Melissi Server Documentation

Release 0.1

Giorgos Logiotatidis

Sep 27, 2017

Contents

1 Installing **1**
1.1 To host your own melissi server (aka *Hive*) 1
1.2 Things To Know 1
1.3 Preparing You System 1
1.4 Getting Melissi Source 3
1.5 Installing Melissi 3
1.6 Configuring Your Hive 4
1.7 Running Your Hive 5
1.8 Updating Your Hive 9

2 Hive Administration **11**
2.1 Things To Know 11
2.2 Adding and Removing Users 11

CHAPTER 1

Installing

To host your own melissi server (aka *Hive*)

Melissi is a cloud storage server written in Django. Each server is called a *Hive*.

Ultimattely the *hives* will be able to communicate with each other, forming a *federated* network of cloud storage servers. You will have control of your data and your infrastracture and still be able to share files with others, even if they live in different hives.

If you want can host your own hive and be the master of you data and your infrastructure, then this document is for you.

If you just want a place to store your data then you can use a hive controlled by someone else. You can join a hive controlled by your friends, your university or your company.

Things To Know

1. Melissi server is currently *alpha* software. Things may and will change drastically, as the project evolves and matures.

2. You shouldn't trust melissi to backup your data. Melissi is a storage platform not a backup platform.

3. If things go bad, or you get stuck feel free to ping us at @melissiproject, or #melissiproject on freenode.

4. Please fill bug reports on github, the more the bug reports and better the software will be.

Preparing You System

To install your own hive:

1. **Make sure you have a recent Python installed.**

Most distributions come with python preinstalled, so probably you need to do nothing. Check if you have python using the following command

```
~$ python --version
```

2. **Install virtualenv and pip**

   Virtualenv creates a virtual python environment, in which we will install melissi and all its dependencies (e.g. Django). This way your melissi installation won't conflict with any other python projects / installs you may have on your server. PIP is a package manager for python.

   - For *Debian* systems:

     ```
     ~$ sudo apt-get install python-pip python-virtualenv
     ```

   - For *Fedora* system:

     ```
     ~$ su -c "yum install python-pip python-virtualenv"
     ```

3. **Install git**

   We use git to version our code. To get the latest melissi source code you need to install git.

   - For *Debian* systems:

     ```
     ~$ sudo apt-get install git
     ```

   - For *Fedora* systems:

     ```
     ~$ su -c "yum install git"
     ```

4. **Install librsync**

   To reduce bandwidth needs librsync is used for calculating patches. You need to install librsync development libraries to build python-librsync module.

   - For *Debian* systems:

     ```
     ~$ sudo apt-get install librsync-dev
     ```

   - For *Fedora* systems:

     ```
     ~$ su -c "yum install librsync-devel"
     ```

5. **Install Extra Packages** (Optional)

   Because we are building melissi inside a virtual enviroment you will need to download some extra packages to be able to compile some python modules, such as the MySQL-python

   - **Mysql on Debian Systems**

     Install python development files, needed to build python mysql connector:

     ```
     ~$ sudo apt-get install python-dev
     ```

     Install MySQL:

     ```
     ~$ sudo apt-get install mysql-server libmysqlclient-dev
     ```

- **PostgreSQL on Debian Systems**

  Install python and postgresql development files, needed to build python postgresql connector:

  ```
  ~$ sudo apt-get install python-dev
  ```

  Install PostgreSQL:

  ```
  ~$ sudo apt-get install postgresql libpq-dev
  ```

- **Mysql on Fedora**

  Install python development files, needed to build python mysql connector:

  ```
  ~$ su -c "yum install python-devel"
  ```

  Install MySQL:

  ```
  ~$ su -c "yum install mysql-server mysql-devel"
  ```

- **PosteSQL on Fedora**

  Install python and postgresql development files, needed to build python postgresql connector:

  ```
  ~$ su -c "yum install python-devel"
  ```

  Install PostgreSQL:

  ```
  ~$ su -c "yum install postgresql-server postgresql-devel"
  ```

Getting Melissi Source

You can get the source from our git repository.

