HashiCorp
Consul

Consul Service Discovery

July 2022

Agenda

1. Service Discovery & Registration
2. Health Checks
3. Consul Agent
4. Service Configuration
5. Geo Failover & Prepared Queries

Consul Enterprise Path to Production

Weeks 1-3

Consul Deployed

- Kick Off Webinar
- Architectural Deep Dive Webinar
- Terraform starter code modules
- Operationalizing Consul

Weeks 4-5

Service Discovery

- Service Discovery and Health Monitoring
- Service Registration and Consumption
- Office Hours

Weeks 6-7

Service Mesh and API Gateway

- Service Mesh Configuration and Consumption
- Ingress & API Gateways
- Demo
- Office Hours

Weeks 8-9

Service Configuration and Integrations

- Consul Integrations & NIA
- Demo
- Office Hours

Weeks 10

Ready for Production!

- Exit Ramp & Operational Readiness Checklist

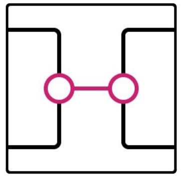
01

Service Registry & Discovery?

Service Registry & Discovery

Dynamically locate any application or infrastructure service to simplify network connectivity

- Eliminate the need for East/West Load Balancing
- Enable other Consul use cases
    - Core building block of a Service Mesh
    - Software Load Balancing
    - Network Infrastructure Automation
- Automate Geographic Failover using Prepared Queries



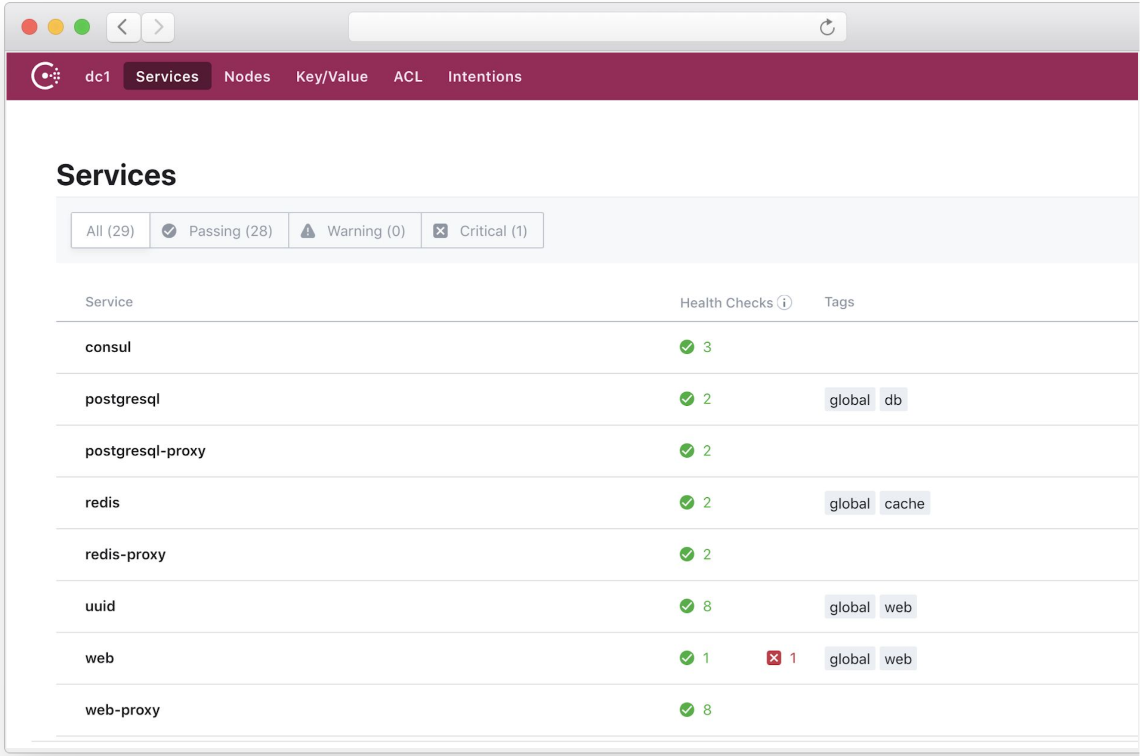
Service Discovery for Deployment

- Cross Platform Deployment
  - Make applications deployed across multiple platforms and clouds available for consumption
  - Simplify operations
- Blue / Green Deployments
  - On-premise to cloud migration
  - Upgrade of a set of hosts for routine maintenance
- Blue / Green / Yellow / Grey
  - Exposing a specific version of an app
  - Leveraging rich metadata to target specific instances of a service

Service Registry

Consul catalog provides a real-time directory which includes:

- What services are running
- Service network location
- Service health status
- Platform agnostic views



Services

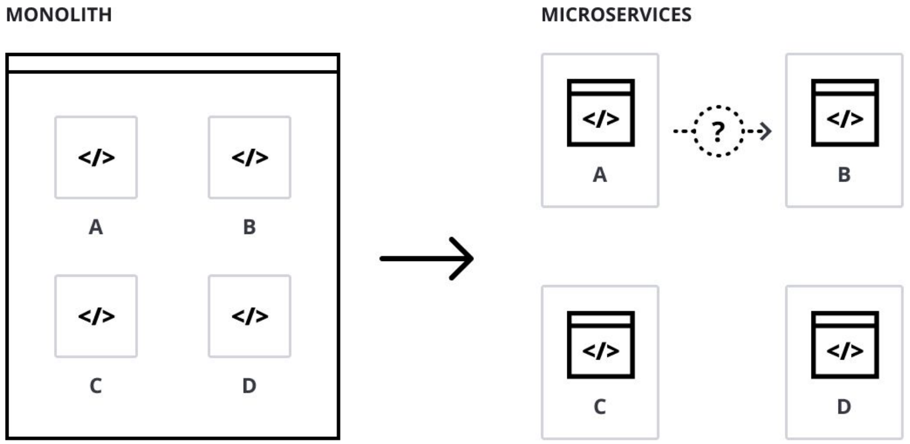
All (29) | Passing (28) | Warning (0) | Critical (1)

Service	Health Checks		Tags
consul	✓ 3		
postgresql	✓ 2		global db
postgresql-proxy	✓ 2		
redis	✓ 2		global cache
redis-proxy	✓ 2		
uuid	✓ 8		global web
web	✓ 1	✗ 1	global web
web-proxy	✓ 8		

Service Registry

Microservices cause east-west traffic growth

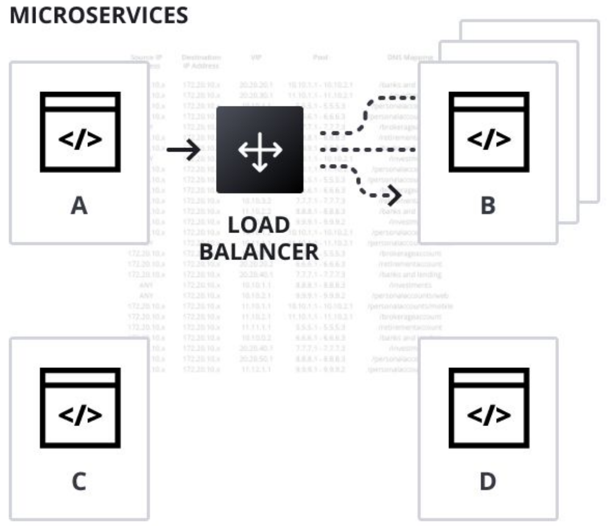
- Microservices communicate over the network in east-west traffic patterns
- Service-to-service traffic needs to be routed dynamically as services scale up and down frequently without long-lived IPs.



