BetterCloud

Modern Service Management with Consul and Vault

Amit Khare | Steve Perkins

INTRO

Intro

SPEAKERS

- Steve Perkins - Software Architect
- Amit Khare - Data Architect

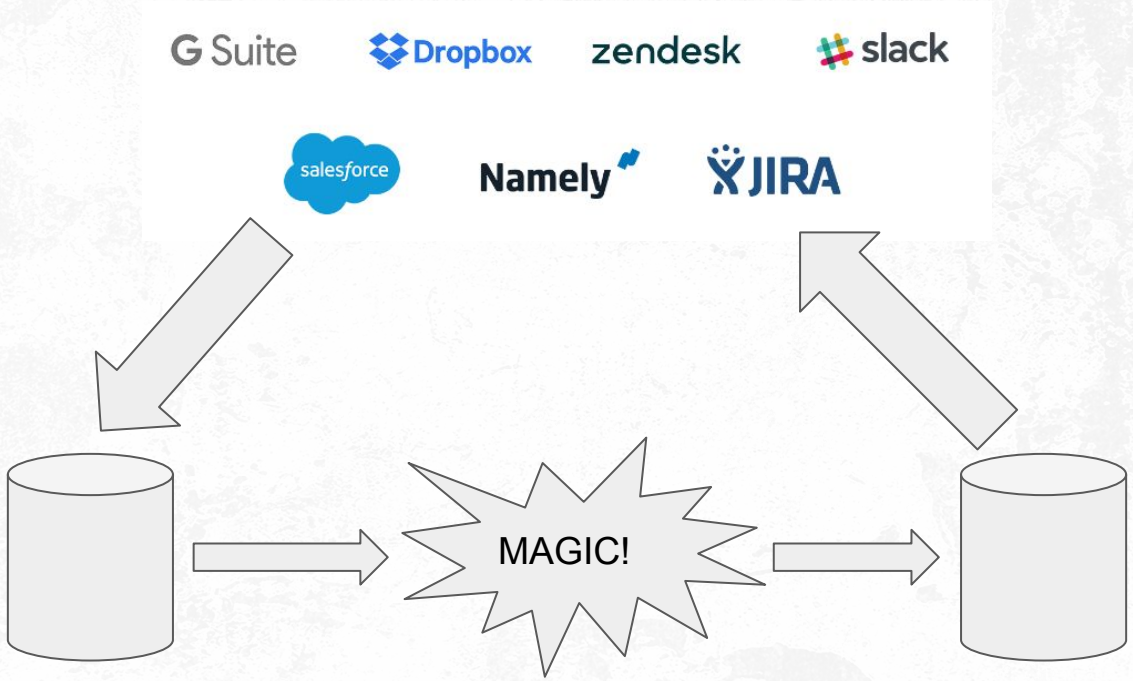
SCOPE

- Service discovery / load balancing / failover
- Configuration management
- Endpoint security and TLS

