The Knowledge

Geographic Data Science Lab

2020-06-09

Contents

Chapter 1

The Knowledge

In preparation to be able to stay productive while having to work fully remote, this document presents a few things jotted down in one place to help with setups, etc.

The first step is a checklist everyone should go through:

1. Do you have a computer to work on at home?
2. Are *all* your relevant files accessible from home?
3. Do you have a webcam, mic and speakers/headphones?
4. Are you able to connect to computers on campus? This involves being setup with the University VPN?

These are the basic elements you will require, so if the answer to any of the above is no, please speak with your PhD supervisor.

Chapter 2

Remote Working

2.1 What is Remote Working?

"Remote work refers to organizational work that is performed outside of the normal organizational confines of space and time. The term telecommuting refers to the substitution of communications capabilities for travel to a central work location. Office automation technology permits many office workers to be potential telecommuters in that their work can be performed remotely with computer and communications support" (Olson, 1983)

Some of the first trials of remote working date back to the eighties as reported by Olson (1983). Since then, the spread and development of Information and Communication Technologies have brought about a significant increase in the popularity of remote work. Nowadays, it is possible to find fully remote jobs advertised particularly in IT and data science.

The current pandemic has forced millions of workers at home, making remote work a necessity rather than an option. An early study on COVID-19 and remote work reported that in the US the fraction of workers who switched to working from home is about 34.1%, while 14.6% were already working from home pre-COVID-19 (Brynjolfsson et al., 2020).

Not all types of work are suitable to be performed from home (Holgersen et al., 2020), but one of the impacts of these dramatic circumstances can be a further increase of the remote work practice.

Computationally intensive tasks can be easily approached in a remote setting by accessing computing resources through the network. This document will guide you in accessing servers located in the Geographic Data Science Lab to perform heavy computations [1].

[1] An other option which is widely implemented by companies, is the use of cloud computing infrastructures such as Amazon Web Services (AWS), Salesforce's CRM system, Microsoft

2.2 Advantages and Disadvantages of Remote Work

One of the most direct consequences of remote work is changes in commuting behaviours, bringing about time saving and a potential reduction in traffic congestion and air pollution. These were among the main points stressed by the early advocates for remote working, but researches in transportation studies have shown conflicting results. Although reductions in number and length of commuting trips is reported in some of the earlies studies (Kitamura et al., 1991; Olson, 1983), more recent research shows that the expectation that home-based telework reduces travel is not so apparent (e Silva and Melo, 2018) and time saving seems not to be a major pull factor (Bailey and Kurland, 2002).

The higher flexibility afforded by remote working is mentioned as an advantage, particularly for those who would have not taken part of the workforce without such settings because of caring committments (Olson, 1983).

Higher productivity of remote workers has been reported in some studies. However, it has to be noted that productivity and concentration at home are strongly dependent from environmental conditions (Bailey and Kurland, 2002). Inequality in living conditions is clearly affecting the ability to work from home during the coronavirus pandemic.

One of the most cited drawbacks of remote working is professional and social isolation (Bailey and Kurland, 2002), which can be also seen as making more difficoult collaborative work and collective workers actions.

2.3 Tips

Here a collection of tips that have been shared on the internet on how to avoid burn out and be effective while working from home:

- Covid-19: How to foster healthy home working
- Tips on working from home with children
- Working from home? Why detachment is crucial for mental health

Azure.

Chapter 3

VPN

3.1 *What is a VPN?*

A VPN (virtual private network) connects a machine that lies outside of the university (ie. outside the firewall) to the internal network. When the VPN is running, your network traffic (e.g. Internet) is routed through the university in the same way as if the computer was on your work desk. This enables you to:

- Access journal websites like you would inside the university
- Access network drives (e.g. M Drive etc) - but be careful when transferring big files
- Access servers (e.g. over the terminal / command line / ftp)

3.2 *How can I setup the VPN for Liverpool?*

This document describes how to set up a VPN.

To access the VPN service:

1. Register

First you will need to submit a request to register for the VPN service via CSD. CSD will need you to explain why you require VPN access and what you intend to do with it.

2. Download the VPN Client

Once your registration is confirmed you will need to download and install a VPN Client.

3. Open the client and connect

Finally you will need to connect to the VPN using your VPN client. Below we provide instructions on how to accomplish this on Windows, Mac and Ubuntu.

