Red Hat OpenStack Platform - Swift

Sachin*

May 18, 2016

1 Theory

- Swift allows users to store unstructured data(objects) with canonical names containing *three* part:
  - `/account`: Think `account` as a storage location and NOT user account. `account` stores meta-data of that account, plus list of all containers in that account. `account` is analogous to `/home` directory which may holds multiple users.
  - `/account/container`: Think of container as a root directory of user(analogous to `/home/<USER>/`). Account can have many containers with no limit.
  - `/account/container/object`: This is actual file. User may start storing a single file(files are stored in container as an object), or hierarchical data like `/photos/alaska/magic-bus/me.jpg` as an object. Swift stores multiple copies of single object across physical locations to ensure the data reliability and availability.
- Remember that, user do not have to know the actual location of the data. In-fact he never knows. User always access the data in the form of `/account/container/object`.

2 Handy commands

- *My settings*:
  - Swift server IP address: `192.168.8.80:8080`
  - AuthURL: `http://192.168.8.80:8080/auth/v1.0`
  - PublicURL/StorageURL: `http://192.168.8.80:8080/v1/AUTH_wasteland`
  - account: `wasteland`
  - container: `keys`
  - object: `mykey.pem`
- If you have access to Swift server, `Pubic-URL/StorageURL`, `Auth-Token`, & `account` can be fetched using following command,

```
swift stat -v
#                      StorageURL: http://192.168.8.80:8080/v1/AUTH_wasteland
#                      Auth Token: AUTH_tk968b0ae7947640be874af6cd897a2b1e
#                         Account: AUTH_wasteland
#                      Containers: 0
#                         Objects: 0
#                           Bytes: 0
# Containers in policy "default": 1
#    Objects in policy "default": 0
#      Bytes in policy "default": 0
#                   Accept-Ranges: bytes
#                     X-Timestamp: 1463138224.81309
#                      X-Trans-Id: tx1fe3ecbeb9f04fdc92287-005735e92c
#                    Content-Type: text/plain; charset=utf-8
```

*psachin@redhat.com

- New user can be added as follows,

```
# --- /etc/swift/proxy-server.conf ---
[filter:tempauth]
use = egg:swift#tempauth
# user_ACCOUNT_USER = PASSWORD [GROUP] <storage URL:8080>
user_wasteland_psachin = psachin .admin .reseller_admin

[app:proxy-server]
use = egg:swift#proxy
allow_account_management = true
account_autocreate = true
# --- File ends here ---

# Restart servers and Proxy
swift-init account restart
swift-init container restart
swift-init object restart
swift-init proxy restart
```

- Managing container and object using `swift` command

  Set following environmental variables

```
# --- ~/.profile ---
export ST_AUTH=http://192.168.8.80:8080/auth/v1.0
export ST_USER=wasteland:psachin
export ST_KEY=psachin
# --- File ends here ---
```

  Source the file before executing any command

```
source ~/.profile
```

  Most of the time, no configuration is needed, if Swift is enabled during packstack. You can actually start from here.

```
# --- Create container: 'keys' ---
swift post keys
# Verify/list containers
swift list
# --- Upload an object to container ---
# Create a file
echo "746c1c636cebe7a888fd77688dbfc252" > mykey.pem
# Upload object-'mykey.pem' to container-'keys'
swift upload keys mykey.pem
# Verify the object
swift list keys
# --- Download object ---
swift download keys mykey.pem
# Download object with different name
swift download keys mykey.pem -o mykey2.pem
```

  - Another way to access swift container/object is to provide username & password.

```
# swift -U <ACCOUNT>:<USER> -K <PASSWORD> COMMAND
swift -U wasteland:psachin -K psachin stat -v
#                      StorageURL: http://saio:8080/v1/AUTH_wasteland
#                      Auth Token: AUTH_tkf551ef98eb1e442b9efcef1261d87c64
#                         Account: AUTH_wasteland
#                      Containers: 1
#                         Objects: 0
#                           Bytes: 0
# Containers in policy "default": 1
```

```
#    Objects in policy "default": 0
#      Bytes in policy "default": 0
#                    Accept-Ranges: bytes
#                      X-Timestamp: 1463292608.77517
#                       X-Trans-Id: txc8bc7ba0659b497fb170f-00573b0ff4
#                     Content-Type: text/plain; charset=utf-8

# Upload an object
# swift -U <PASSWORD>:<PASSWORD> -K <PASSWORD> upload <CONTAINER> <file/object>
swift -U wasteland:psachin -K psachin upload keys mykey.pem
```

