Birdhouse Architecture

Carsten Ehbrecht

ehbrecht@dkrz.de

German Climate Computing Center (DKRZ)

October 2015

Outline

1. Motivation
2. Birdhouse Components
3. Birdhouse Builder
4. Deployment with Docker
5. Security and Interoperability

Overview

1 Motivation

2 Birdhouse Components

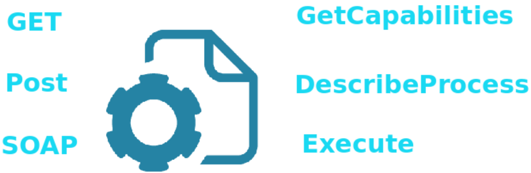
3 Birdhouse Builder
- Conda
- Buildout

4 Deployment with Docker

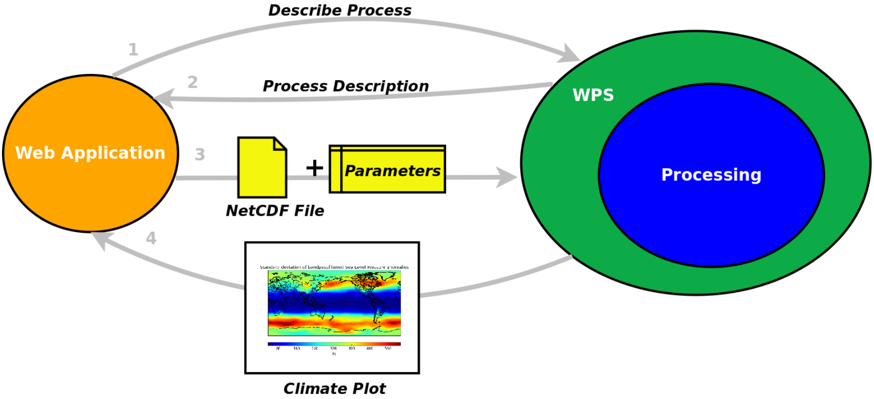
5 Security and Interoperability

Web Processing Service

A web service interface to standardize the way that algorithms are made available on the Internet



WPS Use Case


Describe Process
1
2
Process Description
WPS
Web Application
3
+
Parameters
Processing
NetCDF File
4
Climate Plot

What does WPS provide?

- web access to your algorithms (GET request with key-value, POST request with xml)
- WPS knows about the inputs and outpus of a process
- processes are self-describing (GetCapabilites, DescribeProcess)
- sync and async calls (async calls with status document)
- its a standard interface ... several implementations are available (PyWPS, GeoServer, COWS, ...)
- process definition is easy to write
- not restricted to a specific programming language
- can be used internally to provide enhanced functionality to web portals

Enable your code as WPS process

Use wps decorator for your function

```
@wps
def myplot(nc_file, variable):
    """
    nc_file application/netcdf
    variable string
    """
    # plot variable of nc_file
    return plot.png
```

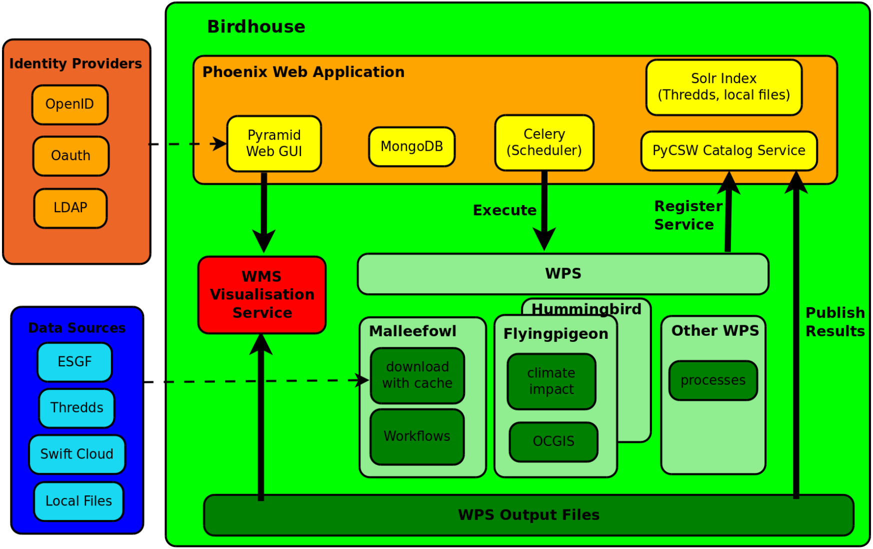
Execute your function with WPS

```
http://localhost/wps?service=WPS&version=1.0.0 \
   &request=execute \
   &identifier=myplot \
   &DataInputs=nc_file=http://;variable=tas
```

