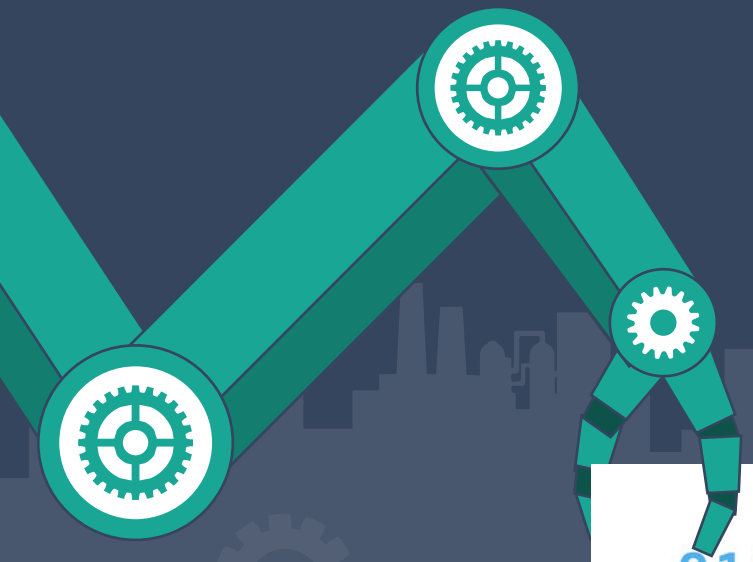
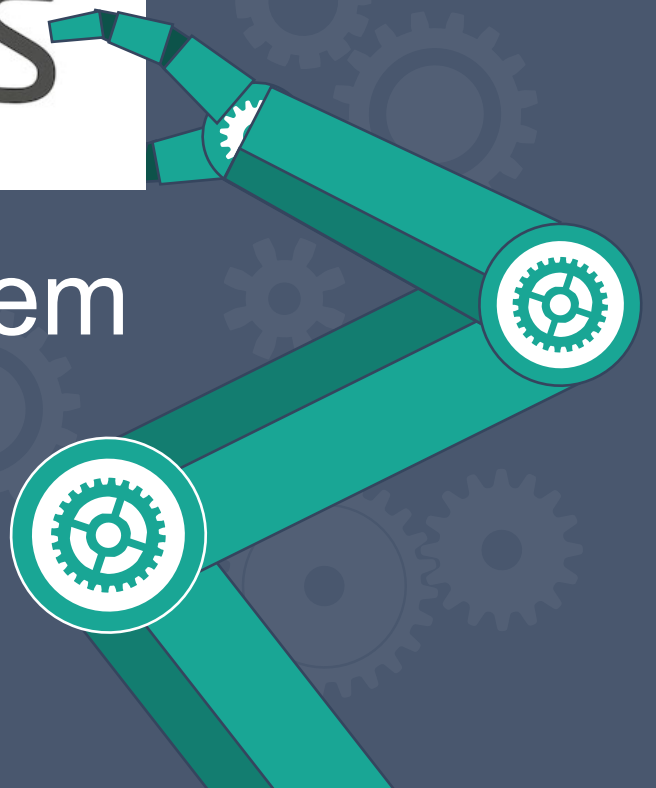


01000101
01010011
01001101

# ESM-Tools



# Infrastructure for Earth System Modelling

ESM User Workshop May 2019

# Agenda

01 **What are the ESM-Tools?**

A unified infrastructure for ESM modelling

02 **Who benefits?**

The Tools are designed to simplify the work of modelers, model developers, software supporters and HPC admins.

03 **How are the ESM-Tools developed?**

Developed within the ESM project, the Tools have more than 30 authors from 6 institutes. Main development is done at AWI.

