Cloud-enabling a Collaborative Research Platform: The GABBs Story

Rajesh Kalyanam
Purdue University
West Lafayette, Indiana 47907
rkalyana@purdue.edu

Rob Campbell
Purdue University
West Lafayette, Indiana 47907
rcampbel@purdue.edu

Derrick Kearney
Purdue University
West Lafayette, Indiana 47907
dsk@purdue.edu

Leif Delgass
Purdue University
West Lafayette, Indiana 47907
ldelgass@purdue.edu

Larry Biehl
Purdue University
West Lafayette, Indiana 47907
biehl@purdue.edu

Lan Zhao
Purdue University
West Lafayette, Indiana 47907
lanzhao@purdue.edu

Carolyn Ellis
Purdue University
West Lafayette, Indiana 47907
carolynellis@purdue.edu

Carol Song
Purdue University
West Lafayette, Indiana 47907
cxsong@purdue.edu

ABSTRACT

Modern cyberinfrastructures typically involve tightly integrated compute, storage and web application resources. They also form the basis of science gateways, which add their own science-specific processing or visualization capabilities. While some science gateways are intended as the central resource provider for a certain scientific community, others provide generic capabilities that are intended for further customization at each installation site. However, replicating their setup is a non-trivial task often involving specific operating system, software package and configuration choices while also requiring allocation of the actual physical computing resources. Cloud computing provides an attractive alternative, simplifying resource provision and enabling reliable replication. We describe our ongoing efforts to cloud-enable a geospatial science gateway hosting general-purpose software building blocks termed GABBs, that provide geospatial data management, analysis, visualization and processing capabilities. We describe the various compute and storage resources and software underlying these building blocks and our automation of the deployment, software installation and configuration of this science gateway on the Amazon Web Services (AWS) cloud platform. Some of the challenges that were encountered and resolved during this cloud-enabling process are also described.

PEARC17, July 09-13, 2017, New Orleans, LA, USA

ACM ISBN 978-1-4503-5272-7/17/07…$15.00
https://doi.org/10.1145/3093338.3093353

CCS CONCEPTS

• **Information systems → Open source software**; *Computing platforms*; • **Software and its engineering → Cloud computing**; *Organizing principles for web applications*; *System administration*; Maintaining software;

KEYWORDS

Cyberinfrastructure, science gateway, cloud computing

ACM Reference format:
Rajesh Kalyanam, Rob Campbell, Derrick Kearney, Leif Delgass, Larry Biehl, Lan Zhao, Carolyn Ellis, and Carol Song. 2017. Cloud-enabling a Collaborative Research Platform: The GABBs Story. In *Proceedings of PEARC17, New Orleans, LA, USA, July 09-13, 2017,* 8 pages.
https://doi.org/10.1145/3093338.3093353

1 INTRODUCTION

Scientific research is now an increasingly collaborative effort. Researchers on a project are often spread globally, contributing data and tools to foster collective progress. Scientific tools are also increasingly publicized to enable widespread and quicker adoption and feedback. This public availability of data and tools aids classroom instruction, keeping future generations of researchers abreast of recent developments in the field. Science gateways offer a natural solution for such collaborative efforts. They are in effect a web platform offering data and scientific tool management, access to high performance computing resources, and user community and social networking support.

While some science gateways function as a central resource provider for a scientific discipline, others are intended as general-purpose frameworks that can be customized and installed at various locations supporting either a specific user community or research initiative. The HUBzero cyberinfrastructure framework [8] is one such customizable framework for collaborative scientific research. Out-of-the-box, HUBzero provides user and community management, data management, digital object identifier (DOI) generation

for data publication and browser-based Linux desktop environments for tool development. HUBzero has been used in a diverse range of disciplines from nanotechnology, material science, and biomedical engineering to earthquake engineering. However, one can notice specific similarities in these usage patterns. Several science gateways that cater to interdisciplinary communities studying drought conditions, effects of climate change on crops, and the economics of global food production are based on HUBzero. A common theme in these gateways is a need for value-added services specific to geospatial data; the ability to preview, overlay, annotate, explore and transform such data. The GABBs (Geospatial Data Analysis Building Blocks) [1] project was developed to add these services, adding a layer of geospatial-data specific capabilities to HUBzero, while retaining the generality that allows for site-specific customization. GABBs is funded by the NSF DIBBs program that encourages the development of data-centric cyberinfrastructure that provides data services and capabilities fostering interdisciplinary research.

