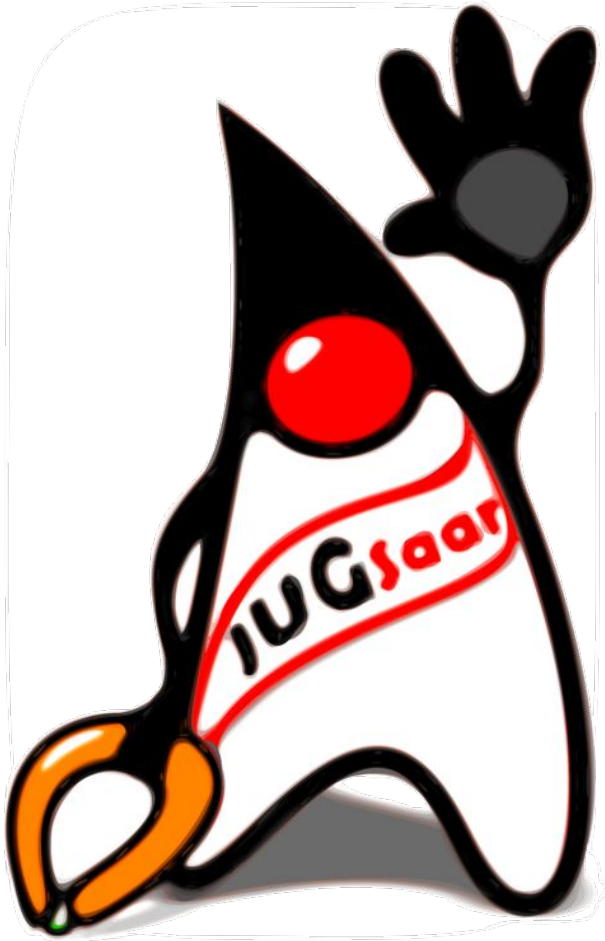
Java User Group Saarland

Service Discovery mit Consul

Thomas Darimont, Jens-Christian Merg eurodata AG

31. Meeting

27 Juni 2017



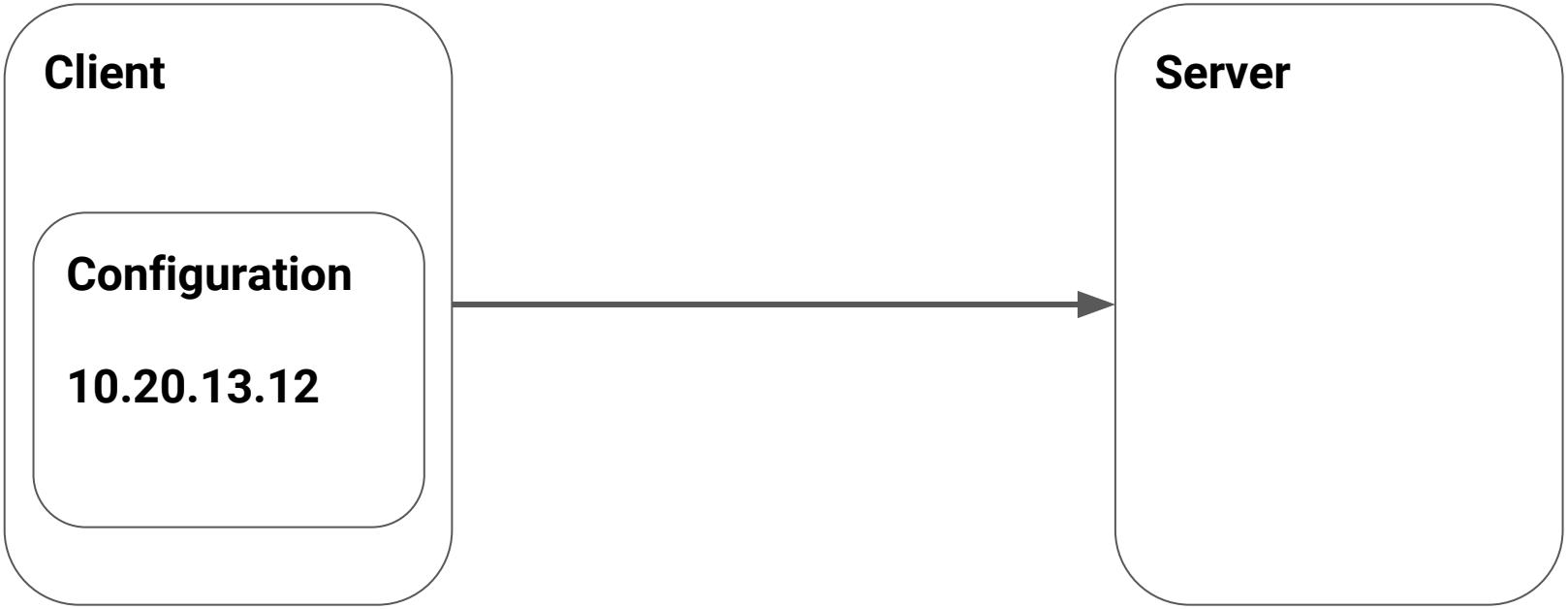
Sponsored by

eurodata

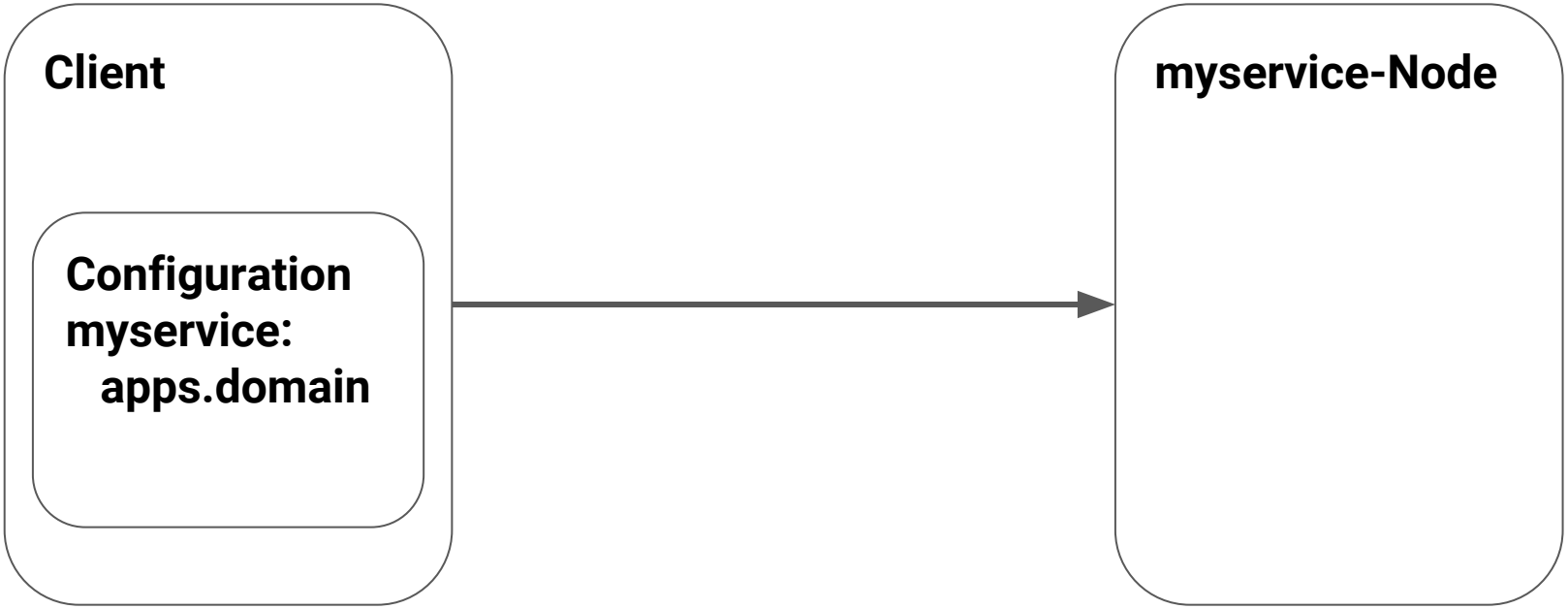
INFOSERVE
>eurodata-Gruppe

Service Discovery?

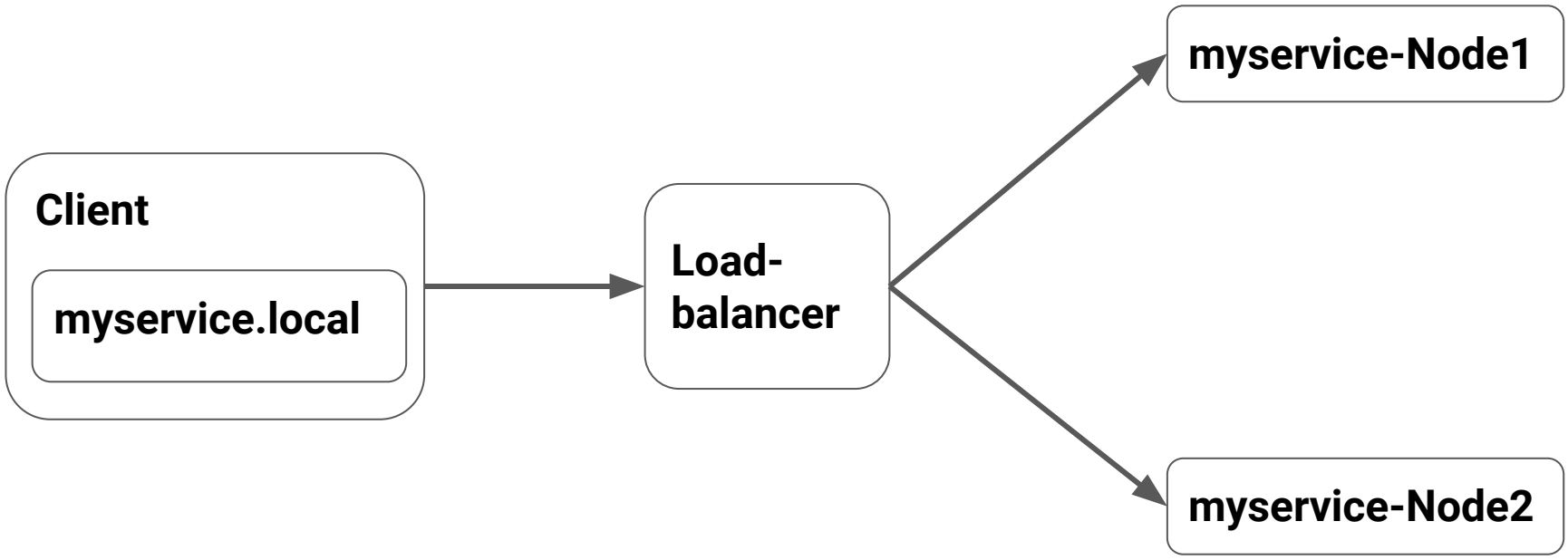
Static Configuration



