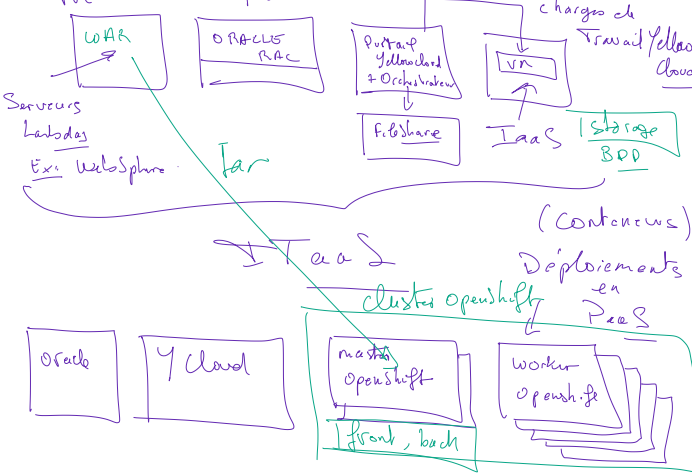
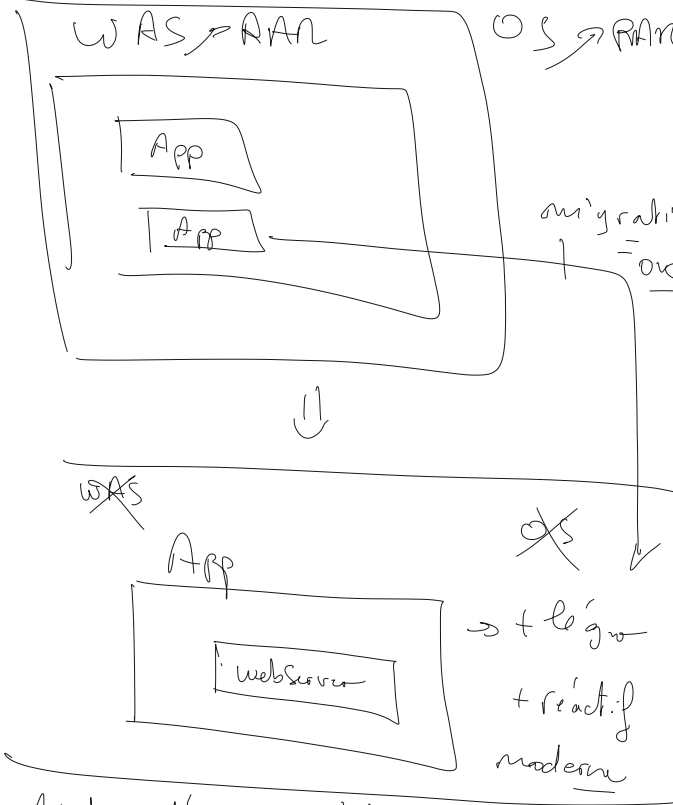
DC :

Machines

VM = OS + Disques





Avantages d'un Openshift Privé (local)

- → Maîtrise de coûts
- → Souveraineté / Gouvernance / Sécurité
- → Moderne / Souple et réactif (Montée en charge)

→ ! Capacity planning

⇒ Compétences — quelques années

Service Level Agreement

SLA 99% = engagement de 99% de qualité (up and running)
autorise 1% de downtime

99% par jour = 864 secondes / jour de downtime =
(2 - 9^H) = 14 minutes

bon
99,99 → 4 - 9^H → 8,64s/jour
excellent
99,9999 → 7.9^th → 0,08 s/j

Disponibilité = système actif (il peut être saturé)

Répartition de la charge = dispatcher le travail

Haute disponibilité = Dispo + Répartition (HA)
↳ SLA 99_ ....

BDD SQL

Oracle SLA = 99 %

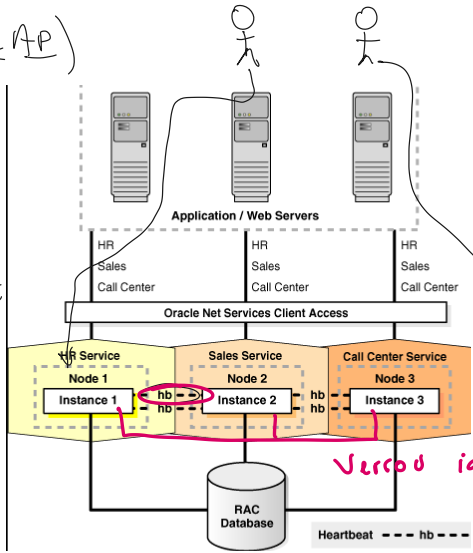
Oracle RAC SLA = 99,999 %

Internet ~~SQL~~ → Pas adapté

→ NoSQL ⇒ SLA = 99,9999 .....

