Roadmap To a Reliable Mission Critical Proxy

A case study

The Background

A Fintech client had 3 outages in the last 3 months, bringing their SLA to 98.4% which was just below the required 98.5%.

The management is highly worried that not only will they have to pay penalties, but they may also lose clients because of this.

They invite me to have a look at their system and create a roadmap for replacement.

Technical Overview

- The environment consists of :
  - 2 on-prem data centers in Vancouver and Toronto, which are leased
  - AWS
  - Azure
  - The plan was to migrate all remaining on-prem systems to Azure by the end of the year and to migrate all AWS assets to Azure next year
- 10,000 virtual machines spread over the 3 environments
- Dozens of VPN connections to clients' "host machines", all on-prem
- Dozens and dozens of technologies in use: VMWare, Windows, Linux, MSSQL, MuleSoft, AWS API Gateway, Apache, Nginx, Tomcat, Jetty, Azure Service Bus, JBoss, Docker, Kubernetes, etc.
- All these pieces are connected and supported by a single-point-of-failure system that was developed in-house more than 20 years ago.
- The system's purpose:
  - A rear-end proxy for all egress calls to the Internet from the network
  - Load-balancer for most intra-network calls among the systems as well as egress calls
  - Firewall between ingress and egress traffic
  - Health-check for defined critical internal and external systems

The Method

1. Define the problem (see above)
2. Analyze:
   1. The environment
   2. The system
   3. The failures
3. Look at the analysis data:
   1. Draw conclusions
   2. Confirm conclusions with the stakeholders
4. Research solutions and alternatives
   1. Long-term solution
   2. Short-term tactical fix
5. Select the preferred solutions
6. Discuss them with the team
7. Create a roadmap for both solutions
8. Present the roadmap to the stakeholders

The Story

2.1

The architecture pattern involving the proxy is valid; one needs to proxy external calls and load-balance internal ones when there is a VPN constraint on the specific machines.

2.2

I read the documentation, checked the system configuration and consulted the dev and support teams to get as much insight as possible. The system was ancient (20+ years old), but it was well-designed and appears to be working well, especially in a clustered environment, because clustering is built in.

2.3

The failures that occurred, 2 were diagnosed, one wasn't. The diagnosed failures were not system failures but vendor failures; vendor connections either took too long or failed. In either case, the system simply ran out of configured connections; the maximum server connection count was set too low.

The support team handled the first failure by adding another server to the cluster, but that didn't prevent the second failure, where a major vendor went offline completely, causing the system to run out of connections again, even with the additional machine in the cluster.

3.1

My conclusion was that the system was :

1. Grossly misconfigured – the connection count was set too low, as it hasn't been increased in 20 years since the system went to production
2. Poorly supported – the support team was using the wrong metrics to monitor the system

The Story

3.2

I confirmed this with both teams, dev and support.

4.

I researched how many connections this type of web server can support and found that, in the current technical environment, a single machine can easily handle thousands of connections, which is about ten times more than the configuration.

But due to the age of the design of the system, I also concluded that the long-term solution was to replace the system with a modern alternative.

5.

