UL HPC School 2017[bis]

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

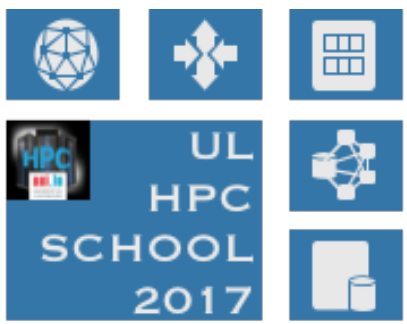
Latest versions available on Github:

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



Summary

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

Summary

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

- 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.

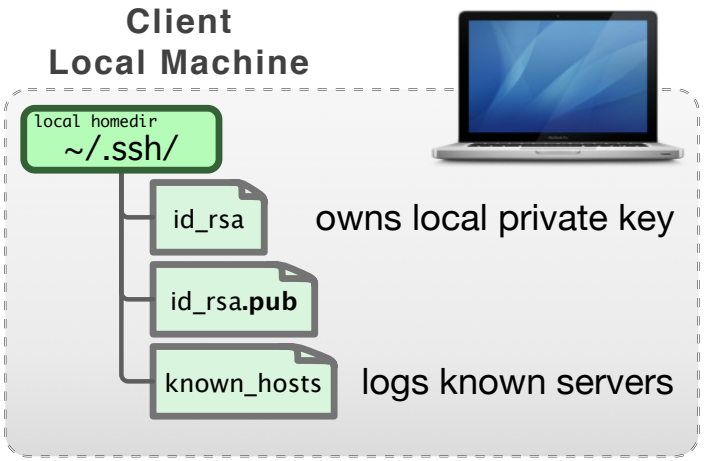
SSH: Secure Shell

- 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

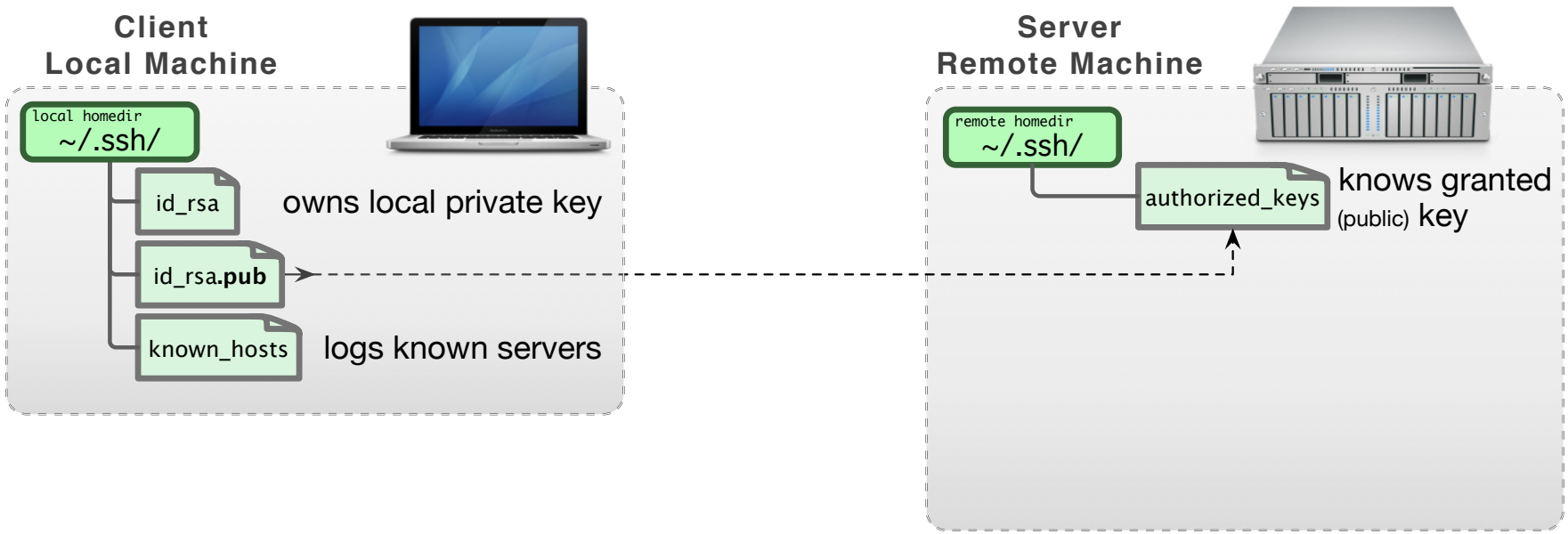
- SSH is used as a secure backbone channel for **many** tools
  - ↪ Remote shell **i.e** remote command line
  - ↪ File transfer: `rsync`, `scp`, `sftp`
  - ↪ versionning synchronization (`svn`, `git`), github, gitlab etc.

- Authentication:
  - ↪ ~~password~~ (disable if possible)
  - ↪ (*better*) **public key authentication**

