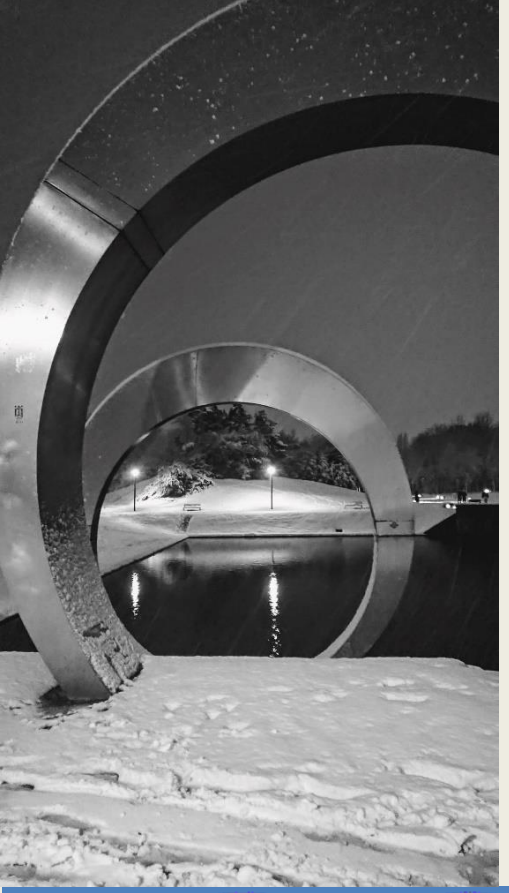


UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

université PARIS-SACLAY

#5

06/01/2026

jean-michel.batto@cea.fr

cea

https://gogs.eldarsoft.com/M2_IHPS

Supervision

La fonction de supervision ➔ télémétrie

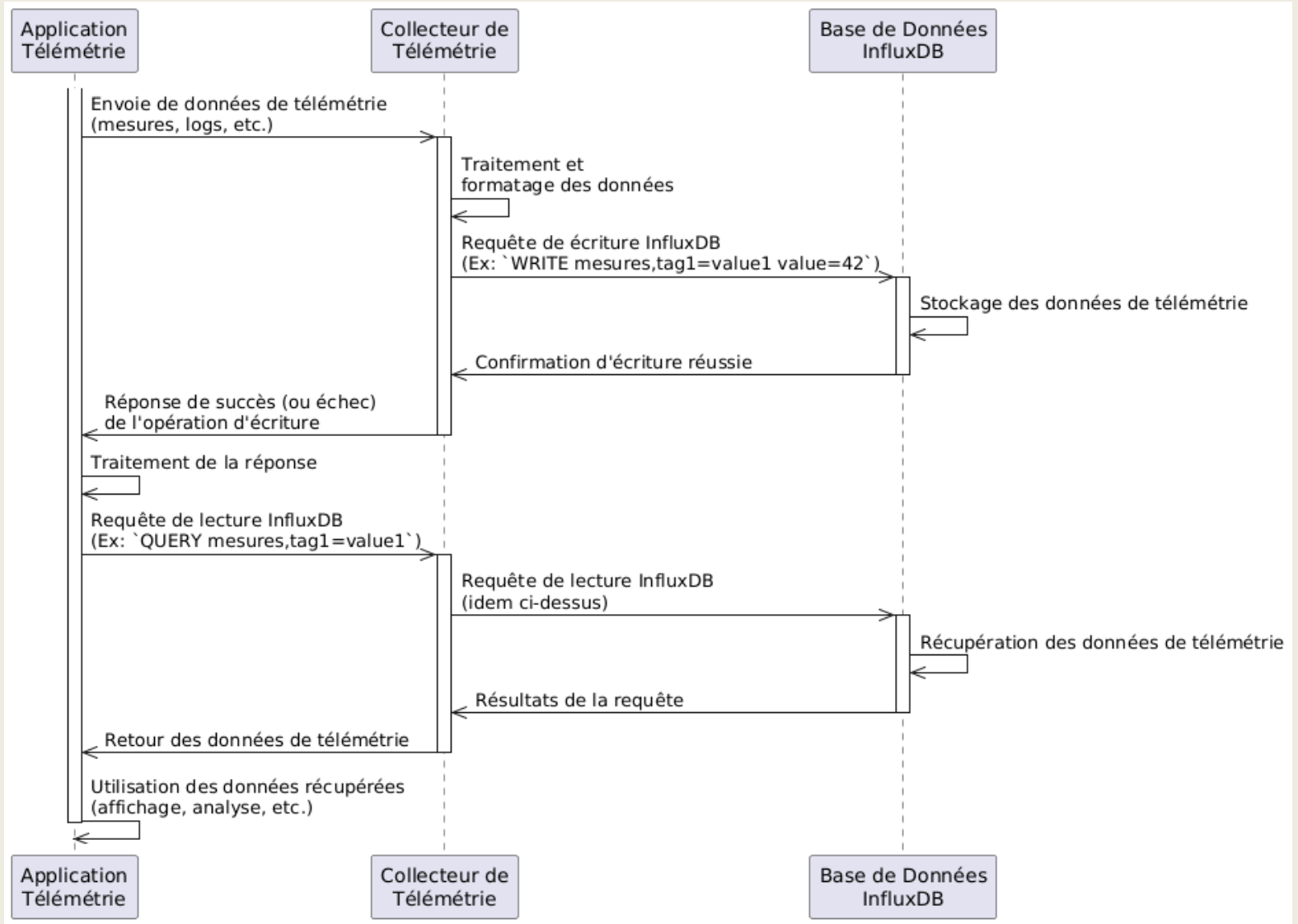
Supervision technique

Supervision applicative

Permettre la détection des défauts du système (technique)

Aider à diagnostiquer les problèmes du point de vue de l'utilisateur (applicative)

Vérifie la bonne allocation des ressources (la saturation ou la sous utilisation) et lance des alertes (technique et applicative) ➔ l'application peut avoir un problème de conception (ie consommation disque).



Le concept de « Time Series »

La métrologie a une dimension de projection : le temps.

Il s'agit d'une Database spécialisée pour les données indexées sur la dimension temps.

DB Spécialisée pour aggréger les données sur un intervalle de temps.

Les 3 principaux (2024)

InfluxDB (Go), Timescaledb (PostgreSQL- C), Prometheus (Go)

INTERACTIF

Quelles sont les principales DB Time Series et en quels langages sont elles écrites ? Peux-tu faire une réponse concise ?

TICK Stack

Influxdb a lancé le concept de TICK Stack

*T : Telegraf, *I : InfluxDB, *C : Chronograf, *K : Kapacitor

Il s'agit d'un ensemble cohérent pour la métrologie, le stockage, la visualisation et l'alerte.

Alternative Prometheus

Que contient le docker-compose ?

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

image disponible sur hub.docker.com

Que contient l'image ?

