Invenio Fabric Documentation

Release 0.2

CERN

August 05, 2015

Contents

Fabric library tasks for working with Invenio

CHAPTER 1

Requirements

- Install Virtualenv

Run `pip install virtualenv` or install via your favourite package manager (e.g. `sudo aptitude install python-virtualenv`).

- Install Virtualenvwrapper:

```
    pip install virtualenvwrapper
    export WORKON_HOME=~/envs
    mkdir -p $WORKON_HOME
    source /usr/local/bin/virtualenvwrapper.sh
```

Add line 2 and 4 to your shell startup file. Also note that depending on your system `virtualenvwrapper.sh` might be installed at a different location than /usr/local/bin. For more elaborate documentation, see http://virtualenvwrapper.readthedocs.org/en/latest/install.html

- Install Pythonbrew (optional)

Pythonbrew is optional, but it allows your to install several different python versions without messing up your system Python. To install Pythonbrew run:

```
curl -kL http://xrl.us/pythonbrewinstall | bash
```

and add following to your shell startup file:

```
[[ -s $HOME/.pythonbrew/etc/bashrc ]] && source $HOME/.pythonbrew/etc/bashrc
```

For more elaborate installation instructions please see https://github.com/utahta/pythonbrew. You should now be able to run e.g. `pythonbrew list` or to install Python 2.4.6 run `pythonbrew install 2.4.6`.

CHAPTER 2

Installation

Create a new virtualenv (optional):

```
mkvirtualenv fabenv
workon fabenv
```

Then install invenio-fabric via PyPI:

```
pip install invenio-fabric
export CFG_SRCDIR=~/private/src
```

Add the last line to your shell startup file.

Important: `CFG_SRCDIR` should not point to your Invenio source directory, but to one level above. Also, your Invenio source code directory should be named `invenio`. See directory layout below.:

```
$ export CFG_SRCDIR=~/src
$ cd CFG_SRCDIR
$ ls -1
invenio
$ cd CFG_SRCDIR/invenio/
$ ls -1
ABOUT-NLS
aclocal.m4
AUTHORS
autom4te.cache
ChangeLog
config
config.guess
...
```

Note, you do not need to specify `CFG_SRCDIR`, in which case the Fabric task will checkout a fresh copy from the GIT repository.

CHAPTER 3

Usage documentation

CHAPTER 4

API documentation

4.1 Apache

Tasks for Apache start/stop/restarting apache.

All tasks assume `env.CFG_INVENIO_APACHECTL` is defined and points to your Apache control script (e.g. `/etc/init.d/apache2` or `../apachectl`). The script must support the following commands: start, stop, configtest, graceful.

Warning: These tasks are not working locally like the rest Invenio Fabric library.

`inveniofab.apache.`**`apache_start`**
Start Apache

`inveniofab.apache.`**`apache_stop`**
Stop Apache

`inveniofab.apache.`**`apache_restart`**
Restart Apache

The task will first test the configuration and afterwards gracefully restart Apache.

4.2 Compound tasks

Compound tasks to perform bigger operations like bootstrapping Invenio.

`inveniofab.compound.`**`bootstrap`**
Bootstrap an Invenio installation

Bootstrap will run the following tasks:

- `mysql_dropdb` - to drop an existing database if it exists.
- `mysql_createdb` - to create database and user.
- `venv_create` - to create a virtual environment.
- `repo_update` - to checkout source code from repositories.
- `venv_requirements` - to install Python requirements.
- `repo_install` - to install all repositories.
- `invenio_conf` - to configure Invenio.

- devserver_conf - to configure the development server.
- invenio_createdb - to create Invenio database tables.

If a task fails, you can re-run bootstrap and skip the initial steps which already was completed.

Parameters

- **quite** – Default False. Set to True to disable user confirmation.
- **with_db** – Default True. Run database related tasks to create the database.

inveniofab.compound.**dump**
Archive installation

Dump a database and virtual environment to an archive which can later be restored with *load()*.

inveniofab.compound.**load**
Load archived installation

Load an archived virtual environment and database which was dumped with *dump()*.

inveniofab.compound.**drop**
Remove installation

Remove virtual environment, database and database user.

inveniofab.compound.**install**
Install repositories

The task will checkout latest changes for the repositories, run make install, inveniocfg --update-all --upgrade.

