



# Backing up your app

---

Scott Patten - [scott@ruboss.com](mailto:scott@ruboss.com)



# 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



# I'm kind of biased....

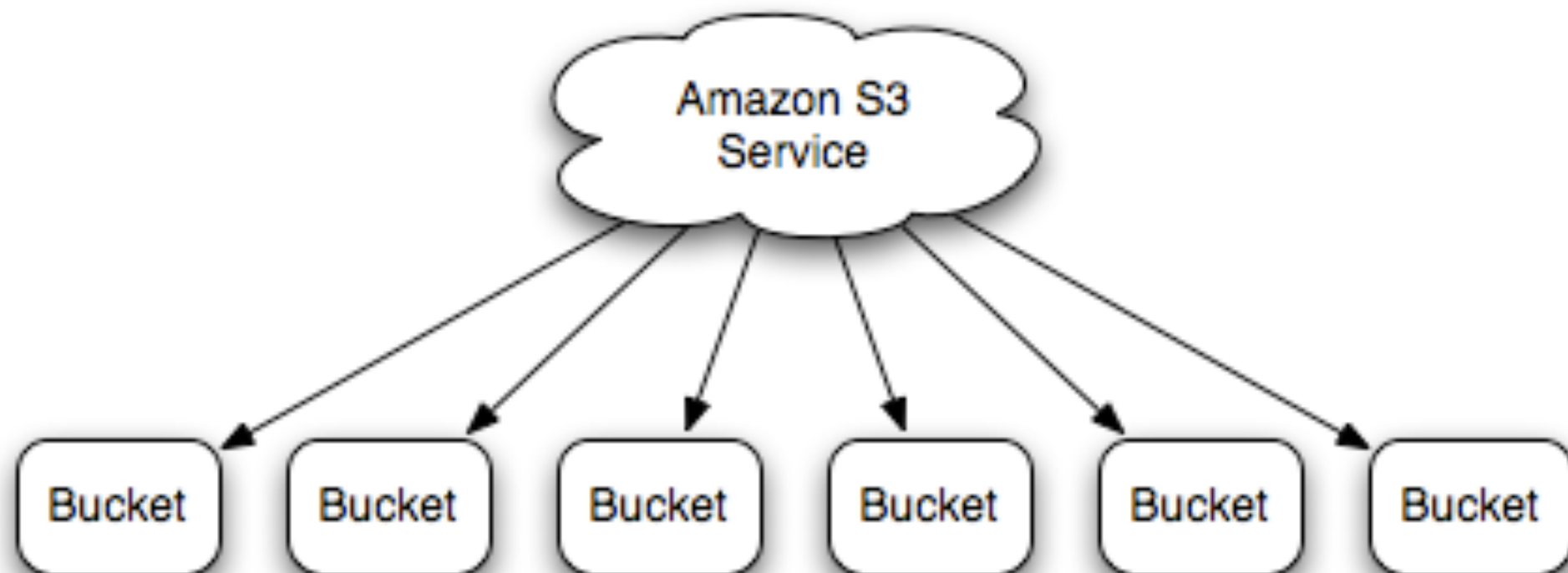
The S3 Cookbook.

