Cannot find a way to configure Eureka client with Docker swarm mode #1820

New issue

Closed **cecchisandrone** opened this issue on 30 Mar · 13 comments

cecchisandrone commented on 30 Mar

When using Eureka client with Docker swarm mode, I didn't find a way to correctly configure the client to notify the correct hostname / ip. With Docker swarm mode you have at least two networks (if the container exposes a port to the public network), the ingress and the overlay network used for internal cluster communication (the correct one to use for Eureka). The problem is that, in each container, you have two interfaces eth0 and eth1; these are assigned randomically to the ingress or overlay ip thus is not possible to benefit from `spring.cloud.inetutils.ignoredInterfaces` property. So what happens is that randomically Eureka clients advertise the correct host configuration. Did you have any similar case to this or is there a specific Eureka configuration that can help me with that?

spencergibb added the `question` label on 30 Mar

spencergibb commented on 30 Mar Member

You have to have some way to know what address to advertise. I've not used swarm before, so I can't help there.

cecchisandrone commented on 30 Mar

I have found two solutions to this problem:

1. Specifying a custom network value in Docker swarm and using `spring.cloud.inetutils.preferredNetworks` property
2. Using `eureka.instance.hostName: service-name` and `eureka.instance.preferIpAddress: false`, where service-name is the same service-name specified in the Docker compose file

What do you think about them?

👍 3

spencergibb commented on 30 Mar Member

I think either of those things sound reasonable if you've found them to work.

diegochavezcarro commented on 6 Jul

Excelent! using eureka.instance.hostName: service-name and eureka.instance.preferIpAddress: false worked for me!

diegochavezcarro added a commit to diegochavezcarro/spmia4 that referenced this issue on 6 Jul

`spring-cloud/spring-cloud-netflix#1820` `9bb906f`

diegochavezcarro added a commit to diegochavezcarro/spmia4 that referenced this issue on 6 Jul

`spring-cloud/spring-cloud-netflix#1820` `3b9036e`

diegochavezcarro commented on 6 Jul

Assignees

No one assigned

Labels

Projects

None yet

Milestone

No milestone

Notifications

6 participants

The problem with option 2 (eureka.instance.hostName: service-name and eureka.instance.preferIpAddress: false) is that I think load balancing is being made in docker swarm.
Look the following code:

```
discoveryClient.getServices().forEach(serviceName -> {
discoveryClient.getInstances(serviceName).forEach(instance->{
services.add( String.format("%s:%s",serviceName,instance.getUri()));
});
});
```

These are the results:

```
[
"organizationservice:http://organizationservice:8085",
"organizationservice:http://organizationservice:8085",
"organizationservice:http://organizationservice:8085",
"organizationservice:http://organizationservice:8085",
"licensingservice:http://licensingservice:8080"
]
```

So load balancing is not on client side.
But using option 1 instead,such as:

```
spring:
cloud
inetutils:
preferredNetworks:
- 192.168
- 10.0
```

I could see is working appropriately,these are the results:

```
[
"licensingservice:http://10.0.0.7:8080",
"organizationservice:http://10.0.0.11:8085",
"organizationservice:http://10.0.0.12:8085",
"organizationservice:http://10.0.0.9:8085",
"organizationservice:http://10.0.0.10:8085"
]
```

diegochavezcarro added a commit to diegochavezcarro/spmia4 that referenced this issue on 7 Jul

`spring-cloud/spring-cloud-netflix#1820` 231c989

diegochavezcarro added a commit to diegochavezcarro/spmia4 that referenced this issue on 7 Jul

`spring-cloud/spring-cloud-netflix#1820` 4daa355

ryanjbaxter commented on 11 Jul — Contributor

Does this issue need to be open anymore? Sounds like there is a solution using option 2 above

👍 1

diegochavezcarro commented on 11 Jul

In think option 1 is more adequate. With option 2 load balancing is being made in docker swarm and not in the client, so we would not have client load balancing which is one of the key advantages of using Eureka plus Ribbon.

spencergibb commented on 11 Jul — Member

Either way, there is a configuration that works. I don't see the need for us to change anything.

ryanjbaxter closed this on 11 Jul

abhijitvk referenced this issue in department-of-veterans-affairs/ascent-sample on 5 Oct

Load balancing not working in docker-demo profile #26 Open

binakot commented 14 days ago • edited

Hello everyone 🖐 . Same problem with my spring-cloud (`Dalston.SR3`) system in docker swarm.

Option 2 (`preferIpAddress: false` & `service-name`) really breaks the client-side load balancing (eureka + ribbon). That's why I am trying to use option 1 (`preferIpAddress: true` & `preferredNetworks`).

When we run a service in docker swarm, service without public ports registers in only overlay network (by default it's `10.0.0.0/24` , but we can create own `docker network create` with another mask, gateway, etc). When we register a service with public ports, docker swarm add this service to overlay (`10.0.0.0/24` or your own) and ingress (`10.255.0.0/16`) networks. Overlay lets services to communicate and ingress lets to swarm's built-in load balancing between service replicas.

This is my configuration `v1` :

```
spring:
  cloud:
    inetutils:
      preferred-networks:
        - 10.0

eureka:
  instance:
    preferIpAddress: true
```

With `v1` configuration Eureka show this:

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
AUTH	n/a (1)	(1)	UP (1) - 10.255.0.11:auth:9999
CONFIG	n/a (1)	(1)	UP (1) - 10.255.0.19:config:8888
GATEWAY	n/a (1)	(1)	UP (1) - 10.255.0.17:gateway:8765
MONITOR	n/a (1)	(1)	UP (1) - 10.255.0.3:monitor:9411
REGISTRY	n/a (1)	(1)	UP (1) - 10.255.0.13:registry:8761
TEMPLATE-SERVICE-1	n/a (1)	(1)	UP (1) - 10.0.0.8:template-service-1:8080
TEMPLATE-SERVICE-2	n/a (1)	(1)	UP (1) - 10.0.0.6:template-service-2:8080

Look at `instance-id` in last column. By default `instance-id` create by pattern:

```
${spring.cloud.client.hostname}:${spring.application.name}:${spring.application.instance_i
```

This means that `spring.cloud.client.hostname` is getting from ingress network while overlay is not ready.

```
"springCloudClientHostInfo": {
    "spring.cloud.client.hostname": "10.255.0.15",
    "spring.cloud.client.ipAddress": "10.255.0.15"
}
```

Eureka uses `instance-id` to distinct between different instances of the same application. But if we check `http://localhost:8761/eureka/apps` , everything is correct except `instance-id` :

```
<application>
  <name>AUTH</name>
  <instance>
```

```
    <instanceId>10.255.0.11:auth:9999</instanceId>
    <hostName>10.0.0.15</hostName>
    <app>AUTH</app>
    <ipAddr>10.0.0.15</ipAddr>
    <status>UP</status>
    <...>
  </instance>
</application>

<application>
  <name>TEMPLATE-SERVICE-1</name>
  <instance>
    <instanceId>10.0.0.8:template-service-1:8080</instanceId>
    <hostName>10.0.0.9</hostName>
    <app>TEMPLATE-SERVICE-1</app>
    <ipAddr>10.0.0.9</ipAddr>
    <status>UP</status>
    <...>
  </instance>
</application>
```

I just change `instance-id` pattern and create my configuration `v2`:

```
spring:
  cloud:
    inetutils:
      preferred-networks:
        - 10.0

eureka:
  instance:
    preferIpAddress: true
    instance-id: ${spring.application.name}:${spring.application.instance_id:${random.valu
```

And Eureka shows:

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
AUTH	n/a (1)	(1)	UP (1) - auth:f48997c36edbc1f1acdb5fb0ff8dc533
CONFIG	n/a (1)	(1)	UP (1) - config:9990eddf202a8512cc5a3d99feb81646
GATEWAY	n/a (1)	(1)	UP (1) - gateway:d72ddcd0683ea095205bf16e69f8b7ff
MONITOR	n/a (1)	(1)	UP (1) - monitor:1dc5189c5d90bc88ce2ed71d21db6ced
REGISTRY	n/a (1)	(1)	UP (1) - registry:cfd30418f61bcfb01bdc187ee80faea5
TEMPLATE-SERVICE-1	n/a (1)	(1)	UP (1) - template-service-1:7b6e38be3c96a44bafced1633b9948f5
TEMPLATE-SERVICE-2	n/a (1)	(1)	UP (1) - template-service-2:f262fd12d6b90bafba6de9b86eb77a33

```
<application>
  <name>AUTH</name>
  <instance>
    <instanceId>auth:f48997c36edbc1f1acdb5fb0ff8dc533</instanceId>
    <hostName>10.0.0.9</hostName>
    <app>AUTH</app>
    <ipAddr>10.0.0.9</ipAddr>
    <status>UP</status>
    <...>
  </instance>
</application>

<application>
  <name>TEMPLATE-SERVICE-1</name>
  <instance>
    <instanceId>template-service-1:7b6e38be3c96a44bafced1633b9948f5</instanceId>
    <hostName>10.0.0.15</hostName>
    <app>TEMPLATE-SERVICE-1</app>
    <ipAddr>10.0.0.15</ipAddr>
    <status>UP</status>
    <...>
  </instance>
</application>
```

Okay. For now the most important feature is scaling. Lets try to run stack in docker swarm with 2 `template-service-1` and 3 `template-service-2`:

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
AUTH	n/a (1)	(1)	UP (1) - auth:a9f4d2ecdf9d38c99ff53f4e8cb752c4
CONFIG	n/a (1)	(1)	UP (1) - config:a0d5fd04f13e8c0d7a43751ffb870eee
GATEWAY	n/a (1)	(1)	UP (1) - gateway:caa93f3685c811554c9c1b17f6a2d7c9
MONITOR	n/a (1)	(1)	UP (1) - monitor:449b178dddb0f75ed8230ab7e6942368
REGISTRY	n/a (1)	(1)	UP (1) - registry:4b911b846accd310b80a557a40383167
TEMPLATE-SERVICE-1	n/a (2)	(2)	UP (2) - template-service-1:7ebf636b8485e81fc878cb9329f6e3c7 , template-service-1:7863ab4afc3059d54c654e7c417b2a17
TEMPLATE-SERVICE-2	n/a (3)	(3)	UP (3) - template-service-2:fb0638bdb669476f6ae03c6c59bb7847 , template-service-2:38b2ed9f2d337463246d0dbeb6bbae0a , template-service-2:995e06b767c000c68f1152b56bd0b1fd

```
<application>
  <name>TEMPLATE-SERVICE-1</name>
  <instance>
    <instanceId>template-service-1:3faecf171f061b2068471b3d866f4805</instanceId>
    <hostName>10.0.0.8</hostName>
    <app>TEMPLATE-SERVICE-1</app>
    <ipAddr>10.0.0.8</ipAddr>
    <status>UP</status>
    <...>
  </instance>
  <instance>
    <instanceId>template-service-1:7e29744497a4045b037c2b03f3801cfa</instanceId>
    <hostName>10.0.0.8</hostName>
    <app>TEMPLATE-SERVICE-1</app>
    <ipAddr>10.0.0.8</ipAddr>
    <status>UP</status>
    <...>
  </instance>
</application>

<application>
  <name>TEMPLATE-SERVICE-2</name>
  <instance>
    <instanceId>template-service-2:f10c8dab85c45e6492af905f8a6a845c</instanceId>
    <hostName>10.0.0.21</hostName>
    <app>TEMPLATE-SERVICE-2</app>
    <ipAddr>10.0.0.21</ipAddr>
    <status>UP</status>
    <...>
  </instance>
  <instance>
    <instanceId>template-service-2:ce5de6eebfb22c092d4b0a6ca85e2829</instanceId>
    <hostName>10.0.0.21</hostName>
    <app>TEMPLATE-SERVICE-2</app>
    <ipAddr>10.0.0.21</ipAddr>
    <status>UP</status>
    <...>
  </instance>
  <instance>
    <instanceId>template-service-2:87a86dafd79723fc023600073cc93614</instanceId>
    <hostName>10.0.0.21</hostName>
    <app>TEMPLATE-SERVICE-2</app>
    <ipAddr>10.0.0.21</ipAddr>
    <status>UP</status>
    <...>
  </instance>
</application>
```

Look at IP addresses of all replicas of both services - they are equal 🤕

And the last test to check run-time scaling `docker service update --replicas 3 template-service-1` & `docker service update --replicas 1 template-service-2`:

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
AUTH	n/a (1)	(1)	UP (1) - auth:bc1723e24973345f4f19b39126b26bff
CONFIG	n/a (1)	(1)	UP (1) - config:dd83d0bc018d77c8e3e1ed0a35dda1f1
GATEWAY	n/a (1)	(1)	UP (1) - gateway:15cfdf547caf309499a4d65373a8df5b
MONITOR	n/a (1)	(1)	UP (1) - monitor:c32cf795c31feebd39cb66abc50a86fc
REGISTRY	n/a (1)	(1)	UP (1) - registry:2c4f187d9f0b4c036321dcf0c62d1ddc
TEMPLATE-SERVICE-1	n/a (3)	(3)	UP (3) - template-service-1:668680a6aaf45ae1be327e5401a81623 , template-service-1:9cc743b19757d3dd7fec293a1d7dfac0 , template-service-1:e3261ddbcf75b4f4cf8c78759d6aca2a
TEMPLATE-SERVICE-2	n/a (1)	(1)	UP (1) - template-service-2:95415aa175692c410d066e92092940a5

```
<application>
  <name>TEMPLATE-SERVICE-1</name>
  <instance>
    <instanceId>template-service-1:6a3d376a43051ee2a1c931b3ed61fb60</instanceId>
    <hostName>10.0.0.8</hostName>
    <app>TEMPLATE-SERVICE-1</app>
    <ipAddr>10.0.0.8</ipAddr>
    <status>UP</status>
    <...>
  </instance>
  <instance>
    <instanceId>template-service-1:3faecf171f061b2068471b3d866f4805</instanceId>
    <hostName>10.0.0.8</hostName>
    <app>TEMPLATE-SERVICE-1</app>
    <ipAddr>10.0.0.8</ipAddr>
    <status>UP</status>
    <...>
  </instance>
  <instance>
    <instanceId>template-service-1:7e29744497a4045b037c2b03f3801cfa</instanceId>
    <hostName>10.0.0.8</hostName>
    <app>TEMPLATE-SERVICE-1</app>
    <ipAddr>10.0.0.8</ipAddr>
    <status>UP</status>
    <...>
  </instance>
</application>

<application>
  <name>TEMPLATE-SERVICE-2</name>
  <instance>
    <instanceId>template-service-2:87a86dafd79723fc023600073cc93614</instanceId>
    <hostName>10.0.0.21</hostName>
    <app>TEMPLATE-SERVICE-2</app>
    <ipAddr>10.0.0.21</ipAddr>
    <status>UP</status>
    <...>
  </instance>
</application>
```

Eureka understands that service's replicas count are changing, but...

