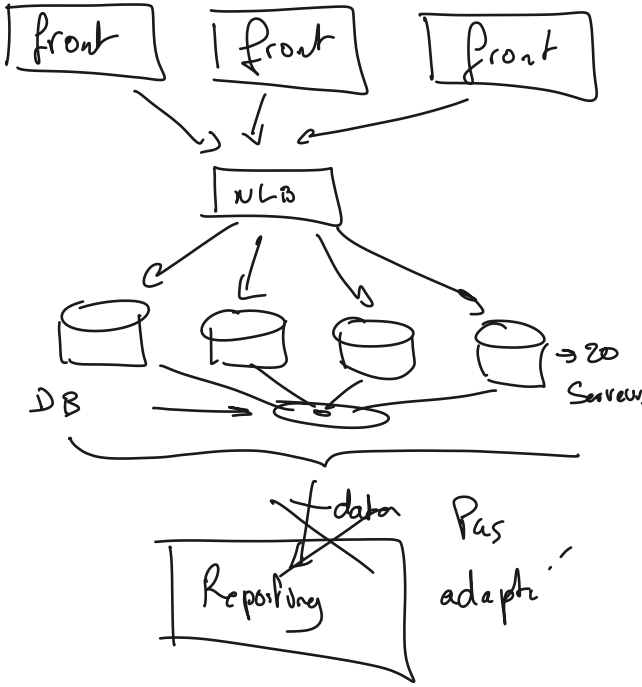
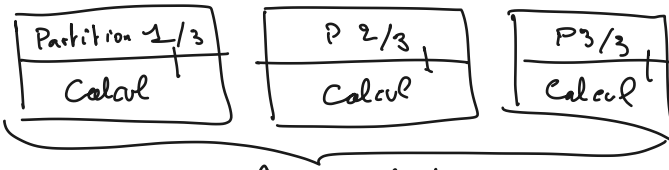

front
front
front
NLB
DB
200
Serveurs
data
Repository
Pas
adapté



données
| (grande quantité)

→ Calcul distribué.

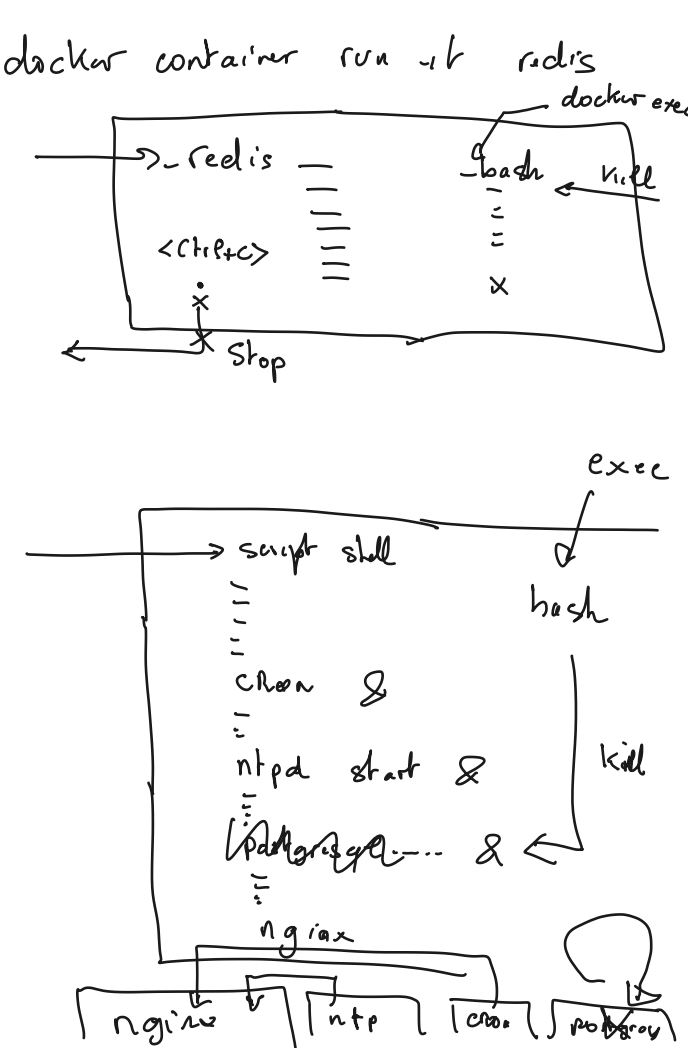
Java 8 Stream

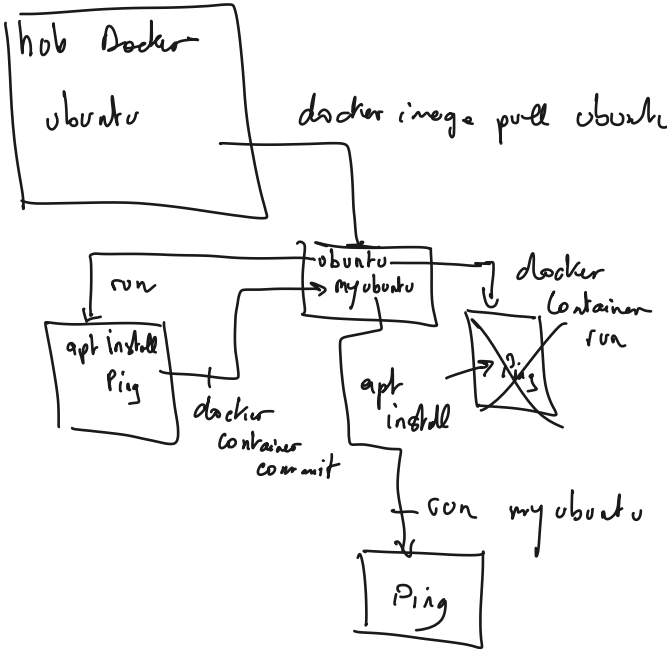
map(__)

