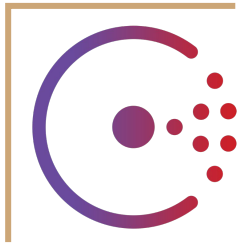


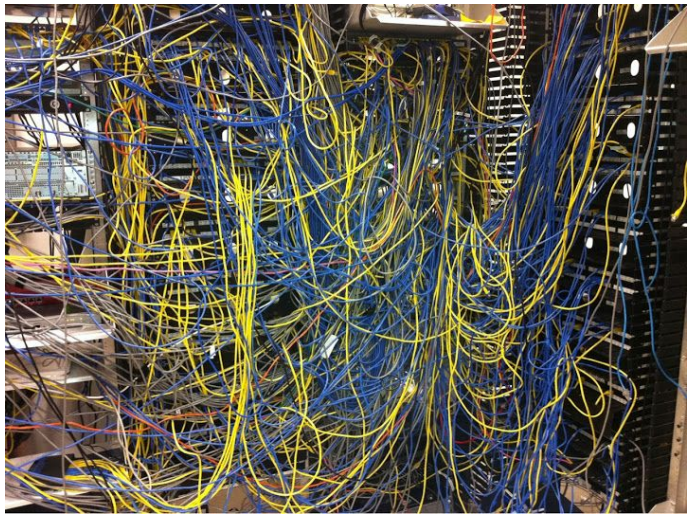
Consul

Service Discovery

Marcin Szuppe e-point S.A.

Czego oczekujemy?

- Systemu monitorującego nasze serwisy, łatwo konfigurowalnego, łatwego w utrzymaniu i uruchamianiu, posiadającego narzędzia do service discovery, konfiguracji serwisów w naszej infrastrukturze.



Istniejące rozwiązania

- ZooKeeper
- etcd
- Nagios - health checking
- Sensu - health checking
- Sky DNS
- Mesos-DNS
- Spartan
- Marathon-lb
- Minuteman
- SmartStack

- Doozerd
- Chef
- Puppet
- Serf
- Eureka

...

- Custom solutions

Istniejące rozwiązania są tylko narzędziami.

Możliwe jest zbudowanie Service Discovery za pomocą istniejących komponentów, ale brak instrukcji, jak to zrobić.

Consul łączy funkcjonalności istniejących narzędzi.

Consul provides a consistent view of services and configuration

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

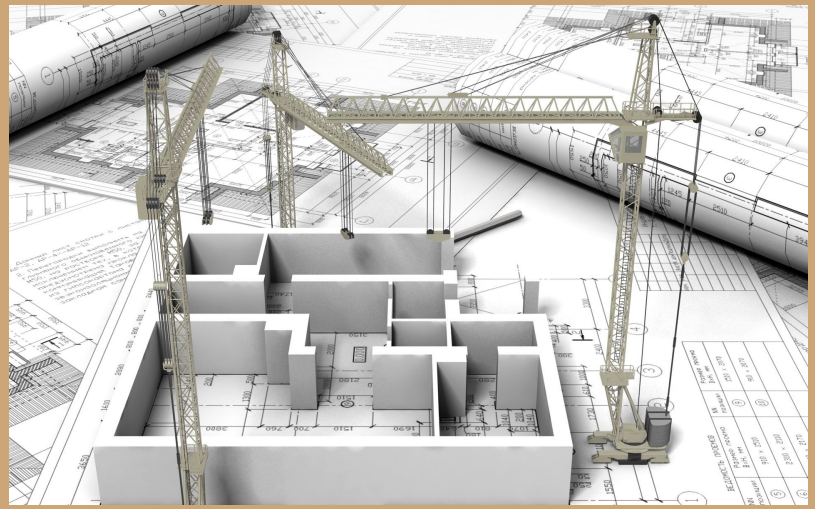
Flexible key/value store for dynamic configuration, feature flagging, coordination, leader election and more. Long poll for near-instant notification of configuration changes.