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

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

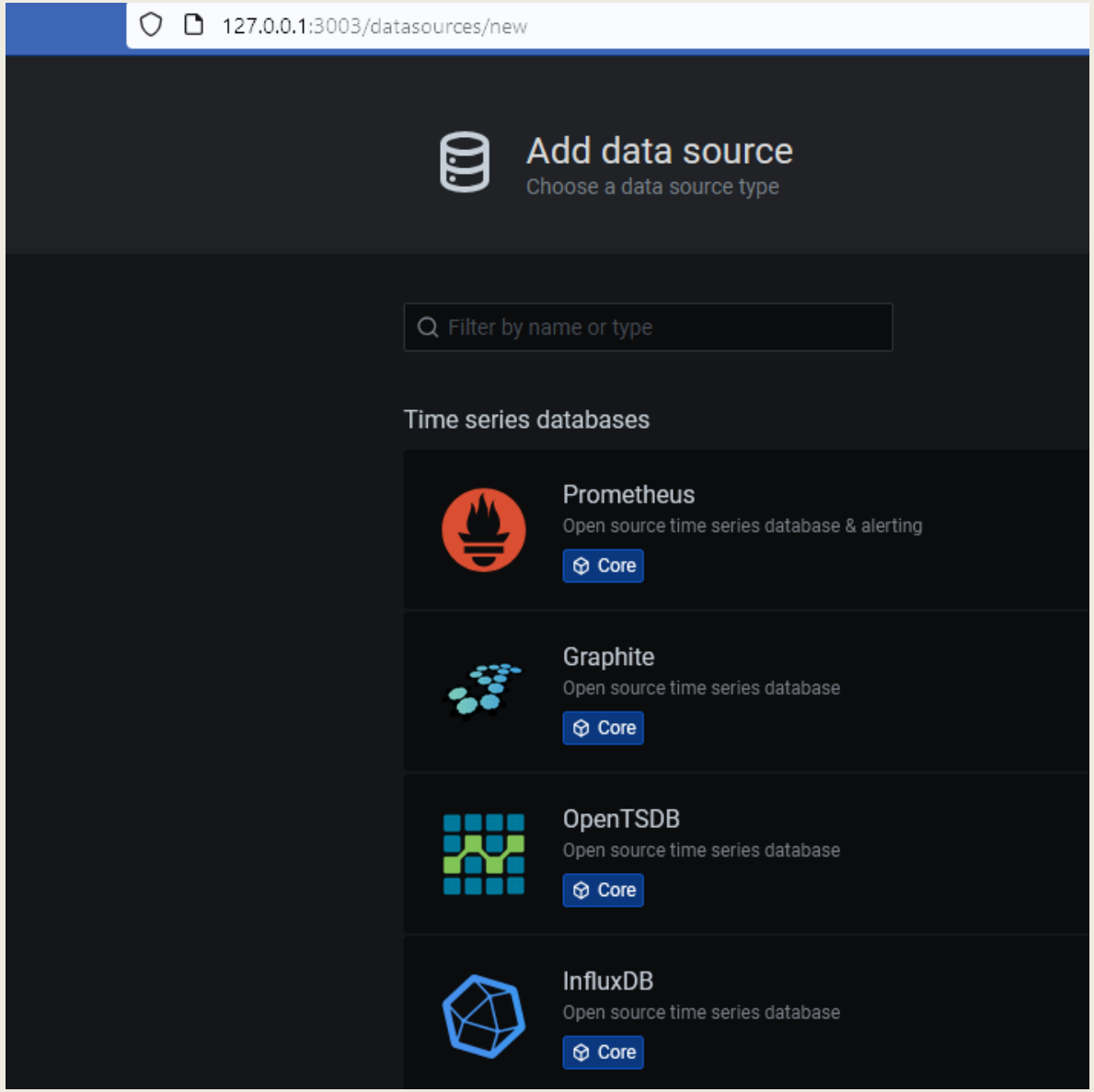
http://127.0.0.1:3003

Login root/root

DB : browser mode

Target : telegraf

➔on ne va pas utiliser grafana



Métrologie chronograf

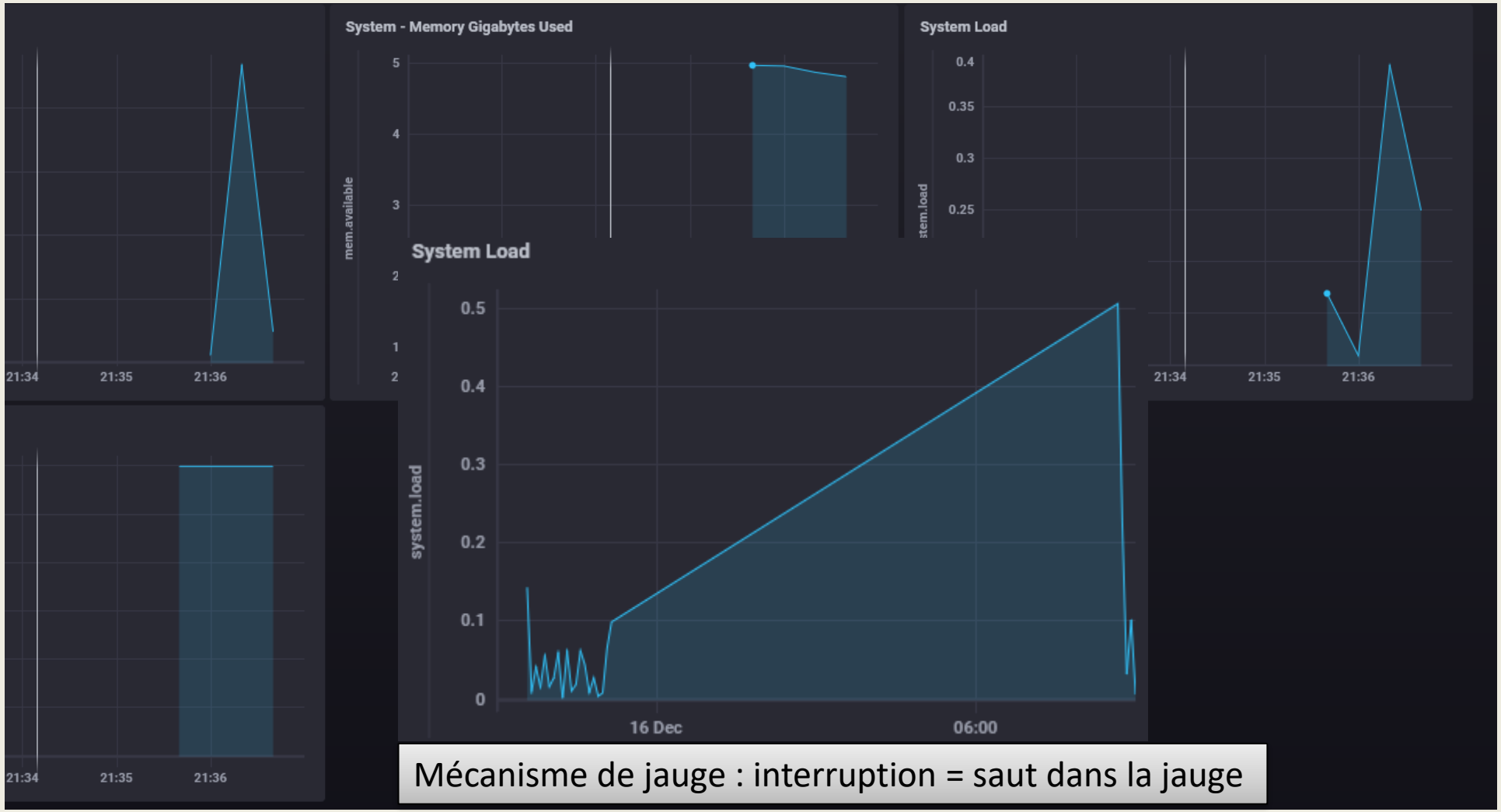
http://127.0.0.1:3004

On va travailler avec chronograf

Vision de la télémétrie technique

INTERACTIF

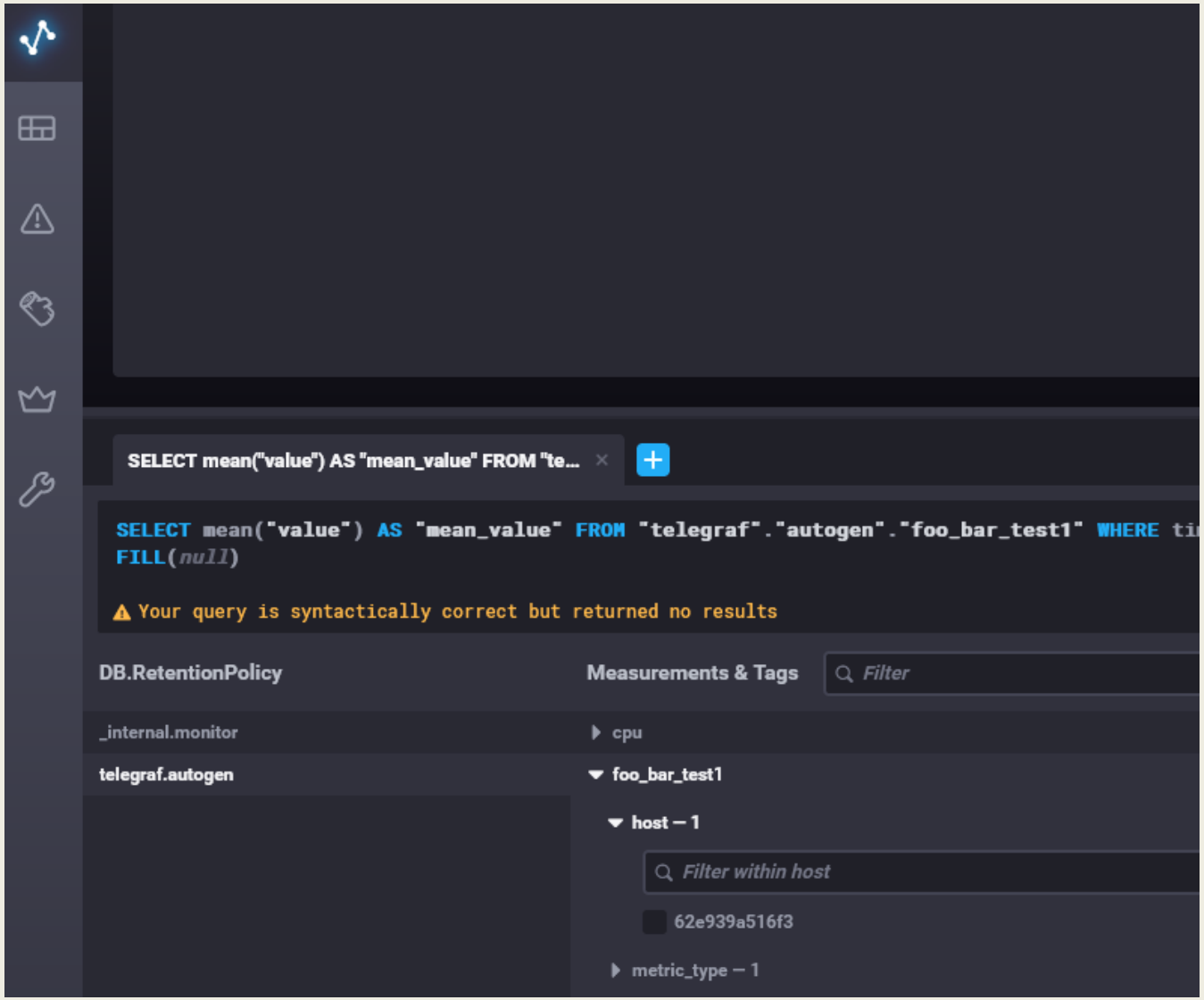
Chaque nœud a son service telegraf – qui peut suivre des applications et des services



