Projet : IaaS collaboratif avec Docker

Cahier des charges

Equipe :

Etudiants RICM5

- **EUDES Robin**
- **DAMOTTE Alan**
- **BARTHELEMY Romain**
- **MAMMAR Malek**
- **GUO Kai**

Etudiants DUT

- **BONNARD Loïc**
- **CAPERAN Théo**

Table des matières :

I. Contexte et définition du problème

Présentation du projet sur le wiki air-imag[1] :

Docker est un logiciel libre qui automatise le déploiement d'applications Linux dans des conteneurs logiciels. Il offre une solution légère pour la virtualisation de machines d'exécution Linux, en comparaison des machines virtuelles systèmes comme Xen, VMVare, Hyper-V.

Le Big Data Analytics est un des principaux moteurs de la croissance des grandes sociétés informatiques. Les technologies Big Data comme Apache Hadoop, Apache Spark révolutionne la "Business Intelligence" en permettant de synthétiser en temps réel des énormes quantités d'informations en utilisant des milliers de machines virtuelles Xen, VMVare, Hyper-V des providers Cloud comme AWS, Azure, Google, …

L'objectif de ce projet est de permettre à un groupe d'utilisateurs (membres) de mettre en commun les machines de bureau et portables pour effectuer les calculs big data des nombreux utilisateurs. Nous visons aussi à offrir une solution de montée en charge souple et à bas coût, auquel chacun pourrait contribuer. Pour cela, la solution doit reposer sur Docker afin de virtualiser les machines des utilisateurs et contrôler l'usage des ressources de chaque machine. Une seconde phase du projet serait alors de ne plus “simplement” proposer une infrastructure, mais également un service assurant la résistance aux pannes par divers moyens.

Différents scénarios d’utilisation sont à étudier (en terme d’implication sur le développement, l’architecture, les choix de conception) :

- Chacun peut proposer de mettre sa machine à disposition
- Même si ma machine est utilisée par quelqu'un d'autre, je dois pouvoir l'éteindre, tant pis pour ceux qui s'en servent.
- La gestion de la configuration réseau
- L’utilisation d’instances ponctuelles
- Tenue à la montée en charge d’un service client

[1] http://air.imag.fr/index.php/IaaS_collaboratif_avec_Docker

II. Objectifs

- Montée en compétence sur les technologies utilisées
- Proposer une alternative open-source viable

III. Périmètre

On fixe le périmètre de notre projet à la réalisation des scénarios évoqués en introduction, et les moyens nécessaires à mettre en oeuvre pour cela. Chaque sous-point représente une contrainte ou une dépendance requise pour réaliser le scénario. Les contraintes sur l'ensemble des scénarios doivent être compatibles entre elles (ou un compromis réalisable à défaut de l'optimum).

"Chacun peut proposer de mettre sa machine à disposition"

- Avoir docker installé sur la machine du collaborateur
- Enregistrement auprès d'un frontend/annuaire

"Gestion de la configuration réseau"

- Gestion de répartition des noeuds de calcul chez des collaborateurs différents
- Rendre accessibles via SSH les instances dockers
- Rendre accessibles les services sur les instances depuis un réseau extérieur

