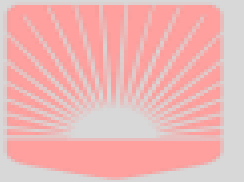




UNIVERSITÉ DE
VERSAILLES
ST-QUENTIN-EN-YVELINES
université PARIS-SACLAY



#10

13/02/2026

jean-michel.batto@cea.fr

cea

https://gogs.eldarsoft.com/M2_IHPS

On approche un prix client de 80k€

Grande dispersion dans les prix (écart type de 50k€)

Vision « low cost » : absence de prudence !

Dans le prompt : ne pas hésiter à demander au LLM de vérifier la cohérence de l'estimation et de lui demander s'il y a des oublis dans votre démarche.

2 approches du Service HPC

❖ Allocation de ressources dans une Grille

- ❖ Crédit d'heure à consommer
- ❖ Nombre de cœurs alloués
- ❖ Interfaces de communication (MPI), système de fichier HPC (Lustre, GPFS, CEPH...)

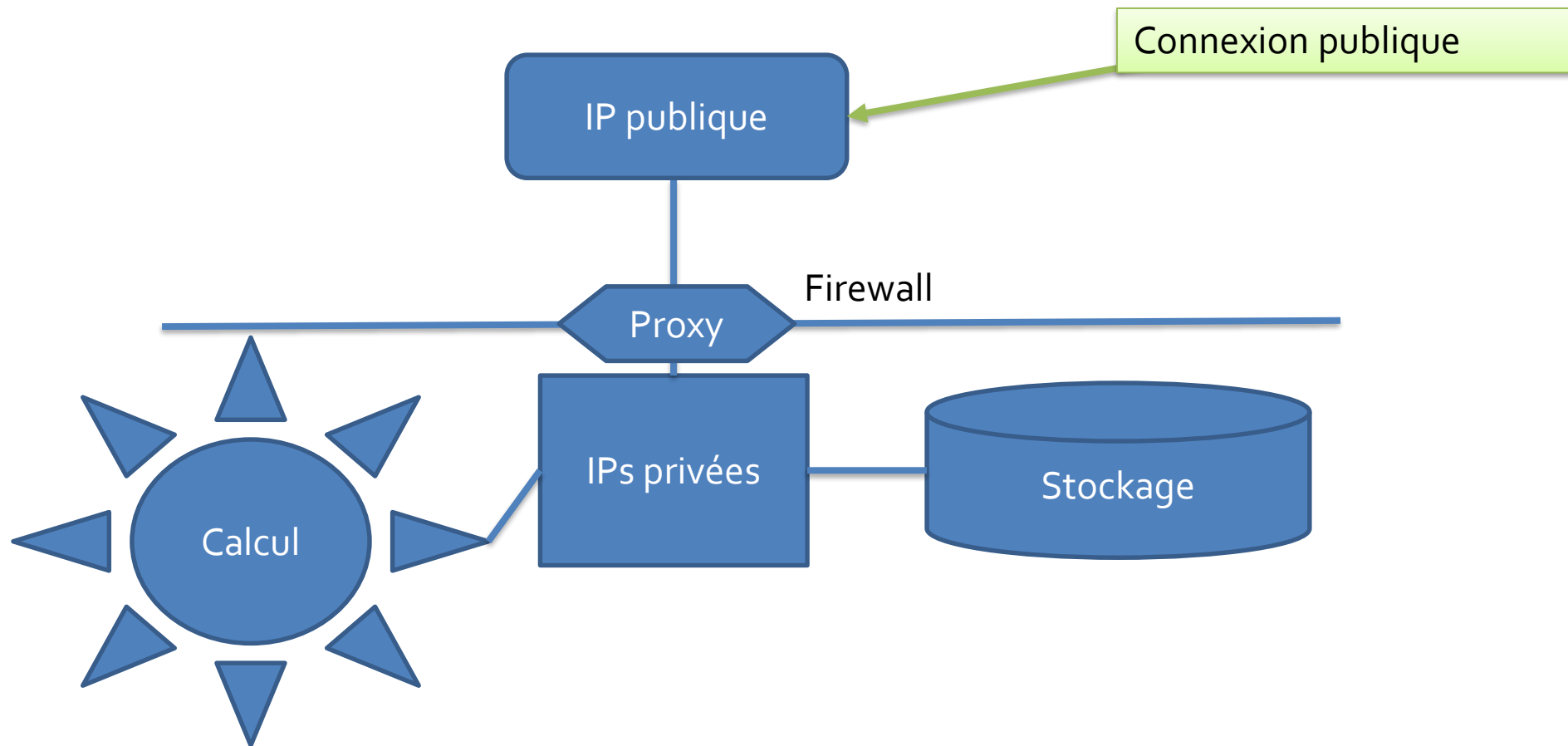
❖ → approche par la disponibilité de la ressource

- ❖ Pb d'efficacité (saturation)
- ❖ Pb de commutation de projet (nbre de nœud dispo/contrat)
- ❖ Pb de priorité → le prix est connu à la fin du traitement?

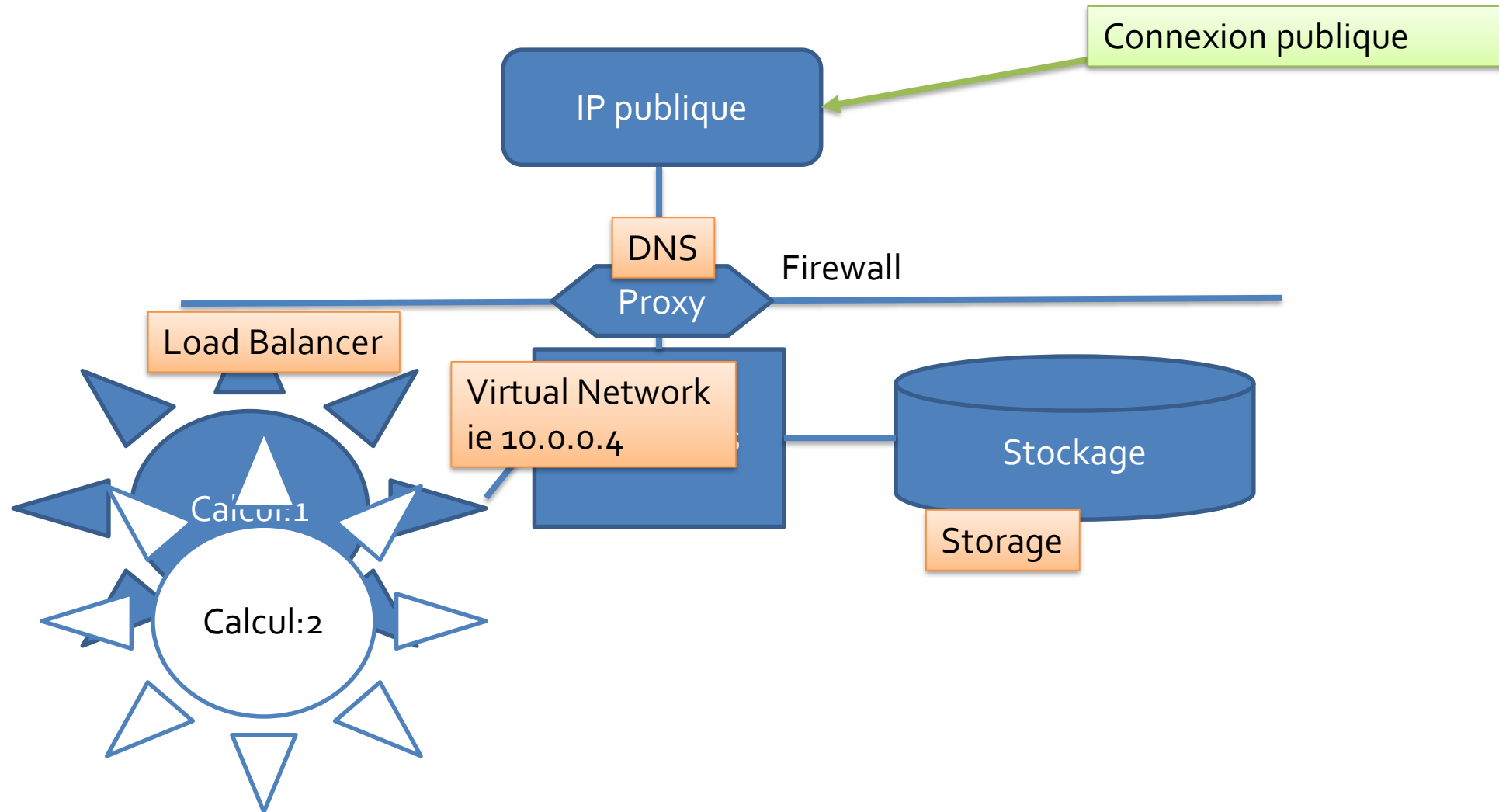
❖ → approche par la saturation de la ressource

- ❖ L'application est atomisée (grain de temps de x ms)
- ❖ Le prix de l'application est calculé en continu
- ❖ Selon le niveau du prix – le grain peut être activé (== priorité), le batch peut être 'failed' et relancé plus tard.

Le cloud



Le cloud : les services indispensables



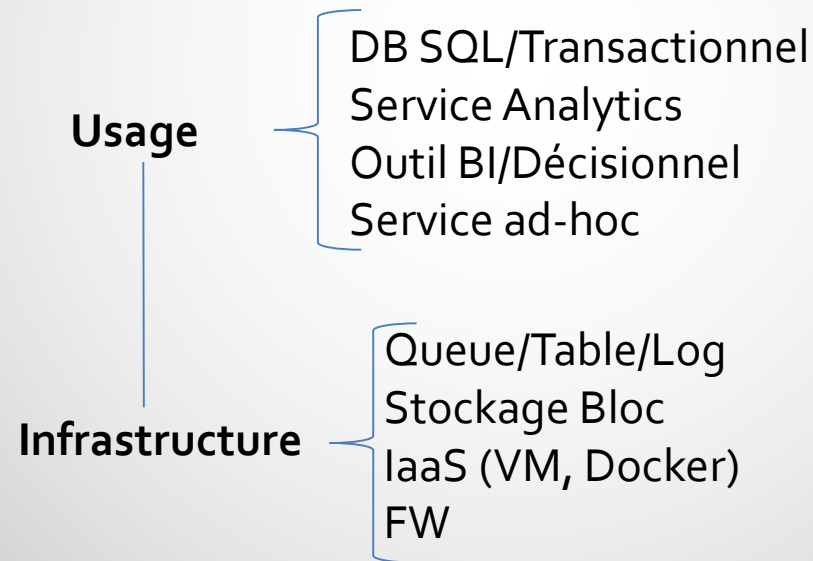
Sa spécialisation

Amazon utilise AWS (Amazon Web Services 2006)

Google propose GCP (Google Cloud Platform 2011)

➔ logique métier interne pour le métier « de base »

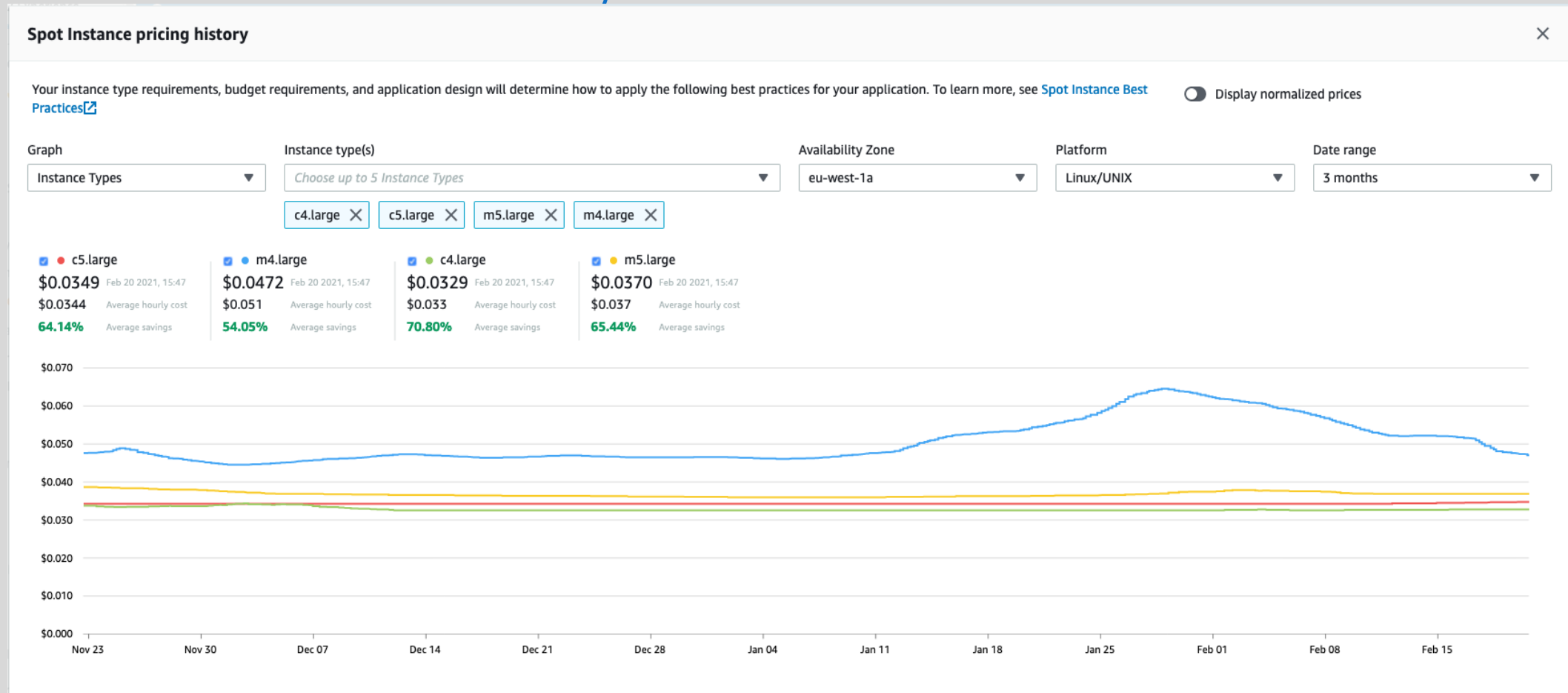
Ensuite la logique métier s'est spécialisée dans 2 directions.



