Symlink

محتویات

مقدمه

تشریح symlink

- نحوه ایجاد symlink درون حسابی
- نحوه ایجاد symlink بین حسابی
- ملاحظات
- انواع symlink: پویا و ایستا
- درخواست HEAD/GET به یک شیء symlink
- درخواست POST به یک شیء symlink
- Content-Type یک symlink
- درخواست DELETE به یک شیء symlink
- درخواست COPY به یک شیء symlink
- درخواست OPTIONS به یک شیء symlink
- Tempurls برای درخواست های یک symlink
- **symlink_path**
- symlink و اشیای بزرگ
- خطاها

استقرار

پیکربندی همگام سازی کیسه

کاربرد

- گرفتن توکن
- ایجاد symlink
- GET شی symlink
- HEAD شی symlink
- GET مستقیم شی symlink
- تنظیم فراداده برای شی symlink
- مشاهده اطلاعات symlink_path

احراز هویت با keystone

- گرفتن توکن
- ایجاد symlink
- GET شی symlink
- HEAD شی symlink
- GET مستقیم شی symlink

مشاهده اطلاعات symlink_path

symlink و ACL

پانویس

منابع

مقدمه

symlink[1] در سوئیفت به عنوان یک میان افزار قابل اتصال پیاده سازی شده است. این میان افزار به منظور ایجاد لینک بین اشیای کیسه های مختلف در یک حساب و یا حساب های مختلف استفاده میشود که در ادامه به تشریح این میان افزار پرداخته شده است.

تشریح symlink

اشیاء symlink در سوئیفت ، مانند symbolic links در سیستم عامل های شبه یونیکس، حاوی ارجاع به یک شیء دیگر تحت عنوان "شیء هدف" هستند. وجود یک شیء symlink به هیچ وجه بر شیء هدف تأثیر نمی گذارد. یک مورد مهم استفاده از شیء symlink، استفاده از یک مسیر در یک کیسه برای دسترسی به یک شیء در یک کیسه دیگر، با یک سیاست ذخیره سازی متفاوت است که این اجازه می دهد تا مصالحه بین هزینه و کارایی سیاست ذخیره سازی بر روی اشیاء انجام شود.

نحوه ایجاد symlink درون حسابی

کلاینت ها با انجام یک درخواست PUT با طول صفر (zero-length PUT) و با هدر **X-Symlink-Target:<container>/<object>**، یک Swift symlink درون حسابی ایجاد می کنند.

نحوه ایجاد symlink بین حسابی

symlink بین حسابی نیز می تواند وجود داشته باشد که به این منظور هدر **X-Symlink-Target-Account: <account>** باید در دستور گنجانده شود و در صورت نبود این هدر، به طور خودکار حساب شیء symlink در فرآیند درخواست PUT در نظر گرفته شده و یک Swift symlink درون حسابی ایجاد می گردد.

ملاحظات

- symlink ها باید اشیای zero-byte باشند، بنابراین تلاش برای قرار دادن یک symlink با **Content-Length** غیر صفر، منجر به خطای سری 400 می شود.
- POST با هدر **X-Symlink-Target** همیشه منجر به خطای سری 400 می شود.
- شیء هدف نیازی نیست در زمان ایجاد symlink وجود داشته باشد.
- یک درخواست GET/HEAD به یک symlink منجر به درخواستی به شیء مورد نظر می شود که توسط هدرهای **X-Symlink-Target-Account** و **X-Symlink-Target** به آن ارجاع داده شده است. پاسخ درخواست GET/HEAD حاوی یک هدر **Content-Location** با موقعیت مسیر شیء مورد نظر خواهد بود. درخواست GET/HEAD به یک symlink با پارامتر پرس و جو **?symlink=get** منجر به این می شود که درخواست خود symlink را هدف قرار دهد.
- یک Symlink می تواند به Symlink دیگر اشاره کند. Symlink های زنجیره‌ای تا زمانی که هدف یک Symlink نباشد، رد می‌شوند. اگر تعداد Symlink های زنجیره ای از حد مجاز **symloop_max** بیشتر شوند، یک پاسخ خطا ایجاد می

شود. مقدار **symloop_max** را می توان در قسمت پیکربندی symlink در proxy-server.conf تعریف کرد که اگر مشخص نشده باشد، مقدار پیش‌فرض آن 2 است و چنانچه مقدار کمتر از 1 برای آن مشخص شود، از مقدار پیش‌فرض استفاده می‌شود.