3.3 Windows and Mac

You must make sure you are registered to access the VPN service first. Once you are registered you can download and install the GlobalProtect VPN Client to connect to the University network. GlobalProtect is compatible with Windows 10 and Macs. Please access admin rights before attempting to download the VPN Client on your MWS PC. It is not possible to use GlobalProtect to connect to the University network on a mobile or tablet device.

To install the Client:

1) Access Admin Rights on you PC.
2) Visit https://vpn.liv.ac.uk
3) Enter your University username and password to login to the VPN portal.
4) Click the appropriate link to download the required version of the VPN client - Windows 32 bit, Windows 64 bit, or Mac OS. (To check which version you require, see your system properties on your device)
5) Once the file has downloaded, double-click to run the installation.
6) Follow the steps through the installation wizard, accepting the default options.
7) Once installed you will see the GlobalProtect "globe" icon appear in the system tray (bottom right, near the clock). It is a globe and it will have a red x on it, showing that it is not currently connected.
8) Double-click on the GlobalProtectglobe icon in the system tray. In the window that opens, enter the following:
    - Username: enter your University username
    - Password: enter your University password
    - Portal: vpn.liv.ac.uk
9) Click Apply.

The GlobalProtect VPN client will then automatically connect to the University network - the red cross should disappear form the icon in the system tray.

You can close the window: the client will stay connected.

3.3.1 To connect and disconnect the client

Once the GlobalConnect VPN client has been installed, the icon will remain in your system tray.

To connect the VPN right click the GlobalProtect icon in the system tray and choose Connect. When you have finished and want to disconnect the VPN, right click on the icon and choose Disconnect.

How to allow third party applications - like Global Protect - to install on a Mac

1) Open System Preferences and click Security & Privacy
2) Select the General tab
3) Click the lock in the lower left-hand corner
4) Enter your computer username and password, then select Unlock

3.4 Ubuntu

The University of Liverpool provides a guide for setting up VPN on Linux, tested with Ubuntu 14.04, 16.04, 18.04 and Centos 7. The guide recommends the installation of the VPN Client VPNc:

```
sudo apt-get install network-manager-vpnc
sudo apt-get install network-manager-vpnc-gnome
```

Alternatively yum can be used to install the software:

```
sudo yum install NetworkManager-vpnc
sudo yum install NetworkManager-vpnc-gnome
```

To recap the remaining steps from the UoL guide:

1) First you will need to download the UoL VPN configuration file: VPN.PCF
2) In the top right corner of your desktop there is a network icon. Click on this icon, then choose:

   VPN Connections > Configure VPN

1) Click “Add” and then “Select Import” in order to locate your *.pcf file.
2) Next enter your MWS username in the User Name field.
3) Click the IPv4 Settings tab.
4) If left blank fill in the Group password field using the group password specified within the *.pcf file.
5) Add DNS servers:
   - 138.253.110.103
   - 138.253.110.104

and search domains (if a search domain field is available):

- liv.ac.uk

- liverpool.ac.uk
- livad.liv.ac.uk

1) Click on Routes.
2) You will need one route with the following details:
    - Address: 138.253.0.0
    - Netmask: 255.255.255.0
    - gateway: 0.0.0.0
    - metric: 0
3) Tick the "Use this connection only for resources on its network" box.
4) Click Save / Apply / Close depending on the distribution.

3.4.1 To connect the client and disconnect the client:

To connect to the network, click on the icon in the top right corner of the desktop and choose

VPN Connections > University of Liverpool VPN Service

You will be promted to authenticate using your MWS password. Upon connecting the network icon will display a locked symbol. You can now use your computera as if you were physically connected to the University of Liverpool network.

To disconnect click on the network icon again and choose:

VPN Connections > Disconnect VPN

Chapter 4

SSH

4.1 What is SSH?

Secure Shell (SSH) is a cryptographic network protocol for accessing a computer over an unsecured network. It gives you secure access to a machine's command-line. Secure Shell provides strong password authentication and public key authentication, as well as encrypted data communications between two computers connecting over an open network, such as the internet. However, all computers within the University of Liverpool Network are not accessible from the open internet for security reasons. Therefore, to access a machine at the University you do not only need to be connected to the internet, but also to the Virtual Private Network (VPN) that *virtually brings you to the University of Liverpool Network*.

