Administration et Optimisation des Performances

CentOS / Ubuntu Systems

Guide complet pour ingénieurs système

Table des Matières

1. Surveillance des Ressources Système

Un ingénieur système doit constamment surveiller les ressources pour maintenir des performances optimales.

1.1 Surveillance de la Mémoire (RAM)

Commandes essentielles pour surveiller l'utilisation de la mémoire :

```
free -h                     # Utilisation mémoire en format lisible
vmstat 1                    # Statistiques mémoire en temps réel
cat /proc/meminfo           # Informations détaillées sur la mémoire
top -o %MEM                 # Trier par utilisation mémoire
```

1.2 Surveillance du CPU

Outils pour monitorer l'utilisation du processeur :

```
top                         # Vue en temps réel des processus
htop                        # Version améliorée et interactive de top
mpstat -P ALL 1             # Statistiques par cœur CPU
lscpu                       # Informations détaillées sur le CPU
uptime                      # Charge moyenne du système
```

1.3 Surveillance du Disque

Commandes pour surveiller l'espace disque et les performances I/O :

```
df -h                       # Espace disque utilisé par partition
du -sh /var/log/*           # Taille des dossiers
iotop                       # Activité I/O en temps réel (root requis)
iostat -x 1                 # Statistiques I/O détaillées
lsblk                       # Liste des périphériques de bloc
```

1.4 Surveillance du Réseau

Outils de monitoring réseau :

```
iftop                       # Bande passante en temps réel
nethogs                     # Utilisation réseau par processus
ss -tunap                   # Connexions réseau actives
ip -s link                  # Statistiques des interfaces réseau
nload                       # Graphique de la bande passante
```

2. Analyse des Logs

Les logs système sont essentiels pour diagnostiquer les problèmes et surveiller l'activité du système.

2.1 Commandes de Base

```
# Logs système (systemd)
journalctl -xe             # Logs récents avec détails
journalctl -f              # Suivre les logs en temps réel
journalctl -u nginx        # Logs d'un service spécifique
journalctl --since "1 hour ago"  # Logs de la dernière heure

# Logs traditionnels
tail -f /var/log/syslog    # Ubuntu/Debian
tail -f /var/log/messages  # CentOS/RHEL
dmesg | tail               # Messages du noyau
cat /var/log/auth.log      # Logs d'authentification
```

2.2 Analyse Avancée

```
# Rechercher des erreurs
journalctl -p err -b       # Erreurs depuis le dernier boot
grep -i error /var/log/syslog  # Recherche dans les logs

# Rotation des logs
logrotate -f /etc/logrotate.conf  # Forcer la rotation
```

3. Optimisation des Performances

3.1 Gestion des Services

```
# Lister les services actifs
systemctl list-units --type=service --state=running

# Désactiver les services inutiles
systemctl disable service_name
systemctl stop service_name

# Analyser le temps de boot
systemd-analyze blame
systemd-analyze critical-chain
```

3.2 Optimisation des Processus

```
# Identifier les processus gourmands
ps aux --sort=-%mem | head -10    # Top 10 RAM
ps aux --sort=-%cpu | head -10    # Top 10 CPU

# Gérer les priorités
nice -n 10 command                # Lancer avec priorité basse
renice -n -5 -p PID               # Augmenter la priorité
```

3.3 Paramètres Kernel (sysctl)

Optimisation des paramètres système :

```
# Voir tous les paramètres
sysctl -a

# Paramètres courants à optimiser
# /etc/sysctl.conf

# Augmenter la taille du cache
vm.swappiness=10                  # Réduire l'utilisation du swap
vm.vfs_cache_pressure=50          # Conserver le cache plus longtemps

# Optimisation réseau
net.ipv4.tcp_fin_timeout=30
net.ipv4.tcp_keepalive_time=300
net.core.netdev_max_backlog=5000

# Appliquer les changements
sysctl -p
```

3.4 Gestion des Packages

```
# Ubuntu/Debian
apt update && apt upgrade -y
apt autoremove
apt autoclean

# CentOS/RHEL
yum update -y
yum clean all
yum autoremove
```

4. Monitoring Continu

4.1 Outils de Monitoring

Solutions professionnelles :

