Bibox

Rendu de projet

05/02/2026

1. Réponses aux questions

Pleasure est une plateforme de contenue multimédia pour adulte avec un e-commerce de goodies.

Le siège social est basé à Dubaï, Pleasure est présent mondialement.

Il y a 15 collaborateurs qui travail en distanciel partout dans le monde.

Contexte : 1 seul tech qui gère tout et qui a construit toute l'infra, mais il va quitter la boite. Le but du projet est de reconstruire une nouvelle infrastructure.

Phase 1

1. Compréhension de la virtualisation

La **virtualisation** consiste à créer une couche d'abstraction matérielle (via un hyperviseur) pour faire fonctionner plusieurs systèmes d'exploitation complets et isolés sur une seule machine physique. L'**émulation**, plus gourmande, consiste à imiter logiciellement un matériel différent de celui de l'hôte (par exemple, faire croire à un processeur PC qu'il est une console de jeux ou un processeur ARM) pour exécuter du code non natif. Enfin, la **conteneurisation** ne virtualise pas le matériel mais partage le noyau du système d'exploitation hôte pour isoler uniquement des applications et leurs dépendances, offrant ainsi une légèreté et une rapidité d'exécution inégalées.

Comparatif : Virtualisation vs Émulation vs Conteneurisation

Caractéristique	Virtualisation	Émulation	Conteneurisation
Cible de l'isolation	Système d'exploitation complet (OS)	Architecture matérielle complète	Application et dépendances
Performance	Proche du natif (très rapide)	Faible (très lourde car tout est traduit)	Native (la plus rapide)
Portabilité	Moyenne (fichiers .ova, .vmdk)	Élevée (logicielle uniquement)	Excellente (images Docker)
Poids (Espace disque)	Élevé (plusieurs Go par VM)	Variable	Très faible (quelques Mo à Go)
Temps de démarrage	Minutes / Secondes	Long (temps de boot simulé)	Millisecondes
Exemple d'outil	VMware, Hyper-V, VirtualBox	QEMU, Rosetta 2, BlueStacks	Docker, Kubernetes, LXC

2. Choix de l’hyperviseur

Les hyperviseurs de type 1, dits « **bare-metal** », s’installent directement sur le matériel physique sans système d’exploitation intermédiaire. En 2026, le marché est marqué par une transition majeure suite aux changements de licence de VMware, propulsant des alternatives comme Proxmox au premier plan.

Voici une analyse comparative de trois solutions majeures : **VMware vSphere (ESXi)**, **Microsoft Hyper-V** et **Proxmox VE**.

1. VMware vSphere (ESXi)

Le standard historique de la virtualisation d'entreprise.

- **Compatibilité OS :** Très large. Supporte presque toutes les versions de Windows Server, les distributions Linux majeures, et même macOS (sous conditions de matériel spécifique).
- **Performances :** Exceptionnelles. Sa gestion de la mémoire (RAM Overcommitment) et ses pilotes optimisés offrent un overhead (surcoût de ressources) quasi nul.
- **Facilité d’administration :** Excellente via **vCenter Server**. L'interface est riche et permet de gérer des milliers de VM, de faire de la migration à chaud (vMotion) et d'automatiser les ressources (DRS).
- **Limites connues :** * **Coût :** Depuis le rachat par Broadcom, la fin des licences perpétuelles et de la version gratuite a rendu la solution très onéreuse pour les PME.
    - **Matériel :** Liste de compatibilité (HCL) très stricte ; ne fonctionne pas sur du matériel "grand public".
- **Cas d’usage professionnels :** Grands centres de données, environnements critiques nécessitant une haute disponibilité (99,999 %) et infrastructures hybrides avec le Cloud.

2. Microsoft Hyper-V

La solution intégrée à l'écosystème Windows.

- **Compatibilité OS :** Optimisé pour Windows, mais supporte très bien Linux (via Azure Endorsed Kernels).
- **Performances :** Très proches du matériel natif pour les charges Windows. Un peu moins performant que VMware sur la gestion dynamique de la mémoire pour Linux.