The addition of these capabilities calls for additional storage and visualization resources and seamless integration into the HUBzero framework. However, these requirements raise the barrier to entry for researchers attempting to set up their own GABBs-enabled gateway. While potential users of this framework can explore its capabilities on a central pre-configured demonstration platform, they will most certainly lack the administrative privileges to modify or add content without relying on dedicated support. It then becomes important to simplify the set up of such gateways, striving for as much automation as possible. Virtual machines (VMs) are the obvious choice for distributing such pre-configured machine images. However, due to the several interconnected resources required, several separate VMs need to be installed and configured to set up these interconnections. More importantly, the storage and visualization servers require non-trivial resources to function optimally. This precludes installation on most personal computers.

Cloud-computing services like Amazon Web Services, Google Cloud Platform and Microsoft's Azure are ideal for such setups due to the availability of a wide variety of machine images for various operating systems, as well as choice of the hardware including CPU, memory and graphics processors. In addition, they often provide deployment management services that enable the automation of creation, configuration and deployment of resources. In this paper, we describe our experience using AWS CloudFormation to configure and deploy the interconnected resources that comprise a typical GABBs-enabled HUBzero gateway. We begin with a description of the GABBs project, the various external resources that are an integral part of GABBs and our reasons for separating them from the central HUBzero installation. Next, we describe how CloudFormation can be employed to automate the software installation and configuration of these individual resources as well as some issues we encountered. Since this is still work in progress, we end with a description of our pending tasks and our vision of the way forward. While the broader GABBs project comprises of several inter-related development efforts involving multiple developers and domain scientists, we focus on a small part of that effort here; namely the packaging and dissemination. However, to establish appropriate context, we start with an overview of the GABBs project and its components.

2 THE GABBS PROJECT

2.1 HUBzero Cyberinfrastructure Framework

Before describing the various resources implementing GABBs capabilities, it is first necessary to describe the HUBzero framework and its extension to include GABBs capabilities.

HUBzero grew out of efforts to generalize the cyberinfrastructure underlying nanoHUB [7], a platform for research and education in nanotechnology. NanoHUB allows users to build simulation tools, share data and results, generate digital object identifiers (DOI) for publication and develop course materials for instruction in nanotechnology. A core component of nanoHUB is the content management system (CMS) that allows all of these pieces to be integrated into a single web platform with a uniform look-and-feel. Another component of nanoHUB and perhaps the defining feature of HUBzero is support for simulation tool creation and use. Any scientific code that can be executed in a Linux environment can be packaged into a tool that can be run on-demand in a containerized environment accessible through the user's web browser. In effect, these tool containers provide a complete Linux desktop environment accessible on a web browser. In addition to the webserver serving up the CMS, one or more execution hosts can be used in conjunction with middleware code that configures, manages and serves these tool containers from these execution hosts. HUBzero resulted from the abstraction of both the CMS and middleware used in nanoHUB into a generic cyberinfrastructure framework. Each instance of HUBzero termed a hub can be configured with its own set of templates, and hosts its own set of tools, resources and web content.

2.2 GABBs Capabilities

The GABBs project was envisioned from observations of common usage patterns among various hubs. For instance, users of the WaterHUB [9], Geoshare [4] and U2U [3] hubs studying various aspects of droughts, agricultural economics and climate science all required a geospatial data management system with similar capabilities. The desired system would support quick data previews, the construction of intuitive overlays and general-purpose tools for manipulating geospatial data.

GABBs accomplishes these goals via the following contributions:

(1) A data management system, iData [6] is integrated into the primary hub collaboration space: hub projects. iData differs from the existing data management capabilities of HUBzero in its special handling of geospatial data. For instance, metadata is automatically extracted from various raster and vector file formats and indexed for subsequent search.
(2) Geospatial files managed by iData can also be previewed directly on the web browser with support for overlays of multiple files.
(3) General-purpose hub tools supporting various operations on geospatial and spreadsheet data can be automatically launched on individual files from the iData file management interface.

2.3 GABBs Components

Underlying iData is an external iRODS [10] server that manages the physical files uploaded to iData. iRODS is a distributed file management system that presents a uniform Unix filesystem view of its files while abstracting away the actual details of where the files are stored. This allows storage resources to be added or expanded without affecting the user's view of their data. More importantly, iRODS supports the execution of code in response to various events during a file's lifecycle. For instance, we can attach a function that automatically processes and extracts metadata from structured geospatial files as soon as they are uploaded. In addition, locating such processing on the data storage system frees up the hub webserver from devoting valuable resources to this task. There is a myriad of ways in which files can be uploaded to iData and not just through the hub's web interface. Locating the processing of these files at the storage site makes it agnostic to the upload mechanism used. While pertinent details of this integration of iRODS and HUBzero can be found later in this paper, we refer interested readers to [5] for a complete account.

Geospatial file preview is a non-trivial task that involves processing the raw physical file before registering it to a map server that serves map layers to be displayed on a web browser using Javascript libraries. GABBs employs GeoServer to serve such map layers. However, additional pre-processing is often required to convert raw files to the specific formats that GeoServer supports. Co-locating GeoServer with the iRODS server reduces the required file transports, improving performance.