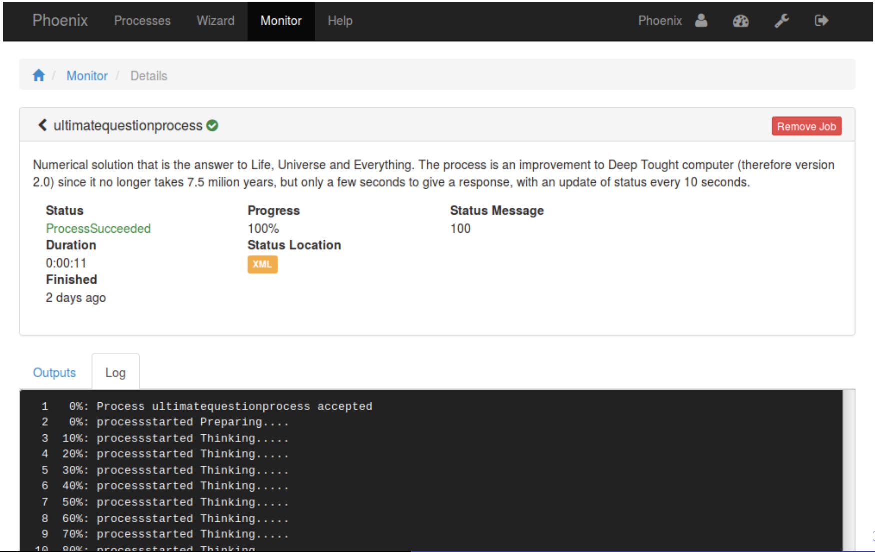
Overview

1. Motivation
2. Birdhouse Components
3. Birdhouse Builder
   - Conda
   - Buildout
4. Deployment with Docker
5. Security and Interoperability

Birdhouse Components



Phoenix web-based WPS client



ultimatequestionprocess ✓

Numerical solution that is the answer to Life, Universe and Everything. The process is an improvement to Deep Tought computer (therefore version 2.0) since it no longer takes 7.5 milion years, but only a few seconds to give a response, with an update of status every 10 seconds.

Status
ProcessSucceeded
Duration
0:00:11
Finished
2 days ago

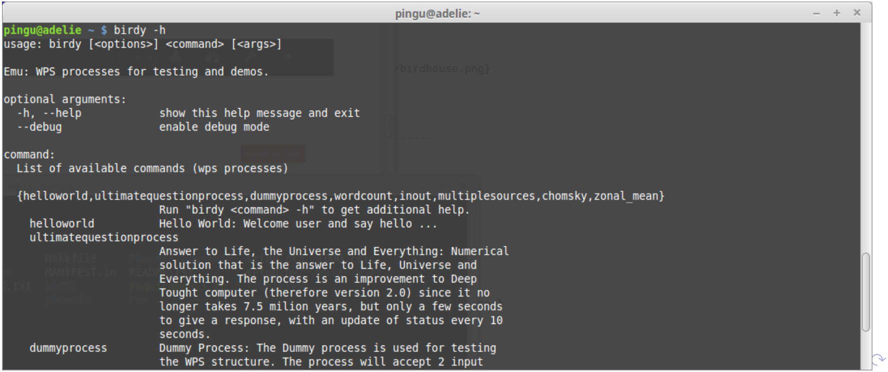
Progress
100%
Status Location
XML

Status Message
100

```
  0%: Process ultimatequestionprocess accepted
  0%: processstarted Preparing....
 10%: processstarted Thinking.....
 20%: processstarted Thinking.....
 30%: processstarted Thinking.....
 40%: processstarted Thinking.....
 50%: processstarted Thinking.....
 60%: processstarted Thinking.....
 70%: processstarted Thinking.....
 80%: processstarted Thinking
```

Birdy command line WPS client

```
>> conda install -c birdhouse birdhouse-birdy
>> export WPS_SERVICS=http://localhost:8094/wps
>> birdy -h
```

