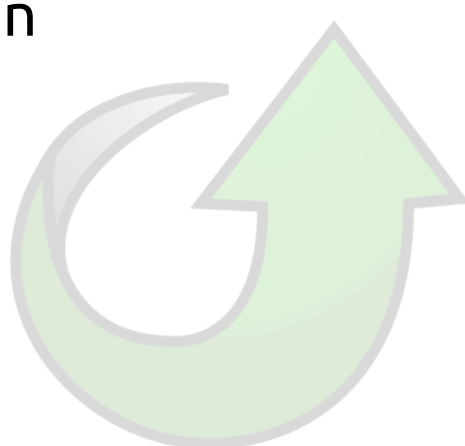
Battle-Hardened Upstart

Linux Plumbers 2013

James Hunt <james.hunt@ubuntu.com>
and
Dmitrijs Ledkovs <dmitrijs.ledkovs@ubuntu.com>

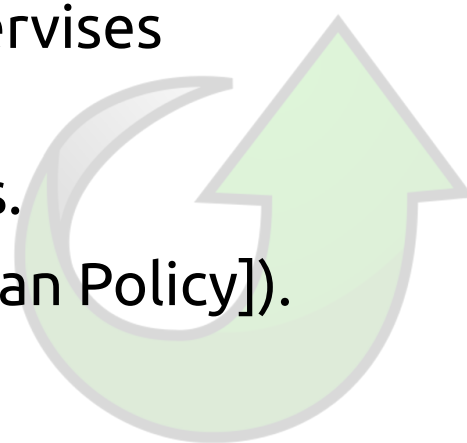
September, 2013

Table of Contents



Overview of Upstart

- Revolutionary event-based `/sbin/init` system.
- Written by Scott James Remnant (Canonical, Google).
- Maintained by Canonical.
- Developed by Canonical and the community.
- PID 1 on every Ubuntu system since 2006 (introduced in Ubuntu 6.10 "Edgy Eft").
- Systems booted using native Upstart jobs (not SysVinit compat) since Ubuntu 9.10 ("Karmic Koala") in 2009.
- Handles system boot and shutdown and supervises services.
- Provides legacy support for SystemV services.
- Upstart is a first-class citizen in Debian ([Debian Policy]).



Availability and Usage

- Runs on any modern Linux system.
- Used by...

ubuntu	openSUSE	redhat	amazonkindle	chrome
6.10 →	11.3/11.4	RHEL6		ChromeOS

- Now available in...

debian

Platform Presence

Upstart runs on all types of systems:

- Desktop systems
- Servers
- Embedded devices
- Thin clients (such as ChromeBooks, Edubuntu)
- Cloud instances
- Tablets
- Phones (Ubuntu Touch)

Cloud

- Upstart is the #1 init system used in the cloud (through Ubuntu).
- Ubuntu, and thus Upstart, is used by lots of large well-known companies such as:
  - HP
  - AT&T
  - Wikipedia
  - Ericsson
  - Rackspace
  - Instagram
  - twitpic …
- Companies moving to Ubuntu...
  - Netflix
  - Hulu
  - eBay

Versatility

- Upstart is simple and versatile
  - The `/sbin/init` daemon only knows about *events* and *processes*: it doesn't dictate runlevel policy.
  - So much of what follows is specific to Ubuntu and Debian as a result.

Design and Architecture

- Written in C.
- Learns from the Unix philosophy of *do one thing and do it well*:
  - PID 1 provides the event engine and service supervision capabilities.
  - PID 1 contains a D-Bus server which includes only services directly related to service management.
  - "Bridges" provide additional functionality "out-of-process".
    - Ensures overall system robustness against failures in non-core features.
- Event-based design.
- Makes heavy use of the powerful NIH utility library.

Design and Architecture (continued)

- Main binaries:
  - `/sbin/init` (daemon)
  - `/sbin/initctl` (control command)
- Declarative job syntax.
- Reads "job files" (`/etc/init/*.conf`).
- Supports override files to modify existing jobs (`/etc/init/*.override`).
- Filesystem mounting is handled by another helper called "`mountall`"
  - Mounts filesystems *in parallel* as the devices become available.
  - Emits a variety of events as these filesystems are mounted

See `init(8)`, `init(5)`, `mountall(8)`, `mounting(7)`, `mounted(7)`, `all-swaps(7)`, `local-filesystems(7)`, `remote-filesystems(7)`, `virtual-filesystems(7)` `filesystem(7)`.