While web-based previews offer a quick overview of the data in a geospatial file, researchers often need to perform non-trivial transformations on the original file. For instance, they may need to reclassify, reproject or run various image processing algorithms on the raw data in the file. In addition, geospatial data from sources such as satellite feeds is often multidimensional, containing diverse data in separate channels. For instance, a single file could contain vegetation cover, CO2 indices and cloud cover data. Researchers are often interested in only a subset of the data. Such extraction as well as all the processing mentioned above can be performed using the MultiSpec [2] tool that can handle such multispectral (multi-channel) or even hyperspectral data. Due to its general-purpose nature, we intend MultiSpec to supplement the GABBs data management framework, providing a geospatial data visualization, processing and transformation tool. Sometimes, researchers may just seek intuitive data displays combining data from various sources. For instance, a researcher may want to display a map of the state of Indiana with markers denoting the various water assessment stations along the Wabash River. Each such station may have corresponding spreadsheet data containing water quality information, contaminant and mineral densities over a certain period. An intuitive display would combine these diverse sources of information, allowing users to quickly study plots of the spreadsheet data to decipher trends over time and location. The GeoBuilder hub tool was developed as a component of GABBs that can be used to build such displays combining any geospatial and spreadsheet data files. However, in addition to flat 2-D maps, GeoBuilder can also display 3-D geospatial data overlaid on the globe. This is especially useful when visualizing terrain data that loses perspective when viewed in 2-D. To support such visualization and optimize the loading of large datasets, a separate visualization server running the osgEarth geospatial and terrain rendering software is used to build GeoBuilder displays. Due to the hardware (graphics card) and memory requirements of such a visualization server, it is separated out from the hub webserver with the middleware supporting load-balancing across one or more such visualization servers.

2.4 GABBs System Design

In addition to the default file storage provided by projects in HUBzero, efforts are underway to integrate various external storage providers such as Dropbox, Google Drive and Globus using PHP's Flysystem abstraction. Rather than implement an iRODS Flysystem adapter from scratch for our integration, it was decided to use the pre-existing local Flysystem adapter to access iRODS files mounted to the hub webserver's local file system. The iRODS FUSE client can be used to mount iRODS file-spaces to appear as a local directory on any remote machine. Reads and writes to iRODS are then as simple as reading and writing to a local directory on the webserver. However, this requires the iRODS FUSE client to be set up and configured to allow access to the hub's web user, which in turn requires knowledge of the iRODS server settings such as hostname, port, username and password. While some performance issues were encountered during file listing operations on the iRODS FUSE mount, caching of filestat information was incorporated to overcome this issue resulting in a ten-fold improvement in time taken to list directory contents.

As mentioned previously, one of the GABBs contributions is that metadata from geospatial files is automatically extracted on upload. Geospatial files are typically highly structured and contain useful metadata such as geographic bounds and projection information that aids in their visualization. Moreover, results of simulations or curated datasets typically contain additional metadata describing the dataset. Rather than have users recreate such metadata, the expectation is that it would be extracted automatically when present. GABBs uses iRODS microservices to implement such processing in response to file uploads. More specifically, iRODS rules can be created to execute microservices in response to various file events such as upload, rename and delete. iRODS rules can also be executed on-demand via iRODS client APIs. This feature is exploited to implement on-demand file previews via a microservice that performs the necessary processing to register a map layer on GeoServer. These additional rules and microservices form part of the GABBs-specific installation of our iRODS server.

While desktop versions of the MultiSpec tool for both the Windows and Mac platforms have been available since the early 1990s, it had to be extended to run on Linux using wxWidgets before packaging and deploying it as a hub tool. GeoBuilder was developed using the HUBzero Rappture tool development kit. While space constraints preclude a complete description of Rappture here, suffice it to say that Rappture simplifies the design of a graphical user interface (GUI) for any simulation tool. As a part of GABBs efforts, new map input and output GUI elements were added to Rappture and leveraged in building the GeoBuilder tool. Finally, the render (visualization) server that composes and serves map layers to GeoBuilder was based on a combination of osgEarth and

Rappture server elements. Figure 1 illustrates the various GABBs components and where they fit into the broader HUBzero paradigm.

3 INSTALLING GABBS

We next describe a canonical installation of GABBs on a HUBzero hub. This illustrates both the various software installation and configurations required, as well as the confines of the HUBzero paradigm that we need to work within. Similar steps need to be followed when replicating such deployment in the cloud.

