

Running jobs with input/output data

You have now learned how to start a batch job and how to start an interactive session. The next question is how to deal with input and output files, where your standard output and error messages will go to and where that you can collect your results.

The current directory and output and error files

Default file names

First go to the directory:

```
cd ~/examples/Running_jobs_with_input_output_data
```



Note

If the example directory is not yet present, copy it to your home directory:

```
cp -r /apps/gent/tutorials/Intro-HPC/examples ~/
```

List and check the contents with:

```
$ ls -l
total 2304
-rwxrwxr-x 1 vsc40000 682 Sep 13 11:34 file1.py
-rw-rw-r-- 1 vsc40000 212 Sep 13 11:54 file1a.pbs
-rw-rw-r-- 1 vsc40000 994 Sep 13 11:53 file1b.pbs
-rw-rw-r-- 1 vsc40000 994 Sep 13 11:53 file1c.pbs
-rw-r--r-- 1 vsc40000 1393 Sep 13 10:41 file2.pbs
-rwxrwxr-x 1 vsc40000 2393 Sep 13 10:40 file2.py
-rw-r--r-- 1 vsc40000 1393 Sep 13 10:41 file3.pbs
-rwxrwxr-x 1 vsc40000 2393 Sep 13 10:40 file3.py
```

Now, let us inspect the contents of the first executable (which is just a Python script with execute permission).

file1.py

```
#!/usr/bin/env python
#
# VSC          : Flemish Supercomputing Centre
# Tutorial     : Introduction to HPC
```

```

# Description: Writing to the current directory, stdout and stderr
#
import sys

# Step #1: write to a local file in your current directory
local_f = open("Hello.txt", 'w+')
local_f.write("Hello World!\n")
local_f.write("I am writing in the file:<Hello.txt>.\n")
local_f.write("in the current directory.\n")
local_f.write("Cheers!\n")
local_f.close()

# Step #2: Write to stdout
sys.stdout.write("Hello World!\n")
sys.stdout.write("I am writing to <stdout>.\n")
sys.stdout.write("Cheers!\n")

# Step #3: Write to stderr
sys.stderr.write("Hello World!\n")
sys.stderr.write("This is NO ERROR or WARNING.\n")
sys.stderr.write("I am just writing to <stderr>.\n")
sys.stderr.write("Cheers!\n")

```

The code of the Python script, is self explanatory:

1. In step 1, we write something to the file `hello.txt` in the current directory.
2. In step 2, we write some text to stdout.
3. In step 3, we write to stderr.

Check the contents of the first job script:

file1a.pbs

```

#!/bin/bash

#PBS -l walltime=00:05:00

# go to the (current) working directory (optional, if this is the
# directory where you submitted the job)
cd $PBS_O_WORKDIR

# the program itself
echo Start Job
date
./file1.py
echo End Job

```

You'll see that there are NO specific PBS directives for the placement of the output files. All output files are just written to the standard paths.

Submit it:

```
qsub file1a.pbs
```

After the job has finished, inspect the local directory again, i.e., the directory where you executed the *qsub* command:

```
$ ls -l
total 3072
-rw-rw-r-- 1 vsc40000  90 Sep 13 13:13 Hello.txt
-rwxrwxr-x 1 vsc40000 693 Sep 13 13:03 file1.py*
-rw-rw-r-- 1 vsc40000 229 Sep 13 13:01 file1a.pbs
-rw----- 1 vsc40000  91 Sep 13 13:13 file1a.pbs.e123456
-rw----- 1 vsc40000 105 Sep 13 13:13 file1a.pbs.o123456
-rw-rw-r-- 1 vsc40000 143 Sep 13 13:07 file1b.pbs
-rw-rw-r-- 1 vsc40000 177 Sep 13 13:06 file1c.pbs
-rw-r--r-- 1 vsc40000 1393 Sep 13 10:41 file2.pbs
-rwxrwxr-x 1 vsc40000 2393 Sep 13 10:40 file2.py*
-rw-r--r-- 1 vsc40000 1393 Sep 13 10:41 file3.pbs
-rwxrwxr-x 1 vsc40000 2393 Sep 13 10:40 file3.py*
```

Some observations:

1. The file `Hello.txt` was created in the current directory.
2. The file `file1a.pbs.o123456` contains all the text that was written to the standard output stream ("stdout").
3. The file `file1a.pbs.e123456` contains all the text that was written to the standard error stream ("stderr").

