Exploring Linux Part 1 - Linux Fundamentals

Prerequisites

At this point:

1. your master node should be cabled;
2. your master node should have a fresh install of CentOS;
3. your master node should have a special account called `admin`;
4. your master node should be configured to live on the 10.0.52.xxx;

Part 1: Access & adding users

We added an `admin` user to the system. We did this because, by default, `ssh` will not allow root to login directly; however, we want to setup individual users for the master.

Let's start by logging into the master. Open at terminal on your Mac laptop while connected to the wired network. Now login using: `ssh admin@10.0.52.xxx`. Enter the password when prompted. You should be given a bash prompt:

```
lowell$ ssh admin@10.0.52.xxx
Last login: Sun Jun  2 18:53:37 2019 from 10.0.52.52
[admin@localhost ~]$
```

We will need to become `root` to add our users. It should have been configured at install time that the `admin` user can have full access to the `sudo` command. `sudo` allows an ordinary user to use their own password to perform actions as another user. We can check what `sudo` privileges we've been given with (using the password of `admin`, not `root`):

```
[admin@localhost ~]$ sudo -l
[sudo] password for admin:
Matching Defaults entries for admin on te-master:
    !visiblepw, always_set_home, match_group_by_gid,
always_query_group_plugin, env_reset, env_keep="COLORS DISPLAY HOSTNAME
HISTSIZE
    KDEDIR LS_COLORS", env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG
LC_ADDRESS LC_CTYPE", env_keep+="LC_COLLATE LC_IDENTIFICATION
    LC_MEASUREMENT LC_MESSAGES", env_keep+="LC_MONETARY LC_NAME LC_NUMERIC
LC_PAPER LC_TELEPHONE", env_keep+="LC_TIME LC_ALL LANGUAGE
    LINGUAS _XKB_CHARSET XAUTHORITY",
secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

User admin may run the following commands on te-master:
    (ALL) ALL
```

The last line of this output tells us that `admin` can run `ALL` commands as `ALL` users using `sudo`.

We can now use this account to run a command as root:

```
[admin@localhost ~]$ sudo whoami
root
```

We can launch a new interactive session as root with the the `-i` option:

```
[admin@localhost ~]$ sudo -i
[root@localhost ~]#
```

Let's get things started by setting the hostname of our master. This guide will use the hostname `te-master`. The `systemd` way of doing this is:

```
# hostnamectl set-hostname te-master
```

Now that we are root, we have the power to add new users for each member of the group. We can do this with the `useradd` command, then set a password with the `passwd` command. Let's suppose our first user login is `jane` (just add one user for now).

```
# useradd jane
# passwd jane
Changing password for user jane.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

Let's look at what this created. These are the files that contain user information. Let's look at what we created (if you're unfamiliar, `grep` can find and print lines in a file that have a matching string):

```
[root@te-master ~]# grep jane /etc/passwd
jane:x:1003:1003::/home/jane:/bin/bash
[root@te-master ~]# grep jane /etc/group
jane:x:1003:
[root@te-master ~]# grep jane /etc/shadow
jane:$6$1i1DFQ0F$/6w6OsZjegp.fWspIWRUHzsZMalvh6NUJ2UKnKJqFhiqBACcvM87bawsF
RL0dj9bgU1kZV8SfDniH9le.6upT/:18051:0:99999:7:::
```

`/etc/passwd` contains the user information; `/etc/group` contains group information (`jane`'s default group is named `jane` as well); `/etc/shadow` contains account age and password hash information.

We want `jane` to have `sudo` power. Let's see if she does:

```
# sudo -l -U jane
User jane is not allowed to run sudo on te-master.
```

Nope. `jane` doesn't have `sudo`. Why is this? We can inspect the `sudo` configuration. We could just open the file: `/etc/sudoers`, but there's a better way. Instead, run `sudoedit`. `sudoedit` will open whatever editor is specified by the `EDITOR` environment variable. By default, this is `vi` (see the supplied `vi` cheatsheet). It has the advantage that it will check the syntax of the file before saving it. This is a ***very*** good idea. A mangled `/etc/sudoers` file could block access to everyone.

