



**Simple and powerful site
deployment with capistrano**

PreviousNext

PreviousNext[®]

Kim Pepper

kim@previousnext.com.au

[@scorchio96](#)

d.o: [kim.pepper](#)

How are you deploying now?

- FTP, SFTP, SCP?
- SSH + GIT?
- Aegir?
- Drush?
- Debian packages?
- Don't know - operations team does it for me!

What are the issues?

- lots of manual processes in deployment
 - backup db
 - update code
 - run updatedb
 - flush caches
- potential for human error

Small organisation?

- too loose server access and permissions

Large organisation?

- too restrictive permissions requiring sysadmin or super user to deploy



You didn't know you had a problem?

A great tool for web ops needs to be automatic, accessible and actionable” – [@briandoll](#)

Sysadmins don't want you touching their stuff





Don't cross the line

- Sysadmins don't want you touching their stuff

Standard server environments:

- conventions
- consistency
- security
- monitoring
- quality control
- automated configuration management tools (e.g. puppet, chef)

Working together to get the job done





Working together to get the job done

Ideally:

- Developers are able to deploy rapidly
- Sys admins have tight control of environments
- All manual processes are automated

Capistrano





Capistrano

Capistrano is a developer tool for deploying web applications. It is typically installed on a workstation, and used to deploy code from your source code management (SCM) to one, or more servers.

Server Directory Structure

- Each deployment lives in its own time-stamped directory
 - e.g. 20111101020759/
- settings.php and files directory are shared across deployments, and don't live in version control
- a 'current' symlink points to the current release, and is used as the site root in apache config

Server Directory Structure

e.g. under **`/var/www/example.com/`**

current -> **`20111101020759`** (*symlink to current release*)

releases/

`20110929042332/`

`20110929045734/`

`20111101001651/`

`20111101020759/`

shared/

`cached-copy/`

`files/`

`settings.php`



Symlinks for shared files

Symlinks are made back to the **shared** folder, so they are not lost between deployments, e.g.:

sites/default/files -> shared/files

sites/default/settings.php -> shared/settings.php



How does it work?

- No software installed on the server, just locally
- Capistrano 'builds' commands to be executed remotely
- executed on the server over ssh

