# JB on programming Documentation

***Release 1.0.0***

**Jacek Bzdak <jacekfoo@gmail.com>**

**Apr 09, 2017**

# Contents


1 How to use pythonbrew and virtualenv 3

2 Remapping keys in linux 7

3 How to use Libre Office serial document (mail merge) functionality 9

4 Disabling Optimus on Debian and using only Nvidia Graphics card 11

5 How to extract images from pdf-s 13

6 How to extract tables from pdf-s 15

7 When your cuda installation stops working (on DEBIAN) 17

8 Set dirs to 755 and files to 644 19

9 Autocropping all images in folder 21

10 W530 battery life 23

11 Reading numpy structured from a text file 25

12 Install node modules without root 27

13 Move legend outside of figure in matplotlib 29

14 How to deploy django application on debian server (UWSGI version) 33

15 Compress a dir to tar.xz 37

16 How to compile, install and debug Callibre in your favorite IDE 39

17 How to change what external program calibre uses to open files 41

18 So you want to tweak docutils? 43

19 Connecting remote display (projector) to `i3` 45

20 How to create password hashed in Linux `/etc/shadow` format (crypted password) 47

21 How to get a table size in `psql` 49

22 CGI on Nginx or how to run man2html on debian 51

23 Problems with nfs shares on Debian (in case of Vagrant) 53

24 New Java Features 55

25 How to disable subpixel rendering in Jetbrains products 67

26 Linux dynamically assigned ports 69

27 Asyncio servers in Python 71

28 How to discard connection in a fast way 75

29 Some remarks about databases 77

30 MVCC in postgresql 81

31 Should you use BRTFS for your laptop? 83

32 How to deploy django application on debian server (UWSGI version) 85

33 What to do when you get [Read error 4] when running `apt` 91

34 How to encrypt a usb drive using cryptsetup 93

35 Cheap and fast way to create your own Maven repository on S3 95

36 How to debug zsh startup time 97

37 Indices and tables 99

Contents:

CHAPTER 1

# How to use pythonbrew and virtualenv

**Warning:** This is outdated and not longer revelant.

In most of my projects I use virtualenvs and install dependencies by pip, well it's nothing special, since virtualenvs have become mostly standard in python developement.

But since among my colleagues I'm sort of early-adopter, and I find myself explaining this again and again, I figured that I'll write this article.

## What is a python virtual environment

Virtual environment is *a way to manage python interpreter and associated installed packages*. Generaly you use separate virtualenvs for all your projects, in which case all these projects can depend on different versions of libraries.

## Advantages of using virtualenv

- Projects using different virtualenvs can use different versions of libraries.
- To install a library inside virtualenv you don't need to have root privelages, which is good even if you are root, because packages installed via *pip* shouldn't be trusted.

## Installing virtualenv

Normally you install virtualenv using your system package manager.

## Using virtualenv

```
virtualenv foo # create virtualenv in foo subfolder of local folder
New python executable in foo/bin/python
Installing distribute (..) done.
Installing pip...............done.

source foo/bin/activate # Activate the virtualenv

pip install django #install packages
Downloading/unpacking django
  Running setup.py egg_info for package django

    warning: no previously-included files matching '__pycache__' found under␣
↪directory '*'
    warning: no previously-included files matching '*.py[co]' found under directory '*
↪'
Installing collected packages: django
  Running setup.py install for django
    changing mode of build/scripts-2.7/django-admin.py from 644 to 755

    warning: no previously-included files matching '__pycache__' found under␣
↪directory '*'
    warning: no previously-included files matching '*.py[co]' found under directory '*
↪'
    changing mode of /tmp/foo/bin/django-admin.py to 755
Successfully installed django
Cleaning up...

python -c 'import django' #Verify django is installed

deactivate # turn off the virtualenv

python -c 'import django' # See that django was installed locally only!
Traceback (most recent call last):
  File "<string>", line 1, in <module>
ImportError: No module named django
```

Used commands

**virtualenv path** creates virtualenv in specified *path*. Interesting options include:

- *virtualenv –python=python2.5* creates virtualenv using specified interpreter
- *virtuaelnv –system-site-packages* allows virtualenv to use packages installed systemwide (it is nice since some packages can't be installed easily via pip).

**source path/bin/activate** Enables virtualenv in current console

**pip install packagename** installs package inside virtualenv (if it was activated)

**deactivate** disables virtualenv in current console

## Pip cache trick

When using virtualenvs you'll find that installing packages via pip can be painfully slow. You can instruct pip to use cache for downloaded packages by adding following line to your *.bashrc* file.

```
export PIP_DOWNLOAD_CACHE=$HOME/.pipcache
```

## Using pythonbrew

Virtual enviorment *borrows* one of your system interpreters, pythonbrew takes it step further: it downloads and compiles your own python interpreter (and has integrated support for virtualenvs.

## Installing pythonbrew

Dont use instructions from pythonbrew repository, as they basically tell you: 'download file via http and execute it in bash' (which is *very insecure*).

Just clone repository stored in github at https://github.com/utahta/pythonbrew and execute *install_pythonbrew.py* script.

Then as instructed paste following into bashrc file:

```
[[ -s "$HOME/.pythonbrew/etc/bashrc" ]] && source "$HOME/.pythonbrew/etc/bashrc"
```

## Using pythonbrew

**pythonbrew install 2.7.1** Installs python *2.7.1* into pythonbrew

**pythonbrew use 2.7.1** Use installed python *2.7.1* in current console.

**pythonbrew venv create name -p 2.7.1** Create virtualenv named *name* for python interpreter version *2.7.1*

**pythonbrew venv use name -p 2.7.1** Activate virtualenv named *name* for python interpreter version *2.7.1*

CHAPTER 2

# Remapping keys in linux

## The Problem

Keyboard on my Thinkpad W530 has one quite moronic feature: page up and page down are beside arrow keys, while Home and End are on the other side of keyboard.

I want to switch these keys.

## The solution (TL;DR;)

Add following line to `/etc/rc.local`

```
setkeycodes e049 102 e051 107 e047 104 e04f 109
```

## Solutions

You may do it in couple of ways:

- Do it at X level remapping using `xmodmap`
- Do it at lower level

I did it previously using `xmodmap`, this solution had many drawbacks, and since upgrade to gnome 3.8 I needed to launch `xmodmap` by hand. Anyways this is what everyone suggests, like everywhere, so just google this solution.

## Changing key bindings using udev

When linux reads a keystroke, first thing that is registered is so called `scancode` than scancode is converted to `keycode`.

First thing you'll need to know is what scancodes you want to remap, then you'll need to know to what keycodes you'll remap. To do this you'll need the `showkey` program.

To know `scancode` of a page up key you'll need to:

```
jb ~ $ sudo showkey --scancodes
# Some text
# I press PgUpXF86MonBrightnessUp
^[[H0xe0 0x47 0xe0 0xc7
```

It outputed four hex numbers: `0xe0 0x47 0xe0 0xc7`, in my case first two were a scancode of the key. First one `e0` is an escape character.

Then youll need to know keycodes of your chars:

```
jb ~ $ sudo showkey --keycodes
# Some text
# I press PgUp
^[[Hkeycode 102 press
keycode 102 release
```

Now we can remap keys, to do this you'll need the `setkeycodes` command, it takes list of pairs of numbers. First item in each pair is a `scancode` sa a hexadecimal number, without the `0x` (so if scancode of `PgUp` is `0xe0 0x47` write: `e047`).

In my case it was:

```
sudo setkeycodes e049 102 e051 107 e047 104 e04f 109
```

Syntax for `setkeycodes` is quite moronic, so here is explanation, following command `setkeycodes e049 102` means: set keycode `102` (102 is in decimal) for scancode combination: `0xe0 0x49`, note that both bytes are concatenated, and are in without leading `0x`.

Changes made by setkeycodes are not permanent, so you'll need to find a way to execute this script at start of the system.

> **Warning:** Using this method can harm your computer. If you'll mess keymap severly, and make changes peremanent you might be unable to login (when you for example remap letter characters).

For example insert this line inside `/etc/rc.local` file.

## When this approach will not work, and what to do then

This will change keyboard bindings for all input devices, which might not be what you want.

You should use `udev` to do this, I have tried and failed, if you'll succeed please let me know :)

## References

1. Ask Ubuntu http://askubuntu.com/questions/69804/how-do-i-change-the-keymap-of-a-single-device-logitech-presenter
2. Arch Linux webpage: https://wiki.archlinux.org/index.php/Map_scancodes_to_keycodes, https://wiki.archlinux.org/index.php/Extra_Keyboard_Keys

CHAPTER 3

# How to use Libre Office serial document (mail merge) functionality

Libre Officess mailmerge functionality is a nice feature that allows you to create *any kind of serial document*, while all documentation seems to indicate that it is only usable for creating e-mails.

I used this tutorial (it has nice screenshots and so on).

So anyways steps are as follows:

## Create data source

Create a data source, this can be:

1. A Libre Office Base database (or any database)
2. A csv file (**remember to have header row**).
3. An Libre Office Spreadsheet file (**also add a header row**)

## Create a document template

Write the text

## Attach database to the document

In writer Edit -> Exchange Database -> Browse -> Select your database file.

In writer View -> Datasources, then select your database file again.

You should see table contents.

## Add fields to the document

Now you should be able to Drag and Drop fields (columns) from the table to the document.

## Create serialized documents

Mail Merge Wizard or click the envelope icon in data sources. This stuff is mostly stupid and deals with preformatted address blocks, etc. I ve never used it.

1. Select starting Document: Select current document.
2. Select document type: Select letter
3. Insert address block: Uncheck all checkboxes.
4. Create salutation: Uncheck all checkboxes.

CHAPTER 4

# Disabling Optimus on Debian and using only Nvidia Graphics card

I had some problems with bumblebee and optirun command — it just suddenly stopped working.

So here is how to switch your computer to use nvidia on debian, and use nvidia proprietary drivers

1. Remove `bumblebee`
2. Install Nvidia proprietary drives (https://wiki.debian.org/NvidiaGraphicsDrivers).

   **Note:** Remember to configure `xorg.conf` precisely as in the instructions.

3. Restart. Go to bios and switch off integrated graphics card.

If you restart your computer before step 3, most probably your X will not be usable.

CHAPTER 5

# How to extract images from pdf-s

You'll need to use `convert` tool from the imagemagic suite.

```
convert -density 450 -quality 100 file.pdf foo.png
```

And you'll get image for each page.

Since by default convert created image files with low resolution, after too much googling I found that you need to fiddle with `density` and `quality` switches.

