

# Composing System Services in GuixSD

or how we came to wonder what a “system service”  
really is

FOSDEM, February 2017

A stylized lowercase letter 'v' composed of two overlapping, curved, ribbon-like shapes. The top-left shape is yellow and curves upwards and to the left. The bottom-right shape is orange and curves upwards and to the right. They meet at a central point, creating a sense of depth and movement.

**v** **GuixSD**

```
(operating-system
  (host-name "schememachine")
  (timezone "Europe/Brussels")
  (locale "fr_BE.utf8")
  (bootloader (grub-configuration (device "/dev/sda")))
  (file-systems (cons (file-system
                      (device "my-root")
                      (title 'label)
                      (mount-point "/")
                      (type "ext4"))
                    %base-file-systems))
  (users (cons (user-account
                (name "charlie")
                (group "users")
                (home-directory "/home/charlie")))
          %base-user-accounts))
  (services (cons* (dhcp-client-service)
                  (service openssh-service-type)
                  %base-services)))
```

```
(service openssh-service-type
  (openssh-configuration
    (x11-forwarding? #t)
    (permit-root-login 'without-password)))
```

```
(operating-system
  ;; ...
  (services (remove (lambda (service)
                      (eq? ntp-service-type
                            (service-kind service)))
                    %desktop-services)))
```

```
(define %my-services
  ;; My very own list of services.
  (modify-services %desktop-services
    (mingetty-service-type config =>
      (mingetty-configuration
        (inherit config)
        (motd (plain-file "motd"
                          "Howdy FOSDEM!")))))
    (upower-service-type config =>
      (upower-configuration
        (inherit config)
        (ignore-lid? #true)
        (percentage-critical 5.)))))
```

```
$ guix system build config.scm
```

```
...
```

```
$ guix system vm config.scm
```

```
...
```

```
$ guix system container config.scm
```

```
...
```

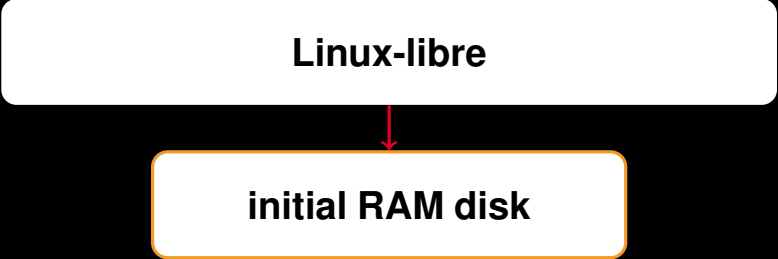
```
$ guix system reconfigure config.scm
```

```
...
```

**Linux-libre**



**Linux-libre**



```
graph TD; A[Linux-libre] --> B[initial RAM disk];
```

**initial RAM disk**

**Linux-libre**



**initial RAM disk**

**Guile**

**Linux-libre**

**initial RAM disk**

Guile

**PID 1: GNU Shepherd**  
services...

**Linux-libre**



**initial RAM disk**

Guile



**PID 1: GNU Shepherd**  
services...

Guile

**Linux-libre**

**initial RAM disk**

Guile

**PID 1: GNU Shepherd**  
services...

Guile

**applications**



```
;; Service definition for the GNU Shepherd (PID 1)
;; embedded in GuixSD.
```

```
(shepherd-service
 (provision '(mysql))
 (documentation "Run the MySQL server.")
 (start (let ((my.cnf (mysql-configuration-file config)))
          #~(make-forkexec-constructor
              (list (string-append # $mysql "/bin/mysqld")
                    (string-append "--defaults-file="
                                    # $my.cnf))
              #:user "mysql" #:group "mysql")))
 (stop #~(make-kill-destructor)))
```

;; Shepherd service to mount/unmount a file system.

```
(with-imported-modules '((gnu build file-systems))
  (shepherd-service
    (provision '(file-system-/home))
    (start #~(lambda ()
              (mount "/dev/foo" "/home" "ext4")))
    (stop #~(lambda ()
              (umount "/home")))))
```