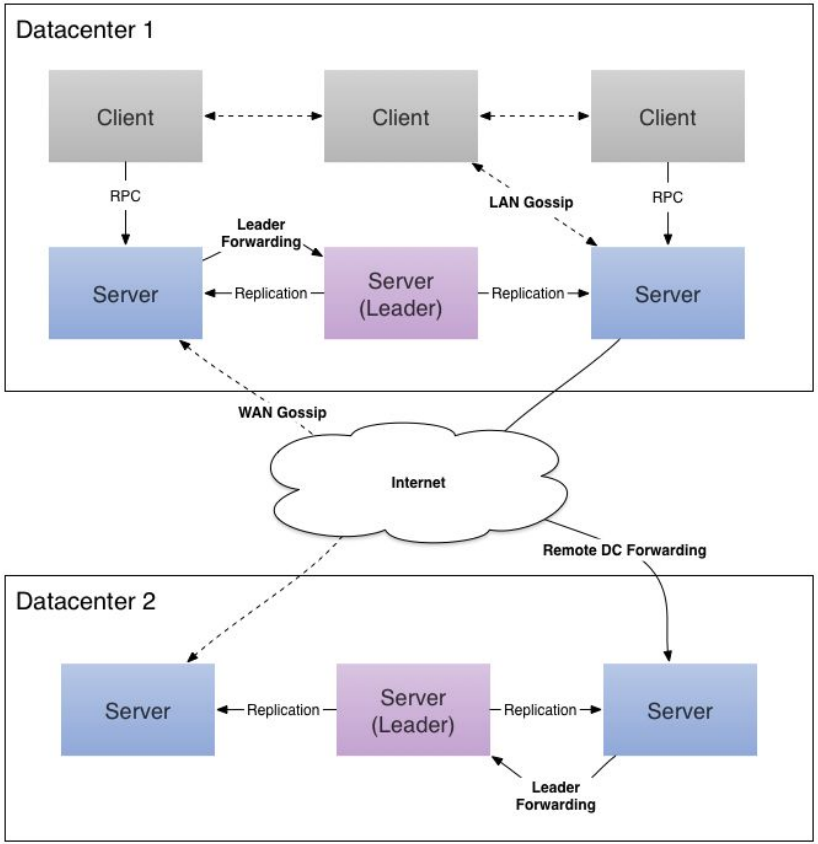
Ficzery

- Service Discovery (przez DNS i REST API)
- Prosta rejestracja serwisów poprzez REST API
- Monitorowanie serwisów oraz kontrola zdrowia serwisów (health check) (rozwiązania wbudowane i customowe)
- Rozproszony hierarchiczny magazyn K/V Store do dynamicznej konfiguracji (oparty na katalogach danych)
- Wsparcie wdrożenia dla wielu centrów danych (DC) (obsługuje pojedyncze DC i może być skalowany w celu obsługi wielu DC)
- Load Balancer (dynamiczne równoważenie obciążenia)
- Orkiestracja kontenerów (eventy)
- Web UI
- Bezpieczeństwo: TLS and ACLs
- Prosta konfiguracja i łatwość uruchomienia

- 3 podstawowe porty:
  - 8400 - RPC
  - 8500 - HTTP
  - 8600 - DNS
-

Architektura




Datacenter 1
Client
Client
Client
RPC
LAN Gossip
RPC
Leader Forwarding
Server
Replication
Server (Leader)
Replication
Server
WAN Gossip
Internet
Remote DC Forwarding
Datacenter 2
Server
Replication
Server (Leader)
Replication
Server
Leader Forwarding

Agent

- Proces (deamon) działający na każdym nodzie klastra konsula
- Agenci są zaagregowani w klastrach dla różnych Centrach Danych, których są świadomi

