dlo ==> part들 무한정
Slo ==> multi part uploader종료이후는 더 이상 파트 업로드 불가.

파트들 업로드후 끝난다는 확인 // dlo는 계속 올릴수 있음. slo는 더 이상 못올림.

개요
스위프트는 업로드 된 단일 객체의 크기에 제한이 있습니다. 기본적으로 이것은 5GB입니다. 그러나 단일 객체의 다운로드 크기는 세그먼트 화의 개념에 따라 거의 무제한입니다. 큰 개체의 세그먼트가 업로드되고 다운로드 할 때 단일 개체로 연결된 모든 세그먼트를 보내는 특수 매니페스트 파일이 만들어집니다. 또한 세그먼트의 병렬 업로드가 가능하므로 업로드 속도가 훨씬 빨라집니다.

사용하기 swift
이 기능을 시험해 볼 수있는 가장 빠른 방법 swift은 python-swiftclient 라이브러리에 포함 된 Swift Tool 을 사용하는 것 입니다. 이 -S 옵션을 사용하여 큰 파일을 분할 할 때 사용할 세그먼트 크기를 지정할 수 있습니다 .

이렇게하면 large_file이 1G 세그먼트로 분할되고 해당 세그먼트를 병렬로 업로드하기 시작합니다. 모든 세그먼트가 업로드되면 swift매니페스트 파일을 만들어 세그먼트를 다운로드 할 수 있습니다.

이제 다음 swift명령은 전체 대형 객체를 다운로드합니다.

이렇게하면 large_file이 1G 세그먼트로 분할되고 해당 세그먼트를 병렬로 업로드하기 시작합니다. 모든 세그먼트가 업로드되면 swift매니페스트 파일을 만들어 세그먼트를 다운로드 할 수 있습니다.

이제 다음 swift명령은 전체 대형 객체를 다운로드합니다.

세그먼트와 매니페스트를 직접 처리하지 않고 HTTP 요청으로 직접 처리 swift할 수도 있습니다. 다른 개체처럼 세그먼트를 업로드 할 수 있으며 매니페스트는 추가 X-Object-Manifest헤더 가있는 0 바이트 (적용되지 않음) 파일입니다 .

모든 오브젝트 세그먼트는 동일한 컨테이너에 있어야하고, 공통 오브젝트 이름 접두어가 있어야하며, 연결해야하는 순서로 정렬되어야합니다. 객체 명은, UTF-8 바이트 캐릭터 라인으로서 사전 식으로 소트됩니다. 매니페스트 파일과 동일한 컨테이너에있을 필요는 없으므로 위에서 설명한대로 컨테이너 목록을 깨끗하게 유지하는 것이 좋습니다 swift.

매니페스트 파일은 추가 헤더 가있는 단순히 0 바이트 (적용되지 않음) 파일이며 , 여기서는 개체 세그먼트가있는 컨테이너 이며 모든 세그먼트의 공통 접두사입니다.X-Object-Manifest: <container>/<prefix><container><prefix>

먼저 모든 세그먼트를 업로드 한 다음 매니페스트를 만들거나 업데이트하는 것이 가장 좋습니

다. 이렇게하면 업로드가 완료 될 때까지 전체 개체를 다운로드 할 수 없게됩니다. 또한 새 세그먼트 집합을 두 번째 위치에 업로드 한 다음이 새 위치를 가리 키도록 매니페스트를 업데이트 할 수 있습니다. 새 세그먼트를 업로드하는 동안 원본 매니페스트를 사용하여 첫 번째 세그먼트 집합을 계속 다운로드 할 수 있습니다.

POST 요청을 사용하여 매니페스트 개체를 업데이트하는 경우 개체가 매니페스트 개체 X-Object-Manifest로 계속 동작하도록 헤더를 포함해야합니다.