Event-based Design

Upstart is *event-based*:

- Upstart starts services ("jobs") when their start conditions (specified as events) are fully satisfied.
- Jobs are therefore started *naturally* rather than...
  - Running than in sequential fashion (SystemV).
  - Having to resolve dependency ordering (Dependency-based init systems)

Job File Example

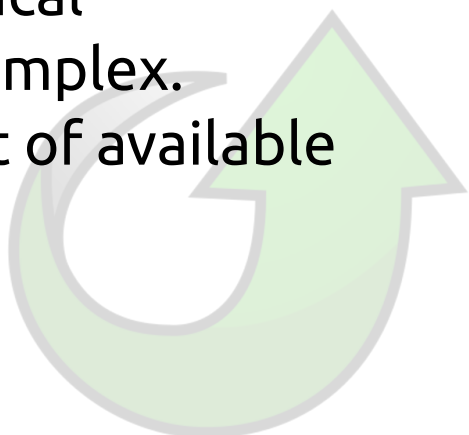
A simple job file:

```
start on started dbus
stop on runlevel [016]

exec mydaemon --foo=bar
expect daemon
```

This job will:

- Start once the `dbus` job has started.
- Execute `mydaemon --foo=bar`.
- Expect the daemon to double-fork.
- Stop on shutdown, single-user mode, or reboot.

The "`start on`" and "`stop on`" conditions support logical operators and bracketing and can be arbitrarily complex. See `started(7)`, `runlevel(7)` and `init(5)` for the full list of available job stanzas.



Job File Example: Details

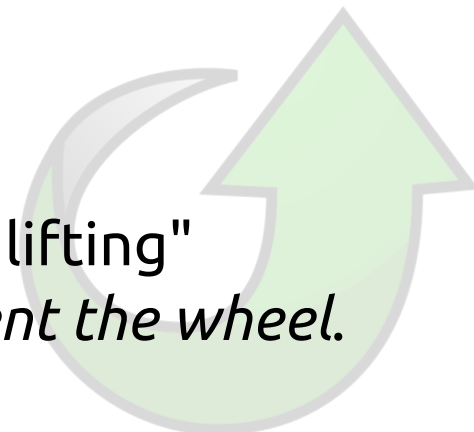
```
start on started dbus
stop on runlevel [016]

exec mydaemon --foo=bar
expect daemon
```

The 4 lines of configuration...