Outil	Type	Usage
Nagios	Complet	Surveillance infrastructure
Zabbix	Complet	Monitoring avancé avec alertes
Prometheus	Métriques	Collecte de métriques
Grafana	Visualisation	Dashboards et graphiques
Netdata	Temps réel	Monitoring instantané
Glances	Système	Vue d'ensemble en terminal

4.2 Script de Surveillance

```
#!/bin/bash
# check_system.sh - Script de vérification quotidien

echo "======================================"
echo "Rapport Système - $(date)"
echo "======================================"

echo -e "\n=== Charge CPU ==="
uptime

echo -e "\n=== Utilisation Mémoire ==="
free -h

echo -e "\n=== Espace Disque ==="
df -h | grep -v tmpfs

echo -e "\n=== Top 5 Processus CPU ==="
ps aux --sort=-%cpu | head -6

echo -e "\n=== Top 5 Processus RAM ==="
ps aux --sort=-%mem | head -6

echo -e "\n=== Connexions Réseau ==="
ss -s

# Sauvegarder dans un fichier
# Utilisation: ./check_system.sh > /var/log/system_report_$(date +%Y%m%d).log
```

5. Optimisations Spécifiques

5.1 Base de Données

Optimisation MySQL/MariaDB :

```
# Configuration /etc/mysql/my.cnf
[mysqld]
innodb_buffer_pool_size = 2G       # 70% de la RAM disponible
max_connections = 200
query_cache_size = 64M
tmp_table_size = 64M
max_heap_table_size = 64M

# Analyser les requêtes lentes
slow_query_log = 1
long_query_time = 2

# Redémarrer MySQL
systemctl restart mysql
```

5.2 Serveur Web

Optimisation Nginx :

```
# /etc/nginx/nginx.conf
worker_processes auto;
worker_connections 2048;

# Compression
gzip on;
gzip_comp_level 5;
gzip_types text/plain text/css application/json application/javascript;

# Cache
proxy_cache_path /var/cache/nginx levels=1:2 keys_zone=my_cache:10m;

# Buffers
client_body_buffer_size 10K;
client_header_buffer_size 1k;
large_client_header_buffers 2 1k;
```

5.3 Sécurité

```
# Fail2ban - Protection contre brute-force
fail2ban-client status
fail2ban-client status sshd

# Firewall (UFW - Ubuntu)
ufw status
ufw allow 22/tcp
ufw allow 80/tcp
ufw allow 443/tcp
ufw enable

# Firewall (firewalld - CentOS)
firewall-cmd --list-all
firewall-cmd --add-service=http --permanent
firewall-cmd --reload

# Audit de sécurité
lynis audit system
```

6. Planification et Automatisation

6.1 Cron Jobs

Automatiser les tâches répétitives :

```
# Éditer le crontab
crontab -e

# Exemples de tâches planifiées

# Sauvegarde quotidienne à 2h du matin
0 2 * * * /usr/local/bin/backup.sh

# Nettoyage des fichiers temporaires (tous les jours)
0 3 * * * /usr/bin/find /tmp -type f -atime +7 -delete

# Vérification système hebdomadaire (dimanche à 1h)
0 1 * * 0 /usr/local/bin/check_system.sh > /var/log/weekly_check.log

# Mise à jour des packages (1er du mois à 4h)
0 4 1 * * apt update && apt upgrade -y

# Rotation des logs personnalisés
0 0 * * * /usr/sbin/logrotate /etc/logrotate.conf
```

6.2 Systemd Timers

Alternative moderne aux cron jobs :

```
# Lister les timers actifs
systemctl list-timers

# Créer un timer personnalisé
# /etc/systemd/system/backup.timer
[Unit]
Description=Backup quotidien

[Timer]
OnCalendar=daily
Persistent=true

[Install]
WantedBy=timers.target

# Activer le timer
systemctl enable backup.timer
systemctl start backup.timer
```

7. Benchmarking

Tests de performance pour évaluer les capacités du système.