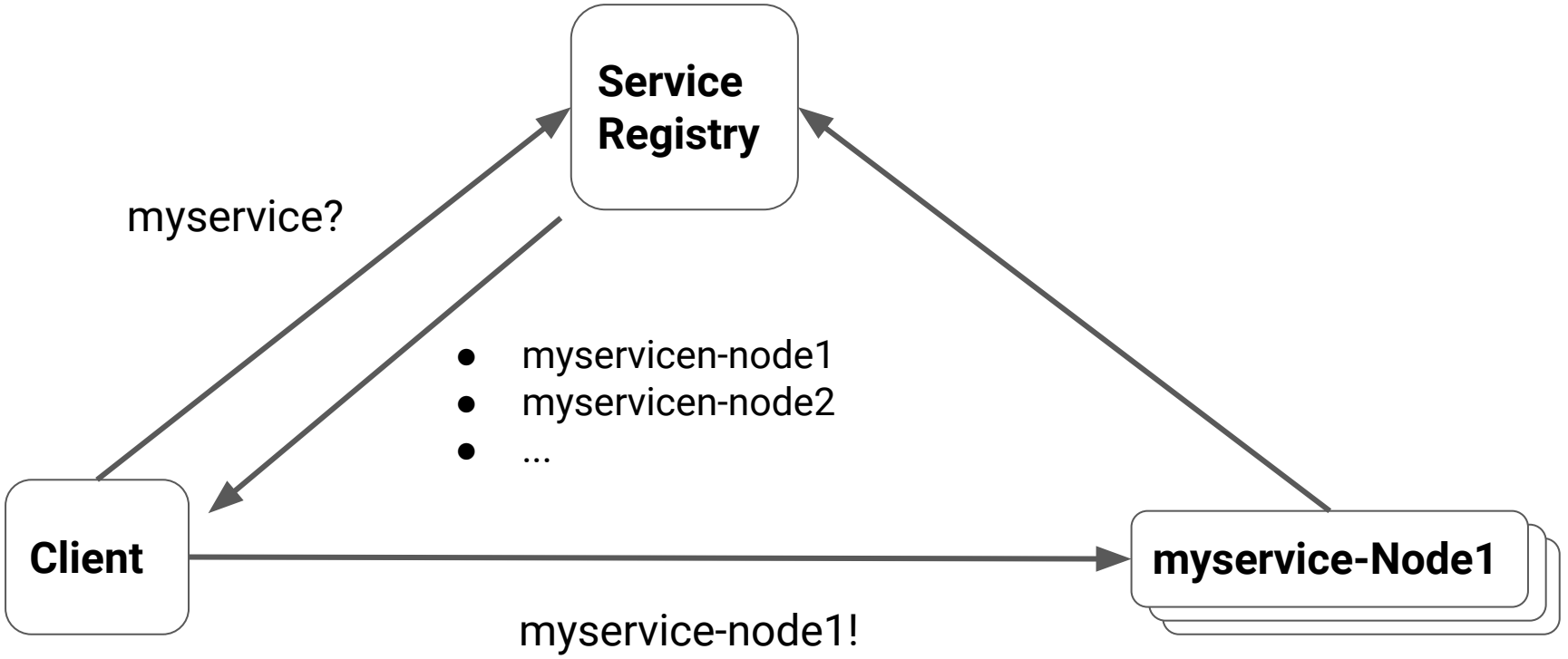
Semantic DNS



DNS and Loadbalancer



Service Discovery with Service Registry



HashiCorp

Consul

Service Discovery and Configuration Made Easy

GET STARTED | DOWNLOAD 0.8.5

Service Discovery

Consul makes it simple for services to register themselves and to discover other services via a DNS or HTTP interface. Register external services such as SaaS providers as well.

