Secure connection with ssh

Parallel Programming of High Performance Systems | 9.3.2020 | Hager – Bader – Weinberg
Overview

• ssh provides secure connections via insecure channels (Internet, telephone lines, …)
• “Secure” means
  • User is authenticated to the system
  • System is authenticated to the user
  • All transmitted data is encrypted
• Technology
  • Asymmetric encryption algorithm („Public Key“) for authentication and determination of a „Session Key“
  • Symmetric encryption of data transfer using Session Key
• Interactive work and data transfer are possible
• X11 and other protocols can be “tunneled” through an ssh connection
• Drawback: encryption and decryption incur computational costs on both ends
Asymmetric Encryption: Basics

• Every user owns a pair of keys (numbers), one private and one public
  • Public key can (and should) be known to everybody wishing to communicate with the user
  • Private key is secret and can be stored in an encrypted format if required
• A message will be encrypted with one of the keys and can only be decrypted with the corresponding other key

With ssh, the session key is negotiated and authentication is performed using this mechanism
Using ssh in Practice: Basic Functions

- Most simple case: password authentication
  
  `ssh [<username>@]<Remote-Host> [<command>]`

  - Runs the command on remote host or opens a shell
  - User is authenticated using password only
  - Host authentication is performed via fingerprint comparison (user responsibility)
    - Warning on first login
  - Passwords and data are always encrypted

- Data transfer: `scp [-r] <source(s)> <target>`
  - Remote source or target:
    `<username>@<Remote-Host>:<path>`
  - Wildcards are allowed (remote wildcards must be protected from local shell!)
  - `-r` allows recursive copying into subfolders

- FTP substitute: `sftp [<username>@]<Remote-Host>`
Using ssh in Practice: Basic Functions

• Examples

```
ssh lu65fok@supermic.smuc.lrz.de

scp -r public_html/* weinberg@webserv.abc.de:public_html
scp -r weinberg@webserv.abc.de:public_html/* public_html
scp -r weinberg@webserv.abc.de:public_html .
scp -r public_html weinberg@webserv.abc.de:

ssh -f weinberg@physik.uni-muenchen.de xterm
```

Option `–f` sends `ssh` to background!
Using ssh in Practice: Basic Functions

- On the first ssh connection to a new host, user must verify the host using its “key fingerprint”:

  
  unrz55@altix:~ > ssh snode023
  The authenticity of host 'snode023 (192.168.80.37)' can't be established. 
  Are you sure you want to continue connecting (yes/no)? yes
  Warning: Permanently added 'snode023,192.168.80.37' (RSA) to the list of known hosts.
  Linux snode023 2.4.21 #1 SMP Wed Jan 7 14:20:51 CET 2004 i686 unknown
  ...

- Calculating the fingerprint of a public key with ‘ssh-keygen -l’:

  unrz55@altix:~ > ssh-keygen -l
  Enter file in which the key is (/home/rzsunhome/unrz/unrz55/.ssh/id_rsa): /tmp/testkey.pub
Authentication Using Public Key

• When the user’s public key is deposited at the remote host, the user can be authenticated without a password
  • Only the private key “fits” the corresponding public key
• Private Key should be encrypted using a passphrase (key theft prevention!)
  • Recommended length of passphrase: > 20 characters
  • Empty passphrase is allowed, but not recommended
• Problem: Key format and file structure/layout is not standardized across different ssh variants
  • OpenSSH (http://www.openssh.org/)
    • free (BSD license)
    • Standard variant unter Linux
  • ssh by SSH Communications Security (http://www.ssh.com/)
    • Computing centers are licensees for non-commercial use
• Version identification: ssh -V
Authentication Using Public Key: Generating Key Pairs

- Different protocol versions (1, [1.5] and 2)
  - Different key formats for different protocols
  - Version 2 is strongly recommended (and should be default today)!
    - Attacks/exploits on earlier versions
    - Problems with V1 fallback
- Generating a key pair using `ssh-keygen`:
  - Protocol V2 (OpenSSH):
    `ssh-keygen -t dsa`  // rsa also available
  - Protocol V2 (ssh2):
    `ssh-keygen`

