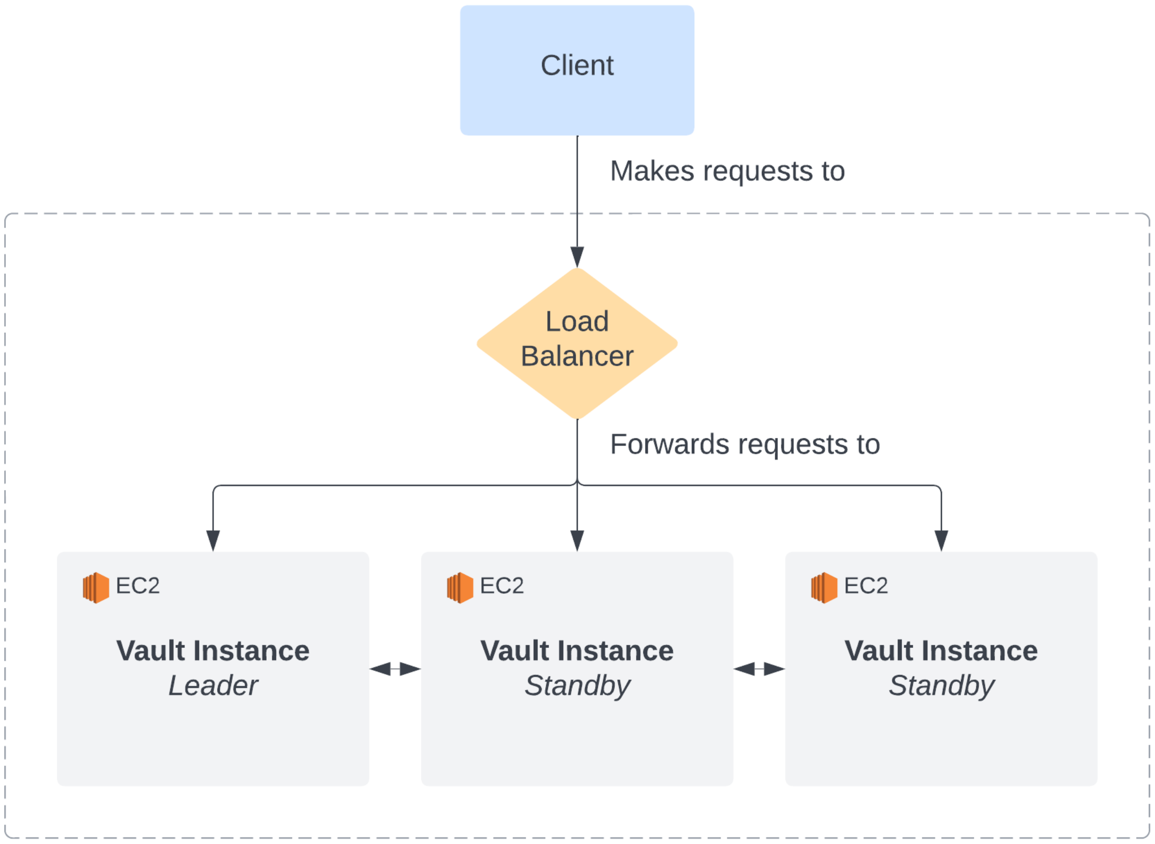
Context

The Foundation Secure team at Zendesk runs dozens of highly-available Hashicorp Vault clusters on AWS. Vault is an open source software that allows humans & machines to store and retrieve secret/sensitive data, as well as control/audit access to that data.

Examples include API keys, passwords, certificates, and more.

A Vault cluster is composed of a few core components:

1. **The leader node** – in charge of handling all traffic/requests
2. **The standby node** – ready to take leadership if the leader node fails
    a. If a standby node receives a request, it will forward it to the active leader node
3. **The load balancer** – an entrypoint for all clients, that distributes traffic evenly across nodes



At any one time, **only the leader node handles requests**. If a standby node receives a request, it simply forwards it to the leader node.

