ESM Tools r3 UserManual

Dirk Barbi, Nadine Wieters, Paul Gierz, Fatemeh Chegini

Mar 23, 2020

CONTENTS:

01000101
01010011
01001101 ESM-Tools

CHAPTER

ONE

INTRODUCTION

This is the user manual for the esm tools. To contribute to this document, please contact the authors for feedback.

The esm-tools are a collection of scripts to download, compile, configure different simulation models for the Earth system, such as atmosphere, ocean, geo-biochemistry, hydrology, sea-ice and ice-sheet models, as well as coupled Earth System Models (ESMs). They include functionality to write unified runscripts to carry out model simulations for different model setups (standalone and ESMs) on different HPC systems.

The ESM-Tools are divided into a number of python packages / git repositories, both to ensure stability of the code as well as reusability:

1.1 esm_tools.git

The only repository to clone by hand by the user, `esm_tools.git` contains the subfolders

configs: A collection of yaml configuration files, containing all the information needed by the python packages to work properly. This includes machine specific files (e.g. `machines/mistral.yaml`) , model specific files (e.g `fesom/fesom-2.0.yaml`), configurations for coupled setups (e.g. `foci/foci.yaml`), but also files with the information on how a certain software works (`batch_systems/slurm.yaml`), and finally, how the esm_tools themselves are supposed to work (e.g. `esm_master/esm_master.yaml`).

1.2 esm_master.git

This repository contains the python files that give the `esm_master` executable in the subfolder `esm_master`.

1.3 esm_runscripts.git

The python package of the `esm_runscripts` executable. The main routines can be found in `esm_runscripts/esm_sim_objects.py`.

1.4 esm_parser.git

In order to provide the additional functionality to the `yaml+` configuration files (like choose blocks, simple math operations, variable expansions etc.). `esm_parser` is an extension of the pyyaml package, it needs the `esm_calendar` package to run, but can otherwise easily be used to add `yaml+` configurations to any python software.

1.5 esm_calendar.git

CHAPTER

TWO

INSTALLATION

2.1 Downloading

`esm_tools` is hosted on https://gitlab.awi.de, with a mirror on https://gitlab.dkrz.de . To get access to the software, you need to be able to login to one of these two servers.

gitlab.awi.de:

- DMAWI login: open to all employees of the Alfred Wegener Helmholtz Institute for Polar and Climate Research. Make sure the DMAWI tab is active, then use your normal AWI LDAP login and password to sign in.
- Shibboleth Login: open to employees of member organizations of the DFN-AAI (see https://tools.aai.dfn.de/entities/ for a list of participating identity providers). Click "Federated Login" on the right side, and choose your organization from the list. User your affiliations username and password to identify yourself, and navigate through the upcoming questionaire. You should then be directed to the server gitlab.awi.de. Please notice that you are asked to choose a password for this server, so that from that moment on you can use your e-mail address and the new password for login / git access. Shibboleth login should work for hundreds of institues, including e.g. DKRZ and GEOMAR.

gitlab.dkrz.de: Open for everyone with a DKRZ account, and activated gitlab acess in the project page.

If you encounter any problems with getting access to gitlab.awi.de, please feel free to contact dirk.barbi@awi.de.

Once you can access the server, you will need to become a member of the group `esm_tools`. Either look for the group and request membership, or directly contact dirk.barbi@awi.de.

Now that you have access to one of the download servers, and to the `esm_tools` group, you can start by cloning the repository `esm_tools.git`:

```
$> git clone https://gitlab.awi.de/esm_tools/esm_tools.git
```

This gives you a collection of yaml configuration files containing all the information on models, coupled setups, machines etc. in the subfolder `config`, default namelists in the folder `namelists`, example runscripts for a large number of models on different HPC systems in subfolder `runscripts`, and this documention in `docs`. Also you will find the installer `install.sh` used to install the python packages.

2.2 Before you continue

You will need python 3 (possibly version 3.5 or newer) and also a version of git that is not ancient (everything newer than 2.10 should be good) to install the esm_tools. That means that on the supported machines, you could for example use the following settings:

ollie.awi.de:

```
module load git
module load python3
```

mistral.awi.de:

```
module load git
module load anaconda3
```

glogin.hlrn.de / blogin.hlrn.de:

```
module load git
module load anaconda3
```

juwels.fz-juelich.de:

```
module load git
module load Python-3.6.8
```

2.3 Installing

To use the new version of the esm-tools, now rewritten in Python, clone this repository:

```
git clone https://gitlab.awi.de/esm_tools/esm_tools.git
```

Then, run the `install.sh`:

```
./install.sh
```

You should now have the command line tools `esm_master` and `esm_runscripts`, which replace the old version.

You may have to add the installation path to your `PATH` variable:

```
export PATH=~/.local/bin:$PATH
```

2.4 Configuration

If you have installed `esm_tools` you need to configure ìt before the first use to setup the hidden file `$HOME/.esmtoolsrc` correctly. This configuration will set required user information that are needed by both `esm-master` and `esm_runscripts` to work correctly. Such information are your user accounts on the different software repositories, your account on the machines you want to compute on, and some basic settings for the esm-runscripts.