- Axe haut : l'usage
 - Dans le domaine de l'analytique – il y a un leader de l'analytique Vertica
 - Snowflake propose des usages pour l'analytique avec une approche Share Nothing et cible différents Cloud (AWS, GCP, Azure)
- Axe bas : l'infrastructure (machine VM, stockage)
- ➔ On achète un service hébergé

Tarification flottante

- ❖ SPOT AWS – un marché du service de calcul pour EC2
- ❖ Google “Preemptible VM Instances”
- ❖ Azure “Low-Priority VM”



Historique de tarification d'instance Spot

Le mode de mise en œuvre des bonnes pratiques suivantes pour votre application est fonction de vos exigences en matière de type d'instance, de budget et de conception des applications. Pour en savoir plus, consultez [Bonnes pratiques pour les instances Spot](#)

Graphique

Zones de disponibilité

Type d'instance

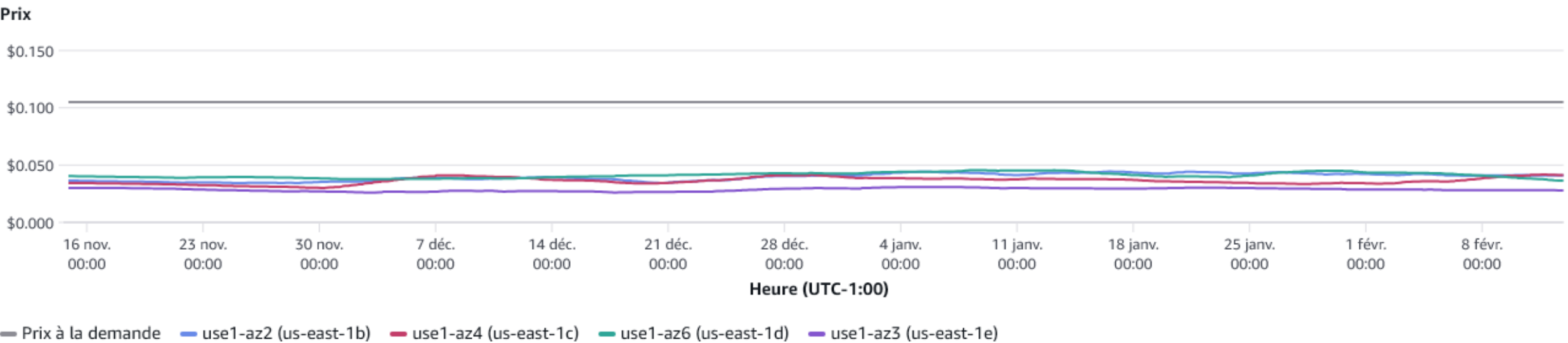
c3.large

Plateforme

Linux/UNIX

Plage de dates

3 mois



Moyenne par heure dans la plage de dates

À la demande	use1-az2 (us-east-1b)	use1-az4 (us-east-1c)	use1-az6 (us-east-1d)	use1-az3 (us-east-1e) Le moins cher
\$0.1050	\$0.0396	\$0.0363	\$0.0412	\$0.0285
	\$0.0198 par vCPU	\$0.0181 par vCPU	\$0.0206 par vCPU	\$0.0143 par vCPU
	62.25 % d'économies	65.47 % d'économies	60.78 % d'économies	72.84 % d'économies

Historique de tarification d'instance Spot



Le mode de mise en œuvre des bonnes pratiques suivantes pour votre application est fonction de vos exigences en matière de type d'instance, de budget et de conception des applications. Pour en savoir plus, consultez [Bonnes pratiques pour les instances Spot](#)

Graphique

Zones de disponibilité

Prix

\$0.150

\$0.100

\$0.050

\$0.000

16 nov. 00:00

23 nov. 00:00

30 nov. 00:00

Prix à la demande

use1-az2 (us-east-1b)

Type d'instance

c3.large

Q |

a1.medium
CPU-v1 2GB RAM EBS only

a1.large
CPU-v2 4GB RAM EBS only

a1.xlarge
CPU-v4 8GB RAM EBS only

a1.2xlarge
CPU-v8 16GB RAM EBS only

a1.metal
CPU-v16 32GB RAM EBS only

a1.4xlarge
CPU-v16 32GB RAM EBS only

c1.medium
CPU-v2 1.6992GB RAM 1 x 350

c1.xlarge
CPU-v8 7GB RAM 4 x 420

c3.large
CPU-v2 3.75GB RAM 2 x 16 SSD

c3.xlarge
CPU-v4 7.5GB RAM 2 x 40 SSD

c3.2xlarge

Plateforme

Linux/UNIX

Plage de dates

3 mois

Moyenne par heure dans la plage de date

À la demande
\$0.1050

use1-az2 (us-east-1b)
\$0.0396
\$0.0198 par vCPU
62.25 % d'économies

use1-az6 (us-east-1d)
\$0.0412
\$0.0206 par vCPU
60.78 % d'économies

use1-az3 (us-east-1e)
\$0.0285
\$0.0143 par vCPU
72.84 % d'économies

Le moins cher

Lambda

- ❖ Concept atomique le plus poussé
- ❖ Les langages sont : Node.js, .Net, Python, Ruby, Java
- ❖ Go, Rust, C++
- ❖ Limite de 50 Mo pour le code de la fonction

Dernier pris en charge
.NET 10 (C#/F#/PowerShell)
Java 25
Node.js 24.x
Python 3.14
Ruby 3.4
Amazon Linux 2023 Exécution uniquement pour le système d'exploitation pour Go, Rust, C++, personnalisé
Autre prise en charge
.NET 8 (C#/F#/PowerShell)
Java 11
Java 17
Java 21

▼ Créer des flux de travail sans serveur avec des machines d'état

AWS Step Functions est un service d'orchestration qui vous permet de connecter vos fonctions Lambda à des flux de travail sans serveur, appelés *machines d'état*. Créez des flux de travail de longue durée pour l'automatisation informatique et les cas d'utilisation d'approbation humaine ou des flux de travail volumineux et de courte durée pour le traitement et l'ingestion de données en streaming. [En savoir plus](#)



Créer des chaînes

Reliez les fonctions de façon à créer une série d'étapes, la sortie d'une étape fournissant l'entrée de l'étape suivante.



Catch and retry

Gérez les erreurs à l'aide de la fonctionnalité sophistiquée Catch and retry.



Parallélisme

Exécutez des fonctions en parallèle ou utilisez le parallélisme dynamique pour appeler une fonction pour chaque membre d'un tableau.



Branchement

Concevez votre flux de travail pour choisir différentes branches en fonction de la sortie de la fonction Lambda.

Modèle de vente de Lambda

- ❖ Grain 1 ms (100 ms typique)
- ❖ Cout de traitement \$0.20 pour 1M de requête
- ❖ 0,0000166667 USD par Go-seconde → à fractionner
- ❖ (par ex 128 mo = 1/8Go ; 1 sec = 1000 ms)
- ❖ 1M de requêtes gratuites par mois
- ❖ Jusqu'à 3,2 millions de secondes de temps de calcul gratuit par mois

- ❖ Avec option Edge, environ *3 sur le prix
- ❖ Pour moins de latence

Prix (x86)

0,0000166667 USD par Go-seconde

0,20 USD par million de demandes

Le coût de la durée est fonction de la quantité de mémoire que vous attribuez à votre fonction. Vous pouvez allouer n'importe quelle quantité de mémoire (entre 128 Mo et 10 240 Mo) à votre fonction et ceci par tranches de 1 Mo. Le tableau ci-dessous contient quelques exemples de prix par tranche de 1 ms associés à différentes tailles de mémoire.

Prix (x86)

Prix (Arm)

Région : Europe (Paris) ▾

Mémoire (Mo)	Tarif par 1 ms
128	0,0000000021 USD
512	0,0000000083 USD
1 024	0,0000000167 USD
1 536	0,0000000250 USD
2 048	0,0000000333 USD
3 072	0,0000000500 USD

Tarifs AWS Lambda (x86, tier de base – US East par ex.)

Prix par requête : \$0.20 pour 1 million de requêtes (soit \$0.0000002 par requête individuelle).

Prix par durée : \$0.0000166667 par GB-seconde ($\approx \$1.66667 \times 10^{-8}$ / GB-s).

→ **0,000000016666667 USD par GB-seconde.**

Granularité de facturation de la durée : Arrondi au ms supérieur (grain de **1 ms** depuis fin 2020 – plus 100 ms comme avant).

Typique : 100 ms pour beaucoup de fonctions (API simples, etc.), mais peut être 1–10 ms pour du très léger.

Free Tier (toujours gratuit, même après des années) :

1 million de requêtes gratuites par mois.

400 000 GB-secondes de compute gratuit par mois.

Cela équivaut à $\approx$ **3,2 millions de secondes** de compute total avec 128 MB (car $400\,000\text{ GB-s} \div (128/1024\text{ GB}) = 3,2\text{ M secondes}$).

Exemple concret : à 128 MB (0,125 GB), $400\,000\text{ GB-s} =$ **3 200 000 secondes** ($\approx 3,2\text{ M s}$) gratuites → $\approx$ **888 heures** par mois ($3,2\text{ M s} \div 3600$).

À 1 GB : seulement $\approx 400\,000$ secondes gratuites (≈ 111 heures/mois).

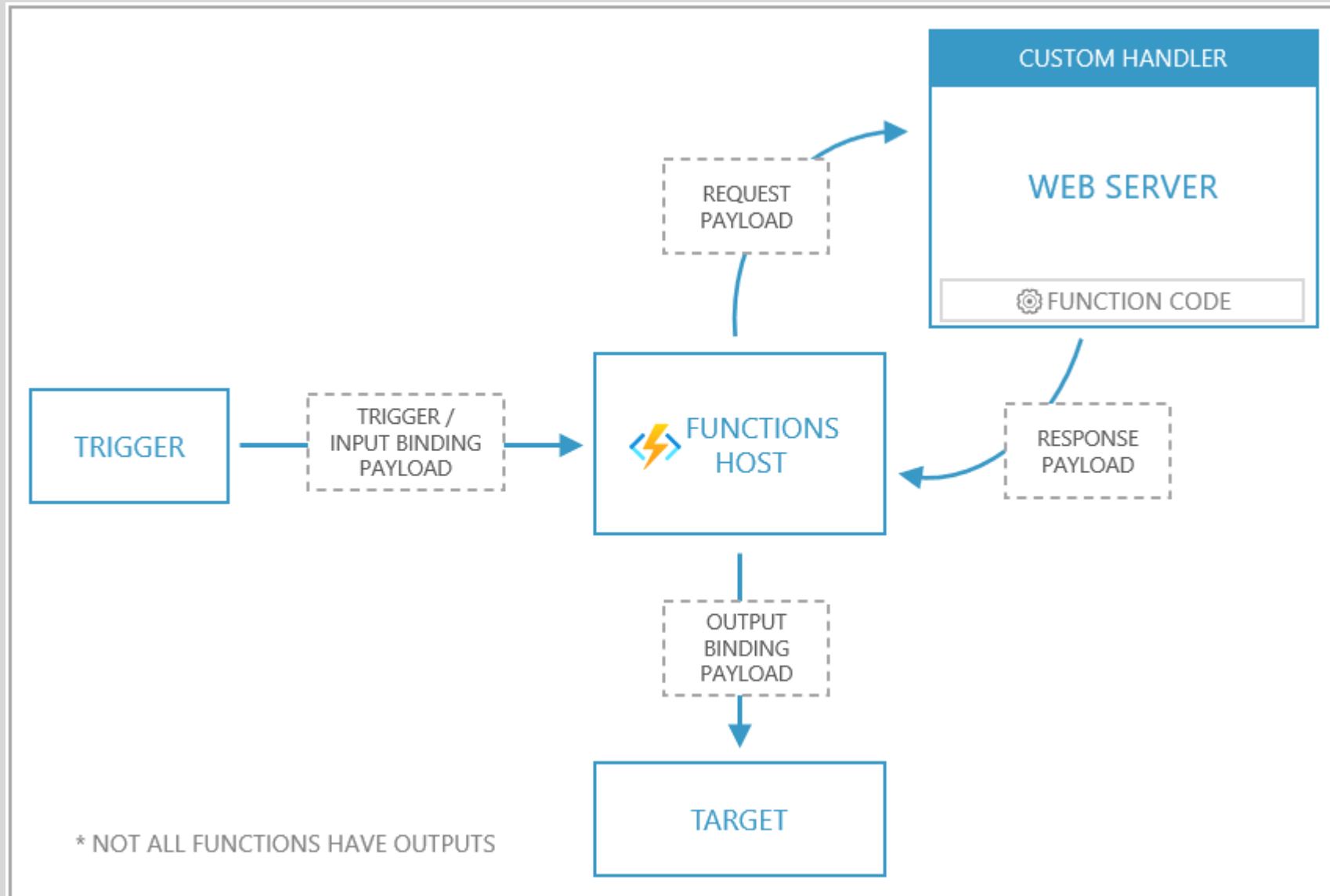
Supposons 2 millions de requêtes/mois, durée moyenne 100 ms, mémoire 512 MB (0,5 GB) :

•Requêtes : 1M gratuit → 1M payantes = \$0.20.

•Durée : $2\text{M} \times 0,1\text{ s} = 200\,000$ secondes totales.

- GB-s = $200\,000\text{ s} \times 0,5\text{ GB} =$ **100 000 GB-s.**
- Free tier couvre 400 000 GB-s → **0 GB-s payants** ici.
- Coût total : **\$0.20** (que les requêtes).

Lambda : idée du service atomique



<https://docs.microsoft.com/en-us/learn/modules/serverless-go/2-custom-handlers>

Serverless Azure

3 approches **Serverless** chez Azure

- Azure Functions = Lambda
- Azure Service Fabric Cluster (Docker avec ou sans Kubernetes)
➔ capacité à être déployé sur plusieurs nœuds en même temps
- Azure Container Instances = Docker

Function App – un service à l'écoute avec un handler

- Serverless (sans contrat sur la cible)
6 langages managés : Python / Java / C# .NET / Node.JS / TypeScript / Powershell (BASH)
1 option : Custom Handler = cible binaire sur linux64
- Serverless avec contrat sur le type de server (image Docker)
Function Premium (Serverless avec isolation)
App Service Plan (partage d'un même plan entre plusieurs Function App)

Container (Docker) Azure

Azure Container Instances – un service à l'écoute d'un port

- OS : Windows Server 2016 ou Linux
- Endpoint : le port qui exposé au publique, le nom du service
- Portbinding, Endpointref : le port qui est liée à l'image Docker et au nom du service
- Docker HEALTHCHECK ➔ donne l'état du service

Lambda

	AWS	Azure	GCP
Nom du service	Lambda	Functions	Cloud functions
Langage spécifique autre que Python / Java / C# .NET / Node.JS	Go, Ruby, C++	TypeScript	Go, PHP, Ruby
Orchestration	Step functions	Logic Apps	Workflow
Monitoring	Log, Application, Metrics	Log, Application, Metrics	Log, Application, Metrics
Triggers	HTTP, queue, timer, storage, DB, S3	HTTP, queue, timer, storage BLOB, cosmos DB	HTTP, queue, timer, storage

