CentOS 8 minimal installation on RPi 3+ ??

Installation

CentOS tuto for installation on RaspberryPi [1] [2].

Just for memory: Fedora link for arm architecture.

SSH

To know ip on local network:

```
arp -a
```

Hosts

On the client machine add the server ip address in `/etc/hosts`.

Add to known hosts

```
ssh-keyscan mycentos8 >> ~/.ssh/known_hosts
```

After, edit the file to keep the *ecdsa* line.

Repo

Install EPEL Repo

```
dnf info epel-release
dnf install epel-release
```

Adduser

Adduser

```
adduser <username>
passwd <username>
```

Adding user to group wheel:

```
gpasswd -a <username> wheel
```

Manage users and groups.

Change shell for zsh

```
usermod --shell /bin/zsh <username>
```

Oh-my-zsh

Install Oh-my-zsh:

```
sh -c "$(curl -fsSL
https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"
```

Copy theme '*amuse*' to the same folder for '*01amuse*'

```
vim ~/.zshrc
```

```
ZSH_THEME='01amuse'
```

Add information to the ligne `PROMPT=` without delete previous options (more tips here):

```
vim ~/.oh-my-zsh/themes/01amuse.zsh-theme
```

```
PROMPT='[...keep prev...]%{$fg[magenta]%}%n%{$reset_color%} at
%{$fg[yellow]%}%m%{$reset_color%}'
```

Use plugins:

```
plugins=(git dnf history tmux)
```

- *dnf*

Alias	Command	Description
dnfl	`dnf list`	List packages
dnfli	`dnf list installed`	List installed packages
dnfgl	`dnf grouplist`	List package groups
dnfmc	`dnf makecache`	Generate metadata cache
dnfp	`dnf info`	Show package information
dnfs	`dnf search`	Search package
dnfu	`sudo dnf upgrade`	Upgrade package
dnfi	`sudo dnf install`	Install package
dnfgi	`sudo dnf groupinstall`	Install package group
dnfr	`sudo dnf remove`	Remove package
dnfgr	`sudo dnf groupremove`	Remove package group
dnfc	`sudo dnf clean all`	Clean cache

- *git*
- *history*

Alias	Command	Description
`h`	`history`	Prints your command history
`hs`	`history | grep`	Use grep to search your command history
`hsi`	`history | grep -i`	Use grep to do a case-insensitive search of your command history

- *vscode*
- *npm*
- *tmux*

Alias	Command	Description

`ta`	tmux attach -t	Attach new tmux session to already running named session
`tad`	tmux attach -d -t	Detach named tmux session
`ts`	tmux new-session -s	Create a new named tmux session
`tl`	tmux list-sessions	Displays a list of running tmux sessions
`tksv`	tmux kill-server	Terminate all running tmux sessions
`tkss`	tmux kill-session -t	Terminate named running tmux session
`tmux`	`_zsh_tmux_plugin_run`	Start a new tmux session

- *nmap*

Alias	Description
`nmap_open_ports`	scan for open ports on target
`nmap_list_interfaces`	list all network interfaces on host where the command runs
`nmap_slow`	slow scan that avoids to spam the targets logs
`nmap_fin`	scan to see if hosts are up with TCP FIN scan
`nmap_full`	aggressive full scan that scans all ports, tries to determine OS and service versions
`nmap_check_for_firewall`	TCP ACK scan to check for firewall existence
`nmap_ping_through_firewall`	host discovery with SYN and ACK probes instead of just pings to avoid firewall restrictions
`nmap_fast`	fast scan of the top 300 popular ports
`nmap_detect_versions`	detects versions of services and OS, runs on all ports
`nmap_check_for_vulns`	uses vulscan script to check target services for vulnerabilities
`nmap_full_udp`	same as full but via UDP
`nmap_traceroute`	try to traceroute using the most common ports
`nmap_full_with_scripts`	same as nmap_full but also runs all the scripts
`nmap_web_safe_osscan`	little "safer" scan for OS version as connecting to only HTTP and HTTPS ports doesn't look so attacking
`nmap_ping_scan`	ICMP scan for active hosts

Note : *elements in zsh arrays are separated by* ***spaces****.*
Do not use commas.