- For security reasons, **always specify a passphrase**!
  - Other systems might be compromised after key theft if private key is not protected
Authentication Using Public Key: Example for Generating Key Pairs

- **OpenSSH**:  
  
  ```bash
  lu65fok@badwlrz-clvwb01:~> ssh-keygen -t dsa
  • Generating public/private dsa key pair.
  • Enter file in which to save the key (/home/lu65fok/.ssh/id_rsa):
  • Enter passphrase (empty for no passphrase):
  • Enter same passphrase again:
  • Your identification has been saved in /home/lu65fok/.ssh/id_rsa.
  • Your public key has been saved in /home/lu65fok/.ssh/id_rsa.pub.
  • The key fingerprint is:
  • The key's randomart image is:
    ++-[ DSA 1024]----+
    |o....            |
    |o.oo .           |
    |+B..+            |
    |oo@o.o |
    |.+E. o S        |
    |  .  . o         |
    |      .          |
    |                 |
    |                 |
    +--[MD5]----------+
  • lu65fok@badwlrz-clvwb01:~>
  ```

  Available since OpenSSH V5
User Configuration, Client Side OpenSSH

- $HOME/
- .ssh/
  - Private key: id_dsa
  - Public key: id_dsa.pub
  - Client-side config: config
  - List of known hosts with public keys: known_hosts
User Configuration, Server Side OpenSSH

$HOME/

.ssh/

authorized_keys

All public keys for authorized access to this account
Authentication Using Public Key: Key Deposit at Remote Host

- Encrypted connections without passwords
  - For convenience, use in shell scripts etc.
- OpenSSH:
  - Append public key on remote host to
    $$HOME/.ssh/authorized_keys$$

Example section from `authorized_keys` file:

```bash
ssh-dss AAAAB3NzaC1kc3MAAACBAMt5lF7uviABLBMQ9hgPw851QPWR/17WGAA5QeRwLFvUrnnMd/y jhVDiq+eQ74+BdSYMvoNgWyH/3H+o+Dq7rHbku3e2Y5h/MQ/6Q13aw0nXHkCQcgNu3K3hKJNg11I1/7 LzriJW0Oax3afP1VR7LSQvgv/lJxU/hD/O4/2gVAAYAFQCwU/Wtr/sbqdp8a8X951anKgBVQAAAE1 bshxkaadcg4y0dLQ1ZI9bNRTQznEUw0KPCaJrL2z6xUscM0e48R1ciLdU73D8bl1It39FOzme4 sww3DCwu6tf4RJaG1H7C7neLBzRcmdv9Z9Cw70EpF12nj13+O54d297D4beTH2Mq/TqnuCQ19y7y0 yrRh0佝d0b8AaAIAI+OMo6LyoKZw3crYr1Nvub/iHqewoMF/kVkJdJi6INf1aELH5Fpos9/0r2z4aS1qoN 1kSgyQfW36xRwp7+bGk4GpPN3EaqHoXLTn/xtXPr16LEYM6i7VIMicPzxx1VypFvB13L4xvm5QW nrNk/z0QhPcE8hsvyi67iNj38jywrzre55@auron
ssh-dss AAAAB3NzaC1kc3MAAACBAKDQhXz5VXy0hnh2Yf6xh2y1elRAJecKbyCa1Yrhn1XxSmD7qXG7 M2THU7fvdzUx0KozUaZc1cQvdc+D/1D/7tDq+7mQ9M1XqKZpY77L2m4nPBvJEpmE0Q4/F4NVMBXq+jk7RF0Mn921i+wsOnps tK6C7TYc1/1bN8nt3F/NFYqO9w9ZK+I+quasfnw1AAPAFQCAma1g3quBv9hJn5hUJikkFjQF8QAAAIAd ut687u0u4eC6AYGrtrNWX1117rveS6EG/6/1TX2s0xoxOJ+qjAl9q7IpV1qC9u9HRV7CphOfJgAeE7 7Mf1R1qB0kaCmNjM4VXzXiEoU1NYx+91TJn2b13GookI30PX0078Gf48/sQ8TFcy1YeUNqbgBf4/7V M60QguxTeAAAABoUCIpbMJBwoso4bK12qZwmmXzEL/k1nv1gBoK/XXsZ/nZj7D7GZflJSwigFPtc NUmtr+1KHz+R63n6e2xvE/7I/ZeLJ0RauJ+veCAW1021hYm7rse6sue08O3r4qoeMu8QHnCK1pfq1gFHK NtVwglW4LmzM7011NM9Mv5gJdwyregeorg@toomba.lanl.gov
```
Authentication Using Public Key:
Key Deposit at Remote Host

