CHAPTER 16

Online storage

Amazon S3 and comparable services are intriguing players to recently enter the world of online storage. As we get produce more digital content to share, we need to store that content in an accessible place to others. Moreover, if you are building a service for others to use and need to have access to lots of storage to use, it would be valuable to be able to scale that storage up quickly without lots of upfront capital investment in storage hardware.

Amazon S3 is the poster child in the arena of offering online storage – and hence is the primary focus of this chapter. I wll also cover some other websites that offer similar offerings but with important twists. For instance, some of these services are meant to be used as backup services and not really for serving up digital objects for web applications.

Some of these services have APIs, which makes them highly mashable. They include:

* Amazon S3
* Box.net (http://box.net/) with its API[1]
* Streamload (http://www.streamload.com/) with its API[2]
* Omnidrive (http://www.omnidrive.com/) and its API[3]
* openomy (http://www.openomy.com/about.html) and its API[4]

This chapter shows the basics of using Amazon S3 and box.net and compares the two systems.

Amazon S3

Amazon S3 (Amazon's "Simple Storage Service"), is described in the following way:[5]

> *Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, fast, inexpensive data*

---

1 http://enabled.box.net/docs

2 http://www.mediamax.com/webservices/omnDocumentation.aspx

3 http://dev.omnidrive.com/HomePage

4 http://documentation.openomy.com/index.php/Main_Page

5 http://aws.amazon.com/s3 – which redirects to http://www.amazon.com/gp/browse.html?node=16427261

> *storage infrastructure that Amazon uses to run its own global network of web sites.*

There is no direct user interface for S3 – S3 is meant as technical infrastructure upon which developers can build services. The only interface to S3 provided by amazon.com is a web services API. You can access S3 through its REST or SOAP interfaces directly or via third-party language-specific API kits that use the REST or SOAP interface. Using the API kits generally make accessing S3 easier, providing that they are well documented and cover the parts of the API that you care about. However, they do not shield you from all the subtleties of the underlying API. For instance, the authorization methods are not fully implemented in the API libraries we will examine in this chapter.)

In the following sections, I will guide you through how to get started with Amazon S3, sketching out how to use the API and refering you to detailed documentation of the API as appropriate.

Rationale for S3

Why use Amazon S3? (These arguments may or may not apply to other online storage systems.)

* S3 is potentially than the alternative if your needs fit certain profiles. Don MacAskill, the CEO of SmugMug has given some the deepest analysis. MacAskill estimates that SmugMug saves about $500K a year by using S3 over buying and maintaining the equivalent amount of storage.[6]
* S3 promises high scalability, both in volume and the rate of change of storage needs. You "pay as you go" and pay for what you use. That means you don't have to invest up front in buying the maximum amount of storage you think you will need. The utitlity model lowers the barrier to entry to a level that a relatively poor individual can afford to create a Web 2.0 application.
* S3 provides the ability to handle strong peaks in demand.
* Presumably, robustness is part of the picture since amazon.com claims that it runs its own infrastructure on S3 – and therefore you can expect reliability similar to that of amazon.com itself.

For specific examples of how S3 is being used, see:

* `http://jeremy.zawodny.com/blog/archives/007641.html` for a list of backup services or tools that use S3.
* http://solutions.amazonwebservices.com/connect/kbcategory.jspa?categoryID=66

Conceptual Structure of Amazon S3

To get started, read the core concepts are documented at:

`http://docs.amazonwebservices.com/AmazonS3/2006-03-01/CoreConcepts.html`

---

[6] http://blogs.smugmug.com/don/2006/11/10/amazon-s3-show-me-the-money/

At its heart, S3 is a coneptually simple. it lets you store "objects" in "buckets". An object is associated with a bucket via a "key". There are authentication and authorization schemes associated with S3 to grant you control over acess to the buckets and objects. You can associate some amount of metadata (in the form of key-value pairs) with objects.