Execute sudo without Password

Link here.
You can configure sudo to never ask for your password.

Open a Terminal window and type:

```
sudo visudo
```

In the bottom of the file, add the following line:

```
$USER ALL=(ALL) NOPASSWD: ALL
```

Where $USER is your username on your system.

Or, in my case, uncomment :

```
## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)       ALL
```

SSH keys

First create a ssh rsa key on computer:

```
ssh-keygen -t rsa
```

Then :

```
ssh myuser@server.com mkdir .sshscp ~/.ssh/id_rsa.pub myuser@server.com:.ssh/authorized_keys
```

GitHub

Use SSH keys

First, check if existing SSH keys are present:

```
ls -al ~/.ssh
```

Generate key pair:

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

Start the SSH agent in background:

```
eval "$(ssh-agent -s)"
```

Add the SSH private key to the ssh-agent:

```
ssh-add ~/.ssh/id_rsa
```

Copy the **public** key to the clipboard and paste it in Github.

To test the connection:

```
ssh -T git@github.com
```

Personnalize motd

Edit motd file:

```
vim /etc/motd
```

For ASCII check this site.

Or, much better, create `motd.sh` in `/etc/profile.d/` with:

```
#!/bin/sh
#


printf "\n"
printf "    ╔╦╥ ┬╔╗ ┌┐ ┌┐┌╔╦ ╔╗╔╗\n"
printf "    ┃┃┃└┘┃  ├┤ ┃┃┃ ┃ ╠ ╠╚╗\n"
printf "    ┴ ┴ ┴ ╚╝└┘┘└┘ ┴ ╚╝╚╝\n"
printf "       ╔╗┬┌┐┬┌─┌┐ ┌┐┬\n"
printf "       ╠╣┃├┤┃├┴┐├┤ │\n"
printf "       ╩ ╩┴└┘┴ ┴└┘└┘\n"


printf "\n"
printf "\t- %s\n\t- Kernel %s\n" "$(cat /etc/redhat-release)" "$(uname -r)"
printf "\n"
```

```
date=`date`
load=`cat /proc/loadavg | awk '{print $1}'`
root_usage=`df -h / | awk '/\// {print $(NF-1)}'`
memory_usage=`free -m | awk '/Mem:/ { total=$2 } /buffers\/cache/ { used=$3 } END {
printf("%3.1f%%", used/total*100)}'`
swap_usage=`free -m | awk '/Swap/ { printf("%3.1f%%", "exit !$2;$3/$2*100") }'`
users=`users | wc -w`
time=`uptime | grep -ohe 'up .*' | sed 's/,/\ hours/g' | awk '{ printf $2" "$3 }'`
processes=`ps aux | wc -l`
ethup=$(ip -4 ad | grep 'state UP' | awk -F ":" '!/^[0-9]*: ?lo/ {print $2}')
ip=$(ip ad show dev $ethup |grep -v inet6 | grep inet|awk '{print $2}')

echo "System information as of: $date"
echo
printf "System load:\t%s\tIP Address:\t%s\n" $load $ip
printf "Memory usage:\t%s\tSystem uptime:\t%s\n" $memory_usage "$time"
printf "Usage on /:\t%s\tSwap usage:\t%s\n" $root_usage $swap_usage
printf "Local Users:\t%s\tProcesses:\t%s\n" $users $processes
echo

[ -f /etc/motd.tail ] && cat /etc/motd.tail || true
```

(Idea from here.)

Vim

Config

File `.vimrc`:

```
# choose theme color
colo desert

# enable syntax highlighting
syntax on
```

```
# set mouse
set mouse=a

# you can show line numbers
set number

# show the editing mode on the last line
set showmode

# tell vim to keep a backup file
set backup

# tell vim where to put its backup files but the file must be created
set backupdir=~/.vim_backup/

# tell vim where to put swap files
# set dir=/private/tmp

# i don't use autoindent, but here's how to configure it:
# set autoindent

# highlight matching search strings
# set hlsearch

# make searches case insensitive
# set ignorecase
```

Plugins

Vim-powerline

```
sudo dnf install vim-powerline
```

Tmux

Installation

> Tmux is a Terminal Multiplexer. It enables a number of terminals to be created, accessed and controlled from a single screen.