```
# sudoedit /etc/sudoers
...
    105 ## Allows people in group wheel to run all commands
    106 %wheel  ALL=(ALL)       ALL
    107
...
```

Somewhere around line `106` we see this specification. It means that members of the group named `wheel` can use all `sudo` commands as all users. So, we just need to add `jane` to the `wheel` group. Let's do this by editing the `/etc/group` file. Like `sudoedit`, there's a safe way to do this:

```
# vigr
...
wheel:x:10:lowell,jane
...
```

We add `jane` to the line starting with `wheel`, after the last colon. This is a coma separated list.

Let's take a look at `jane` now:

```
# id jane
uid=1003(jane) gid=1003(jane) groups=1003(jane),10(wheel)
```

Great! `jane` is now in the wheel group. Let's check sudo again:

```
# sudo -l -U jane
Matching Defaults entries for jane on te-master:
    !visiblepw, always_set_home, match_group_by_gid,
always_query_group_plugin, env_reset, env_keep="COLORS DISPLAY HOSTNAME
HISTSIZE
    KDEDIR LS_COLORS", env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG
LC_ADDRESS LC_CTYPE", env_keep+="LC_COLLATE LC_IDENTIFICATION
    LC_MEASUREMENT LC_MESSAGES", env_keep+="LC_MONETARY LC_NAME LC_NUMERIC
LC_PAPER LC_TELEPHONE", env_keep+="LC_TIME LC_ALL LANGUAGE
```

```
    LINGUAS _XKB_CHARSET XAUTHORITY",
secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

User jane may run the following commands on te-master:
    (ALL) ALL
```

Ok, now that jane has access, we let's add the other users. Now that we know aboug `wheel`, we can do that in one step.

```
# useradd -G wheel <member_login>
# passwd <member_login>
```

That's it! We should now have all members added as users. Each member should now login as their own user and try `sudo`:

```
$ ssh jane@10.0.52.xxx
[jane@te-master] $ sudo -i
[root@te-master] #
```

Part 2: Yum, RPM & repos

For the moment we will need to choose one person to perform commands on our master. We will work up to a more collaborative way to do things, but we first need to be able to install some new software.

Before we do anything else, we need to setup some custom repositories. We will be using local mirrors of the repositories. A file is available that contains all of this information. We can grab it like this:

```
[root@te-master ~]# curl -O http://10.0.52.146/repo/yum.repos.d.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
100  2773  100  2773    0     0   502k      0 --:--:-- --:--:-- --:--:--
541k
[root@te-master ~]# ls
yum.repos.d.tar.gz
```

Now we can extract it:

```
[root@te-master ~]# tar zxvf yum.repos.d.tar.gz
yum.repos.d/
yum.repos.d/CentOS-CR.repo
yum.repos.d/CentOS-Debuginfo.repo
yum.repos.d/CentOS-Media.repo
```

```
yum.repos.d/CentOS-Sources.repo
yum.repos.d/CentOS-Vault.repo
yum.repos.d/CentOS-fasttrack.repo
yum.repos.d/CentOS-Base.repo
yum.repos.d/CentOS-Base.repo.rpmnew
yum.repos.d/epel.repo
yum.repos.d/OpenHPC.repo
yum.repos.d/OpenHPC-updates.repo
yum.repos.d/mlnx.repo
```

Repository specifications live in `/etc/yum.repos.d` (take a look!). We can copy these files into place to use our local repos:

```
# /bin/cp -avf yum.repos.d/* /etc/yum.repos.d/
‘yum.repos.d/CentOS-Base.repo’ -> ‘/etc/yum.repos.d/CentOS-Base.repo’
‘yum.repos.d/CentOS-Base.repo.rpmnew’ -> ‘/etc/yum.repos.d/CentOS-
Base.repo.rpmnew’
‘yum.repos.d/CentOS-CR.repo’ -> ‘/etc/yum.repos.d/CentOS-CR.repo’
‘yum.repos.d/CentOS-Debuginfo.repo’ -> ‘/etc/yum.repos.d/CentOS-
Debuginfo.repo’
‘yum.repos.d/CentOS-fasttrack.repo’ -> ‘/etc/yum.repos.d/CentOS-
fasttrack.repo’
‘yum.repos.d/CentOS-Media.repo’ -> ‘/etc/yum.repos.d/CentOS-Media.repo’
‘yum.repos.d/CentOS-Sources.repo’ -> ‘/etc/yum.repos.d/CentOS-
Sources.repo’
‘yum.repos.d/CentOS-Vault.repo’ -> ‘/etc/yum.repos.d/CentOS-Vault.repo’
‘yum.repos.d/epel.repo’ -> ‘/etc/yum.repos.d/epel.repo’
‘yum.repos.d/mlnx.repo’ -> ‘/etc/yum.repos.d/mlnx.repo’
‘yum.repos.d/OpenHPC.repo’ -> ‘/etc/yum.repos.d/OpenHPC.repo’
‘yum.repos.d/OpenHPC-updates.repo’ -> ‘/etc/yum.repos.d/OpenHPC-
updates.repo’
```

Note: we used `/bin/cp` instad of `cp` to avoid having to verify that we want to overwrite files.

Let's test this out and install a useful package. `vim` is the "improved" version of `vi` and has useful features like syntax highlighting. Let's install it:

```
# yum -y install vim
...
Installed:
  vim-enhanced.x86_64 2:7.4.160-5.el7

Dependency Installed:
  ...
  perl-threads-shared.x86_64 0:1.43-6.el7          vim-common.x86_64
2:7.4.160-5.el7         vim-filesystem.x86_64 2:7.4.160-5.el7

Complete!
```

Now let's look at one of those repo files: `vim /etc/yum.repos.d/CentOS-base.repo`. You should see some nice syntax highlighting.

Here's another one that will be useful later: `yum -y install telnet`

Let's see what we installed. We can list the files in an installed package with the `rpm` command:

```
# rpm -ql telnet
/usr/bin/telnet
/usr/share/doc/telnet-0.17
/usr/share/doc/telnet-0.17/README
/usr/share/man/man1/telnet.1.gz
```

We can list all of the RPMs installed on the system with:

```
# rpm -qa
...
audit-2.8.4-4.el7.x86_64
kernel-3.10.0-957.12.2.el7.x86_64
irqbalance-1.0.7-11.el7.x86_64
aic94xx-firmware-30-6.el7.noarch
microcode_ctl-2.1-47.2.el7_6.x86_64
parted-3.1-29.el7.x86_64
kernel-tools-3.10.0-957.12.2.el7.x86_64
iprutils-2.4.16.1-1.el7.x86_64
sudo-1.8.23-3.el7.x86_64
```

We can do that with yum too:

```
# yum list installed
...
xz.x86_64                       5.2.2-1.el7                @base
xz-libs.x86_64                  5.2.2-1.el7
@base/$releasever
yum.noarch                      3.4.3-161.el7.centos
@base/$releasever
yum-metadata-parser.x86_64      1.1.4-10.el7
@base/$releasever
yum-plugin-fastestmirror.noarch 1.1.31-50.el7
@base/$releasever
zlib.x86_64                     1.2.7-18.el7
@base/$releasever
```

We can look at other lists like `yum list available` to see available packages.

If we needed to update software, we could use:

```
# yum update
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
No packages marked for update
```

...but we may not currently have any updates.

Suppose we want to know what package provided us the `/bin/ls` command:

```
# rpm -q --whatprovides /bin/ls
coreutils-8.22-23.el7.x86_64
```

But this only works for things that are installed. With yum, we can find out what package would provide the `fortune` command even if it's not installed. We can run:

```
# yum provides '*bin/fortune'
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
epel/filelists_db
|  16 MB  00:00:00
fortune-mod-1.99.1-17.el7.x86_64 : A program which will display a fortune
Repo        : epel
Matched from:
Filename    : /usr/bin/fortune
```

Part 3: Sharing a session with `tmux`

Let's install a tool that we will be using to collaborate:

```
yum -y install tmux
...
```

Now, one member of the team should run the `tmux` command as root.

The screen should now have a green bar at the bottom with some text in it. `tmux` is a handy utility that allows you to do quite a few useful things. Most `tmux` commands are of the form `<Ctrl-b>-<command>`.

1. You can create a new terminal in one login `<Ctrl-b-c>`. To switch to the next window, `<Ctrl-b-n>`, and the previous `<Ctrl-b-p>`.
2. You an split your screen into two terminals: `<Ctrl-b-">`. Switch to the top screen `<Ctrl-b-UP>`.

There's a lot more you can do with `tmux`. See the included cheatsheet.

There's one specific thing we want to do with screen though. The other two users should login from their systems, then become root. Each can then run:

```
# tmux at
```

This will "attach" everyone to the same `tmux` session so that what one person types, the others see.

Take a little time to play around with `tmux` using the cheatsheet.

Part 4: files and permissions

It might be useful to have a directory that all three uses can work together in without using root. Let's set that up.

First, to share files, there will need to be a common group. Your group can be named anything you like. We will use `teachers`. To create the group: `groupadd teachers`

We can add `jane` to `teachers` with the `usermod` command. If we want to get fancy, though, we can add all of the users to the group in one line. Let's say the other two users are `francis` and `lucy`. The `bash` shell provides the ability to write `for` loops for this kind of thing:

```
# for u in jane francis lucy; do usermod -a -G teachers $u; id $u; done
uid=1004(jane) gid=1005(jane) groups=1005(jane),1002(teachers)
uid=1002(francis) gid=1003(francis) groups=1003(francis),1002(teachers)
uid=1003(lucy) gid=1004(lucy) groups=1004(lucy),1002(teachers)
```

This added the users to the group, then showed their membership to verify.

We need a place to put our shared directory. Let's put it at `/home/share/teachers`. The directory `/home/share` doesn't exist. We can make both `/home/share` and `/home/share/teachers` in one command:

```
mkdir -p /home/share/teachers
```

Let's check the permissions of our `teachers` directory:

```
# ls -ld /home/share/teachers
drwxr-xr-x 2 root root 6 Jun  3 22:08 /home/share/teachers
```

This isn't quite what we want. We want anyone in `teachers` to be able to write here. We also don't want anyone else to be able to look here. We need to set the group of the directory. Let's also make `jane` the owner:

```
# chown jane:teachers /home/share/teachers
[root@te-master ~]# ls -ld /home/share/teachers
drwxr-xr-x 2 jane teachers 6 Jun  3 22:08 /home/share/teachers
```

Better, but we want the group to be able to write:

```
# chmod 775 /home/share/teachers
[root@te-master ~]# ls -ld /home/share/teachers
drwxrwxr-x 2 jane teachers 6 Jun  3 22:08 /home/share/teachers
```

But we also don't want anyone else to have access:

```
# chmod o-rxw /home/share/teachers
[root@te-master ~]# ls -ld /home/share/teachers
drwxrwx--- 2 jane teachers 6 Jun  3 22:08 /home/share/teachers
```

As your user (not root), create a file in the directory. We can create an empty file with the `touch` command (on an existing file, this would update the modify time):

```
[jane@te-master ~]$ cd /home/share/teachers
[jane@te-master teachers]$ touch afile
[jane@te-master teachers]$ ls -l
total 0
-rw-rw-r-- 1 jane jane 0 Jun  3 22:15 afile
```

Once each user has done this, how would we find out which one was created last?

