Table des matières

[Généralités 1](#_Toc146699154)

[Lire le status 2](#_Toc146699155)

[Démarrer, arrêter, relancer un service 5](#_Toc146699156)

[Activer ou désactiver le service au démarrage / bloquer un service 7](#_Toc146699157)

[Information sur un service 11](#_Toc146699158)

[PS 13](#_Toc146699159)

[Les services propres à systemd 16](#_Toc146699160)

# Généralités

**systemd** est devenu le gestionnaire de systèmes d’initialisation et de systèmes par défaut de nombreuses distributions Linux, mais attention il n’est pas universellement implémenté sur toutes les distributions. En effet certaines distributions peuvent encore utiliser System V autrement appelé SysVinit (qui utilisait la commande service) et upstart (qui a été un système de gestion intermédiaire pas très implémenté). À noter que pour le moment, les distributions qui utilisait ces autres versions et qui sont passées à systemd, ont mis au point une redirection de commande. C’est-à-dire que si vous utilisez la commande service sshd status , le système l’interprète, la traduit et exécute en réalité la ‘nouvelle version’ systemctl status sshd. Mais plus on avance dans le temps, plus ces alias tendent à disparaitre. Donc habituez-vous aux commandes systemctl.

L’objectif fondamental d’un système d’initialisation est d’initialiser les composants qui doivent être démarrés une fois que le noyau Linux est lancé (on les appelle composants « userland »). De plus, le système d’initialisation vous permet de gérer à tout moment les services et les démons du serveur alors que le système est en marche.

Dans systemd, la cible de la plupart des actions sont des « unités », c’est-à-dire des ressources que systemd sait gérer. Les unités sont classées par le type de ressources qu’elles représentent. Leur configuration se fait avec des fichiers que l’on appelle des « fichiers de l’unité ». Vous pouvez déduire le type de chaque unité à l’aide du suffixe qui se trouve à la fin du fichier.

Pour les tâches de gestion de service, l’unité cible correspondra aux unités de service pour lesquelles les fichiers de l’unité se terminent par le suffixe .service. Cependant, en réalité, pour la plupart des commandes de gestion de service, vous n’avez pas nécessairement besoin du suffixe .service. En effet, systemd est assez intelligent pour savoir que, lorsque vous utilisez les commandes de gestion de service, vous souhaitez probablement travailler sur un service.

systemd se manipule grâce à la commande systemctl.

Pour faire une action sur un service, c'est facile, on fera :

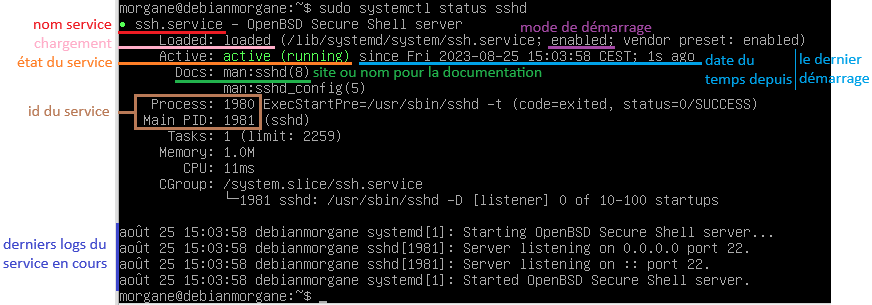
systemctl [action] [nom\_du\_service].service.

Par exemple pour le service ssh, on fera la commande suivante pour voir le status du service :

systemctl status sshd.service

# Lire le status

Décryptage du status d’un service :



**Le chargement** du service (début ligne loaded) peut être :

loaded : si le service est correctement chargé en mémoire

error : s'il y a eu un problème de chargement

not-found : si aucun fichier d'unité n'a été trouvé pour ce service

bad-setting : si un paramètre de fichier de service essentiel n'a pas pu être analysé

masked : si le fichier unité a été masqué

À la suite se trouve le chemin d'accès au fichier du service, cette ligne affichera également l'état d'activation, soit le mode de démarrage.