- W trybie:
  - **klient**
  - **serwer**
- Klient korzysta z usług hostingowych serwera
- Każdy node ma uruchomionego agenta
- Pozostaje w synchronizacji z pozostałymi agentami klastra,
- Komunikuje się poprzez interfejs REST (HTTP) and DHCP (DNS)

Klient

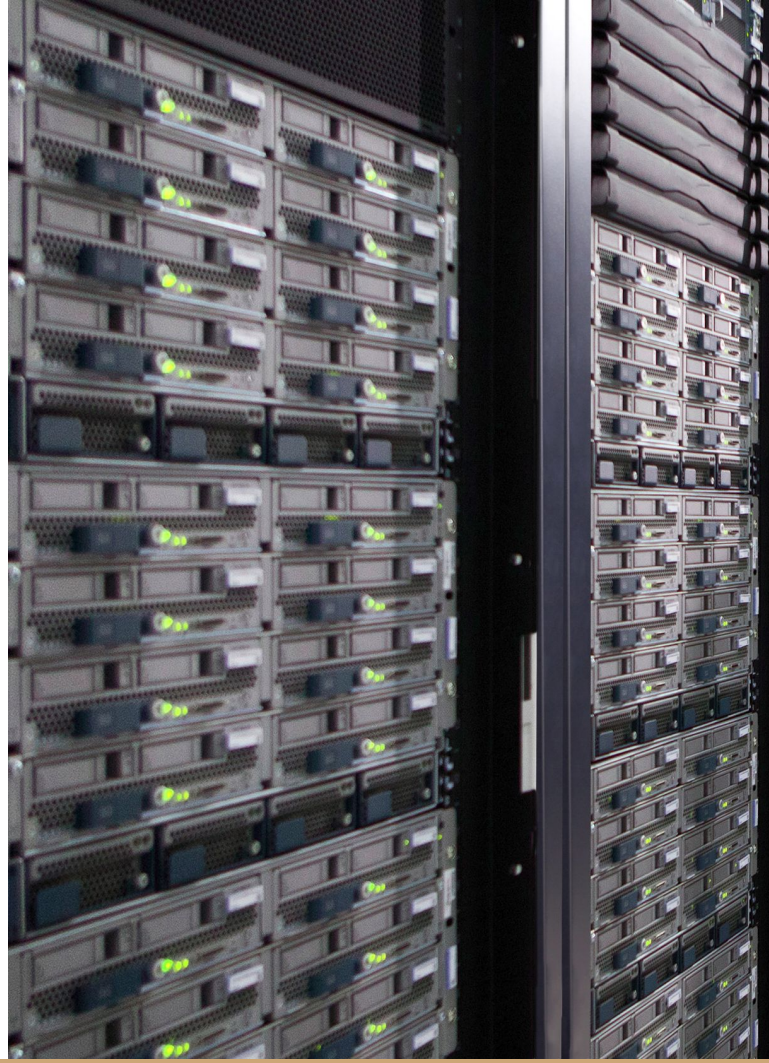
- Lekka odmiana agenta
- Przekazuje żądania do serwera
- Głównie bezstanowy
- Działa w oparciu o protokół LAN gossip (mały ruch)

Serwer

- Agent z większą liczbą zadań
- Jeden lider i fololersy
- Algorytm RAFT quorum do wyboru leadera
- Utrzymuje stan klastra
- Wykorzystuje protokół WAN gossip do komunikacji z innymi DC
- Przekazuje zapytania do lidera
- Przesyła zapytania do innych DC

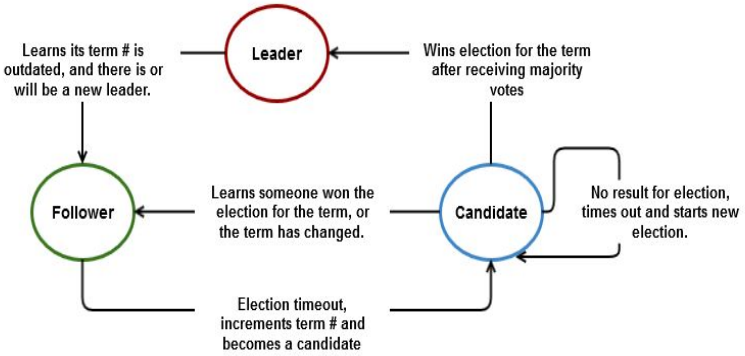
Centrum Danych (DC)