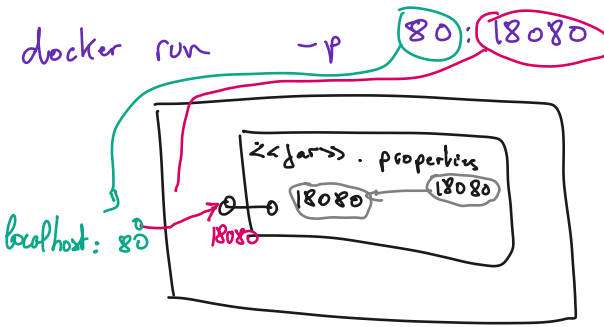
SIMD

CPU

→ for(. . .)

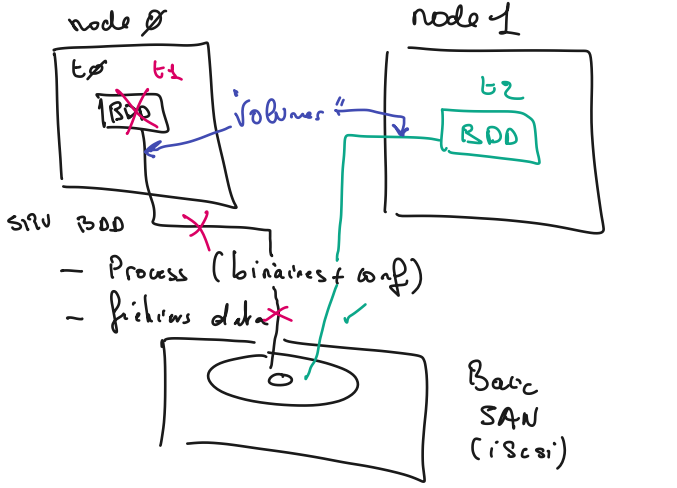

docker container run -it redis
docker exec
redis
bash
kill
<Ctrl+C>
stop
exec
script shell
bash
cron &
ntpd start &
kill
nginx
nginx
ntp
cron
postgres

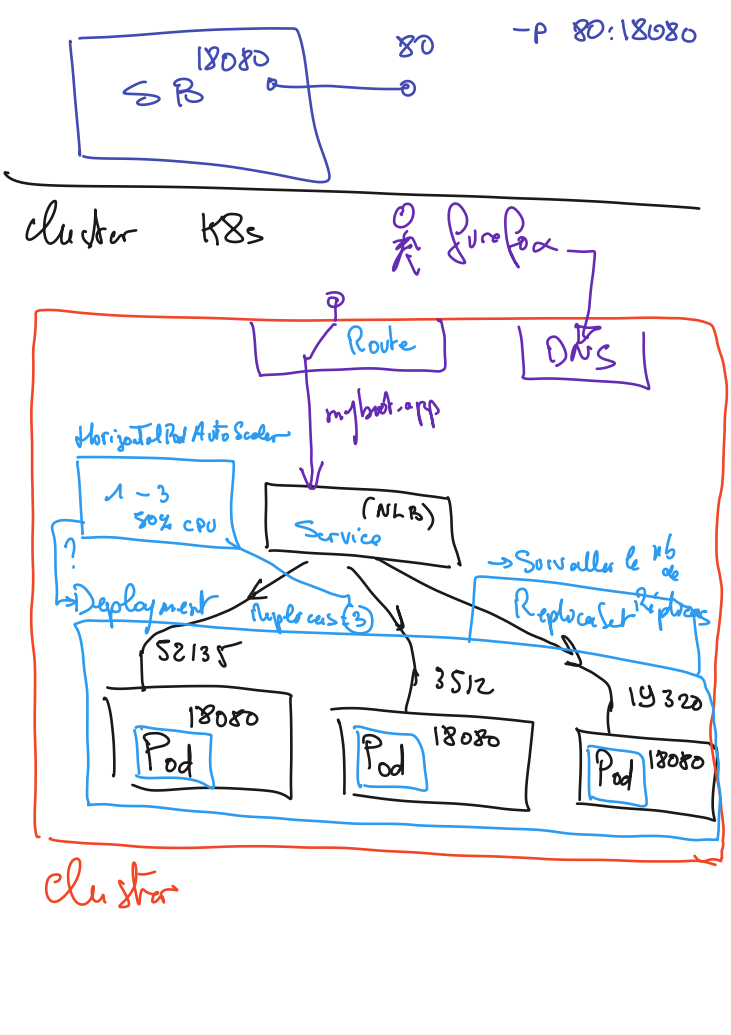

hub Docker
ubuntu
docker image pull ubuntu
ubuntu
myubuntu
run
apt install
Ping
docker
container
commit
docker
container
run
Ping
apt
install
run myubuntu
Ping

