Cent OS 7.0

Initial setup

New user / Configure SSH

Yum

Path	Meaning
`yum provides *bin/dig`	

Configuration files

Path	Meaning
`visudo`	edit sudo configuration
`/etc/ssh/sshd_config`	edit SSH configuration
`rpm -ql wordpress`	Show all files installed by the package

Transport layer

Path	Meaning
`sudo ss -tulpn`	
`sudo ps -ef`	

Samba

Required packages

- `libsemanage-python`
- `samba-common`
- `samba`
- `samba-client`

Services & firewall

Command	Meaning
`nmb`	service
`smb`	
`samba`	firewalld

SELinux

Command	Meaning
`samba_enable_home_dirs`	`seboolean` in ansible
`samba_export_all_rw`	
`public_content_rw_t`	`setype` in ansible

Configuration file

```
# Samba configuration, managed by Ansible. Please don't edit manually
# {{ ansible_managed }}
#
# vim: ft=samba

[global]
 # Server information
 netbios name = {{ samba_netbios_name }}
 workgroup = {{ samba_workgroup|default('WORKGROUP') }}
 server string = {{ samba_server_string|default('Fileserver %m') }}

 # Logging
{% if samba_log is defined %}
 log file = {{ samba_log }}
 max log size = {{ samba_log_size|default('5000') }}
{% else %}
 syslog only = yes
 syslog = 1
{% endif %}

 # Authentication
 security = {{ samba_security|default('user') }}
 passdb backend = {{ samba_passdb_backend|default('tdbsam') }}
 map to guest = {{ samba_map_to_guest|default('bad user') }}

 # Name resolution: make sure \\NETBIOS_NAME\ works
 wins support = yes
 local master = yes
 domain master = yes
 preferred master = yes

{% if samba_load_printers is defined and samba_load_printers == 'no' %}
 # Don't load printers
 load printers = no
 printing = bsd
 printcap name = /dev/null
 disable spoolss = yes

{% endif %}
{% if samba_enable_homes is defined and samba_enable_homes == 'yes' %}
## Make home directories accessible
[homes]
 comment = Home Directories
 browseable = no
 writable = yes

{% endif %}
{% for share in samba_shares %}
[{{ share.name }}]
  comment = {{ share.comment|default(share.name) }}
  path = {{ samba_share_root }}/{{ share.name }}
  public = {{ share.public|default('no') }}
{% if share.valid_users is defined %}  valid users = {{ share.valid_users }}
{% endif %}
{% if share.write_list is defined %}  write list = {{ share.write_list }}
{% endif %}
{% if share.force_group is defined %}  force group = {{ share.force_group }}
{% endif %}
```

```
{% if share.create_mask is defined %}  create mask = {{ share.create_mask }}
{% endif %}
{% if share.create_mode is defined %}  create mode = {{ share.create_mode }}
{% endif %}
{% if share.force_create_mode is defined %}  force create mode = {{ share.force_create_mode }}
{% endif %}
{% if share.directory_mask is defined %}  directory mask = {{ share.directory_mask }}
{% endif %}
{% if share.directory_mode is defined %}  directory mode = {{ share.directory_mode }}
{% endif %}
{% if share.force_directory_mode is defined %}  force directory mode = {{ share.force_directory_mode }}
{% endif %}

{% endfor %}
```

Enterprise Linux 7 (RedHat, CentOS)

This part is based on EL7 by bertvv. I would like to thank bertvv to share this cheatsheet.

Network configuration

Action	Command
List interfaces (and IP addresses)	`ip address`, `ip a`
Route table	`ip route`, `ip r`
DNS servers	`cat /etc/resolv.conf`
Set IP address of an interface*	`ip address add 192.168.56.1/24 dev vboxnet0`

(*) This example is actually a workaround for a bug that causes NetworkManager 0.9.9 to manage virtual network interfaces.

