Enterprise Architecture

Service Discovery

What do you know about Service Discovery?

Service discovery Back Ends

- Stand alone registry
  - Spring supported :
    - Consul (+ registrator for containers)
    - Eureka
  - ZooKeeper, Etcd, ...
- Platform
  - Kubernetes service catalog
    - DNS – service LB based
  - Istio “Service mesh”
    - Control Plane

Feature Overlap!

Capabilities				Spring Cloud with Kubernetes & Istio on IaaS+
DevOps Experience			Kubernetes	Self service, multi-environment capabilities
Auto Scaling & Self Healing			Kubernetes	Pod/Cluster Autoscaler, HealthIndicator, Scheduler, Pool Ejection
Deployment & Scheduling		Istio	Kubernetes	Deployment strategy, DarkLaunch, A/B, Canary, Scheduler Strategy
Resilience & Fault Tolerance	Spring Cloud	Istio	Kubernetes	HealthIndicator, Hystrix, HealthCheck, Process Check, Circuit Breaker/Timeout/Retry
Distributed Tracing	Spring Cloud	Istio	Kubernetes	Zipkin, Jaeger
Centralized Metrics	Spring Cloud	Istio	Kubernetes	Heapster, Prometheus, Grafana
Centralized Logging	Spring Cloud	Istio	Kubernetes	EFK
API Gateway	Spring Cloud	Istio	Kubernetes	Zuul, Traffic Control, Egress
Load Balancing	Spring Cloud	Istio	Kubernetes	Ribbon, Service, Envoy
Chaos Engineering	Spring Cloud	Istio	Kubernetes	Chaos Monkey for Spring Boot, Chaos Toolkit Kubernetes, Envoy
Service Discovery	Spring Cloud	Istio	Kubernetes	Service
Configuration Management	Spring Cloud		Kubernetes	Externalized Configuration, ConfigMaps, Secrets
Application Packaging	Spring Cloud		Kubernetes	Spring Boot Maven/Gradle plugin
Job Management	Spring Cloud		Kubernetes	Spring Batch, Scheduled Jobs
Process Isolation			Kubernetes	Docker, Pods, Envoy
Environment Management			Kubernetes	Namespaces, Authorization
Resource Management			Kubernetes	CPU and Memory Limits, Namespace resource quotas
Operating System	IaaS+			Ubuntu, Atomic, ...
Virtualization	IaaS+			VMWare, Openstack, ...
Hardware, Storage, Networking	IaaS+			AWS, GCP, Azure, ...

https://mrumpf.github.io/spring-cloud-on-k8s/

OK, back to the basics

- What’s container orchestration
  - What’s Kubernetes
    - https://www.youtube.com/watch?v=R-3dfURb2hA

What’s a service mesh

- Control plane , add proxy/side car to a service
  - Communication now goes through the proxy
  - Define policies
    - Routing, security/ACL , traffic management, telemetry , even service discovery,...

Show me how?

- Building cloud-native applications with Kubernetes and Istio, by Kelsey Hightower
  - https://www.youtube.com/watch?v=6BYq6hNhceI

Architecture implications

- Moving cross-cutting logic from services → Implement at platform level (k8s+istio)
  - No need to integrate with a stand alone discovery backend (consul,eureka ,..)
  - Remove circuit breaking,retries, client LB logic from services
  - Telemtry, ACL ,.. : dynamic and no need to change services “almost free”
  - Flexibility: not just java services or framework with integration with certain back ends

Architecture implications

- Cons:
  - Complexity , learning curve
  - More dependence on the deployment environment → less flexibility to change later

Still, we need DIY

- e.g. Deploy a dedicated Registry (Consul) cluster on top of platform (e.g. k8s)
  - Works best for legacy/Java Microservices
  - Offers more flexibility → no lock in platform
  - Cons:
    - Apps need to be configured
- Check this out:
  - https://luizkowalski.net/deploying-spring-cloud-netflix-apps-on-kubernetes/

OK, Let’s see a sample service registry “Consul”

