Rossinante – User guide

Nicolas Casajus

September 13, 2021

Contents

Introduction

This tutorial presents how to use he FRB-CESAB server **Rossinante**, dedicated to medium performance scientific computing (Table 1). You can run programs coded in R, Python, Julia, C, and C++. Unlike traditional clusters, Rossinante does not have a job scheduling system (e.g. SLURM) meaning that you can launch jobs when you want (only if the server is available, see below the **Good practices** section).

Table 1: Rossinante hardware specifications

Hardware	**Specifications**
CPU	2 x Intel Xeon Gold 5218R (total 80 threads)
RAM	12 x Cells 32 Go RDIMM (total 384 GB)
Storage	8 x 960 GB SSD SATA (total 6.1 TB)
GPU	NVIDIA Quadro RTX 6000
OS	Ubuntu Server 20.04 LTS (Focal Fossa)

When do you need to use Rossinante?

- You need to analyse large datasets (RAM operations), and/or,
- you need to repeat tasks many times (parallelization on CPU/GPU).

What are the available software?

- R 4.1 (and RStudio Server 1.4)
- Python 3.8 (and its development environment)
- Julia 1.6
- LaTeX 3.14 and Pandoc 2.5
- git 2.25
- FFmpeg (transcoding multimedia files)
- ImageMagick (image manipulation program)
- Poppler (utility library for PDF)
- Spatial tools (GDAL, GEOS, PROJ4)

And some useful utilities:

- `htop`: CPU and RAM monitoring tool
- `nvtop`: NVIDIA GPU monitoring tool
- `nano`, `vi`, and `vim`: text editors
- `screen` and `tmux`: terminal multiplexers
- `tree`: recursive directory listing program
- `curl` and `wget`: download managers
- `zip` and `unzip`: ZIP files managers

Can you do what you want on Rossinante?

No! Rossinante has only one administrator, Nicolas Casajus[1]. Regular users, like you, have only access to a personal directory, `/home/you/` and to a shared directory `/home/cesab/`. You can store your files in one

[1] nicolas[dot]casajus[at]fondationbiodiversite[dot]fr

of these two folders. But keep in mind that only you have access to your personal space whereas everybody can access the shared space.

Important – Rossinante is **not a storage server**. Its 6 TB storage are shared among all users. You can store large datasets on your personal space to run yours analyses, but once you've finished, please remove your data.

If you need to use a non-installed software, please contact the administrator. Note that each user has a personal R library in which he can install every R package he wants (independently of other users).

1 First steps

When you connect to Rossinante for the first time, you'll be asked to change your password. This first connection is made under the SSH protocol (Secure Shell). This protocol is a cryptographic network protocol that allows you to securely access a remote computer over an unsecured network.

For this tutorial, let's say:

- your name is `Jane DOE`
- your user name on your laptop is `jane`
- your laptop name is `laptop`
- your user name on Rossinante is `jdoe`
- the public IP address[2] of Rossinante is `92.168.45.3`
- the port of the SSH server is `22`

1.1 SSH connection

To open an SSH connection on Unix-based OS (macOS and Linux)[3], open a terminal session and run:

```
# SSH connection to Rossinante ----
jane@laptop:~$ ssh -p 22 jdoe@92.168.45.3
```

You'll be asked to change your password. Enter the old password and set your new password (twice).

Then, your prompt will look like:

```
jdoe@rossinante:~$
```

This means that you are now connected to Rossinante under the user name `jdoe`. You can check your current directory with the command `pwd`:

```
# Print working (current) directory ----
jdoe@rossinante:~$ pwd
## /home/jdoe
```

[2] When you are inside the CESAB, you can use the local IP address of the server.

[3] On Windows, you'll need to install the software Putty (https://www.chiark.greenend.org.uk/~sgtatham/putty/).

To stop the SSH connection, use the command `exit` (or `logout`):

```
jdoe@rossinante:~$ exit
## Connection to 92.168.45.3 closed.
```

1.2 SSH configuration file

It can be painful to remember the IP address and the SSH port of Rossinante, especially if you use several servers. Fortunately you can store Rossinante credentials (except your password) and SSH connection information in a special file located on **your laptop** (not in the server): `~/.ssh/config`.