3.1 iRODS and GeoServer

The simplest installation of iRODS involves a single iCAT-enabled resource server. iCAT is the iRODS catalog that manages the physical files and their metadata. iRODS provides several different installation packages for various OSes and catalog database choices. The only non-trivial aspect of the installation is an interactive configuration process that provides the user with the option to choose the Linux owner, administrator name, password, physical file location, etc. iRODS also provides templates for microservice creation and packaging using the EPM packaging software. EPM automatically detects the operating system where the build occurs and generates an appropriate software package; for instance .deb on a Debian machine and .rpm on RedHat or CentOS. Installation of the microservice on an iRODS server is then just a simple matter of installing one of these packages. If any new iRODS rules are required, they can be added to the standard iRODS rule file on the iRODS server.

GeoServer is written in Java and can be configured to either run on its internal Jetty webserver or an external Tomcat webserver. Various GeoServer binaries are available for download and installation is just a matter of extracting the binary and placing it somewhere on the target server machine. If an appropriate version of the Java runtime environment is present on this machine, GeoServer can be started using the provided startup scripts after configuring environment variables to point to the installation directory and Java. The final step is to configure the iRODS server to point it to the local GeoServer. This step is necessary to allow the file preview microservice to process files and register them as map layers on the local GeoServer. iRODS microservices can access any server configuration variables, making this an ideal approach for handling such additional configuration.

Metadata extracted via the microservices is also indexed into the Apache Solr server running on the hub webserver. This requires another server configuration setting pointing to the Solr server's endpoint.

3.2 GABBs CMS and iRODS FUSE mount

The various GABBs CMS contributions such as the iRODS storage provider for hub projects and the code responsible for metadata management and file previews in iData are designed as HUBzero plugins. HUBzero plugins allow enhancements or overrides of the generic HUBzero CMS without having to tinker with the generic CMS code base. This also allows for seamless integration into the hub projects interface. Plugin installation is as simple as placing the plugin files in the appropriate directories and running a migration script that makes the necessary database updates. Plugins allow for site-specific configuration variables, which can be set in these same migration scripts.

Creation of the iRODS FUSE mount requires the installation of the iRODS client package on the hub webserver, the creation of a directory owned by the web user, addition of the user to the fuse group and the actual creation of the mount using the irodsFs FUSE mount creation command.

3.3 MultiSpec and GeoBuilder Tools

Hub tools normally go through a tool pipeline process consisting of initial development, build, testing and finally publication when the tool is made available to all hub users. Of course, subsequent installations of these tools on new hubs don't need to go through the pipeline process. Instead, we can skip ahead to the last step in the process; i.e. publication. On a new hub, tool publishing scripts can start with a tarball of an installed tool, extract, copy to the installation target, and, make the necessary database entries adding this tool to the set of available hub resources.

Tool containers are essentially generic Linux virtual machines based on a standard template. When a tool is installed on a hub, it is not directly installed into a tool container. Instead, tools are installed on execution hosts at a standard path and the middleware mounts this path onto tool containers when they start up. A similar approach is taken with tool software dependencies that aren't provided by default in the base operating system in the tool containers. Such dependencies are installed at a standard path on the execution host that is mounted onto tool containers. Thus, the tool installation process involves installing the actual tool binaries and the necessary dependencies at separate locations.

3.4 Visualization Server

Going together with the GeoBuilder tool is the visualization (render) server that composes and streams map views to the tool. The render server software assumes certain hardware specifications, but beyond that is a rather straightforward installation of osgEarth, Rappture and other software dependencies.

4 PACKAGING GABBS

4.1 Why Amazon Web Services?

All the GABBs components described so far were developed and deployed on a production science gateway, MyGeoHub.org [11]. Individual users can use MyGeoHub to explore various GABBs capabilities and research groups can host their data and tools there; however, it isn't intended as a central science gateway for geospatial capabilities. GABBs is designed with the DIBBs program goals in mind and provides general-purpose building blocks that can be added to any hub and subsequently configured and extended as the user sees fit. In addition, data services are provided to foster interoperability with non-HUBzero cyberinfrastructures as well. To simply the GABBs-enabling of any HUBzero hub, it is vital that the installation process be as streamlined and straightforward as possible. This typically involves developing generic-enough software packages that contain configuration scripts for site-specific installations.

Amazon Web Services (AWS) and cloud computing platforms in general serve some important roles in this paradigm:

