Building Science Gateway Infrastructure in the Middle of the Pacific and Beyond: Experiences using the Agave Deployer and Agave Platform to Build Science Gateways

Extended Abstract

Sean B. Cleveland
University of Hawaii
Honolulu, HI, 78705
USA
seanbc@hawaii.edu

Rion Dooley
Texas Advanced Compute Center;
University of Texas at Austin
Austin, TX, 78705
USA
deardooley@gmail.com

David Perry
University of Melbourne
Parkville VIC 2010
Australia
perryd@unimelb.edu.au

Joe Stubbs
Texas Advanced Compute Center;
University of Texas at Austin
Austin, TX, 78705
USA
jstubbs@tacc.texas.edu

John M. Fonner
Texas Advanced Compute Center;
University of Texas at Austin
Austin, TX, 78705
USA
jfonner@tacc.texas.edu

Gwen A. Jacobs
University of Hawaii
Honolulu, HI, 78705
USA
gwenj@hawaii.edu

ABSTRACT

In order to increase support for diverse projects amongst a wide range of research areas in accessing advanced computational and data resources, both local and national, the University of Hawai'i at Manoa (UH) and the University of Melbourne, Australia (Melbourne) partnered with the Texas Advanced Computing Center (TACC) to utilize the Agave platform. However, due to distance and the unique geographical locations of Hawai'i and Australia it was necessary to setup local Agave platform instances to provide responsive and robust middleware against which flexible science gateways could be constructed. To lower the entry barrier, required staff, and time required to stand up a local infrastructure, UH became the first external site to use the Agave Deployer which provides a combination of DevOps automation tools and containers to deploy and maintain a functional local Agave authentication/authorization, core science, and data persistence API instances. UH worked with TACC on testing the initial release of the Agave Deployer to provision the local UH infrastructure and later assisted Melbourne in adopting the Agave Deployer to stand up their infrastructure. We present the experiences and lessons learned in deploying and developing science gateway infrastructure and applications at these two institutions.

CCS CONCEPTS

• Information systems → Information system applications; Computing platforms

KEYWORDS

ACM proceedings, Science Gateway, HPC

ACM Reference format:

S.B. Cleveland, R. Dooley, D. Perry, J. Stubbs, J.M Fonner, G.A. Jacobs. 2018. SIG Proceedings Paper in word Format. In *Proceedings of the Practice and Experience in Advance Research Computing conference, Pittsburgh, Pennsylvania USA, July 2018 (PEARC'18)*, 8 pages. <https://doi.org/10.1145/3219104.3219151>

1 INTRODUCTION

The Agave Platform is commonly known as a hosted, Science-as-a-Service cloud platform available free of charge for the open science community[1] (Agave). Agave is also a fully open source project available for deployment and utilization within individual projects and organizations. In this paper we present the motivation, experiences, and lessons learned deploying instances of the Agave Platform on-premise at the University of Hawai'i at Manoa (UH) in the United States, and the University of Melbourne (Melbourne) in Australia. While the two universities are distinctly different, they had two things in common that helped shape their decision

1.1 A Rare Opportunity

Both institutions found themselves in the enviable position of having the opportunity to actively reimagine what transformative cyberinfrastructure (CI) would look like for their users over the next decade. UH was preparing for an award from

This work is licensed under a Creative Commons Attribution International 4.0 License.

PEARC '18, July 22–26, 2018, Pittsburgh, PA, USA
© 2018 Copyright is held by the owner/author(s). ACM ISBN 978-1-4503-6446-1/18/07. <https://doi.org/10.1145/3219104.3219151>

the National Science Foundation (NSF) Established Program to Stimulate Competitive Research (EPSCoR). They needed ways to pull together different data sources, applications, and instrumentation from researchers scattered across the islands in hopes of securing Hawai'i's water future. They envisioned gateways taking a leading role in their future efforts and, thus, wanted infrastructure that could support any sort of science gateway scenario.

Melbourne was investigating technologies that provided simple access to HPC resources. They were looking for API driven solutions that abstracted cluster environment details away from application developers to reduce development time, simplify maintenance, and enable software portability between HPC facilities.

1.2 Experience with Innovation

As each team looked forward, they identified the need to flexibly support current and future computing technologies. At UH, the decision was simply a matter of experience. Their project lead and Director of CI, Gwen Jacobs, has been involved in the evolution of CI in the US for several years through her participation in the advisory boards for TeraGrid[2], iPlant[3, 5], Jetstream[7], Internet2, and EarthCube. UH currently operates campus HPC systems and regularly partners with commercial cloud providers on research pilots to extend their campus infrastructure. Ensuring they could continue in this dual fashion while still leveraging future trends was a lesson they learned long ago.