• OpenSSH:

```
lu65fok@badwlrz-clvwb01:~/.ssh> scp id_rsa.pub lxlogin1.lrz.de:
  100%  415     0.4KB/s   00:00
lu65fok@badwlrz-clvwb01:~/.ssh>

...  
...
lu65fok@lxa191:~> cat id_rsa.pub >> .ssh/authorized_keys
```

• Password:

```
lu65fok@lxa191:~> logout
```

• Connection to lxlogin1 closed.

```
lu65fok@badwlrz-clvwb01:~> ssh lxlogin1
Enter passphrase for key '/home/lu65fok/.ssh/id_rsa':
```

Parallel Programming of High Performance Systems | RRZE | 9.-13.3.2020
### Important Command Line Options

#### OpenSSH:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-l &lt;name&gt;</code></td>
<td>use <code>&lt;name&gt;</code> as username for login (default: local user)</td>
</tr>
<tr>
<td><code>-v</code></td>
<td>be verbose about connection – useful for debugging</td>
</tr>
<tr>
<td><code>-V</code></td>
<td>show version string</td>
</tr>
<tr>
<td><code>-f</code></td>
<td>fork to background after authentication/authorization</td>
</tr>
<tr>
<td><code>-q</code></td>
<td>no warnings (quiet)</td>
</tr>
<tr>
<td><code>-p &lt;port&gt;</code></td>
<td>only <code>ssh</code>: connect to <code>&lt;port&gt;</code> instead port 22 (default)</td>
</tr>
<tr>
<td><code>-P &lt;port&gt;</code></td>
<td>only <code>scp</code>: connect to <code>&lt;port&gt;</code> instead port 22 (default)</td>
</tr>
<tr>
<td><code>-r</code></td>
<td>only <code>scp</code>: copy recursively</td>
</tr>
<tr>
<td><code>-p</code></td>
<td>only <code>scp</code>: keep file attributes and timestamps</td>
</tr>
</tbody>
</table>
Important Command Line Options

- **OpenSSH options (see later for detailed description):**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-X/-x</td>
<td>Enable/disable X11 forwarding</td>
</tr>
<tr>
<td>-L <code>&lt;listen&gt;</code>:&lt;host&gt;:&lt;port&gt;</td>
<td>forward local port <code>&lt;listen&gt;</code> to remote host <code>&lt;host&gt;</code>, port <code>&lt;port&gt;</code></td>
</tr>
<tr>
<td>-R <code>&lt;listen&gt;</code>:&lt;host&gt;:&lt;port&gt;</td>
<td>forward remote port <code>&lt;listen&gt;</code> to local host <code>&lt;host&gt;</code>, port <code>&lt;port&gt;</code></td>
</tr>
<tr>
<td>-g</td>
<td>remote hosts can connect to forwarded ports</td>
</tr>
</tbody>
</table>
Daily Work With ssh

• **Single sign-on**
  • Input of passphrase on each login is inconvenient
  • Agent enables private key storage for duration of session
  • Input of passphrase only once per session!

