삼성 오픈소스 컨퍼런스

SAMSUNG OPEN SOURCE CONFERENCE

OPEN YOUR UNIVERSE WITH SOSCON

오픈스택 swift로 시작하는 오픈소스 분석 삽질기

경희대학교 컴퓨터공학과

조성수

2015.10.28

발표자 소개

조 성 수

- A2 Company 인턴
- 경희대학교 컴퓨터공학과 3학년
- SW Maestro 1기
- 선린인터넷고등학교 졸업

- 관심분야 : 백엔드 개발, Cloud System

오늘 제가 할 이야기는?

지난 1년간의 오픈스택 Swift 와 함께한
'개발'과 관련된 삽질 이야기입니다.

공부 → 사용 → 코드분석 → 개발

발단

2014년 3월. 군대에서 막 전역 후 회사에 인턴으로 들어가게 되었습니다.

들어가자마자 부여받은 업무. ‘Object Storage’를 구축해야한다.

당시 팀에는 저 포함 2명

Openstack Swift에 대해 먼저 공부하자.

우선 공식홈페이지에 있는 문서를 훑어봤습니다

Overview and Concepts ¶

- Object Storage API overview
- Swift Architectural Overview
- The Rings
- Storage Policies
- The Account Reaper
- The Auth System
- Replication
- Rate Limiting
- Large Object Support
- Object Versioning
- Container to Container Synchronization
- Expiring Object Support
- CORS
- Cross-domain Policy File
- Erasure Code Support
- Using Swift as Backing Store for Service Data
- Associated Projects

지금와서 보면 정말 정리가 잘 되어있는 1등급 문서

하지만 당시에는...

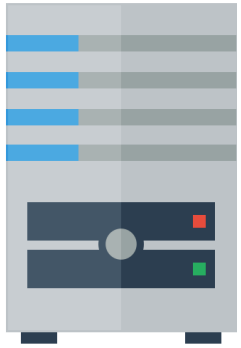
http://hamnice51.tistory.com/entry/파워패스로-하기-싫은-공부-시작하기-ㅠㅠ