Depuis le conteneur slurmctld

Dans un shell faire la cmd

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



apparait foo_bar_test1 dans chronograf

Télémétrie applicative

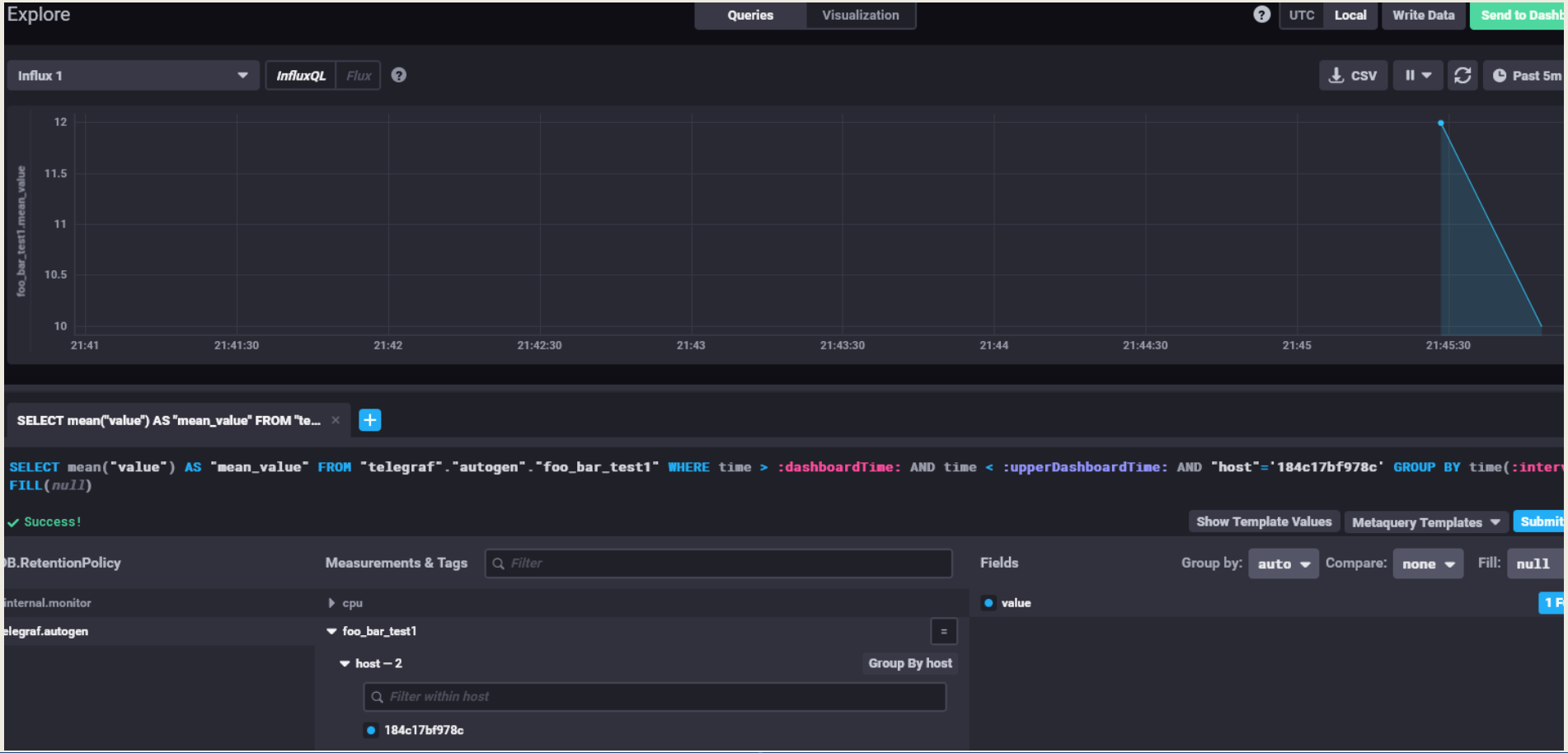
INTERACTIF

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 11 janvier 2026

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()
   }
}
```

Possibilité d'utiliser un shell, et SLURM cf. suite du cours (bonification importante si SLURM)

TD individuel pour le 11 janvier 2026, 23h59

A me rendre :

Mettre dans la Forge Gogs un rapport « TD3 » (**qq pages** PDF – présente un benchmark dans un tableau avec une introduction qui donne le contexte, avec le nom de l'étudiant et la date)

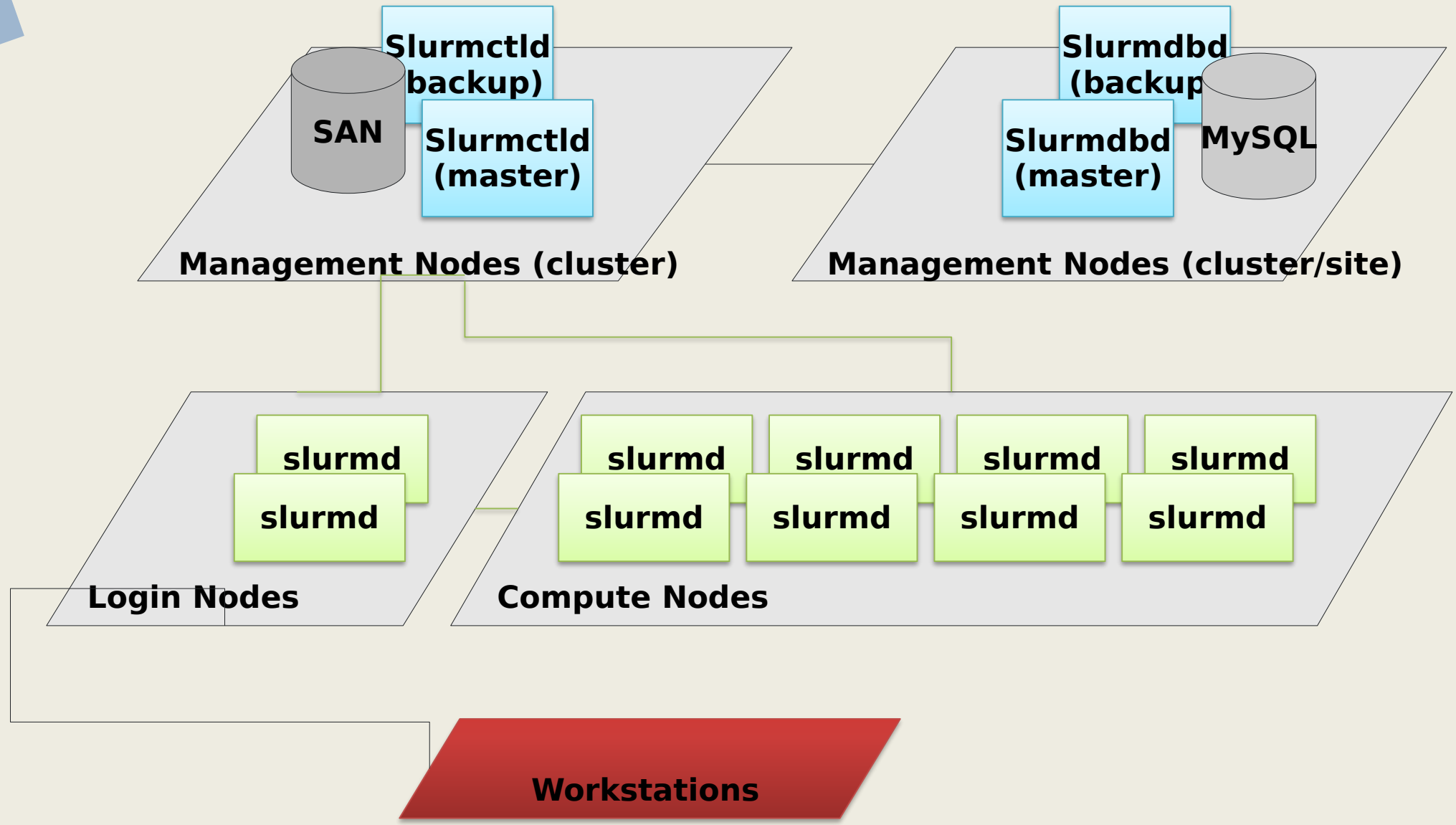
Un diagramme C4 level 2 (réalisé avec PlantUML)

Avec la/les captures « chronograf » commentées, les codes sources (surtout si utilisation de SLURM)

+code source commenté (dedans : date, nom étudiant)

Retour sur SLURM : comment ajouter des nœuds ?

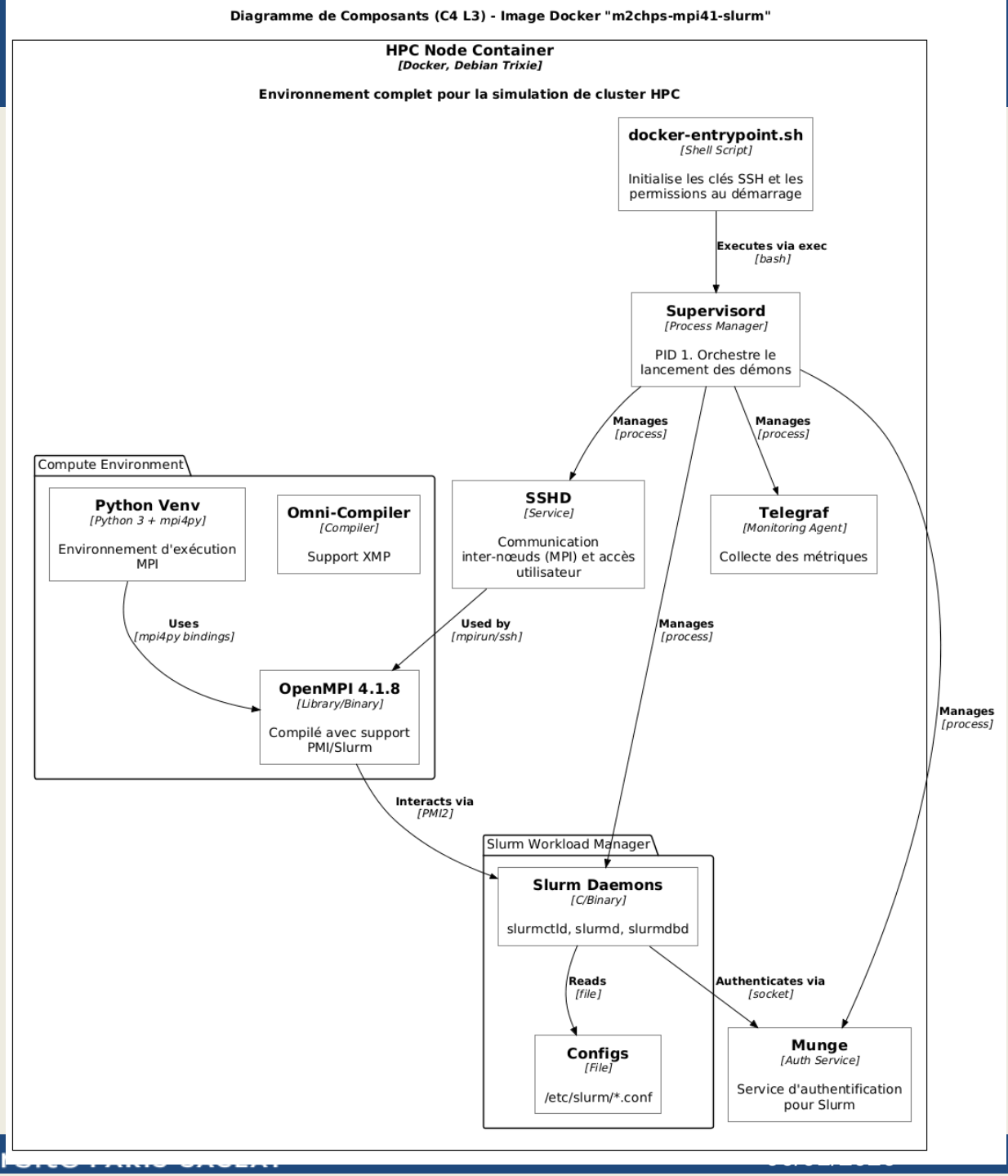
RECAP



Ce que contient l'image jmbatto/m2chps-mpi41-slurm

Slurm (3 instances possibles du nœuds)
Démarrées avec Supervisord (et non avec tini)
➔plusieurs daemon sont démarrés

- sshd
- telegraf
- slurm (avec un choix selon le paramétrage docker-compose)
- munge



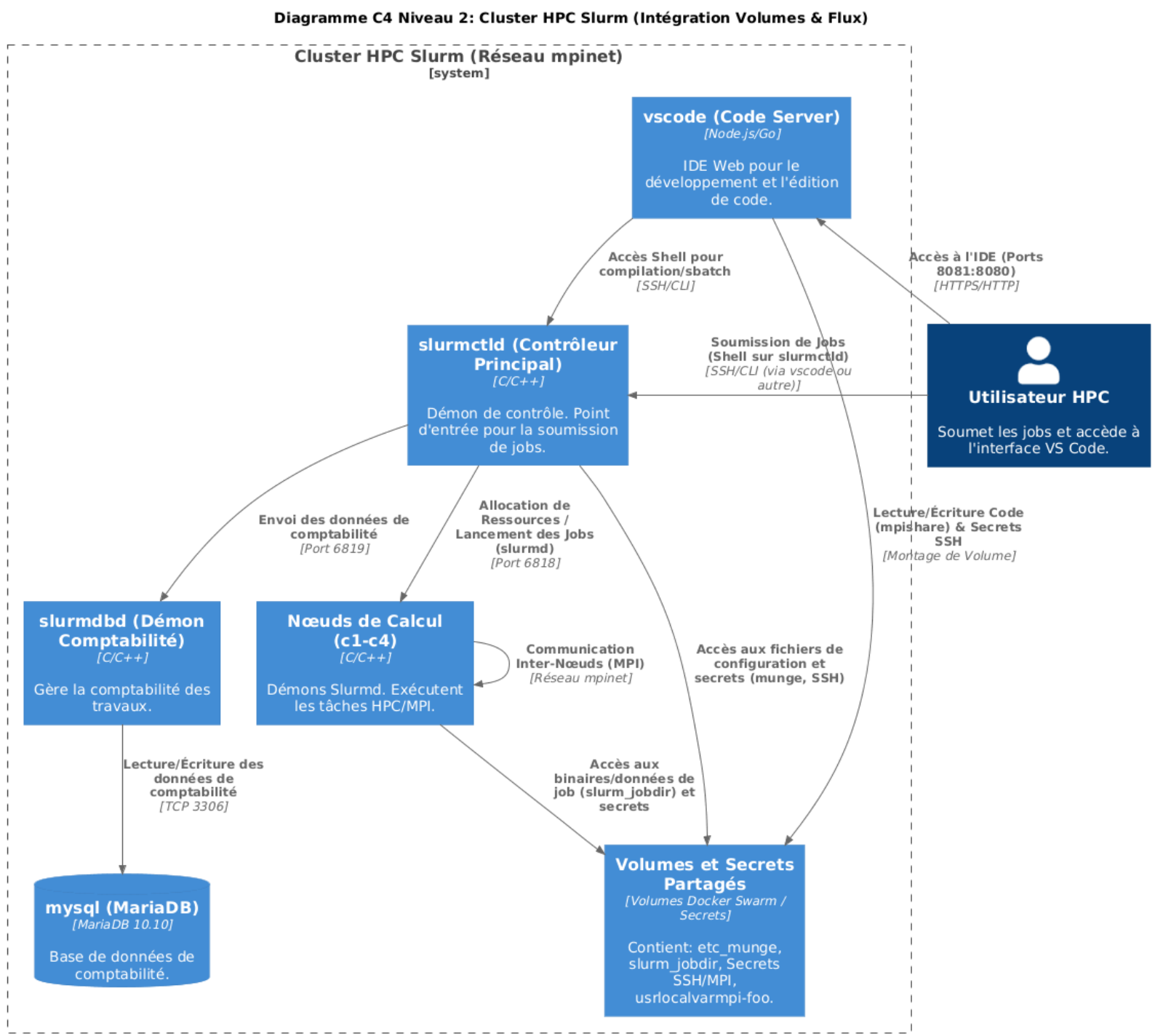


Diagramme C4 Niveau 2: Cluster HPC Slurm (Intégration Volumes & Flux)
Cluster HPC Slurm (Réseau mpinet)
[system]
vscode (Code Server)
[Node.js/Go]
IDE Web pour le développement et l'édition de code.
Accès Shell pour compilation/sbatch
[SSH/CLI]
Accès à l'IDE (Ports 8081:8080)
[HTTPS/HTTP]
slurmctld (Contrôleur Principal)
[C/C++]
Démon de contrôle. Point d'entrée pour la soumission de jobs.
Soumission de Jobs (Shell sur slurmctld)
[SSH/CLI (via vscode ou autre)]
Utilisateur HPC
Soumet les jobs et accède à l'interface VS Code.
Envoi des données de comptabilité
[Port 6819]
Allocation de Ressources / Lancement des Jobs (slurmd)
[Port 6818]
Lecture/Écriture Code (mpishare) & Secrets SSH
[Montage de Volume]
slurmdbd (Démon Comptabilité)
[C/C++]
Gère la comptabilité des travaux.
Nœuds de Calcul (c1-c4)
[C/C++]
Démons Slurmd. Exécutent les tâches HPC/MPI.
Communication Inter-Nœuds (MPI)
[Réseau mpinet]
Accès aux fichiers de configuration et secrets (munge, SSH)
Lecture/Écriture des données de comptabilité
[TCP 3306]
Accès aux binaires/données de job (slurm_jobdir) et secrets
mysql (MariaDB)
[MariaDB 10.10]
Base de données de comptabilité.
Volumes et Secrets Partagés
[Volumes Docker Swarm / Secrets]
Contient: etc_munge, slurm_jobdir, Secrets SSH/MPI, usrlocalvarmpi-foo.

Prise en main de SLURM / TD d'exploration

INTERACTIF

Swarm init ➔ déjà fait dans les TD précédents

➔Docker créé un contexte préfixé par le nom du répertoire…

avec le docker-compose.yml, installation de

- 1 db mysql pour le nœud slurmdbd,
- 1 noeud slurmdbd,
- 1 noeud de contrôle,
- 2 nœuds de calcul c1/c2, le tout sous MPI

`docker compose up -d`

`Dans cette nouvelle version on ajoute 2 nœuds (c3/c4) et 1 conteneur avec vscode et 1 conteneur pour télémétrie`

Ouvrir 2 shells (un pour le nœud de contrôle, un pour le nœud C1)

TD élémentaire

Etapes (step0/1/2) de découvertes par rapport aux contraintes

On travaille dans le conteneur slurmctld

on vérifie s'il y a une partition pour root

