Brent O'Connor's Docs

Release

Brent O'Connor

October 10, 2016

Contents

Brent O'Connor's personal web development and system administration documentation. You can view the docs on Read the Docs or view the source on Github. Please feel free to submit issues or make pull requests.

CHAPTER 1

Django Docs

1.1 Django-celery + Redis on OSX Lion

1.1.1 Installation and Setup

1. Install redis on OSX (10.7) Lion:

```
$ brew install redis
```

2. In the project and virtualenv I wanted to use django-celery in I installed the following:

```
$ pip install django-celery
$ pip install redis
```

3. Add `djcelery` to your `INSTALLED_APPS` in your Django `settings.py` file.
4. Added the following django-celery settings toward the end of my Django `settings.py` file.

```
BROKER_HOST = "localhost"
BROKER_BACKEND="redis"
REDIS_PORT=6379
REDIS_HOST = "localhost"
BROKER_USER = ""
BROKER_PASSWORD =""
BROKER_VHOST = "0"
REDIS_DB = 0
REDIS_CONNECT_RETRY = True
CELERY_SEND_EVENTS=True
CELERY_RESULT_BACKEND='redis'
CELERY_TASK_RESULT_EXPIRES =  10
CELERYBEAT_SCHEDULER="djcelery.schedulers.DatabaseScheduler"
```

Note: If you run several sites that use Celery you will want to increment the number for `REDIS_DB` and `BROKER_VHOST` setting by 1 for each new site. Example for the next site you add, you would want to change those settings to the following:

```
BROKER_VHOST = "1"
REDIS_DB = 1
```

5. In your local development settings file it might be good to add `CELERY_ALWAYS_EAGER = True`. This blocks the run tests (sans celery) that way you can test and develop easier.

6. Open a terminal window and start redis.

```
$ redis-server /usr/local/etc/redis.conf
```

7. Open another terminal window and start a celery worker server for testing.

```
$ python manage.py celeryd -l info
```

1.1.2 Example Task

- Add the following code in a `tasks.py` file in a folder for one of your apps that's in your `INSTALLED_APPS` in your Django `settings.py` file.

```
from celery.decorators import task

@task()
def add(x, y):
    return x + y
```

- Now you should be able to play around with Django-celery from the command line. Open another terminal window and do the following:

```
$ django-admin.py shell
>>> result = add.delay(4, 4)
>>> result.ready() # returns True if the task has finished processing.
False
>>> result.result # task is not ready, so no return value yet.
None
>>> result.get()   # Waits until the task is done and returns the retval.
8
>>> result.result # direct access to result, doesn't re-raise errors.
8
>>> result.successful() # returns True if the task didn't end in failure.
True
```

1.2 Django Celery Notes

1.2.1 Celery Task Query Alert Script

If you want to receive an email alert if your Celery task queue backed by Redis gets too high, you could add the following to a directory like, `/user/local/scripts/celery_queue_alert.py`.

```
#!/usr/bin/env python

import subprocess
import smtplib


EMAIL_HOST = 'localhost'
EMAIL_SUBJECT = "WARNING: Your celery task queue is too high"
EMAIL_TO = "user@gdomain.com"
EMAIL_FROM = "root@localhost"
EMAIL_TEXT = "Your celery task queue currently has %(task_queue_count)s tasks in the queue, which is

out = subprocess.check_output(['redis-cli', 'llen', 'celery'], stderr=subprocess.STDOUT)
```

```
task_queue_count = int(out.strip())
max_queue_count = 6

if task_queue_count >= max_queue_count:
    EMAIL_BODY = "\r\n".join((
        "From: %s" % EMAIL_FROM,
        "To: %s" % EMAIL_TO,
        "Subject: %s" % EMAIL_SUBJECT,
        "",
        EMAIL_TEXT % {'task_queue_count': task_queue_count, 'max_queue_count': max_queue_count})
    )
    email = smtplib.SMTP(EMAIL_HOST)
    email.sendmail(EMAIL_FROM, [EMAIL_TO], EMAIL_BODY)
    email.quit()
```

Then setup a cron job for it by adding the following to `/etc/cron.d/celery_queue_alert`.

```
*/5 * * * * root /usr/local/scripts/celery_queue_alert.py
```

1.3 Django Generic Class Based Views

The following is a PDF I recieved from Jeff Triplett that Jacob Kaplan-Moss that shows mixin and class based view relationships.

Mixin and Class Based View Relationships PDF

1.4 Ubuntu Server Setup Guide for Django Websites

This guide is a walk-through on how to setup Ubuntu Server for hosting Django websites. The Django stack that will be used in this guide is Ubuntu, Nginx, Gunicorn and Postgres. This stack was chosen solely from the reading I've done and talking to other Django developers in order to get their recommendations. This stack seems to be one of the latest "standard" stacks for Django deployment. This guide also assumes that you're familiar with Ubuntu server administration and Django. I needed an example site for this guide so I chose to use my Django Base Site which is available on Github.

I would also like to thank Ben Claar, Adam Fast, Jeff Triplett and Frank Wiles for their suggestions and input on this guide.

1.4.1 Step 1: Install Ubuntu Server

The version of Ubuntu I'm using for this guide is Ubuntu 11.10 64 bit Server. I've installed Ubuntu Server in a VirtualBox VM on my MacBook Pro which is currently running Mac OS X 10.7.2. During the installation of Ubuntu Server I answered the prompts with the following:

```
Language: English
Install Menu: Install Ubuntu Server
Select a language: English
Select your location: United States
Configure the Keyboard: No
Configure the keyboard: English (US)
Configure the keyboard: English (US)
Hostname: ubuntu-vm
Configure the clock: Yes
```

```
Partition disks: Guided - use entire disk and set up LVM
Partition disks: SCSI3 (0,0,0) (sda) - 21.5 GB ATA VBOX HARDDISK
Partition disks: Yes
Partition disks: Continue
Partition disks: Yes
Set up users and passwords: Brent O'Connor
Set up users and passwords: (Enter a username)
Set up users and passwords: ********
Set up users and passwords: ********
Set up users and passwords: No
Configure the package manager: <blank>
Configure taskse1: No automatic updates
Software selection: <Continue>
Install the GRUB boot loader on a hard disk: Yes
Installation complete: <Continue>
```

1.4.2 Step 2: Setup Port Forwarding

Under the settings for your VM in VirtualBox click on the "Network" tab and then click on the "Port Forwarding" button. Now click on the plus and add the following settings to setup port forwarding for web and ssh.

Name	Protocol	Host IP	Host Port	Guest IP	Guest Port
SSH	TCP		2222		22
Web	TCP		8080		80

1.4.3 Step 3: Install Software

Before you begin it might be a good idea to update your system clock:

```
$ sudo ntpdate time.nist.gov
```

Download lists of new/upgradable packages:

```
$ sudo aptitude update
```

OpenSSH

Since I like to connect to my servers using SSH the first thing I install is openssh-server:

```
$ sudo aptitude install openssh-server
```

Since you setup port forwarding in step 2, you should now be able to open up your Terminal and connect to your Ubuntu Server using the following:

```
$ ssh localhost -p 2222
```

Python Header Files

The Python header files are needed in order to compile binding libraries like `psycopg2`.

```
$ sudo aptitude install python2.7-dev
```

PostgreSQL

```
$ sudo aptitude install postgresql postgresql-server-dev-9.1
```

Make your Ubuntu user a PostgreSQL superuser:

```
$ sudo su - postgres
$ createuser --superuser <your username>
$ exit
```

Restart PostgreSQL:

```
$ sudo /etc/init.d/postgresql restart
```

Nginx

```
$ sudo aptitude install nginx
```

Git

```
$ sudo aptitude install git
```

1.4.4 Step 4: Setup a Generic Deploy User

The reason we are setting up a generic deploy user is so that if you have multiple developers who are allowed to do deployments you can easily add the developer's SSH public key to the deploy user's `/home/deploy/.ssh/authorized_keys` file in order to allow them to do deployments.

```
$ sudo useradd -d /home/deploy -m -s /bin/bash deploy
```

1.4.5 Step 5: Install an Example Site

Setup a virtualenv:

```
$ sudo apt-get install python-setuptools
$ sudo easy_install pip virtualenv
$ cd /usr/local/
$ sudo mkdir virtualenvs
$ sudo chown deploy:deploy virtualenvs
$ sudo su deploy
$ cd virtualenvs
$ virtualenv --no-site-packages example-site
$ exit
```

Note: I personally use and setup virtualenvwrapper on all my servers and local development machines so that I can use `workon <virtualenv>` to easily activate a virtualenv. This is why I put all my virtualenvs in `/usr/local/virtualenvs`.