```
[jane@te-master teachers]$ ls -ltr
total 0
-rw-rw-r-- 1 jane    jane    0 Jun  3 22:15 afile
-rw-rw-r-- 1 francis francis 0 Jun  3 22:16 bfile
-rw-rw-r-- 1 lucy    lucy    0 Jun  3 22:17 cfile
```

The `-t` option sorts by time. The `-r` option gives reverse order. This puts the most recently modified at the bottom.

This may not be quite what we want though. We want files made here to retain the `teachers` group permissions. We can set the `setgid` bit on the directory for this.

```
[jane@te-master teachers]$ chmod g+s /home/share/teachers/
[jane@te-master teachers]$ ls -ld /home/share/teachers
drwxrws--- 2 jane teachers 45 Jun  3 22:21 /home/share/teachers
[jane@te-master teachers]$ touch test
[jane@te-master teachers]$ ls -l
total 0
-rw-rw-r-- 1 jane    jane     0 Jun  3 22:15 afile
-rw-rw-r-- 1 francis francis  0 Jun  3 22:16 bfile
```

```
-rw-rw-r-- 1 lucy    lucy     0 Jun  3 22:17 cfile
-rw-rw-r-- 1 jane    teachers 0 Jun  3 22:23 test
```

That's better!

Part 5: Shell tricks

We have been using `bash` as our shell, but we haven't explored its full potential very well.

Here are some handy things we can do:

Redirection and pipes

We can redirect `stdout` (where normal output goes) to a file:

Get all users with `UID > 1000` and put them in the file `high-uid.txt`:

```
[root@localhost ~]# awk -F: '$3>1000{print}' /etc/passwd
jane:x:1001:1001::/home/jane:/bin/bash
francis:x:1002:1002::/home/francis:/bin/bash
lucy:x:1003:1003::/home/lucy:/bin/bash
[root@localhost ~]# awk -F: '$3>1000{print}' /etc/passwd > high-uid.txt
[root@localhost ~]# cat high-uid.txt
jane:x:1001:1001::/home/jane:/bin/bash
francis:x:1002:1002::/home/francis:/bin/bash
lucy:x:1003:1003::/home/lucy:/bin/bash
```

Here we used `awk`. Awk is a powerful parser that specializes in any thing that can be thought of as a grid of columns and rows. `awk` is actually a fully featured programming language in many ways.

Error messages are typically printed on `stderr` not `stdout`. We can redirect `stderr` to a file with `<cmd> 2> <file>`.

```
[root@localhost ~]# ls blah > out 2> err
[root@localhost ~]# cat out
[root@localhost ~]# cat err
ls: cannot access blah: No such file or directory
```

Or, we could combine them into one file:

```
[root@localhost ~]# ls blah > out 2>&1
[root@localhost ~]# cat out
ls: cannot access blah: No such file or directory
```

You can think of this as redirecting `stderr` into `stdout`. Note that this redirection happens after the redirection out.

Lots of commands can operate on data streamed to their standard input. We can pipe the output of one command to another. Suppose we want to see only the non-comment and non-empty lines in `/etc/sudoers`:

```
[root@localhost ~]# grep -vE '^#|^$' /etc/sudoers
Defaults   !visiblepw
Defaults    always_set_home
Defaults    match_group_by_gid
Defaults    always_query_group_plugin
Defaults    env_reset
Defaults    env_keep =  "COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR
LS_COLORS"
Defaults    env_keep += "MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS
LC_CTYPE"
Defaults    env_keep += "LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT
LC_MESSAGES"
Defaults    env_keep += "LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER
LC_TELEPHONE"
Defaults    env_keep += "LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET
XAUTHORITY"
Defaults    secure_path = /sbin:/bin:/usr/sbin:/usr/bin
root    ALL=(ALL)       ALL
%wheel  ALL=(ALL)       ALL
```

But, what if we just want to know how many lines there are?

