Introduction aux systèmes GNU/Linux

Module IntroLinux – S25E02

Philippe Latu / Université de Toulouse
inetdoc.net

Objectif DevOps

De l'idée à la production.

→ Un flux continu de valeur !

Créer un pipeline fluide et continu de la conception à la mise en production, en mettant l'accent sur la livraison rapide de valeur.

Le plan

- Virtualisation et conteneurs
- Logiciel Libre et Licences
- Projets OpenSource

Virtualisation et conteneurs

« Les machines virtuelles ont joué le rôle de brise-glace pour les conteneurs. »

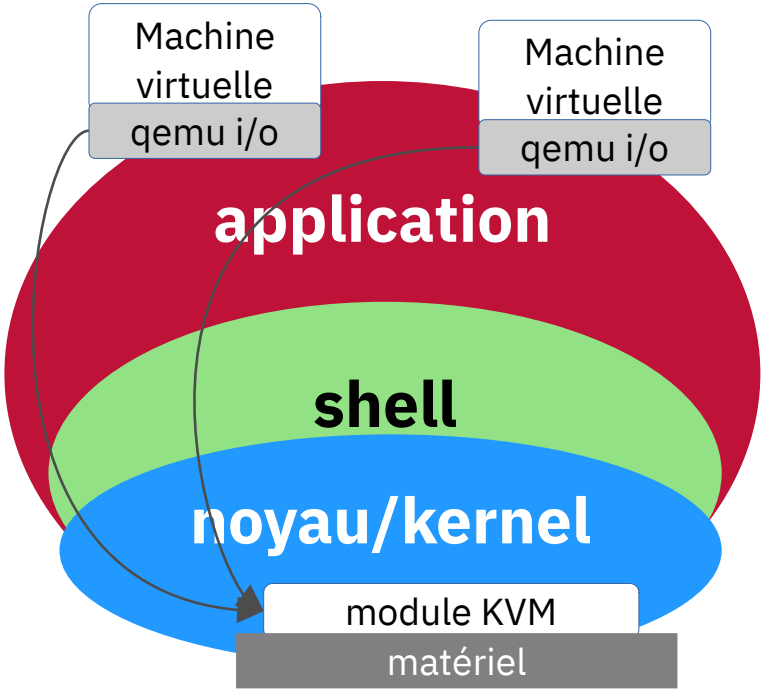
Distinguer virtualisation et para-virtualisation

Virtualisation sans hyperviseur

- Émulation du matériel dans l'espace utilisateur
- Performances limitées
- Adapté aux cibles matérielles obsolètes

Para-virtualisation avec hyperviseur

- Communication directe entre machine virtuelle et matériel du système hôte
- Performances identiques entre système hôte et système virtualisé
- Périphériques « dédiés » à la virtualisation



Définir un hyperviseur

Hyperviseurs de type 1 ou 2

Type 1

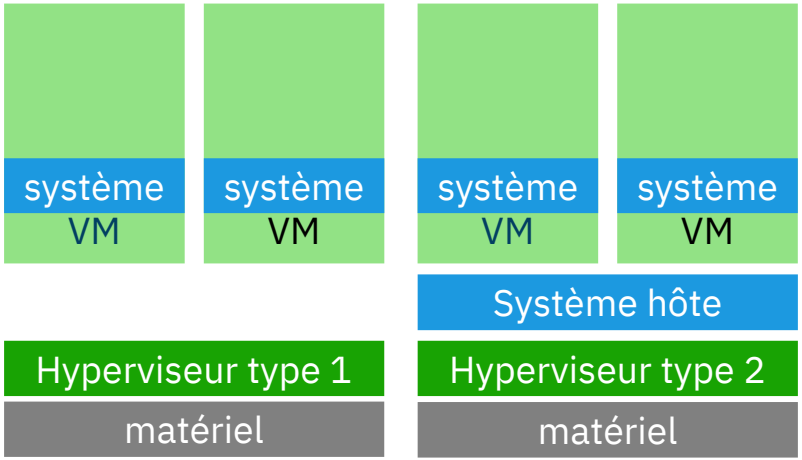
- Accès direct au matériel depuis les machines virtuelles
- Exemple : VMware ESX

Type 2

- Services assurés par le système hôte
- Exemple : KVM + Open vSwitch

Comment différencier les 2 types ?

