Documentation en ligne :
https://www.ibm.com/docs/en/was-liberty
https://openliberty.io/docs/latest/overview.html

productInfo featureInfo -> liste des fonctions installées
featureManager install adminCenter-1.0 -> installer des fonctions

Installation de vscode :
sudo snap install code --classic

lancement :
code <dossier racine pour l'édition>

Utiliser des dossiers de bibliothèques partagées :
https://www.ibm.com/docs/en/was-liberty/core?topic=applications-shared-libraries

Soit utiliser shared :
/usr/shared : y positionner les .jar

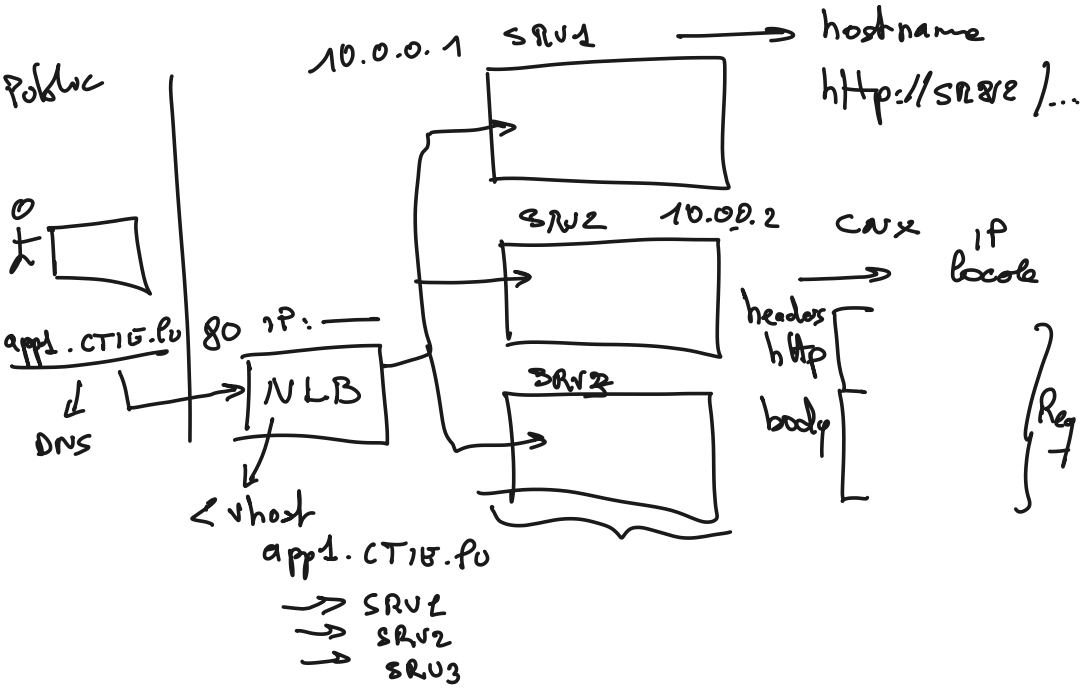
Ou spécifier dans le server.xml :

```
<server>
   <!-- Other server configuration -->

   <library id="sharedLibrary">
      <fileset dir="${shared.resource.dir}/mySharedLib" includes="*.jar"/>
   </library>

   <application location="myApp.war">
      <classloader commonLibraryRef="sharedLibrary"/>
   </application>
</server>
```

You can place global libraries in two locations:

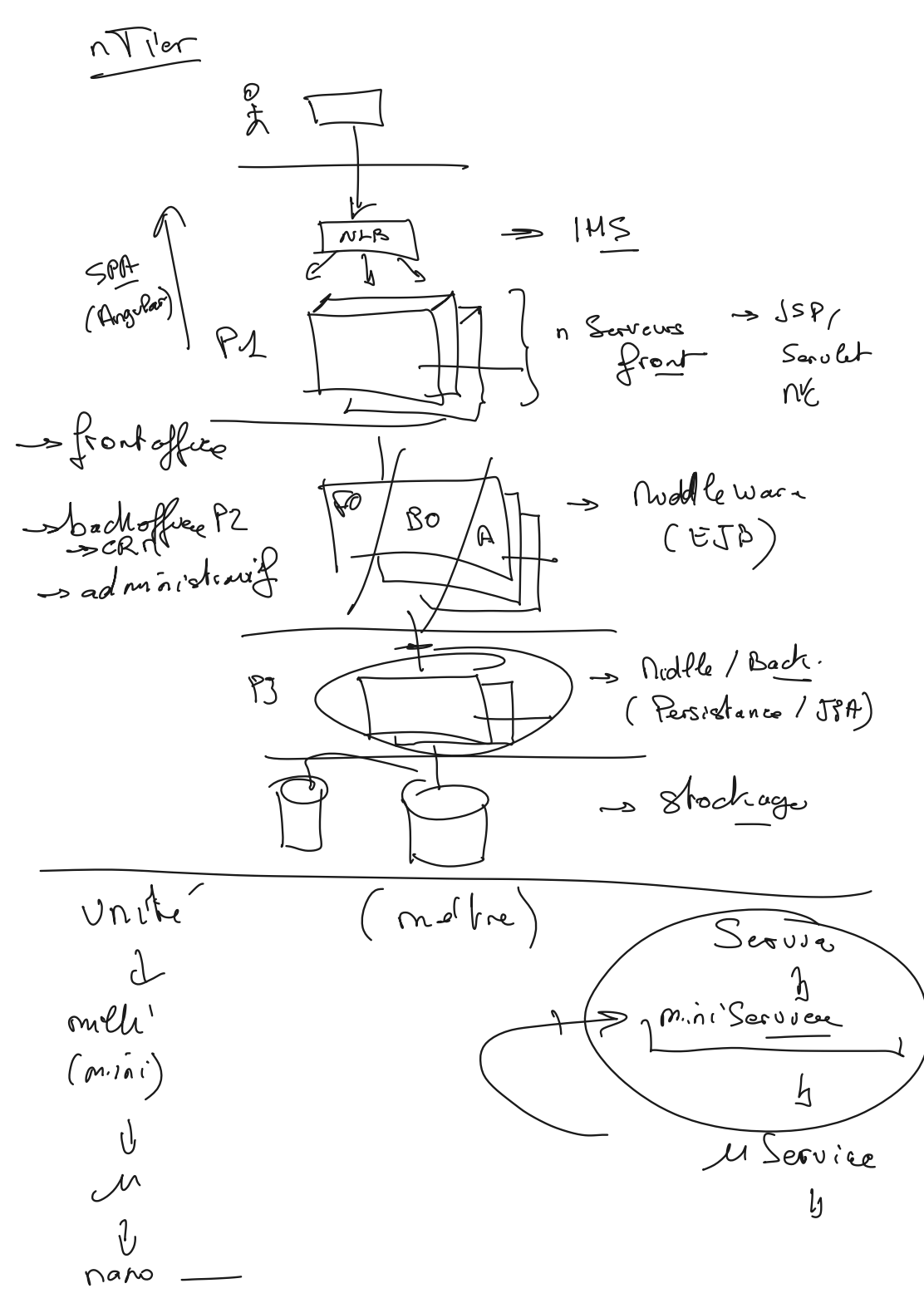
- ${shared.config.dir}/lib/global
- ${server.config.dir}/lib/global

If there are files present in these locations at the time an application is started, and that application does not have a <classloader> element configured, the application uses these libraries. If a class loader configuration is present, **these libraries are not used** unless the global library is explicitly referenced.



Activer la console osgi : https://www.ibm.com/docs/en/was-liberty/core?topic=line-using-osgi-console
La référence vers le Gogo shell : https://learn.liferay.com/w/dxp/liferay-internals/fundamentals/using-the-gogo-shell/gogo-shell-commands

Aller plus loin avec Git : https://learngitbranching.js.org/?locale=fr_FR


nTier
NLB
IHS
SPA
(Angular)
P1
n Services
front
JSP/
Servlet
MVC
front office
back office P2
CRM
administratif
BO
A
Middleware
(EJB)
P3
Middle / Back.
(Persistance / JPA)
Stockage
Unité
(moins)
multi
(mini)
nano
Service
mini Services
Service