• **User’s config file**
  • Specification of full ssh command line with full user/host name is inconvenient
  • Configuration file enables convenient shortcuts (defaults), so that host “nickname” is sufficient
  • Full hostname, username and other parameters are specified in the config file

• **Tunneling of X11 and other connections**
  • If configured, X11 forwarding (tunneling) is automatic
  • Redirection of a local port to some host/port in the remote host network (and vice versa) is possible
Daily Work With ssh: Single Sign-on

- Using the `ssh-agent`

```bash
eval `ssh-agent`
ssh-add
```
starts agent and sets

```
$SSH_AUTH_SOCK
$SSH_AGENT_PID
```

Key Management

use

```
ssh
scp
sftp
```
Daily Work With ssh: Single Sign-on

• Use `ssh-agent -c` if shell is csh-derived

• Useful: agent can be made the mother process of a window manager, e.g. in `/etc/X11/Xsession`:
  ```
  ssh-agent $HOME/.xsession
  
  This way, environment variables will be visible for all applications started in this session
  • This is already configured correctly in most current Linux distros

• Load key into agent:
  ```
  ssh-add [<Private Key File>]
  
  • Sensible defaults for key files in different variants
  • Loading of more than one key is possible

• List of loaded keys: `ssh-add -l`

• Delete all loaded keys: `ssh-add -D`
Daily Work With ssh:
Security Advice on Using Single Sign-on

• Important: set passphrase!
  • Without one, the private key is open for everybody who can get your user permissions
  • Empty passphrase might be required for technical reasons – do not use such keys for general logins
• If keys are loaded into agent: Secure terminal when absent (xlock etc.)!
  • Other systems could be compromised in spite of a non-empty passphrase
• Agent will not divulge passphrase
  • If the system is compromised, however, this is no obstacle for an attacker
Daily Work With ssh: User‘s config file

- Configuration of parameters on a per-host basis
  - username
  - full hostname, nickname
  - compression yes/no
  - ...
- OpenSSH: $HOME/.ssh/config
- Example:

```bash
Host *
  ForwardX11
Host eliza
  HostName eliza.rrze.uni-erlangen.de
  Compression no
  User unrz55
Host vpp300er
  HostName vpp300er.rrze.uni-erlangen.de
  Compression yes
  CompressionLevel 9
  User mpt312
Host rzcip*
  Compression yes
```
Daily Work With ssh: X11 Tunneling

- If allowed, X11 connections will be forwarded automatically from the remote host to the local host
  - X11 clients can be started remotely without setting the $DISPLAY variable (is set to `<remotehost>:<#>`)  
  - Encryption and decryption eats some performance  
    - Especially noticeable with GUI apps  
    - If direct X connection is preferred over tunneling, set $DISPLAY accordingly and do X authentication via xauth  
  - Caveat: X forwarding is not always enabled by default (use `-x` if required)  
    - cluster environments use no default X forwarding because of performance reasons (and problems with simultaneous file access)
- Compression can boost performance significantly, especially over slow connections!
  - However, GUIs over large-latency connections (e.g. DSL) are always pain → use terminal services instead (VNC, NX, RDP)
Free SSH Software

- **UNIX: OpenSSH** ([http://www.openssh.org/](http://www.openssh.org/))
  - Free client and server
  - Interoperability to commercial variants
  - All protocols, available by default on all Linux distros

- **Cygwin: OpenSSH**

- **Windows: puTTY**
  - Can be used without installation (suitable for memory stick)
  - Convenient GUI for configuration
  - Command line scp and sftp clients
  - Key format is not compatible to OpenSSH, but puTTY allows key exports to several formats (including OpenSSH)
  - Does not include X11 server (use Cygwin or Xming for that)
Environment Modules
Parallel Programming of High Performance Systems | 25.02.2019 | Hager – Bader – Weinberg
Controlling the programming environment

• UNIX/Linux systems
  • shell (usually remotely started)

• **System** settings:
  • limits (memory, stack, coresize etc.),
  • quotas (if defined by sys admin)
  • chroot (if defined)
  • user only has limited influence on settings

• **Execution control** settings:
  • environment variables
  • fully under control of user
“Classes” of environment settings

- **Database-like**
  - Example: PATH variable
  - Where does the shell look for executable programs?
  - Format: directory entries separated by colons (‘:’)

```
    echo $PATH
    /usr/bin:/usr/X11R6/bin:/client/bin
```

- pre-defined variables usually impact the run time behaviour of numerous programs

