this potential, the technical expertise required to develop web apps represents a formidable barrier—even for scientists and engineers who are skilled programmers. This paper describes four hurdles that contribute to this barrier and introduces an approach to overcoming these hurdles. We present an open source implementation of this approach, a development and hosting environment for environmental web apps called Tethys Platform. Several case studies are provided that demonstrates how the approach, as implemented within Tethys Platform, successfully lowers the barrier to web app development in the environmental domain.

© 2016 Elsevier Ltd. All rights reserved.

1. Introduction

Hydrologic and other types of environmental model simulations are often used in decision making to estimate and analyze watershed responses to specific scenarios (Bhuyan et al., 2003; Goodrich et al., 2008; Lam et al., 2004; Miller et al., 2007; Santhi et al., 2006). However, typical stakeholders and decision makers do not have the technical expertise required to properly configure a simulation for a particular scenario. The process becomes even more daunting for physics-based hydrologic models, because of the challenges of data collection and management of large spatial and temporal datasets.

Environmental web applications or "web apps" can overcome many of the challenges of using hydrologic simulations in decision-making (e.g.: Demir and Krajewski, 2013; Goodrich et al., 2008; Kulkarni et al., 2014; Sun, 2013). In the context of this work we define an environmental web app as a narrowly-focused, web-accessed application for performing common tasks related to environmental modeling. In a web environment, environmental web apps can be hosted on a remote server that can be accessed simultaneously by multiple users via a web interface. This eliminates the need for the end user to procure and maintain the high performance hardware required by the models, deal with issues related to software installation and operating system incompatibilities, or monitor and install software updates. All that is needed to use the web app is an internet connection and a web browser.

Despite the potential of web apps for promoting the work of environmental scientists and engineers, the technical expertise required to develop them represents a significant barrier for would-be developers whose primary background is environmental modeling. The barrier can be characterized by several hurdles a novice developer would need to overcome to successfully develop an environmental web app.

* Corresponding author.
E-mail address: nswain@aquaveo.com (N.R. Swain).

http://dx.doi.org/10.1016/j.envsoft.2016.08.003
1364-8152/© 2016 Elsevier Ltd. All rights reserved.

One hurdle is the need to discover and select software packages that address the spatial needs of environmental web apps. These spatial needs can be addressed using existing free and open source software (FOSS) for geographic information systems (FOSS4G), but the abundance of FOSS4G that are available can be overwhelming to new developers. For example, the Open Geospatial Consortium (OGC), an organization that creates standards for GIS software, publishes a database of over 800 registered products (Open Geospatial Consortium, 2015). Moreover, of the 34 geospatial software packages promoted by the Open Source Geospatial Foundation (OSGeo) 14 are for web mapping applications (includes fully adopted projects and those in "incubation" mode). Swain et al. (2015) addressed this challenge of navigating the daunting open source technology space by performing a review of the state-of-the-art FOSS4G and FOSS for web software packages that have been used in earth science web apps in the literature.

There are many benefits of using FOSS4G packages to address the needs of environmental web apps, but using FOSS4G packages often requires orchestrating the use of more than one package to achieve the desired functionality. FOSS4G packages tend to be more narrowly focused in terms of functionality than their proprietary counterparts. Whereas proprietary software vendors typically offer a wide variety of GIS functionality (e.g. web mapping services, geoprocessing, and spatial storage) in a single software package, FOSS4G packages tend to focus on a single category of functionality (Steiniger and Weibel, 2010). Consequently, creating an environmental web app using FOSS4G usually requires the developer to synthesize several packages and orchestrate their use via code.

Another hurdle that developers encounter is caused by the multi-lingual nature of web app development. Developing a dynamic, interactive web app requires the use of HTML, CSS, and JavaScript for creating the user interface; a scripting language such as PHP, Python, or Ruby for handling logic on the server; and structured query language (SQL) for interacting with a database. Successful web app development also requires the use of a software architectural pattern such as model-view-controller (MVC) or some variant to prevent the source code from becoming unmanageable (Buschmann et al., 1996; Feng et al., 2011; Gamma et al., 1995; Jansson and Moon, 2001; Mason et al., 2014; Walker and Chapra, 2014). We classify the above-mentioned challenges into four major hurdles: (1) the software hurdle, (2) the orchestration hurdle, (3) the web development hurdle, and (4) the deployment hurdle (summarized in Table 1).

Our primary objective in this work was to demonstrate an approach for lowering the barrier for environmental web app development so as to make it a more viable medium for environmental scientists and engineers who have some scientific programming experience. Our approach lowers the barrier to water resource web development by addressing each of the four hurdles by providing: (1) open source software that meets the spatial and computational capabilities commonly required for environmental modeling; (2) a programmatic means to use each of the recommended software tools in a single programming language; (3) a reduction of the web development skills required to develop web apps; and (4) a web-safe mechanism for deploying the finished web apps that is flexible enough to work on the most common hardware (i.e. university cloud, commercial cloud, private data centers). As a means of illustrating this approach, we present an implementation of this approach, a development and hosting environment for environmental web apps called Tethys Platform.

The remainder of this paper is organized as follows. A description of the three primary components of Tethys Platform and their design is presented in Section 2. The capabilities of Tethys Platform are demonstrated in Section 3 with descriptions of several web apps that were developed. A detailed discussion on how the barrier to web app development has been successfully lowered by Tethys Platform is presented in Section 4.

2. Software description

Tethys Platform is a development and hosting environment for environmental web apps. Although it aims to lower the barrier to water resource web app development, web apps developed using Tethys Platform are created programmatically—not using a graphical drag-and-drop type editor. It is targeted at motivated scientists and engineers who have some scientific programming experience, but not necessarily web development experience. We assume that users of Tethys Platform recognize the value of disseminating their work through web app medium, but are either daunted by the prospect of learning web development and/or have little patience for the sometimes tedious task of developing visually appealing web user interfaces. We selected Python, an all-purpose scripting language (Python Software Foundation, 2016) as the programming language of Tethys Platform, because it is relatively easy to learn and it has gained popularity among scientists and engineers in recent years (Millman and Aivazis, 2011; Oliphant, 2007).

Tethys Platform consists of three major components: Tethys Software Suite, Tethys Software Development Kit (SDK), and Tethys Portal. Each component was designed to address one or more of the four hurdles and effectively lower the barrier for development. Tethys Software Suite overcomes the software hurdle by providing suite of 3rd party FOSS and FOSS4G software tools to address many of the common needs encountered in environmental web app development. Tethys SDK addresses both the web development hurdle and the orchestration hurdle by providing a Python MVC framework for streamlined development of the web apps and Python APIs that allow programmatic control over each of the software suite component. Tethys Portal overcomes the deployment hurdle, by providing the primary runtime environment for Tethys Platform web apps. Fig. 1 summarizes the major components of Tethys Platform, which are discussed in more detail in this section.

2.1. Tethys Software Suite

Tethys Software Suite is the component of Tethys Platform that provides access to resources and functionality that are commonly required to develop environmental web apps. The primary motivation of creating the Tethys Software Suite was to address the software hurdle discussed previously. Some of the more specialized needs environmental app must provide arise from the spatial data components of the models that are used in the apps. Distributed

Table 1
Summary of the four challenges identified.

Challenges	Description
Software hurdle	Selecting from many available FOSS and FOSS4G to provide the spatial and computing capabilities required by environmental web apps.
Orchestration hurdle	Synthesize multiple FOSS and FOSS4G to provide a broad range of capabilities.
Web development hurdle	Learn multiple languages such as HTML, CSS, JavaScript, Python, PHP, Ruby and code management approach such as model-view-controller.
Deployment hurdle	Deploy the completed web apps in a safe and secure way on varying hardware and data center environments.

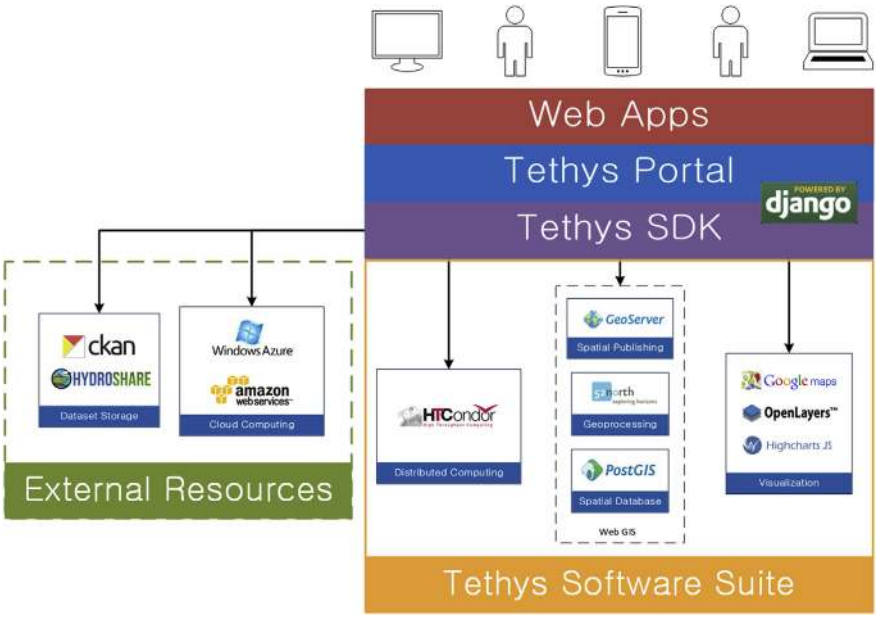

Fig. 1. The component diagram for Tethys Platform.

hydrologic models, for example, are parameterized using raster or vector layers such as land use maps, digital elevation models, and rainfall intensity grids. Consequently, a majority of the 3rd party software tools included in the software suite are web GIS products that can be used to acquire, modify, store, visualize, and analyze spatial data. We have selected GIS software that implement OGC standards in an effort to prevent a dependence on any one software tool. Though only four of the applicable software packages were included in Tethys Platform at the time of writing this paper, eventually it may include support for many more of the primary software that provide OGC standard implementations. Tethys Software Suite includes other 3rd party software tools to address computing and visualization needs of environmental web apps.

The software packages that have been included as part of Tethys Software Suite are the product of open source projects meaning they are governed by an open source license. For convenience, the license of each software described in this section is included as part of its description. Open source licenses not only guarantee free access to the code, but also include the freedom to freely redistribute the software as part of other software distributions where the license of the software distribution does not conflict with the terms of the open source license (Open Source Initiative, 2016). This feature of FOSS is what allows Tethys Platform the right to redistribute the software in Tethys Software Suite.

The following sections describe the included software in terms of the functionality they provide to environmental web app developers. The last section describes the strategy that is taken to reduce the burden of installing the many requisite software packages, which is a side effect of providing a suite of software to address the software hurdle.

2.1.1. Spatial database

We elected to include PostgreSQL (PostgreSQL License) database with PostGIS (General Public License) spatial database extension (Holl and Plum, 2009; Nguyen, 2009) to provide spatial data storage capabilities for Tethys web apps, because it had the most features of the open source spatial database implementations and it is the preferred open-source spatial database solution for earth science web applications (Swain et al., 2015). PostGIS adds spatial field types to the PostgreSQL database including raster, geometry, and geography as well as database functions for basic analysis of GIS objects and coordinate transformation. It also provides an extensive implementation of the applicable OGC standards (Steiniger and Hunter, 2013).

