# HAProxy : le contrôleur d'ingress se détache des clusters

Comment gagner en latence sur un cluster Kubernetes ? Par exemple, en externalisant le contrôleur d'ingress. C'est désormais faisable avec celui associé à l'équilibreur de charge HAProxy.

Sa dernière version permet aussi d'activer l'**authentification basique** sur HTTP. On ajoute pour cela l'annotation auth-type avec la valeur basic-auth dans la définition Ingress ou dans le ConfigMap.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: web-ingress
  namespace: default
  annotations:
    haproxy.org/ssl-redirect: "true"
    haproxy.org/ssl-redirect-code: "301"
    haproxy.org/ssl-certificate: "default/tls-secret"
    haproxy.org/auth-type: basic-auth
    haproxy.org/auth-secret: "default/logins"
# ... other ingress settings...
```

Autre forme d'authentification qui devient activable : l'**authentification TLS mutuelle** entre le contrôleur (client) et les *back-end* (serveur). Les annotations server-ca et server-crt remplissent ce rôle. On peut les intégrer dans les définitions de services.

```
apiVersion: v1
kind: Service
metadata:
  labels:
    run: web
  name: web
  annotations:
    haproxy.org/server-ca: "default/server-tls-secret"
    haproxy.org/server-crt: "default/client-tls-secret"
# ... other service settings...
```

Les annotations donnent aussi la possibilité de paramétrer HAProxy sans avoir à éditer son fichier de configuration – avec son « langage » propre. Cette couche d'abstraction ne donne cependant pas accès à toutes les capacités de l'équilibreur de charge. La dernière version du contrôleur propose un palliatif : l'intégration de directives « natives » dans le ConfigMap comme dans les définitions d'ingress ou de services.

Le code ci-dessous illustre cette approche. Il stocke l'IP du client et l'URL demandée, puis applique un nombre limite de requêtes. Autant de fonctionnalités non exposées à travers les annotations.

```
apiVersion: v1
kind: Service
metadata:
  labels:
    run: web
  name: web
  annotations:
    haproxy.org/backend-config-snippet: |
      stick-table type binary size 1000 store http_req_rate(5s)
      http-request track-sc0 url32+src
      http-request deny if {  url32+src,table_http_req_rate() gt 50 }
# ... other service settings...
```

On soulignera aussi qu'il est maintenant possible de « donner corps » aux erreurs que génère HAProxy. Ce en créant des ressources ConfigMap qui associent des messages HTML à des codes HTTP.

```
apiVersion: v1
  kind: ConfigMap
  metadata:
    name: customerrors
    namespace: default
  data:
    503: |-
      HTTP/1.0 503 Service Unavailable
      Cache-Control: no-cache
      Connection: close
      Content-Type: text/html

      <html><body><h1>Oops, that's embarassing!</h1>
      <p>There are no servers available to handle your request.</p>
      </body></html>
```

*Illustration principale © Macrovector – shutterstock.com*