OSC2012 Tokyo/Spring

openstack JAPAN

openstack

Open source software to build public and private clouds.

Object Storage; Usage

～オブジェクトストレージはこう使え!～

2012.03.17

日本 OpenStack ユーザ会

@irix_jp

- ネタが豊富な Nova に比べ Swift はネタが無い・・・

Nova

自主規制

お寿司

Swift

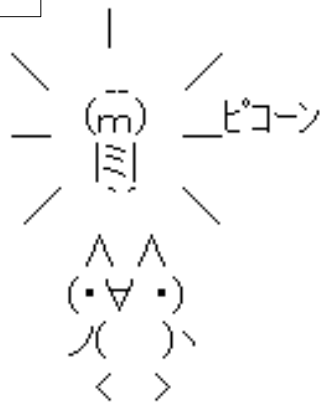
自主規制

卵かけご飯

次は Glance の話も一緒にしよう。

Glance

自主規制
たくあん



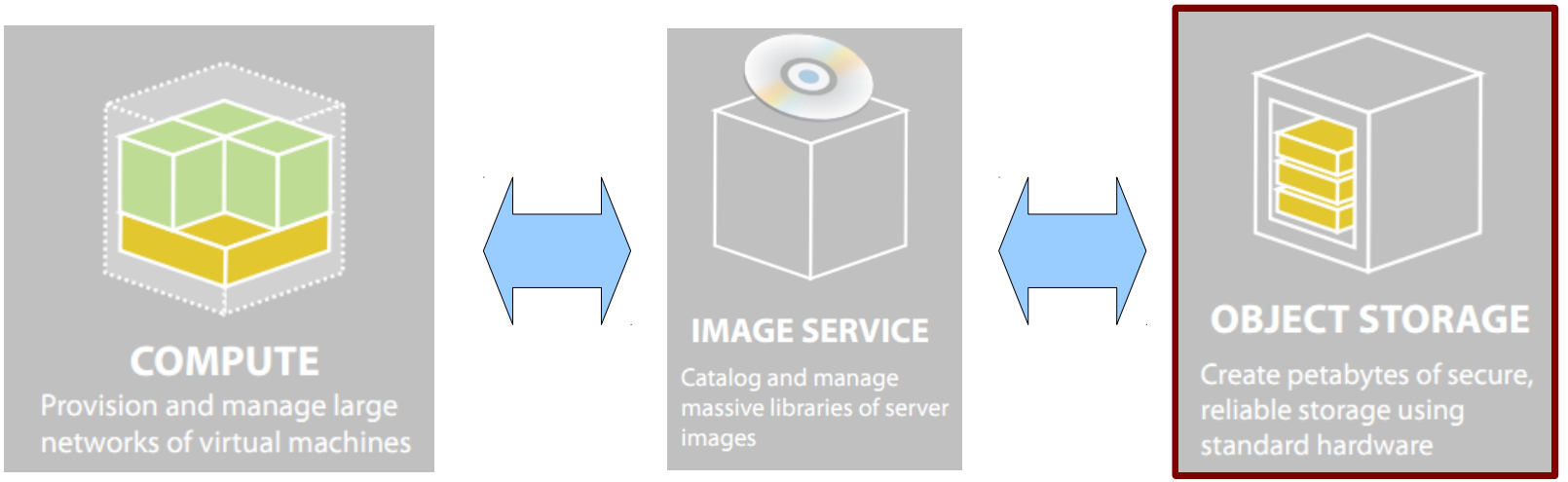
Agenda

- Swift って何？
- Swift が注目されるワケ
- Swift を使う上で抑えておくポイント
- 使い所
- ニュース
- まとめ

Swiftって何？

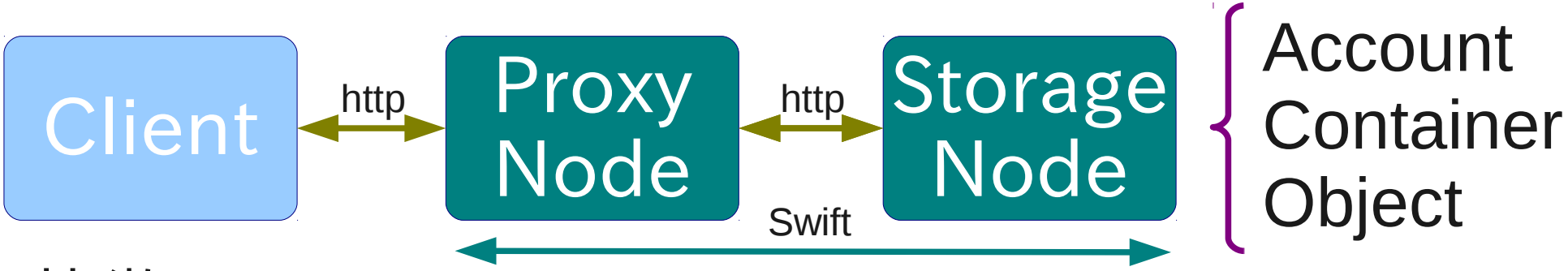
Swiftって何？

- OpenStackの一部でObject Storage機能を担当
- Amazon S3相当（互換API あり）
- 普段はGlanceと連携して、Novaが使う仮想マシンイメージの保存先として動作。



Swift って何？

- Swift の実態はシンプルなファイルサーバ



- 特徴