2.1.2. Geoprocessing

52°North WPS (General Public License) is included in Tethys software suite as one means for supporting geoprocessing needs in environmental web app development. 52°North WPS is a full open-source implementation of the OGC-WPS standard ((52 °North, 2014); Schut and Whiteside, 2007) and was selected over other similar implementations, because it provides the most "out-of-the-box" geoprocessing capabilities (Swain et al., 2015) including support for GRASS (GRASS Development Team, 2014), Sextante (Olaya and Gimenez, 2011), and ArcGIS® Server integration (ESRI, 2004). 52°North WPS also allows developers to publish custom Python (Python Software Foundation, 2016) and R (Chambers, 2013) scripts as web services.

PostGIS provides a second option for performing geoprocessing capabilities on data that are stored in spatially-enabled databases. PostGIS includes a library of SQL functions that can be used to perform geoprocessing operations on raster and vector types. It includes functions for vectorizing rasters, clipping rasters with vectors, and running stats on rasters by geometric region (Holl and Plum, 2009).

2.1.3. Map rendering

GeoServer (General Public License) is included for publishing spatial data as web services. The role of a map server is to render the spatial data in web friendly formats (e.g.: PNG, KML, GML, and GeoJSON) and publish the data as standardized web services. Swain et al. (2015) found that two map servers were preferred in earth science web applications: MapServer and GeoServer. We chose to include GeoServer in Tethys, rather than MapServer, because GeoServer provides a web administration interface that can be used to configure and maintain it without needing to sign on to the actual machine where GeoServer is running, whereas MapServer is configured via a file on the server. Furthermore, GeoServer provides

a configuration API that can be used to programmatically manage the data and resources remotely, a feature of which Tethys SDK takes advantage. GeoServer is a Java-based implementation of several OGC web service standards including Web Map Service (OGC-WMS), Web Feature Services (OGC-WFS), and Web Coverage Service (OGC-WCS) (Michaelis and Ames, 2008; Iacovella and Youngblood, 2013). GeoServer also implements the OGC-Styled Layer Descriptor (OGC-SLD) standard to provide a mechanism for styling layers. It is capable of serving many common spatial files types including Shapefiles, ArcGRID, GeoTIFF and it can be used to publish features stored in PostGIS spatial database tables.

2.1.4. Visualization

Two alternatives for interactively visualizing spatial datasets in web apps are included: OpenLayers 3 (2-Clause BSD License) and Google Maps™ (not open source), both of which were equally preferred for earth science web app development in our review (Swain et al., 2015). OpenLayers is a JavaScript web-mapping client library (Steiniger and Hunter, 2013) for rendering interactive maps that allow users to pan, zoom, and select features on a web page (Hazzard, 2011). It is capable of displaying 2D maps of OGC web services and other spatial formats. Google Maps™ provides the ability to render spatial data in a 2D mapping environment similar to OpenLayers (Google, 2014), but it only supports displaying data in KML formats and data that are added via JavaScript API. Both map libraries provide interactive drawing capabilities to allow users to draw features on the map as a means of obtaining spatial input for web apps.

Plotting capabilities are provided by Highcharts, a JavaScript library created by Highsoft AS, and D3. The plots created using Highcharts are interactive with hovering effects, pan and zoom capabilities, and the ability to export the plots as images. Supported plots include line, spline, area, area spline, column, bar, pie, scatter, angular gauges, area range, area spline range, column range, bubble, box plot, error bars, funnel, waterfall and polar chart types. Highcharts is free to use for some applications, but requires the purchase of a license for government and commercial use (Highsoft AS, 2014).

D3 is provided as an open-source alternative for plotting and does not have the same license restrictions (Bostock, 2015). D3 can be used to create interactive plots similar to those of Highcharts, but we found that D3 required greater effort to implement plotting support than Highcharts. Tethys app developers, however, can use either plotting library interchangeably, because the Tethys SDK provides the same interface for both Highcharts and D3 plots.

2.1.5. Distributed computing

To facilitate the large-scale computing that is often required by environmental applications, Tethys Software Suite leverages the computing management middleware HTCondor (Apache License). HTCondor is both a resource management and a job scheduling software. It was developed at the University of Wisconsin-Madison with the primary goal of scavenging idle computing time on networked desktop workstations (Litzkow et al., 1988). It has matured to be a flexible and powerful computing resource management system that can make use of supercomputers, computing grids, and cloud computing. HTCondor facilitates High-Throughput Computing (HTC). HTC differs from High Performance Computing (HPC) in that its main objective is to provide a large amount of computing power over a long period of time (days to months) whereas HPC focuses on providing a large amount of computing power per second (Livny et al., 1997). HTC systems are very well suited to performing loosely coupled or uncoupled (embarrassingly parallel) tasks. These types of computing tasks, which include stochastic analysis and parameter sweeps, are commonly encountered in environmental modeling.

2.1.6. Docker installation

The 3rd party software tools included in Tethys Software Suite each have their own unique installation instructions, which could make installation of Tethys Platform a burden. To overcome this hurdle, we developed Docker images that simplify installation. Docker builds on Linux virtualization capabilities to provide a lightweight mechanism for packaging and distributing web applications (Docker Inc., 2015). Specifically, we created Docker images for PostgreSQL with the PostGIS extension, GeoServer, and 52°North WPS. Docker images are used to create containers, which are essentially stripped down virtual machines running only the software included in the image. Unlike virtual machines, the Docker containers do not partition the resources of your computer (processors, RAM, storage), but instead run as processes with full access to the resources of the computer. Although Docker virtualization technology is in its infancy, we selected it because it has the advantage of providing a platform agnostic approach to installation of the software suite and reduces the installation procedure to three simple steps: install Docker, download the Docker images, and create Docker containers. Docker also containerizes or sandboxes each of the software components, which provides an added measure of security. It has also been adopted by many well-known companies and organizations as a solution for scalable web development including eBay, BBC News, The New York Times, Uber, Orbitz, and PayPal (Docker Inc., 2016). However, users may still install the 3rd party software tools without Docker and use them with Tethys Platform.

2.2. Tethys Software Development Kit

Tethys SDK was designed to address web development and software orchestration hurdles. It includes a framework for developing web apps, a command line interface (CLI) for simplifying common management tasks, and application programming interfaces (APIs) for each component of Tethys Software Suite. We have assumed the primary developers that would use Tethys Platform would be scientist and engineers with some scientific scripting experience but with limited experience with web development. As such, we selected Python as the language of Tethys SDK, because of its growing popularity within the scientific and engineering communities (Swain et al., 2015). The major components of Tethys SDK are discussed in limited detail in the following sections for brevity. For detailed documentation and explanations of Tethys SDK visit http://docs.tethysplatform.org.

2.2.1. Software APIs

Tethys SDK addresses the software orchestration hurdle by providing Python APIs for each of the software in the software suite. This makes it possible for app developers to orchestrate the functionality of the software in their scripts using a common scripting language. The APIs make use of existing Python modules where possible to prevent reinventing the wheel, but also include custom Python modules to further simplify and fill in the gaps.

For example, the Persistent Stores API consists of a popular database management Python module called SQLAlchemy (Bayer, 2015) and custom functions to streamline the creation of databases (termed persistent stores) as a means for scripting to the PostgreSQL databases. The Spatial Dataset Services API, on the other hand, is an amalgamation of a custom module, tethys_datatset_services, that provides a simplified interface for an existing module, and gsconfig, to provide a method for managing GeoServer programmatically. Additionally, Tethys SDK provides APIs for external services such as the Dataset Services API, which can be

used to manage file datasets using either CKAN instances (Open Knowledge Foundation, 2014) or HydroShare (Tarboton et al., 2014).

Tethys SDK also provides APIs for managing computing infrastructure and job submission. The Compute API provides a way to programmatically provision cloud-computing resources and configure them into an HTC system with HTCondor, while Jobs API can be used to create and submit jobs to those resources. To facilitate jobs that may take a long time to run, submitting jobs is handled asynchronously, meaning that when a request is made to the server to run a job the server will submit the job, but it will not wait for the job to complete before responding to the request. The server will then poll the job periodically to check the job's status, or the job can be configured with a callback URI to let the server know when it has completed. A summary of the key APIs in Tethys SDK and how they are related to Python modules and elements of the software suite is shown in Table 2.

2.2.2. App development framework

Tethys SDK addresses the web development hurdle by providing an app development framework and the Template Gizmos API. Rather than starting from scratch, the Tethys app development framework extends Django, a popular Python web framework (Django Software Foundation, 2015), to give it the core web capabilities required for web apps. The framework simplifies web development by providing a structured approach for building and configuring apps. It helps developers create more manageable code by using the model-view-controller (MVC) software architectural pattern, in which the code is organized into three categories: the model, code that comprises the data management components of the app; the views, which includes the user interface code or web pages of the app; and the controllers, which consists of the logic of the app.

Much of Tethys app development uses the structures and conventions provided by Django, but Tethys includes a layer designed to simplify some aspects of Django development. For example, the mechanism for designing URLs with variables in Django requires the use of regular expressions—a sequence of characters that form a search pattern—using a syntax that would prove difficult for inexperienced developers (see Fig. 2). Tethys provides a layer of abstraction to simplify URL design using a simple syntax. Fig. 2 illustrates this difference in approach to URL design.

Another major simplification provided by the Tethys web framework is that it provides a base template that includes a standard layout for app pages delineated into areas for a header, navigation links, action buttons, and primary content (see Fig. 3). This base template reduces the amount of repetitive template coding required for the creation of new pages often referred to as boilerplate code—essentially the code that is required before you are able to start on the primary task at hand. It also dramatically reduces the amount of HTML and CSS that is required to develop Tethys web apps.

```
1  # Example of Django URL
2  r'^/users/(?P<user_id>[0-9]+)/edit/$'
3
4  # Equivilent URL using Tethys SDK
5  '/users/{user_id}/edit'
```

Fig. 2. Comparison of Django URL specification with regular expressions (top) and Tethys URL specification (bottom).

The other major simplification to web development is afforded by The Template Gizmos API. It simplifies the development of dynamic user interfaces of an app by providing a set of common interface elements called “gizmos”. Using gizmos developers can add date-pickers, plots, and maps to their apps using Python and a Django template tag in the HTML page—without writing JavaScript. Gizmos reduce the need to work in multiple languages by automatically inserting or referencing the necessary HTML, CSS, and JavaScript libraries in place of the gizmo template tag. Fig. 4 shows an example of how to create a map using the Map View gizmo, which inserts an OpenLayers map.

2.2.3. Command Line interface

The Tethys SDK provides a command line interface to automate some of the common management tasks associated with developing web apps with Tethys Platform. For example, it includes a scaffolding command for generating new app projects, a command for managing app databases, commands to assist with installation and updating Tethys Platform, and commands for managing the software suite Docker images and containers.

2.3. Tethys Portal

Tethys Portal was designed to overcome the app deployment hurdle. It is implemented as a Django website project and leverages the capabilities of Django to provide the core website functionality that is often taken for granted in modern web applications. It includes a user account system complete with user profiles and a password reset mechanism for forgotten passwords. User accounts are necessary to allow app developers to customize user experience, grant or restrict access to data or functionality provided by

Table 2
Relationship between APIs, Python modules, and software components.

API	Python modules	Software components
Template Gizmos	Django templating language	OpenLayers Google Maps™ HighCharts
Dataset services	*tethys_dataset_services*	CKAN HydroShare
Spatial dataset services	*tethys_dataset_services* gsconfig	GeoServer
Persistent stores	SQLAlchemy	PostgreSQL
Spatial persistent stores	SQLAlchemy GeoAlchemy	PostgreSQL w/PostGIS
Web processing services	OWSLib	52° North WPS
Jobs	*CondorPy*	HTCondor
Compute	*TethysCluster*	Amazon Web Services Microsoft Azure

Note: The names of custom Python modules developed for Tethys Platform are italicized.