Make a location for the example site:

```
$ cd /srv/
$ sudo mkdir sites
$ sudo chown deploy:deploy sites
$ sudo su deploy
$ cd sites
$ git clone git://github.com/epicserve/django-base-site.git example-site
$ cd example-site/
$ git checkout -b example_site 5b05e2dbe5
$ echo `pwd` > /usr/local/virtualenvs/example-site/lib/python2.7/site-packages/django_project_root.pt
$ mkdir -p static/cache
$ exit
$ sudo chown www-data:www-data /srv/sites/example-site/static/cache
$ sudo su deploy
```

Create the file `/srv/sites/example-site/config/settings/local.py` and add the following. Make sure to change the password and then save the file. I usually use a random string generator to generate a new password for each new Postgresql database and user:

```
from base import *

LOCAL_SETTINGS_LOADED = True

DEBUG = True

INTERNAL_IPS = ('127.0.0.1', )

ADMINS = (
    ('Your Name', 'username@example.com'),
)

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'example_site',
        'USER': 'example_site',
        'PASSWORD': '<enter a new secure password>',
        'HOST': 'localhost',
    }
}
```

Install the sites required python packages:

```
$ source /usr/local/virtualenvs/example-site/bin/activate
$ cd /srv/sites/example-site/
$ pip install -r config/requirements/production.txt
```

Create a PostgreSQL user and database for your example-site:

```
# exit out of the deploy user account
$ exit
$ createuser example_site -P
$ Enter password for new role: [enter the same password you used in the local.py file from above]
$ Enter it again: [enter the password again]
$ Shall the new role be a superuser? (y/n) n
$ Shall the new role be allowed to create databases? (y/n) y
$ Shall the new role be allowed to create more new roles? (y/n) n
$ createdb example_site -O example_site
```

1.4.6 Step 6: Daemonize Gunicorn using Ubuntu's Upstart

Create your Upstart configuration file:

```
$ sudo vi /etc/init/gunicorn_example-site.conf
```

Add the following and save the file:

```
description "upstart configuration for gunicorn example-site"

start on net-device-up
stop on shutdown

respawn

exec /usr/local/virtualenvs/example-site/bin/gunicorn_django -u www-data -c /srv/sites/example-site/c
```

Start the gunicorn site:

```
$ sudo start gunicorn_example-site
```

1.4.7 Step 7: Setup Nginx to proxy to your new example site

Create a new file `sudo vi /etc/nginx/sites-available/example-site.conf` and add the following to the contents of the file:

```
server {

    listen       80;
    server_name  localhost;
    access_log   /var/log/nginx/example-site.access.log;
    error_log    /var/log/nginx/example-site.error.log;

    location = /biconcave {
        return  404;
    }

    location  /static/ {
        root  /srv/sites/example-site/;
    }

    location  /media/ {
        root  /srv/sites/example-site/;
    }


    location  / {
        proxy_pass            http://127.0.0.1:8000/;
        proxy_redirect        off;
        proxy_set_header      Host             $host;
        proxy_set_header      X-Real-IP        $remote_addr;
        proxy_set_header      X-Forwarded-For  $proxy_add_x_forwarded_for;
        client_max_body_size  10m;
    }

}
```

Enable the new site:

```
$ cd /etc/nginx/sites-enabled
$ sudo rm default
$ sudo ln -s ../sites-available/example-site.conf
```

Start nginx:

```
$ sudo /etc/init.d/nginx start
```

1.4.8 Step 8: Test the new example site

While still connected to your Ubuntu server via SSH run the following, which should spit out the HTML for your site:

```
wget -qO- 127.0.0.1:80
```

Since you setup port forwarding in step 2 for web, you should also be able to open up your browser on your local host machine and pull up the website using the URL, http://127.0.0.1:8080.

CHAPTER 2

Git Docs

2.1 General Git Notes

2.1.1 Good Basic Global Git Config Settings

```
[color]
    diff = auto
    status = auto
    branch = auto
[user]
    name = Your Name
    email = yourname@gmail.com
[core]
    excludesfile = /Users/username/.gitignore
    editor = mate
[alias]
    st = status
    d = diff
    ci = commit -v
    cia = commit -v -a
    co = checkout
    cp = cherry-pick
    dci = svn dcommit
    l = log
    ll = log -p
    lm = log master..
    llm = log -p master..
    b = branch
```

2.1.2 Starting a Git repository

```
cd project/
git init                    # initializes the repository
git add -n .                # do a dry-run to see what file are going to be added so you can make sure
git add .                   # add those 'unknown' files
git rm --cached <file>... # use this command to remove any files that where added that you didn't war
git commit                  # commit all changes, edit changelog entry
```

2.1.3 Add the a Remote and Push

```
git remote add origin git@<server-domain-name>:<repository-name>.git
git push origin master
```

If you want to be able to automatically pull from your remote repository you need to run the following command.

```
git branch --set-upstream master origin/master
```

The previous command adds the following to your *.git/config* for your current project.

```
[branch "master"]
    remote = origin
    merge = refs/heads/master
```

2.1.4 Ignoring Files

To add files for just the current project use:

```
cd project/
touch .gitignore
echo "path/to/file" >> .gitignore
```

To add ignore rules for every project just add the *.gitignore* file to your home directory *~/*.

2.1.5 Ways to Remove a file that was added before committing

```
git rm --cached <file>
git reset HEAD <file>
```

2.1.6 Setup a remote repository

Set up the new bare repo on the server:

```
$ ssh myserver.com
$ mkdir /var/git/myapp.git && cd /var/git/myapp.git
$ git --bare init
```

Add the remote repository to your existing local git repository and push:

```
$ cd project/
$ git remote add origin ssh://myserver.com/var/git/myapp.git
$ git push origin master
```

2.1.7 Setup a git repository from a Subversion repository

```
git svn clone -s http://myserver.com/path/to/svn/repository repository
cd repository
git branch -a
git reset --hard remotes/trunk
```

2.1.8 Push Changes back up to SVN

Checkout a branch and make edits:

```
git checkout -b my_branch
```

Make your edits and commit changes:

```
git add .
git commit -m "Made some changed to the branch"
```

Update against any changes made to your svn repo, rebase and then commit changes to your svn repo:

```
git svn dcommit
```

2.1.9 Start and Track a New Remote Branch

Create the new remote branch:

```
git push origin master:newbranch
```

Setup the new remote branch locally so you can push and pull to the new remote branch:

```
git checkout --track -b newbranch origin/newbranch
```

2.1.10 Branch Management

List merged branches:

```
git branch -a --merged
```

Delete a local branch that's been merged:

```
git branch -d merged_branch_name
```

Delete remote branch that's been merged:

```
git push origin :merged_branch_name
```

2.1.11 Share Uncommitted Changes

To share uncommitted changes with someone or if you want to copy uncommitted changes from one git repository to another, first generate a patchfile from the git repository that has the changes you want to share.

```
git diff --no-prefix > patchfile
```

Copy the patchfile to the computer with the git repository that you want to apply the changes too and then run the following command.

```
patch -p0 < patchfile
```

Any untracked files will have to be copied manually.