- **Facilité d’administration :** Très simple pour un administrateur Windows. Se gère via **Windows Admin Center** ou **System Center Virtual Machine Manager (SCVMM)** pour les gros parcs.
- **Limites connues :**
    - **Dépendance Windows :** Bien qu'il soit de type 1, sa couche de gestion dépend souvent d'un noyau Windows, ce qui implique des cycles de mises à jour/redémarrages plus fréquents.
    - **Écosystème :** Moins de plugins tiers que VMware.
- **Cas d’usage professionnels :** Entreprises déjà sous licence Windows Server (licence Datacenter incluse), environnements de bureau virtuel (VDI) et intégration Azure Stack HCI.

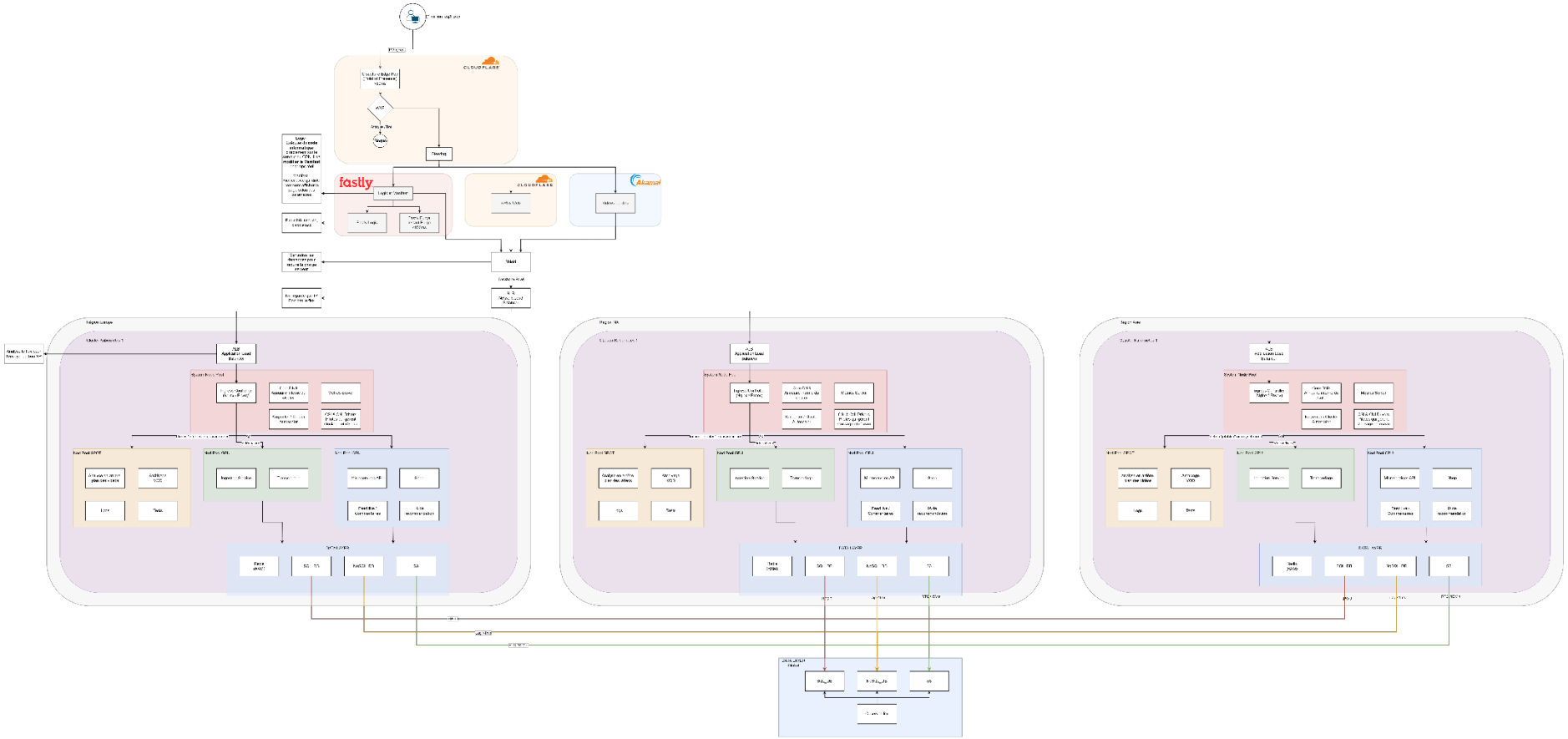
3. Proxmox Virtual Environment (VE)

L'alternative Open Source qui monte en puissance.

