**Systemd**

|  |  |  |  |
| --- | --- | --- | --- |
| **DESC** | | | |
| Gestionnaire système et services (Initialisation système, journalisation kernel et user centralisée…) | | | |
| **FILES** | | | |
| /etc/ |  | |  |
|  | inittab | | Fichier lu au démarrage d’init (runlevel) |
|  | Init.d/ | | Scripts d’init.  Systemd fait la conversion en services systemd via systemd-sysv-generator. |
|  | Rc.d/ | | Détails de tous les run-levels d’Init :  Scripts d’initialisation |
|  | Systemd/ | Journald.conf | Configuration des logs de journald |
|  |  | Timesyncd.conf | Configuration de timesyncd |
|  |  | System/ | Lien symbolique vers la case en dessous  Fichiers unités utilisateurs  Priorité sur ci-dessous |
| /lib/Systemd/system/ | | | Unit files : init, stop…script  Ne pas modifier ces fichiers  Fichiers unités fournis par Debian |
| /usr/lib/systemd/system/ | | |
| /run/systemd/system/ | | | Fichiers Unit crées dynamiquement.  Perdu lors du reboot |
| **QUICK START** | | | |
| Création d’un service :   * Editer un fichier.service dans /etc/systemd/system/ * Renseigner [Unit], [Service] et [Install] pour un service * Démarrer le service | | | |

**Généralités :**

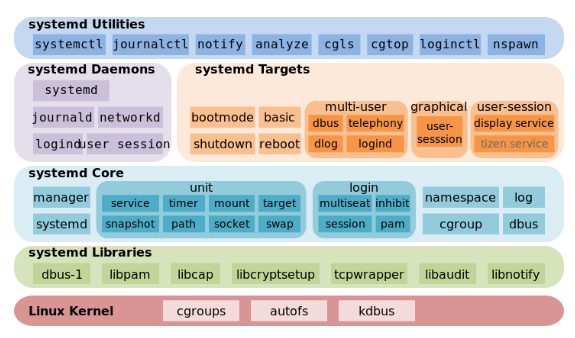
* Successeur d’Init et upstart, de PID 1 (Init est un lien symbolique vers systemd sous Debian 9)
* Gestionnaire de services (boot…), de montage, d’états du système (runlevel)…
* Composants de systemd :
  + Journald (journalctl) : collecte et gère les logs du kernel + userland.
  + Systemd (systemctl) : Gère les unités
  + Timesyncd (timedatectl) : Client NTP fourni par systemd
  + Systemd-cgls : diagnostique système
  + Systemd-analyse : performances du système (Non dispo sous certaines Debian)
* **Service** = scripts ou programme
* **.Targets** : anciennement run-levels, ce sont des groupes d’unités pour un état précis du système (sans echec, sans echec avec réseau etc). Les cibles appellent les unités pour cet état. Une cible peut dépendre d’une autre cible (dossiers .wants/)
* **Unité** : Peut-être une socket, un service, un point de montage…Chacune dépendant d’une autre. Un service peut être démarré via une socket, une modification de fichier, activité sur un bus etc…Permet aussi de restarter un service sans couper la socket.
* **.snapshot** : sauvegarde/réstauration de l’état du système
* Nom.service, nom.socket. Chaque service dans un cgroup
* Journalctl va stocker ses logs au format binaire (permet de les exporter en syslog, en JSON…). Se combine très bien avec rsyslog ou syslog-ng.
* Avantage :
  + approche centralisées des logs
  + beaucoup de méta data avec journalctl
  + Plus besoin de gérer des scripts pour la gestion du service, systemd s’occupe de tout (start, stop, gestion des logs, status…)
  + Séparation des fonctions d’un service au travers de plusieurs types d’unités permet d’optimiser les initialisations parallèles.

De plus, si le service crash ou on l’upgrade, la socket continue de fonctionner : on ne perd pas de messages.