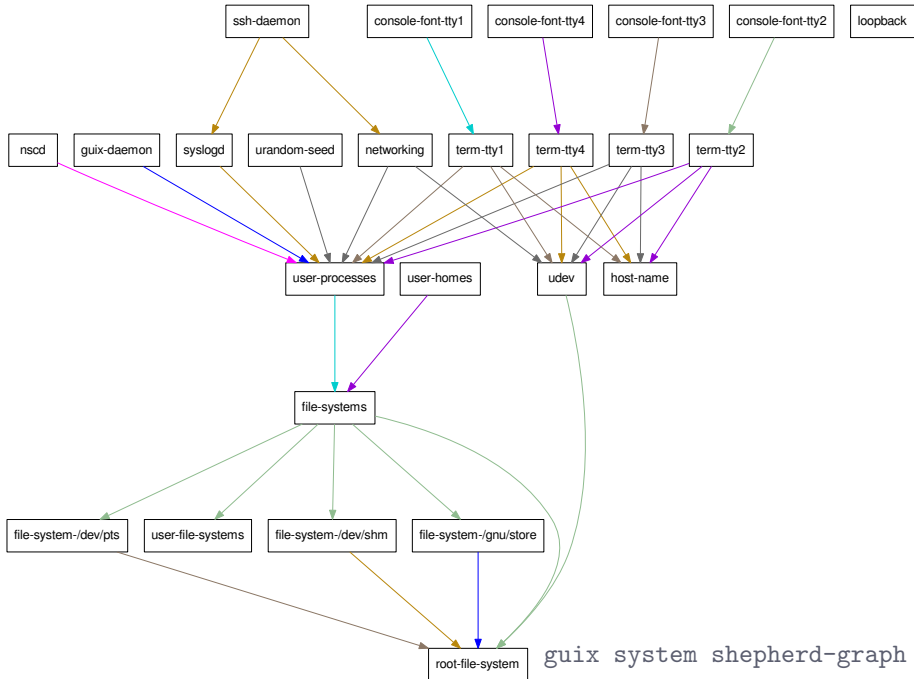
```
;; Shepherd service for the BitlBee IRC gateway daemon.  
;; Running in a container!
```

```
(with-imported-modules '(gnu build linux-container))  
  (shepherd-service  
    (provision '(bitlbee))  
    (requirement '(loopback))  
    (start #~(make-forkexec-constructor/container  
              (list #$(file-append bitlbee "/sbin/bitlbee")  
                    ...)))  
    (stop #~(make-kill-destructor))))
```



world  
première!

**Services, take #1.**



guix system shepherd-graph

```
(service
  (provision '(postgres))
  (requirement '(user-processes loopback))
  (start #~(make-forkexec-constructor #$postgresql ...))
  (stop #~(make-kill-destructor)))
```

```
(service
  (provision '(postgres))
  (requirement '(user-processes loopback))
  (start #~(make-forkexec-constructor #postgresql ...))
  (stop #~(make-kill-destructor))
  (activate #~(begin ...)))
```

```
(service
  (provision '(postgres))
  (requirement '(user-processes loopback))
  (start #~(make-forkexec-constructor #$postgresql ...))
  (stop #~(make-kill-destructor))
  (activate #~(begin ...))
  (user-groups (list (user-group
                      (name "postgres")
                      (system? #t))))
  (user-accounts (list (user-account
                        (name "postgres")
                        (group "postgres")
                        (system? #t)
                        (shell ...))))))
```

```
(service
  (provision '(postgres))
  (requirement '(user-processes loopback))
  (start #~(make-forkexec-constructor #$postgresql ...))
  (stop #~(make-kill-destructor))
  (activate #~(begin ...))
  (user-groups (list (user-group
                     (name "postgres")
                     (system? #t))))
  (user-accounts (list (user-account
                       (name "postgres")
                       (group "postgres")
                       (system? #t)
                       (shell ...))))))
```

```
(service
  (provision '(postgres))
  (requirement '(user-processes loopback))
  (start #~(make-forkexec-constructor #$postgresql ...))
  (stop #~(make-kill-destructor))
  (activate #~(begin ...))
  (user-groups (list (user-group
    (name "postgres")
    + /etc
    (system? #t))))
  (user-accounts (list (user-account
    (name "postgres")
    (group "postgres")
    (system? #t)
    (shell ...))))))
```





