Microservice with Consul, Apache httpd : Introduction

Introduction
Essential contents of the chapter are:

- ✓ **Consul is a very powerful service discovery technology**. If you remember Netflix stack, there I discussed about Eureka, right? So, consul is another choice of Eureka for better features
- ✓ Apache httpd can be used as a load balancer and router for HTTP requests in a microservices system.
- ✓ Consul Template can create a configuration file for the Apache httpd server that includes information about all registered microservices. Consul Template configures and restarts Apache httpd when new microservice instances are started.

Consul is a product of the company Hashicorp, which offers various products in the field of microservices and infrastructure.

Of course, Hashicorp also offers commercial support for Consul.

License & Technology
Consul is an open-source product. It is written in Go Language and licensed under Mozilla Public License 2.0.

The open-source code is available at GitHub (https://github.com/hashicorp/consul)

Consul provides several key features:

Multi-Datacenter - Consul is built to be datacenter aware and can support any number of regions without complex configuration.

Service Mesh/Service Segmentation - Consul Connect enables secure service-to-service communication with automatic TLS encryption and identity-based authorization. Applications can use sidecar proxies in a service mesh configuration to establish TLS connections for inbound and outbound connections without being aware of Connect at all.

Service Discovery - Consul makes it simple for services to register themselves and to discover other services via a DNS or HTTP interface. External services such as SaaS providers can be registered as well.

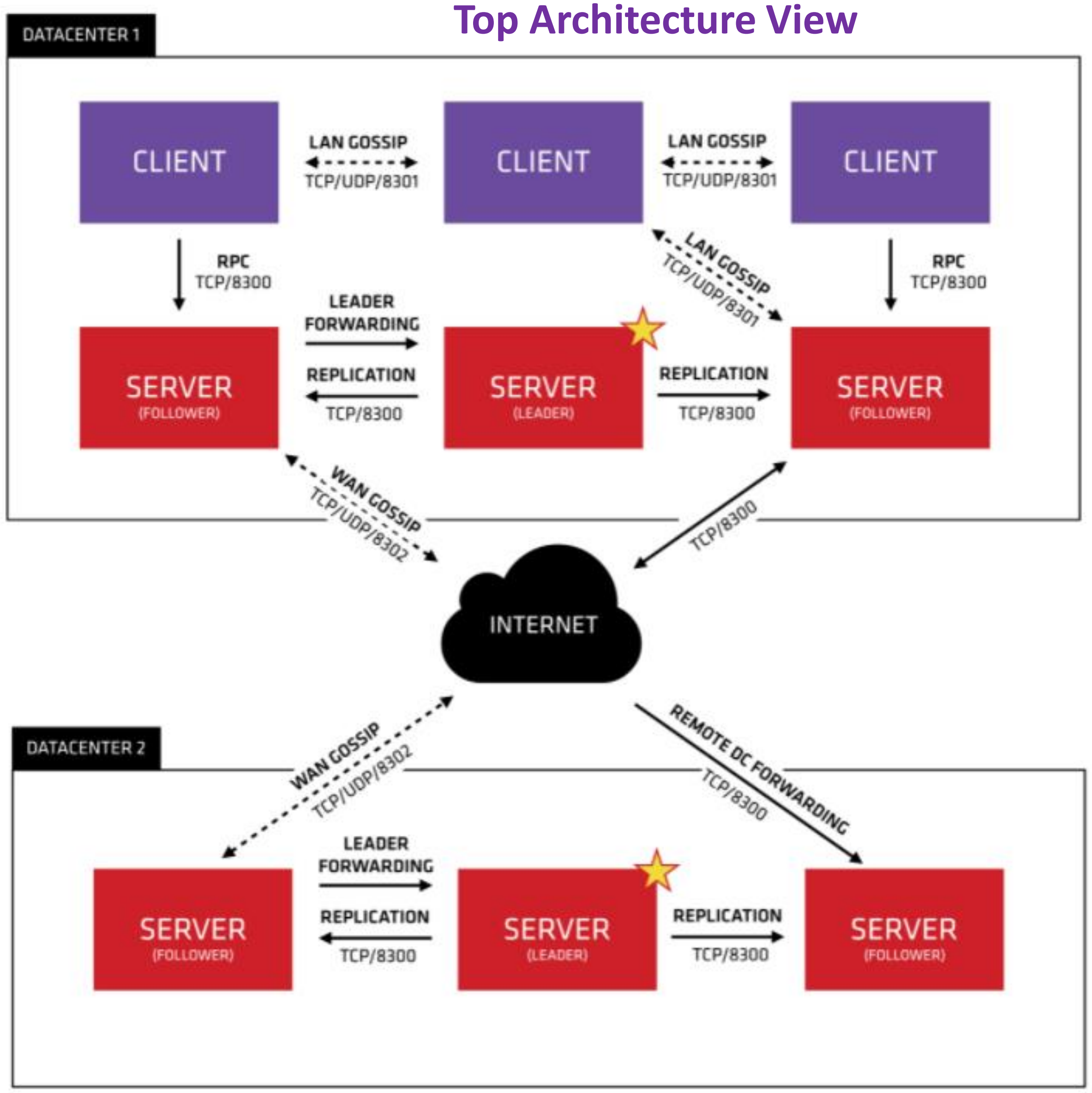
Health Checking - Health Checking enables Consul to quickly alert operators about any issues in a cluster. The integration with service discovery prevents routing traffic to unhealthy hosts and enables service level circuit breakers.