A few more important points:

* Since a bucket name is global across the S3 service (akin to domain name), each developer account can have up to only 100 buckets at any one time. "[B]ucket names can only contain alphanumeric characters, underscore (_), period (.), and dash (-). Bucket names must be between 3 and 255 characters long. Buckets with names containing uppercase characters are not accessible using the virtual hosting method."[7]
* Objects are made up of object data and associated metadata. The value of an object can grow up to 5 gigabytes of data.[8]
* A key is like a filename for an object and must be unique within a bucket. Its UTF-8 encoding must be at most 1024 bytes long.
* You use "prefixes" and "delimiters" in keys to simulate a hierarchical (folder within folder-like) organization within buckets.[9]
* Metadata associated with objects: For both REST and SOAP requests to S3, user metadata size is limited to 2k bytes. They are structured as key-value pairs.[10]
* There is an authentication and authorization system in place. You can have fine-grained authorization (where you can associate permissions with specific users) or with larger preset groups (the owner, everyone, or authenticated users). Permissions are "read", "write", or "full control."[11]
* You can retrieving a .torrent file for any publicly available object by adding a "?torrent" query string parameter at the end of the REST GET request for the object.[12]
* There is "virtual hosting of buckets" that allows one to associate your own non-amazon domain name with a S3 bucket.[13] For example, objects accessible at the URL

```
http://s3.amazonaws.com/[bucket]/[key]
```

is also accessible at (providing the bucket name has no uppercase characters)

```
http://[bucket].s3.amazonaws.com/[key]
```

For example,

```
http://s3.amazonaws.com/raymondyee/858Xtoc___.pdf
```

---

7 http://docs.amazonwebservices.com/AmazonS3/2006-03-01/UsingBucket.html

8 http://docs.amazonwebservices.com/AmazonS3/2006-03-01/UsingObjects.html

9 http://docs.amazonwebservices.com/AmazonS3/2006-03-01/ListingKeysHierarchy.html

10 http://docs.amazonwebservices.com/AmazonS3/2006-03-01/UsingMetadata.html

11 http://docs.amazonwebservices.com/AmazonS3/2006-03-01/UsingAccessControl.html

12 http://docs.amazonwebservices.com/AmazonS3/2006-03-01/S3TorrentRetrieve.html

13 http://docs.amazonwebservices.com/AmazonS3/2006-03-01/VirtualHosting.html

is accessible as

```
http://raymondyee.s3.amazonaws.com/858Xtoc___.pdf
```

SIGNING UP FOR AMAZON AWS

You need an AWS key and secret to use S3. The homepage for AWS is:

```
http://aws.amazon.com
```

The documentation for the E-commerce services part of Amazon AWS is found at

```
http://www.amazon.com/gp/browse.html?node=12738641
```

You need to sign up for an Amazon AWS acocunt to get access keys:
```
http://www.amazon.com/gp/aws/registration/registration-form.html
```

To get your keys if you are already a member:

```
http://aws-
portal.amazon.com/gp/aws/developer/account/index.html/?ie=UTF8&action=access-key
```

You get an Access Key ID and a secret access key -- you can also use a X.509 certificate.

End Sidebar

The Firefox S3 Extension Gets You Started with S3

As I have argued throughout the book, it's very helpful to learn an application well before diving into its API. That S3 has no built-in user interface means that you have to either program S3 yourself right from the start – or make use of someone else's user interface. I recommend installing the *S3 Firefox Explorer Firefox Add-on* to get a UI to not only manage your files on S3 and to learn how S3 works. You can get the extension from:

```
http://www.rjonna.com/ext/s3fox.php
```

The extension is a great learning tool for S3. Using it, for instance, you can quickly create a bucket and populate that bucket with an object. You can then write some code to test whether you can read the list of buckets and what is contained in them. Without a UI tool such as the Firefox extension, you would first have to get your code working to populate the buckets. (See Figure 16-1 for a screenshot of S3 Firefox Explorer.)

Insert Figure 16-1.

Figue 16-1. S3 Firefox Explorer. The left panel is an explorer-like interface to your desktop. The right panel shows you your buckets and objects within the folders. You can edit the ACL (access control list) for each object. You can also copy the URL for an object.

Similarly, you might want to install S3Drive in Microsoft Windows to have fairly seamless integration in Windows. The S3 service makes S3 looks like partition on a local hard drive.[14]

The AWSZone.com Scratchpad as an API Explorer

The AWSZone.com tool is very much like the Flickr API Explorer – very, very useful indeed. It allows you to make calls to the AWS through entering parameters in the web broser. awszone.com covers a wide range of amazon.com APIs, not just S3.

