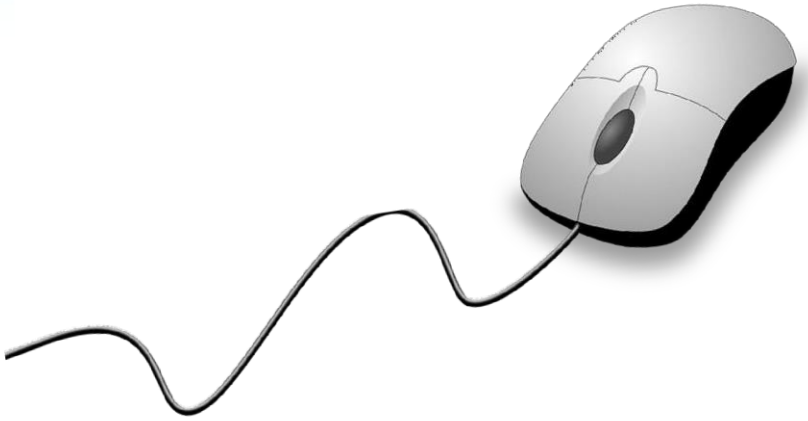
공개SW 솔루션 설치 & 활용 가이드

미들웨어 > 분산시스템SW



HAPROXY

제대로 배워보자

How to Use Open Source Software

Open Source Software Installation & Application Guide

OPENUP

오픈소스 소프트웨어 통합지원센터

Open Source Software Support Center

CONTENTS

1. 개요

소개	• HAProxy는 여러 서버에 요청을 분산시키는 TCP 및 HTTP 기반 응용 프로그램		
주요기능	• 네트워크 스위치에서 제공하는 L4, L7의 기능 및 로드 밸런싱 기능 일부 제공		
대분류	• 미들웨어	**소분류**	• 분산시스템 SW
라이선스 형태	• GPL Version 2	**사전설치 솔루션**	• N/A
운영체제	• Linux, FreeBSD, OpenBSD, Solaris, AIX	**버전**	• 1.8.14 (2018년 10월 기준)
특징	• 리버스 프록시 형태로 동작 • SSL 지원 • 로드밸런싱 3가지 알고리즘(least connection, Source phashing, weight) 지원 • L4, L7의 로드밸런싱 기능 지원 • NAT, DSR, Tunneling 기능 지원 • Caching 기능 지원 • TCP socket 통신에 대하여 이중화 처리 가능		
보안취약점	• 취약점 ID : CVE-2018-14645 • 심각도 : 7.5 HIGH(V3) • 취약점 설명 : HTTP / 2에사용되는1.8.14 이전의 HAProxy의 HPACK 디코더에서결함이발견, hpack_valid_idx ()의 범위를 벗어난 읽기 액세스로 인해 원격 크래시가발생하고 서비스 거부가 발생 • 대응방안 : 최신버전 사용 • 참고 경로 : https://access.redhat.com/errata/RHSA-2018:2882		
개발회사/커뮤니티	• HAProxy Technologies		
공식 홈페이지	• https://www.haproxy.org		

2. 기능요약

- HAProxy의 주요 기능

주요기능	지원여부
load balancing	지원
Proxy Protocol	지원
Health checking	지원
Logging	지원(Connection and HTTP message)
SSL	지원
Multithreading	지원
Rate limiting	지원

3. 실행환경

- 리눅스 운영체제에서만 가능

구분	지원여부
Windows	미지원
MacOS	미지원
Linux	지원
FreeBSD	지원
OpenBSD	지원
Solaris (8/9/10)	지원
AIX (5.1 - 5.3)	지원