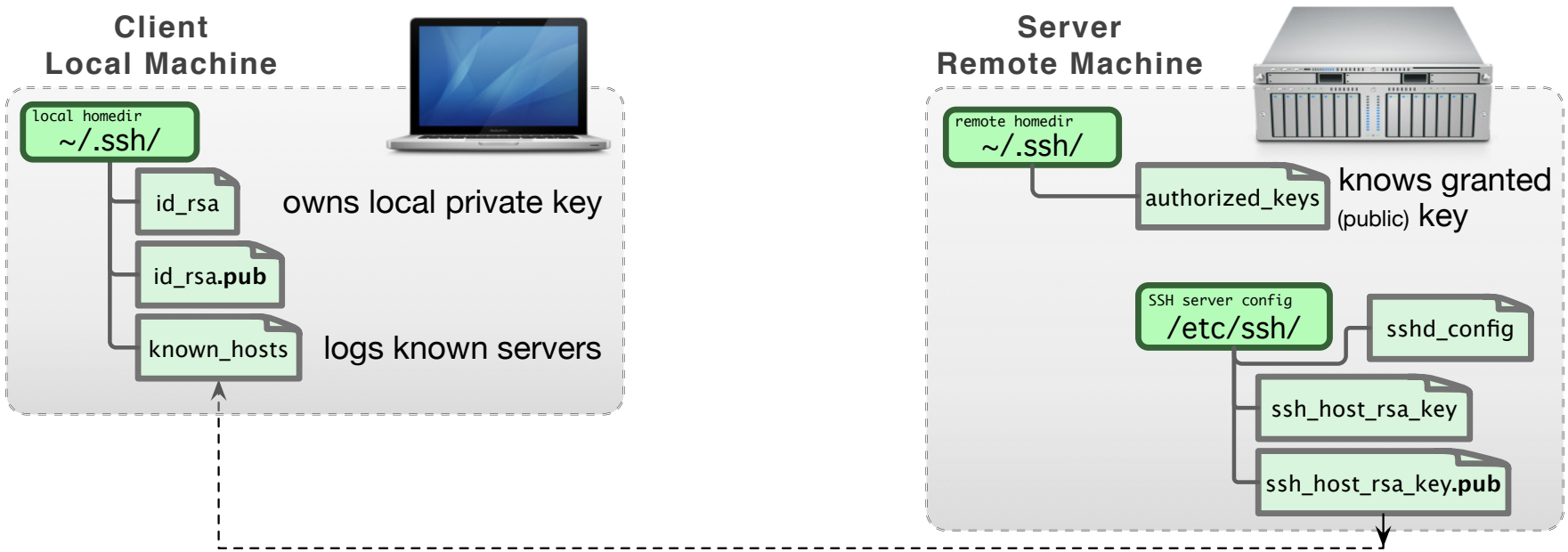
SSH: Public Key Authentication



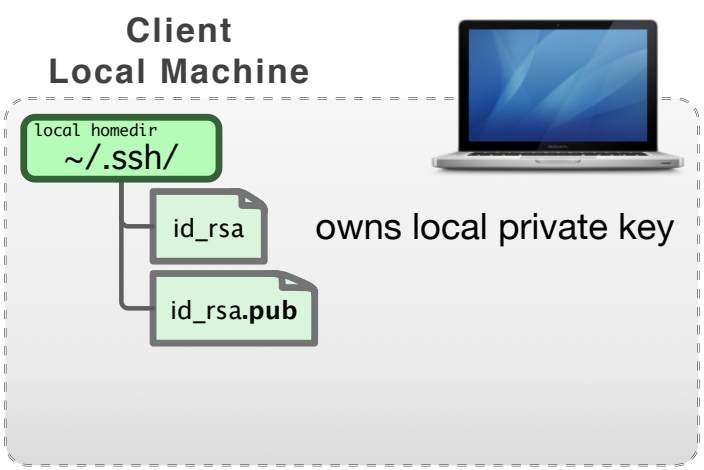
SSH: Public Key Authentication

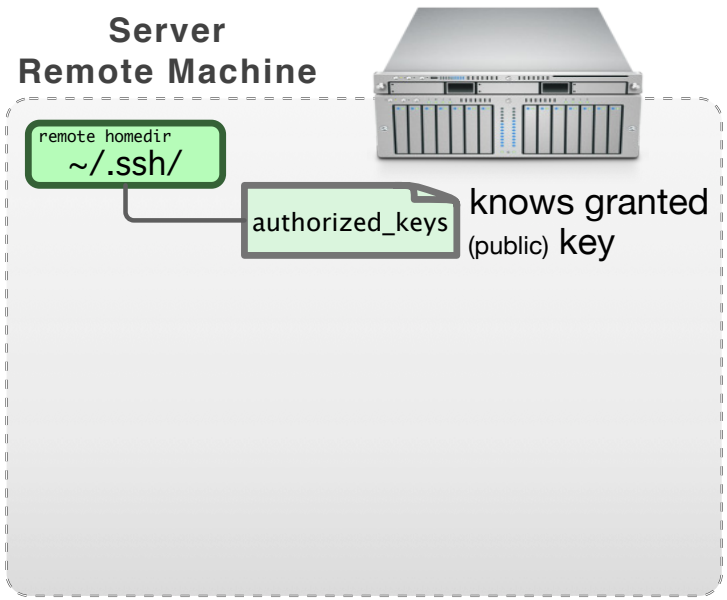


SSH: Public Key Authentication

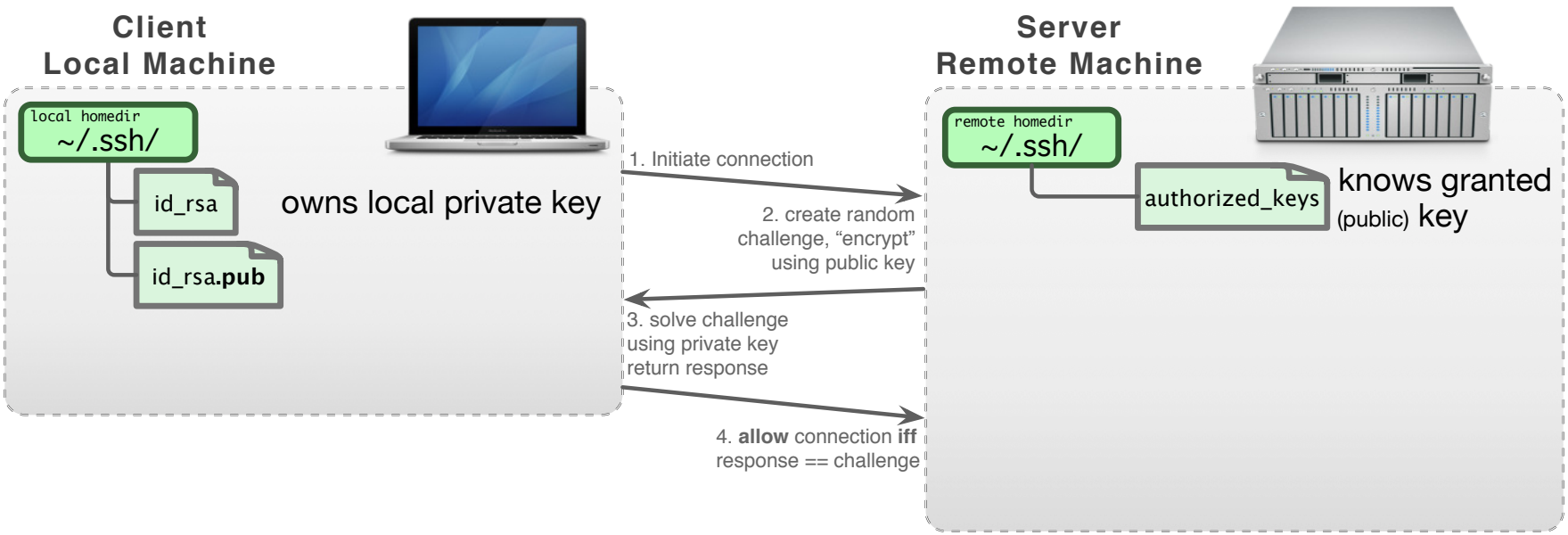


SSH: Public Key Authentication





SSH: Public Key Authentication



- **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 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 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 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!

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!

```
$> ssh-keygen -t rsa -b 4096 -o -a 100          # 4096 bits RSA
(better)  $> ssh-keygen -t ed25519 -o -a 100    # new sexy Ed25519
```

Private (identity) key

`~/.ssh/id_{rsa,ed25519}`

Public Key

`~/.ssh/id_{rsa,ed25519}.pub`

SSH Setup on Windows: the OLD way

- Putty Suite, includes: http://www.chiark.greenend.org.uk/~sgtatham/putty/ - PuTTY, the free SSH client - Pageant, an SSH authentication agent for PuTTY tools - PLink, th PuTTy CLI - PuTTYgen, an RSA and DSA key generation utility

SSH Setup on Windows: the OLD way

- Putty Suite, includes: http://www.chiark.greenend.org.uk/~sgtatham/putty/ - PuTTY, the free SSH client - Pageant, an SSH authentication agent for PuTTY tools - PLink, th PuTTy CLI - PuTTYgen, an RSA and DSA key generation utility

PuTTY $\neq$ OpenSSH

SSH Setup on Windows: the OLD way

- Putty Suite, includes: http://www.chiark.greenend.org.uk/~sgtatham/putty/ - PuTTY, the free SSH client - Pageant, an SSH authentication agent for PuTTY tools - PLink, th PuTTy CLI - PuTTYgen, an RSA and DSA key generation utility

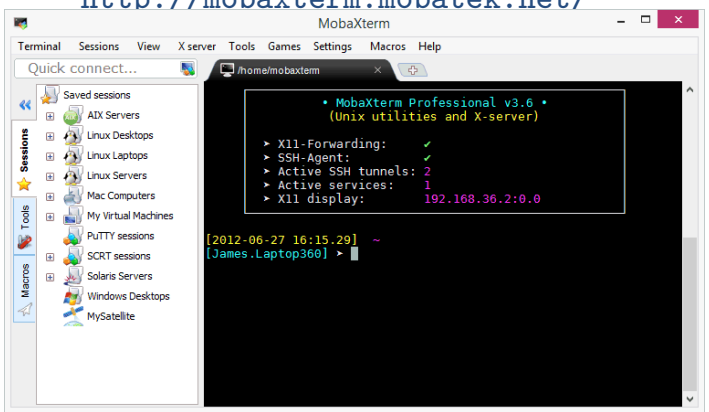
PuTTY ≠ OpenSSH

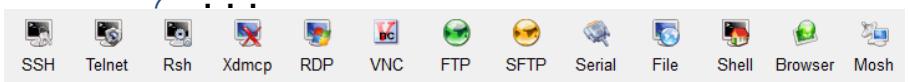
- Putty keys are **NOT** supported by OpenSSH (yet can be exported)
- Binding Pageant with OpenSSH agent is **NOT** natively supported
  - ↪ Third-party tools like ssh-pageant are made for that
  - ↪ Combine nicely with Git bash https://git-for-windows.github.io/
- with PLink, hostnames eventually refer to **PuTTY Sessions**
  - ↪ **NEVER** to SSH entries in `~/.ssh/config`
  - ↪ This usage might be hidden... Ex: `$GIT_SSH` etc.

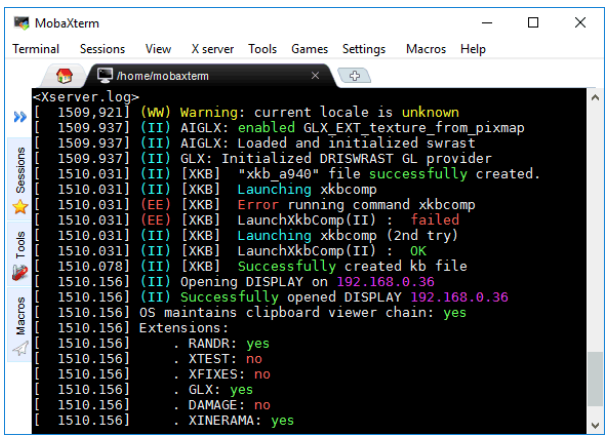
SSH Setup on Windows: the NEW way

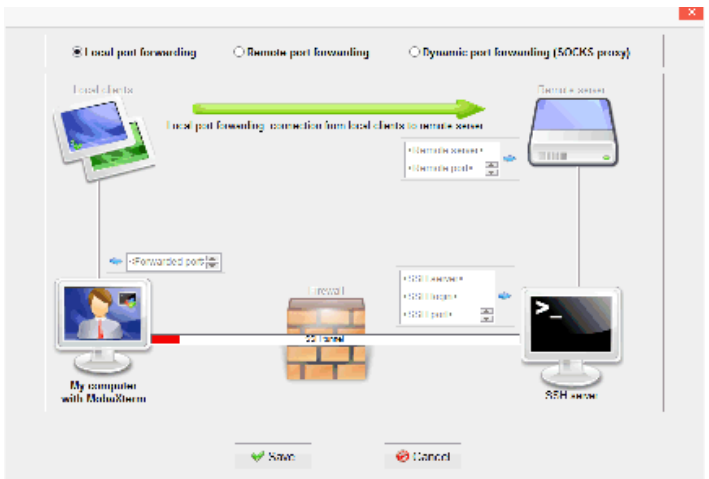
- Use MobaXterm!
  - ↪ [tabbed] Sessions management
  - ↪ X11 server w. enhanced X extensions
  - ↪ Graphical SFTP browser
  - ↪ SSH gateway / tunnels wizards
  - ↪ [remote] Text Editor
  - ↪ ...

