Tutorial: SSH

Secure SHell: Connect remotely anything, anywhere

UL High Performance Computing (HPC) Team

Sebastien Varrette

University of Luxembourg (UL), Luxembourg
http://hpc.uni.lu

Summary

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.

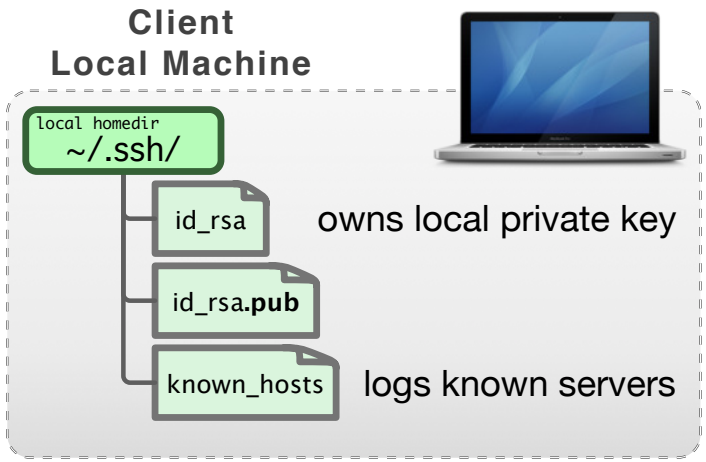
SSH: Secure Shell

- 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

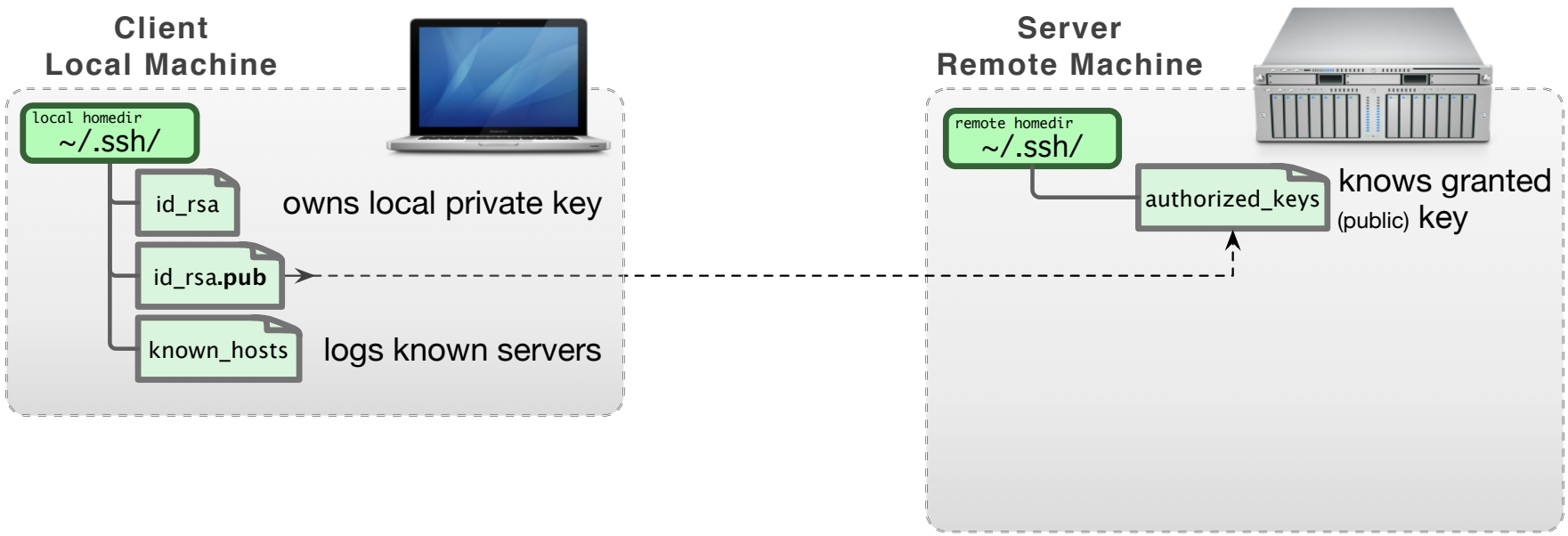
- SSH is used as a secure backbone channel for **many** tools
  - ↪ Remote shell **i.e** remote command line
  - ↪ File transfer: `rsync`, `scp`, `sftp`
  - ↪ versionning synchronization (`svn`, `git`), github, gitlab etc.