4. 설치 및 실행

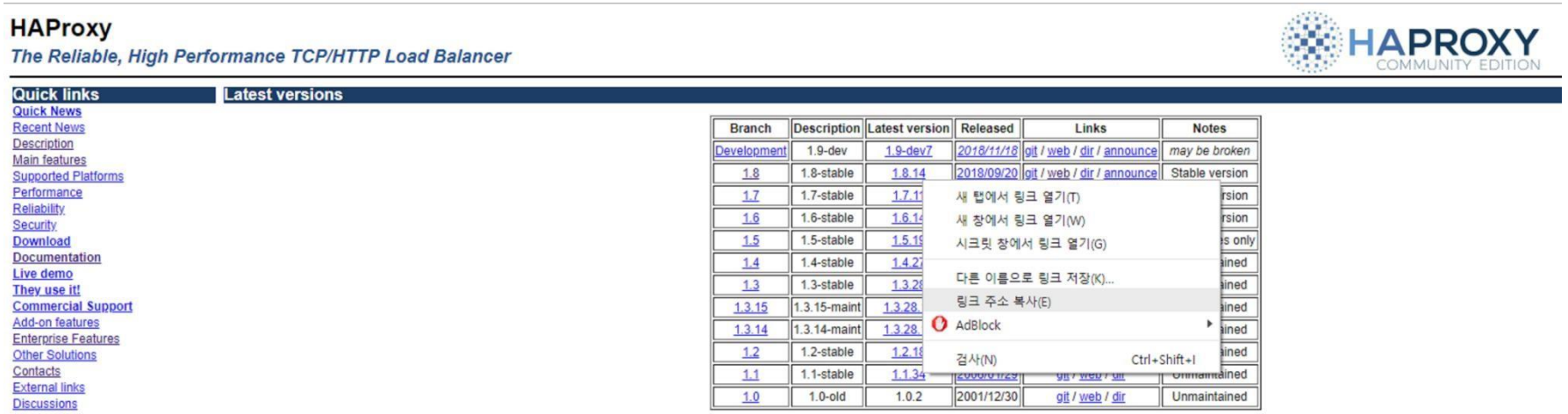
세부 목차

4.1 설치 및 설정

4. 설치 및 실행

4.1 설치 및 설정(1/5)

- http://www.haproxy.org/ 사이트에서 HAPROXY 다운로드링크 주소 복사



HAProxy

The Reliable, High Performance TCP/HTTP Load Balancer

HAPROXY COMMUNITY EDITION

Quick links

Quick News
Recent News
Description
Main features
Supported Platforms
Performance
Reliability
Security
Download
Documentation
Live demo
They use it!
Commercial Support
Add-on features
Enterprise Features
Other Solutions
Contacts
External links
Discussions

Latest versions

Branch	Description	Latest version	Released	Links	Notes
Development	1.9-dev	1.9-dev7	2018/11/18	git / web / dir / announce	may be broken
1.8	1.8-stable	1.8.14	2018/09/20	git / web / dir / announce	Stable version
1.7	1.7-stable	1.7.11			rsion
1.6	1.6-stable	1.6.14			rsion
1.5	1.5-stable	1.5.19			s only
1.4	1.4-stable	1.4.27			ained
1.3	1.3-stable	1.3.28			ained
1.3.15	1.3.15-maint	1.3.28.			ained
1.3.14	1.3.14-maint	1.3.28.			ained
1.2	1.2-stable	1.2.18			ained
1.1	1.1-stable	1.1.34	2006/01/29	git / web / dir	Unmaintained
1.0	1.0-old	1.0.2	2001/12/30	git / web / dir	Unmaintained

4. 설치 및 실행

4.1 설치 및 설정(2/5)

- 터미널에서 다운로드 및 압축해제
- $ wget http://www.haproxy.org/download/1.8/src/haproxy-1.8.14.tar.gz
- $ tar –xvzf haproxy-1.8.14.tar.gz

```
[root@localhost tuser]# wget http://www.haproxy.org/download/1.8/src/haproxy-1.8
.14.tar.gz
--2018-11-23 10:24:36--  http://www.haproxy.org/download/1.8/src/haproxy-1.8.14.
tar.gz
Resolving www.haproxy.org (www.haproxy.org)... 51.15.8.218
Connecting to www.haproxy.org (www.haproxy.org)|51.15.8.218|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2070813 (2.0M) [application/x-tar]
Saving to: 'haproxy-1.8.14.tar.gz'

100%[======================================>] 2,070,813     601KB/s   in 3.4s

2018-11-23 10:24:41 (601 KB/s) - 'haproxy-1.8.14.tar.gz' saved [2070813/2070813]

[root@localhost tuser]# tar -xvzf haproxy-1.8.14.tar.gz
```

4. 설치 및 실행

4.1 설치 및 설정(3/5)

