Woven Documentation

Release 0.9 pre-alpha

Brett Haydon

July 02, 2013

CONTENTS

Woven deploys versioned Django projects onto Linux servers behind Nginx using Apache modwsgi by default. Woven is is built on Fabric.

RTFM.

Note: Woven is still alpha software, and the api and conventions may change between versions. It is not recommended for production deployment at this stage. It has been tested (in the most loose sense of the term) using Ubuntu Server 10.04 hosts and greater. It may work on other debian based distributions (reports welcome). It currently *won't* work at all with Windows, due to use of rsync, and needs some work to be compatible with redhat based distributions.

CHAPTER

ONE

CONTENTS:

1.1 Woven

- utilizes Pip, Virtualenv, Fabric and Django templates to simplify deployment making *release early and often* a reality.
- provides six Django commands/functions; `setupnode`, `deploy`, `patch`, `activate`, `bundle`, `startsites`
- provides a standard Apache & Nginx webserver configuration that you can change as required, and includes experimental support for Gunicorn.
- versions your project with virtualenv
- enables you to define custom linux etc configuration files in your project
- enable you to define roles for your servers with combinations of linux packages and firewall rules
- provides an api for fabfile.py scripts.
- hooks for custom functionality
- integrates with South for migrations
- basic multi-site multi-db capabilities

Woven currently doesn't:

- Create and launch your node from any cloud providers - it just does the setup of a baseline installation with setupnode
- bring you beer

1.2 Installation

You are using pip right?

```
pip install woven
```

You may also want to install the following recommended packages:

```
pip install south
```

1.3 Getting Started

To use Woven you must have root access to a linux host or vm. Woven has currently been tested on Ubuntu >= 10.04.

Woven uses a custom `woven-admin.py` script that serves to replace `django-admin.py` and `manage.py` in your development environment, and allows you to use woven commands without adding woven to your installed apps.

Run `woven-admin.py startproject` which will create a basic django project distribution layout.

Woven provides six management commands for your Django project:

`setupnode`, `deploy`, `patch`, `activate`, `bundle`, `startsites`

You can walk through some of the commands in a simple example django project *Tutorial* or read the *Management Commands* docs. You'll also want to have a look at the *Settings* and the project *Conventions*.

1.4 Integrating with Fabric

Woven is just fabric. To integrate with your own fabfiles you can do:

```
import os

#import any other woven functions you want to use or all of them
from woven.api import *

#Set the environ for Django if you need to
#(assumes you have your project on the pythonpath)
os.environ['DJANGO_SETTINGS_MODULE'] = 'example_project.settings'

#set_env will initialise your env for woven functions and use your settings
set_env()

#define your own fabric functions that use woven
def your_function():
    ...
```

1.5 Custom Hooks

Woven provides hooks into the setupnode, deploy, and post package installation commands.

To add custom functionality to a woven deployment create a `deploy.py` file in your project or django app, and define any of the following.

1.5.1 Post install package

Define a `def post_install_[package_name]()` function to run code if the Ubuntu package is installed by woven, and after any /etc configuration is uploaded. For example you might define `def post_install_postgresql()` to setup a postgresql database. Replace any dashes or fullstops with underscores to make it a valid python function.

A sample hook is defined in Woven for installing postgresql

1.5.2 Post setupnode

`def post_setupnode()` executes at the end of the setupnode process.

1.5.3 Post deploy

`def post_deploy()` executes at the end of deployment of your project but prior to activation.

Hooks execute in a project, app, woven order of precedence. Only one hook per function will execute. A project scope hook for instance will override the same function at app or provided by woven itself.

1.6 Multiple Sites

Woven creates a user on the node for each SITE_ID. The users will be called `site_1`, `site_2` ... The Apache site templates use process groups to launch modwsgi daemons running as the site user. The settings file gets the current user and dynamically sets the SITE_ID in the settings. In this fashion a single settings file can be used for multiple sites.

Since Django sites uses a SITE_ID rather than a domain to represent the site, it doesn't really know about subdomains, but you might want might make a default admin view of your SITE_ID = 1 at admin.example.com. To accommodate this you can make a settings file called admin_settings.py in the sitesettings folder. You would add a prefix for any other alternative view of the same site using [sub_domain]_settings.py

1.7 Development

The core highlevel functions setupnode, deploy, patch, bundle, and activate will not change, but some other parts of the api may still change between versions. I am aiming to release a beta version (something less than 1.0) sometime after the release of Fabric 1.0, since Fabric 1.0 will currently break Woven due to backwards incompatabilities. After that time Woven will depend on Fabric >= 1.0.