Key/Value Storage - A flexible key/value store enables storing dynamic configuration, feature flagging, coordination, leader election and more. The simple HTTP API makes it easy to use anywhere.

Consul runs on Linux, macOS, FreeBSD, Solaris, and Windows and includes an optional browser-based UI. A commercial version called Consul Enterprise is also available.

For Documentation, Please refer : https://www.consul.io/docs

Top Architecture View



Rest with Consul, Apache httpd : Example

Example:
Let’s continue with the same example discussed in my last article over Netflix stack, In this Example we started with three microservices(Customer service, Order Service, and Catalog Service).

- ✓ The catalog microservice manages the information such as price & name for the items
- ✓ The customer microservice stores customer data.
- ✓ The order microservice can accept new orders. It uses the catalog and customer microservice.

The example in this chapter uses Consul for service discovery and Apache httpd server for routing the HTTP requests.

Building the example
download the code with git clone https://github.com/ewolff/microservice-consul.git.

Then the code must be translated with ./mvnw clean package (macOS, Linux) or mvnw.cmd clean package (Windows) in directory microservice-consul-demo.

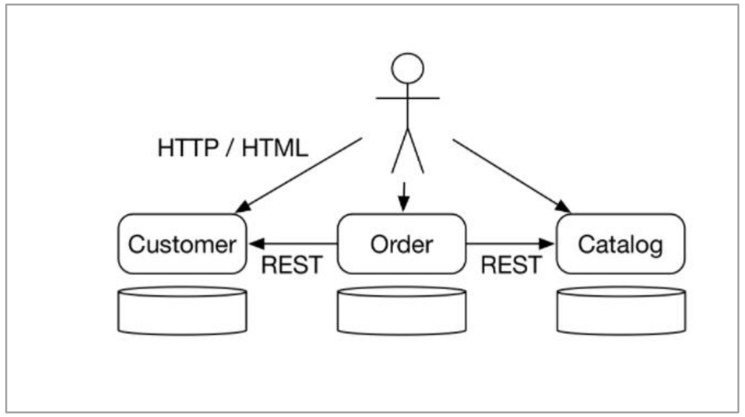
the Docker containers can be built in the directory docker with docker-compose build and started with docker-compose up -d.

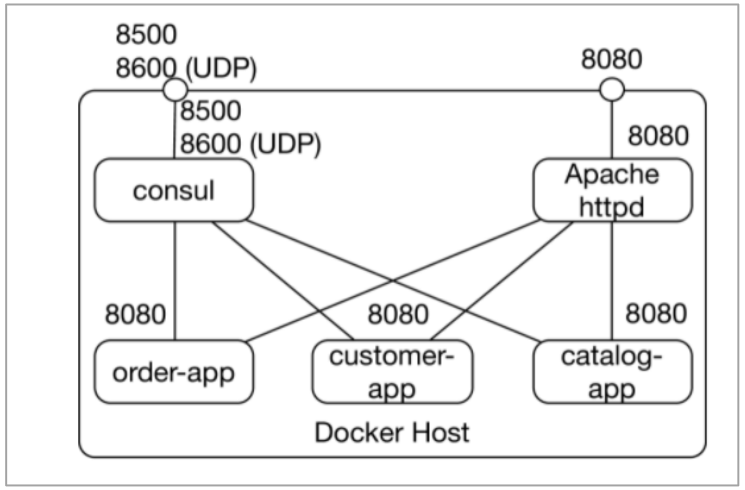
When the Docker containers are running on the local computer, the following URLs are available:

http://localhost:8500 is the link to the Consul dashboard.

http://localhost:8080 is the URL of the Apache httpd server. It can display the web UI of all microservices.