One more time. Look at `hostName` and `ipAddr` of every instances of both services. They are equal 😣. Eureka has 3 instances of my `template-service-1` with same ip `10.0.0.8` which was used on the 1th registration. Let's check out our network `docker network inspect stack_default`:

```
[
    {
        "Name": "stack_default",
        "Id": "fusf35hkd98se12cj83w60idn",
        "Created": "2017-11-22T08:14:29.363971Z",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "10.0.0.0/24",
                    "Gateway": "10.0.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "0f42cbc6278af4a2d7b5dcf886f3946183f8f28569f13a6f7f081b469964d34f": {
                "Name": "stack_template-service-2.2.1mk1nm3k7lkxulq3hu82ytcf6",
                "EndpointID": "a0d09bf5458eaf281867c039897056cc5b290963d6d8f4e7944780b4bc0
                "MacAddress": "02:42:0a:00:00:17",
                "IPv4Address": "10.0.0.23/24",
                "IPv6Address": ""
            },
            "519586a0855dc64d041e76e3e8d647177f689b72552ee4db86fd6c5caa60e058": {
                "Name": "stack_template-service-1.3.r6p74h0lv0825pyt4j9r9rshn",
                "EndpointID": "f2da49faa39dead6a98c5b18388aed58980cc281a3dc74ca046feda765f
                "MacAddress": "02:42:0a:00:00:1b",
                "IPv4Address": "10.0.0.27/24",
                "IPv6Address": ""
            },
            "596154a80c95f21152130099b3738259e2d87f6c66580c78e2422a339cb33135": {
                "Name": "stack_config.1.nc2kmq8no8nha00u37dgxf84k",
                "EndpointID": "34f00a19458a098d0536ecf3ec73c972294befbcda89d441e9930420e74
                "MacAddress": "02:42:0a:00:00:0e",
                "IPv4Address": "10.0.0.14/24",
                "IPv6Address": ""
            },
            "6e05edd58d8b300bad4407efa3462a30e0bc678c15cb93041a516ec88419c4ac": {
                "Name": "stack_rabbitmq.1.n3nio3q50zfmhxawzac0wlpkm",
                "EndpointID": "46b7f2c24927fed77bb66625a338caf8af9581dd1161802eab80e204fb2
                "MacAddress": "02:42:0a:00:00:10",
                "IPv4Address": "10.0.0.16/24",
                "IPv6Address": ""
            },
            "ab62abccad3c50745d1ddfd5b44e5d320c7915e0e880caf6df94dc699138e814": {
                "Name": "stack_template-service-1.1.pe9kmnzhord5e7cqcf6asstvw",
                "EndpointID": "70bb8a08b6d1a2da1fb228e52ec1acbddb7b371b41a73c55af74115e366
                "MacAddress": "02:42:0a:00:00:09",
                "IPv4Address": "10.0.0.9/24",
                "IPv6Address": ""
            },
            "da06a28ab31cefb030cd68729589f45ae82fdfbd5545b431f2cf97a8f0af7e30": {
                "Name": "stack_registry.1.if6gdn48et01zpgyssa7nfq8e",
                "EndpointID": "9dfd032f61bfe8d096e8f7413e690cd02e06d47fc88cf2a9e4477be18a9
                "MacAddress": "02:42:0a:00:00:12",
                "IPv4Address": "10.0.0.18/24",
                "IPv6Address": ""
            },
            "e788ac5083cdb8e6f33264e5cb2de58e5c9f73810286feb593a2f51ae993ea11": {
                "Name": "stack_template-service-1.2.qnlor4tcx6kt9fypadxtt7lgp",
                "EndpointID": "7591b226918a51d1f4dbe6c8b66edb14f104ca658d6c20f298e8f0bfb79
                "MacAddress": "02:42:0a:00:00:0a",
                "IPv4Address": "10.0.0.10/24",
                "IPv6Address": ""
            },
            "ea22b9903c95c62adbf6dd916890e858fce79b7b1cc5241c78d78051aad8cc22": {
                "Name": "stack_gateway.1.y42ihbkq93f5t5spwj321amkz",
                "EndpointID": "62fb9e93c23b3c45e3fc2ee4b83410bca9e70b3ca5c27f4d39a7cfac64f
                "MacAddress": "02:42:0a:00:00:1a",
                "IPv4Address": "10.0.0.26/24",
                "IPv6Address": ""
            },
            "f694dc0cf21f774490413293a328cbf400f26f6cd6739804d71aa08c65b7a0ca": {
```

```
                "Name": "stack_auth.1.xp8tb6h1zam0luh53ucnxsus0",
                "EndpointID": "06021ee2004052bd40a5969c551f217e3db99603a0b09479e34b12d4f39a
                "MacAddress": "02:42:0a:00:00:07",
                "IPv4Address": "10.0.0.7/24",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.driver.overlay.vxlanid_list": "4097"
        },
        "Labels": {
            "com.docker.stack.namespace": "stack"
        },
        "Peers": [
            {
                "Name": "moby-33d8cf6a608e",
                "IP": "192.168.65.2"
            }
        ]
    }
]
```

There are 3 replicas of `template-service-1`:

- "Name": "stack_template-service-1.1.pe9kmnzhord5e7cqcf6asstvw", "IPv4Address": "10.0.0.9/24",
- "Name": "stack_template-service-1.2.qnlor4tcx6kt9fypadxtt7lgp", "IPv4Address": "10.0.0.10/24"
- "Name": "stack_template-service-1.3.r6p74h0lv0825pyt4j9r9rshn", "IPv4Address": "10.0.0.27/24"

Why Eureka registered service's replicas with same ip `10.0.0.12`? 🤕 This IP address is not using anymore in overlay network, it was using for the 1th replica before I changed scale value. This means that Eureka remember 1th IP address of the 1th replica of any service and continue to use it for every other repicas.

What do you think, guys? How can I fix that?

UPDATE1:

I just run the stack in docker swarm. I have gateway service with single replica. When it's running first, it crashed because config server is not available yet. Then container restart again. It is okay in docker environment (restart=always and etc, u know 😄).

The most interesting thing is result IP address of gateway. First container got IP `10.0.0.10`, but was terminated. Second container got IP `10.0.0.11` and was successfully run.

Now lets look at Eureka and network state:

```
<application>
  <name>GATEWAY</name>
  <instance>
    <instanceId>gateway:f1a615945d3b68bccb635f0f0166193a</instanceId>
    <hostName>10.0.0.10</hostName>
    <app>GATEWAY</app>
    <ipAddr>10.0.0.10</ipAddr>
    <status>UP</status>
    <...>
  </instance>
</application>
```

But this is from `docker network inspect stack_default`

```
"03f9423b940d2fd3392212d7f99914ef710db447da7bca15a8e181a91b4665b1": {
        "Name": "stack_gateway.1.pq2s9c7p7zd6hukfr7jax80gg",
        "EndpointID": "f51b467b3da326735fba11bb603caa2a788ff0a7077057469dc0fd7c9fd1e905",
        "MacAddress": "02:42:0a:00:00:0b",
        "IPv4Address": "10.0.0.11/24",
        "IPv6Address": ""
}
```

And from container itself:

```
$ ifconfig
```

```
eth0      Link encap:Ethernet  HWaddr 02:42:0A:FF:00:0A
          inet addr:10.255.0.10  Bcast:0.0.0.0  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1450  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:116 (116.0 B)  TX bytes:0 (0.0 B)

eth1      Link encap:Ethernet  HWaddr 02:42:AC:15:00:0B
          inet addr:172.21.0.11  Bcast:0.0.0.0  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:90 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4066 (3.9 KiB)  TX bytes:582 (582.0 B)

eth2      Link encap:Ethernet  HWaddr 02:42:0A:00:00:0B
          inet addr:10.0.0.11  Bcast:0.0.0.0  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1450  Metric:1
          RX packets:560 errors:0 dropped:0 overruns:0 frame:0
          TX packets:749 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:78216 (76.3 KiB)  TX bytes:89291 (87.1 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:134 errors:0 dropped:0 overruns:0 frame:0
          TX packets:134 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:9088 (8.8 KiB)  TX bytes:9088 (8.8 KiB)
```

Eureka just registered gateway with IP of the 1th failed container. Is this DNS cache or what?

If I ping `gateway` from itself, IP is correct:

```
$ hostname
gateway
$ ping gateway
PING gateway (10.0.0.11): 56 data bytes
64 bytes from 10.0.0.11: seq=0 ttl=64 time=0.053 ms
```

But If I ping from another container, IP is wrong (cached with 1th failed container):

```
$ hostname
auth
$ ping gateway
PING gateway (10.0.0.10): 56 data bytes
64 bytes from 10.0.0.10: seq=0 ttl=64 time=0.042 ms
```

How can I solve this problem, if I understand it correctly? Can I force the Eureka clients to send their IP addresses itself without Eureka server's DNS resolving from already invalid cache? Cheers ☕

UPDATE2:

I found the way to specify hostname and ip for eureka clients:

```yaml
eureka:
  instance:
    instance-id: ${spring.application.name}:${spring.application.instance_id:${random.valu
    prefer-ip-address: true
    hostname: localhost
    ip-address: 127.0.0.1
```

From `/eureka/apps`:

```
<application>
  <name>CONFIG</name>
  <instance>
    <instanceId>config:3a734041536c13e20dff7b3e0c21f9ae</instanceId>
    <hostName>127.0.0.1</hostName>
    <app>CONFIG</app>
    <ipAddr>127.0.0.1</ipAddr>
```

```
    <status>UP</status>
    <...>
  </instance>
</application>
```

Now I need to find the way to set these parameters with actual IP address of the container in my overlay network.