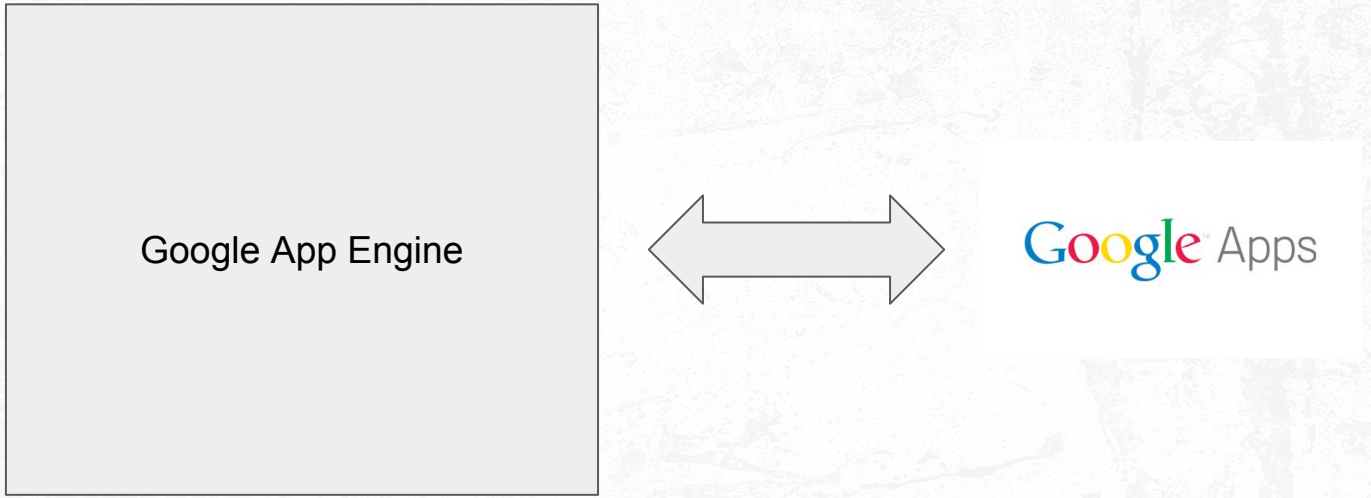
RESOURCES

- https://github.com/steve-perkins/DevNexus-2017

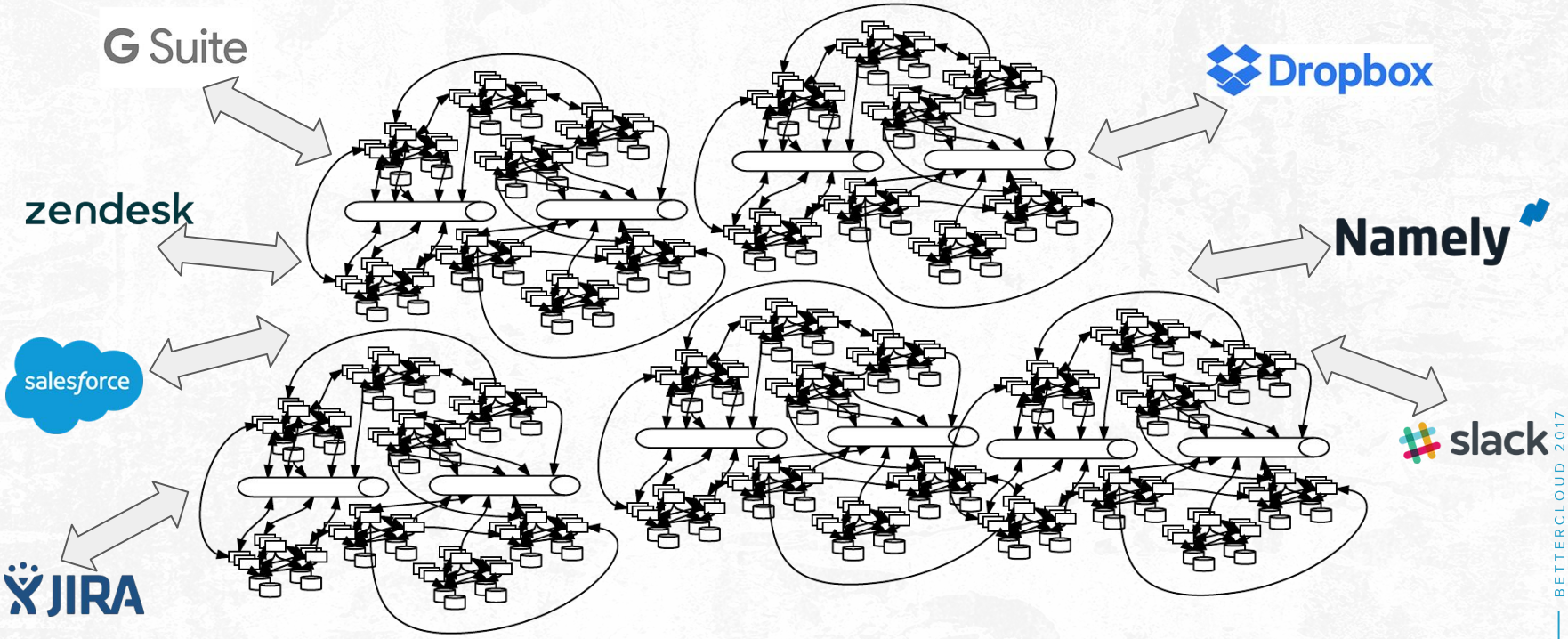
BetterCloud



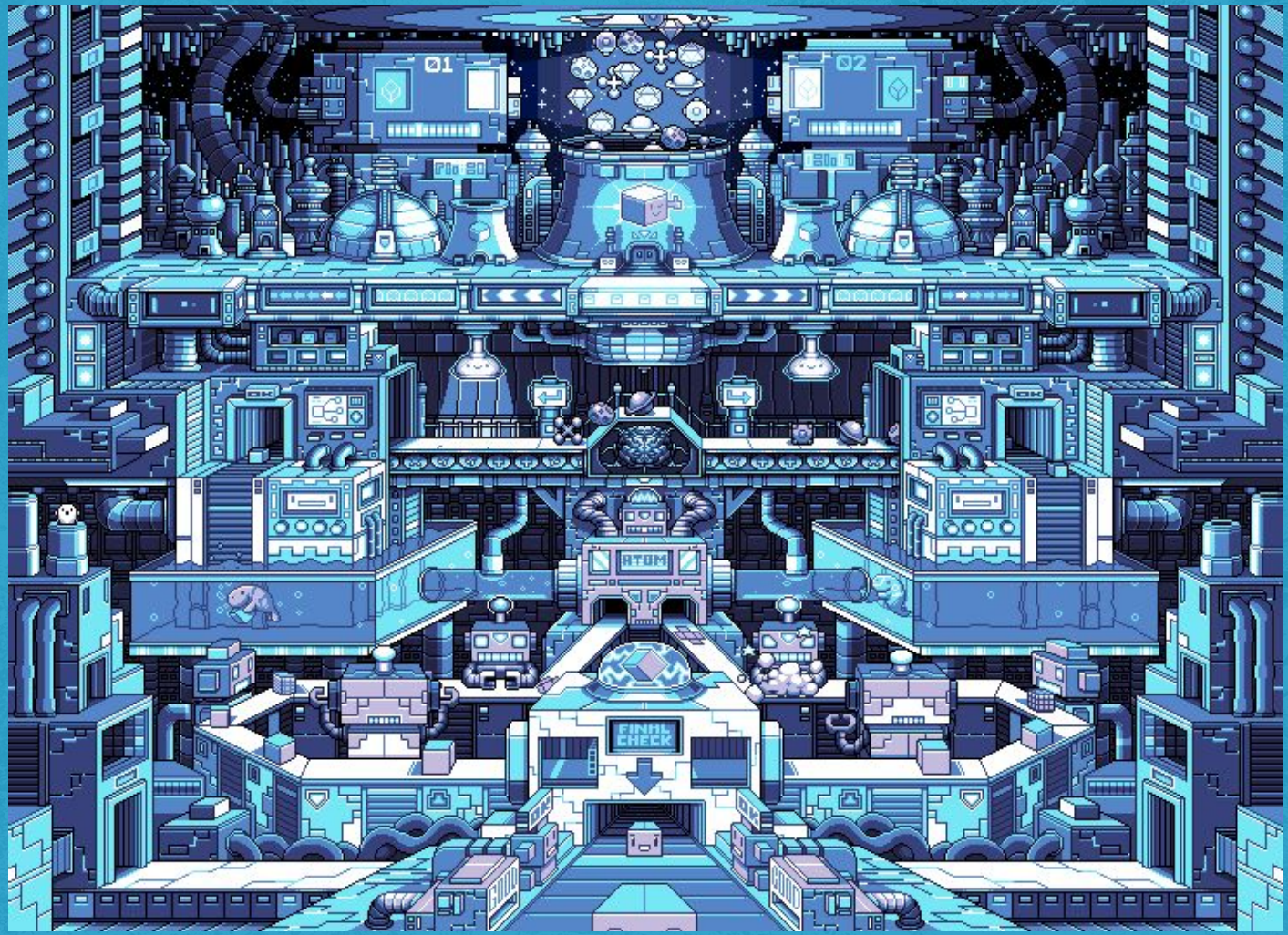
Early Architecture



Current Architecture(*)



(*) not really


01
02
ATOM
FINAL
CHECK

PROBLEM SPACE

Challenges Encountered

LOAD BALANCING / FAILOVER, AND SERVICE DISCOVERY

- How to scale and un-scale linearly without having downtime and dynamically change load balancer based on scaling and removing services?