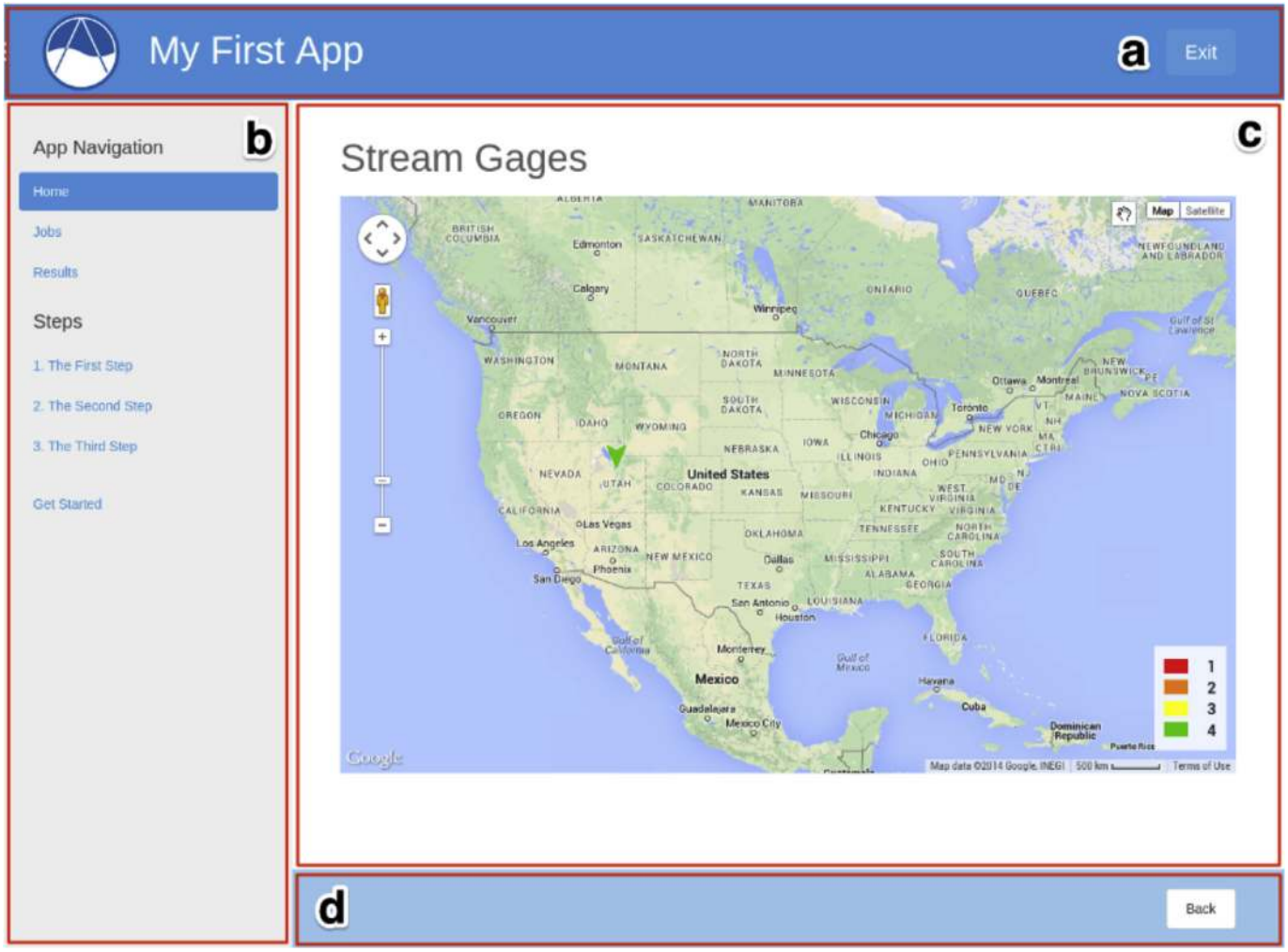

Fig. 3. An illustration of the layout provided by the base template for apps, which includes areas for (a) a header, (b) navigation links, (c) main content, and (d) action buttons.

```python
# Configure in Controller
map_view_options = {'height': '400px',
                    'width': '100%',
                    'controls': ['ZoomSlider',
                                 'Rotate',
                                 'FullScreen',
                                 'ScaleLine',
                                 {'ZoomToExtent': {'projection': 'EPSG:4326',
                                                   'extent': [-135, 22, -55, 54]}},
                                 {'MousePosition': {'projection': 'EPSG:4326'}},
                    ],
                    'layers': [{'WMS': {'url': 'http://demo.opengeo.org/geoserver/wms',
                                        'params': {'LAYERS': 'topp:states'},
                                        'serverType': 'geoserver'}
                               },
                    ],
                    'view': {'projection': 'EPSG:4326',
                             'center': [-100, 40], 'zoom': 3.5,
                             'maxZoom': 18, 'minZoom': 3},
                    'base_map': 'OpenStreetMap'
}

# Add this line to HTML Template
{% gizmo map_view map_view_options %}
```

Fig. 4. Example of how to configure a Map View gizmo using Python and the gizmo tag in the HTML.

the app, and protect privacy rights of the users. App developers are not required to restrict access to their apps to users that are logged-in allowing for openly accessible apps. The user account system can also be used in conjunction with the OAuth 2.0 standard to allow users to log in with social media accounts like Google, Facebook, and LinkedIn.

One important note on the user account system is that it is only used to authorize and grant access rights to the portal itself, and not to each of the underlying software components (e.g.: GeoServer, PostGIS). Access rights to the software components is governed by the app developer and the portal administrator in a safe and secure way. When deploying Tethys Portal, each of the software components are setup as independent services, each with their own controls or access, by the portal administrator. App developers request certain services and the portal administrator links the app with those services during installation of the app in such a way that the app developer doesn't need to know the credentials of the service to use it in the app. In other words, usernames, passwords, and other credentials for the software components are never used anywhere in the code for apps and they are only known to the portal administrator adding security.

Tethys Portal also provides a landing page that can be used to showcase the capabilities of the Tethys Platform instance and an app library page that serves as the access point for installed apps (Fig. 5). It includes an administrator backend that can be used to manage user accounts, permissions, link apps to elements of the software suite, and customize the website to match the hosting organization branding. The title, theme, logos, and content on the homepage can easily be changed through the administrative backend.

3. Results

Tethys Platform was designed to lower the barrier to development for scientists and engineers who wish to convey their data and models via interactive web apps. The purpose of this section is to demonstrate its capabilities. Section 3.1 describes four web applications developed using Tethys Platform and Section 3.2 describes a few of the known deployments of Tethys Portal.

3.1. Applications

To date, Tethys Platform has been used to develop at least 24 known environmental web applications. These existing apps have been developed at the U.S. Army Engineer Research and Development Center, Brigham Young University, University of Texas at Austin, CUAHSI, and by at least one private consulting company (Aquaveo). These Tethys apps have been developed to support a variety of commercial, government, and academic projects including HydroShare, CI-Water, and the CUAHSI Water Data Center. We anticipate the number of available apps and app portals to grow rapidly in the coming years as the software gains wider attention. Most of the existing apps, including those described in this section, are accessible at either the official Tethys Demo Portal (http://demo.tethysplatform.org/apps), the Brigham Young University portal (http://tethys.byu.edu), the HydroShare Apps Portal (https://apps.hydroshare.org) and/or the HydroShare Apps Development Portal (https://appsdev.hydroshare.org).

3.1.1. Streamflow Prediction Tool

Snow (2015) developed a web service that automatically queries the latest global runoff forecasts published by the European Center for Medium-Range Weather Forecasts (ECMWF), downscales the forecast to a higher resolution catchments using Esri's RAPID Toolbox, and routes the resulting runoff through a high resolution stream network using the Routing Application for Parallel computatIon of Discharge (RAPID; David et al., 2011) model. The result is a high-resolution dataset of two-week stream flow forecasts every 12 h with the potential for nationwide or even global coverage. In order to provide a context for the severity of a forecasted event an initial simulation of the watersheds and reaches is performed using the ERA-Interim reanalysis data assembled by ECMWF (Dee et al., 2011). This one-time simulation generates a 35-year "model

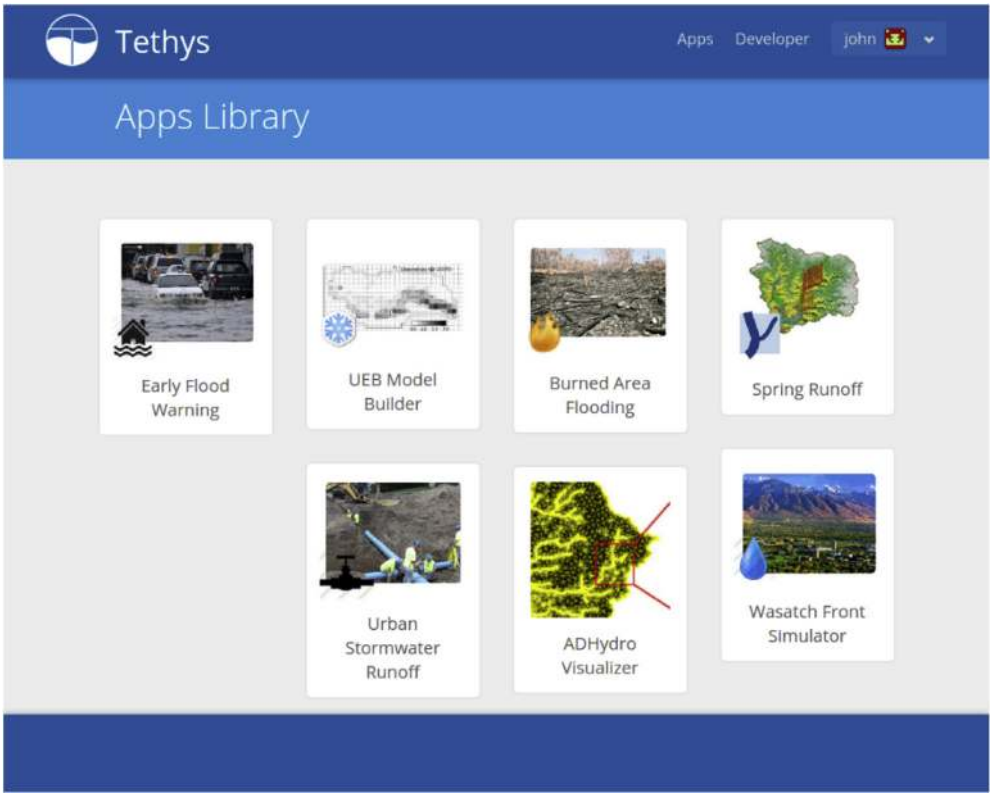

Fig. 5. Tethys Portal includes an app library pages, which serves as the launching point for installed apps.

hydrology" that can be used to derive a return period for each reach modeled that provides a localized context of hydrologic severity for subsequent forecasts.

To manage and visualize resulting forecasts, Snow developed a web app using Tethys Platform to view the recent streamflow forecasts called the Streamflow Prediction Tool (Fig. 6). The computations are performed via the web service twice daily as the ECMWF runoff prediction datasets become available. After the computations are finished, they are then deposited into a data store such as HydroShare or CKAN. The web app automatically retrieves the most recent week-worth of predictions from the data store for visualization by the user.