1. Move to the directory you want to install melissi:

   ```
   ~$ mkdir /srv/melissi
   ~$ cd /srv/melissi
   ```

2. Fetch the source code:

   ```
   ~$ git clone git://github.com/melissiproject/server.git
   ```

Installing Melissi

Move to the directory you cloned melissi server and run the melissi-installer. Melissi installer will download from pypi all the needed python packages to run melissi.

```
~$ cd /srv/melissi/server
~$ ./scripts/melissi-installer.py --install
```

Note: It is recomended that you use melissi with a good database backend like MySQL or PostgreSQL. Do install the needed support you can should use the –mysql and / or –postgresql flags among the –install flag.

```
~$ ./scripts/melissi-install.py --install --mysql
```

If no flags are used then your hive will be able to run only using *sqlite*.

Warning: To install the mysql or postesql backends you need to execute the steps in section *extra-packages*

Configuring Your Hive

Before running your hive you need to configure at least the database settings and the storage path. All configuration options are located in file local_settings.py.

1. **Copy settings template**

```
~$ cp local_settings.py.example local_settings.py
```

2. **Edit using you favorite editor local_settings.py**

   - **Set DATABASES**

     This is the database to be used for melisi. You can refer to Django's documentation on Databases if you need more help.

     Note: When using MySQL or PostgreSQL you need to create a database first. You also need to grant permissions to your database user to access the database

     Use *mysqladmin* command to create MySQL databases

     ```
     ~$ mysqladmin -u root -p create melissi
     ```

   - **Set SECRET_KEY**

     A random secret key used as a seed in secret-key hashing algorithms. For more see Django's documentation on SECRET_KEY

   - **Set MELISSI_STORE_LOCATION**

     Point to a directory to store uploaded data to.

     Note: Since this directory is going to store the data from all user of your hive make sure that you save enough storage for everything.

   - **Set MELISSI_REGISTRATIONS_OPEN** (Default: False)

     Set either to *True* or *False* if you want or not other to be able to create accounts on your hive.

3. **Setup the database**:

```
~$ source env/bin/activate
(env)~$ python manage.py syncdb
(env)~$ python manage.py migrate mlscommon
```

Warning: When executing *syncdb* answer **no** to the question whether to create a superuser or not, or the setup will fail.

4. **Setup a superuser**

```
(env)~$ python manage.py createsuperuser
```

Running Your Hive

Test Setup: Using internal webserver

You can run your hive in *test* mode using django's internal webserver.

```
(env)~$ python manage.py runserver
```

Note: Your hive listens by default on *localhost:8000*. To listen to another port or interface you can execute *runserver* command with extra parameters

```
(env)~$ python manage.py runserver 0.0.0.0:8000
```

bind to *all* available interfaces on port 8000

Warning: The communication between your hive and clients will not be encrypted.

Now you can visit your administration interface at http://localhost:8000/admin/ and login using your superuser account.

Real Setup: Nginx and Gunicorn

1. **Install and Setup Gunicorn**

   Copying from Gunicorn's website:

   > 'Green Unicorn' is a Python WSGI HTTP Server for UNIX. It's a pre-fork worker model ported from Ruby's Unicorn project. The Gunicorn server is broadly compatible with various web frameworks, simply implemented, light on server resources, and fairly speedy.

   (a) **Install**:

   ```
   ~$ cd /path/you/installed/melissi
   ~$ ./scripts/melissi-installer.py --install --gunicorn
   ```

   (b) **Setup**

   Edit */path/you/installed/melissi/scripts/gunicorn.sh* to fit your needs and paths.

2. **Install and Setup Nginx**

   (a) **Install**

- For *Debian* systems:

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

- For *Fedora* system:

```
~$ su -c "yum install nginx"
```

(b) **Setup with SSL** (recommended)

i. Generate an SSL certificate:

Note: If you have a certificate already you can skip this step. Make sure that you set correctly the paths in the next step so nginx can locate your certificate

```
~$ sudo -s
~# cd /etc/nginx
~# sudo mkdir /etc/nginx/ssl
~# cd ssl

# create private key
~# openssl genrsa -des3 -out melissi.key 1024

# create a CSR (Certificate Signing Request)
~# openssl req -new -key melissi.key -out melissi.csr

# optionally remove th passphrase from you key
# so nginx can start without a password
~# cp melissi.key melissi.key.bah
~# openssl rsa -in melissi.key.bak -out melissi.key

# create a CRT
~# openssl x509 -req -days 365 -in melissi.csr -signkey melissi.key -
↪out melissi.crt
```