- Managing container and object using APIs(`curl` command)

```
# --- Get token ---
# Set authURL and publicURL
export authURL="http://192.168.8.80:8080/auth/v1.0/"
export publicURL="http://192.168.8.80:8080/v1/AUTH_wasteland"

curl -v \
         -H "X-Auth-User: wasteland:psachin" \
         -H "X-Auth-Key: psachin" \
         $authURL

# *   Trying 192.168.8.80...
# * Connected to 192.168.8.80 (192.168.8.80) port 8080 (#0)
# > GET /auth/v1.0/ HTTP/1.1
# > Host: 192.168.8.80:8080
# > User-Agent: curl/7.43.0
# > Accept: */*
# > X-Auth-User: wasteland:psachin
# > X-Auth-Key: psachin
# >
# < HTTP/1.1 200 OK
# < X-Storage-Url: http://192.168.8.80:8080/v1/AUTH_wasteland
# < X-Auth-Token-Expires: 82975
# < X-Auth-Token: AUTH_tk968b0ae7947640be874af6cd897a2b1e
# < Content-Type: text/html; charset=UTF-8
# < X-Storage-Token: AUTH_tk968b0ae7947640be874af6cd897a2b1e
# < Content-Length: 0
# < X-Trans-Id: tx9c1bef9065754dd9b68ec-005735c49d
# < Date: Fri, 13 May 2016 12:12:13 GMT
# <
# * Connection #0 to host 192.168.8.80 left intact

export token="AUTH_tk968b0ae7947640be874af6cd897a2b1e"

# Verify account access
curl -v \
         -H "X-Storage-Token: $token" \
         $publicURL

# *   Trying 192.168.8.80...
# * Connected to 192.168.8.80 (192.168.8.80) port 8080 (#0)
# > GET /v1/AUTH_wasteland HTTP/1.1
# > Host: 192.168.8.80:8080
# > User-Agent: curl/7.43.0
# > Accept: */*
# > X-Storage-Token: AUTH_tk968b0ae7947640be874af6cd897a2b1e
# >
# < HTTP/1.1 204 No Content
# < Content-Length: 0
```

```
# < Accept-Ranges: bytes
# < X-Account-Object-Count: 0
# < X-Account-Storage-Policy-Default-Bytes-Used: 0
# < X-Account-Storage-Policy-Default-Object-Count: 0
# < X-Timestamp: 1463138224.81309
# < X-Account-Bytes-Used: 0
# < X-Account-Container-Count: 0
# < Content-Type: text/plain; charset=utf-8
# < X-Account-Storage-Policy-Default-Container-Count: 0
# < X-Trans-Id: tx95142c2182024593 99c88-005735cac1
# < Date: Fri, 13 May 2016 12:38:25 GMT
# <
# * Connection #0 to host 192.168.8.80 left intact

# --- Create a container: 'keys' ---
curl -v \
         -H "X-Storage-Token: $token" \
         -X PUT $publicURL/keys

# *   Trying 192.168.8.80...
# * Connected to 192.168.8.80 (192.168.8.80) port 8080 (#0)
# > PUT /v1/AUTH_wasteland/keys HTTP/1.1
# > Host: 192.168.8.80:8080
# > User-Agent: curl/7.43.0
# > Accept: */*
# > X-Storage-Token: AUTH_tk968b0ae7947640be874af6cd897a2b1e
# >
# < HTTP/1.1 201 Created
# < Content-Length: 0
# < Content-Type: text/html; charset=UTF-8
# < X-Trans-Id: tx39b7aee463b64127adfe2-005735cb92
# < Date: Fri, 13 May 2016 12:41:54 GMT
# <
# * Connection #0 to host 192.168.8.80 left intact

# Verify container
curl -v \
         -H "X-Storage-Token: $token" \
         -X GET $publicURL/keys

# *   Trying 192.168.8.80...
# * Connected to 192.168.8.80 (192.168.8.80) port 8080 (#0)
# > GET /v1/AUTH_wasteland/keys HTTP/1.1
# > Host: 192.168.8.80:8080
# > User-Agent: curl/7.43.0
# > Accept: */*
# > X-Storage-Token: AUTH_tk968b0ae7947640be874af6cd897a2b1e
# >
# < HTTP/1.1 204 No Content
# < Content-Length: 0
# < X-Container-Object-Count: 0
# < Accept-Ranges: bytes
# < X-Storage-Policy: default
# < X-Container-Bytes-Used: 0
# < X-Timestamp: 1463138224.83257
# < Content-Type: text/html; charset=UTF-8
# < X-Trans-Id: tx05408e3d41c246ea930f5-005735cc21
# < Date: Fri, 13 May 2016 12:44:17 GMT
# <
# * Connection #0 to host 192.168.8.80 left intact
```

```
# --- Upload object to container ---
# Create a file
echo "746c1c636cebe7a888fd77688dbfc252" > mykey.pem

# Upload object-'mykey.pem' to container-'keys'
curl -v \
         -H "X-Storage-Token: $token" \
         -X PUT $publicURL/keys/mykey.pem -T mykey.pem

# *   Trying 192.168.8.80...
# * Connected to 192.168.8.80 (192.168.8.80) port 8080 (#0)
# > PUT /v1/AUTH_wasteland/keys/mykey.pem HTTP/1.1
# > Host: 192.168.8.80:8080
# > User-Agent: curl/7.43.0
# > Accept: */*
# > X-Storage-Token: AUTH_tk968b0ae7947640be874af6cd897a2b1e
# > Content-Length: 43
# > Expect: 100-continue
# >
# < HTTP/1.1 100 Continue
# * We are completely uploaded and fine
# < HTTP/1.1 201 Created
# < Last-Modified: Fri, 13 May 2016 12:53:00 GMT
# < Content-Length: 0
# < Etag: 640ebd176639fb6ef9a3227770ee7b17
# < Content-Type: text/html; charset=UTF-8
# < X-Trans-Id: txf33923d6fbfe4523b4451-005735ce2b
# < Date: Fri, 13 May 2016 12:52:59 GMT
# <
# * Connection #0 to host 192.168.8.80 left intact

# Download an object
curl -v \
         -H "X-Storage-Token: $token" \
         -X GET $publicURL/keys/mykey.pem > mykey.pem

# *   Trying 192.168.8.80...
#   % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
#                                  Dload  Upload   Total   Spent    Left  Speed
        #   0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:-- 0* \
#                           Connected to 192.168.8.80 (192.168.8.80) port 8080 (#0)
# > GET /v1/AUTH_wasteland/keys/mykey.pem HTTP/1.1
# > Host: 192.168.8.80:8080
# > User-Agent: curl/7.43.0
# > Accept: */*
# > X-Storage-Token: AUTH_tk968b0ae7947640be874af6cd897a2b1e
# >
# < HTTP/1.1 200 OK
# < Content-Length: 43
# < Accept-Ranges: bytes
# < Last-Modified: Fri, 13 May 2016 12:53:00 GMT
# < Etag: 640ebd176639fb6ef9a3227770ee7b17
# < X-Timestamp: 1463143979.89953
# < Content-Type: application/octet-stream
# < X-Trans-Id: tx6b14a272331b4bc6937db-005735cef1
# < Date: Fri, 13 May 2016 12:56:17 GMT
# <
# { [43 bytes data]
# 100    43  100    43    0     0   2748      0 --:--:-- --:--:-- --:--:--  2866
```

```
# * Connection #0 to host 192.168.8.80 left intact
```

