UL HPC School 2017
PS1: Getting Started on the UL HPC platform

UL High Performance Computing (HPC) Team
C. Parisot

University of Luxembourg (UL), Luxembourg
http://hpc.uni.lu
Latest versions available on Github:

UL HPC tutorials: https://github.com/ULHPC/tutorials
UL HPC School: http://hpc.uni.lu/hpc-school/
PS1 tutorial sources: https://github.com/ULHPC/tutorials/tree/devel/basic/getting_started
Introduction

Summary

1. Introduction

2. SSH Secure Shell

3. Hands-On: Getting Started on ULHPC
Main Objectives of this Session

- Understand SSH
- Connect to the UL HPC Platform
  - SSH configuration
  - Generate your SSH key pair
  - overcome port filtering
- Discovering, visualizing and reserving UL HPC resources
  - Working environment
  - Web monitoring interfaces
  - OAR vs. SLURM Batch Scheduler
  - Job management
  - Software / Environement Modules
SSH Secure Shell

Summary

1. Introduction

2. SSH Secure Shell

3. Hands-On: Getting Started on ULHPC
SSH Secure Shell

**SSH: Secure Shell**

- Ensure **secure** connection to remote (UL) server
  - establish **encrypted** tunnel using **asymmetric keys**
    - ✓ **Public** id_rsa.pub vs. **Private** id_rsa *(without .pub)*
    - ✓ typically on a non-standard port *(Ex: 8022)* *(limits kiddie script)*
    - ✓ Basic rule: 1 machine = 1 key pair
  - the private key is **SECRET**: **never** send it to anybody
    - ✓ Can be protected with a passphrase
SSH Secure Shell

**SSH: Secure Shell**

- Ensure **secure** connection to remote (UL) server
  - establish **encrypted** tunnel using **asymmetric keys**
    - **Public** id_rsa.pub **vs. Private** id_rsa (**without** .pub)
    - typically on a non-standard port (**Ex**: 8022)
    - Basic rule: 1 machine = 1 key pair
  - the private key is **SECRET**: **never** send it to anybody
    - Can be protected with a passphrase

- SSH is used as a secure backbone channel for **many** tools
  - Remote shell **i.e** remote command line
  - File transfer: rsync, scp, sftp
  - versionning synchronization (**svn**, **git**), **github**, **gitlab** etc.
Ensure **secure** connection to remote (UL) server

- establish **encrypted** tunnel using **asymmetric keys**
  - **Public** `id_rsa.pub` vs. **Private** `id_rsa` (without `.pub`)
  - typically on a non-standard port (**Ex**: 8022) *limits kiddie script*
  - Basic rule: 1 machine = 1 key pair
- the private key is **SECRET**: **never** send it to anybody
  - Can be protected with a passphrase

**SSH is used as a secure backbone channel for many tools**

- Remote shell *i.e* remote command line
- File transfer: `rsync`, `scp`, `sftp`
- versionning synchronization (`svn`, `git`), `github`, `gitlab` etc.

**Authentication:**

- **password**
- *(better)* **public key authentication** *(disable if possible)*
SSH: Public Key Authentication

- **Client**
  - **Local Machine**
    - local homedir
    - ~/.ssh/
      - id_rsa
      - id_rsa.pub
      - known_hosts
    - owns local private key
    - logs known servers
SSH Secure Shell

SSH: Public Key Authentication

Client
Local Machine

remote homedir

~/.ssh/

owns local private key

logs known servers

id_rsa

id_rsa.pub

known_hosts

Server
Remote Machine

remote homedir

~/.ssh/

knows granted (public) key

authorized_keys

C. Parisot (University of Luxembourg)
SSH Secure Shell

SSH: Public Key Authentication

Client
Local Machine

- ~/.ssh/
- id_rsa
- id_rsa.pub
- known_hosts

owns local private key
logs known servers

Server
Remote Machine

- /etc/ssh/
- authorized_keys
- sshd_config
- ssh_host_rsa_key
- ssh_host_rsa_key.pub

knows granted (public) key

C. Parisot (University of Luxembourg)
**SSH: Public Key Authentication**

**Client (Local Machine)**
- Local homedir: `~/.ssh/`
- `id_rsa`
- `id_rsa.pub`
  - Owns local private key

**Server (Remote Machine)**
- Remote homedir: `~/.ssh/`
- `authorized_keys`
  - Knows granted (public) key
SSH Secure Shell