UH was also no stranger to building science gateways. Having partnered on the development of multiple gateways in the past, they had experience both with the challenge of building successful gateways and with the advantages that cohesive CI can bring to their development. At the time they were planning, however, very little was known about the gateway or gateways they would be building. Language, technology, hosting, target environment, staff, scope, and even budget were still up in the air. Their choice of infrastructure had to be flexible enough to deliver the capabilities their gateway required while giving them the freedom to develop with the people, tools, and technology best suited for them.

Similarly, Melbourne operates their own HPC facility as well as a region of the Australian National eResearch Collaboration Tools and Resources (Nectar) Cloud, a national program providing support and education around a sizable OpenStack-based IaaS cloud. For the last 4 years they have been exploring different approaches to cloud computing in an effort to better serve classic and emerging computational sciences. Ensuring that their technology choice could be supported regardless of the direction the national CI took going forward was paramount to their decision.

As resource providers for Nectar, Melbourne has participated in the explosion of science gateways and VLS over the past 6 years. They witnessed first hand the diversity of technology, tools, and techniques with which their users leverage the cloud to build their software. Looking forward, they knew that a critical component of future eScience was finding synergy

between HPC and cloud infrastructure. Their choice of infrastructure had to facilitate seamless access to compute, storage, and instrumentation regardless of its operational semantics or physical proximity.

2 TWO INSTITUTIONS, ONE SOLUTION

Having seen first-hand the diversity of ways which science gateways can be built, the decision to deploy Agave came down to a handful of specific drivers for each institution. In the following sections we describe the major decision points that led each to choose the Agave platform and, specifically, to deploy it on-premise.

2.1 UH EPSCoR Evaluation Led To Agave

The initial driving project for UH to deploy science gateway infrastructure was the Hawai'i EPSCoR 'Ike Wai project. The 'Ike Wai gateway needed to provide resources to support water sustainability research in Hawai'i. The gateway infrastructure needed to provide data management, computation, analysis, and visualization tools to support reproducible science, modeling, data discovery, and decision support across multiple domains for the research team and wider Hawai'i water community. An existing collaboration between UH and Agave developers provided consultation opportunities that highlighted the suitability of Agave for the 'Ike Wai gateway.

2.2.1 Many Projects. Both of the potential UH gateway projects needed to tie data and advanced computing resources together to support multiple domains. The 'Ike Wai project alone includes microbiology, geochemistry, geophysics, economics and humanities researchers that all have data, computational and modeling workflows. The success of previous Agave supported projects such as: Araport[4], which offers gene and protein reports with orthology, expression, interactions, and the latest annotation analysis tools along with community apps; the iPlant Collaborative[3, 5] now CyVerse, (<http://www.cyverse.org/>) which provides scientists with powerful computational infrastructure to handle huge datasets and complex analyses that enable data-driven discovery; and Designsafe-CI[6] a web-based research platform of that provides the computational tools needed to manage, analyze, and understand critical data for natural hazards research, made it very desirable to UH as a platform for supporting broad use cases going forward.

2.2.2 Reuse. The ability to utilize common infrastructure for multiple projects had the potential to be a big win in terms of both staff and resources at UH. The fact that the Agave architecture has such functionality baked in was a big check in its favor. It meant that it could enable both the potential gateway projects and scale to support more with minimal effort.

2.2.3 Scale and Flexibility. It is difficult to predict adoption of new CI. Undertaking a large infrastructure build out always carries with it some risk. Having the ability to begin with a modest deployment and scale up as demand increased was critical in managing risk in the early days of the project. Agave's microservice architecture offered flexibility in scaling individual

APIs or the entire platform as needed. The opportunity for predictable investments of time and resources that grow with, rather than ahead of usage helped mitigate much of the risk in choosing Agave.

2.2.4 Support. As with most research projects there is rarely as much personnel time available as would be deemed ideal. UH's experience having essentially half of one FTE responsible for managing all the infrastructure is as tight as it could get and still function as a production environment. This is only possible by leveraging the shared expertise from TACC and other Agave community members. The Agave Deployer, scripts, public Docker images, and Agave Slack channel were invaluable resources where UH could turn for support from Agave developers, operators, and other community members managing infrastructure.