انواع symlink: پویا و ایستا

کلاینت ها ممکن است به صورت اختیاری یک هدر**X-Symlink-Target-Etag: <etag >** را در طول PUT اضافه کنند که وجود این هدر موجب ایجاد یک "static symlink" به جای یک "dynamic symlink" می شود. static symlink ها به جای یک نام خاص به یک شیء خاص اشاره می کنند که این کار را با استفاده از مقدار تنظیم شده در هدر **X-Symlink-Target-Etag** خود انجام می دهند تا تأیید کنند که همچنان با ETag شیء مورد نظر در GET مطابقت دارد. همچنین بر خلاف یک dynamic symlink، شیء هدف مورد اشاره در هدر **X-Symlink-Target** باید وجود داشته باشد و ETag آن باید با **X-Symlink-Target-Etag** مطابقت داشته باشد، در غیر این صورت ایجاد symlink یک خطای کلاینت را برمی گرداند.

اگر یک symlink ایستا (یعنی symlink ایجاد شده با هدر **X-Symlink-Target-Etag**) symlink ایستای دیگری را هدف قرار دهد، هر دو هدر **X-Symlink-Target-Etag** باید با شیء هدف مطابقت داشته باشند تا GET موفق شود. اگر یک symlink ایستا یک symlink پویا را هدف قرار دهد (یعنی symlink ایجاد شده بدون هدر X-Symlink-Target-Etag)، هدر **X-Symlink-Target-Etag** مربوط به symlink ایستا باید Etag شیء zero byte باشد. اگر یک symlink با **X-Symlink-Target-Etag** یک مانیفست شیء بزرگ را هدف قرار دهد، باید با ETag مانیفست مطابقت داشته باشد (به عنوان مثال ETag که توسط **multipart-manifest=get** یا مقدار در هدر **X-Manifest-Etag** برگردانده شده است).

درخواست HEAD/GET به یک شیء symlink

یک درخواست HEAD/GET به یک شیء symlink مانند یک درخواست HEAD/GET معمولی به شیء هدف رفتار می کند. بنابراین، صدور یک درخواست HEAD به symlink، فراداده هدف را برمی گرداند، و صدور یک درخواست GET به symlink، داده ها و فراداده های شیء هدف را برمی گرداند. برای برگرداندن فراداده symlink (با بدنه خالی آن) یک درخواست GET/HEAD با پارامتر پرس و جو **?symlink=get** باید به یک شیء symlink ارسال شود.

درخواست POST به یک شیء symlink

یک درخواست POST به یک symlink منجر به پاسخ تغییر مسیر موقت 307 می شود. پاسخ حاوی یک هدر **Location** با مسیر شیء هدف به عنوان مقدار خواهد بود. درخواست هرگز توسط سوئیفت به شیء هدف هدایت نمی شود. با این وجود، فراداده در درخواست POST بر روی symlink اعمال می شود زیرا سرورهای شیء نمی توانند به طور قطع بدانند که آیا شیء فعلی یک symlink است یا خیر.

Content-Type یک symlink

Content-Type یک symlink کاملاً مستقل از شیء هدف آن است. برای راحتی، سوئیفت به طور خودکار **Content-Type** را اگر به صراحت توسط کلاینت تنظیم نشده باشد بر روی یک Symlink PUT تنظیم می کند. اگر کلاینت یک **X-Symlink-Target-Etag** بفرستد، سویفت **Content-Type** مربوط به symlink را روی شیء هدف قرار می دهد، در غیر این صورت **application/symlink** تنظیم می شود. می‌توانید**Content-Type** یک symlink را با استفاده از **?symlink=get** مرور کنید. با استفاده از درخواست POST می‌توانید **Content-Type** یک symlink را تغییر دهید. **Content-Type** یک symlink در فهرست کیسه ظاهر می شود.

درخواست DELETE به یک شیء symlink

درخواست DELETE به یک symlink، خودsymlink را حذف می کند. شیء مورد نظر حذف نخواهد شد.

