peritor wissensmanagement

Rails on AWS

Jonathan Weiss, Peritor Wissensmanagement GmbH

RubyFools Copenhagen, 2008

Starting Point

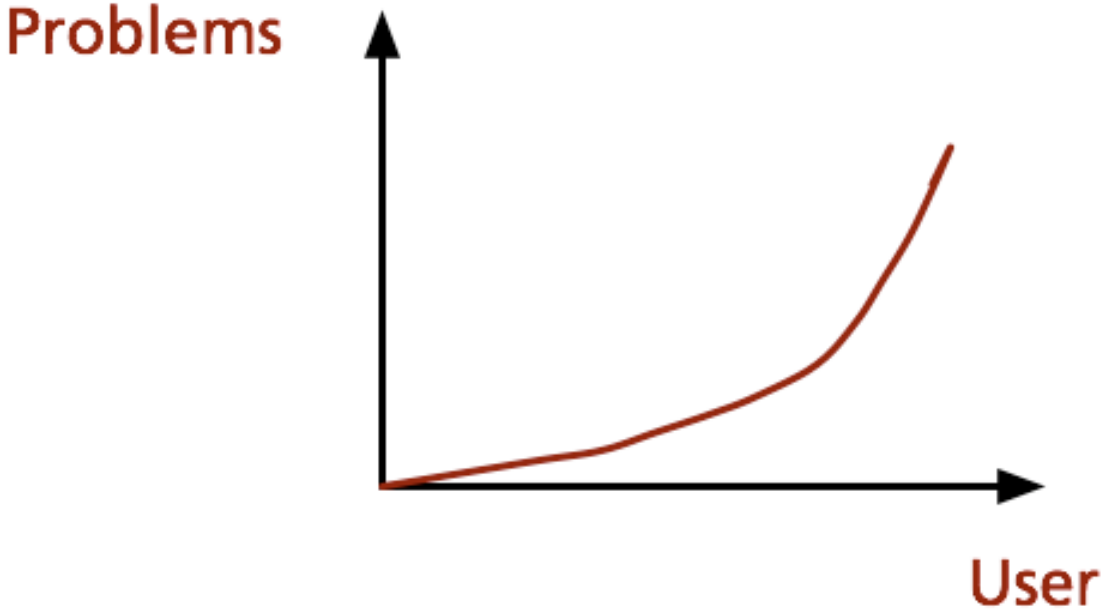
peritor wissensmanagement

One machine:

- Apache
- Ruby / Rails
- MySQL

Worst Case Populariy

peritor wissensmanagement



A Difficult Path

peritor wissensmanagement

Problem: Backup

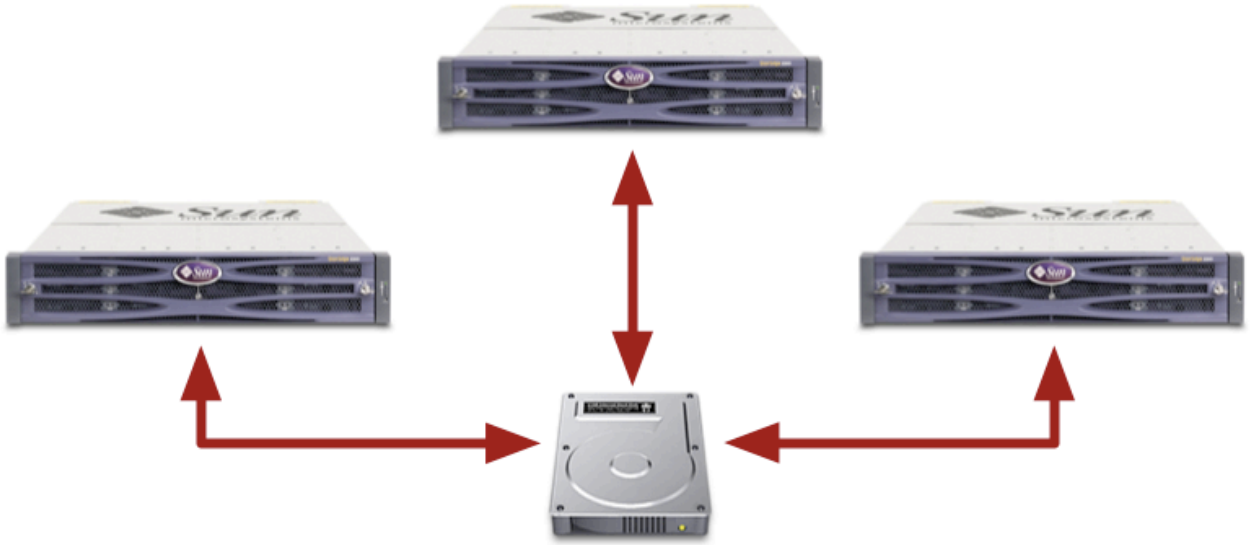
peritor wissensmanagement

- High availability
- Redundancy
- Very big data sets

Problem: File System

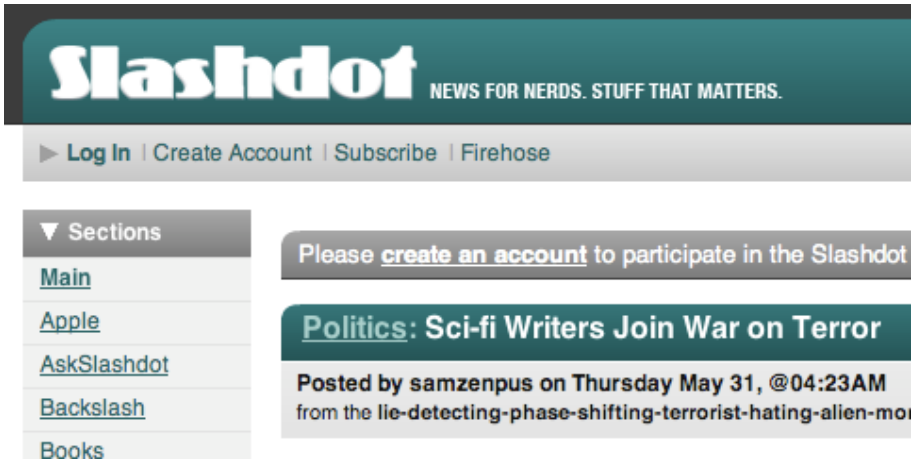
peritor wissensmanagement

- Important files have to be accessed by many servers
- NFS / Samba not practical



