Introduction

Choix technologiques

- Ce laboratoire à été codé en c#, puisque la majorité des équipes utilise ce language depuis au moins deux sessions au moment de l'écriture de ce text.
- Le language n'est pas imposé et vous pouvez créer ou réécrire vos services dans le language de votre choix (à vos risques et périles).
- Le Framework utilisé pour les microservices est ASP.NET Core.
- Bien que le système n'utilise pas Docker Swarm ou Kubernetes, certaines des fonctionnalités ont été reproduites. Nous avions besoin d'une solution facile à déployer avec un minimum de configuration. De plus ce système permet l'ajout de "Chaos Engineering" et d'évaluations spécifique au cours de Log430 en un seul composant déployable sur votre Docker Desktop.
- Docker Desktop: Vous devez utiliser Docker Desktop puisqu'il permet une gestion de conteneur local et devez activer dans les "Settings" => "General" => "Expose daemon on tcp://localhost:2375 without TLS". Votre ordinateur doit permettre la virtualisation.
- Un ServiceMeshHelper Nuget package à été ajouté à vos service pour facilité l'accès au service mesh (et au service discovery). Cependant, il est possible pour vous de faire les mêmes appels par vous même par l'Api du node controller
- MassTransit, la librairie à été choisit pour les intéraction avec RabbitMQ puisqu'elle englobe des attributs de qualité intéressante pour nos besoin. MassTransit
- Les services (incluant le nodecontroller), utilisent swagger (OpenAPI) pour la documentation. Quand vos services sont déployé, vous avez accès à cette adresse: @http://localhost:[Port Number]/swagger/index.html
- Le code qui vous est donné comporte des erreurs, certaines intentionnels pour vous permettre de prendre de meilleurs décisions architectural, d'autre moins intentionnel... vous êtes responsable des répercutions de ces erreurs sur votre système
- Le nodecontroller est "Pull" de DockerHub, si vous pensez qu'il contient une erreur, notifié votre chargé de laboratoire en fournissant le context exact. Si le problème peut être reproduit, il sera corrigé et une nouvelle version sera publiée. Dans la deuxième itération du laboratoire, votre version du nodecontroller sera affiché sur l'interface utilisateur lors de l'évaluation. il sera donc facile de repéré que vous n'avez pas la dernière version (simplement rédeployé avec la dernière version dans ce cas).

Pointeurs généraux

- Par contrainte de temps, tout les scénarios de configuration n'ont pas été testé et plusieurs erreur n'ont pas de messages d'erreur intuitif. En cas d'erreur de configuation, revalider votre docker compose avec les détails définits dans le fichier "Infrastructure (déploiement)"
- Vous êtes encourgés à expérimenter!
- N'ayez pas peur de redéployer souvent.
- Vous pouvez modifier le nombre de Nano CPUs (1000000000 équivaut à un coeur) alloué a chaque conteneurs sur l'api du nodecontroller si une expérience n'est pas en cours.
- Avant une évaluation, je vous conseil de relancer votre Docker Daemon (Dockerd) afin de nettoyer tout memory leaks et autres.
- Vos Dockerfiles sont en mode Debug pour simplifié votre debugging, il pourrait être intéressant de les mettres en Release pour les évaluation 2, 3 et 4.
- Les tests ont été déprecated dû à un manque de maintenance et de temps sur les services que vous utilisés. Plutot que de vous remettre une couverture et qualité de test douteuse, ils ont été enlevé. Le projet de test demeure dans la solution, sentez vous libre d'écrire des tests au début afin de vous assurer que vos changements n'impacte pas la logique. Les Moqs seront essentiel pour vos tests, si vous en faites.
- Durant l'itération 1, vous pouvez testé votre système a partir de l'api du TripComparator

Suite

1- Prenez connaissance du fichier "Architecture Composants"

2- Prenez connaissance du fichier "Architecture Modules Type"

3- Prenez connaissance du fichier "Infrastructure (déploiement)"