Amazon S3 and Rails

Scott Patten - scott@ruboss.com

Amazon Web Services

Amazon Web Services

- Jeff Bezos wants other people to be able to create web apps without the pain he went through

Amazon Web Services

- Jeff Bezos wants other people to be able to create web apps without the pain he went through

Amazon Web Services

- Jeff Bezos wants other people to be able to create web apps without the pain he went through
- No fixed costs

Amazon Web Services

- Jeff Bezos wants other people to be able to create web apps without the pain he went through
- No fixed costs

Amazon Web Services

- Jeff Bezos wants other people to be able to create web apps without the pain he went through
- No fixed costs
- Your costs scale with your usage and (hopefully!) your profits.

S3

S3

- On-line storage using Amazon's hardware

S3

- On-line storage using Amazon's hardware
- Simple

S3

- On-line storage using Amazon's hardware
- Simple
- Reliable

S3

- On-line storage using Amazon's hardware
- Simple
- Reliable
- Scalable

S3

- On-line storage using Amazon's hardware
- Simple
- Reliable
- Scalable
- Flexible

S3

- On-line storage using Amazon's hardware
- Simple
- Reliable
- Scalable
- Flexible
- Pretty cheap

S3 for Rails

S3 for Rails

- Serving Assets

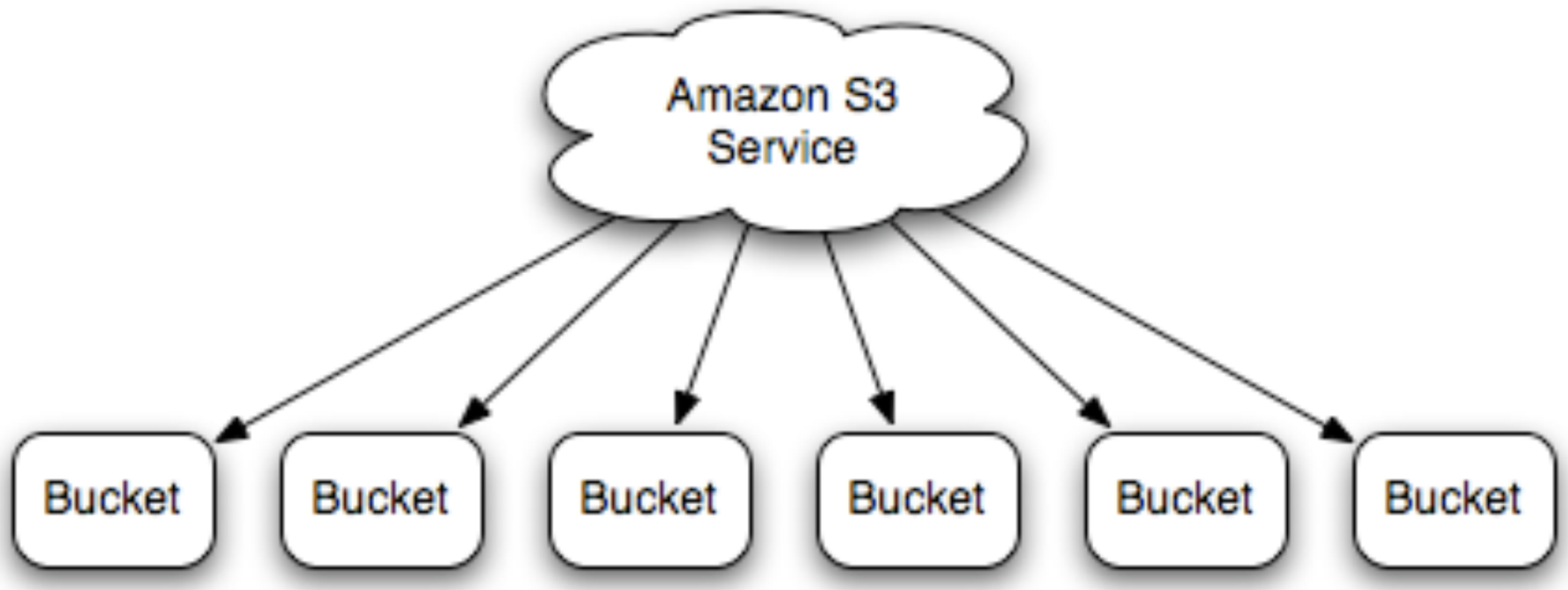
S3 for Rails

- Serving Assets
- Authenticated URLs

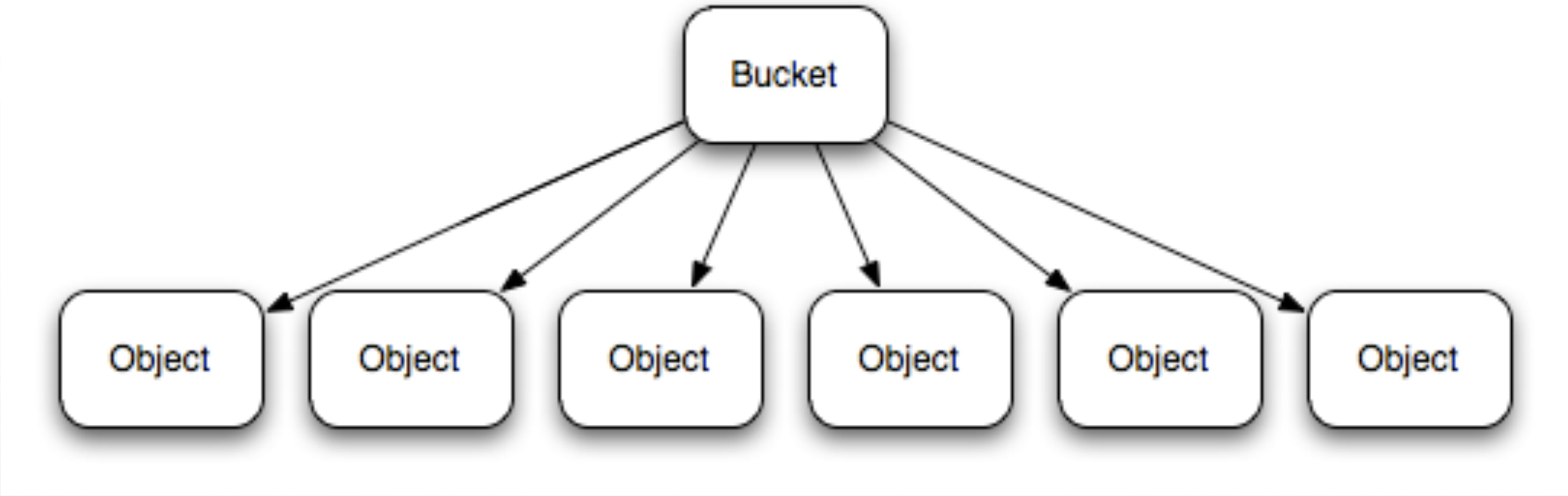
S3 for Rails

- Serving Assets
- Authenticated URLs
- Backup

S3's Architecture



S3's Architecture



S3 is RESTful

http://s3.amazonaws.com/<bucket_name>/<object_key>

http://s3.amazonaws.com/socialbandwagon/logo.png

Getting Started with S3

Sign up for S3:

http://aws.amazon.com/s3/

Getting Started with S3

```
sudo gem install aws-s3
```

Getting Started with S3

set up two environment variables:

`AMAZON_ACCESS_KEY_ID`

and

`AMAZON_SECRET_ACCESS_KEY`

Getting Started with S3