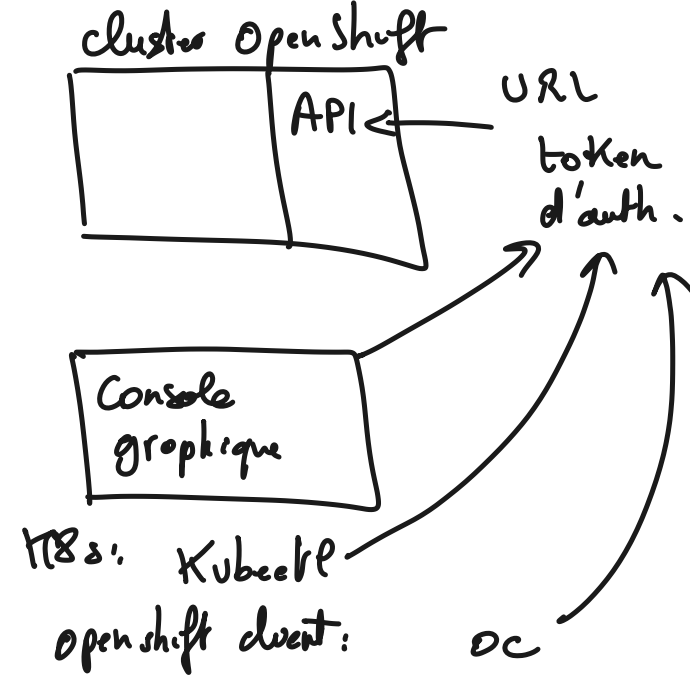


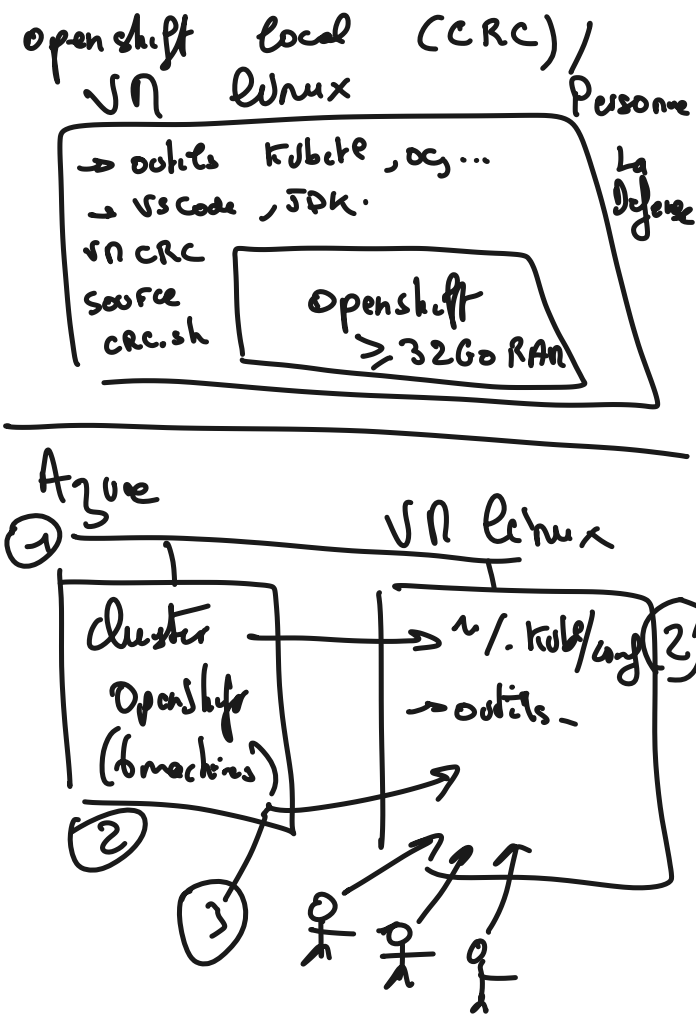
EXPOSE 18080 : purement informatif, indique que cette image fournit un service qui écoute sur ce port

---




-p 80:18080
18080
80
SB
cluster K8s
firefox
Route
DNS
mybot.app
Horizontal Pod Auto Scaler
1 - 3
50% CPU
(NLB)
Service
Deployment
Replicas (3)
Surveille le nb de Replicas
52135
3512
19320
18080
Pod
18080
Pod
18080
Pod
cluster


cluster OpenShift
API
URL
token
d'auth.
Console
graphique
K8s: Kubectl
openshift client: oc


openshift local (CRC)
VM Linux
Personne
→ outils kubctl, oc, ...
→ vscode, JDK.
VM CRC
source
crc.sh
Openshift
≥ 32Go RAM
La Défense
Azure
VM Linux
1
cluster
Openshift
(6 machines)
~/.kube/config
2'
→ outils
2
3

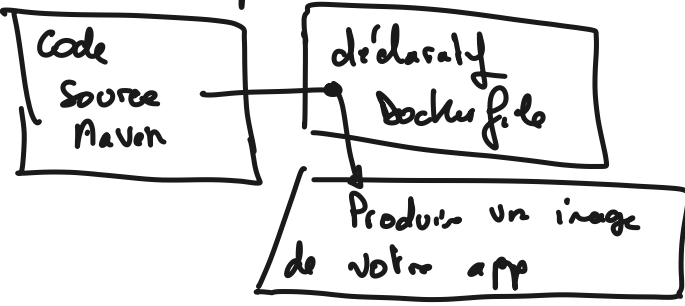
Iaas → VM → Tout est possible, mais nous sommes responsables de tout