- 압축을 해제 한 후 make를 진행해야 하는데 주의할 점은 make를 진행하기 전에 HAProxy의 README를 필수적으로 확인해 보고 진행 해야하며, 중요한 부분은 make TARGET을 지정하는 부분 및 사용중인 리눅스 커널 버전에 따라 TARGET을 지정 해줘야함

- 리눅스 커널 버전 확인
  - $ uname –r

```
[root@localhost tuser]# uname -r
3.10.0-862.el7.x86_64
```

4. 설치 및 실행

4.1 설치 및 설정(4/5)

- https 사용이 필요하기때문에 USE_OPENSSL 옵션을 활성화 함
  - $ cd haproxy-1.8.14
  - $ make TARGET=**linux2628** USE_OPENSSL=1 (사용중인 리눅스 커널 버전에 따라 지정 필요)

```
[root@localhost tuser]# uname -r
3.10.0-862.el7.x86_64
[root@localhost tuser]# cd haproxy-1.8.14
[root@localhost haproxy-1.8.14]# make TARGET=linux2628 USE_OPENSSL=1
```

```
- linux22      for Linux 2.2
- linux24      for Linux 2.4 and above (default)
- linux24e     for Linux 2.4 with support for a working epoll (> 0.21)
- linux26      for Linux 2.6 and above
- linux2628    for Linux 2.6.28, 3.x, and above (enables splice and tproxy)
- solaris      for Solaris 8 or 10 (others untested)
- freebsd      for FreeBSD 5 to 10 (others untested)
- netbsd       for NetBSD
- osx          for Mac OS/X
- openbsd      for OpenBSD 5.7 and above
- aix51        for AIX 5.1
- aix52        for AIX 5.2
- cygwin       for Cygwin
- haiku        for Haiku
- generic      for any other OS or version.
- custom       to manually adjust every setting
```

4. 설치 및 실행

4.1 설치 및 설정(5/5)

- 컴파일 진행중 오류가 발생한다면 HAProxy과 의존성이 있는 라이브러리가 설치가 되어 있지 않아서 발생할 경우가 많음

```
In file included from include/types/global.h:32:0,
                 from src/ev_poll.c:26:
include/types/listener.h:29:25: fatal error: openssl/ssl.h: 그런 파일이나 디렉터
리가 없습니다
 #include <openssl/ssl.h>
                         ^
compilation terminated.
make: *** [src/ev_poll.o] 오류 1
[root@localhost haproxy-1.8.14]# yum -y install openssl-devel
```

  - $ yum –y install openssl-devel

- 정상적으로 설치가 완료되었다면 다시 make 진행
  - $ make TARGET=**linux2628** USE_OPENSSL=1(사용중인 리눅스 커널 버전에 따라 지정 필요)
  - $ sudo make install