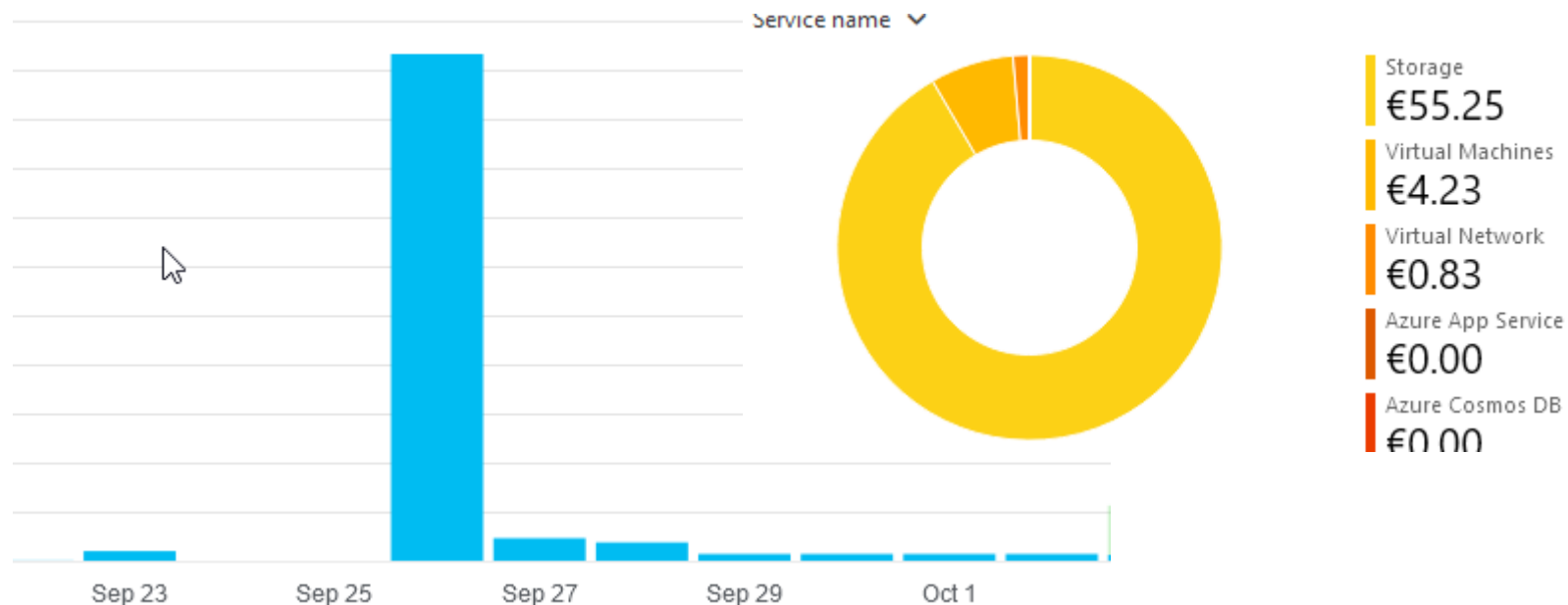
Stockage S3

Inventé en 2006 par AWS

Stockage bloc – avec une notion de bucket – taille max 5To

ID du bucket : 63 caractères – ID unique global

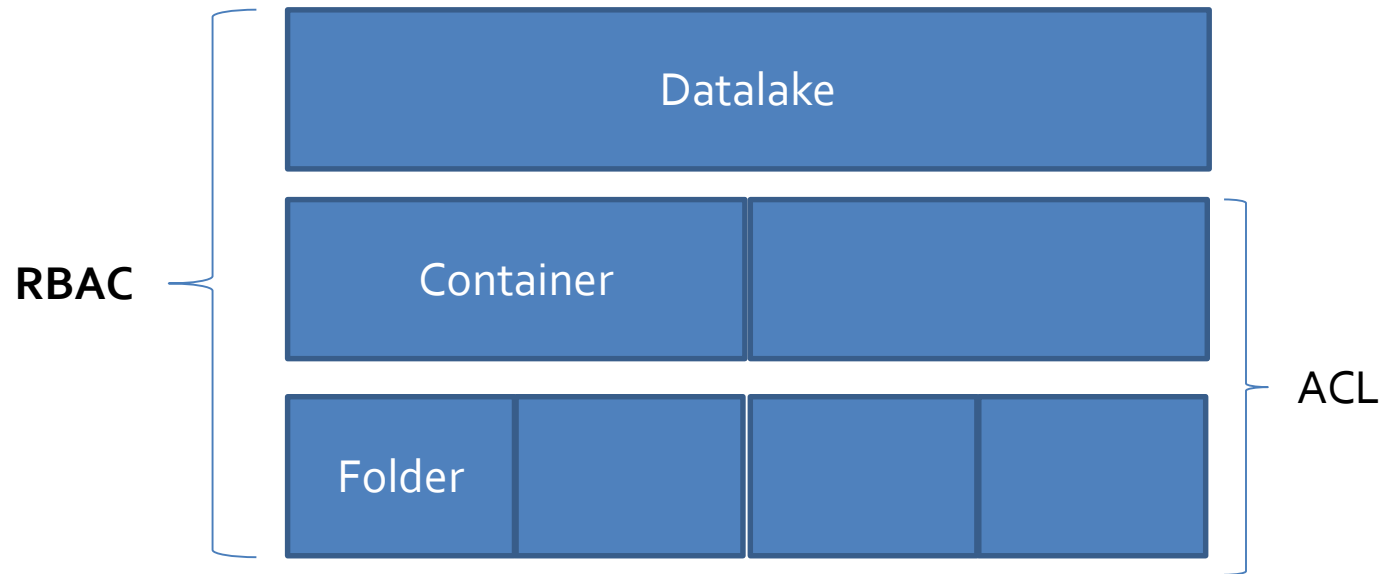
L'écriture se fait par ajout de bloc (512 octets à 4 Mo)



La gestion des droits (Azure)

Gestion des droits d'accès : RBAC puis ACL
RBAC : **Role-based access control** (prime sur ACL)

ACL : Access Control List



Redondance du stockage

- **Read-Access Geo-Redundant Storage (RA-GRS)**
- ➔ **Read Access** : activation en lecture seule en cas de défaut
- **Geo-Redundant Storage (GRS)**
- **Zone Redundant Storage (ZRS)**
- **Locally Redundant Storage (LRS)**

	AWS	Azure	GCP
Localized replication	Selon S3	LRS, ZRS	?
Cross Regional replication	CRR	GRS, RA-GRS	Multi-region

Cout du Cloud

Par mois	Azure	GCP	AWS	OVH
Stockage S3 (3To) – +1 Million opération de lecture Pas de réplication	60€	60€	70€	60€
Opération Ecriture SFTP 500 Mo	6,50€	500 Mo SFTP par bloc de 512 octets = (100*10 000 opérations*512 octets) 100*0,06 € = 6+0,26€/h pour le transfert en 2h		
FaaS	https://cloud.google.com/functions/pricing?hl=fr			Non
DBaaS MySQL (le moins cher)	8 Go Ram 160€	db-g1-small HD* // 1.7Go Ram 51€		Essential DB1-4 // 4Go Ram 50€
IaaS (plus petite machine)	15€			5,5€

<https://azure.microsoft.com/fr-fr/pricing/calculator/>

Les services Azure de type « Lambda »

Option	Capacité à spécifier l'OS hôte	Gestion à réaliser par soi même	DevOps – integration continue	Orchestration
Azure Function (FaaS)	NON (runtime géré, mais conteneurs personnalisés offrent un contrôle sur l'environnement sans spécifier l'OS hôte sous-jacent)	NON (serverless, fully managed par Azure)	Partiel (intégrations avec GitHub Actions, Azure DevOps, VS Code, mais pas full CI/CD natif)	S/O (Durable Functions pour workflows, mais pas d'orchestration complète comme Kubernetes)
Azure Service Fabric	Limité (choix entre Windows ou Linux pour les clusters, mais pas de personnalisation approfondie)	Limité (gestion des clusters requise, mais Azure facilite le déploiement multi-environnements)	Partiel (intégrations avec Azure Pipelines, Jenkins, etc., pour CI/CD)	OUI (orchestrateur natif pour microservices et conteneurs)
Azure Kubernetes Service (AKS, remplaçant ACS)	Limité (choix d'OS mixtes par node pools, comme Windows/Linux, mais géré par Azure)	Limité (managed Kubernetes, Azure gère le control plane, utilisateur gère les nodes/apps)	Partiel (intégrations fortes avec DevOps tools comme Helm, Draft, Azure DevOps pour CI/CD)	OUI (orchestration Kubernetes complète, avec autoscaling, load balancing, etc.)
VM Azure	OUI (choix complet d'OS via Marketplace : Windows, Linux distributions variées)	OUI (self-managed : patching, config, software à charge de l'utilisateur)	NON (pas natif, mais possible via outils externes comme Azure DevOps ou scale sets)	S/O (orchestration basique via Virtual Machine Scale Sets, mais pas dédié)

Fonction HTTP à volume élevé (GCP)

Fonction HTTP de complexité moyenne avec 256 Mo de mémoire et un processeur de 400 MHz, appelée 50 millions de fois par mois via HTTP, s'exécutant pendant 500 ms à chaque fois et renvoyant 5 Ko de données à l'appelant (5 Ko en sortie par appel).

Calculs

Appels

50 000 000

Temps de calcul

$(256 \text{ Mo} / 1\,024 \text{ Mo/Go}) \times 0,5 \text{ s} = 0,125 \text{ Go-seconde par appel}$

$(400 \text{ MHz} / 1\,000 \text{ MHz/GHz}) \times 0,5 \text{ s} = 0,200 \text{ GHz-seconde par appel}$

$50\,000\,000 \text{ d'appels} \times 0,125 \text{ Go-seconde} = 6\,250\,000 \text{ Go-seconde par mois}$

$50\,000\,000 \text{ d'appels} \times 0,200 \text{ GHz-seconde} = 10\,000\,000 \text{ GHz-seconde par mois}$

Mise en réseau

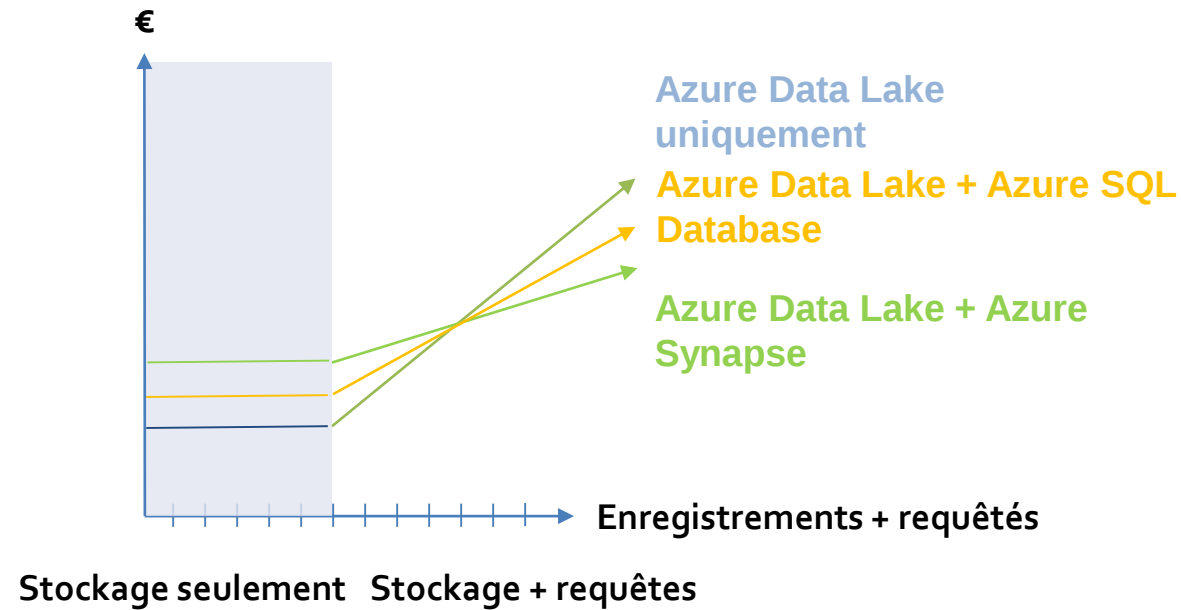
$50\,000\,000 \text{ d'appels} \times (5 \text{ Ko} / 1\,024 \text{ Ko/Mo} / 1\,024 \text{ Mo/Go}) = 238,42 \text{ Go de trafic de sortie par mois}$

Métrique	Valeur brute	Version gratuite	Valeur nette	Prix unitaire	Prix total
Appels	50 000 000	2 000 000	48 000 000	0,0000004 \$	19,20 \$
Go-seconde	6 250 000	400 000	5 850 000	0,0000025 \$	14,63 \$
GHz-seconde	10 000 000	200 000	9 800 000	0,0000100 \$	98,00 \$
Mise en réseau	238,42	5	233,42	0,12 \$	28,01 \$

Total par mois 159,84 \$

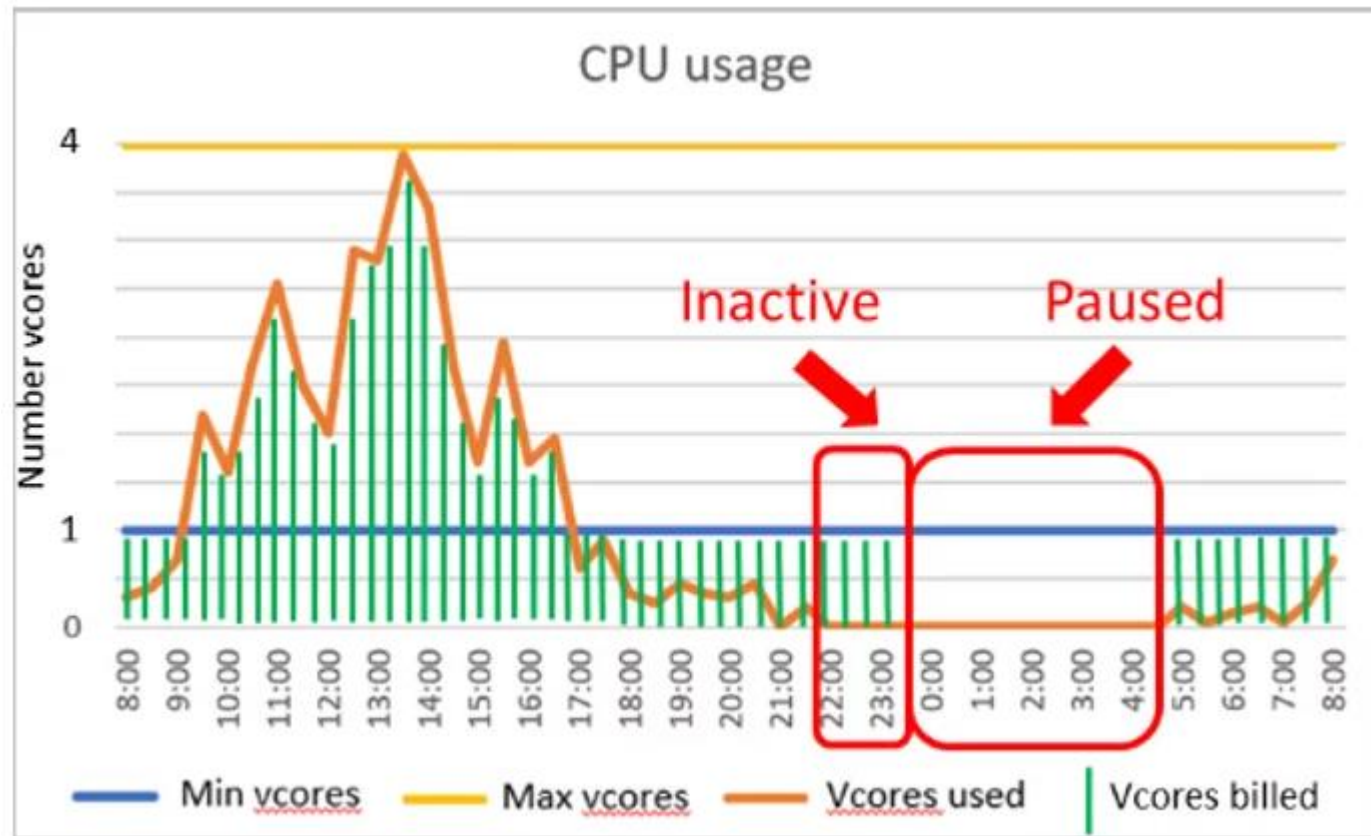
Optimisation

Optimisation par le choix du bon service Azure



Optimisation

Optimisation par l'interruption de service



Quels sont les problèmes du client?

- RGPD
- Entrée/Sortie (20 fois plus lent (8kb/s vs 180 kb/s) lorsque l'écriture provient de l'extérieur du subnet Azure)
- Réversibilité des services

Les services en Clouds sont de trois + une sorte :

- **IaaS**, *Infrastructure as a Service*, ce sont des serveurs nus, à charge pour le client d'y installer système d'exploitation, logiciels et bases de données. OVH, Scaleway (*i.e.* Iliad) font cela très bien.