SSH: Public Key Authentication

Client
Local Machine

- local homedir
  ~/.ssh/
- id_rsa
- id_rsa.pub

owns local private key

Server
Remote Machine

- remote homedir
  ~/.ssh/
- authorized_keys

knows granted (public) key

1. Initiate connection
2. create random challenge, "encrypt" using public key
3. solve challenge using private key return response
4. allow connection iff response == challenge

- Restrict to public key authentication: /etc/ssh/sshd_config:

```
PermitRootLogin no
# Disable Passwords
PasswordAuthentication no
ChallengeResponseAuthentication no
```

```
# Enable Public key auth.
RSAAuthentication yes
PubkeyAuthentication yes
```
SSH Secure Shell

SSH Setup on Linux / Mac OS

- OpenSSH natively supported; configuration directory: ~/.ssh/
  - package openssh-client (Debian-like) or ssh (Redhat-like)
- SSH Key Pairs (public vs private) generation: ssh-keygen
  - specify a strong passphrase
    - ✓ protect your private key from being stolen i.e. impersonation
    - ✓ drawback: passphrase must be typed to use your key
SSH Secure Shell

SSH Setup on Linux / Mac OS

- OpenSSH natively supported; configuration directory: `~/.ssh/`
  - package `openssh-client` (Debian-like) or `ssh` (Redhat-like)
- SSH Key Pairs (public vs private) generation: `ssh-keygen`
  - specify a **strong** passphrase
    - ✓ protect your **private** key from being stolen *i.e.* impersonation
    - ✓ **drawback:** passphrase must be typed to use your key
- `ssh-agent`
SSH Secure Shell

SSH Setup on Linux / Mac OS

- OpenSSH natively supported; configuration directory: `~/.ssh/`
  - package `openssh-client` (Debian-like) or `sshd` (Redhat-like)
- SSH Key Pairs (public vs private) generation: `ssh-keygen`
  - specify a **strong** passphrase
    - ✓ protect your *private* key from being stolen *i.e.* impersonation
    - ✓ **drawback:** passphrase must be typed to use your key *ssh-agent*

| DSA and RSA 1024 bit are deprecated now! |
SSH Secure Shell

SSH Setup on Linux / Mac OS

- OpenSSH natively supported; configuration directory: ~/.ssh/
  
  package openssh-client (Debian-like) or ssh (Redhat-like)

- SSH Key Pairs (public vs private) generation: `ssh-keygen`
  
  specify a strong passphrase

  ✓ protect your private key from being stolen i.e. impersonation

  ✓ drawback: passphrase must be typed to use your key

  `ssh-agent`

DSA and RSA 1024 bit are deprecated now!

\[
\text{ssh-keygen} \ -t \ \text{rsa} \ -b \ 4096 \ -o \ -a \ 100 \quad \# \ 4096 \ bits \ RSA
\]

\[
(\text{better}) \quad \text{ssh-keygen} \ -t \ \text{ed25519} \ -o \ -a \ 100 \quad \# \ \text{new sexy Ed25519}
\]

Private (identity) key

~/.ssh/id_{rsa,ed25519}

Public Key

~/.ssh/id_{rsa,ed25519}.pub
SSH Secure Shell

SSH Setup on Windows

- **Putty Suite**, includes:
  - PutTY, the free SSH client
  - Pageant, an SSH authentication agent for PutTY tools
  - PLink, the PutTY CLI
  - PuTTYgen, an RSA and DSA key generation utility

http://www.chiark.greenend.org.uk/~sgtatham/putty/
SSH Secure Shell

SSH Setup on Windows

- **Putty Suite**, includes:  
  - PuTTY, the free SSH client
  - Pageant, an SSH authentication agent for PuTTY tools
  - PLink, the PuTTY CLI
  - PuTTYgen, an RSA and DSA key generation utility

PuTTY ≠ OpenSSH
SSH Secure Shell

SSH Setup on Windows

  - PuTTY, the free SSH client
  - Pageant, an SSH authentication agent for PuTTY tools
  - PLink, the PuTTY CLI
  - PuTTYgen, an RSA and DSA key generation utility

PuTTY ≠ OpenSSH

- Putty keys are **NOT** supported by OpenSSH (yet can be exported)
- Binding Pageant with OpenSSH agent is **NOT** natively supported
  - Third-party tools like ssh-pageant are made for that
  - Combine nicely with Git bash [https://git-for-windows.github.io/](https://git-for-windows.github.io/)

