

ikaaro 0.20

Admin's Guide

Juan David Ibáñez Palomar*

jdavid@itaapy.com

January 17, 2008

Contents

1	Introduction	2
1.1	Requirements	2
1.2	Download and Install	2
1.3	The command line interface	2
2	Make a new instance	3
2.1	The configuration file	3
3	Start/Stop the server	4
4	A look inside	5
4.1	The logs	5
4.2	The database	6
5	Deployment in a production environment	6
6	Upgrading to a new software version	7
7	Recovering from a crash	7

*Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. There is a copy of the license at <http://www.gnu.org/copyleft/fdl.html>

1 Introduction

1.1 Requirements

Python 2.5 and `itools` 0.20 are required. It is recommended to install the `pil`¹ and `docutils`² packages.

To be able to index office documents (ODF, PDF, etc.) the command line tools `wvText`, `xlhtml`, `pdftotext`, `ppthtml` and `unrtf` are needed.

1.2 Download and Install

Download the latest `ikaaro` version from <http://www.ikaaro.org/ikaaro>. Then to install it just type:

```
$ tar xzf ikaaro-X.Y.Z.tar.gz
$ cd ikaaro-X.Y.Z
$ python setup.py install
```

1.3 The command line interface

The `ikaaro` package includes a collection of command line tools to create and manage instances:

<code>icms-init.py</code>	creates a new <code>ikaaro</code> instance
<code>icms-start.py</code>	starts the web and the mail spool servers
<code>icms-stop.py</code>	stops the both servers
<code>icms-update.py</code>	updates the instance (after a software upgrade)
<code>icms-update-catalog.py</code>	rebuilds the catalog
<code>icms-restore.py</code>	recovers the instance (in case of a crash)

All the scripts are self-documented, just run any of them with the `--help` option. This is an excerpt for the `icms-init.py` script:

```
$ icms-init.py --help
Usage: icms-init.py [OPTIONS] TARGET

Creates a new instance of ikaaro with the name TARGET.

Options:
  --version          show program's version number and
                    exit
  -h, --help        show this help message and exit
  -a ADDRESS, --address=ADDRESS
                    listen to IP ADDRESS
  -e EMAIL, --email=EMAIL
```

¹<http://www.pythonware.com/products/pil/>

²<http://docutils.sourceforge.net>

```

                                e-mail address of the admin user
-p PORT, --port=PORT          listen to PORT number
-r ROOT, --root=ROOT         create an instance of the ROOT
                                application
-s SMTP_HOST, --smtp-host=SMTP_HOST
                                use the given SMTP_HOST to send
                                emails
-w PASSWORD, --password=PASSWORD
                                use the given PASSWORD for the
                                admin user

```

2 Make a new instance

To create a new instance we use the `icms-init.py` script. Example:

```

$ icms-init.py --email=jdavid@itaapy.com my_instance
*
* Welcome to ikaaro
* A user with administration rights has been created for you:
*   username: jdavid@itaapy.com
*   password: 7WEBJr
*
* To start the new instance type:
*   icms-start.py my_instance
*

```

(Take note of the automatically generated password, you will need it to enter the application through the web interface.)

The `icms-init.py` script creates a folder (named `my_instance` in the example) that keeps, among other things, the database and a configuration file:

```

$ tree -F -L 1 --noreport my_instance
my_instance
|-- catalog/
|-- config.conf
|-- database/
|-- log/
`-- spool/

```

2.1 The configuration file

Once the instance is created, it is a good idea to read the self-documented configuration file, `config.conf`, to learn about the available options, and to finish the configuration process.

The different options can be split in a four groups:

- The `modules` option allows to load (import) the specified Python packages when the server starts. This is the way we can extend the `ikaaro` CMS with third party packages.

- The `address` and `port` options define the internet address and the port number the Web server will listen to.

By default connections are accepted from any internet address. In a production environment it is wise to restrict the connections to only those coming from the localhost. Section 5 explains the details.

- The `smtp-host`, `smtp-login` and `smtp-password` are used to define the SMTP relay server that is to be used to send emails; and to provide the credentials for servers that require authentication.

The `contact-email` option must be a valid email address, it will be used for the `From` field in outgoing messages.

It is very important to set these options to proper values, since the `ikaaro` CMS sends emails for several important purposes.

- The `debug` option if set will output extra information to the events log, the `log/events` file.