Upon launching the app, users are prompted to select one or more watersheds to display on a map, from which they can select a reach and view a time series of the forecasted flow two weeks in advance. The ECMWF forecast is an ensemble of 52 different scenarios, so results are displayed showing min, mean, std. deviation, and max. Because the 35-year reanalysis model hydrology exists for each reach a warning point corresponding to the 2 (yellow), 10 (red) and 25 (purple) return period can be indicated both as a warning triangle on the map and in the context of a displayed hydrograph for a selected reach. Where USGS gages are available, observed data can also be queried as a web service and displayed with the current or previous forecast of a selected reach.

The Streamflow Prediction Tool was a demonstration project as part of the National Flood Interoperability Experiment (NFIE) held at the new national water center in Tuscaloosa, Alabama during the summer of 2015 (Snow et al., 2016). The entire 2.7 million reach/watershed of the US National Hydrography Dataset Plus (NHDPlus) was made operational and continues to today to produce the twice-daily forecasts at demo.tethysplatform.org.

3.1.2. Canned GSSHA

The Canned GSSHA web app was developed to demonstrate the Canned Modeling method of flood forecasting (Dolder et al., 2015). The premise of the Canned Modeling method is that when a flood event is imminent, there is limited time to execute hydrologic model runs in an attempt to predict the outcome of the flood event. The solution provided by the Canned Modeling method is to pre-run a large number of models with varying input parameters, called "scenarios", and store or "can" the results for lookup in the time of a crisis. When a potential flood event occurs, the canned model database is queried using the current or forecasted conditions and the model run with the closest match is instantly returned. The Canned Modeling method can be applied to any hydrologic model, or even any combination of different hydrologic models.

A Gridded Surface Subsurface Hydrologic Analysis (GSSHA; Downer and Ogden, 2004) model was developed for a 2.5 km^2 test watershed, and seven input parameters were selected to generate scenarios that would produce both snowmelt and rain-driven floods. A Latin Hypercube approach was used to create different combinations of the selected input parameters uniformly over the entire parameter space resulting in 2187 scenarios. The GSSHA model was executed for each of the generated scenarios and the resulting hydrographs were stored in a database.

The Canned GSSHA app provides an intuitive, single-page user interface that allows users to alter the observed or forecasted input parameters and view the computed hydrograph of the closest match (Fig. 7). A cluster of sliders on the left-hand side of the screen can be used to modify the input parameters. The polar plot at the center shows normalized values of the parameters selected by the user in yellow and the normalized values of the parameters from the pre-computed model that most closely matches the user input in green. The plot on the right-hand side of the screen displays the hydrograph of the closest matching model. Each time the user changes the value of any of the sliders, the lookup is near instantaneous and the polar plot and hydrograph update immediately.

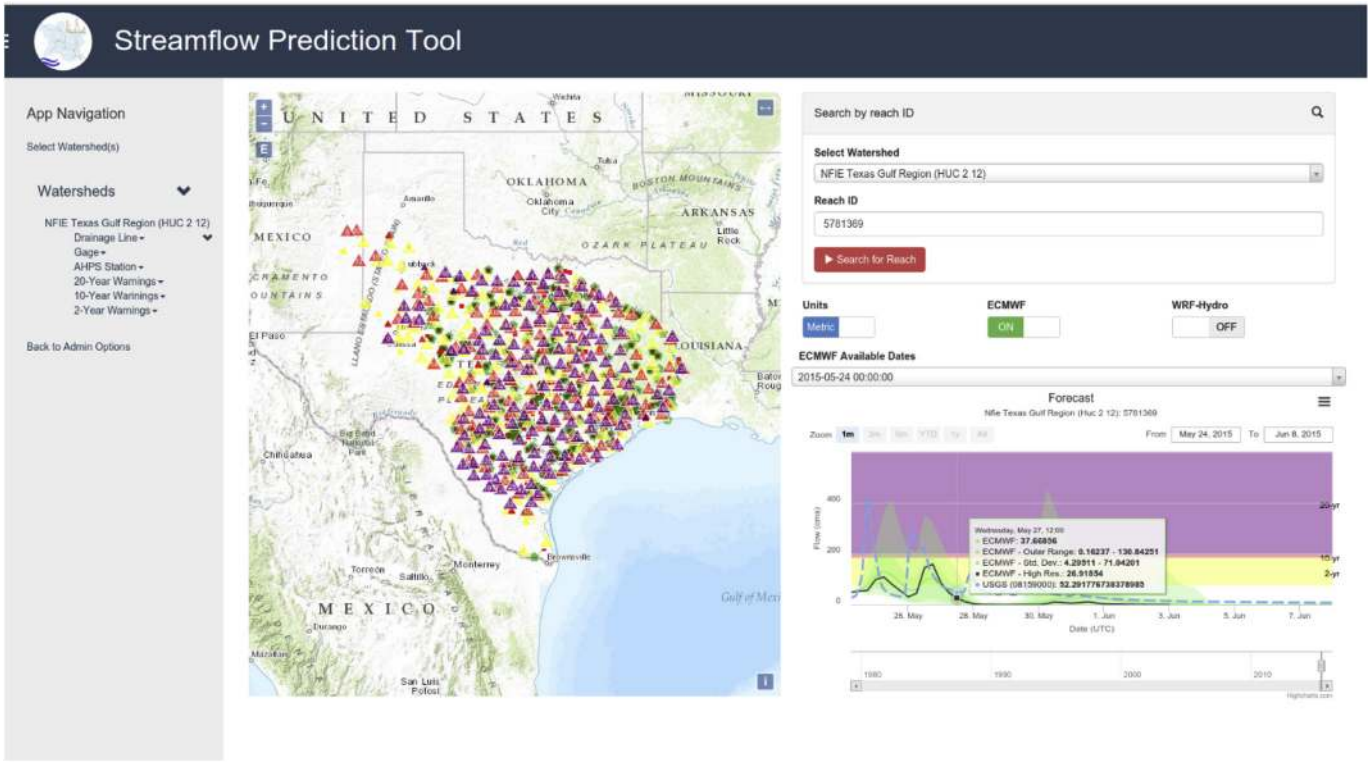

Fig. 6. The Streamflow Prediction Tool web app for computing 15-day streamflow forecasts based on ECMWF weather forecasts.

3.1.3. Parley's Creek Management Tool

The Parley's Creek Management Tool was developed to make water management research products more accessible to decision makers (Goharian and Burian, 2014). Parley's Watershed is one of four major drainages that are included in the Salt Lake City (SLC), Utah, USA protected watershed canyons (USDA Forest Service, 2015). Parley's Creek Basin, located on western slope of Wasatch Mountains, includes two reservoirs, Little Dell and Mountain Dell. The reservoirs were developed with the primary use of municipal and industrial water supply and secondary use of flood control. The main inflows to the reservoirs are generated from Lambs Creek and Dell Creek. Moreover, the outflow from Little Dell reservoir discharges into Mountain Dell. The bypassed water from Parley's water treatment facility flows into the Parley's Creek and passes through the urbanized area of SLC into the Jordan River and ends at the Great Salt Lake.

Goharian et al. (2015) worked with Parley's Creek reservoir managers to develop a systems dynamic model using GoldSim, a Monte-Carlo simulation software for dynamic modeling of complex systems (GoldSim Technology Group, 2015). The model allows managers and stakeholders to explore various management scenarios for the Parley's Creek system. It also allows managers to explore the impact of various climate change projections on reliability and vulnerability of system.

The Parley's Creek Management Tool web app provides a simple user interface and structured workflow for the Parleys Creek Management GoldSim model. The workflow consists of 4 or 5 pages (depending on the type of climate scenario selected) that prompt the user to select a climate scenario, modify reservoir characteristics, adjust the inflow as a multiplier of the historical average on a monthly basis, adjust demand rates on a monthly basis, and view a summary of the parameters. The GoldSim model is hosted as a web service using 52° North WPS, which can be executed from a page in the web app. After a model run has been completed the user can view several plots and download the results as an excel spreadsheet from a results page (shown in Fig. 8). The results include the inflow to the reservoirs, volume of water in reservoirs, releases, and spills as well as the reliability of system to meet the water demand from SLC.

3.1.4. Data Rods Explorer

The Data Rods Explorer web app was developed by the University of Texas in support of a NASA grant focused on enabling access to NASA earth observation datasets as time series data. This app allows users to quickly obtain, plot, and map hydrologic data from various publicly available datasets. The datasets available include (1) LDAS for the North-American (NLDAS) and the global (GLDAS) Noah models (Mitchell, 2004; Xia et al., 2012), (2) the Tropical Rainfall Measuring Missing (TRMM) (Simpson et al., 1996), and (3) the Gravity Recovery and Climate Experiment (GRACE) (Tapley et al., 2004).

The Data Rods Explorer web app combines data from different sources: raster images that come from a WMS server (Goddard Earth Sciences Data and Information Services Center, 2015a) and time series that come from a Data Rods server (Goddard Earth Sciences Data and Information Services Center, 2015b) for comparison. Fig. 9 shows an example of comparing different years for a single location in the Data Rods Explorer. The plot (bottom) overlays total evapotranspiration data (NLDAS-Noah) at Los Angeles, CA for five years: 2010–2014. The map (top) shows the total evapotranspiration raster for California and the southwest on July 1, 2015. The map shows the persistent low evapotranspiration values (yellow) in comparison from larger values (blue).

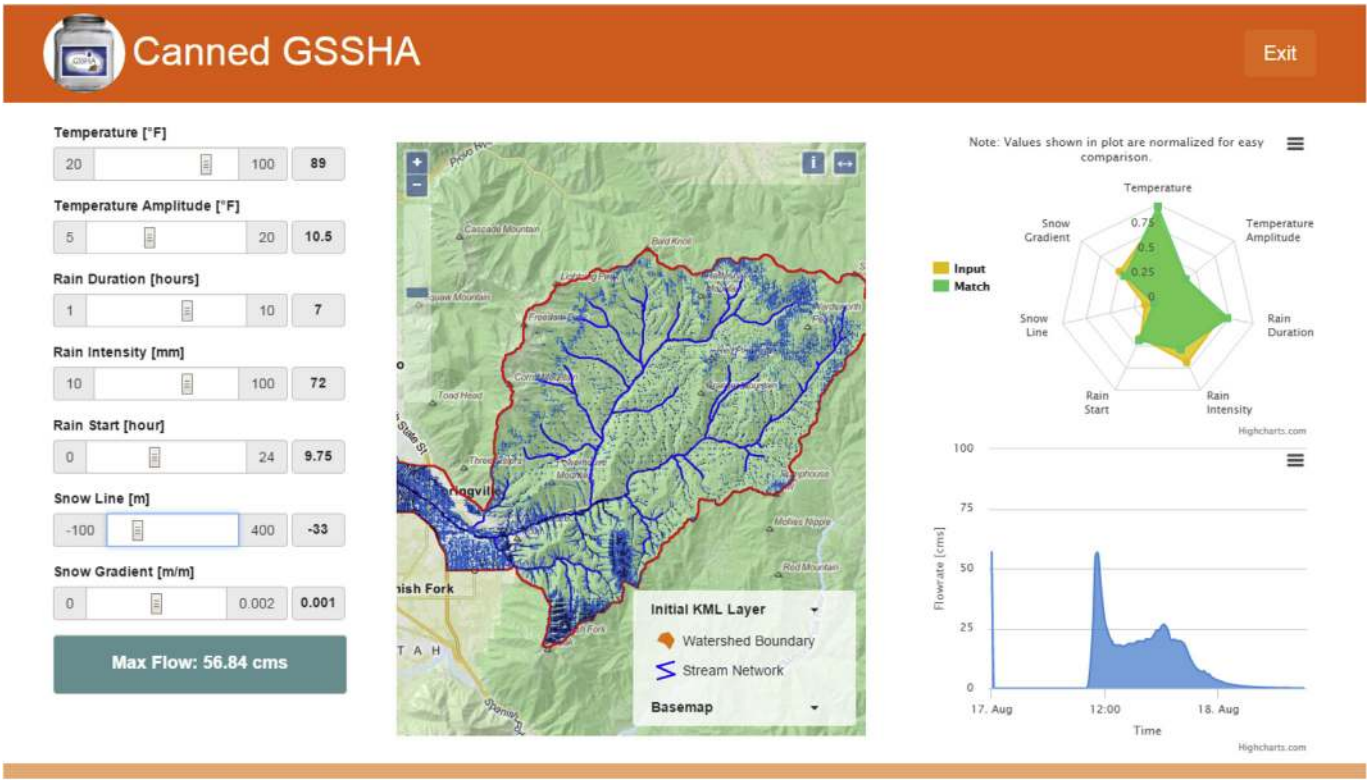

Fig. 7. The Canned GSSHA web app for exploring pre-computed hydrographs for specified conditions.