Service Registry

Load balancers for east-west traffic scale poorly

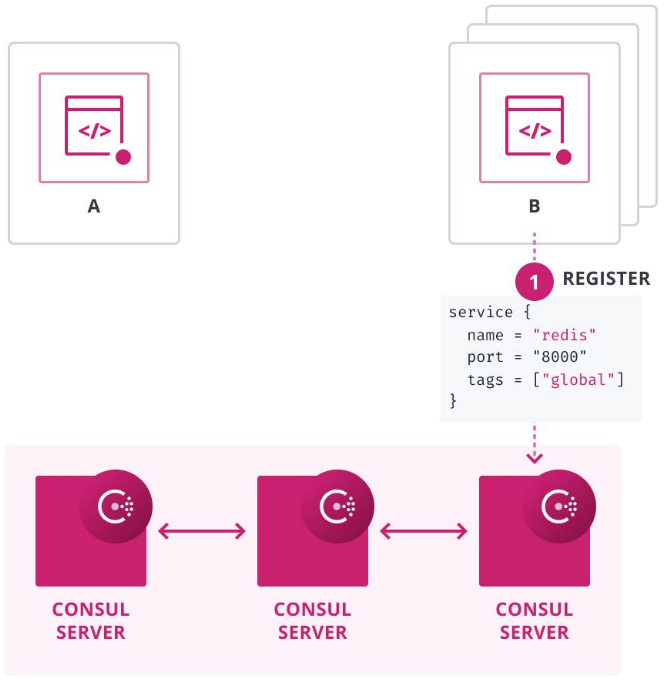
- Load balancers can front a service tier and provide a static IP
- Load balancers add cost, latency, single points of failure, and must be updated as services scale up/down.



Service Registration

Service discovery for connectivity

- Consul provides a registry of all the running nodes and services with current health status
- Services can register to mark themselves (IP + port) as available via config files or API.



```
service {
  name = "redis"
  port = "8000"
  tags = ["global"]
}
```

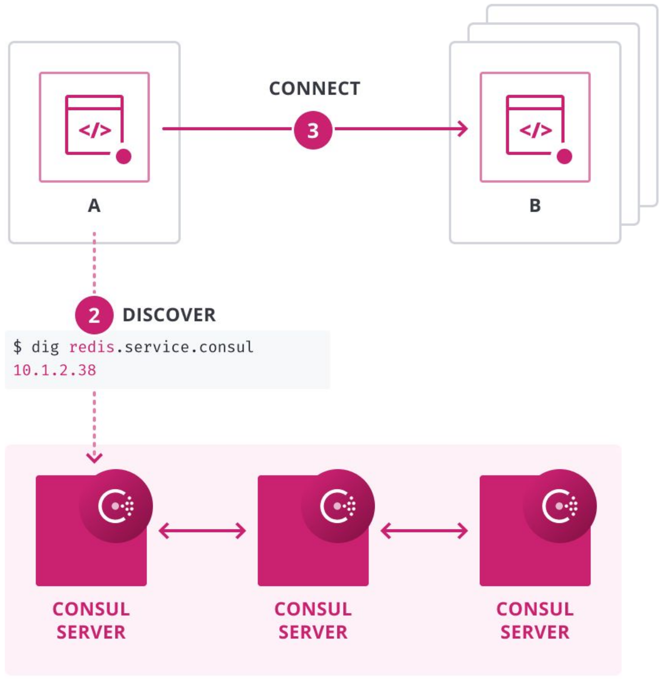
Service Registration

Allow services to connect directly

- For a service to communicate with any other service it queries the registry for the healthy instances of those services

- Two services can connect directly without any operator intervention

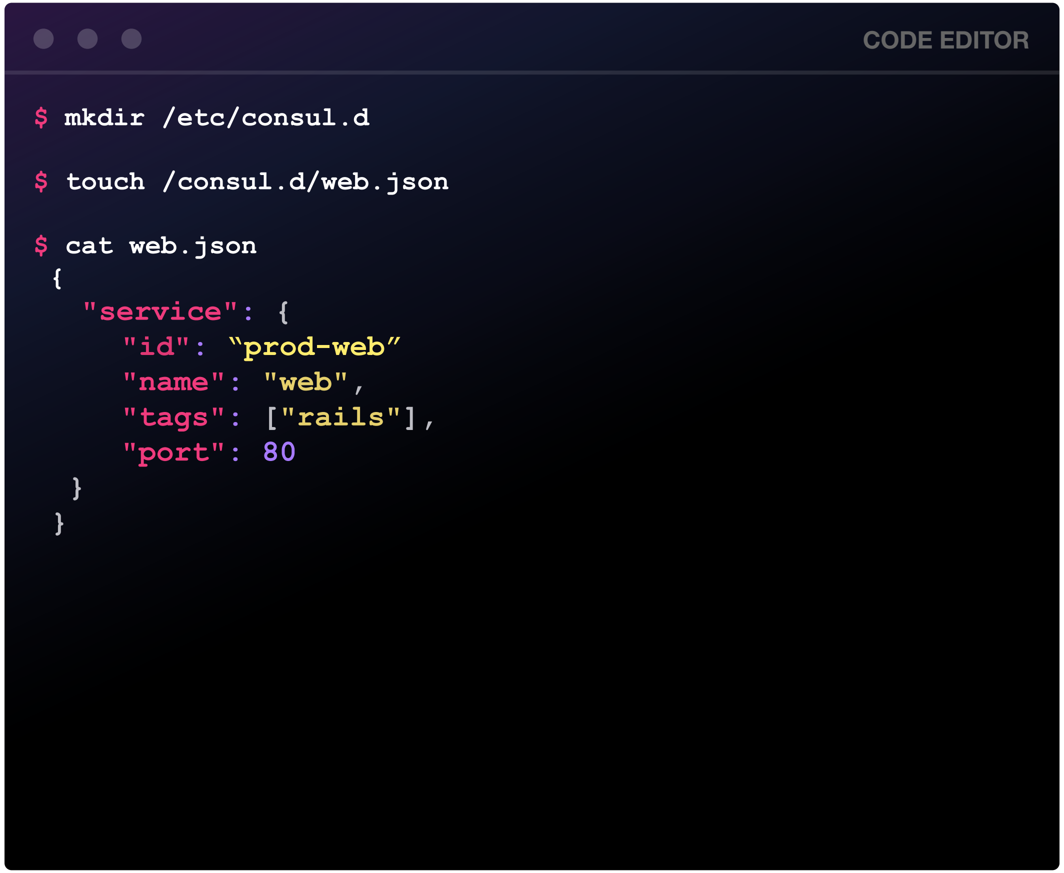
- Service catalog can be queried via DNS or API



```
$ dig redis.service.consul
10.1.2.38
```

