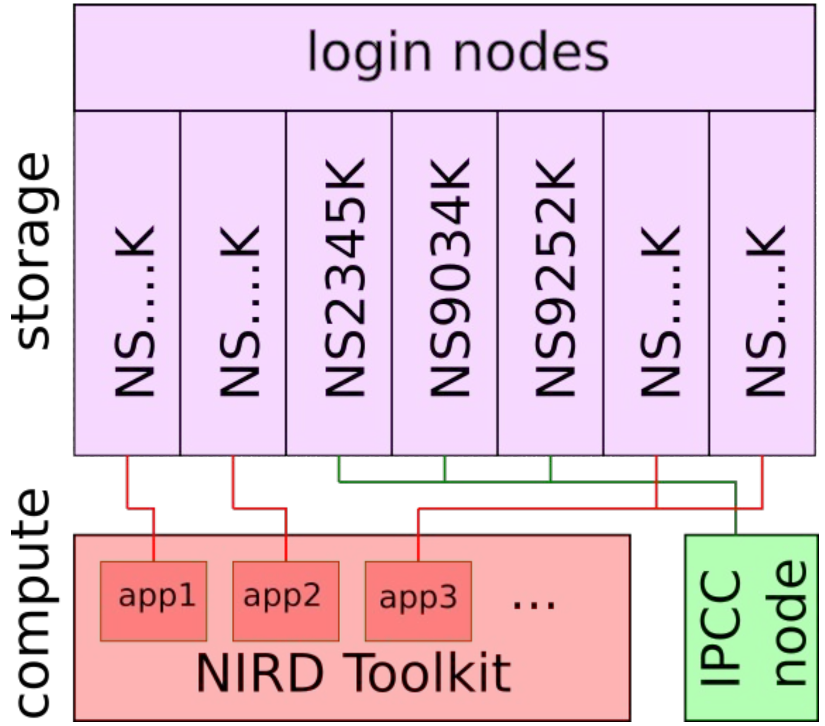
Configure & run ESMValTool on the Nird IPCC node

ESMValTool workshop, 30-31 May 2023

NIRD IPCC node

NIRD is the (Norwegian) National e-Infrastructure for Research Data. It is owned and operated by UNINETT Sigma2.



Work with ESMValTool on Nird

Nird contains complete CMIP5 and CMIP6 datasets for NorESM, and a subset of CMIP5 and CMIP6 datasets for other models. This makes it attractive to run ESMValTool on Nird. Options include:

1. Run ESMValTool on the dedicated ipcc.nird.sigma2.no node
    a. Mainly used for CMORization of NorESM output, can be used for other purposes when idle
    b. Requires specific login permission to access
2. Run ESMValTool through the Nird Toolkit system
    a. Requires access to a Nird Toolkit group, and jupyterlab/jupyterhub application preconfigured with ESMValTool
    b. Access to data across multiple storage volumes is complicated (contact Nird admin)
3. Install ESMValTool in your own storage volume using the conda system
    WARNING: Login nodes are not designed for heavy processing jobs!

ESMValTool on ipcc.nird node

https://noresm-docs.readthedocs.io/en/noresm2/diagnostics/esmvaltool.html#run-esmvaltool-on-nird-service-node

1. Log in on the ipcc node: `ssh -l <username> ipcc.nird.sigma2.no`
   - For workshop, suggest to use personal directory under /scratch/<username>

2. Source conda, do one of the following
   a. Execute source for conda (*every time*)
      `module load Miniconda3`
   b. Add the line above to your .bashrc or .profile file (*one time fix*)

3. Load ESMValTool on Nird (*every login*): `conda activate /diagnostics/esmvaltool/2.8.0`

4. Copy config file and recipe to where you want to execute esmvaltool
   ```
   /projects/NS9560K/users/tomast/esmvaltool/noresmvaltool/esmvaltool/config/config-user.yml
   /projects/NS9560K/users/tomast/esmvaltool/noresmvaltool/esmvaltool/tested_recipes/
   ```
   or
   ```
   git clone https://github.com/NorESMhub/noresmvaltool.git
   ```