**Le mode de démarrage** peut être :

À noter que le chemin spécifié est l’emplacement du fichier de configuration de l’unité de service.

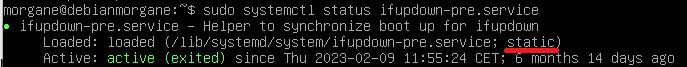
enabled: le service sera activé au moment du démarrage du système.



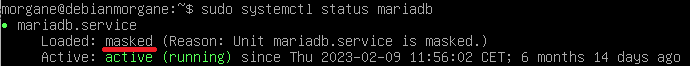
disabled: le service est désactivé et ne sera pas démarré au démarrage du système.



static : cela signifie que le service ne peut pas être géré à l'aide de systemd (ou de la commande systemctl) et que vous devrez disposer d'un autre service d'initialisation ou le gérer manuellement

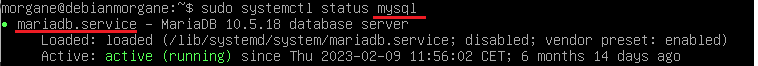


masked : cela signifie que le service est masqué et ne peut pas être déclaré directement à l'aide de la commande systemctl. Cela peut être utile lorsque vous souhaitez éviter un démarrage accidentel du service.



alias : On ne le voit pas écris sur le status directement, mais on peut être face à un alias sur le nom d’un service, cela permet d’appeler un service par un nom (ici mysql) et de lancer un autre service (mariadb) grâce à un lien symbolique pointant vers un autre fichier unité.





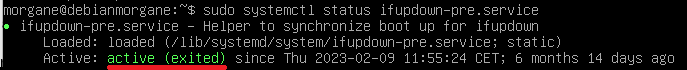
linked : Il indique que le service ou le fichier unité est symboliquement lié à un autre fichier unité.

**L’état** peut être :

active (running) : le service s'exécute activement en arrière-plan.



active (exited) : indique que le service était censé être exécuté une fois ou périodiquement. Le service a donc fait son travail, puis s'est arrêté une fois terminé



active (waiting) : cela indique que le service est en cours d'exécution, mais qu'il attend d'être déclenché par une condition comme un événement spécifique.

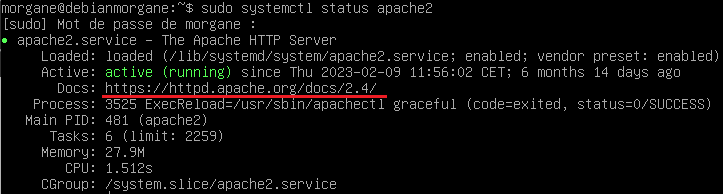
inactive : le service n'est pas en cours d'exécution actuellement.



failed : le service est en erreur, en lisant les logs, vous devriez avoir une indication concernant l’erreur



**Le** **Docs** peut être le nom du paquet à saisir pour voir le manuel, ou alors l’adresse complète d’un site internet pour avoir la documentation complète.



**Main PID ou Process** : Dans tous les cas, si un service est actif, il aura démarré avec l’option ‘ExecStart’ et il aura un Main PID, c’est-à-dire, son identifiant unique et à la suite vous pouvez voir son nom. Si le démarrage de ce service se fait par une autre option, par exemple ‘ExecStartPre’, ou ‘ExecStartPost’ …, alors en plus du Main PID, le service aura la valeur ‘Process’. Il est possible que les valeurs Main PID et Process soit identiques, ou bien différents. De même que vous pouvez avoir plusieurs lignes sur Process, si plusieurs processus ont été démarrés par le service. Sur le process, vous pouvez voir l’option de démarrage, le fichier concerné et l’état du démarrage du service.

Quand on va vouloir tuer un processus, c’est ce Main PID qu’on va devoir utiliser.

**Logs :**

Lisez bien les logs, par exemple, pour le ssh, on peut voir son port d’écoute, là on sait qu’il écoute sur le port 22. Mais quand il y a une erreur et que le service ne se lance pas, c’est aussi écrit ici, dans les derniers logs du service. Donc, prenez le temps de les lire !