거대한 시스템을 다루기 위해서 개념에 대한 이해가 필요하다 생각해서

계속 문서만 열심히 읽었습니다

일단 써보자

Openstack Swift 의 구성 요소

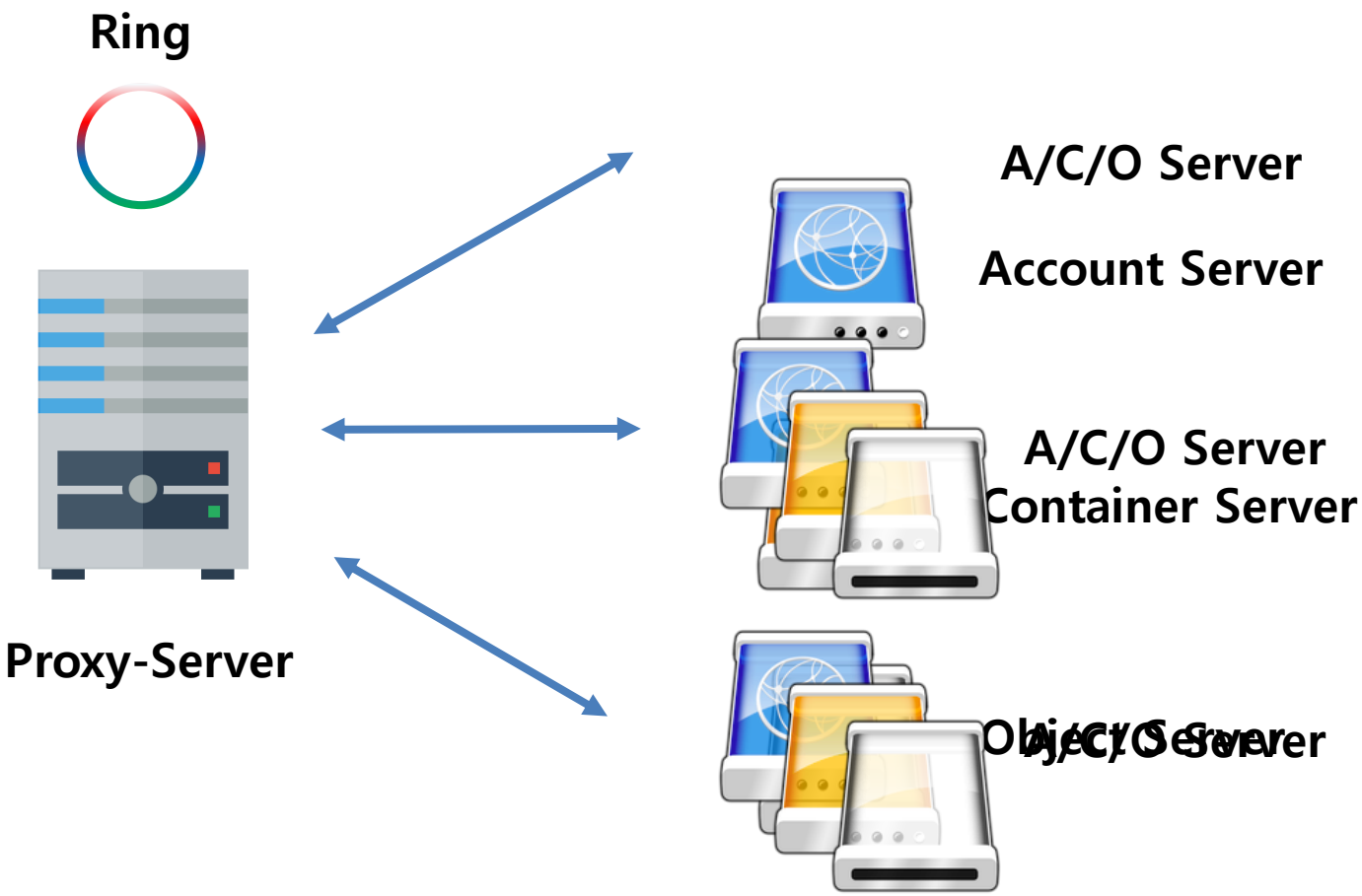


X 4

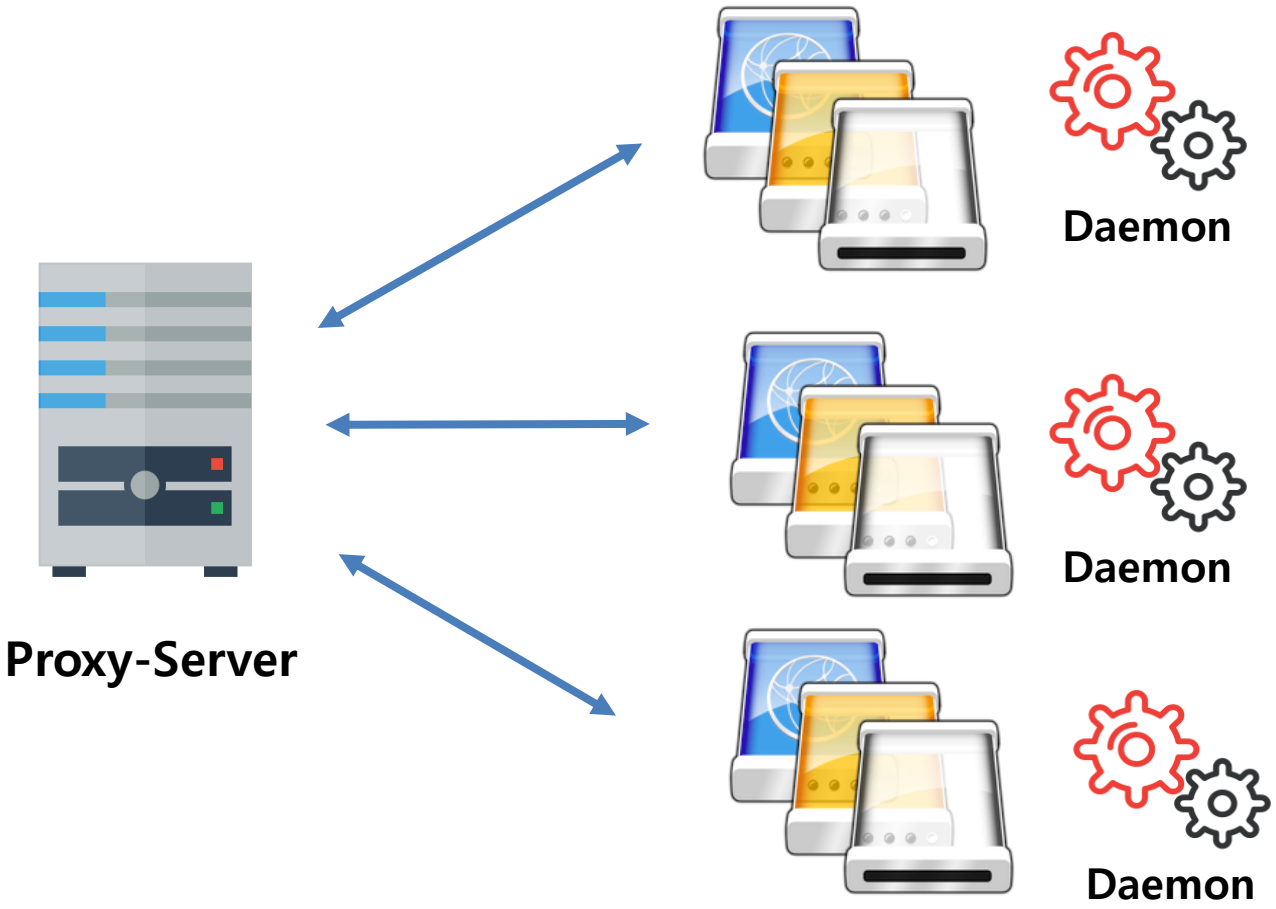
X 6

Daemon

Openstack Swift 의 구성 요소



Openstack Swift 의 구성 요소



Python

Swift 를 써보려고만해도 서버 10대가 필요했습니다

A/C/O Server 를 서버 한 대로 구성해도, 4대의 서버가 필요

당시 저는 MacBook 13 inch Mid 2010 을 쓰고 있었습니다.

Intel Core 2 Duo

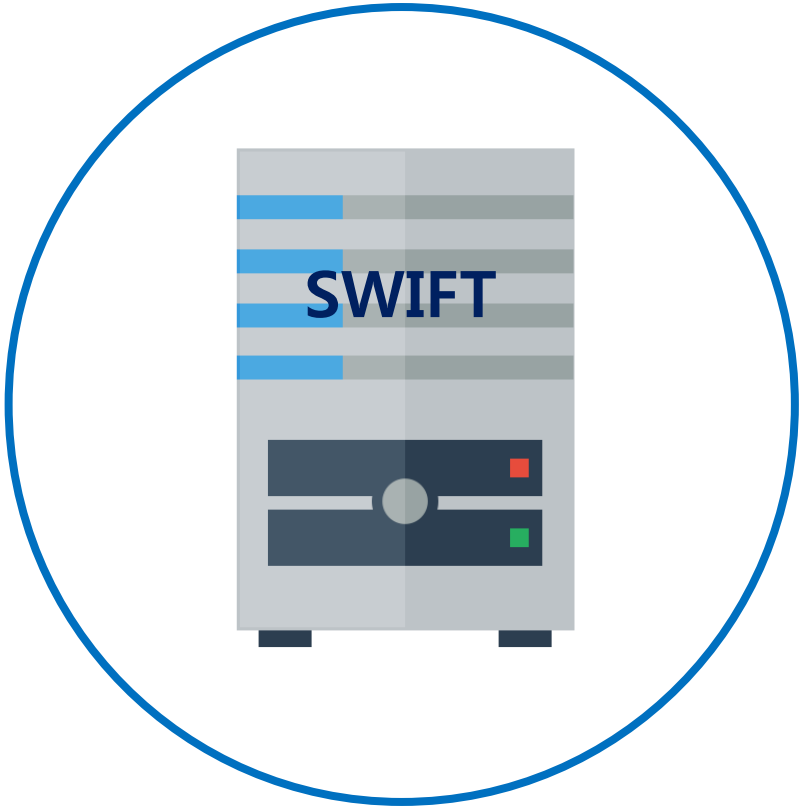
4GB RAM

VM 4개를 돌리기엔

맥북이 힘들어합니다

처음 프로젝트 설명을 들으면서 **SAIO**라는 것이 있다고 들었습니다

SAIO : Swift All In One



Developer Documentation

- Development Guidelines
- SAIO - Swift All In One
- First Contribution to Swift
- Adding Storage Policies to an Existing SAIO
- Auth Server and Middleware
- Middleware and Metadata
- Pluggable On-Disk Back-end APIs

SAIO - Swift All In One

Instructions for setting up a development VM

This section documents setting up a virtual machine for doing Swift development. The virtual machine will emulate running a four node Swift cluster. To begin:

- Get an Ubuntu 14.04 LTS server image or try something Fedora/CentOS.
- Create guest virtual machine from the image.

Using a loopback device for storage

If you want to use a loopback device instead of another partition, follow these instructions:

1. Create the file for the loopback device:

```
sudo mkdir /srv
sudo truncate -s 1GB /srv/swift-disk
sudo mkfs.xfs /srv/swift-disk
```

   Modify size specified in the `truncate` command to make a larger or smaller partition as needed.