```
sacctmgr show association -p user=root
scontrol show partition
scontrol show nodes
sinfo -Nel
```

Sinon

```
sacctmgr --immediate add cluster name=linux
```

Puis un restart des images

Dans le répertoire /usr/local/var/mpishare faire

```
git clone http://gogs.eldarsoft.com/M2_IHPS/glcs_slurm.git
```

Dans les répertoires test1/test2

```
mpicc -g3 -o elementary elementary.c
```

Ajout des nœuds dans la partition SLURM

INTERACTIF

On veut ajouter les nœuds c3 et c4.

Dans les conteneurs slurmctld, c1, c2, c3, c4 : modifier le fichier /etc/slurm/slurm.conf ➔ 2 lignes à changer

```
# COMPUTE NODES
NodeName=c[1-4] RealMemory=1000 State=UNKNOWN
#
# PARTITIONS
PartitionName=docker Default=yes Nodes=c[1-4] Priority=50 DefMemPerCPU=500 Shared=NO MaxNodes=4 MaxTime=5-00:00:00 DefaultTime=5-00:00:00 State=UP
```

Puis redémarrer les conteneurs. ➔ commande sinfo

```
mpiuser@slurmctld:/usr/local/var/mpishare/glcs_slurm$ sudo su -
root@slurmctld:~# sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
docker*      up 5-00:00:00      4   idle c[1-4]
```

TD d'exploration

Etapes du TD

Explorer les répertoires step0, step1, step2

`salloc -n 4 mpirun ./elementary` ➔ attention il faut que le binaire soit visible des nœuds (pb de partage du binaire – modifier les scripts…)

`sbatch script0.sh`

`sacct -j %jobid obtenu au moment du sbatch%`

Dans le répertoire step0, modifier le script pour avoir un code retour différent de 0

Vérifier le résultat du batch

`sbatch -n 4 xxx.sh` //(selon le répertoire)

`squeue -s -j %jobid%`

`sacct -j%jobid%`

`squeue -s -i 30 -j %jobid%`

`sacct -j %jobid%`

Retour à SLURM

INTERACTIF

passer impiuser sur le noeud slurmctld (mais ça fonctionne aussi pour root)