Parameters **quite** – Default False. Set to True to disable user confirmation.

4.3 Development server

Task to work with devserver

inveniofab.devserver.**devserver_conf**
Render and update invenio-devserver configuration

The task will look for the template config_local.py.tpl, and render and write it to config_local.py in the virtual environments site-packages.

The invenio-devserver install two commands:

- serve - Invenio development server based on Werkzeug.
- mailserve - Debug mail server which will print all emails to the console.

Note: The invenio-devserver works with the non-Flask based versions of Invenio. Also, the invenio-devserver is only installed if env.WITH_DEVSERVER is True.

See also:

See also invenio-devserver for further information on the content of config_local.py.tpl: https://bitbucket.org/osso/invenio-devserver

inveniofab.devserver.**devserver_install_flask**
Install a Flask devserver

The task will look for the template `rundevserver.py.tpl`, render it and write it to `bin/rundevserver.py`.

To start the Flask development server, run:

```
(venv)$ rundevserver.py
```

Note: `rundevserver.py` only works with Flask based versions of Invenio.

4.4 Environment

Fabric tasks for defining environments.

4.4.1 Fabric environment dictionary

An integral part of Fabric is what is know as "environments": a Python dictionary subclass which is used as a combination settings registry and shared inter-task data namespace. Most of Fabric's and Invenio Fabrics behaviour is modifiable through env variables.

The environment is accessible through `fabric.api.env`, and allows e.g. a task access to the current user etc (`user` is called an env variable):

```
from fabric.api import env
@task
def sometask():
    print env.user
    print env.host
```

For more information on Fabric environment dictionary please see http://fabric.readthedocs.org/en/latest/usage/env.html

4.4.2 Invenio Fabric environments

The tasks in this module helps you define the env variables which all other Invenio Fabric tasks depends on. Secondly, it provide means for loading env variables, but without putting them in `fabric.api.env`. This is useful for tasks like `fab int mysql_copy:prod` which would copy the database from production to integration, and thus need access to host, user, password etc from both the integration and production environment at the same time.

Usage

Defining a new environment is simple:

```
from fabric.api import task
from inveniofab.api import *

@task
def prod(activate=True):
    # Step 1: Create env with default values
    env = env_create('prod', activate=activate)

    # Step 2: Modify some values
    env.CFG_XYZ = ...
```

```
    # Step 3: Return env
    return env
```

Things to note:

- The task must take a keyword argument `activate` which defaults to `True`. This is used to control if the environment is activated or not (i.e. copied to `fabric.api.env`.
- First parameter to *`env_create()`*, must match the task name (`prod` in this case). Otherwise the environment cannot be loaded without activating it, and template overriding will not work properly (see later in this chapter).
- You must return the environment variable at the end of the task.

Running tasks

As mentioned above, the Invenio Fabric tasks depends on an environment being defined. The pattern to run Invenio Fabric task is:

```
fab <env> <task1> <task2>
```

In the example above, you could for instance run:

```
fab prod bootstrap
```

to bootstrap the production environment. I.e. you always call an environment task as the first task.

Also note, that some task takes parameters:

```
fab <env> <task1>:<arg1>,<arg2>,<kw1>=<val>,<kw2>=<val> <task2> ...
fab prod bootstrap:quite=1
```

Roles

By default the following roles are defined:

- `web` - frontend webservers
- `lb` - load balancers
- `db-master` - master databases
- `db-slave` - slave databases
- `workers` - worker nodes

Some task will use these roles to decide which machine to execute a given command on. For more on Fabric roles please see http://fabric.readthedocs.org/en/1.4.3/usage/execution.html#roles

You may override these roles in your environment task:

```
@task
def prod(activate=True):
    env = env_create('prod', activate=activate)
    env.roledefs['web'] = [...]
    return env
```

4.4.3 Creating tasks

When creating task you can access the variables simply by loading the Fabric env dictionary from `fabric.api.env`. All Invenio Fabric env variables are upper-case (for now, please look in the source code of *`env_defaults()`* to see, which variables are defined).

Example:

```
from fabric.api import env

@task
def sometask():
    print env.CFG_INVENIO_PREFIX
    local("%(CFG_INVENIO_PREFIX)s/bin/python" % env)
```

Template loading

The Invenio Fabric environment by default also configures a Jinja2 environment which can be used to load and render templates.