http://mobaxterm.mobatek.net/

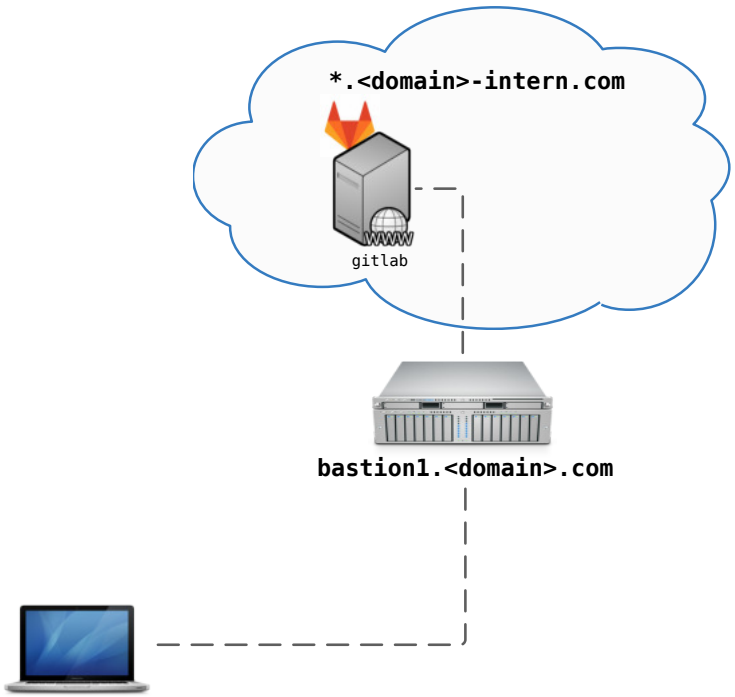




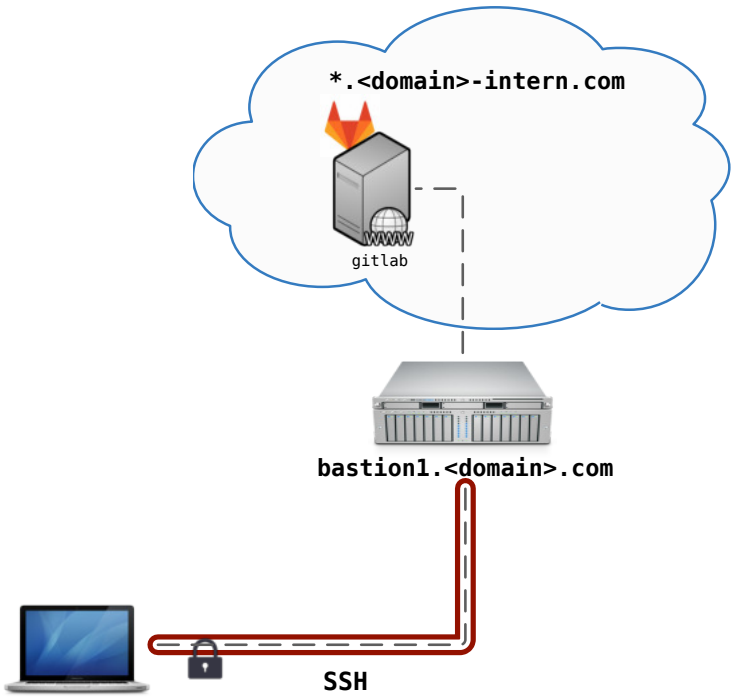




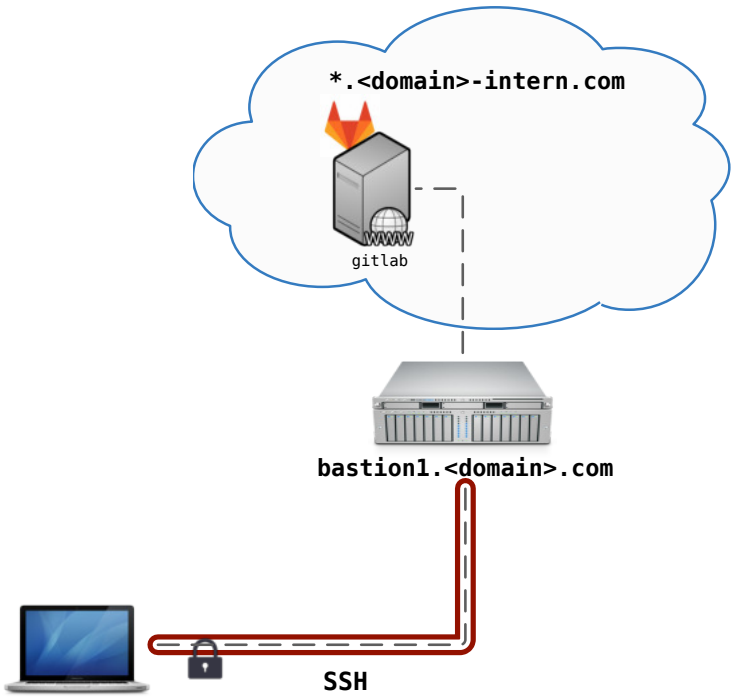
SSH Basic Usage



SSH Basic Usage



SSH Basic Usage

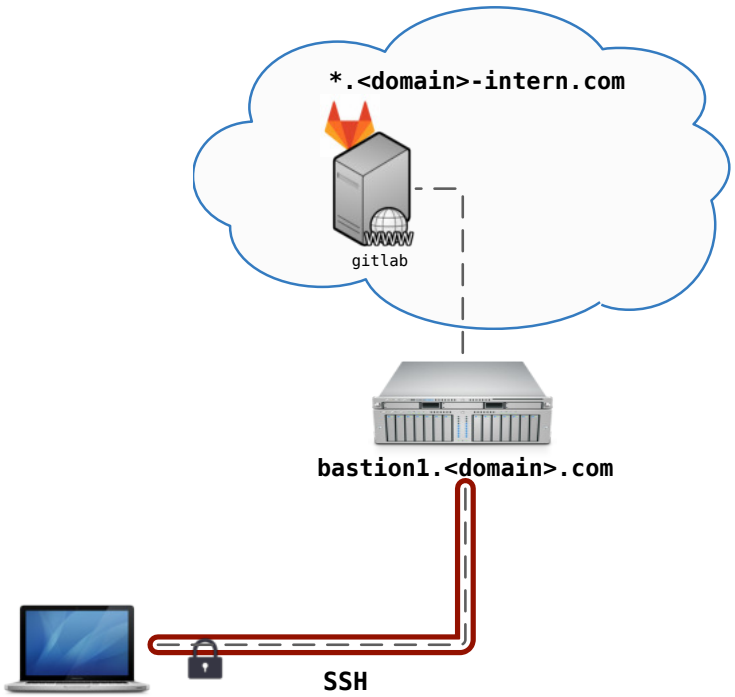


OpenSSH ~/.ssh/config (Mac / Linux)

```
Host uni_*
    User <LOGIN>
    Port 8022

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

SSH Basic Usage



PuTTY / PLink / Pageant (Windows)

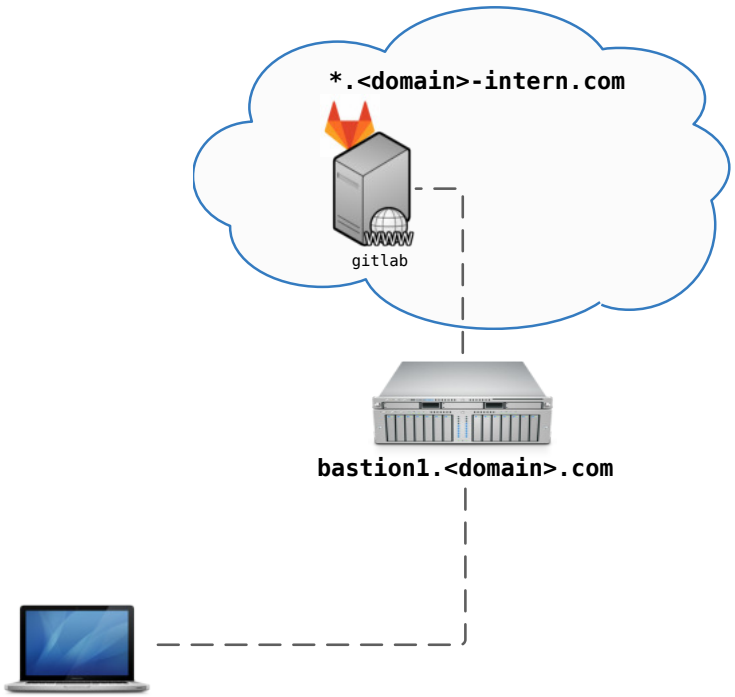
```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
```

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

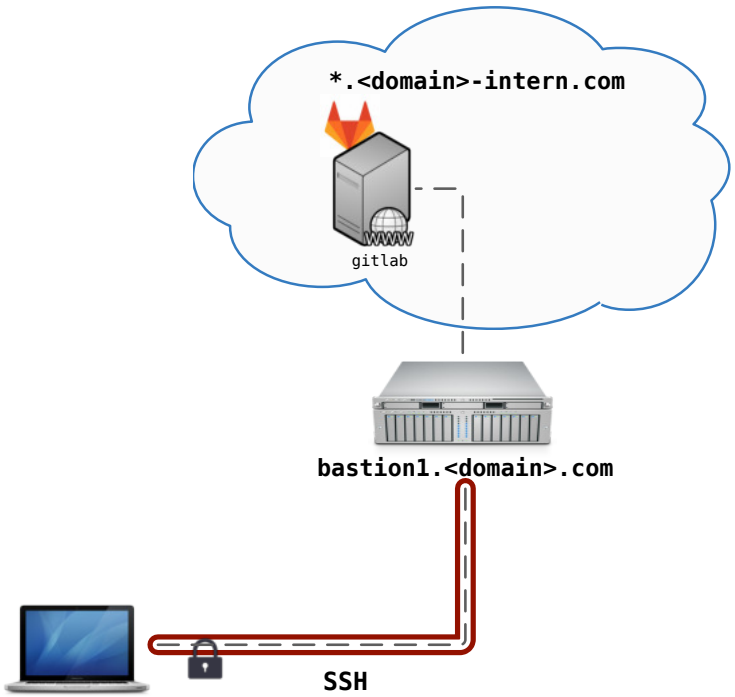
```
Host uni_*
    User <LOGIN>
    Port 8022