Define a Service

Sample service definition


CODE EDITOR

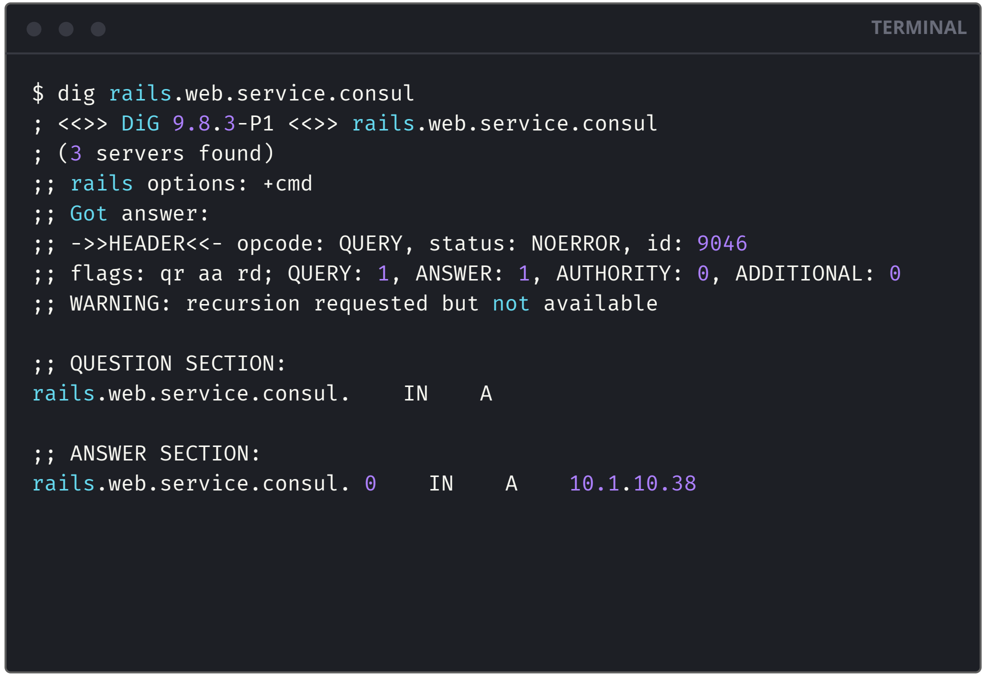
```
$ mkdir /etc/consul.d

$ touch /consul.d/web.json

$ cat web.json
 {
   "service": {
     "id": “prod-web”
     "name": "web",
     "tags": ["rails"],
     "port": 80
   }
 }
```

DNS Query Interface

- Applications can use Consul DNS for service discovery without direct integration with Consul
- Commonly used to enable service discovery for legacy applications
- Leverage existing DNS deployments for service discovery



```
$ dig rails.web.service.consul
; <<>> DiG 9.8.3-P1 <<>> rails.web.service.consul
; (3 servers found)
;; rails options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9046
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
rails.web.service.consul.    IN    A

;; ANSWER SECTION:
rails.web.service.consul. 0    IN    A    10.1.10.38
```

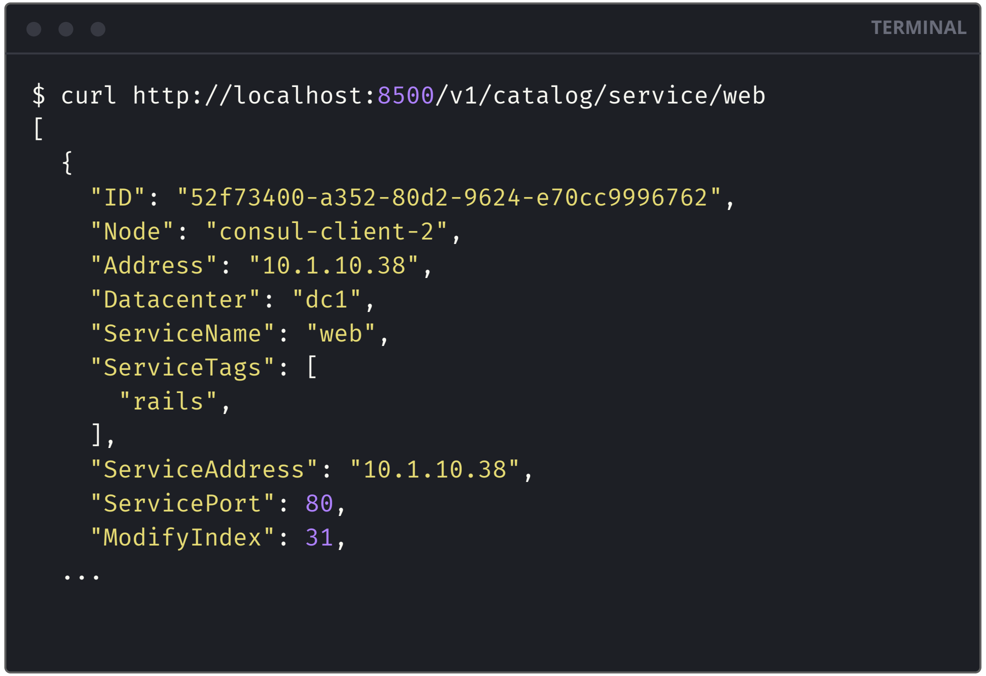
DNS Query Interface

Methods for using the Consul DNS interface

- Custom DNS resolver library pointed at Consul
- Set Consul as the DNS server for node(s) and use a recursive configuration so that non-Consul queries also resolve
- Forward all queries for the "consul." domain to a Consul agent from the existing DNS server

HTTP API Interface

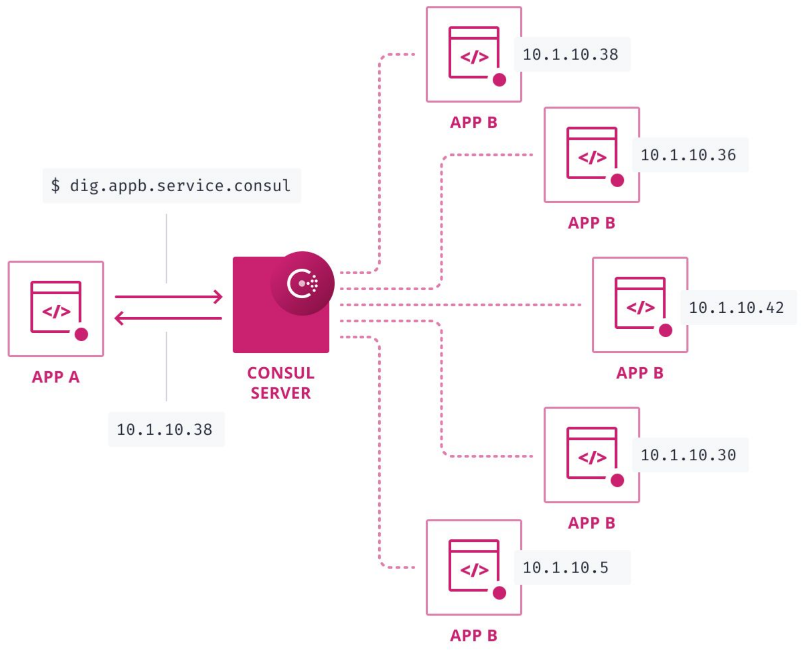
- The Consul service registry API allows for more complex tasks beyond basic DNS functionality
- API calls can query the service registry for nodes, services, and health check information
- API supports blocking queries, or long polling, for changes
- Automation and IAC tools can respond to service registrations or health status changes to update configurations or traffic routing in real time


TERMINAL

```
$ curl http://localhost:8500/v1/catalog/service/web
[
  {
    "ID": "52f73400-a352-80d2-9624-e70cc9996762",
    "Node": "consul-client-2",
    "Address": "10.1.10.38",
    "Datacenter": "dc1",
    "ServiceName": "web",
    "ServiceTags": [
      "rails",
    ],
    "ServiceAddress": "10.1.10.38",
    "ServicePort": 80,
    "ModifyIndex": 31,
  ...
```

