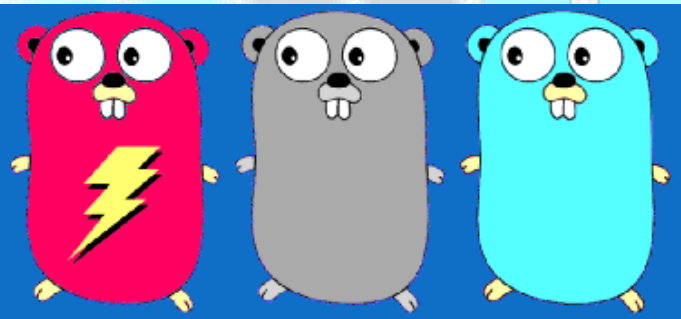
UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

université PARIS-SACLAY

# Génie Logiciel pour le Calcul Scientifique



#4

13/12/2024

jean-michel.batto@cea.fr

cea

https://gogs.eldarsoft.com/M2_IHPS

# Création des nœuds, connexion

INTERACTIF

- D'abord créer un swarm

```
docker swarm init
```

- Puis y attacher un réseau

```
docker network create --driver=overlay --attachable yml_mpinet
```

- Dans le répertoire du cours CM4 (ou CM3)

```
docker-compose up --scale mpihead=1 --scale mpinode=3 --scale grafana=1 -d
```

➔ utilise le fichier docker-compose.yml

➔ L'image docker n'est pas la même (jmbatto/**m2chps-mpi41-xmp**)

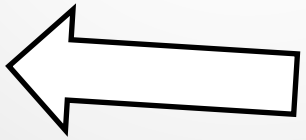
Se connecter à un shell docker (utiliser le bash) : (root car supervisor), ssh localhost:2022 → mpiuser, utiliser la clé ppk

# Que contient le docker-compose ?

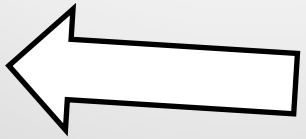
```
grafana:
    container_name: influxdb_local
    image: philhawthorne/docker-influxdb-grafana:latest
```

image disponible sur hub.docker.com

```
mpihead:
    image: jmbatto/m2chps-mpi41-xmp:latest
    shm_size: '512m'
        ports:
            - "2022:22"
```



on alloue 512mbyte pour l'espace IPC (communication inter processus, nécessaire pour MPI)