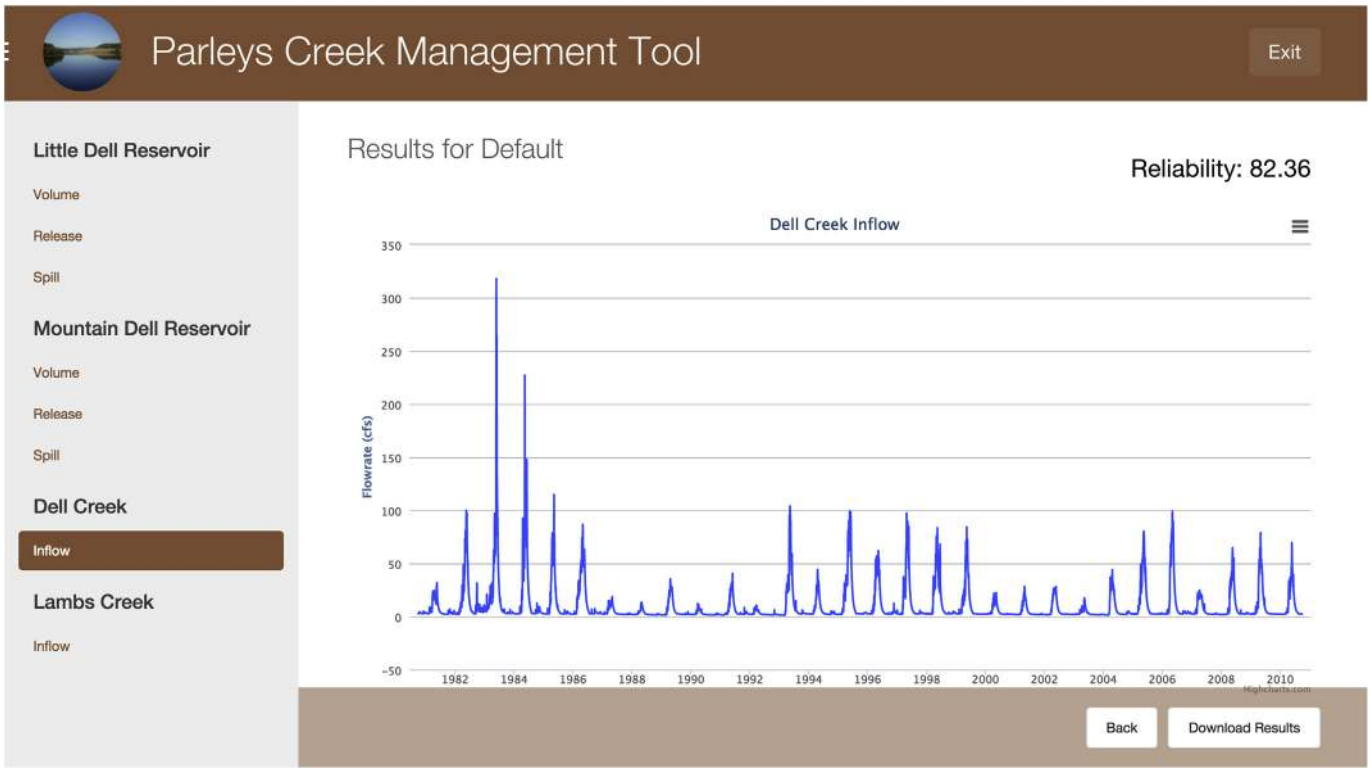

Fig. 8. The Parley's Creek Management web app showing inflows at Dell Creek, Utah.

3.1.5. Snow Inspector App

The Snow Inspector app (Kadlec and Ames, 2014; Kadlec et al., 2016) is another example of an app developed outside of the original CI-Water project. Snow Inspector was developed to enable simple discovery and access of the snow cover history derived from the Terra Snow Cover Daily L3 Global 500 Terra daily snow cover dataset for any point on Earth (Hall et al., 2006). The user can specify a number of historical days to retrieve, then use the interactive map to select a location, retrieve historical snow coverage data at the selected point, display a time series of snow percent coverage, and download the processed data in WaterML or CSV file format (Fig. 10). The data source used by this app is the NASA's Global Image Browse Services (GIBS; Cechini et al., 2013; Thompson et al., 2014) Web Map Tile Service (WMTS) server. The original dataset is the MOD10A1 fractional snow cover grid with 500 m resolution. The app depends on the open-source pypng library for extracting values from the WMTS images.

3.1.6. EPANET web app

The EPANET Web App was developed as a commercial product by Aquaveo LLC for municipalities and water districts. The app provides a means for visualizing EPANET (Rossman, 2000) water distribution system models from any device including tablets and phones of operators in the field. Users upload an EPANET input file and a corresponding report file and the network geometry (e.g.: pipes, junctions, tanks) and model results are displayed on an interactive map that allows users to select network elements, view their properties and values, and plot values over time for extended period models (Fig. 11). Future developments for this app include the ability to make slight modifications to the model and execute the model using cloud computing resources as a means of scenario exploration and performing diagnostics in the field.

3.2. Deployments

Web apps developed using Tethys Platform are deployed through an instance of Tethys Portal. Tethys Platform aims to simplify the process of deployment as much as possible by providing Docker images for installing the software suite and detailed instructions for installing Tethys Portal in a production mode (http://docs.tethysplatform.org). However, there is still a hurdle of obtaining and maintaining the server on which the Tethys Portal runs. The advent of commercial cloud computing services has provided one cost effective means that most organizations could harness. Researchers at universities may also be able to take advantage of services provided by the university at discounted rates. This section provides descriptions of a few of the existing Tethys Portal instances as examples of how Tethys apps have been successfully deployed.

3.2.1. Tethys Platform demo server

The longest running instance of Tethys Portal is the official Tethys Platform Demo Server (http://demo.tethysplatform.org). At the time of writing, the server had been running since the initial Tethys Platform release or about 1 year, was hosting 10 web applications developed by 6 developers and had over 150 registered users. The Tethys Demo server was configured in a distributed fashion consisting of 5 servers, with one server operating as the primary web server with Tethys Portal, and the others dedicated to one element of the software suite (GeoServer, PostGIS, 52° North WPS, and CKAN). The servers are maintained by the Environmental Modeling and Research Laboratory at Brigham Young University and operate out of a data center at the University of Utah.

3.2.2. HydroShare

The next most prominent instances of Tethys Portal are two servers maintained by the HydroShare research group: HydroShare

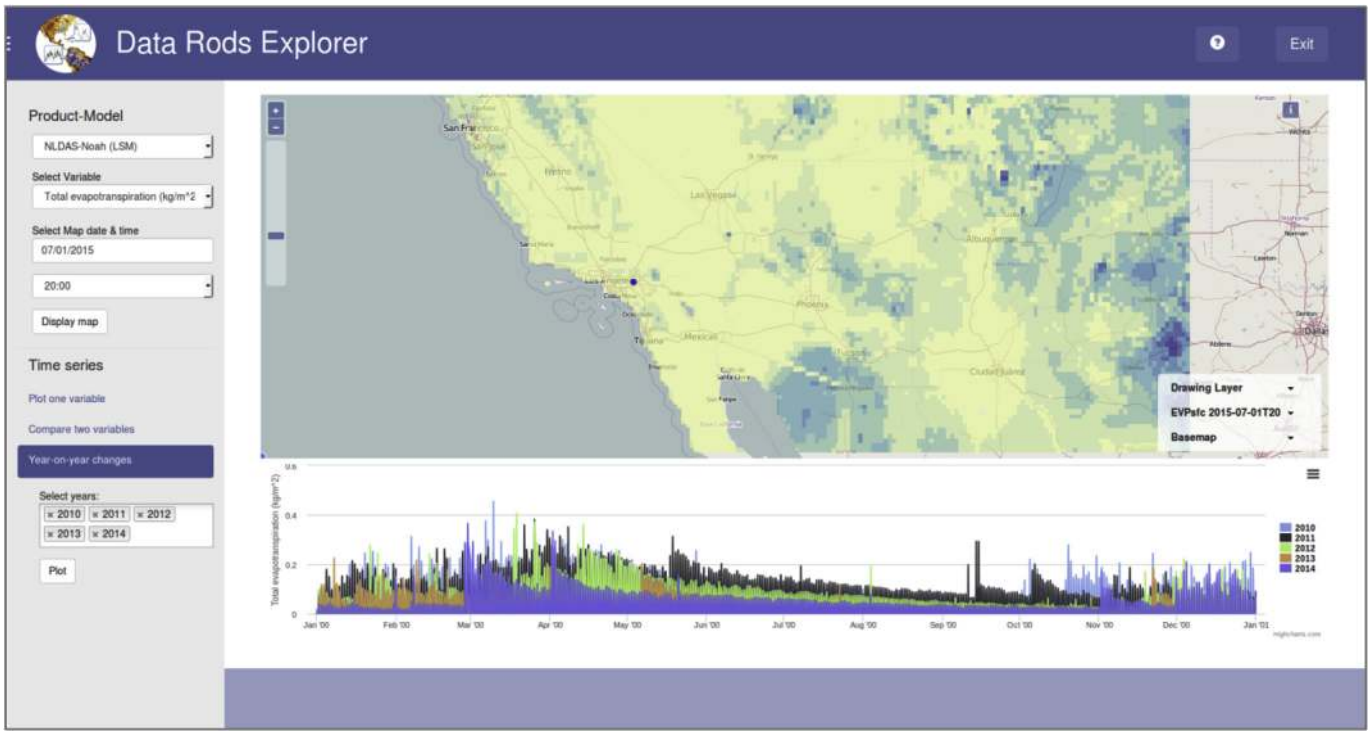

Fig. 9. The NASA Data Rods Explorer web app showing NLDAS-Noah Land Surface Model results.

