The Agave Platform: An Open, Science-as-a-Service Platform for Digital Science

Rion Dooley
Texas Advanced Computing Center
The University of Texas at Austin
Austin, TX, USA
dooley@tacc.utexas.edu

Steven R. Brandt
Louisiana State University
Baton Rouge, LA, USA
sbrandt@cct.lsu.edu

John Fonner
Texas Advanced Computing Center
The University of Texas at Austin
Austin, TX, USA
jfonner@tacc.utexas.edu

ABSTRACT

The Agave Platform first appeared in 2011 as a pilot project for the iPlant Collaborative [11]. In its first two years, Foundation saw over 40% growth per month, supporting 1000+ clients, 600+ applications, 4 HPC systems at 3 centers across the US. It also gained users outside of plant biology. To better serve the needs of the general open science community, we rewrote Foundation as a scalable, cloud native application and named it the Agave Platform. In this paper we present the Agave Platform, a Science-as-a-Service (SaaS) platform for reproducible science. We provide a brief history and technical overview of the project, and highlight three case studies leveraging the platform to create synergistic value for their users.

CCS CONCEPTS

• **Security and privacy** → *Security services; Software and application security*; • **Applied computing** → *Service-oriented architectures; Event-driven architectures; Enterprise data management; Reference models; Enterprise computing infrastructures; Enterprise interoperability; Enterprise application integration; Physical sciences and engineering; Life and medical sciences*;

KEYWORDS

Agave, Science-as-a-Service, web service, REST, API, cloud, HPC, HTC, container, Docker, Singularity, big data, data science, science gateway, web application, platform, platform-as-a-service, software-as-a-service, paas, saas, data management, job submission, cloud service, devops, microservice

ACM Reference Format:

Rion Dooley, Steven R. Brandt, and John Fonner. 2018. The Agave Platform: An Open, Science-as-a-Service Platform for Digital Science. In *PEARC '18: Practice and Experience in Advanced Research Computing*, July 22–26, 2018, Pittsburgh, PA, USA. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3219104.3219129>

1 INTRODUCTION

Between the time this manuscript is submitted and the time it appears in print, a generation of household devices, automobiles, and wearable technology will drive home the reality that the Internet of Things is really the Internet of Everything.

This paper is authored by an employee(s) of the United States Government and is in the public domain. Non-exclusive copying or redistribution is allowed, provided that the article citation is given and the authors and agency are clearly identified as its source.

PEARC '18, July 22–26, 2018, Pittsburgh, PA, USA
2018. ACM ISBN 978-1-4503-6446-1/18/07.
<https://doi.org/10.1145/3219104.3219129>

Such rapid change is now commonplace in our personal lives, yet within the academic research space, the rate of technological innovation proceeds at a more modest pace. One reason for such disparity is that the sprawling Everything-as-a-Service (XaaS) ecosystems enabling rapid iteration in the commercial space has been absent in the research community. It's easy to sync personal data to a cloud storage account, but when the data size grows into the petabytes, and complex multi-step hybrid analysis is required, the number of XaaS solutions one can offload such work to dwindles. Add to that complex identity federation scenarios, loosely defined payment and chargeback mechanisms, and the need for fine user, group, and role based access control over both data and computation, and the list of available options quickly dwindles again.

These were some of the problems we addressed when we started to develop a platform for the open science community. Rather than focus on how to build bridges between HPC, HTC, big data, and private cloud environments, we focused on bringing them to the web and on providing interfaces and standards that were easily consumable from any language and platform. We focused on being inclusive rather than exclusive and on building a platform that remained approachable regardless of the problem size. We focused on building an open platform to enable reproducible Science-as-a-Service (SaaS) that is now known as the Agave Platform.

2 BACKGROUND

Agave traces its roots back to the iPlant Foundation API (Foundation) [5]. The iPlant Collaborative was a 10 year, \$100M investment in cyberinfrastructure by the NSF BIO Directorate [11]. Foundation was an internally sponsored pilot project seeking to make the iPlant cyberinfrastructure accessible to a broader developer community. Based on feedback collected from developers, researchers, and PIs at over 200 workshops, trainings, and direct engagements spanning the first three years of the project, Foundation was released in 2011. It provided authentication and authorization, job submission, data management, and data transformation capabilities through consistent REST interfaces. In its first two years of operation, Foundation usage grew by 40% month over month, eventually supporting 1000+ clients, 600+ applications, and 4 HPC systems at 3 centers.

During that time, Foundation gained traction outside of the plant biology community. The 1000 Bull study [13], iMicrobe [16], and ProseViz [3] projects all incorporated the Foundation APIs and scaled their computational and data management capacity.

While the initial results were promising, it was clear that additional work needed to be done for the APIs to be adopted by the broader community. Features such as accessing other HPC and HTC

systems, interacting with and sharing data spanning multiple data storage systems, and faster job turnaround were common requests.

In January of 2013, we took the feedback and lessons learned from Foundation and began another round of community engagements and iterative design. During that period we expanded the scope to include multi-tenancy, API management, additional storage and execution environments, first-class support for metadata and resource publishing, and better integration with third party services. We rewrote the entire back end with scalability, performance, and web standards in mind.

When we were done, we decided to re-brand the project with a name that paid homage to the project's origins in the iPlant project and the Southwestern United States. We settled on AGAVE, which is an acronym for "A Grid and Virtualization Environment," and launched in November of 2013. Within the first year, we dropped the acronym and started referring to it as the Agave Platform.