Failure Detection

Pairing service discovery with health checking prevents routing requests to unhealthy hosts and enables services to easily provide circuit breakers.

Multi Datacenter

Consul scales to multiple datacenters out of the box with no complicated configuration. Look up services in other datacenters, or keep the request local.

KV Storage

Flexible key/value store for dynamic configuration, feature flagging, coordination, leader election and more. Long poll for near-instant notification of configuration changes.

Consul

“Consul is a tool for service discovery and configuration. Consul is distributed, highly available, and extremely scalable.”

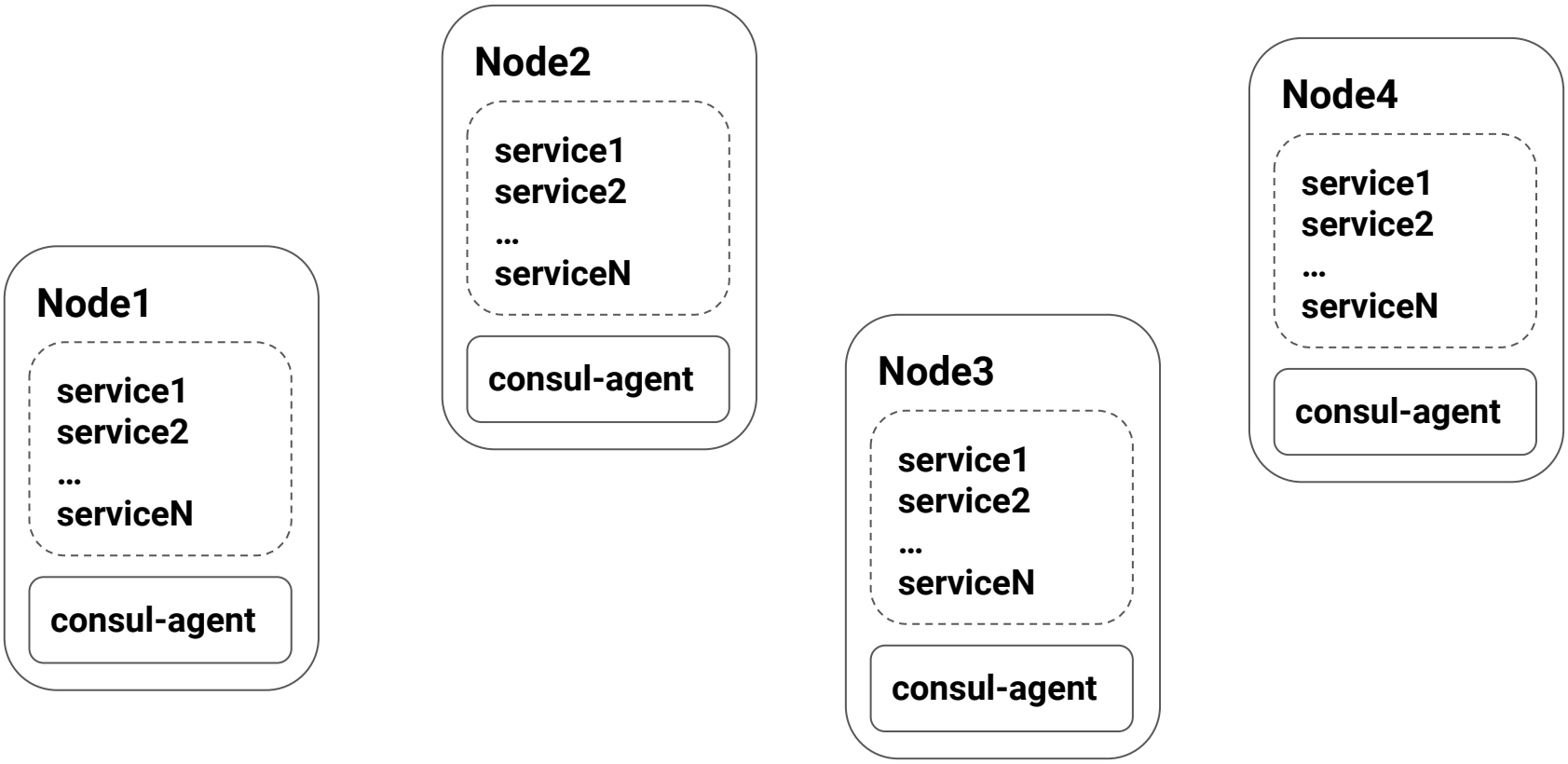
Consul

- Open Source
- Sponsored by Hashicorp (known for Vagrant, Vault…)
- very active development
- Written in Golang
- Features
  - Service Registry
  - HTTP & DNS API
  - Health-Checks
  - Key-Value Store
  - Datacentre Aware
  - Useful Community Tooling