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

SSH Advanced Usage: SOCKS Proxy



SSH Advanced Usage: SOCKS Proxy



PuTTY / PLink / Pageant (Windows)

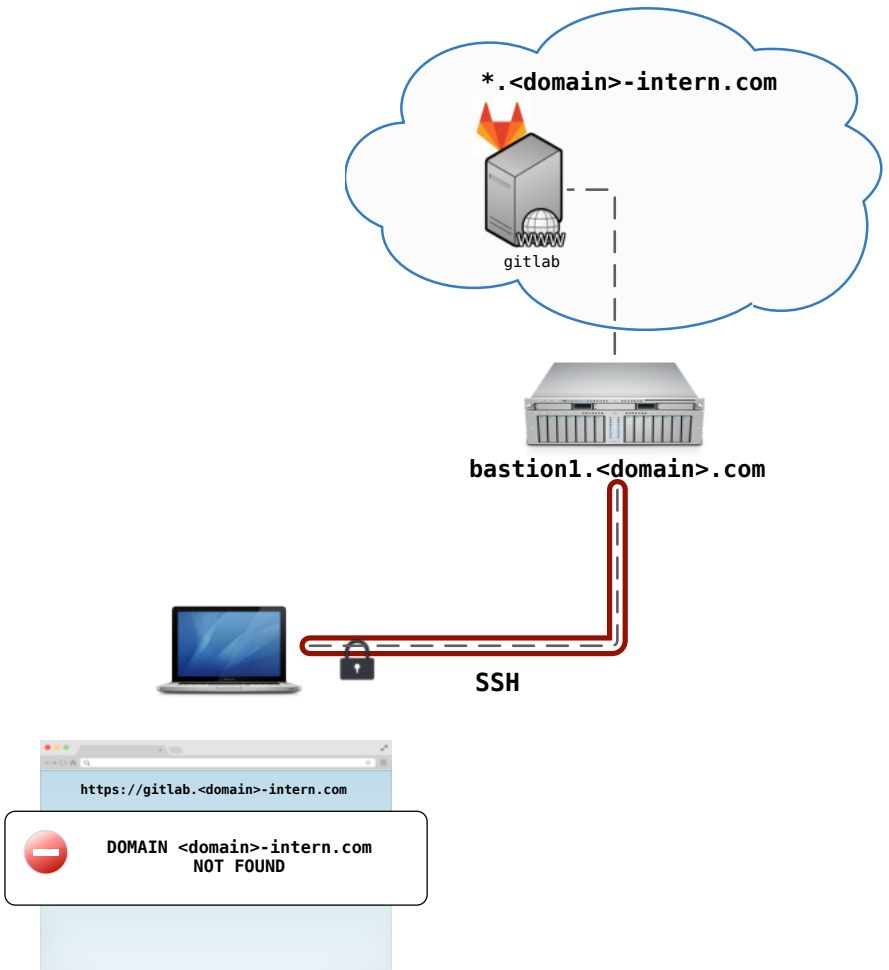
```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
```

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

```
Host uni_*
    User <LOGIN>
    Port 8022

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

SSH Advanced Usage: SOCKS Proxy



PuTTY / PLink / Pageant (Windows)

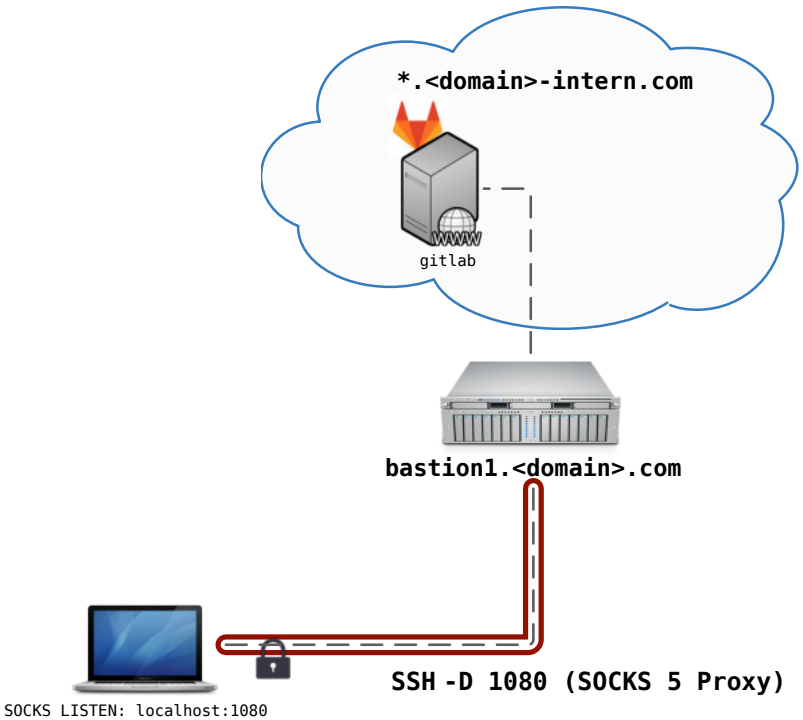
```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
```

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

```
Host uni_*
    User <LOGIN>
    Port 8022

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

SSH Advanced Usage: SOCKS Proxy



PuTTY / PLink / Pageant (Windows)

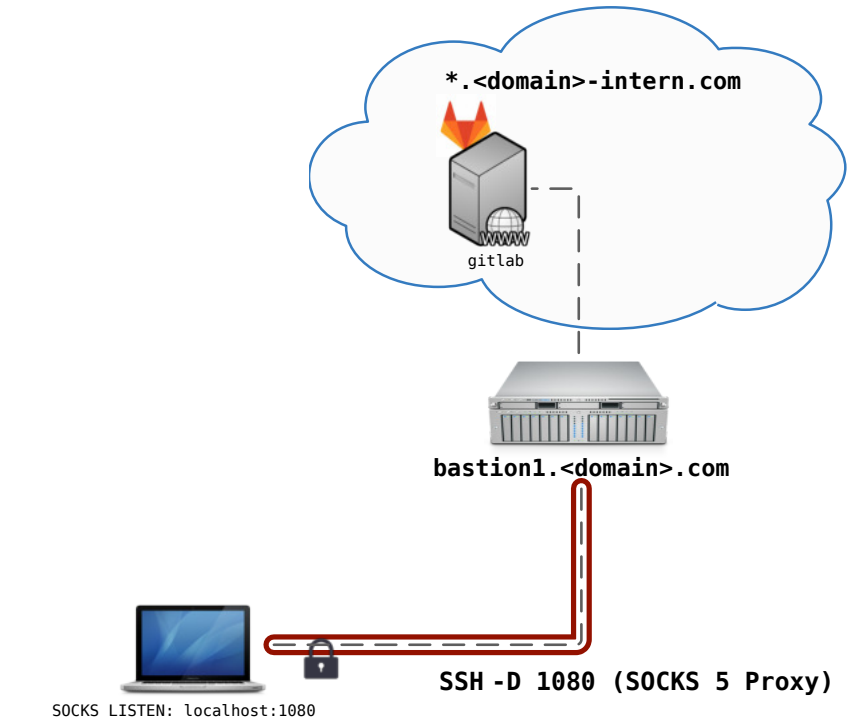
```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
- Connection/SSH/Tunnels: Port 1080, Dynamic
```

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

```
Host uni_*
    User <LOGIN>
    Port 8022

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

SSH Advanced Usage: SOCKS Proxy



PuTTY / PLink / Pageant (Windows)

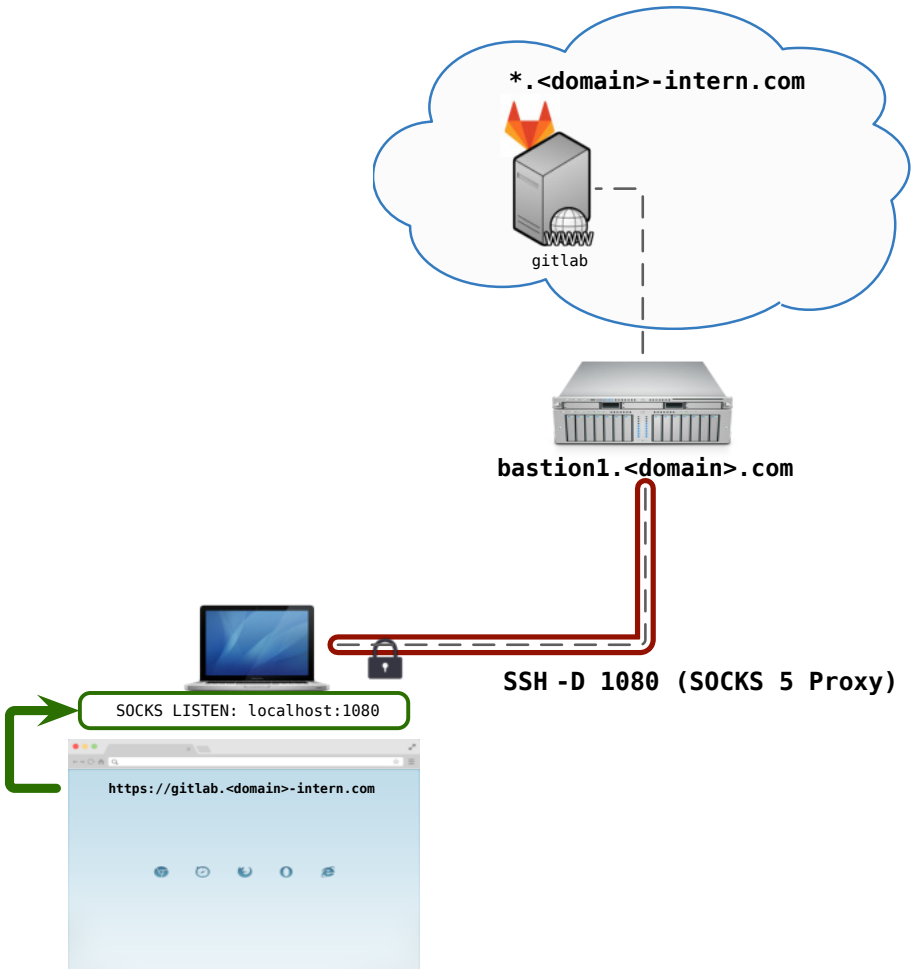
```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
- Connection/SSH/Tunnels: Port 1080, Dynamic
```

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

