# H1DS Documentation

***Release 0.9a***

**David Pretty**

April 21, 2015

Contents

H1DS is an extensible web interface designed for data from magnetically confined plasma experiments.

Contents:

CHAPTER 1

# Installing H1DS

H1DS is designed to run within virtualenv (a python virtual environment). The installation process should therefore be quite similar between different operating systems. To date, H1DS has only been used with Linux, but with modest changes to the code it should run on any other operating system supported by Python (Windows, Linux/Unix, Mac OS X).

H1DS currently supports python 2.6.5+ and 2.7.x

Currently MDSplus is the only supported data system. While H1DS is designed to be modular such that it will work with data systems other than MDSplus, there are no immediate plans to support other systems. If you are interested in using H1DS with a non-MDSplus system, I'd be happy to help you with the required code changes.

## 1.1 Prerequisites

In theory, these prerequisites should be the only part of the installation process which depends on the operating system. However, H1DS has only been tested on Linux (Ubuntu and Arch Linux), so you may encounter some issues on other platforms.

### 1.1.1 Ubuntu 12.04 LTS

First, install git, virtualenv and the python header files (needed for compiling some python libraries). Currently we also need openssh server, as the script which deploys the production server over ssh is also used to set up the development server on the local computer (H1DS issue #10). We also install `gfortran` and `libatlas-base-dev` so we can build `numpy` in our virtualenv, and `libfreetype6-dev` and `libpng12-dev` so we can build `matplotlib`. `mercurial` is needed to fetch the source for django-python-code-field:

```
$ sudo apt-get install git python-virtualenv python-dev ssh gfortran build-essential libatlas-base-de
```

We also use virtualenvwrapper to manage the python virtual environments. Unfortunately, the recent versions of Ubuntu have a problem with the Ubuntu packaged version of virtualenv which causes problems for H1DS. Instead, we install virtualenvwrapper via the pip installer (which should have been installed as a dependency of python-virtualenv):

```
$ sudo pip install virtualenvwrapper
```

**Note:** The Ubuntu virtualenv package only causes a problem when it is used with sudo, which is required when deploying a production server (e.g. when we need to reload the webserver). If you are looking to run H1DS just on your own computer you might be able to use the Ubuntu package. However, to keep things simple here we'll assume virtualenv has been installed via pip.

Now add the following to both your `~/.bashrc` and `~/.profile` files, replacing `my_username` with your own username (you can use whatever you like for `$WORKON_HOME`, it will be where all your virtualenvs are stored):

```
if [ $USER == my_username ]; then
    export WORKON_HOME=$HOME/v
    source /usr/local/bin/virtualenvwrapper.sh
fi
```

Then, either start a new terminal or read in your `.bashrc` file:

```
$ source ~/.bashrc
```

The `~/.profile` is read when you run the Fabric script (which uses a login shell, and therefore checks `~/.profile` (or `~/.bash_profile`) rather than `~/.bashrc`, which is read for non-login shells.

If you don't already have MDSplus installed, follow the installation instructions for Ubuntu which you can find here: http://www.mdsplus.org/index.php/Latest_Ubuntu_Packages

## 1.2 Setting up the development environment

We'll first create our python virtual environment. The `mkvirtualenv` command will activate the new virtualenv for you, and prefixes the virtualenv name to the shell prompt. The `cdvirtualenv` command takes you to the virtualenv directory (here it's `$WORKON_HOME/h1ds_development`).

```
$ mkvirtualenv h1ds_development
(h1ds_development)$ cdvirtualenv
```

---

**Note:** If you have an old version of virtualenv (before version 1.7) you may need to include the `--no-site-packages` flag

---

We'll be using Fabric to automate much of the installation process, so let's install it into our virtualenv now:

```
(h1ds_development)$ pip install fabric
```

Now grab the H1DS project from the git repository:

```
(h1ds_development)$ git clone https://github.com/h1ds/h1ds.git
(h1ds_development)$ cd h1ds
```

In the H1DS project we need to create a couple of initial configuration files from the provided templates; the H1DS fabric script (they call it a *fabfile*) and the Django project settings file:

```
(h1ds_development)$ cp fabfile.py{.template,}
(h1ds_development)$ cp h1ds/h1ds/settings/development.py{.template,}
```

Open up `development.py` in an editor and change the `SECRET_KEY` to something unique and unguessable. For other options in the configuration file, see *settings_(development|staging|production).py*.