5. 기능소개

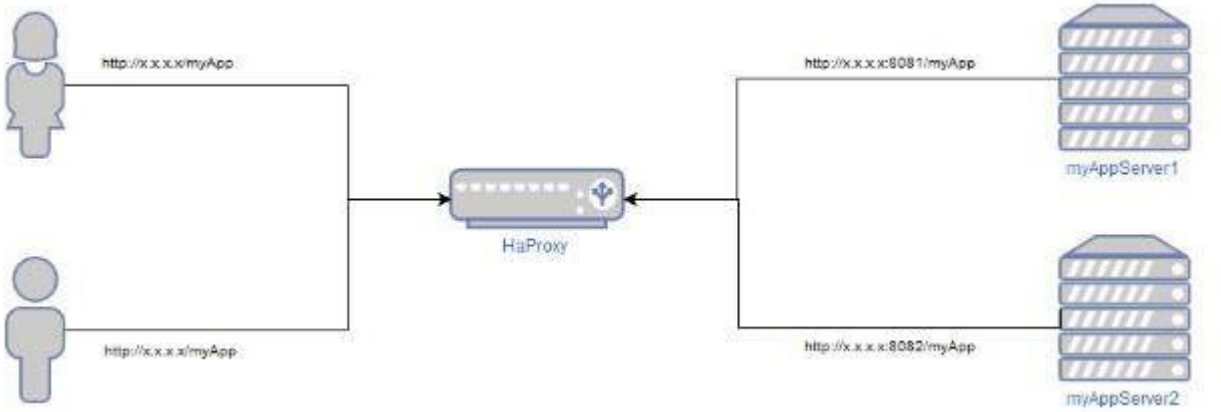
세부 목차

1. Proxying
2. SSL
3. 모니터링
4. 고가용성
5. Load balancing(로드밸런싱)
6. 콘텐츠 전환
7. Logging

5. 기능소개

5.1 Proxying(1/2)

- Proxying 은 두 개의 독립적인 연결을 통해 클라이언트와 서버간에 데이터를 전송하는작업이며, Proxying 및 연결관리와 관련하여 HAProxy가 지원하는 기본기능은 다음과 같음
  - 서버에 깨끗한 연결을 제공하여 클라이언트 측 결함 또는 공격으로부터 보호
  - 여러 IP 주소 및 포트, 심지어 포트 범위까지 Listen
  - Transparent accept : 로컬 시스템에 속하지 않는 임의의 IP 주소를 대상으로 트래픽 차단
  - 서버 포트는 수신 포트와 관련되지 않아도 되며 고정된 오프셋으로 변환 가능
  - Transparent connect : 서버에 연결할 때 클라이언트(또는 기타) IP 주소 스푸핑
  - 버퍼 및 짧은 연결로 인해 서버를 오프로드하여 동시 연결 수와 메모리 공간 줄임



5. 기능소개

1. Proxying(2/2)

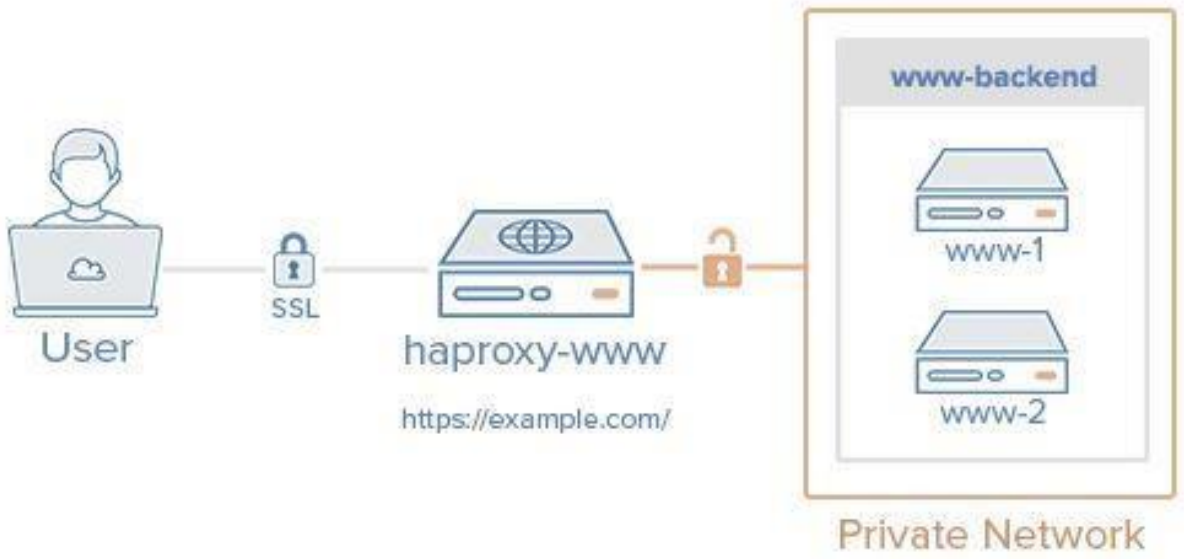
- TCP 스택(예: SACK), 혼잡 제어, RTT 영향 감소
- 양쪽에서 서로 다른 프로토콜 지원 (예: IPv4/IPv6/Unix)

- Timeout enforcement : HAProxy는 연결 단계에 따라 여러 수준의 시간 초과를 지원하므로 클라이언트나 서버 또는 공격자에게 너무 오랫동안 리소스 부여 안됨
- Protocol validation : HTTP, SSL 또는 paylord가 검사되고 허용되지 않는 한 잘못된 프로토콜 요소 거부
- Policy enforcement : 허용된 것만 전달할 수 있는지 확인
- 들어오는 연결과 나가는 연결은 모두 특정 네트워크 네임스페이스(Linux만 해당)로 제한되므로 cross-container, multi-tenant load balancer를 쉽게 구축 가능
- PROXY 프로토콜은 HTTP 트래픽이 아닌 경우에도 클라이언트의 IP 주소의 서버 제공