2. Edit */etc/fstab* and add:

```
/srv/swift-disk /mnt/sdb1 xfs loop,noatime,nodiratime,nobarrier,l
```

3. Create the mount point and the individualized links:

```
sudo mkdir /mnt/sdb1
sudo mount /mnt/sdb1
sudo mkdir /mnt/sdb1/1 /mnt/sdb1/2 /mnt/sdb1/3 /mnt/sdb1/4
sudo chown ${USER}:${USER} /mnt/sdb1/*
for x in {1..4}; do sudo ln -s /mnt/sdb1/$x /srv/$x; done
sudo mkdir -p /srv/1/node/sdb1 /srv/1/node/sdb5 \
              /srv/2/node/sdb2 /srv/2/node/sdb6 \
              /srv/3/node/sdb3 /srv/3/node/sdb7 \
              /srv/4/node/sdb4 /srv/4/node/sdb8 \
              /var/run/swift
sudo chown -R ${USER}:${USER} /var/run/swift
# **Make sure to include the trailing slash after /srv/$x/**
for x in {1..4}; do sudo chown -R ${USER}:${USER} /srv/$x/; done
```

Getting the code

1. Check out the python-swiftclient repo:

```
cd $HOME; git clone https://github.com/openstack/python-swiftclie
```

2. Build a development installation of python-swiftclient:

```
cd $HOME/python-swiftclient; sudo python setup.py develop; cd -
```

Ubuntu 12.04 users need to install python-swiftclient's dependencies before the installation of python-swiftclient. This is due to a bug in an older version of setup tools:

```
cd $HOME/python-swiftclient; sudo pip install -r requirements.txt
```

3. Check out the swift repo:

```
git clone https://github.com/openstack/swift.git
```

4. Build a development installation of swift:

```
cd $HOME/swift; sudo pip install -r requirements.txt; sudo python
```

Fedora 19 or later users might have to perform the following if development installation of swift fails:

```
sudo pip install -U xattr
```

어마어마한 설정 방법


INTRODASTING

One Click, Install Openstack

1. devstack 스크립트를 다운로드 받는다.

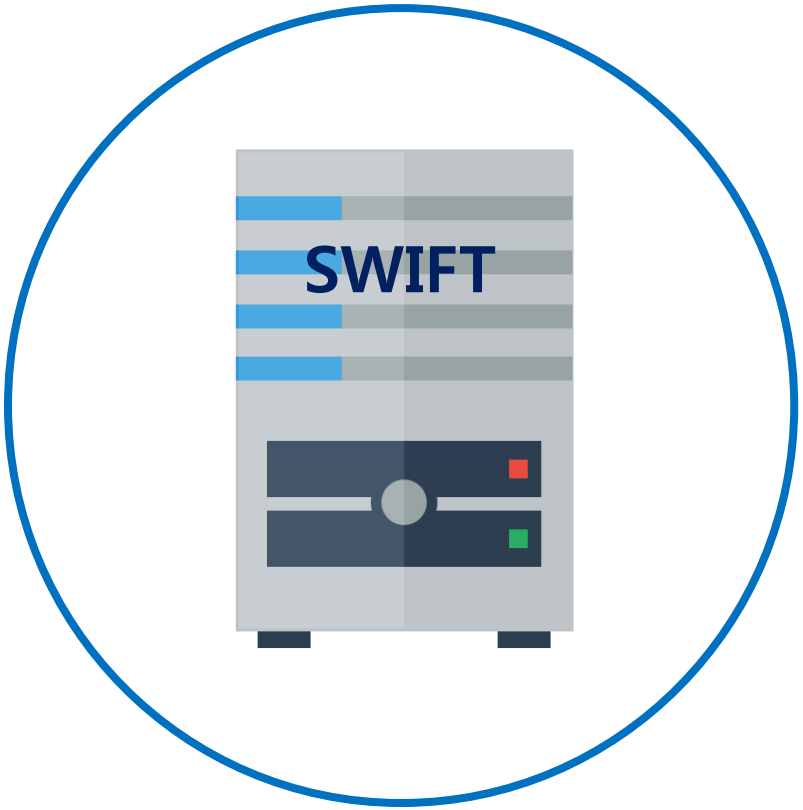
git clone https://git.openstack.org/openstack-dev/devstack

2. swift 관련 간단한 설정을 해준다.

3. 스크립트를 실행한다

cd devstack; ./stack.sh

최소 2core, 2GB VM 으로 간단하게 swift 환경 구축



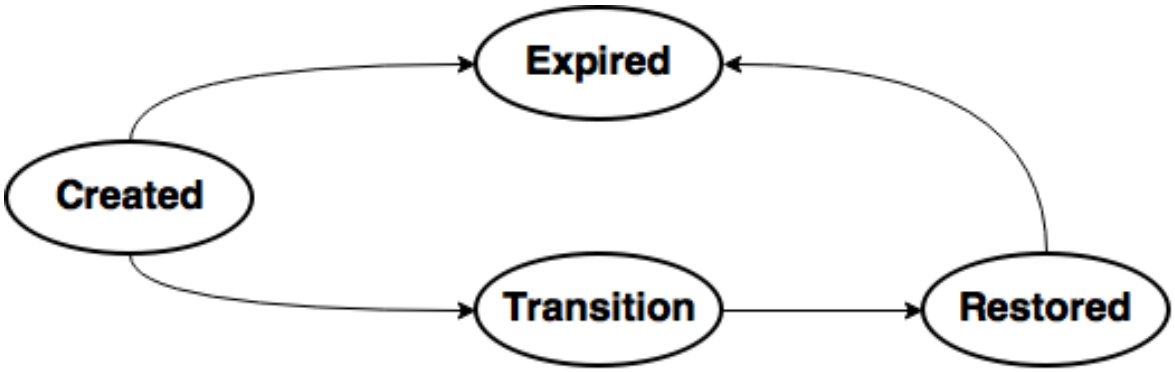

HELL YEAH!