- Środowisko sieciowe
- Prywatne, o małym opóźnieniu i dużej przepustowości
- Duża prędkość komunikacji między węzłami, bez przeskoków, mało routingu



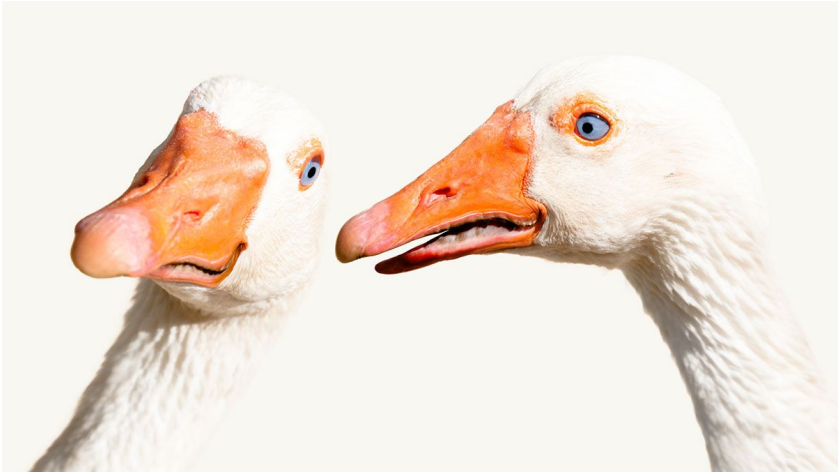
Konsensus

- Porozumienie co do tego, kto jest przywódcą
- Transakcyjna maszyna stanu skończonego
- Replikowany dziennik (log), FSM, zestaw równorzędny (który otrzymuje replikowany dziennik), kworum, zatwierdzony wpis, lider
- Wynik: spójny widok konfiguracji i serwisów
- Algorytm RAFT



Protokół Gossip

- Consul jest zbudowany na Serfie
- Serf jest pełnym protokołem plotkarskim gossip
- Serf zapewnia członkostwo, wykrywanie awarii i mechanizmy rozgłaszania zdarzeń
- Klastrowanie węzłów serwera



Odpowiedzialność Agenta

1. Utrzymywanie:
   a. zestawu usług
   b. sprawdzanie rejestracji usług
   c. informacje o zdrowiu (helath check)
2. Aktualizowanie:
   a. stanu zdrowia (health check)
   b. stanu lokalnego

Katalog

- Wspiera Service Discovery Consula
- Jest katalogiem usług
- Agreguje informacje przesyłane przez agentów
- Utrzymuje widok topologii klastra:
  - dostępne usługi
  - węzły, które uruchamiają te usługi (node = client agent)
  - informacje o zdrowiu
- Usługi i kontrole w katalogu mają mniej informacji niż agent
- Katalog obsługiwany tylko przez węzły agentów serwera
- Katalog jest replikowany za pomocą konsensusu Consula (RAFT)

