The sys admin's daily grind: S3QL

Horror Pictures

Sys admin Charly has been an enthusiastic amateur photographer for many years. Recently, he started worrying about something happening to his rapidly expanding photo collection. Can the cloud save the day? *By Charly Kühnast*

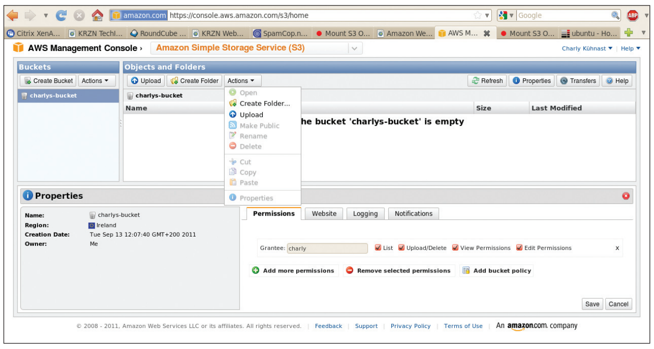
When I bought my first digital camera 10 years ago, backup wasn't an issue. The 2Mpx point-and-shoot box created JPGs so small I could back up my photo gallery to a couple of CDs. Today, I use raw format, and I can easily have 10 or 20GB of material on the card when I get back from a Sunday outing. Even though not all of this ends up in my collection, I still have a fairly substantial amount of material to deal with.

My works of art are stored on a small NAS box at home. But, I would additionally like to back them up somewhere outside my flood-endangered home. I was thinking of a cloud storage service like Amazon's S3, and I would like to encrypt my photos when I store them.

While I was shopping around for a tool to do this for me, I stumbled across the S3QL filesystem [1]. S3QL splits my data into small blocks, encrypts the blocks, fires them off into the cloud, and stores them in an S3 bucket (Figure 1). "Bucket" is Amazon-speak for a leased storage slot.

S3QL uses a SQLite database to remember which data is stored where, which helps dedupe the material. If you have two data blocks with the same content, only one is uploaded to the bucket; S3QL creates a pointer in the database for the other one. Also, copy and move operations only occur in the database and don't generate network traffic.

All of this work is transparent to the user. You simply mount the data in the bucket at some point on your filesystem, which takes a couple of steps. I started by creating a `~/.s3ql/authinfo` file in my home directory. The file contains the S3 access credentials and my encryption password. If you don't relish the idea of this lying around on your disk – which I can totally understand – you can simply omit the `authinfo` file and log in manually when you mount the filesystem. My `authinfo` file looks something like this:



charlys-bucket

he bucket 'charlys-bucket' is empty

Properties

Name: charlys-bucket
Region: Ireland
Creation Date: Tue Sep 13 12:07:40 GMT+200 2011
Owner: Me

Grantee: charly ✓ List ✓ Upload/Delete ✓ View Permissions ✓ Edit Permissions

© 2008 - 2011, Amazon Web Services LLC or its affiliates. All rights reserved. | Feedback | Support | Privacy Policy | Terms of Use | An amazon.com. company

Figure 1: The S3 admin interface; the bucket for S3QL exists but hasn't been filled.

```
backend s3 machine any login CKUEHNAST ⮐
  password FSLMPF42
storage-url s3://charlys-bucket ⮐
  password secret
```

The first line contains my access credentials for the Amazon S3 storage service. S3QL also supports other service providers. The second line contains the bucket name. S3QL needs the `secret` password to decrypt the encrypted data.

Mkfs.cloud

Before I use S3QL for the first time, I create the encrypted filesystem in my bucket:

```
mkfs.s3ql s3://charlys-bucket
```

The command prompts me for an encryption password. I type the password from the second line of the `authinfo` file. I can now mount the filesystem:

```
mount.s3ql s3://charlys-bucket /mnt/
```

Now I can use S3 storage like any other filesystem, and I can fire my photo gallery to the cloud with the help of a backup tool. However, you need to use S3QL to read what S3QL wrote. Other bucket management tools just show you encrypted data blocks. ■■■

INFO

[1] S3QL: *https://code.google.com/p/s3ql/*

AUTHOR

Charly Kühnast is a Unix operating system administrator at the Data Center in Moers, Germany. His tasks include firewall and DMZ security and availability. He divides his leisure time into hot, wet, and eastern sectors, where he enjoys cooking, freshwater aquariums, and learning Japanese, respectively.