Experiments

In this lesson, we'll look at some experiments that can be conducted.

WE'LL COVER THE FOLLOWING

- Supplement with additional microservice & without DNS
- Supplement with additional microservice & with DNS
- Run on a cluster
- Configure Spring Boot
- Replace Apache httpd with another server
- Scaling & load balancing

Supplement with additional microservice & without DNS

Supplement the Consul system without DNS with an additional microservice.

- A microservice that is used by a call center agent to create notes for a call can be used as an example. The call center agent should be able to select the customer.
- You can copy and modify one of the existing microservices.
- Register the microservice in Consul.
- Ribbon has to be used to call the customer microservice and does the lookup of the microservice in Consul. Otherwise, the microservice must be searched explicitly in Consul.
- Package the microservice in a Docker image and reference the image in `docker-compose.yml`. You can also specify the name of the Docker container.

- Create a link in `docker-compose.yml` from the container with the new service to the container `consul`.
- The microservice must be accessible from the homepage. To do this, you have to create a link in the file `index.html` in the Docker container `apache`. Consul Template automatically sets up the routing for the microservice in Apache as soon as the microservice is registered in Consul.

Supplement with additional microservice & with DNS

Supplement the DNS Consul system (see DNS and Registrator) with an additional microservice.

- A microservice that is used by a call center agent to create notes for a call can be used as an example. The call center agent should be able to select the customer.
- The call to the customer microservice has to use the host name `msconsuldns_customer`.
- You can copy and modify one of the existing microservices.
- A registration in Consul is not necessary since Registrator automatically registers each Docker container.
- Package the microservice in a Docker image and reference the image in `docker-compose.yml`. There you can also specify the name of the Docker container.
- Configure the DNS server for the new microservice in `docker-compose.yml` similar to the other microservices.
- The microservice must be accessible from the homepage. To do this you have to create a link in the file `index.html` in the Docker container `apache`. Consul Template automatically sets up the routing for the microservice in Apache as soon as the microservice is registered in Consul.

Run on a cluster

Currently, the Consul installation is not a cluster and is therefore unsuitable for a production environment. Change the Consul installation so that Consul runs in the cluster and the data from the service registry is saved to a hard disk. To do this, the configuration has to be changed in the `Dockerfile` and several instances of Consul have to be started. For more information, see https://www.consul.io/docs/guides/bootstrapping.html.

Configure Spring Boot

Consul can also be used for saving the configuration of a Spring Boot application, see https://cloud.spring.io/spring-cloud-consul/#spring-cloud-consul-config. Use Consul to configure the example application with Consul.

Replace Apache httpd with another server

Replace Apache httpd with nginx, another web server, or HAProxy. To do this, you need to create an appropriate Docker image or search for a matching Docker image in the Docker hub. In addition, the web server must be provided with reverse proxy extensions and configured with Consul Template. Additional documentation about Consul Template can be found on the Github web page. For many systems there are also Consul Template examples.

Scaling & load balancing

Try scaling and load balancing.

- Increase the number of instances of a service, for example with `docker-compose up --scale customer=2`.
- Use the Consul dashboard to check if two customer microservices are running. It is available under port 8500, at http://localhost:8500/ when Docker is running on the local computer.
- Observe the logs of the order microservice with `docker logs -f msconsul_order_1` and see if different instances of the customer microservice are called. This should be the case because Ribbon is used for load balancing. To do this, you must trigger requests to the order application by reloading the homepage.

application by reloading the homepage.

- Add your own health check for one of the microservices. Check whether load balancing actually excludes a service when the health check is no longer successful.

We'll conclude this chapter with a quick summary in the next lesson.