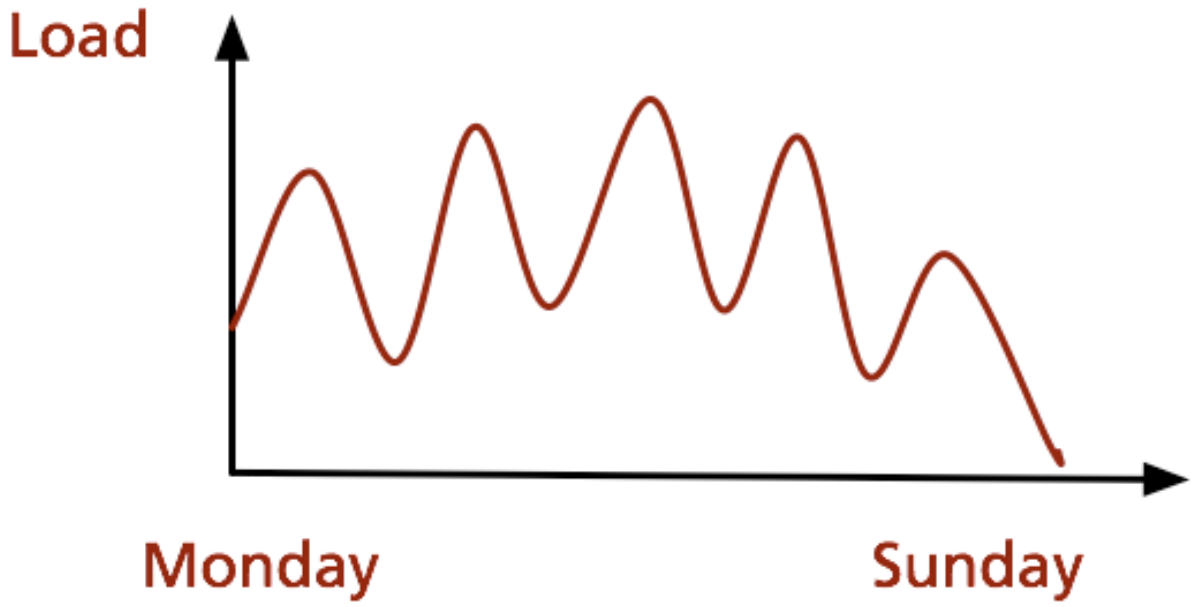
Problem: Spontaneous Traffic

peritor wissensmanagement



Problem: Load Fluctuation

peritor wissensmanagement



peritor wissensmanagement

Don‘t reinvent the wheel!

Amazon Web Services

peritor wissensmanagement

Simple Storage Service - S3	Elastic Compute Cloud - EC2	E-Commerce S..
Redundant Data Store	Virtual server per hour	.....
$0,15 per GB data per month	$0,10 per CPU hour	
$0,10 - $0,20 per GB traffic	$0,10 - $0,20 per GB traffic	

S3 - Simple Storage Service

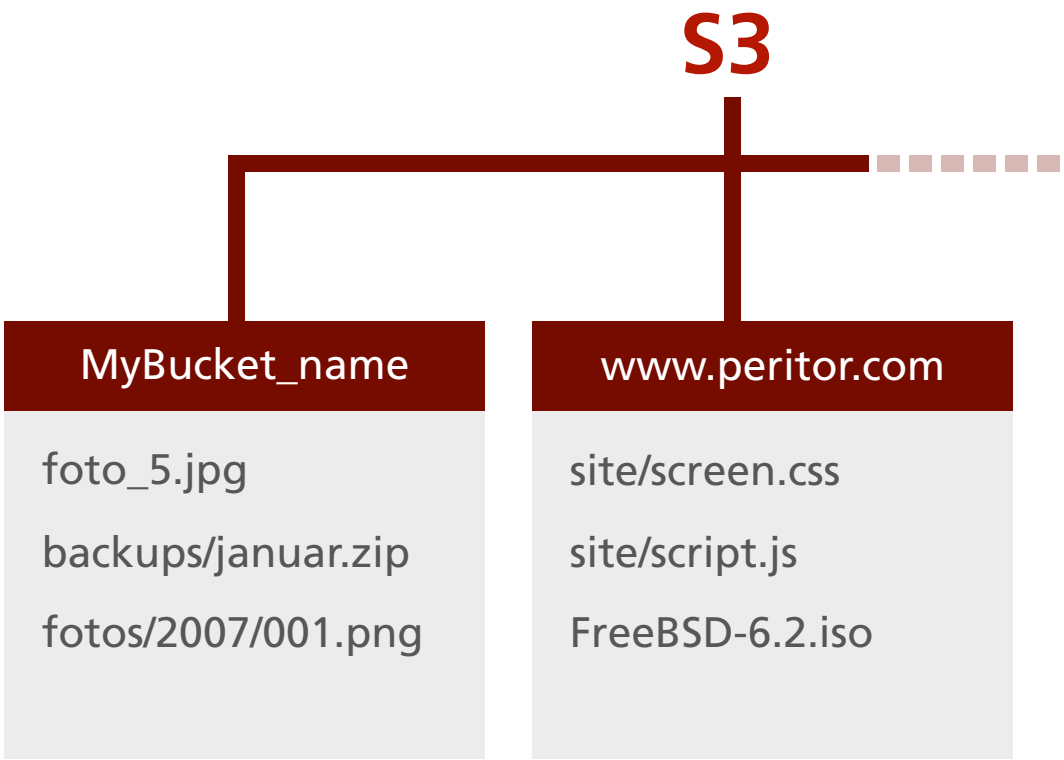
peritor wissensmanagement

- Redundant storage - as much as you like
- max. 5 GB per object
- Organized in „Buckets“
- Web Service API for uploads
- Downloads via
  - Web Service
  - HTTP / HTTPS
  - BitTorrent

S3 - Buckets

peritor wissensmanagement

- Unique over all S3
- Contains many key-value-metadata tupel
- Cannot contain other buckets!
- Key can contain „/“



S3 with AWS::S3

peritor wissensmanagement

Upload

```
S3Object.store('vid.mpeg', open('/tmp/vid.mpeg'), 'my_bucket')
```

Download

```
picture = S3Object.find 'headshot.jpg', 'photos'

# raw data
picture.value

# http(s) download
picture.url # http://s3.amazonaws.com/photos/headshot.jpg

# http://s3.amazonaws.com/photos/headshot.jpg?torrent
picture.torrent
```

EC2 - Elastic Compute Cloud

peritor wissensmanagement

- Based on XEN virtualization
- On demand virtual servers - controlled with Web Service API

	Small	Large	Extra Large
CPU / EC2 Compute Units	1 x 1	2 x 2	4 x 2
RAM	1.75 GB	7.5 GB	15 GB
HDD	160 GB	850 GB	1690 GB

- Use your favorite Linux distro (Linux 2.6.16), Amazon Machine Images (AMI) are stored on S3
- ACLs for hosts/ports access control

EC2 Tools

peritor wissensmanagement

List availabe images

```
> ec2-describe-images
```

Start a new instance

```
> ec2-run-instances ami-5bae4b32 -k $RSA_KEY
```

Login with SSH

```
> ssh root@domU-12-34-31-00-00-05.usma1.compute.amazonaws.com
```

Shutdown instance

```
> ec2-terminate-instances i-10a64379
```

amazon-ec2 gem

peritor wissensmanagement

Setup

```
require 'EC2'

ACCESS_KEY_ID = '--YOUR AWS ACCESS KEY ID--'
SECRET_ACCESS_KEY = '--YOUR AWS SECRET ACCESS KEY--'

ec2 = EC2::Base.new(
  :access_key_id => ACCESS_KEY_ID,
  :secret_access_key => SECRET_ACCESS_KEY )
```

Usage

```
ec2.run_instances(:image_id => 'ami-0cf61365')

ec2.describe_instances

ec2.terminate_instances(:instance_id => 'i-5229c123')
```

peritor wissensmanagement

And now?

How does this solve my problems?

