Google Cloud

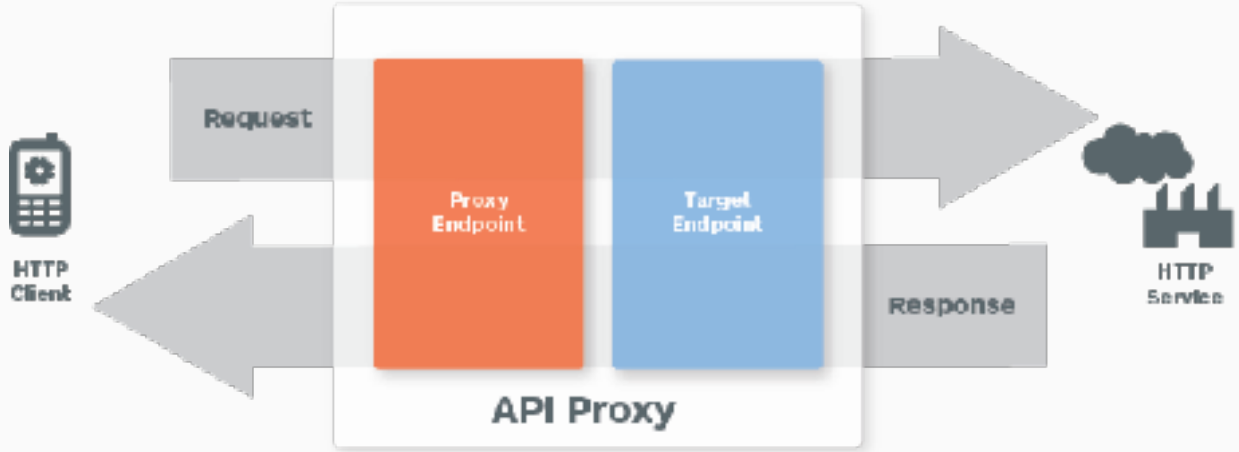
Edge Fundamentals

Named Target Servers

Target Servers: Hard Coded vs Parameterized

In the standard flow, target servers are hard coded

- **Problem:** When you promote your bundle from Dev to Prod, you have to edit the Target Endpoint URL manually
- **Solution:** Use Named Target Servers and set targets by environment



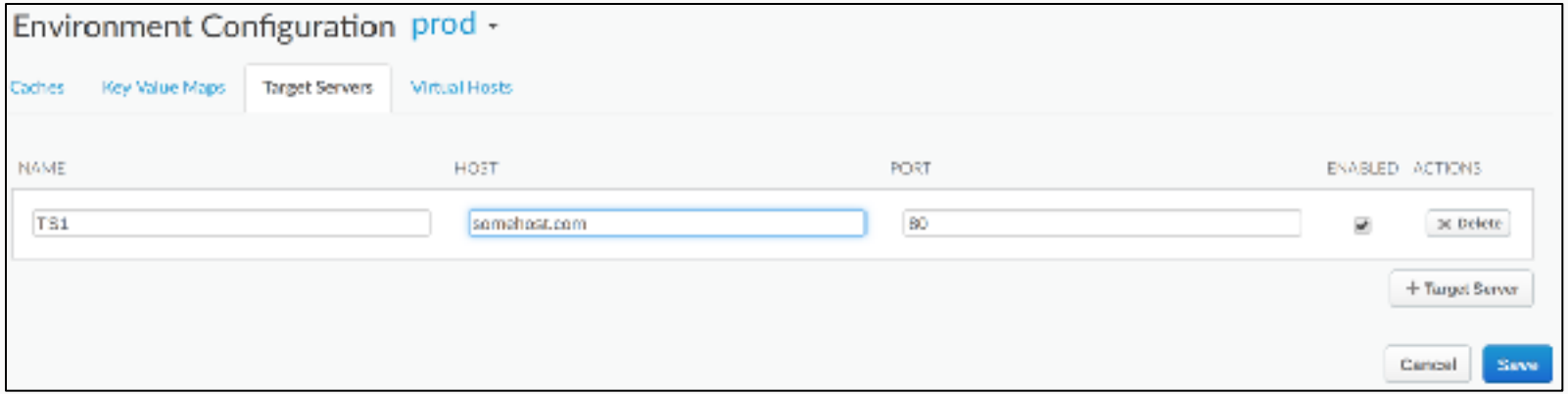
Existing Target URL

```
<TargetEndpoint name="default">
          <Description/>
          <PreFlow name="PreFlow">
                    <Request>
                    </Request>
                    <Response/>
          </PreFlow>
          <Flows/>
          <PostFlow name="PostFlow”/>
    <HTTPTargetConnection>
                    <URL>http://hardcoded-URL</URL>
          </HTTPTargetConnection>
</TargetEndpoint>
```

targets/default.xml

Create a Named Target Server

- TargetServers are used to decouple Target Endpoint HTTPTargetConnections from concrete URLs for backend services
- Target Endpoints can be parameterized for each environment (in "test" TS1 points to a somehost-test.com; and in prod to somehost.com)
- When you promote the proxy with the configurations across different environments, the same code will work seamlessly without any code change