1.7.1 Future Goals

- I would like to see other ways of serving django behind nginx implemented. Ubuntu 11.04 will ship with a nginx that doesn't need to be compiled to use uwsgi, so that will be a good time to add support for it.
- Although it's currently tested on Ubuntu, I'm happy to accept patches or feedback to make it work on debian based distro's or other linux distributions.
- The future is distutils2. I actually think the 1.0 version of woven should make full use of the new packaging system and metadata. When packaging no longer sucks, I can simplify woven to leverage it, and it has implications for django project conventions. I'll be looking for distutils2 to move to beta before I begin to develop for it.

1.7.2 Testing

Woven stores tests in a fabfile.py in the tests directory. The tests are pretty incomplete at this time.

Individual tests are split across multiple modules corresponding to the module they are testing.

Tests require a vm to be setup with `root` login and password `root` on 192.168.188.10

Tests can be run by `fab test` to run all tests or `fab test_[modulename]` to run all tests in the given test module (eg `fab test_env`), or `fab [full_test_name]` to run an individual test from any test module.

1.8 Contact

The woven project is hosted on github at http://github.com/bretth/woven. Feature requests and bug reports are welcome.

You can contact me at brett at haydon dot id dot au

1.9 Tutorial

A simple django project is the best way to illustrate how to get started with woven (and django)

1.9.1 Starting the project

Note: This first bit doesn't have much to do with woven, and is more about personal preference in setting up your development environment but lets walk through *one* way you can get setup. For this you probably want pip, virtualenv and virtualenvwrapper installed and working before you can begin.

We're going to create a virtual python environment `first-django`. You don't need to do this but virtualenv makes it easy to experiment without polluting your system installed packages.

```
mkproject first-django
```

Activate the env if it isn't already..

```
workon first-django
```

Finally create a first-django distribution directory if mkproject hasn't already created one.

Okay lets get into woven.

1.9.2 Installing the packages

Install woven - making sure it correctly installs in the virtualenv.

```
pip install django pip install woven
```

It should also install fabric, and paramiko if they aren't already installed.

1.9.3 Creating the distribution and project

Create a project distribution and files using woven's `woven-admin.py` script. A distribution is all the files including media and other data relating to the project whereas the project is just the Django project. We're going to call the distribution something different from the actual project.

```
woven-admin.py startproject helloadmin woven@example.com --dist=first-django
```

`woven-admin.py` extends `django-admin.py` allowing us to use woven commands without adding woven itself to the `INSTALLED_APPS` setting.

You'll notice that it's a little different from `django-admin.py startproject` in that it creates a `setup.py` and a few other folders, and merges in the syncdb and creates a default superuser based on the user part of the optional email address (in this case 'woven') or the project name if you want one, though you can skip this behaviour with

`--nosyncdb`. As you'll discover, woven's ethos combines convenience with flexibility. Sometimes you need to get somewhere in a hurry, or just want a project you can throw away after using.

The setup.py is where woven gets your distribution name, project name and project version metadata which is used in deployments, it's just not used to package your project... yet. That will come when it morphs into setup.cfg and uses distutils2.

Woven's alternative to django's `startproject` also creates some sensible folders for media, static (app) content, templates, and database, and uses an alternative settings file from the django default. Nothing stopping you from changing it later if you want or you can also use `startproject -t` option to specify alternative starting templates for your project.

In the urls.py uncomment the admin lines then at the end of your urls.py we'll add a simple index page.:

```
urlpatterns += patterns('django.views.generic.simple',
    (r'^$', 'direct_to_template', {'template': 'index.html'}),
)
```

Finally in your templates folder create an index.html template file:

```
<!DOCTYPE html>

<html>
<head>
    <title>Hello admin</title>
</head>

<body>
Hello <a href="/admin/">admin</a>
</body>
</html>
```

To add a package to the PYTHONPATH you would normally install it using setup.py, but since we're developing, link the project folder `helloadmin` into your site packages. I recommend these pylink/pyunlink shell scripts: https://gist.github.com/21649. I don't recommend using add2virtualenv since it puts the setup.py in the path.

Change directory into the distribution folder (the one with setup.py in it) and run `woven-admin.py runserver`, opening http://127.0.0.1:8000/ in your browser.

`woven-admin.py` doesn't require you to set a DJANGO_SETTINGS_MODULE in the environment (though you can). Instead it infers your settings from the `setup.py` project name if you're somewhere under or in the setup.py folder, but you can also use `--settings` option as per normal.

If you have done everything right you should now see `hello admin` and be able to login to the django admin with the superuser. You're ready to deploy!

1.9.4 Setting up your server

Although woven does allow you to scale your deployment to multiple nodes, it doesn't support creating the initial image, so for now you'll need to purchase and startup an Ubuntu virtual machine separately.

Obtain an Ubuntu 10.04 or greater VM on the host of your choice with root and ssh access. For this tutorial I'm going to use a micro instance on Amazon EC2. See http://docs.amazonwebservices.com/AWSEC2/latest/GettingStartedGuide/ for how to get started on EC2.