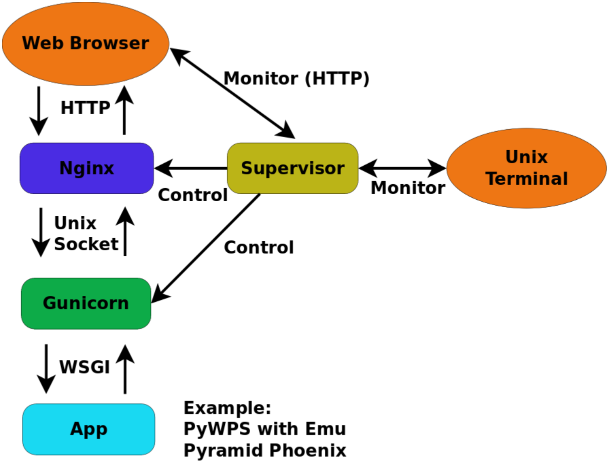


```
pingu@adelie ~ $ birdy -h
usage: birdy [<options>] <command> [<args>]

Emu: WPS processes for testing and demos.

optional arguments:
  -h, --help            show this help message and exit
  --debug               enable debug mode

command:
  List of available commands (wps processes)

  {helloworld,ultimatequestionprocess,dummyprocess,wordcount,inout,multiplesources,chomsky,zonal_mean}
                        Run "birdy <command> -h" to get additional help.
    helloworld          Hello World: Welcome user and say hello ...
    ultimatequestionprocess
                        Answer to Life, the Universe and Everything: Numerical
                        solution that is the answer to Life, Universe and
                        Everything. The process is an improvement to Deep
                        Tought computer (therefore version 2.0) since it no
                        longer takes 7.5 milion years, but only a few seconds
                        to give a response, with an update of status every 10
                        seconds.
    dummyprocess        Dummy Process: The Dummy process is used for testing
                        the WPS structure. The process will accept 2 input
```

Birdhouse WPS serivces

- Malleefowl (for internal use): download data with cache, workflows ...
- Emu: processes for testing and demo
- Hummingbird: processes for general tools used in the climate science community like cdo and cfchecker
- Flyingpigeon: processes for climate data, indices and extreme events

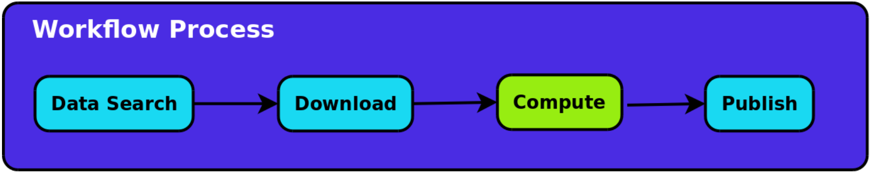
WSGI Application controlled with Supervisor



Example:
PyWPS with Emu
Pyramid Phoenix

Workflow Process in Birdhouse

- the chain of WPS processes is controlled by a Workflow Engine (dispel4py)
- The Workflow Process itself is a WPS process
- data-search, download and publish are internal processes (provided by Malleefowl)
- Compute is a process choosen by the user ... for example cfchecker
- The Phoenix wizard is used to collect the parameters for the workflow process

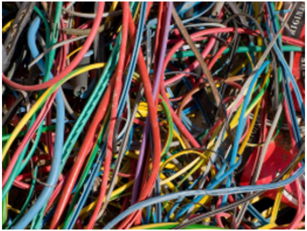


Overview

1. Motivation
2. Birdhouse Components
3. Birdhouse Builder
   - Conda
   - Buildout
4. Deployment with Docker
5. Security and Interoperability

Why Conda and Buildout?

- many components: WPS, WMS, web-server, solr, ...
- lots of dependencies: cdo, cfchecker, ocgis, numpy, R, ...
- many different kinds of config files need to be configured
- installation needs to be reproducible at different locations
- should work with different Linux distributions (Centos, Fedora, Debian, Ubuntu, ...)



conda package manager

- originally for python ... but has a general concept
- does not need admin rights
- manages dependencies

install from birdhouse channel

```
>> conda install -c birdhouse pywps cdo
```

create conda environment=emu

```
>> conda create -n emu -c birdhouse \
        python=2.7 cdo pywps
```