5. 기능소개

5.2 SSL(1/2)

- 와일드카드 인증서를 지원하므로 여러 인증서에 대한 필요성이 줄어듦
- 유효한 인증서를 제시하지 못한 경우 구성 가능한 정책을 사용하여 인증서 기반 클라이언트 인증
- TLS NPN 및 ALPN 확장을 통해 SPDY/HTTP2 연결을 안정적으로 오프로드하여 백엔드 서버에 전달 가능
- OCSP 스테이플링은 클라이언트가 인증서 상태 요청을 요청할 때 OCSP 응답을 인라인으로 전달하여 첫 페이지 로드 시간 단축
- 특정 버전의 OpenSSL에 영향을 미치는 Heartbleed 공격과 같이 취약한 SSL 라이브러리에서도 알려진 특정 공격을 탐지, 로깅 및 차단



5. 기능소개

5.2 SSL(2/2)

- 동적 레코드 크기 조정은 높은 성능과 짧은 대기 시간을 모두 제공하며패킷이 이동 중인 동안 브라우저가 새 개체를 가져오기 시작할 수 있도록 하여 페이지 로드 시간 감소
- logging, access control, reporting 등에 대한 모든 관련 SSL/TLS 계층 정보에 대한 영구적으로 접근 할 수 있으며, 이러한 요소는 HTTP 헤더에 포함되거나 PROXY 프로토콜 확장으로 포함될 수 있으므로 오프로드된 서버가 SSL 종료 자체를 수행할 때 가졌던 모든 정보 획득
- 백엔드 서버 인증은 백엔드 서버가 백엔드 서버에 연결하는 데 필요한 haproxy 노드임을 알 수 있도록 함
- 상태 비저장 세션 재개 지원함
- TLS 티켓은 CLI에서 업데이트 가능

5. 기능소개

5.3 모니터링(1/2)

- 서버별 매개 변수를 사용하여 서버 상태를 지속적으로 모니터링
- 다른 주소/포트/프로토콜로 확인을 보낼 수 있음
- 서버는 다른 서버를 추적하여 동시 다운 가능
- 부하 및 상태를 모니터링하기 위해 에이전트를 서버에 배포할 수 있다. 서버에서 간단한 에이전트를 실행하면 전체 경로의 유효성을 검사하는 상태 확인 뿐 아니라 서버 자체의 상태를 고려 가능
- TCP 연결, HTTP 요청, SMTP hello, SSL hello, LDAP, SQL, Redis, send / expect 스크립트, SSL의 유무에 관계없이 다양한 검사 방법을 사용 가능
- 상태 변경은 로그 및 통계 페이지에서실패이유(예: 오류가 감지된 순간 수신한 HTTP 응답)와 함께 통보

HAProxy

Statistics Report for pid 235

> General process information

pid = 235 (process #1, nbproc = 1)
uptime = 0d 0h42m28s
system limits: memmax = unlimited; ulimit-n = 4035
maxsock = 4035; **maxconn =** 2000; **maxpipes =** 0
current conns = 1; current pipes = 0/0; conn rate = 1/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

localhost

	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	3	-	0	2	2 000	29			13 553	23 470	0	0	1					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
web01	0	0	-	0	3		0	1	-	23	23	6m57s	5 685	9 455		0		0	0	0	0	33m57s UP	L7OK/200 in 0ms	1	Y	-	3	1	10s	-
web02	0	0	-	0	3		0	1	-	14	14	6m57s	3 104	5 376		0		0	0	0	0	33m58s UP	L7OK/200 in 0ms	1	Y	-	3	1	8s	-
web03	0	0	-	0	3		0	1	-	23	23	6m58s	4 764	8 451		0		0	0	0	0	33m58s UP	L7OK/200 in 0ms	1	Y	-	3	1	8s	-
Backend	0	0		0	9		0	1	200	60	60	6m57s	13 553	23 282	0	0		0	0	0	0	33m58s UP		3	3	0		1	8s	