CONFIG MANAGEMENT

- How to manage configuration, and dynamically change it across hundreds of services, without re-builds and re-deploys?

ENDPOINT SECURITY

- How to ensure that REST and MQ endpoints are protected from external attackers, and also from the risk of services mistakenly talking to the wrong endpoint?

TECHNOLOGIES

Consul and Vault

CONSUL

- Tool for Service Discovery and Configuration
- Distributed, highly available and extremely scalable
- Built-in health checks integrated with Service Discovery
- Key/Value Storage
- Datacenter Aware

VAULT

- Highly Secure Secret storage
- Dynamic Secrets
- Data Encryption
- Leasing and Renewal
- Revocation

SERVICE DISCOVERY, ETC

Service Discovery

Problem

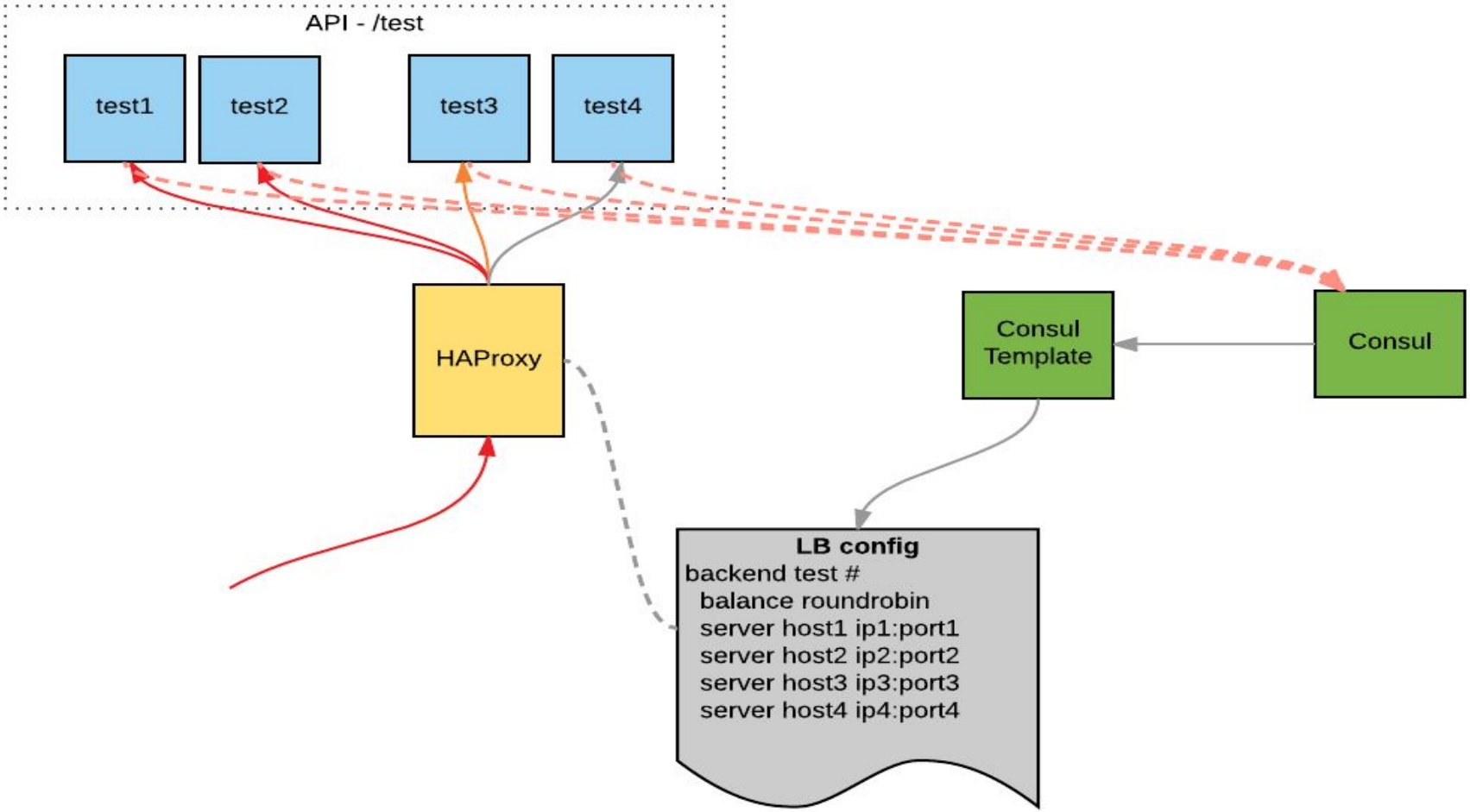
- With Microservice Architecture, we want to scale our infrastructure real-time

