



[Le Dev et l'Ops] Docker en production ?

#dockerprod

Sylvain Révéreault - @srevereault
Nicolas De Loof - @ndeloof



Who's who



CommitStrip.com



Bienvenu Chez MyCompany ...





Docker 101





1 image to Rule them all

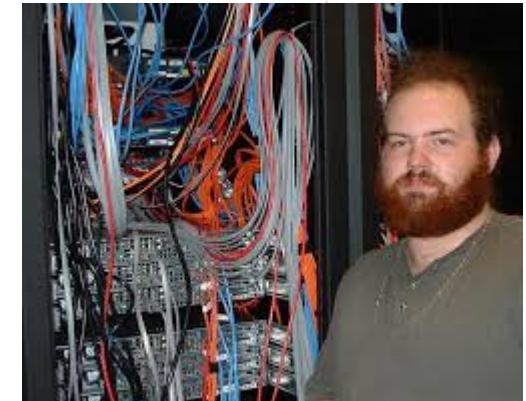
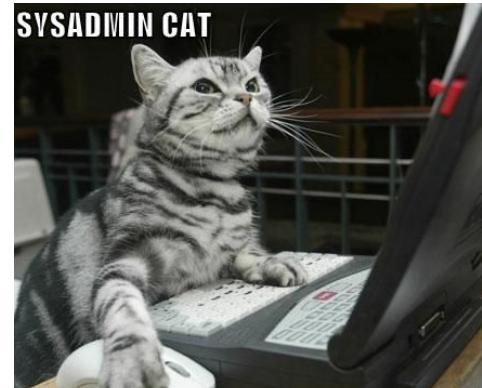
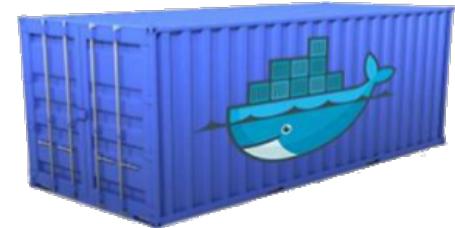
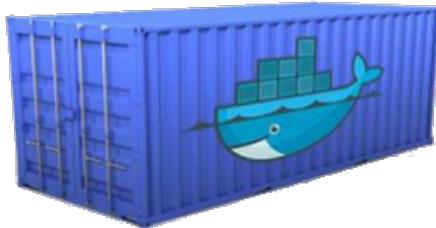
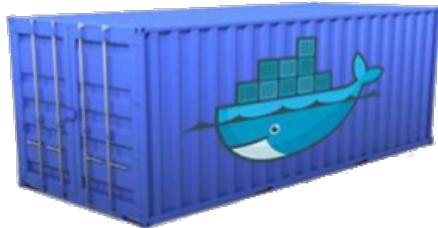
DEV

