## Documentation pour les configurations requises

### Question 2 :

Allez dans le dossier contenant le code source de weka REST :

```
cd <path>/jguwekarest
```

Compiler le fichier war avec maven :

```
mvn clean package
```

Pour build l'image de weka il faut exécuter la commande suivante (il faut remplacer `dockerhubuserID` avec votre ID de docker hub) :

```
docker build -t dockerhubuserID/jguweka:OAS3 .
```

**Optionnel :** Pour consulter la liste des images docker disponibles, il suffit d'exécuter la commande suivante :

```
docker images
```

Pour lancer l'image localement, il faut exécuter les commandes suivantes :

```
docker pull mongo; docker run --name mongodb -d mongo
```

```
docker run -p 8080:8080 --link mongodb:mongodb dockerhubuserID/jguweka:OAS3
```

Enfin, pour accéder à l'interface de weka, il faut consulter le : **`localhost:8080`**

---

**Démonstration :** Voir la vidéo de la question 6.

### Question 3 :

Il faut commencer par changer le Dockerfile en y ajoutant les deux instructions suivantes :

```
RUN wget http://download-keycdn.ej-technologies.com/jprofiler/jprofiler_linux_11_0.tar.gz
-P /tmp/ &&\

 tar -xzf /tmp/jprofiler_linux_11_0.tar.gz -C /usr/local &&\

 rm /tmp/jprofiler_linux_11_0.tar.gz

ENV JPAGENT_PATH=
"-agentpath:/usr/local/jprofiler11.0/bin/linux-x64/libjprofilerti.so=nowait"
```

Ensuite, nous allons mapper le port 8849 au host port 8849 en ajoutant : « `- "8849:8849"` » dans « `ports` » (dans le `docker-compose.yml`) (Dans le cas où on utilise docker-compose)

Et on ajoute l'instruction suivante dans le Dockerfile :

```
EXPOSE 8849
```

À ce moment, il faut rebuild l'image avec l'instruction :

```
docker build -t dockerhubuser/jguweka:OAS3 .
```

Et par la suite, la lancer avec :

```
docker run -p 8080:8080 --link mongodb:mongodb hibaa/jguweka:OAS3
```

Maintenant, on doit accéder au conteneur docker avec :

```
docker exec -it [container-name] bash
```

Pour trouver le nom du conteneur, il suffit de taper : `docker ps` et regarder le nom du conteneur.

Ensuite, nous allons démarrer le « ***attach mode*** » de JProfiler dans le conteneur Docker en exécutant les commandes suivantes :

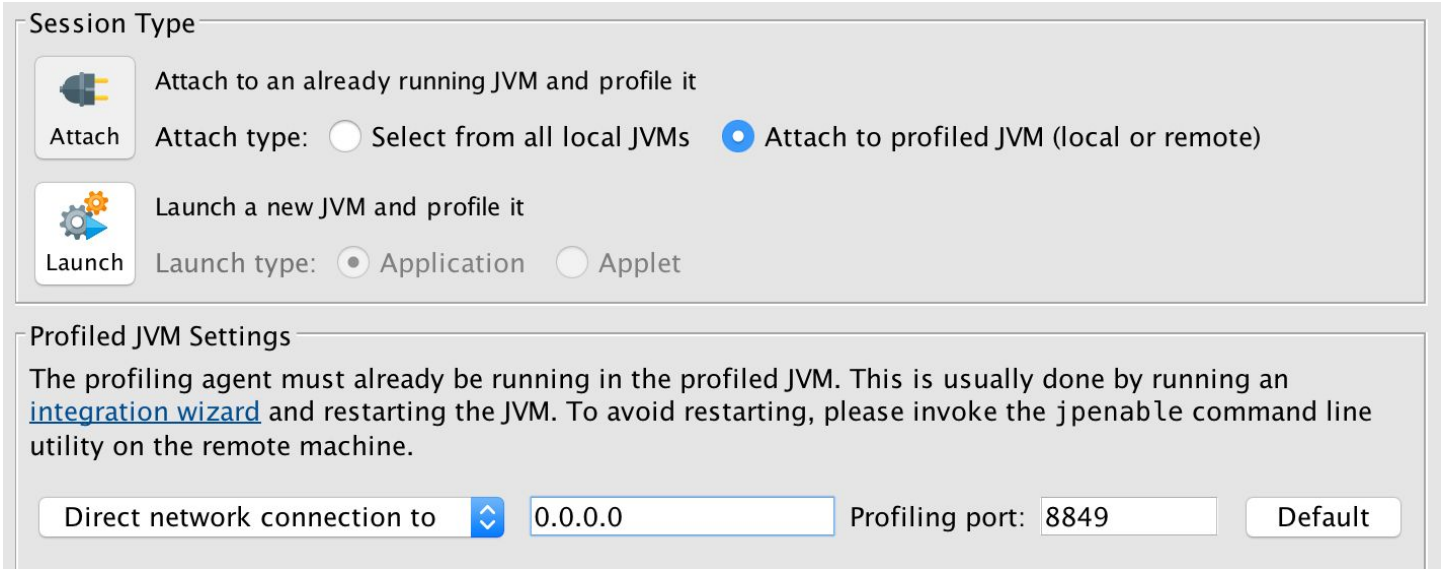
```
cd /usr/local/jprofiler11.0/
```

```
bin/jpenable
```

