HAPROXY

- Configuração e Customização HAPROXY
  - 2 Ways to Enable Sticky Sessions in HAProxy (Guide)
  - HAPROXY com persistência de URL

Configuração e Customização HAPROXY

2 Ways to Enable Sticky Sessions in HAProxy (Guide)

Link: https://www.haproxy.com/blog/enable-sticky-sessions-in-haproxy

HyperText Transfer Protocol (HTTP), the protocol that defines the language browsers use to communicate with web servers, is stateless, meaning that after you make a web request and a server sends back a response, no memory of that interaction remains. Websites need other ways to remember previous user interactions to make anything more sophisticated than a static web page work.

These days, Javascript frameworks like Vue.js, React.js, and others let developers create single-page applications (SPAs) that provide statefulness to the otherwise stateless web. Because they're implemented as Javascript applications running in the user's browser, they can keep track of what the user has done and render the app in a way that accounts for that shared history. However, the user's browser

Nick creates technical content for HAProxy Technologies ranging from documentation and blog posts to Wikipedia articles, GitHub READMEs and Stack Overflow answers. With a background in web development and DevOps, he has fun digging into product features and discovering the optimal path for a new blog tutorial.

Twitter LinkedIn GitHub

Related Posts

How to Enable Health Checks in HAProxy (Guide)
September 14th, 2021

How to Enable Health Checks in HAProxy (Guide)

HAProxy provides three types of health checks: active health checks, passive health checks and agent health checks. In this blog, we will learn about each one.

2 Ways to Implement a Circuit Breaker in HAProxy (Guide)
January 5th, 2021

2 Ways to Implement a Circuit Breaker in HAProxy (Guide)

In this blog post, you'll learn how to implement a circuit breaker with HAProxy in a simple and more complex way, that allows greater customization.

Load Balancing, Affinity, Persistence & Sticky Sessions
March 29th, 2012

Load Balancing, Affinity, Persistence & Sticky Sessions

This blog post shows why and how to use a load balancer, the differences between Affinity, Persistence, and Sticky Sessions, and how to enable them in HAProxy.

Preserve Source IP Address Despite Reverse Proxies (Guide)
June 5th, 2012

Preserve Source IP Address Despite Reverse Proxies (Guide)

In this blog post, we aim to demonstrate how to preserve source IP addresses when using the reverse-proxy mode in HAProxy.

HAPROXY com persistência de URL

Informações geradas pelo ChatGPT.

Sim, é possível configurar o HAProxy para atender tanto **HTTP** (porta 80) quanto **HTTPS** (porta 443) ao mesmo tempo. Isso pode ser feito configurando dois **frontends separados** — um para HTTP e outro para HTTPS — e ambos encaminhando as requisições para o mesmo backend, de modo que o balanceamento de carga e a persistência por URL sejam aplicados de maneira uniforme para ambos os protocolos.

Exemplo de Configuração para HTTP e HTTPS ao Mesmo Tempo

Aqui está um exemplo de configuração onde o HAProxy aceita tráfego HTTP e HTTPS e roteia as requisições para o mesmo backend, mantendo a persistência baseada na URL para ambos os protocolos:

haproxy

Copiar

```
global
    log /dev/log local0
    log /dev/log local1
    notice
   maxconn 2000

defaults
     log     global
     option  httplog
     option  dontlognull
     timeout connect 5000ms
     timeout client  50000ms
     timeout server  50000ms

# Frontend HTTP (porta 80)
frontend http_front
    bind *:80
     acl is_http_request path_reg ^/.*  # Captura qualquer URL
```

```
    use_backend default_backend if is_http_request

# Frontend HTTPS (porta 443)
frontend https_front
    bind *:443 ssl crt /etc/ssl/certs/meucertificado.pem  # Certificado SSL
    acl is_https_request path_reg ^/.*  # Captura qualquer URL
    use_backend default_backend if is_https_request

# Backend
    backend default_backend
    balance roundrobin
  stick-table type string size 200k expire 30m store gpc0
  stick on url_param(url)  # Persistência baseada na URL
    cookie SERVERID insert indirect nocache
  server server1 192.168.1.1:80 check  # Servidor HTTP
    server server2 192.168.1.2:80 check  # Servidor HTTP
    server server3 192.168.1.3:80 check  # Servidor HTTP
```

Explicação da Configuração:

1. **Frontend HTTP**:
   - **`bind *:80`**: O frontend escuta na porta 80 para tráfego HTTP.
   - **`acl is_http_request path_reg ^/.*`**: A ACL captura qualquer URL para redirecioná-la para o backend.
   - **`use_backend default_backend if is_http_request`**: Se a requisição for HTTP, ela é direcionada para o `default_backend`.
2. **Frontend HTTPS**:
   - **`bind *:443 ssl crt /etc/ssl/certs/meucertificado.pem`**: Este frontend escuta na porta 443 para tráfego HTTPS e usa o certificado SSL fornecido no arquivo `/etc/ssl/certs/meucertificado.pem`.
   - **`acl is_https_request path_reg ^/.*`**: A ACL captura qualquer URL, assim como no frontend HTTP.
   - **`use_backend default_backend if is_https_request`**: Se a requisição for HTTPS, ela é direcionada para o `default_backend`.
3. **Backend**:
   - O **backend** é compartilhado entre ambos os frontends HTTP e HTTPS, com **persistência baseada na URL** usando o `stick-table` para garantir que as requisições para a mesma URL sejam sempre roteadas para o **mesmo servidor**.
   - **`cookie SERVERID insert indirect nocache`**: Garante a persistência usando cookies.
   - Os servidores backend continuam na porta **80** (HTTP), porque o HAProxy está fazendo a **terminação SSL**.

Como Funciona:

1. **Requisição HTTP**:
   - O cliente acessa a URL usando `http://meusite.com/pagina1`.
   - O HAProxy recebe a requisição na porta **80** e a encaminha para um dos servidores backend (`server1`, `server2`, `server3`), com base no balanceamento de carga `roundrobin`.
   - O servidor backend responde, e o HAProxy registra a URL `/pagina1` e o servidor que a atendeu.
2. **Requisição HTTPS**:
   - O cliente acessa a URL usando `https://meusite.com/pagina1`.
   - O HAProxy recebe a requisição na porta **443**, faz a descriptografia SSL e encaminha para o mesmo backend (porta 80), utilizando o mesmo servidor que atendeu a primeira requisição HTTP para aquela URL.
   - O HAProxy verifica a tabela de stickiness e direciona a requisição para o **mesmo servidor** que atendeu a primeira requisição, mantendo a persistência.
3. **Persistência para Todos os Usuários**:
   - A persistência é mantida independentemente de ser uma requisição HTTP ou HTTPS, porque a persistência está baseada na URL (`stick on url_param(url)`), não em cookies ou IP do cliente.
   - Todos os usuários que acessarem a mesma URL (seja por HTTP ou HTTPS) serão direcionados para o **mesmo servidor backend** que atendeu a primeira requisição para aquela URL.

Certificado SSL:

O caminho `/etc/ssl/certs/meucertificado.pem` deve apontar para o **certificado SSL** e a **chave privada** do seu domínio. Se você estiver usando o Let's Encrypt ou outra autoridade certificadora (CA), basta colocar o arquivo `.pem` correto que contenha o **certificado público** e a **chave privada**.

Caso você precise gerar um certificado SSL autoassinado, pode usar o seguinte comando `openssl`:

bash
Copiar
```
openssl req -x509 -newkey rsa:2048 -keyout /etc/ssl/private/meuchave.key -out /etc/ssl/certs/meucertificado.pem -days 365
```
Esse comando gera um certificado **autoassinado**, o que é útil para testes ou ambientes internos, mas para produção, recomenda-se o uso de um certificado de uma autoridade certificadora (CA) confiável.

Conclusão:

- **Sim, o HAProxy pode atender tanto HTTP quanto HTTPS ao mesmo tempo**, com a mesma configuração de persistência por URL para ambos os protocolos.
- A configuração utiliza dois **frontends** separados (um para HTTP e outro para HTTPS), mas ambos encaminham as requisições para o **mesmo backend**, garantindo que a persistência de URL seja aplicada de forma consistente, independentemente do protocolo.
- A **terminação SSL** acontece no HAProxy, o que significa que ele descriptografa as requisições HTTPS e as envia como HTTP para os servidores backend.

Se você deseja configurar o HAProxy para rotear tráfego em **portas diferentes de 80 (HTTP)** e **443 (HTTPS)**, o procedimento é semelhante, mas você deve ajustar a configuração de **frontends** e **backends** para escutar nas portas específicas que deseja utilizar.

A lógica de balanceamento de carga e persistência por URL será a mesma, mas em vez de usar as portas padrão (80 e 443), você configurará o HAProxy para escutar nas portas que você escolher.

