



Systemd  
de /dev/null à root


**I'M SORRY**

**FOR ALL THE MEMES**









Most impressive.





**DEVOPS**

**DEVOPS EVERYWHERE**

**YOU BUILD IT**



**YOU RUN IT**







**YAML**

**YAML EVRYWHERE**





**HELP!**

SYSTEM  
ADMINS



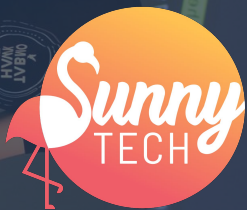








**PLEASE HELP ME**



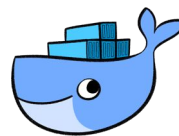
@fteychene

Saagie®




François Teychené

Cloud developer Saagie®



N'a pas peur d'utiliser le terme Monad

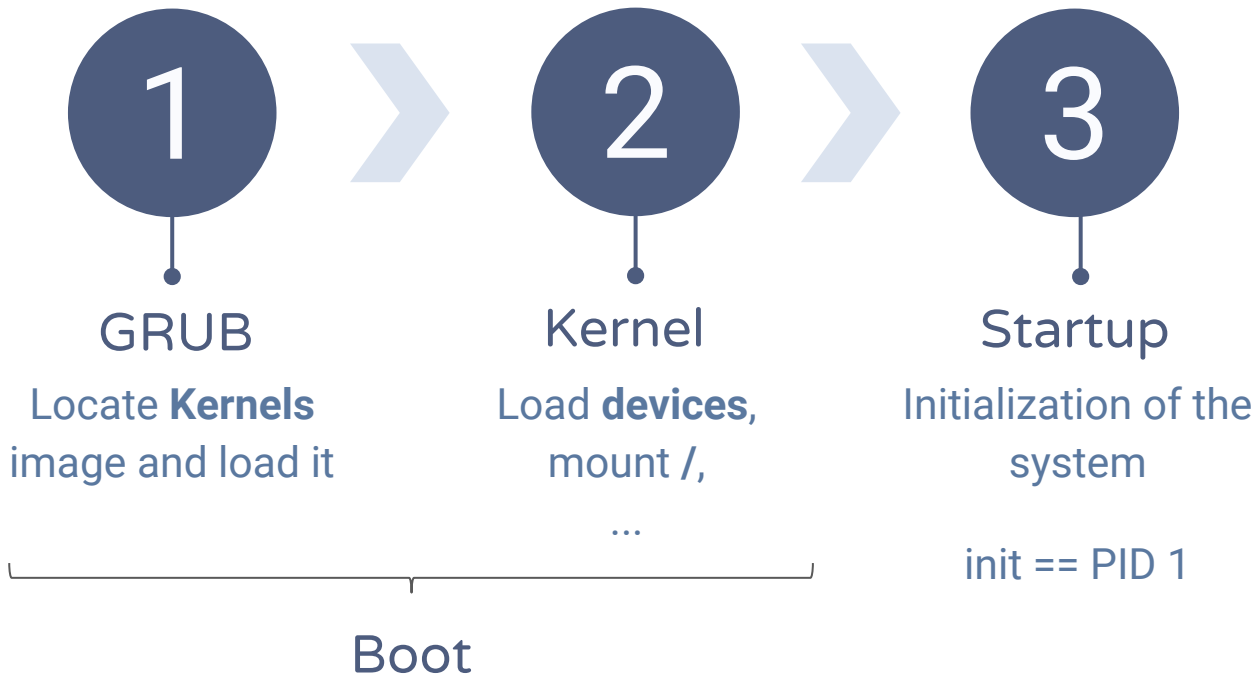




Delegate execution  
to the system

