Introduction aux systèmes GNU/Linux

S23E01 inetdoc.net

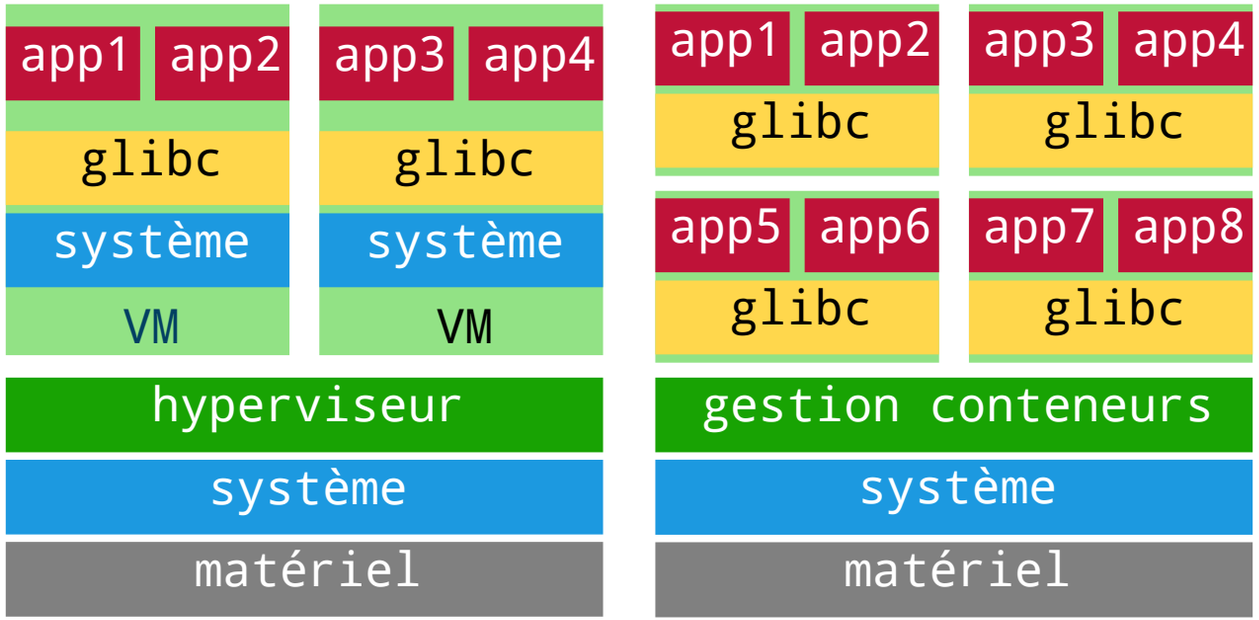
UPSSITECH
ÉCOLE D'INGÉNIEURS

Philippe Latu / Université Toulouse 3

Progression en modules – L3

Administrer un système

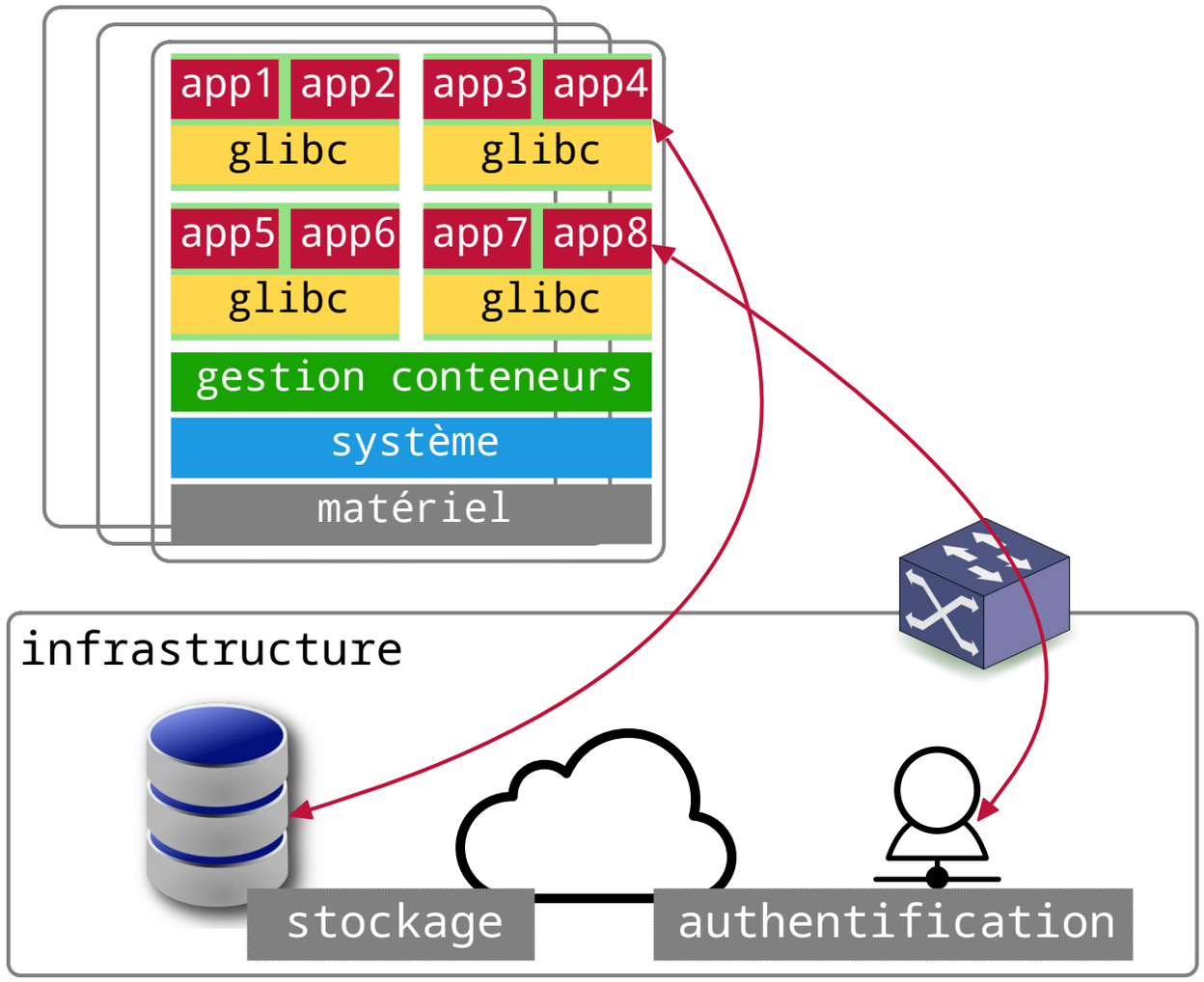
- Identifier les composants d'une distribution Linux : Debian
- Gérer des paquets
- Gérer des conteneurs système
- Gérer des machines virtuelles
- Créer et personnaliser des comptes utilisateurs locaux
- Identifier les ressources systèmes
- Gérer les processus



Progression en modules – M1

Administrer des systèmes en réseau

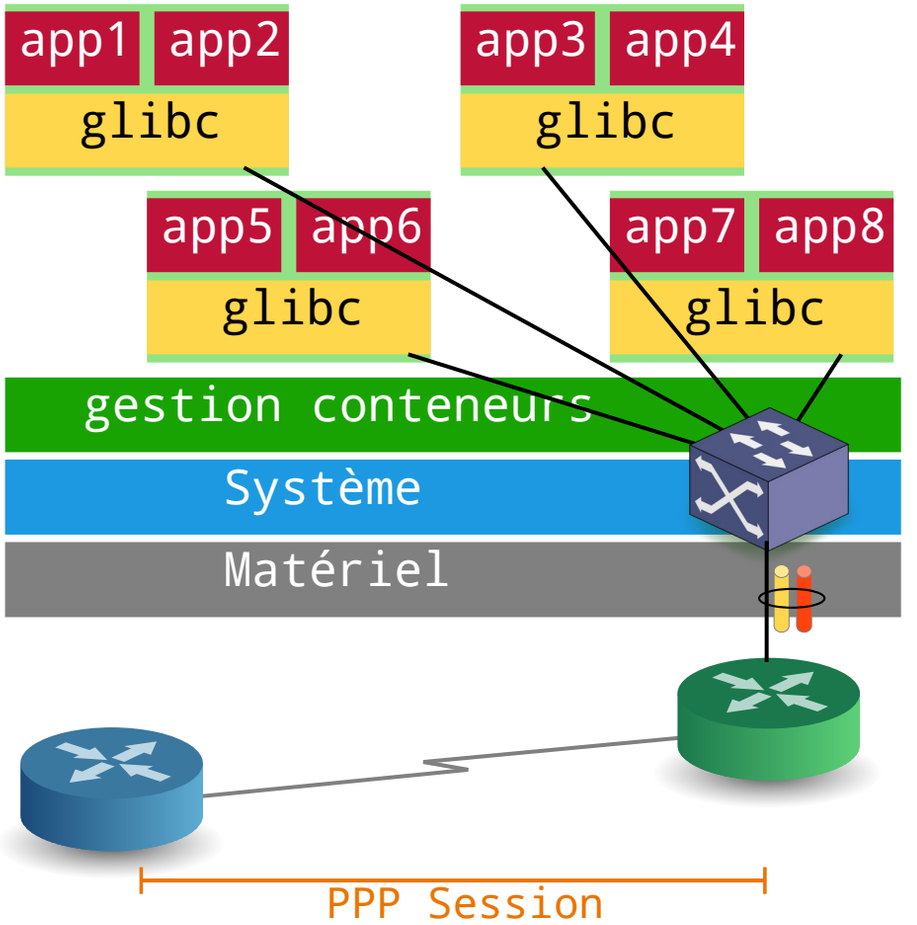
- Définir les types de stockage
- Choisir et configurer une solution de stockage réseau iSCSI ou NFSv4
- Installer et configurer un annuaire LDAP avec automontage NFSv4