< 새로운 임무 >

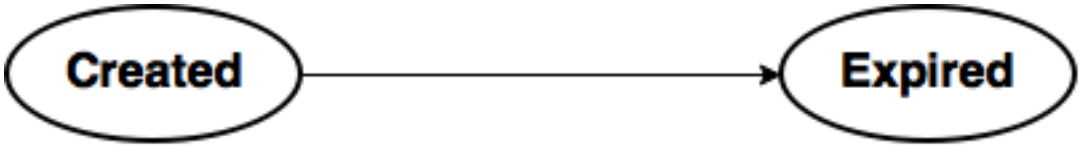
추가 기능 개발

< 추가 기능 >

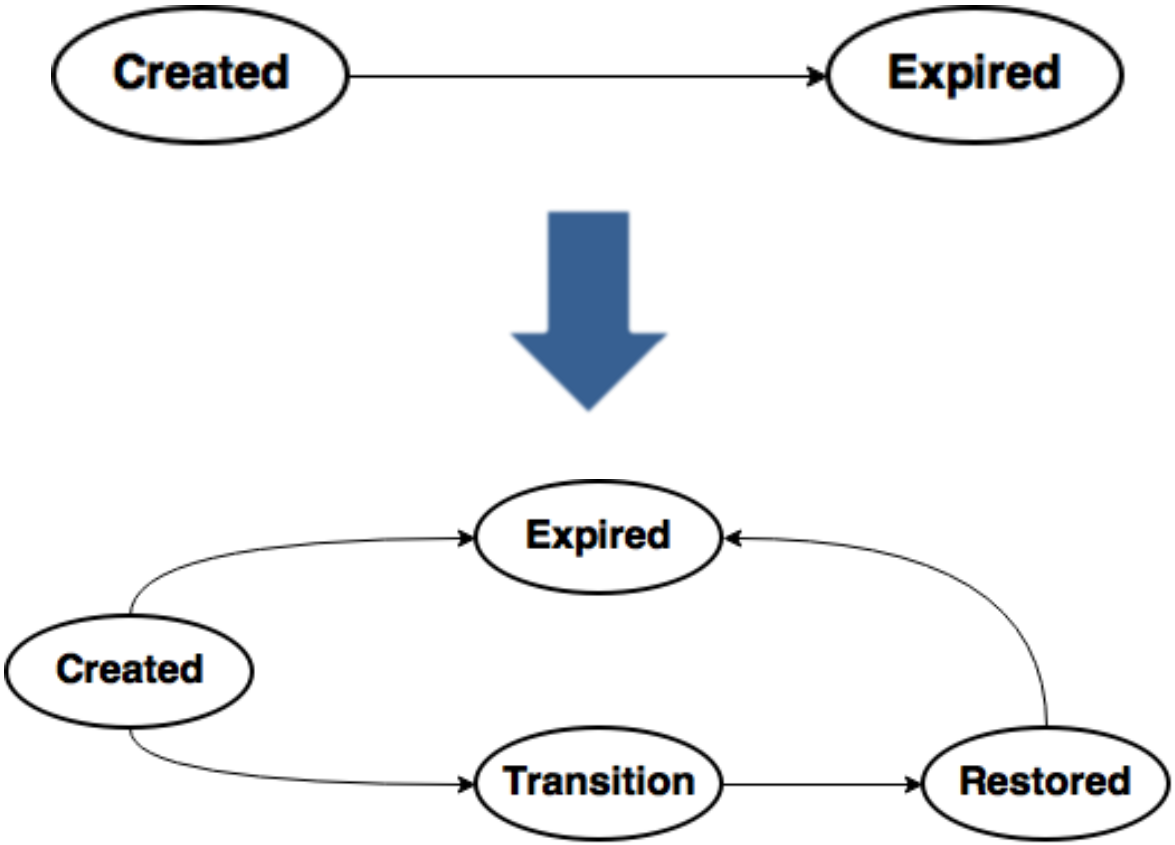
Swift에 Object Lifecycle 기능 추가



사실, Swift에도 object lifecycle 이 있습니다.



Swift에 Object Lifecycle 을 더하다



이 모든것을

Swift 원본 코드 수정 없이!

Swift에 Object Lifecycle 을 더하다

마치 3rd party 처럼 Swift의 부가기능으로

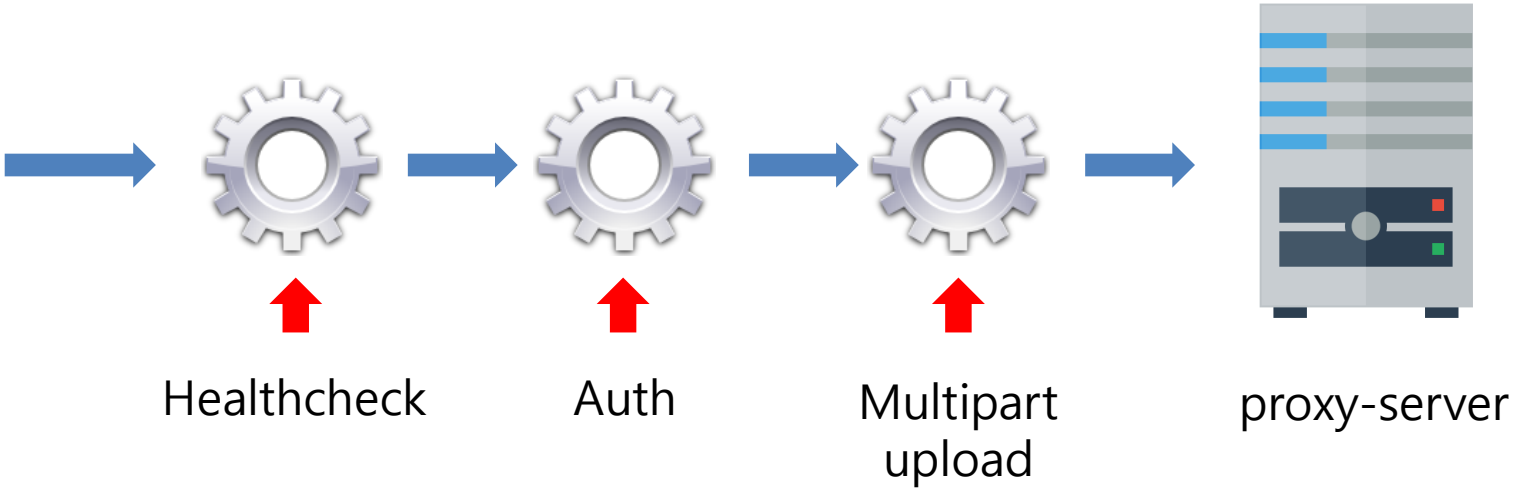
Object Lifecycle을 구현해야한다

Swift

+

Object Lifecycle

이미 Swift에는 부가기능을 하는 Middleware 라는 개념이 있습니다



Developer Documentation

- Development Guidelines
- SAIO - Swift All In One
- First Contribution to Swift
- Adding Storage Policies to an Existing SAIO
- Auth Server and Middleware
- Middleware and Metadata
- Pluggable On-Disk Back-end APIs

Creating Your Own Middleware

The best way to see how to write middleware is to look at examples.

Many optional features in Swift are implemented as *Middleware* and provided in `swift.common.middleware`, but Swift middleware may be packaged and distributed as a separate project. Some examples are listed on the *Associated Projects* page.

A contrived middleware example that modifies request behavior by inspecting custom HTTP headers (e.g. X-Webhook) and uses *System Metadata* to persist data to backend storage as well as common patterns like a **`get_container_info()`** cache/query and **`wsgify()`** decorator is presented below:

Middleware 개발

```
class WebhookMiddleware(object):

    def __init__(self, app, conf):
        self.app = app
        self.logger = get_logger(conf, log_route='webhook')

    @wsgify
    def __call__(self, req):
        obj = None
        try:
            (version, account, container, obj) = \
                split_path(req.path_info, 4, 4, True)
        except ValueError:
            # not an object request
            pass
        if 'x-webhook' in req.headers:
            # translate user's request header to sysmeta
            req.headers[SYSMETA_WEBHOOK] = \
                req.headers['x-webhook']
        if 'x-remove-webhook' in req.headers:
            # empty value will tombstone sysmeta
            req.headers[SYSMETA_WEBHOOK] = ''
        # account and object storage will ignore x-container-sysmeta-*
        resp = req.get_response(self.app)
        if obj and is_success(resp.status_int) and req.method == 'PUT':
            container_info = get_container_info(req.environ, self.app)
            # container_info may have our new sysmeta key
            webhook = container_info['sysmeta'].get('webhook')
            if webhook:
                # create a POST request with obj name as body
                webhook_req = urllib2.Request(webhook, data=obj)
                with Timeout(20):
                    try:
                        urllib2.urlopen(webhook_req).read()
                    except (Exception, Timeout):
                        self.logger.exception(
                            'failed POST to webhook %s' % webhook)
                    else:
                        self.logger.info(
                            'successfully called webhook %s' % webhook)
        if 'x-container-sysmeta-webhook' in resp.headers:
            # translate sysmeta from the backend resp to
            # user-visible client resp header
            resp.headers['x-webhook'] = resp.headers[SYSMETA_WEBHOOK]
        return resp


def webhook_factory(global_conf, **local_conf):
    conf = global_conf.copy()
    conf.update(local_conf)
    def webhook_filter(app, conf):
        return WebhookMiddleware(app)
```

In practice this middleware will call the url stored on the container as X-Webhook on all successful object uploads.