NetworkManager

Action	Command
Show available network connection profiles	`nmcli connection show`
Show active network connection profiles	`nmcli connection show active`
Show network device status	`nmcli device status`
Connect to profile CONNECTION	`nmcli connection up id CONNECTION`
Disconnect profile CONNECTION	`nmcli connection down id CONNECTION`
Query Wifi status	`nmcli radio wifi`
Turn Wifi on/off	`nmcli radio wifi {on,off}`
List available wireless networks	`nmcli device wifi list`
Refresh list of wireless networks	`nmcli device wifi rescan`
Connect to wireless network SSID	`nmcli device wifi connect SSID`

`connection` and `device` can be abbreviated to `con` and `dev`, respectively.

Resources

- RedHat Enterprise Linux 7 Networking Guide
- Fedora Wiki: Networking/CLI

Managing services with `systemctl`

Action	Command
List services	`systemctl list-units --type service`
Query SERVICE status	`sudo systemctl status SERVICE.service`
List failed services on boot	`sudo systemctl --failed`
Start SERVICE	`sudo systemctl start SERVICE.service`
Stop SERVICE	`sudo systemctl stop SERVICE.service`
Restart SERVICE	`sudo systemctl restart SERVICE.service`
Kill SERVICE (all processes) with SIGTERM	`sudo systemctl kill SERVICE.service`
Kill SERVICE (all processes) with SIGKILL	`sudo systemctl kill -s SIGKILL SERVICE.service`
Start SERVICE on boot	`sudo systemctl enable SERVICE.service`
Don't start SERVICE on boot	`sudo systemctl disable SERVICE.service`

Resources

- RedhHat 7 System Administrator's Guide
- Systemd for Administrators, Part IV: Killing Services

Perusing system logs with `journalctl`

Viewing logs requires root privileges. However, users that are members of the `adm` group get access as well. So, add your user to the `adm` group to make viewing logs easier.

Action	Command
Show log since last boot	`journalctl -b`
Kernel messages (like `dmesg`)	`journalctl -k`
Show latest log and wait for changes	`journalctl -f`
Reverse output (newest first)	`journalctl -r`
Show only errors and worse	`journalctl -b -p err`
Filter on time (example)	`journalctl --since=2014-06-00 --until="2014-06-07 12:00:00"`
Since yesterday	`journalctl --since=yesterday`
Show only log of SERVICE	`journalctl -u SERVICE`
Match executable, e.g. `dhclient`	`journalctl /usr/sbin/dhclient`
Match device node, e.g. `/dev/sda`	`journalctl /dev/sda`

Resources

- Systemd for Administrators, Part XVII: Using the journal

Configuring the firewall with `firewalld`

The `firewalld-cmd` should run with root privileges, do always use `sudo`.

Action	Command
Firewall state	`firewall-cmd --state`
Reload permanent rules	`firewall-cmd --reload`
Currently enabled features	`firewall-cmd --list-all-zones`
List supported zones	`firewall-cmd --get-zones`
List preconfigured services	`firewall-cmd --get-services`
Enabled features in current zone	`firewall-cmd --list-all`
Enabled features in zone	`firewall-cmd [--permanent] [--zone=ZONE] --list-all`
Enable a service in zone	`firewall-cmd [--permanent] [--zone=ZONE] --add-service=http`
Remove service frome zone	`firewall-cmd [--permanent] [--zone=ZONE] --remove-service=http`

Action	Command
Enable a port in zone	`firewall-cmd [--permanent] [--zone=ZONE] --add-port=80/tcp`
Remove a port from zone	`firewall-cmd [--permanent] [--zone=ZONE] --remove-port=80/tcp`
Turn panic mode on	`firewall-cmd --panic-on`
Turn panic mode off	`firewall-cmd --panic-off`

- Configuration is stored in `/etc/firewalld` and `/usr/lib/firewalld`
- The default zone is `public`, which you don't have to specify on the command line when adding/removing rules
- Adding permanent rules

Resources

- Using Firewalls, in *RHEL 7 Security Guide*
- FirewallD, in *Fedora Project Wiki*