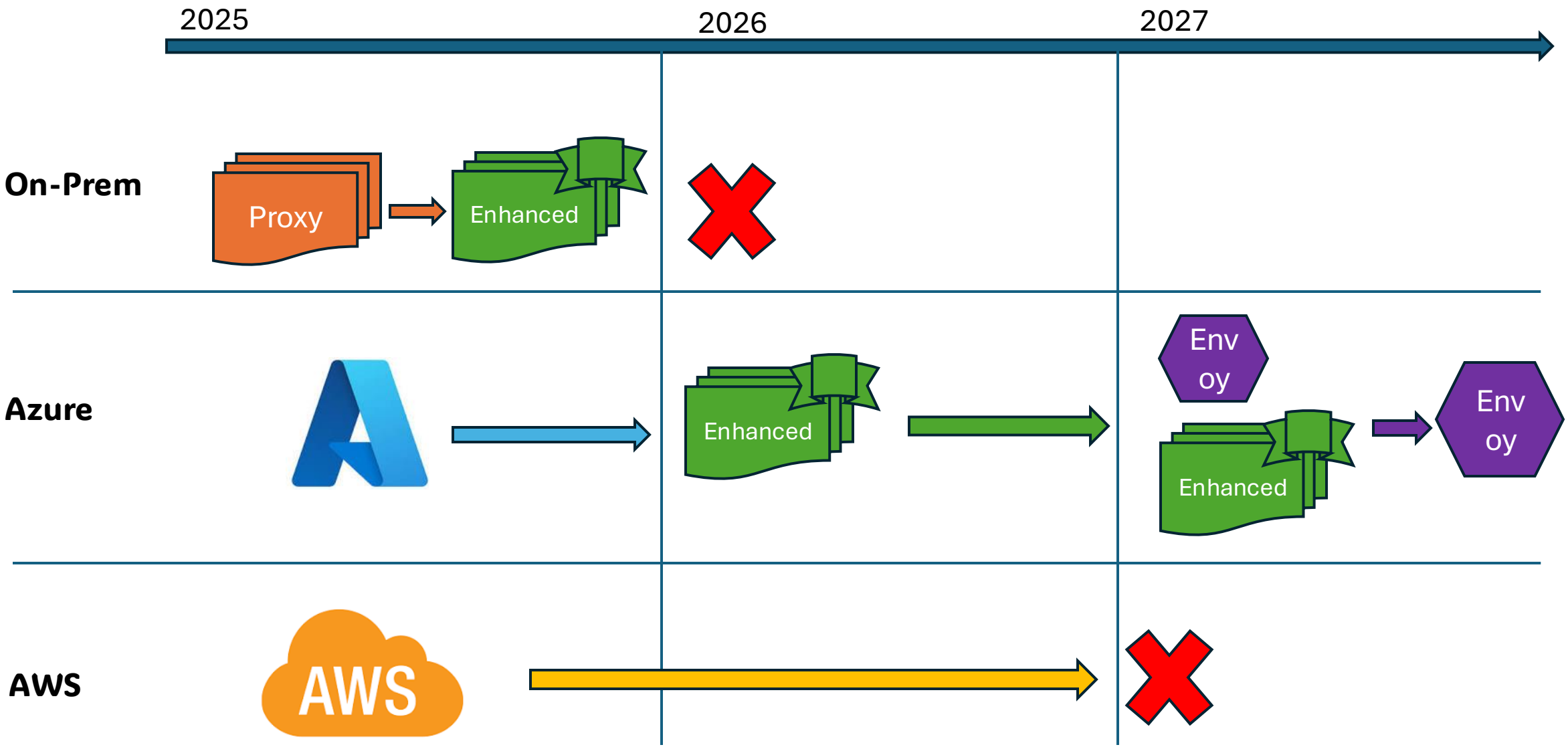
After doing thorough research, I selected Envoy as the replacement system based on a comprehensive feature comparison to other similar systems, such as HAProxy and Nginx.

5.1

I presented my conclusions: the replacement selection as Envoy, as well as interim steps to help stabilize the current system:

1) Significantly increase the connection configuration per machine
2) Change monitoring metrics to send a notification just before the system runs out of connections
3) Try and add a “CIRCUIT BREAKER” functionality to the system

The Roadmap


2025
2026
2027
On-Prem
Proxy
Enhanced
Azure
Enhanced
Env
oy
Enhanced
Env
oy
AWS
AWS

Summary

Conceptual

- The home-grown proxy is stable, reliable, and well-performing, but is grossly under-configured and improperly monitored
- There is no point in trying to replace it amid all the major environmental changes going on currently, especially due to its critical importance
- In the short and medium term, it's advisable to retain the system but improve the configuration and monitoring
- In the long term, the system will need to be replaced by a modern, open-source equivalent. The best candidate would be Envoy, but other systems are available as well, should Envoy, for some reason, fail a POC.

Roadmap description

- Enhance the proxy and continue its use in the on-prem environment
  - Since the circuit breaker is not working in the proxy, drop it; it’s optional
- Once on-prem assets are migrated to Azure, configure a cluster of Azure VMs to run the proxy
- Once AWS assets are fully migrated to Azure, configure Envoy and run it concurrently with the proxy
- Once the Envoy is fully configured and working properly, switch all the traffic to it