To use it, to to

```
http://www.awszone.com/
```

and click on "Amazon S3".

You can try out the ListAllMyBuckets operation at

```
http://www.awszone.com/scratchpads/aws/s3.us/ListAllMyBuckets.aws
```

by filling out your `Access Key ID` and `Secret Access Key`. Note that AWSZone uses the SOAP inteface for S3.

S3 REST Interface

The S3 REST interface is truly RESTful -- you think in terms of resources: service (to get a list of all your buckets), bucket, object -- and they have standard methods. See

```
http://docs.amazonwebservices.com/AmazonS3/2006-03-01/RESTAPI.html
```

for a list of resources and methods.

Using the REST interface is a bit tricky because of:

* how authentication is handled, specifically, how the signature is calculated.[15]
* the use of authorization control lists to handle authorization[16]
* how metadata is implemented.

In this section, I will show one specific, relatively, simple GET example to demonstrate the use of the REST interface. Let's first use the "*Query String Request Authentication Alternative*" which doesn't require the use of HTTP Authorization headers. As the documentation indicates, "[t]he practice of signing a request and giving it to a third-party for execution is suitable only for simple object GET requests."

The REST endpoint is

```
http:// host s3.amazonaws.com
```

You need three query parameters:

* `AWSAccessKeyId`

---

[14] htt://www.suchwerk.net/sodcms_S3Drive_Installation.htm

[15] http://docs.amazonwebservices.com/AmazonS3/2006-03-01/RESTAuthentication.html

[16] http://docs.amazonwebservices.com/AmazonS3/2006-03-01/RESTAccessPolicy.html

* `Expires`: "The time when the signature expires, specified as the number of seconds since the epoch (00:00:00 UTC on January 1, 1970)"
* `Signature`: "The URL encoding of the Base64 encoding of the HMAC-SHA1 of StringToSign, as defined below."

Let's use the example data given in the documents and generate some Python and PHP code to demonstrate how to calculate these parameters.

AWSAccessKeyId 0PN5J17HBGZHT7JJ3X82

AWSSecretAccessKey uV3F3YluFJax1cknvbcGwgjvx4QpvB+leU8dUj2o

Expires 1141889120

Host johnsmith.s3.amazonaws.com

Key /photos/puppy.jpg

Pseudo-code for the calculation is:

```
Signature = URL-Encode( Base64( HMAC-SHA1( UTF-8-Encoding-Of( StringToSign ) ) ));
StringToSign = HTTP-VERB + "\n" +  Content-MD5 + "\n" +  Content-Type + "\n" +
Expires + "\n" + CanonicalizedAmzHeaders + "CanonicalizedResource;
```

We're told that the Signature should be `rucSbHOyNEcP9oM2XNlouVI3BH4%3D`

in the case in which StringToSign is
`"GET\n\n\n1175139620\n/johnsmith/photos/puppy.jpg"` and the request is

```
GET /photos/puppy.jpg?AWSAccessKeyId=0PN5J17HBGZHT7JJ3X82&
    Signature=rucSbHOyNEcP9oM2XNlouVI3BH4%3D&
    Expires=1175139620 HTTP/1.1
Host: johnsmith.s3.amazonaws.com
```

In this case we "won't provide a Content-MD5 or a Content-Type header, nor will it set any x-amz- headers, so those parts of the StringToSign are left blank."

Python code to do validate this calculation:

```
import sha, hmac, base64, urllib

AWSAccessKeyId = "0PN5J17HBGZHT7JJ3X82"
AWSSecretAccessKey = "uV3F3YluFJax1cknvbcGwgjvx4QpvB+leU8dUj2o"
string_to_sign = "GET\n\n\n1175139620\n/johnsmith/photos/puppy.jpg"
sig = base64.encodestring(hmac.new(AWSSecretAccessKey, string_to_sign,
sha).digest()).strip()
print urllib.urlencode({'Signature':sig})
```

which produces

```
Signature=rucSbHOyNEcP9oM2XNlouVI3BH4%3D
```

You can learn to produce the same logic PHP from studying:

```
http://blog.pairwise.com/2006/08/28/a-simple-amazon-php-s3-class/
```

or downloading the PHP code at

```
http://developer.amazonwebservices.com/connect/entry.jspa?categoryID=47&externalID=1
26
```

There are some preliminary PEAR libraries that need to be installed:

* Crypt_HMAC (pear install Crypt_HMAC)
* HTTP_Request

After you get those libraries installed, Unfortunately, because the code requires sending the secret over the wire, I can't recommend using it on a remote host. Run it on your own machine.