PAS FORCEMENT COHERENT

(CAP)



UPDATE id = 1

t0

t : A → B

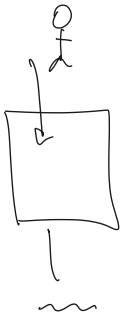
UPDATE id = 1

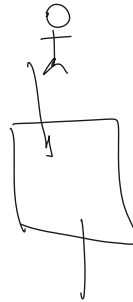
t 1

B

Verrou id = 1

No SQL





id = 1

t0

V = ~~A~~ B

"mise à jour"

Co accès sans synchronisation

id : 1

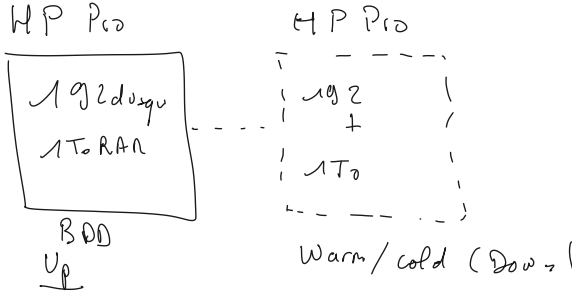
t1

N = R

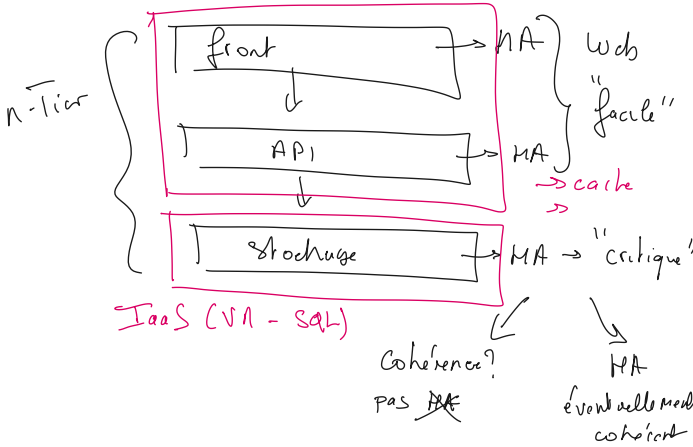
t2 = V = B

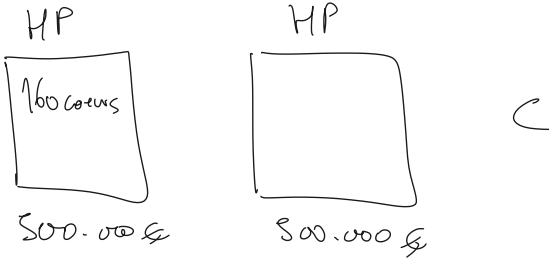
le dernier a raison

HA en SQL