Load Balancing via Consul DNS

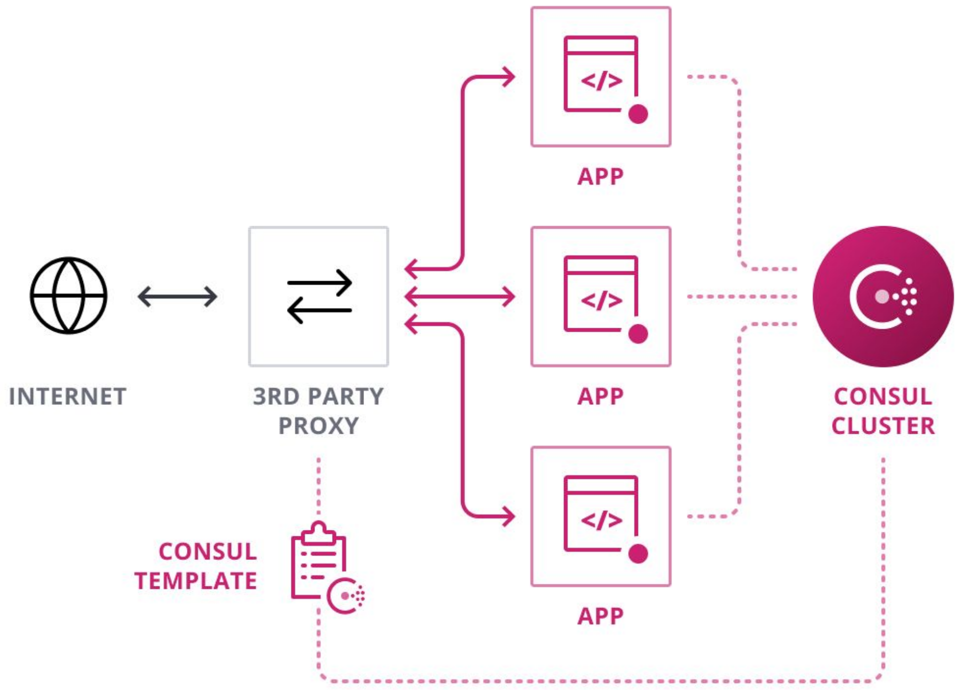
- Leverage Consul's zero-touch DNS interface
- Randomized Round-Robin load balancing
- Integrated with health checks. Entries for services that are failing health checks are automatically filtered out to avoid routing to unhealthy hosts



Network Middleware Automation

Consul enables a "publisher-subscriber" model

- Services "publish" network location automatically
- Middleware "subscribes" to the service changes
- Service changes trigger dynamic reconfiguration automatically



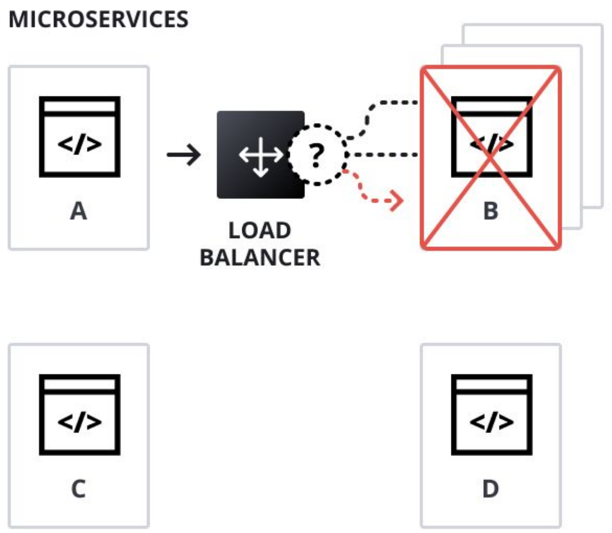
02

Health Checks

Health Checks

Visibility into service health status

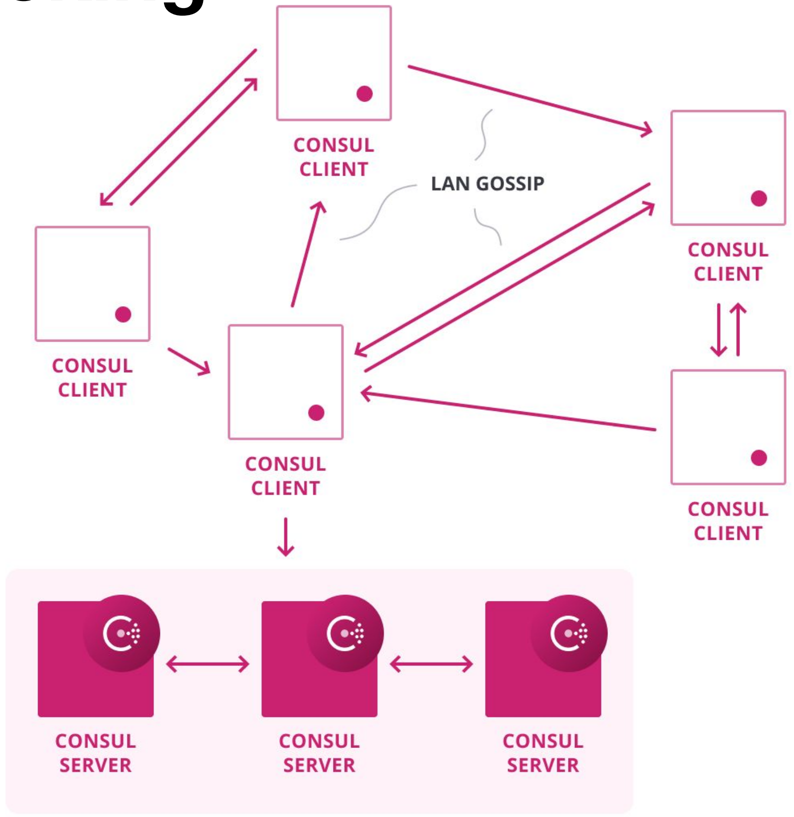
- Health checks are critical to prevent routing to services that are unhealthy
- Centralized approaches relying on heartbeating or periodic updates easily overload servers and lead to scaling issues



Distributed Health Checking

Consul's Gossip Protocol provides an efficient failure detector that scales massively without concentrating the work on any servers.

- Consul agent runs health check locally
  - Only state changes get pushed to Consul servers
  - Removes unhealthy nodes from service discovery layer
- Many types of checks available including: Nagios-compatible scripts, Docker, HTTP and TCP
- Rich set of health checks beyond basic liveness checks



Health Checks

Application-level checks - associated with a specific service

Node checks - monitor the health of the entire node

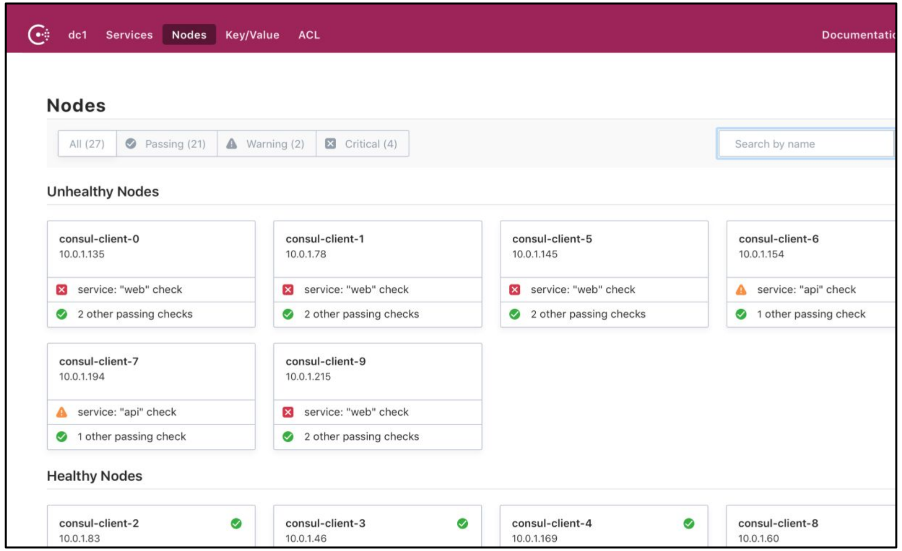
- **Defined via**
  - Configuration file
  - HTTP interface - persist with the node
- **Initially set to “critical”**

