Capistrano

Deploy Magento Project & Others...

By Sylvain Rayé

Zürich Mage Hackathon - 10.03.2013

EGO PAGE ;-)

Sylvain Rayé

- CTO & Senior Web Developer for rissip GmbH
- Computer Engineer Information Systems
- Certified Magento Developer

http://www.sylvainraye.com
http://www.diglin.com
@sylvainraye
@diglin_com

The basics

Capistrano

- Deployment framework for web applications
  - Ruby On Rails applications
  - PHP applications (need to overwrite few core tasks)
- Purpose
  - Quick code deployment (full or partial)
  - Recurring process/tasks
  - Multi Stages
  - Multi Servers
  - Execute command on server (local or remote)
- NOT
  - Completely ready to use

Definitions 1/2

- Task
  - Execute a set of command(s) on server e.g. cap deploy

```
desc "Dummy task"
task :dummy do
    puts 'I am a dummy task!!!'
end
```

- Role
  - Multi Servers role: DB1, DB2, Web Server, LDAP, ...
  - Roles are applied to a task (default is all roles)

```
roles :web, "192.168.0.1", "www.domain.com"'
roles :db, "192.168.0.2", :primary => true

desc "I run a db command on the db server"
task :dummy, :roles => :db do
    run "mysql -u dummy -pdummy -e 'SHOW DATABASES;' "
end
```

Definitions 2/2

- **Namespace**
  - Basic: "deploy:" - Custom: e.g. "tools:", "config:"

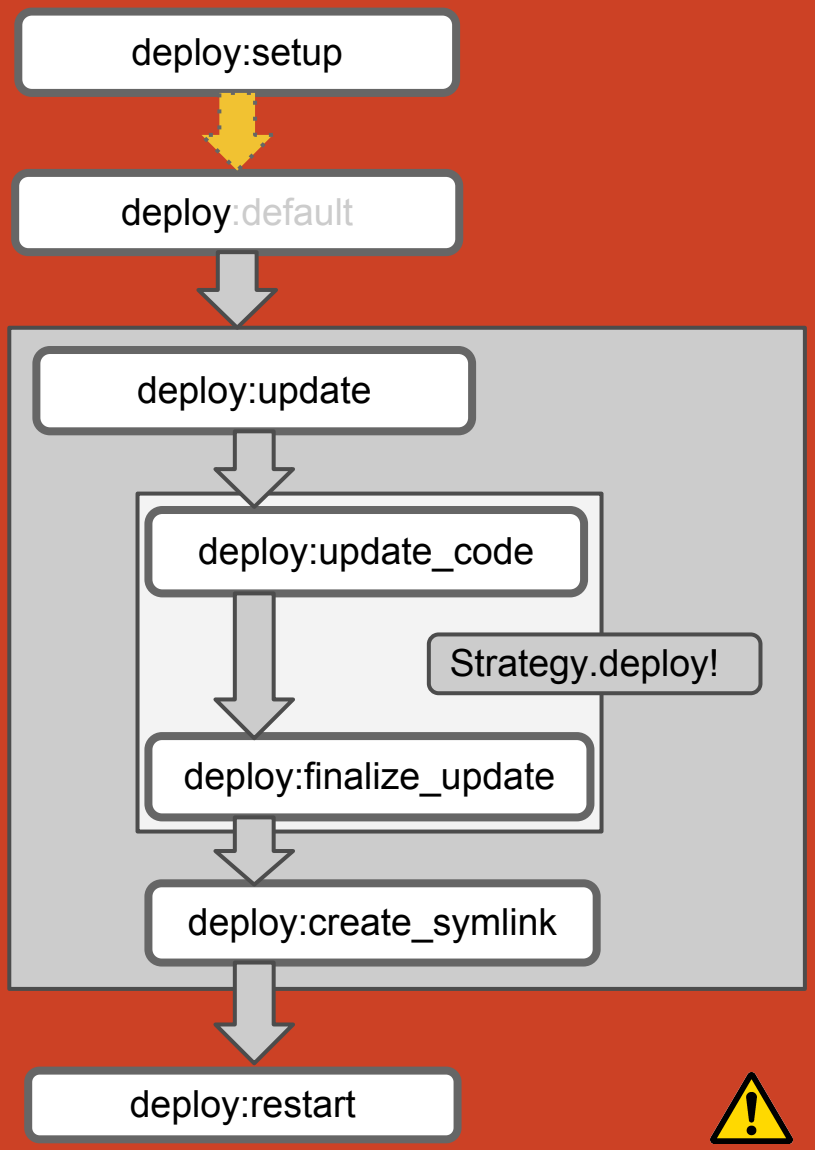
```
namespace :tools do
    desc "Backup task"
    task :backup do
       puts 'I should do a backup but I am a lazy boy!!!'
    end
end
```

- **Hook**
  - before "anytask" - after "anytask"

```
before "deploy", "tools:backup"
after "deploy" do
    puts "Bravo! You deployed your code!!!"
end
```

- **Recipe**
  - Bundle of tasks gathered in a file. e.g. deploy.rb

DISPATCH BEHAVIOR



1rst deployment prepare setup on the server(s)

