

Bread and Butter Capistrano

Kelly Felkins
kelly@eBlock.net

The Goal

```
~/dev/my_app kelly$ # do some editing
~/dev/my_app kelly$ # check it in
~/dev/my_app kelly$ # now deploy it!
~/dev/my_app kelly$ rake remote:deploy
~/dev/my_app kelly$ # changes now on *all*
~/dev/my_app kelly$ # production servers
```

deployment night (the old way)

- ssh to production server #1
- take down production application
- copy code base over (rsync, scp, check-out)
- perform database updates (either manually, or via a script)
- restart application
- check the application

deployment

the rails/capistrano way

- database migrations are part of application
- capistrano automates deployments to multiple servers in parallel
- deployments become casual -- possibly **too** casual

What is Capistrano?

- A tool for automating deployment created by Jamis Buck
- Performs commands on multiple remote servers in parallel using ssh
- Performs commands based on an application configuration

What is Capistrano? (2)

- Plays nicely with Rails
 - deployment configuration is part of rails app
 - can perform migrations during deployment
 - can restart mongrels/cgi's
- Commands are **transactional**
 - if they fail changes are rolled back automatically
 - or you can roll back to a previous release

human issues command here:
`$ rake remote:deploy`

**development
system**

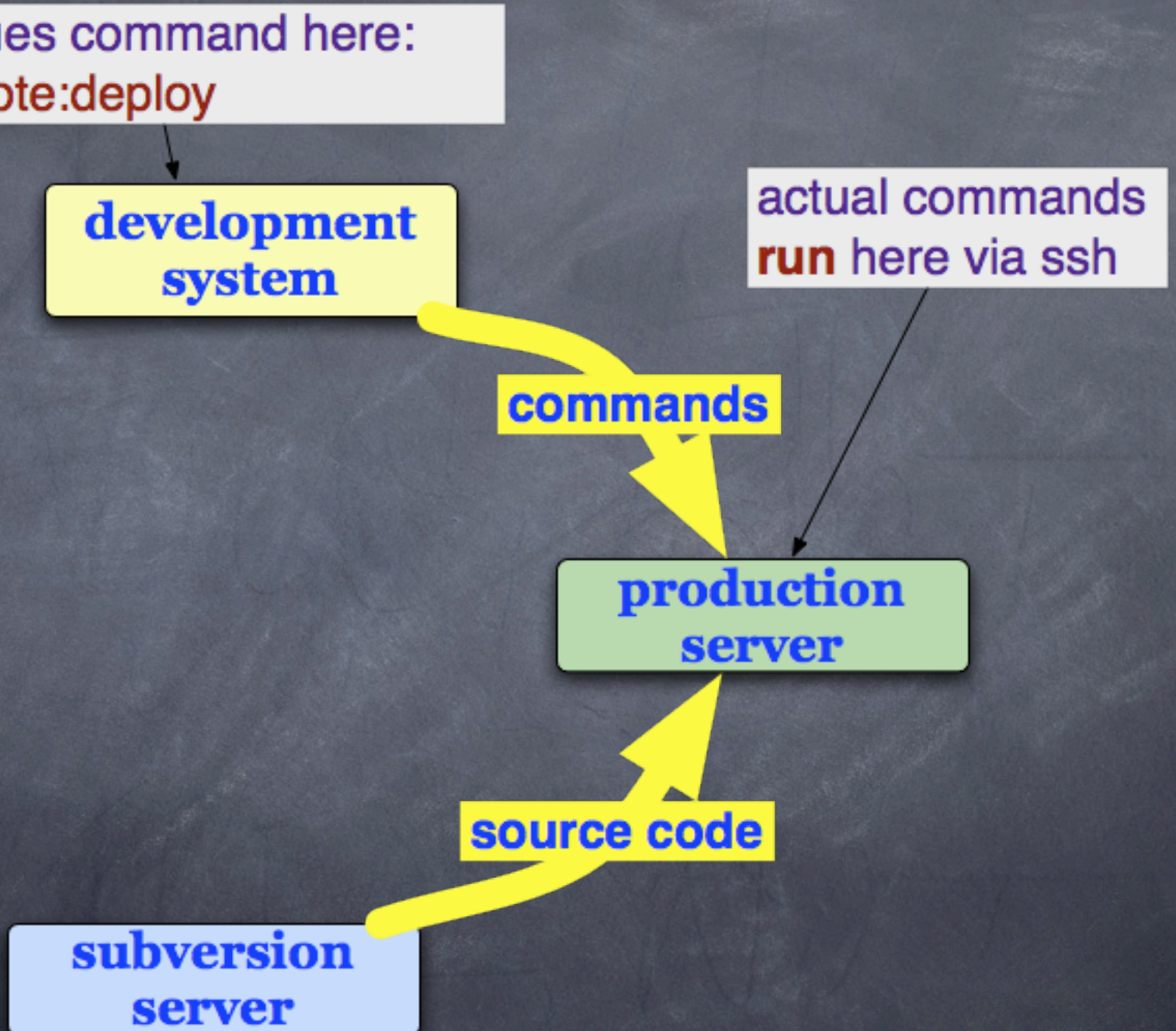
actual commands
run here via ssh

commands

**production
server**

source code

**subversion
server**



Server Directory Structure

application area on server has 3 items

- 'releases' directory containing dated directories for each release
- 'current' sym link to the current release directory
- 'shared' directory for non-release specific stuff

```
/u/apps/myapp
|-- current -> /u/apps/myapp/releases/20070219023318
|-- releases
|   |-- 20070217011123
(stuff deleted)
|   |-- 20070219023318
|       |-- app
|       |-- components
|       |-- config
|       |-- db
|       |-- doc
|       |-- lib
|       |-- log -> /u/apps/myapp/shared/log
|       |-- public
|       |-- script
|       |-- test