S3 - Backup

peritor wissensmanagement

- s3sync.rb
- Brackup
- Jungle Disk
- S3 FUSE
- s3DAV
- Duplicity
- S3Browser
- Firefox S3 Organizer extension
- …

s3sync.rb

peritor wissensmanagement

Backup

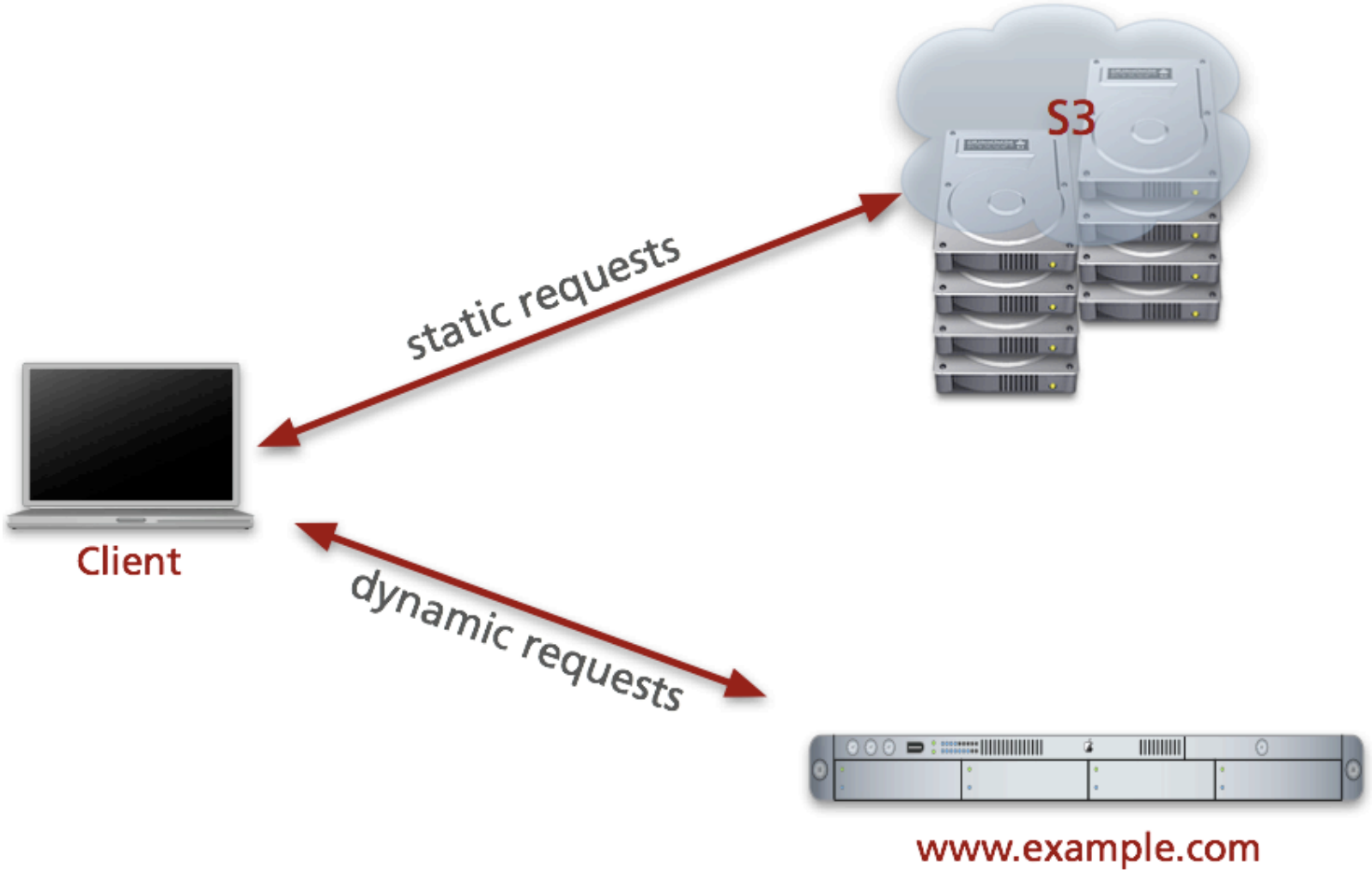
```
export AWS_ACCESS_KEY_ID='AAAAAAAAAAAAAAAAAA'
export AWS_SECRET_ACCESS_KEY='BBBBBBBBBBBBBBBBBBBBBBBBBBB'
export SSL_CERT_FILE='/home/amazon/s3sync/ssl.crt'

./s3sync.rb -v -r --ssl /usr/local/backup  MyBucket:/www.example.com/backup
```

Restore

```
./s3sync.rb -v -r --ssl MyBucket:/www.example.com/backup  /usr/local/backup
```

S3 Asset Host

peritor wissensmanagement



S3 Asset Host

peritor wissensmanagement

Setup DNS

```
> dig static.example.com

static.example.com.     86400   IN      CNAME   s3.amazonaws.com.
s3.amazonaws.com.       86400   IN      A       72.21.203.129
```

Rails configuration

```
config.action_controller.asset_host = "http://static.example.com"
```

S3 Asset Host

peritor wissensmanagement

welcome.rhtml template

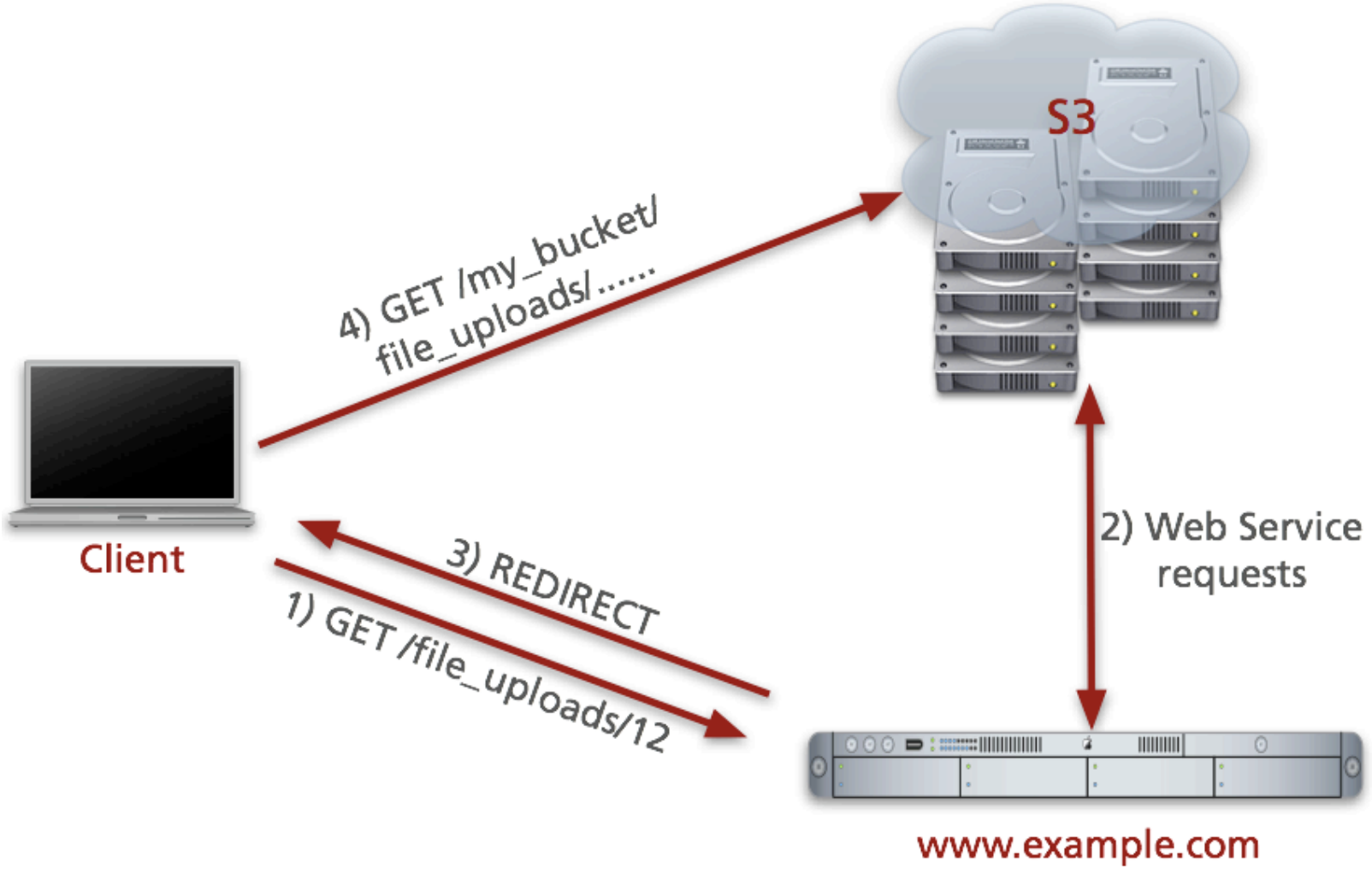
```
<%= image_tag('logo.gif') %> <br />

<strong> Welcome! </strong>
```