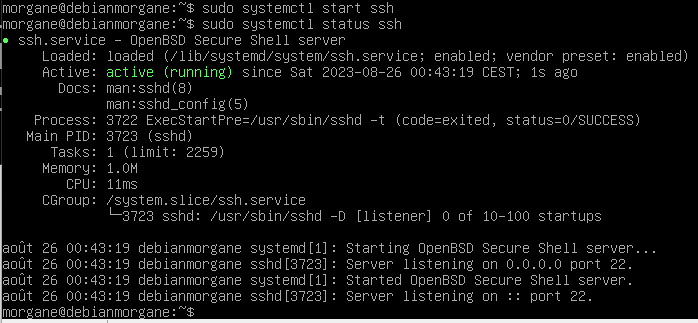
Maintenant que vous savez lire l’état d’un service et comprendre ce que vous voyez, passons à l’administration de ces services.

# Démarrer, arrêter, relancer un service

Dans les exemples ci-dessous, remplacez [nom\_du\_service] par le nom de votre service : par exemple : sshd, mariadb, apache2, dhcpd, bind9 etc.

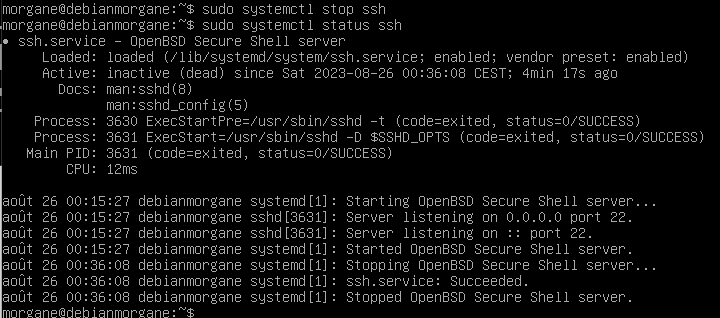
Pour démarrer le service :

systemctl start [nom\_du\_service].service



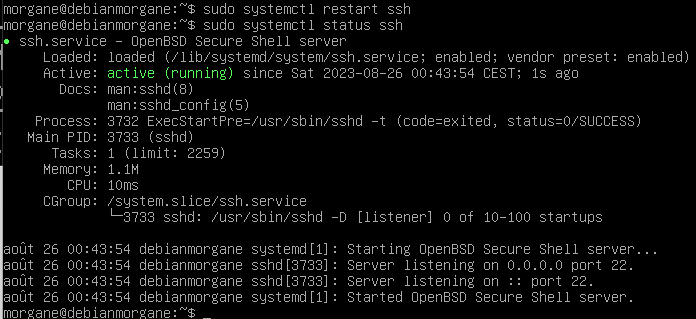
Pour arrêter le service :

systemctl stop [nom\_du\_service].service



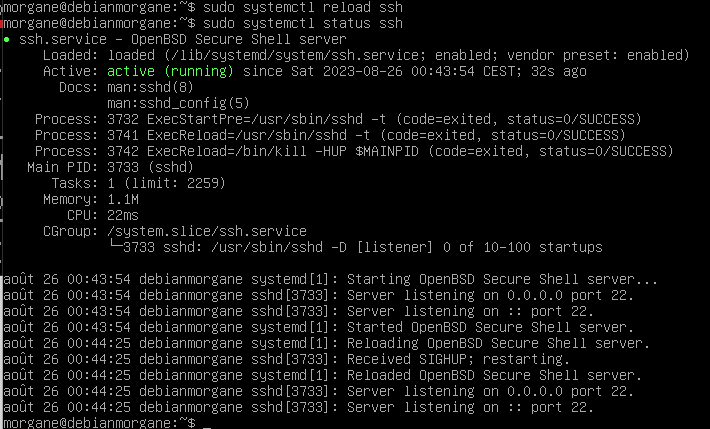
Pour redémarrer un service qui est lancé, faites :

sudo systemctl restart [nom\_du\_service].service



Pour recharger les fichiers de configuration d'un service sans le redémarrer (typiquement, le serveur web Apache), faites :