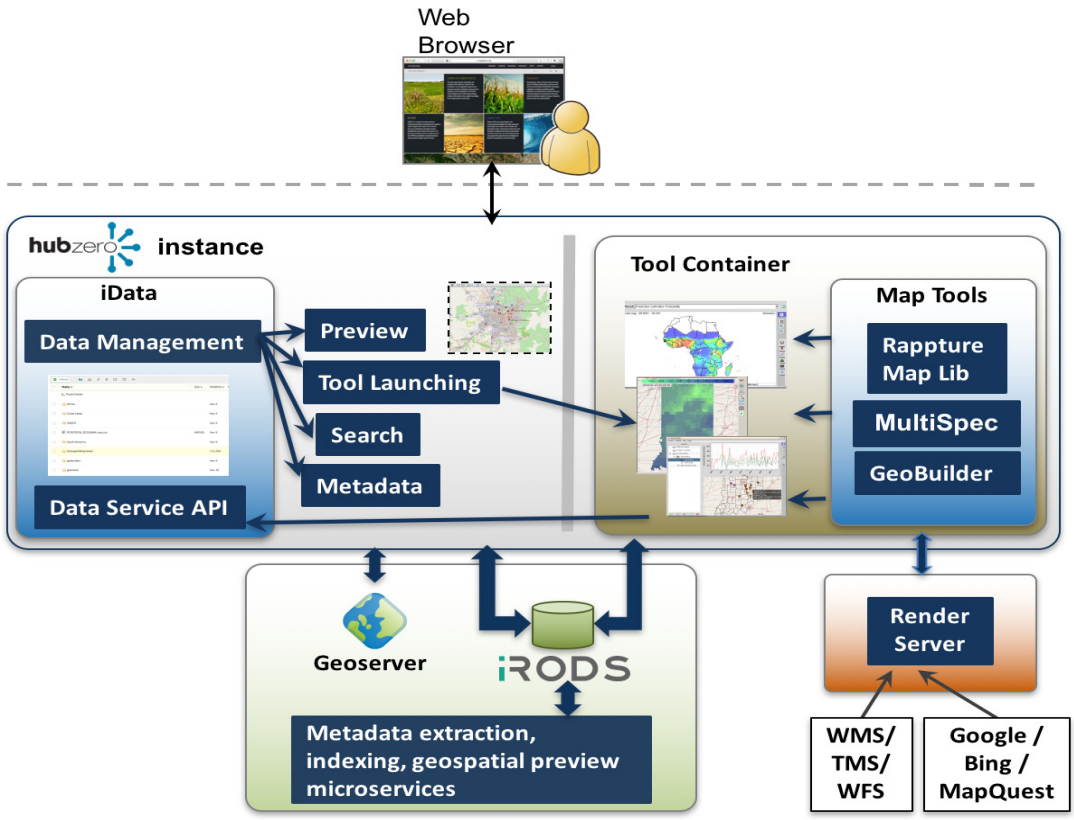

Figure 1: GABBs system diagram. Pieces in dark blue represent GABBs components; blue, green and orange regions represent natural packageable resource blocks.

(1) They make for ideal test-beds when developing such packages. While personal virtual machines (VM) can be used for such tests, an individual VM for each distinct GABBs resource can be a resource drain on personal machines. In addition, resources such as the visualization server have hardware requirements that cannot be satisfied on typical personal computers.
(2) Individual users of GABBs who do not have access to compute resources can get up and running with a working hub and GABBs capabilities in a few hours on nominally priced AWS compute resources. Even users who have pre-existing hubs may choose to first test GABBs on AWS resources before committing to an install on their own hubs. As more organizations and universities move towards cloud-managed resources, the availability of a GABBs installation for AWS increases its visibility.
(3) The necessary rpm and deb packages for external installations of GABBs are a byproduct of this cloud-enabling process. While AWS allows for snapshots of machines to be saved and made public, sustainability is better served by using upgradeable packages for any software installation. In addition, AWS machine images cannot contain hardcoded passwords, necessitating a configurable installation.

The above considerations and the HUBzero team's use of the AWS Marketplace as their dissemination venue led us to select AWS as our cloud-deployment venue of choice. Due to the multiple interconnected resources that constitute a typical GABBs installation, any cloud-based deployment would require the separate deployment of these individual resources, followed by software installation and the configuration of the connections between these resources. While these latter steps can be delegated to the user by simply pointing them to a package repository and providing installation scripts and configuration instructions, our goal is to make the install as simple as hitting a "go" button. This reduces the barrier to entry for the domain scientists that are our primary target audience while also minimizing the possibility of user error.

The AWS CloudFormation service simplifies this task of resource deployment, software installation and configuration. CloudFormation is organized around templates and stacks where templates define the number and type of resources required, their properties and parameters and optional metadata such as additional package repositories and initialization scripts. Stacks on the other hand are the set of actual AWS resources deployed using a template as a blueprint.

