Linux Debian - Administration

m2iformation.fr

Sommaire

Jour 3

- Description du processus de démarrage
- Gestionnaires de boot GRUB
- Gestion des unités service et cible (target) de systemd
- Gestion des services SysVinit
- Dépannage en mode Rescue ou Emergency
- Définition des processus, des threads et de l'ordonnancement
- Analyse de l'activité système (top, htop, pstree, ps...)
- Gestion des signaux (kill)
- Tâches avant et arrière plans (jobs, bg, fg...)
- Planification de tâches

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Jour 3

Description du processus de démarrage

1. **Description du processus de démarrage (Boot Process)**

Le processus de démarrage dans Linux est un ensemble d'étapes cruciales qui commencent dès que l'ordinateur est allumé et se terminent par le chargement du système d'exploitation. Ce processus peut être décomposé en plusieurs étapes :

a) BIOS/UEFI

- Lorsque vous allumez la machine, le **BIOS** (Basic Input/Output System) ou **UEFI** (Unified Extensible Firmware Interface) s'initialise.
- Le BIOS/UEFI effectue une série de tests matériels appelés **POST** (Power-On Self-Test) pour vérifier l'intégrité de l'équipement.
- Ensuite, il localise un périphérique amorçable, généralement le disque dur, qui contient le **secteur de démarrage principal** (MBR) ou **GPT**.

Description du processus de démarrage

b) Bootloader (GRUB)

- Une fois que le BIOS ou UEFI a localisé un disque contenant un secteur d'amorçage valide, il passe le contrôle au bootloader.
- Sur Debian 12, le gestionnaire de boot le plus couramment utilisé est **GRUB** (GRand Unified Bootloader).
- GRUB lit son fichier de configuration et propose un menu permettant à l'utilisateur de sélectionner le noyau Linux ou un autre système d'exploitation à démarrer (si plusieurs systèmes sont installés).
- Le fichier de configuration de GRUB est généralement situé dans `/boot/grub/grub.cfg`. Il contient des informations telles que les noyaux disponibles et les options de démarrage (par exemple, le mode recovery).

c) Chargement du noyau (Kernel)

- Une fois qu'un noyau a été sélectionné (par défaut ou par l'utilisateur), GRUB charge ce noyau en mémoire.
- Le noyau Linux s'initialise, puis charge les modules nécessaires pour interagir avec les matériels de la machine (drivers pour disque, réseau, etc.).

d) Initramfs

- **Initramfs** (initial RAM filesystem) est une image temporaire montée en mémoire par le noyau. Elle contient les fichiers et modules nécessaires pour permettre au noyau de monter le système de fichiers racine (root filesystem).
- L'image initramfs est également chargée par GRUB au démarrage.

Description du processus de démarrage

e) Systemd

- Après le montage du système de fichiers racine, le processus **init** (le premier processus à démarrer sous Linux) prend le relais. Sur Debian 12, c'est **systemd** qui est responsable de gérer ce processus.
- Systemd orchestre le démarrage de tous les services et démons nécessaires pour faire fonctionner le système (comme les interfaces réseau, les services SSH, etc.).
- Systemd lit une série d'unités (unit files) pour lancer les services et configure les cibles (targets) qui déterminent l'état final du système (multi-user, graphical, etc.).

Gestionnaires de boot : GRUB

a) Rôle de GRUB

- GRUB est un gestionnaire de démarrage multi-plateforme utilisé pour charger le noyau Linux et d'autres systèmes d'exploitation installés sur un système.
- Il peut amorcer des noyaux Linux ou des systèmes comme Windows, et permet à l'utilisateur de choisir entre plusieurs options de démarrage (comme un noyau spécifique ou le mode de récupération).

b) Structure de GRUB

- GRUB est divisé en plusieurs parties :
  - **Stage 1** : C'est le tout premier code exécuté à partir du secteur d'amorçage (MBR ou GPT). Il est responsable de charger la deuxième étape.
  - **Stage 2** : Cette étape charge le système de fichiers et affiche le menu GRUB.
  - **Modules** : GRUB a également des modules qui peuvent être utilisés pour ajouter des fonctionnalités supplémentaires (comme le support réseau pour un démarrage PXE).

