Setting up pyWPS in combination with uwsgi and nginx

virtual python environment

If you launch a python process with pyWPS, it should not use the main system python. Your python process might need dedicated libraries, for instance pandas. Installing such a library might conflict with other processes that use python and conflict with that version of the library directly, or via underlying dependencies.. Therefore pyWPS should it's own python instance, in what it called a *virtual environment*. In this tutorial this approach is explained.

WSGI web server and NGINX

For running web services this virtual environment should be owned by the web server that is used to run pywps. In this example WSGI, documented at http://pywps.wald.intevation.org/documentation/pywps-3.2/special/wsgi.html. In fact, WSGI is a bit outdated, and we advise to use gunicorn. Note that WSGI can be hosted by a web server like NGINX. WSGI will distribute load between threads on one computer node (like OpenMP in Fortran), while NGINX will distribute load over different computers (like MPI in Fortran) *.

example

As an example we will show the workings of the pyWPS at our dtvirt5 test machine with WSGI. (being a test server, here we do not run WSGI inside NGINX).

virtual env

```
/var/lib/wsgi/env/
```

pyWPS

```
/var/lib/wsgi/env/bin/wps
```

check python instance for pywps

```
which python
# /usr/bin/python
# do not install anything here

cd /var/lib/wsgi/env/bin/
source activate
# now we can activate the special python instance for pywps

which python
# /var/lib/wsgi/env/bin/python
# install packages for pywps here, e.g.:
# pip install pandas --upgrade

# leave virtual environment
deactivate
```

restarting

```
# automatic
/etc/init.d/uwsgi/apps-enabled

# force manual (required after each update of pywps codes)
sudo service uwsgi restart
```