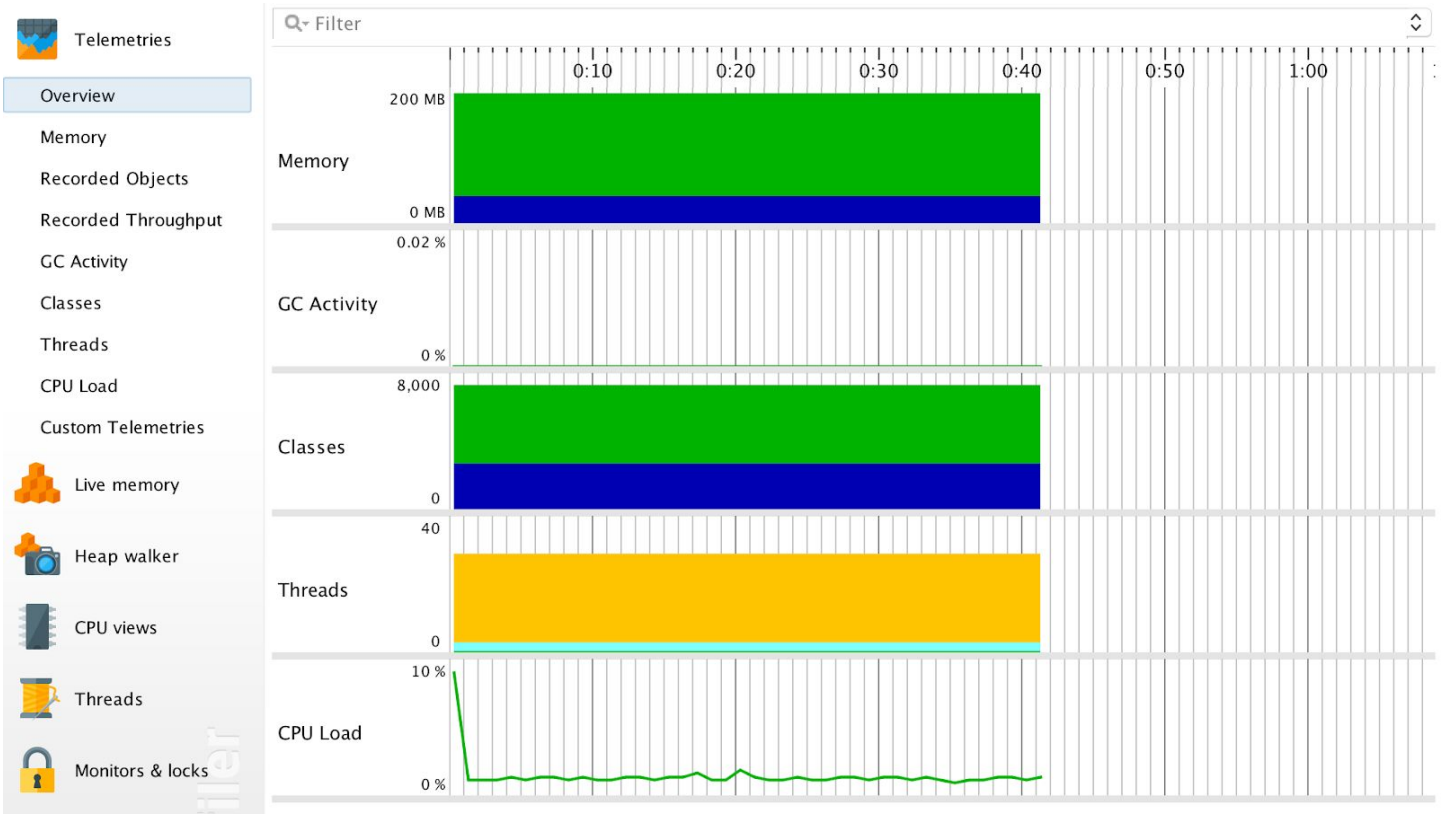
JProfiler nous demandera d'entrer le mode et le port. Nous allons entrer **1** et **8849**.