The purpose of having standby nodes is to ensure that one of them can take over at any time, should the leader node fail, or become unhealthy. Standby nodes are generally placed throughout multiple AWS Availability Zones for resiliency.

Problem Statement

SecretService 1.0 and 2.0 (Zvault) clusters regularly receive a high volume of traffic from many services fetching and rotating secrets. In the past, large bursts of traffic have overloaded our Vault clusters and caused outages. Unfortunately Vault open-source does not allow per-secret or per-namespace traffic throttling, and as a result we have limited ability to protect our Vault instances from these traffic bursts. Moreover, we lack visibility into which services are causing these bursts of traffic to our Vault clusters.

Solution requirements:

To address this issue, we would like to deploy a local service to our Vault hosts that may intercept incoming Vault requests and offer throttling and caching capabilities before requests are proxied through to Vault. Your solution must listen over HTTPS by adopting a generated self-signed certificate. AWS ALBs will accept this certificate by default.

Things to consider:

When considering a potential solutions to this problem, consider the following:

- What throttling algorithm is appropriate for this use case? I.E – should some services be granted greater throughput than others? How would that be implemented? Should throttling be configurable by path or namespace? If yes, how could we implement this?

- How will we ensure throttling state is shared between proxies? The load balancer will round-robin traffic to all healthy Vault instances. If a request to secret `foo` by service `bar` is proxied to Vault Node 1, but the next request will be passed through to Vault Node 2, how will Node 2 know about Node 1's request and take that request into throttling considerations?

- What requests can be cached and how can we prevent cache mining? Services should not be able to fetch cached secrets they don't have access to.

- Should caching be shared among proxies or is per-proxy caching acceptable?

- This service must be lightweight. Assume an available memory footprint of <100MB. Memory leaks must be avoided.
- How will errors be tracked & managed? If Vault returns a 500, what should the proxy do?

