2019-12-01

Improving HydroShare and Web Application Interoperability Through Integrated GIS and HIS Data Services

Kenneth Jack Lippold
Brigham Young University

Improving HydroShare and Web Application Interoperability

Through Integrated GIS and HIS Data Services

Kenneth Jack Lippold

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Daniel P. Ames, Chair
E. James Nelson
Norm L. Jones

Department of Civil and Environmental Engineering

Brigham Young University

ABSTRACT

Improving HydroShare and Web Application Interoperability
Through Integrated GIS and HIS Data Services

Kenneth Jack Lippold
Department of Civil and Environmental Engineering, BYU
Master of Science

HydroShare is a collaborative online system being developed by the Consortium of Universities for the Advancement of Hydrologic Science Inc. (CUAHSI) with the goal of facilitating the dissemination, visualization, and publishing of hydrologic data and models. External web applications serve a key role in extending HydroShare's capabilities, so robust application programming interfaces (APIs) are a vital component of HydroShare's architecture. Hydrologic data stored on HydroShare are defined by a data type, and much of these data are either geospatial or time series data. Although HydroShare's API provides ways to upload and download files, as well as access certain metadata, it does not currently provide GIS services defined by the Open Geospatial Consortium, or Hydrologic Information System (HIS) services developed by CUAHSI. The absence of these services severely limits the capabilities of HydroShare apps while also increasing the development time and complexity of apps that are developed.

To help alleviate this disconnect between HydroShare and HydroShare apps, I have developed a system which helps extend HydroShare's data service capabilities using GeoServer and a Water Data Server to expose GIS and HIS data services for HydroShare content. With this system in place, HydroShare apps have much better access to HydroShare content, allowing them to be developed in less time, and provide much more powerful visualization, access, and analysis services to HydroShare users.

Keywords: HydroShare, GIS, HIS, data services, web applications

ACKNOWLEDGEMENTS

I want to thank the Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI) and Brigham Young University (BYU) for funding this research and providing a collaborative environment which has allowed this project and others to move forward. In particular, I would like to thank Dr. Zhiyu Li, Martin Seul, Dr. David Tarboton, Shawn Crawley, Matthew Bales, Chris Calloway, Scott Black, and everyone else in the BYU Hydroinformatics Lab and on the CUAHSI and HydroShare teams who have provided technical support and feedback on this project.

I would like to especially thank my graduate committee chair, Dr. Dan Ames, for his support throughout my undergraduate and graduate experience at BYU, and for enthusiastically supporting this research. I also want to thank the rest of my BYU graduate committee, Dr. Jim Nelson and Dr. Norm Jones, for their support and encouragement.

Lastly, I want to thank all my friends and family who have helped me get to where I am today. My parents, Nick and Beth, have always encouraged me to push forward and reach for my full potential. I wouldn't be where I am today without their love and support. Finally, I want to thank my wife Michelle. Her constant encouragement and support over the past few years has kept me going, and this work would not have been possible without her.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

Hydrologic and geospatial data plays a vital role in helping understand and model complex natural systems. Massive amounts of data are collected every day by agencies around the world, and collecting, organizing, and disseminating these data efficiently is an ongoing problem in the environmental sciences. Modern data systems frequently rely on service-oriented architectures to address this problem. Several organizations have attempted to develop data standards and web service systems to facilitate the dissemination of hydrologic and geospatial data.

One of these organizations is the Consortium of Universities for the Advancement of Hydrologic Science Inc. (CUAHSI), which has developed several frameworks and systems to help solve these issues, but there are still gaps in the capabilities of these systems which this research will address. One of these systems is HydroShare, which has been in development since 2012. HydroShare allows users to collaborate with other engineers and scientists from around the world by sharing hydrologic data and models (D. G. Tarboton et al., 2014). In addition to the core HydroShare system, HydroShare apps can also be developed to help extend HydroShare's capabilities.

HydroShare organizes data into resources and aggregations. Resources can be used to store data, models, or web app references. Certain resource data can be further organized into aggregations to help users discover certain types of data and define specific metadata for

different types of data (Horsburgh et al., 2015). Some available aggregation types defined by HydroShare include geographic feature, geographic raster, multidimensional, and time series aggregations. HydroShare uses an integrated Rule-Oriented Data System (iRODS) to store data files, which can either be generic, or associated with different aggregation types. HydroShare's iRODS based storage system allows users to upload diverse data files which HydroShare can extract metadata from, and then organize into resources and aggregations without modifying the original data format.

At its core, HydroShare enables users to share and discover hydrologic data. To help extend its capabilities further, HydroShare supports external web applications which can be developed to perform specific or complex tasks that fall outside HydroShare's primary functions. Many of these applications have been developed by the HydroShare community using Tethys Platform. Tethys was created to help simplify the development of water resources web applications (Swain, 2015). Tethys platform is built on top of Django, a Python-based framework for creating web apps, and includes several other technologies such as OpenLayers, PostGIS, and Bootstrap out of the box. One of the first Tethys applications developed for HydroShare was HydroShare GIS, an app designed to help HydroShare users visualize HydroShare geospatial data in a web environment (Crawley et al., 2017). Other apps, such as the HydroShare Resource Creator and Data Series Viewer help transfer, manage, and visualize environmental time series data.

Although many HydroShare apps are developed using Tethys platform, HydroShare apps can be developed on any platform. Generally, a HydroShare app is any web application external to HydroShare itself that either uses or creates HydroShare data or metadata in some form. Whether an app is developed using Tethys platform, or some other web app framework, all

HydroShare apps rely on HydroShare's Representation State Transfer (REST) API to interact with HydroShare data, either by downloading files, uploading files, or getting resource metadata. This architecture is shown in Figure 1-1. Because most HydroShare apps interact with HydroShare data, many apps need to be authenticated by HydroShare users in order to create or edit their data. Tethys Platform makes this easy by including built-in authentication capabilities that allow users to give Tethys apps permission to access their HydroShare data, but these features are available to any app developer who wants to include them in their app.

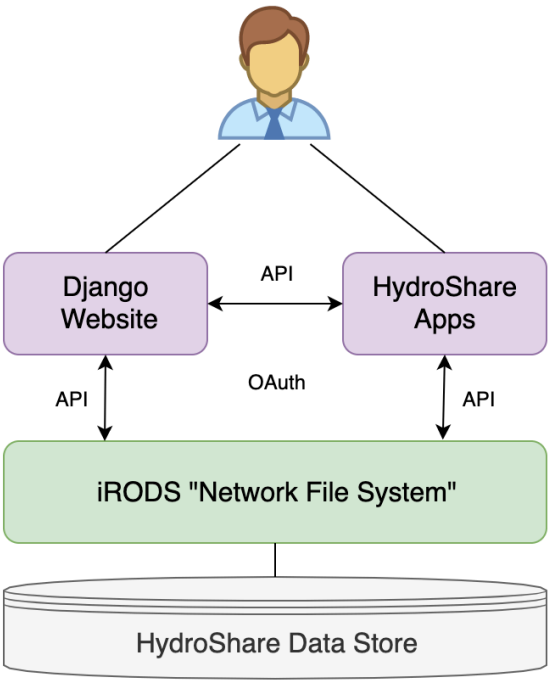

Figure 1-1: The architecture diagram for HydroShare.

One of the useful technologies included with Tethys Platform is GeoServer. Many Tethys apps rely on GeoServer to serve geospatial data over the web, which can be displayed in a web browser using a mapping library such as OpenLayers or Leaflet. GeoServer is a Java application that can expose geospatial databases over the web using data standards established by the Open

Geospatial Consortium (OGC). These standards include Web Feature Services (WFS), Web Map Services (WMS), Web Coverage Services (WCS), and Web Processing Services (WPS) (Youngblood & Iacovella, 2013). The WMS standard defines methods for serving map data over the web as georeferenced images, which are rendered by a server and can be displayed in a client ("Web Map Service," 2019). The WFS standard defines methods for serving geographic feature data over the web in a variety of formats including Shapefiles, GeoJSON, and KML ("Web Feature Service," 2019). The WCS standard defines methods for serving geographic raster data over the web in formats including GeoTIFF and other image formats ("Web Coverage Service," 2019). The WPS standard defines request and response formats for performing processing services over the web ("Web Processing Service," 2019). These standards are widely used and integrated into other GIS technologies such as Esri's ArcGIS Platform and QGIS.

To help create a system capable of sharing hydrologic data over the web, especially time series data, CUAHSI developed the Hydrologic Information System (HIS). The main goal of CUAHSI HIS is to facilitate the transfer of hydrologic data over the web (D. G. Tarboton et al., 2009). CUAHSI HIS consists of three main components whose relationship is shown in Figure 1-2; data publication, data discovery, and data access.

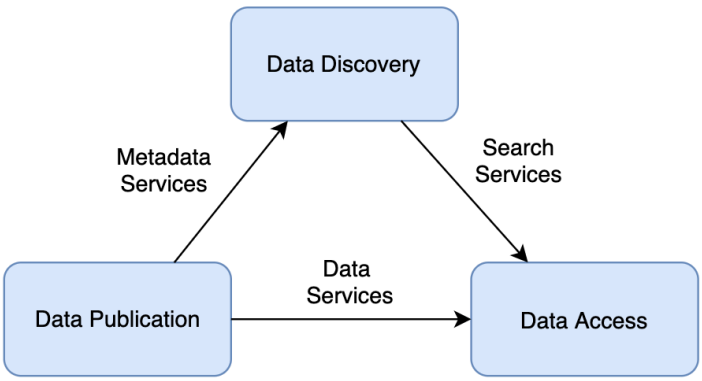

Figure 1-2: The component diagram for the CUAHSI Hydrologic Information System.

Data publication is handled by HydroServer, which uses WaterOneFlow (WOF) to standardize how applications exchange hydrologic data over the web ("WaterOneFlow Web Services & WaterML," 2010). HIS Central handles data discovery and uses metadata services to connect HydroServers (Horsburgh et al., 2010). Finally, data access is handled by various clients such as HydroDesktop (Ames et al., 2012) and the CUAHSI HydroClient. CUAHSI HIS is used by several large organizations, including NASA and the USGS to store and publish their data, but it is also used by smaller organizations such as universities to publish smaller datasets. The development of CUAHSI HIS preceded HydroShare, but they share similar goals. CUAHSI HIS offers more robust data services thanks to WaterOneFlow, but HydroShare is more accessible to users thanks to its web interface and social media capabilities.

Hydrologic and geospatial data services are powerful tools, but HydroShare does not currently have a way to expose data to the web in accordance with data standards such as OGC OWS and CUAHSI WOF. By not offering these services, Tethys apps and other external applications either can't interact as effectively with HydoShare data, or they must use complex workarounds to achieve a similar result. Offering these services through HydroShare would be advantageous for many reasons, and I will describe three of them below.

The first advantage of providing data services through HydroShare is improved levels of interoperability between it and other applications thanks to data standards. WMS, WFS, WCS, and WOF are all standards that, when followed properly, allow applications to transfer data between themselves in a reliable way. For example, many Tethys apps use mapping libraries like OpenLayers or Leaflet to display geospatial data. These libraries rely on OGC web service standards to correctly interpret incoming data, so they cannot import data directly from HydroShare. Apps such as HydroShare GIS get around this issue by downloading HydroShare

data to a Tethys server and then uploading it to a separate Tethys GeoServer before displaying the data in OpenLayers. This workaround is only possible because Tethys Platform includes GeoServer but would not work for simpler single-page applications not developed with Tethys Platform. Even for Tethys apps, the overhead required to deploy and maintain a GeoServer may make this solution difficult for some organizations that use Tethys Platform to deploy apps.

Other GIS applications such as ArcGIS Pro and QGIS can interact with OGC web services, but in order to ingest HydroShare data, the user must download the files to their computer before opening them in these programs. In the case of ArcGIS Online, a user would need to download files from HydroShare and then reupload them to ArcGIS Online, which can be a time-intensive process for large amounts of data. This type of workflow can be done entirely in the cloud if HydroShare offers geospatial data services.

The second advantage of data services is their ability to subset data. Hydrologic time series and geospatial data are often large and complex. Individual datasets often require many megabytes of storage, and data stored in a HydroShare resource can often approach or even exceed a gigabyte. Collectively, these data require terabytes of memory to store. HydroShare uses Esri Shapefiles to store geographic feature data, GeoTIFF files to store geographic raster data, and ODM2 SQLite files to store hydrologic time series data. Individually, these types of files can be very large, but they generally include internal metadata that allows the whole dataset to be divided into smaller subsets of data. Shapefiles contain subsets of data called features, GeoTIFF files can be sliced into smaller chunks using different coverage extents, and ODM2 SQLite files can be queried using metadata such as site, variable, or temporal extent. HydroShare doesn't provide a way to request and retrieve data based on this metadata, but data services do. Both OGC web services and CUAHSI's WaterOneFlow include methods for requesting subsets

of data stored in a database based on these parameters. In cases where a database is very large, but an application only needs a small portion of the data at a given time, data services allow for only that portion to be requested and transferred. As a result, instead of frequently transferring large files between applications, it becomes possible to transfer small chunks of data at a time. This is especially important for web applications where web browser storage space can be very limited, but also useful for desktop applications when a user doesn't want to dedicate a large amount of memory to the data.

Finally, the third advantage of these data services is the ability to provide data in a wide variety of formats. When a user or application requests data from HydroShare, HydroShare can only return the data in the format it was uploaded in. For the data being discussed here, that means Shapefiles, GeoTIFF files, and ODM2 SQLite files. Each of these file formats are useful for storing data, but they are not efficient, or even compatible, with all applications. This is especially true for web applications where the types of files that can be opened is limited by available JavaScript libraries and web browser capabilities. WCS and WMS services can return data in a variety of image formats, WFS services can return data as GeoJSON or KML, and WOF can return data as XML or potentially JSON. These standards are generally technology neutral, so they can be further extended to include other file types as technology advances. These types of formats are much more compatible in web environments than their native formats, which simplifies their use and offers greater flexibility in Tethys apps and other applications.

HydroShare can benefit greatly by integrating OGC and CUAHSI data services into its core architecture. The objectives of this research effort include: 1) develop a technology neutral data services framework for HydroShare 2) implement this framework by deploying a GeoServer and a Water Data Server that are linked to HydroShare's data repository and expose these data to

the web in accordance with OGC and CUAHSI data standards 3) demonstrate the capabilities of this system with a Jupyter Notebook designed to view HIS time series data, a demo showing how to connect to HydroShare data to Esri's ArcGIS Platform, a HydroShare Data Viewer application to visualize these data in a web browser, and a HydroShare Time Series Manager application to facilitate the management of HydroShare time series data using time series data services.

2 HYDROSHARE HYDROLOGIC AND GEOSPATIAL DATA SERVICES

2.1 Data Services Framework

Many platforms have been developed to serve data over the web in accordance with various data standards. For geospatial data, some of these technologies include GeoServer, Esri ArcGIS Server ("What is ArcGIS Server," 2019), MapBox Atlas ("Atlas," 2019), and many more. For hydrologic time series data, CUAHSI has developed or been involved with the development of HydroServer (Horsburgh et al., 2010), HydroServer Lite (Conner et al., 2013), and WOFpy (Wilson & Pothina, 2011). Additionally, there are several other non-hydrologic time series data servers such as TimescaleDB ("TimescaleDB Overview," 2019), OpenTSDB ("How does OpenTSDB work?," 2019), and InfluxDB ("InfluxDB Documentation," 2019). These services each involve different technology stacks, some of which are free and open source, while others are paid subscription-based platforms. Before I describe my implementation of data services for HydroShare, I want to present it as a technology neutral framework that could ultimately be implemented using any combination of technology platforms and services.

Figure 2-1 below shows how this framework extends HydroShare's architecture shown in Figure 1-1. The key difference between HydroShare's original architecture and the updated architecture shown below is that without implementing data services, HydroShare is essentially a file-sharing platform that provides useful discovery metadata. This makes HydroShare a very powerful tool for data discovery, and it makes data accessible to users in exactly the format that

the owner of the data uploaded to the system. The data services framework goes a step further by giving users and applications much more control over what can be done with the data once it has been discovered. Rather than simply being able to download files, it becomes possible to access the data contained in the files with more flexibility and greater granularity.