2.2.5 Production Ready. Agave's components have been built, tested and hardened in production environments for more than seven years; powering gateways with thousands of users. The Agave team publishes the same images that run the hosted cloud instance of Agave to their public Docker Hub account and makes their DevOps tooling available as open source projects. Adopting the existing automation and DevOps best practices that manages a dozen other tenants and powers hundreds of existing gateways dramatically accelerates the path to production.

2.3 Melbourne Evaluation Led to Agave

Australia supports science gateways through Nectar. Melbourne led a program in collaboration with Nectar and TACC to investigate how Agave could be used by local science gateways, with an emphasis on simplifying HPC integration, and accelerating development. The initial goal was to deploy a local instance of Agave to inform requirements for ongoing service support.

2.3.1 Cloud Native. As a resource provider for Nectar, Melbourne wanted a solution that, ideally, could run on existing Nectar resources. The fact that Agave can be deployed as a cloud native application meant that it could run as close as possible to the gateways it would support. It also meant that support staff could leverage the best practices and lessons learned from operating a large distributed platform on Nectar to help other users utilizing the cloud.

2.3.2 General Science Support. A platform is only as successful as those who find success using it. To achieve widespread adoption within Nectar's eResearch community, infrastructure must support researchers from a wide range of disciplines. Agave's proven ability to support researchers spanning both life and physical sciences gives it a good opportunity to find traction.

2.3.3 Flexible For Future Use Cases. In studying the existing deployments of Agave, it was clear that aside from basic common functionality, each tenant was tailored uniquely for its user community. Even for large organizations like CyVerse, Agave was able to evolve over time to support new computing paradigms, storage resources, and identity scenarios. This

flexibility was critical as Australia's federal government is still deciding what the nation's national eResearch infrastructure will look like over the next 5-10 years.

2.3.4 APIs Over UIs. An important criteria when evaluating gateway support infrastructure was the availability of application programming interfaces (API) to programmatically access the infrastructure. The fact that Agave had both a REST API and client libraries in multiple languages meant end users would be able to leverage HPC and cloud resources natively from their application in the way best suited for their use case.

2.3.5 Support. While Melbourne was initially only seeking to deploy Agave for evaluation purposes, the availability to have questions answered and pose questions to the community were important factors in building confidence prior to the first deployment. Given that an external contractor would be hired to stand up the proof of concept, the ability to get timely answers to questions had the potential to make or break the evaluation. Thankfully support was readily available throughout the deployment.

2.3.6 Proof of Concept With Path to Production Unlike UH, Melbourne did not enter the evaluation with a clear path to production. At the time of this writing, funding for the Nectar project is currently undecided awaiting word from the Australian government on the scope, duration, and budget for the project going forward. The desired outcome of the deployment was to get a sense of the effort, time, and resources it would take to operation a production instance of Agave for the entire Australian eResearch community.

2.4 Establishing a Partnership

As Tier 1 research organizations, both UH and Melbourne have been through evaluations and rollouts of large software projects before. As such, both organizations placed a significant value on the ability to establish an ongoing partnership with the Agave team at TACC and its community at large. TACC operates at UTC-05:00. UH operates at UTC-10:00, and Melbourne operates at UTC+10:00. Finding regular meeting times across those time zones can be difficult, let alone finding someone half a world away who can help when a problem appears, or provide answers when questions come to mind in off hours. Because neither group was alone in their adoption or utilization of Agave, they could draw from a global community of users and practitioners to answer questions at all hours of the day.

2.5 On Premise Agave

Agave has been successful functioning primarily as a hosted platform for a number of successful gateways. However, for UH and Melbourne, several factors necessitated local Agave deployments and prevent usage of the U.S. mainland Agave instances.

2.5.1 Autonomy. While many things can be offloaded to the cloud, UH and Melbourne both had a strong requirement to retain full autonomy over their infrastructure. UH campus IT expressed concerns about opening up LDAP access to an entity

outside the UH network. Potential issues involving restricted data access to local data repositories also caused some concern. Melbourne had strong federal regulations prohibiting certain information from residing outside the country. In both cases, the only option for a production instance would be an on-premise deployment.

2.5.2 Physics and Geography. Even over the fastest networks, latency and timeouts are real considerations when making frequent TCP request to services hosted on the mainland. Initial tests from UH using the Agave Public Tenant suffered unpredictable service degradation on more than one occasion. While this was later somewhat mitigated by configuration changes within TACC's data center, the difference in user experience when compared with an on-premise deployment was night and day. After several attendees at a UH training workshop expressed frustration at the impact it had on their experience, UH made the decision to explore an on-premise installation as soon as possible.