Gestionnaires de boot : GRUB

c) Configuration de GRUB

- Le fichier de configuration principal de GRUB se trouve dans `/boot/grub/grub.cfg`. Il ne doit pas être modifié directement, car il est automatiquement généré à partir du fichier `/etc/default/grub` et des scripts dans `/etc/grub.d/`.
- Pour mettre à jour la configuration de GRUB, vous devez modifier le fichier `/etc/default/grub` et exécuter la commande `update-grub`. Cela régénérera le fichier de configuration final.

d) Personnalisation de GRUB

- Vous pouvez personnaliser GRUB pour changer le délai avant qu'il ne sélectionne automatiquement une option par défaut, ajouter des paramètres au noyau (comme des options de dépannage), ou même ajouter des fonds d'écran au menu de démarrage.
- Exemple : pour réduire le délai du menu GRUB, vous pouvez modifier la ligne `GRUB_TIMEOUT=5` dans `/etc/default/grub` et exécuter `update-grub`.

e) Utilisation en mode dépannage

- GRUB peut être utilisé pour démarrer en **mode single-user** ou en **mode de récupération** pour dépanner un système cassé.
- Vous pouvez également modifier temporairement les paramètres du noyau au démarrage depuis l'interface GRUB pour par exemple désactiver un module problématique ou ajouter un mode verbose.

Gestionnaires de boot : GRUB

f) Gestion du dual boot avec GRUB

- Si vous avez plusieurs systèmes d’exploitation (comme Debian et Windows), GRUB détecte automatiquement les autres systèmes et les affiche dans son menu de démarrage.
- L'utilisateur peut configurer GRUB pour démarrer automatiquement un système spécifique ou définir un délai d'attente avant le démarrage automatique.

Exercice - GRUB

Objectif :

Ajouter une image de fond personnalisée dans le menu GRUB pour améliorer l'apparence du gestionnaire de démarrage sur Debian 12.

En tant qu'administrateur système, vous souhaitez personnaliser l'apparence de GRUB sur votre machine Debian 12 en ajoutant une image de fond dans le menu de démarrage.

1. Choisir une image de fond :

- Sélectionnez une image (au format PNG) qui sera utilisée comme fond d'écran pour le menu GRUB.

2. Copier l'image dans le bon répertoire :

- Placez l'image choisie dans le répertoire adéquat sur votre système pour que GRUB puisse y accéder.

3. Modifier le fichier de configuration de GRUB :

- Configurez GRUB pour qu'il utilise cette image comme fond dans son menu d'amorçage.

4. Tester et vérifier l'affichage :

- Appliquez les changements et redémarrez votre machine pour vérifier si l'image apparaît correctement dans le menu de démarrage.

Gestion des unités service et cible (target) dans systemd

Systemd est le gestionnaire de démarrage et des services par défaut sur Debian 12. Il remplace l'ancien système **SysVinit** et permet de gérer les services et d'autres aspects du système de manière plus flexible et efficace. Les unités de systemd (services, cibles, etc.) jouent un rôle central dans l'initiation et la gestion des composants du système.

1. Qu'est-ce qu'une unité (unit) dans systemd ?

Une **unité** dans systemd est un fichier de configuration qui décrit une ressource ou un service à gérer. Il peut s'agir d'un service, d'un périphérique, d'un montage de disque, d'un socket, ou même d'une cible regroupant plusieurs unités.

Les unités sont stockées dans des fichiers ayant l'extension **.service**, **.target**, **.socket**, etc., et sont situées dans les répertoires suivants :

- `/etc/systemd/system/` : pour les unités personnalisées ou spécifiques à l'utilisateur.
- `/lib/systemd/system/` : pour les unités installées par les paquets logiciels.
- `/usr/lib/systemd/system/` : pour les configurations par défaut du système.

2. Unités de service (.service)

Les unités **service** gèrent les services démarrés, arrêtés, redémarrés et surveillés par systemd. Chaque unité de service contient une définition du service, ses dépendances et ses actions spécifiques.

