Generalizing Workbench

Thoughts on generalizing what we currently call "Labs Workbench" as a general platform that can be used to support multiple distinct use cases.

Potential Use Cases

NDS Labs Workbench

This is the existing primary use case: a service for the NDS community to explore, develop, share, and test a variety of data management tools.

Education and training

One of the clearest proven uses of the platform is for education and training purposes. Labs Workbench was used for:

- IASSIST 2016 for a workshop on integrating Dataverse and iRODS (30 users)
- NDSC6 for a workshop for the development of Docker containers (20 users)
- Phenome 2017 for a workshop on how to use the TERRA-REF reference data platform and tools (40+ users)
- Planned iSchool pilot for data curation educators
- Possible deployment by Big Data Hub on commercial provider, such as Microsoft Azure, AWS, GCE to provide sandbox

Each environment is unique, but there are a few basic requirements:

- Custom catalog only of the tools needed for the training environment
- User accounts that can be created with and without requiring registration (e.g., batch import) or approval.
- Optional TLS and vulnerability management. Not all services will be around long enough to merit it.
- Wildcard DNS (www.x.ndslabs.org)
- Short term scalable resources (e.g. 40 users, 4 hours) as well as longer-term stable resources (11 weeks, 24x7, maintenance allowed)
- Custom shared data available to users of the system. For example, Phenome 2017 wanted sample data for each user.
- Ability to deploy a dedicated environment, scale it up, and tear it down. At the end of the workshop/semester, access can be revoked.
- Ability to backup/download data
- Ability to deploy system under a variety of architectures (commercial, local, OpenStack, etc)
- Ability to host/manage system at NDSC/SDSC/TACC with failover
- Monitoring
- Support and clear SLA
- Worth considering:
    - Authentication that ties to existing systems (e.g., Shibboleth, Oauth)
    - Custom documentation and branding/skinning
    - Configurable quotas (not one-size fits all)

Scalable analysis environment

We can also envision the platform working as a replacement for the TERRA-REF toolserver or as a DataDNS analysis environment. In this case, the requirements include:

- Custom catalog of tools supported for the environment as well as user-defined catalogs
- User accounts that can be created without requiring registration (API). For example, shared auth with Clowder for TERRA-REF.
- Authentication that ties to existing systems (e.g., Shibboleth, Oauth)
- Long-term stable and scalable resources. Ability to add/remove nodes as needed.
- Ability to terminate long-running containers to reclaim resources
- Custom documentation and branding, although the UI itself may be optional
- Ability to mount data stored on remote systems (e.g., ROGER) as read-only and possibly read-write scratch space
- Ability to add data to a running container, retrieved from a remote system?
- Clear REST API to
    - List tools; list environments for a user; launch tools; stop tools;
- Security/TLS/vulnerability assessment

Platform for the development and deployment of research data portals

Another use case, really a re-purposing of the platform, is to support the development and deployment of research data portals – aka, the Zuhone case. In this case we have something like workbench to develop and test services, with the ability to "push" or "publish", which is still a bit unclear.

Requirements include:

- Ability to develop data portal using common tools (e.g., development environments or portal toolkits)
- Ability to deploy (host) data portal services "near" datasets (e.g., ythub)
- Security (TLS)
- Monitoring (Nagios)
- Custom DNS entries (gcmc.hub.yt)
- Optional authentication into portal services (i.e., restricting access to service from end users)

Other: Working with Whole Tale

Whole Tale will also support launching Jupyter and R notebooks, but is more focused on 1) bringing data to the container and 2) reproducibility. Datasets are registered via Girder. Authentication is handed via Globus auth. User's home directory will be iRODS. WT will be deployed at NCSA and TACC. Containers will be time-constrained and launched at the site with the right data. A key component is the Data Management System which handles caching data locally and exposing via fuse filesystem to the container (and therefore handling permissions). They are hoping to leverage Labs Workbench – or at least Kubernetes – for container management.

- Is there a way that WT can leverage Workbench to launch remote containers? At the very least, relying on a Kubernetes federation?

Other: Cyverse (work in progress)

Another case coming out of the Phenome conference is the possibility of using workbench to provide R/Jupyter support for Cyverse:

> "I am very interested in setting up the CyVerse DataStore with iRODS on the Workbench. CyVerse has been talking for months about integrating Jupyter and RStudio into our ecosystem. The Labs workbench appears to be just the sort of thing we (or at least, I) need."

The Cyverse Data Store supports iRODS iCommands, FUSE, or an API (http://www.cyverse.org/data-store). We can envision several approaches: 1) Workbench mounts the Cyverse data directly; 2) Workbench mounts data via iRods; 3) Workbench retrieves data via API.

