**PCR-GLOBWB GUIDE**

**Author: Valentina Bonato**

**Year: 2024-2025**

**CONTENTS**

**1. Setup environments**

**1.1 Login to the supercomputer snellius: create a SSH connection**

**1.2 Load all software for PCR-GLOBWB**.

**2. Introduction to screen, useful for multi-tasking in your terminal**

**3. Run PCR-GLOBWB at the global extent – 30arcmin**

* 1. **Download PCR-GLOBWB\_model**

**3.2 Adjusting configuration files .ini**

**3.3 Run PCR-GLOBWB using sbatch**

**3.4 Viewing your results**

**4. How snellius works**

* 1. **Entering a computational node**

**4.2 Leaving snellius**

1. **Setup environments**

**1.1 Create a SSH connection**

Open xquartz (if you don’t have it download it here <https://www.xquartz.org>) to create a SSH connection to the supercomputer snellius using the xquarz terminal. If you are using a different supercomputer, follow their instructions.

# on your xquartz terminal on your laptop, make a SSH connection to snellius, please add the option/argument "-XY"."-X" enables X11 forwarding, which allows graphical applications from the remote SSH server to be displayed on your local machine. "-Y" specifies trusted X11 forwarding, which is a more secure variant of X11 forwarding. It's generally used when you trust the remote host and its users.

# ssh -XY <your\_snellius\_username>@snellius.surf.nl

$ ssh -XY vbonato@snellius.surf.nl

([vbonato@snellius.surf.nl](mailto:vbonato@snellius.surf.nl)) Password:

# Insert password, you won’t see it while typing, just send it

If the connection is successful, you should arrive on the snellius terminal. Please see the screenshot below for an illustration (it is not going to be exactly the same; but the machine name “snellius” and your user name should appear):

A screenshot of a computer

Description automatically generated



**1.2 Load all software for PCR-GLOBWB**.

Next, load all software that is required for running and working with PCR-GLOBWB on snellius. For this, you can just use the existing bash script that is stored in the file “/home/sutan101/load\_all\_defaultcd.sh”. This will load a conda/python environment that has all modules required for PCR-GLOBWB, such as pcraster, netcdf4, ncview and cdo.

PS: You can also setup (and load) your own PCR-GLOBWB conda environment by following the steps 1-3 of <https://github.com/UU-Hydro/PCR-GLOBWB_model#how-to-install>.

# After opening snellius, make sure that you are now on your home directory using the command cd or cd $HOME

$ cd $HOME

# print your current working directory

$ pwd

/home/vbonato

# copy /home/edwin/load\_all\_default.sh to your home directory and then load it. This script will load a conda environment with all the modules needed (PCRaster, netcdf4…)

$ cp /home/edwin/load\_all\_default.sh $HOME

# list the content of my home (to check if the copy is successful)

$ ls -l

total 10

**-rw-r----- 1 vbonato vbonato 4109 Mar 15 15:00 load\_all\_default.sh**

drwxr-x--- 12 vbonato vbonato 4096 Mar 13 21:20 PCR-GLOBWB\_model

-rw-r----- 1 vbonato vbonato 1142 Mar 14 13:47 sbatch\_example.sh

-rw-r----- 1 vbonato vbonato 1288 Mar 14 13:48 slurm-5537673.out

# Load the file “load\_all\_default.sh”

$ . load\_all\_default.sh

If the loading is successful, you should see something similar appearing on your screen

A screenshot of a computer

Description automatically generated

# you can use “geany” (text editor) to check the content of “load\_all\_default.sh”

$ geany load\_all\_default.sh

To this point, we have loaded all the software required. To test them, please type the following commands.

# pcraster, aguila, gdal, ncview are some of the ones we will be using.

$ pcrcalc

$ ncview

# ncview <netcdf\_file> is the command to view a netcdf file

$ ncview https://opendap.4tu.nl/thredds/dodsC/data2/pcrglobwb/version\_2019\_11\_beta/pcrglobwb2\_input/global\_30min/meteo/forcing/daily\_temperature\_cru\_era-interim\_1979\_to\_2010.nc

$ python

$ exit()

1. **Introduction to screen**

Using a screen allows you to:

1. Use multiple shell windows from a single SSH session (analogy: opening several tabs in your internet browser).
2. Keep a shell active even through network disruptions.
3. Disconnect and re-connect to a shell sessions from multiple locations.
4. Run a long running process without maintaining an active shell session.

To start screen:

$ screen

To detach a screen

$ screen -d

To resume a screen

$ screen -r

You may have several screen sessions. To list all screen sessions that are currently active

$ screen –ls

There are screens on:

18666.pts-20.int1 (Detached)

734.pts-47.int1 (Attached)

2 Sockets in /var/run/screen/S-vbonato.

To continue one of them: screen –r <screen\_session\_name>

$ screen –r 18666.pts-20.int1

To delete all the screens:

$ screen -y

Within a screen session, you can also make multiple shell tabs/windows (RECOMMENDED).