- with PLink, hostnames eventually refer to **PuTTY Sessions**
  - **NEVER** to SSH entries in ~/.ssh/config
  - This usage might be hidden... Ex: $GIT_SSH etc.
SSH Secure Shell

SSH Basic Usage

*.<domain>-intern.com

gitlab

bastion1.<domain>.com

C. Parisot  (University of Luxembourg)
SSH Secure Shell

SSH Basic Usage
SSH Secure Shell

SSH Basic Usage

- `<domain>-intern.com`
- `OpenSSH ~/.ssh/config (Mac / Linux)`
- `Host uni_*`
  - `User <LOGIN>`
  - `Port 8822`
- `Host uni_bastion1`
  - `Hostname bastion1.<domain>.com`

```
*.<domain>-intern.com
```

```
SSH
```

```
OpenSSH ~/.ssh/config (Mac / Linux)
Host uni_*
  User <LOGIN>
  Port 8822
Host uni_bastion1
  Hostname bastion1.<domain>.com
```
SSH Secure Shell

SSH Basic Usage

*.<domain>-intern.com

OpenSSH   ~/.ssh/config (Mac / Linux)
Host uni_*
User <LOGIN>
Port 22
Hostname bastion1.<domain>.com

PuTTY / PLink / Pageant (Windows)
Session "uni bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>

SSH

bastion1.<domain>.com

gitlab
SSH Advanced Usage: SOCKS Proxy
SSH Secure Shell

SSH Advanced Usage: SOCKS Proxy

---

SSH

*.<domain>-intern.com

OpenSSH   ~/.ssh/config    (Mac / Linux)

Host uni_*
User <LOGIN>
Port 8022
Hostname bastion1.<domain>.com

PuTTY / PLink / Pageant (Windows)

Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8822
- Connection/Data: username: <LOGIN>

bastion1.<domain>.com

SSH

https://gitlab.<domain>-intern.com

---

C. Parisot  (University of Luxembourg)
SSH Secure Shell

SSH Advanced Usage: SOCKS Proxy

**SSH**

*.<domain>-intern.com

OpenSSH  ~/.ssh/config    (Mac / Linux)

Host uni_*
User <LOGIN>
Port 8022
Hostname bastion1.<domain>.com

PuTTY / PLink / Pageant (Windows)
Session “uni bastion1”
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>

bastion1.<domain>.com

HTTPS/gitlab.<domain>-intern.com

DOMIAN <domain>-intern.com
NOT FOUND

C. Parisot (University of Luxembourg)
SSH Secure Shell

SSH Advanced Usage: SOCKS Proxy

---

**OpenSSH**

`.ssh/config` (Mac / Linux)

- `Host uni_*`
  - `Username: <LOGIN>`
  - `Port: 8822`
  - `Connection/Data: username: <LOGIN>`
  - `Connection/SSH/Tunnels: Port 1080, Dynamic`

**PuTTY / PLink / Pageant (Windows)**

- `Session "uni_bastion1"`
  - `Hostname: bastion1.<domain>.com`
  - `Port: 8822`
  - `Connection/Data: username: <LOGIN>`
  - `Connection/SSH/Tunnels: Port 1080, Dynamic`

**SSH -D 1080 (SOCKS 5 Proxy)**

- SOCKS LISTEN: localhost:1080

---

C. Parisot  (University of Luxembourg)
SSH Secure Shell

SSH Advanced Usage: SOCKS Proxy

SSH -D 1080 (SOCKS 5 Proxy)

PuTTY / PLink / Pageant (Windows)
- Session "uni bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8822
- Connection/SSH/Tunnels: Port 1080, Dynamic

OpenSSH ~/.ssh/config (Mac / Linux)
- Host uni *
  - User: <LOGIN>
  - Port: 8822
- Host uni_bastion1
  - Hostname bastion1.<domain>.com
SSH Secure Shell

SSH Advanced Usage: SOCKS Proxy

- For Mac/Linux users, use:
  ```bash
  Host uni_*)
  User <LOGIN>
  Port 8022
  Connection/SSH/Tunnels: Port 1080, Dynamic
  ```

- For Windows users, use PuTTY, PLink, or Pageant:
  - Session "uni_bastion1"
  - Hostname: bastion1.<domain>.com
  - Port: 8022
  - Connection/Data: username: <LOGIN>
  - Connection/SSH/Tunnels: Port 1080, Dynamic

- For SOCKS Proxy setup:
  ```bash
  SOCKS LISTEN: localhost:1080
  ```

- Access GitLab via SOCKS Proxy:
  ```bash
  https://gitlab.<domain>-intern.com
  ```
SSH Secure Shell