<http://groups.google.com/group/thes3cookbook>



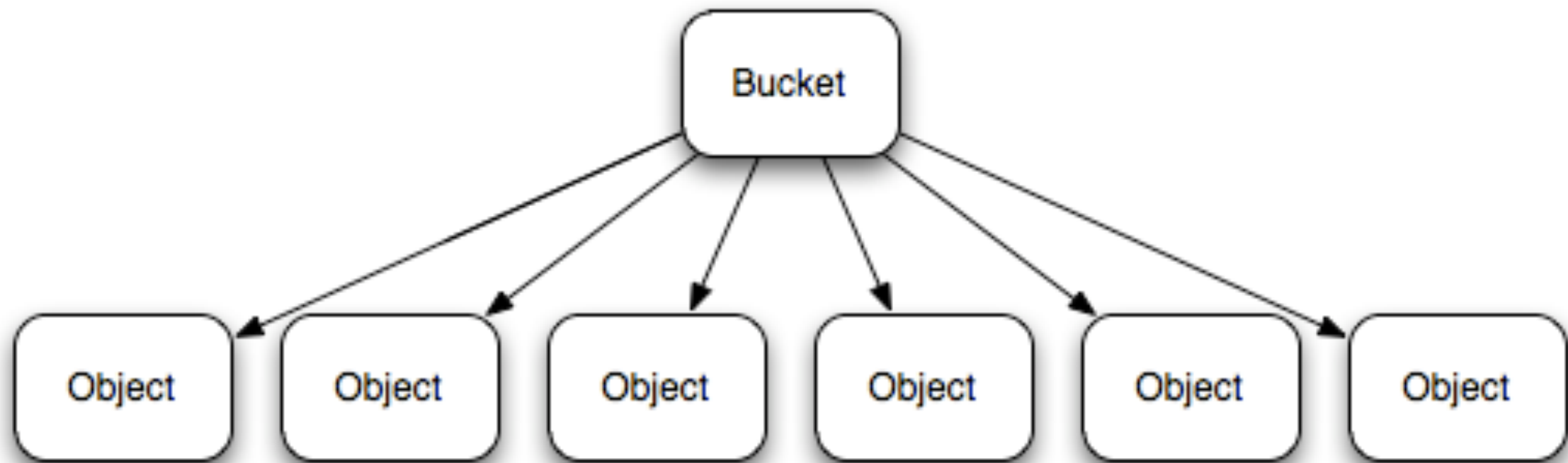


# S3's Architecture





# 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

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']
)

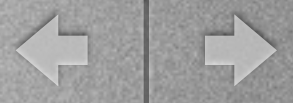
bucket = ARGV[0]
file = ARGV[1]

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





# Backing up your DB





# Don't be these guys


[Tech](#) [Gadgets](#) [Mobile](#) [Enterprise](#) [CrunchBase](#) [More ▼](#)



[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

I. 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**





# Full and incremental backups



# Full and incremental backups

- Once again, there are full and incremental backups





# Full and incremental backups

- Once again, there are full and incremental backups
- Full backups give you a complete snapshot of your repository. They can get huge



# Full and incremental backups

- Once again, there are full and incremental backups
- Full backups give you a complete snapshot of your repository. They can get huge
- Incremental backups give you a diff of a single commit





# Backup SVN to S3

Full dump:

```
svnadmin dump <repo path> \  
--revision 0:<last_revision> > filename
```



# Backup SVN to S3

## Incremental dump:

```
svnadmin dump <repo path> \  
--revision <revision> --incremental \  
> filename
```





# Backup SVN to S3

```
def create_full_dump
  puts "Creating full dump for revision #{@rev}"
  STDOUT.flush
  cmd = "/usr/local/bin/svnadmin dump '#{@repos}' --revision " +
        "'0:#{@rev}' > #{filename_with_path}"
  `#{cmd}`
  if @zip_full
    `gzip #{filename_with_path}`
  end
end
```



# Backup SVN to S3

```
def create_incremental_dump
  cmd = "/usr/local/bin/svnadmin dump '#{@repos}' --revision " +
        "'#{@rev}' --incremental > '#{filename_with_path}'"
  `#{cmd}`
  if @zip_incremental
    `gzip #{filename_with_path}`
  end
end
```





# Backup SVN to S3



# Backup SVN to S3

- Put it in a post-commit hook or a cron job





# Backup SVN to S3

- Put it in a post-commit hook or a cron job
- Don't just backup the latest commit.



# 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**



# Backing up your server configuration





# Backing up your server configuration

**Poll: Does anyone do this?**



# Creating an AMI on EC2





# Creating an AMI on EC2

**Create an image (AMI)**



# Creating an AMI on EC2

## Create an image (AMI)

```
$> ec2-bundle-vol -d /mnt -k <key file> -c \  
<cert file> -u <aws user id> -r i386
```





# Creating an AMI on EC2



# Creating an AMI on EC2

**upload the image to S3**





# Creating an AMI on EC2

**upload the image to S3**

```
$> ec2-upload-bundle -b <your-s3-bucket> \
```



# Creating an AMI on EC2

**upload the image to S3**

```
$> ec2-upload-bundle -b <your-s3-bucket> \  
-m /mnt/image.manifest.xml \  
-a <aws-access-key-id> \
```





# Creating an AMI on EC2

**upload the image to S3**

```
$> ec2-upload-bundle -b <your-s3-bucket> \  
-m /mnt/image.manifest.xml \  
-a <aws-access-key-id> \  
-s <aws-secret-access-key>
```



# Creating an AMI on EC2





# Creating an AMI on EC2

**Register the AMI**



# Creating an AMI on EC2

## Register the AMI

```
ec2-register <your-s3-bucket>/image.manifest.xml
```





# Creating an AMI on EC2



# Creating an AMI on EC2

**Create a new instance using the AMI:**





# Creating an AMI on EC2

**Create a new instance using the AMI:**

```
$> ec2-run-instances ami-c9bc58a0 -k <ssh_key>
```



# Creating an AMI on EC2

[http://docs.amazonwebservices.com/AWSEC2/2007-08-29/  
GettingStartedGuide/creating-an-image.html](http://docs.amazonwebservices.com/AWSEC2/2007-08-29/GettingStartedGuide/creating-an-image.html)





# VMWare and Xen

- The AMI packaging tool is a wrapper around the Xen packaging tool.
- You can do the same thing with VMware
- Any other solutions out there?



# Thanks!

<http://groups.google.com/group/thes3cookbook>

<http://spattendesign.com>

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