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

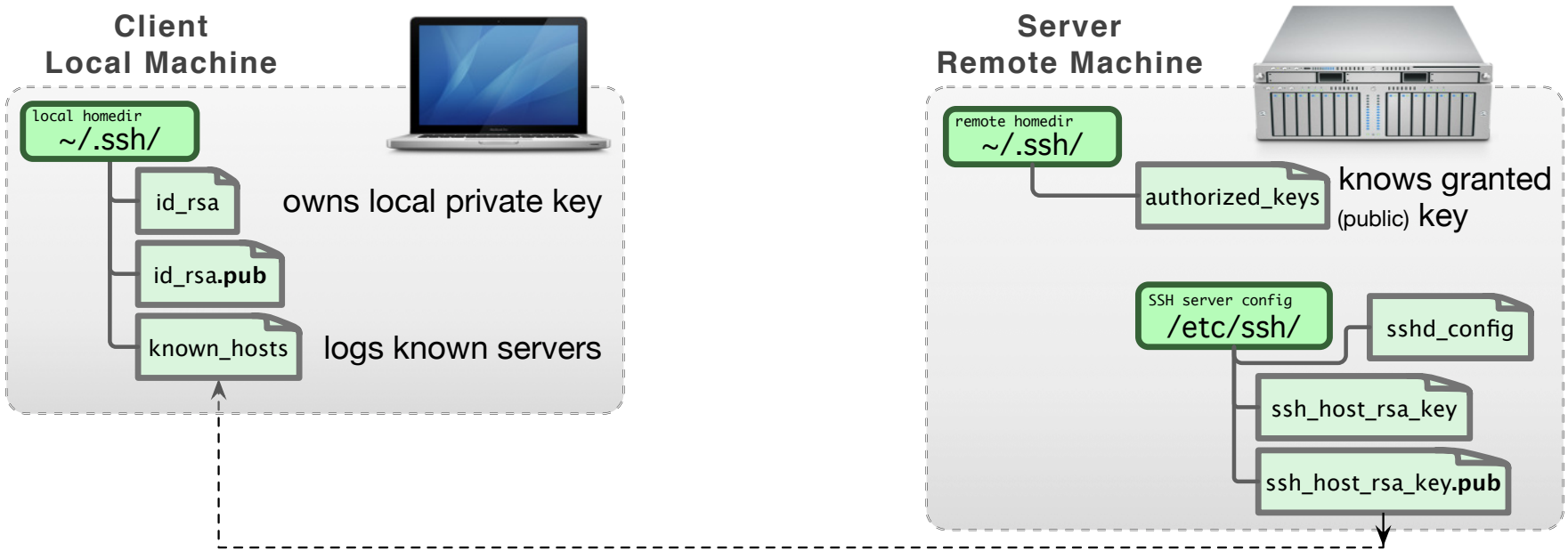
SSH: Public Key Authentication



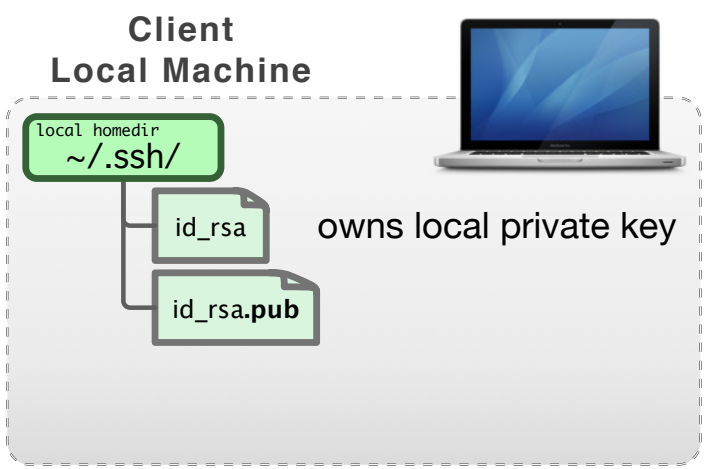
SSH: Public Key Authentication

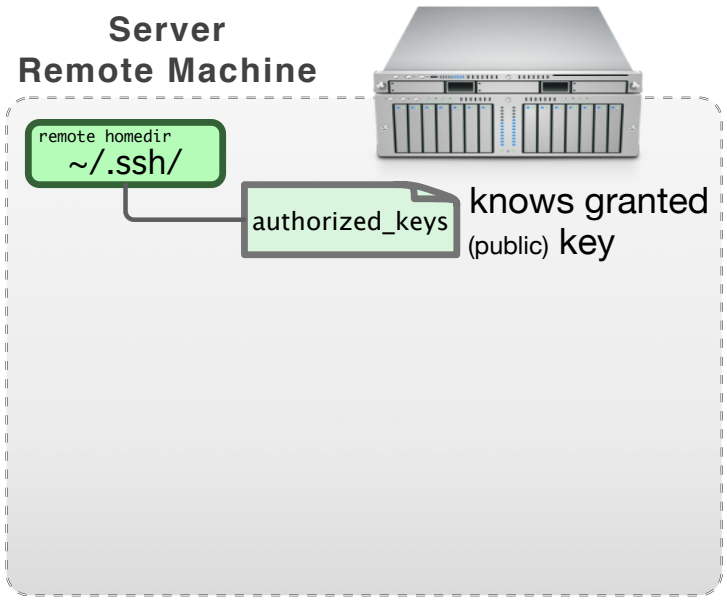


SSH: Public Key Authentication

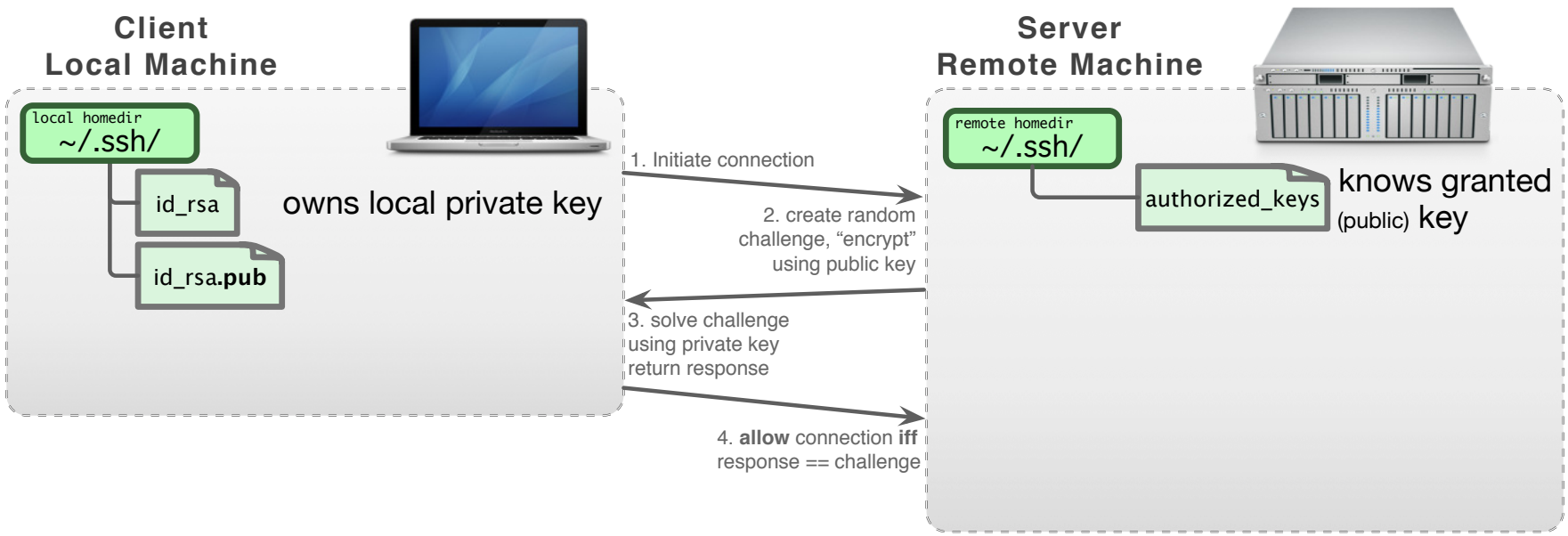


SSH: Public Key Authentication





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
```

Summary

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

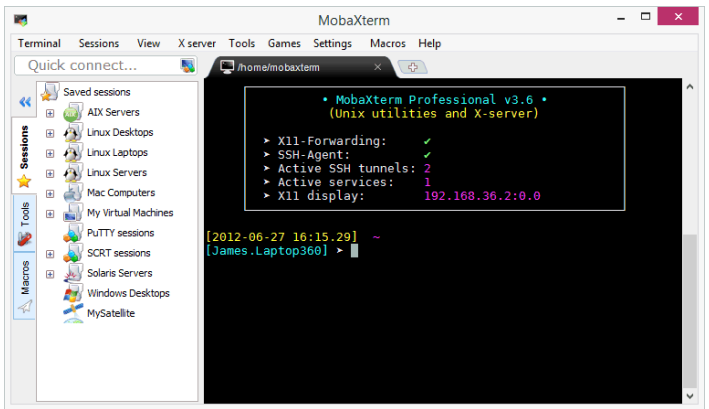
PuTTY ≠ OpenSSH

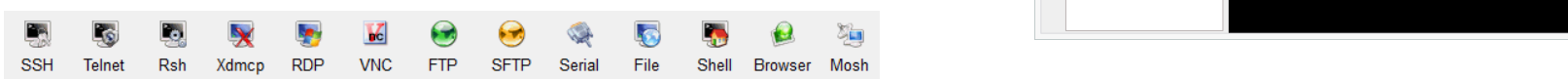
- Putty keys are **NOT** supported by OpenSSH (yet can be exported)
- Binding Pageant with OpenSSH agent is **NOT** natively supported
  - ↪ Third-party tools like ssh-pageant are made for that