```
//vérifie l'état de la queue
squeue
//puis allocation de 4 nœuds (il s'agit de c1 / c2 / c3 / c4)
salloc --time=05:00 -N 4
//vérification des noeuds
srun hosname
de la queue
squeue
//permet d'avoir les informations sur les jobs
sacct --format=JobID,elapsed,ncpus,ntasks,state,node
```

Sur le nœud de control

INTERACTIF

scontrol show node

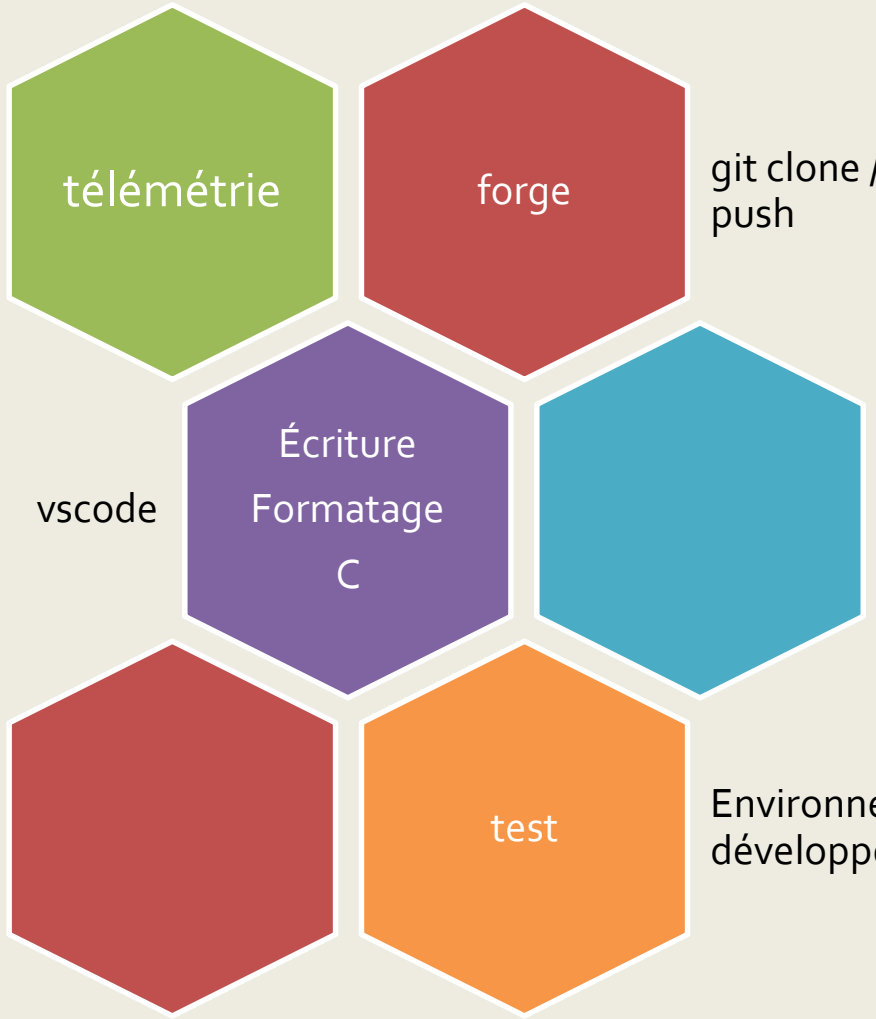
Ou alors

sinfo -Nel

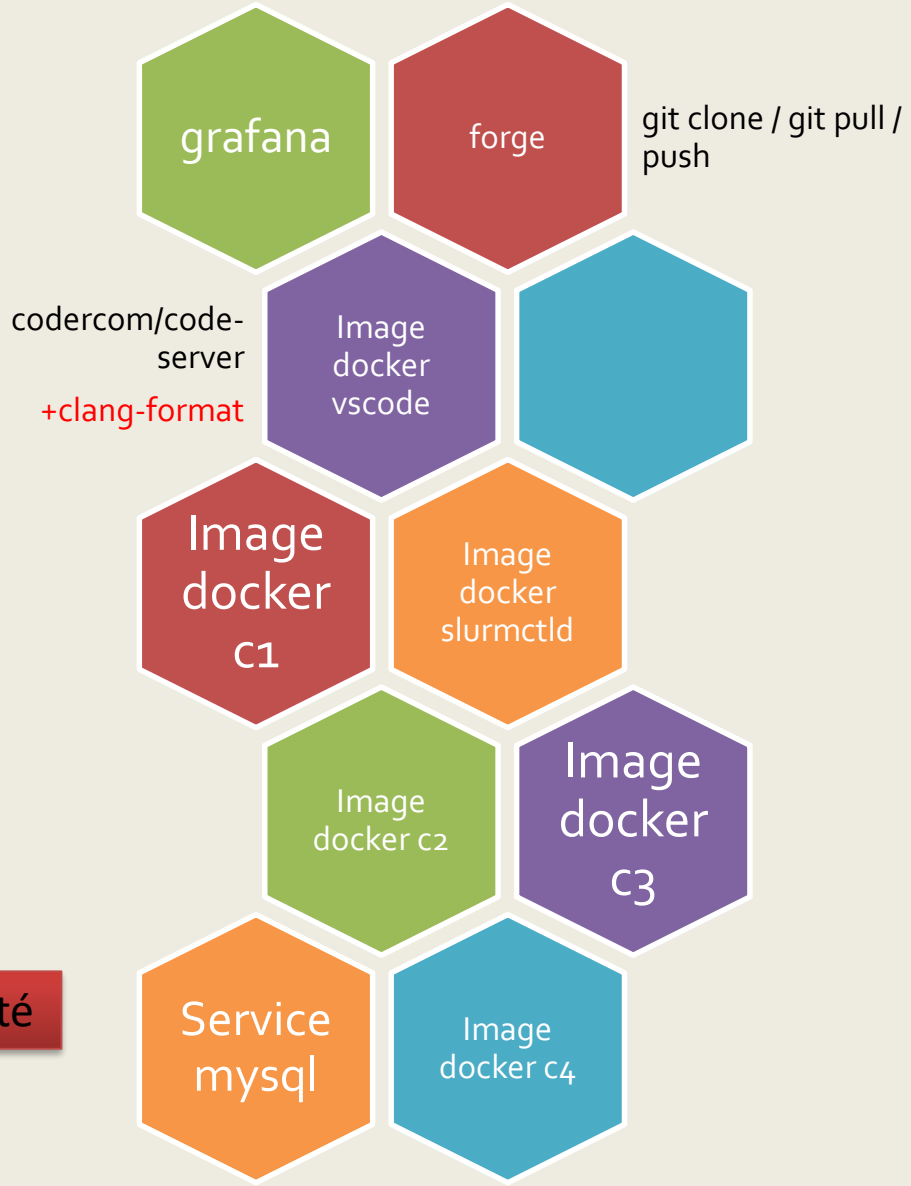
Pour voir l'historique de la conso

sacct

Cycle de vie du développement



Infrastructure du Supercalculateur



Attention : il manque la cybersécurité

Activité TD

INTERACTIF

Nous allons avoir 9 conteneurs:

- Grafana : la télémétrie
  - philhawthorne/docker-influxdb-grafana:latest
- Vscode : une version http de visual code studio
  - codercom/code-server:latest
- 4 nœuds MPI nommés c1-4 : 4 nœuds esclaves
- 3 nœuds = SLURMctld / SLURMdb / Mysql

Remarques:

- Partage de scripts (certif.sh et install_clang_format_go.sh) via les secrets !
- Le filesystem /usr/local/var/mpishare est partagé avec vscode (attention aux UID)
- Les nœuds sont nommés pour avoir un hostname ! (et pas de scale possible)
- Refresh des images au démarrage – sauf si c'est dans l'espace partagé docker-compose

Installation du plugin clang-format

INTERACTIF

Installation de clang-format

Dans le dossier .ssh-source

sh install_clang_format_go.sh

➔ Installation du package et en plus il y a le compilateur go

Installation de l'extension depuis vscode

Activité « découverte de XMP »

Idée : manipuler un bout de code dans un environnement de test multi-nœuds MPI

Vous allez créer un repo sur gogs

Vous allez cloner le repo https://gogs.eldarsoft.com/jmbatto/GLCS-CM5-TDXMP et copier le contenu dans votre repo local

Et nous allons ensemble explorer quelques idées.

XMP : XcalableMP

https://xcalablemp.org/handbook

Auteur : Mitsuhisa Sato

https://icl.utk.edu/newsletter/presentations/2012/Sato-Updates-on-XcalableMP-PGAS-Language-2012-08-29.pdf (dans le dossier support biblio)

Modèle de programmation « type SIMD », pour C et Fortran

Extension du langage basée sur des pragmas

Programmation parallèle évolutive et tenant compte des performances

Compilateur source (C+XMP) vers source (C+MPI)

Cartographie des données et des tâches selon patron de programmation

Mode opératoire

INTERACTIF

~~1/installation code-server et noeuds~~

2/paramétrage de git dans le terminal vscode

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

3/ dans gogs, vous allez créer un projet, puis faire un `git clone` https://login:password@votreprojet

L'idée est de vous permettre de travailler efficacement !

4/ vous récupérer l'exemple `GLCS-CM5-TDXMP`

Lancement du code

INTERACTIF

On peut faire un make run

Si on se positionne comme root avec la commande ifconfig –a, on voit 2 réseaux (eth0 et eth1)

mpirun --mca orte_base_help_aggregate 0 --mca btl_tcp_if_include 10.0.1.0/24 -host c1,c2,c3,c4 -n 4 worker_program

Exploration du code init.c

```
#pragma xmp nodes p(2, 2) ➔ on décrit 4 noeuds
#pragma xmp template t(0 : 3, 0 : 3) ➔ la tâche est une matrice 4 * 4
#pragma xmp distribute t(block, block) onto p ➔ la tâche t va être distribuée sur les 4 nœuds (on va donc avoir t(0,0), t(0,1), t(1,0), t(1,1) ➔ p(0) = le head (rank = 0) )
XMP_Matrix A[4][4]; ➔ une matrice 4x4
#pragma xmp align A[i][j] with t(j, i) ➔ inversion des indices
#pragma xmp shadow A[4][4] ➔ on veut un shadow pour la consolidation
```

Plus loin dans le code

