USP HPC

Release 0.0.1

Victor Ivamoto

May 27, 2021

CONFIGURATION

High-Performance Computing (HPC) at Universidade de Sao Paulo (USP) comprises two clusters of servers for parallel and distributed computing for scientific research. Use Python libraries to train machine and deep learning models in a single server on GPU and scale up to the full cluster for large datasets.

Common attributes:

- Workload manager (Slurm) that schedules jobs in all servers.
- Shared network file system visible to all servers.
- Limited local storage.
- Login node with no GPU in **lince** cluster.

Cluster **aguia** contains servers for CPU processing while **lince** cluster is for GPU processing. Access to the clusters through **shark**.

Both clusters allow parallel and distributed computing for scientific purposes, for example training of machine learning and deep learning models.

This documentation explains how to:

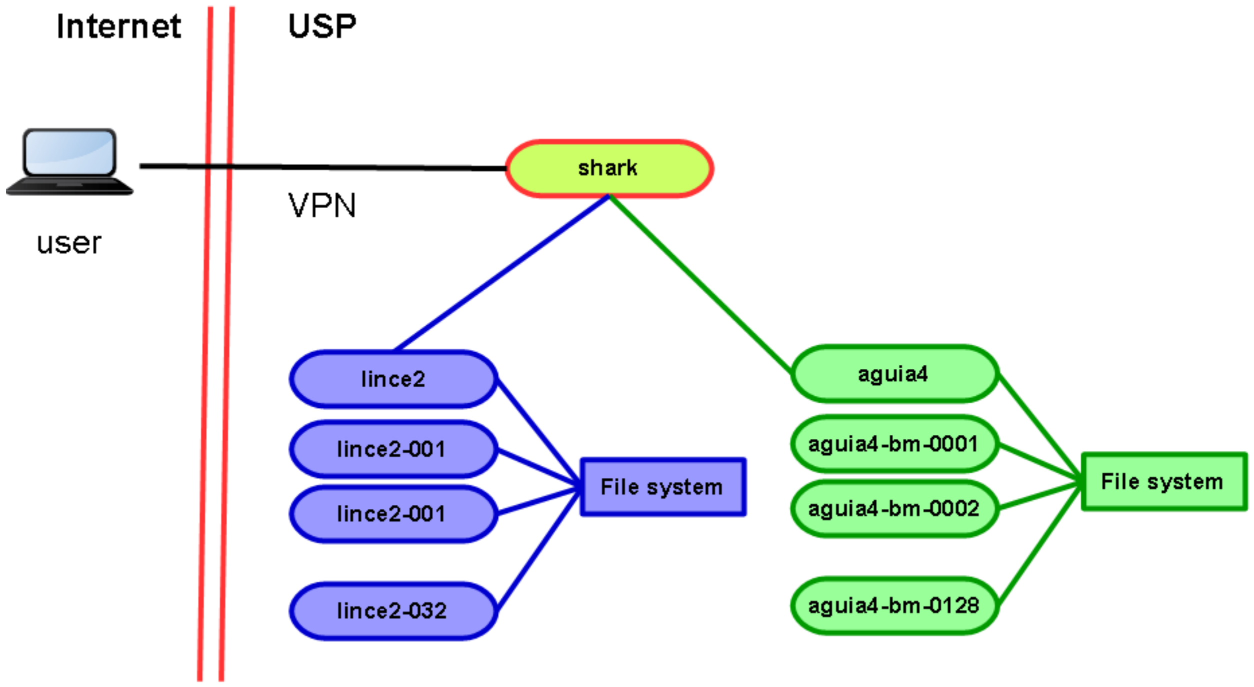
1. Setup a development environment for testing and debugging.
2. Schedule and manage Slurm jobs.
3. Use Python libraries to train machine learning and deep learning models.

This is a quick start guide for new users and may save several hours of searching and testing. Detailed and complete information on each topic is available in the Internet.

CHAPTER

ONE

CLUSTER ARCHITECTURE



Cluster **aguia**:

- 128 servers
- 512GB RAM
- 20 cores
- Intel(R) Xeon(R) CPU E7- 2870 @ 2.40GHz
- 256TB filesystem

Cluster **lince**:

- 32 servers
- 128GB RAM
- 16 cores
- Intel(R) Xeon(R) E5- 2680 @ 2.70GHz
- 2 GPUs NVIDIA Tesla K20m
- 55TB filesystem

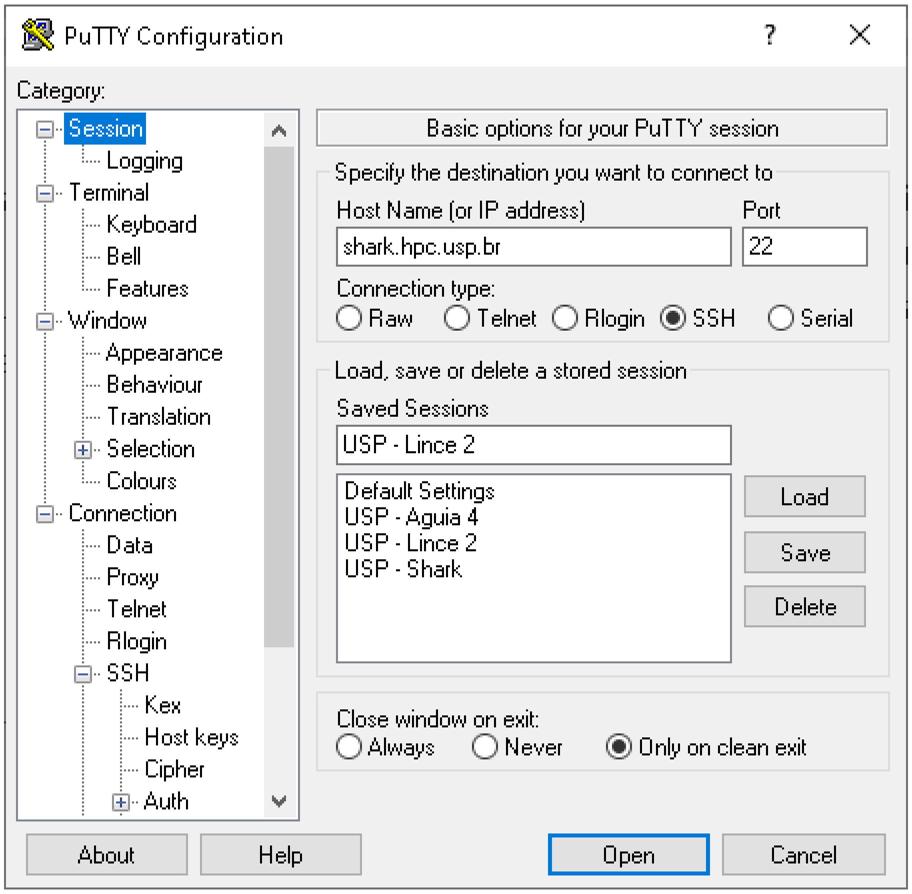
1.1 Client configuration

This section shows how to configure a development environment in your computer and how to connect to the servers. The instructions are for Microsoft Windows 10 and minor changes may be required for other operating systems.

Cluster access is done using command line interface (CLI) via SSH. PuTTY is a free terminal for remote access. This section shows how to setup PuTTY to connect and create a tunnel for file transfer and remote code execution.

1.1.1 PuTTY Configuration

Download PuTTY from the developer's website and install in your computer. After installation, open PuTTY:

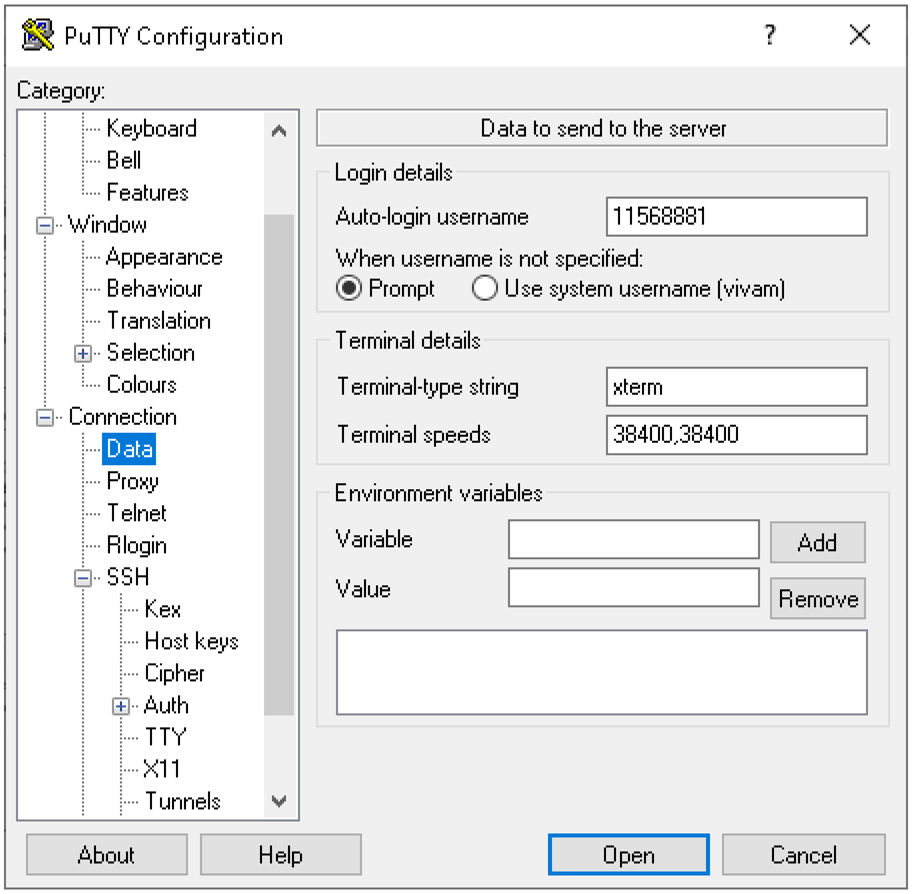


Specify the destination you want to connect to
Host Name (or IP address): shark.hpc.usp.br
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session
Saved Sessions: USP - Lince 2
Default Settings
USP - Aguia 4
USP - Lince 2
USP - Shark

Close window on exit: Always Never [✓ Only on clean exit]

In the **Connection->Data** menu, insert your user name (USP number). This avoids PuTTY asking for this on each login.



Login details

Auto-login username: 11568881

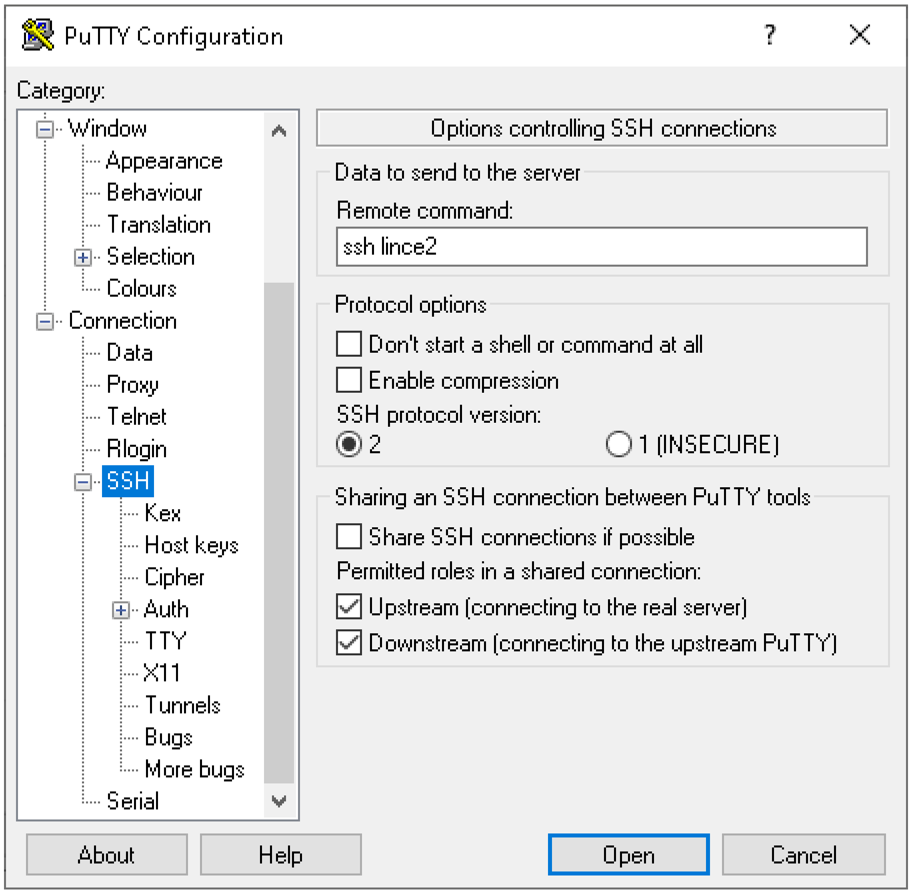
When username is not specified: [✓ Prompt] Use system username (vivam)

Terminal details

Terminal-type string: xterm

Terminal speeds: 38400,38400

PuTTY allows to connect directly in **lince** or **aguia** after connecting to **shark**. Just add the command `ssh lince2` in the **Remote command** field.