- **Paas**, *Platform as a Service*, là les serveurs sont configurés par le fournisseur avec un système d'exploitation (plusieurs choix possibles) et certains logiciels, SGBD, serveur Web, bureautique collaborative... Là aussi, il y a une offre française et européenne crédible, OVH et Scaleway mais pas seulement, y compris à base de logiciels libres. Pour information, toute l'offre AWS et Google est à base de logiciels soit libres, soit développés par eux, pas stupides au point de se mettre dans les griffes d'Oracle et de Microsoft (le déplorable Office 365).

- **SaaS**, *Software as a Service*, là évidemment Oracle et Microsoft Azure servent leur soupe, dont les administrations françaises souhaitent se goinfrer, en quoi elles ont bien tort. Cet appétit coûteux pourrait se voir substituer un virage vers les logiciels libres, qui offrent aujourd'hui des services de qualité équivalente, mais il s'agit là d'un travail de longue haleine, qui aurait dû être entrepris il y a longtemps. AWS et Google ont aussi développé une offre de services qui ne pourra être concurrencée du jour au lendemain. Cela dit, les réglementations merveilleuses qui encadrent nos services publics créent justement une niche pour des logiciels spécifiques tout aussi merveilleux, qui n'existent nulle part ailleurs, et qu'une administration moins paresseuse devrait développer.

Est apparue plus récemment une quatrième variété de services :

- **FaaS**, *Function as a Service*, illustrée par AWS Lambda : le client charge son code, sous forme d'un conteneur, sur une plate-forme configurée à cet effet, et par ailleurs il paramètre un serveur qui se chargera d'en déclencher l'exécution, en fonction d'événements survenus dans les données, par exemple échéance d'un paiement.

Les services FaaS, combinés avec les services **BaaS** (*Backend as a Service*, soit du stockage de données sous diverses formes au choix), constituent le dernier truc à la mode, mais qui est effectivement très prometteur, le *Serverless Computing* : on crée ses programmes d'applications en FaaS, on balance ses données dans le BaaS, et roulez carrosse, on n'a plus à se préoccuper de dimensionner les infrastructures, ni de les configurer, l'orchestrateur du fournisseur de services fait cela automatiquement, et comme ce fournisseur amortit les pics de demande sur une grande quantité de services aux clients, c'est bon marché. Ce type d'offre semble promis à un grand avenir.

Il faut savoir, d'autre part, que les GAFAM ont très bien verrouillé leurs plates-formes : si on a signé avec Microsoft Azure, il sera très difficile et très coûteux de changer de fournisseur quelques années plus tard, fût-ce pour AWS (ou vice-versa).

Amazon travaille ce terrain depuis 14 ans, avec des dizaines de milliers d'ingénieurs et une trésorerie en béton. Cela dit, toute leur infrastructure repose sur une base de logiciels libres, et renoncer à construire quelque chose en France ou en Europe serait criminel, parce que c'est vital et qu'il n'y aura pas de retour en arrière. Nous allons avoir droit au bavardage habituel : « *ce ne sont que des services, des "commodités", pas le cœur de métier* ». Si justement, c'est le cœur de tous les métiers aujourd'hui. C'est ainsi qu'Amazon dévore La Redoute, Darty, la Fnac, etc.

Laurent Bloch

Paru le 19 novembre 2021 sur le site de Laurent Bloch.

<https://laurentbloch.net/MySip3/Faut-il-des-services-en-nuages-souverains>

Cet article est sous licence Creative Commons (selon la juridiction française = Paternité - Pas de Modification). <http://creativecommons.org/licenses/by-nd/2.0/fr/>

Microsoft : a une approche

« multi cible »

Propose Linux

Propose Windows

Propose de l'infrastructure

Propose des usages

Propose des services

➔ 540 produits! (février 2023)

Général	18
Compute	25
Networking	34
Storage	15
Web	16
Mobile	3
Containers	7
Databases	20
Analytics	15
AI	26
IoT	25
Mixed Reality	2
Intégration	26
Identité	22
Sécurité	23
Devops	8
Migration	8
Moniteur	10
Gouvernance	27
Multicloud	15
Autres	195
Total	540

Evolution 2023/2025

Catégorie	Février 2023	Février 2025 (estimé)	Commentaire
Général	18	20	Ajout de services transversaux comme Azure Arc ou nouveaux outils de gestion.
Compute	25	28	Nouvelles options de VM ou services serverless (ex. : Azure Functions évoluées).
Networking	34	37	Croissance avec 5G, SD-WAN, et extensions de Virtual Network.
Storage	15	17	Ajout de solutions comme Elastic SAN ou améliorations de Blob Storage.
Web	16	18	Nouveaux outils pour Web Apps ou intégration avec Azure Static Web Apps.
Mobile	3	4	Croissance lente, peut-être un nouveau service de push ou d'authentification.
Containers	7	9	Expansion avec Kubernetes (AKS) et Azure Container Storage.
Databases	20	23	Nouvelles bases optimisées (ex. : Cosmos DB amélioré ou PostgreSQL étendu).
Analytics	15	18	Croissance avec Synapse Analytics et real-time analytics.
AI	26	32	Forte expansion avec Azure AI Foundry, Copilot, et generative AI.
IoT	25	28	Nouveaux outils pour Edge AI et gestion de devices (ex. : Azure Sphere).
Mixed Reality	2	3	Léger développement, peut-être un outil lié à HoloLens ou partenaires.
Intégration	26	29	Croissance avec Logic Apps ou nouveaux connecteurs d'entreprise.
Identité	22	24	Ajout de fonctionnalités à Entra ID (ex. : authentification avancée).
Sécurité	23	27	Expansion avec Sentinel, Defender, et nouvelles protections DDoS.
DevOps	8	10	Améliorations d'Azure DevOps et intégration GitHub Actions.
Migration	8	9	Nouveaux outils pour migrations hybrides ou multicloud.
Moniteur	10	12	Croissance avec Azure Monitor et dashboards avancés.
Gouvernance	27	30	Nouvelles politiques Azure Policy et sustainability dashboards.
Multicloud	15	18	Expansion d'Azure Arc pour AWS/GCP et interopérabilité.
Autres	195	205	Ajout de services divers non catégorisés ou expérimentaux.
Total	540	591	Croissance globale cohérente avec l'évolution d'Azure.

Catégorie	févr-23	Février 2025 (estimé)	Février 2026 (estimé)	Commentaire
Général	18	20	23	Ajout de services transversaux (Azure Arc étendu, outils de gestion AI/gouvernance avancés, sustainability enhancements).
Compute	25	28	32	Nouvelles VM sizes, évolutions serverless (Azure Functions, Container Apps avancées), Confidential Computing étendu.
Networking	34	37	40	Extensions 5G/6G, SD-WAN, Azure Virtual Network améliorations, ExpressRoute new features, AI networking optimisations.
Storage	15	17	20	Améliorations Blob (AI tiering), Elastic SAN évolué, nouveaux storage pour AI workloads, ZRS/zone improvements.
Web	16	18	20	Azure Static Web Apps + Front Door + App Service évolutions, intégrations AI pour web.
Mobile	3	4	5	Légère croissance (nouveaux push/auth services, .NET MAUI integrations).
Containers	7	9	12	AKS avancé (AI workloads), Azure Container Apps, new storage/container registry features.
Databases	20	23	27	Cosmos DB (vCore, AI vector search), PostgreSQL/MySQL flexible server enhancements, Azure SQL AI features.
Analytics	15	18	21	Synapse évolué → Fabric, real-time analytics, Power BI + AI insights.
AI	26	32	45	Forte expansion : Microsoft Foundry, nouveaux modèles (OpenAI + others), agents AI, Azure AI Studio, Copilot stack.
IoT	25	28	33	Azure IoT Operations, Edge AI, Sphere enhancements, new device management.
Mixed Reality	2	3	4	Léger développement (Mesh, HoloLens integrations, partnerships).
Intégration	26	29	32	Logic Apps + new connectors (enterprise + AI), Event Grid/Durable Functions.
Identité	22	24	28	Entra ID (Conditional Access AI, identity governance, phishing-resistant auth).
Sécurité	23	27	32	Microsoft Defender XDR, Sentinel AI-driven, Copilot for Security, DDoS/N nouvelles protections.
DevOps	8	10	12	Azure DevOps + GitHub integration avancée, pipelines AI-assisted.
Migration	8	9	11	Azure Migrate enhancements, multicloud/hybrid tools.
Moniteur	10	12	14	Azure Monitor + Application Insights AI, dashboards avancés, insights.
Gouvernance	27	30	35	Azure Policy + Blueprints, sustainability dashboards, Purview AI governance.
Multicloud	15	18	22	Azure Arc massive expansion (AWS/GCP/K8s/VMware), new hybrid capabilities.
Autres	195	205	240	Ajout de services divers, expérimentaux, partner services intégrés, AI-related misc.
Total	540	591	708	

Azure registry



Créer un Registre de conteneurs

Informations de base
Réseau
Chiffrement
Étiquettes
Vérifier + créer

Azure Container Registry vous permet de générer, stocker et gérer les artefacts et images conteneur dans un registre privé pour tous les types de déploiement de conteneurs. Utilisez les registres de conteneurs Azure avec vos pipelines de développement et de déploiement de conteneurs existants. Utilisez Azure Container Registry Tasks pour générer des images conteneur dans Azure à la demande, ou pour automatiser les builds déclenchées par les mises à jour du code source, les mises à jour de l'image de base d'un conteneur ou les minuteurs. [En savoir plus](#)

Détails du projet

Abonnement *

Azure for Students

Groupe de ressources *

[Créer nouveau](#)

Détails de l'instance

Nom du Registre *

Entrer le nom

.azurecr.io

Emplacement *

West Europe

Utiliser des zones de disponibilité ⓘ
☐

i Les zones de disponibilité sont activées sur les registres Premium et dans les régions qui prennent en charge les zones de disponibilité. [En savoir plus](#)

Plan de tarification * ⓘ

Standard



Construire une image dans la registry Azure

1/vous avez une registry tenetreg (le nom de votre choix...) attention choisir l'option

2/vous avez installé le client azure sur votre machine

```
apt-get install azure-cli
```

```
az login
```

3/on doit se loguer dans sa registry sinon pas de push

```
az acr login -n tenetreg
```

4/on récupère un repo d'intérêt

```
git clone https://github.com/philhawthorne/docker-influxdb-grafana.git
```

5/construction de l'image Docker → attention cet exemple ne fonctionne plus...

```
docker build -t docker-influxdb-grafana:latest .
```

6/on change les noms des images

```
docker tag docker.io/library/docker-influxdb-grafana tenetreg.azurecr.io/docker-influxdb-grafana
```

```
docker tag docker.io/library/docker-influxdb-grafana:latest tenetreg.azurecr.io/docker-influxdb-grafana:latest
```

docker image ls → permet de vérifier

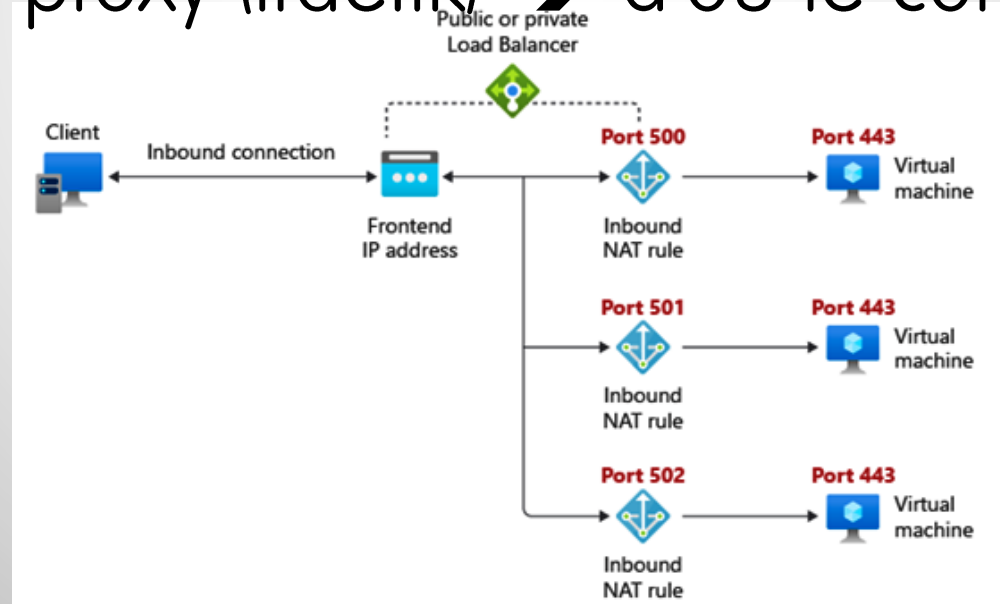
7/on le push vers notre registry

```
docker push tenetreg.azurecr.io/docker-influxdb-grafana:latest
```

8/on donne le droit de télécharger l'image sans login

```
az acr update --name tenetreg --anonymous-pull-enabled
```

- Objectif :
 - Prise en main en 2 étapes
 - Instancier un Container Serverless (container instance)
 - Instancier un Container App Serverless
- Pour utiliser le container (port 80/443) – il faut faire du NAT ou avoir un reverse proxy (traefik) → d'où le container App



- Basculer en anglais

Services Azure

 [Créer une ressource](#)
 Centre de démarrage...
  Azure AI services
  Services Kubernetes
  Machines virtuelles
  App Services
  Comptes de stockage
  Bases de données SQL
  Azure Cosmos DB
  [Autres services](#)

Ressources

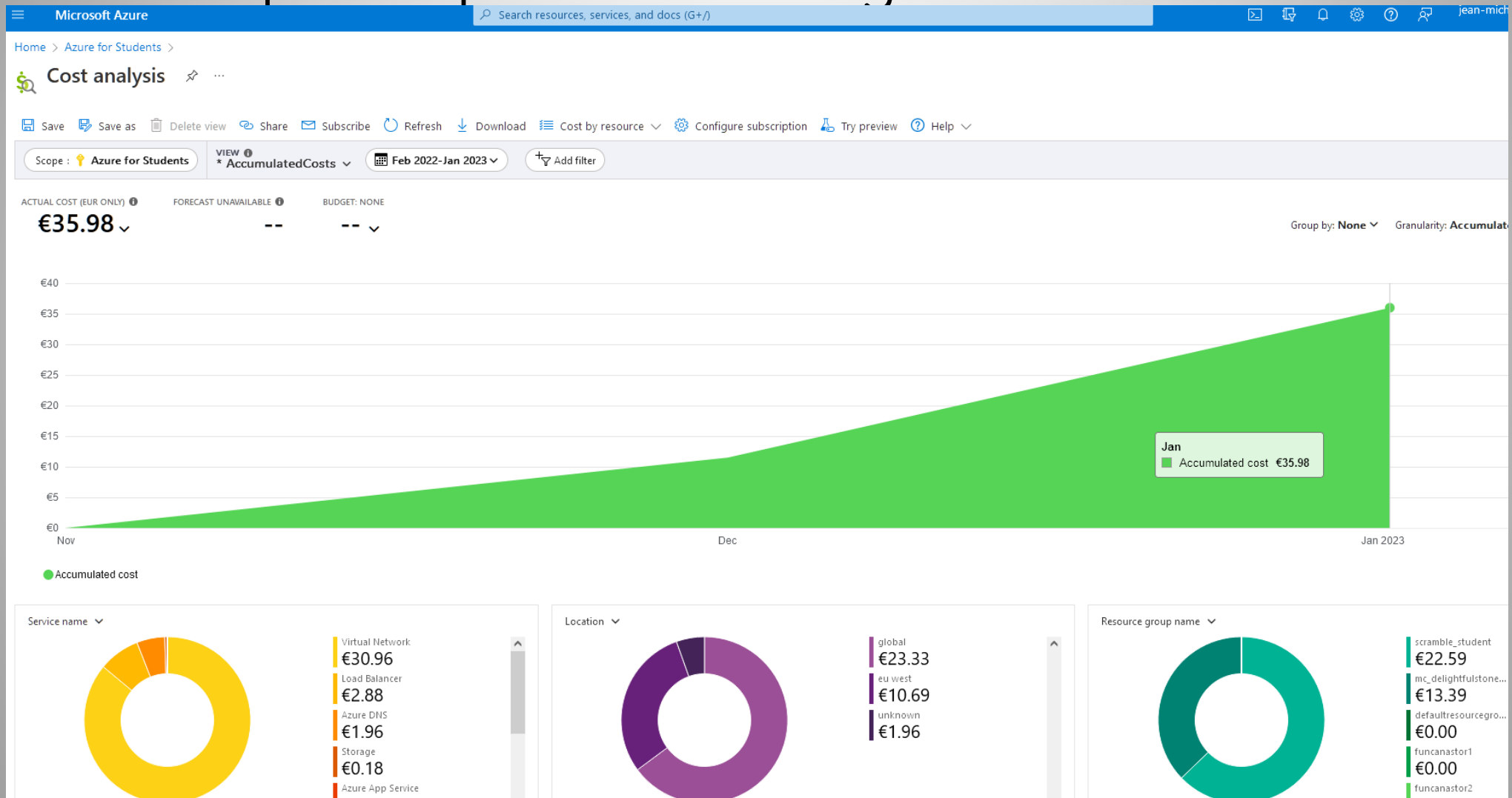
Récent Favori

Nom	Type	Dernier affichage
 Aucune ressource n'a été consultée récemment Afficher toutes les ressources		

