peritor consulting

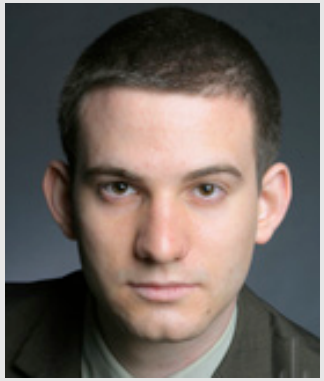
Deploying and Monitoring Ruby on Rails

A practical guide

Mathias Meyer and Jonathan Weiss, 02.09.2008
Peritor GmbH

Who are we?

peritor consulting



Jonathan Weiss

- Consultant for Peritor GmbH in Berlin
- Specialized in Rails, Scaling, Deployment, and Code Review
- Webistrano - Rails deployment tool
- FreeBSD Rubygems and Ruby on Rails maintainer

http://www.peritor.com

http://blog.innerewut.de

Who are we?

peritor consulting

Mathias Meyer

- Independent Contractor
- Specialized in Rails, Performance/Database Tuning, Deployment, and Refactoring
- Macistrano –
  Webistrano-Client for Mac OS X (and soon the iPhone)

http://www.paperplanes.de

Agenda

peritor consulting

Infrastructure → Deployment → Practical Session → Monitoring → Q & A

Infrastructure

Simple Rails Setup

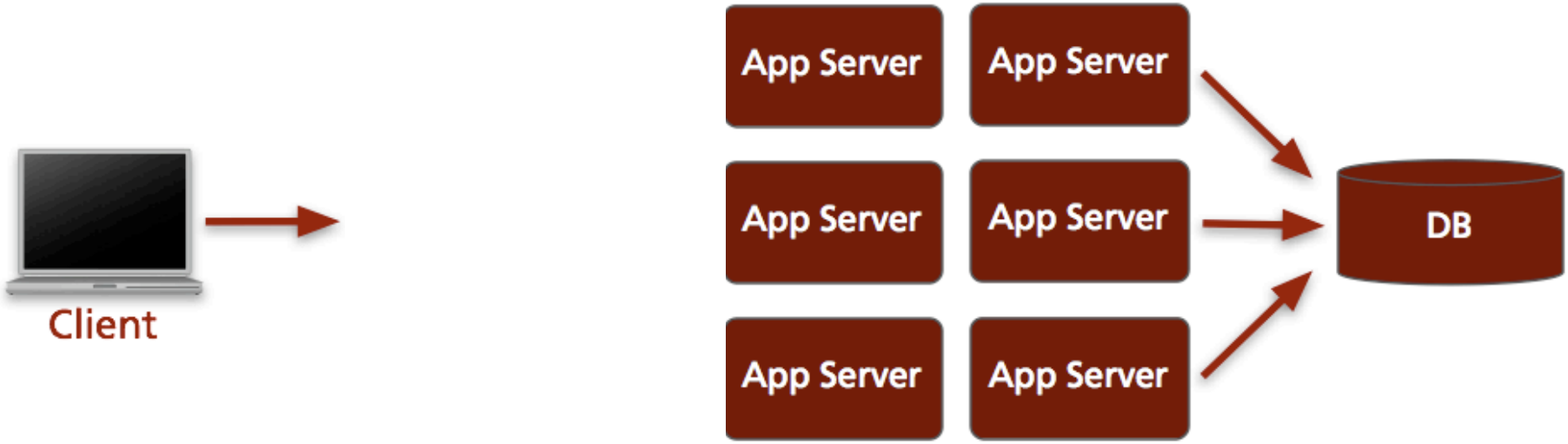
peritor consulting

One Rails instance handles all requests

Rails is single-threaded: There is only one concurrent request

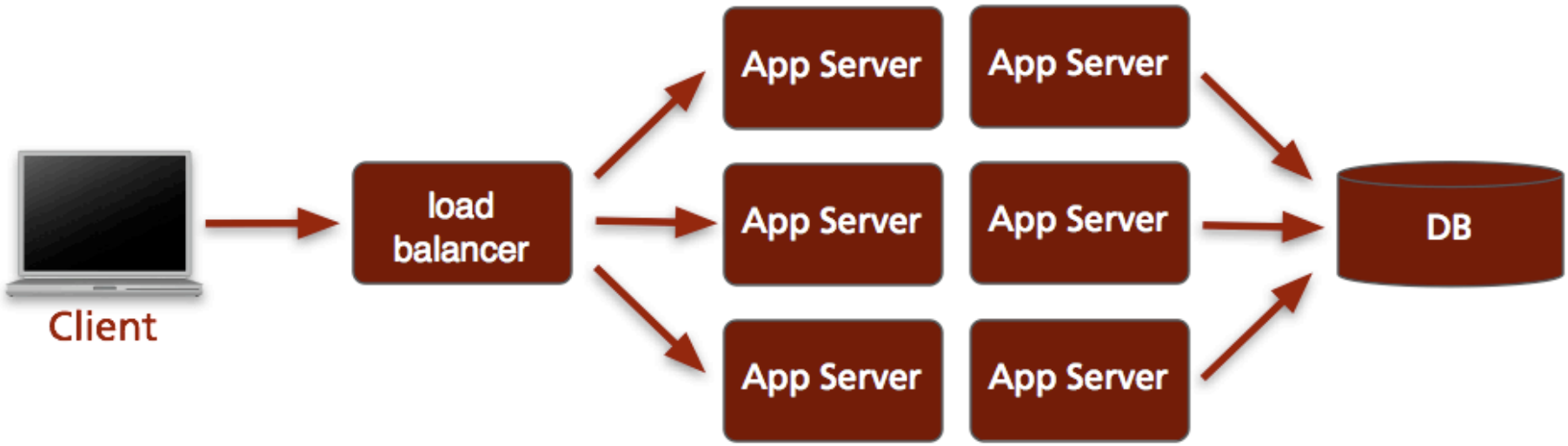
Rails Setup

peritor consulting



Rails Setup

peritor consulting



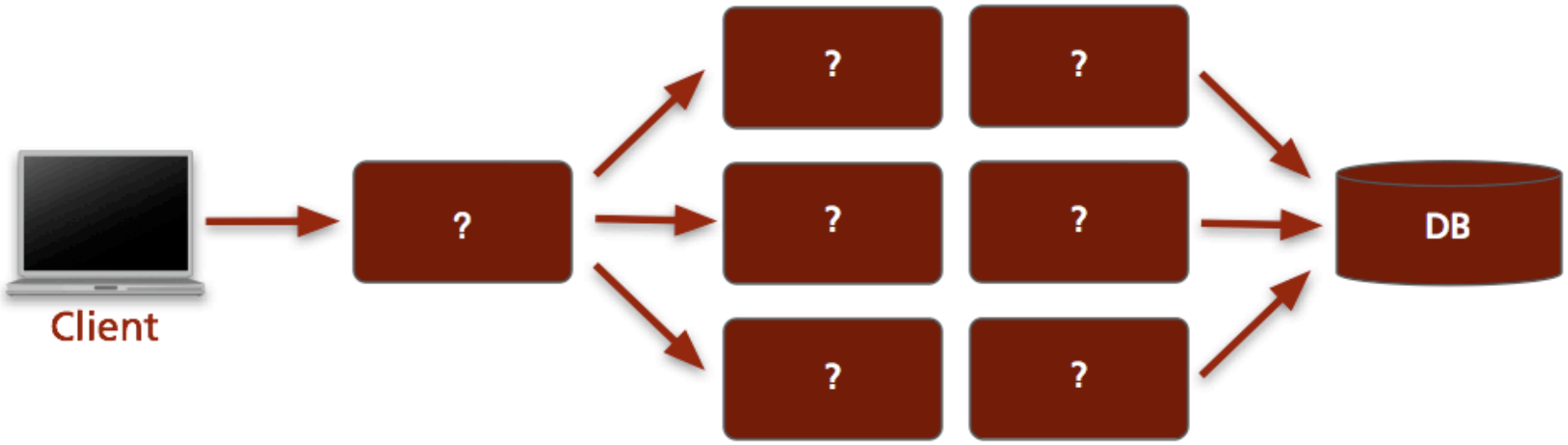
Typical Rails Setup

peritor consulting

- A load-balancer distributes the incoming requests
- Some load-balancers will deliver static requests themselves
- Several Rails instances handle all requests
- Number of concurrent requests equals number of Rails instances

Rails Setup Options

peritor consulting



Deployment Questions

peritor consulting

Apache?

Pound?

mod_proxy_balancer?

FastCGI?

Mongrel?

Load-balancer?

Proxy?

Nginx?

mod_rails?

Ebb?

HA-Proxy?

Reverse Proxy?

Swiftiply?

Phusion Passenger?

Thin?

Pen?

Rails Application Server?

Lighttpd?

What we are going to cover today

peritor consulting

Rails Application Server

- FastCGI
- Mongrel
- mod_rails / Phussion Passenger
- JRuby + Glassfish & Co.

Proxy/Web Server

- Apache2
- Nginx
- Lighttpd
- HA-Proxy

FastCGI

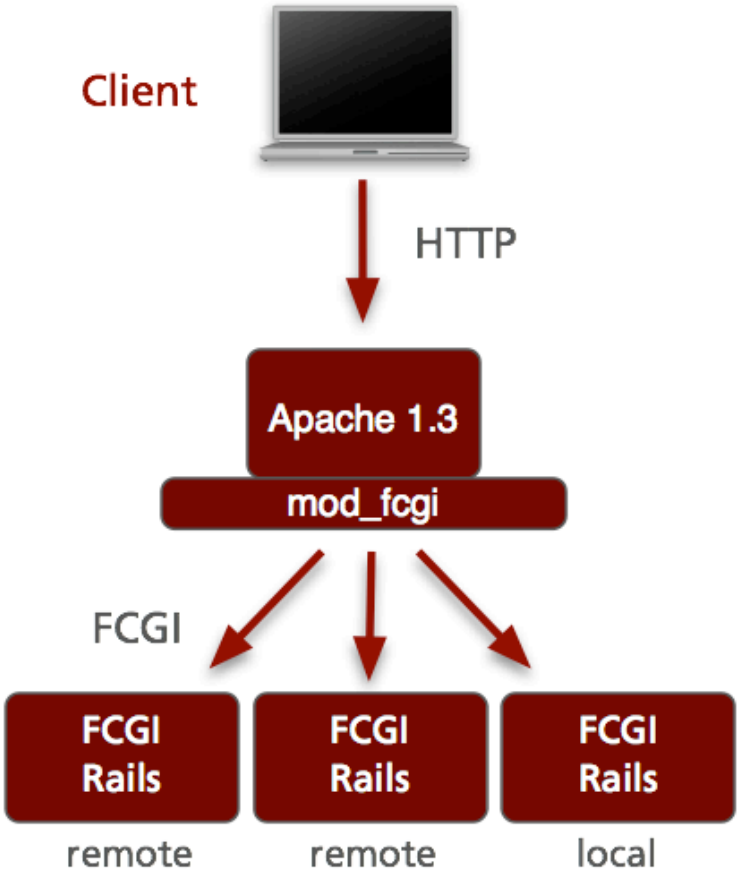
FastCGI

peritor consulting

- Protocol to communicate with long-running CGI applications
- Usage of either mod_fcgi with Apache 1.3 or mod_fcgi with Lighttpd
- Proxy local and remote FastCGI instances
- Oldest way of deploying Rails
- Deprecated and unstable
- Hard to debug (FastCGI protocol)

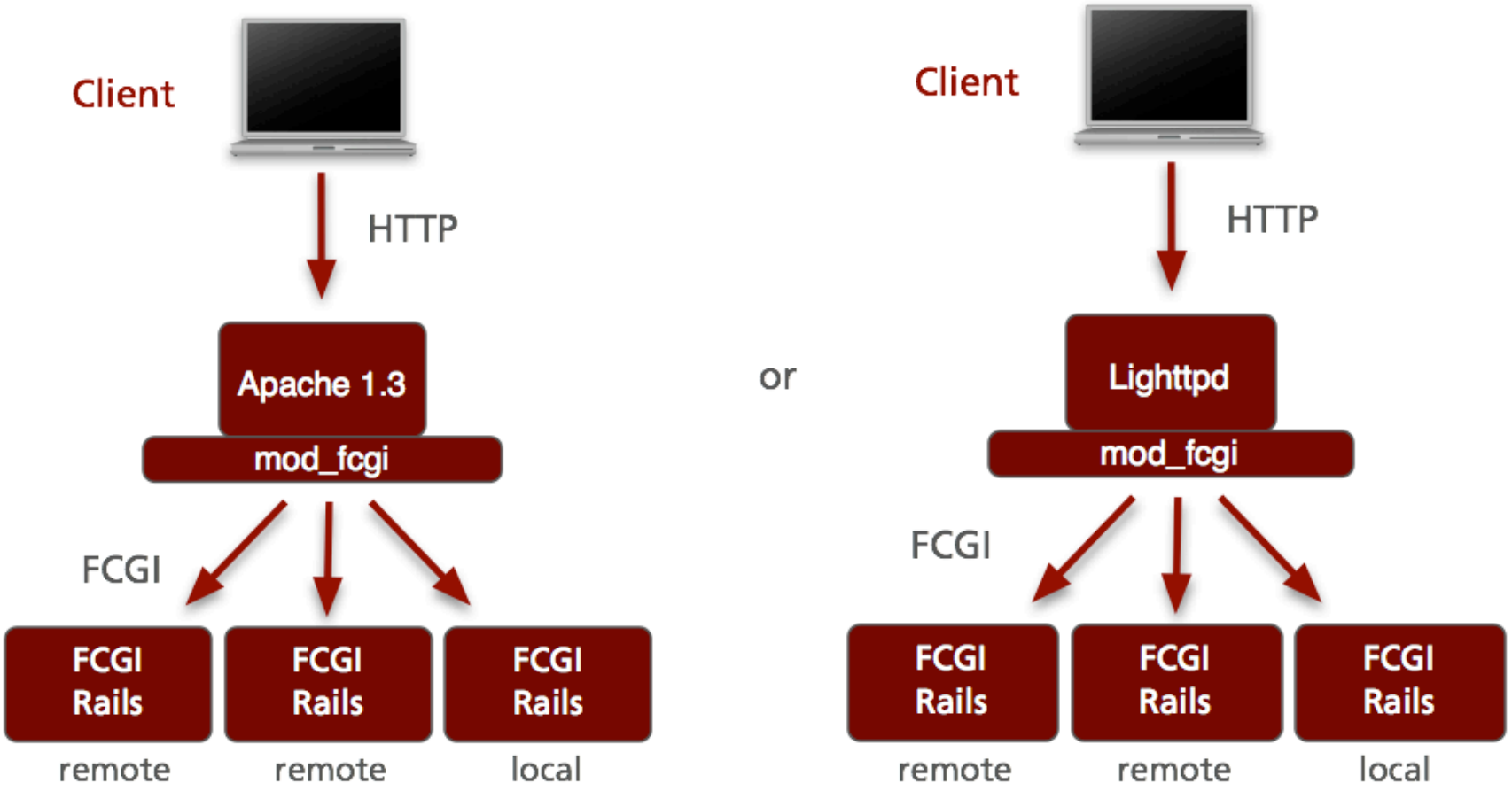
FastCGI

peritor consulting



FastCGI

peritor consulting



peritor consulting

Mongrel

Mongrel

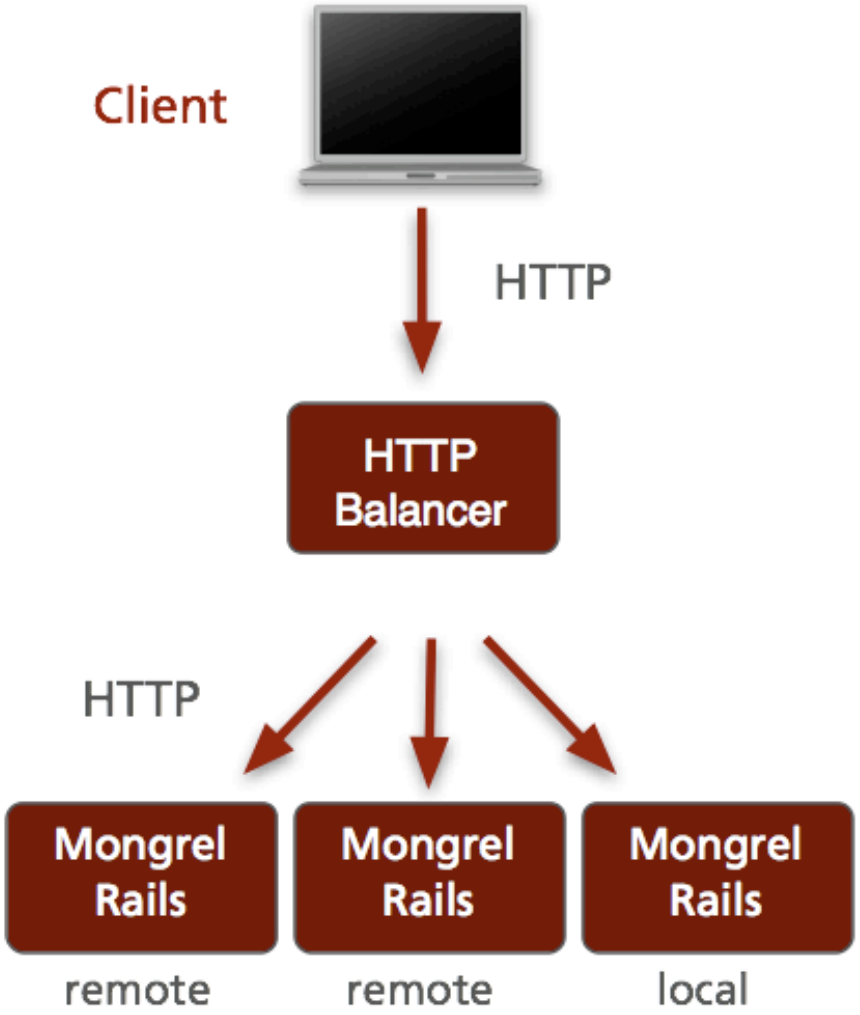
peritor consulting

- Developed by Zed Shaw as an alternative to FastCGI
- Complete HTTP-Server that can load arbitrary Ruby-servlets
- Built-in Rails support

```
$ sudo gem install mongrel

$ cd my/rails/project
$ mongrel_rails start -d -e production -p 80
```