The situation was even more apparent at Melbourne. Australia and the U.S. truly are on opposite sides of the world. Even leading content delivery networks can take minutes to propagate changes there that regularly show up in a matter of moments in the mainland US. Even before evaluating Agave, the Melbourne team's experiences in evaluating other technologies heavily suggested they pursue on-premise solutions. Thus, after walking through several Agave tutorial and gathering feedback from other tenant stakeholders, they decided to move forward with an on-premise deployment.

3 DEPLOYMENT

Prior to standing up infrastructure, it was important to have a roadmap for its deployment and management. Neither team had an abundance of staff dedicated to managing infrastructure. UH had a single FTE tasked with both front-end gateway development and managing backend operations. Melbourne hired a contractor just long enough to get their instance up. Fortunately in both cases, this proved to be enough.

Agave is a multi-tenant, cloud-native distributed system. All services within the platform run as Docker containers, orchestrated as a single microservice architecture. The platform can be viewed as three logical tiers: platform services, science APIs, and support services.

Platform Services contain services providing identity and API management, client registration, tenant admin services, and documentation.

Science APIs contain the primary Science as a Service (SaaS) functionality used to power science gateways such as data and job management, app publishing, notifications, etc. This tier is further divided into a frontend collection of loosely coupled services, microservices, and backend service workers. The microservices serve the user HTTP requests to the REST APIs, and service workers handle processing of asynchronous requests such as data movement and app publishing.

Support services include databases, message queues, caches, object stores, service discovery, notification relays, and websocket proxies.

A tenant represents a group of users, applications, and entitlements. Each instance of Agave can support one or more tenants. Individual tiers and microservices can be replicated, configured, and scaled in support of a specific tenant, or shared to better leverage a single, consolidated resource footprint.

3.2 Automation

Automation is critical in the operation of any production software environment. As the complexity of the system increases, so too must the quality and capability of the tooling used to manage it. Going in, both sites engaged the Agave development team to learn about how they managed their existing installations.

The Agave team takes a DevOps approach to management of the hosted platform at TACC. They use Jenkins as both a continuous integration/continuous deployment (CI/CD) server, and as the primary interface to the Agave Deployer, a homegrown tool used to manage Agave's infrastructure and configuration. Each site adopted these tools as the basis their own automation.

3.1.1 Agave Deployer. The Agave Deployer utilizes Ansible, an IT automation tool, and a collection of custom Ansible playbooks, created by the Agave development team, to automate the deployment, testing, management, and scaling of the entire platform. The Deployer currently supports deploying to CentOS 7 Linux hosts via SSH. Very little is required of the deployment host other than a SSH server and login account with administrative privileges. The Deployer will install optimal versions of all libraries, and required software including the Docker runtime, configure DNS resolution, and set the appropriate firewall rules to secure the server.

The Deployer inherits Ansible's idempotent philosophy, ensuring that it can be repeatedly run to return a given tenant to a specific state. This is helpful when updating individual services as a single deployment playbook can safely be rerun to update a single service without tearing down the entire deployment to do so. The Deployer itself is stateless and ships as a Docker image, making it easy to integrate into existing CI/CD solutions such as Jenkins, RunDeck, and CircleCI.

It is worth noting that secrets and passwords are not included in the Deployer repository or code base by either site for security reasons. Instead, those are pushed to a dedicated CI/CD server where they can be secured and managed independently.

3.1.2 Continuous Integration/Deployment. A standard Jenkins server is the recommended way to provide CI/CD to Agave deployments. Jenkins provides solutions for managing secrets, configurations, processing webhooks, and role based access control out of the box. By leveraging the Deployer Docker image, Jenkins jobs can be created to mirror the desired Deployer functionality without requiring any software to be installed on the Jenkins host aside from Docker itself. Jenkins CI/CD server was integrated with the Agave Deployer so that specific workflows could be defined for updating specific

infrastructure components or restarting components. In addition to the Jenkins server UH setup a deployment server for using the Agave Deployer and to hold keys and credential information for managing the Agave servers. These two services coupled together provide a simpler interface for managing the Agave infrastructure as well as automated logging of activities. Further, UH was able to leverage the Jenkins server to integrate with the user interface deployment via Github webhooks to allow developers to deploy the user interface updates to the various servers by simple branch commits.