Progression en modules – M1

Interconnecter des réseaux hétérogènes

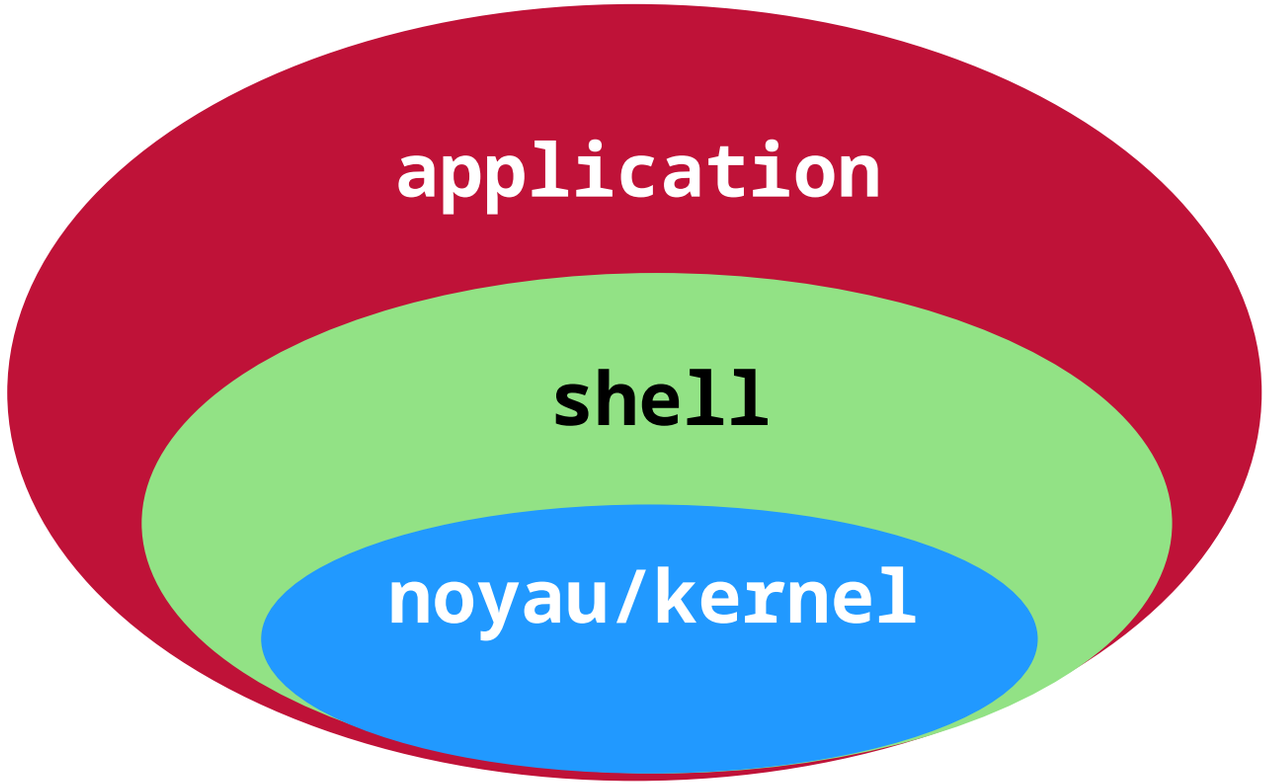
- Configurer une liaison WAN PPPoE
- Utiliser le routage inter-VLAN
- Configurer un réseau de conteneurs
- Configurer le protocole de routage dynamique OSPF



Introduction aux systèmes GNU/Linux

Séance 1

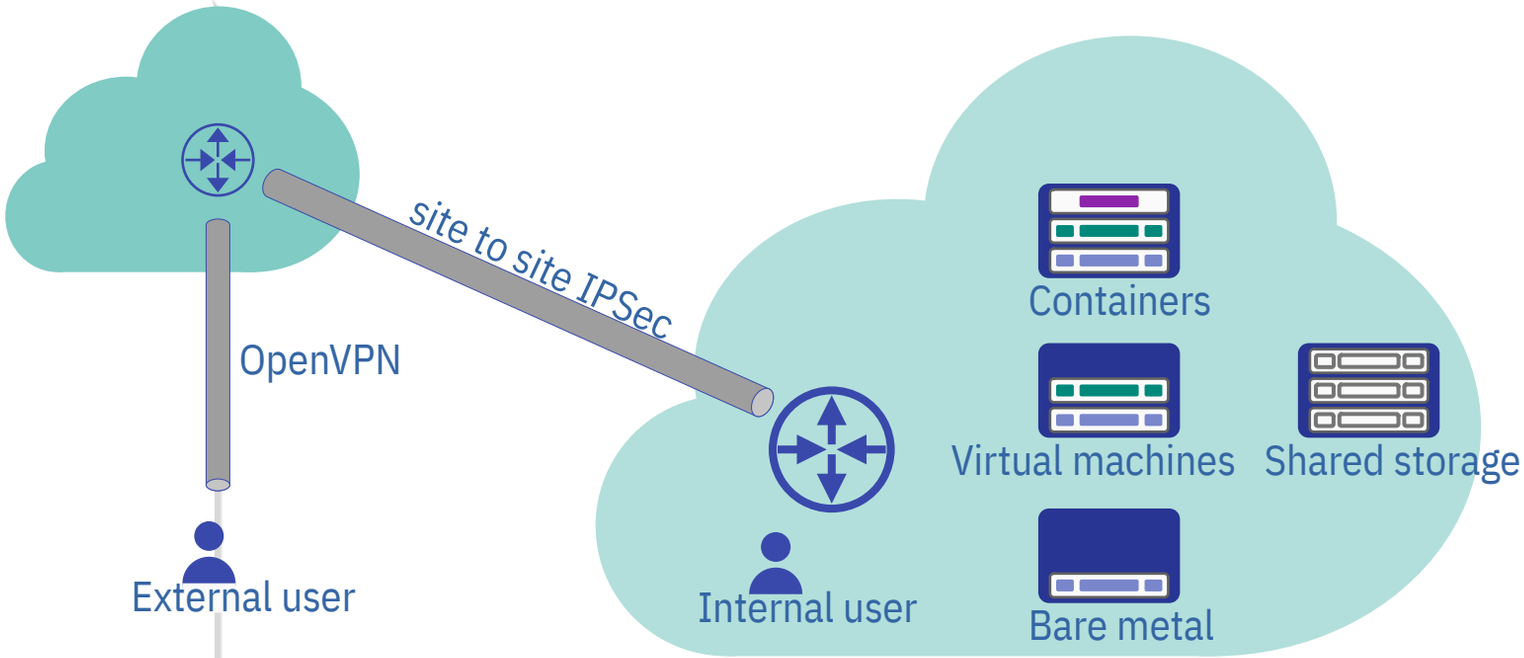
- Définir les 5 fonctions de base des systèmes Unix
- Définir les 3 couches de la modélisation d'un système d'exploitation
- Définir les types de virtualisation et de conteneurs
- Définir une distribution GNU/Linux
- Identifier une licence de logiciel libre



Introduction aux systèmes GNU/Linux

Séance 2

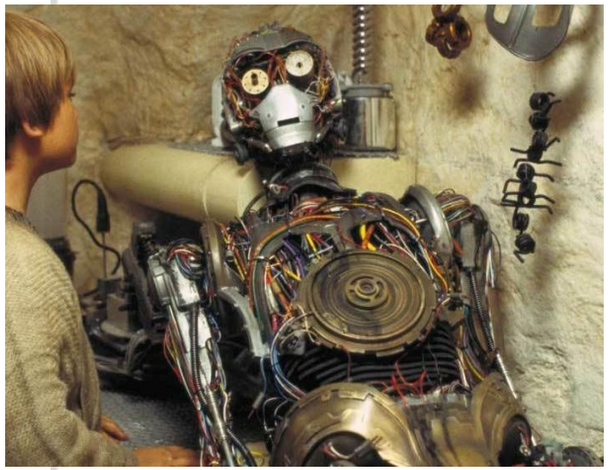
- Identifier les types et les modes d'accès aux "Clouds"
- Utiliser un VPN
- Lancer un système virtualisé
- Configurer un gestionnaire de conteneurs



Opération C-3PO

- Installer Open vSwitch (OvS)
- Installer les premiers conteneurs avec LXD

https://gist.github.com/platu/



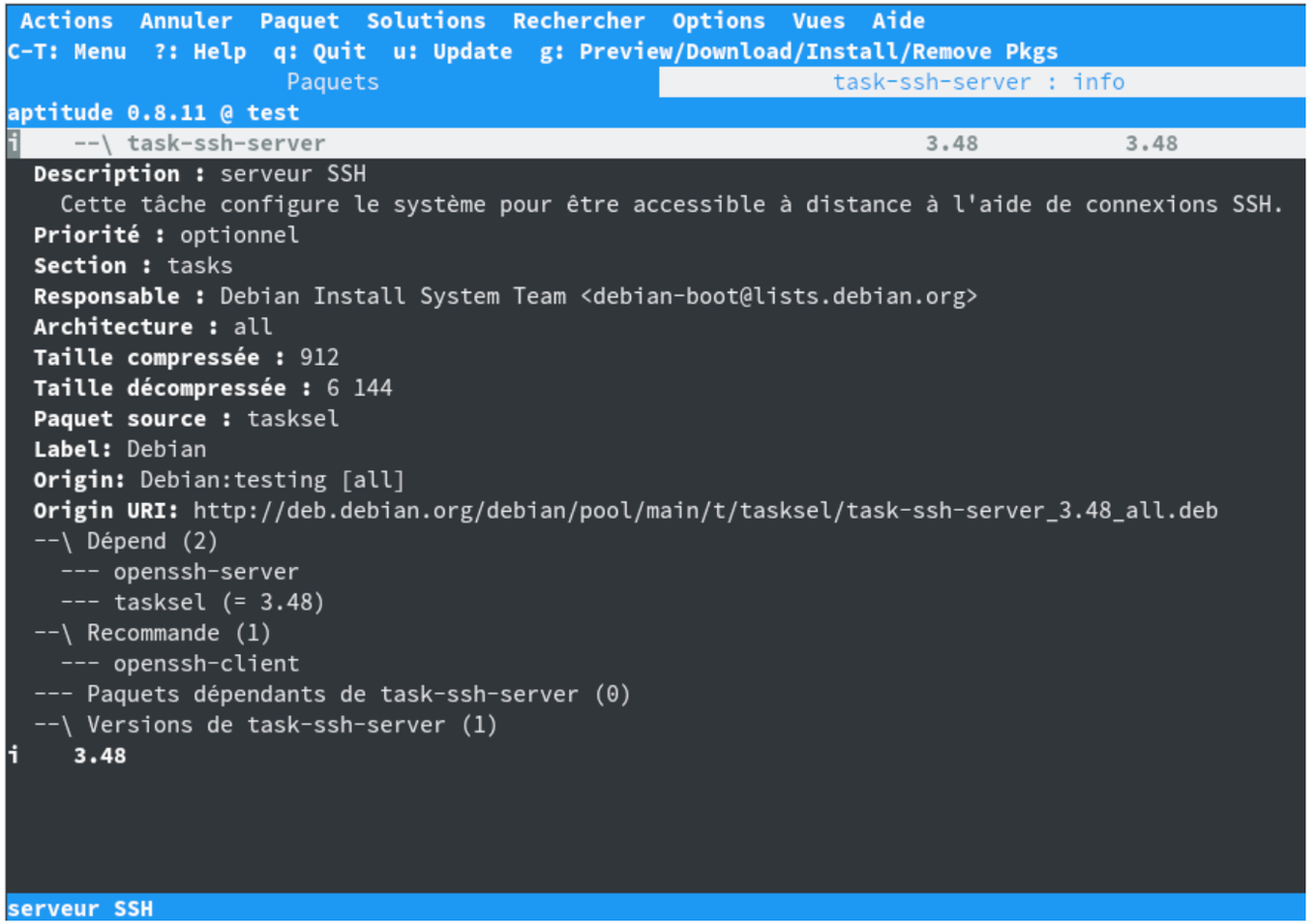
Introduction aux systèmes GNU/Linux

Séance 3

- Identifier les environnements graphiques et les chaînes de développement associées
- Gérer les mises à jour des paquets de la distribution
- Identifier les dépendances entre applications et bibliothèques
- Utiliser les méta données