درخواست COPY به یک شیء symlink

یک درخواست COPY یا یک درخواست PUT با هدر **X-Copy-From** به یک symlink، شیء هدف را کپی می کند. همان درخواست به symlink با پارامتر پرس و جو**?symlink=get**، خود symlink را کپی می کند.

درخواست OPTIONS به یک شیء symlink

یک درخواست OPTIONS به یک symlink فقط با گزینه‌های symlink پاسخ می‌دهد. درخواست به شیء هدف هدایت نمی شود. لطفاً توجه داشته باشید که اگر شیء هدف symlink در کیسه دیگری با تنظیمات CORS باشد، پاسخ تنظیمات را منعکس نخواهد کرد.

Tempurls برای درخواست های یک symlink

Tempurls را می توان برای درخواست های GET/HEAD یک symlink استفاده کرد، اما PUT مجاز نیست و منجر به خطای سری 400 می شود. Tempurls های GET/HEAD محدوده کلید tempurl را رعایت می کنند. tempurl کیسه فقط روی symlink کار می کند که کیسه هدف با symlink یکسان است. در صورتی که یک symlink شیء را در یک کیسه دیگر هدف قرار دهد، یک درخواست GET/HEAD منجر به خطای 401 Unauthorized خواهد شد. tempurl سطح حساب، symlink بین کیسه را مجاز می‌کند، اما symlink بین حسابی را نمی‌دهد.

اگر یک شیء symlink در حالی که در یک کیسه نسخه بندی شده است بازنویسی شود، خود شیء symlink نسخه می شود، نه شیء ارجاع شده.

symlink_path

یک درخواست GET با پارامتر پرس و جو **?format=json** به کیسه ای که حاوی symlink است، با اطلاعات اضافی **symlink_path** برای هر شیء symlink در فهرست کیسه پاسخ می‌دهد. مقدار **symlink_path** مسیر هدف symlink است. کلاینت‌ها می‌توانند symlink و سایر اشیاء را با این تابع متمایز کنند. توجه داشته باشید که پاسخ‌ها در هر قالب دیگری (به عنوان مثال **?format=xml**) شامل اطلاعات **symlink_path** نیستند. اگر یک هدر **X-Symlink-Target-Etag** در symlink گنجانده شده باشد، فهرست‌های کیسه JSON آن مقدار را در یک کلید **symlink_etag** و **Content-Length** شیء مورد نظر در کلید **symlink_bytes** گنجانده می‌شود.

symlink و اشیای بزرگ

اگر یک symlink ایستا یک شیء بزرگ ثابت را هدف قرار دهد، اندازه SLO و slo_etag را با استفاده از کلیدهای **symlink_bytes** و **slo_etag** در فهرست کیسه منتقل می کند. با این حال، مانیفست‌هایی که قبل از swift نسخه 2.12.0 (منتشر شده در دسامبر 2016) ایجاد شده‌اند، حاوی فراداده کافی برای انتشار اطلاعات اضافی SLO در فهرست نیستند. کلاینت ممکن است قبل از ایجاد یک symlink ایستا برای افزودن فراداده مورد نیاز، مانیفست (COPY w/ **?multipart-manfiest=get**) را دوباره ایجاد کند.

خطاها

- PUT با هدر **X-Symlink-Target** با **Content-Length** غیر صفر یک خطای **BadRequest 400** ایجاد می کند.
- POST با هدر **X-Symlink-Target** یک خطای **BadRequest 400** ایجاد می کند.
- پیمایش GET/HEAD بیش از symlinkهای زنجیره ای **symloop_max** باعث ایجاد خطای **Conflict 409** می شود.
- PUT/GET/HEAD روی یک symlink که شامل هدر **X-Symlink-Target-Etag** است که با هدف مطابقت ندارد، خطای **Conflict 409** ایجاد می کند.
- POST ها یک خطای **تغییر مسیر موقت 307** ایجاد می کنند.

استقرار

symlink ها با افزودن میان افزار **symlink** به خط لوله **WSGI** سرور پروکسی و گنجاندن بخش پیکربندی فیلتر مربوطه در فایل **proxy-server.conf** فعال می شود. میان‌افزار **symlink** باید **بعد از میان‌افزارهای dlo، slo و versioned_writes** قرار گیرد، اما **قبل از میان‌افزار encryption** در خط لوله قرار گیرد. برای جزئیات بیشتر به فایل proxy-server.conf-sample مراجعه کنید. اگر از ویژگی همگام سازی کیسه استفاده می شود، مراحل بیشتری لازم است.