CHAPTER 6

# How to extract tables from pdf-s

Extracting tables from pdf files is hard, as in pdf there are not tables, just lines and letters.

I use this tool, it is able to extract most of tabular data and recovers structure very well.

Downsides are that it is painfuly slow (launches a process to extract each cell).

CHAPTER 7

# When your cuda installation stops working (on DEBIAN)

There can be many signs for this error:

`deviceQuery` returns:

```
cudaGetDeviceCount returned 38-> no CUDA-capable device is detected
```

or you get:

```
no CUDA-capable device is detected
```

In my case it was version mismatch between installed `nvidia` kernel module and `libcuda1` library.

**Note:** Normally this module is installed via `DKMS` so kernel module version should match version of `nvidia-kernel-dkms`, but this is not always the case. . .

To check version of installed kernell module:

```
sudo modinfo nvidia-current | grep version
```

For now it should be `319.xxx`. If it has version mismatch `libcuda1` you have source of your errors (yay!).

CHAPTER 8

# Set dirs to 755 and files to 644

So I dont forget, if you want to share a directory structure with people you can change permissions in following way:

```
find dir -type d -exec chmod 755 {} \;
find dir -type f -exec chmod 644 {} \;
```

In this way others will be able to enter any directory and won't be able to execuye files.

CHAPTER 9

# Autocropping all images in folder

Another piece of code I'm putting here for posteriority. Anyways, let's say I have couple hundred imahes in folder (autogenerated) and I need to crop them automatically (remove extra background).

Here is a quic command that does the trick:

```
mogrify -trim +repage *.png
```

This comes with imagemagic suite.

CHAPTER 10

# W530 battery life

After about a year of usage maximal charge of my battery dropped about 40%, which is way too much. On Windows there are some settings that "enhance battery lifespan".

**Note:** It seems that to enhance lifespan of your batteries you'll need to:

- Charge it for longer periods (ideally allways to full charge)
- Don't charge it to 100%, set full charge to 90-something%

There are no generic tools to configure such behaviour — as it all sits inside BIOS. This question on ubuntu SO helped me to find solution, but nevertheless it didnt work on my W530 laptop.

On newer thinkpats you'll need to use: tcpapi-bat (you might install it from this PPA (it's for ubuntu but also works on Debian.

After instal you'll need to insert following into */etc/rclocal.d* (it will be called after startup):

```
tpacpi-bat -s ST 0 80
tpacpi-bat -s SP 0 90
```

It will do the following: battery will stop chaging after charging to `90%` ( of current maximal charge) and battery will start charging only after it is discharged to `80%` (which will cause that you will charge it for 10% capacity or greater).

It would be great it I could (for example) inhibit charging for (let's say) 15 minutes after connectiong to AC, as I often connect the computer and then decide I need to go somewhere else, which is not healthy for battery.

CHAPTER 11

# Reading numpy structured from a text file

Numpy has a very nice feature: a structured array, that is array in which rows have some structure, and can store different types of data in each column.

For example:

```
>>> import numpy as np
>>> arr = np.zeros(10, dtype=[['id', np.uint16], ['position', np.dtype('3float32')], [
↪'momentum', np.dype('3float32')]])
```

We have defined a structured array in each row we store: id of a particle (unsigned int), its position (three floats) and momentum (again three floats).

You can easily select from this array:

```
>>> arr['position']
>>> arr[0]['position']
>>> arr[arr['id']=1]['position']
```

This is a nice format because:

- Your data has structure. No more off-by-one errors: particle position is labeled.
- Very easy to load from binary files

Loading from text files is a entirely different matter — because writing to such arrays is kind of pain.

My requirements were:

- Array structure is the same as source file structure (order of fields is the same)
- Array structure is defined only in single place: that is dtype defintion

## Solution

Solution is to:

- Read file line by line parsing contents to unstructured array.
- Create structured view
- Should be fast, that means no copying of large arrays.

Actual dtype used:

```
URQMD_DATA_DTYPE = [
    ("time", np.float32),
    ("position", np.dtype("3float32")),
    ("energy", np.float32),
    ("momentum", np.dtype("3float32")),
    ("mass", np.float32),
    ("particle_type", np.float32),
    ("additional", np.dtype("5int32")),
]
```

Helper function that takes structured dtype, and turns it to dtype that has the same number of fields but is unstructured:

```
def serialize_dtype(dt):
    dt = np.dtype(dt)
    newdt = []
    for item in dt.descr:
        if len(item) == 2:
            count = 1,
            name, type = item
        else:
            name, type, count = item
        if len(count) > 1:
            raise ValueError()
        count = count[0]
        for ii in range(count):
            newdt.append(type)
    return np.dtype(", ".join(newdt))
```

Now `frame` is a list of lines from text file.

```
parsed = np.zeros(len(frame), dtype=serialize_dtype(URQMD_DATA_DTYPE)) # Create array␣
↪without structure
for ii, line in enumerate(frame):
    data = [float(x) for x in line.split()] # Parse lines
    #-- ignoring wheher it is a float or int
    parsed[ii] = tuple(data) # Now numpy will convert single row to proper types
parsed = parsed.view(URQMD_DATA_DTYPE) # Create a structured view (no copy!)
```

Sound simple but took me some time to get it right.

CHAPTER 12

# Install node modules without root

Sometimes you need to install some node modules, but every tutorial says: "Do `sudo npm install -g something`. I dont like to mix sudo and downloading some stuff from unsecure location. There is a very easy way to install such modules in a easy way.

There is how:

1. Download/untar node js. Let's say it is in `/tmp/node`
2. Create directory named "`$HOME/.local`" (or any other really)
3. `cd /tmp/node`
4. `configure --prefix="$HOME/.local`
5. `make && make install`

Now you have a working installation of node in `$HOME/.local/bin`, add this directory to `PATH` and you'll be able to install modules "globally" to your home folder, for example:

```
npm install -g coffee-script
```

and `coffee` will be accessible from `$HOME/.local/bin`.

**Note:** I used this for inspiration.

CHAPTER 13

# Move legend outside of figure in matplotlib

Another one so I won't forget. Let's do some setup:

```
%matplotlib inline
from matplotlib import pylab
from matplotlib.font_manager import FontProperties
```

Lets assume you have a plot and you want to move legend outside of the plot window. Like this:

```
pylab.plot(range(10), label="Plot 1")
pylab.plot(range(10, 0, -1), label="Plot 2")
pylab.legend()
```

See how legend overlaps with the plot. Fortunately matplotlib allows me to move legend out of the way, kinda sorta.

```
pylab.plot(range(10), label="Plot 1")
pylab.plot(range(10, 0, -1), label="Plot 2")
pylab.legend(loc=9, bbox_to_anchor=(0.5, -0.1))
```

Little bit better! `bbox_to_anchor` kwarg sets legend to be centered on `X` axis and below that axis. For some unfanthomable reason you'll need to add `loc=9` so legend is actually centered.

Let's tweak this a bit:

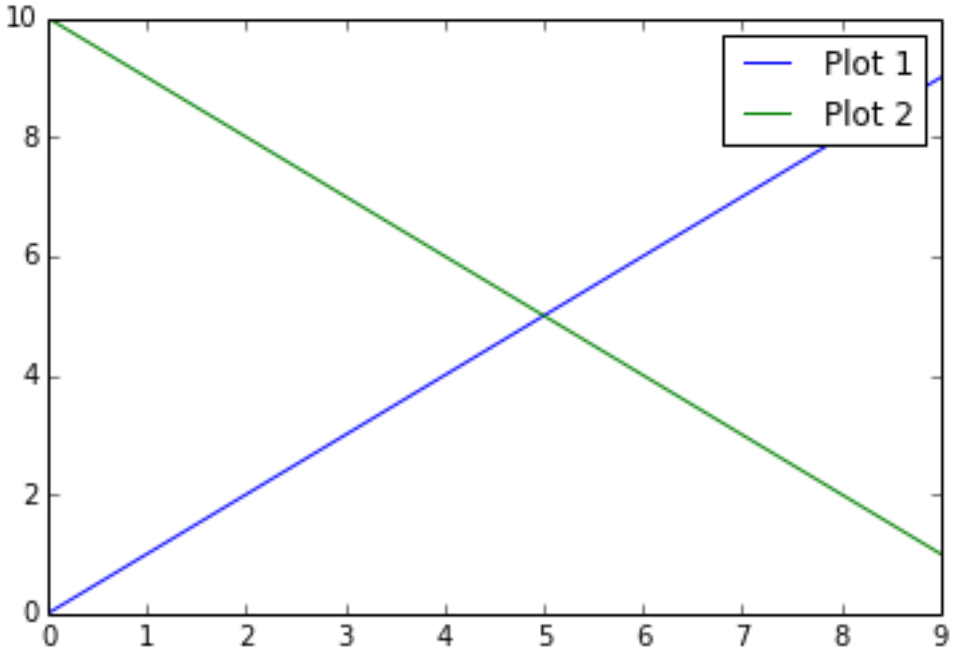
```
pylab.plot(range(10), label="Plot 1")
pylab.plot(range(10, 0, -1), label="Plot 2")
pylab.legend(loc=9, bbox_to_anchor=(0.5, -0.1), ncol=2)
```

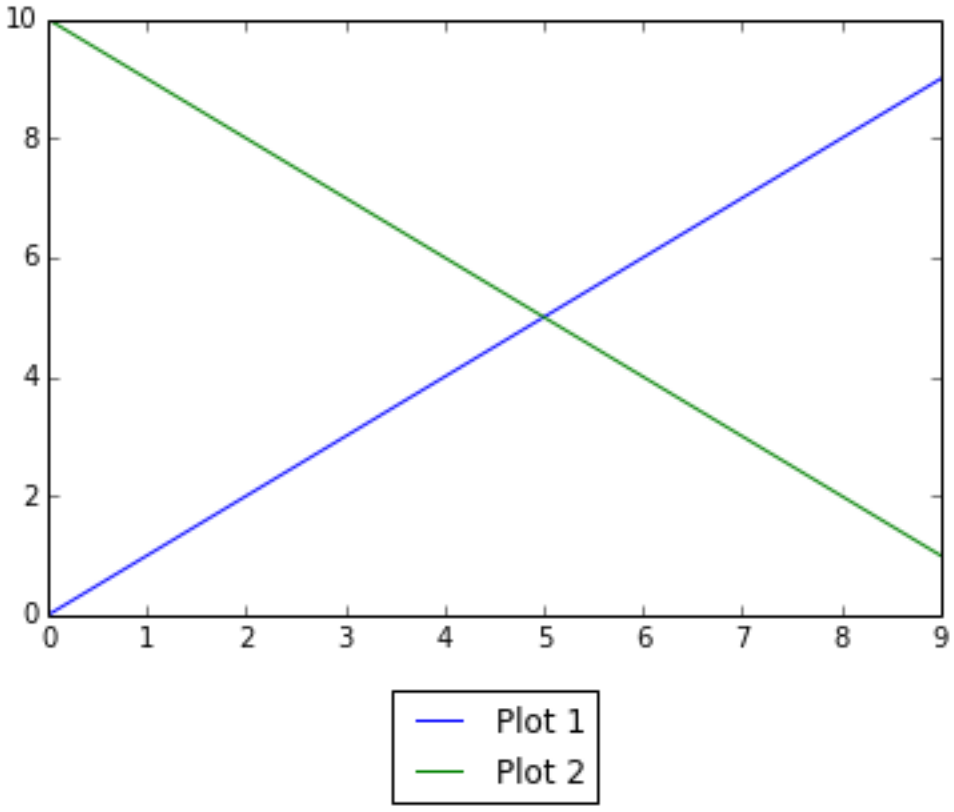
You may pass `ncol` kwarg that sets number of columns in the legend.