  Can be override by specifying the “status” field in the definition
- **Multiple check definitions**

  Multiple check definitions can be defined in a configuration file

Health Check Types

- **Script + interval** - Invokes an external application that performs the health check
- **HTTP + interval** - “GET” request to specified URL, wait specified interval between requests
- **TCP + interval** - connection attempt to IP/hostname & port, configurable interval between attempts, defaults to localhost if no hostname set
- **Time to Live (TTL)** - “dead man’s switch” operational mode, check’s state must be updated periodically



Nodes

All (27) | Passing (21) | Warning (2) | Critical (4)

Search by name

Unhealthy Nodes

consul-client-0
10.0.1.135
service: "web" check
2 other passing checks

consul-client-1
10.0.1.78
service: "web" check
2 other passing checks

consul-client-5
10.0.1.145
service: "web" check
2 other passing checks

consul-client-6
10.0.1.154
service: "api" check
1 other passing check

consul-client-7
10.0.1.194
service: "api" check
1 other passing check

consul-client-9
10.0.1.215
service: "web" check
2 other passing checks

Healthy Nodes

consul-client-2
10.0.1.83

consul-client-3
10.0.1.46

consul-client-4
10.0.1.169

consul-client-8
10.0.1.60

Health Check Types

- **Docker + interval** - invoke an external application packed in a Docker Container
- **gRPC + Interval** - gRPC health checking protocol based, updates configured endpoint with configurable interval, can be TLS enabled
- **H2ping + interval** - http2 based ping, assumed to be TLS by default
- **Alias** - check the health state of another node or service

Health Check Definitions

Service-level circuit breaker

Consul enables services to easily provide circuit breakers with custom scripts.

Script Check JSON

```json
{
  "check": {
    "id": "mem-util",
    "name": "Memory Utilization",
    "script": "/usr/local/bin/check_mem.py",
    "interval": "10s"
  }
}
```

TTL Check HCL

```hcl
check = {
 id = "web-app"
 name = "Web App Status"
 notes = "Web app does a curl internally every 10 seconds"
 ttl = "30s"
}
```

03

Consul Agent

Consul Agent

- Consul agent gets deployed on **every** Consul **server node**
- Consul agent gets deployed on **every client** that participates in service discovery, service mesh, and/or active health checks
- Gets deployed on **every** Kubernetes **worker node**
- Only non-default values must be set in agent configuration file
- Configuration can be <u>read from multiple files</u>

Consul Agent Configuration

- Client Node
- Service Registration
- Health Checks

Agent Configuration HCL

```hcl
node_name  = "consul-client"
server     = false
datacenter = "dc1"
data_dir   = "consul/data"
log_level  = "INFO"
retry_join = ["consul-server"]
service {
 id      = "dns"
 name    = "dns"
 tags    = ["primary"]
 address = "localhost"
 port    = 8600
 check {
   id       = "dns"
   name     = "Consul DNS TCP on port 8600"
   tcp      = "localhost:8600"
   interval = "10s"
   timeout  = "1s"
 }
}
```

Kubernetes Pod Resource Manifest

dashboard.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: dashboard
spec:
  serviceAccountName: dashboard # Authenticate Kube workload with
Consul
  containers:
    - name: dashboard
      image: hashicorp/dashboard-service:0.0.4
      ports:
        - containerPort: 9002
      env:
        - name: COUNTING_SERVICE_URL
          value: "http://counting:9001" # Transparent Proxy
automatically configures mesh routing
---
apiVersion: v1
kind: ServiceAccount
metadata:
  # Service Account used to authenticate with Consul ACL system
  # Service Account name becomes Consul service name (unless
otherwise annotated)
   name: dashboard
```

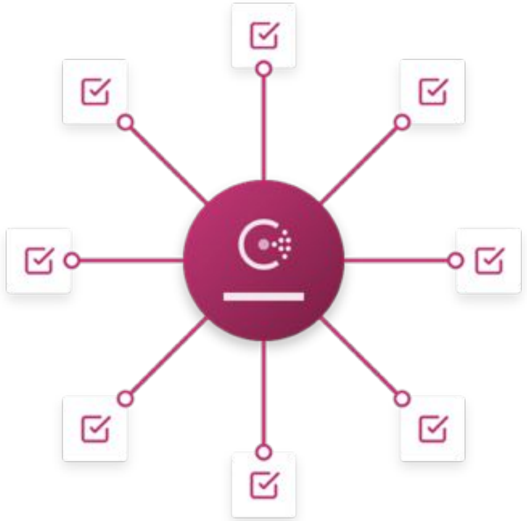
04

Service Configuration

Service Configuration

Dynamic configuration across distributed services in milliseconds.

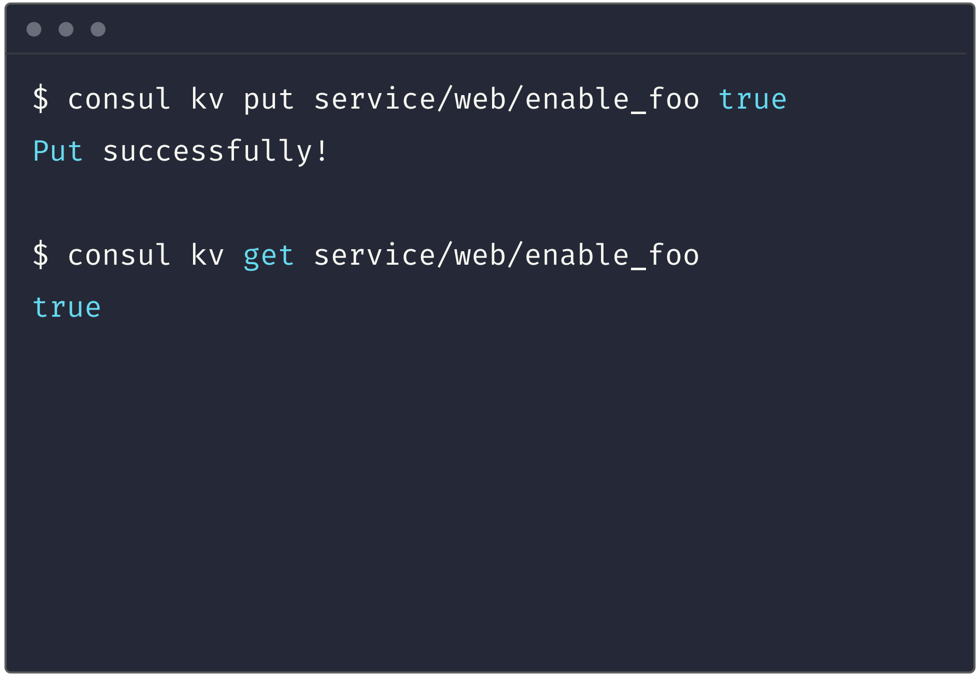
- **Improve Productivity** by avoiding manual updates to thousands of service instances
- **Reduce Risk** by pushing consistent configuration changes across all distributed services in real-time
- **Reduce Cost** by eliminating the need for config management tools for runtime configuration



Hierarchical Key Value Store

Store and retrieve dynamic configuration, feature flagging, coordination and more metadata.

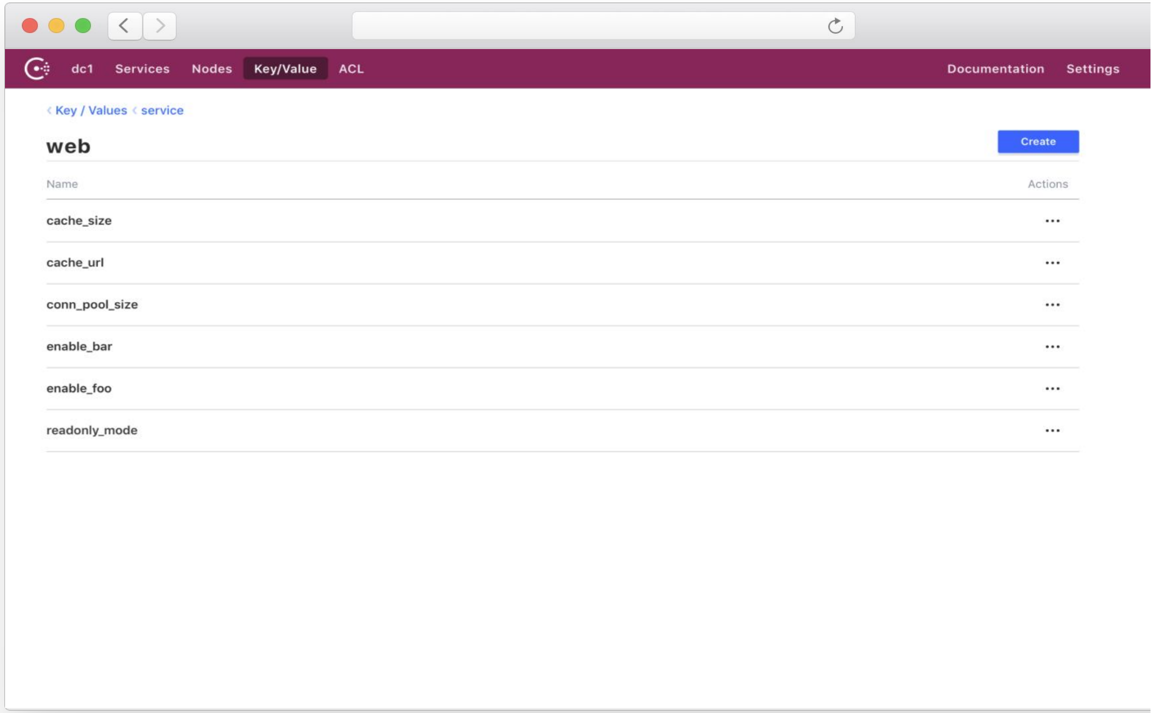
- Highly-available, globally accessible key-value store
- Folder-like architecture allows for easy organization
- ACLs to enforce policy and access
- Bulk export and import of key value pairs
- Accessible via HTTP API
- Can be used via the CLI or tools like curl
- Automated backup via snapshot agent