  - ↪ Combine nicely with Git bash https://git-for-windows.github.io/
- with PLink, hostnames eventually refer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

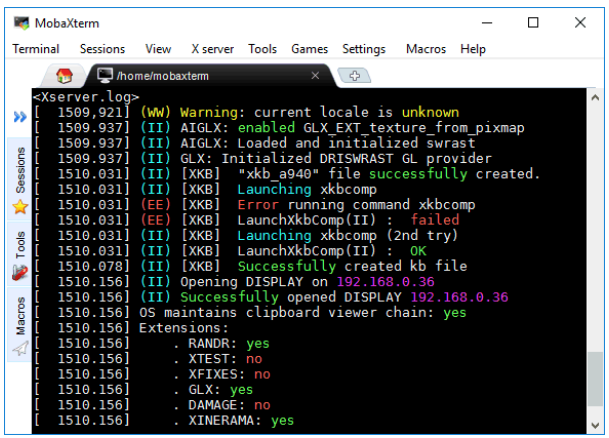
SSH Setup on Windows: the NEW way

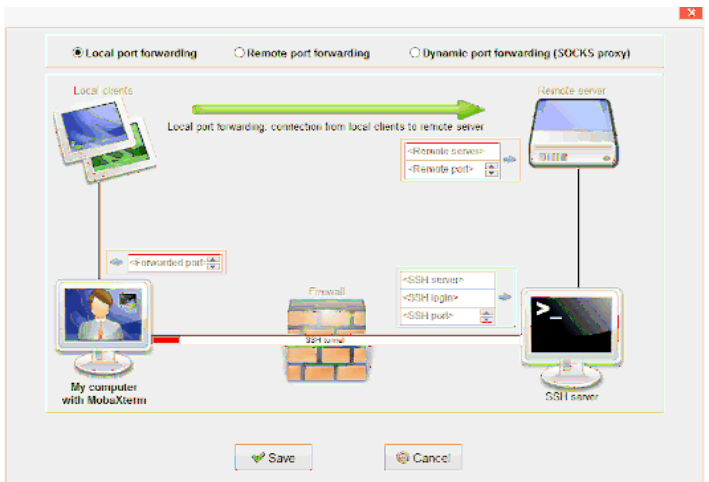
- Use MobaXterm!
  - ↪ [tabbed] Sessions management
  - ↪ X11 server w. enhanced X extensions
  - ↪ Graphical SFTP browser
  - ↪ SSH gateway / tunnels wizards
  - ↪ [remote] Text Editor
  - ↪ ...

http://mobaxterm.mobatek.net/



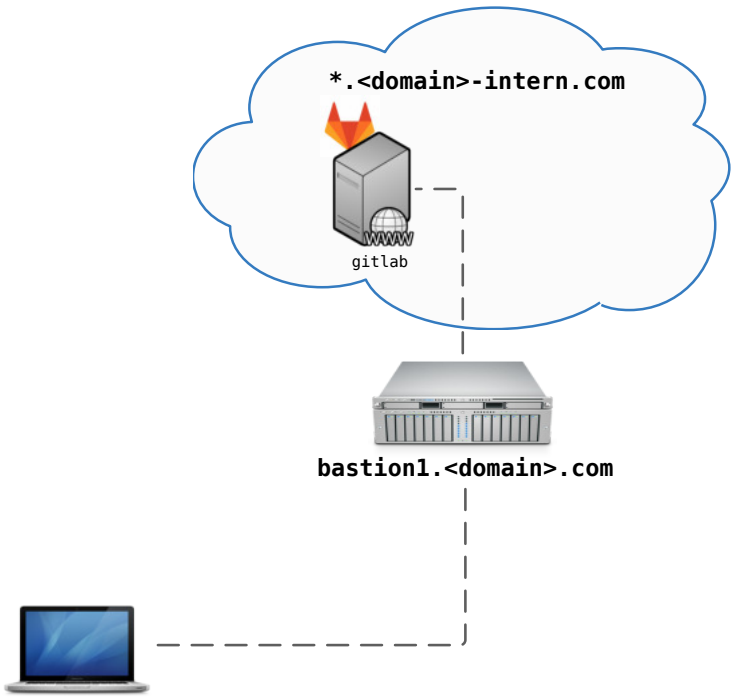




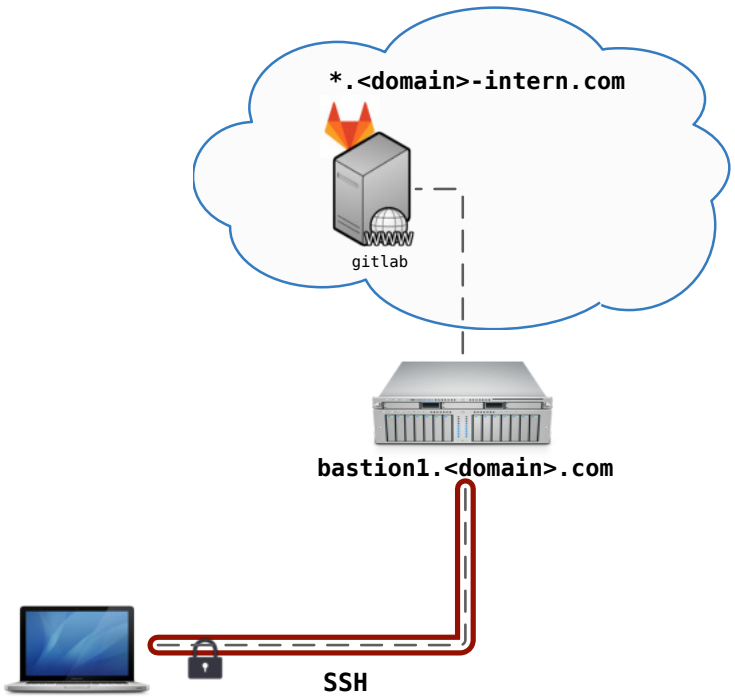


Summary

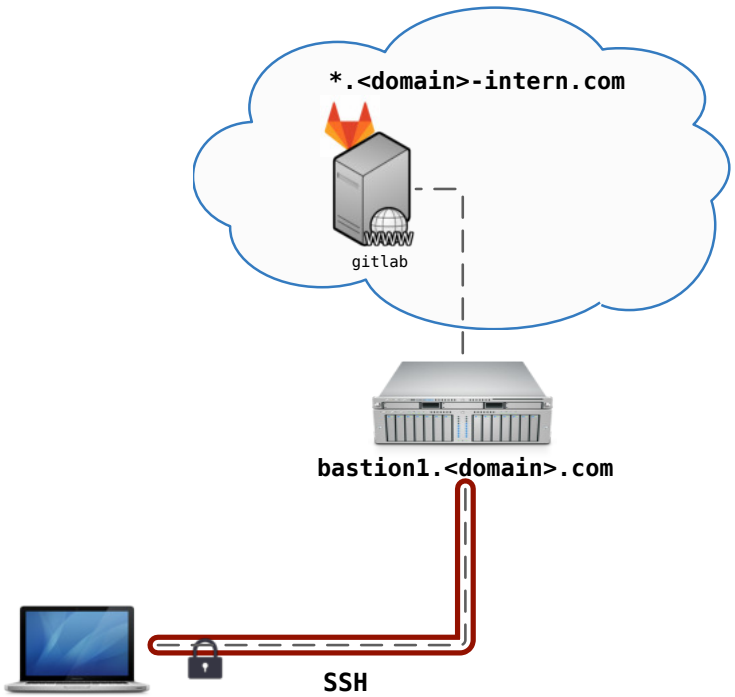
SSH Basic Usage



SSH Basic Usage



SSH Basic Usage

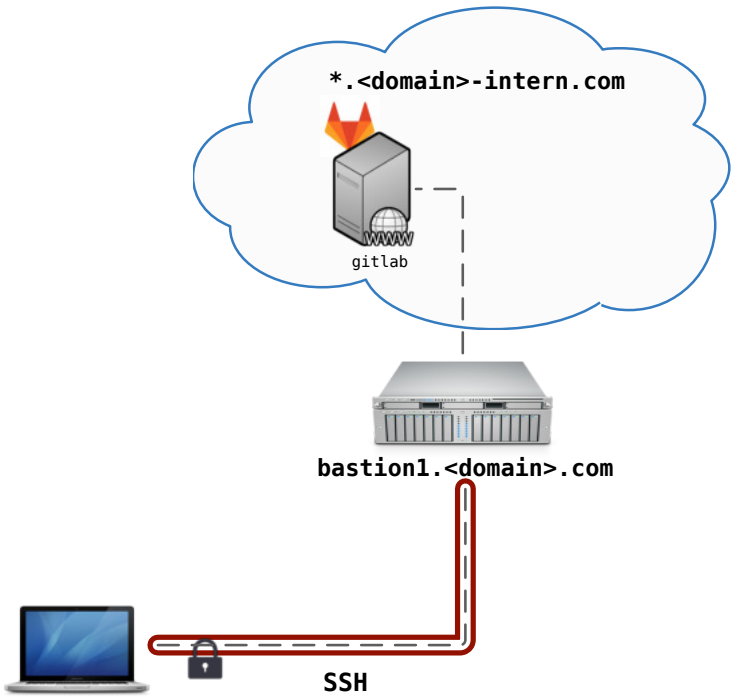


OpenSSH ~/.ssh/config (Mac / Linux)

