

# Systemd for fun & profits en BUT

Jean-Marc Pouchoulon

Avril 2024

## 1 Compétences à acquérir lors du TP.

### Compétence principale:

- Utiliser systemd afin de gérer un serveur Linux.

### Niveaux dans la compétence:

- Niveau 1 utilisateur:
  - Concepts et utilisation basique de Systemd (exemple d'un service "ping").
  - Gestion d'une unité de service (ssh, systemd-networkd, systemd-resolved).
  - Manipulations avec systemd-runtime.
  - Gestion de systemd.
- Niveau 2 fonctionnalités avancées:
  - Activation d'une unité de service par DropIn.
  - Templating avec systemd.
  - Gestion de systemd au niveau de l'utilisateur.

### Savoirs liés:

- Processus.
- Cgroups.
- Capabilities.
- Services & units.

### Savoir-faire:

- Créer, lancer, auditer un service avec systemd.
- Utiliser systemd en tant qu'administrateur systèmes.
- Surcharger un service.
- Analyser le comportement d'un système au travers de systemd.
- Analyser le boot d'un ordinateur.
- Utiliser un DropIn pour réveiller un service.
- Utiliser un template systemd pour démarrer des applications versionnées.
- Limiter les ressources consommées par un service.
- Sécuriser une unité de service systemd.

## 2 Pré-requis, recommandations et notation du TP.

Vous travaillerez individuellement. Il vous explicitement demandé de faire valider votre travail par l'enseignant. Ces "checks" permettront de vous noter. Un compte rendu succinct ( fichiers de configuration , copie d'écran montrant la réussite de votre construction ...) est demandé et à rendre sur Moodle.

## 2.1 Obtenir de l'aide sur systemd.

### 2.1.1 Changements à réaliser sur votre VM

Vous travaillerez avec une VM en utilisant l'OVA Ubuntu sur <http://store.iutbeziers.fr>

## 3 Configurez la version deux des cgroups

Systemd utilise par défaut la version 1 des cgroups<sup>1</sup>.

Afin de bénéficier de la pleine puissance de systemd modifiez le fichier `/etc/default/grub` en rajoutant à `GRUB_CMDLINE_LINUX_DEFAULT` l'item `"systemd.unified_cgroup_hierarchy=1"`.

Après un `"update-grub"` rebootez votre système.

### 3.1 Aide sur systemd.

```
man systemd.unit
man systemd.service
man systemd.socket
man systemd.resource-control
man systemd.timer
...
```

La complétion avec la touche `tab` fonctionne aussi.

```
apt install bash-completion
```

