The C-ALE (Cloud & Container Apprentice Linux Engineer) is a series of seminars held at existing conferences covering topics which are fundamental to a Linux professional in the Linux Cloud and Container field of computing.

This seminar will spend equal time on lecture and hands on labs at the end of each seminar which allow you to practice the material you’ve learned.

This material makes the assumption that you have minimal experience with using Linux in general, and a basic understanding of general industry terms. The assumption is also made that you have access to your own computers upon which to practice this material.

More information can be found at [https://cm.ossessentials.org/](https://cm.ossessentials.org/)

This material is licensed under [CC-BY SA4](https://creativecommons.org/licenses/by-sa/4.0/)
# Contents

1 systemd 1
   1.1 Introduction to systemd for Administrators  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
   1.2 systemd for Administrators  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
   1.3 systemd examples  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
   1.4 systemd optional features  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
   1.5 Labs  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
Chapter 1

systemd

1.1 Introduction to systemd for Administrators .................................................. 2
1.2 systemd for Administrators ............................................................................ 5
1.3 systemd examples ......................................................................................... 11
1.4 systemd optional features ............................................................................. 20
1.5 Labs .............................................................................................................. 30
1.1 Introduction to systemd for Administrators

Hi, I’m Lee Elston, your presenter for this session.
You can contact me at:
lee@matildasystems.com

I am a course maintainer and instructor for The Linux Foundation and main-tain:

- LFS311
- LFS430
- LFS416
Introduction

This presentation will focus on the following administration aspects of systemd including the following:

- What is systemd
- Getting your system services started
- Configuration files/directories
- builtin and optional services
To start a Unix/Linux system we need an initialization program to start the system environment. We also would like an easy method to launch services both system and user related. Several methods were used:

- An initialization program and one big script running on the default shell
- An initialization program and a bunch of little scripts
- An initialization program and the old bunch of scripts with the beginnings of dependency processing.
- An initialization program no scripts, just configuration files with dependencies and sequencing available.
What is systemd

**systemd** is the latest evolution in system startup programs for **Linux**. The main objectives are:

- **startup or system manager**
  - runs as "init" or "pid 1"
  - launched by the kernel during initialization
  - starts the rest of the system

- **session or service manager**
  - start, stop and status of background processes
  - SysV init.d script compatible
systemd has a control program **systemctl** to manage the background processes and services it starts. Some of the options are:

- `systemctl` outputs a list of all "units" and their state
- `systemctl -t services` list the "service units" and their state
- `systemctl status cups` show detailed status for "cups.service"
  (the .service is default other types must be specified)
- `systemctl start cups` start the "cups.service" now
- `systemctl stop cups` stop the "cups.service" now
- `systemctl enable cups` start the "cups.service" at boot time
- `systemctl disable cups` do not auto start this service at boot time

Please consult **man systemctl** for additional details.
systemd has a three-tiered configuration, if one of the tiers is not present the next highest priority files is used.

- Configuration files/directories, priority lowest to highest.
  - vendor/packager
  - dynamic
  - administrator
The typical directory locations

- the vendor supplied configuration file is at:
  /lib/systemd/system/<service-name>.service
  or
  /usr/lib/systemd/system/<service.name>.service
  or
  /usr/lib/systemd/system/<service-name>.d/*.conf

- temporary files created at runtime:
  /run/systemd/

- local administrator configuration files:
  /etc/systemd/system
## Configuration File Types

**systemd** has several types of configuration elements, called **units**.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service</td>
<td>which start and control daemons and the processes they consist of</td>
</tr>
<tr>
<td>socket</td>
<td>local IPC or network socket</td>
</tr>
<tr>
<td>target</td>
<td>group of units or boot up points</td>
</tr>
<tr>
<td>device</td>
<td>exposes kernel devices to systemd, allows device based activation</td>
</tr>
<tr>
<td>mount</td>
<td>control the mount points</td>
</tr>
<tr>
<td>automount</td>
<td>provides on demand mounting of filesystems</td>
</tr>
<tr>
<td>timer</td>
<td>trigger activation of other units based on timer</td>
</tr>
<tr>
<td>swap</td>
<td>mount swap files</td>
</tr>
<tr>
<td>path</td>
<td>activate other units if filesystem objects change</td>
</tr>
<tr>
<td>slice</td>
<td>group units for easier management</td>
</tr>
<tr>
<td>scope</td>
<td>like service units but manage foreign processes as well</td>
</tr>
</tbody>
</table>
## ”Unit” documentation