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				1	2	-	1	2	2 000	8			4 864	219 272	0	0	1					OPEN								
Backend	0	0		0	0		0	0	200	0	0	0s	4 864	219 272	0	0		0	0	0	0	42m28s UP		0	0	0		0		

5. 기능소개

5.3 모니터링(2/2)

- 서버 상태는 통계 인터페이스에서 보고되며 트래픽의 크기또는상태에따라(예: DC 간 링크 손실) 다른 팜으로 트래픽이 전송될 수 있도록 라우팅 결정을 내리는 데 사용 가능
- 서버는 상태 점검을 통해 On/Off 상태보다 더 자세한 상태를 보고 가능
- HAProxy는 상태 확인 요청을 사용 하여 이름, 무게, 팜의 다른 서버 수 등과 같은 정보를 서버에 전달함으로써 서버가 이에 따라 응답 및 결정을 조정 가능
- HAProxy 자체는 라우터 또는 기타 로드 밸런서와 같은 외부 구성요소에 상태를 보고할 수 있으므로매우 완전한 다중 경로 및 다중 계층 인프라를 구축 가능

5. 기능소개

5.4 고가용성

- 유효한 서버만 사용되며 다른 서버들은 자동으로 로드 밸런싱 팜에서 제외되며, 특정 조건에서도 서버를 강제로 사용 가능
- 연결에 영향을 미치지 않고 팜에서 서버를 꺼낼 수 있도록 정상 종료 지원
- 활성 서버가 다운되면 백업 서버가 자동으로 사용되고 가능한 한 세션이 손실되지 않도록 대체함
- 또한 여러 경로를 구축해 동일한 서버에 도달 가능
- 너무 많은 서버가 다운된 경우 팜에 대해 글로벌 장애 상태를 반환할 수 있음
- Stateless 설계로 클러스터를 쉽게 구축 가능
- 표준 VRRP 데몬 keepalived와 잘 통합됨

5. 기능소개

5.5 load balancing

- 9개 이상의 로드 밸런싱 알고리즘 지원
  - 가장 일반적인 서버는 round-robin : 짧은 연결의 경우 각 서버를 차례로 선택
  - Lestconn : 긴 연결의 경우 가장 적은 연결 개수의 서버 사용
  - 소스 : SSL 팜 또는 터미널 서버의 경우
  - URI : HTTP 캐시의 경우 서버는 HTTP URI에 직접 종속됨
  - Hdr : 서버는 특정 HTTP 헤더 필드의 내용에 직접 종속됨
- 위의 모든 알고리즘은 서버별 가중치를 지원하므로 팜에서 서로 다른 서버 세대로부터 트래픽을 일부만 특정 서버로 전송 가능(디버그 모드, 다음 버전의 소프트웨어 실행 등)
- Round-robin, leastconn 및 consistent hashin에 대해 동적 가중치가 지원되며, 이를 통해 CLI에서 또는 서버에서 실행 중인 에이전트에 의해 서버 가중치를 즉시 수정 가능
- 해싱은 클라이언트의 소스 주소, URL 구성 요소, 쿼리 문자열 요소, 헤더 필드 값, POST 매개 변수, RDP 쿠키 등의 다양한 요소에 적용될 수 있음
- 서버당 연결 수, 백엔드 당 연결 슬롯 수, 백엔드에서 사용 가능한 연결 슬롯의 양과 같은 여러 내부 메트릭을 통해 매우 고급 로드 밸런싱 전략 구축 가능
- 일관된 해싱은 팜에 서버를 추가하거나 제거할 때 서버 팜을 대규모 재분산으로부터 보호
- 대형 캐시 팜에서는 매우 중요하며 저속 시동 기능을 사용하여 콜드 캐쉬를 재충전 가능

5. 기능소개

5.6 콘텐츠 전환

- HAProxy는 컨텐츠 기반 스위칭 이라고 하는 메커니즘 구현
- 원칙적으로 연결 또는 요청이 프런트 엔드에 도착한 다음 이 요청 또는 연결과 함께 전송되는 정보가 처리되며 이 시점에서 ACL 기반 조건을 작성하여 이 정보를 사용하여 어떤 백엔드가 프로세스를 처리할지 결정 가능
- 따라서 트래픽은 요청 내용을 기준으로 한 백엔드 또는 다른 백엔드로 연결
- 콘텐츠 교환의 또 다른 사용 사례는 다양한 기준에 따라 서로 다른 로드 밸런싱 알고리즘을 사용
- 애플리케이션이 라운드 로빈을 사용하는 동안 캐시는 URI 해시를 사용 가능