```
#!/usr/bin/env ruby
# create_bucket.rb
# Usage: create_bucket.rb <bucket> <file>
require 'rubygems'
require 'aws/s3'
include AWS::S3

AWS::S3::Base.establish_connection!(
    :access_key_id     => ENV['AMAZON_ACCESS_KEY_ID'],
    :secret_access_key => ENV['AMAZON_SECRET_ACCESS_KEY']
)

bucket = ARGV[0]
file = ARGV[1]

Bucket.create(bucket)
S3Object.store(file, File.open(file), bucket)
```

S3SH demo

Cool, how did they make IRB do that?

```ruby
#!/usr/bin/env ruby
s3_lib   = File.dirname(__FILE__) + '/../lib/aws/s3'
setup    = File.dirname(__FILE__) + '/setup'
irb_name = RUBY_PLATFORM =~ /mswin32/ ? 'irb.bat' : 'irb'

exec "#{irb_name} -r #{s3_lib} -r #{setup} --simple-prompt"
```

Assets on S3

S3 and Paperclip

```
class Book < ActiveRecord::Base

  has_many :authors

  ...

  has_attached_file :titlepage, :storage => :s3,
                                :s3_credentials => 'config/s3.yml',
                                :s3_permissions => 'public-read',
                                :bucket => S3_BUCKET,
                                :path => "titlepages/:id_:basename_:style.:extension",
                                :styles => {:small => "360x360>", :thumb => "60x60>"},
                                :default_url => "/titlepages/default_titlepage_:style.png"

  ...

end
```

Asset Hosting on S3

In environment.rb, config block:

```
config.action_controller.asset_host =
    "http://assets.sopobo.com"
```

Asset Hosting on S3

In environment.rb, config block:

```
config.action_controller.asset_host =
    "http://assets%d.sopobo.com"
```

Asset Hosting on S3

Create 4 buckets:

assets0.sopobo.com

assets1.sopobo.com

assets2.sopobo.com

assets3.sopobo.com

Asset Hosting on S3

Create CName entries for those buckets:

assets0.sopobo.com =>
assets0.sopobo.com.s3.amazonaws.com

Asset Hosting on S3

Upload everything in /public when you deploy.