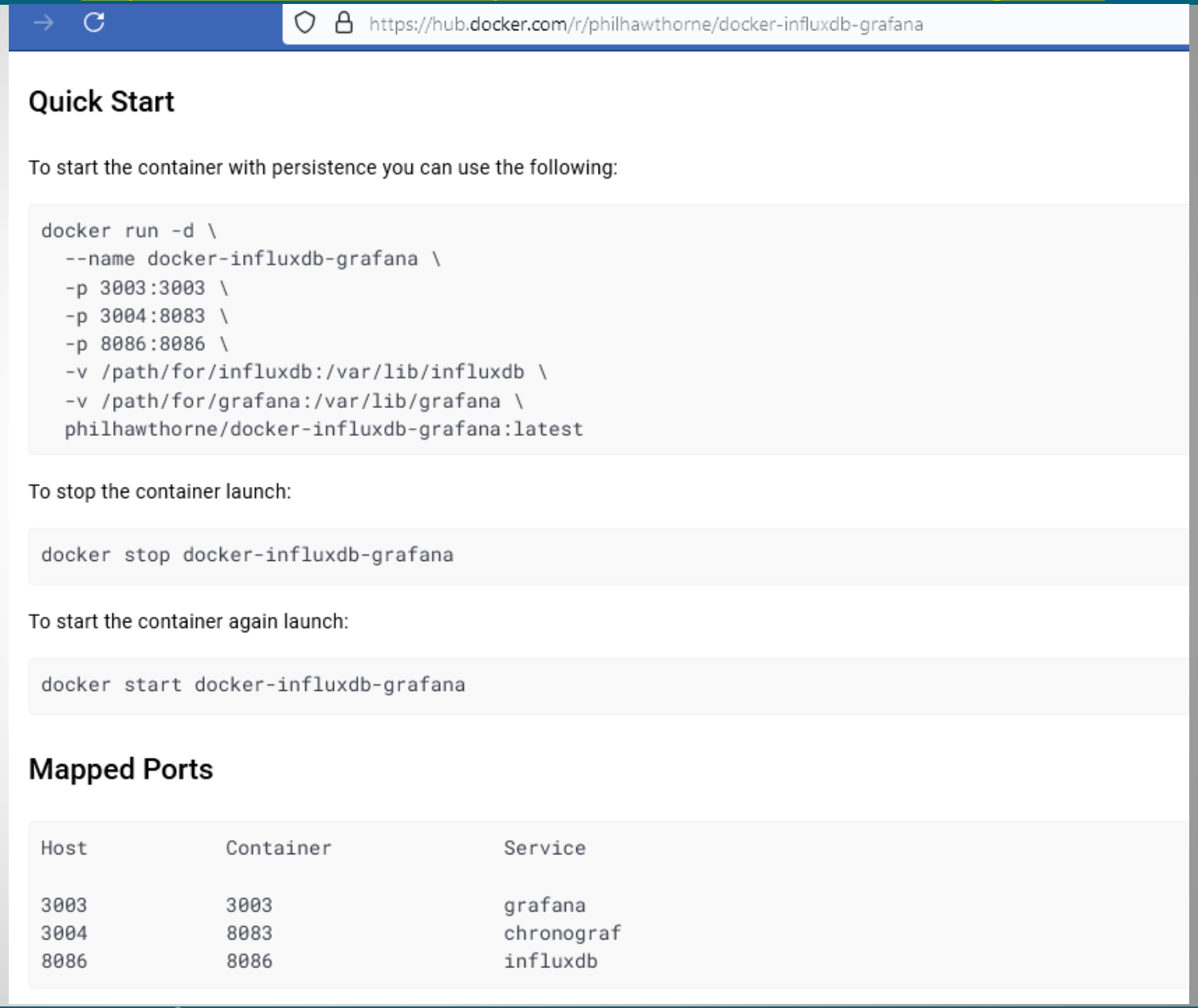


mapping du port SSH vers le port 2022

# Que contient l'image ?

https://hub.docker.com/r/philhawthorne/docker-influxdb-grafana

- Doc depuis le site hub.docker.com
- Image qui contient
  - grafana (:3003)
  - chronograf (:3004)
  - Influxdb (:8086)



https://hub.docker.com/r/philhawthorne/docker-influxdb-grafana

## Quick Start

To start the container with persistence you can use the following:

```
docker run -d \
  --name docker-influxdb-grafana \
  -p 3003:3003 \
  -p 3004:8083 \
  -p 8086:8086 \
  -v /path/for/influxdb:/var/lib/influxdb \
  -v /path/for/grafana:/var/lib/grafana \
  philhawthorne/docker-influxdb-grafana:latest
```

To stop the container launch:

```
docker stop docker-influxdb-grafana
```

To start the container again launch:

```
docker start docker-influxdb-grafana
```

## Mapped Ports

| Host | Container | Service |
|---|---|---|
| 3003 | 3003 | grafana |
| 3004 | 8083 | chronograf |
| 8086 | 8086 | influxdb |

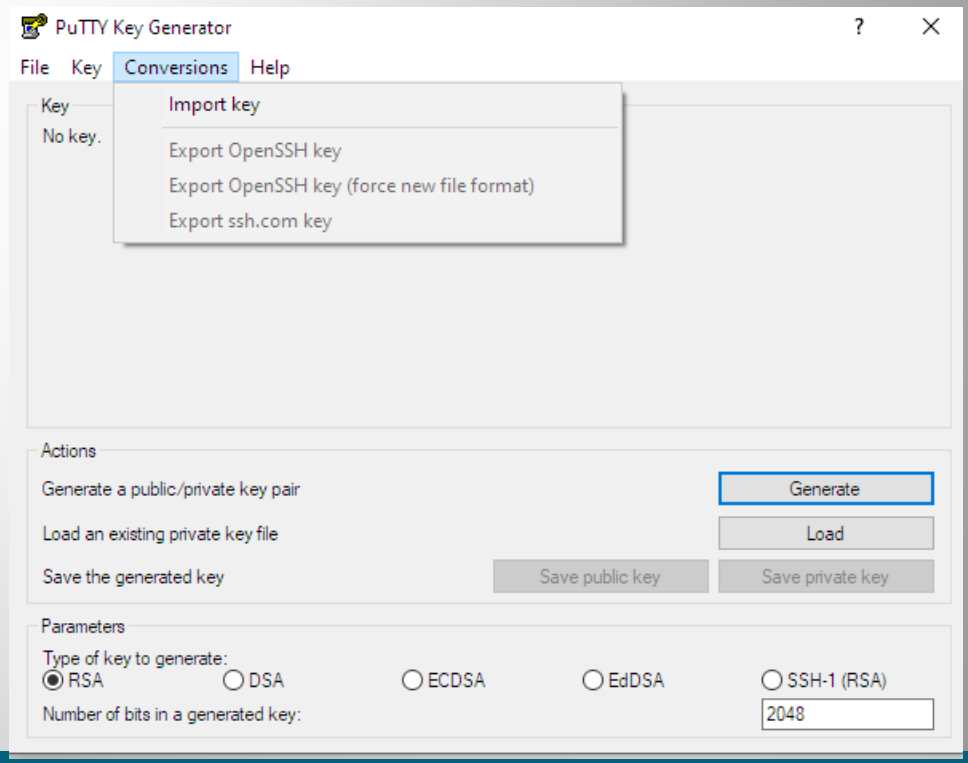
# Création des nœuds, connexion

INTERACTIF

Se connecter à un shell docker (utiliser le bash) : (root car supervisor), ssh localhost:2022 → mpiuser, utiliser la clé ppk

