Build your own Cloud

An Open Source approach to Imagery Storage

James Klassen
SharedGeo
2012 FOSS4G-NA Washington, DC

The Mission

- Support the Great Lakes Restoration Initiative (GLRI) partners by providing access to remote sensing imagery for US Fish & Wildlife
- Provide access in a variety of formats

The Datasets

- Support up to 50TB of data initially
- Satellite Imagery, Aerial Ortho-photos, (and other formats: radar, stereo pairs)
- Workload: Write once, read many

The Constraints

- Limited initial budget
- Limited staff time to manage system
- Ability to grow seamlessly as data grows
- Limit access to sensitive/licensed datasets

The Solution

- Bulk image store
- Image catalog
- Services/Viewers

Image Storage

- OpenStack Object Storage (a.k.a. Swift)

What's Swift?

- Distributed Key/Value store
- Developed by Rackspace, NASA and others
- Part of the OpenStack Suite
- Optimized for long term storage
- Apache 2.0 License

Why Swift?

- Distributed Hash Table design
- No single point of failure
- Gracefully handles large objects (>5GB)
- HTTP/REST based API
- Supports HTTP 1.1 Range Queries

Why Swift

- Graceful reconfiguration when change storage nodes
- Resilient to failure
  - Server/Drive/Network
  - Misconfiguration
- Security built in (not all datasets are public)

Why Swift

- Scales (nearly) linearly with addn'l hardware
  - Transactions/sec (throughput)
  - Storage Capacity
- Low hardware cost, free software cost.

Alternatives Considered

- Multiple Server RAID + Apache
- NoSQL – BigCouch (and friends)
- Distributed Filesystems
- S3

Our Implementation

- Rather small by Swift standards
- 2 Proxy/Auth Nodes
- 4 Storage Nodes

Proxy Nodes

- Provide “public” HTTP interface
- Provide authentication/authorization
- Map requests to correct (set) of storage nodes
- Handle storage node failure

Storage Nodes

- Backing store for:
  - Accounts
  - Containers
  - Objects

The Ring

- Assigns partitions to storage nodes
- Every node has a copy of the ring
- Replication/Auditing not centrally coordinated

The Tradeoffs

- Latency >> nginx serving from files
- Network, network, network
- Dedicated Servers (min of 5 suggested)
- Learning Curve

Image Catalog

- Store and Index image metadata
- PostGIS backend
- Follow OpenAerialMap schema (with some extensions)
- Enable image search tools

Services/Viewers

- Orthorectified-imagery
  - WMS/WCS/Tiles via MapServer and MapCache
- “Raw” Files
  - Served directly from Swift

Capacity Planning

- Processing nodes to seed Tilesets
- HTTP/REST enables upstream caching (e.g. using AWS/Rackspace)

Relevant Links

- http://swift.openstack.org/index.html
- http://cyberduck.ch/
- https://github.com/oam/oam

Questions?

James Klassen

SharedGeo

jklassen@sharedgeo.org