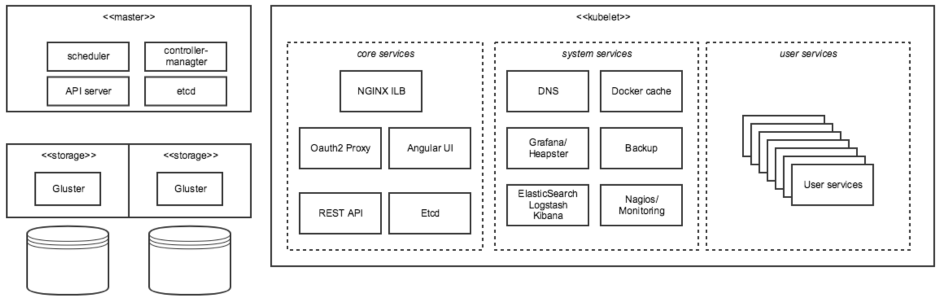
Workbench Architecture

This document refers to release 1.1.

Overview

The Labs Workbench consists of a set of core services (Angular UI, REST API, service catalog) deployed in a Kubernetes cluster that allow users to start applications/services in the cluster.

The overall architecture includes a Kubernetes cluster, typically deployed on OpenStack, with core services (UI, API, etc), system services (in-cluster DNS, monitoring, logging), and user services. The system also supports optional distributed storage across user containers (i.e., you can access the same "home" directory in your running applications across containers/applications).



Workbench on Kubernetes

(Note: We are in the process of upgrading our deployment process to use a new terraform approach that will allow us to upgrade to Kubernetes 1.10. The deploy-tools process referenced below still deploys a 1.5 cluster).

Kubernetes is a container orchestration environment. This section describes Kubernetes as applied in Labs Workbench. For more detailed information, please see their documentation.

The Labs Workbench platform assumes a Kubernetes environment that has been deployed using out deploy-tools container. This is a set of Ansible playbooks that provisions instances and volumes via the OpenStack API. The deployment process has been tested on Jetstream, NCSA Nebula, and SDSC Cloud. For single-node/development environments, you can also use our ndslabs-startup process, which deploys Kubernetes via Hyperkube.

Kubernetes concepts as applied in Labs Workbench:

- Master node: runs the scheduler, controller/manager, API server, and etcd. May be in high-availability configuration.
- Nodes: Schedulable resources (i.e. VMs) running the kubelet and kube-proxy services. Nodes may be labeled if they have specialized properties. Labs Workbench typically has multiple compute nodes for scheduling user services and at least two storage nodes to run Gluster FS for distributed storage across containers. The storage nodes can optionally run user services.
- Namespaces: A way to divide and constrain cluster resources. Labs Workbench uses the default namespaces to run core services (UI, API server, etc) and per-user namespaces to run user services. Each user namespace has a quota to restrict CPU/Memory. (Storage quotas are also supported via Gluster, but this is independent of Kubernetes)
- Services, Daemonsets, Replication Controllers, Deployments, and Pods: These are many of the most basic Kubernetes abstractions. In Labs Workbench, all users services run in Pods (which may contain multiple containers). Replication controllers and deployments are used to manage running Pods. DaemonSets are used to run services across nodes in the cluster (e.g., monitoring clients). Services provide a network abstraction.
- Ingress rules: Via the NGINX ingress controller, Labs Workbench uses ingress rules to support routing traffic to user services. For example https://mynotebook.workbench.ndslabs.org would be routed to a Pod running in my namespace.
- Overlay network: Workbench currently uses the Flannel overlay network, although other options (Weave, Calico) should be transparent.

Labs Workbench adds the following abstractions:

- Service catalog: A set of JSON specs typically stored in a github repository that define which applications are available to users.
- Service spec: An individual specification representing a component of a service (e.g., https://github.com/nds-org/ndslabs-specs/blob/master/clowder/clowder.json). These can be complex, multi-component stacks with optional dependencies.
- Stack: An individual user's running instance of an application stack. Users can launch multiple instances of the same stack, if desired.
- Account: User account information, stored in the etcd key-value store

Additional features:

- Additional shared volumes (currently requires host mounting via NFS)
- In-cluster SMTP relay for email
- Optional approval workflow
- Optional Oauth support
- Configurable inactivity timeout (automatically shutdown running user services)
- Configurable user resources quotas

- Quickstart links – shortcuts to automatically start instances from the click of a link/button.

Current priorities

We are currently working toward a 2.0 release that will include the following:

- Terraform deployment process based on Data8 kubeadm bootstrap.
- Upgrade Kubernetes to 1.9+
- Real Kubernetes volume support to enable transportability to commercial cloud
  - Support for both Rook and NFS provisioners for persistent volume claims in OpenStack
- RBAC support
- Packaging workbench as a Helm chart to simplify deployment
- Commercial cloud support, with initial focus on deployment to AWS