Install with `dnfi tmux -y` to get version 1.8, but there isn't plugins available... so, see next!

Build Tmux

Source here.

First, install C compiler:

```
sudo dnf install -y gcc
```

Build Libevent

Libevent source releases.

```
wget https://github.com/libevent/libevent/releases/download/release-2.1.8-stable/libevent-2.1.8-stable.tar.gz
tar zxvf libevent-*.tar.gz
cd libevent-2.1.8-stable
mkdir -p $HOME/.local
./configure --prefix="$HOME/.local"
make -j && make install
```

Build Ncurses

Ncurses source releases.

```
wget http://ftp.gnu.org/pub/gnu/ncurses/ncurses-6.1.tar.gz
tar zxvf ncurses-6.1.tar.gz
cd ncurses-6.1
./configure --prefix="$HOME/.local"
make -j && make install
```

Build Tmux

Tmux source releases.

```
wget https://github.com/tmux/tmux/releases/download/2.8/tmux-2.8.tar.gz
tar zxvf tmux-2.8.tar.gz
cd tmux-2.8
./configure --prefix=$HOME/.local \
 > CPPFLAGS="-I$HOME/.local/include -I$HOME/.local/include/ncurses" \
 > LDFLAGS="-L$HOME/.local/lib"
make -j && make install
export PATH=$HOME/.local/bin:$PATH
export LD_LIBRARY_PATH=/home/mickael/.local/lib
```

To make folders persistent:

```
echo 'export PATH=$HOME/.local/bin:$PATH' >> ~/.zshrc
echo 'export LD_LIBRARY_PATH=$HOME/.local/lib' >> ~/.zshrc
```

Chech with :

```
$PATH
env | grep '^LD_LIBRARY_PATH'
```

Configuration

First param

Edit an other configuration file:

```
vim ~/.tmux.conf
```

Here an example of config's file.

And write:

```
# C-b is not acceptable -- Vim uses it
set-option -g prefix C-a
bind C-a last-window

# windows starts at 1
set -g base-index 1
set -g pane-base-index 1
```

```
# display messages for a second
set -g display-time 2500

# List of plugins
#set -g @plugin 'tmux-plugins/tpm'
#set -g @plugin 'tmux-plugins/tmux-sensible'

# Mouse support - set to on if you want to use the mouse
set -g mode-mouse on

# Toggle mouse mode to allow mouse copy/paste
# set mouse on with prefix m
bind m \
set -g mouse on \; \
display-message "Mouse: ON"
# set mouse off with prefix M
bind M \
set -g mouse off \; \
display-message "Mouse: OFF"

# bind reload
bind r source-file ~/.tmux.conf

# Use Vi mode
set -g mode-keys vi

# Other examples:
# set -g @plugin 'github_username/plugin_name'
# set -g @plugin 'git@github.com/user/plugin'
# set -g @plugin 'git@bitbucket.com/user/plugin'

# Initialize TMUX plugin manager (keep this line at the very bottom of tmux.conf)
#run -b '~/.tmux/plugins/tpm/tpm'
```

It gives access to vim keys:

Function	**vi**		**Function**	**vi**
Back to indentation	^		Half page up	C-u
Clear selection	Escape		Next page	C-f

Function	vi		Function	vi
Copy selection	Enter		Next word	w
Cursor down	j		Paste buffer	p
Cursor left	h		Previous page	C-b
Cursor right	l		Previous word	b
Cursor to bottom line	L		Quit mode	q
Cursor to middle line	M		Scroll down	C-Down or J
Cursor to top line	H		Scroll up	C-Up or K
Cursor up	k		Search again	n
Delete entire line	d		Search backward	?
Delete to end of line	D		Search forward	/
End of line	$		Start of line	0
Goto line	:		Start selection	Space
Half page down	C-d			