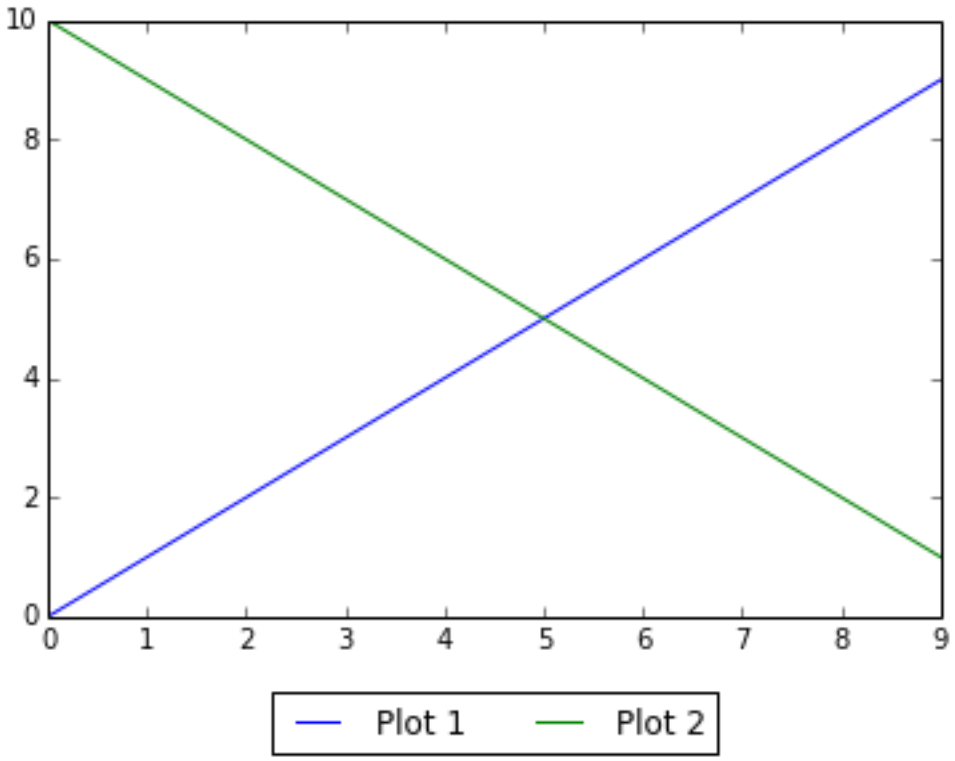
Now plot labels are on the same level and we have saved some space.

---

**Note:** If you have long legend legend it might be worth to decrease font size, like that:


Plot 1
Plot 2
0
2
4
6
8
10
0
1
2
3
4
5
6
7
8
9


Plot 1
Plot 2
0
2
4
6
8
10
0
1
2
3
4
5
6
7
8
9




```
fontP = FontProperties()
fontP.set_size('small')
pylab.legend(prop = fontP, ....)
```

---

So far so good, but if you try to save this image legend will get cut in half.

To fix this you'll need to:

1. Set `bbox_inches="tight"` keyword argument
2. Pass legend as `additional_artists` kwarg argument. This argument takes a list so you'll need to append legend somewhere.

Here is an example:

```
pylab.plot(range(10), label="Plot 1")
pylab.plot(range(10, 0, -1), label="Plot 2")
art = []
lgd = pylab.legend(loc=9, bbox_to_anchor=(0.5, -0.1), ncol=2)
art.append(lgd)
pylab.savefig(
    "/tmp/foo-fixed.png", additional_artists=art,
    bbox_inches="tight")
```

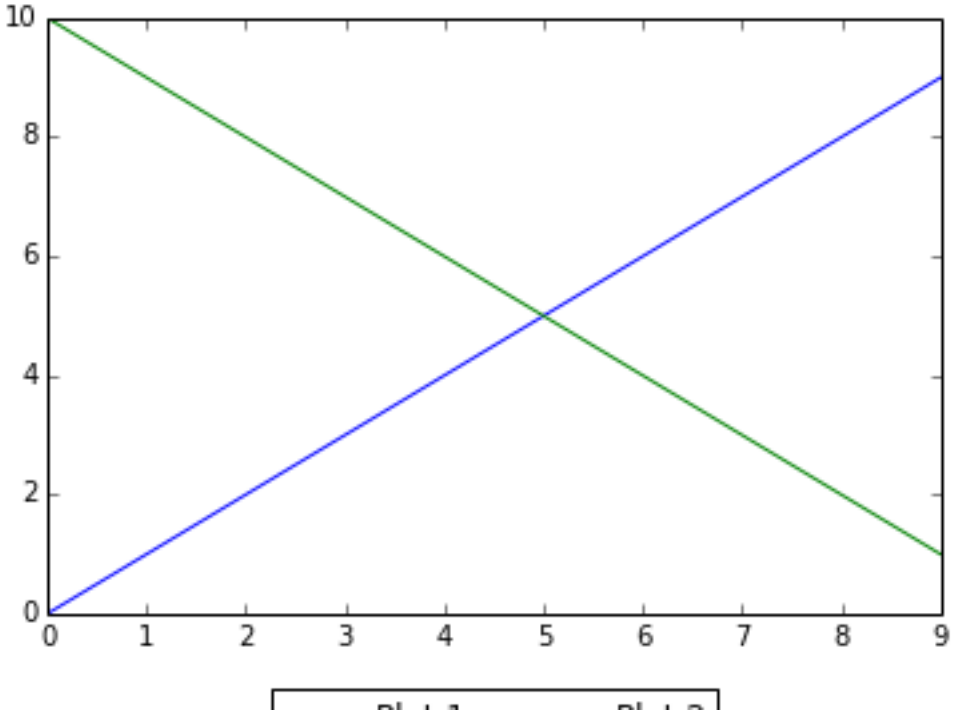



Fig. 13.1: How this figure is saved.

CHAPTER 14

# How to deploy django application on debian server (UWSGI version)

## Install proper version of python

On debian I'd discourage using `system` python for deployment — mostly becaluse they tend to upgrade minor python versions without notice, which breaks `C ABI` in installed virtualenvs.

So either roll your own `deb` files that install pythons somewhere in `/usr/local` or compile python on server (if you frown on having developement tools on your server roll `debs`).

Pythonz is a nice tool to compile (and manage) many versions of python.

## Install proper version of virtualenv

**Note:** This is more-or less irrevelant as Python (from version 3.4 onwards) has it's own built-in virtualenv tool, so if you use non-system one it'll have its own virtualenv.

Debian comes with ancient python/virtualenv version, and (at least on `testing`) it often breaks this install. If you install virtualenv locally it'll be much more stable.

To install virtualenv locally just download virtualenv, unpack package and use `virtualenv.py`.

This hsa the added benefit that you can have two versions of virtualenv one for python2 and one for python3.

## Install your application into virtualenv

You know how to do that don't you?

Now you can tests whether your setup is correct, just run

```
./manage.py runserver
```

and see if you can connect to your application.

## Install uwsgi into your virtualenv

Install `uwsgi` **into your virtualenv** from `pip`. Now you can run your application using uwsgi:

```
uwsgi --http :8000 --module webapp.wsgi
```

I strongly discourage you from using uwsgi bundled with system.

## Use supervisord to launch your applicaiton

You'll need to run your application on system start, simplest way is to use `supervisord` (you can install it via aptitude).

First create a script that will:

1. Source virtualenv
2. cd to your app directory
3. run uwsgi.

Something along the lines:

```
#!/bin/bash
export HOME=/home/user
source $HOME/venv/bin/activate
cd $HOME/webapp
uwsgi --socket :8000 --module webapp.wsgi
exit $?
```

Notice that `--http` turned into `--socket`. Now uwsgi will speak `uwsgi` protocol (which should be faster than http).

Then add configuration to supervisord. All services are defined as `*conf` files (in `ini` format) inside `/etc/supervisor/conf.d`.

Create file containing something like:

```
[program:webapp]
command=/home/webapp/webapp.sh
autostart=true
autorestart=true
stderr_logfile=/var/log/webapp.err.log
stdout_logfile=/var/log/webapp.out.log
user=webapp
```

For all configuration options see the documentation.

Now after calling: `service supervisor restart` your django application should be running.

## Connect uwsgi and nginx

**Note:** This part is more or less ripoff from: http://uwsgi-docs.readthedocs.org/en/latest/tutorials/Django_and_nginx.html

Add following sections to nginx configuration:

```
upstream django {
    # server unix:///path/to/your/mysite/mysite.sock; # for a file socket
    server 127.0.0.1:8000; # for a web port socket (we'll use this first)
}

# configuration of the server
server {
    # the port your site will be served on
    listen      80;
    # the domain name it will serve for
    server_name .example.com; # substitute your machine's IP address or FQDN
    charset     utf-8;

    # max upload size
    client_max_body_size 75M;   # adjust to taste


    # Finally, send all non-media requests to the Django server.
    location / {
        uwsgi_pass  django;
        include     /path/to/your/mysite/uwsgi_params; # the uwsgi_params file you␣
↪installed
    }
}
```

Now you should see something on your server port `80`. To finalize our setup we need to create static and media directories.

## Connect ngingx and uwsgi via linux file sockets

Because of many (peformance, safety) reasons it is better to connect nginx with uwsgi via linux domain sockets.