Scénario

- Installer un conteneur LXD avec les paquets d'un service Web



Introduction aux systèmes GNU/Linux

Séance 4

- Utiliser les ressources du shell Bash
- Coder un script Bash
- Identifier et gérer les processus
- Gérer les permissions sur les objets de l'arborescence système

Scénario

- Installer un blog statique

```
etu@vm0:~$ ls -lAh
total 20K
-rw------- 1 etu etu  131 nov.  26 21:37 .bash_history
-rw-r--r-- 1 etu etu  220 nov.  26 17:01 .bash_logout
-rw-r--r-- 1 etu etu 3,5K nov.  26 17:01 .bashrc
drwx------ 3 etu etu 4,0K nov.  26 17:35 .gnupg
-rw-r--r-- 1 etu etu  807 nov.  26 17:01 .profile
```

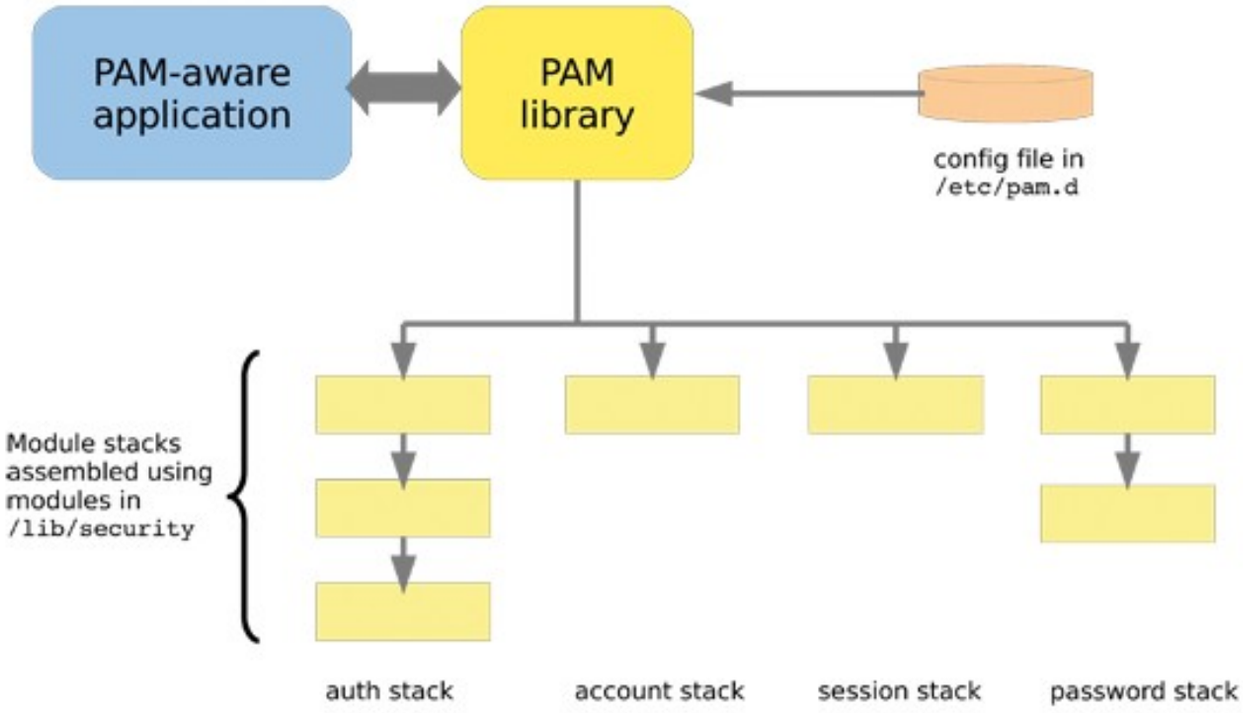
Introduction aux systèmes GNU/Linux

Séance 5

- Gérer les comptes utilisateurs locaux
- Personnaliser les comptes locaux
- Gérer les groupes
- Changer d'identité avec sudo ou su
- Exploiter les messages systèmes
- Planifier les tâches avec cron

Scénario

- Gérer les permissions de groupes pour le développement de site Web



Introduction aux systèmes GNU/Linux

Séance 6

- Présenter les étapes de l'initialisation d'un système
- Identifier le rôle du gestionnaire d'amorce
- Distinguer les espaces mémoire noyau et utilisateur
- Reconnaître les services lancés au démarrage → systemd

Scénario

- Appliquer des contraintes sur l'accès aux ressources pour un conteneur

BIOS	Basic Input Ouput System → recherche MBR
MBR	Master Boot Record → recherche GRUB
GRUB	Grand Unified Bootloader → recherche noyau
kernel	Noyau Linux → identification des ressources
systemd	→ exécution /sbin/init → lancement des services

Concepts Unix et GNU/Linux

Une définition de la notion de *culture système*

« un ensemble lié de manières de penser, de sentir et d'agir plus ou moins formalisées qui, étant apprises et partagées par une pluralité de personnes, servent, d'une manière à la fois objective et symbolique, à constituer ces personnes en une collectivité particulière et distincte. »

Guy Rocher – https://fr.wikipedia.org/wiki/Culture

Développer une culture système, c'est :

- Identifier les étapes d'une histoire continue sur 5 décennies
- Comparer la genèse des systèmes Unix avec celle des services Internet
- Identifier les processus d'assurance qualité des distributions
- Décrire les nouveaux processus métier → DevOps !

Comment évaluer ?

Maîtriser les outils du logiciel libre, c'est démontrer un niveau de culture système

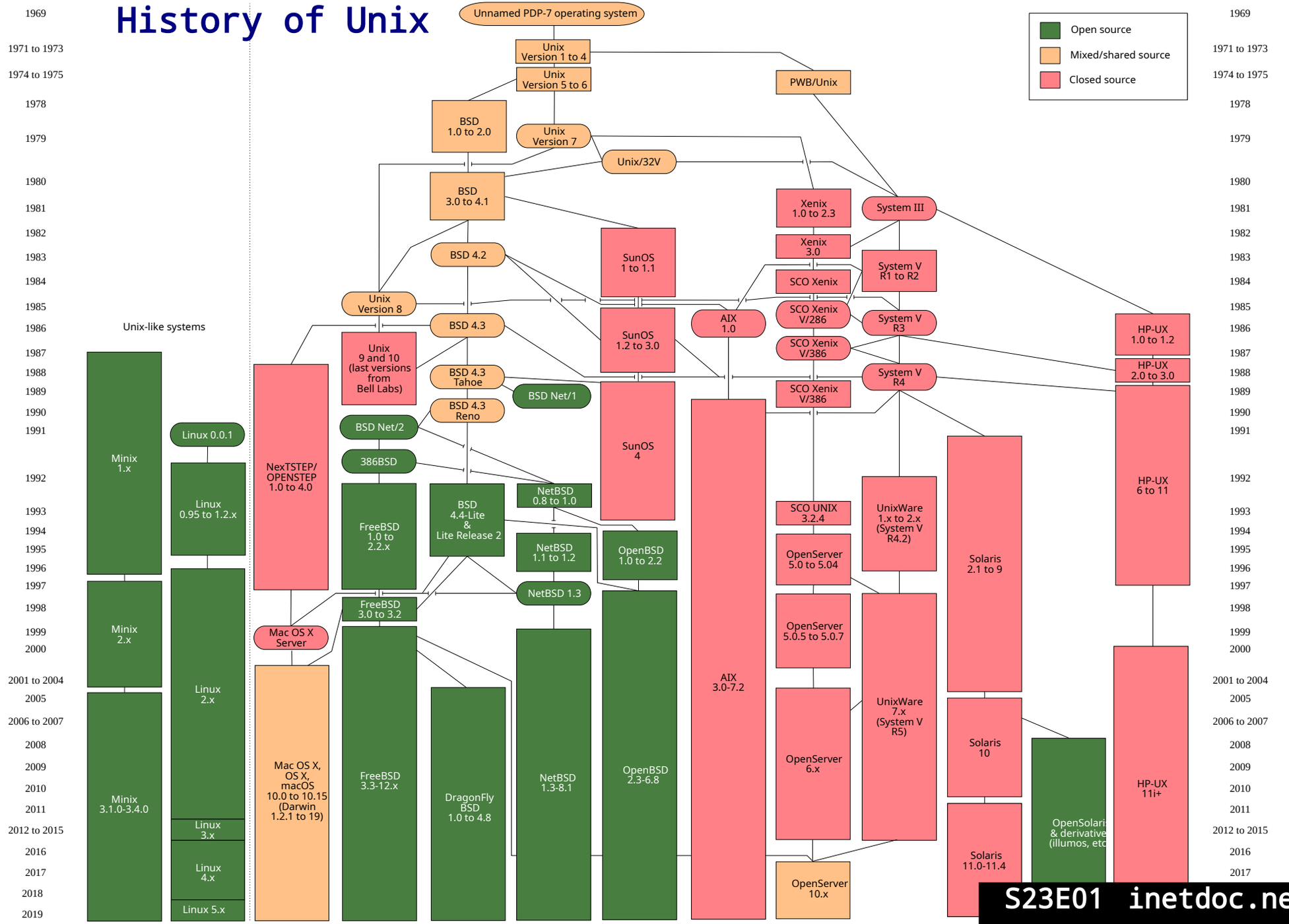
5 fonctions de base des systèmes Unix

- **Multi-tâches**
  - Temps processeur partagé entre plusieurs programmes
- **Multi-utilisateurs**
  - Ressources système partagées entre plusieurs utilisateurs
- **Portabilité**
  - Outils système partagés entre cibles matérielles différentes
- **Bibliothèques de développement standard**
  - Optimiser la qualité des développements en partageant le code source
- **Applications communes**

  Services système, services Internet, etc.