Asset Hosting on S3

Asset Hosting on S3

You might want to automate that

Asset Hosting on S3

You might want to automate that

Asset host syncher:

Asset Hosting on S3

You might want to automate that

Asset host syncher:

http://spattendesign.com/projects/synching-your-amazon-s3-asset-host-using-capistrano

Authenticated URLs

The Problem

- Someone buys something
- You want them to be able to download it
- You want to give them a URL in an email
- The URL shouldn't last forever
- You want to track how many times a user has downloaded the file

Authenticated URL demo

Backups with S3

What should you back up?

Everything

Everything

- Data

Everything

- Data
- Code

Everything

- Data
- Code
- User created content

Everything

- Data
- Code
- User created content
- Your server configuration

General backup principles

Automate it

General backup principles

Test Recovery

General backup principles

Automate Recovery

General backup principles

Save Everything

General backup principles

DO IT!

Where should I backup to?

Where should I backup to?

- Off-site

Where should I backup to?

- Off-site
- Easy to automate

Where should I backup to?

- Off-site
- Easy to automate
- Reliable

Where should I backup to?

- Off-site
- Easy to automate
- Reliable
- Secure

Where should I backup to?

- Off-site
- Easy to automate
- Reliable
- Secure
- Cheap

Amazon S3

Backing up your DB

Don’t be these guys

Tech Gadgets Mobile Enterprise CrunchBase More ▼

TechCrunch

www.momentumdesignlab.com

About Advertise Archives Company Index Contact Jobs MashLogic Subscribe:

JournalSpace Drama: All Data Lost Without Backup, Company Deadpooled

by **Robin Wauters** on January 3, 2009 **139 Comments**

Solution #1

Solution #1

1. Use ec2-on-rails

Solution #1

1. Use ec2-on-rails
2. Go have a beer

Solution #1

1. Use ec2-on-rails
2. Go have a beer

http://github.com/pauldowman/ec2onrails

Solution #1

1. Use ec2-on-rails
2. Go have a beer

http://github.com/pauldowman/ec2onrails

http://pauldowman.com

Solution #2

Solution #2

Roll your own

Solution #2

Roll your own
(i.e. borrow Paul's code)

Full MySQL Backup

mysqldump

Full MySQL Backup

```
mysqldump --quick --single-transaction --create-options
-u<username> --flush-logs --master-data=2
--delete-master-logs -p'<mysql password>' <database name>
| gzip  >   <dump file>
```

perhaps we should
automate that

Full backup script

```
#!/usr/bin/env ruby
require "common"
begin
  FileUtils.mkdir_p @temp_dir
  dump_file = "#{@temp_dir}/dump.sql.gz"
  cmd = "mysqldump --quick --single-transaction --create-options " +
  "-u#{@mysql_user} --flush-logs --master-data=2 " +
  "--delete-master-logs"
  cmd += " -p'#{@mysql_password}'" unless @mysql_password.nil?
  cmd += " #{@mysql_database} | gzip > #{dump_file}"
  run(cmd)
  AWS::S3::S3Object.store(File.basename(dump_file),
                          open(dump_file),@s3_bucket)
ensure
  FileUtils.rm_rf(@temp_dir)
end
```

Incremental MySQL backups

Incremental MySQL backups

- MySQL creates something called Binary Logs

Incremental MySQL backups

- MySQL creates something called Binary Logs
- To backup, you just copy the binary logs somewhere.