4.2 Connecting to a remote host

The use of SSH to connect to a remote host is performed through the following command:

```
ssh <username>@<server.ip.address>
```

If you are connecting to a server for the first time, then you may receive the following warning:

```
The authenticity of host 'hostIPaddress' cannot be established.
 DSA key fingerprint is 01:23:45:67:89:ab:cd:ef:ff:fe:dc:ba:98:76:54:32:10.
 Are you sure you want to continue connecting (yes/no)?
```

Windows users need to install an SSH client in order to access a server remotely through SSH. There are several clients available, MobaXterm is the option we advise to employ as it provides a number of useful functions for remote com-

puting in a single application. The free edition can be downloaded here. To establish a connection with your remote server you open mobaXterm and click on session and then SSH. The Remote Host is the machine you want to access which can be identified by an IP address. Here you can find a simple demo of the SSH client.

If this is your first time connecting to the server, or if the server has recently been reconfigured with a new key, then the above message is perfectly normal. You can proceed by typing yes and enter.

Once you have accessed the server the first thing to do is to change the temprorary password we assigned to your user.

```
user@host:~$ passwd
```

4.3 Unix Commands

Servers often run unix operating systems such as GNU/Linux. Unix commands are essential to perform operations from the terminal.

The following are the most frequently used commands:

4.3.1 List Files:

```
user@host:~$ ls -lh
```

In the above example two flags have been added to the ls command:

- -l List with long format, e.g., show file read/write/execute permissions.
- -h List files with readable file size, e.g. MB, GB, etc.

4.3.2 Make Directory:

A new directory can be created using the mkdir command:

```
user@host:~$ mkdir <new_directory_name>
```

4.3.3 Move:

The mv command command can be used to either rename or move files and folders:

```
user@host:~$ mv <current_filename> <new_filename>  # This is to change a file name
user@host:~$ mv <filepath> <target_directory> # This is to move a file to a folder
user@host:~$ mv -r <directory_path> <target_directory> # This is to move a folder with
```

We add the -r (recursive) flag to move a directory and all its contents (subdirectories and files).

4.3.4 Remove:

Files and folders and can be deleted using the rm (remove) command:

```
user@host:~$ rm <filepath> # To delete a file
user@host:~$ rm -r <directory> # To delete a folder and its content
```

Again, we add the -r (recursive) flag to remove a directory and all its contents (subdirectories and files).

Warning: rm is to be used with caution. There is no trash folder from which the files can be recovered. Upon using this command the files are deleted.

4.3.5 Change your Working Directory:

To change your working directory use the cd (change directory) command, specifying your target directory or .. to move back:

```
user@host:~$ cd <target_directory>
user@host:~$ cd ..
```

4.3.6 Copy:

To copy files use cp. Again the -r flag can be added to recursively copy all files and subdirectories within a directory:

```
user@host:~$ cp <filename> <filecopy>
user@host:~$ cp -r <directory> <target>
```

4.4 Command Line Editors

There exists a large number of command line editors that can be used to edit files directly within the terminal. To create a file from the terminal you can simply type the following command:

```
user@host:~$ > filename.txt
```

Below we provide instructions for using the nano editor. To open a file in nano run the following command inside the terminal:

```
user@host:~$ nano <filename>
```

Once the file opens you can move your cursor using the arrow keys, and edit content as in any text editor. You can save any changes that you have made to the file using **Ctrl + O**. To close the editor press **Ctrl + X**. Before the editor closes you will be asked if you want to save your changes. Type Y for Yes to save changes, and N for No if you want to close the editor without saving.

For an overview of nano shortcut keys you can press **Ctrl + G**, which will output the following list.

4.5 Practical

1) Use SSH to access one of the lab servers (the IP address of the server you have been assigned will be provided).
2) **Change the current temporary password with a password of your choice.**
3) Create a directory with your project name.
4) List files to check that the directory is created.
5) Change your working directory with the project directory.
6) Create a file, edit and save using nano.

Chapter 5

Files

You need to make sure that:

a) You have access to all of your files;
b) Your files are backed up so your setup is not entirely reliant on a single device;
c) Each device on which your files are copied or from which they are accessed is encrypted.

5.1 Backup/sync

The simplest and recommended way to do this at Liverpool is to keep all your files and data on your university account at OneDrive. This is part of the Office 365 Suite available from the university, you can find more info at:

`https://www.liverpool.ac.uk/csd/working-from-home/`

There are Windows and Mac clients that work relatively well (equivalent to Dropbox client).

Once you are set up, copy all your files onto your OneDrive account, which will create a copy of them in Microsoft's secure cloud. The exception is where you have data that has requirements to be managed in particular ways - e.g. only from a single machine etc; not in the cloud.

Please, be sure to speak with your PhD supervisor if you access data that may pose some challenges when moving from local machines or within the university network (remember OneDrive is in the Cloud, not the university servers!).

