# 搭建环境

微服务需要用到配置中心，配置中心依赖配置文件库，这里直接采用的Githup，如果想搭建本地Git，可以用docker安装GitLab。

POC环境与实际环境有区别，Docker的实际环境中，需要制定网络策略和DNS  
服务器，按照实际环境配置host和port。Docker默认采用桥接网络，通常情况下，一个镜像都运行在单独的虚拟机内，所以在镜像内访问其他docker镜像，是不能用localhost的，这时需要配置DNS Server设置hostname访问，或者使用LAN网络访问。

## Docker

POC环境搭建在Docker上，从官方网站下载Docker，本文主要讲述Microservice，对于其他产品或技术不做描述。

常用命令如下，具体命令请查看官方文档：

docker search consul 搜索关键字镜像

docker pull consul 下载镜像

docker ps –a 查看实例

docker stop consul 停止运行

docker run –d –p 8500:8500 –name consul consul –server –bootstrap 启动

docker rm consul 移除实例

docker exec –it consul /bin/bash 打开镜像实例命令行

docker rmi consul 移除镜像

要想移除镜像得先停止实例，再删除实例，最后删除镜像。

Maven项目打包到远程docker镜像，需设置环境变量或者JVM参数：

DOCKER\_HOST=tcp://192.168.1.101:2375

Maven集成插件：

<groupId>com.spotify</groupId>  
<artifactId>dockerfile-maven-plugin</artifactId>

Maven打包命令

mvn clean install -DskipTests dockerfile:build

运行docker镜像

docker run -d -p 35001:35001 -t springcloud/configuration-server

## Mysql

MySql这里作为zipkin的storage，所以要安装mysql。

docker search mysql

docker pull mysql

docker run --name mysql -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=123456 -d mysql

执行sql，这个sql按照zipkin的版本不同加载不同，请选择兼容版本。

|  |
| --- |
| --  -- After install mysql , running this sql file to create zipkin db.  --  create database zipkin default charset utf8 COLLATE utf8\_general\_ci;  use zipkin;  CREATE TABLE IF NOT EXISTS zipkin\_spans (  `trace\_id\_high` BIGINT NOT NULL DEFAULT 0 COMMENT 'If non zero, this means the trace uses 128 bit traceIds instead of 64 bit',  `trace\_id` BIGINT NOT NULL,  `id` BIGINT NOT NULL,  `name` VARCHAR(255) NOT NULL,  `parent\_id` BIGINT,  `debug` BIT(1),  `start\_ts` BIGINT COMMENT 'Span.timestamp(): epoch micros used for endTs query and to implement TTL',  `duration` BIGINT COMMENT 'Span.duration(): micros used for minDuration and maxDuration query'  ) ENGINE=InnoDB ROW\_FORMAT=COMPRESSED CHARACTER SET=utf8 COLLATE utf8\_general\_ci;  ALTER TABLE zipkin\_spans ADD UNIQUE KEY(`trace\_id\_high`, `trace\_id`, `id`) COMMENT 'ignore insert on duplicate';  ALTER TABLE zipkin\_spans ADD INDEX(`trace\_id\_high`, `trace\_id`, `id`) COMMENT 'for joining with zipkin\_annotations';  ALTER TABLE zipkin\_spans ADD INDEX(`trace\_id\_high`, `trace\_id`) COMMENT 'for getTracesByIds';  ALTER TABLE zipkin\_spans ADD INDEX(`name`) COMMENT 'for getTraces and getSpanNames';  ALTER TABLE zipkin\_spans ADD INDEX(`start\_ts`) COMMENT 'for getTraces ordering and range';  CREATE TABLE IF NOT EXISTS zipkin\_annotations (  `trace\_id\_high` BIGINT NOT NULL DEFAULT 0 COMMENT 'If non zero, this means the trace uses 128 bit traceIds instead of 64 bit',  `trace\_id` BIGINT NOT NULL COMMENT 'coincides with zipkin\_spans.trace\_id',  `span\_id` BIGINT NOT NULL COMMENT 'coincides with zipkin\_spans.id',  `a\_key` VARCHAR(255) NOT NULL COMMENT 'BinaryAnnotation.key or Annotation.value if type == -1',  `a\_value` BLOB COMMENT 'BinaryAnnotation.value(), which must be smaller than 64KB',  `a\_type` INT NOT NULL COMMENT 'BinaryAnnotation.type() or -1 if Annotation',  `a\_timestamp` BIGINT COMMENT 'Used to implement TTL; Annotation.timestamp or zipkin\_spans.timestamp',  `endpoint\_ipv4` INT COMMENT 'Null when Binary/Annotation.endpoint is null',  `endpoint\_ipv6` BINARY(16) COMMENT 'Null when Binary/Annotation.endpoint is null, or no IPv6 address',  `endpoint\_port` SMALLINT COMMENT 'Null when Binary/Annotation.endpoint is null',  `endpoint\_service\_name` VARCHAR(255) COMMENT 'Null when Binary/Annotation.endpoint is null'  ) ENGINE=InnoDB ROW\_FORMAT=COMPRESSED CHARACTER SET=utf8 COLLATE utf8\_general\_ci;  ALTER TABLE zipkin\_annotations ADD UNIQUE KEY(`trace\_id\_high`, `trace\_id`, `span\_id`, `a\_key`, `a\_timestamp`) COMMENT 'Ignore insert on duplicate';  ALTER TABLE zipkin\_annotations ADD INDEX(`trace\_id\_high`, `trace\_id`, `span\_id`) COMMENT 'for joining with zipkin\_spans';  ALTER TABLE zipkin\_annotations ADD INDEX(`trace\_id\_high`, `trace\_id`) COMMENT 'for getTraces/ByIds';  ALTER TABLE zipkin\_annotations ADD INDEX(`endpoint\_service\_name`) COMMENT 'for getTraces and getServiceNames';  ALTER TABLE zipkin\_annotations ADD INDEX(`a\_type`) COMMENT 'for getTraces';  ALTER TABLE zipkin\_annotations ADD INDEX(`a\_key`) COMMENT 'for getTraces';  ALTER TABLE zipkin\_annotations ADD INDEX(`trace\_id`, `span\_id`, `a\_key`) COMMENT 'for dependencies job';  CREATE TABLE IF NOT EXISTS zipkin\_dependencies (  `day` DATE NOT NULL,  `parent` VARCHAR(255) NOT NULL,  `child` VARCHAR(255) NOT NULL,  `call\_count` BIGINT,  `error\_count` BIGINT  ) ENGINE=InnoDB ROW\_FORMAT=COMPRESSED CHARACTER SET=utf8 COLLATE utf8\_general\_ci;  ALTER TABLE zipkin\_dependencies ADD UNIQUE KEY(`day`, `parent`, `child`); |