Maintenant, on peut utiliser le JProfiler GUI pour se connecter au port 8849. On doit ouvrir une nouvelle session dans le GUI de JProfiler (**Session -> New Session** ou **« cmd » + N**).

Il faut choisir **Attach** puis **Attach to profiled JVM** et par la suite spécifier l'adresse **IP** et **8849** comme profiling port dans la section Profiled JVM settings section. Enfin, cliquez sur **Ok**.



Une fois le processus de connexion est terminé, les chartes de profiling seront affichées.

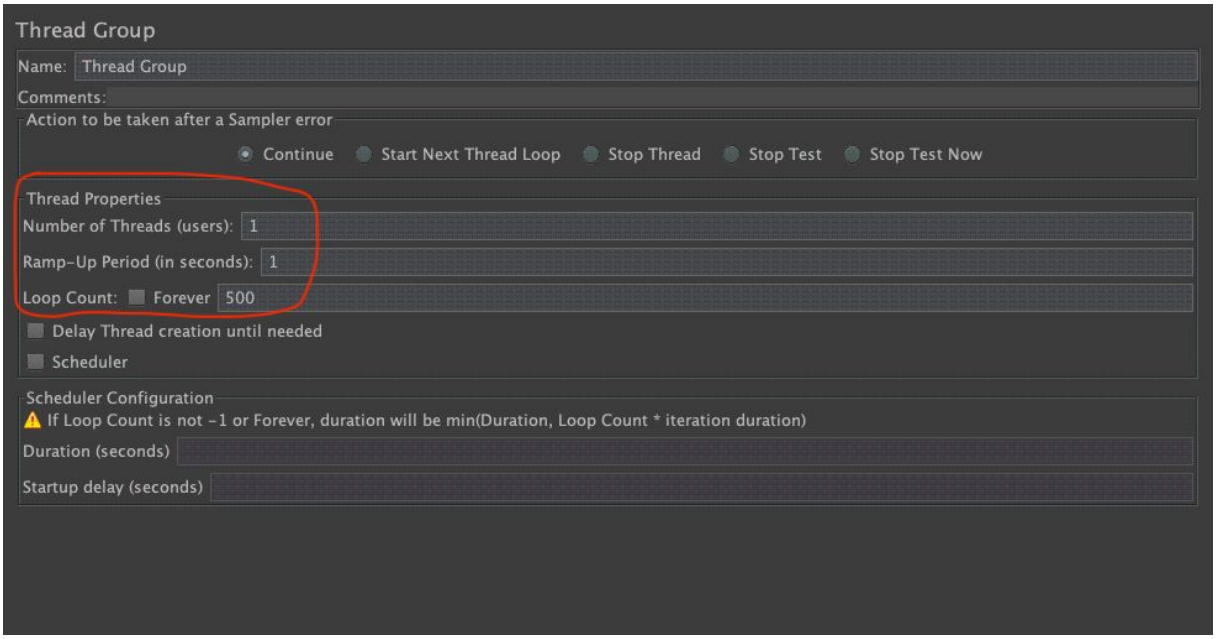


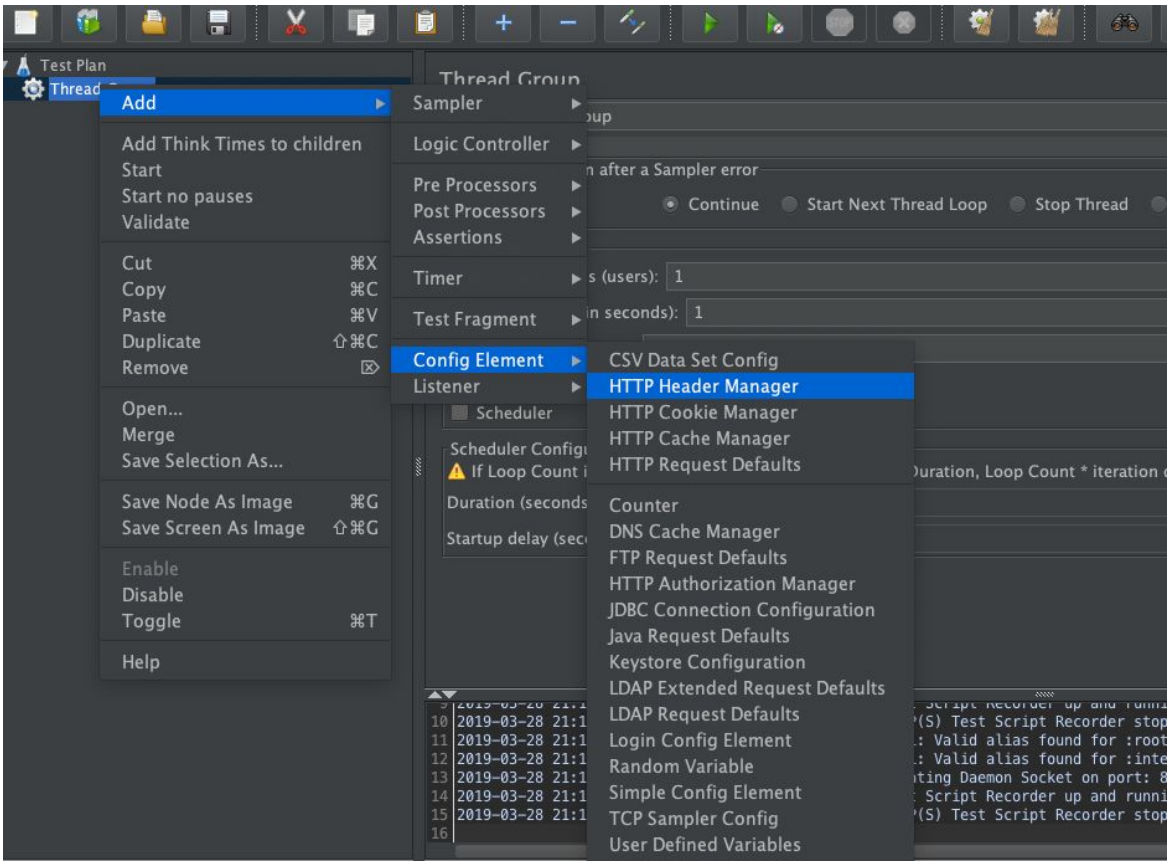

