Master 1 GIL - Document technique - Outils annexes

Groupe 1 Agora V3-1 : Partie bibliothèque de jeux

19 mai 2024

Version	1
Date	19 mai 2024
Rédigé par	BINGINOT Etienne

Mises à jour du document

Version	Date	Modification réalisée
1	19 mai 2024	Création du document

Table des matières

1 Frankenphp

Frankenphp est une application moderne de serveur PHP, basée sur un serveur Caddy (un serveur web écrit en Go). Son intérêt pour ce projet est de permettre l'utilisation d'un serveur web Go, qui est un des serveurs webs les plus performants à l'heure actuelle, avec un serveur PHP optimisé et une intégration complète.

Frankenphp possède également un système de worker, permettant de charger entièrement en mémoire l'application dans un ou plusieurs workers, ceux-ci pouvant ensuite répondre aux requêtes beaucoup plus rapidement. Ce système a actuellement été désactivé en environnement de développement, ceux-ci ne permettant pas un rafraîchissement automatique de leur contenu (le chargement en mémoire se fait une seule fois).

De plus, l'intégration à Symfony se fait naturellement et simplement, l'auteur du projet Frankenphp étant un développeur de Symfony. Un serveur Mercure est également facilement déployable directement en parallèle du serveur Caddy et PHP, permettant une communication quasi-instantanée entre ces 3 entités et des performances accrues.

1.1 Mercure

Mercure est un protocole de communication permettant à une application web d'envoyer des données HTTP à des navigateurs de façon efficace et rapide. Son intérêt pour ce projet est de pouvoir facilement mettre à jour les affichages des différents joueurs quand l'état du jeu a changé, sans avoir besoin de recharger la page. Ainsi, l'affichage de chaque joueur reste cohérent peu importe le déroulement de la partie, que la page soit chargée ou non.

Pour fonctionner, Mercure utilise un système d'URL :
- MERCURE_URL : utilisé par l'application pour envoyer les données aux navigateurs
- MERCURE_PUBLIC_URL : utilisé par les navigateurs pour écouter les envois de données via Mercure

Ensuite, chaque navigateur peut écouter via l'URL publique les envois de données, en utilisant un système de topics. Ainsi, chaque navigateur écoute, via du code javascript, les topics de son choix et sera mis au courant quand une mise à jour du topic sera publiée.

2 Traefik

Traefik est le reverse-proxy de l'application. Son rôle est de servir d'entrée vers l'application, en filtrant les différentes requêtes et en utilisant un système de cache. Son rôle est de permettre d'utiliser l'HTTPS pour l'ensemble de l'application, tout en utilisant le protocole HTTP à l'intérieur de l'application (sécurisé vers l'extérieur mais rapide à l'intérieur). De plus, le système de cache permet de répondre rapidement aux requêtes simples.

Cet outil a premièrement été mis en place dans un objectif de mise en place du protocole HTTPS. En effet, du fait du protocole Mercure, la communication interne entre les différents composants devait utiliser le protocole HTTP, mais devait utiliser HTTPS avec l'extérieur. Ce reverse-proxy a donc été la solution à ce soucis.

Un dashboard de contrôle est accessible au port 8080 en HTTP

3 Prometheus

La mise en place de Traefik permet également la mise en place d'outils annexes de récolte de données, comme Prometheus. Son rôle est de récolter les différentes métriques (temps de réponse, code de retour HTTP, nombre de requêtes, etc) via les données de Traefik pour analyser l'état du serveur, et vérifier que celui-ci fonctionne bien.

Il s'utilise en parallèle avec Grafana, qui est l'interface graphique de Prometheus. Il est ainsi possible de voir des graphiques, des courbes, et des statistiques sur le serveur, permettant d'assurer sa maintenance au long terme.

L'application graphique est accessible au port 3000 en HTTP, avec le nom d'utilisateur admin et le mot de passe admin.

4 Jaeger

Traefik permet également de rajouter l'outil Jaeger, pour analyser les différentes routes appelées. Le but ici est d'avoir un visuel sur quelles sont les routes les plus demandées, les plus longues à répondre, celle avec le plus de code d'erreur, etc. Il s'agit donc d'une vue route par route, contrairement à Prometheus qui se concentre sur l'application globale.

L'application est accessible au port 16686 en HTTP

5 Stack ELK

La stack ELK a pour but de récolter les logs de l'application et permettre une analyse rapide de ceux-ci via elastic search. Ainsi, les logs sont centralisés et via l'utilisation de grammaire définissant la syntaxe des logs, des recherches poussées peuvent être réalisées, pour par exemple rechercher l'ensemble des erreurs critiques.

L'application est divisée en 3 éléments :
— Logstash : récolte les logs des différents éléments
— Elastic Search : réalise une recherche élastique dans les logs, et les indexe
— Kibana : interface graphique de la stack ELK

De plus, un outil supplémentaire a été rajouté : Metricbeats. Son but est de suivre l'état de vie des différents conteneurs Docker (CPU, RAM, etc) afin de vérifier que ceux-ci ne sont pas en surcharge. Il est également possible de le configurer pour qu'il analyse également la machine hôte (hébergeant l'application Docker).

L'application graphique est accessible au port 5601 en HTTP