- **Simple settings**
  - Example:

```
    echo $OPT_TMP
    /gpfs/scratch/pr28fa/lu65fok/
```

- convenient abbreviation for heavily used values

- specific variables often needed by specific applications
- more rarely also critical for general run time behaviour e.g., $HOME + X11
How to administer environment settings:
Environment modules

• package written in Tcl
  • allows setting and unsetting of environment variables
  • allows adding and deleting entries from database type environment variables
  • does this in a shell-independent fashion (necessary information is stored in a module file)
  • allows versioning of software packages:
    1. multiple versions installed, which one to be used?
    2. separate module files for separate versions
    3. enforce that separate versions cannot be loaded at same time

• has nothing at all to do with Linux kernel modules
Loading and unloading of modules

• Example: want to use likwid tool

```bash
lu65fok@login12:~> likwid-topology
-bash: likwid-topology: command not found
lu65fok@login12:~>
lu65fok@login12:~> module load likwid
lu65fok@login12:~> type likwid-topology
likwid-topology is /lrz/sys/tools/likwid/likwid-4.0/bin/likwid-topology
lu65fok@login12:~>
lu65fok@login12:~> likwid-topology
```

```
CPU name: Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz
```

```
lu65fok@login12:~> module unload likwid
lu65fok@login12:~> type likwid-topology
-bash: type: likwid-topology: not found
lu65fok@login12:~>
```

$PATH was extended by this entry

Entry deleted from $PATH
But what does loading the module change?

lu65fok@login12:~> module show likwid/3.0
-----------------------------------------------
/lrz/sys/share/modules/files/tools/likwid/3.0:

module-whatis  Enable usage of LIKWID, LIKWID, Light weight performance tools
setenv LIKWID_BASE   /lrz/sys/tools/likwid/likwid-3.0
setenv LIKWID_LIBDIR /lrz/sys/tools/likwid/likwid-3.0/lib
setenv LIKWID_DOC    /lrz/sys/tools/likwid/likwid-3.0/man
setenv LIKWID_WWW    http://code.google.com/p/likwid/
setenv LIKWID_SRC    /lrz/sys/tools/likwid/likwid-3.0/src
setenv LIKWID_INSTALL_DOC /lrz/sys/tools/likwid/likwid-3.0/INSTALL
prepend-path PATH   /lrz/sys/tools/likwid/likwid-3.0/bin  :
prepend-path MANPATH /lrz/sys/tools/likwid/likwid-3.0/man  :
prepend-path LD_LIBRARY_PATH /lrz/sys/tools/likwid/likwid-3.0/lib  :
prepend-path CPATH    /lrz/sys/tools/likwid/likwid-3.0/include  :
prepend-path INCLUDE  /lrz/sys/tools/likwid/likwid-3.0/include  :
prepend-path LIB     /lrz/sys/tools/likwid/likwid-3.0/lib  :
setenv LIKWID_INC   -I/lrz/sys/tools/likwid/likwid-3.0/include
setenv LIKWID_LIB    -L/lrz/sys/tools/likwid/likwid-3.0/lib -llikwid likwidpin
setenv LIKWID_SHLIB  -L/lrz/sys/tools/likwid/likwid-3.0/lib -llikwid likwidpin
prereq mpi.intel mpi.ibm
-----------------------------------------------
lu65fok@login12:~>
And how do I know what modules there are?

```
lu65fok@mpp3-login8:~> module av
------------------ /lrz/sys/share/modules/files/applications -------------------
ansys/18.2(default)        gromacs/2016.3(default)      vasp/5.4(default)
cfx/18.2(default)          gespresso/6.1(default)      wannier90/1.2(default)
fluent/18.2(default)       R/3.3.2(default)            wannier90/2.0
fluent/19.0                schrodinger/2017-4(default)
gaussian/16.A.03(default)  schrodinger/2018-1
-------------------- /lrz/sys/share/modules/files/compilers -------------------
gcc/4.9         gcc/7                intel/17.0.5       pgi/16
gcc/5           intel/16.0           intel/18.0         pgi/17(default)
gcc/6(default)  intel/17.0(default)  java/1.8(default)
------------------- /lrz/sys/share/modules/files/environment -------------------
admin/1.0(default)  lrz/default  lrz/nodev tempdir/1.0(default)
-------------------- /lrz/sys/share/modules/files/graphics ---------------------
gnuplot/5.0(default)  plplot/5.11(default)  qt/5.8
gnuplot/5.2               qt/5.7(default)  tecplot/2017r3(default)
```
And how do I know what modules there are?

```
lu65fok@mpp3-login8:~> module av likwid
---------------------- /lrz/sys/share/modules/files/tools ----------------------
likwid/4.2(default)   likwid/4.3
```

```
lu65fok@mpp3-login8:~> module av -d likwid
---------------------- /lrz/sys/share/modules/files/tools ----------------------
likwid/4.2(default)
```