====

QA

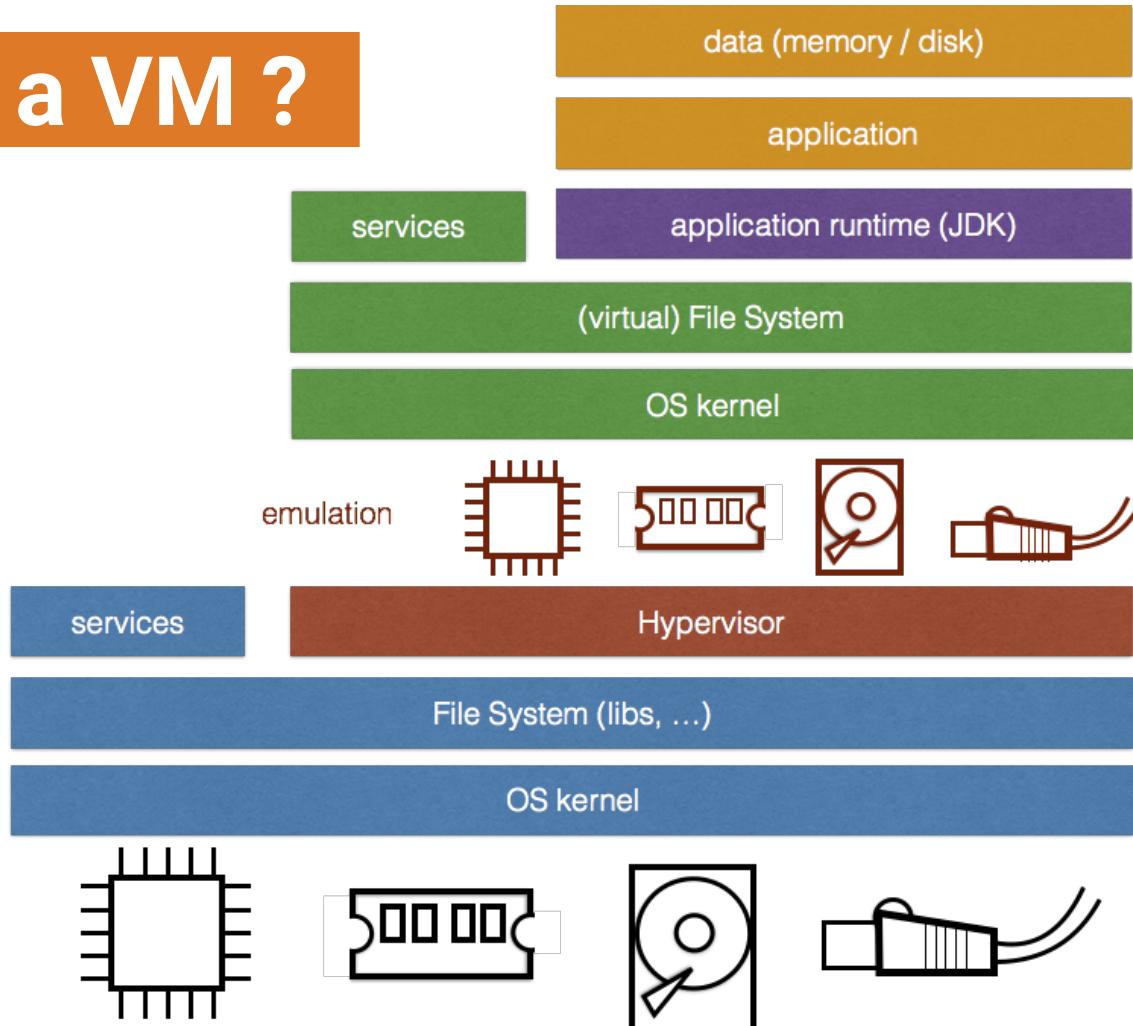
====

Prod





Why not a VM ?