Inspect their contents ... and remove the files

```
$ cat Hello.txt
$ cat file1a.pbs.o123456
$ cat file1a.pbs.e123456
$ rm Hello.txt file1a.pbs.o123456 file1a.pbs.e123456
```



Tip

Type `cat H` and press the Tab button (looks like `Tab ↵`), and it will **expand** into `cat Hello.txt`.

Filenames using the name of the job

Check the contents of the job script and execute it.

```
file1b.pbs
```

```
#!/bin/bash

# Specify the "name" of the job
#PBS -N my_serial_job

cd $PBS_O_WORKDIR
echo Start Job
date
./file1.py
echo End Job
```

Inspect the contents again ... and remove the generated files:

```
$ ls
Hello.txt file1a.pbs file1c.pbs file2.pbs file3.pbs my_serial_job.e123456
file1.py* file1b.pbs file2.py* file3.py* my_serial_job.o123456
$ rm Hello.txt my_serial_job.*
```

Here, the option "`-N`" was used to explicitly assign a name to the job. This overwrote the `JOBNAME` variable, and resulted in a different name for the *stdout* and *stderr* files. This name is also shown in the second column of the "`qstat`" command. If no name is provided, it defaults to the name of the job script.

User-defined file names

You can also specify the name of *stdout* and *stderr* files explicitly by adding two lines in the job script, as in our third example:

```
file1c.pbs

#!/bin/bash

# redirect standard output (-o) and error (-e)
#PBS -o stdout.$PBS_JOBID
#PBS -e stderr.$PBS_JOBID

cd $PBS_O_WORKDIR
echo Start Job
date
./file1.py
echo End Job
```

Where to store your data on the HPC

The HPC cluster offers their users several locations to store their data. Most of the data will reside on the shared storage system, but all compute nodes also have their own (small) local disk.

Pre-defined user directories

Three different pre-defined user directories are available, where each directory has been created for different purposes. The best place to store your data depends on the purpose, but also the size and type of usage of the data.

The following locations are available:

Variable	Description
<i>Long-term storage slow filesystem, intended for smaller files</i>	
<code>\$VSC_HOME</code>	For your configuration files and other small files, see the section on your home directory . The default directory is <code>user/Gent/xxx/vsc40000</code> . The same file system is accessible from all sites, i.e., you'll see the same contents in <code>\$VSC_HOME</code> on all sites.
<code>\$VSC_DATA</code>	A bigger "workspace", for datasets , results, logfiles, etc. see the section on your data directory . The default directory is <code>data/Gent/xxx/vsc40000</code> . The same file system is accessible from all sites.
<i>Fast temporary storage</i>	
<code>\$VSC_SCRATCH_NODE</code>	For temporary or transient data on the local compute node, where fast access is important; see the section on your scratch space . This space is available per node. The default directory is <code>/tmp</code> . On different nodes, you'll see different content.
<code>\$VSC_SCRATCH</code>	For temporary or transient data that has to be accessible from all nodes of a cluster (including the login nodes). The default directory is <code>scratch/Gent/xxx/vsc40000</code> . This directory is cluster- or site-specific: On different sites, and sometimes on different clusters on the same site, you'll get a different directory with different content.
<code>\$VSC_SCRATCH_SITE</code>	Currently the same as <code>\$VSC_SCRATCH</code> , but could be used for a scratch space shared across all clusters at a site in the future. See the section on your scratch space .
<code>\$VSC_SCRATCH_GLOBAL</code>	Currently the same as <code>\$VSC_SCRATCH</code> , but could be used for a scratch space shared across all clusters of the VSC in the future. See the section on your scratch space .

Variable	Description
<code>\$VSC_SCRATCH</code> <code>H_CLUSTER</code>	The scratch filesystem closest to the cluster.
<code>\$VSC_SCRATCH</code> <code>H_ARCANINE</code>	A separate (smaller) shared scratch filesystem, powered by SSDs. This scratch filesystem is intended for very I/O-intensive workloads.

Since these directories are not necessarily mounted on the same locations over all sites, you should always (try to) use the environment variables that have been created.

We elaborate more on the specific function of these locations in the following sections.