5. 기능소개

5.7 Logging

- HAProxy는 밀리초 단위의 정확도와 방화벽 로그에서 검색 할 수 있는 정확한 연결 승인 시간 (예 : NAT 상관 관계)을 포함한 매우 자세한 로그 제공
- 기본적으로 TCP 및 HTTP 로그에는 원본 IP 주소 및 포트, 프론트 엔드, 백엔드, 서버, 타이머 (요청 수신 기간, 대기열 지속 시간, 대기열 지속 시간, 응답 시간, 데이터 전송 시간), 전역 프로세스 상태, 연결 수, 대기열 상태, 재시도 횟수, 자세한 고정 작업 및 연결 해제 이유, 안전 출력 인코딩의 헤더 캡처등이포함
- 그런 다음 샘플 형식의 데이터, 변수, 캡처를 포함하도록 이 형식을 확장하거나 대체하여 매우 자세한 정보를 얻을 수 있음

6. 활용예제

세부 목차

1. HAProxy 설정
2. HAProxy 명령어

6. 활용예제

6.1 HAProxy 설정(1/3)

- haproxy.cfg 파일 생성
- examlple 디렉토리 내의 cfg파일을 참고하여 /etc/haproxy 디렉토리 내에 haproxy.cfg 파일 생성

  - $ mkdir -p /etc/haproxy
  - $ mkdir-p /var/lib/haproxy
  - $ touch /var/lib/haproxy/stats
  - $ vi /etc/haproxy/haproxy.cfg

6. 활용예제

6.1 HAProxy 설정(2/3)

- default-server inter 1s fall 3 rise 2 : 1초 마다 서버에 접속하여 헬스체크를 행하되 3번 실패하면 접속 불가로 판단하고 2번 성공하면 정상 상태로 간주
- Option httpchk GET / : http GET 으로 [서버주소] 요청을 날려서 http-check expect status 200 에 따라 200 OK에 해당하는 응답이 나올 시 정상 상태라고 판단

- 서비스 등록
- /etc/init.d 디렉토리는 서비스 데몬 제어 파일을 저장하는 디렉토리. service에 등록하기 위해 init.d 디렉토리 내에 example/haproxy.init 파일을 복사한 후 permission을 변경함
  - $ cp ~/haproxy-1.8.14/examples/haproxy.init /etc/init.d/haproxy
  - $ chmod 755 /etc/init.d/haproxy

- 명령어 등록
- /user/sbin 디렉터리는 명령어가 저장되는 디렉터리
- haproxy 명령어를 사용할 수 있도록 /usr/sbin 디렉터리에 링크를 설정
  - $ ln -s /usr/local/sbin/haproxy /usr/sbin/haproxy

- 사용자 등록
  - $ sudo useradd -r haproxy

6. 활용예제

6.1 HAProxy 설정(3/3)

- 서비스 시작 및 등록
- 서비스 시작 및 등록을 하기 앞서 설정한 값들을 적용시키기 위해 재시작 진행
  - $ reboot
  - $ systemctl start haproxy
- 부팅할 때 자동 실행
  - $ systemctl enable haproxy
- 방화벽
- 방화벽에 의해 HAProxy 모니터링 사이트에 접속이 되지 않는다면 다음 명령어를 입력하여 방화벽 제한 해제
  - $ firewall-cmd --permanent --zone=public --add-service=http
  - $ firewall-cmd --permanent --zone=public --add-port=[PORT]/tcp
  - $ firewall-cmd --reload