Deploys your project

Deploys your project and create symlink

Copies your project to the remote server

deploy_via (local: copy, remote: checkout, export, remote_cache)

[internal] Touches up the released code

Updates the symlink to the most recent release

Do nothing - To overwrite if needed

Source: https://github.com/capistrano/capistrano/wiki/2.x-Default-Deployment-Behaviour

Remote deployment structure

[deploy_to] (e.g. /www-data/hosts/www.domain.com/capiroot/apps/magento)
[deploy_to]/releases
[deploy_to]/releases/20130310001122
[deploy_to]/releases/20130105000010
[deploy_to]/releases/...
[deploy_to]/shared
[deploy_to]/shared/media
[deploy_to]/shared/var
[deploy_to]/shared/...
[deploy_to]/current -> [deploy_to]/releases/20130310001122

Installation & Use

Installation

- Requirements:
  - Version Source Control (git, svn, ...) - Local: Ruby, Gem Server: executed bin, same user/pass on all server, SSH
- Installation
  - gem install capistrano
  - Binaries: capify and cap
- Init a project
  - cd myproject && capify .
  - 2 files created
    - Capfile: bootstrap
    - config/deploy.rb: recipe, config, hooks, ...

Structure of Local Project

```
|_MyProject
   |__Capfile                  # Capistrano bootstrap
   |__config
   |    |__deploy.rb           # Recipe, global config
   |__app
   |__cron.php
   |__index.php
   |__media (set it as shared on remote server)
   |__RELEASE_NOTES.txt
   |__shell
   |__skin
   |__js
   |__var (set it as shared on remote server)
   |__...
```

Default Capile / deploy.rb files

- Capfile

```
load 'deploy'
# Uncomment if you are using Rails' asset pipeline
    # load 'deploy/assets'
load 'config/deploy' # remove this line to skip loading any of the default tasks
```

- config/deploy.rb

```
require 'capistrano/ext/multistage'

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

# set :scm, :git # You can set :scm explicitly or Capistrano will make an intelligent guess based on known version control directory names
# Or: `accurev`, `bzr`, `cvs`, `darcs`, `git`, `mercurial`, `perforce`, `subversion` or `none`

role :web, "your web-server here"                          # Your HTTP server, Apache/etc
role :app, "your app-server here"                          # This may be the same as your `Web` server
role :db,  "your primary db-server here", :primary => true # This is where Rails migrations will run
role :db,  "your slave db-server here"
```

Default Tasks

```
> cd myproject
> cap -vT
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:create_symlink   # Updates the symlink to the most recently deployed version.
cap deploy:finalize_update  # [internal] Touches up the released code.
cap deploy:pending          # Displays the commits since your last deploy.
cap deploy:pending:diff     # Displays the `diff' since your last deploy.
cap deploy:restart          # Blank task exists as a hook into which to install your own environment
cap deploy:rollback         # Rolls back to a previous version and restarts.
cap deploy:rollback:cleanup # [internal] Removes the most recently deployed release.
cap deploy:rollback:code    # Rolls back to the previously deployed version.
cap deploy:rollback:revision # [internal] Points the current symlink at the previous revision.
cap deploy:setup            # Prepares one or more servers for deployment.
cap deploy:start            # Blank task exists as a hook into which to install your own environment
cap deploy:stop             # Blank task exists as a hook into which to install your own environment
cap deploy:update           # Copies your project and updates the symlink.
cap deploy:update_code      # Copies your project to the remote servers.
cap deploy:upload           # Copy files to the currently deployed version.
cap invoke                  # Invoke a single command on the remote servers.
cap shell                   # Begin an interactive Capistrano session.
```

Tasks to overwrite for non Rails

- deploy:start, deploy:stop, deploy:restart
  - Set them to an empty task (Capistrano v 1.x)
- deploy:migrate, deploy:migrations
  - Set them to an empty task
- deploy:default, deploy:cold
  - Not mandatory: do a redirect to task deploy:update
- deploy:finalize_update
  - e.g. files and folder fix permissions

- Sample code at http://bit.ly/13yukoP

Configure &
Create your own task(s)

Configure

- SSH Authentication with keys
  - Otherwise, after each 'cap', password will be asked !

```
> ssh-keygen -t rsa
> cat ~/.ssh/id_dsa.pub | ssh sylvain@staging.magehackathon.local "cat - >>.ssh/authorized_keys"
```

- Deploy.rb

```
set :http_domain,  'staging.magehackathon.local' # local variable
set :location, http_domain
set :user, "sylvain" # SSH user
# set :scm_username, “foo”
# set :use_sudo, false
# set :ssh_options, { :forward_agent => true, :port: 22 }
default_run_options[:pty] = true
set :application, "capistranosimple"
set  :keep_releases,  5
set :repository, "adress of my repo" # Git remote repo for example
set :scm, :git  # other version control system are also available: e.g. svn
set :branch, "master"
set :deploy_via, :remote_cache # possible values: copy, checkout, remote_cache, export
set :deploy_to, "/www-data/hosts/#{http_domain}/capiroot/apps/#{application}"
role :web, location                       # Your web server domain or IP. e.g. Apache or nginx
role :db,  location, :primary => true
```