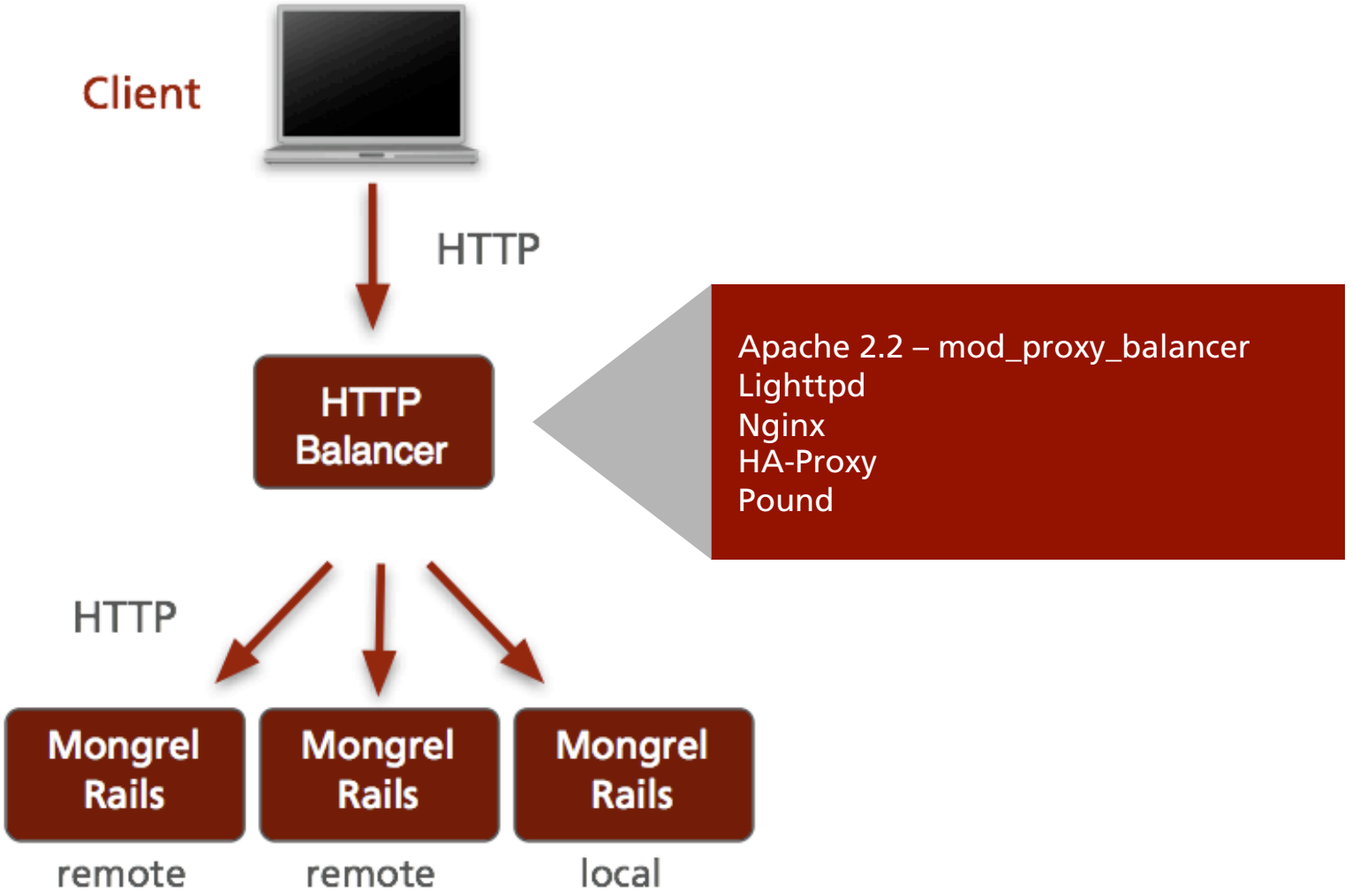
Mongrel

peritor consulting



Mongrel

peritor consulting



Mongrel Cluster

peritor consulting

Utility to manage several Mongrel instances

```
$ sudo gem install mongrel_cluster

$ cat /srv/www/www.example.com/config/mongrel.conf
---
cwd: /srv/www/www.example.com/current
port: 8000
pid_file: /srv/www/www.example.com/shared/log/mongrel.pid
servers: 10
environment: production
```

Mongrel Cluster

peritor consulting

Control Mongrels

```
$ mongrel_rails cluster::start -C /srv/www/www.example.com/config/mongrel.conf

$ mongrel_rails cluster::stop -C /srv/www/www.example.com/config/mongrel.conf

$ mongrel_rails cluster::restart -C /srv/www/www.example.com/config/mongrel.conf
```

Mongrel and Apache 2.2

peritor consulting

```
<Proxy balancer://rails_cluster>
    # local
    BalancerMember http://127.0.0.1:8000 loadfactor=1 max=1 acquire=1
    BalancerMember http://127.0.0.1:8001 loadfactor=1 max=1 acquire=1
    BalancerMember http://127.0.0.1:8002 loadfactor=1 max=1 acquire=1
    BalancerMember http://127.0.0.1:8003 loadfactor=1 max=1 acquire=1

    # remote 1
    BalancerMember http://10.0.1.10:8000 loadfactor=1 max=1 acquire=1
    BalancerMember http://10.0.1.10:8001 loadfactor=1 max=1 acquire=1
    BalancerMember http://10.0.1.10:8002 loadfactor=1 max=1 acquire=1
    BalancerMember http://10.0.1.10:8003 loadfactor=1 max=1 acquire=1

    # remote 2
    BalancerMember http://10.0.1.11:8000 loadfactor=1 max=1 acquire=1
    BalancerMember http://10.0.1.11:8001 loadfactor=1 max=1 acquire=1
    BalancerMember http://10.0.1.11:8002 loadfactor=1 max=1 acquire=1
    BalancerMember http://10.0.1.11:8003 loadfactor=1 max=1 acquire=1
</Proxy>
```

Define Mongrel Cluster in Apache

Simple Mongrel and Apache 2.2

peritor consulting

Redirect all traffic to the Mongel cluster

```
<VirtualHost 10.0.10.1:80>
    ServerAdmin webmaster@www.example.com
    ServerName www.example.com

    ErrorLog /var/log/apache2/www.example.com/error.log
    CustomLog /var/log/apache2/www.example.com/access.log combined

    ProxyPass / balancer://rails_cluster/
    ProxyPassReverse / balancer://rails_cluster/
</VirtualHost>
```

A more complex example

- Redirect dynamic requests
- Serve static content
- Support cached pages
- Support maintenance page
- Enable client-side caching of images, stylesheets, and JavaScript
- Compress output if supported

```
<VirtualHost 10.0.10.1:80>
    ServerAdmin webmaster@www.example.com
    ServerName www.example.com
    ErrorLog /var/log/apache2/www.example.com/error.log
    CustomLog /var/log/apache2/www.example.com/access.log combined
    DocumentRoot /srv/www/www.example.com/current

    <Directory "/srv/www/www.example.com/current/public">
      Options FollowSymLinks
      AllowOverride None
      Order allow,deny
      Allow from all
    </Directory>

    RewriteEngine On
    # Don't do forward proxying
    ProxyRequests Off

    # static content dirs - add slash
    RewriteCond %{DOCUMENT_ROOT}/%{REQUEST_FILENAME} -d
    RewriteRule ^(.+[^/])$ $1/ [R]

    # hide .svn dirs
    RewriteRule ^(.*/)?\.svn/ - [F,L]
    ErrorDocument 403 "Access Forbidden"

    # Check for maintenance file and redirect all requests
    #       ( this is for use with Capistrano's disable_web task )
    RewriteCond %{DOCUMENT_ROOT}/system/maintenance.html -f
    RewriteCond %{SCRIPT_FILENAME} !maintenance.html
    RewriteRule ^.*$ /system/maintenance.html [L]

    # Rewrite index to check for static
    RewriteRule ^/$ /index.html [QSA]

    # Rewrite to check for Rails cached page
    RewriteRule ^([^.]+)$ $1.html [QSA]

    # Redirect all non-static requests to cluster
    RewriteCond %{DOCUMENT_ROOT}/%{REQUEST_FILENAME} !-f
    RewriteRule ^/(.*)$ balancer://rails_cluster%{REQUEST_URI} [P,QSA,L]

    # compress output if supported
    AddOutputFilterByType DEFLATE text/html text/plain text/xml
    BrowserMatch ^Mozilla/4 gzip-only-text/html
    BrowserMatch ^Mozilla/4\.0[678] no-gzip
    BrowserMatch \bMSIE !no-gzip !gzip-only-text/html

    # Caching Header
    <IfModule mod_expires.c>
      <LocationMatch "/(javascripts|stylesheets|images)/">
        ExpiresActive on
        ExpiresDefault "access plus 30 days"
      </LocationMatch>
    </IfModule>

     # disable ETag
     FileETag none
</VirtualHost>
```

Mongrel

peritor consulting

- Very robust
- Strict HTTP parser
- Easy to debug (HTTP!)
- Defacto standard deployment with Apache 2.2 and mod_proxy_balancer
- Can be a bit difficult to setup (mongrel_cluster, ports, Apache)
- Not so easy on mass/virtual hosting

mod_rails

mod_rails a.k.a Phusion Passenger

peritor consulting

- Fairly new module for Apache 2.2
- Allows Apache to control Rails instances
- Apache starts and stops application instances depending on the application load
- Very easy to setup
- Able to run any RACK-compatible Ruby application (Merb & Co.)
- Only manages Rails on one host - no remote instances
- Combine with HTTP-Proxy / balancing solution

Install Phusion Passenger

peritor consulting

Install Apache module

```
$ sudo gem install passenger

$ sudo passenger-install-apache2-module
```

Load and activate in Apache

```
LoadModule passenger_module /usr/local/libexec/apache22/mod_passenger.so
RailsSpawnServer /usr/local/bin/passenger-spawn-server
RailsRuby /usr/local/bin/ruby


<VirtualHost *:80>
    ServerName www.example.com
    DocumentRoot /srv/www/www.example.com/current/public
</VirtualHost>
```

Customized Phusion Passenger

peritor consulting

Control Rails instance number

```
LoadModule passenger_module /usr/local/libexec/apache22/mod_passenger.so
RailsSpawnServer /usr/local/bin/passenger-spawn-server
RailsRuby /usr/local/bin/ruby

PassengerMaxPoolSize 10
PassengerMaxInstancesPerApp 5
PassengerPoolIdleTime 300

<VirtualHost *:80>
    ServerName www.example.com
    DocumentRoot /srv/www/www.example.com/current/public

    RailsEnv production
</VirtualHost>
```

Control Phusion Passenger

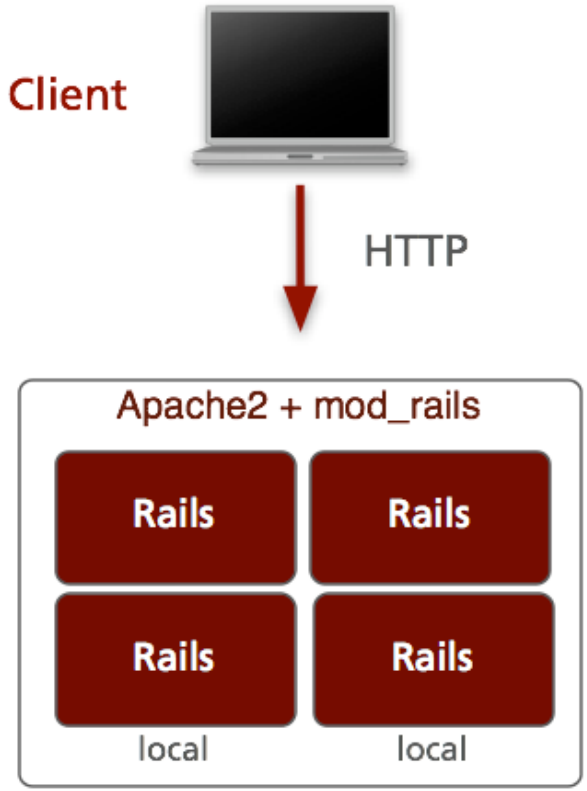
peritor consulting

Restart after deployment:

```
touch /srv/www/www.example.com/current/tmp/restart.txt
```

