redhat.

# OpenStack Swift 101

## Highly available, durable and scalable object storage

Christian Schwede
Principal Software Engineer
5. October 2016, LinuxCon Europe

# OpenStack and OpenStack Swift

# Short overview on OpenStack

- Open Source Software for creating private and public clouds
- OpenStack founded by Rackspace and NASA in 2010
  - Nova (Compute) and Object Storage (Swift)
- Today: 6 core services, 13 optional services and many more subprojects
- Tens of thousands of contributors, hundreds of companies and countries involved
- REST-based APIs, written in Python
- Well-tested, reviewed by elected community members

# Object Storage

Highly available, durable and scalable

- Traditional storage systems: filesystems and block storage
  - Structured data
  - Expensive to scale
  - Private access
- Object Storage
  - Best suited for unstructured data
  - Multi-tenant, shared-nothing architecture
  - Highly available, scalable, and durable
  - Running on commodity hardware
  - Easy to access using REST API

# Use cases & usage

Object storage in general and Swift especially

- REST-based API makes it easy to use in large-scaled **web applications**
- Public and private **storing of large sets of files** - photos, videos, blob data
- Highly reliable **backups**, and even available if parts of the cluster are unreachable
- **Archiving** large sets of unstructured data for analytics
- Biggest deployment at Rackspace - **more than 100PB**
- Many other deployments in the range of tens of PB (OVH, HPE, IBM, Symantec, NTT)

