Heartbeat

Labo-5

*Racine*

*Delessert*

*Baehler*

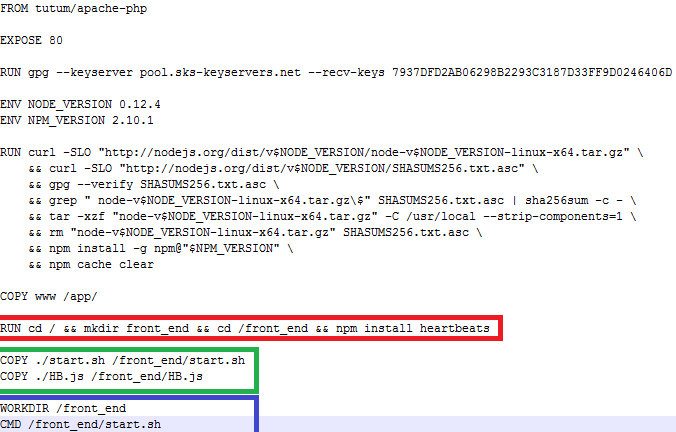
*Djeulezeck*

# Introduction

Cette documentation décrit la mise en place du Heartbeat pour le front\_end et le back\_end de notre application, il démontre aussi son comportement ainsi qu’une démo prouvant son bon fonctionnement.

# Dockerfile

Dans un premier lieu nous allons nous intéresser au Dockerfile qui se présente sous la forme suivante



Afin de simplifier l’installation de Node.Js au sien de notre conteneur Docker, nous nous somme basé sur un Dockerfile déjà existant disponible ici :

<https://github.com/joyent/docker-node/blob/428d5e69763aad1f2d8f17c883112850535e8290/0.12/wheezy/Dockerfile>

Puis nous avons ajouté les trois parties encadrées en couleur

* En rouge : nous créons le répertoire front\_end au niveau du répertoire courant, puis nous nous déplaçons dedans, et pour finir nous y installons le module nodeJS « heartbeats » (module que nous n’avons pas utilisé finalement, un chapitre y sera dédié plus bas).
* En vert : Nous plaçons nos deux fichiers (start.hs et HB.js) dans notre conteneur
  + Start.sh sera responsable du lancement d’apache et de HB.js
* En bleu : nous définissons noter répertoire de travail et nous y lançons le script start.sh

# Heartbeats de NodeJS

Durant le cours nous avons beaucoup parlé de NodeJS ansi que du terme HearthBeat ce qui nous a tournés vers le module NodeJS nommée Heartbeat disponible ici :

<https://www.npmjs.com/package/heartbeats>

Le module semblait faire ce que nous voulions, malheureusement il n’est pas bloquant est notre scripte JS s’arrêtait. Une solution plus standard a donc été envisagée.

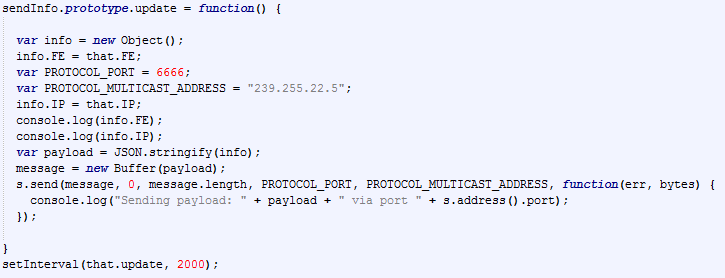
# Heartbeat

Petit définition pour commencer :

*« In computer science, a heartbeat is a periodic signal generated by hardware or software to indicate normal operation or to synchronize other parts of a system. Usually a heartbeat is sent between machines at a regular interval on the order of seconds. If a heartbeat isn't received for a time—usually a few heartbeat intervals—the machine that should have sent the heartbeat is assumed to have failed. »*

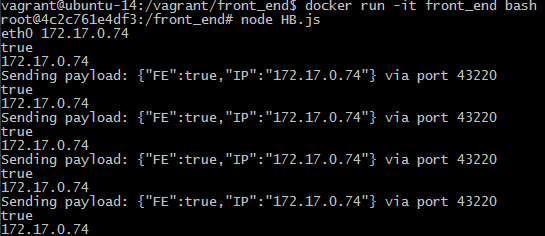
L’intérêt du heartbeat dans notre cas est d’informer le revers proxy de notre existence en lançant un broatcast toute les X seconde et en disant qui nous sommes et ce que nous savons faire. Dans notre cas le front\_end enverra son adresse id ainsi que un boolean valant « true » indiquant qu’il s’agit d’un front\_end, il en va de même dans le cas du back\_end mais celui-ci envera « false » indiquant qu’il ne s’agit pas du front\_end, par déduction nous avons qu’il s’agit du back\_end.

Dans notre cas nous envoyons un objet sérialisé en json dans un datagram udp, cette objet serialisé contient l’ip de notre machine lançant le scripte JS ainsi que un « true » dans le cas du front\_end et un « false » dans le cas du back\_end, et cela toutes les deux seconde en broatcast.



# Test

Afin de démonter et de tester le bon fonctionnement de notre heartbeat nous lançons notre front end en interactif suivit de « bash » pour ouvrir un bash. Nous pouvons observer que un message avec l’ip de notre machin et un « true » sont envoyé sous la forme d’un objet serialisable.



Ici nous observons qu’il y a bien une communication qui se crée entre le front\_end et le controlleur du revers proxy. L’objet sériablisable est bien récupéré par le contolleur