Second param

Edit the tmux.plugin.zsh.

```
vim ~/.oh-my-zsh/plugins/tmux/tmux.plugin.zsh
```

Set autostart:

```
# Automatically start tmux
: ${ZSH_TMUX_AUTOSTART:=true}
```

Full:

```
if ! (( $+commands[tmux] )); then
  print "zsh tmux plugin: tmux not found. Please install tmux before using this plugin." >&2
  return 1
fi

# ALIASES

alias ta='tmux attach -t'
alias tad='tmux attach -d -t'
alias ts='tmux new-session -s'
```

```
alias tl='tmux list-sessions'
alias tksv='tmux kill-server'
alias tkss='tmux kill-session -t'

# CONFIGURATION VARIABLES
# Automatically start tmux
: ${ZSH_TMUX_AUTOSTART:=true}
# Only autostart once. If set to false, tmux will attempt to
# autostart every time your zsh configs are reloaded.
: ${ZSH_TMUX_AUTOSTART_ONCE:=true}
# Automatically connect to a previous session if it exists
: ${ZSH_TMUX_AUTOCONNECT:=true}
# Automatically close the terminal when tmux exits
#: ${ZSH_TMUX_AUTOQUIT:=$ZSH_TMUX_AUTOSTART}
: ${ZSH_TMUX_AUTOQUIT:=false}
# Set term to screen or screen-256color based on current terminal support
: ${ZSH_TMUX_FIXTERM:=true}
# Set '-CC' option for iTerm2 tmux integration
: ${ZSH_TMUX_ITERM2:=false}
# The TERM to use for non-256 color terminals.
# Tmux states this should be screen, but you may need to change it on
# systems without the proper terminfo
: ${ZSH_TMUX_FIXTERM_WITHOUT_256COLOR:=screen}
# The TERM to use for 256 color terminals.
# Tmux states this should be screen-256color, but you may need to change it on
# systems without the proper terminfo
: ${ZSH_TMUX_FIXTERM_WITH_256COLOR:=screen-256color}
# Set the configuration path
: ${ZSH_TMUX_CONFIG:=$HOME/.tmux.conf}
# Set -u option to support unicode
: ${ZSH_TMUX_UNICODE:=false}

# Determine if the terminal supports 256 colors
if [[ $terminfo[colors] == 256 ]]; then
  export ZSH_TMUX_TERM=$ZSH_TMUX_FIXTERM_WITH_256COLOR
else
  export ZSH_TMUX_TERM=$ZSH_TMUX_FIXTERM_WITHOUT_256COLOR
fi
```

```
# Set the correct local config file to use.
if [[ "$ZSH_TMUX_ITERM2" == "false" && -e "$ZSH_TMUX_CONFIG" ]]; then
  export ZSH_TMUX_CONFIG
  export _ZSH_TMUX_FIXED_CONFIG="${0:h:a}/tmux.extra.conf"
else
  export _ZSH_TMUX_FIXED_CONFIG="${0:h:a}/tmux.only.conf"
fi

# Wrapper function for tmux.
function _zsh_tmux_plugin_run() {
  if [[ -n "$@" ]]; then
    command tmux "$@"
    return $?
  fi

  local -a tmux_cmd
  tmux_cmd=(command tmux)
  [[ "$ZSH_TMUX_ITERM2" == "true" ]] && tmux_cmd+=(-CC)
  [[ "$ZSH_TMUX_UNICODE" == "true" ]] && tmux_cmd+=(-u)

  # Try to connect to an existing session.
  [[ "$ZSH_TMUX_AUTOCONNECT" == "true" ]] && $tmux_cmd attach

  # If failed, just run tmux, fixing the TERM variable if requested.
  if [[ $? -ne 0 ]]; then
    if [[ "$ZSH_TMUX_FIXTERM" == "true" ]]; then
      tmux_cmd+=(-f "$_ZSH_TMUX_FIXED_CONFIG")
    elif [[ -e "$ZSH_TMUX_CONFIG" ]]; then
      tmux_cmd+=(-f "$ZSH_TMUX_CONFIG")
    fi
    $tmux_cmd new-session
  fi

  if [[ "$ZSH_TMUX_AUTOQUIT" == "true" ]]; then
    exit
  fi
}

# Use the completions for tmux for our function
```

```
compdef _tmux _zsh_tmux_plugin_run
# Alias tmux to our wrapper function.
alias tmux=_zsh_tmux_plugin_run

# Autostart if not already in tmux and enabled.
if [[ -z "$TMUX" && "$ZSH_TMUX_AUTOSTART" == "true" && -z "$INSIDE_EMACS" && -z "$EMACS" && -z
"$VIM" ]]; then
  # Actually don't autostart if we already did and multiple autostarts are disabled.
  if [[ "$ZSH_TMUX_AUTOSTART_ONCE" == "false" || "$ZSH_TMUX_AUTOSTARTED" != "true" ]]; then
    export ZSH_TMUX_AUTOSTARTED=true
    _zsh_tmux_plugin_run
  fi
fi
```