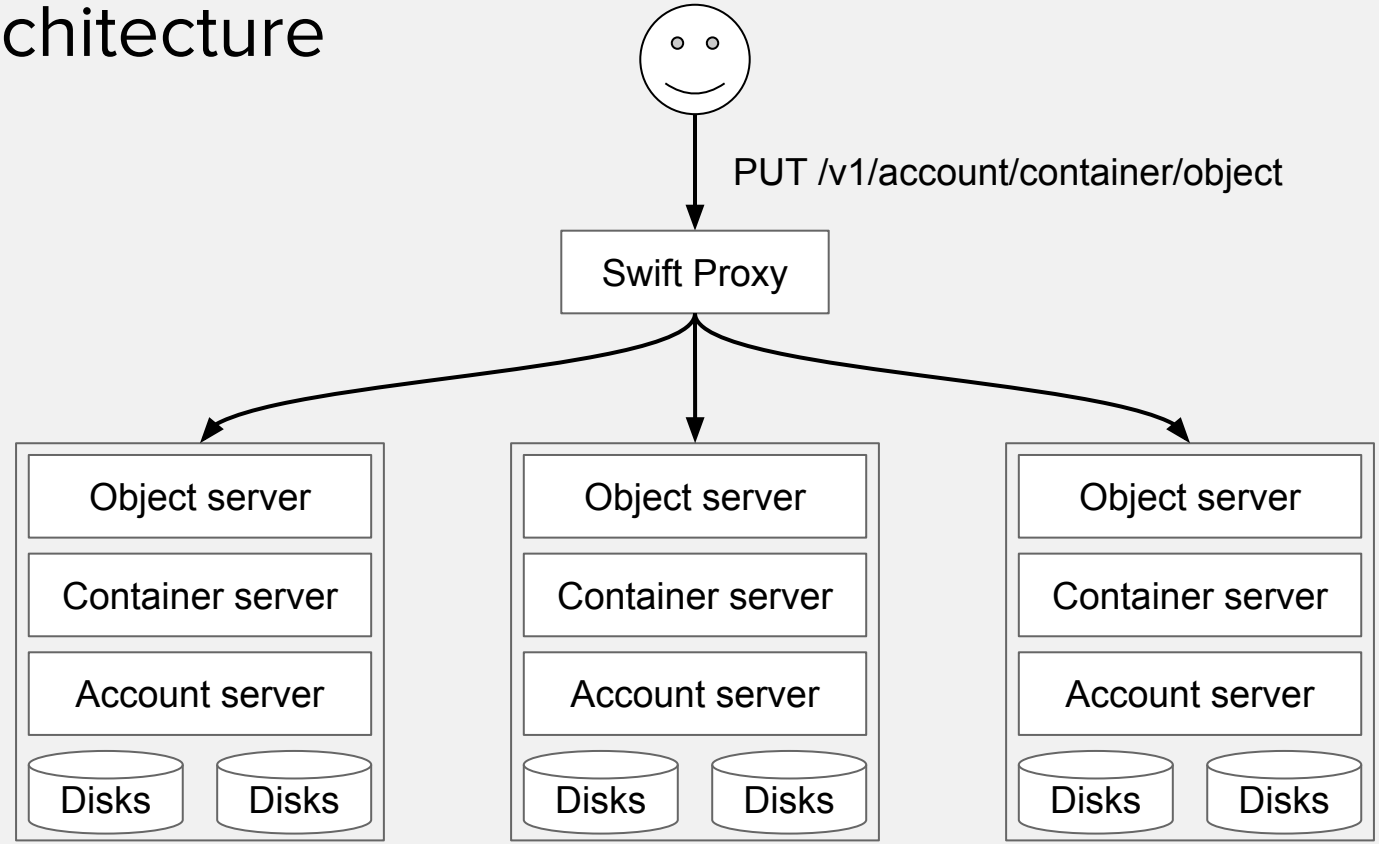
# OpenStack Swift

# Storing & retrieving data

Accounts, containers & objects

- **Flat namespace**: accounts, containers and objects
  - No nested directories
- **Account: collection of containers**
  - List containers: GET /v1/accountname/
  - Create container: PUT /v1/accountname/containername/
- **Containers: collection of objects**
  - List objects: GET /v1/accountname/containername/
  - Upload object: PUT /v1/accountname/containername/objectname
  - Retrieve object: GET /v1/accountname/containername/objectname

# Architecture



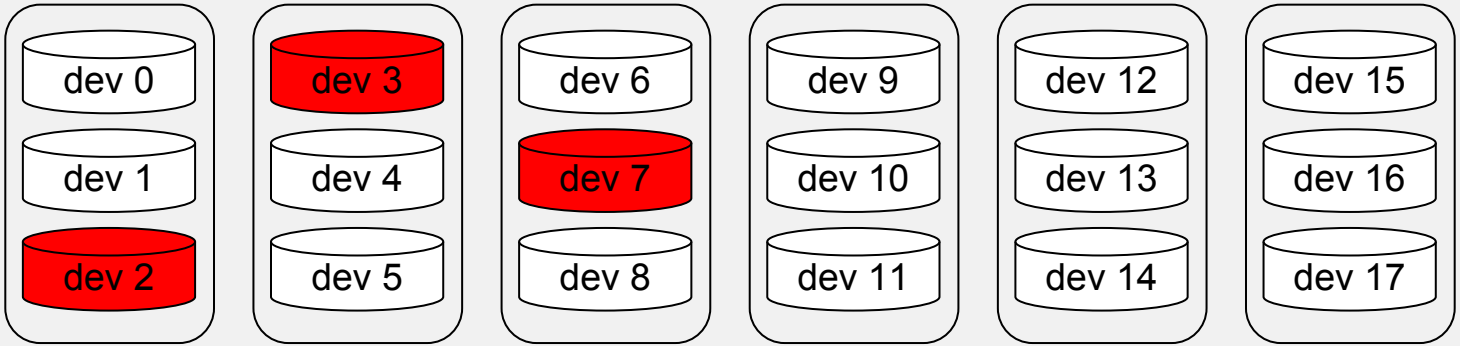
# Data durability

Ensuring your data is still the same for ages

- Replicated or Erasure Coded?
  - Depends on your use case
- Proxy returns data only if content matches stored checksum
- Continuously running background processes
  - Auditors: ensuring there is no bit-rot
    - Quarantining replicas if checksum mismatch
  - Replicators: ensuring all replicas are stored multiple times on remote nodes
  - Reconstructors: recomputing missing erasure-coding fragments