به عنوان نمونه در زیر پیکربندی نمونه‌ای از فایل **proxy-server-.conf** که از **symlink** پشتیبانی‌می‌کند، آورده شده است:

```
.
.
pipeline = catch_errors gatekeeper healthcheck proxy-logging cache bulk tempurl slo dlo ratelimit
crossdomain authtoken keystoneauth staticweb container-quotas account-quotas symlink proxy-logging proxy-
.
.
[filter:symlink]
use = egg:swift#symlink
# Symlinks can point to other symlinks provided the number of symlinks in a
# chain does not exceed the symloop_max value. If the number of chained
# symlinks exceeds the limit symloop_max a 409 (HTTPConflict) error
# response will be produced.
# symloop_max = 2
.
.
```

دقت شود پس از اعمال تغییرات در فایل مذکور، داخل کانتینر هیولایی خود دستور زیر را اجرا کنید و از راه‌اندازی مجدد همه سرویس‌ها اطمینان حاصل کنید:

```
swift-init all restart
```

توجه: هنگامی که میان‌افزار symlink را در خط لوله خود مستقر کردید، نه باید میان‌افزار symlink را حذف کنید و نه swift را به نسخه‌ای زودتر از symlinkهایی که پشتیبانی می‌شوند تنزل دهید. انجام این کار ممکن است منجر به نتایج غیرمنتظره فهرست کیسه شود، علاوه بر این که اشیاء symlink مانند یک شیء معمولی رفتار می کنند.

پیکربندی همگام سازی کیسه

اگر از همگام‌سازی کیسه استفاده می‌شود، باید میان‌افزار symlink به خط لوله سرویس گیرنده داخلی همگام‌سازی کیسه اضافه شود. مراحل پیکربندی زیر مورد نیاز است:

- یک فایل پیکربندی کلاینت داخلی سفارشی برای همگام سازی کیسه (اگر قبلاً استفاده نشده است) بر اساس فایل **internal-client.conf-sample** ایجاد کنید. برای مثال **internal-client.conf-sample** را در **etc/swift/container-sync-client.conf** کپی کنید.
- این فایل را طوری تغییر دهید که میان افزار symlink را به همان روشی که در بالا برای سرور پراکسی توضیح داده شد در خط لوله قرار دهد.
- بخش **container-sync** را در تمام فایل‌های پیکربندی سرور کیسه تغییر دهید تا با استفاده از گزینه **internal_client_conf_path** به فایل پیکربندی کلاینت داخلی اشاره کنید. مثلا:

```
internal_client_conf_path = /etc/swift/container-sync-client.conf
```

توجه: این مراحل پیکربندی همگام‌سازی کیسه برای انجام آزمون های کاوشگر [۲]همگام‌سازی کانتینر ضروری خواهد بود، اگر میان‌افزار symlink در خط لوله پروکسی یک خوشه آزمایشی گنجانده شود.

کاربرد

در ادامه با ذکر مثال نحوه استفاده از symlink تشریح شده است. دستورات زیر با احراز هویت tempauth است .

گرفتن توکن

- برای اجرای دستورات ابتدا باید آدرس ذخیره سازی و توکن را از طریق دستور زیر بدست آورد:

```
curl -v -H 'X-Storage-User: test:tester' -H 'X-Storage-Pass: testing' http://127.0.0.1:8080/auth/v1.0
```

ایجاد symlink

- با دستور زیر شیء sym_obj_3/ c2 یک symlink به c1/f01 خواهد داشت:

```
curl -X PUT -H "Content-Length: 0" -H "X-Auth-Token: <token_from_above_command>" -H "X-Symlink-Target:
c1/f01" http://127.0.0.1:8080/v1/AUTH_test/c2/sym_obj_3
```