Archi nTier → Simples

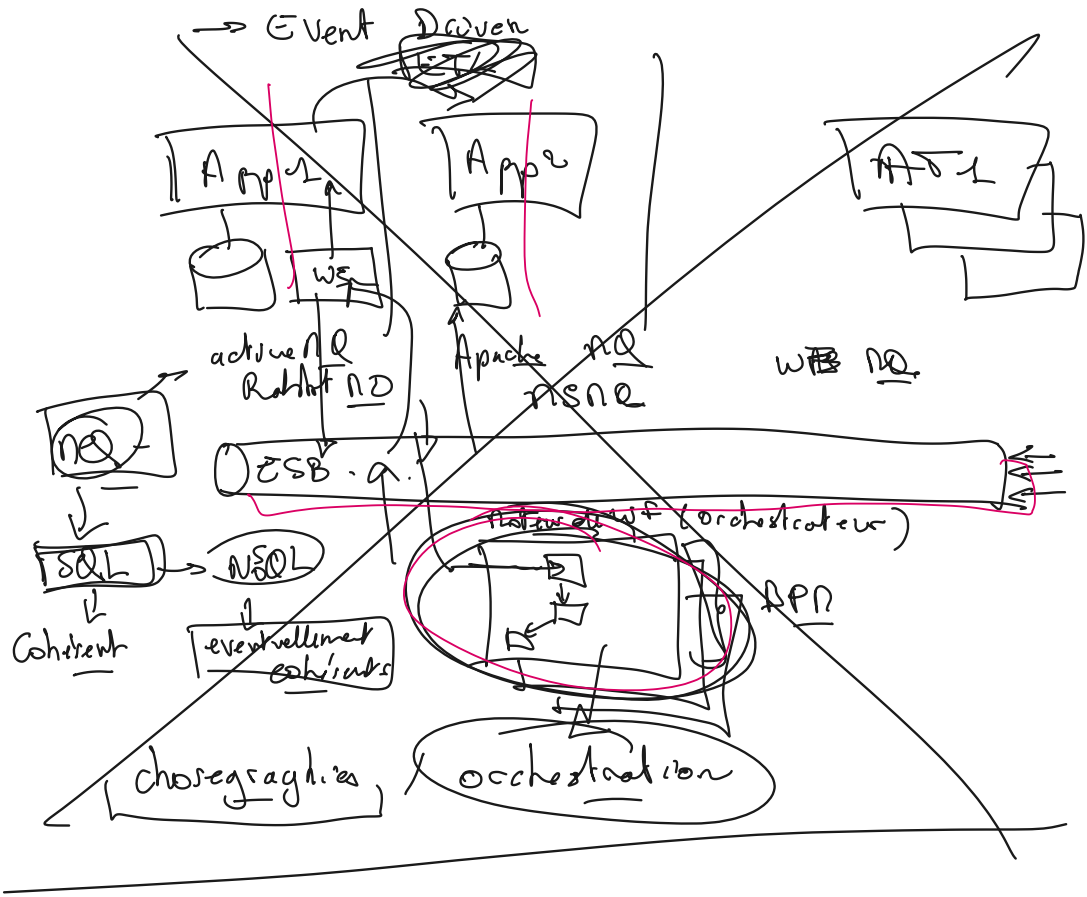
SOA / ESB
EAI / BPM.

≃ 2000 → Complexes

Acche

Hautement Disponible

→ Event Driven



archi en µ Services

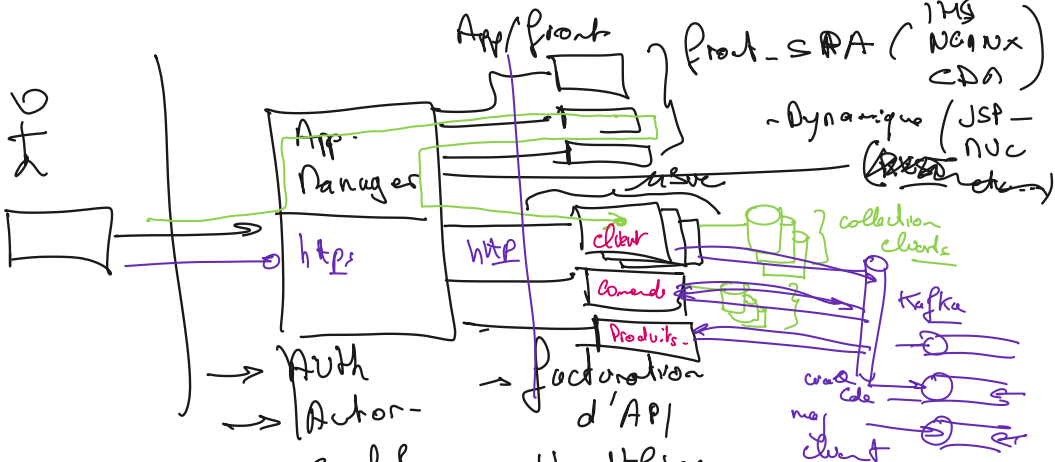
n 9th

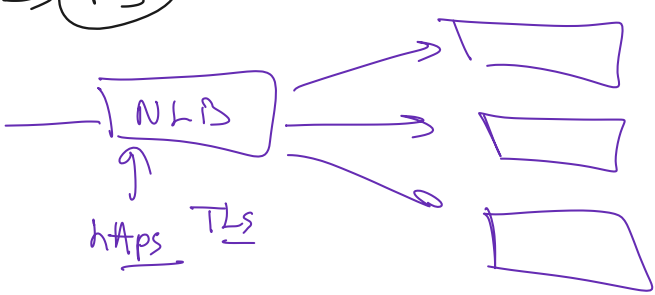
5.9th-

30 min → 99.999%

99.99999999

≈ 1,5s/mois





Implémenter une architecture collective :