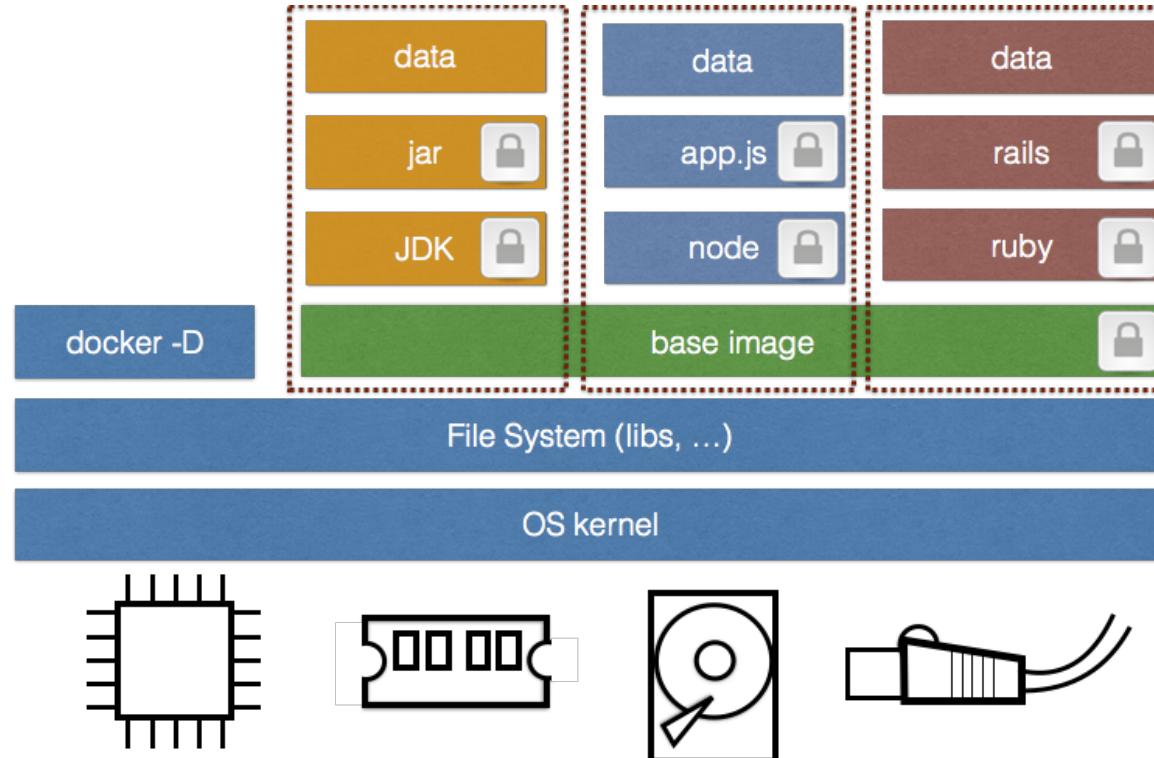




quid ?









l'avis du DEV





l'avis de l'OPS





1 an plus tôt...





Objectifs Ops

- Sécurité
- Performance
- Exploitabilité



Côté Ops : virtualisation

Historiquement : VMWare

- Solution packagée
- Objectif de rentabilisation des infrastructures
- Simplification
 - Déploiement
 - Monitoring
 - Haute dispo

=> Déport de la gestion de la disponibilité

=> Souvent un effet “double couche”
- Gestion du réseau par les hyperviseurs
- Stockage déporté (SAN)
- Adoption, stabilisation lente :
 - Perfs I/O
 - Licensing



© House of Brick Technologies



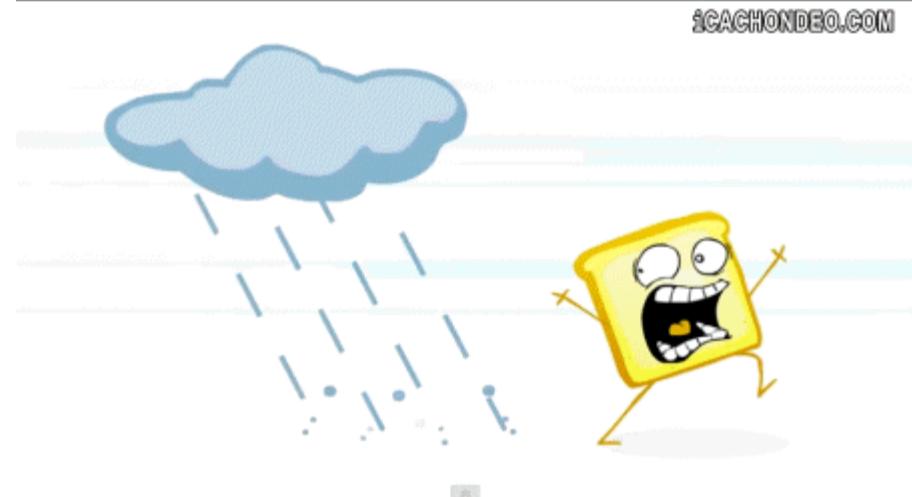
Côté Ops : Cloud, Infra as code

Evolution vers le Cloud :

- Changement de paradigme
- Cloud patterns
 - Scalabilité horizontale
 - Instances éphémères
 - Infrastructure dynamique
 - Délégation des responsabilités
 - Service Discovery

Infra as Code :

- Nouveau changement de paradigme
 - Gestion de configuration
 - Automatisation du provisioning, des déploiements
 - Passage du scripting au développement
- => Adoption lente, encore beaucoup d'administration par GUI





Côté Ops : SLA, outillage

Réponses standards Ops pour les besoins de SLA :

- Outillage de supervision
- Outillage de sauvegarde
- Experience d'exploitation
 - Capacité à configurer simplement
 - Capacité à débugger simplement
 - Capacité à mettre à jour sans risque