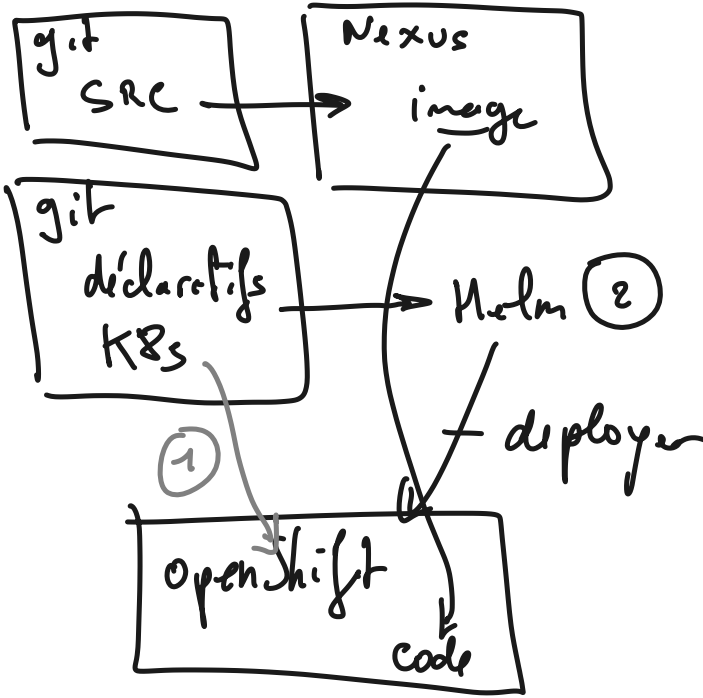
(a éviter)

Préférer : PaaS

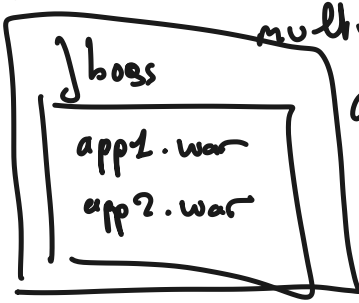
① PaaS "hebergement"
→ DB géré (postgres ....)
→ Serveur web (php, Python...)

② PaaS "Conteneurs"
(openshift)




git
SRC
Nexus
image
git
déclaratifs
K8s
Helm
2
deployer
1
OpenShift
code

Jboss = Conteneur Web multi apps



Solution recommandée = "auto hostée"

Spring boot

"juste ce qu'il faut" → + léger que Jboss.