  - 安価なハードウェアで安全にファイル保存が可能
    - デフォルトで 3 つのレプリカを作成
    - 強力な自己修復機能
  - HTTP(REST) でファイルの入出力＆操作
  - 容量と性能がリニアにスケールし、単一障害点無し
  - シングルネームスペースで数百 PB を管理

Swift って何？

- 高い完成度と実績
  - 商用サービス（ Rackspace Cloud Files ）をベースとして OSS 化されているため OpenStack の中でも相対的に完成度が高い。
  - 既に大規模環境下での運用実績
    - 1PB（ KR ）
    - 5.5PB（ US ）
    - Etc...
  - Swift 単体でも利用可能
  - 構造がシンプル（大規模環境において超重要）

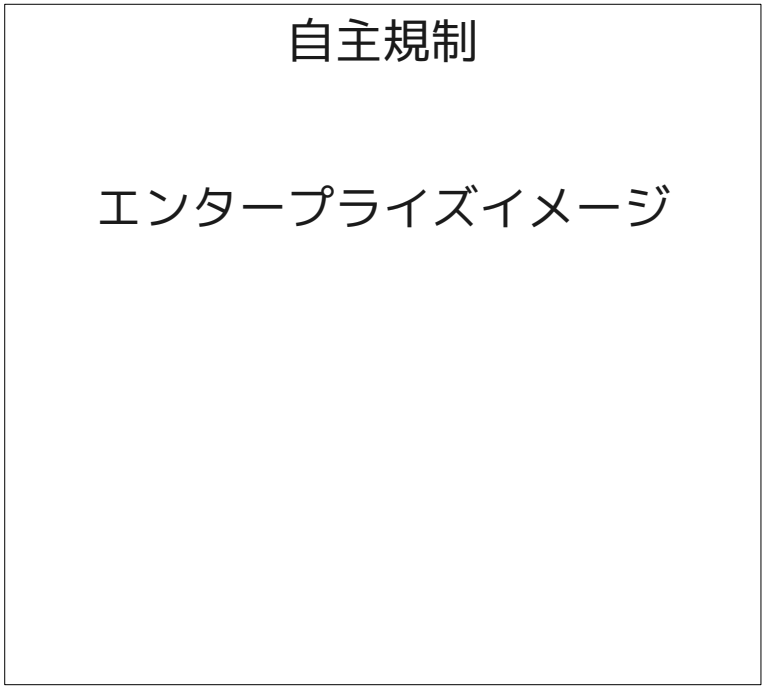
Swift って何？

- 参考資料
  - Swift の内部構造はこのあたりの資料を参照
    - http://www.slideshare.net/irix_jp/openstack-object-storage-overview
  - インストール方法
    - http://techtarget.itmedia.co.jp/tt/news/1109/20/news02.html
  - OpenStack の開発に参加したい人はここを参照
    - http://wiki.openstack.org/DevQuickstart/ja