- Accès à la console → type 2



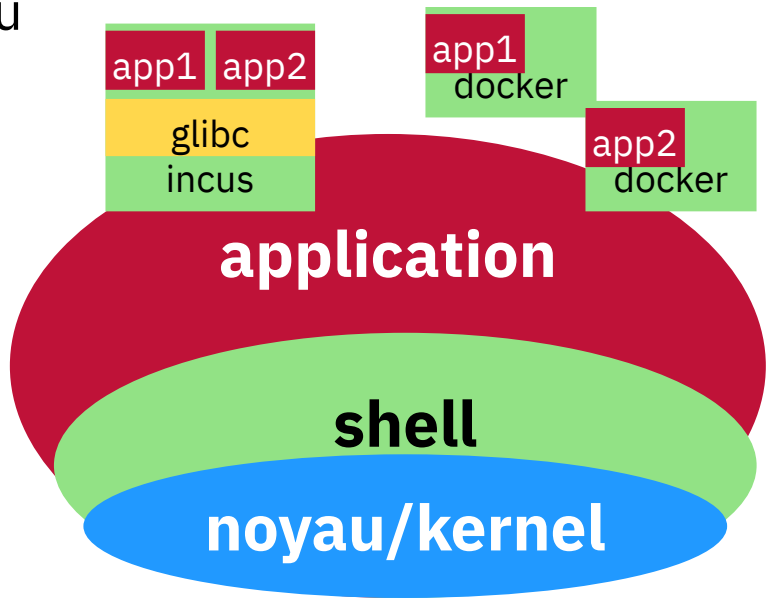
Distinguer conteneurs système et micro services

Conteneurs système → Incus

- Système de fichiers complet sans noyau
- Données stockées dans le conteneur ou en dehors via montage
- Pile d'applications hébergées
- Coût d'administration système complet

Micro services → Docker

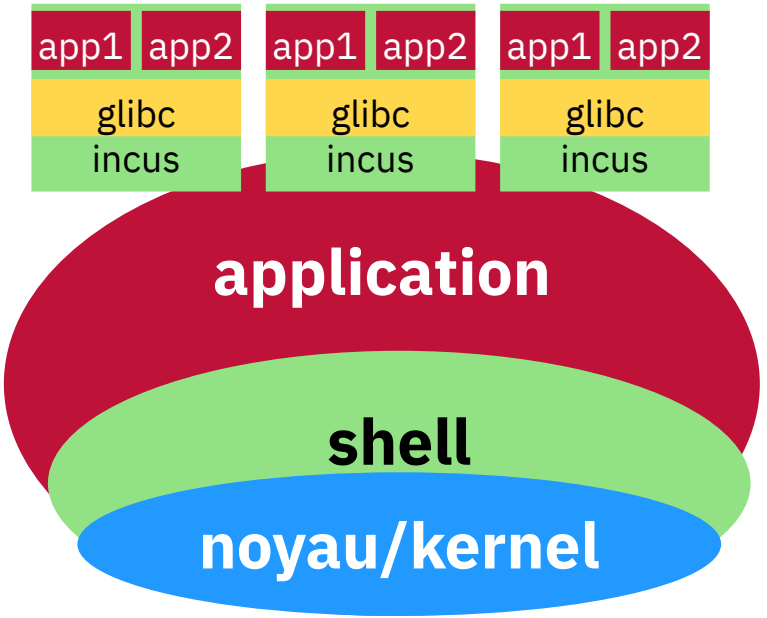
- Une application par conteneur Docker
- Données stockées en dehors
- Pile d'applications → pile de conteneurs



