Science Gateway Patterns and Practices: Experiences Deploying Web Portals for Science at the NERSC Supercomputing Center

Shreyas Cholia*, Annette Greiner and Rollin Thomas
Lawrence Berkeley National Laboratory
*Corresponding author address: 1 Cyclotron Road MS 943-2239, Lawrence Berkeley National Laboratory, Berkeley, CA, 94609, USA; email: scholia@lbl.gov

Abstract: *In this work we catalog patterns, practices and trends we have seen from our experiences deploying science gateways at the NERSC supercomputing center. We cover the following topics: Sharing Data Over The Web, Web Frameworks for Science, Web IDEs and Interactive HPC, HTTP APIs, Federated Identity and Single Sign-On, Edge Services, Data Transfer Services, Cloud Hosted Portals, and Containers. This is our attempt to share what we have learned with the community, and to identify key aspects of science gateway deployment and development.*

1. Introduction

Science gateways that enable web access to high performance computing and data resources have played a key role in enabling scientific discovery at the National Energy Research Scientific Computing Center (NERSC). NERSC is the primary scientific computing facility for U.S. Department of Energy Office of Science researchers. Science gateways provide an abstraction layer for users that don't need or want to deal with backend middleware, UNIX command lines and batch scripts. Web interfaces create science-centric views to data and enable new discoveries and insights through collaborative tools and rich interfaces. Our work over the last few years has focused on building and deploying science gateways and related infrastructure that can interact with scientific data and computation at NERSC.

Through our experiences, we have learned a number of useful lessons, developed several best practices and identified patterns that seem to be repeated in many of the applications that we have worked on. This paper is an attempt to enumerate and describe some of the trends, patterns and practices that we have come across so that others in the community may benefit from our experience.

Proceedings of Gateways 2017, University of Michigan, Ann Arbor, MI; October 23-25, 2017.
https://gateways2017.figshare.com/.

2. Patterns and Practices

2.1 Sharing Data Over The Web

For a significant number of our users, simple web access to data stored at NERSC is a huge win. Since science is now a collaborative effort, involving large teams of people, the ability to simply share data easily with collaborators, as well as with the broader community becomes critical. To ensure a broad reach, it is also important to do this without requiring special software on part of the end user.

At NERSC we offer scientific users the ability to make files easily available over HTTP by simply placing them in a special directory in their project space. Files placed in this directory are automatically visible over the web at a project URL - this is the simplest way to create web access to data with no administrator intervention, but it enables a wide range of scientific sharing and collaboration. For example, the ATLAS experiment at CERN takes advantage of this simple but scalable method of data sharing.

For public data, we typically mount the file-system in read-only mode on the web server to prevent unauthorized writing to the files.

With the advent of rich JavaScript and Static HTML sites, this model also allows for more sophisticated sites built using the JAM (JavaScript, APIs, and Markup) Stack [1] since the web-enabled directories can serve up HTML and JavaScript as well.

2.2 Web Frameworks for Science

To truly leverage the power of the web, we expect science gateways to go provide more advanced tools that go beyond static data sharing. More complex web portals often require the use of web frameworks that provide full-stack web application functionality. A number of web applications that we see at NERSC are built using a variant of the Model-View-Controller pattern, and make use of common open-source frameworks like Django or Flask (for Python), Ruby on Rails (Ruby) or Angular or Ember (JavaScript). These frameworks provide a separation of concerns between the data store (typically called the "model"), the front end (the "view") and the business logic (typically called the "controller").

Beyond the MVC model, we also see higher-level frameworks that also provide tools and functionality for developing scientific applications and integration with HPC. For instance, tools like Galaxy [2] allow users to build workflows into their scientific applications. HubZero [3] enables full-featured scientific portals, with hooks to run jobs or manage data on backend resources.

These frameworks can be hosted in a number of different ways – we describe some of the deployment models later in sections 2.6, 2.8 and 2.9 (edge service nodes, cloud deployments and containers)

2.3 Web IDEs and Interactive HPC

An interesting and increasingly popular method to interact with scientific resources is through interactive web consoles and notebooks. These give users the power of a full programming language environment with the convenience and usability of the web.

In particular, the Jupyter platform has been extremely popular as a means for exposing scientific computing through web-based notebooks. These notebooks combine narrative inline documentation, live code and interactive visualizations in a single interface and can make for a very powerful tool to enable iterative human-in-the-loop analysis, as well as to serve as a live document of the scientific process. Originally built around Python, Jupyter now supports over 100 different language kernels [4].

At NERSC we have deployed the Jupyterhub platform to enable a multi-user notebook environment, with direct access to the underlying HPC system, compute jobs and large datasets stored on NERSC filesystems [5]. Similarly, we have also deployed R-Studio - a full IDE for the R programming language that can give users the power of a programming environment directly over the web, with access to the HPC environment. This means that users no longer need to think about sub-setting and downloading data and pulling it down on their local machines. Instead, the analysis and programming tools operate on the data directly at the HPC center over the web.