Options controlling SSH connections

Data to send to the server

Remote command: ssh lince2

Protocol options

- [ ] Don't start a shell or command at all
- [ ] Enable compression

SSH protocol version: [✓ 2] 1 (INSECURE)

Sharing an SSH connection between PuTTY tools

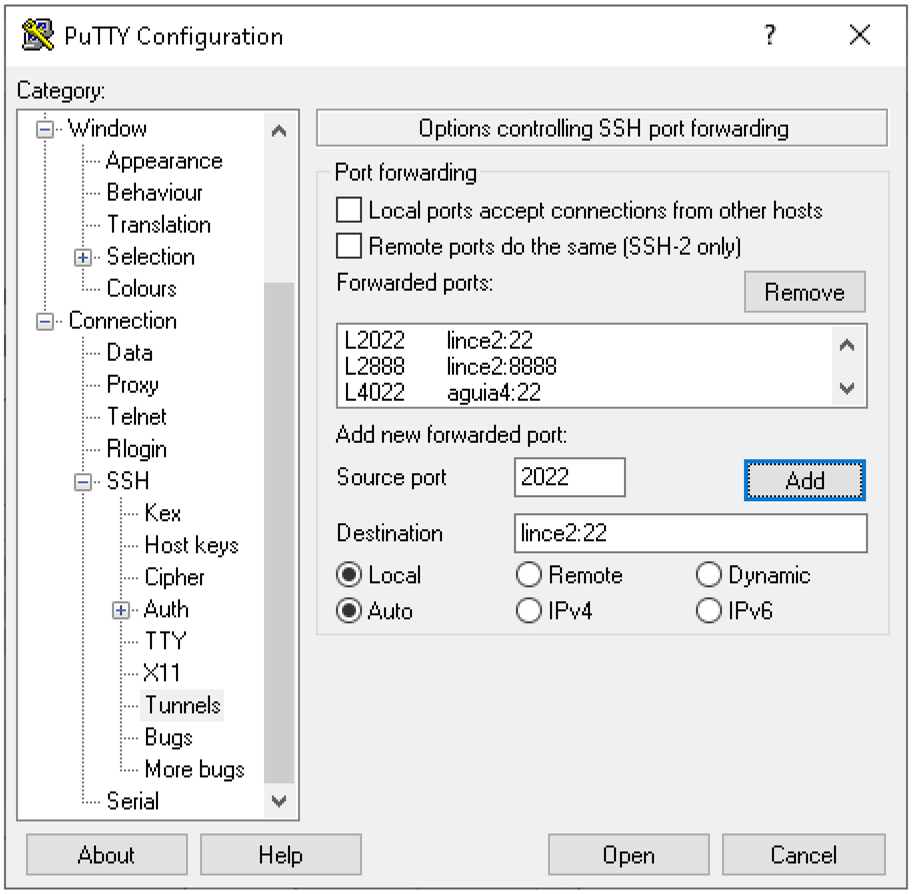
- [ ] Share SSH connections if possible

Permitted roles in a shared connection:

- [x] Upstream (connecting to the real server)
- [x] Downstream (connecting to the upstream PuTTY)

You may create a tunnel from your computer directly to **aguia** or **lince** for file transfer or to use Jupyter Notebook. Insert a local port and the port number of the server you will access. For instance, the image below assigns local port 2022 to port 22 in **lince2**. Port 22 is use for ssh connection and file transfer via sftp. So, to transfer files just use this command:

```
$ sftp -P 2022 user@localhost
```



Options controlling SSH port forwarding

Port forwarding

- [ ] Local ports accept connections from other hosts
- [ ] Remote ports do the same (SSH-2 only)

Forwarded ports:

```
L2022    lince2:22
L2888    lince2:8888
L4022    aguia4:22
```

Add new forwarded port:

Source port: 2022

Destination: lince2:22

[✓ Local] Remote Dynamic

[✓ Auto] IPv4 IPv6

Port settings suggestion:

Server	Local port	Remote port	Service
lince2	2022	22	SSH and SFTP
lince2	2888	8888	Jupyter Lab and Jupyter Notebook
aguia4	4022	22	SSH and SFTP
aguia4	4088	8888	Jupyter Lab and Jupyter Notebook

1.1.2 SSH Configuration

SSH uses login and password for authentication, or a pair of public and private keys. The latter case you can connect directly to the server without entering the password.

Create a pair of public and private keys in your computer to connect to **aguia** and **lince** while the tunnel is open. There's no need to enter the passphrase, just the default values:

```
C:\Users\vivam>ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\vivam/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\vivam/.ssh/id_rsa.
Your public key has been saved in C:\Users\vivam/.ssh/id_rsa.pub.
The key fingerprint is:
```

(continues on next page)

(continued from previous page)

```
SHA256:jRAhGYfJkD9Zj4ew97SCLImf98m5FiJdhlnKwbK06Jg vivam@DESKTOP-TAO705M
The key's randomart image is:
+---[RSA 2048]----+
|  +=o=+*         |
|  + *oo.         |
| o * @.+         |
|. O . *.+o       |
|oo + * +S..      |
|E.+ = o o        |
| . + . o         |
|  + .+.o         |
|   . .*.         |
+----[SHA256]-----+

C:\Users\vivam>
```

The key pair is saved in folder `.ssh`:

```
C:\Users\vivam>dir .ssh
O volume na unidade C é OS
O Número de Série do Volume é 963B-B7F6

Pasta de C:\Users\vivam\.ssh

03/04/2021  23:06    <DIR>          .
03/04/2021  23:06    <DIR>          ..
03/04/2021  23:06             1,679 id_rsa
03/04/2021  23:06               404 id_rsa.pub
                           2 arquivo(s)          2,083 bytes
                           2 pasta(s)    8,864,555,008 bytes disponíveis

C:\Users\vivam>
```

The next step is to append the public key content to the file `~/.ssh/authorized_keys` in **aguia** and **lince**:

```
C:\Users\vivam\.ssh>cat id_rsa.pub
ssh-rsa␣
↪AAAAB3NzaC1yc2EAAAADAQABAAABAQDKdaBWyChrn7wsR6+SolgFV8cPxR3hBdPTjGJgI5prPH25vK6XqSvXq8+mDvdXlBI2w9MQKI
↪ELu1n2vTFzJIcmAPY1qk8DsynQYU4CzD5+VVh+sMmVrzLUTTZ+3rC3gWWoMSWqn3IwFNiUsHgQhn6HqIzWhaUVyMF62e3YClfSEnc
↪qbUyQoflsd5I/9BgMLj1Tcz+b/SXT866aI5JPmIu9yKZH2b1u/
↪ZMtFEtydG9UlCxk+Deptlxryi2fIe9wFQuBq1CwZkh0Ikt93SOceksuK6ReW+pJwcocu2MzznCZMAgTiNVvGQAhQxFY7␣
↪vivam@DESKTOP-TAO705M

C:\Users\vivam\.ssh>
```

1.1.3 Test the conection

Open a terminal window and test the connection with SFTP:

```
(base) C:\Users\vivam>sftp -P 2022 11568881@localhost
Connected to 11568881@localhost.
sftp> pwd
Remote working directory: /scratch/11568881
sftp>
```

1.1.4 Running remote code

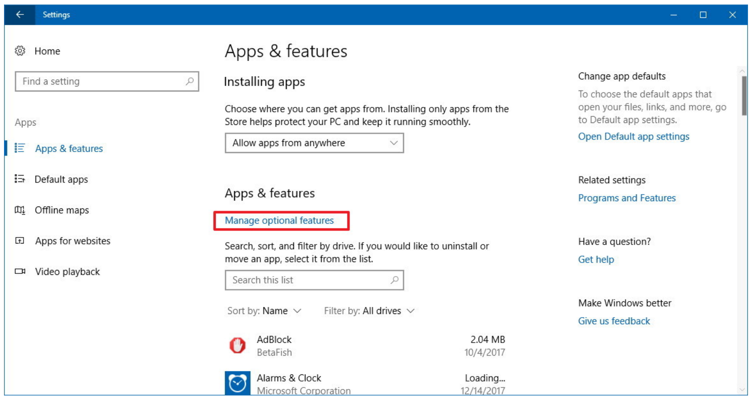
Editing and debugging code in a Linux terminal is not as efficient as in an IDE or in Jupyter. The **PyCharm Professional** edition allows remote execution via SSH and won't be discussed here because it's a paid version. Microsoft **Visual Studio Code** (VS Code) extension **Remote - SSH** also do the job. VS Code requires **OpenSSH**, so we'll install this first.

1.1.5 Install OpenSSH

The **Remote - SSH** extension uses OpenSSH; the PuTTY version is not supported.

In Windows 10 settings, open the windows **Applications and Resources**, then click in **Optional Resources** and check wether **OpenSSH Client** is installed. If it isn't, click in **+ Add resource** to install.

Open **Settings**, then go to **Apps** and click **Apps & features**. Under "Apps & features," click the **Manage optional features** link.



Choose where you can get apps from. Installing only apps from the Store helps protect your PC and keep it running smoothly.

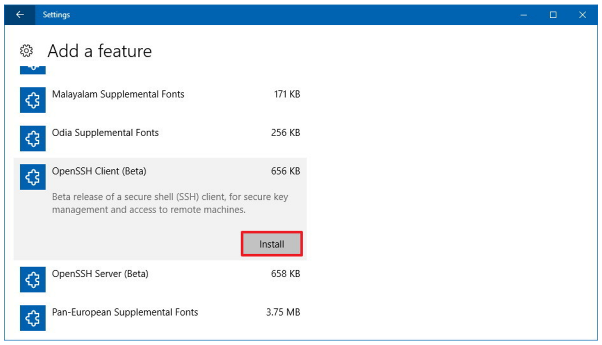
Search, sort, and filter by drive. If you would like to uninstall or move an app, select it from the list.

To choose the default apps that open your files, links, and more, go to Default app settings.

AdBlock BetaFish 2.04 MB 10/4/2017

Alarms & Clock Microsoft Corporation Loading... 12/14/2017

Click the **Add a feature** button, select the **OpenSSH Client** option and click **Install**.



Malayalam Supplemental Fonts	171 KB
Odia Supplemental Fonts	256 KB
OpenSSH Client (Beta)	656 KB
OpenSSH Server (Beta)	658 KB
Pan-European Supplemental Fonts	3.75 MB

Beta release of a secure shell (SSH) client, for secure key management and access to remote machines.

1.1.6 SSH configuration

Create the `ssh_config` file in your Windows user's `.ssh` folder with this content:

```
(base) C:\Users\vivam\.ssh>cat config
# Read more about SSH config files: https://linux.die.net/man/5/ssh_config
Host lince2
        HostName localhost
        User <YOUR NUSP>
        IdentityFile ~/.ssh/id_rsa
        Port 2022
Host aguia4
        HostName localhost
        User <YOUR NUSP>
        IdentityFile ~/.ssh/id_rsa
        Port 4022
```

User is your USP number (NUSP) used to login into the servers and **Port** is the SSH tunnel port.

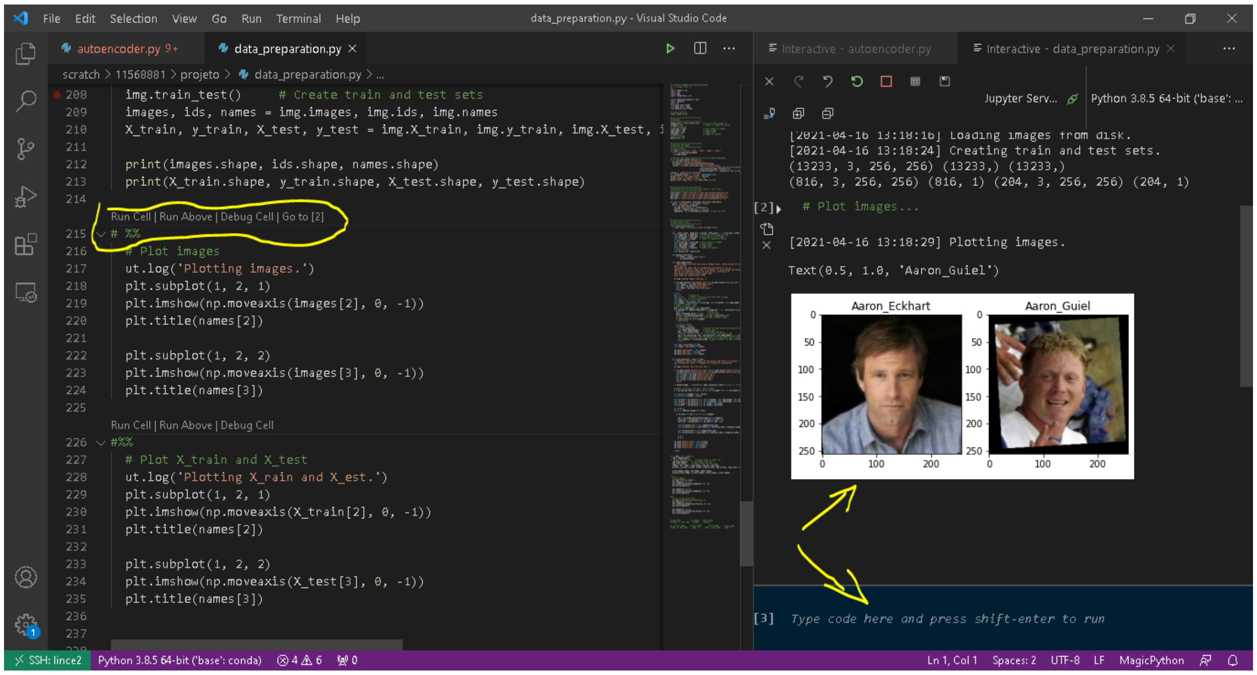
1.1.7 VS Code configuration

Install VS Code and the following extensions:

- Remote - SSH (Microsoft)
- Remote - SSH: Editing Configuration Files (Microsoft)
- Jupyter (Microsoft)
- Python (Microsoft)
- Python Extension Pack (Don Jayamanne)

With the **Jupyter** extension you can edit and execute notebooks in interactive mode, which is handy to make small tests and debugging.

The **Python Extension Pack** allows you to run small chunks of code like in Jupyter. It also shows charts and images in a side bar window. Just insert `#%%` in a new line to convert the code below in a Jupyter cell, as shown in the image below.

