

Commanding Your SSH Universe with Capistrano

Ryan Carmelo Briones
Server Monkey / Code Samurai, **Edgecase, LLC**



<http://brionesandco.com/ryanbriones>

<http://theedgecase.com/blog>

<http://blogs.theedgecase.com>

04:45:21 PM



04:45:21 PM



04:45:21 PM



04:45:21 PM

DEPLOY APPLICATION

Update Code to Latest

**DEPLOY
APPLICATION**

Update Code to Latest

DEPLOY APPLICATION

Update Schema

Update Code to Latest

Dependancies

DEPLOY
APPLICATION

Update Schema

Update Code to Latest

Dependancies

DEPLOY
APPLICATION

Restart Application

Servers

Update Schema

Update Code to Latest

Dependancies

Restart
Web APPLICATION
Server

Restart Application
Servers

Update Schema

\$ cd .

\$ cd ..

\$ c d ..

\$ cd ..



1
Position arm
cuff as shown
in diagram.



2
Press start button
and remain quiet.
Readings will appear
in less than a minute.



To stop test
at any time,
press STOP.



Systolic
Pressure
mmHg

166

Diastolic
Pressure
mmHg

104

Pulse
Rate
BPM

60

Test
BL
PRE

General Reference Chart
For Adults

Systolic	Diastolic
Normal Below 130	Normal Below 85
High Normal 130 - 139	High Normal 85 - 89
High 140 - 159	High 90 - 99

TESTING
DON'T MOVE

TEST
COMPLETE

Use of the Vita-Stat
a physician. ONLY A
INTERPRET THE SI
PRESSURE MEASU
OR SELF ADJUSTM
DANGEROUS.

Blood pressure norm
changes can range
within a few minutes
REPEATED MEASUR
TIME IS THE BEST IN
PRESSURE.

The cuff is designed



Capistrano

Capistrano

Can automate any* SSH task

Application Deployment

Software Installation

Ad-hoc Monitoring

with Parallel Execution



Caveat Emptor

Multiple Passwords

Caveat Emptor

Multiple Passwords

Public Key Identification to the Rescue

Caveat Emptor

POSIX targets only
Sorry No Windows Server

Caveat Emptor

POSIX targets only
Sorry No Windows Server

Unless

Caveat Emptor

POSIX targets only
Sorry No Windows Server

Unless
You want to use Cygwin

The Basics

Installation

```
$ gem install capistrano -y  
# -y is included by default in RubyGems >1.0  
$ which cap  
/usr/bin/cap  
$ which capify  
/usr/bin/capify
```

Usage

```
$ ssh admin@myserver.mydomain.com
Password:
admin@myserver$ df -h
[output disk space]
admin@myserver$ cd /path/to/files
admin@myserver:/path/to/files$ du -h
[output folder usage]
admin@myserver$ exit
$
```

```
$ cap free_space myfiles_usage
Password:
* [out :: myserver.mydomain.com] [output disk space]
* [out :: myserver.mydomain.com] [output folder usage]
$
```

Configuration using Ruby DSL

```
# Capfile
task :freespace do
  run 'df -h'
end

task :myfiles_usage do
  run 'cd /path/to/files; du -h'
end
```

Available Tasks

```
$ cap -T  
cap invoke # Invoke a single command on the remote servers.  
cap shell  # Begin an interactive Capistrano session.  
$
```

Available Tasks

```
$ cap -T
cap invoke # Invoke a single command on the remote servers.
cap shell  # Begin an interactive Capistrano session.
$
```

Hidden Tasks

```
dumont:Servers ryanbriones$ cap -Tv
cap freespace      #
cap invoke         # Invoke a single command on the remote
servers.
cap myfiles_usage #
cap shell          # Begin an interactive Capistrano session.
$
```

Describe what's Important

```
desc 'Lookup freespace'  
task :freespace do  
  contrived_example  
  run 'df -h'  
end
```

```
desc 'See disk usage in myfiles. This text only appears with -e'  
task :myfiles_usage do  
  run 'cd /path/to/files; du -h'  
end
```

```
task :contrived_example do  
  puts 'running disk freespace'  
end
```

Describe what's Important

```
$ cap -T
cap freespace      # Lookup freespace
cap invoke         # Invoke a single command on the remote...
cap myfiles_usage # See disk usage in myfiles
cap shell         # Begin an interactive Capistrano session.
$ cap -Tv
cap freespace      # Lookup freespace
cap invoke         # Invoke a single command on the remote...
cap myfiles_usage # See disk usage in myfiles.
cap shell         # Begin an interactive Capistrano session.
cap contrived_example #
$
```

Describe what's Important

```
$ cap -e myfiles_usage
```

```
-----  
cap myfiles_usage  
-----
```

```
See disk usage in myfiles. This text only appears with -e
```

```
$
```

Roles and Tasks in Parallel

```
role :web, 'web.mydomain.com'  
role :app, 'app1.mydomain.com', 'app2.mydomain.com'  
  
task :some_task, :role => :web do  
  # runs only on web server  
end  
  
task :some_other_task, :role => :app do  
  # runs on both app servers in parallel  
end  
  
task :global_task do  
  # runs on all three servers in parallel  
end
```