Requirements might include:

- Ability to install Labs Workbench at Cyverse or
- Ability to use Labs Workbench to access Cyverse data
- Start an R or Jupyter container that can access the Cyverse data store via iRODS
    - Data mounted directly (local install at Cyverse)
    - Data transfered via iRODS
- Ability to handle Cyverse authentication

Other: Collaborative Development Cloud (work in progress)

One issue that has come up recently on the KnowEnG UI development is the need for TLS-protected development instances with basic auth in front. Since we offer a slew of development environments with built-in TLS and basic auth, this seemed like a natural fit.

We also offer Jenkins CI. ARI already has set up for some of the KnowEnG folks, but could help other similar teams gain experience with setting up their own CI, and even testing applications that they could develop from within Labs. I played around over the weekend, and discovered that there are also several GitLab and Atlassian suite (JIRA + Confluence) images floating around that might be usable from within Labs.

Given the above, we have the potential to offer the following powerful combination of tools for any team of collaborating developers:

- Private Source Control (via GitLab)
- Project Documentation / Internal Development Wiki (via Confluence)
- Ticketing workflow system (via JIRA)
- Continuous Integration (via Jenkins CI)
- Development Environments for several popular languages (via Cloud9 and friends - with the potential to add more)

True, you could outsource for any one of these (Atlassian provides the first three), but Labs is the only place I can think of where you could get them all! 😉

Pros:

- development-in-a-box: give teams all the tools they need to succeed right away
- no need to remember 10 different URLs (if development teams shared a project in Labs) - access all of your development tools from one place!
- automatic TLS with basic auth protecting all services (albeit self-signed, unless you have a CA)
- quickly spin up new team members without spending a week installing dependencies and preparing environments

Cons:

- full disclosure: I made this use case up... I have no idea if this is a real need that is unmet
- storage is flaky, and hosting a source-code repository or ticket backlog directly violates my original ideology
  - "DO NOT store anything critical on Workbench. Storage is volatile and may go away at any point - save a hard copy externally."
- requires that any service developed be runnable from within Labs, or else testing your code becomes more difficult than on a VM
  - currently: this would require that all services run from Labs (as well as, by extension, all things developed in Labs) be available via Docker Hub, which is may be too public for KnowEnG / ARI's current licensing needs

Other: Workflow Orchestration (work in progress)

See **NDS-664** - Getting issue details... **STATUS**

Another need that has come up on the KnowEnG project is the ability to run a cluster of compute resources for scheduling analysis jobs. These jobs come in the form of a DAG (directed acyclical graph) and are effectively Docker containers with dependencies. Since the API server already contains much of the logic to talk with etcd and Kubernetes, it might not be so difficult to extend Workbench to run these types analysis jobs.

Our "spec" architecture is already set up to handle running dependent containers and ensuring that they are running before continuing on to the next containers in the chain. If we were to add a flag (i.e. type == "job") to the specs, that could signal to the API server to run a job, instead of a service/rc /ingress, and to wait for the job to be "Completed" before running the next dependency.

I created a simple example of a Job spec YAML on raw Kubernetes just to see how a multi-container job would run and be scheduled. Apparently multiple Jobs can be scheduled at once, containing multiple containers. Each container within the Job will run sequentially (in the order listed in the spec).

I still need to ask for an example of a real life example of both a simple and a complex DAG to gather more details and create a more realistic prototype. We had previously discussed investigating Kubernetes to handle the scheduling, but we decided to look into BD2K's cwltoil framework instead.

Pros:

- seems relatively small-effort to extend Labs in this way
- more control over the scheduler than with raw Kubernetes, with direct access to the developers (ourselves)
- we offer a user interface, which toil and kubernetes do not (aside from the mesos / kubernetes dashboard, which are fairly limited)

Cons:

- BD2K created cwltoil, and KnowEnG is a product out of the BD2K, so we miss out on a political win by using Labs
- toil was created for exactly this purpose: scalable DAG / CWL jobs
- toil would allow us to run jobs using the CGC's CWL system
- still some kinks in our platform (actual bugs, storage, commercial cloud deployment is not formalized, etc)

Current features/components

Deployment (OpenStack)

We currently have two methods of deploying the Labs Workbench service: 1) ndslabs-startup (single node) and 2) deploy-tools (multi-node OpenStack)

The ndslabs-startup tool provides a set of scripts to deploy NDS Labs services to a single VM. This is intended primarily for development and testing. The deployment is incomplete (no shared storage, NRPE, LMA, backup), but adding these services would be minor. Minikube was considered as an option, but is problematic when running on a VM in OpenStack and might require additional investigation.

The deploy-tools image provides a set of Ansible plays designed specifically to support the provisioning and deployment of a Kubernetes cluster on OpenStack with a hard dependencies on CoreOS and GlusterFS. It's unclear whether this can be replaced by openstack-heat. Deploy-tools has 3 parts: 1) OpenStack provision, 2) Kubernetes install, and 3) Labs components install. The OpenStack provision uses the OpenStack API and Ansible support to provision instances and volumes. The Kubernetes install is based on the contrib/ansible community tools with very minor local modifications. The Labs components install is primarily deploying Kubernetes objects.