- با مراجعه به مسیر ذخیره سازی هر سه نسخه شی مشاهده میشود که شیء مورد نظر با **اندازه صفر** وجود دارد:

```
~$ ll /srv/2/node/sdb2/objects/885/9e6/dd5da836964026d5d1a5b2bbea5a89e6/
total 4
drwxr-xr-x 2 user user 35 Dec  5 10:12 ./
drwxr-xr-x 3 user user 46 Dec  5 10:12 ../
-rwxr-xr-x 1 user user  0 Dec  5 10:12 1638686573.90341.data*

~$ ll /srv/1/node/sdb1/objects/885/9e6/dd5da836964026d5d1a5b2bbea5a89e6/
total 4
drwxr-xr-x 2 user user 35 Dec  5 10:12 ./
drwxr-xr-x 3 user user 46 Dec  5 10:12 ../
-rwxr-xr-x 1 user user  0 Dec  5 10:12 1638686573.90341.data*

~$ ll /srv/4/node/sdb4/objects/885/9e6/dd5da836964026d5d1a5b2bbea5a89e6/
total 4
drwxr-xr-x 2 user user 35 Dec  5 10:12 ./
drwxr-xr-x 3 user user 46 Dec  5 10:12 ../
-rwxr-xr-x 1 user user  0 Dec  5 10:12 1638686573.90341.data*
```

GET شی symlink

- با دستور زیر میتوان شیء را GET کرد:

```
curl -H "X-Auth-Token: <token>" http://127.0.0.1:8080/v1/AUTH_test/c2/sym_obj_3
```

HEAD شی symlink

- با انجام دستور HEAD مشاهده میشود همانگونه که در بالا ذکر شد هدر **Content-Location** نیز تنظیم شده است:

```
~$ curl -I -H "X-Auth-Token: <token>" http://127.0.0.1:8080/v1/AUTH_test/c2/sym_obj_3
HTTP/1.1 200 OK
Content-Length: 6
Content-Type: application/octet-stream
Accept-Ranges: bytes
Last-Modified: Sun, 05 Dec 2021 05:41:31 GMT
Etag: b17ee36a3a70c8373b415675d5ee21df
```

```
X-Timestamp: 1638682890.68689
X-Object-Meta-Mtime: 1638253548.340000
Content-Location: /v1/AUTH_test/c1/f01
X-Trans-Id: txa57bdf90df564aea9e014-0061ac636f
X-Openstack-Request-Id: txa57bdf90df564aea9e014-0061ac636f
Date: Sun, 05 Dec 2021 06:59:59 GMT
```

- از طریق کد زیر نیز میتوان فراداده شیء را مشاهده نمود:

```python
from swift.obj.diskfile import read_metadata
import sys

datafile = sys.argv[1]

with open(datafile, 'rb') as fp:
    try:
        metadata = read_metadata(fp)
        print(metadata)
    except EOFError:
        print("Invalid metadata")
        sys.exit(1)
```

- خروجی کد بالا به شکل زیر میباشد:

```
{
  'Content-Length': '0',
  'name': '/AUTH_test/c2/sym_obj_3',
  'X-Object-Sysmeta-Symlink-Target': 'c1/f01',
  'ETag': 'd41d8cd98f00b204e9800998ecf8427e',
  'X-Timestamp': '1638686573.90341',
  'X-Object-Sysmeta-Container-Update-Override-Etag': 'd41d8cd98f00b204e9800998ecf8427e;
symlink_target=c1/f01',
  'Content-Type': 'application/symlink'
}
```

GET مستقیم شی symlink

- با دستور زیر میتوان مستقیما خود شی symlink را GET کرد:

```
curl -H "X-Auth-Token: <token>" http://127.0.0.1:8080/v1/AUTH_test/c2/sym_obj_3?symlink=get
```

تنظیم فراداده برای شی symlink

- با دستور زیر میتوان برای symlink فراداده تنظیم کرد:

```
curl -X POST -H "X-Auth-Token: <token>" -H "X-Object-Meta-mymetadata: mystuff"
http://127.0.0.1:8080/v1/AUTH_test/c2/sym_obj_3
```