Introduire le scénario des manipulations

Opération sauver C-3PO

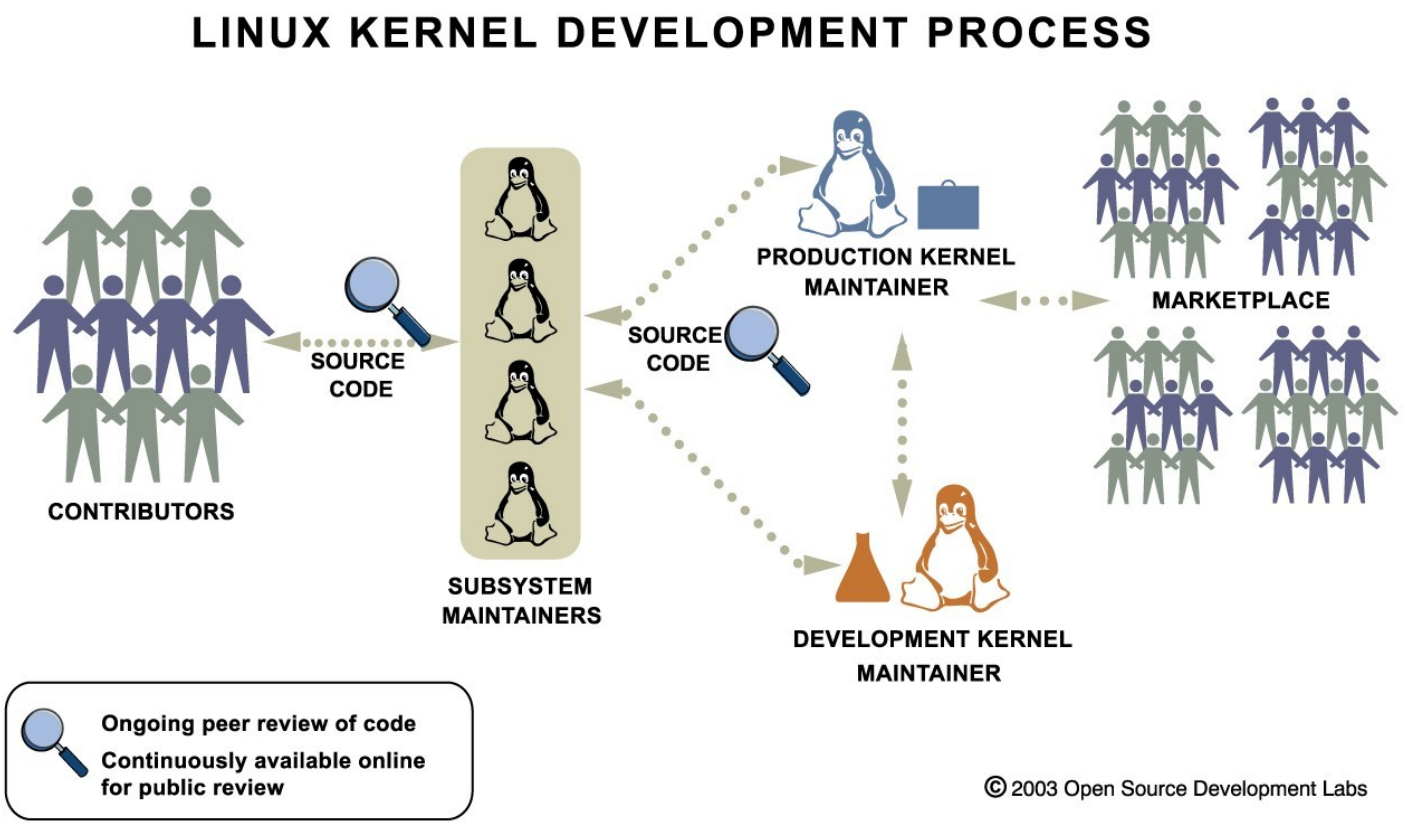
- Instancier des services dans les conteneurs systèmes Incus
- Gérer les conteneurs dans une machine virtuelle
- Identifier les fonctions d'un profil
  - stockage et réseau
- Illustrer le fonctionnement des services « cloud »