```
#pragma xmp reflect(A) ➔ synchronisation de l'espace mémoire
```

https://xcalablemp.org/handbook/distribute.html

Questions

INTERACTIF

Que fait ce code ?

A quoi sert le `usleep(100)` ?

A quoi servent les barrières MPI

Est-ce que ce code est explicable par un LLM ?

Les 23 patterns du GoF

Elements of Reusable Object-Oriented Software

– Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides

Chez Addison-Wesley, 1995.

Ces patterns trés célèbres ont été conçus par 4 informaticiens surnommés le "Gang of Four" (d'où le terme GoF pour ces patterns).

Ils proposent des solutions élégantes, et toujours différentes pour résoudre différents problèmes récurrents rencontrés par les architectes logiciels.

Utilité de l'approche du GoF

Patterns et conception : standardiser la conception.

Formalisation de savoir-faire.

Augmenter le niveau d'abstraction ➔ Eviter de construire des objets trop adhérents du réel ➔ simplifie le réemploi.

Arbitrage entre flexibilité et performance ➔ l'adaptation aux évolutions.

GoF

En 3 séances (7, 7, 9)

Objectif : apprendre à bien utiliser les patterns de votre choix

+ en Golang : refaire quelques patterns pour bien comprendre le principe des patterns.

Golang en PPCS ?

Paradigme	Support dans Go	Niveau de support	Commentaires pour le calcul scientifique
Impératif	Oui, natif	Excellent	• Boucles for, variables mutables, pointeurs, etc. C'est même le style dominant de Go.
Fonctionnel	Oui, partiel mais très utilisable	Bon	• Fonctions de première classe - Closures - Fonctions anonymes et littérales - Types de fonctions (func comme type) - Méthodes comme valeurs de fonction • Mais : pas de pattern matching, pas d'immutabilité forcée, pas de tail-call optimization, pas de higher-order types (generics de fonctions limités avant Go 1.18, maintenant possibles mais verbeux). • En pratique, on écrit très bien du code fonctionnel (map/filter/reduce avec des closures).
Orienté objet	Oui, mais sans héritage classique	Très bon (style « composition over inheritance »)	• Méthodes sur n'importe quel type (struct, types de base…) – • Embedding (composition + promotion de méthodes) – • Interfaces implicites (duck typing) – • Polymorphisme via interfaces. C'est l'un des points forts de Go pour le calcul scientifique : on peut faire du code très propre avec des interfaces comme io.Reader, sort.Interface, ou des interfaces personnalisées pour les tenseurs, matrices, etc.
Déclaratif	Oui, dans une certaine mesure	Moyen à bon	• Go n'est pas un langage déclaratif pur (comme Prolog ou SQL), • mais : - Les interfaces sont déclaratives (« je veux quelque chose qui satisfait ça ») – • Le système de concurrence (channels + select) est souvent décrit comme déclaratif – • Les generics (depuis Go 1.18) permettent d'écrire du code plus déclaratif - On utilise beaucoup les littéraux de struct et de slice/map qui sont déclaratives • En calcul scientifique, on compense souvent avec des DSL embarqués (ex. gonum, expr via des libs tierces).
Parallèle / Concurrence	Oui, support de première classe	Excellent	• Goroutines ultra-légères – • Channels (communication sécurisée) - sync package (WaitGroup, Mutex, atomic, etc.) - go keyword + select + sync/atomic + context • C'est probablement le meilleur support natif de la concurrence parmi les langages systèmes modernes. • Très utile pour le calcul scientifique (parallélisation de boucles numériques, pipelines de traitement, simulations Monte-Carlo, etc.).

Les patrons / patterns

Les 23 patterns décrits :

<table>
<tr><th></th><th>Creational</th><th>Structural</th><th>Behavioral</th></tr>
<tr><td rowspan="2">Class</td><td rowspan="2">Factory Method</td><td rowspan="2">Adapter (class)</td><td>Interpreter</td></tr>
<tr><td>Template Method</td></tr>
<tr><td rowspan="8">Object</td><td>Abstract Factory</td><td>Adapter (object)</td><td>Chain of Responsibility</td></tr>
<tr><td>Builder</td><td>Bridge</td><td>Command</td></tr>
<tr><td>Prototype</td><td>Composite</td><td>Iterator</td></tr>
<tr><td rowspan="5">Singleton</td><td>Decorator</td><td>Mediator</td></tr>
<tr><td>Facade</td><td>Memento</td></tr>
<tr><td>Flyweight</td><td>Observer</td></tr>
<tr><td rowspan="3">Proxy</td><td>State</td></tr>
<tr><td>Strategy</td></tr>
<tr><td>Visitor</td></tr>
</table>

Typologie

Typologie	Nom du Design Pattern	Ce qui doit être ajuster	verbe	composition sous jacente	mélange de classes ?
Creational	Abstract Factory	famille d'objets dépendants		structure	non
	Builder	Comment créer un objet composite dont la structure du composite est indépendante		liste	non
	Factory Method	Sous classe d'un objet qui est instanciée sans connaitre la classe ancêtre (connaissance retardée)			filtrage vtab
	Prototype	Classe d'objet qui est instanciée grâce à un constructeur de copie	copie=verbe	liste	non
	Singleton	La seule instance d'une classe	copie=o=verbe		non
Structural	Adapter	accède à un objet en modifiant l'interface			non
	Bridge	Fait l'implémentation d'un objet par découplage	découplage		non
	Composite	structure et composition d'un objet vue de manière uniforme		arbre	non
	Decorator	Responsabilité d'un objet sans héritage - ajout dynamique			filtrage vtab
	Facade	Exposer une interface à un sous-système			filtrage vtab
	Flyweight	cout de stockage d'un objet, partage de l'état	état=verbe	liste	non
	Proxy	Comment un objet est accédé, son emplacement (mémoire, disque) - effet miroir		queue	non
Behavioral	Chain of Responsibility	Un objet qui peut répondre à une demande avec découpplage	découplage	queue	filtrage vtab
	Command	quand et comment une commande peut être faite - la commande devient un objet		structure	non
	Interpreter	grammaire et interprétation d'un objet			non
	Iterator	se déplacer dans une structure d'objet sans en connaitre le détail		liste	filtrage vtab
	Mediator	Comment et avec quels objets sont décrites les interactions			non
	Memento	Quelles sont les informations privées qui sont stockée à part et quand?	état=verbe	état (queue=infinie)	non
	Observer	l'effectif des objets observés et quand s'effectue la mise à jour		queue	non
	State	les états d'un objet sont des variables, avec un handler()	état=verbe	structure	filtrage vtab
	Strategy	un algorithme	extension=verbe	structure	filtrage vtab
	Template Method	les étapes/squelette d'un algorithme		structure	non
	Visitor	les opérations élémentaires sont appliquées à un objet sans modifier sa classe		liste	non

Réemploi des objets

Spécifier la classe d'un objet explicitement	Abstract Factory
	Factory Method
	Prototype
Coder explicitement les comportements	Command
	Chain of Responsibility
Prise en compte des dépendances	Abstract Factory
	Bridge
Dépendance sur les représentations et les objets	Abstract Factory
	Memento
	Bridge
	Proxy
Dépendance algorithmique	Strategy
	Builder
	Iterator
	Template Method
	Visitor
Couplage léger	Facade
	Mediator
	Observer
	Command
	Abstract Factory
	Bridge
Sous-classe pour étendre le comportement	Bridge
	Composite
	Decorator
	Chain of Responsibility
	Strategy
Incapacité à altérer la classe ancêtre	Visitor
	Decorator
	Adapter

Dans notre étude : le réemploi

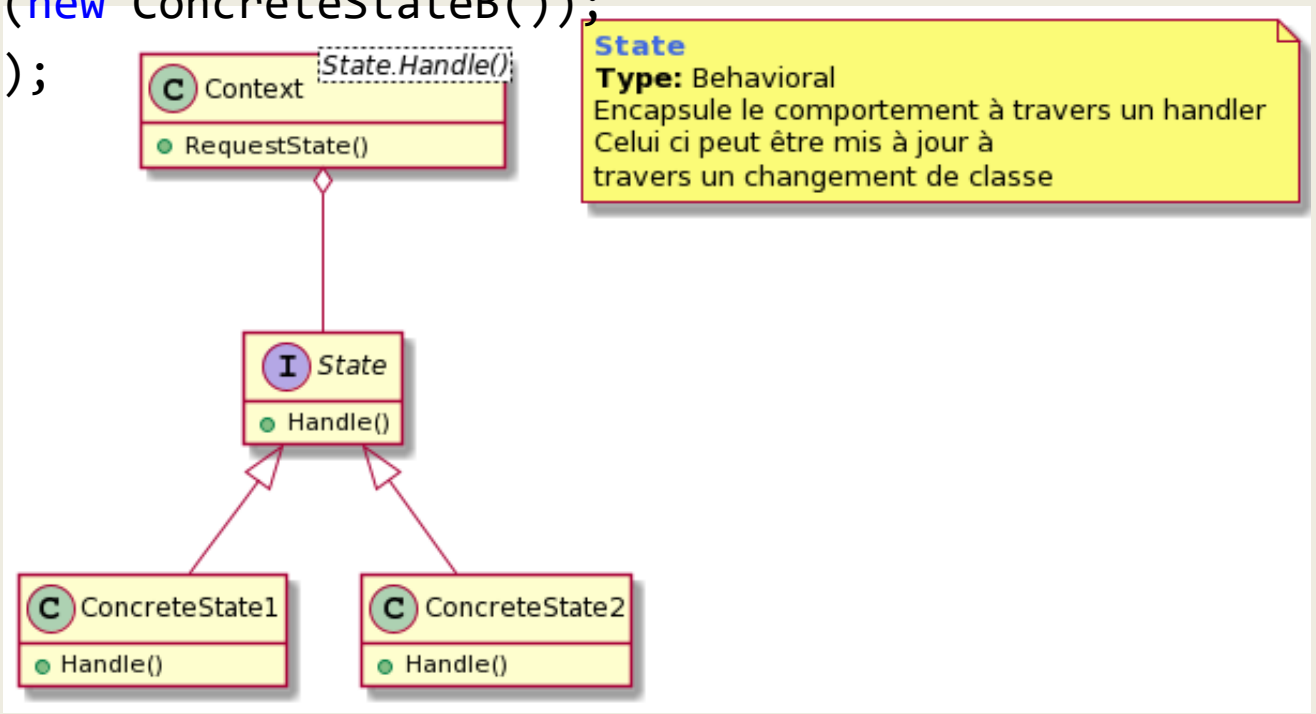
Spécifier la classe d'un objet explicitement	Abstract Factory
	Factory Method
	Prototype
Coder explicitement les comportements	Command
	Chain of Responsibility
Prise en compte des dépendances	Abstract Factory
	Bridge
Dépendance sur les représentations et les objets	Abstract Factory
	Memento
	Bridge
	Proxy
Dépendance algorithmique	Strategy
	Builder
	Iterator
	Template Method
	Visitor
Couplage léger	Facade
	Mediator
	Observer
	Command
	Abstract Factory
	Bridge
Sous-classe pour étendre le comportement	Bridge
	Composite
	Decorator
	Chain of Responsibility
	Strategy
Incapacité à alterer la classe ancêtre	Visitor
	Decorator
	Adapter

Typologie	Nom du Design Pattern	Ce qui doit être ajusté	verbe	composition sous jacente	mélange de classes dynamique ?
Creational	Abstract Factory	famille d'objets dépendants		structure	non
	Builder	Comment créer un objet composite dont la structure du composite est indépendante		liste	non
	Factory Method	Sous classe d'un objet qui est instanciée sans connaitre la classe ancêtre (connaissance retardée)			filtrage vtab
	Prototype	Classe d'objet qui est instanciée grâce à un constructeur de copie	copie=verbe	liste	non
	Singleton	La seule instance d'une classe	copie=o=verbe		non
Structural	Adapter	accède à un objet en modifiant l'interface			non
	Bridge	Fait l'implémentation d'un objet par découplage	découplage		non
	Composite	structure et composition d'un objet vue de manière uniforme		arbre	non
	Decorator	Responsabilité d'un objet sans héritage - ajout dynamique			filtrage vtab
	Facade	Exposer une interface à un sous-système			filtrage vtab
	Flyweight	cout de stockage d'un objet, partage de l'état	état=verbe	liste	non
	Proxy	Comment un objet est accédé, son emplacement (mémoire, disque) - effet miroir		queue	non
Behavioral	Chain of Responsibility	Un objet qui peut répondre à une demande avec découpplage	découplage	queue	filtrage vtab
	Command	quand et comment une commande peut être faite - la commande devient un objet		structure	non
	Interpreter	grammaire et interprétation d'un objet			non
	Iterator	se déplacer dans une structure d'objet sans en connaitre le détail		liste	filtrage vtab
	Mediator	Comment et avec quels objets sont décrites les interactions			non
	Memento	Quelles sont les informations privées qui sont stockée à part et quand?	état=verbe	état (queue=infinie)	non
	Observer	l'effectif des objets observés et quand s'effectue la mise à jour		queue	non
	State	les états d'un objet sont des variables	état=verbe	structure	filtrage vtab
	Strategy	un algorithme indépendant du client	extension=verbe	structure	filtrage vtab
	Template Method	les étapes/squelette d'un algorithme		structure	non
	Visitor	les opérations élémentaires sont appliquées à un objet sans modifier sa classe		liste	non

Behavioral / State

https://godbolt.org/z/TxojreTaY

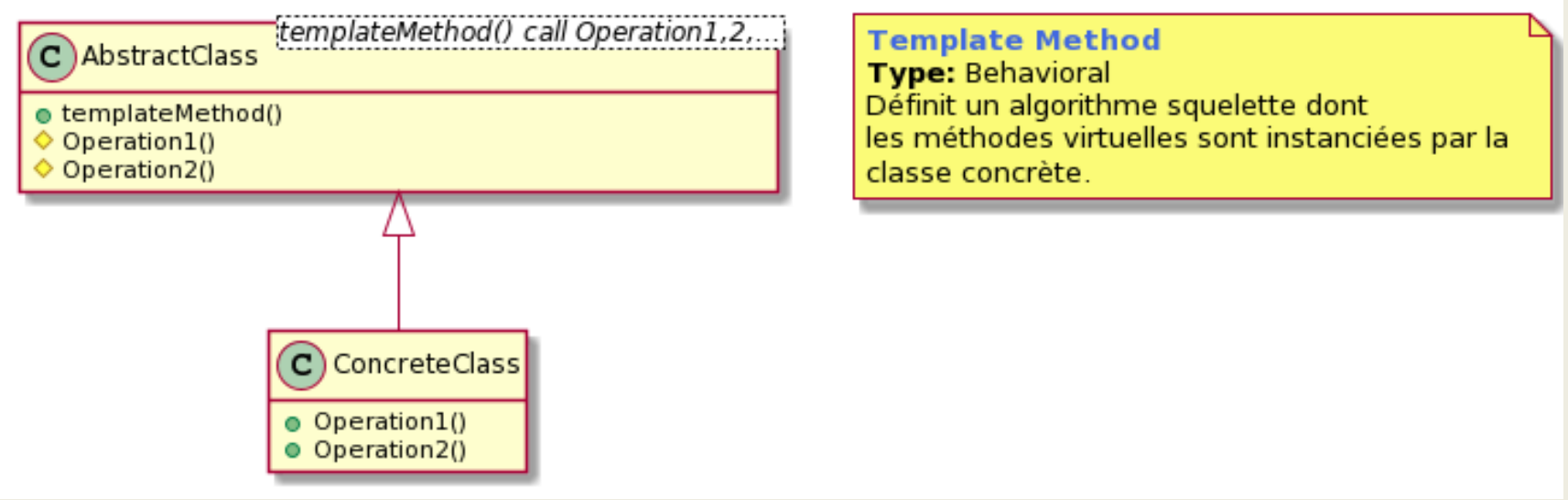
```
int main() {
    Context* context = new Context();