We saw many encouraging things in our initial user engagements. One of the most unexpected results was how far Agave lowered the technical competencies needed to interact with a distributed computational environment. Functionality that, historically, required at least one full time senior developer a year or more to construct could now be done in a matter of days by using Agave's REST APIs, HTML, CSS, and Javascript.

This was particularly relevant in light of a survey conducted by the Science Gateway Community Institute [17] in which more than 26,000 NSF-funded PIs, science gateway developers, and leaders in higher education (i.e., CIOs, CTOs, and others) participated. Over 67% of respondents said they needed help adapting existing technologies to the needs of their gateway. Another 78% of administrative and 66% of technical decision makers said they needed help simply identifying what technologies were available and how they might be leveraged [18].

Agave fills these needs by providing a platform to leverage infrastructure ranging from campus datacenters to commercial clouds to run code, manage data, collaborate meaningfully, and integrate anywhere. Agave's developer tooling allows users to quickly stand up white labeled science gateways in multiple languages and frameworks, launch enhanced Jupyter and RStudio notebook servers locally and in the cloud, and easily integrate new and existing applications in one of the many languages Agave's SDK supports.

3 RELATED WORK

The areas of scientific application development and web-enabled science are extremely broad, with problems and solutions spanning software development from iron to IDE, and user experiences from terminal to tablet. To identify where the Agave Platform sits in the larger ecosystem, we briefly review the high level technology areas tangential to Agave and the use cases it supports.

It should be noted that these classifications are neither exhaustive nor static. "On a long enough time line, the survival rate for everyone drops to zero" [24]. The gateways that last the longest tend to evolve. What starts as a science gateway may also become a public API. Over time those services may grow and expand to power a rich discovery environment, which in turn may shift to operate as a separate science PaaS with independent SaaS offerings. Such

has been the path of HubZero [22], Galaxy [10], Airavata [21], WS-PGrade, UNICORE[29], eScience Central, and Agave. Such change is healthy and necessary, so if some of the projects mentioned below seem misplaced, it is likely because they have simply evolved.

3.1 Science gateways

Science gateways are (usually web-based) applications allowing science and engineering communities to access shared data, software, computing services, instruments, educational materials, and other resources. Projects like iMicrobe, CIPRES [23], OpenNeuro, and the IPT Portal empower their communities to conduct their science without worrying about the complex infrastructure involved.

While the value of science gateways is undeniable, so too is the effort required to build and sustain them. Projects like WS-PGRADE, Galaxy Zooniverse [2], CHAIN-RED [27], and our Agave ToGo project provide frameworks to accelerate the development of new gateways, which can save thousands of person-hours. This frees up brain cycles for other areas like innovation, customizations to the framework, and maintenance.

The National Science Foundation has continued pushing the topic of sustainability to the forefront with the new Cyberinfrastructure for Sustained Scientific Innovation program, bringing sustainability planning to the discussion [9]. One of the ways to offset the costs of developing and maintaining a gateway is to reduce their technical debt through the use of third-party cloud and middleware services.

3.2 Middleware services

Science gateways have historically turned to middleware solutions deployed within academic data centers for higher-level abstractions to low-level hardware and storage infrastructure. The Globus Toolkit [8], UNICORE, and gLite [15] have made giant leaps forward in terms of making HPC accessible outside of the command line. Work by the Open Grid Forum [7] and others to standardize and implement interfaces to common middleware services through GridPort [35], the Vine Toolkit [28], and SAGA [12], made the first web-based science gateways possible.

As the web evolved, an impedance formed between research middleware and the most popular web technologies. This was the problem Foundation was originally created to solve. Well-funded cyberinfrastructure projects in the US like XSEDE, the Open Science Grid, EarthCube, and DataOne, as well as the newly formed Software Institutes [18, 34] do a great job of educating and supporting gateway projects through the integration process, but acknowledge that they are unable to serve all the requests they get from new projects, let alone the needs of projects past their original funding cycle. At a macro level, the fundamental issue is that middleware has evolved slower than the applications wishing to consume it. This is where cloud services come in.

3.3 Cloud Services

As more science moves to the cloud, the proliferation of web-based science has increased. Scalable compute, storage, IAM, and communication services have been commoditized in cloud environments. Persistent, scalable data management is available as a service from dedicated SaaS providers as well as every major cloud provider.

Hundreds of popular scientific codes are available on-demand from vendors in the commercial cloud ecosystems, with the vast majority providing a web-based user interface.

The hype of such cloud services is that the words “service” and “solution” become interchangeable. As the world learned on Feb. 28, 2017, cloud services are scalable and reliable, until they are not [31]. Even when they work as advertised, effort is required to build the last mile in any third-party integration. When the third-party services doubles as a file system, operating system, process manager, etc. the last mile might feel longer. For projects without the desire and/or resources to do the integration needed, leveraging a PaaS can offer the functionality many associate with the adoption of cloud services.

3.4 Science Platform-as-a-Service (PaaS)

An emerging trend in the academic space is the development of science-oriented Platform-as-a-Service solutions. A science PaaS provides the same high level functionality advertised by many cyberinfrastructure projects, but without managing the underlying infrastructure. Instead of providing storage, compute, persistence, IAM, etc., the PaaS ties into one or more underlying services and exposes them through common, standards-based interfaces. In doing this, platforms accelerate the initial development of scientific applications as well as extend their lifetime by shielding them from changes in the underlying research infrastructure. Some examples of science PaaS are SciGap [25], Globus[1], CyberGIS[20], and eScience Central. The Agave Platform falls into this category.

