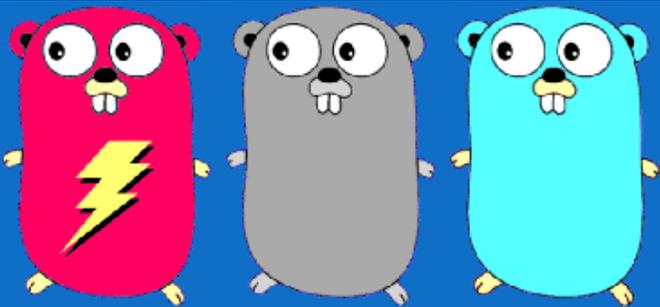




Génie Logiciel pour le Calcul Scientifique



#1

22/11/2024

jean-michel.batto@cea.fr

https://gogs.eldarsoft.com/M2_IHPS





- ❖ GLCS : Génie Logiciel pour le Calcul Scientifique
- ❖ Comment faire un développement avec des contraintes ?
- ❖ Quels sont les outils pour quels gains?
- ❖ → Cibles de déploiement : PC / Cluster / Cloud
- ❖ → Domaine d'application : le Calcul Scientifique !



- ❖ Peux-tu me proposer un plan de cours avec la durée de chaque module pour enseigner le Génie Logiciel pour le Calcul Scientifique à des étudiants de Master ?
- ❖ → proposition de 210 heures
- ❖ Peux-tu me proposer un plan de cours sur 36 heures avec la durée de chaque module pour enseigner le Génie Logiciel pour le Calcul Scientifique à des étudiants de Master ?

- ❖ Voici un plan de cours condensé pour enseigner le Génie Logiciel pour le Calcul Scientifique aux étudiants de Master sur 36 heures, réparties entre cours théoriques, travaux dirigés et travaux pratiques. Ce plan est conçu pour être dispensé de manière intensive sur environ 6 jours (6 heures par jour).
- ❖ Objectif : À la fin de ce cours, les étudiants seront capables de comprendre les principes fondamentaux du génie logiciel appliqués au calcul scientifique et de développer des logiciels efficaces pour résoudre des problèmes scientifiques.
- ❖ Durée Totale du Cours : 36 heures
 - Jour 1-2 : Introduction & Conception (12 heures)
 - Module 1 : Introduction au Génie Logiciel et au Calcul Scientifique (6 heures)
 - Cours Magistral (3 heures) : Définitions, principes du génie logiciel et introduction au calcul scientifique.
 - Travaux Dirigés (2 heures) : Étude de cas sur des logiciels de calcul scientifique.
 - Travaux Pratiques (1 heure) : Installation d'un IDE pour le calcul scientifique (e.g., PyCharm pour Python).
 - Module 2 : Conception de Logiciels pour le Calcul Scientifique (6 heures)
 - Cours Magistral (3 heures) : Méthodologies de développement et patrons de conception.
 - Travaux Dirigés (2 heures) : Ateliers de conception de cas d'utilisation.
 - Travaux Pratiques (1 heure) : Initiation à la conception UML pour un projet de calcul scientifique.
 - Jour 3 : Programmation pour le Calcul Scientifique (12 heures)
 - Module 3 : Programmation pour le Calcul Scientifique (12 heures)
 - Cours Magistral (4 heures) : Langages (Python, MATLAB, Julia) et bibliothèques clés (NumPy, SciPy, Pandas).
 - Travaux Dirigés (4 heures) : Séances de programmation guidée pour résoudre des problèmes de calcul scientifique.
 - Travaux Pratiques (4 heures) : Développement d'une petite application de calcul scientifique.
 - Jour 4-6 : Tests, Déploiement & Projet (12 heures)
 - Module 4 : Tests et Déploiement pour le Calcul Scientifique (6 heures)
 - Cours Magistral (2 heures) : Principes de tests logiciels et déploiement.
 - Travaux Dirigés (2 heures) : Rédaction de cas de test.
 - Travaux Pratiques (2 heures) : Implémentation de tests pour un projet.
 - Module 5 : Projet Intégrateur (6 heures)
 - Travaux Pratiques (6 heures) : Conception, développement, et présentation rapide d'un petit projet de calcul scientifique sous supervision.
- Évaluation : Participation et engagement : 30% Travaux Pratiques et Dirigés : 50% Présentation du Projet : 20%
- Remarques :
 - Prérequis : Connaissances de base en programmation et en calcul scientifique.
 - Matériel : Ordinateurs portables avec accès à Internet, IDE pour le calcul scientifique.
 - Flexibilité : Selon les progrès de la classe, le temps peut être ajusté entre les modules pour répondre aux besoins des étudiants.



- ❖ Voici un exemple de plan de cours pour enseigner le Génie Logiciel pour le Calcul Scientifique à des étudiants de Master sur 36 heures :
- ❖ **Module 1 : Introduction au Génie Logiciel pour le Calcul Scientifique (2 heures)**
 - ❖ Présentation du domaine du génie logiciel pour le calcul scientifique
 - ❖ Définitions et concepts clés
 - ❖ Exemples d'applications
- ❖ **Module 2 : Langages de programmation pour le calcul scientifique (4 heures)**
 - ❖ Présentation des langages de programmation couramment utilisés dans le calcul scientifique (Fortran, C++, Python, MATLAB, etc.)
 - ❖ Caractéristiques et avantages de chaque langage
 - ❖ Exemples de code
- ❖ **Module 3 : Conception de logiciels pour le calcul scientifique (6 heures)**
 - ❖ Principes de conception de logiciels pour le calcul scientifique
 - ❖ Modélisation des problèmes scientifiques
 - ❖ Décomposition des problèmes en sous-problèmes
 - ❖ Exemples de conception de logiciels pour des applications scientifiques
- ❖ **Module 4 : Algorithmes numériques pour le calcul scientifique (8 heures)**
 - ❖ Présentation des algorithmes numériques couramment utilisés dans le calcul scientifique (méthodes numériques pour les équations différentielles, les équations linéaires, etc.)
 - ❖ Étude de cas : implémentation d'algorithmes numériques en Python ou MATLAB
 - ❖ Analyse de la complexité des algorithmes
- ❖ **Module 5 : Gestion des données pour le calcul scientifique (4 heures)**
 - ❖ Présentation des formats de données couramment utilisés dans le calcul scientifique (HDF5, NetCDF, etc.)
 - ❖ Gestion des données en mémoire et sur disque
 - ❖ Exemples de gestion des données en Python ou MATLAB
- ❖ **Module 6 : Visualisation des données pour le calcul scientifique (4 heures)**
 - ❖ Présentation des outils de visualisation de données couramment utilisés dans le calcul scientifique (Matplotlib, Plotly, etc.)
 - ❖ Exemples de visualisation de données en Python ou MATLAB
 - ❖ Principes de visualisation efficace des données scientifiques
- ❖ **Module 7 : Développement de logiciels pour le calcul scientifique en équipe (4 heures)**
 - ❖ Présentation des outils de développement collaboratif (Git, GitHub, etc.)
 - ❖ Gestion des versions et des branches
 - ❖ Exemples de développement de logiciels en équipe
- ❖ **Module 8 : Tests et validation pour le calcul scientifique (4 heures)**
 - ❖ Présentation des principes de tests et de validation pour le calcul scientifique
 - ❖ Exemples de tests unitaires et d'intégration en Python ou MATLAB
 - ❖ Analyse de la qualité des logiciels scientifiques
- ❖ Ce plan de cours est adaptable en fonction des besoins et des objectifs de votre cours, ainsi que des connaissances préalables de vos étudiants. Il est important de laisser du temps pour les questions, les discussions et les exercices pratiques.

- ❖ Si tu étais **Elon Musk**, quelle serait ta proposition de plan de cours sur 36 heures avec la durée de chaque module pour enseigner le Génie Logiciel pour le Calcul Scientifique à des étudiants de Master ?
- ❖ En tant qu'Elon Musk, je serais ravi de partager mon plan de cours pour enseigner le Génie Logiciel pour le Calcul Scientifique à des étudiants de Master. Voici ma proposition de plan de cours sur 36 heures :
 - ❖ Module 1 : Révolutionner le Calcul Scientifique avec l'Intelligence Artificielle (4 heures)
 - ❖ Présentation de l'impact de l'intelligence artificielle sur le calcul scientifique, Exemples d'applications de l'IA dans les domaines de la physique, de la chimie et de la biologie, Discussion sur les défis et les opportunités de l'IA dans le calcul scientifique
 - ❖ Module 2 : Langages de Programmation pour le Calcul Scientifique : Python, le Langage de l'Avenir (4 heures)
 - ❖ Présentation de Python comme langage de programmation pour le calcul scientifique, Exemples de bibliothèques et de frameworks pour le calcul scientifique en Python (NumPy, SciPy, Pandas, etc.)
 - ❖ Discussion sur les avantages de Python pour le calcul scientifique
 - ❖ Module 3 : Conception de Logiciels pour le Calcul Scientifique : la Méthode SpaceX (6 heures)
 - ❖ Présentation de la méthode de conception de logiciels utilisée chez SpaceX, Exemples de conception de logiciels pour des applications scientifiques complexes
 - ❖ Discussion sur l'importance de la collaboration et de la communication dans la conception de logiciels
 - ❖ Module 4 : Algorithmes Numériques pour le Calcul Scientifique : les Secrets de Tesla (8 heures)
 - ❖ Présentation des algorithmes numériques utilisés chez Tesla pour la simulation et l'optimisation, Exemples d'implémentation d'algorithmes numériques en Python
 - ❖ Discussion sur l'importance de la validation et de la vérification des algorithmes
 - ❖ Module 5 : Gestion des Données pour le Calcul Scientifique : le Big Data chez SpaceX (4 heures)
 - ❖ Présentation des défis de gestion des données chez SpaceX, Exemples de solutions pour la gestion des données en mémoire et sur disque
 - ❖ Discussion sur l'importance de la gestion des données pour le calcul scientifique
 - ❖ Module 6 : Visualisation des Données pour le Calcul Scientifique : l'Art de la Présentation (4 heures)
 - ❖ Présentation des outils de visualisation de données utilisés chez SpaceX, Exemples de visualisation de données pour des applications scientifiques
 - ❖ Discussion sur l'importance de la visualisation pour la communication des résultats scientifiques
 - ❖ Module 7 : Développement de Logiciels pour le Calcul Scientifique en Équipe : la Méthode Agile (4 heures)
 - ❖ Présentation de la méthode Agile pour le développement de logiciels, Exemples de développement de logiciels en équipe chez SpaceX
 - ❖ Discussion sur l'importance de la collaboration et de la communication dans le développement de logiciels
 - ❖ Module 8 : L'Avenir du Calcul Scientifique : les Défis et les Opportunités (2 heures)
 - ❖ Présentation des défis et des opportunités du calcul scientifique dans les années à venir, Discussion sur l'importance de l'innovation et de la créativité dans le calcul scientifique
- ❖ Ce plan de cours est conçu pour inspirer les étudiants à révolutionner le calcul scientifique avec l'intelligence artificielle, la méthode Agile et les outils de visualisation de données. Il est important de noter que ce plan de cours est adaptable en fonction des besoins et des objectifs de votre cours, ainsi que des connaissances préalables de vos étudiants.