To create this `config` file, follow these steps:

```
# Navigate to your home directory (symbolized by ~) ----
jane@laptop:~$ cd ~

# Create a new hidden folder ----
jane@laptop:~$ mkdir .ssh

# Change folder permissions ----
# (only Jane can read, write, and execute this folder) ----
jane@laptop:~$ chmod 700 .ssh

# Create the (empty) SSH config file ----
jane@laptop:~$ touch .ssh/config

# Change config file permissions ----
# (only Jane can read and write this file) ----
jane@laptop:~$ chmod 600 ~/.ssh/config

# Open the SSH config file with the CLI editor nano ----
jane@laptop:~$ nano ~/.ssh/config
```

Now add the follow lines in the SSH Config file:

```
## Host rossinante
##     HostName 92.168.45.3
##     Port 22
##     User jdoe
```

To save changes press `CTRL + X` and `Y` (or `O` if the language of your system is French) and press `Enter`.

You can now connect to Rossinante as follow:

```
jane@laptop:~$ ssh rossinante
```

1.3 Generating SSH keys

Though SSH supports password-based authentication, it is generally recommended that you use SSH keys instead. SSH keys are a more secure method of logging into an SSH server, because they are not vulnerable

to common brute-force password hacking attacks. Generating an SSH key pair creates two long strings of characters: a public and a private key. You can place the public key on any server, and then connect to the server using an SSH client that has access to the private key.

Let's create a new SSH keys pair using the cryptosystem `RSA` and a key size of `4096` bits.

```
# Create a new SSH key pair ----
jane@laptop:~$ ssh-keygen -f ~/.ssh/id_rossinante -t rsa -b 4096 -C "jane.doe@mail.com"
```

If you want you can add a passphrase to increase the security of your key pair but each time you will connect to Rossinante you will be asked to enter it.

This SSH key pair has been stored in `~/.ssh/`.

```
# Content of the ~/.ssh folder ----
jane@laptop:~$ ls ~/.ssh
## config       id_rossinante       id_rossinante.pub
```

The private key is `id_rossinante` and the public one `id_rossinante.pub`. Nobody (except you) can have access to your private key. So we need to change the permissions of this file.

```
# Change private key permissions ----
# (only Jane can read this file) ----
jane@laptop:~$ chmod 400 ~/.ssh/id_rossinante
```

On the opposite your public can be deployed everywhere. In our case, we will store it on the Rossinante server.

```
# Copying public key to Rossinante ----
jane@laptop:~$ ssh-copy-id -i ~/.ssh/id_rossinante.pub rossinante
```

Now we can connect to Rossinante without entering password (except if you have added a passphrase to your SSH key pair).

```
jane@laptop:~$ ssh rossinante
```

The first time you use your new SSH key pair you will see:

```
## The authenticity of host '[92.168.45.3]:22' can't be established.
## RSA key fingerprint is ...
## Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Just write `yes` and press `Enter`.

Our public key on Rossinante has been stored under the name `authorized_keys`.