- با اجرای دستور بالا فایل meta در مسیر شی symlink ساخته میشود:

```
~$ ll /srv/2/node/sdb2/objects/885/9e6/dd5da836964026d5d1a5b2bbea5a89e6/
total 4
drwxr-xr-x 2 user user 64 Dec  5 14:18 ./
drwxr-xr-x 3 user user 46 Dec  5 10:12 ../
-rwxr-xr-x 1 user user  0 Dec  5 14:03 1638700396.30131.data*
-rwxr-xr-x 1 user user  0 Dec  5 14:18 1638701319.72296.meta*
```

```
~$ ll /srv/1/node/sdb1/objects/885/9e6/dd5da836964026d5d1a5b2bbea5a89e6/
total 4
drwxr-xr-x 2 user user 64 Dec  5 14:18 ./
drwxr-xr-x 3 user user 46 Dec  5 10:12 ../
-rwxr-xr-x 1 user user  0 Dec  5 14:03 1638700396.30131.data*
-rwxr-xr-x 1 user user  0 Dec  5 14:18 1638701319.72296.meta*


~$ ll /srv/4/node/sdb4/objects/885/9e6/dd5da836964026d5d1a5b2bbea5a89e6/
total 4
drwxr-xr-x 2 user user 64 Dec  5 14:18 ./
drwxr-xr-x 3 user user 46 Dec  5 10:12 ../
-rwxr-xr-x 1 user user  0 Dec  5 14:03 1638700396.30131.data*
-rwxr-xr-x 1 user user  0 Dec  5 14:18 1638701319.72296.meta*
```

- با استفاده از اسکریپ read_metadata.py (https://opengit.ir/smartlab/swift_utility/-/blob/master/read_metadata.py) میتوان فایل meta را خواند که خروجی آن به شکل زیر میباشد:

```
{
    'X-Timestamp': '1638701319.72296',
    'name': '/AUTH_test/c2/sym_obj_3',
    'X-Object-Meta-Mymetadata': 'mystuff'
}
```

- با مراجعه به مسیر شی هدف مشاهده میشود که فراداده برای آن تنظیم **نشده** است:

```
~$ ll /srv/3/node/sdb3/objects/1008/6f4/fc2be2c443bb226138231143ed90b6f4/
total 8
drwxr-xr-x 2 user user 35 Dec  5 09:11 ./
drwxr-xr-x 3 user user 46 Nov 30 15:51 ../
-rwxr-xr-x 1 user user  6 Dec  5 09:11 1638682890.68689.data*

~$ ll /srv/2/node/sdb2/objects/1008/6f4/fc2be2c443bb226138231143ed90b6f4/
total 8
drwxr-xr-x 2 user user 35 Dec  5 09:11 ./
drwxr-xr-x 3 user user 46 Nov 30 15:51 ../
-rwxr-xr-x 1 user user  6 Dec  5 09:11 1638682890.68689.data*


~$ ll /srv/1/node/sdb1/objects/1008/6f4/fc2be2c443bb226138231143ed90b6f4/
total 8
drwxr-xr-x 2 user user 35 Dec  5 09:11 ./
drwxr-xr-x 3 user user 46 Nov 30 15:51 ../
-rwxr-xr-x 1 user user  6 Dec  5 09:11 1638682890.68689.data*
```

مشاهده اطلاعات symlink_path

- با اجرای دستور زیر میتوان اطلاعات symlink_path را مشاهده نمود:

```
curl -i -X GET -H "X-Auth-Token: <token>" http://127.0.0.1:8080/v1/AUTH_test/c2?format=json
```

- خروجی دستور بالا به شکل زیر خواهد بود:

```
HTTP/1.1 200 OK
Content-Length: 369
X-Container-Object-Count: 2
Accept-Ranges: bytes
X-Storage-Policy: gold
Last-Modified: Sun, 05 Dec 2021 06:29:07 GMT
X-Container-Bytes-Used: 6
X-Timestamp: 1638274860.07922
```

```
Content-Type: application/json; charset=utf-8
X-Trans-Id: txd03faaaeac544516b0ef4-0061ac9f12
X-Openstack-Request-Id: txd03faaaeac544516b0ef4-0061ac9f12
Date: Sun, 05 Dec 2021 11:14:26 GMT

{
    "hash": "d41d8cd98f00b204e9800998ecf8427e",
    "name": "sym_obj_3",
    "bytes": 0,
    "last_modified": "2021-12-05T10:48:39.722960",
    "content_type": "application/symlink",
    "symlink_path": "/v1/AUTH_test/c1/f01"
}
```