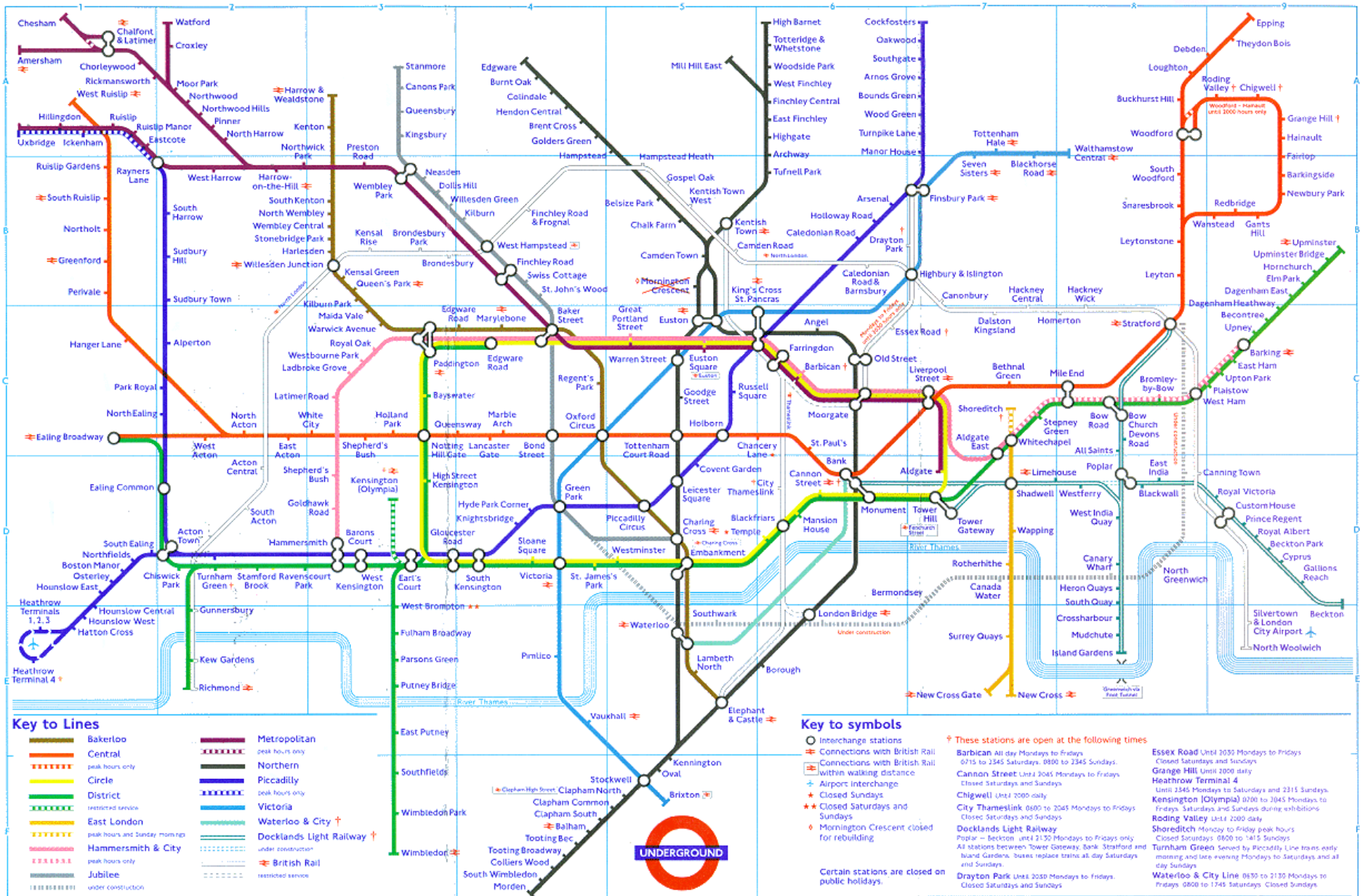


Executing a deploy

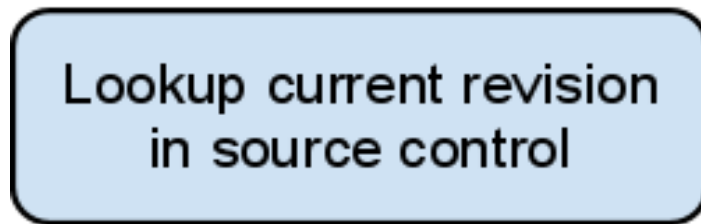
It's a simple one-line command to deploy from your local machine:

```
cap deploy
```