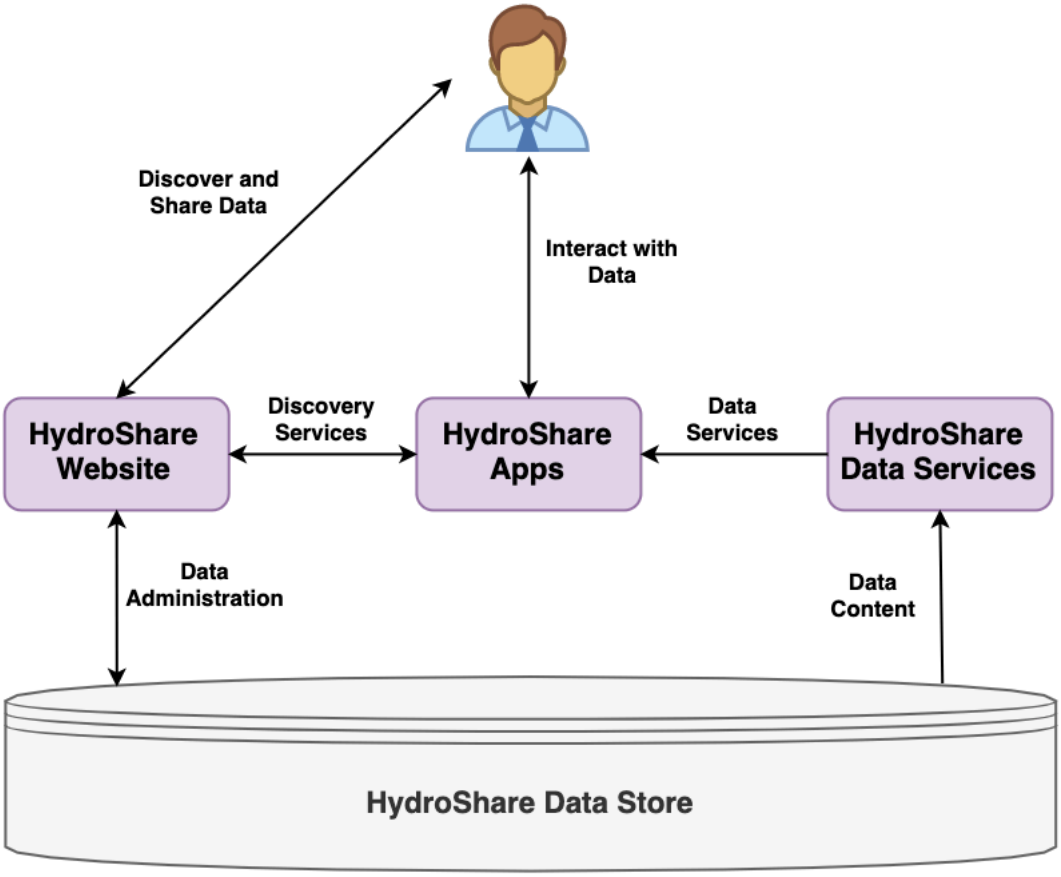

Figure 2-1: A technology neutral diagram of the HydroShare data services framework.

HydroShare data services in general are extensible and can be applied to any type of data HydroShare supports or may support in the future. Figure 2-2 shows a general model for how a data service can access HydroShare data, shown in blue, transform it in accordance with some data standard, shown in yellow, and serve it to an application or user in an alternative,

transformed, or subset output, shown in green. As I describe my implementation of this data services framework, I will use this model applied to specific data services. Using this framework, more data services can be added to HydroShare to support additional data types, and existing data services can evolve alongside HydroShare as new technologies are adapted for the project in the future.

Figure 2-2: A diagram of the general HydroShare data services model.

2.2 Data Services Implementation

For the initial implementation of HydroShare data services, I have chosen to focus on geospatial and time series data, which are both commonly used hydrologic data formats. I have chosen to use GeoServer to handle geospatial data, and a Water Data Server to handle hydrologic time series data. Later in this paper, I will explain why I chose each of these servers and how they function. First, I will describe how these services fit into the general data services framework.

One of HydroShare's core architectural components is the iRODS federated file system (Bedard, 2015). HydroShare places, edits, and retrieves resource files in iRODS using the iCommands API. Resource files are physically stored in an iRODS zone located on a RENCI server along with the HydroShare application, with iCommands allowing them to communicate.

GeoServer and the Water Data Server need to have read access to the resource files in order to expose them to the web but doing so bypasses the iCommands API. This limitation leaves two methods for providing file access to GeoServer and the Water Data Server.

The first method is to use the iCommands API to copy files from iRODS storage to a directory that GeoServer and the Water Data Server have access to, which is illustrated in Figure 2-3. Although this method does not bypass the iRODS system, it is difficult to scale because at least two copies of each data file will need to be maintained. In addition to increasing the required storage space for these files, it also introduces synchronization issues since the copied files cannot be edited by iRODS. This means that if any changes are made to the file or its metadata in iRODS, a separate process would need make those changes to the copied data. Another major drawback of this solution is speed. Using HydroShare's API to download files to external locations takes additional time as the size of the file increases.

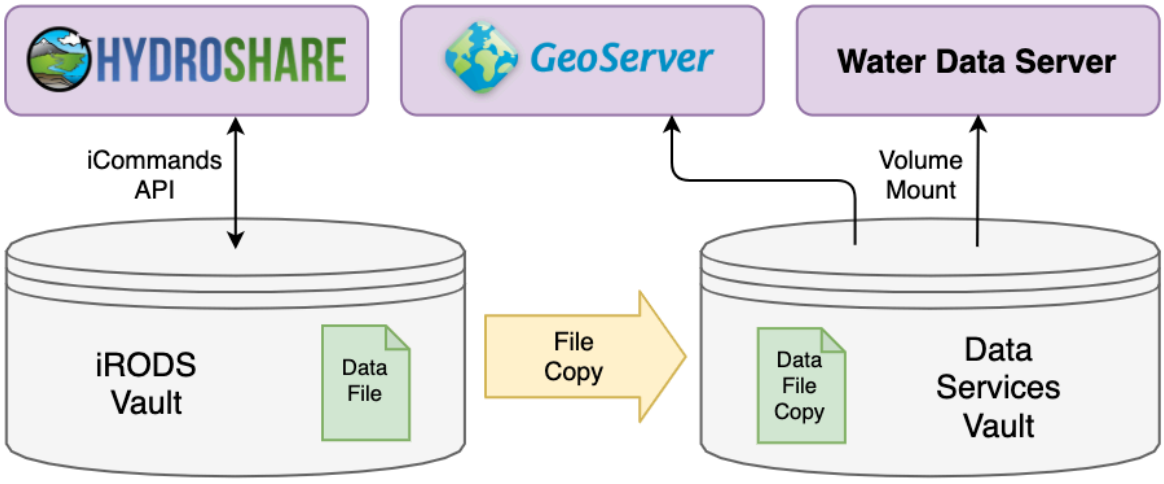

Figure 2-3: The data services file copy method.

The second method is to allow GeoServer and the Water Data Server to share data with HydroShare by exposing the data files to the servers. This method helps alleviate

synchronization, storage, and speed issues because only one copy of the data needs to be maintained, and no copying of the data is necessary. There are two ways to allow these applications to share data. The first is to use an iRODS extension called Fuse to expose iRODS data to external applications (Bedard, 2015), and the second is to bypass iRODS and mount the data vault directly into the external applications.

The GABBS Project uses Fuse to help integrate HUBzero, iRODS, and GeoServer, but they decided to expose iRODS files directly to GeoServer for efficiency reasons (Kalyanam et al., 2016). Although Fuse can be used to expose files to external systems like GeoServer, it reduces the speed at which these systems can read files. For an application like GeoServer, which frequently needs to read large amounts of data from files, this reduction in speed results in decreased performance.

Mounting the iRODS data into GeoServer and the Water Data Server is useful because there is no reduction in speed, but the solution does introduce two other major issues. The first is that it bypasses the iCommands API, possibly exposing the data to unauthorized modifications by the data services application. This issue can be resolved by simply using a read-only mount of the data, ensuring the data are readable but not writable by any applications. If GeoServer or the Water Data Server need to create files, such as index files, they can be created in different directories (GeoServer, 2013). The other issue is that this solution only works if the data are stored in a single location. One of the defining features of iRODS is its ability to distribute the storage of data files in different locations (Bedard, 2015). Although this feature is supported by iRODS, HydroShare uses iRODS to store all data in a single location. In the event that HydroShare ever does use distributed data storage, one of the other methods mentioned above (file copy or Fuse) would need to be used for any data GeoServer and the Water Data Server

don't have physical access to. Figure 2-4 illustrates how the data volume mount method would work.

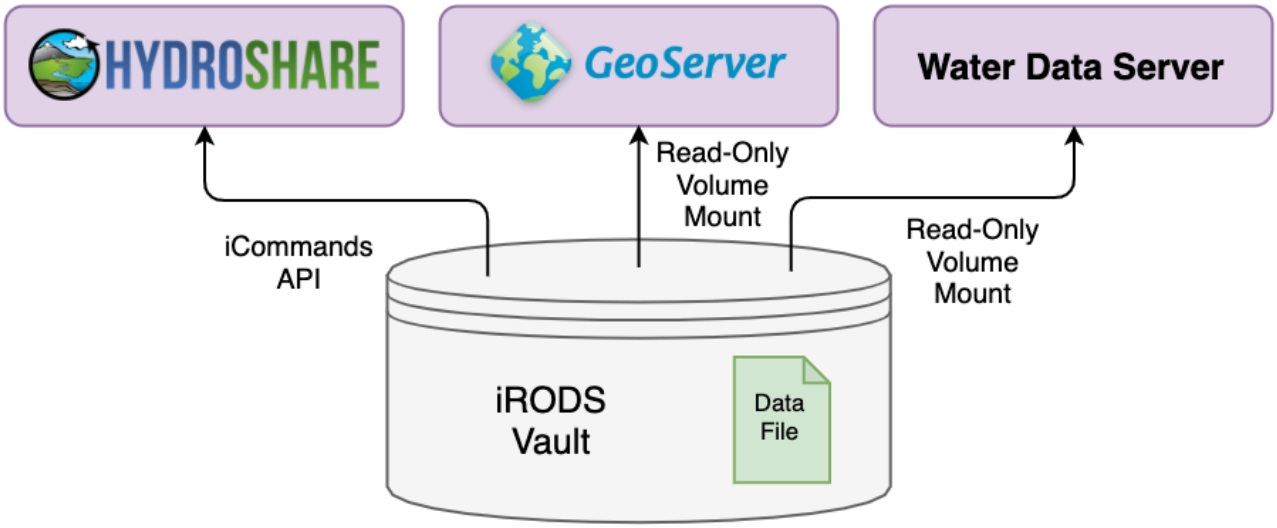

Figure 2-4: The data services volume mount method.

I have decided to use the data volume mount method because it provides superior performance and because HydroShare stores its data in one central location. Volume mounts will expose HydroShare data to GeoServer and the Water Data Server, but HydroShare needs to be able to register the appropriate data with each of these servers before data can be exposed via the data servers.

This process will require an intermediate system, the web services manager. To simplify development, this component will be run separate from HydroShare but could be included as a core HydroShare module in the future. Whenever data on HydroShare is assigned a valid content type and becomes public, HydroShare will notify the web services manager to register that data on the data service systems. If other types of data servers are added to HydroShare, they would communicate with HydroShare through the web services manager similar to GeoServer and the

Water Data Server. This process requires that any data servers used to serve HydroShare data have an API that the web services manager can use to register data. The architecture for this framework is shown in Figure 2-5.

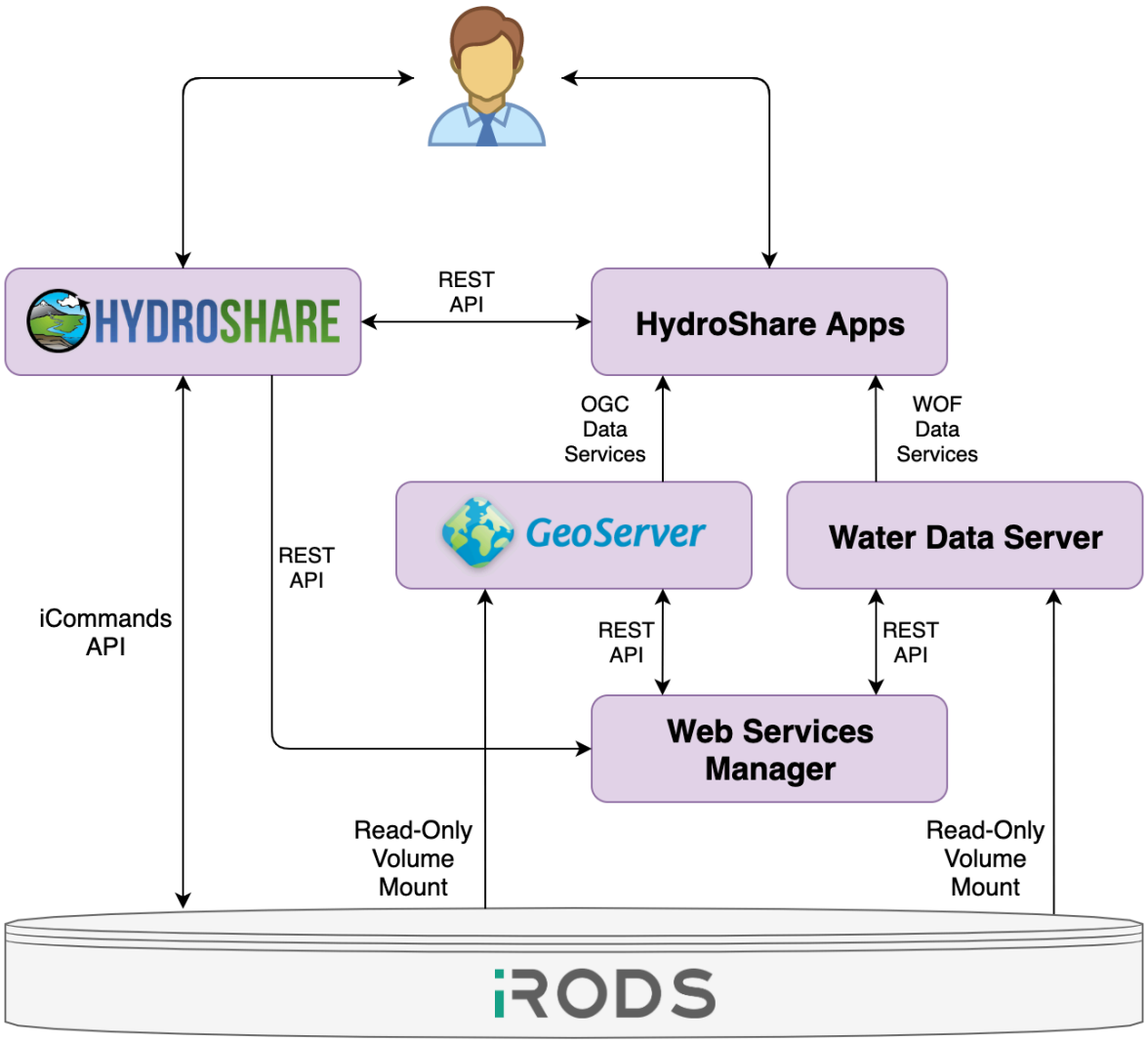

Figure 2-5: The HydroShare data services architecture.

Figure 2-6 illustrates the process for registering data on GeoServer. The process is designed to be mostly invisible to the end user. When a user creates a resource with valid data and makes it public, the system will register the data automatically and communicate the

existence of the data services to the user through HydroShare's resource landing page. The web services manager prepares requests to register data on the GeoServer, and it will handle errors raised by either of these data servers. The process is similar for the Water Data Server.

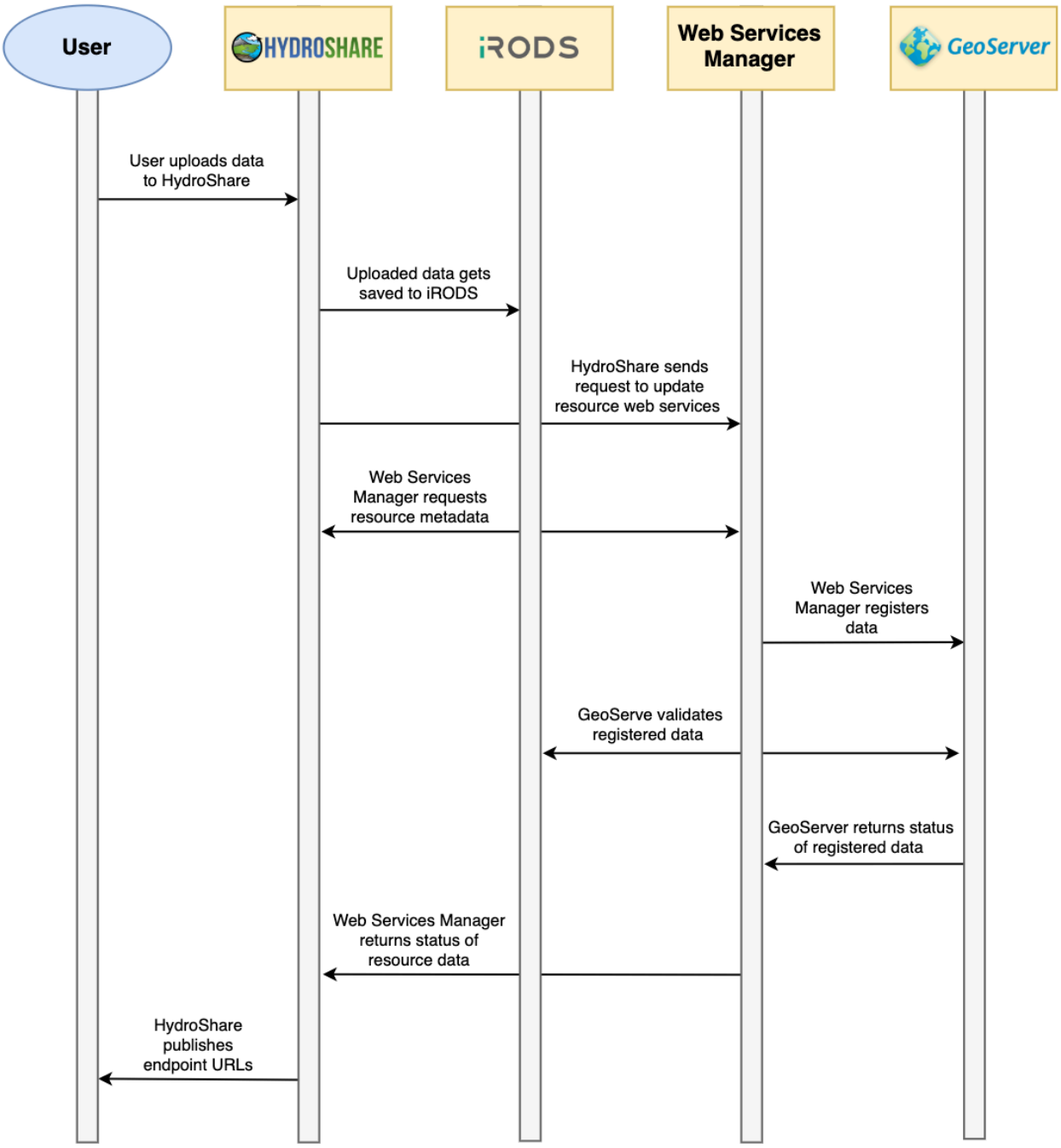

Figure 2-6: The HydroShare data services registration process.

