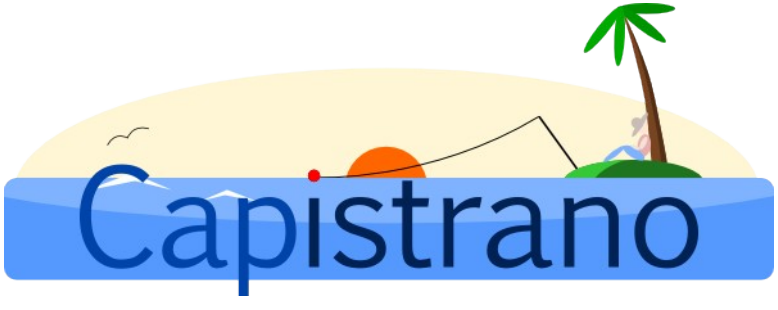
Capistrano Everywhere

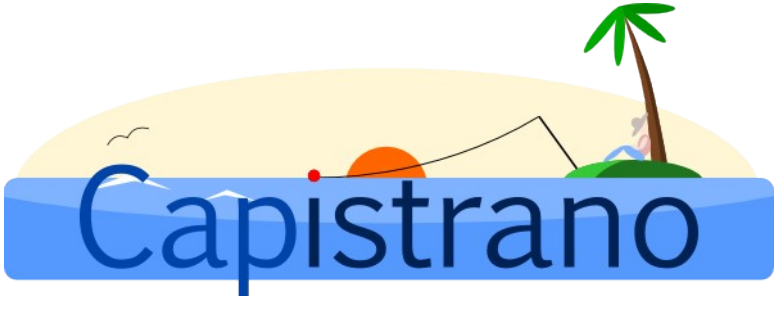
A brief overview of the possibilities

Kenneth Kalmer
www.opensourcery.co.za



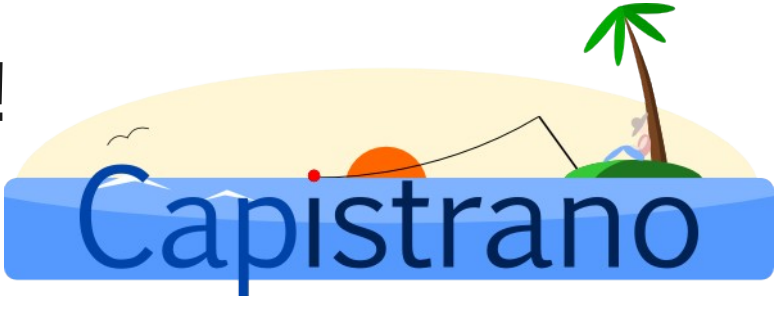
Capistrano is

- Great for deploying any application, not just Rails
- Great for probing multiple servers quickly
- Automating long tasks
- Good for your health
- On the fly maintenance
- Much much more...



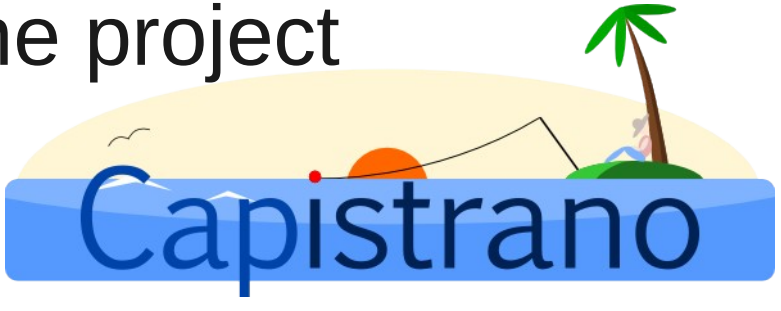
Problems with FTP

- Nice progress bars
- Uploading a local copy, not a clean revision from VCS
- Why FTP the same code to more than one server?
- How do you rollback a vanilla FTP upload?
- What if your connection timeouts in the middle of the upload?
- Everyone does it differently!



