# Tutorial Compute Canada

This document is designed to help you get started with Compute Canada. It covers how to connect to the system, manage your files, and run code efficiently. Most of the information presented here is based on the official Compute Canada documentation. I also recommend checking out this excellent tutorial, which provides a clear and practical overview.

## Contents




**Author:** Côme Beauquier beauquiercome@gmail.com

# 1 Connect to Compute Canada (SSH Key)

The first step is to connect your computer to a Compute Canada server. To do this, we will use an SSH key. I recommend watching the following video, which clearly explains how to use an SSH key to connect your computer to Compute Canada. If the video doesn't work, don't worry. Let's go through the steps together.

1. **Open a command terminal**. For example, use PowerShell on Windows, or Terminal on macOS/Linux. For remember, here the common terminal commands.

2. **Create an SSH key** with the following command:

   ```
   ssh-keygen -t ed25519
   ```

   This command will generate two files in a folder named /.ssh:

   - your private key (used for authentication)
   - your public key (used to authorize access)

3. **Get your public key**. You can open the '.pub' file in a text editor, or use the following command in your terminal (in the folder /.ssh):

   ```
   cat yourfile.pub
   ```

   Copy the entire public key (it will start with 'ssh-ed25519').

4. **Paste your key into Compute Canada**. Go to *My Account* > *Manage SSH Keys* on the Compute Canada website, and paste the public key into the form.

5. **Connect your terminal to a Compute Canada server**. Compute Canada has five main servers: Béluga, Narval, Cedar, Graham, and Niagara. Use the following command to connect using your username and the name of the chosen server:

   ```
   ssh username@servername.computecanada.ca
   ```

   If the above command doesn't work, you can try connecting by explicitly providing the path to your private key:

   ```
   ssh -i path/to/privatekey username@servername.computecanada.ca
   ```

You are now connected to Compute Canada through your terminal, congratulations! I also recommend enrolling your Compute Canada account in DUO (multi-factor authentication). You can do this under *My Account* > *Multifactor Authentication Management*. Without this, you may encounter login issues in the future.

Once you are connected, you will see that there are three directories in your home folder:

- **scratch**: A high-performance working space, ideal for running jobs. However, data stored here will be deleted after 60 days.
- **projects**: Used to store and share data with collaborators.
- **nearline**: Intended for long-term data storage.

You can also create new folders in your home directory for small personal files, such as scripts.

# 2 Managing Files on the Compute Canada Server

There are two main ways to manage files on Compute Canada: using the terminal, or using software such as Globus or WinSCP. Let's explore both methods.

## 2.1 Using the Terminal

To transfer files between your local computer and the server, you can use the `scp` (secure copy) command.

1. **Open a terminal** without connecting to Compute Canada.
2. **Download a file or folder** from Compute Canada using:

```
scp username@servername.computecanada.ca:path/to/fileOrFolder
"C:\path\to\yourLocalStorage"
```

3. **Upload a file or folder** to Compute Canada using:

```
scp "C:\path\to\yourFileOrFolder"
username@servername.computecanada.ca:path/to/remoteStorage
```

**Note:**

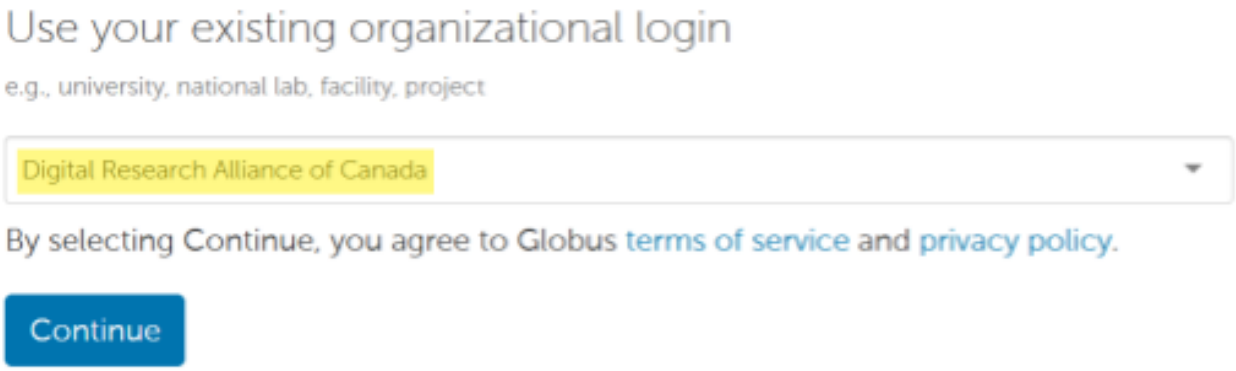
- The `scp` command may not work properly in some terminals such as PowerShell. I personally used Git Bash, which works well.
- The path on Compute Canada should start from the child directory inside your home folder. Here's an example:

```
$ scp -r come@cedar.computecanada.ca:projects/def-nbl/come/codes "C:\Users\beauq
\Desktop\Test"
```