3.5 Data Science Environments

One step above Science PaaS are data science environments (DSE). These primarily web-based environments combine data analytics, visualization, and collaboration tools in a common interface and back them with persistent storage and elastic computational capacity. Commercial DSE offerings integrate with many cloud providers, storage services, and frameworks, e.g. Hadoop, Spark, and Tensorflow. They offer many options for authentication, web, mobile, programmatic interfaces, and data publication. Products in this space are Domino Data Lab, DataScience.com, RapidMiner, Dataiku, MathWorks, KNIME, and SAS. Market trends steer commercial products, so integrations into unique architectures, MPI codes, etc. may not be present. The aforementioned middleware libraries, including the Agave client SDK, make access to traditional academic HPC possible.

In the academic space, projects such as Cyverse, CyberGIS, NanoHub, OOI, etc. provide DSE's for their specific scientific communities. Outside of those domains, notebook servers such as Jupyter, RStudio, and Zeplin dominate. These projects provide open, well-designed user interfaces for interactive data science while also enabling batch, streaming, and asynchronous paradigms. One of the key challenges in using these products versus their commercial counterparts is the need for custom code to integrate with external systems and services.

4 TECHNICAL OVERVIEW

In this section we present an overview of the Agave project. We begin by presenting the architecture and high level operational

(a) Sandbox deployment architecture of a single tenant. (b) Scaled production deployment architecture of one or more tenants.

Figure 1: Staging vs Scaled Production Agave Deployment Architecture

details. Next we detail the Platform Services that provide multi-tenancy, IAM, and the API Management that expose the Science APIs described in the following section. We then move on to discuss the tooling and technology ecosystem surrounding the platform. We conclude by highlighting three projects leveraging Agave to expose synergistic value to their users, and touch on several areas of future work.

4.1 Architecture

Agave's multi-tenant capability comes from its ability to securely serve multiple tenants from a single deployment of the platform. A tenant is a group of users, usually representing a community or organization, who share access to a common configuration of the platform. Within a tenant, IAM, data, services, documentation, and global default settings are configured for all users. Typically, a single Agave tenant will support tens to hundreds of client applications and thousands of users with no additional configuration.

Agave is implemented and run as a collection of microservices deployed as Docker containers. We use a collection of Ansible playbooks we refer to as the Agave Deployer as the basis for our DevOps approach to continuous integration (CI) and continuous deployment (CD). A Jenkins master-worker cluster powers all of our CI tasks and runs the various Deployer playbooks.

The Deployer contains playbooks to deploy and scale tenants from a minimal sandbox installation to a high-availability, shared installation spanning multiple data centers. Figure 1 shows two ends of the spectrum. The architecture in figure 1b on the left shows the simplest, fully functional deployment of the platform. Roughly 50 containers will be started across 4 hosts. One host for platform services, another for Science APIs, another for Science API workers, and another for infrastructure services.

In practice, most tenants do some capacity planning up front, then adjust their static footprint to handle 80-90% of peak throughput at any given moment. Adding additional workers, load balancing the IS, APIM, and Science APIs, clustering the MariaDB database, and sharding the default MongoDB server for better capacity and performance are standard to scale up higher capacity tenants. The architecture in figure 1b shows a production deployment scaled in this manner across 20+ hosts. With this architecture, upwards of 200 containers would be running at a given time. Given sufficient networking, such a deployment would be capable of moving moving upwards of a Petabyte of data, managing 100s of thousand of jobs, and serving 10s of millions of requests per day across one

Figure 2: Diagram of the standard components deployed in a vanilla Agave tenant server.

or more tenants. It is important to note that while most of these tasks can be automated by the Deployer, local database and system administrators should be included in the process.

4.2 Platform services

Agave's Platform Services handle everything associated with deploying, testing, configuring, and administering a tenant, its APIs, and its users. Earlier we mentioned that a vanilla deployment of Agave results in a base footprint of 20 containers supporting platform services. Figure 2 illustrates the services those containers represent. Additional support services are also available to handle user signup and password reset, changelog parsing, json parsing and filtering, slack integration, webhook logging, tenant discovery, and cumulative usage reporting.

4.2.1 Identity Management. Agave uses the WSO2 Identity Server (IS) to handle identity management and provide a proven, standards-compliant, enterprise ready OAuth2 server. By default, a lightweight IS bundled with the WSO2 API Manager (APIM) is used. This is sufficient for most use cases. For high availability deployments and in situations where tenants require advanced identity and/or federation scenarios including OpenID Connect (OIDC), SAML, and social login, the IS is deployed as a standalone component.

Both the bundled and standalone IS support a wide range of popular identity providers (IdP) out of the box including LDAP, Active Directory, OIDC, and custom database schema. Basic attribute mappings as well as custom groups and roles can be mapped from an IdP to Agave user roles as needed. This is true whether Agave is given write access to the IdP or configured as a read-only client. A custom REST API is also included to enable basic search and, when enabled, CRUD operations on user profiles.

4.2.2 API Management. Agave uses the APIM as the default API Management service and API Gateway. This brings security, rate limiting, analytics, access control, and full OAuth2 protection to the Science APIs as well as any other custom API added by stakeholders or users. When combined with APIM's data mediation features, organizations have a clear path to modernize their legacy infrastructure by publishing their existing SOAP services, static data sources, and other data sources as RESTful JSON APIs without altering the original code base.

4.2.3 Admin APIs. Every tenant has a set of Admin APIs to allow programmatic access to create and manage roles, service accounts, API clients, and register and version APIs. Through these

Figure 3: Architecture of Agave's optional FaaS solution, Abaco.

Admin APIs, tenant owners can fully configure and manage their tenant from the Agave tooling, or any client application that can make HTTP calls.