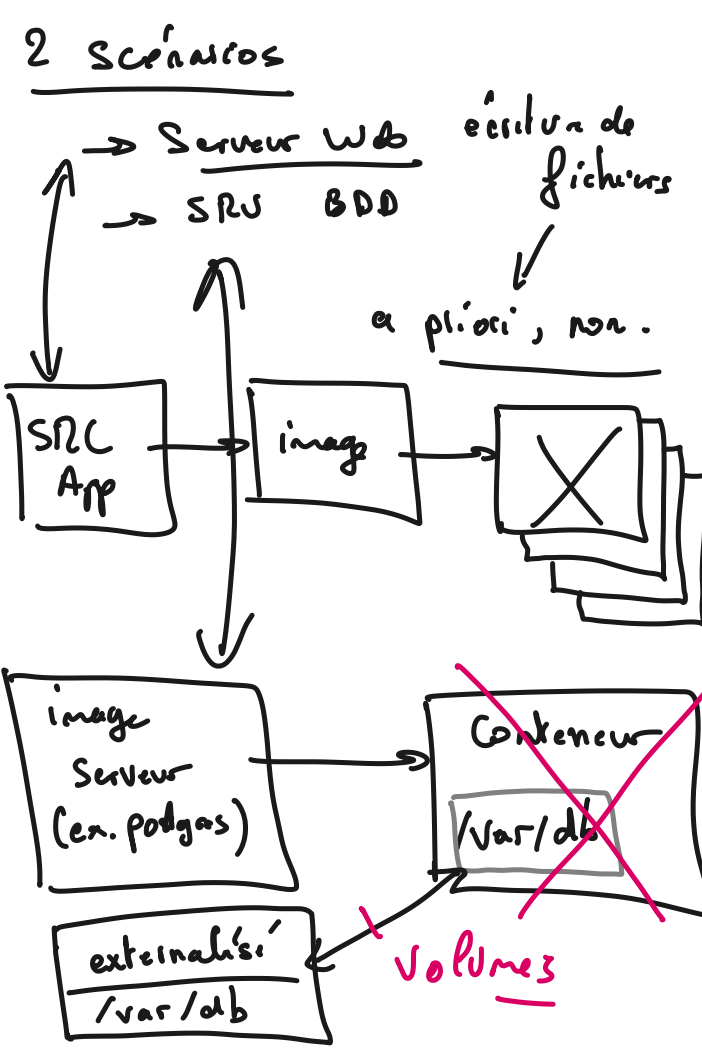
2 scénarios

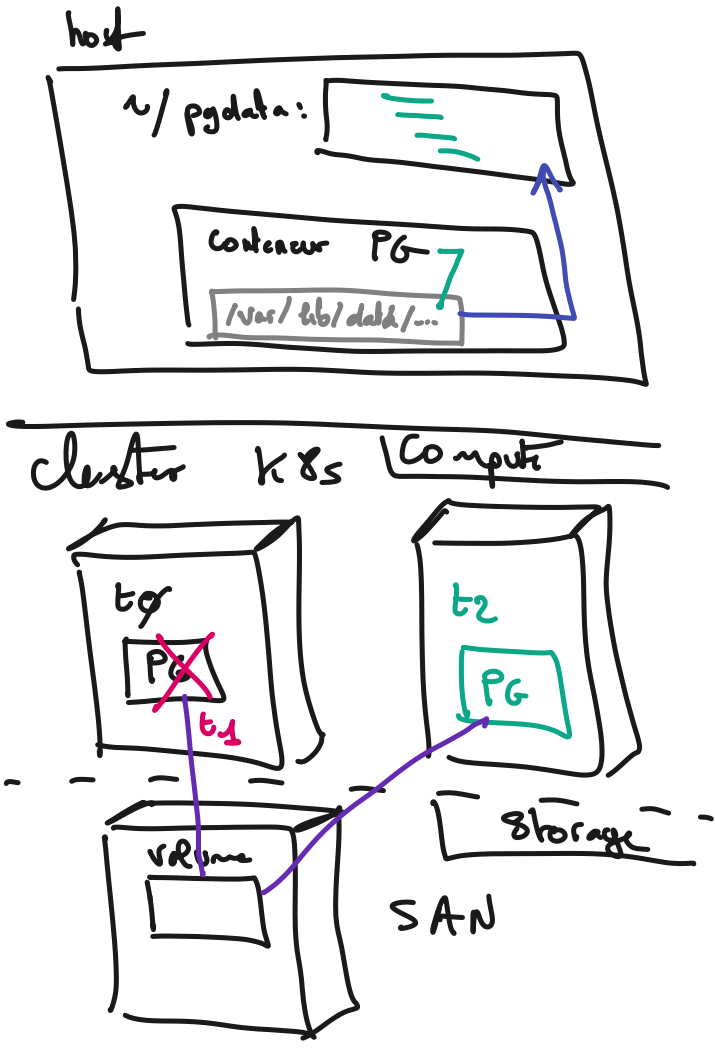
→ Serveur Web

→ SRV BDD

écriture de fichiers

a priori, non.




host
~/pgdata:
container PG
/var/lib/data/...
cluster k8s
Compute
t0
PG
t1
t2
PG
storage
volume
SAN

https://start.spring.io/ :
Projet Maven, Jar, java 17
Dependances :
Spring Web
Spring Boot Actuator

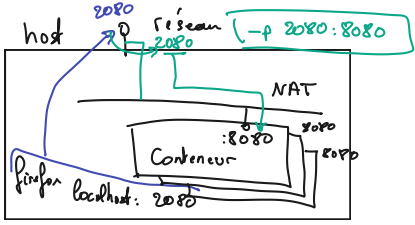
-> generate, décompresser,
sous la console :
st20@StudentVM:~$ **unzip Downloads/demo.zip**
cd demo : vous êtes dans le modèle de votre projet.

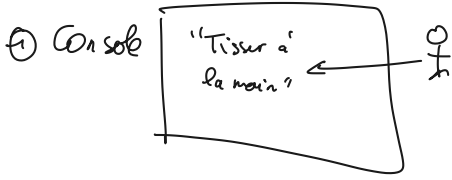
Compiler : ./mvnw package -e
Executer : ./mvnw spring-boot:run -Dmaven.test.skip=true
configurer le port d'écoute :

lancer l'éditeur : code .
editer src/main/ressources/application.properties :
#debug=true
server.port=18080 (par exemple)
management.endpoint.health.show-details=always

connectez vous sur : http://localhost:<votre port>/actuator/health

Conteneuriser ce projet :
Objectifs :
Créer une image, qui ne doit pas être root pour être compatible avec openshift (et donc ne pas écouter sur un port <1024)
ensuite pousser cette image dans le référentiel docker en ligne (hub) -> créez un compte.
Tester cette image dans openshift





2 tisser "en ligne de Commande"

→ créer le Déploiement
→ exposer le Deploi → un service
→ exposer le Service → Une Route

oc status
In **project default** on server https://api.yfjhunug.francecentral.aroapp.io:6443

svc/openshift - kubernetes.default.svc.cluster.local
svc/kubernetes - 172.30.0.1:443 -> 6443

View details with 'oc describe <resource>/<name>' or list resources with 'oc get all'.

oc project **st20**
Now using project "st20" on server
"https://api.yfjhunug.francecentral.aroapp.io:6443".
st20@StudentVM:~/demo$ oc status
In **project st20** on server https://api.yfjhunug.francecentral.aroapp.io:6443

You have no services, deployment configs, or build configs.
Run 'oc new-app' to create an application.

st20@StudentVM:~/demo$ oc status
In project st20 on server https://api.yfjhunug.francecentral.aroapp.io:6443

You have no services, deployment configs, or build configs.
Run 'oc new-app' to create an application.
st20@StudentVM:~/demo$ oc create deployment demospringboot --image=nboost/demospringboot
deployment.apps/demospringboot created
st20@StudentVM:~/demo$ oc expose deployment/demospringboot --port=8888
service/demospringboot exposed
st20@StudentVM:~/demo$ oc expose svc/demospringboot
route.route.openshift.io/demospringboot exposed
st20@StudentVM:~/demo$ oc status
In project st20 on server https://api.yfjhunug.francecentral.aroapp.io:6443

http://demospringboot-st20.apps.yfjhunug.francecentral.aroapp.io to pod port 8888 (svc/demospringboot)
  deployment/demospringboot deploys nboost/demospringboot
    deployment #1 running for 30 seconds - 1 pod

1 info identified, use 'oc status --suggest' to see details.

