VOLUMES AND DATA

Volumes and Data

Overview

Container engines have traditionally not offered storage that outlives the container. As containers are considered transient, this could lead to a loss of data, or complex exterior storage options. A Kubernetes **volume** shares the Pod lifetime, not the containers within. Should a container terminate, the data would continue to be available to the new container.

A volume is a directory, possibly pre-populated, made available to containers in a Pod. The creation of the directory, the backend storage of the data and the contents depend on the volume type. As of v1.13, there were 27 different volume types ranging from rbd to gain access to Ceph, to NFS, to dynamic volumes from a cloud provider like Google's gcePersistentDisk. Each has particular configuration options and dependencies.

The Container Storage Interface (CSI) adoption enables the goal of an industry standard interface for container orchestration to allow access to arbitrary storage systems. Currently, volume plugins are "in-tree", meaning they are compiled and built with the core Kubernetes binaries. This "out-of-tree" object will allow storage vendors to develop a single driver and allow the plugin to be containerized. This will replace the existing Flex plugin which requires elevated access to the host node, a large security concern.

Should you want your storage lifetime to be distinct from a Pod, you can use Persistent Volumes. These allow for empty or pre-populated volumes to be claimed by a Pod using a Persistent Volume Claim, then outlive the Pod. Data inside the volume could then be used by another Pod, or as a means of retrieving data.

There are two API objects which exist to provide data to a Pod already. Encoded data can be passed using a Secret and non-encoded data can be passed with a ConfigMap. These can be used to pass important data like SSH keys, passwords, or even a configuration file like **`/etc/hosts`**.