Big Data Analytics 732A54 and TDDE31

Technical Introduction

Erik Rosendal

Based on slides by Maximilian Pfundstein

LINKÖPING UNIVERSITY

Deadline for lab groups today!

Do not forget to sign up to lab groups in WebReg

732A54:
https://www.ida.liu.se/webreg3/732A54-2021-1/LAB/

TDDE31:
https://www.ida.liu.se/webreg3/TDDE31-2021-1/LAB/

Aims

This presentation should give you some hints how to use the NSC Sigma cluster along with some theoretical and practical information.

The aim of the labs is not only to learn PySpark, but also to learn how to connect to a cluster and give you an opportunity to broaden your technical knowledge.

This introduction does not cover the programming part of PySpark.

Table of Contents

- Theoretical Introduction
  - Linux Systems
  - Shells
  - Virtual Environments and Modules
  - Apache Spark and PySpark
- git
- Practical Introduction
  - Secure Shell & Keys
  - Connecting
  - Developing
  - Submit a job

Linux Systems

Theoretical Introduction

Linux Systems

- Prefer using the CLI rather than GUIs, simplifies the "how-to" long-term
- ThinLinc is available for the most parts of your labs

Shells

Theoretical Introduction

Shells

- The Terminal is the application, the shell the actual interactor
- Command line shells:
  - sh
  - bash (default on most Linux systems)
  - cmd.exe (default on Windows)
  - zsh (default on macOS since Catalina)

Virtual Environments and Modules

Theoretical Introduction

Virtual Environments and Modules

- There exist programs, that set up environments (venv) or modules for you
  - `module`: http://modules.sourceforge.net/
  - `conda`: https://www.anaconda.com/
- Modules are actually doing a bit more, but this will not be part of this introduction
- https://www.nsc.liu.se/software/modules/

Apache Spark and PySpark

Theoretical Introduction

Apache Spark and PySpark

- Apache Spark is written in Java and thus needs the Java JVM to run
- APIs are available for Scala, Java, SQL, Python, R
- This course uses Python and therefore the PySpark API
- Stand-alone and cluster mode
- https://spark.apache.org/docs/2.4.3/
- https://spark.apache.org/docs/2.4.3/api/python/index.html

git

Introduction

git

- git is a distributed source version-control system
- git is *distributed* and *decentralized*
  - GitHub, **GitLab**, Gitea, bitbucket etc are "always running" clients
- git is already installed on unix systems
- Windows: Must install it manually
  - https://git-scm.com/download/win
- We will cover some basics in Demo 2 and 4

git

- "Forking" is copying a repository on a hosted git-instance from one user to another
- Make private then grant access rights to:
  - Lab Partner
  - Lab Assistants
  - Teachers
- Read the readme and Lab Compendium!

git

- The lab is hosted on a self-hosted GitLab instance
  - https://gitlab.liu.se/olaha93/bigdata
- Fork it to your repository
- Bring a copy to your local machine
  - SSH
    - git clone git@gitlab.liu.se: *liuID*/bigdata.git
  - HTTPS
    - git clone https://gitlab.liu.se/*liuID*/bigdata
  - Download as zip file (or other format)

Demo 1

GitLab and forking repository

LINKÖPING UNIVERSITY

git

- git add {file}
  - -A (stages all files)
- git commit -m “Some informative comment"
- git pull origin master
- git push origin master

git

- Merge conflicts happen and are normal!
  - You can prevent them by not working on the same file, for example by pair programming
- If it happens: Open the conflicted files, search for the conflict, solve it
  - `git mergetool`
- Then stage, commit and eventually push the file

git

- Demos will use CLI, but there are GUI clients for git
- GUI clients:
  - GitKraken
  - SourceTree
  - Sublime Merge
  - and many more…