Note: `$VSC_SCRATCH_KYUKON` and `$VSC_SCRATCH` are the same directories ("kyukon" is the name of the storage cluster where the default shared scratch filesystem is hosted).

For documentation about VO directories, see [the section on VO directories](#).

Your home directory (`$VSC_HOME`)

Your home directory is where you arrive by default when you login to the cluster. Your shell refers to it as "~" (tilde), and its absolute path is also stored in the environment variable `$VSC_HOME`. Your home directory is shared across all clusters of the VSC.

The data stored here should be relatively small (e.g., no files or directories larger than a few megabytes), and preferably should only contain configuration files. Note that various kinds of configuration files are also stored here, e.g., by MATLAB, Eclipse, ...

The operating system also creates a few files and folders here to manage your account. Examples are:

File or Directory	Description
<code>.ssh/</code>	This directory contains some files necessary for you to login to the cluster and to submit jobs on the cluster. Do not remove them, and do not alter anything if you don't know what you are doing!
<code>.bash_profile</code>	When you login (type username and password) remotely via ssh, <code>.bash_profile</code> is executed to configure your shell before the initial command prompt.

File or Directory	Description
.bashrc	This script is executed every time you start a session on the cluster: when you login to the cluster and when a job starts.
.bash_history	This file contains the commands you typed at your shell prompt, in case you need them again.

Your data directory (\$VSC_DATA)

In this directory you can store all other data that you need for longer terms (such as the results of previous jobs, ...). It is a good place for, e.g., storing big files like genome data.

The environment variable pointing to this directory is \$VSC_DATA. This volume is shared across all clusters of the VSC. There are however no guarantees about the speed you will achieve on this volume. For guaranteed fast performance and very heavy I/O, you should use the scratch space instead.

If you are running out of quota on your _\$VSC_DATA filesystem you can join an existing VO, or request a new VO. See [the section about virtual organisations](#) on how to do this.

Your scratch space (\$VSC_SCRATCH)

To enable quick writing from your job, a few extra file systems are available on the compute nodes. These extra file systems are called scratch folders, and can be used for storage of temporary and/or transient data (temporary results, anything you just need during your job, or your batch of jobs).

You should remove any data from these systems after your processing them has finished. There are no guarantees about the time your data will be stored on this system, and we plan to clean these automatically on a regular basis. The maximum allowed age of files on these scratch file systems depends on the type of scratch, and can be anywhere between a day and a few weeks. We don't guarantee that these policies remain forever, and may change them if this seems necessary for the healthy operation of the cluster.

Each type of scratch has its own use:

Node scratch (\$VSC_SCRATCH_NODE). Every node has its own scratch space, which is completely separated from the other nodes. On some clusters, it will be on a local disk in the node, while on other clusters it will be emulated through another file server. Some **drawbacks** are that the storage can only be accessed on that particular node and that the capacity is often very limited (e.g., 100 GB). The performance will depend a lot on the particular implementation

in the cluster. In many cases, it will be significantly slower than the cluster scratch as it typically consists of just a single disk. However, if that disk is local to the node (as on most clusters), the performance will not depend on what others are doing on the cluster.

Cluster scratch (\$VSC_SCRATCH). To allow a job running on multiple nodes (or multiple jobs running on separate nodes) to share data as files, every node of the cluster (including the login nodes) has access to this shared scratch directory. Just like the home and data directories, every user has its own scratch directory. Because this scratch is also available from the login nodes, you could manually copy results to your data directory after your job has ended. Also, this type of scratch is usually implemented by running tens or hundreds of disks in parallel on a powerful file server with fast connection to all the cluster nodes and therefore is often the fastest file system available on a cluster. You may not get the same file system on different clusters, i.e., you may see different content on different clusters at the same institute.

Site scratch (\$VSC_SCRATCH_SITE). At the time of writing, the site scratch is just the same volume as the cluster scratch, and thus contains the same data. In the future it may point to a different scratch file system that is available across all clusters at a particular site, which is in fact the case for the cluster scratch on some sites.

Global scratch (\$VSC_SCRATCH_GLOBAL). At the time of writing, the global scratch is just the same volume as the cluster scratch, and thus contains the same data. In the future it may point to a scratch file system that is available across all clusters of the VSC, but at the moment of writing there are no plans to provide this.