5.2 Encryption

Disk encryption helps protect data on your devices through converting them into an unreadable format. Deciphering the data without access to the required keys is challenging. Therefore, should your devices be lost or stolen, encrypting your devices therefore introduces an additional barrier for someone attempting to access potentially sensitive data. Please note that, as per University of Liverpool guidelines, "the security of confidential information is the responsibility of the individual member of staff or student NOT the University, nor the line manager or Head of Department". Encryption methods are platform dependent. A list of relevant guides is provided below:

1) Windows
2) Mac
3) Ubuntu
4) iOS
5) Android

5.3 File Transfer

Below we shall discuss two approaches that can be used to transfer files between two servers:

1) File Transfer Protocol (FTP);
2) Secure Copy Protocol (scp).

5.3.1 File Transfer Protocol (FTP)

If you need to move large and/or many files from a local machine to a remote server (e.g. from your laptop to a Linux machine at the lab), you can do so using a drag and drop interface with an FTP client (e.g., filezilla for Windows/Mac/Ubuntu or WinSCP for Windows). To access a remote server you will need to enter the following into respective fields within your FTP client:

- Host (Remote Server IP Address);
- Username (Your username on the remote host);
- Password (Remote host password).

5.3.2 Secure Copy Protocol (scp)

Alternatively Mac and Linux users can copy files between servers using the scp command. To copy a local file to a remote server:

```
scp <filepath> <username>@<server.ip.address>:<target_directory>
```

This command can also be use to copy a file from the remote server to your local machine:

```
scp <username>@<server.ip.address>:<filepath> <local_target_directory>/
```

As with copying file locally -r can be added to the above command to recursively copy all files within a directory. However, if a directory contains a large number of files then zip your directory before executing scp. The zip file can be unzipped using:

```
unzip <filename>.zip -d <target_directory>
```

Tar files meanwhile can be extracted using:

```
tar -xvzf <filename>.tar.gz -C <target_directory>
```

5.4 File Download

Often large datasets, etc can be downloaded from the web directly. The wget command can be used to download files from both http(s)

```
wget '<file_url>'
```

and ftp servers:

```
wget -r 'ftp://<username>:<password>@<server.ip.address>/<directory>'
```

Once the data has been downloaded we must verify that the integrity of the file. Typically the websites from which data can be downloaded provide a md5 checksum. This allows us to verify that a file has not been changed:

```
md5sum <filename>
```

5.5 Practical

5.5.1 File Download

For the second task we shall download the CIFAR-10 dataset from https://www.cs.toronto.edu/~kriz/cifar.html using the wget command.

1) First we will need to determine the url from which we can download the dataset. Visit the CIFAR-10 website, right click on the "CIFAR-10 python version", and choose the "copy link" option, which will copy the link to your clipboard.
2) Next, ssh into your remote server.
3) Type "wget" and paste the url into to the terminal by pressing (ctrl + shift + V).
4) Press enter to start the file download.
5) Upon downloading the CIFAR-10 dataset verify that the md5 checksum matches the one specified on the website.

5.5.2 File Transfer (Optional)

1) Create a file named "Test.txt" locally and enter some random text. From your local machine copy "Test.txt" to your remote server using either scp or a ftp client (e.g., using WinSCP or filezilla).
2) Upon transferring the file, ssh into the remote server and verify that the file is within the specified target directory using the ls command.
3) (Optional) Open the file using a command line editing interface from the terminal, e.g, nano or vim:

```
nano ./Test.txt
```

Chapter 6

Data Science Stack

Once you have access from home to all your files and (remote) university computers, next step is easily being able to bootstrap a full data science stack that allows you to carry out scientific work. There are several ways of achieving this, but our preferred strategy is to rely on container technology, in particular on Docker. This will allow you to rapidly install the platform and set of libraries you are familiar with in a way that can be easily reproduced and redeployed (e.g. on a remote computer on campus).

Here are a series of pages that will help you get a stack ready to go:

- `setup_docker.md`: instructions to install and get Docker up and running on different platforms
- `setup_jupyterlab.md`: instructions to run a JupyterLab server within a Docker container both on local (e.g. laptop) and remote (e.g. server) machines
- `setup_rstudio.md`: instructions to run Rstudio server within a Docker container both on local (e.g. laptop) and remote (e.g. server) machines.

Chapter 7

Docker

This document describes how to install and use Docker on different platforms.