Detail Information :
https://github.com/ewolff/microservice-consul/blob/master/HOW-TO-RUN.md describes the necessary steps for building and running the example in detail.





Docker host architecture view

Service Discovery : Distinguishing Features

This configuration is unsuitable for a production environment because data can be lost, and a failure of the Consul Docker container would bring the entire system to a standstill. In production, a cluster of Consul servers should be used, and the data should be stored persistently.

Consul is a service discovery technology that ensures microservices can communicate with each other.

Distinguishing Feature of Consul

Consul has some features that set it apart from other service discovery solutions.

1. **HTTP REST API & DNS support**
Consul has an HTTP REST API and supports DNS.

- ✓ DNS (Domain Name System) is the system that maps host names such as www.innoq.com to IP addresses on the Internet.
- ✓ In addition to returning IP addresses, it can return ports at which a service is available.
- ✓ This is a feature of the SRV DNS records.

2. **Configuration File Generation**
Using consul template, you can generate config file. For template you can refer the link (https://github.com/hashicorp/consul-emplate/blob/master/config/template.go)

- ✓ The files may contain IP addresses and ports of services registered in Consul.
- ✓ Consul Template also provides Consul's service discovery to systems that cannot access Consul via the API.
- ✓ The systems must use configuration file, which they often already do anyway.

3. **Health Check**
Consul can perform health checks and exclude services from service discovery when the health check fails.

There are few key cases where health check is very important, like
- ✓ For example, a health check can be a request to a specific HTTP resource to determine whether the service can still process requests.
- ✓ A service may still be able to accept HTTP requests, but it may not be able to process them properly due to a database failure.
- ✓ The service can signal this through the health check.

4. **Replication**
Consul supports replication and can ensure high availability.

If a Consul server fails, other servers with replicated data take over and compensate for the failed server.

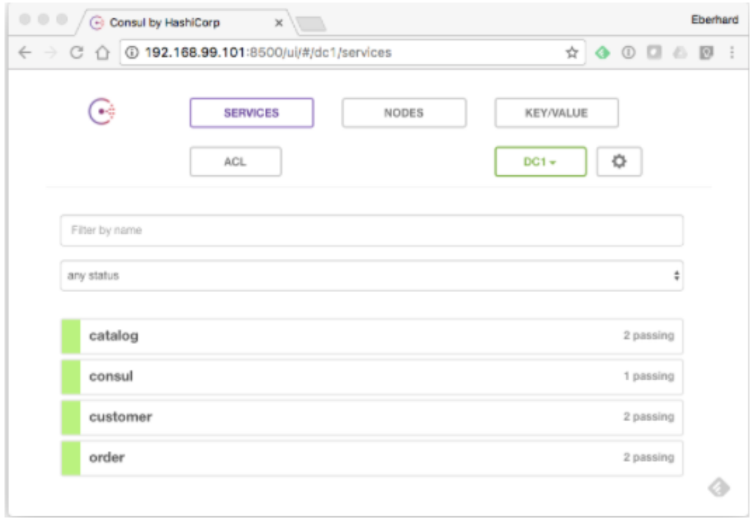
5. **Multiple data centres**
Consul also supports multiple data centers.

- ✓ Data can be replicated between data centers to further increase availability and protect Consul against the failure of a data center.
- ✓ **The search for services can be limited to the same data center.**
- ✓ Services in the same data center usually deliver higher performance.

6. **Service configuration**

Consul can be used not only for service discovery, but for the configuration of services.

Consul dashboard



catalog 2 passing
consul 1 passing
customer 2 passing
order 2 passing

Reverse Proxy, Load balancing : Apache httpd

The **Apache httpd server** is one of the most widely used web servers.
There are modules that adapt the server to different usage scenarios. In the example, modules are configured that turn Apache httpd into a reverse proxy.
Let's understand that **how Apache httpd working as a reverse proxy.**

Reverse Proxy
For easy explanation, Proxy can be used to process the traffic from a network to the outside, a reverse proxy is a solution for inbound network connections.

It has also capability to forward some external request to some specific services. This means that the entire microservice system can be accessible under one url but can use different microservices internally.
Load balancer
Apache httpd serves as a load balancer by distributing network traffic to make the application scalable.