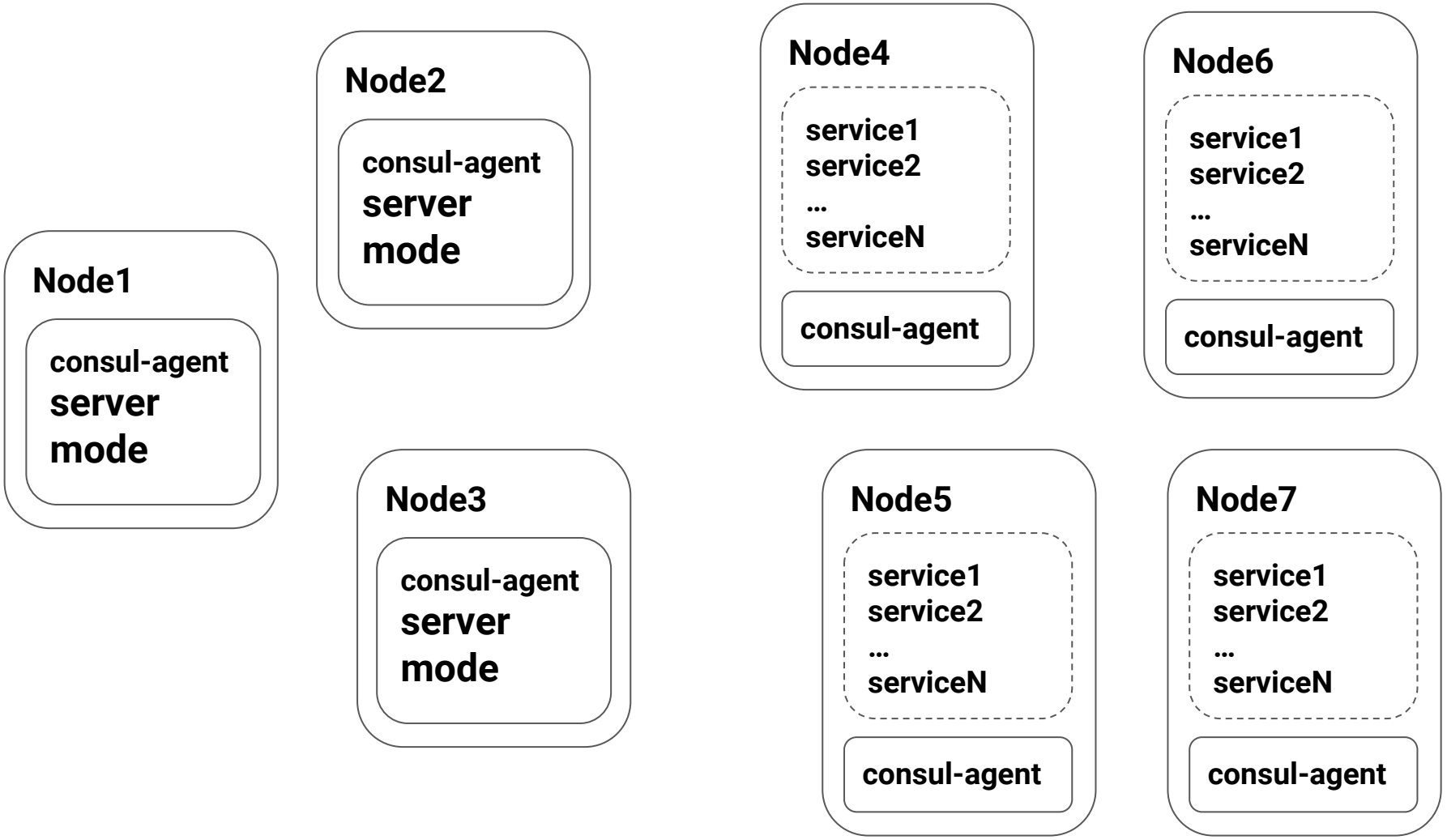
Consul Agent

- Single binary (36mb)
- Small footprint (~32mb)
- Deployed on every Service Node
  - Exposes consul service every to Node
  - Can integrate with DNS via dnsmasqd
- Multiple modes
  - Server
  - Client
- Multiple Consul Agents form a Cluster
  - Usually 3 or 5 Nodes for Quorum
  - Single-Node Cluster for Development possible