Pour recharger une unité modifiée.

```
systemctl daemon-reload
```

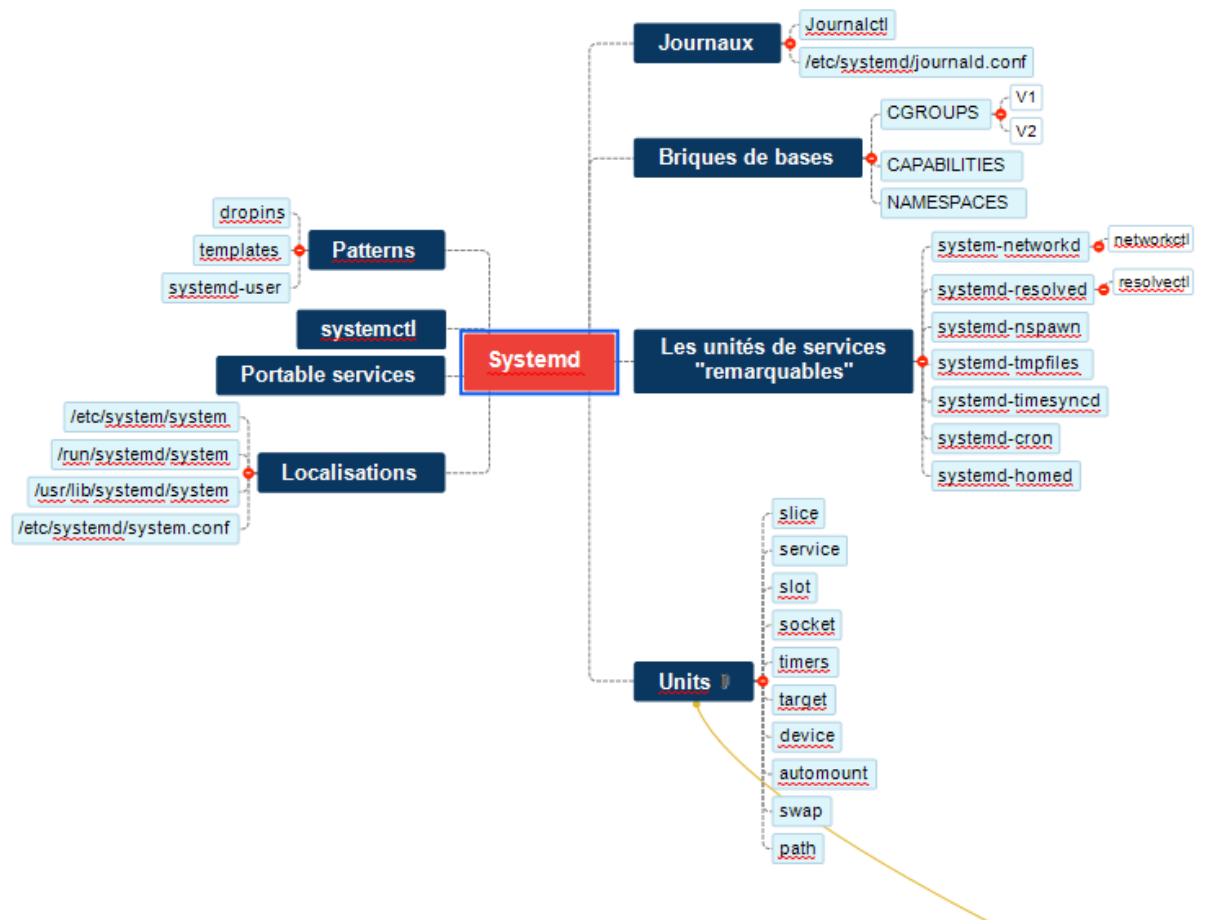
Pour éditer avec vim:

```
export SYSTEMD_EDITOR=/usr/bin/vim
```

---

1. <https://wiki.archlinux.org/index.php/cgroups>

FIGURE 1 – Vue d'ensemble de systemd:



## 4 Niveau 1: concepts et utilisation basique de systemd

### 4.1 Quelques éléments sur systemd

Systemd distingue plusieurs types de "units" (voir schéma ci-dessus). Chaque "units" est une abstraction d'une ressource de l'O.S.. Le type service est bien entendu essentiel.

Le type Socket permet un fonctionnement de type xinetd. // Un exemple de "unit file" de cups (daemon d'impression):

```
# /lib/systemd/system/cups.service
[Unit]
Description=CUPS Scheduler
Documentation=man:cupsd(8)
After=network.target nss-user-lookup.target nslcd.service
Requires=cups.socket

[Service]
ExecStart=/usr/sbin/cupsd -l
Type=notify
Restart=on-failure
```

Le fichier permet de faire le lien avec le binaire à démarrer mais aussi d'explicitier les relations avec les autres units. Il indique aussi que la "unit" cups.socket doit être actif, que le service sera redémarré en cas de "fail" et que entre autre le réseau doit être fonctionnel.

La commande maître afin d'interagir avec systemd est **systemctl**. Le fichier maître qui définit le comportement général de systemd est `/etc/systemd/system.conf` accompagné de `/etc/systemd/journald.conf` pour paramétrer la journalisation des logs.

Les concepteurs de systemd ont astucieusement prévu les interactions des gestionnaires de paquets et des administrateurs systèmes avec systemd: les concepteurs de package installent les configurations systemd dans `/lib/systemd`, les administrateurs systèmes locaux dans `/etc/systemd`. Une configuration pour la durée de la session peut être faite `/run/systemd`. Attention `/etc` a la précedence sur `/run` qui a la précedence sur `/lib`. Il ne sert donc à rien de modifier pour la session si une configuration du service existe déjà dans `/etc`.