# PID 1 & System boot

---



Startup

---

systemd

vs



Choose your hero

---



# Startup

---



VS

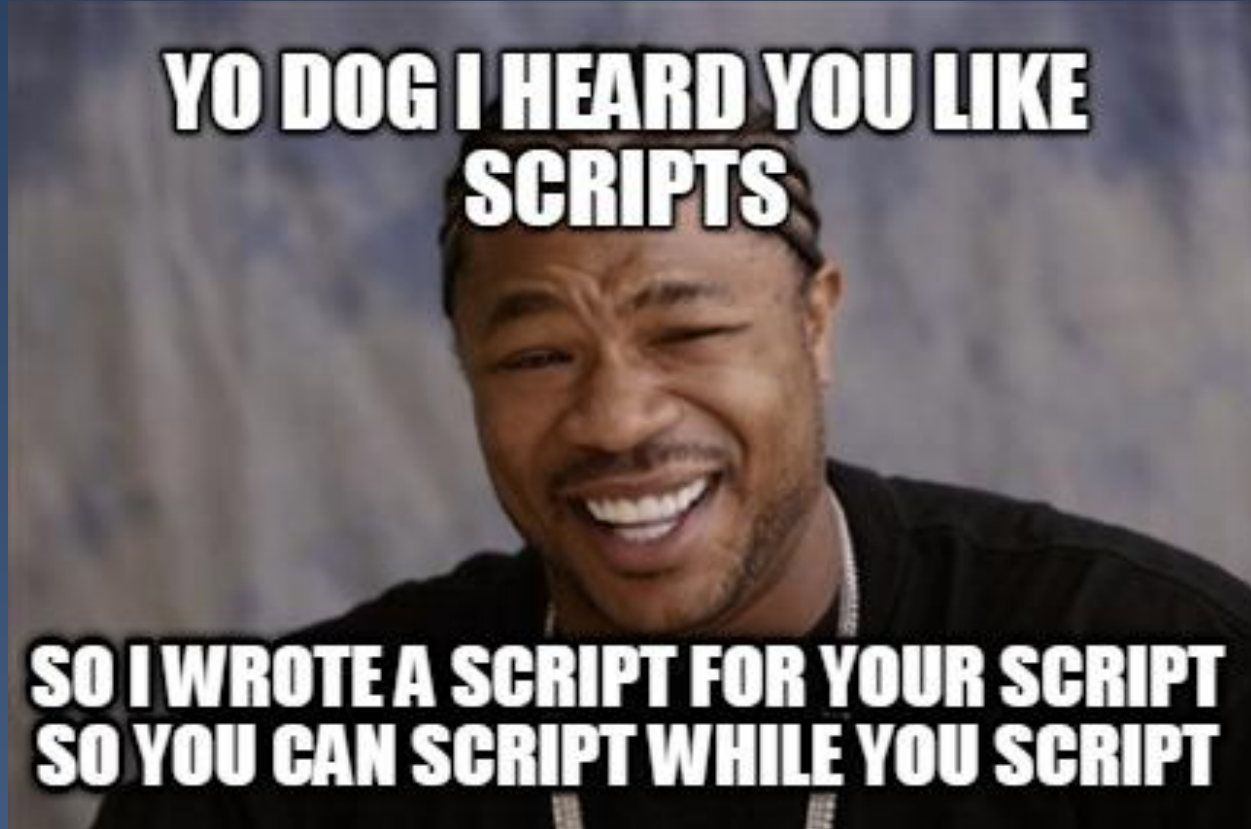
INITIATED FOREVER





# Scripts & “convention”

---



A man with dark, curly hair and a goatee, wearing a dark jacket over a dark shirt and a white t-shirt, has a wide-eyed, shocked expression. He is looking directly at the camera. The background is a dark, out-of-focus interior with a white frame on the left.

**and??**

Everything is manual !

... IN SHELL !!

.. AS ROOT !!!

... EVEN PARSING START !!!!