Unix

Family tree



Concepts Unix et GNU/Linux

1969 – Unics – AT&T – Système V

- Unix est un système «accidentel»
  - AT&T Bell labs – Ken Thompson - Dennis Ritchie
  - 1973 réécriture en Langage C
  - Diffusion sous licence AT&T incluant la totalité du code source
  - 1975 publication RFC681 NETWORK UNIX
- **Apparition des variantes propriétaires**
  - Coût de licence prohibitif
  - Versions constructeurs incompatibles entre elles
  - Segmentation en parts de marché captives

Concepts Unix et GNU/Linux

1977 – Berkeley University – BSD

- **Branche Unix lancée à partir d'une licence AT&T**
  - Nombreuses améliorations
    - Gestion mémoire
    - Sous-système réseau TCP/IP
  - Diffusion entre universités
  - Début de l'Internet universitaire
- **Procès AT&T**
  - BSD est devenu un système complet autonome
  - Éclatement de la branche BSD
  - FreeBSD, NetBSD et OpenBSD

Concepts Unix et GNU/Linux

1984 – GNU – Not Unix

- **Projet lancé par Richard Stallman**
- **2 objectifs**
  - Promouvoir le développement des logiciels libres
    - Protection des travaux des développeurs à l'aide de licences
  - Fédérer les développements libres
    - Applications GNU
- **Unix comme modèle**
  - Fonctions de base déjà éprouvées
  - 1990 – chaîne de développement stable
    - GNU Compiler Collection
  - 1991 – arrivée du noyau Linux

Concepts Unix et GNU/Linux

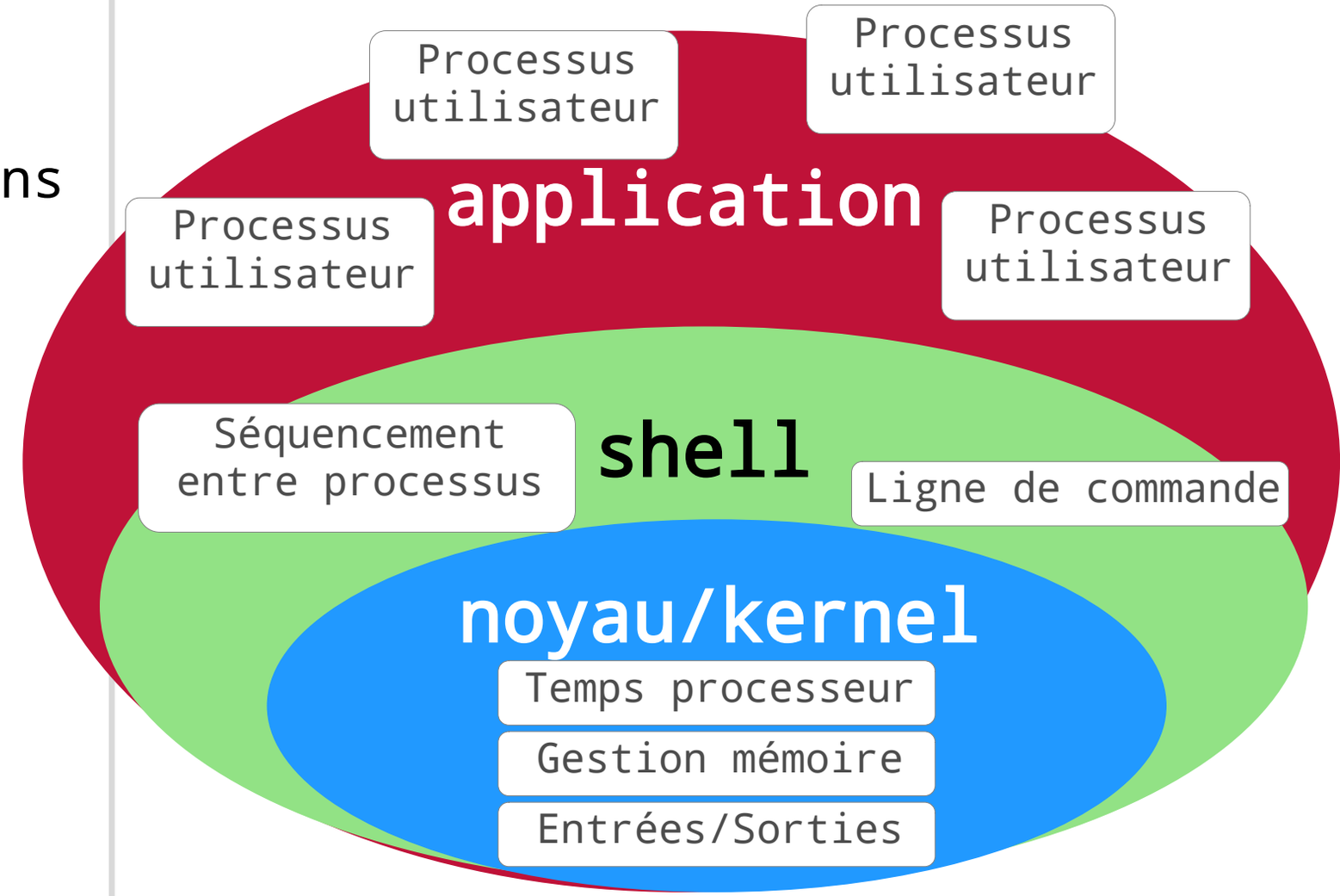
1991 – Débuts du noyau Linux

- Développement initié par Linus Torvalds
  «divergences de vues» avec A.S. Tanenbaum
  - http://www.oreilly.com/catalog/opensources/book/appa.html
- **Fonctions de base Unix + quelques spécificités**
  - Multi-tâches
  - Multi-utilisateurs
  - Gestionnaire mémoire
    - Mémoire virtuelle = utilisation répétitive et étendue
  - Mode protégé (processeurs Intel)
    - Contrôle d'accès ← Débuts sur l'isolation mémoire entre processus

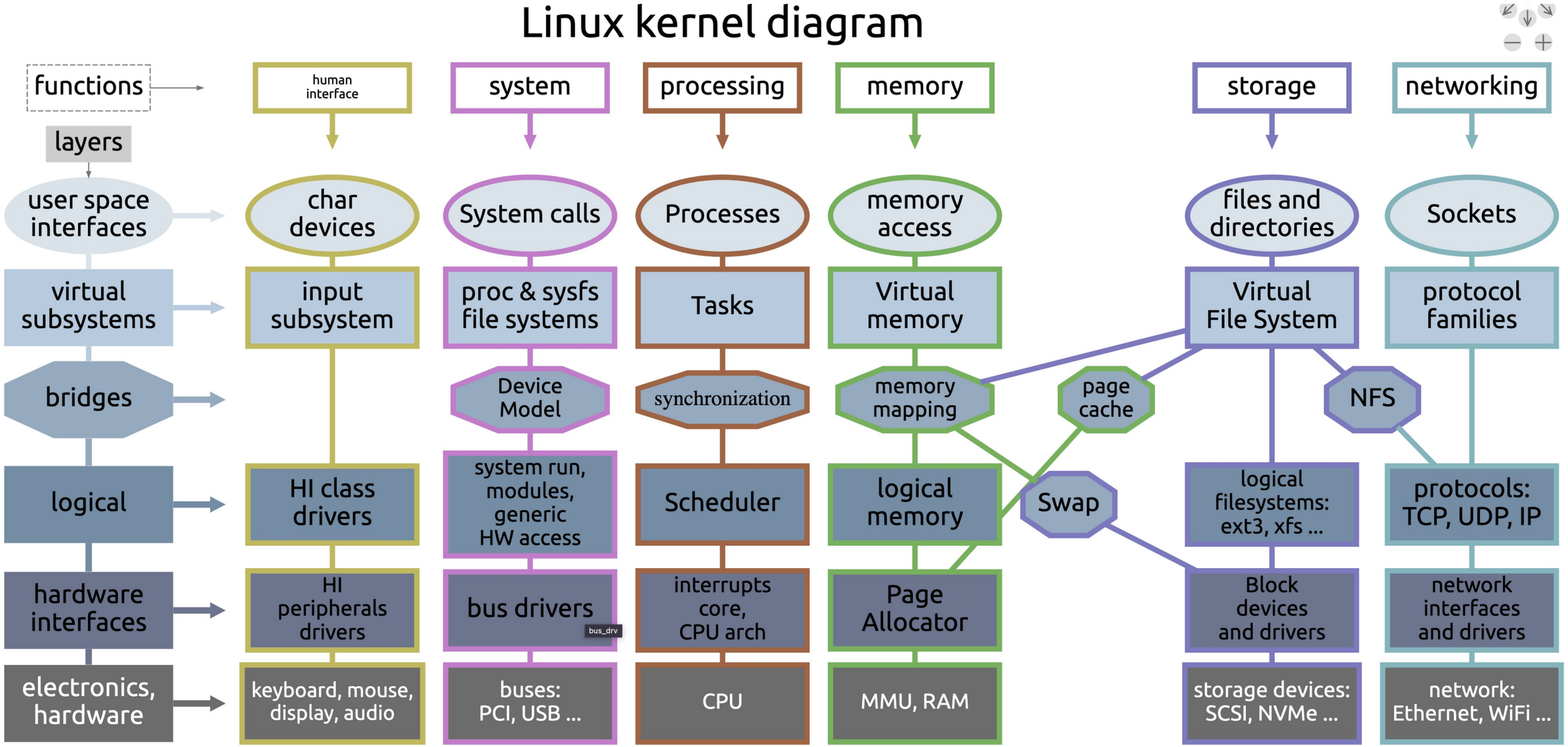
Système GNU/Linux

Modélisation d'un système GNU/Linux

- Noyau/kernel → 3 fonctions
  - Temps processeur
  - Mémoire
  - Entrées/Sorties
- Shell → 2 fonctions
  - Séquencement entre processus
  - Ligne de commande
- Application
  - Tous les processus utilisateur



