S3 on Rails

Scott Patten
scott@ruboss.com www.ruboss.com
twitter: @scott_patten

Ruboss

Amazon Web Services

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

Amazon Web Services

No fixed costs

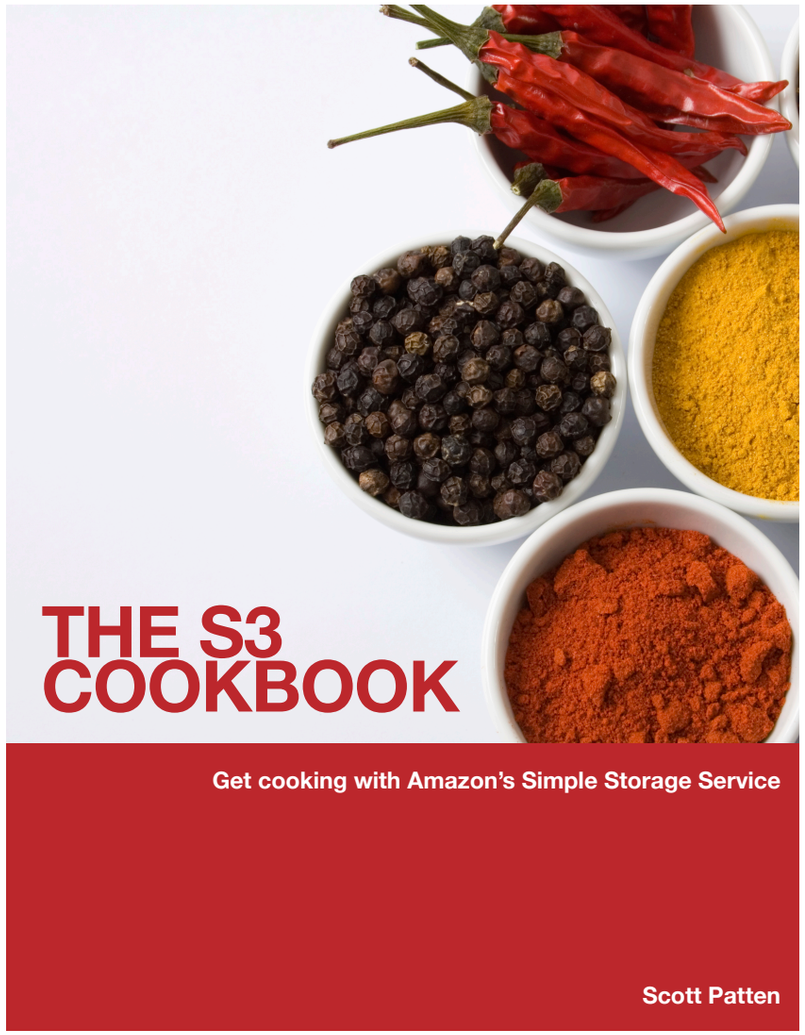
Amazon Web Services

Your costs scale with your usage and (hopefully!) your profits.