- Get statistics

```
# Auth related information
swift auth
# export OS_STORAGE_URL=http://192.168.8.80:8080/v1/AUTH_wasteland
# export OS_AUTH_TOKEN=AUTH_tkf551ef98eb1e442b9efcef1261d87c64

swift auth -v
# export ST_AUTH=http://192.168.8.80:8080/auth/v1.0
# export ST_USER=wasteland:psachin
# export ST_KEY=psachin

# To obtain Storage URL and Auth-Token
swift stat -v

# Get statistics of container and/or object
swift stat [container]
swift stat [container] [object]

# Retrive capability of proxy
swift capabilities
# Core: swift
#  Options:
#   account_autocreate: True
#   account_listing_limit: 10000
#   allow_account_management: True
#   container_listing_limit: 10000
#   extra_header_count: 0
#   max_account_name_length: 256
#   max_container_name_length: 256
#   max_file_size: 5368709122
#   max_header_size: 8192
#   max_meta_count: 90
#   max_meta_name_length: 128
#   max_meta_overall_size: 4096
#   max_meta_value_length: 256
#   max_object_name_length: 1024
#   policies: [{u'default': True, u'name': u'default', u'aliases': u'default'}]
#   strict_cors_mode: True
#   version: 2.7.1.dev83
# Additional middleware: bulk_delete
#  Options:
#   max_deletes_per_request: 10000
#   max_failed_deletes: 1000
# Additional middleware: bulk_upload
#  Options:
#   max_containers_per_extraction: 10000
#   max_failed_extractions: 1000
# Additional middleware: container_sync
#  Options:
#   realms: {u'TEST': {u'clusters': {u'SAIO': {u'current': True}}}}
# Additional middleware: slo
#  Options:
#   max_manifest_segments: 1000
#   max_manifest_size: 2097152
#   min_segment_size: 1
# Additional middleware: staticweb
# Additional middleware: tempauth
#  Options:
```

```
#   account_acls: True
# Additional middleware: tempurl
#  Options:
#   incoming_allow_headers: []
#   incoming_remove_headers: [u'x-timestamp']
#   methods: [u'GET', u'HEAD', u'PUT', u'POST', u'DELETE']
#   outgoing_allow_headers: [u'x-object-meta-public-*']
#   outgoing_remove_headers: [u'x-object-meta-*']

# List container's details(Similar to `ls -lh`)
swift list --lh [container]
```

