



Chapters

- 1. [Introduction](#)
 - [What is Capistrano?](#)
 - [What can it do?](#)
 - [What assumptions does it make?](#)
- 2. [Quick Start](#)
 - [Getting started](#)
 - [Installing Capistrano](#)
 - [Deployment Recipe](#)
 - [Setup](#)
 - [Apache Configuration](#)
 - [Deploying](#)
 - [Rolling back a release](#)
- 3. [A More Complicated Example](#)
 - [Getting started](#)
 - [Deployment Recipe](#)
 - [Spinner](#)
 - [Spawner](#)
 - [Reaper](#)
 - [Deploying](#)
- 4. [Recipes](#)
 - [Introduction](#)
 - [Variables](#)
 - [Roles](#)
 - [Tasks](#)
 - [Extending Tasks](#)
- 5. [Standard Tasks](#)
 - [Overview](#)
 - [cleanup](#)
 - [cold_deploy](#)
 - [deploy](#)
 - [diff_from_last_deploy](#)
 - [disable_web](#)
 - [enable_web](#)
 - [invoke](#)
 - [migrate](#)
 - [restart](#)
 - [rollback](#)
 - [rollback_code](#)
 - [setup](#)
 - [show_tasks](#)
 - [spinner](#)
 - [symlink](#)
 - [update_code](#)
- 6. [Creating Tasks](#)
 - [Overview](#)
 - [run](#)
 - [sudo](#)
 - [put](#)
 - [delete](#)
 - [render](#)
 - [transaction](#)
 - [on_rollback](#)
- 7. [Extending Capistrano](#)
 - [Task Libraries](#)
 - [Writing a Task Library](#)
 - [Using a Task Library](#)
 - [Extension Libraries](#)

Options

- [exports](#)
- [recent changes](#)
- [rss 2.0](#) | [atom](#)

Authors

- [Login](#) [Signup](#)

1. Introduction

1.1 What is Capistrano?

To say that Capistrano is a utility for deploying web applications would be akin to saying that computers are machines that let you type school papers. It's a gross understatement. Capistrano is actually capable of doing far, far more than just deploying web apps. However, because deployment of web apps is what Capistrano was originally created for, this manual will focus on that, and then spend a little time at the end showing some of the other possibilities.

Historically, Capistrano was originally called SwitchTower. The name was changed in March 2006 in response to a trademark conflict.

1.2 What can it do?

Ultimately, Capistrano is a utility that can execute commands in parallel on multiple servers. It allows you to define *tasks*, which can include commands that are executed on the servers. You can also define *roles* for your servers, and then specify that certain tasks apply only to certain roles.

Capistrano is very configurable. The default configuration includes a set of basic tasks applicable to web deployment. (More on these tasks will be said later.)

Capistrano can do just about anything you can write shell script for. You just run those snippets of shell script on remote servers, possibly interacting with them based on their output. You can also upload files, and Capistrano includes some basic templating to allow you to dynamically create and deploy things like maintenance screens, configuration files, shell scripts, and more.

1.3 What assumptions does it make?

As with the rest of Rails, Capistrano makes many assumptions, both about your code, and the way you do things with it (like deployment).

There are basically two levels of assumptions in Capistrano: *core* assumptions, and assumptions made by the default tasks.

The core assumptions of Capistrano tend to be quite general (though there are some exceptions), and are not usually possible to override. They are:

- You are interacting with at least one remote server.
- You may need to tunnel through a gateway server to access your target server.
- You are using SSH to connect to the servers.
- The remote server is capable of understanding POSIX shell commands. (Windows, by default, does *not* fall into this category. Neither do shells like csh and tcsh. Sorry.)
- The password for all servers is the same.
- Some things you only want to execute on a subset of your production environment, rather than on all of your production servers.

The assumptions made by the default tasks are more specific, but are either configurable or overridable. Some of them are:

- You are deploying a web application.
- You are using Ruby on Rails to develop your application.
- You are using subversion to manage your source code.
- You are deploying your application to `"/u/apps/#{appname}"` on every machine.
- You are using FastCGI to power your application.
- You are fronting your app with either lighttpd or apache.

This manual will hold to these same assumptions, but will also show (where applicable) how to configure or override them.

[next chapter »](#)