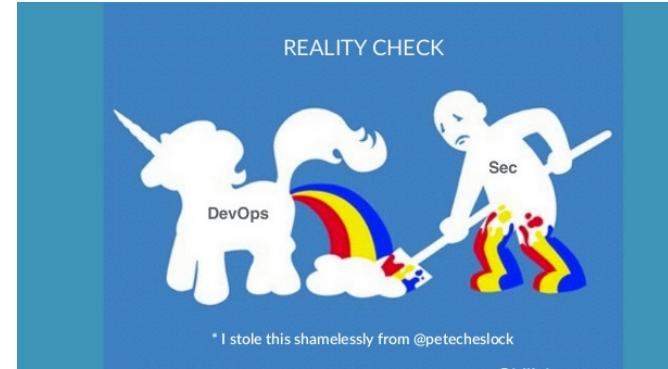
Sécurité





Sécurité

- Recherche de stabilité, maturité
- Préférence pour la version connue et suivie de la distrib
- Présence d'une équipe de sécurité “upstream”, réactivité sur les vulnérabilités
- Notion de “Long Term Support”





What about Docker ?

Maturité ? c'est en 1.6 !

Oui mais...

- Version par défaut dans Ubuntu 14.04 : Docker 1.0.1
- Support RHEL dans la section “Extras” :

Is Docker fully supported?

Yes, it is fully supported by Red Hat, but **Red Hat is requesting that customers do not deploy Docker in mission-critical production environments** due to the continuous and constant code changes upstream and rebased in RHEL 7.

Note: Red Hat supports the version of Docker provided for RHEL 7 only. Red Hat does not support docker.io or boot2docker.



What about Docker ?

On peut l'installer facilement depuis le site officiel...

=> guide d'install : wget | bash (Grrrr)
- sudo ... # et la sécurité nom de nom !

Problème de reproductibilité / traçabilité
... et de sécurité !



<https://thejh.net/misc/website-terminal-copy-paste>

<http://www.seancassidy.me/dont-pipe-to-your-shell.html>



Sécurité : packaging

Exemple de node.js : pas de QA sur les paquets npm

=> Versions figées par les développeurs quid de la gestion des CVE ?

Même symptôme sur le docker registry

=> Effet “boîte noire” pour les Ops

Over 30% of Official Images in Docker Hub Contain High Priority Security Vulnerabilities

ShellShock, HeartBleed, Poodle, ...

Source : <http://www.banyanops.com/blog/analyzing-docker-hub/>



Sécurité : packaging

Quelles solutions ?

- Rôle Ops chez les Dev ! Nom de #
- DockerHub parent image rebuild
- Docker Registry Privé... et maintenu !
- Continuous Testing
 - => Traiter tout changement de la même manière, avec la même chaîne
- Bonnes pratiques dans le Dockerfile :
 - => figer la version (reproductible) vs bénéficier des upgrades de version



Sécurité : système

Sécurité du “moteur” Docker :



Mécanismes intrinsèques :

- group “docker” Unix == root, montage de volume => 777 + partage de volumes
- pas de User Namespace pour le moment
- TLS pas activé par défaut sur le démon Docker

=> isolation : un conteneur n'est pas une boîte noire du point de vue de l'host



Sécurité : réseau

Une implémentation réseau qui sent la peinture...

- Utilisation de Linux Bridges (étanchéité ?)
- Utilisation massive d'IPTable
 - => Nécessité d'adapter les règles pour cas d'usages spécifiques
- Possibilité d'utiliser --net=host pour la perf
 - => Mais dans ce cas on perd l'isolation
- non-gestion du multi-host rend la config complexe et pénible

Solutions :

- Utilisation d'OpenVSwitch
- Weave (aka “*p##### de hack pourri*”)
- libNetwork (ex SocketPlane) = VXLAN uniquement (pour le moment) mais pas sécurisé
 - => What about IpSec ?



Sécurité : application

Bonnes pratiques de sécurité applicative (OWASP, ...)

=> A priori pas d'impact de la “conteneurisation”, plutôt un gain

Traçabilité

- Comment faire de l’Intrusion Prevention System (surveillance du FS)
=> Utiliser seulement des images Docker Read Only ?

Gestion de la restriction des accès (filesystem notamment)

- Difficultés à transposer les bonnes pratiques, sur les volumes notamment
- Difficulté à auditer, Nécessité d’auditer les conteneurs externes (cf étude Bayan)

Mise à jour des applications :

- Quelle alternative à l’infrastructure immuable ?