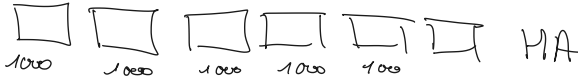


Le bon Coin "Moderne" PAAS openshift





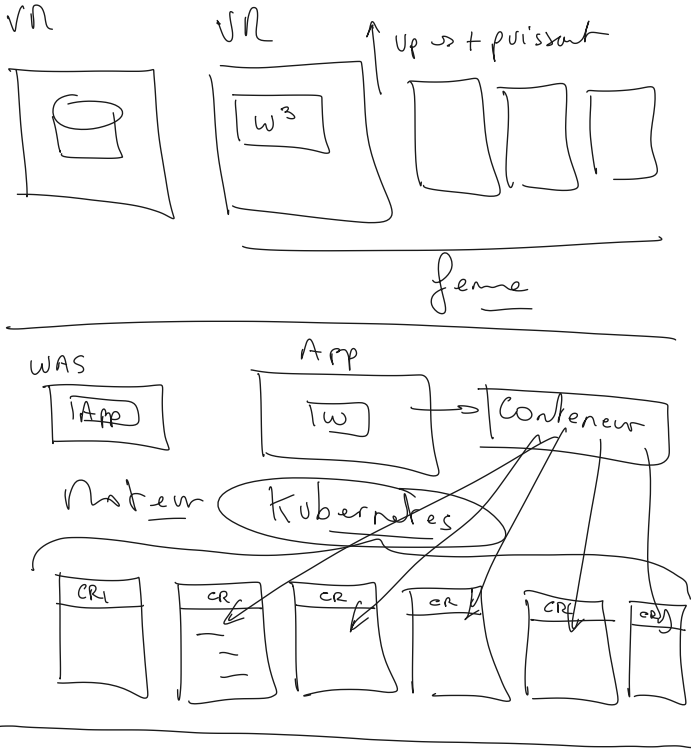
8 coeurs
64 go



→ ex. | 256 coeurs, 2 To RAM } ≃ 80.000 €

→ Maj "Rolling" - Sans coupure

→ Maj "Recreate" - avec coupure

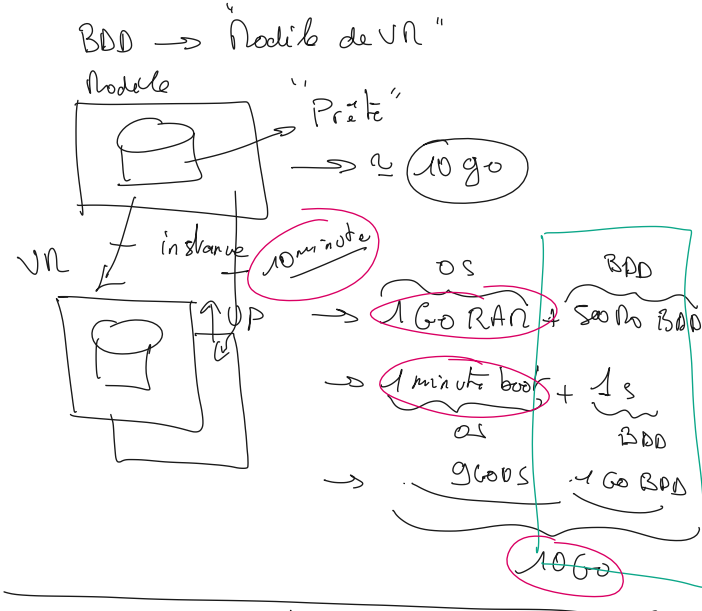


OpenShift

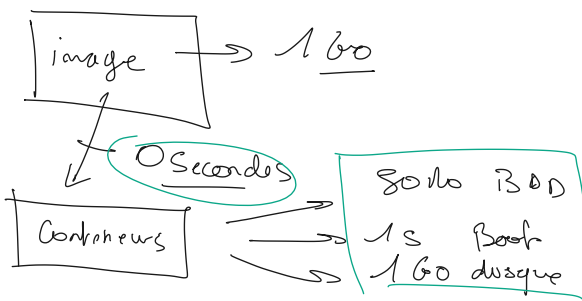
Kubernetes → HA. très technique "pas user friendly"

Docker → couche Basse (Pas HA)

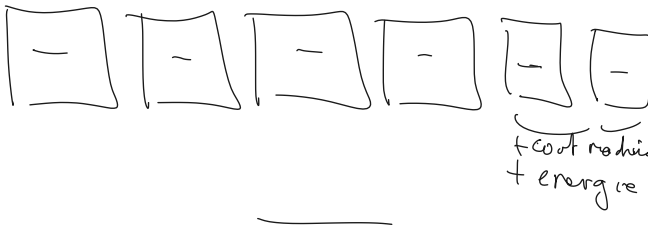
IaaS :



