

Voilà une architecture “coût minimal mais scalable” pour StrategyBuzzer qui couvre app mobile, millions de joueurs, monétisation, chat texte & vocal, rapidité et synchro inter-continents.

# Architecture cible (rentable → scalable)

## 1. Fronts

- **Web:** Laravel Blade (existant) + composants JS légers (HTMX/Alpine) pour les écrans non temps réel.
- **Mobile: Flutter** (1 codebase iOS/Android) qui consomme l’API Laravel et le temps réel.
- **Admin/Stats:** mini SPA (React/Vite) branchée sur l’API + flux temps réel.

## 2. Backends (micro-modules simples)

- **API & Auth:** **Laravel** (Sanctum) + **Octane** (Swoole ou RoadRunner) pour gagner ~2-3x de perfs.
- **Temps réel:** **Laravel WebSockets** auto-hébergé (bcp moins cher que Pusher) + **Redis** (pub/sub, jobs).
- **Matchmaking & Rooms:** service Node.js (petit serveur ws dédié) ou package Laravel WebSockets “rooms” si on veut rester 100% PHP.
- **Chat texte:** canaux WebSocket (salons/rooms) persistés en DB (court) + export froid en stockage objet.
- **Chat vocal:** **WebRTC** via **coTURN** (serveur TURN/STUN auto-hébergé), appels P2P quand possible, fallback relai TURN.
- **Paielements:** **Stripe** (Coins/achats in-app, Apple Pay/Google Pay) + webhooks.
- **Média & CDN:** assets dans **Cloud Storage** + **Cloud CDN**.
- **Analytique:** events dans **BigQuery** (batch) + **OpenTelemetry** pour traces.

## 3. Données

- **OLTP** (comptes, inventaire, parties): **MySQL/MariaDB** (réplique en lecture).
- **Temps réel global & état de salle:** **Redis** (volatile) + “snapshots” MySQL.
- **Historique à long terme** (scores, logs): stockage objet (parquet) + **BigQuery**.
- (Option si tu veux rester dans ta logique actuelle) **Firestore multi-région** pour présence et feed léger; tu l’utilises déjà pour certaines features, on peut le garder comme “bus”

simple pour certains écrans.

#### 4. **Infra (GCP, low-cost d'abord)**

- **Nginx reverse proxy + Let's Encrypt.**
- **1 VM e2-standard-2** (Laravel API + Octane + queue) au début.
- **1 VM e2-micro** (Redis seul).
- **1 VM e2-micro** (Laravel WebSockets).
- **1 VM e2-micro** (coTURN).
- **Cloud SQL**: start petit (Shared/Core), passer à HA quand la charge le justifie.
- **Cloud Load Balancer + Cloud CDN** devant Nginx quand tu dépasses ~5-10k CCU.
- **Autoscaling**: mig gérée pour API & ws quand nécessaire.

#### 5. **Scalabilité & latence inter-continents**

- **Régions**: primaire en **Montréal (N. America)**, secondaire en **Belgique/Finland** pour l'Europe.
- **Affinity par salle**: un room est "hébergé" sur une seule région; matchmaking place les joueurs proches.
- **Réplication**: MySQL lecture répliquée, jobs cross-region via **Pub/Sub**.
- **Clock**: horloge serveur unique (NTP) et latence compensée côté client (affichage UX).

#### 6. **Sécurité & coûts**

- **Octane** réduit CPU → moins de VM.
- **WebSockets self-hosted** évite la facture PaaS.
- **coTURN** open-source (10–20\$/mo de VM) vs API vocale managée coûteuse.
- **CDN** pour tout ce qui est statique (cache 95%).
- **Queues** (Redis) pour lisser les pics.

#### 7. **Monétisation**

- **Stripe** pour coins/achats.

- **IAP** Flutter (StoreKit/Billing) pour coins côté mobile (avec **server-side verify** dans Laravel).
  - **Webhooks**: idempotence + enregistrement des transactions (journal).
- 

## Flux temps réel (résumé)

- Le client rejoint un **lobby** → matchmaking → assigne **room:region**.
  - Le client ouvre **WS** vers le **gateway régional**.
  - Les événements de jeu (buzzer, réponses, chrono) passent WS → **Redis pub/sub** → diffusés aux joueurs du room.
  - L'API persiste checkpoints/score finaux en **MySQL**.
  - Chat texte = même WS (channels séparés).
  - Chat vocal = **WebRTC**; STUN/TURN = coTURN. P2P si possible, relay sinon.
- 

## Plan mobile (Flutter)

- UI native fluide, animations, notifications, IAP.
  - Connexion à **/api/** (Sanctum token) + **ws://** pour en temps réel.
  - Modules: Auth, Lobby, Matchmaking, Rooms, Chat, Boutique (Stripe+IAP), Profil, Avatars.
- 

## Déploiement GCP — commandes (une par une)

Contexte: connecté en SSH sur ta VM (Ubuntu), domaine déjà pointé.