systemctl reload [nom\_du\_service].service

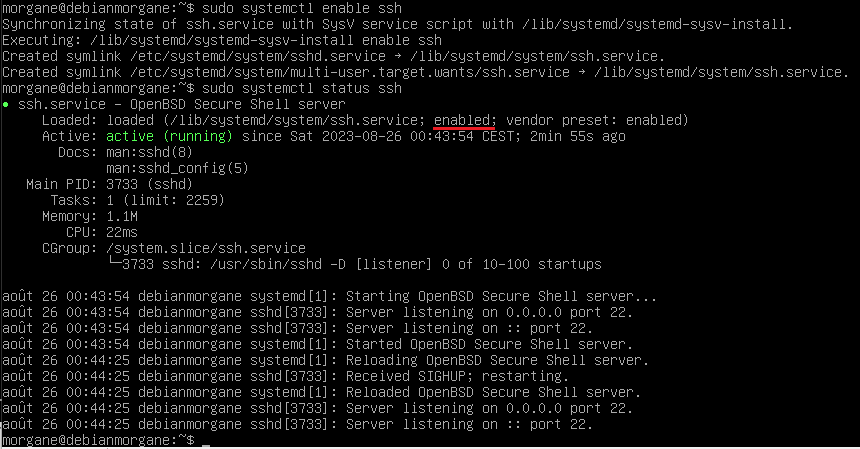


# Activer ou désactiver le service au démarrage / bloquer un service

Pour que le service soit lancé au démarrage du système :

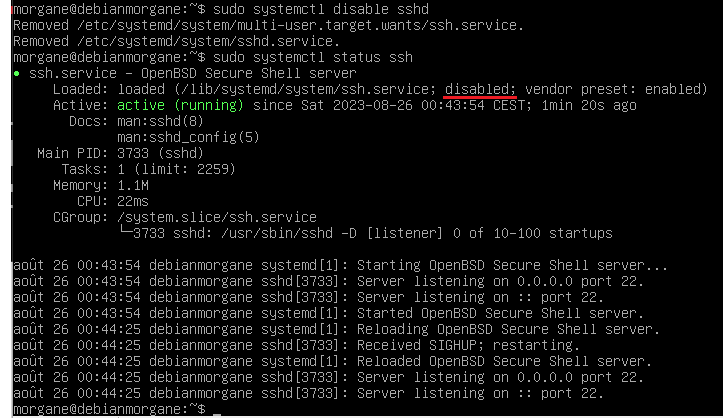
systemctl enable [nom\_du\_service].service

Cela créera un lien symbolique à partir de la copie du fichier de service du système (généralement dans /lib/systemd/system ou /etc/systemd/system) à l’emplacement du disque où systemd cherche les fichiers de démarrage automatique (généralement /etc/systemd/system/some\_target.target.wants.



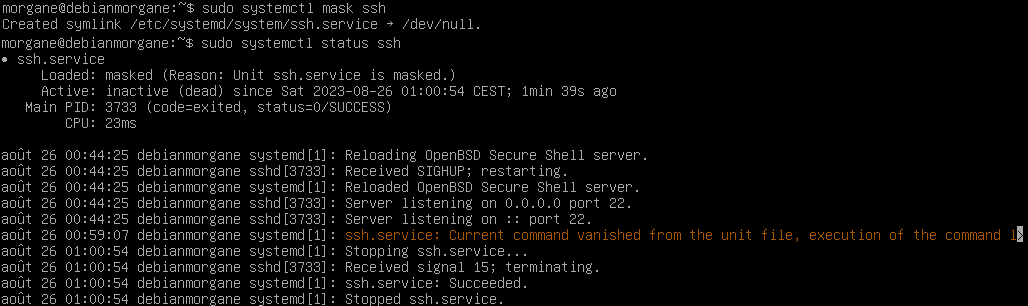
Pour que le service ne soit pas lancé au démarrage du système :

systemctl disable [nom\_du\_service].service



Pour empêcher l'activation d'un service (par exemple on masquera apache2 car on veut utiliser nginx et donc éviter que l'admin système démarre apache2 par inadvertance) :

systemctl mask [nom\_du\_service].service



Tant qu’elle est masquée, tout démarrage automatique ou manuel du service qui est masqué est impossible, ici j’ai masqué le service ssh.