# Exemple : Gitlab startup\*

\*Gitlab init.d script - <https://gitlab.com/gitlab-org/gitlab-ce/blob/master/lib/support/init.d/gitlab>



Systemd

# Infos

---



## Releases

30 march 2010

Current Version : 239

Compiled binary



## Integrations

Ubuntu 15.04

CentOs 7

Debian 8

A young woman with long brown hair and a wide-eyed, toothy grin is the central focus. She is wearing a light blue t-shirt. The background is a crowded room with many people sitting at tables, suggesting a social gathering or party. The lighting is bright and indoor.

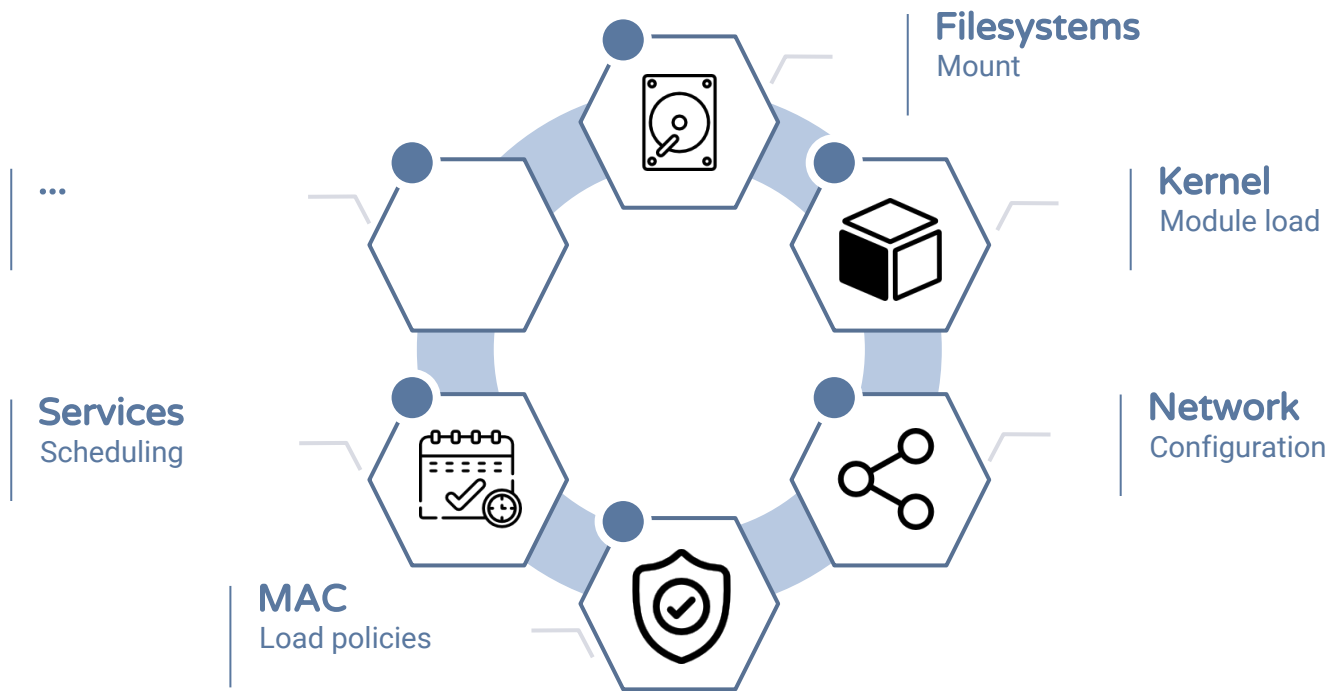
**GOOD**

**NOW YOU HAVE NOWHERE TO  
HIDE**

memegenerator.net

# What does it do

---



# Configuration over scripting

# Units

---

**Service**  
Execution unit



**Target**  
Group units



**Socket**  
Socket activation



**Path**  
Path activation



**Timer**  
Time activation



...  
device, mount, automount, swap  
snapshot, slice, scope



Service



# Service

---

```
[Unit]
Description="Graphql proxy"
Requires=redis.service
Wants=notify-failed.service
After=redis.service x-install.target
OnFailure=notify-failed@%N

[Service]
Environment=JAVA_HOME=/usr/lib/java8
Environment=ENVIRONMENT=prod

WorkingDirectory=/var/graphql-proxy
Restart=on-failure
RestartSec=4

User=graphql
Group=applications

ExecStart=/var/graphql-proxy/bin/graphql-proxy \
  -Dconfig.file=/var/graphql-proxy/resources/application.prod.conf \
  -Dlogger.file=/var/graphql-proxy/resources/logback.prod.xml \
  -J-javaagent:/usr/local/bin/jolokia.jar=port=7777,host=0.0.0.0
```