Your UGent home drive and shares

In order to access data on your UGent share(s), you need to stage-in the data and stage-out afterwards. On the login nodes, it is possible to access your UGent home drive and shares. To allow this you need a ticket. This requires that you first authenticate yourself with your UGent username and password by running:

```
$ kinit yourugentusername@UGENT.BE
Password for yourugentusername@UGENT.BE :
```

Now you should be able to access your files running

```
$ ls /UGent/yourugentusername
home shares www
```

Please note the shares will only be mounted when you access this folder. You should specify your complete username - tab completion will not work.

If you want to use the UGent shares longer than 24 hours, you should ask a ticket for up to a week by running

```
kinit yourugentusername@UGENT.BE -r 7
```

You can verify your authentication ticket and expiry dates yourself by running klist

```
$ klist
...
Valid starting      Expires            Service principal
14/07/20 15:19:13  15/07/20 01:19:13  krbtgt/UGENT.BE@UGENT.BE
    renew until 21/07/20 15:19:13
```

Your ticket is valid for 10 hours, but you can renew it before it expires.

To renew your tickets, simply run

```
kinit -R
```

If you want your ticket to be renewed automatically up to the maximum expiry date, you can run

```
krenew -b -K 60
```

Each hour the process will check if your ticket should be renewed.

We strongly advise to disable access to your shares once it is no longer needed:

```
kdestroy
```

If you get an error "*Unknown credential cache type while getting default ccache*" (or similar) and you use conda, then please deactivate conda before you use the commands in this chapter.

```
conda deactivate
```

UGent shares with globus

In order to access your UGent home and shares inside the globus endpoint, you first have to generate authentication credentials on the endpoint. To do that, you have to ssh to the globus endpoints from a loginnode. You will be prompted for your UGent username and password to authenticate:

```
$ ssh globus01 # also do this for globus02
UGent username:ugentusername
Password for ugentusername@UGENT.BE:
Shares are available in globus endpoint at /UGent/ugentusername/
Overview of valid tickets:
Ticket cache: KEYRING:persistent:xxxxxxx:xxxxxxx
Default principal: ugentusername@UGENT.BE

Valid starting      Expires            Service principal
```

```
29/07/20 15:56:43 30/07/20 01:56:43 krbtgt/UGENT.BE@UGENT.BE
renew until 05/08/20 15:56:40
Tickets will be automatically renewed for 1 week
Connection to globus01 closed.
```

Your shares will then be available at /UGent/ugentusername/ under the globus VSC tier2 endpoint. Tickets will be renewed automatically for 1 week, after which you'll need to run this again. We advise to disable access to your shares within globus once access is no longer needed:

```
$ ssh globus01 destroy
Succesfully destroyed session
```

Pre-defined quotas

Quota is enabled on these directories, which means that the amount of data you can store there is limited. This holds for both the total size of all files as well as the total number of files that can be stored. The system works with a soft quota and a hard quota. You can temporarily exceed the soft quota, but you can never exceed the hard quota. The user will get warnings as soon as he exceeds the soft quota.

To see your a list of your current quota, visit the VSC accountpage: <https://account.vscentrum.be>. VO moderators can see a list of VO quota usage per member of their VO via <https://account.vscentrum.be/django/vo/>.

The rules are:

1. You will only receive a warning when you have reached the soft limit of either quota.
2. You *will* start losing data and get I/O errors when you reach the hard limit. In this case, data loss will occur since nothing can be written anymore (this holds both for new files as well as for existing files), until you free up some space by removing some files. Also note that you *will not* be warned when data loss occurs, so keep an eye open for the general quota warnings!
3. The same holds for running jobs that need to write files: when you reach your hard quota, jobs will crash.

We do realise that quota are often observed as a nuisance by users, especially if you're running low on it. However, it is an essential feature of a shared infrastructure. Quota ensure that a single user cannot accidentally take a cluster down (and break other user's jobs) by filling up the available disk space. And they help to guarantee a fair use of all available resources for all users. Quota also help to ensure that each folder is used for its intended purpose.

Writing Output files

**Tip**

Find the code of the exercises in "`~/examples/Running_jobs_with_input_output_data`"

In the next exercise, you will generate a file in the `$VSC_SCRATCH` directory. In order to generate some CPU- and disk-I/O load, we will

1. take a random integer between 1 and 2000 and calculate all primes up to that limit;
2. repeat this action 30.000 times;
3. write the output to the "`primes_1.txt`" output file in the `$VSC_SCRATCH`-directory.