SSH Advanced Usage: SOCKS Proxy

#### SSH Secure Shell

**SSH Secure Shell**

*<domain>*-intern.com

**OpenSSH**   ~/.ssh/config    (Mac / Linux)

- Host: uni_*<domain>*-intern.com
- User: <LOGIN>
- Port: 8022
- Hostname: bastion1.<domain>.com
- Connection/SSH/Hosts: Port 1080, Dynamic

**PuTTY / PLink / Pageant (Windows)**

- Session “uni_bastion1”
  - Hostname: bastion1.<domain>.com
  - Port: 8022
  - Connection/Data: username: <LOGIN>
  - Connection/SSH/Tunnels: Port 1080, Dynamic

- SOCKS LISTEN: localhost:1080

**C. Parisot**  (University of Luxembourg)
SSH Secure Shell

SSH Advanced Usage: SOCKS Proxy

*.<domain>-intern.com

PuTTY / PLink / Pageant (Windows)
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8822
- Connection/Data: username: <LOGIN>
- Connection/SSH/Tunnels: Port 1080, Dynamic

OpenSSH ~/.ssh/config (Mac / Linux)
Host uni.*
  User <LOGIN>
  Port 8822
Host uni_bastion1
  Hostname bastion1.<domain>.com

SSH -D 1080 (SOCKS 5 Proxy)

SOCKS LISTEN: localhost:1080

https://gitlab.<domain>-intern.com

C. Parisot (University of Luxembourg)
SSH Advanced Usage: ProxyCommand

```
.*.<domain>-intern.com
```

```
gitlab
```

```
bastion1.<domain>.com
```

---

C. Parisot (University of Luxembourg)
SSH Secure Shell

SSH Advanced Usage: ProxyCommand

```plaintext
*.<domain>-intern.com

OpenSSH  ~/.ssh/config  (Mac / Linux)
Host uni_*
  User <LOGIN>
  Port 8822
  Hostname bastion1.<domain>.com

PuTTY / PLink / Pageant (Windows)
Session "uni_bastion1"
  Hostname: bastion1.<domain>.com
  Port: 8822
  Connection/Data: username: <LOGIN>

SSH
```

C. Parisot  (University of Luxembourg)
SSH Secure Shell

SSH Advanced Usage: ProxyCommand

PuTTY / PLink / Pageant (Windows)

- Session "uni_bastion1"
  - Hostname: bastion1.<domain>.com
  - Port: 8022
  - Connection/Proxy:
    - type: local
    - Proxy hostname: bastion1.<domain>.com
    - Port: 8022
    - Username: <LOGIN>
    - Local proxy command:
      plink -load "uni_bastion1" -nc %host:%port

OpenSSH ~/.ssh/config (Mac / Linux)

- Host uni_*
  - User <LOGIN>
  - Port 8822
- Host uni_bastion1
  - Hostname bastion1.<domain>.com
- Host uni_gitlab
  - Hostname gitlab.<domain>-intern.com
  - Port 8022
  - Connection/Proxy:
    ProxyCommand ssh -q uni_bastion1 "nc %h %p"
SSH Advanced Usage: ProxyCommand

**SSH Secure Shell**

- **SSH**
  - Host: uni_1
  - User: <LOGIN>
  - Port: 8022
  - Connection/Data: username: <LOGIN>

- **Host uni_bastion1**
  - Hostname: bastion1.<domain>.com
  - Port: 8022
  - Username: <LOGIN>
  - ProxyCommand: ssh -q uni_bastion1 "nc %h %p"

- **Host uni_gitlab**
  - Hostname: gitlab.<domain>-intern.com
  - Port: 8022
  - Connection/Data: username: <LOGIN>
  - Connection/Proxy:
    - Type: local
    - Proxy hostname: bastion1.<domain>.com
    - Port: 8022
    - Username: <LOGIN>
    - Local proxy command:
      plink -load "uni_bastion1" -nc %host:%port

**PUTTY / PLink / Pageant (Windows)**

- Session "uni_bastion1"
  - Hostname: bastion1.<domain>.com
  - Port: 8022
  - Connection/Data: username: <LOGIN>

- Session "uni_gitlab"
  - Hostname: gitlab.<domain>-intern.com
  - Port: 8022
  - Connection/Data: username: <LOGIN>
  - Connection/Proxy:
    - Type: local
    - Proxy hostname: bastion1.<domain>.com
    - Port: 8022
    - Username: <LOGIN>
    - Local proxy command:
      plink -load "uni_bastion1" -nc %host:%port

