Spin: A Docker-based Platform for Deploying Science Gateways at NERSC

Cory Snavely
NERSC
Lawrence Berkeley Laboratory
Berkeley, USA
CSnavely@lbl.gov

Gonzalo P. Rodrigo Alvarez
CRD
Lawrence Berkeley Laboratory
Berkeley, USA
GPRodrigoAlvarez@lbl.gov

Valerie Hendrix
CRD
Lawrence Berkeley Laboratory
Berkeley, USA
VCHendrix@lbl.gov

Shreyas Cholia
CRD / NERSC
Lawrence Berkeley Laboratory
Berkeley, USA
SCholia@lbl.gov

***Abstract*—This demonstration presents Spin, a Docker-based, on-premise cloud platform at NERSC that enables researchers to design, build, and manage their own science gateways and other services using container technology. After explaining the rationale behind building Spin and describing its basic architecture, staff will show how simple services can be created in a few minutes using basic tools. A discussion of science gateways that have been implemented with Spin will follow.**

Keywords—cloud, container, Docker, IaaS, infrastructure, science gateway, workflow

I. Introduction

The National Energy Research Scientific Computing Center (NERSC), at Lawrence Berkeley National Laboratory, is the mission high performance scientific computing resource for researchers directly supported by the Department of Energy's Office of Science (SC). Scientific projects supported by NERSC—especially those that involve the production, handling, or analysis of large data sets—increasingly demand additional tools and services that complement conventional computational processes. The function of these tools can vary widely and can include, for example,

- presenting and disseminating data and results from computational jobs
- tracking and managing iterative research workflows, making *in situ* flow decisions or parameter adjustments
- storing compiled summary data across many separate job runs

Traditional approaches to building science gateways and other network services such as these require not only costly infrastructure deployment and ongoing management, but also a significant amount of ongoing staff assistance for configuration and testing, installation of supporting software packages, and provisioning and deprovisioning of access to shared resources such as large database instances.

By leveraging Docker container technology, Spin (Scalable Platform Infrastructure at NERSC) helps to remove barriers for the increasing number of users who need to develop customized services to perform their research projects while also reducing the need for staff intervention and oversight. The portability of Docker containers is key, allowing for rapid development iteration and testing on private workstations, and, at the same time, helping to ensure that releases to the robust, redundant Spin platform will function as tested.

II. Key Components

The key architectural and functional components of the Spin system are as follows:

- *Management and Orchestration.* This is the core of the system, responsible for starting containers on the nodes of Spin, monitoring their health, handling failover and fault recovery, and all other aspects of container runtime. In addition, this component provides the user-facing API and CLI through which users create and manage their services. (Spin uses the Rancher system for management and orchestration.)
- *Docker Image registry.* This is a versioned repository for Docker images that are created by users and used as the building blocks for their science gateways. Examples include common web service software such as nginx or Apache Tomcat, databases or key-value stores, or homegrown network services.
- *Private networking.* A common feature in Docker environments, this subsystem isolates each service into its own encrypted network and manages routing to the outbound network.
- *Servers and Storage.* CPU cores, RAM, and physical networking provided by commodity servers and storage are allocated to containers on demand by the management and orchestration system.
- *Security Controls and Policy Enforcement.* This set of software and services includes user authentication, access controls, log collection and analysis, and specialized business logic that monitors user activity in Spin and enforces security rules.

III. Science Gateway Design in Spin

Science gateways in Spin follow microservice design principles, where each microservice generally corresponds to

Presented at Gateways 2018, University of Texas, Austin, TX, September 25–27, 2018.
https://gateways2018.figshare.com/

one Docker container. Simple examples include, e.g., an Apache web server and Javascript-based application code framework (in one container) and a MySQL database (in another).

Complete services are declared using the common Docker compose syntax, based on YAML. The above example would be declared for instantiation in Spin with code similar to the following, which specifies two containers that together make up a working service:

```
version: '2'
services:
  apache:
    image: httpd
    environment:
      MYSQL_DATABASE: "gatewaydb"
      MYSQL_PASSWORD: "test123"
    volumes:
    - /home/fred:/var/www/html:ro
  mysql:
    image: mysql
    environment:
      MYSQL_RANDOM_ROOT_PASSWORD: yes
      MYSQL_USER: "gatewaydb"
      MYSQL_DATABASE: "gatewaydb"
      MYSQL_PASSWORD: "test123"
```

Creating the running service from the declaration is done with a simple command:

```
$ rancher up
INFO[0001] [apache]: Creating
INFO[0001] [mysql]: Creating
```

Services can become quite complex, including many more containers for additional functionality or for scale-out performance.