Table 1.2: unit type man pages

<table>
<thead>
<tr>
<th>Type</th>
<th>man page</th>
</tr>
</thead>
<tbody>
<tr>
<td>service</td>
<td>systemd.service(5)</td>
</tr>
<tr>
<td>socket</td>
<td>systemd.socket(5), daemon(7)</td>
</tr>
<tr>
<td>target</td>
<td>systemd.target(5)</td>
</tr>
<tr>
<td>device</td>
<td>systemd.device(5)</td>
</tr>
<tr>
<td>mount</td>
<td>systemd.mount(5)</td>
</tr>
<tr>
<td>automount</td>
<td>systemd.automount(5)</td>
</tr>
<tr>
<td>timer</td>
<td>systemd.timer(5)</td>
</tr>
<tr>
<td>swap</td>
<td>systemd.swap(5)</td>
</tr>
<tr>
<td>path</td>
<td>systemd.path(5)</td>
</tr>
<tr>
<td>slice</td>
<td>systemd.slice(5)</td>
</tr>
<tr>
<td>scope</td>
<td>systemd.scope(5)</td>
</tr>
<tr>
<td>”see also”</td>
<td>systemd.special(7), systemd.units(5), systemd-system.conf(5)</td>
</tr>
</tbody>
</table>
1.3 systemd examples

This section illustrates some examples of **systemd** configuration files.

- default configuration
- customize a configuration
- systemd-timedated
- systemd-delta
- systemd-sysctl
default configuration

Extract the list of files, looking for **systemd** configuration files.

```
root@ubuntu:~# dpkg -L vsftpd | grep systemd
/lib/systemd
/lib/systemd/system
/lib/systemd/system/vsftpd.service
```

In this package **vsftpd** the packager has added a **systemd** configuration file.

Notice the location of the config file.
default service config

The contents of the default file:

root@ubuntu:~# cat /lib/systemd/system/vsftpd.service

[Unit]
Description=vsftpd FTP server
After=network.target

[Service]
Type=simple
ExecStart=/usr/sbin/vsftpd /etc/vsftpd.conf
ExecReload=/bin/kill -HUP $MAINPID
ExecStartPre=/bin/mkdir -p /var/run/vsftpd/empty

[Install]
WantedBy=multi-user.target
customizing service file

If the packager supplied configuration file is modified it may get overwritten by a package update. There may be additional changes, so a configuration directory will be used.

```
root@ubuntu:~# mkdir -p /etc/systemd/system/vsftpd.service.d

root@ubuntu:~# cat /etc/systemd/system/vsftpd.service.d/
    00-vsftpd.conf

[Service]
ExecStart=
ExecStart=/usr/sbin/vsftpd /etc/vsftpd.conf -o ftpd_banner="Whoo Hoo it works"

root@ubuntu:~# systemctl daemon-reload
root@ubuntu:~# systemctl restart vsftpd

root@ubuntu:~# ftp localhost
Connected to localhost.
220 Whoo Hoo it works
Name (localhost:student):
The configuration file for vsftpd in **Fedora** is a little different.

```
[root@fedora ~]# rpm -ql vsftpd | grep systemd
/usr/lib/systemd/system-generators/vsftpd-generator
/usr/lib/systemd/system/vsftpd.service
/usr/lib/systemd/system/vsftpd.target
/usr/lib/systemd/system/vsftpd@.service
```

Notice the directory is slightly different. **systemd** looks in both directories so it is a a choice.
CHAPTER 1. SYSTEMD

no-frills service file

One of the service files supplied with vsftpd in Fedora is almost identical to the Ubuntu counterpart.

[root@fedora ~]# cat /usr/lib/systemd/system/vsftpd.service
[Unit]
Description=Vsftpd ftp daemon
After=network.target

[Service]
Type=forking
ExecStart=/usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf

[Install]
WantedBy=multi-user.target
some additional features

```bash
[root@fedora ~]# cat /usr/lib/systemd/system/vsftpd@.service
[Unit]
Description=Vsftpd ftp daemon
After=network.target
PartOf=vsftpd.target

[Service]
Type=forking
ExecStart=/usr/sbin/vsftpd /etc/vsftpd/%i.conf

[Install]
WantedBy=vsftpd.target

Notice:

- the **PartOf** option that connects this service file to the `vsftp.target`
- the `@` in the name indicates this is a **template**
- the `%i` variable that contains the **instance** name.
```
• **generators** are usually run very early in the initialize process to create or customize configuration files.