The code below doesn't help, because `InetAddress.getLocalHost()` get the externally advertised FQDN for the host (it's cached in DNS too).

```
    @Bean
    @Autowired
    public EurekaInstanceConfigBean eurekaInstanceConfig(final InetUtils inetUtils) throws
        final EurekaInstanceConfigBean config = new EurekaInstanceConfigBean(inetUtils);
        config.setHostname(InetAddress.getLocalHost().getHostName());
        config.setIpAddress(InetAddress.getLocalHost().getHostAddress());
        return config;
    }
```

Lets check all network interfaces:

```
Enumeration<NetworkInterface> networkInterfaces = NetworkInterface.getNetworkInterfaces();
        for (NetworkInterface netInt : Collections.list(networkInterfaces))
            for (InetAddress inetAddress : Collections.list(netInt.getInetAddresses()))
                System.out.printf("InetAddress: %s %s\n", inetAddress.getHostName(), inetA
```

```
InetAddress: 3f5ba074d5f7 10.0.0.3  # ACTUAL
InetAddress: 172.21.0.3 172.21.0.3
InetAddress: 10.255.0.4 10.255.0.4
InetAddress: 10.0.0.2 10.0.0.2  # EUREKA CLIENT USE
InetAddress: 10.255.0.3 10.255.0.3
InetAddress: localhost 127.0.0.1
```

And lets check actual `hostname` of service:

```
$ hostname
3f5ba074d5f7
$ hostname -i
10.0.0.3
```

But Eureka client is using 4th row with `10.0.0.2 10.0.0.2`. What is that? IP address `10.0.0.2` of what? There is no anything with this IP address `docker network inspect stack_default`:

```
[
    {
        "Name": "stack_default",
        "Id": "qm3mqslq6jnozp3h9552ymuh7",
        "Created": "2017-11-22T13:19:16.1862187Z",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "10.0.0.0/24",
                    "Gateway": "10.0.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "3c0f0b69977dbca3bc527c66cc800bec9d047db52eaf62d4ef92c925c59226fb": {
                "Name": "stack_rabbitmq.1.yuohjhatwvleff0y02vr4b3vs",
                "EndpointID": "c99aa7e7ba2bd61ea8d54d937173b89f62f72c20f32999a4753f23094ac
```

```
            "MacAddress": "02:42:0a:00:00:07",
            "IPv4Address": "10.0.0.7/24",
            "IPv6Address": ""
        },
        "3f5ba074d5f7cc1da1186ce64bd5f89d2c10b4ac786888dba8bf9e9cebc4a686": {
            "Name": "stack_config.1.oumx45sc8zwzf2emalozhnc1n",
            "EndpointID": "5f179bbafc0b2f46810deae171672fbaeaf645da94d9b01c04676f92f63
            "MacAddress": "02:42:0a:00:00:03",
            "IPv4Address": "10.0.0.3/24",
            "IPv6Address": ""
        },
        "437a3db4e2195fec210a42d8ef87027e63e68f018c93c77a817d9d5087ffa618": {
            "Name": "stack_registry.1.j0kbsf6zn6aew8x9dbzervn8d",
            "EndpointID": "00fee061dc430885804cf73bf62ee175985383b918dbbdd33720dc2d25a
            "MacAddress": "02:42:0a:00:00:05",
            "IPv4Address": "10.0.0.5/24",
            "IPv6Address": ""
        }
    },
    "Options": {
        "com.docker.network.driver.overlay.vxlanid_list": "4097"
    },
    "Labels": {
        "com.docker.stack.namespace": "stack"
    },
    "Peers": [
        {
            "Name": "moby-33d8cf6a608e",
            "IP": "192.168.65.2"
        }
    ]
  }
]
```

UPDATE3:

I did the dirty trick:

```
@Bean
@Autowired
public EurekaInstanceConfigBean eurekaInstanceConfig(final InetUtils inetUtils) throws IOE
        final String hostName = System.getenv("HOSTNAME");
        String hostAddress = null;
        final Enumeration<NetworkInterface> networkInterfaces = NetworkInterface.getNetwor
        for (NetworkInterface netInt : Collections.list(networkInterfaces)) {
                for (InetAddress inetAddress : Collections.list(netInt.getInetAddresses())
                        if (hostName.equals(inetAddress.getHostName())) {
                                hostAddress = inetAddress.getHostAddress();
                        }
                }
        }

        final EurekaInstanceConfigBean config = new EurekaInstanceConfigBean(inetUtils);
        config.setHostname(hostName);
        config.setIpAddress(hostAddress);
        return config;
}
```

`System.getenv("HOSTNAME")` returns exactly what I need: `3f5ba074d5f7`. And I just find the real IP address of this container. Lets look at Eureka:

```
<application>
  <name>CONFIG</name>
  <instance>
    <instanceId>config:ba555440d63dcb11add385710c429f3a</instanceId>
    <hostName>10.0.0.3</hostName>
    <app>CONFIG</app>
    <ipAddr>10.0.0.3</ipAddr>
    <status>UP</status>
    <...>
  </instance>
</application>
```

Looks like working thing 🔨 Next step is check the swarm scaling and client-side load balancing...

binakot commented 13 days ago • edited

UPDATE4:

Day 2️⃣. I added code below in all my `spring-boot` apps in a cluster.

```
@Configuration
public class EurekaClientConfig {

    @Bean
    @Autowired
    @Profile("docker")
    public EurekaInstanceConfigBean eurekaInstanceConfig(final InetUtils inetUtils) throws
        final String hostName = System.getenv("HOSTNAME");
        String hostAddress = null;

        final Enumeration<NetworkInterface> networkInterfaces = NetworkInterface.getNetwor
        for (NetworkInterface netInt : Collections.list(networkInterfaces)) {
            for (InetAddress inetAddress : Collections.list(netInt.getInetAddresses())) {
                if (hostName.equals(inetAddress.getHostName())) {
                    hostAddress = inetAddress.getHostAddress();
                }

                System.out.printf("%s: %s / %s\n", netInt.getName(),  inetAddress.getHostN
            }
        }

        if (hostAddress == null) {
            throw new UnknownHostException("Cannot find ip address for hostname: " + hostN
        }

        final EurekaInstanceConfigBean config = new EurekaInstanceConfigBean(inetUtils);
        config.setHostname(hostName);
        config.setIpAddress(hostAddress);
        return config;
    }
}
```

Lets check `config` and `registry` services.