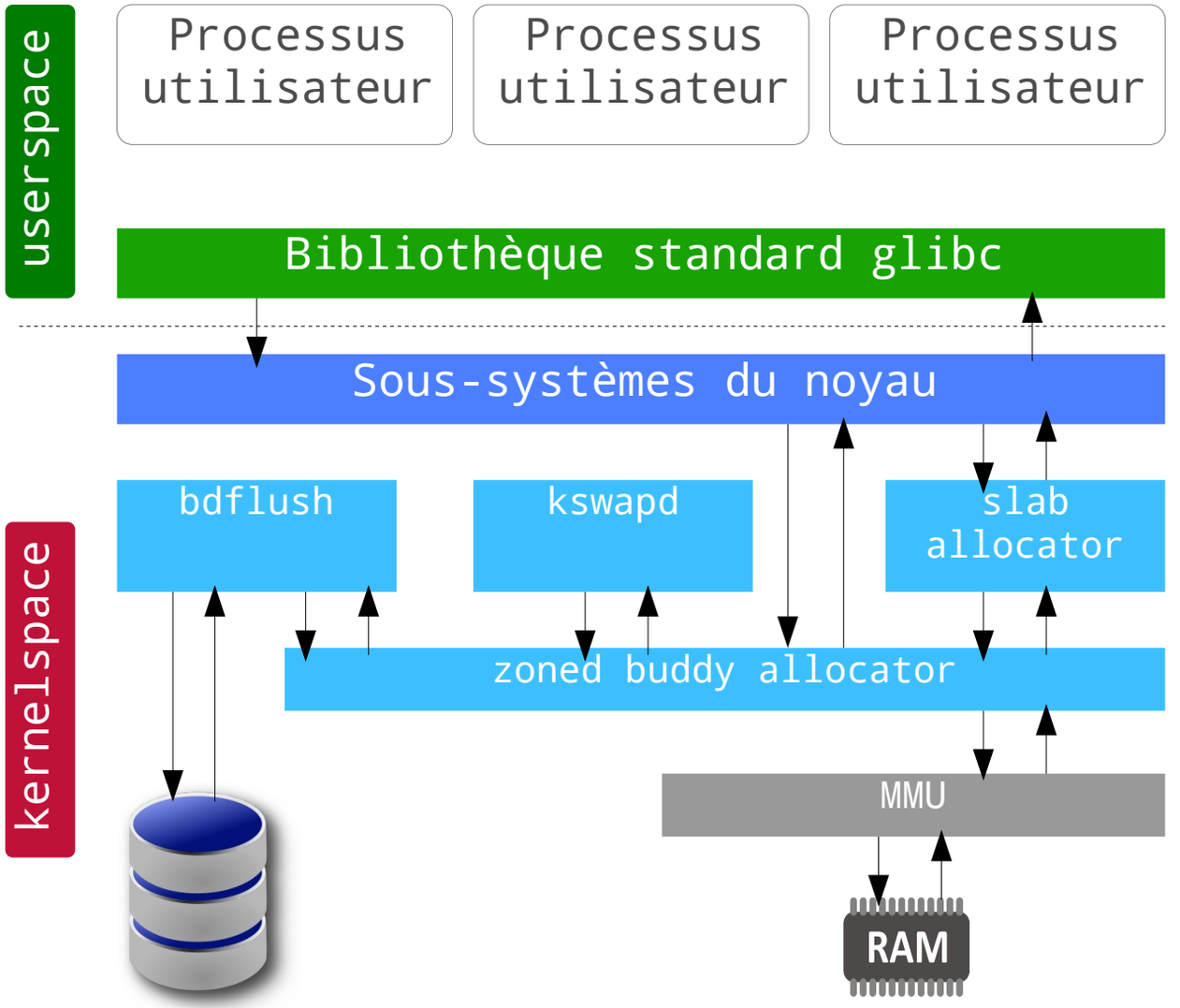
Linux kernel diagram



https://makelinux.github.io/kernel/diagram/

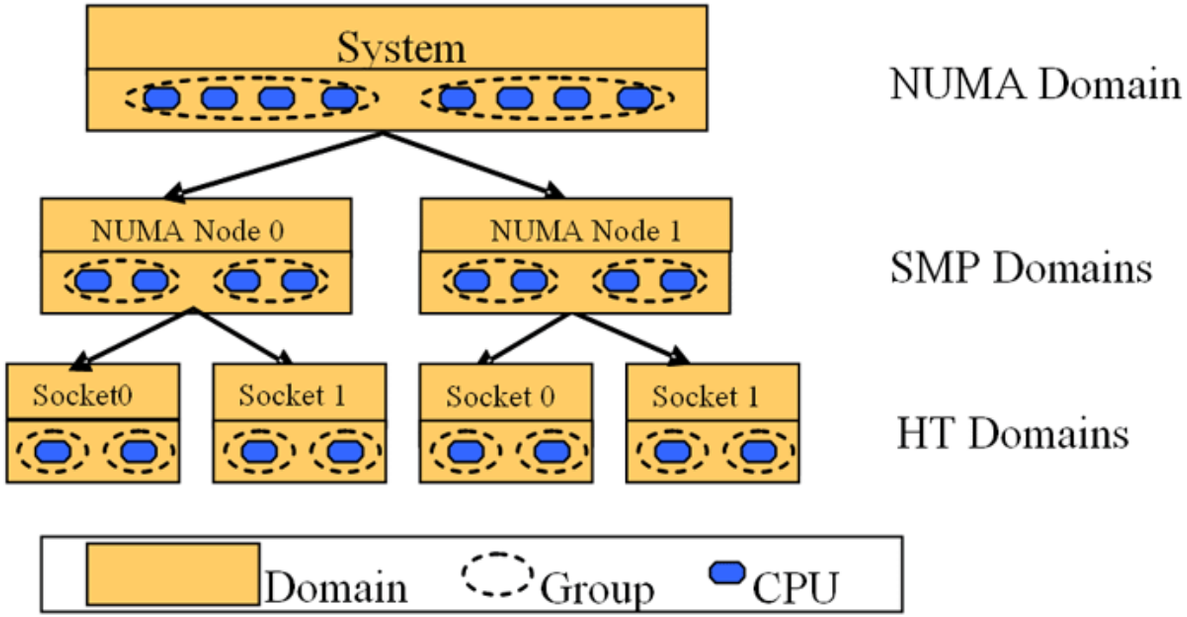
Mémoire virtuelle

- userspace
  - Programmes utilisateurs
  - Bibliothèque standard glibc
- kernelspace
  - Memory Management Unit (MMU)
  - Zoned buddy allocator
    - Allocation pages mémoires
  - Slab allocator
    - Cache dans les pages mémoire
  - Kernel threads
    - Réutilisation de la mémoire



Ordonnanceur/Scheduler

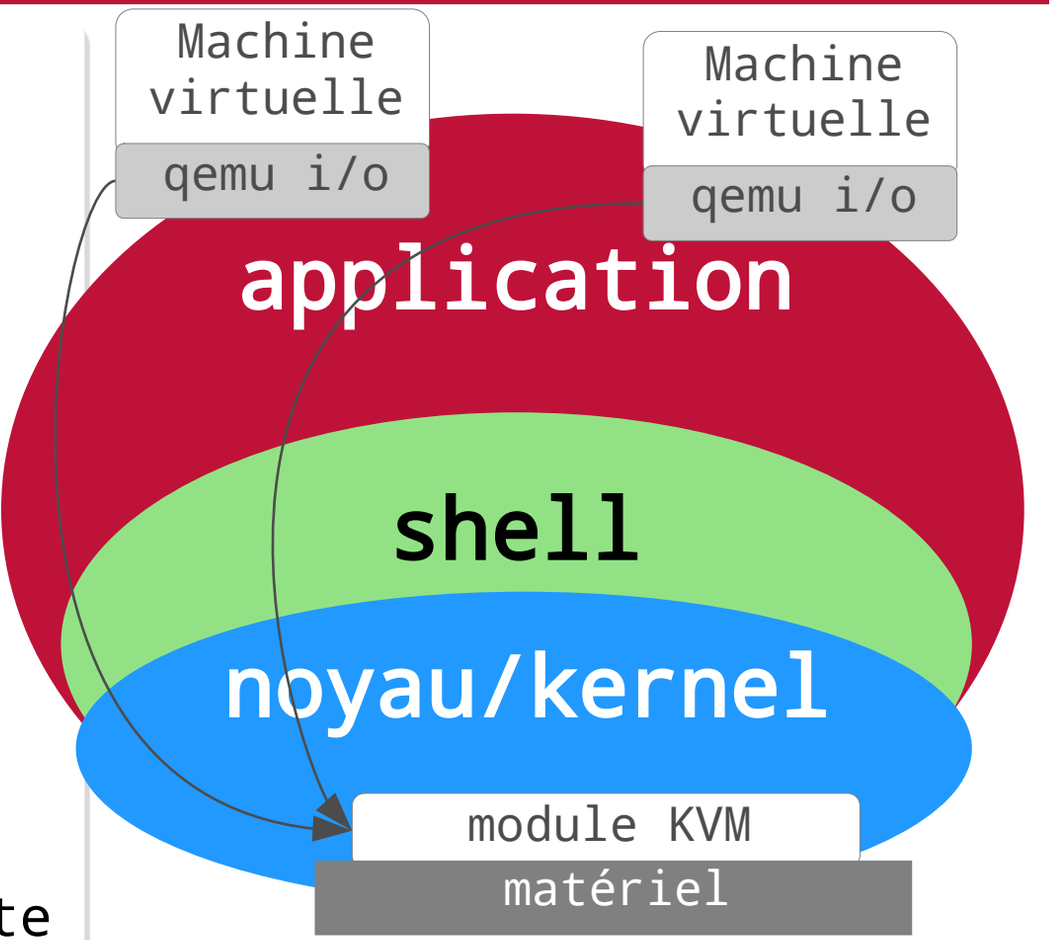
- 3 domaines ou types de tâches
  - Domaine temps réel
    - Contraintes de temps d'exécution
    - Fréquence d'exécution garantie
  - Domaine entrées/sorties
    - Attente de disponibilité des périphériques
  - Domaine CPU
    - Temps consacré aux calculs
- Tranche de temps CPU / time slice
  - Durée d'exécution d'un processus sur un cœur
- Préemption
  - Interruption d'un processus par un second de priorité plus élevée



Système GNU/Linux

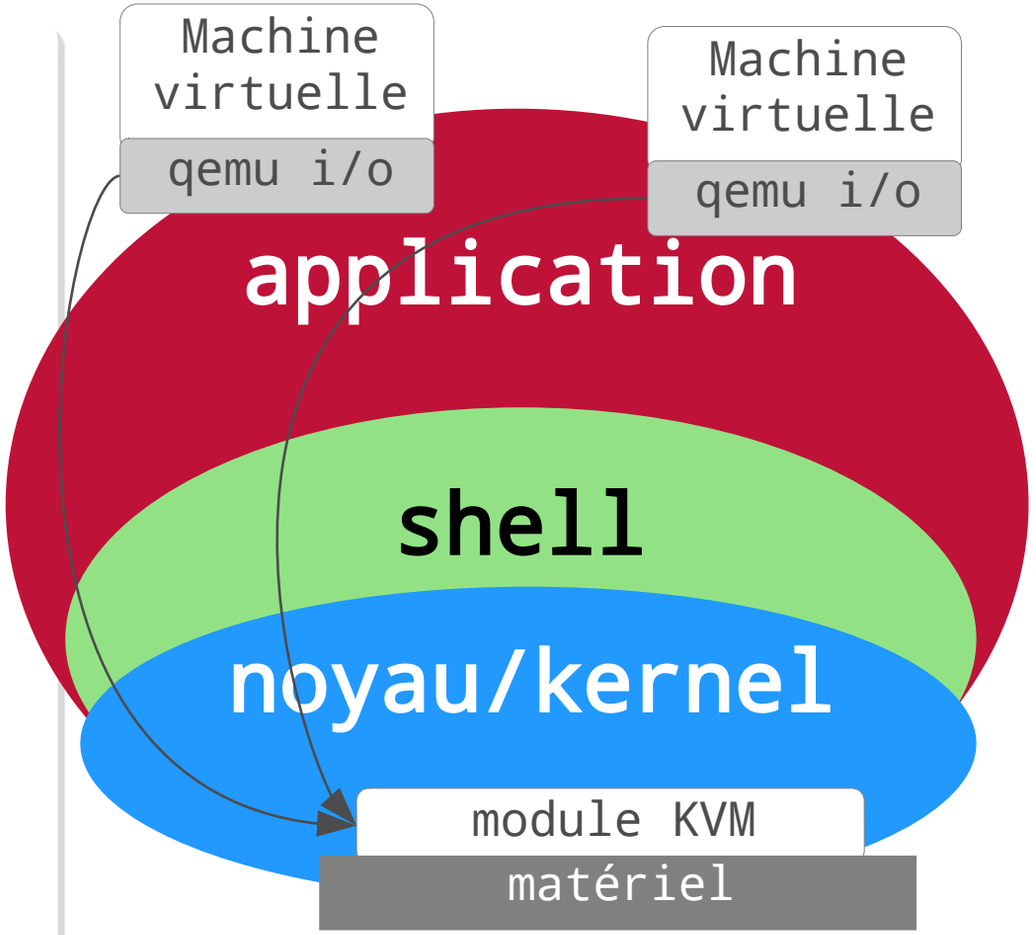
Virtualisation ou para-Virtualisation ?