# Dependencies

---

[Unit]

Description="Graphql proxy"

Requires=redis.service

Wants=notify-failed.service

After=redis.service x-install.target

OnFailure=notify-failed@%N

[Service]

Environment=JAVA\_HOME=/usr/lib/java8

Environment=ENVIRONMENT=prod

WorkingDirectory=/var/graphql-proxy

Restart=on-failure

RestartSec=4

User=graphql

Group=applications

ExecStart=/var/graphql-proxy/bin/graphql-proxy \

-Dconfig.file=/var/graphql-proxy/resources/application.prod.conf \

-Dlogger.file=/var/graphql-proxy/resources/logback.prod.xml \

-J-javaagent:/usr/local/bin/jolokia.jar=port=7777,host=0.0.0.0

# Configuration

---

```
[Unit]
Description="GraphQL proxy"
Requires=redis.service
Wants=notify-failed.service
After=redis.service x-install.target
OnFailure=notify-failed@%N

[Service]
Environment=JAVA_HOME=/usr/lib/java8
Environment=ENVIRONMENT=prod

WorkingDirectory=/var/graphql-proxy
Restart=on-failure
RestartSec=4

User=graphql
Group=applications

ExecStart=/var/graphql-proxy/bin/graphql-proxy \
  -Dconfig.file=/var/graphql-proxy/resources/application.prod.conf \
  -Dlogger.file=/var/graphql-proxy/resources/logback.prod.xml \
  -J-javaagent:/usr/local/bin/jolokia.jar=port=7777,host=0.0.0.0
```

# Failure management

---

```
[Unit]
Description="GraphQL proxy"
Requires=redis.service
Wants=notify-failed.service
After=redis.service x-install.target
OnFailure=notify-failed@%N
```

```
[Service]
Environment=JAVA_HOME=/usr/lib/java8
Environment=ENVIRONMENT=prod
```

```
WorkingDirectory=/var/graphql-proxy
Restart=on-failure
RestartSec=4
```

```
User=graphql
Group=applications
```

```
ExecStart=/var/graphql-proxy/bin/graphql-proxy \
  -Dconfig.file=/var/graphql-proxy/resources/application.prod.conf \
  -Dlogger.file=/var/graphql-proxy/resources/logback.prod.xml \
  -J-javaagent:/usr/local/bin/jolokia.jar=port=7777,host=0.0.0.0
```

```
RestartSec=...
TimeoutStartSec=...
TimeoutStopSec=...
RestartPreventExitStatus=...
RestartForceExitStatus=...
```

# Execution

---

```
[Unit]
Description="GraphQL proxy"
Requires=redis.service
Wants=notify-failed.service
After=redis.service x-install.target
OnFailure=notify-failed@%N
```

```
[Service]
Environment=JAVA_HOME=/usr/lib/java8
Environment=ENVIRONMENT=prod
```

```
WorkingDirectory=/var/graphql-proxy
Restart=on-failure
RestartSec=4
```

```
User=graphql
Group=applications
```

```
ExecStart=/var/graphql-proxy/bin/graphql-proxy \
  -Dconfig.file=/var/graphql-proxy/resources/application.prod.conf \
  -Dlogger.file=/var/graphql-proxy/resources/logback.prod.xml \
  -J-javaagent:/usr/local/bin/jolokia.jar=port=7777,host=0.0.0.0
```

```
ExecStartPre=...
ExecStartPost=...
ExecReload=...
ExecStop=...
ExecStopPost=...
PIDFile=...
```