2.2 Git SVN Notes

2.2.1 Setup a New SVN Branch that tracks with a Local Git branch

See the following for more docs, http://stackoverflow.com/questions/266395/git-svn-how-do-i-create-a-new-svn-branch-via-git

This is what worked for me:

```
$ git svn branch -m "Branch for new feature" new_feature
$ git co -b local_new_feature remotes/new_feature
```

2.2.2 Changing your Git svn commit url

Tried the following method that I originally got from, http://translate.org.za/blogs/wynand/en/content/changing-your-svn-repository-address-git-svn-setup and couldn't get it to work. I resorted to checking out the repository again and then ran *git clean -dXn* to see what files in the old copy that I needed to copy into the new copy of the repository.

1. Back up your git repo. If you make a mistake with repository rewriting you'll be in for a lot of fun.
2. `git gc`
3. `git filter-branch --msg-filter "sed 's|git-svn-id:  http://svn.wenworld.com|git-svn-id https://svn.wenatcheeworld.com|g' $(cat .git/packed-refs | awk '// {print $2}' | grep -v 'pack-refs')"`
4. `rm -rf .git/svn`
5. edit .git/config and change `http://svn.wenworld.com` in all the git-svn URLs to `https://svn.wenatcheeworld.com`
6. git svn rebase (to update your repo and to let the git-svn data be rebuilt)

2.3 Convert SVN Repositories to Git Repositories

This guide on how to convert an SVN repository to a git repository was mostly taken from John Albin Wilkins post on Converting a Subversion repository to Git.

2.3.1 1. Retrieve a list of all Subversion committers

```
$ svn log -q https://svn.example.com/repository_name | \
awk -F '|' '/^r/ {sub("^ ", "", $2); sub(" $", "", $2); \
print $2" = "$2" <"$2">"}' | sort -u > authors-transform.txt
```

That will grab all the log messages, pluck out the usernames, eliminate any duplicate usernames, sort the usernames and place them into a "authors-transform.txt" file. Now edit each line in the file. For example, convert:

```
username = username <username>
```

into this:

```
username = Firstname Lastname <username@example.com>
```

2.3.2 2. Clone the Subversion repository using git-svn

```
git svn clone [SVN repo URL] --no-metadata -A authors-transform.txt --stdlayout ~/temp
```

2.3.3 3. Convert svn:ignore properties to .gitignore

If your svn repo was using svn:ignore properties, you can easily convert this to a *.gitignore* file using:

```
cd ~/temp
git svn show-ignore -i trunk > .gitignore
git add .gitignore
git commit -m 'Convert svn:ignore properties to .gitignore.'
```

2.3.4 4. Push repository to a bare git repository

First, create a bare repository and make its default branch match svn's "trunk" branch name.

```
git init --bare ~/new-bare.git
cd ~/new-bare.git
git symbolic-ref HEAD refs/heads/trunk

cd ~/temp
git remote add bare ~/new-bare.git
git config remote.bare.push 'refs/remotes/*:refs/heads/*'
git push bare
```

You can now safely delete the *~/temp* repository.

2.3.5 5. Rename "trunk" branch to "master"

Your main development branch will be named "trunk" which matches the name it was in Subversion. You'll want to rename it to Git's standard "master" branch using:

```
cd ~/new-bare.git
git branch -m trunk master
```

2.3.6 6. Clean up branches and tags

git-svn makes all of Subversions tags into very-short branches in Git of the form "tags/name". You'll want to convert all those branches into actual Git tags using:

```
cd ~/new-bare.git
git for-each-ref --format='%(refname)' refs/heads/tags |
cut -d / -f 4 |
while read ref
do
  git tag "$ref" "refs/heads/tags/$ref";
  git branch -D "tags/$ref";
done
```

2.3.7 7. Move bare repository to central remote repository

Example of how to move your local bare repository to a gitolite repository:

```
$ mv new-bare.git repository_name.git
$ tar czvf repository_name.git.tar.gz repository_name.git/
$ scp repository_name.git.tar.gz remote_host:
$ ssh remote_host
$ tar xzvf repository_name.git.tar.gz
$ sudo chown -R git:staff repository_name.git/
$ cd repository_name.git/
$ sudo find . -type f -exec chmod go= {} \;  # remove group and world permissions
$ sudo find . -type d -exec chmod go= {} \;  # remove group and world permissions
$ sudo su
$ cd /Users/git/repositories/
$ rm -rf repository_name.git/
$ mv ~username/repository_name.git .
```

2.3.8 8. Clone new local copy

```
mv old-svn-copy old-svn-copy.backup
git clone git@remote_host:repository_name.git
```

List all unversioned files from your old local svn repository and copy them to the new local git repository:

```
cd old-svn-copy.backup
git clean -dXn  # Using this command because the old copy was a git-svn clone
cp example-file.txt ../repository_name/  # copy all files and directories from the list that you need
```

You can now move the local svn copy backup to your trash. It might be a good idea not to empty your trash until your sure everything is working correctly.

2.3.9 9. Done

CHAPTER 3

SVN Docs

3.1 SVN Notes

3.1.1 Creating a svn repository

Create the svn repository where all the files and different versions will be kept:

```
svnadmin create /www/svn/myproject
```

Add directories for your main trunk and tags, and branches:

```
sudo svn mkdir file:///www/svn/myproject/trunk \
file:///www/svn/myproject/branches \
file:///www/svn/myproject/tags \
-m "Adding standard subversion directory structure"
```

Double check your repository by listing what is in your repository:

```
svn list file:///www/svn/myproject/
```

Look at your repository info:

```
svn info file:///www/svn/myproject/
```

Import an existing project into your repository:

```
cd /www/myproject/
cd ..
svn import myproject/ file:///www/svn/myproject/trunk -m "Importing my project"
```

Double check your repository:

```
svn list file:///www/svn/myproject/trunk
```

Check out your repository to the working directory:

```
rm -rf /www/myproject/*
cd /www/myproject/
svn co file:///www/svn/myproject/trunk .
```

3.1.2 Working with your checked out copy

Add a file and commit it to the repository:

```
cd /www/myproject
touch my-test-file.txt
svn add my-test-file.txt
svn commit . -m "Added the my-test-file.txt"
```

Rename a file and commit changes:

```
cd /www/myproject
svn mv my-test-file.txt my-test-file2.txt
svn commit . -m "Renamed my-test-file.txt to my-test-file2.txt"
```

Delete a file and commit changes:

```
cd /www/myproject
svn rm my-test-file2.txt
svn commit . -m "Deleted my-test-file2.txt"
```

List the changes locally and in the repository:

```
svn status -u
```

Making tags or branches:

```
svn copy -m "Notes about the project, r<revision-number-before-change>"
file:///www/svn/myproject/trunk
file:///www/svn/myproject/tags/before-some-major-change
```

To switch your working copy to the new branch you just made:

```
svn switch \
file:///www/svn/myproject/tags/before-some-major-change
```

3.1.3 Merging changes from another branch or trunk

Before merging you should commit any of your current changes so that you can revert back to your current state if the merge doesn't go well.

Go to your working copy and then use the version numbers one above the one where you made the changes you want to merge in and then use the last revision number where you made the changes. Then put the path to repository you are merging from. After you run the following command it shows what changes it made to your working copy. Make sure it made the correct changes and if it did you can then commit those changes to the branch or trunk that your current working copy is checked out from.