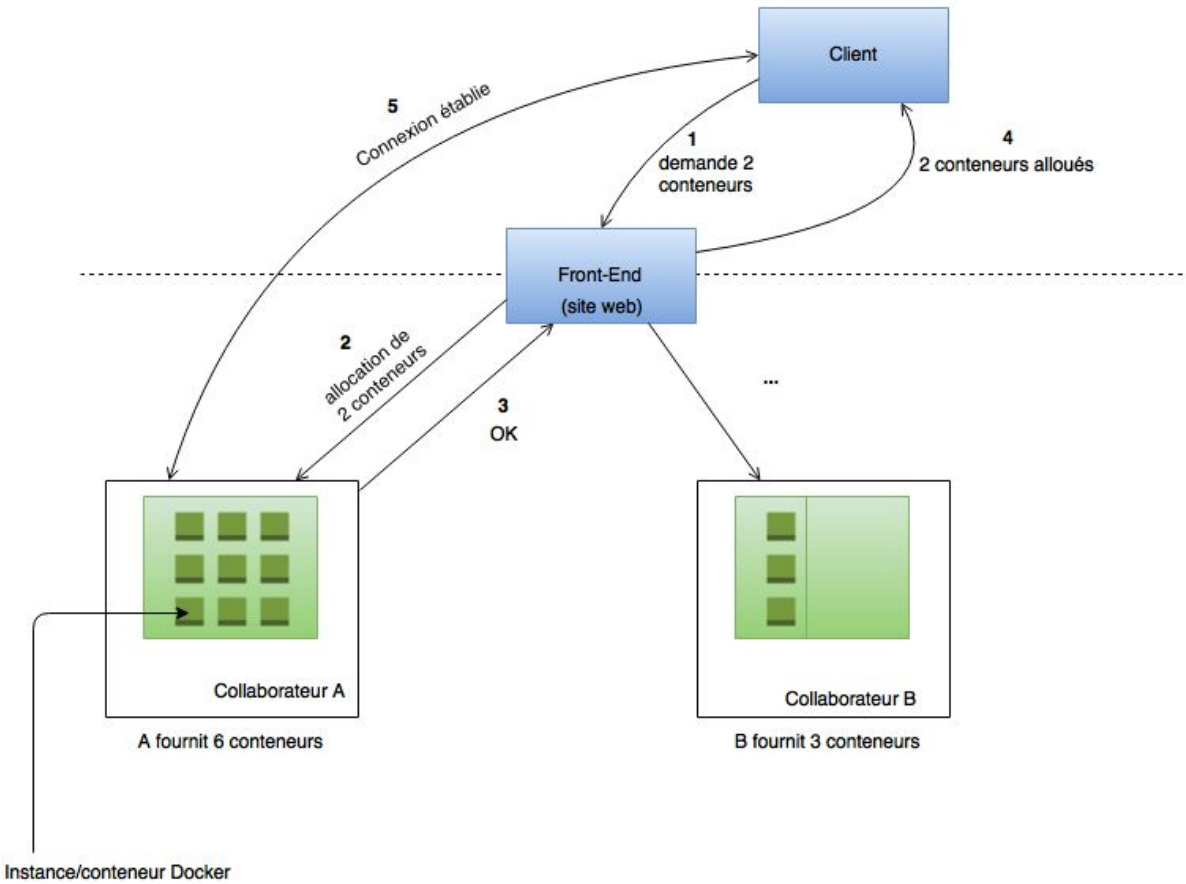
"*Un collaborateur éteint sa machine (perte des instances en cours)/Instances ponctuelles*"

- Redistribution des calculs
- Redondance/ réplication

IV. Description fonctionnelle

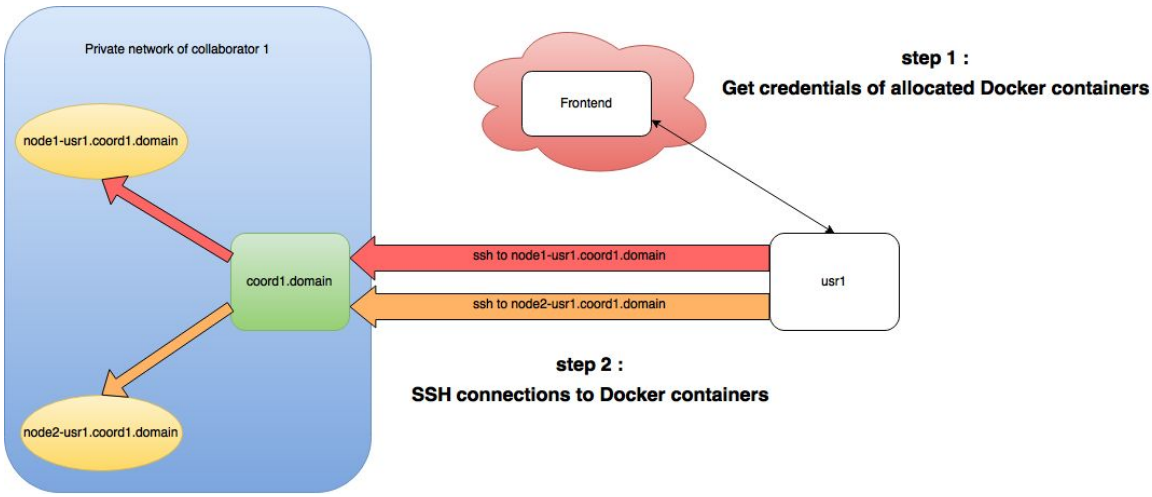
A/ Allocation d'une instance

Objectif : Le client a besoin de faire tourner son application sur 2 conteneurs



Au niveau de l'étape 4, les informations de connexions sont récupérées par l'utilisateur pour ensuite se connecter (étape 5) aux instances. Ces étapes sont détaillées dans le paragraphe suivant.