7.1 Installation

If you are on Mac, Linux or Windows 10 Pro/Student editions, installing Docker is relatively straightforward:

- Mac
- Linux official instructions
- Windows 10 Pro/Student

It is important to note that, on Mac and Windows, Docker runs under a virtual machine so it will not use up all of the resources of your machine (conversely, it'll equate to be working on a more limited machine). This can be changed. But if you need more firepower, the idea is that you develop on your laptop and scale to a server (e.g. running out of the lab).

The steps to install Docker include:

- Obtain a copy of Docker and install it:
  - `Windows10 Pro/Enterprise`: Install Docker Desktop for Windows
  - `macOS`: Get started with Docker Desktop for Mac
- Once Docker is successfully installed, make sure to enable access to your main drive (e.g. `C:\\`):
  - `Windows10 Pro/Enterprise`: Open the preferences for Docker and click the "Shared Drives" tab; click on the drive you want to add and then "Apply"
  - `macOS`: this feature is automatically enabled

7.2 Image and Container

Docker can be seen as a tool to generate a computer within your computer (the host). This creates a working environment that employs the host's resources but follows its own internal rules.

Image and **Container** are two key concepts to understand Docker processes.

Images are files containing all the instructions to build a complete and executable version of an application, relying on the host OS kernel.

Containers are instantiations of images, meaning that they are instances of the images running in an isolated environment. The same image can be instantiated in multiple containers.

7.3 Useful Docker Commands

List available images :

```
docker image ls
```

Pull a new image:

```
docker pull image-tag
```

See what containers are running (this also shows you the ID and which port is occupying):

```
docker ps
```

The run command instantiates an image in an isolated container. The generic command is:

```
docker run [OPTIONS] IMAGE[:TAG] [COMMAND] [ARG...]
```

Options that are generally added are:

```
--detach or -d -> Run container in background and print container ID
```

If you do not add this option to the run command you can detach from a running docker session without exiting the shell, the escape sequence *Ctrl + p* followed by *Ctrl + q* can be used.

To re-attach the terminal to the docker session enter:

```
docker attach <session_id>
```

Other commonly used flags include:

```
--rm  -> Automatically remove the container when it exits
```

```
--env or -e  -> Set environment variables.
```

Environment variables in linux act as placeholders for information stored within the system that passes data to programs launched in shells. A common environment variable is *HOME* which is associated with the path of your home directory (*/home/your-user*). By adding *-e* to the run command you pass the environment variable to the container where a certain image is running. Cases when this option is necessary will be discussed.

```
--publish or -p -> Publish a container's port(s) to the host (This is key when running server-bas
```

```
--volume or -v -> Bind mount a volume
```

All other options are listed here

To stop the container you have been running you need to detach from the container using the escape sequence (unless you added -d to the run command) and send the following commands:

```
docker ps   to get the Container ID
docker stop <container-ID>
```

If you have not added -rm option you also need to manually remove the container

```
docker rm -f <container-ID>
```

An exemplary run command is the following:

```
docker run -d --rm -p 8080:8787 -v ${PWD}:/data rocker/geospatial
```

After the -p option you add the ports mapping: :. In this example, docker runs an image with r studio server which uses the port 8787 by default, but the host (the server where you are running docker) has the port 8787 already in use by other services. Therefore, you need to map the port of the image (8787) with a free port of the host (in this example 8080).

After the -v option you add the paths to mount a volume::. In this example, the working directory (the directory where you are running the command from) is mapped to a data directory in the container. ${PWD} stands for **Print Working Directory**.

Chapter 8

JupyterLab

This document shows how to pull and run a JupyterLab server locally and remotely.

8.1 Run Jupyter Lab Locally

This guide assumes you meet the following requirements in your personal machine (eg. laptop):

1. You have admin rights over your machine
2. You are running either Windows 10 Pro, macOS, or Linux

Assuming Docker is up and running (check `setup_docker.md` for that), you can pull an "image", which will let you run containers, by typing on a command line (`Terminal.app` or `PowerShell` are both good options):

```
docker pull darribas/gds:4.0
```

Upon executing the above command you will see output providing information regarding the download progress. Once the above command has finished installing your GDS stack, you are ready to go! To get a Jupyter session started, you can follow these steps:

1. Run on the same terminal as above the following command:

   ```
   docker run --rm -ti --user root -e NB_UID=$UID -e NB_GID=100 -p 8888:8888 -v ${PWD}:/home/jo
   ```

The command above spins up a container of the `gds` image, version 4.0 and ensures it is connected through two main bridges:

- Mapping your laptop's file system from where you have launched the command (`${PWD}`) to a folder called `work` on the home directory of the container. When you login to Jupyter (see below), you will see a `work` folder

and, if you click into it, you should see the content of your laptops folder in there.
- Mapping port `8888` from the container to your laptop, so you can connect to it through a browser.