```
$ consul kv put service/web/enable_foo true
Put successfully!

$ consul kv get service/web/enable_foo
true
```

K/V Store Web UI



Key / Values ‹ service

web

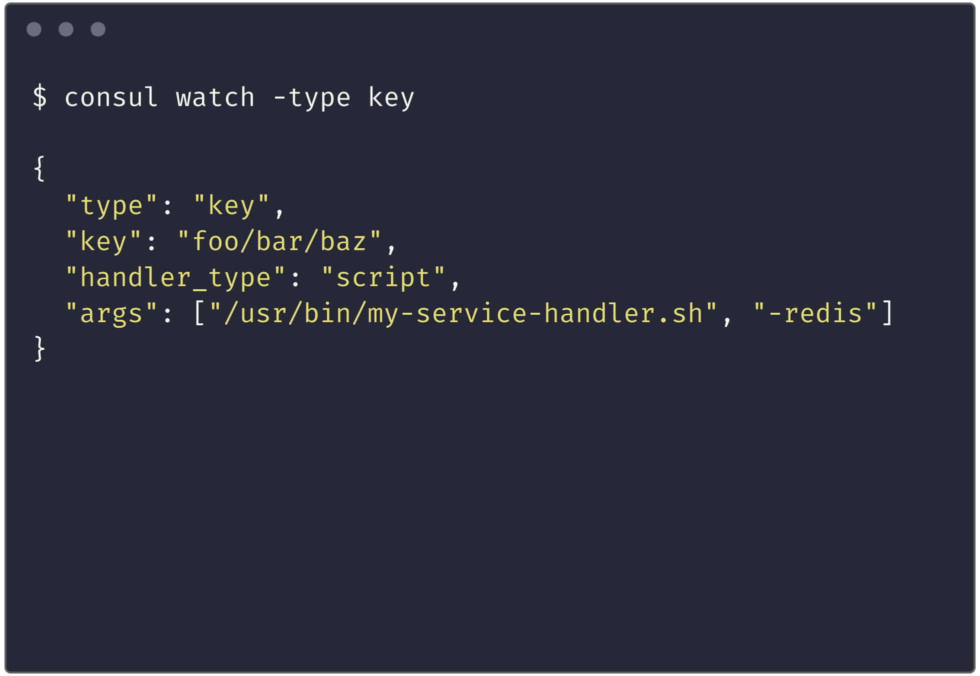
Name	Actions
cache_size	•••
cache_url	•••
conn_pool_size	•••
enable_bar	•••
enable_foo	•••
readonly_mode	•••

Watches

React to changes dynamically

Watches are the simplest way to react to changes using Consul.

- Watch for changes in K/V, services, nodes, health checks, and events
- Invoke external handlers when a change is detected. The handler can be any executable, letting operators customize behavior



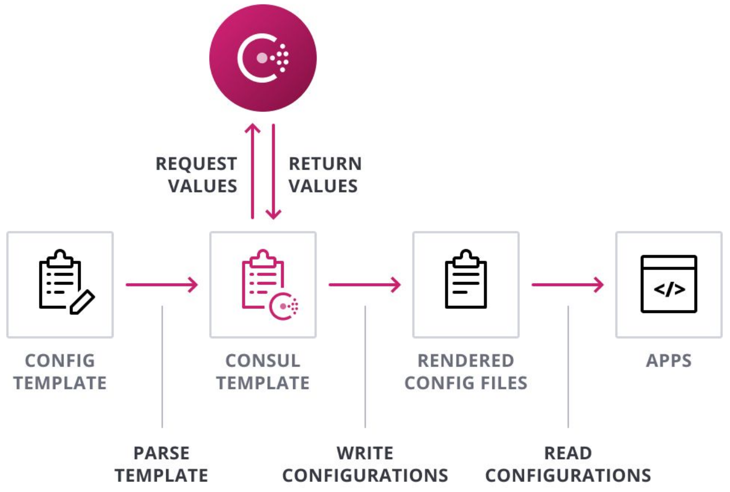
```
$ consul watch -type key

{
  "type": "key",
  "key": "foo/bar/baz",
  "handler_type": "script",
  "args": ["/usr/bin/my-service-handler.sh", "-redis"]
}
```

Consul Template

Link 3rd party config files to Consul K/V

- Standalone application that populates values from Consul and dynamically renders updates to any third party configurations
- Automatically triggers a reload of third party tools when the template is updated



HAProxy - Consul Template

HAPROXY CONFIG