conda recipe

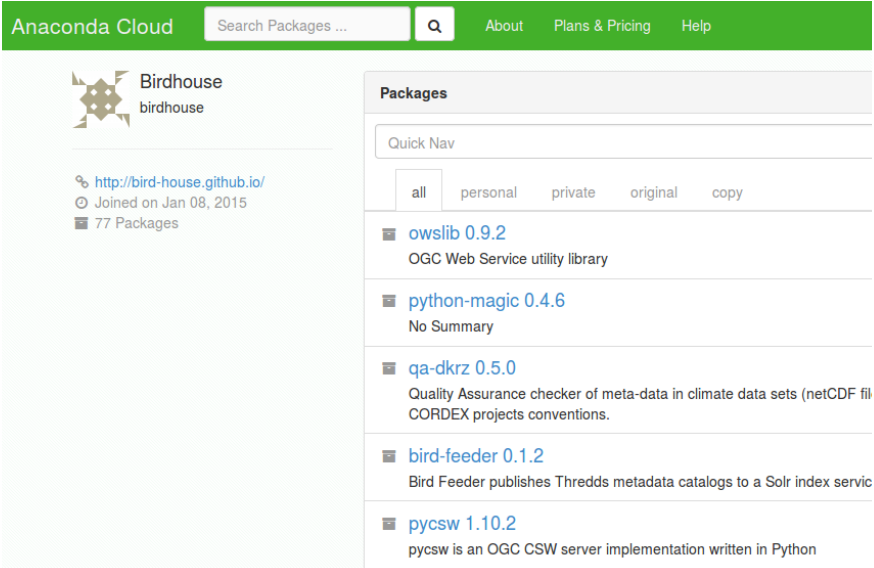
meta.yaml

```
package:
  name: owslib
  version: !!str 0.9.2
source:
  url:
requirements:
  run:
    python
    lxml
```

build conda package

```
>> ls
meta.yaml build.sh
>> conda build .
```

Anaconda Cloud



Birdhouse

birdhouse

http://bird-house.github.io/

Joined on Jan 08, 2015

77 Packages

Packages

owslib 0.9.2

OGC Web Service utility library

python-magic 0.4.6

No Summary

qa-dkrz 0.5.0

Quality Assurance checker of meta-data in climate data sets (netCDF fil CORDEX projects conventions.

bird-feeder 0.1.2

Bird Feeder publishes Thredds metadata catalogs to a Solr index servic

pycsw 1.10.2

pycsw is an OGC CSW server implementation written in Python

Buildout

- Python based build system
- creates application with multiple components including configuration files
- works also for non-Python parts
- using a buildout configuration
- can be extended with recipes

Buildout configuration

buildout.cfg

```
[buildout]
parts = conda pywps

[settings]
hostname = localhost

[conda]
recipe = birdhousebuilder.recipe.conda
pkgs = ipython cdo

[pywps]
recipe = birdhousebuilder.recipe.pywps
title = Emu WPS
hostname = ${settings:hostname}
```

Buildout Recipe

birdhousebuilder.recipe.pywps

```python
from birdhousebuilder.recipe import conda, supervisor, nginx

class PyWpsRecipe(object):
    def __init__(self, buildout, name, options):
        # set default options

    def install(self):
        # install pywps with nginx, gunicorn and supervisor
        # generate config files from templates according the options

    def update(self):
        # update configuration
```

Install Birdhouse Component with Buildout

- for convienience there is a Makefile to call the buildout commands
- all Birdhouse components (WPS, Phoenix) are installed in the same way

First installation

```
>> git clone https://github.com/bird-house/emu.git
>> cd emu
>> make clean install
>> make start
```

Update configuration like hostname, port

```
>> vim custom.cfg
>> make update
>> make restart
```

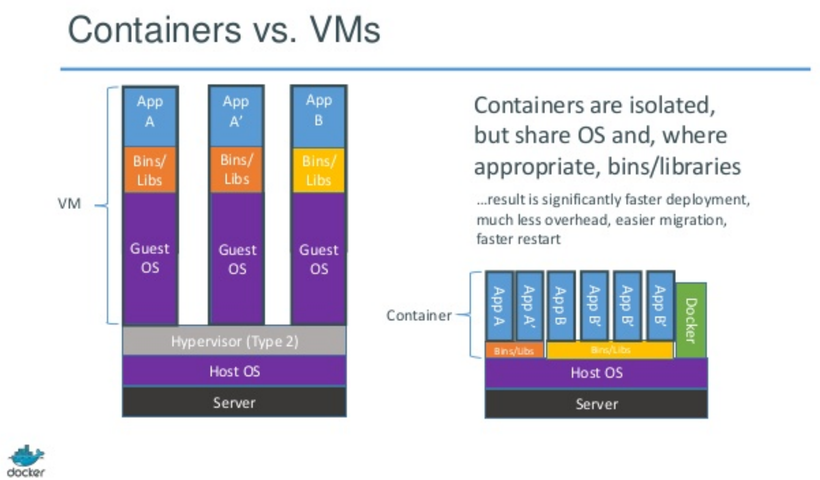
Overview

1. Motivation
2. Birdhouse Components
3. Birdhouse Builder
   - Conda
   - Buildout
4. Deployment with Docker
5. Security and Interoperability

What is Docker?