3.2 First Steps: Sandbox Deployment

This shift from the original plan of utilizing a UH tenant hosted within the TACC data center required additional coordination in April 2016 with the networking, security, identity management and technology infrastructure teams within UH ITS along with the TACC Agave team to architect an approved infrastructure solution to host the local UH Agave platform. In May of 2016 the approved infrastructure solution consisting of 3 virtual machines, each running CentOS 7 with 4 cores, 8GB of RAM and 50GB of local disk, hosted on the UH ITS enterprise VMWare cluster were provisioned and networked within the UH science DMZ to host the authentication/authorization, core science APIs and database persistence Agave middleware stack. In June of 2016 UH achieved a fully functional Agave platform local to UH.

As one of the first Agave deployments outside of TACC, the UH team worked closely with the Agave team at TACC to work through some of the initial pain points they experienced. One of the challenges they ran into were identifying, copying, and customizing the correct Ansible configuration files to create a new tenant. Prior to that, tenant names and configurations were hard coded into the Deployer's playbooks, so this information had to be abstracted out to allow for new tenants to be created on demand. Additional changes to add more flexibility in firewall configuration, networking, OS dependencies, and logging behavior were also made and fed back upstream into the project.

Once a tenant was successfully deployed, UH forked the Deployer to a private git repository where they added additional playbooks to manage their science gateway, support services, publish documents, and manage other project related tasks.

Melbourne began their deployment in May of 2017 after UH went to production and, as such, benefited from the experiences of the UH team. They engaged consultants Aptira to construct an instance on the Nectar cloud using the Agave Deployer and a new playbook developed by the Agave team from the experiences working with UH to deploy a "sandbox" instance of Agave. Due to some of the unique characteristics of the Nectar cloud, several customizations needed to be made to the sandbox playbook.

First, Nectar's installation of OpenStack restricts user access to software defined networking and automatically assigns public IP addresses to all new instances. This broke the networking and security groups configuration that Deployer attempted to do when provisioning new hosts.

Second, because all hosts have public IP addresses, additional work needed to be done to lock down the host firewalls to guard against attacks that would normally be mitigated by the default security groups.

Third, the instance flavors available on Nectar had slightly less disk than comparable flavors on the US Jetstream cloud and EC2, so roles to create and attach additional disk needed to be created to ensure sufficient disk for the services. In practice, larger instance sizes were selected rather than managing volumes simply out of convenience.

In June of 2017, Melbourne achieved a fully functional Agave sandbox local to the Nectar cloud.

3.3 Three Host Architecture

The initial deployment for both UH and Melbourne was a standard 3 host installation where each tier of the platform was deployed on a separate host. This represented the minimum fully functional deployment of Agave while still retaining the distributed nature of the platform and allowing for horizontal scaling without taking down entire tiers of services to do so.

3.4 Cloud, VM, or Metal

For UH the decision to use VMs was based on a few factors. First the CI team could rely on another entity in the central IT department to support maintaining all the hardware, software, storage and networking with more personnel and attention than they could hope to have in maintaining their own physical hardware. Second the VM infrastructure has additional space to support the gateways use case so there was little additional cost. Third, the connectivity and costs associated with cloud services made them less desirable than local VMs for UH.

For Melbourne, the decision to use local Nectar cloud resources was predetermined. The only consideration was assuring sufficient allocation and placement for persistent hosting. Agave did not require any exceptional instance flavors, and any additional disk requirements could be met through attaching additional block storage to the instances.

Both institutions initially provisioned virtual machines, each running CentOS 7 with 4 cores, 8GB of RAM and 50GB of local disk. UH VMs ran in their ITS enterprise VMWare cluster and were provisioned and networked within the UH science DMZ. Melbourne provisioned in their default Nectar availability zone and let the Deployer create the proper security groups and firewall rules to secure the hosts.

3.5 Validation and Testing

The Agave team uses a suite of Postman tests, Javascript code written to test API responses, to perform validation and regression testing on individual tenant updates and deployments. UH and Melbourne adapted TACC's validation suite to run against their on-premise deployment to validate their correct operation. The validation tests are integrated into the Deployer, but do not run by default on every deployment due to specific details related to how TACC performs rolling

deployments. For the initial deployments, both sites manually ran the collection using Postman's Newman runner. UH has since begun working to integrate these tests into their Jenkins CI/CD tasks.