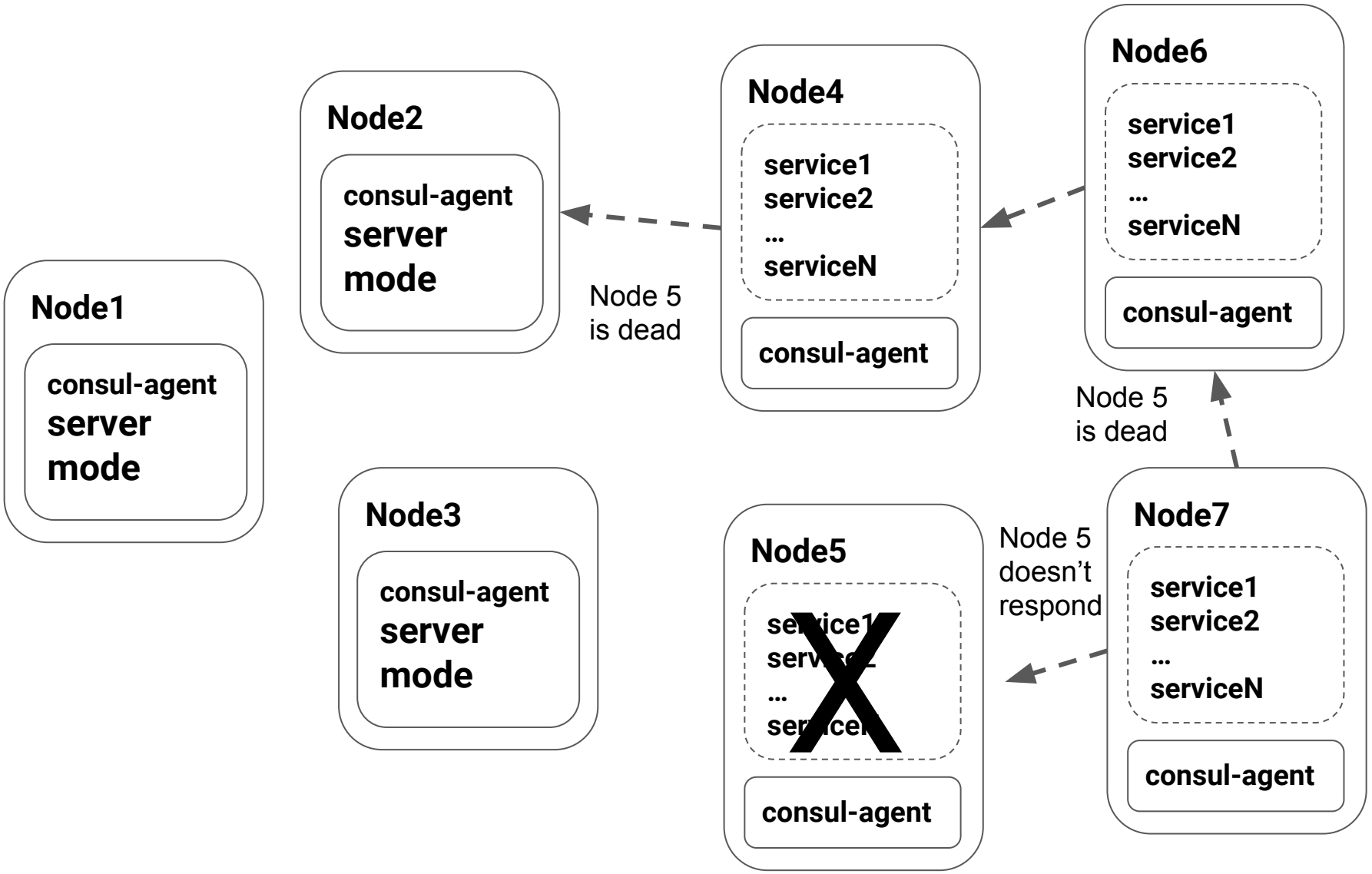
Consul Agents



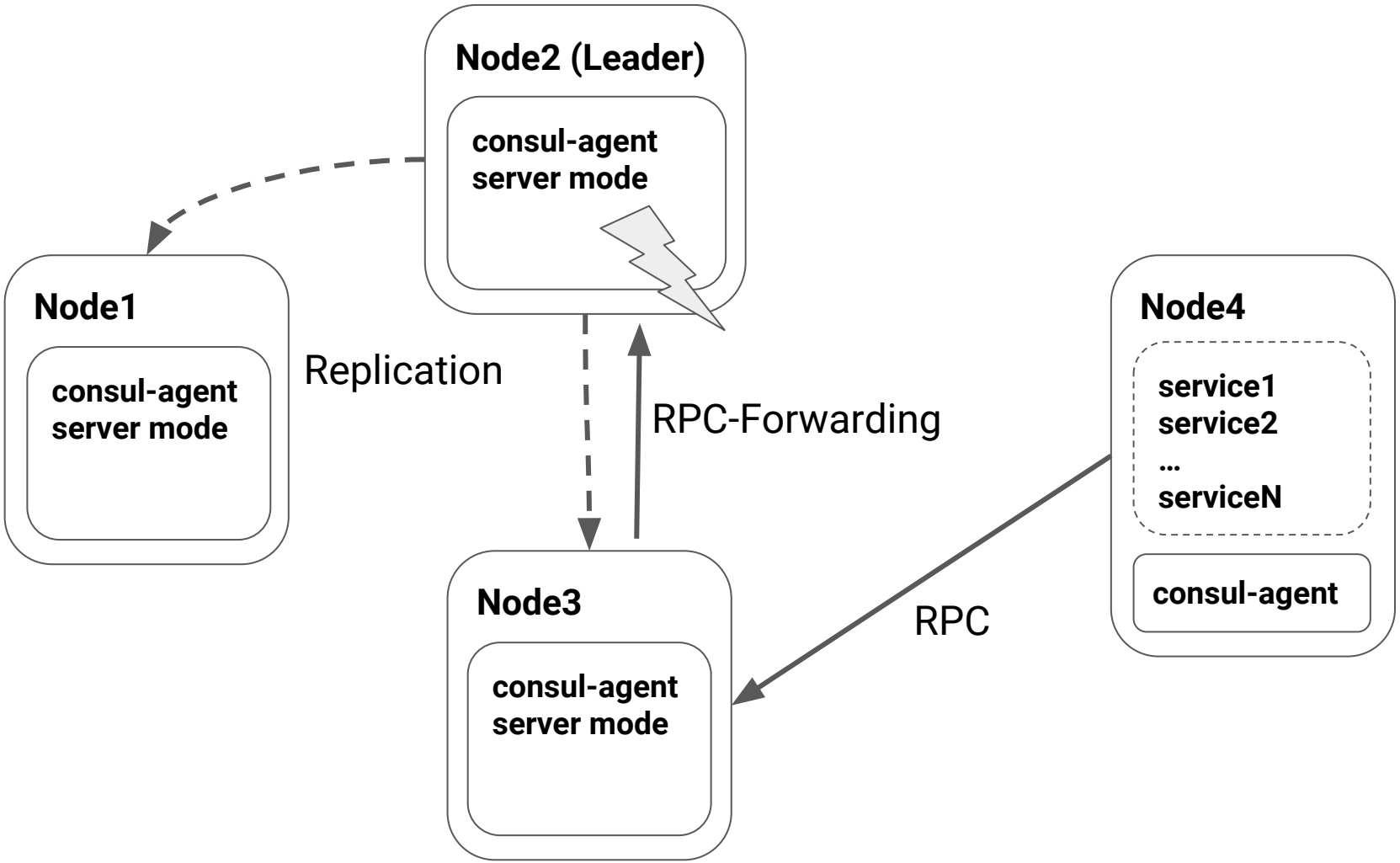
Consul Server Nodes know about Cluster State



Cluster Communication



Intra-Server Communication

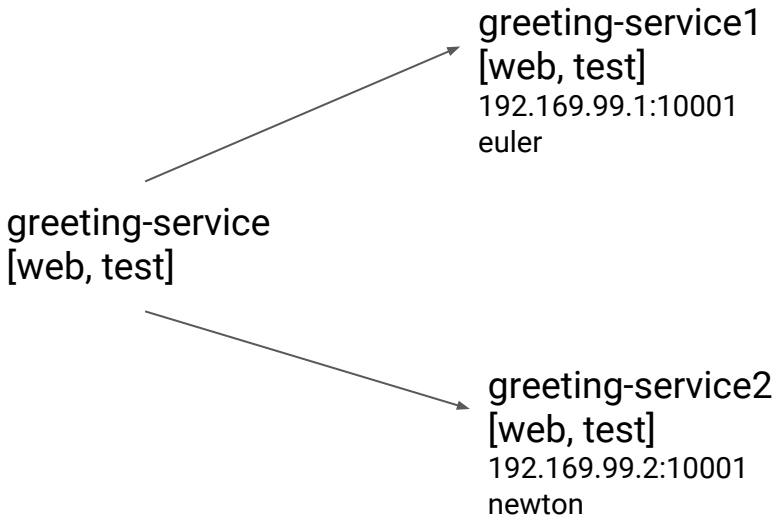


Consul Service Registry

- Binds Service Instances to Logical Services
- Service Registration
  - HTTP API
  - Config File
- Service Query
  - Report all, healthy or problem hosts
  - Query methods: HTTP API, DNS
- Clustered Setup
  - Quorum via RAFT Consensus Algorithm

Anatomy of a Service

- Service vs. Service Instance
- Service
  - Name
  - Tags
- Service Instance
  - Name
  - Tags
  - Address / Port
  - Node *(Host running the service)*
  - Health-Checks

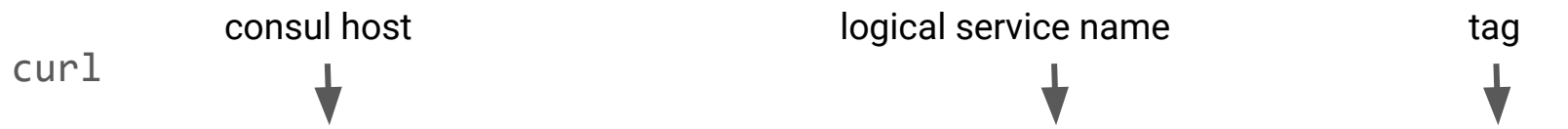


HTTP API

All endpoints fall into one of several categories:

- kv - Key/Value store
- agent - Agent control
- catalog - Manages nodes and services
- health - Manages health checks
- session - Session manipulation
- acl - ACL creations and management
- event - User Events
- status - Consul system status

HTTP API example: Query for Service Instances




```
[{  "ID": "8004086d-20d3-06ca-3c50-a4d242191118",
    "Node": "euler",
    "Address": "127.0.0.1",
    "Datacenter": "dc1",
    "TaggedAddresses": {"lan": "127.0.0.1","wan": "127.0.0.1"},
    "NodeMeta": {},
    "ServiceID": "greeting-service-instance1-test-consul-10001",
    "ServiceName": "greeting-service",
    "ServiceTags": ["test","web"],
    "ServiceAddress": "192.168.178.77",
    "ServicePort": 10001,
    "ServiceEnableTagOverride": false,
    "CreateIndex": 2986,
    "ModifyIndex": 2986
 },...
```

