1 Hardware

The department of Meteorology and Geophysics has access to 2 servers (**SRVX1**, **SRVX8**) as well as one cluster (**JET**). Additionally the department has access to private nodes on **VSC 4/5**. As a **staff** member, **PhD** student or **external** researcher you will have **access to all** of these resources if necessary. **Students** get access to the **TeachingHub**. Master students can get access to everything as well. **Please talk to your supervisor.**

Servers & Clusters

Name	Location	# Nodes	Type
SRVX1	Arsenal	1	development
SRVX8	Arsenal	1	visual
JET	Arsenal	7	compute
VSC4	Arsenal	5	compute
VSC5	Arsenal	11	compute
VSC5	Arsenal	1	GPU

CPU Architectures

Node	CPU	# Cores	RAM
jet0**X**	Intel(R) Xeon(R) Gold 6148	2x20	768GB
srvx1	Intel(R) Xeon(R) Gold 6148	4x20	768GB
srvx8	Intel(R) Xeon(R) CPU E5-2697	2x14	510GB

File systems & Quotas (Default)

mountpoint	space	quotas	fs
/users/staff	400TB	100GB	local
/users/students	10TB	50GB	local
/scratch	400TB	no	local
/jetfs/home	100TB	100GB	global
/jetfs/scratch[a]	1.4PB	no	global

[a]Note /scratch and /jetfs/scratch are shared by all users. Please use responsibly.

2 Access

An IMGW server account can be requested by your supervisor. Please connect to the servers using one of the following suggestions:

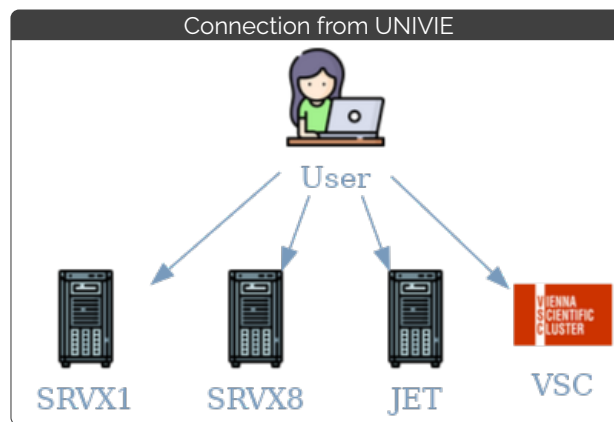- **ssh** available on Linux, iOS
- **MobaXterm** available on Windows

any other software is fine too.

Password Manager

Please to use a password manager e.g. Bitwarden, KeepassXC. Remember that using the TeachingHub requires a 2FA Authenticator.

Access via secure shell

```
ssh <username>@srvx1.img.univie.ac.at
ssh <username>@jet01.img.univie.ac.at
ssh <username>@jet02.img.univie.ac.at
ssh <username>@srvx8.img.univie.ac.at
ssh <username>@vsc4.vsc.ac.at
ssh <username>@vsc5.vsc.ac.at
```
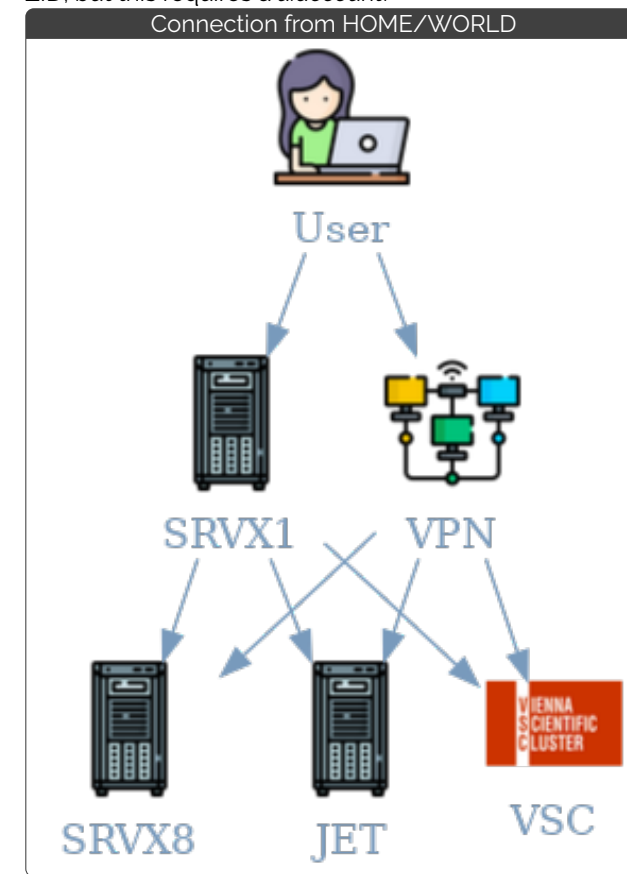

Connection from UNIVIE

If you are **outside** the university network, e.g. at home, EDUroam, outside AT, ... it is required to login via a jump host to most of the servers.