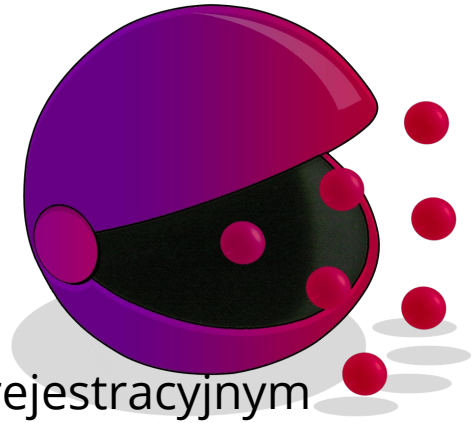
Użycie

```
$ consul
usage: consul [--version] [--help] <command> [<args>]

Available commands are:
    agent          Runs a Consul agent
    event          Fire a new event
    exec           Executes a command on Consul nodes
    force-leave    Forces a member of the cluster to enter the "left" state
    info           Provides debugging information for operators
    join           Tell Consul agent to join cluster
    keygen         Generates a new encryption key
    keyring        Manages gossip layer encryption keys
    leave          Gracefully leaves the Consul cluster and shuts down
    lock           Execute a command holding a lock
    members        Lists the members of a Consul cluster
    monitor        Stream logs from a Consul agent
    reload         Triggers the agent to reload configuration files
    version        Prints the Consul version
    watch          Watch for changes in Consul
```

-server
for server
mode

Service Discovery



Flow rejestracji serwisu:

1. Usługa wywołuje powiadamia klienta komunikatem rejestracyjnym
2. Klient komunikuje rejestrację serwerowi
3. Klient sprawdza stan zdrowia usługi i oznacza healthy / unhealthy oraz przesyła wynik serwerowi
4. Kiedy jest zapytanie o daną usługę, to zwracane są tylko zdrowe usługi

Sposoby rejestracji serwisu

- Rejestracja przez HTTP / plik konfiguracyjny
- Autorejestracja serwisu
- Rejestracja przez trzecią stronę

```
$ curl -X PUT -d '{"Datacenter": "dc1", "Node": "google",
   "Address": "www.google.com",
   "Service": {"Service": "search", "Port": 80}}'
   http://127.0.0.1:8500/v1/catalog/register
```

```
{
  "Name": "service1",
  "address": "10.0.0.12",
  "port": 8080,
  "Check": {
     "http": "http://10.0.0.12:8080/health",
     "interval": "5s"
  }
}
```

Pliki konfiguracyjne

Mogą być przechowywane lokalnie i używane do bootstrapowania konfiguracji Consula

```
{
  "service": {
    "name": "redis",
    "tags": ["master"],
    "address": "127.0.0.1",
    "port": 8000,
    "checks": [
      {
        "script": "/usr/local/bin/check_redis.py",
        "interval": "10s"
      }
    ]
  }
}
```

Health checks

- skrypty odpalane okresowo
- HTTP
- TCP
- TTL
- Docker Container

Kod HTTP:

- 200-299 -> OK
- 429 -> WARNING
- pozostałe -> FAIL

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

```
{
  "check": {
    "id": "api",
    "name": "HTTP API on port 5000",
    "http": "http://localhost:5000/health",
    "interval": "10s",
    "timeout": "1s"
  }
}
```

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

Health checks

Skrypty

- Consul uruchamia skrypty w określonych w konfiguracji interwałach
- Statusy: 0 - zdrowy, 1 - warning, inna liczba - nie działa