Si vous vérifiez la list-unit-files, vous verrez que le service est maintenant répertorié comme étant masqué :

systemctl list-unit-files





Si vous tentez de lancer le service, vous verrez s’afficher le message suivant :

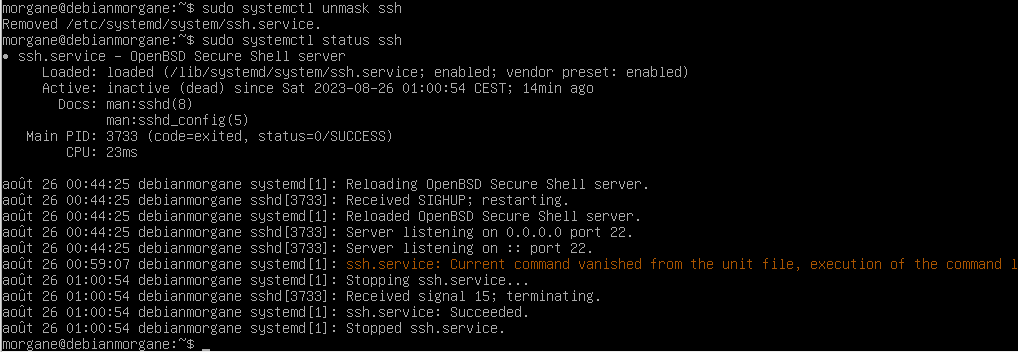
systemctl start ssh



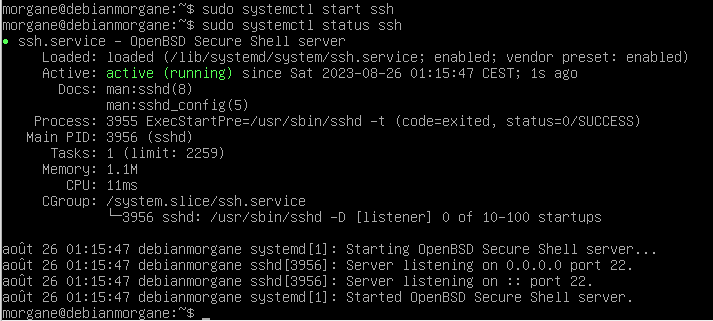
Pour rendre son utilisation à nouveau possible, utilisez la commande unmask :

systemctl unmask ssh

Cela renverra l’unité à l’état précédent, ce qui lui permettra son démarrage automatique ou son activation.



Je relance le service :



Pour envoyer un signal d'arrêt (SIGTERM) à tous les processus du service (plus élégant qu'un killall qui tue en fonction d'une chaine de caractère) :

systemctl kill [nom\_du\_service].service



# Information sur un service

Pour voir tous les services disponibles et leur statut, y compris les services de statut inactif :

systemctl list-unit-files --type=service –all





Pour vérifier si le service est démarré, arrêter, afficher des informations pour le niveau d'exécution en cours :

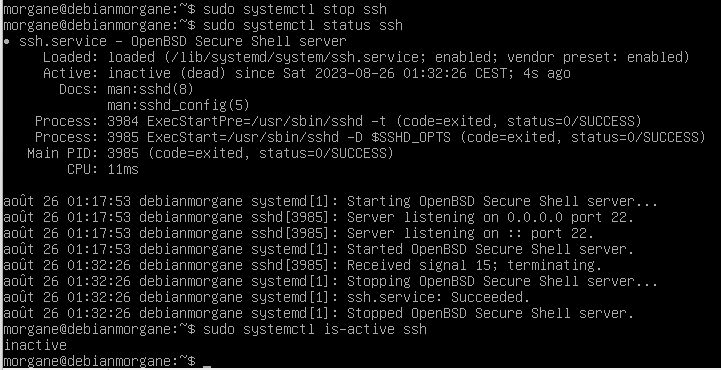
systemctl status [nom\_du\_service].service