7.1 Tests CPU

```
# Sysbench - Test CPU
sysbench cpu --cpu-max-prime=20000 run

# Stress test
stress --cpu 4 --timeout 60s
```

7.2 Tests Disque

```
# Test d'écriture
dd if=/dev/zero of=/tmp/test.img bs=1G count=1 oflag=direct

# Test de lecture
dd if=/tmp/test.img of=/dev/null bs=1G count=1 iflag=direct

# FIO - Test avancé
fio --name=random-write --ioengine=posixaio --rw=randwrite --bs=4k \
    --size=1g --numjobs=1 --runtime=60 --time_based
```

7.3 Tests Réseau

```
# iperf3 - Test de bande passante
# Serveur
iperf3 -s

# Client
iperf3 -c server_ip -t 30

# Ping et latence
ping -c 100 server_ip
mtr server_ip              # Traceroute amélioré
```

8. Gestion des Ports Réseau

Contrôler et gérer les ports utilisés sur le système.

8.1 Commandes de Vérification

```
# ss (moderne, remplace netstat)
ss -tuln                    # Tous les ports en écoute
ss -tulnp                   # Avec les processus (root requis)
ss -tan                     # Toutes les connexions TCP
ss -tan | grep :443         # Vérifier un port spécifique

# netstat (ancienne méthode)
netstat -tuln               # Tous les ports en écoute
netstat -tulnp              # Avec les processus
netstat -an | grep LISTEN   # Seulement les ports en écoute

# lsof (list open files)
lsof -i                     # Toutes les connexions réseau
lsof -i :80                 # Qui utilise le port 80
lsof -i TCP:22              # Port SSH spécifique
```

8.2 Scanner de Ports

```
# nmap - Scanner de ports
nmap localhost              # Scanner les ports locaux
nmap -p 1-65535 localhost   # Tous les ports
nmap -p 80,443,22 localhost # Ports spécifiques

# netcat - Test de connexion
nc -zv localhost 80         # Tester si le port est ouvert
nc -zv google.com 443       # Test externe
```

8.3 Identification du Processus

```
# Trouver quel processus utilise un port
sudo lsof -i :3306
sudo ss -tulnp | grep :3306
sudo netstat -tulnp | grep :3306

# Avec fuser
sudo fuser 3306/tcp         # Affiche le PID

# Tuer un processus sur un port
sudo kill $(sudo lsof -t -i:8080)
sudo fuser -k 8080/tcp      # Force kill
```

8.4 Ports Standards

Port	Service	Description
22	SSH	Connexion sécurisée
80	HTTP	Web non-sécurisé
443	HTTPS	Web sécurisé (SSL/TLS)
3306	MySQL	Base de données MySQL
5432	PostgreSQL	Base de données PostgreSQL
6379	Redis	Cache en mémoire
27017	MongoDB	Base de données NoSQL

8080	HTTP Alt	Serveur web alternatif
9090	Prometheus	Monitoring
3000	Grafana	Visualisation

8.5 Gestion du Firewall

```
# Ubuntu (ufw)
sudo ufw status
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw deny 23/tcp       # Bloquer telnet
sudo ufw enable

# CentOS (firewalld)
sudo firewall-cmd --list-ports
sudo firewall-cmd --add-port=80/tcp --permanent
sudo firewall-cmd --add-service=http --permanent
sudo firewall-cmd --reload

# iptables
sudo iptables -L -n
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

9. Configuration Nginx et Services Systemd

9.1 Structure des Répertoires Nginx

```
/etc/nginx/
├── nginx.conf   // fichier de configuration de nginx
├──
│
├── sites-available/
│   ├── default    // ici se trouve la meme chose que en bas mais c est le
│   │              // dossier par defaut je peut l'utiliser ou je creer ma
│   │              // part comme cequi est en bas
│   ├── app.conf   // ici se trouve ou je trouve le port 80 de nginx les
│                  // reverse proxy ect ..ce dossier je le creer pour mon site
│
├── sites-enabled/
│   ├── default -> ../sites-available/default
│   ├── app.conf -> ../sites-available/monsite.conf
│
│
├── ssl/
    ├── cert.pem
    ├── key.pem   // cle des certificat
```

9.2 Configuration du Reverse Proxy avec Nginx

Créer et activer app.conf :

```
# Créer le fichier de configuration
sudo nano /etc/nginx/sites-available/app.conf