- Object versioning

  When an object is overwritten, it's older version is lost, but there is a way we can store older version(s) of an object, no matter how many times is was overwritten.

  To enable object versioning, set `allow_versions` option to `true` in container configuration file.

```
# --- /etc/swift/container-server.conf ---
[app:container-server]
allow_versions = true
# --- File ends here ---

# --- Create containers ---
# Create 'archive' container to hold 'current' container's object versions
swift post archive

# Now create 'current' container with header 'X-Versions-Location'
# pointing to 'archive'
swift post current -H "X-Versions-Location: archive"

# --- Other similar ways(Optional) ---
# May also define content length at the time of creating a container
swift post archive -H "content-length: 0"
swift post current -H "content-length: 0" -H "X-Versions-Location: archive"

# And also specify Read ACL(World readable) during container creation
swift post -r ".r:*" archive -H "content-length: 0"
swift post -r ".r:*" current -H "content-length: 0" -H "X-Versions-Location: archive"
# --- xxx ---
```

- Managing container's quota

```
# Limit maximum of 2 objects in container 'keys'
swift post -H "X-Container-Meta-Quota-Count: 2" keys

# Max size of an object should be not more than 512 bytes in container 'keys'
swift post -H "X-Container-Meta-Quota-Bytes: 512" keys
```

3 Builder files

- Acts as a database
- Python pickle

```
import pickle
print(pickle.load(open('object.builder')))
```

- Ring builder command

```
# Account server runs on port 6002
swift-ring-builder add account.builder <region><zone>-<IP>:6002/<device><weight>
# Container server runs on port 6001
swift-ring-builder add container.builder <region><zone>-<IP>:6001/<device><weight>
# and the Object server runs on port 6000
swift-ring-builder add object.builder <region><zone>-<IP>:6000/<device><weight>
swift-ring-builder add object-N.builder <region><zone>-<IP>:6000/<device><weight>
```

- Region: Geographical location
- Zone: within region isolation
- Weight: Relative number of partition a drive will have
  - 1TB ~ Weight of 100.0
  - 2TB ~ Weight of 200.0..

4 Swift Ring

- Data structure
- Describes your cluster
- One ring each for `account`, `container`, & `object`

```
# swift-ring-builder account.builder create <PartitionPower> <Replicas> <MinPartHrs>

cd /etc/swift/
swift-ring-builder account.builder create 10 3 1
swift-ring-builder container.builder create 10 3 1
swift-ring-builder object.builder create 10 3 1
```

- How to decide value of Partition Power?