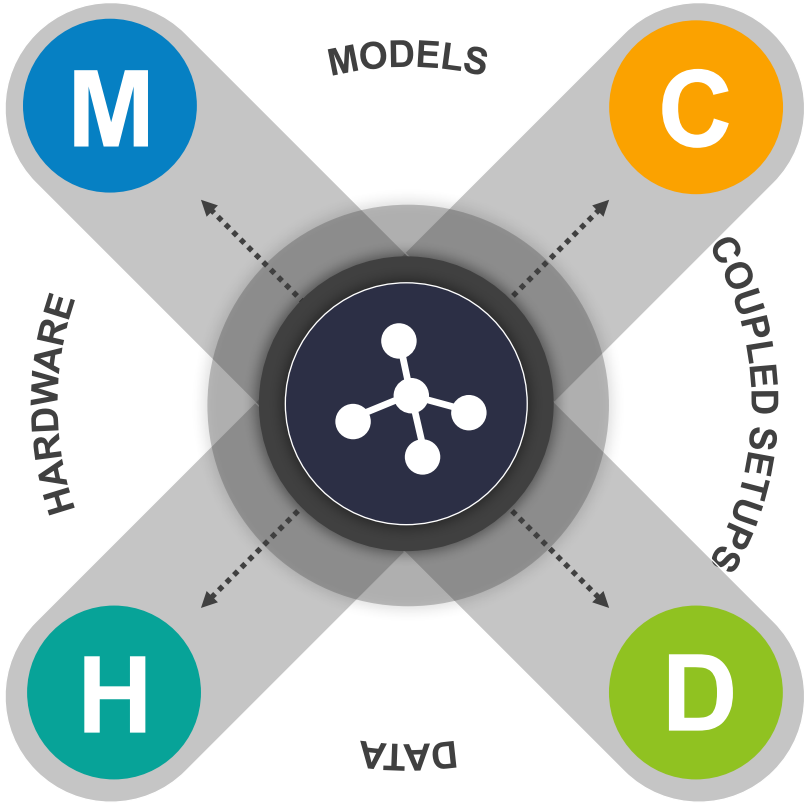
04 **How can I get the ESM-Tools?**

ESM-Tools can be obtained for free, it is open source under GPL.

# What are the ESM-Tools?

**Models**

The Tools provide an easy and standardized way to obtain, configure and compile model components. As of May 2019, we support 10 independent components, including 4 ocean, 2 atmosphere, 1 ice sheet, 1 BGC, 1 GIA model and a coupler, all organized under version control.




**Coupled Setups**

Out of the mentioned models, a variety of coupled systems can be combined. A graphical interface helps choosing the wanted components. Among the supported combinations are MPIESM, AWICM (1 and 2), FESOM-OIFS and FOCI-OIFS.

**Hardware**

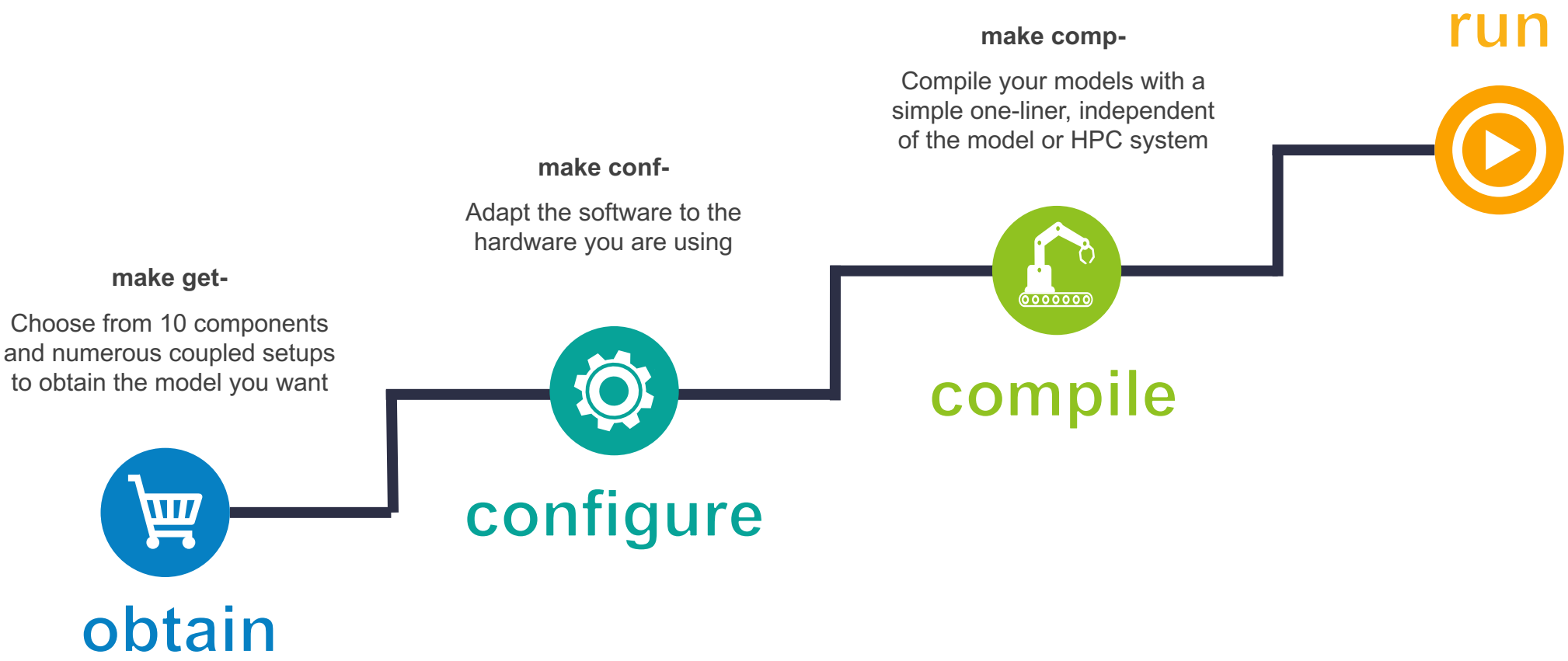
Currently the Tools are running on 6 HPC systems, holding the machine specifications in a central place that can be used by the models during compile and run time, leaving more time for the user to focus on scientific questions.