1. Create a micro Ubuntu 10.04 x32instance ami-95c694d0 ebs on us-west (or similar) and a woven keypair.
2. Save the woven.pem key into your distribution folder
3. `chmod 400 woven.pem`

4. Create a security group with ports 22,80,10022 tcp open
5. Create and associate an elastic IP to the instance

Because django uses `example.com` as it's first site, we'll stick with that for this tutorial deployment. In your local `/etc/hosts` file add an entry for example.com pointing to the ip address of the ubuntu host (and on osx, run `dscacheutil -flushcache` just to be sure).

Just to be sure - check that you can login to your ec2 instance:

```
ssh -i woven.pem ubuntu@example.com
```

1.9.5 Setupnode

Now run setupnode to setup a baseline node for your intended user - in this case woven. The first thing woven will do is setup an additional user and if 'root' is the default user it will be disabled. It will also change the ssh port which is how woven determines that it has been pre-installed.

```
woven-admin.py setupnode -i woven.pem woven@example.com
```

or for non ec2, just woven-admin.py setupnode woven@example.com Answer according to your instance. Your root user may vary according to the provider. In our case with ec2 it is *ubuntu*.

Note: You might have noticed that setupnode uploads some files to the ubuntu `etc` directories. *Your node (host) configuration is stored in your project*. Woven allows you to define your own etc configuration files for ubuntu packages as standard templates in your project. If you want to modify the woven default templates you can copy them from the installed woven package into a woven folder in your projects templates folder like any other django app templates.

You can re-run setupnode at any time to alter your node configuration and update, upgrade and install new debian packages.

Now that your server is setup it's time to deploy our helloadmin project.

1.9.6 Deploy

Deploy early. Deploy often.

Lets collect the static media assets and deploy.

```
woven-admin.py collectstatic
woven-admin.py deploy woven@example.com
```

Deploy sets up a virtual environment on the server and deploys your sqlite3 database, django, and your project and all your dependencies into it. Sqlite3 is the default but again there's nothing stopping you dumping to a file and importing into Postgres or Mysql.

Everything in a deployment is versioned right down to the web configuration files. The only thing that isn't versioned is your database and MEDIA_ROOT. If you get errors, from misconfiguration or package installs, you can just fix your issue and run it again until it completes and activates your environment.

Note: Versions are critical to woven, and how woven differs from most deployment tools. Woven deploys a separate virtualenv just like the one we created earlier for *each* version of your distribution. This means you don't destroy an existing working environment when you deploy a new version. You could use this feature to test different features, or simply to rollback from a failed release. Not that you'll ever have a failed release. Ever.

You'll also notice woven has created a pip `requirements.txt` file and a `sitesettings` folder with a setting file inside. Requirements are your pip requirements for the project. The sitesettings.settings.py will import and override your local settings file on the node.

1.9.7 Patch

Of course mistakes are made, but to avoid stupidity and overwriting a working installation you cannot re-deploy the same version of your project with deploy (though the `--overwrite` option will do the trick if you're desperate). To get around having to deploy a new version for small changes you can run:

```
woven-admin.py patch woven@example.com
```

This will update existing files in your project, media and webserver configurations. It won't delete any files or update any dependencies. To update dependencies to a new library version you should increase your setup.py version and re-run deploy.

Patch can also just upload a specific part of your project using a subcommand. For example to just patch your webconf files:

```
woven-admin.py patch webconf woven@example.com
```

The different subcommands are `project|static|media|templates|webconf`

1.9.8 Where to now

If you want to work directly on the server (perhaps you need to debug something in staging) you can `ssh woven@example.com -p10022` into your host and type:

```
workon helloadmin
```

This will use virtualenvwrapper to activate your current virtualenv and drop you into the project sitesettings manage.py directory. A convenience manage.py is provided to run ./manage.py from there on the first site.

Of course installing packages from a requirements file can be problematic if pypi or a particular site is down . Make use of the `woven-admin.py bundle` command. This will use pip to bundle all the requirements into a dist folder in the distribution for deploy command to use.

We also haven't covered in this tutorial features such as roles, integrated South migrations and multi-site creation with `startsites`. Have a read of the woven django management *Management Commands* for more.

1.10 Management Commands

Commands that run on a host require a hoststring, role or have HOSTS or ROLEDEFS defined in your settings.

As per fabric a hoststring can be username@hostname, or use just the hostname or ip address. If no username is defined then woven will use the current user or settings.HOST_USER. You never need to set the port. It will always use port 10022 though another port can be defined using the HOST_SSH_PORT setting.