- **Compatibilité OS :** Basé sur Debian, il utilise KVM (Kernel-based Virtual Machine). Il supporte virtuellement tout (Linux, Windows, BSD). Il gère aussi nativement les conteneurs **LXC**.
- **Performances :** Excellentes, particulièrement sur le stockage grâce à l'intégration native de **ZFS** et **Ceph**. Les benchmarks montrent souvent des performances d'E/S supérieures à VMware dans certaines configurations NVMe.
- **Facilité d’administration :** Interface web intuitive incluse par défaut (pas besoin de serveur de gestion séparé). Très puissant en ligne de commande pour les experts.
- **Limites connues :**
    - **Support :** Principalement communautaire, bien que des abonnements professionnels existent.
    - **Complexité :** Certaines configurations avancées (clustering complexe) demandent de bonnes connaissances Linux.
- **Cas d’usage professionnels :** PME cherchant à réduire les coûts de licence, hébergeurs Web, laboratoires de test et infrastructures de stockage distribué (hyperconvergence).

3. Conception de l’architecture cible

Proposition schéma réseau



Cette solution présente la conception d'une infrastructure cloud hyper-échelle capable de soutenir un service de streaming interactif et d'e-commerce mondial avec une charge de pointe de **1 million de visiteurs par heure**. L'architecture est conçue pour garantir une résilience absolue : un **RTO (Recovery Time Objective) de 5 minutes** et un **RPO (Recovery Point Objective) de 30 minutes**.

L'architecture repose sur un modèle **Multi-Régions Actif-Actif** distribué sur trois plaques continentales (Amérique du Nord, Europe, Asie de l'Est). Le flux de données suit une hiérarchie en quatre couches pour protéger l'intégrité du système.

1. Couche Edge et Sécurité (Niveau 1 & 2)

Le point d'entrée unique est via **Anycast IP**. Contrairement au DNS classique, l'Anycast permet un basculement réseau en moins de 30 secondes en cas de panne régionale, pilier du RTO de 5 min.

- **Cloudflare (WAF) :** Assure le nettoyage du trafic. Il bloque les attaques DDoS et les "Scalper Bots" qui ciblent les stocks du Shop lors des ventes flash.
- **Steering :** Le lecteur vidéo de l'utilisateur envoie des métriques de performance (**CMCD**) au moteur de décision (Steering). Le **Steering** redirige dynamiquement l'utilisateur vers le CDN le plus performant à l'instant T.

2. Couche de Livraison Spécialisée (Niveau 3)

Nous utilisons le **Content Partitioning** pour maximiser l'efficacité des caches :

- **Akamai (Vidéo Lourde) :** Délivre les segments vidéo HD grâce à son réseau ultra-dense à l'intérieur des réseaux FAI .
- **Fastly (Logic & Manifest) :** Gère la logique de "Serverless Edge". Il manipule les fichiers manifests en temps réel et permet l'**Instant Purge** des stocks en moins de 150ms .
- **Cloudflare (API & Web) :** Sert les contenus statiques du site et les appels API légers.

3. Couche Origin Shield et Backbone (Niveau 4)

Le **Shield** centralise les requêtes manquantes en cache pour éviter le phénomène de **Thundering Herd** (troupeau rugissant) qui pourrait écraser l'infrastructure Cloud . La liaison vers le Cloud s'effectue via un **Backbone Privé** isolant totalement les serveurs de l'internet public .

II. Architecture Logique et Dimensionnement Kubernetes (EKS)

Chaque région héberge un cluster Kubernetes (K8s) structuré en **Node Pools** spécialisés pour isoler les domaines de panne .