Each of these components work together using their REST APIs. When HydroShare sends a request to the web services manager to register or unregister data, it only includes the resource ID in the request. The web services manager then uses HydroShare's REST API to determine what changes, if any, need to be made. This is done to minimize the modifications to HydroShare necessary for deploying this system. Data registration will be handled automatically in real time so other applications will be able to take advantage of these services when using HydroShare's REST API to upload or edit resource content. Because there is an intermediate service handling data registration, there is still some potential for metadata on HydroShare and registered on the GeoServer and the Water Data Server to become out of sync. Since the data are shared, the data itself can't be out of sync, but the URLs published on the resource landing page could potentially become out of sync with available data. It will be necessary to run a periodic batch process to resynchronize any out of sync metadata.

Currently, one of the limitations of this framework is that in order for data to be published by GeoServer and the Water Data Server, the HydroShare resource must be public. Applications can use OAuth2 to access restricted HydroShare data, and GeoServer has an OAuth2 module available (GeoServer, 2013; OAuth2, 2019). It may be possible to leverage these capabilities to allow GeoServer to require HydroShare authentication to access private data. Although this may be possible, many third-party systems, notably ArcGIS Pro and ArcGIS Online, do not provide a way to use OAuth2 to access restricted OGC web services. Instead, they require users to provide a username and password to access the data. In other words, while it may be possible to expose data services on private HydroShare data, its usefulness in many external systems would be severely limited.

2.3 HydroShare Web Services Manager

2.3.1 Web Services Manager Design

The web services manager consists of two components. The first is an update to HydroShare's codebase to generate update requests, and the second is a system that receives the requests and registers data on GeoServer and the Water Data Server. HydroShare is a Django application, and Django has a feature called signals. Signals can be triggered throughout the application, and receivers handle the signals ("Django Documentation," 2019). HydroShare will generate a signal and send an update request to the web services manager after the following events; a file aggregation is added, a file aggregation is removed, a resource is deleted, or a resource's access level changes. The update request simply contains the resource ID of the affected resource. Since the web services manager is an external application, the requests are performed asynchronously to avoid slowing down HydroShare. Due to the asynchronous nature of the requests, there may be a short delay between editing the resource and seeing any available data services on HydroShare. Generally, the delay will not be noticeable to the user because no data needs to be transferred to register it on GeoServer or the Water Data Server. Most delays will be due to GeoServer's data validation process, which becomes more noticeable with larger data files.

The web services manager application is built using Django and the Django Rest Framework ("Django Documentation," 2019). The application has only one request endpoint for updating resource data registration. HydroShare uses an OAuth2 authentication token to send the update requests, so only HydroShare will be able to trigger any changes to the data services. When one of these requests is received, the web services manager will use HydroShare's REST

API to determine if data should be registered or unregistered, and whether it should be registered on GeoServer or the Water Data Server.

The web services manager needs to be able to determine the content type of resource data files. HydroShare's REST API does not provide endpoints for determining resource aggregation information, so this project included an update to HydroShare's GET file list request to include the content type of each file in the resource. The web services manager uses this information to filter geographic feature, geographic raster, and time series files into a list. Geographic feature and geographic raster content are registered on GeoServer, and time series content is registered on the Water Data Server. If the resource is private or no longer exists, any content associated with the resource is removed from GeoServer and the Water Data Server.

Because GeoServer's data validation process differs from HydroShare's, it is possible that data HydroShare sees as valid may not be able to be registered on GeoServer. The web services manager is designed to handle as many of these errors as possible but sometimes a request to register HydroShare content on GeoServer will not be able to finish successfully. In these situations, the web services manager will return a description of the error back to HydroShare where it can be communicated to the user. Some errors, such as file naming errors, can be easily fixed while others may be more complex and require another program like ArcGIS Pro to fix.

Once data has been registered by the web services manager, it returns information about what was registered back to HydroShare. Both HydroShare resources and aggregations can be assigned metadata. Based on information returned from the web services manager, HydroShare will update associated resource and aggregation metadata to let the user know about the available data services.

2.3.2 Web Services Manager Implementation

The web services manager is a Django application that uses the Django REST Framework to interact with HydroShare. The Django application is run using Gunicorn as a web server gateway interface and NGINX as a reverse proxy service (Chesneau, 2018; "NGINX Documentation," 2019). All of these processes are managed by Supervisord and contained inside a Docker container to simplify deployment ("Docker Documentation," 2019; "Supervisor: A Process Control System," 2019). This structure is illustrated in Figure 2-7.

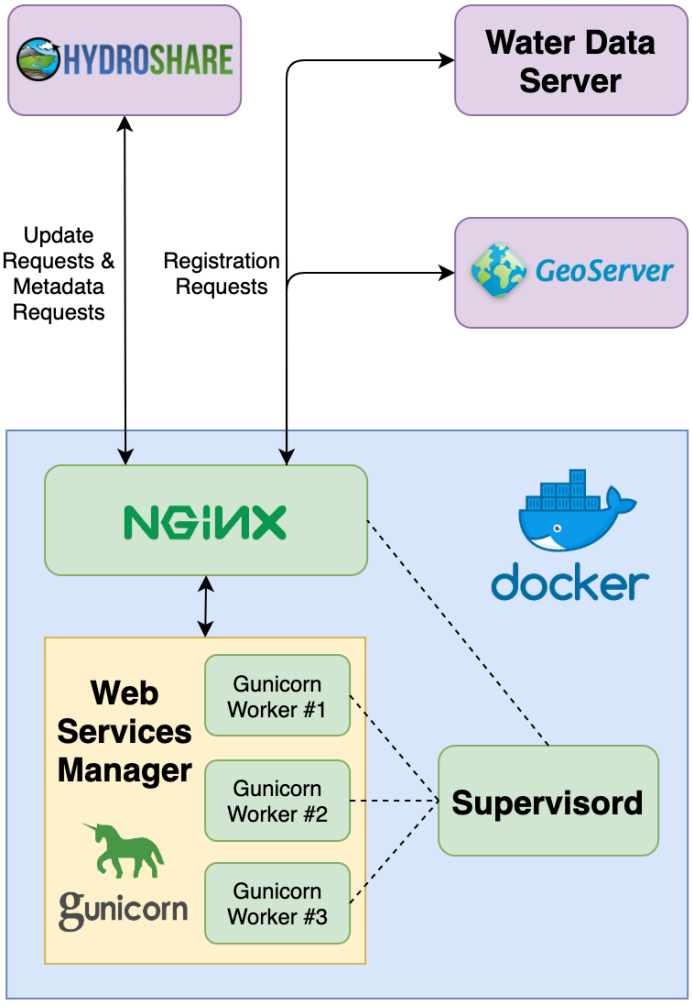

Figure 2-7: A diagram of the web services manager implementation architecture.

2.4 GeoServer

2.4.1 GeoServer Background

GeoServer is an open-source Java application designed to serve geospatial data over the web in accordance with OGC web service standards (OWS), including WFS, WMS, WCS, and WPS (Youngblood & Iacovella, 2013). In addition to providing OWS services, GeoServer also includes a REST interface and a graphical web interface for configuring the server and data. Because GeoServer serves OWS compliant data, it is compatible with many other geospatial software packages including OpenLayers ("OpenLayers Documentation," 2019), Leaflet (Agafonkin, 2019), QGIS ("QGIS Documentation," 2019), and ArcGIS Platform ("Documentation for ArcGIS," 2017). GeoServer uses a software package called GeoTools to support both geographic vector and geographic raster data ("GeoTools Documentation," 2019). GeoServer supports many database source types, including Esri Shapefiles, OGC GeoPackages, PostGIS servers, GeoTIFF files, and NetCDF files. Figure 2-8 illustrates a typical function of GeoServer in a geospatial web application stack, where GeoServer acts as an intermediate service between a data server and a web client.

Figure 2-8: A diagram of GeoServer in a typical web app stack.

GeoServer is integrated into Tethys Platform, and many Tethys apps rely on services provided by GeoServer to let users interact with geospatial data (Swain, 2015). Tethys Platform

uses Docker to deploy an instance of GeoServer alongside the Tethys server that apps can communicate with using a Python REST API wrapper. Geospatial data for Tethys apps can either be stored inside the GeoServer container or stored in a PostGIS database, a service also supported by Tethys. Tethys also integrates with HydroShare using OAuth2, allowing apps to access HydroShare data, but only through HydroShare's REST API, so any HydroShare data needed by a Tethys app must be downloaded to the Tethys server, and if it is geospatial data, uploaded to GeoServer or converted to a PostGIS table.

2.4.2 GeoServer Implementation

HydroShare uses Shapefiles to store geographic feature (vector) data, and GeoTIFF files to store geographic raster data. GeoServer recognizes both Shapefiles and GeoTIFF files as valid database files for serving vector data and raster data respectively. This similarity means HydroShare geospatial aggregation data stored in iRODS can be mounted into and registered with GeoServer. Shapefiles can be registered as data stores, GeoTIFF files can be registered as coverage stores, and both can be published as layers in GeoServer.

GeoServer also optionally allows data stores, coverage stores, and layers to be associated with a workspace to help organize the data. When the web services manager registers data on GeoServer, it also creates a workspace to associate the data with. The name of the workspace is the HydroShare resource ID preceded by the characters "HS-" because GeoServer workspaces cannot begin with a number. Due to the way GeoServer implements the WFS standard, workspace names are used in WFS XML response tags, and XML does not allow tag names to begin with a number.

The result is an organizational structure on GeoServer that can be mapped to HydroShare's with a list of workspace IDs (representing resource IDs) containing data stores,

coverage stores, and layers associated with HydroShare aggregations, as shown in Figure 2-10. GeoServer will provide OGC WMS and WFS services for HydroShare's geographic feature data and OGC WMS and WCS services for HydroShare's geographic raster data, as shown in Figure 2-9. The GetCapabilities URLs for these services should be published by HydroShare in some form to help users integrate these data into their workflow, but the publishing of URLs on HydroShare's resource landing page is not necessary for HydroShare apps to access the data services. An example of these URLs being published on HydroShare's resource landing page is shown in Figure 2-11.

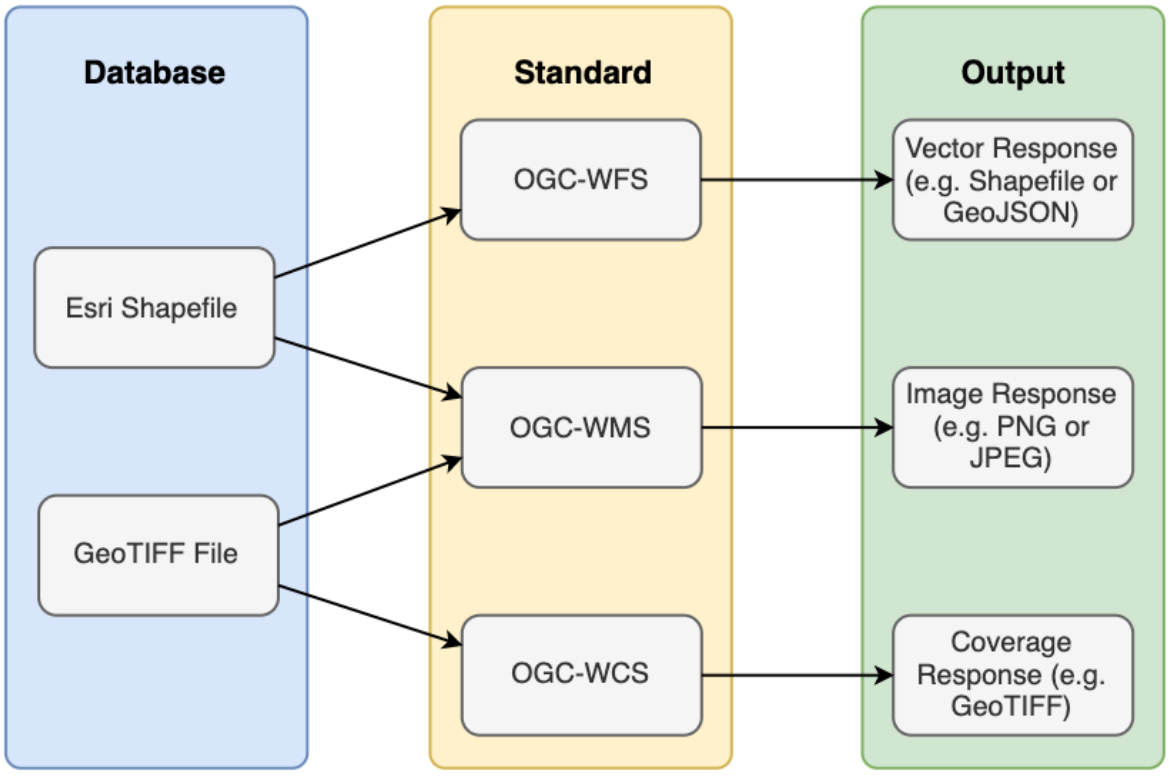

Figure 2-9: A diagram of the HydroShare GeoServer data services model.

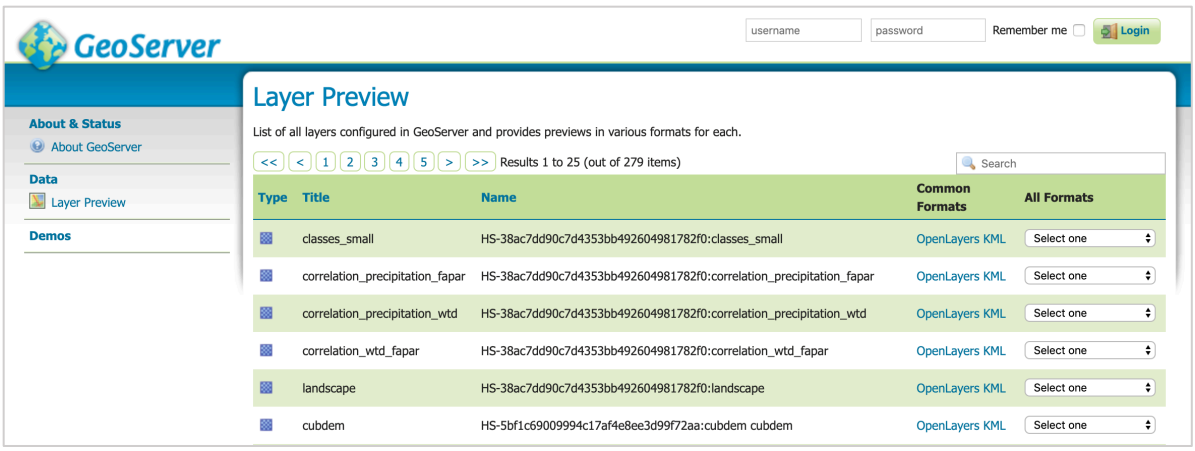

Figure 2-10: A screenshot of GeoServer's layer preview page with HydroShare data.