A common and recommended deployment pattern is to separate out staging servers from production servers. To do this in woven you would define your hoststrings in the ROLEDEFS settings (like fabfile roledefs). For example to have one staging server and two production servers you could define:

```
ROLEDEFS = {'staging':['user@192.168.188.10'], 'production':['user@host1.example.com', user@host2.exa
```

You would then use the role in place of the hoststring e.g. `woven-admin.py deploy staging`

1.10.1 startproject

Alternative to Django startproject. Creates a distribution folder as well as the project folder. You can also use your own alternative project layout template.

Usage:

`woven-admin.py startproject [project_name] [options]`

Options:

`-t --template` path to an alternative template directory.

`-d --dist` an alternate distribution name for the project. Defaults to the project name.

`--noadmin` don't uncomment admin.

1.10.2 setupnode

Setup Ubuntu host[s]. By default this will just setup a baseline host for django deployment but it can be used to setup other types of host such as postgresql nodes and varnish.

By defining ROLEDEFS in your settings you define packages for those hosts in the ROLE_PACKAGES settings. For example to setup a postgresql server you might define a role ROLEDEFS = {'database':['woven@db1.example.com']}, then set ROLE_PACKAGES = {'database':['postgresql']}, and finally set the firewall to ROLE_UFW_RULES = {'database':['allow tcp/5432']}. Finally postgresql configuration can be set in the project TEMPLATES_DIR woven/etc subdirectory.

Basic Usage:

`` ``woven-admin.py setupnode [hoststring] [options]`` ``

Lets go through what this actually does:

1. Creates the new *user* and disables the *root* user
2. Changes the default ssh port to 10022
3. Uploads your public ssh-key
4. Restricts ssh login to the ssh-key and adds a few other restrictions
5. Adds additional sources *universe* to sources.list
6. Updates and upgrades your packages
7. Installs UFW firewall
8. Installs a baseline of Ubuntu packages including Apache, Nginx, and mod-wsgi
9. Install any etc templates/files sourced from the woven or project woven/etc template directories
10. Sets the timezone according to your settings file

1.10.3 bundle

Pip bundle your requirements into pip zip bundles for efficient deployment

`woven-admin.py bundle`

1.10.4 deploy

Deploy your project to host[s] run syncdb and activate

Basic Usage:

`woven-admin.py deploy [hoststring] [options]`

options

The `--overwrite` option will remove and re-deploy the entire project for that version. By default deploy will never overwrite an existing version of your project.

South migration options

deploy integrates with south if it is installed. By default *all* migrations are run.

`-m --migration` Specify a specific migration to run (see South documentation)

`--fake` Fake a South migration (see South documentation)

`--nomigration` Do not migrate

`--manualmigration` Manage the database migration manually. With this option you can drop out of the current deployment to migrate the database manually, or pause the deployment while migrating in a separate shell. To migrate the database you could login to your host and then run `workon [yourproject-version]` to drop into the new versions environment and migrate your database using south, then logout and re-run deploy or continue the existing deploy.

The deploy command does the following:

1. For your first deployment it will deploy your development sqlite database (if it exists)
2. Create a virtualenv for the distribution version
3. Install django. By default it will install the local version. You can set a pip requirements string DJANGO_REQUIREMENT in your settings.py if you want svn trunk or some other specific version.
4. Install dependencies from one or more requirement req* files. eg. req, requirements.txt etc. If one doesn't exist then it will create one locally and add woven in it by default.
5. Creates a local sitesettings folder and a settings file for your server settings.py if it doesn't already exist. You can see how woven lays out your project on the server in the sitesettingssettings.py file.
6. Deploys your project to the virtualenv on the server
7. Deploys your root (shortest path) TEMPLATE_DIR into a templates directory on the server.
8. Deploys admin media or STATIC_ROOT setting (if you use django-staticfiles) into a virtualenv static directory.
9. Deploys anything at MEDIA_ROOT into a non-virtualenv public directory.
10. Deploys your wsgi file into a virtualenv wsgi directory as settings.wsgi
11. Renders your apache and nginx templates and deploys them into the sites-available with the version in the name.
12. Stops the webservices
13. Syncs the database
14. Runs South migrate if you have South installed
15. Symlinks the webserver conf versions into sites-enabled
16. Symlinks the project virtualenv version to the active virtualenv.
17. Starts the webservices

1.10.5 patch

Patch the current version of your project on host[s] and restartreload webservices Includes project, web configuration, media, and wsgi but does not pip install

Basic Usage:

`woven-admin.py patch [subcommand] [hoststring] [options]`

You can just patch a part of the deployment with a subcommand.

The possible subcommands are:

```
project, templates, static, media, wsgi, webconf
```

Example:

`woven-admin.py patch media woven@host.example.com`

1.10.6 activate

Activate a project version

Usage:

`woven-admin.py activate version [options]`

Example:

`woven-admin.py activate 0.1 woven@host.example.com`

1.10.7 node

Run a no arguments management command on host[s]. You can supply command options through the –options option –options="[option ...]"