```bash
[root@fedora ~]# cat /usr/lib/systemd/system-generators/vsftpd-generator
#!/usr/bin/bash
confdir=/etc/vsftpd
unitdir=/usr/lib/systemd/system
targetdir=$1/vsftpd.target.wants

mkdir -p ${targetdir}

for f in $(ls -1 ${confdir}/*.conf | awk -F "." '{print $1}' | awk -F "/" '{print $4}')
do
  echo "Generating systemd units for $f"
  ln -s ${unitdir}/vsftpd\@.service ${targetdir}/vsftpd\@$f.service > /dev/null 2>&1
done

exit 0
```

See **man 7 systemd.generator** for more information.
1. shutdown vsftpd
2. copy the config file /etc/vsftpd/vsftpd.conf to /etc/vsftpd/blue.conf
3. modify the port number in the new file
4. tell systemd
   # systemctl --system daemon-reload
5. start the vsftpd.target
6. confirm the number of active ftp servers
1.4 systemd optional features

This is a small sampling of the optional features available with `systemd`. See the official home page at: https://www.freedesktop.org/wiki/Software/systemd/ for more information.
This command/feature **systemd-delta** is an essential command for administrators, it finds the configuration files that are:

- overridden
- masked
- equivalent
- redirected
- extended
- unchanged
systemd-timedated

- system service that terminates when unneeded
- user interface **timedatectl**
- change system clock and timezone
- enable and disable NTP service

The **systemd-timedated** will launch an NTP client when the boolean **set-ntp true** is set, assuming the previous state was **false**. **systemd-timedated** will attempt to launch:

- **chronyd.service**
- **ntpd.service**
- **systemd-timesyncd.service**

The content of the list and order can be altered by altering the **$SYSTEMD_TIMEDATED_NTP_SERVICES** environment variable.
systemd-timesyncd.service is a time synchronization client with the following properties:

- client only, no server functions
- uses SNTP protocol
  - "steps" time in to synchronization
  - uses one time source at a time
- configuration files is: /etc/systemd/timesyncd.conf
systemd-sysctl applies kernel parameters early in the boot sequence.

- applies the `.conf` file in `/etc/sysctl.d` directory
- if the original `/etc/sysctl.conf` is desired create a soft link into the `/etc/sysctl.d` directory
systemd-networkd

systemd-networkd is an optional component that manages networks.

- consistent configuration files across distributions
- does not read the historical network configuration files
- configuration files in "INI" format
- main configuration files: /{lib,run,etc}/systemd/network/ directories see: systemd.network(5)
- low level link settings are independently set see systemd.link(5)
- non-physical devices are defined in: systemd.netdev(5) files

The default configuration will start a dhcp client on the interfaces at boot time.

Debian and friends have a tool that generates either NetworkManager or systemd.networkd configuration files from a "yaml" input file.
test drive systemd-networkd

Assuming that NetworkManager is the active configuration tool, testing systemd-networkd only takes a few steps for a DHCP example.

• create a configuration file
• switch off netplan (if applicable)
• disable NetworkManager
• enable systemd-networkd
• reboot (clears any temporary configs)

Test the new configuration and explore the networking options in systemd-networkd
create a config file

- first record the name of the adapter:
  
  ```
  $ ip -br address
  lo    UNKNOWN    127.0.0.1/8   ::1/128
  enp1s0 UP        192.168.122.42/24 fe80::c2dd:87d:5eb3:9ca5/64
  ```

- build a config file in the "local administrator" directory:
  
  ```
  $ cat /etc/systemd/network/20-dhcp.network
  [Match]
  Name=enp*