4.2.4 Functions-as-a-Service (FaaS). Finally, tenants with FaaS use cases can add Agave's serverless component, Abaco, to their deployment architecture. Abaco is an open source, serverless solution that uses Linux containers and the Actor model of concurrent computing to implement FaaS [33]. Abaco was originally developed in 2015 as an internal proof of concept by members of the Agave team. In 2017, it spun off as an independently funded project with an award from NSF [14]. The architecture is shown in Figure 3. When deployed, Abaco will appear as a custom API in the platform. New functions are implemented as Docker images and registered by making a POST request to the API. Abaco handles image fetching, caching, container execution, monitoring, data management, and provenance internally. A permission model consistent with the Science APIs enables sharing and publishing of individual functions. Actors enable native interactions with other Agave services by injecting delegated bearer tokens into every container and creating events available for subscription through Agave's Notifications API.

4.3 Science APIs

At the heart of Agave is a battle-tested collection of REST APIs which underpin every other part of the Platform and provide the SaaS functionality for which the platform is known. A key feature of the Science APIs is their understanding and implicit handling of the concerns common to computational science. Things like automated job output archiving, dynamic subscriptions to events, automated metadata validation, deep provenance tracking, and cascading permission grants allow users to skip much of the plumbing otherwise required to efficiently implement features like searchable job histories, push notifications, attribution statistics, and reliable data replication in their applications.

The Science APIs are implemented as a collection of message-driven, asynchronous micro services. Each micro service is built, tested, tagged and published as a publicly available image in the Docker Public Registry by a single, vanilla Jenkins server in response to notifications from the API's git repository. By shipping each microservice as a separate container, the platform can better leverage contributions from the open source community, play to the strengths of internal developers with different skill sets, manage multiple API versions, and elastically scale independent containers to meet capacity and hybrid deployment requirements.

Every tenant comes online with the Science APIs preregistered. When a new client application is created, it is automatically subscribed to all of the Science APIs so applications can immediately use any services they need without first learning what is available.

By delegating the heavy lifting and low-level, tedious tasks to Agave, developers can achieve tremendous speedups when bootstrapping new gateways and scaling production ones. Last month alone (Feb 2018), users moved over a million files, created and shared nearly 100 runnable codes, and managed the consumption of over 18.5 million compute hours spanning 80 different user-defined systems. All that without ever asking an end user to login to a terminal, install a piece of software, or learn a new programming language. The remainder of this section lists the available Science APIs in use today. The remainder of this section describes each of the APIs that comprise the Agave Science APIs

4.3.1 Apps API. An “app” in Agave represents the combination of an executable code and an execution system on which it should run. Apps are defined using JSON. The schema only requires a name, version, execution system, and location where Agave should fetch the app assets. Additional attributes are available to define everything from ontological terms to rules for server-side validation of input files and parameters. Users may define their own apps and share them with others at their discretion. Apps may also be published by administrators for public consumption.

The Apps service is the central registry for all Agave apps. The Apps service provides permissions, validation, archiving, and revision information about each app in addition to common search and listing capabilities.

4.3.2 Events API. Agave is an event-driven platform. The Events API allows users to discover the events for each Science API and define custom events to be thrown in user-defined situations.

4.3.3 Files API. The Agave Files service allows users to manage data across multiple storage systems, services, and protocols using a single, consistent interface. It supports traditional file operations such as directory listing, renaming, copying, deleting, and upload/download that are found in most file management services. It also supports file importing from arbitrary locations, metadata assignment, in transit data transformation, logging, provenance, and a full access control layer allowing users to keep data private, share it with colleagues, or make it publicly available without sharing credentials.

4.3.4 Groups API. The Groups API allows users to define and manage groups of users. Groups in Agave are treated like users, with cascading permissions from group members resolved and applied prior to assigning individual user permissions.

4.3.5 Jobs API. The Jobs API is a basic execution service that allows users to run apps registered with Agave’s Apps API across multiple, distributed, heterogeneous systems through a common REST interface. The service manages all aspects of the job lifecycle from data staging, job submission, monitoring, output archiving, event logging, sharing, and notifications. The Jobs API also provides a persistent reference to the job’s output data and a mechanism for sharing all aspects of the job with others.

4.3.6 Metadata API. The Metadata API allows users to manage metadata and associate it with Agave entities via associated UUIDs. It is backed by MongoDB and supports a JSON schema to define and validate structured JSON metadata; it also accepts any valid JSON-formatted metadata or plain text string when no schema is specified. As with other Agave APIs, a full access control layer is available, enabling users to keep metadata private or share it with colleagues.

4.3.7 Monitors API. The Monitors API provides a familiar paradigm for monitoring the usability and accessibility of storage and execution systems registered with Agave. Similar to services like Pingdom, Pagerduty, and WebCron, the Monitors API allows users to schedule regular health checks on a registered system. Unlike standard uptime services, Agave will check that the system is responsive and accessible by checking availability (ping), accessibility (authentication), and functionality (listing or echo). Each check result is persisted and the history and is queryable through the API. As with all resources in the Agave Platform, a full event model is available so the user can subscribe to the events they care about—such as failed checks, restored system availability, and system disablement. An option to automatically enabling or disabling the system based upon the results of a check is also available.

4.3.8 Notifications API. The Notifications API is the primary mechanism by which consumers create and manage event subscriptions within a tenant. A typical use case is a user subscribing for a slack notification when her job completes, a text message when a permission changes on a shared file, or a webhook callback when an app definition is updated.