Output

```
<img src="http://static.example.com/images/logo.gif"> <br />

<strong> Welcome! </strong>
```

S3 - Authenticated User Data

peritor wissensmanagement



attachment_fu Rails plugin

peritor wissensmanagement

Setup

```
class FileUpload < ActiveRecord::Base
  has_attachment :storage => :s3, :s3_access => :private, :max_size => 15.megabytes

  def public_s3_url(ttl_seconds=10)
    AWS::S3::S3Object.url_for(self.full_filename,
                              self.bucket_name, {
                                :use_ssl => true,
                                :expires_in => ttl_seconds
                              })
  end

end
```

attachment_fu Rails plugin

peritor wissensmanagement

Upload

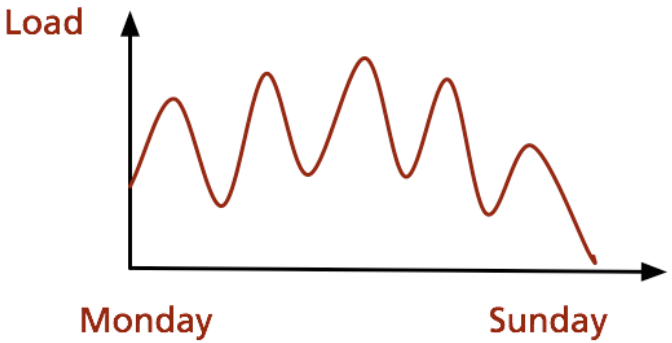
```
file_upload = FileUpload.new(params[:file_upload])
file_upload.save
```

Download

```
file_upload = FileUpload.find(params[:id])

file_upload.public_s3_url
# https://s3.amazonaws.com/my_bucket/file_uploads/12/README.txt?AWSAccessKeyId=XXXXXXX

file_upload.value
# raw data
```

On-Demand Computing with EC2

peritor wissensmanagement

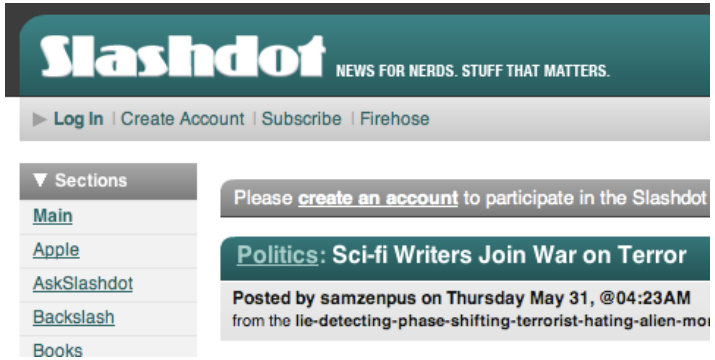


Time based, e.g. with cron

```
> crontab -l -u ec2

0 8 * * *     /home/ec/start_additional_instances.sh

0 18 * * *     /home/ec/stop_additional_instances.sh
```