Support for the Docker compose syntax in Spin helps to simplify service deployment to different hosting environments. For example, with minor variation, the same Docker compose file can be used to deploy a service on a laptop, on a Docker Swarm environment, and on Spin. Combined with the use of community-supported Docker images as a best practice, a service can be made extremely portable.

IV. Underlying Technology

In late 2016, NERSC staff began evaluating three container-based technologies mature enough to base the Spin system upon: Docker Data Center, Kubernetes, and Rancher. While each had its strengths, the Rancher system was chosen based on several important factors:

- *Orchestration Options.* Rancher offered the flexibility to use either of two orchestration subsystems, "Cattle" (a custom, built-in system compatible with Docker compose) or Kubernetes (which originated at Google, and was already gaining popularity at the time).
- *Open Source with Commercial Support.* The software license model allowed for an extended evaluation period with the option to add support to production infrastructure when critical workloads were introduced.
- *Infrastructure Management.* The tools provided by the Rancher system enabled NERSC staff to perform routine operational duties as is customary with other virtualization technologies, e.g. automated failover in case of failures, rolling upgrades, basic oversight of system utilization and all deployed services, etc.
- *Comprehensive API and CLI.* The Rancher system featured an API-based design exposed through a simple CLI. While its UI provided ease of use, the API and CLI were seen not only as a favorable architectural design, but one that would also offer future flexibility for integration to other systems.

In late 2017, NERSC had gained sufficient experience with internally-managed services and began planning for user self-service. This required developing a subsystem to enforce security controls and policy, as described in II above. For example, Docker containers can typically run with arbitrary user IDs and privileges, but these actions must be suppressed for security purposes under specific circumstances, such as when containers access global file systems.

The subsystem that enforces these security controls and policy is integrated with Rancher through a capability called the API interceptor. When users deploy services via the Rancher CLI, underlying calls to the API are examined and either modified or rejected based on policy. This subsystem is used, for example, to whitelist allowed functionality, limit container privileges, and enforce file system access controls.

In mid-2018, Rancher 2 was released. This new release incorporates Kubernetes internally as the orchestration system, replacing Cattle. NERSC is currently evaluating Rancher 2, which is likely to offer additional functionality and simplify (but not wholly eliminate) the existing security subsystem.

V. Pilot Phase and Adoption

Spin entered into a pilot stage in mid-2017 and now underlies a variety of user-developed science gateway services, as well as the staff-operated tools JupyterHub, RStudio, and others. The engineers building Spin partnered closely with users for implementation, developing additional knowledge and a documented set of conventions and best practices. This knowledge was collected and formed into an instructional program for users, just launched in May 2018.

One of Spin's early adopters is the Science Search project, situated in Berkeley Lab's Computational Research Division (CRD). Science Search uses machine-learning techniques to extract metadata to build a search index across many scientific data sets. According to Gonzalo Rodrigo Alvarez, a postdoctoral researcher working on the project, this required infrastructure for hosting web-based services that was tightly integrated to NERSC compute and storage resources. The Spin platform suited these requirements while also affording Alvarez a simple, Docker-based software development and deployment model.

ESS-DIVE (Environmental Systems Science Data Infrastructure for a Virtual Ecosystem) is another project being coordinated in CRD and built on Spin through close engagement with NERSC. This data archive is leveraging NERSC storage via Spin to offer a scalable, reliable repository for earth science data. Data packages in the repository are also immediately accessible to NERSC compute systems, enabling the deep analysis and processing of data sets and metadata that is envisioned by ESS-DIVE project personnel.

VI. Future Direction

NERSC expects demand for Spin to continue to grow, and will offer additional instructional workshops on an ongoing basis. Users who have already developed science gateways at NERSC through more conventional means are able to attend these workshops and use the skills acquired to migrate their services into Spin, where they can manage them more easily and with greater autonomy.

The platform also provides an ideal resource for staff to incubate new services of their own, which in turn helps familiarize them with the system and prepare for answering user inquiries.

The software-defined, declarative nature of the system also provides exciting possibilities for instantiating services on demand through completely automated means, such as during compute job prologue processing or via external API calls from other workflow managers or user facilities. These capabilities are being explored at NERSC under the rubric of a fully comprehensive facility API that would expose functions such as automated data movement, batch job submission, and instantiation of services in Spin via simple calls.