```
# Content of the ~/.ssh folder ----
jdoe@rossinante:~$ ls ~/.ssh
## authorized_keys
```

2 Sending files

...

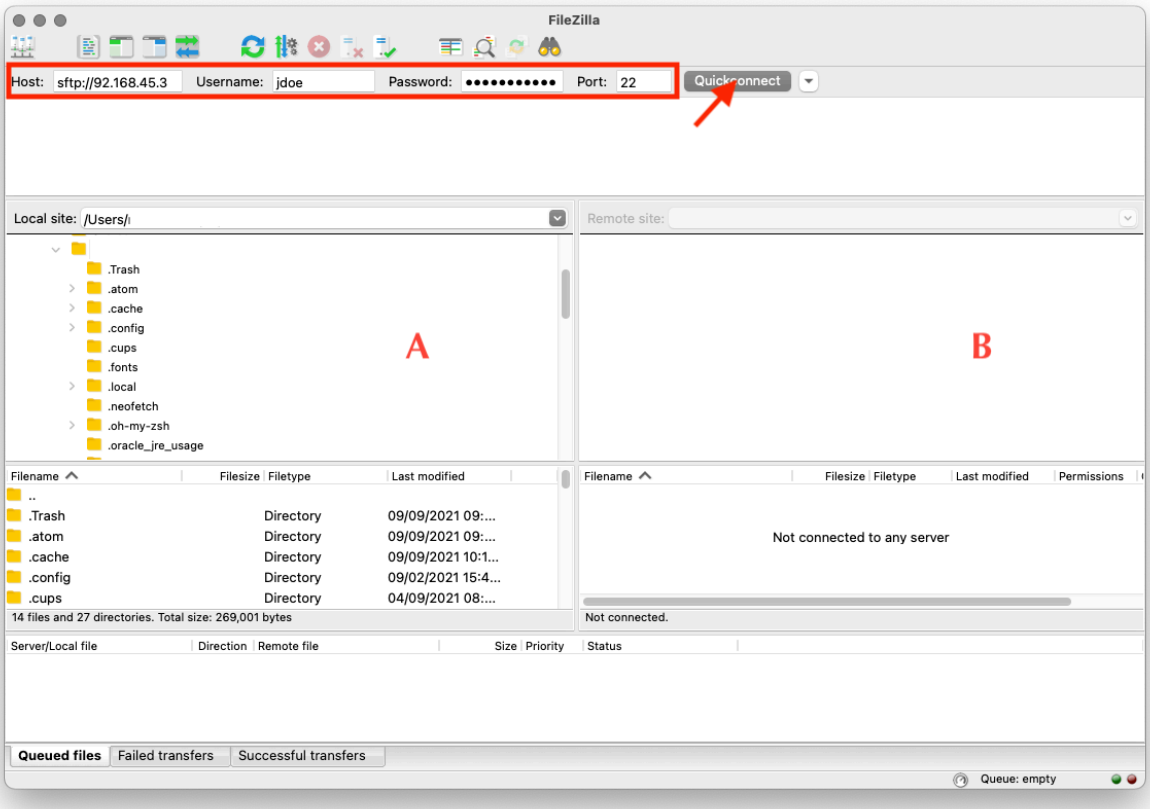
2.1 sFTP

The easiest way to transfer files from your laptop to Rossinante (or vice versa) is by using the sFTP (Secure File Transfer Protocol) protocol. Filezilla client is a freeware that supports this protocol.

You will need to define these following parameters:

- Host: `sftp://92.168.45.3`
- Username: `jdoe`
- Password: your Rossinante user's password
- Port: `22`

To make the connection, click on **Quick connect**.



The left panel (A) lists your local folders/files. The right panel (B) shows the content of your personal directory on Rossinante.

To send local files to Rossinante, first select the directory in B to send these files in. Then select the files in A, right click, and click on Upload.

To send remote files to your laptop, first select the directory in A to send these files in. Then select the files in B, right click, and click on Download.

Important: If your project is tracked by git do not use this method. See section 2.3.

2.2 scp

An alternative way to transfer files is by using the command `scp` that allows to copy files using the SSH protocol.

Let's say we want to copy the local file **script.R**, located in the **Documents/** folder, to Rossinante (in the folder **projects/** in our personal directory). We will use `scp` as follow:

```
# Send a file from local to Rossinante ----
scp ~/Documents/script.R rossinante:projects/
```

If we want to download a file from Rossinante:

```
# Send a file from Rossinante to local ----
scp rossinante:projects/script.R ~/Documents/
```

To copy folders we will add the option `-r` (for recursive):

```
# Send a folder from local to Rossinante ----
scp -r ~/Documents/project_1 rossinante:projects/