If this example was at `<swift-repo>/swift/common/middleware/webhook.py` - you could add it to your proxy by creating a new filter section and adding it to the pipeline:

```
[DEFAULT]
log_level = DEBUG
user = <your-user-name>

[pipeline:main]
pipeline = healthcheck webhook proxy-server

[filter:webhook]
paste.filter_factory = swift.common.middleware.webhook:webhook_factory

[filter:healthcheck]
use = egg:swift#healthcheck

[app:proxy-server]
use = egg:swift#proxy
```

Most python packages expose middleware as entrypoints. See PasteDeploy documentation for more information about the syntax of the `use` option. All middleware included with Swift is installed to support the `egg:swift` syntax.


INTREDASTING

In practice this middleware will call the url stored on the container as X-Webhook on all successful object uploads.

If this example was at `<swift-repo>/swift/common/middleware/webhook.py` you could add it to your proxy by creating a new filter section and adding it to the pipeline:

```
[DEFAULT]
log_level = DEBUG
user = <your-user-name>

[pipeline:main]
pipeline = healthcheck webhook proxy-server

[filter:webhook]
paste.filter_factory = swift.common.middleware.webhook:webhook_factory

[filter:healthcheck]
use = egg:swift#healthcheck

[app:proxy-server]
use = egg:swift#proxy
```

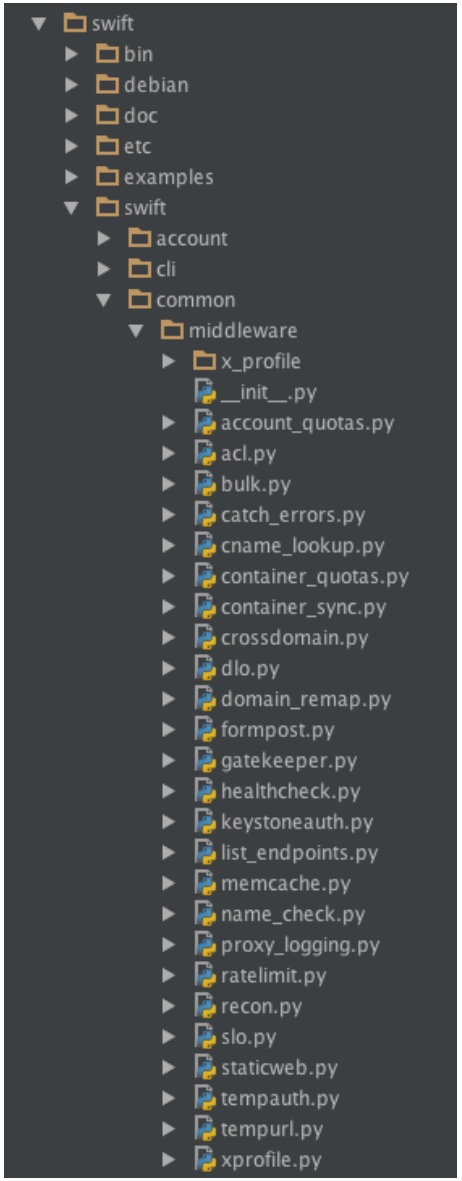
Most python packages expose middleware as entrypoints. See PasteDeploy documentation for more information about the syntax of the `use` option. All middleware included with Swift is installed to support the `egg:swift` syntax.

Middleware 개발

공식홈페이지의 예제는 빈약했지만

Swift 코드 내에 참고할만한 많은

Middleware 코드가 있었습니다



Middleware 분석

공식 홈페이지의 Middleware 소개 문서를 보고, 제일 쉬울만한 것을 하나 골라 코드 분석을 시작했습니다

Middleware

Account Quotas

`account_quotas` is a middleware which blocks write requests (PUT, POST) if a given account quota (in bytes) is exceeded while DELETE requests are still allowed.

`account_quotas` uses the `x-account-meta-quota-bytes` metadata entry to store the quota. Write requests to this metadata entry are only permitted for resellers. There is no quota limit if `x-account-meta-quota-bytes` is not set.

The `account_quotas` middleware should be added to the pipeline in your `/etc/swift/proxy-server.conf` file just after any auth middleware. For example:

```
[pipeline:main]
pipeline = catch_errors cache tempauth account_quotas proxy-server

[filter:account_quotas]
use = egg:swift#account_quotas
```

To set the quota on an account:

```
swift -A http://127.0.0.1:8080/auth/v1.0 -U account:reseller -K secret post -m qu
```

눈으로

코드를 한줄 한줄

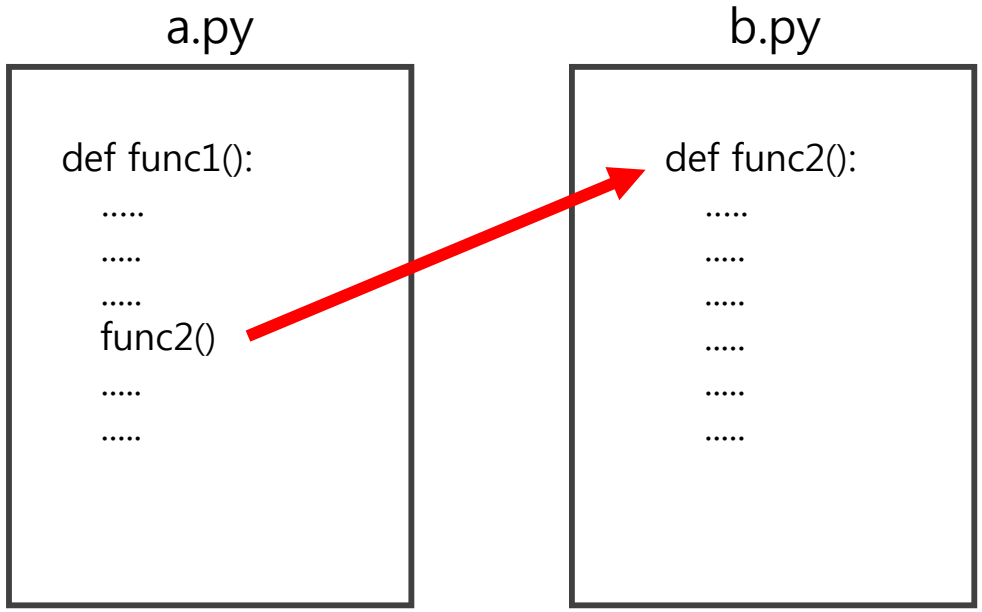
분석했습니다

눈으로

분석 하는데 한계가 생겼습니다.

사실 PyCharm 이라는 IDE를 쓰고 있어서 분석은 용이했습니다.

예를 들면, 코드 reference 따라가기 / 되돌아오기



어렴풋이 어떻게 동작하는지는 알겠는데..