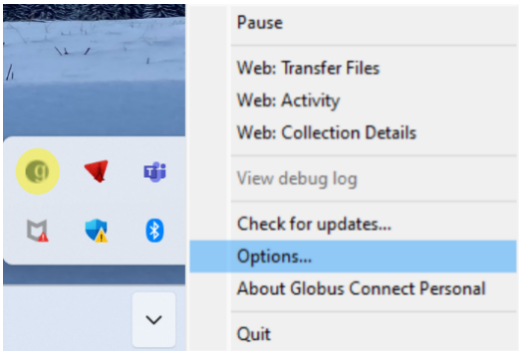
## 2.2 Using a software

In this example, we'll use Globus, but the process should be similar for WinSCP or other software.

1. **Set up your Globus account** here, using your Compute Canada login, and **download Globus Connect Personal**.

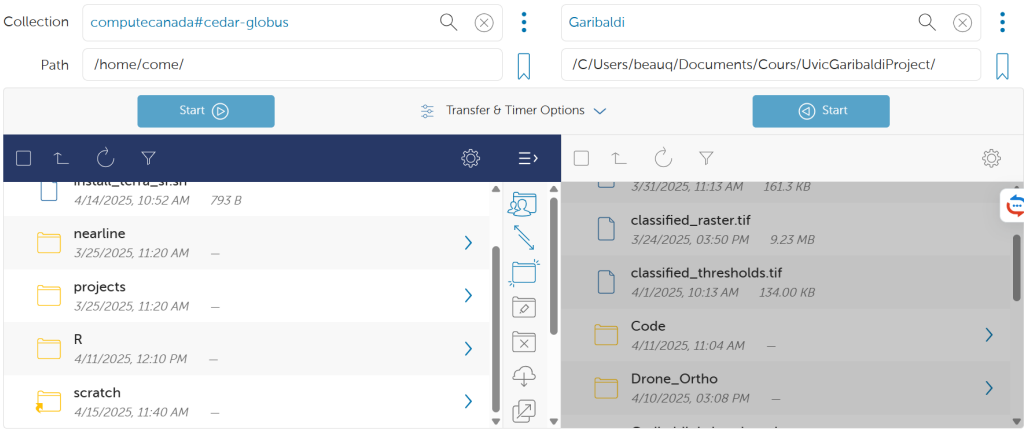


2. **Connect your working folders**: By default, Globus Connect Personal is linked to your `Documents` folder. You can add additional folders by right-clicking the Globus icon and selecting "Options":




   Then, in the options menu, click the "+" button to add the folder(s) you want to use.

3. **Transfer your data**: Select your Compute Canada server and your local folder. Choose the files or folders to transfer and click "Start" to begin the transfer.




# 3 Running R Code on Compute Canada

Before running R code on Compute Canada, let's first go over how to install R packages.

## 3.1 Installing Packages

1. **Open a terminal** and connect to a Compute Canada server using:

```
ssh username@servername.computecanada.ca
```

2. **Start an R session** with:

```
R
```

   Choose the R module (version) you want to work with.

3. **Install your package** using:

```
install.packages("yourPackage", repos =
"https://cloud.r-project.org")
```

This works for most standard packages, but not all. For example, I encountered issues when trying to install the ‘terra‘ package.