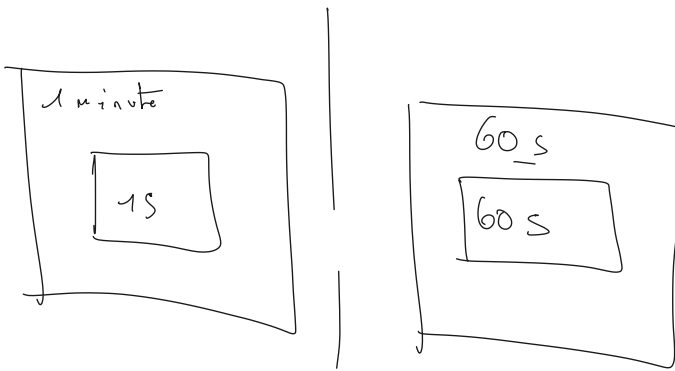
Conteneurs



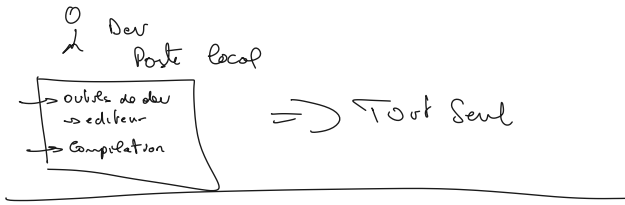
VM

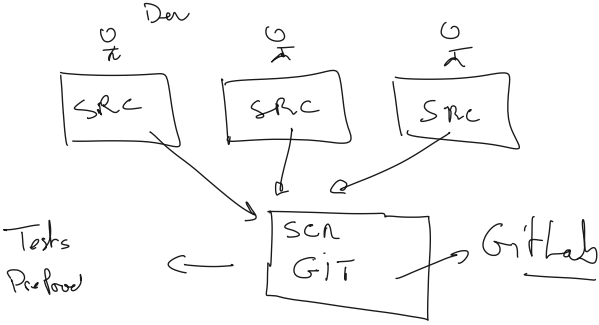


Conteneur



Dev "Simple"



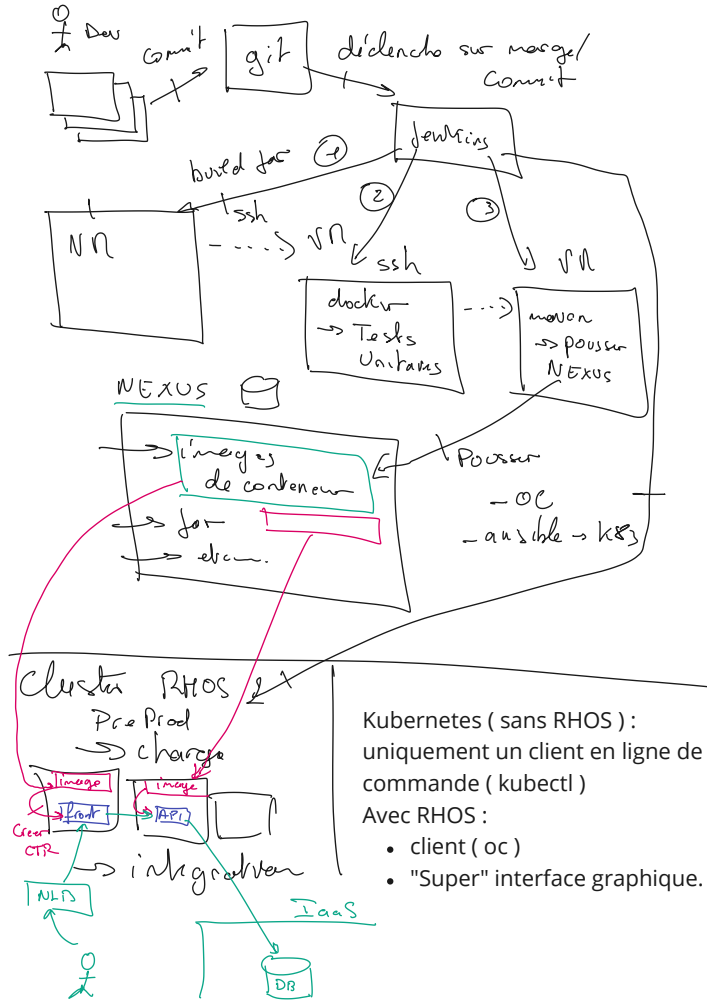


→ Tests Unitaires → Qualité
- failles
- Normes
- dépendances (CVE, virus)
→ Intégration
- charge
- Recette
→ build de l'image → etc...