변수에 어떤 값이 들어가고, 어떻게 값이 변화하는지를 모르니 완벽하게 이해하기 어려웠습니다

```
def __call__(self, env, start_response):
    req = Request(env)
    removed = remove_items(req.headers, self.inbound_condition)
    if removed:
        self.logger.debug('removed request headers: %s' % removed)

    def gatekeeper_response(status, response_headers, exc_info=None):
        removed = filter(
            lambda h: self.outbound_condition(h[0]),
            response_headers)
        if removed:
            self.logger.debug('removed response headers: %s' % removed)
            new_headers = filter(
                lambda h: not self.outbound_condition(h[0]),
                response_headers)
            return start_response(status, new_headers, exc_info)
        return start_response(status, response_headers, exc_info)
    return self.app(env, gatekeeper_response)
```

변수값을 확인하고 싶다

실행할 수 있는 환경은 이미 있다

간단하지만 만능의 디버깅도구 - print

원하는 곳에 print [변수] 만 하면 언제든지 변수 값 확인이 가능합니다.

나름 체계화한다고 print [식별자] [변수]

Middleware 분석

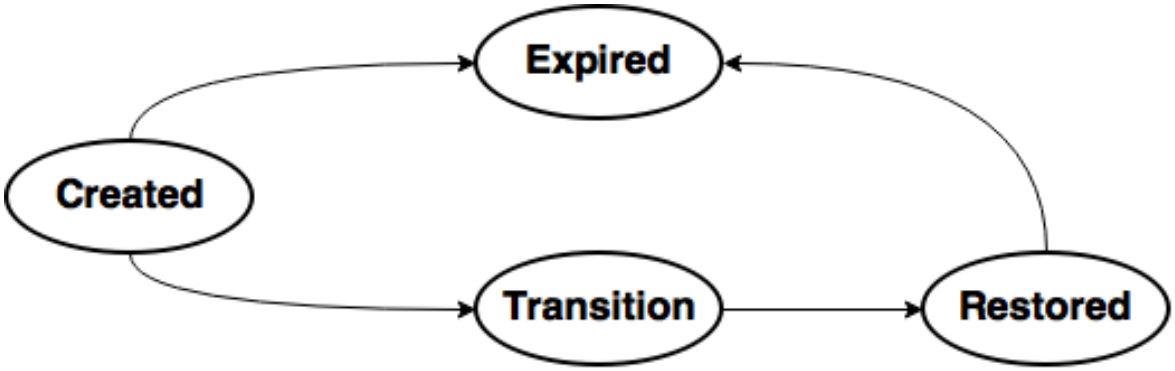
실상은 눈알 빠지게 PRINT 된 내용을 찾아야하지만,

그럭저럭 할만했습니다

```
proxy-server: Authenticating user token
proxy-server: Invalid user token - deferring reject downstream
proxy-server: Authenticating service token
proxy-server: Received request from
proxy-server: Authorizing from an overriding middleware (txn: tx408905e8347848bb804db-00561edf09)
proxy-server: STDOUT: Hello World!! (txn: tx408905e8347848bb804db-00561edf09)
proxy-server: - - 14/Oct/2015/23/02/33 HEAD /v1/AUTH_5d8f8a9c6f714c31bc5a760c6fb1071c HTTP/1.0 204 - Swift - - - - tx408905e8347848bb804db-00561edf0
9 - 0.0023 RL - 1444863753.181557894 1444863753.183897972 -
proxy-server: Authenticating user token
proxy-server: Storing token in cache
proxy-server: Authenticating service token
proxy-server: Received request from user: user_id 27c9a13f01fb4906b9bff1cc10f4bc2b, project_id 5d8f8a9c6f714c31bc5a760c6fb1071c, roles Member,_membe
r_
proxy-server: Using identity: {'roles': [u'_member_', u'Member'], 'user': u'nexusz99', 'tenant': (u'5d8f8a9c6f714c31bc5a760c6fb1071c', u'nexusz99')}
 (txn: tx40[illegible])
proxy-server: STDOUT: Hello World!! (txn: tx408905e8347848bb804db-00561edf09)
proxy-server: [illegible] admin (txn: tx408905e8347848bb804db-00561edf09) (client_ip: 127.0.0.1)
proxy-server: 127.0.0.1 127.0.0.1 14/Oct/2015/23/02/33 GET /v1/AUTH_5d8f8a9c6f714c31bc5a760c6fb1071c HTTP/1.0 204 - curl/7.35.0 cd0197188727... - -
- tx408905e8347848bb804db-00561edf09 - 0.6387 - - 1444863753.179616928 1444863753.818366051 -
```

Middleware 분석이 어느정도 되고나니 개발에 감이 잡혔습니다

근데 제가 개발해야하는 것이...



Swift 내부 원리를 이해하지 못하면, 구현하지 못하는 기능

결국 Swift 코드도 분석하고 디버깅하기 시작했습니다.

http://m.bdkrnews.com/bbs/board.php?bo_table=good_02&wr_id=8887

Print , 즉 로그 분석만으로는 감당하기 힘든 스케일.

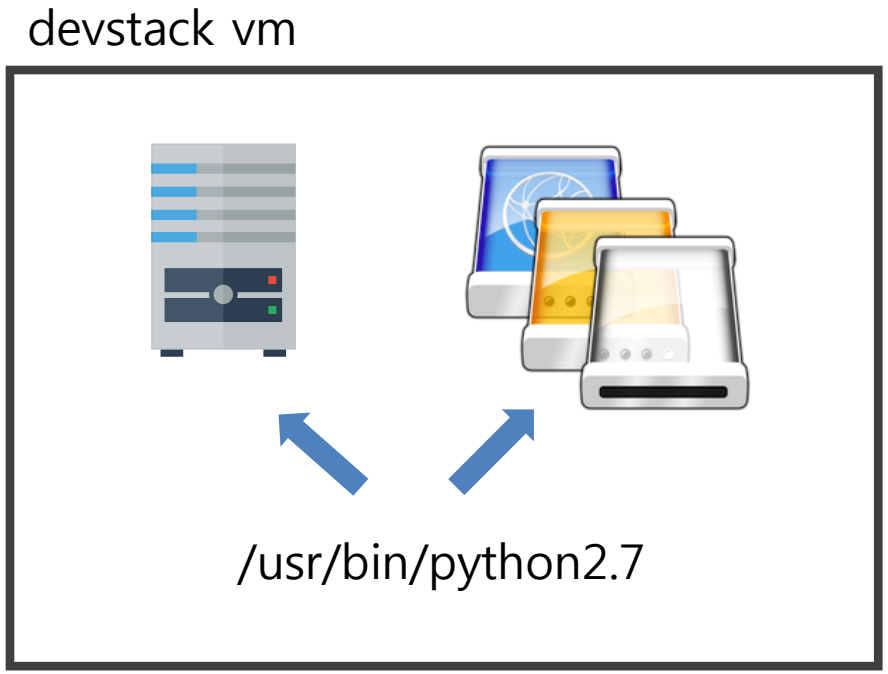
IDE를 쓰는데 IDE를 쓰는게 아닌 느낌..