Notifications can be single use, or persistent and have a per-use retry policy built in. In situations where notifications cannot be delivered, a dead letter queue is available so users can query for messages they may have missed.

4.3.9 PostIts API. The PostIts API is a URL shortening service similar to bit.ly and goo.gl. It allows users to create pre-authenticated, disposable URLs to any resource to which they have access within the platform. The lifetime, number of uses, request method, and the URL can be configured. PostIts can be manually refreshed or expired at any time. As with all Science API resources, a full set of events is available to track usage and integrate the lifecycle of a PostIt into external applications.

The most common use of PostIts is to create single use, short-lived URLs to protected files and metadata that can be sent to colleagues or provided to a third-party service. For example, using the PostIts service, one could share the output(s) of an experimental run, distribute materials for a class, or serve up assets for a website like, Agave ToGo.

Other use cases for PostIts include creating “drop” folders to which anyone with the link can upload data, allowing a job to be reproducibly rerun for peer review, publishing metadata for public consumption, publishing a canonical reference to a user profile, and passing short-lived URL into a job to enable pre-authenticated API access from within a running job. The possibilities go on and on.

4.3.10 Profiles API. The Profiles API is a hosted identity service providing organizations a way to create and manage user accounts within their Agave tenant. The service is backed by an OpenLDAP

server, usually clustered for high availability. Tenants configured for Agave to use their organization's existing, on-premise identity solution have the option of allowing the Profiles API to create, edit, and delete user accounts in addition to providing read access.

As with many commercial identity providers, the Profiles service provides the means to interact with what can often be complicated backend identity solutions in a secure, technology agnostic way. This shields the details of the identity solution from consumers while providing them a secure, convenient way to obtain the basic information needed for meaningful collaboration.

4.3.11 Realtime API. The Realtime API provides websocket connections to Agave's event stream for a user, resource, or tenant. No buffering or retry efforts are made if the consumer disappears. Streams are, however, persistent, so a dropped client can reconnect to a predefined stream at any time.

4.3.12 Stats API. The Stats API provides statistical usage information about API resources, clients, and tenants. Responses from the Stats API can easily be fed into the Metadata API as a way to create and save daily usage reports, or given as the input to a popular graphing package like Google Charts, Flot Charts, Plotly, or Matplotlib to build a custom dashboard.

4.3.13 Systems API. A "system" in Agave represents a physical, virtual, digital resource that stores data and/or runs code. Systems are identified and referenced in Agave by a unique ID unrelated to their ip address or hostname. Because of this, a single physical system may be registered multiple times. This allows multiple users to configure and use a single shared resource, such as an HPC cluster, in the way(s) that best meet their individual needs. Systems may also have a virtual root and/or virtual home directory defined. If defined, Agave will resolve all requests for absolute paths relative to the virtual root directory, and resolve all requests for relative paths relative to the virtual home directory. This allows multiple users to register a single physical system or cloud service as multiple Agave systems, each with a root directory pointing to a different folder, and maintain total data isolation within the platform.

Systems come in two flavors: storage and execution. Storage systems are only used for storing and interacting with data. Execution systems are used for running apps (a.k.a. jobs or batch jobs) as well as storing and interacting with data. All systems require some basic data connectivity information including hostname, port, account, authentication details, and protocol. Agave currently supports FTP, SFTP, GridFTP, S3, and iRODS data protocols. Swift, WebDav, Dropbox, and Box are currently in development.

Execution systems require additional login information to connect to the remote system and launch an app. SSH, GSI, OpenSSH, and Local access methods are supported as well as tunneling through a proxy server. Once connected, Agave needs to know how to invoke the app code. The type of execution dictates which, if any, scheduler should be used. LSF, LoadLeveler, PBS, Open Grid Engine, CondorHT, Torque, Moab, SLURM, and Condor are all supported in additional standard forked execution.

The Systems API gives users the ability to create and discover storage and execution systems for use in the rest of the Science APIs. Newly registered systems are private to the user by default. Systems can be shared with one or more users through role assignment and

published into the public space by administrators. Larger tenants commonly publish one or more storage systems to provide free cloud storage and execution services to all their users.

4.3.14 Tags API. The Tags API provides free form tagging of any resolvable resource in the platform. Tags work like metadata, but unlike the Metadata API, tag names must be unique for a given user or group.

4.3.15 Tenants API. The Tenants API is a readonly information service providing a single discovery point to find known tenants and their configuration information. For on-premise deployments, the local tenant service is where white-labeled Agave tooling for individual tenants look to discover URLs for registration, authentication, and Science APIs.

4.3.16 Transfers API. The Transfers API picks up where the Files API leaves off by providing managed, multi protocol file transfers. Data may be moved between any Agave systems and pushed/pulled from URI with supported schemata. The Transfer API manages all data movement in the platform, so in addition to providing a REST interface to schedule and query transfers, it is also the interface to query transfer history of job inputs and outputs, published app assets, and PostIt redemptions.

4.3.17 Transforms API. The Transforms API handles transformation of data between known formats. The supported file types are configured on a tenant by tenant basis and can be discovered by querying the transforms service. Transformations may be made as part of a file transfer, an asynchronous request on existing data at rest, or synchronously as part of a file download. A full event model is available to track the transformation progress.

4.3.18 UUID API. The UUID API resolves the type and representation of one or more Agave UUIDs. This is helpful when you need to expand the hypermedia response of a resource, get the URL corresponding to a UUID, or fetch the representations of multiple resources in a single request.