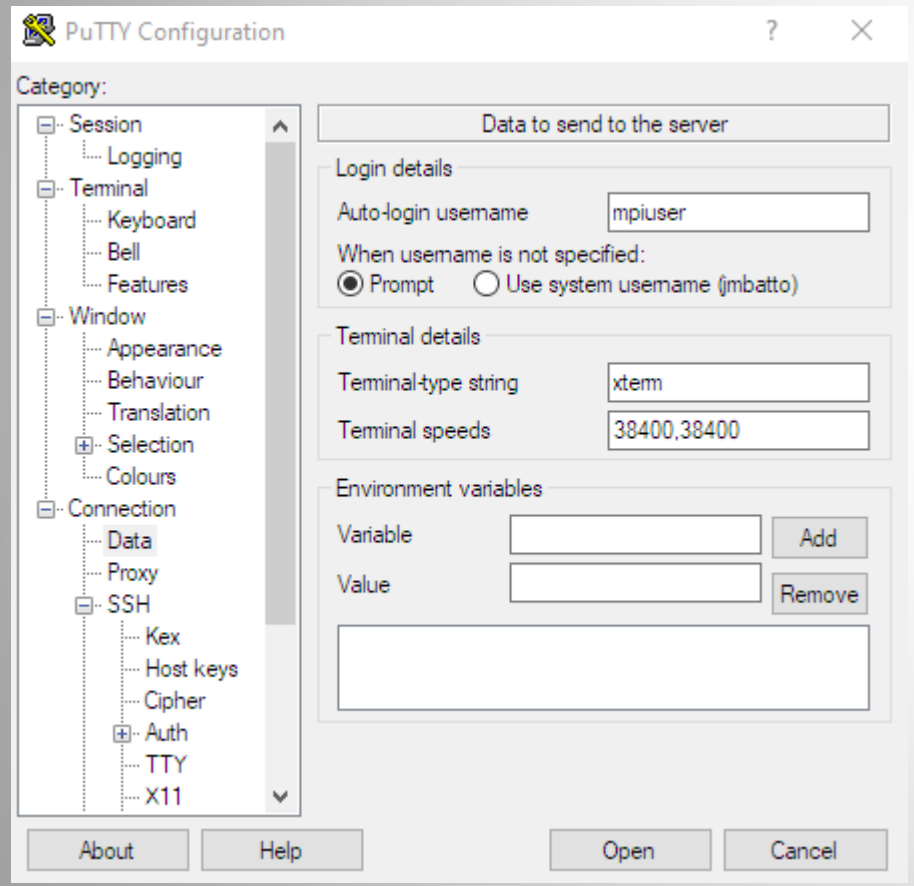
Si utilisation de Putty sous Windows

Pb : la clé fournie n'est pas au format ppk utilisé par Putty ➔ conversion de la clé au format PPK avec PuTTYgen