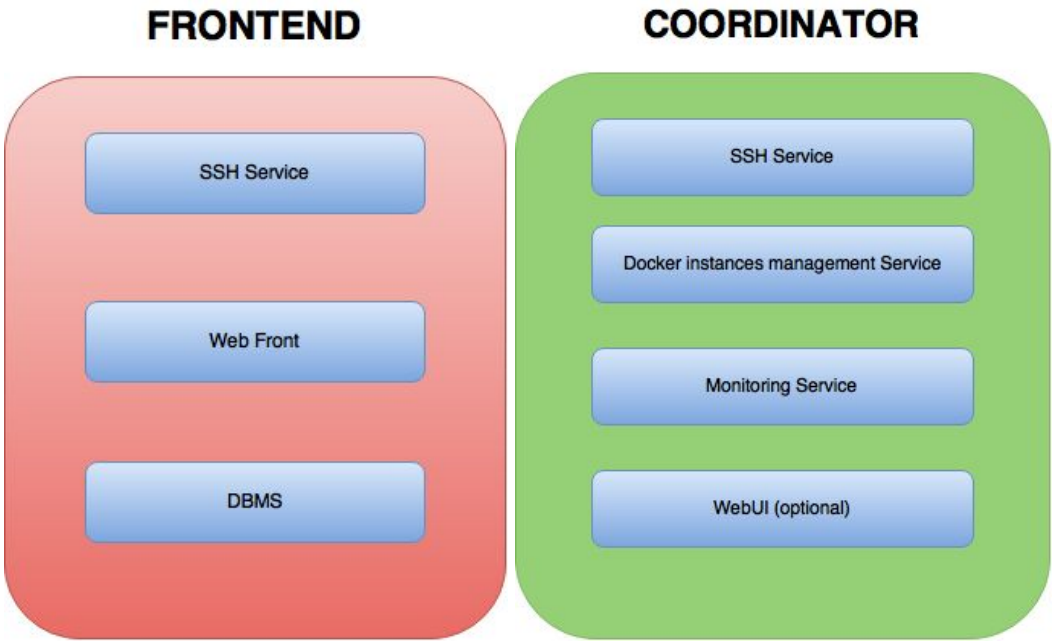
B/ Connexions SSH vers les instances



Caption :

Afin de permettre à de multiples instances d'être accessibles par SSH depuis un réseau externe, une technique envisagée est celle du "SSH Jump Host". Le flux SSH est redirigé par la box du collaborateur vers le noeud "coordinateur". Chaque connexion vers une instance Docker peut se décomposer en deux étapes : une première connexion vers le coordinateur, qui initie une connexion vers une machine locale du réseau auquel nous n'avons pas accès : l'instance docker. Le coordinateur a ici un simple rôle de "Proxy SSH". Le collaborateur devra juste ouvrir son port SSH, et faire du NAT vers le coordinateur.