```
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 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
```

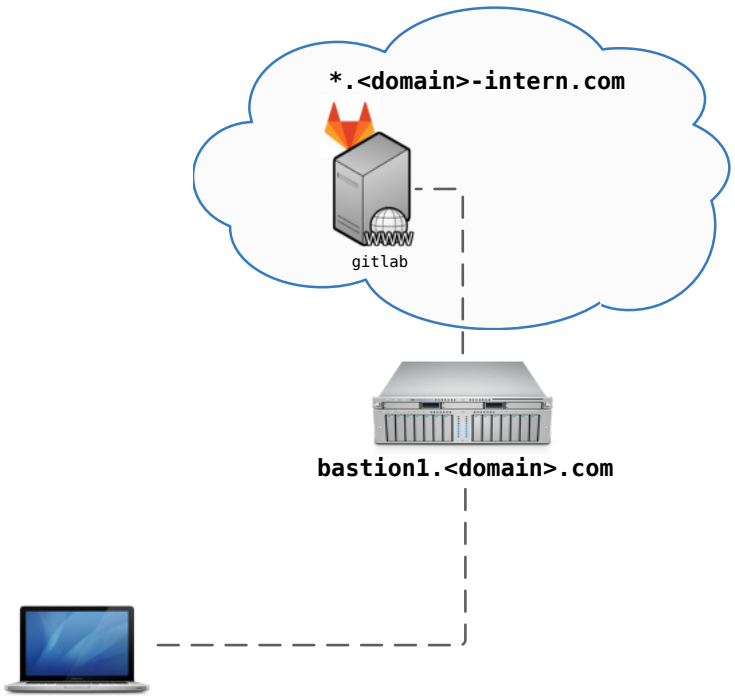
```
Host *.ext_ul
    ProxyCommand ssh -q chaos-cluster \
                 "nc -q 0 %h %p"
# UL HPC Platform -- http://hpc.uni.lu
Host chaos-cluster
    Hostname     access-chaos.uni.lu
Host gaia-cluster
    Hostname     access-gaia.uni.lu
Host iris-cluster
    Hostname     access-iris.uni.lu
Host *-cluster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

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

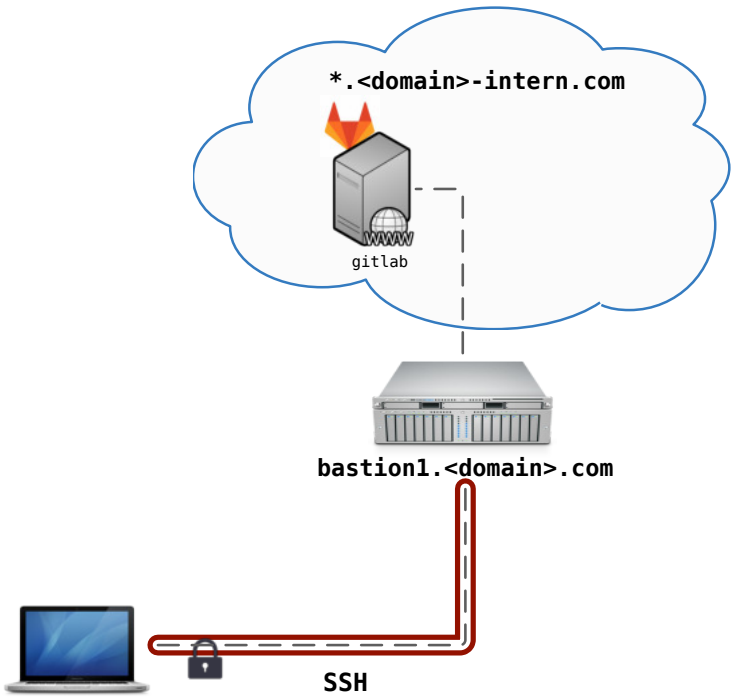
- ~/.ssh/config:
  - ↪ Simpler commands
  - ↪ Bash completion
    - $> ssh cha<TAB>

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

SSH Advanced Usage: SOCKS Proxy



SSH Advanced Usage: SOCKS Proxy



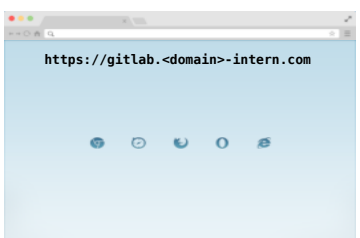
PuTTY / PLink / Pageant (Windows)

