Rapport AIT - Lab04 - Docker

Nathan Séville, Julien Quartier

Table des matières

Introduction

Tasks

Task 0: Identify issues and install the tools

> **[M1]** Do you think we can use the current solution for a production environment? What are the main problems when deploying it in a production environment?

Non, cette solution n'est pas adaptée à un environnement de production. En cas d'arrêt inopiné de *node*, aucun monitoring, ni procédure automatique n'est configurée. En cas de grande charge, aucune stratégie de *scaling* n'est définie. L'ajout de nouveau *container* est compliquée dans l'infrastructure courante (CF. **M2**).

> **[M2]** Describe what you need to do to add new `webapp` container to the infrastructure. Give the exact steps of what you have to do without modifiying the way the things are done. Hint: You probably have to modify some configuration and script files in a Docker image.

///// To complete

1. Ajouter une *webapp* dans le fichier `docker-compose.yml`.
2. Ajouter une *node* dans le fichier de configuration de *haproxy*, `haproxy.cfg`.

> **[M3]** Based on your previous answers, you have detected some issues in the current solution. Now propose a better approach at a high level.

Une meilleure solution serai de surveiller les *container* de type `webapp` et de les ajouter / retirer de manière dynamique du *load balancer*.

> **[M4]** You probably noticed that the list of web application nodes is hardcoded in the load balancer configuration. How can we manage the web app nodes in a more dynamic fashion?

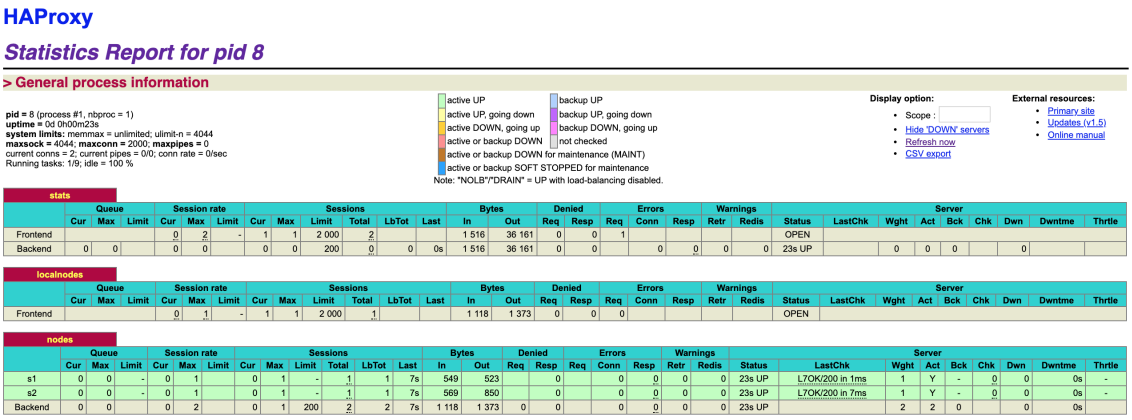
Il est possible de gérer dynamiquement les *web app nodes* de manière plus dynamique en générant le fichier de configuration du *load balancer* au démarrage, ainsi que lorsqu'une application disparaît ou disparaît dans le *pool* d'applications. Le *pool* d'applications pourrait être géré de di□érente manière en surveillant l'activité des *containers* qui peuvent par exemple s'enregistrer ou sortir du *pool*.

> **[M5]** In the physical or virtual machines of a typical infrastructure we tend to have not only one main process (like the web server or the load balancer) running, but a few additional processes on the side to perform management tasks.
>
> For example to monitor the distributed system as a whole it is common to collect in one centralized

> place all the logs produced by the di□erent machines. Therefore we need a process running on each machine that will forward the logs to the central place. (We could also imagine a central tool that reaches out to each machine to gather the logs. That's a push vs. pull problem.) It is quite common to see a push mechanism used for this kind of task.
>
> Do you think our current solution is able to run additional management processes beside the main web server / load balancer process in a container? If no, what is missing / required to reach the goal? If yes, how to proceed to run for example a log forwarding process?