احراز هویت با keystone

گرفتن توکن

- برای اجرای دستورات ابتدا باید آدرس ذخیره سازی و توکن را از طریق دستور زیر بدست آورد:

```
curl -i --location --request POST 'http://localhost:5000/v3/auth/tokens' --header 'Content-Type:
application/json' --data-raw '{ "auth": {
    "identity": {
      "methods": ["password"],
      "password": {
        "user": {
          "name": "admin",
          "domain": { "id": "default" },
          "password": "ADMIN_PASS"
        }
      }
    },
    "scope": {
      "project": {
        "name": "admin",
        "domain": { "id": "default" }
      }
    }
  }
}'
```

ایجاد symlink

- با دستور زیر شیء sym_obj_3/ c2 یک symlink به c1/f01 خواهد داشت:

```
curl --location --request PUT 'http://localhost:8080/v1/AUTH_6a6816199c6548119ddf1c65aa67e934/c2/sym_obj_3'
\
--header 'Content-Type: application/json' \
--header 'X-Auth-Token: gAAAAABlFCARVAiEq0jyLSag-FJmcjtDnYlnPL6dyclCSwi7yWz-
o89Fg4JN4Z7KvfORuBogrSRSyxbUBYr7BoBArSUR9y6DBANlhc-
nIdiGl95IGMM4wFMkV8kyj8mtHF4jPZax1cVihckdd3FgkaJMpaRFgHVV8P5f0jHZQX5jf9hZtdDfWGU' \
--header 'X-Symlink-Target: c1/f01' \
--header 'Content-Length: 0' \
--data ''
```

GET شی symlink

- با دستور زیر میتوان شیء را GET کرد:

```
curl -H "X-Auth-Token: gAAAAABlL2dCQBbONCN7GmEL_CnS96IVhUMcVjfV8-tWUBiBEfetDu1wlMgmXL_lkc_ONC-
cePOofFQlvbMBSKf4NdRhIBjV_T3rkIF6TaRtcQVqY"
'http://localhost:8080/v1/AUTH_278b2760978c4fde88146992e6cb7d8f/c2/'
```

HEAD شی symlink

- با انجام دستور HEAD مشاهده میشود همانگونه که در بالا ذکر شد هدر **Content-Location** نیز تنظیم شده است:

```
curl -I -H "X-Auth-Token: gAAAAABlL2dCQBbONCN7GmEL_CnS96IVhUMcVjfV8-tWUBiBEfetDu1wlMgmXL_lkc_ONC-
cePOofFQlvbMBSKf4NdRhIBjV_T3rkIF6TaRtcFmO8irb1uTBROHSgyM2TSAyUDaT2gMii0atpYLl_ztoBPK1CX8zc19UBOzq4N5NTE6vqVy
QVqY" 'http://localhost:8080/v1/AUTH_278b2760978c4fde88146992e6cb7d8f/c2/'
```

GET مستقیم شی symlink

- با دستور زیر میتوان مستقیما خود شی symlink را GET کرد:

```
curl -H "X-Auth-Token: gAAAAABlL2dCQBbONCN7GmEL_CnS96IVhUMcVjfV8-tWUBiBEfetDu1wlMgmXL_lkc_ONC-
cePOofFQlvbMBSKf4NdRhIBjV_T3rkIF6TaRtcFmO8irb1uTBROHSgyM2TSAyUDaT2gMii0atpYLl_ztoBPK1CX8zc19UBOzq4N5NTE6vqVy
QVqY" http://localhost:8080/v1/AUTH_278b2760978c4fde88146992e6cb7d8f/c2/sym_obj_3?symlink=get
```

مشاهده اطلاعات symlink_path

- با اجرای دستور زیر میتوان اطلاعات symlink_path را مشاهده نمود:

```
curl -H "X-Auth-Token: gAAAAABlL2pm-K-MimyOgBrbIrI1uOKcOuMsyLpu1E1O1zadjTfJoPornSE62RFbH8K8qUmS6TNT-
ea2bEOiCtOWuJBcC1gZZMl1SSHl1hZq07yRjSURCkXtt3dsxcwASVchOC2hxzBvf1PUsrsk-puX-Wo91_VR-97qTicGJwK27qJpveLAk28"
'http://localhost:8080/v1/AUTH_99068acf5e33455da0203a400546676f/c2?format=json
```

symlink و ACL

به منظور بررسی نحوه ایجاد symlink بین دو کیسه از یک حساب که سطح دسترسی هر کیسه به یک کاربر محدود شده است، سناریوی زیر دنبال شد.