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

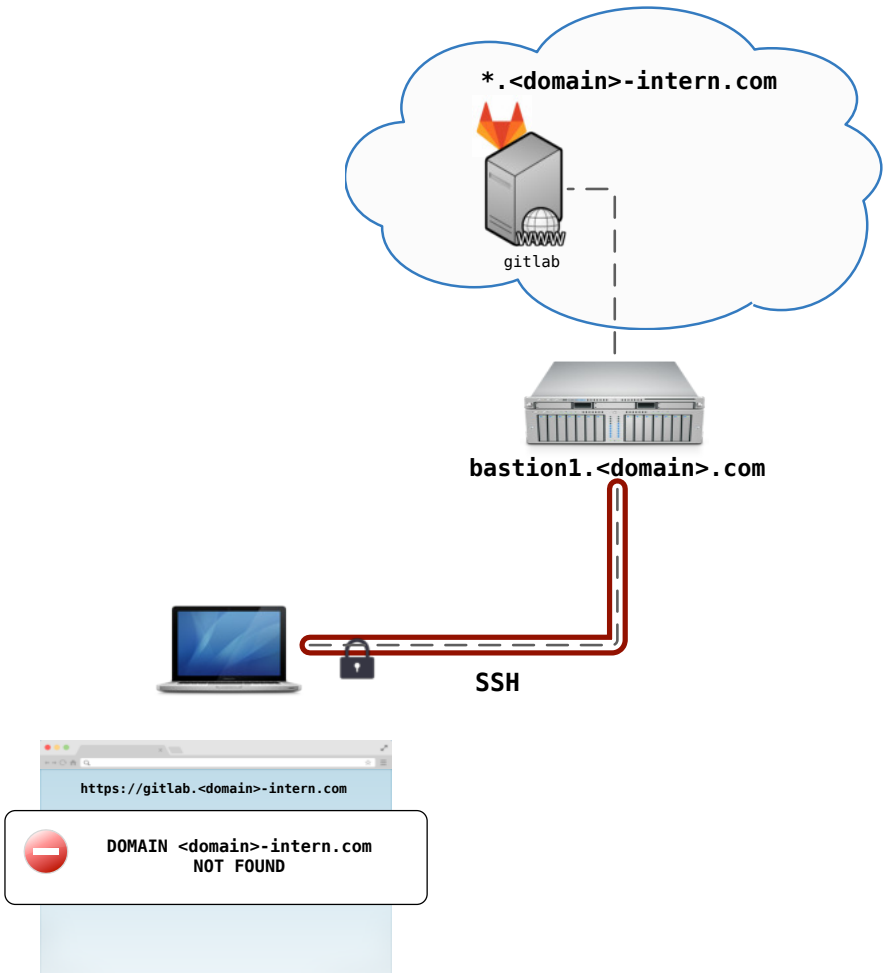
OpenSSH ~/.ssh/config (Mac / Linux)

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

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



SSH Advanced Usage: SOCKS Proxy



PuTTY / PLink / Pageant (Windows)

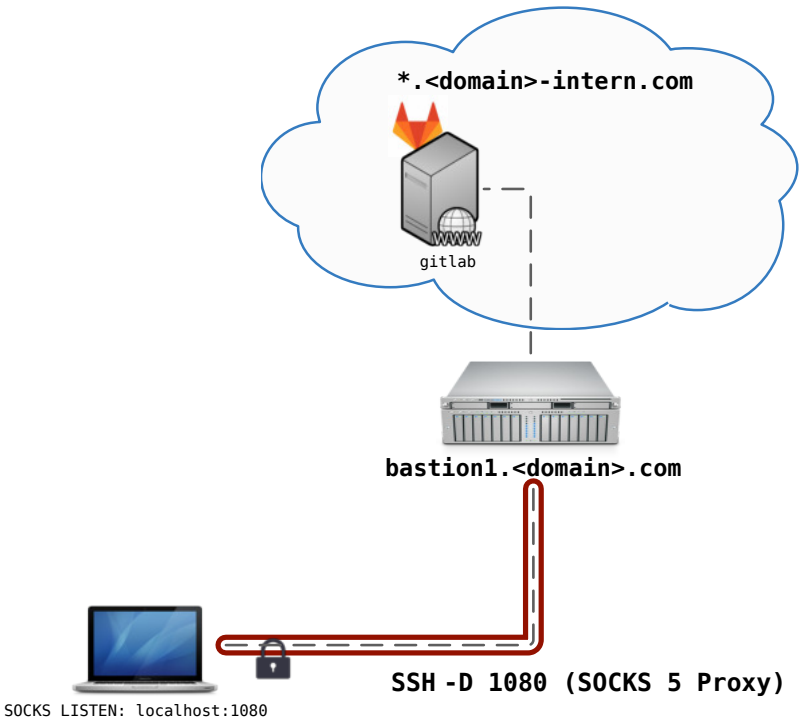
```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
```

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)

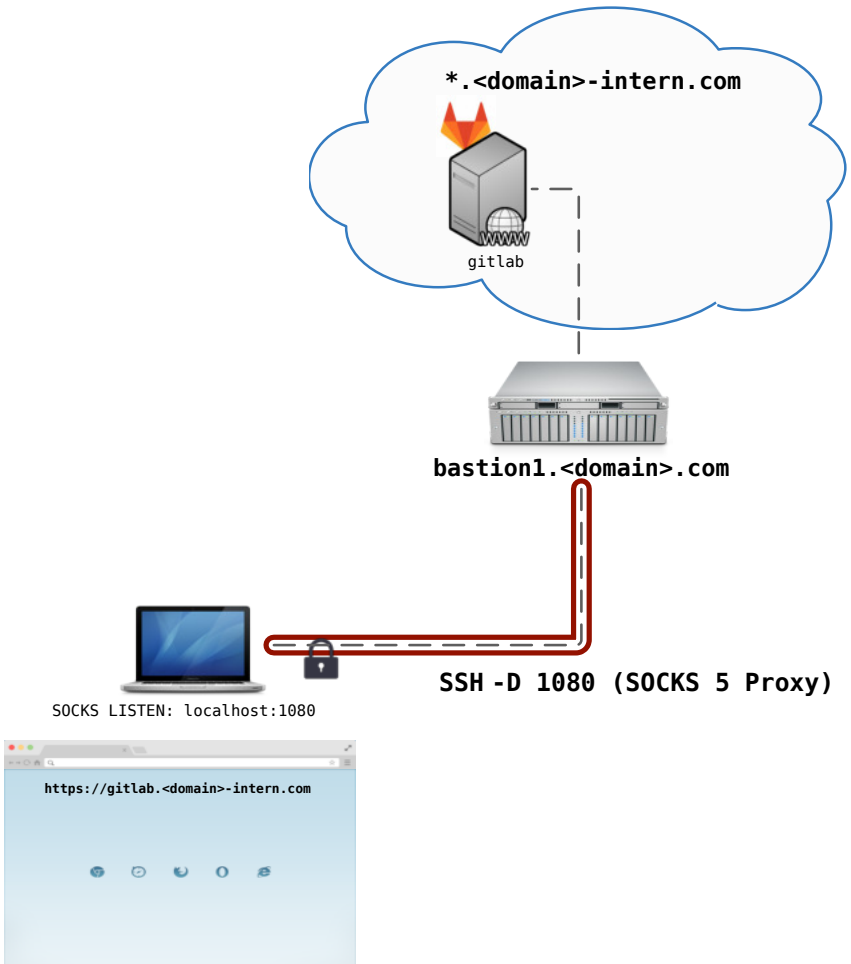
```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
- Connection/SSH/Tunnels: Port 1080, Dynamic
```

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)