Gestion des unités service et cible (target) dans systemd

Structure d'une unité de service

Voici un exemple d'unité de service `apache2.service` :

```
[Unit]
Description=The Apache HTTP Server
After=network.target

[Service]
ExecStart=/usr/sbin/apachectl start
ExecStop=/usr/sbin/apachectl stop
ExecReload=/usr/sbin/apachectl graceful
Type=forking

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

Gestion des unités service et cible (target) dans systemd

Commandes pour gérer les services

- **Démarrer un service** :

```
sudo systemctl start apache2
```

- **Arrêter un service** :

```
sudo systemctl stop apache2
```

- **Redémarrer un service** :

```
sudo systemctl restart apache2
```

- **Vérifier le statut d'un service** :

```
sudo systemctl status apache2
```

- **Activer un service au démarrage** :

```
sudo systemctl enable apache2
```

- **Désactiver un service au démarrage** :

```
sudo systemctl disable apache2
```

Gestion des unités service et cible (target) dans systemd

3. Unités cible (target)

Les **cibles** (targets) sont des unités spéciales qui regroupent d'autres unités afin de coordonner le démarrage de groupes de services ou d'unités. Les targets ne démarrent pas de services eux-mêmes, mais organisent le démarrage des services en fonction de l'état voulu du système.

Principales cibles dans systemd

- **default.target** : C'est la cible par défaut qui est atteinte au démarrage du système. Sur un système en mode texte, elle est souvent liée à `multi-user.target`.
- **graphical.target** : Cible utilisée pour démarrer le système avec une interface graphique.
- **multi-user.target** : Cible pour un système multi-utilisateur en mode texte, souvent utilisé pour les serveurs.
- **rescue.target** : Cible pour le mode de récupération (single-user mode) pour effectuer des réparations.
- **shutdown.target** : Cible utilisée pour éteindre proprement le système.

Utilisation des cibles

Les cibles permettent de spécifier des états systèmes plus larges, comme démarrer tous les services nécessaires pour l'interface graphique, ou préparer le système pour un shutdown.

Gestion des unités service et cible (target) dans systemd

4. Relation entre services et cibles

Les services peuvent être associés à des cibles via la section `[Install]` des fichiers de service. Par exemple, un service peut être démarré automatiquement avec la cible `multi-user.target` si son fichier d'unité contient la directive `WantedBy=multi-user.target`.

De cette manière, **systemd** coordonne le démarrage, l'arrêt et la gestion de différents services et groupes de services en fonction des besoins de l'administrateur système.

Exercice

Objectif :

Comprendre et manipuler les unités de service et les cibles (targets) dans systemd sur Debian 12. Cet exercice vous permettra de créer, modifier et gérer des services, ainsi que d'utiliser les cibles pour gérer les groupes de services au démarrage.

En tant qu'administrateur système, vous devez gérer divers services sur un serveur Debian 12. Certains services doivent démarrer automatiquement, d'autres doivent être désactivés, et vous devez également modifier le comportement des cibles pour répondre aux besoins spécifiques du système.

1. Créer une unité de service personnalisée :

- Créez un service systemd personnalisé qui exécute un script simple au démarrage.
- Le script doit afficher un message dans les logs système.

2. Activer et démarrer le service :

- Activez ce service pour qu'il démarre automatiquement au démarrage du système.
- Vérifiez ensuite que le service fonctionne correctement en consultant les journaux.

Exercice

3. Désactiver un service existant :

- Identifiez un service actuellement actif sur le système qui n'est pas nécessaire pour cet exercice.
- Désactivez et arrêtez ce service pour qu'il ne démarre plus automatiquement.

4. Changer la cible par défaut de systemd :

- Modifiez la cible par défaut pour passer du mode graphique au mode multi-utilisateur en mode texte uniquement.
- Redémarrez le système pour vérifier que le changement est effectif.

5. Créer une nouvelle cible personnalisée :

- Créez une cible systemd personnalisée qui inclut votre service créé à l'étape 1.
- Démarrez cette cible manuellement et assurez-vous que votre service se lance correctement avec cette nouvelle cible.

Dépannage en mode Rescue ou Emergency

Le **mode Rescue** et le **mode Emergency** sont deux options importantes de **systemd** pour le dépannage du système sur Debian 12. Ces modes sont utilisés lorsque le système ne peut pas démarrer normalement ou lorsque des problèmes critiques doivent être résolus (comme la corruption du système de fichiers, une mauvaise configuration, ou des erreurs empêchant les services de fonctionner correctement).

1. Mode Rescue (rescue.target)

Le **mode Rescue** est similaire au mode single-user dans les systèmes basés sur **SysVinit**. Il permet d'obtenir une session root avec le système de fichiers monté en lecture-écriture mais ne démarre que les services essentiels, comme le montage des partitions.

Utilisation typique :

- Problèmes liés à des services critiques empêchant le système de démarrer correctement.
- Configuration réseau cassée.
- Problèmes de mot de passe root.

Dépannage en mode Rescue ou Emergency

Étapes pour accéder au mode Rescue :

1. **Redémarrer et accéder à GRUB** :
   - Démarrez ou redémarrez le système.
   - Au menu GRUB, appuyez sur la touche `e` pour éditer les options de démarrage.
2. **Modifier les options de démarrage** :
   - Localisez la ligne qui commence par `linux`, qui contient le chemin vers le noyau à démarrer.
   - Ajoutez `systemd.unit=rescue.target` à la fin de cette ligne. Cela indiquera à systemd de démarrer en mode rescue.