Install ESMValTool on your own storage volume

Install release version of ESMValTool

https://docs.esmvaltool.org/en/latest/quickstart/installation.html#mamba-conda-installation

```
mamba create --name esmvaltool esmvaltool
```

Install development version from source

https://docs.esmvaltool.org/en/latest/quickstart/installation.html#install-from-source

```
git clone https://github.com/ESMValGroup/ESMValTool
mamba env create --name esmvaltool --file environment.yml
conda activate esmvaltool
pip install --editable '.[develop]'
```

See also ESMValTool Tutorial: Installation

Test installation

After activating the esmvaltool environment, test the installation by running

```
esmvaltool version
```

Expected output:

```
ESMValCore: 2.8.0
ESMValTool: 2.8.0
```

Get help

Try the following

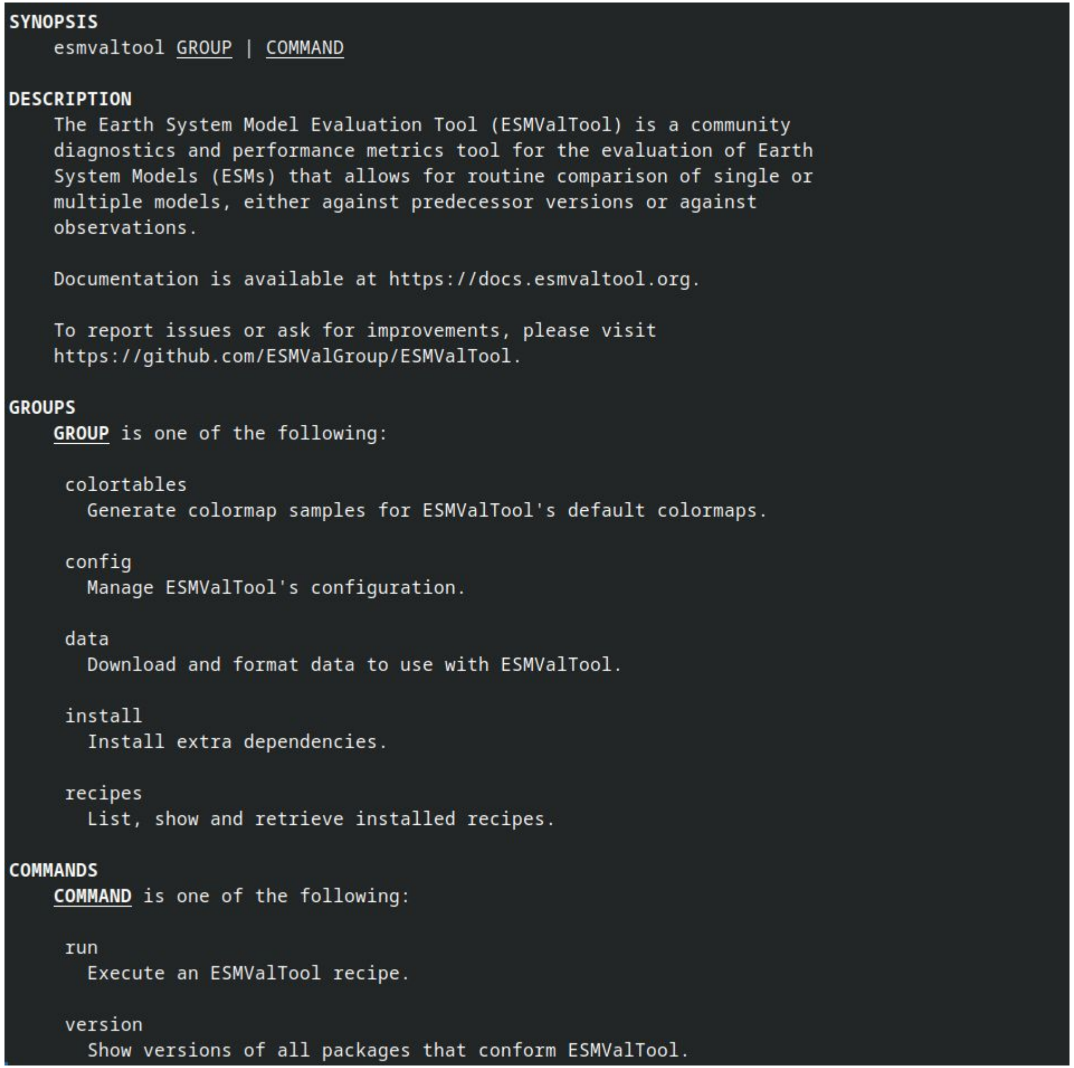
esmvaltool --help

esmvaltool config --help

esmvaltool data --help

esmvaltool recipes --help

esmvaltool run --help

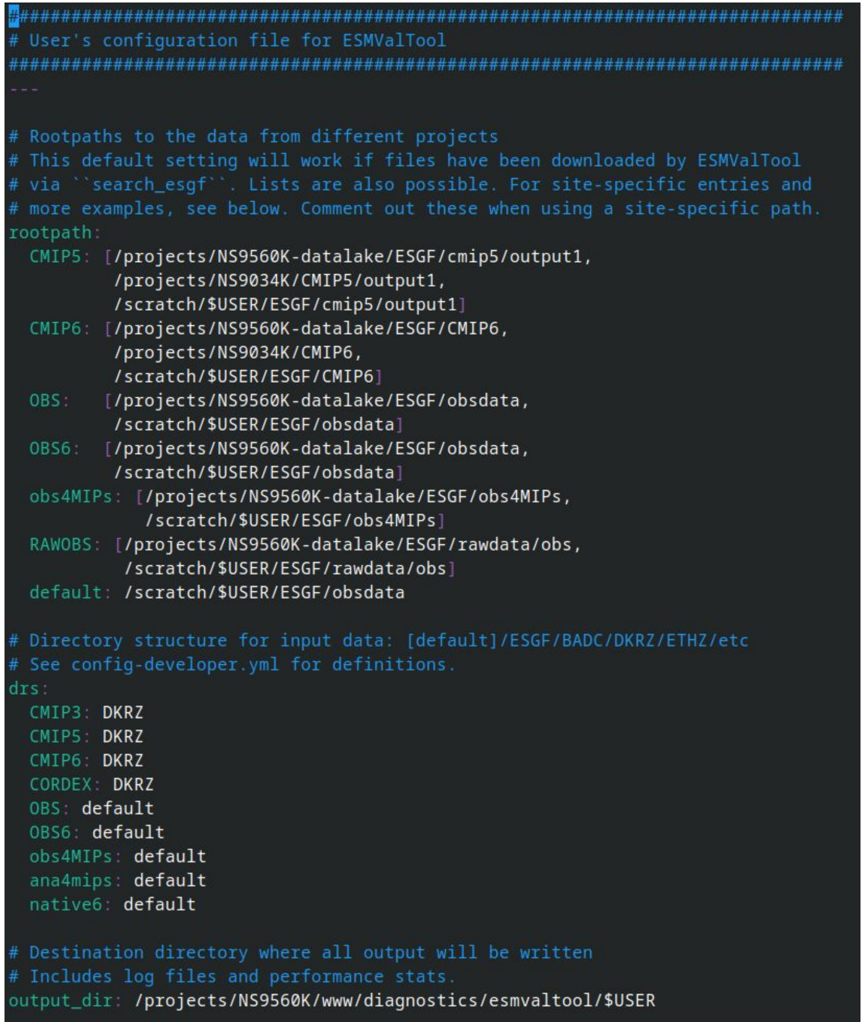


```
SYNOPSIS
    esmvaltool GROUP | COMMAND

DESCRIPTION
    The Earth System Model Evaluation Tool (ESMValTool) is a community
    diagnostics and performance metrics tool for the evaluation of Earth
    System Models (ESMs) that allows for routine comparison of single or
    multiple models, either against predecessor versions or against
    observations.

    Documentation is available at https://docs.esmvaltool.org.

    To report issues or ask for improvements, please visit
    https://github.com/ESMValGroup/ESMValTool.

GROUPS
    GROUP is one of the following:

     colortables
       Generate colormap samples for ESMValTool's default colormaps.

     config
       Manage ESMValTool's configuration.

     data
       Download and format data to use with ESMValTool.

     install
       Install extra dependencies.

     recipes
       List, show and retrieve installed recipes.

COMMANDS
    COMMAND is one of the following:

     run
       Execute an ESMValTool recipe.

     version
       Show versions of all packages that conform ESMValTool.
```