Pour pouvoir lancer plusieurs processus sur une même *container* `docker`, il est possible de procéder de di□érentes manière[1], un approche privilégie l'utilisation de script à lancer lors du lancement du *container* avec la commande `CMD` de `docker` dans le *Dockerfile*. Une autre approche possible est de lancer un *process manager* comme processus principal de notre *container* et c'est lui qui s'occupera de lancer nos autre processus, par exemple la documentation `docker` propose l'utilisation de `supervisord` .

> **[M6]** In our current solution, although the load balancer configuration is changing dynamically, it doesn't follow dynamically the configuration of our distributed system when web servers are added or removed. If we take a closer look at the `run.sh` script, we see two calls to `supervisord`

1. 

HAProxy

Statistics Report for pid 8

> General process information

pid = 8 (process #1, nbproc = 1)
uptime = 0d 0h00m23s
system limits: memmax = unlimited; ulimit-n = 4044
maxsock = 4044; **maxconn =** 2000; **maxpipes =** 0
current conns = 2; current pipes = 0/0; conn rate = 0/sec
Running tasks: 1/9; idle = 100 %

active UP
active UP, going down
active DOWN, going up
active or backup DOWN
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
backup UP
backup UP, going down
backup DOWN, going up
not checked
Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:
- Scope :
- Hide 'DOWN' servers
- Refresh now
- CSV export

External resources:
- Primary site
- Updates (v1.5)
- Online manual

stats

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	2	-	1	1	2 000	2			1 516	36 161	0	0	1					OPEN								
Backend	0	0		0	0		0	0	200	0	0	0s	1 516	36 161	0	0		0	0	0	0	23s UP		0	0	0		0		

localnodes

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	1	-	1	1	2 000	1			1 118	1 373	0	0	0					OPEN								

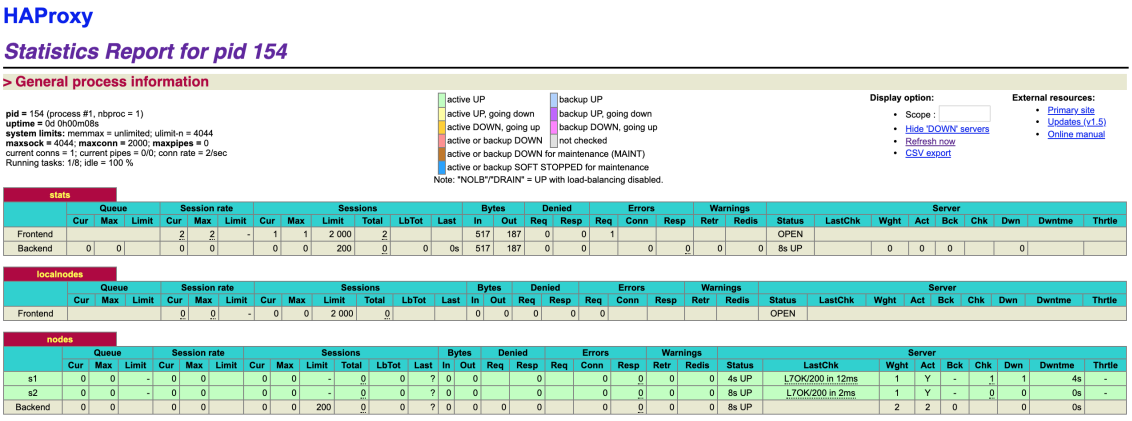
nodes

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
s1	0	0	-	0	1		0	1	-	1	1	7s	549	523		0		0	0	0	0	23s UP	L7OK/200 in 1ms	1	Y	-	0	0	0s	-
s2	0	0	-	0	1		0	1	-	1	1	7s	569	850		0		0	0	0	0	23s UP	L7OK/200 in 7ms	1	Y	-	0	0	0s	-
Backend	0	0		0	2		0	1	200	2	2	7s	1 118	1 373	0	0		0	0	0	0	23s UP		2	2	0		0	0s	

2. https://github.com/nathanseville/Teaching-HEIGVD-AIT-2019-Labo-Docker

Task 1: Add a process supervisor to run several processes

Deliverables

1. 