On-Demand Computing with EC2

peritor wissensmanagement

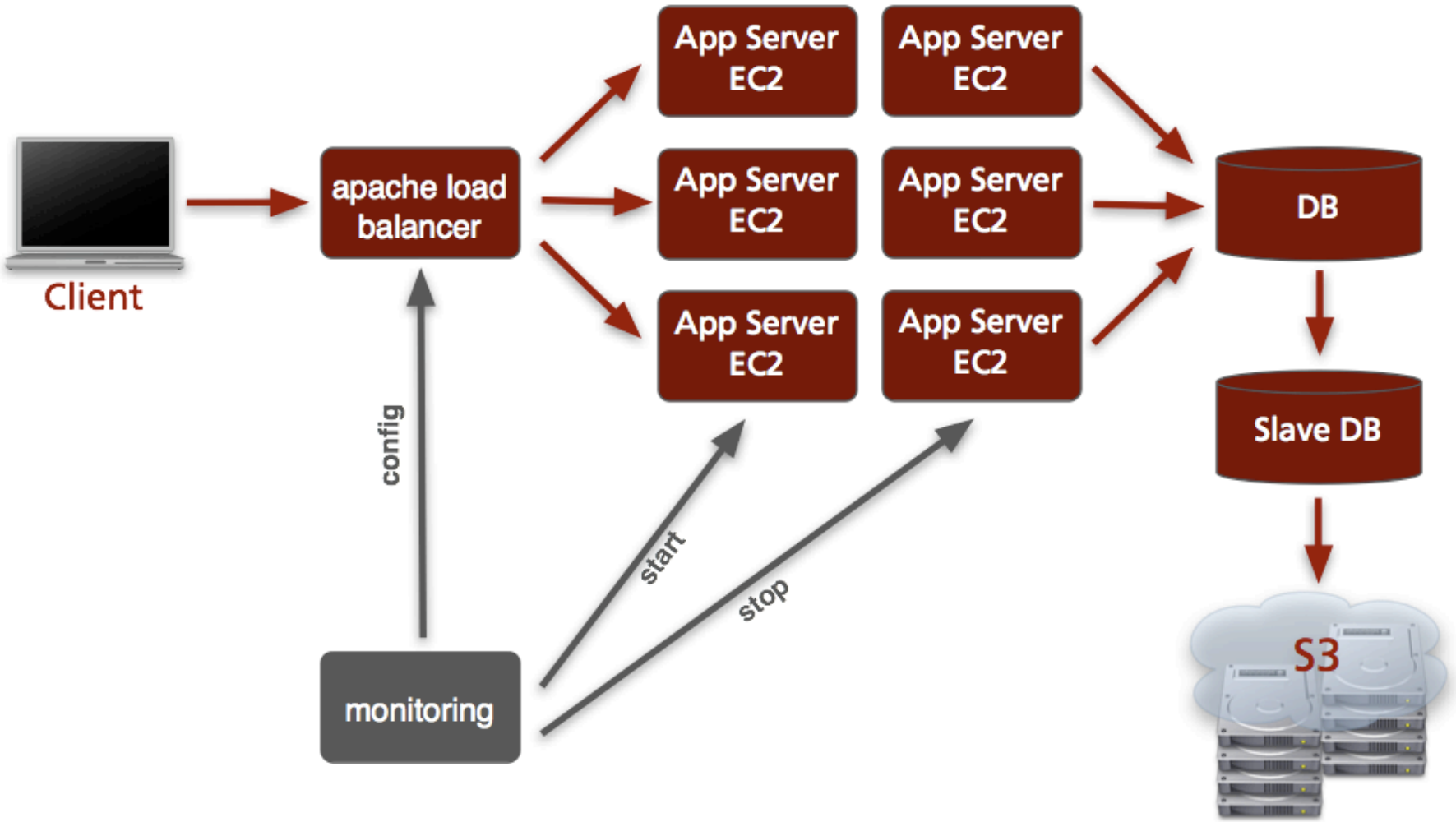


Load based, e.g. with Monit

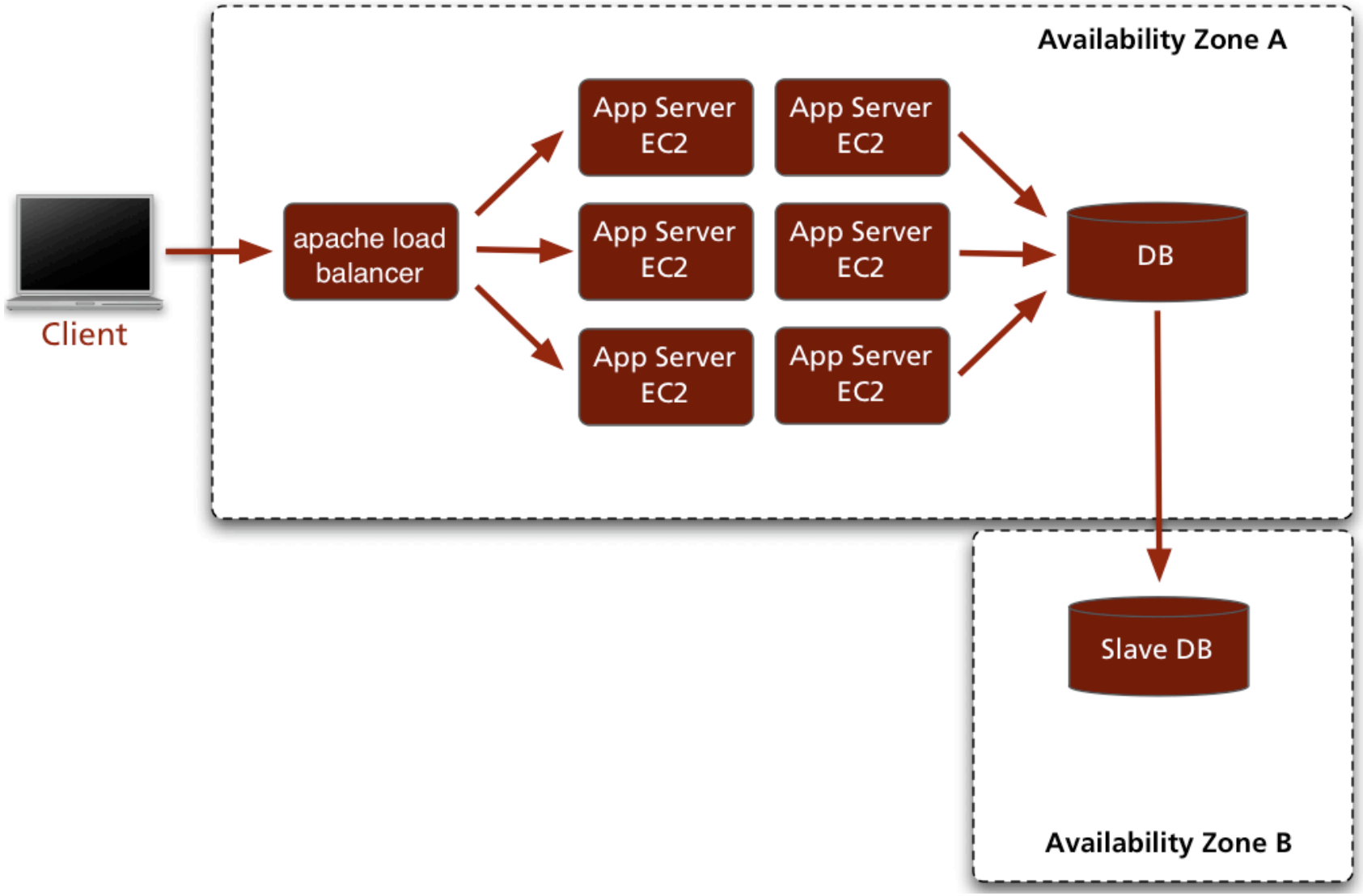
```
> tail /etc/monitrc

check system example.com
    if cpu usage (user) > 70% for 5 cycles then
      exec "/home/ec2/add_additional_instance.sh"
    if cpu usage (user) < 50% for 10 cycles then
      exec "/home/ec2/remove_additional_instance.sh"
```