Available versions

```
module avail -d
```

module class

shows only default versions
Resolving conflicts

- Cannot have two different versions of Gaussian at the same time
  - configured in module file to prevent destructive interactions between packages with overlapping functionality or dependencies

```
$ module load gaussian
$ module load gaussian/09.A.02
WARNING: gaussian/09.A.02 cannot be loaded due to a conflict.
HINT: Might try "module unload gaussian" first.
$ module switch gaussian gaussian/09.A.02
```

the same as:

```
module unload gaussian
module load gaussian/03.B.01
```
Which modules are presently loaded?

```
lu65fok@mpp3-login8:~> module list
Currently Loaded Modulefiles:
  1) admin/1.0   3) intel/17.0   5) mpi.intel/2017   7) likwid/4.2
  2) tempdir/1.0 4) mkl/2017    6) lrz/default
```

• Notes:
  • an initial set of modules may be automatically loaded at login
Module stacks

- **Internal dependencies:**
  - detect which settings are needed depending on the existing environment
  - perform these settings or refuse to load module

- **Manual dependency resolution**
  - user must manually set up module stack according to documentation
  - failures unfortunately sometimes possible
  - start new shell if environment too corrupted

- **Automatic dependency resolution**
  - automatically load missing modules
  - only possible if dependency is unique
Further module commands

$ module help
• provides information about usage of module command

$ module [help|whatis] <name>
• provides information about module <name>

$ module apropos <string>
• will run whatis on every module containing <string> in its whatis section

$ module purge
• removes all loaded modules
• beware: on LRZ systems, this will impair your usage. At least the environment modules
  • admin
  • tempdir
should be reloaded after a purge
Caveats

• Manipulating the environment
  • setting database variables may conflict with settings performed by module system
  • typical error message: „... does not agree with XXX_modshare ...“
  • uses additional environment variables „_modshare“ to maintain reference counts
  • therapy: put own settings into self-written module files
  • see next slide for setup

• Using modules in batch jobs
  • requires user profiles to run
  • batch systems (SGE, PBS) provide no support or incomplete support
  • therapy: execute
    . /etc/profile
    or
    . /etc/profile.d/modules.sh
  • see example scripts on LRZ web server
Writing your own module files

• requires a subdirectory
  ~/.modules
• within this directory
  • init file with the same name as your shell (e.g., bash)
  • containing something like

```
module use $HOME/.modules/files
module load ...
```

• the “files” subdirectory then can contain self-written module files

• implementation of module files
  • look at / copy / modify LRZ module files for your own purposes

whatever you want in addition to LRZ defaults
LRZ-provided add-ons

- **ismoduleloaded:**
  - is argument module loaded?
    - YES/NO to stdout
    - 0/1 exit status
  - argument versioned module name: check exact version
  - otherwise: any version
  - `-s` command line switch: suppress output to stdout

- **defaultversion:**
  - return the default version of argument module on stdout.
  - exit status 1 if the module does not exist.