2.4 HTTP APIs

The past decade has seen rapid growth in the web services model, where web applications interact with backend services through a programmatic interface delivered over HTTP. REST [6] and other similar models for exposing data over a programmatic API have become ubiquitous. We also see this trend in Science Gateway services, where the gateway front end communicates with a REST API layer that provides most of the core functionality for interfacing with scientific data and compute resources.

Functionality is exposed via HTTP URI endpoints that encapsulate resources in conjunction with HTTP verbs (GET, PUT, POST, DELETE, HEAD etc.) and request parameters, with results being returned in a structured data object such as JSON. This makes it possible to capture common pieces of functionality used by multiple gateways under a single programmatic API.

At NERSC we have deployed the NEWT [7] service to provide a REST API for HPC. NEWT enables key features like file transfer, querying of data, scientific computations, job workflow management, user information through a common web API. We also see this pattern in similar tools at other sites, such as the Agave [8] platform and the Apache Airavata [9] framework.

Web APIs facilitate another pattern that we often find helpful for making large datasets available. That is, they allow for downloading or working with subsets of the full dataset. In the context of shared datasets for download, one can allow the user to specify a specific slice through an HDF5 file using tools like PyDAP [10], for example. In a web application, one can enable user interactions that work with relatively small subsets of the data at a time or that leverage HPC resources

to operate on larger data chunks. Rather than requiring the user to download a very large file over HTTP and attempt to manipulate it locally, one can operate on a subset on the HPC side and send only the result to the client. This improves the perceived performance of the application and the network.

The Advanced Light Source SPOT portal [17] illustrates the power of APIs as a means to interact with large scientific resources where the science gateway communicates with the a common REST API to submit jobs, browse datasets and visualize results in real-time.

2.5 Federated Identity and Single Sign-On

Single Sign-On in conjunction with Federated Identity implement common authentication and authorization services for science gateways, especially in the academic and governmental research space where users typically belong to a "home" institution like a university or national laboratory. In this context, it makes a lot of sense for the science gateway to delegate authentication back to the home institution and to simply consume the identity, along with appropriate authorization privileges. It also means that the science gateway no longer needs to implement its own custom authentication or store sensitive information.

The Shibboleth framework and InCommon federation [11] have greatly simplified this process, as most major universities and national laboratories are part of this federation. Tools like CILogon [12] facilitate the use of a common X509 credential, using Shibboleth as the underlying framework that can be used for accessing backend resources.

Single sign-on is not just limited to academic federations. We also see commercial single-sign-on services like Janrain and Auth0 provide mechanisms for users to login via external identities (such as Google or Facebook accounts). Protocols like OAuth and OpenID Connect have become commonplace, providing a secure underlying mechanism for facilitation and delegation of authentication tokens.

2.6 Edge Services

In the context of the HPC center, an edge service is a component, which is exposed to the public Internet that acts as a gateway to the internal platform (i.e., high-performance computing systems and data resources). The web portals we deploy are common examples of these services, but the portal ecosystem often required interactions with other edge service systems such as database services, message queues, continuous integration services, data transfer nodes, web-services/APIs etc. The concept of the edge service node becomes important in the deployment and planning of HPC centers and network configurations that wish to deploy science gateways.

Edge services often live in what is referred to as the "Science DMZ" [13] and act as the bridge between external facing services and internal resources. These nodes may have some access to the internal network, but they only expose resources through well-defined and narrowly scoped interfaces.

2.7 Data Transfer Services

While data sharing over HTTP is very popular, there is also a need for managed high-performance data transfer mechanisms. This is typically done via a data transfer node that is tuned for high-performance transfers over the WAN and resides in the Science DMZ portion of the center's network.

In particular, the cloud-based Globus [14] service has offered gateway providers an effective way to leverage the data transfer node concept. Globus exposes a REST API and single-sign-on service that can be integrated with a science gateway. It allows users to upload and download files through a Globus Connect or GridFTP server, while the Globus service manages the actual transfer.

2.8 Cloud Hosted Portals

With the advent of REST APIs that can capture all interactions with the HPC center, we are seeing an increase in web gateways hosted in the cloud. Under this model the entire application portal is hosted on an external cloud platform like Amazon Web Services, while communication with the HPC Center happens directly from the client via a REST API.

This allows the developer a significant amount of freedom from policy constraints placed by the HPC center, while maintaining full functionality, since all useful interactions with the HPC side are still available. Client-side JavaScript with the

appropriate Cross-Origin Resource sharing (CORS) [15] can be used to create powerful applications that can communicate with the cloud portal as well as the HPC REST service.

