

Using Capistrano

Chirb — February 5, 2007

Presentation by Michael H. Buselli
cosine@cosine.org

<http://www.cosinewave.net/ruby/cap>

What is Capistrano?

- A tool to deploy applications remotely.
- Uses SSH to contact remote systems and issue commands in a POSIX shell.
- Also uses SFTP if available.
- Can tunnel SSH connections through a gateway host.
- Comes from Jamis Buck of 37signals, the same team that brings us Rails.
- Latest version is 1.4.0.

About Capistrano

- Like Rails, makes default assumptions about your environment to reduce configuration.
- If your environment is identical to 37signals's, you probably don't need a configuration file. [just kidding :)]
- Default assumes Rails, Subversion, Apache 1.x, FastCGI.
- Future versions will de-couple Rails support from the Capistrano core.

Getting Capistrano

- `sudo gem install capistrano -y`
- You should also make sure to pick up the following optional gems:
 - net-sftp
 - for reliable file transport
 - termios
 - so your password doesn't echo to the screen

Capistrano Commands

run	Execute a remote command
sudo	Same as run, but as root
put	Upload a file
delete	Remove a remote file
render	Return string from ERb template
get	Download a file (but only from one system)

run

```
run 'uname -a'
```

```
run "mv #{some_file} /path/to/somewhere"
```

- Give run a block to process output:

```
run "find /path" do |chan, strm, data|
```

```
  if strm == :err and data =~ /denied/
```

```
    logger.info "bad perm on #{chan[:host]}:
      #{data}"
```

```
  end
```

```
end
```

sudo

- Run commands as root with sudo:

```
sudo 'apachectl restart'
```

```
sudo "cp #{staging_area}/file /etc/file"
```

- Like run, sudo can also take a block for additional processing.

put

```
put "Hello World\n",  
    '/remote/path/hello_world'  
put File.read('/local/file'),  
    '/remote/file'
```


render

- Render a local ERb template file:

```
render "template.file"
```

- Render ERb template string:

```
render :template => "Hello <%= place %>",  
      :place => 'world'
```

- Use with put:

```
put render('config'),  
     "/path/to/#{app}/config"
```

delete

```
delete "/file/we/dont/like"
```

```
delete "/directory/we/dont/like",  
      :recursive => true
```

get

```
get "/remote/file/to/get",  
    "/where/to/put/it"
```

Add your own Commands!

```
module CapExtension
  def hello_world (greeting)
    run "echo hello #{greeting} world"
  end
end
```

```
Capistrano.plugin :ext, CapExtension
```

- Now call it from within a task:

```
ext.hello_world 'cruel'
```

Capistrano Tasks

- Commands are organized into tasks:

```
desc "Tell the world hello, then destroy  
it."
```

```
task :hello_world_destroy do  
  run 'echo "hello world" '  
  delete '/path/to/the/world'  
end
```

Capistrano Tasks

- Commands are organized into tasks:

```
desc "Tell the world hello, then destroy  
it."
```


```
task :hello_world_destroy do
```

```
  run 'echo "hello world"'
```

```
  delete '/path/to/the/world'
```

```
end
```

All systems run
this concurrently



Capistrano Tasks

- Commands are organized into tasks:

```
desc "Tell the world hello, then destroy  
it."
```

```
task :hello_world_destroy do
```

```
  run 'echo "hello world"'
```

```
  delete '/path/to/the/world'
```

```
end
```



Runs on success of
previous command
on all systems.

Tasks Calling Tasks

- Tasks often call other tasks:

```
task :hello_world do
  run 'echo "hello world"'
end
```

```
task :destroy_world do
  delete '/path/to/the/world'
end
```

```
task :hello_world_destroy do
  hello_world
  destroy_world
end
```

Task Hooks

- When a task, say *the_task*, is called Capistrano looks for other tasks called *before_the_task* and *after_the_task* and calls them before and after the *the_task* respectively.

```
task :before_hello_world do
  run 'echo "prepare to be greeted"'
end
```

```
task :after_hello_world do
  run 'echo "you have been greeted"'
end
```


Capistrano Roles

- Machines are grouped into roles that define what tasks apply to which systems.

```
role :web, 'www1', 'www2'
```

```
role :app, 'app1', 'app2'
```

```
role :db, 'db-master', :primary => true
```

```
role :db, 'db-slave1', 'db-slave2'
```

Limit Tasks by Role

- Each task can be limited to what role(s) it applies to.

```
task :hello_web_servers, :roles => :web do
  run 'echo "hello web server"'
end
```

```
task :hello_app_servers, :roles => :app do
  run 'echo "hello web server"'
end
```

```
task :hello_world do
  hello_web_servers
  hello_app_servers
end
```