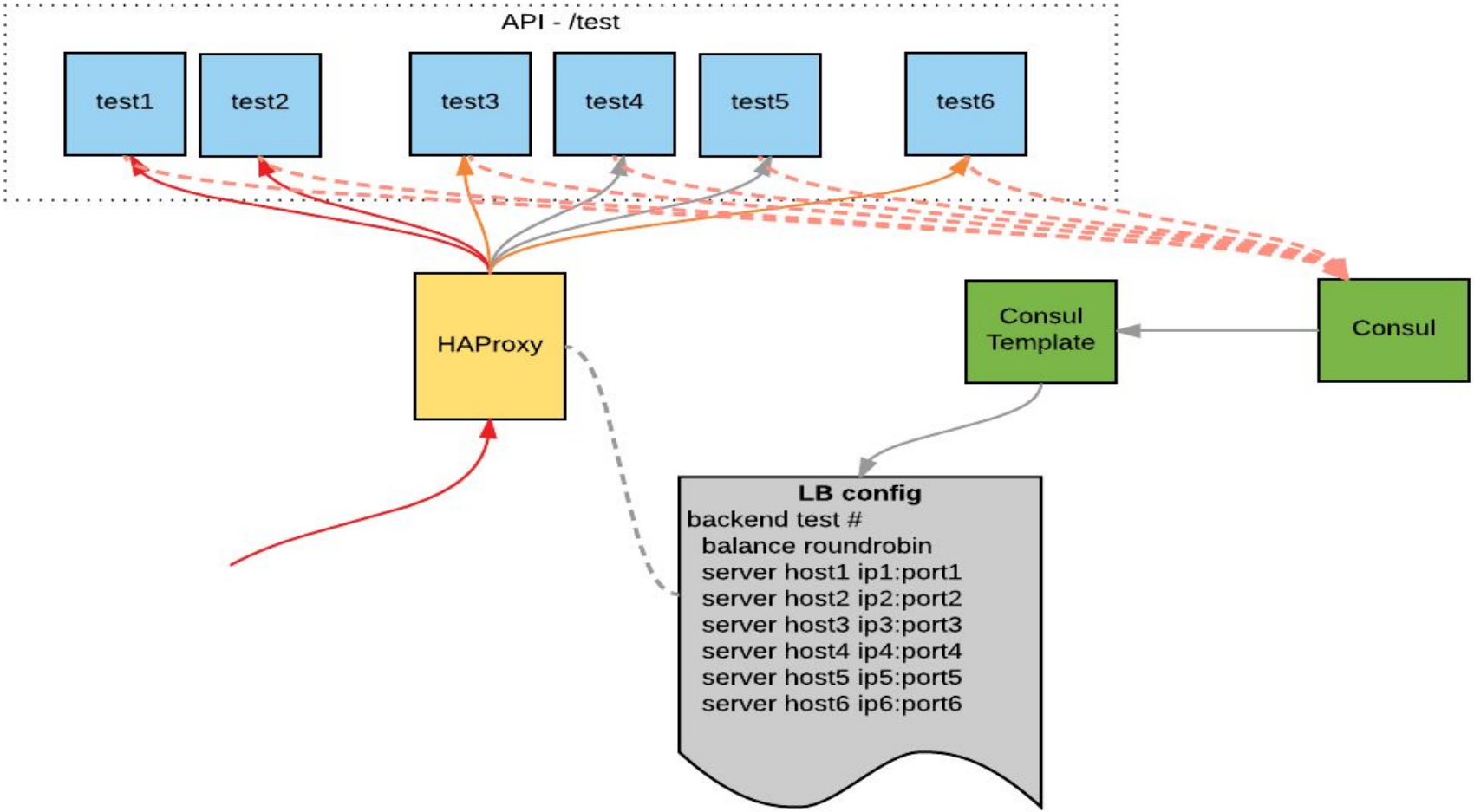
Options

- Use existing Nagios health checks and update LB configs with custom python scripts
- Use Service Discovery integrated with Health Checks and update LB configs - Consul

Solution

- Use Health Checks integrated with Consul Service Discovery
- Use Consul-Template to update LB Configs based on changes on Service Discovery


API - /test
test1
test2
test3
test4
HAProxy
Consul Template
Consul
LB config
backend test #
balance roundrobin
server host1 ip1:port1
server host2 ip2:port2
server host3 ip3:port3
server host4 ip4:port4


API - /test
test1
test2
test3
test4
test5
test6
HAProxy
Consul Template
Consul
LB config
backend test #
balance roundrobin
server host1 ip1:port1
server host2 ip2:port2
server host3 ip3:port3
server host4 ip4:port4
server host5 ip5:port5
server host6 ip6:port6

SERVICES | NODES | KEY/VALUE | ACL | [redacted] | 

assets-data | any status | EXPAND

[redacted]micro 256 passing

Consul Template

```
backend test #
  balance roundrobin{{range service "test@<dc>"}}
    server {{.Node}} {{.Address}}:{{.Port}}{{end}}
```

[redacted]micro

TAGS

production

NODES

[redacted] 2 passing

Serf Health Status serfHealth passing

Service '[redacted]-micro' check

[redacted] passing

[redacted] 2 passing

Serf Health Status serfHealth passing

Service '[redacted]-micro' check

[redacted] passing

[redacted] 2 passing

Serf Health Status serfHealth passing

CONFIGURATION / PROPERTIES MANAGEMENT

Traditional Approaches

PROPERTIES FILES

- Sometimes part of the build artifact, sometimes elsewhere on the filesystem
  - The former requires rebuilds and redeploys
  - The latter requires solutions for deployment and source control

- Usually no perfect ownership model
  - If devs own it, then they can see things they shouldn't
  - If ops/security, then it's painful to manage changes

- Can be cumbersome to manage differences across environments

Traditional Approaches

DATABASE TABLE

- Lots of advantages
  - There's usually an environment-specific DB anyway
  - Good for locking down access to "secret" values
  - Don't need to rebuild or redeploy when properties changes

- But still problems
  - Tracking change / source control
  - Facilitating change of "non-secret" values
  - Query DB every time, or only at startup?

Ideal Requirements

LARGE SCALE

- Able to manage configuration across hundreds of apps, each in multiple environments