Example:

```
svn merge -r 20:22 file:///www/svn/cms/trunk
```

If you mess up you can always run `svn revert --recursive .`

3.1.4 Using svn over ssh

If you want to checkout your repository from another computer you can use:

```
svn+ssh://hostname/path/to/repos instead of file:///path/to/repos/
```

3.1.5 Undoing a bad commit

If you make a commit and it's reverse number was 4 and you want to undo the bad commit you can do the following in your local copy.

```
svn merge -r 4:3 file:///www/svn/project/trunk
```

CHAPTER 4

System Administration

4.1 Common Linux Commands

4.1.1 Files Commands

list sub directories:

```
$ ls -la | grep ^d
```

To watch a file as it changes:

```
$ tail -f /path/to/file
```

To check what directories are taking the most space:

```
$ du -xm / | sort -n | tail -50
```

Check what devices are taking the most space:

```
$ ls -lrSh /dev | tail -50
```

Do a grep regex search:

```
$ find . -name "*.xml" | xargs grep -Pzo "(?s)<pub10>.*?</pub10>"
```

4.1.2 Computer Information

Print operating system name:

```
$ uname -a
```

On Ubuntu:

```
$ lsb_release -a
```

4.1.3 Network Commands

View tcp connection for an ip address:

```
$ tcpdump -x host 192.168.0.1
```

4.1.4 Common Tar Commands

Extract the contents of example.tar and display it's contents:

```
$ tar xvf example.tar
```

Extract and uncompress a compressed tar archive:

```
$ tar xzvf example.tar.gz
```

Create a tar archive of a directory:

```
$ tar -cf example.tar /path/to/example
```

Create a tar archive and of a directory and compress the directory:

```
$ tar -czvf example.tar.gz /path/to/example
```

View the contents of a tar archive:

```
$ tar -tvf example.tar
```

4.1.5 How to send other users messages from the terminal

```
$ write username tty
message text
CTRL+C
```

4.1.6 Process Commands

List all processes:

```
$ ps -ef
```

Another way to list all processes:

```
$ ps -aux
```

List Process Tree:

```
$ ps -axf
```

4.1.7 Search and Replace Text in Text Files

Search for all python files and replace foo with bar:

```
$ find . -name "*.py" | xargs perl -i.bak -wpe 's/\bfoo\b/bar/g'
```

Command to move your files back to the original if something went back with the search and replace:

```
$ for i in *txt;do cp ${i}.bak $i;done
```

Another way to move your backup files back to the original if something went bad with the search and replace:

```
$ find . -iname '*.txt' -exec mv {}.bak {} \;
```

4.1.8 Rename files in a batch

Rename files by number:

```
$ c=0; for i in *.jpg; do (( c++ )); mv "$i" "$c".jpg; done
```

Change a files extention:

```
$ for i in *.txt; do mv $i ${i/.txt/.doc}; done
```

4.2 Compiling Notes

To log configure output to a file:

```
$ ./configure > log.txt 2>&1
```

To view a files dynamically linkes libraries on OSX:

```
otool -L /Library/Python/2.5/site-packages/xapian/_xapian.so
```

4.3 Create a Private Signed Cert Example

Last time I created a privatly signed cert I did it this way:

```
$ cd /usr/share/ssl/
$ openssl req -config openssl.cnf -new -out /usr/local/ssl/certs/webmail.epicserve.com.csr
```

Answer the following prompts:

```
Enter PEM pass phrase: <enter something that is 4 chars>
Verifying - Enter PEM pass phrase: <re-enter pass phrase>
Country Name (2 letter code) [GB]: US
State or Province Name (full name) [Berkshire]: Kansas
Locality Name (eg, city) [Newbury]: Manhattan
Organization Name (eg, company) [My Company Ltd]: Epicserve
Organizational Unit Name (eg, section) []: Web Hosting
Common Name (eg, your name or your server's hostname) []: webmail.epicserve.com
Email Address []:
A challenge password []:
An optional company name []:
```

Run the following command:

```
$ openssl rsa -in privkey.pem -out /usr/local/ssl/private/webmail.epicserve.com.key
```

Answer the following prompts:

```
Enter pass phrase for privkey.pem: <enter the same pass phrase you enterned in the last step>
```

Then run the following:

```
$ openssl x509 -in /usr/local/ssl/certs/webmail.epicserve.com.csr \
-out /usr/local/ssl/certs/webmail.epicserve.com.crt \
-req -signkey /usr/local/ssl/private/webmail.epicserve.com.key \
-days 365
```

Restart Apache

4.4 Debian Notes

4.4.1 Debian Packages

Note: To get emails sent to you automatically on what updates are available for your system, install apticron.

The `dpkg` command

To list packages:

```
$ dpkg -l
```

To list files in a package:

```
$ dpkg -L <pkg_name>
```

The `apt-cache` command

To search all available packages:

```
$ apt-cache search <pkg_name>
```

To find out more about a potential package:

```
$ apt-cache showpkg <pkg_name>
```

The `apt-get` command

To update package database with latest avail versions:

```
$ apt-get update
```

To upgrade all installed packages to latest versions:

```
$ apt-get -u upgrade
```

Use this if it says the following packages where kept back:

```
$ apt-get dist-upgrade
```

To install a particular package:

```
$ apt-get install <pkg>
```

Uninstall package:

```
$ apt-get remove <pkg>
```

The `dpkg-reconfigure` command

To reconfigure a package if you config'd it wrong:

```
$ dpkg-reconfigure <pkg>
```

4.4.2 Debian Services

Notes on services taken from http://www.shallowsky.com/linux/debnotes.html.

Enable service at boot time. (chkconfig svc on)

```
$ update-rc.d svc defaults
```

Disable service at boot time. (chkconfig svc off) `-f` means force removal of the `/etc/rc.?` scripts while leaving the basic script in `/etc/init.d` (so you can change your mind later). `--purge` means remove the script from init.d.

```
$ update-rc.d -f svc remove
```

Enable service at boot time in the given runlevels, like `chkconfig svc --levels 2345 on`

```
$ update-rc.d svc start 20 2 3 4 5 . stop 20 0 1 6
```

4.4.3 Other Resources

- Debian notes: doing things the "Debian Way"

4.5 Downloading a Sequence of Files

Example of how to download `foo-01.txt` through `foo-17.txt`:

```
$ for i in $(seq -f "%02g" 1 17); do wget http://example.com/foo-$i.txt; done
```

4.6 Incron - Monitoring Linux File System Events

4.6.1 Install and Setup

```
$ sudo aptitude install incron
```

Add *root* to the config file */etc/incron.allow* to allow incron to run as root.

```
$ sudo echo "root" >> /etc/incron.allow
```

4.6.2 Example

The following is an example of how you would setup *Incron* to watch a directory for new files that have been uploaded and then trigger a script to run with the full path to the file that was just uploaded in the directory Incron is watching as the script's first command line arguement.

Run the following to edit the list of dirtories or files Incron is watching:

```
$ sudo incrontab -e
```

Add something similar to following:

```
/path/to/dirtory_to_watch IN_CLOSE_WRITE /path/to/script/my_script.py $@/$#
```

Save the file

4.7 IPtables Notes

To reset iptables:

```
$ iptables -t filter -P INPUT ACCEPT
$ iptables -t filter -P OUTPUT ACCEPT
$ iptables -t filter -P FORWARD ACCEPT
$ iptables -F
```

To list current chains:

```
$ iptables -L
$ iptables iptables -t nat -n -L
```

To load your iptables config file:

```
$ iptables-restore /etc/network/iptables
```

To save your current chains that are in memory:

```
$ iptables-save > /etc/network/iptables
```

4.8 Linux/Unix Security

Scan for open ports:

```
$ nmap ip_address
```

4.8.1 Resources

- Linux Security Administrator's Guide: Intrusion Detection

4.9 Monit Notes

Monit is a good service for sending alerts and restarting services based on simple rules.

4.9.1 Installing

```
$ sudo aptitude install monit
```

4.9.2 Setup Alerting

Add the following to the */etc/monit/conf.d/global_settings* file:

```
set mailserver localhost
set alert sysadmin@company.com

## Uncomment the following if you don't want alerts every time you reload or
## restart monit.
#
# check system localhost
#     alert sysadmin@company.com but not on { instance }
```

Reload monit *sudo service monit reload*

4.9.3 Setup Basic System Alerts

Add the following to the */etc/monit/conf.d/system_checks* file:

```
check system localhost
    if memory usage > 85% then alert
    if cpu usage (user) > 80% for 3 cycles then alert
    if cpu usage (system) > 80% for 3 cycles then alert

check filesystem rootfs with path /
    if space usage > 80% then alert
```

Reload monit *sudo service monit reload*

4.9.4 Monitoring Gunicorn Example

Add the following to the */etc/monit/conf.d/gunicorn* file:

```
check process gunicorn with pidfile /var/run/gunicorn_www.example.com.pid
    start program = "/sbin/start gunicorn_www.example.com"
    stop program  = "/sbin/stop gunicorn_www.example.com"
    if failed unixsocket /tmp/gunicorn_www.example.com.sock then start
    if totalcpu > 20% for 1 cycles then alert
    if totalmemory usage > 60% then restart
```

Reload monit *sudo service monit reload*

Note: Make sure you use *totalcpu* and *totalmemory*, because you want to make sure it totals the memory and cpu usage for all child processes.