Load balancer

- Edge enhances the availability of your API by providing built-in support for load balancing and failover across multiple backend server instances
- The load balancer supports three load balancing algorithms:
  - Round Robin (default)
  - Weighted
  - Least Connection
- Other configurations that can be used:
  - MaxFailures
  - RetryEnabled
  - IsFallback

```
<TargetEndpoint name="default">
   <Description>My Target</
Description>
   <HTTPTargetConnection>
        <LoadBalancer>
           <Server name="TS1" />
        </LoadBalancer>
        <Path>/v1/path</Path>
        <Properties/>
    </HTTPTargetConnection>
</TargetEndpoint>
```

Target Server with SSL

Target servers with SSL can be configured using the Management API

```
<TargetServer  name="TargetServer 1">
    <IsEnabled>true</IsEnabled>
    <Host>www.example.com</Host>
    <Port>443</Port>
    <SSLInfo>
        <Ciphers/>
        <ClientAuthEnabled>true</ClientAuthEnabled>
        <Enabled>true</Enabled>
        <IgnoreValidationErrors>false</IgnoreValidationErrors>
        <KeyAlias>keystore-alias</KeyAlias>
        <KeyStore>keystore-name</KeyStore>
        <Protocols/>
        <TrustStore>truststore-name</TrustStore>
    </SSLInfo>
</TargetServer>
```

Health Monitoring

- Health monitoring enables you to enhance load balancing configurations by actively polling the backend service URLs defined in the TargetServer configurations
- Two types of health monitors
  - TCP - a TCP client simply ensures that a socket can be opened
  - HTTP - configure the HTTP client to submit a valid HTTP request to the backend service
- Health monitors come into play when a backend server becomes unreachable
- In a traditional load balancer configuration if a server becomes unreachable it will drop it from the pool of servers
- The health monitor is designed to re-add those servers to the pool

Health Monitor - Types

TCP Monitor Example

```
<HealthMonitor>
    <IsEnabled>true</IsEnabled>
    <IntervalInSec>5</IntervalInSec>
    <TCPMonitor>
<ConnectTimeoutInSec>10</
ConnectTimeoutInSec>
      <Port>80</Port>
    </TCPMonitor>
  </HealthMonitor>
```

HTTP Monitor Example

```
<HTTPMonitor>
    <Request>
      <ConnectTimeoutInSec>10</ConnectTimeoutInSec>
     <SocketReadTimeoutInSec>30</
      SocketReadTimeoutInSec>
      <Port>80</Port>
      <Verb>GET</Verb>
      <Path>/healthcheck</Path>
      <Headers>
        <Header name="Authorization">Basic
12e98yfw87etf</Header>
      </Headers>
    </Request>
    <SuccessResponse>
      <ResponseCode>200</ResponseCode>
      <Headers>
        <Header name=”ImOK”>YoureOK</Header>
      </Headers>
    </SuccessResponse>
  </HTTPMonitor>
```

Example Healthchecks

```
<HTTPMonitor>
    <Request>
      <ConnectTimeoutInSec>10</ConnectTimeoutInSec>
      <SocketReadTimeoutInSec>30</
SocketReadTimeoutInSec>
      <Port>80</Port>
      <Verb>GET</Verb>
      <Path>/healthcheck</Path>
      <Headers>
        <Header name="Authorization">Basic
12e98yfw87etf</Header>
      </Headers>
    </Request>
    <SuccessResponse>
      <ResponseCode>200</ResponseCode>
      <Headers>
        <Header name=”ImOK”>YoureOK</Header>
      </Headers>
    </SuccessResponse>
  </HTTPMonitor>
```

HTTP Monitor Example

```
<HealthMonitor>
    <IsEnabled>true</IsEnabled>
    <IntervalInSec>5</IntervalInSec>
    <TCPMonitor>
<ConnectTimeoutInSec>10</
ConnectTimeoutInSec>
      <Port>80</Port>
    </TCPMonitor>
  </HealthMonitor>
```

TCP Monitor Example

Lab

Please follow the steps provided here

THANK YOU