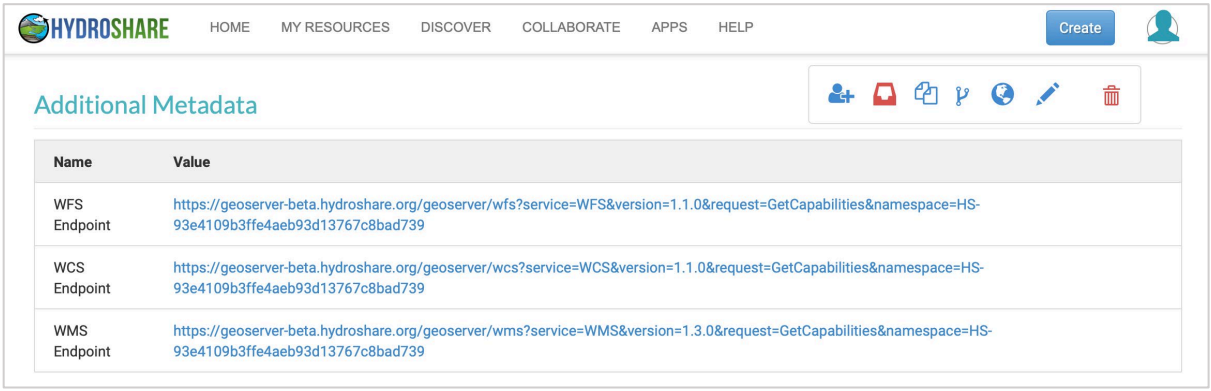

Figure 2-11: A screenshot of OWS URLs on HydroShare's resource landing page.

I have chosen to implement GeoServer for HydroShare based on how it was implemented in Tethys Platform (Swain, 2015). GeoServer is run as an Apache Tomcat server behind NGINX to handle incoming and outgoing requests, and Supervisord is used to manage these components inside a Docker container. The HydroShare iRODS data are mounted inside the container as a read-only volume for GeoServer to access to ensure GeoServer does not write any files to the data vault. This architecture is illustrated in Figure 2-12.

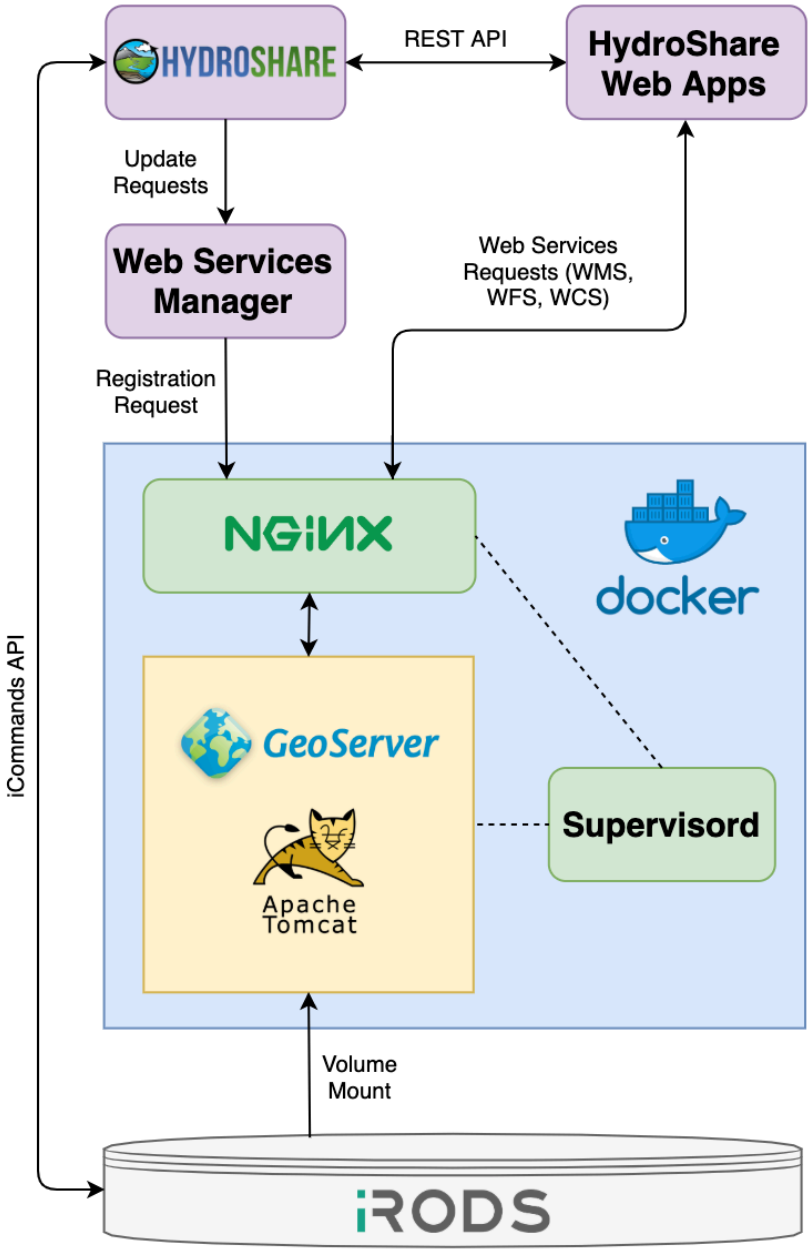

Figure 2-12: A diagram of the HydroShare GeoServer implementation architecture.

GeoServer does not automatically parse raster data to determine raster statistics. GeoServer uses Styled Layer Descriptor (SLD) files to define styling information for WMS requests. An example SLD file for a raster layer is shown in Figure 2-13, and they require raster statistics to function properly. Without this information, GeoServer's default raster styling will

not be compatible with registered raster data and WMS requests will return blank images similar to the image shown on the left of Figure 2-14. To alleviate this issue, the web services manager will use HydroShare's REST API to determine minimum, maximum, and no data values for the raster and build a custom styled layer descriptor file to upload to GeoServer resulting in WMS results similar to the image shown on the right of Figure 2-14. Although users and web clients can define their own styles to be used for WMS requests, including a usable default style can help simplify their use.

```
<?xml version="1.0" encoding="UTF-8"?><sld:StyledLayerDescriptor xmlns="
http://www.opengis.net/sld" xmlns:sld="http://www.opengis.net/sld" xmlns:gml
="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc" version
="1.0.0">
  <sld:NamedLayer>
    <sld:Name>logan</sld:Name>
    <sld:UserStyle>
      <sld:Name>logan</sld:Name>
      <sld:Title>Default raster style</sld:Title>
      <sld:Abstract>Default greyscale raster style</sld:Abstract>
      <sld:FeatureTypeStyle>
        <sld:Name>name</sld:Name>
        <sld:Rule>
          <sld:RasterSymbolizer>
            <sld:ColorMap>
              <sld:ColorMapEntry color="#000000" opacity="0.0" quantity="
              -3.402823466385289e+38" label="nodata"/>
              <sld:ColorMapEntry color="#000000" quantity="1358.3345947266"
              label="values"/>
              <sld:ColorMapEntry color="#FFFFFF" quantity="3031.4431152344"
              label="values"/>
            </sld:ColorMap>
            <sld:ContrastEnhancement/>
          </sld:RasterSymbolizer>
        </sld:Rule>
      </sld:FeatureTypeStyle>
    </sld:UserStyle>
  </sld:NamedLayer>
</sld:StyledLayerDescriptor>

```

Figure 2-13: An example SLD file for a GeoServer raster layer.

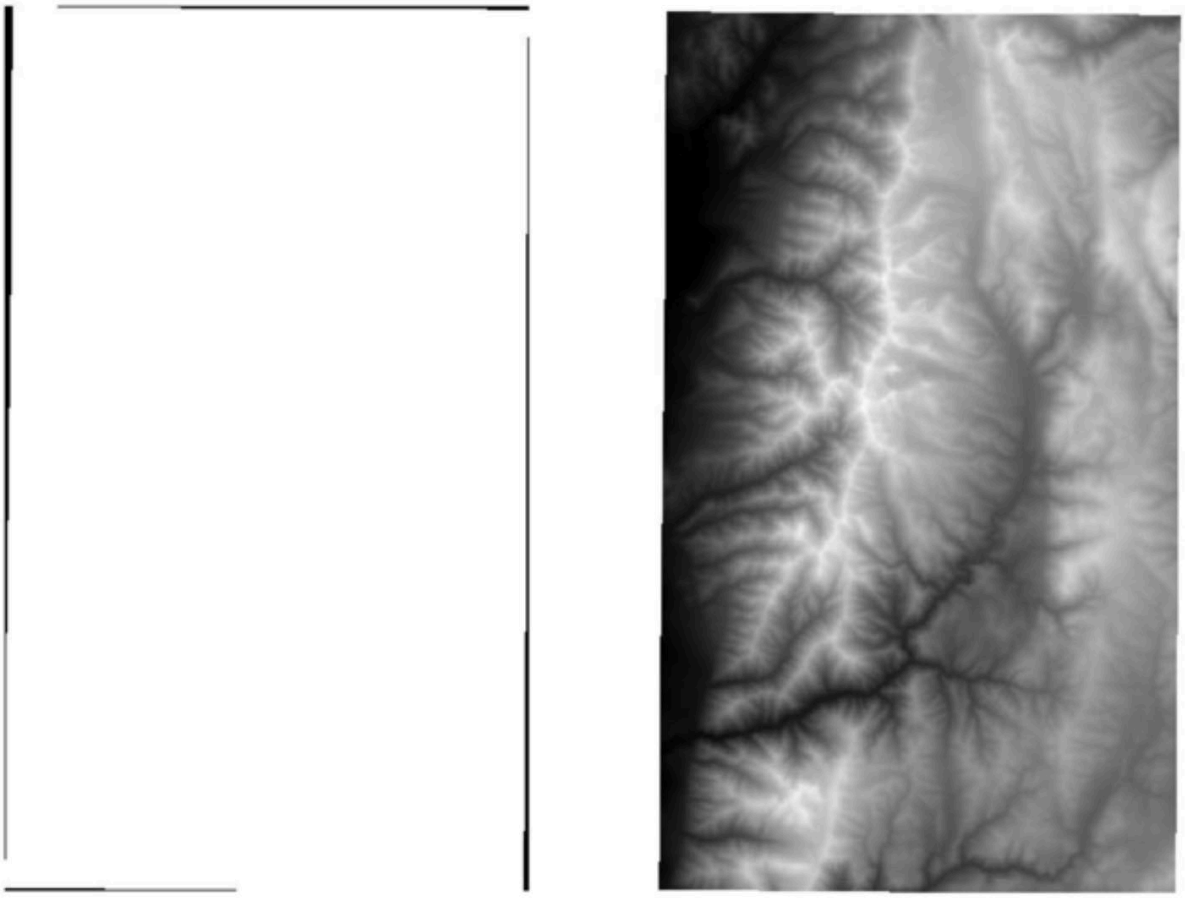

Figure 2-14: A WMS response without an SLD (Left) and with an SLD (Right).

2.5 Water Data Server

2.5.1 Water Data Server Background

The design of the Water Data Server is loosely based on the GeoServer architecture but should fill a role similar to HydroServer in CUAHSI's Hydrologic Information System. HydroServer can serve hydrologic time series data via WaterOneFlow data services, but it was not designed to serve data stored on HydroShare (Horsburgh et al., 2010). Although HydroShare

uses the ODM2 data model to define hydrologic time series data, it stores that data in SQLite files (Sadler et al., 2016). These files are organized into resources and aggregations, all of which can be added to or removed from HydroShare on the fly by their owners. HydroServer was designed to serve data from MySQL servers, and although the data could be updated, the connections cannot be updated on the fly. HydroServer LITE was designed as a lightweight version of HydroServer that could be more easily deployed and maintained, but it was not designed to read data from SQLite files (Conner et al., 2013). WOFpy is a Python wrapper for WaterOneFlow, and it provides more flexibility than HydroServer and HydroServer LITE (Wilson & Pothina, 2011). HydroServer, HydroServer LITE, and WOFpy all essentially follow the model shown in Figure 2-15, where they connect to some backend database, then use WaterOneFlow to serve the data as WaterML using the SOAP protocol.

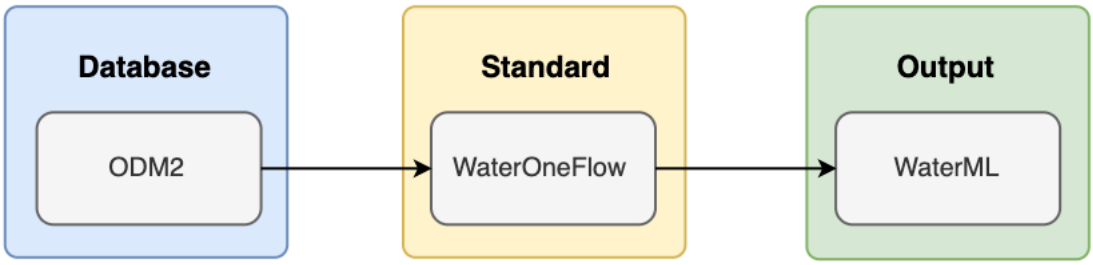

Figure 2-15: A diagram of the HydroServer data services model.

WOFpy is a Python Flask application and can serve data from a variety of backend sources, including ODM2 SQLite files. It also includes a beta feature for serving data via REST as well as SOAP. In these ways, WOFpy is more flexible than HydroServer and HydroServer LITE, but the main barrier to using it in HydroShare's data services framework is its inability to register and unregister databases on the fly. HydroServer, HydroServer LITE, and WOFpy all

treat database connections as static, requiring a manual process to set them up. Systems that use SOAP are also limited in the types of responses they can provide, as they can only return XML data (W3C, 2007). According to the SOAP protocol, WaterOneFlow services must return WaterML.

2.5.2 Water Data Server Design

I designed the Water Data Server to fulfill the time series requirements of the HydroShare data services implementation. The Water Data Server can register and unregister time series databases on the fly, unlike the previously mentioned servers which require databases to be registered manually. In addition to this feature, I have also designed the Water Data Server to be more flexible than HydroServer, HydroServer Lite, and WOFpy in terms of the databases and outputs it supports. The Water Data Server framework is comprised of three main components; database models, data services data access objects, and response formats.

The database model refers to the data formats that the server can read in data from. This would include a combination of both the data model and the file/server type. For example, ODM2 is a data model, but it can be represented as MySQL, PostgreSQL, or SQLite. NetCDF is another example of a database that can contain time series data, but it is not related to ODM2 and it uses a unique file type (Rew & Davis, 1990). The Water Data Server can be extended to read time series data from NetCDF files or other time series data formats as long as the format can be mapped to the second component; the data services data access object. This component represents the data services standard being used to serve the data, and the Water Data Server can be extended to support any number of them. The data services data access object represents a data structure for serving data, such as WaterOneFlow. This data access object should define what types of requests can be made to the server and what data are necessary to respond to those

requests. For example, in WaterOneFlow there are five types of requests that can be made, each of which requires certain data from the database. I have modified the requests slightly because the Water Data Server uses REST rather than SOAP to handle requests. Table 2-1 shows the original WaterOneFlow SOAP requests, the modified REST requests, and a description of each of them.

Table 2-1: WaterOneFlow SOAP vs REST

SOAP Request	REST Operation	REST Object	Description
GetSiteInfo	GET	site	Returns metadata for a specific site.
GetSiteInfoObject	N/A	N/A	Returns metadata for a specific site as an object.
GetSitesXML	GET	sites	Returns metadata for an array of sites.
GetSites	N/A	N/A	Returns metadata for an array of sites as an object.
GetVariableInfo	GET	variable	Returns metadata for a specific variable.
GetVariableInfoObject	N/A	N/A	Returns metadata for a specific variable as an object.
GetValues	GET	values	Returns a time series.
GetValuesObject	N/A	N/A	Returns a time series as an object.

The main difference between SOAP requests and REST requests in WaterOneFlow is the separation of the operation and object in a REST request. In the SOAP requests, the Get

operation is lumped with the object the request should be applied to. In the REST requests, operations are limited to HTTP methods; POST, GET, PUT, PATCH, DELETE. The GET method applies to each of the requests, so the only variable is the object.

Another difference between the SOAP and REST requests in WaterOneFlow is the formats the data can be returned in. WaterOneFlow has defined separate requests depending on whether the response should be formatted as a string or an object. For REST requests, this information can be included in the header of the request, so there is no need to create separate requests for each format type available. Parameters for each WaterOneFlow REST request can be included in the request as query parameters. Each request has a set of associated parameters, some which are optional, and others which are required. A GET site and GET variable must include a site code and a variable code respectively. The GET sites request does not require any parameters. The GET values request must include both a site code and variable code and may optionally include a start date and an end date. All requests may include a response format parameter, but the server will return WaterML by default.

The Water Data Server uses network IDs to organize data by resource, and database IDs to identify specific aggregations. This structure is similar to how GeoServer uses workspaces and data/coverage stores to relate data to HydroShare resources and aggregations. All WaterOneFlow requests must include a network ID and database ID to connect to the data. Figure 2-16 outlines the detailed architecture of the Water Data Server, specifically how it receives requests for data, retrieves the requested data, transforms it to the desired format, and returns it to the user. A description of these steps is included below the diagram.