매니페스트 파일에는 내용이 없어야합니다. 그러나 이것은 시행되지 않습니다. 매니페스트 경로 자체가에 지정된 컨테이너 / 접두사를 준수하고 X-Object-Manifest매니페스트에 일부 콘텐츠 / 데이터가있는 경우 세그먼트로 간주되며 매니페스트의 콘텐츠는 연결된 GET 응답의 일부가됩니다. 연결 순서는 세그먼트 이름을 정렬 할 때 반환되는 순서에 따라 연결 순서가 따르는 일반적인 DLO 논리를 따릅니다.

Here’s an example using curl with tiny 1-byte segments:

```
# First, upload the segments
curl -X PUT -H 'X-Auth-Token: <token>'        http://<storage_url>/container/
myobject/00000001 --data-binary '1'
curl -X PUT -H 'X-Auth-Token: <token>'        http://<storage_url>/container/
myobject/00000002 --data-binary '2'
curl -X PUT -H 'X-Auth-Token: <token>'        http://<storage_url>/container/
myobject/00000003 --data-binary '3'

# Next, create the manifest file
curl -X PUT -H 'X-Auth-Token: <token>'        -H 'X-Object-Manifest: container/
myobject/'        http://<storage_url>/container/myobject --data-binary ''

# And now we can download the segments as a single object
curl -H 'X-Auth-Token: <token>'        http://<storage_url>/container/myobject
```

Static Large Objects

실습 > swift-init proxy start (https://gist.github.com/drewkerrigan/2876196)

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

dlo.py ==> put 시 dlo.py slo.py 를 항상 거침

```
khoj@ubuntu0411:~$ curl -i -X PUT -H 'X-Auth-Token:
AUTH_tkf34921ca4c7d4439a329e88b484cfbec'  http://127.0.0.1:8080/v1/
AUTH_test/swift_0531
HTTP/1.1 201 Created

khoj@ubuntu0411:~$ curl -i -X PUT -H 'X-Auth-Token:
```

AUTH_tkf34921ca4c7d4439a329e88b484cfbec' http://127.0.0.1:8080/v1/AUTH_test/swift_0530
HTTP/1.1 202 Accepted

```
__call__(self, env, start_response):
"""WSGI entry point"""
req = Request(env)
try:
    vrs, account, container, obj = req.split_path(4, 4, True)
except ValueError:
    return self.app(env, start_response)

if ((req.method == 'GET' or req.method == 'HEAD') and
req.params.get('multipart-manifest') != 'get'):
    return GetContext(self, self.logger).\
        handle_request(req, start_response)
elif req.method == 'PUT':
    error_response = self._validate_x_object_manifest_header(req)
    if error_response:
        return error_response(env, start_response)
return self.app(env, start_response)
```

그냥 업로드

curl -i -X PUT --data-binary '1' -H "X-Auth-Token:AUTH_tkf34921ca4c7d4439a329e88b484cfbec" http://127.0.0.1:8080/v1/AUTH_test/swift_0531/000001
 521 curl -i -X PUT --data-binary '2' -H "X-Auth-Token: AUTH_tkf34921ca4c7d4439a329e88b484cfbec" http://127.0.0.1:8080/v1/AUTH_test/swift_0530/000002
 522 curl -i -X PUT --data-binary '3' -H "X-Auth-Token: AUTH_tkf34921ca4c7d4439a329e88b484cfbec" http://127.0.0.1:8080/v1/AUTH_test/swift_0530/000003
 523 curl -v -H 'X-Auth-Token: AUTH_tkf34921ca4c7d4439a329e88b484cfbec' http://127.0.0.1:8080/v1/AUTH_test/swift_0530/

끝났다는 신호 (dlo는 이것만 봄 : put 인데 x-object-manifest 만 찾음.)
 550 curl -i -X PUT -H "X-Auth-Token: AUTH_tkf34921ca4c7d4439a329e88b484cfbec" -H "X-Object-Manifest: AUTH_test/mydlocontainer/" http://127.0.0.1:8080/v1/AUTH_test/swift_0531/ --data-binary ''

가져오기
551 curl -H 'X-Auth-Token: AUTH_tkf34921ca4c7d4439a329e88b484cfbec' http://127.0.0.1:8080/v1/

AUTH_test/swift_0530/

///Pycharm setting ///