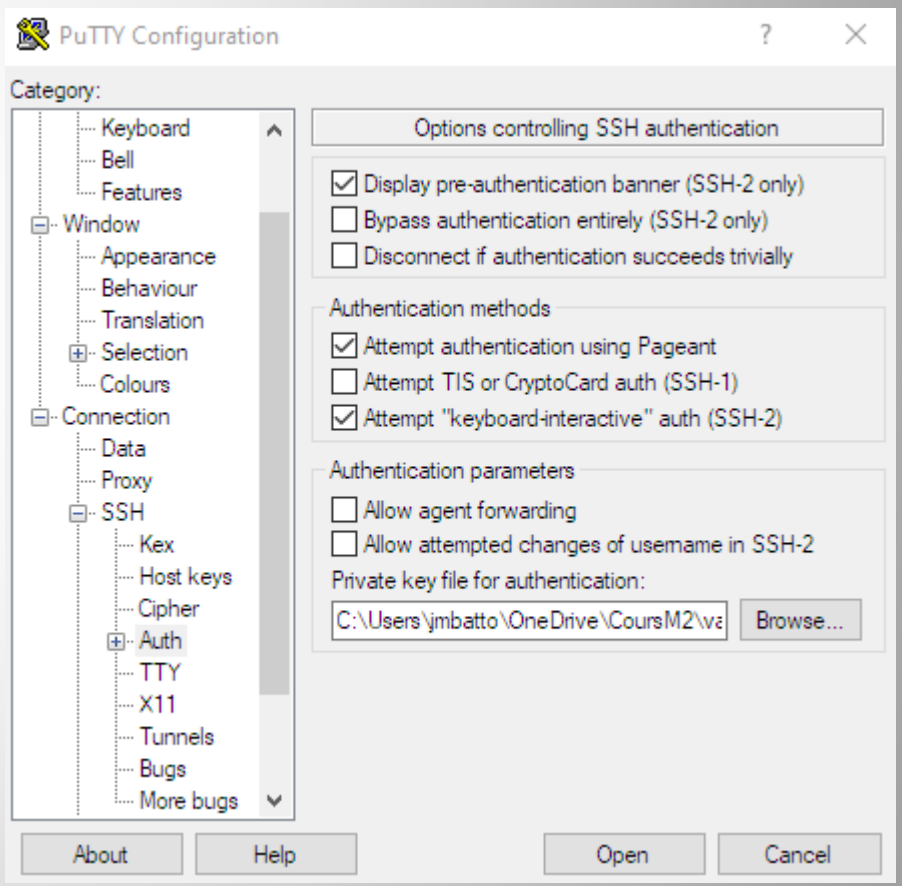
INTERACTIF

# Connexion avec PuTTY, 2 paramétrages à effectuer



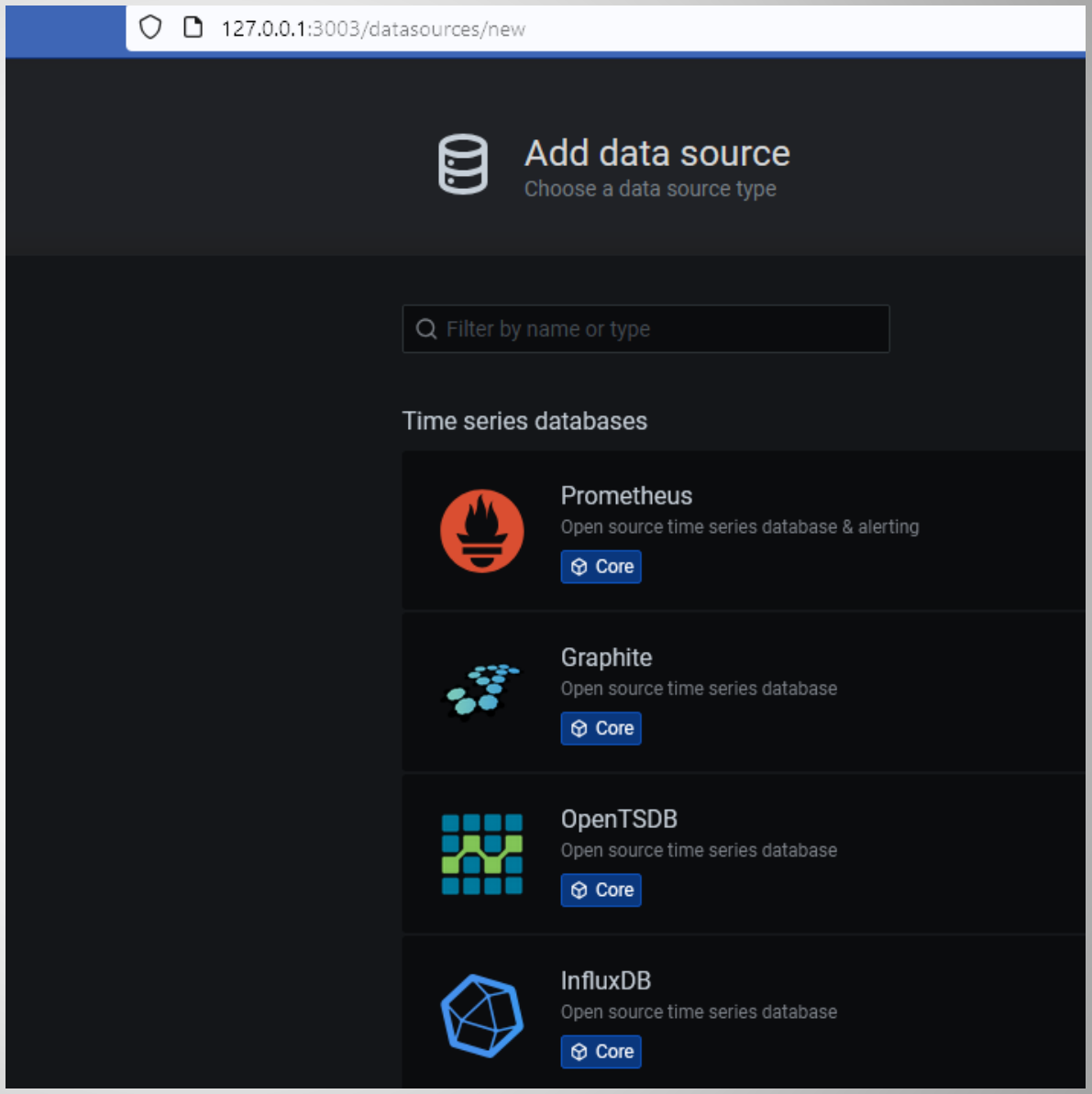

Le nom de login

La private key



- http://127.0.0.1:3003
- Login root/root
- DB : browser mode
- Target : telegraf