To configure esm-master you should run the executable:

```
$> ./esm_master
```

Running it for the first time after installation, you will be asked to type in your user settings. This interactive configuration includes the following steps:

```
$> Please enter your username for gitlab.dkrz.de (default: anonymous)
$> Please enter your username for swrepo1.awi.de (default: anonymous)
```

CHAPTER

THREE

TRANSISTIONING FROM THE SHELL VERSION

3.1 ESM-Master

The Makefile based esm_master of the shell version has been replaced by a (python-based) executable called `esm_master` that should be in your PATH after installing the new tools. The command can be called from any place now, models will be installed in the current work folder. The old commands are replaced by new, but very similar calls:

```
OLD WAY:                              NEW WAY:
make                        -->       esm_master                      (to get the list of␣
↪available
                                                                       targets)
make get-fesom-1.4          -->       esm_master get-fesom-1.4        (download)
make conf-...               -->       esm_master conf-...             (configure)
make comp-...               -->       esm_master comp-...             (compile)
make clean-...              -->       esm_master clean-...            (clean)
```

Apart from that, the new esm_master offers certain new functionality:

```
esm_master fesom          (lists all available targets containing the string "fesom")
esm_master install-...    (shortcut for: get- , then conf- , then comp-)
esm_master recomp-...     (shortcut for: conf-, then clean-, then comp-)
esm_master log-...        (overview over last commits of the model, e.g. git log)
esm_master status-...     (changes in the model repository since last commit, e.g.␣
↪git status)
```

If the user wants to define own shortcut commands, that can be done by editing `esm_tools/configs/esm_master/esm_master.yaml`. New wrappers for the version control software can be e.g. added in `esm_tools/configs/vcs/git.yaml`. Adding commands in these configuration files is sufficient that they show up in the list of targets.

The details about models, setups, etc. are now to be found in `esm_tools/configs/esm_master/setups2models.yaml`. This file is a strucutred list instead of a barely readable, and rapidly growing, makefile. If you want to change details of your model, or add new components, this is where it should be put. Please refer to the chapter ESM-Master down below for further details.

3.2 ESM-Environment

A visible tool, like esm-environment used to be, doesn't exist anymore. The information about the environment needed for compiling / running a model is contained:

- in the machine yaml file (e.g. `esm_tools/configs/machines/ollie.yaml`): This contains a default environment that we know works for a number of models / setups, but maybe not in an optimal way,
- in the model yaml file (e.g. `esm_tools/configs/fesom/fesom-2.0.yaml`): The model files are allowed to contain deviations from the default environment defined in the machine file, indicated by the keywords `environment_changes`, `compiletime_environment_changes` or `runtime_environment_changes`.

Please note that even though there still is a python package called `esm_environment`, this is just the collection of python routines used to assemble the environment. It does not contain anything to be configured by the user.

3.3 ESM-Runscripts

One main thing that has changed for the runtime tool is the way it is evoked:

```
OLD WAY:                                    NEW WAY:
./runscriptname -e experiment_id            esm_runscripts runscriptname -e experiment_id
```

Instead of calling your runscript directly, it is now interpreted and executed by the wrapper `esm_runscripts`, the second executable to be added to your PATH when installing the Tools. Internally, `esm_runscripts` reads in the script file line by line and converts it into a python dictionary. It is therefore also possible to write the "runscripts" in the form of a yaml file itself, which can be imported by python much easier. The user is invited to try the yaml-style runscripts, some example can be found in `esm_tools/runscripts`.

Some of the variables which had to be set in the script when using the shell version are now deprecated, these include:

- FUNCTION_PATH
- FPATH
- machine

Also the last two lines of the normel runscript for the shell version of the tools, `load_all_functions` and `general_do_it_all`, don't do anything anymore, and can be safely removed. They don't hurt though.

(. . . to be continued. . .)

3.4 Namelists

No changes. Namelists can be found in `esm_tools/namelists`.

CHAPTER

FOUR

ESM MASTER

4.1 Usage: esm-master

To use the command line tool `esm_master`, just enter at a prompt:

```
esm_master
```

The tool may ask you to configure your settings; which are stored in your home folder under `${HOME}/.esmtoolsrc`. A list of avaiable models, coupled setups, and available operations are printed to the screen, e.g.:

```
setups:
   awicm:
      1.0: ['comp', 'clean', 'get', 'update', 'status', 'log', 'install', 'recomp']
      CMIP6: ['comp', 'clean', 'get', 'update', 'status', 'log', 'install', 'recomp']
      2.0: ['comp', 'clean', 'get', 'update', 'status', 'log', 'install', 'recomp']
[...]
```

As can be seen in this example, `esm_master` supports operations on the coupled setup awicm in the versions 1.0, CMIP6 and 2.0; and what the tool can do with that setup. You execute `esm_master` by calling:

```
$> esm_master operation-software-version,
```

e.g.:

```
$> esm_master install-awicm-2.0
```

By default, `esm_master` supports the following operations:

get: Cloning the software from a repository, currently supporting git and svn

conf: Configure the software (only needed by mpiesm and icon at the moment)

comp: Compile the software. If the software includes libraries, these are compiled first. After compiling the binaries can be found in the subfolders `bin` and `lib`.