  [Network]
  DHCP=yes
  ```
switch off netplan

For this example disable netplan if it is enabled.

- rename the primary netplan input file directory
  
  # mv /etc/netplan /etc/netplan-off
Switch network management and reboot

Set the automatic start on `systemd-networkd`

- disable NetworkManager
  
  ```
  $ sudo systemctl disable NetworkManager
  ```

- enable `systemd-networkd`
  
  ```
  $ sudo systemctl enable systemd-networkd
  ```

- reboot
  
  ```
  $ sudo systemctl reboot
  ```
1.5 Labs

Exercise 1.1: systemd

There is a high probability that your system has systemd installed and running. If your system does not then this exercise set will not function.

This exercise is going to create several configuration files to illustrate how systemd organizes its configuration.

Normally administrator added services would be added to etc/systemd directory but in this exercise we are going to create two levels of configuration, the "vendor" and "administration" files.

Looking at the man pages to determine the location of the systemd configuration files can be a little complex as systemd looks in various places for its files. Taking a practical approach, we will look at an existing service to determine where your distro has placed the files.

1. An easy way to determine where the systemd configuration files are is to look. Locate the configuration files for cups and/or sshd.

   # systemctl status sshd

   The output should be similar to:

   sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: disabled)
   Active: active (running) since Sat 2019-08-10 06:27:29 CDT; 2 days ago
   Docs: man:sshd(8)
       man:sshd_config(5)
   Main PID: 687 (sshd)
   Tasks: 1 (limit: 2356)
   Memory: 4.5M
   CGroup: /system.slice/sshd.service
       | _687 /usr/sbin/sshd -D -oCiphers=aes256-gcm@openssh.com,chacha20-poly1305@openssh.com
   
   ..... some text removed ....
Notice the line beginning with **Loaded** as it is pointing to the initial configuration file as supplied by the packager. This example shows the directory `/usr/lib/systemd/system` is the directory the packager used.

Record the directory name for future use.

2. Copy the default configuration to a user home directory.

   ```bash
   # cp /usr/lib/systemd/system/sshd.service /home/student/sshd.service
   ```

3. The existing file should look a lot like:

   ```bash
   [root@student]# cat sshd.service
   [Unit]
   Description=OpenSSH server daemon
   Documentation=man:sshd(8) man:sshd_config(5)
   After=network.target sshd-keygen.target
   Wants=sshd-keygen.target
   
   [Service]
   Type=notify
   EnvironmentFile=/etc/crypto-policies/back-ends/opensshserver.config
   EnvironmentFile=/etc/sysconfig/sshd
   ExecStart=/usr/sbin/sshd -D $OPTIONS $CRYPTO_POLICY
   ExecReload=/bin/kill -HUP $MAINPID
   KillMode=process
   Restart=on-failure
   RestartSec=42s
   
   [Install]
   WantedBy=multi-user.target
   ```

4. Rename the copy of the `sshd.service` file to `test_sshd.service`

5. Alter the contents of `test_sshd.service` to match the following:
# cat /usr/lib/systemd/system/test_sshd.service

[Unit]
Description=test
After=network.target sshd-keygen.target
Wants=sshd-keygen.target

[Service]
Type=notify
EnvironmentFile=-/etc/crypto-policies/back-ends/opensshserver.config
EnvironmentFile=-/etc/sysconfig/sshd
ExecStart=/usr/sbin/sshd -D $OPTIONS $CRYPTO_POLICY -p 4222
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process

[Install]
WantedBy=multi-user.target

6. The service name is the same as the file name. Copy the customized file back to the default directory.

    # cp test_sshd.service /usr/lib/systemd/system/

7. Tell systemd to re-read its configuration files.

    # systemctl daemon-reload

8. The service should be ready to start. Start the new service and request its status.

    # systemctl start test_sshd

    # systemctl status test_sshd
    test_sshd.service - test
        Loaded: loaded (/usr/lib/systemd/system/test_sshd.service; disabled; vendor preset: disabled)
        Active: active (running) since Mon 2019-08-12 10:50:24 CDT; 2s ago
        Main PID: 5449 (sshd)
Tasks: 1 (limit: 2356)
Memory: 1.2M
CGroup: /system.slice/test_sshd.service
| _5449 /usr/sbin/sshd -D -oCiphers=aes256-gcm@openssh.com,chacha20-poly1305@openssh.com

Aug 12 10:50:24 rt.example.com systemd[1]: Starting test...
Aug 12 10:50:24 rt.example.com sshd[5449]: Server listening on 0.0.0.0 port 4222.
Aug 12 10:50:24 rt.example.com sshd[5449]: Server listening on :: port 4222.

9. To enable the new service at boot time:

# systemctl enable test_sshd

**Exercise 1.2: Test a ”Drop-In” configuration in the admin directory**

This exercise will modify the newly created service and will the local admin directory /etc/systemd/system. One of the advantages of using the local admin directory is changes to the service file added by the vendor or packager could be made during update, overwritten local customization.

1. A Drop-In file can be a single file or a directory of many files, this example uses a directory. Create the new directory for the drop-in file.

# mkdir /etc/systemd/system/test_sshd.service

2. The Drop-In file will add to the existing service file. Sometimes if a value is already defined it must be cleared before changes can occur, the actual ExecStart command is one. Create drop-in file with the following. The port number was changed.

# cat /etc/systemd/system/test_sshd.service.d/00-first.conf

[Service]
ExecStart=
ExecStart=/usr/sbin/sshd -D $OPTIONS $CRYPTO_POLICY -p4242
3. Inform systemd that the configuration has been changed, then restart the service.