Logiciel libre et Licences

« Je compare souvent l'open source à la science. La science a adopté cette notion de développer des idées ouvertement et d'améliorer les idées des autres. C'est ce qui a fait de la science ce qu'elle est aujourd'hui et a rendu possibles les incroyables avancées que nous avons connues. » - Linus Torvalds

Décrire le processus de développement Linux



Décrire le processus de développement Linux

Collaboratif et ouvert

- Le plus grand projet de logiciel libre au monde
- Plus de 13 000 développeurs contribuent à l'échelle mondiale
- Processus de développement 24/7/365

Cycle de publication basé sur le temps

- Nouvelle version toutes les 9 à 10 semaines
- Les versions ne sont pas retardées pour des fonctionnalités spécifiques
- Cycles de développement et d'intégration parallèles

Décrire le processus de développement Linux

Rythme de publication en phases structurées

- Fenêtre de fusion de 2 semaines pour les nouveaux développements majeurs
- Phase de Release Candidate (RC) pour la stabilisation
- Versions RC hebdomadaires jusqu'à ce que le noyau soit considéré comme prêt

Hiérarchie des mainteneurs

- Linus Torvalds supervise l'arbre principal du noyau
- Les responsables de sous-systèmes gèrent des domaines spécifiques
- Les demandes Git PR sont utilisées pour la soumission de code

Différencier code source et code exécutable

Tout programme est écrit dans un langage

- Le noyau Linux est écrit en Langage C
- Le code source n'est pas directement utilisable

Chaîne de développement

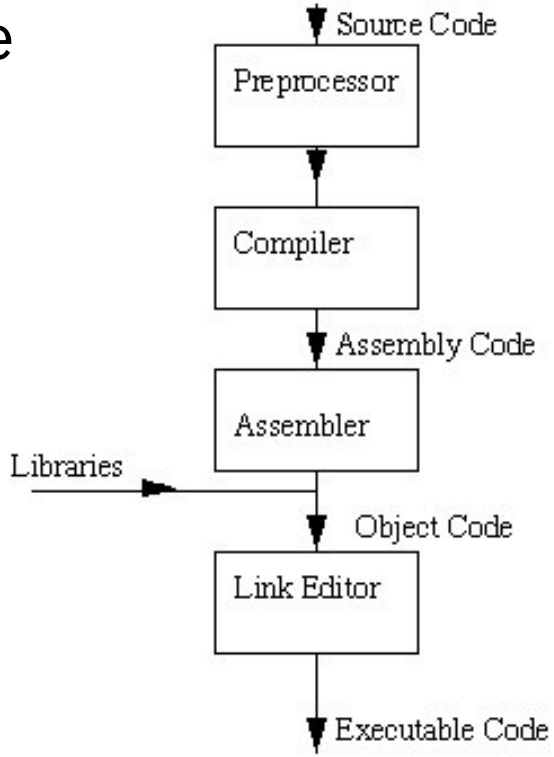
- Transformation du code source en code exécutable
- Transformation inverse «impossible»
- Code exécutable = binaire

Logiciel propriétaire

- Droit d'utilisation d'un code exécutable

Logiciel libre

- Droit d’accès au code source



Décrire les licences de logiciel libre

Licence BSD → restrictions possibles

- Création de versions propriétaires autorisée
- Restrictions possibles sur les droits de redistribution
- Restrictions rarement appliquées dans les faits

Licence GNU → Copyleft

- Copyright != Copyleft
- Principe de protection du Logiciel libre et des concepteurs
- Restrictions interdites sur les conditions de redistribution