# Moar dependencies

---



## *unit.wants* folder

Symbolic link to other unit =  
Wants configuration

Also exist for Requires



# Base commands

---

```
systemctl --all
```

```
systemctl --all -t service
```

```
systemctl start ...
```

```
systemctl status ...
```

```
systemctl stop ...
```

# Enable/Disable

---

```
systemctl enable # Activate on boot
```

```
systemctl disable # Disable on boot
```

```
systemctl mask # Deactivate unit in systemd
```

```
systemctl unmask # Activate unit on systemd
```



A man with dark, curly hair and a beard is wearing a gold crown. He is looking down, and his face is partially obscured by a white rectangular box containing text. The background is a plain, light gray wall.

```
systemctl daemon-reload
```

In memory configuration



Only daemons?

# No, of course

---

simple (default)

forking

oneshot

dbus

notify

idle



# Install service

---



**/lib/systemd/system**

Server service installation

Packages default installation



**/etc/systemd/system**

Override by system administrator

Override lib installation

# Partial override

---

*/lib/systemd/system/nginx.service*

```
[Unit]
Description=A high performance web server and a reverse proxy server
Documentation=man:nginx(8)
After=network.target

[Service]
Type=forking
PIDFile=/run/nginx.pid
ExecStartPre=/usr/sbin/nginx -t -q -g 'daemon on; master_process on;'
ExecStart=/usr/sbin/nginx -g 'daemon on; master_process on;'
ExecReload=/usr/sbin/nginx -g 'daemon on; master_process on;' -s reload
ExecStop=-/sbin/start-stop-daemon --quiet --stop --retry QUIT/5 --pidfile /run/nginx.pid
TimeoutStopSec=5
KillMode=mixed

[Install]
WantedBy=multi-user.target
```

*/etc/systemd/system/nginx.service*

```
[Service]
ExecStart=/usr/sbin/nginx -c /var/test.conf -g 'daemon on; master_process on;'
```



**FUUUUSION... HA!**

# Why use systemd

---



## Standard

Based on **standard**  
and not only on good  
behavior



## Declarative

Configuration by  
**configuration files**  
that can be validated



## Portable

Systemd  
configuration files  
can be deployed on  
all **distro**



## Extensible

Override by  
configuration and  
positioning

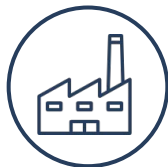
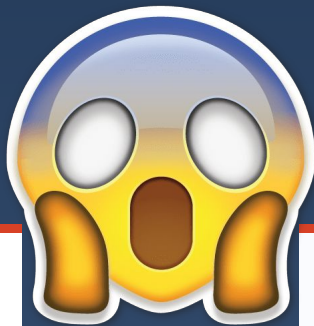


**MORE!**



# Systemd for user

---



## System

Setup system

Global



## User

Instance for user

Scoping for user

`~/.local/share/systemd/user`

`~/.config/systemd/user`

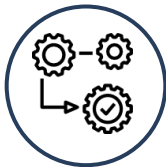
`systemctl --user ...`



Templates !

# Templating

---



**/run/systemd/system**

Runtime generated units



**xxx@.service**

Instance configuration

%i / %l specifiers

*start vpn@saagie.service*

# Logging

---

```
journalctl # All logs
```

```
journalctl -u nginx.service # Specific unit logs
```

```
journalctl --since yesterday # Logs since yesterday
```

```
journalctl -f -u nginx.service # Live logs
```

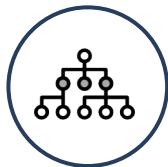
```
journalctl /usr/bin/dbus-daemon # Logs from a specific binary
```