It is important to know this command starts a Jupyter server on your machine and keeps it running, so please do not quit the window until you are done using Jupyter, otherwise it will crash.

2. Open your favorite browser (preferably Firefox or Chrome) and point it to `localhost:8888`

3. You will be asked for a password or a token. To find the correct one, check the terminal where you started the `docker run ...` command in 1) and look for the long token in the logs. Your prompt should look something (albeit not exactly) like this:

```
docker run --rm -ti -p 8888:8888 -v ${PWD}:/home/jovyan/work darribas/gds:4.0
Executing the command: jupyter notebook
[I 11:38:40.234 NotebookApp] Writing notebook server cookie secret to /home/jovya
[I 11:38:41.328 NotebookApp] Loading IPython parallel extension
[I 11:38:41.612 NotebookApp] JupyterLab extension loaded from /opt/conda/lib/pyth
[I 11:38:41.612 NotebookApp] JupyterLab application directory is /opt/conda/share
[I 11:38:43.091 NotebookApp] Serving notebooks from local directory: /home/jovyan
[I 11:38:43.091 NotebookApp] The Jupyter Notebook is running at:
[I 11:38:43.091 NotebookApp] http://ee20e7549b49:8888/?token=4dc814ee44c64383d5d3
[I 11:38:43.091 NotebookApp]  or http://127.0.0.1:8888/?token=4dc814ee44c64383d5d
[I 11:38:43.091 NotebookApp] Use Control-C to stop this server and shut down all
[C 11:38:43.114 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///home/jovyan/.local/share/jupyter/runtime/nbserver-6-open.html
    Or copy and paste one of these URLs:
        http://ee20e7549b49:8888/?token=4dc814ee44c64383d5d32dfd439fe62bbc17d9803
     or http://127.0.0.1:8888/?token=4dc814ee44c64383d5d32dfd439fe62bbc17d9803d9a
```

   The token you want to copy is the long series of letter and numbers right after `?token=`, starting by `4dc814ee`.

4. The token should let you into your Jupyter Lab session. Congratulations! You can then access the files in your computer through the `work` directory on the left-side pane.

8.2 Run Jupyter Lab Remotely

It is also possible to start a Jupyter server as above but, instead of run it on your local machine, it can run on a remote machine and you connect to that through your browser over the internet. The process in this context is a bit

more intricate because you need to ensure that the connection is secure, but overall it follows a similar pattern. The following steps below assume you can login to the remote server where you want to run Jupyter through `ssh` and the serve already has a Docker image installed, ready to be run.

- Login to the remote machine:

  `ssh <username>@<server.ip.address>`

1. Launch the container:

   `docker run --rm -ti --user root -e NB_UID=$UID -e NB_GID=100 -p <mapping_port>:8888 -v ${PWD`

   Note we are appending `start.sh` so it drops us into the command line of the container rather than launching the server directly

2. Run `jupyter notebook --generate-config`

3. Generate password as in the official tutorial

4. Since we will be using the created password we shall also enable SSL (secure sockets layer). SSL is a protocol for web browsers and servers that allows for the authentication, encryption and decryption of data sent over the Internet. Therefore, by enabling SSL our password won't be sent unencypted by our browser when we login to the server. We shall generate a self-signed SSL certificate with:

   `openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout mykey.key -out mycert.pem`

5. Next we shall update the `/home/jovyan/.jupyter/jupyter_notebook_config.py` file:

```
# Set options for certfile, ip, password, and toggle off
# browser auto-opening
c.NotebookApp.certfile = u'/home/jovyan/mycert.pem'
c.NotebookApp.keyfile = u'/home/jovyan/mykey.key'
# Set ip to '*' to bind on all interfaces (ips) for the public server
c.NotebookApp.ip = '*'
c.NotebookApp.password = u'sha1:bcd259ccf...<your hashed password here>'
c.NotebookApp.open_browser = False

# It is a good idea to set a known, fixed port for server access
c.NotebookApp.port = 8888
```

6. Launch secure Lab: `jupyter lab`

7. On your own machine (laptop/tablet), log in to `https://<server.ip.address>:<mapping_port>` with the password you have set. Since we are using SSL make sure you specify **https://** in your browser.