A single-sign-on system with a common federated identity plays an important role in facilitating this type of access in a seamless manner.

2.9 Containers

Container technologies like Docker [16] have created a new paradigm shift in this space. Containers enable a lightweight form of virtualization where the user can bundle the entire application stack, including the underlying OS, into a virtual container. This means that the service provider no longer needs to maintain complex software dependencies and versions for various stacks. Instead users can simply create a container with all the appropriate software dependencies in a self-contained environment that is isolated from other containers. The science gateway provider runs a Docker orchestration service, which can then run the user-supplied container.

The added benefit of containers is that this provides portability across multiple environments, since the same container can run on different platforms as long as they can run Docker. Thus users can develop a containerized application on their laptops, and the app can then be run at an HPC center (using tools like Shifter [18] or Singularity [19]) or in a cloud-based Docker service.

This also provides an easy path to horizontal scaling, as the same container can simply be cloned to provide more instances of the service. NERSC has deployed a Docker platform called SPIN that can run user-supplied Docker containers (such as the Sloan Digital Sky Survey Mirrors), that also provides access to filesystem and HPC resources.

3. Conclusion

We hope that our work in identifying common science gateways patterns, practices and trends proves to be useful to the community at large and can serve to inform future gateway deployments and development efforts. Web technologies move very rapidly and we do not expect this to be a static list. As we move forward we fully expect new additions to this catalog that will allow us to track ongoing developments in this space.

4. Acknowledgments

This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

5. References

[1] https://jamstack.org/

[2] Enis Afgan, Dannon Baker, Marius van den Beek, Daniel Blankenberg, Dave Bouvier, Martin Čech, John Chilton, Dave Clements, Nate Coraor, Carl Eberhard, Björn Grüning, Aysam Guerler, Jennifer Hillman-Jackson, Greg Von Kuster, Eric Rasche, Nicola Soranzo, Nitesh Turaga, James Taylor, Anton Nekrutenko, and Jeremy Goecks. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. Nucleic Acids Research (2016) 44(W1): W3-W10 doi:10.1093/nar/gkw343

[3] M. McLennan, R. Kennell, "HUBzero: A Platform for Dissemination and Collaboration in Computational Science and Engineering," Computing in Science and Engineering, 12(2), pp. 48-52, March/April, 2010.

[4] https://github.com/jupyter/jupyter/wiki/Jupyter-kernels

[5] Rollin Thomas, Shane Canon, Shreyas Cholia, Lisa Gerhardt, and Evan Racah, "Toward Interactive Supercomputing at NERSC with Jupyter", CUG 2017 Proceedings

[6] Fielding, R. T. (2000). *REST: Architectural Styles and the Design of Network-based Software Architectures.*

[7] S. Cholia, D. Skinner and J. Boverhof, "NEWT: A RESTful service for building High Performance Computing web applications," 2010 Gateway Computing Environments Workshop (GCE), New Orleans, LA, 2010, pp. 1-11.J. Clerk Maxwell, *A Treatise on Electricity and Magnetism,* 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp. 68-73.

[8] Dooley, Rion, et al. "Software-as-a-Service: The iPlant Foundation API", 5th IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS). IEEE, 2012.

[9] 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, 21-28. http://dx.doi.org/10.1145/2110486.2110490

[10] http://pydap.readthedocs.io/en/latest/

[11] https://www.incommon.org/

[12] Jim Basney, Terry Fleury, and Jeff Gaynor. 2013. CILogon: a federated X.509 certification authority for cyberinfrastructure logon. In *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery* (XSEDE '13). ACM, New York, NY, http://dx.doi.org/10.1145/2484762.2484791

[13] Eli Dart, Lauren Rotman, Brian Tierney, Mary Hester, and Jason Zurawski. 2013. The Science DMZ: a network design pattern for data-intensive science. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (SC '13). ACM, New York, NY, USA, Article 85 , 10 pages. https://doi.org/10.1145/2503210.2503245

[14] Ian Foster. 2011. Globus Online: Accelerating and Democratizing Science through Cloud-Based Services. IEEE Internet Computing 15, 3 (May 2011), 70-73. http://dx.doi.org/10.1109/MIC.2011.64

[15] https://www.w3.org/TR/cors/

[16] Dirk Merkel. 2014. Docker: lightweight Linux containers for consistent development and deployment. *Linux J.* 2014, 239, pages.

[17] SPOT Suite Portal for Light-Source Data and Simulations. https://spot.nersc.gov/

[18] Jacobsen, D. M., and R. S. Canon. "Shifter: Containers for HPC." Cray Users Group Conference (CUG'16). 2016.

[19] Souza, P., G. M. Kurtzer, C. Gomez-Martin, and PM Cruz e Silva. "HPC Containers with Singularity." In Third EAGE Workshop on High Performance Computing for Upstream. 2017.