**Question4 :**

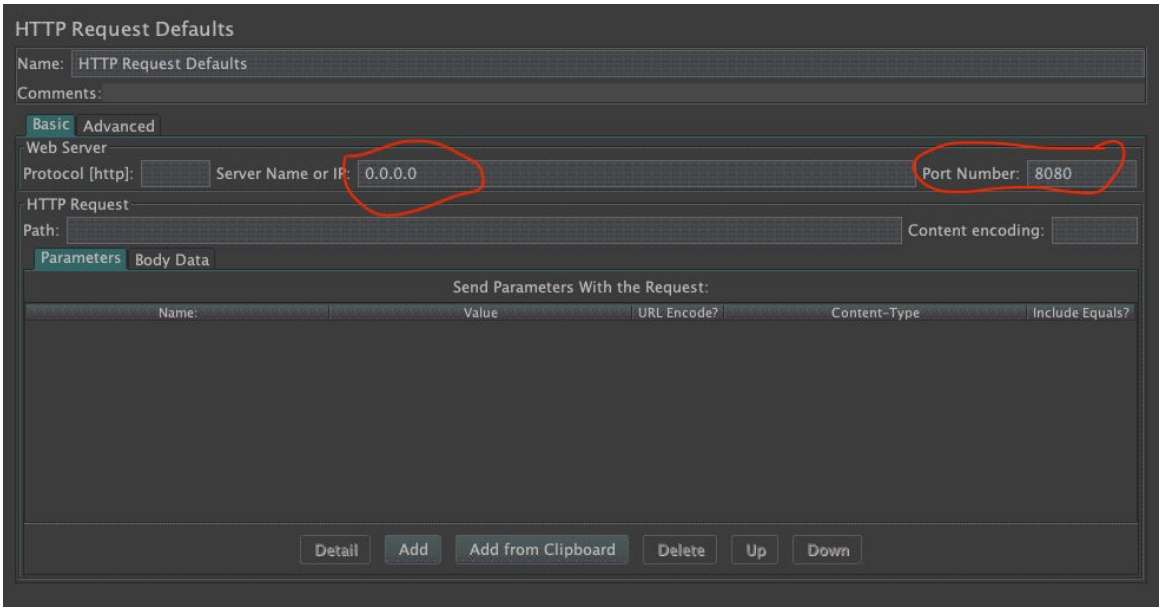
1. Télécharger JMeter à partir de ce site: http://jmeter.apache.org/download_jmeter.cgi
2. Extraire les fichiers
3. Lancer JMeter GUI et exécuter cette commande :

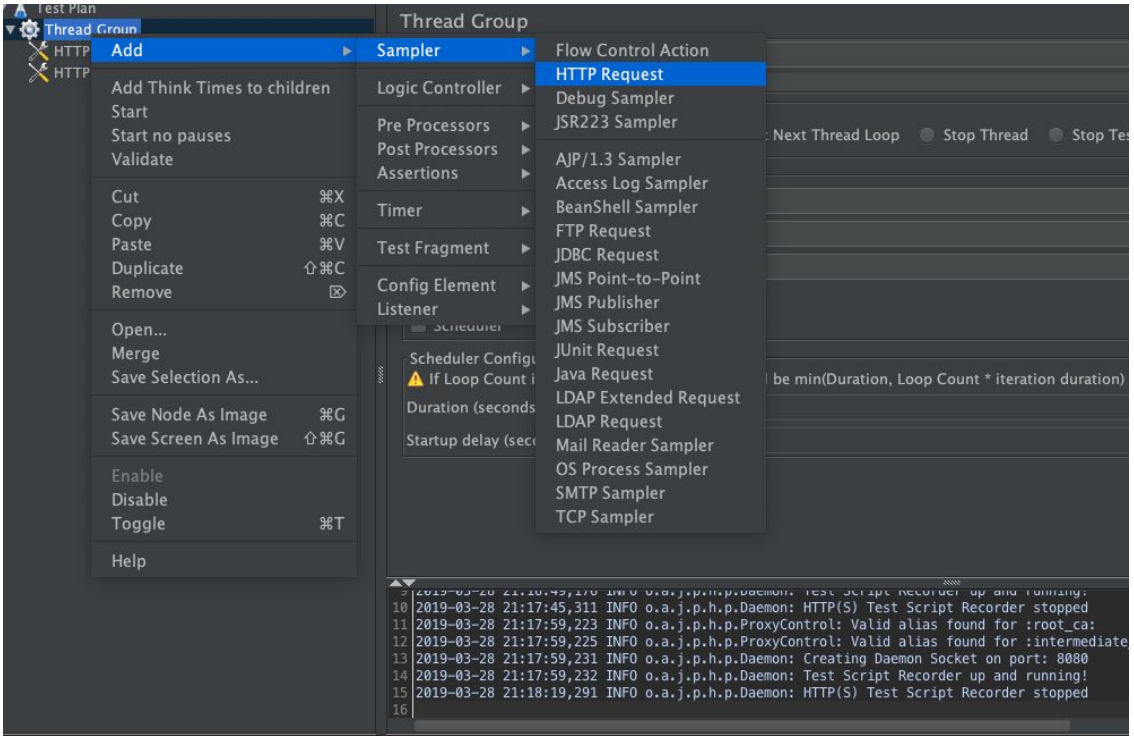
```
cd apache-jmeter-5.1.1/bin/./jmeter
```

