搭建基于ZIPKIN的数据追踪系统

****1. 配置 Java 环境****

*安装 JDK*

Zipkin 使用 Java8

yum install java-1.8.0-openjdk\* -y

安装完成后，查看是否安装成功：

java -version

[root@192 ~]# java -version

openjdk version "1.8.0\_212"

OpenJDK Runtime Environment (build 1.8.0\_212-b04)

OpenJDK 64-Bit Server VM (build 25.212-b04, mixed mode)

****2. 安装 Zipkin****

新建目录

mkdir -p /data/release/zipkin && cd "$\_"

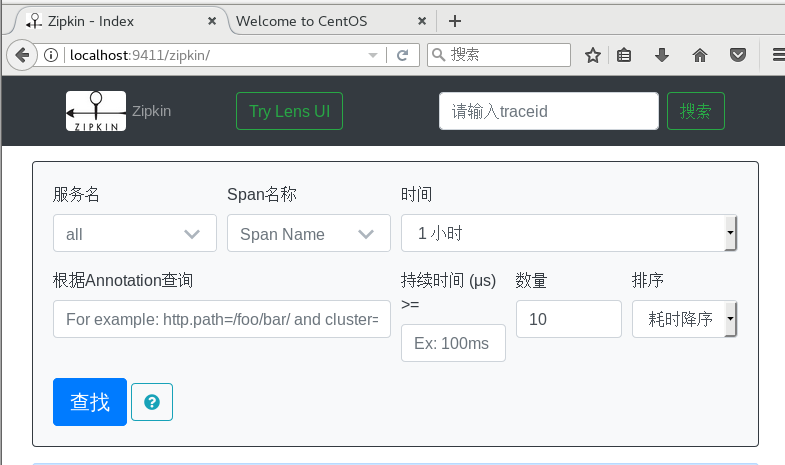
下载 Zipkin

wget -O zipkin.jar 'https://search.maven.org/remote\_content?g=io.zipkin.java&a=zipkin-server&v=LATEST&c=exec'

启动 Zipkin

java -jar zipkin.jar

Zipkin 默认监听 9411 端口， 使用浏览器访问 [http://yourip:9411](http://yourip:9411/" \t "https://blog.csdn.net/qq_26954773/article/details/_blank) 或 [http://yourdomain.com:9411](http://yourdomain.com:9411/" \t "https://blog.csdn.net/qq_26954773/article/details/_blank) 即可看到 Zipkin 自带的图形化界面



****3. 配置 MySQL 数据持久化方案****

Zipkin 支持的持久化方案很多，如： Cassandra, MySQL, Elasticsearch。本实验使用 MySQL 5.7 作为数据持久化方案。

*安装 MySQL 5.7*

使用 Ctrl + C 退出上个步骤的 Java 进程并下载 rmp 包

wget <http://dev.mysql.com/get/mysql57-community-release-el7-9.noarch.rpm>

安装 rpm 包

rpm -Uvh mysql57-community-release-el7-9.noarch.rpm

安装 MySQL

yum install mysql-community-server -y

启动 MySQL 服务

systemctl start mysqld.service

*设置 MySQL 密码*

获取 root 临时密码

grep 'temporary password' /var/log/mysqld.log | awk '{print $NF}'

[root@192 ~]# grep 'temporary password' /var/log/mysqld.log | awk '{print $NF}'

qdegjkHtE7/1

使用上一步的获得的临时密码登入 MySQL

mysql -uroot -p

[root@192 ~]# mysql -uroot -pqdegjkHtE7/1

设置 MySQL 账户 root 密码

ALTER USER 'root'@'localhost' IDENTIFIED BY 'Xx$Zipkin2017';

退出 MySQL, 回到 Bash shell

exit;

*初始化 Zipkin 数据库*

编写初始化脚本

请在 /data/release/zipkin 目录下创建 zipkin\_init.sql，参考下面的内容。

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

) ENGINE=InnoDB ROW\_FORMAT=COMPRESSED CHARACTER SET=utf8 COLLATE utf8\_general\_ci;

ALTER TABLE zipkin\_dependencies ADD UNIQUE KEY(`day`, `parent`, `child`);

登录 Mysql

mysql -u root --password='Xx$Zipkin2017'

创建 Zipkin 数据库

create database zipkin;

切换数据库

use zipkin;

初始化表及索引

source /data/release/zipkin/zipkin\_init.sql

执行以下命令会看到zipkin\_annotations, zipkin\_dependencies, zipkin\_spans 三张数据表，说明初始化成功了

mysql> show tables;

+---------------------+

| Tables\_in\_zipkin |

+---------------------+

| zipkin\_annotations |

| zipkin\_dependencies |

| zipkin\_spans |

+---------------------+

3 rows in set (0.00 sec)

退出 MySQL, 回到 Bash shell

Exit

*启动 Zipkin*

注： 此处x默认使用教程生成的密码

cd /data/release/zipkin

STORAGE\_TYPE=mysql MYSQL\_HOST=localhost MYSQL\_TCP\_PORT=3306 MYSQL\_DB=zipkin MYSQL\_USER=root MYSQL\_PASS='Xx$Zipkin2017' \

nohup java -jar zipkin.jar &

****4. 创建具有数据上报能力的Demo****

搭建 NodeJS 环境

curl --silent --location https://rpm.nodesource.com/setup\_8.x | sudo bash -

yum install nodejs -y

*创建Demo目录*

创建 /data/release/service\_a 目录

mkdir -p /data/release/service\_a && cd "$\_"

*使用 NPM 安装相关依赖*

请在 /data/release/service\_a 目录下创建并编辑 package.json，参考下面的内容。

{

"name": "service\_a",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {},

"author": "",

"license": "ISC",

"dependencies": {

"express": "^4.15.3",

"zipkin": "^0.7.2",

"zipkin-instrumentation-express": "^0.7.2",

"zipkin-transport-http": "^0.7.2"

}

}

安装相关依赖

npm install

*创建并编辑 app.js*

请在 /data/release/service\_a 目录下创建 app.js，参考下面的内容。

const express = require('express');

const {Tracer, ExplicitContext, BatchRecorder} = require('zipkin');

const {HttpLogger} = require('zipkin-transport-http');

const zipkinMiddleware = require('zipkin-instrumentation-express').expressMiddleware;

const ctxImpl = new ExplicitContext();

const recorder = new BatchRecorder({

logger: new HttpLogger( {

endpoint: 'http://127.0.0.1:9411/api/v1/spans'

})

});

const tracer = new Tracer({ctxImpl, recorder});

const app = express();

app.use(zipkinMiddleware({

tracer,

serviceName: 'service-a'

}));

app.use('/', (req, res, next) => {

res.send('hello world');

});

app.listen(3000, () => {

console.log('service-a listening on port 3000!')

});

*启动服务*

*node app.js*

该服务监听 3000 端口， 使用浏览器访问 [http://yourip:3000](http://yourip:3000/) 或 [http://yourdomain.com:3000](http://yourdomain.com:3000/)后，看到“hello world” 的文本字样说明服务已经正常工作

****5. 部署完成****