On-Demand Computing with EC2

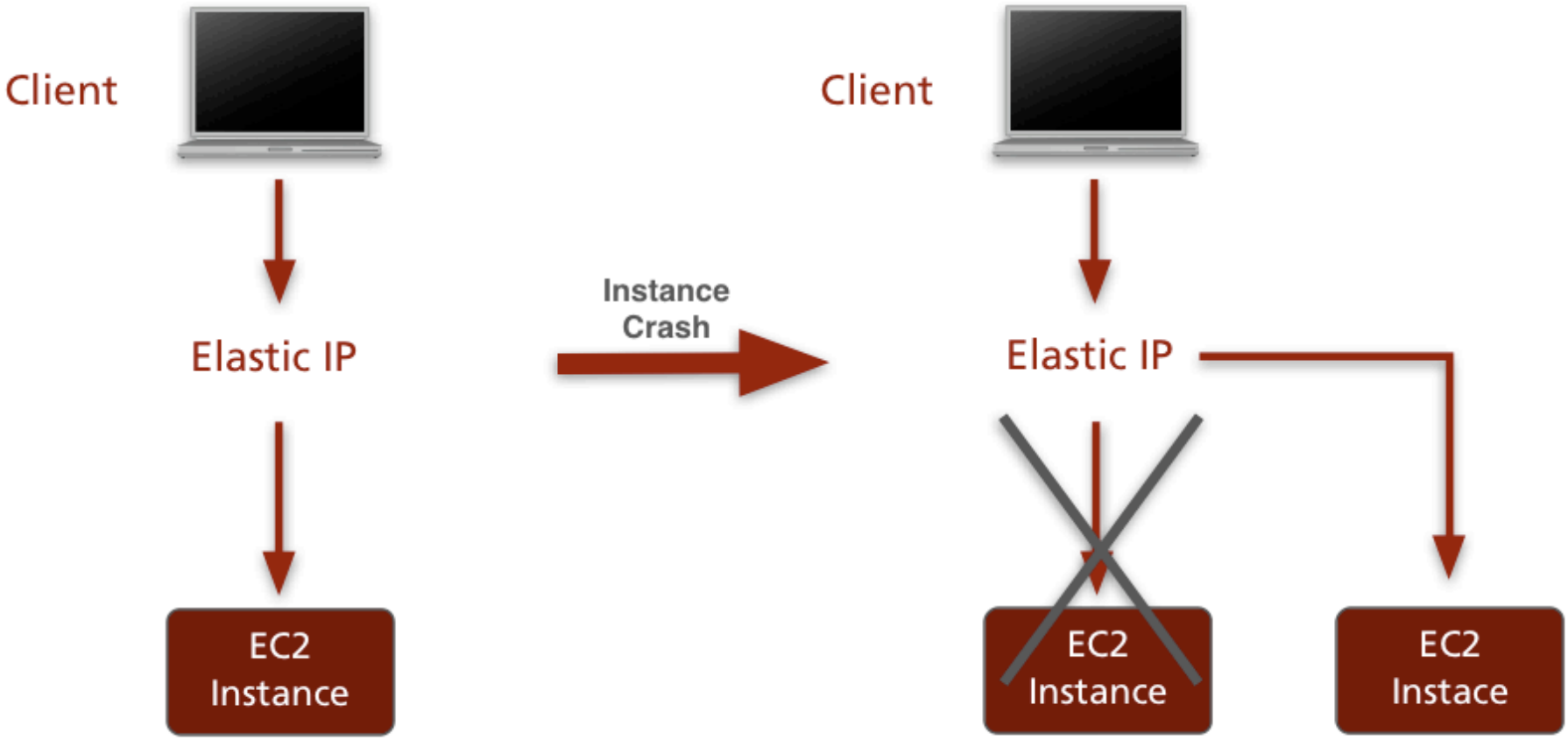
peritor wissensmanagement



Availability Zones

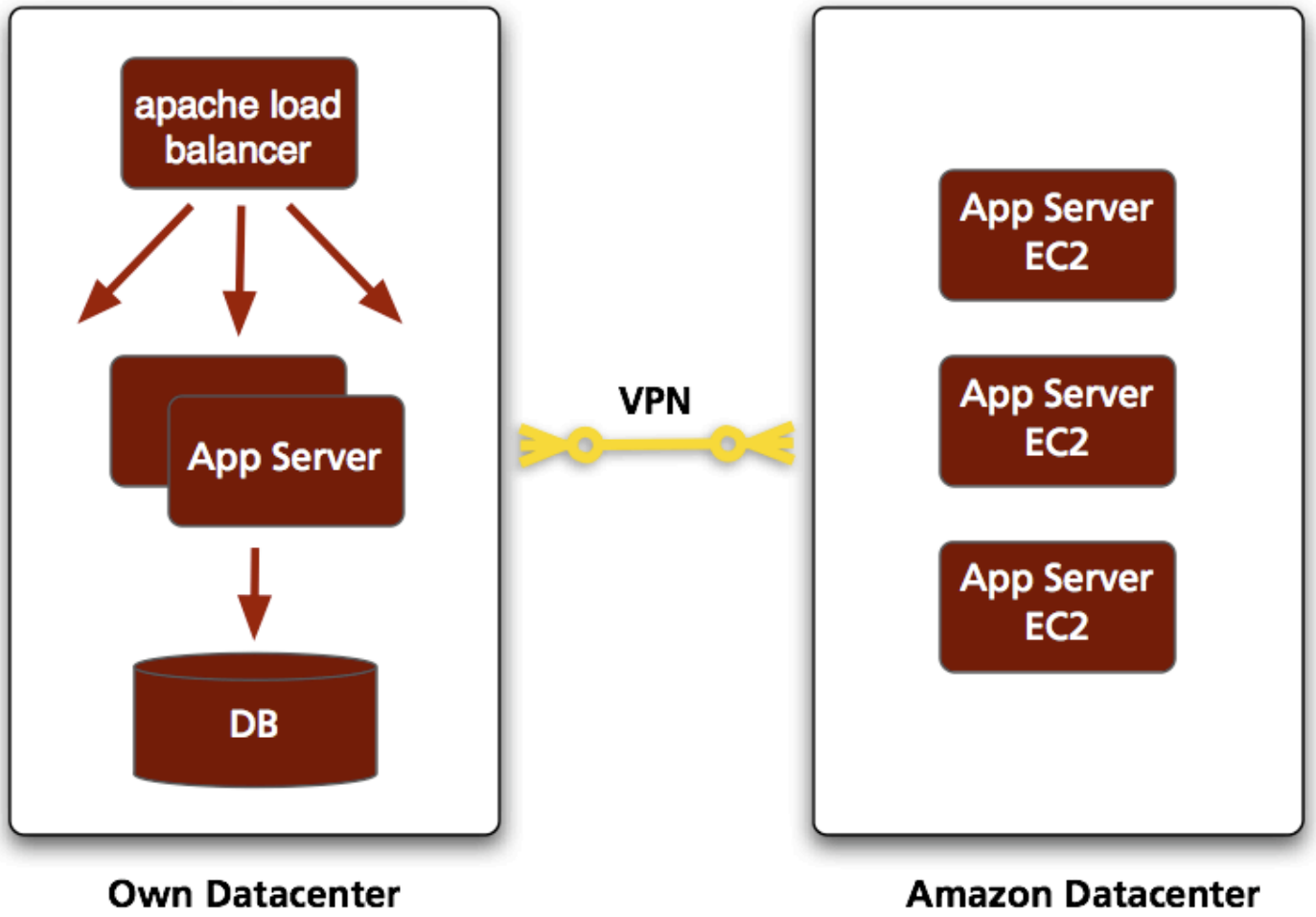


Elastic IPs


Client
Elastic IP
EC2
Instance
Instance
Crash
Client
Elastic IP
EC2
Instance
EC2
Instace