```
Host uni_*
    User <LOGIN>
    Port 8022

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

SSH Advanced Usage: SOCKS Proxy



PuTTY / PLink / Pageant (Windows)

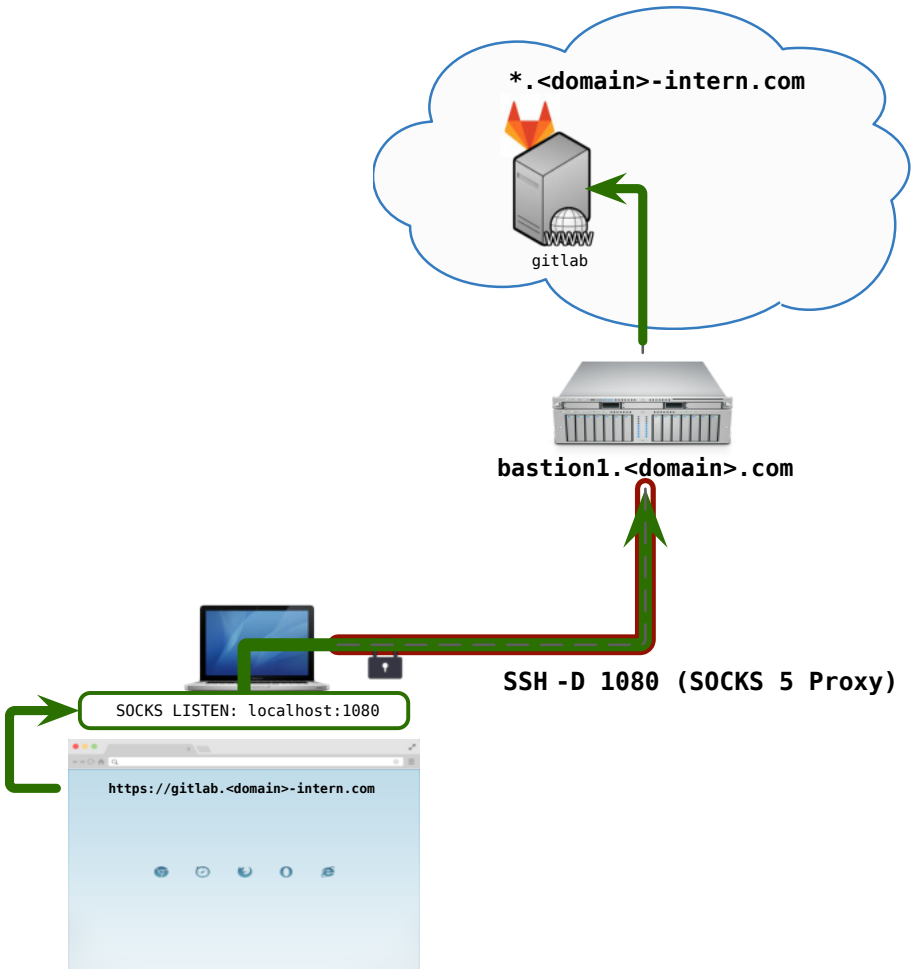
```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
- Connection/SSH/Tunnels: Port 1080, Dynamic
```

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

```
Host uni_*
    User <LOGIN>
    Port 8022

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

SSH Advanced Usage: SOCKS Proxy



PuTTY / PLink / Pageant (Windows)

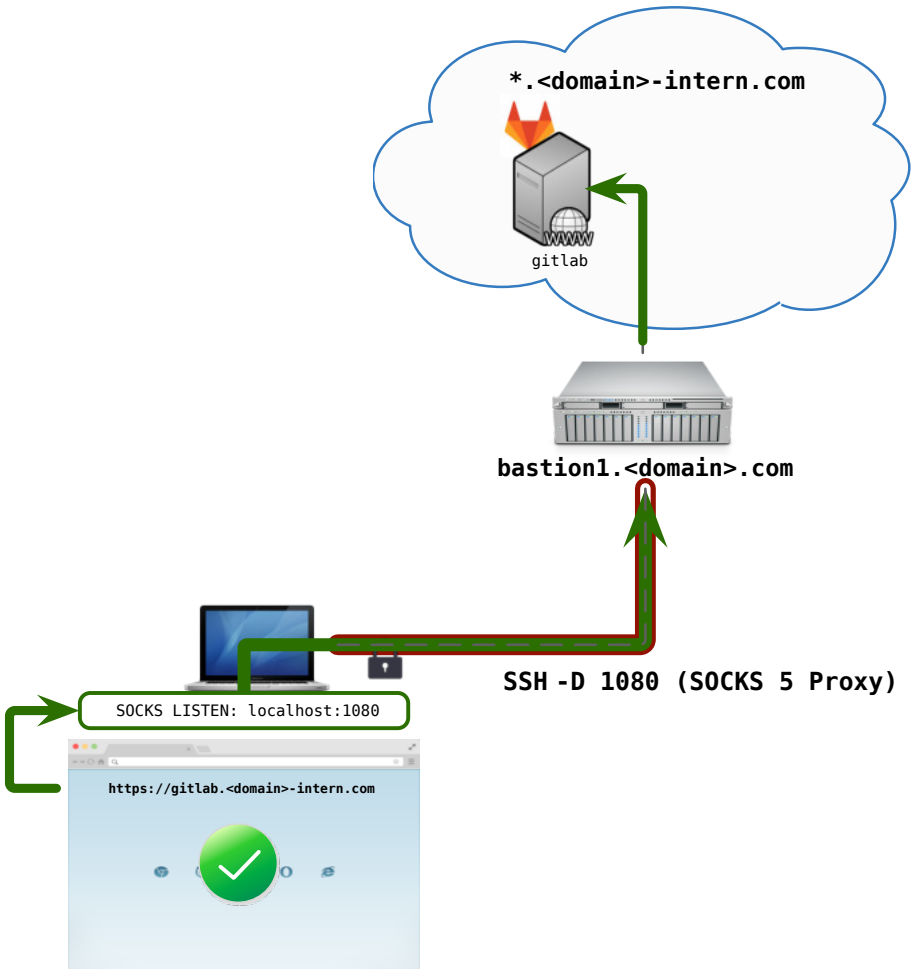
```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
- Connection/SSH/Tunnels: Port 1080, Dynamic
```

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

```
Host uni_*
    User <LOGIN>
    Port 8022

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

SSH Advanced Usage: SOCKS Proxy



PuTTY / PLink / Pageant (Windows)

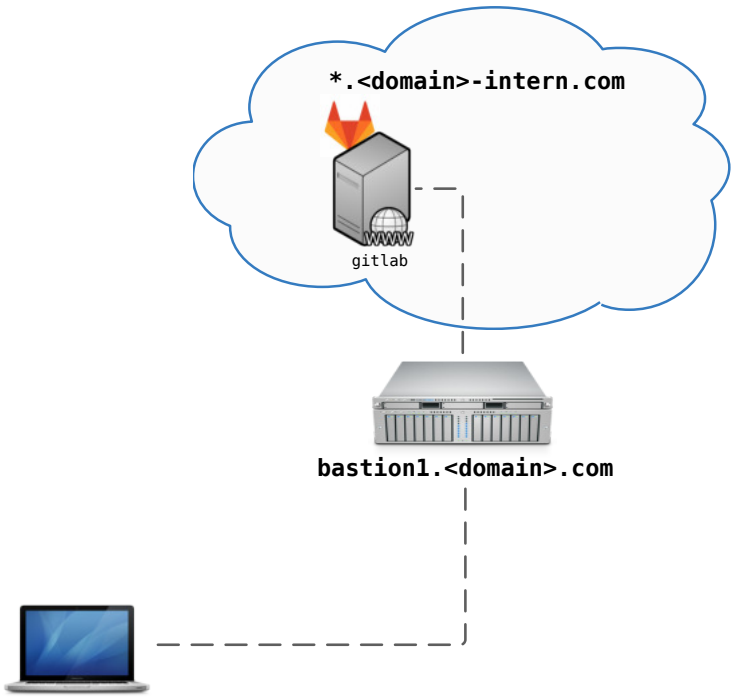
```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
- Connection/SSH/Tunnels: Port 1080, Dynamic
```