Config:

```
eth2: 979c83c891fa / 10.0.0.3
eth1: 172.21.0.3 / 172.21.0.3
eth0: 10.255.0.4 / 10.255.0.4
lo: 10.0.0.2 / 10.0.0.2
lo: 10.255.0.3 / 10.255.0.3
lo: localhost / 127.0.0.1
```

```
<application>
  <name>CONFIG</name>
  <instance>
    <instanceId>config:b9701dfcc9f70001937fc53e9c3dbebb</instanceId>
    <hostName>10.0.0.3</hostName>
    <app>CONFIG</app>
    <ipAddr>10.0.0.3</ipAddr>
    <status>UP</status>
    <...>
  </instance>
</application>
```

Registry:

```
eth2: b2c8b40d16de / 10.0.0.5
eth1: 172.21.0.4 / 172.21.0.4
eth0: 10.255.0.6 / 10.255.0.6
lo: 10.0.0.4 / 10.0.0.4
lo: 10.255.0.5 / 10.255.0.5
lo: localhost / 127.0.0.1
```

```
<application>
  <name>REGISTRY</name>
  <instance>
    <instanceId>registry:07942e8b1a5cb63062d0caa7da9529f9</instanceId>
    <hostName>10.0.0.5</hostName>
    <app>REGISTRY</app>
    <ipAddr>10.0.0.5</ipAddr>
    <status>UP</status>
    <...>
  </instance>
</application>
```

Overlay network

```
[
    {
        "Name": "stack_default",
        "Id": "mnln7pi1y53chqcqfl5njmap3",
        "Created": "2017-11-23T05:37:47.3809552Z",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "10.0.0.0/24",
                    "Gateway": "10.0.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "5210e982c5dac5f60295fe8b838c0125ef57af59ef674109aaf311976f48ea90": {
                "Name": "stack_rabbitmq.1.9ai08boa6ts8v6aanwd3fpjbv",
                "EndpointID": "32d9066c6daaee194ebd771ea959b8783d012744fd564ca791776aa43fa
                "MacAddress": "02:42:0a:00:00:07",
                "IPv4Address": "10.0.0.7/24",
                "IPv6Address": ""
            },
            "979c83c891fa39b4d1ed3c2c95de9209eba15217afde8a5f115ab68c609381fa": {
                "Name": "stack_config.1.84it2lq7j67vmxeibtvp0q7eq",
                "EndpointID": "aee5c133b60f4b21c92f4b6196ef1a29ec4803414348b2363b41e240fbf
                "MacAddress": "02:42:0a:00:00:03",
                "IPv4Address": "10.0.0.3/24",
                "IPv6Address": ""
            },
            "b2c8b40d16de987d87d6ec6cd44acb036246249c1316daf0511b7a07ad41236d": {
                "Name": "stack_registry.1.x7j9gzh4gu1id44j60d0ilx8e",
                "EndpointID": "0126dd6894d961bfe4a6f7788dc1f5a6d9b20bc53021798a5e4c41126f4
                "MacAddress": "02:42:0a:00:00:05",
                "IPv4Address": "10.0.0.5/24",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.driver.overlay.vxlanid_list": "4097"
        },
        "Labels": {
            "com.docker.stack.namespace": "stack"
        },
        "Peers": [
            {
                "Name": "moby-e6a48a47d001",
                "IP": "192.168.65.2"
            }
        ]
    }
]
```

Remember, I asked about `10.0.0.2 / 10.0.0.2`, `10.0.0.4 / 10.0.0.4` and so on which Eureka uses by default. Network interfaces show this one as Loopback. Actually every service in Docker Swarm has 3 `loobpack` and 3 `eth` interfaces. When we set to prefer `10.0` addresses in yaml configuration, we force to use this `loopback`, because the real eth hostname in hex format (`979c83c891fa`, `b2c8b40d16de`, etc).

Okay, we got correct IP addresses on Eureka server from their clients. But we lost other configurations: urls, ports and so on. That's why requests to gateway and other communications through Eureka stop working...

We need something like this:

```
<application>
  <name>CONFIG</name>
  <instance>
    <instanceId>config:effc6bb2ad867575d2c83a451bd1c04b</instanceId>
    <hostName>10.0.0.16</hostName>
    <app>CONFIG</app>
    <ipAddr>10.0.0.16</ipAddr>
    <status>UP</status>
    <overriddenstatus>UNKNOWN</overriddenstatus>
    <port enabled="true">8888</port>
    <securePort enabled="false">443</securePort>
    <countryId>1</countryId>
    <dataCenterInfo class="com.netflix.appinfo.InstanceInfo$DefaultDataCenterInfo">
      <name>MyOwn</name>
    </dataCenterInfo>
    <leaseInfo>
      <renewalIntervalInSecs>30</renewalIntervalInSecs>
      <durationInSecs>90</durationInSecs>
      <registrationTimestamp>1511418416077</registrationTimestamp>
      <lastRenewalTimestamp>1511418416077</lastRenewalTimestamp>
      <evictionTimestamp>0</evictionTimestamp>
      <serviceUpTimestamp>1511418414924</serviceUpTimestamp>
    </leaseInfo>
    <metadata class="java.util.Collections$EmptyMap"/>
    <homePageUrl>http://10.0.0.16:8888/</homePageUrl>
    <statusPageUrl>http://10.0.0.16:8888/info</statusPageUrl>
    <healthCheckUrl>http://10.0.0.16:8888/health</healthCheckUrl>
    <vipAddress>config</vipAddress>
    <secureVipAddress>config</secureVipAddress>
    <isCoordinatingDiscoveryServer>false</isCoordinatingDiscoveryServer>
    <lastUpdatedTimestamp>1511418416077</lastUpdatedTimestamp>
    <lastDirtyTimestamp>1511418265181</lastDirtyTimestamp>
    <actionType>ADDED</actionType>
  </instance>
</application>
```

But with new custom `EurekaClientConfig` configuration we have:

```
<application>
  <name>CONFIG</name>
  <instance>
    <instanceId>config:1d4a762c49306da5e8038bfa74dd72b9</instanceId>
    <hostName>10.0.0.7</hostName>
    <app>CONFIG</app>
    <ipAddr>10.0.0.7</ipAddr>
    <status>UP</status>
    <overriddenstatus>UNKNOWN</overriddenstatus>
    <port enabled="true">80</port>
    <securePort enabled="false">443</securePort>
    <countryId>1</countryId>
    <dataCenterInfo class="com.netflix.appinfo.InstanceInfo$DefaultDataCenterInfo">
      <name>MyOwn</name>
    </dataCenterInfo>
    <leaseInfo>
      <renewalIntervalInSecs>30</renewalIntervalInSecs>
      <durationInSecs>90</durationInSecs>
      <registrationTimestamp>1511418992485</registrationTimestamp>
      <lastRenewalTimestamp>1511419172515</lastRenewalTimestamp>
      <evictionTimestamp>0</evictionTimestamp>
      <serviceUpTimestamp>1511418991404</serviceUpTimestamp>
    </leaseInfo>
    <metadata class="java.util.Collections$EmptyMap"/>
    <homePageUrl>http://10.0.0.7:80/</homePageUrl>
    <statusPageUrl>http://10.0.0.7:80/info</statusPageUrl>
    <healthCheckUrl>http://10.0.0.7:80/health</healthCheckUrl>
```

```
    <vipAddress>config</vipAddress>
    <secureVipAddress>config</secureVipAddress>
    <isCoordinatingDiscoveryServer>false</isCoordinatingDiscoveryServer>
    <lastUpdatedTimestamp>1511418992485</lastUpdatedTimestamp>
    <lastDirtyTimestamp>1511418869926</lastDirtyTimestamp>
    <actionType>ADDED</actionType>
  </instance>
</application>
```

UPDATE5:

I override bean from autoconfiguration `org.springframework.cloud.netflix.eureka.EurekaClientAutoConfiguration#eurekaInstanceConfigBean()` like this:

```
@Configuration
@EnableConfigurationProperties
@Profile("docker")
public class EurekaClientConfig {

    private ConfigurableEnvironment env;
    private RelaxedPropertyResolver propertyResolver;

    public EurekaClientConfig(final ConfigurableEnvironment env) {
        this.env = env;
        this.propertyResolver = new RelaxedPropertyResolver(env);
    }

    @Bean
    @Primary
    public EurekaInstanceConfigBean eurekaInstanceConfigBean(final InetUtils inetUtils) th
        final String hostName = System.getenv("HOSTNAME");
        String hostAddress = null;

        final Enumeration<NetworkInterface> networkInterfaces = NetworkInterface.getNetwor
        for (NetworkInterface netInt : Collections.list(networkInterfaces)) {
            for (InetAddress inetAddress : Collections.list(netInt.getInetAddresses())) {
                if (hostName.equals(inetAddress.getHostName())) {
                    hostAddress = inetAddress.getHostAddress();
                }

                System.out.printf("Inet %s: %s / %s\n", netInt.getName(),  inetAddress.getI
            }
        }
        if (hostAddress == null) {
            throw new UnknownHostException("Cannot find ip address for hostname: " + hostNa
        }

        final int nonSecurePort = Integer.valueOf(propertyResolver.getProperty("server.por

        final EurekaInstanceConfigBean instance = new EurekaInstanceConfigBean(inetUtils);
        instance.setHostname(hostName);
        instance.setIpAddress(hostAddress);
        instance.setNonSecurePort(nonSecurePort);
        System.out.println(instance);
        return instance;
    }
}
```