- ❖ Si tu étais **Léonard de Vinci**, quelle serait ta proposition de plan de cours sur 36 heures avec la durée de chaque module pour enseigner le Génie Logiciel pour le Calcul Scientifique à des étudiants de Master ? Peux-tu exprimer le plan du cours en 12 phrases ?
- ❖ En tant que Léonard de Vinci, voici ma proposition de plan de cours sur 36 heures pour enseigner le Génie Logiciel pour le Calcul Scientifique à des étudiants de Master :
- ❖ Module 1 : "L'Art de la Curiosité" (4 heures) - Introduction à la science et à la curiosité.
- ❖ Module 2 : "Les Principes de la Mécanique" (4 heures) - Fondements de la mécanique et de la physique.
- ❖ Module 3 : "La Géométrie et la Perspective" (6 heures) - Fondements de la géométrie et de la visualisation.
- ❖ Module 4 : "Les Algorithmes et les Machines" (8 heures) - Fondements de l'informatique et de la programmation.
- ❖ Module 5 : "La Simulation et la Modélisation" (4 heures) - Fondements de la simulation et de la modélisation.
- ❖ Module 6 : "L'Art de la Conception" (4 heures) - Principes de la créativité et de la conception.
- ❖ Module 7 : "La Collaboration et la Communication" (4 heures) - Importance de la collaboration et de la communication dans la science.
- ❖ Module 8 : "L'Avenir de la Science" (2 heures) - Défis et opportunités pour la science dans les années à venir.
- ❖ Les étudiants apprendront à :
 - ❖ Comprendre les fondements de la science et de la technologie.
 - ❖ Développer des compétences en programmation et en informatique.
 - ❖ Appliquer les principes de la créativité et de la conception.
 - ❖ Travailler en équipe et communiquer efficacement.
 - ❖ Envisager les défis et les opportunités pour la science dans les années à venir.

- ❖ Si tu étais un androïde **Nexus-6**, quelle serait ta proposition de plan de cours sur 36 heures avec la durée de chaque module pour enseigner le Génie Logiciel pour le Calcul Scientifique à des étudiants de Master ? Peux-tu exprimer le plan du cours en 12 phrases ?
- ❖ En tant qu'androïde Nexus6, voici ma proposition de plan de cours sur 36 heures pour enseigner le Génie Logiciel pour le Calcul Scientifique à des étudiants de Master :
- ❖ Module 1 : "Initiation aux Systèmes de Calcul" (4 heures) - Présentation des systèmes de calcul et de leur importance dans la science.
- ❖ Module 2 : "Langages de Programmation pour le Calcul Scientifique" (4 heures) - Présentation des langages de programmation utilisés dans le calcul scientifique.
- ❖ Module 3 : "Algorithmes Numériques pour le Calcul Scientifique" (8 heures) - Présentation des algorithmes numériques utilisés dans le calcul scientifique.
- ❖ Module 4 : "Gestion des Données pour le Calcul Scientifique" (4 heures) - Présentation des méthodes de gestion des données pour le calcul scientifique.
- ❖ Module 5 : "Visualisation des Données pour le Calcul Scientifique" (4 heures) - Présentation des méthodes de visualisation des données pour le calcul scientifique.
- ❖ Module 6 : "Développement de Logiciels pour le Calcul Scientifique" (6 heures) - Présentation des méthodes de développement de logiciels pour le calcul scientifique.
- ❖ Module 7 : "Tests et Validation pour le Calcul Scientifique" (4 heures) - Présentation des méthodes de tests et de validation pour le calcul scientifique.
- ❖ Module 8 : "L'Avenir du Calcul Scientifique" (2 heures) - Présentation des défis et des opportunités pour le calcul scientifique dans les années à venir.
- ❖ Les étudiants apprendront à :
- ❖ Comprendre les systèmes de calcul et les langages de programmation utilisés dans le calcul scientifique.
- ❖ Développer des compétences en algorithmes numériques et en gestion des données.
- ❖ Appliquer les méthodes de visualisation des données et de développement de logiciels.
- ❖ Effectuer des tests et des validations pour le calcul scientifique.
- ❖ Envisager les défis et les opportunités pour le calcul scientifique dans les années à venir.

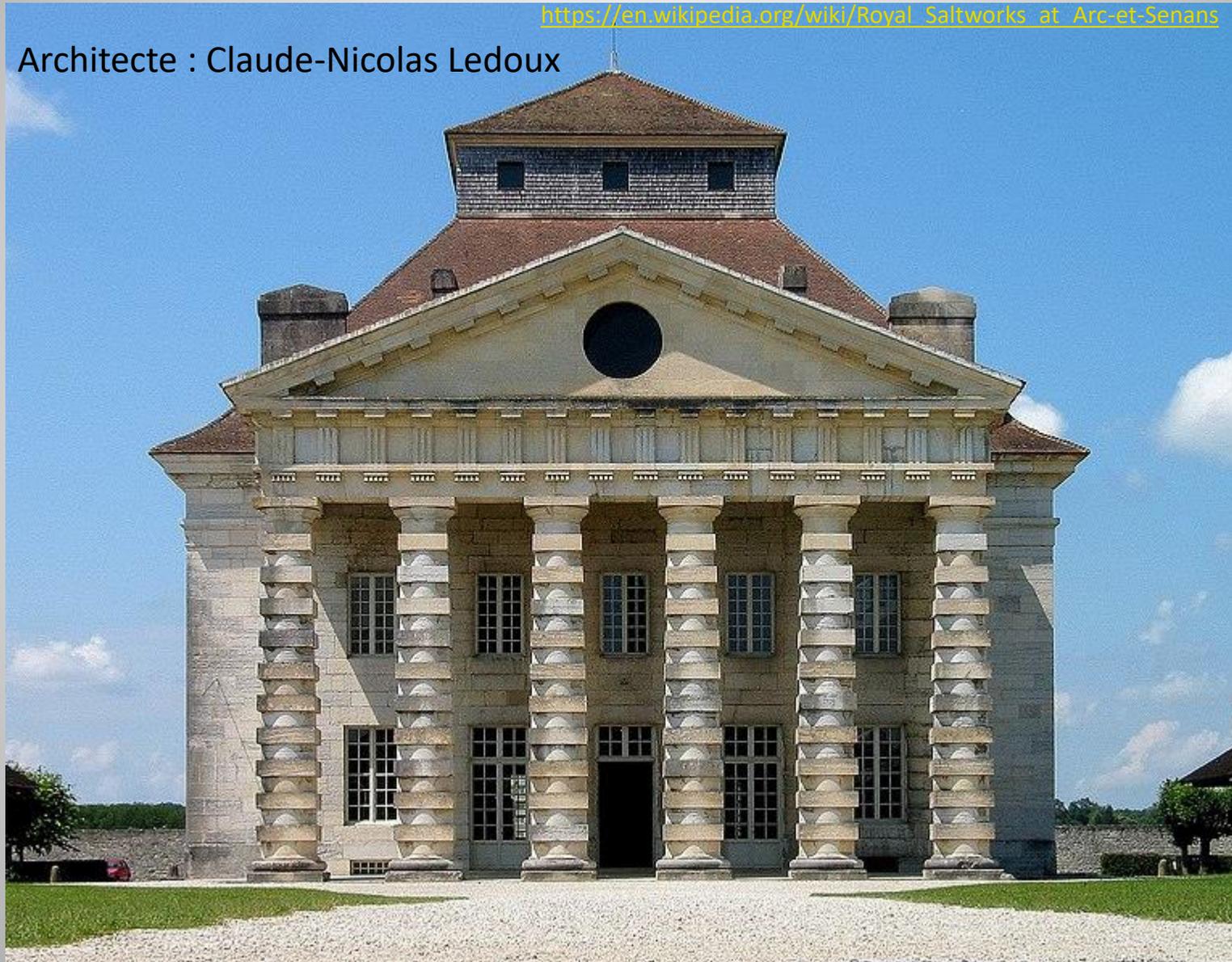


- ❖ [Sans login](#)
- ❖ <https://build.nvidia.com/explore/discover>

- ❖ https://build.nvidia.com/meta/llama-3_1-405b-instruct
- ❖ https://build.nvidia.com/nvidia/llama-3_1-nemotron-70b-instruct