JAR

Intégration Continue

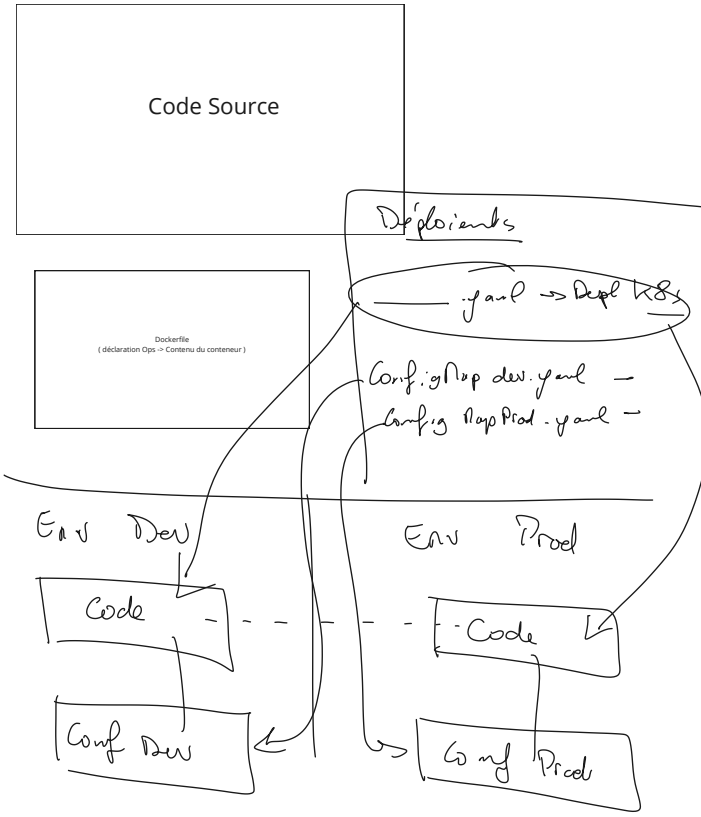
Orchestrateur = Jenkins



Kubernetes (sans RHOS) : uniquement un client en ligne de commande (kubectl)

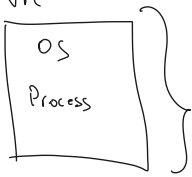
Avec RHOS :

- client (oc)
- "Super" interface graphique.


Code Source
Déploiements
.yaml -> Depl K8s
Dockerfile
(déclaration Ops -> Contenu du conteneur)
Config Map dev.yaml
Config Map Prod.yaml
Env Dev
Env Prod
Code
Code
Conf Dev
Conf Prod

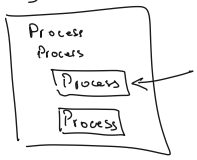
Conteneurisation est un vieux concept

VM



"Virtualisation d'OS"

OS



"Virtualisation d'application"

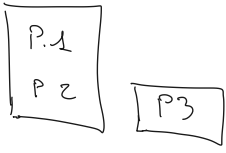
Anciennes Technos de Conteneurs:

- AIX Workload partitions
-
-

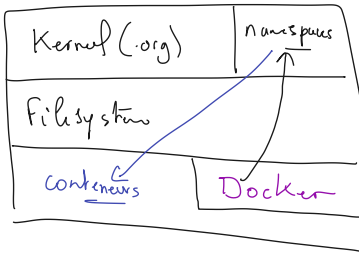
Linux (Kernel - noyau): "namespaces"

↓

isolation

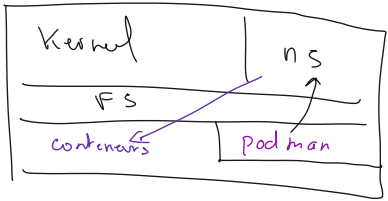


OS (Distrib)



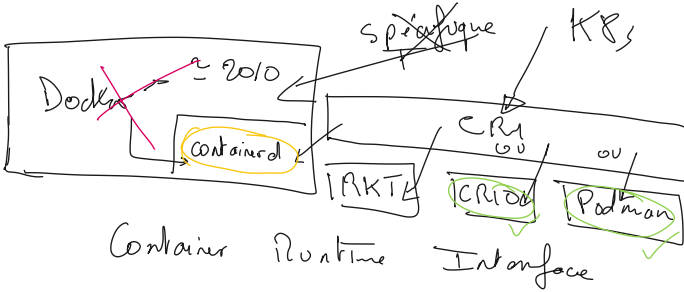
Distribution
ex Centos 6
Ubuntu 22
etc.