Par exemple pour empêcher une connexion ssh sur votre machine pour la durée de la session il suffit d'arrêter le service ssh:

```
systemctl stop ssh
```

On va utiliser `ssh.socket` qui a pour particularité de démarrer un process ssh quand il y a une connexion. On surcharge le service `ssh.socket` pour le *runtime de la machine*:

```
sudo systemctl edit --runtime ssh.socket
```

avec la configuration suivante:

```
[Socket]
IPAddressDeny=any
IPAddressAllow=127.0.0.1,IP_Du_COPAIN
```

et de démarrer le service `ssh.socket` en runtime:

```
sudo systemctl start --runtime ssh.socket
```

Il suffit de redémarrer la machine ou le service ssh pour que la modifications ne soient plus prise en compte (précédence de /etc sur /run) Vous pouvez lister les chemins des configurations des unit-files au travers de cette commande:

```
systemd-analyze unit-paths
/etc/systemd/system.control
/run/systemd/system.control
/run/systemd/transient
/run/systemd/generator.early
/etc/systemd/system
/etc/systemd/system.attached
/run/systemd/system
/run/systemd/system.attached
/run/systemd/generator
/usr/local/lib/systemd/system
/lib/systemd/system
/usr/lib/systemd/system
/run/systemd/generator.late
```

Cette façon de préserver les modifications des fichiers de configuration de systemd lors des mises à jour s'appelle un "DropIn". permet de travailler en surchargeant une configuration existante.

Les fichiers suivants intéresseront particulièrement les administrateurs réseaux:

- /etc/systemd/networkd.conf (configuration du réseau)
- /etc/systemd/resolved.conf (configuration de la résolution de nom).

Les commandes *hostnamectl* , *networkctl*, *resolvectl* sont aussi des commandes usitées.

## 4.2 Quelques options de bases de systemctl

1. Listez les "units".
2. Listez les "unit-files".
3. Listez les "units" de type service.
4. Listez les propriétés du daemon cups.
5. Affichez la configuration du service ssh. Quel est le service qui doit précéder son démarrage?
6. Comment empêcher le re-démarrage de ce service? testez la solution.
7. Vérifiez que le service ssh est redémarré automatiquement si vous "killez" son processus. Visualisez le journal du service pour le confirmer.
8. Listez les dépendances du service ssh.
9. Quel est la cible par défaut du démarrage de la machine virtuelle?
10. Donnez la localisation du service ssh dans l'arborescence de la machine.
11. Est-ce que ce service est démarré? activé? en échec?
12. Listez les dépendances du service ssh.
13. En une seule commande listez toutes les unités liées à ssh? est ce que l'unité ssh.socket est activée?

## 4.3 Création et gestion d'un service de "ping"

1. Créez le service ip-accounting-test.service<sup>2</sup> en utilisant la commande:

```
systemctl edit --full --force ip-accounting-test.service
```

L'option `--force` n'est utile qu'à la création de l'unité de service. L'option `--full` permet d'éditer directement un service de niveau "OS".

Renseignez un simple ping:

---

2. <http://0pointer.net/blog/ip-accounting-and-access-lists-with-systemd.html>

#### [Service]

```
ExecStart=/usr/bin/ping 8.8.8.8
IPAccounting=yes
```

2. Démarrez le service.
3. Quel est le status du service ?
4. A l'aide de journalctl listez les messages relatifs au service créé. Listez le dernier évènement avec journalctl l'option -n 1 ? rajoutez l'option "-o verbose" afin d'afficher les variables d'environnements.
5. A l'aide de "systemctl status" déduire ce que fait l'option IPAccounting.
6. Utilisez la commande:

```
systemctl show ip-accounting-test -p IPIngressBytes -p IPIngressPackets -p IPEgressBytes -p IPEgressPackets
```