Phusion Passenger

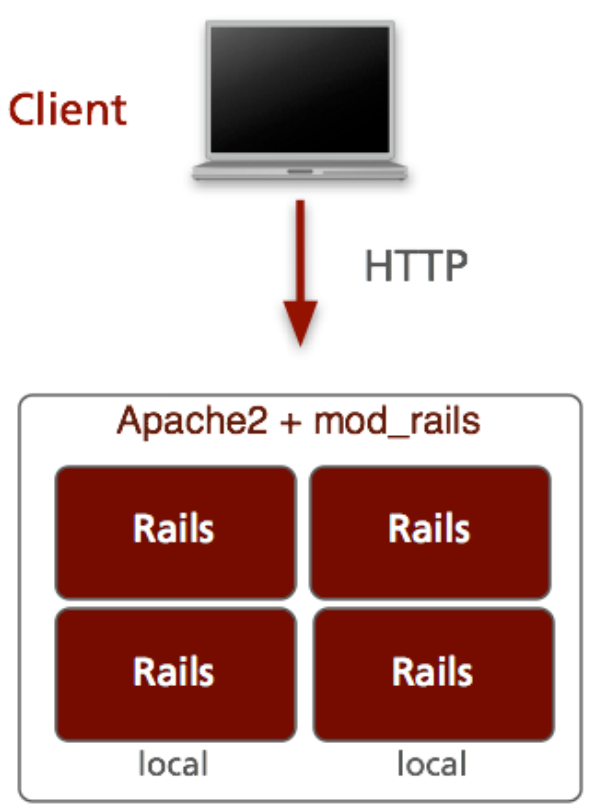
peritor consulting



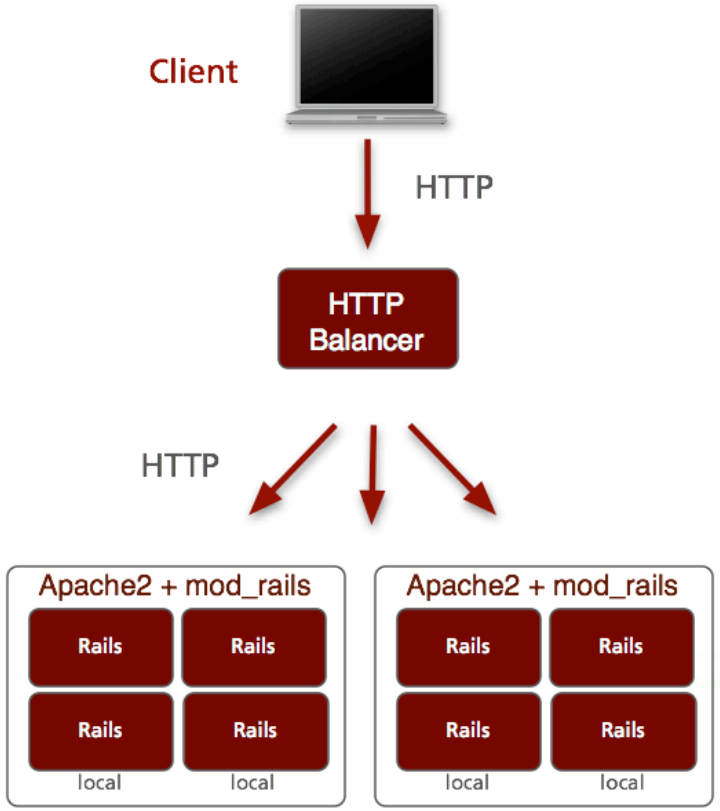
One machine

Phusion Passenger

peritor consulting



One machine



Multiple machines

Phusion Passenger

peritor consulting

- Fairly new but ready for production
- Makes setup easier – on the single machine level
- Multiple servers still require load balancer
- Suitable for mass-hosting
- The upcoming standard way of deploying Rails

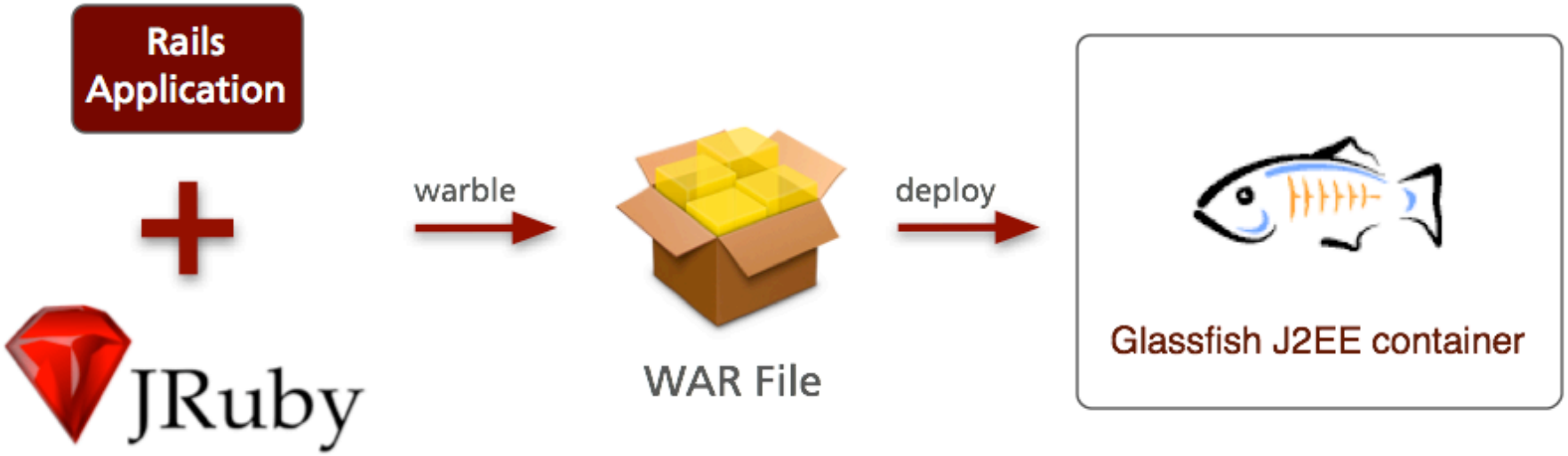
JRuby

peritor consulting

- Ruby Runtime on the Java Virtual Machine
- Implemented in Java and Ruby
- Compiles Ruby into Java-bytecode
- Integrates with Java code and libraries
- Java's promises of native threads and JIT
- Allows for Ruby/Rails applications to be packaged as WAR files
- WAR files deployable on any J2EE-container:
  Glassfish, JBoss, Tomcat, Jetty, …

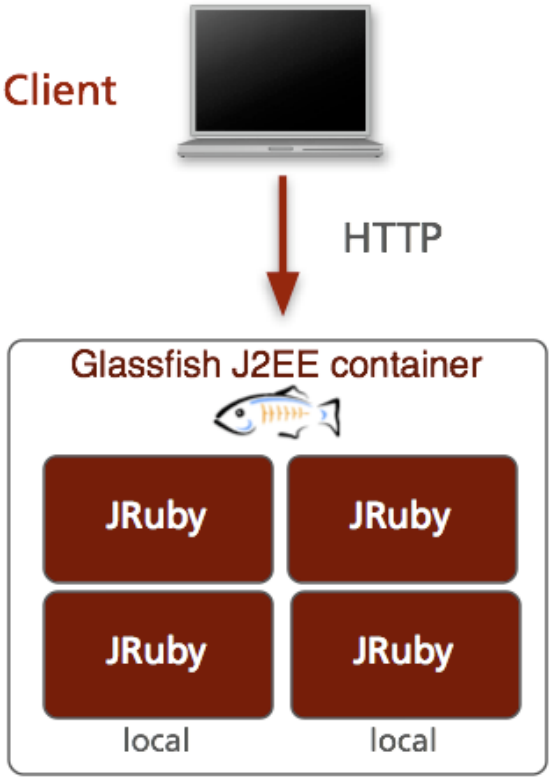
JRuby on Rails

peritor consulting



JRuby on Glassfish

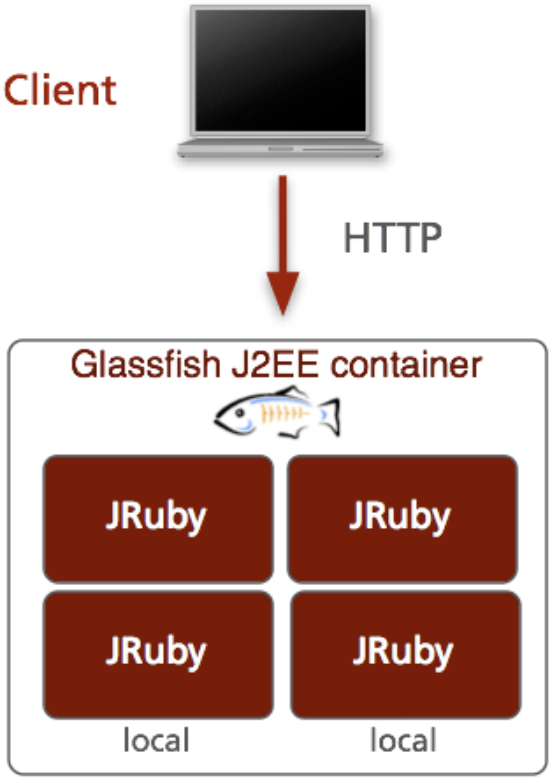
peritor consulting



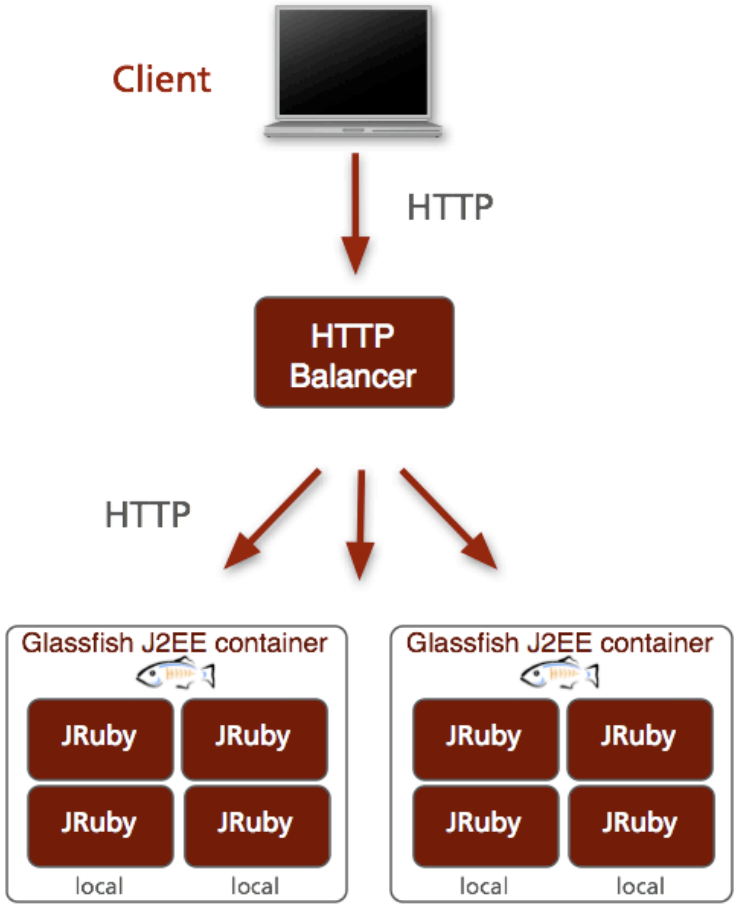
One machine

JRuby on Glassfish

peritor consulting



One machine



Multiple machines

Setup JRuby on Glassfish

peritor consulting

1. Download JRuby and Glassfish
2. From http://blog.headius.com/2008/08/zero-to-production-in-15-minutes.html

```
~/work ➔ java -Xmx256M -jar ~/Downloads/glassfish-installer-v2ur2-b04-darwin.jar
~/work ➔ cd glassfish
~/work/glassfish ➔ chmod a+x lib/ant/bin/*
~/work/glassfish ➔ lib/ant/bin/ant -f setup.xml
~/work/glassfish ➔ bin/asadmin start-domain
~/work/glassfish ➔ cd ~/work/testapp
~/work/testapp ➔ jruby -S gem install warbler
~/work/testapp ➔ jruby -S gem install activerecord-jdbcmysql-adapter
~/work/testapp ➔ vi config/database.yml
~/work/testapp ➔ warble
~/work/testapp ➔ ../glassfish/bin/asadmin deploy --contextroot / testapp.war
```

Warble Configuration

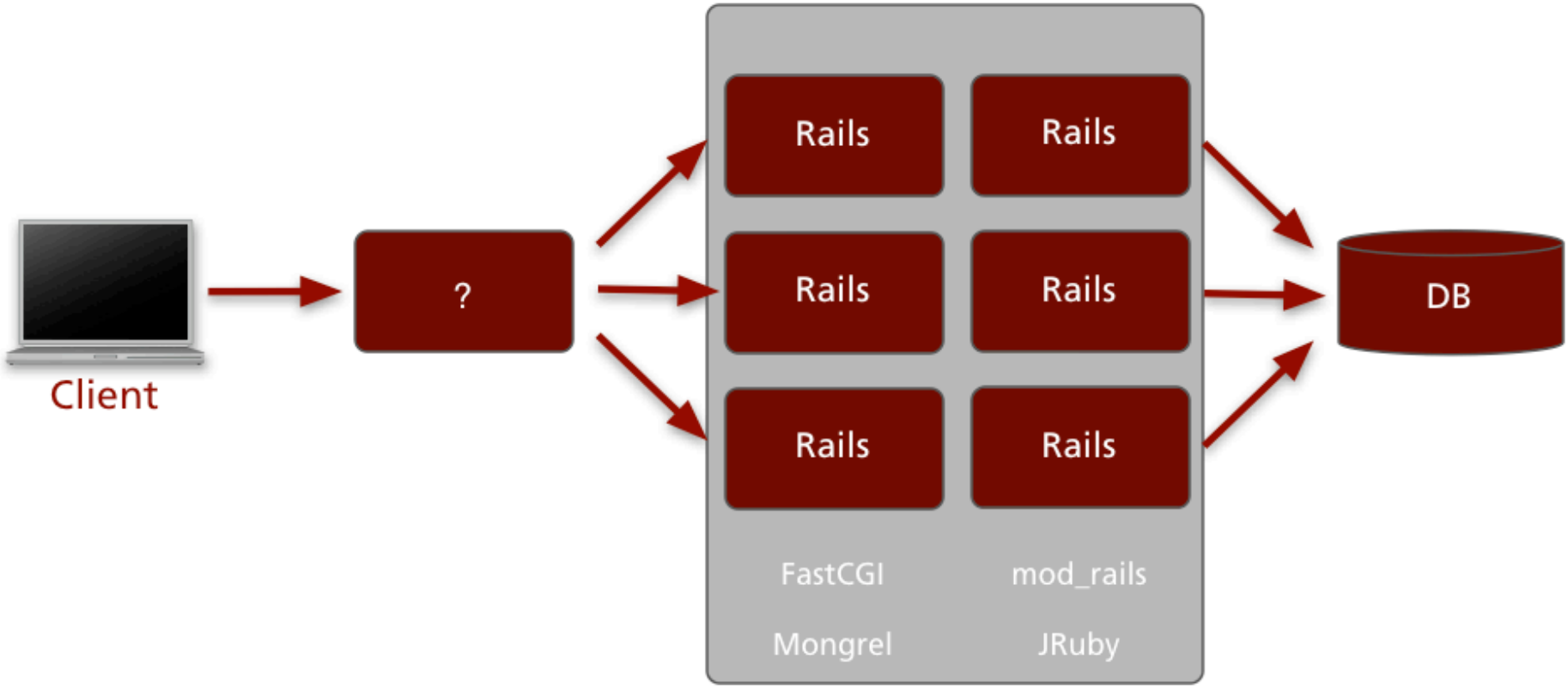
peritor consulting

```
Warbler::Config.new do |config|
  # Application directories to be included in the webapp.
  config.dirs = %w(app config lib log vendor tmp)

  # Additional files/directories to include, above those in config.dirs
  # config.includes = FileList["db"]

  # config.gems += ["activerecord-jdbcmysql-adapter", "jruby-openssl"]
  # config.gems << "tzinfo"

  # Include gem dependencies not mentioned specifically
  config.gem_dependencies = true

  # Name of the war file (without the .war) -- defaults to the basename
  # of RAILS_ROOT
  # config.war_name = "mywar"

  # Value of RAILS_ENV for the webapp -- default as shown below
  # config.webxml.rails.env = ENV['RAILS_ENV'] || 'production'

  # Control the pool of Rails runtimes. Leaving unspecified means
  # the pool will grow as needed to service requests. It is recommended
  # that you fix these values when running a production server!
  config.webxml.jruby.min.runtimes = 2
  config.webxml.jruby.max.runtimes = 4
end
```

Define min/max
Rails runtimes

Rails Setup

peritor consulting



peritor consulting

Proxy Options

Proxy Requirements

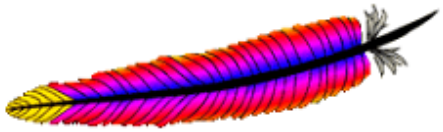
peritor consulting

- Hide cluster backend from the user
- Load-balancer backend instances
- Recognize down hosts
- Fair scheduler
- (Deliver static content)

Apache 2.2

peritor consulting

- Apache 2.2 introduced mod_proxy_balancer
- mod_proxy_balancer can speak to multiple backends and balance requests
- Apache can acts as a pure proxy or can also serve static files