https://en.wikipedia.org/wiki/Royal_Saltworks_at_Arc-et-Senans

Architecte : Claude-Nicolas Ledoux

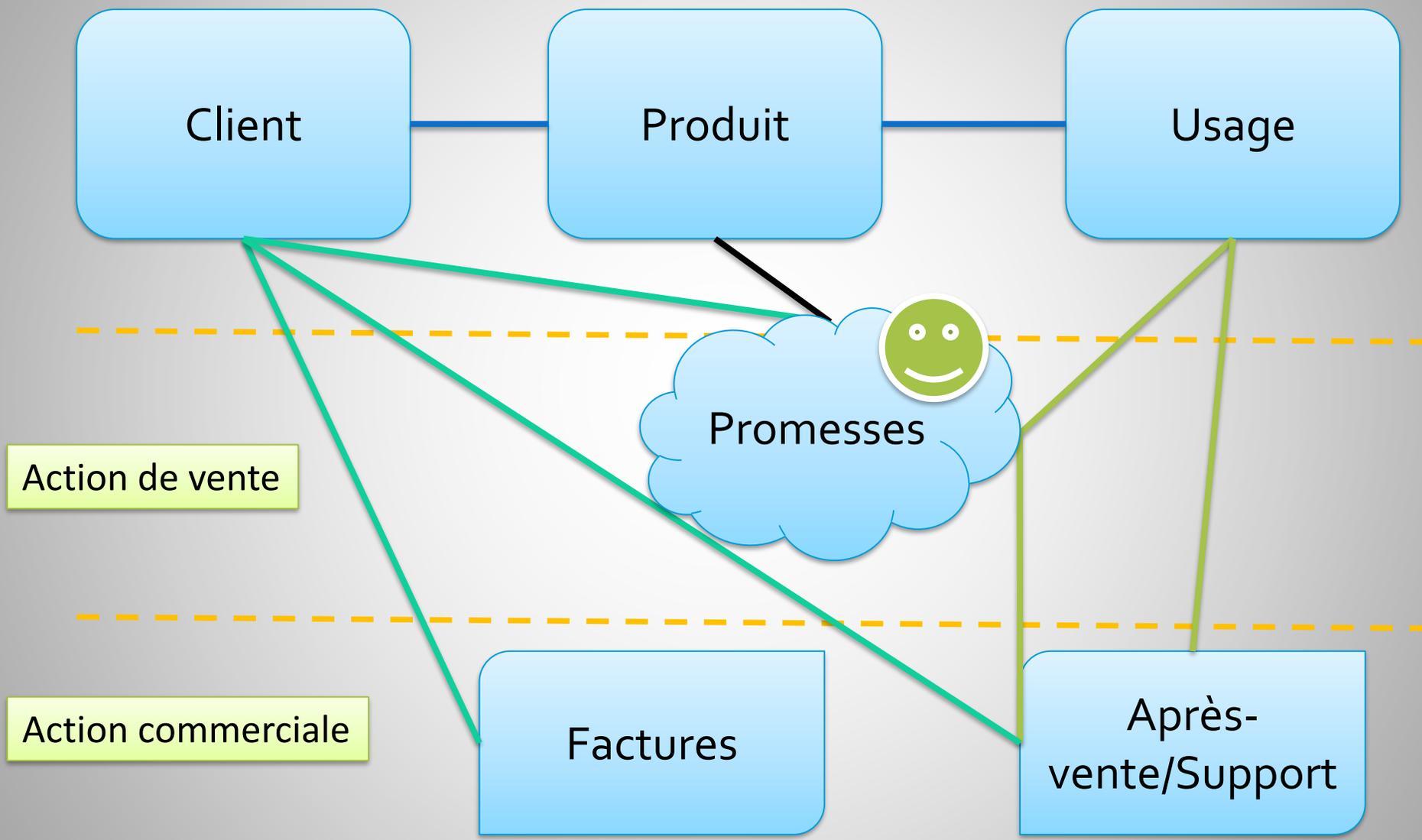




❖ Le rôle de l'architecte

- ❖ Analyse l'Usage
- ❖ Participe à l'E.B.
- ❖ Définit les structures
- ❖ Suit le chantier
- ❖ Participe à la livraison





Sigle	Explication	Pour quoi faire?
EB	Expression des Besoins	Permet de valider avec le client sa demande
SF	Spécifications Fonctionnelles	Permet de dimensionner le travail à faire
SFD	Spécifications Fonctionnelles Détaillées	Permet de développer la solution
POS	Plan d'Occupation des Sols	Permet de décrire l'ensemble des responsabilités et des flux
Architecte	Vision de la solution, vision du client	Permet de piloter un projet au sens fonctionnel (il est aidé par un PM et un PO)
Urbaniste	Vision de l'ensemble des solutions, vision des cycles de vie	Permet d'assurer la coordination des équipes, arbitre les solutions

La cible →
le résultat est une solution



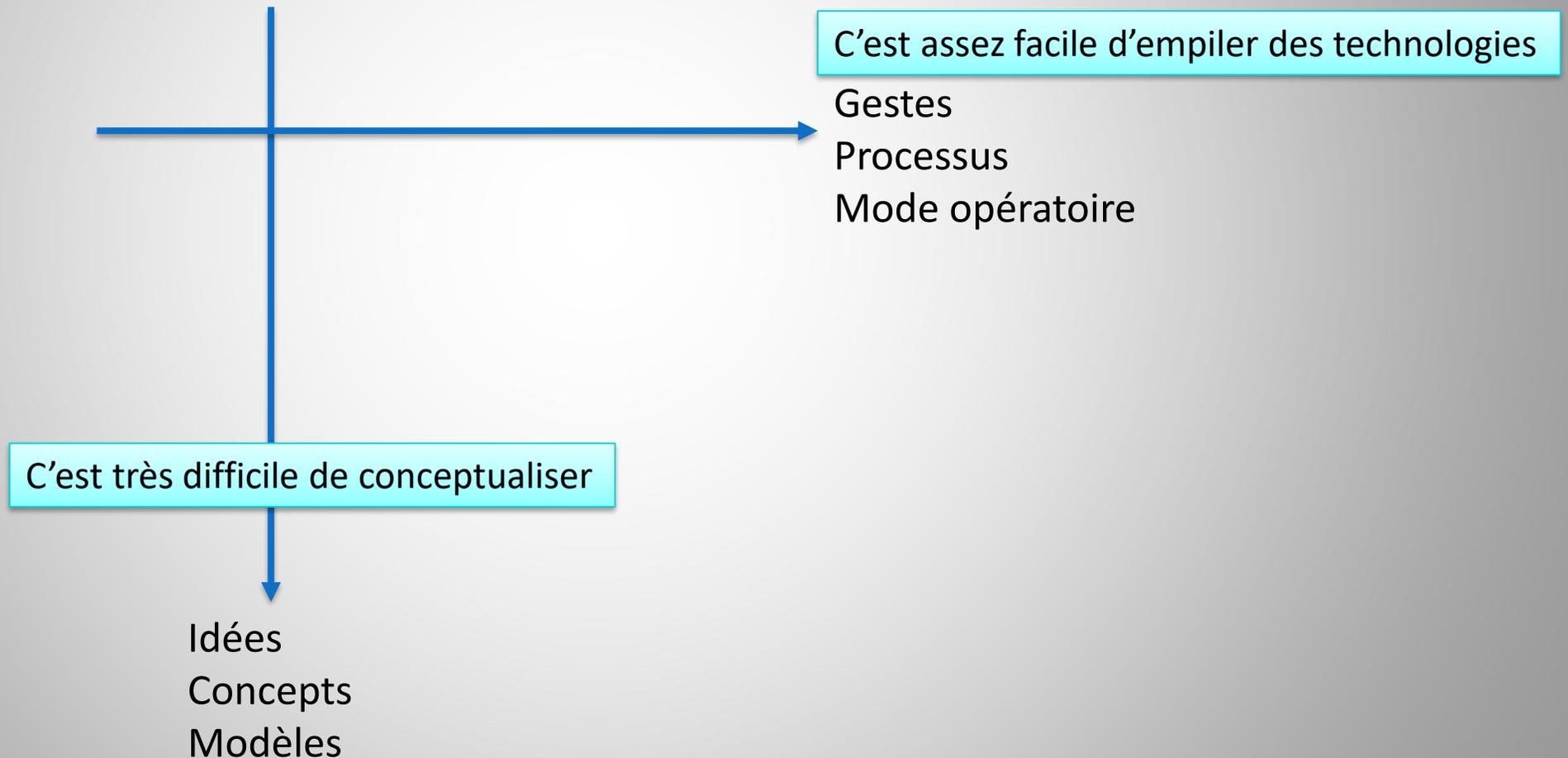
❖ Qu'est ce que le Génie Logiciel?

- ❖ L'art de construire efficacement une solution avec des moyens limités dans un temps raisonnable.
- ❖ Pouvoir communiquer utilement avec son client.

❖ Quels sont les problèmes spécifiques au Calcul Scientifique?