Check the Python and the PBS file, and submit the job: Remember that this is already a more serious (disk-I/O and computational intensive) job, which takes approximately 3 minutes on the HPC.

```
$ cat file2.py
$ cat file2.pbs
$ qsub file2.pbs
$ qstat
$ ls -l
$ echo $VSC_SCRATCH
$ ls -l $VSC_SCRATCH
$ more $VSC_SCRATCH/primes_1.txt
```

Reading Input files

**Tip**

Find the code of the exercise "`file3.py`" in "`~/examples/Running_jobs_with_input_output_data`".

In this exercise, you will

1. Generate the file "`primes_1.txt`" again as in the previous exercise;
2. open the file;
3. read it line by line;
4. calculate the average of primes in the line;
5. count the number of primes found per line;
6. write it to the "`primes_2.txt`" output file in the `$VSC_SCRATCH`-directory.

Check the Python and the PBS file, and submit the job:

```
$ cat file3.py
$ cat file3.pbs
$ qsub file3.pbs
$ qstat
$ ls -l
$ more $VSC_SCRATCH/primes_2.txt
```

How much disk space do I get?

Quota

The available disk space on the HPC is limited. The actual disk capacity, shared by all users, can be found on the "Available hardware" page on the website. (<https://vscdocumentation.readthedocs.io/en/latest/hardware.html>) As explained in [the section on predefined quota](#), this implies that there are also limits to:

- the amount of disk space; and
- the number of files

that can be made available to each individual HPC user.

The quota of disk space and number of files for each HPC user is:

Volume	Max. disk space	Max. # Files
HOME	3 GB	20000
DATA	25 GB	100000
SCRATCH	25 GB	100000

Tip

The first action to take when you have exceeded your quota is to clean up your directories. You could start by removing intermediate, temporary or log files. Keeping your environment clean will never do any harm.



Tip

If you obtained your VSC account via UGent, you can get (significantly) more storage quota in the DATA and SCRATCH volumes by joining a Virtual Organisation (VO), see [the section on virtual organisations](#) for more information. In case of questions, contact hpc@ugent.be.

Check your quota

You can consult your current storage quota usage on the HPC-UGent infrastructure shared filesystems via the VSC accountpage, see the "Usage" section at <https://account.vscentrum.be>.

VO moderators can inspect storage quota for all VO members via <https://account.vscentrum.be/django/vo/>.

To check your storage usage on the local scratch filesystems on VSC sites other than UGent, you can use the `show_quota` command (when logged into the login nodes of that VSC site).

Once your quota is (nearly) exhausted, you will want to know which directories are responsible for the consumption of your disk space. You can check the size of all subdirectories in the current directory with the `du` (**Disk Usage**) command:

```
$ du
256 ./ex01-matlab/log
1536 ./ex01-matlab
768 ./ex04-python
512 ./ex02-python
768 ./ex03-python
5632
```

This shows you first the aggregated size of all subdirectories, and finally the total size of the current directory `.` (this includes files stored in the current directory).

If you also want this size to be "human-readable" (and not always the total number of kilobytes), you add the parameter `-h`:

```
$ du -h
256K ./ex01-matlab/log
1.5M ./ex01-matlab
768K ./ex04-python
512K ./ex02-python
768K ./ex03-python
5.5M .
```

If the number of lower level subdirectories starts to grow too big, you may not want to see the information at that depth; you could just ask for a summary of the current directory:

```
$ du -s
5632 .
$ du -s -h
```

If you want to see the size of any file or top-level subdirectory in the current directory, you could use the following command:

```
$ du -h --max-depth 1
1.5M ./ex01-matlab
512K ./ex02-python
768K ./ex03-python
768K ./ex04-python
256K ./example.sh
1.5M ./intro-HPC.pdf
700M ./cache
```

Finally, if you don't want to know the size of the data in your current directory, but in some other directory (e.g., your data directory), you just pass this directory as a parameter. The command below will show the disk use in your home directory, even if you are currently in a different directory:

```
$ du -h --max-depth 1 $VSC_HOME
22M /user/home/gent/vsc400/vsc40000/dataset01
36M /user/home/gent/vsc400/vsc40000/dataset02
22M /user/home/gent/vsc400/vsc40000/dataset03
3.5M /user/home/gent/vsc400/vsc40000/primes.txt
24M /user/home/gent/vsc400/vsc40000/.cache
```

Groups

Groups are a way to manage who can access what data. A user can belong to multiple groups at a time. **Groups can be created and managed without any interaction from the system administrators.**

Please note that changes are not instantaneous: it may take about an hour for the changes to propagate throughout the entire HPC infrastructure.