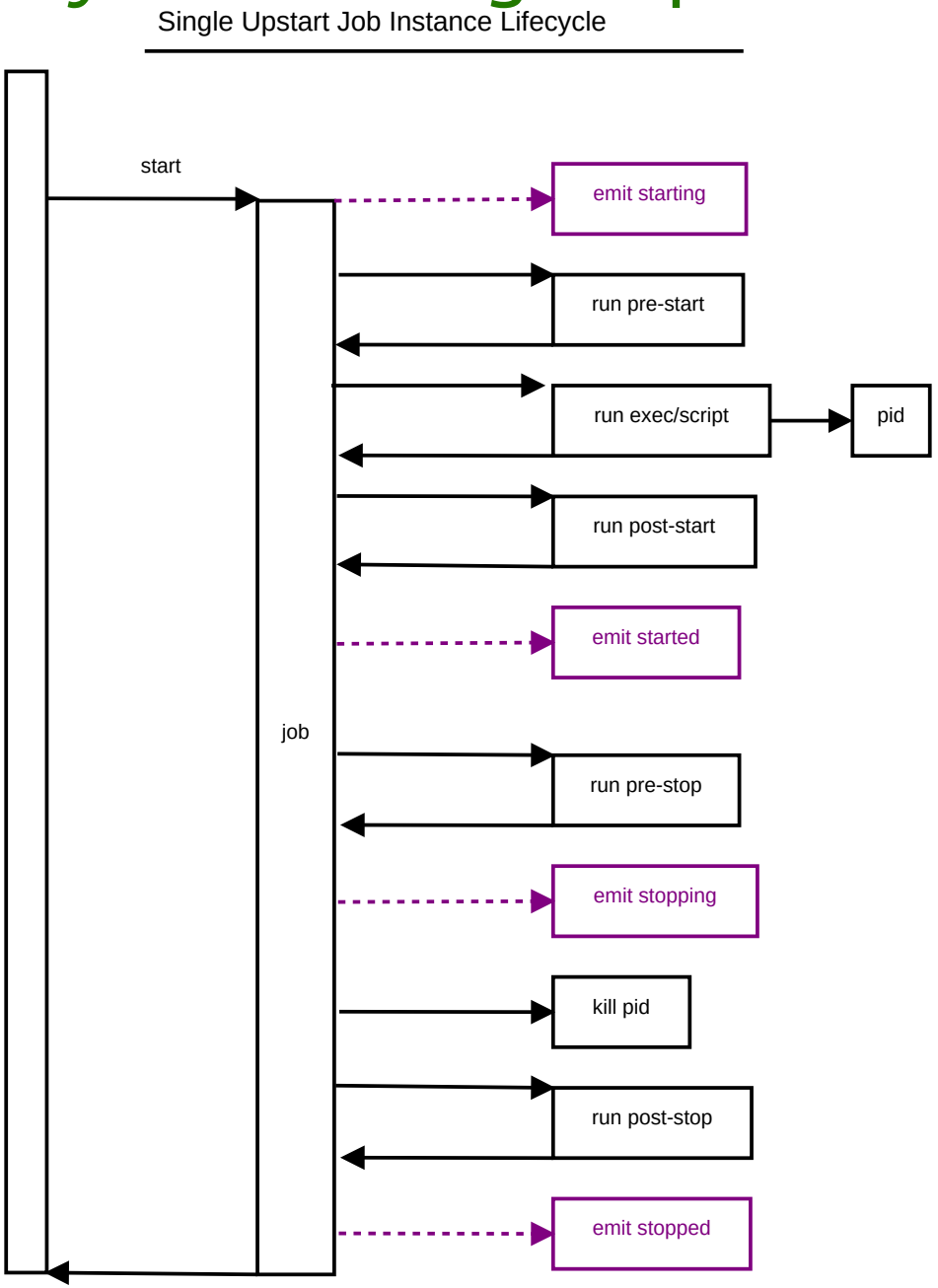
- Tell Upstart when the job should be started and stopped.
- Allow an admin to manually control the job's execution.
- Provide the job with automatic logging.
- Provide the job with a minimal environment.

Note:

- The job is simple and human-readable.
- Unlike SystemV init, Upstart does the "heavy lifting" avoiding the need for every service to *re-invent the wheel*.



Lifecycle of a Single Upstart Job Instance



SystemV Support

Upstart supports SystemV services:

- Supports running of SysV services (`/etc/init.d/*`) without modification.
- Provides the usual SysV commands (`shutdown`, `reboot`, `telinit`, `runlevel`, ...).
- Emulates SysV runlevels using events.

However, converting SysV services to Upstart jobs allows Upstart to do the "heavy lifting" and allows services to run as soon as possible on boot.

SystemV Runlevels

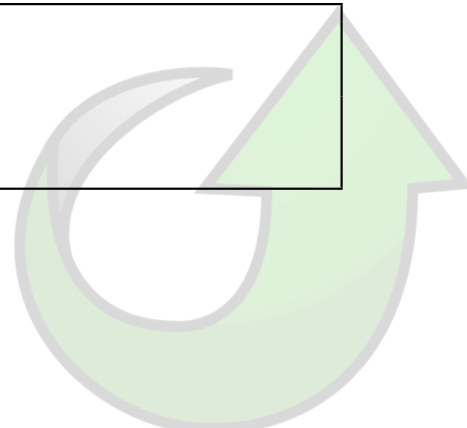
Handled elegantly: *the* `/sbin/init` *daemon knows nothing about runlevels!*

1. `rc-sysinit` job initiates runlevel changes simply by calling `telinit` at the appropriate point:

```
start on (filesystem and static-network-up)
exec telinit "${DEFAULT_RUNLEVEL}"
```

2. `telinit` emits the `runlevel` event.
3. `rc` job runs the SystemV service scripts:

```
start on runlevel [0123456]
exec /etc/init.d/rc $RUNLEVEL
```



See `runlevel(7)`.