- **removemodules.[c]sh:**
  - completely remove all module settings from environment
  - source into running shell
  - all modules are unloaded
  - module alias unset
  - module-related variables unset.
  - will **cripple** environment → use with care
Spack Generated Modules

lu65fok@mpp3-login8:/lrz/sys/courses/PPHPS20> module list
Currently Loaded Modulefiles:
  1) admin/1.0     3) intel/19.0      5) mpi.intel/2019       7) lrz/default
  2) tempdir/1.0   4) mkl/2019     6) spack/release/19.2

lu65fok@mpp3-login8:/lrz/sys/courses/PPHPS20>
Spack Generated Modules

• **Spack Generated Modules**
  
  • With the installation of SuperMUC-NG, LRZ gradually changes its software deployment process from manually built packages to automated builds with Spack.
  
  • **Spack** is a tool to automatize and manage builds of software packages. Collections of packages built with Spack will be released on LRZ clusters periodically and are available through the environment module system. The latest release of packages will be listed at the bottom when using the 'module av' command.
  
  • Environment modules generated by Spack use the naming scheme
  
    • `<package name>/<full version number>[-<suffix1>][-<suffix2>][…]

  It is highly recommended to use the complete module name when using the spack generated modules. If you need a specific supplied version, you will have to specify the version number explicitly.

•
Spack Branches and Releases

• Spack Branches
  • **release**: stable version for long term support
  • **staging**: addon to current release or preview of a future release
  • **master**: development version of LRZ - accessible for test installations, may be subject to full rebuild

• LRZ will periodically release a new version of the software stack built with Spack and change the default version to the new release. Older versions will remain available by loading specific modules, e.g.
  • module switch spack/release/18.2

• Sometimes we will provide new software prior to a new Spack software-stack release in a staging branch. To access such versions, load the staging module
  • module switch spack/staging[/<version>]
Documentation

- Web
  
  https://doku.lrz.de/display/PUBLIC/Environment+Modules

- UNIX manual
  - man module
  - man modulefile
Managing terminal sessions with “screen”

Overview

• What is “screen”?
  • “screen” is a **text-based window manager** that can handle multiple shell sessions from a single session
  • Sessions can be given names for easy access
  • User can detach from the screen session anytime and re-attach later
  • Typical usage model:

```
ssh screenhost
$ screen -r
$ ssh abc
$ ssh xyz
$ mutt
$ emacs
```
Starting, attaching and detaching screen

• Starting a new screen session from a normal shell

   $ screen [command [args]]

• Attaching to a running but unattached screen session

   $ screen -r [pid.TTY.host]

• Attaching to a running but attached screen session

   $ screen -r [sessionID.TTY.host] -d

• Detaching from a screen session: Type  Ctrl-A d (^A d)
• Killing all terminal windows and terminating session: ^A \
Important command line options

• List my running screen sessions
  
  $ screen -ls
  There is a screen on:
  7462 pts-25 grid (Detached)
  1 Socket in /var/run/uscreens/S-unrz55.

• Wipe a screen session (delete every trace of it)

  $ screen -wipe [pid.TTY.host]

• Attach to a non-detached screen session (multi display mode)

  $ screen -x

• Attach to a specific screen window by number or name

  $ screen -p <number>|<name>
Working within screen

• After startup:
  • the screen session contains a single window (#0)
  • the name of the window is the name of the shell or the binary that was started
• Screen emulates a VT100 terminal
• Screen commands are initiated with ^A
• Important commands:
  • New window ^A c (or ^A ^c)
  • Rename window ^A A
  • Toggle to previous window ^A ^A
  • Jump to named window ^A ’<name>
  • Send ^A to shell ^A a
  • Detach session ^A d
  • List of commands ^A ?
More screen commands

- Split window  \(^A\) S
- Close region  \(^A\) X
- Change focus  \(^A\) \(<\text{tab}>\)
- Window list    \(^A\) ”

- Copy mode  \(^A\) [  
- In copy mode you can move around and edit like in vi. \(<\text{space}>\) marks the start and end of a region to copy
- Paste buffer  \(^A\) ]

- Identify window  \(^A\) N
- Kill window    \(^A\) k

- Next window    \(^A\) n
- Previous window \(^A\) p
- Switch to win 0-9 \(^A\) 0-9

- Hardcopy window (to hardcopy.#) \(^A\) h
- Toggle window log (to screenlog.#) \(^A\) H