Décrire les principes du copyleft

- Appliquer un copyright sur le logiciel
- Fixer les conditions de distribution

  « Donner à tout utilisateur le droit d'utiliser, de modifier et de redistribuer le programme sans changer les conditions de distribution »
- Code source et libertés associées non dissociables

  http://www.gnu.org/copyleft/copyleft.fr.html

Projets Open Source

« Avec l'open source, vous avez le droit de contrôler votre propre destin » - Linus Torvalds

Décrire les cadres de développement

Fondations → écosystèmes structurés

- Open Source Initiative
  - https://opensource.org
- Linux Foundation
  - https://www.linuxfoundation.org
- Mozilla Foundation
  - https://foundation.mozilla.org
- The Document Foundation
  - https://www.documentfoundation.org

Décrire les évolutions « métier »

DevOps Mantra

- Toujours plus d’automatisation
- Toujours plus d’applications intégrées ou dépendantes
- Tout est programmable :
  « Le déploiement d'infrastructures et de logiciels utilise les mêmes processus. »

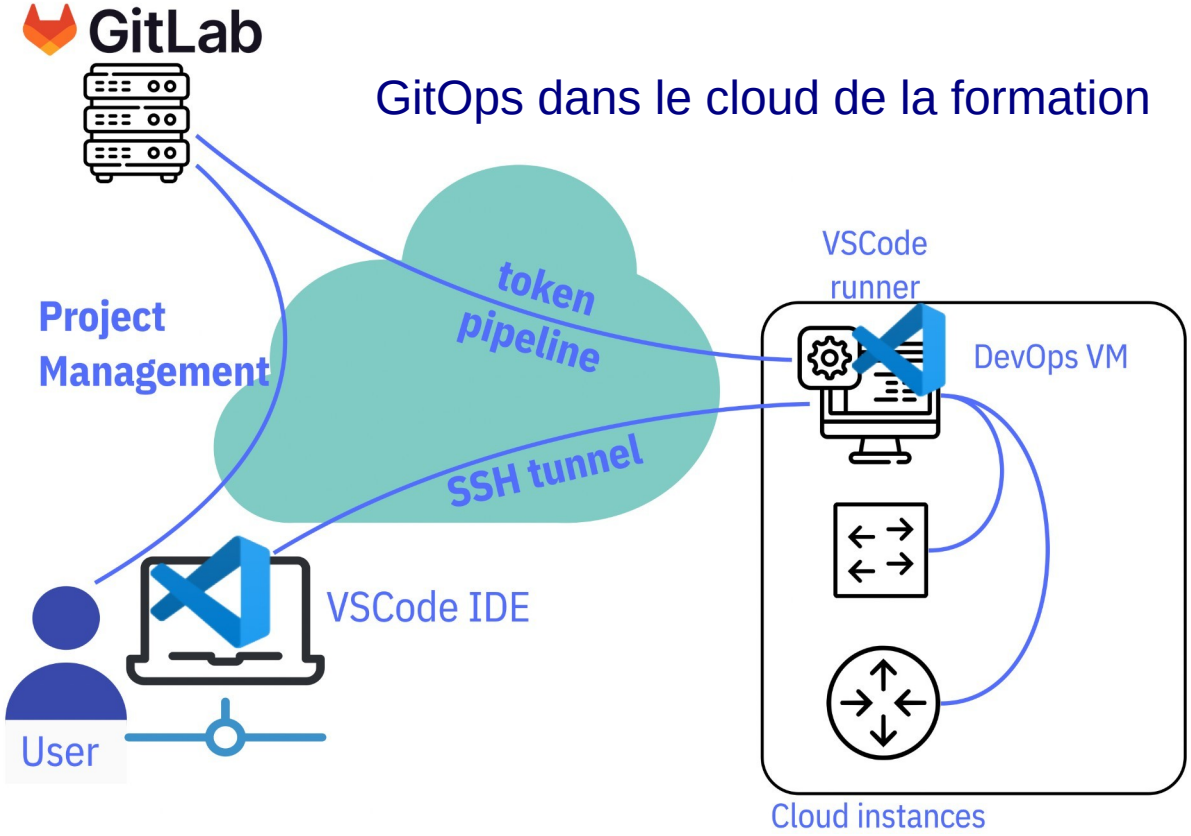
Fusion des métiers du développement et de l’administration système