Also, the name of the process doesn't seem to matter. In the above example I could have used "*check process django_gunicorn*" if I wanted instead of "*check process gunicorn*." The name you use is what will be used in alert emails.

4.10 Mount a Windows File Share From the CLI

The method that worked best for me:

```
$ sudo mount -t cifs -o username=username //server/share /home/username/share/
```

Another method that might work in another situation:

```
$ sudo mount -t smbfs -o username=username //server/share /mnt/share
```

4.11 MySQL Admin Notes

4.11.1 Database Administration

Backup all dbs:

```
$ mysqldump --all-databases | gzip > all_databases.sql.gz
```

Backup one db:

```
$ mysqldump --opt db_name | gzip > backup-file.sql.gz
```

Read a db dump back in to the database:

```
$ gunzip < backup-file.sql.gz | mysql db_name
```

Create a new Database:

```
$ mysqladmin -u root -p create databasename
```

Rename a database:

```
$ mysqldump -u root -p old_db_name > old_db_name.sql
$ mysqladmin -u root -p create new_db_name
$ mysql -u root -p new_db_name < old_db_name.sql
$ mysql -u root -p -e "DROP DATABASE old_db_name"
```

4.11.2 User Administration

Create a user for a database:

```
mysql> GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER, CREATE
TEMPORARY TABLES, LOCK TABLES
ON db_name.*
TO 'user_name'@'localhost' IDENTIFIED BY 'password';
```

Change the root password:

```
mysql> use mysql;
# The following is to list hostnames for root
mysql> select Host, User, Password FROM user WHERE User = root;
mysql> SET PASSWORD FOR 'root'@'hostname'=PASSWORD('my-password');
mysql> flush privileges;
```

4.12 Perl Search and Replace Examples

Find all text files then back them up by making a copy with the `.bak` extention and then search for `foo` and replace it with `bar`.

```
$ find . -name '*.txt' | xargs perl -i.bak -wpe "s/foo/bar/g"
```

Example using a perl positive lookup assertion to avoid errors if the option isn't found.

```
$ find . -name '*.php' | \
xargs perl -i.bak -wpe 's/include((?:_once)?)\(INCLUDES?\. ?['\''"]?([^'\''^"^\)]+)['\''"]?\)/include
```

4.13 Postgres Backup with Wal-e

The following are the steps I took to setup Wal-e 0.6.2 on Ubuntu 12.04.2 LTS and Postgres 9.1.9. After following the installation instructions, every minute Wal-e will make incremental backups to Amazon S3.

4.13.1 Installation

```
$ sudo apt-get install libevent-dev python-all-dev daemontools lzop pv postgresql-client
$ sudo pip install wal-e
$ umask u=rwx,g=rx,o=
$ mkdir -p /etc/wal-e.d/env
$ echo "secret-key-content" > /etc/wal-e.d/env/AWS_SECRET_ACCESS_KEY
$ echo "access-key" > /etc/wal-e.d/env/AWS_ACCESS_KEY_ID
$ echo 's3://some-bucket/directory/or/whatever' > /etc/wal-e.d/env/WALE_S3_PREFIX
$ sudo chown -R root:postgres /etc/wal-e.d
```

Added the following to the end of the file, */etc/postgresql/9.1/main/postgresql.conf*:

```
wal_level = archive
archive_mode = on
archive_command = 'envdir /etc/wal-e.d/env /usr/local/bin/wal-e wal-push %p'
archive_timeout = 60
```

Restart postgres:

```
$ sudo service postgresql restart
```

Make a base backup:

```
$ sudo -u postgres bash -c "envdir /etc/wal-e.d/env /usr/local/bin/wal-e backup-push \
/var/lib/postgresql/9.1/main"
```

4.13.2 Other Notes

You can use the following query (as superuser) to monitor how many segments are 'ready' for archiving but have not yet been sent (those are 'done'). [1]

[1] Thanks to Daniel Farina for the tip.

```
SELECT suffix, count(*)
    FROM (
        SELECT (regexp_matches(name, E'\[0-9A-F]+\.([^\.]*)$'))[1] AS suffix
        FROM pg_ls_dir('pg_xlog/archive_status') name
    ) AS matches
    GROUP BY suffix;
```

4.14 PostgreSQL Notes

4.14.1 Postgres Administration

Postgres Shell

Run the following to access the Postgres shell:

```
$ psql -d template1
template1=#
```

User Management

Create a user:

```
$ createuser dbusername
Shall the new role be a superuser? (y/n) n
Shall the new role be allowed to create databases? (y/n) n
Shall the new role be allowed to create more new roles? (y/n) n
```

Create a user with a password and no privileges:

```
$ createuser -DRSW dbusername
```

List all users:

```
template1=# select usename from pg_user;
```

Delete a user:

```
$ dropuser dbusername
```

Database Management

Create a database:

```
$ createdb dbname -O dbusername
```

Delete a database:

```
template1=# drop database <db_name>;
```

List all databases:

```
template1=# select datname from pg_database;
```

List all databases sizes:

```
template1=# SELECT pg_database.datname, pg_size_pretty(pg_database_size(pg_database.datname)) AS size
```

List the size of each table in a database:

```
db_name=# SELECT relname, reltuples, relpages * 8 / 1024 AS "MB" FROM pg_class ORDER BY relpages DES
```

4.14.2 Helpful Queries

List the number of concurrent connections:

```
template1=# select * from pg_stat_activity;
```

4.14.3 Sync a Production Postgres DB with a local development DB

Open up terminal and do the following:

```
$ ssh remote_host
$ pg_dump -U db_user db_name > db_name.sql
```

Open a new terminal tab for localhost and do the following:

```
$ dropdb db_name
$ createdb db_name -O db_user
$ psql -U db_user -d db_name < db_name.sql
```

Switch back to the remote host tab and do the following:

```
$ rm ads.sql
```

4.14.4 Extracting a single database from a pg_dumpall PostgreSQL dump

If you need to restore a single database from a backup where you've backed up all your databases using the `pg_dumpall` command, then you can use the following script to extract a single database from your `pg_dumpall` file. I found this script on Mads Sülau Jørgensen's blog.

Warning: This hasn't personally been tested yet.