**Data**

The life cycle of model data has become increasingly important. The tools try to help – by referring to standardized input data pools, managing model output and restarts, and functionality for tar-balling and archiving data. CMOR support is planned.

# Obtaining models

*esm-master*



# Running Simulations

## *esm-runscripts*

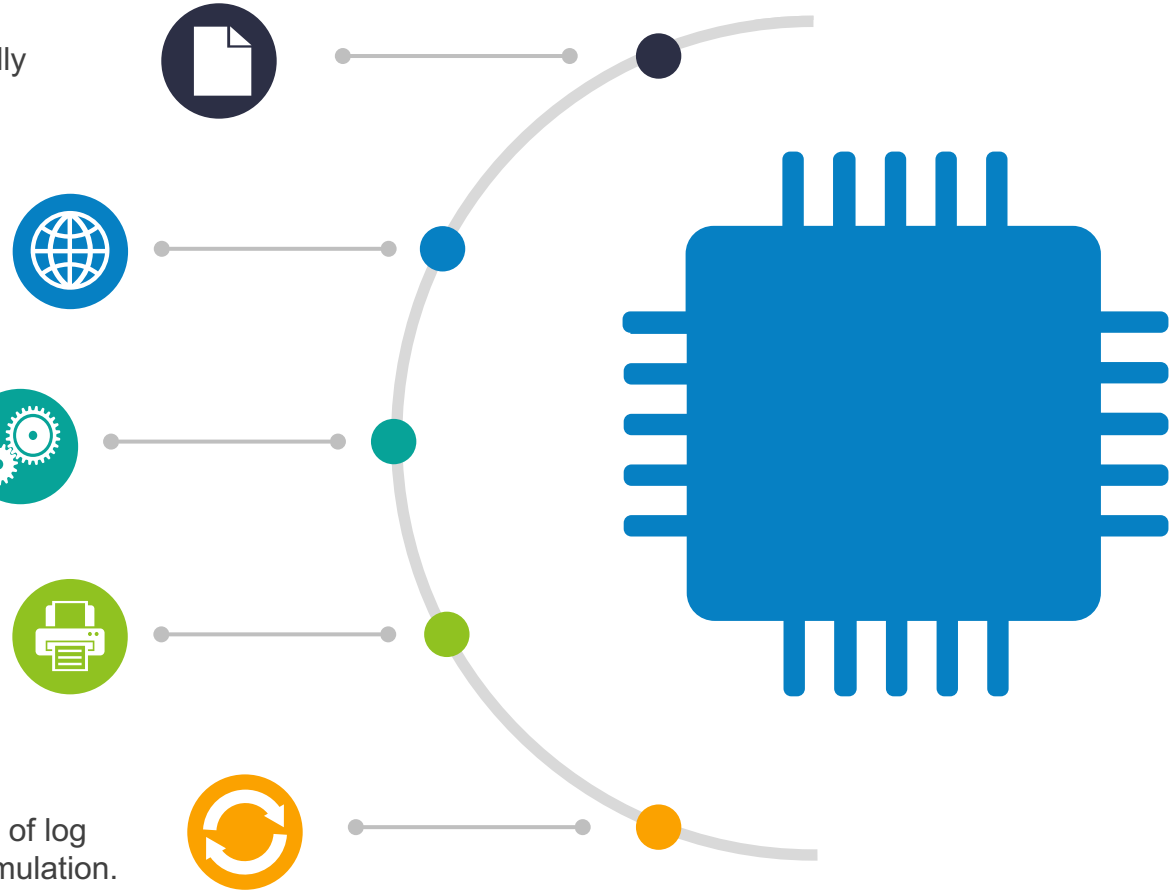
**Small runscripts**
Focus on your simulation, defaults are pre-defined. Typically around 40 lines, similar for all models.

**Input / Forcing Data**
The Tools organize the data, recognizing standard data pools on the machines.

**Runtime Functionality**
A lot of nice stuff – restarts, monitoring, iterative coupling,

**Output Data**
Model Output is automatically sorted and postprocessed, even tarred and archived.

**Reproducibility / Documentation**
Simulation folders contain (almost) everything, and a ton of log files provide all the information to re-run or continue a simulation.



# Who benefits from the Tools?

**1st**

**System Admins**

Standardized Compile and Runtime Environments means fewer needed software packages. Deploy optimal machine settings or new modules easily.