# Send a folder from Rossinante to local ----
scp -r rossinante:projects/project_1 ~/Documents/
```

N.B. If you want you can also use the command `rsync`.

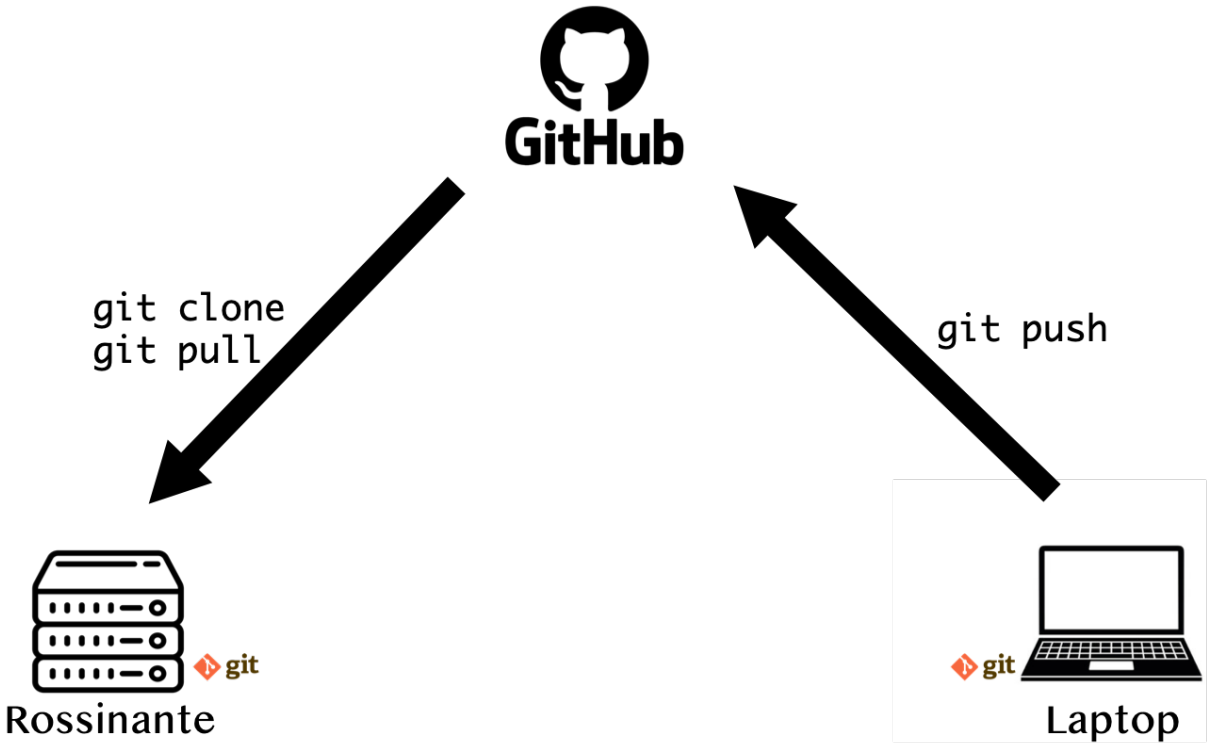
2.3 Git and GitHub

If your project is tracked by the versioning system control **git**, you may prefer sending files through GitHub (or GitLab).

This method has the advantage of keeping your project tracked by git, synchronized with GitHub, and backed up on Rossinante.

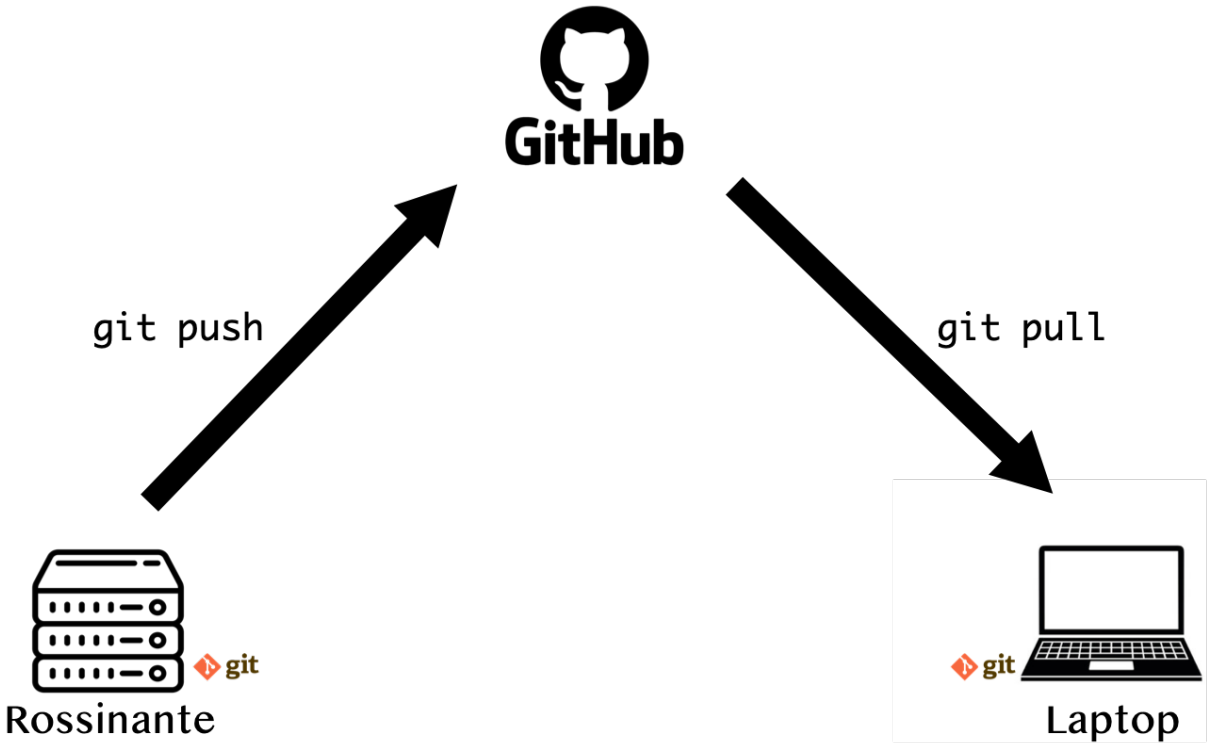
The workflow is the following:

1. On your laptop, commit changes
2. Then push changes to your repository on GitHub
3. Connect to Rossinante via SSH