DNS Query Interface

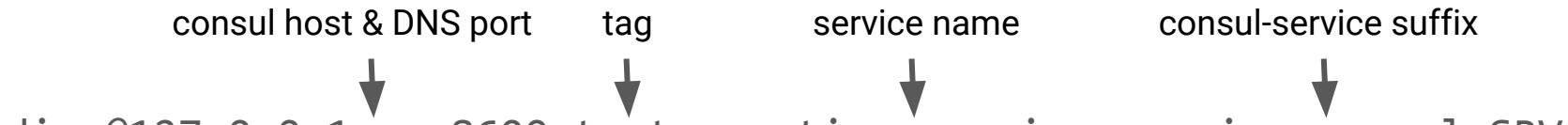
Look up services using Consul's built-in DNS server. Support existing infrastructure without any code change.

```
admin@hashicorp.com: dig web-frontend.service.consul. ANY
; <<>> DiG 9.8.3-P1 <<>> web-frontend.service.consul. ANY
;; global options: +cmd

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29981
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;web-frontend.service.consul. IN ANY

;; ANSWER SECTION:
web-frontend.service.consul. 0 IN A 10.0.3.83
web-frontend.service.consul. 0 IN A 10.0.1.109
```

DNS Query Example




```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> @127.0.0.1 -p 8600  test.greeting-service.service.consul SRV
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24162
;; flags: qr aa rd; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 2
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;test.greeting-service.service.consul. IN SRV

;; ANSWER SECTION:
test.greeting-service.service.consul. 0 IN SRV 1 1 10001
test.greeting-service.service.consul. 0 IN SRV 1 1 10002

;; ADDITIONAL SECTION:
c0a8b24d.addr.dc1.consul. 0  IN  A   192.168.178.77
c0a8b24d.addr.dc1.consul. 0  IN  A   192.168.178.77

;; Query time: 0 msec
;; SERVER: 127.0.0.1#8600(127.0.0.1)
;; WHEN: Mon Jun 26 23:41:58 CEST 2017
;; MSG SIZE  rcvd: 144
```

Configure dnsmasq to forward DNS requests

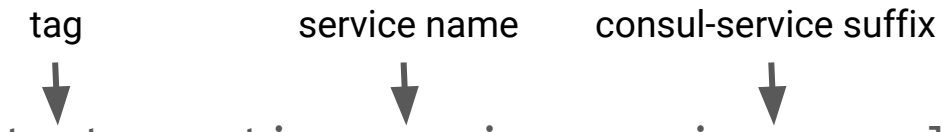
- `dnsmasq` should forward all DNS request for UDP on port 53 to consul DNS port 8600 UDP

- `localhost` could be configured in `resolv.conf` to allow system wide name resolution

- Simple config:
  - `/etc/dnsmasq.d/10-consul`

```
# Enable forward lookup of the 'consul'
domain: server=/consul/127.0.0.1#8600
```

DNS with dnsmasq




```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> @127.0.0.1 -p 8600  test.greeting-service.service.consul SRV
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24162
;; flags: qr aa rd; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 2
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;test.greeting-service.service.consul. IN SRV

;; ANSWER SECTION:
test.greeting-service.service.consul. 0 IN SRV 1 1 10001
test.greeting-service.service.consul. 0 IN SRV 1 1 10002

;; ADDITIONAL SECTION:
c0a8b24d.addr.dc1.consul. 0  IN  A   192.168.178.77
c0a8b24d.addr.dc1.consul. 0  IN  A   192.168.178.77

;; Query time: 0 msec
;; SERVER: 127.0.0.1#8600(127.0.0.1)
;; WHEN: Mon Jun 26 23:41:58 CEST 2017
;; MSG SIZE  rcvd: 144
```

Health Checks

- Script + Interval
  - Runs a script on a given interval
  - Exit status: 0=healthy, 1=warning, other=failed
- TTL - like a *“dead man switch”*
  - Application reports it’s status periodically
  - No report → failed status

Key Value Storage

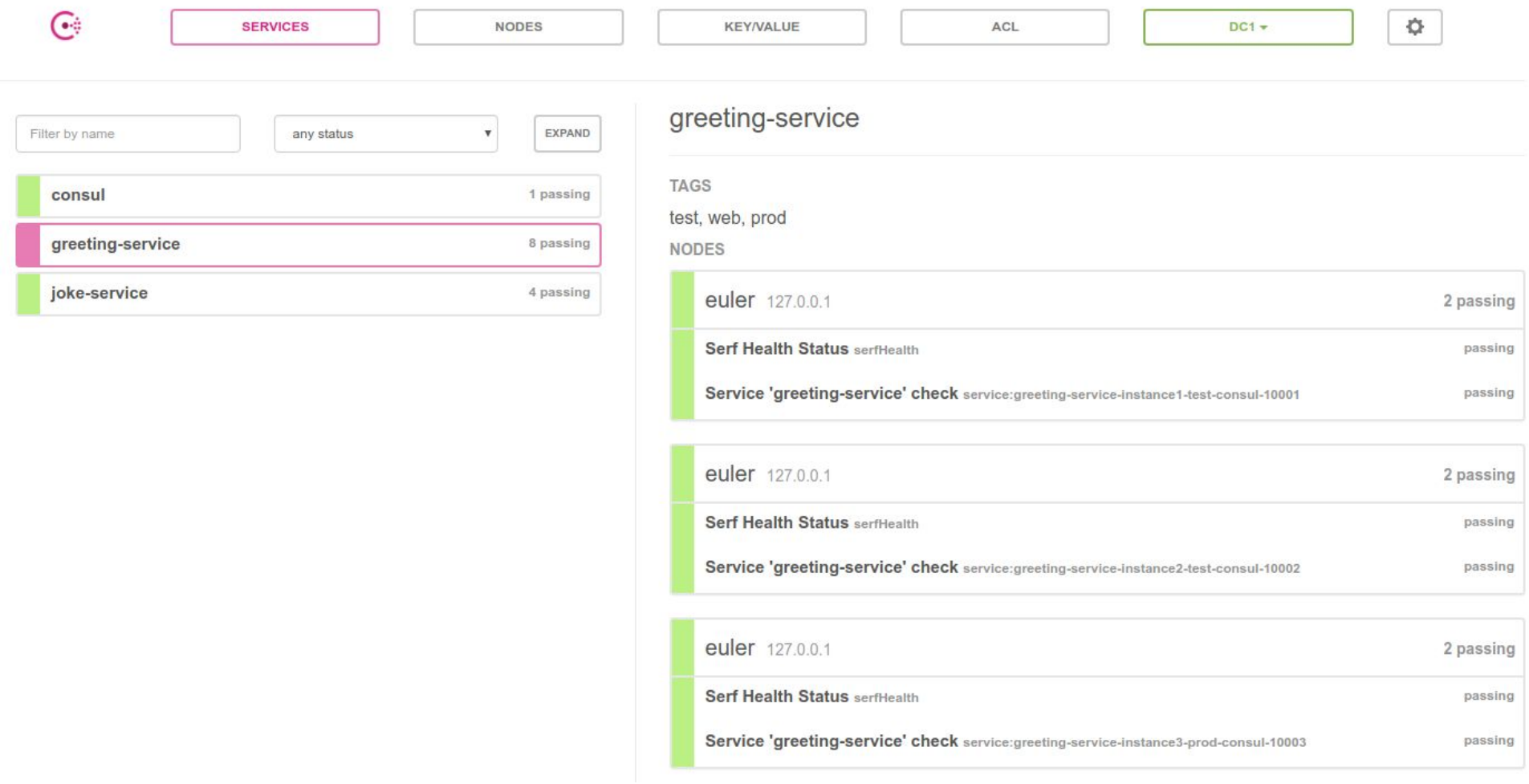
Consul provides a hierarchical key/value store with a simple HTTP API. Managing configuration has never been simpler.

```
admin@hashicorp: consul kv put foo bar
Success! Data written to: foo
admin@hashicorp: consul kv get foo
bar
admin@hashicorp: consul kv get -detailed foo
CreateIndex 5
Flags 0
Key foo
LockIndex 0
ModifyIndex 5
Session -
Value bar
admin@hashicorp: consul kv delete foo
Success! Deleted key: foo
```