First replace `uwsgi` call with something like that:

```
uwsgi --module webapp.wsgi --socket $HOME/sock/webapp.sock  --chown-socket=webapp-
↪user:www-data --chmod-socket=660
```

This does the following: creates a socket in `$HOME/sock/webapp.sock`, sets its group ownership to `www-data` (which is user/group used by both `nginx` and `apache` on default debian configuration).

**Note:** Linux file sockets use normal file permissions, so `nginx` has to have read-write access to it.

Then replace:

```
upstream django {
    server 127.0.0.1:8001; # for a web port socket (we'll use this first)
}
```

with:

```
upstream django {
    server unix:///path/to/your/mysite/webapp.sock; # for a file socket
}
```

# Update media and static root

## Update settings py

You'll need to update `STATIC_ROOT`, `STATIC_URL`, `MEDIA_ROOT`, `MEDIA_URL` settings of youe app.

Something along the lines:

```
MEDIA_ROOT = '/var/drigan-media'
MEDIA_URL = '/media/'
STATIC_ROOT = '/var/drigan-static'
STATIC_URL = '/static/'
```

## Update nginx

```
location /media  {
    alias /path/to/your/mysite/media;  # your Django project's media files - amend as␣
↪required
}

location /static {
    alias /path/to/your/mysite/static; # your Django project's static files - amend␣
↪as required
}
```

# Tweak uwsgi so it scales

You might want tweak `uwsgi` so it launches more processes/workers, but this well outside scope of this tutorial.

CHAPTER 15

# Compress a dir to tar.xz

Everyone is using tar.gz and tar.bz2 formats, and these compression algorithms (while stable and installed everywhere) are definetely not state-of-the-art.

Anyways most modern systems have much better compression ratios (or much less compression/decompression overhead). Namely `xz` (better ration) `lzo` (much better resource usage).

Anyways since I allways forget syntax to do compression here is proper command, but today I found this nice tar switch `-a` which means: "Try to guess compression format from file extension", so:

```
tar -caf foo.tar.xz data
```

will compress to `xz`, while

```
tar -caf foo.tar.lzo data
```

will compress to `lzo`. At least `tar` has sane API.

This is nice, but sometimes you'll want to tweak compression ratio for used compressor — in this case just use pipes. If you pass `-` (if anything else is non-obvious just use `man`).

```
tar -c data - | xz -9c > data2.tar.xz
```

# CHAPTER 16

## How to compile, install and debug Callibre in your favorite IDE

Calibre is a very nice ebook organizer, however developement documentation is, quite laconic.

Installing it for the first time turned out to be a major pain in the ass, when I installed it for the first time, as I had to do some experimentation.

I have installed callibre to a virtual enviorment, which also is a pain, nevertheless here is what to do.

1. Download callibre source

```
git clone https://github.com/kovidgoyal/calibre
```

2. Install all system dependencies (there quite many of them). For systems not based on debian install them by hand.

```
sudo aptitude build-dep calibre
```

3. Create virtual enviorment. You'll have to pass `--system-site-packages` switch that allows virtualenv to use system python packages.

```
cd calibre
virtualenv -p python2 --system-site-packages venv
source venv/bin/activate
```

4. Try to launch `setup.py`

```
python setup.py
Traceback (most recent call last):
  (...)
  File "/tmp/calibre2/setup/build_environment.py", line 103, in get_sip_dir
  (...)
EnvironmentError: Failed to find the location of the PyQt5 .sip files
```

   Callibre can't find system SIP files for PyQt5, you'll beed to edit `setup/build_environment.py` find a line that contains:

```
pyqt['pyqt_sip_dir'] = get_sip_dir(sys.prefix if iswindows else os.path.join(sys.
↪prefix, 'share', 'sip'))
```

and replace it with:

```
pyqt['pyqt_sip_dir'] = get_sip_dir(sys.prefix if iswindows else os.path.join('/
↪usr/', 'share', 'sip'))
```

Just replace `sys.prefix` to `/usr`, if you installed PyQt and other dependencies for calibre setup.py should work now

5. Launch `python setup.py bootstrap`. This command does some stuff I don't really understand, but it is neccessary to build calibre.
6. Install Calibre in developement mode: `python setup.py develop`.
7. Now to debug Calibre you'll just need to launch `venv/bin/calibre` in debug mode in your favorite IDE.

CHAPTER 17

# How to change what external program calibre uses to open files

Here are two (mine) related questions:

- http://stackoverflow.com/q/32256039/7918
- http://unix.stackexchange.com/q/225103/5612

Abbreviated solution is here:

Calibre uses `QDesktopServices.openUrl(qurl)` method, that uses `xdg-utils` under the hood to open files.

IMO `xdg-utils` is a stinking mess of fragile bash scripts, that ... behaves erratically at best.

To set what program is used to launch a certain file you'll need to:

1. Find mimetype of this file, this can be done by:

   ```
   xdg-mime query filetype <filename>
   ```

   This will print out mimetype of that file, for example `application/pdf`.

2. Then you'll need to find a `desktop` file associated with application you want to use. In debian desktop files are in `/usr/share/applications`.

   In case of pdf file I wanted to open it using `evince` that has `desktop.file` named `evince.desktop`.

3. Associate desktop file with mimetype, using command: `xdg-mime default evince.desktop application/pdf`.

4. Werify it works using `xdg-open`. Now try to open this file using `xdg-open`, if it opens in a proper application it should open in proper application in calibre.

CHAPTER 18

# So you want to tweak docutils?

Docutils is a very nice python package that converts text documents (namely: documents formatted in restructured text) to various usefull formats (like latex, open office and so on). Sometimes default docutils converters need a little bit tweaking do fit your needs.

I needed to tweak latex generated by docutils to my specific needs, and I discovered that I can't find any tutorial on how docutils work intenally, so here is what I found out in my brief encounter with it.

**Note:** This article is based on my experiences from tweaking docutils myself (I couldn't find revelant pages in the documentation), so this post is in no way authoritative.

## How to tweak output of (for example) rst2latex

You need to create module that defines your `Writer` that needs to inherit from `docutils.writers.latex2e.Writer`, latex contents are actually generated by `Writer.translator_class` that is an instance of `NodeVisitor`.

Here is an example of my `Writer` that renders `code-block` directive as a `verbatim` block in LaTeX.

```
# -*- coding: utf-8 -*-

import re

from docutils import nodes

from docutils.writers.latex2e import Writer as LatexWriter, LaTeXTranslator

class TweakedLatexWriter(LatexWriter):

  def __init__(self):
    super().__init__()
    self.translator_class = TweakedTranslator
```

```
class TweakedTranslator(LaTeXTranslator):

  def visit_literal_block(self, node):
    if 'code' in node.attributes['classes']:
      # code-block
      self.requirements['upquote'] = '\\usepackage{upquote}'
      self.out.append('\n\\begin{verbatim}\n%s\n\\end{verbatim}\n' % node.astext())
      raise nodes.SkipNode
    super().visit_literal_block(node)
```

Basics are simple enough: create own writer, and this writer should use your own translator.

## How does `NodeVisitor` works

`NodeVisitor` works in following way (when rendering contents, there are many other node visitors that work in other ways — I guess).

Visiting a node is implemented in `Node.walkabout` method.

1. Revelant method names are generated from node class name. If node class name is `foo` two method names are generated `visit_foo` and `depart_foo`, if such methods are not found on `NodeVisitor` default method is called.
2. First `visit` method is called.
3. Then `walkabout` method is called for every child of current node.
4. `depart` method is called

You can suppress almost every step of this algorithm by raising a proper exception from `visit` method. If you want to skip rendering of children nodes just raise `SkipNode` exception.

# CHAPTER 19

## Connecting remote display (projector) to `i3`

It turns out to be suprisingly easy `randr` and `i3` are integrated well enough.

I have decided that I'll add a workspace dedicated for projector.

`i3.config` snippet:

```
# switch to workspace
bindsym $mod+p workspace 11

bindsym $mod+Shift+p move container to workspace 11

workspace 11 output VGA-1
```

To enable display issue:

```
xrandr --output VGA-1 --auto --right-of LVDS-1
```

To disable:

```
xrandr --output VGA-1 --off
```

CHAPTER 20

# How to create password hashed in Linux `/etc/shadow` format (crypted password)

To (easily) create passwords crypted with `crypt(3)`, just use `mkpasswd` program, on debian it is in `whois` package.

**Note:** This observation might be obvious, but before I learned about this program I ended up creating my own implementation of it using python `crypt` module.

**Note:** If you wonder why `mkpasswd` is in `whois` package, here is the answer.

CHAPTER 21

# How to get a table size in `psql`

If you have full `pgadmin3` you'll get very nice table statistics, but if all you have is `psql`, matter is more complicated.

You'll need to execute this SQL:

```
SELECT
    table_name,
    pg_size_pretty(table_size) AS table_size,
    pg_size_pretty(indexes_size) AS indexes_size,
    pg_size_pretty(total_size) AS total_size
FROM (
    SELECT
        table_name,
        pg_table_size(table_name) AS table_size,
        pg_indexes_size(table_name) AS indexes_size,
        pg_total_relation_size(table_name) AS total_size
    FROM (
        SELECT ('"' || table_schema || '"."' || table_name || '"') AS table_name
        FROM information_schema.tables
    ) AS all_tables
    ORDER BY total_size DESC
) AS pretty_sizes
```

CHAPTER 22

# CGI on Nginx or how to run man2html on debian

I don't enjoy reading manpages, `man` has old clunky interface, however do enjoy reading documentation in html, some good people developen `man2html` tool that is a `CGI` script that serves man pages in a browser.

This is why in 2015 year I spend part of an evening trying to configure modern web server to run CGI scripts. This might not be a suprise, that nginx doesn't serve CGI by default. It can serve `FastCGI`, which is a streamlied version of CGI, that unfortunately is incompatible with plain old CGI scripts.

However there were some other good people that wrote `fcgiwrap`, which is a a wrapper that talks FCGI protocol, and then launches plain old `cgi` scripts.

So to launch `man2html` on Debian, you'll need to:

1. Install required software: `aptitude install man2html nginx-full fcgiwrap`
2. There is a very good example fcgi config at `/usr/share/doc/fcgiwrap/examples/nginx.conf`, so just copy it to: `/etc/nginx/fcgiwrap.conf`
3. JUst include `/etc/nginx/fcgiwrap.conf` in your server config.

CHAPTER 23

# Problems with nfs shares on Debian (in case of Vagrant)

If you are having problems with running Vagrant box with NFS shares, here are possible solutions.