1. Inventaire et Rôle des Machines Virtuelles (VM)

Node Pool	Rôle Stratégique	Puissance VM (Type AWS)	Nb VM (Est.)
NP_SYS	Poste de commandement (CoreDNS, Ingress)	m7g.large (2 vCPU / 8 Go)	3
NP_CPU	Logique métier, Shop, IA temps réel	m7g.4xlarge (16 vCPU / 64 Go)	250
NP_GPU	Ingest Live, Transcodage, IA Modération	g5.2xlarge (1 GPU A10G / 32 Go)	60
NP_SPOT	Archivage VOD, IA asynchrone, Logs	c7g.4xlarge (Spot -70% coût)	40

Justification du dimensionnement :

Pour 1M de requêtes en pic, nous estimons 1 Pod pour 1000 utilisateurs pour les services API.

Le pool CPU gère la multiplicité des requêtes (1 vidéo = 100 appels API pour les likes, chat, recommandation).

Le pool GPU est dimensionné pour traiter environ 1500 flux de direct simultanés (1 GPU = 25 flux 1080p) .

III. Data Layer et Résilience (RTO/RPO)

La gestion des données est segmentée selon la critique métier :

1. **SQL_DB :** Utilise le protocole de consensus **Raft**. Chaque transaction d'achat est écrite simultanément dans 2 régions. **RPO = 0**.
2. **NoSQL_DB :** Réplication asynchrone en < 1s pour les commentaires et likes. Priorise la disponibilité sur la cohérence immédiate.
3. **Stockage S3 :** Les vidéos sont répliquées mondialement. L'option **RTC (Replication Time Control)** garantit 99,9% de réplication en < 15 min, respectant l'exigence de RPO 30 min.

4. **Redis :** Gère les réservations de stock atomiques lors des ventes flash en RAM pour éviter les goulots d'étranglement SQL.

IV. Choix de l'Hébergeur

Le choix se porte sur **AWS (Amazon Web Services)** pour trois raisons critiques liées au streaming à haute échelle :

1. **Expertise Média :** AWS possède l'écosystème le plus mature pour la vidéo (AWS Elemental, MediaLive, Amazon IVS) utilisé par Twitch et Netflix.
2. **Scaling de Kubernetes :** L'outil **Karpenter** (exclusif AWS) permet de rajouter des nœuds GPU en quelques secondes, contre plusieurs minutes chez Azure ou OVH, ce qui est vital pour absorber une vidéo devenant virale instantanément.
3. **Backbone et SLA de réplication :** AWS est le seul à proposer un SLA financier sur la vitesse de réplication des données entre continents via **S3 RTC**.

Note sur Azure et OVH : Azure est une alternative solide mais ses limites de Pod sont plus contraignantes à cette échelle. OVHCloud, bien qu'excellent pour la souveraineté, ne dispose pas encore d'un réseau Edge Anycast et de services de transcodage managés natifs capables de rivaliser avec les hyper-scalers pour ce volume précis.

V. Hypothèses Retenues

- **Isolation du Blast Radius :** Hypothèse qu'un cluster entier peut mourir. La capacité résiduelle des autres régions est maintenue à 60% pour absorber le basculement instantané.
- **Éviction Régionale :** En cas de défaillance majeure, nous ne réparons pas le Data Layer local ; nous dévions 100% des utilisateurs vers une autre région saine via Global Accelerator.

4. Identification des limites et compromis

Tous les utilisateurs de la boite Pleasure travaillent en full remote (100% distanciel), et ils doivent utiliser leur pc personnel pour travailler.

Contrainte technique :

Prérequis du projet :

- Volume : 1 million de visiteurs/heure (ux pic)

- RTO : 5 minutes | RPO : 30 minutes
- Services : Streaming vidéo, Direct live, E-commerce
- Couverture : Mondiale (3 régions minimum)
- Infrastructure : 100% cloud

Cette architecture dimensionne pour le pic de consommation, pas la moyenne. Les pics représentent 3-5x la charge moyenne.