**OpenSSH ~/.ssh/config (Mac / Linux)**

- Host uni *
  - User: <LOGIN>
  - Port 8022

- Host uni_bastion1
  - Hostname bastion1.<domain>.com

- Host uni_gitlab
  - Hostname gitlab.<domain>-intern.com
  - ProxyCommand ssh -q uni_bastion1 "nc %h %p"
SSH Advanced Usage: ProxyCommand

**OpenSSH** ~/.ssh/config (Mac / Linux)
- Host uni_*
  - Hostname: bastion1.<domain>.com
  - Port: 8822
  - Connection/Data: username: <LOGIN>
- Host uni_bastion1
  - Hostname: bastion1.<domain>.com
  - Port: 8022
  - Connection/Data: username: <LOGIN>
  - Connection/Proxy:
    - Protocol: ssh
    - Type: local
    - Proxy hostname: bastion1.<domain>.com
    - Port: 8022
    - Username: <LOGIN>
    - Local proxy command:
      `plink -load "uni_bastion1" -nc %host:%port`

**PuTTY / PLink / Pageant (Windows)**
- Session “uni_bastion1”
  - Hostname: bastion1.<domain>.com
  - Port: 8822
  - Connection/Data: username: <LOGIN>
- Session “uni_gitlab”
  - Hostname: gitlab.<domain>-intern.com
  - Port: 8822
  - Connection/Data: username: <LOGIN>
  - Connection/Proxy:
    - Type: local
    - Proxy hostname: gitlab.<domain>-intern.com
    - Port: 8822
    - Username: <LOGIN>
    - Local proxy command:
      `plink -load "uni_bastion1" -nc %host:%port`

SSH + ProxyCommand + netcat

C. Parisot (University of Luxembourg)
SSH Secure Shell

SSH in Practice

~/.ssh/config

$> ssh [-X] [-p <port>] <login>@<hostname>

# Example: ssh -p 8022 svarrette@access-chaos.uni.lu

~/.ssh/config:

- Simpler commands
- Bash completion

$> ssh cha<TAB>
SSH Secure Shell

SSH in Practice

~/.ssh/config

```bash
$> ssh [-X] [-p <port>] <login>@<hostname>

# Example: ssh -p 8022 svarrette@access-chaos.uni.lu

Host *.*ext_ul
  ProxyCommand ssh -q chaos-cluster \ 
    "nc -q 0 %h %p"
  # UL HPC Platform -- http://hpc.uni.lu
Host chaos-cluster
  Hostname access-chaos.uni.lu
Host gaia-cluster
  Hostname access-gaia.uni.lu
Host iris-cluster
  Hostname access-iris.uni.lu
Host *-*cluster
  User login #ADAPT accordingly
  Port 8022
  ForwardAgent no

Host <shortname>
  Port <port>
  User <login>
  Hostname <hostname>

~/.ssh/config:
  ⇫ Simpler commands
  ⇫ Bash completion

$> ssh cha<TAB>
```

C. Parisot (University of Luxembourg)
SSH Secure Shell

SSH in Practice

```bash
$> ssh [-X] [-p <port>] <login>@<hostname>

# Example: ssh -p 8022 svarrette@access-chaos.uni.lu
```

### ~/.ssh/config

- **Host *.ext_ul**
  - `ProxyCommand ssh -q chaos-cluster \ "nc -q 0 %h %p"
  - # UL HPC Platform -- http://hpc.uni.lu
- **Host chaos-cluster**
  - `Hostname access-chaos.uni.lu`
- **Host gaia-cluster**
  - `Hostname access-gaia.uni.lu`
- **Host iris-cluster**
  - `Hostname access-iris.uni.lu`
- **Host *-cluster**
  - `User login` #ADAPT accordingly
  - `Port 8022`
  - `ForwardAgent no`

- **Host <shortname>**
  - `Port <port>`
  - `User <login>`
  - `Hostname <hostname>`

~/.ssh/config:
- Simpler commands
- Bash completion

```bash
$> ssh cha<TAB>
```

```bash
$> ssh chaos-cluster
$> ssh work
$> ssh work.ext_ul
```
SSH in Practice: Main CLI commands
SSH Secure Shell

DSH – Distributed / Dancer’s Shell

http://www.netfort.gr.jp/~dancer/software/dsh.html.en

- SSH wrapper that allows to run commands over multiple machines.
  - Linux / Mac OS only

```
$> { apt-get | yum | brew } install dsh
# Installation
```