Apache 2.2

peritor consulting

- Apache 2.2 introduced mod_proxy_balancer
- mod_proxy_balancer can speak to multiple backends and balance requests
- Apache can acts as a pure proxy or can also serve static files

```
<Proxy balancer://rails_cluster>
    # local
    BalancerMember http://127.0.0.1:8000 loadfactor=1 max=1 acquire=1
    BalancerMember http://127.0.0.1:8001 loadfactor=1 max=1 acquire=1
    BalancerMember http://127.0.0.1:8002 loadfactor=1 max=1 acquire=1
    BalancerMember http://127.0.0.1:8003 loadfactor=1 max=1 acquire=1

    # remote 1
    BalancerMember http://10.0.1.10:8000 loadfactor=1 max=1 acquire=1
    BalancerMember http://10.0.1.10:8001 loadfactor=1 max=1 acquire=1
    BalancerMember http://10.0.1.10:8002 loadfactor=1 max=1 acquire=1
    BalancerMember http://10.0.1.10:8003 loadfactor=1 max=1 acquire=1

    # remote 2
    BalancerMember http://10.0.1.11:8000 loadfactor=1 max=1 acquire=1
    BalancerMember http://10.0.1.11:8001 loadfactor=1 max=1 acquire=1
    BalancerMember http://10.0.1.11:8002 loadfactor=1 max=1 acquire=1
    BalancerMember http://10.0.1.11:8003 loadfactor=1 max=1 acquire=1
</Proxy>
```

```
<VirtualHost 10.0.10.1:80>
    ServerAdmin webmaster@www.example.com
    ServerName www.example.com

    ErrorLog /var/log/apache2/www.example.com/error.log
    CustomLog /var/log/apache2/www.example.com/access.log combined

    ProxyPass / balancer://rails_cluster/
    ProxyPassReverse / balancer://rails_cluster/
</VirtualHost>
```

```
<Proxy balancer://rails_cluster>
    # local
    BalancerMember http://127.0.0.1:8000 loadfactor=1 max=1 acquire=1
    BalancerMember http://127.0.0.1:8001 loadfactor=1 max=1 acquire=1
    BalancerMember http://127.0.0.1:8002 loadfactor=1 max=1 acquire=1
    BalancerMember http://127.0.0.1:8003 loadfactor=1 max=1 acquire=1

    # remote 1
    BalancerMember http://10.0.1.10:8000 loadfactor=1 max=1 acquire=1
    BalancerMember http://10.0.1.10:8001 loadfactor=1 max=1 acquire=1
    BalancerMember http://10.0.1.10:8002 loadfactor=1 max=1 acquire=1
    BalancerMember http://10.0.1.10:8003 loadfactor=1 max=1 acquire=1

    # remote 2
    BalancerMember http://10.0.1.11:8000 loadfactor=1 max=1 acquire=1
    BalancerMember http://10.0.1.11:8001 loadfactor=1 max=1 acquire=1
    BalancerMember http://10.0.1.11:8002 loadfactor=1 max=1 acquire=1
    BalancerMember http://10.0.1.11:8003 loadfactor=1 max=1 acquire=1
</Proxy>
```

```
<VirtualHost 10.0.10.1:80>
    ServerAdmin webmaster@www.example.com
    ServerName www.example.com
    ErrorLog /var/log/apache2/www.example.com/error.log
    CustomLog /var/log/apache2/www.example.com/access.log combined
    DocumentRoot /srv/www/www.example.com/current

    <Directory "/srv/www/www.example.com/current/public">
      Options FollowSymLinks
      AllowOverride None
      Order allow,deny
      Allow from all
    </Directory>

    RewriteEngine On
    # Don't do forward proxying
    ProxyRequests Off

    # static content dirs - add slash
    RewriteCond %{DOCUMENT_ROOT}/%{REQUEST_FILENAME} -d
    RewriteRule ^(.+[^/])$ $1/ [R]

    # hide .svn dirs
    RewriteRule ^(.*/)?\.svn/ - [F,L]
    ErrorDocument 403 "Access Forbidden"

    # Check for maintenance file and redirect all requests
    #       ( this is for use with Capistrano's disable_web task )
    RewriteCond %{DOCUMENT_ROOT}/system/maintenance.html -f
    RewriteCond %{SCRIPT_FILENAME} !maintenance.html
    RewriteRule ^.*$ /system/maintenance.html [L]

    # Rewrite index to check for static
    RewriteRule ^/$ /index.html [QSA]

    # Rewrite to check for Rails cached page
    RewriteRule ^([^.]+)$ $1.html [QSA]

    # Redirect all non-static requests to cluster
    RewriteCond %{DOCUMENT_ROOT}/%{REQUEST_FILENAME} !-f
    RewriteRule ^/(.*)$ balancer://rails_cluster%{REQUEST_URI} [P,QSA,L]

    # compress output if supported
    AddOutputFilterByType DEFLATE text/html text/plain text/xml
    BrowserMatch ^Mozilla/4 gzip-only-text/html
    BrowserMatch ^Mozilla/4\.0[678] no-gzip
    BrowserMatch \bMSIE !no-gzip !gzip-only-text/html

    # Caching Header
    <IfModule mod_expires.c>
      <LocationMatch "/(javascripts|stylesheets|images)/">
        ExpiresActive on
        ExpiresDefault "access plus 30 days"
      </LocationMatch>
    </IfModule>

     # disable ETag
     FileETag none
</VirtualHost>
```

Apache 2.2

peritor consulting

Pro

- Stable, robust, and mature
- Many people know how to work with Apache
- Integrates well with other modules (SVN, DAV, Auth, …)

Con

- Apache can be complicated to configure
- The stock Apache is quite resource-hungry compared to pure proxy solutions

peritor consulting

- Nginx - popular Russian webserver with good proxy support
- Can load-balance multiple backends and deliver static content
- Quite popular with Mongrel as the Rails backend

Nginx Configuration

Simple proxy example

Get complete version here:

http://brainspl.at/nginx.conf.txt

```
worker_processes  2;

error_log  logs/error.log notice;
pid        logs/nginx.pid;

events {
    worker_connections  16384;
}

http {
    include       conf/mime.types;
    default_type  application/octet-stream;

    sendfile        on;
    tcp_nopush     on;

    keepalive_timeout  65;
    tcp_nodelay        on;

    upstream mongrel {
        server 127.0.0.1:3000;
        server 127.0.0.1:3001;
        server 192.168.1.10:3000;
        server 192.168.1.10:3001;
        server 10.10.100.25:5000;
    }

    server {
        listen       8084;
        server_name  localhost;

        access_log  off;

        location / {
            proxy_pass  http://mongrel;
        }
    }
}
```

Nginx

peritor consulting

Pro

- Stable, robust, and fast
- Uses fewer resources (CPU and RAM) than Apache for proxy-mode and static files
- Simpler configuration file
- Can directly talk to memcached - SSI

Con

- More documentation would be nice
- No equivalent for many Apache modules

Lighttpd

peritor consulting

- Lightweight and fast webserver
- Balancing proxy support
- Good FastCGI support
- Used to be popular – until Mongrel came around

Lighttpd Configuration

peritor consulting

Simple proxy example

```
server.modules = ( "mod_rewrite", "mod_redirect",
                   "mod_access", "mod_accesslog",
                   "mod_compress", "mod_proxy")

$HTTP.host? == "www.example.com" {

  server.document-root = "/my/path/to/app/public/current"

  proxy.balance = "fair"
  proxy.server = ( "/" => (
                            ( "host" => "127.0.0.1", "port" => 8001 ),
                            ( "host" => "127.0.0.1", "port" => 8002 ),
                            ( "host" => "127.0.0.1", "port" => 8003 ),
                            ( "host" => "127.0.0.1", "port" => 8004 )
                          ) )

}
```

Lighttpd

peritor consulting

Pro

- Fast and lightweight
- Uses fewer resources (CPU and RAM) than Apache for proxy-mode and static files
- Simpler configuration file

Con

- Unstable for some people
- Slow development cycle
- More documentation would be nice
- Configuration file can be too simple (virtual host aliasing)
- No equivalent for many Apache modules

peritor consulting

- HAProxy – reliable, high performance TCP/HTTP load balancer
- Proxying and content inspection
- No content serving, just a proxy
- Mature proxy module (fair scheduler)
- ACL support

See also similar Pound and Pen

HAProxy

peritor consulting

Simple proxy example

```
listen app_a_proxy 0.0.0.0:80
  # - equal weights on all servers
  # - check health of app. server every 20 seconds
  server a1 127.0.0.1:8000 weight 1 check inter 20000
  server a1 127.0.0.1:8001 weight 1 check inter 20000
```

HAProxy

Pro

- Mature, stable, robust, and fast
- TCP and HTTP balancing

Con

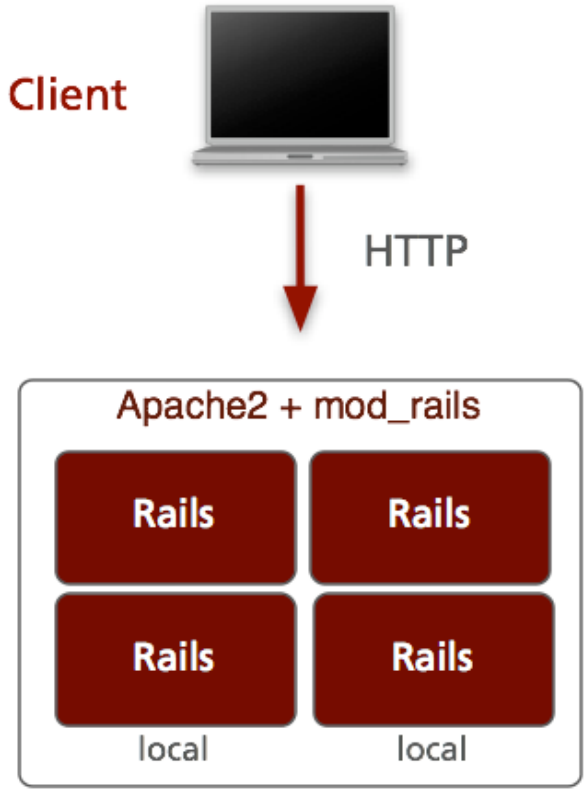
- Few Rails examples
- Usually not needed in a Rails setup

peritor consulting

Recommended Setups

Small Site

peritor consulting

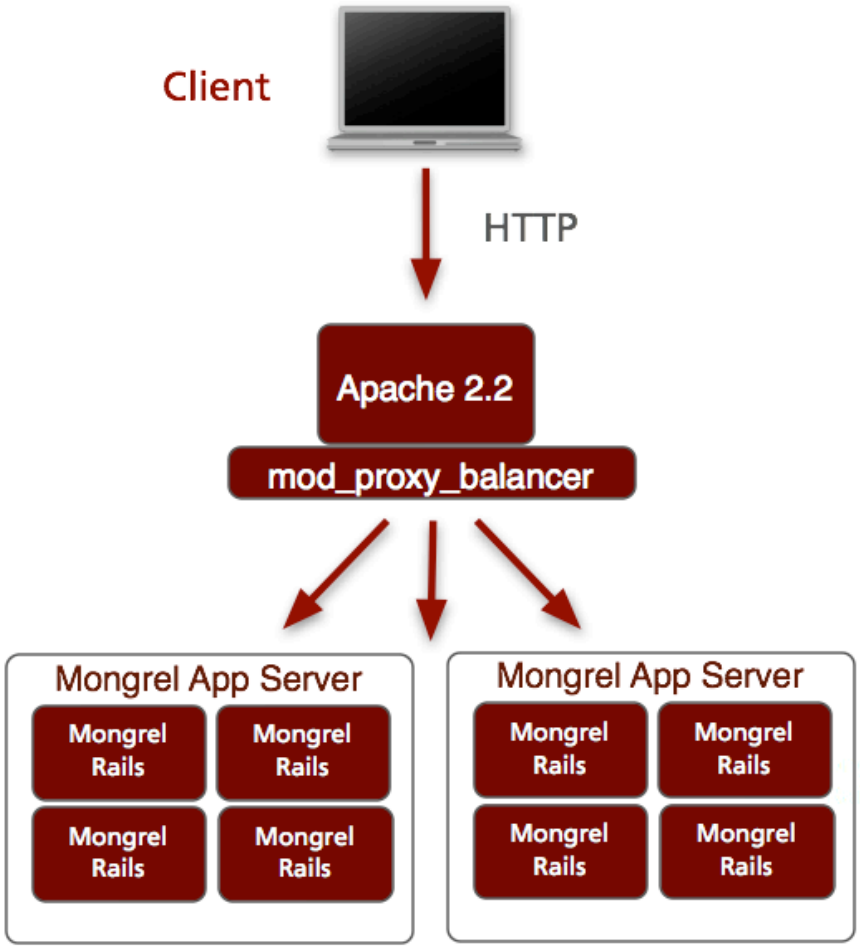


Recommendation

Apache 2.2 with mod_rails / Phusion Passenger

Medium Site

peritor consulting

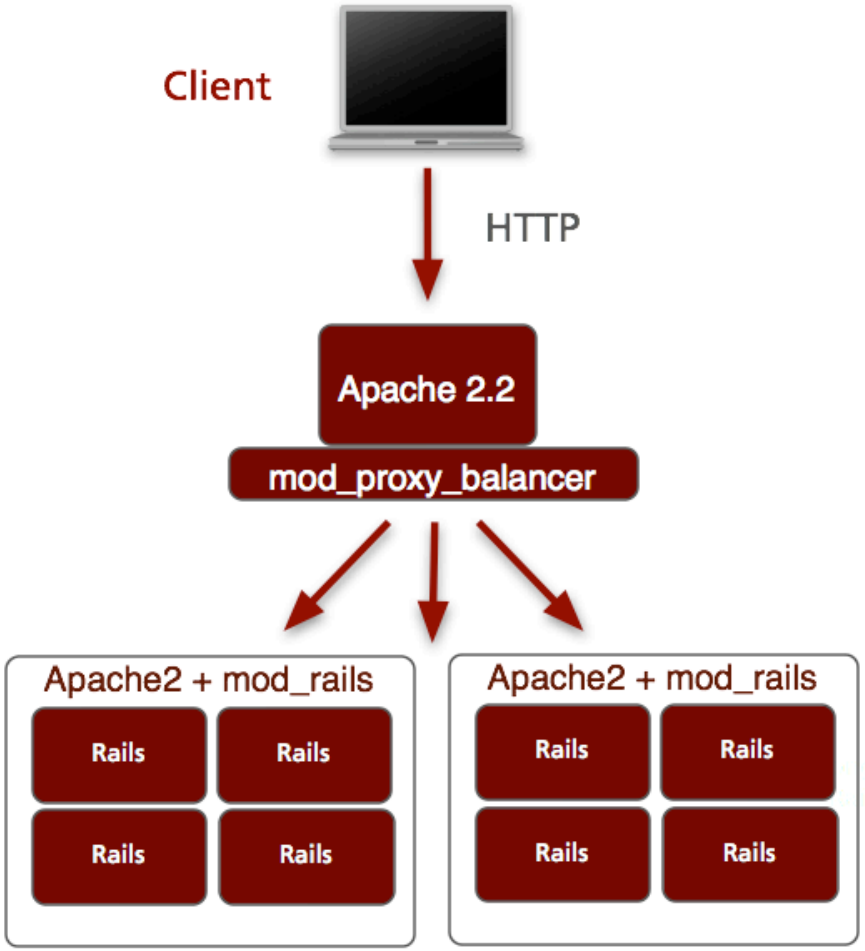


Recommendation

- Apache 2.2 as the frontend proxy
- Use Mongrel or mod_rails as the backend
- Deliver static files with Apache