Pour voir si le service est actuellement démarré :

systemctl is-active [nom\_du\_service].service



Pour voir ce que ça donne comme résultat, j’ai éteint le service et relancé la commande :



Pour vérifier si le service sera démarré au démarrage du système :

systemctl is-enabled [nom\_du\_service].service





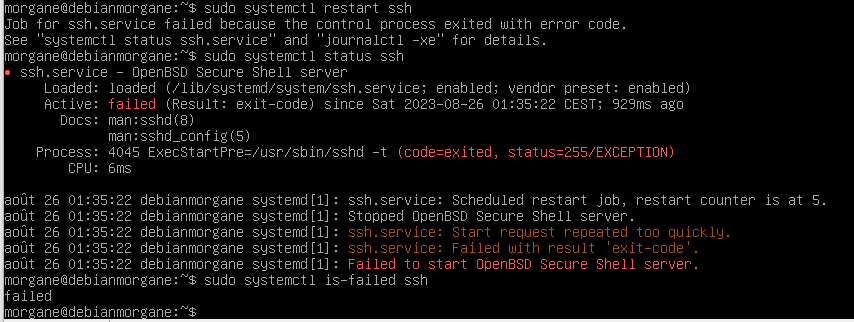
Donc le service ssh est démarré au démarrage du système, ce qui n’est pas le cas du service mariadb.

Pour vérifier s'il y a eu un problème lors du démarrage d'un service :

systemctl is-failed [nom\_du\_service].service



Pour voir ce que ça donne, je créer une erreur sur le service, je le redémarre, et je retape la commande :



On voit bien sur la dernière commande ‘failed’ comme résultat, mais sur le log du status, on voit aussi qu’il est en erreur, et surtout lors de la commande de restart le système indique clairement qu’il y a une erreur.

Pour voir tous les services qui ont un problème :

systemctl --failed --type=service

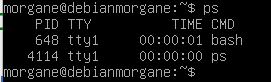


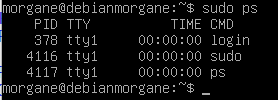
# PS

Évidemment les commandes Unix classiques vous permettent aussi de comprendre ce qui se passe au niveau des services, notamment, vous pouvez par exemple lister les processus actifs avec la commande ps.

La commande ps par défaut affiche les commandes qui sont actuellement utilisées dans son invite de commande locale (TTY1).

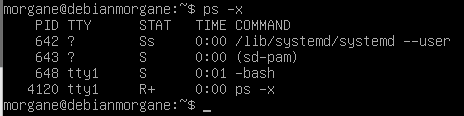
ps





Lister tous les processus actifs de l’utilisateur courant

ps -x



Donc avec ps seul, on voyait que mon utilisateur avait 2 commandes actives dans mon invite de commande (bash et ps), mais en listant les processus actifs de mon utilisateur, j’ai 4 résultats. Le 1er et le 2eme permettent l’identification de mon utilisateur et la récupération de son mot de passe dans la base PAM. Le 3ème processus correspond au lancement de mon bash, et le 4ème à l’exécution de la commande ps -x elle-même.

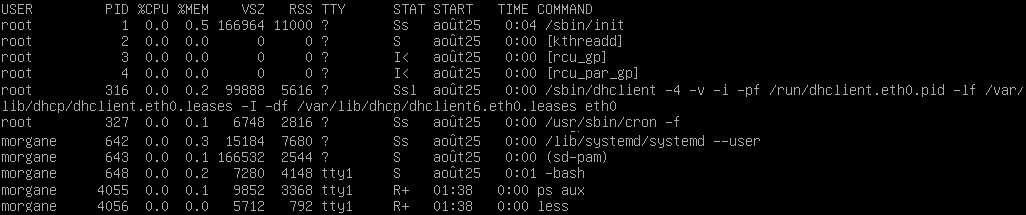
Si je teste la commande sudo ps -x, vu que j’exécuterai la commande en tant que root, le système va m’afficher la liste de tous les processus de root, autrement dit, tous ceux de la machine, sauf ceux de mon utilisateur courant.