8.2.1 Self-signed Certificate Warnings

Upon accessing the notebook server your browser might warn you that your self-signed certificate is insecure or unrecognized. A fully compliant self-signed certificate is required to prevent these warnins. One approach towards solving this issue is to acquire a free SSL certificate via Let's Encrypt.

8.3 Using sudo within a container

For the above image password authentication has been disabled for the NB_USER jovyan. However, you might want to install additional programs using a package management tool (e.g., apt). To do so you can grant the within-container NB_USER passwordless sudo access by adding -e GRANT_SUDO=yes and –user root when launching the image:

```
docker run --rm -ti -e GRANT_SUDO=yes --user root -e NB_UID=$UID -e NB_GID=100 -p 8889
```

8.4 Useful Python Docker Images

- `gds_env`: a containerised platform for Geographid Data Science in Jupyter (Python & R)
- `jupyter-stacks`: official Jupyter stacks (the `gds_env` is based on these)

8.5 Practical

Follow the steps in the Remote Install section to run jupyter lab on one of the remote servers.

Note that you will need to use the command line editing software nano to edit update jupyter_notebook_config.py:

```
nano /home/jovyan/.jupyter/jupyter_notebook_config.py
```

A guide to using nano can be found here.

Chapter 9

Docker Containers for R

A widely-used suite of docker images for R has been developed by the Rocker project (Boettiger and Eddelbuettel, 2017). 29 repositories of rocker's images are listed on the docker hub providing R environments customized to perform a variety of tasks. Most of them include R studio server, allowing to work remotely from the R studio server web interface accessible through any browser.

Rocker images are versioned, meaning that if you want to employ a specific version of R rather than the latest you can specify an R version tag in the image name, i.e. *rocker/verse:3.4.0*, if no tag is requested you will automatically pull the latest version (Boettiger and Eddelbuettel, 2017).

9.1 Run Docker Locally

To run one of the rocker images locally on your laptop you need to install docker as described in `docker.md` and pull the image of your choise from the docker hub.

An useful image for running geospatial analysis, including the tidyverse + various geospatial packages, is *rocker/geospatial*:

You can pull the repository running the following command:

```
docker pull rocker/geospatial
```

Once the image is on your local machine, you can run it with the following command:

```
docker run -e USER=<your-user> -e PASSWORD=<your-password> -e USERID=$UID -e GROUPID=$GID -d --rm
```

All the docker run options used here have been described in `docker.md`. Here, it is assumed that the port 8787 is not occupied on your local machine. If you do not have an instance of R studio server running, that port should be free.

Once you have run the command you can access Rstudio server running on your local host through a browser at: `http://localhost:8787` and adding your credentials.

9.2 Run Docker Remotely

Docker is installed on all our shared resources. Most machines already have images that can be used for gds. *Rocker/geospatial* has been extended to have all libraries that GDSL members generally use. This extended image has been named *rstudio_gdsl*.

Once you accessed the server you can list the existing images with the following command:

```
docker image ls
```

You can pick the image that you need for your analysis, or pull a new image with the following command:

```
docker pull image-tag
```

Once you decided which image you want to use, you can start building the run command with the appropriate options:

- -d and -rm are advised, the first will automatically detach you from the container while the second will remove the container when you stop it. Do not forget to remove the container manually whether you have not added the -rm option.
- -p is needed to map the host's port with the container's port. To do this, you first need to verify which port is free to use. Ports that are permanently occupied on the servers are detailed on the servers_description file. In addition, you need to run *docker ps* to list the containers are running (if any) and verify which ports are using.
- -e to set the environment variables.

The final run command employing the extended image will look as follows:

```
docker run -e USER=<your-user> -e PASSWORD=<your-password> -e USERID=$UID -e GROUPID=$(
```

The environment variables necessary to run an image with R studio server are the following: * USER -> you can add your user name * PASSWORD -> a secret password of your choice, along with the USER these env variables will be used to access the R studio server web interface. * USERID and GROUPID -> these ids are needed to have the user of the host machine be the same of the docker container. When these do not match it will change the ownership of the volume you are mounting on the container, generating issues particularly when you create or save new files.

With the run command you generate a container where the image is running.

You can check your process with the *docker ps* command. As already mentioned above almost all rocker images include R studio server, therefore you can access the server-based application through its web interface. To do that, open your browser and go to the following address:

```
<host-ip>:<port>
```

You can find the host IP address in the document describing the servers.