- a lightweight Virtual-Machine using Linux Containers
- isolated environment for your Linux installation
- runs on the same hardware as the host
- run latest Ubuntu on an older Centos
- rapid startup time
- only changed parts of docker image need to loaded on update

Containers vs VMs (slide taken from docker)



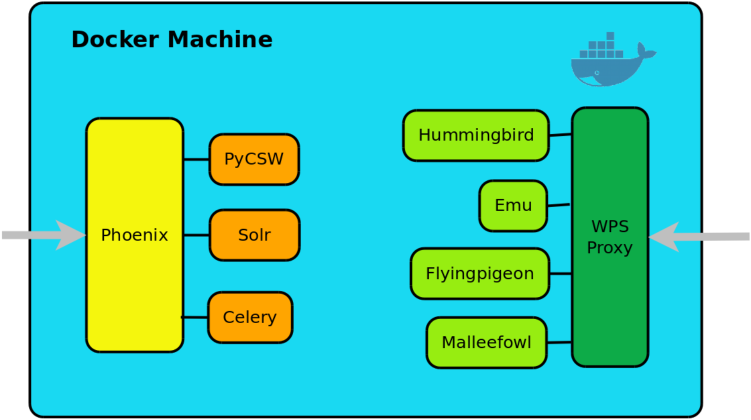
Containers vs. VMs

Containers are isolated, but share OS and, where appropriate, bins/libraries

...result is significantly faster deployment, much less overhead, easier migration, faster restart

Deploy Birdhouse with Docker

- each service is running in a Docker container
- docker containers can be *linked* to other containers
- only some containers (Phonix, WPS Proxy) need to be exposed to external use



Dockerfile

```
FROM ubuntu:14.04

# Add application sources
ADD . /opt/birdhouse

# cd into application
WORKDIR /opt/birdhouse

# Install system dependencies
RUN bash bootstrap.sh -i && bash requirements.sh

# Run install
RUN make clean install

# Volume for data, cache, logfiles, ...
VOLUME /data

# Ports used in birdhouse
EXPOSE 8090 8094

# Update config and start supervisor ...
CMD ["make", "start"]
```

Try a Docker ...

Images are available on DockerHub

```
https://hub.docker.com/u/birdhouse/
```

Start a docker image with Emu WPS

```
>> docker run -it -p 8090:8090 -p 8094:8094 \
         --name=emu_wps birdhouse/emu
```

Run WPS GetCapabilities Request

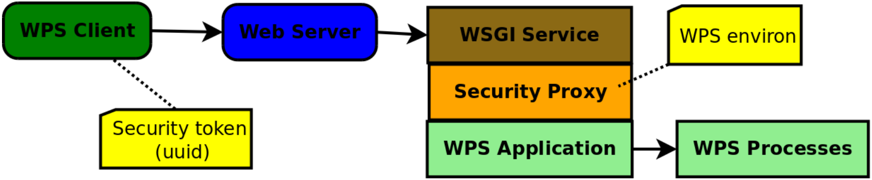
```
http://localhost:8094/wps? \
   service=WPS&version=1.0.0&request=getcapabilities
```

Overview

1. Motivation
2. Birdhouse Components
3. Birdhouse Builder
   - Conda
   - Buildout
4. Deployment with Docker
5. Security and Interoperability

WPS Security Proxy (planned)

- using string token (uuid) as part of URL to protect WPS execute access
- X509 certificates to access (remote) data from ESGF are provided by proxy (using environ)
- implemented as WSGI application layer



Best Practises to make WPS interchangeable

- support of complete WPS protocol (literal type, complex types, ...)
- separation of WPS definition and functional code (provided as Python library)
- convenience and integration code provided as library (e.a. Malleefowl provides functions used by WPS processes)
- using WPS profiles (common WPS process definitions)
- use self-describing possibilities of WPS
- parameters relevant for the process should be part of the process definition

Overview

Further Reading I

- Birdhouse
  http://bird-house.github.io/
- Buildout
  http://www.buildout.org/
- Anaconda
  https://www.continuum.io/why-anaconda
- Evaluation of WPS Frameworks
  http://www.slideshare.net/mepa1363/foss4g-ebrahim
- Web Processing Service
  http://www.slideshare.net/GasperiJerome/20130530-web-processing-service-cct-cloud-toulouse-29423710

The End