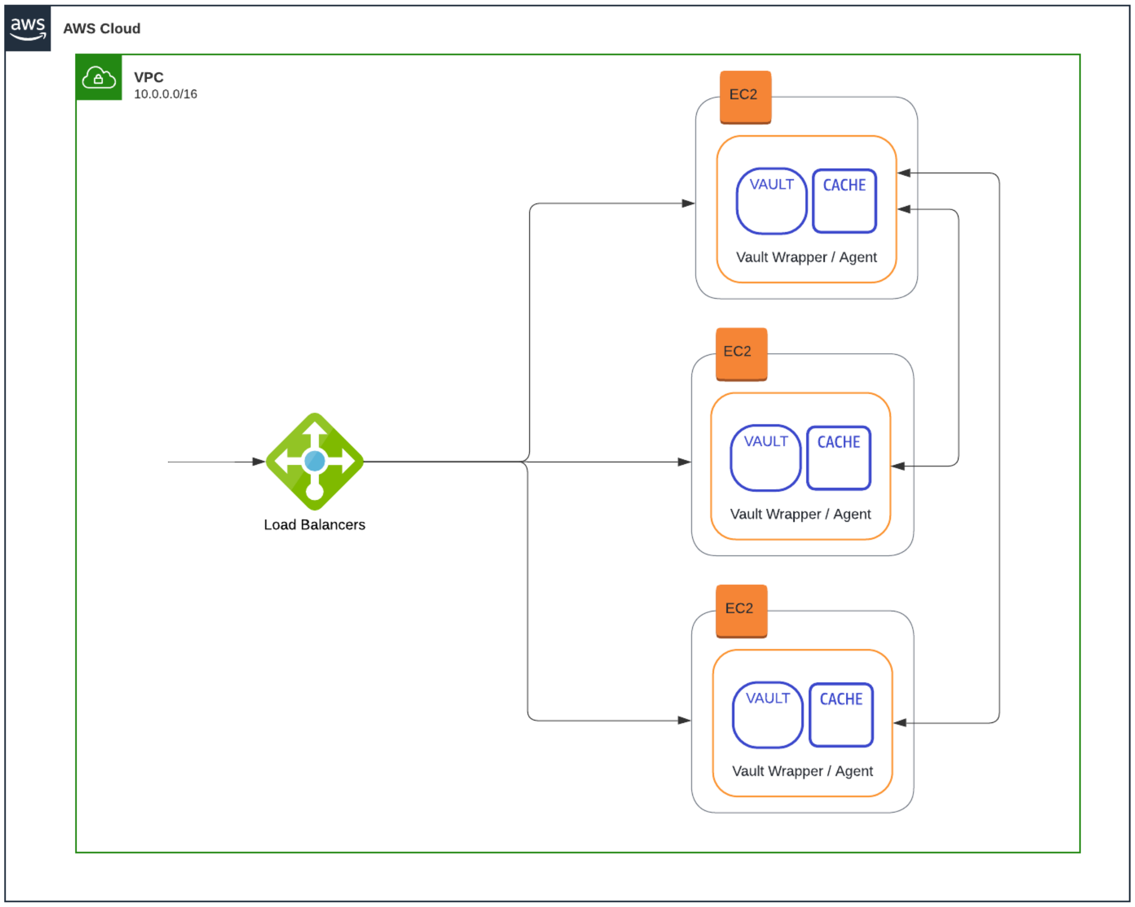
Links/Resources

GitHub Repositories

- vault-proxy: This repository can act as inspiration or a starting point for taking on this challenge. This contains a simple implementation of a local proxy with a caching mechanism to vault. This does not contain any throttling logic and may require updates depending on your planned solution.