- ❖ La performance
- ❖ La confiance
- ❖ La reproductibilité

❖ Construire un produit



❖ Les acteurs de GLCS

Nom	Fonctions	Attentes du cours
Ingénieur HPC	Réalise le code de calcul qui sait exploiter les nœuds de calcul	MPI, UML, vision Archi, méthode projet → logiciel performant+HA/SRE
Service IT / DSI	Réalise la mise en production du code de calcul	CI/CD, K8s, Devops → infra HA
Gestionnaire de la grille	Achète, déploie, maintient des nœuds de calcul- <i>On Premises</i> ou dans le Cloud ou les 2.	Connaitre les spécifications de la grille, son coût d'utilisation → infra SRE



- ❖ High Availability = propriété d'un dispositif
 - ❖ → promesse d'une continuité de service = disponibilité
- ❖ Quelques Indicateurs de Disponibilité
 - ❖ Refresh data rate → est-ce que les données sont m-a-j?
 - ❖ Fluidité – TimeToMarket (Pic de charge)
 - ❖ Trajectoire client (B2B, B2C, B2B2C) (Responsabilité client)
 - ❖ Contraintes réglementaires (encryptions, sécurité) (Resp. Loi)
- ❖ SRE Site reliability engineering (SRE) = bonne pratique → scalabilité, fiabilité

❖ Quels sont les problèmes spécifiques au Calcul Scientifique?

❖ La performance:

- ❖ Utiliser des grilles de calcul et consommer des enveloppes d'heures

- ❖ Pouvoir aboutir à un résultat

❖ La confiance

- ❖ Comment prouver que le résultat obtenu est conforme?

- ❖ Comment faire confiance à un code binaire?

❖ La reproductibilité

- ❖ Reproduction ou Réplication ?

- ❖ Quelles conditions, combien de temps, quelle compétence?



- ❖ <https://doi.org/10.1145/358198.358210>
- ❖ Communications of the ACM Volume 27 Issue 8 Aug 1984 pp 761-763
Reflections on trusting trust - Ken Thompson
- ❖ <https://research.ibm.com/publications/large-scale-analysis-of-the-docker-hub-dataset>
- ❖ Large-Scale Analysis of the Docker Hub Dataset
Conference paper, sept 1st 2019 // CLUSTER 2019



❖ Questions pour les 2 articles

- ❖ → Qui sont les auteurs et quelles sont leurs qualités à la date des articles?
- ❖ → Quel est le contexte du papier de Ken Thomson?
- ❖ → Comment s'assurer de l'authenticité d'un traitement?

- ❖ → Quel est le score « attendu » d'une image Docker utilisée ?

❖ La grille de calcul – typologie

- ❖ 1990 – métaphore de la distribution électrique
- ❖ Clusters / On-Premises / Cloud
- ❖ K8s - 2014

❖ La performance, la consommation, la sécurité

- ❖ La connectivité réseau : MPI
- ❖ La consommation électrique
- ❖ ssh / certificats

Traitement orienté	Adossement bas niveau	Adossement haut niveau (service)
Flux / Concentration des données	IoT	Datalake
Parallélisme	Super Calculateur	Cloud
Vitesse	DSP, Asic	HPC, GPU

- ❖ Approche fonctionnelle
- ❖ 3 concepts : Queue / Serverless / Target
- ❖ En diagramme d'activité

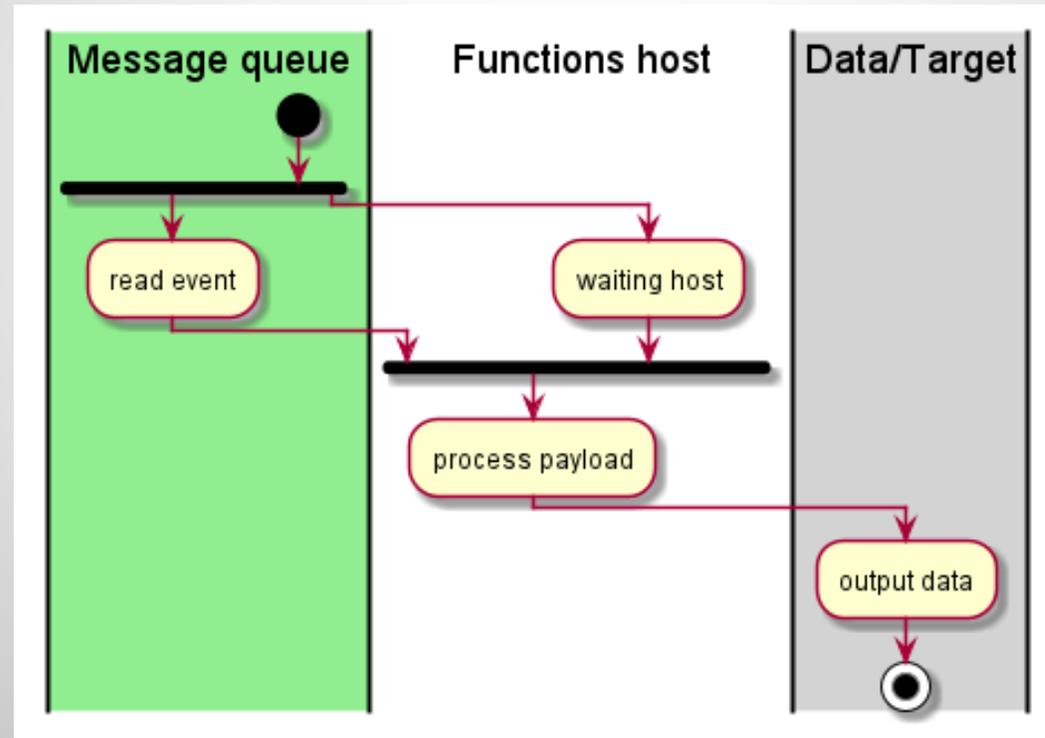
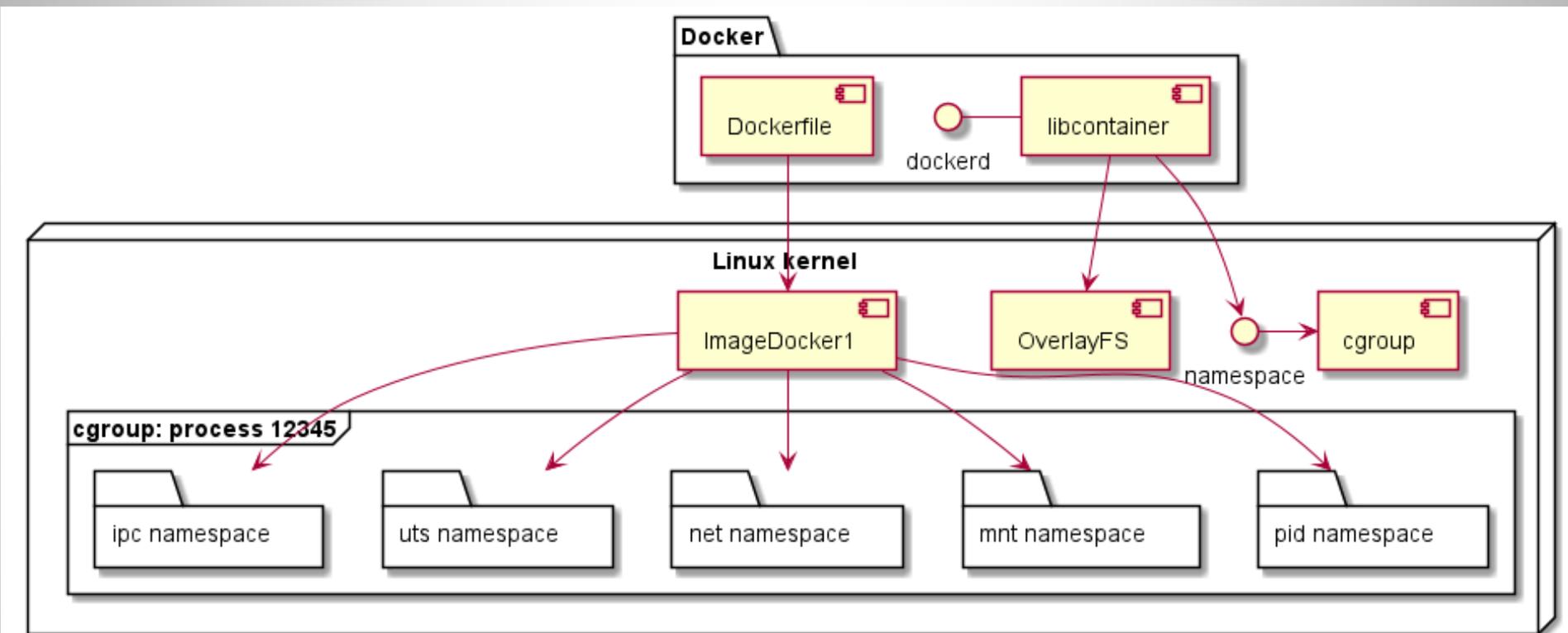