Plugins

- Tpm is a plugin manager:
  - `prefix` + `I` to install new plugin.
  - `prefix` + `U` to update plugins.
  - `prefix` + `alt` `u` to remove/uninstall plugins not on the plugin list.
- Tmux-resurrect to save & restore session after reboot:
  - `prefix` + `Ctrl` `s` to save.
  - `prefix` + `Ctrl` `r` to restore.
- Tmux-yank to copy to system clipboard (need configuration steps).
- Tmux-copycat enable regex search with `prefix` + `/`:
  - `prefix` + `ctrl` `f` : simple file search.
  - `prefix` + `ctrl` `g` : jumping over git status files (best used after git status command).
  - `prefix` + `alt` `h` : jumping over SHA-1/SHA-256 hashes (best used after git log command).
  - `prefix` + `ctrl` `u` : url search (http, ftp and git urls).
  - `prefix` + `ctrl` `d` : number search (mnemonic d, as digit).
  - `prefix` + `alt` `i` : ip address search.
- Themes pack.
  - To unable theme packs like powerline, you need to install powerline fonts on ssh machine as well (here) and change font for a powerline font in the terminal preferencies.

Various app

- Links : Web browser running in both graphics and text mode `sudo dnf install -y links`.
- Tree : Tree structure for folders `sudo dnf install -y tree` (tips).

Security

SSH Key authentication

From Nicolas Kovacs from microlinux.fr.

From host machin (it takes a while):

```
ssh-keygen -t rsa -b 16384
> Generating public/private rsa key pair.
> Enter file in which to save the key (/Users/mickael/.ssh/id_rsa):
> Enter passphrase (empty for no passphrase):
> Enter same passphrase again:
> Your identification has been saved in /Users/mickael/.ssh/id_rsa.
> Your public key has been saved in /Users/mickael/.ssh/id_rsa.pub.
> The key fingerprint is: [...]
```

Send public key to the distant server:

```
ssh-copy-id -i .ssh/id_rsa.pub mycentos
> /usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: ".ssh/id_rsa.pub"
> /usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
> /usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
> <username>@mycentos's password:

> Number of key(s) added:        1

> Now try logging into the machine, with: "ssh 'mycentos'" and check to make sure that only the key(s) you wanted were added.
```

Connect and check:

```
ssh mycentos
> Last login: Tue Apr 28 14:01:03 2020 from 192.168.1.198
```

```
cat ~/.ssh/authorized_keys
```

GPG Keys

Check gpg keys:

```
gpg --list-keys
```

Create:

```
gpg --gen-key
```

Re-check.

Echanging key:

```
# For export
gpg --output <name>.gpg --export <UID>
```

```
# For ASCII format
gpg --armor --export <UID>
```

Edit `~/.zshrc` and add:

```
# Fix gpg passphrase
export GPG_TTY=$(tty)
```

To test:

```
echo "test" | gpg2 --clearsign
```

Packages

Packages search for Linux & Unix.

Nmap

> Nmap is one of the most popular tools for network mapping. You can discover active hosts within a network, and a wide range of other detection features. Nmap has functions for host discovery, port scanning, OS detection, app versions, and other scripting interactions.

```
sudo dnf install -y nmap
```

Others

- Kismet Wireless
- John the Ripper
- THC Hydra
- Metasploit Framework
- OpenVAS

Configuration

- FirewallD sur un serveur LAN by MicroLinux □□□□.
- FirewallD sur un serveur dédié by MicroLinux □□□□.
- Bloquer les attaques par force brute avec Fail2ban sous CentOS 7 by MicroLinux □□□□.