- **Configuration**: in ~/.dsh/
  - ~/.dsh/dsh.conf: main configuration file
  - ~/.dsh/machines.list: list of all nodes
  - ~/.dsh/group/: holds group definition

- `<name>` Group definition: ~/.dsh/group/<name>:
  - simply list SSH shortnames (one name by line)

- Bash completion file for DSH:
  - https://gist.github.com/920433.git
### ~/.dsh/dsh.conf

# Configuration file for dsh (Distributed / Dancer’s Shell).
# ‘man dsh.conf’ for details

```
verbose = 0
remoteshell = ssh
showmachinenames = 1

# Specify 1 to make the shell wait for each individual invocation.
# See -c and -w option for dsh(1)
waitshell = 0  # whether to wait for execution

# Number of parallel connection to create at the same time.
# forklimit=8

remoteshellopt = -q
```
SSH Secure Shell

DSH Basic Usage

$> dsh [-c | -w] { -a | -g <group> | -m <hostname> } <command>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-c</td>
<td>run the commands in parallel (default)</td>
</tr>
<tr>
<td>-w</td>
<td>run the commands in sequential</td>
</tr>
<tr>
<td>-a</td>
<td>run the command on all nodes listed in machines.list</td>
</tr>
<tr>
<td>-g &lt;group&gt;</td>
<td>restrict the commands to the hosts group &lt;group&gt;</td>
</tr>
<tr>
<td>-m &lt;hostname&gt;</td>
<td>run the command only on hostname</td>
</tr>
</tbody>
</table>

FAQ: sudo: sorry, you must have a tty to run sudo

→ requires to change the default configuration of sudo

→ Ex to **not** requiring a tty to launch a sudo command

Defaults: <login> !requiretty
Summary

1 Introduction

2 SSH Secure Shell

3 Hands-On: Getting Started on ULHPC
Hands-On: Getting Started on ULHPC

Hands-On 1: SSH Setup

http://ulhpc-tutorials.readthedocs.io/en/latest/basic/getting_started/

Your Turn!

- **Generating you SSH Key pair**
- **Connect** to UL HPC (Linux / Mac OS / Unix / Windows)
  - Connect from your laptop/workstation to UL HPC access
  - Connect from one cluster to the other

- **Transferring files**
**Hand-on 2: First steps on UL HPC**

- **UL HPC Environment**
  - **Operating System:**
    - ✓ Debian 7 on gaia, chaos
    - ✓ CentOS 7 on iris
  - **Job Management:**
  - **Environment modules:**
    - ✓ **Not** available on frontends, *Only* on compute nodes
  - (advanced) discovering **GNU screen**

---

**Directory Table**

<table>
<thead>
<tr>
<th>Directory</th>
<th>Max size</th>
<th>Max #files</th>
<th>Backup</th>
</tr>
</thead>
<tbody>
<tr>
<td>$HOME (gaia, chaos)</td>
<td>100 GB</td>
<td>1.000.000</td>
<td>YES</td>
</tr>
<tr>
<td>$HOME (iris)</td>
<td>500 GB</td>
<td>1.000.000</td>
<td>YES</td>
</tr>
<tr>
<td>$WORK (except iris)</td>
<td>3 TB</td>
<td>1.000.000</td>
<td>NO</td>
</tr>
<tr>
<td>$SCRATCH (except iris)</td>
<td>10 TB</td>
<td></td>
<td>NO</td>
</tr>
</tbody>
</table>
ULHPC Web monitoring interfaces

http://hpc.uni.lu/status/overview.html
Hands-On: Getting Started on ULHPC

ULHPC Web monitoring interfaces

http://hpc.uni.lu/{iris,gaia,chaos,g5k}/monika
Hands-On: Getting Started on ULHPC

ULHPC Web monitoring interfaces

http://hpc.uni.lu/{iris,gaia,chaos,g5k}/drawgantt
Hands-On: Getting Started on ULHPC

ULHPC Web monitoring interfaces

http://hpc.uni.lu/{iris,gaia,chaos,g5k}/ganglia
Thank you for your attention...

Questions?

The UL High Performance Computing (HPC) Team
University of Luxembourg, Belval Campus:
Maison du Nombre, 4th floor
2, avenue de l'Université
L-4365 Esch-sur-Alzette
mail: hpc@uni.lu

1 Introduction

2 SSH Secure Shell

3 Hands-On: Getting Started on ULHPC