→ côté Dev

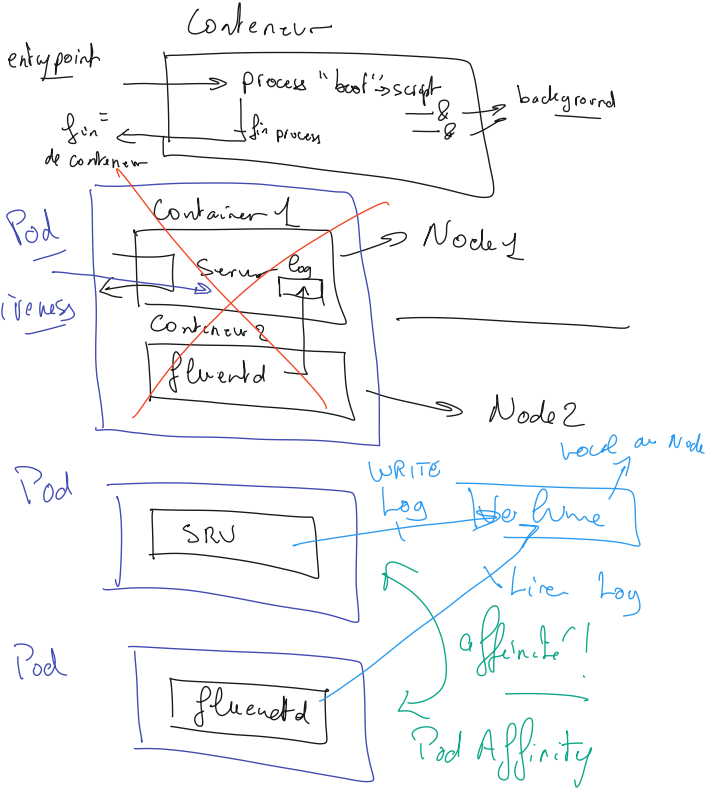


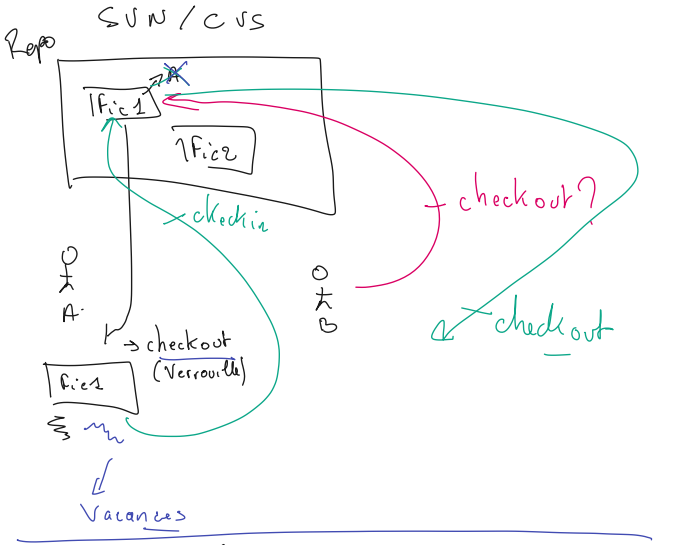
?