# Activer le site en créant un lien symbolique
sudo ln -s /etc/nginx/sites-available/app.conf /etc/nginx/sites-enabled/

# Créer la route où je mets mon frontend
sudo mkdir -p /var/www/app  // ici app = html car mon front est dans html
```

Configuration complète du fichier app.conf :

```
sudo vi /etc/nginx/sites-available/app.conf

server {
    listen 80;
    server_name 192.168.3.49;

    # Taille max des uploads
    client_max_body_size 20M;

    # Sert les fichiers statiques Angular
    root /var/www/html;
    index index.html;

    # Fallback Angular routes => index.html
    location / {
        try_files $uri $uri/ /index.html;
    }

    # Proxy vers le backend Spring Boot pour les appels API
    location /api/ {
        proxy_pass http://192.168.3.49:8080;

        # Corriger erreur 417 Expectation Failed
        proxy_set_header Expect "";

        # Headers utiles
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # WebSocket support (facultatif)
        proxy_http_version 1.1;
```

```
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
        }
    }
```

Rôle de ce service :

• écouter HTTP/HTTPS (80/443) • recevoir les requêtes web • les rediriger vers ton app (proxy_pass)

9.3 Création d'un Nouveau Service Web

Étape 1 : Créer le dossier du site

```
sudo mkdir -p /var/www/monservice
sudo chown -R www-data:www-data /var/www/monservice

# Exemple de fichier de test :
echo "<h1>Mon service fonctionne ■</h1>" | sudo tee /var/www/monservice/index.html
```

Étape 2 : Créer le fichier de configuration Nginx

```
sudo nano /etc/nginx/sites-available/monservice

# Exemple minimal (HTTP)
server {
    listen 80;
    server_name monservice.local;

    root /var/www/monservice;
    index index.html index.htm;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

Étape 3 : Activer le service (site)

```
sudo ln -s /etc/nginx/sites-available/monservice /etc/nginx/sites-enabled/
```

Étape 4 : Tester la configuration

```
sudo nginx -t

# ✔■ Si tout est OK, recharge Nginx :
sudo systemctl reload nginx
```

9.4 Service Systemd pour Application Spring Boot

Fichier de service : /etc/systemd/system/mon-app.service

Rôle : • lancer ton backend Spring Boot • gérer Java (start / stop / restart) • dépendre de MySQL • tourner en arrière-plan au boot ■ C'est le système Linux qui s'en occupe.

```
sudo vi /etc/systemd/system/mon-app.service

[Unit]
Description=Mon Application Spring Boot
After=mysql.service
Requires=mysql.service

[Service]
Type=simple
User=
Group=
WorkingDirectory=/opt/mon-app
# je met le parcourt de l'app et le nom du jar
ExecStart=/usr/bin/java -jar /opt/mon-app/mon-app-1.0.0.jar
EnvironmentFile=/opt/mon-app/config/application.env
Restart=always
RestartSec=10
StandardOutput=journal
StandardError=journal

# Limites de ressources
LimitNOFILE=65536
TimeoutStopSec=30

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

Gestion du Service

Parcourt du service : /etc/systemd/system# (tu va voir mon-app.service tu tape les commandes ci-dessous sans .service et ça va gérer)

```
# Recharger systemd
sudo systemctl daemon-reload

# Activer le service
sudo systemctl enable mon-app

# Démarrer le service
sudo systemctl start mon-app

# Vérifier le statut
sudo systemctl status mon-app

# Voir les logs
sudo journalctl -u mon-app -f
```

10. Git et Pull Requests

Un développeur peut faire une pull request de deux façons : principalement par ligne de commande Git puis sur l'interface web (GitHub/GitLab), ou directement via l'interface web.

10.1 Méthode 1 : Ligne de Commande + Interface Web (LA PLUS COURANTE)

Étape 1 : Cloner le projet

```
# Cloner le repository
git clone https://github.com/username/project.git
cd project
```

Étape 2 : Créer une nouvelle branche

```
# Créer et basculer sur une nouvelle branche
git checkout -b feature/ma-nouvelle-fonctionnalite

# Ou en deux commandes
git branch feature/ma-nouvelle-fonctionnalite
git checkout feature/ma-nouvelle-fonctionnalite
```

Étape 3 : Modifier le code