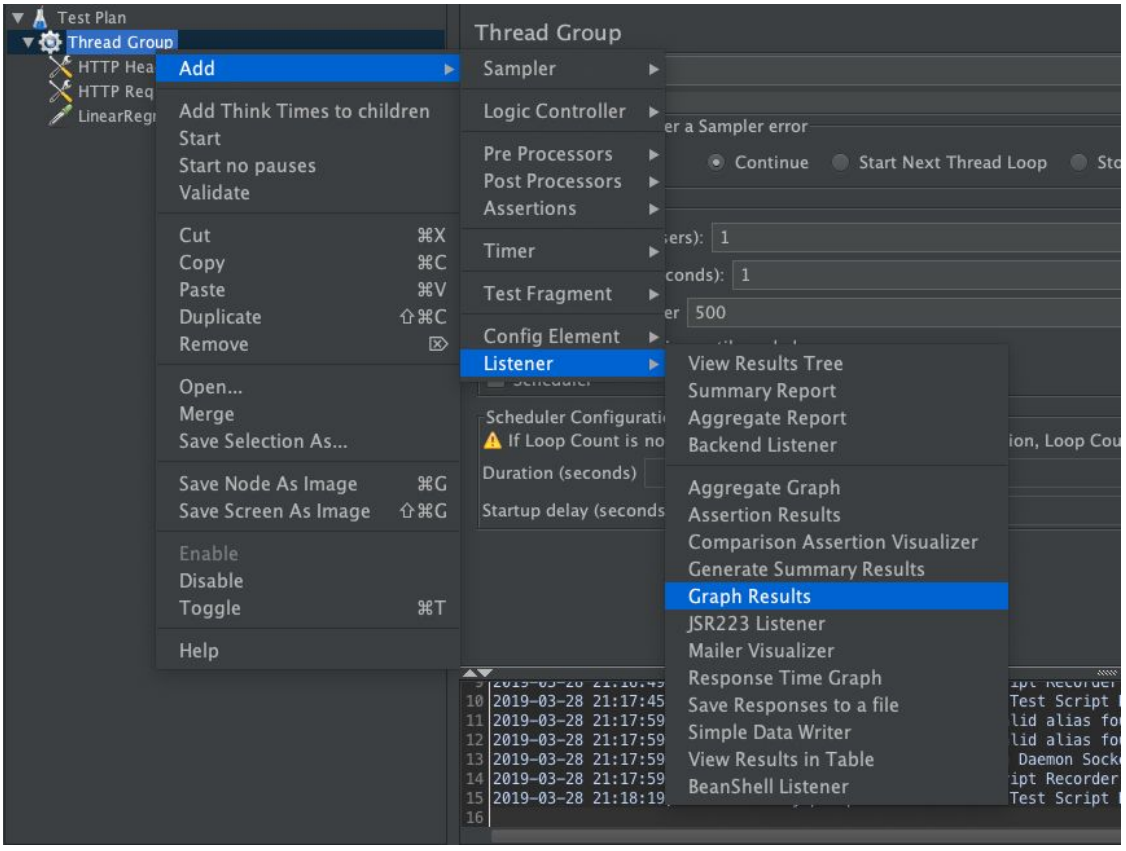
4. Construire un plan de test en suivant les étapes suivantes : http://jmeter.apache.org/usermanual/build-test-plan.html