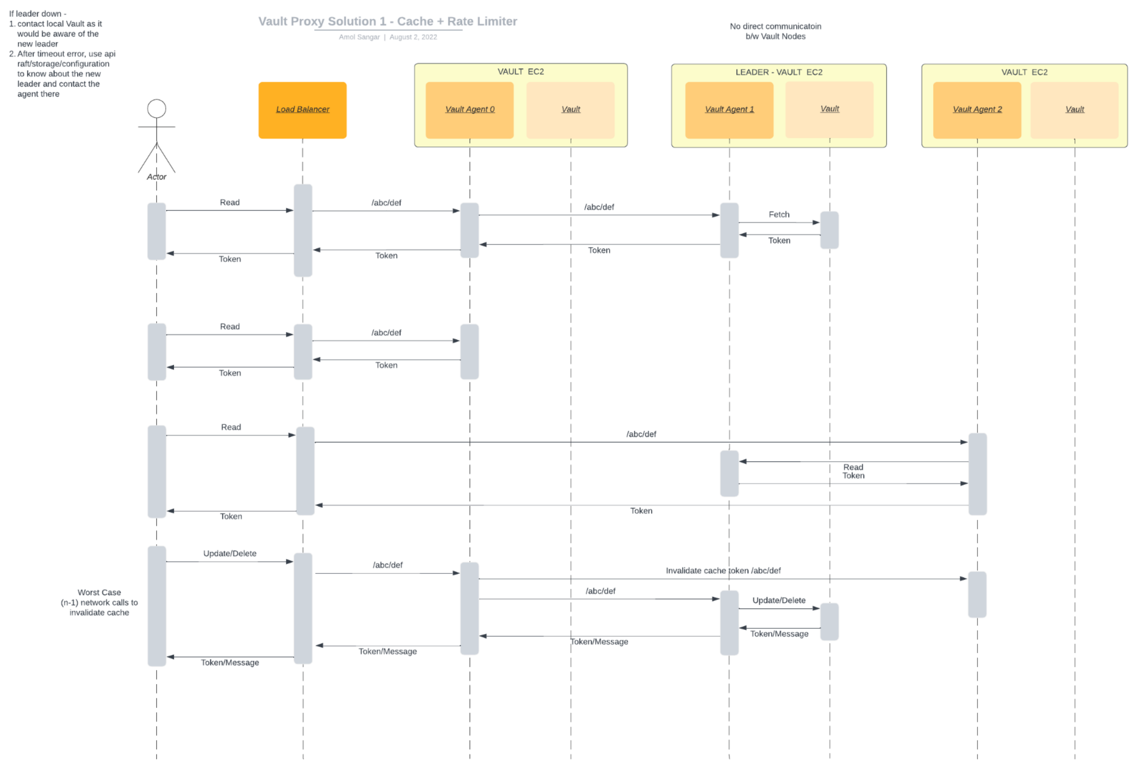
Solution Proposals

Architecture Diagram Solution 1



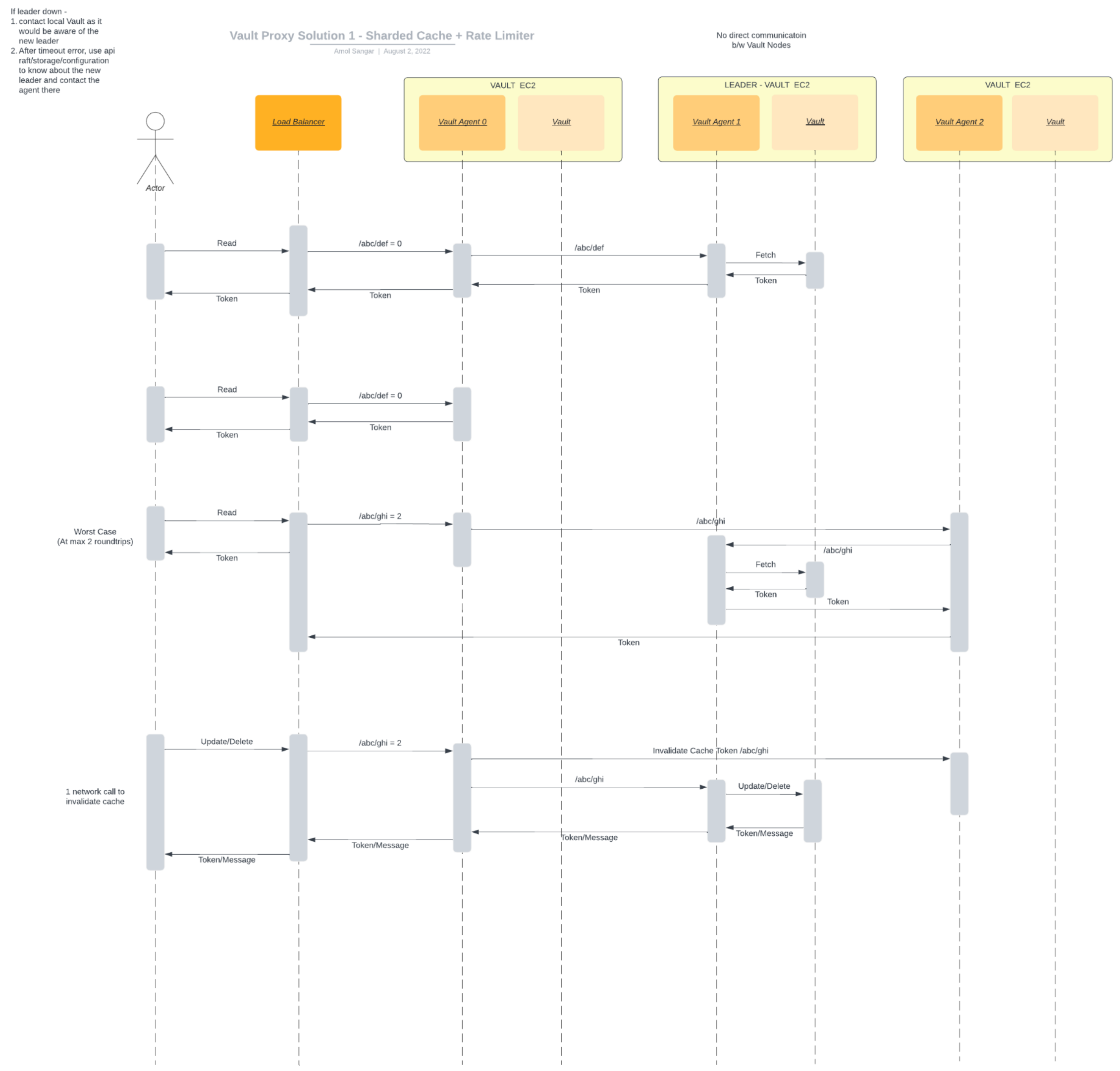
Sequence Diagram

Solution 1 - One Cache + RateLimiting

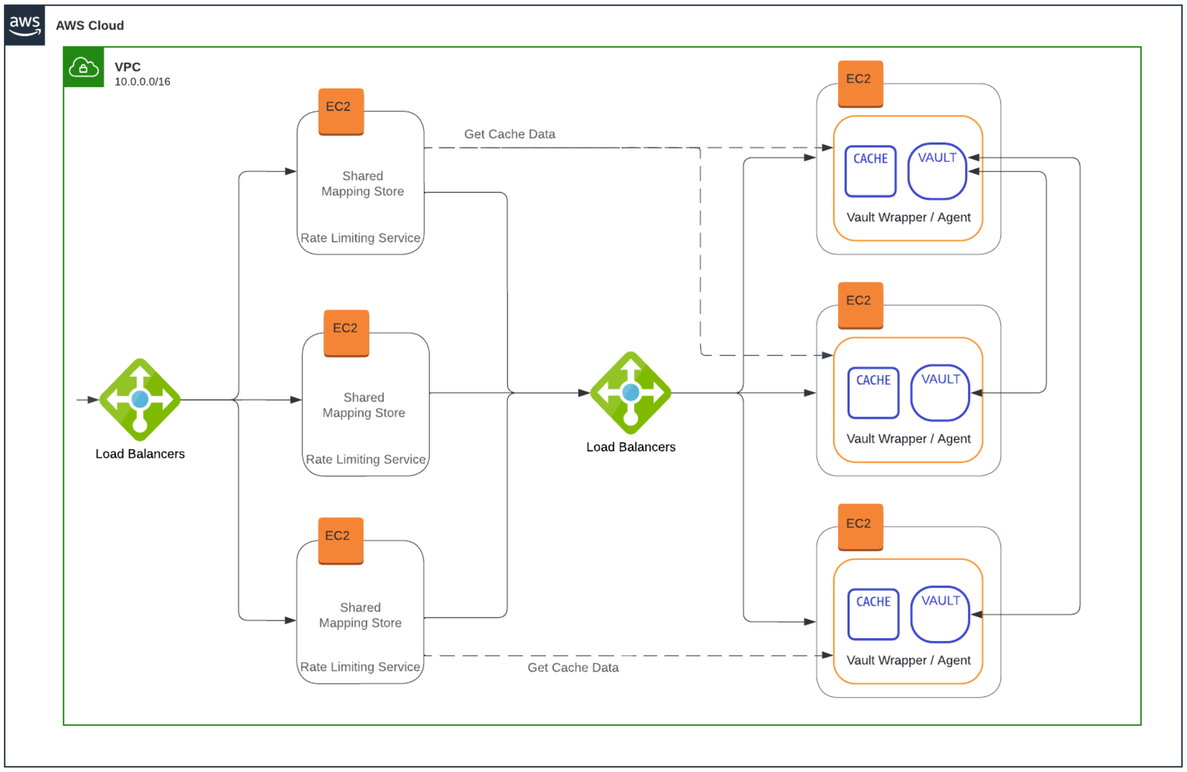

If leader down -
1. contact local Vault as it would be aware of the new leader
2. After timeout error, use api raft/storage/configuration to know about the new leader and contact the agent there
Vault Proxy Solution 1 - Cache + Rate Limiter
Amol Sangar | August 2, 2022
No direct communicatoin b/w Vault Nodes
VAULT EC2
LEADER - VAULT EC2
VAULT EC2
Load Balancer
Vault Agent 0
Vault
Vault Agent 1
Vault
Vault Agent 2
Vault
Actor
Read
/abc/def
/abc/def
Fetch
Token
Token
Token
Token
Read
/abc/def
Token
Token
Read
/abc/def
Read
Token
Token
Token
Update/Delete
/abc/def
Invalidate cache token /abc/def
/abc/def
Update/Delete
Token/Message
Token/Message
Token/Message
Token/Message
Worst Case
(n-1) network calls to invalidate cache