Check

For check open ports and write it in a file:

```
mkdir tmp
nmap -p 0-65535 portquiz.net > tmp/nmaptest
grep filtered tmp/nmaptest
```

Git

Config

Git tools - signing your work

```
git config --global user.email <user@exemple.com>
git config --global user.name <Username>
```

```
git config --global user.signingkey = <UID>
git config --global color.ui = always
git config --global color.branch = always
git config --global color.diff = always
git config --global color.interactive = always
git config --global color.status = always
git config --global push.default = simple
git config --global gpg.program = gpg2
git config --global commit.gpgSign = true
```

To get in `.gitconfig`:

```
[user]
        email = <user@exemple.com>
        name = <Username>
        signingkey = <signingkey>
[color]
        ui = always
        branch = always
        diff = always
        interactive = always
        status = always
[push]
        default = simple
[gpg]
        program = gpg2
[commit]
        gpgSign = true
```

Tips

Zsh

- Official site.
- Find all files in `path` that contain `text` : `sudo grep -Rnil 'text' path`.
- 75 dnf commands, tips & plugins.
- Fonction used for ignore tmux.plugin.zsh:

```
var="$HOME/.oh-my-zsh" ; find "$var" | while read -r line; do echo "$var/$line"; done >>
~/.gitignore
```

- Add timestamp to a file:

```
h >> "~/list-$(date '+%s').txt"
```

- Test script with ShellCheck.
- Make dir and cd it:

```
mkdir "<folder>" && cd "$_"
```

- Or paste following lines in `.zshrc` and refresh after:

```
mkcdir ()
{
    mkdir -p -- "$1" &&
cd -P -- "$1"
}
```

Dnf

- Dnf doc.
- 25 useful dnf command examples.
- How to install and remove packages with DNF in Fedora.
- Dnf, le gestionnaire de paquets de Fedora □□□□.

Tmux

- Shortcuts & cheatsheet.
- Cheat sheet 1.
- Cheat sheet 2.
- Cheat sheet 3.
- Manual.
- Help : `Ctrl` `b` + `?` (or `Ctrl` `a` + `?` in my case).

Git

- Git tutorial.
- Git tutorial from Bitbucket.
- Devenez un expert de git.
- List files not add and not in .ignore file : `git ls-files -o --exclude-standard | cut -d/ -f1 | uniq`.
- Add user sign key: `git config --global user.signingkey <gpg-key-id>`.
- Sign a commit : `git commit -a -S -m 'commit signé'`.

Vim

- Vim The editor original tips.
- Cheat sheet.
- Save sudo without root permission: `:w! sudo tee %`.
- Delete tips.
- Comment lines 66 to 70: `:66,70s/^/#`
- To yank all the lines and copy to the clipboard : `:% y +` or `:g g " + y G`.
- Clear the last search: `:let @/ = ""` or `noh`.
- Search & Replace.
- Vim Tips form Alvin Alexander.

pip

- Upgrade pip: `pip install --upgrade pip`

Various

- Check if a package if present `rpm -qa | grep -i packageName`.
- Use `history`.
- Interactive search : `Ctrl` `r`.
- How to install an RPM package into a different directory in CentOS/RHEL/Fedora.
- Special-use IPv4 addresses.
- To know its ip address: `host myip.opendns.com resolver1.opendns.com`.
- Remove user and every trace.

- Tar command with examples.
- Bash reference manual.
- Command not found.

Various

Sources

- MicroLinux □□□□.
- It-Connect.fr □□□□.
- Pwet/man/linux □□□□.
- IP addresses tools.
- Linux help.
- PhoenixNAP.
- GPG with LinuxPedia.

Go further

- How to set up your own private Git server on Linux.
- RaspberryPi OwnCloud.
- RaspberryPi projetcs.
- Installer un serveur de réseau local & Make a samba server on RHEL 7.
- How to cross-compile

Todo

- ☐ Find cross-compiling for nodeJS : 1 2 3.
- ☐ To try to install Joplin.
- ☐ Find to install on hdd or ssd.

Révision #7
Créé 13 avril 2021 10:51:18 par Mickaël G.
Mis à jour 22 octobre 2022 06:58:02 par Mickaël G.