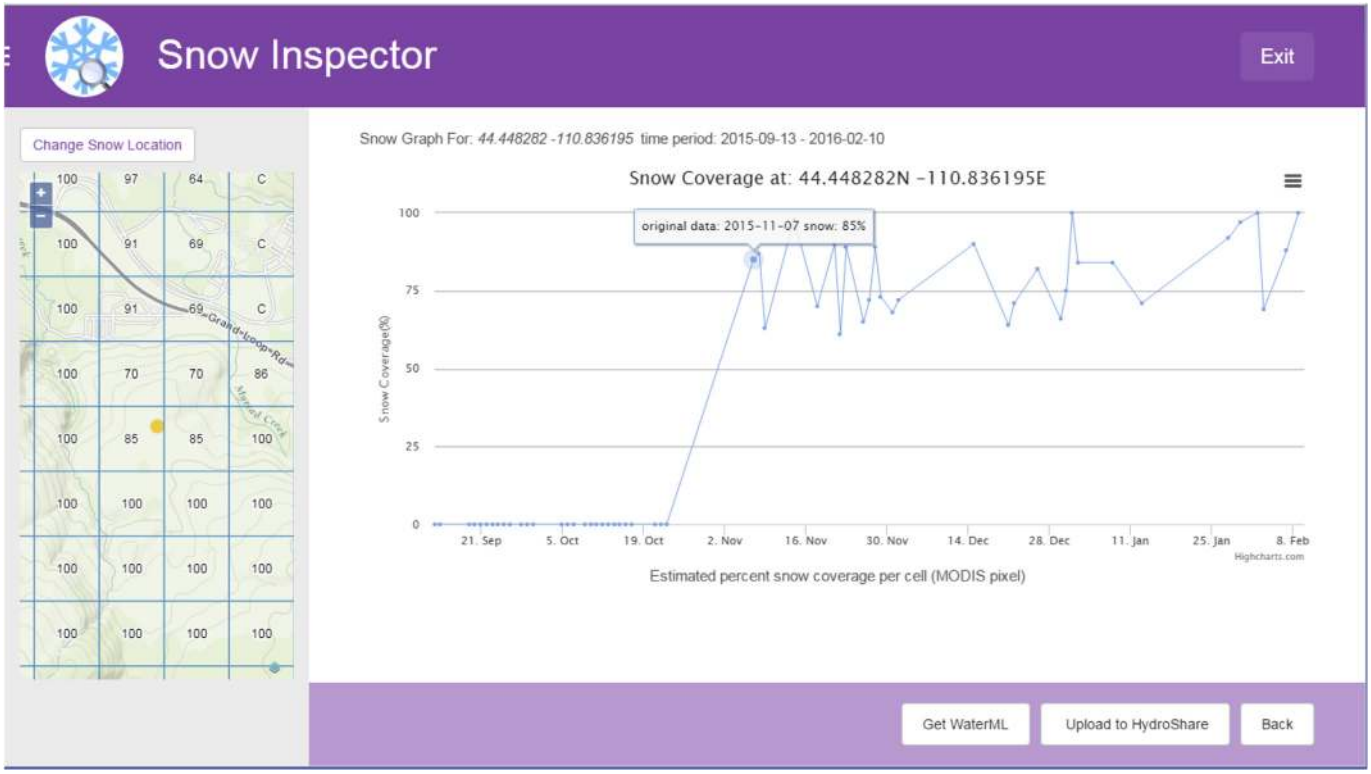

Fig. 10. The Snow Inspector web app showing historical percent snow coverage near Old Faithful Geyser, Yellowstone National Park, USA.

Apps Server (https://apps.hydroshare.org) and HydroShare Apps Dev Server (https://appsdev.hydroshare.org). The former hosts officially released apps for the HydroShare project, while the latter is used as a staging server for apps under development or in experimental release. The HydroShare Apps server had been running for 8 months, though recently wiped clean and re-deployed, and had 5 apps running on it and had 40 registered users. The HydroShare Apps Dev server had been operating for

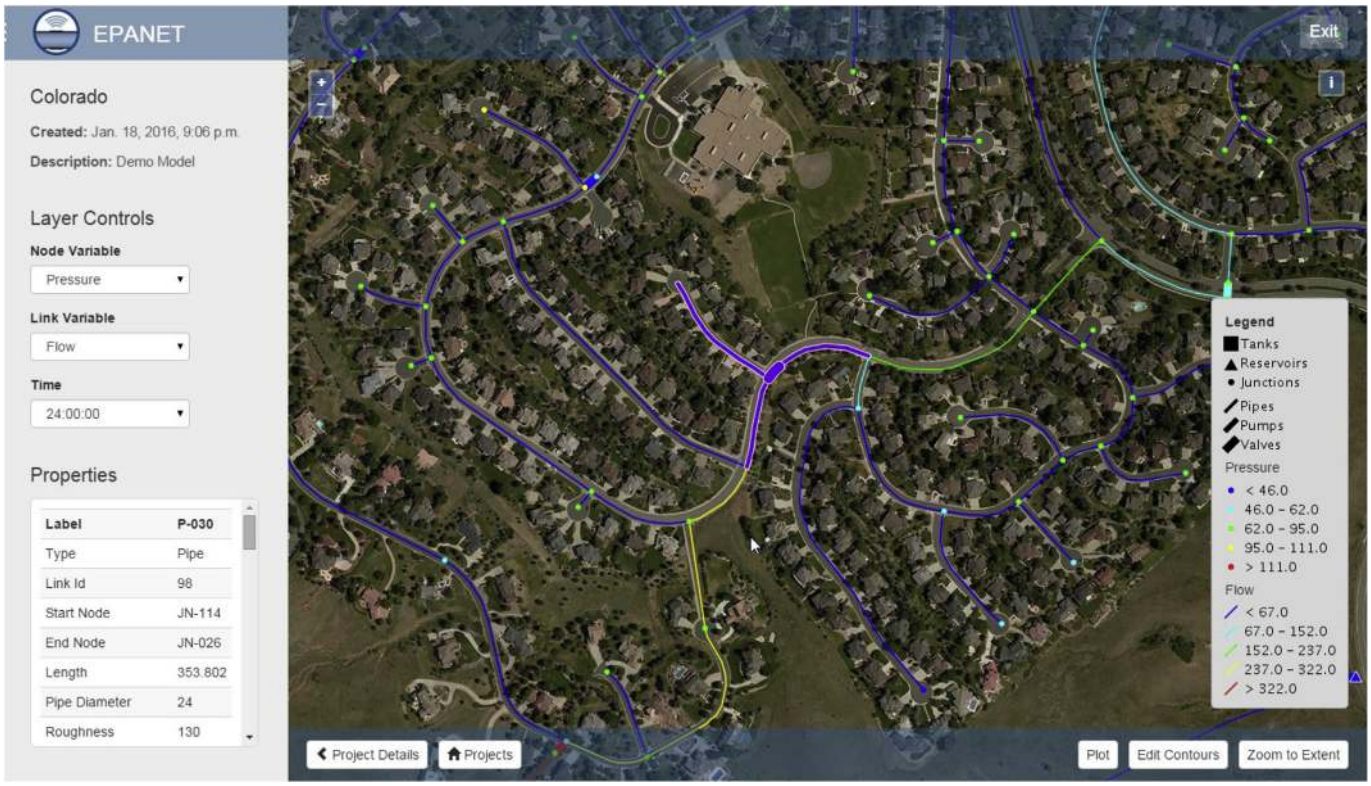

Fig. 11. The EPANET web app for visualizing EPANET water distribution system models.

about 5 months at the time of writing, had 21 registered users and 10 web applications. Both HydroShare Tethys Portals were configured with the entire Tethys Platform installed on a single, though more powerful, server. The HydroShare Apps Server was originally deployed in the Digital Ocean (Digital Ocean Inc., 2016) commercial cloud provider and later moved to a server provided by the Renaissance Computing Institute (Rennaissance Computing Institute, 2016) where the HydroShare Apps Dev server is also operating.

3.3. Community

Tethys Platform has a growing number of users and efforts have been made to accommodate its use and ensure it is sustainable. In an effort to promote the development of a community a forum has been setup for users to post questions and document answers (https://groups.google.com/forum/#!members/tethysplatform) and the issues tracker on the primary GitHub repository is being used to track bugs and feature requests (https://github.com/tethysplatform/tethys/issues). To date there are 15 registered members of the Tethys Platform forum, half of which are not members of the primary Tethys development team with approximately 25 answered posts. The GitHub issue tracker is being actively used with 48 open issues and 153 closed issues and 2 contributions by developers other than the primary development team via pull request.

Tethys Platform is backed by a consortium of partners from academia, government, and the private sector including Brigham Young University, CUAHSI, the Army Corp of Engineers, and Aquaveo LLC. The project is governed by a project steering committee that consists of at least 3 members who meet regularly to discuss development priorities. The proposals for new features and selection of project steering committee members is performed democratically according to the procedures outlined in the Tethys Platform Project Steering Committee document (Tethys Platform, 2016).

3.4. Qualitative analysis

As a more qualitative measure of how Tethys Platform lowers the barrier, we performed an analysis of the language composition of each app and compared it with the language composition of Tethys Platform as an example of a website project created from scratch. We used the Count Lines of Code (CLOC) program to count the number of lines of each of the four major languages (Python, JavaScript, HTML, and CSS). The line counts exclude blank lines, comment lines, and third party libraries. We also subtracted number of lines of code that were generated by the scaffold from the totals for each app (Python: 65, JavaScript: 0, HTML: 42, CSS: 0). Absent from this analysis is the language composition of the EPANET web app, which is proprietary. A summary of the language composition of Tethys Platform and each app is shown in Table 3.

One of the aims of Tethys Platform is to reduce the need to learn multiple languages to overcome the web development hurdle. This is evident by the reduced amount of CSS used in each app, ranging from 0 to 9% as compared to 17% for Tethys Platform, and the reduced amount of HTML for three of the apps, ranging from 1 to 14% as compared to 17% for Tethys Platform. However, most of the apps still display a high usage of JavaScript with only one app displaying a significant reduction in the use of JavaScript as compared with Tethys Platform. Despite this fact, all of the apps used as much or more Python, ranging from 33 to 63% as compared to 33% for Tethys Platform, indicating a heavier reliance on Python.

Using Tethys Platform results in a significant reduction in the total number of lines of code written for each app as compared to Tethys Platform. Table 4 shows the lines of code for each app divided by the lines of code for Tethys Platform by language. At the

Table 3
Language composition of each web app and Tethys Platform.

Language	Tethys Platform		Canned GSSHA		NASA data rods		Snow inspector		Parleys creek		Streamflow	
	Lines	Percent	Lines	Percent	Lines	Percent	Lines	Percent	Lines	Percent	Lines	Percent
Python	6729	33%	420	63%	570	34%	637	39%	1281	63%	2693	33%
JavaScript	6786	33%	179	27%	659	39%	714	44%	110	5%	4221	52%
CSS	3433	17%	58	9%	0	0%	78	5%	143	7%	88	1%
HTML	3454	17%	8	1%	450	27%	201	12%	499	25%	1149	14%
Totals	20,402	100%	665	100%	1679	100%	1630	100%	2033	100%	8151	100%

Table 4
Lines of code for each app divided by lines of code for Tethys Platform.

Language	Canned GSSHA	NASA data rods	Snow inspector	Parleys creek	Streamflow prediction
Python	6%	8%	9%	19%	40%
JavaScript	3%	10%	11%	2%	62%
CSS	2%	0%	2%	4%	3%
HTML	0%	13%	6%	14%	33%
All	3%	8%	8%	10%	40%

time of writing Tethys platform was composed of over 20,000 lines of code and the apps ranged from about 600 to 8100 lines of code. On average the apps had only about 14% the total number of lines of code as Tethys Platform.