- Vidéo → DevOps vs SRE

Définir les concepts CI/CD

- Intégration continue – *continuous integration*
  Déposer souvent le code dans un système de contrôle de version. Toute modification déclenche une instance automatisée de génération et de test.
- Livraison continue – *continuous delivery*
  Automatiser le processus de publication du logiciel. Un projet amont peut lancer la construction des paquets pour toutes les distributions.
- Déploiement continu – *continuous deployment*
  Automatiser complètement l'intégration et la livraison sans aucune étape manuelle.

Utiliser les concepts CI/CD



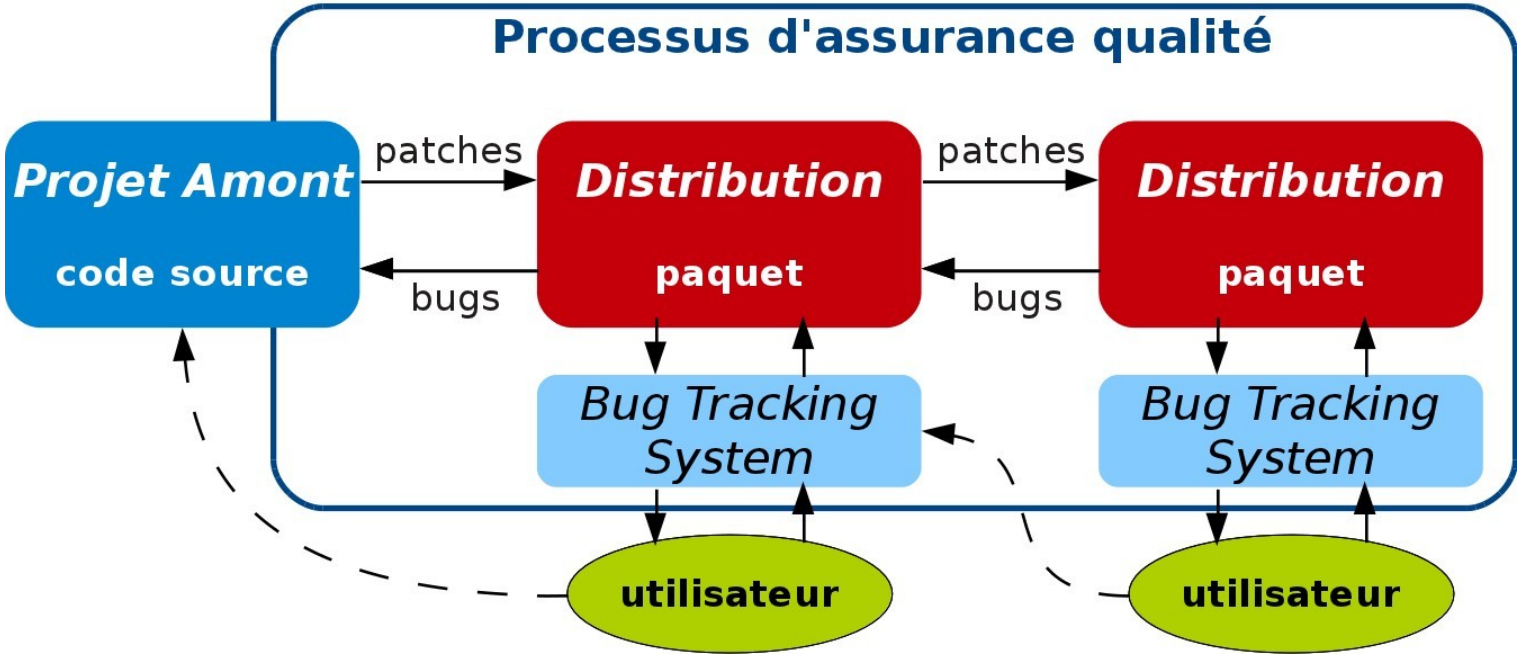
Définir les fonctions d’une distribution