뭔가 다른 방법이 필요하다

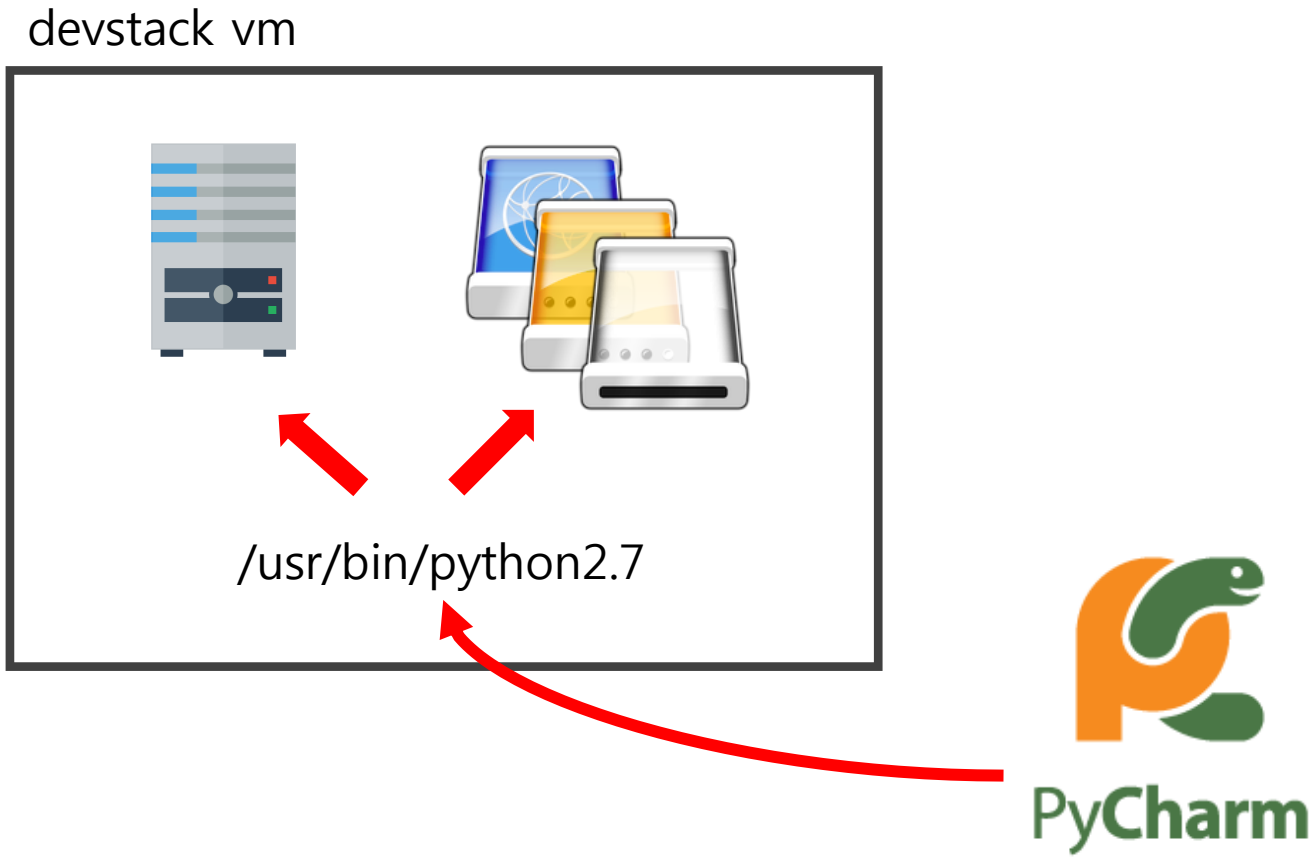
Remote Debugging with Remote Interpreter

현재 Swift 서버가 실행중인 모습

devstack vm안에서 직접 실행되어있다.



Pycharm이 vm의 Python Interpreter 를 직접 실행하도록 설정



IDE를 IDE답게 사용할 수 있게 되었다!!

STEP INTO / OVER

변수 여러개의 상태를 동시에 확인

심지어 변수값도 실행중에 바꿀 수 있다.


HELL YEAH!


PyCharm

그래서

약 6개월 동안

하얗게 불태웠어

Swift 코드 분석도 하고

Object Lifecycle 도 개발하고

Middleware : 4개

Daemon : 5개

만들어 냈습니다

코드는..

http://toothfairy.tistory.com/467

Swift 코드 분석 과정

1. API 의 흐름 파악

API를 하나하나 호출해가면서 Swift 코드의 어느 부분을 거쳐가는지 대략적으로 파악

API 흐름에 있는 코드만 제대로 분석해도 Swift 의 절반을 알 수 있다.

나머지는 50%는 Consistency 유지를 위한 Daemon과, 유틸리티

2. 기능의 개발 과정을 역추적

지금 내가 보고 있는 이 코드는 수 많은 고민과 토론을 통해 나온 코드

처음에 코드만 봐서는 왜 이렇게 처리했는지 파악이 잘 안될 수 있다

커밋로그, 코드리뷰 사이트를 통해 기능 개발 과정을 역추적하며 파악

2. 기능의 개발 과정을 역추적

openstack CLOUD SOFTWARE

All Projects Documentation
Open Merged Abandoned

Change-Id:	I5619b045e4e741f98c2fa7564a4de30e510f39eb
Owner	Samuel Merritt
Project	openstack/swift
Branch	master
Topic	ratelimit
Uploaded	Sep 12, 2015 2:51 AM
Updated	Sep 29, 2015 1:10 AM
Status	**Abandoned**

Commit Message *Permalink*

```
Improve SLO PUT error checking

Previously, if you uploaded a segment reference with '{"size_bytes":
null, ...}' in it and the referenced segment was less than 1 MiB,
you'd get a response that looked like this:

    HTTP/1.1 400 Bad Request
    Content-Length: 62
    Content-Type: text/html; charset=UTF-8
    X-Trans-Id: txd9ee3b25896642098e4d9-0055dd095a
    Date: Wed, 26 Aug 2015 00:33:30 GMT

    Each segment, except the last, must be at least 1048576 bytes.

This is true, but not particularly helpful, since it doesn't tell you
which of your segments violated the rule.

Now you get something more like this:

    HTTP/1.1 400 Bad Request
    Content-Length: 49
    Content-Type: text/plain
    X-Trans-Id: tx586e52580bac4956ad8e2-0055dd09c2
    Date: Wed, 26 Aug 2015 00:35:14 GMT

    Errors:
    /segs/small, Too Small (minimum 1048576B)

It's not exactly a tutorial on SLO manifests, but at least it names
the problematic segment.

I also split the error checking into two steps: first, SLO does all
the checks it can without making any subrequests. If there are errors
in that first pass, like a size_bytes that's *specified* as too small
or a self-referential manifest, then SLO returns those errors. This
way we don't waste time doing a bunch of HEAD requests for something
we already know is bogus. Second pass is all the
HEAD-request-and-check-matching stuff that SLO already does.

Change-Id: I5619b045e4e741f98c2fa7564a4de30e510f39eb
```

3. UML을 찾기란 하늘의 별따기

ERD, Flowchart 등 UML은 의외로 없었다.

기능을 제안한 사람의 github, blog에만 간단히 남아있는 정도

운이 좋으면 IRC 로그에서 다이어그램 링크를 획득할 수도 있다.

4. Openstack Swift Contributor 들의 작업 과정 추적

의외로 공식 문서를 뒤적이다보면 유용한 링크가 많이 나온다

프로젝트 대시보드, 아이디어목록, 우선순위 별 리뷰목록 등..