4 SCALING UP FOR PRODUCTION

After gaining experience operating their single tenant for several months, the UH team decided to deploy two distinct production tenants; one in support of the 'Ike Wai project, and the other in support of UH. The decision for a separate UH tenant was due to the expectation that the 'Ike Wai gateway would have more external UH stakeholders and user, so by managing those users from a separate tenant, the UH gateway could be completely managed by the UH LDAP and UH account provisioning processes. This allowed better customization to each tenant while making permissions within each tenant a bit easier to maintain. In October of 2016 UH scaled up from a three host architecture to 8 VMs in the production infrastructure and transitioning the original sandbox deployment into a development/testing infrastructure.

4.1 Beyond Three Hosts Architecture

The Agave infrastructure deployments for development and production environments at UH currently requires five virtual machines (VMs) to support the Agave middleware components, this count excludes the web application servers, Agave development servers and CI/CD server (Fig. 1). As of this writing all VMs have been provisioned with 50GB of local disk space and the current VM core and memory resources for all services except the authentication/authorization database server and web servers are: 4 cores and 8GB of RAM. The authentication/authorization MySQL servers and the two web servers that host gateway user interface web applications each have 2 cores and 4GB of RAM.

The local disk space appears to be sufficient for the storage of the Docker images and assets. At an earlier release of Agave's files service, the core science APIs VM was not cleaning up after itself, and UH ran into an issue during a heavy file transfer period where the local disk filled. A quick wipe of the Docker containers and redeploy of the services fixed the issue within minutes. Creating a cron task to monitor and clean up intermediate files was another recommended workaround. An updated release by the Agave developers fixed the offending service and no further issues have been found with local disk space constraints.

The UH Agave platform services, science APIs and web application servers are hosted in the 10G science DMZ portion of the UH network to allow for fast data transfers and access response. The database VMs and support services are hosted behind the UH firewall to protect the data and services. The central gateway file data repository is physically hosted on an enterprise grade network attached storage (NAS) appliance located in the UH data center.

4.2 Understanding Bottlenecks

As UH prepared for a production deployment it was determined that maintaining all the databases on one server was going to be a performance bottleneck based on TACC's own experiences of managing their public Agave tenant deployment. We therefore heeded their advice and added dedicated servers for the MySQL database instances. In reviewing behavior and performance, we also determined that only the platform services and science APIs needed to be in the Science DMZ for the production deployment, and the other services could live behind the UH firewall with ports limited by service and IPs.

UH experienced an edge-case bottleneck when attempting a one-time generation of a large number of JSON records, >700,000, using the metadata API. Since the API is accessible via a REST endpoint, we built some scripts using the Agave CLI tools to create the JSON records and store them via parallel API calls (https://github.com/UH-CI/landuse_ingestor). However, like any web service, only so many requests could be handled at a time, and we determined that the UH deployment could handle about 30 of these requests a second before the server would start responding with errors indicating it could not process additional requests. Working with the developers at TACC, we increased throughput by reducing the number of API calls required to ingest the records and by identifying infrastructure pieces that could be put in place in the future to increase performance as needed. For instance, since the metadata API service is a web application itself, it could be moved onto a dedicated VM with more resources or replicated to multiple VMs with load balancing to increase request handling. The potential to easily scale portions of the infrastructure services is a desirable feature in a long-term gateway deployment.

Melbourne also ran into several bottlenecks while inadvertently stress-testing their sandbox deployment through use in a training workshop. They found that the number of requests for remote data being made along with the number of concurrent jobs being submitted consumed all available memory on the host. The problem was quickly resolved by increasing the memory on the science API host to 16GB and moving the science API service workers to a separate host with another 4 core and 16GB of RAM. This change addressed the memory issues and enabled the training to continue the next day with an order of magnitude more throughput.

In the following months, the larger sandbox instance was repeatedly tested against training and pilot project workloads. It was determined that with the addition of monitoring and CI/CD similar to what UH deployed, that the instance would be suitable for moderate production loads at its current configuration.

4.3 Deploy, Debug, and Destroy

For any production level service, low downtime is desirable. To minimize downtime due to issues or errors with deployments and updates, UH has a staging and test environment in place to assess the impact of new updates on the system and applications prior to rolling them into production (Fig. 1). When new releases of Agave components are available, UH deploys the new code into test environments to verify functionality with gateway software. There are minor differences between the TACC and

UH deployments in the network configuration, security, etc., that are accounted for in the test environment. This setup also helps identify unexpected behavior since UH can compare production responses to the latest updates in the test environment.