Create your own code

- VERY Basic knowledge in Ruby
  - To Ruby from PHP: http://bit.ly/Z7y3mn
- Define variable

```
set :application, "capistranomagento"
set :keep_releases,  5
set :app_shared_dirs, ["/app/etc", "/media", "/var"]
```

- Create methods, tasks

```
def remote_file_exists(full_path)
   'true' ==  capture("if [ -e #{full_path} ]; then echo 'true'; fi").strip
end
namespace :tools do
   desc "Import sample data with specific information to database. Executed after deploy:setup"
   task :import_sample_data, :roles => [:db], :only => { :primary => true } do
      ....
      abort  "#{localsamplesqlfile} doesnt exist locally" unless File.exists?( localsamplesqlfile)
      if !remote_file_exists("#{shared_path}/shell/#{samplesqlfile}") then
         run "mkdir -p #{shared_path}/shell"
         upload localsamplesqlfile, "#{shared_path}/shell/#{samplesqlfile}"
      end
   end
end
```

Basic Capistrano Action

- run: execute commands on one or more servers
- parallel: execute multiple commands on multiple servers in parallel
- put: store the contents of a file on multiple servers
- get: transfers a file from a single remote server to the local host
- upload: transfers a file or directory from the local host to multiple remote hosts, in parallel.
- download: transfers a file or directory from multiple remote hosts to the local host, in parallel.
- capture: executes a command on a single host and returns ("captures") the output as a string
- stream: very similar to run, but optimized for displaying live streams of text (like tailed log files) from multiple hosts

Simple Demo
One stage

see code at http://bit.ly/ZS0iYI

Multi Stages

Multi Stages - 1/2

- Without Capistrano Extension
  - A task per stage in config/deploy.rb

```
task :production do
  role :web, "www.capify.org"
  set :deploy_to, "/u/apps/#{application}-production/"
  set :deploy_via, :remote_cache
  after('deploy:create_symlink', 'cache:clear')
end

task :staging do
  role :web, "localhost"
  set :deploy_to, "/Users/capistrano/Sites/#{application}-staging/"
  set :deploy_via, :copy
  after('deploy:create_symlink', 'cache:clear', 'mage:enable_profiler')
end
```

```
> cap <stagename> <taskname> <optional_parameter>
```

```
> cap staging deploy
```

Multi stages - 2/2

- With Capistrano Extension (Preferred)
  - A ruby file per stage
    - config/deploy/local.rb - http://bit.ly/161gZ82
    - config/deploy/staging.rb - http://bit.ly/ZdSnDR
    - config/deploy/production.rb - http://bit.ly/ZS8rft
  - Settings in config/deploy.rb

```
require 'capistrano/ext/multistage'
set :stages, %w(local production staging)
set :default_stage, "staging"
...
```

```
> cap <stagename> <taskname> <optional_parameter>
> cap staging deploy:upload FILES=index.php,app/code/local
```

Structure of Local Project

```
|_MyProject
   |__Capfile                  # Capistrano bootstrap
   |__config
   |    |__deploy.rb           # Recipes, global config
   |    |__deploy              # In case of multi stages
   |         |__local.rb       # Config, Hook specific to local
   |         |__staging.rb     # Config, Hook specific to staging
   |         |__production.rb  # Config, Hook specific to production
   |__index.php
   |__skin
   |__media (shared)
   |__app
   |__cron.php
   |__js
   |__var (shared)
   |__RELEASE_NOTES.txt
   |__shell
   |__...
```

Simple Demo
Two stages

see code at http://bit.ly/ZS0iYI

Magentify

Magentify a project

- Install Magentify
  - gem install magentify
- Init a project
  - Same as with Capify
  - > cd myproject && magentify .
- Advantage
  - Some predefined behaviors and tasks

```
cap mage:cc                    # Clear the Magento Cache
cap mage:clean_log             # Clean the Magento logs
cap mage:compiler              # Run the Magento compiler
cap mage:disable               # Disable the Magento install by creating the maintenance.flag in the web root.
cap mage:disable_compiler      # Disable the Magento compiler
cap mage:enable                # Enable the Magento stores by removing the maintenance.flag in the web root.
cap mage:enable_compiler       # Enable the Magento compiler
cap mage:finalize_update       # Touches up the released code.
cap mage:indexer               # Run the Magento indexer
cap mage:setup                 # Prepares one or more servers for deployment of Magento.
```

Magento Demo
Two stages

see code at http://bit.ly/ZS0iYI

Links

Links

- This presentation and its code
  - http://bit.ly/ZSOiYI
- Capistrano
  - http://capistranorb.com
- Magentify
  - http://bit.ly/ZZSyYa
- PHP deployment with Capistrano
  - http://bit.ly/Z7y3mn
  - http://bit.ly/MzqH9o
- Capistrano GUI
  - https://github.com/joelmoss/strano

THANKS !!!