### 1) Mises à jour

- `sudo apt-get update`
- `sudo apt-get upgrade -y`

## 2) Nginx + certbot

- `sudo apt-get install -y nginx`
- `sudo ufw allow 'Nginx Full'`
- `sudo apt-get install -y certbot python3-certbot-nginx`

## 3) PHP & extensions (ex: PHP 8.3)

- `sudo add-apt-repository ppa:ondrej/php -y`
- `sudo apt-get update`
- `sudo apt-get install -y php8.3 php8.3-fpm php8.3-mysql  
php8.3-redis php8.3-xml php8.3-mbstring php8.3-curl php8.3-zip`

## 4) Composer

- `php -r "copy('https://getcomposer.org/installer',  
'composer-setup.php');"`
- `php composer-setup.php`
- `sudo mv composer.phar /usr/local/bin/composer`

## 5) Node (pour build front & outils)

- `curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -`
- `sudo apt-get install -y nodejs`

## 6) Cloner le projet

- `cd /var/www`
- `sudo git clone https://github.com/toncompte/strategybuzzer.git`
- `sudo chown -R www-data:www-data strategybuzzer`
- `cd strategybuzzer`

## 7) Dépendances Laravel

- `composer install --no-dev --optimize-autoloader`
- `cp .env.example .env`
- `php artisan key:generate`

## 8) Octane (Swoole)

- `composer require laravel/octane`
- `php artisan octane:install`
- `sudo apt-get install -y php8.3-swoole`

## 9) Redis (VM dédiée conseillée, sinon même VM pour débiter)

- `sudo apt-get install -y redis-server`
- `sudo sed -i 's/^supervised no/supervised systemd/'  
/etc/redis/redis.conf`
- `sudo systemctl enable redis-server`
- `sudo systemctl start redis-server`

## 10) Laravel WebSockets

- `composer require beyondcode/laravel-websockets`
- `php artisan vendor:publish  
--provider="BeyondCode\LaravelWebSockets\WebSocketsServiceProvide  
r"`
- `php artisan migrate`

## 11) coTURN (chat vocal)

- `sudo apt-get install -y coturn`

- `sudo sed -i 's/^#TURNSEVER_ENABLED=.* /TURNSEVER_ENABLED=1 /' /etc/default/coturn`
- `sudo bash -c 'echo "listening-port=3478" >> /etc/turnserver.conf'`
- `sudo bash -c 'echo "fingerprint" >> /etc/turnserver.conf'`
- `sudo bash -c 'echo "lt-cred-mech" >> /etc/turnserver.conf'`
- `sudo bash -c 'echo "realm=strategybuzzer.com" >> /etc/turnserver.conf'`
- `sudo bash -c 'echo "user=sbturn:MotDePasseSolideIci" >> /etc/turnserver.conf'`
- `sudo systemctl enable coturn`
- `sudo systemctl restart coturn`

## 12) Nginx vhost (Laravel + WS)

- `sudo nano /etc/nginx/sites-available/strategybuzzer`
  - (je te donnerai le vhost complet dès que tu m'envoies un screenshot de ton Nginx actuel pour aligner chemins et sockets)
- `sudo ln -s /etc/nginx/sites-available/strategybuzzer /etc/nginx/sites-enabled/`
- `sudo nginx -t`
- `sudo systemctl reload nginx`

## 13) Certificat TLS

- `sudo certbot --nginx -d strategybuzzer.com -d www.strategybuzzer.com`

## 14) Opti Laravel

- `php artisan config:cache`

- `php artisan route:cache`
- `php artisan view:cache`

#### 15) Démarrage Octane (service)

- `php artisan octane:start --server=swoole --watch` (dev)
- (prod: je te fournis un unit file systemd propre quand on a validé les chemins exacts)

#### 16) Démarrage WebSockets (service)

- `php artisan websockets:serve`

#### 17) Files queues (si nécessaire)

- `php artisan queue:work --tries=3`

#### 18) Stripe webhook (local → prod)

- `php artisan make:controller StripeWebhookController`
- (on activera l'URL dans Stripe Dashboard après capture d'écran)

---

## Paramétrage côté app (résumé)

- **.env:** REDIS, BROADCAST\_DRIVER=pusher (pointant sur ton WebSockets), OCTANE\_SERVER=swoole, URL TURN.
- **Broadcasting.php:** adapter Pusher pour Laravel WebSockets (host interne, port 6001, wss via Nginx).
- **Flutter:** endpoints `/api/*`, ws `wss://strategybuzzer.com/app/....`

---

## Observabilité & anti-triche

- **Rate-limit** par IP/compte sur endpoints sensibles (buzzer).

- **Server authority**: le serveur confirme l'ordre des buzzers (timestamp serveur).
  - **Metrics**: Prometheus + Grafana (faible coût) ou Cloud Monitoring.
  - **Logs**: Loki/Cloud Logging, rétention courte, archivage objet.
- 

## Ce que j'ai besoin maintenant (captures d'écran)

Pour te livrer les fichiers **Nginx vhost**, **services systemd** d'Octane/WebSockets prêts à coller:

1. Page **Nginx**: contenu actuel de `/etc/nginx/sites-available/default` (ou `strategybuzzer` si déjà créé).
2. **Arborescence** réelle du projet sur la VM (`/var/www/...`).
3. Sortie de `php -v` et `php -m`.
4. Si tu utilises déjà Cloud SQL: **host**, **port**, **dbname** (sans poster les mots de passe ici).