Now your key and self-signed certificated are located in:

```
/etc/nginx/ssl/melissi.key
/etc/nginx/ssl/melissi.crt
```

i. Create */etc/nginx/sites-availables/melissi* with the following contents:

```
upstream app_server_melissi {
        # should match your paths
        server unix:///srv/melissi/sockets/melissi.sock fail_timeout=0;
}

server {
   server_name example.com;
   listen *:8000 default ssl;

   ssl_certificate ssl/melissi.crt;
   ssl_certificate_key ssl/melissi.key;

   # should match your paths
   access_log /srv/melissi/logs/nginx.access.log;
   error_log /srv/melissi/logs/nginx.error.log;
```

```
    # set this to the maximum file size allowed in your hive
    client_max_body_size 100m;

    location  /static/admin/ {
          # should match your paths
          alias /srv/melissi/server/env/lib/python2.6/site-packages/django/
↪contrib/admin/media/;
          expires 30d;
          add_header Cache-Control public;
    }

    location /storage/ {
          internal;
          # should match your paths
          alias /srv/storage/;
    }

    location / {
          proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
          proxy_set_header Host $http_host;
          proxy_set_header X-Forwarded-Protocol ssl;
          proxy_redirect off;
          proxy_pass http://app_server_melissi;
     }
}
```

(c) **Setup without SSL** (not recommended)

Warning: The communication between your clients and the server will be unecrypted. Your passwords and data will be viewable by others.

Create */etc/nginx/sites-availables/melissi* with the following contents:

```
upstream app_server_melissi {
        # should match your paths
        server unix:///srv/melissi/sockets/melissi.sock fail_timeout=0;
}

server {
   server_name example.com;
   listen *:8000;

   # should match your paths
   access_log /srv/melissi/logs/nginx.access.log;
   error_log /srv/melissi/logs/nginx.error.log;

   # set this to the maximum file size allowed in your hive
   client_max_body_size 100m;

   location  /static/admin/ {
         # should match your paths
         alias /srv/melissi/server/env/lib/python2.6/site-packages/django/
↪contrib/admin/media/;
         expires 30d;
         add_header Cache-Control public;
```

```
    }

    # /storage will be used in melissi configuration
    location /storage/ {
          internal;
          # should match your paths
          alias /srv/storage/;
    }

    location / {
          proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
          proxy_set_header Host $http_host;
          proxy_redirect off;
          proxy_pass http://app_server_melissi;
     }
}
```

(d) **Enable the site**:

```
~$ sudo ln -s /etc/nginx/sites-available/melissi ln -s /etc/nginx/sites-
↪enabled/melissi
```

(e) **Check configuration**:

```
~$ sudo /etc/init.d/nginx configtest
```

2. **Install and Setup Supervisor**

(a) **Install**:

```
~$ sudo apt-get install supervisor
```

(b) **Setup**

Create */etc/supervisor/conf.d/melissi.conf* with the following contents:

```
[program:melissi]
directory = /srv/melissi/server/melisi/
user = melissi
command = /srv/melissi/server/scripts/gunicorn.sh
autostart=true
autorestart=true
redirect_stderr=True
```

Warning: Make sure that the paths match your installation

(c) **Load new config**

```
~$ sudo /etc/init.d/supervisor force-reload
```

3. **Configure your Hive**

Some configuration is needed so that your Hive can take advantage of Nginx's X-Accel-Redirect directive.

Open *local_settings.py* and set:

```
SENDFILE='accel-redirect'
ACCEL_REDIRECT_PATH='/storage' # or whichever value you used in your nginx␣
↪configuration
```

4. Start services:

```
~$ sudo supervisorctl start melissi
~$ sudo /etc/init.d/nginx reload
```

5. Enjoy your hive!

Updating Your Hive

1. Update the source

```
~$ cd /path/you/installed/melissi
~$ ./scripts/melissi-installer.py --upgrade
```

2. If first step completes without errors, when run the install script, to download new packages

```
~$ ./scripts/melissi-install.py --install
```

3. Synchronize and migrate database

```
~$ source env/bin/activate
```

```
(env)~$ cd melisi
(env)~$ python manage.py syncdb
(env)~$ python manage.py migrate mlscommon
```

4. Restart your server

CHAPTER 2

Hive Administration

Things To Know

Adding and Removing Users

You can use Django's Admin with you superuser account to add and remove users.