**2nd**

**Model developers**

Organize your developments, deploy them on different machines. Co-work with other institutes on the same code.

**3rd**

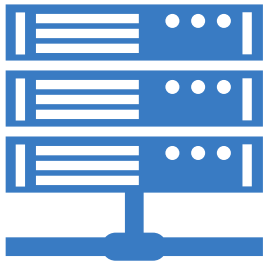
**Modellers**

Run your simulations in an easy and unified way, independent of the model and hardware. Have lots of functions with a few lines of runscript.

**4th**

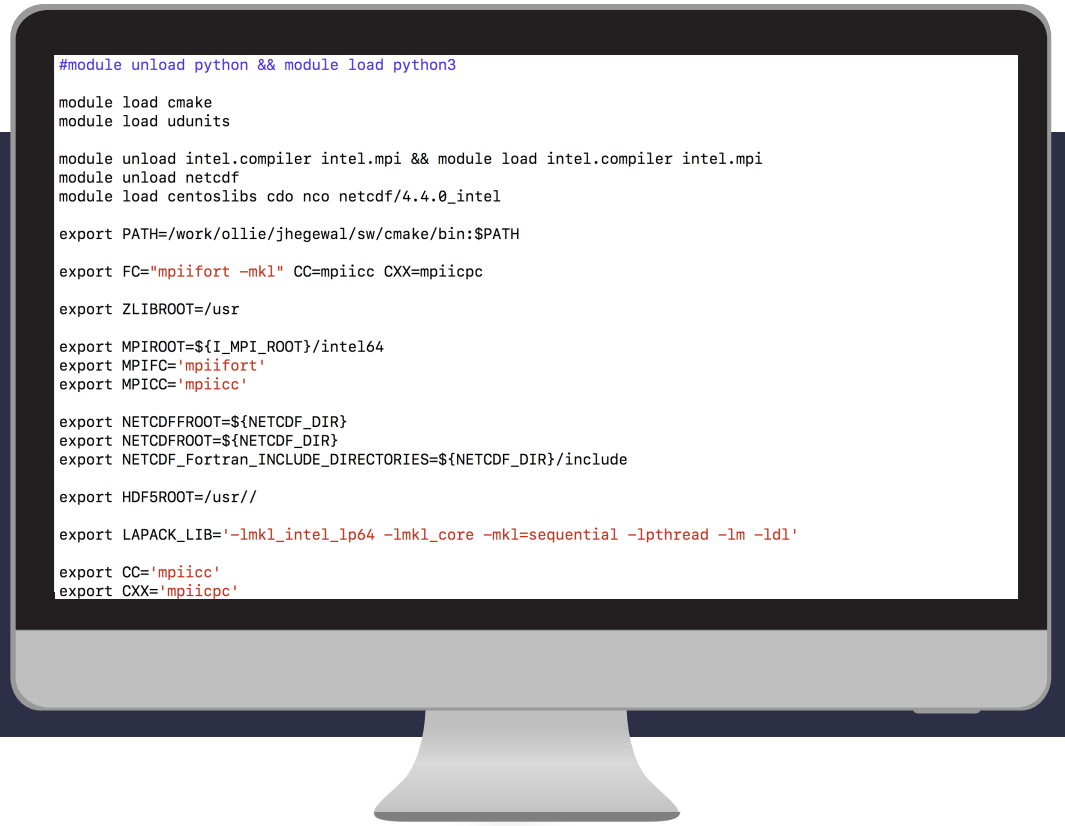
**Model supporters**

Solve problems once, not over and over again. Deploy bugfixes / new hardware configurations quickly to all users. Same experiment layout also means less context switching.



# System Admins

- A group of users all use the same configuration file – solve problems once, and deploy the solution quickly.
- Provide an “optimal” standard configuration to your users.
- Get less requests for additional tools / packages or incompatible combinations.
- Get a reduced ’.sad’ – runscript only containing the information needed by the batch system. (WIP)



```
#module unload python && module load python3

module load cmake
module load udunits

module unload intel.compiler intel.mpi && module load intel.compiler intel.mpi
module unload netcdf
module load centoslibs cdo nco netcdf/4.4.0_intel

export PATH=/work/ollie/jhegewal/sw/cmake/bin:$PATH

export FC="mpiifort -mkl" CC=mpiicc CXX=mpiicpc

export ZLIBROOT=/usr

export MPIROOT=${I_MPI_ROOT}/intel64
export MPIFC='mpiifort'
export MPICC='mpiicc'

export NETCDFFROOT=${NETCDF_DIR}
export NETCDFROOT=${NETCDF_DIR}
export NETCDF_Fortran_INCLUDE_DIRECTORIES=${NETCDF_DIR}/include

export HDF5ROOT=/usr//

export LAPACK_LIB='-lmkl_intel_lp64 -lmkl_core -mkl=sequential -lpthread -lm -ldl'

export CC='mpiicc'
export CXX='mpiicpc'
```