Listing Buckets using the REST Interface

Let's get the list of buckets in Python:

```
def listBuckets(AWSAccessKeyId,AWSSecretAccessKey):
    """
    use the REST interface to get the list of buckets
    """
    import time
    # give an hour for the request to expire (3600s)
    expires = int(time.time()) + 3600
    string_to_sign = "GET\n\n\n%s\n/" % (expires)
    print expires, string_to_sign
    sig = base64.encodestring(hmac.new(AWSSecretAccessKey, string_to_sign,
sha).digest()).strip()
    request = "http://s3.amazonaws.com?AWSAccessKeyId=%s&Expires=%s&Signature=%s" %
\
               (AWSAccessKeyId, expires, sig)
    return request

if __name__ == "__main__":
    AWSAccessKeyId='[AWSAccessKeyID]'
    AWSSecretAccessKey = '[SecretAccessKey]'
    print listBuckets(AWSAccessKeyId,AWSSecretAccessKey)
```

SOAP Inteface to S3

The following code sample works to illustrate the use of the SOAP interface to S3.

```
# list buckets for Amazon s3

AWSAccessKeyId='[AWSAccessKeyID]'
AWSSecret = '[SecretAccessKey]'

from SOAPpy import WSDL

import sha

def calcSig(key,text):
    import hmac, base64
    sig = base64.encodestring(hmac.new(key, text, sha).digest()).strip()
    return sig
```

```
def ListMyBuckets(s):
    from time import gmtime,strftime
    method = 'ListAllMyBuckets'
    ts = strftime("%Y-%m-%dT%H:%M:%S.000Z", gmtime())
    text = 'AmazonS3' + method + ts
    sig = calcSig(AWSSecret,text)
    print "ListMyBuckets: ts,text,sig->", ts, text, sig
    return s.ListAllMyBuckets(AWSAccessKeyId=AWSAccessKeyId,
Timestamp=ts,Signature=sig)

def CreateBucket(s, bucketName):
    from time import gmtime,strftime
    method = 'CreateBucket'
    print 'method: ', method
    ts = strftime("%Y-%m-%dT%H:%M:%S.000Z", gmtime())
    text = 'AmazonS3' + method + ts
    sig = calcSig(AWSSecret,text)
    print "CreateBuckets: ts,text,sig->", ts, text, sig
    return s.CreateBucket(Bucket=bucketName, AWSAccessKeyId=AWSAccessKeyId,
Timestamp=ts,Signature=sig)

if __name__ == '__main__':
    s = WSDL.Proxy("http://s3.amazonaws.com/doc/2006-03-01/AmazonS3.wsdl")
    print ListMyBuckets(s)
```

Things that we can learn about S3 from this code

* You need to have an Amazon AWS Access Key ID and Secret Access Key – which you can get if you sign up for an account for Amazon AWS (See Chapter 7 or the Sidebar for more details.)
* Although S3 is accessible by REST or SOAP interfaces, this code uses SOAP and WSDL. The WSDL for the service at the time of writing is at `http://s3.amazonaws.com/doc/2006-03-01/AmazonS3.wsdl` -- you can get the location of the latest WSDL URL at: `http://aws.amazon.com/s3`
* Two methods are used in this code sample: `ListMyBuckets` and `CreateBuckets`. You can get the list of all the methods at: `http://www.awszone.com/scratchpads/aws/s3.us/index.aws` (The technical documentation is at `http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=48` which leads to `http://docs.amazonwebservices.com/AmazonS3/2006-03-01/`)
* One of the complicated aspects is to calculate a signature. You can use AWSZone to test whether you are correctly calculating your signature.