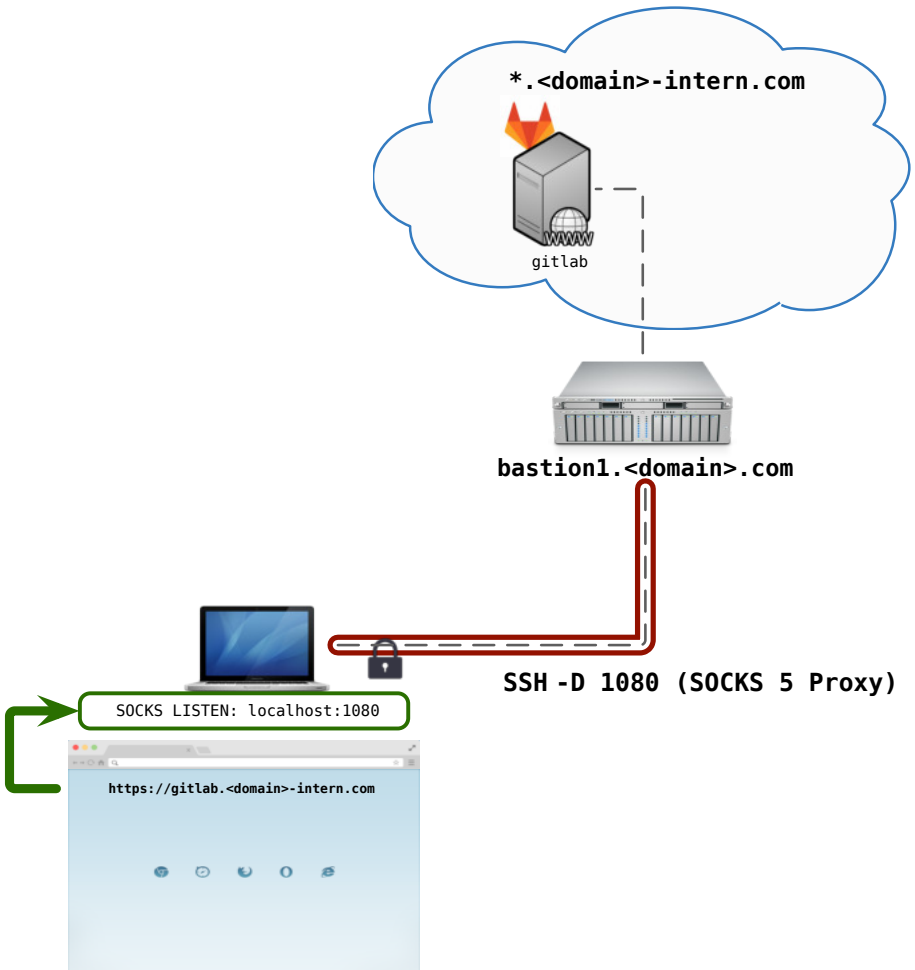
```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
- Connection/SSH/Tunnels: Port 1080, Dynamic
```

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)

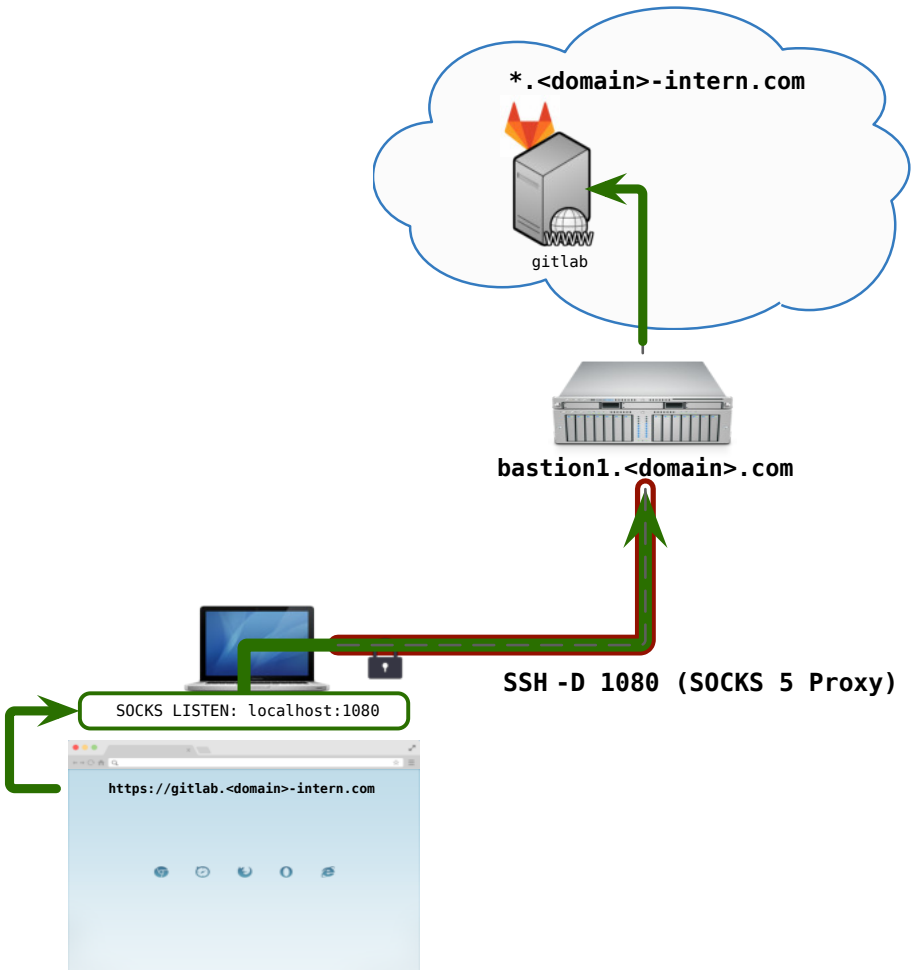
```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
- Connection/SSH/Tunnels: Port 1080, Dynamic
```

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)

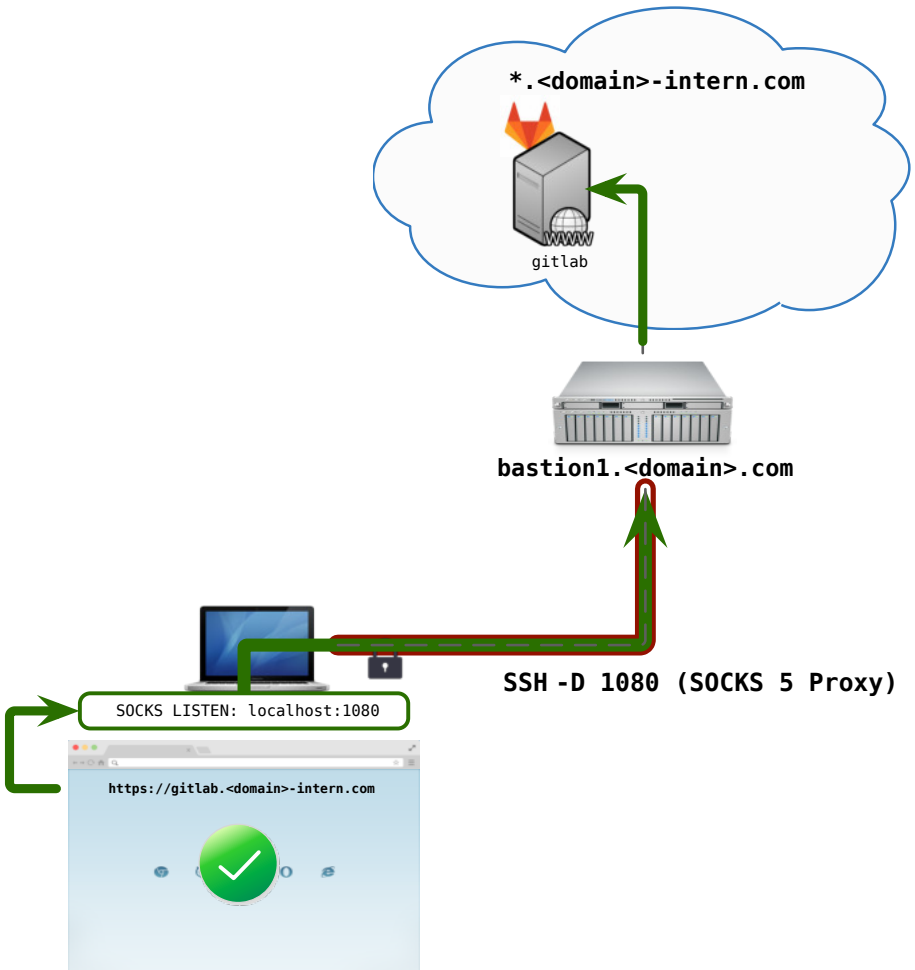
```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
- Connection/SSH/Tunnels: Port 1080, Dynamic
```