Medium Site

peritor consulting

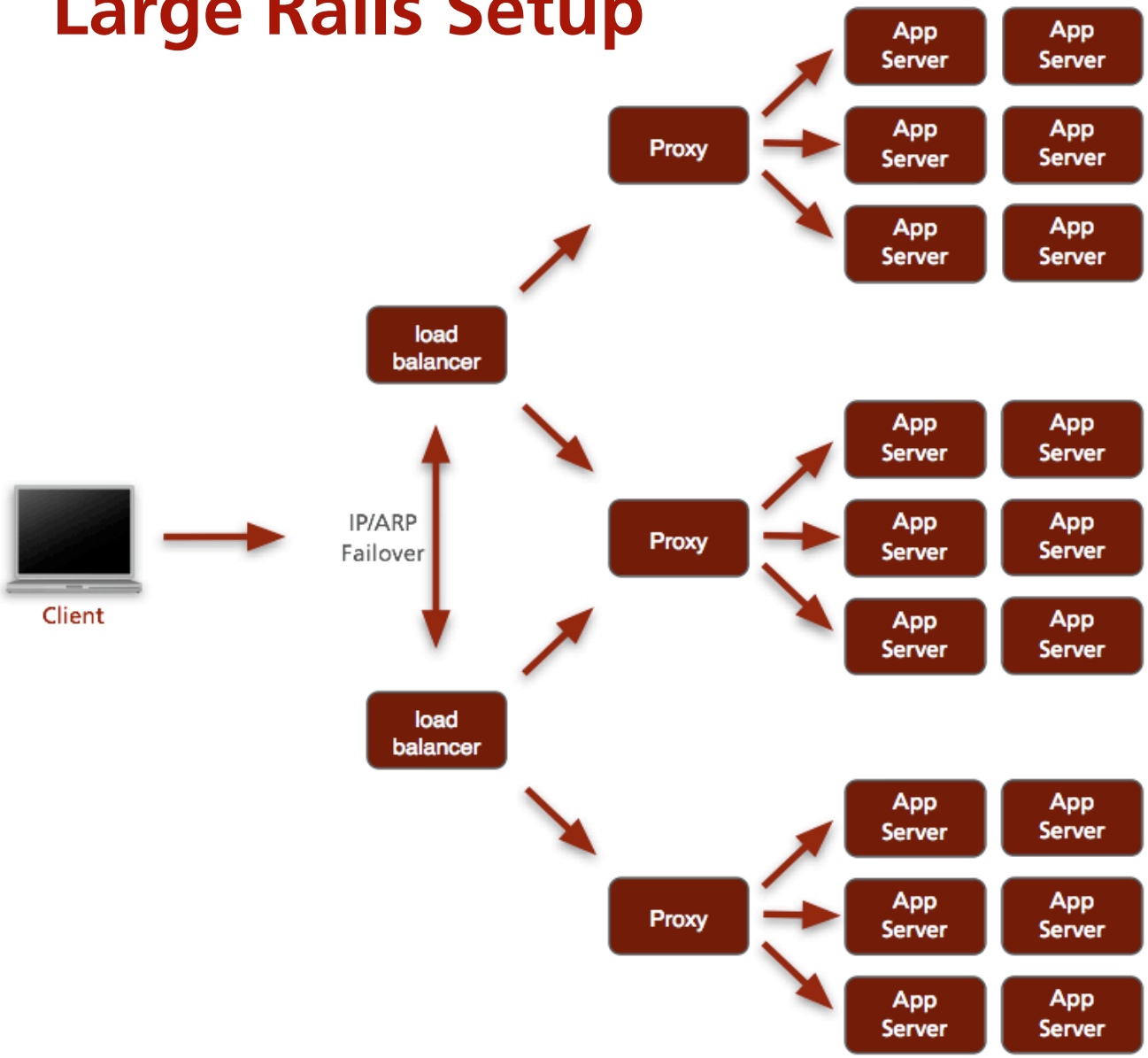


Recommendation

- Apache 2.2 as the frontend proxy
- Use Mongrel or mod_rails as the backend
- Deliver static files with Apache

Large Rails Setup

peritor consulting



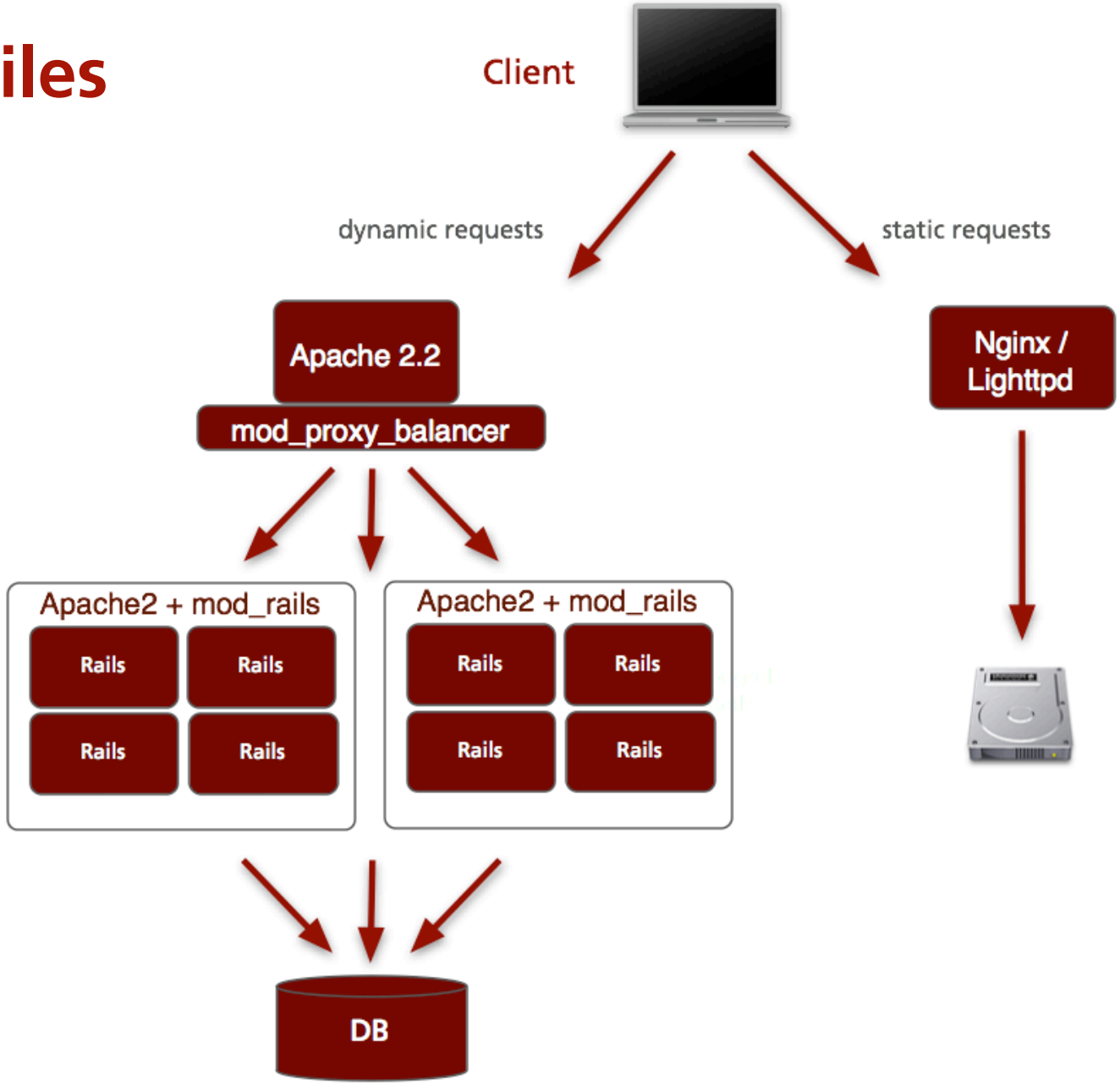
Recommendation

- Redundant load-balancer
- Redundant proxy / web
- Mongrel / mod_rails

Heavy Static Files

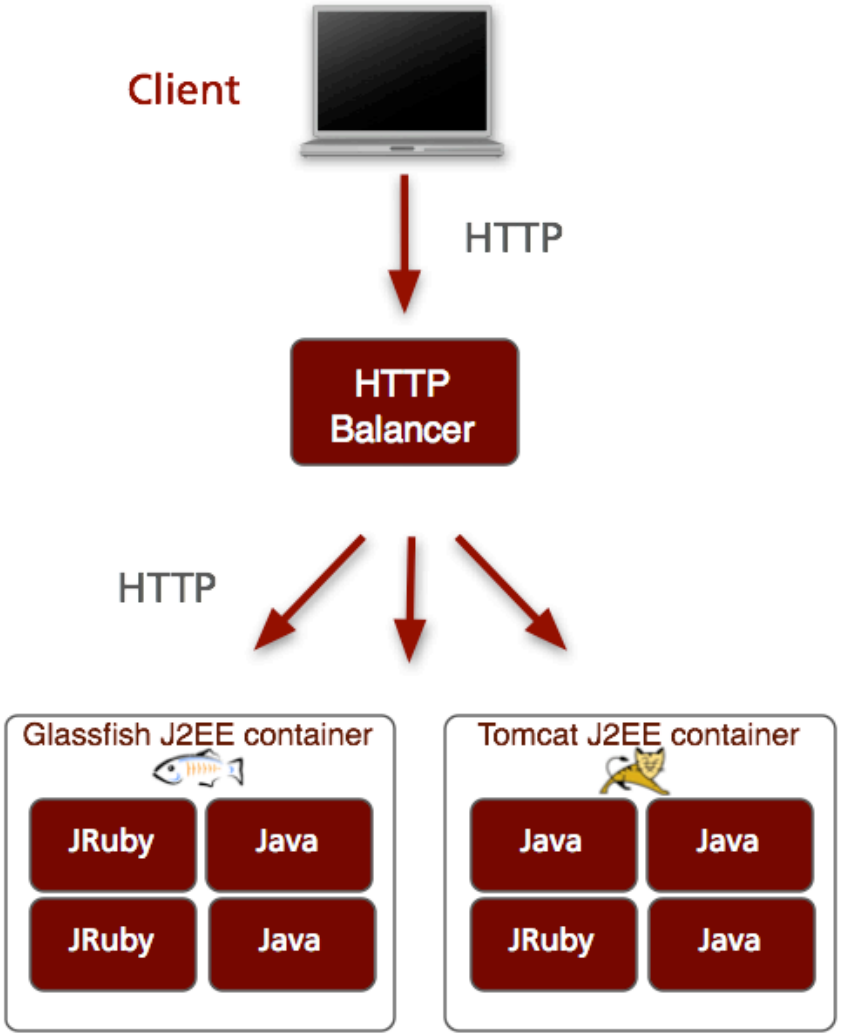
Recommendation

- Deliver static assets through separate web server farm
- Mongrel or mod_rails



Java Shop

peritor consulting



Recommendation

- Deliver a Rails-WAR file and you are done
- Integrate with existing Java landscape and infrastructure

peritor consulting

Remarks

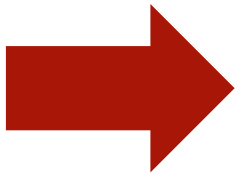
Ruby Enterprise Edition

- Copy-On-Write patches to Ruby 1.8
- Saves memory when spawning several Rails instances
- Used by Phusion Passenger if available

Thin, Ebb, Evented Mongrel & Co.

peritor consulting

- Alternatives to Mongrel
- Claim to be faster, lighter, and what have you
- Rendering “Hello World” is usually not your bottleneck



Stick with stable and robust Mongrel

In The Future, Watch

Fuzed

Erlang based load balancing

Dynamic registration

JRuby

Faster Runtime

Java Integration

Phusion Passenger

Enterprise Ruby

Performance

Usability

Deployment

Deployment Options

peritor consulting

FTP

Deployment Options

peritor consulting

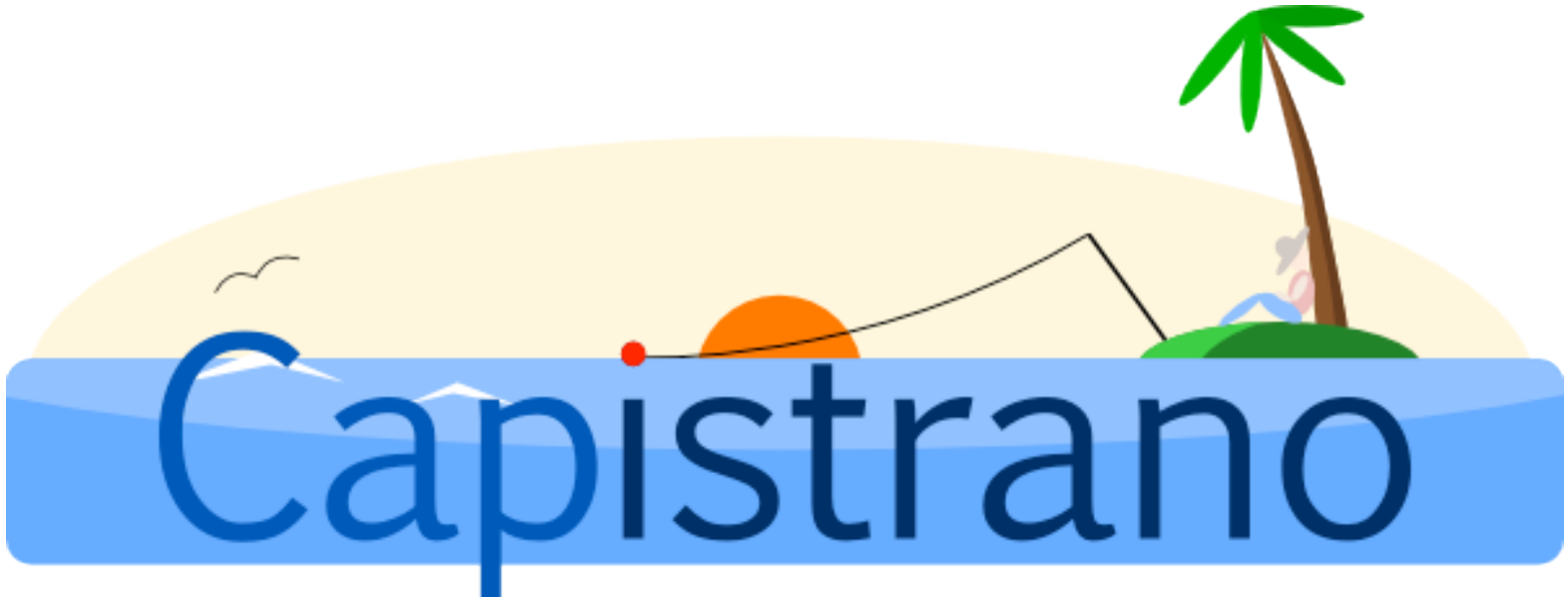
SCP

Deployment Options

peritor consulting

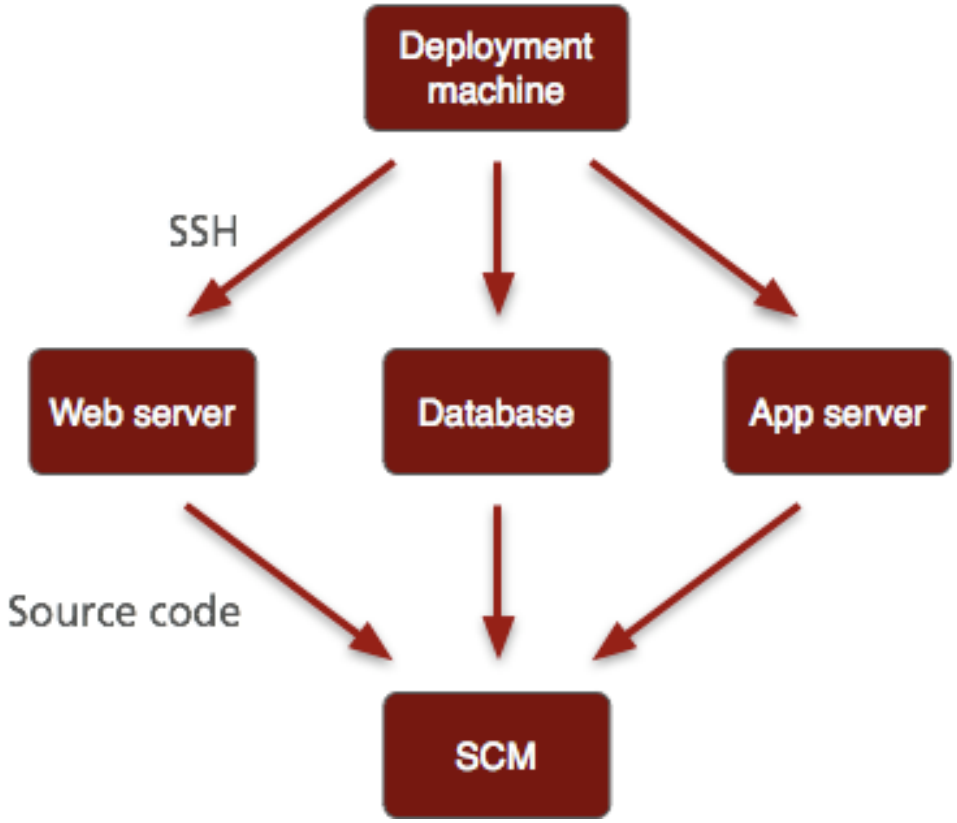
Deployment Options

peritor consulting



What does Capistrano do?

