Cheat Sheet | CAPISTRANO 1.2.0

Helper Methods

run
Takes a single string identifying any valid shell command to execute

```
run <<-CMD
 if [[ -d #{release_path}/status.txt
]]; then
  cat #{release_path}/status.txt
 fi
CMD
```

sudo
Exactly like the `run` command, except that it executes the command via `sudo`

```
sudo "apachectl graceful"
```

put
Lets you transfer data to a file on the remote host. Takes two parameters: a string containing the data to transfer, and the name of the file to receive the data on each remote host

```
put(File.read('templates/database.yml'
),"#{release_path}/config/database.yml
",:mode => 0444)
```

delete
A convenience for executing `rm` via `run`. It just attempts to do an `rm -f` on the remote server(s), for the named file. To do a recursive delete, pass `:recursive => true`

```
delete "#{release_path}/certs",
:recursive => true
```

on_rollback
Allows a task to specify a callback to use if that task raises an exception when invoked inside of a `transaction`

```
task :update_code do
    on_rollback { delete release_path,
:recursive => true }
    ...
end
```

render
An interface for rendering ERb* templates and returning the result. If you pass a string to render, it is interpreted as the name of a template file with .rhtml appended, relative either to the current directory, or to `capistrano/recipes/templates` to be rendered. If you don't want to render a file, but instead have a string containing an ERb template that you want to render, use the second example below *Embedded Ruby

```
render "maintenance"
render :template => "Hello <%= target
%>", :target => "world"
```

config/deploy.rb

Setting and using variables

```
1 set :application, "flipper"
2 set :user, "homersimpson"
3 puts "The application name is
  #{application}"
4 puts "The user is #{user}"
```

Redefining the `restart` task

```
1 desc "This task restarts the web
  server"
2 task :restart, :roles =>
  :app do
3     sudo "apachectl graceful"
4 end
```

Defining tasks

```
1 task :hello_world do
2     run "echo Hello, $HOSTNAME"
3 end

1 task :hello_world, :roles =>
  [:db, :app] do
2     puts "calling hello_world..."
3     hello_world
4 end
```

Transactions

```
1 task :cold_deploy do
2     transaction do
3         task_one_here
4         task_two_here
5     end
6     task_three_not_in_transaction
7 end
```

Capturing output with `run`

```
1 run "sudo ls -la" do |channel,
  stream, data|
2   if data =~ /^Password:/
3     logger.info "#{channel[:host]}
  asked for password"
4     channel.send_data "mypass\n"
5   end
6 end
```

Parsing & saving an ERb template with `render`

```
1 buffer = render(:template =>
  <<EXAMPLE_TEMPLATE)
2 This template will be rendered
  replacing variables
  <%= like_this_variable =>
  with their values.
3 EXAMPLE_TEMPLATE
4 put buffer,
  "path/to/save/file.txt",
5 :mode => 0755
```

Directory structure

```
releases
|-- 20050725121411
|-- 20050801090107
|-- 20050802231414
|   ...
|-- 20050824141402
|   |-- Rakefile
|   |-- app
|   |-- config
|   |-- db
|   |-- lib
|   |-- log-->/shared/log
|   |-- public
|   |   `-- system-->/shared/system
|   |-- script
|   |-- test
|   `-- vendor
shared
|-- log
|-- system
current-->/releases/20050824141402
```

Capistrano Shell

Start the interactive Capistrano shell

```
cap -v shell
```

Execute Capistrano tasks

```
!deploy
!update_code symlink
!setup deploy
on app2.foo.com !setup
with app,db !setup deploy
```

Shell commands

Installation

```
gem install capistrano
```

Add your application to Capistrano (capistranize)

```
cap --apply-to /path/to/your/app
YourApplicationName
```

Execute the `setup` task

```
rake remote:exec ACTION=setup
```

Execute the `cold_deploy` task

```
rake remote:exec ACTION=cold_deploy
```

Deploy your application

```
rake deploy
```

Rollback a release from production

```
rake rollback
```

Execute the `disable_web` task

```
rake remote_exec ACTION=disable_web \
          UNTIL="tomorrow morning" \
          REASON="vital upgrade"
```