Swiftが注目されるワケ

Swift が注目される背景

- 従来のストレージシステム
  - 企業をターゲットにして発展
    - 高価で高信頼なシステム
    - データを統合して集約管理



Swift が注目される背景

- 今後データはますます増えていくと言われるが・・・

自主規制

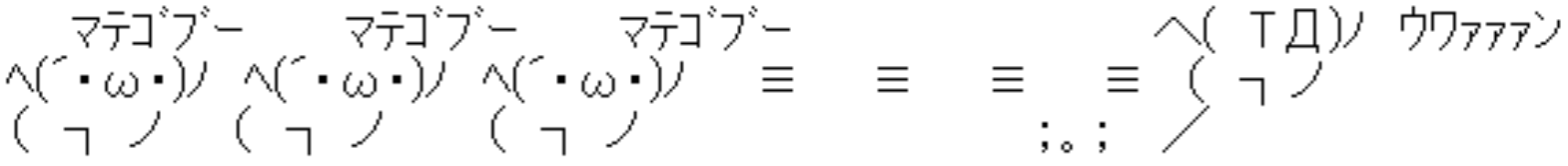
IDC
国内ディスクストレージシステム市場
ファイルベース／ブロックベース別出荷
容量実績と予測、 2005 年～ 2015 年

- 実際増えているのは企業内のデータではなく WEB 系
  - 写真、動画、アクセスログ、ゲーム等々・・・

http://www.idcjapan.co.jp/Press/Current/20110831Apr.html

Swift が注目される背景

- しかし爆発的に増えている WEB 系データに対して従来のストレージは親和性が低い
  - WEB の分散スケールアウトについていけない
    - 従来ストレージはスケールアップでの対応
    - シングルネームスペースで管理できる容量の上限が低い
    - プロトコルがインターネットを経由することを前提としていない

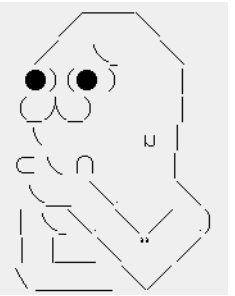


Swift が注目される背景

- しかし爆発的に増えている WEB 系データに対して従来のストレージは親和性が低い
  - データ容量が企業に比べると桁違いに多いため、コストがかかりすぎる。
    - 社員 5000 人 x 10GB/ 人 = 50TB
    - ユーザ 5000 万人 x 1GB/ 人 = 50PB

ビッグデータ → 自主規制
ラーメンやさいましまし

マシマシってレベル
じゃないだろ・・・



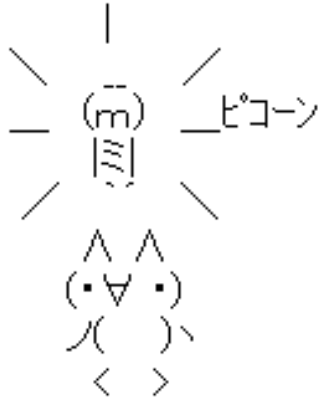
Swift が注目される背景

- そこで分散オブジェクトストレージが注目されるようになった。

- 技術的な問題
  - HTTP を利用した入出力
  - スケールアウト構造
- コスト的な問題
  - 安価なハードウェアを使い、高信頼な構造

情報貯蔵 SWIFT OBJECT STORAGE すういフト

の予感！



Swiftを使う上で
抑えておくポイント

BASE と CAP 定理

- Swift のトランザクションモデルは BASE
  - **B**asically **A**vailable
    - Proxy と Storage Node の並列化による冗長性
  - **S**oft-State
    - 取り出すデータが最新かどうかは判断しない
  - **E**ventually Consistent
    - 楽観的レプリケーションによる結果整合性

  - 【参考】Cloud の技術的特徴について
    - http://qcontokyo.com/tokyo-2009/pdf/GeneralSession-Day2-Maruyama.pdf

BASE と CAP 定理

- CAP 定理との関係
  - Consistency（一貫性）
  - Availability（可用性）
  - Partition-tolerance（分断耐性）

	通常の ファイルサーバ	Swift
Consistency	○	×
Availability	△	○
Partition-tolerance	×	○

CAP も BASE も DB の話なので無理やり当てはめてます。

BASE と CAP 定理

- 重要なのは「結果整合性」
  - 楽観的レプリケーション
  - データの取り出しはレプリケーションからランダム
  - ノード障害からの復旧時等、レプリケーションが終わっていない状態において古いデータが取り出される可能性がある
  - この特性を理解した上でシステムに適応するストレージシステムを採用する必要がある。