- ابتدا در فایل پیکربندی پروکسی (proxy-server.conf) دو کاربر test:tester4 و test:tester5 به صورت زیر ایجاد شد:

```
[filter:tempauth]
use = egg:swift#tempauth
user_admin_admin = admin .admin .reseller_admin
user_test_tester = testing .admin
user_test_tester4 = testing4
user_test_tester5 = testing5
```

- سپس کیسه های c4 و c5 به ترتیب با سطح دسترسی زیر برای test:tester4 و test:tester5 ایجاد شد. لازم به ذکر است که با توکن مربوط به test:tester ایجاد کیسه c4 (درج شی f01) و c5 و مجوزهای دسترسی به شکل زیر انجام شد.

```
curl -X POST -i -H "X-Auth-Token: AUTH_tkd40501d8a0454fb8b0670b95f2a3e55f" -H 'X-Container-Read:
test:tester4' http://127.0.0.1:8080/v1/AUTH_test/c4
```

```
curl -X POST -i -H "X-Auth-Token: AUTH_tkd40501d8a0454fb8b0670b95f2a3e55f" -H 'X-Container-Read:
test:tester5' http://127.0.0.1:8080/v1/AUTH_test/c5
```

- در ادامه با توکن test:tester یک symlink بین c5/sym_c4_f01 و c4/f01 ایجاد شد.

```
curl -X PUT -H "Content-Length: 0" -H "X-Auth-Token: AUTH_tkd40501d8a0454fb8b0670b95f2a3e55f" -H "X-Symlink-
Target: c4/f01" http://127.0.0.1:8080/v1/AUTH_test/c5/sym_c4_f01
```

- سپس مشاهده شد که کاربر test:tester5 با توکن خود امکان لیست گرفتن از کیسه c5 را دارد.

```
~$ curl -H "X-Auth-Token: AUTH_tk8fa64ee0cc7b40c4ad55325aa8d25852" http://127.0.0.1:8080/v1/AUTH_test/c5/
sym_c4_f01
```

- اما امکان GET شی sym_c4_f01 را ندارد.

```
~$ curl -H "X-Auth-Token: AUTH_tk8fa64ee0cc7b40c4ad55325aa8d25852"
http://127.0.0.1:8080/v1/AUTH_test/c5/sym_c4_f01
<html><h1>Forbidden</h1><p>Access was denied to this resource.</p></html>
```

- همچنین مشاهده شد که کاربر test:tester4 با توکن خود بدلیل داشتن مجوز دسترسی، امکانGET شی f01 را دارد.

```
~$ curl -H "X-Auth-Token: AUTH_tkc0dd71ac7af549499b48140bbf6d9614" http://127.0.0.1:8080/v1/AUTH_test/c4/f01
salam
```

- در ادامه به منظور فراهم آوردن امکان GET شی sym_c4_f01 توسط کاربر test:tester5 با توکن خود، با استفاده از توکن test:tester مجوز دسترسی ایجاد شد.

```
~$ curl -X POST -i -H "X-Auth-Token: AUTH_tkd40501d8a0454fb8b0670b95f2a3e55f" -H 'X-Container-Read:
test:tester5' http://127.0.0.1:8080/v1/AUTH_test/c4
```

- در نهایت مشاهده شد که کاربر test:tester5 با توکن خود امکان GET شی sym_c4_f01 را دارد.

```
~$ curl -H "X-Auth-Token: AUTH_tk8fa64ee0cc7b40c4ad55325aa8d25852"
http://127.0.0.1:8080/v1/AUTH_test/c5/sym_c4_f01
salam
```

پانویس

1. https://docs.openstack.org/swift/latest/middleware.html#symlink
2. probe tests

منابع

https://github.com/openstack/swift/blob/master/swift/common/middleware/symlink.py

برگرفته از «https://kateb.burna.ir/w/index.php?title=Symlink&oldid=45763»

مشارکت‌کنندگان: بهمن هاشمی, مهسا شادی, مبینا حاج ابراهیمی, مرتضی جوان, مرتضی شهبازی, پانیذ یزدان‌مهر

این صفحه آخرین‌بار در ۳۰ مهر ۱۴۰۲ ساعت ۰۹:۴۷ ویرایش شده‌است.