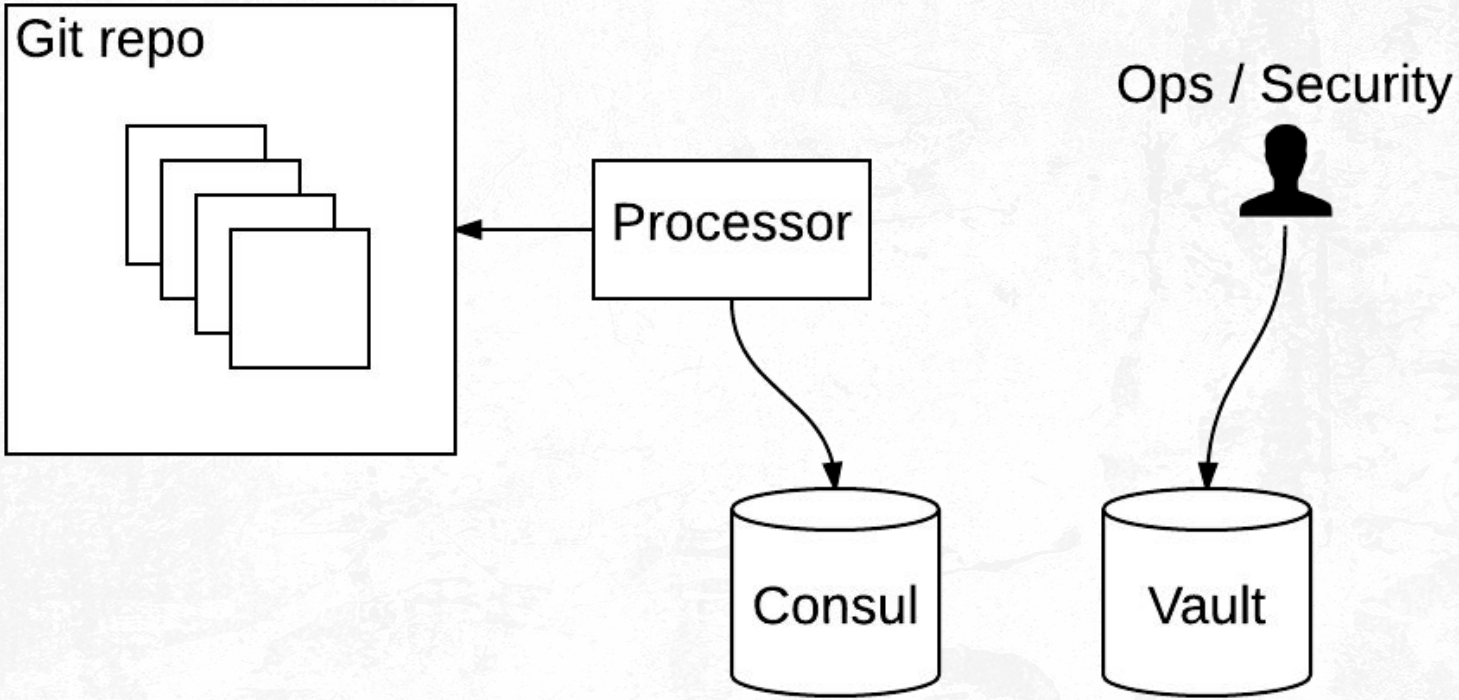
DYNAMIC

- No need to rebuild or redeploy apps when their configuration changes
- However, able to track (and revert) changes safely

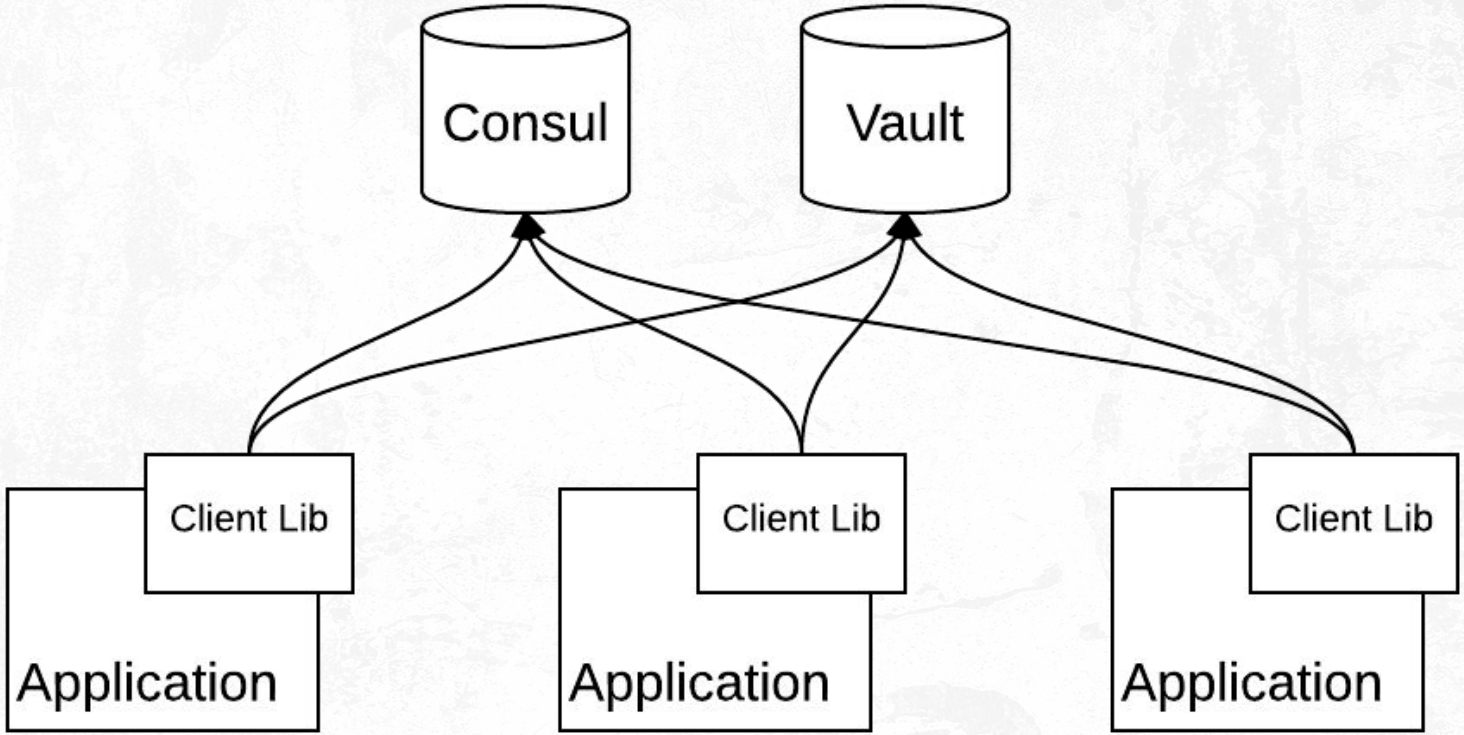
SEPARATION OF RESPONSIBILITIES

- Developers can maintain values where appropriate
- Security / Ops can lock down values where appropriate