HAProxy

Statistics Report for pid 154

> General process information

pid = 154 (process #1, nbproc = 1)
uptime = 0d 0h00m08s
system limits: memmax = unlimited; ulimit-n = 4044
maxsock = 4044; **maxconn =** 2000; **maxpipes =** 0
current conns = 1; current pipes = 0/0; conn rate = 2/sec
Running tasks: 1/8; idle = 100 %

active UP
active UP, going down
active DOWN, going up
active or backup DOWN
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
backup UP
backup UP, going down
backup DOWN, going up
not checked
Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:
- Scope :
- Hide 'DOWN' servers
- Refresh now
- CSV export

External resources:
- Primary site
- Updates (v1.5)
- Online manual

stats

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				2	2	-	1	1	2 000	2			517	187	0	0	1					OPEN								
Backend	0	0		0	0		0	0	200	0	0	0s	517	187	0	0		0	0	0	0	8s UP		0	0	0		0		

localnodes

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	2 000	0			0	0	0	0	0					OPEN								

nodes

	Queue			Session rate			Sessions							Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
s1	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	4s UP	L7OK/200 in 12ms	1	Y	-	1	1	4s	-
s2	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	8s UP	L7OK/200 in 2ms	1	Y	-	0	0	0s	-
Backend	0	0		0	0		0	0	200	0	0	?	0	0	0	0		0	0	0	0	8s UP		2	2	0		0	0s	

2. L'installation d'un *process supervisor* tel que `S6` va nous permettre de lancer plusieurs processus par *container* `docker`, c'est important qu'on puisse le faire afin de pouvoir ajouter par exemple un processus qui communique sur l'état de la *node* pour pouvoir les manager correctement en fonction de leur état.

// TO COMPLETE

Task 2: Add a tool to manage membership in the web server cluster

1. Les logs sont dans le dossier `logs` à la racine du git.
2. Le problème c'est que notre `Serf` sur le HAProxy ne prend pas en charge la gestion des membres des autres *containers*.

3. Le `GOSSIP` protocole est principalement basé sur le protocole `SWIM`, il permet de propager l'information rapidement en broadcastant à tout ses voisins son état actuel.Le protocole se base essentiellement sur `UDP` pour accomplir sa tâche. La détection de *node* "down" se fait en trois étapes, les nodes e□ectuent des vérifications régulièrement pour savoir si leur voisin sont toujours "up" à l'aide de simple requête avec demande d'un `ACK`. (1) Si la *node* ne répond pas on demande à nos voisin d'e□ectuer la même requête, (2) si la *node* ne répond pas elle est marquée comme suspicieuse et l'information est *broadcatsée* au autre *node*, (3) si la node ne répond toujours pas après un intervalle de temps configurable elle est considérée comme "down" et son état est *broadcasté*.

   Une alternative serait de privilégier une méthode de type *pull*, un *manager* principal serait seul responsable de vérifier qu'il n'y ait pas de nouvelle *node* où des *nodes* en moins en e□ectuant un scan du réseau pour la détection de nouvelle *node* et en gardant l'état du *pool* de *node* afin de les interroger régulièrement pour vérifier qu'elles soient toujours "up".

Task 3: React to membership changes

1 et 2. Tout est dans le dossier `logs` à la racine du git.

Task 4: Use a template engine to easily generate configuration files

1. /// TO COMPLETE
2. Une meilleure approche est de chainer les images, une image de base avec `S6` et les instructions commune puis chaque nouvelle image di□érente demandant quelques instructions supplémentaires basée sur celle possédant les instructions commune à l'aide de la commande `FROM <image>`, ceci permet de limiter la taille des containers et images comme chaque nouvelle image partagera les *layers* de base avec les autre.
3. Dans le dossier `logs` à la racine du git.
4. //// TO COMPLETE

Task 5: Generate a new load balancer configuration when membership changes

1, 2 et 3. Dans le dossier `logs` à la racine du git.

Task 6: Make the load balancer automatically reload the new configuration

1. Tout dans le dossier `logs` à la racine du git.
2. ///// TO COMPLETE

Difficulties

Conclusion