La plupart du temps, sur internet vous trouverez la commande ps aux, car elle permet d’afficher tous les processus de la machine, quel que soit leurs utilisateurs (en spécifiant son nom), et d’indiquer les ressources associées.

ps aux

Voici une sélection des résultats. Dans tous les cas, les colonnes signifient :

* user : le nom de l’utilisateur du processus
* pid : l’identificateur unique du processus qui a été définis par le système
* %cpu : pourcentage d’utilisation du processeur par ce processus
* %mem : pourcentage d’utilisation du processeur par ce processus
* vsz : précise l’utilisation des bibliothèques partagées et la mémoire utilisée pour son fonctionnement
* rss : indique l'utilisation de la mémoire physique utilisée en kilobytes par le processus (hors swap)
* tty : indique le numéro de port du terminal (le "?" indique que la commande n'est pas associée à un terminal)
* stat : affiche l'état actuel du processus, R comme Run Démarrer, T pour terminé, S pour sleep (endormi/en attente) …
* start : indique l'heure à laquelle le processus a commencé.
* time : affiche le temps processeur utilisé par ce processus
* command : le nom de la commande qui utilise ce PID



Le pid du processus c’est ce qui va nous permettre de l’identifier afin de le gérer.

Pour connaitre directement le pid d’un processus :

pgrep bash

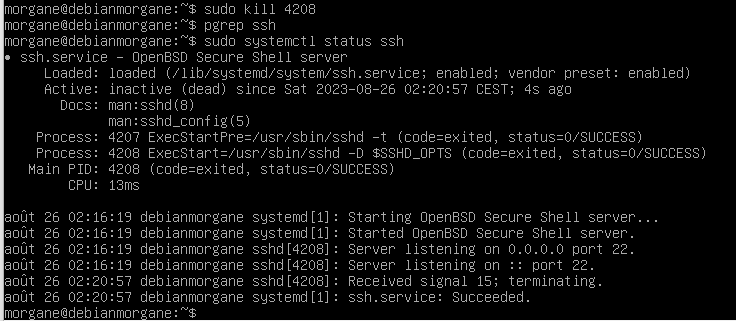


pgrep ssh



On a vu précédemment la commande systemctl kill [nom\_du\_service].service qui permet d’envoyer un signal d’arrêt propre à un service en utilisant son nom. Mais si vous préférez utiliser le pid d’un service, la demande d’arrêt se fera avec :

kill [PID\_à\_kill]



Cela envoie le signal TERM au processus. Le signal TERM indique au processus de bien vouloir se terminer. Cela permet au programme d’effectuer des opérations de nettoyage et de s’arrêter en douceur, tout comme avec la commande systemctl kill [nom\_du\_service].

Si le programme se comporte mal et ne se ferme pas lorsque le signal TERM est actionné, nous pouvons escalader le signal en passant le signal KILL :

kill -KILL [PID\_à\_kill]

Il s’agit d’un signal spécial que n’est pas envoyé au programme.

Au lieu de cela, il est envoyé au noyau du système d’exploitation qui interrompt le processus. Vous pouvez l’utiliser pour contourner les programmes qui ignorent les signaux qui leur sont envoyés.

# Les services propres à systemd

Systemd utilise en interne un certain nombre de services pour la gestion du système, citons en quelques-uns :

À vous d’apprendre à les utiliser !

systemd-journald : messages du système et des programmes ;

systemd-logind : connexion des utilisateurs ;

systemd-vconsole-setup : configuration des consoles virtuelles ;

systemd-remount-api-vfs : remontage du système de fichier, en tenant compte des options de fstab ;

systemd-sysctl : application des paramètres système de /etc/sysctl.conf ;

systemd-stdout-syslog-bridge : utilisation de syslog pour les processus de systemd ;

systemd-tmpfiles-setup : création de fichiers et répertoires temporaires ;

La commande systemd-analyze blame, permet de lister le temps de démarrage des différents services. C'est assez pratique pour déterminer le ou les services qui mettent du temps à se lancer.