Sécurité : disponibilité

Docker ne fournit pas les fonctionnalités bas niveau qu'offrent la virtualisation

- live migration de conteneurs
- fault tolerance

La philosophie est différente, et se rapproche plus du Cloud

- conteneurs éphémères
- répartition forte
- service discovery
=> infrastructure immuable

Attention à ne pas créer de nouveau “Single Point Of Failure” :

- Docker Registry
- Annuaire de Service Discovery
- Outil de provisioning
- Orchestration



Performance



ForGIFS.com



Performances - scalabilité

Gestion du dimensionnement des plateformes :

- On oublie les recettes “métrologie avant mise en prod + scalabilité verticale”
... mais c’était déjà vrai pour les infras Cloud

- On vise une approche en microservices distribués
 - Spécialisation des briques
 - Idéalement, scalabilité automatique
 - Mécanisme de load balancing

Comment “load-balancer” ?

- Pas de solution native, mais des implémentations existent : <http://blog.tutum.co/2015/05/05/load-balancing-the-missing-piece-of-the-container-world/>
- Approche SDN = Load Balancing niveau TCP et/ou HTTP. Pas implémenté (pour le moment ?) dans LibNetwork
- Ou solution “standard” (HA Proxy par exemple)



Performance - tuning

Tuning : plus optimal sur Docker que sur une VM (on arrête de tirer tout un OS pour une appli)

- Mais manque d'outillage / littérature
- Quid de la capacité réelle des cgroups à limiter les impacts inter-conteneurs ?
 - Usage CPU, RAM
 - I/Os
- Beaucoup de paramètres à prendre en compte
 - I/Os disques : devicemapper, aufs, btrfs, ...
 - Topologie réseau
 - Paramétrage des cgroups
-

Quelques ressources :

- <http://devconf.cz/files/slides2015/friday/Performance%20Tuning%20of%20Docker%20and%20RHEL%20Atomic.pdf>
- <http://www.breakage.org/2014/06/06/getting-started-with-performance-analysis-of-docker/>



Performance : placement sur l'infra

But : optimisation du placement des conteneurs en fonction du type de contenu, des perfs (CPU, mémoire, I/Os) attendues

Côté virtualisation, disponible sur les outils avancés

- Dans VMWare vSphere, optimisation via DRS notamment
- Dans OpenStack, algorithmes disponibles dans Nova
- Amazon : voir les travaux de benchmark de Netflix : <http://fr.slideshare.net/brendangregg/performance-tuning-ec2-instances>

Côté Docker, algorithmes de placement des orchestrateurs encore incomplets :

- libswarm : mécanismes d'affinité / anti-affinité, utilisation de "constraints"
- kubernetes : prévu mais pas implémenté <https://github.com/GoogleCloudPlatform/kubernetes/blob/master/docs/resources.md>



Exploitabilité





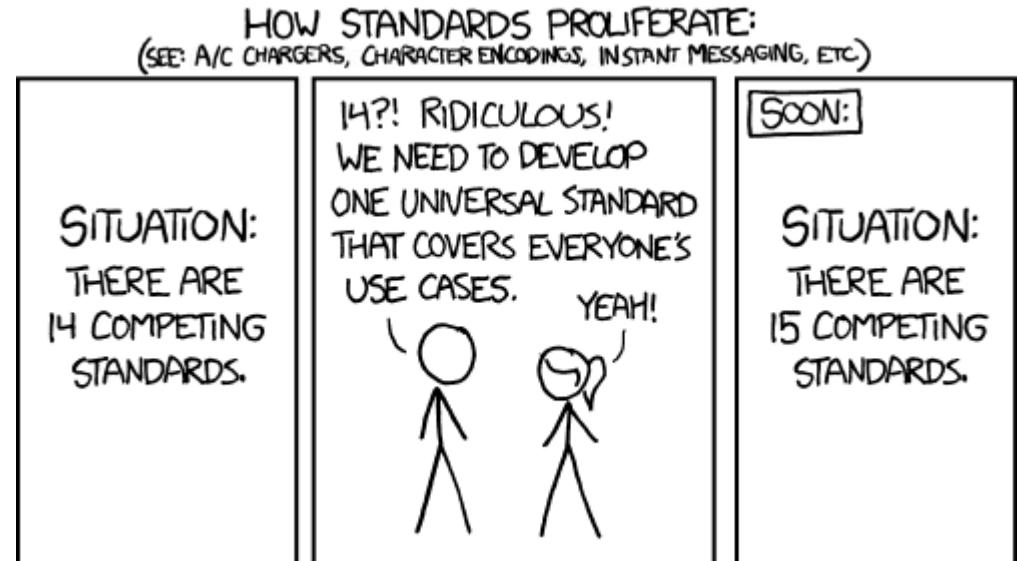
Exploitabilité : distribution

Chouette, un nouveau format de packaging !