Impératif

- cmd 1
- " 2
- 3
- X (exception)

} usine de code

script de création → alias

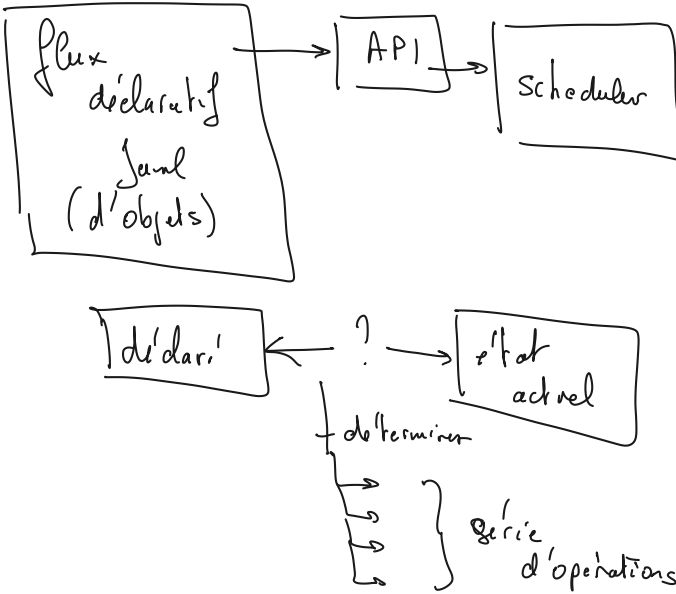
script de modif. → alias

Script de soo

déclaratif

déclarer ce qui est souhaité (stratégique) (peut y avoir un certain ordre)

- Pas d'ordre sous K8s
- ordre en gestion de conf (Ansible)



Kubernetes → Types d'objets
↳ instances.

ex: Pod
Service
Route
Deployment

StatefulSet
HorizontalPodAutoScaler
Job
DaemonSet.

(1) oc create << comme Cundo >>

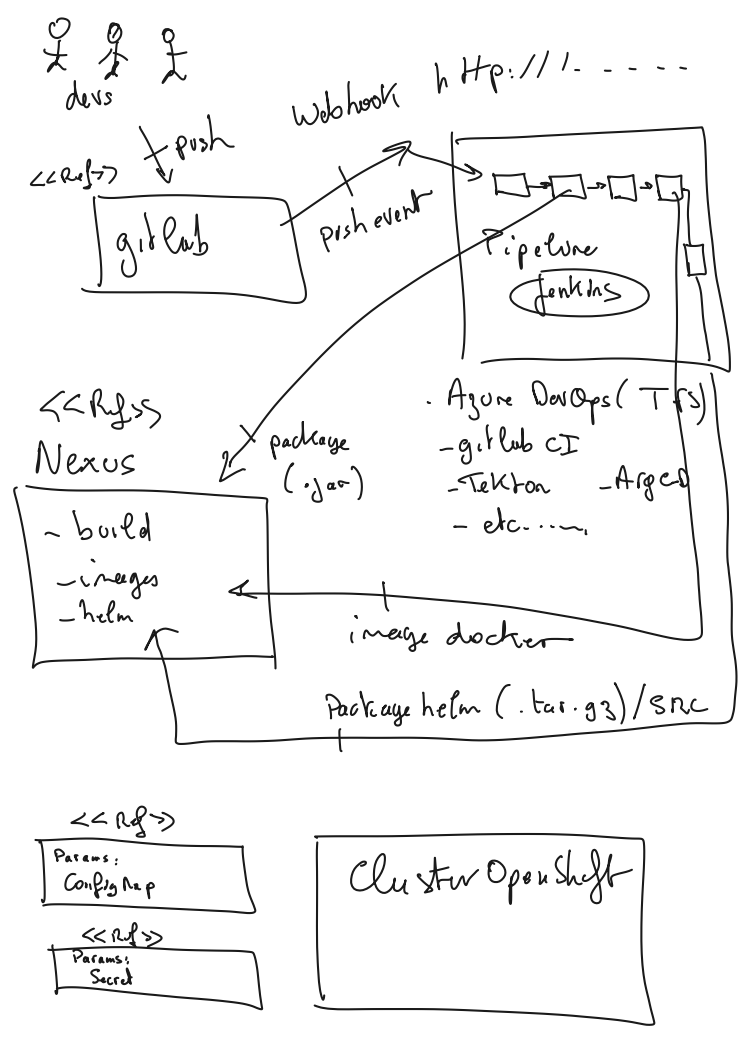
(2) Récupérer le yaml par rétro ingénierie

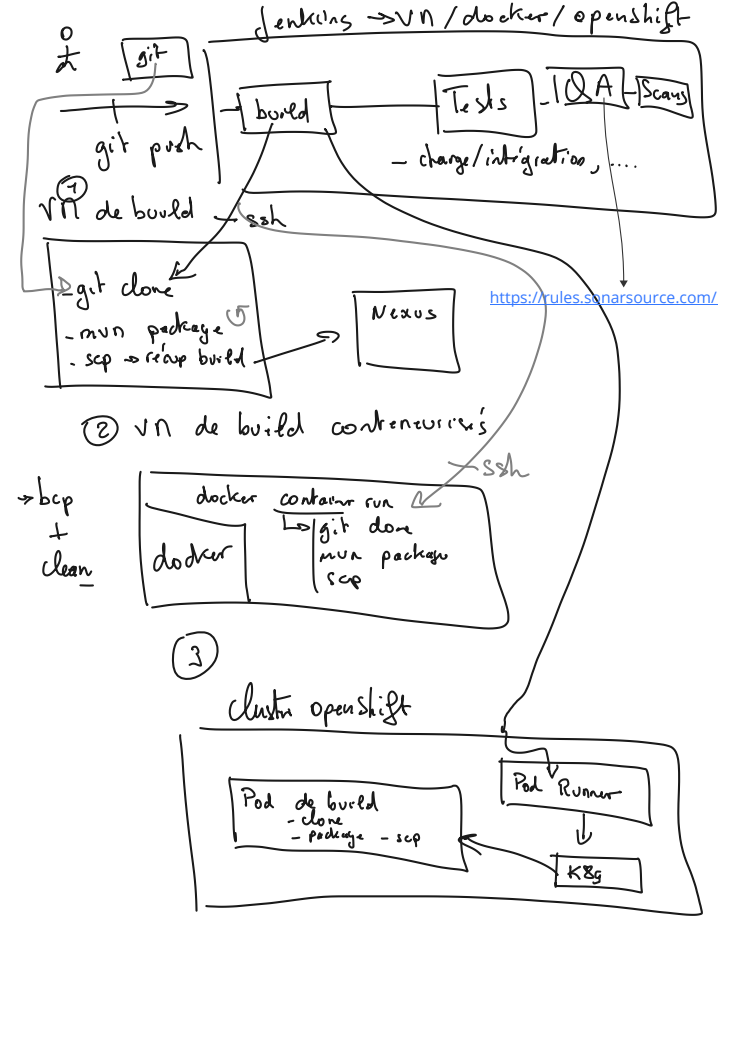
```
oc get deployment.apps/demospringboot -o yaml
oc delete deployment.apps/demospringboot
oc apply -f .....yaml
```