# Model Developers

```
Available compile options:
                comp-amip
                comp-awicm-CMIP6
                comp-awicm-1.1
                comp-awicm-1.0
                comp-awicm-2.0
                comp-awicm-3.0
                comp-awicm-test
                comp-echam-6.3.04p1
                comp-echam-6.3.02p4
                comp-fesom-1.4
                comp-fesom-1.4-recom
                comp-fesom-1.4-recom-modular
                comp-fesom-2.0
                comp-fesom-2.0-mesh-part
                comp-fesom-2.0-recom-modular
                comp-mpiesm-1.2.01p1
                comp-mpiesm-1.2.01
                comp-mpiesm-1.2.00p4
                comp-nemo-3.6
                comp-oasis3-mct
                comp-oifs-40r1
                comp-oifsamip
                comp-recom-lib
                comp-rnfmap
                comp-xios-2.0_r982
```

## Concentrate more on your model, and less on the technical side

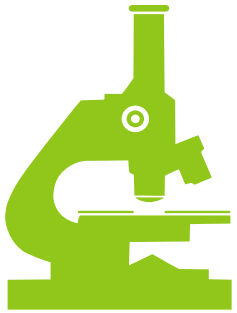
The ESM-Tools provide standard configurations for a number of HPC systems. Chances are good that you can compile and run your development on these out of the box.

## Get your developments to the users

New standard versions of your codes can be distributed with esm-master.

## Runscripts with a ton of functions for a new model - without coding

All you need in a runscript is already there. You can just go and use it.



# Modellers

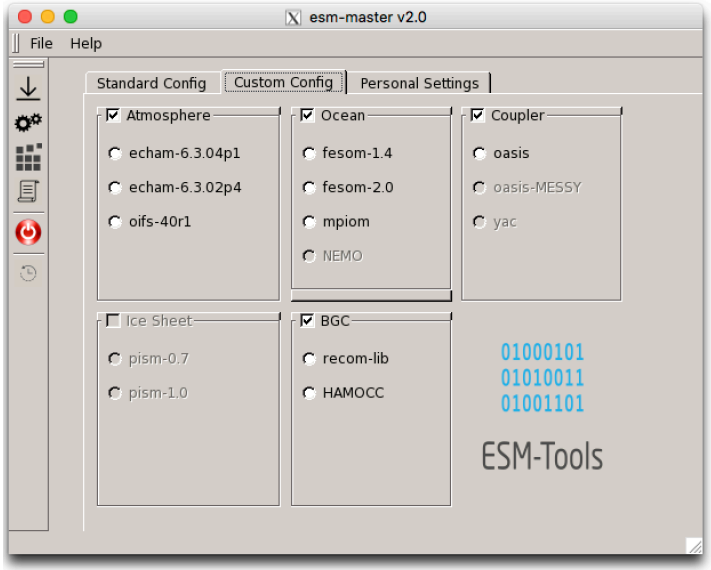
**Start running simulations quickly. With a steep learning curve.**

New colleague, about to work with an unknown model? Maybe even new to programming, shell scripting, HPC?

Changing to a new model (system), maybe for intercomparison, or exchanging one implementation of a model with a different one?

Changing to a different HPC system?

Getting lost in too many simulations, and needing an automatic way to ”document” them?




**Easy access to a huge variety of model components and coupled setups**

**Short (but do-it-all) runscripts, almost independent of the model and hardware you use**

```
export FUNCTION_PATH=${WORK}/esm-master/esm-runscripts/functions/all
export FPATH=${FUNCTION_PATH}:$FPATH

machine_name="ollie"
setup_name="fesom_standalone"
#check=1

compute_time="05:00:00"
#############################################################################
#Xsrun  I know what I am doing

INITIAL_DATE_fesom_standalone=2008-01-01        # Initial exp. date
FINAL_DATE_fesom_standalone=2010-01-01          # Final date of the experiment

RES_fesom=CORE2


MODEL_DIR_fesom_standalone=${WORK}/esm-master/fesom_standalone/

BIN_DIR_fesom=${MODEL_DIR_fesom_standalone}/fesom_cpl/
EXE_fesom=fesom.x

BASE_DIR=${WORK}/esm-experiments/

POOL_DIR_fesom_standalone=/work/ollie/pool/FESOM/

MESH_DIR_fesom=/work/ollie/pool/FESOM/meshes_default/core/

NYEAR_fesom_standalone=1          # Number of years per run

#############################################################################
```