Secure Shell & Keys

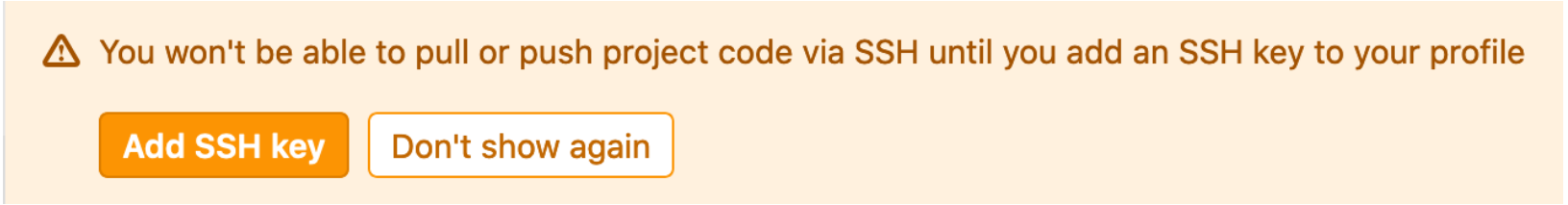
Practical Introduction

Secure Shell & Keys

- Enables creating a remote secure shell, a tunnel
- Can do forward and backwards forwarding
- As well as x-forwarding
- Uses a keypair of a public and a private key, default location is `.ssh`. Unix systems have a default key pair which you can use.
- If not: `ssh-keygen`
- On Windows (e.g. PuTTY) you must create them on your own or use WSL

Secure Shell & Keys

- git can use https or ssh as the underlying protocol
- ssh uses key pairs instead of username and password
- If you log into any git system (GitHub, GitLab) the first time, they usually want you to upload your **public** key for authentication



```
(base) ➜  .ssh cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC5o2fgA3WMD0IsadxA07xcm/PyCdfqddRm8xC/D
E6jWZjYdRcf2UbrckBx78VJcSpxf8PiWxQBwOrsXgZa6Qp7z6GOYja03EOHux8m2ERX0D+0+UVnKR
LepiHtwlMjbWCShck1hrrzRA5BQ/MSYW41hTZ78+IPO8aeYogkH97RAscD2HiX/oMPlkRxJAl7taj
tGEApKQeAGJUggRFs3D8K20RFLOng1iwlMz7Or3uz10pTK+ABAQAuRoEXorRrbQFWNZ1wMb9cRELY
BQZhpG7EEEiXDCNgcxUnB2c8ns9DDyl0OcfsmgVyHe3SegQkmxYMy07fp88pQbupGECCtcJZ flen
nic@Tridir
```

Add an SSH key

To add an SSH key you need to generate one or use an existing key.

Key

Paste your public SSH key, which is usually contained in the file '~/.ssh/id_ed25519.pub' or '~/.ssh/id_rsa.pub' and begins with 'ssh-ed25519' or 'ssh-rsa'. Don't use your private SSH key.

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQC5o2fgA3WMD0IsadxA07xcm/PyCdfqddRm8xC
/DE6jWZjYdRcf2UbrckBx78VJcSpxf8PiWxQBwOrsXgZa6Qp7z6GOYja03EOHux8m2ERX0D+0
+UVnKRLepiHtwlMjbWCShck1hrrzRA5BQ/MSYW41hTZ78+IPO8aeYogkH97RAscD2HiX
/oMPlkRxJAl7tajtGEApKQeAGJUggRFs3D8K20RFLOng1iwlMz7Or3uz10pTK+ABAQAuRoEXorR
rbQFWNZ1wMb9cRELYBQZhpG7EEEiXDCNgcxUnB2c8ns9DDyl0OcfsmgVyHe3SegQkmxYMy
07fp88pQbupGECCtcJZ flennic@Tridir

Title

flennic@Tridir

Expires at

dd / mm / 2021

Give your individual key a title

Add key

Demo 2

SSH setup and git

LINKÖPING UNIVERSITY

Connecting to Sigma

Practical Introduction

Connecting

- Request Project Membership at SNIC/NSC
  - Project is "LiU-compute-2021-4"
  - https://supr.snic.se/project/request/?search=LiU-compute-2021-4
- Request a login account for Sigma
  - https://supr.snic.se/ login with SWAMID
  - Choose Linköping University, use liuID to log in
- General info about Sigma: https://www.nsc.liu.se/systems/sigma

Connecting

- CLI (SSH)
- ThinLinc
- GUI (SSH, X-Forwarding)
- More Information for GUI: https://www.nsc.liu.se/support/graphics/

Connecting

- If using Windows, need to enable OpenSSH Client
  - Windows 10: “Add an optional feature”
- If using Windows, you need a terminal:
  - Git Bash: https://git-scm.com/
  - WSL: https://docs.microsoft.com/en-us/windows/wsl/install-win10
  - https://cmder.net/

Connecting

- Connect
  - `ssh -X ${account}@sigma.nsc.liu.se`
    - `${account}` = NSC account name, e.g. x_user
    - Asked for password, password chosen when requesting account for Sigma
- Close connection
  - `exit`

Connecting

- Want to be lazy? Upload your public key!
  - `ssh-copy-id ${account}@sigma.nsc.liu.se`
  - Issue that command in your **local** terminal!