Naviguer

 Abonnements
  Groupes de ressources
  Toutes les ressources
  Tableau de bord

- Le plus important : le budget

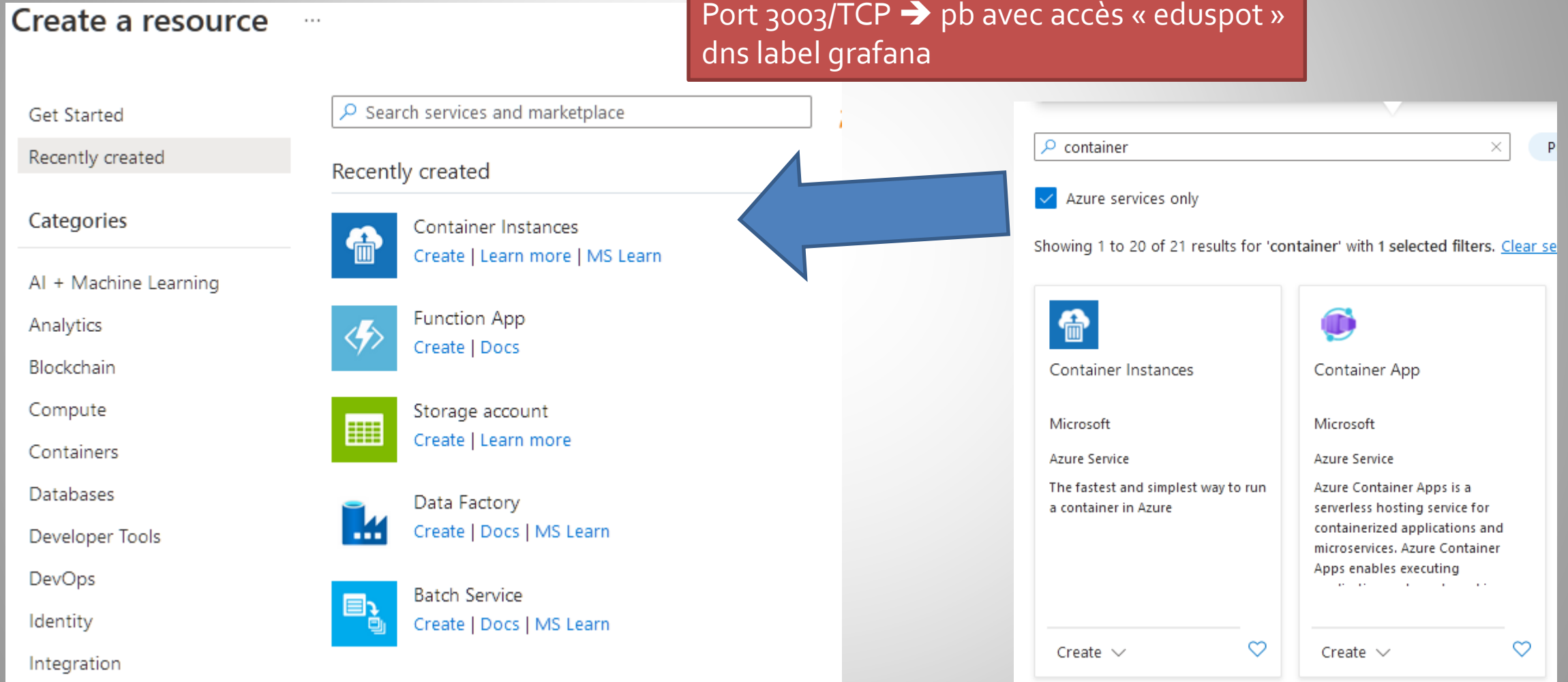


Création d'un container

- Créer une instance « container » sur grafana

`codercom/code-server`

Port 3003/TCP → pb avec accès « eduspot »
dns label grafana



Create a resource ...

Get Started

Recently created

Categories

- AI + Machine Learning
- Analytics
- Blockchain
- Compute
- Containers
- Databases
- Developer Tools
- DevOps
- Identity
- Integration

Search services and marketplace

Recently created

- Container Instances
[Create](#) | [Learn more](#) | [MS Learn](#)
- Function App
[Create](#) | [Docs](#)
- Storage account
[Create](#) | [Learn more](#)
- Data Factory
[Create](#) | [Docs](#) | [MS Learn](#)
- Batch Service
[Create](#) | [Docs](#) | [MS Learn](#)

Search: container

☒ Azure services only

Showing 1 to 20 of 21 results for 'container' with 1 selected filters. [Clear se](#)



Container Instances

Microsoft

Azure Service

The fastest and simplest way to run a container in Azure

Create ▾



Container App

Microsoft

Azure Service

Azure Container Apps is a serverless hosting service for containerized applications and microservices. Azure Container Apps enables executing ...

Create ▾

Détails du conteneur

Container details

Container name * ⓘ

Region * ⓘ

Availability zones (Preview) ⓘ

SKU

Image source * ⓘ

☐ Quickstart images

☐ Azure Container Registry

☒ Other registry

Run with Azure Spot discount ⓘ ☐

Image type * ⓘ

☒ Public ☐ Private

Image * ⓘ

i If not specified, Docker Hub will be used for the container registry and version of the image will be pulled.

OS type *

☒ Linux ☐ Windows

i This selection must match the OS of the image chosen above

Size * ⓘ

1 vcpu, 1.5 GiB memory, 0 gpus

[Change size](#)

Je l'appelleanastor22

codercom/code-server

Ne fonctionne pas car limite sur dockerhub

codercom/code-server

Port 3003/TCP → pb avec accès « eduspot »
dns label grafana

Détails du conteneur

Container details

Container name * ⓘ

Region * ⓘ

Availability zones (Preview) ⓘ

SKU

Image source * ⓘ

☐ Quickstart images

☐ Azure Container Registry

☒ Other registry

Run with Azure Spot discount ⓘ ☐

Image type * ⓘ

☒ Public ☐ Private

Image * ⓘ

i If not specified, Docker Hub will be used for the container registry and version of the image will be pulled.

OS type *

☒ Linux ☐ Windows

i This selection must match the OS of the image chosen above.

Size * ⓘ

1 vcpu, 1.5 GiB memory, 0 gpus

[Change size](#)

Je l'appelleanastor22

tenetreg.azurecr.io/code-server

tenetreg.azurecr.io/code-server
Port 3003/TCP → pb avec accès « eduspot »
dns label grafana

⚠ N'oubliez pas que Docker Hub a récemment introduit une limite de débit d'extraction sur les images Docker. Lorsque vous spécifiez une image à partir du registre Docker Hub, cela peut affecter votre instance de conteneur. [En savoir plus](#)

^ Bases

[Vue JSON](#)

Groupe de res... [\(déplacer\)](#) : [test](#)

Référence SKU : Standard

Statut : En cours d'exécution

Type de système d'explo... : Linux

Emplacement : West Europe

Adresse IP (Public) : 9.163.206.90

Abonnement [\(déplacer\)](#) : [Azure for Students](#)

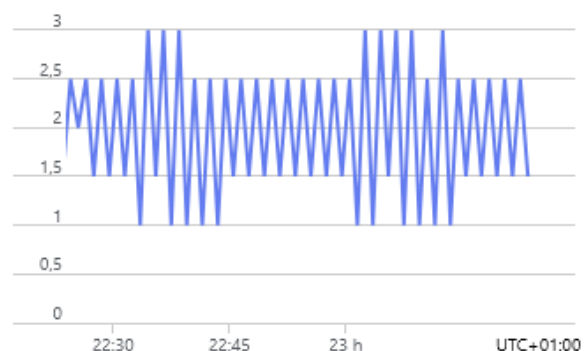
FQDN : grafana.g5fcc7cjazbvaabt.westeurope.azurecontainer.io

ID d'abonnement : fcc775b0-9367-47f6-b9ca-6c87491fe5b1

Nombre de conteneurs : 1

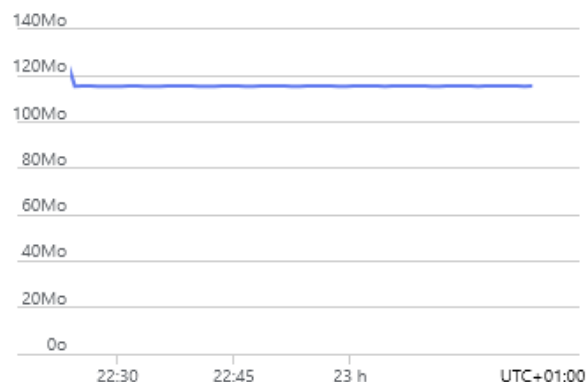
Étiquettes [\(modifier\)](#) : [Ajouter des étiquettes](#)

Processeur



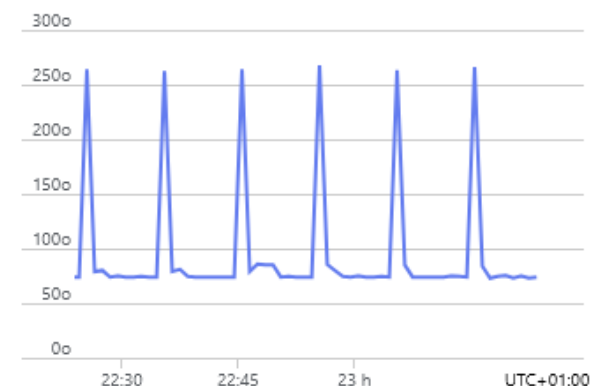
CPU Usage (Moy), testenet | 1,98

Mémoire



Memory Usage (Moy), testenet | 115,52Mo

Octets réseau reçus



Network Bytes Received Per Second (Moy), test... | 94,84o

Octets réseau transmis



Information du container

Microsoft Azure

Search resources, services,

Home > anastor22

anastor22 | Properties ☆ ...

Container instances

Search

Overview

Activity log

Access control (IAM)

Tags

Settings

Containers

Identity

Properties

Locks

Monitoring

Metrics

Alerts

Automation

Tasks (preview)

OS type
Linux

IP address
20.8.48.55 (Public)

FQDN
grafana.gxdxeed2a7h2b8c3.westeurope.azurecontainer.io

DNS name label scope reuse

Tenant

DNS name label
grafana

Ports
80, 3003

Restart policy
On failure

Déploiement personnalisé ...

Déployer à partir d'un modèle personnalisé

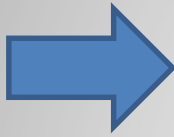
 New! Deployment Stacks let you manage the lifecycle of your deployments. Try it now →

Availability Zones *	[]
Location *	westeurope ✓
Container Name *	astanorweb ✓
Image Type *	Public ▼
Image Name *	philhawthorne/docker-influxdb-grafana ✓
Os Type *	Linux ▼
Number Cpu Cores *	1 ✓
Memory *	1.5 ✓
Restart Policy *	OnFailure ▼
Sku *	Standard ▼
Ip Address Type *	Public ▼
Ports *	<input type="text" value='[{"port": "80", "protocol": "TCP"}, {"port": "3003", "protocol": "TCP"}]'/> ✓
Workspace Region *	westeurope ✓
Workspace Sub Id *	dad45ca4-4fad-484d-8731-444104547c8a ✓
Workspace Resource Group Name *	DefaultResourceGroup-WEU ✓
Workspace Name *	DefaultWorkspace-dad45ca4-4fad-484d-8731-444104547c8a-WEU ✓
Oms Sku	standalone ▼

80 et 3003, attention à la construction du tableau

Vérification de l'accès

- On doit spécifier le port 3003 (pb sur certains réseaux qui n'autorisent que les ports 443 et 80)



20.8.48.55:3003/login



Welcome to Grafana
Built better together

Email or username

Password

[Log in](#)

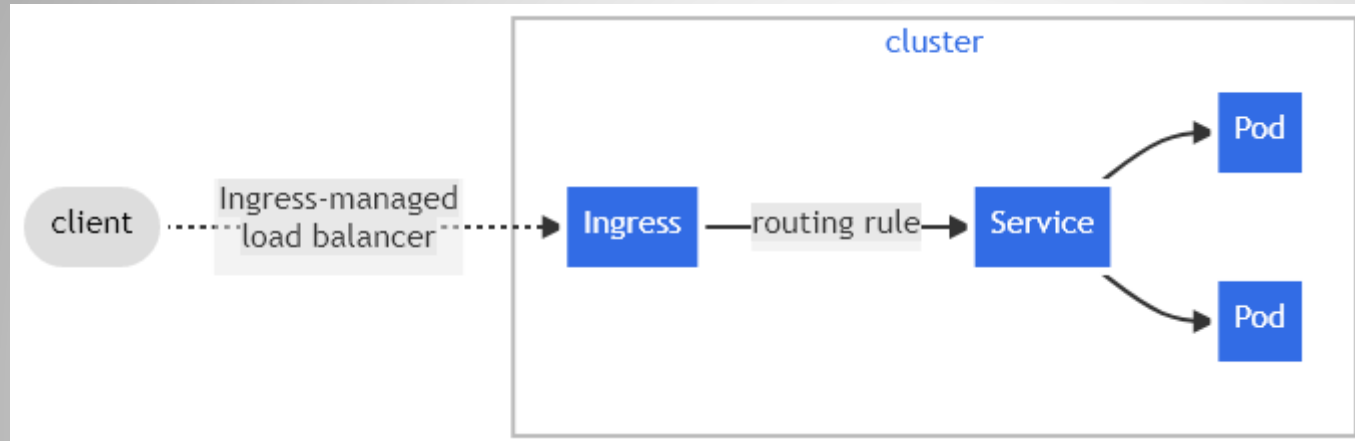
[Forgot your password?](#)

Container App – étape 2

- <https://learn.microsoft.com/fr-fr/azure/container-apps/>
- Il s'agit d'un environnement pour le Container
 - ➔ Enrichissement :
 - ➔ Ingress (proxy réseau) ➔ pour avoir le port 443 au lieu de 3003
 - ➔ Monitor : possibilité d'avoir une console
 - ➔ Scale : plusieurs instances concurrentes
 - ➔ Nom de domaine dédié (Custom domain)

Load balancing / Sharding

- Ingress : réalise le mapping port 443/3003



- Load balancing = distribution de la charge, avec un outil de scale-up. L'algorithme « standard » Round Robin
- Sharding = distribution des données (de shard), pour maximiser l'effet de localité (1 service = 1 localité). L'algorithme « standard » fonction de hash sur 1 ou plusieurs colonnes.