7. Surchargez afin de modifier indirectement (via systemctl edit sans l'option -full) le service ip-accounting-test pour pinguer l'IP 1.1.1.1 en lieu et place de l'IP 8.8.8.8 ?
8. Comment fonctionne la "surcharge" d'un service ? quel est l'intérêt de ce mode de fonctionnement ?
9. Bloquez les accès réseaux à votre VM en utilisant la commande suivante:

```
systemctl set-property system.slice IPAddressDeny=any IPAddressAllow=localhost
```

10. Autorisez uniquement l'IP de votre hôte à se connecter.(Systemctl se base sur l'"IP NAT").

## 5 Niveau 2: administration système avec systemd

### 5.1 Manipulations avec systemd-runtime

1. Créez un bash contenant la commande:

```
wall "c'est l'heure de la pause"
```

Utilisez systemd-run afin de programmer le lancement ce programme dans 2 minutes.  
Mais n'écoutez pas cet ordre, la pause viendra après...

2. Lancez un bash avec l'option DynamicUser.

```
systemd-run -p DynamicUser=1 -t /bin/bash
```

Quelle est la caractéristique de cet utilisateur ? Quel est l'intérêt de cette fonctionnalité.

3. Quels sont les effets des options ProtectHome et ProtectSystem ? Vérifiez-le en lançant un shell avec systemd-run.
4. Lancez firefox avec systemd-run en limitant la mémoire à 1G. Réessayer avec l'option -scope. Quel est l'effet de l'option -scope ?

### 5.2 Administration de systemd

1. Quelle est votre version de systemd ?
2. Que vous indique la commande suivante ?

```
ls -l /proc/1/exe
```

3. Utilisez les commandes systemd-cgls et systemd-cgtop. A quoi correspondent les slices ? quelle est la slice "mère" ? quelle est la fonctionnalité du kernel utilisée par systemd ?
4. Listez le status des unités ? leur nombre ?
5. Quelle est la différence entre "systemctl --all list-units" et "systemctl list-units" et "systemctl list-unit-files" ?

6. Listez les services ?
7. Listez les services "failed" des sockets ?
8. Utilisez journalctl, l'option vacuum et ses dérivés pour positionner la taille maximale des journaux à 500M et à deux journées de rétention.
9. Testez systemd-resolved en mode debug en utilisant:

```
export SYSTEMD_LOG_LEVEL=debug /lib/systemd/systemd-resolved
```

10. Installez apache2 et masquez son unité de service par la commande:

```
systemctl mask apache2.service
```

Quel est l'effet de l'option "mask".

11. Analyser le temps de boot du système avec systemd-analyze. Graphez le temps de boot avec l'option plot.
12. Vérifiez ce qui est les surcharges de votre configuration avec la commande "systemd-delta".
13. Testez que nous sommes dans une machine virtualisée avec la commande :

```
systemd-analyze condition 'ConditionVirtualization=true'
```

14. Quelle est la target (c'est l'équivalent des run level init) par défaut de systemd ? Passez temporairement en multi-user.target jusqu'au prochain reboot.
15. Listez les variables d'environnement avec systemctl.
16. Trouvez le fichier de configuration de systemd.

## 6 Niveau 3: administration système avancée avec systemd

### 6.1 Activation d'une unité "service" ssh par une unité "socket"

Systemd est doté d'une fonctionnalité intéressante dite "dropin" : l'activation d'une unité de service par une unité "socket". Il s'agit d'un changement de paradigme : c'est la connexion d'un client à une socket réseau qui va réveiller un service. L'intérêt est de ne pas consommer de ressources sans connexion d'un client de l'unité de service.

Continuons avec notre exemple sur ssh:

Si "ssh.socket" n'est pas lancée, la configuration du port ssh va se faire classiquement dans /etc/ssh/sshd\_config.

On se propose donc d'activer l'unité ssh.socket sur le port 2222 et le port 2223. C'est ssh.socket qui va demander le lancement d'un daemon ssh à chaque connexion.