CAP も BASE も DB の話なので無理やり当てはめてます。

使い所

Swift の使い所

評価

- オンラインストレージ

- Swift を使うと・・・
  - 多数のクライアントからの細かな同時アップロード、ダウンロードにも対応可能
  - 安価に大容量の確保が可能
  - ファイルに属性付加可能

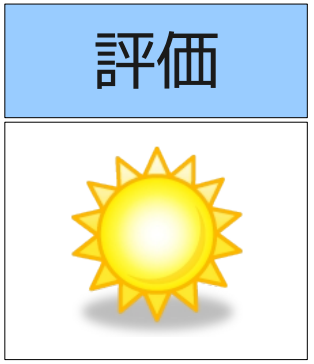
Swift の使い所

評価

- EC サイトのバックエンドストレージ
  - 例）ユーザからの写真アップロードの保存先

- Swift を使うと・・・
  - プログラムを簡潔に記述可能
    - ストレージとクライアントを直結できる
    - ネームスペースを単一にできる
  - 突発的な負荷にも強い

Swift の使い所



- ストリーミング用ストレージ

- Swift を使うと・・・
  - 安価に高スループットな配信環境を構築
    - 複製からランダムにデータを取得するので、レプリカを増やすことでリニアにスループットを向上させられる
    - ファイルを途中から取得可能
    - ファイルを複数のノードに分散可能

Swift の使い所

評価

- 大量ログの保管先＆取り出し
  - 例）多数のサーバからのログアップロード
  - 例）解析用 Hadoop への大量ロード

- Swift を使うと・・・
  - スケールアウトするので、性能の確保が安価に可能

Swift の使い所

評価

- 社内のファイルサーバ

- Swift を使うと・・・
  - 使えなくは無いけど・・・
  - OS 標準でサポートされないので、CIFS/NFS に比べると使い勝手悪い。
  - 社内のデータはせいぜい数十 TB