    context->setState(new ConcreteStateA());
    context->Request();

    context->setState(new ConcreteStateB());
    context->Request();
}
```



Behavioral / Template Method

https://godbolt.org/z/8vhM5PaoG

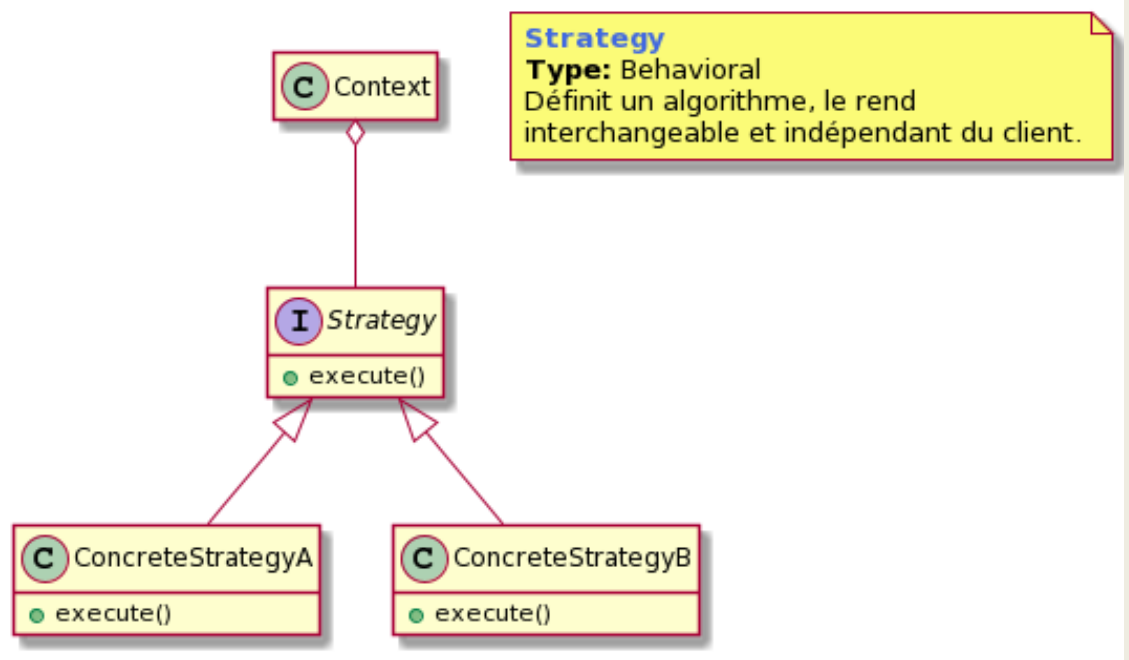
```
int main() {
  AbstractClass *templateClass = new ConcreteClass();
  templateClass->templateMethod();
}
```



Behavioral / Strategy

https://godbolt.org/z/zbaK3Wr5T

```
int main() {
    Context context(new ConcreteStrategyB());
    context.execute();
}
```

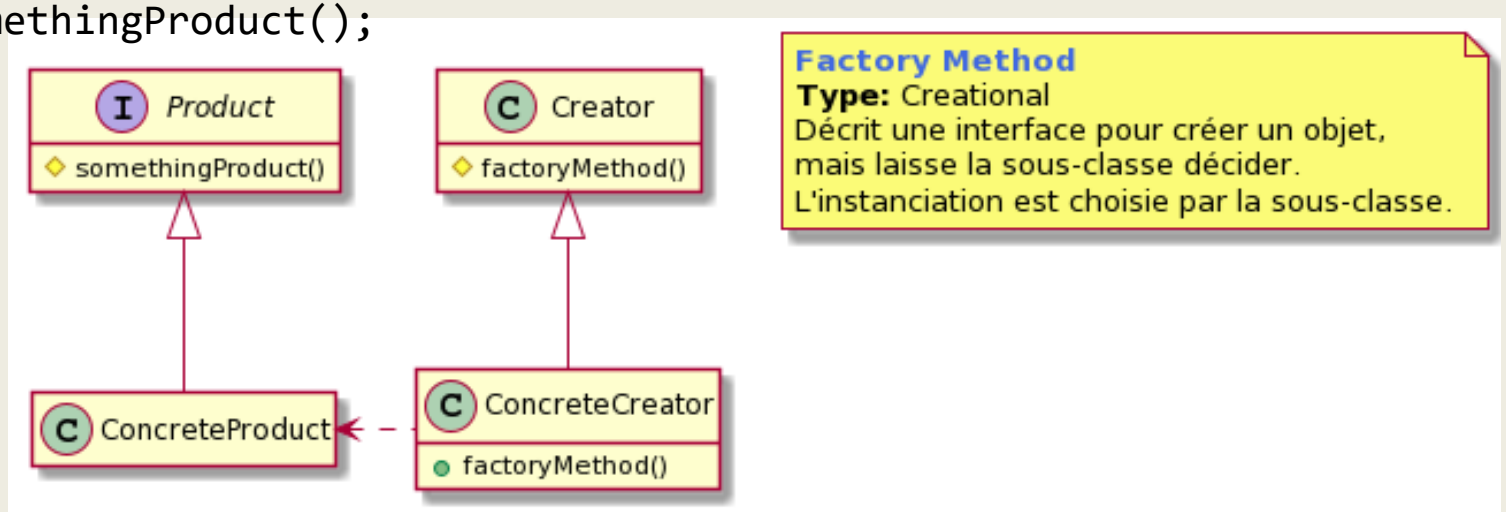


Creational / Factory Method

Permettre à une classe de créer des objets dont elle ne connaît pas la classe

https://godbolt.org/z/7WPYfzvsa

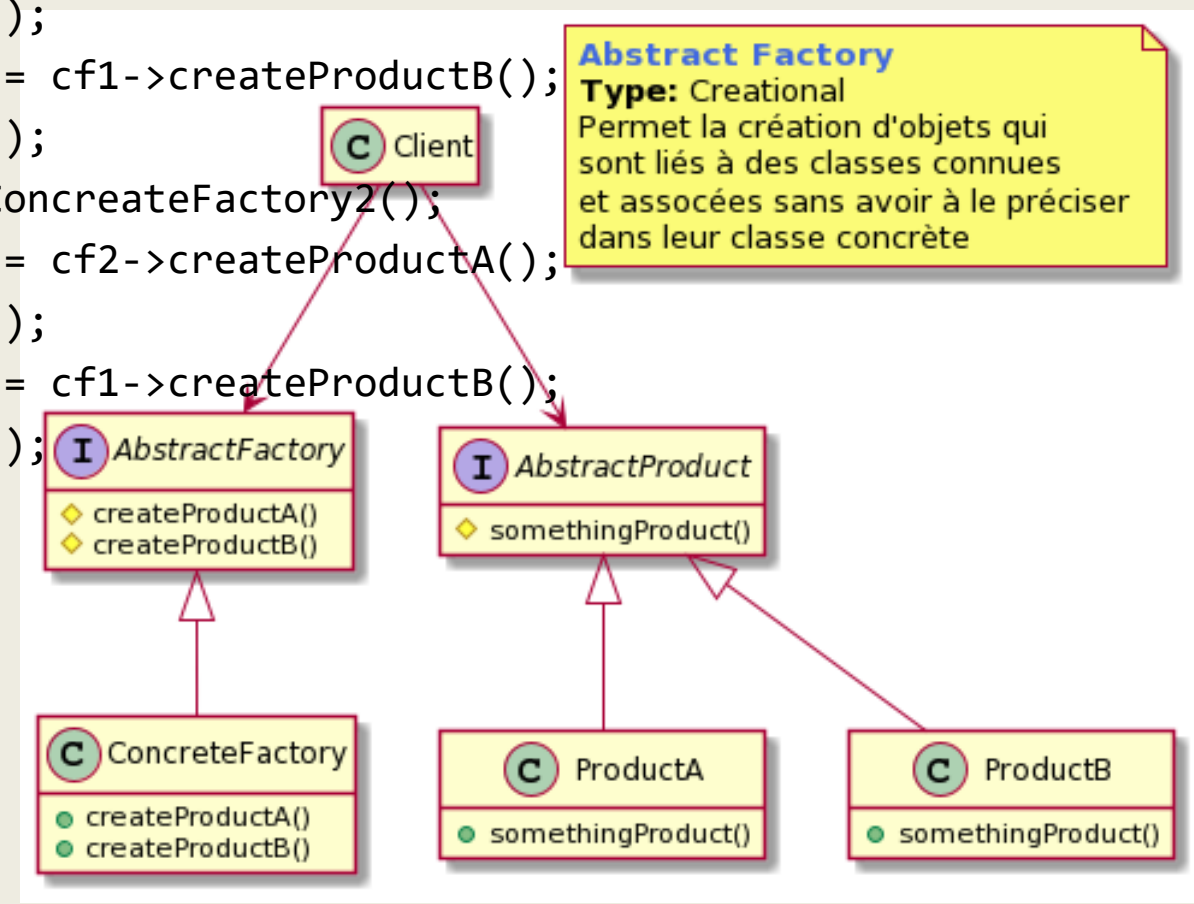
```
int main(int argc, char* argv[]) {
  Creator *creator = new ConcreateCreator(ConcreateCreator::A);