4.2 CloudFormation for GABBs

While AWS CloudFormation templates can include any of the AWS resource types such as elastic compute cloud servers (EC2), databases, DNS servers, load balancers or security groups; the simplest GABBs installation only requires three separate EC2 instances and a security group or more simply, a firewall. The goal is to allow all traffic between these instances by default while allowing the user to set up inbound access from their personal machines or organization as needed. The three EC2 instances correspond to distinct pieces identified before:

(1) An iRODS server with a local installation of GeoServer and the necessary microservices
(2) A HUBzero webserver that also serves as the execution host for tool containers, includes the GABBs CMS, the iRODS FUSE mount and the MultiSpec and GeoBuilder tools and their dependencies
(3) A visualization (render) server

An EC2 resource definition in a CloudFormation template usually requires a few mandatory properties; for example, the type of instance (i.e. hardware specifications) and the Amazon Machine Image (AMI) to be used for this instance. Every EC2 instance is based on an AMI, akin to a virtual machine's disk image. AWS provides both official and other hosted AMIs for different versions of common operating systems. The HUBzero AWS Marketplace image is based on CentOS 6.5, which we use for the iRODS server as well. The render server is based on Debian 8 and requires an AWS GPU (graphics processing unit) instance. In the simplest sense, AMIs are just snapshots of an instance. Any software installed on the instance before taking the snapshot, is available on any new EC2 instance created using that snapshot. However, the AWS Marketplace places restrictions on the contents of AMIs that can be published. For instance, they cannot contain hardcoded administrator usernames or passwords. This restriction does not affect the render server whose AMI can be created from a snapshot of a Debian 8 machine that has all the necessary software installed. However, in the case of the hub and the iRODS servers it necessitates starting with a barebones AMI and scripting the installation of additional software using randomly generated or user-specified usernames and passwords.

As mentioned previously, such installation could be delegated to the user, providing them with instructions for accessing the instance and installing necessary software. However, to simplify GABBs setup, it was decided to automate as much of it as possible. This requires scripting these installations to run after instance boot up. CloudFormation supports two separate but related initialization mechanisms, UserData and cloud-init. Broadly speaking, UserData allows a bash script to be provided as a parameter for an EC2 resource in a CloudFormation template and is then run on instance boot up. Cloud-init provides directives for placing files at certain locations on the instance, adding package repositories, running commands on boot up and creating additional system users to name a few. Thus, a combination of cloud-init and UserData can be used to add an external repository (containing our GABBs packages in this case) and then install packages from that repository after instance boot-up.

Since we use CentOS for both the HUBzero and iRODS servers, the necessary software is packaged as rpms. A useful component of an rpm package is the post installation script that supports sequential command execution just like in a shell script. The task of installing a few packages followed by some configuration steps can be automated by creating a new rpm package that includes these packages as dependencies and performs the configuration actions in its post installation script. A simple randomizer function can be added to the script to generate random passwords for use in conjunction with other configuration commands. However, there are some configuration tasks that do not fit quite so nicely into this cloud-init, UserData and post installation script paradigm. Some of these issues stem from CloudFormation deployment essentially being a sequential process (i.e. completing deployment and initialization of an instance before moving on) that does not support interleaving instance deployment and configurations. Other issues result from lack of parameterization support for rpm installation; in fact, rpms are intended for non-interactive installation. We describe both these shortcomings below in the context of GABBs installation and our approach for getting around them. It should be noted that modern configuration management tools such as Puppet and Chef can be employed in conjunction with CloudFormation to simplify some of these software installation and configuration tasks; however it is our belief that these shortcomings would still persist in those solutions. Figure 2 illustrates a simplified, partial GABBs CloudFormation template and some of these workarounds.

5 GABBS PACKAGING ISSUES

5.1 Interleaved Deployment and Configuration

There are certain dependencies between the various EC2 instances making up the GABBs installation. For example, the iRODS server requires knowledge of the Apache Solr endpoint on the hub webserver. This allows automatically captured file metadata to be indexed in Solr in support of hub search for hub project files. Obviously, the hub webserver needs to be aware of the iRODS server's IP address or DNS to configure the FUSE mount and GABBs plugins. While it is straightforward to reference the DNS or IP addresses of EC2 resources in a CloudFormation template, this sets up a dependency chain between an EC2 resource and a resource that references its attributes. CloudFormation resolves such dependencies by determining a sequential ordering for resource deployment; a referenced resource is always deployed before the referrer. It is obvious that our scenario here has a cyclic dependency between the iRODS and hub webserver that CloudFormation prohibits. It should also be obvious that these dependencies essentially occur at the initialization/configuration step. Such issues could be alleviated by a two-step process where all the resources are deployed first followed by an initialization step where all resources can be configured with complete knowledge of attributes such as DNS and IP address of all the deployed resources. Lacking such a feature in CloudFormation, we are forced to seek alternate solutions for this issue.

Since the iRODS server needs to be installed before it can be configured to point to the hub's Solr endpoint, it is forced to be deployed and configured first by referencing its DNS in the hub resource's UserData. The hub's UserData script uses the referenced

iRODS server's DNS to set up the FUSE mount. All that remains then is to update the IRODS server configuration for hub Solr access. Fortunately, iRODS provides a microservice that can be used to execute commands (essentially, shell scripts) on the server. The microservice also allows specifying arguments for the command. Recall that iRODS microservices can be executed on-demand via an iRODS rule. We exploit this capability by creating an iRODS rule file (on the hub webserver) that executes this microservice to modify the server configuration appropriately. The microservice in effect executes a shell script installed on the iRODS server and is provided the hub's DNS as an argument. iRODS rules can be executed on-demand using iRODS client commands that have been installed on the hub webserver.