To install nfs you should need only to: `apt-get install portmap nfs-kernel-server`.

If vagrant whines with:

```
It appears your machine doesn't support NFS, or there is not an
adapter to enable NFS on this machine for Vagrant. Please verify
that `nfsd` is installed on your machine, and try again. If you're
on Windows, NFS isn't supported. If the problem persists, please
contact Vagrant support.
```

- Check if there is `nfsd` entry in `/proc/filesystems` if there is no such entry, or there is only `nfs` or `nfs4` entry something is wrong with your host.
- Maybe `nfs` kernel module is not loaded? Try `modprobe nfs; modprobe nfsd` and see if it helps,
- Maybe you forgot to install rules for `nfs` to your firewall?

  Check if you can telnet to:

  - Port 111 on your machine from your machint
  - Port 2049 on your machine from your machint
  - Port 111 on your machine from your vagrant box
  - Port 2049 on your machine from your vagrant box
- If you disabled `ufw` it might still screw your network communication, so:

  ```
  ufw enable
  ufw default allow
  ufw disable
  ```

  In my case after this and after restarting `nfs-kernel-server` everything worked like a charm.

- On debian I had a weird error in which nfs didn't start, because there were no exports defined, which in turn caused Vagrant to bail out with `"It appears your machine doesn't support NFS... "`. This can be diagnosed if:

```
root@karmapachyenno:~# systemctl status nfs-kernel-server.service
 nfs-kernel-server.service - LSB: Kernel NFS server support
   Loaded: loaded (/etc/init.d/nfs-kernel-server; bad; vendor preset: enabled)
   Active: active (exited) since Tue 2015-12-29 17:13:42 CET; 33min ago
     Docs: man:systemd-sysv-generator(8)
  Process: 949 ExecStart=/etc/init.d/nfs-kernel-server start (code=exited,␣
↪status=0/SUCCESS)

Dec 29 17:13:42 karmapachyenno systemd[1]: Starting LSB: Kernel NFS server␣
↪support...
Dec 29 17:13:42 karmapachyenno nfs-kernel-server[949]: Not starting NFS kernel␣
↪daemon: no exports. ... (warning).
Dec 29 17:13:42 karmapachyenno systemd[1]: Started LSB: Kernel NFS server support.
```

  This:

  - Can be fixed by adding any exports to `/etc/exports`.
  - I fixed it by `modprobe nfs; modprobe nfsd`, then running vagrant, which will add `/etc/exports`, then reloading `kernel-server` and restarting vagrant.

CHAPTER 24

# New Java Features

I decided to refresh my Java knowledge (last version I used was java 1.6), and because I learn by coding (much better than I learn by reading) here are code samples I prepared.

**Note:** This might be updated.

## Java 1.7

**Note:** All examples were actually made on Java `1.8` JVM.

I used this as a reference list of changes ( Oracle comparison is way to comprehensive).

### `java.nio.path` package

Java `File` class is awful, but a very nice api was introduced in this version of Java.

Package `java.nio.path` has a very nice tutorial by Oracle, so I won't describe it here.

Here is my (very simple) example, it works similarly to Disk Usage Analyzer. or `du` command on Linux, that is: it summarizes disk usage for a folder.

To do this I just needed to implement a `FileVisitor` instance, and then pass it to `Files.walkFileTree`.

#### Code Highlights

Most of the logic is in `Visitor` that subclasses `FileVisitor`, this class is used to traverse whole directory tree. Inside this instance we keep track of where in the directory tree we are, by using a stack.

```
Queue<Long> fileSizes = Collections.asLifoQueue(new ArrayDeque<>());
```

Each entry in the stack corresponds to a parent directory of currently processed path, and each contains a total size of that directory.

To add size of current object to size of the parent following code is used:

```
private void pushSize(long size){
   long lastSize = fileSizes.poll();
   lastSize+=size;
   fileSizes.add(lastSize);
}
```

```
@Override
public FileVisitResult preVisitDirectory(Path dir, BasicFileAttributes attrs) throws␣
↪IOException {
    fileSizes.add(0L);
    return FileVisitResult.CONTINUE;
}

@Override
public FileVisitResult postVisitDirectory(Path dir, IOException exc) throws␣
↪IOException {
    long dirSize = fileSizes.poll();
    pushSize(dirSize);
    if (maxDepthToDisplay<0 || maxDepthToDisplay >= fileSizes.size()) {
        System.out.println(level(fileSizes.size()) + dir + " " +␣
↪humanReadableByteCount(dirSize, true));
    }
    return FileVisitResult.CONTINUE;
}

@Override
public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) throws␣
↪IOException {
    if (Files.isRegularFile(file)) {
        pushSize(Files.size(file));
    }
    return FileVisitResult.CONTINUE;
}

@Override
public FileVisitResult visitFileFailed(Path file, IOException exc) throws IOException
↪{
    return FileVisitResult.CONTINUE;
}
```

### Complete example

```
import java.io.IOException;
import java.nio.file.*;
import java.nio.file.attribute.BasicFileAttributes;
import java.util.ArrayDeque;
import java.util.Collections;
import java.util.Queue;
```

```
public class PathExamples {

    private static class Visitor extends SimpleFileVisitor<Path>{

        long maxDepthToDisplay;

        Queue<Long> fileSizes = Collections.asLifoQueue(new ArrayDeque<>());

        protected Visitor(long maxDepthToDisplay) {
            super();
            this.maxDepthToDisplay = maxDepthToDisplay;
            fileSizes.add(0L);
        }

        private void pushSize(long size){
            long lastSize = fileSizes.poll();
            lastSize+=size;
            fileSizes.add(lastSize);
        }

        private String level(int depth){
            StringBuilder sbr = new StringBuilder();
            for (int ii = 0; ii < depth; ii++) {
                sbr.append(" ");
            }
            return sbr.toString();
        }

        /**
         * http://stackoverflow.com/a/3758880
         */
        public static String humanReadableByteCount(long bytes, boolean si) {
            int unit = si ? 1000 : 1024;
            if (bytes < unit) return bytes + " B";
            int exp = (int) (Math.log(bytes) / Math.log(unit));
            String pre = (si ? "kMGTPE" : "KMGTPE").charAt(exp-1) + (si ? "" : "i");
            return String.format("%.1f %sB", bytes / Math.pow(unit, exp), pre);
        }

        @Override
        public FileVisitResult preVisitDirectory(Path dir, BasicFileAttributes attrs)␣
↪throws IOException {
            fileSizes.add(0L);
            return FileVisitResult.CONTINUE;
        }

        @Override
        public FileVisitResult visitFileFailed(Path file, IOException exc) throws␣
↪IOException {
            return FileVisitResult.CONTINUE;
        }

        @Override
        public FileVisitResult postVisitDirectory(Path dir, IOException exc) throws␣
↪IOException {
            long dirSize = fileSizes.poll();
            pushSize(dirSize);
            if (maxDepthToDisplay<0 || maxDepthToDisplay >= fileSizes.size()) {
```

```
                System.out.println(level(fileSizes.size()) + dir + " " +␣
↪humanReadableByteCount(dirSize, true));
            }
            return FileVisitResult.CONTINUE;
        }

        @Override
        public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) throws␣
↪IOException {
            if (Files.isRegularFile(file)) {
                pushSize(Files.size(file));
            }
            return FileVisitResult.CONTINUE;
        }
    }

    public static void main(String[] args) throws IOException {
        Path path = Paths.get(args[0]);
        Files.walkFileTree(path, new Visitor(3));

    }
}
```

## Fork Join Framework

Java 1.7 has very nice Fork-Join Framework, that allows one to dynamically split between cores, but here is the catch: we don't know the amount of work needed upfront.

I have decided to try this framework, to (once again) summarize size of a directory tree.

This framework is nicely explained in the tutorials.

Overall I'm surprised with the performance of both naive and parralel implementation, naive version takes 6 seconds (when ran on my 150GB home directory), while parralel takes 3sec.

### Code Highlights

Task result is a POJO object, containing path, it's size, information whether this path is a directory, and sub directories (if any). Here is the definition:

```
private static class WalkFileResult{
      public final Path dirPath;
      public final boolean isDir;
      public final long dirSize;
      public final List<WalkFileResult> subdirs;

      public WalkFileResult(Path dirPath, long dirSize) {
          this(dirPath, dirSize, Collections.emptyList());
      }

      public WalkFileResult(Path dirPath, long dirSize, List<WalkFileResult> subdirs)
↪{
          super();
          this.dirPath=dirPath;
          this.dirSize=dirSize;
          this.isDir=Files.isDirectory(dirPath);
```

```
            this.subdirs=subdirs;
        }
    }
```

Single task has following logic:

1. If we are looking at a file, calculate file size and return it.
2. If we are looking at a directory, create task for each child of the directory, execute these tasks in parrarel and then calculate the size.

In Java it is:

```
@Override
protected WalkFileResult compute() {
    try {
        if (Files.isRegularFile(currentPath)) {
            return new WalkFileResult(currentPath, Files.size(currentPath));
        } else if(Files.isDirectory(currentPath)) {
            List<WalkFileTask> subTasks = getSubtasks();
            return joinOnSubtasks(subTasks);
        }
    }catch (IOException | InterruptedException e){
        throw  new RuntimeException(e);
    }catch (ExecutionException e){
        throw new RuntimeException(e.getCause());
    }
    return new WalkFileResult(currentPath, 0L);
}

 private List<WalkFileTask> getSubtasks() throws IOException{
          // This visitor just returns immediate children of current path
          Visitor v = new Visitor(currentPath);
          Files.walkFileTree(currentPath, v);
          return v.subtasks;
      }

     private WalkFileResult joinOnSubtasks(List<WalkFileTask> subTasks) throws␣
↪ExecutionException, InterruptedException {
          long size = 0;
          List<WalkFileResult> subDirs = new ArrayList<>();
          for (WalkFileTask res: invokeAll(subTasks)){
              WalkFileResult wfr = res.get();
              size+=wfr.dirSize;
              if (wfr.isDir){
                  subDirs.add(wfr);
              }
          }
          return new WalkFileResult(currentPath, size, subDirs);
      }
```

