

Workshop #1 - Daniel Ramirez Gomez

miércoles, 7 de febrero de 2024 1:07 a. m.

Git clone was done from the repository

```
root@ubuntu:/home/danigomez/Documents/Workshop-1# git clone https://github.com/icesi-ops/training_microservices.git
```

We will proceed first with the app-config server.

First, access its Dockerfile to change its listening port to 8888, which is the common port for a configuration server in Spring Boot.

Also, verify that in its application.properties, it is also listening on that port.

```
spring.application.name=app-config
server.port=8888

# Config
spring.cloud.config.server.git.uri=https://github.com/icesi-ops/training_microservices.git
spring.cloud.config.server.default-label=master
spring.cloud.config.server.git.search-paths=pay-app-spring-microservices/config
spring.cloud.config.server.git.skip-ssl-validation=true

# Consul
spring.cloud.consul.host=consul
spring.cloud.consul.port=8500
spring.cloud.consul.discovery.health-check-interval=5s
spring.cloud.consul.discovery.prefer-ip-address=true
root@ubuntu:/home/danigomez/Documents/Workshop-1/training_microservices/pay-app-spring-microservices/app-config
```

Now the network creation

docker network create microservicenetwork

```
danigomez@ubuntu:~/Documents/pr1/sd-workshop1$ docker network ls
NETWORK ID      NAME                DRIVER  SCOPE
4437b54fd4e7    bridge              bridge  local
ae2c872c25ff    distribuidos        bridge  local
38e3620d1c21    host                host    local
d50b1af80e23    microservicenetwork bridge  local
a9da5aa925db    none               null    local
```

In the build.gradle, it is noticed that a service called consul is used. Therefore, a container must be created to keep the consul server running.

Starting Consul:

```
docker run -d -p 8500:8500 -p 8600:8600/udp --network microservicenetwork --name consul
consul:1.15 agent -server -bootstrap-expect 1 -ui -data-dir /tmp -client=0.0.0.0
```

Now, build the image of the config server service.

Specify the platform that the image will use and also provide a tag. Use the Dockerfile from the current directory with ..

docker build -t ventana1901/app-config --platform=linux/amd64 .

Now, build his container

```
docker run -dit -p 8888:8888 --network microservicenetwork --name app-config ventana1901/app-config
```

Verify that the service is up.

```
curl --location --request GET 'http://localhost:8888/app-invoice/dev'
```

```
{
  "name": "app-invoice",
  "profiles": [
    "dev"
  ],
  "label": null,
  "version": "76ca227bc376a79cb1321ffcea2dab90a0f00bd9",
  "state": null,
  "propertySources": [
    {
      "name": "https://github.com/icesi-ops/training_microservices.git/pay-app-spring-microservices/config/app-invoice-dev.properties",
      "source": {
        "spring.application.name": "app-invoice",
        "server.port": "8006",
        "spring.kafka.consumer.bootstrap-servers": "servicekafka:9092",
        "spring.kafka.admin.properties.bootstrap.servers": "servicekafka:9092",
        "spring.kafka.consumer.key-deserializer": "org.apache.kafka.common.serialization.IntegerDeserializer",
        "spring.kafka.consumer.value-deserializer": "org.apache.kafka.common.serialization.StringDeserializer",
        "spring.kafka.consumer.group-id": "invoice-events-listener-group",
        "logging.level.org.hibernate.SQL": "debug",
        "spring.jpa.properties.hibernate.enable_lazy_load_no_trans": "true",
        "spring.jpa.hibernate.ddl-auto": "create",
        "spring.datasource.url": "jdbc:postgresql://postgres:5432/db_invoice",
        "spring.datasource.username": "postgres",
        "spring.datasource.password": "postgres",
        "spring.datasource.driver-class-name": "org.postgresql.Driver",
        "spring.jpa.database-platform": "org.hibernate.dialect.PostgreSQL95Dialect",
        "spring.cloud.consul.host": "consul",
        "spring.cloud.consul.port": "8500",
        "spring.cloud.consul.discovery.health-check-interval": "5s",
        "spring.cloud.consul.discovery.prefer-ip-address": "true"
      }
    }
  ]
}
```

Now, move to the app-invoice folder to start the microservice.

Check that in the app-invoice's curl, the port on which the service listens is 8006. Therefore, it needs to be modified in the Dockerfile in the EXPOSE section.