  Assume that I have a system with 4 drives right now, but the maximum drives I can go up-to is 10.

```
# Partition Power
2^part_power > (Nos. of drives you think you will have at-scale) * 100

# I may have 10 drives in future
2^part_power > 10 * 100
2^part_power > 1000
2^10 > 1000
1024 > 1000  # 2^10 = 1024 just goes above 1000, which is perfect.
```

- Calculate Partition Power(Python snippet)

```
# Use python3 interpreter
from math import log2, ceil
print(ceil(log2(10 * 100)))  # 10 <- Partition Power
```

- Partition in Swift

```
/srv/2/node/sdb2/objects/171/a56/2ae7be8de859228d6575cc9fe5518a56/1479968148.23926.data

/srv/2/node/sdb2/objects/171 # partition number
/srv/2/node/sdb2/objects/171/a56 # last 3 chars from hashed objectname
/srv/2/node/sdb2/objects/171/a56/2ae7be8de859228d6575cc9fe5518a56/ # hashed objectname
/srv/2/node/sdb2/objects/171/a56/2ae7be8de859228d6575cc9fe5518a56/1479968148 # timestamp
```

- Swift partition table

	Replica # 1	Replica # 2	Replica #3
Partition # 0	Device # 0	Device # 3	Device # 2
Partition # 1	Device # 1	Device # 2	Device # 0
Partition # 2	Device # 4	Device # 1	Device # 3

```
[vagrant@saio ~]$ swift-get-nodes /etc/swift/object.ring.gz test hoota CHANGELOG

Account         test
Container       hoota
Object          CHANGELOG


Partition       932
Hash            e90ec799284002b7f85ce1149841bbd0

Server:Port Device      127.0.0.1:6040 sdb4
Server:Port Device      127.0.0.1:6030 sdb3
Server:Port Device      127.0.0.1:6010 sdb1
Server:Port Device      127.0.0.1:6020 sdb2      [Handoff]


curl -g -I -XHEAD "http://127.0.0.1:6040/sdb4/932/test/hoota/CHANGELOG"
curl -g -I -XHEAD "http://127.0.0.1:6030/sdb3/932/test/hoota/CHANGELOG"
curl -g -I -XHEAD "http://127.0.0.1:6010/sdb1/932/test/hoota/CHANGELOG"
curl -g -I -XHEAD "http://127.0.0.1:6020/sdb2/932/test/hoota/CHANGELOG" # [Handoff]


Use your own device location of servers:
such as "export DEVICE=/srv/node"
ssh 127.0.0.1 "ls -lah ${DEVICE:-/srv/node*}/sdb4/objects/932/bd0/e90ec799284002b7f85ce1149841bbd0"
ssh 127.0.0.1 "ls -lah ${DEVICE:-/srv/node*}/sdb3/objects/932/bd0/e90ec799284002b7f85ce1149841bbd0"
ssh 127.0.0.1 "ls -lah ${DEVICE:-/srv/node*}/sdb1/objects/932/bd0/e90ec799284002b7f85ce1149841bbd0"
ssh 127.0.0.1 "ls -lah ${DEVICE:-/srv/node*}/sdb2/objects/932/bd0/e90ec799284002b7f85ce1149841bbd0" # [Handoff]

note: `/srv/node*` is used as default value of `devices`, the real value is set in the config file on each storage node.
```

5 Additional notes

Swift consistency processes:

- *Auditor*: Will walk through the storage nodes, read the data and the checksum, ensure the checksum matched with the database checksum. If the checksum didn't match, the data is moved to the Quarantine.
- *Replicator*: The replicator, will also scan each drive and ensures that the replicas of data is stored where is supposed to live. If it does not finds the data in that place(may be the data, due to corruption was moved to Quarantine), it will push the data to that place.

6 Slides notes

- Multiple HDD, where is my data store?
- HDD failure
- Storage problem
- Ownership of your data
- Access to data, HTTP, FTP, ReST
  - Mobile, Laptop..
- Swift
  - loosely tied to storage media
  - Scalable