* Peut remplacer Fstab et Cron (vu que systemd s’occupe des points de montage et de timers)

**Override de certaines directives d’un fichier unit :**

* Systemctl edit <unité>
* Ajout d’un dossier programme.service.d juste à côté du fichier unit, puis X.conf dans ce dossier



**Types d’unités :**

|  |  |
| --- | --- |
| **Unit** |  |
| .service | Cette unité décrit comment gérer le service ou l’application :  Start, stop, conditions de démarrage, dépendances, infos… |
| .socket | Décrit un réseau, un IPC ou une FIFO et est associé à un service.  Systemd l’utilise pour l’activation d’un service par le réseau. |
| .device | Ce device doit être géré par systemd |
| .mount | Ce point de montage doit être géré par systemd |
| .automount | Déinit un point de montage automatiquement monté  Doit être associé à un .mount |
| .swap |  |
| .target | Points de synchronisation lors du boot ou d’un changement d’état du serveur (run-level).  Les autres unités spécifient leur relation à une target  Groupement de plusieurs unités |
| .path | Systemd l’utilise pour l’activation d’un service par un chemin |
| .timer | Similaire à cron :  Un service peut être démarré quand le timer est atteint |
| .snapshot | Point de restauration : sauvegarde et restauration état actuel |
| .slice | Groupe d’unités (scopes et services) organisées hiérarchiquement  Associé à un cgroup : contrôle des ressources par services.  On assigne des scopes et services à des slices.  4 slices crées par défaut :   * -.slice : slice root * System.slice : pour les services system * User.slice : pour toutes les sessions user * Machine.slice : pour les VMs et les containers |
| .scope | Utilisé pour gérer des sets de processus : encapsule un parent et tous ses fils |

**Syntaxe générale d’un fichier unit :**

#Case sensitive

[Section]