It should also be pointed out that CloudFormation by default does not wait until all UserData and cloud-init steps for an instance have been performed before moving on to create and configure dependent resources. However, CloudFormation provides another AWS resource, WaitConditions that can be used for precisely this purpose. WaitConditions, just like EC2 resources, can be set to either depend on or have other resources depend on them. We will not describe how WaitConditions work in detail here; essentially, they can be signaled from UserData on EC2 resources and are considered created only when such a signal is received. By placing such signaling at the end of the UserData section, dependent resources can be set to wait until an EC2 instance has been created and initialized before being created themselves.

5.2 Shared Parameters

Some parameters are required during the configuration of multiple EC2 resources in a template. For instance, the hub webserver requires an iRODS username and password to set up the FUSE mount. Any such users need to be configured and added on the iRODS server first and shared with the hub webserver. Fortunately, CloudFormation supports such template-wide parameters, but not without some shortcomings. For one, while such parameters can have default values, they need to be hardcoded and cannot be assigned using randomizer functions. Subsequently, the only option to avoid using hardcoded values is to prompt the user for them. Another downside of requiring template-wide parameters is that configuration tasks using these parameters cannot be accomplished in rpm post installation scripts. Instead, additional scripts need to be created for such configuration tasks, installed via rpms on the various EC2 instances and run in conjunction with these parameters via UserData. Prompting for such internal configuration parameters (such as the iRODS username and password) places an unnecessary burden on the user while also necessitating explanatory text describing the parameter's role.

6 CONCLUSIONS AND FUTURE WORK

As science gateways begin to find widespread use, their ease of deployment can often affect their popularity and adoption. With institutions increasingly turning to cloud computing resources and data curation playing an increasing role in research projects, cloud-enabling a general-purpose data management framework such as GABBs is a natural progression. In this paper, we described our ongoing experience deploying the various resources that comprise GABBs onto the AWS cloud computing platform and some of the quirks of the CloudFormation paradigm that we had to work around. Going forward, we will seek to better exploit the strengths of cloud computing platforms. In particular, we hope to use autoscaling and load balancers to scale up execution hosts and render servers in response to variable resource loads, with the middleware automatically detecting and exploiting these new resources as they get deployed. We also plan to introduce additional flexibility in our CloudFormation template, allowing advanced users to choose their desired resource configurations.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation grant number 1261727. We would also like to thank David Benham and Nicholas Kisseberth from the HUBzero team for providing us with the baseline HUBzero AMI. The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions.

REFERENCES

[1] 2013. GABBs: NSF DIBBs: Geospatial Data Analysis Building Blocks. (2013). http://www.nsf.gov/awardsearch/showAward?AWD_ID=1261727
[2] L. Biehl and D. Landgrebe. 2002. MultiSpec - A Tool for Multispectral-Hyperspectral Image Data Analysis. *Computers and Geosciences* 28, 10 (2002), 1153–159.
[3] L. Biehl, L. Zhao, C. X. Song, and C. G. Panza. 2017. Cyberinfrastructure for the Collaborative Development of U2U Decision Support Tools. *Journal of Climate Risk Management* 15 (2017), 90–108.
[4] T. Hertel and N. B. Villoria. 2014. GEOSHARE: Geospatial Open Source Hosting of Agriculture, Resource and Environmental Data for Discovery and Decision Making. (2014). https://mygeohub.org/resources/723
[5] R. Kalyanam, R. A. Campbell, S. P. Wilson, P. Meunier, L. Zhao, B. A. Hillery, and C. Song. 2016. Integrating HUBzero and iRODS: Geospatial Data Management for Collaborative Scientific Research. In *The 2016 iRODS User Group Meeting*.
[6] R. Kalyanam, L. Zhao, C. X. Song, Y. L. Wong, J. Lee, and N. B. Villoria. 2013. iData: A Community Geospatial Data Sharing Environment to Support Data-driven Science. In *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment*. ACM.
[7] G. Klimeck, M. McLennan, S.P. Brophy, G. B. Adams III, and M. S. Lundstrom. 2008. nanoHUB.org: Advancing Education and Research in Nanotechnology. *Computing in Science and Engineering* 10, 5 (2008), 17–23.
[8] M. McLennan and R. Kennell. 2010. HUBzero: A Platform for Dissemination and Collaboration in Computational Science and Engineering. *Computing in Science and Engineering* 12, 2 (2010), 48–52.
[9] V. Merwade, W. Feng, L. Zhao, and C. Song. 2012. WaterHUB - A Resource for Students and Educators for Learning Hydrology. In *Proceedings of the XSEDE12 Conference*.
[10] Arcot Rajasekar, Reagan Moore, Chien-yi Hou, Christopher A Lee, Richard Marciano, Antoine de Torcy, Michael Wan, Wayne Schroeder, Sheau-Yen Chen, Lucas Gilbert, et al. 2010. iRODS Primer: integrated rule-oriented data system. *Synthesis Lectures on Information Concepts, Retrieval, and Services* 2, 1 (2010), 1–143.
[11] L. Zhao, C. X. Song, and L. Biehl. 2016. MyGeoHub Science Gateway for Spatial Data and a Model for Sustainability. In *Gateways 2016*.