Shared Global Configuration

```
set :user, 'admin'  
set :myfiles, '/path/to/myfiles'  
  
task :myfile_usage do  
  run "cd #{myfiles}; du -h"  
end
```

Namespacing

```
namespace :sys do
  task :freespace do
    run 'df -h'
  end
end

namespace :mycompany do
  task :deploy do
    # do my own thing
  end
end
```

```
$ cap -Tv
cap mycompany:deploy #
cap sys:freespace    #
$
```

Events

```
# cap 1.x
task :after_update_code do
  # this happens after the deploy task
end
```

```
# cap 2.x
after 'deploy:update_code', :my_well_named_and_descriptive_task
task :my_well_named_and_descriptive_task do
  # do stuff here
end
```

```
after 'deploy:update_code' do
  # do stuff here
end
```

Deployment

A typical deployment story

1. capify .
 2. cap deploy:setup
 3. cap deploy:check
 4. cap deploy:cold
 5. cap deploy
- cap deploy:migrations
 - cap deploy:rollback
 - cap deploy:cleanup

A typical deployment story

1. `capify` .

2. `cap deploy:setup`

3. `cap deploy:check`

4. `cap deploy:cold`

5. `cap deploy`

- `cap deploy:migrations`
- `cap deploy:rollback`
- `cap deploy:cleanup`

Deployment

Basic Setup

```
set :application, 'my_app_name'  
  
role :app, 'myserver.com'  
  
# deploy_to default: "/u/apps/#{application}"  
set :deploy_to, "/path/to/deploy/to/#{application}"  
set :user, 'deployusername'
```

A typical deployment story

1. capify .

2. cap deploy:setup

3. cap deploy:check

4. cap deploy:cold

5. cap deploy

- cap deploy:migrations
- cap deploy:rollback
- cap deploy:cleanup

Deployment Setup

`/deploy_to/path`

shared

releases

A typical deployment story

1. capify .
 2. cap deploy:setup
 3. cap deploy:check
 4. cap deploy:cold
 5. cap deploy
- cap deploy:migrations
 - cap deploy:rollback
 - cap deploy:cleanup

Deploy Dependencies

```
# config/deploy.rb
depend :remote, :command, 'custom_command'
depend :local, :command, 'custom_command'

# Version is required
depend :remote, :gem, :rcov, '>=0.8.0.2'

# Directory exists?
depend :remote, :directory, '/path/to/some/files'

# File is writable?
depend :remote, :writable, '/path/to/logs/production.log'

# Command outputs what's expected
depend :remote, :match, 'some_command ARGS', /expected/
```

A typical deployment story

1. capify .
 2. cap deploy:setup
 3. cap deploy:check
 4. cap deploy:cold
 5. cap deploy
- cap deploy:migrations
 - cap deploy:rollback
 - cap deploy:cleanup

Deployment Setup

/deploy_to/path

current

shared

releases

20080715135531

20080715150757

20080716160011

20080717150428

20080718145759



A typical deployment story

1. capify .
 2. cap deploy:setup
 3. cap deploy:check
 4. cap deploy:cold
 5. cap deploy
- cap deploy:migrations
 - cap deploy:rollback
 - cap deploy:cleanup

Deployment Repositories

```
set :scm, :subversion
set :repository, 'url to your repository'
set :scm_username, 'username'
set :scm_password, 'password'
```

AccuRev

Bazaar

CVS

darcs

Git

Mercurial

Perforce

Subversion

Deployment Repositories

```
set :scm, :subversion  
set :repository, 'url to your repository'  
set :scm_username, 'username'  
set :scm_password, 'password'
```

AccuRev

Bazaar

CVS

darcs

Git

Mercurial

Perforce

Subversion

None

Deployment Strategies

- **Checkout (Default)** - remotely checked out for each release
- **Export** - remotely exported for each release
- **Copy** - prepared locally, compressed, SFTP, decompressed to release directory
- **Remote Cache** - checked out ONCE remotely, copied to release directory
- **SCM None** - sftp local directory to release directory

“Advanced” Capistrano

```
STAGES = %w(staging production uat)
STAGES.each do |name|
  desc "Set the target stage to '#{name}'."
  task(name) do
    set :stage, name.to_sym
    load "config/deploy/#{stage}"
  end
end

on :start, :except => STAGES do
  if !exists?(:stage)
    abort "no stage specified, please choose one of #{STAGES.join(", ")}"
  end
end

$ cap staging deploy
$ cap production deploy
```

```
require 'capistrano/ext'

set :stages, %w(staging production uat)
set :stage_dir, 'config/deploy'
```

```
def remote_file_exists?(remote_file)
  run "if [ -f #{remote_file} ]; then echo 1; fi" do |channel, stream, data|
    data.to_i.nonzero? == data.to_i
  end
end

task :ensure_config_file do
  unless remote_file_exists?("#{shared_path}/config_file")
    run "cp #{release_path}/config_file.#{stage} #{shared_path}/config_file"
  end
  run "ln -nfs #{shared_path}/config_file #{release_path}/config_file"
end
```

```
set(:prompted_value) do
  Capistrano::CLI.ui.ask("What do you want to do today?: ")
end

task :use_prompted_value do
  puts "I want to: #{prompted_value}"
end
```

Questions?