```
# Le développeur modifie ses fichiers avec son éditeur
# Par exemple : vim, nano, VSCode, etc.
nano fichier.py
```

Étape 4 : Vérifier les modifications

```
# Voir les fichiers modifiés
git status

# Voir les changements en détail
git diff
```

Étape 5 : Ajouter les fichiers modifiés (Staging)

```
# Ajouter un fichier spécifique
git add fichier.py

# Ajouter tous les fichiers modifiés
git add .

# Ajouter plusieurs fichiers
git add fichier1.py fichier2.py fichier3.js
```

Étape 6 : Commit les changements

```
# Commit avec un message descriptif
git commit -m "Ajout de la fonctionnalité de connexion utilisateur"

# Ou commit avec éditeur pour message détaillé
git commit
```

Étape 7 : Pousser la branche vers GitHub/GitLab

```
# Pousser la branche vers le remote
git push origin feature/ma-nouvelle-fonctionnalite

# Si c'est la première fois (créer la branche distante)
git push -u origin feature/ma-nouvelle-fonctionnalite
```

Étape 8 : Créer la Pull Request sur l'interface web

Sur GitHub : 1. Aller sur https://github.com/username/project 2. Un message apparaît automatiquement : "Compare & pull request" 3. Cliquer sur le bouton vert "Compare & pull request" 4. Remplir : Titre de la PR, Description détaillée, Reviewers, Labels, Milestone, etc. 5. Cliquer sur "Create pull request" Sur GitLab : 1. Aller sur le projet GitLab 2. Menu "Merge Requests" → "New merge request" 3. Sélectionner la branche source et destination 4. Remplir les détails 5. Cliquer sur "Create merge request"

10.2 Méthode 2 : Directement sur l'Interface Web

Sur GitHub - Interface Web Complète : 1. Naviguer vers le fichier sur GitHub 2. Cliquer sur l'icône "═■ Edit" (crayon) 3. Modifier le code directement dans l'éditeur web 4. Scroller en bas 5. Remplir : Commit message, Description (optionnel) 6. Sélectionner "Create a new branch and start a pull request" 7. Cliquer "Propose changes" 8. Remplir les détails de la PR 9. Cliquer "Create pull request"

10.3 Workflow Complet Détaillé

Scénario Réel : Développeur travaillant sur une fonctionnalité

```
# 1. Mettre à jour la branche principale
git checkout main
git pull origin main

# 2. Créer une branche pour la fonctionnalité
git checkout -b fix/correction-bug-login

# 3. Travailler sur le code
# ... modifications des fichiers ...

# 4. Vérifier ce qui a changé
git status
git diff

# 5. Ajouter les modifications
git add src/auth/login.py
git add tests/test_login.py

# 6. Commit avec un bon message
git commit -m "Fix: Correction du bug d'authentification

- Correction de la validation du token
- Ajout de tests unitaires
- Mise à jour de la documentation

Fixes #123"

# 7. Pousser vers GitHub
git push origin fix/correction-bug-login

# 8. Aller sur GitHub et créer la PR
```

10.4 Commandes Git Utiles pour les PR

Avant de faire la PR

```
# Mettre à jour depuis la branche principale
git checkout main
git pull origin main
git checkout ma-branche
git merge main  # Ou git rebase main

# Résoudre les conflits si nécessaire
git status
# ... résoudre les conflits ...
git add .
git commit -m "Merge main into feature branch"
git push origin ma-branche
```

Modifier une PR existante

```
# Faire des modifications supplémentaires
git add fichier.py
git commit -m "Ajout de tests supplémentaires"
git push origin ma-branche
# La PR se met à jour automatiquement !
```

Squash des commits (nettoyer l'historique)

```
# Combiner les 3 derniers commits en un seul
git rebase -i HEAD~3

# Dans l'éditeur, changer "pick" en "squash" pour les commits à fusionner
# Sauvegarder et quitter

# Forcer le push (attention !)
git push origin ma-branche --force
```

10.5 Bonnes Pratiques pour les Pull Requests

1. Message de commit clair

```
# ■ Mauvais
git commit -m "fix"
git commit -m "update"