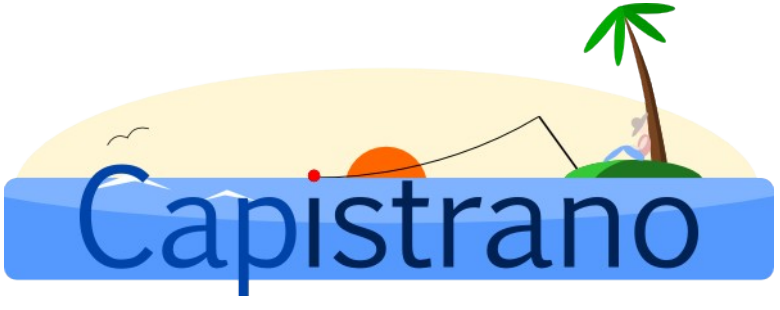
Successful Deployment Recipes

- Well defined process with consistent results
- Error handling and rollbacks
- Clear role definition (even on one server)
- Complimentary tasks
- Separation between application and user generated data (uploads, etc...)
- Leverage complimentary technologies (rake)
- Individual users deploy same project



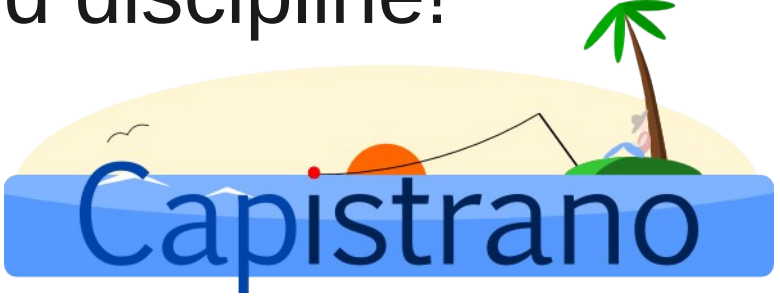
Rails Highlights

- Plugins can extend deployment recipes
- Easy multi stage deployments
- Remote console access
- Remote dependencies
  - Rubygems
  - Commands
- Run tests on deployment



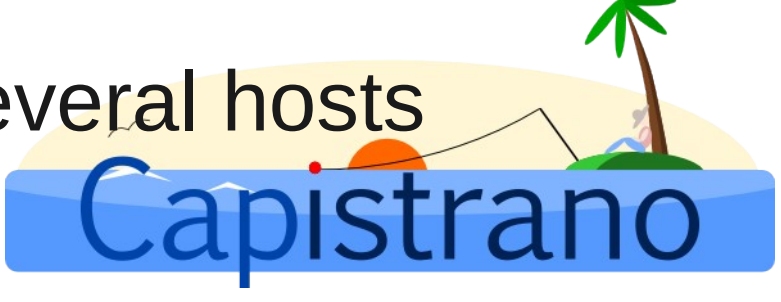
But I use PHP!

- Conceptually the same
- Update the DocumentRoot to serve out of the “current” symlink
- Update Options to have FollowSymlinks
- Add blank “deploy:restart”, “deploy:start” and “deploy:stop” tasks to Capfile
- Get a host that will give you shell access
- You gain better structure and discipline!
- *Switch to Ruby*



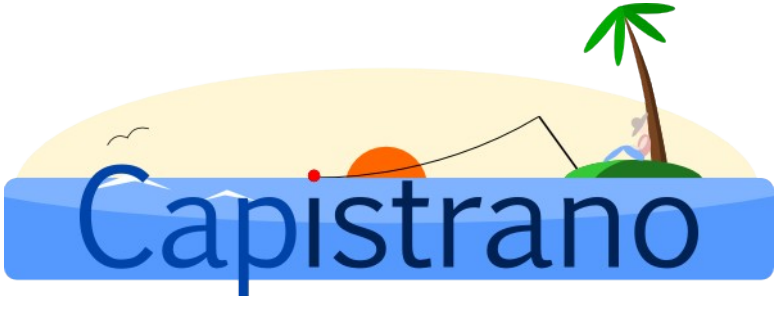
Custom Deployment Recipes

- “deploy” namespace provided by deploy.rb
- Can write custom deployment steps as long as:
  - Process is well defined and repeatable
  - Process is well defined and repeatable
  - Errors are handled
- Deploy Xen management scripts across several hosts
- Update backup scripts across several hosts
- Deploy SSH keys across several hosts