Limit Tasks by Role

- Each task can be limited to what role(s) it applies to.

```
task :hello_web_servers, :roles => :web do
  run 'echo "hello web server"'
end
```

```
task :hello_app_servers, :roles => :app do
  run 'echo "hello web server"'
end
```

```
task :hello_world do
  hello_web_servers
  hello_app_servers
end
```

Runs only on
web servers

Limit Tasks by Role

- Each task can be limited to what role(s) it applies to.

```
task :hello_web_servers, :roles => :web do
  run 'echo "hello web server"'
end
```

```
task :hello_app_servers, :roles => :app do
  run 'echo "hello web server"'
end
```

```
task :hello_world do
  hello_web_servers
  hello_app_servers
end
```

Runs only on
app servers

Capistrano Variables

- Variables are used by various tasks and methods.

```
set :scm, :subversion
```

```
set :deploy_to, '/path/to/application'
```

- If a block is passed, it is evaluated on first use and result cached for further uses.

```
set :deploy_to do
```

```
  "/path/to/#{application}"
```

```
end
```

Capistrano with Rails

- In your RAILS_HOME run
`cap --apply-to . AppName`
- **Edit** `config/deploy.rb`
 - Configure mandatory variables
 - `:application`, `:repository`
 - Configure roles
 - `:web`, `:app`, `:db`, and one primary `:db`
 - Optional settings
 - `:scm`, `:deploy_to`, `ssh_options`, and many more

Run Capistrano

- Run Capistrano directly (preferred):

```
cap setup
```

- Run via Rake (deprecated):

```
rake remote:cold_deploy
```

- Capistrano sets up deploy and rollback targets directly in rake (deprecated):

```
rake deploy
```

- Run arbitrary tasks via Rake with remote:exec (deprecated):

```
rake remote:exec ACTION=my_task
```

Important Tasks for Rails

These some important Capistrano tasks for deploying a Rails application:

- `setup`
- `cold_deploy`
- `deploy`
- `rollback`
- `cleanup`

Run `cap show_tasks` or `rake --tasks` to see more of what's available.

setup

Creates initial directories in the deployment area.

```
% cap setup
* executing task setup
* executing "umask 02 &&\n      mkdir -p
/u/apps/myapp /u/apps/myapp/releases
/u/apps/myapp/shared /u/apps/myapp/shared/system
&&\n      mkdir -p /u/apps/myapp/shared/log &&\n
mkdir -p /u/apps/myapp/shared/pids"
servers: ["localhost"]
[localhost] executing command
command finished
```

cold_deploy

Deploys application and starts FastCGI processes by calling script/spin

- Make sure you create script/spin:

```
#!/bin/sh
/u/apps/myapp/current/scripts/process/spinner -c
'/u/apps/myapp/current/scripts/process/spawner
-p 7000 -i 5' -d
```

- Note use of spinner and spawner to keep the FastCGI processes alive.

deploy

- Deploy your application to your servers.
- Takes care of copying files out of Subversion (or other SCM) and to the deployment area, then restart necessary processes.

```
% cap deploy 2>&1 | grep 'executing task'  
* executing task deploy  
* executing task update  
* executing task update_code  
* executing task set_permissions  
* executing task symlink  
* executing task restart
```

Deployment Path

`/u/apps/myapp/`

- `current -> releases/20070205225159`
- `releases/`
 - `20070204080051/`
 - `20070205060519/`
 - `20070205225159/`
 - **This directory is RAILS_ROOT.**
- `revisions.log`
- `shared/`
 - `log/`
 - `pids/`
 - `system/`

Deployment Path

`/u/apps/myapp/`

- `releases/`
 - `20070205225159/`
 - `app/`
 - `config/`
 - `db/`
 - `script/`
 - `public/`
 - `system -> /u/apps/myapp/shared/system`
 - `log -> /u/apps/myapp/shared/log`
 - `tmp/`
 - `pids -> /u/apps/myapp/shared/pids`

rollback

Oops! We screwed up. Retreat!

```
% cap rollback
```

- Re-links current to previous deployed release.
- Deletes the release rolled back from.

cleanup

`% cap cleanup`

- Deletes old releases, only retaining the previous five.

Resources

- Capistrano Manual:

<http://manuals.rubyonrails.com/read/book/17>

- Capistrano Mailing List:

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

- LRUG Presentation by Tom Ward:

http://skillsmatter.com/downloads/Tom_Ward_Doing_Bad_Things_With_Capistrano.pdf

Resources

- Capistrano Screencast (\$9):

<http://peepcode.com/products/capistrano-concepts>

- O'Reilly Online Book (requires subscription):

<http://safari.oreilly.com/0596529627>

- My page on this presentation:

<http://www.cosinewave.net/ruby/cap>

Using Capistrano

Chirb — February 5, 2007

Presentation by Michael H. Buselli
cosine@cosine.org

<http://www.cosinewave.net/ruby/cap>