User config file

- Where are input data files located?
- What is the directory structure of the input data?
- Where should output be placed?
- What should ESMValTool consist of?
  - Output file types
  - Should intermediate files be preserved?
  - Level of detail for logs
  - Profile the diagnostics for efficiency (Python only)
- Run tasks in parallel?
- Use custom config-developer file?

See also
    [ESMValTool Tutorial: Configuration]()



```
###############################################################################
# User's configuration file for ESMValTool
###############################################################################
---

# Rootpaths to the data from different projects
# This default setting will work if files have been downloaded by ESMValTool
# via ``search_esgf``. Lists are also possible. For site-specific entries and
# more examples, see below. Comment out these when using a site-specific path.
rootpath:
  CMIP5: [/projects/NS9560K-datalake/ESGF/cmip5/output1,
          /projects/NS9034K/CMIP5/output1,
          /scratch/$USER/ESGF/cmip5/output1]
  CMIP6: [/projects/NS9560K-datalake/ESGF/CMIP6,
          /projects/NS9034K/CMIP6,
          /scratch/$USER/ESGF/CMIP6]
  OBS:   [/projects/NS9560K-datalake/ESGF/obsdata,
          /scratch/$USER/ESGF/obsdata]
  OBS6:  [/projects/NS9560K-datalake/ESGF/obsdata,
          /scratch/$USER/ESGF/obsdata]
  obs4MIPs: [/projects/NS9560K-datalake/ESGF/obs4MIPs,
             /scratch/$USER/ESGF/obs4MIPs]
  RAWOBS: [/projects/NS9560K-datalake/ESGF/rawdata/obs,
           /scratch/$USER/ESGF/rawdata/obs]
  default: /scratch/$USER/ESGF/obsdata

# Directory structure for input data: [default]/ESGF/BADC/DKRZ/ETHZ/etc
# See config-developer.yml for definitions.
drs:
  CMIP3: DKRZ
  CMIP5: DKRZ
  CMIP6: DKRZ
  CORDEX: DKRZ
  OBS: default
  OBS6: default
  obs4MIPs: default
  ana4mips: default
  native6: default

# Destination directory where all output will be written
# Includes log files and performance stats.
output_dir: /projects/NS9560K/www/diagnostics/esmvaltool/$USER
```

User config file: CMIP and obsdata on Nird

CMIP data stored using the **DKRZ** directory structure.

NS9034K : NorESM archive for CMIP5 and CMIP6

```
/projects/NS9034K/CMIP5/{output1 , output2}/
/projects/NS9034K/CMIP6/
```

NS9560K-datalake : (mount from /nird/datalake/NS9560K/
CMIP products from various ESMs

```
/projects/NS9560K-datalake/ESGF/cmip5/{output , output1}/
/projects/NS9560K-datalake/ESGF/CMIP6/
```

Observational data (for some recipes) are stored in

```
/projects/NS9560K-datalake/obsdata/{Tier1 , Tier2 , Tier3}/
```