Consul UI

consul Service accessible on every node!

http://consul.service.consul:8500/ui/#/dc1/services/greeting-service



Additional Features

- Node Registry
  - Determine which Nodes host web apps
- KV-Store
  - Allows to store scoped Key-Value pairs
  - Watch for Key-Value Changes
  - Distributed Locks
- ACLs
  - Restrict access to services / config store
- Cluster Management
  - Leader Election

Consul Integrations

- DNS
  - Easiest way to integrate legacy applications
  - just use a .consul address
  - Beware of DNS Packet Sizes & Caching (TTL)
- HTTP API
  - Register or Query Services via HTTP requests
- Use dedicated Client Libraries (Java, C#, NodeJS, PHP, Python…)
  - Consul Java Client library
- Framework Integrations
  - Spring Cloud Consul
    - Service Discovery
    - Centralized Configuration Store
    - Client-Side Load-Balancing

Consul Integrations - DNS Setup

- Integrating a Consul Cluster into a running DNS Setup via Zone Forwarding
  - Delegate .consul Zone on the primary DNS resolver
  - Centralized discovery of `.consul` zone for consul cluster
    - service discovery without reconfiguration of local DNS config
    - `consul.service.dc.consul` -> resolves to `consol-agent` on `localhost`
  - Bind forwards all DNS requests to one member of the consul server cluster (round robin)
  - Default consul `DNS TTL = 0`

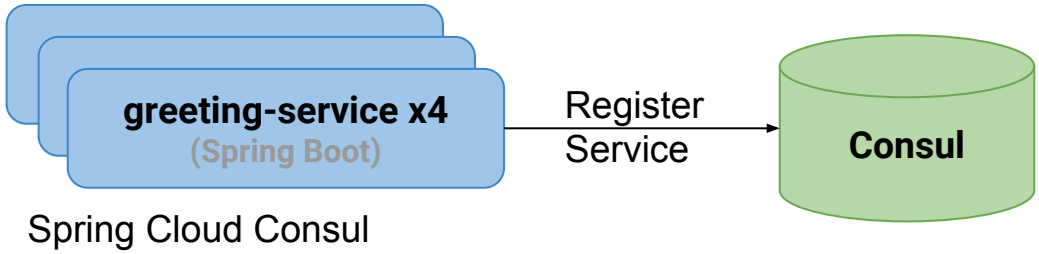
- Example, bind9

```
// consul
zone "consul" IN {
        type forward;
        forward only;
        forwarders { 1.2.3.4 port 8600; 1.2.3.n port 8600; };
};
```