|       |-- tmp
|       |-- vendor
|-- revisions.log
`-- shared
    |-- config
    |   |-- database.yml
    |   |-- mongrel_cluster.yml
    |-- log
    |   |-- mongrel.8000.pid
    |   |-- mongrel.log
    |   |-- mongrel_debug
    |   |-- production.log
    |-- public
    |   |-- image
    |-- system
```

Getting Started

One Time

- Install capistrano and associated gems

```
~ kelly$ sudo gem install capistrano -y
Successfully installed capistrano-1.4.0
Successfully installed net-ssh-1.0.10
Successfully installed needle-1.3.0
Successfully installed net-sftp-1.1.0
~ kelly$ sudo gem install termios
Successfully installed termios-0.9.4
~ kelly$
```

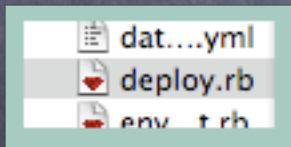
Each App (step one)

Create deploy.rb file

```
~/dev kelly$ cap -A myapp/  
exists  config  
create  config/deploy.rb  
exists  lib/tasks  
create  lib/tasks/capistrano.rake  
~/dev kelly$
```

Each App (step two)

Update deploy.rb



```
# repository
# correspond
# form the ro

set :application, "application"
set :repository, "http://svn.yourhost.com/#{application}/trunk"

# -----
# ROLES
# -----
# You can define any number of roles, each of which contains any num
# machines. Roles might include such things as :web, or :app, or :db
# what the purpose of each machine is. You can also specify options
# be used to single out a specific subset of boxes in a particular r
# :primary => true.

role :web, "www01.example.com", "www02.example.com"
role :app, "app01.example.com", "app02.example.com", "app03.example.
role :db,
role :db,
```

```
set :application, "application"
set :repository, "http://svn.yourhost.com/#{application}/trunk"
```

```
role :web, "www01.example.com", "www02.example.com"
role :app, "app01.example.com", "app02.example.com"
role :db, "db01.example.com", :primary => true
```

Each App (step three)

Initialize 1 or more production servers

```
$ rake remote:setup
```

I almost always have trouble with permissions on this step. All it does is:

```
mkdir -p /u/apps/{app name}/shared  
mkdir -p /u/apps/{app name}/releases
```

If you have trouble, go fix it.