# Model Support

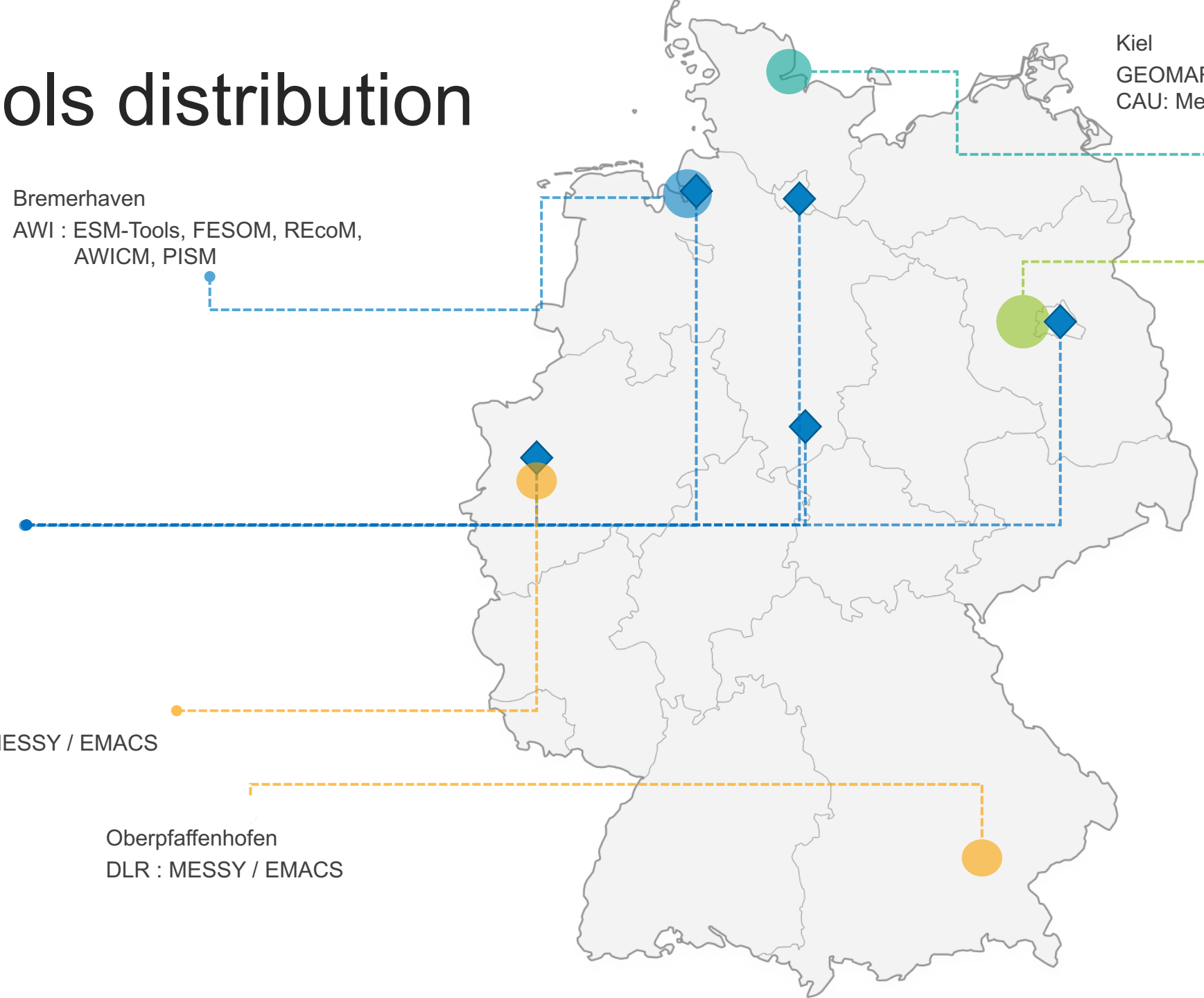
**One environment configuration**
If anything goes wrong with the environment, you know where to look.

**One runscript**
Effectively people can use the same runscript for a variety of models, so no need to work through a new one each time. Plus ESM-Tools support.