Dès que j'ai ces captures, je te renvoie:

- Le **vhost Nginx complet** (copier/coller).
- Les **unit files systemd** pour Octane & WebSockets.
- Le **.env** (variables clés, prêtes à coller).
- Les **endpoints** Stripe (webhooks) et routes temps réel.

Tu auras ainsi une base **peu coûteuse** au repos, qui **scale** linéairement avec l'usage, compatible **mobile**, **chat texte & vocal**, et **multi-continents** avec **latence minimale**.

**Vous avez dit :**

Tu estimes les coûts à combien par mois

**ChatGPT a dit :**

12 octobre 2025, 10:51 (Québec)

Excellente question Steve — voici une **estimation détaillée des coûts mensuels** pour faire tourner **StrategyBuzzer** sur Google Cloud Platform (GCP) avec tout ce que tu veux :

⚙️ Back-end complet Laravel + Flutter mobile, chat texte/vocal, monétisation Stripe, synchronisation multi-régions, et possibilité d'échelle jusqu'à des millions de joueurs.

---



## 1 Phase DÉMARRAGE / DÉVELOPPEMENT

(100 à 2 000 joueurs actifs/jour — minimal mais fluide)

Élément	Détails	Coût mensuel estimé
VM principale (Laravel + Octane)	e2-standard-2 (2 vCPU, 8 Go RAM), 30 Go SSD	\$30–35 USD
VM Redis + Queue	e2-micro	\$6 USD
VM WebSockets	e2-micro	\$6 USD
VM coTURN (chat vocal)	e2-micro	\$6 USD
Cloud SQL (MySQL)	instance partagée f1-micro (ou db-shared-core)	\$15 USD
Stockage Cloud Storage + CDN	images/avatars, musiques, caches CDN	\$10 USD
Trafic sortant / CDN Europe-Canada	200 Go/mo	\$10–15 USD
Nom de domaine + DNS (Namecheap)	déjà payé (≈ \$12/an)	—
Sauvegardes & logs	Cloud Storage coldline (backup DB + logs)	\$3–5 USD
Stripe / paiements	sans frais fixes (commissions 2.9 % + 0.30 \$/txn)	variable

 **Total phase démarrage : ≈ \$85–100 USD/mois**

 suffisant pour ~2 000 joueurs actifs/jour, chat inclus, sans latence notable.

---

## 2 Phase CROISSANCE / PRÉ-LANCEMENT

(10 000 à 100 000 joueurs actifs/jour, 3 régions)

Élément	Ajustement	Coût mensuel estimé
Load Balancer + CDN multi-région	Amérique + Europe	\$20–25 USD
VM API / Octane x 2 (autoscale)	e2-standard-4	\$60–80 USD
VM WebSockets x 2 (autoscale)	e2-medium	\$25 USD

Redis (plus de connexions)	e2-small	\$15 USD
MySQL Cloud SQL HA (2 zones)	db-n1-standard-2	\$60 USD
coTURN (2 VM régionales)	e2-micro x 2	\$12 USD
Trafic CDN	~1 To/mo	\$40 USD

💰 **Total phase croissance : ≈ \$220–260 USD/mois**

- ➡ Supporte **jusqu'à 100 000 joueurs/jour** avec chat texte + vocal en simultané.
- ➡ Ajout automatique de VM via autoscaling GCP selon charge.

## 3 Phase INTERNATIONALE / ÉCHELLE

(500 000 + joueurs/jour, Amérique + Europe + Asie)

Élément	Détails	Coût
API / Octane + WebSockets autoscaling	Kubernetes ou MIG GCP	\$800–1 200 USD
Redis clusterisé (MemoryStore)	haute dispo	\$100 USD
MySQL HA multi-région	réplicas de lecture	\$150 USD
Cloud CDN 5 To/mo	jeux, sons, avatars	\$100 USD
TURN multi-régions (3-4)	pour vocal stable	\$40 USD
Logs, backups, BigQuery	analytics global	\$40 USD

💰 **Total phase mondiale : ≈ \$1 400–1 700 USD/mois**

- ➡ Capacité **1 million+ joueurs/jour** sans latence.
- ➡ Toujours < \$2 k/mo pour un jeu planétaire.



## Comparatif AWS / Firebase

Plateforme	Avantage	Inconvénient	Coût moyen équivalent
Firestore + Cloud Functions	déploiement rapide, auto-scale	très cher au-delà de 100 000 req/j	\$600–1 000/mo
AWS (ECS + RDS)	robuste, support mondial	complexité, facturation confuse	+20 % vs GCP

<b>GCP (notre choix)</b>	simple, francophone, scalable linéaire	configuration manuelle initiale	<b>85 → 1 500 USD selon usage</b>
--------------------------	---	------------------------------------	---------------------------------------