«««< HEAD ======= »»»> d0040a5 (no include)

Systemd for fun & profits en BUT 1

Jean-Marc Pouchoulon

Mai 2023

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.

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 (Validation du service ping). 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 Debian en dernière version. sur `http://store.iutbeziers.fr`

2.2 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
```

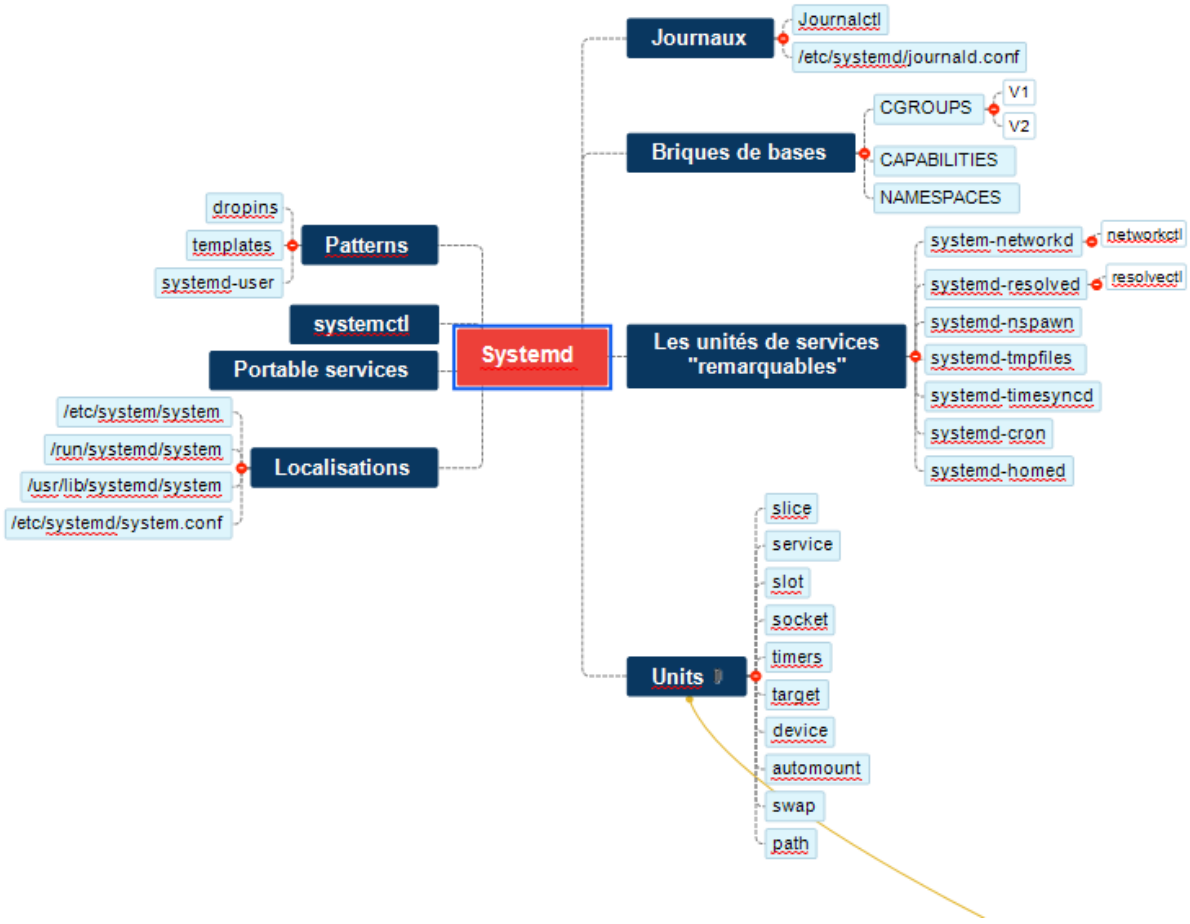
Pour recharger une unité modifiée.

```
systemctl daemon-reload
```

Pour éditer avec vim:

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

FIGURE 1 – Vue d'ensemble de systemd:



3 Niveau 1: concepts et utilisation basique de systemd

3.1 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 proche de ce qu'est capable de faire le daemon xinetd. [1] // 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'expliciter 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 premiers installent les configurations systemd dans "/lib/systemd", les seconds dans "/etc/systemd". Une configuration pour la durée de la session peut être faite /run/systemd. Attention /etc a la précédence sur /run qui a la précédence 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: cat /etc/netplan/01-netcfg.yaml # Let NetworkManager manage all devices on this system network: version: 2 renderer: networkd ethernets: eth0: dhcp4: true dhcp6: false eth1: addresses: - 10.3.0.14/24 nameservers: search: [iutbeziers.fr] addresses: [1.1.1.1] routes: - to: 0.0.0.0 via: 10.3.0.2

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

et de démarrer le service ssh.socket en runtime:

1. xinetd est un capable de recevoir des requêtes pour plusieurs types de service et evite que chaque service écoute sur un port particulier

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

Vérifiez que cet configuration fonction en faisant un ssh depuis la VM sur la loopback. 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". Elle 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.

3.2 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.

Solution:

```
C'est marqué au début du fichier
    # /lib/systemd/system/apache2.service
```

systemctl list-units –type service systemctl cat ssh systemctl list-dependencies sshd.service systemctl get-default systemctl show –all cups

11. Est-ce que ce service est démarré ? activé ? en échec ?

Solution:

```
systemctl is-active ssh.service
active
systemctl is-enabled ssh.service
systemctl is-failed ssh.service
active
```

12. Listez les dépendances du service ssh.

Solution:

```
systemctl list-dependencies ssh.service
```

13. En une seule commande listez toutes les unités liées à ssh ? est ce que l'unité ssh.socket est activée ?

Solution:

```
# Attention ne marche pas avec zsh
systemctl list-unit-files  ssh.*
sudo bash -c "systemctl list-unit-files ssh.*"
UNIT FILE   STATE    VENDOR PRESET
ssh.service enabled  enabled
ssh.socket  disabled enabled
```

3.3 Travailler avec les services réseaux fournis par systemd

1. Donnez l'option de resolvectl qui permet de lister les DNS pour chacune des interfaces.
2. A l'aide de resolvectl obtenez l'adresse IP de www.iutbeziers.fr .
3. Combien de requêtes dns ont été faites au service systemd-resolved ? quel est l'efficience de votre cache DNS en pourcentage ?
4. Quel est la chaîne de fichiers qui permet de configurer le service systemd-resolved.
5. Rajoutez deux cartes réseaux eth1 et eth2 sur votre machine virtuelle. configurez eth1 via systemd-networkd en créant un fichier .network dans /etc/systemd/network .

```
#/etc/systemd/network/50-static.network
[Match]
Name=eth1

[Network]
DHCP=yes


[DHCP]
RouteMetric=100
UseMTU=true
```

```
# Let NetworkManager manage all devices on this system
# /etc/netplan/01-netcfg.yaml
network:
  version: 2
```

```
renderer: networkd
ethernets:
  eth0:
    dhcp4: true
    dhcp6: false
  eth1:
    addresses:
      - 10.3.0.14/24
    nameservers:
        search: [iutbeziers.fr]
        addresses: [1.1.1.1]
    routes:
      - to: 0.0.0.0
        via: 10.3.0.2
```

6. Configurer eth2 à l'aide de Netplan et avec une IP fixe. Vérifiez avec la commande netplan ip leases la configuration de la carte via le DHCP.
7. Comment netplan fait-il le lien est-il avec systemd-networkd (cherchez dans /run/systemd) ?
8. Avec hostnamectl changez le nom de votre VM.

Solution:

```
resolvectl dns
root@debian:/run/systemd/network# resolvectl statistics
DNSSEC supported by current servers: no

Transactions
Current Transactions: 0
  Total Transactions: 109

Cache
  Current Cache Size: 1
          Cache Hits: 2
        Cache Misses: 17

DNSSEC Verdicts
               Secure: 0
             Insecure: 0
                Bogus: 0
      Indeterminate: 0
17/

vim   /etc/systemd/network/50-static.network
[Match]
Name=eth1

[Network]
DHCP=yes

root@debian:/run/systemd/network# cat /etc/netplan/01-netcfg.yaml
# Let NetworkManager manage all devices on this system
# /etc/netplan/01-netcfg.yaml
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: true
      dhcp6: false
    eth1:
      addresses:
        - 10.3.0.14/24
      nameservers:
          search: [iutbeziers.fr]
          addresses: [1.1.1.1]
      routes:
        - to: 0.0.0.0
          via: 10.3.0.2


root@debian:/run/systemd/network# cat 10-netplan-eth0.network
[Match]
Name=eth0

[Network]
DHCP=ipv4
LinkLocalAddressing=ipv6

[DHCP]
RouteMetric=100
UseMTU=true
root@debian:/run/systemd/network# cat 10-netplan-eth1.network
[Match]
Name=eth1

[Network]
LinkLocalAddressing=ipv6
Address=10.3.0.14/24
```

9. Analyser le temps de boot du système avec systemd-analyse. Graphez le temps de boot avec l'option plot.

Solution:

```
Startup finished in 6385ms (kernel) + 3228ms (initrd) + 49335ms (userspace) = 58949ms
12673ms cups.service
11788ms dkms_autoinstaller.service
5264ms network.service
5156ms ip6tables.service
5045ms avahi-daemon.service
$ systemd-analyze plot > plot.svg
$ systemd-analyze blame
$ eog plot.svg
...
 $ systemd-analyze time
```

3.4 Création et gestion d'un service de "ping"

1. Créez le service ip-accounting-test.service[2] 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:

```
[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
```

Solution:

```
journalctl -u ip-accounting-test -n 1 -o verbose
systemctl show ip-accounting-test  -n 1 -o verbose
```

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 ? La surcharge se fait en initialisant à blanc l'item que l'on veut modifier puis en lui affectant sa nouvelle valeur.
8. Comment fonctionne la "surcharge" d'un service ? quel est l'intérêt de ce mode de fonctionnement ?

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

Solution:

```
[Service]
ExecStart=
IPAccounting=
ExecStart=/usr/bin/ping 1.1.1.1
IPAccounting=yes
```

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").