Extracted from:

Deploying Rails

Automate, Deploy, Scale, Maintain, and Sleep at Night

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers

Deploying Rails

Automate, Deploy, Scale, Maintain, and Sleep at Night

Anthony Burns and Tom Copeland

Edited by Brian P. Hogan

5.1 Deploying Faster by Creating Symlinks in Bulk

Sometimes we'll see variables defined in block form; here's an example:

```
set(:deploy_to) { "/var/massiveapp/deploy/#{name}" }
```

Blocks are used in Capistrano for variables whose values must be lazy-loaded. They become especially useful when you must interpolate the value of another Capistrano variable and as a rule of thumb should always be used whenever a reference to another Capistrano variable is made.

Let's look at a more complex use of block variables and speed up our deploys at the same time. As part of the default deployment process, Capistrano symlinks certain directories in the #{deploy_to}/shared directory after updating the application code. By default, each of these symlinks is created using a separate call to run, which in turn creates a separate SSH connection to make each symlink. Establishing and tearing down these SSH connections can take some time. Using Capistrano's block variables, though, we can replace the symlink task with one that gives us more speed, as well as additional flexibility.

Here's a set of variables assignments that show all the directories that need to be cleaned up, created, or removed during a deploy. There are a lot, so if we can avoid making a connection for each, our deploys will be much faster.

capistrano2/sample_variables.rb

```
set :cleanup_targets, %w(log public/system tmp)
set :release_directories, %w(log tmp)
set :release_symlinks do
  {
    "config/settings/#{stage}.yml" => 'config/settings.yml',
    "config/database/#{stage}.yml" => 'config/memcached.yml',
  }
end
set :shared_symlinks, {
  'log'     => 'log',
  'pids'    => 'tmp/pids',
  'sockets' => 'tmp/sockets',
  'system'  => 'public/system'
}
```

To get started, let's redefine the deploy:symlink task. We'll include a description, and we'll run this task with the same restrictions as the built-in tasks, only on app servers.

capistrano2/lib/deploy/deploy_symlink_override.rb

```
namespace :deploy do
  desc "Create symlinks to stage-specific configuration files and shared resources"
  task :symlink, :roles => :app, :except => { :no_release => true } do
  end
end
```

Now we can fill in the body of the task. We'll iterate over the cleanup_targets array and create a shell command to remove each of the target files/directories.

```
symlink_command = cleanup_targets.map { |target| \
  "rm -fr #{current_path}/#{target}" }
```

Next we'll do the same with the release_directories variable; it contains directories to be re-created on each deploy.

```
symlink_command += release_directories.map { |directory| "mkdir -p #{directory} }
```

Then we build more commands from the release_symlinks variable that contains symlinks that should be created from items in the release directory. These will typically be stage-specific configuration files that we don't mind checking in to our repository. The -s flag tells the ln utility to create a symlink, and the -f flag tells ln that if the symlink exists, it should remove and re-create it.

```
symlink_command += release_symlinks.map { |from, to| \
  "rm -fr #{current_path}/#{to} && \
  ln -sf #{current_path}/#{from} #{current_path}/#{to}" }
```

We build still more commands from the data in shared_symlinks. In this variable, *from* is relative to shared_path, and *to* is relative to the current_path.

```
symlink_command += shared_symlinks.map { |from, to| \
  "rm -fr #{current_path}/#{to} && \
  ln -sf #{shared_path}/#{from} #{current_path}/#{to}" }
```

Finally, we concatenate all of these into a single shell command that runs all of the directory and symlink commands at once.

```
run "cd #{current_path} && #{symlink_command.join(' && ')}"
```

Here's the entire task that we've built up line by line:

capistrano2/deploy_task.rb

```
namespace :deploy do
  desc "Create symlinks to stage-specific configuration files and shared resources"
  task :symlink, :roles => :app, :except => { :no_release => true } do
    symlink_command = cleanup_targets.map \
      { |target| "rm -fr #{current_path}/#{target}" }
    symlink_command += release_directories.map \
      { |directory| "mkdir -p #{directory} }
    symlink_command += release_symlinks.map \
      { |from, to| "rm -fr #{current_path}/#{to} && \
        ln -sf #{current_path}/#{from} #{current_path}/#{to}" }
    symlink_command += shared_symlinks.map \
      { |from, to| "rm -fr #{current_path}/#{to} && \
        ln -sf #{shared_path}/#{from} #{current_path}/#{to}" }
    run "cd #{current_path} && #{symlink_command.join(' && ')}"
  end
end
```

This technique nicely separates the configuration data values from the code that processes them, making the deployment configuration more readable. It's a win on both performance and clarity.