Usage:

```
$ postgresql_dump_extract.sh <dump_file> <db_name> > database.sql
```

Script:

```
#!/bin/sh

if [ $# -ne 2 ]
then
    echo "Usage: $0 <postgresql sql dump> <db_name>" >&2
    exit 1
fi

db_file=$1
db_name=$2
```

```
if [ ! -f $db_file -o ! -r $db_file ]
then
    echo "error: $db_file not found or not readable" >&2
    exit 2
fi

grep -b "^\connect" $db_file | grep -m 1 -A 1 "$db_name$" | while read line
do
    bytes=`echo $line | cut -d: -f1`

    if [ -z "$start_point" ]
    then
        start_point=$bytes
    else
        end_point=$bytes
    fi
done

if [ -z "$start_point" -o -z "$end_point" ]
then
    echo "error: start or end not found" >&2
    exit 3
fi

db_length=`expr $end_point - $start_point`

tail -c +$start_point $db_file | head -c $db_length | tail +3
```

4.14.5 Common Postgres Shell Commands

View table structure:

```
postgres=> \d <table name>
```

To add a column to a table:

```
ALTER TABLE products ADD COLUMN description text;
```

To remove a not-null constraint:

```
ALTER TABLE products ALTER COLUMN product_no DROP NOT NULL;
```

To remove any default value, use:

```
ALTER TABLE products ALTER COLUMN price DROP DEFAULT;
```

Output a SELECT query to a csv file:

```
open terminal
su postgres
psql <db_name>
\f ','
\a
\t
\o outputfile.csv
select ..... (your sql select statement)
\o
\q
```

4.15 Redhat Notes

4.15.1 Redhat Packages

List any packages that need updating:

```
$ up2date --list
```

Search for a package already installed:

```
$ rqm -qa | grep foo
```

List all packages available for download:

```
$ up2date --showall
```

Show all packages available that aren't installed:

```
$ up2date --show-available
```

Install a package:

```
$ up2date <package-name>
```

List package contents:

```
$ rpm -lpq <package-name>
```

4.15.2 Redhat Services

List services and their runlevels:

```
$ chkconfig --list
```

Turn a service on or off at boot:

```
$ chkconfig <service> on|off
```

4.16 Redis Cheat Sheet

Connect to redis:

```
$ redis-cli
```

Select a DB:

```
redis 127.0.0.1:6379> SELECT 4
```

List all keys:

```
redis 127.0.0.1:6379> KEYS *
```

Delete a key:

```
redis 127.0.0.1:6379> DEL my_key_name
```

Script to list all key sizes:

```
#!/usr/bin/env bash

# This script prints out all of your Redis keys and their size in a human readable format
# Copyright 2013 Brent O'Connor
# License: http://www.apache.org/licenses/LICENSE-2.0

human_size() {
        awk -v sum="$1" ' BEGIN {hum[1024^3]="Gb"; hum[1024^2]="Mb"; hum[1024]="Kb"; for (x=1024^3; x
}

redis_cmd='redis-cli'

# get keys and sizes
for k in `$redis_cmd keys "*"`; do key_size_bytes=`$redis_cmd debug object $k | perl -wpe 's/^.+seria

# sort the list
sorted_key_list=`echo -e "$size_key_list" | sort -n`

# print out the list with human readable sizes
echo -e "$sorted_key_list" | while read l; do
    if [[ -n "$l" ]]; then
        size=`echo $l | perl -wpe 's/^(\d+).+/$1/g'`; hsize=`human_size "$size"`; key=`echo $l | perl
    fi
done
```

4.17 Redis Notes

If you're having issues with Redis crashing, *check the logs* (/var/log/redis/redis-server.log) because you might find helpful notes like ...

```
# WARNING overcommit_memory is set to 0! Background save may fail under low memory
# condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf
# and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to
# take effect.
```

Doing what it suggested fixed my issue.

4.18 Sed Examples

Examples of how to use sed to search and replace text inside of text files.

```
$ find . -name '*.php' | xargs sed -i "s/\$cfg\['siteurl'\]/SITE_URL/g"
$ find . -name '*.php' | xargs sed -i 's/SITE_URL}\//SITE_URL ."/g'
$ find . -name '*.php' | xargs sed -i "s/\$cfg\['public_site'\]/PUBLIC_SITE_URL/g"
$ find . -name '*.php' | xargs sed -i 's/"{INCLUDES}/INCLUDES . "/g'
$ find . -name '*.php' | xargs sed -i "s/\$cfg\['photo_batch_upload_dir'\]/PHOTO_BATCH_UPLOAD_DIR/g"
$ find . -name '*.php' | xargs sed -i "s/\$cfg\['photo_thumbs'\]/PHOTO_THUMBS/g"
$ find . -name '*.php' | xargs sed -i "s/\$cfg\['photo_large'\]/PHOTO_LARGE/g"
$ find . -name '*.php' | xargs sed -i "s/\$cfg\['photo_orig'\]/PHOTO_ORIG/g"
$ find . -name '*.php' | xargs sed -i "s/\$cfg\['photo_thumbs_url'\]/PHOTO_THUMBS_URL/g"
$ find . -name '*.php' | xargs sed -i "s/\$cfg\['photo_medium_url'\]/PHOTO_MEDIUM_URL/g"
$ find . -name '*.php' | xargs sed -i "s/\$cfg\['photo_orig_url'\]/PHOTO_ORIG_URL/g"
```

Example of how to move your backup files (`.bak`), back to the origional file:

```
$ for x in `find -name \*.php.bak | sed -re 's/(.*).bak/\1/'`; \
do mv -i -f ${x}.bak ${x}; done
```

4.19 SFTP Chroot

I've tested following SFTP Chroot setup on Ubuntu 11.10. It should be noted that the only way that I was able to give users write access to their root folder was to allow all sftp users the ability to see other sftp user root folders. Users won't be able to see other user's files in other users home directories, they will only be able to see what the name of other users top level home directory. I would be happier if I could find a configuration that works which only allowed sftp users to login to their home directory and only see their home directory.

4.19.1 Setup

Make a directory where all your sftp chroot'ed users files will live:

```
$ cd /srv
$ sudo mkdir sftp
$ sudo chmod 755 sftp
```

Make an *sftp* group:

```
$ sudo groupadd sftp
```

Edit the sshd config file:

```
$ sudo vi sshd_config
```

Comment out this line:

```
# Subsystem sftp /usr/lib/openssh/sftp-server
```

Add this line directly below the line you just added:

```
Subsystem sftp internal-sftp
```

At the bottom of the file add the following:

```
Match group sftp
    X11Forwarding no
    ChrootDirectory /srv/sftp/
    AllowTcpForwarding no
    ForceCommand internal-sftp
```

Save the file:

```
:wq
```

Reload ssh:

```
$ sudo reload ssh
```

4.19.2 Add a SFTP User

Run the following which sets the new user's home directory to `/srv/sftp/user` and their shell to `/bin/false` so they can't login.

```
$ sudo useradd -d /srv/sftp/user -s /bin/false user
```

Set the users group to `sftp`:

```
$ sudo usermod -g sftp user
```

Set the users password:

```
$ sudo passwd user
```

Setup the users home directory:

```
$ cd /srv/sftp
$ sudo mkdir user
$ sudo chown user:user user
```

4.20 Sharing Directories on Linux

Add a Group:

```
$ sudo addgroup webadmin
```

Add an existing user to a group:

```
$ sudo usermod -a -G webadmin myusername
```

Change an existing users primary group:

```
$ sudo usermod -g webadmin myusername
```

View the a user's identity and primary group information:

```
$ id myusername
```

Create the shared folder:

```
$ mkdir myshareddir
```

Change the group owner for the group:

```
$ chown -R myusername:webadmin myshareddir
```

Give write permissions to the group:

```
$ chmod -R 775  myshareddir
```

Set an inherited group on files created within a directory, you can use the sgid bit on the directory permissions:

```
$ chmod -R 2775 myshareddir
```

or...

```
$ chmod -R g+s myshareddir
```

To make the default file permissions `rw` for group as well as user, you can change the `umask` to `0002` (either for all users in `/etc/profile`, or for individual users in `/etc/.bashrc`). Just append `umask 0002` to either of those files.

4.21 SSH Key Notes

Copying your public key to another computer:

```
$ cat ~/.ssh/id_rsa.pub | ssh username@server.domain.tld "cat >> .ssh/authorized_keys"
```

Generating your keys on a new computer:

```
$ ssh-keygen -t rsa -C "your_email@example.com"
```

4.22 Sudo Notes

Add an admin to sudo:

```
$ sudo usermod -G admin username
```

If they are members of other groups:

```
$ sudo usermod -a -G admin username
```

4.23 Ubuntu - Setup Local Email Forwarding

- Install Postfix Choosing "Internet Site" during the setup:

```
$ sudo aptitude install postfix
```

- Setup your Aliases:

```
$ sudo vi /etc/aliases

# Add something like the following to the file
postmaster:    root
www-data: root
root: myemail@example.com

$ sudo newaliases
```

- Test your aliases:

```
# You might have to install mailutils

$ echo "Testing" | mail root -s "Testing"

# You should get an email in the inbox of whatever email you set root to forward to
```

4.24 Ubuntu Firewall

Two guides I've found on the subject:

- https://help.ubuntu.com/community/UFW
- http://1000umbrellas.com/2010/04/29/how-to-set-up-the-firewall-using-ufw-on-ubuntu-lucid-lynx-server

4.25 Ubuntu FTP Server Setup Guide

This guide will walk you through the steps necessary to setup and install a vsFTPd FTP server for FTP only users with no shell login access and home directories that are chrooted.