Let’s now look at how I installed the ‘terra‘ package — this process can also be applied to other complex packages.

## 3.2 Creating a Job Script

To run R code on Compute Canada, we need to create a job script that instructs the server how to run the script. This script is written in Bash.

- **Open a terminal and connect** to Compute Canada.
- **Create a folder** to store your job script, for example in your ‘projects‘ or ‘scratch‘ directory:

```
mkdir my_job
```

- **Create the job script file**:

```
nano job_script.sh
```

This opens a text editor where you can write your script.

- **Write a basic job script:**

```
#!/bin/bash
#SBATCH --time=00:15:00
#SBATCH --account=def-user
echo 'hello world'
```

Here’s what each line does:

  - ‘–time‘: Sets the maximum job duration.
  - ‘–account‘: Specifies your project ID (e.g., ‘def-nbl‘) so Compute Canada knows which project the job belongs to.

Save the file with ‘Ctrl + S‘ and exit with ‘Ctrl + X‘.

- **Submit your job:**

```
sbatch job_script.sh
```

- **Monitor your job:**

```
squeue -u yourUsername
```

When the job finishes, an ‘.out‘ file will appear in your folder. You can view it using:

```
cat yourFile.out
```

You should see the message “hello world.”

- **Example of a more advanced job script:**

```
#!/bin/bash
#SBATCH --time=01:00:00
#SBATCH --account=def-user
#SBATCH --cpus-per-task=4
#SBATCH --mem=16G
#SBATCH --output=log_%j.txt
#SBATCH --error=error_%j.txt
```

Additional options:

  - ‘–cpus-per-task‘: Number of CPUs.
  - ‘–mem‘: RAM allocation.
  - ‘–output‘: Output log file.
  - ‘–error‘: Error log file.

For more information, visit the official documentation.

## 3.3 Running an R Script

In the ‘.sh‘ file, load the required R module and any dependencies. For example, to load R version 4.2.2:

```
module load StdEnv/2020 r/4.2.2
```

Then run your R script with:

```
Rscript /path/to/yourscript.r
```

Note: As with the ‘scp‘ command, paths should start after your home directory. For example:

```
Rscript /project/def-nbl/come/codes/test.R
```

Also be careful when accessing paths inside Compute Canada: use ‘/project/‘ not ‘/projects/‘.
You can now save and launch the job script as shown before.

## 3.4 Installing the ‘terra‘ Package

To install the ‘terra‘ package, I first changed the repository source. By default, R uses ‘https://cloud.r-project.org‘, which is fast and stable, but it doesn’t always work with complex packages like ‘terra‘.
Instead, use:

```
install.packages("terra", repos =
"https://mirror.csclub.uwaterloo.ca/CRAN/",
dependencies = TRUE)
```

Although this works, it can be very slow on personal computers. To make installation faster and more reliable, I ran it on Compute Canada using the following job script:

```
#!/bin/bash
#SBATCH --account=def-nbl
#SBATCH --job-name=install_r_packages
#SBATCH --output=install_r_packages_%j.log
#SBATCH --time=01:00:00
#SBATCH --cpus-per-task=4
#SBATCH --mem=16G

# Load required modules
module load StdEnv/2020 gcc/9.3.0 udunits/2.2.28 \
gdal/3.5.1 r/4.2.2

# Create a folder to store the packages
mkdir -p $HOME/R/x86_64-pc-linux-gnu-library/4.2
export R_LIBS="$HOME/R/x86_64-pc-linux-gnu-library/4.2:$R_LIBS"

# Install packages
R -e "install.packages(c('terra', 'sf'),
repos='https://mirror.csclub.uwaterloo.ca/CRAN/',
dependencies=TRUE)"
```

This script requests 4 CPUs and 16 GB of RAM. The required modules are:

- **StdEnv**: Standard environment.
- **gcc**: C/C++ compiler.
- **udunits**: Unit handling library.
- **gdal**: Library for geospatial data.
- **r**: The R language itself.

Once installation completes, you can verify it in an R session by running:

```
library(terra)
```