Since **HAProxy** is located between users and servers, it is aware of anything that happened during the request.
The present memo introduces the very verbose HAProxy HTTP logs.

HAProxy Log line example

Mar 9 15:08:05	LB1	local0.info	haproxy[21843]:	10.0.0.1:1028	[09/Mar/2012:15:08:05.179]	FT	BK/SRV	0/0/1/8/9	304	12	- -	--VN	4/4/0/1/0	0/0	"GET / HTTP/1.1"
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Field name and definition

#	Example's Value	Name	Custom log tag	Short description
1	Mar 9 15:08:05			Date at which the log has been emitted
2	LB1			Aloha's name
3	local0.info			Syslog facility
4	haproxy[21843]:	process_name '[' pid ']':		HAProxy process' name in the Aloha and its PID.
5	10.0.0.1:1028	client_ip ':' client_port	%Ci %Cp	**client_ip**: IP address of the client which initiated the TCP connection to HAProxy **client_port**: TCP port of the client which initiated the connection
6	[09/Mar/2012:15:08:05.179]	accept_date	%t	exact date when the TCP connection was received by haproxy
7	FT	frontend_name	%f	name of the frontend (or listener) which received and processed the connection
8	BK/SRV	backend_name'/'server_name	%b/%s	**backend_name**: name of the backend (or listener) which was selected to manage the connection to the server **server_name**: name of the last server to which the connection was sent
9	0/0/1/8/9	Tq '/' Tw '/' Tc '/' Tr '/' Tt*	%Tq %Tw %Tc %Tr %Tt	**Tq**: total time in milliseconds spent waiting for the client to send a full HTTP request, not counting data **Tw**: total time in milliseconds spent waiting in the various queues **Tc**: total time in milliseconds spent waiting for the connection to establish to the final server, including retries **Tr**: total time in milliseconds spent waiting for the server to send a full HTTP response, not counting data **Tt**: total time in milliseconds elapsed between the accept and the last close. It covers all possible processings
10	304	status_code	%st	HTTP status code returned to the client
11	12	bytes_read	%B	total number of bytes transmitted to the client when the log is emitted
12	- -	captured_request_cookie captured_response_cookie	%cc %cs	**captured_request_cookie**: optional "name=value" entry indicating that the client had this cookie in the request **captured_response_cookie**: optional "name=value" entry indicating that the server has returned a cookie with its response
13	--VN	termination_state cookie_status	%tsc	**termination_state**: condition the session was in when the session ended **cookie_status**: status of cookie persistence
14	4/4/0/1/0	actconn '/' feconn '/' beconn '/' srv_conn '/' retries	%ac %fc %bc %sc %rc	**actconn**: total number of concurrent connections on the process when the session was logged **feconn**: total number of concurrent connections on the frontend when the session was logged **beconn**: total number of concurrent connections handled by the backend when the session was logged **srv_conn**: total number of concurrent connections still active on the server when the session was logged **retries**: number of connection retries experienced by this session when trying to connect to the server
15	0/0	srv_queue'/'backend_queue	%sq/%bq	**srv_queue**: total number of requests which were processed before this one in the server queue **backend_queue**: total number of requests which were processed before this one in the backend's global queue
	N/A for the example above	captured_request_headers captured_response_headers	%hr %hs	**captured_request_headers**: list of headers captured in the request due to the presence of the "capture request header" statement in the frontend **captured_response_headers**: list of headers captured in the response due to the presence of the "capture response header" statement in the frontend
16	"GET / HTTP/1.1"	http_request	%{+Q}r	the complete HTTP request line