Example:

```
from fabric.api import env

@task
def sometask():
    tpl = env.jinja.get_template("etc/invenio-local.conf")
    output = tpl.render(env)
    ...
```

Above code will look for the template first in `<env>/etc/invenio-local.conf`, then `common/etc/invenio-local.conf`. So assuming your directory structure looks like below:

```
fabfile.py
common/etc/invenio-local.conf
int/etc/invenio-local.conf
```

Running `fab int sometask` would use the template `int/etc/invenio-local.conf`, while `fab prod sometask` and `fab loc sometask` would use the template `common/etc/invenio-local.conf`.

This allows you to define common templates for configuration files, and selectively override them for certain environments if needed.

4.4.4 Relation to invenio-local.conf

Many variables map directly to their Invenio counterpart - e.g. `CFG_DATABASE_NAME` can be defined in invenio-local.conf as well as in Invenio Fabric `env.CFG_DATABASE_NAME`.

Usually when deploying or working with a specific Invenio project, some configuration variables depend only on the project (e.g. `CFG_WEBSTYLE_TEMPLATE_SKIN`), while others depend on the deployment environment (e.g. `CFG_DATABASE_HOST`).

As a general rule, project-wide configuration should go in a common `invenio-local.conf` template, while deployment dependent configuration should be defined in a Fabric environment task. The `invenio-local.conf` can then be render with the Fabric env variables.

4.4.5 Tasks

inveniofab.env.**env_create**(*envname*, *defaults_func=None*, *activate=True*, ***kwargs*)
Setup a new environment (e.g. integration, production, local).

See module documentation above for detailed explanation of enviroments.

Parameters

- **envname** – str, Name of environment (must be same as task envname).
- **defaults_func** – callable taking a dictionary as argument and returns the same dictionary. Used to setup defaults in the environment. By default the *env_defaults()* is used. Take great care if overriding this, since many tasks expects specific variables to be defined.
- **activate** – True to activate the environment, or False to just load it (i.e. should config be put in fabric.api.env or not).

inveniofab.env.**env_defaults**(*env*, *name='invenio'*, *prefix=None*, *python=None*, ***kwargs*)
Setup defaults in environment.

The method will by default try to guess

Parameters

- **name** –
- **prefix** –
- **python** –

inveniofab.env.**env_get**(*name*)
Get environment by name (does not activate the environment).

An environment is defined in a local task. This task will look for a task with the name <name> and execute. Hence the task defining the environment.

Example:

```
from fabric.api import env

@task
def sometask():
    another_env = env_get('prod')
    print env.CFG_SITE_URL
    print another_env.CFG_SITE_URL
```

Running fab int sometask would print first the integration CFG_SITE_URL, then the production CFG_SITE_URL.

inveniofab.env.**env_override**(*env*, *this_repo*, *this_ref*, *override={}*, *global_overrides=None*)
Override default values for repository

inveniofab.env.**env_make_name**(*prefix*, *python*, *ref*)
Generate a MySQL friendly environment name.

4.5 GIT version control

Task for checking out source code from a repository, and running configure, make on it.

inveniofab.git.**default_configure_hook**(*ctx*)
Default way to configure a repo. Assumes repo has a configure script.

inveniofab.git.**default_prepare_hook**(*ctx*)
Default way to prepare source code which uses autotools.

inveniofab.git.**git_checkout**(*repo*, *ref*)
Checkout a specific git reference.

inveniofab.git.**git_isdirty**(*dir*)
Check working directory for uncommitted changes

inveniofab.git.**repo_configure**
Configure repository

inveniofab.git.**repo_install**
Run configure and make

inveniofab.git.**repo_make**
Run make in repository

inveniofab.git.**repo_prepare**
Prepare source code after fresh checkout

inveniofab.git.**repo_setup**
Clone repository

inveniofab.git.**repo_update**
Pull repository updates

4.6 MySQL

Library tasks for configuring and running MySQL for Invenio.

inveniofab.mysql.**mysql_conf_type**
Upload and update MySQL configuration

inveniofab.mysql.**mysql_copy**
Copy database from latest available dump.