Then two dependencies must be added to the app-invoice's build.gradle.

```
implementation 'org.springframework.cloud:spring-cloud-starter-consul-discovery'
implementation 'org.springframework.boot:spring-boot-starter-actuator'
```

```

danigomez@ubuntu:~/Documents/Workshop-1/training_microservices/pay-app-spring-microservices/app-invoice$ cat build.gradle
plugins {
    id 'org.springframework.boot' version '2.3.10.RELEASE'
    id 'io.spring.dependency-management' version '1.0.11.RELEASE'
    id 'java'
}

group = 'com.aforo'
version = '1.0.0'
sourceCompatibility = '11'

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

ext {
    set('springCloudVersion', "Hoxton.SR11")
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.kafka:spring-kafka'
    implementation 'org.springframework.cloud:spring-cloud-starter-config'
    implementation 'org.springframework.cloud:spring-cloud-starter-consul-discovery'
    implementation 'org.springframework.boot:spring-boot-starter-actuator'
    compileOnly 'org.projectlombok:lombok'
    runtimeOnly 'org.postgresql:postgresql'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testImplementation 'org.springframework.kafka:spring-kafka-test'
}

dependencyManagement {
    imports {
        mavenBom "org.springframework.cloud:spring-cloud-dependencies:${springCloudVersion}"
    }
}

```

Verify in the config folder that in the app-invoice properties, consul is configured as variables

```

# Consul
spring.cloud.consul.host=consul
spring.cloud.consul.port=8500
spring.cloud.consul.discovery.health-check-interval=5s
spring.cloud.consul.discovery.prefer-ip-address=true

```

Build the image located in the folder containing the Dockerfile app-invoice

docker build -t ventana1901/app-invoice --platform=linux/amd64 .

Before starting its container, check that the service depends on its database and Kafka, so these microservices that provide operability to app-invoice must be started first.

To start Postgres, go to the folder containing its Dockerfile and execute the build.

docker build -t ventana1901/postgres --platform=linux/amd64 .

Now run the container

docker run -p 5434:5432 --name postgres --network distribuidos -e POSTGRES_PASSWORD=postgres -e POSTGRES_DB=db_invoice -d postgres:12-alpine

Now execute the Kafka service

docker run -p 2181:2181 -d -p 9092:9092 --name servicekafka --network microservice -e ADVERTISED_HOST=servicekafka -e NUM_PARTITIONS=3 johnnypark/kafka-zookeeper:2.6.0

Now app-invoice can be executed

docker run -dit -p 8006:8006 --name app-invoice --network microservicenetwork ventana1901/app-invoice

Once all the containers are ready, they should be executed in the correct order. First, stop them

Docker stop \$(docker ps -aq)

Now start them in the following order with the docker start command (Container ID):

- 1) Consul
- 2) Kafka
- 3) Postgres
- 4) app-config
- 5) app-invoice

And i get the following

```

danigomez@ubuntu:~/Documents/pr1/sd-workshop1$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
12df1157cdf0   ventana1901/app-invoice:latest      "java -jar /app-invo..." 15 minutes ago Up 9 minutes  0.0.0.0:8006->8006/tcp, :::8006->8006/tcp
3aa56d0f04b9   postgres:12-alpine                 "docker-entrypoint.s..." 2 hours ago    Up 14 minutes  0.0.0.0:5434->5432/tcp, :::5434->5432/tcp
41bfca6e3e0d   ventana1901/app-config:v2          "java -jar /app-conf..." 3 hours ago    Up 14 minutes  0.0.0.0:8888->8888/tcp, :::8888->8888/tcp
126c76f3d972   consul:1.15                         "docker-entrypoint.s..." 5 hours ago    Up 15 minutes  8300-8302/tcp, 8600/tcp, 8301-8302/udp, 0.0.0.0:8500->8500/tcp, :::8500->8500/tcp, 0.0.0.0:8600->8600/tcp, :::8600->8600/tcp
0f123f3dc893   johnnypark/kafka-zookeeper:2.6.0    "supervisord -n"         5 hours ago    Up 14 minutes  0.0.0.0:2181->2181/tcp, :::2181->2181/tcp, 0.0.0.0:9092->9092/tcp, :::9092->9092/tcp