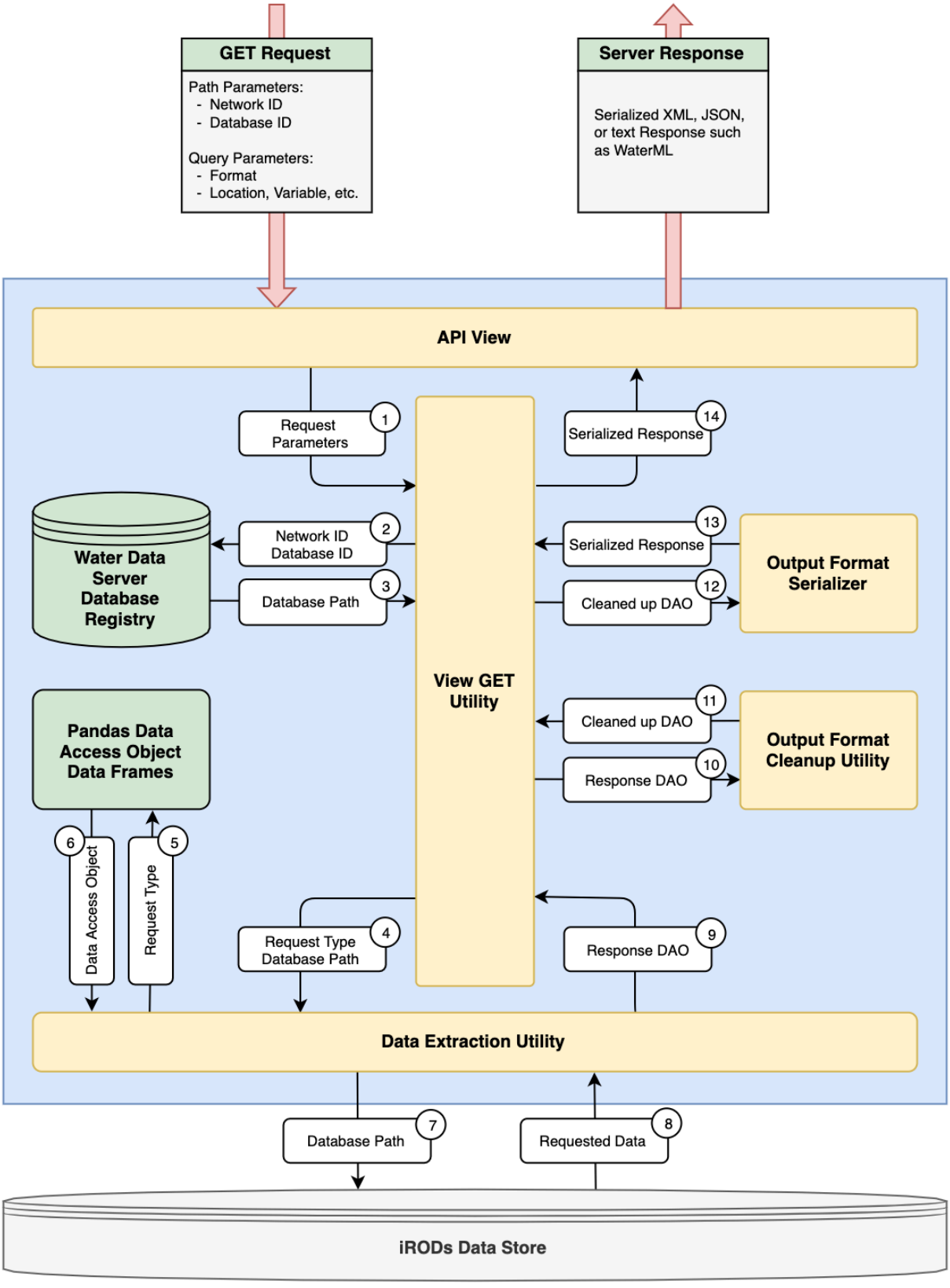

Figure 2-16: A detailed diagram of the Water Data Server data extraction process.

1. The request is received by the API view associated with the request and parameters are extracted from the request and passed along to a central GET data process.
2. The network ID and database ID are used to request the path of the ODM2 SQLite file from the Water Data Server's database registry.
3. The path of the ODM2 SQLite file is returned to the GET data process.
4. The database path and request type are passed to a data extraction process unique to each request type.
5. The data extraction process requests a data access object template in the form of a Pandas data frame.
6. The data access object template is returned to the data extraction process. The templates are built based on the request and used to load data into.
7. The data extraction utility performs SQL queries to obtain the required data from the ODM2 SQLite file.
8. Query responses are returned to the data extraction utility and loaded into the data access object template.
9. The data access object template is passed back to the GET data process.
10. The data access object is passed through a cleanup process depending on the output format requested. This ensures the data are compatible with the output format.
11. The data access object is returned to the GET data process ready to be serialized into a response.
12. The data access object is given to a serializer function that converts the data access object to a serialized response.
13. A serialized response is returned to the GET data process.

14. The serialized response is passed back to the API view to be returned to the sender of the request.

2.5.3 Water Data Server Implementation

Based on the HydroServer model shown in Figure 2-15, I have developed an implementation of this server to be compatible with HydroShare data. The Water Data Server can extract data from ODM2 SQLite files, provide WaterOneFlow data services, and return data as either WaterML, or a JSON encoding of WaterML called WaterJSON. The JSON encoding of the data is useful for web applications where parsing JSON may be easier than parsing XML. The implemented model is shown in Figure 2-17.

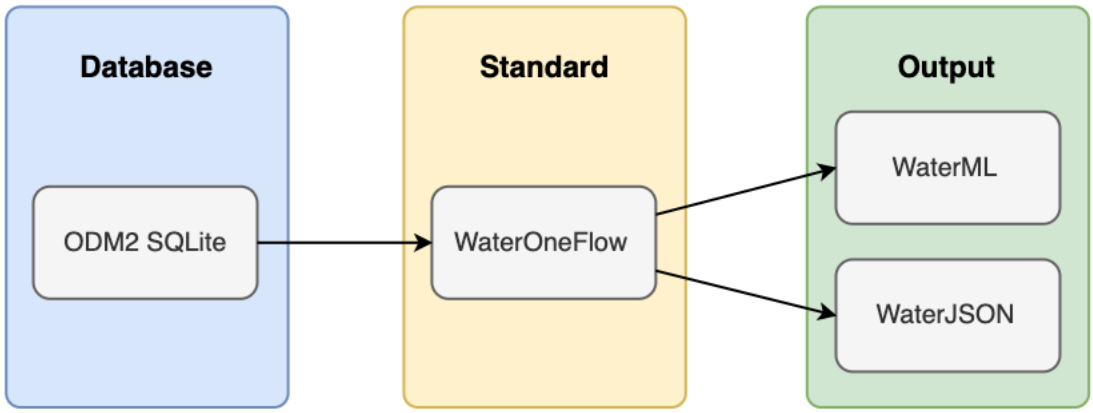

Figure 2-17: A diagram of the Water Data Server data services model.

Like the web services manager, the Water Data Server is a Django application. I have chosen to deploy it similar to the web services manager and GeoServer. The server processes are run by Gunicorn and incoming requests are handled by NGINX. These processes are managed by Supervisord, and all of these processes are contained inside a Docker container for easy

deployment. The Water Data Server can be easily deployed alongside both GeoServer and the web services manager, or in a separate location if necessary. The Water Data Server has a REST API that the web services manager can use to create or delete networks and databases based on changes made to HydroShare resources. Like GeoServer, HydroShare's iRODS data store must be mounted as a read-only volume into the Docker container so the Water Data Server can access data and respond to WaterOneFlow requests. This architecture is illustrated in Figure 2-18.

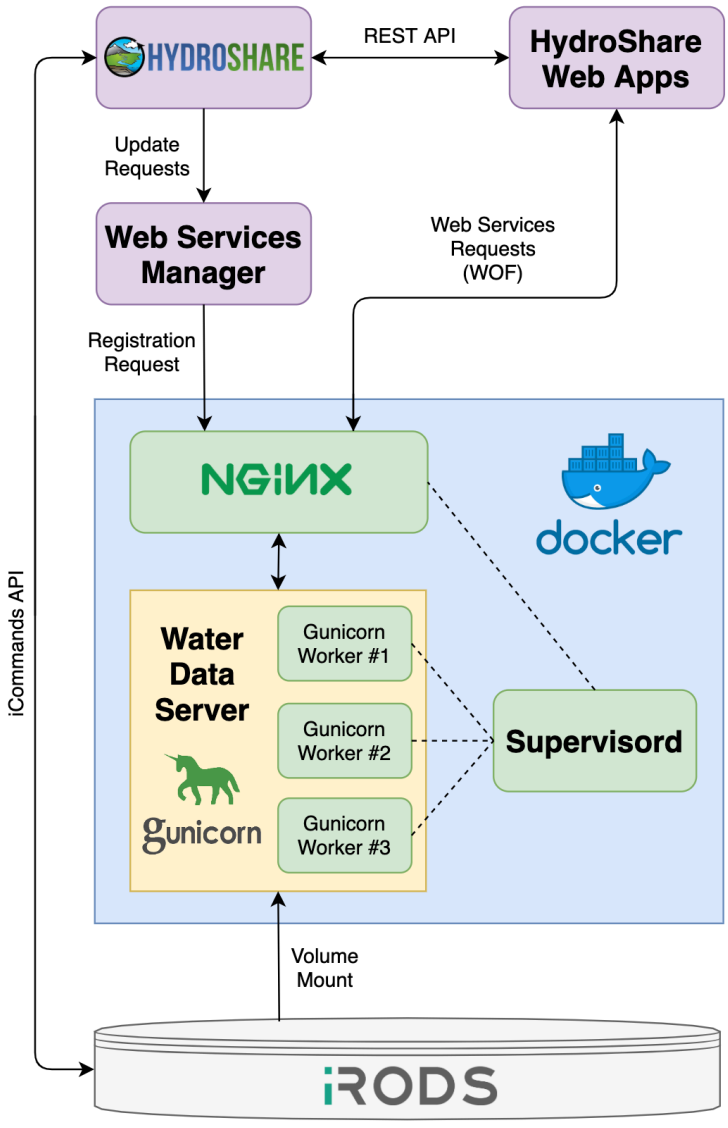

Figure 2-18: A diagram of the Water Data Server implementation architecture.

The Water Data Server's API is split into separate components depending on what it needs to do. Each of these components would represent a data services model such as WaterOneFlow, but there is also an additional component dedicated to registering and unregistering databases called "manage". This part of the API is used primarily by the web services manager application. It includes GET, POST, and DELETE requests for networks and databases, and all POST and DELETE requests require an API access token known by the web services manager. The API also includes a section dedicated to providing reference time series (REFTS) data. This type of data is used by HydroShare to reference time series data services. Figure 2-19 shows the REST API Swagger graphical interface which includes sections for the general REST API, WaterOneFlow, and REFTS.

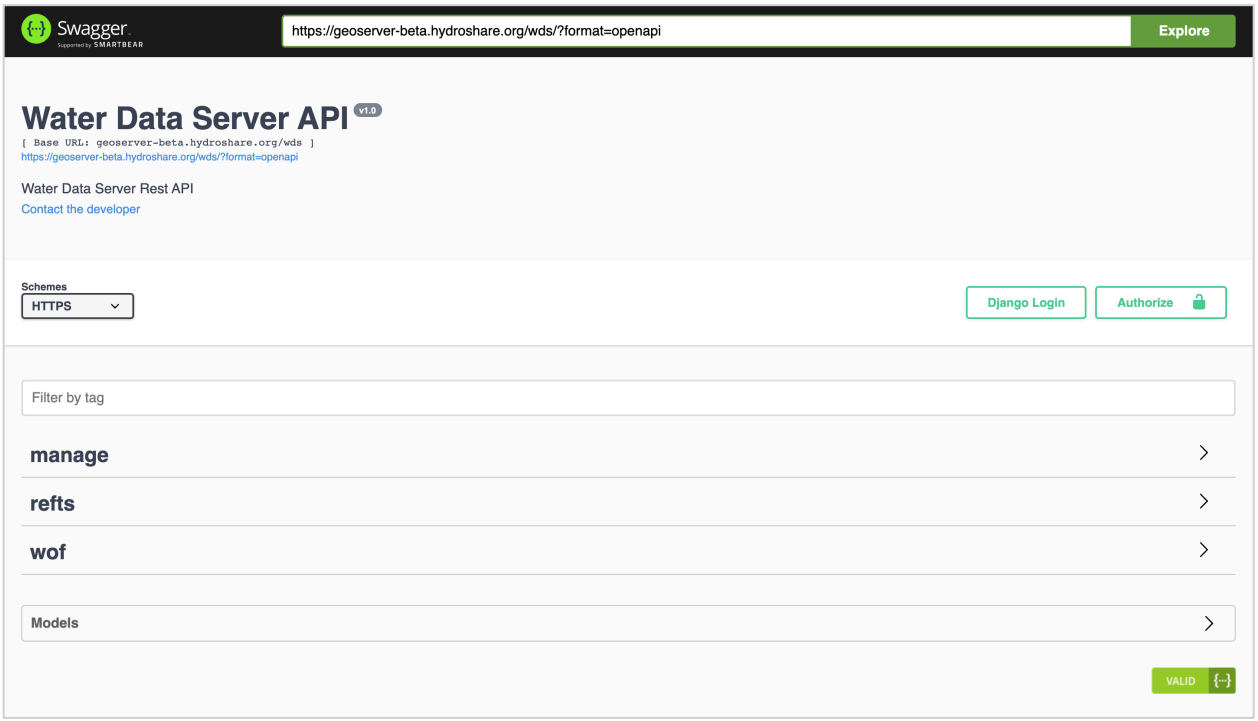

Figure 2-19: A screenshot of the Water Data Server Swagger documentation interface.

2.5.4 Water Data Server Future Work

I designed the Water Data Server out of necessity because HydroServer, HydroServer Lite, and WOFpy are not currently compatible with the HydroShare data services framework I have described. I designed the Water Data Server to be able to evolve as HydroShare does. Thanks to the data access object framework, extending the Water Data Server to include more database types such as ODM2 MySQL servers or NetCDF files, or output formats such as TimeseriesML is relatively simple. It is also possible to extend the Water Data Server to support other web service standards such as OGC Web Sensor Enablement (SWE). Although I have not developed these extensions, I have designed the Water Data Server to make the development of these types of extensions as easy as possible.

I mentioned previously that the Water Data Server includes a JSON implementation of WaterML. Other WOF applications such as WOFpy have also partially implemented some JSON form of WaterML, however there is currently no standard for WaterJSON. A WaterJSON standard should be created and implemented into the Water Data Server and other implementations of HydroServer in the future.

3 APPLICATIONS AND USE CASES

3.1 Geographic Information System Interoperability

HydroShare allows users to upload and share geospatial data, but many users use other applications to modify and use these data. There are a number of applications, both desktop and online, that are designed to work with geospatial data. Because OGC OWS is a common data standard when working with geospatial data, many of these applications support these services. With GeoServer, HydroShare data can be easily imported into these applications and used without needing to download files from HydroShare and import them into an application separately.

Esri's ArcGIS platform includes many applications that can be used to work with geospatial data. ArcGIS Online is a web application that allows users to create, modify, and share geospatial data. ArcGIS Pro and ArcMap are both desktop applications that provide access to many scripts and tools that can be used for data creation, modification, and analysis ("Documentation for ArcGIS," 2017). All of these systems can interact with each other via the ArcGIS REST API, and they can all interact with external geospatial data using OGC OWS.

ArcMap can load WMS and WCS sources from GeoServer, and WFS sources can be added by using the Data Interoperability extension. Each of these services can be connected to using a GetCapabilities document. The URLs for these documents are posted on the HydroShare Resource Landing page and can simply be copied and pasted into ArcMap's dialog for connected

to external servers. ArcMap will recognize each available layer provided by the GetCapabilities document, so a namespace parameter is included in the URL to limit the provided layers to only include the layers in the given resource. Once a connection is established, individual layers can be added to a map. Figure 3-1 below shows a Utah aquifer recharge area vector layer stored in HydroShare that has been added to an ArcMap project via WMS. WMS can be less taxing on the hardware running ArcMap because the application never needs to download the entire dataset, it only downloads images representing the data on the fly as they are needed by the application.