It is also recommended that you use use a database server such as Postgres or MySQL, rather than the default SQLite. For instructions on how to configure for various databases, see **'https://docs.djangoproject.com/en/1.5/ref/settings/#databases'_**.

Currently H1DS requires MDS, so you'll need to install the MDS python bindings into your virtualenv.

```
(h1ds_development)$ mkdir $VIRTUAL_ENV/src
(h1ds_development)$ cp -rp /usr/local/mdsplus/mdsobjects/python $VIRTUAL_ENV/src/python-mdsplus
(h1ds_development)$ cd $VIRTUAL_ENV/src/python-mdsplus
(h1ds_development)$ python ./setup.py install
```

Then, install the rest of the required software using the fabric script:

```
(h1ds_development)$ fab dev update
```

---

**Note:** If the above doesn't work, make sure you added the virtualenvwrapper code in your `~/.profile` or `~/.bash_profile` file and you have `ssh` installed):

---

During the update you'll be asked if you want to create a Django superuser account. Answer `yes` and provide the requested details (name, email etc).

You can now start the development server via:

```
(h1ds_development)$ export PYTHONPATH=$VIRTUAL_ENV/h1ds/h1ds:$PYTHONPATH
(h1ds_development)$ export DJANGO_SETTINGS_MODULE=h1ds.settings.development
(h1ds_development)$ django-admin.py runserver
```

You can update H1DS any time by repeating the `fab dev update` command.

## 1.3 Setting up a staging environment

If you are making changes to the H1DS code for a production server, it helps to have the production environment replicated in a staging server so you can make sure your code changes behave as expected before changing the code on your public website.

Here we use VirtualBox to replicate the production server, run on the development system (i.e. laptop) with a host-only network connection between the development system and staging server. We will use Ubuntu 12.04 LTS for the staging server.

First, you'll need to install VirtualBox and start a new Ubuntu 12.04 guest operating system. There are plenty of resources on the web to help you with that, so I won't go into any detail here on how to do it. Once you have your Ubuntu virtual server working, follow the prerequisite steps above (see *Prerequisites*).

You can set up H1DS to use either the Nginx or Apache web servers. Nginx is recommended, as it better handles long connections and server side events, which allow us to have server-side events and data streaming. For an Nginx setup, edit your *fabfile.py* settings to include:

```
STAGING_WEBSERVER = "nginx"
```

For Apache, use:

```
STAGING_WEBSERVER = "apache"
```

**Using Nginx (recommended)**

You'll need to install the nginx webserver:

```
$ sudo apt-get install nginx
```

As Nginx acts as a reverse proxy, we need a server running the actual django code. We'll use gunicorn and gevent, which the fabfile will install into the virtualenv. However, to build gevent we need to install another system library:

```
$ sudo apt-get install libevent-dev
```

The default Nginx install includes a default site configuration which we need to remove. We remove the symbolic line from */etc/nginx/sites-enabled*, the original file can be found at */etc/nginx/sites-available/default* if you need to refer to it at a later time:

```
$ sudo rm /etc/nginx/site-enabled/default
```

**Using Apache (deprecated & no longer tested)**

You'll need to install the apache webserver and wsgi module:

```
$ sudo apt-get install apache2 libapache2-mod-wsgi
```

Also deactivate the default apache site on your staging server:

```
$ sudo a2dissite 000-default
$ sudo service apache2 reload
```

Next, set up a host-only network connection for your staging server. You may need to load the `vboxnetadp` and `vboxnetflt` kernel modules on your host (development) system. Then, in the general VirtualBox settings (`File -> Preferences...`) go to the network settings and create a new host-only network. Then in the VirtualBox settings for your staging server select `Network` and add a new adapter attached to host-only adapter and select the newly created host-only network as its name (you may need to power off the virtual machine to edit the settings).

With your staging server powered up, type `ip addr` to find the IP address of your staging server on the host-only network, it should be something like `192.168.56.101`, and will likely be `eth1`. Edit the staging server settings in *fabfile.py* in your development environment:

```
STAGING_USER = "username" # user on VirtualBox guest system
STAGING_HOST = "192.168.56.101" # Host-only IP address of VirtualBox guest system
```

Next, in your development virtualenv, run:

```
(h1ds_development)$ fab staging setup
(h1ds_development)$ cd h1ds/h1ds/settings
(h1ds_development)$ cp staging.py{.template,}
```

Make any desired changes to `staging.py` – you should at least edit `SECRET_KEY` to something unique and unguessable. Then update the staging server:

```
(h1ds_development)$ fab staging update
```

You should be able to see H1DS running in a browser at the host-only IP address of the staging server (i.e. `http://192.168.56.101`).