Sequence Diagram

Solution 1 v2 - Sharded cache + Rate Limiting

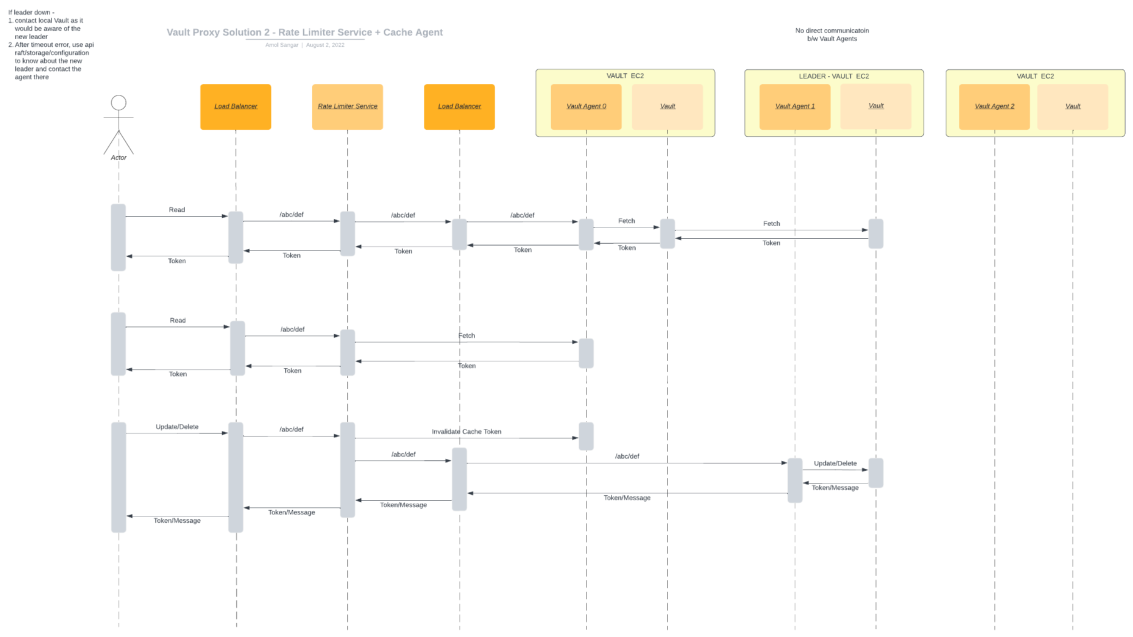


Architecture Diagram Solution 2



Sequence Diagram

Solution 2 - Rate Limiting Service + Cache Agent



If leader down -
1. contact local Vault as it would be aware of the new leader
2. After timeout error, use api raft/storage/configuration to know about the new leader and contact the agent there
Vault Proxy Solution 2 - Rate Limiter Service + Cache Agent
Amol Sangar | August 2, 2022
No direct communicatoin b/w Vault Agents
Actor
Load Balancer
Rate Limiter Service
Load Balancer
VAULT EC2
Vault Agent 0
Vault
LEADER - VAULT EC2
Vault Agent 1
Vault
VAULT EC2
Vault Agent 2
Vault
Read
/abc/def
/abc/def
/abc/def
Fetch
Fetch
Token
Token
Token
Token
Token
Token
Read
/abc/def
Fetch
Token
Token
Token
Update/Delete
/abc/def
Invalidate Cache Token
/abc/def
/abc/def
Update/Delete
Token/Message
Token/Message
Token/Message
Token/Message
Token/Message