Swift の使い所

評価

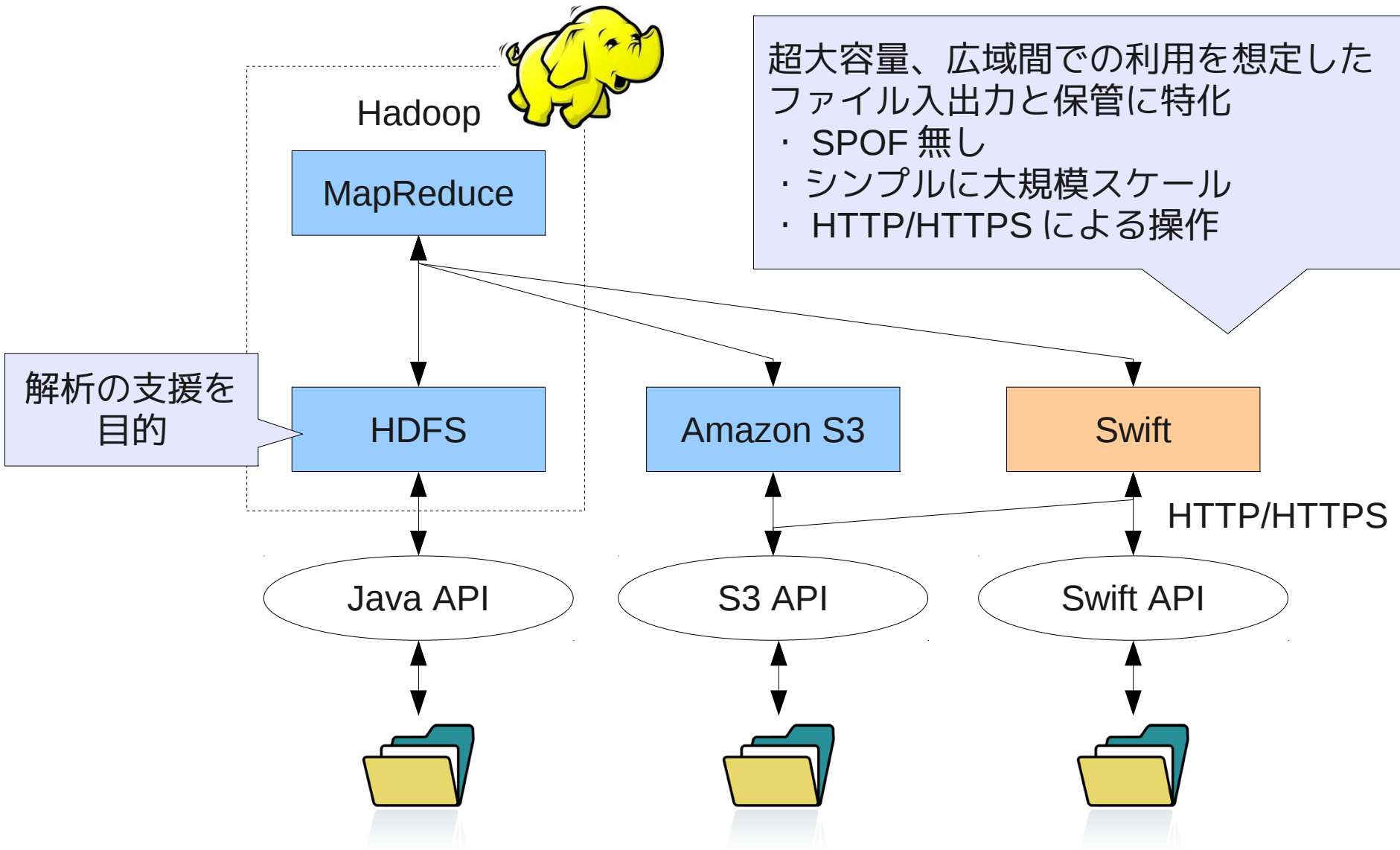
- データベースの保管先
  - 例）RDBMS の DBF 格納先

- Swift を使うと・・・
  - そもそも対応してる DB が無い
  - 仮に対応したとしてもブロックレベルでのアクセスできないから遅そう・・・

Swift の使い所

- まとめ
  - 必ずしも全てのシステムに使えるわけでは無い
    - 汎用的に使われる IF をサポートしてないため
  - 特にパッケージ系はまだまだ使えるものが少ない
    - バックアップソフトを中心に Swift API/S3 API に対応しているものが徐々に増えてきている
  - 使えばアプリ開発が楽になる
    - 従来ストレージにありがちな問題をほぼ解決できる

Hadoop/HDFSとの関係



ニュース

Essex リリース

Blueprint	1.4.4	1.4.5	1.4.6	1.4.7	1.4.8
	11-11-24	12-01-09	12-02-10	12-03-09	12-03-22
Number of Fixed Bugs	14	8	11	10	予定
Self Destructing Files	✔				
add more detail to rate limit errors	✔				
add swift man pages	✔				
better ring builder error messages	✔				
change ring builder exit codes	✔				
create swift recon docs	✔				
swift recon socket stats	✔				
tempauth autocreate accounts	✔				
zone specific recon	✔				
Old and/or hung process tools		✔			
suppot marker queries in swift cli		✔			
add memcache.conf			✔		
form post			✔		
remove rfc.sh			✔		
Remove old swift-stats-populate, swift-stats-report, and etc/stats-conf-sample				✔	

Essex リリース

- 機能拡張はおとなしめ
  - ファイルの時限消去
  - Formpost/TempURL
- 運用面の支援機能が充実
  - ゾーン統計情報
  - ソケット統計情報(/proc/net/socketstat)
  - 各種 man ページ / ドキュメント整備
  - エラーメッセージ、コードの見直し

まとめ

まとめ

- Swift は・・・
  - HTTP（REST）で通信するファイルサーバです。
  - 安価なハードで安全に動きます。RAID 不要。
  - 容量・性能がスケールし、SPOF がありません。
  - 汎用性が高く Swift 単体で使えます。
  - 構造がシンプルでトラブル要素が少ない。

  - 実用レベルまで達しており、PB クラスの商用サービス実績もあり！
    - ただし用途には向き不向きあり。

ご静聴ありがとうございました。

おぷ☆すた
Open ★ Stack

参考サイト

- OpenStack(本家)
  - http://www.openstack.org/
- 日本 OpenStack ユーザ会
  - http://openstack.jp/
- API マニュアル（本家）
  - http://docs.openstack.org/api/openstack-object-storage/1.0