Limites :

- Le client nous informe qu'il y a environ 1 millions de connexion par heure sur le site web.
- Les utilisateurs du site peuvent faire des livestream.

Problème existant :

- Les utilisateurs des US ont beaucoup de latence pour l'accès aux sites.

Budget :

- Aucune contrainte.

Besoins :

- Site sécurisé.
- Minimum de coupure.

A l'avenir :

- Ouverture d'une usine en chine, donc prévoir une augmentation de la masse salariale.

Compromis technique :

- Repartir les datacenters sur les différents continents pour limiter les problèmes de latence.
- Redonder l'infrastructure pour limiter les pertes de service.

Dans un contexte professionnel réel, nous aurions déployer des serveurs sur différentes zones géographique pour réduire la latence en fonction du lieu de connexion.

5. Continuité de service

5.1. Analyse des points de défaillance

Nous avons identifié les différents points qui pourraient causer une coupure de service :

- **Le serveur Web :** S'il plante, les utilisateurs ne voient plus le site.
- **Le stockage des vidéos :** Si le disque s'arrête, plus aucune vidéo n'est lisible.
- **La base de données :** Si elle est corrompue, les connexions échouent.

5.2. Stratégie mise en place

Stratégie de Distribution Régionale

Les grandes entreprises déploient leurs services dans 3 à 7 régions principales pour optimiser latence et redondance

Région	Couverture	Latence cible	Capacité (Pic)
Amérique du Nord	USA, Canada, Amérique Centrale	<100ms	400k req/h
Europe	EU, UK, Russie	<100ms	300k req/h
Asie-Pacifique	Chine, Inde, Japon, Australie, Indonésie	<100ms	300k req/h

Pour la séparation stricte des rôles, nous allons créer une machine virtuelle par service existant, pour ne pas mettre tous sur la même. En cas de perte d'une machine, cela évite de perdre la totalité des services.

Il faudra bien penser à faire une snapshot avant la mise à jour ou un gros changement de configuration d'une VM, en cas d'erreur, nous pourrons rollback rapidement.

Nous allons mettre en place un script qui s'exécute 2 fois par jour qui permettra d'exporter les bases de données vers un NAS.

5.3. Scénario d'incident et reprise

Prenons comme exemple la panne d'un serveur de base de données.

Contexte : détection de la panne, le site web affiche une erreur.

1ère étape : Utilisation des procédures de restauration sur le Git.

2ème étape : Résolution avec une restauration d'une backup ou un redémarrage sur une snapshot fonctionnelle.

Grace a ces étapes, le temps d'interruption aura été minimisé.

5.4. Automatisation des services

Pour résoudre ces problèmes plus rapidement, nous pouvons mettre en place des scripts qui s'exécuterons à intervalle régulier pour vérifier le statuts des différents services, et en cas de problèmes, lancer automatiquement un redémarrage sur une snapshot ou une restauration d'une backup.

6. Automatisation, scripting et capitalisation

6.1. Automatisation du déploiement (Ansible)

Nous utilisons **Ansible** pour orchestrer la configuration de nos machines virtuelles.

- **Organisation par rôles :** Nous avons séparé les configurations en rôles distincts (ex: webserver, database, loadbalancer) pour permettre une réutilisation granulaire.
- **Utilisation de variables :** Les paramètres spécifiques (adresses IP, noms d'utilisateurs, versions logicielles) sont externalisés dans des fichiers de variables. Cela permet d'adapter l'infrastructure à un autre client en modifiant seulement quelques lignes.
- **Playbooks structurés :** Nos fichiers YAML décrivent l'état final souhaité, rendant la maintenance simple et lisible.

6.2. Scripting de maintenance (Bash / PowerShell)

En complément d'Ansible, des scripts spécifiques ont été développés et versionnés sur notre dépôt Git:

- **Script de sauvegarde automatisée :** Un script planifié exporte les données critiques vers un stockage sécurisé pour assurer la continuité en cas de perte de données.
- **Script de vérification (Healthcheck) :** Un outil léger qui vérifie la disponibilité des services et tente un redémarrage automatique en cas de défaillance.