**Automatic testing**
You can know about problems before your users do, and also where it might come from.

**Deploy solutions**
Once a problem is fixed, the solution can be picked up by all users directly.

# ESM-Tools distribution



Kiel
GEOMAR : NEMO / FOCI-OIFS
CAU: Metos3D

Bremerhaven
AWI : ESM-Tools, FESOM, REcoM,
AWICM, PISM

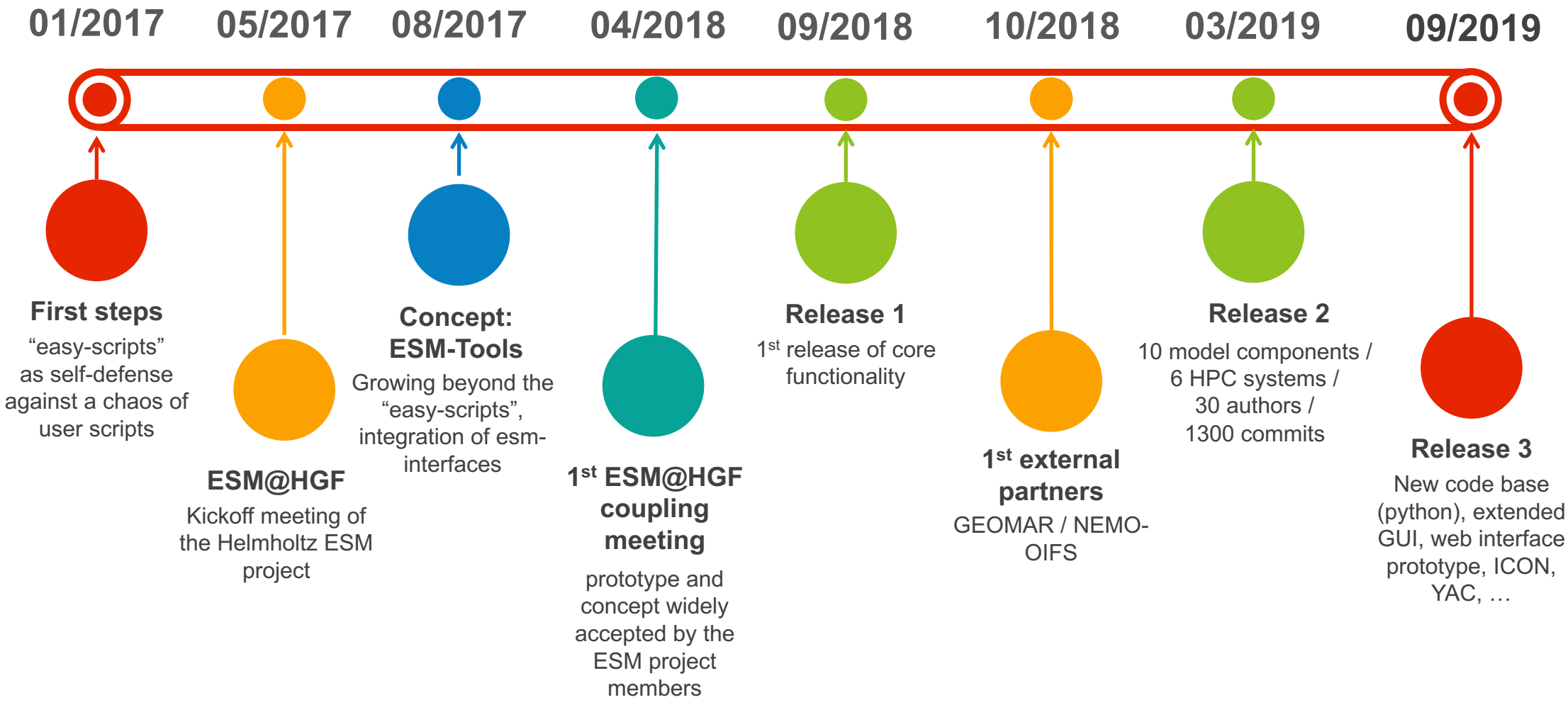
Potsdam
AWI : Echam / ICON
GFZ: VILMA

Supported machines:
AWI: ollie
JSC: juwels, jureca
HLRN: blogin, glogin
DKRZ: mistral

Bonn
Uni Bonn : MESSY / EMACS

Oberpfaffenhofen
DLR : MESSY / EMACS

# Timeline


01/2017
05/2017
08/2017
04/2018
09/2018
10/2018
03/2019
09/2019
First steps
“easy-scripts” as self-defense against a chaos of user scripts
ESM@HGF
Kickoff meeting of the Helmholtz ESM project
Concept: ESM-Tools
Growing beyond the “easy-scripts”, integration of esm-interfaces
1st ESM@HGF coupling meeting
prototype and concept widely accepted by the ESM project members
Release 1
1st release of core functionality
1st external partners
GEOMAR / NEMO-OIFS
Release 2
10 model components / 6 HPC systems / 30 authors / 1300 commits
Release 3
New code base (python), extended GUI, web interface prototype, ICON, YAC, …

# Development strategy

**ESM-Tools TEAM@AWI**

Nadine Wieters, Dirk Barbi, Luisa Cristini (Paul Gierz)
New colleague (position is open for applications) starting in summer ‘19.


**Model Developers**

All component models are available in VCS repositories, most of them in modular_esm. Frequent communication ensures that the versions are up-to-date. Regular tagging.