Complimentary Deployment Tasks

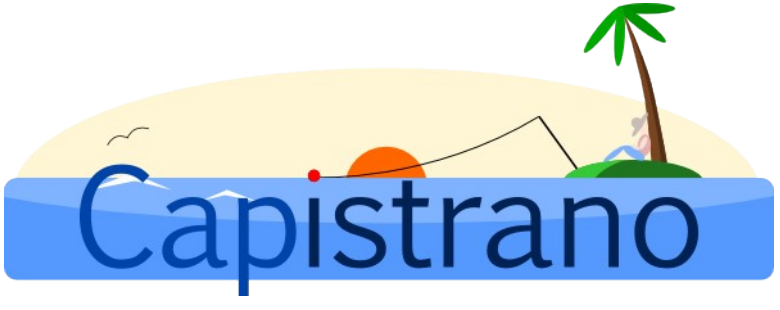
- Using rake to
  - Download copies of the production database
  - Download copies of the production logs
  - Minify assets
- Update external dependencies (local copy of Google Analytics JavaScript)
- Disable monit during deployment



System Administration

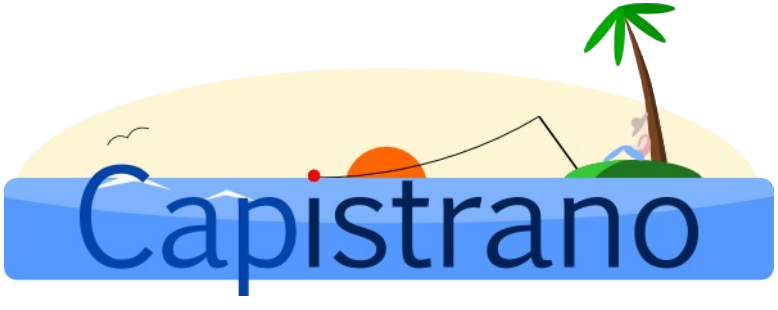
- Inspect the health of mail queues

```
$ cap mail:queues

  * executing `mail:queues'

  * executing "postqueue -p | tail -n 1"

    servers: ["192.168.2.15", "192.168.2.16", "192.168.2.17", "192.168.2.18",
   "192.168.2.14"]

# ADDITIONAL OUTPUT REMOVED

Mail queues on servers at 2008-05-29 00:38:10

192.168.2.15 -- 4354 Kbytes in 44 Requests.

192.168.2.16 -- 4635 Kbytes in 72 Requests.

192.168.2.14 -- 0 Kbytes in 1 Request.

192.168.2.17 -- 9079 Kbytes in 24 Requests.

192.168.2.18 -- 13168 Kbytes in 1 Request.
```



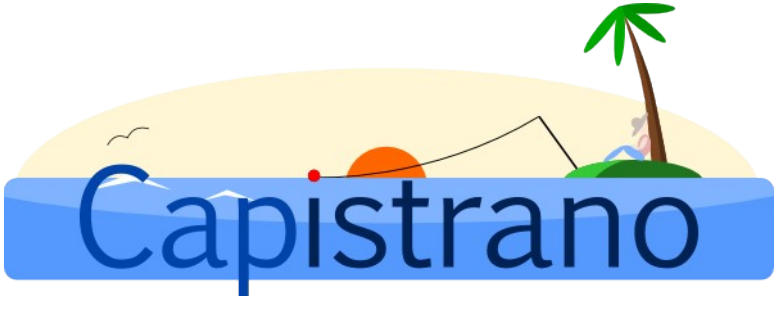
More for administrators

- Check uptime
- Check memory usage
- Check disk usage
- Search log files
- Leverage sudo
- On the fly audits
- Restart services

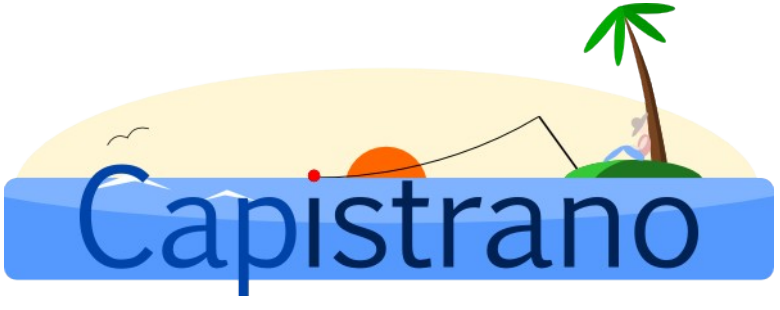


Things to remember

- It's a Linux server on the other side, think of Capistrano as an automated putty session to multiple hosts
- It is written in Ruby, so it can be bent to your will
- Default deployment recipes are not cast in stone, they just follow good principles
- It's not just for deployments, deployments was the birth of Capistrano



Q & A



Thank You

www.opensourcery.co.za