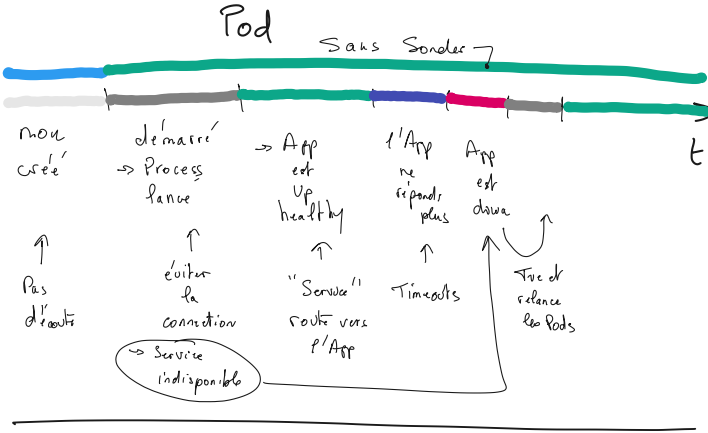
```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: demospringboot
  name: demospringboot
  namespace: st20
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: demospringboot
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: demospringboot
    spec:
      containers:
      - image: nboost/demospringboot
        imagePullPolicy: Always
        name: demospringboot
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
```

```
118  oc expose svc/demospringboot
119  oc status
120  oc new-project st20-01
121  oc status
122  history
123  sudo docker image ls
124  oc status
125  oc project st20
126  oc status
127  mkdir nginx
128  cd nginx/
129  code dep.yaml
130  oc apply -f dep.yaml
131  oc delete -f dep.yaml
132  history
133  oc create deployment demospringboot --image=nboost/demospringboot
134  oc get all
135  oc get deployment.apps/demospringboot
136  oc get deployment.apps/demospringboot -o wide
137  oc get deployment.apps/demospringboot -o yaml
138  oc get deployment.apps/demospringboot -o yaml > demosb.yaml
139  oc delete deployment.apps/demospringboot
140  code demosb.yaml
141  oc apply -f demosb.yaml
142  oc status
143  oc get all
144  export EDITOR=nano
145  oc edit deployment.apps/demospringboot
146  oc status
147  oc expose --help
148  oc expose deployment/demospringboot --port=8888
149  oc expose svc/demospringboot
150  oc get all
151  oc get service/demospringboot -o yaml > svcsb.yaml
152  oc get route/demospringboot -o yaml > rtsb.yaml
153  code .
154  oc delete route/demospringboot
155  oc delete svc/demospringboot
156  oc apply -f svcsb.yaml
157  oc get all
158  oc apply -f rtsb.yaml
159  oc delete -f myspringboot.yaml
160  oc apply -f myspringboot.yaml
161  history
```

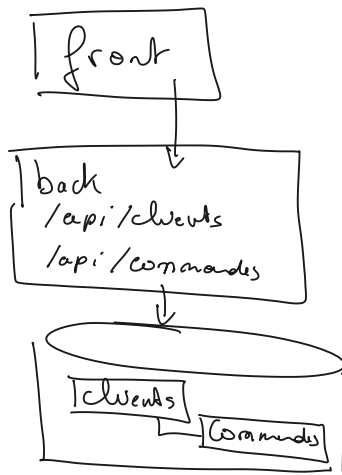
```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: demospringboot
  name: demospringboot
  namespace: st20
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: demospringboot
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: demospringboot
    spec:
      containers:
      - image: nboost/demospringboot
        imagePullPolicy: Always
        name: demospringboot
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
---
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  labels:
    app: demospringboot
  name: demospringboot
  namespace: st20
spec:
  port:
    targetPort: 8888
  to:
    kind: Service
    name: demospringboot
    weight: 100
  wildcardPolicy: None
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: demospringboot
  name: demospringboot
  namespace: st20
spec:
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - port: 8888
    protocol: TCP
    targetPort: 8888
  selector:
    app: demospringboot
  sessionAffinity: None
  type: ClusterIP
```


devs
push
<<Ref>>
gitlab
Webhook
http://.....
push event
Pipeline
Jenkins
<<Ref>>
Nexus
package
(.jar)
- Azure DevOps (TFS)
-gitlab CI
-Tekton
-ArgoCD
- etc.....
- build
-images
-helm
image docker
Package helm (.tar.gz)/SRC
<<Ref>>
Params:
ConfigMap
<<Ref>>
Params:
Secret
Cluster OpenShift