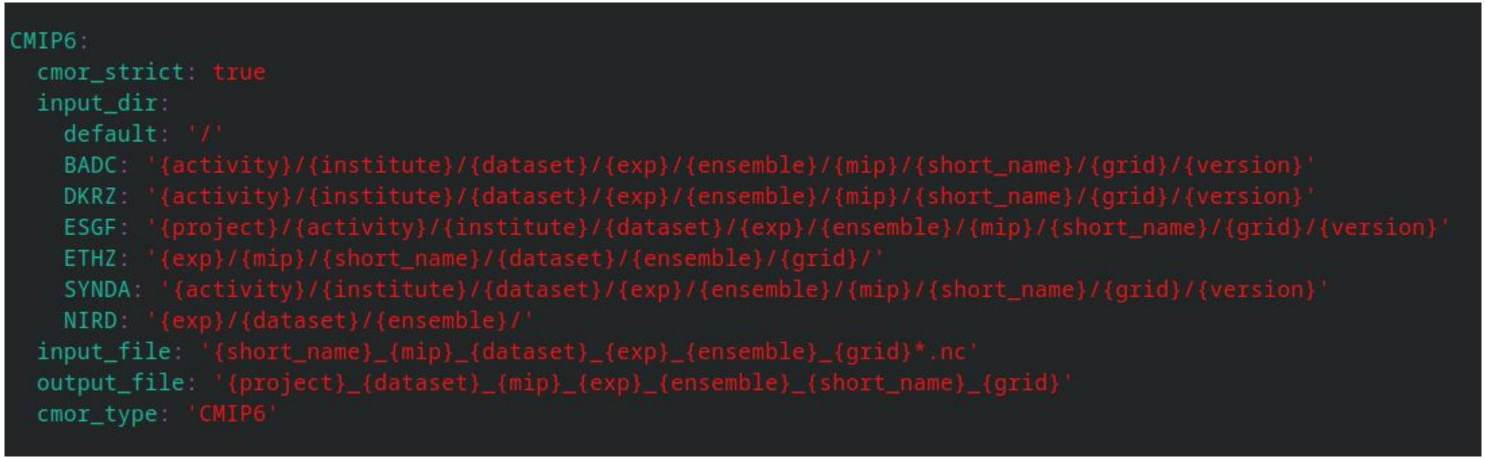
User config file: options, [...]=default

- output_dir: /projects/NS9560K/www/diagnostics/esmvaltool/$USER
  - Available for access from web browser: http://ns9560k.web.sigma2.no/diagnostics/esmvaltool/
- auxiliary_data_dir: /projects/NS9560K-datalake/ESGF/auxiliary_data
  - e.g. mask files and shapefiles for data selection
- search_esgf: { [never], when_missing, always }
  - Select if missing data should be downloaded from ESGF to download_dir
- download_dir: /scratch/$USER/ESGF
- max_parallel_tasks: { [null], 1, 2, 3, … } # null = use all available CPUs
- log_level: { debug, [info], warning, error }
- exit_on_warning: { true, [false] }
- output_file_type: { [png], pdf, ps, eps, epsi }
- remove_preproc_dir: { [true], false }
- compress_netcdf: { true, [false] }
- save_intermediate_cubes: { true, [false] }
- config_developer_file: { [null], custom “config-developer.yml”
- profile_diagnostic: { true, [false] } # profiler for python diagnostics

Developer’s configuration file

config-developer.yml :

Project- and machine-dependent directory and file name definitions of the input and output data.



```
CMIP6:
  cmor_strict: true
  input_dir:
    default: '/'
    BADC: '{activity}/{institute}/{dataset}/{exp}/{ensemble}/{mip}/{short_name}/{grid}/{version}'
    DKRZ: '{activity}/{institute}/{dataset}/{exp}/{ensemble}/{mip}/{short_name}/{grid}/{version}'
    ESGF: '{project}/{activity}/{institute}/{dataset}/{exp}/{ensemble}/{mip}/{short_name}/{grid}/{version}'
    ETHZ: '{exp}/{mip}/{short_name}/{dataset}/{ensemble}/{grid}/'
    SYNDA: '{activity}/{institute}/{dataset}/{exp}/{ensemble}/{mip}/{short_name}/{grid}/{version}'
    NIRD: '{exp}/{dataset}/{ensemble}/'
  input_file: '{short_name}_{mip}_{dataset}_{exp}_{ensemble}_{grid}*.nc'
  output_file: '{project}_{dataset}_{mip}_{exp}_{ensemble}_{short_name}_{grid}'
  cmor_type: 'CMIP6'
```