In addition to testing new code releases, we are also able to test recovering the system from issues with the network, VM, OS, or database. The use of VMs provides another layer of recovery when using VM snapshots in case package updates happen to conflict with Docker or database services. This architecture has allowed UH to successfully maintain the gateway infrastructure with minimal downtime for the past two years. For instance, some UH data center VMs experienced file system issues; a network went down in the virtual host cluster, and connections to storage mounts disappeared. The recovery of VM images and following redeployment took less than 20 minutes once network connections were reestablished to the VM's storage endpoints. In another example, one of the development authentication databases became corrupted on disk. The UH team was able to recover the VM from a snapshot, re-deploy the database, and then recover the backup in about an hour thanks to the infrastructure setup and operations.

Figure 1: Diagram of Agave Platform infrastructure deployed at UH. Yellow boxes denote development/testing virtual machine components. Green boxes denote production components. Dotted lines denote network boundaries.

4.4 OAuth2 Flows

The Agave ToGo application comes with implicit grant authorization implemented for user logins into the configured tenant. However, the implicit grant flow issues one-hour tokens, and 'Ike Wai' users tended to use the gateway for longer than an hour, which resulted in a break in their workflow to issue a new authorization token. This caused some complications when users would; for example, upload a large number of files. In order to extended initial authorization token time limits, offer refresh tokens, and integrate a user whitelist into the authentication workflow, a Nodejs authentication API (<https://github.com/UH-CI/agave-login-api>) was built and integrated into the 'Ike Wai' gateway.

4.5 Persistence

As mentioned previously UH heeded the advice of moving to dedicated servers for the authentication/authorization and core science API MySQL databases and core science API Mongo database. Further, to increase data redundancy UH setup hourly database dumps for backups via logrotate and use cron to shift a daily dump of each database to another storage system. Additionally, the 10TB storage volume to store gateway data files is backed up to a second storage location in the UH data center nightly as well as encrypted and backed up to a cloud storage location offsite for disaster recovery.

4.6 Secrets

When UH first started managing Agave everything was happening from an administrators own workstation. This did not scale and did not reveal to the rest of the team how management was taking place. Therefore, in addition to the servers hosting Agave, UH deployed a server for storing configurations, secrets, passwords and keys related to managing and deploying the various Agave components. This server sits behind the UH data center internal firewall with limited access from the UH network. Further, this server also hosts the Jenkins instance that runs the Agave Deployer scripts used for development/staging and production environments.

4.7 API Manger and Roles

A powerful feature of the API Manager bundled into Agave is role-based permission on the APIs themselves. For the 'Ike Wai' gateway, UH locked down the creation of client API keys to administrators only thereby restricting access to the APIs through the gateway UI and clients created for specific application integrations. In the more general UH gateway the default roles deployed with Agave remain allowing all UH users the ability to generate client API keys to build applications and utilize the Agave CLI Tools, which require a client API key.

5 GATEWAY DEVELOPMENT

For the past two years UH has been developing gateway software and user interfaces against the local Agave platform. The resulting 'Ike Wai' gateway built on Agave ties data and advanced computing resources together to support domains including microbiology, geochemistry, geophysics, economics and humanities with computational and modeling workflows delivered in a user friendly web based interface..

Melbourne deployed a pre-packaged gateway called Agave ToGo. Agave ToGo is an example gateway application written in Angularjs, JavaScript and HTML demonstrating all Agave's features. Melbourne uses their Agave ToGo instance for a training and pilot projects.

5.1 ToGo for Rapid Prototyping

Following the deployment of the Agave sandbox infrastructure at both UH was were able to deploy an Agave ToGo gateway web application instance and customize it to use their tenant instances. For UH the Agave ToGo instance enabled a proof of

concept for user feedback and interaction cycles to occur every two week very early in the development process. This paved a path to additional gateway specific features and customizations such as, hydrology file metadata annotation, map based tagging, free-text search and spatial querying for data. All this could take place while still allowing access to basic gateway features early on in the 'Ike Wai project.

The use of Agave ToGo also availed the UH developers to a community of developers, adopters, and evangelists where they could ask questions and receive advice on implementing features such as the backend authentication service used to whitelist users and provide persistent token refresh. To date the 'Ike Wai gateway extends ToGo with interfaces to manage computation, data transfer, annotation, and search with full-text and spatial indexes of annotated metadata and files, with filtering of results using an interactive leaflet map (Fig. 2).

Figure 2: Screenshot of the 'Ike Wai gateway's spatial search interface.