4. Clone the GitHub repository on Rossinante or pull changes if your project is already cloned
5. Run analysis on Rossinante



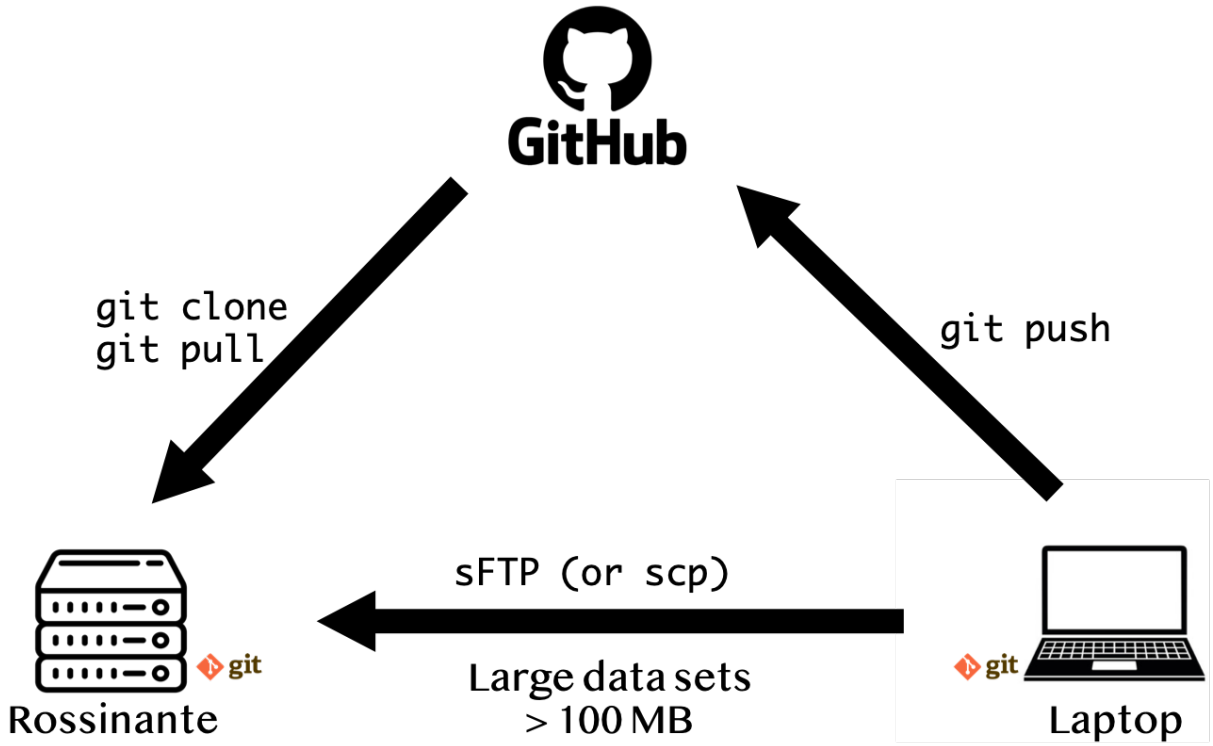
Once your analysis is finished, you can:

1. Commit changes
2. Push changes to your repository on GitHub
3. On your laptop, pull changes



At this stage, the project on your laptop, GitHub and Rossinante is in the same state.

Important: GitHub does not accept file > 100MB. If your project contains large datasets (added in the `.gitignore`), you need to send these files through sFTP or SCP.



3 Git credentials

...

3.1 Configuring git

When you first use **git** on Rossinante, you need to set your user name and email (required for commits). Run the following lines:

```
# Connection to Rossinante ----
jane@laptop:~$ ssh rossinante

# Set Git user name (globally) ----
jdoe@rossinante:~$ git config --global user.name "Jane Doe"
jdoe@rossinante:~$ git config --global user.email jane.doe@mail.com
```

3.2 GitHub SSH key

If you want to communicate with GitHub through the SSH protocol (recommended) you need to generate a new SSH key pair (different from the one used to connect with Rossinante).

```
jdoe@rossinante:~$ ssh -T git@github.com
## Hi janedoe! You've successfully authenticated, but GitHub does not provide
## shell access.
```

4 RStudio Server

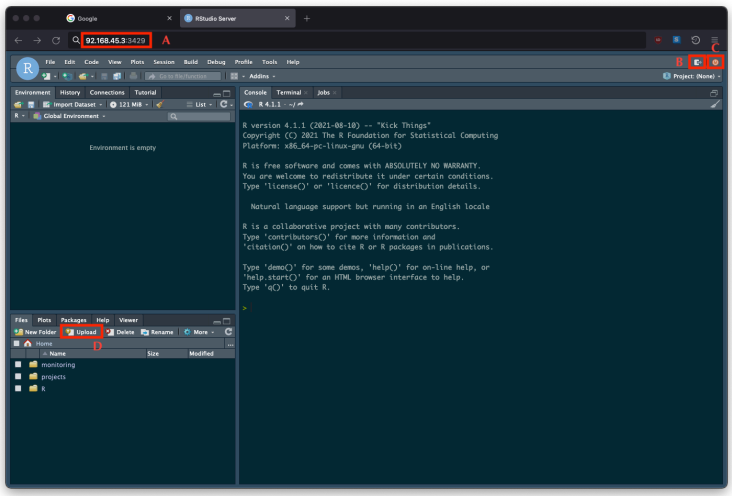
...

4.1 Connection

Open a web browser (Firefox, Chrome, etc.) and enter the URL of the RStudio Server:

```
92.168.45.3:3528
```

After entering your Rossinante login information, you are connected to an RStudio Server.



You can now use this interface as the one you knows (RStudio Desktop).

```
rm -rf ~/.local/share/rstudio/sessions/active/session-*
```

4.2 Installing packages

...

4.3 R in the terminal

...

- screen

5 Python

...