하나하나 들어가서 구경하면서, 어떤식으로 작업이 이루어지는지 파악

4. Openstack Swift Contributor 들의 작업 과정 추적

새로운 기능

Blueprint -> 검토 -> 승인 -> 개발 -> 코드리뷰 -> 반영

Spec제안 -> 검토 -> 승인 -> 개발 -> 코드리뷰 -> 반영

4. Openstack Swift Contributor 들의 작업 과정 추적

Comment 를 통해 어디서 해당 기능이 토의되었는지 적어주는 경우도 있다

```
Original discussion was in working session at Vancouver Summit.
Etherpads are around here:

https://etherpad.openstack.org/p/liberty-swift-contributors-meetup
https://etherpad.openstack.org/p/liberty-container-listing-update

TODO:
Add response example to https://github.com/openstack/api-site

DocImpact

Change-Id: Iba0503916f1481a20c59ae9136436f40183e4c5b
```

4. Openstack Swift Contributor 들의 작업 과정 추적

Container Listing (GET account) Update (Support Last Modified timestamp)

Description: Currently swift returns "count", "bytes" and "name" attributes for each container when container listing (with json/xml format) requ
container table on account db) to the response as well as object listing (GET container)

Purpose:

To make retrieving container metadata effective.

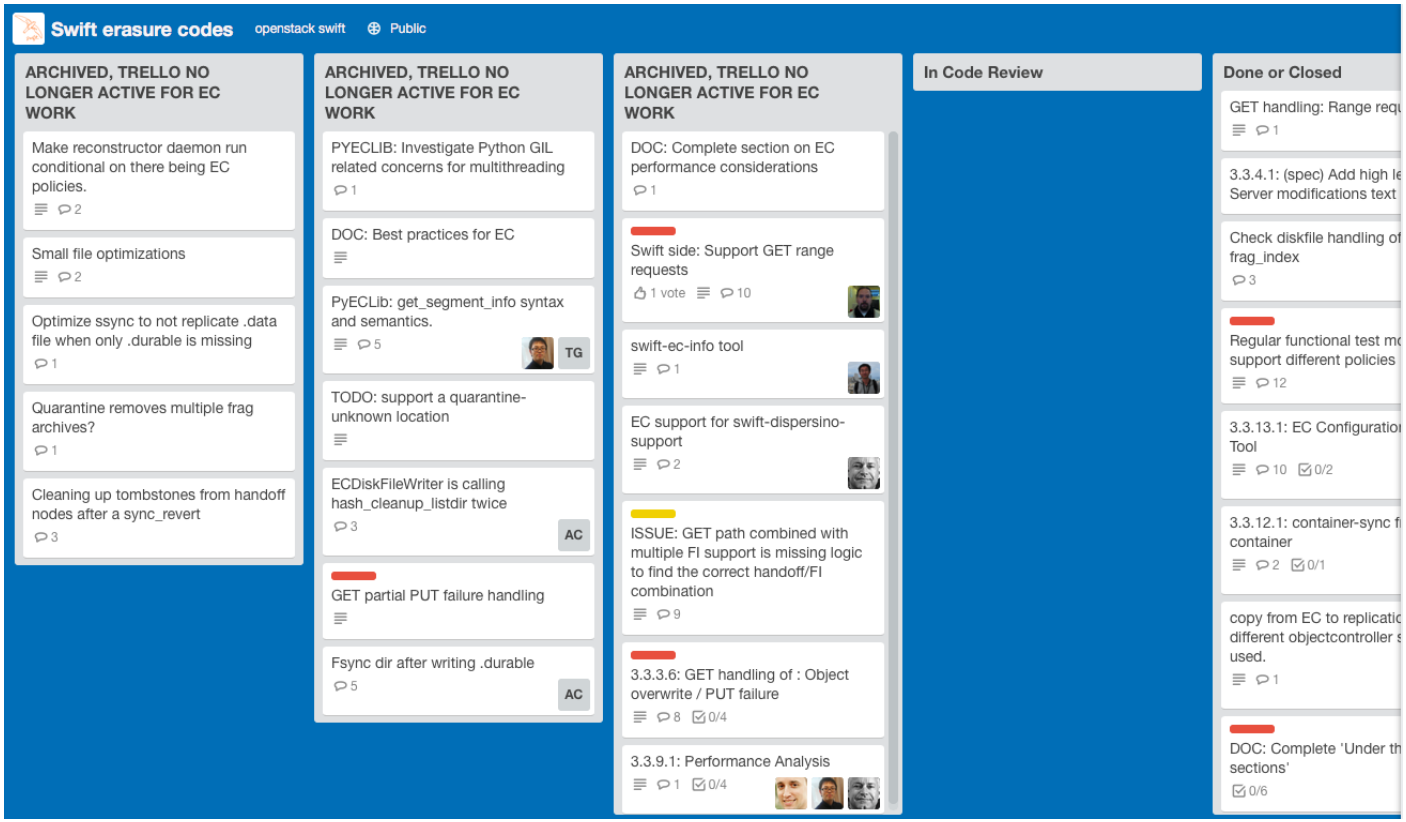
Use Case:

Imagine retrieving the metadata (e.g. x-container-meta-xxx, acl or so on) PERIODICALLY against to whole exisiting containers. Currently w
the metadata. Assuming a lot of containers (e.g. >1000) exist in the Swift and almost of containers DOES NOT be updated so frequently, if each
are same as the one retrieved before.

If we keep the previous result (something like as a local cache) and there is the last modified timestamp in the GET account response, wha
container's last modified timestamp and making HEAD requests for the containers which were updated . (i.e. if only one container updated sinc

4. Openstack Swift Contributor 들의 작업 과정 추적

공식 문서에서는 볼 수 없던, 기능 개발 Trello 사이트 발견



5. 모르는건 IRC에 질문

```
2014-08-13T08:02:49  <nexusz99> Question! In DLO middleware, the segments are over 10ea, the segment container listing is weird.

2014-08-13T09:36:20  <mattoliverau> nexusz99: listing our of order is probably because the list is being sorted alphabetically. So 1 and 10 are before 2 and 21 etc
2014-08-13T09:36:38  <mattoliverau> *out of order
```

이상하게 말해도 찰떡같이 알아들으니, 철판깔고 질문을 하면 됩니다

기능하나 개발하려다

Swift를 분석해버리고 말았습니다

정리

오픈소스 라이브러리던, 시스템이던

일단 써본다

공식 문서를 꼼꼼히 읽어가면서

기능, 구조를 명확히 파악하고

디버깅 가능한 환경을 구축하고

디버깅을 하며 코드를 분석한다

IRC와 같은 개발자 채널이 있다면

무작정 들어가서 어떤 이야기가 오고가는지 알아보고

어떻게 개발이 이루어지는지 파악한다

그리고 현재 진행중인 논의가 무엇인지도..

그러다보면 어느새 버그리포팅도 하고

Contribution도 하는 날이 오지않을까요?

긴 시간 저의 삽질 이야기를 들어주셔서

대단히 감사합니다

휴.. 끝났다

짤방은 구글링으로 찾았고, 사진아래 출처를 달아뒀습니다

짤방을 제외한 아이콘은 IconFinder 에서 가져왔습니다.

그외 그림은 직접 캡쳐했습니다


SOSCON
삼성 오픈소스 컨퍼런스
SAMSUNG OPEN SOURCE CONFERENCE
OPEN YOUR UNIVERSE WITH SOSCON

THANK YOU!