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

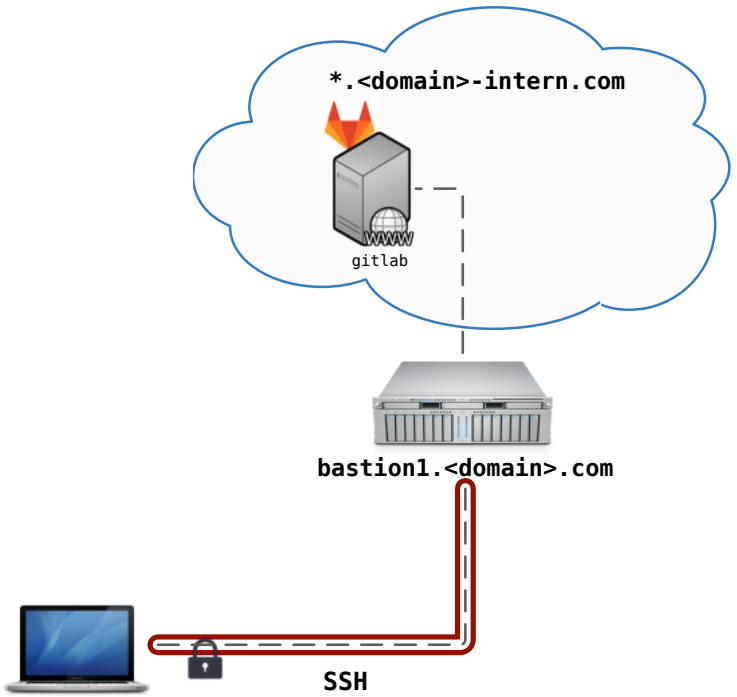
```
Host uni_*
    User <LOGIN>
    Port 8022

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

SSH Advanced Usage: ProxyCommand



SSH Advanced Usage: ProxyCommand



PuTTY / PLink / Pageant (Windows)

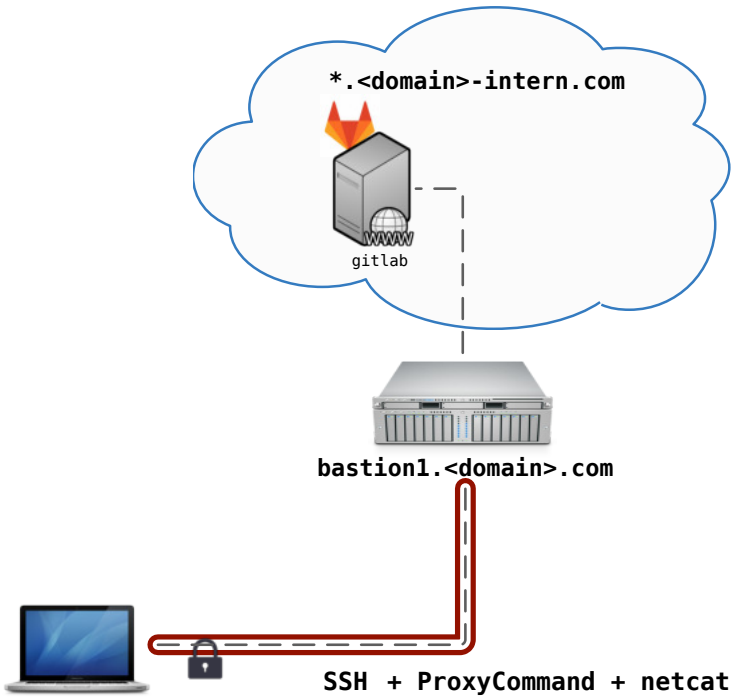
```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
```

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

```
Host uni_*
    User <LOGIN>
    Port 8022

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

SSH Advanced Usage: ProxyCommand



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
```

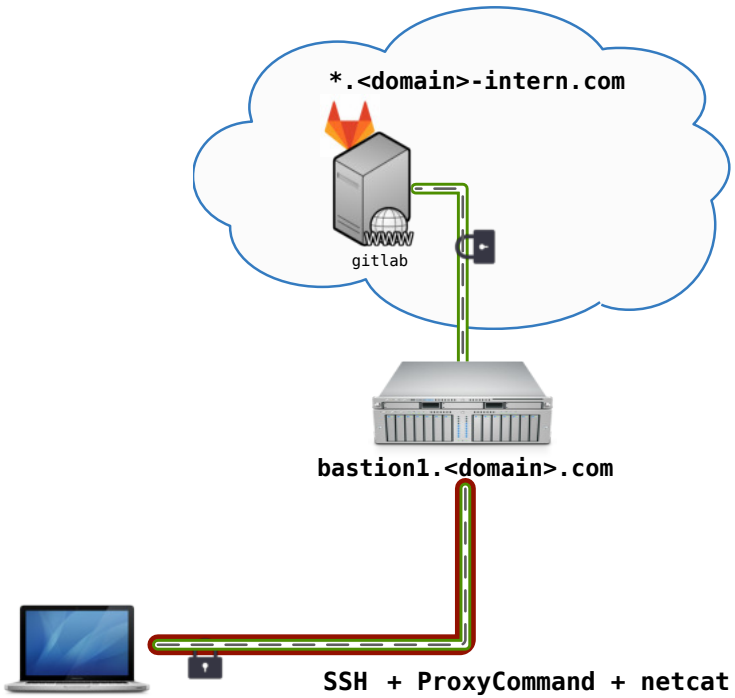
OpenSSH ~/.ssh/config (Mac / Linux)

```
Host uni_*
    User <LOGIN>
    Port 8022

Host uni_bastion1
   Hostname bastion1.<domain>.com

Host uni_gitlab
   Hostname gitlab
   ProxyCommand ssh -q uni_bastion1 "nc %h %p"
```

SSH Advanced Usage: ProxyCommand



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
```

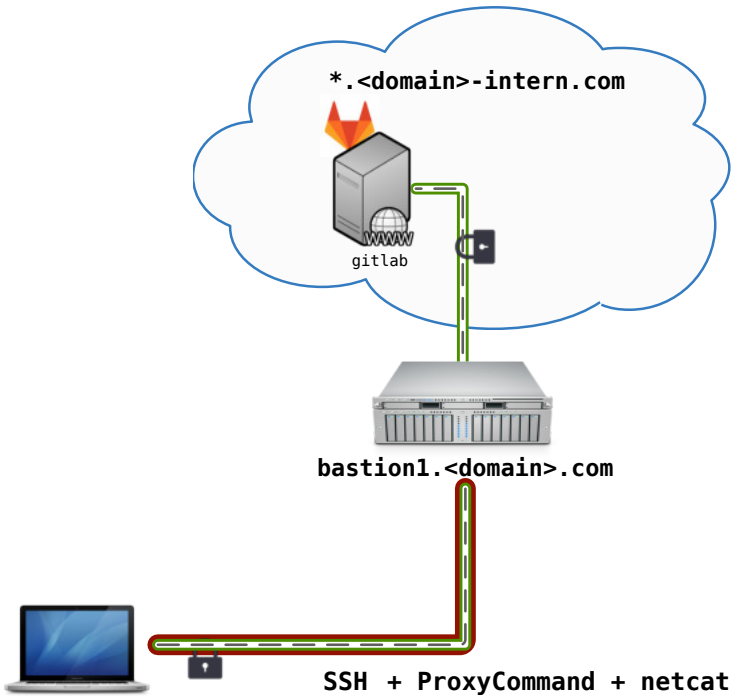
OpenSSH ~/.ssh/config (Mac / Linux)

```
Host uni_*
    User <LOGIN>
    Port 8022

Host uni_bastion1
   Hostname bastion1.<domain>.com

Host uni_gitlab
   Hostname gitlab
   ProxyCommand ssh -q uni_bastion1 "nc %h %p"
```

SSH Advanced Usage: ProxyCommand



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
   ProxyCommand ssh -q uni_bastion1 "nc %h %p"
```

SSH in Practice ~/.ssh/config

```
$> ssh [-X] [-p <port>] <login>@<hostname>
# Example:  ssh -p 8022 svarrette@access-chaos.uni.lu
```

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

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

    $> ssh cha<TAB>

SSH in Practice ~/.ssh/config

```
$> 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>

SSH in Practice ~/.ssh/config

```
$> 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>

```
$> ssh chaos-cluster
$> ssh work
$> ssh work.ext_ul
```

SSH in Practice: Main CLI commands

DSH - Distributed / DancerâĂŹs 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
```

DSH - Distributed / DancerâĂŹs 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 configuration ~/.dsh/dsh.conf

```
##########################################################
# ~/.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
```

DSH Basic Usage

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

Option	Description
`-c`	run the commands in parallel (default)
`-w`	run the commands in sequential
`-a`	run the command on all nodes listed in `machines.list`
`-g <group>`	restrict the commands to the hosts group `<group>`
`-m <hostname>`	run the command only on `hostname`