- ➔on ne va pas utiliser grafana



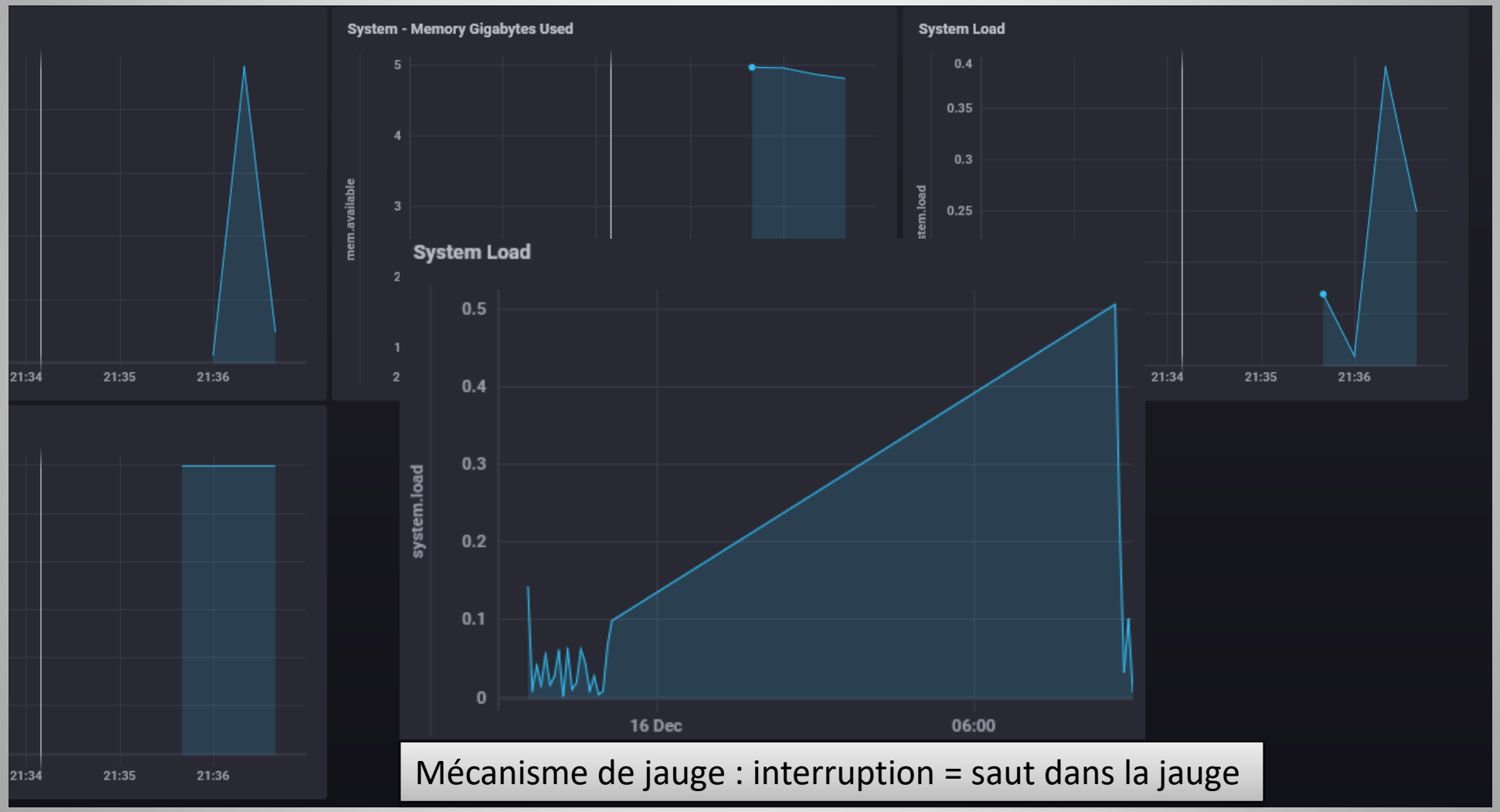
# Métrologie chronograf

- http://127.0.0.1:3004
- On va travailler avec chronograf

# Vision de la télémétrie technique

INTERACTIF

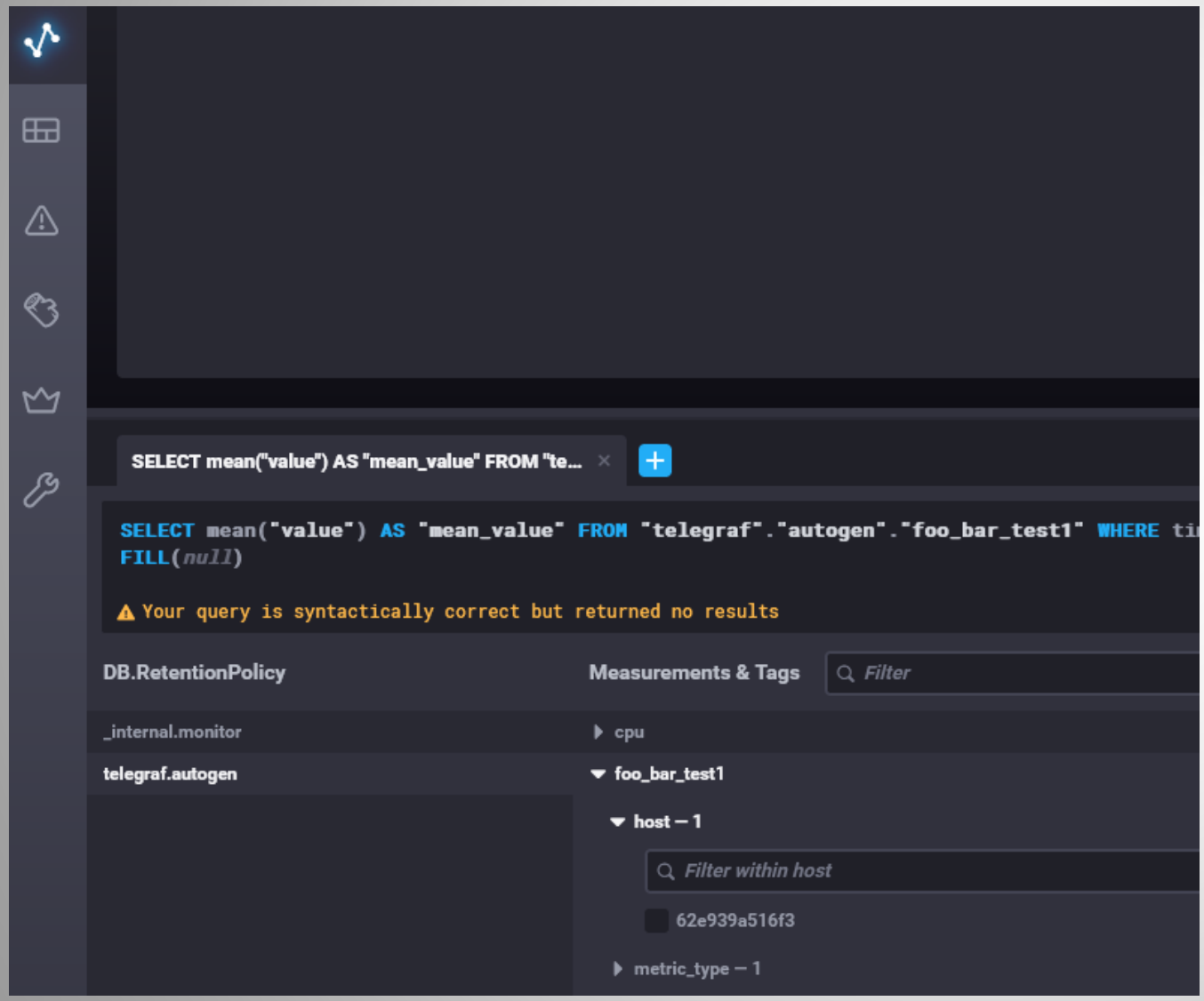
**Chaque nœud a son service telegraf – qui peut suivre des applications et des services**