  - Direct client access

- Terminology
  - Proxy: provides API access/ Coordinates requests to storage servers
  - Account: user namespace
  - Container: User defined segment of an account(root directory)
  - Object: Actual data
- Flow Proxy request -> Storage nodes(account, container, obj)
- Data placement
  - triple replication by default(as unique as possible)
  - Show Region/Zone pic
- Drive failures
  1. Umount failing drive
  2. Replicate/rebalance data
- Server failures
  1. Network, Power
  2. New data that is to be written will be placed elsewhere within a cluster/server
  3. Rebalancing happens
- Currupt data
  1. Stores checksum of the data with data itself
  2. Matches checksum of data periodically
     - If checksum doesnt match, the object is quarantined and the replication process rebalances the data/object
- Storage policies
  - Decide where you want to store data
    * Between swift clusters
    * Subset of hardware
  - Erasure coding <- Data availability policies
    * Based in frequency of access
    * Example:

```
swift post -H "X-Storage-Policy: gold" container_gold
swift post -H "X-Storage-Policy: silver" container_silver
swift post -H "X-Storage-Policy: ec42" container_ec42

swift upload container_gold cirros-0.3.4-x86_64-disk.img
swift upload container_silver cirros-0.3.4-x86_64-disk.img
swift upload container_ec42 cirros-0.3.4-x86_64-disk.img
```

- Erasure Codes
  - At the time if building a ring for Erasure codes, number of replicas are replaced with number of fragments

    ```
    swift-ring-builder object.builder create 10 6 1
    ```

    where $4 + 2 = 6$

    4 data fragments 2 parity data

    So that the system can sustain 4 disk failures before the data is treated to be lost
  - Erasure coding is implemented in Swift as storage policies

```
# /etc/swift/swift.conf
[storage-policy:2]
name = ec42
policy_type = erasure_coding
ec_type = liberasurecode_rs_vand
ec_num_data_fragments = 4
ec_num_parity_fragments = 2
```

  - https://www.youtube.com/watch?v=kH3DXMKlEr8
  - https://www.youtube.com/watch?v=GDNK1S4FJBQ

- ACLs
  - Container ACL

```
# World readable
swift post -r ".r:*" photos

# Allow .welcome.com but deny .noisy.com
swift post -r ".r:*.welcome.com,.r:-noisy.com" photos

# Enable object listing within a container
swift post -r ".r:*,.rlistings" photos
```

- Hashing
  - Swift hashing function

```
1  # Use python3 interpreter
2  # Swift hashing is based on MD5
3  # hash(path) = md5(path + per-cluster suffix)
4
5  # Python snippet to know on which drive the object will be stored,
6  # assuming I have 4 drives
7  from hashlib import md5
8
9  m = md5()
10 m.update("/account/container/object")  # Hypothetical path
11 digest = m.hexdigest()
12 print(digest)
13
14 # hex to int
15 hex2int = int(digest, 16)
16 print(hex2int)
17 # digest % (number of drives) = Drive number
18 print(hex2int % 4)  # 2
```

7 Swift handoff partitions

- How is a handoff partition flagged versus a partition that is marked to be moved during a rebalance?

  Answer (notmyname): "handoff" is only a thing defined by the results of the call to `get_more_nodes()`. it's not a concept that means anything with regards to rebalancing. ie it's not "flagged" or anything. handoffs are just an ordered walk through the ring

- How should one think of handoff devices?

  Answer (mattoliverau): A hand off device is a non primary node for a certain partition in the ring. Things are placed to hand off nodes when either

  - there wasn't enough primary nodes to keep it durable.
  - when write affinity has been set and you want to get your object durability written to a closer region or zone
  - on a ring rebalance

  When looking for an object (GET) swift will check all primary nodes for the object and then some of the hand off nodes.

  But in essence once on a handoff node, we have durability which is the most important. but if the primaries are busy or down you may not get your object back until swift corrects it self

  The replicators will look at the objects they have, and if its a partition they're a hand off for, becuase they received it cause other primaries where down, or a rebalance suddenly has now suddenly made them a handoff node for a partition, they will replicate it out to the primary nodes and then if successful, delete it.

  handoff nodes + eventual consistancy helps swift keep its awesome durability

- is it (handoff node) meant to be a temporary holding place?

  Answer (mattoliverau): Yeah

8 Links

- https://gitlab.cee.redhat.com/psachin/bootcamp
- HTML version of this[1] doc is available at: https://gitlab.cee.redhat.com/psachin/bootcamp/blob/master/2016/scripts/notes.org
- Slides: https://redhat.slides.com/psachin/rhosp-swift-2016
- Swift All In One on Fedora: https://github.com/psachin/fedora-saio

[1] Made with Love, LaTeX, & GNU Emacs