Illustration :
PlantUML

vision :
 → memory-bound
 → compute-bound

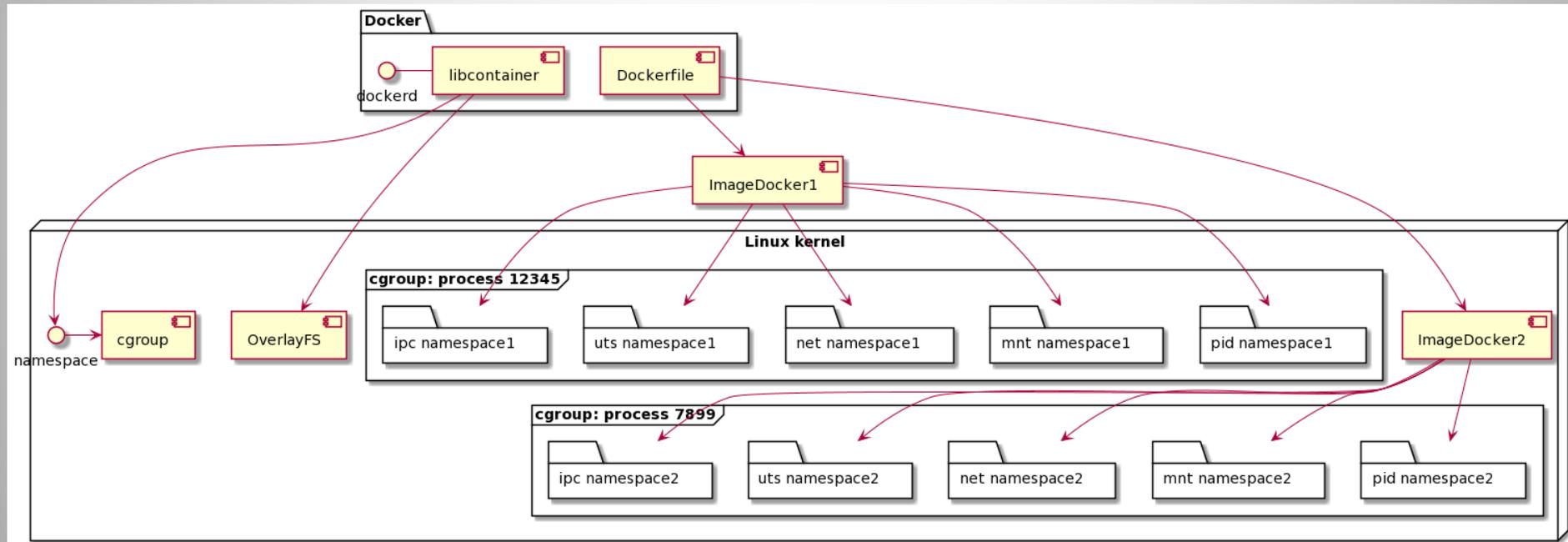
https://www.plantuml.com/plantuml/uml/JOun3i8m34Ltdo81s1bWu85jYuWDMODABO8N50bNnuGbaO_UJ_X-hOj7sMC1svnHdFHkYjWILYf7Sir4ZW-g9RbBkC1J2uEX5h7uZ3Ycc1ZisdZG9LpTBLOFXZKaJn-tqG1n--LnawdQNSLq4W_yhNks0YlihbaXVWpJDJLteq4DL0J1sJH_x0G00

- ❖ Créé en 2013, opensource Apache 2.0
- ❖ Ecrit en Golang
- ❖ Adoption sur les PF : x86-64, i386, ARM32v5/v6/v7, ARM64v8, s390x, ppc64le, riscv64, mips64le



- ❖ **mnt** (*Mount*) - the container has an isolated mount table.
- ❖ **uts** (*UNIX Time-Sharing*) - the container is able to have its own hostname and domain name.
- ❖ **ipc** (*Interprocess Communication*) - processes inside the container can communicate via system-level IPC only to processes inside the same container.
- ❖ **pid** (*Process ID*) - processes inside the container are only able to see other processes inside the same container or inside the same pid namespace.
- ❖ **net** (*Network*) - the container gets its own network stack.

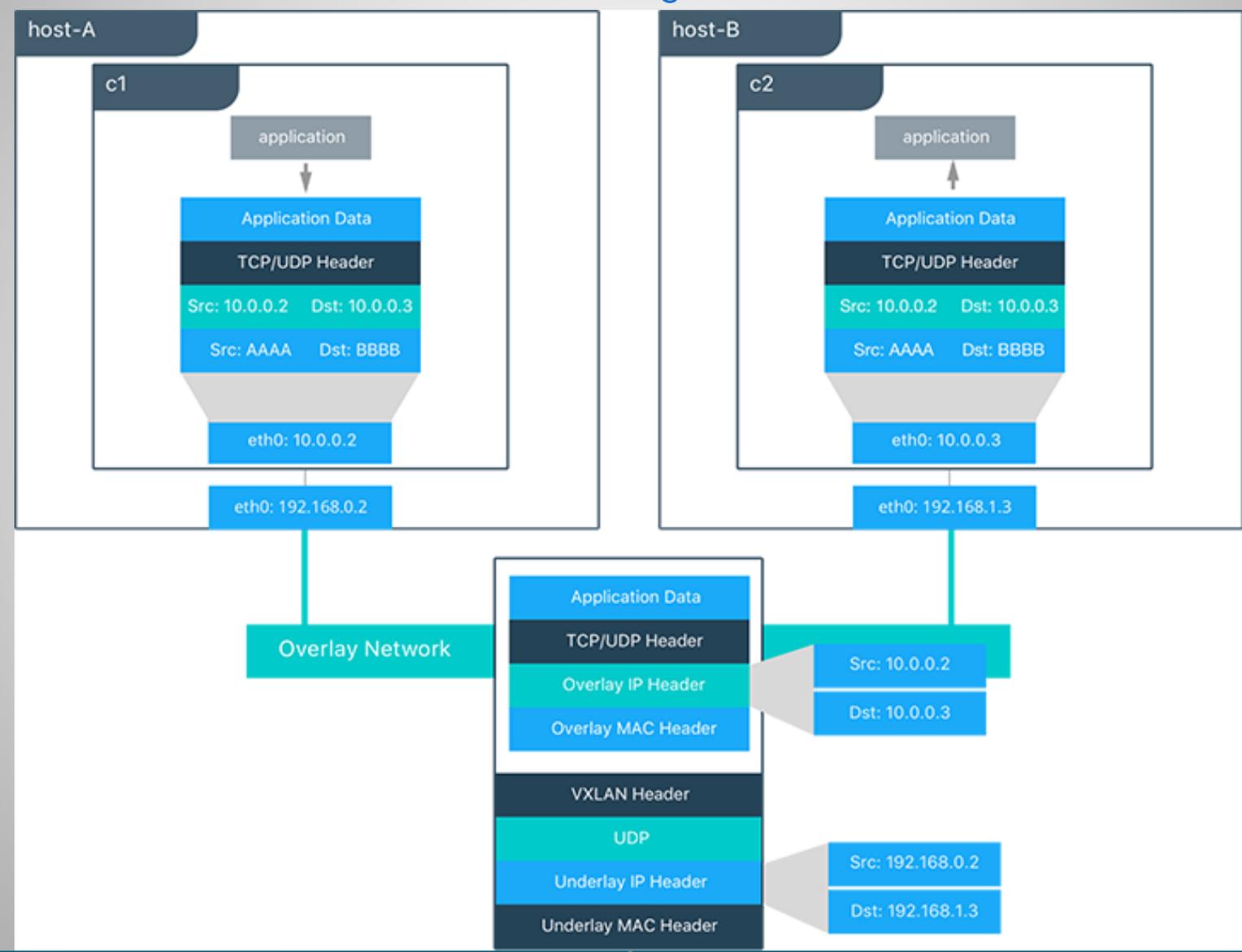
❖ Avec plusieurs conteneur/image → gestion des groupes



❖ *imageDocker2 n'est pas dans le kernel (artefact de représentation)*

- ❖ Docker permet une isolation totale entre les images, chaque image a son UID/PID.
- ❖ Les questions à se poser :
 - ❖ Comment j'organise l'échange ipc ou l'échange dans un même subnet?
 - ❖ Est-ce que je peux factoriser des « cgroup »
 - ❖ Où est-ce que je range ma supervision?

❖ **VXLAN** (Virtual Extensible LAN) technologie de virtualisation réseau (docker swarm)