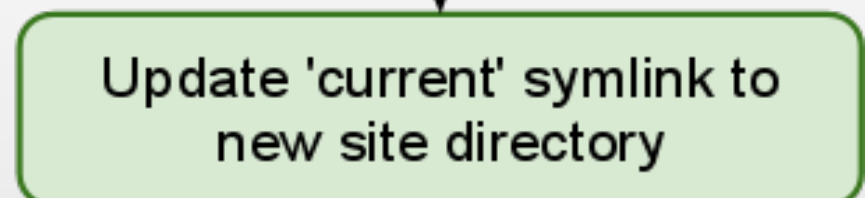
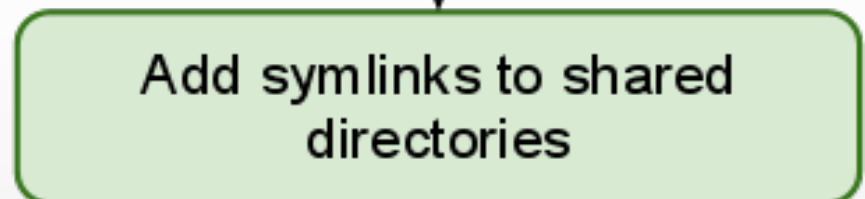
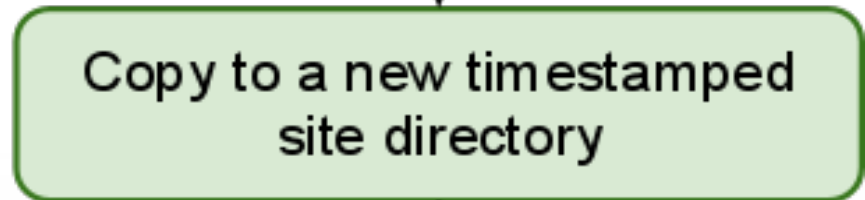
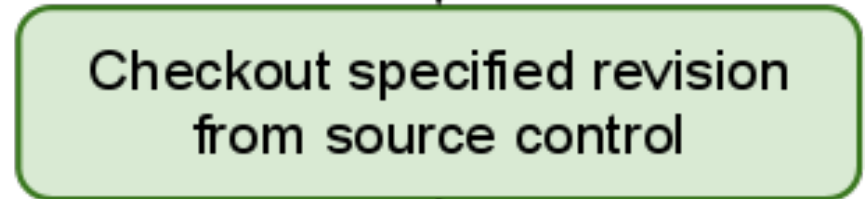
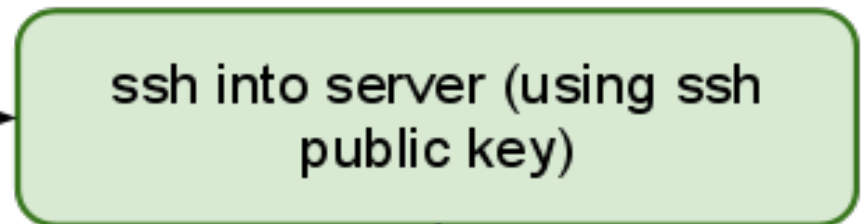
Deployment Workflow



Local



Server



Simple Deployment Workflow

Hooks

- *before* and *after* hooks at each stage of the deployment lifecycle
- extend or customise default capistrano behaviour

Examples:

- `deploy`
- `deploy:setup`
- `deploy:symlink`
- `deploy:update_code`
- `deploy:rollback`

Rolling-back a deploy

Made a mistake? Rollback with:

```
cap deploy:rollback
```

- Sets the 'current' symlink to the previous release directory
- No automatic re-import from backed up database

Full list of available tasks with:

```
cap -T
```



Set-up and Configuration

What is it written in?



Ruby! (I know, it's not made with Drupal).

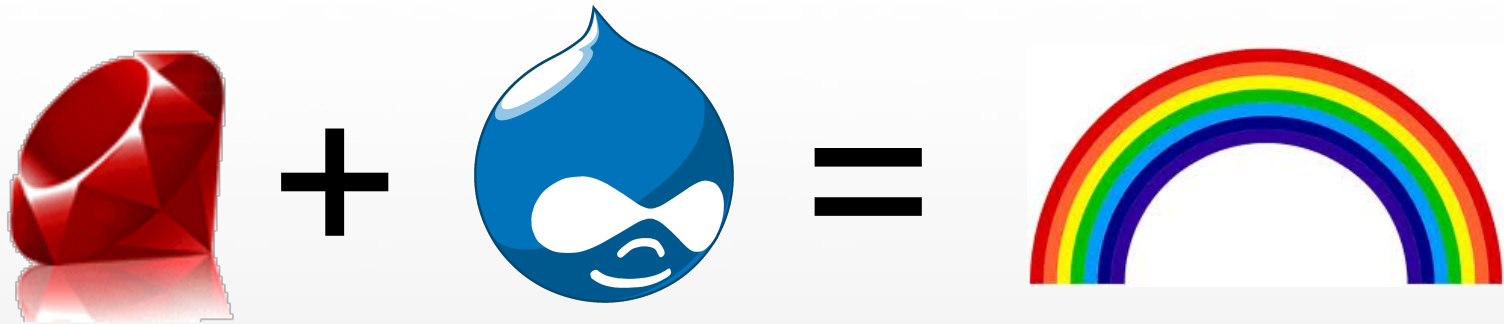
- Originally designed for deploying Rails apps
- Cross-platform
- Many OS's have it already (e.g. Mac OSX)
- RubyInstaller for Windows users
<http://rubyinstaller.org/>
- Rubygems installed from <http://rubygems.org/>