Connecting

- Some useful Linux commands
  - Connect: `ssh`
  - List directory: `ls`
  - Create directory: `mkdir`
  - Change directory: `cd`
  - Secure copy (run on local machine): `scp (-r)`
- Word editors
  - `emacs`
  - `vim`

Connecting

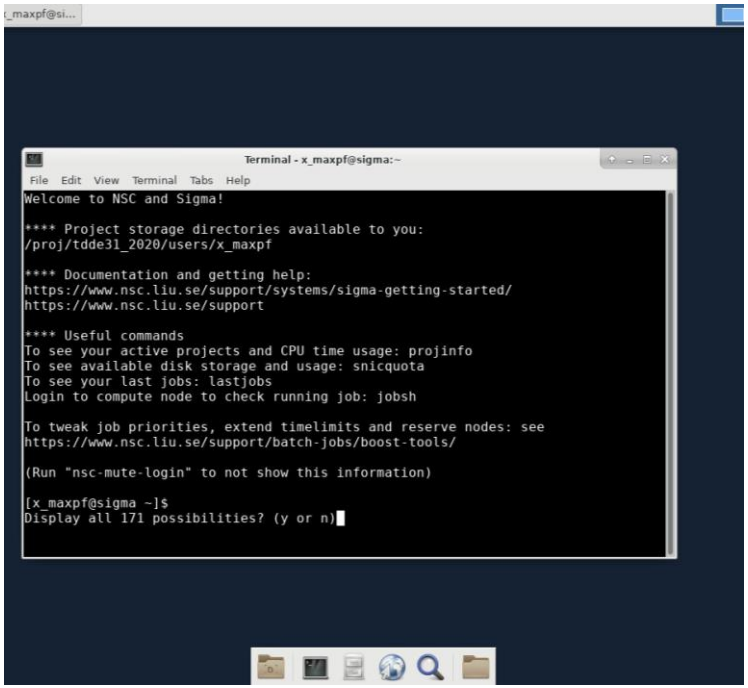
- SSH can do X-forwarding, meaning that you can display a remote GUI applications locally
- When you ssh into a machine, add the option **–X**
- You need a X Window system
  - Linux: xauth
  - macOS: https://www.xquartz.org/
  - Windows: PuTTY & Xming
- For details on setting up for your system: Google