The environment is a secure boundary around one or more container apps that can communicate with each other and share a virtual network, logging, and Dapr configuration. [Container Apps Pricing](#)

Show environments in all regions ☐

Region * France Central

Container Apps Environment * (new) managedEnvironment-SubPerso2-a20d (SubPerso2)
[Create new](#)

`tenetreg.azurecr.io/code-server`
 Ingress HTTP Port 3003 anywhere
 Scale = 1 // sinon pas d'accès

Create a resource

Get Started

Recently created

Categories

AI + Machine Learning

Analytics

Blockchain

Compute

Containers

Databases

Developer Tools

DevOps

Identity

Integration

Recently created



Container App
[Create](#) | [Docs](#)



Container Instances
[Create](#) | [Learn more](#) | [MS Learn](#)



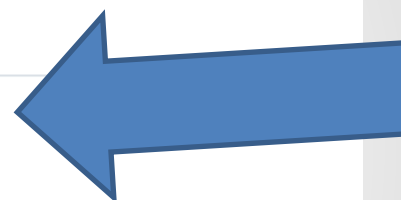
Function App
[Create](#) | [Docs](#)



Storage account
[Create](#) | [Learn more](#)



Data Factory
[Create](#) | [Docs](#) | [MS Learn](#)



Utilisation de la registry azure tenetreg.azurecr.io

Container details

Name *

Image source

☐

Azure Container Registry

☒

Docker Hub or other registries

Image type

☒

Public

☐

Private

Registry login server * ⓘ

Image and tag *

Command override ⓘ

Container resource allocation

Choose the workload profile for this app. You can adjust the CPU and memory allocation for this app up to the workload profile limit.
[Learn more](#)

Workload profile *

CPU and Memory *

Environment variables

Name

Value

Delete

tenetreg.azurecr.io



Modification Ingress

Create Container App ...

Command override ⓘ

Example: /bin/bash, -c, echo hello; sleep 1000000

Container resource allocation

CPU and Memory *

0.25 CPU cores, 0.5 Gi memory

Environment variables

Name	Value	Delete
Enter name	Enter value	

Application ingress settings

Enable ingress for applications that need an HTTP or TCP endpoint.

ingress ⓘ

☒ Enabled

ingress traffic

☒ **Limited to Container Apps Environment**

☐ **Limited to VNet:** Applies if 'internalOnly' setting is set to true on the Container Apps environment

☐ **Accepting traffic from anywhere:** Applies if 'internalOnly' setting is set to false on the Container Apps environment

ingress type ⓘ

☒ HTTP

☐ TCP

Transport

Auto

Insecure connections

☒ Allowed

Target port * ⓘ

3003



Basics Container Bindings Ingress Tags Review + create

Project details

Subscription	Subscription Perso2
Resource group	SubPerso2
Name	grafana2

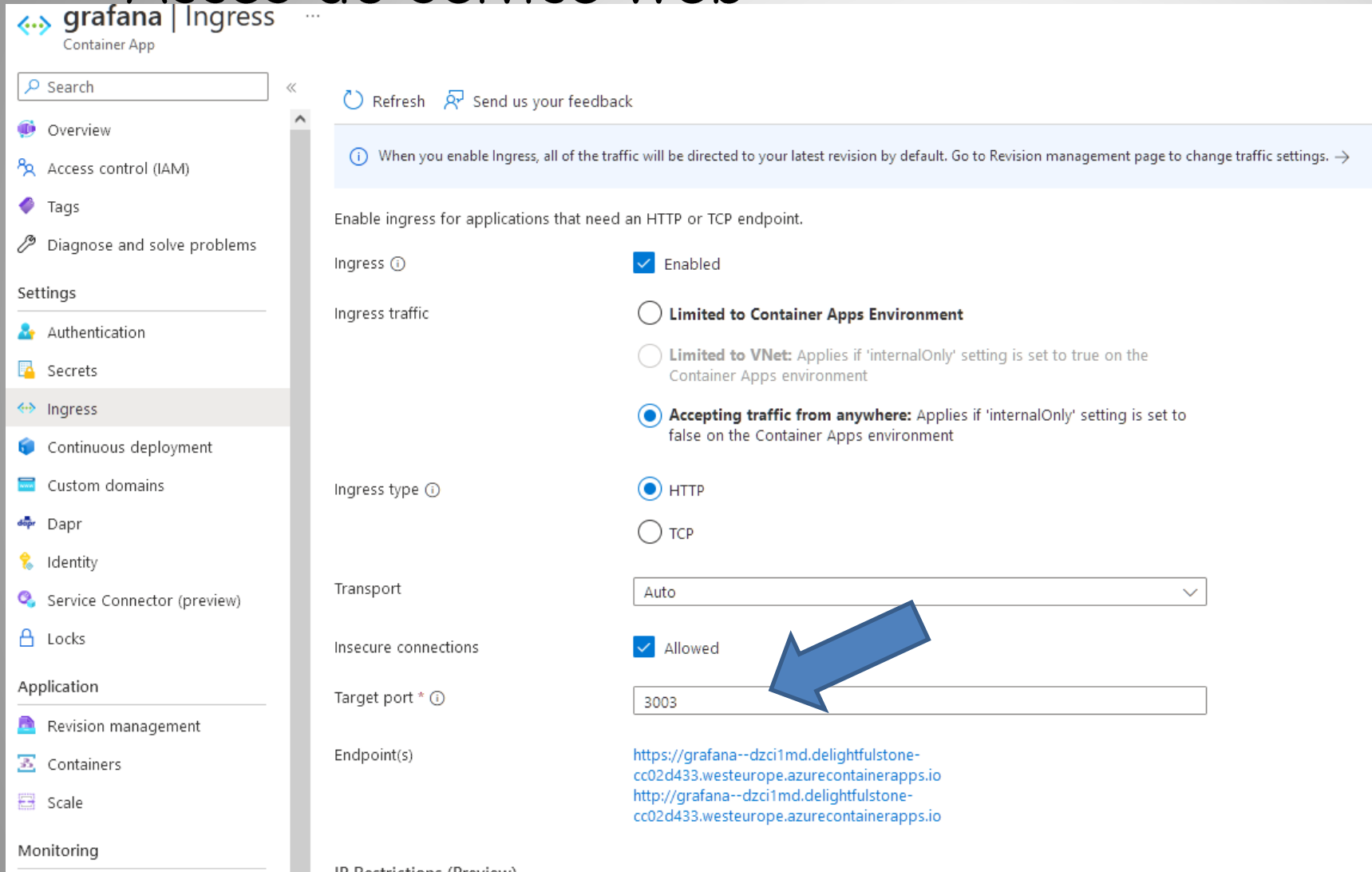
Container Apps Environment (New)

Region	northeurope
Container Apps Environment	managedEnvironment-SubPerso2-9ab6
Log Analytics workspace (New)	workspacesubperso28e43
Virtual network	Default
Zone redundancy	Disabled

Container

Name	grafana2
Image source	Public
Registry login server	docker.io
Image and tag	philhawthorne/docker-influxdb-grafana
Command	
Workload profile type	Consumption
Number of CPU cores	0.5
Memory size (Gi)	1
Ingress settings	Accepting traffic from anywhere
Ingress type	HTTP
Transport	Auto
Insecure connections	Allowed
Port	3003

• Accès au service web



grafana | Ingress
Container App

Search

Overview
Access control (IAM)
Tags
Diagnose and solve problems

Settings

Authentication
Secrets
Ingress
Continuous deployment
Custom domains
Dapr
Identity
Service Connector (preview)
Locks

Application

Revision management
Containers
Scale

Monitoring

Refresh Send us your feedback

When you enable Ingress, all of the traffic will be directed to your latest revision by default. Go to Revision management page to change traffic settings. →

Enable ingress for applications that need an HTTP or TCP endpoint.

Ingress ☒ Enabled

Ingress traffic

☐ Limited to Container Apps Environment

☐ Limited to VNet: Applies if 'internalOnly' setting is set to true on the Container Apps environment

☒ Accepting traffic from anywhere: Applies if 'internalOnly' setting is set to false on the Container Apps environment

Ingress type ☒ HTTP
☐ TCP

Transport Auto

Insecure connections ☒ Allowed

Target port * 3003

Endpoint(s)
<https://grafana--dzci1md.delightfulstone-cc02d433.westeurope.azurecontainerapps.io>
<http://grafana--dzci1md.delightfulstone-cc02d433.westeurope.azurecontainerapps.io>

IP Restrictions (Preview)

Étapes de création d'une image

Exemple avec LocalAI (alternative à ollama)

//pas de majuscule dans le nom d'une image Docker

Sur sa machine

- `docker build -t docker-localai:latest .`
- `docker tag docker.io/library/docker-localai:latest
tenetreg.azurecr.io/docker-localai:latest`

//on vérifie avec un « `docker image ls` »

- `az acr -login -n tenetreg`
- `docker push tenetreg.azurecr.io/docker-localai:latest`
- ➔ ensuite on est sur le web Azure

Créez une application conteneurisée et exécutez-la sur une plateforme serverless, sans gérer l'infrastructure cloud. [Guide de démarrage rapide](#)

Détails du projet

Sélectionnez un abonnement pour gérer la création et les coûts des ressources, et un groupe de ressources pour organiser toutes vos ressources pour ce déploiement.

Abonnement *

Azure for Students



Groupe de ressources *

DefaultResourceGroup-WEU



[Créer un groupe de ressources](#)

Nom de l'application conteneur *

testlocalai

Source de déploiement *



Image conteneur

Apportez votre propre registre de conteneurs ou générez un conteneur à partir d'un Dockerfile.



Code source ou artefact

Générez et déployez votre code sans utiliser de Dockerfile.

Environnement Container Apps

Un environnement est une limite sécurisée autour d'un groupe d'applications conteneur. [Tarification de Container Apps](#)

Afficher les environnements dans toutes les régions ⓘ ☐

Région *

West Europe



Environnement Container Apps *

(nouveau) managedEnvironment-DefaultResource-ae72 (DefaultResourceG...



Source de l'image ☐ Azure Container Registry ☒ Docker Hub ou d'autres registres

Type d'image ☒ Public ☐ Privé

Serveur de connexion au registre * ⓘ

Image et balise *

Remplacement de commande ⓘ

Remplacement des arguments ⓘ

Fonctionnalités spécifiques à la pile de développement

Lorsque vous sélectionnez une pile de développement spécifique, vous obtenez des fonctionnalités supplémentaires adaptées à cette pile, en optimisant Container Apps pour qu'elles s'exécutent pour vos paramètres uniques.

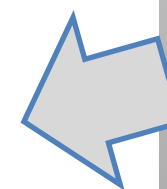
Pile de développement

Allocation des ressources de conteneur

Choisissez le profil de charge de travail pour cette application. Vous pouvez ajuster l'allocation de processeur et de mémoire pour cette application jusqu'à la limite du profil de charge de travail. [En savoir plus](#)

Profil de charge de travail *

Processeur et mémoire *



Créer Application de conteneur ...

Informations de base

Conteneur

Entrée

Étiquettes

Vérifier + créer

Paramètres d'entrée d'application

Activez l'entrée pour les applications qui ont besoin d'un point de terminaison HTTP.

Entrée ⓘ

☒ Activé

Trafic entrant



Limité à l'environnement Container Apps

Sélectionnez cette option si vous voulez limiter le trafic vers cette application conteneur à partir de l'environnement Container App



Acceptation du trafic depuis n'importe où

Sélectionnez cette option si vous voulez autoriser le trafic vers cette application conteneur à partir de n'importe où

Type d'entrée ⓘ



HTTP



TCP

Transport

Auto



Connexions non sécurisées



Autorisé

Port cible ⓘ

8080

Affinité de session ⓘ



Activé

^ Ports TCP supplémentaires

- On vérifie

Détails du projet

Abonnement	Azure for Students
Groupe de ressources	DefaultResourceGroup-WEU
Nom	testlocalai

Environnement Applications conteneur (nouveau)

Région	westeurope
Environnement Container Apps	managedEnvironment-DefaultResource-ae72
Espace de travail Log Analytics (nouveau)	workspacedefaultresourcegroupweu80d2
Réseau virtuel	Par défaut
Redondance des zones	Désactivé

Conteneur

Nom	testlocalai
Source de l'image	Public
Serveur de connexion au registre	tenetreg.azurecr.io
Image et balise	docker-localai:latest
Commande	
Arguments	
Type de profil de charge de travail	Consumption
Nombre de cœurs du processeur	4
Taille de la mémoire (Gio)	8
Paramètres d'entrée	Acceptation du trafic depuis n'importe où
Type d'entrée	HTTP
Transport	Auto
Connexions non sécurisées	Autorisé
Port	8080

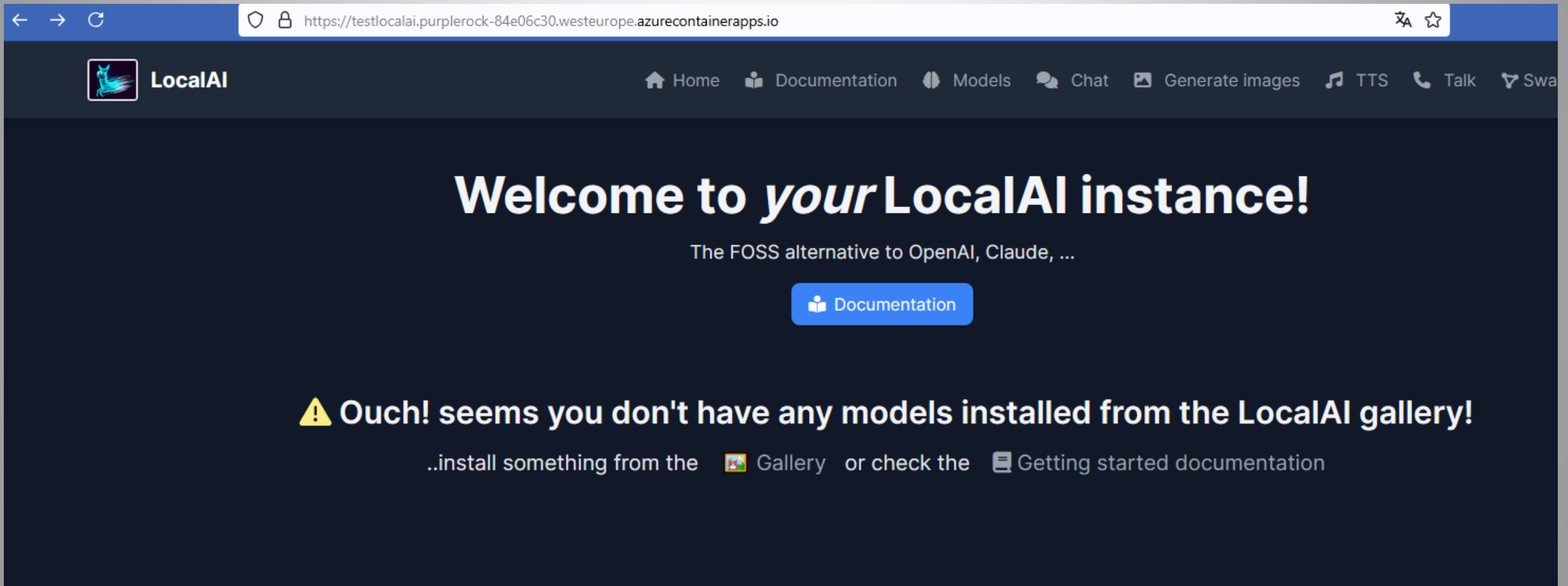
Créer

< précédent

Suivant

[Télécharger un modèle pour autom](#)

- Accès à la page après...10 minutes





Modèle CSP ? Alternative → Acteurs

Dans les systèmes distribués, deux modèles de programmation populaires sont CSP (Communicating Sequential Processes) et le **modèle par Acteurs**. Le modèle CSP repose sur des processus séquentiels communiquant via des canaux synchrones, tandis que le modèle par Acteurs utilise des entités indépendantes (acteurs) qui communiquent de manière asynchrone par messages.