     Product *productA = creator->factoryMethod();
  productA->somethingProduct();

     Product *productB = creator->factoryMethod();
  productB->somethingProduct();

   }
```



Creational / Abstract Factory

https://godbolt.org/z/hh5se4o7j

```
int main(int argc, char* argv[]) {
  AbstractFactory* cf1 = new ConcreateFactory1();
  AbstractProductA* productA1 = cf1->createProductA();
  productA1->somethingProduct();
  AbstractProductB* productB1 = cf1->createProductB();
  productB1->somethingProduct();
  AbstractFactory* cf2 = new ConcreateFactory2();
  AbstractProductA* productA2 = cf2->createProductA();
  productA2->somethingProduct();
  AbstractProductB* productB2 = cf1->createProductB();
  productB2->somethingProduct();
   }
```

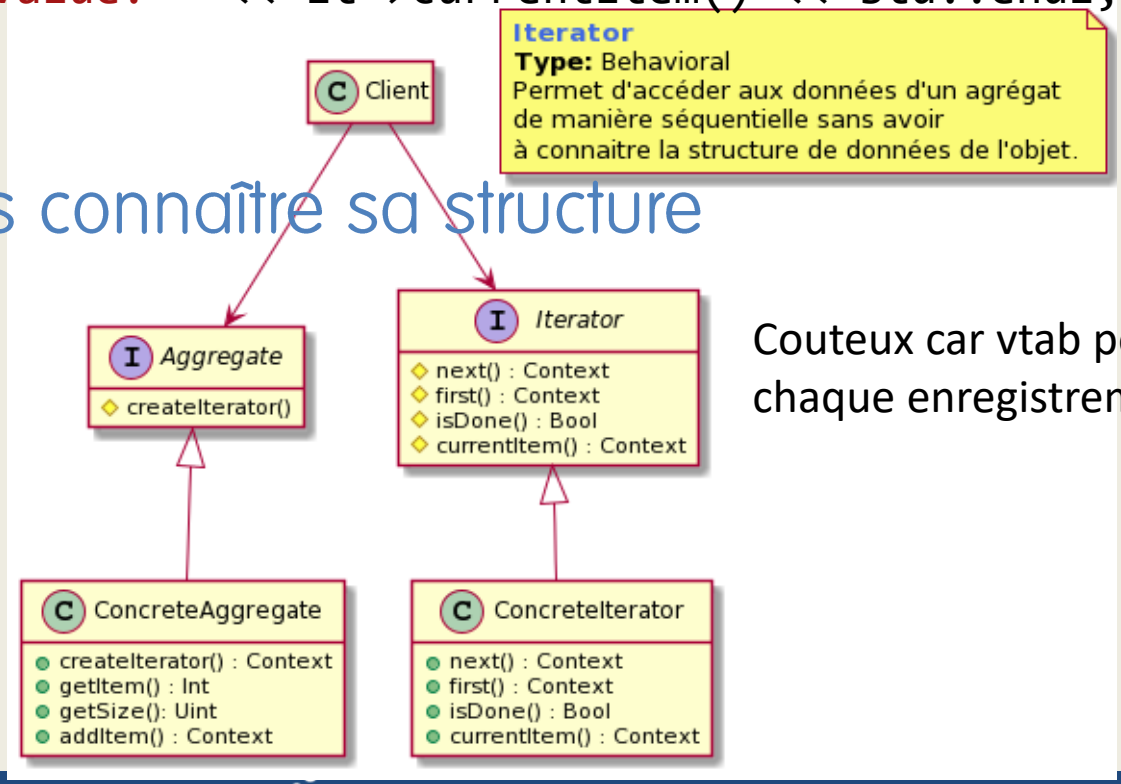


Behavioral / Iterator

https://godbolt.org/z/bqxMrsr6K

```
int main() {
  ConcreteAggregate *concreteAggregate = new ConcreteAggregate(10);
  concreteAggregate->addItem(123);
     Iterator* it = concreteAggregate->createIterator();
  for ( ; !it->isDone(); it->next()) {
    std::cout << "Item value: " << it->currentItem() << std::endl;
  }
 }
```

Visiter un objet sans connaître sa structure

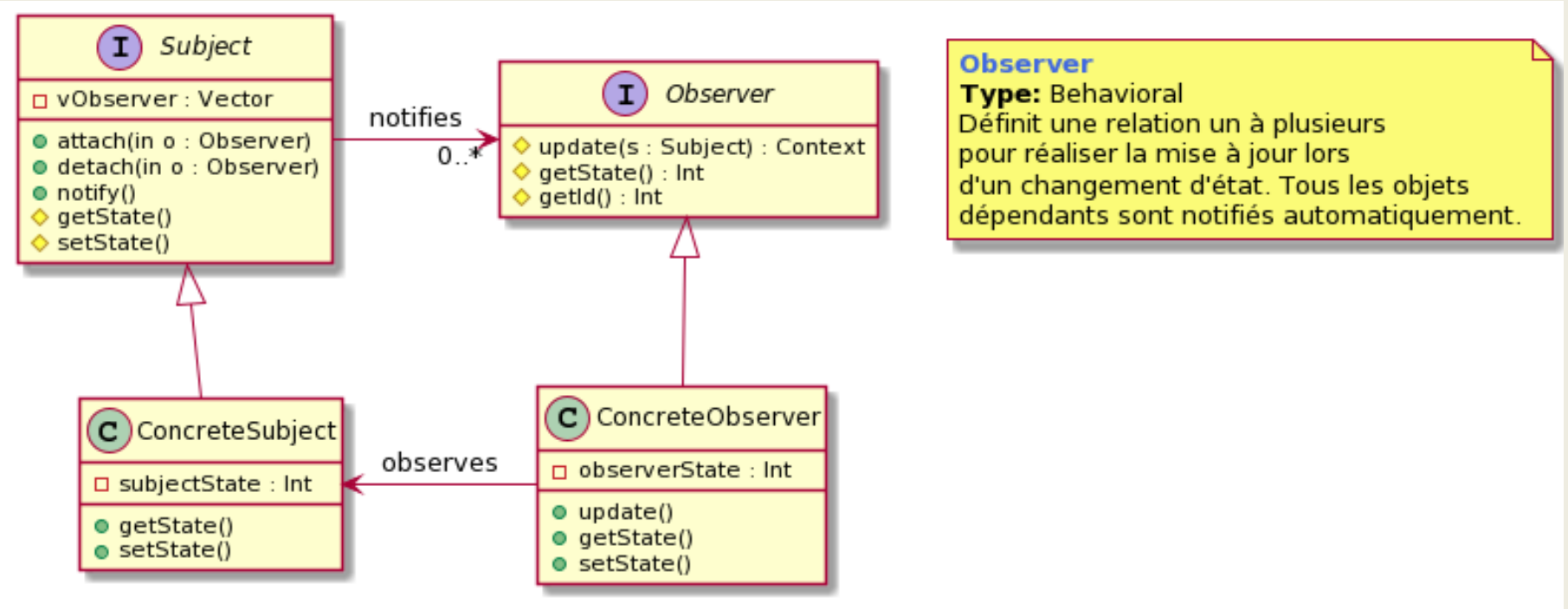


Couteux car vtab pour chaque enregistrement

Behavioral / Observer

https://godbolt.org/z/qMKrsoY7M

Dépendance Publish-Subscribe



```
int main() {
    ConcreteObserver observer1(1000, 1);
    ConcreteObserver observer2(2000, 2);
    std::cout << "Observer1 state: " << observer1.getState() << std::endl;
    std::cout << "Observer2 state: " << observer2.getState() << std::endl;

    Subject* subject = new ConcreteSubject();
    subject->attach(&observer1);
    subject->attach(&observer2);

    subject->setState(10);
    subject->notify();

    std::cout << "Observer1 state: " << observer1.getState() << std::endl;
    std::cout << "Observer2 state: " << observer2.getState() << std::endl;

    subject->detach(1);
    subject->setState(100);
    subject->notify();

    std::cout << "Observer1 state: " << observer1.getState() << std::endl;
    std::cout << "Observer2 state: " << observer2.getState() << std::endl;
}
```