RubyGems you'll need

- capistrano
- capistrano-ext
- railsless-deploy
- capistrano-drupal

```
gem install capistrano capistrano-ext \
  railsless-deploy capistrano-drupal
```

A Drupal Helper Gem



capistrano-drupal gem

<http://rubygems.org/gems/capistrano-drupal>

- performs common Drupal deployment tasks
- also calls **Drush** commands on the server
- implements *before* and *after* hooks

deploy:setup

- Creates files directory and sets correct group-writable permissions

drupal:symlink_shared

- re-creates symlinks to settings.php and files directory
- called after "deploy:symlink"

Capistrano-drupal gem *cont...*

drush:backupdb

- Calls the drush backup-migrate command to create a db backup
- called before "drush:updatedb"

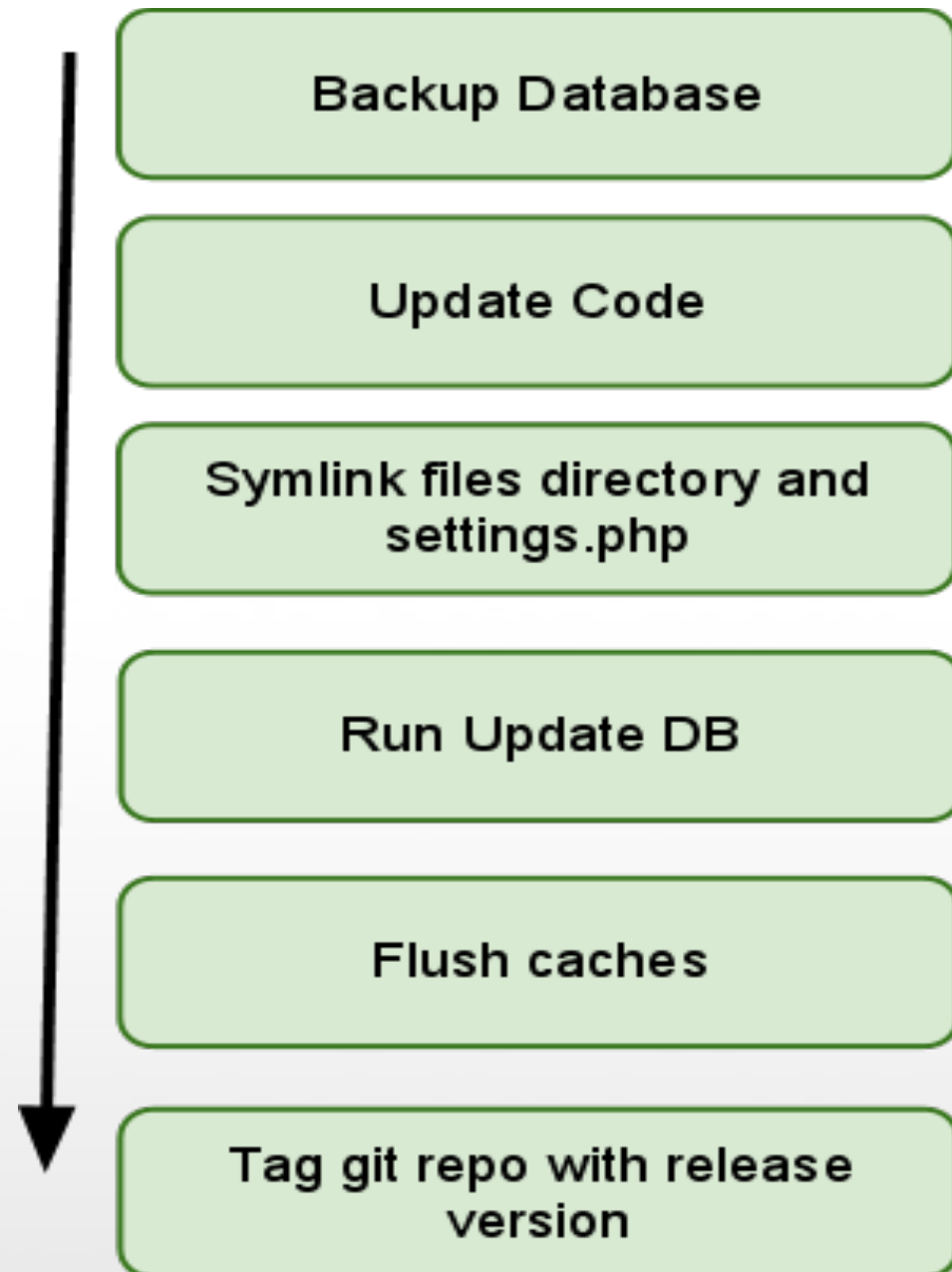
drush:updatedb

- Calls the drush updatedb command
- called after "deploy:symlink"

drush:cache_clear

- Calls the drush cc all command to clear the Drupal caches
- called after "deploy:symlink"

Drupal deployment workflow





Using Capistrano in Your Project

Site Directory Structure

Drupal site root lives under an *app* directory so it stays separated from the *deploy config* directory.

e.g. locally under **~/Sites/example.com/**

app/

<drupal root>

config/

deploy.rb

Capfile

Deploy Configuration

Configure your server settings in your **deploy.rb** file

```
# example drupal site
set :application, "example.com"
set :scm, "git"
set :repository, "git@github.com:example/example.git"
set :branch, "master"
set :port, 22
set :deploy_to, "/var/www/#{application}"
set :user, "exampleuser"