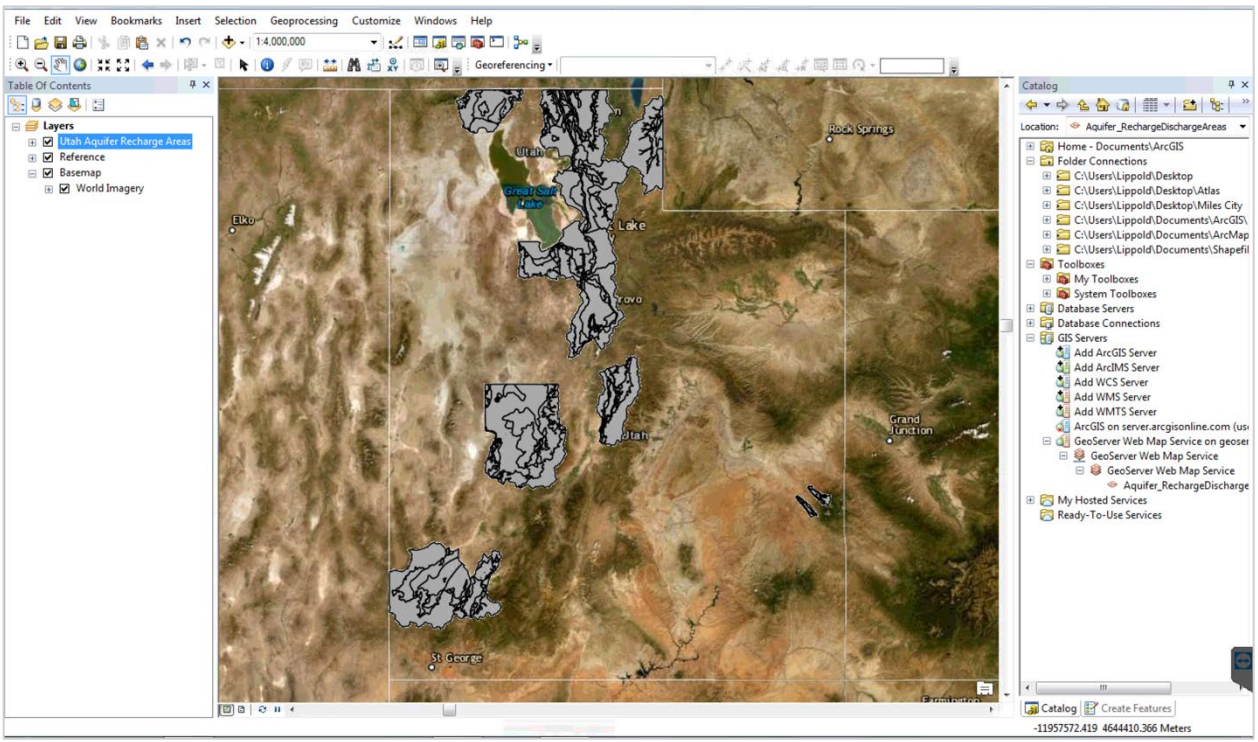

Figure 3-1: A screenshot of a polygon layer displayed in ArcMap via HydroShare WMS.

ArcGIS Pro and ArcGIS Online can load WMS, WFS, and WCS data sources. They both work similarly to ArcMap in that users can enter a GetCapabilities URL into the applications to

connect to the data. Figure 3-2 below shows a vector layer representing Utah watersheds loaded into ArcGIS Pro via WFS data services. To display a layer provided through WFS, the application must download the entire dataset from GeoServer. However, this service allows users to perform more complex analysis of the data and create custom styles to display it. Additionally, WFS can be used to download subsets of a vector layer if not all features of the layer are needed.

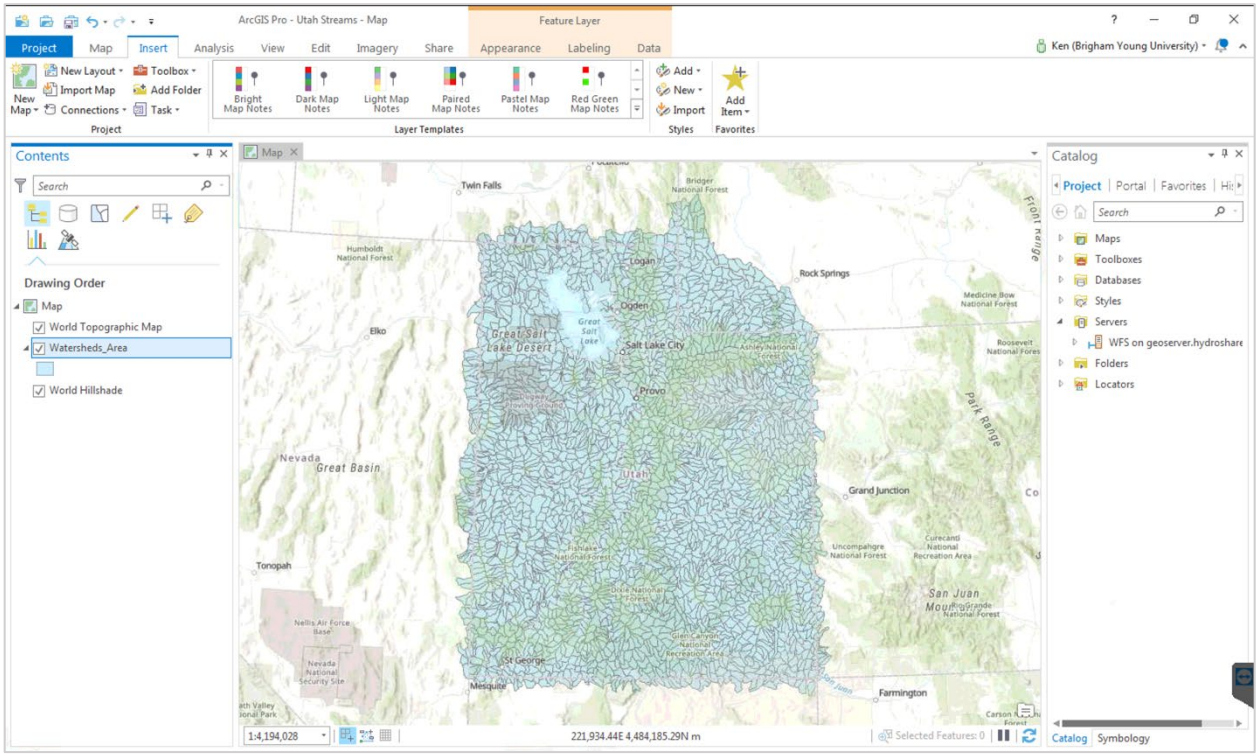

Figure 3-2: A screenshot of a polygon layer loaded into ArcGIS Pro via WFS.

QGIS is a free and open-source GIS application similar to ArcGIS platform ("QGIS Documentation," 2019). Both QGIS and ArcGIS platform have desktop and web-based components, but unlike ArcGIS Pro and ArcMap, QGIS is available on Linux and Mac operating systems, in addition to Windows. QGIS can load data from WMS, WFS, and WCS sources, so it

is compatible with HydroShare's GeoServer. Figure 3-3 below shows a DEM raster layer loaded into QGIS desktop via WCS. WCS is similar to WMS in that they both handle image formats, but WCS provides the actual data values used to produce the image, unlike WMS which just provides a pre-rendered image.

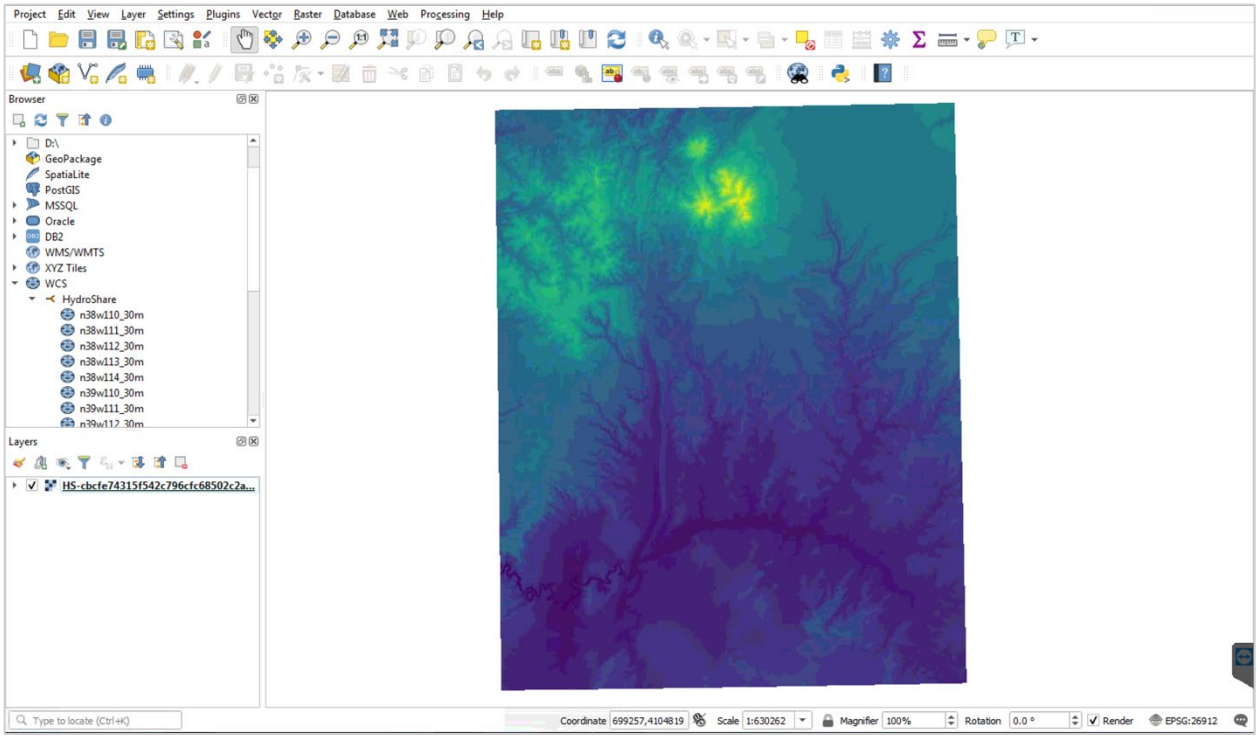

Figure 3-3: A screenshot of a raster layer loaded into QGIS via WCS.

3.2 Jupyter Notebook

3.2.1 HydroShare Jupyter Notebook Server

Jupyter Notebook is a web application that allows users to create, edit, and run code in a web browser ("Jupyter Documentation," 2019). It supports many programming languages

commonly used in hydrologic and data science, including Python, R, and Julia. HydroShare users can use the HydroShare Jupyter Notebook Server to access HydroShare resources, perform data analysis, and run models (Castronova, 2019). Jupyter Notebooks created in this environment can be saved as part of a HydroShare resource and shared with other users. Links to these notebooks can also be saved as HydroShare Web App resources, allowing users to easily launch them in their browser.

3.2.2 Time Series Data Viewer Notebook

I created a Jupyter Notebook that can use HydroShare WaterOneFlow data services to connect to HydroShare time series databases and let users view the data. This Jupyter Notebook is only a simple example to help users learn how to access and visualize data using HydroShare WaterOneFlow data services, but these notebooks can also be used to perform analysis and even run models using the data. The notebook is written in Python and uses Python's requests library to request WaterJSON data from the Water Data Server. To connect to a database, the user needs to provide a network ID and database ID. The network ID is the same as the HydroShare resource ID where the data are stored, and the database ID is the name of the time series aggregation. All of this information is available on the resource landing page and is used to build a request URL as shown in Figure 3-4.

```
In [2]: # Define database connection parameters
        network_id = "df798141ab764d769519b74dd0c0efb4"
        database_id = "utah_nwis"
        rest_url = f"https://geoserver.hydroshare.org/wds/wof/{network_id}/{database_id}"
```

Figure 3-4: A screenshot of a Jupyter Notebook Water Data Server connection.

Once a connection is established, the Jupyter Notebook will provide a list of sites available in the database. The response includes basic metadata for each site, and the site name and site code for each site are displayed in a table shown in Figure 3-5. The site code is used for other WaterOneFlow requests. At this point, the user may select a site code they are interested in and make a request for additional site metadata. The GET siteInfo request will return additional metadata for the selected site, including a series catalog. The series catalog is a list of each available time series at the selected site with a description of each variable. This information is listed in a table and includes the variable code which will be needed for the next request.

In [3]:
```
# Define data parameters
params = {"format": "waterjson"}

# Send request to Water Data Server
response = requests.get(f"{rest_url}/sites/", params=params)
content = json.loads(response.content)["sitesResponse"]

# Get data from Water Data Server response
data = (
    (s["site"]["siteInfo"]["siteName"],
     s["site"]["siteInfo"]["siteCode"])
    for s in content["sites"]
)

# Create and display table
pd.DataFrame(data, columns=["Site Name", "Site Code"])
```

Out[3]:

	Site Name	Site Code
0	FORT CREEK AT ALPINE, UTAH	10166000
1	Harbor Drive Advanced Aquatic	PR_HD_AA
2	NO FK PROVO RIV AT WILDWOOD UTAH	10160800
3	PROVO RIVER BELOW DEER CREEK DAM, UT	10159500
4	STRAWBERRY RIVER NEAR SOLDIER SPRINGS, UT	09285000

Figure 3-5: A screenshot of a Jupyter Notebook GET sites Python code and response.

From the two tables generated by the Python script, the user may now select a site code and a variable code for a time series they want to plot. Additionally, they may select a start date and end date for the time series. Given a site code, variable code, and optional temporal extent,

the notebook will request a time series. Once the response is received, it uses Python's matplotlib library to plot the time series and generate an image for the user to see similar to Figure 3-6.

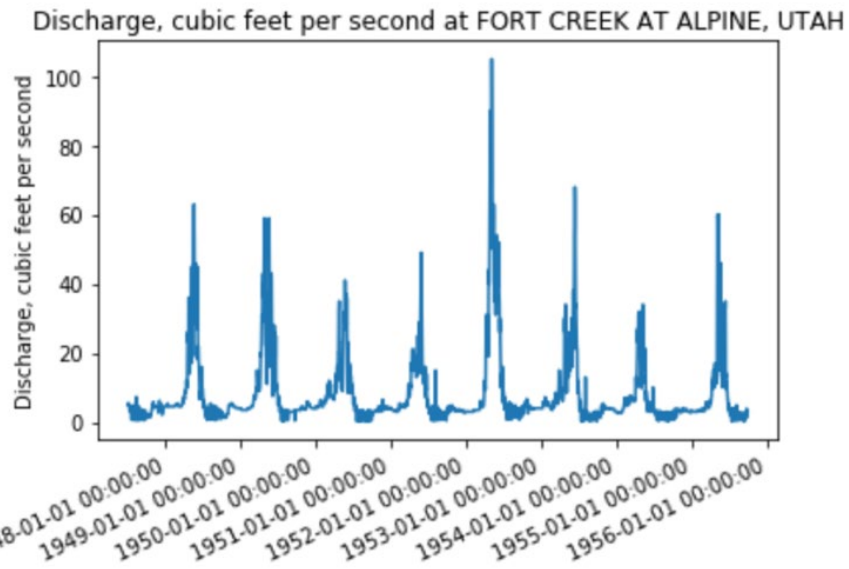

Figure 3-6: A screenshot of the time series viewer notebook plot output.

3.3 HydroShare Data Viewer App

3.3.1 Previous Work: HydroShare GIS and the Data Series Viewer

One of HydroShare's limitations is its inability to natively allow users to preview data without downloading it and using a separate desktop application to do so. This limitation illustrates a situation where HydroShare web apps can extend HydroShare's core capabilities. Several HydroShare apps have been designed to let users view and interact with data in their web browser. Two apps in particular were designed to let users preview geospatial data and time series data; HydroShare GIS, and the Data Series Viewer, respectively.

HydroShare GIS was developed at Brigham Young University and allows users to preview geographic feature and geographic raster content (Crawley et al., 2017). Users can also create resources through the app interface and export map data to HydroShare. Other features include basic symbology editing and an attribute table viewer. The app relies on the process described in Figure 2-3 to download data from HydroShare, upload it to a GeoServer not connected to HydroShare, and then load the data into the app. The code required to accomplish this adds significant overhead to the app making it difficult to scale and maintain. The app does not have the capability to synchronize HydroShare data with data uploaded to the GeoServer, so changes made to HydroShare content will not be reflected in the app. This also means that users who remove data from HydroShare, or make it private, may still find their data published through this app.

The Data Series Viewer was developed at Brigham Young University to help users preview both HydroShare and HIS time series content. The app can display time series data from a variety of different sources and also provides additional metadata for each time series. The app can be launched from either HydroShare or the CUAHSI HydroClient, depending on whether the user wants to preview HydroShare data or HIS data. Like HydroShare GIS, the Data Series Viewer must download data files from HydroShare and save them to a Tethys server before it can read data from them. Although this process is simpler with time series data than with geospatial data, it still adds overhead to the app and makes it difficult to load subsets of large datasets from HydroShare.

3.3.2 HydroShare Data Viewer Design

The HydroShare Data Viewer is based on the original HydroShare GIS app, but it uses HydroShare's geospatial data services to avoid downloading files directly from HydroShare.

Because layers are automatically published on HydroShare's GeoServer, there is no need for the app to download files and upload them to a separate GeoServer. As a result, the app requires much less overhead and is easier to scale and maintain.

In addition to using GeoServer to let users preview geospatial data, it also uses the Water Data Server to display and plot time series data. The time series component of the app will only plot one time series at a time, so it is not as robust as the Data Series Viewer. However, it does illustrate how the Water Data Server makes accessing HydroShare time series content much simpler.

The HydroShare Data Viewer uses a built-in discovery interface to let users search for public HydroShare data and add it to their map as shown in Figure 3-7. The discovery interface uses HydroShare's REST API to display a list of HydroShare resources. Users can select a resource and retrieve a list of aggregations with data layers published on GeoServer and the Water Data Server to add to the app workspace.

Once a user has added layers to the map, they can interact with them using the workspace interface shown in Figure 3-8. Each layer has several available options, depending on its type. All layers have symbology options available to change how they appear on the map. Raster layers have a list of predefined color maps that can be applied to them, and users can click on a raster to see a value at that point. Vector layers can have labels added, and if numerical data are available, users can apply attribute-based styling to them. Users can also view the attributes of vector layer features in a table and can select them from the table or by clicking them on the map interface as shown in Figure 3-9. In addition to these layer options, users can also edit the base map, create a legend for the visible layers, and add an inset map. The app relies on both WMS and WFS data services provided by GeoServer to deliver these features.