参考サイト

- 使用させていただいた素材
  - http://cool-liberty.com/
  - http://tanukifont.sblo.jp/article/41432838.html
  - http://office.microsoft.com/ja-jp/images/

Appendix
Swiftの認証

Swift の認証機構

- 3つの認証方式
  - Keystone
    - OpenStack 共通認証方式
  - TempAuth
    - Swift 専用認証機能、簡易的な認証機能
    - WSGI ミドルウェア（ Proxy 上で稼働）
  - SWAuth
    - Swift 専用認証機能、Swift に特化した認証機能
    - WSGI ミドルウェア（ Proxy 上で稼働）

Swift の認証機構

- 認証方式比較

項目	Keystone	TempAuth	SWAuth
範囲	OpenStack 共通認証方式	Swift 専用	Swift 専用
DB	MySQL+Swift	Swift	Swift
アカウント操作	専用コマンド	Proxy 設定ファイルへベタ書き	専用コマンド
サーバ形態	専用サーバ	Proxy 上のミドルウェア	Proxy 上のミドルウェア
SPOF	有り ＊1	無し	無し
スケールアウト	しない ＊1	する	する

*1) diablo 時の実装

SWAuth

- SWAuth はかつて Swift 標準の認証機構だったが、Keystone が OpenStack 全体の共通認証として採用される事が決定しオワコン化

SWAuth

- SWAuth はかつて Swift 標準の認証機構だったが、Keystone が OpenStack 全体の共通認証として採用される事が決定しオワコン化

- ・・したかの様に見えたが、別プロジェクトとして継続。
- OpenStack Authentication (for Swift)
  - https://launchpad.net/swauth

SWAuth

- 新生 SWAuth のいいところ
  - Proxy のミドルウェアとして動作
    - Swift のメリットを享受
      - スケール＆耐障害性
  - バックエンド DB が Swift
    - Swift 外部に DB を持つ必要が無い
      - 一貫した ID 管理
    - Swift のメリットを享受
      - スケール＆耐障害性
  - WebConsole もある

SWAuth

- Swift を単体で利用する場合、SWAuth の利用がおすすめ
  - Keystone は Swift 的にはまだ使い勝手が悪い
    - 耐障害性、性能面でも課題あり
    - データの二重持ち（ Keystone/Swift ）
  - TempAuth はテスト用
    - ユーザの追加・変更のたびに Proxy の再起動
    - アカウントの削除不可
  - もちろん独自に作り込んでも良い。
    - その場合は TempAuth/SWAuth の実装が参考になる。

SWAuth

- SWAuth を使う
  - TempAuth で Swift が動く状態にしておく
  - Proxy サーバ上で実施
    - cd ~
    - git clone git://github.com/gholt/swauth.git
    - cd swauth
    - python setup.py build
    - python setup.py install

SWAuth

- SWAuth を使う
  - 設定ファイルを書き換え

```
[pipeline:main]
#pipeline = healthcheck cache tempauth proxy-server
pipeline = healthcheck cache swauth proxy-server

[filter:swauth]
use = egg:swauth#swauth
set log_name = swauth
super_admin_key = swauthkey
```