TTL

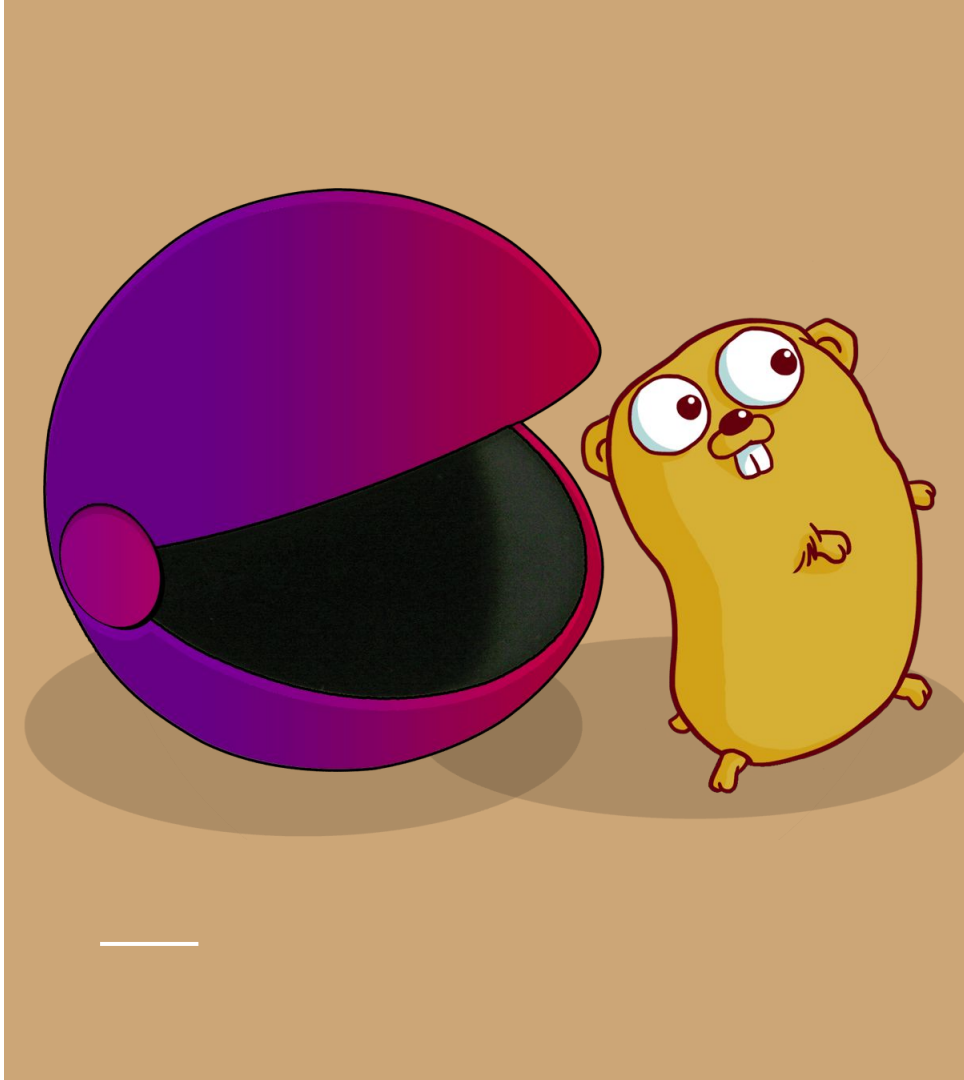
- aplikacja reportuje swój status okresowo
- brak raportu oznacza, że aplikacja nie działa