- **Sans hyperviseur → virtualisation**
  - Émulation du matériel dans l'espace utilisateur
  - Performances limitées
  - Adapté aux cibles matérielles obsolètes
- **Avec hyperviseur → para-virtualisation**
  - Communication directe entre machine virtuelle et matériel du système hôte
  - Performances identiques entre système hôte et système virtualisé
  - Périphériques dédiés dans le noyau de la machine virtuelle



Virtualisation ou para-Virtualisation ?

- **Outil de l'espace utilisateur → qemu**
  - qemu seul → virtualisation
  - 2 fonctions principales
    - Manipulations sur les fichiers image de machine virtuelle
    - Description du matériel des machines virtuelles
- **Outil de l'espace noyau → kvm**
  - qemu + kvm → para-virtualisation
  - Utilisation des périphériques de la bibliothèque virtio



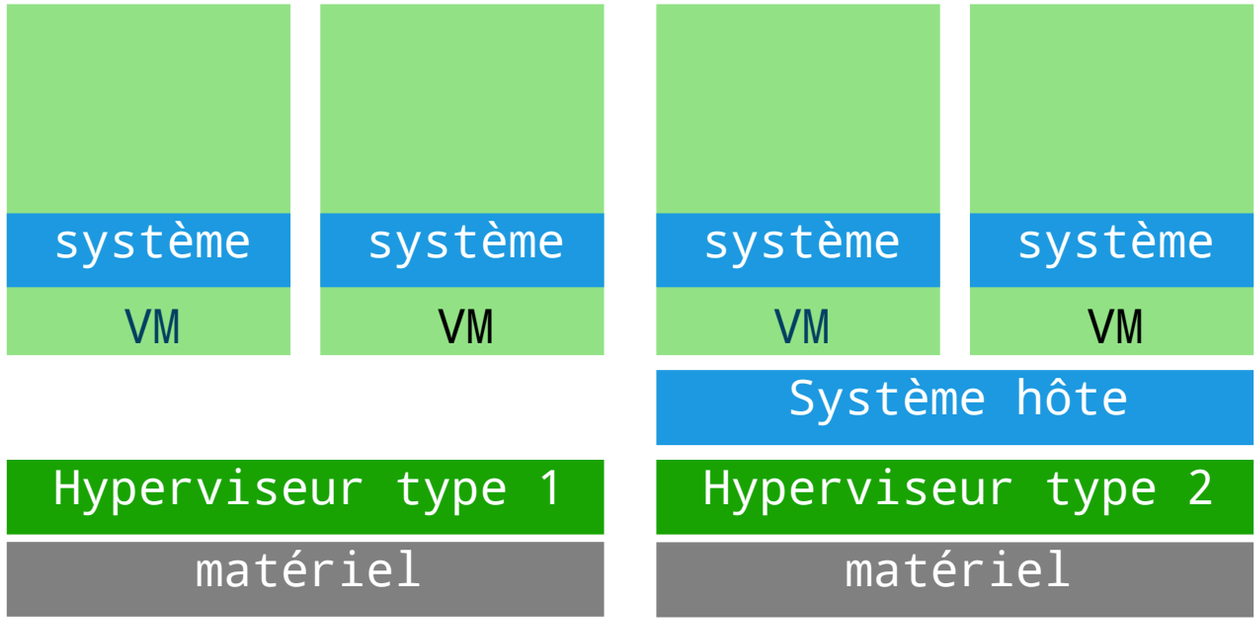
Système GNU/Linux

Hyperviseurs de type 1 ou 2

- **Type 1**
  - Accès direct au matériel depuis les machines virtuelles
  - VMware ESX
- **Type 2**
  - Services assurés par le système hôte → Open vSwitch
  - GNU/Linux KVM

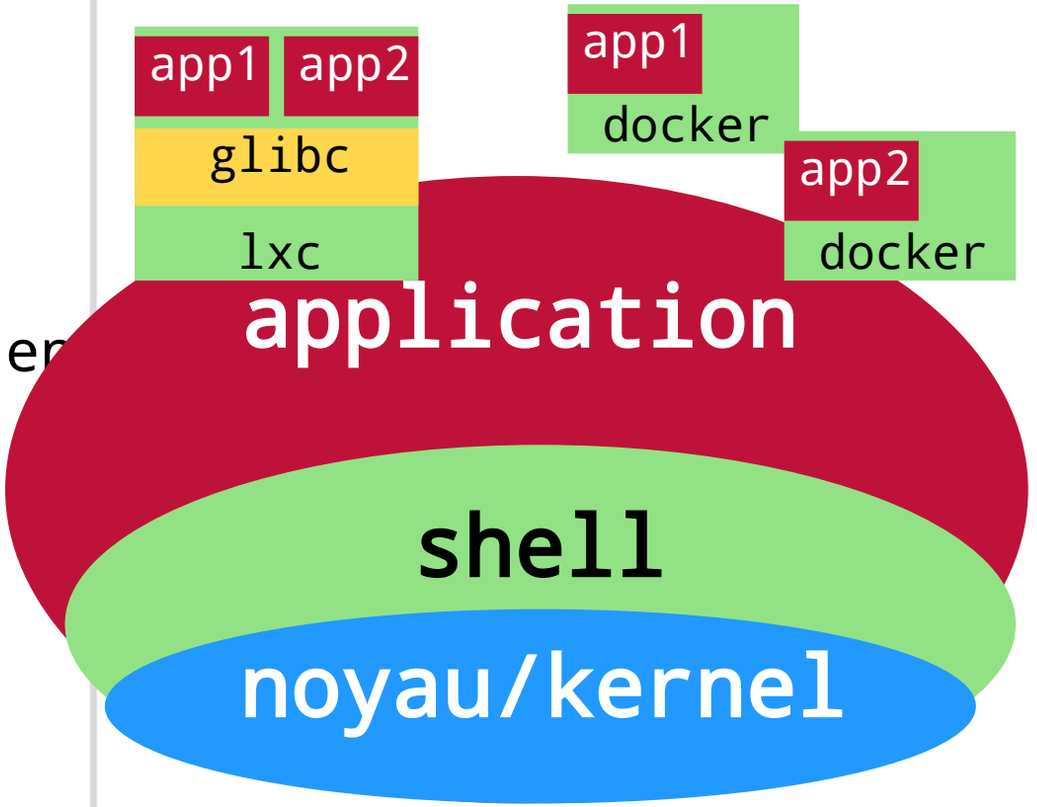
Comment différencier les 2 types ?

Accès à la console → type 2



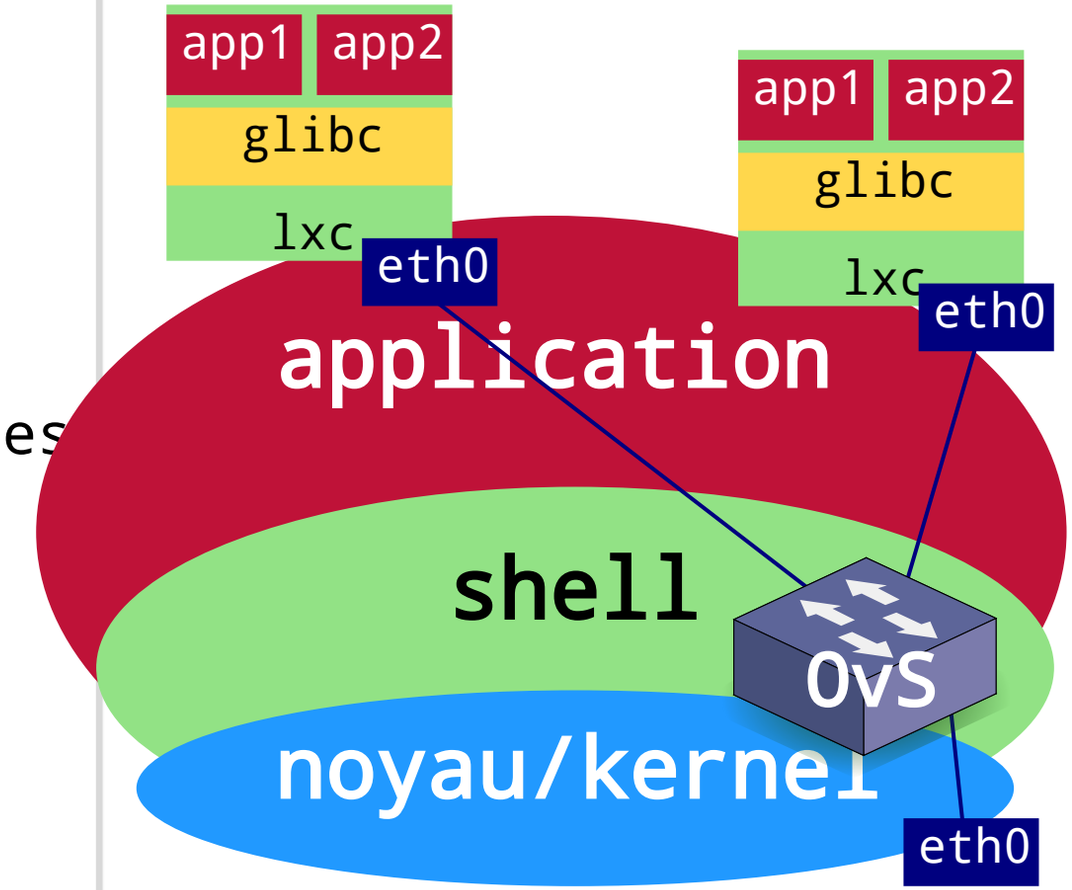
Conteneurs ou micro services

- **Conteneurs système → LXD ou Incus**
  - Système 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