  - swift-init proxy reload

SWAuth

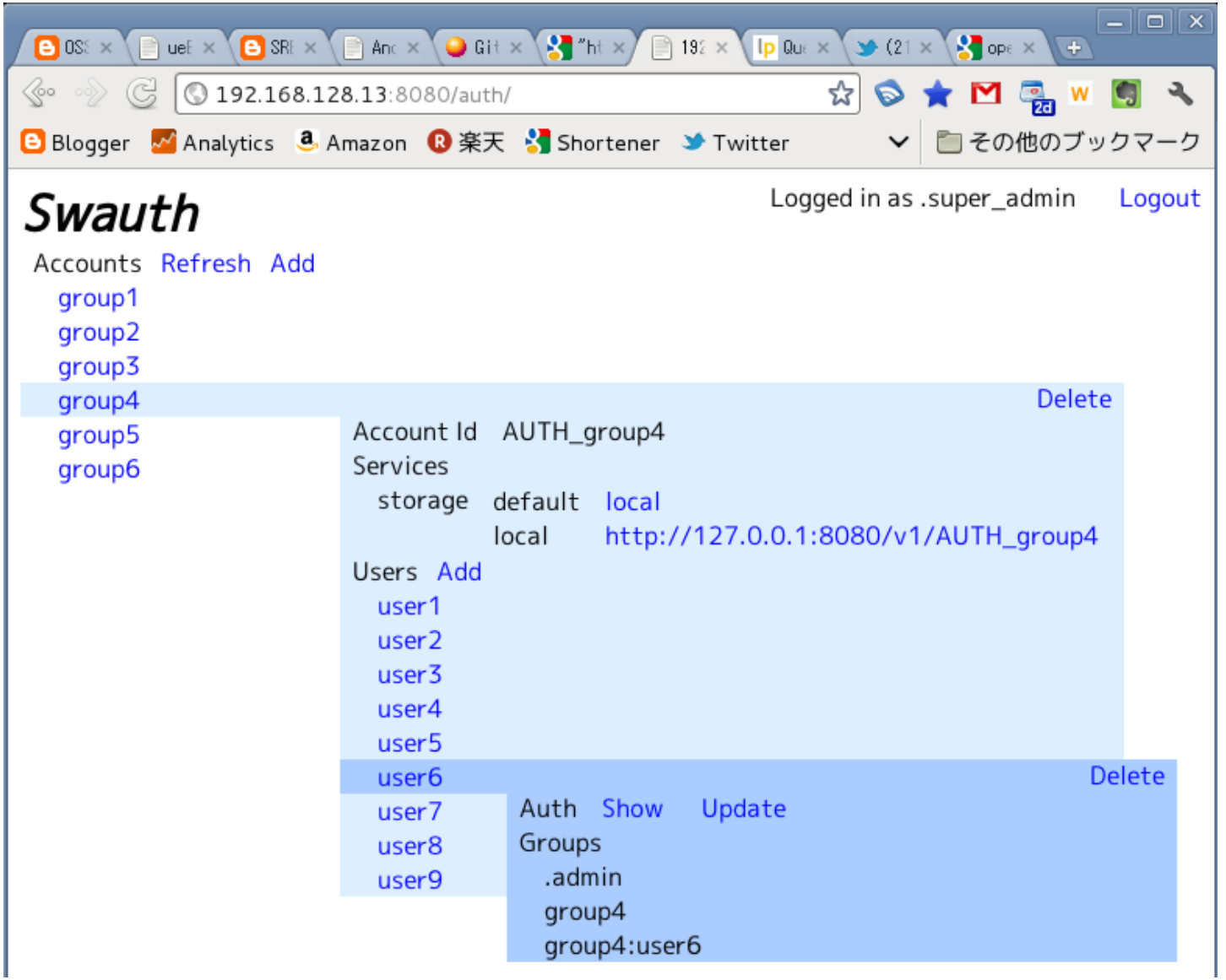
- SWAuth を使う
  - 初期化 (.super_admin:.super_admin を作成)
    - swauth-prep -A http://xxx/auth -K swauthkey
  - アカウント(グループ)の追加
    - swauth-add-account -A http://xxx/auth -K swauthkey group1
  - ユーザの追加
    - swauth-add-user -A http://xxx/auth -K swauthkey -a group1 user1 pass1

SWAuth

- WebConsole を使う
  - swauth の .super_admin ユーザの .webadmin コンテナへ以下のファイルをアップロードする
    - <git>/swauth/webadmin/index.html

    - ```
      swift -A http://127.0.0.1:8080/auth/v1.0 \
            -U .super_admin:.super_admin \
            -K swauthkey upload .webadmin index.html
      ```
      ```

SWAuth


192.168.128.13:8080/auth/
Blogger
Analytics
Amazon
楽天
Shortener
Twitter
その他のブックマーク
Swauth
Logged in as .super_admin
Logout
Accounts Refresh Add
group1
group2
group3
group4
group5
group6
Delete
Account Id AUTH_group4
Services
storage default local
local http://127.0.0.1:8080/v1/AUTH_group4
Users Add
user1
user2
user3
user4
user5
user6
user7
user8
user9
Delete
Auth Show Update
Groups
.admin
group4
group4:user6