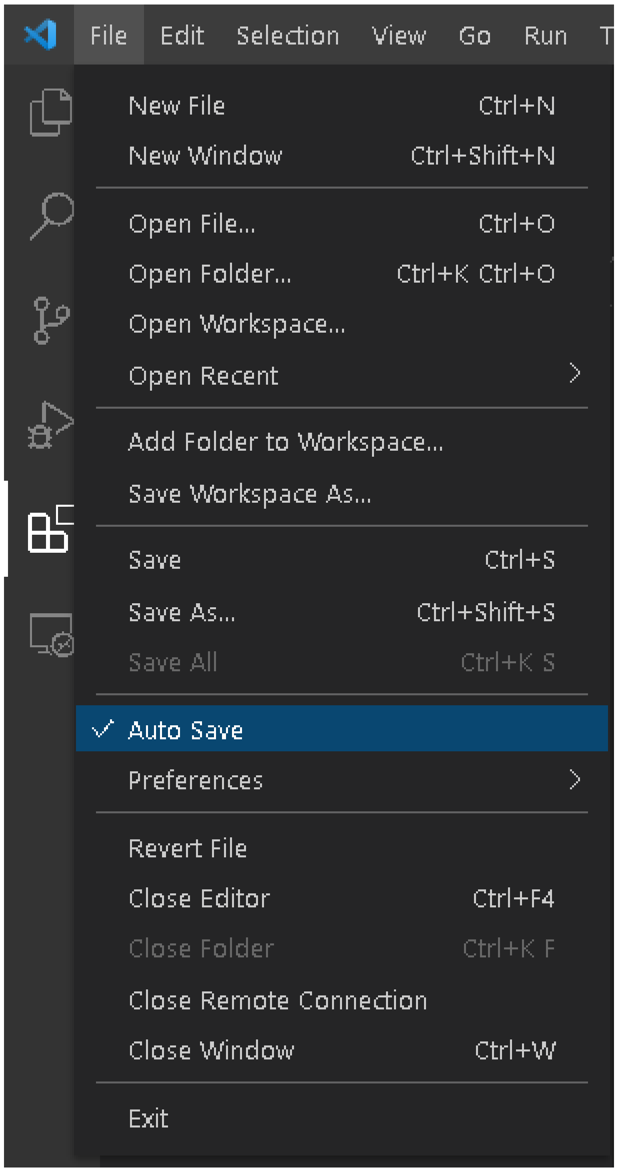


File
Edit
Selection
View
Go
Run
Terminal
Help
data_preparation.py - Visual Studio Code
autoencoder.py 9+
data_preparation.py
scratch > 11568881 > projeto > data_preparation.py > ...
Run Cell | Run Above | Debug Cell | Go to [2]
Run Cell | Run Above | Debug Cell
Interactive - autoencoder.py
Interactive - data_preparation.py
Jupyter Serv...
Python 3.8.5 64-bit ('base': ...
Type code here and press shift-enter to run
SSH: lince2
Python 3.8.5 64-bit ('base': conda)
Ln 1, Col 1
Spaces: 2
UTF-8
LF
MagicPython
Aaron_Eckhart
Aaron_Guiel

```
    img.train_test()      # Create train and test sets
    images, ids, names = img.images, img.ids, img.names
    X_train, y_train, X_test, y_test = img.X_train, img.y_train, img.X_test, i

    print(images.shape, ids.shape, names.shape)
    print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)

# %%
    # Plot images
    ut.log('Plotting images.')
    plt.subplot(1, 2, 1)
    plt.imshow(np.moveaxis(images[2], 0, -1))
    plt.title(names[2])

    plt.subplot(1, 2, 2)
    plt.imshow(np.moveaxis(images[3], 0, -1))
    plt.title(names[3])

#%%
    # Plot X_train and X_test
    ut.log('Plotting X_rain and X_est.')
    plt.subplot(1, 2, 1)
    plt.imshow(np.moveaxis(X_train[2], 0, -1))
    plt.title(names[2])

    plt.subplot(1, 2, 2)
    plt.imshow(np.moveaxis(X_test[3], 0, -1))
    plt.title(names[3])
```

```
[2021-04-16 13:18:16] Loading images from disk.
[2021-04-16 13:18:24] Creating train and test sets.
(13233, 3, 256, 256) (13233,) (13233,)
(816, 3, 256, 256) (816, 1) (204, 3, 256, 256) (204, 1)
[2]  # Plot images...

[2021-04-16 13:18:29] Plotting images.

Text(0.5, 1.0, 'Aaron_Guiel')
```

Prevent data loss while editing code enabling **Auto Save**:



1.1.8 Test the connection

See details at https://code.visualstudio.com/docs/remote/ssh-tutorial

1.1.9 File transfer with SFTP

SFTP usage:

```
C:\>sftp
usage: sftp [-46aCfpqrv] [-B buffer_size] [-b batchfile] [-c cipher]
            [-D sftp_server_path] [-F ssh_config] [-i identity_file] [-l limit]
            [-o ssh_option] [-P port] [-R num_requests] [-S program]
            [-s subsystem | sftp_server] destination
```

SFTP commands:

```
sftp> help
Available commands:
bye                                Quit sftp
cd path                            Change remote directory to 'path'
chgrp grp path                     Change group of file 'path' to 'grp'
chmod mode path                    Change permissions of file 'path' to 'mode'
chown own path                     Change owner of file 'path' to 'own'
df [-hi] [path]                    Display statistics for current directory or
                                                                filesystem containing
↪'path'
exit                               Quit sftp
get [-afPpRr] remote [local]       Download file
reget [-fPpRr] remote [local]      Resume download file
reput [-fPpRr] [local] remote      Resume upload file
help                               Display this help text
lcd path                           Change local directory to 'path'
lls [ls-options [path]]            Display local directory listing
lmkdir path                        Create local directory
ln [-s] oldpath newpath            Link remote file (-s for symlink)
lpwd                               Print local working directory
ls [-1afhlnrSt] [path]             Display remote directory listing
lumask umask                       Set local umask to 'umask'
mkdir path                         Create remote directory
progress                           Toggle display of progress meter
put [-afPpRr] local [remote]       Upload file
pwd                                Display remote working directory
quit                               Quit sftp
rename oldpath newpath             Rename remote file
rm path                            Delete remote file
rmdir path                         Remove remote directory
symlink oldpath newpath            Symlink remote file
version                            Show SFTP version
!command                           Execute 'command' in local shell
!                                  Escape to local shell
?                                  Synonym for help
sftp>
```

1.2 Server configuration

1.2.1 Install Miniconda and Python libraries

Python libraries installed in HPC are outdated and you may want to use newer releases. This section shows how to install Miniconda in the user's home directory, without affecting the original installation.

Miniconda installs the most recent release of Python and `pip` in the user's folder. The libraries installed with `pip` and `conda` are also installed in your folder.

All commands shall be executed in the server's Linux terminal.

Check CUDA release

Before installing the libraries, you need the current CUDA release to choose the right package. Run this command:

```
$ nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2019 NVIDIA Corporation
Built on Wed_Apr_24_19:10:27_PDT_2019
Cuda compilation tools, release 10.1, V10.1.168
```

The output shows the current release is 10.1.

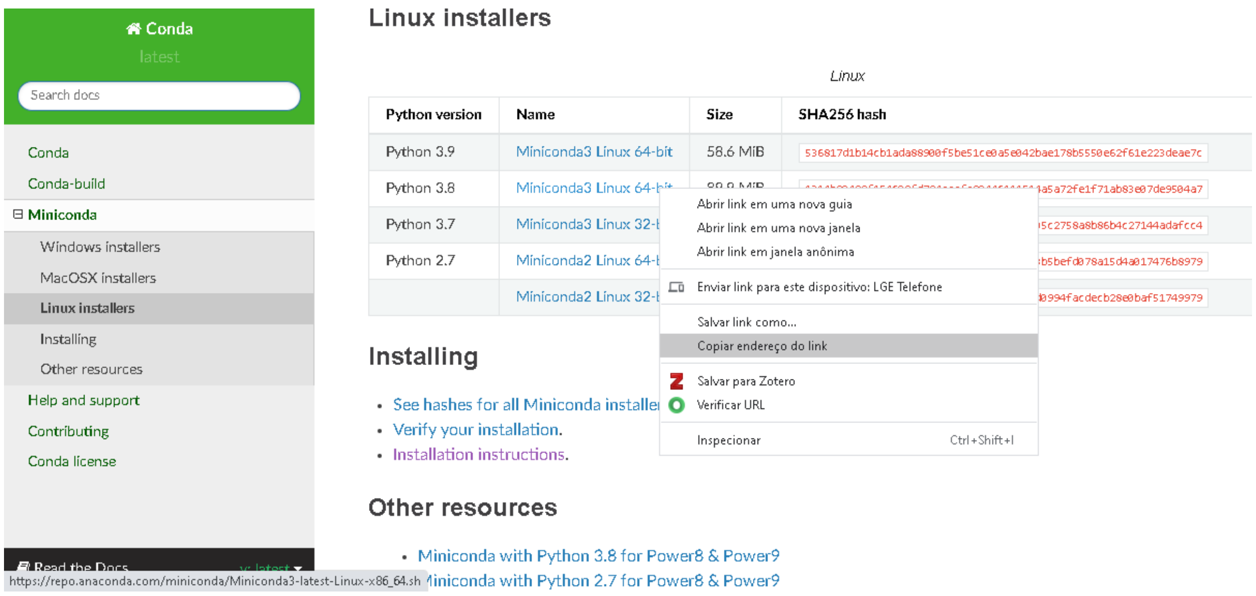
Install Miniconda

Miniconda is a package management system for Python and provides `pip`, `conda` and the most recent Python release with basic libraries. Miniconda requires less disk space than Anaconda and is faster to install. After installing Miniconda, you may install just the libraries that you'll use. Anaconda installs many packages and applications that won't be used.

Select a Miniconda release with the Python version compatible with the libraries that you need, since not all libraries are compatible with the newest release of Python. For example, the latest release of Tensorflow doesn't work with the latest release of Python and CUDA 10.1.

> **Warning:** Check Python, CUDA and libraries compatibility before installing Miniconda. Many libraries only work with specific Python and CUDA versions.

Access Miniconda website and copy the link with the installation script that you need. This example uses release 3.8.



Linux installers

Linux

Python version	Name	Size	SHA256 hash
Python 3.9	Miniconda3 Linux 64-bit	58.6 MiB	536817d1b14cb1ada88900f5be51ce0a5e042bae178b5550e62f61e223deae7c
Python 3.8	Miniconda3 Linux 64-bit	[illegible]	[illegible]a5a72fe1f71ab83e07de9504a7
Python 3.7	Miniconda3 Linux 32-[illegible]		5c2758a8b86b4c27144adafcc4
Python 2.7	Miniconda2 Linux 64-[illegible]		b5befd078a15d4a017476b8979
	Miniconda2 Linux 32-[illegible]		0994facdecb28e0baf51749979

Installing

- See hashes for all Miniconda installe
- Verify your installation.
- Installation instructions.

Other resources

- Miniconda with Python 3.8 for Power8 & Power9
- Miniconda with Python 2.7 for Power8 & Power9

Use the `wget` command to download the script with the link you copied from Miniconda site:

```
$ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
$ bash Miniconda3-latest-Linux-x86_64.sh
```

The installation script asks a few questions, just press ENTER to accept the default values and accept the license terms. When prompted to initialize Miniconda, answer “yes”. This creates the default base environment that will be automatically activated when you log into the cluster. All libraries will be installed in this environment.

> Do you wish the installer to initialize Miniconda3 by running conda init? [yes|no] [no] >>> yes

Now reload `~/.bashrc` to update the environment variables and activate the base environment:

```
$ source ~/.bashrc
```

Warning: Make sure to select the right environment before installation and always check Python and libraries after installation.

Install Python libraries for your project. Tensorflow and PyTorch use custom installation which depends on Python and CUDA versions:

```
$ conda install -c conda-forge numpy pandas matplotlib scikit-learn
```

Optionally update the libaries for the most recent version. In this example, Miniconda installed scikit-learn version 0.23 and this command upgrades to 0.24

```
# Update scikit-learn
$ conda upgrade -c conda-forge scikit-learn
```

Install Tensorflow

From Tensorflow website, select the correct version according to Python and CUDA versions. Since we have Python 3.8 and CUDA 10.1, the best Tensorflow version is 2.3:

```
$ pip install tensorflow==2.3
```

Install PyTorch

Similarly, check PyTorch website to install the correct version:

```
# Install PyTorch
# 1. Version with GPU to install in lince (CUDA 10.1 - Python 3.8)
$ conda install pytorch torchvision torchaudio cudatoolkit=10.1 -c pytorch


# 2. Version without GPU to install in aguia
$ conda install pytorch torchvision torchaudio cpuonly -c pytorch
```

Install Dask

Dask is a library for parallel and distributed computing. Dask's schedulers scale to thousand-node clusters and its algorithms have been tested on some of the largest supercomputers in the world. It easily integrates with NumPy, Pandas and scikit-learn:

```
$ conda install dask distributed
```

Install RAPIDS

The RAPIDS suite of open source software libraries and APIs gives you the ability to execute end-to-end data science and analytics pipelines entirely on GPUs. Use the release selector to get the right installation command:

```
$ conda install -c rapidsai -c nvidia -c conda-forge rapids-blazing=0.19 python=3.8␣
↪cudatoolkit=10.1
```

1.2.2 Installation tests

After installing the libraries, run Python and import the libraries to confirm the correct version:

```
$ cat system_info.py
#!/scratch/<YOUR_NUSP>/miniconda3/bin/python3
import sys
import numpy as np
import pandas as pd
import matplotlib as mpl
import sklearn as sk

print('='*20, 'Software version', '='*20)
print("Python:", sys.version.split('\n')[0])
print("NumPy:", np.__version__)
print("Pandas:", pd.__version__)
print('Matplotlib:', mpl.__version__)
print("Sklearn:", sk.__version__)
```

Warning: Check Tensorflow, PyTorch and RAPIDS on the processing node, since the login server doesn't have access to GPU.