4.4 Agave Tooling

Agave provides a rich set of tooling to help onboard users and make integration easier for new and existing applications. The command line interface (CLI) provides a convenient way to leverage Agave from existing scripts and terminal environments. Client software development kits (SDK) in Java, PHP, Python, R, AngularJS, and Node provide light wrappers around Agave's REST API and enable programmatic integration into third-party applications.

4.4.1 Agave ToGo. Agave ToGo [6] is an open source, static web application written in AngularJS and serves as the reference science gateway for the Agave Platform. ToGo was designed to be both functional and informative. New projects can fork ToGo into their own Github repository and have a fully functional science gateway hosted directly out of Github Pages. Existing projects can look at ToGo's source to find clean implementations of common science gateway use cases using nothing more than Agave APIs.

4.4.2 Jupyter and RStudio. Agave also provides deep integrations into both Jupyter and RStudio notebook servers by providing Docker images of each server with preconfigured authentication,

token refresh, quickstart tutorials, and pre-installed Agave SDK for each kernel/runtime. These images are the default environment for all of our training events, and provide convenient ways to run code, manage data, collaborate meaningfully, and integrate anywhere, all from within a local notebook environment.

5 CASE STUDIES

The Texas Advanced Computing Center operates an enterprise, hosted version of the Agave Platform. At the time of this writing, it powers 10 production tenants for organizations across the US and Canada. Here we briefly highlight three of those tenants and their driving use cases for the platform.

5.1 Cyverse

CyVerse is the follow-on to the iPlant project, a 10 year cyberinfrastructure project funded by the National Science Foundation's Directorate for Biological Sciences. It extends the scope of the project to provide large-scale computational infrastructure to all life sciences with the goal of enabling new discoveries through data driven science.

As the project from which Agave was birthed, CyVerse has a significant investment in the Platform and its technology ecosystem. Agave integrates with CyVerse's existing Central Authentication Service (CAS) to provide a seamless experience from the web to the desktop to the data center. Agave's APIM publishes CyVerse's internal REST API alongside the Agave's Science APIs with consistent authentication, metering, etc, thereby enabling programmatic access to all aspects of the CI from their web-based Discovery Environment[19], to their petascale iRODS Data Store, to their public OpenStack cloud and its custom management interface: Atmosphere[32]. The CyVerse SDK is a branded version of the Agave CLI with additional configurations and templates designed to onboard new users the CyVerse resources. Agave ToGo serves as an alternative UI to interact with the HPC infrastructure powering CyVerse and quickstart for developers seeking to build internal and 3rd party gateways using their CI.

5.2 DesignSafe

Description: "DesignSafe is the web-based research platform of the NHERI Network that provides the computational tools needed to manage, analyze, and understand critical data for natural hazards research" [26]. DesignSafe chose Agave to power their Discovery Workspace due to its ability to simplify the use of complicated computational infrastructure. The REST interfaces to the APIM and Science API provided them with straightforward, language agnostic methods to integrate existing data and content and accelerate the development of web interfaces. Agave's ability to build, run, and track loosely coupled workflows, expose a deep, event-driven provenance model, and create reproducible experiments meant they could save significant time and resources versus integrating a third-party solution or rolling their own.

Early returns from the project were very promising. Because of the time savings they achieved by leveraging the platform, they were able to ship the first release of their Discovery Workspace and transition completely off of the prior NEES infrastructure in under 6 months [4]. By exposing the Science APIs and registering

several boutique APIs, they have been able to expose their entire infrastructure through a REST API. This played a major part in their ability to release a fully functional developer portal just 3 months later, thereby allowing users to build and publish their own Apps, discover new tooling, and create project driven studies with select groups of collaborators.

Finally, one of the early requests from DesignSafe users was for a notebook environment that was tied into the project fabric. Agave's Jupyter Hub integration provided a solution by giving them a scalable, managed Jupyter Hub cluster with persistent cloud storage, integrated login via OAuth2 server, preconfigured user environments, and custom Jupyter Kernels with the Agave SDK already installed and initialized for the user's tenant.

5.3 SD2E

The Synergistic Discovery and Design Environment (SD2E) is a web-based analysis platform for the DARPA SD2 program that enables collaborative analysis, data sharing, and application development backed by national-scale computing, analytics, and storage capacity at the Texas Advanced Computing Center (TACC). Its goal is to develop data-driven methods to accelerate scientific discovery and robust design in domains such as synthetic biology, neuro-computation, and polymer chemistry that lack complete high-fidelity models[30].

SD2E heavily relies upon Agave for authentication and integration with an external identity provider. Interactive, reactive, and near realtime compute scenarios are enabled across the project through integration with the Science APIs and Abaco. Periodic data synchronization and streaming data ingress are achieved through a combination of dedicated hardware, subscription to one or more Agave webhook notifications, and use of the Agave SDK and CLI. Jupyter notebooks are also extensively used within the project for interactive analysis and presentations.

6 FUTURE WORK

Agave will continue to evolve to better serve the existing user community while ensuring it remains an attractive choice for innovators. Updates to platform services will introduce support for OpenID Connect, SCIM, and group management. The Science APIs will receive updates across the board to reduce the resource utilization of each service, improve data management performance, and introduce retry policies into all data operations. Additional support for Box, Dropbox, and several popular object storage services is also currently underway. Server-side implementations of Agave ToGo in Python, PHP, and Node are in progress with contributions from external projects leveraging Agave making up the bulk of the code. A new AP (Agave Platform) training series is also under development and will roll out over the coming year.