peritor consulting



Capistrano Deployment Cycle

peritor consulting

Requirements

peritor consulting

Shell

SCM

What doesn't Capistrano do?

peritor consulting

- Plan your initial server setup
- Configure basic services

Basic Ingredients

peritor consulting

- The cap command
- Variables
- Roles
- Tasks
- Namespaces

Basic Ingredients - cap

peritor consulting

Your one-stop deployment shop

```
$ cap deploy
```

Basic Ingredients - Variables

peritor consulting

- Configure basic project information
- Override Capistrano's default assumptions
- Once set, variables are available globally
- Defined using the set method

```
set :user, "ruth"
set :home, "/home/#{user}"
```

Basic Ingredients - Roles

peritor consulting

- Define types of servers
- Default roles
  - :www
  - :app
  - :db
- All can point to the same server
- But all three must be defined

```
role :app, "app1.example.com"
role :www, "web1.example.com"
role :db, "db1.example.com", :primary => true
```

- At least one database server needs to be primary

Basic Ingredients - Roles

peritor consulting

Define custom roles as you please

```
role :solr, "search.example.com"
role :activemessaging, "poller.example.com"
```

Can be reused when defining tasks

Basic Ingredients - Tasks

peritor consulting

- Define an atomic set of actions for Capistrano
- Can be called from the command line
- Or other tasks

Basic Ingredients - Tasks

peritor consulting

To find all the tasks available in your project, use

```
$ cap -T
cap deploy                # Deploys your project.
cap deploy:check          # Test deployment dependencies.
cap deploy:cleanup        # Clean up old releases.
cap deploy:cold           # Deploys and starts a `cold' application.
cap deploy:migrate        # Run the migrate rake task.
cap deploy:migrations     # Deploy and run pending migrations.
cap deploy:pending        # Displays the commits since your last deploy.
cap deploy:pending:diff   # Displays the `diff' since your last deploy.
cap deploy:restart        # Restarts your application.
cap deploy:rollback       # Rolls back to a previous version and restarts.
cap deploy:rollback_code  # Rolls back to the previously deployed version.
cap deploy:setup          # Prepares one or more servers for deployment.
....
```

Basic Ingredients - Namespaces

peritor consulting

Group tasks together logically

```
cap deploy:restart
cap solr:reindex
```

Namespaces and tasks are separated with ":"

Get Your Capistrano On

peritor consulting

```
$ gem install capistrano
$ cd ~/development/rails_petstore/
$ capify .
```

Get Your Capistrano On

peritor consulting

Capfile, the place to include more recipes

```
load 'deploy' if respond_to?(:namespace) # cap2 differentiator
Dir['vendor/plugins/*/recipes/*.rb'].each { |plugin| load(plugin) }
load 'config/deploy'
```

Get Your Capistrano On

peritor consulting

config/deploy.rb, application specific configuration

```
set :application, "set your application name here"
set :repository,  "set your repository location here"

# If you aren't deploying to /u/apps/#{application} on the target
# servers (which is the default), you can specify the actual location
# via the :deploy_to variable:
# set :deploy_to, "/var/www/#{application}"

# If you aren't using Subversion to manage your source code, specify
# your SCM below:
# set :scm, :subversion

role :app, "your app-server here"
role :web, "your web-server here"
role :db,  "your db-server here", :primary => true
```

Capistrano's Defaults

peritor consulting

- Your SCM is Subversion
- Deployment directory is /u/apps/#{application_name}
- User for SCM and SSH is the currently logged-in user
- Commands are run with sudo

Get Your Capistrano On

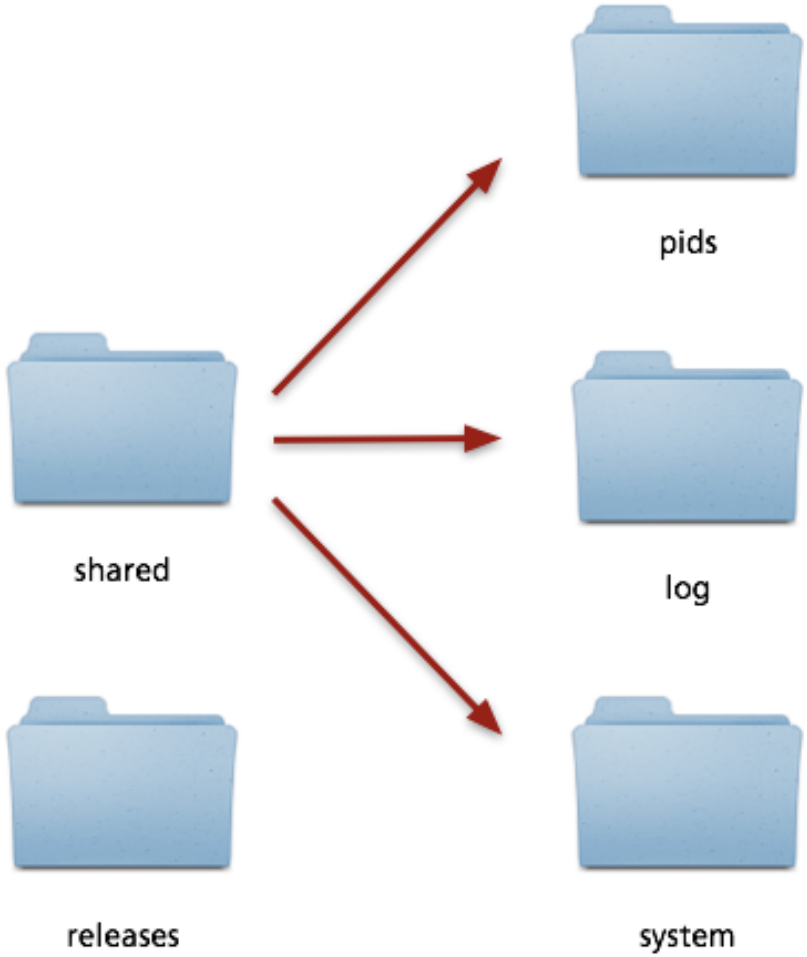
peritor consulting

```
set :application, "rails_petstore"
set :repository,  "git://github.com/mattmatt/rails_petstore.git"
set :deploy_to, "/var/www/#{application}"
set :use_sudo, false

set :scm, :git

role :app, "rails.petstore.com"
role :web, "www.petstore.com"
role :db,  "db.petstore.com", :primary => true
```

Get Your Capistrano On

peritor consulting

- Capistrano expects a directory structure
- Can be created with cap deploy:setup



The Deployment Lifecycle

The Deployment Lifecycle

peritor consulting

Check the prerequisites:

```
$ cap deploy:check
* executing 'deploy:check'
* executing "test -d /home/cftuser/simplelog/releases"
  servers: ["192.168.1.55"]
Password:
  [192.168.1.55] executing command
  command finished
* executing "test -w /home/cftuser/simplelog"
  servers: ["192.168.1.55"]
  [192.168.1.55] executing command
  command finished
* executing "test -w /home/cftuser/simplelog/releases"
  servers: ["192.168.1.55"]
  [192.168.1.55] executing command
  command finished
* executing "which git"
  servers: ["192.168.1.55"]
  [192.168.1.55] executing command
  command finished
You appear to have all necessary dependencies installed
```

The Deployment Lifecycle

peritor consulting

Set up your application for the first time

```
$ cap deploy:setup
* executing 'deploy:setup'
* executing "umask 02 && mkdir -p /home/ruth/rails_petstore /home/ruth/rails_petstore/releases
             /home/ruth/rails_petstore/shared /home/ruth/rails_petstore/shared/system
             /home/ruth/rails_petstore/shared/log /home/ruth/rails_petstore/shared/pids"
  servers: ["192.168.1.55"]
Password:
  [192.168.1.55] executing command
  command finished
```

The Deployment Lifecycle

peritor consulting

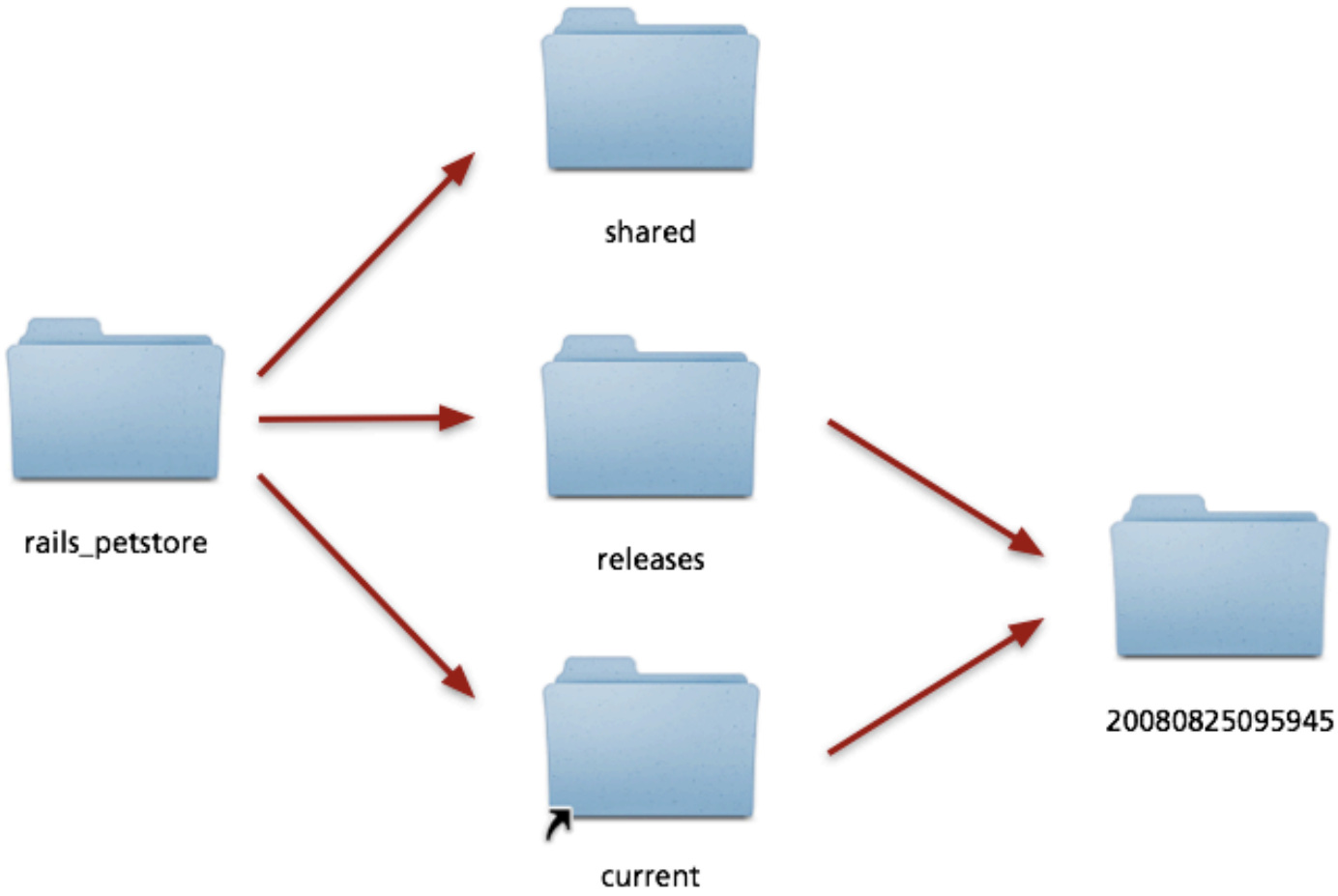
The initial deployment

```
$ cap deploy:cold
```

1. Checks the revision from the local machine
2. Checks out the code on the remote machines
3. Sets a link called current pointing to the lates release
4. Runs the migrations
5. Fires up application servers

The Deployment Lifecycle

peritor consulting



The Deployment Lifecycle

peritor consulting

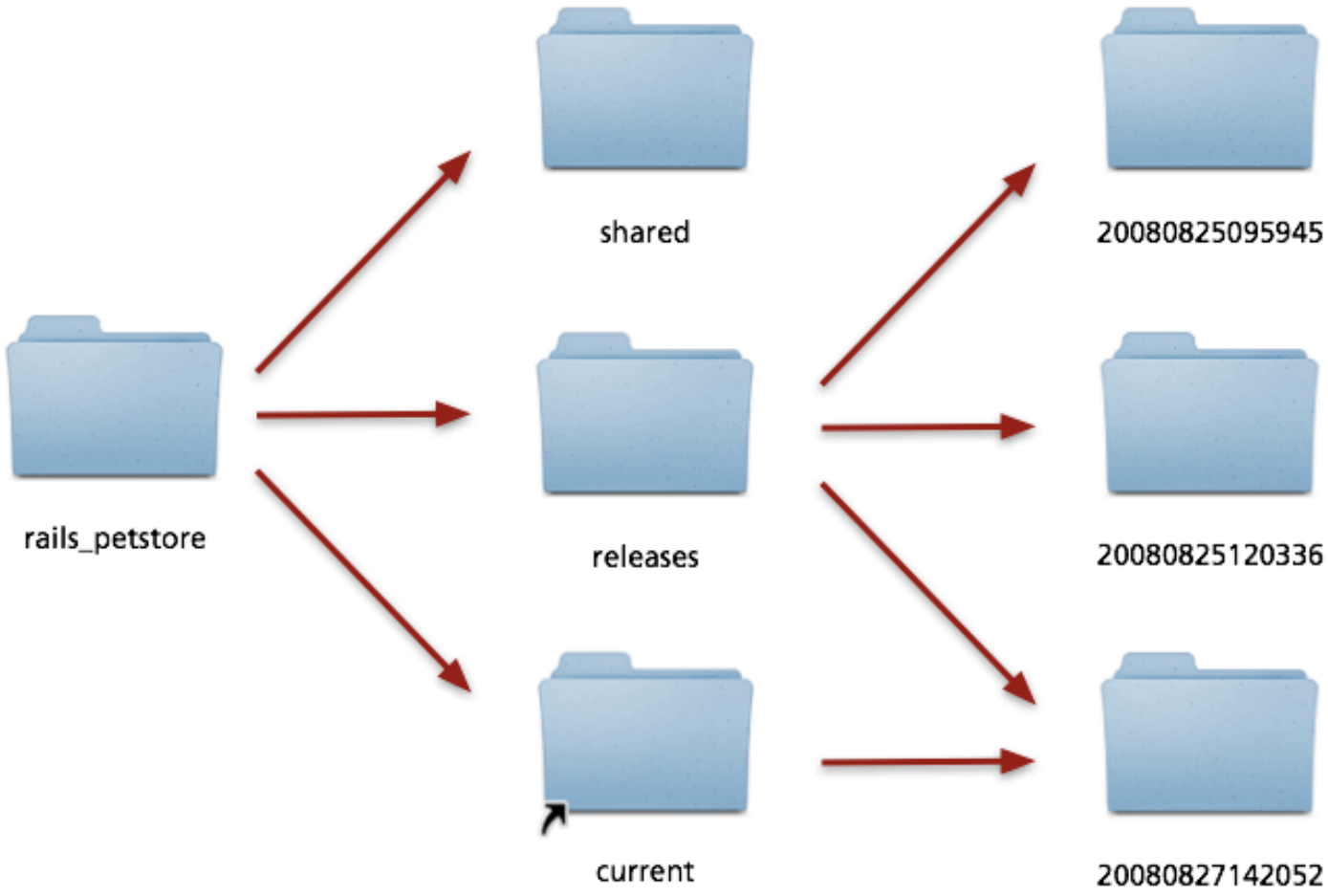
Subsequent deployments

```
$ cap deploy
```