- Distribution → canal de diffusion du logiciel libre
- Association de 3 éléments
  - Noyau
  - Un ou plusieurs Shells
  - Un ensemble d'applications
- Applications distribuées sous forme de paquets
  - Code binaire exécutable
  - Configuration type
  - Base de données
- Gestionnaire de paquets
  - Pierre angulaire de la vie d'une distribution

debian

Définir les fonctions d'une distribution

- Distribution → filiations et processus qualité
- Modèle du pipeline



Choisir une distribution

- Facilité d'adaptation
  - Obtenir une configuration type par contexte
  - Bénéficier de l’assurance qualité
- Continuité lors des mises à jour
  - Cohérence des évolutions et corrections
  - Évolution continue
    - Continuité de service
    - Haute disponibilité

Pourquoi Debian GNU/Linux ?

- Principes du logiciel libre selon Debian
  - Règles à suivre pour garantir qu'un logiciel est bien libre
  - http://www.debian.org/social_contract.html
- Charte Debian
  - Procédure qualité du projet
  - http://www.debian.org/devel/index.fr.html
- Gestionnaire de paquets APT
  - Synthèse de toutes les qualités de la distribution
  - Continuité indépendante des versions
  - Adaptabilité en séparant la configuration de l'application
  - Automatisation de la publication des mises à jour

debian

Décrire les points forts de Debian dans le cloud

- Stabilité et fiabilité reconnues
- Large base de paquets logiciels disponibles
- Processus de développement communautaire transparent
- Support à long terme des versions stables
- Flexibilité d'installation et de configuration

Exemples d’hébergeurs : 1&1, Hetzner

Décrire les avantages de Debian en sécurité

- Rejeter le principe de la « sécurité par l'obscurité »
- Publier rapidement des correctifs de sécurité
- Mettre à jour pour toutes les versions stables pendant au moins 5 ans
- Assurer les mises à jour automatiques sans intervention
  - Paquet unattended-upgrades
- Respecter une politique de sécurité stricte
  - Installation minimale par défaut pour réduire la surface d'attaque
  - Séparation des privilèges et cloisonnement des processus
  - Audits de code réguliers et respect des bonnes pratiques

debian

Bilan du cours 2

« Avec l'open source, vous avez le droit de contrôler votre propre destin » - Linus Torvalds

Identifier les évolutions de la virtualisation

- Passage de la virtualisation à la para-virtualisation
- Distinction entre les types d’hyperviseurs 1 et 2
- Émergence des conteneurs
  - Système → Incus
  - Applications ou micro-services → Docker/podman
- Tendance vers plus de légèreté et d'efficacité
- Adoption croissante dans les environnements de production

Décrire les principes du logiciel libre

- Accès au code source comme base du logiciel libre
- Importance des licences (BSD, GNU) et du concept de copyleft
- Développement collaboratif à grande échelle (ex: noyau Linux)
- Rôle des fondations dans la structuration des écosystèmes
- Impact sur l'innovation et le contrôle des utilisateurs

Identifier les évolutions des pratiques DevOps

- Fusion des métiers du développement et de l'administration système
- Adoption des principes d'intégration, livraison et déploiement continus (CI/CD)
- Automatisation croissante des processus de développement et d'infrastructure
- Importance des distributions Linux stables et flexibles → Debian
- Focus sur la sécurité et les mises à jour automatiques dans les environnements cloud