Available standard recipes

Standard recipes bundled with the esmvaltool installation: list and copy

```
esmvaltool recipes list
esmvaltool recipes get <standard_recipe.yml>
```

List of standard recipes tested on NIRD with ESMValTool 2.8.0:

https://docs.google.com/spreadsheets/d/1V7epqQgzPZXdLk_VBBIRH0Zq6yg_6gbRI-kgQ9UgDHE/edit?usp=sharing

(location may change to a more permanent site after workshop)

ESMValTool run files

To run an ESMValTool diagnostics, it is necessary (and usually sufficient) to provide the config and recipe files

- config-user.yml : General settings for ESMValTool
- recipe_python.yml : How to execute the diagnostics

A diagnostics run is executed from the command line

```
    esmvaltool   run   --config_file=config-user.yml
examples/recipe_python.yml
```

Optional: ESMValTool looks for a default user config file in

```
    $HOME/.esmvaltool/config-user.yml
```

Copy your config file to this location to avoid using the `--config_file` flag for every run.

Run a standard recipe

List available recipes:

```
esmvaltool recipes list
```

Show content of a recipe:

```
esmvaltool recipes show examples/recipe_python.yml
```

Copy a recipe to local directory (optional for standard recipes):

```
esmvaltool recipes get examples/recipe_python.yml
```

Run esmvaltool with recipe:

```
esmvaltool run --config_file=config-user.yml ./recipe_python.yml
```

See output: http://ns9560k.web.sigma2.no/diagnostics/esmvaltool/

ESMValTool output

An ESMValTool run creates a directory with the following output subdirectories

- run/ : Run logs, copy of recipe, summary of resource usage
- preproc/ : NetCDF output from preprocessing runs (optional)
  - This is removed when config option remove_preproc_dir: true is set
- plots/ : Graphical output
- work/ : Any output that is not a plot, e.g. NetCDF files after diagnostic run

Most standard recipes also create an index.html file in the root directory that shows a summary and usually the graphical output of the recipe.

ESMValTool output: logs

Main logs:

- run/main_log.txt - short log
- run/main_log_debug.txt - full debug log
- run/<diag_name>/log.txt - log for each diagnostic task

See also ESMValTool Tutorial: Running your first recipe

Problems when running esmvaltool recipes

NIRD does not contain a full set of CMIP5/6 and observational datasets.

Running an **untested** standard recipe will **probably** fail on Nird at the first attempt

1. Missing observational datasets
    a. Solution1: Download from ESGF if available, using `--search_esgf=when_missing`
    b. Solution2: Add dataset, CMORize if needed (contact esmvaltool-on-nird discussion group)
2. Missing model datasets
    a. Solution1: Try to run with `--skip_nonexistent=True` flag
    b. Solution1: Download from ESGF if available, using `--search_esgf=when_missing`
    c. Solution2: Replace missing dataset with something available (e.g. NorESM data)
3. Missing auxiliary data, e.g. shapefile or mask
    a. Solution1: Copy from `/project/NS9560K-datalake/ESGF/auxiliary_data/`
    b. Solution2: Add auxiliary data in directory according to your config file
4. Missing support for diagnostic script (mainly problem with Julia)

Resources: Further reading and getting help

NorESM documentation in ReadTheDocs: ESMValTool diagnostics

From GitHub: NorESMhub/noresmvaltool

GitHub NorESMhub team: esmvaltool-on-nird discussion group