9.3 Extend an image

Some images may not have all the libraries needed for your study. In that case you can extend the image with new libraries. Sometimes you can simply install a library from the R studio interface and it works with no errors. More frequently you will need to add some dependencies to make the installation work.

To do that you need to access the container's shell with the following command:

```
docker exec -it <mycontainer> bash
```

Once you have installed the packages and dependences you need, you can save it by creating a new image through the commit command:

```
docker commit -m "commit message" <container id> <new image name>
```

Chapter 10

Best Practices

10.1 Booking a server

Before booking a server look at the server information sheet to verify that the resource can meet your needs.

Book the Server with our booking system detailing your name, description of tasks and, when possible, forseen resources used (i.e. gpu, cpu threads,ram).

It is better to have no more than one person per server to avoid overload in the machine. However, non expensive tasks can be performed simultaneously. Always check what tasks will be performed on the booking calendar and communicate with each other.

10.2 Docker Considerations

- Always check what are the existing images before pulling a new one with:

  `docker image ls`

- To check what is running and which ports are occupied always run:

  `docker ps`

10.3 System Monitoring

It is highly recommended that you monitor the systems processes during before and while running your processes. This can be done via the interactive system-monitor process-viewer and process-manager **htop**, which can be started by typing htop into the terminal and pressing enter:

`htop`

Things to keep an eye on:

- How many threads are currently being used?
- How much memory is available?

If your process is taking longer than expected, then this may be caused by either too many processes running in parallel, or insufficient memory being available. The later can result in swap memory being used, which will significantly slow down your processes.

10.3.1 Memory Leaks

There is also the potential for "memory leaks" within the code that you are using. Memory leaks can be caused when:

- large objects are not released when they are no longer required;
- reference cycles within the code you are running.

Before running your code on shared resources, run smaller tests (when possible) on you local system while monitoring the memory usage over time. If the amount of memory remains more or less consistent, then you should be safe to run your application on a shared resource. Although note that memory leaks may be difficult to notice at first, and may only become apparent after running a program for hours/days. See the following guide for discovering memory leaks in python.

10.4 GPU Considerations

10.4.1 Monitoring

Another resource that requires monitoring are GPUs. A GPU can be monitored using the NVIDIA System Management Interface (nvididia-smi):

```
watch -n 0.1 nvidia-smi
```

This interface will allow you to monitor the memory usage, volatile GPU utility, temperature and fan speed.

10.4.2 Memory Growth

If there is no memory available, then it is worth enquiring with the other individual using the GPU if they are using TensorFlow and have enabled memory growth. If memory growth has not been enabled, then TensorFlow will by default allocate the all available GPU memory to a task. See the following discussion for more information. Memory growth can be enabled as follows within TensorFlow:

```
config = tf.ConfigProto()
config.gpu_options.allow_growth=True
```

```
sess = tf.Session(config=config)
```

10.4.3 Temperature Monitoring

When servers are situated within a non-air conditioned room it is also worth keeping an eye on the GPUs temperature, in particular when the server houses multiple GPUs.

Bibliography

Bailey, D. E. and Kurland, N. B. (2002). A review of telework research: Findings, new directions, and lessons for the study of modern work. *Journal of Organizational Behavior: The International Journal of Industrial, Occupational and Organizational Psychology and Behavior*, 23(4):383–400.

Boettiger, C. and Eddelbuettel, D. (2017). An Introduction to Rocker: Docker Containers for R. *The R Journal*, 9(2):527–536.

Brynjolfsson, E., Horton, J., Ozimek, A., Rock, D., Sharma, G., and Ye, H. Y. T. (2020). Covid-19 and remote work: An early look at us data. *Unpublished work*.

e Silva, J. d. A. and Melo, P. C. (2018). Does home-based telework reduce household total travel? a path analysis using single and two worker british households. *Journal of Transport Geography*, 73:148–162.

Holgersen, H., Jia, Z., and Svenkerud, S. (2020). Who and how many can work from home? evidence from task descriptions and norwegian job advertisements. *Evidence from Task Descriptions and Norwegian Job Advertisements.(April 20, 2020)*.

Kitamura, R., Nilles, J. M., Conroy, P., and Fleming, D. M. (1991). Telecommuting as a Transportation Planning Measure : Initial Results of California Pilot Project Ryuichi Kitamura Reprint No . 58 of California. *Transportation Research Record*, 1285:98–104.

Olson, M. H. (1983). Remote office work: changing work patterns in space and time. *Communications of the ACM*, 26(3):182–187.