And now I got next Eureka client registration:

```
<application>
  <name>CONFIG</name>
  <instance>
    <instanceId>config:a1b85f942f8d075984ff6ca490b476b1</instanceId>
    <hostName>10.0.0.5</hostName>
    <app>CONFIG</app>
    <ipAddr>10.0.0.5</ipAddr>
    <status>UP</status>
    <overriddenstatus>UNKNOWN</overriddenstatus>
    <port enabled="true">8888</port>
    <securePort enabled="false">443</securePort>
    <countryId>1</countryId>
    <dataCenterInfo class="com.netflix.appinfo.InstanceInfo$DefaultDataCenterInfo">
      <name>MyOwn</name>
    </dataCenterInfo>
    <leaseInfo>
      <renewalIntervalInSecs>30</renewalIntervalInSecs>
```

```
      <durationInSecs>90</durationInSecs>
      <registrationTimestamp>1511430156036</registrationTimestamp>
      <lastRenewalTimestamp>1511430252197</lastRenewalTimestamp>
      <evictionTimestamp>0</evictionTimestamp>
      <serviceUpTimestamp>1511430131155</serviceUpTimestamp>
    </leaseInfo>
    <metadata class="java.util.Collections$EmptyMap"/>
    <homePageUrl>http://10.0.0.5:8888/</homePageUrl>
    <statusPageUrl>http://10.0.0.5:8888/info</statusPageUrl>
    <healthCheckUrl>http://10.0.0.5:8888/health</healthCheckUrl>
    <vipAddress>config</vipAddress>
    <secureVipAddress>config</secureVipAddress>
    <isCoordinatingDiscoveryServer>false</isCoordinatingDiscoveryServer>
    <lastUpdatedTimestamp>1511430156036</lastUpdatedTimestamp>
    <lastDirtyTimestamp>1511430100452</lastDirtyTimestamp>
    <actionType>ADDED</actionType>
  </instance>
</application>
```


And from container:

```
$ hostname
7aa5c788c463
$ hostname -i
10.0.0.5
```

It is exactly what I want ☕
I need to test docker swarm scaling with replicas and my problem looks solved.

UPDATE6:

Finish 🥇
I got this works. Docker swarm service replications and Eureka are friends now.

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
AUTH	n/a (1)	(1)	UP (1) - auth:06abc313d13cb3f5d3bd15df81a2ec0d
CONFIG	n/a (1)	(1)	UP (1) - config:92bc558cb2f8e9071b29b10de2bdb07d
GATEWAY	n/a (1)	(1)	UP (1) - gateway:52bf1adec8146486239819706d2b97fa
MONITOR	n/a (1)	(1)	UP (1) - monitor:6b79ad31f42fa49a0d2be2dc09de0e9d
REGISTRY	n/a (1)	(1)	UP (1) - registry:d2e9fe38b4ba2787232164c8d9c38cac
TEMPLATE-SERVICE-1	n/a (2)	(2)	UP (2) - template-service-1:431109dab3de8f77c471dc2719ba9c28 , template-service-1:2bef4192a3106139a306e9ad07fdeac2
TEMPLATE-SERVICE-2	n/a (3)	(3)	UP (3) - template-service-2:4d066c17e11121460efb29fc82facd34 , template-service-2:8f4e5399cb6c20d0384c1ac4327bf8da , template-service-2:96290f047823d00e46175f55b90c9cb2


```
<application>
  <name>TEMPLATE-SERVICE-1</name>
  <instance>
    <instanceId>template-service-1:431109dab3de8f77c471dc2719ba9c28</instanceId>
    <hostName>10.0.0.3</hostName>
    <app>TEMPLATE-SERVICE-1</app>
    <ipAddr>10.0.0.3</ipAddr>
    <status>UP</status>
    <overriddenstatus>UNKNOWN</overriddenstatus>
    <port enabled="true">8080</port>
    <securePort enabled="false">443</securePort>
    <countryId>1</countryId>
    <dataCenterInfo class="com.netflix.appinfo.InstanceInfo$DefaultDataCenterInfo">
      <name>MyOwn</name>
```

```
      </dataCenterInfo>
      <leaseInfo>
        <renewalIntervalInSecs>30</renewalIntervalInSecs>
        <durationInSecs>90</durationInSecs>
        <registrationTimestamp>1511433565078</registrationTimestamp>
        <lastRenewalTimestamp>1511434508066</lastRenewalTimestamp>
        <evictionTimestamp>0</evictionTimestamp>
        <serviceUpTimestamp>1511433517989</serviceUpTimestamp>
      </leaseInfo>
      <metadata class="java.util.Collections$EmptyMap"/>
      <homePageUrl>http://10.0.0.3:8080/</homePageUrl>
      <statusPageUrl>http://10.0.0.3:8080/info</statusPageUrl>
      <healthCheckUrl>http://10.0.0.3:8080/health</healthCheckUrl>
      <vipAddress>template-service-1</vipAddress>
      <secureVipAddress>template-service-1</secureVipAddress>
      <isCoordinatingDiscoveryServer>false</isCoordinatingDiscoveryServer>
      <lastUpdatedTimestamp>1511433565078</lastUpdatedTimestamp>
      <lastDirtyTimestamp>1511433517697</lastDirtyTimestamp>
      <actionType>ADDED</actionType>
    </instance>
    <instance>
      <instanceId>template-service-1:2bef4192a3106139a306e9ad07fdeac2</instanceId>
      <hostName>10.0.0.18</hostName>
      <app>TEMPLATE-SERVICE-1</app>
      <ipAddr>10.0.0.18</ipAddr>
      <status>UP</status>
      <overriddenstatus>UNKNOWN</overriddenstatus>
      <port enabled="true">8080</port>
      <securePort enabled="false">443</securePort>
      <countryId>1</countryId>
      <dataCenterInfo class="com.netflix.appinfo.InstanceInfo$DefaultDataCenterInfo">
        <name>MyOwn</name>
      </dataCenterInfo>
      <leaseInfo>
        <renewalIntervalInSecs>30</renewalIntervalInSecs>
        <durationInSecs>90</durationInSecs>
        <registrationTimestamp>1511434087896</registrationTimestamp>
        <lastRenewalTimestamp>1511434508066</lastRenewalTimestamp>
        <evictionTimestamp>0</evictionTimestamp>
        <serviceUpTimestamp>1511434087833</serviceUpTimestamp>
      </leaseInfo>
      <metadata class="java.util.Collections$EmptyMap"/>