Basic Usage:

`woven-admin.py node command [hoststring] [options]`

Example:

`woven-admin.py node flush woven@host.example.com --options="--noinput"`

1.10.8 startsites

Deploy webconf for the new sites and create a new user `site_n` where n is the SITE_ID of the new site(s).

Within Django sites are created on the database but use the SITE_ID in the settings file to designate which site is loaded. This command does not create the sites in the database but merely creates and deploys the configuration files needed to serve them.

Basic Usage:

`woven-admin.py startsites [hoststring] [options]`

1.11 Settings

Woven has a number of configuration options that can be set in your project's Django settings.py. They are all optional.

```
#List of hoststrings to setup on as per Fabric.
HOSTS = [] #eg ['woven@example.com','example.com','10.0.0.1']
#You can group collections of servers instead of HOSTS as per fabric
ROLEDEFS = {} #This would be used instead of HOSTS  eg {'staging':['woven@example.com']}
#The ssh port to be setup. We change the port for security
HOST_SSH_PORT = 10022 #default
#User can be defined here instead of in the hosts/roledefs
HOST_USER = ''
#Since password login will normally be disabled you can define it here
#for just ssh key security per host
HOST_PASSWORD = ''

#As per fabric KEY_FILENAME option to specify a path to an ssh key to use
SSH_KEY_FILENAME  = ''

#The first setup task is usually disabling the default root account and changing the ssh port.
ROOT_USER = 'root', #optional - mostly the default administrative account is root
DISABLE_ROOT = False, #optional - disable the default administrative account
ROOT_PASSWORD = '', #optional - blank by default
#The default ssh port, prior to woven changing it. Defaults to 22
DEFAULT_SSH_PORT = 22 #default
DISABLE_SSH_PASSWORD = #optional - setting this to true will disable password login and use ssh keys
#Firewall rules (note HOST_SSH_PORT/tcp is always allowed)
ENABLE_UFW = True #default - setting this to false will disable UFW
#Any rule(s) defined will overwrite the default
UFW_RULES = ['allow 80,443/tcp'] #default - see Ubuntu UFW for more details about rules
ROLE_UFW_RULES = {} # eg {'postgresql':['allow 5432/tcp']}

#The default ubuntu packages that are setup. It is NOT recommended you overwrite these
#use ROLE_PACKAGES if you want to define all your packages
HOST_BASE_PACKAGES = [
    'subversion','git-core','mercurial','bzr', #version control
    'gcc','build-essential', 'python-dev', 'python-setuptools', #build
    'apache2','libapache2-mod-wsgi', #wsgi server
    'nginx', #webserver
    'python-imaging', #pil
    'python-psycopg2','python-mysqldb','python-pysqlite2'] #default database drivers

#Package notes:
#If you define gunicorn in your extra packages then apache and mod-wsgi will not be
#installed, or will be removed. psycopg2 or mysqldb will only be installed
#if they are required by your DATABASES engine settings.

#Put any additional packages here to save overwriting the base_packages
HOST_EXTRA_PACKAGES = []

#define a list of repositories/sources to search for packages
#Current just handles Personal Package Archives (PPAs)
LINUX_PACKAGE_REPOSITORIES = [] # eg ['ppa:bchesneau/gunicorn']

#Role packages give you complete flexibility in defining packages with ROLEDEFS.
#By default any role that does not have role packages defined installs the HOST_BASE_PACKAGES + EXTRA
ROLE_PACKAGES = {} #eg ROLE_PACKAGES = {'postgresql':['postgresql']}
```

```
#Apache list of modules to disable for performance and memory efficiency
#defaults to the following:
APACHE_DISABLE_MODULES=['alias','auth_basic','authn_file','authz_default','authz_groupfile',
                          'authz_user','autoindex','cgid','dir',
                      'setenvif','status'],
#Virtualenv/Pip
DEPLOYMENT_ROOT = ''# defaults to /home/$USER.

PIP_REQUIREMENTS = [] # list of text pip requirements files (not pybundles). Defaults to any file in
# Note: Woven will also look for zip files matching the requirements in the dist directory.
#you can use the bundle management command to create these.

PROJECT_APPS_PATH = '' #a relative path from the project package directory for any local apps. See al

#Application media - as per build_static app
STATIC_URL = '' #by default this is set to the ADMIN_MEDIA_PREFIX
STATIC_ROOT = '' #by default this gets set to the admin media directory if admin is used

#Database migrations
MANUAL_MIGRATION = False #Manage database migrations manually
```

1.12 Conventions

Woven will use the following conventions to layout your project on the target host, and uses some basic coventions on the local development machine.

