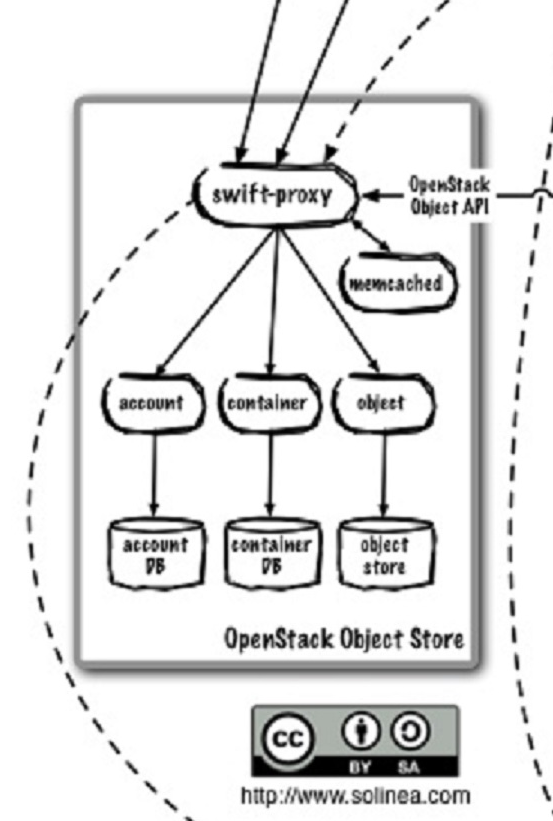
**Swift – Software to reliably store billions of objects distributed across standard hardware.**

HTTP(S) Python/CLI

API Horizon API



Nova-api

Identity (Keystone) API

**Swift** - aims to provide a massively scalable and redundant object store conceptually similar to Amazon’s S3 service. To provide this scalability and redundancy, it writes multiple copies of each object to multiple storage servers within separate “zones.” Zones are a logical grouping of storage servers that have been isolated from each other to guard against failures. The level of isolation is up to the cloud operator; they can be isolated on differing servers (ability to lose individual servers), different racks (ability to lose entire rack), different sections of the data center, or even different data centers. Each choice provides a different level of isolation and cost.

**swift-proxy** -  Proxy Server is responsible for tying together the rest of the Swift architecture. For each request, it will look up the location of the account, container, or object in the ring (see below) and route the request accordingly. The public API is also exposed through the Proxy Server.

**the ring** - A ring represents a mapping between the names of entities stored on disk and their physical location.

the rings:

account

container

object

**Storage Policies** - provide a way for object storage providers to differentiate service levels, features and behaviors of a Swift deployment. Each Storage Policy configured in Swift is exposed to the client via an abstract name. Each device in the system is assigned to one or more Storage Policies. This is accomplished through the use of multiple object rings, where each Storage Policy has an independent object ring, which may include a subset of hardware implementing a particular differentiation.

**Object Server** - very simple blob storage server that can store, retrieve and delete objects stored on local devices.

**Container Server** - primary job is to handle listings of objects. It doesn’t know where those object’s are, just what objects are in a specific container. The listings are stored as sqlite database files, and replicated across the cluster similar to how objects are. Statistics are also tracked that include the total number of objects, and total storage usage for that container.

**Account Server** - very similar to the Container Server, excepting that it is responsible for listings of containers rather than objects.

**memcached** - Several of the Services rely on Memcached for caching certain types of lookups, such as auth tokens, and container/account existence. Swift does not do any caching of actual object data.

**Replication** - designed to keep the system in a consistent state in the face of temporary error conditions like network outages or drive failures.

**Updaters** - There are times when container or account data can not be immediately updated. This usually occurs during failure scenarios or periods of high load. If an update fails, the update is queued locally on the filesystem, and the updater will process the failed updates.

**Auditors** -  crawl the local server checking the integrity of the objects, containers, and accounts. If corruption is found (in the case of bit rot, for example), the file is quarantined, and replication will replace the bad file from another replica.