S3

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

The S3 Cookbook



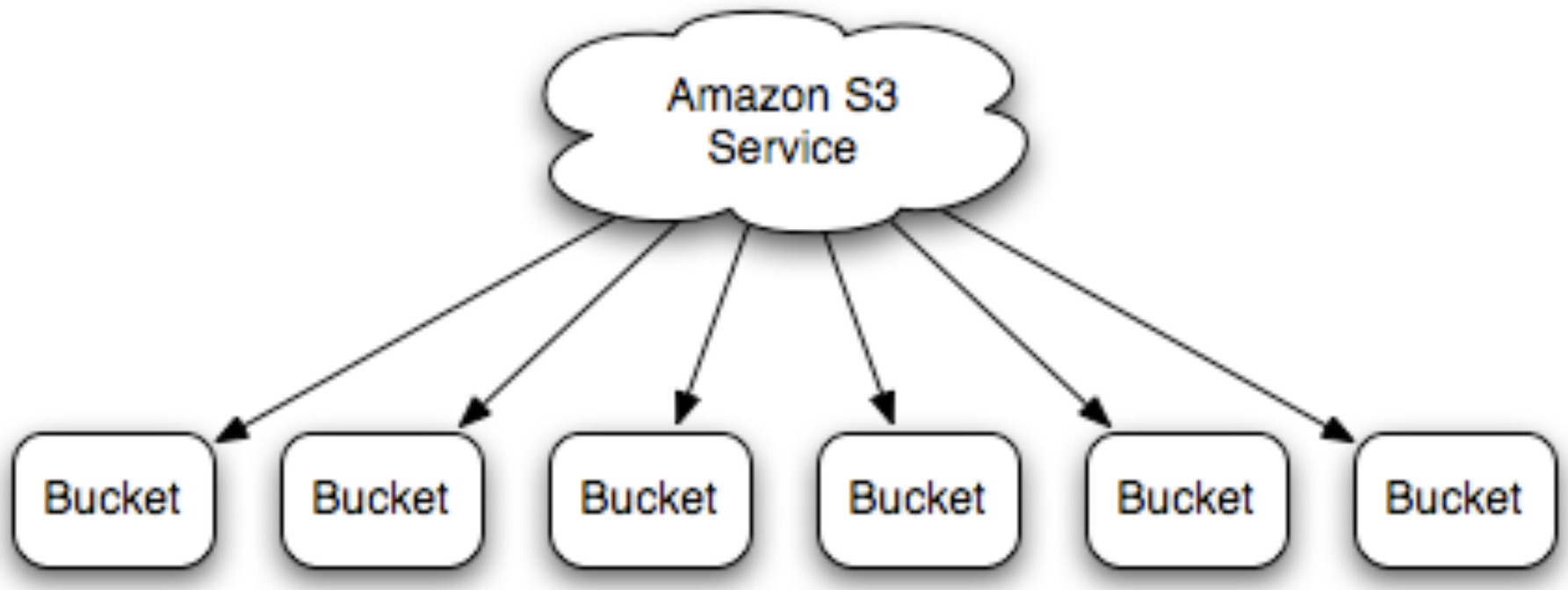
http://thes3cookbook.sopobo.com

http://github.com/spatten/thes3cookbook

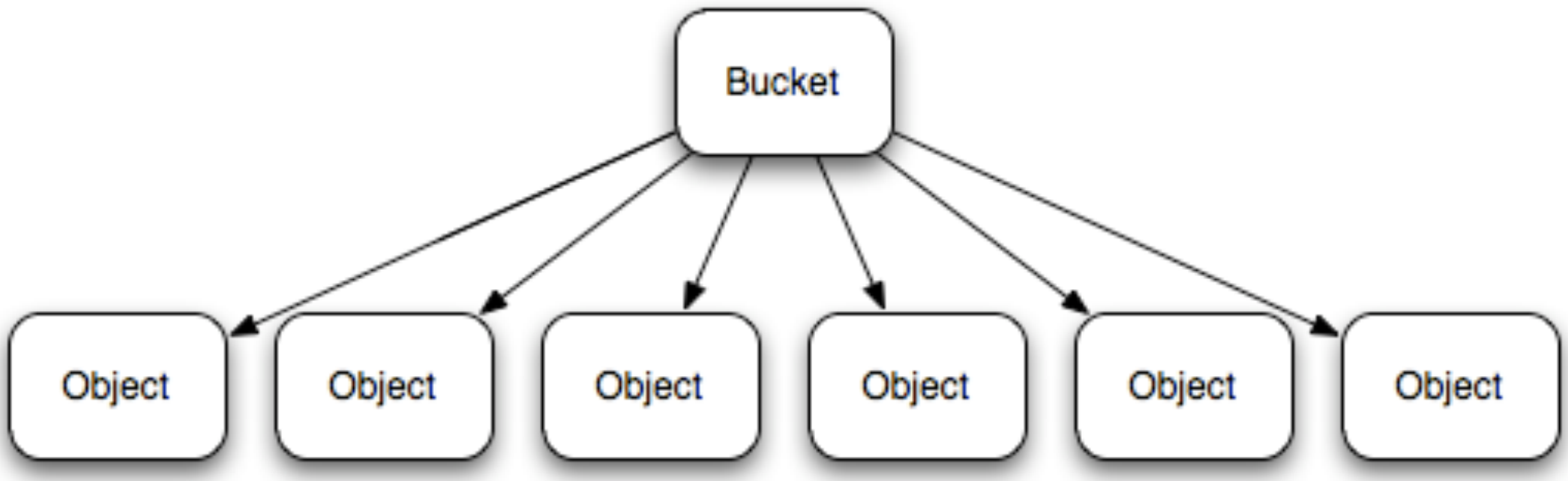
S3 for Rails

- **Serving Assets**
- **Authenticated URLs**
- **Backup**

S3's Architecture



S3's Architecture



REST and S3

	Bucket	Object	ACL
PUT	create	create content = POST body	create
GET	get bucket info	get object + metadata	get ACL info
DELETE	destroy bucket	destroy object	destroy ACL
HEAD	N/A	get object's metadata	N/A

REST and S3

Buckets:

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

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

REST and S3

Objects:

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

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

REST and S3

ACLs:

```
http://s3.amazonaws.com/<bucket_name>?acl
```

```
http://s3.amazonaws.com/socialbandwagon?acl
```

REST and S3

ACLs:

```
http://s3.amazonaws.com/<bucket_name>/<object_key>?acl
```

```
http://s3.amazonaws.com/socialbandwagon/logo.png?acl
```

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_object.rb
# Usage: create_object.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)
```

Getting Started with S3

```
#!/usr/bin/env ruby
# create_object.rb
# Usage: create_object.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)
```

Getting Started with S3

```
#!/usr/bin/env ruby
# create_object.rb
# Usage: create_object.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)
```

Getting Started with S3

```
#!/usr/bin/env ruby
# create_object.rb
# Usage: create_object.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)
```

Getting Started with S3

```
#!/usr/bin/env ruby
# create_object.rb
# Usage: create_object.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)
```

Getting Started with S3

```
#!/usr/bin/env ruby
# create_object.rb
# Usage: create_object.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

Why asset hosting?