To change the group of a directory and it's underlying directories and files, you can use:

```
chgrp -R groupname directory
```

Joining an existing group

1. Get the group name you want to belong to.
2. Go to <https://account.vscentrum.be/django/group/new> and fill in the section named "Join

group". You will be asked to fill in the group name and a message for the moderator of the group, where you identify yourself. This should look something like in the image below.

3. After clicking the submit button, a message will be sent to the moderator of the group, who will either approve or deny the request. You will be a member of the group shortly after the group moderator approves your request.



View Account	Edit Account	View Groups	New/Join Group	Edit Group	New/Join VO	Log Out
--------------	--------------	-------------	----------------	------------	-------------	---------

New/Join Group

⚠ Your institute is Gent, which means all the groups from your institute start with 'g'. Groups starting with any other letter are from other institutes. Join them at your own risk.

Join group

Select a group to join. The groupmoderator will be sent your message (including your full name and VSC ID) and asked to confirm your request.

Group *

Message *

Submit

Creating a new group

1. Go to <https://account.vscentrum.be/django/group/new> and scroll down to the section "Request new group". This should look something like in the image below.
2. Fill out the group name. This cannot contain spaces.
3. Put a description of your group in the "Info" field.
4. You will now be a member and moderator of your newly created group.

[View Account](#)[Edit Account](#)[View Groups](#)[New/Join Group](#)[Edit Group](#)[New/Join VO](#)[Log Out](#)

New/Join Group

⚠ Your institute is Gent, which means all the groups from your institute start with 'g'. Groups starting with any other letter are from other institutes. Join them at your own risk.

-- scroll down --

Create new group

⚠ We will automatically prepend the letter 'g' to your groupname.

Groupname *

Info

Managing a group

Group moderators can go to <https://account.vscentrum.be/django/group/edit> to manage their group (see the image below). Moderators can invite and remove members. They can also promote other members to moderator and remove other moderators.

[View Account](#)[Edit Account](#)[View Groups](#)[New/Join Group](#)[Edit Group](#)[New/Join VO](#)[Log Out](#)

Edit group

General information

Group: example

Vsc_id: example

Vsc_id_number: 1234567

Status: active

Manage members

Members *

Remove members by clicking the ✕ in front of their names.

✕ vsc40000 (John Smith)

Invited users

Invite new accounts by clicking in the field below and start typing their vsc id.

Selected members are already invited.

Remove invites by clicking the ✕ in front of their names.

Inspecting groups

You can get details about the current state of groups on the HPC infrastructure with the following command (`example` is the name of the group we want to inspect):

```
$ getent group example
example:*:1234567:vsc40001,vsc40002,vsc40003
```

We can see that the VSC id number is 1234567 and that there are three members in the group: `vsc40001`, `vsc40002` and `vsc40003`.