```
backend frontend
  balance roundrobin
  server web1 web1.yourdomain.com:80 check
  server web2 web2.yourdomain.com:80 check
```

CONSUL TEMPLATE

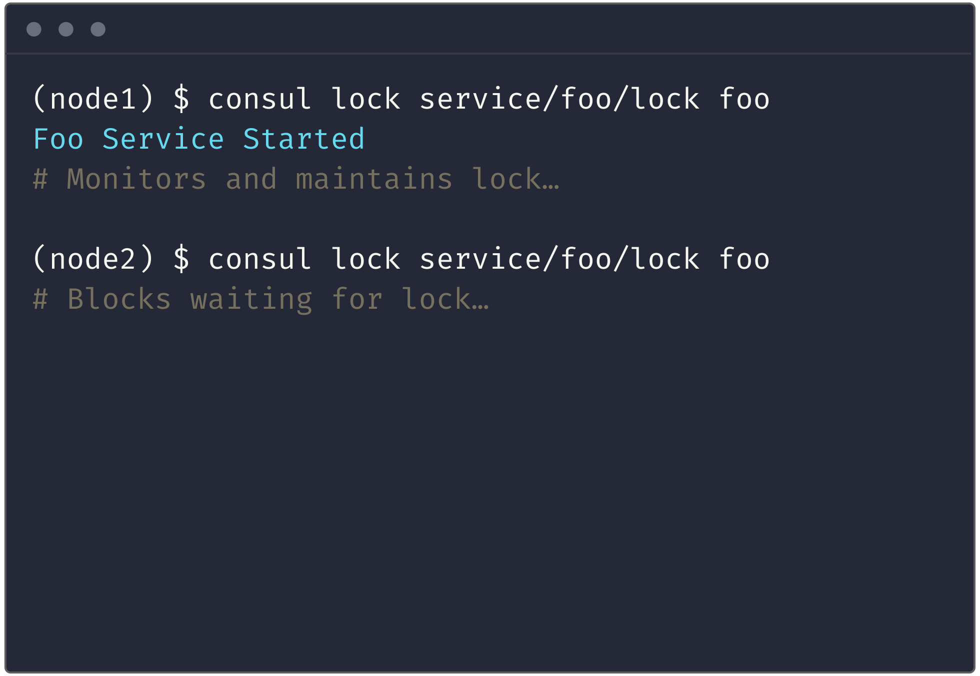
```
backend frontend
  balance roundrobin{{range: "app.frontend"}}
  server {{.ID}} {{.Address}}:{{.Port}}{{end}}
```

```
backend frontend
  server 104.131.148.171:80
  server 104.131.148.171:80
  server 104.131.148.171:80
```

Distributed Locks & Semaphores

Controlling access to a shared resource through mutual exclusion, leader election or semaphores (when N>1)

- Locks built into K/V store
- Client-side leader election
- Baked into Consul's Golang API client library
- Integration with client application via APIs, or zero-touch consul lock command



```
(node1) $ consul lock service/foo/lock foo
Foo Service Started
# Monitors and maintains lock…

(node2) $ consul lock service/foo/lock foo
# Blocks waiting for lock…
```

05

Geo Failover & Prepared Queries

Geo Failover

Scaling to multiple data centers is challenging

- Multi-data center deployments provide redundancy, data locality, scalability, and resiliency
- Failover logic doesn’t exist in all applications or is difficult to add or implement
- Failures require manual updates to load balancers or DNS so that traffic routes to healthy instances and/or data centers

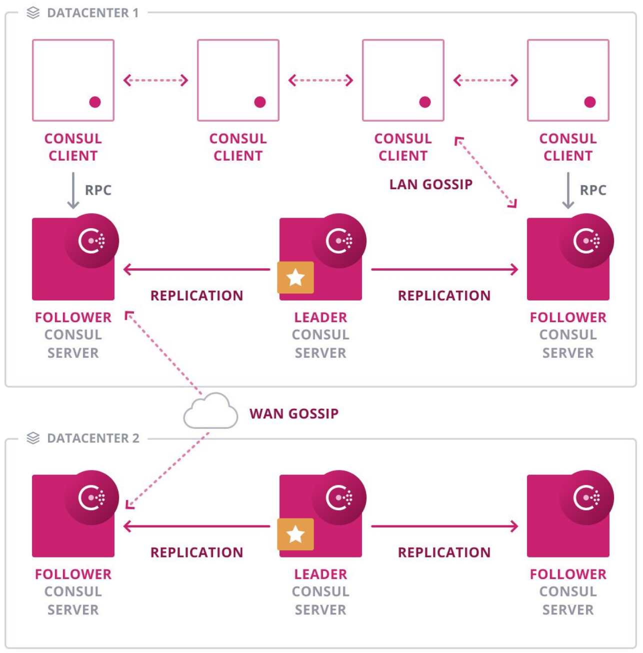
Automated Geo Failover

Consul Federation loosely couples and provides connectivity across multiple data centers

- Data centers are independent and failures do not impact others
- Service lookups locally and/or remote federated data centers

Prepared queries allow for creation and transparent management of complex complex failover policies

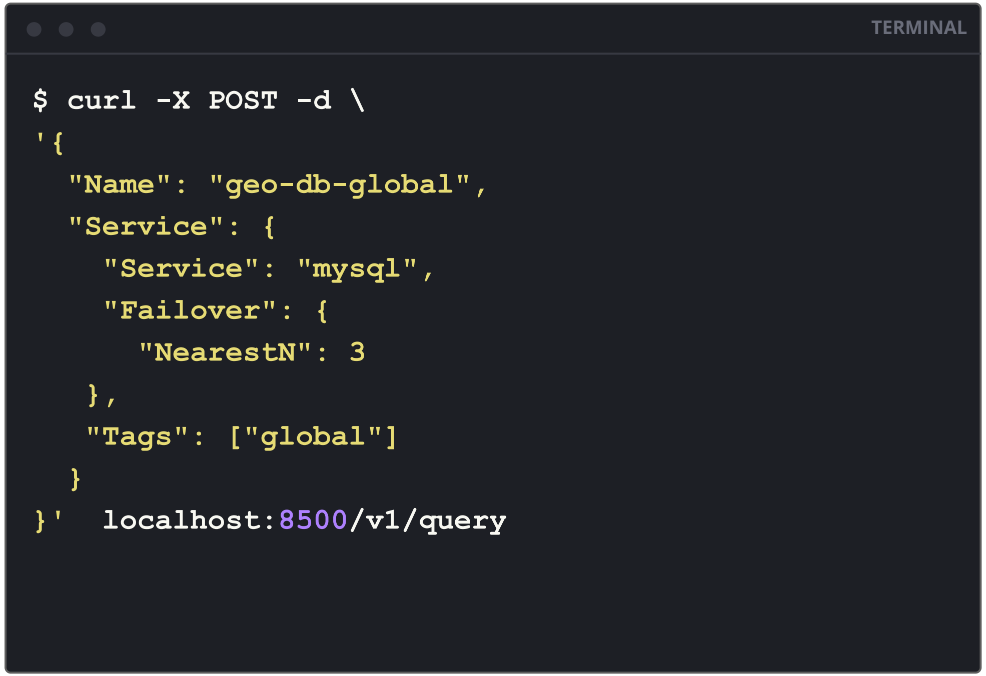
- Are registered in a datacenter-level namespace
- Define which services to look up, and rules for what to do if none are available in the local datacenter
- Can be executed via HTTP or DNS



Prepared Query

Database Template Example

- Applications lookup "`geo-db-global.query.consul`"
- Resolving "`geo-db-global.query.consul`" will return a database service instance with the tag "master" from the local datacenter
- If none are available, it will try the next 3 datacenters in order of increasing RTT

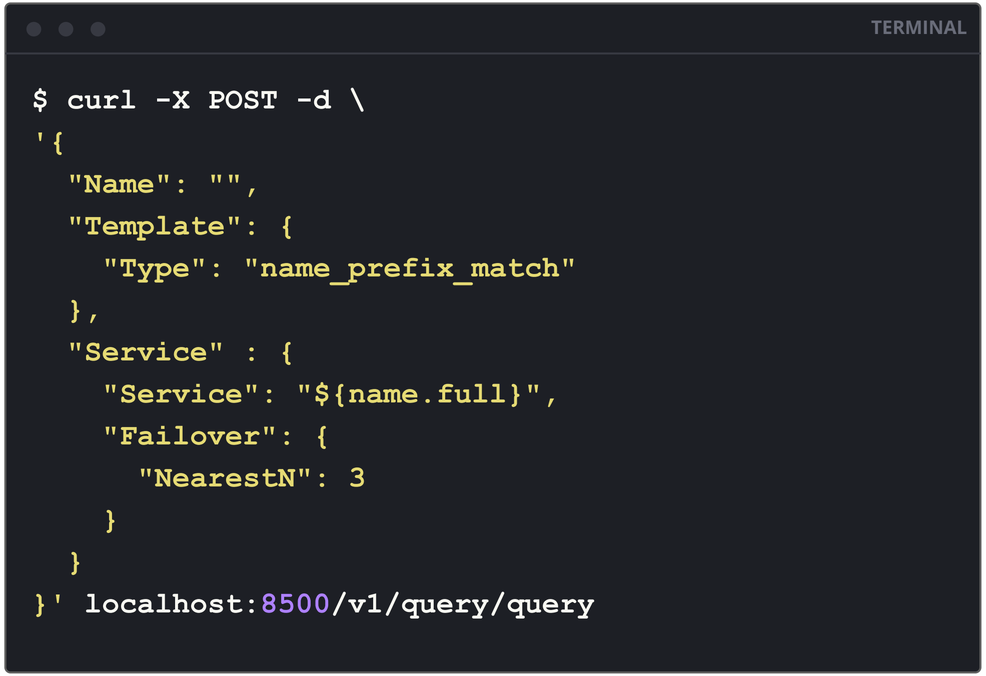

TERMINAL