Service Discovery with Consul & Spring Cloud

Consul Use Case Examples

1. Dynamic Service Registration

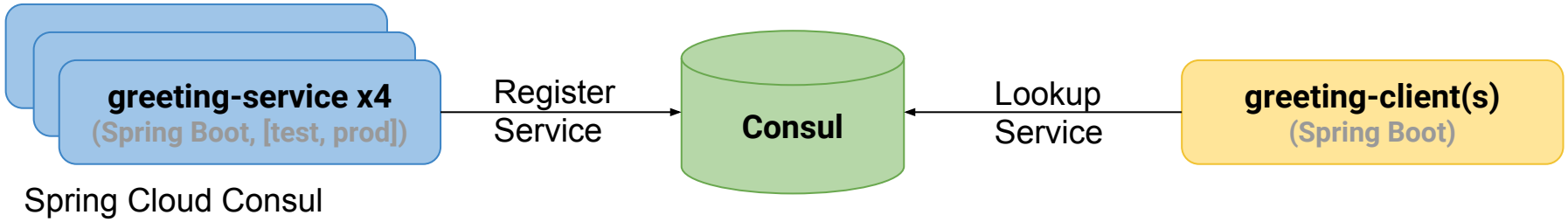


Spring Cloud Consul

Consul Use Case Examples

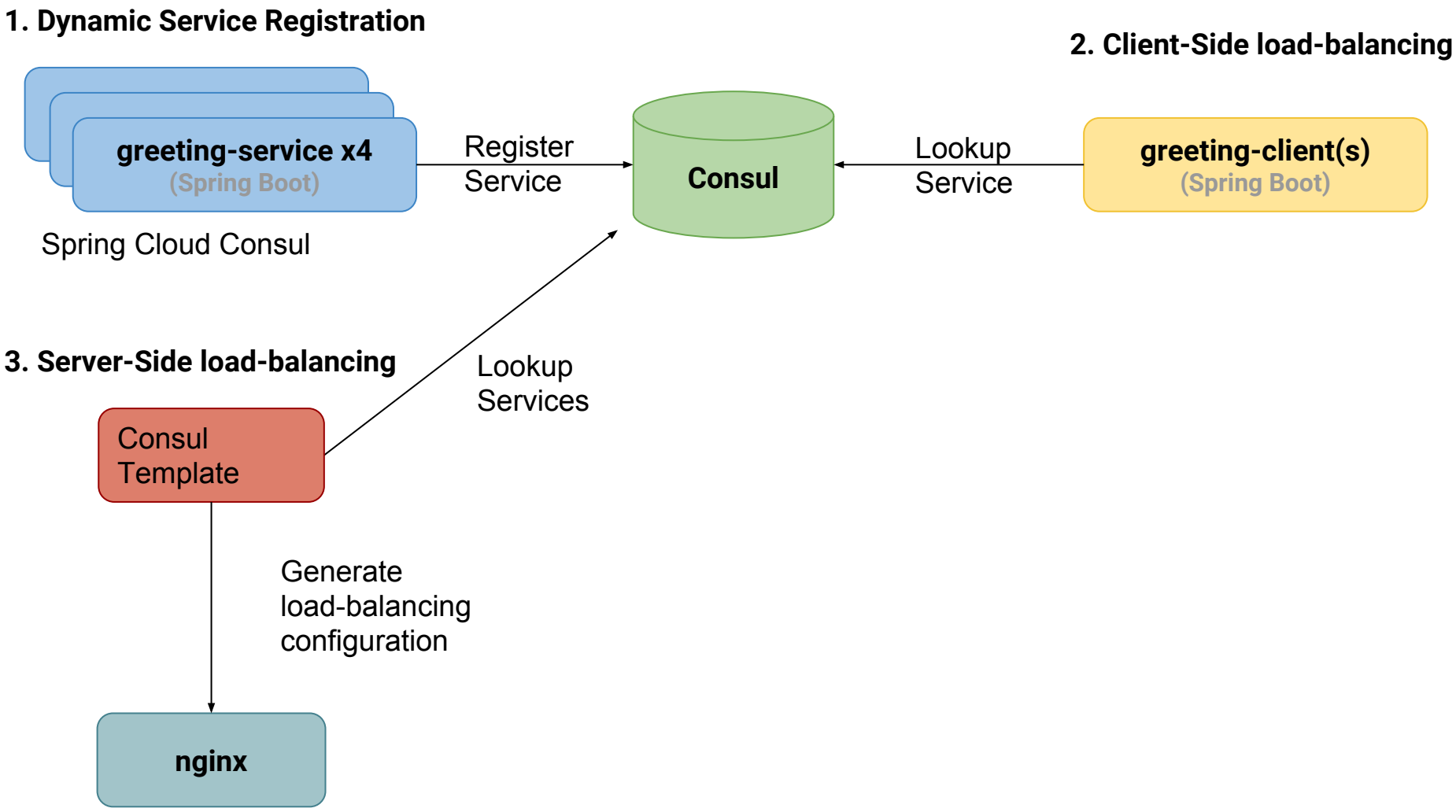
1. Dynamic Service Registration

2. Client-Side load-balancing

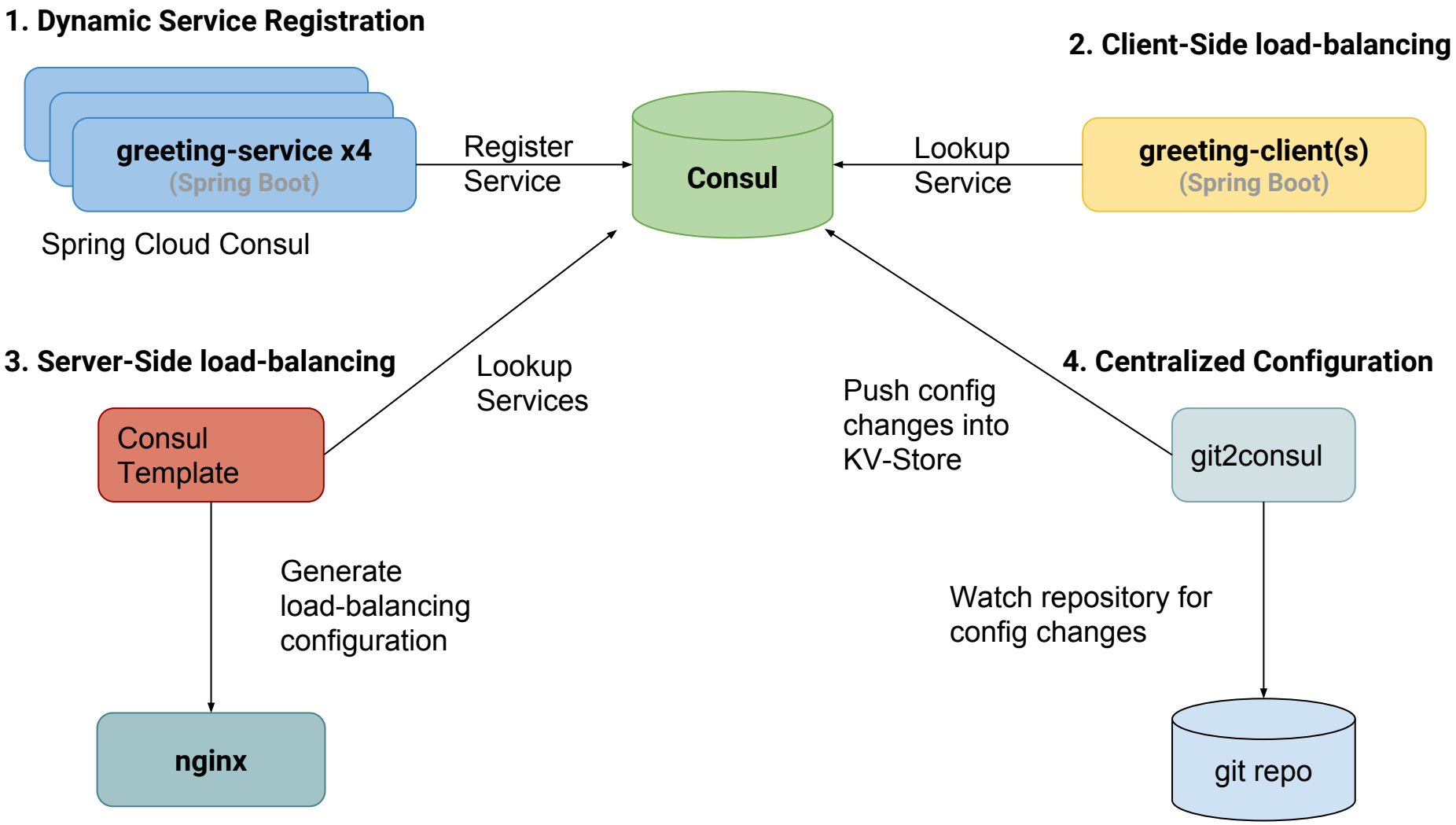


Spring Cloud Consul

Consul Use Case Examples



Consul Use Case Examples



https://github.com/jugsaar/jugsaar-meeting-31

Technologies

- Spring Boot
- Spring Cloud
- Spring Cloud Consul
- Spring Cloud Config Server (Consul)
- Spring RestTemplate
- Ribbon
- Netflix Feign
- Netflix Hystrix
- consul-template
- git2consul

Summary

- Very stable
- Easy to setup & use
- Good documentation
- Active Community
- Lot of Tooling
- Plays well with others
  - Java, Vault, Docker, Kubernetes and other Platforms

Links

- Code & Slides https://github.com/jugsaar/jugsaar-meeting-31
- Consul https://www.consul.io/
- Consul Intro https://www.consul.io/intro/index.html
- Consul Github https://github.com/hashicorp/consul
- Consul Tools https://www.consul.io/downloads_tools.html
- Consul in practice https://stripe.com/blog/service-discovery-at-stripe
- Spring Cloud Consul http://cloud.spring.io/spring-cloud-consul/
- Consul Template https://github.com/hashicorp/consul-template
- Git2Consul https://github.com/Cimpress-MCP/git2consul
- Ribbon https://github.com/Netflix/ribbon
- Hystrix https://github.com/Netflix/Hystrix/wiki
- Feign https://github.com/OpenFeign/feign