```
# systemctl daemon-reload
# systemctl restart test_sshd
```

4. The test_sshd service should be listening on a new port.

```
[root@rt system]# systemctl status test_sshd
test_sshd.service - test
   Loaded: loaded (/usr/lib/systemd/system/test_sshd.service; disabled; vendor preset: disabled)
   Drop-In: /etc/systemd/system/test_sshd.service.d
      |-00-first.conf
   Active: active (running) since Mon 2019-08-12 11:28:32 CDT; 8s ago
   Main PID: 5733 (sshd)
     Tasks: 1 (limit: 2356)
     Memory: 1.0M
   CGroup: /system.slice/test_sshd.service
            |-5733 /usr/sbin/sshd -D -oCiphers=aes256-gcm@openssh.com,chacha20-poly1305@openssh.com
Aug 12 11:28:32 rt.example.com systemd[1]: Starting test...
Aug 12 11:28:32 rt.example.com sshd[5733]: Server listening on 0.0.0.0 port 4242.
Aug 12 11:28:32 rt.example.com sshd[5733]: Server listening on :: port 4242.
```

**Exercise 1.3: Locating the customization's** This exercise looks at determining which configuration files are active for a **systemd unit**. This exercise assumes the successful completion of the previous exercises.

1. the **systemctl status test_vsftpd** command will show the following:

```
# systemctl status test_sshd
```

The output will show:

- service has a vendor default of "enabled" or "disabled"
- service has been manually "enabled" or "disabled"
- what the name and location of the vendor supplied config file
1.5. LABS

- the Drop-In directory and associated files
- limits and cgroup information
- a few lines of the log associated with this service

The output from the test system:

test_sshd.service - test
    Loaded: loaded (/usr/lib/systemd/system/test_sshd.service; disabled; vendor preset: disabled)
    Drop-In: /etc/systemd/system/test_sshd.service.d
        |-00-first.conf, 60-second.conf
    Active: active (running) since Wed 2019-08-14 08:44:50 CDT; 2s ago
    Main PID: 2629 (sshd)
    Tasks: 1 (limit: 3)
    Memory: 2.2M
    CGroup: /system.slice/test_sshd.service
        |-2629 /usr/sbin/sshd -D -oCiphers=aes256-gcm@openssh.com,chacha20-poly1305@openssh.com,aes256-ct>

Aug 14 08:44:50 rt.example.com systemd[1]: Starting test...
Aug 14 08:44:50 rt.example.com sshd[2629]: Server listening on 0.0.0.0 port 4242.
Aug 14 08:44:50 rt.example.com sshd[2629]: Server listening on :: port 4242.
Aug 14 08:44:50 rt.example.com systemd[1]: Started test.

2. It would be tedious to check every service for Drop-In files. **systemd-delta** is designed to find overridden configuration files. The output is a little wide so redirecting it to a file may be helpful.

# systemd-delta > /tmp/delta.txt

display the output:

# cat /tmp/delta.txt

[EQUIVALENT] /etc/systemd/system/default.target -> /usr/lib/systemd/system/default.target
[EXTENDED]  /etc/systemd/system/foo.service -> /etc/systemd/system/foo.service.d/00-foo.conf
[EXTENDED]  /usr/lib/systemd/system/systemd-udev-trigger.service ->
[EXTENDED]  /usr/lib/systemd/system/test_sshd.service -> /etc/systemd/system/test_sshd.service.d/00-first.conf
CHAPTER 1. SYSTEMD

Exercise 1.4: time synchronization with systemd-timedated

When synchronization of time is required or when the time client changes the operation must be verified. This exercise will determine if time synchronization is active, which client is being used and if necessary set the client to be systemd-timedated

1. determine which time clients are installed