Some shortcuts:

* 1. “ Ctrl-a ” ‘ c ’  creating a new window/tab

(at the same time, press “Ctrl” and “a”; then release both keys; then press “c”)

* 1. “ Ctrl-a ” ‘ n ’  go to the next window
  2. “ Ctrl-a ” ‘ p ’  go to the previous window
  3. “ Ctrl-a ” ‘ ” ’  show window list
  4. “ Ctrl-a ” ‘ A ’  provide a title for each window
  5. “ Ctrl-a ” ‘ d ’  detaching from screen
  6. “ Ctrl-a ” ‘ ? ‘  help

For more and better understanding, please see: <https://www.rackaid.com/blog/linux-screen-tutorial-and-how-to/>

Other links:

<http://aperiodic.net/screen/quick_reference>

<http://neophob.com/2007/04/gnu-screen-cheat-sheet/> (or google-ing: screen cheat sheet)

Problem with X11 $DISPLAY: If you reattach your screen, you may have to synchronize the variables $DISPLAY in your screen windows with the one in your main shell.

Before you reattach/resume your screen, check your current $DISPLAY variable in your main shell.

# still in your main shell

$ echo $DISPLAY

localhost:10.0

Resume your screen and check the current $DISPLAY in your screen window.

# still in your main shell

$ screen –r

# inside your screen session

$ echo $DISPLAY

localhost:10.0

If you try to visualize something, you may get an error message like this.

# inside your screen session

$ aguila

aguila: cannot connect to X server localhost:10.0

Here, you have to synchronize the $DISPLAY variable in your screen window to the $DISPLAY variable in your main shell.

# inside your screen session

$ export DISPLAY=localhost:11.0

Now, you should be able to visualize something (e.g. try aguila again).

1. **Run PCR-GLOBWB at the global extent –30arcmin**

**3.1 Download PCR-GLOBWB\_model**

On your computational machine, clone/download the PCR-GLOBWB\_model from the github website: <https://github.com/UU-Hydro/PCR-GLOBWB_model.git>.

# for example, I want to download it on my home directory

$ cd

$ pwd

/home/vbonato

# clone/download from the github repo

$ git clone <https://github.com/UU-Hydro/PCR-GLOBWB_model.git>

Cloning into 'PCR-GLOBWB\_model'...

remote: Enumerating objects: 32729, done.

remote: Counting objects: 100% (11376/11376), done.

remote: Compressing objects: 100% (3007/3007), done.

remote: Total 32729 (delta 8380), reused 11222 (delta 8249), pack-reused 21353

Receiving objects: 100% (32729/32729), 83.32 MiB | 29.79 MiB/s, done.

Resolving deltas: 100% (24127/24127), done.

# go to the directory PCR-GLOBWB\_model

$ cd PCR-GLOBWB\_model

$ pwd

/home/edwin/PCR-GLOBWB\_model

# list all files in PCR-GLOBWB\_model

$ ls

LICENSE README.md README.txt clone\_landmask\_maps conda\_env config exercise known\_issues.txt misc model modflow various\_tools

You may want to use “mc” (midnight commander) to explore the content of “PCR-GLOBWB\_model”. The two important folders that will be used are “model” and “config”. The folder “model” contains the model codes (python scripts) of PCR-GLOBWB, while the folder “config” contains the configuration (.ini) files to setup PCR-GLOBWB model runs.

**3.2 Adjusting configuration files .ini**

If you explore the folder “config” on the folder “PCRGLOBWB\_model”, you will find a setup/configuration file (.ini) called “**setup\_30min.ini**”. This configuration file is basically using the model parameters and forcing input files as used in the GMD paper run of PCR-GLOBWB 2 (Sutanudjaja et al., 2018, <https://gmd.copernicus.org/articles/11/2429/2018/>).

#change directory to the one where all the inputs examples are stored and open the file setup\_30min.ini

$ cd /projects/0/dfguu/data/hydroworld

$ls

$cd config

$geany setup\_30min.ini

#create a new directory for the output files called scratch-shared. This directory will be able to store TB of data and since our model outputs will be heavy it’s a good option.

$cd/scratch-shared/

$mkdir vbonato

#inside geany text editor, edit the inputDir, CloneMap and outputDir as written below, make sure that the field InputDir under the [globalOptions] is consistent to the one on your computational machine.

inputDir = /projects/0/dfguu/data/hydroworld/pcrglobwb2\_input\_release/version\_2019\_11\_beta/pcrglobwb2\_input/global\_30min/cloneMaps/clone\_global\_30min.map

outputDir = /scratch-shared/vbonato/pcrglobwb2\_output/30min\_global/

CloneMap = /projects/0/dfguu/data/hydroworld/pcrglobwb2\_input\_release/version\_2019\_11\_beta/pcrglobwb2\_input/

**3.3 Run PCR-GLOBWB using the sbatch\_example.sh**

We will use a computational node to run PCR-GLOBWB, using the file “sbatch\_example.sh” that is located on “/home/edwin/jobs/example/”. Please copy it to your directory and open it using geany.