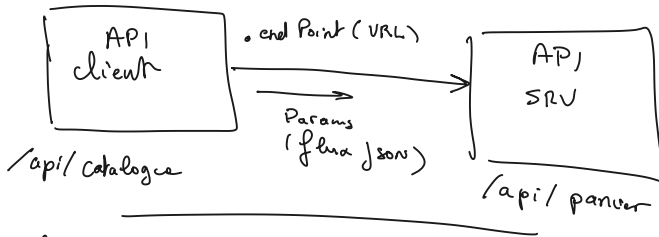

jenkins -> VM / docker / openshift
git
git push
build
Tests
IQA
Scays
- charge / intégration, ...
1 VM de build
ssh
git clone
- mvn package
- scp -> récup build
Nexus
https://rules.sonarsource.com/
2 VM de build containerisés
ssh
-> bcp + clean
docker container run
docker
git clone
mvn package
scp
3
cluster openshift
Pod de build
- clone
- package - scp
Pod Runner
K8s



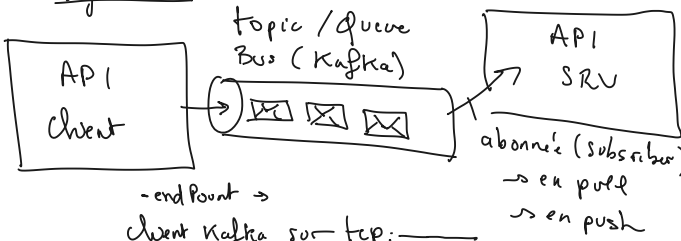
Services / Archis n-Tier



Synchrone



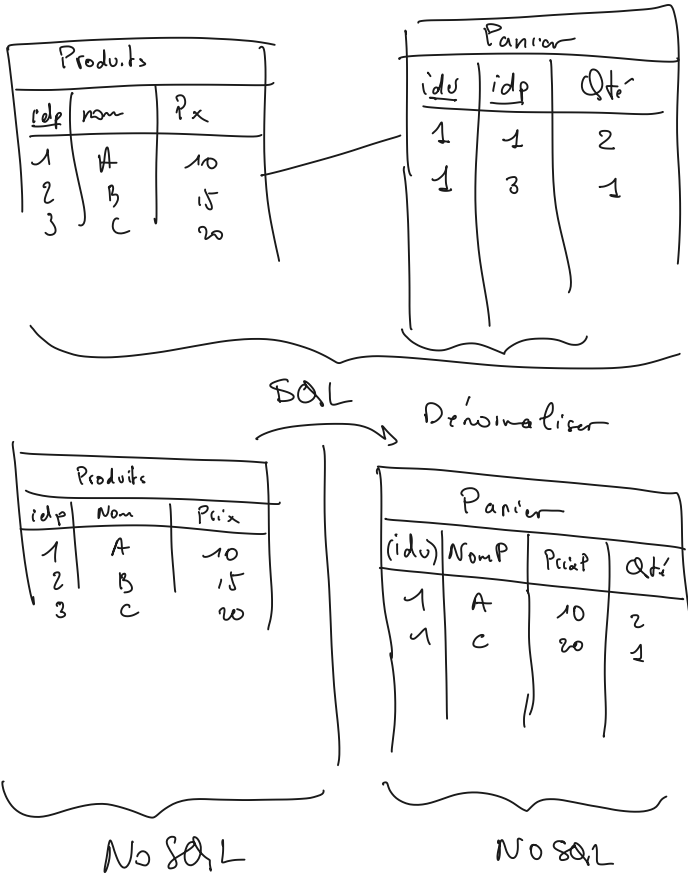
Asynchrone



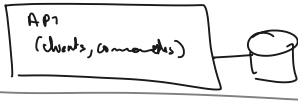
Pull : on scrute les nouveaux messages

Push : on reçoit les messages.

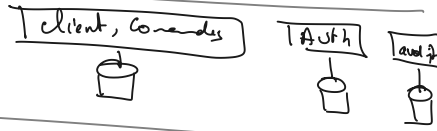
Kafka = pull, mais push ok en développement


Produits
idp
nom
Px
1
A
10
2
B
15
3
C
20
Panier
idu
idp
Qté
1
1
2
1
3
1
SQL
Dénormaliser
Produits
idp
Nom
Prix
1
A
10
2
B
15
3
C
20
Panier
(idu)
NomP
PrixP
Qté
1
A
10
2
1
C
20
1
NoSQL
NoSQL

Service (Synch - non Hautement dispo
Techniquement Simple)



Micro Services
→ Synch / Asynch ?

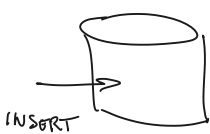


µ Service
→ Asynch !

nano Service

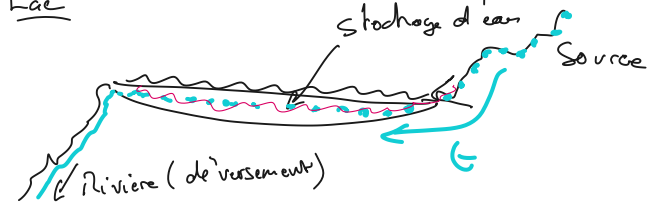
Pattern CQRS

https://learn.microsoft.com/en-us/azure/architecture/patterns/cqrs

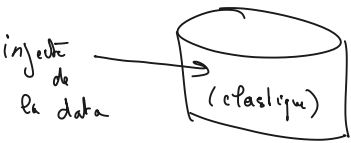


→ saturation du stockage -
→ pas de purge automatique

Lac



Datalake:



→ Règles de
→ Temporalité
→ Volume max

Volume → varie