```

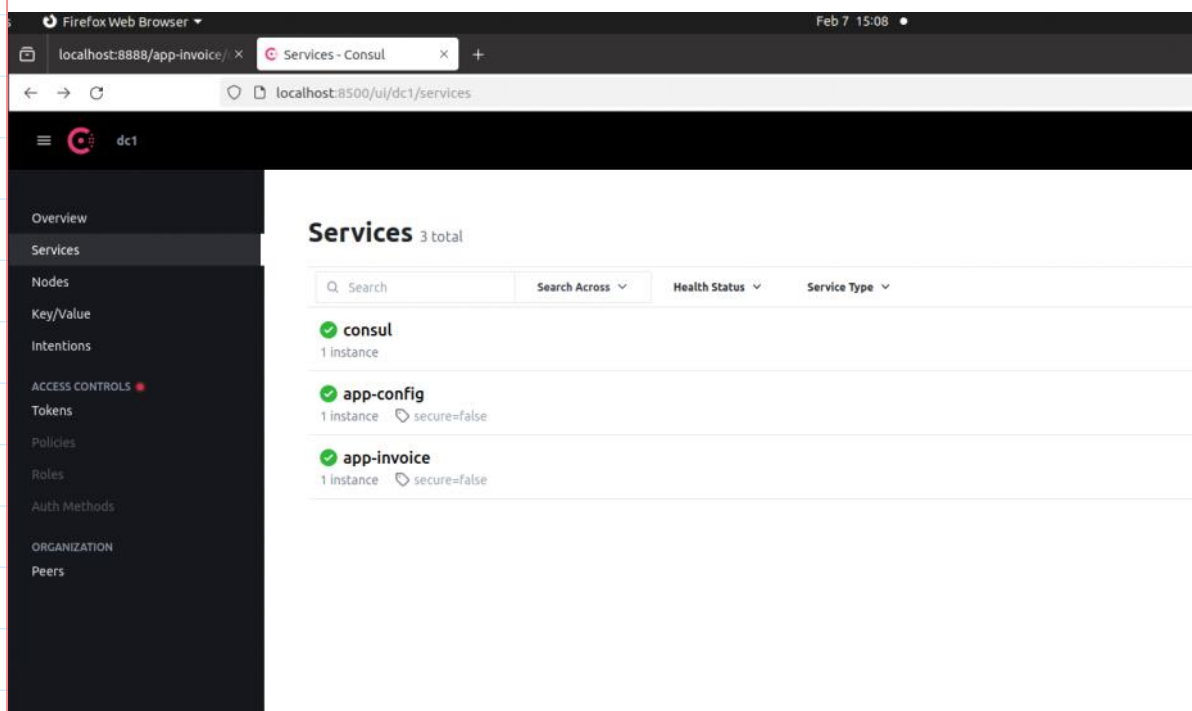
Here, you can also see all the created images

```

danigomez@ubuntu:~/Documents/pr1/sd-workshop1$ docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
ventura1901/app-invoice  latest     455b23438204  21 minutes ago  407MB
ventura1901/app-config   v2         43fb89325122  5 hours ago    379MB
ventura1901/postgres     latest     9c7948f5598b  6 hours ago    234MB
ventura1901/app-config   latest     b258d52cb9f8  7 hours ago    366MB
postgres             12-alpine  84afb3167771  6 weeks ago    234MB
consul                 1.15       3295d4f4567b  2 months ago   155MB
johnnypark/kafka-zookeeper latest     bb7d8cc6a364  2 months ago   416MB
ubuntu/kafka           latest     eee90c2b0c90  7 months ago   383MB
johnnypark/kafka-zookeeper 2.6.0      753c08c7e13f  3 years ago    366MB

```

Now, you can verify that various microservices are in Consul



Finally, push the created images.

ventura1901 / postgres Contains: Image Last pushed: less than a minute ago	Inactive	0	3	Public
ventura1901 / app-config Contains: Image Last pushed: 1 minute ago	Inactive	0	6	Public
ventura1901 / app-invoice Contains: Image Last pushed: 1 minute ago	Inactive	0	4	Public

To now add data with PostgreSQL

First, you must execute this command in the container folder for the data, data.sql

```
PGPASSWORD=postgres psql -h localhost -d db_invoice -U postgres -f data.sql
```

Then, the data are added

```
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
```

Now, if we go to the URL <http://localhost:8006/all>, we can see the following.

The screenshot shows a web browser window with the address bar displaying 'localhost:8006/all'. The page content is a JSON array of five invoice objects. The browser's developer tools are open, showing the JSON data in the 'JSON' tab. The data is as follows:

Index	idInvoice	amount	state
0	1	1000	0
1	2	5000	1
2	3	300	0
3	4	600	0
4	5	400	0