Upstart Bridges

Upstart functionality is extended with "bridges" which proxy events from other parts of the system into Upstart. Such bridges:

- Run out-of-process for safety and to minimise PID 1 complexity.
- Inject events by communicating with Upstart via D-Bus.
- Run as Jobs: if they die, they get auto-restarted!
- Examples of some current bridges:
  - **`upstart-udev-bridge`**:
    Converts kernel udev events into Upstart events.
  - **`upstart-socket-bridge`**:
    Provides "socket activation" to start services "on demand" (not used for a variety of reasons).
  - **`upstart-file-bridge`**:
    Exposes inotify file events as Upstart events.
  - **`upstart-dbus-bridge`**:
    Converts D-Bus signals into Upstart events.

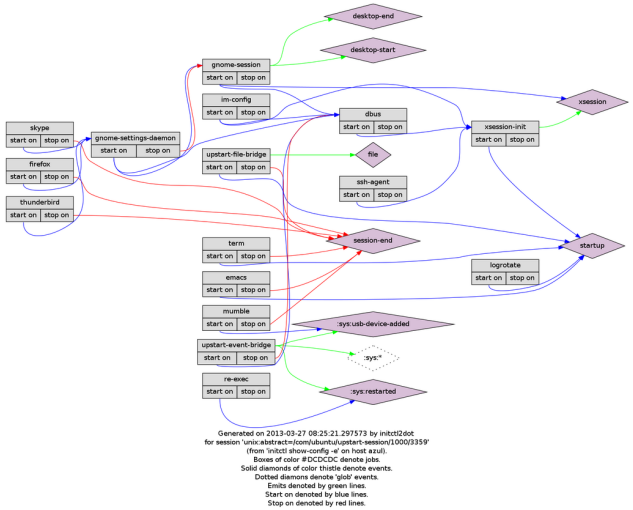
More Events

- The Debian package provides `upstart-events(7)`: detailed man page listing all "well-known" events, what emits them, when they are emitted, *et cetera*.
- If Upstart and existing bridges don't provide the events you want...
  - Talk to us!
  - Submit a new bridge to Upstart (skeleton is 300 lines).
  - Write a program using `libupstart`.
  - Just call "`initctl emit $event`"!

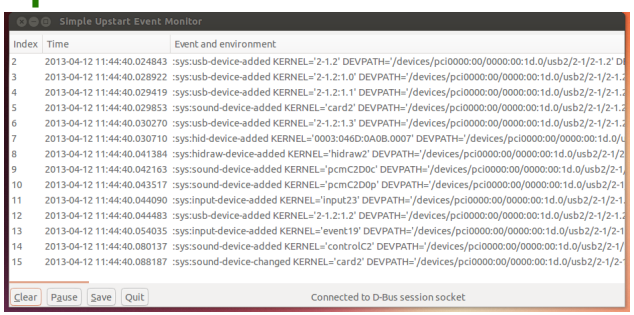
Upstart Utilities

Upstart provides a small set of utilities:

- **`initctl2dot`**: Produce GraphViz diagrams of events.



- **`upstart-monitor`**: Command-line and GUI event monitor.



- **`init-checkconf`**: Tool to syntax-check a job file.

```
$ init-checkconf /tmp/procenv.conf
ERROR: File /tmp/procenv.conf: syntax invalid:
init:procenv.conf:1: Unknown stanza
```

cloud-init

- Developed by Scott Moser (Canonical) to provision cloud guests.
- Provides an elegant solution to the problem of how to configure a newly-created (generic) cloud guest.
  - Such systems are extremely pared down and initially even disallow logins!
  - Each guest may need to be configured differently (web server, RDBMS, proxy, ...)
- `cloud-init` is like an amalgam of "preseed"/"kickstart", a first-boot facility and CFEngine/Puppet.

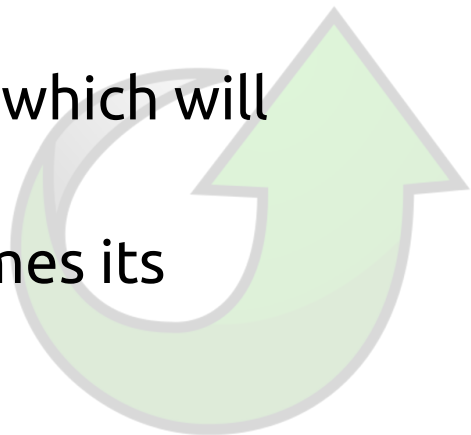
cloud-init: How it works

- `cloud-init` works by installing a handful of Upstart jobs that run in early boot:

```
start on mounted MOUNTPOINT=/
task
exec /usr/bin/cloud-init init
```

  - Note that `mounted` is a *blocking event* (no further jobs will run until `cloud-init` has finished!)