6. 활용예제

6.2 HAProxy 명령어(1/2)

- HAProxy 프로그램을 호출 함으로써 시작

  - $ haproxy [<options>] *

    * [<options>] * 는 여러 옵션이며, 옵션은 항상 – 로 시작
    * -<cfgfile> * : - 다음에 오는 모든 인수는 구성 경로이며, 파일 / 디렉터리를 선언 순서대로 로드하고 처리
    * -f <cfgfile | cfgdir> : <cfgfile>을 로드할 구성 파일 목록 추가
    * -C <dir> : 구성을 로드하기전에 <dir> 디렉터리로 변경하며, 이 기능은 상대 경로를 사용할 때 유용
    * -D : 데몬으로 시작함, 프로세스가 현재 터미널에서 분리되며, 분기 및 오류는 터미널에서 더 이상 보고 안됨
    * -L <name> : 로컬 피어 이름을 <name> 으로 변경하며, 기본값은 로컬 호스트 이름
    * -N <limit> : 기본값(일반적으로 2000) 대신 프록시당 기본 maxconn을 "limit" 설정
    * -V : 상세 모드를 활성화하며, -q 또는 –quiet 의 효과 되돌림
    * -W : master-worker 모드
    * -Ws : notification(알림) 방식의 systemd 서비스를 지원하는 master-worker(마스터-worker) 모드
    * -c : 구성 파일의 확인만 수행하고바인딩을시도하기전에종료하며, 모든 항목이 정상인 경우 종료상태는 0이고 오류가 발생한 경우 0이 아님
    * -d : 디버그 모드를 활성화하며, 데몬 모드가 비활성화되고 프로세스가 화면 전면에 유지되고 수신 및 송신 이벤트 표시
    * -dG : getaddrinfo()를 사용하여 호스트 이름을 주소로 확인할 수 없도록 설정

6. 활용예제

2. HAProxy 명령어(2/2)

* -dM[<byte>] : memory poisoning 강제하며, malloc() 또는 pool_alloc()으로 할당된 모든 메모리 영역이 호출자에게 전달되기 전에 <byte> 로 채워지며, <byte>가 지정되지 않은 경우 기본값은 0x50(' P ') 임
  * -dS : splice() system 호출의 사용 비활성화
  * -dV : 서버 측에서 SSL 확인을 사용 불가능으로 설정
  * -db : 백그라운드 모드와 다중 프로세스 모드 비활성화
  * -de : "epoll" 폴러의 사용 비활성화
  * -dk : "kqueue" 폴러의 사용 비활성화
  * -dp : "poll" 폴러의 사용 비활성화
  * -dr : 서버 주소 확인 실패 무시
  * -m <limit> : 모든 프로세스에서 할당 가능한 총 메모리를 <limit>MB 제한
  * -n <limit> : 프로세스별 연결 제한을 <limit> 제한
  * -p <file> : 시작 시 모든 프로세스의 pid를 <file> 기록
  * -q : "quiet"모드를 설정한다. 이렇게하면 구성 중에 일부 메시지 비활성화
* -sf <pid>* : 부팅이 완료된 후 "완료" 신호(SIGUSR1)를 이전 프로세스에 전송하여 작업 중인 작업을 완료하고 떠나도록 요청 <pid>는 신호를 보낼 패드 리스트(인수당 1개) 목록은 "-"로 시작하는 모든 옵션으로 끝남
  * -v : 버전 및 제조일자 보고
  * -vv : 버전, 빌드 옵션, 라이브러리 버전 및 사용 가능한 폴러 표시
  * -x <unix_socket> : 지정된 소켓에 연결하고 이전 프로세스에서 수신 소켓을 검색한 후 새 소켓을 바인딩하는 대신 사용

7. FAQ

Q **[ALERT] 015/013939 (22640) : Starting frontend public: cannot bind socket [0.0.0.0:80] 오류가 나요?**

A

SELinux 보안 정책과 충돌이 나서 그렇습니다. haproxy가 생성한 포트를 모두 허용하도록 설정하면 됩니다. 설정 방법은 $ setsebool -P haproxy_connect_any 1 입니다.

8. 용어정리

용어	설 명
ssl	SSL(Secure Socket Layer)은 웹 서버와 브라우저 간에 암호화된 링크를 설정하기 위한 표준 보안 기술이며, 보안 링크를 사용하면 전송된 모든 데이터가 비공개 유지됨
Load balancing	부하분산 또는 로드 밸런싱(load balancing)은 컴퓨터 네트워크 기술의 일종으로 둘 혹은 셋이상의 중앙처리장치 혹은 저장장치와 같은 컴퓨터 자원들에게 작업을 나누는 것 의미하며, 가용성 및 응답시간을최적화가능

Open Source Software Installation & Application Guide