4. Discussion

In the introduction we described four hurdles that need to be addressed to effectively lower the barrier to environmental web app development including: (1) the software hurdle, (2) the orchestration hurdle, (3) the web development hurdle, and (4) the deployment hurdle. We also presented an approach that can be used to address these four hurdles and we introduced Tethys Platform as an implementation of this approach. In this section we discuss how this approach, as demonstrated by Tethys Platform, overcomes each of these hurdles in more detail. We also discuss other important considerations such as sustainability and broader implications.

4.1. Software hurdle

In our approach, we recommend overcoming the software barrier by providing software that meets the spatial and computational capabilities commonly required for environmental modeling. Our implementation of this approach is embodied in the Tethys Software Suite, a prepackaged suite of software which was the result of an extensive review of FOSS and FOSS4G (Swain et al., 2015). Overcoming this hurdle involved over a year of research and careful consideration for the 3rd party software tools that are included in the Tethys Software Suite.

We emphasize that there is a one aspect of the software hurdle that would not be addressed by simply providing a recommendation of software that is useful for environmental web app development: installing many of these 3rd party FOSS and FOSS4G can be an intensive process and a deterrent for potential developers. Installation instructions for FOSS4G are written for experienced developers and often require a great deal of troubleshooting to account for the differences in operating system environments. Encapsulated in each of the Docker images that were developed to automate installation are weeks-to months-worth of effort toward installation, troubleshooting, and configuration of the software.

4.2. Orchestration hurdle

Our recommendation overcoming the orchestration hurdle is to provide a programmatic means of managing each of the recommended software in a single programming language. This approach is demonstrated by the Python software APIs included in the Tethys SDK. Most of the software in the software suite provide some means of programmatic management, but they vary in form. For example, SQL can be used to interact with the PostGIS database, while GeoServer provides a web API. However, using these native APIs would require the developer to learn both SQL and how to formulate web requests to the control these two elements of the software suite. The Python APIs in Tethys SDK lessen the learning curve by simplifying the specifics of interacting with the software to a series of objects and functions in a common programming language. More concretely, rather than writing SQL to add data to the database, the developer uses Python objects and instead of having to learn how to create a multipart form request to upload a shapefile to GeoServer through its native web API, the developer uses a Python function that takes the path to the shapefile.

One criticism of this approach is that the effort to simplify and prescribe often results in limited functionality and lessens the control the developer has over the software. However, the simplified APIs provided by Tethys SDK do not restrict access to the software through their native APIs. Building on the examples previously, if an advanced developer wishes to write SQL, rather than use the Python approach, they can do so. But if that same developer is not familiar with formulating web requests, she can still benefit from the Python API provided for GeoServer.

4.3. Web development hurdle

Our recommended approach for overcoming the web development hurdle was to reduce the web development skills required to develop web apps. The part of Tethys Platform that demonstrates this approach is the app development framework included in Tethys SDK. A major task of web development is writing the foundational code, often called boilerplate code, which includes the HTML that is common to all pages and provides unified layout the web app, the CSS that provides a theme and style for the web app, and JavaScript that is used for dynamic elements such as menus and user interface logic. Tethys SDK provides a base template, theme, and core JavaScript libraries to overcomes this significant web

development hurdle. The design and development of the base template for Tethys SDK required several months of tedious work. For scientists and engineers who are primarily concerned with developing cutting edge workflows and models or collecting and analyzing data, the boilerplate code for a web app is often perceived as a distraction to the main objective—to make their work accessible. However, this results in the poor user experience typical of existing scientific- or engineering-based web applications and, ironically, does not convey the idea that the user is participating in the latest science or technology. The base template allows developers to immediately start working on the logic and workflow of the app without the need of worrying about how it looks or feels.

Another of the challenges faced by environmental web app developers is the multilingual nature of web development. HTML and CSS are required to provide the structure and theme of each page of the web app and JavaScript is used to provide dynamic capabilities such as plotting and mapping. A scripting language such as PHP, Python, or Ruby is also required on the server to handle the logic of the app and interact with the database. The Template Gizmos API, one of the APIs that is part of the app development framework in Tethys, demonstrates an approach that can be taken to reduces the amount of multilingual coding.

As an example, the Map View gizmo allows developers to add a dynamic OpenLayers-based map to a page in their app. The Map View gizmo consists of HTML, CSS and over 1000 lines of JavaScript and represents several months of development. In contrast, Tethys app developers are able to insert a map by first defining the map object in Python and then adding a single template tag to the HTML page—no JavaScript or CSS required. The template tag is automatically replaced by the necessary HTML, CSS, and JavaScript of the gizmo to create the map. To be fair, creating maps with advanced features such as clicking on a feature on a map and displaying a plot associated with that feature would require additional JavaScript and CSS, but the amount of coding in multiple languages is significantly reduced by the use of the Template Gizmos API.

4.4. Deployment hurdle

Our recommendation to overcoming the deployment hurdle was to provide a web-safe mechanism for deploying the finished web apps that is flexible enough to work on the most common means for obtaining hardware. The implementation of Tethys Portal illustrates one way this can be achieved. Tethys Portal is a fully featured web site for hosting the finished web apps. It is built on Django, a mature web development framework that is under active development and provides web security features to keep the portal safe from the hostile environment of the internet. Tethys Portal can be installed on most Linux distributions that are commonly available at most data centers, giving it flexibility for deployment.

Tethys Portal eliminates the boilerplate code associated with creating a new web site, similar to the manner in which the base template reduces the amount of boilerplate code needed to develops apps. Specifically, it provides a user account management system including user profiles that allow users to edit their identifying information and a mechanism for resetting forgotten passwords, authentication and authorization a homepage, an apps library page that acts as an access point for the installed apps, administrator pages, and web security features. Tethys Platform leverages Django features to provide much of this functionality, but development of these features still required several months.

Tethys Portal is also easily customizable allowing the theme and content to be changed via the admin pages, so that deployed instances can be rebranded to match the organization that hosts it. Tethys Portal makes it significantly easier for organizations to host web apps that they have developed using Tethys Platform.

5. Conclusion

We surmise there are four major hurdles that deter scientists and engineers, even those with scientific programming experience, from developing environmental web applications: (1) the software hurdle, (2) the orchestration hurdle, (3) the web development hurdle, and (4) the deployment hurdle. We have also presented an approach for overcoming these hurdles which includes providing: (1) software that meets the spatial and computational capabilities commonly required for environmental modeling; (2) a programmatic means to use each of the recommended software in a single programming language; (3) a reduction to the web development skills required to develop web apps; and (4) a web-safe mechanism for deploying the finished web apps that is flexible enough to work on the most common means for obtaining hardware (i.e. university cloud, commercial cloud, private data centers). Tethys Platform, a development and hosting environment for environmental web apps, was presented as an implementation of this approach. Tethys Platform consists of three primary components that were designed to address one of the four hurdles: Tethys Software Suite, Tethys Software Development Kit, and Tethys Portal.

Tethys Software Suite addresses the software hurdle by including free and open source software solutions for GIS and distributed computing functionality in environmental web apps. It includes four FOSS4G packages to address the web GIS needs of environmental web apps: PostGIS, 52° North Web Processing Service, and OpenLayers for creating dynamic interactive maps. Additionally, HTCondor is included to manage computing resources. Tethys Software Development kit addresses both the orchestration hurdle and the web development hurdle by providing APIs for each of the software and a framework for developing web apps. Tethys Portal addresses the deployment hurdle by providing a runtime environment for Tethys web apps. This web portal can be easily rebranded and customized to match the organization hosting it. Tethys Platform facilitates making environmental web apps more commonplace, which will serve to narrow the gap between research and practice.

Software availability

Tethys Platform is available under the BSD 2-Clause open source license and the source code is available in the following GitHub repositories:

- https://github.com/tethysplatform/tethys
- https://github.com/tethysplatform/tethys_dataset_services
- https://github.com/tethysplatform/tethys_docker

An overview of Tethys Platform and links to documentation, bug reporting, and support forum are available online at http://www.tethysplatform.org. Live demos of apps developed using Tethys Platform can be found at

- http://demo.tethysplatform.org/apps
- https://apps.hydroshare.org or https://appsdev.hydroshare.org

The source code for all of the example web apps, with the exception of the commercially developed EPANET Web App, are available on GitHub in the following repositories:

- Streamflow Prediction Tool: https://github.com/erdc-cm/tethysapp-streamflow_prediction_tool

- Canned GSSHA: https://github.com/CI-WATER/tethysapp-canned_gssha
- Parely's Creek Management Tool: https://github.com/CI-WATER/tethysapp-parleys_creek_management
- NASA Data Rods Explorer: https://github.com/gespinoza/datarodsexplorer
- HydroShare Snow Inspector App: https://github.com/jirikadlec2/snow-inspector

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1135482.