      <homePageUrl>http://10.0.0.18:8080/</homePageUrl>
      <statusPageUrl>http://10.0.0.18:8080/info</statusPageUrl>
      <healthCheckUrl>http://10.0.0.18:8080/health</healthCheckUrl>
      <vipAddress>template-service-1</vipAddress>
      <secureVipAddress>template-service-1</secureVipAddress>
      <isCoordinatingDiscoveryServer>false</isCoordinatingDiscoveryServer>
      <lastUpdatedTimestamp>1511434087896</lastUpdatedTimestamp>
      <lastDirtyTimestamp>1511434087766</lastDirtyTimestamp>
      <actionType>ADDED</actionType>
    </instance>
  </application>

  <application>
    <name>TEMPLATE-SERVICE-2</name>
    <instance>
      <instanceId>template-service-2:4d066c17e11121460efb29fc82facd34</instanceId>
      <hostName>10.0.0.5</hostName>
      <app>TEMPLATE-SERVICE-2</app>
      <ipAddr>10.0.0.5</ipAddr>
      <status>UP</status>
      <overriddenstatus>UNKNOWN</overriddenstatus>
      <port enabled="true">8080</port>
      <securePort enabled="false">443</securePort>
      <countryId>1</countryId>
      <dataCenterInfo class="com.netflix.appinfo.InstanceInfo$DefaultDataCenterInfo">
        <name>MyOwn</name>
      </dataCenterInfo>
      <leaseInfo>
        <renewalIntervalInSecs>30</renewalIntervalInSecs>
        <durationInSecs>90</durationInSecs>
        <registrationTimestamp>1511433565079</registrationTimestamp>
        <lastRenewalTimestamp>1511434506703</lastRenewalTimestamp>
        <evictionTimestamp>0</evictionTimestamp>
        <serviceUpTimestamp>1511433517053</serviceUpTimestamp>
      </leaseInfo>
      <metadata class="java.util.Collections$EmptyMap"/>
      <homePageUrl>http://10.0.0.5:8080/</homePageUrl>
      <statusPageUrl>http://10.0.0.5:8080/info</statusPageUrl>
      <healthCheckUrl>http://10.0.0.5:8080/health</healthCheckUrl>
      <vipAddress>template-service-2</vipAddress>
      <secureVipAddress>template-service-2</secureVipAddress>
      <isCoordinatingDiscoveryServer>false</isCoordinatingDiscoveryServer>
```

```
    <lastUpdatedTimestamp>1511433565079</lastUpdatedTimestamp>
    <lastDirtyTimestamp>1511433516707</lastDirtyTimestamp>
    <actionType>ADDED</actionType>
  </instance>
  <instance>
    <instanceId>template-service-2:8f4e5399cb6c20d0384c1ac4327bf8da</instanceId>
    <hostName>10.0.0.20</hostName>
    <app>TEMPLATE-SERVICE-2</app>
    <ipAddr>10.0.0.20</ipAddr>
    <status>UP</status>
    <overriddenstatus>UNKNOWN</overriddenstatus>
    <port enabled="true">8080</port>
    <securePort enabled="false">443</securePort>
    <countryId>1</countryId>
    <dataCenterInfo class="com.netflix.appinfo.InstanceInfo$DefaultDataCenterInfo">
      <name>MyOwn</name>
    </dataCenterInfo>
    <leaseInfo>
      <renewalIntervalInSecs>30</renewalIntervalInSecs>
      <durationInSecs>90</durationInSecs>
      <registrationTimestamp>1511434220315</registrationTimestamp>
      <lastRenewalTimestamp>1511434519656</lastRenewalTimestamp>
      <evictionTimestamp>0</evictionTimestamp>
      <serviceUpTimestamp>1511434219932</serviceUpTimestamp>
    </leaseInfo>
    <metadata class="java.util.Collections$EmptyMap"/>
    <homePageUrl>http://10.0.0.20:8080/</homePageUrl>
    <statusPageUrl>http://10.0.0.20:8080/info</statusPageUrl>
    <healthCheckUrl>http://10.0.0.20:8080/health</healthCheckUrl>
    <vipAddress>template-service-2</vipAddress>
    <secureVipAddress>template-service-2</secureVipAddress>
    <isCoordinatingDiscoveryServer>false</isCoordinatingDiscoveryServer>
    <lastUpdatedTimestamp>1511434220315</lastUpdatedTimestamp>
    <lastDirtyTimestamp>1511434219577</lastDirtyTimestamp>
    <actionType>ADDED</actionType>
  </instance>
  <instance>
    <instanceId>template-service-2:96290f047823d00e46175f55b90c9cb2</instanceId>
    <hostName>10.0.0.19</hostName>
    <app>TEMPLATE-SERVICE-2</app>
    <ipAddr>10.0.0.19</ipAddr>
    <status>UP</status>
    <overriddenstatus>UNKNOWN</overriddenstatus>
    <port enabled="true">8080</port>
    <securePort enabled="false">443</securePort>
    <countryId>1</countryId>
    <dataCenterInfo class="com.netflix.appinfo.InstanceInfo$DefaultDataCenterInfo">
      <name>MyOwn</name>
    </dataCenterInfo>
    <leaseInfo>
      <renewalIntervalInSecs>30</renewalIntervalInSecs>
      <durationInSecs>90</durationInSecs>
      <registrationTimestamp>1511434221171</registrationTimestamp>
      <lastRenewalTimestamp>1511434520412</lastRenewalTimestamp>
      <evictionTimestamp>0</evictionTimestamp>
      <serviceUpTimestamp>1511434220659</serviceUpTimestamp>
    </leaseInfo>
    <metadata class="java.util.Collections$EmptyMap"/>
    <homePageUrl>http://10.0.0.19:8080/</homePageUrl>
    <statusPageUrl>http://10.0.0.19:8080/info</statusPageUrl>
    <healthCheckUrl>http://10.0.0.19:8080/health</healthCheckUrl>
    <vipAddress>template-service-2</vipAddress>
    <secureVipAddress>template-service-2</secureVipAddress>
    <isCoordinatingDiscoveryServer>false</isCoordinatingDiscoveryServer>
    <lastUpdatedTimestamp>1511434221171</lastUpdatedTimestamp>
    <lastDirtyTimestamp>1511434220463</lastDirtyTimestamp>
    <actionType>ADDED</actionType>
  </instance>
</application>
```