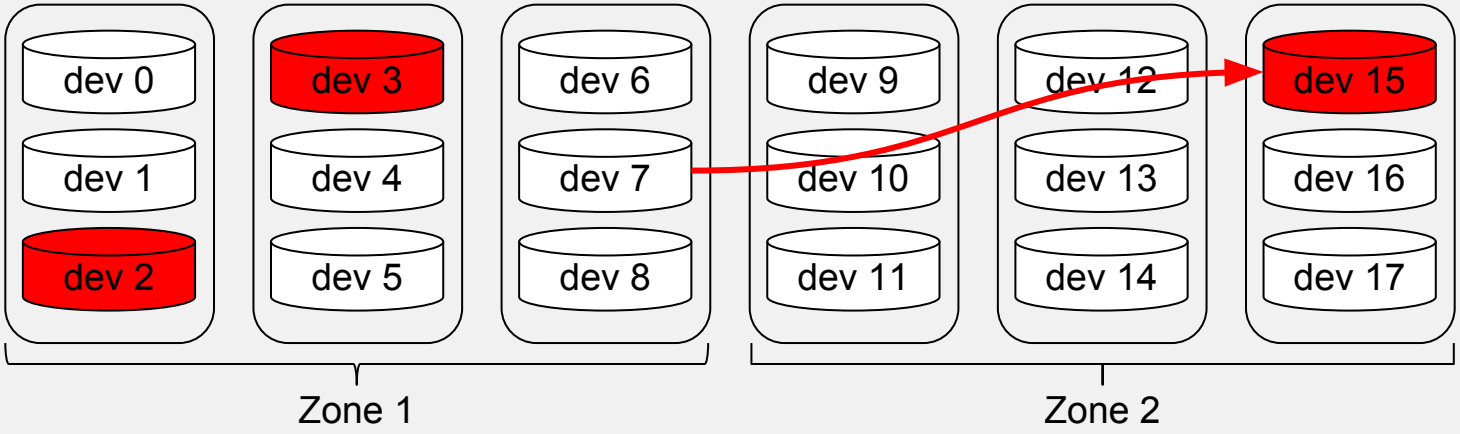
# Failure domains

Ensuring high availability and durability



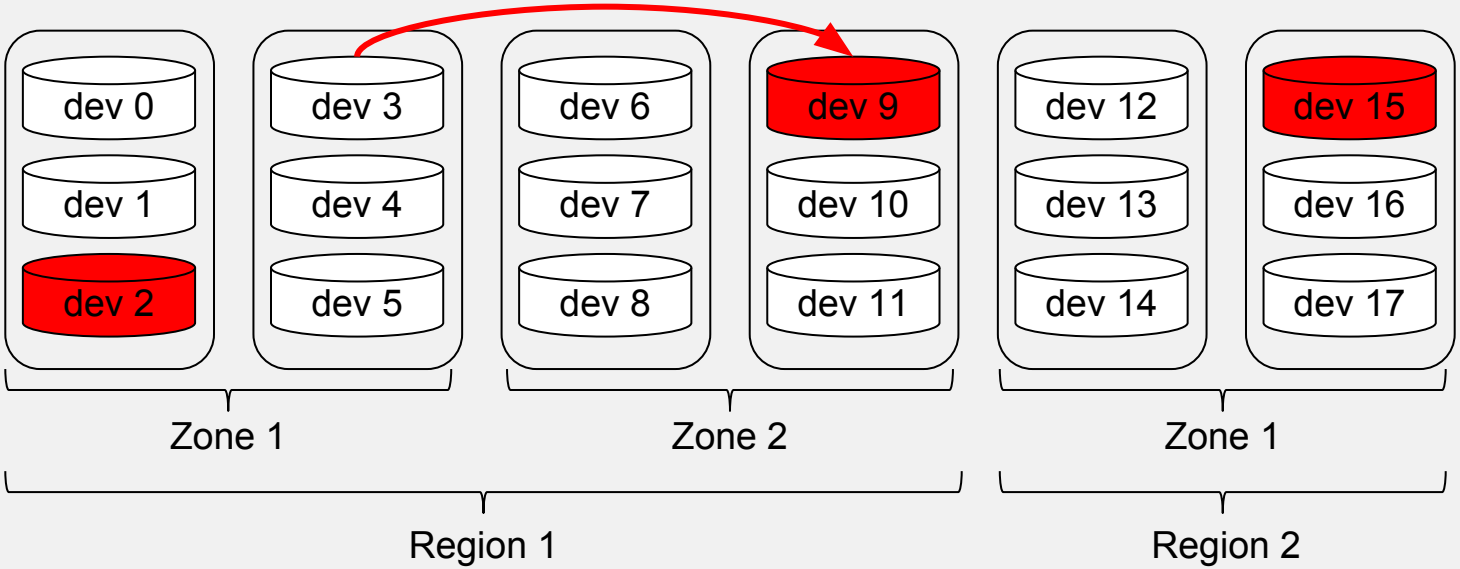
# Failure domains

Ensuring high availability and durability



# Failure domains

Ensuring high availability and durability



# The Rings

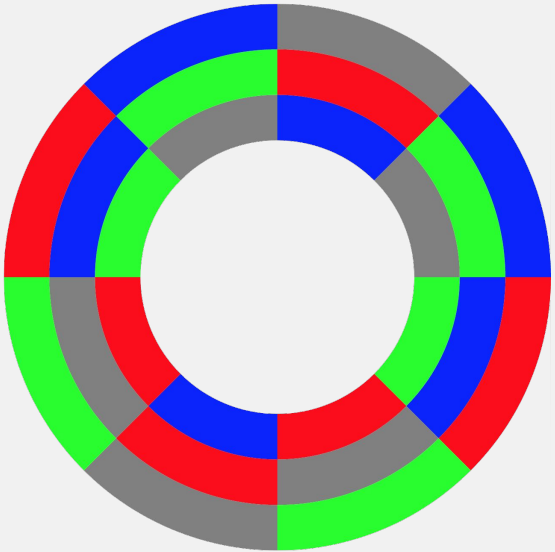
Swift usage of consistent hash rings

- Object name keyspace: hash values of objectnames
- Divide object name keyspace into partitions
- Each partition has multiple replicas
- Each color maps to a device - an actual disk