Możliwość usuwania niedziałających nodów

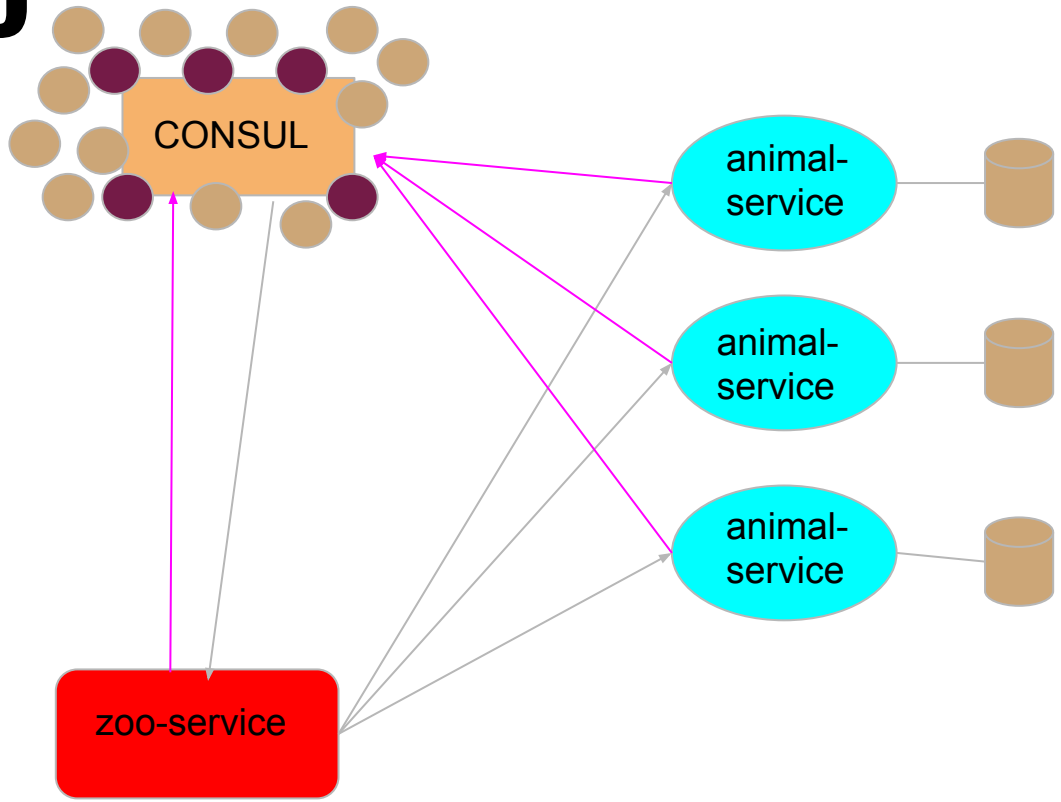
HTTP API

- K/V
- agent
- catalog
- health check
- session
- acl
- event
- status

DEMO



e-point ZOO


CONSUL
animal-service
animal-service
animal-service
zoo-service

Uruchamiamy Consula.

```
consul agent -config-file=${HOME}/work/prezentacja_consul/consul/server1-config/server1.json -bind=127.0.0.1

consul agent -config-file=${HOME}/work/prezentacja_consul/consul/server2-config/server2.json -bind=127.0.0.1

consul agent -config-file=${HOME}/work/prezentacja_consul/consul/client-config/client.json -bind=127.0.0.1

consul info

consul members

curl -s http://localhost:8500/v1/agent/members | jq .[]

curl -s http://localhost:8500/v1/catalog/datacenters | jq .[]

curl -s http://localhost:9500/v1/catalog/datacenters | jq .[]
```

Zapytania

DNS:

```
dig @127.0.0.1 -p 8600 animal-service.service.consul
```

```
dig @127.0.0.1 -p 8600 animal-service.service.consul SRV
```

HTTP:

```
curl 127.0.0.1:8500/v1/agent/services
```

```
curl http://localhost:8500/v1/catalog/service/animal-service
```

```
curl -s http://localhost:8500/v1/health/service/animal-service | jq .[]
```

```
curl 'http://localhost:8500/v1/health/service/animal-service?passing'
```

Rejestracja serwisu z palca

```
curl -H PUT http://localhost:8500/v1/agent/service/register -d @register_redis.json

curl -H PUT http://localhost:8500/v1/agent/service/register -d @register_health.json

curl -s http://localhost:8500/v1/agent/services | jq .[]

curl -s http://localhost:8500/v1/health/service/redis | jq .[]

curl http://localhost:8500/v1/agent/check/pass/service:redis1

curl -H PUT http://localhost:8500/v1/agent/service/deregister/redis1
```

Properties	Consul	Etcd	Zookeeper
User Interface	Available		
RPC	Available	Available	
Health Check	HTTP API	HTTP API	TCP
Key Value	3 Consistency modes	Good Consistency	Strong Consistency
Token System	Available		
Language	Golang	Golang	Java

DNS server + K/V Store + ZooKeeper + Nagios + … = Consul

https://www.consul.io

mszuppe@e-point.pl