```
linux /boot/vmlinuz-5.10.0-8-amd64 root=/dev/sda1 ro quiet systemd.unit=rescue.target
```

3. **Démarrer en mode Rescue** :
   - Appuyez sur `Ctrl + X` ou `F10` pour démarrer avec les nouvelles options.
   - Vous serez redirigé vers un shell avec des privilèges root après avoir fourni le mot de passe root.

Dépannage en mode Rescue ou Emergency

4. **Résoudre les problèmes** :
    - En mode rescue, seules les partitions nécessaires sont montées. Vous pouvez effectuer des opérations comme :
        - **Modifier les fichiers de configuration** :

```
nano /etc/fstab  # Exemple pour corriger une erreur dans les montages
```

        - **Vérifier les systèmes de fichiers** (si nécessaire) :

```
fsck /dev/sda1
```

5. **Redémarrer le système** une fois les correctifs apportés :

```
reboot
```

Dépannage en mode Rescue ou Emergency

2. Mode Emergency (emergency.target)

Le **mode Emergency** est encore plus minimal que le mode Rescue. Il ne monte pas les systèmes de fichiers automatiquement, et seul le shell root est disponible. Ce mode est utilisé lorsque les systèmes de fichiers ne peuvent pas être montés, ou lorsqu'une erreur critique empêche le montage des partitions ou le lancement des services de base.

Utilisation typique :

- Systèmes de fichiers corrompus ou non montables.
- Problèmes critiques de montage dans `/etc/fstab`.
- Problèmes de démarrage du noyau.

Dépannage en mode Rescue ou Emergency

Étapes pour accéder au mode Emergency :

1. **Redémarrer et accéder à GRUB** :
   - Comme pour le mode Rescue, redémarrez et accédez au menu GRUB en appuyant sur `e`.
2. **Modifier les options de démarrage** :
   - Localisez la ligne commençant par `linux`.
   - Ajoutez `systemd.unit=emergency.target` à la fin pour démarrer directement en mode Emergency.

     ```
     linux /boot/vmlinuz-5.10.0-8-amd64 root=/dev/sda1 ro quiet systemd.unit=emergency.target
     ```
3. **Démarrer en mode Emergency** :
   - Appuyez sur `Ctrl + X` ou `F10`.
   - Vous accéderez à un shell minimal sans montage des partitions. Le système attendra le mot de passe root avant de vous accorder l'accès.

Dépannage en mode Rescue ou Emergency

4. **Effectuer des actions de dépannage** :
   - Comme aucun système de fichiers n'est monté par défaut, vous devrez peut-être les monter manuellement :

```
mount -o rw,remount /  # Remonter la partition root en lecture/écriture
mount /dev/sda1 /mnt   # Monter des partitions supplémentaires
```

   - **Vérifier et réparer les systèmes de fichiers** si nécessaire :

```
fsck /dev/sda1
```

5. **Redémarrer le système** après avoir résolu le problème :