Currently it is assumed that the dump file is accessible on the same path on both environment host systems. This usually means that the dumps are stored on a shared network storage.

inveniofab.mysql.**mysql_createdb**
Create database and user

inveniofab.mysql.**mysql_dropdb**
Drop database and user

inveniofab.mysql.**mysql_dump**
Dump database to file

inveniofab.mysql.**mysql_load**
Load MySQL dump file

4.7 Test

Tasks for creating and loading a test fixture package to easily run tests against a known state.

inveniofab.test.**test_clean**
Clean Invenio logs and temporary files

inveniofab.test.**test_dump**
Dump a test environment

inveniofab.test.**test_load**
Load test environment

inveniofab.test.**test_reset_admin**
Reset admin password

4.8 Utils

inveniofab.utils.**is_local**()
Determine if env.host is localhost

inveniofab.utils.**prompt_and_check**(*questions*, *check_func*, *cache_key=None*, *stored_answers=None*)
Ask user for questions, and check answers with supplied function

inveniofab.utils.**python_version**()
Determine Python version

inveniofab.utils.**pythonbrew_versions**()
Get all installed Pythonbrew versions

inveniofab.utils.**run_local**(*command*, *shell=True*, *pty=True*, *combine_stderr=True*, *capture=False*, *warn_only=False*)
run/local function based on host

inveniofab.utils.**sublime_project**
Write sublime project file

inveniofab.utils.**sudo_local**(*command*, *shell=True*, *pty=True*, *combine_stderr=True*, *user=None*, *capture=False*, *warn_only=False*)
sudo/local function based on host

inveniofab.utils.**symlinks**
Create symlinks specified by CFG_SYMLINKS

inveniofab.utils.**template_hook_factory**(*tpl_file*, *filename*, *warn_only=True*)
Factory method for generating hook functions that renders a template, and writes it to a specific location.

Filename may include string replacement like e.g. %(CFG_INVENIO_PREFIX)s.

inveniofab.utils.**upload_files**
Upload files specified by CFG_FILES

inveniofab.utils.**write_template**(*filename*, *context*, *tpl_str=None*, *tpl_file=None*, *remote_tpl_file=None*, *append=False*, *mark=None*, *use_sudo=False*)
Render template and write output to file

@param filename: File to write @param template: Name of template @param context: Dictionary for the template context (usually just env) @param append: bool, True if you want to append content to file instead of overwriting. @param mark: str, If append is true and mark is defined, then the rendered

template will replace any previous appened version.

4.9 Virtualenv

Tasks for creating and installing Python virtual environments.

All task works on local system.

inveniofab.venv.**venv_create**
Create virtualenv environment

The virtualenv is created in `env.CFG_INVENIO_PREFIX`, and will also create `lib/python/invenio/` and symlink it the virtualenv's site-packages, as well as `var/tmp/ooffice-tmp-files` (via sudo). If `env.WITH_DEVSCRIPTS` is `True`, invenio-devscripts will be installed. If `env.WITH_WORKDIR` is `True` git-new-workdir will be installed.

Lastly, it will append render the template `activate-profile.tpl` and append it to `bin/activate`. The script will setup common needed environment variables that e.g. invenio-devscripts depend on.

If an existing environment already exists, the user will be asked for confirmation to remove the directory (using sudo, due to the directory `var/tmp/ooffice-tmp-files` which is created using sudo).

inveniofab.venv.**venv_drop**
Drop virtualenv environment

inveniofab.venv.**venv_load**
Load an archived virtualenv

The task will extract an archived virtual environment created with *venv_dump()*. Normally this command is invoked indirectly via the compound task *inveniofab.compound.load()* which takes care of loading the database after extracting the virtual environment.

inveniofab.venv.**venv_dump**
Archive a virtualenv

The task will create an archive `<virtualenv name>.tar.gz` of the entire virtual environment. If an existing archive already exists, the user will be asked for confirmation to remove it. Normally this command is invoked indirectly via the compound task *inveniofab.compound.dump()* which takes care of dumping the database prior to archiving the virtual environment.

inveniofab.venv.**venv_requirements**
Install Python packages

The task will install Python packages defined in PIP requirements file.

inveniofab.venv.**venv_pyuno_install**
Install Python OpenOffice binding

The task will try to locate `uno.py` and `unohelper.py` in `/usr/`, and copy it to your virtualenv's site-packages.

> **Warning:** The Python OpenOffice bindings from your system is specific to your system's Python interpreter, hence if your system Python is 2.7 and you are installing the bindings into virtualenv with Python 2.4, the bindings will not work.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

i

V

W