1. Checks the revision from the local machine
2. Checks out the code on the remote machines
3. Updates current link
4. Restarts application servers

The Deployment Lifecycle

peritor consulting



Common Capistrano Tasks

peritor consulting

Deploy and run migrations

```
$ cap deploy:migrate
```

Run only the migrations

```
$ cap deploy:migrations
```

Restart application servers

```
$ cap deploy:restart
```

Rollback to the previous release

```
$ cap deploy:rollback
```

Have Your Shell, and Eat It Too

peritor consulting

```
$ cap shell
* executing 'shell'
================================================================================
Welcome to the interactive Capistrano shell! This is an experimental
feature, and is liable to change in future releases. Type 'help' for
a summary of how to use the shell.
--------------------------------------------------------------------------------
cap> ls
[establishing connection(s) to 192.168.2.106]
Password:
** [out :: 192.168.2.106] rails_petstore
** [out :: 192.168.2.106] simplelog
cap>
```

Invoking any Command

peritor consulting

```
$ cap invoke COMMAND="ls -l"
* executing 'invoke'
* executing "ls -l"
  servers: ["192.168.2.106"]
Password:
  [192.168.2.106] executing command
** [out :: 192.168.2.106] total 2
** [out :: 192.168.2.106] drwxrwxr-x 4 cftuser cftuser 1024 Aug 24 23:54 rails_petstore
** [out :: 192.168.2.106] drwxrwxr-x 4 cftuser cftuser 1024 Aug 23 22:15 simplelog
  command finished
```

Deployment Strategies

peritor consulting

Deployment Strategies

peritor consulting

Direct checkout (from scratch) on the servers

```
set :deploy_via, :checkout
set :deploy_via, :export
```

Deployment Strategies

peritor consulting

Keep a cached copy of the current SCM head

```
set :deploy_via, :remote_cache
set :repository_cache, "/var/www/checkout/rails_petstore"
```

Deployment Strategies

peritor consulting

Check out locally and transfer

```
set :deploy_via, :copy
set :copy_strategy, :checkout
set :copy_remote_dir, "/var/tmp/rails_petstore"
set :copy_dir, "/tmp/rails_petstore"
```

Give it a little Spin

peritor consulting

Capistrano expects a script called spin in script/process

```
#/bin/sh
/var/www/rails_petstore/script/process/spawner -p 11000 -i 3
```

For Passenger

```
#/bin/sh
touch /var/www/rails_petstore/tmp/restore.txt
```

Customizing Capistrano

Write your own Tasks

peritor consulting

```
task :gem_list do
  run "gem list"
end
```

Write your own Tasks

peritor consulting

```
desc "Starts the poller service"
task :start, :roles => :poller, :except => { :no_release => true } do
  ["long_running", "short_running"].each do |group|
    invoke_command "cd #{deploy_to}/current &&
                    DISABLE_MASOCHISM=1 TARGET_ENV='#{site_target}'
                    RAILS_ENV=#{rails_env}
                    ruby script/poller start -- process-group=#{group}; echo '0' ",
                    :via => run_method
  end
end
```

Namespace your Tasks

peritor consulting

```
namespace :poller do
  desc "Starts the poller service"
  task :start, :roles => :poller, :except => { :no_release => true } do
    ...
  end

  desc "Stops the poller service"
  task :stop, :roles => :poller, :except => { :no_release => true } do
    ...
  end

  desc "Restarts the poller service"
  task :restart, :roles => :poller, :except => { :no_release => true } do
  end
end
```

Callbacks

peritor consulting

Execute a task before another runs

```
before "deploy:restart", "deploy:web:disable"
```

Execute a task after another has finished

```
after "deploy:update_code", "poller:restart"
```

Callbacks are run in the order they're defined

Useful Variables

peritor consulting

```
current_path:     /var/www/rails_petstore/current
release_path:     /var/www/rails_petstore/releases/20080829104344
shared_path:      /var/www/rails_petstore/shared
current_release:  /var/www/rails_petstore/releases/20080829082827
previous_release: /var/www/rails_petstore/releases/20080828194529
```

Transactions

peritor consulting

```
task :reindex do
  transaction do
    store_existing_index
    create_new_index
    replace_index
  end
end
```

Transactions

peritor consulting

```
task :store_existing_index do
  # copy index
end
```

```
task :create_new_index do
  on_rollback {
    # remove new index
  }
  # indexing...
end
```

```
task :replace_index do
  on_rollback {
    # restore old index
  }
  # moving around indexes...
end
```

Transactions

peritor consulting

```
task :replace_index do
  on_rollback {
    # restore old index
  }
  # moving around indexes...
end
```

Transactions

peritor consulting

```
task :reindex do
  transaction do
    store_existing_index
    create_new_index
    replace_index
  end
end
```

```
task :store_existing_index do
  # copy index
end
```

```
task :create_new_index do
  on_rollback {
    # remove new index
  }
  # indexing...
end
```

```
task :replace_index do
  on_rollback {
    # restore old index
  }
  # moving around indexes...
end
```

The Rest

peritor consulting

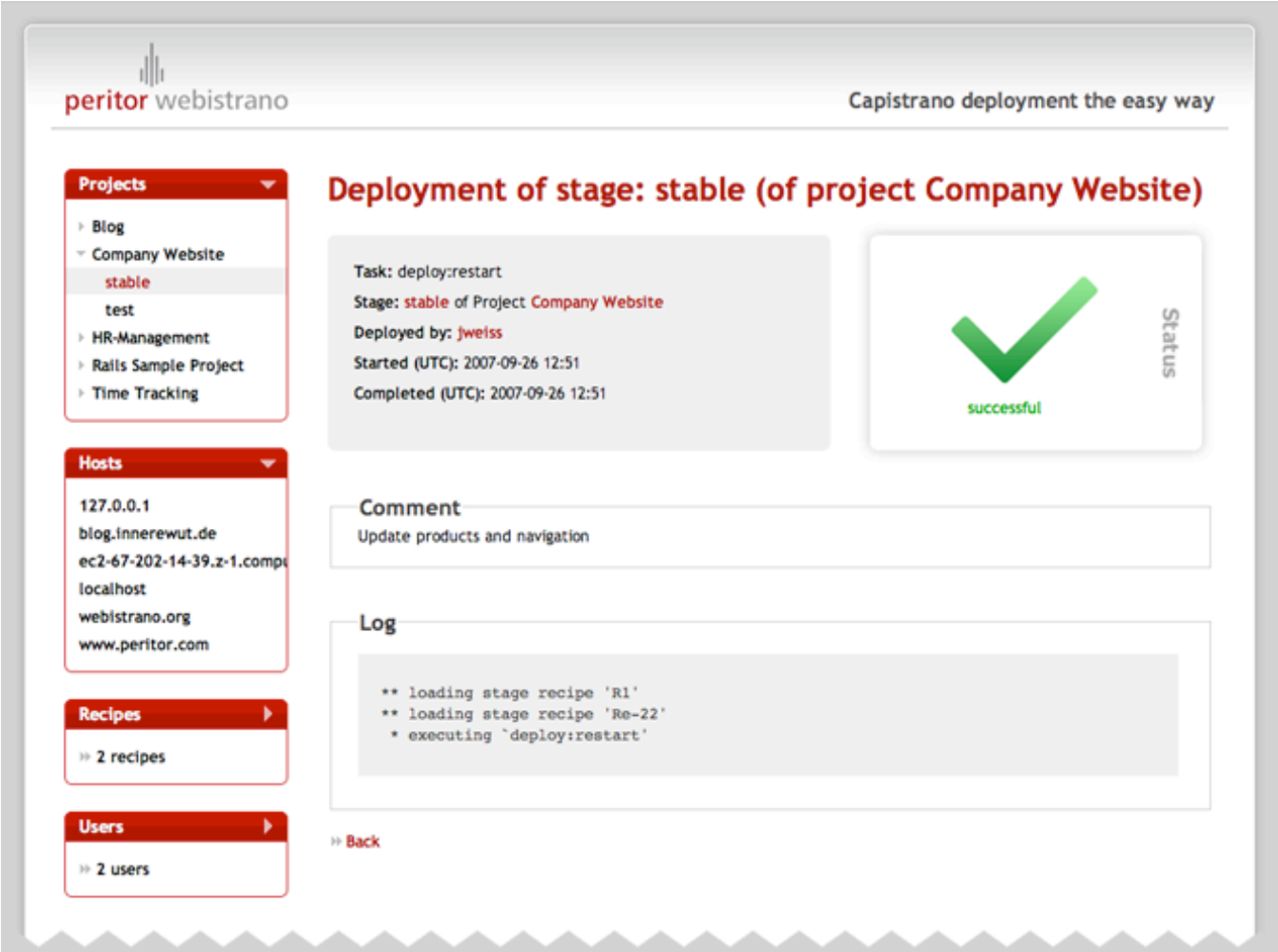
- Gem dependencies
- Support for deploying through gateway servers
- Server setup with deprec gem
- Lack of documentation

peritor consulting

One Click Deploy

Webistrano

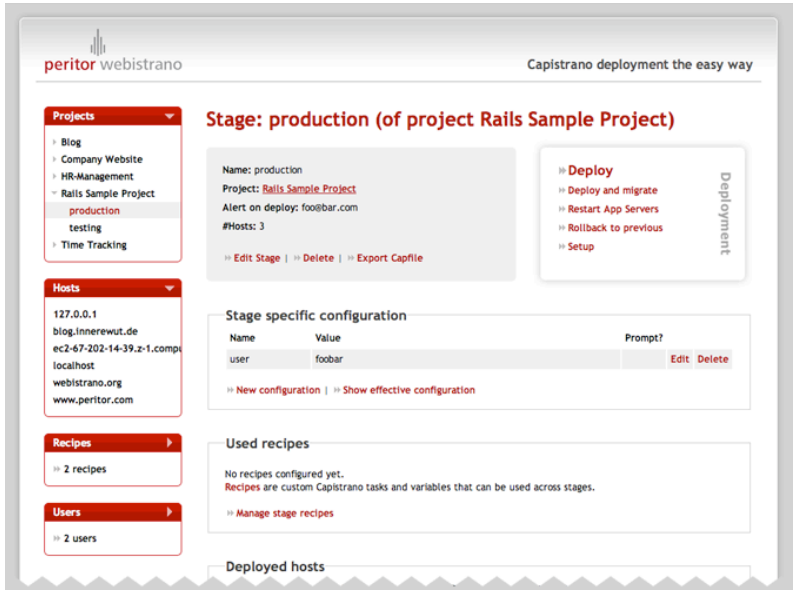
peritor consulting



Webistrano

peritor consulting

- Web-UI to Capistrano
- Manages projects and their stages
- Alerting and Accounting
- Scriptable and extendable
- BSD License

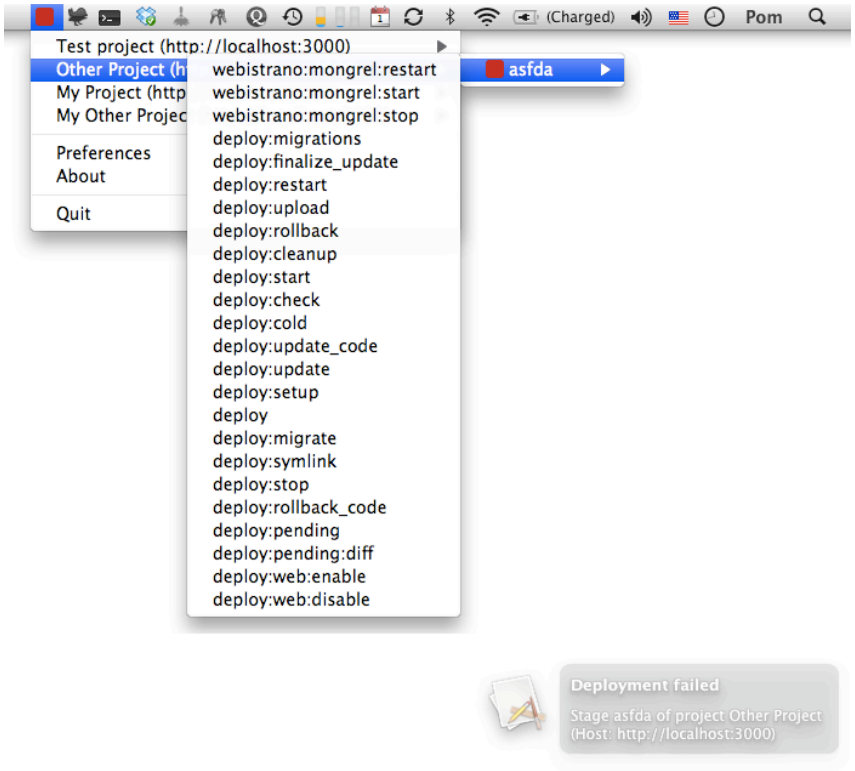


http://labs.peritor.com/webistrano

Macistrano

peritor consulting

- Mac-GUI to Webistrano
- Fire and monitor deployments from your desktop



http://github.com/mattmatt/macistrano

Pratical Session

peritor consulting

Practical Capistrano

Get the slides

- Connect to “RailsConf Deployment” WLAN
- The slides are at http://10.0.0.1/
- The VMs are at 10.0.0.3 – 10.0.0.41
- Sample configurations files are in the GIT/SVN repositories

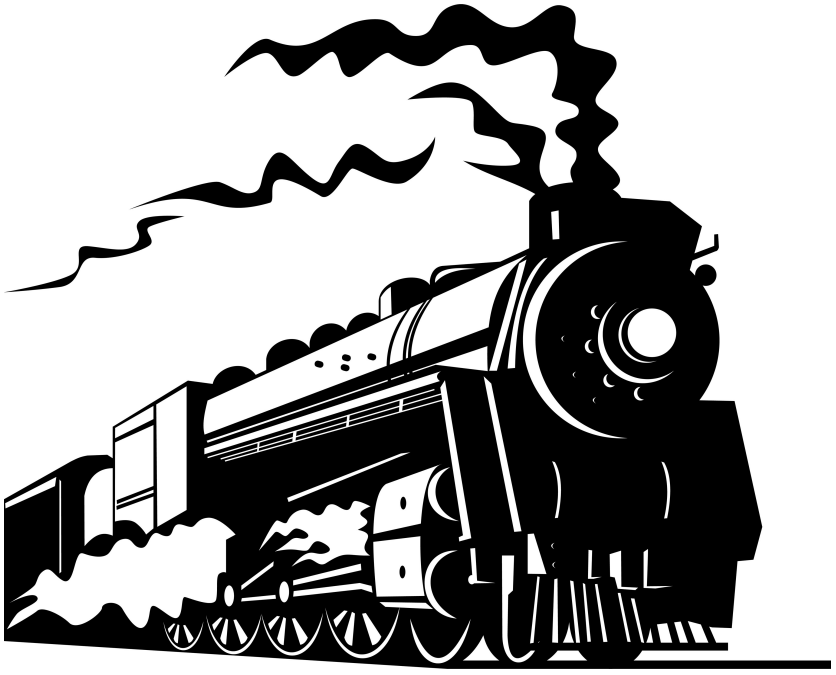
Monitoring

peritor consulting

The two questions of monitoring

peritor consulting

1. Is everything still running?



peritor consulting

2. What are the trends?

Monit

peritor consulting

- Process-level monitoring
- Checks PID-files, ports, and permissions
- Reacts by executing a script and/or alerting

Monit

peritor consulting

MySQL

```
# mysql
check process mysql with pidfile /var/db/mysql/master.peritor.com.pid
  group database
  start program = "/usr/local/etc/rc.d/mysql-server start"
  stop program = "/usr/local/etc/rc.d/mysql-server stop"
  if failed host localhost port 3306 protocol mysql then restart
  if 5 restarts within 5 cycles then timeout
```