SSH with jump host

```
ssh -J <username>@srvx1 <username>@jet01
ssh -J <u:account>@login.univie.ac.at <username>@jet01
```

Alternatively, it is also possible to use the VPN from the university of Vienna. More information can be viewed at the ZID, but this requires a *u:account*.

Connection from HOME/WORLD

3   Help & Documentation

There is extensive information available on these sites, depending on your access level. The most convenient is to use the **documentation** available on Wolke, which is available as well on GitLab. For staff members there are information on the u:wiki

Contact

```
IT support .......... it.img-wien@univie.ac.at
JET group .......... jet.img-wien@univie.ac.at
VSC group .......... vsc.img-wien@univie.ac.at
IT HPC ......... michael.blaschek@univie.ac.at
VSC support ................ service@vsc.ac.at
Gitlab support . support.phaidra@univie.ac.at
```

4   Software

All servers at the department as well as all HPC systems have a software stack that is based on **modules**. These so called environment modules allow **dynamic loading of different versions of the same software and dependencies**. Common commands are shown here:

```
module av ...................... Show available modules
module load <name> .............. Load module <name>
module unload <name> .......... Unload module <name>
module purge ........................ Unload all module
module list ........................ List loaded modules
module show <name> .............. Show module <name>
```

4.1   Software Stacks

There are software stacks usually grouped by **compiler**:

- GNU Compilers (`gcc`, `gfortran`, `g++`)
- Intel Compilers (`icc`, `ifort`, `icpc`)
- Intel Oneapi Compilers (`icx`, `ifx`, `icpx`)
- AMD Compilers AOCC (`clang`, `flang`, `clang++`)

and then there are usually different version of these compilers and message parsing interface (**MPI**) implementations

- `openmpi`
- `intel-mpi`
- `mpich`

As you can see this evolves really quickly into a large tree with a lot of different options and configurations. A simple example:

Loading a software stack

```
# Load the openmpi version 3.1.6 with GNU compiler 8.5.0
module load openmpi/3.1.6-gcc-8.5.0
# show what is loaded
module list
Currently Loaded Modulefiles:
 1) openmpi/3.1.6-gcc-8.5.0   2) gcc/8.5.0-gcc-8.5rhel8
```

How to compile a Fortran program?

```
# Load the module
module load gcc-8.5.0
# Compile the program
gfortran -x -o test.exe test.f90
# unload all modules
module purge
# load intel compiler
module load intel-oneapi-compiler
# Compile the program
ifort -x -o test.i.exe test.f90
```

5   Job scheduler

Most HPC systems use a job scheduler. **VSC and JET use slurm**, which uses the following commands to control jobs:

```
sinfo ................................ partition information
squeue ................................ queue information
sqos ........................ quality of service information
scontrol show job <jobid> ........ show job information
salloc ............................ request an allocation
srun -N<n> <cmd> ............................ run a cmd
sbatch <job file> ........................ run job script
scancel <jobid> ........................... cancel a job
scontrol update job <jobid> <option> ..... Change job
settings while running, e.g. TimeLimit
```

Example script on JET

```
#!/bin/bash
# SLURM specific commands
#SBATCH --job-name=test-run
#SBATCH --output=test-run.log
#SBATCH --ntasks=1
#SBATCH --mem=1MB
#SBATCH --time=05:00
#SBATCH --mail-type=BEGIN
#SBATCH --mail-user=<email@address.at>

# Your Code below here
module load miniconda3
# Execute the miniconda Python
# use /usr/bin/time -v [program]
# gives statistics on the resources the program uses
# nice for testing
/usr/bin/time -v python3 -v
```

Common sbatch options:

```
--job-name=<name> ............................. job name
--nodes=<n> ......................... number of nodes
--ntasks=<n> ......................... number of tasks
--ntasks-per-node=<n>   tasks in parallel on a single node
--ntasks-per-core=<n> ........... tasks on a single core
--mem=<mem> ..................... max memory: 1MB, 1GB
--time=<time> ......... estimated run time: D-HH:MM:SS
```

Please note that when you do not supply the `--output=` option, a file called `slurm-<jobid>.out` will be created by default.