### Complete example

```
package examples;

import javax.sound.midi.SysexMessage;
import java.io.IOException;
```

```
import java.nio.file.*;
import java.nio.file.attribute.BasicFileAttributes;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ForkJoinPool;
import java.util.concurrent.ForkJoinTask;
import java.util.concurrent.RecursiveTask;



public class ForkJoinPath{

    private static class WalkFileResult{
        public final Path dirPath;
        public final boolean isDir;
        public final long dirSize;
        public final List<WalkFileResult> subdirs;

        public WalkFileResult(Path dirPath, long dirSize) {
            this(dirPath, dirSize, Collections.emptyList());
        }

        public WalkFileResult(Path dirPath, long dirSize, List<WalkFileResult>␣
↪subdirs) {
            super();
            this.dirPath=dirPath;
            this.dirSize=dirSize;
            this.isDir=Files.isDirectory(dirPath);
            this.subdirs=subdirs;
        }
    }

    private static class WalkFileTask extends RecursiveTask<WalkFileResult>{

        private static class Visitor extends SimpleFileVisitor<Path>{

            public final Path root;

            public List<WalkFileTask> subtasks = new ArrayList<>();

            protected Visitor(Path root) {
                super();
                this.root = root;
            }

            @Override
            public FileVisitResult preVisitDirectory(Path dir, BasicFileAttributes␣
↪attrs) throws IOException {
                if(Files.isSameFile(dir, root)){
                    return FileVisitResult.CONTINUE;
                }
                if (Files.isReadable(dir)) {
                    subtasks.add(new WalkFileTask(dir));
                }
                return FileVisitResult.SKIP_SUBTREE;
            }
```

```
            @Override
            public FileVisitResult visitFile(Path file, BasicFileAttributes attrs)␣
↪throws IOException {
                if (Files.isReadable(file) && Files.isRegularFile(file)) {
                    subtasks.add(new WalkFileTask(file));
                }
                return FileVisitResult.CONTINUE;
            }

            @Override
            public FileVisitResult visitFileFailed(Path file, IOException exc) throws␣
↪IOException {
                return FileVisitResult.CONTINUE;
            }
        }

        private final Path currentPath;

        public WalkFileTask(Path currentPath) {
            super();
            this.currentPath=currentPath;
        }

        private List<WalkFileTask> getSubtasks() throws IOException{
            // This visitor just returns immediate children of current path
            Visitor v = new Visitor(currentPath);
            Files.walkFileTree(currentPath, v);
            return v.subtasks;
        }

        private WalkFileResult joinOnSubtasks(List<WalkFileTask> subTasks) throws␣
↪ExecutionException, InterruptedException {
            long size = 0;
            List<WalkFileResult> subDirs = new ArrayList<>();
            for (WalkFileTask res: invokeAll(subTasks)){
                WalkFileResult wfr = res.get();
                size+=wfr.dirSize;
                if (wfr.isDir){
                    subDirs.add(wfr);
                }
            }
            return new WalkFileResult(currentPath, size, subDirs);
        }

        @Override
        protected WalkFileResult compute() {
            try {
                if (Files.isRegularFile(currentPath)) {
                    return new WalkFileResult(currentPath, Files.size(currentPath));
                } else if(Files.isDirectory(currentPath)) {
                    List<WalkFileTask> subTasks = getSubtasks();
                    return joinOnSubtasks(subTasks);
                }
            }catch (IOException | InterruptedException e){
                throw  new RuntimeException(e);
            }catch (ExecutionException e){
                throw new RuntimeException(e.getCause());
```

```
            }
            return new WalkFileResult(currentPath, 0L);
        }
    }

    private static String level(int depth){
        StringBuilder sbr = new StringBuilder();
        for (int ii = 0; ii < depth; ii++) {
            sbr.append(" ");
        }
        return sbr.toString();
    }

    /**
     * http://stackoverflow.com/a/3758880
     */
    public static String humanReadableByteCount(long bytes, boolean si) {
        int unit = si ? 1000 : 1024;
        if (bytes < unit) return bytes + " B";
        int exp = (int) (Math.log(bytes) / Math.log(unit));
        String pre = (si ? "kMGTPE" : "KMGTPE").charAt(exp-1) + (si ? "" : "i");
        return String.format("%.1f %sB", bytes / Math.pow(unit, exp), pre);
    }


    private static void printResult(WalkFileResult wfr, int depth){
        if (depth >= 3){
            return;
        }

        System.out.println(level(depth) + wfr.dirPath + " " +␣
↪humanReadableByteCount(wfr.dirSize, false));
        for (WalkFileResult child: wfr.subdirs){
            printResult(child, depth+1);
        }


    }

    public static void main(String[] args) throws ExecutionException,␣
↪InterruptedException {
        long start = System.nanoTime();
        Path path = Paths.get(args[0]);
        ForkJoinPool pool = new ForkJoinPool();
        WalkFileTask task = new WalkFileTask(path);
        pool.execute(task);

        printResult(task.get(), 0);
        double duration = (System.nanoTime() - start) * 1E-9;
        System.out.println(duration);



    }
}
```

### Notable mentions

There is also very nice `WatchService`, that allows to monitor filesysem for file changes.

## Java 1.8

### Streams and Lambdas

Third attempt to do the same task: to summarize size of a directory tree.

This times using `Streams` and `Lambdas`. Solution is most concise, but least readable IMO. Also, while other solutions transparently handle unreadable directories, this one explodes with AccessDenied exception.

#### Code Highlights

Result POJO:

A function that can throw an exception:

```
@FunctionalInterface
public interface CheckedFunction<T, R> {
    R apply(T t) throws IOException;
}
```

A lambda that calculates file size:

```
CheckedFunction<Path, DirSize> mapper = (Path p) -> new DirSize(p,
    Files.walk(p).parallel()
    .filter(Files::isReadable)
    .mapToLong(StreamExamples::safeSize).sum());
```

A stream that walks over FS calculating size of each directory:

```
Files.walk(path, 3)
  .parallel().filter(Files::isDirectory).filter(Files::isReadable).map(
      (Path p) -> {
          try {
              return mapper.apply(p);
          } catch (IOException e) {
              return new DirSize(p, -1);
          }
      }).forEach(
        (DirSize d) ->
          System.out.println(d.path + " " + humanReadableByteCount(d.size, false)));
```

#### Complete example

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

/**
```

```
 * Created by jb on 12/3/15.
 */
public class StreamExamples {

    private static class DirSize{
        public final Path path;
        public final long size;

        public DirSize(Path path, long size) {
            this.path = path;
            this.size = size;
        }
    }

    @FunctionalInterface
    public interface CheckedFunction<T, R> {
        R apply(T t) throws IOException;
    }

    public static long safeSize(Path p){
        try {
            return Files.size(p);
        } catch (IOException e) {
            return 0;
        }

    }

    public static String humanReadableByteCount(long bytes, boolean si) {
        int unit = si ? 1000 : 1024;
        if (bytes < unit) return bytes + " B";
        int exp = (int) (Math.log(bytes) / Math.log(unit));
        String pre = (si ? "kMGTPE" : "KMGTPE").charAt(exp-1) + (si ? "" : "i");
        return String.format("%.1f %sB", bytes / Math.pow(unit, exp), pre);
    }

    public static void main(String[] args) throws IOException {
        long start = System.nanoTime();
        Path path = Paths.get(args[0]);

        CheckedFunction<Path, DirSize> mapper = (Path p) -> new DirSize(p,
                Files.walk(p, Integer.MAX_VALUE).parallel().filter(Files::isReadable).
↪mapToLong(StreamExamples::safeSize).sum());

        Files.walk(path, 3)
            .parallel().filter(Files::isDirectory).filter(Files::isReadable).map(
                (Path p) -> {
                    try {
                        return mapper.apply(p);
                    } catch (IOException e) {
                        return new DirSize(p, -1);
                    }
                }).forEach((DirSize d) -> System.out.println(d.path + " " +␣
↪humanReadableByteCount(d.size, false)));

        double duration = (System.nanoTime() - start) * 1E-9;
        System.out.println(duration);
    }
```

```
}
```

CHAPTER 25

# How to disable subpixel rendering in Jetbrains products

If you want to disable subpixel rendering (another term is font anitaliasing) in any JetBrains product (like: Idea, Pycharm and so on), you need to:

1. Go to `bin` dir of install folder, find file that ends with `*vmoptions` (`idea.vmoptions`, `pycharm.vmoptions`).
2. There should be a line `-Dawt.useSystemAAFontSettings=lcd` in this file, delete this line.
3. Add line: `-Dprism.lcdtext=false`.

These two lines define properties passed to Java Virtual Machine that control antialiasing, both of them are undocumented, so they might not work in future JVM releases (I tested on `1.8.51`). These JVM settings seem to clash with either JetBrains settings (that override these) and Gnome settings, so disabling them makes sense.

CHAPTER 26

# Linux dynamically assigned ports

So you want to assign your application a `high` port number, maybe you do some testing (and you'll launch many instances of servers), maybe you want to set some service on unusual port (which is ok in some cases).

What you don't want is your selected port to clash with the range of ports that are automatically assigned to outgoing connections. This range is sadly OS dependent, IANA defines it as: 49152–65535, however linux chooses different range, that can be read by: ‘‘cat /proc/sys/net/ipv4/ip_local_port_range ‘‘.

For more information see this SO post.

CHAPTER 27

# Asyncio servers in Python

From what I read: "normal" architecture for a web-server is that one assings single thread to a client, this is mostly OK, but after certain number of clients your performance drops (because of thread memory overhead, context switching costs, and other things). Specific limit is hart to guess, and depends on OS, app, hardware and so on.

Asynchronous servers were supposedly a solution to this problem, but I didn't really believe this. From what I understood you could always buy more frontend boxes and scale it this way.

But today I actually written a async server, this server is very simple, has no error control (it took me less than an hour, including reading manuals), protocol is very simple:

0. Protocol is line based
1. When any client sends a line of text to the server
2. This line is sent to every another client.

It was written in Python 3.5, server uses PEP-492 and the `asyncio` library. Go ahead and read this PEP, there is even a working example (that I based upon).

I didn't do extensive tests, but it seems that this server handles 4000 connections **easily** on single core, throughput is about 40K messages per second (still on single core).

---

**Note:** I wouldn't rely on these numbers much, to do a proper tests I would have to:

1. Add error handling to the server.
2. Change the protocol to acknowledge results.
3. Write some proper tests.

Tests are based on running 4 processes each spawning 1000 sockets to the server, and then writing one message per 100ms in yet another thread. Results were observed by looking at the output :)

---

Server works as follows:

1. Keeps a set of all connections (sockets in essence)
2. Each relieved line is sent to all sockets

Server code is here:

```
# -*- coding: utf-8 -*-

import asyncio
import threading
from asyncio.streams import StreamReader, StreamWriter

import time

from multiprocessing import Process, Lock, Condition


class AsynChat(object):

  def __init__(self, port, client_may_end_connection=False):
    self.clients = set()
    self.port = port
    self.client_may_end_connection=client_may_end_connection
    self.__loop = None

  async def handle_connection(self, reader:StreamReader, writer:StreamWriter):
    self.clients.add(writer)
    while True:
      data = await reader.readline()
      print(data.strip())
      if self.client_may_end_connection and data.strip() == b'END':
        print("STOP")
        self.__loop.stop()
      for w in self.clients:
        if w == writer:
          continue
        w.write(data)
        await w.drain()
      if not data:
        if writer in self.clients:
          self.clients.remove(writer)
          try:
            writer.write_eof()
          except OSError:
            pass # Sometimes it explodes if socket was closed very soon, didn't␣
↪investigate
          return

  async def echo_server(self):
      await asyncio.start_server(self.handle_connection, 'localhost', self.port)

  @classmethod
  def run_in_process(cls, *args, **kwargs) -> Process:
    c = Condition(Lock())
    def build_and_run(*args, **kwargs):
      ac = cls(*args, **kwargs)
      ac.run_loop(c)

    p = Process(target=build_and_run, args=args, kwargs=kwargs)
    p.start()
    with c:
      c.wait()
```

```
    return p

  def run_loop(self, c:Condition=None):
    self.__started = True
    self.__loop = asyncio.get_event_loop()
    self.__loop.run_until_complete(self.echo_server())

    def notif():
      with c:
        c.notify()

    try:
      if c:
        self.__loop.call_soon(notif)
      self.__loop.run_forever()
    finally:
      self.__loop.close()



if __name__ == "__main__":
  a = AsynChat(1234, True)
  a.run_loop()
```

CHAPTER 28

# How to discard connection in a fast way

Just a quick note, here is how to drop a TCP connection fast:

- Client sends `SYN` packet
- Server responds with `RST` packet, which promptly kills the connection without any state.

CHAPTER 29

# Some remarks about databases

**Note:** I was refreshing my knowledge on the broad subject of databases, mostly by reading Wikipedia articles, which resulted (apart from notes attached here) in creating this wikipedia book (in case some future problems with Wikipedia `you can try version hosted here`).

## Read anomalies Anomalies

What can happen if your database doesn't serialize transactions properly:

**Dirty read** When one transaction sees uncommitted data of another one

**Short example**:

We have two transactions T1 and T2.

- T1 reads value of a row R
- T2 writes new value of row R
- T1 sees a change in row value

**Non-Repeatable Read** When value of a row changes during a transaction.

**Short example**:

We have two transactions T1 and T2.

- T1 reads value of a row R
- T2 writes new value of row R
- T2 commits <- **Difference from Dirty Read**
- T1 sees a change in row calue

**Phantom read** When a result of query changes during the transaction.

> **Note:** Database without Non-Repeatable Reads cam still suffer from phantom reads, non-repeatable reads apply when other transaction updates or deletes records while phantom also apply to inserted rows.

**Short example**:

- T1 executes a query: `SELECT AVG(value) FROM account WHERE ...GROUP BY ...`
- T2 executes: `INSERT .... INTO account`
- T2 commits
- T1 executes a query: `SELECT AVG(value) FROM account WHERE ...GROUP BY ...` and gets different results.

**Write Skew** This is an anomaly for MVCC databases, which occours each transaction works on a snapshot of database, and don't see changes done by each other.

> **Note:** This anomaly is not defined in SQL standard, as SQL standard had locking databases in mind.

**Short example**:

Let's consider a banking application, with following constraint: balance of account must be non-negative, but we don't store it explicitly, it is just calculated by `SELECT SUM(t.value) FOR account .... INNER JOIN transaction as t ON ...`

- At the start of the transaction balance of account A is 100PLN
- Transaction T1 starts
- Transaction T2 starts
- Transaction T1 adds a transaction for A that withdrawns 100PLN (which is valid as constraint is held)
- T1 Commits
- Transaction T2 does the same (which still is valid as constraint is held inside a snapshot for T2 — constraint is not held "outside of" this snapshot).

> **Note:** Wikipedia says that you might resolve this issue by explicitly writing to a dummy row just to force write-write confilct.

## Transaction isolation Levels

**Serializable** Highest transaction isolation. Transactions are executed *as if* they were executed serially.

No read anomalies can occur.

In database that uses locks it requires to put locks on:

- Every row you write to
- Every row you read
- Range lock for every query (for example lock all records for which specified condition is true).

These locks are held until the end of the transaction.

**Snapshot Isolation** This transaction isolation level works as follows: Each transaction sees database in a (consistent) state the database was at the beginning of the transaction (with any changes it did).

This is very different from Serializable, as it allows "Write Skew" anomalies.

---

**Note:** This isolation level is not defined in SQL standard.

---

You can define Serializable isolation on top of Snapshot Isolation relatively easy by:

- This can be done relatively easy by the DMBS, see this article:

  Cahill, M. J., Röhm, U., & Fekete, A. D. (2009). Serializable isolation for snapshot databases. ACM Transactions on Database Systems, 34(4), 1–42. doi:10.1145/1620585.1620587

- Writing proper code that introduces artificial write conflicts between data.

  ---

  **Note:** These conflicts are artificial, because they exist only to introduce write conflicts, which will abort transaction that tries to write to the database as the second one.

  ---

**Repeatable Read** This isolation level reading a row will always produce the same value, even of these rows were changed by other transactions. However query results are can change during the transaction (especially for aggregate queries).

In database that uses locks it requires to put locks on:

- Every row you write to
- Every row you read

These locks are held until the end of the transaction.

**Read Committed** In this isolation level transaction doesn't see changes made by another uncommitted transactions, however it may see changes made by commited transactions.

In database that uses locks it requires to put locks on:

- Every row you write to
- Every row you read

Write locks are held until the end of the transaction, however read locks are released after each select.

**Read Uncommitted** In this level you can see uncommitted data sent by saved by other transactions.

## Serializability

We have a some set of concurrent transactions, these transactions are serializable, if one can produce a schedule containing these transactions executed serially in some order.

Serializability is important because:

If DBMS checks if database is in a consistent state **after each of the transactions**, and transactions are serializable — it means that the database is in a consitent state after all transactions.

CHAPTER 30

# MVCC in postgresql

**Note:** This is based on my lecture on the databases, which (is still available in Polish)

There are two ways to implement proper transaction isolation:

- First is by using locking. In this padagrim whenever transaction reads data a lock is issued, and any write to that data will wait until reading transaction finishes (this might be simplified).
- Second is by using MVCC — that is multi version concurrency. It works as follows: each transaction sees database in a state **at the time the transaction**, so reads and writes don't need to wait for each other (there is a problem with write skew anomaly, which is solved by the postgresql 9.1 and newer.

## How does MVCC work

### Start of the transaction

When transaction starts following things happen (or may happen depending on isolation level):

- Transaction is assigned a `txid` a transaction ID, transaction id's are ordered 32 bit integers (that may wrap around at some point in time, but Postgres handles it).
- `txid``s of all committed transactions are stored (possibly in a more efficient way than storing all ``txids)`

### Data constraints

Each row contains couple of magic columns:

**`xmin`** This is the `txid` of transaction that inserted this row

**`xmax`** This is the `txid` of transaction that deleted this column

**cmin, cmax** Index of statement in transaction that added/deleted that row

### Basic operations

**INSERT** When a transaction inserts a row it `xmin` is set to `txid` of this transaction.

**DELETE** When a transaction deletes a row it just sets `xmax` to it's `txid`

**UPDATE** Updates are replaced with a delete and insert pair.

### Data visibility

Row is visible for transaction `txid` if (all statements must be true):

- It's `xmin < txid` (row was inserted by a transaction before this one).
- Transaction `xmin` is commited (in case of Read Commited isolation level), or `xmin` was commited before start of current transaction (other isolation levels)
- It's `xmax` is empty or `xmax > txid` (row was deleted by a transaction that started after this one).

In case of transaction that issue multiple statements `cmin`, `cmax` are used for example to have a cursor that consistently iterates over a table, even if the same transaction alters the table.

## VACUUM

Data can't be deleted from disk immediately in databases using MVCC, because ongoing transactions might not 'see' the delete, and still need to access deleted row. Data can be deleted only after all transactions with `txid` lower row's than `xmax` have either committed or have been rolled back.

Postgresql does this (more or less) automatically, but you might call `VACUUM` by hand if you need to reclaim space (this space will not necessarily be freed to the OS, rather it will be accessible for new inserts).

## Sources

- BRIUCE MOMJIAN `MVCC Unmasked`: http://momjian.us/main/writings/pgsql/mvcc.pdf (I have also cached it `locally`, due to permissive CC license)
- Serializable Snapshot Isolation on Postgresql Wiki, (due to even more permissive license it is also cached `locally`.
- My old lectures: http://db.fizyka.pw.edu.pl/~bzdak/bazy_danych_ed_20/wyklad10/wyk10.html#mvcc-w-postgresq
- Wikibook I have collected (in case some future problems with Wikipedia `you can try version hosted here`).

CHAPTER 31

# Should you use BRTFS for your laptop?

TL; DR; Probably not.

Recently I re-installed by Debian desktop, and enabled full disk encryption, (and LVM for that matter). I was also toying with the idea of using BTRFS, which has many features I always wanted to have, including:

- Fast fs-level compression
- Optional, out of write path, deduplication.
- Very nice features including super-easy resizing (adding more GBs to your file system takes seconds)
- Copy on write semantics

BTRFS is stable, yet I believe is unusable for a most of people. If you are not a linux nerd, don't try, if you are you might, but remember:

- Do backups, you should always do backups, especially if you use encrypted filesystems on ssd drives.
- Do yourself a favour and buy an USB stick, and burn there a linux live-cd.
- BTRFS **will** suprise you.

Here are two nasty suprises I had.

## BTRFS COW doesnt play well with some usage patterns

If you are a linux nerd, you probably have some virtual machines, COW (copy on write) doesn't play with them. Well everywhere where you have large files that are written to often, it doesn't play well with COW.

COW can be disabled on directory level, do to this issue *chattr +C /dir* command, this will disable COW for everything under */dir*. Keep in mind that it works **only on empty files and directories**, turning off COW on a file with data, has **undefined behaviour**, and most often is bad. Turning off COW for directories with files is safe, but existing files will have COW enabled.

In my case Virtualbox failed with very non-obvious errors, before I disabled COW.

## BTRFS needs garbage colection (or something similar)

Basically BTRFS kind-of lies to the OS when reporting free space. You can have full filesystem and yet BTRFS will report 100GB of free space. I don't try to understand what it is, BTRFS wiki says things like: "The primary purpose of the balance feature is to spread block groups accross all devices so they match constraints defined by the respective profiles."

Usable free space on your disk can be seen using *btrfs fi show* Which will display: total system size, and how much space is currently used by btrfs.

In my case:

```
# btrfs fi show

Label: none  uuid:
    Total devices 1 FS bytes used 613.29GiB
    devid    1 size 745.06GiB used 649.06GiB path /dev/mapper/
```

I have 745GB partition, of which 649GB is used by btrfs, however, only 613 is used for files.

This can be fixed by issuing something like:

```
btrfs balance start -dusage=<magic number> / &
```

<<magic number> is a percentage value, and BTRFS will try to "rebalance" only chunks filled in less than this <<magic number>>, the bigger number you put there the longer will it take, and the more space you'll reclaim. You can start with percentage value of used space you have on your drive.

CHAPTER 32

# How to deploy django application on debian server (UWSGI version)

## Install proper version of python

On debian I'd discourage using `system` python for deployment — mostly becaluse they tend to upgrade minor python versions without notice, which sometimes breaks `C ABI` in installed virtualenvs.

So either roll your own `deb` files that install pythons somewhere in `/usr/local` or compile python on server (if you frown on having development tools on your server roll `debs`).

---

**Note:** For development environment Pythonz is a nice tool to compile (and manage) many versions of python.

If you want deploy your server by hand just download and compile python.

If you are into automatic deployment (and you **should** be)

---

## Assumptions about the system

I'll assume that you will configure your system in following way:

- Django application will be using `www-client` user
- Code will be inside `/home/www-client/repo`
- There will be a django generated uwsgi file in `/home/www-client/repo/webapp.uwsgi`
- Virtualenv will be in `/home/www-client/repo/venv`.

## Install your application into virtualenv

You know how to do that don't you?

Now you can tests whether your setup is correct, just run

```
./manage.py runserver
```

and see if you can connect to your application.

## Install uwsgi into your virtualenv

Install `uwsgi` **into your virtualenv** from `pip`. Now you can run your application using uwsgi:

```
uwsgi --http :8000 --module webapp.wsgi
```

I strongly discourage you from using uwsgi bundled with system.

You can configure `uwsgi` using a variety of ways, most of which are better than using a command line arguments :). For example you can create an ini file named `uwsgi.ini`:

```
module=webapp.wsgi
pythonpath=/home/www-client/webapp
http=8000
```

And then start uwsgi using: `uwsgi --ini uwsgi.ini`.

## Use systemd to launch your applicaiton

Now use systemd to launch the application.

Systemd is a very nice init system that is becoming a standard in most recent distributions (it's even on Debian stable).

If your distribution has no systemd you can use supervisord (which is even on debian oldstable), tutorial to deploy django with it *is here*.

So create `/home/webapp/uwsgi.ini` file with following contents:

```
module=webapp.wsgi
http=8000
pythonpath=/home/www-client/repo
```

Create a `webapp.service` file and put it in `/etc/systemd/system/`, file should have following contents:

```
[Unit]
Description=Description
After=syslog.target

[Install]
WantedBy=multi-user.target

[Service]
# What process to start
ExecStart=/home/www-client/venv/bin/uwsgi --ini /home/www-client/uwsgi.ini
# What user chown to
User=www-client
# Working directory
WorkingDirectory=/home/www-client/webapp
Restart=always
```

```
# Kill by SIGQUIT signal --- this is what asks wsgi to die nicely
KillSignal=SIGQUIT
# Notify type, in this type uwsgi will inform systemd that it is ready to handle␣
↪requests
Type=notify
StandardError=syslog
NotifyAccess=all
```

Then:

```
sudo systemctl --system enable webapp
sudo systemctl start webapp
```

Now you should have a working uwsgi configuration, which is reachable at `localhost:8000`.

## Connect `uwsgi` and `nginx`

**Note:** This part is more or less ripoff from: http://uwsgi-docs.readthedocs.org/en/latest/tutorials/Django_and_nginx.html

In this part we will put `uwsgi` behind `nginx` server.

Add following sections to nginx configuration:

```
upstream django {
    # server unix:///path/to/your/mysite/mysite.sock; # for a file socket
    server 127.0.0.1:8000; # for a web port socket (we'll use this first)
}

# configuration of the server
server {
    # the port your site will be served on
    listen      80;
    # the domain name it will serve for
    server_name .example.com; # substitute your machine's IP address or FQDN
    charset     utf-8;

    # max upload size
    client_max_body_size 75M;   # adjust to taste


    # Finally, send all non-media requests to the Django server.
    location / {
        uwsgi_pass  django;
        include     /etc/nginx/uwsgi_params; # the uwsgi_params file you installed
    }
}
```

Now you should see something on your server port `80`. To finalize our setup we need to create static and media directories.

## Connect ngingx and uwsgi via linux file sockets

Because of many (peformance, safety) reasons it is better to connect nginx with uwsgi via linux domain sockets.

First replace `uwsgi.ini` file with something like:

And also create `/home/www-client/sock/`.

This does the following: creates a socket in `$HOME/sock/webapp.sock`, sets its group ownership to `www-data` (which is user/group used by both `nginx` and `apache` on default debian configuration).

**Note:** Linux file sockets use normal file permissions, so `nginx` has to have read-write access to it.

Then replace:

```
upstream django {
    server 127.0.0.1:8001; # for a web port socket (we'll use this first)
}
```

with:

```
upstream django {
    server unix:///path/to/your/mysite/webapp.sock; # for a file socket
}
```

## Update media and static root

Now we'll update settings so static media is served by nginx.

### Update settings py

You'll need to update `STATIC_ROOT`, `STATIC_URL`, `MEDIA_ROOT`, `MEDIA_URL` settings of youe app.

Something along the lines:

```
MEDIA_ROOT = '/var/drigan-media'
MEDIA_URL = '/media/'
STATIC_ROOT = '/var/drigan-static'
STATIC_URL = '/static/'
```

### Update nginx

```
location /media  {
    alias /path/to/your/mysite/media;  # your Django project's media files - amend as␣
↪required
}

location /static {
    alias /path/to/your/mysite/static; # your Django project's static files - amend␣
↪as required
}
```

## Tweak uwsgi so it scales

You might want tweak `uwsgi` so it launches more processes/workers, but this well outside scope of this tutorial.

CHAPTER 33

# What to do when you get [Read error 4] when running `apt`

When you run `apt` and have following errors:

```
E: Read error - read (5: Input/output error)
E: The package lists or status file could not be parsed or opened.
```

This means that you are in serious trouble, it's not hopeless but serious.

This error is caused by `apt` having some problems reading some of it's config files. Too bad `apt` can't say in which file the problem is, and what is the problem.

Obviously there is no single answer on how to fix it, here are some hints.

1. Remove package cache149.202.188.158: `apt-get clean`. Will purge some stuff. Probably won't help but it is safe (yes rest of the commands isn't safe).
2. Remove package lists:

   ```
   sudo rm -rf /var/lib/apt/lists
   sudo mkdir /var/lib/apt/lists
   sudo mkdir /var/lib/apt/lists/partial
   ```

3. Restore old version of dpkg status files. This is **the** file where all information on packages installed is stored. Sometimes it is corrupted, however it has automatic backups. The file is `/var/lib/dpkg/status`, first thing to try would be:

   ```
   cp /var/lib/dpkg/status /var/lib/dpkg/status.broken
   cp /var/lib/dpkg/status.old /var/lib/dpkg/status
   ```

   This will restore previous version of this file, there are older ones in `/var/backup/dpkg.status*`, but probably try the next step first.

4. `apt` (or `aptitue`) also store it's copy of dpkg state, with additional information (like which package was installed by user, and which is a as a dependency). This can also be corrupted, so:

   ```
   cp /var/lib/apt/extended_states /var/lib/apt/extended_states.broken
   cp /var/backups/apt.extended_states.0 /var/lib/apt/extended_states
   ```

5. One of the reasons for corrupted files can be file-system or hard drive problems. If you reached this point please consider backing up your data, running fsck, and diagnosing you drive.
6. Some people reported success with manually editing status files.

There is also very extensive list of commands that might help you with diagnosing problems with apt: https://help.ubuntu.com/community/PackageManagerTroubleshootingProcedure I wouldn't recommend running them without understanding though.

CHAPTER 34

# How to encrypt a usb drive using cryptsetup

Actuall I didn't manage to set up it using console `cryptsetup` commands, however there is a very nice dommand `gnome-disks` (which comes in package `gnome-disk-utility`) that does everything automatically!

CHAPTER 35

# Cheap and fast way to create your own Maven repository on S3

Maven is de-facto standard build system for Java apps, it's a nice and mature piece of software.

To meaningfully use it you need to host your own maven repository — that allows you and your teammates to download and release software artifacts you use. Most of these repositories is written in Java, and is not very light — most probably you'd need to have a dedicated VPS, which is OK — unless you use this repository for hobby project that is in maintenenance mode and you touch it once a year.

Cheap and easy way to host Maven repositories is to use S3, there you pay only for storage, and transfer, which is preatty cheap.

Simple guide is here: https://github.com/spring-projects/aws-maven.

CHAPTER 36

# How to debug zsh startup time

My ZSH shell started to be slow, by slow I mean startup time was about ~5sec.

## Step 1: measure startup time:

```
time  zsh -i -c exit
```

## Step 2: Inject profiling code

At the beginning of your `.zshrc` add following: `zmodload zsh/zprof` and at the end add: `zprof`, this will load zprof mod and display what your shell was doing durung your initialization.

After start of the shell you'll see something along the lines:

```
num  calls                time                       self            name
-----------------------------------------------------------------------------------
 1)    1          49.02    49.02   26.87%     49.02    49.02   26.87%  a command
 2)  ....
```

Commands ZSH spent most of the time are at the top of the output.

In my case culprit was `compinit`.

## Step 3: Make compinit faster

Compinit is a function that initializes shell completion. After some googling I found out that the problem might be "too bit" `.zcompdump` file, in my case it was about `1mb`, so I deleted it (this file is autogenerated), and ZHS magically started to be faster.

After that I have added deleting this file to my user startup script.

## Step 4: Cleanups

Comment out or delete addions you made to `.zshrc` file.

Comments:

- If you have any comments please send me an e-mail to `jacekfoo@gmail.com`.

CHAPTER 37

# Indices and tables

- genindex
- modindex
- search