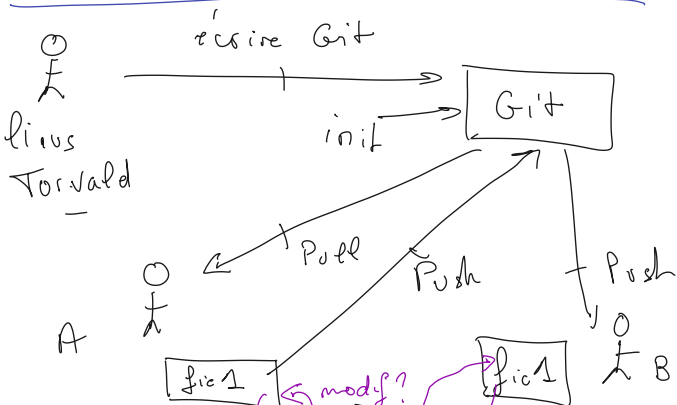
→ Node Kubernetes

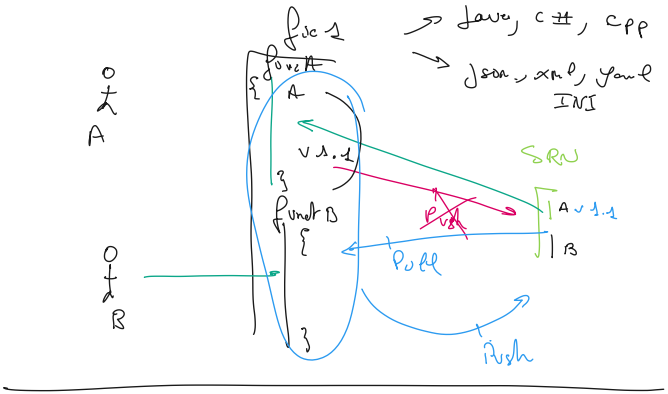


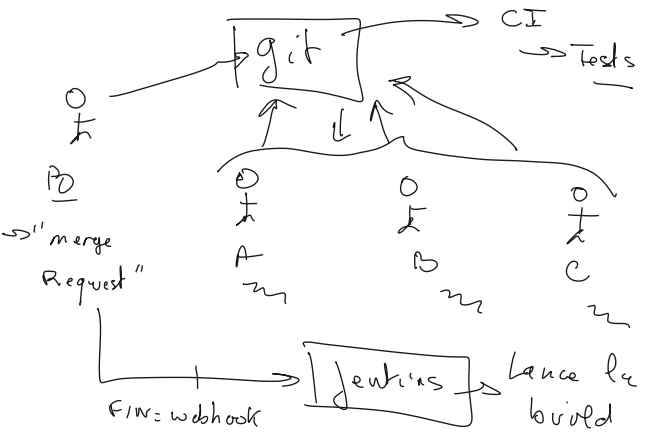
Container Runtime Interface

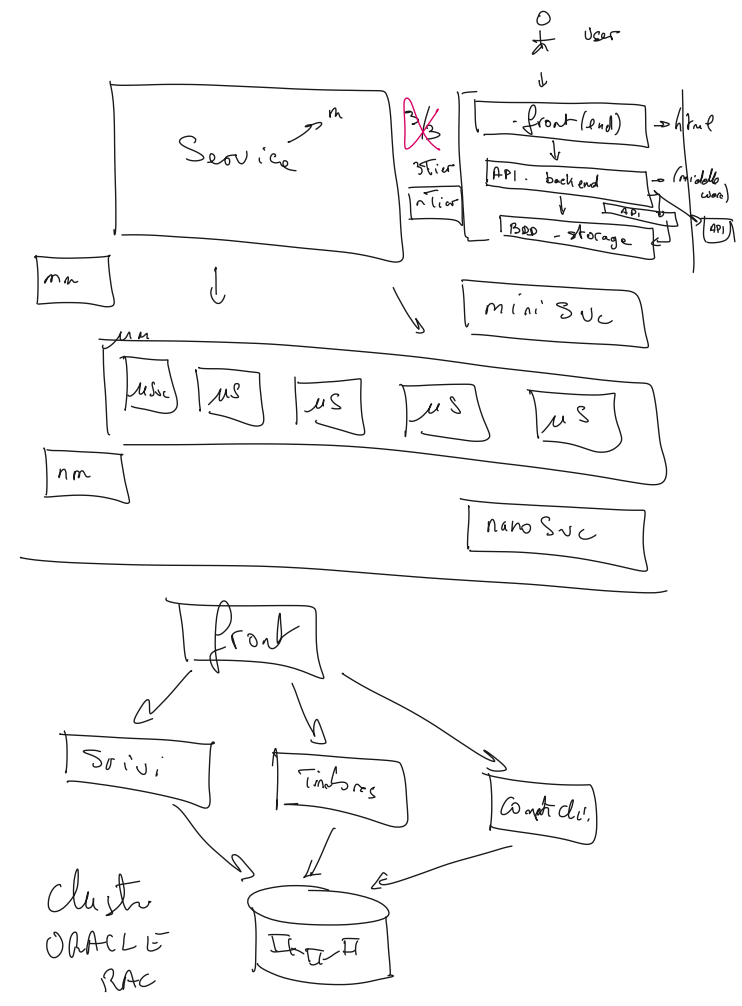

Container
entrypoint
process "boot" -> script
&
&
background
fin =
de container
fin process
Pod
Container 1
Node 1
Serv
log
liveness
Container 2
fluentd
Node 2
Pod
WRITE
Log
Volume
Local au Node
SRV
Live Log
affinité !
Pod
fluentd
Pod Affinity