7 CONCLUDING REMARKS

The Agave Platform is an open, SaaS platform for reproducible science. Agave uses standards-based technologies and community-promoted best practices to enable users to run code, manage data, collaborate meaningfully, and integrate anywhere. Since its first launch in 2011 as a proof of concept for what is now the CyVerse project, Agave has grown to power dozens of production science

gateways while extending and enhancing the functionality of hundred more applications. During that time, a rich technology ecosystem has developed around the platform to provide client SDK in multiple languages, a CLI, reference web applications, and integrations with many of today's most popular web frameworks and cloud services. It has never been easier, faster, or cheaper to bring HPC to the web than it is today with the Agave Platform.

8 ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation Plant Cyberinfrastructure Program (DBI-0735191), the National Science Foundation Plant Genome Research Program (IOS-1237931 and IOS-1237931), the National Science Foundation Division of Biological Infrastructure (DBI-1262414), the National Science Foundation Division of Advanced Cyberinfrastructure (1127210), and the National Institute of Allergy and Infectious Diseases (1R01A1097403).

REFERENCES

- [1] Rachana Ananthakrishnan, Kyle Chard, Ian Foster, and Steven Tuecke. 2015. Globus platform-as-a-service for collaborative science applications: GLOBUS PLATFORM-AS-A-SERVICE FOR COLLABORATIVE SCIENCE APPLICATIONS. *Concurrency and Computation: Practice and Experience* 27, 2 (Feb. 2015), 290–305. <https://doi.org/10.1002/cpe.3262>
- [2] K. D. Borne and Zooniverse Team. 2011. The Zooniverse: A Framework for Knowledge Discovery from Citizen Science Data. *AGU Fall Meeting Abstracts* (Dec. 2011), ED23C–0650.
- [3] Tanya Clement. 2012. Methodologies in the Digital Humanities for Analyzing Aural Patterns in Texts. In *Proceedings of the 2012 iConference (iConference '12)*. ACM, New York, NY, USA, 287–293. <https://doi.org/10.1145/2132176.2132213>
- [4] Josue Balandrano Coronel and Stephen Mock. 2017. DesignSafe: Using Elasticsearch to Share and Search Data on a Science Web Portal. In *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact (PEARC17)*. ACM, New York, NY, USA, Article 25, 3 pages. <https://doi.org/10.1145/3093338.3093386>
- [5] Rion Dooley and et. al. 2012. Software-as-a-Service: The iPlant Foundation API. 5th IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS).
- [6] Rion Dooley, Joe Stubbs, and Manuel Rojas. 2017. Agave ToGo Micro Sites: Phenomenal Gateway Power, Itty Bitty Living Space. (2017). <https://doi.org/10.6084/m9.figshare.4494479.v2>
- [7] Open Grid Forum. 2018. Open Grid Forum. (2018). <https://www.ogf.org/ogf/doku.php/about/introduction>
- [8] Ian Foster and Carl Kesselman (Eds.). 1999. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- [9] National Science Foundation. 2018. Cyberinfrastructure for Sustained Scientific Innovation (CSSI) - Data and Software. (2018). <https://www.nsf.gov/pubs/2018/nsf18531/nsf18531.htm>
- [10] Jeremy Goecks, Anton Nekrutenko, and James Taylor. 2010. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology* 11 (Aug. 2010), R86. <https://doi.org/10.1186/gb-2010-11-8-r86>
- [11] Stephen Goff and et al. 2011. The iPlant Collaborative: Cyberinfrastructure for Plant Biology. *Frontiers in Plant Science* 2 (2011), 34. <https://doi.org/10.3389/fpls.2011.00034>
- [12] Tom Goodale, Shantenu Jha, Hartmut Kaiser, Thilo Kielmann, Pascal Kleijer, Gregor von Laszewski, Craig Lee, Andre Merzky, Hrabri Rajic, and John Shalf. 2006. SAGA: A Simple API for Grid Applications. High-level application programming on the Grid. *Computational Methods in Science and Technology* 12, 1 (2006), 7–20. <https://doi.org/10.12921/cmst.2006.12.01.07-20>
- [13] Benjamin Hayes and et al. 2013. The 1000 bull genomes project - Toward genomic selection from whole genome sequence data in dairy and beef cattle. (2013).
- [14] Joe Stubbs and Matthew Vaughn. 2017. NSF Award Search: Award#1740288 - S12-SSE: Abaco - Flexible, Scalable, and Usable Functions-As-A-Service via the Actor Model. (Aug. 2017). https://www.nsf.gov/awardsearch/showAward?AWD_ID=1740288&HistoricalAwards=false
- [15] Peter Z Kunszt, M Sgaravatto, A Aimar, F Hemmer, D Groep, M Livny, A Di Meglio, F Prelz, A Edlund, S Fisher, F Pacini, M Barroso, L Guy, E Laure, P Buncic, O Mulmo, and S Beco. 2004. Middleware for the next generation Grid infrastructure. (2004). <https://doi.org/10.5170/CERN-2005-002.826>
- [16] Hurwitz Lab. 2018. iMicrobe. (2018). <https://www.imicrobe.us/>
- [17] Katherine A. Lawrence and Nancy Wilkins-Diehr. 2012. Roadmaps, Not Blueprints: Paving the Way to Science Gateway Success. In *Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the eXtreme to the Campus and Beyond (XSEDE '12)*. ACM, New York, NY, USA, 40:1–40:8. <https://doi.org/10.1145/2335755.2335837>