Mécanisme de jauge : interruption = saut dans la jauge

# Depuis mpihead-1

- Dans un shell faire la cmd `echo "foo.bar.test1:+1|g" | nc -u localhost 8125`




apparait foo_bar_test1 dans chronograf

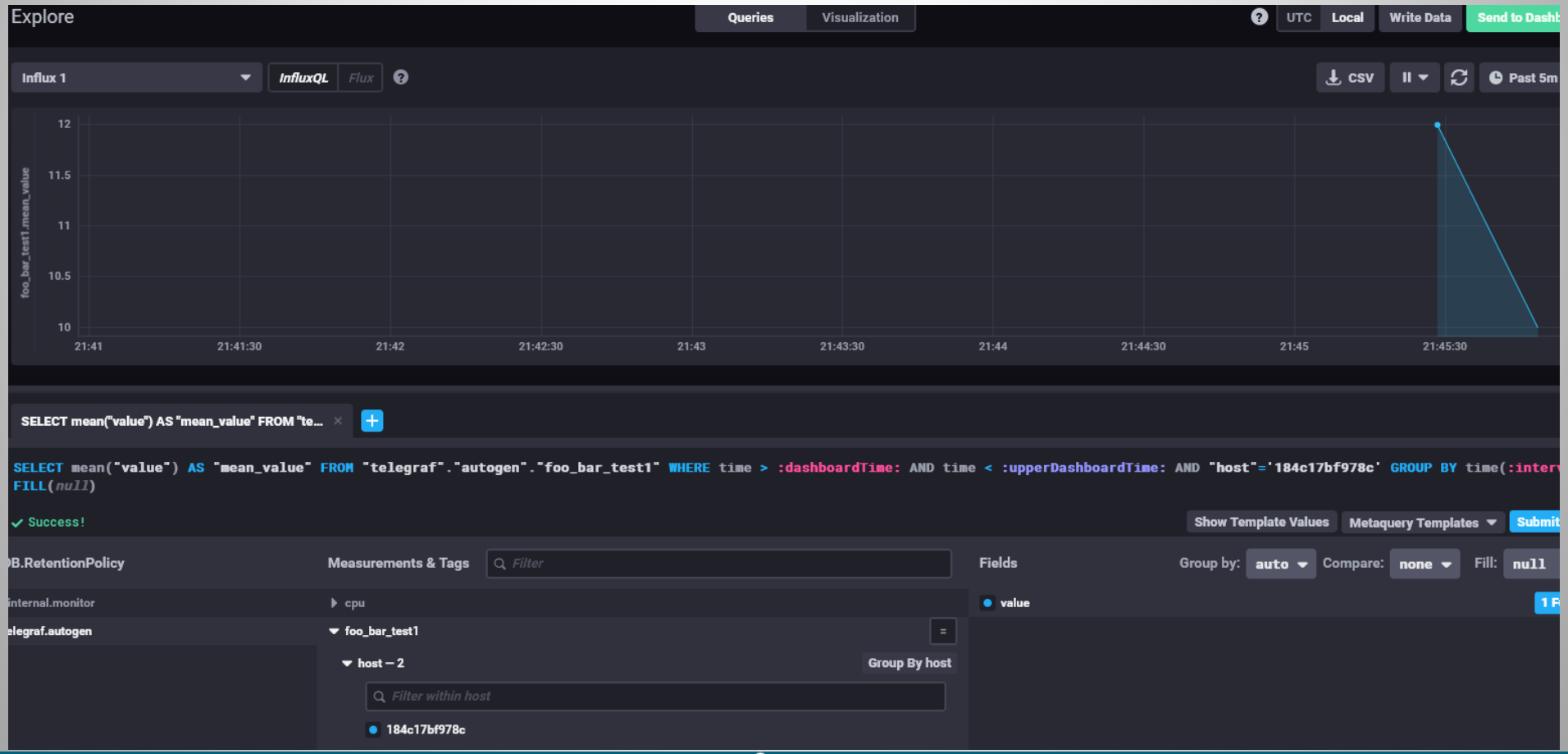
# Télémétrie applicative

INTERACTIF

- statsd (2011) est un service d'audit répandu qui écoute sur un port (socket)
- Écriture avec netcat sur le port 8125 → redirection vers influxdb

```
echo "foo.bar.test1:+1|g" | nc -u localhost 8125
```

**Telegraf ajoute une encapsulation https et redirige localhost vers un collecteur**



# Paramétrage telegraf

INTERACTIF

- ➔fichier telegraf.conf
- Le paramétrage est « anonyme » - on ne précise pas le nom du nœud
- Le service telegraf peut encrypter la communication (confidentialité)

# Question : comment faire un netcat applicatif?

INTERACTIF

- 2 pistes (git clone) :
- https://github.com/guzlewski/netcat.git

- https://github.com/romanbsd/statsd-c-client.git
- (attention à LD_LIBRARY_PATH)

# TD individuel pour le 22 décembre 2024

INTERACTIF

- EB : faire la télémétrie applicative de 2 nœuds MPI (osu_bibw.c)
- SFD : voici un exemple de fonction en langage Go, réalisé une fonction en C et utilisez là pour mesurer la durée de chaque benchmark.

```
//transaction : le nom de la portion de code instrumentée
//since une durée
func Chrono(since time.Duration, transaction string) {
        var client *statsd.Client
        client = statsd.NewClient("localhost:8125", statsd.MaxPacketSize(1400),
   statsd.MetricPrefix(ChronoGeneralTag+"."))← le point est supprimé
        client.PrecisionTiming("requestTime",
                since,
                statsd.StringTag("transaction", transaction))
                        client.Close()
   }
}
```

# TD individuel pour le 22 décembre 2024 minuit

- A me rendre :
- Mettre dans la Forge Gogs un rapport du TD2 (**1 page** PDF – présente un benchmark dans un tableau avec une introduction qui donne le contexte, avec le nom de l'étudiant et la date)