set :runner, "exampleuser"
role :app, 'www.example.com'
```



Optional Extras


Server Roles

- Support for multiple servers
 - e.g 2 app servers, 1 db server
- Specified as roles:

role :app, 'www.example.com'

role :db, '10.1.1.10'

- Capistrano tasks can be run only on servers with specific roles (e.g. only deploy code on app server)



Password-less Deploy

- Use ssh public keys
- Less manual process involved
- Uses ssh key-forwarding for accessing github repo

Multi-Stage Deployments

- Have different deployment configuration for dev, staging, and production environments (or more)
- e.g. different hosts, git branches, root directory
-

config/

deploy.rb

deploy/

dev.rb

staging.rb

prod.rb

Deploy using a stage name:

```
cap prod deploy
```

Custom commands

Add your own 'tasks' by defining them in `deploy.rb`.

This will echo 'hello' on the server after updating code.

```
before "deploy:update_code", "mynamespace:say_hello"
namespace :myspace do
  desc "Says hello to all the people out there"
  task :say_hello do
    run 'echo hello'
  end
end
```



Filtering

Add filters to specific 'roles' or hosts:

```
cap deploy HOSTS=app2.server.hostname  
cap deploy ROLE=app
```



Gateways

You can set up a gateway server to tunnel deployments through:

```
set :gateway, 'deployer@example.com.au:2222'
```



Git Submodules

You can enable capistrano to automatically fetch updates for git-submodules on deploy



Demo

PreviousNext





Questions?

PreviousNext



PreviousNext[®]

Kim Pepper

kim@previousnext.com.au

[@scorchio96](#)

d.o: [kim.pepper](#)