```
reboot
```

Dépannage en mode Rescue ou Emergency

3. Comparaison entre mode Rescue et mode Emergency

Caractéristique	Mode Rescue	Mode Emergency
Accès réseau	Non (seulement les services critiques démarrent)	Non (aucun service démarré)
Systèmes de fichiers	Les systèmes de fichiers critiques sont montés	Aucun système de fichiers n'est monté
Démarrage de services	Quelques services essentiels sont démarrés	Aucun service n'est démarré
Utilisation	Problèmes liés aux services ou à des configurations	Problèmes critiques liés au noyau ou aux systèmes de fichiers

Dépannage en mode Rescue ou Emergency

4. Cas pratiques de dépannage en mode Rescue et Emergency

Exemple 1 : Problème de fichier `/etc/fstab` corrompu

Si le fichier `/etc/fstab` contient une erreur, le système peut échouer à démarrer en mode normal, bloquant sur le montage des partitions. Le mode Emergency vous permettra de corriger cette erreur.

1. Démarrez en mode Emergency.
2. Montez le système de fichiers root en lecture/écriture :

```
mount -o remount,rw /
```

3. Corrigez le fichier `/etc/fstab` avec un éditeur de texte.
4. Redémarrez le système.

Dépannage en mode Rescue ou Emergency

Exemple 2 : Problème avec un service réseau empêchant le démarrage

Si un service réseau est mal configuré et empêche le démarrage du système, le mode Rescue permet d'intervenir.

1. Démarrez en mode Rescue.
2. Désactivez le service problématique :

```
systemctl disable nom_du_service
```

3. Redémarrez et vérifiez que le système démarre correctement.

Définition des processus, des threads et de l'ordonnancement

1. **Définition des processus**
   - **Qu'est-ce qu'un processus ?**
     - Un processus est une instance d'un programme en exécution. Il inclut des éléments comme le code, les données en cours, et les ressources système nécessaires pour son exécution. Dans Debian 12, chaque processus a un **PID (Process ID)** unique qui permet de le gérer au niveau du système.
   - **Création de processus :**
     - Un processus peut être créé via la commande `fork()` pour générer un processus fils. Chaque processus a un processus parent.
     - Commandes utiles :
       - `ps -aux` : Affiche la liste des processus.
       - `top` ou `htop` : Outils interactifs pour surveiller les processus en temps réel.
   - **Gestion des processus :**
     - **Exécuter un processus en arrière-plan** : Utilisation du `&` à la fin d'une commande.
     - **Tuer un processus** : Utilisation de `kill` suivi du PID (`kill -9 <PID>` pour forcer la fermeture).
     - **Changer les priorités** : Utilisation des commandes `nice` et `renice`.

Définition des processus, des threads et de l'ordonnancement

2. **Définition des threads**
   - **Qu'est-ce qu'un thread ?**
     - Un thread est la plus petite unité d'exécution dans un processus. Un processus peut contenir plusieurs threads partageant les mêmes ressources (mémoire, fichiers ouverts, etc.). Sous Linux, Debian 12 gère les threads via le modèle **N:M**, où N threads sont mappés sur M noyaux.
   - **Différence processus vs thread :**
     - Un processus a son propre espace mémoire, tandis que les threads partagent la mémoire du processus parent.
     - Les threads sont plus légers et permettent un traitement parallèle plus rapide au sein du même processus.
   - **Commande liée :**
     - `ps -eLf` : Liste tous les threads du système.

Définition des processus, des threads et de l'ordonnancement

3. **Ordonnancement**
   - **Qu'est-ce que l'ordonnancement ?**
     - L'ordonnancement est le processus par lequel le noyau Linux attribue du temps CPU aux processus et aux threads. L'algorithme d'ordonnancement détermine quel processus ou thread aura accès au CPU à un moment donné.
   - **Types d'ordonnancement :**
     - **Ordonnancement préemptif** : Le noyau peut interrompre un processus en cours d'exécution pour donner la priorité à un autre, plus urgent.
     - **Ordonnancement non préemptif** : Un processus ne peut pas être interrompu tant qu'il n'a pas libéré volontairement le CPU.
   - **Classes d'ordonnancement dans Linux (Debian 12) :**
     - **SCHED_OTHER** : Classe par défaut pour les processus normaux.
     - **SCHED_FIFO** et **SCHED_RR** : Classes pour les processus temps réel, avec des priorités plus strictes.

top : Surveillance en temps réel des processus

- **Description :**
  La commande `top` affiche une liste des processus actifs en temps réel. Elle fournit des informations sur l'utilisation du CPU, de la mémoire, et d'autres statistiques importantes du système.
- **Exemple :**