3 Start/Stop the server

The `ikaaro` CMS uses two processes to get the job done: a Web server and a process to send emails asynchronously. There are two scripts to start either one or the other of these processes: `icms-start-server.py` and `icms-start-spool.py`. But usually we will use the `icms-start.py` script, which starts both:

```
$ icms-start.py my_instance
[my_instance] Web Server listens *:8080
[my_instance] Start Mail Spool.
```

By default the processes remain attached to the console, to stop them just type `Ctrl+C`. They are stopped *gracefully*, what means that pending requests will be handled and the proper responses sent to the clients.

To detach from the console use the `--detach` option. Then, to stop the servers started this way use the `icms-stop.py` script:

```
$ icms-start.py --detach my_instance
...
$ icms-stop.py my_instance
[my_instance] Web Server shutting down (gracefully)...
[my_instance] Mail Spool shutting down (gracefully)...
```

With the Web server running, we can open our favourite browser and go to the `http://localhost:8080` URL, to reach the user interface (see Figure 1).



Figure 1: The ikaaro Web interface.

4 A look inside

The content of an ikaaro instance is:

- The configuration file (see Section 2.1).
- The logs folder (see below).
- The database (see below).
- The catalog keeps the indexes needed to quickly search in the database.
- The mail pool keeps the emails to be sent by the spool server.

4.1 The logs

There are five log files:

- The access log uses the *Common Log Format*³, useful for example to build statistics about the usage of the web site.
- The error log keeps information about every *internal server* error, specifically the request headers and the Python tracebacks.
- By default the events log keeps record of the database transactions. In debug mode (see Section 2.1), more low-level information is recorded.
- The spool log keeps track of the emails sent by the spool server.
- The spool error log keeps information about every error coming from the spool server.

³<http://XXX>

4.2 The database

The data is stored directly in the file system. This is what a new instance looks like:

```
$ tree --noreport -F my_instance/database
my_instance/database
|-- .metadata
|-- users/
|   |-- 0.metadata
|   |-- users.metadata
```

The database is made up of regular files and folders. For instance, a Web Page will be stored in the database as an XHTML file, an image or an office document will be stored as it is.

This is extremely useful for introspection and manipulation purposes, since we can use the old good Unix tools: `grep`, `vi`, etc. But of course, *don't make any changes unless you know what you are doing!*

Metadata

Every `ikaaro` object is defined by a metadata file. As the example shows, a new instance has three objects: the root (defined by the `“.metadata“` file), the users folder, and the admin user created by the init script.

A metadata file looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<metadata format="user" version="20071215">
  <password>FNp6/Vb9cFeAMTlQNcFylxbToQ%3D%0A</password>
  <email>jdavid@itaapy.com</email>
</metadata>
```

5 Deployment in a production environment

In a production environment it is highly recommended to deploy `ikaaro` behind Apache⁴, using it as a proxy. The rewrite rule cannot be simpler:

```
<VirtualHost *:80>
  ServerName example.com
  RewriteEngine On
  RewriteRule ^/(.*) http://localhost:8080/$1 [P]
</VirtualHost>
```

To finish, the configuration of the `ikaaro` server should restrict (for security reasons) the internet addresses it accepts connections from to the localhost:

```
address = 127.0.0.1
```

⁴<http://http.apache.org>

Virtual Hosts

Most of the setup required for virtual hosting is defined in the `ikaaro` side, and through the Web interface. Regarding the rewrite rule the only thing needed is to add as many *aliases* as virtual hosts, for example:

```
<VirtualHost *:80>
  ServerName example.com
  ServerAlias vhost1.example.com
  ServerAlias vhost2.example.com
  RewriteEngine On
  RewriteRule ^/(.*) http://localhost:8080/$1 [P]
</VirtualHost>
```

6 Upgrading to a new software version

Generally major versions of `ikaaro` include changes to the layout or to the format of the information stored in the database that require an upgrade.

The update process has two steps:

```
# 1. Update the database
$ icms-update.py --yes my_instance
...
# 2. Rebuild the catalog
$ icms-update-catalog.py --yes my_instance
...
```

(The `icms-update-catalog` script may be used on a regular basis for maintenance purposes, since it has the side-effect of compressing the indexes.)

Anyway, any major version of `ikaaro` includes upgrade notes that detail any particular procedure. Start a version upgrade by reading these notes.

7 Recovering from a crash

Though unlikely, it may happen that the server crashes leaving a transaction in the middle, for example, if there is a power failure at the bad time. If this happens, the server will refuse to start again:

```
$ icms-start.py my_instance
The database is not in a consistent state, to fix it up type:

$ icms-restore.py <instance>
```

To recover the instance (by completing or removing the aborted transaction), just run the `icms-restore.py` script:

```
$ icms-restore.py my_instance
Restore database from backup (y/N)? y
* Restoring... DONE
```