EUROfusion actor release procedure for IMAS

Documentation

Documentation is also available on PSNC's Confluence page

Content of the repository

- actor_install.py: script that automates the build of IMAS actor(s)
- TEMPLATE.yml: template description of a released actor
- *.yml: currently released projects/actors (as provided by the community)

Pre-requisites

The script depends on a the yaml python packages which might not be part of most standard installs.

It expects that you can access the repositories (GIT or SVN) that contain the targeted actors/projects.

For project stored on Gateway GFORGE, svn+ssh is not allowed, so checking out sources will work only if you have stored previously username and password. For other repository, we advise that you use access through ssh-key. In case you have set a passphrase with for your private key, you should use ssh-agent and make sure that you have typed the passphrase (using ssh-add) before running calling the actor_install.py script.

Description of a release

Start by taking a look at TEMPLATE.yml, and adapt the different information to your own needs for the targeted project/actors.
This YAML file contains five sections (mappings) as follow:
- `SOURCES` : *required* information on Version Control System where the sources of the project(s) are stored.
- `MODULES` : *required* information on module environment needed to attempt the build of the libraries
- `BUILDS` : *required* information on how to build the libraries
- `ACTORS` : *required* information on where FC2K project files are stored
- `COMMENTS` : *optional* any comments, notes, documentation from developers of the code

Note: all paths are given relatively to the working directory (created automatically to be unique inside the current directory) or specified with CLI option `-D <working_dir_path>`.

SOURCES section

Content

A *list (sequence) of mappings* with:
- `VCS` : *required* type of Version Control System
- `REPO` : *required* URL of the repository, including path to branch/tag
- `DIR` : *required* target directory in which to store this project sources
- `VERSION` : *optional* target version of the sources

Description

This step will checkout/clone the source code and store it under the specified sub-directory `DIR`.

Associated command line options

- `--skipSources` : do not checkout/clone sources (makes sense only with `-D <working_dir_path>`)
- `-R/--checkRevision <version>` : specifies and check revision number or a commit hash

MODULES section

Content

A sequence of modules (e.g. `[imasenv, fc2k/4.3.0]`).

Description

After doing module purge to ensure reproducibility, the specified list of modules will be loaded, while already loaded modules will be switched to specified version.

Associated command line options

- `-M <premodule>` : load any site specific module(s) before the ones listed in the YAML file (e.g `-M cineca` on the Gateway)
- `--skipModules` : do not purge and load specified modules but use currently loaded ones

BUILDS section

Content

A sequence of build mappings, with:
- `DIR` : directory from which the build command should be executed
- `CMD` : build command (script, makefile, etc...)

Description

The specified commands `CMD` will be executed from their associated directory `DIR` to build all the libraries required by the actors.

Associated command line options

- `--skipBuilds` : do not attempt to build the libraries (makes sense only with `-D <working_dir_path>`)

ACTORS section

Content

A sequence of FC2K mappings, with:
- `DIR` : directory from which fc2k should be executed
- `XML` : path to FC2K project file for a given actor

Description

FC2K will be executed from specified directory `DIR` on associated project file `XML` in order to install the actor.

Associated command line options

- `--skipActors` : do not attempt to create the actor (makes sense only with `-D <working_dir_path>`)
- `-K/--setKepler <KEPLER>` : sets non-standard KEPLER variable, now mandatory as Kepler module do no set `$KEPLER` but `$KEPLER_SRC` (since version 2.5p4-3.0.6), unless `--skipModules` option is used

(then make sure you have done a `kepler_load` before calling this script)

COMMENTS section

Content

A long string (starts with `|`) that can span on several lines with line breaks preserved as long as initial indent is there.

Description

At the moment this is only informative for people reading the yaml file, not processed.

How-to test and publish a release

One the YAML file is written, you can use the actor_install.py script to install it in your Kepler. As explain in the help accessible with option -h / –help, It takes as last argument(s) the YAML file(s).

If no error where discover during all the steps (you can use option -p / –pedantic to stop at first encounter error) you can commit you YAML file to the repository.

Building examples on the EUROfusion Gateway

Minimal required environment

```
module load cineca itm-python/3.6
```

Kepler

You need to have a Kepler at your disposal to be the target for actors installation.
This target Kepler can be installed publickly (only if you have sufficient access rights) or locally (through usual commands `kepler_install`, `kepler_avail`, `kepler_load`).

Selecting this target Kepler depends on either you skip the modules section of the install procedure (`--skipModules`) or not.

Default mode

When installing an actor using the modules specified in the Yaml file, as the install script will start by purging the environment you need to specify the exact location of the target Kepler through the command line `-K/--setKEPLER`. If you are not sure where your locally installed Kepler is stored, simply do:

```
kepler_avail
kepler_load <name_of_my_kepler>
./actor_install.py -M cineca -K $KEPLER <yaml file(s)>
```

Skip modules mode

In this mode you can control the entire environment in which a code is compiled and the actor installed. Make sure you have loaded the wanted target Kepler (`kepler_load <name_of_my_kepler>`) before running the install script with `--skipModules` option:

```
kepler_avail
kepler_load <name_of_my_kepler>
./actor_install.py --skipModules <yaml file(s)>
```

Example commands

- print help:

```
./actor_install.py -h
```

- installs chease:

```
./actor_install.py -M cineca -K $KEPLER chease.yml
```

- installs cyrano, verbose mode:

```
./actor_install.py -M cineca -K $KEPLER -v cyrano.yml
```

- installs gkmhd, pedantic mode (stops at first error):

```
./actor_install.py -M cineca -K $KEPLER -p gkmhd.yml
```

- installs both gray and neowes:

```
./actor_install.py -M cineca -K $KEPLER gray.yml neowes.yml
```

- attempts to install all present actors/projects (but TEMPLATE):

```
./actor_install.py -M cineca -K $KEPLER *.yml
```

- installs all actors using tmpdir as working dir for checkout and build:

```
./actor_install.py -M cineca -K $KEPLER -D tmpdir *.yml
```

Some more advanced commands

- build and install chease actor, without checking out sources again (need to be present in tmpdir):

```
./actor_install.py -M cineca -K $KEPLER -D tmpdir --skipSources chease.yml
```

- install chease actor, sources and compiled lib need to be present in tmpdir already:

```
./actor_install.py -M cineca -K $KEPLER -D tmpdir --skipSources --skipBuilds chease.yml
```

- install all actors using locally prepared environment:

```
./actor_install.py --skipModules *.yml
```