- [18] Katherine A. Lawrence, Michael Zentner, Nancy Wilkins-Diehr, Julie A. Wernert, Marlon Pierce, Suresh Marru, and Scott Michael. 2015. Science gateways today and tomorrow: positive perspectives of nearly 5000 members of the research community. *Concurrency and Computation: Practice and Experience* 27, 16 (Nov. 2015), 4252–4268. <https://doi.org/10.1002/cpe.3526>
- [19] Andrew Lenards, Nirav Merchant, and Dan Stanzione. 2011. Building an Environment to Facilitate Discoveries for Plant Sciences. In *Proceedings of the 2011 ACM Workshop on Gateway Computing Environments (GCE '11)*. ACM, New York, NY, USA, 51–58. <https://doi.org/10.1145/2110486.2110494>
- [20] Yan Liu, Anand Padmanabhan, and Shaowen Wang. 2015. CyberGIS Gateway for enabling data-rich geospatial research and education. *Concurrency and Computation: Practice and Experience* 27, 2 (Feb. 2015), 395–407. <https://doi.org/10.1002/cpe.3256>
- [21] Suresh Marru, Lahiru Gunathilake, Chathura Herath, Patanachai Tangchaisin, Marlon Pierce, Chris Mattmann, Raminder Singh, Thilina Gunarathne, Eran Chinthaka, Ross Gardler, Aleksander Slominski, Ate Douma, Srinath Perera, and Sanjiva Weerawarana. 2011. Apache Airavata: A Framework for Distributed Applications and Computational Workflows. In *Proceedings of the 2011 ACM Workshop on Gateway Computing Environments (GCE '11)*. ACM, New York, NY, USA, 21–28. <https://doi.org/10.1145/2110486.2110490>
- [22] M. McLennan and R. Kennell. 2010. HUBzero: A Platform for Dissemination and Collaboration in Computational Science and Engineering. *Computing in Science Engineering* 12, 2 (March 2010), 48–53. <https://doi.org/10.1109/MCSE.2010.41>
- [23] M. A. Miller, W. Pfeiffer, and T. Schwartz. 2010. Creating the CIPRES Science Gateway for inference of large phylogenetic trees. In *2010 Gateway Computing Environments Workshop (GCE)*. 1–8. <https://doi.org/10.1109/GCE.2010.5676129>
- [24] Chuck Palahniuk. 2005. *Fight Club: a novel*. Norton, New York. OCLC: 61711337.
- [25] M. Pierce, S. Marru, B. Demeler, R. Singh, and G. Gorbet. 2014. The Apache Airavata Application Programming Interface: Overview and Evaluation with the UltraScan Science Gateway. In *2014 9th Gateway Computing Environments Workshop*. 25–29. <https://doi.org/10.1109/GCE.2014.15>
- [26] Ellen M. Rathje and et. al. 2017. DesignSafe: New Cyberinfrastructure for Natural Hazards Engineering. *Natural Hazards Review* 18, 3 (08 2017), 06017001. [https://doi.org/10.1061/\(ASCE\)NH.1527-6996.0000246](https://doi.org/10.1061/(ASCE)NH.1527-6996.0000246)
- [27] F Ruggieri, G Andronico, R Barbera, L Matyska, R Mayo Garcia, and O Prnjat. 2013. Introduction to the CHAIN-REDS Project: objectives and achievements. In *Proceedings of e-Infrastructures for e-Sciences 2013*. Beijing, China, 10. <https://pos.sissa.it/199/001/pdf>
- [28] Michael Russell, Piotr Dziubecki, Piotr Grabowski, Michał Krysiński, Tomasz Kuczyński, Dawid Szejnfeld, Dominik Tarnawczyk, Gosia Wolniewicz, and Jarosław Nabrzyski. 2007. The Vine Toolkit: A Java Framework for Developing Grid Applications. In *Parallel Processing and Applied Mathematics (Lecture Notes in Computer Science)*. Springer, Berlin, Heidelberg, 331–340. https://doi.org/10.1007/978-3-540-68111-3_35
- [29] Schuller, B. Rybicki, J., Petrova-ElSayed, M., and Benedyczak, K. 2016. UNICORE 7 - Middleware Services for Distributed and Federated Computing. In *Proceedings of the International Conference on High Performance Computing & Simulation (HPCS)*, HPCS2016. Innsbruck, Austria. <https://doi.org/2128/12214>
- [30] SD2E. 2018. Synergistic Discovery and Design Environment (SD2E). (2018). <https://sd2e.org/>
- [31] Amazon Web Services. 2018. Summary of the Amazon S3 Service Disruption in the Northern Virginia (US-EAST-1) Region. (2018). <https://aws.amazon.com/message/41926/>
- [32] Edwin Skidmore, Seung-jin Kim, Sangeeta Kuchimanchi, Sriramu Singaram, Nirav Merchant, and Dan Stanzione. 2011. iPlant Atmosphere: A Gateway to Cloud Infrastructure for the Plant Sciences. In *Proceedings of the 2011 ACM Workshop on Gateway Computing Environments (GCE '11)*. ACM, New York, NY, USA, 59–64. <https://doi.org/10.1145/2110486.2110495>
- [33] Joe Stubbs, Rion Dooley, and Matthew Vaughn. 2017. Containers-as-a-service via the Actor Model. In *Proceedings of The 11th Gateway Computing Environments Conference*. San Diego, CA. <https://doi.org/10.6084/m9.figshare.4490747.v2>
- [34] T. Daniel Crawford and et. al. 2018. The Molecular Science Institute. (2018). <http://molssi.org/>
- [35] M. Thomas, S. Mock, M. Dahan, K. Mueller, D. Sutton, and J. R. Boisseau. 2001. The GridPort toolkit: a system for building Grid portals. In *Proceedings 10th IEEE International Symposium on High Performance Distributed Computing*. 216–227. <https://doi.org/10.1109/HPDC.2001.945191>