Directive1=value

Directive2=value

Directive3= #Reset de la valeur

…

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Section** | **Directive** | | | | **Comments** |
| [Unit] |  | | | | Métadatas  Relation avec les autres unités  Première Section d’un fichier Unit |
|  | Description= | | | |  |
|  | Documentation= | | | | URLs de docs |
|  | Requires= | | | | Liste d’unités qui doivent être démarrés avec succès et dont cette unité dépend.  Si ces dépendances ne sont pas satisfaites, l’unité ne démarrera pas. |
|  | Wants= | | | | Similaire à Requires mais en moins strict : si les dépendances ne sont pas satisfaites, l’unité continuera de fonctionner quand même |
|  | Bindsto= | | | | Similaire à Requires, mais cette unité se terminera si les unités associés se termineront. |
|  | Before= | | | | Les unités ici ne seront pas démarrées avant l’unité courante |
|  | After= | | | | Les unités ici seront démarrées avant l’unité courante |
|  | Conflicts= | | | | Les unités qui ne peuvent pas fonctionnées en même temps |
|  | Condition= | | | | Autre conditions pour que l’unité démarre et fonctionne |
|  | Assert= | | | | Idem Condition mais plus strict |
| [Install] |  | | | | Dernière section d’un fichier Unit  Optionnelle : Définit le comportement de l’unité si elle est enabled ou disabled (lors du boot donc)  Les directives listées listent quoi faire si l’unité est enabled |
|  | WantedBy= | | | | Spécifie une relation de dépendance, comme Wants= de Unit.  Mais != : cette directive est inclue dans l’unité auxiliaire :  Creation d’un dossier dans /etc/systemd/system/unité\_auxiliaire.wants/ qui contient des liens symboliques vers l’unité courante. |
|  | RequiredBy= | | | | Creation d’un directory .requires/ |
|  | Alias= | | | | Autre nom du service |
|  | Also= | | | | Autorise cette unité à être dans un set ou groupe |
|  | DefaultInstance= | | | | Nom par défaut pour les templates |
| [Service] |  | | | | Propre à l’unité service |
|  | Type= | |  | | Management du service  Un seul type à la fois |
|  |  | | simple | | Service up directement (pas de fork) |
|  |  | | forking | | Service up quand fork et parent se termine |
|  |  | | oneshot | | Processus subliminal : systemd doit attendre sa fin avant de continuer les autres unités. |
|  |  | | dbus | | Service up quand son nom apparait sur D-Bus |
|  |  | | notify | | Ce service va procéder à une notification quand il a fini de démarrer : systemd le mettra ensuite up |
|  |  | | idle | |  |
|  | RemainAfterExit= | | | | Utiliser avec oneshot.  Processus se termine, le service reste actif |
|  | PIDFile= | | | | Utiliser avec forking.  PID du fils principal qui doit être monitoré |
|  | BusName= | | | | A utiliser avec dbus |
|  | NotifyAccess= | | |  | A utiliser avec notify |
|  |  | | | none | Ignore les messages status |
|  |  | | | main | Ecoute les messages du processus principal. |
|  |  | | | all | Ecoute les messages de tous les services du control group. |
|  | ExecStart= | | | | Cmd à lancer + arguments   * Cmd + args :   Si le processus retourne !=0, le service n’est pas failed |
|  | ExecStartPre= | | | | Commande(s) à lancer avant le processus principal. |
|  | ExecStartPost= | | | | Idem mais juste après que le processus principal à démarrer. |
|  | ExecReload= | | | | Commande nécessaire au reload de la configuration |
|  | ExecStop= | | | | Commande pour stopper le service.  Si absent, processus va être kill quand le service est stoppé. |
|  | ExecStopPost= | | | |  |
|  | RestartSec= | | | | Si restart automatique spécifié,  Temps restant avant de redémarrer le service. |
|  | Restart= |  | | | Circonstances de restart quant à la manière dont le service à été stoppé. |
|  |  | always | | |  |
|  |  | on-success | | |  |
|  |  | on-failure | | |  |
|  |  | on-abnormal | | |  |
|  |  | on-abort | | |  |
|  |  | on-watchdog | | |  |
|  | TimeoutSec= | | | | Temps d’attente entre le moment ou le service est stoppé et le moment où il est déclaré failed. |
| [Socket] |  | | | | Associé à une unité service qui sera démarré quand la socket reçoit des informations. |
|  | ListenStream= | | | | @ avec connexion fiable (TCP) |
|  | ListenDatagram= | | | | @ avec connexion non fiable (UDP) |
|  | ListenSequentialPacket= | | | | @ avec connexion séquentielle fiable et taille datagrammes fixe (Unix socket). |
|  | ListenFIFO= | | | |  |
|  | Accept= | | | | Détermine si une autre instance du service doit être lancé si plusieurs connexions. |
|  | SocketUser= | | | | Propriétaire de la socket avec une socket Unix.  Root si non spécifié. |
|  | SocketGroup= | | | | Idem |
|  | SocketMode= | | | | Permissions de la socket Unix ou pile FIFO |
|  | Service= | | | | Si service != du nom de la socket (nom.socket) |
| [Mount] |  | | | | Point de montage géré par systemd |
|  | What= | | | | Chemin absolu de la ressource qui doit être montée |
|  | Where= | | | | Chemin absolu de là où la ressource doit être montée. |
|  | Type= | | | | FS |
|  | Options= | | | | Options séparés par une virgule |
|  | SloppyOptions= | | | | Booléen : option non reconnue : montage va échouer |
|  | DirectoryMode= | | | | Permissions si création de dossier nécessaires pour le point de montage |
|  | TimeoutSec= | | | | Temps d’attente avant que l’opération de montage soit failed |
| [Automount] |  | | | | Associé avec un .mount, permet de monter automatiquement lors du boot.  Autre manière : edition /etc/fstab |
|  | Where= | | | | Chemin absolu du point d’automontage sur le FS |
|  | DirectoryMode= | | | | Permissions si création de dossier nécessaires pour le point de montage |
| [Swap] |  | | | | Création d’une zone de swap via fstab ou bien via une unité |
|  | What= | | | | Chemin absolu de la localisation de l’espace de swap, si c’est un fichier ou un périphérique |
|  | Priority= | | | | Priorité du swap configuré |
|  | Options= | | | |  |
|  | TimeoutSec= | | | | Temps d’attente d’activation du swap avant failure |
| [Path] |  | | | | Désigne un FS que systemd peut monitorer.  Activation d’une unité si changement sur le path |
|  | PathExists= | | | | Test de l’existence du chemin.  Si oui, activation de l’unité |
|  | PathExistsGlob= | | | | Idem mais supporte expressions glob |
|  | PathChanged= | | | | Activation de l’unité si fichier fermé et changements |
|  | PathModified= | | | | Activation de l’unité si changements peu importe que le fichier soit ouvert ou fermé |
|  | DirectoryNotEmpty= | | | | Activation de l’unité si dossier non vide |
|  | Unit= | | | | Unité à démarrer |
|  | MakeDirectory= | | | | Création de dossiers si besoin |
|  | DirectoryMode= | | | | Permissions si MakeDirectory activé |
| [Timer] |  | | | | Similaire à cron |
|  | OnActiveSec= | | | | Autorise l’unité associée à être activée par cette unité timer. |
|  | OnBootSec= | | | | Temps d’attente après le boot avant que l’unité associée soit activée |
|  | OnStartupSec= | | | | Idem au-dessus mais une fois que systemd à démarré |
|  | OnUnitActiveSec= | | | | Timer depuis que l’unité associée a été activée. |
|  | OnUnitInactiveSec= | | | | Timer depuis que l’unité associée a été désactivée. |
|  | OnCalendar= | | | |  |
|  | AccuracySec= | | | | Précision du timer |
|  | Unit= | | | | Unité associée |
|  | Persistent= | | | |  |
|  | WakeSystem= | | | | Réveil du système de sa suspension à la fin du timer |
| [Slice] |  | | | |  |
| [Scope] |  | | | |  |