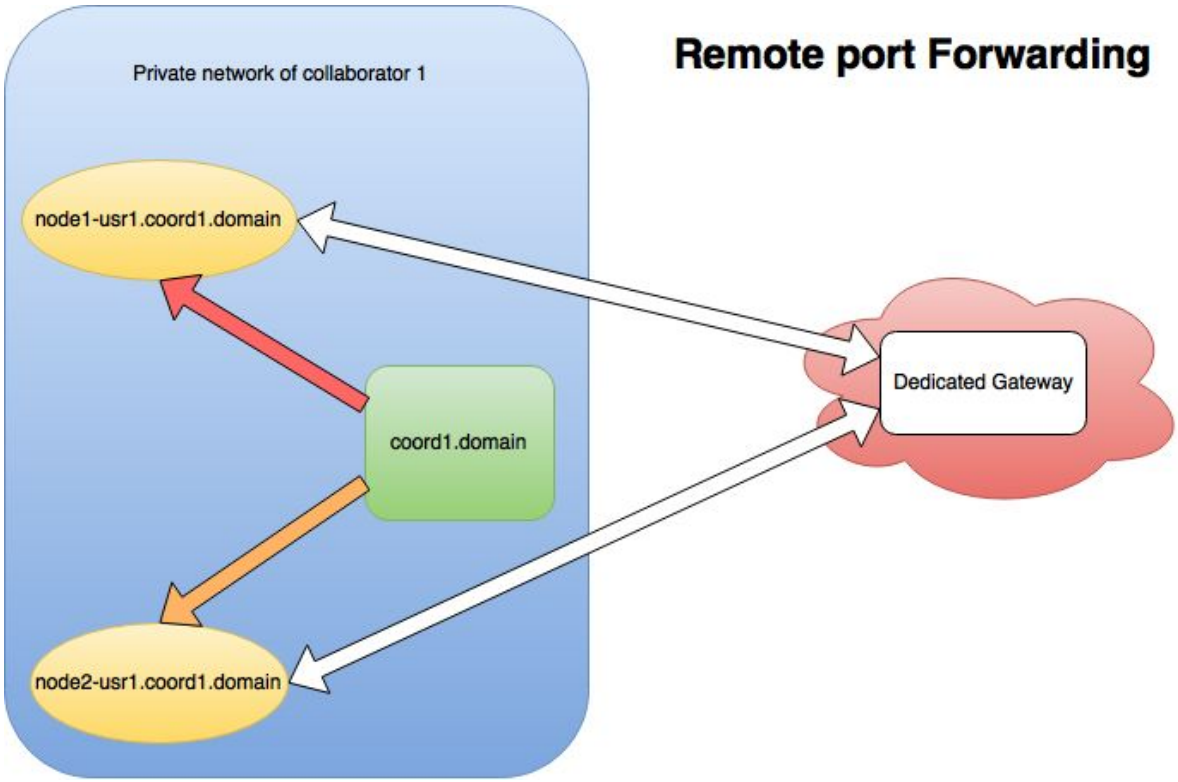
C/ Détails des composants chez le collaborateur et sur le frontend



Le frontend va centraliser plusieurs services, il est la vitrine de notre service. Les clients comme les collaborateurs s'inscrivent dessus. Le collaborateur déclare combien d'instances il sera en mesure d'héberger, par exemple. Le client devra passer par le frontend pour demander une allocation d'instances, avant de pouvoir se connecter directement via ssh aux dites instances.

Le noeud "coordinateur" chez le collaborateur peut être vu un peu comme un "client lourd", c'est lui qui aura la charge de communiquer avec le frontend, créer les instances, les monitorer, rediriger les flux ssh vers les destinataires finaux (SSH Jump Host).

D/ Accès public à des services tournant sur les instances



Problématique : Comment rendre accessible un service, par exemple Apache, tournant sur une instance derrière un réseau privé, sans pouvoir ouvrir les ports sur le pare-feu ?

Une solution envisagée consisterait à effectuer un tunneling SSH entre notre service hébergé localement dans un réseau privé et un portail dédié accessible (IP publique). Ce portail dédié serait dans notre cas un service supplémentaire facturé au client. Le client pourrait également selon son choix effectuer le port forwarding vers une machine accessible publiquement de son choix.

V. Choix technologiques

Les technologies que nous avons choisies d'utiliser lors de ce projet sont les suivantes:

Chaque composant (frontend, noeud coordinateur, instance docker compte un démon SSH). Le client doit également avoir un client SSH, bien entendu, pour pouvoir se connecter en SSH à ses instances allouées...

Au niveau du frontend :

- Bootstrap, pour le CSS
- MongoDB, pour la base de données
- Angular-Meteor, pour la réalisation de la webui (Meteor+ AngularJS)

Au niveau du collaborateur :

- Docker, pour les instances à héberger, pour le noeud coordinateur.
- Shinken, pour le monitoring sur le noeud coordinateur

VI. Licence

Ce projet ayant comme politique d'être open-source, tout ce qui sera utilisé lors de ce projet sera libre de droits sous licence GPL v3.

VII. Délais

La date limite de rendu du projet est fixée au 17 mars 2016, la soutenance ayant lieu le lendemain matin. Il sera cependant nécessaire de proposer d'ici là plusieurs versions se rapprochant des spécifications désirées afin de fournir un projet complet à cette date. Dans le cas où les spécifications sont réalisées plus tôt dans le projet, nous pourrons étoffer le cahier des charges.