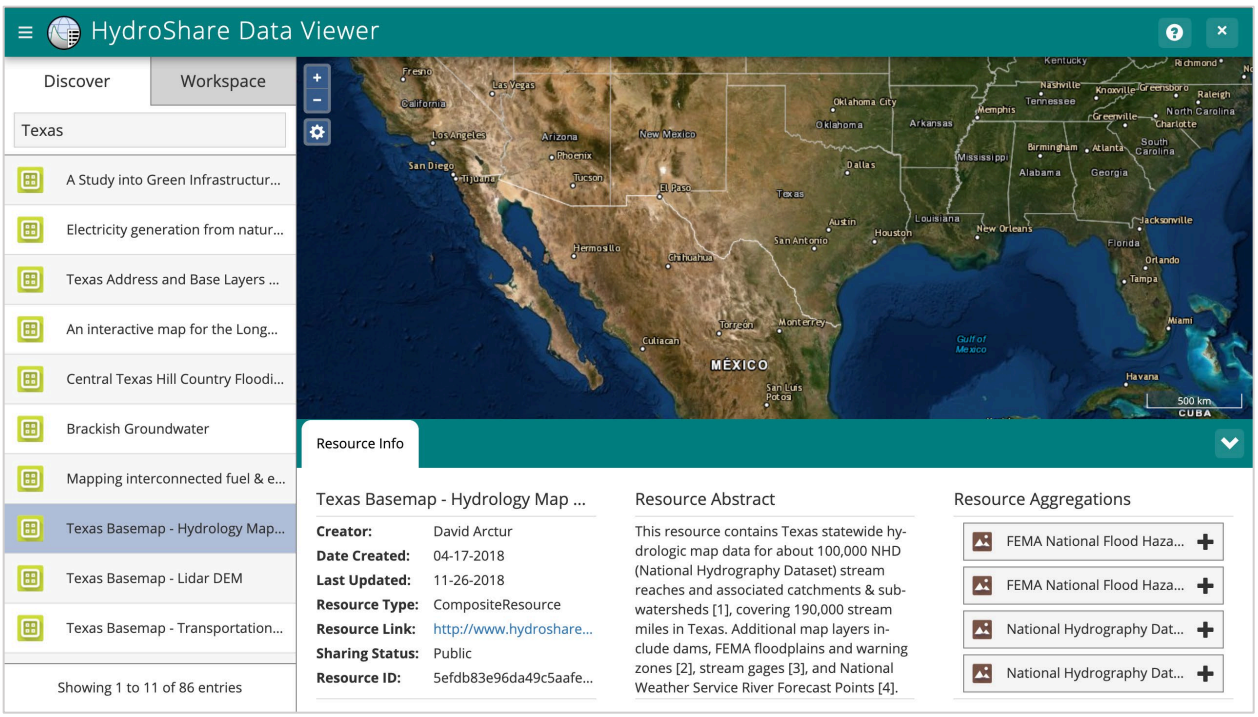

Figure 3-7: A screenshot of the HydroShare Data Viewer discovery interface.

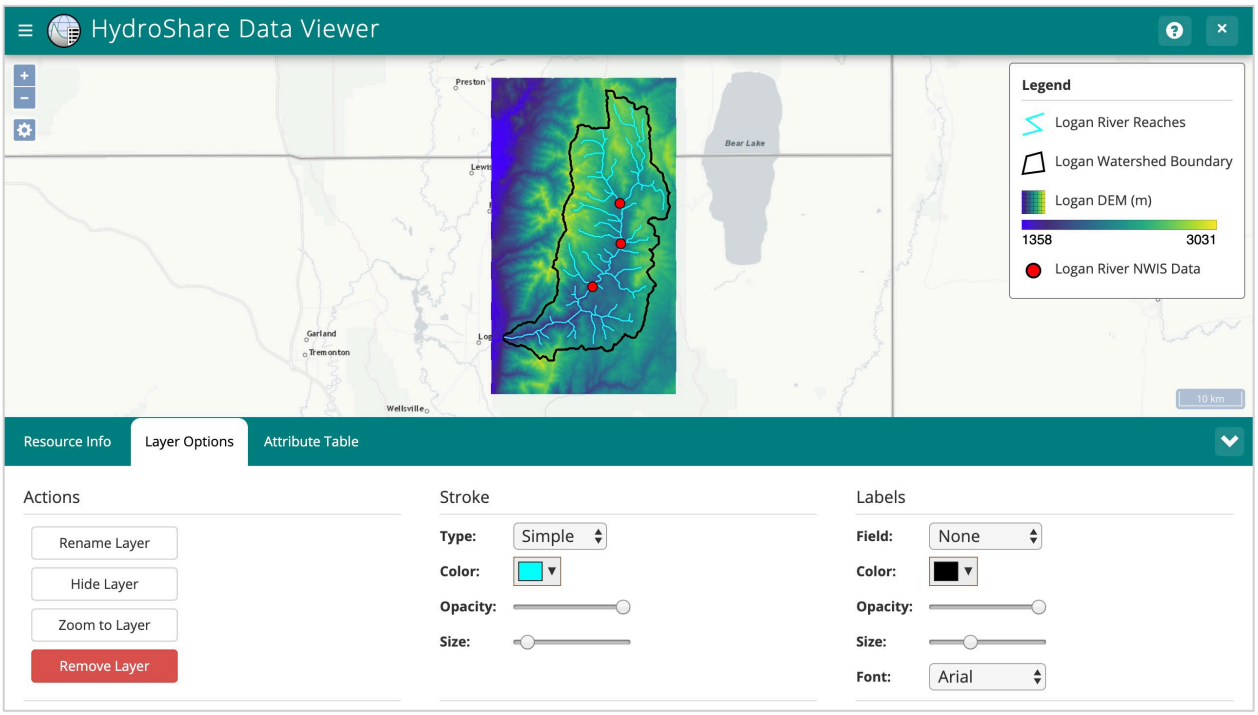

Figure 3-8: A screenshot of the HydroShare Data Viewer workspace interface.

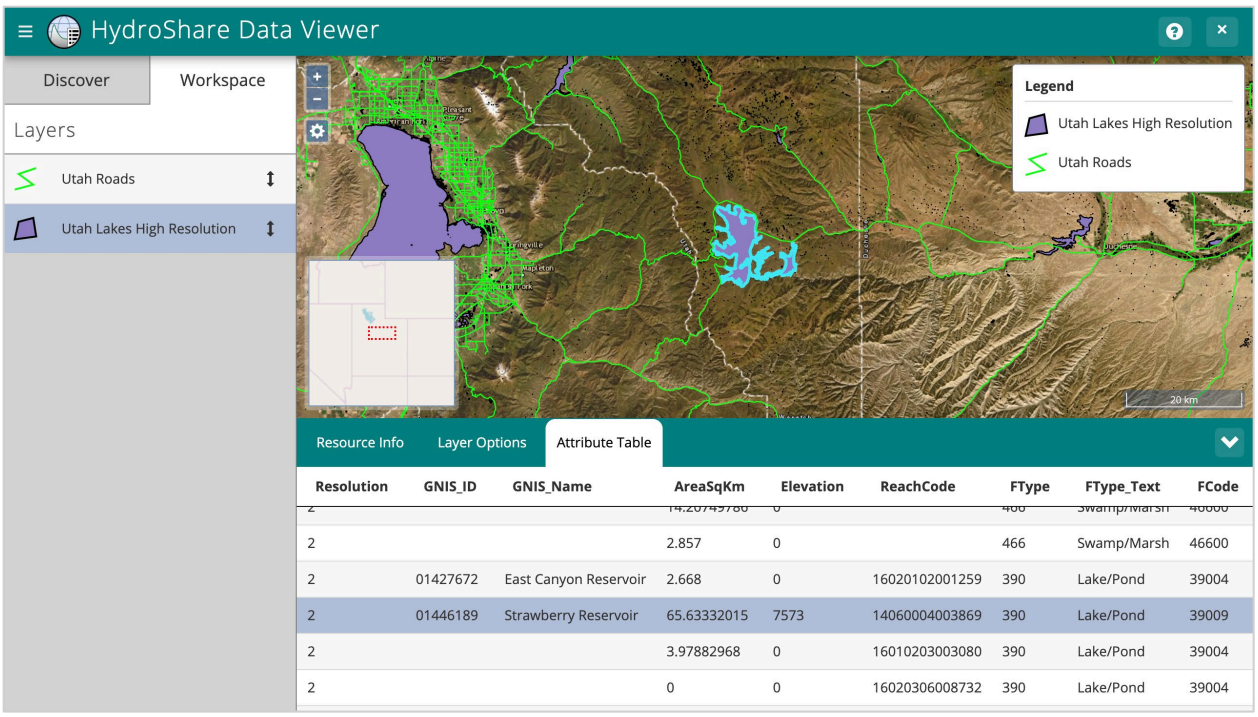

Figure 3-9: A screenshot of the HydroShare Data Viewer attribute table view.

As I mentioned, this app also supports HydroShare time series content. The Water Data Server does not provide WFS services, but it does include a REST endpoint for downloading the reference time series data in a GeoJSON format. The Water Data Server will also create a Shapefile using these data and register it on GeoServer using the pattern "TS-{Resource ID}" for its namespace. These features allow the app to map the locations of each site available in the database. The attribute table for these layers contains reference time series information which can be used to send requests to the Water Data Server. Users may select a time series they want to plot from the table or on the map, and the app will send a request to the Water Data Server and plot the WaterML response. Figure 3-10 shows a time series dataset containing NWIS gage height and discharge data collected during Hurricane Harvey where a user has clicked on a point of interest and is able to see a plot displaying gage height data measured at that point.

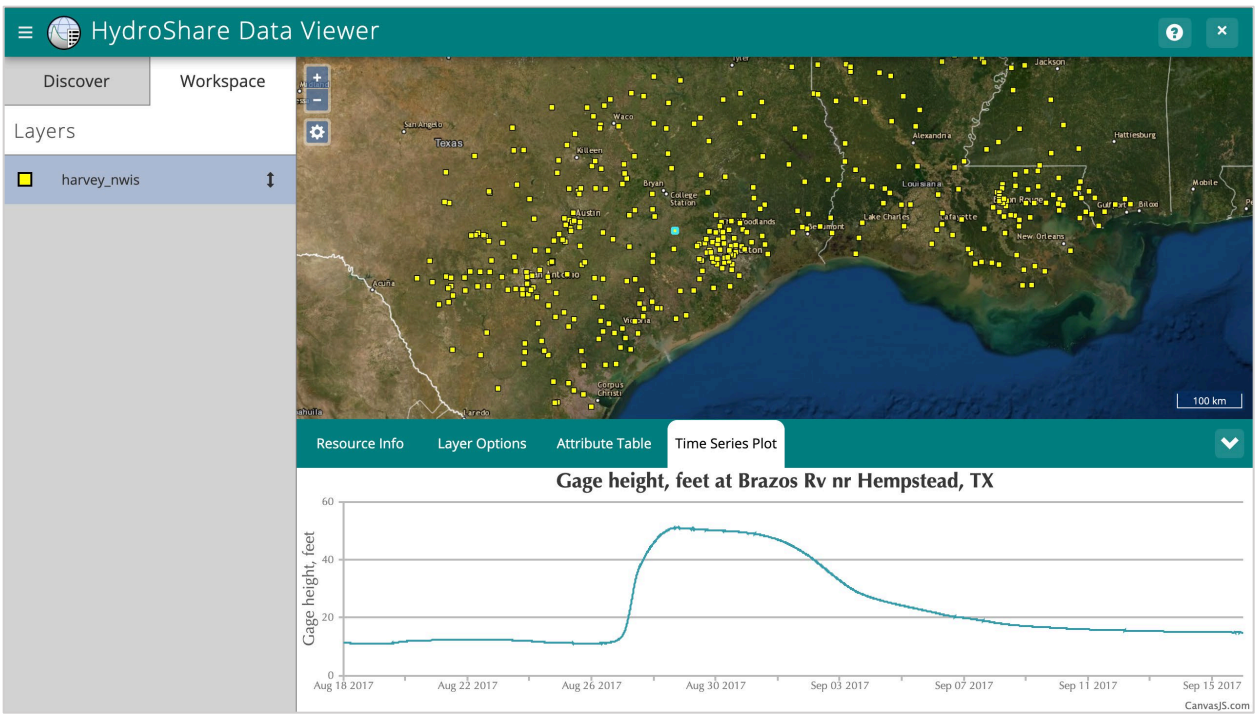

Figure 3-10: A screenshot of the HydroShare Data Viewer time series plot view.

3.4 HydroShare Time Series Manager App

Although HydroShare and CUAHSI HIS were both developed by CUAHSI, there is no built-in method allowing these systems to interact. With the Water Data Server, HydroShare has the capability to provide HIS data services, but there needs to be a way to transfer data from CUAHSI HIS to HydroShare. Initially, the HydroShare Resource Creator app was developed to transfer data from the CUAHSI HydroClient to HydroShare. With the introduction of HydroShare HIS data services, I developed the HydroShare Time Series Manager to replace the HydroShare Resource Creator. This app acts as a link between CUAHSI HIS and HydroShare, as shown in Figure 3-11.

Figure 3-11: A diagram of the HydroShare Time Series Manager app workflow.

The HydroShare Time Series Manager uses reference time series data generated by the CUAHSI HydroClient and HydroShare HIS data services to download data using WaterOneFlow. Once the data has been downloaded, a user can export the data to HydroShare as an ODM2 SQLite file. One of the limitations of the HydroShare resource creator is its inability to quickly download large amounts of time series data. Because of this limitation, the app can only create databases containing up to ten time series datasets containing a combined total of 100,000 data values. I designed the Time Series Manager app to download data concurrently instead of in sequence, so it can create time series content much faster. This app enables users to easily compile large time series databases that can be accessed via WaterOneFlow data services.

Users can search for time series data on the CUAHSI HydroClient and then export it to the Time Series Manager. It is also possible to load HydroShare time series data into the app through its interface which allows users to subset and combine data already on HydroShare. Data loaded into the Time Series Manager can be exported directly to a new HydroShare resource or added to an existing resource. The app interface is shown in Figure 3-12 with data loaded from the CUAHSI HydroClient.

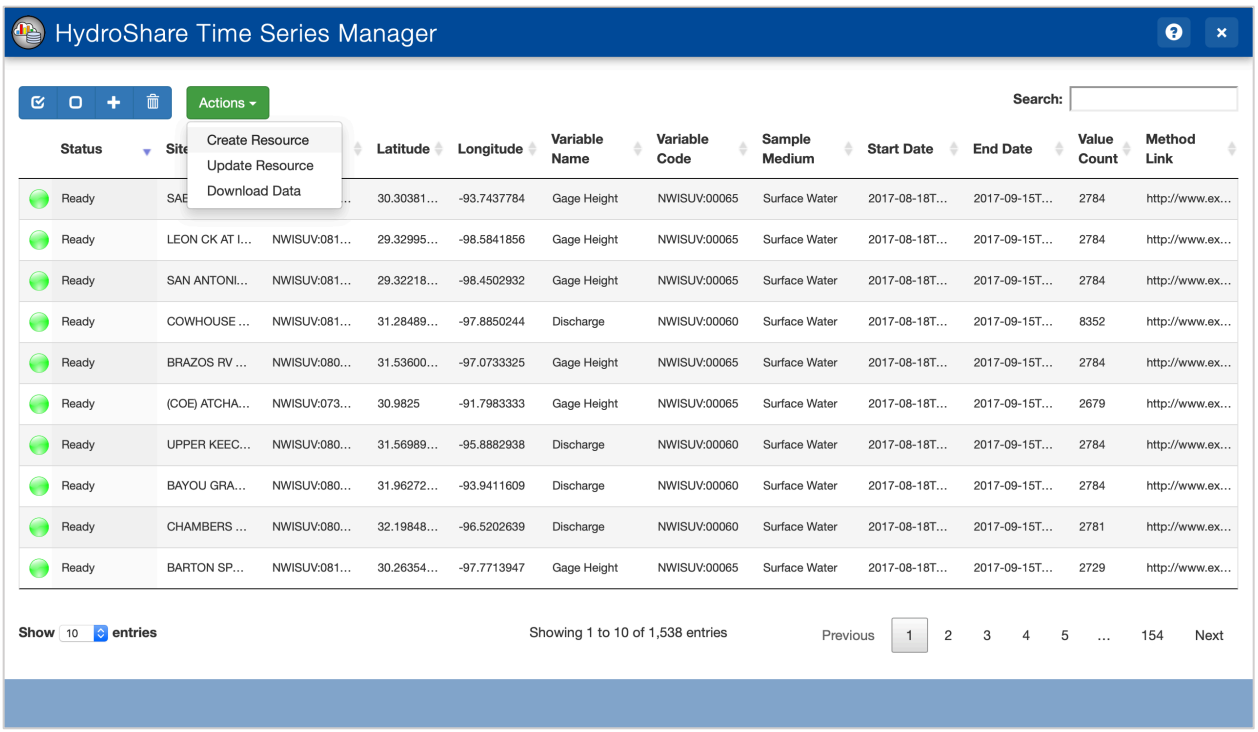

Figure 3-12: A screenshot of the HydroShare Time Series Manager interface.

