Utilisation de la virtualisation:

- Rationalisation:
    - Utilisation poussée
    - OS variée
- Économie: réduction nb de serveurs mais augmentation des disponibilités
- Flexibilité: virtualisation d'anciens systèmes d'exploitation, d'applications anciennes, abstraction du matériel, automatisation des déploiements,
- Disponibilité: redondance (répliquer sur un autre serveur la machine virtuelle), sauvegarde, clonage, snapshot

Hyperviseur:

- de type II => l'OS émulé n'a pas connaissance qu'il est virtualisé (vmware, virtualbox)
- de type I => couche émulation intégré dans l'OS => avoir connaissance qu'il est virtualisé à cause des drivers (vSphere, Hyper-V, Xen, KVM)

Isolateur:

- Faire tourner plusieurs processus de la même famille (ex: serveurs web), sur la même machine et les autres processus n'auront pas connaissances des uns des autres (Linux chroot, freebsd jail, solaris zone, linux lxc (docker))

Fonctionnalités attendues:

- P2V: Migration d'une machine physique vers une machine virtuelle (conversion)
- Déplacement de VM (vmmotion)
- Répartition dynamique des vm (DRS) => déplacement d'une machine virtuelle sur un autre serveur si encombrement du disque (fait automatiquement)
- Reprise en cas de sinistre (HA, FT, SRM) => réplication de la machine au cas où un risque surviendrait => la machine virtuelle sur un site alors le second prend le relais

Infrastructures de virtualisation:

- un seul serveur
- plusieurs serveurs
- infrastructure avec stockage centralisé (vmmotion possible) => volumes montés sur les serveurs
- infrastructure de virtualisation redondante (cloud) => serveurs virtuels peuvent piloter d'autres serveurs
- infra avec service (cloud privé): ls, firewall, proxy, dispo

Utilisation en entreprises:

- Mutualisation et partage des ressources
- Indépendance vis-a-vis du matos
- haute dispo (PCA, PRA): plan de continuité et de reprise d'activité
- Virtualisation du poste de travail (VDI): virtual desktop interface
- Dématérialisation

Service providers:

- Mutualisation et partage des ressources => remplir les baies avec un max de VMs
- haute dispo (PCA, PRA)
- Isolation des VMs
- Encapsulation d'environnement complet

Conséquences de la virtualisation:
- Définition des offres de services
- Infrastructure as a Service: on ne s'occupe que de la partie application et serveur
- Plateform as a Service: on ne s'occupe que de l'application
- Software as a service: on ne s'occupe que des données

Cloud:
- Utilisation de l'application sur une machine distante
- Ressources allouées à la demande: par tranche d'utilisation
- Caractérise par sa grande souplesse

Clouds privés (entreprise)
Clouds publics: amazon, google, hewlett-packard, ibm, microsoft, oracle, sap, thales, orange

Clouds hybrides: lorsqu'une architecture d'une entreprise subit trop de fréquentation, il est possible de déborder le nombre de machines virtuels chez des clouds publics pour pouvoir profiter au max

Transfert de responsabilité:
- Réseau (SDN)
- Stockage (SDS)

VM = virtualisations CPU + mémoire + carte réseau et disque
Faire attention à leur dimensionnement

afnic: association attribution adresses.fr => nom de domaines

Nouveaux outils:
- Gestion de configuration
- Automatisation des dépendances
    - Chef
    - Capistrano
    - Puppet
    - SaltStack
- Provisioning automatique:
    - Teraform

    - Consul
    - Homemade

Capacity Planning:
- Surveille l'utilisation des ressources
    - Achat de matériel
    - Débordement sur le cloud public

Vagrant: Piloter des images de machines virtuels

Docker

- Conteneur linux qui se charge de lancerun service, décrit l'interface entre le container et son hote, contenu du container
- Enveloppe logique isolant
- Un ou plusieurs processus
- en exécution
- image d'une vm

Avantages:
- portable: facilement déployable et installable sous un environnement linux, indépendant de l'OS, propriétés natives -> références,
- disposer comme on veut
- live
- social: fonctionne en team, communication

Gestion des images quand on dev web => déploiement d'une application

Infrastructure immutable; obtention du même résultat peu importe le moment où il est exécuté

La plateforme Docker doit être tolérante à la panne
compatible blue/green: migration de version des logiciels => refaire des container et tuer les autres

Exécution des containers dans le même namespaces possible => isolation
Cgroups: permet de limiter et isoler l'utilisation des ressources (quand il y a un crash => conso mémoire trop grande => il butte le conteneur

Outils d'orchestration de container: kubernetes:
- placement sur un hôte

- élasticité
- fail-over (perte de l'hôte, perte d'un container)
- dépendances (service multi-containers)
- exposition réseau du service

Kubernetes: orchestrateur de containers, utilisé en parallèle avec Docker

POD: un ou plusieurs conteneurs
ReplicaSet: Création de plusieurs exemplaires d'un POD, s'il le pod tombe, la réplique prend le relais
Deployment: basé sur le ReplicaSet, même chose
CronJob: tâches à exécuter
Secrets: stockage mdp et identifiant de BD (coffre fort kubernetes)
Services: Démarrage de services depuis le conteneur
Ingress: les rendre accessibles de l'extérieur

Tout le contenu de PODs s'exécute sur le même NODE

Apprendre a utiliser kubernetes: minikube
https://kubernetes.io/docs/tasks/tools/install-minikube/

minkube start
kubectl get nodes
minkube dashboard

Recrutement Oxalyde:
- Questionnaires de mise en situation
- Entretien technique