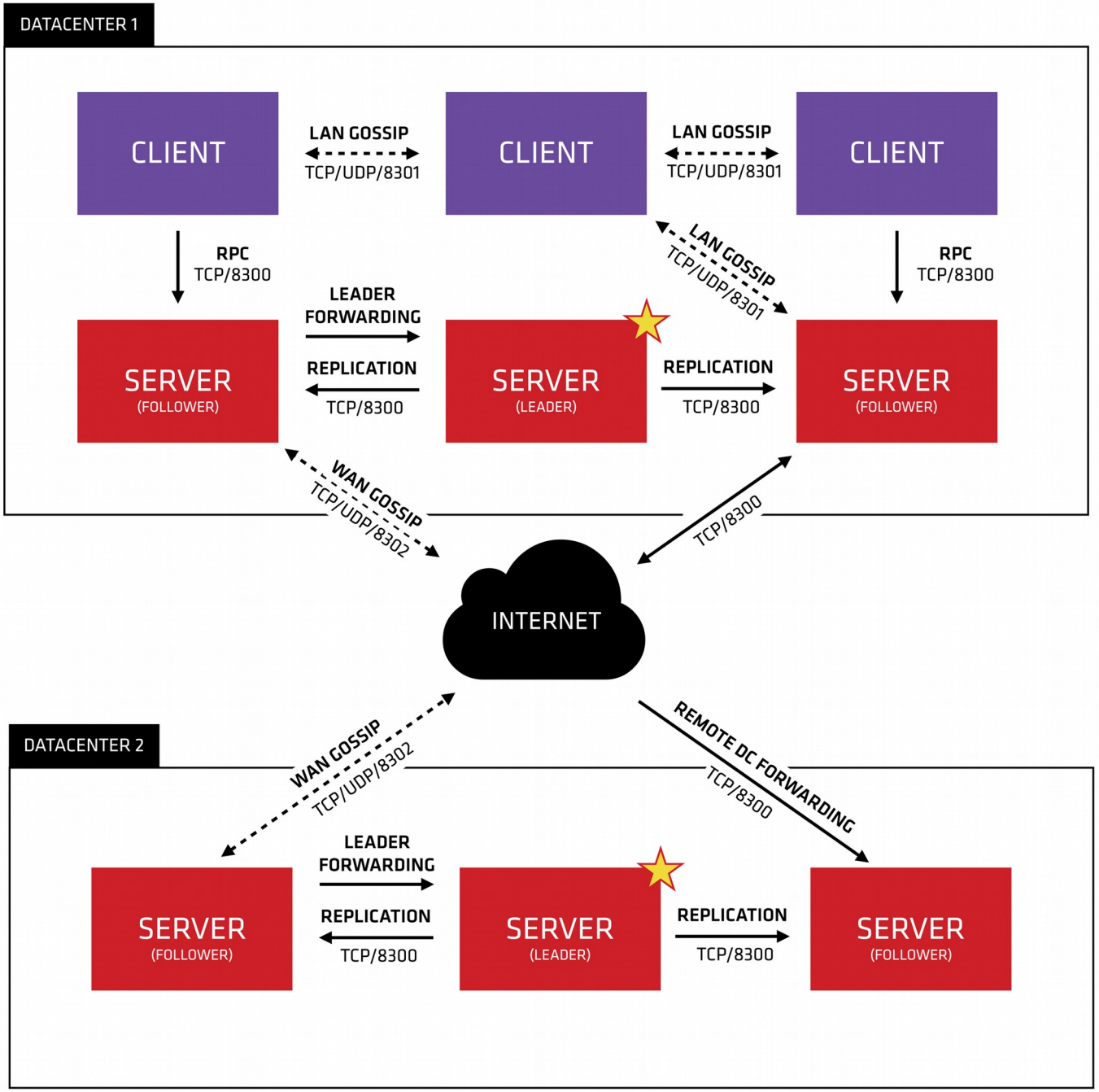
Confused?

- Check Consul vs other:
  - https://www.consul.io/intro/vs/index.html
- Checkout:
  - https://attx-project.github.io/Service-Discovery-Solutions.html

Service Discovery
Use case : Consul

Why we need it

- Let’s hear the creator
  - Introduction to HashiCorp Consul | Armon Dadgar
    - https://www.youtube.com/watch?v=mxeMdl0KvBI



HashiCorp

https://www.consul.io/docs/internals/architecture.html

What’s going on?

- How nodes inside DC find each other?
- How leader election works?
- How followers consent on the values?
- How nodes between 2 Data centers connect?
- What’s stored in each DC?
  - Hint: Data is different, but request can be forwarded

how a node knows about other nodes ?

But how a node knows about other nodes ?

Gossiping!

Gossip protocol

- Also known as epidemic protocol
- A family of protocol styles and implementations
- Consul is based on serf library which implements **SWIM** protocol
  - **S**calable **W**eakly-consistent **I**nfection-style Process Group **M**embership Protocol

SWIM Gossiping

- Got plenty of time, read the paper:
  - http://www.cs.cornell.edu/info/projects/spinglass/public_pdfs/swim.pdf

Consul Gossiping

- LAN:
  - All Nodes participate (server and clients)
- WAN:
  - Across Internet between data centers
  - Only servers participate

How they elect a master and consent on values?

How they elect a master and consent on values?

Consensus Protocol

Raft

- In CAP terms : Strong consistency (CP)
- Solves distributed consensus problem
- Don’t expect high throughput
  - Implemented in systems like
    - ZooKeeper , Consul , chubby ...

Raft

- Let’s see how it works
  - http://thesecretlivesofdata.com/

Raft In consul

- Only server nodes keeps state (not clients)
- Limit # servers
  - Raft communication is noisy
  - Typically odd # server , e.g. 3,5 to maintain majority

Distributed configuration

- Consul
- Spring cloud config
  - Default storage is git, among many other
- Etcd
- ...

Final Thoughts

- Now we can register services and discover them
  - By HTTP or DNS (SRV records)
- Monitor health checks!
- Also store K/V for distributed configuration!
  - Treat profiles as hierarchy