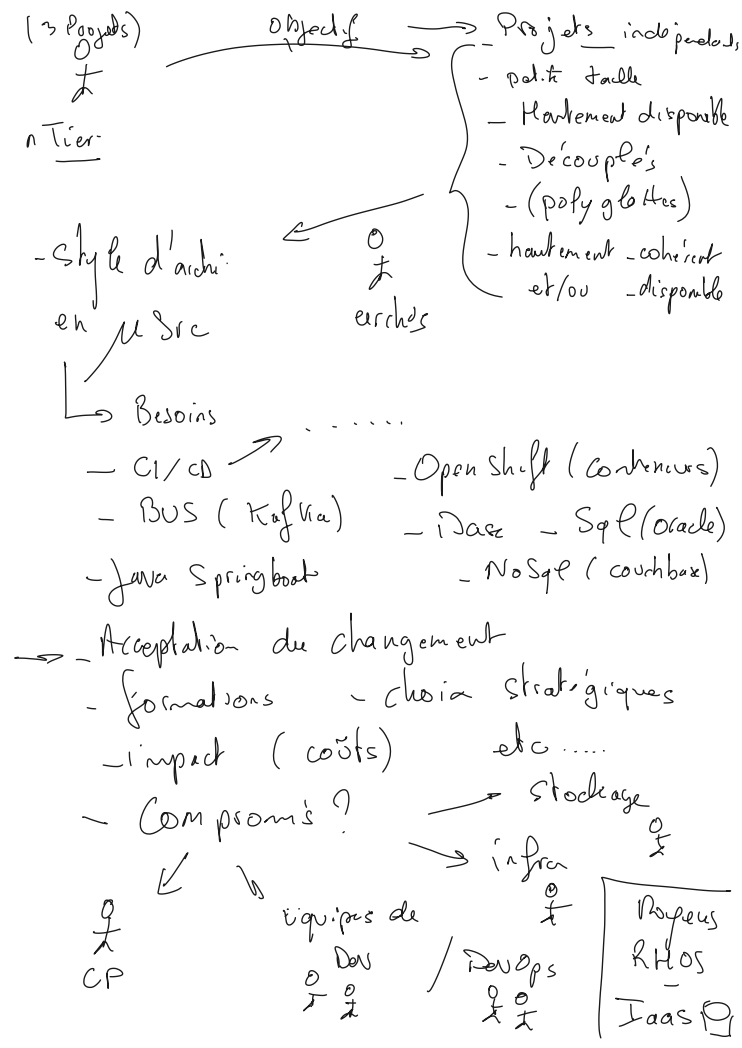

SVN / CVS
Repo
Fic 1
Fic 2
checkin
checkout ?
checkout
A
B
checkout
(Verrouillé)
fic1
Vacances


écrire Git
Linus
Torvald
Git
init
Pull
Push
Push
A
B
fic 1
modif ?
fic 1


fic 1
Java, C#, Cpp
Json, xml, yaml
INI
A
A
v1.1
SRV
?
Function B
A v1.1
B
Pull
Push
B


git
CI
Tests
PO
"merge Request"
A
B
C
Jenkins
Lance la build
Fire webhook

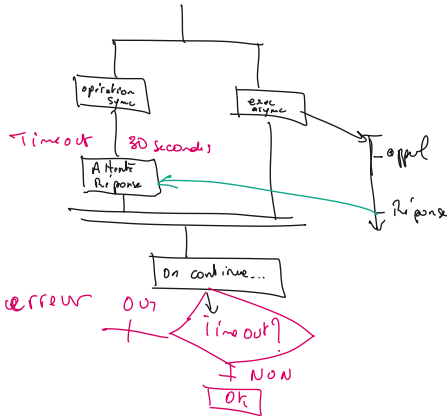

User
Service
n
3 Tier
n Tier
- front (end)
html
API - back end
(variable code)
API
BDD - storage
API
m
mini SVC
μs
μs
μs
μs
μs
nano SVC
Front
Service
Timbres
Compt cli.
Cluster
ORACLE
RAC


(3 Paquets)
n Tiers
Objectifs
Projets indépendants
- petite taille
- Hautement disponible
- Découplés
- (polyglottes)
- hautement - cohérent
et/ou - disponible
- Style d'archi
en µ Src
archis
Besoins
- CI/CD
- BUS (Kafka)
- Java Springboot
- OpenShift (conteneurs)
- IAas - Sql (Oracle)
- NoSql (couchbase)
- Acceptation du changement
- formations
- impact (coûts)
- Compromis ?
- choix stratégiques
etc ....
stockage
infra
CP
équipes de Dev / DevOps
Paqueus
RHOS
Iaas

opération = exécution de code

message = contenu (données)

- Asynchrone → + complexe
  - pas de gestion des pbm
  - "email / Texto"
- Synchrone → + simple
  - gestion des conséquences
  - ↳ Dépends de la disponibilité
  - → "appel Téléphonique"



SQL

- Recherche Texte → occurences
- Recherche FullText (+puissant)
  ↳ nb occurences
  ....

≃ 2000 Doug Cutting → "Lucene" → indexeur Sémantique très puissant
→ mots proches d'autres mots - Correspondances
→ phonétique -

↓ évolué côté Serveur

- Elastic Search - → Serveur Technique.

↓ évolué

Elastic Search

- Récupération de Logs Beats → FLUENTD
- Transform & nettoyage de Logs Logstash
- Portail graphique d'exploitation Kibana

"Stack" ELK