Example #1: Framework-agnostic



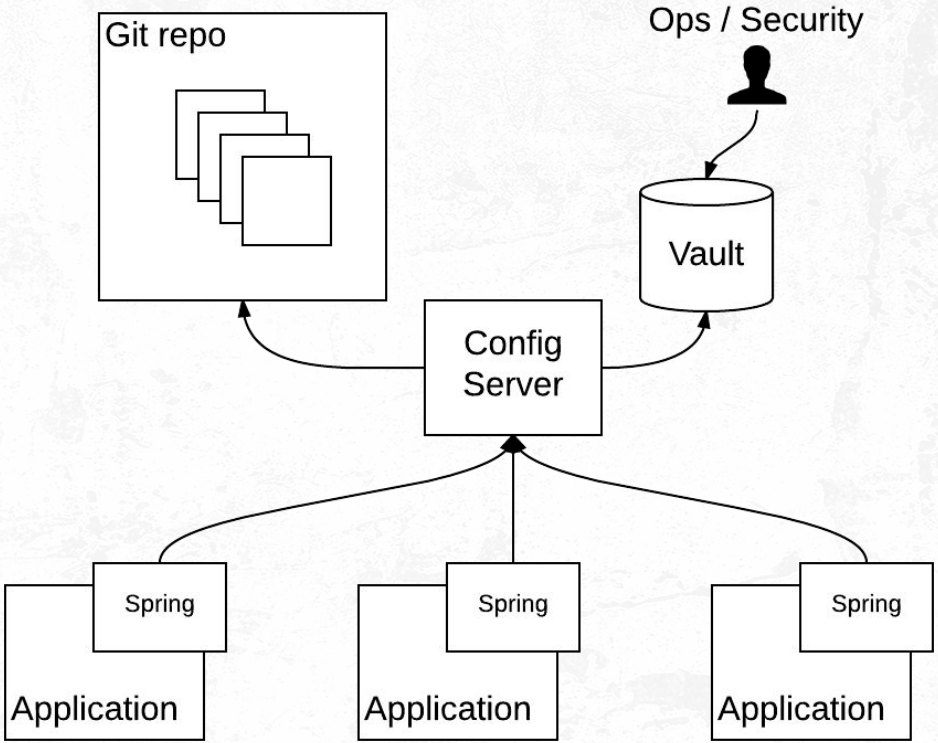
Example #1: Framework-agnostic



Example #1: Framework-agnostic

- https://github.com/steve-perkins/config-properties - the properties files and Processor component
- https://github.com/steve-perkins/config-client-lib - the shared library
- https://github.com/steve-perkins/config-sample-app - a demo application under config managment

Example #2: Spring Cloud Config



Example #2: Spring Cloud Config

- https://github.com/steve-perkins/spring-config-properties - the “non-secret” properties files
- https://github.com/steve-perkins/spring-config-server - the Spring Cloud Config Server instance, configured for Vault/Git composite
- https://github.com/steve-perkins/spring-config-sample-app - a demo application under config managment

SECURITY AND TLS

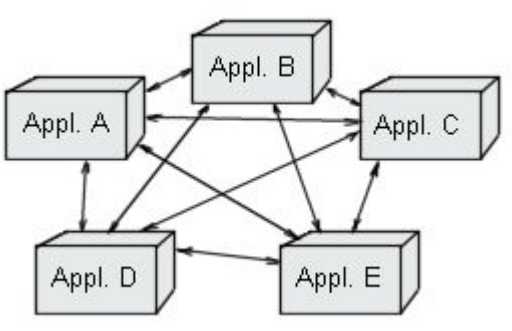
Microservices == More Internal Traffic

SEPARATING PROCESSES LEADS TO INCREASED IPC

- HTTP
- MQ

INCREASED IPC LEADS TO INCREASED RISK

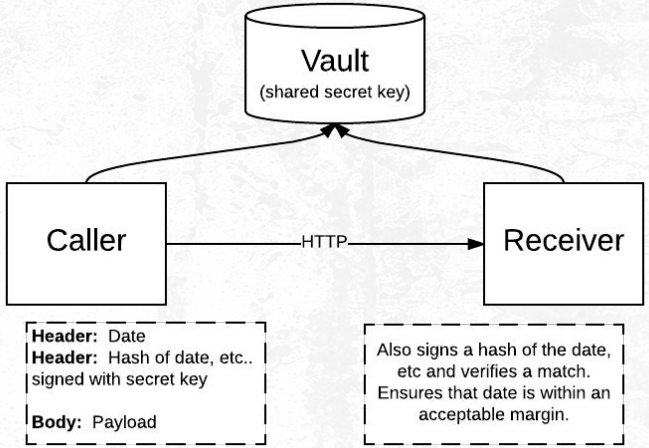
- External attackers
- Internal negligence



Secure HTTP Endpoints

HMAC

- Uses a shared secret key (which could be stored in Vault)
- Callers send a header, a hash of some values signed with the shared key
- Receivers perform the same hash and verify a match