- Dommage, on avait (enfin) habitué les Devs à fournir du RPM/Deb
- Au passage, on perd les pre/post-install hooks...

=> Modification du système de configuration management.

=> Importance du registry privé



source : <http://xkcd.com/>



Exploitabilité : supervision

La supervision “traditionnelle” passe souvent par un agent :

- Agent SNMP
- Agent NRPE (Nagios)

=> ne pas reproduire les habitudes prises avec les VMs (1 VM = 1 agent) : va à l'encontre du principe “un process par conteneur”

Avec Docker, l'approche est différente :

- Agent sur l'hôte, avec plugin spécifique (Nagios-docker, Sensu, ...)
- SaaS monitoring : New Relic, DataDog, ...
- Outils spécifiques de supervision
 - Docker stats
 - Google CAdvisor
 - ...

Ne pas oublier de moniter le démon Docker !





Exploitabilité : logs

Ici aussi, changement de paradigme :

- Pas de serveur Syslog embarqué dans chaque container
- Eventuellement, envoi vers le Syslog de l'hôte
- Ou encore log fichier dans un volume
- Docker logs... really ?
- Idéalement, solution de centralisation des logs
 - Syslog centralisé
 - Logstash for the win !
... mais privilégier un système asynchrone pour les applications à fort volume de log



Exploitabilité : configuration

Enjeu : donner la possibilité aux Ops d'être autonomes sur un changement de configuration “à chaud”

- modification d'une adresse de base de données
- changement d'un plan d'adressage
- ...

=> Externalisation des paramétrages

Bonnes pratiques Docker :

- Utiliser les variables d'environnement (--env)
- Linker les containers (mais limité au même host)
- Utiliser les mécanismes basés sur le fichier host (--add-host)
- Service Discovery, pattern “ambassador”



Exploitabilité : backups/restore

Les solutions sur virtualisation classique permettent des cas d'usage très évolués :

- Backup incrémental
- Restauration fine (fichier par fichier)
- Snapshot
- Déduplication
- ...

Docker impose une nouvelle approche data-centric

- volumes, volumes, volumes
- séparation forte data/application
- quid du backup des layers de containers (et surtout de la restauration) ?
- quelques pointeurs :
 - <https://registry.hub.docker.com/u/yaronr/backup-volume-container/>
 - <http://alvinhenrick.com/2015/01/26/docker-backup-and-restore-volume-container/>
 - <https://github.com/docker-infra/docker-backup>



Conclusion

Docker ne facilite pas vraiment la vie de tes Ops, pour la majorité ça leur rajoute plutôt des contraintes.

Nécessité d'un travail commun entre Dev et Ops sur la mise en place d'une infra faite pour héberger du conteneur :

- => beaucoup de questions en suspens, de "work in progress"
- => paradigme d'infrastructure immuable
- => nouveaux modèles (scaling, service discovery, ...)



Cependant, les architectures systèmes à base de conteneurs représentent une évolution positive :

- => depuis 10 ans, la virtu pousse vers une mauvaise direction "OS over OS"
- => émergence d'un nouveau socle Ops "plateforme" : CoreOS, Ubuntu Core, Rancher OS, ...
- => nécessité d'intégrer ces nouveaux modèles tout en maintenant la qualité de service existante



Conclusion

Ne mettez pas Docker en production...

- Si vous avez un patrimoine important à migrer
- Si vous pensez gérer vos containers comme vous gérez (mal?) vos VMs
- Si vous pensez que l'isolation va résoudre tous vos problèmes de sécurité
- Pour faire plaisir au copain développeur

Mettez Docker en production...

- Si vous partez "from scratch" ou presque
- Si vous mettez en place des architectures micro-service
- Si vous mettez en place une infrastructure immuable
- Si vous croyez, comme nous, au modèle des OS "Core"
- Si vous êtes prêts à faire les investissements nécessaires pour bénéficier des avantages de ce modèle



Questions et discussion