# ■ Bon
git commit -m "Fix: Correction de la validation email dans le formulaire"
git commit -m "Feature: Ajout du système de notifications en temps réel"
```

2. Branche bien nommée

```
# ■ Mauvais
git checkout -b test
git checkout -b nouvelle-branche

# ■ Bon
git checkout -b feature/user-authentication
git checkout -b fix/payment-validation-bug
git checkout -b hotfix/security-vulnerability
```

3. PR avec description détaillée - Template de PR :

```
## Description
Ajout d'un système d'authentification à deux facteurs

## Type de changement
- [ ] Bug fix
- [x] Nouvelle fonctionnalité
- [ ] Breaking change
- [ ] Documentation

## Changements
- Ajout de la table 2FA en base de données
- Implémentation de l'envoi de code par SMS
- Interface utilisateur pour activer/désactiver 2FA
- Tests unitaires et d'intégration

## Tests
- [x] Tests unitaires passent
- [x] Tests d'intégration passent
- [x] Testé manuellement

## Captures d'écran
[Si applicable]

## Checklist
- [x] Code respecte les standards du projet
- [x] Documentation mise à jour
- [x] Tests ajoutés
- [x] Pas de warning ni erreur
```

10.6 Outils CLI pour Faciliter les PR

GitHub CLI (gh)

```
# Installer GitHub CLI
# Ubuntu
sudo apt install gh

# Se connecter
gh auth login

# Créer une PR directement depuis la ligne de commande
gh pr create --title "Fix authentication bug" --body "Description détaillée"

# Lister les PRs
gh pr list

# Voir une PR
gh pr view 123

# Merger une PR
gh pr merge 123
```

Git Flow

```
# Installer git-flow
sudo apt install git-flow

# Initialiser
git flow init

# Créer une feature
git flow feature start ma-fonctionnalite

# Terminer une feature (crée automatiquement une PR)
git flow feature finish ma-fonctionnalite
```

10.7 Résumé : Les Deux Approches

Aspect	Ligne de Commande	Interface Web
Flexibilité	■ Totale	■■ Limitée
Rapidité	■ Rapide	■■ Plus lent
Modifications multiples	■ Facile	■ Difficile
Gestion des conflits	■ Puissant	■■ Basique
Connaissances Git	■■ Oui	■ Non
Idéal pour	Développeurs expérimentés	Petites modifs

En pratique, la plupart des développeurs utilisent : • 95% du temps : Ligne de commande pour coder + Interface web pour créer la PR • 5% du temps : Interface web directe pour des petites corrections (typos, docs)

Conclusion

Un administrateur système efficace combine tous ces outils et techniques pour maintenir un système Linux performant, stable et sécurisé. La surveillance proactive, l'optimisation continue et l'automatisation des tâches répétitives sont les clés du succès. Points clés à retenir : • Surveillez régulièrement les ressources système (CPU, RAM, disque, réseau) • Analysez les logs pour anticiper les problèmes • Optimisez les services et les paramètres kernel selon vos besoins • Mettez en place un système de monitoring continu • Automatisez les tâches répétitives avec cron ou systemd timers • Sécurisez votre système avec un firewall et fail2ban • Effectuez des benchmarks pour évaluer les performances • Configurez correctement Nginx pour vos applications web • Gérez vos services avec systemd pour une meilleure fiabilité • Utilisez Git et les Pull Requests pour un développement collaboratif efficace • Documentez vos configurations et procédures La maîtrise de ces compétences vous permettra de gérer efficacement des infrastructures Linux, qu'elles soient simples ou complexes, et d'assurer leur disponibilité et leurs performances optimales. La combinaison de ces connaissances système avec les bonnes pratiques de développement (Git, CI/CD) fait de vous un DevOps complet.

Ressources Additionnelles

• Documentation officielle : Ubuntu (help.ubuntu.com), CentOS (docs.centos.org) • Man pages : man command_name • Communautés : Stack Overflow, Server Fault, Reddit (r/linux, r/linuxadmin) • Livres recommandés : "UNIX and Linux System Administration Handbook", "Linux Performance Tools" • Certifications : LPIC, RHCSA, RHCE