```
# rpm -qa | grep -e ntp -e chrony -e syncd
yast2-ntp-client-4.1.8-lp151.1.1.noarch
chrony-3.2-lp151.8.6.x86_64
```

This indicates the package **chrony** is installed.

2. Check the current status with **timedatectl**

```
# timedatectl

Local time: Wed 2019-08-14 13:00:06 EDT
Universal time: Wed 2019-08-14 17:00:06 UTC
RTC time: Wed 2019-08-14 17:45:31
Time zone: America/Toronto (EDT, -0400)
Network time on: no
NTP synchronized: no
RTC in local TZ: no
```

3. Disable **chrony**.
1. Check the current configuration to verify which network management tools are being used.

Check to see if systemd-networkd is controlling the adapter.

Exercise 1.5: switch the network configuration to use systemd-networkd

1. Check the current configuration to verify which network management tools are being used.
# networkctl
WARNING: systemd-networkd is not running, output will be incomplete.

<table>
<thead>
<tr>
<th>IDX</th>
<th>LINK</th>
<th>TYPE</th>
<th>OPERATIONAL</th>
<th>SETUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lo</td>
<td>loopback</td>
<td>n/a</td>
<td>unmanaged</td>
</tr>
<tr>
<td>2</td>
<td>eth0</td>
<td>ether</td>
<td>n/a</td>
<td>unmanaged</td>
</tr>
</tbody>
</table>

2 links listed.

Notice the adapters are unmanaged.

2. Check to see if NetworkManager is controlling the adapter.

# nmcli
eth0: connected to Wired connection 1
    "Red Hat Virtio network device"
    ethernet (virtio_net), 52:54:00:B5:02:FF, hw, mtu 1500
    ip4 default
    inet4 192.168.122.112/24
    route4 0.0.0.0/0
    route4 192.168.122.0/24
    inet6 fe80::3eff:1998:d47:16d8/64
    route6 ff00::/8
    route6 fe80::/64
    route6 fe80::/64

lo: unmanaged
    "lo"
    loopback (unknown), 00:00:00:00:00:00, sw, mtu 65536

DNS configuration:
    servers: 192.168.122.1
    interface: eth0

Use "nmcli device show" to get complete information about known devices and
"nmcli connection show" to get an overview on active connection profiles.
This example is using NetworkManager for the network management.

3. Disable NetworkManager and check it is disabled.

   # systemctl disable --now NetworkManager

   # nmcli

4. Create a new configuration for systemd-networkd.

   # cat /etc/systemd/network/20-dhcp.network
   [Match]
   Name=eth0

   [Network]
   DHCP=yes

5. Enable systemd-networkd and verify.

   # systemctl enable --now systemd-networkd

   # networkctl

   IDX | LINK   | TYPE   | OPERATIONAL | SETUP  
   --  | ------ | ------ | ----------- | ------ 
   1   | lo    | loopback | carrier     | unmanaged 
   2   | eth0  | ether   | routable    | configuring

   2 links listed.

   Not quite ready yet.

   # networkctl

   IDX | LINK   | TYPE   | OPERATIONAL | SETUP  
   --  | ------ | ------ | ----------- | ------ 
   1   | lo    | loopback | carrier     | unmanaged 
   2   | eth0  | ether   | routable    | configured

   2 links listed.