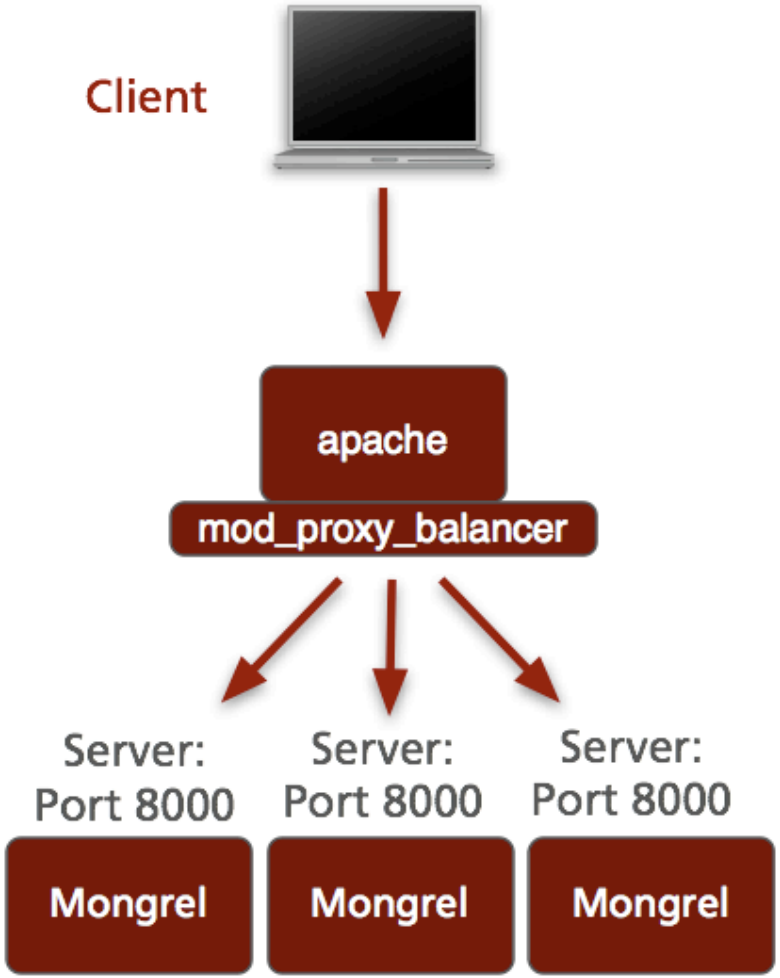
EC2 for extra capacity

peritor wissensmanagement



Load Balancer / Proxy

peritor wissensmanagement

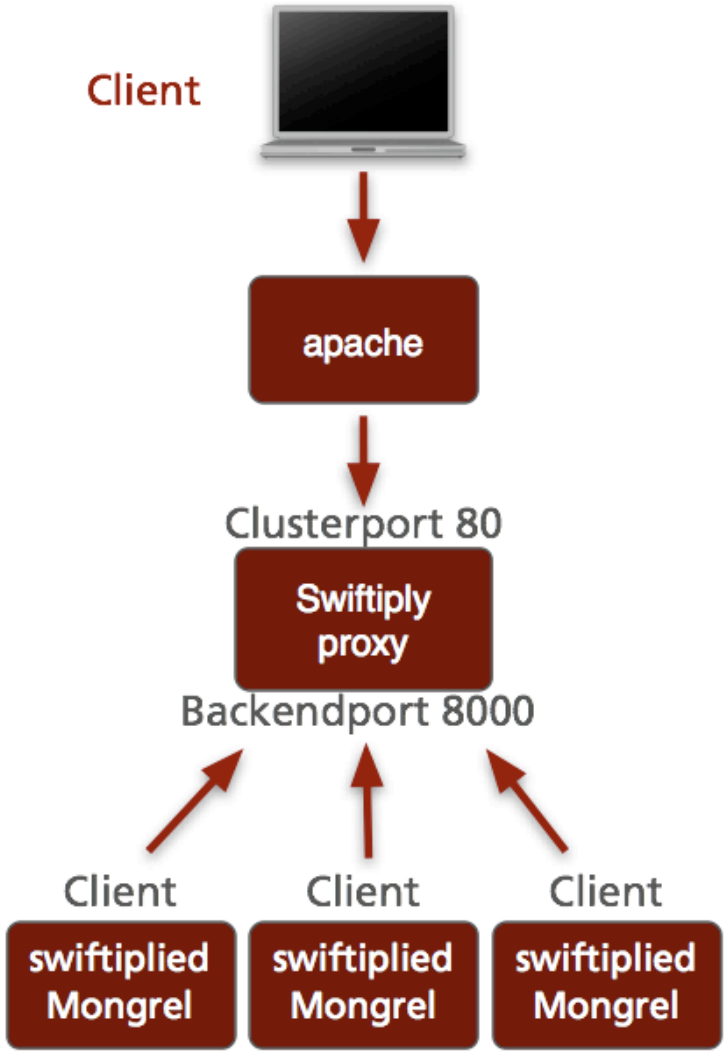


Example mod_proxy_blancer:

- Talks to multiple backend servers (Mongrel)
- Central Proxy/Load-Balancer configuration that has knowledge about nodes
- Typically proxy restart on config change

Swiftiply

peritor wissensmanagement



Swiftiply Proxy:

- Multiple backend clients have a persistent connection to the backend port
- Proxy listens on cluster port for requests and forwards them

No re-configuration

Swiftiply Proxy

peritor wissensmanagement

Installation

```
> gem install swiftiply -y
```

Start

```
> swiftiply -c swiftiply.yml
```

Configuration

```
> cat swiftiply.yml

cluster_address: 0.0.0.0
cluster_port: 80
daemonize: true
map:
  - incoming:
    - example.com
    - www.example.com
    outgoing: 127.0.0.1:30000
    default: true
  - incoming: www.meinprof.de
    outgoing: 127.0.0.1:30010
```

Swiftiplied Mongrel

peritor wissensmanagement

- Gem plugin that patches Mongrel
- Transforms Mongrel in Swiftiply client
- Experimental

Start

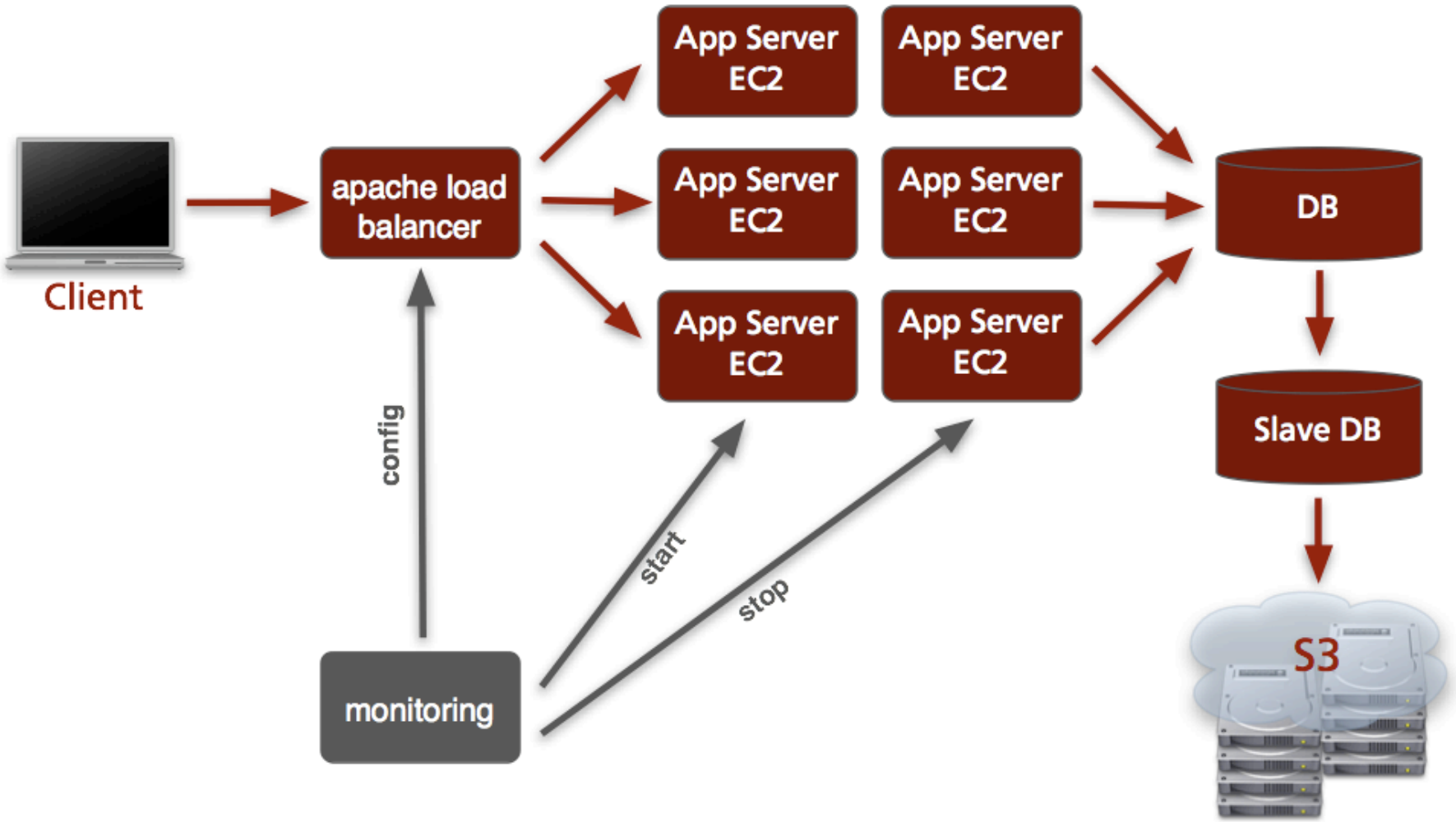
```
> env SWIFT=1 mongrel_rails start -p 8000
```