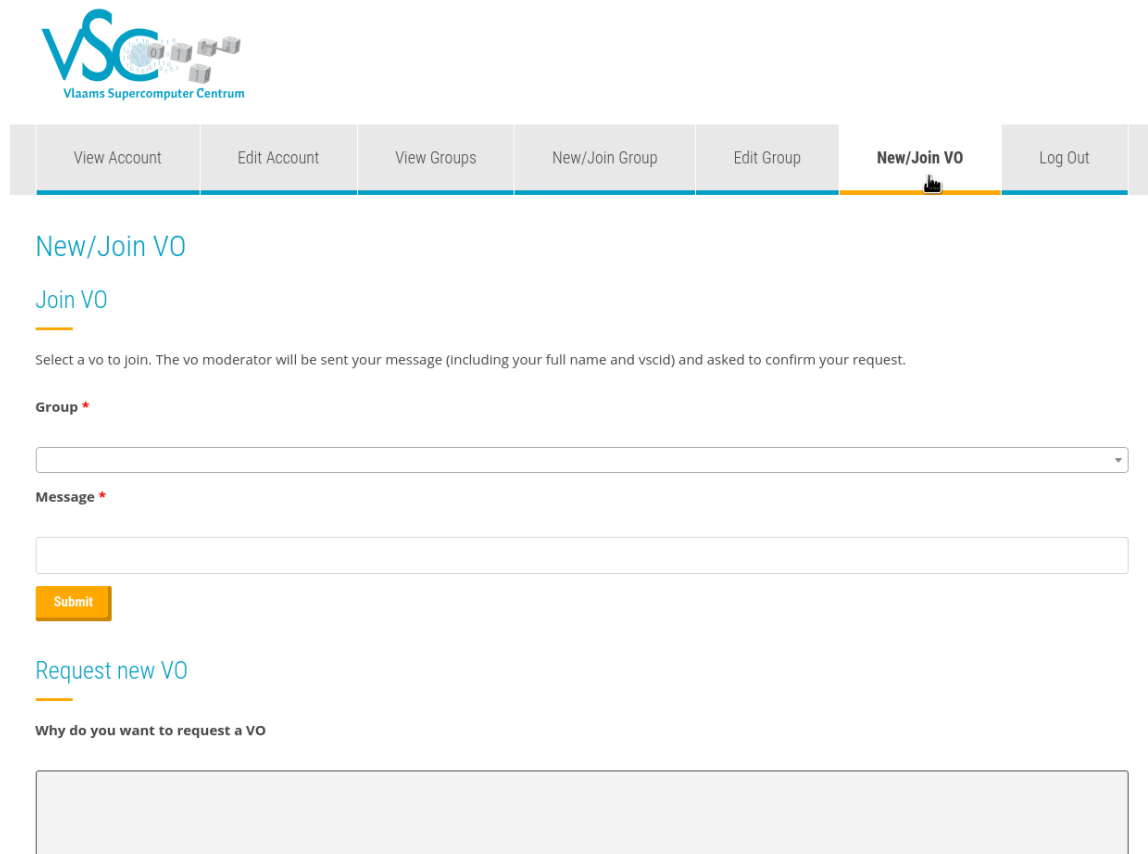
Virtual Organisations

A Virtual Organisation (VO) is a special type of group. You can only be a member of one single

VO at a time (or not be in a VO at all). Being in a VO allows for larger storage quota to be obtained (but these requests should be well-motivated).

Joining an existing VO

1. Get the VO id of the research group you belong to (this id is formed by the letters `gvo`, followed by 5 digits).
2. Go to <https://account.vscentrum.be/django/vo/join> and fill in the section named "Join VO". You will be asked to fill in the VO id and a message for the moderator of the VO, where you identify yourself. This should look something like in the image below.
3. After clicking the submit button, a message will be sent to the moderator of the VO, who will either approve or deny the request.



The screenshot shows the VSCentrum logo at the top left. Below it is a navigation bar with buttons: View Account, Edit Account, View Groups, New/Join Group, Edit Group, New/Join VO (highlighted with a mouse cursor), and Log Out. The main content area is titled 'New/Join VO' and 'Join VO'. It contains a form with a 'Group' dropdown menu and a 'Message' text area. A yellow 'Submit' button is at the bottom of the form. Below the form, there is a section titled 'Request new VO' with a sub-header 'Why do you want to request a VO' and a large empty text box for input.

Creating a new VO

1. Go to <https://account.vscentrum.be/django/vo/new> and scroll down to the section "Request new VO". This should look something like in the image below.
2. Fill why you want to request a VO.
3. Fill out the both the internal and public VO name. These cannot contain spaces, and should be 8-10 characters long. For example, `genome25` is a valid VO name.

4. Fill out the rest of the form and press submit. This will send a message to the HPC administrators, who will then either approve or deny the request.
5. If the request is approved, you will now be a member and moderator of your newly created VO.



View Account	Edit Account	View Groups	New/Join Group	Edit Group	New/Join VO	View VO	Edit VO	Reservations	Log Out
-----------------	-----------------	----------------	---------------------------	---------------	----------------	------------	------------	--------------	------------

--- scroll down ---

Create new group

⚠ We will automatically prepend the letter 'g' to your groupname.

Groupname *

Info

Submit

Requesting more storage space

If you're a moderator of a VO, you can request additional quota for the VO and its members.