srv1 : IBM IHS
srv2 : Collective controller
srv3 & srv4 : Collective members

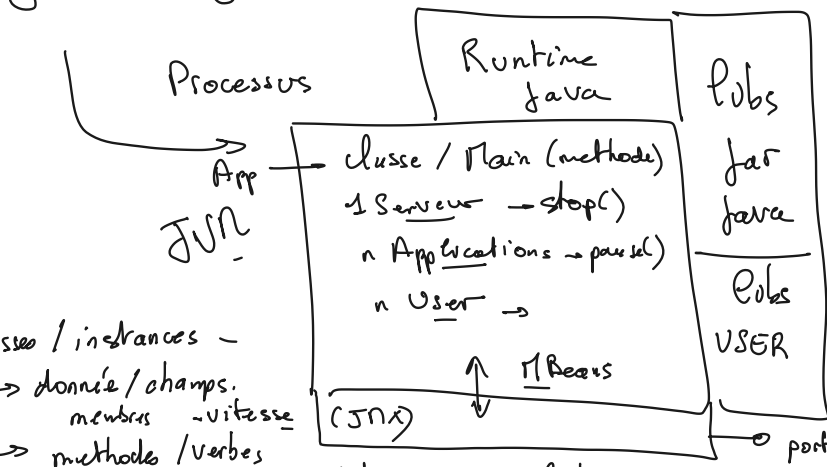
objectif : arrêter un membre quand tout est up, l'app doit continuer de s'afficher.

Basé sur : https://developer.ibm.com/tutorials/simple-load-balancing-websphere-liberty-apps/
App à déployer : .war de ce matin.

Installer l'installer pour le scanner d'App :
https://www14.software.ibm.com/cgi-bin/weblap/lap.pl?popup=Y&la_formnum=&li_formnum=L-SSQH-NHSBLH&title=Software%20License%20Agreement&accepted_url=https://public.dhe.ibm.com/ibmdl/export/pub/software/websphere/wasdev/downloads/wamt/ApplicationBinaryTP/binaryAppScannerInstaller.jar&declined_url=https://www.ibm.com/support/pages/node/6250913

L'iinstaller : java -jar binaryAppScannerInstaller.jar

java -jar ——— .jar



classes / instances —
→ données / champs. membres - vitesse
→ methodes / verbes
accelerer()
freiner()
getVitesse() → lecture (accesseurs)
setEtat("ok") — (getters)

Java Management eXtensions (JMX)

Managed Beans → exporté via JMX.

clients JMX

- Java → jconsole (swing)
  → host / port (user/mdp)
- bin/server start → lance Java
  " " stop → cnx JMX vers le serveur
- (Java)
  - groovy
  - jython
  → interface JMX → écrire de scripts
  → objets "JMX" génériques.

- utiliser la lib "rest Connector" de Liberty.
  → objets qui représentent les objets réels

Scripting jython sur Liberty :

- installer jython (sudo apt install jython)
- copier restConnector.py (dossier client de Liberty) dans le dossier Lib de jython :
  - cp clients/jython/restConnector.py /usr/share/jython/Lib/
- spécifier le JYTHONPATH vers le .jar :
  - export JYTHONPATH=$JYTHONPATH:/opt/install/dev-wlp14/clients/restConnector.jar
- Créer un fichier .jy avec notre script de base Jython (récupérer les bonnes clefs et référence vers le fichier keystore) :
  (voir : https://www.ibm.com/docs/en/was-liberty/base?topic=manually-establishing-jmx-mbean-liberty-server-connection)

```
from restConnector import JMXRESTConnector
JMXRESTConnector.trustStore = "~~/opt/install/dev-wlp14/usr/servers/master/resources/security/key.jks~~"
JMXRESTConnector.trustStorePassword = "~~fYnrUGlVKjl1JwS8qx5Fepy~~"

connector = JMXRESTConnector()
connector.connect("master",9443,"~~admin~~","~~adminpwd~~")
mconnection = connector.getMBeanServerConnection()
# mconnection.invoke(...)
connector.disconnect()
```

- Se connecter à un objet
  - notifier1=javax.management.ObjectName("web:name=Notifier1")
- Bibliothèque à voir : JMXQuery :
  - https://github.com/dgildeh/JMXQuery/tree/master/python
- Outil célèbre pour simplifier les appels via REST :
  - https://jolokia.org/

Exercice ansible :
Mettez à jour les 4 VMs (cache de package, et packages installés) - **en une seule opération**

https://docs.ansible.com/ansible/latest/collections/ansible/builtin/apt_module.html#ansible-collections-ansible-builtin-apt-module