- Avec la/les captures « chronograf » commentées
- +code source commenté (dedans : date, nom étudiant)

# Idée de la séquence

- 1/ tester en mode multi-nœud MPI un code XMP
  - https://omni-compiler.org/

XcalableMP, XMP for short, is a directive-based language extension which allows users to develop parallel programs for distributed memory systems easily and to tune the performance by having minimal and simple notations.

Support typical parallelization under "global-view model"

XMP enables parallelizing the original sequential code using minimal modification with simple directives.

Support coarray to use one-sided communication easily under "local-view model"

Programmer can use coarray syntax in both XMP/Fortran and XMP/C. In particular, XMP/Fortran is designed to be compatible with Coarray Fortran.

Combination of MPI and OpenMP

In order to call an MPI program from an XMP program, XMP provides the MPI programming interface. Moreover, OpenMP directives can be combined into XMP as a hybrid programming.

# Idée de la séquence

- 2/test d'un code XMP, lancement du code
- 3/adaptation du code pour afficher le nom du noeud

```
mpiuser@62e939a516f3:~/YMLEnvironment/test-spawn-xmp$ cd /usr/local/var/mpishare/
mpiuser@62e939a516f3:/usr/local/var/mpishare$ mpirun --mca orte_base_help_aggregate 0 --mca btl_tcp_if_inclu
de 10.0.1.0/24 -n 4 -host 10.0.1.7,10.0.1.4,10.0.1.5,10.0.1.6 worker_program
rank = 0 ; Processeur 0 - Nom : 62e939a516f3 (Longueur du nom : 12)

(0, 0, 0) Processeur 2 - Nom : 21cdccdc88b1 (Longueur du nom : 12)
rank = 2 ;
(2, 2, 0)
(2, 2, 1)
(2, 3, 0)
(2, 3, 1) rank = 3 ; Processeur 1 - Nom : f387910ddb9b (Longueur du nom : 12)
Processeur 3 - Nom : 3fca871ee6fd (Longueur du nom : 12)

(3, 2, 2)
(3, 2, 3)
(3, 3, 2)
(3, 3, 3)
(0, 0, 1)
(0, 1, 0)
(0, 1, 1) rank = 1 ;
(1, 0, 2)
(1, 0, 3)
(1, 1, 2)
(1, 1, 3) mpiuser@62e939a516f3:/usr/local/var/mpishare$
```