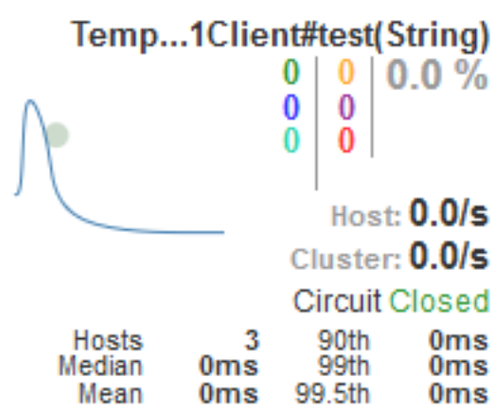


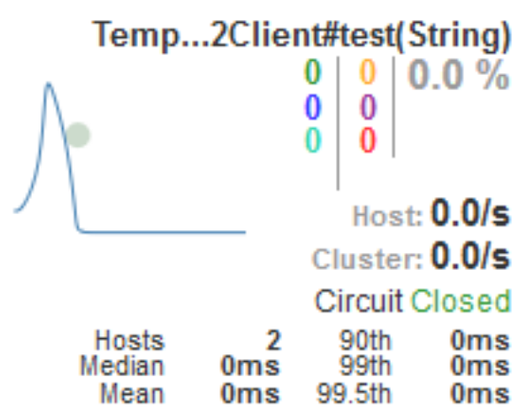
```
[
    {
        "Name": "stack_default",
        "Id": "vyei6hxull9le7ajma5n6r8gf",
        "Created": "2017-11-23T10:34:18.6336376Z",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
```

```
                {
                    "Subnet": "10.0.0.0/24",
                    "Gateway": "10.0.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "189f0ae2a2924ac45d844c957d9668030381f51a522c4d782c9c6806fc1d968b": {
                "Name": "stack_rabbitmq.1.lrpow8l5lxy051jsl1nhcxaje",
                "EndpointID": "990376d02a606c1a8bd7a61018321c6a99586245e0b233a40ba60d2877d
                "MacAddress": "02:42:0a:00:00:07",
                "IPv4Address": "10.0.0.7/24",
                "IPv6Address": ""
            },
            "1fef95280748ec36508c7b6d52a156f1332476211a9d7f5fb22e4d5e1f9e21d8": {
                "Name": "stack_template-service-2.3.2anjsglhw44jpxp47relufvv1",
                "EndpointID": "aa20fdd4d8cb783bb6eda8f191f4039f33464f94a43fec6f1a985514c28
                "MacAddress": "02:42:0a:00:00:14",
                "IPv4Address": "10.0.0.20/24",
                "IPv6Address": ""
            },
            "2a3af4cee6c8b14143c2408358aaebf0251fe3525fd6fc3a97c755518848c576": {
                "Name": "stack_monitor.1.rczns5q61fmflmwkmm2npl3iz",
                "EndpointID": "e70f108a4fff31a1445387bd549708b2e4a8a30d30e8173a3a5e083a61d
                "MacAddress": "02:42:0a:00:00:11",
                "IPv4Address": "10.0.0.17/24",
                "IPv6Address": ""
            },
            "6d06a459037fbed038b978353f4bd7838dcf049da35594a325cad6821aac7400": {
                "Name": "stack_registry.1.yecxlmhaezmnt2k76gzuw26jm",
                "EndpointID": "e6ee97f92c281b8a5cc64d800a474522c44daa5869dd5e01b0dce4746d5
                "MacAddress": "02:42:0a:00:00:0b",
                "IPv4Address": "10.0.0.11/24",
                "IPv6Address": ""
            },
            "a26eaa9094bef35a237d732c48f1771103f4874b62453a2bf98d7ba2da6b7653": {
                "Name": "stack_template-service-1.1.qpqya9b2ld8f5fl6if6evit0j",
                "EndpointID": "bcf7b2e685ee471652056963f9eb0d9e351ba969b14f813705a34c96e22
                "MacAddress": "02:42:0a:00:00:03",
                "IPv4Address": "10.0.0.3/24",
                "IPv6Address": ""
            },
            "aae4495d4012f9321fd1d69af5da3708b201e9bf95dea387a26d9865984cd0f7": {
                "Name": "stack_template-service-2.1.s6nvdc8cywj766t6h3wxssaam",
                "EndpointID": "48fb0ea7e8ad706a2f2a14c08b6ea1c16dabbb5664e86c449ed447d725c
                "MacAddress": "02:42:0a:00:00:05",
                "IPv4Address": "10.0.0.5/24",
                "IPv6Address": ""
            },
            "b84a6706b89d0d2ee2784385849ba3b7d094391cb8e6e445464f0b6f6e12e1a0": {
                "Name": "stack_config.1.6q03r8grjvq05ak9g0jgga3zg",
                "EndpointID": "c07ff9a1f5d5c302a0ca65104515994c004c4dbe1761ac12bc29a17dded
                "MacAddress": "02:42:0a:00:00:09",
                "IPv4Address": "10.0.0.9/24",
                "IPv6Address": ""
            },
            "cec008adc2e31257ab73ff495c709973e90ee54175c63947d0cf2b3ceec2734f": {
                "Name": "stack_auth.1.e3rw5edvumgzf3kq5078yvwsn",
                "EndpointID": "e79e03c37672cff86e5647e389bdcb50fc517f42d1d7561938743126e99
                "MacAddress": "02:42:0a:00:00:0d",
                "IPv4Address": "10.0.0.13/24",
                "IPv6Address": ""
            },
            "d28b064353ba9776c0703eda529db61d47c1fb065e0494533fbfea7850e35544": {
                "Name": "stack_template-service-2.2.trxb2kaxg8uqvosf4sp887hcu",
                "EndpointID": "80ec6a0b57bdbec7b43b5e0784347292b9b5096384a0b49adbb47b9786b
                "MacAddress": "02:42:0a:00:00:13",
                "IPv4Address": "10.0.0.19/24",
                "IPv6Address": ""
            },
            "e3d0db794f525828b530f64900365bac9ff06bbd30cccd135556d0c8521c610b": {
                "Name": "stack_template-service-1.2.vl038gbz29jrralvhkp80ed7j",
                "EndpointID": "44d6fb2306db9ab936efbd7fb2c84e5691cf1ed4e845d178cff28c4db3e
                "MacAddress": "02:42:0a:00:00:12",
                "IPv4Address": "10.0.0.18/24",
                "IPv6Address": ""
            },
```

```
            "eec959f70e20d3c1ee4ee709ba6034d2d69a950628722c7d1435c3392d58ccaa": {
                "Name": "stack_gateway.1.r6expujs6l8q26htayzxc49ut",
                "EndpointID": "de7b895408d84eca063ee7ce032ac372ddbfc1e6d07881fc150e9935184
                "MacAddress": "02:42:0a:00:00:0f",
                "IPv4Address": "10.0.0.15/24",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.driver.overlay.vxlanid_list": "4097"
        },
        "Labels": {
            "com.docker.stack.namespace": "stack"
        },
        "Peers": [
            {
                "Name": "moby-e6a48a47d001",
                "IP": "192.168.65.2"
            }
        ]
    }
]
```

Even Hystrix Dashboard understands replicas count via `Feign` commucations:





At this moment I think I solved my problem with Spring Cloud + Docker Swarm. If not, I let you know 😄

binakot commented 12 days ago • edited

Also if u don't wanna fight with Eureka, client-side load balancing and other things in Docker Swarm, it can work with other discovery backends ☺️ https://docs.docker.com/swarm/discovery

meatfly commented 7 days ago • edited

thank you **@binakot**
for you soluion. I have one extension, if you want to use eureka in HA and then provide eureka client all overlay ipaddress you can get it from dns request tasks. {{SwarmEurekaServiceName}}. Eureka in HA need eureka.client.defaultZone.serviceUrl of another replica, so this can be done like this:

```
@Bean
    @Primary
    EurekaClientConfigBean eurekaClientConfigBean() {
        EurekaClientConfigBean client = new EurekaClientConfigBean();
        if ("bootstrap".equals(propertyResolver.getProperty("spring.config.name"))) {
            // We don't register during bootstrap by default, but there will be another
            // chance later.
            client.setRegisterWithEureka(false);
        } else {
            String serviceName = propertyResolver.getProperty("eureka.client.swarmServiceNa
            String eurekaPort = propertyResolver.getProperty("eureka.client.servicePort");
            String myHostAddress = findHostAddress();

            List<String> eurekaIps = getDnsIps("tasks." + serviceName);
            //remove my ip maybe Iam eureka app
            eurekaIps.remove(myHostAddress);

            Map<String, String> eurekaServiceUrls = new HashMap<>();
            for (String eurekaServiceUrl : eurekaIps) {
                String url = "http://" + eurekaServiceUrl + ":" + eurekaPort + "/eureka/";
                eurekaServiceUrls.put("defaultZone", url);
                log.info("eureka service url" + url);
            }
            client.setServiceUrl(eurekaServiceUrls);
        }
```

```
        return client;
    }

    private static List<String> getDnsIps(String host) {
        List<String> ips = new ArrayList<>();
        try {
            for (InetAddress inetAddress : InetAddress.getAllByName(host)) {
                String hostAddress = inetAddress.getHostAddress();
                ips.add(hostAddress);
            }
        } catch (UnknownHostException e) {
            throw new RuntimeException("error", e);
        }
        return ips;
    }

    private static String getHostName() {
        try {
            return InetAddress.getLocalHost().getHostName();
        } catch (UnknownHostException e) {
            e.printStackTrace();
            throw new RuntimeException("error", e);
        }
    }

    private String findHostAddress() {
        String hostAddress = null;
        String hostName = getHostName();

        try {
            final Enumeration<NetworkInterface> networkInterfaces;
            networkInterfaces = NetworkInterface.getNetworkInterfaces();
            for (NetworkInterface netInt : Collections.list(networkInterfaces)) {
                for (InetAddress inetAddress : Collections.list(netInt.getInetAddresses())
                    if (hostName.equals(inetAddress.getHostName())) {
                        hostAddress = inetAddress.getHostAddress();
                    }
                    log.info("inet {} : {} / {}", netInt.getName(), inetAddress.getHostNam
                }
            }
            if (hostAddress == null) {
                throw new IllegalStateException("Cannot find ip address for hostname: " + 
            }
        } catch (SocketException e) {
            e.printStackTrace();
            throw new RuntimeException(e);
        }
        return hostAddress;
    }
```

eureka.client.servicePort and eureka.client.swarmServiceName must be set

binakot commented 6 days ago

@meatfly Thank you 👍. You are right. At this moment I'm using Eureka with standalone mode in testing environment. The HA Eureka configuration will be my next step for the stage/production environment. I'll try to use your code snippets. When it will work, I write here my solution and experience.