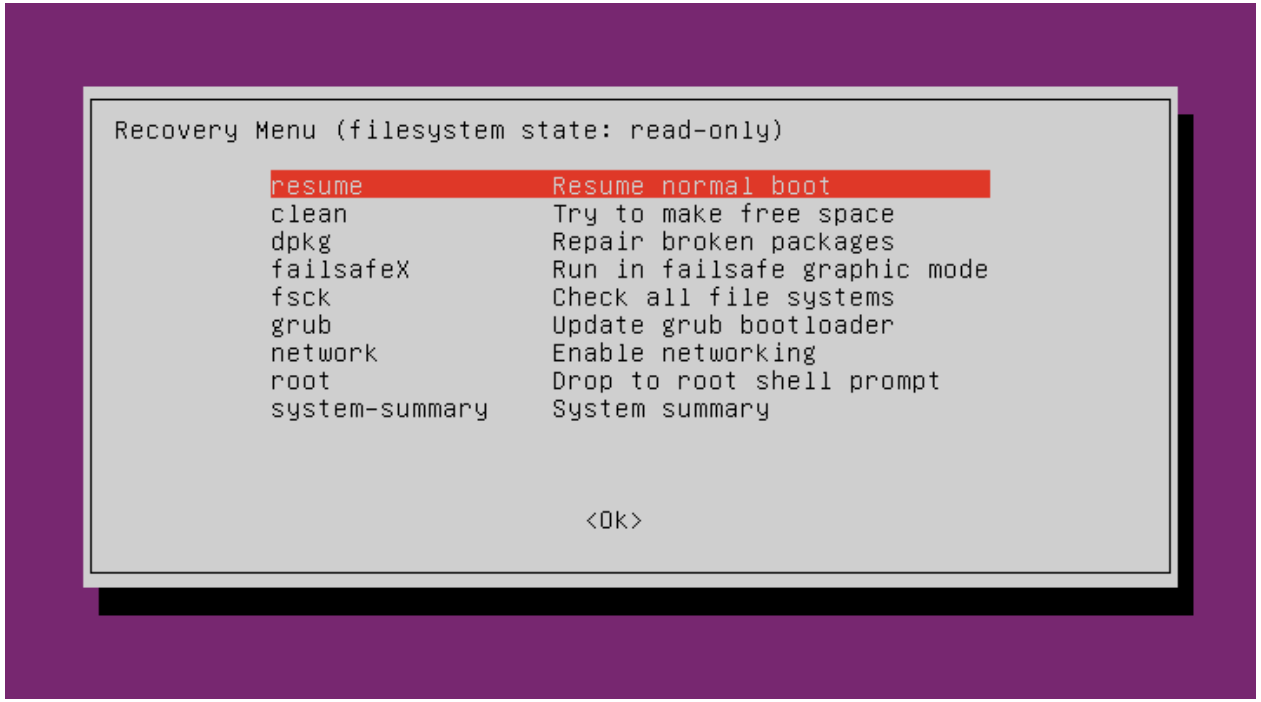
- `cloud-init` then:
  - Sets hostname.
  - Sets up ssh securely.
  - Installs and configures all required services.
- Note that `cloud-init` calls `apt-get update+upgrade` which will potentially upgrade Upstart itself *mid-boot!*
- Once `cloud-init` has finished, the system resumes its normal boot path.



See `mounted(7)` and `upstart-events(7)`.

friendly-recovery

Ubuntu provides a facility called "friendly-recovery":



- Menu-based utility.
- Allow any user to fix a broken system... easily.
- Makes clever use of Upstart to subvert system boot *temporarily*.

friendly-recovery: How it works

If the user selects the `recovery` option at the Grub prompt:

- The initramfs starts Upstart like this:

```
init --startup-event=recovery
```

- Upstart emits an event called "`recovery`".
- Upstart then starts the `friendly-recovery` job which specifies

```
start on recovery
```

- The recovery job starts the menu-based `friendly-recovery` application.
- The user can then fix various parts of their system.
- Once `friendly-recovery` exits, the recovery job runs:

```
post-stop script
    initctl emit startup
end script
```

- The system will then continue booting as normal!

gpg-key-compose

- In Ubuntu 13.10, Upstart is also used to manage graphical login sessions
- This means you get a consistent view of system events all the way up to the user level, and can easily create jobs as a user that take action on, e.g., plugging in a USB stick.