Examples - resource control inside the job script

```
# openmp
# --ntasks=1 --cpus-per-task=n
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
# --ntasks=n
export OMP_NUM_THREADS=$SLURM_NTASKS
<exe>

# MPI
module purge
module load openmpi/xx.yy.zz
mpirun <exe>
```

Interactive job

```
# Request resources from slurm (-N 1, a full Node)
salloc -N 1 -p <partition> --qos <only on VSC> --no-shell
# Once the node is assigned / job is running
# Check with
squeue -u $USER
# connect to the Node with ssh
ssh [Node]
# test and debug the model there.
```

5.1 JET special commands

On the Jet cluster there are a few special commands to make your life easier.

```
jobinfo <jobid> ..... Shows information on a running job
jobinfo_remaining  Shows how long the current jobs lasts
nodeinfo ........... Shows usage of JET compute nodes
queueinfo .......... Shows queue information per node
watchjob <jobid> ................. Monitor a running job
```

JET slurm partitions

```
hub ................................... jet03-jet06
compute ............................. jet04-jet09
time limit ................................... no
```

more information on the details can be found in the documentation, check the help section.

JET jobs

Jobs can share nodes on JET. Not on VSC.

6 User services

All department servers have **special scripts** that are meant to make the users life easier. Please find a list of useful commands here:

`userservices <service>` . Master function for all services

All of these services have a help (`-h`) and example section.

List of Services:

```
containers ......... Show available apptainer containers
fetch-sysinfo ............... Display system information
filesender .................... Transfer files to ACONET
fix-permissions ........... fix file/directory permissions
modules ........................ list environment modules
numfiles ...... Check number of files in all sub directories
quota ................................. Report user quotas
server-schedule ................. Show Server schedule
sshtools ........................... SSH agent user help
transfersh ..................... Transfer files/directories
ucloud .......... Upload files/directories to your u:cloud
weather .................... Retrieve weather information
yopass ........... Send messages/small files encrypted
vnc ...................... Manage a VNC session (SRVX8)
vnc-geometry ... Change VNC desktop resolution (SRVX8)
```

Other useful commands:

```
ncdu -x <dir> ........... Show disk usage of <dir> (man)
htop .......................... Show running processes
jobs ................................................ Show
```

Learn more about linux commands from **explainshell.com** or check the **man.cx** for manual pages of linux commands.

7 Vienna Scientific Cluster (VSC)

The **VSC is Austria's university HPC system** (part of EuroCC), which the department has private nodes and **researchers can request projects to get more resources.** Please note that on VSC only full nodes can be requested, whereas on JET nodes can be shared. Check their wiki Currently available VSC HPC clusters:

1. VSC4 since 2019
2. VSC5 since 2022

Connect to VSC

```
ssh <vscname>@vsc4.vsc.ac.at
ssh <vscname>@vsc5.vsc.ac.at
# jump host
ssh -J <username>@srvx1 <vscname>@vsc4.vsc.ac.at
```

VSC Node Setup

HPC	#	CPU	# Cores	RAM
VSC4	5	Intel(R) Xeon(R) Platinum 8174	2x24	384GB
VSC5	11	AMD EPYC 7713	2x64	512GB
VSC5	1	GPU Nvidia A100	2x64	512GB

VSC slurm partitions

```
p71386_0384 ........................... VSC4 qos
skylake_0384 ..................... VSC4 partition
p71386_0384, time limit .............. 10 days
p71386_0512 ........................... VSC5 qos
zen3_0512 ........................ VSC5 partition
p71386_a100dual ................ VSC5 GPU qos
zen3_0512_a100x2 ........... VSC5 GPU partition
p71386_0512, time limit ................ 3 days
```

VSC Storage shared VSC4/VSC5

name	space	# files	comment
HOME	200GB	2.000.000	global fs
DATA	100TB	2.000.000	global fs

Please note that these storage quotas are for all members of the department together. That means sharing and responsible behavior for the benefit of all.

Example VSC job

```
#!/bin/bash
#
#SBATCH -J TEST_JOB
#SBATCH -N 2
#SBATCH --ntasks-per-node=64
#SBATCH --ntasks-per-core=1
#SBATCH --mail-type=BEGIN
#SBATCH --mail-user=<email@address.at>
#SBATCH --partition=zen3_0512
#SBATCH --qos=p71386_0512
#SBATCH --account=p71386
#SBATCH --time=<time>

# when srun is used, you need to set (Different from Jet):
<srun -l -N2 -n64 a.out >
# or
<mpirun -np 64 a.out>
```

8 ECMWF

It possible for you to get an account on ECMWF Bologna HPC system via your supervisor.

Connect to ECMWF

```
# requires teleport
module load teleport
# start ssh-agent
startagent
# run browserless login
python3 -m teleport.login
# login using your ECMWF credentials
ssh -J <user>@jump.ecmwf.int <user>@ecs-login
```

On all department servers a module called `ecaccess-webtoolkit` is installed, that can be used to **monitor, submit jobs, transfer files.** For convenient transfer use `ectrans` to transfer files from and to ECMWF. Predefined **associations** can be configured using boaccess.ecmwf.int. Other useful service @ECMWF:

- confluence.ecmwf.int Documentation
- desktop.ecmwf.int Virtual Machine
- ECcharts Maps + JupyterNotebooks

9 Spack

Spack is an open source project that offers a package management framework and tool for installing complex scientific software. It is designed to support multiple versions and configurations of a software on many different platforms and environments.