Lince login node doesn't provide GPU access, so you need to connect to a processing node to check Tensorflow, PyTorch and RAPIDS:

```
$ ssh lince2-001
```

Once connected in lince2-001, connect to a processing node and make sure that Tensorflow and PyTorch recognize the GPU:

```
$ python
Python 3.8.5 (default, Sep  4 2020, 07:30:14)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
```

Check Tensorflow installation

Import Tensorflow:

```
>>> import tensorflow as tf
2021-05-06 10:09:05.807604: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:48] Successfully opened dynamic library libcudart.so.10.1
```

Check Tensorflow version:

```
>>> tf.__version__
'2.3.0'
```

Check if Tensorflow can list both GPUs:

```
>>> tf.config.list_physical_devices()
2021-05-06 10:09:19.154886: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:48] Successfully opened dynamic library libcuda.so.1
2021-05-06 10:09:19.167369: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1716]␣
↪Found device 0 with properties:
pciBusID: 0000:05:00.0 name: Tesla K20m computeCapability: 3.5
coreClock: 0.7055GHz coreCount: 13 deviceMemorySize: 4.63GiB deviceMemoryBandwidth: 193.
↪71GiB/s
2021-05-06 10:09:19.168426: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1716]␣
↪Found device 1 with properties:
pciBusID: 0000:83:00.0 name: Tesla K20m computeCapability: 3.5
coreClock: 0.7055GHz coreCount: 13 deviceMemorySize: 4.63GiB deviceMemoryBandwidth: 193.
↪71GiB/s
2021-05-06 10:09:19.168477: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:48] Successfully opened dynamic library libcudart.so.10.1
2021-05-06 10:09:19.173624: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:48] Successfully opened dynamic library libcublas.so.10
2021-05-06 10:09:19.176772: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:48] Successfully opened dynamic library libcufft.so.10
2021-05-06 10:09:19.177907: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:48] Successfully opened dynamic library libcurand.so.10
2021-05-06 10:09:19.181156: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:48] Successfully opened dynamic library libcusolver.so.10
2021-05-06 10:09:19.183197: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:48] Successfully opened dynamic library libcusparse.so.10
2021-05-06 10:09:19.188812: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:48] Successfully opened dynamic library libcudnn.so.7
2021-05-06 10:09:19.192994: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1858]␣
↪Adding visible gpu devices: 0, 1
[PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU'), PhysicalDevice(name='/
↪physical_device:XLA_CPU:0', device_type='XLA_CPU'), PhysicalDevice(name='/physical_
↪device:XLA_GPU:0', device_type='XLA_GPU'), PhysicalDevice(name='/physical_device:XLA_
↪GPU:1', device_type='XLA_GPU'), PhysicalDevice(name='/physical_device:GPU:0', device_
↪type='GPU'), PhysicalDevice(name='/physical_device:GPU:1', device_type='GPU')]
```

Check PyTorch installation

Import PyTorch:

```
>>> import torch
```

Check PyTorch version:

```
>>> torch.__version__
'1.7.0'
>>>
```

Check the number of GPUs available:

```
>>> torch.cuda.device_count()
2
```

Check GPU name:

```
>>> torch.cuda.get_device_name(torch.cuda.current_device())
'Tesla K20m'
```

Check RAPIDS installation

RAPIDS automatically detects the GPU when you import a library:

```
>>> import cudf
/scratch/11568881/miniconda3/lib/python3.8/site-packages/cudf/utils/gpu_utils.py:92:␣
↪UserWarning: You will need a GPU with NVIDIA Pascal™ or newer architecture
Detected GPU 0: Tesla K20m
Detected Compute Capability: 3.5
  warnings.warn(
```

1.2.3 System information

You may need the hardware information to choose the right software release. The following commands show the main hardware devices and the Linux release. The commands may be executed directly in the Linux terminal, or you may save in a script and run in SLURM job. Note that PyTorch provides a custom version for each CUDA version:

```
$ cat system_info.sh
#!/usr/bin/bash
echo ========================
echo SLURM: ID of job allocation
echo ========================
echo $SLURM_JOB_ID              # ID of job allocation

echo ========================
echo SLURM: Directory job where was submitted
echo ========================
echo $SLURM_SUBMIT_DIR          # Directory job where was submitted

echo ========================
```

(continues on next page)

(continued from previous page)

```
echo SLURM: File containing allocated hostnames
echo ========================
echo $SLURM_JOB_NODELIST        # File containing allocated hostnames

echo ========================
echo SLURM: Total number of cores for job
echo ========================
echo $SLURM_NTASKS              # Total number of cores for job

echo ========================
echo SLURM: GPU devide ID that assigned to the job to use
echo ========================
echo $CUDA_VISIBLE_DEVICES

echo ========================
echo Hostname
echo ========================
hostname

echo ========================
echo Memory Info \(GB\):
echo ========================
free -g

echo ========================
echo CPU Info:
echo ========================
lscpu

echo ========================
echo Disk space
echo ========================
df -h

echo ========================
echo GPU 1
echo ========================
nvidia-smi

echo ========================
echo GPU 2
echo ========================
lshw -C display

echo ========================
echo CUDA Version
echo ========================
nvcc --version

echo ========================
echo Linux version
echo ========================
```

(continues on next page)

(continued from previous page)

```
cat /etc/os-release

echo ========================
echo PATH
echo ========================
echo $PATH

echo ========================
echo Python
echo ========================
which python
which python3

echo ========================
echo Conda
echo ========================
which conda
conda --version

echo ========================
echo Pip
echo ========================
which pip
pip --version

echo ========================
echo Python Library Versions
echo ========================
python system_info.py
```

1.3 Install Google Drive

Since data in the server are deleted periodically, keeping an external backup makes the recover easy. Storing in Google Drive is handy since the USP account has unlimited storage. This section shows how to access Google Drive in lince with **gdrive**.

Caution: Anyone with access to **gdrive** may access your files in Google Drive. If you have sensitive information in your USP account, create another Google account just to store HPC backups.

According to the developer, gdrive is no longer maintained although it's still possible to connect and sync folders with Google Drive.

Steps:

1. Create Google credentials
2. Download GDrive
3. Edit configuration
4. Code compilation

5. Access GDrive

First you need to create a credential in Google to give permission for an application access GDrive. When you do this, you'll receive an ID and a key. This step is done only once. Follow the instructions in this site .

1.3.1 Download GDrive

Run all Linux commands in the server. The GDrive code was written in `go` and may be downloaded from GitHub with this command:

```
$ go get github.com/prasmussen/gdrive
```

1.3.2 Edit configuration

The code will be downloaded to the folder `~/go/src/github.com/prasmussen/gdrive`. Go to this folder, open the file `handlers_drive.go` with a text editor of your choice and update the following lines with the credentials you got from the previous step:

```
const ClientId = "367116221053-7n0v**.apps.googleusercontent.com"
const ClientSecret = "1qsNodXN*****jUjmvhoO"
```

1.3.3 Code compilation

Now, compile the code in this folder with the following command. No message should appear after running it:

```
$ go build
```

The `gdrive` file is created in this folder. Change permission to make it executable:

```
$ chmod 755 gdrive
```

1.3.4 Access GDrive

Run the `gdrive list` command to start the authentication process:

```
$ ./gdrive list
Authentication needed
Go to the following url in your browser:
https://accounts.google.com/o/oauth2/auth?access_type=offl...
Enter verification code:
```

Copy and paste the link in your browser to allow access. After inserting the verification code, the root folder in your Google Drive is listed in terminal. Then, move the `gdrive` file to folder `~/.local/bin`:

```
$ mkdir ~/.local/bin
$ mv ~/go/src/github.com/prasmussen/gdrive/gdrive ~/.local/bin
```

The complete list of options in `gdrive` is available at the developer's website .

1.3.5 Folder synchronization

`gdrive` creates a `fileId` for each file and folder stored in drive. Use this `fileId` to sync a folder. In this example, the `fileId` is `0B3X9GlR6EmbnOEd6cEh6bU9XZWM` and the folder `_release/bin` is being synced.

Create directory on drive:

```
$ gdrive mkdir drive-bin
Directory 0B3X9GlR6EmbnOEd6cEh6bU9XZWM created
```

Sync to drive:

```
$ gdrive sync upload _release/bin 0B3X9GlR6EmbnOEd6cEh6bU9XZWM
Starting sync...
Collecting local and remote file information...
Found 32 local files and 0 remote files

6 remote directories are missing
[0001/0006] Creating directory drive-bin/bsd
[0002/0006] Creating directory drive-bin/linux
[0003/0006] Creating directory drive-bin/osx
[0004/0006] Creating directory drive-bin/plan9
[0005/0006] Creating directory drive-bin/solaris
[0006/0006] Creating directory drive-bin/windows

26 remote files are missing
[0001/0026] Uploading bsd/gdrive-dragonfly-x64 -> drive-bin/bsd/gdrive-dragonfly-x64
[0002/0026] Uploading bsd/gdrive-freebsd-386 -> drive-bin/bsd/gdrive-freebsd-386
[0003/0026] Uploading bsd/gdrive-freebsd-arm -> drive-bin/bsd/gdrive-freebsd-arm
[0004/0026] Uploading bsd/gdrive-freebsd-x64 -> drive-bin/bsd/gdrive-freebsd-x64
[0005/0026] Uploading bsd/gdrive-netbsd-386 -> drive-bin/bsd/gdrive-netbsd-386
[0006/0026] Uploading bsd/gdrive-netbsd-arm -> drive-bin/bsd/gdrive-netbsd-arm
[0007/0026] Uploading bsd/gdrive-netbsd-x64 -> drive-bin/bsd/gdrive-netbsd-x64
[0008/0026] Uploading bsd/gdrive-openbsd-386 -> drive-bin/bsd/gdrive-openbsd-386
[0009/0026] Uploading bsd/gdrive-openbsd-arm -> drive-bin/bsd/gdrive-openbsd-arm
[0010/0026] Uploading bsd/gdrive-openbsd-x64 -> drive-bin/bsd/gdrive-openbsd-x64
[0011/0026] Uploading linux/gdrive-linux-386 -> drive-bin/linux/gdrive-linux-386
[0012/0026] Uploading linux/gdrive-linux-arm -> drive-bin/linux/gdrive-linux-arm
[0013/0026] Uploading linux/gdrive-linux-arm64 -> drive-bin/linux/gdrive-linux-arm64
[0014/0026] Uploading linux/gdrive-linux-mips64 -> drive-bin/linux/gdrive-linux-mips64
[0015/0026] Uploading linux/gdrive-linux-mips64le -> drive-bin/linux/gdrive-linux-
↪mips64le
[0016/0026] Uploading linux/gdrive-linux-ppc64 -> drive-bin/linux/gdrive-linux-ppc64
[0017/0026] Uploading linux/gdrive-linux-ppc64le -> drive-bin/linux/gdrive-linux-ppc64le
[0018/0026] Uploading linux/gdrive-linux-x64 -> drive-bin/linux/gdrive-linux-x64
[0019/0026] Uploading osx/gdrive-osx-386 -> drive-bin/osx/gdrive-osx-386
[0020/0026] Uploading osx/gdrive-osx-arm -> drive-bin/osx/gdrive-osx-arm
[0021/0026] Uploading osx/gdrive-osx-x64 -> drive-bin/osx/gdrive-osx-x64
[0022/0026] Uploading plan9/gdrive-plan9-386 -> drive-bin/plan9/gdrive-plan9-386
[0023/0026] Uploading plan9/gdrive-plan9-x64 -> drive-bin/plan9/gdrive-plan9-x64
[0024/0026] Uploading solaris/gdrive-solaris-x64 -> drive-bin/solaris/gdrive-solaris-x64
[0025/0026] Uploading windows/gdrive-windows-386.exe -> drive-bin/windows/gdrive-windows-
↪386.exe
[0026/0026] Uploading windows/gdrive-windows-x64.exe -> drive-bin/windows/gdrive-windows-
↪x64.exe
```

(continues on next page)

(continued from previous page)

```
Sync finished in 1m18.891946279s
```

Add new local file:

```
$ echo "google drive binaries" > _release/bin/readme.txt
```

Sync again:

```
$ gdrive sync upload _release/bin 0B3X9GlR6EmbnOEd6cEh6bU9XZWM
Starting sync...
Collecting local and remote file information...
Found 33 local files and 32 remote files

1 remote files are missing
[0001/0001] Uploading readme.txt -> drive-bin/readme.txt
Sync finished in 2.201339535s
```

Modify local file:

```
$ echo "for all platforms" >> _release/bin/readme.txt
```

Sync again:

```
$ gdrive sync upload _release/bin 0B3X9GlR6EmbnOEd6cEh6bU9XZWM
Starting sync...
Collecting local and remote file information...
Found 33 local files and 33 remote files

1 local files has changed
[0001/0001] Updating readme.txt -> drive-bin/readme.txt
Sync finished in 1.890244258s
```

1.3.6 List of options

Use the command `gdrive help` to list the available options:

```
$ gdrive help
gdrive usage:

gdrive [global] list [options]                                 List files
gdrive [global] download [options] <fileId>                    Download file or directory
gdrive [global] download query [options] <query>               Download all files and␣
↪directories matching query
gdrive [global] upload [options] <path>                        Upload file or directory
gdrive [global] upload - [options] <name>                      Upload file from stdin
gdrive [global] update [options] <fileId> <path>               Update file, this creates␣
↪a new revision of the file
gdrive [global] info [options] <fileId>                        Show file info
gdrive [global] mkdir [options] <name>                         Create directory
gdrive [global] share [options] <fileId>                       Share file or directory
gdrive [global] share list <fileId>                            List files permissions
gdrive [global] share revoke <fileId> <permissionId>           Revoke permission
```