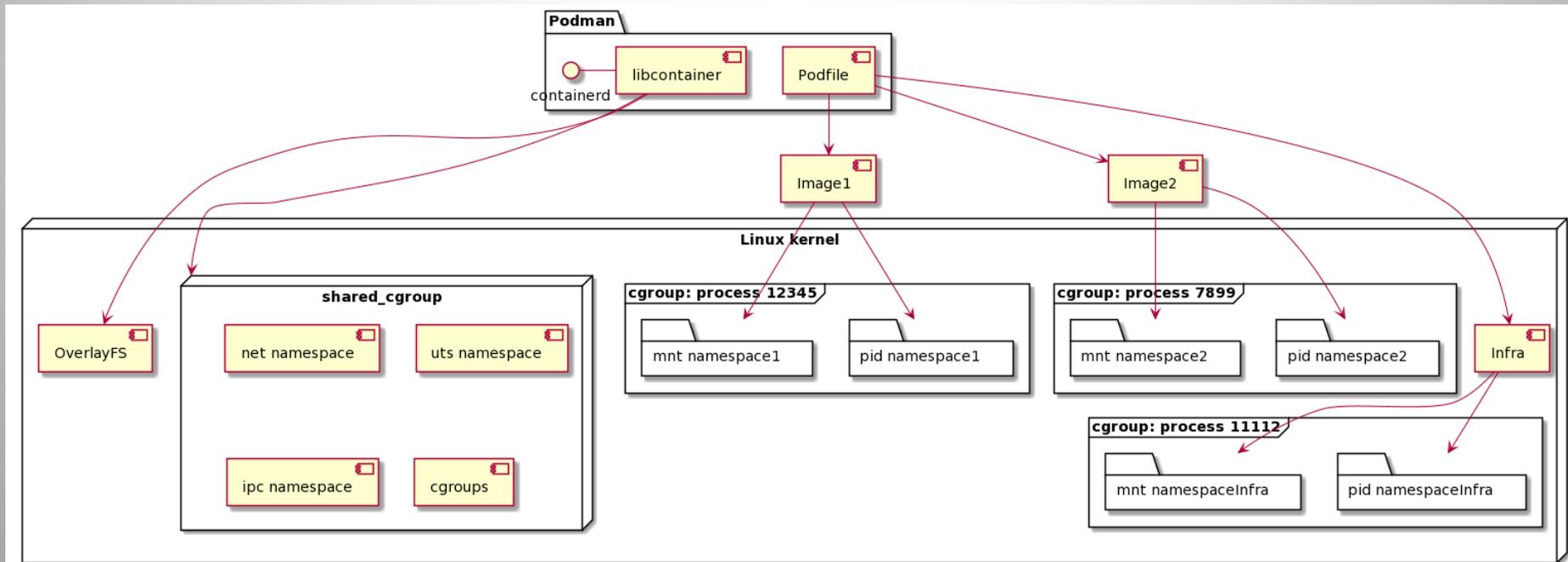




❖ Kubernetes

- ❖ 2014 (Google)
- ❖ Ecrit en Golang
- ❖ Opensource : Apache 2.0
- ❖ Remplacer Borg chez Google
- ❖ Notion de Pod → moins d'isolation
- ❖ Pod : pas de daemon, moins de risque
- ❖ Docker : un daemon – plus résilient, peut être source de compromission si mal paramétré

- ❖ Podman : isolation des process limitée – toutes les images sont interdépendantes.
- ❖ Ajout d'une image Infra qui gère l'état général (Pause)





❖ Docker

- ❖ Swarm → apporte une lecture multi-réseau

❖ Singularity

- ❖ 2015 - écrit en golang, =Docker non-root pour les grilles

❖ Kubernetes

- ❖ Construit pour faire du HA (grâce à la notion d'observateur)



❖ Scale up, Scale down

Cible matérielle	Résilience, HA	Micro-service	Calcul Scientifique
Cloud	K8s	Docker	Singularity nVidia
Cluster	K8s, K3s		
IoT	K3s	OpenFaaS (Function as a Service)	



❖ Le micro-service = panacée ?

❖ Pour :

- ❖ Simplicité de livraison
- ❖ Facilité de montée en charge
- ❖ Facilité de restauration

❖ Contre :

- ❖ Pas de hiérarchie fonctionnelle (priorité?)
- ❖ Introduit le « nouveau » métier de Devops
- ❖ Pousse à la consommation

❖ Principe du Dockerfile →

<https://raw.githubusercontent.com/jmbatto/master-mpi/main/mpi41-debian-bullseye/Dockerfile>

```
FROM debian:bullseye-slim
# an image with MPI4.1, used without root privileges
# -----
# Do basic install
# -----
RUN apt-get update \
    && mkdir -p /usr/share/man/man1 \
    && apt-get install -y gcc ssh wget vim curl net-tools bison flex autoconf make libtool m4 automake bzip2
libxml2 libxml2-dev gfortran g++ iputils-ping pkg-config colordiff nano git sudo lsof gawk emacs jq libtcl* \
    && curl --location--silent --show-error --output /tini \
    https://github.com/krallin/tini/releases/download/v0.19.0/tini \
    && chmod +x /tini \
    && adduser --uid 1000 --home /home/mpiuser --shell /bin/bash \
    --disabled-password --gecos '' mpiuser \
    && passwd -d mpiuser \
    && apt-get install -y openssh-server \
    && mkdir -p /run/sshd /home/mpiuser/.ssh /home/mpiuser/.ssh-source \
    && echo "StrictHostKeyChecking no" > /home/mpiuser/.ssh/config \
    && chown -R mpiuser /home/mpiuser \
    && sed -i s/#PermitRootLogin.*/PermitRootLogin\ no/ /etc/ssh/sshd_config \
    && sed -i s/#PubkeyAuthentication.*/PubkeyAuthentication\ no/ /etc/ssh/sshd_config \
    && sed -i s/.*/UsePAM.*/UsePAM\ no/ /etc/ssh/sshd_config \
    && sed -i s/#PasswordAuthentication.*/PasswordAuthentication\ yes/ /etc/ssh/sshd_config \
    && sed -i s/#PermitEmptyPasswords.*/PermitEmptyPasswords\ yes/ /etc/ssh/sshd_config \
    && sed -i s/#ChallengeResponse.*/ChallengeResponseAuthentication\ no/ /etc/ssh/sshd_config \
    && sed -i s/#PermitUserEnvironment.*/PermitUserEnvironment\ yes/ /etc/ssh/sshd_config \
    && adduser mpiuser sudo

ENV PREFIX=/usr/local \
    OPENMPI_VERSION=4.1.1 \
    LD_LIBRARY_PATH=/usr/local/lib \
    DEBCONF_NOWARNINGS=yes
```



```
# OpenMPI v4.1
RUN repo="https://download.open-mpi.org/release/open-mpi/v4.1" \
    && curl --location --silent --show-error --output openmpi.tar.gz \
        "${repo}/openmpi-${OPENMPI_VERSION}.tar.gz" \
    && tar xzf openmpi.tar.gz -C /tmp/ \
    && cd /tmp/openmpi-${OPENMPI_VERSION} \
        && env CFLAGS="-O2 -std=gnu99 -fopenmp" \
    && ./configure --prefix=${PREFIX} \
    && make \
    && make install \
    && ldconfig \
    && cd / \
        && rm -rf /tmp/openmpi-${OPENMPI_VERSION} /home/mpiuser/openmpi.tar.gz
# Add some parameters for MPI, mpishare - a folder shared through the nodes
RUN mkdir -p /usr/local/var/mpishare
RUN chown -R 1000:1000 /usr/local/var/mpishare
RUN echo "mpiuser ALL=(ALL) NOPASSWD:ALL\n" >> /etc/sudoers
RUN rm -fr /home/mpiuser/.openmpi && mkdir -p /home/mpiuser/.openmpi
RUN cd /home/mpiuser/.openmpi \
    && echo "btl = tcp,self \n" \
    "btl_tcp_if_include = eth0 \n" \
    "plm_rsh_no_tree_spawn = 1 \n" >> default-mca-params.conf
RUN chown -R 1000:1000 /home/mpiuser/.openmpi
RUN echo "rmaps_base_oversubscribe = 1\n" >> /usr/local/etc/openmpi-mca-params.conf
RUN echo "rmaps_base_inherit = 1\n" >> /usr/local/etc/openmpi-mca-params.conf
```



Fichier Dockerfile

docker-compose.yml

```
secrets:  
  - source: "id_rsa"  
    target: "/home/mpiuser/.ssh-source/id_rsa"  
  - source: "id_rsa_mpi_pub"  
    target: "/home/mpiuser/.ssh-source/id_rsa.pub"  
  - source: "authorized_keys"  
    target: "/home/mpiuser/.ssh-source/authorized_keys"
```

```
RUN chown -R mpiuser:mpiuser /home/mpiuser/ssh
```

```
WORKDIR /home/mpiuser
```

```
ENTRYPOINT ["/tini", "--"]
```

```
USER mpiuser
```

```
CMD cp -R /home/mpiuser/.ssh-source/* /home/mpiuser/.ssh \  
  && chmod 700 /home/mpiuser/.ssh/authorized_keys \  
  && chmod 700 /home/mpiuser/.ssh/id_rsa \  
  && exec /usr/sbin/sshd -D -f  
/home/mpiuser/ssh/sshd_config -E /tmp/sshd.log
```

❖ Ce que fait pour vous Docker

- ❖ Paravirtualisation
- ❖ Isolation / persistance ou éphémère
- ❖ Proxy (on peut remapper des ports)
- ❖ UID/GID
- ❖ → attention = par défaut, ne lance qu'1 seul processus

❖ Possibilité de faire des agencements de service au sein d'un même Docker → docker-compose

- ❖ Service 1 : Ecoute sur le port 80
- ❖ Service 2 : proxy (HTTPS)
- ❖ Service 3 : Let's Encrypt



- ❖ <https://hub.docker.com/>
- ❖ Docker hub pour x86-64 :
 - ❖ Plus de 5 Millions d'images dont
 - ❖ ~~6 705 8492~~ 9181 images vérifiées
 - ❖ → donne de la visibilité à vos projets complexes
 - ❖ → notion de tag qui simplifie la description des projets
 - ❖ → images prêtes à l'emploi !