- **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

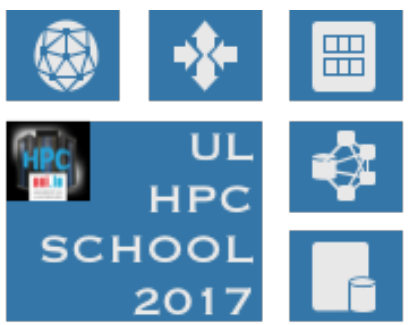
```
Defaults:<login> !requiretty
```

Summary

Reference Tutorial Source

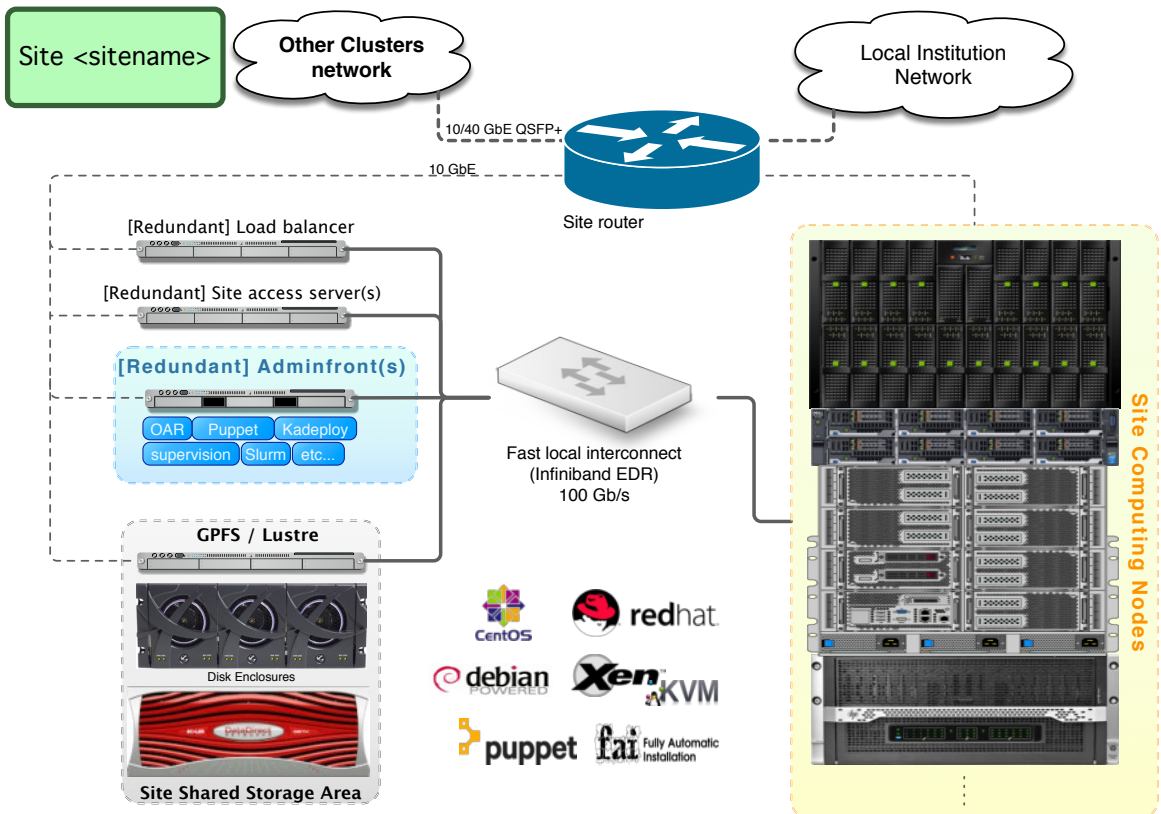
Tutorial Page:

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



Platform overview.

- Quick presentation of **UL HPC platform** and the new **Iris cluster**
  - ↪ as of 2017: **206.772 TFlops, 7952.4TB (shared)**
  - ↪ For more details: `http://hpc.uni.lu`



First connection & SSH setup

- **Obj**: Connecting for the 1st time & preparing your SSH environment

- **Step 1a**: Connect to UL HPC (Linux / Mac OS / Unix)
- **Step 1b**: Connect to UL HPC (Windows)
  - ↪ using **MobaXTerm** or **Putty**.
- **Step 2**: Connect from one cluster to the other
  - ↪ Learn how to connect from **one cluster to the other**.
- **Step 2bis**: Using SSH `proxycommand` to access the clusters
  - ↪ allow access from everywhere despite port filtering
  - ↪ use of *SSH aliases* to easier connection
- **Step 3**: Transferring data files
  - ↪ from your laptop to the clusters
  - ↪ cover both Linux / Mac and Windows users

First connection & SSH setup

- **Step 3a**:Transferring data files on Linux / OS X / Unix
  - ↪ use **command line tools** (**SCP**, **Rsync**)
- **Step 3b**: Windows / Linux / OS X / Unix GUI tools
  - ↪ Learn how to configure **Filezilla**
    - ✓ a **graphical tool** to transfer files to/from the clusters.
- **Step 3c**: Windows [MobaXterm] file transfert

Discovering & reserving HPC resources

- **Obj**: How to reserve resources & use them to **run your code** on it ?

Step 1: the working environment

- What **software** is installed on the nodes
- **where can I put my files**, my data, my results ?
  - ↪ How many **space** is available ?

Step 2: web monitoring interfaces

- What is the **status of the platform** ?
- **How many ressources** are available and when ?
- Why is my job in pending state ?

Discovering & reserving HPC resources

Step 3a: Reserving resources with Slurm

- Now I want to **run my script on the platform**.
  - ↪ What should I do ?
  - ↪ How to use **Slurm** scheduler on **iris** cluster ?

Step 3b: Reserving resources with OAR

- As above, yet using the OAR scheduler
  - ↪ available on **gaia**, **chaos**, **g5k** clusters ?

Discovering & reserving HPC resources

Step 4: Using modules

- I want to run a specific **version of my software**.
  - ↪ What software is available ?
  - ↪ How can I use them ?

Step 5 (advanced): Job management and Persistent Terminal Sessions using GNU Screen

- Each time I close my SSH connection, my job is killed.
  - ↪ How can I **make persistent terminal sessions**
  - ↪ . . . to execute my code wthout disconnections.
    - ✓ Pre-requisite: screen configuration file `~/.screenrc`
    - ✓ Basic commands
    - ✓ Sample Usage on the UL HPC platform: Kernel compilation

Summary

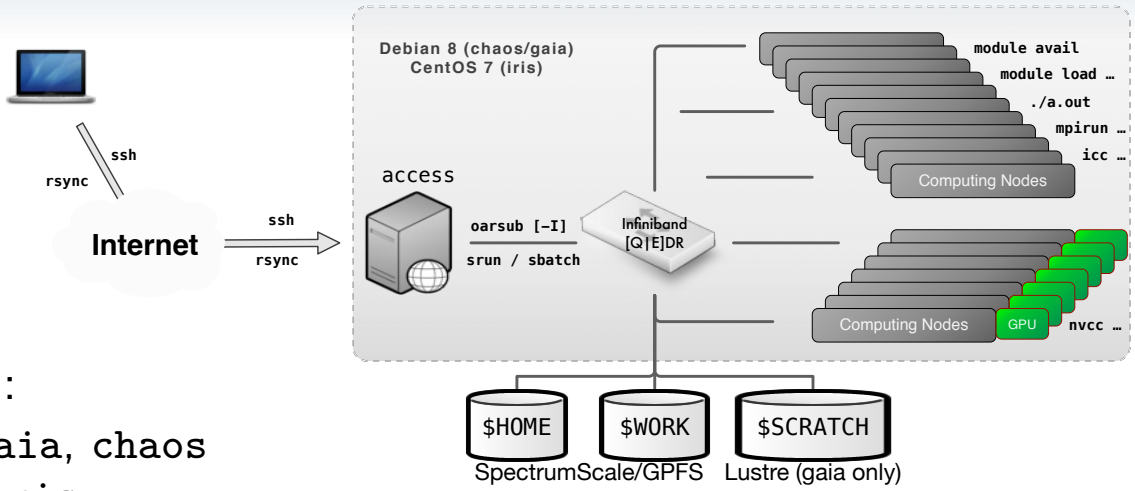
Hands-On 1: SSH Setup

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**

Hands-on 2: First steps on UL HPC

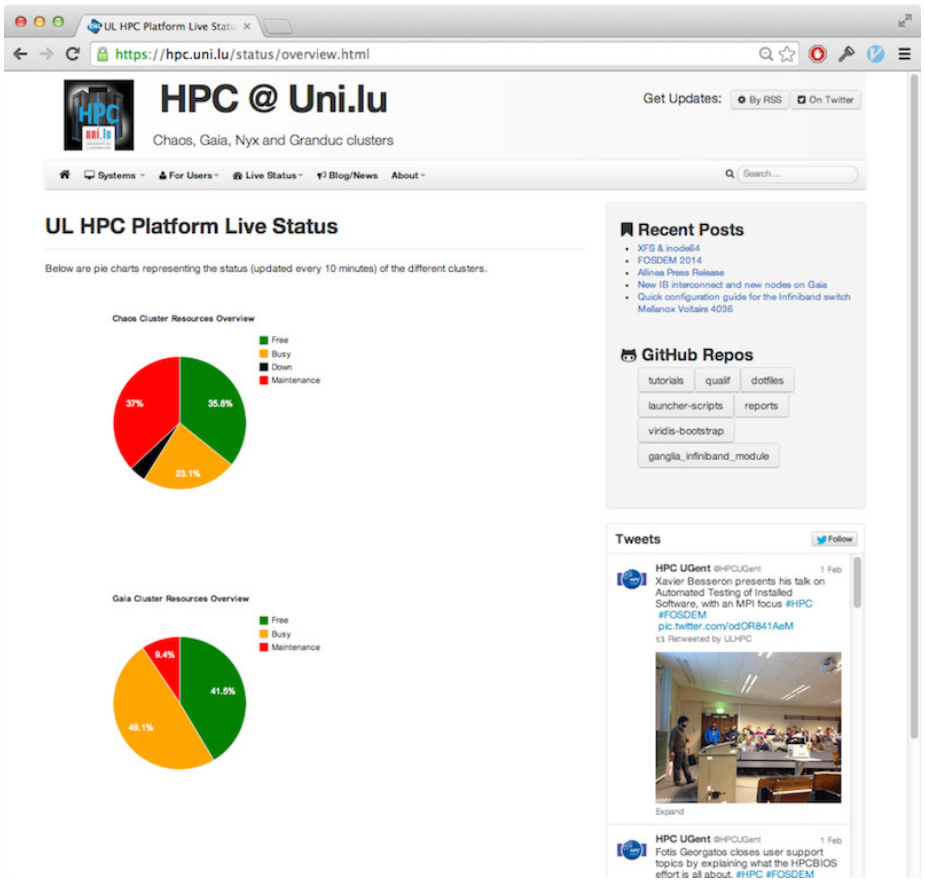


- UL HPC Environment
  - ↪ **Operating System**:
    - ✓ Debian 7 on `gaia`, `chaos`
    - ✓ CentOS 7 on `iris`
  - ↪ **Job Management**: `{ oarsub | srun/sbatch }`
  - ↪ **Environment modules**: `modules`
    - ✓ **Not** available on frontends, ***Only*** on compute nodes
  - ↪ (advanced) discovering GNU screen