Setting up binary logging

Put this line in my.cnf:

```
log_bin = /var/db/mysql/binlog/mysql-bin
```

Setting up binary logging

Give the user RELOAD and SUPER privileges:

```
GRANT RELOAD ON *.* TO 'user_name'@'%'
IDENTIFIED BY 'password';

GRANT SUPER ON *.* TO 'user_name'@'%'
IDENTIFIED BY 'password';
```

Incremental backup script

```
#!/usr/bin/env ruby
require "common"
begin
  FileUtils.mkdir_p @temp_dir
  execute_sql "flush logs"
  logs = Dir.glob("#{@mysql_bin_log_dir}/mysql-bin.[0-9]*").sort
  logs_to_archive = logs[0..-2] # all logs except the last
  logs_to_archive.each do |log|
    AWS::S3::S3Object.store(File.basename(log), open(log), @s3_bucket)
  end
  execute_sql "purge master logs to '#{File.basename(logs[-1])}'"
ensure
  FileUtils.rm_rf(@temp_dir)
end
```

Stick it in your cron

```
# Incremental backup every 10 minutes
*/10 * * * * root /usr/local/bin/incremental_backup.rb
# Full backup every day at 05:01
1 5 * * * root /usr/local/bin/full_backup.rb
```

Backing up your code

Backing up your code

== backing up your repository

Version Control Survey

What are people using?

Solution #1

Solution #1

- Use GitHub

Solution #1

- Use GitHub
- Go have a beer

Solution #1

- Use GitHub
- Go have a beer
- Have another. GitHub is awesome.

Solution #2

Backup SVN to S3

Backing up user contributed content

Backing up user contributed content

Grab all of the files and put them on S3

S3Sync

- Ruby script to back up a directory to an S3 bucket
- http://s3sync.net
- rsync-like syntax

The S3 Cookbook recipe

Get the code:

http://github.com/spatten/backing-up-your-app

get everything in code/sync_directory

configure it

Create a YAML file that looks something like this:

```
avatars:
  directory: /mnt/app/shared/images/avatars
  bucket: socialbandwagon-avatars-backup

pictures:
  directory: /mnt/app/shared/images/pictures
  bucket: socialbandwagon-pictures-backup
```

run it

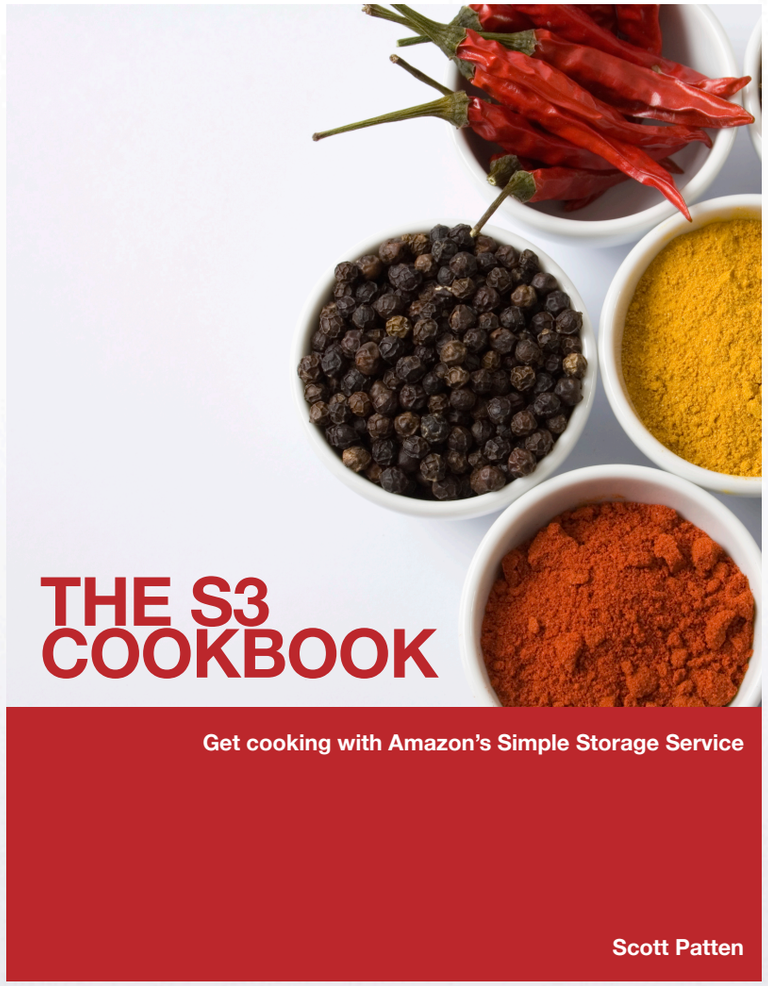
```
$> sync_multiple_directories.rb multi_sync.yml
```

stick it in your cron

```
15 4 * * * /mnt/app/current/script/sync_multiple_directories.rb \
/mnt/app/current/config/multi_sync.yml _>> \
/mnt/app/current/log/s3backup.log 2>&1
```

Pro Tip

Serve the files directly from S3

Thanks!

http://thes3cookbook.sopobo.com