  => Each disk holds many partitions

  => Only one replica per disk

- Each node keeps local copy of the rings



# Storage Policies

Multiple rings for your objects

- Different data has different storage requirements
  - Backups: cold storage - erasure coded?
  - Videos: cheap SATA disks?
  - Preview Images: less replicas, but fast SSDs?
- Storage Policies make it possible to use different rings based on your needs
  - Defined by an operator, chosen by an user
  - Assigned during container creation

# Eventual consistency

Know what to expect

- **CAP theorem**: **Consistency**, **Availability**, **Partition** tolerance
  - Cluster partitioning: choose two of them
  - Swift' design favors availability and partition tolerance
- Eventual consistency affects you if some or all replicas are down
  - Overwrite existing object: might return older copy
  - Container listings: might not show all objects immediately after upload

# Swift proxy server

# Swift proxy server

Doorway for your data

- WSGI pipeline with selected middlewares
- Authentication: Keystone, tempurl, formpost
- ACLs
- Expiring objects
- Versioning
- Quotas
- Large-object support
- Encryption

# Proxy middlewares

Missing feature: write your own!

- Most features are actually middlewares in the proxy config
- Makes it very easy to extend functionality
  - Write own middleware, install it, add to config
- Most popular: Swift3 middleware
- Elasticsearch, Notification, data processing
- Develop your own middleware: https://www.youtube.com/watch?v=Y7YSuo1iwKU

# Using Swift

# Getting Started

Swift-All-In-One (SAIO): the developers playground

- http://docs.openstack.org/developer/swift/development_saio.html
- https://github.com/redhat-cip/openstack-swift-webapps-examples
- Deploy Swift using Ansible, Vagrant & RDO on Fedora 24
- Ready in a few minutes
- Default Swift user
  - Username: test:tester
  - Password: testing

# REST API

Using the swift CLI

```
swift stat -v

swift post mycontainer

swift upload mycontainer myfirstobject.jpg

Swift list mycontainer

swift stat mycontainer myfirstobject.jpg

swift download mycontainer myfirstobject.jpg
```

# REST API

Using curl and any HTTP client

```
swift download --debug mycontainer myfirstobject.jpg

curl "http://saio:8080/v1/AUTH_test/mycontainer/myfirstobject.jpg" \
     -H "X-Auth-Token: AUTH_tk172d02c6fbbf49749005487c6f524dfb"
```

# Token-less authentication

Using your browser to up- and download data

```
swift post -m temp-url-key:secret

swift tempurl GET 3600 /v1/AUTH_test/mycontainer/myobject.jpg secret

/v1/AUTH_test/mycontainer/myobject.jpg?temp_url_sig=5e47ca9ab3b9824bfeecdd
0edc91f93b8827c50c&temp_url_expires=1475176278
```

# Token-less authentication

Using your browser to up- and download data

```
swift-form-signature /v1/AUTH_test/mycontainer/myobject.jpg "" 1000 1 3600
secret

<form action="/v1/AUTH_test/mycontainer/myobject.jpg" method="POST"
enctype="multipart/form-data">
  <input type="hidden" name="max_file_size" value="10000" />
  <input type="hidden" name="max_file_count" value="5" />
  <input type="hidden" name="expires" value="1475176317" />
  <input type="hidden" name="signature"
value="649170b56e54b88097635f5e607280e21f72c542" />
  <input type="file" name="file0" />
  <br />
  <input type="submit" />
</form>
```

# Demo Time!

# Wrapping Up

# Do’s and don’ts

A few things to keep in mind

- Don't store tens millions of objects in a single container (yet)
  - Might be ok when you're using SSDs for container DBs
- Don't mimic renames using COPY & DELETE
  - Upload once, change metadata in a separate DB
- Keep eventual consistency in mind
  - Container listing might not be updated yet
- Check object metadata on formpost/tempurl uploads
  - Remove X-Delete-At and other metadata if required

redhat.

# THANK YOU!

Mail: cschwede@redhat.com

IRC: cschwede on #openstack-swift (freenode)