clean: Remove all the compiled object files.

install: Shortcut to get, then conf, then comp.

recomp: Shortcut to conf, then clean, then comp.

update: Get the newest commit of the software from the repository.

status: Get the state of the local database of the software (e.g. `git status`)

log: Get a list of the last commits of the local database of the software (e.g. `git log`)

To download, compile, and install `awicm-2.0`; you can say:

```
esm_master install-awicm-2.0
```

This will trigger a download, if needed a configuration, and a compilation process. Similarly, you can recompile with `recomp-XXX`, clean with `clean-XXX`, or do individual steps, e.g. `get`, `configure`, `comp`.

The download and installation will always occur in the **current working directory**.

You can get further help with:

```
esm_master --help
```

CHAPTER

FIVE

FREQUENTLY ASKED QUESTIONS

5.1 Installation

1. **Q**: My organization is not in the pull-down list I get when trying the Federated Login to gitlab.awi.de.

   A: Then maybe your institution just didn't join the DFN-AAI. You can check that at https://tools.aai.dfn.de/entities/.

2. **Q**: I am trying to use the Federated Login, and that seems to work fine. When I should be redirected to the gitlab server though, I get the error that my uid is missing.

   A: Even though your organization joined the DFN-AAI, gitlab.awi.de needs your organization to deliver information about your institutional e-mail address as part of the identity provided. Please contact the person responsible for shibboleth in your organization.

5.2 ESM Runscripts

1. **Q**: I get the error: `load_all_functions:  not found [No such file or directory]` when calling my runscript like this:

   ```
   $> ./my_run_script.sh -e some_expid
   ```

   A: You are trying to call your runscript the old-fashioned way that worked with the shell-script version, until revision 3. With the new python version, you get a new executable `esm_runscripts` that should be in your PATH already. Call your runscript like this:

   ```
   $> esm_runscripts my_run_script.sh -e some_expid
   ```

   All the command line options still apply. By the way, "load_all_function" doesn't hurt to have in the runscript, but can savely be removed.

2. **Q**: What should I put into the variable `FUNCTION_PATH` in my runscript, I can't find the folder `functions/all` it should point to.

   A: You can safely forget about `FUNCTION_PATH`, which was only needed in the shell script version until revision 3. Either ignore it, or better remove it from the runscript.

5.3 ESM Master

1. **Q**: How can I define different environments for different models / different versions of the same model?

   A: You can add a choose-block in the models yaml-file, e.g.:

```
choose_version:
        40r1:
                environment_changes:
                        add_export_vars:
                                - 'MY_VAR="something"'
                        add_module_actions:
                                - load my_own_module

        43r3:
                environment_changes:
                        add_export_vars:
                                - 'MY_VAR="something_else"'
```

CHAPTER

SIX

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

6.1 Types of Contributions

6.1.1 Report Bugs

Report bugs at https://gitlab.awi.de/esm_tools/esm_tools/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

6.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

6.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

6.1.4 Write Documentation

ESM Tools could always use more documentation, whether as part of the official ESM Tools docs, in docstrings, or even on the web in blog posts, articles, and such.

6.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://gitlab.awi.de/esm_tools/esm_tools/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

6.2 Get Started!

Ready to contribute? Here's how to set up *esm_tools* for local development.

1. Fork the *esm_tools* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone https://your_name_here@gitlab.awi.de/esm_tools/esm_tools.git
   ```

   (or whatever subproject you want to contribute to).
3. By default, `git clone` will givw you the release branch of the project. You might want to consider checking out the development branch, which might not always be as stable, but usually more up-to-date than the release branch:

   ```
   $ git checkout develop
   ```

4. Create a branch for local development:

   ```
   $ git checkout -b name-of-your-bugfix-or-feature
   ```

   Now you can make your changes locally.
5. When you're done making changes, check that your changes pass flake8:

   ```
   $ flake8 esm_tools
   ```

6. Commit your changes and push your branch to GitHub:

   ```
   $ git add .
   $ git commit -m "Your detailed description of your changes."
   $ git push origin name-of-your-bugfix-or-feature
   ```

7. Submit a pull request through the gitlab website.

6.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/dbarbi/esm_tools/pull_requests and make sure that the tests pass for all supported Python versions.

6.4 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

CHAPTER

SEVEN

CREDITS

7.1 Development Lead

- Dirk Barbi <dirk.barbi@awi.de>
- Paul Gierz <paul.gierz@awi.de>
- Nadine Wieters <nadine.wieters@awi.de>

7.2 Project Management

- Luisa Cristini <luisa.cristini@awi.de>

7.3 Contributors

- Sara Khosravi <sara.khosravi@awi.de>
- Fatemeh Chegini <fatemeh.chegini@mpimet.mpg.de>
- Joakim Kjellsson <jkjellsson@geomar.de>
- ...

7.4 Beta Testers

- Tido Semmler <tido.semmler@awi.de>
- Christopher Danek <christopher.danek@awi.de>
- ...

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search