```
[root@localhost ~]# grep -vE '^#|^$' /etc/sudoers | wc -l
13
```

Environment variables

Environment variables can control or track many things about your login session. Some of the special ones control `bash` itself, while others might be used for specific programs.

One helpful one is `$HOME`. This is your home directory. You can use it like this;

```
[jane@localhost ~]$ ls -ld $HOME
drwx------. 2 jane jane 83 Jun  4 05:17 /home/jane
```

Suppose you put some executables in `$HOME/scripts`, and you want them to be found automatically. You need to update your path:

```
[jane@localhost ~]$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/jane/.local/b
in:/home/jane/bin
[jane@localhost ~]$ export PATH=$PATH:$HOME/scripts
[jane@localhost ~]$ echo $PATH
```

```
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/jane/.local/b
in:/home/jane/bin:/home/jane/scripts
```

You can change your prompt by setting the `PS1` variable.

```
[jane@localhost ~]$ export PS1="[nifty] $PS1"
[nifty] [jane@localhost ~]$
```

If you want this to persist, add it as a line to your `$HOME/.bash_profile`. To reload that in the current session, you can do that in one of two equivalent ways:

```
$ source $HOME/.bash_profile
$ . $HOME/.bash_profile
```

Working with history

What were the last 3 commands we ran?

```
# history | tail -n 3
  168  grep -vE '^#|^$' /etc/sudoers
  169  grep -vE '^#|^$' /etc/sudoers | wc -l
  170  history | tail -n 3
```

What was that crazy `awk` command again?

```
[root@localhost ~]# history | grep awk
  178  awk -F: '$3>1000{print}' /etc/passwd
  179  awk -F: '$3>1000{print}' /etc/passwd > high-uid.txt
  171  history | grep awk
```

I just want to re-run that.

```
[root@localhost ~]# !178
awk -F: '$3>1000{print}' /etc/passwd
jane:x:1001:1001::/home/jane:/bin/bash
francis:x:1002:1002::/home/francis:/bin/bash
lucy:x:1003:1003::/home/lucy:/bin/bash
```

If you want to search up through your history, you can use `<Ctrl-r>`, then start typing your search. When you find the match you want, hit enter (or `<Ctrl-u>` if you want to edit it).

```
(reverse-i-search)`rpm': rpm -q --whatprovides /bin/ls
```

When all else fails, `man`

So some one shows you this command that recursively prints the line count of every file less than a day old in a directory:

```
[root@localhost ~]# find . -type f -ctime -1 -exec wc -l {} \;
183 ./.bash_history
11 ./.lesshst
10 ./yum.repos.d.tar.gz
28 ./yum.repos.d/CentOS-CR.repo
21 ./yum.repos.d/CentOS-Debuginfo.repo
22 ./yum.repos.d/CentOS-Media.repo
42 ./yum.repos.d/CentOS-Sources.repo
...
```

Wait, how does `-exec` work in find again? Find out!

```
# man find
```

What man pages are availabe that mention `vim`?

```
# man -k systemd
...(lots and lots and lots and lots)
systemd.mount (5)    - Mount unit configuration
systemd.path (5)     - Path unit configuration
systemd.preset (5)   - Service enablement presets
systemd.resource-control (5) - Resource control unit settings
systemd.scope (5)    - Scope unit configuration
systemd.service (5)  - Service unit configuration
systemd.slice (5)    - Slice unit configuration
systemd.snapshot (5) - Snapshot unit configuration
systemd.socket (5)   - Socket unit configuration
systemd.special (7)  - Special systemd units
systemd.swap (5)     - Swap unit configuration
systemd.target (5)   - Target unit configuration
systemd.time (7)     - Time and date specifications
systemd.timer (5)    - Timer unit configuration
systemd.unit (5)     - Unit configuration
```

Load a specific man page number:

```
man 5 systemd.target
```