(continues on next page)

(continued from previous page)

```
gdrive [global] delete [options] <fileId>                       Delete file or directory
gdrive [global] sync list [options]                             List all syncable␣
↪directories on drive
gdrive [global] sync content [options] <fileId>                 List content of syncable␣
↪directory
gdrive [global] sync download [options] <fileId> <path>         Sync drive directory to␣
↪local directory
gdrive [global] sync upload [options] <path> <fileId>           Sync local directory to␣
↪drive
gdrive [global] changes [options]                               List file changes
gdrive [global] revision list [options] <fileId>                List file revisions
gdrive [global] revision download [options] <fileId> <revId>    Download revision
gdrive [global] revision delete <fileId> <revId>                Delete file revision
gdrive [global] import [options] <path>                         Upload and convert file␣
↪to a google document, see 'about import' for available conversions
gdrive [global] export [options] <fileId>                       Export a google document
gdrive [global] about [options]                                 Google drive metadata,␣
↪quota usage
gdrive [global] about import                                    Show supported import␣
↪formats
gdrive [global] about export                                    Show supported export␣
↪formats
gdrive version                                                  Print application version
gdrive help                                                     Print help
gdrive help <command>                                           Print command help
gdrive help <command> <subcommand>                              Print subcommand help
```

1.3.7 Schedule daily backup

Schedule daily backup of your folder with `crontab`. Use this template:

```
$ crontab -e
# Crontab - Crontab template to automate virtual world
# Template from https://gist.github.com/bretonics/9a48a3b9ef32d93d15f45c3f007550b4
# Andrés Bretón ~ http://andresbreton.com, dev@andresbreton.com
#
# ===============================================================================
# .----------------- minute (0 - 59)
# |  .-------------- hour (0 - 23)
# |  |  .----------- day of month (1 - 31)
# |  |  |  .-------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |   + command
# *  *  *  *  *  CMD
# ===============================================================================
#
# Set Path
PATH=/bin:/usr/bin:/usr/local/bin:/scratch/<USER_ID>
#
# Backup of work folder daily at 02:00 AM
# (change NUSP, folder name and fileId)
```

(continues on next page)

(continued from previous page)

```
00 02 * * *   /scratch/<Seu NUSP>/.local/bin/gdrive sync upload _release/bin␣
↪0B3X9GlR6EmbnOEd6cEh6bU9XZWM   >   backup.log  2>&1
```

1.4 Code development

The job scheduled in HPC goes to the last position in a queue that may take several days to start execution. For this reason, HPC is not suited for testing and debugging. This sections shows how to create a test environment and schedule a job.

1.4.1 Test and debug

The GPU is disabled in the login node of **lince**, and this node is used mainly for job schedule. A feasible alternative is to test and debug in Google Colab or Kaggle before submitting to SLURM.

An easy way to distinguish between debug and production modes is the use of and environment variable, for example `DEBUG_MODE`. In debug mode, the code loads a subset of the dataset and runs for a few epochs, since the goal is just make sure the code is error free. Then, in production mode the code processes the entire dataset and full epochs. In production mode you may also want to test several model architectures. The examples in this section show how to test an autoencoder with different number of layers.

Insert this piece of code in the first cell of Google Colab. The second line enables debug mode, and the third and fourth lines are the model configuration.:

```python
import os
os.environ['DEBUG_MODE'] = 'true'
os.environ['AE_ARCH'] = '1_2_4_6_8_10_12'    # Number of filters in each layer
os.environ['AE_KERNEL'] = '5_5_3_3_3_3_3'    # Filter size in each layer
```

You may want to test in VS Code before sending the script to SLURM. This can be done by setting the debug mode as default when you login, and just disable in the schedule script.

Enable debug mode in **lince** inserting this code in your `~/.bashrc` file:

```
export DEBUG_MODE=TRUE          #
export AE_ARCH=1_2_4_6_8_10_12  # Number of filters in each autoencoder layer
export AE_KERNEL=5_5_3_3_3_3_3  # Filter size in each layer
```

Read the debug mode in the Python script body:

```python
# Debug mode is lightweight components to be able to run in Google Colab
debug_mode = os.getenv('DEBUG_MODE').title() in ['True', '1']
```

Set 5 epochs in debug mode and 5000 otherwise:

```python
num_epochs = 5 * debug_mode or 5000
```

1.4.2 Google Colab setup

In the first cell of Colab, insert this code to mount GDrive in Colab:

```
# Mount Google Drive
from google.colab import drive
drive.mount("/content/drive")
```

Now you have access to the scripts in **lince** saved in Drive. Example of code to copy scripts from GDrive to Colab:

```
import os
os.system('cp /content/drive/MyDrive/<FOLDER>/*.py .')
```

Copy trained models from folder `model/<MODEL_NAME>` to Drive:

```
import os, glob

def save_models_GDrive():
  # Create folder to save models
  for fdir in glob.glob('model/*'):
        gdir = '/content/drive/MyDrive/lince-colab/' + fdir.split('/')[1]
        if not os.path.exists(gdir):
          os.system('mkdir {}'.format(gdir))

  # Copy trained models do Google Drive
  for model in glob.glob('model/*/*'):
        fdir = model.split('/')[1]
        fname = model.split('/')[2]
        cmd = 'cp model/{}/{} /content/drive/MyDrive/lince-colab/{}/{}'.format(fdir,␣
↪fname, fdir, fname)
        print(cmd)
        os.system(cmd)
```

1.5 Tensorflow configuration

Reduce training time with proper Tensorflow and system configuration. This section covers the following topics:

1. Install Tensorflow optimized for performance
2. Save the model
3. Data format
4. OpenMP parameters
5. CPU optimization
6. GPU optimization
7. Compiler optimization
8. Distributed computing

1.5.1 Tensorflow installation

Intel created a Tensorflow version optimized for CPU. Use this installation if you won't use GPU:

```
$ conda install tensorflow -c intel
$ pip install intel-tensorflow==2.4.0
```

Read full instructions at Intel website.

1.5.2 Save the model

It's possible to create checkpoints to save the model during training after each epoch, then resume the training from the last checkpoint. This is useful if the training time is larger than the scheduled time, or to prevent hardware failure or broken connection. Saving the model after training for later use is also possible.

Follow the instructions in this tutorial to save the model.

1.5.3 Data format

Tensorflow stores and processes image arrays with the channel in the last dimension (channel last), also known as NHWC. The format used by Intel is channel last, or NCHW. The meaning of each letter is:

- N: Batch size, indicating number of images in a batch.
- C: Channel, indicating number of channels in an image.
- W: Width, indicating number of pixels in horizontal dimension of an image.
- H: Height, indicating number of pixels in vertical dimension of an image.

When training on Intel CPU only, force Tensorflow to use channel first with this code:

```
import tensorflow as tf
# force channels-first ordering
keras.backend.set_image_data_format('channels_first')
```

1.5.4 OpenMP settings

OpenMP implements parallel computing among different processors. Intel recommends the use these environment variables to configure OpenMP. For convenience, save them in your *~/.bashrc* file or setup in Python.

- **OMP_NUM_THREADS**
  - Maximum number of threads to use for OpenMP parallel regions if no other value is specified in the application.
  - Recommend: start with the number of physical cores/socket on the test system, and try increasing and decreasing
- **KMP_BLOCKTIME**
  - Time, in milliseconds, that a thread should wait, after completing the execution of a parallel region, before sleeping.
  - Recommend: start with 1 and try increasing
- **KMP_AFFINITY**

  - Restricts execution of certain threads to a subset of the physical processing units in a multiprocessor computer. Only valid if Hyperthreading is enabled.
  - Recommend: granularity=fine,verbose,compact,1,0
- **KMP_SETTINGS**
  - Enables (TRUE) or disables (FALSE) printing of OpenMP run-time library environment variables during execution
  - Recommend: Start with TRUE to ensure settings are being utilized, then use as needed

Python example:

```
import os
os.environ["OMP_NUM_THREADS"] = "8"  # Number of physical cores
os.environ["KMP_AFFINITY"] = "granularity=fine,compact,1,0"
os.environ["KMP_BLOCKTIME"] = "0"  #(or 1)
os.environ["KMP_SETTINGS"] = "TRUE"
```

1.5.5 CPU optimization

Set the number of CPU cores that Tensorflow can use with these parameters:

- **intra_op_parallelism_threads**
  - Number of threads used within an individual op for parallelism
  - Recommend: start with the number of cores/socket on the test system, and try increasing and decreasing
- **inter_op_parallelism_threads**
  - Number of threads used for parallelism between independent operations.
  - Recommend: start with the number of physical cores on the test system, and try increasing and decreasing
- **device_count**
  - Maximum number of devices (CPUs in this case) to use
  - Recommend: start with the number of cores/socket on the test system, and try increasing and decreasing
- **allow_soft_placement**
  - Set to True/enabled to facilitate operations to be placed on CPU instead of GPU

Example:

```
import tensorflow as tf
tf.config.threading.set_inter_op_parallelism_threads(8)  # Use 8 physical cores
tf.config.threading.set_intra_op_parallelism_threads(8)  # Use 8 physical cores
tf.config.set_soft_device_placement(True)
```

Reference: https://software.intel.com/content/www/us/en/develop/articles/guide-to-tensorflow-runtime-optimizations-for-cpu.html

1.5.6 GPU optimization

Insert this code in Google Colab to make sure GPU is enabled:

```
import tensorflow as tf
# Show available devices: CPU and GPU
print(tf.config.list_physical_devices())

# Check that we are using a GPU, if not switch runtimes
#   using Runtime > Change Runtime Type > GPU
assert len(tf.config.list_physical_devices('GPU')) > 0
```

1.5.7 Compiler optimization

XLA (Accelerated Linear Algebra) is a domain-specific compiler for linear algebra that can accelerate TensorFlow models with potentially no source code changes. Enable XLA in Python or save the environment variable in `` ~/.bashrc``:

```
import os
os.environ['TF_XLA_FLAGS'] = '--tf_xla_enable_xla_devices'
```

1.5.8 Pipeline optimization

Data input pipeline used during training may impact performance. An efficient pipeline reads data from disk for the next batch while the GPU processes the current batch.

See how to achieve better performance with the tf.data API to build an optimized data pipeline.

1.5.9 Distributed computing

Tensorflow can distribute computing in more than one GPU in the same computer or in several servers.

This example shows how to enable distributed computing in one or more GPUs or use the default strategy if no GPU is found:

```
import tensorflow as tf
# Distributed training: GPU settings
if tf.config.list_physical_devices('GPU'):
  strategy = tf.distribute.MirroredStrategy()
else:  # use default strategy
  strategy = tf.distribute.get_strategy()
```

Then use the **strategy** to create the model, optimizer and compile the model:

```
with strategy.scope():
        optimizer = tf.keras.optimizers.SGD()
        model = tf.keras.Sequential([tf.keras.layers.Dense(1, input_shape=(1,))])
        model.compile(loss='mse', optimizer=optimizer)
```

See distributed training with TensorFlow for complete explanation about distributed computing.

1.6 Working with modules

Modules are a convenient way to manage environment variables for applications use. Unless you use the default installation of Anaconda available in HPC, you'll need to create custom modules. This section briefly explains how to work with modules and provides a custom module for Miniconda. See the references[1] to learn more about modules.

Environment modules set environment variables with specific values for each application. Run **module avail** to list all modules available:

```
$ module avail

---------------------------------------------------------------- /opt/ohpc/pub/
↪moduledeps/gnu7-openmpi3 -------------------------------------------------------------
↪--
   adios/1.13.1        imb/2018.1     netcdf-cxx/4.3.0        phdf5/1.10.2        py2-
↪scipy/1.1.0     scalasca/2.3.1    superlu_dist/5.3.0
   boost/1.67.0 (D)    mfem/3.3.2     netcdf-fortran/4.4.4    pnetcdf/1.9.0       py3-
↪mpi4py/3.0.0    scorep/4.0        tau/2.27.1
   fftw/3.3.7          mpiP/3.4.1     netcdf/4.6.1            ptscotch/6.0.4      py3-
↪scipy/1.1.0     sionlib/1.7.1     trilinos/12.12.1
   hypre/2.14.0        mumps/5.1.2    petsc/3.9.1             py2-mpi4py/3.0.0   ␣
↪scalapack/2.0.2     slepc/3.9.1

-------------------------------------------------------------------- /opt/ohpc/pub/
↪moduledeps/gnu7 ---------------------------------------------------------------------
   R/3.5.0    hdf5/1.10.2    mpich/3.2.1    openblas/0.2.20    openmpi3/3.1.0 (L)    py2-
↪numpy/1.14.3    superlu/5.2.1
   gsl/2.4    metis/5.1.0    ocr/1.0.1      openmpi/1.10.7     pdtoolkit/3.25        py3-
↪numpy/1.14.3

---------------------------------------------------------------------- /opt/ohpc/pub/
↪modulefiles ------------------------------------------------------------------------
   EasyBuild/3.8.1           clustershell/1.8    gnu7/7.3.0  (L)    intel/19.0.4.243    ␣
↪   papi/5.6.0        singularity/3.1.0
   autotools            (L)    cmake/3.13.4        gnu8/8.3.0         llvm5/5.0.1         ␣
↪   pmix/2.2.2        valgrind/3.14.0
   charliecloud/0.9.7        gnu/5.4.0           hwloc/2.0.3        ohpc             (L)␣
↪   prun/1.3   (L)

------------------------------------------------------------------------------ /apps/
↪modulefiles -------------------------------------------------------------------------
   Anaconda/2-2019.03        Gromacs/5.1.4-cuda-mpi    Gromacs/2019.3-cuda (D)    amber/
↪19-cpu        canal/1.5      lammps/7Aug19         mkl/2019.4.243
   Anaconda/3-2019.03 (D)    Gromacs/5.1.4-cuda        Gromacs/2019.3-mpi         amber/
↪19-gpu        curves/3.0     lammps/29Oct20 (D)    ox/8.02-0-gnu
   Gromacs/4.0.7             Gromacs/5.1.4-mpi         NAMD/2.13-CUDA             amber/
↪20-gpu (D)    dssp           magma/2.5.1           pgi/19.10
   Gromacs/4.6.7             Gromacs/2018.3-cuda       amber/18                   boost/
↪1_71_0        g_mmpbsa       megacc/10.2.5         relion/3.1

------------------------------------------------------------------------------- /opt/
↪modulefiles -------------------------------------------------------------------------
```