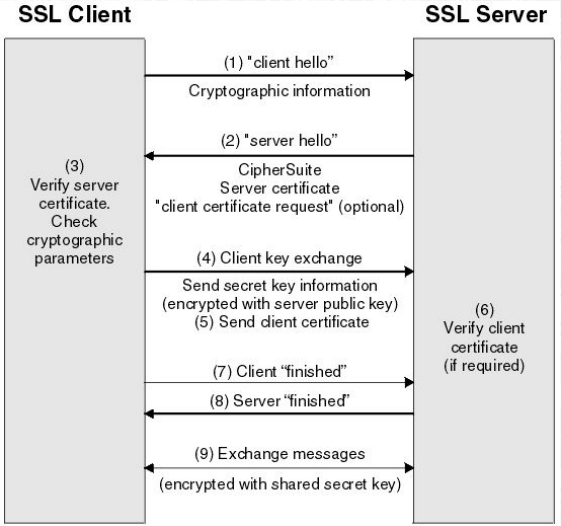
RESOURCES

- https://en.wikipedia.org/wiki/Hash-based_message_authentication_code
- Class **`javax.crypto.Mac`** in the JDK
- Apache Commons Codec

Secure MQ (and other infrastructure)

TRANSPORT-LAYER SECURITY (TLS)

- Basic versus client authentication



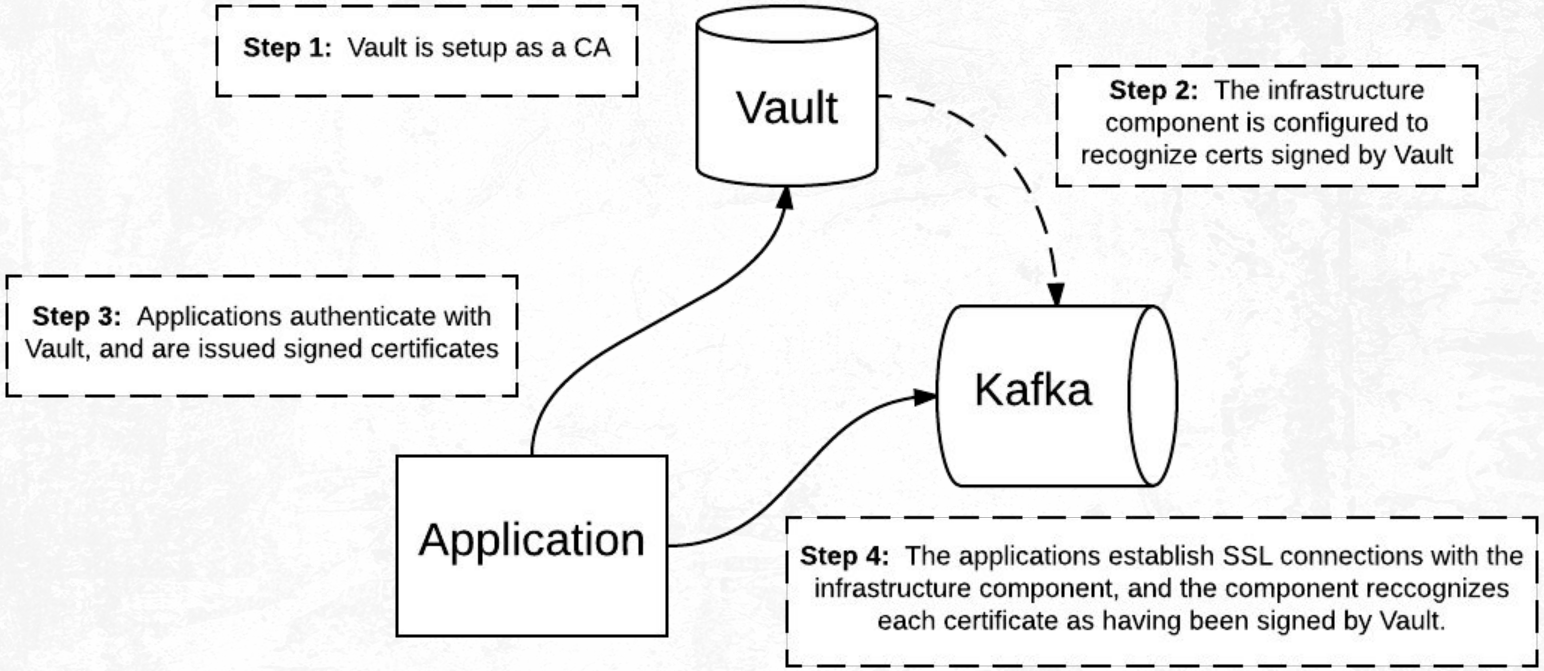
VAULT AS A CERTIFICATE AUTHORITY

- Vault can issue signed x509 certificates
- Message brokers and other infrastructure components can be setup to recognize Vault-signed certs

RUNTIME ACCESS

- Applications can request certs from Vault on the fly, and use them to connect to infrastructure components
- Access controls can limit what a given cert is authorized to do

Example #3: TLS with Vault as a CA



Example #3: TLS with Vault as a CA

- https://github.com/steve-perkins/kafka-tls-demo - Instructions for configuring Kafka with Vault, sample message producer and consumers.

Thanks!

https://github.com/steve-perkins/DevNexus-2017