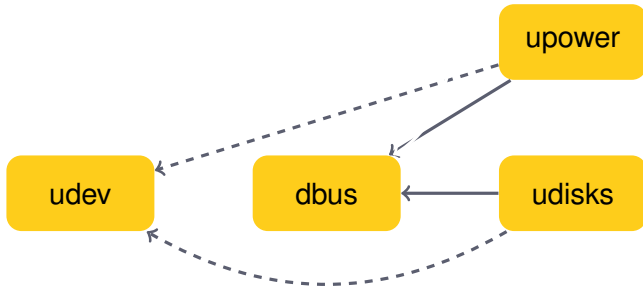
**freedesktop.org**

udev

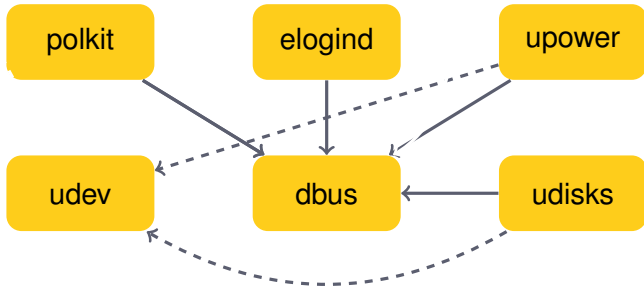
dbus



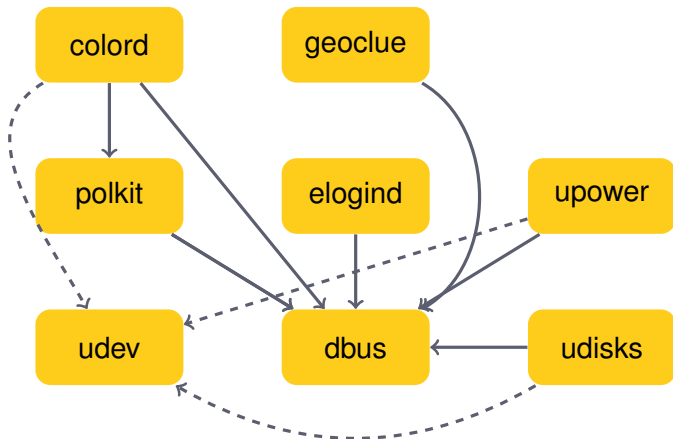
**freedesktop.org**



**freedesktop.org**



**freedesktop.org**



**freedesktop.org**



**Composable services.**

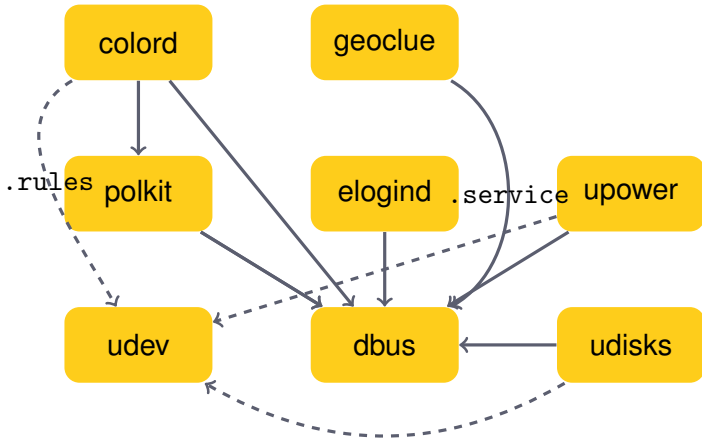
**Key insight:  
services “extend” each  
other.**