Thread Group
Name: Thread Group
Comments:
Action to be taken after a Sampler error
Continue
Start Next Thread Loop
Stop Thread
Stop Test
Stop Test Now
Thread Properties
Number of Threads (users): 1
Ramp-Up Period (in seconds): 1
Loop Count: Forever 500
Delay Thread creation until needed
Scheduler
Scheduler Configuration
If Loop Count is not -1 or Forever, duration will be min(Duration, Loop Count * iteration duration)
Duration (seconds)
Startup delay (seconds)


Test Plan
Add
Add Think Times to children
Start
Start no pauses
Validate
Cut ⌘X
Copy ⌘C
Paste ⌘V
Duplicate ⇧⌘C
Remove
Open...
Merge
Save Selection As...
Save Node As Image ⌘G
Save Screen As Image ⇧⌘G
Enable
Disable
Toggle ⌘T
Help
Sampler
Logic Controller
Pre Processors
Post Processors
Assertions
Timer
Test Fragment
Config Element
Listener
CSV Data Set Config
HTTP Header Manager
HTTP Cookie Manager
HTTP Cache Manager
HTTP Request Defaults
Counter
DNS Cache Manager
FTP Request Defaults
HTTP Authorization Manager
JDBC Connection Configuration
Java Request Defaults
Keystore Configuration
LDAP Extended Request Defaults
LDAP Request Defaults
Login Config Element
Random Variable
Simple Config Element
TCP Sampler Config
User Defined Variables


HTTP Request Defaults
Name: HTTP Request Defaults
Comments:
Basic
Advanced
Web Server
Protocol [http]:
Server Name or IP:
0.0.0.0
Port Number:
8080
HTTP Request
Path:
Content encoding:
Parameters
Body Data
Send Parameters With the Request:
Name:
Value
URL Encode?
Content-Type
Include Equals?
Detail
Add
Add from Clipboard
Delete
Up
Down


Test Plan
Thread Group
HTTP
HTTP
Thread Group
Add
Add Think Times to children
Start
Start no pauses
Validate
Cut ⌘X
Copy ⌘C
Paste ⌘V
Duplicate ⇧⌘C
Remove ⌦
Open...
Merge
Save Selection As...
Save Node As Image ⌘G
Save Screen As Image ⇧⌘G
Enable
Disable
Toggle ⌘T
Help
Sampler
Logic Controller
Pre Processors
Post Processors
Assertions
Timer
Test Fragment
Config Element
Listener
Flow Control Action
HTTP Request
Debug Sampler
JSR223 Sampler
AJP/1.3 Sampler
Access Log Sampler
BeanShell Sampler
FTP Request
JDBC Request
JMS Point-to-Point
JMS Publisher
JMS Subscriber
JUnit Request
Java Request
LDAP Extended Request
LDAP Request
Mail Reader Sampler
OS Process Sampler
SMTP Sampler
TCP Sampler
Next Thread Loop
Stop Thread
Scheduler Configu
If Loop Count i
be min(Duration, Loop Count * iteration duration)
Duration (seconds
Startup delay (sec
2019-03-28 21:17:45,311 INFO o.a.j.p.h.p.Daemon: HTTP(S) Test Script Recorder stopped
2019-03-28 21:17:59,223 INFO o.a.j.p.h.p.ProxyControl: Valid alias found for :root_ca:
2019-03-28 21:17:59,225 INFO o.a.j.p.h.p.ProxyControl: Valid alias found for :intermediate
2019-03-28 21:17:59,231 INFO o.a.j.p.h.p.Daemon: Creating Daemon Socket on port: 8080
2019-03-28 21:17:59,232 INFO o.a.j.p.h.p.Daemon: Test Script Recorder up and running!
2019-03-28 21:18:19,291 INFO o.a.j.p.h.p.Daemon: HTTP(S) Test Script Recorder stopped




5. Configurer votre navigateur si vous utiliser JMeter Proxy.

6. Une fois que toutes les étapes de la création du plan de test sont faites, exécuter la commande :

```
docker run -p 8080:8080 --link mongodb:mongodbuserID/jguweka:OAS3
```

Puis ouvrir votre navigateur ex: Google Chrome

1. Aller au localhost:8080
2. Exécuter la requête que vous voulez tester.
3. Revenez à JMeter et cliquer sur le bouton Start du ‘listener’ choisi.