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)

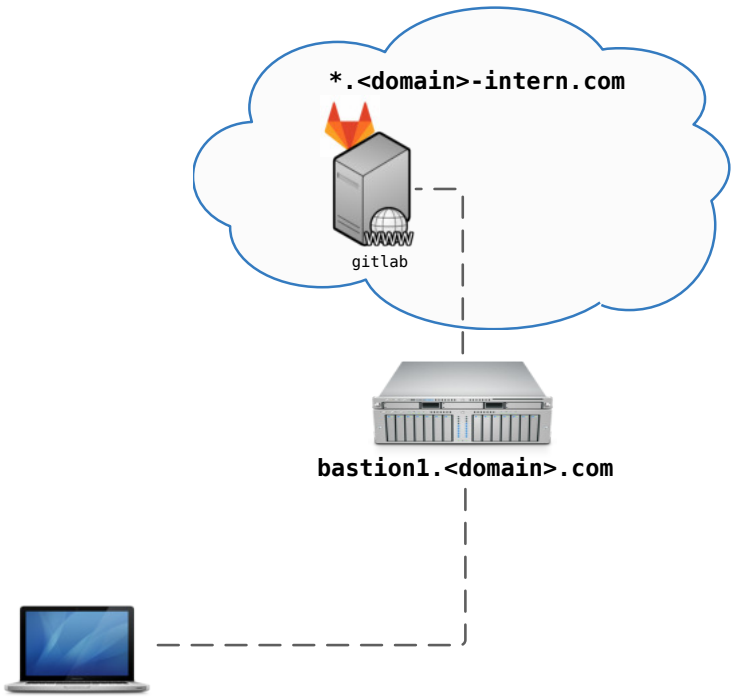
```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
- Connection/SSH/Tunnels: Port 1080, Dynamic
```

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

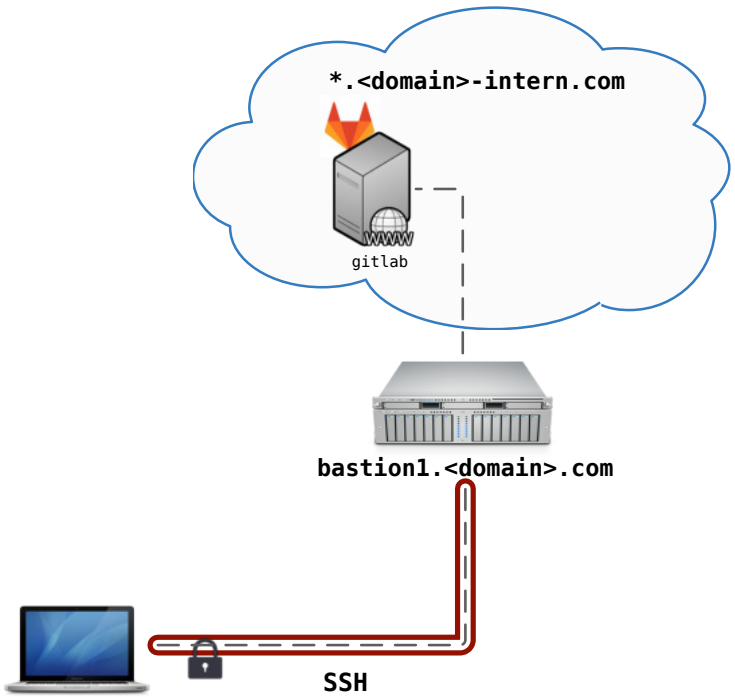
```
Host uni_*
    User <LOGIN>
    Port 8022

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

SSH Advanced Usage: ProxyCommand



SSH Advanced Usage: ProxyCommand



PuTTY / PLink / Pageant (Windows)

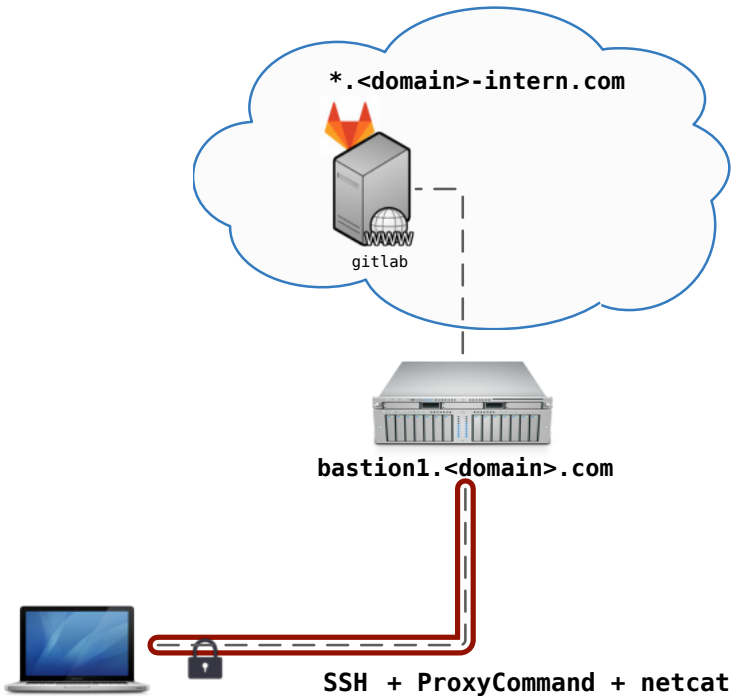
```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
```

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
```

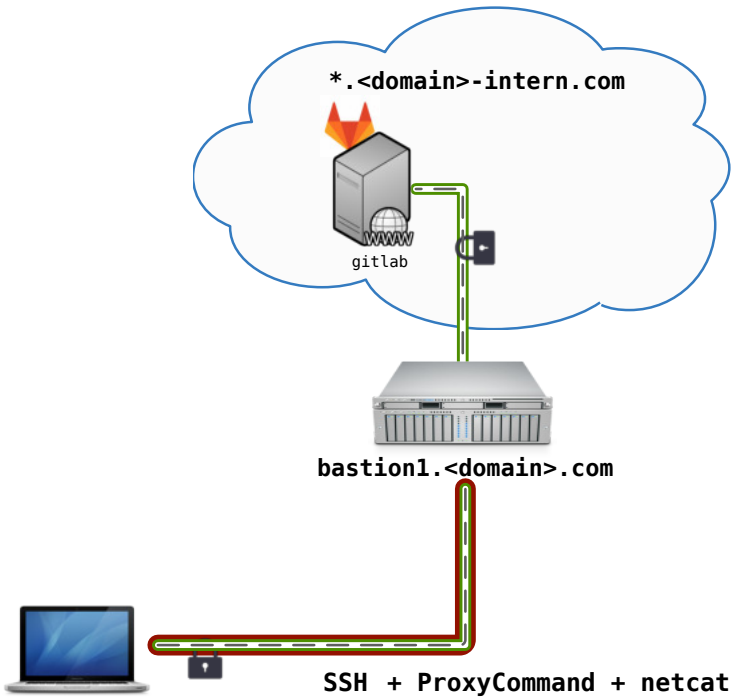
OpenSSH ~/.ssh/config (Mac / Linux)

```
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
```