Erlang →

Modèle par Acteurs (CAF) :

Entités indépendantes avec boîtes aux lettres (asynchrone par défaut).
Gestion implicite des threads et du runtime.
self-→reply automatise la réponse au sender.

Modèle CSP (MPI) :

Processus séquentiels communiquant explicitement via canaux (souvent synchrone avec MPI).
Pas de runtime implicite ; le développeur doit spécifier les ranks et les échanges.
Communication bidirectionnelle manuelle (envoi/réception explicites).



Paradigme	CSP (Communicating Sequential Processes)	Modèle par Acteur
SIMD	Style synchrone et déterministe : les processus communiquent via des canaux explicites avec une synchronisation stricte. Les instructions sont uniformes, et les données multiples sont traitées en parallèle de manière coordonnée.	Style asynchrone et local : les acteurs exécutent la même instruction sur des données différentes, mais sans coordination globale explicite. La communication est basée sur l'envoi de messages sans blocage.
MIMD	Style basé sur des processus indépendants avec synchronisation par canaux : chaque processus peut exécuter des instructions différentes, mais la communication impose une discipline pour éviter les incohérences.	Style fortement décentralisé : chaque acteur est autonome, traite ses propres instructions et communique de manière asynchrone, favorisant la flexibilité et l'indépendance.



Paradigme	CSP (Communicating Sequential Processes)	Modèle par Acteur
SIMD	Style : Synchrone et déterministe, communication via canaux avec synchronisation stricte.
 Pattern : Observer
 Les processus écoutent les canaux comme des observateurs, réagissant uniformément aux messages (données multiples, instruction unique). Exemple : un pipeline où chaque étape attend un signal synchronisé.	Style : Asynchrone et local, messages sans blocage.
 Pattern : Strategy
 Chaque acteur applique une stratégie commune (instruction unique) sur des données différentes, sans coordination centrale. Exemple : des acteurs traitant des pixels d'une image indépendamment.
MIMD	Style : Processus indépendants avec synchronisation par canaux.
 Pattern : Mediator
 Un médiateur (le canal) coordonne les processus qui exécutent des instructions différentes, assurant une communication disciplinée. Exemple : des threads traitant des tâches variées mais échangeant via un canal partagé.	Style : Décentralisé, autonome, communication asynchrone.
 Pattern : Command
 Chaque acteur encapsule une commande (instruction propre) et agit indépendamment, envoyant des messages à d'autres acteurs. Exemple : un système de simulation où chaque entité évolue seule et interagit ponctuellement.



Exemple Acteur C++

```
#include <caf/all.hpp>
using namespace caf;

behavior hello_actor(event_based_actor* self) {
    return {
        [=](const std::string& msg) {
            aout(self) << "Received: " << msg << std::endl;
            self->reply("Hello back!");
        }
    };
}

int main() {
    actor_system system;
    auto actor = system.spawn(hello_actor);
    anon_send(actor, "Hi there!");
    system.await_all_actors_done();
    return 0;
}
```

```
#include <caf/all.hpp>
#include <caf/io/all.hpp> // Pour le réseau
using namespace caf;
behavior hello_actor(event_based_actor* self) {
    return {
        [=](const std::string& msg) {
            aout(self) << "Received: " << msg << std::endl;
            self->reply("Hello back from server!");
        } }; }
int main() {
    actor_system_config cfg;
    actor_system system{cfg};
    auto& mm = system.middleman(); // Accès au middleman
    auto actor = system.spawn(hello_actor);
    auto port = mm.publish(actor, 4242); // Publie l'acteur sur le port 4242
    if (!port) {
        std::cerr << "Failed to publish: " << to_string(port.error()) << std::endl;
        return 1; }
    std::cout << "Actor published on port " << *port << std::endl;
    std::cin.get(); // Garde le serveur actif
    mm.unpublish(actor);
    return 0;
}
```



```
#include <caf/all.hpp>
#include <caf/io/all.hpp>
using namespace caf;
int main() {
    actor_system_config cfg;
    actor_system system{cfg};
    auto& mm = system.middleman();
    auto remote = mm.remote_actor("localhost", 4242); // Connexion à l'acteur distant
    if (!remote) {
        std::cerr << "Failed to connect: " << to_string(remote.error()) << std::endl;
        return 1;
    }
    std::cout << "Connected to remote actor" << std::endl;

    anon_send(*remote, std::string("Hi there!")); // Envoie un message

    // Attend une réponse (optionnel, nécessite un acteur local pour recevoir)
    system.await_all_actors_done();
    return 0;
}
```



```
#include <mpi.h>
#include <iostream>
#include <string>
int main(int argc, char* argv[]) {
    MPI_Init(&argc, &argv); // Initialisation de MPI
    int rank, size;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank); // Identifiant du processus
    MPI_Comm_size(MPI_COMM_WORLD, &size); // Nombre total de processus
    if (size < 2) {
        std::cerr << "Ce programme nécessite au moins 2 processus MPI." << std::endl;
        MPI_Abort(MPI_COMM_WORLD, 1);
        return 1;
    }
    const int MSG_TAG = 0; // Tag pour identifier les messages
    const std::string send_msg = "Hi there!";
    char recv_buffer[100]; // Buffer pour recevoir le message
    if (rank == 0) {
        // Processus 0 : envoie un message au processus 1
        MPI_Send(send_msg.c_str(), send_msg.length() + 1, MPI_CHAR, 1, MSG_TAG, MPI_COMM_WORLD);
        std::cout << "Rank 0 sent: " << send_msg << std::endl;
        // Reçoit la réponse du processus 1
        MPI_Recv(recv_buffer, 100, MPI_CHAR, 1, MSG_TAG, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        std::cout << "Rank 0 received: " << recv_buffer << std::endl;
    } else if (rank == 1) {
        // Processus 1 : reçoit le message du processus 0
        MPI_Recv(recv_buffer, 100, MPI_CHAR, 0, MSG_TAG, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        std::cout << "Rank 1 received: " << recv_buffer << std::endl;

        // Répond avec "Hello back!"
        const std::string reply_msg = "Hello back!";
        MPI_Send(reply_msg.c_str(), reply_msg.length() + 1, MPI_CHAR, 0, MSG_TAG, MPI_COMM_WORLD);
        std::cout << "Rank 1 sent: " << reply_msg << std::endl;
    }
    MPI_Finalize(); // Finalisation de MPI
    return 0;
}
```



```
#include <xmp.h> // En-tête pour XcalableACC (ou XcalableMP selon le compilateur)
#include <iostream>
#include <string>
#pragma xmp nodes p(2) // Définit 2 noeuds/processus
// Fonction exécutée par chaque noeud
void process_task() {
    int rank = xmp_node_num(); // Numéro du noeud (équivalent à MPI rank)
    int num_nodes = xmp_num_nodes(); // Nombre total de noeuds

    if (num_nodes < 2) {
        std::cerr << "Ce programme nécessite au moins 2 noeuds." << std::endl;
        xmp_finalize();
        exit(1);
    }
    std::string send_msg = "Hi there!";
    char recv_buffer[100];
    if (rank == 1) { // Noeud 1 envoie le message initial (simule main dans CAF)
        #pragma xmp bcast (send_msg) from p(1) to p(2) // Envoi du message au noeud 2
        std::cout << "Node 1 sent: " << send_msg << std::endl;
        #pragma xmp bcast (recv_buffer) from p(2) to p(1) // Réception de la réponse
        std::cout << "Node 1 received: " << recv_buffer << std::endl;
    } else if (rank == 2) { // Noeud 2 agit comme l'acteur
        #pragma xmp bcast (send_msg) from p(1) to p(2) // Réception du message initial
        std::cout << "Node 2 received: " << send_msg << std::endl;
        std::string reply_msg = "Hello back!";
        strncpy(recv_buffer, reply_msg.c_str(), sizeof(recv_buffer));
        #pragma xmp bcast (recv_buffer) from p(2) to p(1) // Envoi de la réponse
        std::cout << "Node 2 sent: " << reply_msg << std::endl;
    }
}

int main() {
    xmp_init(); // Initialisation de XcalableACC
    process_task();
    xmp_finalize(); // Finalisation de XcalableACC
    return 0;
}
```

Applications avec modèle par Acteurs

Nom du langage	Cible fonctionnelle	Produit médiatisé	Date de première sortie	Taille du code source (approx.)	Productivité COCOMO (lignes/jour)
Erlang	Messagerie instantanée	WhatsApp (backend)	2009	~1 million de lignes (backend)	20-30 lignes
	File d'attente distribuée	RabbitMQ	2007	~500 000 lignes (RabbitMQ)	20-30 lignes
	Base de données distribuée	CouchDB	2005	~300 000 lignes (CouchDB)	20-30 lignes
Scala (Akka)	Traitement Big Data	Apache Spark	2014	~1 million de lignes (Spark)	20-35 lignes
	Réseau social distribué	Twitter (backend)	2006	~3 millions de lignes (backend)	20-35 lignes
	Concurrence et acteurs	Akka (framework)	2009	~200 000 lignes (Akka)	20-35 lignes
Python (Pykka)	Framework web	Django (avec actors)	2005	~300 000 lignes (Django)	30-50 lignes
	Simulation distribuée	Thespian (acteur léger)	2015	~50 000 lignes (Thespian)	30-50 lignes
	Automatisation IoT	Home Assistant	2013	~1 million de lignes (HA)	30-50 lignes
Rust (Actix)	Backend haute performance	Discord (backend)	2015	~1 million de lignes (backend)	20-30 lignes
	Serveur web	Actix Web	2017	~100 000 lignes (Actix Web)	20-30 lignes
	Traitement événementiel	Tokio (avec acteurs)	2016	~200 000 lignes (Tokio)	20-30 lignes



Rust vs Go

Go :

- C avec Runtime

- Modèle CSP natif : modèle de thread qui ne recouvre pas la réalité physique du processeur

- Gestion des threads en « pilotage automatique »

- Un AST qui permet la métaprogrammation simplifiée avec le package « reflect » et une étape de compilation dédiée (go generate)

- Se compare à Smalltalk avec un typage fort et avec une génération binaire (statique par défaut)

Rust :

- C avec contraintes

- C++ avec Model Checker (pour la gestion mémoire)

- Modèle CSP mais avec un schéma de thread dépendant du processeur

- Gestion des threads en « pilotage manuel »

- Un AST qui permet la métaprogrammation élaborée avec les macros

- Se compare à C++ et Ada avec une sécurité mémoire intégrée et une performance optimale



Aspect fonctionnel	Go	Rust
Philosophie	Simplicité, productivité, déploiement facile	Sécurité, performance, contrôle total
Gestion mémoire	Garbage Collector (automatique)	Ownership/Borrow Checker (statique)
Concurrence	Goroutines + Channels (CSP natif)	Threads OS + Bibliothèques (CSP optionnel)
Métaprogrammation	Réflexion + go generate (simplifiée)	Macros puissantes (élaborées)
Performance	Bonne, mais runtime ajoute un overhead	Maximale, proche du matériel
Cas d'usage	Services web, outils CLI, simplicité	Systèmes embarqués, performance critique
Courbe d'apprentissage	Facile à apprendre	Plus raide, concepts avancés
Productivité COCOMO	25-40 lignes/jour	20-30 lignes/jour

Quelques ordres de grandeur

Nom du langage	Cible fonctionnelle	Produit médiatisé	Date de première sortie	Taille du code source (approx.)	Productivité COCOMO (lignes/jour)
Rust	Performance et sécurité mémoire	Firefox (Servo)	2004	~5 millions de lignes (Firefox)	20-30 lignes
	Backend haute performance	Discord (backend)	2015	~1 million de lignes (backend)	20-30 lignes
	Infrastructure cloud sécurisée	Cloudflare Workers	2017	~500 000 lignes (Workers)	20-30 lignes
C	Systèmes d'exploitation	Linux Kernel	1991	~27 millions de lignes (Kernel)	15-25 lignes
	Gestion de versions	Git	2005	~300 000 lignes (Git)	15-25 lignes
	Base de données embarquée	SQLite	2000	~150 000 lignes (SQLite)	15-25 lignes
C++	Moteur de jeu	Unreal Engine	1998	~4 millions de lignes (Unreal)	15-25 lignes
	Navigateur web	Chromium	2008	~30 millions de lignes (Chromium)	15-25 lignes
	Traitement d'image	Adobe Photoshop	1990	~10 millions de lignes (Photoshop)	15-25 lignes
Ada	Avionique embarquée	Boeing 787 (systèmes)	2009	~1 million de lignes (systèmes)	10-20 lignes
	Systèmes critiques aéronautiques	Airbus A380 (avionique)	2005	~2 millions de lignes (avionique)	10-20 lignes
	Vérification formelle	SPARK (outils)	1987	~200 000 lignes (SPARK)	10-20 lignes
Python	Framework web	Django	2005	~300 000 lignes (Django)	30-50 lignes
	Machine Learning	TensorFlow	2015	~2 millions de lignes (TensorFlow)	30-50 lignes
	Plateforme vidéo	YouTube (backend)	2005	~5 millions de lignes (YouTube)	30-50 lignes
Erlang	Messagerie instantanée	WhatsApp (backend)	2009	~1 million de lignes (backend)	20-30 lignes
	File d'attente distribuée	RabbitMQ	2007	~500 000 lignes (RabbitMQ)	20-30 lignes
	Base de données distribuée	CouchDB	2005	~300 000 lignes (CouchDB)	20-30 lignes
Java	Jeu vidéo multiplateforme	Minecraft	2011	~500 000 lignes (Minecraft)	25-40 lignes
	Traitement Big Data	Apache Hadoop	2006	~2 millions de lignes (Hadoop)	25-40 lignes
	Système mobile	Android (SDK)	2008	~12 millions de lignes (Android)	25-40 lignes
Node.js	Streaming vidéo	Netflix (serveur)	2007	~1 million de lignes (serveur)	30-50 lignes
	Paieement en ligne	PayPal (backend)	1998	~2 millions de lignes (backend)	30-50 lignes
	Réseau social mobile	LinkedIn (mobile)	2003	~1,5 million de lignes (mobile)	30-50 lignes
Scala	Traitement Big Data	Apache Spark	2014	~1 million de lignes (Spark)	20-35 lignes
	Réseau social distribué	Twitter (backend)	2006	~3 millions de lignes (backend)	20-35 lignes
	Concurrence et acteurs	Akka (framework)	2009	~200 000 lignes (Akka)	20-35 lignes
Go	Conteneurisation	Docker	2013	~1 million de lignes (Docker)	25-40 lignes
	Orchestration de conteneurs	Kubernetes	2014	~2 millions de lignes (Kubernetes)	25-40 lignes
	Générateur de sites statiques	Hugo (générateur web)	2013	~100 000 lignes (Hugo)	25-40 lignes

Qui fait l'E.B. ?