5.2 Onboarding Stakeholders and Training

The 'Ike Wai gateway includes a comprehensive help section consisting of answers to frequently asked questions (FAQ) and a walkthrough that takes a new user from initial login through all the available actions in the system. Both the walkthrough and the FAQ sections include short videos showing each individual action and are combined with individual and group training sessions for users.

UH has held workshops related to Agave and used the general UH tenant as the infrastructure in the training. Example genomics training using the UH infrastructure here (<https://github.com/UH-CI/genomics-workshop-04122017>). At the time of this writing the 'Ike Wai gateway and UH gateway have has 43 and 64 user accounts respectively.

5.3 Feature Update Automation

UH gateway developers work through a release cycle by planning a set of features and bug fixes and then rolling those out as they are completed to the testing and staging application servers as we go and then deploying the full release to the production application server once validation is complete. To enable quicker deployments of gateway testing and staging code for the 'Ike Wai gateway we integrated Jenkins tasks to poll code

repositories on specific branches and when updates occur deploy the latest code to the web servers. This allows the UH developers to easily deploy updates without accessing the backend servers themselves.

6 FUTURE WORK

Currently the UH is partnering with the Science Gateways Community Institutes for cybersecurity audits and user experience consulting to improve the gateways. Additionally, UH is working to add Jetstream resources into the UH Agave infrastructure to support interactive computing environments and notebooks for UH researchers. Both UH and Melbourne will continue to partner closely with TACC and the larger Agave community to evolve and improve Agave and its ecosystem of gateway development and management tools.

7 CONCLUSION

Deploying and maintaining CI is full of challenges, and science gateway infrastructure is no exception. We have observed multiple paths to deploying Agave infrastructure that can scale with staffing and users to fit both administration and research requirements. There are many ways to manage these resources, but by using the practices described in this paper, along with community developed DevOps codes and validation and testing scripts, the amount of time UH and Melbourne were required to dedicate towards deploying and maintaining testing and production environments was significantly reduced. In the end both institutions have successful science gateway infrastructure serving users and moving research ahead.

ACKNOWLEDGMENTS

This work was made possible by grant funding from the following: Collaborative Research: SS2-SSI: The Agave Platform: An Open Science-As-A-Service Cloud Platform for Reproducible Science NSF ACI #1450413, RII Track 1: 'Ike Wai Securing Hawai'i's Water Future NSF OIA #1557349

REFERENCES

- [1] Dooley, R., Fonner, J., Jacobs, G. and Brandt, S. 2018. The Agave Platform: An Open Science-As-A-Service Cloud Platform for Reproducible Science. (2018). DOI:<https://doi.org/10.6084/m9.figshare.4675765.v2>.
- [2] Catlett, C.E. 2005. TeraGrid: A Foundation for US Cyberinfrastructure. *Network and Parallel Computing* (Berlin, Heidelberg, 2005), 1–1.
- [3] Goff, S.A. et al. 2011. The iPlant Collaborative: Cyberinfrastructure for Plant Biology. *Frontiers in Plant Science*. 2, (2011). DOI:<https://doi.org/10.3389/fpls.2011.00034>.
- [4] Krishnakumar, V. et al. 2015. Araport: the Arabidopsis Information Portal. *Nucleic Acids Research*. 43, D1 (Jan. 2015), D1003–D1009. DOI:<https://doi.org/10.1093/nar/gku1200>.
- [5] Merchant, N., Lyons, E., Goff, S., Vaughn, M., Ware, D., Micklos, D. and Antin, P. 2016. The iPlant Collaborative: Cyberinfrastructure for Enabling Data to Discovery for the Life Sciences. *PLOS Biol.* 14, 1 (Jan. 2016), e1002342. DOI:<https://doi.org/10.1371/journal.pbio.1002342>.
- [6] Rathje, E.M. et al. 2017. DesignSafe: New Cyberinfrastructure for Natural Hazards Engineering. *Natural Hazards Review*. 18, 3 (2017), 6017001. DOI:[https://doi.org/10.1061/\(ASCE\)NH.1527-6996.0000246](https://doi.org/10.1061/(ASCE)NH.1527-6996.0000246).
- [7] Tudoran, R., Nano, O., Santos, I., Costan, A., Soncu, H., Bougé, L. and Antoniu, G. 2014. JetStream: Enabling High Performance Event Streaming Across Cloud Data-centers. *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems* (New York, NY, USA, 2014), 23–34.