(continues on next page)

[1] References:

(continued from previous page)

```
   cuda/8.0    cuda/10.0    cuda/10.1 (L,D)


  Where:
   D:  Default Module
   L:  Module is loaded

Use "module spider" to find all possible modules.
Use "module keyword key1 key2 ..." to search for all possible modules matching any of␣
↪the "keys".
```

Notice that default modules have a `(D)` besides the name and loaded modules come with a `(L)`. You can load a module with `module load`:

```
$ module load Anaconda/3-2019.03
```

Clean all loaded modules with `module purge`:

```
$ module purge
```

Run `module show` to list the commands executed in the module:

```
$ module show Anaconda/3-2020.11
-----------------------------------------------------------------------------------------
↪-----------------------------------------
   /opt/ohpc/pub/modulefiles/Anaconda/3-2020.11.lua:
-----------------------------------------------------------------------------------------
↪-----------------------------------------
help([[This module loads /scratch/apps/gnu/anaconda3
]])
conflict("Anaconda","Anaconda3","anaconda","python")
setenv("INSTALL_DIR","/scratch/apps/gnu/anaconda3")
prepend_path("LD_LIBRARY_PATH","/scratch/apps/gnu/anaconda3")
prepend_path("LD_LIBRARY_PATH","/scratch/apps/gnu/anaconda3/libexec")
prepend_path("INCLUDE","/scratch/apps/gnu/anaconda3/include")
prepend_path("PATH","/scratch/apps/gnu/anaconda3/sbin")
prepend_path("PATH","/scratch/apps/gnu/anaconda3/bin")
```

1.6.1 Create custom module

You can create custom modules to modify the environment variables. The custom module presented here is a copy of **lince**'s `Anaconda/3-2019` module modified for Miniconda.

Custom modules are saved in the `modulefiles` folder. The folder structure is `modulefiles/<app_name>/<version>`, where `<app_name>` is the application name and `<version>` is the module version. Actually, it may be any name but we use this convention for simplicity.

> $ mkdir -p ~/modulefiles/Miniconda/

Now, use a text editor such as `vim` or `nano` to create the file `1.0` with the following content.:

```
$ cd ~/modulefiles/Miniconda
# use a text editor to create the module file called 1.0 (which is the version)
```

(continues on next page)

(continued from previous page)

```
$ cat 1.0
#%Module###########################################

set INSTALL_DIR /scratch/<YOUR_NUSP>/miniconda3

conflict pyhton  anaconda  Anaconda miniconda Miniconda

prepend-path PATH ${INSTALL_DIR}/bin
prepend-path INCLUDE ${INSTALL_DIR}/include/
prepend-path -d  " " CPPFLAGS   -I${INSTALL_DIR}/include
prepend-path -d " " LDFLAGS -L${INSTALL_DIR}/lib
prepend-path LD_LIBRARY_PATH ${INSTALL_DIR}/lib
prepend-path MANPATH ${INSTALL_DIR}/share/man
```

Where YOUR_NUSP is your user id.

1.6.2 Add the module path to MODULEPATH

Now that the module file has been created, one just needs to add the following line to your `~/.bashrc` file so that it will be found:

```
module use --append /scratch/<USER_ID>/modulefiles/
```

The next time you log in you will be able to run **module avail** or **module load** on the new module.

You also need to add these lines in your SLURM schedule script to load the environment variables:

```
module use --append /scratch/<USER_ID>/modulefiles/
module load Miniconda/1.0
```

1.6.3 Module usage

Just run `module` to list all available options:

```
$ module

Modules based on Lua: Version 7.8.15  2019-01-16 12:46 -06:00
        by Robert McLay mclay@tacc.utexas.edu

module [options] sub-command [args ...]

Help sub-commands:
------------------
  help                              prints this message
  help                module [...]  print help message from module(s)

Loading/Unloading sub-commands:
-------------------------------
  load | add          module [...]  load module(s)
  try-load | try-add  module [...]  Add module(s), do not complain if not found
  del | unload        module [...]  Remove module(s), do not complain if not found
```

(continues on next page)

(continued from previous page)

```
  swap | sw | switch  m1 m2          unload m1 and load m2
  purge                              unload all modules
  refresh                            reload aliases from current list of modules.
  update                             reload all currently loaded modules.

Listing / Searching sub-commands:
---------------------------------
  list                               List loaded modules
  list                s1 s2 ...      List loaded modules that match the pattern
  avail | av                         List available modules
  avail | av          string         List available modules that contain "string".
  spider                             List all possible modules
  spider              module         List all possible version of that module file
  spider              string         List all module that contain the "string".
  spider              name/version   Detailed information about that version of the␣
↪module.
  whatis              module         Print whatis information about module
  keyword | key       string         Search all name and whatis that contain "string".

Searching with Lmod:
--------------------
  All searching (spider, list, avail, keyword) support regular expressions:


  -r spider           '^p'           Finds all the modules that start with `p' or `P'
  -r spider           mpi            Finds all modules that have "mpi" in their name.
  -r spider           'mpi$          Finds all modules that end with "mpi" in their name.

Handling a collection of modules:
--------------------------------
  save | s                           Save the current list of modules to a user defined
↪"default" collection.
  save | s            name           Save the current list of modules to "name"␣
↪collection.
  reset                              The same as "restore system"
  restore | r                        Restore modules from the user's "default" or system␣
↪default.
  restore | r         name           Restore modules from "name" collection.
  restore             system         Restore module state to system defaults.
  savelist                           List of saved collections.
  describe | mcc      name           Describe the contents of a module collection.
  disable             name           Disable a collection.

Deprecated commands:
--------------------
  getdefault          [name]         load name collection of modules or user's "default"␣
↪if no name given.
                                                                    ===> Use "restore
↪" instead <====
  setdefault          [name]         Save current list of modules to name if given,␣
↪otherwise save as the default list for you the
                                                                    user.
```

(continues on next page)

(continued from previous page)

```
                                                                          ===> Use "save"␣
↪instead. <====

Miscellaneous sub-commands:
---------------------------
  is-loaded             modulefile    return true if module is loaded
  is-avail              modulefile    return true if module can be loaded
  show                  modulefile    show the commands in the module file.
  use [-a]              path          Prepend or Append path to MODULEPATH.
  unuse                 path          remove path from MODULEPATH.
  tablelist                           output list of active modules as a lua table.

Important Environment Variables:
--------------------------------
  LMOD_COLORIZE                       If defined to be "YES" then Lmod prints properties␣
↪and warning in color.

        ----------------------------------------------------------------------------------
↪-----------------------------------------------

Lmod Web Sites

  Documentation:    http://lmod.readthedocs.org
  Github:           https://github.com/TACC/Lmod
  Sourceforge:      https://lmod.sf.net
  TACC Homepage:    https://www.tacc.utexas.edu/research-development/tacc-projects/lmod

  To report a bug please read http://lmod.readthedocs.io/en/latest/075_bug_reporting.html
        ----------------------------------------------------------------------------------
↪-----------------------------------------------
```

https://researchcomputing.princeton.edu/support/knowledge-base/modules

https://researchcomputing.princeton.edu/support/knowledge-base/custom-modules

1.7 Slurm

Slurm Workload Manager is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters. It is used by many of the world's supercomputers and computer clusters.

Slurm manages the amount of resources allocated to each job. The number of nodes, CPU cores, memory, GPUs and period are examples of resources one can allocate to a particular job. This is ideal for distributed computing among several nodes.

Machine learning and deep learning models can be trained in HPC with Tensorflow, PyTorch, Dask or other distributed computing library.

Read the official Quick Start User Guide for an overview of the architecture, commands and examples.

Princeton Research Computing also provides a good introduction to Slurm.

1.7.1 Commands

This introductory video shows some useful commands.

https://youtu.be/U42qlYkzP9k

Here's a list of some commonly used user commands. See Slurm man pages for a complete list of commands or download the `command summary PDF`. Note that all Slurm commands start with **'s'**.

Command	Description
sbatch <slurm_script>	Submit a job script for later execution.
scancel <jobid>	Cancel a pending or running job or job step
srun	Parallel job launcher (Slurm analog of mpirun)
squeue	Show all jobs in the queue
squeue -u <username>	Show jobs in the queue for a specific user
squeue –start	Report the expected start time for pending jobs
squeue -j <jobid>	Show the nodes allocated to a running job
scontrol show config	View default parameter settings
sinfo	Show cluster status

1.7.2 Job schedule

Submit a script to the queue with `sbatch <script>`:

```
$ sbatch script.sh
```

The options of `sbatch` command may be inserted into the script following the `#SBATCH` directive:

```
$ cat script.sh
#!/bin/bash -v
#SBATCH --partition=GPUSP4      # partition name. lince = 'GPUSP4', aguia = 'SP2'
#SBATCH --job-name=tr-ae        # job name
#SBATCH --nodes=1               # number of nodes allocated for this job
#SBATCH --ntasks=2              # total number of tasks / mpi processes
#SBATCH --cpus-per-task=8       # number OpenMP Threads per process
#SBATCH --time=08:00:00         # total run time limit ([[D]D-]HH:MM:SS)
#SBATCH --gres=gpu:tesla:2      # number of GPUs
# Get email notification when job begins, finishes or fails
#SBATCH --mail-type=ALL         # type of notification: BEGIN, END, FAIL, ALL
#SBATCH --mail-user=your@mail   # e-mail address

# OpenMP settings used for parallel processing.
# Check your library documentation for custom configuration (Tensorflow, PyTorch, Dask,␣
↪etc)
# Reference: https://www.openmp.org/spec-html/5.0/openmpch6.html#openmpse50.html
export OMP_NUM_THREADS=1
export MKL_NUM_THREADS=1
export OMP_PLACES=threads
export OMP_PROC_BIND=spread

# Slurm controller sets these variables in the environment of the batch script
# More variables at https://slurm.schedmd.com/sbatch.html#lbAK
echo $SLURM_JOB_ID              # ID of job allocation
```

(continues on next page)

(continued from previous page)

```
echo $SLURM_SUBMIT_DIR          # The directory from which sbatch was invoked
echo $SLURM_JOB_NODELIST        # List of nodes allocated to the job
echo $SLURM_NTASKS              # Total number of cores for job

# Load modules. Use "module avail" to list available modules.
# This example loads a custom module.
module use --append /scratch/11568881/modulefiles/
module load Miniconda/1.0

# Run the application.
echo [`date '+%Y-%m-%d %H:%M:%S'`] Running $AE_ARCH
srun <train_model.py>
```

UiT The Arctic University of Norway provides additional job script examples.

1.8 GPU Usage

Each **lince** server has 2 NVIDIA Tesla GPUs installed. This cluster should be used to run GPU jobs; if you don't need GPU then use **aguia** instead.

You may check the GPU usage with the `nvidia-smi` command in any server except the login server which is used only for job scheduling.

Example of GPU using 67MiB of memory usage, 74% and 0% utilization on GPU 0 and 1 respectively and process ID 29150:

```
$ nvidia-smi
Sun May  2 16:54:50 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.67       Driver Version: 418.67       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K20m          Off  | 00000000:05:00.0 Off |                    0 |
| N/A   68C    P0   106W / 225W |     78MiB /  4743MiB |     74%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K20m          Off  | 00000000:83:00.0 Off |                    0 |
| N/A   30C    P8    16W / 225W |      0MiB /  4743MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+


+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0     29150      C   ...gramas/intel/gromacs-5.1.4-cuda/bin/gmx    67MiB |
+-----------------------------------------------------------------------------+
```

Message "No running processes found" on idle GPUs:

```
$ nvidia-smi
Sun May  2 16:55:18 2021
```

(continues on next page)

(continued from previous page)

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.67       Driver Version: 418.67       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K20m          Off  | 00000000:05:00.0 Off |                    0 |
| N/A   36C    P8    25W / 225W |     11MiB /  4743MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K20m          Off  | 00000000:83:00.0 Off |                    0 |
| N/A   28C    P8    16W / 225W |      0MiB /  4743MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
$
```

Example of error message:

```
$ nvidia-smi
Unable to determine the device handle for GPU 0000:05:00.0: GPU is lost.  Reboot the␣
↪system to recover this GPU
```

1.8.1 Check all servers

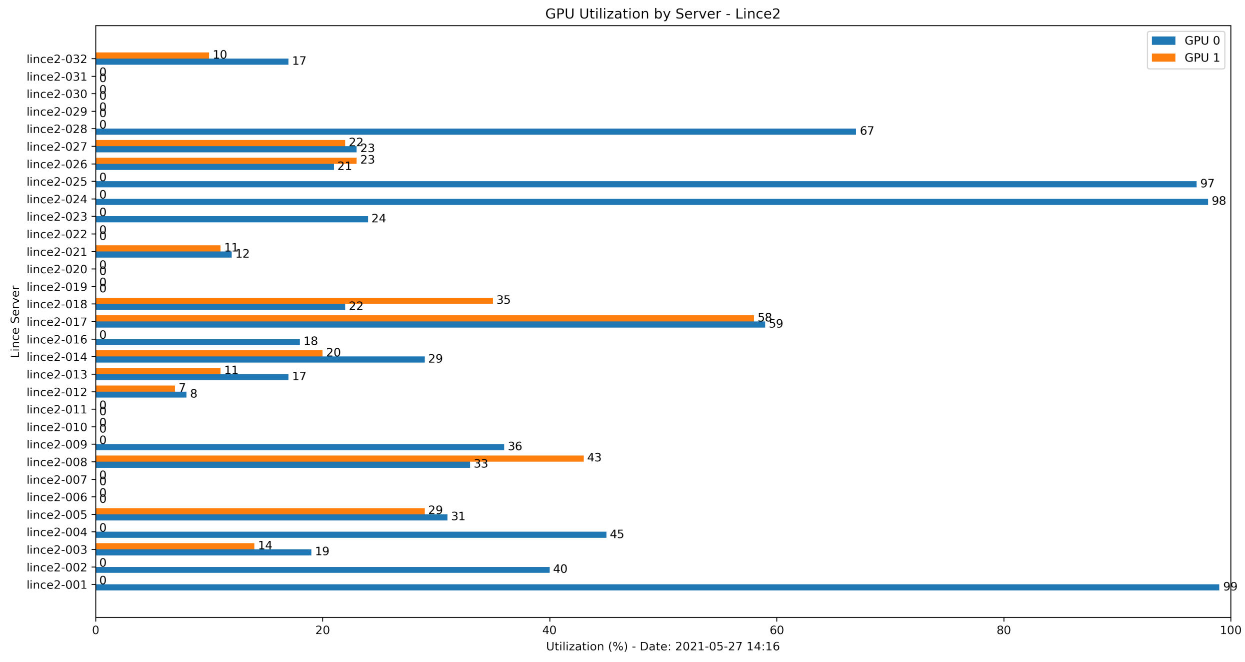
A more practical way to check the GPU usage in all servers is using a script. The `gpu_mon.py` script connects to each server and checks the GPU status. Then, it prints a list of servers with idle and falty GPUs, creates a bar plot and sends an e-mail with GPU usage:

```
$ python gpu_mon.py
Number of servers: 32

Two GPUs in use: 13 servers.
------------------------------
lince2-003
lince2-005
lince2-008
lince2-012
lince2-013
lince2-014
lince2-017
lince2-018
lince2-020
lince2-021
lince2-026
lince2-027
lince2-032
```

(continues on next page)

(continued from previous page)

```

One GPUs in use: 10 servers.
------------------------------
lince2-001
lince2-002
lince2-004
lince2-009
lince2-011
lince2-016
lince2-023
lince2-024
lince2-025
lince2-028

No GPUs in use: 8 servers.
------------------------------
lince2-006
lince2-007
lince2-010
lince2-019
lince2-022
lince2-029
lince2-030
lince2-031

Faulty GPUs: 0 servers.
------------------------------

Connection failure: 1 servers.
------------------------------
lince2-015
```



gpu_mon script:

```
#!/scratch/11568881/miniconda3/bin/python
#%%
"""
This module collects GPU utilization on all servers in lince cluster. This is useful to␣
↪help
identify possible improvements in job speed and free resources for other users.
Ideally GPU utilizatin should be high for the most part of the time.

Process:
1. Connect to all servers via SSH and collect GPU usage.
2. Create a data frame with server and both GPUs usage.
3. Create a horizontal bar chart of GPU usage by server.
4. Send a summary and plot by e-mail.
"""
import os, re, datetime
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import smtplib, mimetypes
from email.message import EmailMessage

def message(msg, servers):
        """Format server status message."""
        text = '\n'
        text += msg + str(len(servers)) + " servers.\n"
        text += "-"*30 + "\n"
        for server in servers:
                text += server + '\n'
        return text

def gpustatus(result_fname, summary_fname):
        """Connect to each server and collect GPU information.
Result is saved in a log file.
"""
        gpu = {}        # GPU utilization
        no_gpu = []     # Servers with 0 GPUs in use
        one_gpu = []    # Servers with 1 GPU in use
        two_gpu = []    # Servers with 2 GPUs in use
        gpudown = []    # Servers with faulty GPUs
        no_route = []   # Servers with connection failure
        servers = []    # List of servers
        df = pd.DataFrame(columns = ['Server', 'GPU 0', 'GPU 1'])

        for n in range(1, 33):
                # Connect to each server in the cluster and send commands
                server_name = 'lince2-' + ("000" + str(n))[-3:]
                servers.append(server_name)
                cmd = 'ssh {} "hostname;nvidia-smi"'.format(server_name)
                pipe = os.popen(cmd,'r')

                print("Processing server:", server_name)
```

(continues on next page)

(continued from previous page)

```
                for row in pipe.read().split('\n'):
                        #lince2-001.hpc.usp.br
                        server_re = re.search(r'(lince\d-(\d+))\.hpc', row)
                        #|   0  Tesla K20m          Off  | 00000000:05:00.0 Off |        ␣
↪             0 |
                        gpuId_re = re.search(r'\|\s+(\d)\s+Tesla', row)
                        #| N/A   62C    P0    104W / 225W |      78MiB /  4743MiB |     73
↪%       Default |
                        utilization_re = re.search(r'B \|\s+(\d+)%\s+', row)

                        # Read server name
                        if server_re:
                                server = server_re.group(1)
                        # Read GPU error message
                        elif "Unable to determine the device handle for GPU" in row:
                                gpudown.append(server)
                        # Read GPU ID: 0 or 1
                        elif gpuId_re:
                                gpuId = int(gpuId_re.group(1))      # GPU 0 or 1
                        # Read GPU utilization
                        elif utilization_re:
                                gpu[gpuId] = int(utilization_re.group(1))
                                if gpuId:
                                        df.loc[len(df) + 1] = server, gpu[0], gpu[1]
                                        # Identify number of GPUs in use
                                        if not (gpu[0] or gpu[1]):
                                                no_gpu.append(server)   # 0 GPUs in use
                                        elif gpu[0] and gpu[1]:
                                                two_gpu.append(server)  # 1 GPU in use
                                        else:
                                                one_gpu.append(server)  # 2 GPUs in use

                pipe.close()

        # Connection failure
        checked_servers = two_gpu +  one_gpu +  no_gpu +  gpudown
        for server in servers:
                if not server in checked_servers:
                        no_route.append(server)
        checked_servers += no_route
        # Summary of GPU usage
        n =  len(checked_servers)
        summary = "Number of servers: {} \n".format(str(n))
        summary += message("Two GPUs in use: ", two_gpu)
        summary += message("One GPUs in use: ", one_gpu)
        summary += message("No GPUs in use: ", no_gpu)
        summary += message("Faulty GPUs: ", gpudown)
        summary += message("Connection failure: ", no_route)

        print(summary)
        # Save data frame and summary
        df.to_csv(result_fname)
```

(continues on next page)

(continued from previous page)

```
        with open(summary_fname, 'w') as f:
                f.write(summary)

        return

def create_plot(result, plot):
        """Create plot of GPU usage per server."""
        df = pd.read_csv(result)
        # Create plot
        x = np.arange(len(df['Server']))  # the label locations
        width = 0.35  # the width of the bars

        fig, ax = plt.subplots(figsize=(15,8))
        rects1 = ax.barh(x - width/2, df['GPU 0'], width, label='GPU 0')
        rects2 = ax.barh(x + width/2, df['GPU 1'], width, label='GPU 1')

        # Add some text for labels, title and custom x-axis tick labels, etc.
        now = datetime.datetime.now()
        dt = now.strftime("%Y-%m-%d %H:%M")

        ax.set_title('GPU Utilization by Server - Lince2')
        ax.set_xlabel('Utilization (%) - Date: {}'.format(dt))
        ax.set_xlim(0, 100)
        ax.set_ylabel('Lince Server')
        ax.set_yticks(x)
        ax.set_yticklabels(df['Server'])
        ax.legend()

        ax.bar_label(rects1, padding=3)
        ax.bar_label(rects2, padding=3)

        fig.tight_layout()

        # Save and show plot
        plt.savefig(plot, dpi=300, bbox_inches='tight')
        plt.show()

        return

def send_email(receiver, message, plot_fname):
    """Send email with summary of GPU usage and plot."""
    # Create message and set text content
    sender = 'no-reply@lince2.hpc.usp.br'
    msg = EmailMessage()
    msg['Subject'] = 'Lince: GPUs Status'
    msg['From'] = sender
    msg['To'] = receiver

    # Message content
    body = """*** Automatic e-mail, do not reply. ***

Status of lince servers.
```

(continues on next page)

(continued from previous page)

```
See attached plot.

"""
   body += message
   msg.set_content(body)

   # Attach plot
   with open(plot_fname, 'rb') as fp:
          file_data = fp.read()
          maintype, _, subtype = (mimetypes.guess_type(plot_fname)[0] or 'application/
→octet-stream').partition("/")
          msg.add_attachment(file_data, maintype=maintype, subtype=subtype,␣
→filename=plot_fname)

   # Send e-mail
   with smtplib.SMTP('localhost') as server:
          server.sendmail(sender, receiver, msg.as_string())
          print("Successfully sent email")


if __name__ == '__main__':
        now = datetime.datetime.now()
        dt = now.strftime("%Y%m%d_%H%M")
        plot_fname = 'plot/gpu_status_{}.png'.format(dt)
        result_fname = 'gpu.csv'
        summary_fname = 'summary.txt'

        # 1. Execute GPU checks
        gpustatus(result_fname, summary_fname)

        # 2. Create plot
        create_plot(result_fname, plot_fname)

        # 3. Send result by e-mail
        receiver = 'your@email.com'
        with open(summary_fname, 'r') as f:
                rows = f.readlines()
                msg = ''
                for row in rows:
                        msg += row
        send_email(receiver, msg, plot_fname)

        # 4. Delete files
        for fname in [plot_fname, result_fname, summary_fname]:
                os.unlink(fname)
```

1.9 Useful Linux Commands

1.9.1 Manual pages

The man command provides an online reference manual of any command. For example, use this command to see reference of ls:

```
$ man ls
```

1.9.2 Linux processes

The top command shows CPU and memory usage, running processes and other information. It provides a dynamic view of the sytem usage:

```
$ top
top - 14:32:01 up 546 days, 5 min,  6 users,  load average: 0.40, 0.33, 0.27
Tasks: 351 total,   1 running, 331 sleeping,  17 stopped,   2 zombie
%Cpu(s):  0.2 us,  0.2 sy,  0.0 ni, 97.0 id,  2.6 wa,  0.0 hi,  0.1 si,  0.0 st
KiB Mem : 13182566+total,  2305124 free, 78298656 used, 51221880 buff/cache
KiB Swap:        0 total,        0 free,        0 used. 49155016 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 6763 ganglia   20   0  406312 131972   1376 S   2.7  0.1  17495:02 gmond
 6695 ganglia   20   0  632268  15936    396 S   1.8  0.0  21331:43 gmetad
30101 11568881  20   0  162268   2568   1612 R   1.4  0.0   0:00.07 top
11222 mysql     20   0   54.8g  49.1g   4480 S   0.9 39.0   1790:49 mysqld
        1 root      20   0  193924   6028   1728 S   0.0  0.0 295:43.67 systemd
        2 root      20   0       0      0      0 S   0.0  0.0   0:13.83 kthreadd
        3 root      20   0       0      0      0 S   0.0  0.0 402:24.01 ksoftirqd/0
        8 root      rt   0       0      0      0 S   0.0  0.0   0:25.18 migration/0
        9 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
   10 root      20   0       0      0      0 S   0.0  0.0 544:55.47 rcu_sched
   11 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 lru-add-drain
   12 root      rt   0       0      0      0 S   0.0  0.0   3:27.55 watchdog/0
   13 root      rt   0       0      0      0 S   0.0  0.0   3:18.02 watchdog/1
   14 root      rt   0       0      0      0 S   0.0  0.0   1:22.39 migration/1
   15 root      20   0       0      0      0 S   0.0  0.0 370:18.79 ksoftirqd/1
```

1.9.3 Running in background

Connection may drop for several reasons, including computer hibernation. When this occurs, Linux terminates all processes running in the disconnected terminal and you have to run them again. The simplest way to keep the process running is sending to background by adding & to the command:

```
$ sleep 60 &
```

A message appears when the process ends:

```
[1]+  Done                    sleep 60
```

To send a running process to the background, suspend execution with <CTRL> + z and execute bg to send to background:

```
$ sleep 60
^Z
[1]+  Stopped                 sleep 60
$ bg
[1]+ sleep 60 &
```

Bring back to foreground with fg.

1.9.4 screen

The screen command opens a virtual terminal that keeps running even when the main terminal is disconnected. Another real terminal can connect to it, which makes it useful when installing packages.

Steps to use screen:

1. Open a virtual terminal with the screen command.
2. Run any command or script in this terminal.
3. While the command is running, disconnect the terminal with <CTRL> + a and <CTRL> + d.
4. Run screen -ls to list all virtual terminals.
5. Reconnect with screen -r.

Common options:

- <CTRL> + a c Create a new window (with shell)
- <CTRL> + a " List all window
- <CTRL> + a 0 Switch to window 0 (by number)
- <CTRL> + a A Rename the current window
- <CTRL> + a S Split current region horizontally into two regions
- <CTRL> + a | Split current region vertically into two regions
- <CTRL> + a tab Switch the input focus to the next region
- <CTRL> + a <CTRL> + a Toggle between the current and previous region
- <CTRL> + a Q Close all regions but the current one
- <CTRL> + a X Close the current region

1.10 Resources for Researchers

1.10.1 Systematic literature review

Parsifal is an online tool designed to support researchers to perform systematic literature reviews within the context of Software Engineering. Geographically distributed researchers can work together within a shared workspace, designing the protocol and conducting the research.

As well as providing a way to document the whole process, the tool will help you remind what is important during a systematic literature review. During the planning phase, Parsifal will help you with the objectives, PICOC, research questions, search string, keywords and synonyms, selecting the sources, the inclusion and exclusion criterias. Will also provide mechanisms to build a quality assessment checklist and data extraction forms.

During the conducting phase, you will be able to import bibtex files and select the studies, find duplicates among all the different sources, execute the quality assessment and extract data from the papers.

1.10.2 Advanced Information Research Skills (AIRS)

Advanced Information Research Skills (AIRS) is a coursework for Higher Degree Research (HDR) students enrolled in a Doctor of Philosophy (PhD) or Master of Philosophy (MPhil) at Queensland University of Technology (QUT), Australia. The curriculum content is openly accessible, creative commons licensed and available for use by all.

The curriculum includes:

- formulating a good research question
- advanced search strategies
- sourcing and evaluating quality literature
- bibliographic and data management
- note-taking strategies
- citation analysis and research impact
- collaboration tools
- authorship and academic integrity
- publishing and pathways.

https://youtu.be/Z1mBTHw3xVo

1.10.3 Researcher Academy

Researcher Academy provides free access to countless e-learning resources designed to support researchers on every step of their research journey. Browse our extensive module catalogue to uncover a world of knowledge, and earn certificates and rewards as you progress.

RESEARCH PREPARATION

- Funding
- Research data management
- Research collaborations

WRITING FOR RESEARCH

- Fundamentals of manuscript preparation
- Writing skills
- Technical writing skills
- Book writing

PUBLICATION PROCESS

- Fundamentals of publishing
- Finding the right journal

- Ethics
- Open science
- How to publish in premium journals
- Publishing in the Chemical Sciences

NAVIGATING PEER REVIEW

- Certified Peer Reviewer Course
- Fundamentals of peer review
- Becoming a peer reviewer
- Going through peer review

COMMUNICATING YOUR RESEARCH

- Social impact
- Ensuring visibility
- Inclusion and Diversity for Researchers

1.10.4 Reference managers

Reference managers help collect, organize and share references and create citations in various formats. Mendeley and Zotero are free reference managers.

Mendeley

Mendeley simplifies your workflow, so you can focus on achieving your goals.

Zotero

Zotero is a free, easy-to-use tool to help you collect, organize, cite, and share research.

1.10.5 Datasets

Mendeley

Mendeley Data is a secure cloud-based repository where you can store your data, ensuring it is easy to share, access and cite, wherever you are.

Search 28.1 million datasets from domain-specific and cross-domain repositories.

OpenML

The Open Machine Learning is a public repository for machine learning data and experiments, that allows everybody to upload open datasets. It integrates with scikit-learn.

https://youtu.be/1N3qATxXrpE

Example:

```
from sklearn import ensemble
from openml import tasks, flows, Runs

task = tasks.get_task(3954)
clf = ensemble.RandomForestClassifier()
flow = flows.sklearn_to_flow(clf)
run = runs.run_flow_on_task(task, flow)
result = run.publish()
```

Key features:

- Query and download OpenML datasets and use them however you like
- Build any sklearn estimator or pipeline and convert to OpenML flows
- Run any flow on any task and save the experiment as run objects
- Upload your runs for collaboration or publishing
- Query, download and reuse all shared runs

Tensorflow Datasets

Tensorflow Datasets (TFDS) provides a collection of ready-to-use datasets for use with TensorFlow, Jax, and other Machine Learning frameworks.

https://www.tensorflow.org/datasets/catalog/overview

https://youtu.be/-nTe44WT0ZI

Google Research

Google periodically releases data of interest to researchers in a wide range of computer science disciplines.

https://research.google/tools/datasets/

Google dataset search

Google provides a search engine for datasets. Discover datasets hosted in thousands repositories.

https://datasetsearch.research.google.com/

PyTorch

Torch Audio: https://pytorch.org/audio/stable/datasets.html

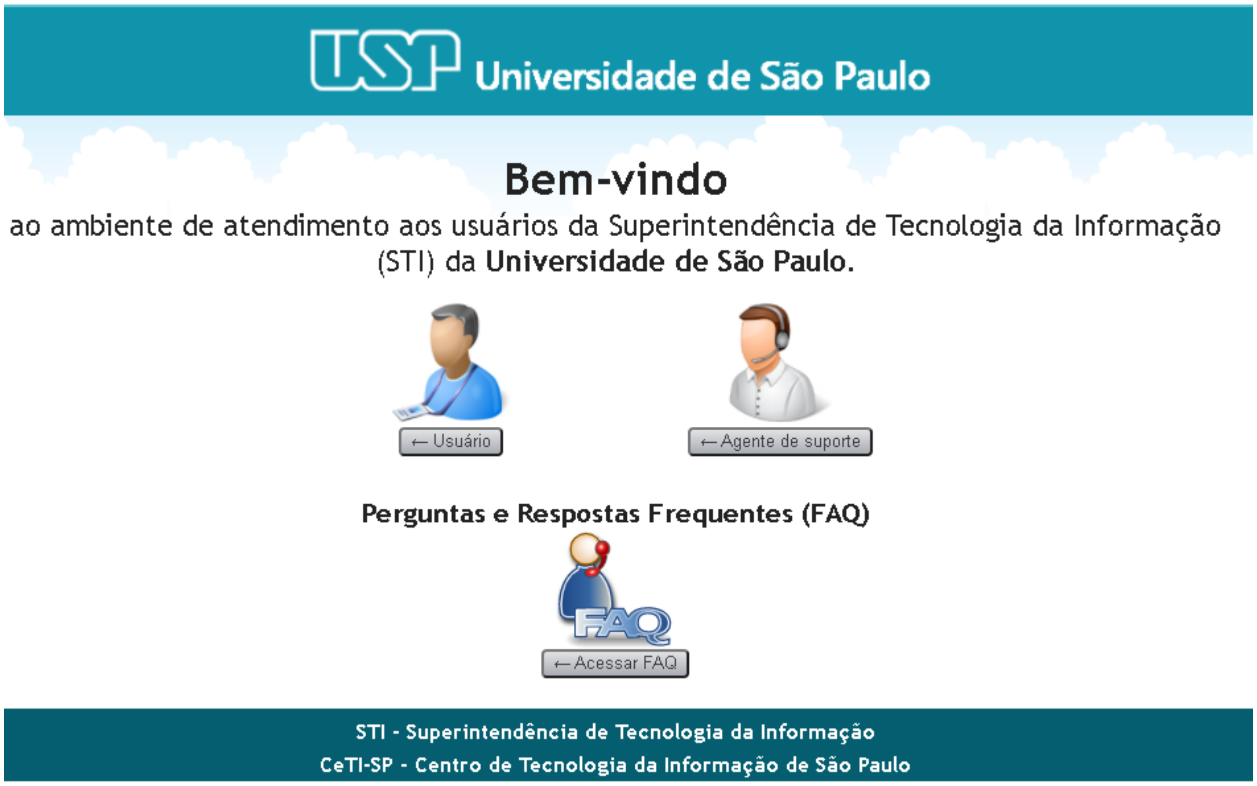
Torchvision: https://pytorch.org/vision/stable/datasets.html

Torch text: https://pytorch.org/text/stable/datasets.html

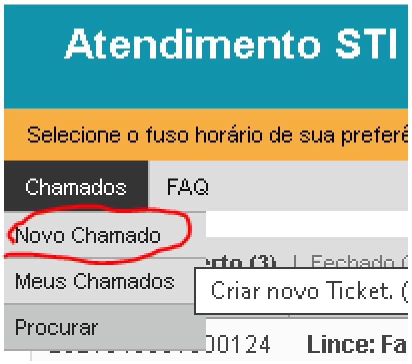
1.11 Getting Support

Before getting support, check the FAQ page.

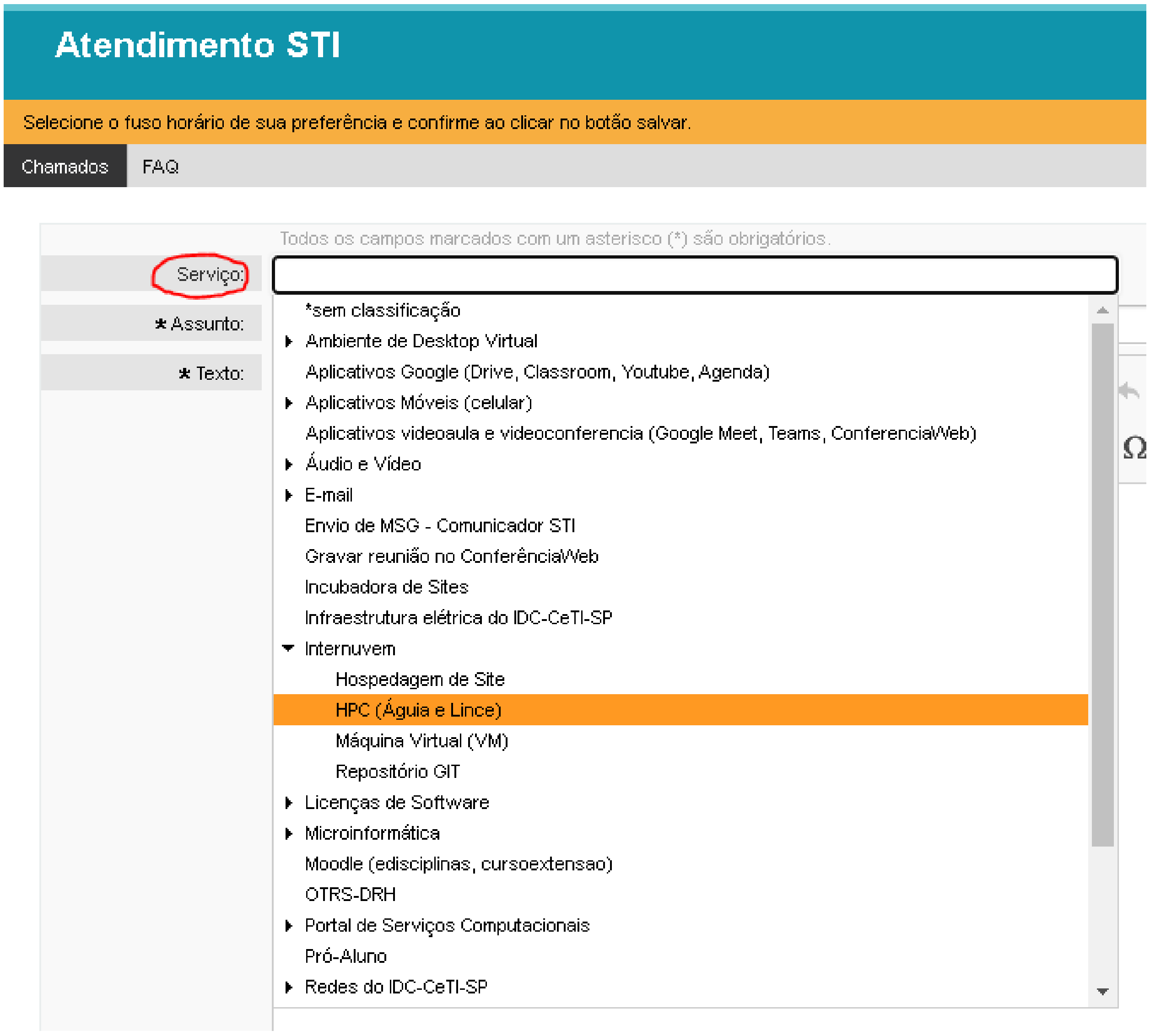
If you need support for shark, aguia or lince, access the support website and click in **Usuário**:


USP Universidade de São Paulo
Bem-vindo
ao ambiente de atendimento aos usuários da Superintendência de Tecnologia da Informação (STI) da Universidade de São Paulo.
← Usuário
← Agente de suporte
Perguntas e Respostas Frequentes (FAQ)
FAQ
← Acessar FAQ
STI - Superintendência de Tecnologia da Informação
CeTI-SP - Centro de Tecnologia da Informação de São Paulo

Then log into **sistema USP** and click “Chamados->Novo Chamado”



Under “Serviço:” select “Internuvem->HPC (Águia e Lince)”


Atendimento STI

Selecione o fuso horário de sua preferência e confirme ao clicar no botão salvar.

Todos os campos marcados com um asterisco (*) são obrigatórios.

Serviço:

* Assunto:

* Texto:

*sem classificação
Ambiente de Desktop Virtual
Aplicativos Google (Drive, Classroom, Youtube, Agenda)
Aplicativos Móveis (celular)
Aplicativos videoaula e videoconferencia (Google Meet, Teams, ConferenciaWeb)
Áudio e Vídeo
E-mail
Envio de MSG - Comunicador STI
Gravar reunião no ConferênciaWeb
Incubadora de Sites
Infraestrutura elétrica do IDC-CeTI-SP
Internuvem
Hospedagem de Site
HPC (Águia e Lince)
Máquina Virtual (VM)
Repositório GIT
Licenças de Software
Microinformática
Moodle (edisciplinas, cursoextensao)
OTRS-DRH
Portal de Serviços Computacionais
Pró-Aluno
Redes do IDC-CeTI-SP

1.12 About

This is the unofficial guide to High-Performance Computing clusters at Universidade de Sao Paulo. I created this document while configuring the environment to train a machine learning model during my master's degree research. It's quick start guide to help new users setup the development environment, schedule jobs and train ML/DL models fast. For detailed information on each topic, I suggest searching the Internet.

LinkedIn: https://www.linkedin.com/in/victor-ivamoto/

GitHub: https://github.com/vivamoto/

RPubs: https://rpubs.com/vsi

Tableau: https://public.tableau.com/profile/victor.s.ivamoto#!/

1.13 License