References

52 °North, 2014. Home — 52°North Initiative for Geospatial Open Source Software GmbH. Retrieved from. http://52north.org/.
Bayer, M., 2015. SQLAlchemy: the Database Toolkit for Python. Retrieved from. http://www.sqlalchemy.org/.
Bhuyan, S.J., Koelliker, J.K., Marzen, L.J., Harrington Jr., J.A., 2003. An integrated approach for water quality assessment of a Kansas watershed. Environ. Model. Softw. 18 (5), 473–484. http://dx.doi.org/10.1016/S1364-8152(03)00021-5.
Bostock, M., 2015. D3.js - Data Driven Documents. Retrieved from. http://d3js.org/.
Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M., 1996. In: Pattern-oriented Software Architectures: a System of Patterns, vol. 1. Wiley.
Cechini, M., Murphy, K., Boller, R., Schmaltz, J., Thompson, C., Huang, T., McGann, J., Ilavajhala, S., Alarcon, C., Roberts, J., 2013. Expanding Access and Usage of NASA Near Real-Time Imagery and Data, AGU Fall Meeting Abstracts, p. 04.
Chambers, J., 2013. The R Project for Statistical Computing. Retrieved from. http://www.r-project.org/.
David, C.H., Maidment, D.R., Niu, G.-Y., Yang, Z.-L., Habets, F., Eijkhout, V., 2011. River network routing on the NHDPlus dataset. J. Hydrometeorol. 12 (5), 913–934.
Dee, D.P., Uppala, S.M., Simmons, A.J., Berrisford, P., Poli, P., Kobayashi, S., Bechtold, P., 2011. The ERA-Interim reanalysis: configuration and performance of the data assimilation system. Q. J. R. Meteorol. Soc. 137 (656), 553–597.
Demir, I., Krajewski, W.F., 2013. Towards an integrated flood information system: centralized data access, analysis, and visualization. Environ. Model. Softw. 50, 77–84.
Digital Ocean Inc, 2016. Simple Cloud Infrastructure for Developers | Digital Ocean. Retrieved from. https://www.digitalocean.com/.
Django Software Foundation, 2015. The Web Framework for Perfectionists with Deadlines — Django. Retrieved from. https://www.djangoproject.com/.
Docker Inc, 2015. Docker: Build, Ship, and Run Any App, Anywhere. Retrieved from. https://www.docker.com/.
Docker Inc, 2016. Customers | Docker. Retrieved from. https://www.docker.com/customers/.
Dolder, H., Jones, N., Nelson, E.J., 2015. Simple method for using precomputed hydrologic models in flood forecasting with uniform rainfall and soil moisture pattern. J. Hydrol. Eng. 0 (0), 04015039. http://dx.doi.org/10.1061/(ASCE)HE.1943-5584.0001232.
Downer, C.W., Ogden, F.L., 2004. GSSHA: model to simulate diverse stream flow producing processes. J. Hydrol. Eng. 9 (3), 161–174. http://dx.doi.org/10.1061/(asce)1084-0699(2004)9:3(161).
ESRI, 2004. ArcGIS Server: ESRI's Enterprise GIS Application Server. Retrieved from. http://downloads.esri.com/support/whitepapers/other_/arcgis-server_90.pdf. http://downloads.esri.com/support/whitepapers/other_/arcgis-server_90.pdf.
Feng, M., Liu, S., Euliss Jr., N.H., Young, C., Mushet, D.M., 2011. Prototyping an online wetland ecosystem services model using open model sharing standards. Environ. Model. Softw. 26 (4), 458–468. http://dx.doi.org/10.1016/j.envsoft.2010.10.008.
Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1995. Design Patterns: Elements of Reusable Object-oriented Software. Addison-Wesley.
Goddard Earth Scienes Data and Information Services Center, 2015a. OGC WMS for NASA Giovanni. Retrieved November 1, 2015, from. http://giovanni.gsfc.nasa.gov/giovanni/daacbin/wms_ag4?VERSION=1.1.1&REQUEST=Getcapabilities&service=wms.
Goddard Earth Scienes Data and Information Services Center, 2015b. Data Rods (Time Series Data). Retrieved November 1, 2015, from. http://disc.sci.gsfc.nasa.gov/hydrology/data-rods-time-series-data.
Goharian, E., Burian, S.J., 2014. Integrated urban water resources modeling in a semi-arid mountainous region using a cyber-infrastructure framework. In: Proceedings of 11th International Conference on Hydroinformatics (HIC2014), New York City, New York, USA. CUNY Academic Works. http://academicworks.cuny.edu/cc_conf_hic/230.
Goharian, E., Burian, S., Bardsley, T., Strong, C., 2015. Incorporating potential severity into vulnerability assessment of water supply systems under climate change conditions. J. Water Resour. Plann. Manage. 10.1061/(ASCE)WR.1943-5452.0000579, 04015051.
GoldSim Technology Group, 2015. Monte Carlo Simulation Software - GoldSim. Retrieved from. http://www.goldsim.com/Home/.
Goodrich, D.C., Guertin, D.P., Burns, I.S., Nearing, M.A., Stone, J.J., Wei, H., Pierson, F., 2008. RHEM Web Tool: Rangeland Hydrology Erosion Model Web Tool. Rangelands. Retrieved from. http://apps.tucson.ars.ag.gov/rhem/.
Google, 2014. Google Maps JavaScript API v3. Retrieved from. https://developers.google.com/maps/web/.
GRASS Development Team, 2014. GRASS GIS: the World's Leading Free GIS Software. Retrieved from. http://grass.osgeo.org/download/.
Hall, D., Salomonson, V., Riggs, G., 2006. MODIS/Terra Snow Cover Daily L3 Global 500m Grid. Version 5. National Snow and Ice Data Center, Boulder, Colorado USA.
Hazzard, E., 2011. OpenLayers 2.10: Beginner's Guide. Packt Publishing, Birmingham.
Highsoft AS, 2014. Highcharts - Interactive JavaScript Charts for Your Web Projects. Retrieved from. http://www.highcharts.com/.
Holl, S., Plum, H., 2009. PostGIS. GeoInformatics, 03/2009, 34–36 doi:citeulike-article-id:4463470.
Iacovella, S., Youngblood, B., 2013. GeoServer Beginner's Guide. Packt Publishing.
Jansson, P.-E., Moon, D.S., 2001. A coupled model of water, heat and mass transfer using object orientation to improve flexibility and functionality. Environ. Model. Softw. 16 (1), 37–46. http://dx.doi.org/10.1016/S1364-8152(00)00062-1.
Kadlec, J., Ames, D.P., 2014, December. Design of a high resolution open access global snow cover web map service using ground and satellite observations. In: AGU Fall Meeting Abstracts, 1, p. 1208.
Kadlec, J., Woodruff Miller, A., Ames, Daniel P., 2016. Extracting snow cover time series data from open access web mapping tile services. J. Am. Water Resour. Assoc. (JAWRA) 52 (4), 916–932. http://dx.doi.org/10.1111/1752-1688.12387.
Kulkarni, A., Mohanty, J., Eldho, T., Rao, E., Mohan, B., 2014. A web GIS based integrated flood assessment modeling tool for coastal urban watersheds. Comput. Geosci. 64, 7–14.
Lam, D., Leon, L., Hamilton, S., Crookshank, N., Bonin, D., Swayne, D., 2004. Multi-model integration in a decision support system: a technical user interface approach for watershed and lake management scenarios. Environ. Model. Softw. 19 (3), 317–324. http://dx.doi.org/10.1016/S1364-8152(03)00156-7.
Litzkow, M.J., Livny, M., Mutka, M.W., 1988, 13–17 Jun 1988. Condor-a hunter of idle workstations. Paper presented at the Distributed Computing Systems, 1988. In: 8th International Conference on.
Livny, M., Basney, J., Raman, R., Tannenbaum, T., 1997. Mechanisms for high throughput computing. SPEEDUP J. 11 (1), 36–40.
Mason, S.J.K., Cleveland, S.B., Llovet, P., Izurieta, C., Poole, G.C., 2014. A centralized tool for managing, archiving, and serving point-in-time data in ecological research laboratories. Environ. Model. Softw. 51 (0), 59–69. http://dx.doi.org/10.1016/j.envsoft.2013.09.008.
Michaelis, C., Ames, D.P., 2008. Web Mapping Service (WMS) web feature service (WFS) web processing service (WPS). In: Encyclopedia of GIS. Sashi Shekhar and Hui Xiong. Springer, New York, pp. 1259–1261.
Miller, S.N., Semmens, D.J., Goodrich, D.C., Hernandez, M., Miller, R.C., Kepner, W.G., Guertin, D.P., 2007. The automated geospatial watershed assessment tool. Environ. Model. Softw. 22 (3), 365–377. http://dx.doi.org/10.1016/j.envsoft.2005.12.004.
Millman, K.J., Aivazis, M., 2011. Python for scientists and engineers. Comput. Sci. Eng. 13 (2), 9–12.
Mitchell, K.E., 2004. The multi-institution North American Land Data Assimilation System (NLDAS): utilizing multiple GCIP products and partners in a continental distributed hydrological modeling system. J. Geophys. Res. 109 (D7), D07S90. http://dx.doi.org/10.1029/2003JD003823.
Nguyen, T.T., 2009. Indexing PostGIS databases and spatial query performance evalutations. Int. J. Geoinform. 5, 1–9.
Olaya, V., Gimenez, J.C., 2011. SEXTANTE, a Versatile Open-source Library for Spatial Data Analysis.
Oliphant, T.E., 2007. Python for scientific computing. Comput. Sci. Eng. 9 (3), 10–20.
Open Geospatial Consortium, 2015. Implementing Products. Retrieved from. http://www.opengeospatial.org/resource/products.
Open Knowledge Foundation, 2014. Ckan — the Open Source Data Portal Software. Retrieved from. http://ckan.org/.
Open Source Initiative, 2016. The Open Source Definition (Annotated). Retrieved from. http://opensource.org/osd-annotated.
Python Software Foundation, 2016. Python. Retrieved from. http://python.org/about/.
Rennaisance Computing Institute, 2016. Rennaisance Computing Institute. Retrieved from. http://renci.org/.
Rossman, L.A., 2000. EPANET 2 Users Manual, US Environmental Protection Agency. Water Supply and Water Resources Division. National Risk Management Research Laboratory, Cincinnati, OH, p. 45268.
Santhi, C., Srinivasan, R., Arnold, J.G., Williams, J.R., 2006. A modeling approach to evaluate the impacts of water quality management plans implemented in a watershed in Texas. Environ. Model. Softw. 21 (8), 1141–1157. http://dx.doi.org/10.1016/j.envsoft.2005.05.013.
Schut, P., Whiteside, A., 2007. OpenGIS Web Processing Service. OGC Project Document.
Simpson, J., Kummerow, C., Tao, W.-K., Adler, R.F., 1996. On the Tropical Rainfall Measuring Mission (TRMM). Meteorol. Atmos. Phys. 60 (1–3), 19–36. http://doi.org/10.1007/BF01029783.
Snow, A., 2015. A New Global Forecasting Model to Produce High-resolution Stream Forecasts. (Master of Science). Brigham Young University, Provo, Utah, USA.
Snow, A.D., Christensen, S.D., Swain, N.R., Nelson, E.J., Ames, D.P., Jones, N.L.,

Ding, D., Noman, N.S., David, C.H., Pappenberger, F., Zsoter, E., 2016. A high-resolution national-scale hydrologic forecast system from a global ensemble land surface model. J. Am. Water Resour. Assoc. 52 (4), 950–964. http://dx.doi.org/10.1111/1752-1688.12434.

Steiniger, Stefan, Hunter, Andrew J.S., May 2013. The 2012 free and open source GIS software map – a guide to facilitate research, development, and adoption. Comput. Environ. Urban Syst. 39, 136–150. ISSN 0198-9715 http://dx.doi.org/10.1016/j.compenvurbsys.2012.10.003.

Steiniger, S., Weibel, R., 2010. GIS Software: a Description in 1000 Word. Retrieved from London CB. http://www.zora.uzh.ch/41354/1/Steiniger_Weibel_GIS_Software_2010.pdf.

Sun, A., 2013. Enabling collaborative decision-making in watershed management using cloud-computing services. Environ. Model. Softw. 41, 93–97.

Swain, N.R., Latu, K., Christensen, S.D., Jones, N.L., Nelson, E.J., Ames, D.P., Williams, G.P., 2015. A review of open source software solutions for developing water resources web applications. Environ. Model. Softw. 67 (0), 108–117. http://dx.doi.org/10.1016/j.envsoft.2015.01.014.

Tapley, B.D., Bettadpur, S., Watkins, M., Reigber, C., 2004. The gravity recovery and climate experiment: mission overview and early results. Geophys. Res. Lett. 31 (9), 1–4. http://dx.doi.org/10.1029/2004GL019920.

Tarboton, D., Idaszak, R., Horsburgh, J., Heard, J., Ames, D., Goodall, J., Arrigo, J., 2014. HydroShare: advancing collaboration through hydrologic data and model sharing. In: Ames, D.P., Quinn, N. (Eds.), Paper Presented at the International Environmental Modelling and Software Society (IEMSs) 7th International Congress on Environmental Modelling and Software San Diego, California, USA. In: http://www.iemss.org/society/index.php/iemss-2014-proceedings.

Tethys Platform, 2016. Tethys Platform Project Steering Committee. Retrieved from. http://www.tethysplatform.org/project-steering-committee.

Thompson, C., Cechini, M., Huang, T., Roberts, J., Alarcon, C., Boller, R., Murphy, K., Plesea, L., Ilavajhala, S., Schmaltz, J., 2014. Gibs: a Rich Visual Interface to NASA's Earth Science Data Holdings, AGU Fall Meeting Abstracts, p. 3799.

USDA Forest Service, 2015. Uinta-wasatch-cache National Forest - Resource Management. Retrieved from. http://www.fs.usda.gov/detailfull/uwcnf/landmanagement/resourcemanagement/?cid=fsem_035491&width=full.

Walker, J.D., Chapra, S.C., 2014. A client-side web application for interactive environmental simulation modeling. Environ. Model. Softw. 55 (0), 49–60. http://dx.doi.org/10.1016/j.envsoft.2014.01.023.

Xia, Y., Ek, M., Wei, H., Meng, J., 2012. Comparative analysis of relationships between NLDAS-2 forcings and model outputs. Hydrol. Process. 26 (3), 467–474. http://dx.doi.org/10.1002/hyp.8240.