Monit

peritor consulting

Apache

```
# apache
check process apache with pidfile /usr/local/apache/logs/httpd.pid
  start program = "/etc/init.d/httpd start"
  stop program  = "/etc/init.d/httpd stop"
  if cpu > 60% for 2 cycles then alert
  if cpu > 80% for 5 cycles then restart
  if totalmem > 200.0 MB for 5 cycles then restart
  if children > 250 then restart
  if loadavg(5min) greater than 10 for 8 cycles then stop
  if failed host www.tildeslash.com port 80 protocol http
    and request "/monit/doc/next.php"
    then restart
  if failed port 443 type tcpssl protocol http
    with timeout 15 seconds
    then restart
  if 3 restarts within 5 cycles then timeout
  depends on apache_bin
  group server
```

Monit

peritor consulting

Mongrel

```
# mongrel
check process mongrel-8000 with pidfile /home/user/rails/current/log/mongrel.8010.pid
  start program = "/usr/bin/mongrel_rails cluster::start -C /var/www/app/conf/mongrel.conf --only 8000"
  stop program  = "/usr/bin/mongrel_rails cluster::stop -C /var/www/app/conf/mongrel.conf --only 8000"

  if totalmem is greater than 60.0 MB for 5 cycles then restart       # eating up memory?
  if cpu is greater than 50% for 2 cycles then alert                  # send an email to admin
  if cpu is greater than 80% for 3 cycles then restart                # hung process?
  if loadavg(5min) greater than 10 for 8 cycles then restart          # bad, bad, bad
  if 3 restarts within 5 cycles then timeout                          # something is wrong, call the sys-admin

  if failed port 8000 protocol http                                   # check for response
    with timeout 10 seconds
    then restart
  group mongrel
```

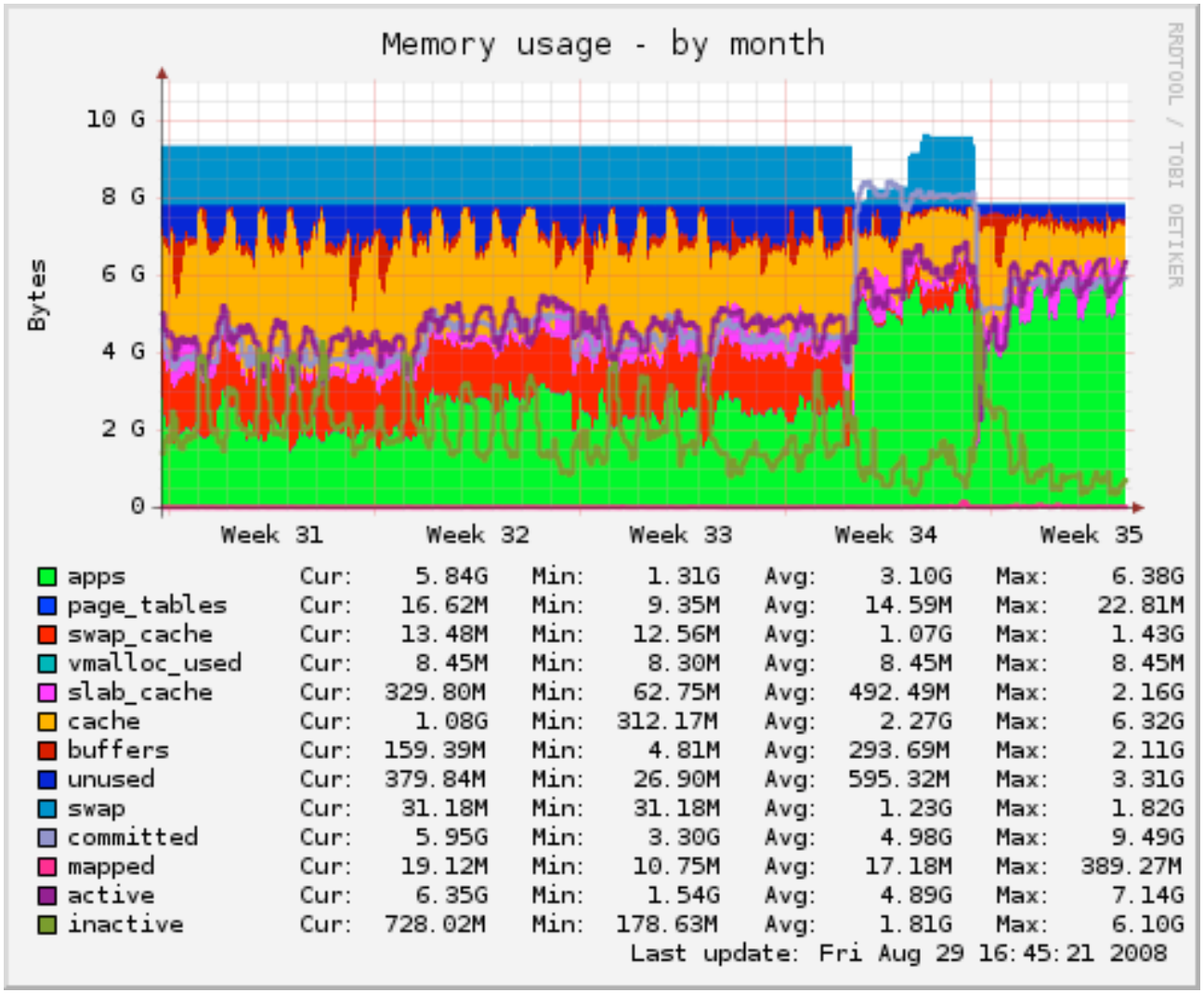
Munin

peritor consulting

- Host-level monitoring
- Master periodically asks nodes for local data
- Checks system resources and records historical data
- Allows to recognize trends and make predictions
- Alerting support

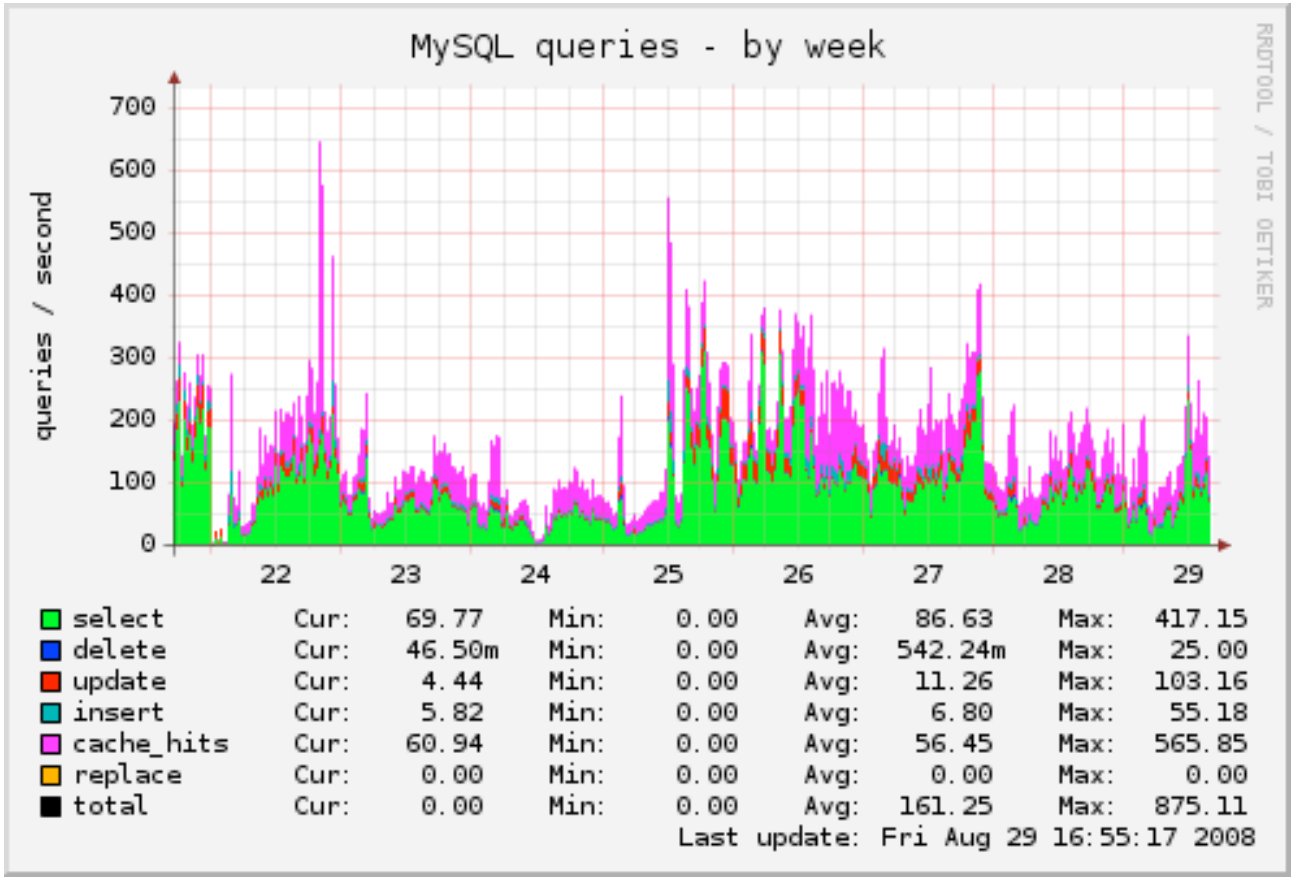
Munin

peritor consulting



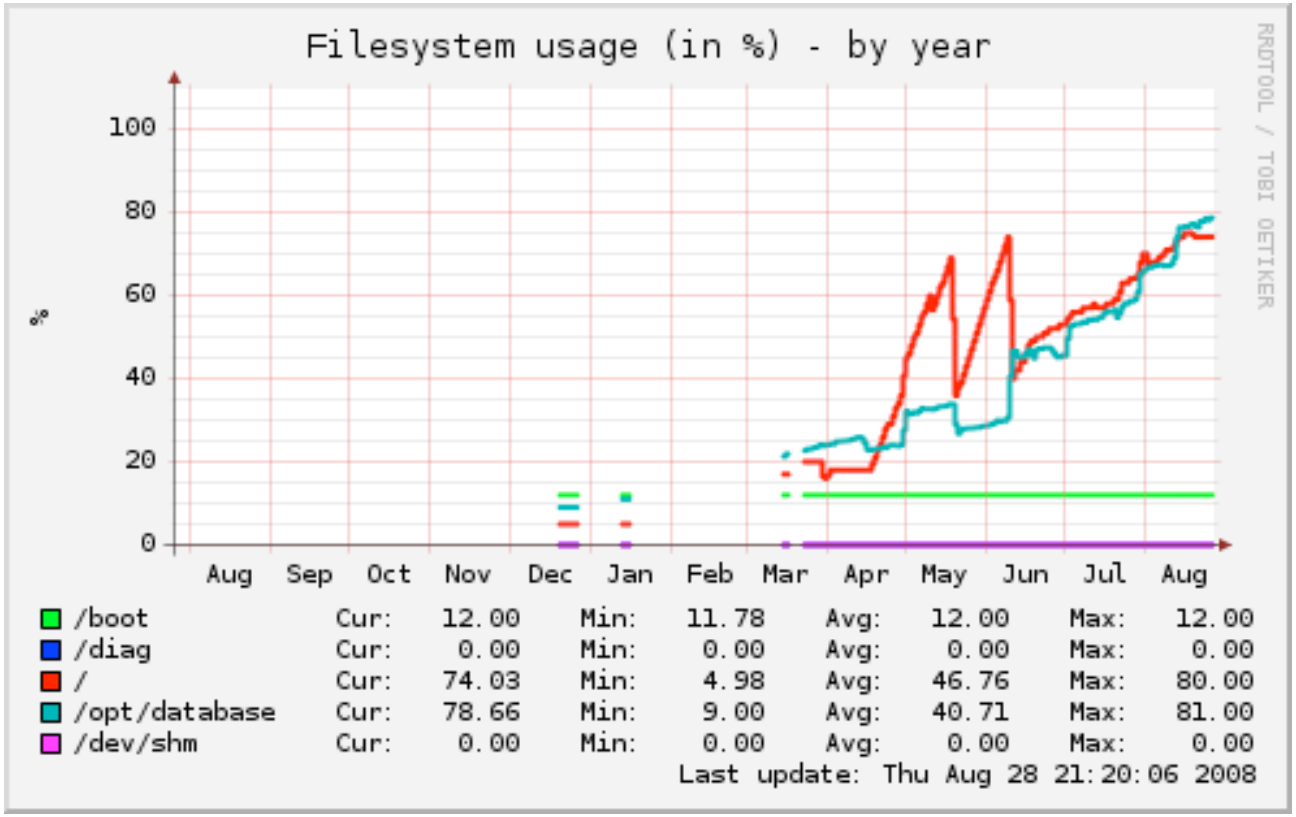
Munin

peritor consulting



Munin

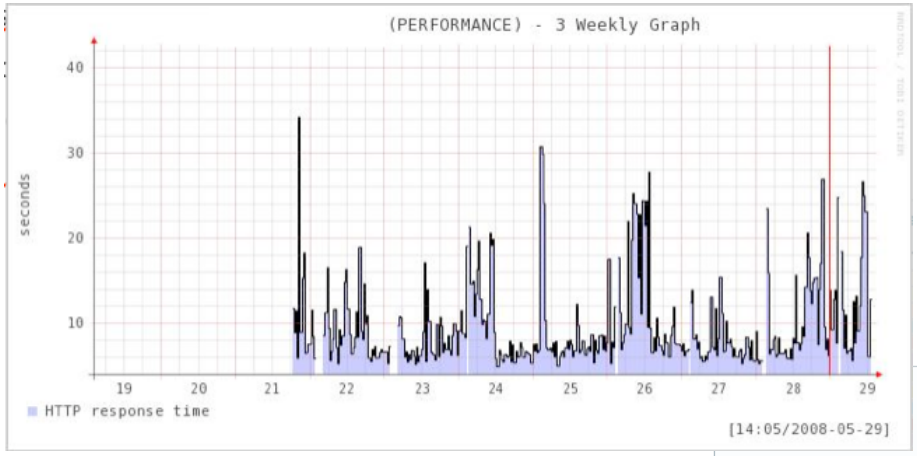
peritor consulting

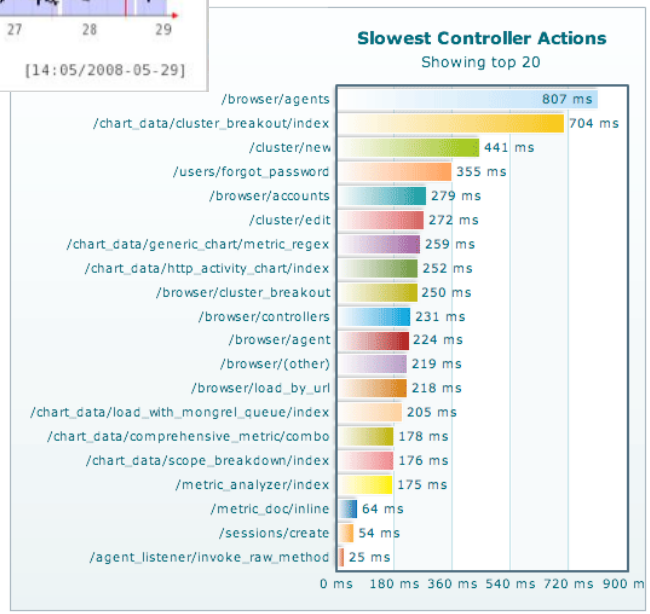


Other Tools

peritor consulting

- Nagios
- Big Brother
- New Relic RPM
- FiveRuns
- JMX





peritor consulting

Q & A

peritor consulting

Peritor GmbH

Teutonenstraße 16
14129 Berlin

Telefon: +49 (0)30 69 20 09 84 0
Telefax: +49 (0)30 69 20 09 84 9

Internet: www.peritor.com
E-Mail: kontakt@peritor.com