```
$ curl -X POST -d \
'{
  "Name": "geo-db-global",
  "Service": {
    "Service": "mysql",
    "Failover": {
      "NearestN": 3
   },
   "Tags": ["global"]
  }
}'  localhost:8500/v1/query
```

Prepared Query

Catch All Template Example

- Applications lookup "*.query.consul"
- With a single query template, all services can fail over to the nearest healthy instance in a different datacenter


TERMINAL

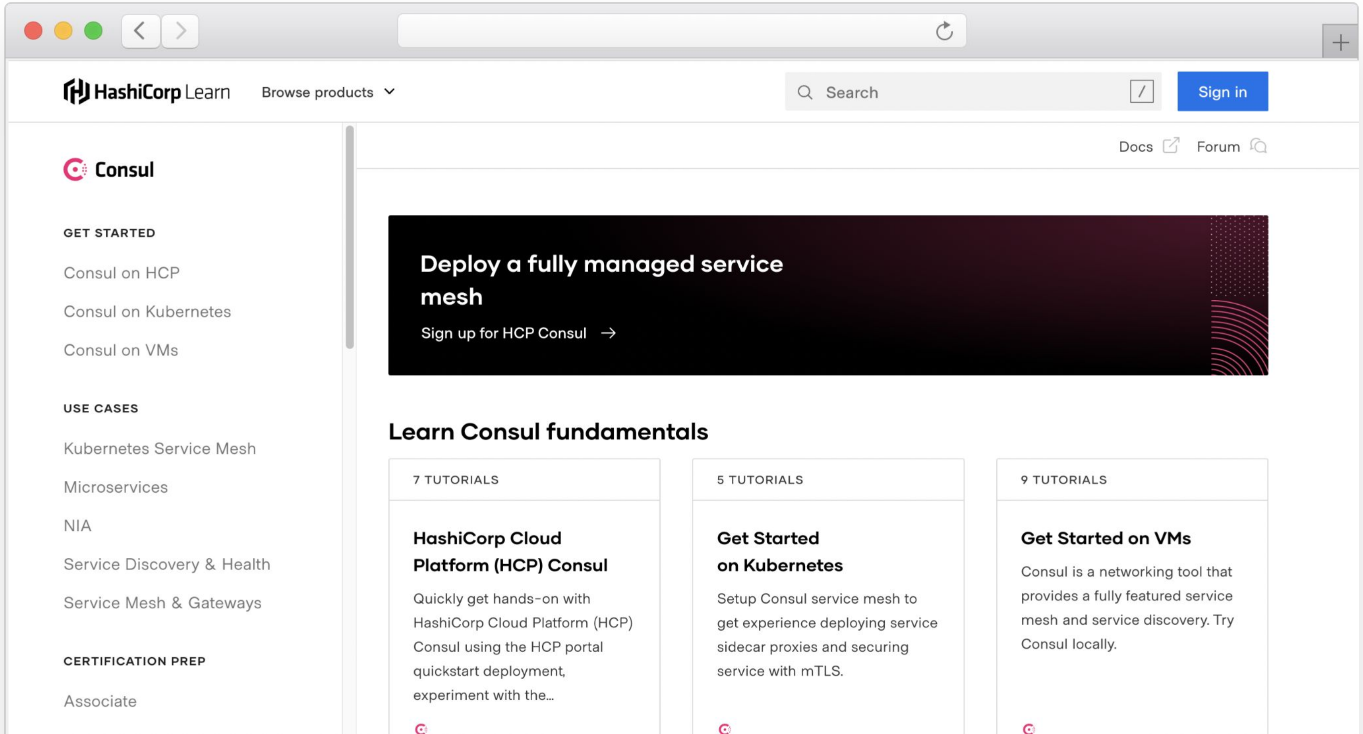
```
$ curl -X POST -d \
'{
  "Name": "",
  "Template": {
    "Type": "name_prefix_match"
  },
  "Service" : {
    "Service": "${name.full}",
    "Failover": {
      "NearestN": 3
    }
  }
}' localhost:8500/v1/query/query
```

06

Next Steps

Learn https://learn.hashicorp.com/consul

Step-by-step guides to accelerate deployment of Consul



GET STARTED

Consul on HCP

Consul on Kubernetes

Consul on VMs

USE CASES

Kubernetes Service Mesh

Microservices

NIA

Service Discovery & Health

Service Mesh & Gateways

CERTIFICATION PREP

Associate

Deploy a fully managed service mesh

Sign up for HCP Consul →

Learn Consul fundamentals

7 TUTORIALS

HashiCorp Cloud Platform (HCP) Consul

Quickly get hands-on with HashiCorp Cloud Platform (HCP) Consul using the HCP portal quickstart deployment, experiment with the...

5 TUTORIALS

Get Started on Kubernetes

Setup Consul service mesh to get experience deploying service sidecar proxies and securing service with mTLS.

9 TUTORIALS

Get Started on VMs

Consul is a networking tool that provides a fully featured service mesh and service discovery. Try Consul locally.

Resources

- Consul Service Registration Tutorial
- Service Definition Documentation
- Consul DNS Documentation
- Health Checks Documentation & Examples
- Consul Agent Documentation
- Consul Template & Load Balancers
- Geo-Failover with Prepared Queries Tutorial

Need Additional Help?

Customer Success

Contact our Customer Success Management team with any questions. We will help coordinate the right resources for you to get your questions answered.

customer.success@hashicorp.com

Technical Support

Something not working quite right? Engage with HashiCorp Technical Support by opening a ticket for your issue at support.hashicorp.com.

Discuss

Engage with the HashiCorp Cloud community including HashiCorp Architects and Engineers

discuss.hashicorp.com

Upcoming Onboarding Webinars

Office Hour

An interactive open forum to discuss specific questions about your environment and Use Cases.

Please bring your questions!

Webinar

Service Mesh & Gateways

Topics include: Patterns and best practices around Implementing Service Mesh, supported proxies and gateways, and using PKI to secure Service Mesh

Office Hour

An interactive open forum to discuss specific questions about your environment and Use Cases.

Please bring your questions!

Q & A

Thank You

customer.success@hashicorp.com
www.hashicorp.com