In the following examples we are deploying a distribution `example_distribution` with the django project `example_project` to a site `example.com`

1.12.1 Setup.py

A setup file must be defined in your distribution.

setup.py:

```
from distutils.core import setup

setup(name='example_distribution', #This is what your virtualenvs will be called
      version='0.1',
      packages=['example_project'],
      )
```

1.12.2 Project Development Layout

While woven tries to be agnostic about your project layout there are some conventions.

```
distribution folder
    |--dist (a dist folder will be created if you *bundle* requirements)
    |    |--requirements.zip (bundle will zip up python packages here with matching req*.txt file nar
    |--setup.py (a minimal setup.py is required with name, version, and packages defined)
    |--requirements.txt (a requirements.txt will be created if one doesn't exist)
    |--req2 (extra requirements can be defined prefixed with 'req')
    |--example_project (the standard django startproject)
             |--__init__.py
```

```
          |--deploy.py (optional hooks for custom setupnode or deploy functions)
          |--manage.py
          |--settings.py (global settings & local development settings)
          |--urls.py
          |--sitesettings (will be created if it doesn't exist)
                  |--__init__.py
                  |--settings.py (the sites specific setting overrides)
                  |--subdomain_settings.py (a site subdomain with the same SITE_ID)
                  |--manage.py (a convenience for running on node against site settings.py)
          |--local_apps (you can define an optional PROJECT_APPS_PATH in settings that will hold a
                  |--app1
                       |--deploy.py (hooks can be at app level as well)
                  |--...
 |--media (actual location as defined in settings)
 |    |--(any user content here)
 |--static  (actual location as defined in settings)
 |    |--(any application media here)
 |--templates (actual location as defined in settings)
     |  #woven comes with a number of default templates in the distribution
     |  #you can copy them and override them in your template directory.
     |--woven
          |--etc (optional - copy and override from woven package)
              |--apache2
                  |--ports.conf
              |--init.d
                  |--nginx
              |--nginx
                  |--nginx.conf
                  |--proxy.conf
              |--ufw
                  |--applications.d
                      |--woven_project (optional - see notes*)
              |--ssh
                  |--sshd_config
          |--django-apache-template.txt (sites conf)
          |--django-wsgi-template.txt
          |--maintenance.html (for when your site is being deployed)
          |--nginx-template.txt (sites conf)
          |--requirements.txt
          |--sitesettings.txt (default sitesetting)
          |--ufw.txt (ssh firewall rules)
          |--ufw-woven_project.txt (project firewall rules)
     |--[template.html] (global projecttemplates here)
```

Notes:

etc templates

Templates in the etc folder are uploaded using the following rules:

- templates are uploaded from the project directory first, and if they don't exist the woven package installed templates as per the standard django template loader
- If an etc package subdirectory exists on the node (eg apache2), all templates in the folder are uploaded.
- If an etc subdirectory is not package related (eg init.d) the template will only be uploaded to overwrite an existing template.

etc templates are uploaded if the template has changed or if the packages on the node change.

UFW firewall rules

UFW can use app templates defined in the */etc/ufw/applications.d* directory. Woven uploads *ufw.txt* as *woven* in this directory to set the SSH firewall rules.

A firewall rule for all nodes can be defined at UFW_RULES or exclusively for a role at ROLE_UFW_RULES. A rule is something like *allow 5432/tcp*. See UFW documentation for rule syntax. Removing a rule will delete it from the firewall next time setupnode is run.

1.12.3 Project Deployment Layout

Within the root folder on the node are the following:

```
~/.staging (all rsynced files are staged here before copying to final destination for network efficie
~/.pip (pip installation logs)
 |  |--cache (Pip will cache packages here)
 |  |--src (pip will store any editable source repositories here)
~/--database (for sqlite if it is used)
 |   |--example_project.db (will always be deployed as the [project-name].db)
 |--env (The root directory for all virtual environments)
     |--example_distribution (symlink to the current virtualenv version)
     |--example_distribution-0.1 (The virtualenv root for this version)
         |--bin
         |--dist
              |--requirements.pybundle
         |--include
         |--lib
         |--project
             |--example_project (package directory - symlinked to site-packages)
                 |--manage.py (your development manage.py)
                 |--settings.py (global & dev specific settings)
                 |--sitesettings (site local setting files)
                         |--__init__.py
                         |--manage.py (you run this on the node)
                         |--settings.py (primary settings file for nodes)
         |--templates (your project templates go here)
         |--static (admin and other app media)
         |--wsgi (web server scripts go here including modwsgi python file)
              |--settings.wsgi (for modwsgi)
    |--example_distribution-0.2 (next release version - as above)
 ...
 |--log (symlinks to /var/log)
 | Another media directory for files that in the user domain (MEDIA_URL) rather than required for the
 |--public  (for single domain deployments any project media goes here if you are hosting media local
```

1.12.4 Apache & Nginx Configuration files

/etc/apache2/sites-available/ By convention the configuration file will be saved using the domain name as follows.