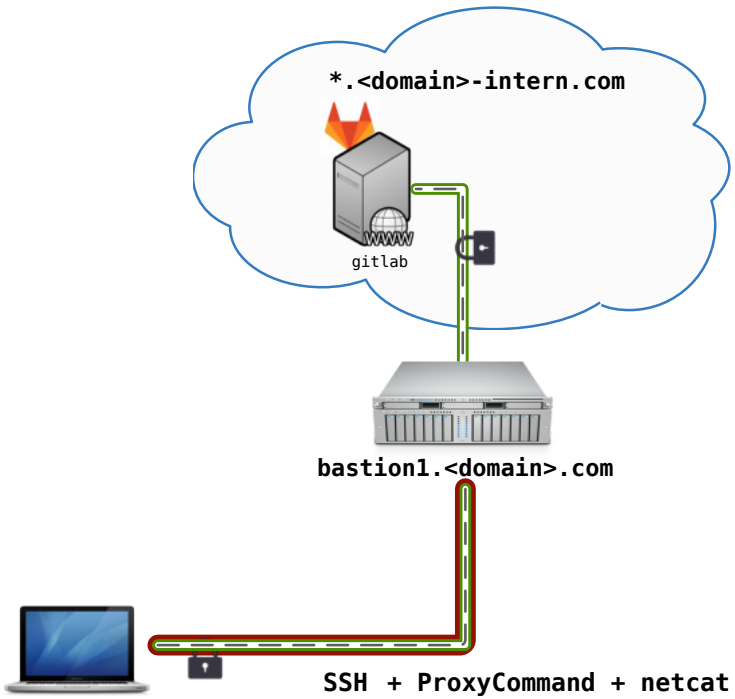
OpenSSH ~/.ssh/config (Mac / Linux)

```
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"
```

Summary

assh - Advanced SSH config

https://github.com/moul/advanced-ssh-config

- Transparent wrapper that make `~/.ssh/config` easier to manage
  - ↪ support for **templates**, **aliases**, **defaults**, **inheritance** etc.
  - ↪ **gateways**: transparent ssh connection chaining
  - ↪ more flexible command-line
    - ✓ **Ex**: Connect to `hosta` using `hostb` as a gateway

```
$> ssh hosta/hostb
```

  - ↪ drastically simplify your SSH config
  - ↪ Linux / Mac OS **only**

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

assh - Advanced SSH config

https://github.com/moul/advanced-ssh-config

- YAML-based **configuration**: in `~/.ssh/assh.yml`
  - ↪ use `.yml` extension, **NOT** `.yaml`
  - ↪ you can split configuration **i.e.** `~/.ssh/config.d/*.yml`
- **Hooks** support with advanced templated capabilities
  - ↪ Events: `BeforeConnect`, `OnConnect`, `OnDisconnect`
  - ↪ Exec drivers: `exec <binary> [args...]`
  - ↪ Notify driver: `notify <line:string...>`
- **Compilation**:

```
$> assh config build --ignore-known-hosts >  /.ssh/config
```

assh configuration ~/.ssh/assh.yml

```
# ~/.ssh/assh.yml - Advanced SSH Config
################################################
# Global (default) SSH flags
defaults:
  ForwardX11:      no
  ForwardAgent:    no
  ConnectTimeout: 15
  #AddKeysToAgent: yes
  Compression:     yes
  HashKnownHosts: no
  ServerAliveInterval: 60
  ServerAliveCountMax: 30
  #ControlMaster:  auto
  #ControlPath:    ~/.ssh/sockets/ssh-socket-%r-%h-%p.sock
  #ControlPersist: 600
includes:
- ~/.ssh/config.d/*.yml
- ~/.ssh/config.d/custom/*.yml
```

assh configuration: Templates

```
# ~/.ssh/config.d/templates.yml - General templates
##################################################
templates:
  # Public zone, feat. servers typically reachable from the outside
  DMZ:
    Hostname: "%h.domain.org"
    User: $USER
    Port: 22
  # internal [private] zone for the WORK domain
  WORKi:
    Inherits: DMZ
    Gateways:
    - direct         # try direct connection first...
    - gw             # ... then try through this [public] host
    - anotherserver # ... then try through this other [public] host
```

assh configuration: Hosts

```
# ~/.ssh/config.d/work.yml - ASSH config for your working place
hosts:
  #### WORK gateways / Externally accessible nodes
  mygatewayserver:    # 'ssh mygatewayserver'
    Inherits: DMZ     # eq. ssh -p 22 $USER@mygatewayserver.domain.org
    Aliases:
    - gw              # more easier to type 'ssh gw'
  anotherserver:      # 'ssh anotherserver'
    Inherits: DMZ     # eq. ssh -p 2222 $USER@anotherserver.domain.org
    Port: 2222        # custom port here
  #### WORK internal servers
  workstation:
    Inherits: WORKi
    Hostname: 10.XX.XX.XX     # if fixed IP and no DNS
    User: local
  gitlab:
    Inherits: WORKi
  storage:
    Inherits: WORKi
    IdentityFile: ~/.ssh/id_special_rsa
```

assh Basic Usage

```
$> assh config build --ignore-known-hosts >  /.ssh/config
```

assh Basic Usage

```
$> assh config build --ignore-known-hosts >  /.ssh/config
```

- Once ~/.ssh/config is compiled:

```
$> ssh <host>                                  # connect to <host>
```

```
$> ssh <host>/<gw>       # connect though (non-configured) <gw>
```

assh Basic Usage

```
$> assh config build --ignore-known-hosts >  /.ssh/config
```

- Once ~/.ssh/config is compiled:

```
$> ssh <host>                                  # connect to <host>
```

```
$> ssh <host>/<gw>       # connect though (non-configured) <gw>
```

```
$> assh connect --dry-run <host>  # dry-run verbose connection
```

assh Advanced Usage

- If you enable multiplexing / `Control{Master,Path}` settings

```
defaults:
  ControlMaster:  auto
  ControlPath:    ~/.ssh/sockets/ssh-socket-%r-%h-%p.sock
  ControlPersist: 600
```

```
$> assh sockets list            # list (opened) control sockets
```

- if you start to experience multiplexing issues:
  - ↪ **Ex**: non-responding SSH connection etc.

```
$> assh sockets flush                    # Close control sockets
```

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
```

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 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
```

Questions?

http://hpc.uni.lu

Prof. Pascal Bouvry
Dr. Sebastien Varrette & The UL 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