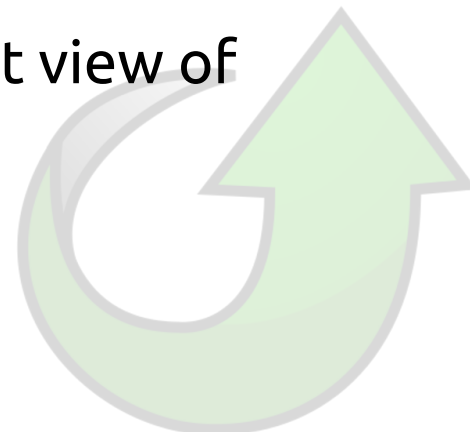
```
start on :sys:block-device-added KERNEL=sd*1 ID_SERIAL=_Patriot_Memory_07BC1D01B2F98B2A-0:0
stop on :sys:block-device-removed KERNEL=sd*1 ID_SERIAL=_Patriot_Memory_07BC1D01B2F98B2A-0:0

pre-start script
        # wait 5 seconds for udisks to mount the disk before we try to
        # reassemble; ideally we would be able to just listen for a mounting
        # event but udisks doesn't emit those for us currently.
        sleep 5
        gfcombine -o $XDG_RUNTIME_DIR/secring.gpg \
                /media/vorlon/PATRIOT/secring.gpg.209 \
                ~/.gnupg/secring.gpg.051
end script

post-stop script
        # don't leave this behind where it can be stolen
        rm -f $XDG_RUNTIME_DIR/secring.gpg
end script
```

- Can be keyed on any udev event, and any attributes of that event (use **`upstart-monitor`** to see what's available)

Enablements - Summary

The facilities outlined in this section are made possible by:

- The ability for an Upstart job to "hook" into the boot process *at any point*.
- The ability for an Upstart job to "pause" the emission of a blocking event until that job has run.
- The fact that the entire boot pivots upon the "`startup`" event, and that Upstart allows this initial event to be overridden (`friendly-recovery`).
- (optionally) The ability to provide a consistent view of events all the way up to the desktop.



Quality Checks

- Code is heavily commented and cleanly designed.
- All changes peer-reviewed.
- Source tree built and all tests run *atleast daily*.
- Source tree checked regularly with `smatch`, `scan-build` (LLVM) and Coverity Scan.
- All changes require unit-tests (where appropriate).
- Upstart provides a comprehensive set of both unit tests and integration tests.
- Upstart and `libnih` are extremely well tested:

Project	Test Type	Number
Upstart	unit	1291
Upstart	integration scenarios	7[1]
`libnih`	unit	2864

[1] More currently in development.

Kernel and Plumbing pain-points and Ideas

Console devices

- Multiple devices can be specified.
  Many users specify multiple `console=` parameters on kernel command-line.
  If `>1 console` values, when `init` opens `/dev/console`, output only goes to *last* `console` specified.
- Possible to create/destroy console devices dynamically?
  Could output be multiplexed to all/some using an `ioctl`?

Kernel and Plumbing pain-points and Ideas

Network layer notifications

- expect `listen(2)`
  For a database job, the best indication of "readiness" is when it calls `listen(2)`. Providing an API to allow applications to register interest in a socket state change would solve this problem (and introduce some parity since it's possible for filesystem events and (will soon) be possible for `AF_DBUS` names).

Kernel and Plumbing pain-points and Ideas

Non-unified filesystems features

- `d_type` not available on all filesystems (XFS in-progress).
- `inotify` not reliable with recursive watches.
- …not implemented on all filesystems.
- …or doesn't report all events.
- …or doesn't report anything (e.g. R/O overlayfs base)[2]

[2] https://launchpad.net/bugs/882147.

Links

- Edgy Eft Release Notes
  http://www.ubuntu.com/news/610released
- Upstart Home
  http://upstart.ubuntu.com
- Upstart Cookbook
  http://upstart.ubuntu.com/cookbook/
- Upstart Code
  http://launchpad.net/upstart
- "The Bible"
  `man 5 init`
- NIH Code
  https://github.com/keybuk/libnih
- Upstart internal objects diagram
  http://people.canonical.com/~jhunt/upstart/devel/upstart_objects.dia

References

http://www.debian.org/doc/debian-policy/ch-opersys.html#s-upstart

THE END.