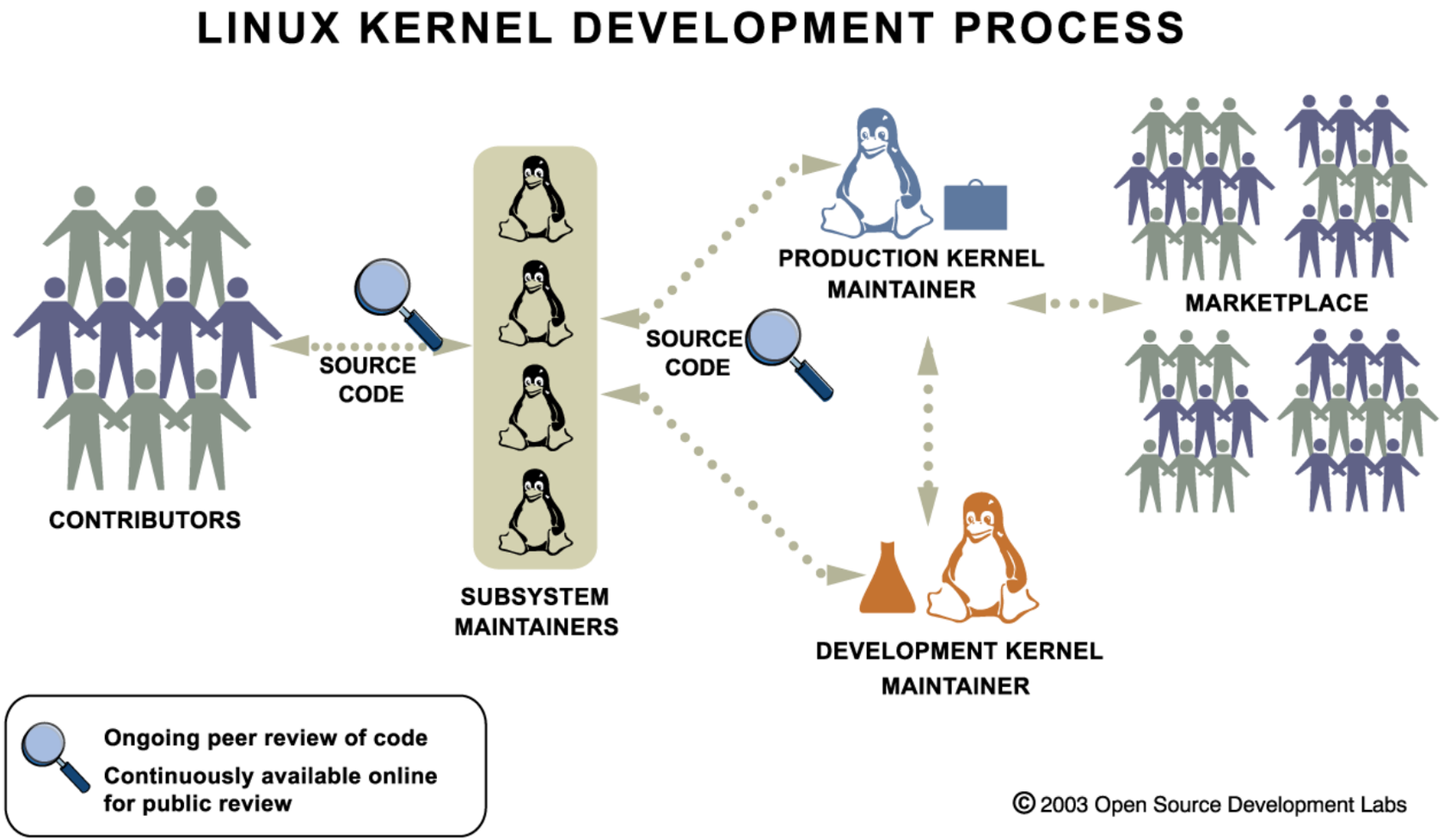


Opération C-3PO : conteneurs & réseaux sans orchestration

- **Objectifs**
  - Aucun compromis entre commutation virtuelle et commutation physique
  - Pas de recours à la traduction d'adresses
- **Conteneurs système**
  - Configuration simplifiée via snap
  - Raccordement réseau automatique
- **Open vSwitch (OvS)**
  - Table CAM (*Content Addressable Memory*)
  - Base de VLANs



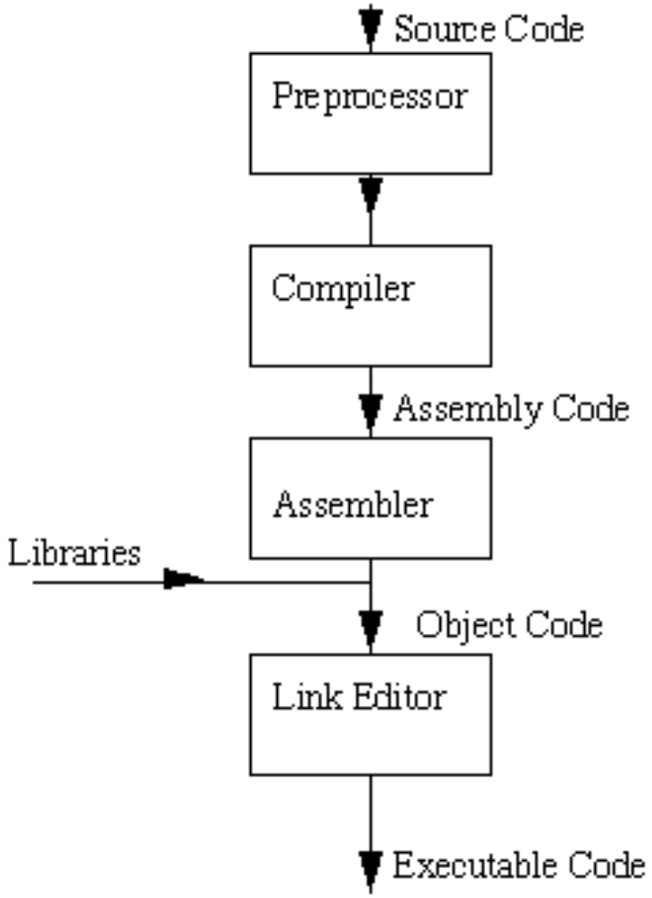
Système GNU/Linux



Logiciel Libre & Licences

Code source → 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
- **Compilation**
  - Transformation du code source en code exécutable
  - Transformation inverse «impossible»
  - Code exécutable = binaire
- **Logiciel propriétaire**
  - Droit d'utilisation d'un code exécutable
- **Logiciel libre**
  - Accès au code source
  - Droit d'utilisation, d'échange,
    de modification et de redistribution



Logiciel Libre & Licences

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

Logiciel Libre & Licences

Application du Copyleft

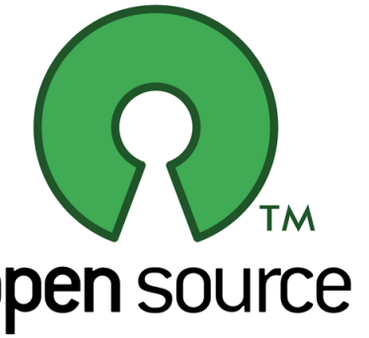
- Appliquer un copyright sur le logiciel
- Fixer les conditions de distribution

  «*Donner à tout utilisateur le droit d'utiliser, de modifier et de redistribuer le programme sans changer les conditions de distribution*»
- Le code source et les libertés associées sont inséparables
  - http://www.gnu.org/copyleft/copyleft.fr.html

Projets OpenSource

Applications OpenSource & systèmes GNU/Linux

- Une histoire commune
  - Évolutions, processus et méthodes
- Exemples de services Internet
  - Bind : noms de domaines
  - Postfix : courrier électronique
  - Apache/NGINX : services web



- Terminologie OpenSource
  - Modifications distribuées dans les mêmes conditions que l'original
  - Restrictions possibles sur la redistribution des correctifs
  - Licence sans restrictions sur d'autres logiciels associés

Projets OpenSource

Modèle 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

Projets OpenSource

Quelles sont les évolutions métier en cours ?

- **Documentation**
  - Sources d'alimentation pour les intelligences artificielles
  - https://stackoverflow.com

- **DevOps Mantra**
  - Toujours plus d'automatisation
  - Toujours plus d'applications intégrées ou dépendantes
  - Tout est programmable
    «le déploiement d'infrastructure utilise les mêmes processus que le déploiement logiciel»
  - Fusion des métiers du développement et de l'administration système

- **Infrastructure as Code →** Page Wikipedia

Projets OpenSource

Quelles sont les évolutions métier en cours ?

- **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.

- Exemples
  - Transformation de github ou gitlab en outils de développement intégrés

Distribution

Distributions GNU/Linux & BSD

- Canaux de diffusion du logiciel libre
- Distribution → association de 3 éléments
  - Noyau
  - Un ou plusieurs Shells
  - Un ensemble d'applications
- Composants distribués sous forme de paquets
  - Code binaire exécutable
  - Configuration type
- Gestion de paquets
  - Principal enjeu dans la vie d'une distribution

debian

Distribution

Des logiques complémentaires

- **Publier très régulièrement → DevOps**
  - Fournir les outils les plus récents
  - Planification → Release Cycle
- **Publier à partir des critères qualité → Long-Term Support (LTS)**
  - Fournir les outils des infrastructures critiques
  - Garantir la continuité de service
- **Un paquet d'application est un livrable**
  - Tendance → publication pilotée à partir des projets amonts
- **Responsable de paquet → un rôle historique**
  - Capitalisation des compétences d'exploitation
  - Qualités humaines dans la coordination
  - Démarche qualité lors des évolutions

Distribution

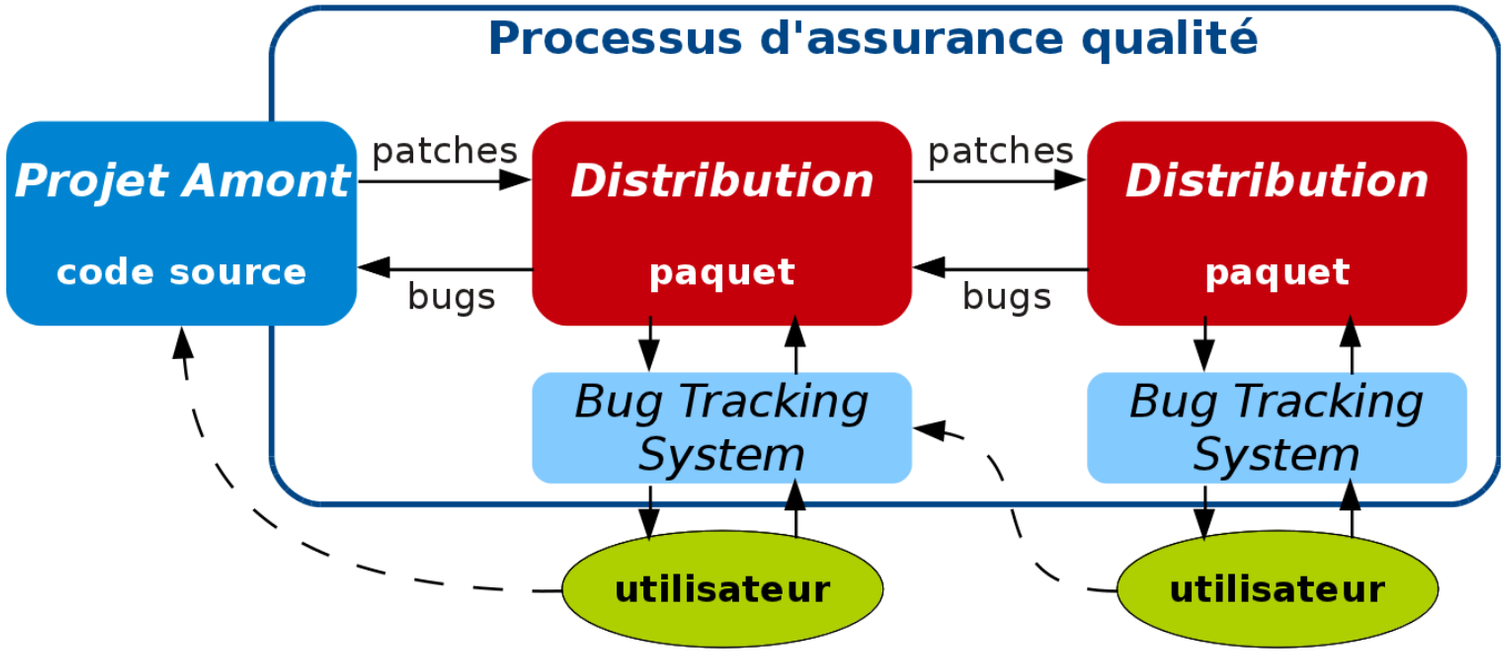
Critères de choix d'une distribution

- 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
- Adaptation + évolution continue
  - Continuité de service
  - Haute disponibilité

