Drupal Capistrano - Cookbook for using CUL Development Server

The development server is capdev.library.cornell.edu (aka sf-lib-web-006.serverfarm.cornell.edu). Each developer maintains a site, [netid].capdev.library.cornell.edu, on capdev. Look in /libweb/sites/ for your site.

Prepare the site for deployment

The deployment is based on the git repository for this site stored on git.library.cornell.edu. You have to make sure the repository accurately reflects your development environment before deploy.

1. Update the drush make file.

```
cd mysite
cul_capistrano/update_enabled_modules_list.sh
```

> *This script ends showing the differences between an automatically generated drush make file and your local.make.*
> *Be sure the versions of contributed modules match between these two files. Edit the versions in drupal_config/local.make to make them match.*
> *(The generated make use * for the drupal core version. You should use a specific core version in your local.make)*

2. Run the make file to be sure all changes are in your dev site.

```
cd mysite
cul_capistrano/do_make.sh
```

3. Commit all changes to your local git repo

```
git status
git add --all
git commit drupal_config/ -m 'my enabled modules, themes and make files'
git commit public/ -m 'my new drupal core, module, and theme changes'
git commit --all -m 'anything else, like maybe config/ changes'
```

> *In real life you'll be committing each change as you go with an informative message attached to each commit, right?*

4. Push the local git repo to it's origin

```
git status
git push
```

> *git status should return something like:*
> *# On branch master*
> *nothing to commit, working directory clean*

Deploy a site

You can specify as many target remote machines as you like in your cinfig/deploy.rb set :stages directive, but normally there'll be staging (victoria02) and production (victoria01). Normally staging is set as the :default_stage so you don't have to specify anything on the cap command to target the staging server.

1. follow the steps in "Prepare the site for deployment"
2. deploy to the staging server (victoria02)

```
cap deploy
```

3. or deploy to the production server (victoria01 or victoria03)

```
cap production deploy
```

What this does is clones the git repo to the target machine, moves all the site code to a timestamped directory there, updates a symbolic link pointing the document root at the new code, does the Drupal update.php procedure, reverts all features, enables all the modules enabled in the dev site, clears the cache, and a few other Drupal things. It makes a snapshot of the Drupal database before all this, but leaves the target database intact (except for the module updates), so the target site's new content is preserved.

Update Drupal modules

You can use several methods to update the modules. In Drupal, go to admin/modules/update and follow the instructions. On the command line do this:

```
cd mysite
cd public
drush up
```

You can also update the version numbers in drupal_config/local.make, then

```
cd mysite
cul_capistrano/do_make.sh
```

Update Drupal core & modules

The best way is to update the Drupal core version number in drupal_config/local.make then run the make file:

```
cd mysite
cul_capistrano/do_make.sh
```

Apply patches

The best way is to update the drupal_config/local.make file with the patch info:

```
projects[kaltura][version] = "2.0"
projects[kaltura][patch][] = "http://drupal.org/files/kaltura-1567302-dispaly.patch"
```

then run the make file:

```
cd mysite
cul_capistrano/do_make.sh
```

Copy a remote database to the dev machine

```
cd mysite
cap production drupal:db:grab
```

Install a database dump on remote machine

```
cd mysite
ls backup/

README.txt    staging-snapshot-20140114162844.sql

cap staging drupal:db:install -s file=backup/staging-snapshot-20140114162844.sql
```

Install a database dump on the dev machine

```
cd mysite

ls backup/

db-features.test.library.cornell.edu staging-snapshot-2013-10-23-16-5-59.sql staging-snapshot-2013-10-28-11-44-
8.sql
local-snapshot-2013-10-24-16-19-52.sql staging-snapshot-2013-10-24-11-32-41.sql staging-snapshot-2013-10-29-10-
57-19.sql
production-snapshot-2013-10-30-12-35-22.sql staging-snapshot-2013-10-24-9-24-42.sql
staging-snapshot-2013-10-23-14-17-14.sql staging-snapshot-2013-10-25-13-17-23.sql

cul_capistrano/local_db_install.sh production-snapshot-2013-10-30-12-35-22.sql
```

Create a development branch

Merge a development branch

Prepare a new site on the dev machine

Make a git repo for the site on git.library.cornell.edu. Use the test site name, e.g. seadina.test.library.cornell.edu.

Run the script to create the initial site directory.

```
cd my_cap_dev_area
setup_cap_dev_site.sh seadina_test_library_cornell_edu.git
```

Create the database, make a symbolic link from htdocs to the public directory (where Drupal is), install Drupal.

Then add everything to git, commit, and push.

Prepare a new site on a remote machine

```
cd mysite
cap staging deploy:setup
cap staging drupal:site_setup_permissions
cap staging drupal:install -s db=databasename -s dbu=databaseuser -s dbpw=databasepassword \
        -s d_user=Drupal_user#1 -s d_pw=Drupal_user#1_password -s d_email=Drupal_user#1_email
```

Update cul_capistrano

The cul_capistrano directory is a 'git subtree'. If the separate git repo cul_capistrano comes from has been updated you can update your copy:

```
cd mysite
git subtree pull --prefix cul_capistrano git@git.library.cornell.edu:cul_capistrano.git master --squash -m
'merge cul_capistrano updates'
```

Undo a site deploy

```
cd mysite
cap staging deploy:rollback
```

Note: this also restores the database to the state it was at before the previous deploy.

List all the capistrano commands

```
cd mysite
cap -T
```

Enable passwordless login to remote machines

```
drush dl drush_extras
drush pushkey victoria02.library.cornell.edu
drush pushkey victoria01.library.cornell.edu
drush pushkey victoria03.library.cornell.edu
```