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

You might want to automate that

Asset Hosting on S3

Asset host syncher:

http://spattendesign.com/projects

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?

- **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

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

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
```

The S3 Cookbook recipe

Get the code:

http://github.com/spatten/thes3cookbook

get everything in

/backing_up_a_mysql_database_to_s3_recipe

Backing up your code

Backing up your code

== backing up your repository

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

The S3 Cookbook recipe

Get the code:

http://github.com/spatten/thes3cookbook

get everything in

/backing_up_your_svn_repository_recipe

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/thes3cookbook

get everything in

/synchronizing_multiple_directories_recipe

sync_multiple_directories.rb

```
# Files should be uploaded if
# The file doesn't exist in the bucket
# OR
# The MD5 hashes don't match
def select_files_to_upload
  @files_to_upload = @local_files.select do |file|
    case
    when File.directory?(local_name(file))
      false # Don't upload directories
    when !@bucket[s3_name(file)]
      true # Upload if file does not exist on S3
    when @bucket[s3_name(file)].etag !=
      Digest::MD5.hexdigest(File.safe_read(local_name(file)))
      true # Upload if MD5 sums don't match
    else
      false # the MD5 matches and it exists already,
            # so don't upload it
    end
  end
end
```

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
```

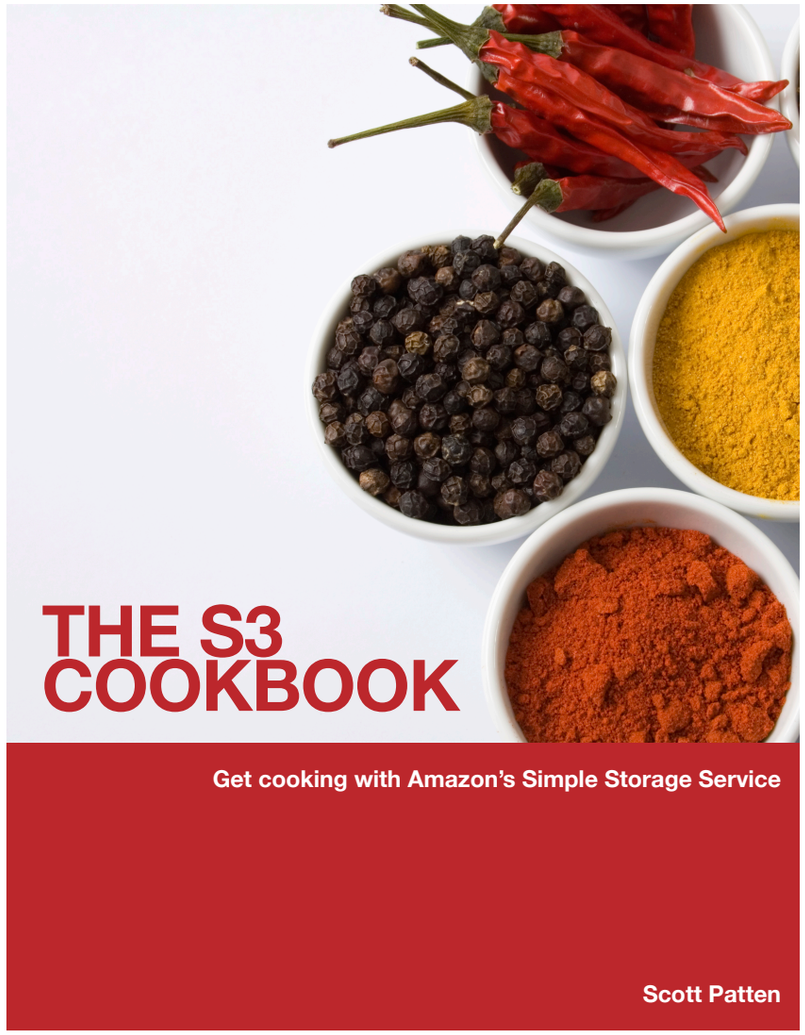
stick it in your cron

```
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!



The Book:

http://thes3cookbook.sopobo.com

The Slides:

http://github.com/spatten/s3-bof-railsconf2009

Scott Patten

scott@ruboss.com www.ruboss.com

twitter: @scott_patten

Ruboss