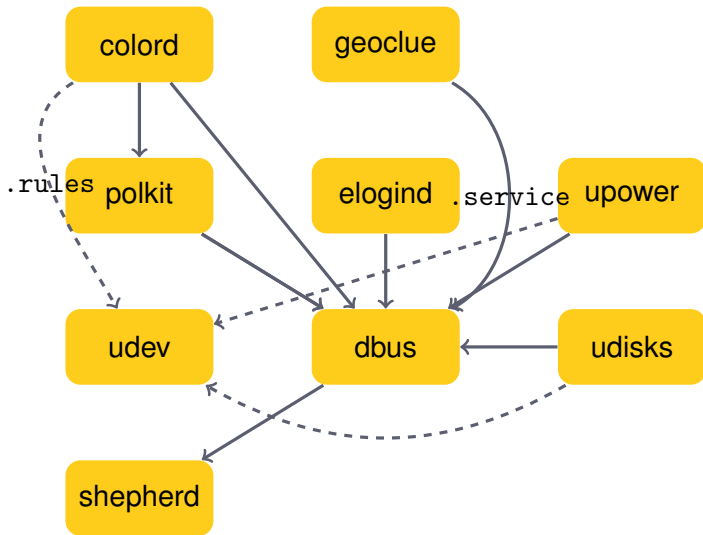


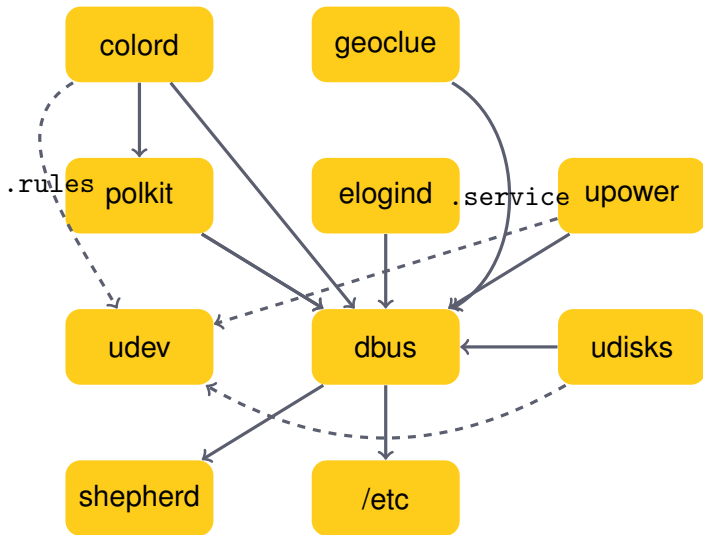
**Digression:  
NixOS configuration.**

```
{ config, lib, pkgs, ... }:
let
  cfg = config.services.openssh;
in {
  options = ...;

  config = mkIf cfg.enable {
    users.extraUsers.sshd = { isSystemUser = true; };
    environment.etc = authKeysFiles //
      { "ssh/moduli".source = cfg.moduliFile; };
    systemd.services.sshd-service =
      { wantedBy = "multi-user.target";
        # ...
      };
    security.pam.services.sshd =
      { startSession = true;
        unixAuth = cfg.passwordAuthentication;
      };
  }
}
```