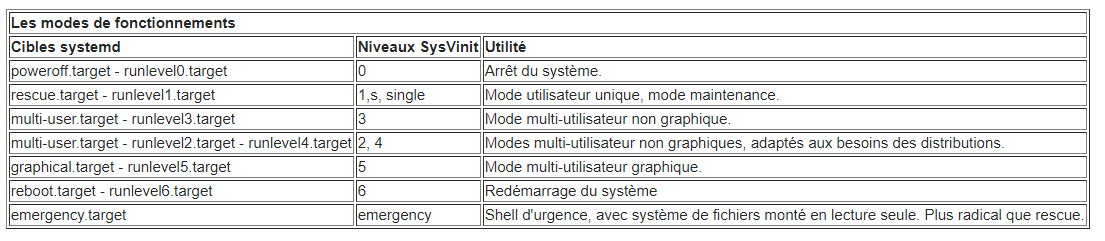
**Template**

Permet de configurer un fichier Unit de manière dynamique.

[example@.service](mailto:example@.service) : template

[example@instance1.service](mailto:example@instance1.service) : instance du template

**Niveaux d’Exécution :**

****.target = mode de fonctionnement du système (=run-levels), qui correspond à un démarrage d’un certain nombre de services.

**Configuration journald.conf :**

|  |  |  |
| --- | --- | --- |
| **Section** | **Directive** | **Comments** |
| [Journal] |  |  |
|  | Storage=persistent | Sauvegarde des précédents boots |
|  | SystemMaxUse= | Taille maxi du journal en stockage persistant. |
|  | SystemKeepFree= | Taille que le journal doit laisser libre |
|  | SystemMaxFileSize= | Taille maxi d’une entrée avant d’être rotated |
|  | RuntimeMaxUse= | Taille maxi du journal en stockage volatile (/run) |
|  | RuntimeKeepFree= |  |
|  | RuntimeMaxFileSize= |  |