# copy the file to your own directory currently running

$ cp /home/edwin/jobs/example/sbatch\_example.sh /home/vbonato

# open it using geany

$ geany sbatch\_example.sh

A screenshot of a computer

Description automatically generated

# change the type of node from thin to genoa and the maximum wall clock time from 59 min to 120h. Don’t change the number of node (1) and the cores (32).

Finally delate the infinite loop and put the run of PCR-GLOBWB

#SBATCH -p genoa

#SBATCH -t 120:00:00

Instead of infinite loop put:

$ cd $HOME

$ . load\_all\_default.sh

$ cd /PCR-GLOBWB\_model/model

# to run PCR-GLOBWB you use deterministic\_runner.py, you can also add debug at the end of this line to get more information printed on screen while our model is running

$ python deterministic\_runner.py ../config/setup\_30min.ini

#remember to change the setup file before running this

Please submit the “sbatch\_example.sh” job using the following sbatch command.

$ sbatch sbatch\_example.sh

sbatch: Single node jobs run on a shared node by default. Add --exclusive if you want to use a node exclusively.

sbatch: You will be charged for 0.25 node. A full node consists of 128 CPU cores, 229376 MiB of memory and 0 GPUs and can be shared by up to 4 jobs.

Submitted batch job 1859254

You can check its status, by using the following “squeue” command.

# squeue

JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)

1859254 thin example edwinmod PD 0:00 1 (Priority)

# At this moment the job was still queueing. After a while, please check it again:

$ squeue

JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)

1849203 staging touch-ed edwinmod PD 0:00 1 (BeginTime)

1859254 thin example edwinmod R 0:26 1 tcn81

# now it was already running; this job, which contains an infinite loop, would run until its wall clock time finished (59 mins)

Please check the slurm output of the job (for example using geany).

# geany slurm-<your\_job\_id>.out

$ geany slurm-1859254.out

I can access the computational node where my job is working:

#entering the computational node, I can load the software and start python scripts

$ ssh -XY tcn81

$ . load\_all\_default.sh

$ sbatch sbatch\_example.sh

#if I want to exit the computing node

$exit

In other future run, if your job is still running and you may want to cancel it using the following command

# to cancel a job: scancel <your\_job\_id>

$ scancel 1859254

**3.4 Viewing your results**

While your model is still running, you may already want to explore the files/folders that are written to the output directory “outputDir”.

printed on screen while our model is running

$ cd /scratch-shared/vbonato/pcrglobwb2\_output/30min\_global/

$ls

log maps netcd scripts states tmp

Explore to all sub-folders and try to get some ideas about the files that are written in each sub-folder (also one of the slides from the presentation in the morning):

- log : logger, backup/copy of ini file that you used

- maps : usually empty (mainly used during debugging process)

- netcdf : netcdf output files are stored here

- scripts : backup of the python script files that you used

- states : model states at the end of the each simulation year (\*12-31.map)

- tmp: : used during resampling/cropping processes

You may want to visualize your netcdf output files

$ cd /scratch-shared/vbonato/pcrglobwb2\_output/30min\_global/netcdf/

$ncview discharge\_dailyTot\_output.nc

1. **How snellius works**

**4.1 Entering a computational node**

See the picture below to understand the difference between the login nodes and computational nodes:

A diagram of a computer program

Description automatically generated

There are a few login nodes and you only use them for light tasks: adjusting your code, preparing your input data, writing job scripts nodes. Longer job will be killed in login nodes so they need to be run in computational nodes only. You do not directly interact with any computational node. Instead, you can request computational resources with a job script, and snellius will assign a computational node to this job using a SLURM job scheduler. If all the resources are occupied, your job will be placed in a queue, and scheduled when resources are available.

To know more about snellius you can look at: <https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/tutorial1/Lisa_Cluster.html#:~:text=You%20do%20not%20directly%20interact,using%20a%20SLURM%20job%20scheduler>

#typing sview you can see your jobID, partition you are working on (for example later we will be using genoa computational node, user name, state, time running, number of node and the name of the node)

$ sview

Type the following commands (that are available for Snellius only).

# to check your account information, remaining budget (SBUs), etc

$ accinfo

# to check the SBUs that were used (not always up to date)

$ accuse

$ accuse -d

# to check the status of SBUs (better, including running jobs)

$ budget-overview

Enter a computational node to run our model so that we have enough time to run the simulation.

**4.2 Leaving snellius**

Use the command “exit” at the end of the guide. If you try to use it now remember to re-enter into snellius following the previous steps. You will return on your xquarz terminal this way.

# leaving snellius (and return exiting to xquartz terminal)

$ exit

logout

Connection to snellius.surf.nl closed