Birdhouse Bootstrap Documentation

Release 0.1

Birdhouse

Aug 16, 2018

Contents

Birdhouse uses Buildout to setup Web Processing Service applications like Emu and Flyingpigeon with all configurations.

All Birdhouse WPS applications have a common way to bootstrap the buildout installation. Part of this bootstrap process is to install system packages required by the application and to initialize the buildout installation (`bootstrap.py`, install Anaconda, . . .). In addition there is a Makefile to simplify some tasks like cleaning the sources and running buildout.

The installation is using the Python distribution system Anaconda to maintain software dependencies. You may use an existing (shared, read-only access possible) Anaconda installation. For this set an environment variable to the location of your existing Anaconda, for example:

```
$ export ANACONDA_HOME=/opt/anaconda
```

If Anaconda is not available then a minimal Anaconda will be installed during the installation processes in your home directory `~/anaconda`.

The installation process setups a conda environment named `birdhouse`. All additional packages and configuration files are going into this conda environment. The location is `~/.conda/envs/birdhouse`.

There are two main files in this project: `bootstrap.sh` and `Makefile`.

bootstrap.sh A copy of this script needs to be in each Birdhouse application. It will fetch the current `Makefile` from the bootstrap github repo and install the essential system packages (`wget`, `make`, . . .) to start an installation.

Makefile This `Makefile` has targets to bootstrap and run buildout to install the application.

CHAPTER 1

Getting Started

1.1 Usage

1.1.1 bootstrap.sh

Run `bootstrap.sh -h` in your Birdhouse application root folder to see the available options:

```
$ cd MyApp
$ bash bootstrap.sh -h
```

This will output:

```
Usage : bootstrap.sh [option]

Options:
     -h   - Print this help message.
     -i   - Installs required system packages for Birdhouse build. You *need* 'sudo'␣
↪priviliges!"
     -u   - Updates Makefile for Birdhouse build. Python needs to be installed."
     -b   - Both system packages will be installed (-i) and Makefile will be updated␣
↪(-u). Default."
```

1.1.2 Makefile

Run `make help` in your Birdhouse application root folder to see the available options:

```
$ cd MyApp
$ make help
```

This will output:

```
make [target]

targets:

     all            - Does a complete installation. Shortcut for 'make sysinstall␣
↪clean install.' (Default)
     help           - Prints this help message.
     version        - Prints version number of this Makefile.
     info           - Prints information about your system.
     install        - Installs your application by running 'bin/buildout -c custom.cfg
↪'.
     test           - Run tests (but skip long running tests).
     testall        - Run all tests (including long running tests).
     clean          - Deletes all files that are created by running buildout.
     distclean      - Removes *all* files that are not controlled by 'git'.
                      WARNING: use it *only* if you know what you do!
     sysinstall     - Installs system packages from requirements.sh. You can also␣
↪call 'bash requirements.sh' directly.
     selfupdate     - Updates this Makefile.

Supervisor targets:

     start          - Starts supervisor service: /home/pingu/.conda/envs/birdhouse/
↪etc/init.d/supervisord start
     stop           - Stops supervisor service: /home/pingu/.conda/envs/birdhouse/etc/
↪init.d/supervisord stop
     restart        - Restarts supervisor service: /home/pingu/.conda/envs/birdhouse/
↪etc/init.d/supervisord restart
     status         - Supervisor status: /home/pingu/.conda/envs/birdhouse/bin/
↪supervisorctl status

Docker targets:

     Dockerfile     - Generates a Dockerfile for this application.
     dockerbuild    - Build a docker image for this application.
```

1.2 Examples

1.2.1 General Questions

Just build my app

For convenience applications come already with a Makefile. So, the simplest way to build the application is:

```
$ git clone https://github.com/bird-house/MyApp.git
$ cd MyApp
$ make
```

Start the application with:

```
$ make start    # start supervisor
$ make status   # check status
```

Check the log files for errors:

```
$ cd ~/.conda/envs/birdhouse
$ ls var/log/
```

Generated config files are in etc/:

```
$ cd ~/.conda/envs/birdhouse
$ ls etc/
```

Just rebuild my app

Your application is checked out and system requirements are already installed:

```
$ cd MyApp
$ make install
```

To get the latest eggs from pypi you should run:

```
$ make clean install
```

If you have changed system requirements in your `requirements.sh` file:

```
$ cd MyApp
$ vim requirements.sh  # add system packages for your application
$ make sysinstall      # install requirements
$ make clean install   # run a clean build
```

Restart your application:

```
$ make restart
$ make status
```

Keep my data at a save place

By default all configuration and data files are stored below the installation root folder in the conda environment `birdhouse`:

```
$ cd ~/.conda/envs/birdhouse
$ ls
bin  conda-meta  etc  Examples  html  include  lib  libexec  man  opt  plugins  sbin ␣
↪share  ssl  var
```

Configuration files are in the `etc/` folder. Data (databases, caches, . . .) and log files are in the `var/` folder.

If you want to keep your data files in a location of your choice (the birdhouse location might change in the future . . .) then move the `var/` folder to that place and replace it with a softlink:

```
$ cd ~/.conda/envs/birdhouse
$ mv var /home/myself/Shared/var
$ ln -s /home/myself/Shared/var .
```

If you use a new `var/` folder (contains no birdhouse files) then you need to run the installation again:

```
$ cd ~/.conda/envs/birdhouse
$ mv var var.old
$ ln -s /home/myself/Shared/var .
# run installation ...
$ cd ~/sandbox/bridhouse/myapp
$ make clean install
```

attention:: Make sure you have write access to the `var/` folder.

Install my app with an unprivileged user

Your installation user has no `sudo` rights:

```
nobody$ git clone https://github.com/bird-house/MyApp.git
nobody$ cd MyApp
```

Run `make sysinstall` with a user who has sudo rights to install system requirements:

```
admin$ make sysinstall
```

The application build itself does not need sudo rights:

```
nobody$ make clean install
nobody$ make start
nobody$ make status
```

Update to the latest Makefile . . .

Just do:

```
$ make selfupdate
```

There is no "make" on my system

Just do:

```
$ bash boostrap.sh # will install make and wget
$ make
```

1.2.2 Anaconda

Use a shared Anaconda installation

You can use an existing Anaconda installation which might be read-only and shared with others. For this set an environmet variable to point to this shared Anaconda location:

```
$ export ANACONDA_HOME=/opt/anaconda
```

The run your installation again:

```
$ make clean install
```

Choose a non default conda enviroment root directory

By default installation goes into the birdhouse environment which is in conda enviroments directory: `~.conda/envs/`. You can overwrite the conda envs dir by settings the environment variable `CONDA_ENVS_DIR`:

```
$ export CONDA_ENVS_DIR=/opt/conda_envs
```

The run your installation again:

```
$ make clean install
```

Use my birdhouse conda environment

To activate the birdhouse environment do the following:

```
$ source activate birdhouse
```

Read the conda docs for further information:

http://conda.pydata.org/docs/faq.html#env-creating

1.2.3 Docker

Generate a docker image for my app

Just do:

```
$ make dockerbuild
```

Just generate a Dockerfile . . .

Just do:

```
$ make Dockerfile
```

You can change the default docker base image in your `custom.cfg`:

```
$ vim custom.cfg
[docker]
image-name = centos
image-version = centos6
maintainer = MyApp
```

CHAPTER 2

Indices and tables

- genindex
- modindex
- search