1. Go to <https://account.vscentrum.be/django/vo/edit> and scroll down to "Request additional quota". See the image below to see how this looks.
2. Fill out how much *additional* storage you want. In the screenshot below, we're asking for 500 GiB extra space for `VSC_DATA`, and for 1 TiB extra space on `VSC_SCRATCH_KYUKON`.
3. Add a comment explaining why you need additional storage space and submit the form.

4. An HPC administrator will review your request and approve or deny it.



View Account	Edit Account	View Groups	New/Join Group	Edit Group	New/Join VO	View VO	Edit VO	Reservations	Log Out
-----------------	-----------------	----------------	-------------------	---------------	----------------	------------	--------------------	--------------	------------

Edit gvo00020

General information

VO: gvo00020
Institute: Gent
Vsc_id: gvo00020
Vsc_id_number: 2640143
Status: active

--- scroll down ---

Request additional quota

⚠ Every value entered is an increase to your current quota. Note that we might grant less quota than requested. This is looked at on a case by case basis. The quota is bounded by the available storage space so we can not guarantee you will be able to use all of your quota.

Size quota

VSC_DATA	74.85% (2.59TiB/3.46TiB)	+ 500	GiB
VSC_DATA_SHARED	0.0% (0B/2.0GiB)	+ 0	GiB
VSC_SCRATCH_KYUKON	38.93% (3.71TiB/9.52TiB)	+ 1	TiB
VSC_SCRATCH_PHANPY	22.75% (2.17TiB/9.52TiB)	+ 0	GiB

Setting per-member VO quota

VO moderators can tweak how much of the VO quota each member can use. By default, this is set to 50% for each user, but the moderator can change this: it is possible to give a particular user more than half of the VO quota (for example 80%), or significantly less (for example 10%).

Note that the total percentage can be above 100%: the percentages the moderator allocates per user are *the maximum percentages* of storage users can use.

1. Go to <https://account.vscentrum.be/django/vo/edit> and scroll down to "Manage per-member quota share". See the image below to see how this looks.
2. Fill out how much percent of the space you want each user to be able to use. Note that the total can be above 100%. In the screenshot below, there are four users. Alice and Bob can use up to 50% of the space, Carl can use up to 75% of the space, and Dave can only use

10% of the space. So in total, 185% of the space has been assigned, but of course only 100% can actually be used.



View Account	Edit Account	View Groups	New/Join Group	Edit Group	New/Join VO	View VO	Edit VO	Reservations	Log Out
-----------------	-----------------	----------------	-------------------	---------------	----------------	------------	--------------------	--------------	------------

Edit gvo00020

General information

VO: gvo00020
Institute: Gent
Vsc_id: gvo00020
Vsc_id_number: 2640143
Status: active

--- scroll down ---

Manage per-member quota share

⚠ You are able set a per user quota share of more then 100% (oversubscription), members are not guaranteed to be able to fill their allocated quota. The total VO quota is still an upper bound for individual usage.

VSC_DATA

Total used: 185.00%
Total assigned: 185.00%

vsc40001 (Alice)	0.00% (3.25MiB/4.76TiB)	50	%
vsc40002 (Bob)	0.00% (0B/4.76TiB)	50	%
vsc40003 (Carl)	30.33% (2.17TiB/7.14TiB)	75	%
vsc40004 (Dave)	0.00% (0B/4.76TiB)	10	%

VO directories

When you're a member of a VO, there will be some additional directories on each of the shared filesystems available:

VO scratch (`$VSC_SCRATCH_VO`): A directory on the shared *scratch* filesystem shared by the members of your VO, where additional storage quota can be provided (see [the section on requesting more storage space](#)). You can use this as an alternative to your personal `$VSC_SCRATCH` directory (see [the section on your scratch space](#)).

VO data (`$VSC_DATA_VO`): A directory on the shared *data* filesystem shared by the members of your VO, where additional storage quota can be provided (see [the section on requesting more storage space](#)). You can use this as an alternative to your personal `$VSC_DATA` directory (see

[the section on your data directory](#)).

If you put `_USER` after each of these variable names, you can see your personal folder in these filesystems. For example: `$VSC_DATA_VO_USER` is your personal folder in your VO data filesystem (this is equivalent to `$VSC_DATA_VO/$USER`), and analogous for `$VSC_SCRATCH_VO_USER`.