If everything appears to be working, you probably won't need the graphical interface to your staging server as you really only need ssh and tcp (for http) access. It may be more convenient to run VirtualBox in headless mode. For example, if your staging server is called `Ubuntu 12.04 LTS` this would be:

```
$ VBoxHeadless --startvm "Ubuntu 12.04 LTS"
```

When you want to close the virtual machine, without powering it off, type:

```
$ vboxmanage controlvm "Ubuntu 12.04 LTS" savestate
```

## 1.4 Setting up a production environment

The setup procedure for the production environment is essentially the same as for the staging environment. You'll just need to install the prerequisites and Apache, edit the `PRODUCTION_USER` and `PRODUCTION_HOST` in your `fabfile.py`, create `production.py` from the template and run `fab production setup` and `fab production update` from within your development environment.

CHAPTER 2

# Overview of H1DS

CHAPTER 3

# H1DS Concepts

This page gives a brief overview of some of the central concepts behind H1DS.

## 3.1 Data access as a web service

## 3.2 A RESTful approach

## 3.3 Based on Django

## 3.4 Modular

CHAPTER 4

# H1DS configuration

## 4.1 settings_(development|staging|production).py

The `settings_development.py`, `settings_staging.py` and `settings_production.py` files inherit from the standard django `settings.py`. You should familiarise yourself with the available Django settings at https://docs.djangoproject.com/en/1.5/ref/settings/.

Listed below are module-specific configuration options for modules used by H1DS. Defaults are those values specified in `settings.py`, and are overridden by values in `settings_(development|staging|production).py`.

### 4.1.1 Settings for h1ds

#### H1DS_EXTRA_SUBLINKS

Syntax: (name, url, description)

Default: `(("Wiki", "/wiki", "Documentation wiki"), ("Activity", "/wiki/RecentChanges", "Latest changes to documentation"),)`

Extra links to be displayed in the header and on the frontpage.

#### WIKI_ACL_RIGHTS_BEFORE

Default: `u""`

Example: `WIKI_ACL_RIGHTS_BEFORE = u"BoydBlackwell:read,write,delete,revert,admin"`

#### WIKI_ACL_RIGHTS_DEFAULT

Default: `u""`

Example: `WIKI_ACL_RIGHTS_DEFAULT = u"EditorGroup:read,write,delete +All:read -All:write,delete,revert,admin"`

### 4.1.2 Settings for h1ds_mdsplus

#### H1DS_MDSPLUS_ROOT_URL

Default: `"mdsplus"`

Root URL for module.

#### H1DS_MDSPLUS_NODE_BLACKLIST

Default: `[]`

A list of any MDSPlus nodes to ignore (e.g. if they crash the server).

#### EXTRA_MDS_TREES

Default: `[('test', os.path.join(VENV_DIR, 'test_mds_data'))]`

list of extra mds trees to load into environment each entry should be a (name, path), for example `EXTRA_MDS_TREES = [('extratree1', 'mdsserver::'), ('anothertree', '/data/tree'),]`

#### DEFAULT_TREE

Default: `"test"`

#### SHOT_TRACKER

Default: `"ping"`

Method for tracking shot changes.

Options:

- "ping" - periodically ask MDSplus for the latest shot.
- "inotify" [not implemented] - (linux only) listen for changes to shotid.sys.

### 4.1.3 Settings for h1ds_summary

#### H1DS_SUMMARY_ROOT_URL

Default: `"summary"`

Root URL for module.

### 4.1.4 Settings for h1ds_configdb

#### H1DS_CONFIGDB_ROOT_URL

Default: `"configurations"`

Root URL for module.

### 4.1.5 Settings for django_openid_auth

#### OPENID_CREATE_USERS

Default: `True`

#### LOGIN_URL

Default: `'/openid/login'`

#### LOGIN_REDIRECT_URL

Default: `'/'`

### 4.1.6 Settings for djcelery

#### BROKER_URL

Default: `"django://"`

CHAPTER 5

# Indices and tables

- *genindex*
- *modindex*
- *search*