All the libraries and compilers are installed using spack and different version can be installed as you go. There is also the possibility as a user to use spack and install and build applications that have specific requirements. However, this is rather complex.

```
spack list ............ List and search all avail. packages
spack find ..................... List all installed packages
spack info <pkg> ....... Show information on a package
spack spec -I <pkg> .............. Show dependencies
spack install <pkg>@<version> ....... install a package
spack compiler list ............... List avail. compilers
spack load <pkg> .......................... Load module
```

```
spack unload <pkg> ...................... Unload module
```

spack specs

spec	meaning	example
@	custom version	mpileaks@3.3
+/-/~	build options/-variants	mpileaks@3.3 +threads
target=	Set CPU architecture	target=cascadelake
^	dependency information	mpileaks ^mpich@3.2
/	specify by hash	spack load /h4jqiw

spack user environment

```
# init spack
# For bash/zsh/sh
. $SPACK_ROOT/share/spack/setup-env.sh
# create a spack directory
mkdir $HOME/myspack
# create spack environment
spack env create -d $HOME/myspack
# activate spack environment
spack env activate $HOME/myspack
# install a package to your new environment
spack install gcc@10.3.1
# deactivate
spack env deactivate
```

more information can be found in the spack.readthedocs.io documentation.

spack files/storage

Remember that using spack a lot of files are created. Please keep in mind what you do and the impact this might have on others.

Spack can be used to build a containerized version of your library stack.

10 Python / Conda

On all servers `conda` is installed via the module system. Load one of the available modules, e.g. `miniconda3` or `micromamba` and start developing your Python code. Micromamba is a C version of conda and much faster.

Setup a Python environment

```
module load miniconda3
# install a python version
conda create --name myenv python=3.11 <other pkg>
# install package with version
conda install -n myenv scipy=0.18
# show what environment you have
conda env list
```

```
conda ............................. Master conda process
conda env list ................. List conda environments
conda install -n <env> <pkg> ..... Install a pkg into env
conda update ........................... Update packages
```

More information can be found in the documentation
It is possible to create a kernel for the TeachingHub or the ResearchHub from your environment. Just install `conda install -n <env> ipykernel` into your environment and you should be able to select it for notebooks.

11 Singularity / Apptainer

Singularity/Apptainer is a container technology for HPC that is designed to execute applications at high performance while being secure, portable, and reproducible. It is installed on all servers and VSC. Some applications are installed as containers, e.g. `userservices containers`
Usually commands `apptainer` and `singularity` can be used interchangeable.
Some common commands:

```
apptainer help .............................. Show help
apptainer run docker://alpine ... Run alpine linux from dockerhub
apptainer pull docker://alpine ....... Pull image from dockerhub
apptainer build image.sif docker://alpine ..... Build image and save in file
apptainer shell image.sif .. Interact with the container
```

`apptainer exec image.sif date` .... Execute `date` inside container

`apptainer run image.sif` Run default app from container

More complex things can be done, and are documented a bit here: gitlab.phaidra.org/imgw/singularity

12 Recipes

12.1 Build an app with modules

When developing an application, there are usually some dependencies or libraries that you do want to use, but not deliver. Other people shall install these themselves. This is very common in software development. And a big mess. HPC centers use environment modules to make sure that libraries are available in different versions. These modules set the following environment variables:

- LIBRARY

- INCLUDE

Therefore you should include these in your `Makefile` and make use of these paths during the build process.

Makefile with environment modules

```
# use the environmental variable $INCLUDE
# split the paths separated by :
INC = $(subst :, ,$(INCLUDE))
# add a -I/path/to/include
INC := $(INC:%=-I%)
# use the environmental variable $LIBRARY
LIBS = $(subst :, ,$(LIBRARY))
LIBS := $(LIBS:%=-L%)
```

12.2 GNU Fortran compiler options

`-fdefault-real-8` .... double precision real
`-fbounds-check` .... check array bounds
`-fbacktrace` .... call chain traceback

`-fconvert=big-endian/little-endian` . convert little/big endian
`-O0` .... default optimization
`-O3` .... highest optimization
`-frecord-marker=8` .... Length of record marker for unformatted files

12.3 Intel Fortran compiler options

`-r8` .... double precision real
`-check` .... check array bounds
`-backtrace` .... call chain traceback
`-convert big_endian/little_endian` .. convert little/big endian
`-O2` .... default optimization
`-O2 or -O3 or -fast` .... highest optimization
`-mcmodel=medium` .... Memory Model