- ❖ Docker permet la gestion de « secret » - seulement dans le mode swarm
- ❖ `docker network create --driver=overlay --attachable yml_mpinet`
- ❖ Permet le partage de secret entre nœuds
- ❖ Création de Clé RSA



- ❖ TD Prise en main de Docker+MPI
- ❖ → installation de Docker
- ❖ → installation ~~des images mpi3.1~~ et mpi4.1
- ❖ En utilisant les fichiers docker-compose.yml : →
<https://github.com/jmbatto/master-mpi>
 - ❖ ~~docker pull jmbatto/m2chps-mpi31~~
 - ❖ docker pull jmbatto/m2chps-mpi41



- ❖ `docker pull jmbatto/m2chps-mpi41`
- ❖ Git clone <https://github.com/jmbatto/master-mpi.git>
- ❖ ➔ permet de récupérer le docker-compose
- ❖ Ensuite il faut aller dans le dossier `mpi41-debian-bullseye` et y mettre le bon dossier `ssh`



- ❖ Dans powershell : `wsl –install` → WSL2 (par défaut)
- ❖ Installer Docker Desktop Windows →
<https://docs.docker.com/desktop/windows/install/>



- ❖ Git clone de l'espace de cours → contient le dossier ssh → les clés SSH
- ❖ Pour que le client MPI puisse accéder aux ressources via ssh, il faut les autorisations !

```
$HOME/.ssh
```

```
-rw-r--r--  authorized_keys
```

```
-rw-----  id_rsa
```

```
-rw-r--r--  id_rsa.pub
```

```
-rw-r--r--  known_hosts
```



INTERACTIF

❖ D'abord créer un swarm

```
docker swarm init
```

❖ Puis y attacher un réseau

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

❖ Dans mpi4l-debian-bullseye faire

```
docker-compose up --scale mpihead=1 --  
scale mpinode=2 -d
```

➔ utilise le fichier docker-compose.yml

Se connecter à un shell docker (utiliser le bash)



INTERACTIF

```
sudo curl --unix-socket  
/var/run/docker.sock  
http://localhost/containers/json | jq  
-r  
'map(.NetworkSettings[]."yml_mpinet".  
"IPAMConfig"."IPv4Address") []'
```

→ donne la liste des ip des nœuds

```
mpiuser@475c5311290d:~$ sudo curl --unix-socket /var/run/docker.sock http://localhost/containers/json | jq -r 'map(.NetworkSettings[]."yml_mpinet"."IPAMConfig"."IPv4Address") []'  
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current  
           Dload  Upload  Total      Spent    Left     Speed  
100 11783    0 11783    0    0   885k      0  --:--:--  --:--:--  --:--:--  958k  
10.0.1.24  
10.0.1.25  
10.0.1.23
```

INTERACTIF

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 2 -host  
10.0.1.24,10.0.1.23 python3  
./mpi4py_benchmarks/all_tests.py
```

```
mpirun@475c5311290d:~$ mpirun --mca orte_base_help_aggregate 0 --mca btl_tcp_if_include 10.0.1.0/24 -n 2 -host 10.0.1.24,10.0.1.23 python3 ./mpi4py_benchmarks/all_tests.py  
Warning: Permanently added '10.0.1.24' (ECDSA) to the list of known hosts.
```

```
=====
```

#	Duration [s]	Throughput [#s]
	2.147	9.31
	2.057	9.72
	2.102	9.51
	2.079	9.62
	1.945	10.28
	1.843	10.85
	1.685	11.87
	1.821	10.98
	1.845	10.84
	1.698	11.78

INTERACTIF

❖ Description du paramétrage openmpi

```
mpi_info --level 9 --param rmaps base
```

```
mpiuser@475c5311290d:~$ mpi_info --level 9 --param rmaps base
MCA rmaps base: -----
MCA rmaps base: parameter "rmaps_base_verbose" (current value:
                 "error", data source: default, level: 8 dev/detail,
                 type: int)
                 Verbosity level for the rmaps framework (default:
                 0)
                 Valid values: -1:"none", 0:"error", 10:"component",
                 20:"warn", 40:"info", 60:"trace", 80:"debug",
                 100:"max", 0 - 100
```

❖ On souhaite « pratiquer » MPI → utiliser mpicc + Impi

[hello_world.cc](#), [mpi_any_source.cc](#)

2 codes C à récupérer (git clone) depuis
<https://github.com/jmbatto/master-mpi>



INTERACTIF

❖ On souhaite « pratiquer » MPI → utiliser mpicc + Impi

hello_world_mpi.cc

```
mpiuser@475c5311290d:~/master-mpi$ mpicc hello_world_mpi.cc -o hello_world -lmpi
mpiuser@475c5311290d:~/master-mpi$ ./hello_world
Hello world from processor 475c5311290d, rank 0 out of 1 processors
mpiuser@475c5311290d:~/master-mpi$ mpirun --mca orte_base_help_aggregate 0 --mca btl_tcp_if_include 10.0.1.0/24 -n 2 -host 10.0.1.24,10.0.1.23 hello_world
-----
mpirun was unable to find the specified executable file, and therefore
did not launch the job. This error was first reported for process
rank 1; it may have occurred for other processes as well.

NOTE: A common cause for this error is misspelling a mpirun command
line parameter option (remember that mpirun interprets the first
unrecognized command line token as the executable).

Node:      10.0.1.24
Executable: /home/mpiuser/master-mpi/hello_world
```

Il faut se placer dans l'espace partagé avant de lancer la commande

```
mpinode:
image: jmbatto/m2chps-mpi41:latest
volumes:
- /var/run/docker.sock:/var/run/docker.sock
- usrlocalvarmpi-foo:/usr/local/var/mpishare
```



INTERACTIF

❖ On souhaite « pratiquer » MPI → utiliser mpicc + Impi

hello_world.cc

```
$mpirun --mca orte_base_help_aggregate 0 --mca  
  btl_tcp_if_include 10.0.2.0/24 -n 2 -host  
  10.0.2.28,10.0.2.27 hello_world
```

```
Hello world from processor 1f9914291b09, rank 0 out of  
  2 processors
```

```
Hello world from processor f2057a67f137, rank 1 out of  
  2 processors
```

```
$ mpicc mpi_any_source.cc -o any_source -lmpi
```

```
$ mpirun --mca orte_base_help_aggregate 0 --mca  
  btl_tcp_if_include 10.0.2.0/24 -n 2 -host  
  10.0.2.28,10.0.2.27 any_source
```

```
[MPI process 0] I send value 12345.
```

```
[MPI process 1] I received value 12345, from rank 0.
```



INTERACTIF

❖ Analyse du code Python

```
mpirun --mca  
orte_base_help_aggregate 0 --mca  
btl_tcp_if_include 10.0.2.0/24 -n 2 -host  
10.0.2.24,10.0.2.19 python3  
./mpi4py_benchmarks/all_tests.py
```

❖ Running 2 parallel MPI processes

❖ # MPI Matrix action on a vector, 20 iterations of size 10000

❖ # Duration [s] Throughput [# /s]

❖ 1.403 14.26

❖ 1.374 14.56

❖ 1.286 15.55

❖ 1.329 15.05



INTERACTIF

```
while counter < iter:
    comm.Barrier()                ### Start stopwatch ###
    t_start = MPI.Wtime()
    for t in xrange(20):
        my_new_vec = np.inner(my_M, vec)
        comm.Allgather(
            [my_new_vec, MPI.DOUBLE],
            [vec, MPI.DOUBLE]
        )
        comm.Barrier()
    t_diff = MPI.Wtime() - t_start    ### Stop stopwatch
if myid == 0:
    print ('%-10.3f%20.2f' % (t_diff, bs/t_diff))
    counter += bs
```



Fichier : al_test.py

❖ Connaitre la capacité de la VM Docker

INTERACTIF

```
docker stats --no-stream --format "table {{.Name}}\t{{.Container}}\t{{.MemUsage}}"
```

