Rapport AIT - Lab04 - Docker

Nathan Séville, Julien Quartier

Table des matières

Introduction **3**

Tasks **3**

Task 0: Identify issues and install the tools 3

Task 1: Add a process supervisor to run several processes 4

Task 2: Add a tool to manage membership in the web server cluster 5

Difficulties **5**

Conclusion **5**

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.

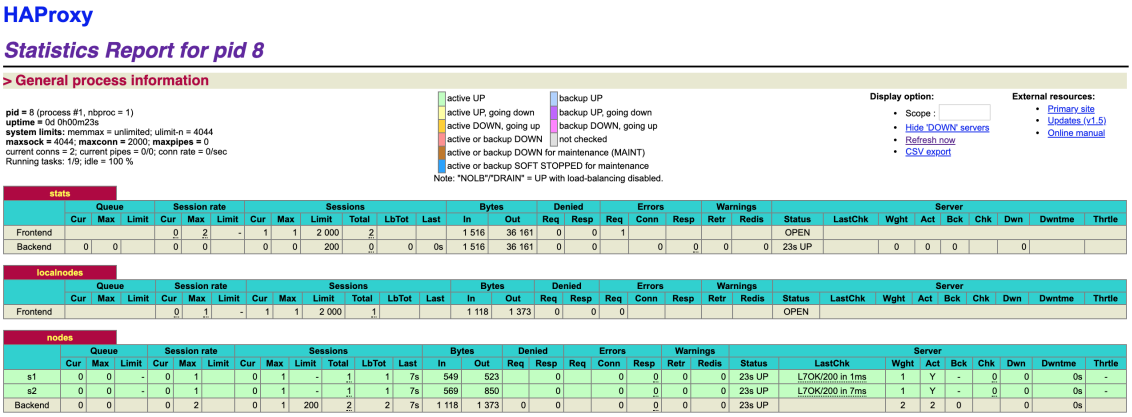
> **[M4]** You probably noticed that the list of web application nodes is hardcoded in the load balancer configuration. How can we manage the web app nodes in a more dynamic fashion?

> **[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 different 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?

> **[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 `sed` which will replace two lines in the `haproxy.cfg` configuration file just before we start `haproxy`. You clearly see that the configuration file has two lines and the script will replace these two lines.
>
> What happens if we add more web server nodes? Do you think it is really dynamic? It's far away from being a dynamic configuration. Can you propose a solution to solve this?

Deliverables

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 | backup UP
active UP, going down | backup UP, going down
active DOWN, going up | backup DOWN, going up
active or backup DOWN | not checked
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
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								

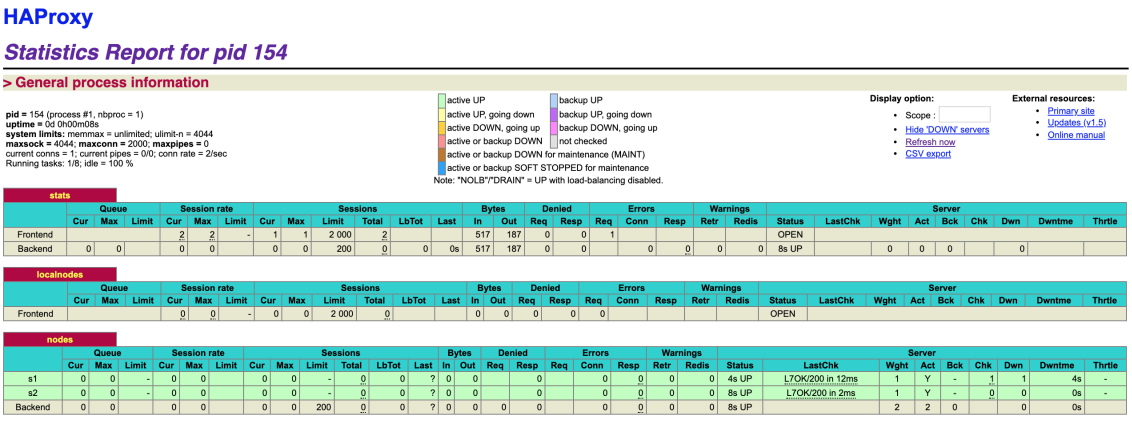
nodes

	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 | backup UP
active UP, going down | backup UP, going down
active DOWN, going up | backup DOWN, going up
active or backup DOWN | not checked
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
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. Un process supervisor est important dans une infrastructure load balancée pour pouvoir manager nos *nodes* en fonction de la charge ainsi que de relancer les *nodes* qui peuvent possiblement "crasher". Il peut s'avérer utile de supprimer les *nodes* non utilisées d'autant plus si la stratégie du *load balancer* et *first* par exemple.

// TO COMPLETE

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

Difficulties

Conclusion