# Test du code XMP

INTERACTIF

- Dans /YMLEnvironment/test-spawn-xmp → faire un make
- sudo curl --unix-socket /var/run/docker.sock http://localhost/containers/json | jq -r 'map(.NetworkSettings[]."yml_mpinet"."IPAMConfig"."IPv4Address") []`
- →pb dans la liste obtenue, il y a l'ip de l'image influxdb → faire un ifconfig –a sur l'image pour récuper l'ip

En rouge ce qui doit être adapté

```
mpirun --mca orte_base_help_aggregate 0 --mca btl_tcp_if_include
  10.0.1.0/24 -n 4 -host 10.0.1.7,10.0.1.4,10.0.1.5,10.0.1.6
  worker_program
```

# Init.c ➔ explication

```
#include <stdio.h>
#include "Matrix.xmptype.h"
#pragma xmp nodes p(2,2)
#pragma xmp template t(0:3,0:3)
#pragma xmp distribute t(block,block) onto p
XMP_Matrix A[4][4];
#pragma xmp align A[i][j] with t(j,i)
XMP_Matrix B[4][4];
#pragma xmp align B[i][j] with t(j,i)
XMP_Matrix C[4][4];
#pragma xmp align C[i][j] with t(j,i)
int main(int argc, char ** argv){
  int rank;
  MPI_Barrier(MPI_COMM_WORLD);
  MPI_Comm_rank(MPI_COMM_WORLD,&rank);
  int i,j,n;
  n=4;
  fprintf(stderr,"rank = %d ; ",rank);

  #pragma xmp loop (i,j) on t(j,i)
  for (i=0;i<n;i++){
    for(j=0;j<n;j++){
      fprintf(stderr,"\n(%d, %d, %d) ",rank,i,j);
      C[i][j] = 0;
      A[i][j] = 1;
      B[i][j] = i*n+j+1;
    }
  }
  MPI_Barrier(MPI_COMM_WORLD);
}
```

# Modification du code init.c

INTERACTIF

```
// Allouer de la mémoire (dans cet exemple, pour une chaîne de caractères)
    char* processor_name = malloc(256 * sizeof(char)); // Alloue de l'espace
pour le nom du processeur
    int name_len;
    MPI_Get_processor_name(processor_name, &name_len);

    // Afficher le résultat
    printf("Processeur %d - Nom : %s (Longueur du nom : %d)\n", rank,
processor_name, name_len);

    // Libérer la mémoire allouée
    free(processor_name);
```

# Le code modifié

INTERACTIF

mpiuser@62e939a516f3: ~/YMLEnvironment/test-spawn-xmp

```
#pragma xmp nodes p(2,2)
#pragma xmp template t(0:3,0:3)
#pragma xmp distribute t(block,block) onto p

XMP_Matrix A[4][4];
#pragma xmp align A[i][j] with t(j,i)

XMP_Matrix B[4][4];
#pragma xmp align B[i][j] with t(j,i)

XMP_Matrix C[4][4];
#pragma xmp align C[i][j] with t(j,i)

int main(int argc, char ** argv){
  int rank;
  MPI_Barrier(MPI_COMM_WORLD);
  MPI_Comm_rank(MPI_COMM_WORLD,&rank);
  int i,j,n;
  n=4;
  fprintf(stderr,"rank = %d ; ",rank);
// Allouer de la mémoire (dans cet exemple, pour une chaîne de caractères)
    char* processor_name = malloc(256 * sizeof(char)); // Alloue de l'espace pour le nom du processeur
    int name_len;
    MPI_Get_processor_name(processor_name, &name_len);

    // Afficher le résultat
    printf("Processeur %d - Nom : %s (Longueur du nom : %d)\n", rank, processor_name, name_len);

    // Libérer la mémoire allouée
    free(processor_name);
  #pragma xmp loop (i,j) on t(j,i)
  for (i=0;i<n;i++){
    for(j=0;j<n;j++){
      fprintf(stderr,"\n(%d, %d, %d) ",rank,i,j);
      C[i][j] = 0;
      A[i][j] = 1;
      B[i][j] = i*n+j+1;
    }
  }
  MPI_Barrier(MPI_COMM_WORLD);
}


mpiuser@62e939a516f3:~/YMLEnvironment/test-spawn-xmp$
```

# Test du code XMP

INTERACTIF

- Dans /YMLEnvironment/test-spawn-xmp → faire un make
- sudo curl --unix-socket /var/run/docker.sock http://localhost/containers/json | jq -r 'map(.NetworkSettings[]."yml_mpinet"."IPAMConfig"."IPv4Address") []`
- →pb dans la liste obtenue, il y a l'ip de l'image influxdb → faire un ifconfig -a sur l'image pour récuper l'ip

En rouge ce qui doit être adapté

```
mpirun --mca orte_base_help_aggregate 0 --mca btl_tcp_if_include
  10.0.1.0/24 -n 4 -host 10.0.1.7,10.0.1.4,10.0.1.5,10.0.1.6
  worker_program
```