/etc/apache2/sites-available/example_com-0.1.conf

Nginx for media is done the same way

1.12.5 Server-side State

Woven keeps track of server state and other housekeeping functions using the

/var/local/woven/ directory

Currently state is stored as a filename with or without content.

Backups of configuration files are stored at

/var/local/woven-backup

1.13 Troubleshooting

Stuff happens.

Adding –verbosity=2 to setupnode or deploy will show the debug view of fabric. This is more than you need but will help in diagnosing opaque error messages.

setupnode hangs on upgrade

Most likely some package in the apt-get upgrade process is incorrectly asking for user input.

do not kill the process at the client end

This will most likely leave your Ubuntu package database in a broken state.

The best solution is to log directly onto the host without killing the hung setupnode and then run sudo apt-get upgrade directly on the host. It will need you to dpkg configure -a and possibly remove the /var/lib/dpkg/updates if you are still having issues.

ERROR: There was an error running django-admin.py on the node

By deploying early and often it will be easier to diagnose misconfiguration issues. Usually this error will mean that django can't initialize your project due to import or other related issues, due to a particular app or code within your project. In the event it is a particular version of one of your requirements you can overwrite an existing installation by running `woven-admin.py deploy --overwrite` which will wipe your existing failed deployment.

1.14 Changelog

1.14.1 Release 0.8.0 (19-Dec-2010)

** Changes from 0.7**

- A woven-admin.py script is now the preferred way to run django commands. Woven no longer needs to be in the INSTALLED_APPS setting, and woven no longer installs fabric or python-paramiko on the node.
- Removed the DJANGO_REQUIREMENT setting. Django now gets added to the requirement file instead.

New Features

- The woven-admin.py script replaces the need to use the django-admin.py or project manage.py script. It can automatically pick up the settings from the setup.py file or be overriden with the usual –settings option. It injects itself into the INSTALLED_APPS for the execution of woven django commands so that woven will 'appear' to be installed.
- A custom startproject command creates a much more complete project layout with distribution parent folder, setup.py and some sensible default settings to get started quickly. It also has the option of using your own template directory to source a starting layout.

1.14.2 Release 0.7.0 (8-Dec-2010)

Changes from 0.6

- Multi-site functionality has changed dramatically. Instead of using one wsgi file and settings file per domain, woven now uses dynamic SITE_IDs in a single settings file based on the OS user, and process groups in modwsgi. Subdomains with the same SITE_ID can also be catered for by prefixing the settings filename. For example, if the SITE_ID=1 matches example.com, a settings file for the subdomain admin.example.com on the same SITE_ID would be admin_settings.py
- The ubuntu module has become linux. Woven may be compatible with other debian based distributions, and it would be nice to add support for redhat if someone wants to. I don't imagine there is too much difference beyond apt/yum (and ufw) between lsb distros for the purposes of using woven.
- run_once_per_host_version has changed name to run_once_per_version and another decorator has been added run_once_per_node. Both have moved to new decorators module.

New Features

- Implemented hooks. Can define a deploy.py with post_setupnode, post_install_package, or post_deploy functions. The deploy.py can be at project level or app level.
- Added a DISABLE_APACHE_MODULES setting to disable a number of unneeded modules on setup.
- Can add Personal Package Archives (PPA's) to a LINUX_PACKAGE_REPOSITORIES setting
- Basic support and template for gunicorn added, and auto adds a PPA for gunicorn
- Backups of configuration files don't pollute the package directories anymore.
- Auto-uninstalls packages that are no longer listed in settings
- Auto-uninstalls or doesn't install Apache modwsgi if you use gunicorn.
- Only uploads templates if they have changed or if any packages are installed
- only installs postgres/mysql db adaptors if your project uses them.

Bug fixes

- fix per role/host firewall rules and package installation
- lots of other small improvements..

1.14.3 Release 0.6.1 (26-Nov-2010)

- Fix UFW rules
- Don't try and restart webservers if they don't exist.
- Fixed problem with upload_etc logic
- added very basic templates for postgresql
- don't upload local_settings.py

1.14.4 Release 0.6 (22-Nov-2010)

Changes from 0.5

- Bundles are now just .zip files instead of .pybundles. Just rename any existing .pybundles or re-run bundle
- ROOT_DISABLED setting has been named DISABLE_ROOT instead.