Distribution

Relations & filiations entre distributions

- Assurance qualité & coordination
  - Exemple : Debian → Ubuntu → LinuxMint
  - Modèle «pipeline»



Distribution

Debian GNU/Linux

- Évolution cohérente et continue depuis 1993
- Contrat social + 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

Distribution

Debian GNU/Linux

- Choix pédagogique
  - Mille et unes distributions spécialisées
  - Très peu de distributions généralistes
- Transparence des processus
  - Qualité : https://tracker.debian.org
  - Sécurité : https://www.debian.org/security
- Processus métiers difficiles à illustrer
  - Coût d'accès au support
  - Diffusion restreinte de l'information
  - Difficulté d'accès à l'information
  - Communautés peu ouvertes

debian

Bilan séance 1

Logiciels Libres & Projets OpenSource

- Rôle majeur dans le monde des technologies de l'information
  - Modèle de développement original et éprouvé
  - «La Cathédrale» versus «Le Bazar»
- Évolutions en écosystèmes

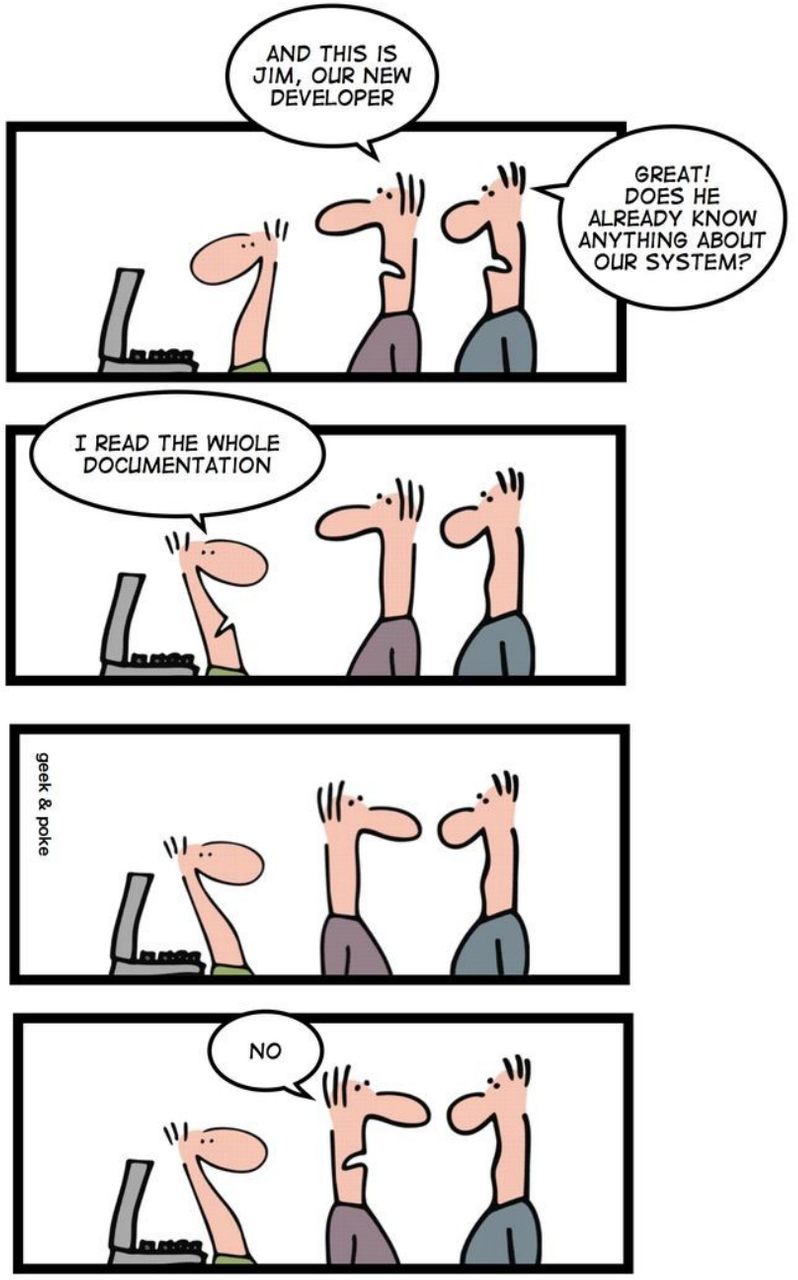
  Nouvelle répartition des rôles entre entreprises de services et fondations
- La valeur ajoutée est générée à partir du partage de biens communs

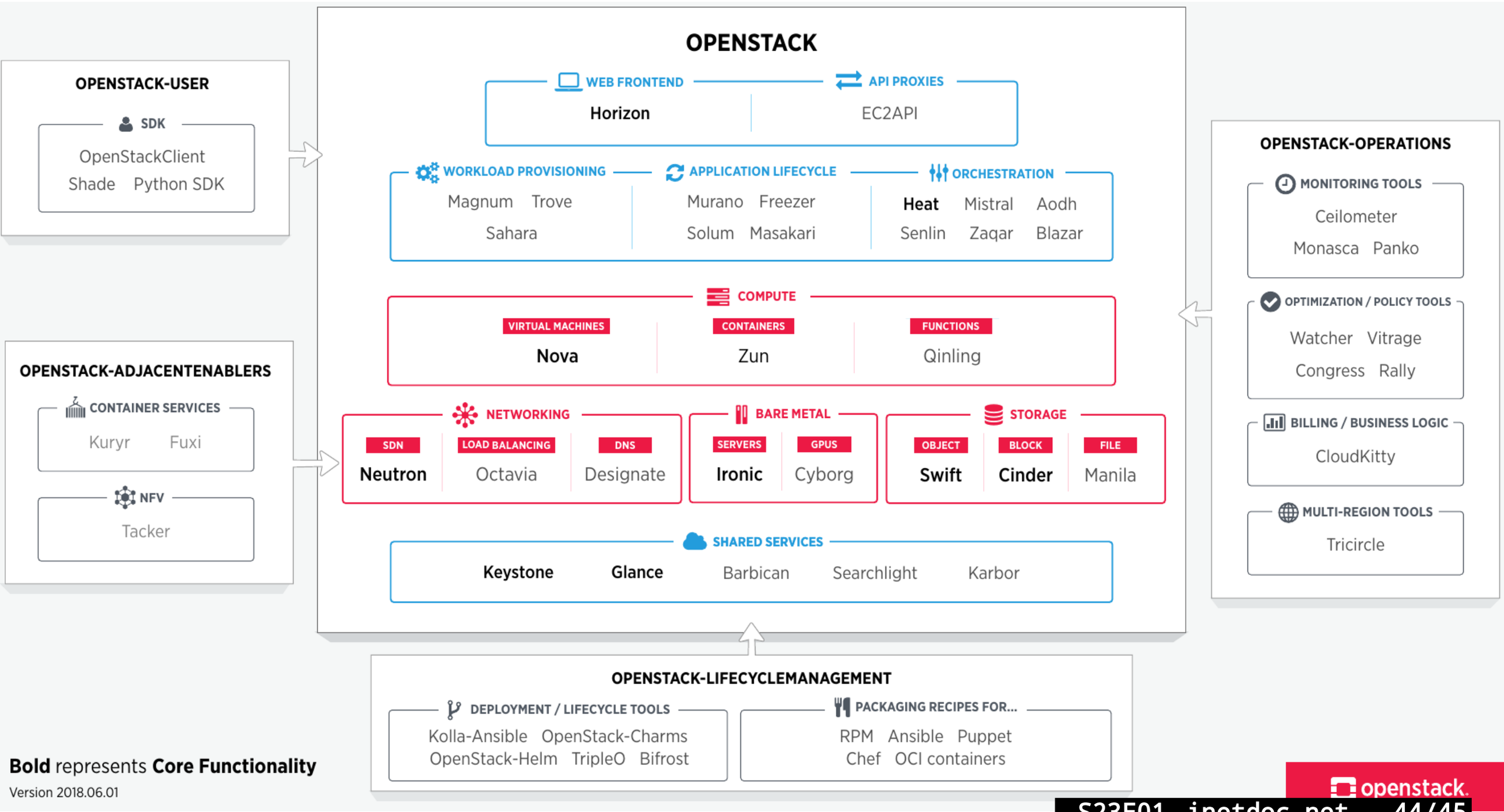
  «La culture système est devenue une condition importante pour l'accès à l'emploi»

Bilan séance 1

Étapes du développement des systèmes Unix

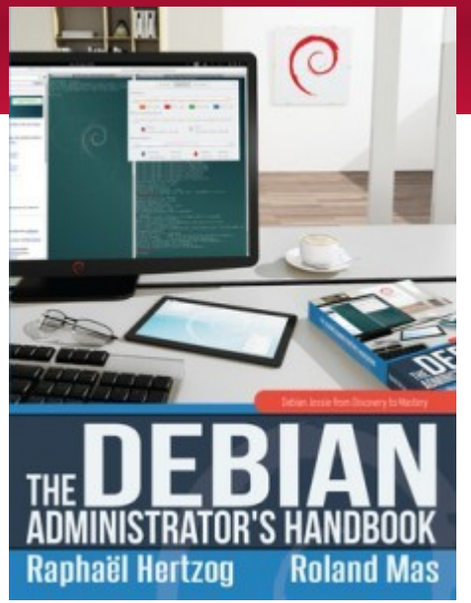
- **Histoire «continue» sur plus de 50 ans**
  - Histoire + Mémoire = culture
  - Opposition entre culture et obscurantisme
  - Savoir-faire != «recettes de cuisine»
- **Coût d'apprentissage important**
  - Investissement sur le long terme
  - Capitalisation des savoir-faire = autonomie
- **Il faut être motivé !**



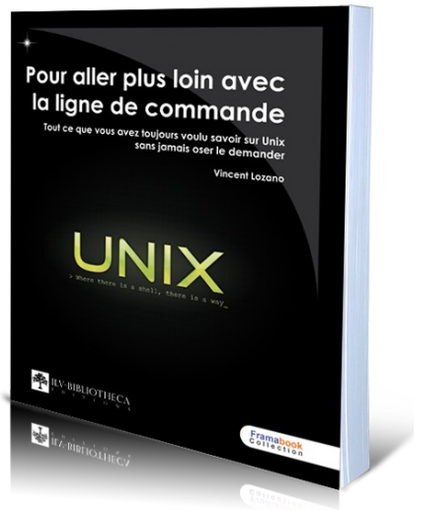


Ressources

- Cahier de l'Admin Debian
  - https://debian-handbook.info



- Framabook
  - Unix. Pour aller plus loin avec la ligne de commande
  - https://framabook.org/unixpou-allerplusloinaveclalignedecommande/



- Manuel d'installation Debian
  - http://www.debian.org/releases/stable/installmanual