Deployments: Updating Remote Servers

```
~/dev/my_app kelly$ # do some editing
~/dev/my_app kelly$ # check it in
~/dev/my_app kelly$ # now deploy it!
~/dev/my_app kelly$ rake remote:deploy
~/dev/my_app kelly$ # changes now on *all*
~/dev/my_app kelly$ # production servers
```

No, really...that's it.

Common Commands

<code>rake remote:deploy</code>	deploy application
<code>rake remote:migrate</code>	run migrations
<code>rake remote:deploy_with_migrations</code>	deploy and run migrations
<code>rake remote:restart</code>	restart mongrels/fcgi's
<code>rake remote:disable_web</code> (or <code>enable_web</code>)	display a static page indicating site is off-line

Note: rake tasks for capistrano commands are deprecated. Unfortunately, I don't know the 'cap' versions yet...

Recipes

- * Keep Production DB Passwords Secure
- * Deploy to Staging Or Production
- * Fastcgi, but no Sudo

Capistrano Events

- * `after_setup`
- * `before_setup`
- * `after_cold_deploy`
- * `after_update_code`
- * `before_restart_mongrel_cluster`
- * `restart`

Commonly Used Variables

application	used as scm project name and deployment directory
repository	url of the repository, for check-outs
user	User name to use for ssh operations
deploy_to	directory to deploy to, defaults to /u/apps/{application}
use_sudo	use sudo for operations
scm	scm to use, defaults to subversion

Recipe: Keep Production DB Passwords Secure

Problem: database.yml is in config dir, but contains sensitive passwords

Solution:

- * remove database.yml from scm
- * create host specific database.yml's in /u/apps/[my_app]/shared/
- * create sym link to host specific database.yml in 'after_update_code' event.

Recipe: Keep Production DB Passwords Secure

```
task :after_update_code do
  run "ln -s #{shared_path}/config/database.yml "+
    "#{release_path}/config/database.yml"
end
```

Recipe: Deploy to Staging Or Production

Problem: You need to deploy to a staging environment for testing prior to deployment to the production environment

Solution:

- * create multiple server configurations
- * select server config based on environment variable

Recipe: Deploy to Staging Or Production

```
if ENV["DEPLOY_TYPE"] == 'public'  
  role :web, "master.bogus.org"  
  role :app, "master.bogus.org"  
  role :db, "master.bogus.org", :primary => true  
else  
  role :web, "stage.bogus.org"  
  role :app, "stage.bogus.org"  
  role :db, "stage.bogus.org", :primary => true  
end
```

```
~/dev/myapp kelly$ rake remote:deploy DEPLOY_TYPE=production
```

Recipe: Fastcgi but no Sudo?

Problem: You need to restart fastcgi processes but your sys admin is a control freak and won't give you sudo

Solution:

- * fastcgi processes are started by the web server, so let the web server restart them.
- * Create a cgi script that restarts fastcgi processes.
- * Invoke it with the 'restart' action

Recipe: Fastcgi but no Sudo?

```
#!/usr/bin/perl
use File::Basename;
print "Content-type: text/plain\n\n";
my $base_dir = dirname($0);
my $psCommand =
    "ps -aef | grep \"ruby $base_dir/dispatch.fcgi\" | grep -v grep";
print "Before kill commands:\n", ` $psCommand ` , "\n\n";
my @processList = ` $psCommand `;
foreach my $processLine (@processList) {
    my ($processId) = (split /\s+/, $processLine)[1];
    if ($processId =~ /\d+$/) {
        my $killCommand = "kill -TERM $processId";
        print $killCommand, "\n";
        print ` $killCommand `;
    }
}
sleep(1);
print "\n\n", "After kill commands:\n", ` $psCommand `;
```

Yikes! Perl!

Recipe: Fastcgi but no Sudo?

```
desc "Restart the FCGI processes on the app server."  
task :restart, :roles => :app do  
  run "wget \"http://master.bogus.org/reaper.cgi\""  
end
```

Questions?

Kelly Felkins
kelly@eblock.net