**Question 5 :**

Puisque cette question ne peut pas être réalisé (à moins qu’on utilise Kubernetes, ce qui n’est pas possible dans notre cas puisqu’il faut avoir un cluster existant. Une autre option serait d’implémenter notre propre solution en Golang par exemple, la publier sur docker hub et utiliser cette image par la suite), nous l’avons réalisée d’une autre manière.

Il faut commencer par changer le fichier « docker-compose.yml », en ajoutant un répartiteur de charge « ***load balancer*** ». Pour ce faire, nous avons décidé d’utiliser HAProxy (vu et utilisé dans le cadre d’un laboratoire du cours LOG3000). Les modifications apportées au fichier sont les suivantes :

```yaml
version: '3'
services:
  mongo:
    image: mongo
    restart: always
  jguweka:
    image: jguweka/jguweka:OAS3
    restart: always
    links:
      - "mongo:mongodb"
    depends_on:
      - mongo
    labels:
      - "service-name:jguweka"
      - "service-type:app-srv"
      - "environment:test"
  loadb:
    image: dockercloud/haproxy
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"
    ports:
      - "8082:80"
    links:
      - jguweka
```

Ensuite, il faut exécuter la commande :

**`docker-compose up --scale jguweka=5`** (Ici, nous avons mis 5 conteneurs, mais on peut spécifier le nombre souhaité)

Pour vérifier le nombre de conteneurs déployés, il faut exécuter :

**`docker ps`**

Dans la figure suivante, nous pouvons voir les 5 conteneurs créés :

```
service-WIFI-EDUROAM-LA-8-1:jguwekarest-master hibabagane$ docker ps
CONTAINER ID        IMAGE                  COMMAND                  CREATED             STATUS              PORTS                                  NAMES
a9374853931f        dockercloud/haproxy    "/sbin/tini -- docke…"   About a minute ago  Up About a minute   443/tcp, 1936/tcp, 0.0.0.0:8082->80/tcp   jguwekarest-master_loadb_1
e23a77f51cd2        jguweka/jguweka:OAS3   "catalina.sh run"        About a minute ago  Up About a minute   8080/tcp                               jguwekarest-master_jguweka_1
101e5ebbfd7e        jguweka/jguweka:OAS3   "catalina.sh run"        About a minute ago  Up About a minute   8080/tcp                               jguwekarest-master_jguweka_5
c1f10dfc0de0        jguweka/jguweka:OAS3   "catalina.sh run"        About a minute ago  Up About a minute   8080/tcp                               jguwekarest-master_jguweka_2
96e6b6f759a3        jguweka/jguweka:OAS3   "catalina.sh run"        About a minute ago  Up About a minute   8080/tcp                               jguwekarest-master_jguweka_4
d03e4be0ae91        jguweka/jguweka:OAS3   "catalina.sh run"        About a minute ago  Up About a minute   8080/tcp                               jguwekarest-master_jguweka_3
4838871d78cf        mongo                  "docker-entrypoint.s…"   About a minute ago  Up About a minute   27017/tcp                              jguwekarest-master_mongo_1
be3a76d9c7fe        hibaa/jguweka:OAS3     "catalina.sh run"        23 hours ago        Up 23 hours         8080/tcp, 0.0.0.0:8849->8849/tcp       friendly_jackson
8acd30b5f84b        mongo                  "docker-entrypoint.s…"   8 days ago          Up 8 days           27017/tcp                              mongodb
cd1f0f0b1474        nginx                  "nginx -g 'daemon of…"   8 days ago          Up 8 days           0.0.0.0:80->80/tcp                     webserver
```

Pour accéder à l’interface de weka, il faut consulter : http://localhost:8082/