For commercial cloud providers, we cannot use our deployment process. Fortunately, these services already have the ability to provision Kubernetes clusters: AWS, Azure, and GCE.

CoreOS (Operating system)

The deploy-tools image assumes that you are deploying CoreOS instances. This choice is arbitrary, but there are many assumptions in the deploy-tools component that are bound to the OS choice. Different providers make different OS decisions. Kubernetes seems to lean toward Fedora and Debian. GCE itself is Debian. Azure Ubuntu, etc. This may not be important if we can rely on Kuberenetes deployment provided by each commercial cloud provider.

Docker (Container)

The Labs Workbench system assumes Docker, but there are other container options. Kubernetes also supports rkt. This is something we've discussed but never explored.

Orchestration (Kubernetes)

Labs Workbench relies heavily on Kubernetes itself. The API server integrates directly with the Kubernetes API. Of all basic requirements, this seems to be one that's unlikely to change.

Gluster FS (Storage)

Labs Workench uses a custom Gluster FS solution for shared storage. A single Gluster volume is provisioned (4 GFS servers) and mounted to each host. The shared volume is accessed via hostPath by containers.

This approach was necessary due to lack of support for persistent volume claims for OpenStack. For commercial cloud providers, we'll need to re-think this approach. We can either have a single volume claim (giant shared disk), volume claim per user, or volume claim per application. There are benefits /weaknesses in all of these approaches. For example, in a cloud provider, you don't want to have a giant provisioned disk with no usage. The per account approach may be better.

Other storage includes mounted volumes for /var/lib/docker and /var/lib/kubelet.

Dedicated etcd

We no longer rely on the Kubernetes etcd service, and provide our own that runs within the cluster.

SMTP Server / Relay

We now provide an in-cluster SMTP relay that can be configured to use Google credentials. This makes it very simple to use your Google credentials to send verification / approval / support e-mails.

REST API Server/CLI

Labs Workbench provides a thin REST interface over Kubernetes. Basic operations include: authentication, account management (register, approve, deny, delete), service management (add/update/remove), application instance management (add/update/remove/start/stop/logs), console access. The primary purpose of the REST API is to support the Angular Web UI. The API depends on Kubernetes API, etcd, Gluster for shared volume support, and SMTP support.

Web UI

The Web UI is a monolithic AngularJS application that interfaces with the REST API.

Application Catalog

Labs workbench provides the ability to support custom application catalogs via Github. Eventually, it may be nice to provide a more user-friendly method for adding/removing services.

Ingress Controller

Labs Workbench relies on the Kubernetes contrib Nginx ingress controller (reverse proxy) to provide access to running services including authentication. We've made only minor modifications to some of the configuration options.

We know that GCE uses a version of the Nginx controller, but it's unclear whether it's the same as the version we use.

Wildcard DNS and TLS

Labs Worbench relies on wildcard DNS (*.workbench.nds.org) to provide access to running services. For security purpose, this also requires a wildcard TLS certificate (revokable, 1 year).
For short-term deployments, TLS can be disabled (DNS is still required). It's unclear how this relates to commercial cloud providers.

Backup

A backup container is provided to backup Gluster volumes, etcd, and Kubernetes configs. This is tightly coupled to the Workbench architecture. The backup server is hosted at SDSC. We should be able to generalize this solution, if needed.

Monitoring (Nagios/Qualys)

A Nagios NRPE image is provided to support monitoring instances with some Kubernetes support. We also use the contrib addons (Grafana, etc), deployed as standard services.

Commercial cloud providers provide their own monitoring tools, e.g., GCE Monitoring.

Docker cache

The Labs Workbench system deployed via deploy-tools includes a local Docker cache to minimize network traffic for image pulls

Private Docker registry

The Labs Workbench system deployed via deploy-tools includes a private Docker registry to privately share images within your cluster without needing to share them out to Docker Hub

- This will need to be tested

Automated Testing

The Angular Web UI includes a facility for executing automated Selenium smoke tests.

What would need to change?

- Deployment process needs to be generalized to support more environments than OpenStack and likely more OSes than CoreOS
- Volume/storage will not work on cloud providers
- Potentially allow for Web UI customization
- Better custom catalog support
- Confirm ingress (including DNS/TLS) support with Commercial providers
- We should work towards adding options for some of the above components, to reduce minimal deployment size

Other thoughts

- TERRA-REF case:
    - We can imagine a couple of cases. First, TERRA-ref as a full install with a system catalog and user catalogs. Second, TERRA-REF as a user of the current system with a custom catalog and no individual user namespaces. We could also have a TERRA-REF data provider to get data into containers.
- Cyverse case:
    - Similarly, we can imagine a Cyverse user launching notebooks in the current system with a Cyverse data provider (FUSE, iRODS, etc)
    - Or a full install of Workbench by Cyverse