**Model Users**

Reporting problems using the issue tracking system of gitlab helps us to provide quick solutions.

**System Admins and Supporters**

Contribute changes in the machine settings and bugfixes back to the Tools to deploy them.

# How can I get the ESM-Tools?

https://gitlab.dkrz.de/esm-tools

(https://gitlab.awi.de/esm-tools)

**Contact us:**

info@esm-tools.net

0471/4831-1561 /

www.esm-tools.net

twitter.com/ToolsESM

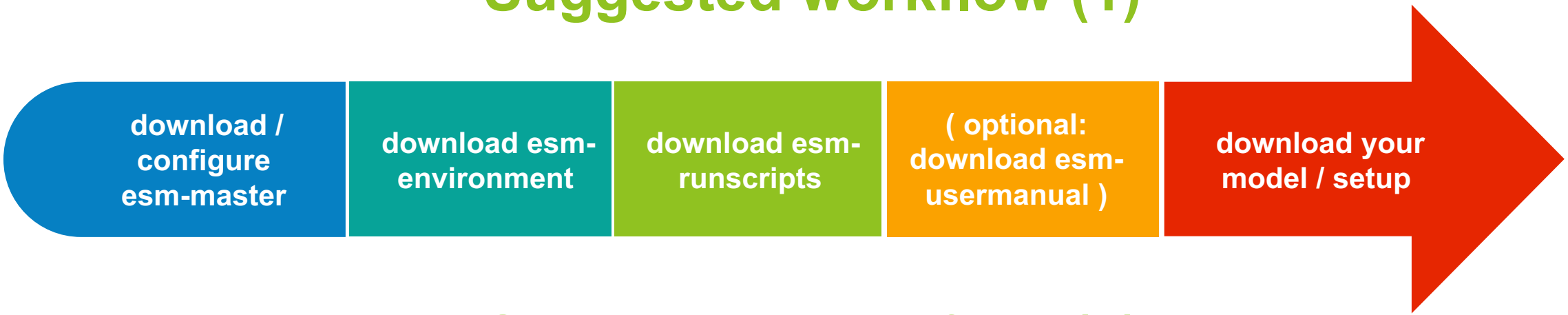
01000101
01010011
01001101

# ESM-Tools

## Get the Tools

# Getting the esm-tools

## Suggested workflow (1)




## Suggested workflow (2)


01000101
01010011
01001101 ESM-Tools

# Download esm-master

**git clone https://gitlab.dkrz.de/esm-tools/esm-master.git**

**(git clone https://gitlab.awi.de/esm-tools/esm-master.git)**

# Configure esm-master

**In the esm-master folder, type**

***make***

**and answer the questions that pop up. The configuration is written to a file called**

***esm-master.conf***

**which can be edited by hand. If it is messed up, remove it, or run**

***make conf-esm-master***

# Download esm-environment

**In the esm-master folder, type:**

**make get-esm-environment**

**(Same as:**
**git clone https://gitlab.dkrz.de/esm-tools/esm-environment.git )**

# Download esm-runscripts

**In the esm-master folder, type:**

```
make get-esm-runscripts
```

**(Same as:**

```
git clone https://gitlab.dkrz.de/esm-tools/esm-runscripts.git
cd esm-runscripts/functions
./set_links
cd ../..        )
```

# Download esm-usermanual

**In the esm-master folder, type:**

**make get-esm-usermanual**

**(Same as:**

**git clone https://gitlab.dkrz.de/esm-tools/esm-usermanual.git )**

**Or:**

**Download the pdf from www.esm-tools.net**

01000101
01010011
01001101 ESM-Tools

# Shortcut 1:

**After downloading and configuring esm-master, type:**

**make get-esm-tools**

**(Same as:**

**make get-esm-environment**

**make get-esm-runscripts**

**make get-esm-usermanual )**

# Shortcut 2:

**Download and run the esm-tools installer:**

```
git clone https://gitlab.dkrz.de/esm-tools/esm-tools.git
cd esm-tools
./install.sh
```

**In this case, esm-master still needs to be configured!**

# Three ways to get the Tools

**01**

```
git clone https://gitlab.dkrz.de/esm-tools.de/esm-master.git
cd esm-master; make
make get-esm-environment
make get-esm-runscripts
make get-esm-usermanual
```

**02**

```
git clone https://gitlab.dkrz.de/esm-tools.de/esm-master.git
cd esm-master; make
make get-esm-tools
```

**03**

```
git clone https://gitlab.dkrz.de/esm-tools.de/esm-tools.git
cd esm-tools
./install.sh
```

# How can I get Information / Documentation?

**www.esm-tools.net**

https://gitlab.dkrz.de/esm-tools

**Contact us:**
info@esm-tools.net
0471/4831-1561 /
www.esm-tools.net
twitter.com/ToolsESM