- changed the name of the function deploy_public to deploy_media

New Features

- added new setting PROJECT_APPS_PATH for local apps, so that you can put an apps folder under your project folder.
- can now name your distribution differently from your project name
- –overwrite option in deploy to completely remove an existing virtualenv
- handles simple multi database, multi site migrations
- added new setting SSH_KEY_FILENAME which maps to fabric's KEY_FILENAME
- default webconf templates now listen on any ip (not fixed to one)
- can set ufw port rules in a template *woven_project*
- added ENABLE_UFW setting - defaults to true

Bug fixes

- fix an issue where you had to name your database after your project
- fix an issue when there is no files in a directory to deploy
- corrected the sitesettings template MEDIA_ROOT and STATIC_ROOT paths

1.14.5 Release 0.5.3 (9-Nov-2010)

- fix missing dist directory
- fix trailing / in manifest

1.14.6 Release 0.5.2 (7-Nov-2010)

- fix missing import

1.14.7 Release 0.5.1 (25-Aug-2010)

- fix to setupnode sshd_config rollback
- fix issue where setupnode fails if TEMPLATE_DIRS has not been set.

1.14.8 Release 0.5.0 (16-Aug-2010)

Note: To upgrade from 0.4 to 0.5 you should move your first [your_project].sitesettings.domain.py to [your_project].sitesettings.settings.py and create a new [your_project].sitesettings.domain.py that just imports [your_project].sitesettings.settings

- changed the [project_name].sitesettings.settings file to be the primary settings file for all non-dev sites
- enable hosts to be setup with different packages through the ROLEDEFS, ROLE_PACKAGES, and ROLE_UFW_RULES settings.
- added a maintenance.html template for nginx
- added a default deny all nginx conf to setupnode

- domains are now determined by dumping sites from the 1st database in the host list being executed against. If the project hasn't deployed we determine from the hostname or ask for it.
- simplified deployment_root and added back in a django setting to override it.
- added –noprofile to env.shell to make peace with virtualenvwrapper
- removed –overwrite option from setupnode
- fixed an issue with syncdb & migrate using the wrong settings file
- changed the name of function deploy_webservers to deploy_webconf
- setupnode now starts/restarts apache2 & nginx at the end of setup

1.14.9 Release 0.4.0 (10-Aug-2010)

Note: This release is backwards incompatable with earlier releases. You will need to re-run setupnode and re-deploy your project.

- can now use ROLEDEFS to define roles to group node functionality and use them in commands. ie ./manage.py deploy staging
- moved logs back to /var/log/apache2 nginx etc and link into them instead
- moved almost all woven /etc templates files into a new woven/etc template directory.
- user can create their own woven/etc templates to upload any arbitrary /etc/ files into their corresponding directories on the host
- changed deployment_root to the users home directory to allow integration with virtualenvwrapper
- integrate with virtualenvwrapper. Can now run workon [projectname] to drop into the current version on the node
- added a convenience settings.py, manage.py to sitesettings. The settings.py just imports the first sites settings
- integrate with south for migrations, and added syncdb to activation
- added manage.py patch subcommand where subcommand is an individual part of the deploy process.
- removed unattended upgrades - due to unreliability
- added an modified nginx init.d conf - the default init.d doesn't work under some boot timing circumstances
- use nginx reload command instead of start stop
- symlink the project directory to site-packages

1.14.10 Release 0.3.1 (1-Aug-2010)

- fixed a failure where trying to disable apparmor
- shifted from apache2ctl to init.d for starting and stopping apache2
- fixed an issue with requirements files
- uses the first domain SITE_ID = 1 sitesettings for project settings

1.14.11 Release 0.3 (22-Jul-2010)

- Major api refactor. Moved away from classes to function with decorator pattern. Codebase should be much clearer now.
- abstracted out a generic `deploy_files` function into deployment module that uses rsync but is more useful than fabric rsync_project where the remote_dir is not the same as the local parent dir. Stages files for network efficiency, and can deploy specific patterns of files in a directory and render templates if needed.
- new decorator `run_once_per_host_version` and state functions simplify where a function may be called multiple times but needs only finish once per host and project version.
- The public api can be imported `from woven.api import *`
- Allow any host strings to be used instead of just ip addresses.
- Resolves the host string where an ip is needed for apache/nginx
- implements an activate command to activate to a specific project version (env + webserver conf etc)
- `bundle` command bundles the requirements files for efficient deployment
- added a template pip requirements file
- added a `node` command to run arbitrary django management commands on hosts

1.14.12 Release 0.2.1 (4-Jul-2010)

- Fixed issue with installation fabric dependency

1.14.13 Release 0.2 (3-Jul-2010)

- Added deploy and patch management commands

1.14.14 Release 0.1.1 (22-Jun-2010)

- Changed serverserver to setupnode

1.14.15 Release 0.1 (21-Jun-2010)

- Initial Release

CHAPTER

TWO

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*