Connecting - ThinLinc

Practical Introduction

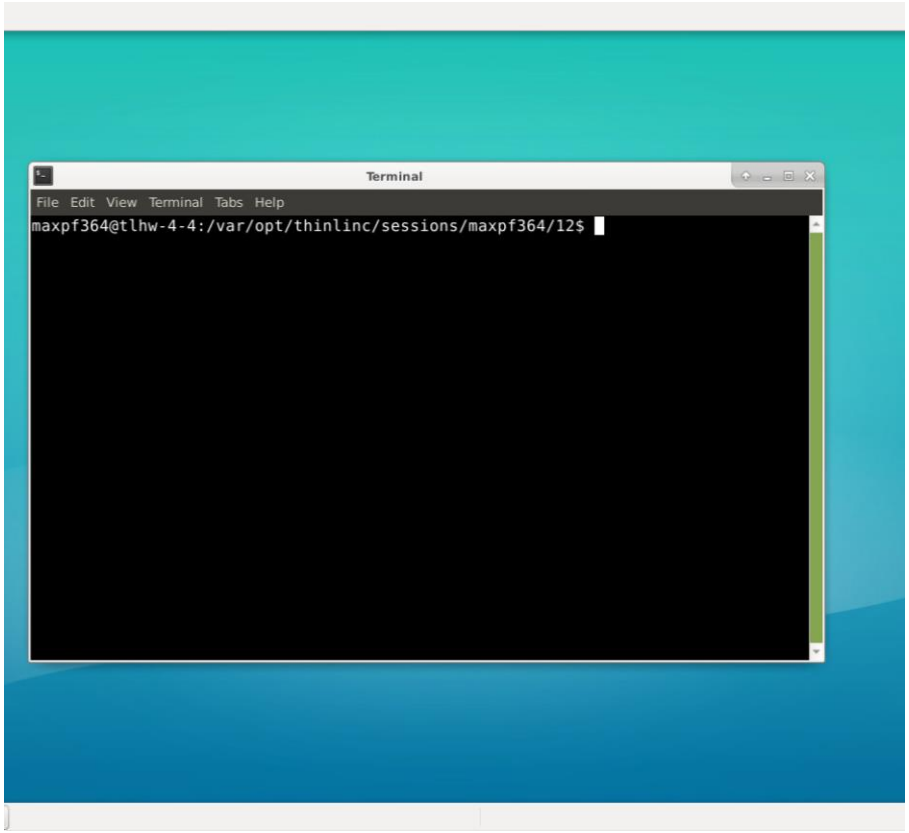
Connecting - ThinLinc

- Directly use ThinLinc to connect to the cluster
  - `sigma.nsc.liu.se`
  - `${account}`
  - Password
- Max one login per lab group



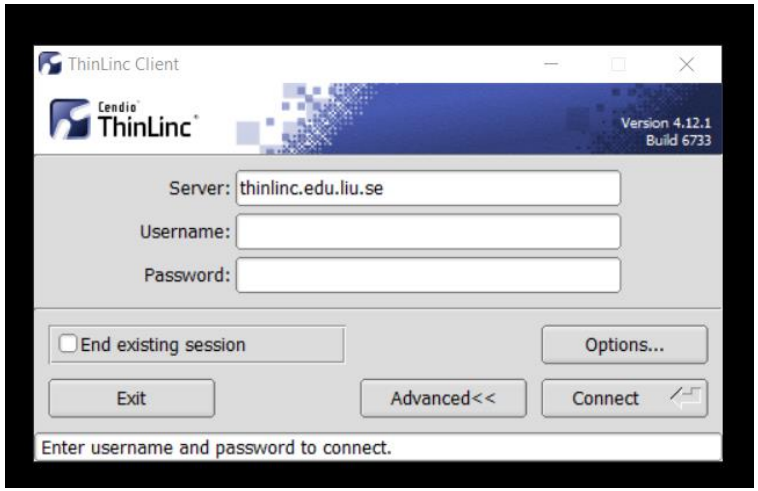
Connecting - ThinLinc

- Connect to LiU via ThinLinc (Linux Mint) and do everything from there
  - `thinlinc.edu.liu.se`
  - `{liuid}@student.liu.se`
  - password



Connecting - ThinLinc

- More info about ThinLinc on liu.se
- https://www.student.liu.se/studentstod/itsupport/linuxdatorsalar/fjarrinloggning?l=en&sc=true



Demo 3

Connecting

LINKÖPING UNIVERSITY

Developing

Practical Introduction

Developing

- “Disconnect” between coding and execution
- Code in separate IDE and execute on cluster
  - VS Code with python Plugin
  - PyCharm
  - JupyterLab
  - GitLab Web IDE
- Develop directly on the cluster
  - vim/emacs

Submit a Job

Practical Introduction

Submit a Job

1. Copy files
2. Submit Job
3. Monitor Job
4. Retrieve Results

Submit a Job | Copy files

- To copy entire folder on Sigma use `cp -R`
  - `cp -R {FROM} {TO}`
- To copy to or from local computer use `scp -r`
  - `scp -r {FROM} {TO}`
- For script files you can use git (through GitLab)

Submit a Job | Submit Job

- Add job to queue
  - `sbatch -A … --reservation … run.q`
- Reservation: Check compendium or
  - `listreservations`
- Look at queue
  - `squeue`
  - `squeue -A liu-compute-2021-4`
  - `squeue -u ${account}`

Submit a Job | Retrieve results

- Look at last entries in file
  - `tail -f ${file}`

Submit a Job | Copy files

- Copy results
  - `scp -r ${account}@sigma.nsc.liu.se:/home/${account}/.../output ./`
  - `scp ${account}@sigma.nsc.liu.se:/home/${account}/.../output/* ./`

Demo 4

Development and Submit jobs

LINKÖPING UNIVERSITY