Amazon S3 API Kits

In this section, we look at libraries to S3 written in PHP and Python.

PHP

The following API kits are available:

* php-aws[17]
* s3.class.zip at neurofuzzy.net, which looks like a popular class implementation [18]
* edoceo's phps3tk [19]

In this section, we concentrate on how to use php-aws. You can access the source, using SVN. In the web browser, you can download the library from:

`http://php-aws.googlecode.com/svn/trunk/class.s3.php`

Documentation for the S3 class can be found at

`http://code.google.com/p/php-aws/wiki/S3Class`

The following blog entry introduces php-aws:

`http://sitening.com/blog/2007/01/30/introducing-php-aws/#comment-29430`

There are a few things to note about this library:

* its use of curl means that it is built to handle larger files.
* only the public-read or private ACL is currently implemented
* there is no implementation of user metadata for objects.

I've run into problems with the latter two, so let's focus on the first one (php-aws). How to use?

1. Download http://php-aws.googlecode.com/svn/trunk/class.s3.php to your favorite local PHP directory. (In my case: /home/rdhyee/phplib as `php-aws.s3.php`) You might need to have the Crypt_HMAC Pear module installed.

2. Try the following sample code to get you started:

```
<?php
ini_set(
  'include_path',
    ini_get( 'include_path' ) . PATH_SEPARATOR . "/home/rdhyee/pear/lib/php" .
PATH_SEPARATOR . "/home/rdhyee/phplib"
    );
require_once("php-aws.s3.php");

$key = "[AWSAccessKeyID]";
$secret = "[SecretAccessKey]";
```

---

[17] (http://code.google.com/p/php-aws/source) – specifically http://php-aws.googlecode.com/svn/trunk/class.s3.php

[18] http://neurofuzzy.net/2006/08/26/amazon-s3-php-class-update/ and http://www.neurofuzzy.net/wp-content/2006/03/s3.class.zip

[19] http://www.edoceo.com/creo/phps3tk/

```
$s3 = new S3($key,$secret);

// get list of buckets
$buckets = $s3->getBuckets();
print_r($buckets);

// if the bucket "mashupguidetest" doesn't exist, create it
$BNAME = "mashupguidetest";
if (! $s3->bucketExists($BNAME)) {
  $s3->createBucket($BNAME);
}

// get list of buckets again
$buckets = $s3->getBuckets();
print_r($buckets);
```

Python

Some Python-based S3 libraries are:

* boto[20]
* HanzoiArchive's S3 tools[21]
* BitBucket[22]

I would recommend looking at boto as a good choice of a library.

Box.net

See

`http://www.box.net/info`

The ToS for the API: http://enabled.box.net/terms/ which contains the conditions that I need to think a bit about before accepting:

> *With the Box Enabled platform, Box.net aims to make user-centric storage available to developers. This is different from other storage API platforms which provide application-specific storage. For example, while it's okay to develop a widget that will preview someone's music or photos, it's NOT okay to use Box.net as the back-end data store of your Web site. By general-purpose, we mean situations in which the access or storage of one's data from your application will be generally useful outside the context of your application. We realize that the distinction between "general-purpose" and non-general-purpose*

---

20 http://code.google.com/p/boto/

21 http://www.hanzoarchives.com/development-projects/s3-tools/

22 http://cheeseshop.python.org/pypi/BitBucket/0.4a

> *might be a little unclear since there are a lot of different possibilities, so if you're at all unclear, please email info@box.net with any questions you might have before you start coding.*

Box.net is free as long as you use up to only 1GB

According to

`http://enabled.box.net/docs`

there are three types of APIs for box.net

* SOAP
* XML Post
* REST

How to use this:

* Sign up for a key for a specific application at http://enabled.box.net/my-projects/editor/add/0/page-1-1
* Create a folder and upload a test file.
* Study http://enabled.box.net/docs/rest#getting_started

I don't know whether I'll work this out – there's a lot of work to do something simple.

Streamload

A free account gives you up to 25GB of storage.

I'm still working on learning the streamload API. Time permitting, I'll fill this out – otherwise, this will be a S3 only chapter.