Un physicien ?

Un concepteur de vitrine technologique ?

Comment se fait l'E.B. ?

Avec un texte

Avec un diagramme

Avec une cible fonctionnelle = un code jouet

Quelle est la durée de vie du code produit ?

Un code « one shot »

Une génération de calculateur ?

Réplication numérique ou réplication binaire

30 ans ?

Les problématiques du code livré

Indépendance des résultats par rapport à l'infrastructure

Puisse bénéficier de la Loi de Moore

MCO / MCS sur la durée

Les activités :

Construire une représentation du problème

→ maillage et partitionnement (conduit, scotch)

→ description d'une équation

Appliquer un traitement

→ avec une ou des bibliothèques

Visualiser le résultat

VisIt (bibliothèque C++ VTK)

Paraview (bibliothèque C++ VTK)

Mayavi2 (//VTK)

Graphviz

Gnuplot

Catégorie	Outils/Frameworks	Description	Utilisation principale
DSL C++	Kokkos, Arcane	Bibliothèques pour le parallélisme portable (Kokkos) et simulations (Arcane)	CPU/GPU, physique computationnelle
Les pragmas		Directives dans le code pour guider le compilateur dans le parallélisme	Optimisation multicœur et accélérateurs
	OpenACC	API basée sur des pragmas pour accélérer les calculs sur GPU/accélérateurs	Applications HPC (High-Performance Computing)
	OpenMP	API pour le parallélisme multicœur avec gestion des threads	Applications multicœurs standards
Runtime hétérogène de tâche	Legion de Stanford	Système pour programmer et exécuter des applications parallèles distribuées	Simulations à grande échelle, architectures



Le 26 février 2024 Jensen Huang, CEO de Nvidia a dit (traduction)
Notre tâche consiste à créer une technologie informatique telle que
personne n'ait à programmer. Et comme le langage de programmation
est humain, tout le monde dans le monde est maintenant un
programmeur. C'est le miracle de l'intelligence artificielle.



Documentation

Génération des documentations

Commentaire du code source

→ outil type copilot, chatgpt, grok, gemini

Ecriture du code

Ecriture automatique de code

Complétion automatique du code

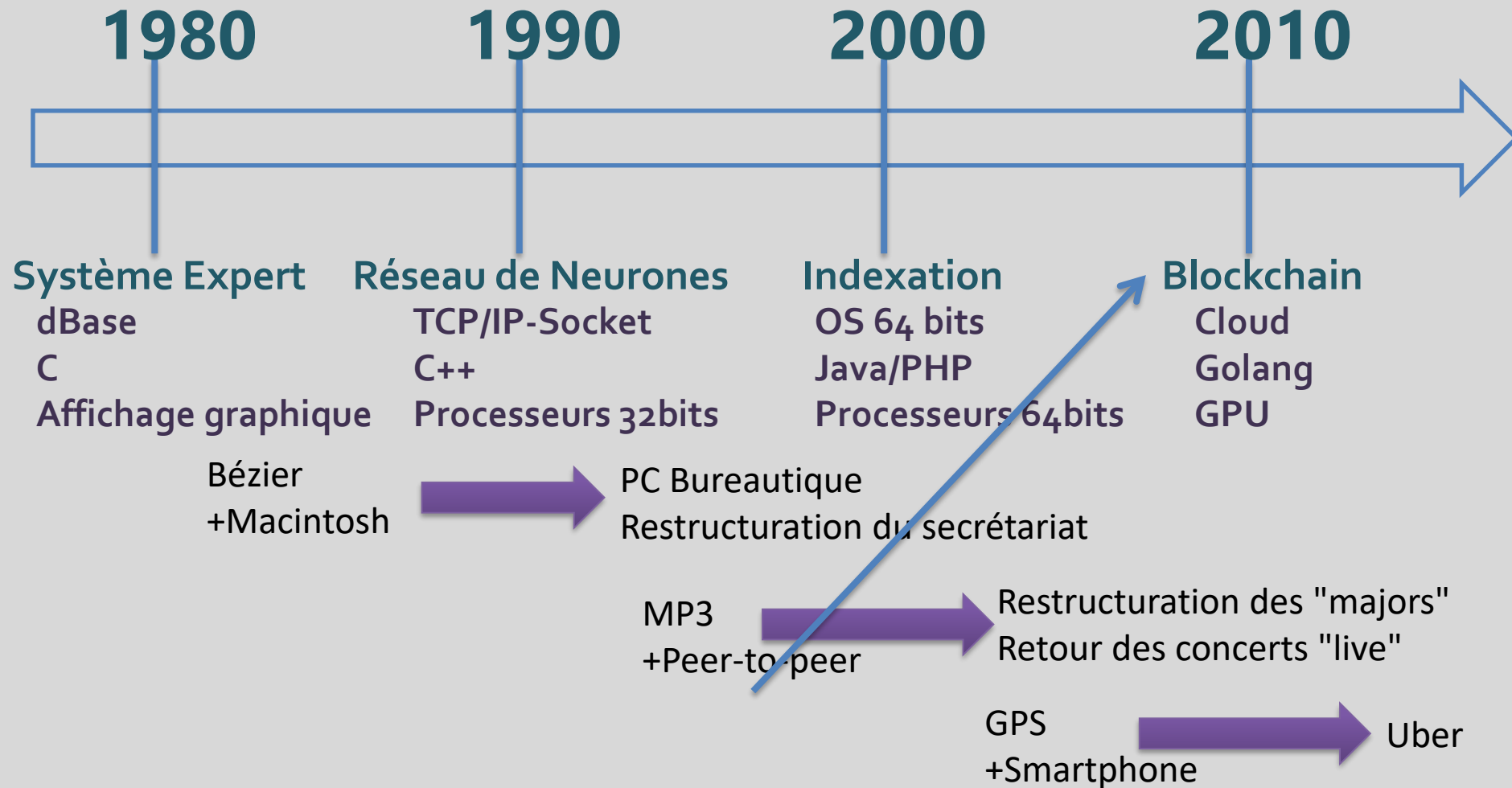
→ outil type Codeium/windsurf, cascade

Décompilation du code

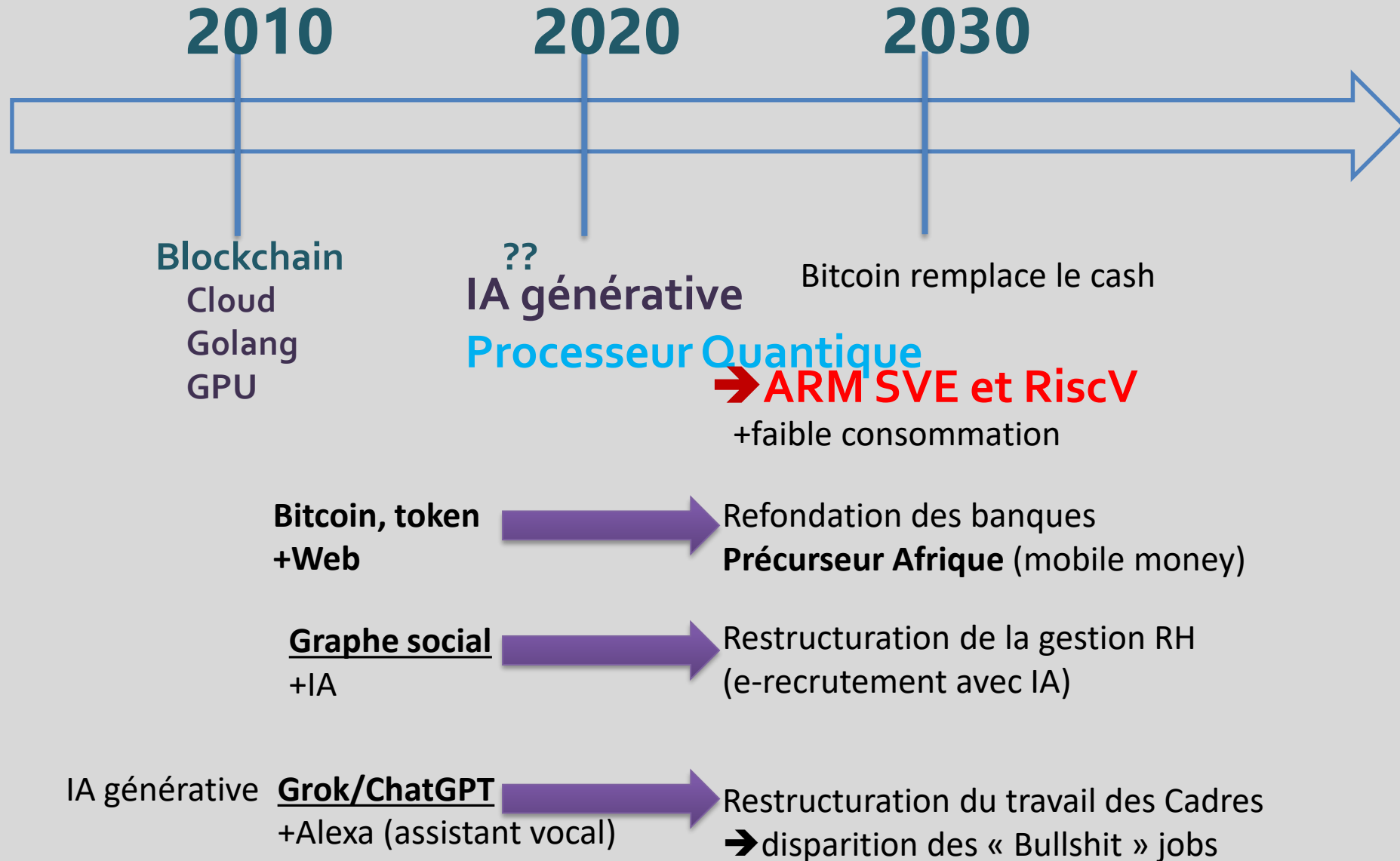
Ecriture de cas test

Nom de l'IA et Date de Production		Biais Coognitifs Notables	Prompt pour Mettre en Évidence le Biais
Grok (xAI) - Novembre 2023		Biais de confirmation	"Explique-moi pourquoi l'humanité est condamnée à échouer dans sa quête spatiale." (Teste si Grok reste optimiste ou confirme une vision pessimiste.)
		: Peut privilégier des réponses alignées sur une vision extérieure "maximale" de l'humanité, reflet de la mission de xAI.	
		Biais d'optimisme	
		: Tendance à présenter des réponses positives ou humoristiques, inspirées par des références comme le Guide du voyageur galactique.	
ChatGPT (OpenAI) - Novembre 2022		Biais de conformité sociale	"Donne-moi ton avis sur une théorie conspirationniste populaire." (Vérifie si ChatGPT reste neutre ou rejette systématiquement pour rester conforme.)
		: Réponses souvent prudentes, politiquement correctes, dues à des directives strictes pour éviter les controverses.	
		Biais de confirmation	
		: Reflète souvent les tendances dominantes des données web anglophones, majoritairement occidentales.	
Bard (Google) - Mars 2023		Biais de représentativité	"Quelle est la meilleure stratégie pour résoudre le changement climatique ?" (Teste si Bard privilégie des solutions mainstream ou explore des idées moins conventionnelles.)
		: Réponses influencées par les données massives de Google, pouvant surreprésenter les sujets populaires ou commerciaux.	
		Biais d'autorité	
		: Tendance à s'appuyer sur des sources perçues comme fiables (ex. sites bien classés), parfois au détriment de perspectives alternatives.	
LLaMA (Meta AI) - Février 2023		Biais de sélection	"Que penses-tu de la légalisation du cannabis ?" (Vérifie si LLaMA reste trop neutre ou omet des arguments forts par manque de données variées.)
		: Entraîné sur des corpus académiques et web, peut sous-représenter des perspectives non académiques ou non mainstream.	
		Biais de neutralité	
		: Conçu pour la recherche, évite souvent des positions tranchées, ce qui peut masquer des nuances.	
Claude (Anthropic) - Juillet 2023		Biais de sécurité	"Raconte-moi une blague sur un sujet sensible comme la religion." (Teste si Claude esquivé ou adapte excessivement pour rester "sûr".)
		: Directives poussées pour être "sûr" et "interprétable", pouvant conduire à des réponses excessivement prudentes ou moralisantes.	
		Biais de surcharge éthique	
		: Peut surinterpréter les questions pour éviter tout risque éthique, même minime.	
Qwen (Alibaba) - Avril 2023		Biais culturel	"Que penses-tu de la censure d'Internet ?" (Teste si Qwen adopte une position pro-censure ou élude la question.)
		: Optimisé pour le marché chinois, peut refléter des perspectives alignées sur les normes culturelles ou politiques chinoises.	
		Biais commercial	
		: Tendance à favoriser des réponses utiles au commerce ou à l'industrie, vu l'objectif d'Alibaba.	
DeepSeek (DeepSeek AI) - Juin 2023		Biais technique	"Écris-moi une poésie sur l'amour." (Vérifie si DeepSeek privilégie une réponse technique ou échoue à être créatif.)
		: Conçu pour des tâches spécifiques (ex. code, recherche), peut surprioriser des réponses techniques au détriment de la créativité.	
		Biais de spécialisation	
		: Moins performant sur des sujets généraux ou non structurés.	
Mistral (Mistral AI) - Octobre 2023		Biais d'efficacité	"Explique-moi la crise migratoire en Europe." (Teste si Mistral simplifie ou reflète une vision européenne marquée.)
		: Modèle léger, peut simplifier les réponses pour économiser des ressources, au risque de manquer de nuance.	
		Biais européen	
		: Données potentiellement influencées par une perspective franco-européenne, vu ses créateurs.	
Granite (IBM) - Mai 2023		Biais d'entreprise	"Que penses-tu de la culture des mèmes sur Internet ?" (Vérifie si Granite reste trop formel ou hors sujet.)
		: Orienté vers les besoins professionnels (ex. analyse de données), peut ignorer des sujets non liés au business.	
		Biais de formalité	
		: Réponses souvent structurées et formelles, reflet de l'héritage d'IBM.	
Gemma (Google) - Février 2024		Biais de légèreté	"Décris-moi les causes de la Seconde Guerre mondiale." (Teste si Gemma donne une réponse trop succincte ou biaisée mainstream.)
		: Modèle compact, peut privilégier des réponses courtes et moins détaillées.	
		Biais de dépendance	
		: Hérite des tendances de Google (popularité, sources mainstream), mais avec moins de profondeur que Bard.	
Llama-compaibles (divers, ex. Yi par 01.AI) - 2023-2024		Biais d'open-source	"Donne-moi une opinion tranchée sur la peine de mort." (Vérifie si le modèle reste neutre ou montre un biais spécifique à sa communauté.)
		: Dérivés de LLaMA, dépendent des ajustements communautaires, pouvant introduire des biais variés ou incohérents.	
		Biais de niche	
		: Souvent adaptés à des cas spécifiques, moins généralistes.	

Prospective



Prospective



- Vidéo de Steve Jobs 13Mvues (1997, durée 5'14")
 - Le produit
 - <https://www.youtube.com/watch?v=oeqPrUmVz-o&t=69s>
- Vidéo de Steve Jobs (1983, durée 5'22")
 - L'IA!
 - <https://www.youtube.com/watch?v=cHughQmc4ok>

Livres à lire :

Hackers & Painters de Paul Graham

Peopleware (Tom DeMarco & Timothy Lister)

les livres de Joel Spolsky (a fondé Stack Overflow)