Exemplo de Configuração para Portas Diferentes:

Vamos imaginar que você deseja usar as portas **8080** para HTTP e **8443** para HTTPS, em vez das portas padrão (80 e 443). A configuração seria:

```
global
    log /dev/log local0
    log /dev/log local1 notice
    maxconn 2000

defaults
    log     global
    option  httplog
    option  dontlognull
    timeout connect 5000ms
    timeout client  50000ms
    timeout server  50000ms

# Frontend para HTTP (porta 8080)
frontend http_front
    bind *:8080  # Escutando na porta 8080
    acl is_http_request path_reg ^/.*  # Captura qualquer URL
    use_backend default_backend if is_http_request

# Frontend para HTTPS (porta 8443)
frontend https_front
    bind *:8443 ssl crt /etc/ssl/certs/meucertificado.pem  # Certificado SSL
    acl is_https_request path_reg ^/.*  # Captura qualquer URL
```

```
    use_backend default_backend if is_https_request

# Backend
backend default_backend
    balance roundrobin
    stick-table type string size 200k expire 30m store gpc0
    stick on url_param(url)  # Persistência baseada na URL
    cookie SERVERID insert indirect nocache
    server server1 192.168.1.1:80 check  # Servidor HTTP
    server server2 192.168.1.2:80 check  # Servidor HTTP
    server server3 192.168.1.3:80 check  # Servidor HTTP
```

Explicação das Modificações:

1. **Frontend HTTP**:
   - **`bind *:8080`**: O HAProxy agora escuta na porta `8080` para tráfego HTTP, em vez de `80`.
   - **`acl is_http_request path_reg ^/.*`**: A regra ACL captura qualquer URL que for solicitada e a redireciona para o backend.
2. **Frontend HTTPS**:
   - **`bind *:8443 ssl crt /etc/ssl/certs/meucertificado.pem`**: O HAProxy escuta na porta `8443` para tráfego HTTPS e usa o certificado SSL localizado no caminho `/etc/ssl/certs/meucertificado.pem`. O certificado SSL deve ser válido, seja autoassinado ou de uma autoridade certificadora.
3. **Backend**:
   - A configuração do **backend** permanece a mesma, com balanceamento de carga baseado em `roundrobin`, persistência por URL usando `stick-table`, e a inserção do cookie `SERVERID` para garantir a persistência.
4. **Servidores Backend**:
   - **`server server1 192.168.1.1:80 check`**: Os servidores backend continuam sendo configurados na porta `80`, porque o HAProxy irá fazer a **terminação SSL** para o tráfego HTTPS e, em seguida, encaminhá-lo para os servidores backend na porta `80`.

Fluxo de Funcionamento:

1. **Requisição HTTP (porta 8080)**:

- O cliente acessa a URL `http://meusite.com:8080/pagina1`.
- O HAProxy recebe a requisição na porta `8080`, e com base no balanceamento de carga (roundrobin), encaminha a requisição para um dos servidores backend.
- O servidor que responder à primeira requisição para `/pagina1` será registrado no **stick-table** com a URL `/pagina1`, garantindo que futuras requisições para essa URL sejam

roteadas para o **mesmo servidor**.

2. **Requisição HTTPS (porta 8443)**:

- O cliente acessa a URL `https://meusite.com:8443/pagina1`.
- O HAProxy recebe a requisição na porta `8443`, faz a **terminação SSL** (descriptografa o tráfego HTTPS) e, em seguida, redireciona a requisição para os servidores backend na porta `80`.
- O HAProxy verifica na **stick-table** qual servidor foi associado à URL `/pagina1` e direciona a requisição para o **mesmo servidor** que atendeu a primeira requisição.

3. **Persistência para Todos os Usuários**:

- Independentemente de ser uma requisição HTTP ou HTTPS, a persistência será mantida para todos os usuários com base na URL, garantindo que a URL `/pagina1` seja sempre tratada pelo **mesmo servidor**.

Caso Você Queira Usar Portas Diferentes para Cada Backend:

Se você quiser que o tráfego seja roteado para diferentes portas nos servidores backend, você pode ajustar a configuração para cada servidor backend. Por exemplo, se você tiver servidores que escutam em portas diferentes, a configuração seria algo como:

```
backend default_backend
    balance roundrobin
    stick-table type string size 200k expire 30m store gpc0
    stick on url_param(url)
    cookie SERVERID insert indirect nocache
    server server1 192.168.1.1:8081 check  # Servidor backend escutando na porta 8081
    server server2 192.168.1.2:8082 check  # Servidor backend escutando na porta 8082
    server server3 192.168.1.3:8083 check  # Servidor backend escutando na porta 8083
```

Essa configuração faria com que cada servidor backend estivesse escutando em portas diferentes, mas o HAProxy ainda aplicaria a persistência e o balanceamento de carga.

Resumo:

- **Sim, é possível configurar o HAProxy para escutar em portas diferentes de 80 e 443** (por exemplo, 8080 para HTTP e 8443 para HTTPS).
- A persistência de URL e o balanceamento de carga continuam funcionando da mesma forma, **independentemente da porta** que está sendo utilizada.

- A configuração de **terminação SSL** no HAProxy permite que o tráfego HTTPS seja manipulado na porta `8443` e redirecionado para os servidores backend na porta `80` (ou qualquer porta configurada).