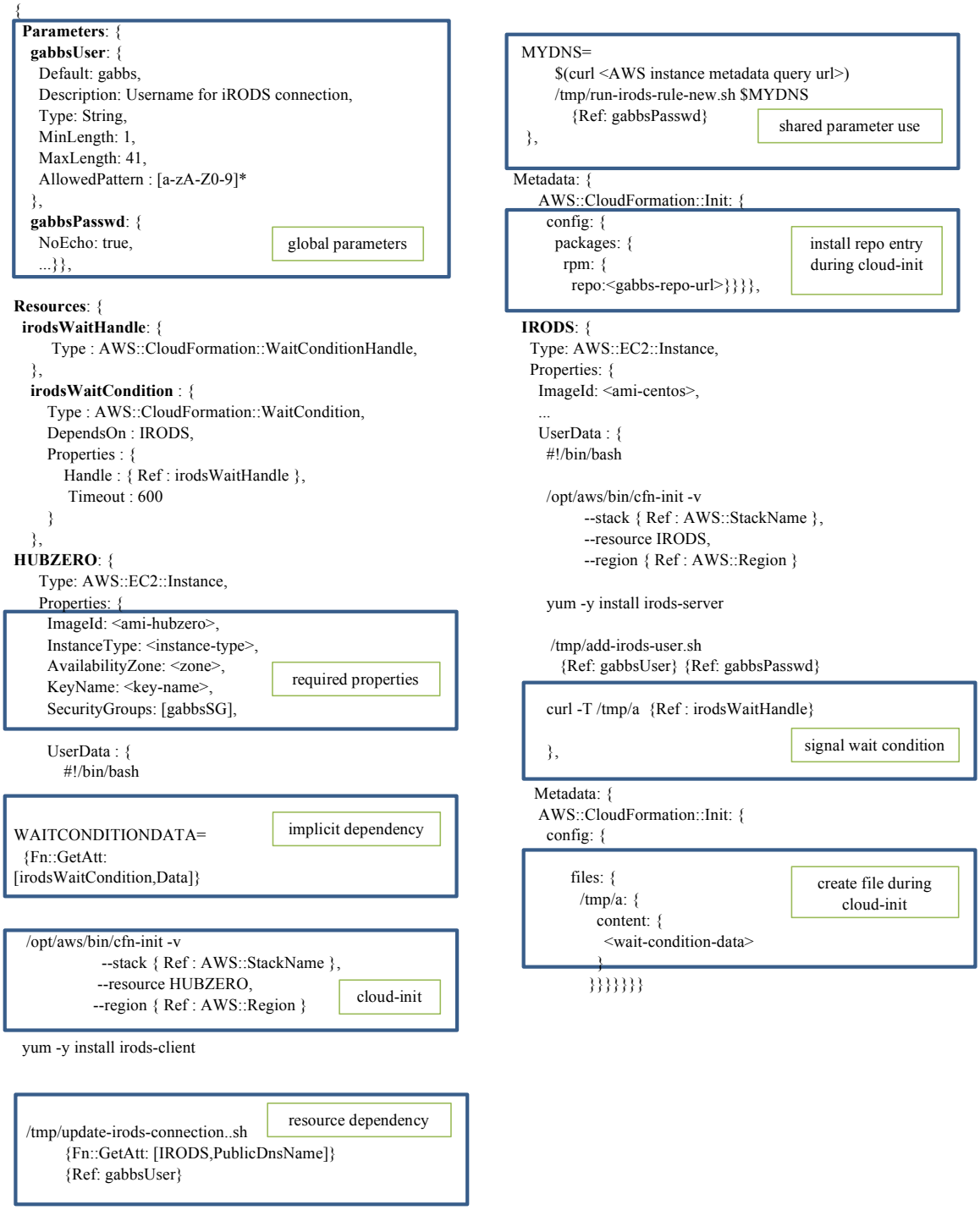

Figure 2: Partial AWS CloudFormation Template