Other units

# Target

---



## Group units

Manage a group of unit as one

Easy dependency management

```
/etc/systemd/system/enterprise.target
```

```
[Unit]
Description=My enterprise boot target
Requires=multi-user.target
Wants=custom-configuration.service
Conflicts=rescue.service rescue.target
After=multi-user.target rescue.service rescue.target
```

```
/etc/systemd/system/enterprise.target.wants/:
total 8.0K
drwxr-xr-x  2 root root 4.0K Nov  4 21:20 .
drwxr-xr-x 15 root root 4.0K Nov  4 21:18 ..
lrwxrwxrwx  1 root root   33 Nov  4 21:20 nginx.service -> /etc/systemd/system/nginx.service
```

# Path

---



## Activate on path

Activate on filesystem notify

React on configuration, ...

*/etc/systemd/system/nginx-reload.path*

```
[Unit]
Description=Nginx configuration reload
[Path]
PathModified=/etc/nginx/sites-enabled
```

*/etc/systemd/system/nginx-reload.service*

```
[Unit]
Description=Nginx configuration reload
[Service]
ExecStart=/usr/sbin/nginx -s reload
```

# Socket

---



## Activate on socket

Activate on socket call

*/etc/systemd/system/metrics.socket*

```
[Unit]
Description=Streaming service activation on socket

[Socket]
ListenStream=/run/metrics.sk

[Install]
WantedBy=sockets.target
```





**WELCOME TO OUR GROUP**

**LEAVING IS NOT AN OPTION**

# Slices

## CGroups

Tag unit in Cgroups

Resource limitation (Maybe)

```
control group /:
- .slice
  - user.slice
    - user-1000.slice
      - session-15.scope
        - 14233 sshd: vagrant [priv]
        - 14310 sshd: vagrant@pts/0
        - 14311 .bash
        - 14319 systemd-cgls
        - 14320 systemd-cgls
      - user@1000.service
        - init.scope
          - 14235 /lib/systemd/systemd --user
          - 14236 (sd-pam)
    - init.scope
      - 1 /sbin/init
    - system.slice
      - lvm2-lvmetad.service
        - 433 /sbin/lvmetad -f
      - lxcfs.service
        - 668 /usr/bin/lxcfs /var/lib/lxcfs/
      - snapd.service
        - 662 /usr/lib/snapd/snapd
      - iscsid.service
        - 833 /sbin/iscsid
        - 834 /sbin/iscsid
      - dbus.service
        - 624 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
      - accounts-daemon.service
        - 618 /usr/lib/accounts-service/accounts-daemon
      - vboxadd-service.service
        - 981 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-service.sh
      - ssh.service
        - 965 /usr/sbin/sshd -D
      - system-getty.slice
        - getty@tty1.service
          - 845 /sbin/agetty -o -p -- \u --noclear tty1 linux
      - networkd-dispatcher.service
```

device, mount, automount, swap,  
snapshot, scope



Systemd-\*



# systemd-analyze

Analyse & debug system manager

# systemd-analyse

---

```
systemd-analyze time # kernel & user space boot time
```

```
systemd-analyze blame # prints all running units ordered by init time
```

```
systemd-analyze dot # generate unit dependency graph
```

```
systemd-analyze verify # load & check a unit
```

# systemd-cgls

Show control group content



# systemd-cgtop

Top control group for their resource usage

# systemd-cat

Send output to journald

# systemd-run

Run programs in transient scope units

systemd-ask-password, systemd-delta, systemd-detect-virt,  
systemd-escape, systemd-hwdb, systemd-inhibit,  
systemd-machine-id-setup, systemd-notify, systemd-path,  
systemd-resolve, systemd-stdio-bridge, systemd-tmpfiles, ...



# systemd-nspawn

Run containers



# Systemd FTW





More to discover

# Devoxx university

---

DEVOXX France

7ème édition - 18 au 20 avril 2018, Paris



## SystemD c'est quoi ?

Remplacement par défaut d'init sur Linux  
Il permet de gérer les services et la configuration  
des services Linux.



#DevoxxFR

man



# Thanks

Saagie®



| @fteychene