This guide was originally adapted from Ubuntu's documentation.

4.25.1 Install and Setup vsFTPd

Install vsFTPd using the following command:

```
$ sudo apt-get install vsftpd
```

To allow users to upload files, edit `/etc/vsftpd.conf` and make sure the following line is uncommented:

```
write_enable=YES
```

Update the umask so that files uploaded will have global read permissions for servering the files up using a web server. Edit `/etc/vsftpd.conf` and make sure the following line is uncommented:

```
local_umask=022
```

Limit users to their home directories by uncommenting the following line in `/etc/vsftpd.conf`:

```
chroot_list_enable=YES
```

To allow users with a shell of `/usr/sbin/nologin` access to FTP, but have no shell access, edit `/etc/shells` adding the nologin shell:

```
# /etc/shells: valid login shells
/bin/csh
/bin/sh
/usr/bin/es
/usr/bin/ksh
/bin/ksh
/usr/bin/rc
/usr/bin/tcsh
/bin/tcsh
/usr/bin/esh
/bin/dash
/bin/bash
/bin/rbash
/usr/bin/screen
/usr/sbin/nologin
```

Add any users that you don't want to have FTP access to their files to `/etc/ftpusers`. For security reasons I would add any of your regular system users to this file.

```
# /etc/ftpusers: list of users disallowed FTP access. See ftpusers(5).

root
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
nobody
my_local_user
```

Make a group that you will assign to all FTP only users:

```
$ sudo groupadd ftp-users
```

Make a directory where all your FTP only users files will be located:

```
$ sudo mkdir /home/ftp_users
```

Now restart vsftpd:

```
$ sudo restart vsftpd
```

4.25.2 Create a FTP only user

To create a FTP only user run the following, replacing `my_user` with the username you would like to use.

```
$ sudo useradd my_user \
--home-dir /home/ftp_users/my_user \
--gid ftp-users \
--create-home \
--no-user-group \
--shell /usr/sbin/nologin; \
sudo passwd my_user;
```

4.25.3 How to fix: 500 OOPS: vsftpd: refusing to run with writable root inside chroot()

The following solution was found on askubuntu.com.

Install a different version of vsftpd:

```
sudo add-apt-repository ppa:thefrontiergroup/vsftpd
sudo apt-get update
sudo apt-get install vsftpd
```

Add the following to the bottom of the config file, /usr/sbin/vsftpd:

```
allow_writeable_chroot=YES
```

Restart vsftpd:

```
sudo service vsftpd restart
```

4.26 Ubuntu Munin Setup Guide

This guide originally borrowed from the Linode Library with modifications and was originally setup for Ubuntu 11.10.

4.26.1 Installing Munin

Make sure your package repositories and installed programs are up to date by issuing the following commands:

```
apt-get update
apt-get upgrade --show-upgraded
```

The Munin system has two components: a master component often referred to as simply "munin," and a "node" component, or "munin-node," which collects the data and forwards it to the master node. In Lucid, you need to install both the munin and munin-node packages if you wish to monitor the Linode you plan to use as the primary Munin device. To install these packages, issue the following command:

```
apt-get install munin munin-node
```

On all of the additional machines you administer that you would like to monitor with Munin, issue the following command:

```
apt-get install munin-node
```

The machines that you wish to monitor with Munin do not need to run Ubuntu. The Munin project supports monitoring for a large number of operating systems. Consult the Munin project's installation guide for more information installing nodes on additional operating systems.

4.26.2 Configuring Munin

Munin Master Configuration

The master configuration file for Munin is `/etc/munin/munin.conf`. This file is used to set the global directives used by Munin, as well as the hosts monitored by Munin. This file is large, so we've opted to show the key parts of the file. For the most part, the default configuration will be suitable to your needs.

The first section of the file contains the paths to the directories used by Munin. Note that these directories are the default paths used by Munin and can be changed by uncommenting and updating the path. When configuring your web server with Munin, make sure to point the root folder to the path of htmldir.

File excerpt: `/etc/munin/munin.conf`

```
# dbdir /var/lib/munin
# htmldir /var/cache/munin/www
# logdir /var/log/munin
# rundir  /var/run/munin
```

There are additional directives after the directory location block such as `tmpldir`, which shows Munin where to look for HTML templates, and others that allow you to configure mail to be sent when something on the server changes. These additional directives are explained more in depth on the munin.conf page of the Munin website. You can also find quick explanations inside the file itself via hash (#) comments. Take note that these global directives must be defined prior to defining hosts monitored by Munin. Do not place global directives at the bottom of the munin.conf file.

The last section of the `munin.conf` file defines the hosts Munin retrieves information from. For a default configuration, adding a host can be done in the form shown below:

File: `/etc/munin/munin.conf`

```
[example.org]
        address example.org
```

For more complex configurations, including grouping domains, see the comment section in the file, reproduced below for your convenience:

File: `/etc/munin/munin.conf`

```
# A more complex example of a host tree
#
## First our "normal" host.
# [fii.foo.com]
#       address foo
#
## Then our other host...
# [fay.foo.com]
#       address fay
#
## Then we want totals...
# [foo.com;Totals] #Force it into the "foo.com"-domain...
#       update no   # Turn off data-fetching for this "host".
#
#   # The graph "load1". We want to see the loads of both machines...
#   # "fii=fii.foo.com:load.load" means "label=machine:graph.field"
#       load1.graph_title Loads side by side
#       load1.graph_order fii=fii.foo.com:load.load fay=fay.foo.com:load.load
#
#   # The graph "load2". Now we want them stacked on top of each other.
#       load2.graph_title Loads on top of each other
#       load2.dummy_field.stack fii=fii.foo.com:load.load fay=fay.foo.com:load.l$
#       load2.dummy_field.draw AREA # We want area instead the default LINE2.
#       load2.dummy_field.label dummy # This is needed. Silly, really.
#
#   # The graph "load3". Now we want them summarised into one field
#       load3.graph_title Loads summarised
#       load3.combined_loads.sum fii.foo.com:load.load fay.foo.com:load.load
#       load3.combined_loads.label Combined loads # Must be set, as this is
#                                                 # not a dummy field!
#
## ...and on a side note, I want them listen in another order (default is
## alphabetically)
#
# # Since [foo.com] would be interpreted as a host in the domain "com", we
# # specify that this is a domain by adding a semicolon.
# [foo.com;]
#       node_order Totals fii.foo.com fay.foo.com
#
```

Munin Node Configuration

The default `/etc/munin/munin-node.conf` file contains several variables you'll want to adjust to your preference. For a basic configuration, you'll only need to add the IP address of the master Munin server as a regular expression. Simply follow the style of the existing allow line if you're unfamiliar with regular expressions.

File: `/etc/munin/munin-node.conf`

```
# A list of addresses that are allowed to connect.  This must be a
# regular expression, due to brain damage in Net::Server, which
# doesn't understand CIDR-style network notation.  You may repeat
# the allow line as many times as you'd like

allow ^127\.0\.0\.1$

# Replace this with the master munin server IP address
allow ^123\.45\.67\.89$
```

The above line tells the munin-node that the master Munin server is located at IP address `123.45.67.89`. After updating this file, restart the `munin-node`. In Ubuntu, use the following command:

```
/etc/init.d/munin-node restart
```

Web Interface Configuration

You can use Munin with the web server of your choice, simply point your web server to provide access to resources created by Munin. By default, these resources are located at `/var/cache/munin/www`.