4 CONCLUSION

The purpose of this research has been to improve interoperability between HydroShare and other applications through the use of OGC and CUAHSI data service standards. This research resulted in the development of a technology neutral framework that expands HydroShare's data services capabilities through integrated data servers. I have implemented this framework through the deployment of a GeoServer, which supports OGC OWS, and a Water Data Server which supports CUAHSI's WaterOneFlow data services. These services expose HydroShare data to the web using widely recognized data standards. This not only simplifies the development of HydroShare web applications, but also exposes HydroShare data to other applications not directly associated with CUAHSI or HydroShare. This increased exposure helps users include HydroShare more directly in their workflow and tie it in with other applications they use. The availability of these data services also removes many of the barriers app developers face when trying to incorporate HydroShare data into their apps.

I have demonstrated the capabilities of this framework and the resulting implementation through the development of several applications and use cases; first, a time series data viewer Jupyter Notebook which can import and plot time series data from the Water Data Server; second, a demonstration of interoperability between HydroShare GIS data services through GeoServer and Esri's ArcGIS platform; third, the HydroShare Data Viewer application which allows HydroShare users to preview HydroShare GIS and HIS data served by GeoServer and the

Water Data Server in a web browser; fourth, the HydroShare Time Series Manager application which facilitates the management of HydroShare time series data and helps further integrate CUAHSI HIS and HydroShare with the help of the Water Data Server.

I designed and implemented each of these data services systems to be able to grow and evolve alongside HydroShare, and to meet the changing needs of the growing HydroShare community. I hope that these data services will continue to expand to incorporate more types of HydroShare data as HydroShare users familiarize themselves with the framework and as more apps are developed which rely on these data services. I believe that robust data services serve a key role in HydroShare's goal of enabling scientists and engineers to more easily discover, access, and use hydrologic data.

REFERENCES

Agafonkin, V. (2019). Leaflet Documentation. Retrieved from https://leafletjs.com/reference-1.5.0.html

Ames, D. P., Horsburgh, J. S., Cao, Y., Kadlec, J., Whiteaker, T., & Valentine, D. (2012). HydroDesktop: Web services-based software for hydrologic data discovery, download, visualization, and analysis. *Environmental Modelling & Software, 37*, 146-156. https://www.sciencedirect.com/science/article/pii/S1364815212001053

Atlas. (2019). Retrieved from https://www.mapbox.com/atlas/

Bedard, D. (2015). iRODS Client Interfaces. Retrieved from https://irods.org/2015/12/update-irods-client-interfaces/

Castronova, A. M. (2019). Jupyter Python Notebook. Retrieved from https://help.hydroshare.org/apps/jupyter-python-notebook/

Chesneau, B. (2018). Gunicorn - WSGI Server. Retrieved from http://docs.gunicorn.org/en/stable/

Conner, L. G., Ames, D. P., & Gill, R. A. (2013). HydroServer Lite as an open source solution for archiving and sharing environmental data for independent university labs. *Ecological Informatics, 18*, 171-177. https://www.sciencedirect.com/science/article/pii/S1574954113000770

Crawley, S., Ames, D. P., Li, Z., & Tarboton, D. G. (2017). HydroShare GIS: Visualizing Spatial Data in the Cloud. *Open Water, 4*(1). https://scholarsarchive.byu.edu/cgi/viewcontent.cgi?article=1081&context=openwater

Django Documentation. (2019). Retrieved from https://docs.djangoproject.com/en/2.1/

Docker Documentation. (2019). Retrieved from https://docs.docker.com

Documentation for ArcGIS. (2017). Retrieved from https://doc.arcgis.com/en/

GeoServer. (2013). GeoServer Documentation. Retrieved from https://docs.geoserver.org

GeoTools Documentation. (2019). Retrieved from https://docs.geotools.org

Horsburgh, J. S., Morsy, M. M., Castronova, A. M., Goodall, J. L., Gan, T., Yi, H., et al. (2015). HydroShare: Sharing Diverse Environmental Data Types and Models as Social Objects with Application to the Hydrology Domain. *Journal of the American Water Resources Association, 52*(4), 873-889. https://onlinelibrary.wiley.com/doi/full/10.1111/1752-1688.12363

Horsburgh, J. S., Tarboton, D. G., Schreuders, K. A. T., Maidment, D. R., Zaslavsky, I., & Valentine, D. (2010). *HYDROSERVER: A PLATFORM FOR PUBLISHING SPACE-TIME HYDROLOGIC DATASETS*. Paper presented at the AWRA 2010 SPRING SPECIALTY CONFERENCE, Orlando, Florida. http://citeseerx.ist.psu.edu/viewdoc/downloadjsessionid=C86CB2E3E46DB283DA27CC40245CEE8E?doi=10.1.1.185.1225&rep=rep1&type=pdf|

How does OpenTSDB work? (2019). Retrieved from http://opentsdb.net/overview.html

InfluxDB Documentation. (2019). Retrieved from https://docs.influxdata.com/influxdb/v1.7/

Jupyter Documentation. (2019, 24 October). Retrieved from https://jupyter.org/documentation

Kalyanam, R., Campbell, R. A., Wilson, S. P., Pascal, M., Zhao, L., Hillery, E. A., & Song, C. (2016). *Integrating HUBzero and iRODS: Geospatial Data Management for Collaborative Scientific Research.* Paper presented at the iRODS User Group Meeting.

NGINX Documentation. (2019). Retrieved from https://nginx.org/en/docs/

OAuth2. (2019). OAuth 2.0. Retrieved from https://oauth.net/2/

OpenLayers Documentation. (2019). Retrieved from https://openlayers.org/en/latest/doc/

QGIS Documentation. (2019). Retrieved from https://www.qgis.org/en/docs/index.html

Rew, R., & Davis, G. (1990). NetCDF: An Interface for Scientific Data Access. *IEEE Computer Graphics and Applications, 10*(4), 76-82. https://ieeexplore.ieee.org/abstract/document/56302

Sadler, J. M., Ames, D. P., & Livingston, S. J. (2016). Extending HydroShare to enable hydrologic time series data as social media. *Journal of Hydroinformatics, 18*(2). https://iwaponline.com/jh/article/18/2/198/22/Extending-HydroShare-to-enable-hydrologic-time

Supervisor: A Process Control System. (2019). Retrieved from http://www.supervisord.org

Swain, N. R. (2015). *Tethys Platform: A Development and Hosting Platform for Water Resources Web Apps.* (Doctor of Philosophy Prospectus), Brigham Young University, Retrieved from https://scholarsarchive.byu.edu/cgi/viewcontent.cgi?article=6831&context=etd

Tarboton, D. G., Horsburgh, J. S., Maidment, D. R., Whiteaker, T., Zaslavsky, I., Piasecki, M., et al. (2009). *Development of a Community Hydrologic Information System*. Paper presented at the 18th World IMACS/MODSIM Congress, Cairns, Australia. https://www.researchgate.net/profile/Jeffery_Horsburgh/publication/228415007_Development_of_a_Community_Hydrologic_Information_System/links/5632332c08ae0530378ff7a4.pdf

Tarboton, D. G., Idasek, R., Horsburgh, J. S., Heard, J., & Ames, D. P. (2014). *HydroShare: Advancing Collaboration through Hydrologic Data and Model Sharing*. Paper presented at the 7th International Congress on Environmental Modelling and Software, San Diego, California. https://scholarsarchive.byu.edu/iemssconference/2014/Stream-A/7/

TimescaleDB Overview. (2019). Retrieved from https://docs.timescale.com/latest/introduction

W3C. (2007). SOAP Version 1.2 Part 0: Primer (Second Edition). Retrieved from https://www.w3.org/TR/2007/REC-soap12-part0-20070427/

WaterOneFlow Web Services & WaterML. (2010). Retrieved from http://his.cuahsi.org/wofws.html

Web Coverage Service. (2019). Retrieved from https://www.opengeospatial.org/standards/wcs

Web Feature Service. (2019). Retrieved from https://www.opengeospatial.org/standards/wfs

Web Map Service. (2019). Retrieved from https://www.opengeospatial.org/standards/wms

Web Processing Service. (2019). Retrieved from https://www.opengeospatial.org/standards/wps

What is ArcGIS Server. (2019). Retrieved from https://enterprise.arcgis.com/en/server/latest/get-started/windows/what-is-arcgis-for-server-.htm

Wilson, A., & Pothina, D. (2011). *Using Python, Partnerships, Standards and Web Services to provide Water Data for Texans*. Paper presented at the 10th Python in Science Conference. http://conference.scipy.org/proceedings/scipy2011/pdfs/dharhas_pothina.pdf

Youngblood, B., & Iacovella, S. (2013). *GeoServer Beginner's Guide*: Packt Publishing Ltd.

APPENDIX A. HYDROSHARE DATA SERVICES INSTALLATION

A.1 Web Services Manager Installation

Run Web Services Manager Docker instance:

```
$ sudo docker run -d -p {host_port}:8060 -v /static/his/:/static/his/ --name
  hydroshare_web_services_manager
```

Enter the Web Services Manager container:

```
$ sudo docker exec -it hydroshare_web_services_manager /bin/bash
```

Edit Web Services Manager settings:

```
$ sudo vi /home/hisapp/hydroshare_his/hydroshare_his/settings.py
```

Add your host URL to `CSRF_TRUSTED_ORIGINS`, `CSRF_COOKIE_DOMAIN`, and `ALLOWED_HOSTS`. Set `DEBUG` to False for production environments. Edit the `STATIC_URL` and `STATIC_ROOT` to point to your static files on the container and on the host. Change the `PROXY_BASE_URL` to `{host_url}/wds`.

The HIS setting should include connection information to GeoServer, HydroServer, and HydroShare REST APIs (e.g. "https://beta.hydroshare.org/hsapi" for `hydroshare_url`, "https://geoserver-beta.hydroshare.org/geoserver/rest" for `geoserver_url`, and "https://geoserver-beta.hydroshare.org/wds" for hydroserver_url). The GeoServer namespace setting is used to precede GeoServer workspace names and must start with a letter (e.g. "HS-"). The data directory settings for both GeoServer and HydroServer should match the path inside each of those containers to the mounted HydroShare resource directory. Finally, usernames and passwords for GeoServer and HydroServer should be provided to give the web services manager POST and DELETE permissions for those servers.

Save and close the file.

Exit the container:

```
$ exit
```

Restart the Web Services Manager:

```
$ sudo docker restart hydroshare_web_services_manager
```

The default username and password are admin, hydroshare. From the admin page of the Web Services Manager ({hostname}/his/), change the admin password. Create a new admin token to be given to HydroShare.

To connect this service to HydroShare, some settings must be edited in HydroShare's local_settings.py file. `HSWS_URL` should point to the Web Service Manager update endpoint (e.g. "https://geoserver.hydroshare.org/his/services/update"). `HSWS_API_TOKEN` should be set to the token value created in the previous step. `HSWS_TIMEOUT` should be set to tell HydroShare how long to wait for a response from the Web Services Manager. `HSWS_PUBLISH_URLS` should be set to True if you wish to publish data service connection URLs on HydroShare's resource landing page as extra metadata. `HSWS_ACTIVATED` should be set to True to tell HydroShare to send signals to the Web Services Manager.

A.2 GeoServer Installation

Run GeoServer Docker instance:

```
$ sudo docker run -d -p {host_port}:8181 -v host_data_vault}:/irods_vault:ro --name
  his_geoserver kjlippold/his_geoserver:latest
```

Enter the GeoServer container:

```
$ sudo docker exec -it his_geoserver /bin/bash
```

Give GeoServer IRODS Vault access:

```
$ usermod -u {host_privileged_user_id} gsapp
```

Allow anonymous REST GET access (necessary for some apps):

```
$ sudo vi /var/geoserver/data/security/rest.properties
```

Add the following lines to the file:

```
/**;GET=IS_AUTHENTICATED_ANONYMOUSLY
/**;POST,DELETE,PUT=ADMIN
```

Save and close the file.

Exit the container:

```
$ exit
```

Restart GeoServer:

```
$  sudo docker restart his_geoserver
```

The default username and password are admin, geoserver. From the admin page of GeoServer (hostname}/geoserver/web/), change the admin password. Change the contact settings. Then add a Proxy Base URL to Global Settings. (e.g. https://geoserver.hydroshare.org/geoserver). Finally, remove existing workspaces and create a new default workspace called "hydroshare".

A.3 Water Data Server Installation

Run HydroServer Docker instance:

```
$  sudo docker run -d -p 8090:8090 -v host_data_vault}:/irods_vault:ro -v
   /static/wds/:/static/wds/ --name his_hydroserver kjlippold/his_hydroserver:latest
```

Enter the HydroServer container:

```
$  sudo docker exec -it his_hydroserver /bin/bash
```

Give HydroServer IRODS Vault access:

```
$  usermod -u host_privileged_user_id} hsapp
```

Change ownership of main repository to hsapp user:

```
$  sudo chown -R hsapp /home/hsapp/hydroserver
```

Make startup file executable:

```
$  sudo chmod +x /home/hsapp/hydroserver/hydroserver.sh
```

Open application settings file:

```
$  sudo vi /home/hsapp/hydroserver/hydroserver/settings.py
```

Add your host URL to `CSRF_TRUSTED_ORIGINS`, `CSRF_COOKIE_DOMAIN`, and `ALLOWED_HOSTS`. Set `DEBUG` to False for production environments. Edit the `STATIC_URL` and `STATIC_ROOT` to point to your static files on the container and on the host. Change the `PROXY_BASE_URL` to {host_url}/wds.

Save and close the file.

Navigate to main app repository:

```
$  cd /home/hsapp/hydroserver/
```

Activate application Conda environment:

```
$  source activate hydroserver
```

Make application migrations:

```
$  python manage.py migrate
```

Exit application container:

```
$  exit
```

Restart HydroServer:

```
$  sudo docker restart his_hydroserver
```

The default username and password are admin, default. From the admin page of HydroServer ({hostname}/wds/admin/), change the admin password.

APPENDIX B. APPLICATION USER DOCUMENTATION

B.1 HydroShare Data Viewer User Instructions

This app is designed to run on Tethys Platform and helps support CUAHSI's HydroShare project. Its purpose is to allow HydroShare users to quickly preview hydrologic geospatial and time series content stored in HydroShare resources. You can open this app from the HydroShare apps portal, or from the resource landing page.

If you open the app from the apps portal, you will be brought to the resource discovery page of the app. From here you can search for HydroShare resources. When you click on a resource, the map will zoom to the spatial coverage of the resource and display a summary of the resource's metadata. This summary includes a list of available aggregations you can add to the app's workspace.

Once you add an aggregation to the workspace, or if you opened the app from the resource landing page, you can select a layer in the workspace view. Once you select a layer, will be presented with several options depending on the layer type. All layers have resource metadata and layer options. Geographic feature and time series layers will have an attribute table, and time series layers will also have a plot view. Additionally, once you've selected a layer, you can interact with it on the map. You can click on a raster layer to display the value at that point, and you can click on a geographic feature layer to select a feature. You can also select features from

the attribute table. When you select a time series feature, the plot view will become available and the app will retrieve and display a time series for the selected feature.

From the layer options view, you can rename a layer for your session, hide a layer, zoom to a layer, and remove a layer from your session. Several symbology options are also available depending on the layer type. If the layer is a geographic feature and contains numerical data, you can switch the fill/stroke type from simple to gradient. This will apply a color gradient to the layer's features based on the numerical field you choose.

Finally, there are several map options you can toggle. You can choose from a list of base maps including satellite, street, grey, and none. You may also toggle a reference layer, terrain layer, legend and inset map. The legend will only include visible layers and more detailed information for gradient styling.

B.2 HydroShare Time Series Manager User Instructions

This app is designed to run on Tethys Platform and helps support CUAHSI's HydroShare project. Its purpose is to facilitate the transfer of hydrologic time series data from CUAHSI's HydroClient application to HydroShare. The app can also be used to combine, subset, and edit existing time series content within HydroShare. You can open this app from the HydroShare apps portal, the resource landing page, or the CUAHSI HydroClient.

If you open this app from a resource landing page or the CUAHSI HydroClient, it will load any reference time series data it can find into the table. If you open the app from the apps portal, the table will be empty. The toolbar above the table includes several tools for interacting with the table data. You can use the search bar to filter the table data and the select all button to select filtered rows. You can also deselect all rows and remove all selected rows. The add data button

will let you add data from a HydroShare resource, Water Data Server endpoint, or reference time series JSON file.

Once you've selected all rows of interest, you may use the actions dropdown to either create a new HydroShare resource, update an existing resource, or download your data locally. To create or edit a resource, you will be prompted to log in to HydroShare. If you choose to create a resource, the app will automatically build a title, abstract, and keywords which you can edit. You may also choose to make the resource public or private. If you choose to download your data locally, you may choose to download it as WaterML, CSV, ODM2, or reference time series JSON.