1. Arrêtez le service ssh puisqu'il sera réveillé par ssh.socket.
2. Editez ssh.socket et surchargez le port d'écoute de a manière suivante.

```
systemctl edit ssh.socket
[Socket]
ListenStream=
ListenStream=2222
ListenStream=IP-DE-VOTRE-VM:2223
FreeBind=true
```

3. Regardez la directory /etc/systemd/system/ssh.service.d et déduisez le mode d'action du dropin et de l'édition. Faites prendre en compte la modification par systemd au travers de:

```
systemctl daemon reload
systemctl enable ssh.socket
systemctl start ssh.socket
systemctl stop ssh.service
```

4. Lancez une connexion ssh sur le port 2222 et le port 2223. Pourquoi cette activation par socket n'est pas faite par défaut ?
5. Dumppez la configuration ssh.socket. Comment prévenir le démarrage du daemon ssh ?
6. Vérifiez les logs de connexions ssh via la commande suivante:

```
journalctl -u ssh.service
```

## 6.2 Utilisation des templates systemd

Systemd propose aussi une fonctionnalité intéressante qui est celle des templates. Un cas d'utilisation est la gestion des versions d'une application avec systemd.

Installez le "framework" Python flask sur votre VM:

```
pip3 install flask
```

Récupérez le projet:

```
cd /home/student
git clone https://registry.iutbeziers.fr:11443/pouchou/systemd-demo.git
```

Il est composé d'un code très simple en Python affichant la version de l'application. Cette application est lancée par le script lance\_monapp.sh. Systemd est en charge de gérer cette application. Créez son fichier de configuration via la commande suivante<sup>3</sup>:

```
systemctl edit --force --full monapp@
# /etc/systemd/system/monapp@.service
[Unit]
Description=appliquette python
After=network.target

[Service]
Type=simple
User=student
Environment="FLASK_RUN_PORT=500%i"
Environment="VERSION_MONAPP=%i.0"
WorkingDirectory=/home/student/systemd-demo
ExecStart=/home/student/systemd-demo/lance_monapp.sh
Restart=always

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

```
root@ubuntu:/home/bin/systemd-demo# cat lance_monapp.sh
#!/bin/bash
export FLASK_ENV=development
export FLASK_APP=monapp.py
flask run
```

Le code Python est le suivant:

---

3. Attention l'arobace caractérise un template et il ne faut pas l'omettre



```

1 from flask import Flask,jsonify
2 import os
3
4 app = Flask(__name__)
5 version = os.environ.get("VERSION_MONAPP")
6
7 @app.route('/')
8 def hello_world():
9     return jsonify("Le Python c'est bon mangez en")
10
11 @app.route('/version')
12 def hello_color():
13     return jsonify(version)
14
15 if __name__ == '__main__':
16     app.run(debug=True,host='0.0.0.0',port=5000)

```

Vérifiez le bon fonctionnement de ce mode "template":

```

systemctl start monapp@1
systemctl start monapp@2
systemctl start monapp@3
systemctl start monapp@4
curl http://127.0.0.1:5001/version
curl http://127.0.0.1:5002/version
curl http://127.0.0.1:5003/version
curl http://127.0.0.1:5004/version

```

Expliquer le fonctionnement de ce mode "template".

## 7 "Hardening" de la configuration de notre appliquette Python

1. Analysez la sécurité de l'appliquette avec :

```
systemd-analyze --user security monappstudent@.service
```

2. Améliorez le score de sécurité jusqu'au "feu vert" de systemd-analyze.
3. Limitez la consommation de cpu (CPUWeight=10=min...=100=default...10000=max) et de mémoire (MemoryMax) de l'appliquette Python.
4. Protégez votre application des connexions autre que celle de l'adresse IP de votre machine physique.

### 7.1 Gestion par systemd au niveau de utilisateur

On se propose de faire fonctionner notre appliquette Python directement sous l'utilisateur student de votre VM. Vous serez donc connecté sur la machine en tant que user student.<sup>4</sup>.

1. Recréez l'application précédente en ajoutant `-user` à la commande `systemctl` et en supprimant les directives `USER` et `GROUP` du fichier de configuration du service.

Vous pouvez donc créer le fichier de configuration avec la commande suivante:

```
systemctl edit --user --force --full monappstudent@
```

Vérifiez que le templating fonctionne aussi.

---

4. Ne pas utiliser un login ssh avec le compte student

FIGURE 2 – Options de sécurité des unités