If you are using the Nginx HTTP Server you can create a Virtual Host configuration to serve the reports from Munin. In this scenario, we've created a subdomain in the DNS Manager and are now creating the virtual host file:

File: `/etc/nginx/sites-available/stats.example.org`

```
server {

    listen       80;
    server_name  stats.example.org;
    access_log   /var/log/nginx/stats.example.org.access.log;
    error_log    /var/log/nginx/stats.example.org.error.log;
    root         /var/cache/munin/www/;

    location = /favicon.ico {
        return  404;
    }

    location / {
        auth_basic             "Restricted";
        auth_basic_user_file  /etc/nginx/htpasswd;
        index  index.html;
    }

}
```

If you haven't already setup the htpasswd file for controlling access to the site. Run the following command to encrypt a password.

```
perl -le 'print crypt("my_secret_password", "salt")'
```

Use the encrypted password generated and add htpasswd file.

File: `/etc/nginx/htpasswd`

```
<username>:<encrypted_password>
```

Enable the new site and reload `Nginx`:

```
cd /etc/nginx/sites-enabled/
sudo ln -s ../sites-available/stats.example.org
sudo nginx -s reload
```

4.26.3 More Information

You may wish to consult the following resources for additional information on this topic. While these are provided in the hope that they will be useful, please note that we cannot vouch for the accuracy or timeliness of externally hosted materials.

- Munin Homepage
- Munin Exchange
- Installing Munin on Other Linux Distributions
- Installing Munin on Mac OSX
- Installing Munin on Solaris

4.27 Ubuntu Security Guide

Good security resources that this is based off of ...

- My First 5 Minutes On A Server; Or, Essential Security for Linux Servers
- How To Set Up Your Linode For Maximum Awesomeness

4.27.1 Setup UFW

```
sudo ufw allow from {your-ip} to any port 22
sudo ufw allow http
sudo ufw allow https
sudo ufw enable
sudo ufw logging off
```

4.27.2 Upgrade packages

```
sudo aptitude update
sudo aptitude upgrade
```

4.27.3 Install fail2ban

```
sudo apt-get install fail2ban
```

4.27.4 Setup a deploy user

```
sudo useradd deploy
sudo mkdir /home/deploy
sudo mkdir /home/deploy/.ssh
sudo chmod 700 /home/deploy/.ssh
sudo chown -R deploy:deploy /home/deploy
```

4.27.5 Add your public key to the deploy and {your-username} user

```
sudo vim /home/deploy/.ssh/authorized_keys
sudo vim /home/{your-username}/.ssh/authorized_keys
sudo chmod 400 /home/deploy/.ssh/authorized_keys
sudo chmod 400 /home/{your-username}/.ssh/authorized_keys
```

4.27.6 Lock down ssh

Edit `/etc/ssh/sshd_config` and change settings to match the following:

```
PermitRootLogin no
PasswordAuthentication no
AllowUsers {your-username}@{your-ip} deploy@{your-ip}
```

4.27.7 Installed Logwatch

```
sudo aptitude install logwatch
```

Change the execute line in `/etc/cron.daily/00logwatch` to ...

```
/usr/sbin/logwatch --output mail --mailto {your-addy}@gmail.com --detail high
```

4.27.8 Enable Automatic Security Updates

```
sudo aptitude install unattended-upgrades
```

Edit `/etc/apt/apt.conf.d/10periodic` to look like:

```
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Download-Upgradeable-Packages "1";
APT::Periodic::AutocleanInterval "7";
APT::Periodic::Unattended-Upgrade "1";
```

Edit `/etc/apt/apt.conf.d/50unattended-upgrades` to look like:

```
Unattended-Upgrade::Allowed-Origins {
        "${distro_id}:${distro_codename}-security";
//      "${distro_id}:${distro_codename}-updates";
//      "${distro_id}:${distro_codename}-proposed";
//      "${distro_id}:${distro_codename}-backports";
};
```

4.27.9 Setup the timezone

```
dpkg-reconfigure tzdata
ntpdate ntp.ubuntu.com
```

Edit */etc/cron.daily/ntpdate* to match the following:

```
#!/bin/bash

ntpdate ntp.ubuntu.com
```

Change the files permissions:

```
sudo chmod 755 /etc/cron.daily/ntpdate
```

4.28 Ubuntu SSL Guide

I basically used this guide, https://help.ubuntu.com/6.06/ubuntu/serverguide/C/httpd.html and did the following:

```
$ mkdir /etc/apache2/ssl/
$ openssl genrsa -out phpmyadmin.ciy.com.key
$ openssl req -new -key phpmyadmin.ciy.com.key -out phpmyadmin.ciy.com.csr
```

Entered the following information:

```
Enter pass phrase for phpmyadmin.ciy.com.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:Missouri
Locality Name (eg, city) []:Joplin
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Christ In Youth
Organizational Unit Name (eg, section) []:Web
Common Name (eg, YOUR name) []:phpmyadmin.ciy.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Self-Signed the certificate:

```
$ openssl x509 -req -days 365 -in phpmyadmin.ciy.com.csr -signkey phpmyadmin.ciy.com.key -out phpmyad
```

CHAPTER 5

Web Development Docs

5.1 Combine Multiple PDFs into a Single PDF

First install Ghostscript. You can install Ghostscript using brew with `brew install ghostscript`.

After Ghostscript is installed run the following:

```
$ gs -q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -sOutputFile=merged.pdf file1.pdf file2.pdf
```

5.2 Sublime Text 2 Notes

5.2.1 User File Settings

```
{
    "color_scheme": "Packages/Theme - Flatland/Flatland Dark.tmTheme",
    "draw_indent_guides": false,
    "fold_buttons": true,
    "font_face": "Meslo LG S",
    "font_size": 13,
    "highlight_line": true,
    "ignored_packages":
    [
    ],
    "open_files_in_new_window": false,
    "pep8_ignore":
    [
        "E501"
    ],
    "pyflakes_ignore":
    [
        "import *"
    ],
    "rulers":
    [
        120
    ],
    "scroll_past_end": true,
    "sublimelinter": true,
    "tab_size": 4,
    "theme": "Flatland Dark.sublime-theme",
    "translate_tabs_to_spaces": true,
    "trim_trailing_white_space_on_save": true,
    "use_tab_stops": true,
    "word_wrap": false,
    "wrap_width": 120
}
```

5.2.2 Setup Sublime Command Line Helper

```
ln -s "/Applications/Sublime Text 2.app/Contents/SharedSupport/bin/subl" ~/usr/local/bin/subl
```

I also added the following to my bash profile (*.bash_profile*):

```
alias e=subl
```

Now you can load a folder into Sublime from the command line.

```
cd ~/Sites/mysite
subl .
```

5.2.3 Packages I've Added

- Sublime Package Control - Use this to manage and install all your packages.
- Djaneiro
- sublime-2-zencoding (usage notes)
- SublimeLinter
- sublime-text-2-git
- sublime-github
- Flatland
- SublimeCodeIntel
- SideBarEnhancements

5.3 Submitting Patches

5.3.1 Submitting Patches Using diff

```
$ diff -urN original-directory altered-directory > something.patch
```

5.3.2 Submitting Patches Using SVN

```
$ svn diff checkout-directory > something.patch
```

5.4 Website Pre-Launch Check List

- Make a favicon
- Make sprites
- Combine, Minify and Compress CSS and JS
- Test all pages in all major browsers
- Install a warning message for IE 6
- Check all meta data
- Check to make sure you have a styled 404 and 500 error pages
- Make sure you're using page caching
- Make sure you have google Analytics installed
- Make sure the site works with out Javascript turned on or that you have a javascript required message.
- Add a print style sheet
- Add a Sitemap for search engines
- Proofread content
- Double check your links (http://validator.w3.org/checklink/)
- Test all forms for validation and functionality
- Optimize the site for performance, http://developer.yahoo.com/performance/rules.html