- Refresh window  \(^A\) I
- Resize to fit    \(^A\) F
- XON/XOFF       \(^A\) s/q
- Command mode    \(^A\) :
screen customization

- At startup, screen reads commands from ~/.screenrc
- Can also input commands after ^A:

  Selection of commands:
  - Kill all regions but the current one only
  - Set a password for attaching password [crypt]
  - Disable password checking password none
  - Create new window screen [n] [cmd [args]]
  - Scrollback buffer for new windows defscrollback <num>
  - Set env variable for new windows setenv <var> <string>
  - Execute cmds from file source <file>
  - Set window caption display caption always|splitonly
A neat example of screen customization

• How do we use `ssh-agent` inside screen windows?

`~/.screenrc:`

```bash
setenv SSH_AUTH_SOCK $HOME/.screen-ssh-agent
screen 100 ssh-agent -a $SSH_AUTH_SOCK $SHELL
```

• This opens a window (#100) and runs ssh-agent there
• All subsequent shells in windows get `$SSH_AUTH_SOCK` set, and ssh knows where to ask for the key 😊

• Type `ssh-add` in any window to enter the passphrase

• Use option `-t <seconds>` to ssh-add to limit lifetime identity or employ a password for screen (see above)
Virtual Network Computing VNC
Efficient remote GUI access

- Using X11 based GUIs over the WAN:
  - bandwidth problem (becoming less relevant)
  - latency problem (staying relevant \(\leftarrow\) many small packages transferred)
- Idea for solution:
  - pack up complete screen and transfer a (possibly reduced-resolution) version of that over the network
  - VNC implements this using a portable approach
- not limited to X11

- What is VNC?
  - client-server – server on the system whose screen is desired, client connects from remote desktop
- Some variants:
  - **Tightvnc** – GPL version, can run on Windows and Linux. Java version of client is available. Compression capable.
  - **Xvnc** – comes with X11
  - **TurboVNC** – used to perform remote visualization; multi-pipe and OpenGL support. Compression capable.
Step 1: Starting the server

- Will discuss simplest usage pattern here:
  - using tightvnc with server running on Linux/UNIX
  - on the remote machine, type
    \[
    \text{vncserver \textit{-geometry 1024x768 \textit{-depth 16 \textit{-name MYSERVER}}}
    \]
  - this will start a detached X11 server (a "virtual screen")
  - including window manager – may need to set up via \texttt{xsession} file on server
  - name of virtual screen is printed to standard output:
    \[
    \text{new 'MYSERVER' desktop is lx64e14:4}
    \]
  (might also see some warnings about already taken displays \rightarrow ignore)
Security

- Various security levels available for connecting to the server:
  - simplest model: set a password (on the server side)

```
vncpasswd
Using password file /home/asdf/.vnc/passwd
Password: <enter, not echoed>
Verify: <repeat same>
Would you like to enter a view-only password (y/n)? n
```

- provides 64 bit DES encryption (not very strong)
- but can reset before each session
- other VNC variants support plugins with stronger encryption schemes (128 Bit RC4 or AES, or even Active Directory)
Step 2: Starting the client

**Scenario 1:**
- client is a Linux desktop
- direct network connection to server is possible
- type in the desktop name e.g., `lx64e14:4`
- then the password:
- finally, the desktop should appear as a separate window
- operate with mouse inside that window, start programs etc.

**Notes:**
- can connect multiple times to the desktop ("shared desktop")
- might want to provide view-only password
- can reconnect if only viewer window is removed
- exiting the window manager or the controlling terminal inside the viewer takes down the session
- recommend using a light-weight window manager
Problems connecting?

- **Server may require port forwarding**
  - local and remote hostnames are different
  - load balancer etc.

- **Scenario 2: forwarding**
  - port used is typically 5900 + (display number)
  - for the previous example: log in from the client via

```bash
ssh -L 5900:localhost:5904 lx64ia3
```

- then try to connect again with vncviewer
  - specify `localhost:0` as host/port
Cleaning up

• Clean shutdown with

  \texttt{vncserver -kill :display}

• removes temporary files
• in our example: display \rightarrow 4