```
top
```

- **Colonnes importantes** :
  - **PID** : Identifiant unique du processus.
  - **USER** : Nom de l'utilisateur propriétaire du processus.
  - **PR** : Priorité du processus.
  - **%CPU** : Pourcentage d'utilisation du CPU.
  - **%MEM** : Pourcentage d'utilisation de la mémoire.
  - **TIME+** : Temps total d'utilisation du CPU par le processus.

top : Surveillance en temps réel des processus

- **Utilisation avancée :**
- Pour trier les processus par utilisation de la mémoire :

```
top
(puis appuyer sur Shift + M)
```

- Pour tuer un processus directement depuis `top` :
  - Appuyer sur `k`, puis entrer le **PID** du processus à tuer.

htop : Version améliorée de top

- **Description :**
  `htop` est une version plus conviviale de `top`. Elle permet de naviguer de manière interactive dans la liste des processus avec des touches directionnelles et d'obtenir une vue plus visuelle de l'utilisation du CPU et de la mémoire.
- **Exemple :**

```
htop
```

  - **Avantages par rapport à `top` :**
    - Interface colorée et intuitive.
    - Utilisation de la souris pour sélectionner et interagir avec les processus.
    - Permet de voir facilement les différents cœurs du CPU et leur utilisation.
- **Utilisation avancée :**
  - Pour trier les processus par CPU ou mémoire, il suffit d'utiliser les touches de direction et de cliquer sur les en-têtes des colonnes.
  - Pour tuer un processus : Sélectionner le processus avec les flèches, puis appuyer sur `F9`.

pstree : Visualisation des processus en forme d'arbre

- **Description :**
  La commande `pstree` affiche les processus actifs sous forme d'arbre, ce qui permet de voir les relations parent-enfant entre eux. C'est un outil pratique pour comprendre la hiérarchie des processus.
- **Exemple :**

```
pstree
```

  - Cela affiche tous les processus en cours avec leur hiérarchie. Par exemple, si tu exécutes un programme à partir de ton terminal, il apparaîtra comme un enfant de ton processus shell (comme `bash`).
- **Utilisation avancée :**
  - Pour afficher l'arbre avec les PID :

```
pstree -p
```

  - Pour limiter l'affichage à un utilisateur spécifique :

```
pstree <nom_utilisateur>
```

ps : Instantané des processus

- **Description :**
  La commande `ps` affiche une liste instantanée des processus. Contrairement à `top` et `htop`, `ps` ne met pas à jour l'affichage en temps réel, mais fournit un instantané à l'instant de la commande.
- **Exemple :**

```
ps aux
```

- **Options courantes** :
  - `a` : Affiche tous les processus d'autres utilisateurs.
  - `u` : Affiche les informations sous un format détaillé (utilisateur, CPU, mémoire, etc.).
  - `x` : Affiche les processus n'ayant pas de terminal associé (par exemple, les processus en arrière-plan).

ps : Instantané des processus

- **Exemple de sortie** :

```
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.1  16124  2328 ?        Ss   09:44   0:03 /sbin/init
user        1931  0.0  0.2  56724  4144 pts/0    Ss   09:45   0:00 bash
user        2112  0.0  0.5  94876  9232 pts/0    R+   10:00   0:00 ps aux
```

- **Colonnes importantes** :
  - **PID** : Identifiant du processus.
  - **USER** : Utilisateur propriétaire du processus.
  - **%CPU** : Pourcentage d'utilisation du CPU.
  - **%MEM** : Pourcentage d'utilisation de la mémoire.
  - **COMMAND** : Commande exécutée par le processus.

ps : Instantané des processus

- **Utilisation avancée** :
- Pour lister les processus d'un utilisateur spécifique :

```
ps -u <nom_utilisateur>
```

- Pour afficher les threads en cours d'exécution :

```
ps -eLf
```

- Pour trouver un processus spécifique par nom :