or

```
> swiftiply_mongrel_rails -C config/mongrel.yml
```

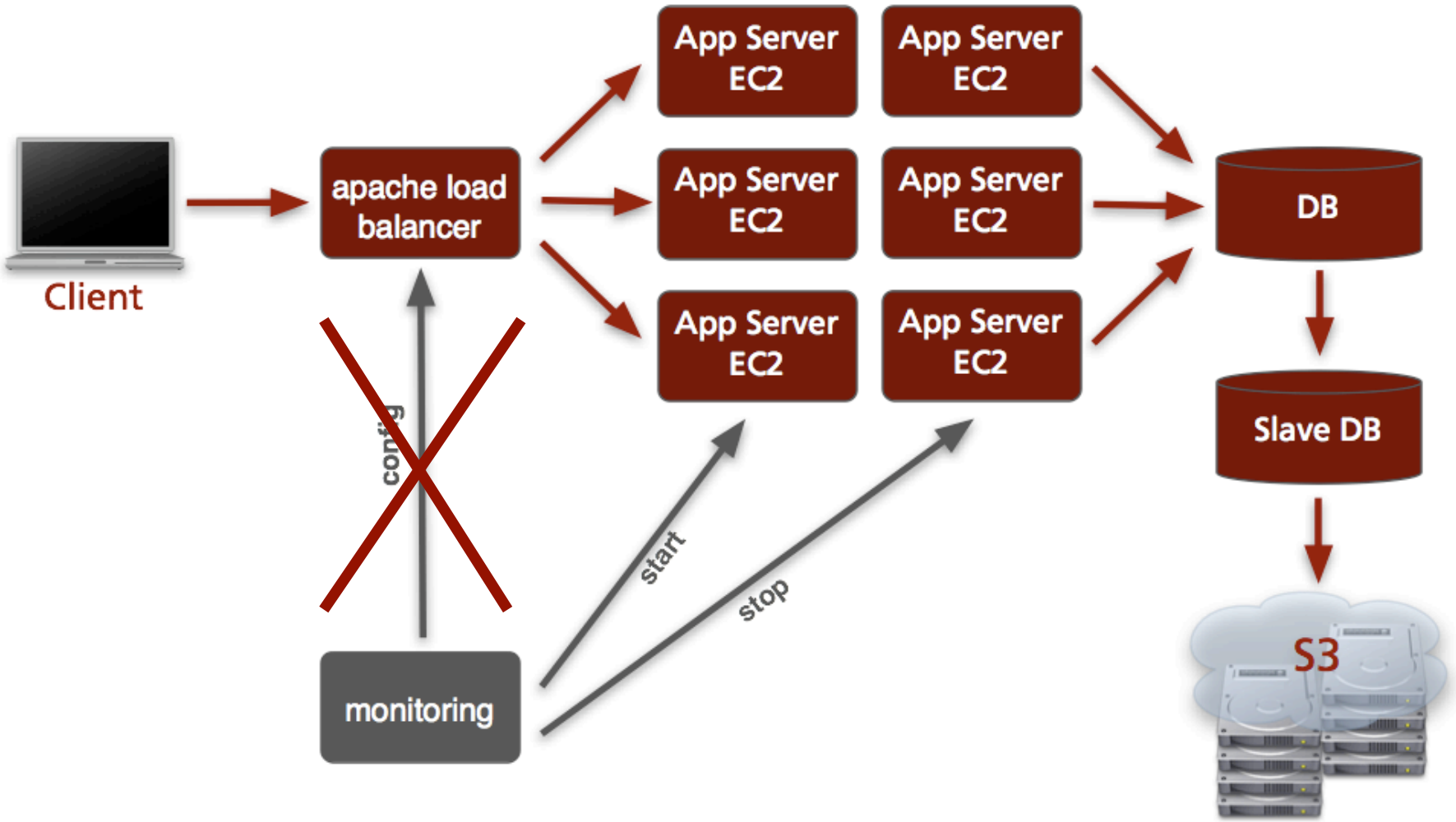
EC2 on Demand before Swiftiply

peritor wissensmanagement



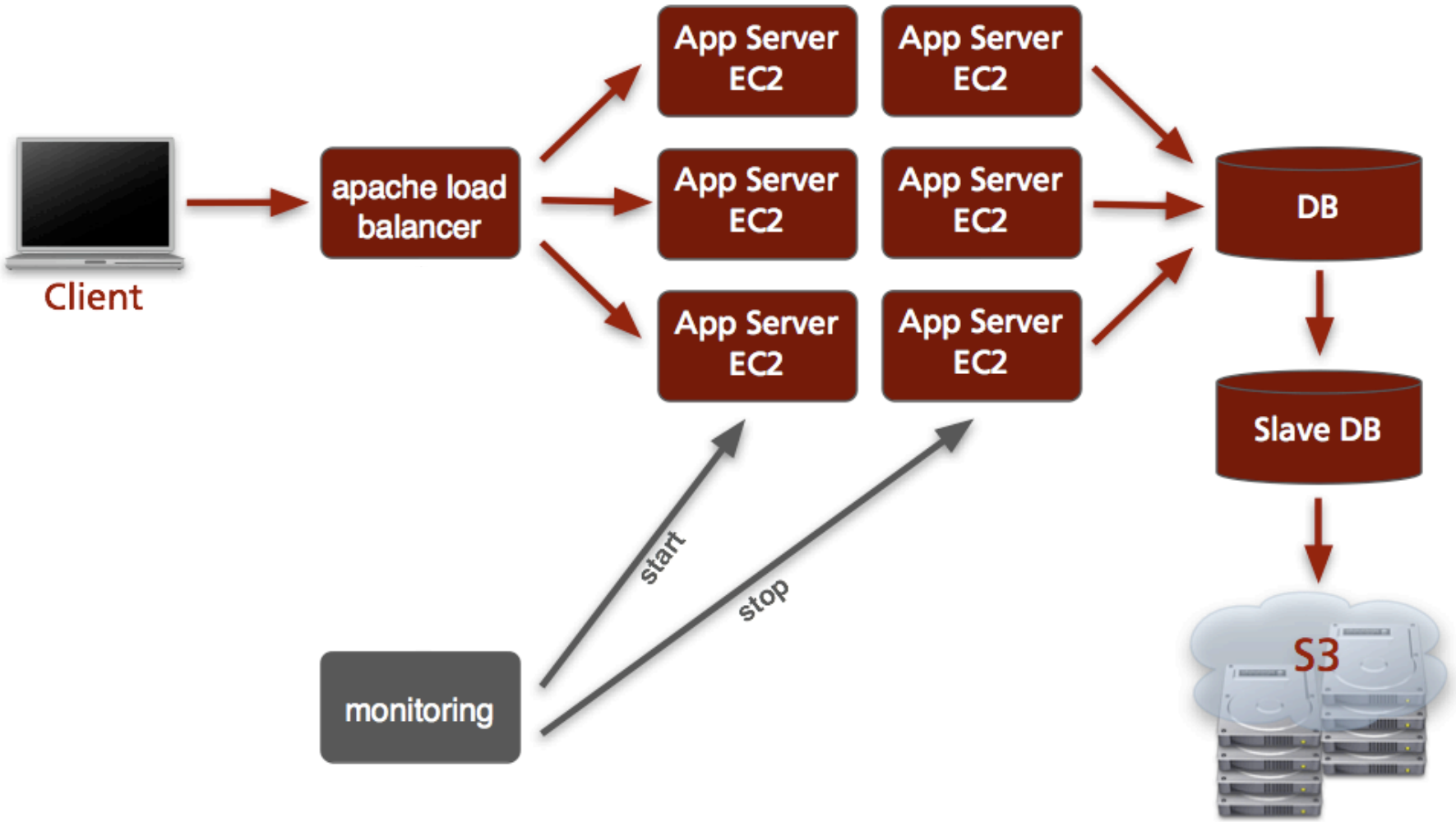
EC2 on Demand with Swiftiply

peritor wissensmanagement



EC2 on Demand with Swiftiply

peritor wissensmanagement



Ressources

peritor wissensmanagement

- Amazon Web Services
  http://aws.amazon.com
- Swiftiply
  http://swiftiply.swiftcore.org
- Attachment_fu
  http://svn.techno-weenie.net/projects/plugins/attachment_fu/
- Elastic Rails
  http://www.elasticrails.com
- Capazon
  http://capazon.rubyforge.org
- Rubber
  http://rubber.rubyforge.org
- RightScale
  http://www.rightscale.com

peritor wissensmanagement

Peritor Wissensmanagement GmbH

Lenbachstraße 2
12157 Berlin

Internet: www.peritor.com
Email: info@peritor.com

Telefon: +49 (0)30 69 40 11 94
Telefax: +49 (0)30 69 40 11 95