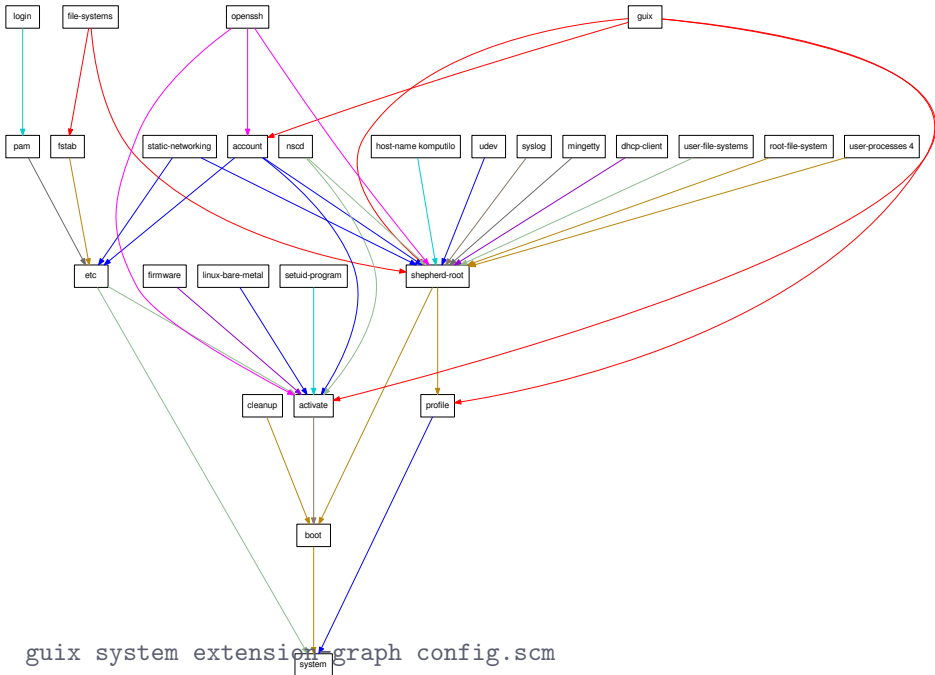




# what users type

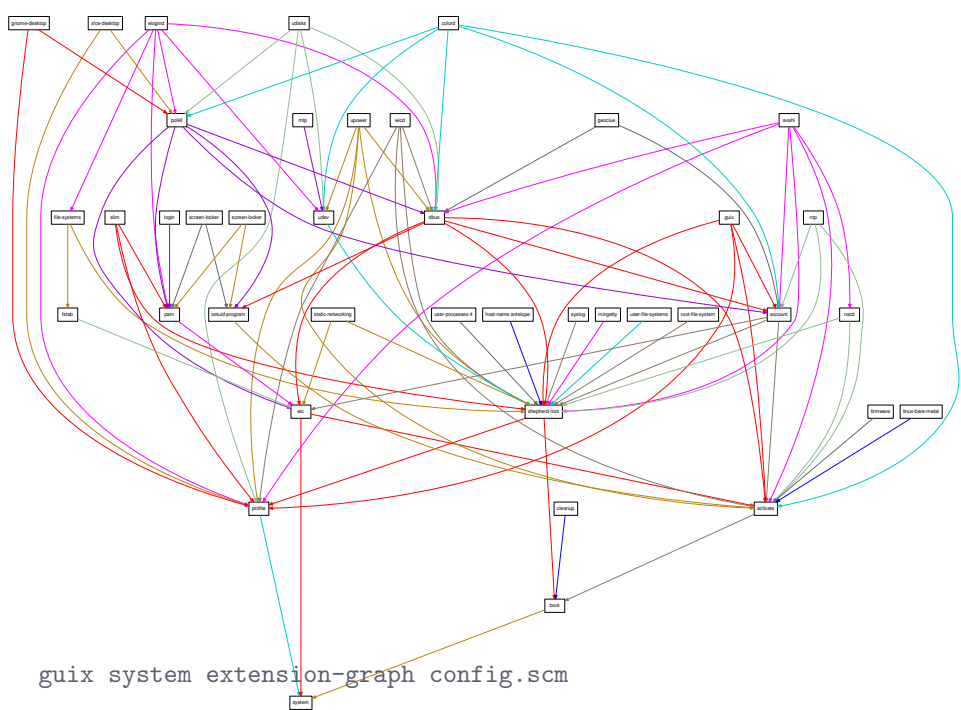
```
(operating-system
  (host-name "schememachine")
  ;; ...
  (services (cons* (dhcp-client-service)
                   (service openssh-service-type
                             (openssh-configuration
                              (x11-forwarding? #t)
                              (permit-root-login
                               'without-password)))
                 (service nginx-service-type ...)
                 %base-services)))
```

Services,  
service types.



guix system extension graph config.scm





fold-services.

Dear Haskell,  
this is a monoid!

**Wrap-up.**

**GuixSD leverages  
a holistic approach  
to system services.**

- ▶ services can **use and extend** PID 1
- ▶ “service extensions” capture ***all the service aspects***
- ▶ makes complex configurations **tractable**

- ▶ services can **use and extend** PID 1
- ▶ “service extensions” capture ***all the service aspects***
- ▶ makes complex configurations **tractable**
- ▶ **come up with your own services!**



`ludo@gnu.org`

`https://gnu.org/software/guix/`



Copyright © 2010, 2012–2017 Ludovic Courtès [ludo@gnu.org](mailto:ludo@gnu.org).

GNU GuixSD logo, CC-BY-SA 4.0, <http://gnu.org/s/guix/graphics>

Copyright of other images included in this document is held by their respective owners.

This work is licensed under the **Creative Commons Attribution-Share Alike 3.0** License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/3.0/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

At your option, you may instead copy, distribute and/or modify this document under the terms of the **GNU Free Documentation License, Version 1.3 or any later version** published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is available at <http://www.gnu.org/licenses/gfdl.html>.

The source of this document is available from <http://git.sv.gnu.org/cgiit/guix/maintenance.git>.