```
ps aux | grep <nom_du_processus>
```

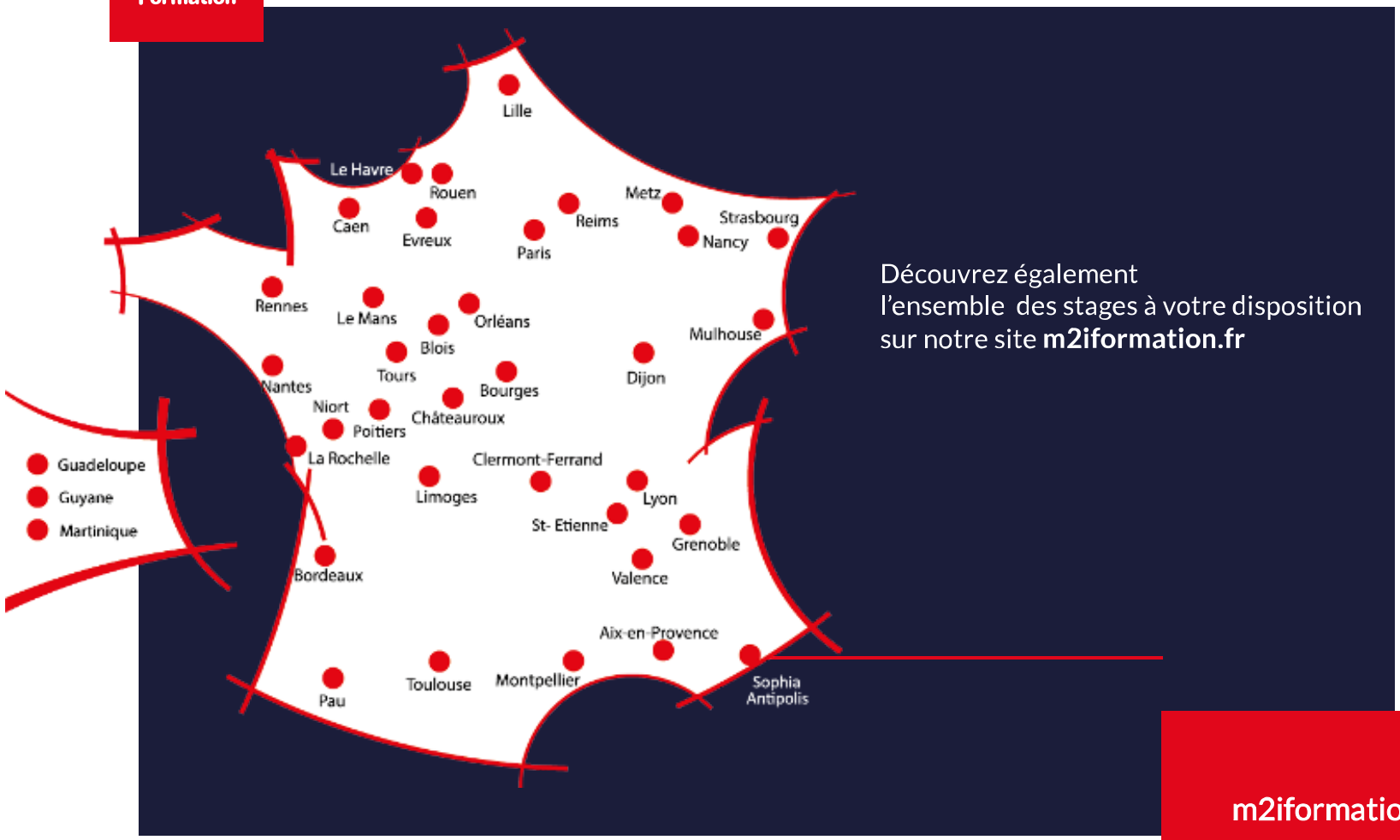
Utilisation combinée pour une analyse approfondie

- **Trouver des processus et tuer :**
  Tu peux combiner plusieurs commandes pour effectuer des analyses et prendre des actions. Par exemple, pour identifier un processus consommateur de mémoire et le tuer :

```
ps aux --sort=-%mem | head
kill -9 <PID>
```

- **Vérification de l'arbre des processus avec détails :**
  Pour voir la hiérarchie d'un processus spécifique, par exemple, avec son PID, utilise `pstree` :

```
pstree -p <PID>
```



Découvrez également
l'ensemble des stages à votre disposition
sur notre site **m2iformation.fr**

m2iformation.fr