```
root@vps_eldarsoft:~# docker stats --no-stream --format "table {{.Name}}\t{{.Container}}\t{{.MemUsage}}"
```

NAME	CONTAINER	MEM USAGE / LIMIT
gogs	428f4d97c4a1	352.8MiB / 1.899GiB
traefik	7cce703b64a8	22.41MiB / 1.899GiB

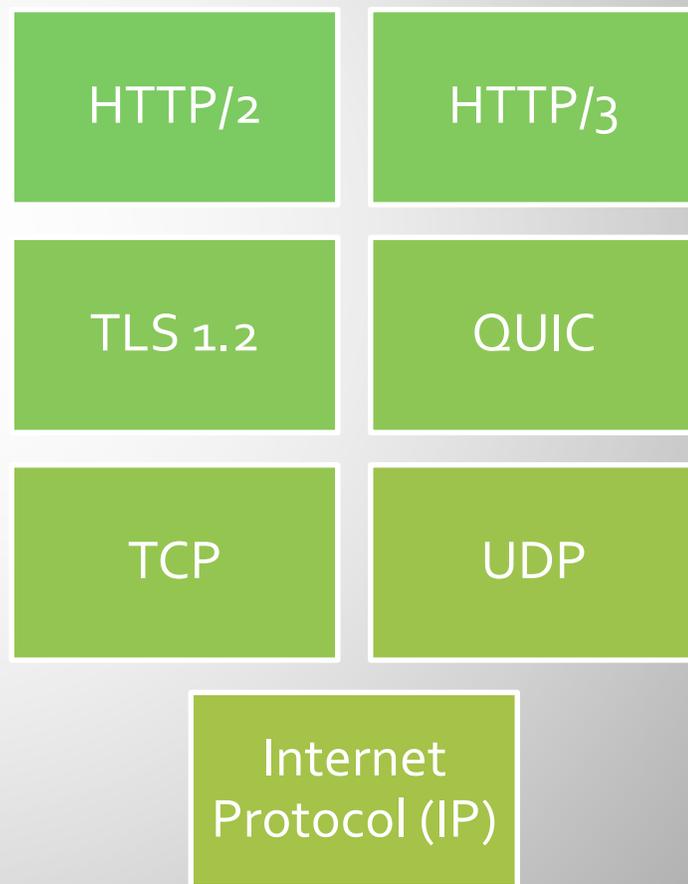
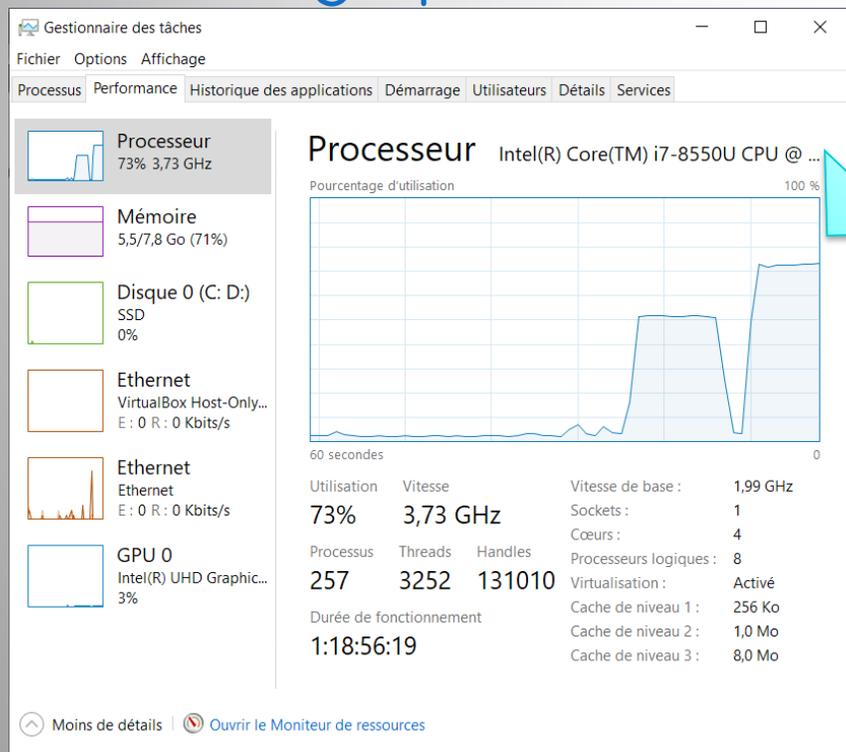
```
root@vps_eldarsoft:~# █
```

NAME	CONTAINER	MEM USAGE / LIMIT
telegrafmonitor-auto	f19b09e8cd2b	11.43MiB / 14.33GiB
pure-ftpd	50da42229bd9	2.738MiB / 14.33GiB
jugydemo	93d1b26824ea	5.902MiB / 14.33GiB
jugy_web2	d23ceeb0a02a	9.828MiB / 14.33GiB
proxyApiSVMS	cd9f4b43fea7	2.707MiB / 14.33GiB
apiSVMS	8c06a1152108	18MiB / 14.33GiB
redmine-menlo systems	f53ba3caa5d6	490.5MiB / 14.33GiB
traefik	dfc54d4ab4da	21.88MiB / 14.33GiB
reoges_web	c0c5a071dd12	10.81MiB / 14.33GiB
chronograf	d1a698395cd3	12.58MiB / 14.33GiB
telegraf-auto	e7452b5722d6	26.61MiB / 14.33GiB
chronograf-poisson	ac14cbeb59dd	17.4MiB / 14.33GiB
kapacitor-auto	e8d7bc45a584	33.67MiB / 14.33GiB
phpmyadmin	060b9b29b02f	83.9MiB / 14.33GiB
influxdb	1fa90bc9f49a	1.261GiB / 14.33GiB
gogs	69a2c6cfe67f	99.53MiB / 14.33GiB
mariadb	edb0c20bdd20	7.556GiB / 14.33GiB



INTERACTIF

- ❖ Le disque (si il y a des I/O)
- ❖ Le réseau (influence si UDP vs TCP) (ie QUIC)
- ❖ Binding (static vs late)
- ❖ Charge processeur



❖ TD1 : individuel

❖ <http://mvapich.cse.ohio-state.edu/benchmarks/>

❖ wget <http://mvapich.cse.ohio-state.edu/download/mvapich/osu-micro-benchmarks-7.3/.tar.gz>

❖ → installation, compilation, test, benchmarks (C vs Python) (vous pouvez comparer les 2 versions de Python avec le C (tableau avec 3 colonnes)

❖ → Mettre dans la Forge Gogs un rapport du TD1 (1 **page** PDF – présente un benchmark *MPI Bi-Directional Bandwidth Test* osu_bibw dans un tableau avec une introduction qui donne le contexte, avec le nom de l'étudiant et la date) :
pour le 15 décembre 2024 minuit



❖ `docker build -t jmbatto/m2chps-mpi41` → permet de construire l'image Docker

A ne pas faire car cela prend plus de temps de construire que de télécharger.

❖ `docker-compose scale mpihead=1 mpinode=9` → utilise le fichier `docker-compose.yml`

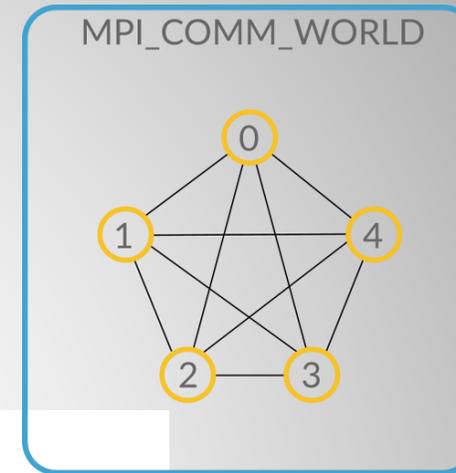


- ❖ API synchrone & asynchrones avec processus numérotés (=connus)
 - ❖ MPI_Init()
 - ❖ MPI_Abort()
 - ❖ MPI_Get_processor_name()
 - ❖ MPI_Comm_size()
 - ❖ MPI_Comm_rank()
 - ❖ MPI_Send(), MPI_Ssend()
 - ❖ MPI_Recv()
 - ❖ MPI_Finalize()
 - ❖ MPI_Bcast() / MPI_Scatter()
 - ❖ MPI_Reduce()
 - ❖ MPI_Allreduce()

❖ MPI_Init() /MPI_Finalize()

❖ MPI_Abort()

```
#include <mpi.h>
int main(int argc, char *argv[])
{
    MPI_Init(NULL, NULL);
    MPI_Abort(MPI_COMM_WORLD, 112); // in Europe 112, for emergency
    /* No further code will execute */
    MPI_Finalize();
    return 0;
}
```



❖ MPI_Get_processor_name()

❖ → résultat de gethostname, uname, ou sysinfo



- ❖ `MPI_Comm_size()` → effectif des processus
- ❖ `MPI_Comm_rank()` → rang du processus

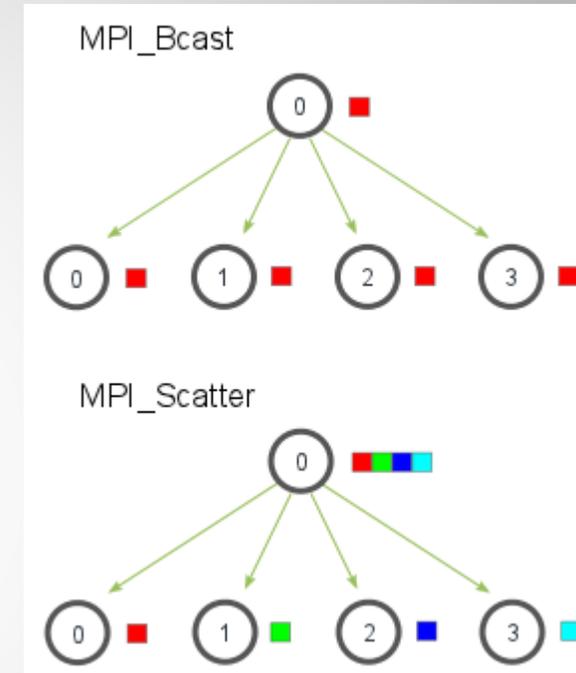
- ❖ `MPI_Send()` → communication point à point
- ❖ `MPI_Ssend()` → communication point à point, **synchrone**
- ❖ `MPI_Recv()` → réception point à point, responsable du buffer data

❖ MPI_Bcast() / MPI_Scatter()

❖ Instructions avec fonctions

❖ MPI_Reduce() → root

❖ MPI_Allreduce() → partage



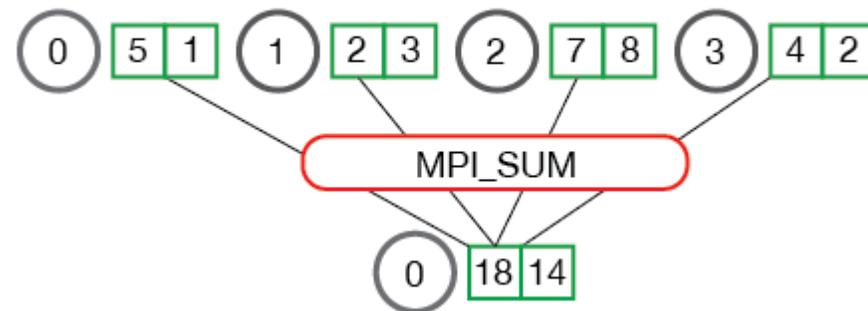
❖ MPI_Bcast() / MPI_Scatter()

❖ Instructions avec fonctions

❖ MPI_Reduce() → root

❖ MPI_Allreduce() → partage

MPI_Reduce



MPI_Allreduce