Using the `invoke` task

```
rake remote_exec ACTION=invoke \
          COMMAND="svn up
/u/apps/flipper/current/app/views" \
          ROLES=app
```

Roles

```
1 role :web•,    "www.capistrano.com"
2 role :app•,    "app1.capistrano.com",
  "app2.capistrano.com"
3 role :db•,     "master.capistrano.com",
  :primary => true
4 role :db•,     "slave.capistrano.com"
5 role :spare▪, "genghis.capistrano.com
```

- • standard, predefined roles
- ▪ user-defined roles

Pre-defined variables

`:application`	*(required)*	The name of your application. Used to build other values, like the deployment directory.
`:repository`	*(required)*	The location of your code's scm repository
`:gateway`	*nil*	The address of the server to use as a gateway. If given, all other connections will be tunneled through this server.
`:user`	*(current user)*	The name of the user to use when logging into the remote host(s)
`:password`	*(prompted)*	The password to use for logging into the remote host(s).
`:deploy_to`	*"/u/apps/#{application}"*	The root of the directory tree on the remote host(s) that the application should be deployed to
`:version_dir`	*"releases"*	The directory under `deploy_to` that should contain each deployed revision
`:current_dir`	*"current"*	The name to use (relative to `deploy_to`) for the symlink that points at the current release
`:shared_dir`	*"shared"*	The name of the directory under `deploy_to` that will contain directories and files to be shared between all releases
`:revision`	*(latest revision)*	This specifies the revision you want to check out on the remote machines
`:scm`	*:subversion*	The source control module to use. Current supported are `:subversion, :cvs, :darcs`
`:svn, :cvs, :darcs`	*(path)*	The location on the remote host of the source control executable
`:checkout`	*"co"*	The subversion operation to use when checking out code on the remote host. Can be set to `"export"`
`:ssh_options`	*Hash.new*	Hash of additional options passed to the SSH connection routine. This lets you set (among other things) a non-standard port to connect on `(ssh_options[:port] = 2345)`
`:use_sudo`	*true*	Whether or not tasks that can use `sudo`, ought to use `sudo`. In a shared environment, this is typically not desirable (or possible), and in that case you should set this variable to false
`:sudo`		sets the path to `sudo`

Standard tasks

`cleanup`	cleans up the `releases` directory, leaving the five most recent releases
`cold_deploy`	used when deploying an application for the first time. Starts the application's spinner (via the `spinner` task) and then does a normal `deploy`
`deploy`	updates all the code on your server (via `update_code` and `symlink` tasks), then restarts the FastCGI listeners on the application servers (via the `restart` task).
`diff_from_last_deploy`	prints the difference between what was last deployed, and what is currently in your repository
◆`disable_web`	puts up a static maintenance page that is displayed to visitors
`enable_web`	removes the maintenance page
`invoke`	allows you to send commands directly
`migrate`	changes to the directory of your current release (as indicated by the current symlink), and runs `rake RAILS_ENV=production migrate`
`restart`	restarts all FastCGI listeners for your application by calling the `reaper` command without arguments. Only executed on `:app` servers
`rollback`	rolls your application back to the previously deployed version
`rollback_code`	determines the previous release , updates the current symlink to point to that, and then deletes the latest release
`setup`	Creates and chmods the directory tree: `releases_path` directory 0775 `shared_path` directory `shared_path/system` 0775 `shared_path/log` 0777
`show_tasks`	inspect the existing tasks and display them to standard out in alphabetical order, along with their descriptions
`spinner`	starts the `spinner` process for your application
◆`symlink`	updates the current symlink to the latest deployed version of the code
◆`update_code`	Checks out your source code, deletes the `log` and `public/system` directories in your new release, symlinks `log` to `#{shared_path}/log`, symlinks `public/system` to `#{shared_path}/system`

◆task already has a defined `on_rollback` handler when using **`transactions`**