Directory	Max size	Max #files	Backup
`$HOME` (`gaia`, `chaos`)	100 GB	1.000.000	YES
`$HOME` (`iris`)	500 GB	1.000.000	YES
`$WORK` (except `iris`)	3 TB		NO
`$SCRATCH` (except `iris`)	10 TB		NO

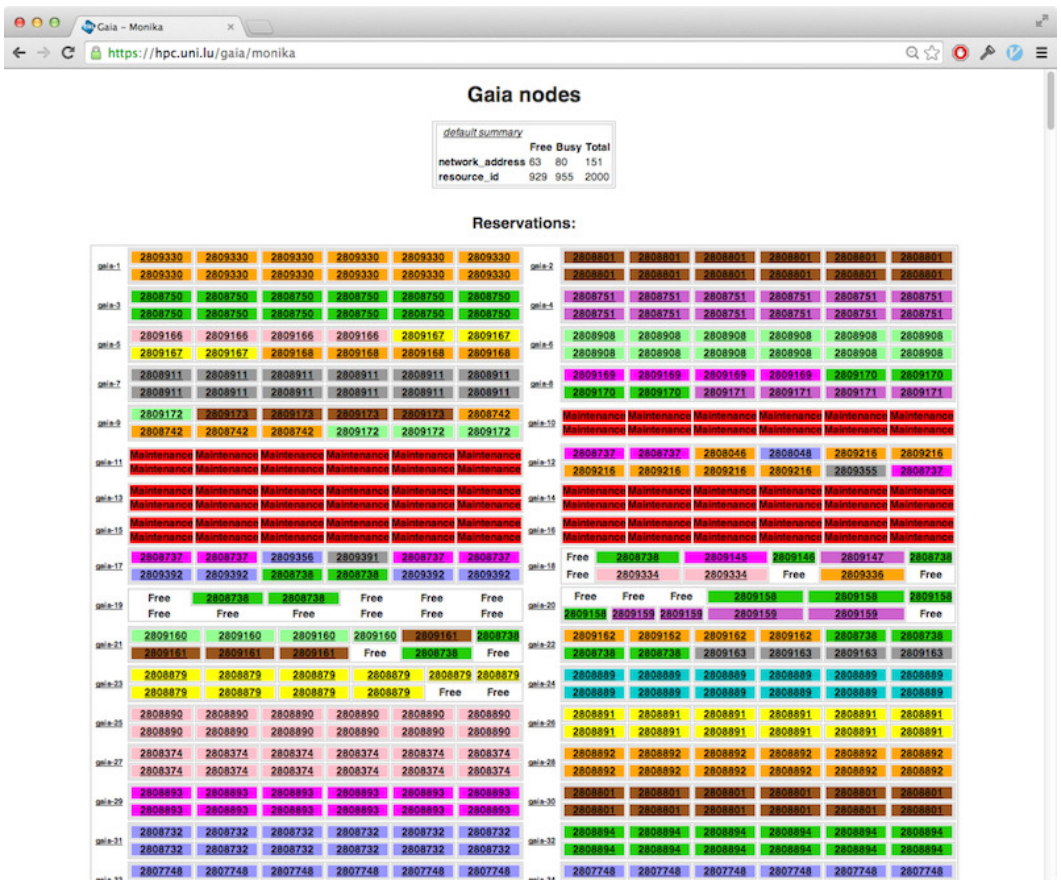
ULHPC Web monitoring interfaces

http://hpc.uni.lu/status/overview.html



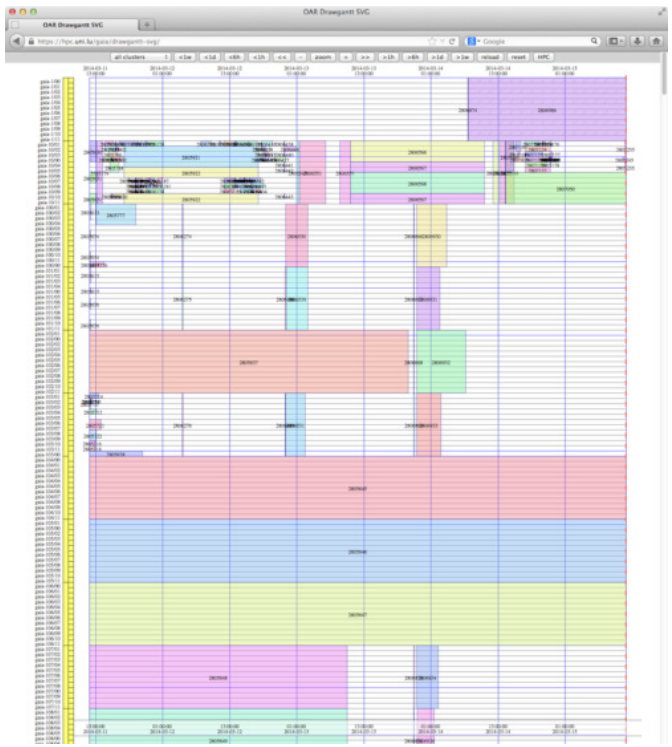
ULHPC Web monitoring interfaces

http://hpc.uni.lu/{iris,gaia,chaos,g5k}/monika



ULHPC Web monitoring interfaces

`http://hpc.uni.lu/{iris,gaia,chaos,g5k}/drawgantt`

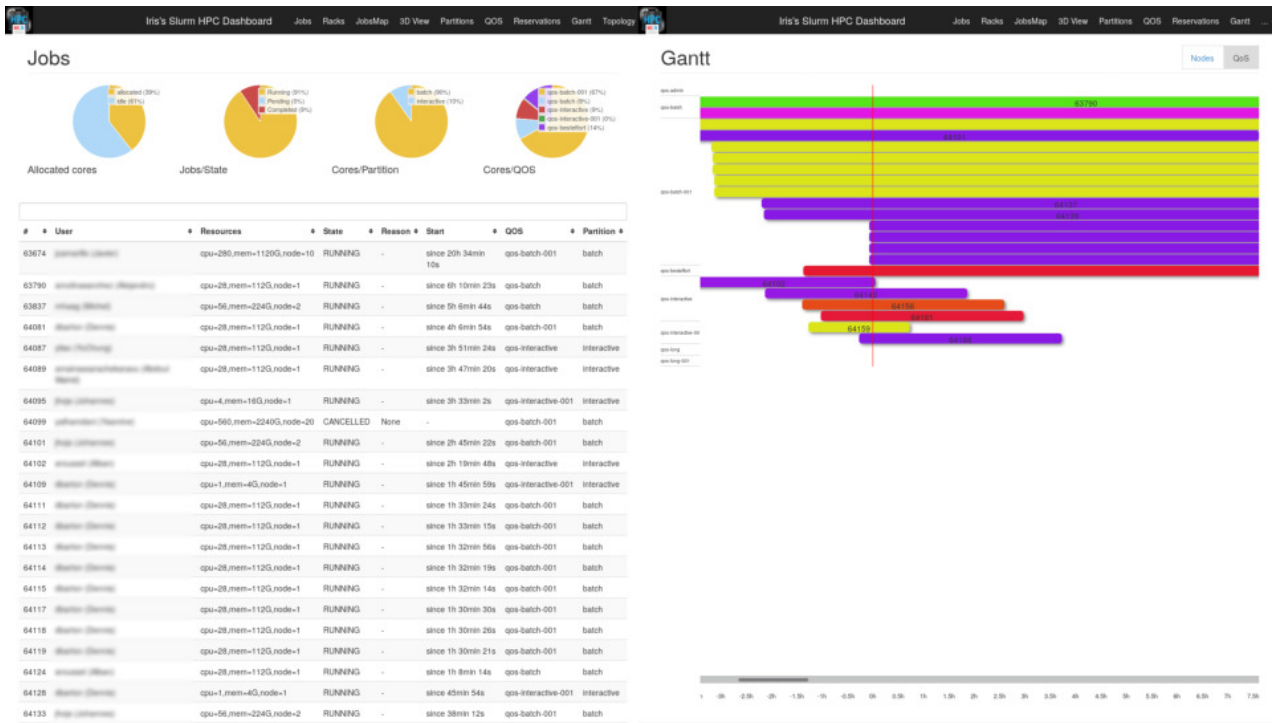


ULHPC Web monitoring interfaces

http://hpc.uni.lu/{iris,gaia,chaos,g5k}/ganglia

ULHPC Web monitoring interfaces

https://access-iris.uni.lu/slurm



Job management

If there are not enough resources available, use our reservations, add the parameters in red to your submission commands:

- OAR (Gaia)

```
$> oarsub -I -t inner=4248619
```

- SLURM (Iris)

```
$> srun -reservation=hpcschool -pty bash
```

Programming, quick start

- choose a command line text editor
- load modules
- run a Matlab script
- run a R script
- use the available compilers
- compile and run a simple MPI program

Questions?

http://hpc.uni.lu

High Performance Computing @ UL
Prof. Pascal Bouvry
Dr. Sebastien Varrette & the UL HPC Team
(V. Plugaru, S. Peter, H. Cartiaux & C. Parisot)
University of Luxembourg, Belval Campus
Maison du Nombre, 4th floor
2, avenue de l'Université
L-4365 Esch-sur-Alzette
mail: hpc@uni.lu