In the example, there is only one Apache httpd, **that functions simultaneously as a reverse proxy and a load balancer for requests from the outside**.

The requests the microservices send to each other are not handled by this load balancer. **For the communication between the microservices, the library Ribbon is used, as we already saw in the Netflix**

It is also possible to write a library that distributes requests to other microservices to different instances. This library must read the currently available microservice instances from the service discovery and then, for each request, select one of the instances. This is how Ribbon Ribbon works.

Syntax for Ribbon Api to create url with host IP and port is like below

```
private LoadBalancerClient loadBalancer;
  // Spring injects a LoadBalancerClient
ServiceInstance instance = loadBalancer.choose("CUSTOMER");
url = String.format("http://%s:%s/customer/",
 instance.getHost(), instance.getPort());
```

So, in continuation of discussion about Load balancing,
There are few more options here like nginx(NG-INX), and Fabio. You need to explore more to understand that how these are different than Apache httpd and do we have any additional feature if we go for nginx or Fabio ?

Apache	NGINX
Apache runs on all Unix like systems such as Linux, BSD, etc. as well as completely supports Windows.	Nginx runs on modern Unix like systems; however it has limited support for Windows.
Apache uses a multi-threaded approach to process client requests.	Nginx follows an event-driven approach to serve client requests.
Apache cannot handle multiple requests concurrently with heavy web traffic.	Nginx can handle multiple client requests concurrently and efficiently with limited hardware resources.
Apache processes dynamic content within the web server itself.	Nginx can't process dynamic content natively.
Apache is designed to be a web server.	Nginx is both a web server and a proxy server.
Modules are dynamically loaded or unloaded, making it more flexible.	Since modules cannot be loaded dynamically, they must be compiled within the core software itself.
A single thread can only process one connection.	A single thread can handle multiple connections.
The performance of Apache for static content is lower than Nginx.	Nginx can simultaneously run thousands of connections of static content two times faster than Apache and uses little less memory.

DNS & Registration

If you want to register the microservice using consul API, you will need some code for this process. But if you want to register the Docker Container with Consul than that would not require any coding activity. Why so, because Architecture is promoting the Docker container for Microservice which eventually making more convenient way if managing microservice and their dependencies.

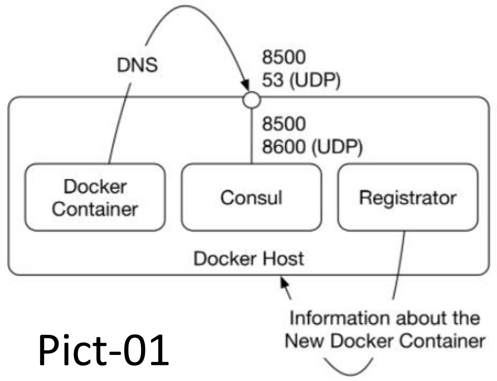
This is possible because it helps to manage your container more efficiently. When the Docker containers are configured in such a way that they use Consul as a DNS server, the lookup of other microservices can also occur without code dependencies.

The drawing at right(Pict-01) shows an overview of the approach.

✓ Registration runs in a Docker container. Via a socket, Registration collects information from the Docker daemon about all newly launched Docker containers. The Docker daemon runs on the Docker host and manages all Docker containers.

✓ The DNS interface **of Consul is bound to the UDP port 53 of the Docker host**. This is the default port for DNS.

✓ **The Docker containers use the Docker host as a DNS server.**



Pict-01

Configuring DNS access
The dns setting in the docker-compose.yml is configured to use the IP address in the environment variable CONSUL_HOST as the IP address of the DNS server. Therefore, the IP address of the Docker host must be assigned to CONSUL_HOST before starting docker-compose.

Unfortunately, it is not possible to configure DNS access from the Docker containers in a way that does not use this environment variable.

Consul
Registration registers every Docker container started with Consul; not only the microservices, but the Apache httpd server or Consul itself can be found among the services in Consul.

Consul registers the Docker containers with .service.consul added to the name. In docker-compose.yml, dns_search is set to .service.consul so that this domain is always searched. In the end, the order microservice uses the URLs http://msconsuldns_customer:8080/ and http://msconsuldns_catalog:8080/ to access the customer and catalog microservices.

Service meshes

Service meshes provide a lot of useful features for resilience, monitoring, tracing, and logging. Istio is an example of a service mesh.

A service mesh injects proxies into the communication between the microservices. Istio supports Consul to achieve that.

With Istio, Consul can be extended to become a **complete platform for the operation** of microservices.