## Zipkin

下载zipkin镜像

docker search zipkin

docker pull openzipkin/zipkin

这里采用docker-compose命令安装，需进入docker-compose.yml平级目录

docker-compose up -d

docker-compose.yml文件：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | version: '2' | |  |  | |  | services: | |  | # The zipkin process services the UI, and also exposes a POST endpoint that | |  | # instrumentation can send trace data to. Scribe is disabled by default. | |  | zipkin: | |  | image: openzipkin/zipkin | |  | container\_name: zipkin | |  | environment: | |  | - STORAGE\_TYPE=mysql | |  | # Point the zipkin at the storage backend | |  | - MYSQL\_DB=zipkin | |  | - MYSQL\_USER=root | |  | - MYSQL\_PASS=123456 | |  | - MYSQL\_HOST=www.junyee.org | |  | - MYSQL\_TCP\_PORT=3306 | |  | # Uncomment to enable scribe | |  | # - SCRIBE\_ENABLED=true | |  | # Uncomment to enable self-tracing | |  | # - SELF\_TRACING\_ENABLED=true | |  | # Uncomment to enable debug logging | |  | # - JAVA\_OPTS=-Dlogging.level.zipkin=DEBUG -Dlogging.level.zipkin2=DEBUG | |  | ports: | |  | # Port used for the Zipkin UI and HTTP Api | |  | - 9411:9411 | |  | # Uncomment if you set SCRIBE\_ENABLED=true | |  | # - 9410:9410 | |

## RabbitMQ

POC中RabbitMQ用作同步configuration file的bus，下载management版本，因为有UI操作。

docker pull rabbitmq:3.7.4-rc.4-management

docker run -d -p 4369:4396 -p 5671:5671 -p 5672:5672 -p 15671:15671 -p 15672:15672 -p 25672:25672 --name rabbitmq rabbitmq:3.7.4-rc.4-management

对于刷新配置做的操作，默认是不开启的，需要在configuration file内开启该命令：

/actuator/bus-refresh

## Consul

安装镜像

docker search consul

docker install progrium/consul

运行

docker run -d -p 8500:8500 --name consul progrium/consul -server –bootstrap

每次重启时可能会运行不成功，这时需要删除实例，再重新运行。

# Spring Cloud工程

POC中附带最基础的Spring Cloud源码，由于注册中心采用consul，所以无需用Java实现。Zipkin在写此POC文档时，官方zipkin还不支持Spring boot 2.0版本，所以安装的docker镜像。其余包括配置中心，Turbine Hystrix，Welcome Service Consul Client Discovery和API Gateway，基本是最基础的微服务框架了。

## Configuration Center

配置中心常用的是刷新配置，更新本地缓存配置文件访问/actuator/bus-refresh。

## Turbine hystrix

Turbine作为集群断融管理工具，管理了若干个在该工程中配置的服务发现客户端，相关地址如下：

<http://localhost:34001/hystrix> 打开dashboard页面

<http://localhost:34001/turbine.stream?cluster=welcome-service> 访问某服务的断融数据

如果是查看单个已配置断融机制的服务可以单独访问该服务的断融统计接口：

<http://localhost:35001/actuator/hystrix.stream>

## Welcome Service

作为示例代码中的服务发现客户端工程，包含了Ribbon和Feign插件，是实际的业务开发模块，是以后各种单一服务开发的基础Demo工程。

## API Gateway

采用Zuul实现的API Gateway示例工程，作为以后API Gateway的Demo工程。

## 启动顺序

依次启动docker，mysql，zipkin，RabbitMQ和Consul。

所有的工程都依赖Configuration Center，所以最先启动。

再依次启动Welcome Service，Turbine Hystrix和API Gateway。